UNIVERSIDAD DE OVIEDO

ESCUELA DE INGENIERÍA INFORMÁTICA

TRABAJO FIN DE MÁSTER

SIGICCS: Sistema Integrado para la Gestión de Incidencias, Clientes, Contratos y Servicios vía Web

DIRECTORA: Dra. Ana Belén Martínez Prieto

AUTOR: David Cabañeros Blanco

Vº Bº de la Directora del Proyecto

Agradecimientos

A mis padres, por darme la oportunidad y acompañarme en esta etapa que termina con estas líneas. Mamá, te prometo que a partir de ahora iré más a menudo por casa

A mi hermano Jose Antonio, inmerso actualmente en un mar de clases y métodos, quien ha seguido de cerca esta aventura a lo largo de este año, más concretamente, desde un piso por debajo de la clase del Máster en el Valdés Salas. Fue una experiencia tan insólita como predecible bajar juntos a clase durante este curso. Que este párrafo sirva para inspirarle durante el tiempo que pasará documentando su Trabajo Fin de Grado. Recuerda: ¡esto es easy!

A Melodie. Si tuviera que agradecerla todo lo que ha hecho por mí durante este tiempo, me vería obligado a escribir un tomo anexo. Como voy apurado de tiempo, déjame decirte que sabes tan bien como yo que todo esto no habría sido posible sin tu cariño y apoyo. Todos los fines de semana de trabajo contrarreloj, plazos y prisas han dado su fruto y al final uno se queda siempre con lo bueno de las cosas. Por eso, yo me quedo contigo

A Belén, mi Directora de Proyecto, siempre dispuesta a echar una mano en lo que hiciera falta. Gran profesora y mejor persona

A mis compañeros Borja Garrido y Mario Rey. Sus inquietudes, ideas y ganas de ayudar han sido de gran importancia para este trabajo

Al resto de familia y amigos que no he nombrado, por su interés y apoyo constantes

A todos, ***UN MILLÓN DE GRACIAS***

Oviedo, julio de 2013

Resumen

La aplicación desarrollada en el presente proyecto tiene como objetivo servir de herramienta para la gestión de las incidencias comunicadas por los clientes de EcoComputer[1], persiguiendo una intervención, actuación y resolución lo más rápida y efectiva posible.

Sin embargo, con el objetivo de desarrollar un sistema completo que proporcione la capacidad de gestionar cada uno de los aspectos relacionados con la Gestión de Incidencias, tales como la Gestión de Clientes, Contratos y Servicios, la empresa contará con un sistema integrado, desarrollado completamente a medida. Estas particularidades lo convierten en un proyecto interesante, puesto que el producto del mismo está destinado a un uso en producción en un cliente real, colaborando además en relación directa con éste desde el comienzo.

Se trata de una necesidad surgida del día a día en la actividad de la empresa. Actualmente se hace uso de un conjunto heterogéneo de utilidades, desde gestores de proyectos de código abierto y suites de colaboración, hasta clásicas hojas de cálculo. Esta diversidad de herramientas dificulta y ralentiza el trabajo, y lo que es aún peor, introduce numerosos errores y complicaciones en el proceso que pueden dar lugar a problemas con sus clientes.

El Sistema de Gestión de Incidencias, Clientes, Contratos y Servicios vía Web (SIGICCS), aportará funcionalidad, robustez y simplicidad. Por medio de su interfaz web, los empleados de EcoComputer serán capaces de estar al día de todos aquellos aspectos relevantes en el desempeño de su labor habitual, desde el personal de desarrollo y técnicos hasta el departamento de administración, pasando por los gestores de proyectos y gerentes de la empresa, otorgando a cada uno de ellos un identificador de acceso al sistema, basado en su rol o perfil dentro de la organización.

Para lograr un rendimiento óptimo, el sistema se integrará con los existentes en el cliente. Es por ello que el desarrollo se llevará a cabo con tecnologías Java y será desplegado en un servidor web Apache Tomcat. El sistema de gestión de bases de datos empleado será SQL Server 2005, de Microsoft. Además, la construcción de las interfaces de la aplicación estará basada en el conocido Twitter Bootstrap, lo que otorgará un buen nivel de usabilidad y accesibilidad, a la vez que resultará familiar y amigable para los usuarios.

El sistema completo reposará sobre una Intranet, en la cual coexistirá con otros módulos de diversa funcionalidad, todos los cuales han sido desarrollados en proyectos individuales.

[1] EcoComputer, S.L. es una empresa dedicada a la prestación de soluciones TIC fundada en Avilés (Asturias) en el año 1999. Mantiene relaciones comerciales con clientes de diversas áreas de la península, principalmente en Andalucía, Asturias, Extremadura , Castilla y León y País Vasco.

Palabras Clave

Gestión Empresarial, Incidencias, Clientes, Contratos, Servicios, Aplicación Web, JEE, Struts, Bootstrap

Abstract

The developed application is intended to be used as ticket management appliance for processing the ones notified by EcoComputer[1] clients, in order to provide fast an effective response, intervention and resolution.

Nevertheless, with the goal of developing a complete system that provides de ability to manage each of the aspects related to Ticket Management, such as Clients, Contracts and Services Management, the company will receive an integrated system, fully custom-made designed and developed. These features make it an interesting project, given that its product is intended for production use in a real client, also collaborating in relationship with it from the project beginning.

The project is a need that arises from everyday business activity. Currently, EcoComputer makes use of a heterogeneous set of tools, from open-source project management software and collaboration suites to classic spreadsheets. This variety of mechanisms complicates and draws out employees' activity, and even worse, produces numerous mistakes and complexities in the process that can give rise to problems with customers.

Tickets, Clients, Contracts and Services Management System will provide functionality, robustness and simplicity. Through its web interface, EcoComputer crew will be able to keep abreast of all aspects relevant to the performance of their regular work, from development and technical staff to administration department, including project management team and company managers, giving each of them an ID to access into the system, based on their role or profile within the organization.

In order to achieve optimal performance, the system will be integrated with EcoComputer existing infrastructure. That's why its development will be Java-based and deployed on its Apache Tomcat web server. The employed relational database management system will be Microsoft SQL Server 2005. Furthermore, application's interface building will be based on well-known Twitter Bootstrap, which will give a high-grade of usability, at the same time as will be familiar and friendly for users.

The whole system will rest on an Intranet, in which will coexist with a set of modules of diverse purpose, all of them developed in single projects.

[1] EcoComputer, S.L. is an IT solutions provider founded in Avilés (Asturias) in 1999. The company maintains business relationships with clients from different areas of the country, mainly in Andalucía, Asturias, Extremadura, Castilla y León and País Vasco.

Keywords

Business Management, Tickets, Clients, Contracts, Services, Web Application, JEE, Struts, Bootstrap

Índice General

Índice de Figuras

Capítulo 1. Memoria del Proyecto

1.1 Resumen de la Motivación, Objetivos y Alcance del Proyecto

El proyecto surge de la necesidad de contar con un sistema centralizado para la gestión de las incidencias surgidas a los clientes de Ecocomputer en relación a los servicios prestados por la empresa, reduciendo al mínimo el tiempo necesario para la comunicación, coordinación, procesado y resolución de las mismas. La solución desarrollada en este proyecto pretende servir como herramienta de gestión personalizada, desarrollada en colaboración directa con el cliente para ajustarse completamente a sus necesidades.

El principal objetivo es ofrecer a cada perfil de usuario un panel de gestión desde donde pueda acceder de manera sencilla a aquellas opciones que le resulten útiles para el desempeño de sus tareas. Es importante abordar el enfoque de la aplicación como un sistema modular, basado en perfiles con diferentes permisos en el cual, cada usuario, dispondrá de unas opciones y funciones acordes al rol que desempeña dentro de la empresa. Este planteamiento requiere la implementación de un sistema dedicado para la gestión de usuarios, roles y permisos, el cual se aborda en otro proyecto debido a su complejidad.

Un empleado con un perfil cuyas competencias incluyan la participación en el proceso de tratamiento de incidencias podrá acceder y consultar cuáles le han sido asignadas para su resolución. Tras realizar una actuación, anotará las intervenciones llevadas a cabo, el tiempo empleado, una referencia al albarán de materiales necesarios y actualizará el estado de la incidencia.

Para dotar a la aplicación de mayor funcionalidad y poder definirla como un sistema completo, este proyecto comprenderá el desarrollo de cuatro módulos complementarios entre sí, enumerados a continuación

- Gestor de usuarios (se desarrolla en otro proyecto)
- Gestor de clientes
- Gestor de servicios y tarifas
- Gestor de contratos
- Gestor de incidencias

A modo de resumen de todo lo anterior, se puede concluir que la aplicación desarrollada servirá de soporte a la gestión de los clientes, sus contratos asociados y los servicios que engloban, así como las incidencias comunicadas por los clientes que han suscrito un contrato de mantenimiento con Ecocomputer. El personal de la empresa precisa de un sistema sencillo y fiable para controlar y mantener organizado todo el proceso, desde el momento en el que el cliente se pone en contacto con la empresa hasta que recibe una solución satisfactoria.

1.2 Justificación del Proyecto

El proyecto abarca el desarrollo de una aplicación web que resuelva la problemática de la gestión de incidencias comunicadas por clientes a Ecocomputer. Se pretende desarrollar un sistema centralizado, ágil y fácil de usar para lograr que sea útil a los distintos perfiles que participarán de uno u otro modo, desde el cliente que comunica el problema hasta los técnicos encargados de proporcionar la solución final, pasando por el personal de administración, gestor de proyectos, etc.

Dado que el proyecto se desarrollará en colaboración directa con el cliente, todas las decisiones que sea necesario tomar para el diseño de la aplicación y la elección de diferentes opciones tecnológicas serán acorde a las propias necesidades del negocio.

A pesar de la existencia de soluciones alternativas enfocadas a un fin similar, ninguna de ellas se ajusta completamente a lo deseado. Tras implantar alguna de ellas durante algún tiempo sin conseguir los resultados deseados, se ha terminado optando por una solución personalizada que se ajuste por completo a lo que se necesita, sin elementos superfluos ni complicaciones ni funciones no necesarias que conllevan, además de retrasos y errores en el proceso de atención a los clientes, un alto esfuerzo de adaptación, sobre todo en lo relativo al personal no técnico.

Capítulo 2. Introducción

2.1 Objetivos del Proyecto

El presente proyecto tiene como objetivo principal el desarrollo de una aplicación web para la gestión de las incidencias vinculadas a contratos, con los servicios asociados a estos, que los clientes suscriben con una empresa de servicios informáticos. El siguiente listado enumera el desglose de objetivos que componen el proyecto en su totalidad.

2.1.1 Gestión de Usuarios

Dado que la aplicación será utilizada por diversas categorías de usuarios, es preciso mantenerlos organizados de un modo concreto. Para ello, se define el sistema modular de la aplicación como primer nivel de organización de usuarios. Se pretende otorgar acceso de lectura y/o escritura a cada módulo de forma independiente para cada usuario. En el caso de, por ejemplo, un usuario cuyas funciones sean las de gestionar únicamente la cartera de clientes y sus contratos asociados, no sería preciso que tuviera acceso al módulo de gestión de incidencias.

En el siguiente nivel organizativo se definirá un planteamiento basado en grupos de usuarios, de acuerdo al cargo que ocupa cada uno en la empresa. Algunos ejemplos de usuarios son el gestor de proyectos, grupo de técnicos, personal de administración, grupo de desarrollo, etc.

Como último nivel de organización, se optará por un sistema basado en perfiles. Con ello, cada usuario estará asociado a su perfil correspondiente, pudiendo así controlar qué acciones puede llevar a cabo cada uno. Por ejemplo: un técnico de sistemas podrá tener acceso al módulo de gestión de incidencias, con el objetivo de consultar y modificar aquellas que le sean asignadas, pero no podrá dar de alta un nuevo cliente en el sistema o modificar los contratos que ha suscrito.

Gracias a este planteamiento se conseguirá un alto nivel de granularidad en lo referente a la gestión de los usuarios y las acciones que estos pueden llevar a cabo dentro del sistema. Sin embargo, debido a la complejidad que conlleva dicha propuesta, el desarrollo de esta parte se llevará a cabo en un proyecto individual, por lo que desde el presente se hará uso de dicha implementación, lo cual implicará, previsiblemente, una serie de dificultades relativas al diseño, dependencias, coordinación y posterior integración entre sistemas.

2.1.2 Gestión de Clientes

Será el primer módulo a desarrollar por su simplicidad respecto a los demás, puesto que no implica dependencias directas de ningún otro en lo referente a su implementación.

Servirá esto además como adecuación al entorno y las herramientas, además de permitir establecer el punto de partida y marcar el método para el resto de desarrollo.

Consistirá en implementar las operaciones básicas de creación, actualización, consulta y eliminación de entidades cliente, para lo cual se definirá el conjunto de propiedades que se precisan y el modo en el que se presentará la información. Por último, para implementar de forma completa el módulo será necesario diseñar el esquema de la base de datos e implementarlo.

2.1.3 Gestión de Servicios y Tarifas

De forma análoga al módulo de gestión de clientes, el de servicios y tarifas resulta directamente independiente del resto. Se mantendrá un catálogo de servicios del portafolio de la empresa, con unas tarifas específicas para cada uno de ellos, las cuales serán completamente personalizables.

Es importante advertir que este módulo requerirá mayor grado de dedicación, debido a que conlleva la implementación de distintas restricciones o particularidades del modelo de negocio de la empresa, por lo cual será muy importante el diseño y la implementación de la lógica de negocio, así como la creación y ejecución de un plan de pruebas extensivo. Errores de concepto podrían ocasionar fallos en la implementación que serían fatales para el conjunto de la aplicación, pudiendo incluso hacerla inservible, por lo que será una de las partes en las que más colaboración será necesaria por parte de la empresa, desde la fase de análisis, hasta el diseño, la implementación y las diferentes iteraciones de las pruebas.

2.1.4 Gestión de Contratos

Llegado este punto, los dos módulos anteriores serán plenamente funcionales, lo cual resulta imprescindible puesto que el presente es directamente dependiente de ellos. Por una parte, los contratos deben estar asociados a un cliente existente en el sistema. Por la otra, un contrato estará compuesto por uno o más servicios que el cliente requiere y las respectivas tarifas que la empresa establece.

Será necesario además definir las propiedades específicas de las que se precisa, ajustándose al dominio del problema, como pueden ser periodos de validez, métodos de facturación, etc.

2.1.5 Gestión de Incidencias

Constituirá el núcleo del sistema, dependiendo directamente de los tres módulos ya descritos. Por ello, soportará toda la carga de trabajo generada por las peticiones de soporte formuladas por los clientes que suscriban algún contrato de servicio con la empresa.

En este módulo convergen todos los usuarios de la aplicación, desde el personal encargado de introducir la petición del cliente en el sistema, creando así un ticket de

asistencia. La incidencia será propuesta para su resolución y asignada inicialmente a un empleado y pasará posteriormente por distintas fases hasta su cierre.

De forma agregada a lo anterior, una incidencia no solo conllevará el tratado de sus datos básicos, tales como asignación, descripción, etcétera, sino que precisará de elementos agregados tales como intervenciones, adjuntos y relaciones con otras incidencias. Todos estos elementos darán lugar a un aumento de la complejidad y será esencial diseñar la implementación de forma lógica y estructurada, de forma que no se introduzcan limitaciones que compliquen el desarrollo en las fases más avanzadas del mismo.

Se precisará para ello de un análisis exhaustivo de los requisitos relativos a la lógica del flujo de trabajo que genera el proceso de gestión de una incidencia. Tomando en cuenta la complejidad y la necesidad de exactitud en lo referente a lo que la empresa necesita, será la parte del sistema a la que más tiempo y esfuerzo se dedicará.

2.2 Estudio de la Situación Actual

A continuación se presenta el estudio de las diferentes alternativas encontradas al sistema a tratar en el proyecto. Algunas de ellas son utilizadas en la actualidad por el cliente, empleándose cada una de forma aislada para resolver parcialmente cada tipo de problemática derivada del desempeño de su actividad.

2.2.1 Evaluación de Alternativas

2.2.1.1 Redmine

Actualmente en la versión 2.1.2, Redmine es una herramienta de gestión de proyectos desarrollada en Ruby on Rails y publicada como software libre bajo la licencia GNU v2. Entre sus características se encuentra el soporte a múltiples proyectos simultáneos, control de acceso flexible basado en roles, gestión de documentación, notificaciones por correo electrónico y el soporte a *plugins* desarrollados por terceros que extienden su funcionalidad básica.

Logotipo 1. Redmine

La mayor de sus ventajas no es otra que el soporte con el que cuenta. Redmine es una herramienta ampliamente utilizada por diferentes organismos y empresas, tales como universidades, centros de investigación, gobiernos, etc. Esto hace que la herramienta cuente con una gran comunidad a sus espaldas que se encarga de reportar errores, corregirlos e incluso añadir nuevas funcionalidades.

De hecho, el cliente de este proyecto cuenta hasta ahora con una instancia de Redmine empleada exclusivamente como herramienta de comunicación y colaboración entre el personal del departamento de desarrollo. Esto es debido a que la herramienta no está concebida para el manejo de incidencias con las particularidades del proceso que se lleva a cabo en su gestión.

2.2.1.2 Mantis Bug Tracker

La primera versión de Mantis aparece en el año 2000, publicada bajo la licencia GNU v2. Se trata de un sistema de seguimiento de errores en el desarrollo de software escrita en PHP, utilizando MySQL como sistema de gestión de bases de datos. Tiene la particularidad de permitir dividir un proyecto en subcategorías, por lo que el rastreo de errores es más preciso.

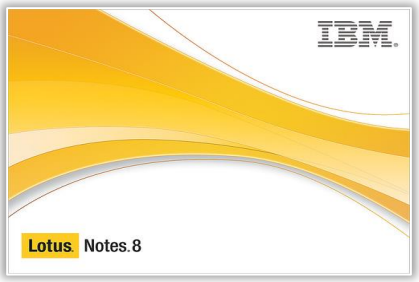
Basado en roles y flujos de trabajo, Mantis permite definir quién puede informar de errores, quién puede analizarlos y quién repararlos.

Logotipo 2. Mantis Bug Tracker

Mantis es uno de los sistemas implantados en la actualidad en Ecocomputer. Se utiliza principalmente como punto de entrada de las incidencias comunicadas por los clientes y como lugar de anotación de las intervenciones y materiales requeridos por los técnicos que realizan las visitas. No obstante, esto ocasiona un sobreesfuerzo al departamento de administración, puesto que se precisa la inspección manual e individual de todas las incidencias que han sido resueltas para conocer el coste de tiempo y materiales necesarios y proceder a la facturación al cliente.

2.2.1.3 Lotus Notes

Aunque en un principio no se intuya la relación de este software con los objetivos de este proyecto, Lotus Notes se utiliza en la actualidad en Ecocomputer, además de como gestor de correo electrónico y gestor documental interno, como repositorio de información con datos de los clientes, tales como datos de contacto y otros detalles particulares.



Logotipo 3. Lotus Notes

A día de hoy, la información contenida en su repositorio local ha quedado obsoleta, lo cual, unido al uso imperativo desde un cliente de escritorio, hace que sea necesario buscar nuevas opciones ante su inminente desecho como solución de gestión.

2.2.1.4 DotProject - ProjectOpen

DotProject es un gestor de proyectos con acceso web implantado en Ecocomputer durante 2008 con escaso éxito, ya que no se ajustaba a las necesidades específicas de la empresa. El principal motivo de su fracaso fue su enfoque a la planificación de tareas y gestión

temporal, prescindiendo además de todas aquellas funciones relativas a la gestión documental.

Logotipo 4. DotProject

Durante el año siguiente, se decidió probar la herramienta ProjectOpen, obteniendo un resultado idéntico al de DotProject, por lo que ambos fueron abandonados.

Logotipo 5. ProjectOpen

2.2.1.5 PANDORA - Gestión de Incidencias Informáticas de la Secretaría de Estado de Administraciones Públicas

Se trata de una Herramienta Web de Gestión de Incidencias Informáticas, con diferentes y configurables niveles de resolución, aspectos particulares para centros de atención a usuarios, gestión de tareas, asociación a inventario informático, notificaciones, conversaciones y estadísticas, desarrollada por la División de Sistemas de Información y Comunicaciones en la Secretaría de Estado de Administraciones Públicas del Ministerio de Hacienda y Administraciones Públicas, teniendo como destinataria a cualquier Administración Pública del Estado.

En cuanto a su descripción técnica, PANDORA es un desarrollo propio cuyo origen parte de la Delegación de Gobierno en Madrid y ha sido evolucionada por la Subdirección General de Tecnologías de la Información y Comunicaciones (SGTIC) del Ministerio de Administraciones Públicas del Gobierno de España. La aplicación está orientada a un entorno Web y las características en que se basa su desarrollo son:

- **Apache HTTPD 2** como servidor web
- **MySQL 5** como sistema de gestión de base de datos
- **PHP 5** para el desarrollo del interfaz de usuario
- **AJAX** para la agilización de partes del interfaz de usuario. Se utiliza la librería Dojo de JavaScript

PANDORA es una aplicación concebida para recopilar de una forma cómoda cualquier problema que necesite de soporte. La herramienta permite la tipificación de dichas incidencias así como grupos de resolución para poder realizar una correcta distribución de las mismas. Está integrada con LDAP (Protocolo Ligero de Acceso a Directorios) de forma que cualquier usuario puede comunicar una incidencia vía Web especificando sólo los datos relativos a la misma, ya que la información del usuario se completa automáticamente.

Las incidencias están distribuidas en tareas, de forma que una incidencia puede pasar por distintos grupos de resolución si fuera necesario, para que cada uno pueda resolver la parte que le compete. Cuando se asigna una tarea a un grupo, sus miembros reciben una notificación por correo electrónico. La incidencia quedará resuelta cuando finalicen todas las tareas de que se componga, notificándose dicha resolución al usuario y a todos los miembros del grupo de resolución. Cada tramitador puede resolver tareas, escalarlas a otro grupo de resolución si fuera necesario, o conversar con el usuario o con el grupo a que pertenece mediante el envío de correo electrónico gestionado desde la aplicación.

Entre sus ventajas destaca la mejora el servicio al evitar la pérdida de información en la resolución de incidencias, la agilización del flujo de tramitación, el análisis del rendimiento del servicio por medio de datos objetivos y el hecho de tratarse de una solución de código abierto, por lo que no tiene coste alguno por licencia.

Logotipo 6. Pandora

No obstante, a pesar del amplio conjunto de posibilidades que ofrece, PANDORA no es una herramienta adecuada a las necesidades del cliente, puesto que carece de funcionalidades esenciales, tales como la gestión de los clientes, contratos que suscriben y servicios que requieren, con sus tarifas asociadas, todo lo cual es una parte importante y de la cual depende la gestión de incidencias en la empresa.

2.2.2 Conclusiones

Tal y como se ha descrito, la empresa cuenta actualmente con varios sistemas de distinta naturaleza para la gestión de las incidencias que recibe. Es por ello que, aunque alguna de las alternativas citadas anteriormente sirven en la actualidad como solución parcial para el día a día, se precisa de un nuevo sistema que englobe todas las funciones necesarias y sirva como gestor del flujo de trabajo que conlleva la entrada de una incidencia para todo el personal involucrado en el proceso.

Es importante recordar que el objetivo del proyecto es proporcionar una herramienta con una serie de características concretas, adaptadas a las necesidades de una empresa en

particular, por lo que resultaría muy complicado encontrar un producto desarrollado con una funcionalidad completamente acorde a lo que se requiere.

Capítulo 3. Aspectos Teóricos

La presente sección tiene como objetivo enumerar y describir someramente diferentes aspectos teórico-tecnológicos relacionados de algún modo con este proyecto.

Es necesario apuntar que todos los aspectos relacionados con el estudio y selección de herramientas y tecnologías a emplear en el proyecto han sido responsabilidad única del autor. Todas las decisiones tomadas a la hora de escoger entre diversas opciones para el desarrollo del sistema se han llevado a cabo valorando, principalmente, el conocimiento previo de las mismas y la adecuación a la solución a desarrollar, además de haber tomado en cuenta los riesgos derivados del uso de tecnologías poco conocidas o experimentales.

La única restricción impuesta por el cliente ha sido el uso del sistema de gestión de bases de datos implantado en la empresa, como se describe posteriormente.

3.1 Framework

Un framework o marco de trabajo define el conjunto estandarizado de conceptos, prácticas y criterios a tomar en consideración a la hora de enfocar un tipo de problema particular que se utiliza como referencia para la resolución de nuevos problemas de similares características. En el campo del desarrollo de software, el término framework se refiere a una estructura conceptual y tecnológica de soporte definido, basada normalmente en componentes software concretos y utilizada como base para la estructuración y desarrollo de nuevas aplicaciones. Usualmente se complementa con el soporte de herramientas de desarrollo y bibliotecas que contribuyen al desarrollo en conjunto de los diferentes componentes de un proyecto.

Por el tipo de proyecto que nos ocupa, resulta importante centrarse en los enfocados en aplicaciones web, un tipo concreto de frameworks con la particularidad de ser diseñados para el desarrollo de sitios web dinámicos, aplicaciones y servicios web. Tienen como objetivo reducir la sobrecarga comúnmente asociada al desarrollo web por medio de técnicas como el uso de librerías para el acceso a base de datos, plantillas y gestión de sesiones, todo ello bajo la máxima de la reutilización de código.

Llegados a este punto resulta adecuado mencionar el patrón Modelo-Vista-Controlador (MVC), puesto que es el más ampliamente adoptado por los frameworks enfocados al desarrollo de aplicaciones web. No obstante, este concepto se describe individualmente en la sección posterior.

3.2 Modelo-Vista-Controlador (MVC)

Descrito por primera vez en 1979 por T. Reenskaug, se trata de un patrón de diseño destinado a sistemas complejos en los que las responsabilidades son separadas en capas bien diferenciadas. Es muy habitual en aplicaciones web, aunque no exclusivamente, ya que el desarrollo de numerosas aplicaciones de escritorio están basadas en este patrón.

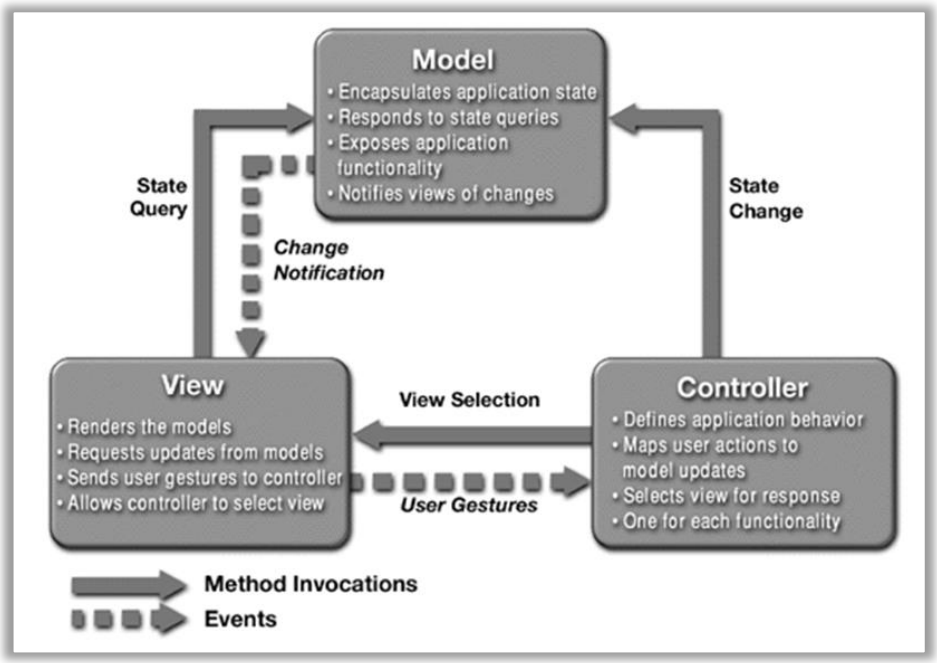


Figura 3.1. Patrón MVC

3.2.1 Modelo

El modelo es la capa responsable de mantener el estado de los objetos empleados por el usuario a lo largo de la sesión, implementa las reglas de negocio y garantiza el almacenamiento persistente de los cambios realizados.

3.2.2 Vista

La vista es la capa encargada de recoger las peticiones del usuario, transmitirlas al controlador y mostrar los resultados derivados de las operaciones realizadas en el modelo a través de los datos depositados en la sesión, el contexto o la petición.

3.2.3 Controlador

Se encarga de coordinar el flujo de la aplicación, conectando adecuadamente las peticiones de la vista con los servicios del modelo y actualizando los resultados en la petición, la sesión o el contexto para que los cambios sean accesibles a la vista.

3.3 Servidor Web

La función principal de un servidor web es la de servir páginas web de acuerdo a las peticiones de los clientes por medio del protocolo HTTP. Esto supone la entrega de documentos HTML y cualquier otro contenido adicional que forme parte de un documento, como pueden ser imágenes, hojas de estilo o scripts, entre otros.

Un agente de usuario, o lo que es lo mismo, un navegador web, será el encargado de iniciar la comunicación con el servidor. Para ello, realiza una petición HTTP sobre un recurso concreto y el servidor responderá con el contenido solicitado, o bien, con un mensaje de error si no es posible servir esa petición. Un servidor web cuenta con otras funciones distintas a servir contenido. Una implementación completa del protocolo HTTP supone la capacidad de recibir elementos desde el cliente, por ejemplo, en el caso del envío de formularios y carga de ficheros.

Diversos servidores web genéricos soportan la programación del lado del servidor. Eso significa que el comportamiento del servidor web puede ser extendido y modificado por medio de scripts, todo ello sin modificar el software base del servidor. Con frecuencia, esta capacidad se utiliza para generar documentos HTML de forma dinámica o al vuelo, de forma opuesta a como sucede con los documentos estáticos. Esto resulta útil cuando el contenido con el que se trabaja supone el uso de operaciones de lectura/escritura sobre bases de datos, mientras que el uso de contenidos estáticos favorece el cacheado en memoria, lo que supone un incremento considerable de la velocidad de respuesta por parte del servidor.

Para el presente proyecto, el servidor web empleado será Apache Tomcat, puesto que, además de funcionar como servidor web, cuenta con la capacidad adicional de soporte de JavaServer Pages (JSPs).

3.4 Java Enterprise Edition

Java Enterpise Edition (Java EE o JEE) es un entorno independiente de la plataforma centrado en Java para desarrollar, crear e implementar aplicaciones web empresariales. Java EE incluye diversos componentes de Java Standard Edition (Java SE) y consta de un conjunto de servicios, APIs y protocolos que proporcionan la funcionalidad necesaria para desarrollar aplicaciones web de varios niveles, simplificando el diseño y la implementación, y por tanto, reduciendo la necesidad de formación para los programadores al crear componentes modulares normalizados y reutilizables, así como al permitir controlar muchos aspectos de la programación automáticamente por nivel.

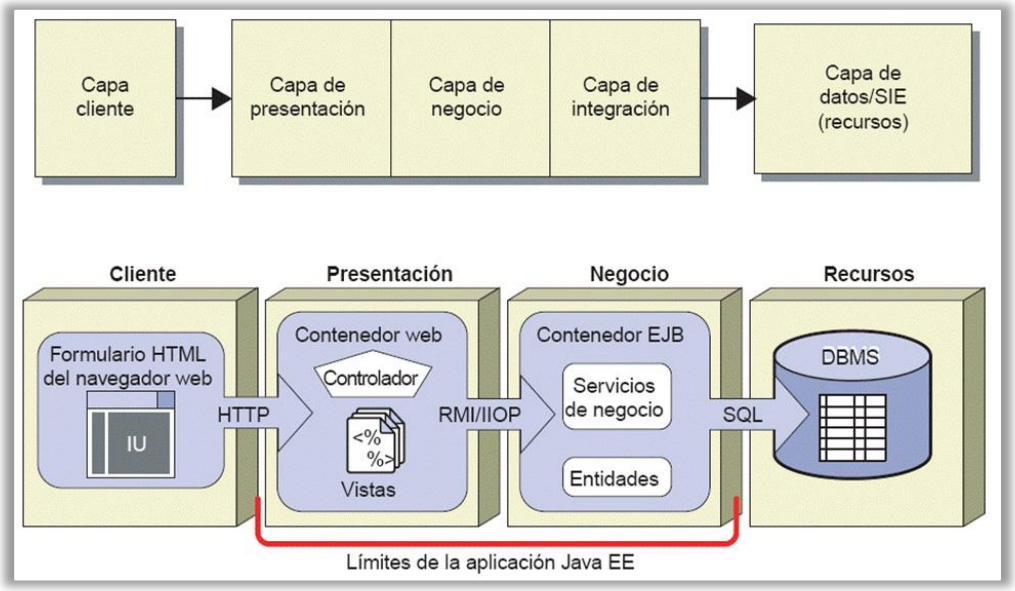


Figura 3.2. Java Enterprise Edition

La plataforma Java EE define un conjunto de especificaciones que facilitan el desarrollo y despliegue de aplicaciones multicapa, además de diversas técnicas que proveen soluciones completas, seguras, estables y escalables para el desarrollo, despliegue y gestión de aplicaciones de múltiples niveles de funcionalidad basadas en servidores, a la vez que se reduce el coste y complejidad de la implementación, lo cual da lugar a servicios fácilmente desplegables y extensibles.

3.5 Apache Struts

Tal y como se ha dicho en las secciones previas, el patrón MVC se emplea en multitud de aplicaciones web, lo cual es motivo de que diversos frameworks enfocados al desarrollo de este tipo de sistemas lo implementen. Apache Struts es uno de ellos.

Sus orígenes se remontan al año 2000, cuando su creador Craig McClanahan donó el proyecto a la Apache Foundation. Al inicio, se trataba de una parte del proyecto Jakarta, hasta que en 2005 se convirtió en un proyecto independiente y de alto nivel.

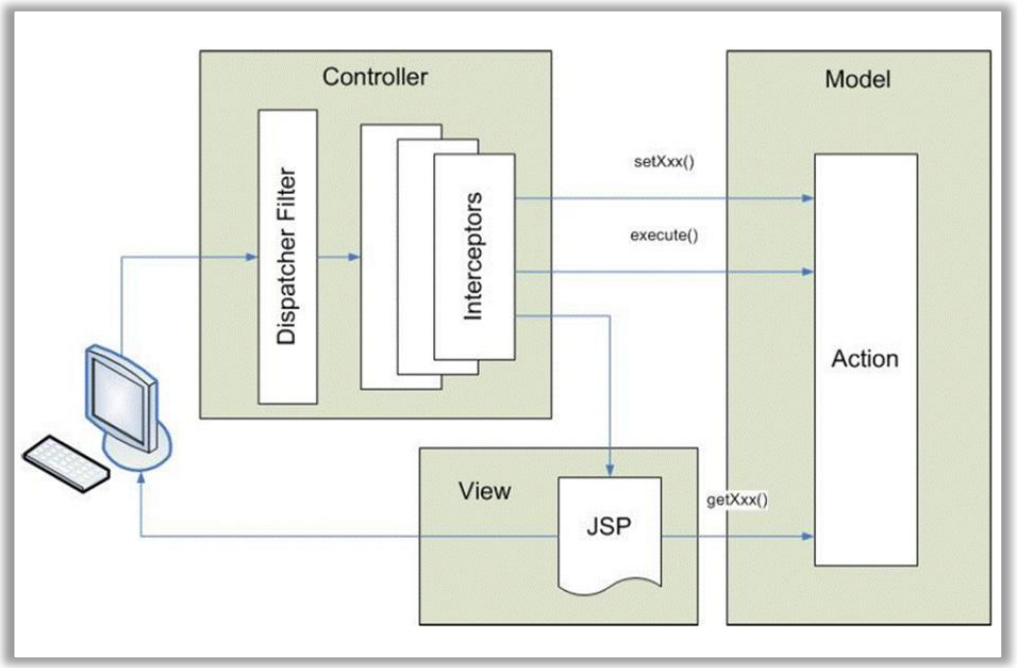


Figura 3.3. Apache Struts 2

Struts 2 es un framework concebido para el desarrollo de aplicaciones web Java bajo el patrón MVC en la plataforma Java EE. En la versión 2 del framework se introdujeron una serie de mejoras sobre la primera versión enfocadas a la simplificación de las tareas más comunes y sistemáticas en el desarrollo de aplicaciones web.

3.6 JavaServer Pages

JavaServer Pages (JSP) es una tecnología desarrollada por Sun Microsystems y lanzada por primera vez en 1999. Se encuentra actualmente en la versión 2.2, la cual ha sido empleada en el desarrollo de este proyecto.

JSP permite generar contenido dinámico para la web en forma de documentos HTML por medio de scriptlets de código Java, además de mediante acciones JSP predefinidas utilizando etiquetas. Estas etiquetas pueden ser enriquecidas mediante Bibliotecas de Etiquetas o TagLibs externas o desarrolladas a medida.

Las JSP pueden considerarse como un método alternativo y simplificado para la construcción de servlets, por lo que una página JSP podrá hacer todo lo que un servlet podría y viceversa. Cada versión de la especificación de JSP está fuertemente vinculada a una versión particular de la especificación de servlets.

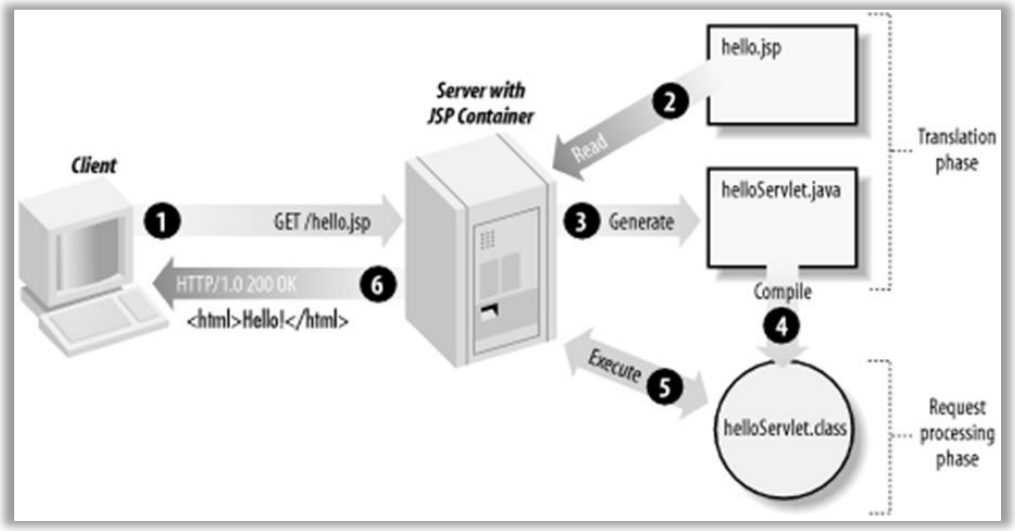


Figura 3.4. JavaServer Pages

El funcionamiento de la tecnología JSP se basa en que el servidor de aplicaciones interpreta el código contenido en la propia página JSP para construir el código Java del servlet a generar. Este servlet será el componente que finalmente generará el documento HTML que se presentará en el agente de usuario o navegador.

3.7 Lenguaje Unificado de Modelado

El Lenguaje Unificado de Modelado (UML, por sus siglas en inglés, Unified Modeling Language) es el leguaje de modelado de sistemas software más conocido y adoptado en la actualidad, contando con el respaldo del OMG (Object Management Group). Se trata de un lenguaje gráfico concebido para especificar, construir, visualizar y documentar un sistema. UML propone un método estándar para describir el modelo del sistema, incluyendo aspectos conceptuales tales como procesos de negocio y funciones del sistema. Soporta además el modelado de aspectos concretos, como son las expresiones propias de determinados lenguajes de programación, esquemas de bases de datos y componentes de más alto nivel.

Fue desarrollado por Grady Booch, Ivar Jacobson y Jim Rumbaugh en la década de los 90, y adoptado por la Object Management Group en 1997. No fue aceptado como estándar ISO hasta el año 2000. Tras una serie de revisiones diversa relevancia, la versión actual de la especificación UML, 2.5, fue lanzada en octubre de 2012.

Logotipo 7. Unified Modeling Language

UML se puede aplicar en el desarrollo de software haciendo uso de gran variedad de artefactos que dan soporte al uso de una metodología de desarrollo de software, sin especificar en sí mismo qué metodología o proceso usar de forma concreta.

3.8 MÉTRICA Versión 3

MÉTRICA es una metodología de planificación, desarrollo y mantenimiento de sistemas de información promovida por el Ministerio de Administraciones Públicas del Gobierno de España para la sistematización de actividades del ciclo de vida de los proyectos software en el ámbito de las administraciones públicas. Esta metodología está basada en el modelo de procesos del ciclo de vida de desarrollo ISO/IEC 12207 (Information Technology - Software Life Cycle Processes), así como en la norma ISO/IEC 15504 SPICE (Software Process Improvement And Assurance Standards Capability Determination).

La metodología MÉTRICA Versión 3 ofrece a las organizaciones un instrumento útil para la sistematización de las actividades que dan soporte al ciclo de vida del software dentro del marco que permite alcanzar los siguientes objetivos:

- Proporcionar o definir Sistemas de Información que ayuden a conseguir los fines de la organización mediante la definición de un marco estratégico para el desarrollo de los mismos
- Dotar a la organización de productos software que satisfagan las necesidades de los usuarios dando una mayor importancia al análisis de requisitos
- Mejorar la productividad de los departamentos de Sistemas y Tecnologías de la Información y las Comunicaciones, permitiendo una mayor capacidad de adaptación a los cambios y teniendo en cuenta la reutilización en la medida de lo posible
- Facilitar la comunicación y entendimiento entre los distintos participantes en la producción de software a lo largo del ciclo de vida del proyecto, teniendo en cuenta su papel y responsabilidad, así como las necesidades de todos y cada uno de ellos
- Facilitar la operación, mantenimiento y uso de los productos software obtenidos

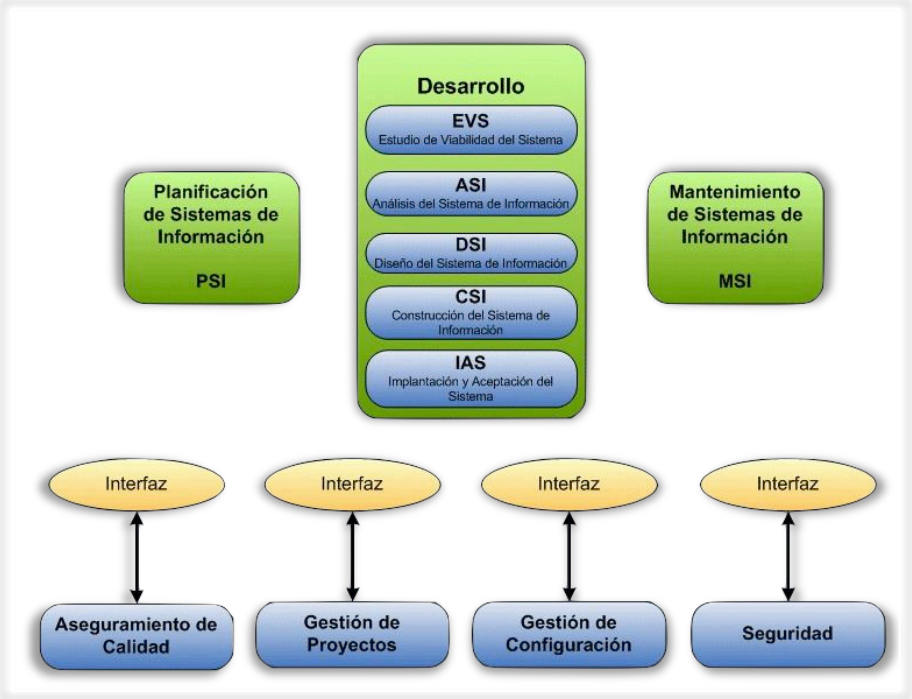


Figura 3.5. MÉTRICA Versión 3

En la elaboración de MÉTRICA Versión 3 se han tenido en cuenta los métodos de desarrollo más extendidos, así como los últimos estándares de ingeniería del software y calidad, además de referencias específicas en cuanto a seguridad y gestión de proyectos. También se ha tenido en cuenta la experiencia de los usuarios de las versiones anteriores para solventar los problemas o deficiencias detectados.

En una única estructura, la metodología MÉTRICA Versión 3 facilita a través de interfaces la realización de los procesos de apoyo y organizativos, tales como la Gestión de Proyectos, Gestión de Configuración, Aseguramiento de Calidad y Seguridad.

MÉTRICA Versión 3 es la metodología empleada para el desarrollo de esta documentación. Este documento es una adaptación de dicha metodología, contemplando solo aquellos apartados que se han considerado más adecuados para un trabajo fin de máster.

Capítulo 4. Planificación del Proyecto

A continuación se presentan los siguientes documentos relativos a la planificación del proyecto:

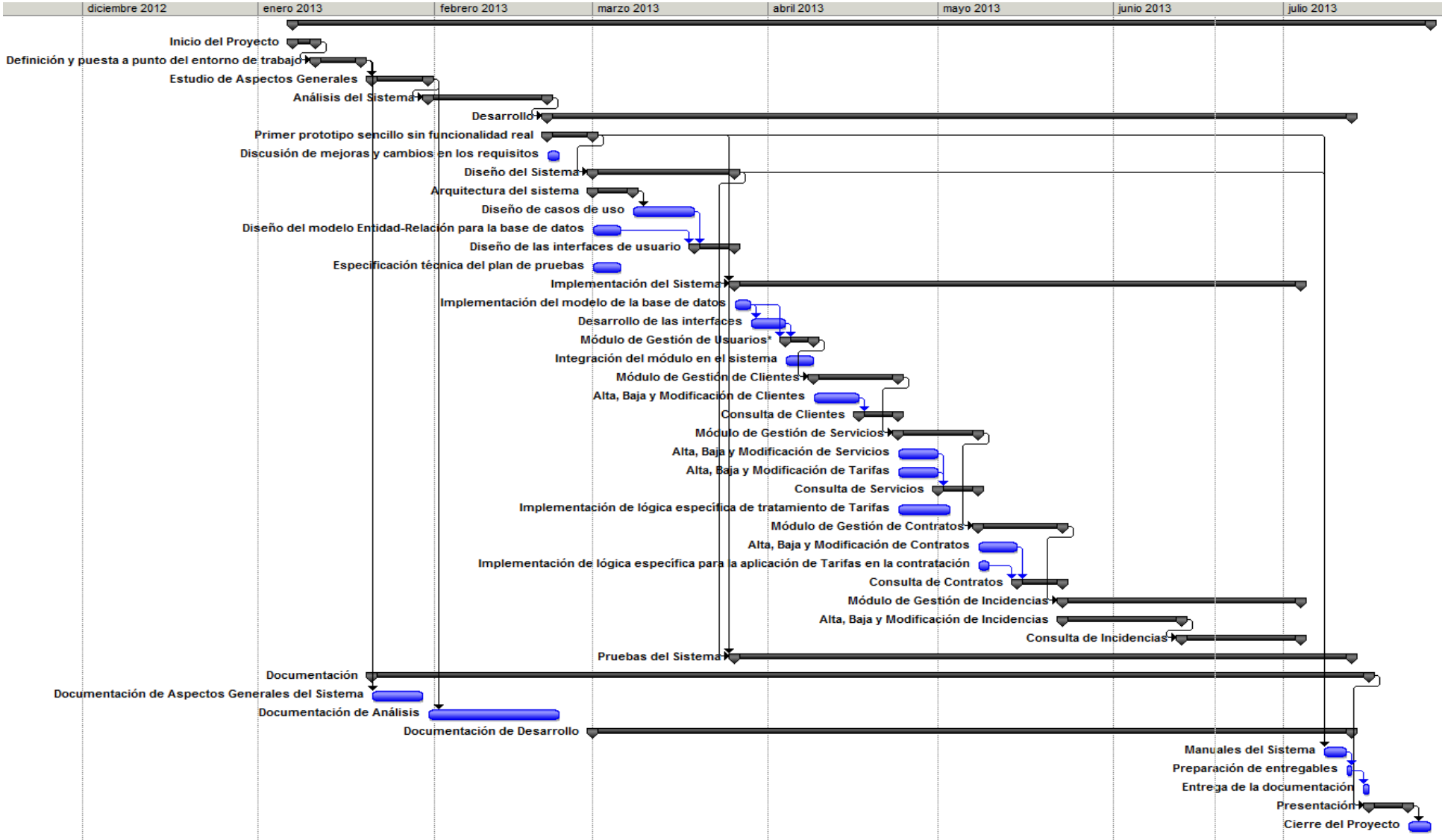
- Listado de Tareas: muestra un listado resumido de las tareas que componen la totalidad del proyecto
- Diagrama de Gantt: ofrece una vista global del desarrollo temporal de las tareas descritas en el punto anterior
- Resumen de la Planificación: recoge los valores de las variables más significativas de la planificación

Es importante resaltar que todos los datos presentados se muestran en una vista resumida, donde se han contraído las tareas de niveles más inferiores para no desvirtuar en la medida de lo posible el formato de este documento. No obstante, se dispone al lector del fichero con la planificación completa en el formato de Microsoft Office Project 2007.

4.1 Tareas

Id	Nombre de tarea	Nivel de esquema	Duración	Comienzo	Fin	Predecesoras
1	**SISTEMA DE GESTIÓN DE INCIDENCIAS, CLIENTES, CONTRATOS Y SERVICIOS PARA ECOCOMPUTER S.L.**	**1**	**145 días**	**lun 07/01/13**	**vie 26/07/13**	
2	**Inicio del Proyecto**	**2**	**4 días**	**lun 07/01/13**	**jue 10/01/13**	
6	**Definición y puesta a punto del entorno de trabajo**	**2**	**6 días**	**vie 11/01/13**	**vie 18/01/13**	**2**
14	**Estudio de Aspectos Generales**	**2**	**8 días**	**lun 21/01/13**	**mié 30/01/13**	**6**
21	**Análisis del Sistema**	**2**	**15 días**	**jue 31/01/13**	**mié 20/02/13**	**14**
29	**Desarrollo**	**2**	**102 días**	**jue 21/02/13**	**vie 12/07/13**	**21**
30	**Primer prototipo sencillo sin funcionalidad real**	**3**	**6 días**	**jue 21/02/13**	**jue 28/02/13**	
33	Discusión de mejoras y cambios en los requisitos	3	2 días	jue 21/02/13	vie 22/02/13	
34	**Diseño del Sistema**	**3**	**17 días**	**vie 01/03/13**	**lun 25/03/13**	**30**
35	**Arquitectura del sistema**	**4**	**5 días**	**vie 01/03/13**	**jue 07/03/13**	
39	Diseño de casos de uso	4	7 días	vie 08/03/13	lun 18/03/13	35
40	Diseño del modelo Entidad-Relación para la base de datos	4	3 días	vie 01/03/13	mar 05/03/13	
41	**Diseño de las interfaces de usuario**	**4**	**5 días**	**mar 19/03/13**	**lun 25/03/13**	**39;40**
47	Especificación técnica del plan de pruebas	4	3 días	vie 01/03/13	mar 05/03/13	
48	**Implementación del Sistema**	**3**	**72 días**	**mar 26/03/13**	**mié 03/07/13**	**30;34**
49	Implementación del modelo de la base de datos	4	3 días	mar 26/03/13	jue 28/03/13	
50	Desarrollo de las interfaces	4	4 días	vie 29/03/13	mié 03/04/13	49
51	**Módulo de Gestión de Usuarios***	**4**	**3 días**	**jue 04/04/13**	**lun 08/04/13**	**49;50**
52	Integración del módulo en el sistema	5	3 días	jue 04/04/13	lun 08/04/13	
53	**Módulo de Gestión de Clientes**	**4**	**11 días**	**mar 09/04/13**	**mar 23/04/13**	**51**
54	Alta, Baja y Modificación de Clientes	5	6 días	mar 09/04/13	mar 16/04/13	
55	**Consulta de Clientes**	**5**	**5 días**	**mié 17/04/13**	**mar 23/04/13**	**54**
59	**Módulo de Gestión de Servicios**	**4**	**10 días**	**mié 24/04/13**	**mar 07/05/13**	**53**
60	Alta, Baja y Modificación de Servicios	5	5 días	mié 24/04/13	mar 30/04/13	
61	Alta, Baja y Modificación de Tarifas	5	5 días	mié 24/04/13	mar 30/04/13	
62	**Consulta de Servicios**	**5**	**5 días**	**mié 01/05/13**	**mar 07/05/13**	**60;61**
66	Implementación de lógica específica de tratamiento de Tarifas	5	7 días	mié 24/04/13	jue 02/05/13	
67	**Módulo de Gestión de Contratos**	**4**	**11 días**	**mié 08/05/13**	**mié 22/05/13**	**59**
68	Alta, Baja y Modificación de Contratos	5	5 días	mié 08/05/13	mar 14/05/13	
69	Implementación de lógica específica para la aplicación de Tarifas en la contratación	5	2 días	mié 08/05/13	jue 09/05/13	
70	**Consulta de Contratos**	**5**	**6 días**	**mié 15/05/13**	**mié 22/05/13**	**68;69**
74	**Módulo de Gestión de Incidencias**	**4**	**30 días**	**jue 23/05/13**	**mié 03/07/13**	**67**
75	**Alta, Baja y Modificación de Incidencias**	**5**	**15 días**	**jue 23/05/13**	**mié 12/06/13**	
85	**Consulta de Incidencias**	**5**	**15 días**	**jue 13/06/13**	**mié 03/07/13**	**75**
95	**Pruebas del Sistema**	**3**	**79 días**	**mar 26/03/13**	**vie 12/07/13**	**30;34**
107	**Documentación**	**2**	**126 días**	**lun 21/01/13**	**lun 15/07/13**	
108	Documentación de Aspectos Generales del Sistema	3	7 días	lun 21/01/13	mar 29/01/13	6
109	Documentación de Análisis	3	17 días	jue 31/01/13	vie 22/02/13	14
110	**Documentación de Desarrollo**	**3**	**96 días**	**vie 01/03/13**	**vie 12/07/13**	
114	Manuales del Sistema	3	4 días	lun 08/07/13	jue 11/07/13	30;34
115	Preparación de entregables	3	1 día	vie 12/07/13	vie 12/07/13	114
116	Entrega de la documentación	3	1 día	lun 15/07/13	lun 15/07/13	115
117	**Presentación**	**2**	**5 días**	**mar 16/07/13**	**lun 22/07/13**	**107**
121	Cierre del Proyecto	2	4 días	mar 23/07/13	vie 26/07/13	117

4.2 Diagrama de Gantt



4.3 Resumen

La siguiente tabla muestra un resumen de las principales propiedades de la planificación del proyecto, las cuales permiten obtener una síntesis de la misma.

Propiedad	Valor
Número total de tareas	121
Duración total del proyecto (días laborables)	145
Duración de la tarea más extensa (días) – *Exceptuando tareas de pruebas y documentación, las cuales se realizan de forma concurrente al desarrollo*	30
Duración de la tarea más breve (días)	1 (varias)
Media de holguras (días)	8,3
Número de tareas críticas	7
Número máximo de tareas críticas simultáneas	2

Tabla 4.1. Vistazo general de la Planificación del Proyecto

Capítulo 5. Análisis del Sistema

En este capítulo se enumeran con mayor detalle los requisitos del sistema a desarrollar. Partiendo de una especificación textual de los mismos se extraerá una lista clasificada y se diseñaran los casos de uso asociados.

Posteriormente, se identifican las clases asociadas a cada caso de uso y se muestra el diagrama de clases preliminar junto con una breve descripción de las clases y subsistemas identificados, junto con un análisis en profundidad de los casos de uso obtenidos en el paso anterior.

Tras esto, se describe la especificación del proceso de diseño de las interfaces de usuario, el cual cuenta con particularidades acordadas al inicio del proyecto, para finalmente, presentar una especificación del plan de pruebas a llevar a cabo durante la fase de implementación del sistema.

5.1 Definición del Sistema

5.1.1 Determinación del Alcance del Sistema

Lo que se pretende con el desarrollo de este sistema es crear una herramienta completamente a medida para el control y gestión del proceso de atención de incidencias relacionadas con los servicios prestados por una empresa de informática, tanto de mantenimiento de hardware e instalaciones como del software distribuido y los programas desarrollados. Todas las incidencias recibidas serán procesadas por el sistema, donde pasarán por diferentes estados desde su creación hasta su cierre.

Se conformará como una aplicación web, basada en perfiles de usuario y roles. Cada usuario y grupo de usuarios tendrá unos permisos determinados y podrá realizar determinadas acciones, por lo que la información que verá cada perfil de usuario será diferente, aunque para todos ellos estará organizada en contenedores o módulos. La clasificación de las vistas y sus funcionalidades se define en el siguiente listado:

- **Módulo de Clientes**: Contendrá todas las vistas de presentación de información relativa a los clientes de la empresa, así como diversos formularios de introducción de información para la creación, edición, eliminado y búsqueda.
- **Módulo de Contratos**: Al igual que el anterior, este módulo mostrará el listado de contratos suscritos por los clientes con la empresa, así como el detalle de cada uno de ellos. Dado que un contrato está compuesto por los servicios que engloba, dentro de este módulo será necesario proveer de los mecanismos necesarios para la interoperabilidad con el módulo de servicios. Finalmente, se habilitarán una serie de formularios para la creación, edición y búsqueda de contratos.

- **Módulo de Servicios**: En este módulo se perfilará un catálogo con los servicios ofertados por la empresa. Cada servicio tiene, además de unos atributos descriptivos, una serie de tarifas asociadas, por lo que será necesario dotar al módulo de las funciones necesarias para la gestión de dichas tarifas. Esto conlleva la creación de formularios para las operaciones básicas de creación, modificación y eliminado de entidades, así como para la presentación de los datos existentes.
- **Módulo de Incidencias**: Perfilado como módulo central del sistema, será necesario establecer las conexiones necesarias entre este y el resto de los ya descritos. Para el manejo de una incidencia será necesario conocer qué cliente la comunica, respecto a qué contrato suscrito lo hace y con qué servicio asociado. Conllevará una mayor complejidad a la hora de insertar o modificar datos en una entidad, puesto que, además de los datos textuales de la misma, deberá contemplarse la posibilidad de gestionar las intervenciones, adjuntos y relaciones de cada una. Además, será imprescindible la implementación de una función que permita el clonado de una incidencia, así como la generación de informes en formato imprimible. En lo referente a las operaciones de consulta de datos, será necesario disponer de un buscador parametrizable, además de permitir ordenar el listado de incidencias en base a distintos criterios.

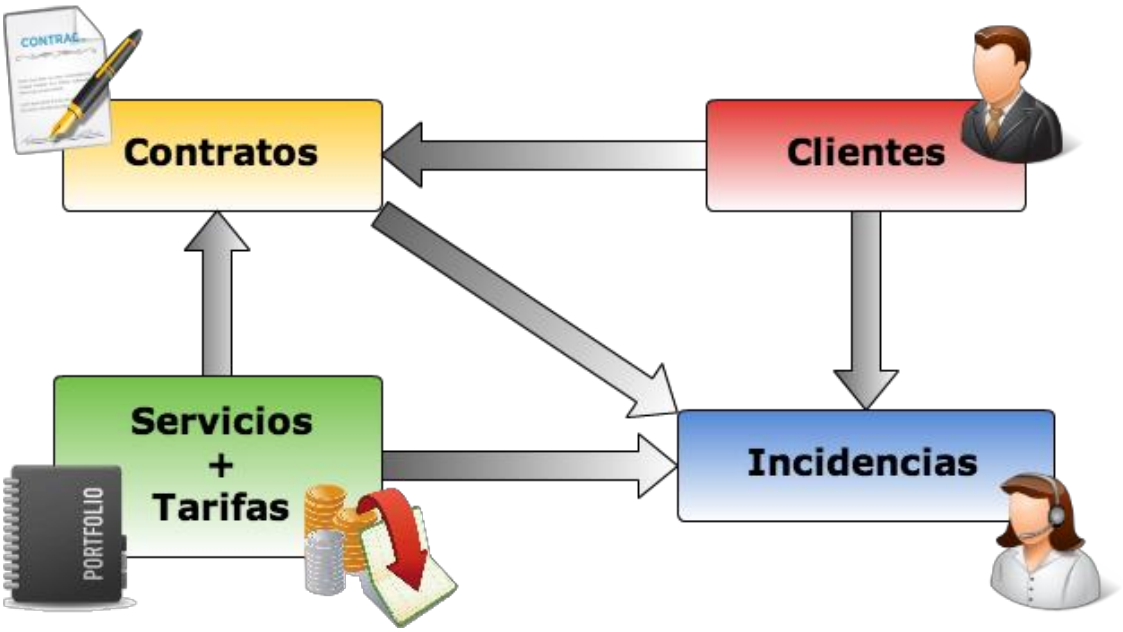


Figura 5.1. Conexión entre componentes del sistema

Dado que toda la funcionalidad relativa al módulo de Gestión de Usuarios será desarrollada en un proyecto independiente de este, se obviarán los detalles del mismo, haciendo referencia únicamente a aquellas particularidades de aquel que resulten necesarias para comprender algún aspecto de este sistema.

5.2 Requisitos del Sistema

5.2.1 Obtención de los Requisitos del Sistema

A continuación se presenta la especificación de la aplicación, tal y como el cliente la ha descrito, de forma textual. Posteriormente, se han extraído y organizado todos los requisitos aprobados para el proyecto, obtenidos a partir del filtrado y discusión individual de cada uno de los requerimientos proporcionados en la especificación textual.

5.2.1.1 Especificación textual

Existe un problema de eficiencia y complejidad relativo al proceso mediante el cual se gestionan las incidencias que los clientes de EcoComputer ponen en conocimiento. Por ello, se precisa de una herramienta creada a medida que centralice y simplifique las tareas derivadas de la su gestión. Dicha herramienta se desarrollará para funcionar como una aplicación web, haciendo uso de perfiles de usuario y permisos de acceso basados en roles y grupos.

El acceso al sistema por parte de los usuarios estará restringido mediante autenticación con usuario y contraseña. Cada usuario de la aplicación deberá estar asociado a un perfil, además de pertenecer a uno o más grupos dependiendo de sus funciones dentro de la empresa. El gestor de proyectos será el encargado de gestionar las altas, bajas y modificaciones de los usuarios, así como sus permisos y pertenencia a grupos. Un usuario podrá recuperar su contraseña mediante el envío de una nueva al correo electrónico que proporcionó al ser dado de alta en el sistema.

El modelo de negocio que establece la relación entre la empresa prestadora de los servicios y el cliente que los percibe se basa en un acuerdo entre ambas partes por medio de un contrato de servicio en el que se acordará una cantidad de horas-hombre para un tipo de servicio determinado (microinformática, desarrollo, mantenimiento, etc.). Cuando un cliente requiere una actuación por parte del personal de la empresa, se abrirá una incidencia dentro de uno de sus contratos. Esta incidencia requerirá una actuación por parte del personal y consumirá una cantidad de tiempo que será descontada del total del tipo de servicio contratado. Se debe tratar este aspecto para su gestión de manera flexible, puesto que es posible que una actuación determinada requiera más horas de las que ofrezca el contrato y sea necesario facturarlas de forma independiente, lo cual quedará a criterio del departamento de administración.

Una incidencia tendrá por tanto unas características concretas que deberán ser gestionadas por los actores implicados, suministrando toda la información necesaria para el correcto control y seguimiento de la misma, conformando una especie de tablero de asignaciones donde los empleados puedan consultar las tareas que tiene pendientes de realizar.

Adicionalmente, existirán otras funcionalidades complementarias al control de incidencias, como son la búsqueda de incidencias en base a filtros, anotación de intervenciones, observaciones y documentos relacionados, relaciones entre incidencias, etc.

En cuanto al aspecto visual de la aplicación, se requiere una interfaz sencilla y organizada, por lo que resultará de vital importancia una organización lógica y estructurada de la información presentada al usuario.

Como medida de seguridad y control, se mantendrá un registro de accesos a la aplicación, así como de las acciones llevadas a cabo por cada usuario en la sesión, aunque no se precisará de una interfaz dedicada a la visualización de estos datos, sino únicamente su almacenado en la base de datos.

5.2.1.2 Requisitos Funcionales

5.2.1.2.1 Requisitos del Módulo de Gestión de Usuarios

La siguiente tabla muestra los requisitos de los que precisa el módulo de Gestión de Usuarios. Si bien su cumplimiento no es competencia de este proyecto, se presentan con el objetivo de facilitar al lector la comprensión del sistema en su conjunto, puesto que se trata de una parte de la que este sistema depende directamente.

RF1 - Requisitos del Módulo de Gestión de Usuarios		
Código	**Nombre Requisito**	**Descripción del Requisito**
RF1.1	Acceso al sistema	El acceso al sistema será restringido mediante un nombre de usuario y contraseña
RF1.2	Organización de usuarios en grupos	Los usuarios pertenecerán a grupos. Un usuario puede pertenecer a más de un grupo
RF1.3	Grupos de usuarios predefinidos	Los grupos de usuarios definidos inicialmente en el sistema serán: • Administración • Centro de Atención a Usuarios (CAU) • Desarrollo • Gerencia • Técnicos
RF1.4	Modificación de grupos de usuarios	Será posible dar de alta, baja y modificar los grupos de usuarios de la web
RF1.5	Perfiles de usuario	Existirán varios perfiles de usuario en el sitio web: • Administración • Administrador (del Sistema) • Desarrollador • Organizador • Técnico
RF1.6	Modularidad	La aplicación estará dividida en menús con módulos. Los permisos de los perfiles de usuario y de los usuarios se asignarán unitariamente para cada uno de esos módulos
RF1.7	Asignación de permisos y	Al dar de alta un nuevo usuario en el sistema, éste

RF1 - Requisitos del Módulo de Gestión de Usuarios		
Código	**Nombre Requisito**	**Descripción del Requisito**
	herencia de los mismos	heredará los permisos del perfil de usuario al que pertenece (aunque se podrán posteriormente particularizar para cada uno de ellos)
RF1.8	Gestión de perfiles y permisos de usuario	Debe ser posible dar de alta, baja y modificar los perfiles de usuario y sus permisos de acceso
RF1.9	Administración de usuarios	El administrador tendrá permisos para dar de alta a nuevos usuarios en el sistema
RF1.10	Consulta de documentación para el grupo de técnicos	Los técnicos tendrán acceso a toda la documentación de los contratos de clientes e incidencias de su grupo de usuarios
RF1.11	Actualización de documentación por el administrador y los técnicos	El personal de Ecocomputer podrá consultar, descargar y también actualizar toda la documentación desde su cuenta de usuario
RF1.12	Recuperación de contraseña de usuario	Habrá una opción llamada “Recordar contraseña”, de forma que el sistema enviará por e-mail al usuario su contraseña de acceso al repositorio

Tabla 5.1. Requisitos del Módulo de Gestión de Usuarios

5.2.1.2.2 Requisitos del Módulo de Gestión de Clientes

RF2 - Requisitos del Módulo de Gestión de Clientes		
Código	**Nombre Requisito**	**Descripción del Requisito**
RF2.1	Alta de nuevo cliente	Se podrán incluir nuevos clientes en el sistema introduciendo los siguientes datos: • Nombre del cliente • Dirección • Población • Provincia • Teléfono • Fax (Opcional) • Logotipo (Opcional)
RF2.2	Modificación de datos del cliente	Se podrán editar todos los datos de los clientes existentes en el sistema
RF2.3	Eliminación de cliente	Se podrán eliminar los clientes almacenados en el sistema, previa confirmación de la acción
RF2.4	Información del cliente	La información de cada cliente se organizará en dos secciones: • Datos generales • Incidencias vinculadas a sus contratos
RF2.4.1	Datos generales del cliente	La vista *Datos Generales* presentará la siguiente información referente al cliente: • Nombre del cliente • Logotipo • Dirección • Población • Provincia • Teléfono y/o Fax
RF2.4.2	Visualización de incidencias relacionadas con el cliente	La vista *Ver Incidencias* proporcionará un acceso directo al historial de incidencias vinculadas a los contratos del cliente. Ver requisitos RF5.x
RF2.5	Listado de clientes	Se podrá consultar un listado con todos los clientes albergados en el sistema. Esta vista mostrará todos los campos descritos en RF2.1 excepto el logotipo
RF2.6	Filtrado de registros de clientes	Existirá un filtro de búsqueda en base a texto, además de filtros particulares en base a la provincia y población de los clientes. Se podrá además seleccionar la cantidad de registros a mostrar y paginar los resultados en caso de existir más de los visualizados
RF2.7	Ordenación de registros	Los registros de clientes mostrados en el listado serán ordenables en base a los campos que los componen

Tabla 5.2. Requisitos del Módulo de Gestión de Clientes

5.2.1.2.3 Requisitos del Módulo de Gestión de Contratos

RF3 - Requisitos del Módulo de Gestión de Contratos		
Código	**Nombre Requisito**	**Descripción del Requisito**
RF3.1	Alta de nuevo contrato	Se permitirá asociar un nuevo contrato a un cliente existente en el sistema. Para ello, se precisarán los siguientes datos: • Cliente que suscribe el contrato • Fecha de alta • Fecha de inicio • Fecha de fin • Tipo de facturación • Estado • Título • Descripción
RF3.2	Modificación de las propiedades de un contrato	Se podrán editar todos los datos de los contratos suscritos con los clientes, salvo el cliente involucrado
RF3.3	Eliminación de contrato	Se permitirá la eliminación de los contratos almacenados en el sistema, previa confirmación de la acción
RF3.4	Gestión de los servicios de un contrato	Para un contrato existente, se permitirá la gestión de los servicios a englobar en el mismo
RF3.4.1	Inclusión de un servicio en el contrato	Se podrán asociar, del catálogo de servicios ofrecidos, aquellos que el cliente precise. Para ello, se precisarán los siguientes datos: • Nombre del servicio a asociar • Tarifa a aplicar • Si se precisa, precio especial • Número de horas contratadas • Opción de desplazamiento • Descripción
RF3.4.2	Edición de los datos de un servicio asociado al contrato	Se podrán editar todos los datos de los servicios asociados a un contrato
RF3.4.3	Desvinculación de un servicio asociado	Se permitirá eliminar los servicios asociados al contrato
RF3.4.4	Listado de servicios asociados a un contrato	Se podrá consultar un listado con los servicios incluidos en el contrato. Este listado incluirá todos los campos descritos en RF3.4.1
RF3.5	Listado de contratos	Se podrá consultar un listado con todos los contratos incluidos en el sistema. Esta vista mostrará todos los campos descritos en RF3.1 salvo la descripción y la fecha de alta. Además, se incluirá un campo denominado *Importe total* que presentará la suma de los importes de los servicios asociados al contrato
RF3.6	Filtrado de registros de contratos	Existirán los siguientes filtros: • Búsqueda en base a texto

RF3 - Requisitos del Módulo de Gestión de Contratos		
Código	**Nombre Requisito**	**Descripción del Requisito**
		• Búsqueda de contratos activos en base a una fecha • Búsqueda por nombre del cliente • Búsqueda por tipo de facturación Se podrá además seleccionar la cantidad de registros a mostrar y paginar los resultados en caso de existir más de los visualizados
RF3.7	Ordenación de registros	Los registros de contratos mostrados en el listado serán ordenables en base a los campos que los componen

Tabla 5.3. Requisitos del Módulo de Gestión de Contratos

5.2.1.2.4 Requisitos del Módulo de Gestión de Servicios

RF4 - Requisitos del Módulo de Gestión de Servicios		
Código	**Nombre Requisito**	**Descripción del Requisito**
RF4.1	Alta de nuevo servicio	La aplicación será capaz de gestionar el catálogo de servicios ofrecidos por la empresa. Para ello, se podrán incluir nuevos servicios en el sistema introduciendo simplemente la descripción del mismo
RF4.2	Modificación de un servicio	Se permitirá editar la descripción de los servicios
RF4.3	Eliminación de servicio	Si un servicio deja de estar disponible, será posible eliminarlo del catálogo, previa confirmación de la acción
RF4.4	Gestión de las tarifas de un servicio	Para un servicio existente, se permitirá la gestión de las tarifas del mismo
RF4.4.1	Inclusión de una tarifa en el servicio	Para un servicio existente, se podrá definir el conjunto de tarifas aplicables. Para ello, se precisarán los siguientes datos: • Nombre de la tarifa • Precio/hora base • Fecha de inicio de validez • Fecha de expiración
RF4.4.2	Edición de los datos de una tarifa	Se podrán modificar todos los datos de las tarifas asociadas a un servicio
RF4.4.3	Eliminación de una tarifa	Se permitirá eliminar las tarifas asociadas a un servicio, previa confirmación de la acción
RF4.4.4	Listado de tarifas de un servicio	Se podrá consultar un listado con las tarifas asociadas a un servicio. Este listado incluirá todos los campos descritos en RF4.4.1
RF4.4.5	Filtrado de registros de tarifas	Existirá un filtro de búsqueda de tarifas de un servicio basado en texto. Se podrá además seleccionar la cantidad de registros a mostrar y paginar los resultados en caso de existir más de los visualizados
RF4.4.6	Ordenación de registros	Los registros de tarifas mostrados en el listado serán ordenables en base a los campos que los componen
RF4.5	Listado de servicios	Se podrá consultar un listado con todos los servicios incluidos en el catálogo. Esta vista mostrará la descripción del servicio y el listado de tarifas del mismo, descrito en RF4.4.4

Tabla 5.4. Requisitos del Módulo de Gestión de Servicios

5.2.1.2.5 Requisitos del Módulo de Gestión de Incidencias

RF5 - Requisitos del Módulo de Gestión de Incidencias		
Código	**Nombre Requisito**	**Descripción del Requisito**

RF5.1	Alta de nueva incidencia	Se podrán incluir en el sistema las incidencias comunicadas por los clientes y asociadas a sus contratos. Para ello, se necesitarán los siguientes datos:
		• Título
		• Cliente afectado
		• Contrato al que se vinculará
		• Tipo de incidencia
		  ○ Control de Accesos
		  ○ Desarrollo
		  ○ Diseño Gráfico
		  ○ Dominios
		  ○ Electrónica
		  ○ LOPD
		  ○ Laboratorio Informático
		  ○ Pedido
		  ○ Presupuesto
		  ○ Redes
		  ○ Soporte Farmacias
		  ○ Soporte Técnico
		  ○ Web
		• Prioridad de actuación
		  ○ Crítica
		  ○ Alta
		  ○ Media
		  ○ Baja
		• Visibilidad
		  ○ Pública
		  ○ Privada
		• Modo de realización
		  ○ Presencial
		  ○ Remoto
		• Fecha de alta
		• Tiempo de resolución estimado
		• Estado
		  ○ No asignada
		  ○ Propuesta
		  ○ Asignada
		  ○ Resuelta
		  ○ Pendiente
		  ○ Suspendida
		  ○ Cerrada
		• Asignar a
		• Descripción

RF5.2	Modificación de las propiedades de una	Se podrán editar todos los datos de las incidencias almacenadas en el sistema

RF5 - Requisitos del Módulo de Gestión de Incidencias		
Código	**Nombre Requisito**	**Descripción del Requisito**
	incidencia	
RF5.3	Eliminación de incidencia	Se permitirá la eliminación individual de incidencias, previa confirmación de la acción
RF5.4	Gestión de las intervenciones asociadas a una incidencia	Se podrán gestionar las intervenciones relacionadas con una incidencia
RF5.4.1	Inclusión de nueva intervención	Se podrán añadir nuevas intervenciones a las incidencias. Para ello, será necesario facilitar los siguientes datos: • Motivo de la intervención • Referencia al parte de trabajo (de haberlo) • Tiempo empleado • Observaciones
RF5.4.2	Edición de los datos de una intervención	Se podrán modificar todos los datos de una intervención
RF5.4.3	Eliminación de intervenciones	Se podrán eliminar las intervenciones de una incidencia de forma individual, previa confirmación de la acción
RF5.4.4	Listado de intervenciones	Se mostrará, para cada incidencia, un listado con las intervenciones asociadas. Los campos a incluir en dicho listado son: • Usuario que intervino • Fecha de intervención • Número de parte asociado (de haberlo) • Motivo de la intervención • Tiempo empleado • Observaciones
RF5.5	Gestión de ficheros adjuntos a una incidencia	Se podrán gestionar los ficheros adjuntos empleados en el tratamiento de la incidencia
RF5.5.1	Alta de fichero adjunto	Se permitirá adjuntar nuevos ficheros a la incidencia. Para cargar un nuevo archivo se precisarán los siguientes datos: • Descripción del fichero • Ruta del fichero dentro del sistema de archivos del equipo del usuario Para la selección del fichero, se habilitará una ventana emergente que liste los directorios del equipo del usuario. No habrá restricciones en cuanto al tipo de fichero o tamaño del mismo, salvo las impuestas por la tecnología empleada
RF5.5.2	Eliminación de fichero adjunto	Se podrán eliminar los adjuntos de una incidencia, de forma individual, previa confirmación de la acción
RF5.5.3	Listado de adjuntos	Se mostrará, para cada incidencia, un listado con los adjuntos asociados a la misma. Este listado contendrá los siguientes campos:

RF5 - Requisitos del Módulo de Gestión de Incidencias		
Código	**Nombre Requisito**	**Descripción del Requisito**
		• Identificador interno del adjunto • Descripción • Enlace para su descarga
RF5.6	Gestión de relaciones	El sistema permitirá gestionar las relaciones entre incidencias existentes
RF5.6.1	Alta de relación	Se podrán incluir nuevas relaciones de la incidencia visualizada con el resto. Para ello, se solicitarán los siguientes datos: • Tipo de relación • Identificador de la incidencia a relacionar
RF5.6.2	Eliminación de relación	Se podrán eliminar individualmente las relaciones de la incidencia visualizada, previa confirmación de la acción
RF5.6.3	Listado de relaciones directas	Se mostrará un listado con aquellas incidencias con las que la visualizada guarda un vínculo directo. Los campos a presentar serán: • Tipo de relación • Identificador de la incidencia relacionada • Nombre de la incidencia relacionada • Enlace a la incidencia relacionada • Usuario que estableció la relación • Fecha de establecimiento de la relación
RF5.6.4	Listado de relaciones inversas	Se mostrará un listado con aquellas incidencias relacionadas indirectamente con la visualizada. Los campos a presentar serán: • Tipo de relación • Identificador de la incidencia relacionada • Nombre de la incidencia relacionada • Enlace a la incidencia relacionada • Usuario que estableció la relación • Fecha de establecimiento de la relación
RF5.7	Clonado de incidencias	Se podrá, a partir de una incidencia existente, crear una idéntica, con la salvedad de las intervenciones, adjuntos y relaciones de la misma
RF5.8	Generación de informes	Se habilitará una función para la generación del informe de una incidencia en formato imprimible desde el propio navegador web
RF5.9	Listado de incidencias	Se podrá consultar un listado con todas las incidencias existentes en el sistema. Esta vista mostrará los siguientes campos para cada incidencia: • Identificador • Título • Usuario informador • Usuario al que se asignó

RF5 - Requisitos del Módulo de Gestión de Incidencias		
Código	**Nombre Requisito**	**Descripción del Requisito**
		• Cliente involucrado • Contrato al que hace referencia • Visibilidad • Tipo de incidencia • Modo de realización • Prioridad de intervención • Fecha de alta • Tiempo estimado para su resolución • Tiempo dedicado • Estado actual
RF5.10	Filtrado de registros de incidencias	La información presentada se podrá filtrar en base a los siguientes criterios: • Búsqueda en base a texto • Búsqueda por intervalo de fecha de alta • Usuario informador • Usuario responsable • Cliente involucrado • Contrato al que se refiere • Visibilidad • Tipo • Modo de realización • Prioridad • Tiempo estimado • Tiempo dedicado • Estado Se podrá además seleccionar la cantidad de registros a mostrar y paginar los resultados en caso de existir más de los visualizados
RF5.11	Ordenación manual de registros	Los registros de incidencias mostrados en el listado serán ordenables en base a los campos que los componen
RF5.11.1	Primer orden de priorización automática	Se mostrarán primero las incidencias asignadas al usuario en sesión de forma automática
RF5.11.2	Segundo orden de priorización automático	En caso de existir más de una incidencia asignada al usuario, se mostrarán primero las que tengan el tipo de prioridad más alto de forma automática

Tabla 5.5. Requisitos del Módulo de Gestión de Incidencias

5.2.1.3 Requisitos No Funcionales

5.2.1.3.1 Requisitos de Usuario

RNF1 - Requisitos de Usuario		
Código	**Nombre Requisito**	**Descripción del Requisito**
RNF1.1	Conocimientos previos	El usuario no precisará de ningún requisito de formación especial para utilizar la aplicación, si bien es necesario que pueda desenvolverse con soltura en el uso de otras aplicaciones web, tales como correo electrónico o herramientas colaborativas
RNF1.2	Pertenencia a la plantilla	El usuario deberá formar parte de la plantilla de la empresa y obtener un identificador para acceder al sistema

Tabla 5.6. Requisitos de Usuario

5.2.1.3.2 Requisitos de la Organización

RNF2 - Requisitos de la Organización		
Código	**Nombre Requisito**	**Descripción del Requisito**
RNF2.1	Definición de departamentos	La empresa deberá definir el conjunto de departamentos que la integran y los empleados que forman parte de cada uno
RNF2.2	Definición de roles y permisos	La empresa debe estipular qué empleados deben disponer de qué permisos y el nivel de privilegios que precisa cada uno
RNF2.3	Definición de responsabilidades de nivel superior	Si bien todos los empleados deberían tener acceso al Módulo de Incidencias, se deber definir un subconjunto de los mismos que actúe como supervisor o encargado para la gestión de entidades de nivel superior, como clientes, contratos y servicios

Tabla 5.7. Requisitos de la Organización

5.2.1.3.3 Requisitos de Seguridad

RNF3 - Requisitos de Seguridad		
Código	**Nombre Requisito**	**Descripción del Requisito**
RNF3.1	Acceso protegido al sistema mediante credenciales	Cada usuario tendrá asignado un identificador único y una contraseña de acceso al sistema
RNF3.2	Comunicaciones seguras	Debe garantizarse la privacidad e integridad de la comunicación entre cliente y servidor
RNF3.3	Integridad de la base de datos	Se procurarán los mecanismos necesarios para garantizar la integridad de la base de datos ante eventuales fallos en las operaciones que se realicen sobre la misma

Tabla 5.8. Requisitos de Seguridad

5.2.1.3.4 Requisitos Tecnológicos

RNF4 - Requisitos Tecnológicos		
Código	**Nombre Requisito**	**Descripción del Requisito**
RNF4.1	Sistema operativo del servidor de bases de datos	El sistema operativo para la máquina que alojará el SGBD será Microsoft Windows Server 2008 Standard R2
RNF4.2	Sistema de Gestión de Bases de Datos (SGBD)	Se empleará el SGBD Microsoft SQL Server 2005, implementado en la empresa en la actualidad sobre la máquina mencionada en RNF4.1
RNF4.3	Sistema operativo del servidor de aplicaciones	La máquina que alojará la aplicación contará con el sistema operativo Ubuntu Server 12.04 LTS
RNF4.4	Servidor web y contenedor de servlets/JSPs	Se utilizará Apache Tomcat 7 como servidor de despliegue de la aplicación sobre la máquina mencionada en RNF4.3
RNF4.5	Plataforma Java EE	Se empleará la plataforma Java Enterprise Edition 7
RNF4.6	Cliente – Conexión a Internet	El equipo del usuario deberá contar con una conexión a Internet funcional, o bien una conexión a la LAN si se usa desde la empresa
RNF4.7	Cliente – Navegador web	El usuario podrá utilizar la aplicación desde, al menos, los siguientes navegadores: • Google Chrome v.24 • Mozilla Firefox v.20

Tabla 5.9. Requisitos Tecnológicos

5.2.1.3.5 Requisitos de Usabilidad

RNF5 - Requisitos de Usabilidad		
Código	**Nombre Requisito**	**Descripción del Requisito**
RNF5.1	Diseño sencillo	Se procurará un diseño y presentación de la información austero, reduciendo al mínimo necesario el número de elementos que precise cada vista de la aplicación y diversificando aquellas que resulten sobrecargadas
RNF5.2	Uso de colores seguros	Se procurará que los colores empleados en la aplicación sean seguros, aunque no se conoce ningún caso de daltonismo entre los empleados por el momento

Tabla 5.10. Requisitos de Usabilidad

5.2.2 Identificación de Actores del Sistema

Se han identificado los siguientes actores relacionados con el sistema, de acuerdo a la siguiente jerarquía:

- **Usuario anónimo:** cualquier usuario que no se haya autenticado en el sistema. Solamente podrá realizar dicha acción.
- **Personal:** el personal de la empresa tendrá poder para realizar aquellas acciones para las que haya sido autorizado en su perfil. Es por ello que empleados de diferentes perfiles podrán llevar a cabo diferentes acciones, de acuerdo a los módulos para los que se habilite su acceso y los permisos de lectura y/o escritura que posea.
    - **Encargado:** los usuarios supervisores o encargados son un subconjunto del grupo de personal. Tendrán autoridad para llevar a cabo procesos de administración ligeros, tales como manejo de contratos, de servicios y de tarifas, además de todas las operaciones permitidas por definición para su perfil.
        - **Administrador:** el administrador de la aplicación tendrá control total sobre todas las operaciones permitidas en el sistema.

5.2.3 Especificación de Casos de Uso

En esta sección se presentan los diferentes casos de uso extraídos a partir de los requisitos funcionales del sistema descritos anteriormente.

5.2.3.1 Gestión de Usuarios

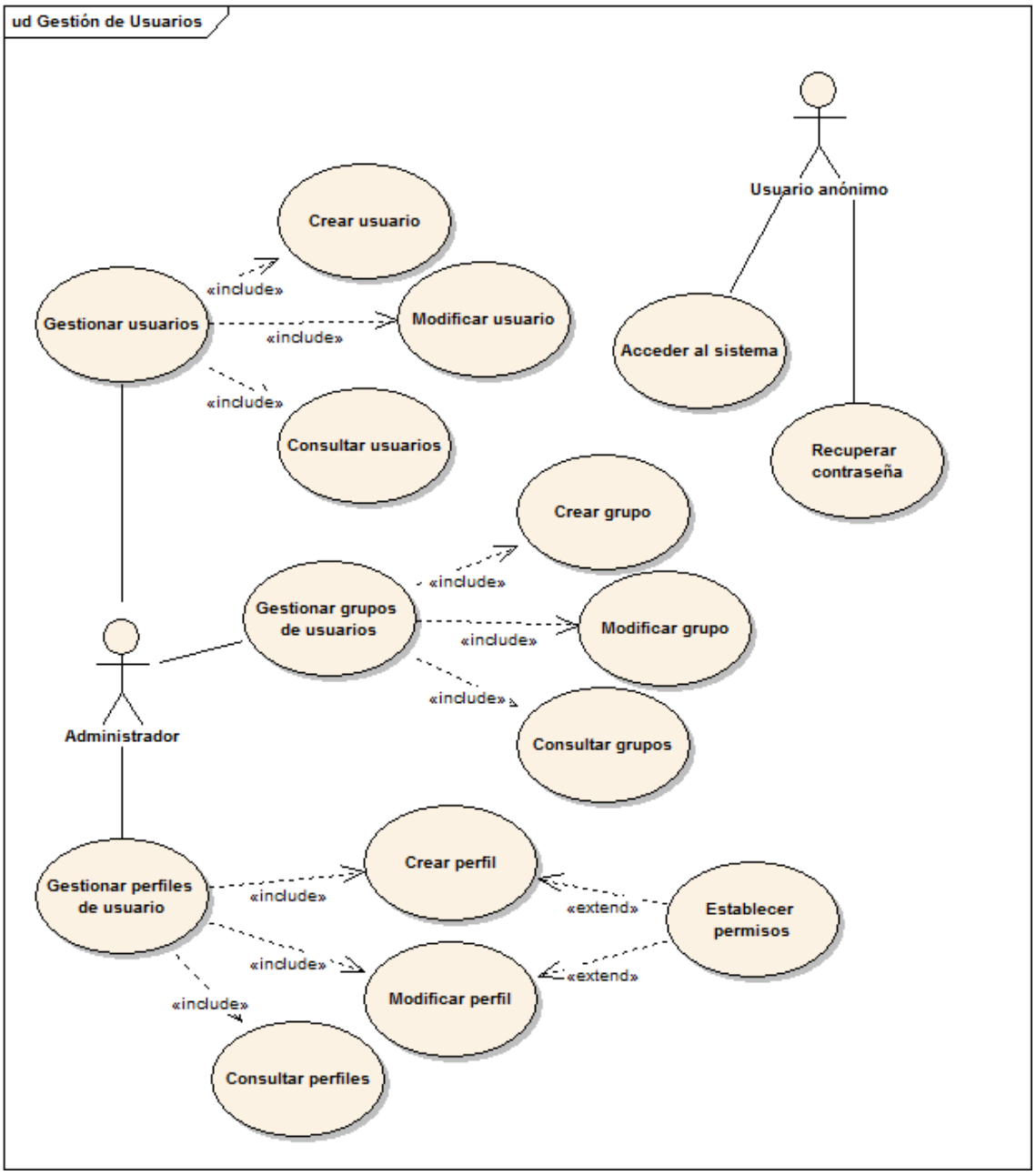


Figura 5.2. Diagrama de Casos de Uso - Gestión de Usuarios

Nombre del Caso de Uso
Crear usuario
Descripción
El administrador creará un nuevo usuario en el sistema. Para ello, proveerá de los datos necesarios para ello

Nombre del Caso de Uso
Modificar usuario
Descripción

Será posible modificar los datos almacenados de un usuario existente en el sistema

Nombre del Caso de Uso
Consultar usuarios
Descripción
Se podrá visualizar un listado con los usuarios actualmente dados de alta en el sistema

Nombre del Caso de Uso
Crear grupo
Descripción
El administrador podrá crear nuevos grupos de usuarios proporcionando una descripción del grupo y seleccionando aquellos usuarios que pertenecerán al mismo

Nombre del Caso de Uso
Modificar grupo
Descripción
Se podrán editar tanto la descripción del grupo de usuarios como los miembros que pertenecen al mismo

Nombre del Caso de Uso
Consultar grupos
Descripción
Se podrá visualizar un listado con los grupos de usuarios existentes en el sistema y el nombre de los miembros de cada uno

Nombre del Caso de Uso
Crear perfil
Descripción
El administrador podrá definir nuevos perfiles en el sistema. Para ello, proporcionará una descripción del perfil y seleccionará los módulos a los que otorga permiso de acceso, además de permisos de modificación, si es preciso

Nombre del Caso de Uso
Modificar perfil
Descripción
Se podrá modificar la descripción de un perfil, así como los módulos y permisos a los que afecta

Nombre del Caso de Uso
Consultar perfiles
Descripción
Se podrá visualizar un listado con los perfiles de usuario existentes en el sistema, los miembros de dicho perfil y los módulos a los que da acceso

Nombre del Caso de Uso
Acceder al sistema
Descripción
Un usuario anónimo (sin identificar) podrá acceder a la pantalla de login del sistema y autenticarse con su nombre de usuario y contraseña

Nombre del Caso de Uso
Recuperar contraseña
Descripción
Un usuario sin identificar que no recuerde su contraseña de acceso al sistema, podrá solicitar una nueva introduciendo su nombre de usuario. Las instrucciones para restablecer la contraseña serán enviadas a la dirección de correo electrónico que facilitó cuando se creó su cuenta de usuario

5.2.3.2 Gestión de Clientes

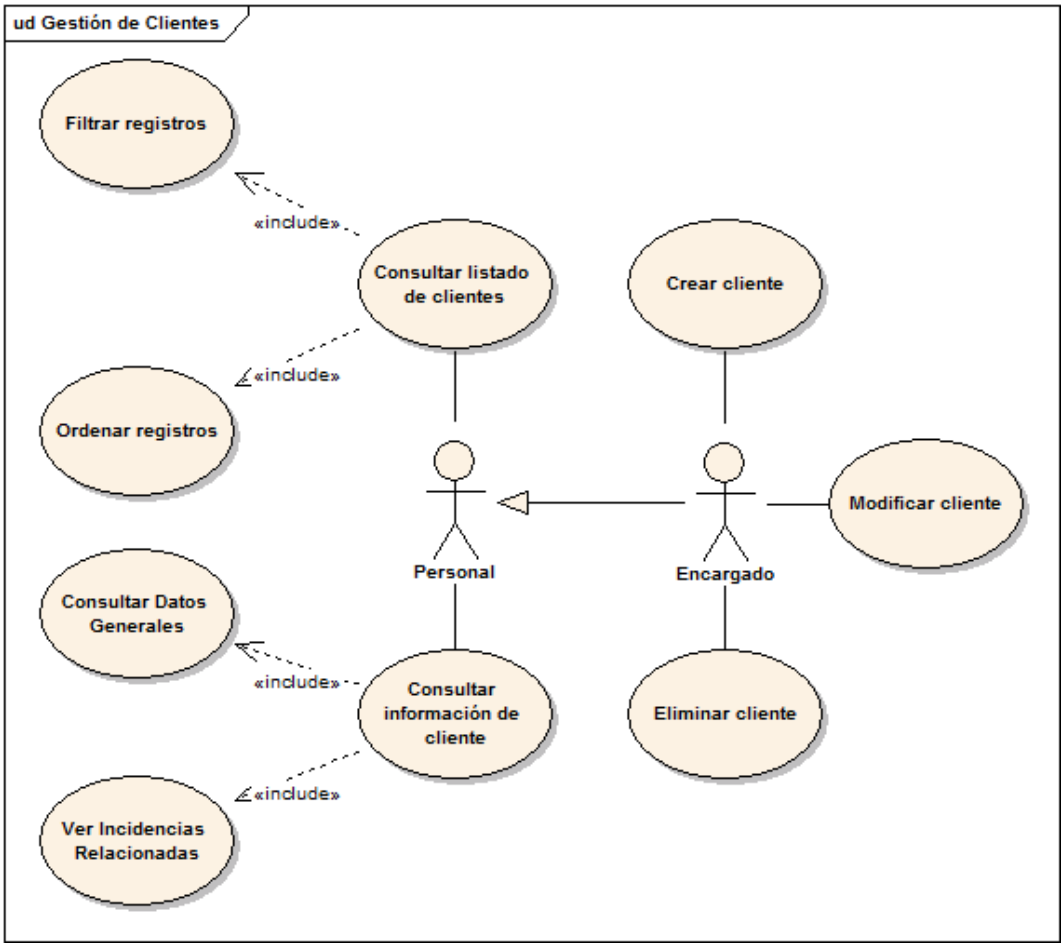


Figura 5.3. Diagrama de Casos de Uso - Gestión de Clientes

Nombre del Caso de Uso
Crear cliente
Descripción
Un encargado podrá dar de alta un nuevo cliente en el sistema. Para ello, proporcionará los datos necesarios por medio de un formulario

Nombre del Caso de Uso
Modificar cliente
Descripción
Podrá editarse la información disponible de un cliente existente en el sistema

Nombre del Caso de Uso
Eliminar cliente
Descripción
Podrá efectuarse la baja de un cliente, previa confirmación de la acción

Nombre del Caso de Uso
Consultar listado de clientes
Descripción
Se podrá consultar un listado con todos los clientes dados de alta en el sistema

Nombre del Caso de Uso
Filtrar registros de clientes
Descripción
Se podrá filtrar la información presentada en el listado de clientes en base a un filtro basado en texto, además de en base a su población y provincia

Nombre del Caso de Uso
Ordenar registros de clientes
Descripción
El listado de clientes se podrá ordenar de acuerdo a los campos que componen cada entrada

Nombre del Caso de Uso
Consultar detalle del cliente
Descripción
Se podrá consultar toda la información disponible referente a un cliente

Nombre del Caso de Uso
Consultar datos generales del cliente
Descripción
Se mostrará una vista con los datos personales del cliente, así como su logotipo

Nombre del Caso de Uso
Ver incidencias del cliente
Descripción
Se mostrará un enlace directo al módulo de Gestión de Incidencias, donde se aplicará automáticamente un filtro para mostrar solamente las incidencias relacionadas con el cliente en cuestión

5.2.3.3 Gestión de Contratos

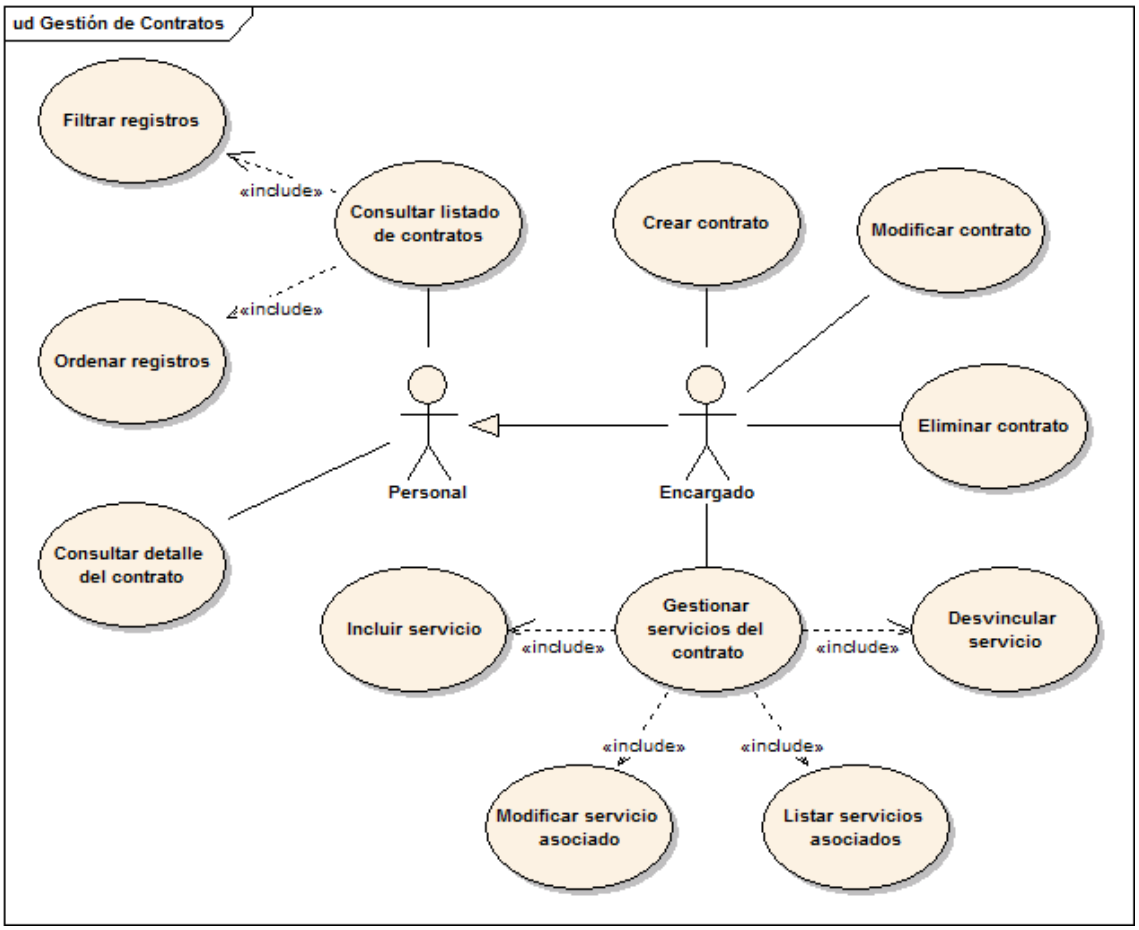


Figura 5.4. Diagrama de Casos de Uso - Gestión de Contratos

Nombre del Caso de Uso
Crear contrato
Descripción
Un encargado podrá crear nuevos contratos suscritos con los clientes proporcionando todos los datos de los mismos

Nombre del Caso de Uso
Modificar contrato
Descripción
Se podrán editar todas las propiedades de un contrato existente, salvo el cliente asociado al mismo

Nombre del Caso de Uso
Eliminar contrato
Descripción
Se podrán eliminar aquellos contratos que dejen de tener validez, previa confirmación de la acción

Nombre del Caso de Uso
Gestionar servicios del contrato
Descripción
Se permitirá gestionar los servicios asociados a un contrato suscrito por un cliente

Nombre del Caso de Uso
Incluir servicio
Descripción
Se podrá asociar un nuevo servicio a un contrato existente proporcionando los datos necesarios para ello

Nombre del Caso de Uso
Modificar servicio asociado
Descripción
A partir de un servicio ya asociado a un contrato, se permitirá editar sus propiedades

Nombre del Caso de Uso
Desvincular servicio
Descripción
Se podrá desvincular un servicio asociado a un contrato determinado, previa confirmación de la acción

Nombre del Caso de Uso
Listar servicios asociados
Descripción
Se mostrará una vista con todos los servicios asociados a un contrato determinado

Nombre del Caso de Uso
Consultar listado de contratos
Descripción
Se podrá consultar un listado con todos los contratos suscritos por los clientes

Nombre del Caso de Uso
Filtrar registros de contratos
Descripción
Se podrán filtrar los registros de contratos mostrados en la vista de listado de contratos en base a un filtro de texto, a los activos a una fecha concreta, al nombre del cliente y al tipo de facturación aplicado al contrato

Nombre del Caso de Uso
Ordenar registros de contratos

Descripción
Se podrán ordenar los registros de contratos en base a los campos mostrados en el listado

Nombre del Caso de Uso
Consultar detalle del contrato
Descripción
Se podrá consultar el detalle del contrato, mostrando en una vista individual todas sus propiedades y servicios asociados

5.2.3.4 Gestión de Servicios

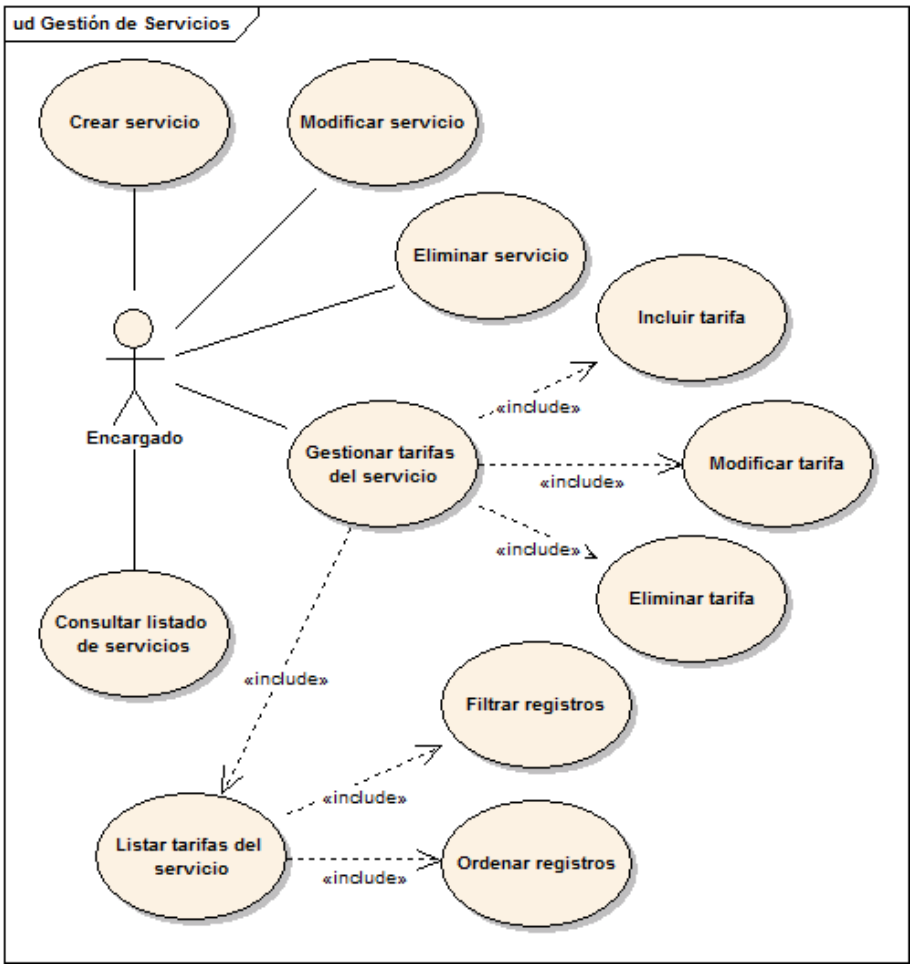


Figura 5.5. Diagrama de Casos de Uso - Gestión de Servicios

Nombre del Caso de Uso
Crear servicio
Descripción
El encargado podrá añadir un nuevo servicio al catálogo proporcionando su descripción

Nombre del Caso de Uso
Modificar servicio
Descripción
Se podrá modificar la descripción de los servicios existentes en el catálogo

Nombre del Caso de Uso
Eliminar servicio
Descripción
Cuando un servicio deja de estar disponible, podrá ser eliminado del catálogo de servicios, previa confirmación de la acción

Nombre del Caso de Uso
Consultar listado de servicios
Descripción
Se podrá consultar el catálogo de servicios ofrecidos

Nombre del Caso de Uso
Gestionar tarifas del servicio
Descripción
Podrán gestionarse las tarifas que componen cada servicio

Nombre del Caso de Uso
Incluir tarifa
Descripción
Podrán incluirse nuevas tarifas asociadas a los servicios proporcionando su nombre, precio y periodo de validez

Nombre del Caso de Uso
Modificar tarifa
Descripción
Podrán editarse las propiedades de las tarifas ya existentes asociadas a los servicios

Nombre del Caso de Uso
Eliminar tarifa
Descripción
Las tarifas asociadas a servicios podrán ser eliminadas, previa confirmación de la acción

Nombre del Caso de Uso
Listar tarifas del servicio
Descripción
Podrán consultarse todas las tarifas asociadas a cada servicio de los disponibles en el catálogo

Nombre del Caso de Uso
Filtrar registros de tarifas
Descripción
Podrán filtrarse los registros de tarifas de cada servicio de acuerdo a un filtro de texto que buscará en base al nombre de la tarifa

Nombre del Caso de Uso
Ordenar registros de tarifas
Descripción

Las tarifas de un servicio podrán ser ordenadas de acuerdo a los campos que las componen

5.2.3.5 Gestión de Incidencias

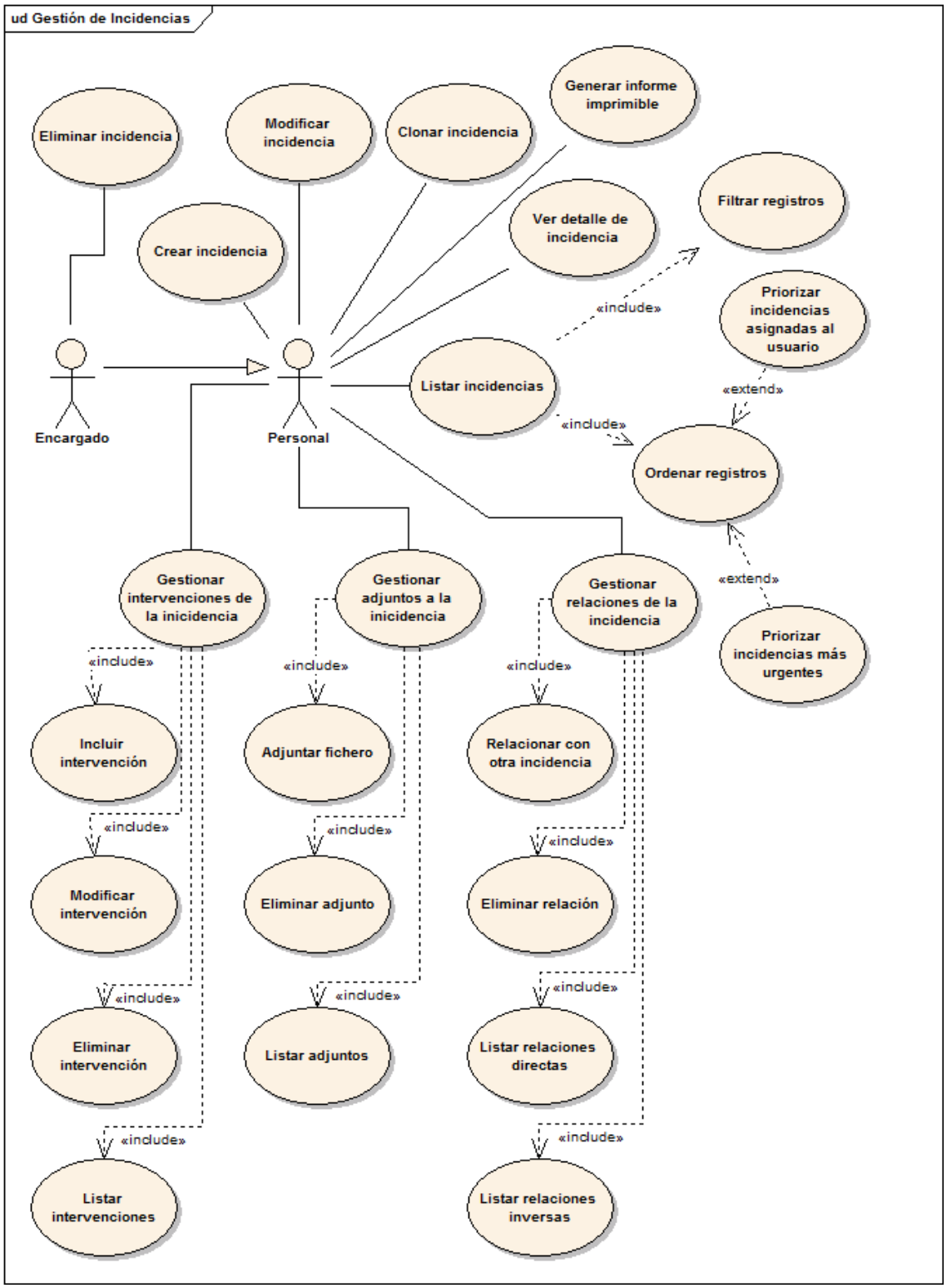


Figura 5.6. Diagrama de Casos de Uso - Gestión de Incidencias

Nombre del Caso de Uso
Crear incidencia
Descripción
Cualquier miembro del personal con permisos en el módulo de incidencias podrá dar de alta una nueva incidencia en el sistema que le haya sido comunicada

Nombre del Caso de Uso
Modificar incidencia
Descripción
Será posible modificar los datos descriptivos de las incidencias existentes

Nombre del Caso de Uso
Eliminar incidencia
Descripción
Será posible eliminar cualquier incidencia de las existentes en el sistema, previa confirmación de la acción. Esta operación queda reservada a usuarios con privilegios de encargado

Nombre del Caso de Uso
Ver detalle de incidencia
Descripción
Se podrá consultar una vista con todos los detalles de la incidencia, incluidos intervenciones, adjuntos y relaciones, tanto directas como inversas

Nombre del Caso de Uso
Clonar incidencia
Descripción
Será posible crear una nueva incidencia a partir de una ya existente. Se copiarán todos los datos de la incidencia original, salvo sus intervenciones, adjuntos y relaciones

Nombre del Caso de Uso
Generar informe imprimible
Descripción
Será posible generar e imprimir un informe detallado de cada incidencia, con un formato similar al presentado en la aplicación, pero adecuándolo a una vista de impresión

Nombre del Caso de Uso
Listar incidencias
Descripción
Se podrá consultar un listado con todas las incidencias existentes en el sistema

Nombre del Caso de Uso
Filtrar registros
Descripción
Se podrán filtrar los registros de incidencias mostrados de acuerdo a un filtro basado en texto, su fecha de alta (en intervalo), el usuario que informó de la misma, el usuario al que fue asignada, el cliente, el contrato, la visibilidad, el tipo de incidencia, el modo de realización, la prioridad y su estado actual

Nombre del Caso de Uso
Ordenar registros
Descripción
Se podrán ordenar los registros de incidencias de acuerdo a los campos mostrados de cada una. Además, se priorizarán automáticamente aquellas incidencias que han sido asignadas al usuario en sesión y las que cuentan con un tipo de prioridad más alto

Nombre del Caso de Uso
Gestionar intervenciones de la incidencia
Descripción
Se podrán gestionar completamente las intervenciones realizadas sobre cada incidencia de forma individual

Nombre del Caso de Uso
Incluir intervención
Descripción
Cuando un empleado realiza una intervención en una incidencia, será posible anotar los datos relativos a dicha actuación

Nombre del Caso de Uso
Modificar intervención
Descripción
Será posible modificar los datos de cada una de las intervenciones realizadas a una incidencia

Nombre del Caso de Uso
Eliminar intervención
Descripción
Será posible eliminar cada una de las intervenciones de la incidencia de forma individual, previa confirmación de la acción

Nombre del Caso de Uso
Listar intervenciones
Descripción
Se proporcionará una vista en el detalle de la incidencia con un listado de las intervenciones llevadas a cabo en la misma

Nombre del Caso de Uso
Gestionar adjuntos a la incidencia
Descripción
Se podrán gestionar completamente los ficheros adjuntos en las incidencias que los precisen

Nombre del Caso de Uso
Adjuntar fichero
Descripción
Será posible adjuntar un fichero con información de interés para el tratamiento de la incidencia. Este fichero podrá ser, en principio, de cualquier tipo y tamaño

Nombre del Caso de Uso
Eliminar adjunto
Descripción
Será posible eliminar los adjuntos de una incidencia que no sean necesarios para su tratamiento, previa confirmación de la acción

Nombre del Caso de Uso
Listar adjuntos
Descripción
Se proporcionará una vista en el detalle de la incidencia con el listado de ficheros adjuntos a la misma. Existirá además un enlace en cada uno de los adjuntos listados para su descarga

Nombre del Caso de Uso
Gestionar relaciones de la incidencia
Descripción
Podrán gestionarse todas las relaciones habidas entre la incidencia visualizada y el resto de las existentes, tanto directa como inversamente relacionadas

Nombre del Caso de Uso
Relacionar con otra incidencia
Descripción
Será posible indicar una relación entre la incidencia mostrada y otra existente en el sistema. Para ello, será necesario proveer el tipo de relación que guardan y el identificador de la incidencia relacionada

Nombre del Caso de Uso
Eliminar relación
Descripción
En caso de precisarlo, será posible eliminar una relación de la incidencia establecida anteriormente, previa confirmación de la acción

Nombre del Caso de Uso
Listar relaciones directas
Descripción
Se mostrará un listado con las incidencias con las que la presente está directamente relacionada, permitiendo acceder mediante un enlace a la incidencia en cuestión

Nombre del Caso de Uso
Listar relaciones inversas
Descripción
Se mostrará un listado con las incidencias que guardan relación con la presente, permitiendo acceder mediante un enlace a la incidencia en cuestión

5.3 Identificación de los Subsistemas en la Fase de Análisis

El objetivo de esta sección es analizar el sistema para poder descomponerlo en sistemas más pequeños (subsistemas) que faciliten su posterior análisis.

5.3.1 Descripción de los Subsistemas

La aplicación estará compuesta por varios subsistemas, cada uno de los cuales implementará una parte concreta de la funcionalidad del sistema, los cuales son descritos a continuación de forma individual.

5.3.1.1 Subsistema de Gestión de Usuarios

Tiene como objetivo proveer de los mecanismos necesarios para identificar a los usuarios de la aplicación, además de servir como herramienta de administración para la creación, modificación y eliminación de usuarios, perfiles y grupos.

Su desarrollo no será competencia de este proyecto, sino que se hará uso de su funcionalidad importándolo al mismo como librería externa.

5.3.1.2 Subsistema de Gestión de Clientes

Su objetivo es ofrecer la funcionalidad necesaria para gestionar la cartera de clientes de la empresa. Desde este subsistema, el usuario podrá dar de alta nuevos clientes, modificar los datos de los existentes y eliminar aquellos que ya no lo sean, además de poder consultar sus datos, formas de contacto e incidencias relacionadas.

5.3.1.3 Subsistema de Gestión de Contratos

La parte de la aplicación encargada de la gestión de los contratos suscritos por los clientes con la empresa contendrá las funciones necesarias para dar de alta nuevos contratos, modificar los existentes y eliminar aquellos que fueran rescindidos.

Además de esto, desde el subsistema de gestión de contratos, el usuario podrá visualizar los detalles de cada contrato almacenado, incluidos los servicios que formen parte de dicho acuerdo, permitiéndole gestionarlos a su discreción.

5.3.1.4 Subsistema de Gestión de Servicios

Desde el subsistema de gestión de servicios, el usuario de la aplicación tendrá la oportunidad de manejar el catálogo de servicios ofrecidos por la empresa. Es por ello que

tendrá a su disposición las opciones de incorporar nuevos servicios al catálogo, modificar los ya existentes y eliminar aquellos que se dejen de ofrecer a los clientes para su contratación.

Dado que cada servicio tendrá asociadas unas tarifas determinadas, en este subsistema se contemplarán además todas las operaciones necesarias para la gestión de las mismas.

Será posible, como en el resto de subsistemas, visualizar un listado y detalle de todas las entidades gestionables desde este subsistema.

5.3.1.5 Subsistema de Gestión de Incidencias

Concebido como el núcleo de la aplicación, el subsistema de gestión de incidencias abarcará toda la lógica de negocio particular del proceso de tratamiento de incidencias en la empresa. Para ello, implementará las operaciones de alta, baja y modificación de incidencias y lo que es más importante, la parte de visualización y consulta de las mismas. Se trata de un subsistema complejo, en el que será muy importante cumplir al pie de la letra las particularidades del proceso descritas por la empresa.

Constará además de funcionalidades avanzadas, no vistas hasta ahora en los demás subsistemas de la aplicación, como lo son la gestión de ficheros adjuntos, intervenciones y relaciones entre incidencias.

5.3.2 Descripción de los Interfaces entre Subsistemas

La comunicación entre los subsistemas que componen la aplicación se rige en base a dos puntos clave

- La comunicación entre la parte de modelo-lógica y presentación viene dada por los mecanismos proporcionados por Struts, además de la arquitectura en capas del patrón Modelo-Vista-Controlador que proveerán al sistema de todo lo necesario para ello, haciendo uso de los Servlets, que se encargarán de atender las peticiones recibidas desde el navegador e invocar la ejecución de los Actions encargados de su correcto procesamiento.
- En lo referente a la comunicación de los datos que maneja la aplicación viene dada por la capa de persistencia de la misma, mediante el paso de parámetros en Java y su posterior entrada o salida al sistema desde y hacia la base de datos por medio de JDBC, a través del protocolo IP, puesto que la base de datos estará alojada en una máquina independiente.

Además de esto, el subsistema de gestión de usuarios, o más concretamente la intranet, será el encargado de identificar al usuario y mantener sus datos en la sesión por medio de una *cookie*.

5.4 Identificación de Clases Preliminar del Análisis

A continuación se presenta el diagrama de clases preliminar del análisis. Por simplicidad y para facilitar su lectura y comprensión, se han omitido en la figura los métodos y atributos propuestos para cada clase. Además, aquellos elementos comunes a todos los paquetes, que se repiten por motivo de la arquitectura de Struts, se han sintetizado en un único elemento de cada tipo, sirviendo así al lector como referencia del componente para cada subsistema.

Este diagrama no debe interpretarse como un diseño final de la arquitectura de clases y paquetes del sistema, puesto que, habiéndolo realizado en la fase de análisis, es muy probable que surjan diferentes eventualidades que obliguen a redefinir alguna clase o componente de los aquí mostrados.

Es importante resaltar nuevamente que el paquete de gestión de usuarios no corresponde al ámbito de este desarrollo, pero se muestra en este diagrama para poder ofrecer una visión global del sistema, incluido este elemento del que se hace uso de forma externa.

5.4.1 Diagrama de Clases

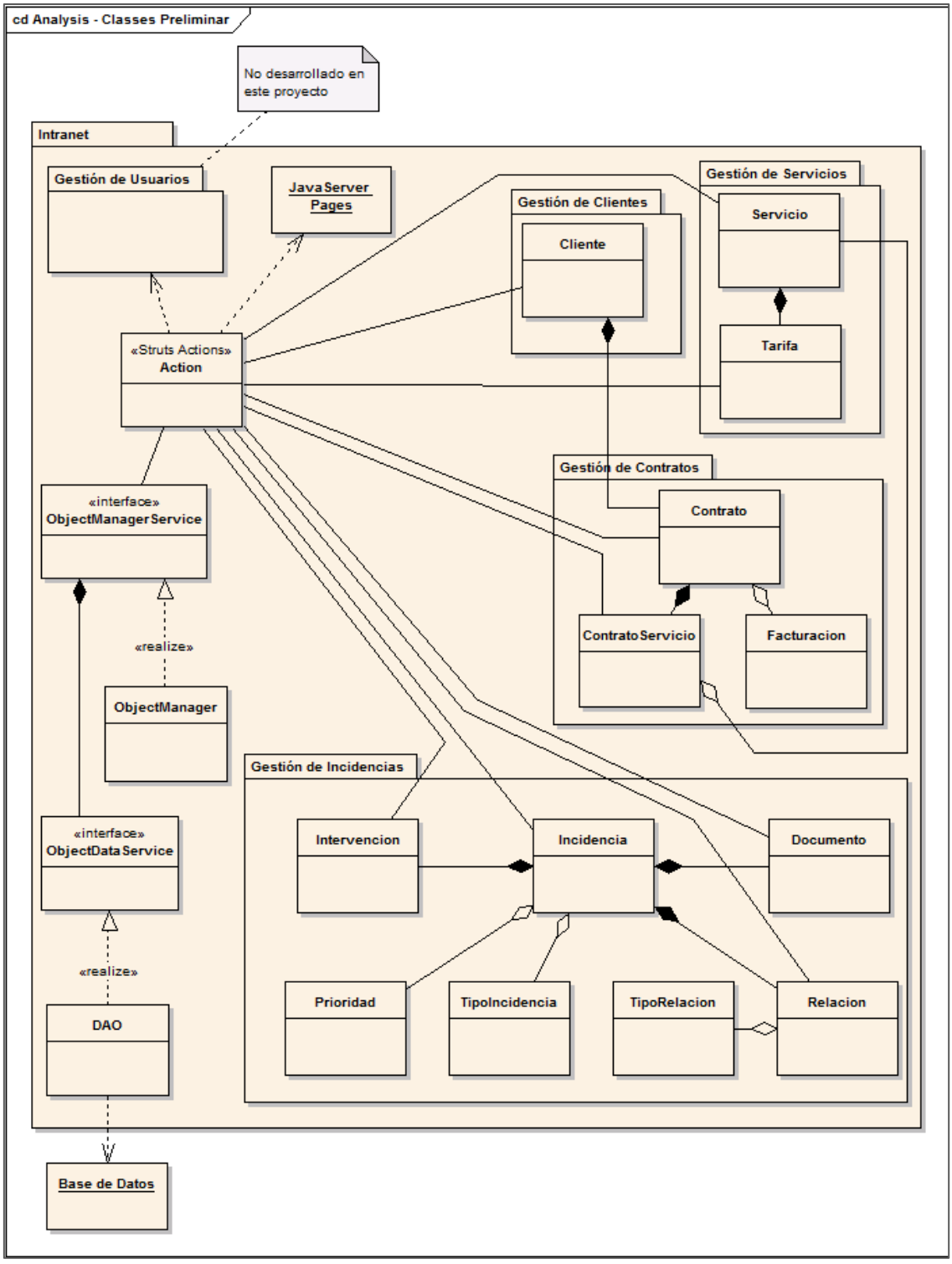


Figura 5.7. Diagrama de Clases Preliminar

5.4.2 Descripción de las Clases

La siguiente descripción define las clases mostradas en el diagrama anterior pertenecientes a cada subsistema. Debe tomarse en consideración que las clases (Actions, ObjectManagers y DAOs) e interfaces (ObjectManagerServices y DataManagerServices) que agrupan funciones similares en cada paquete solamente serán descritas una vez, para evitar redundancias innecesarias. En el capítulo posterior se detallarán cada una de ellas de forma individual.

5.4.2.1 Subsistema de Gestión de Usuarios

Este subsistema se encuentra fuera del plan de desarrollo del proyecto. Su función es la de proporcionar las operaciones de gestión de los usuarios, grupos y perfiles del sistema. No se dispone de la información relativa a la descripción de sus clases, por lo que no será presentada aquí.

5.4.2.2 Subsistema de Gestión de Clientes

Nombre de la Clase
Cliente
Descripción
Representa el modelo de una entidad cliente dentro del sistema
Responsabilidades
Contiene todos los datos propios de un cliente y ofrece los métodos necesarios para la lectura y modificación de sus atributos
Atributos Propuestos
Nombre: cadena de texto que almacena el nombre identificativo del cliente **DireccionPostal**: cadena de texto que almacena la ubicación física del cliente (calle, número y código postal) **Poblacion**: cadena de texto que recoge la población del cliente **Provincia**: cadena de texto que recoge la provincia del cliente **Teléfono**: cadena de texto que almacena el teléfono de contacto del cliente **Fax**: cadena de texto que almacena el número de fax del cliente (opcional) **Logo**: cadena de texto que almacena la identificación del fichero de imagen con el logotipo corporativo del cliente
Métodos Propuestos
Ninguno destacable, además de los *getters* y *setters* para los atributos

5.4.2.3 Subsistema de Gestión de Contratos

Nombre de la Clase
Contrato
Descripción
Representa el modelo de una entidad contrato dentro del sistema
Responsabilidades
Almacena toda la información relativa a un contrato suscrito por un cliente y ofrece los

métodos necesarios para la lectura y modificación de sus atributos
Atributos Propuestos
IDContrato: número entero utilizado como identificador único de cada contrato **IDCliente**: número entero utilizado como referencia al identificador único del cliente que suscribe el contrato **IDTipoFacturacion**: número entero utilizado como referencia al identificador único del tipo de facturación aplicado al contrato **Titulo**: cadena de texto que almacena el nombre identificativo del contrato **Descripcion**: cadena de texto que recoge una descripción breve del fundamento del contrato **FechaAlta**: marca temporal que representa la fecha en la que se celebra el contrato **FechaInicio**: marca temporal que representa la fecha en la que el contrato se hace efectivo **FechaFin**: marca temporal que representa la fecha en la que el contrato deja de tener validez **Estado**: cadena de texto que representa la fase en la que se encuentra el contrato
Métodos Propuestos
Ninguno destacable, además de los *getters* y *setters* para los atributos

Nombre de la Clase
ContratoServicio
Descripción
Representa la relación generada con la asociación de un servicio a un contrato
Responsabilidades
Almacenar la información relativa a los servicios que se asocian a cada contrato y las condiciones pactadas para dicha asociación
Atributos Propuestos
IDContrato: número entero utilizado como identificador único del contrato al que se asociarán los servicios **IDServicio**: número entero utilizado como identificador único del servicio a asociar al contrato **Horas**: número de horas/hombre acordadas para la prestación del servicio **PrecioHora**: número decimal que representa el precio, expresado en euros, que se cobrará por cada hora-hombre contratada para el servicio **Desplazamiento**: booleano que representa si el desplazamiento al cliente para la prestación del servicio está incluido o no en el precio **Descripcion**: cadena de texto que recoge una descripción breve de las particularidades de la asociación del servicio al contrato
Métodos Propuestos
Ninguno destacable, además de los *getters* y *setters* para los atributos

Nombre de la Clase
Facturacion
Descripción
Modela los datos relativos a los métodos de facturación aplicables a los contratos
Responsabilidades
Almacenar la información relativa a cada tipo de facturación para los contratos ofrecido a los clientes
Atributos Propuestos
IDTipoFacturacion: número entero utilizado como identificador único del tipo de facturación **Descripcion**: cadena de texto que recoge las particularidades del tipo de facturación
Métodos Propuestos
Ninguno destacable, además de los *getters* y *setters* para los atributos

5.4.2.4 Subsistema de Gestión de Servicios

Nombre de la Clase
Servicio
Descripción
Modela los datos relativos a los servicios ofrecidos por la empresa
Responsabilidades
Almacenar la información relativa a cada servicio de los disponibles en el catálogo de la empresa
Atributos Propuestos
IDServicio: número entero utilizado como identificador único del servicio **Descripción**: cadena de texto que recoge las propiedades descriptivas del servicio
Métodos Propuestos
Ninguno destacable, además de los *getters* y *setters* para los atributos

Nombre de la Clase
Tarifa
Descripción
Modela los datos relativos a las tarifas de los servicios existentes
Responsabilidades
Contener la información asociada a cada una de las tarifas que se aplican a cada servicio de los ofrecidos por la empresa
Atributos Propuestos
IDTarifa: número entero utilizado como identificador único de la tarifa **IDServicio**: número entero utilizado como referencia al identificador único del servicio al que se asocia la tarifa **FechaInicio**: marca temporal que representa el momento en el que la tarifa comienza a ser aplicable **FechaFin**: marca temporal que representa el momento en el que la tarifa deja de tener validez **Precio**: número decimal que representa el precio por hora, expresado en euros, de la tarifa
Métodos Propuestos
Ninguno destacable, además de los *getters* y *setters* para los atributos

5.4.2.5 Subsistema de Gestión de Incidencias

Nombre de la Clase
Incidencia
Descripción
Modela los datos relativos a las incidencias almacenadas en el sistema
Responsabilidades
Contener la información relativa a cada una de las incidencias que han sido dadas de alta en el sistema, así como las referencias al cliente y contrato al que se asocian
Atributos Propuestos
IDIncidencia: número entero utilizado como identificador único de la incidencia **IDInformador**: número entero utilizado como referencia al identificador único del usuario que

dio de alta la incidencia en el sistema **IDAsignada**: número entero utilizado como referencia al identificador único del usuario al que se asigna el tratamiento de la incidencia **IDCliente**: número entero utilizado como referencia al identificador único del cliente que comunica la incidencia **IDContrato**: número entero utilizado como referencia al identificador único del contrato al que se asocia la incidencia **IDTipoIncidencia**: número entero utilizado como referencia al tipo de incidencia que se trata, en base a su naturaleza **IDPrioridad**: número entero utilizado como referencia al tipo de prioridad de resolución asignada a la incidencia **Titulo**: cadena de texto que representa de forma breve y descriptiva el motivo de la incidencia **Descripcion**: cadena de texto que representa el detalle descriptivo de la incidencia **RealizableRemoto**: booleano que indica si la incidencia es realizable de forma remota o presencial **TiempoEstimado**: número decimal que representa una estimación del tiempo necesario para la resolución de la incidencia **TiempoDedicado**: número decimal que acumula el tiempo empleado en el tratamiento de la incidencia **FechaAlta**: marca temporal que representa el momento en el que la incidencia fue dada de alta en el sistema **Visibilidad**: booleano que representa la privacidad de la incidencia. Puede ser privada o pública **Estado:** cadena de texto que representa la fase en la que se encuentra el tratamiento de la incidencia
Métodos Propuestos
Ninguno destacable, además de los *getters* y *setters* para los atributos

Nombre de la Clase
Intervencion
Descripción
Modela los datos relativos a las intervenciones realizadas en cada incidencia durante su resolución
Responsabilidades
Almacenar los datos asociados a cada una de las intervenciones que se llevan a cabo sobre cada incidencia en su tratamiento
Atributos Propuestos
IDIntervencion: número entero utilizado como identificador único de la intervención **IDUsuario**: número entero utilizado como referencia al identificador único del usuario que realiza la intervención **IDIncidencia**: número entero utilizado como referencia a la incidencia intervenida **Fecha**: marca temporal que representa el momento en el que se llevó a cabo la intervención **Parte**: cadena de texto que hace referencia al parte o albarán producto de la intervención, de haberlo **Horas**: número decimal que acumula el tiempo empleado en la intervención **Motivo**: cadena de texto que indica brevemente el fundamento de la intervención **Observaciones**: cadena de texto que representa el detalle descriptivo de la intervención
Métodos Propuestos
Ninguno destacable, además de los *getters* y *setters* para los atributos

Nombre de la Clase
Documento
Descripción
Modela los datos relativos a los ficheros adjuntos a cada incidencia
Responsabilidades
Almacenar los datos asociados a cada uno de los ficheros adjuntos relevantes para el tratamiento de la incidencia
Atributos Propuestos
IDDocumento: número entero utilizado como identificador único del documento **IDIncidencia**: número entero utilizado como referencia al identificador único de la incidencia a la que se asocia el documento **Descripcion**: cadena de texto que almacena una breve descripción del fichero adjunto **Ruta**: cadena de texto que almacena la ruta del documento dentro del sistema de archivos del servidor
Métodos Propuestos
Ninguno destacable, además de los *getters* y *setters* para los atributos

Nombre de la Clase
Relacion
Descripción
Modela los datos relativos a las relaciones o conexiones entre las incidencias existentes en el sistema
Responsabilidades
Almacenar los datos asociados a cada uno de las vínculos existentes entre una incidencia y aquellas que guardan relación con ella
Atributos Propuestos
IDRelacion: número entero utilizado como identificador único de la relación **IDIncidenciaMain**: número entero utilizado como referencia al identificador único de la incidencia que genera la relación (origen) **IDIncidenciaSub**: número entero utilizado como referencia al identificador único de la incidencia a la que se refiere la relación (destino) **IDTipoRelacion**: número entero utilizado como referencia al tipo de relación que existe entre dos incidencias **IDUsuario**: número entero que hace referencia al identificador único del usuario que estableció la relación entre incidencias
Métodos Propuestos
Ninguno destacable, además de los *getters* y *setters* para los atributos

Nombre de la Clase
Prioridad
Descripción
Modela los datos relativos a cada uno de los tipos de prioridad aplicables a las incidencias
Responsabilidades
Almacenar los datos asociados a cada uno de los tipos de prioridad de intervención definidos en el sistema
Atributos Propuestos
IDPrioridad: número entero utilizado como identificador único del tipo de prioridad

Descripcion: cadena de texto que recoge el nombre descriptivo del tipo de prioridad
Métodos Propuestos
Ninguno destacable, además de los *getters* y *setters* para los atributos

Nombre de la Clase
TipoIncidencia
Descripción
Modela los datos relativos a cada uno de los tipos de incidencia
Responsabilidades
Almacenar los datos asociados a cada uno de los tipos de incidencia definidos en el sistema
Atributos Propuestos
IDTipoIncidencia: número entero utilizado como identificador único del tipo de incidencia **Descripcion**: cadena de texto que recoge el nombre descriptivo del tipo de incidencia
Métodos Propuestos
Ninguno destacable, además de los *getters* y *setters* para los atributos

Nombre de la Clase
TipoRelacion
Descripción
Modela los datos relativos a cada uno de los tipos de relación entre incidencias
Responsabilidades
Almacenar los datos asociados a cada uno de los susceptibles tipos de relación entre incidencias
Atributos Propuestos
IDTipoRelacion: número entero utilizado como identificador único del tipo de relación **Descripcion**: cadena de texto que recoge el nombre descriptivo de la relación
Métodos Propuestos
Ninguno destacable, además de los *getters* y *setters* para los atributos

5.4.2.6 Comunes a todos los subsistemas

Nombre de la Clase
Action
Descripción
Conjunto de clases que representan de forma genérica las diferentes acciones que el usuario ejecuta en la interacción con el sistema. Se encuadran dentro de la aplicación entre el Modelo y la Vista de la misma. Su configuración se lleva a cabo por medio de los ficheros XML designados a tal efecto por el framework Struts2
Responsabilidades
Recibir las peticiones recibidas desde la capa de presentación de la aplicación y trasladarlas a la capa de negocio para resolver la funcionalidad requerida. Proporcionar la lógica de acceso a datos. Inyectar los datos correspondientes en las entidades afectadas. Manejar situaciones anómalas propiciadas por datos incoherentes o errores propios de la

base de datos. Crear un nexo con los métodos de validación de datos de entrada.
Atributos Propuestos
ObjectManagerService: objeto genérico que representa a los managers de los diferentes objetos precisados en cada Action. Se encarga además de la comunicación con la capa de persistencia **Session**: contiene todos los datos de la sesión en curso del usuario. Se utiliza, en ocasiones puntuales, para la comunicación con las JSP
Métodos Propuestos
execute(): método ejecutado por defecto cuando el framework invoca al Action encargado de resolver la funcionalidad de una interacción concreta por parte del usuario. Recoge la información recibida y la transmite a la siguiente capa de la arquitectura por medio del ObjectManagerService **validate()**: método opcional que se ejecutará previamente al execute() para certificar que los datos introducidos por el usuario cumplen los requisitos estipulados. Destacar que principalmente se utilizará validación basada en XML, aunque este método se empleará en aquellas Actions en las que la validación XML no proporcione toda la funcionalidad necesaria

Nombre de la Clase
ObjectManagerService <<interface>>
Descripción
Interfaz que representa todas las operaciones disponibles para un tipo ObjectManager
Responsabilidades
Mantener una colección de métodos en los que se especificará que operaciones estarán disponibles. Su implementación correrá a cargo de las clases que las implementen de forma particular
Atributos Propuestos
Ninguno
Métodos Propuestos
getAllObjects(): retornará la colección de entidades a la que hace referencia **getObjectByID(int)**: retornará un objeto basado en el identificador único del mismo **insertObject(object)**: insertará un nuevo objeto en el sistema **updateObject(object)**: actualizará un objeto existente **removeObject(object)**: marcará como eliminado un objeto existente

Nombre de la Clase
ObjectManager
Descripción
Clase que implementará la interfaz correspondiente para cada tipo de objeto
Responsabilidades
Implementar la interfaz concreta de un tipo de objeto, invocando al DataService correspondiente para completar cada operación
Atributos Propuestos
objectDataService: almacenará una referencia al objeto DataService específico del objeto
Métodos Propuestos
getAllObjects(): retornará la colección de entidades a la que hace referencia **getObjectByID(int)**: retornará un objeto basado en el identificador único del mismo **insertObject(object)**: insertará un nuevo objeto en el sistema

updateObject(object): actualizará un objeto existente **removeObject(object)**: marcará como eliminado un objeto existente

Nombre de la Clase
ObjectDataService <<interface>>
Descripción
Interfaz que representa todas las operaciones disponibles para un tipo DataService
Responsabilidades
Mantener una colección de métodos en los que se especificará que operaciones de lectura y modificación de datos estarán disponibles. Su implementación correrá a cargo de las clases que las implementen de forma particular
Atributos Propuestos
Ninguno
Métodos Propuestos
getAllObjects(): retornará la colección de entidades a la que hace referencia **getObjectByID(int)**: retornará un objeto basado en el identificador único del mismo **insertObject(object)**: insertará un nuevo objeto en el sistema **updateObject(object)**: actualizará un objeto existente **removeObject(object)**: marcará como eliminado un objeto existente

Nombre de la Clase
DAO
Descripción
Conjunto de clases que representan de forma genérica los diferentes objetos de acceso a datos empleados en la comunicación entre la capa de persistencia de la aplicación y la base de datos
Responsabilidades
Implementar los métodos de creación, actualización, borrado y lectura para aquellos objetos que precisen el acceso a la base de datos. Implementar, cuando sea preciso, los métodos de utilidad necesarios para la manipulación de datos de entrada o salida al sistema, de acuerdo a las directrices de formato que se apliquen particularmente a cada uno.
Atributos Propuestos
Ninguno
Métodos Propuestos
getAllObjects(): retornará la colección de entidades a la que hace referencia **getObjectByID(int)**: retornará un objeto basado en el identificador único del mismo **insertObject(object)**: insertará un nuevo objeto en el sistema **updateObject(object)**: actualizará un objeto existente **removeObject(object)**: marcará como eliminado un objeto existente

5.5 Análisis de Casos de Uso y Escenarios

En esta sección se describirán los casos de uso identificados anteriormente de forma detallada, a través de sus escenarios. Los escenarios describen las interacciones entre los usuarios y el sistema e incluyen información acerca de los objetivos, expectativas, motivaciones, acciones y reacciones que se llevan a cabo. La intención de los mismos es definir el comportamiento deseado del *software* de manera que complementen a los requisitos funcionales antes descritos.

Nuevamente, se recuerda que los casos de uso y escenarios relativos al módulo de Gestión de Usuarios serán omitidos por no resultar relevantes para el presente desarrollo, mostrando únicamente aquellos con los que el presente sistema guarda una dependencia directa.

5.5.1 Casos de Uso de la Gestión de Usuarios

5.5.1.1 Iniciar sesión

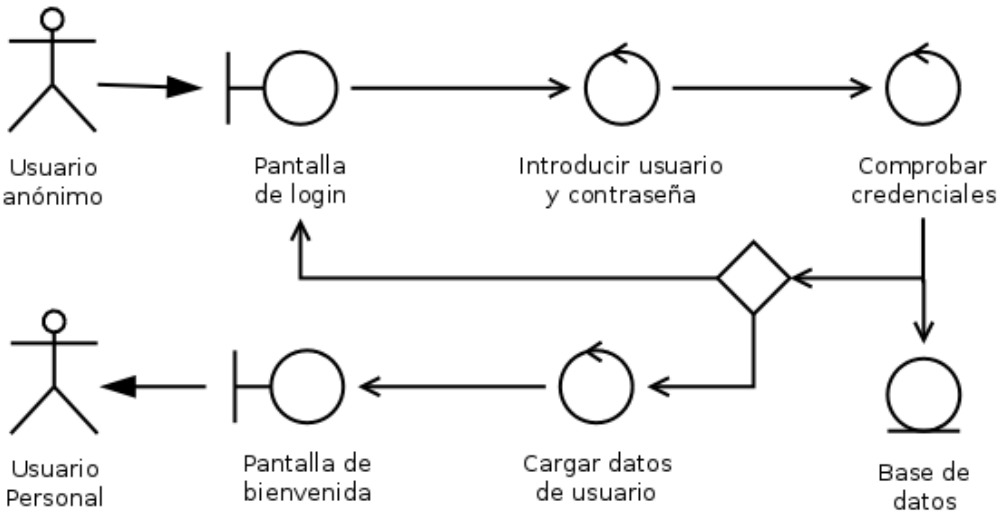


Figura 5.8. Diagrama de Robustez - Iniciar sesión

Iniciar sesión	
Precondiciones	El usuario ha sido dado de alta en el sistema y conoce sus credenciales de acceso
Poscondiciones	El usuario es reconocido y se inicia la sesión
Actores	Usuario anónimo, Usuario Personal
Descripción	Se trata de autenticar al usuario y abrir su sesión de trabajo en el sistema
Variaciones	• **Escenario alternativo 1**: El usuario intenta acceder al sistema sin contar con unas credenciales de acceso válidas ○ Se muestra una error en el formulario de login

5.5.1.2 Cerrar sesión

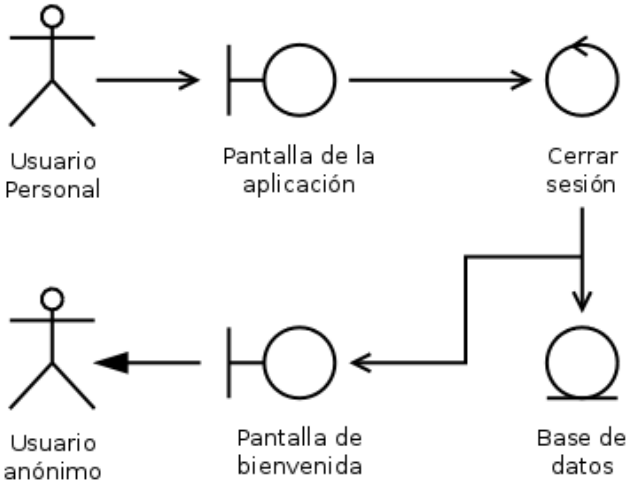


Figura 5.9. Diagrama de Robustez - Cerrar sesión

Cerrar sesión	
Precondiciones	El usuario autenticado se encuentra trabajando en el sistema, con una sesión en curso
Poscondiciones	El usuario es finaliza la sesión correctamente
Actores	Usuario Personal, Usuario anónimo
Descripción	Se trata de finalizar la sesión del usuario, cerrándola de forma ordenada y dejando el sistema en un estado coherente
Variaciones	• **Escenario alternativo 1**: El usuario cierra el navegador sin haber cerrado la sesión ○ La sesión expira automáticamente en el servidor tras 20 minutos de inactividad en la misma

5.5.2 Casos de Uso de la Gestión de Clientes

5.5.2.1 Crear cliente

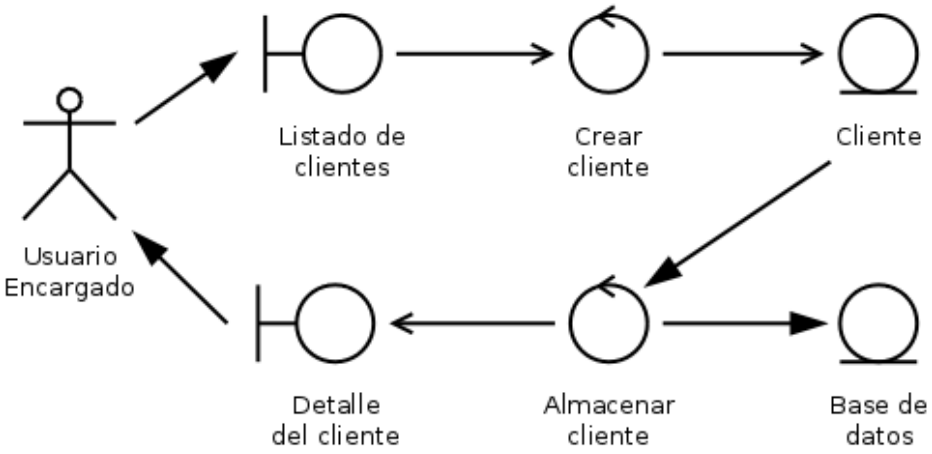


Figura 5.10. Diagrama de Robustez - Crear cliente

Crear cliente	
Precondiciones	El usuario en sesión tiene privilegios de encargado
Poscondiciones	El sistema cuenta con un nuevo cliente
Actores	Usuario encargado
Descripción	Se trata de crear un nuevo cliente en el sistema. Para ello, el encargado completará los datos necesarios para su ficha

Excepciones	• **El formulario de entrada de datos contiene errores de validación**
	  ○ Se informa al usuario con un error localizado
	• **La base de datos no se encuentra disponible**
	  ○ Se informa al usuario con un error descriptivo

5.5.2.2 Modificar cliente

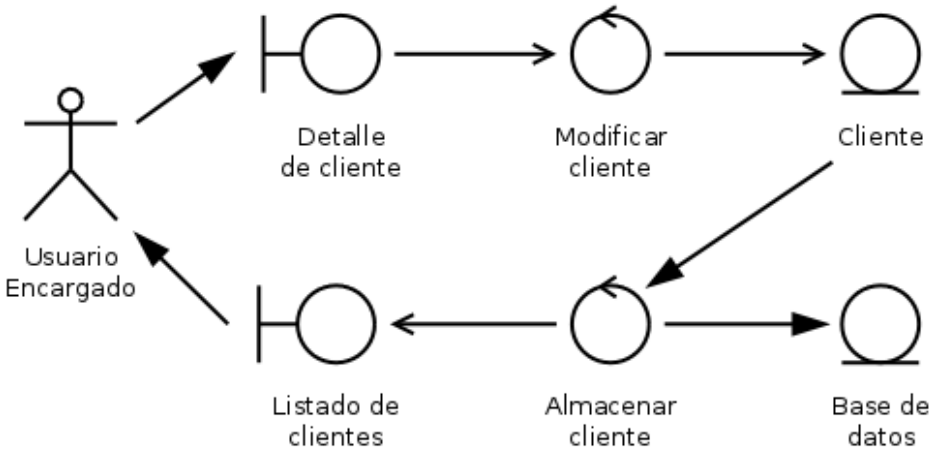


Figura 5.11. Diagrama de Robustez - Modificar cliente

Modificar cliente	
Precondiciones	El usuario en sesión tiene privilegios de encargado El cliente a modificar existe en el sistema
Poscondiciones	Los datos del cliente son actualizados
Actores	Usuario encargado
Descripción	Se trata de modificar los datos de la ficha de un cliente existente en el sistema

Excepciones	• **Los datos introducidos en el formulario no cumplen la validación**
	  ○ Se informa al usuario de qué campos debe corregir
	• **La base de datos no se encuentra disponible**
	  ○ Se informa al usuario con un error descriptivo

5.5.2.3 Eliminar cliente

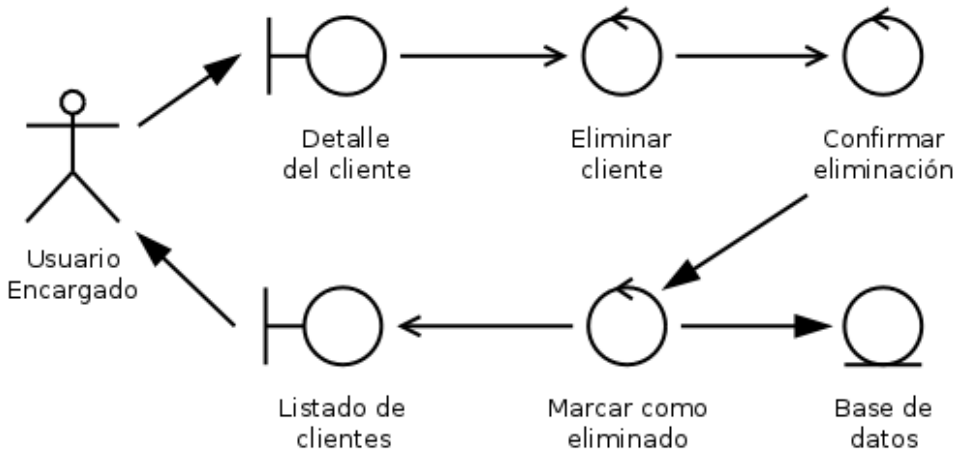


Figura 5.12. Diagrama de Robustez - Eliminar cliente

Eliminar cliente	
Precondiciones	El usuario en sesión tiene privilegios de encargado. El cliente a modificar existe en el sistema.
Poscondiciones	El cliente eliminado no aparece en listado de clientes
Actores	Usuario encargado
Descripción	Se trata de eliminar un cliente que, por la circunstancia que sea, ya no lo es
Excepciones	• **La base de datos no se encuentra disponible** ○ Se informa al usuario con un error descriptivo
Notas	El cliente no se elimina completamente de la base de datos. Únicamente se marca como eliminado y ya no aparecerá en la vista de la aplicación

5.5.2.4 Consultar listado de clientes

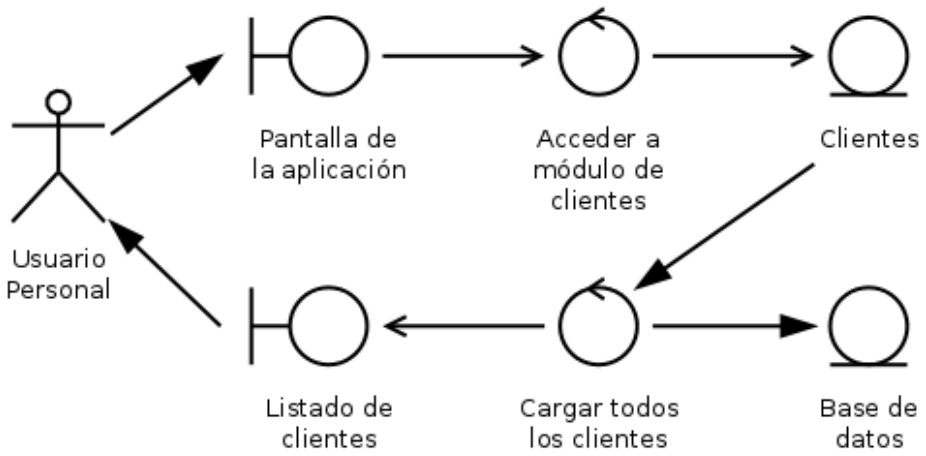


Figura 5.13. Diagrama de Robustez - Consultar listado de clientes

Consultar listado de clientes	
Precondiciones	El usuario se encuentra validado en el sistema y tiene permisos de acceso al módulo de clientes
Poscondiciones	Se muestra un listado con los clientes existentes
Actores	Usuario personal
Descripción	Se trata de consultar y mostrar un listado completo de los clientes dados de alta en el sistema
Excepciones	• **La base de datos no se encuentra disponible** ○ Se informa al usuario con un error descriptivo
Notas	El listado mostrará por defecto 10 clientes, aunque esta cantidad será configurable. Se mostrarán controles de paginación para el listado.

5.5.2.4.1 Filtrar registros de clientes

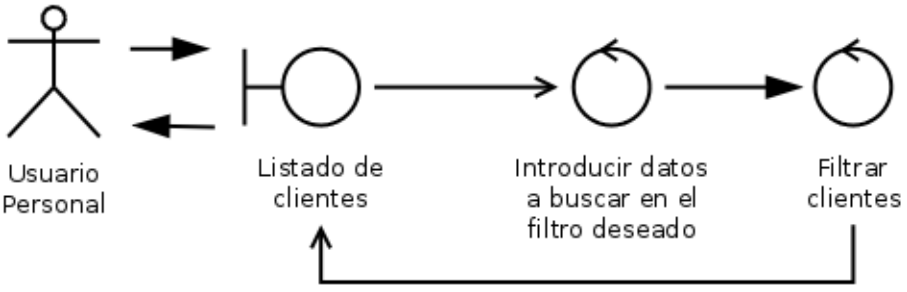


Figura 5.14. Diagrama de Robustez - Filtrar registros de clientes

Filtrar registros de clientes	
Precondiciones	El usuario se encuentra visualizando el listado de clientes

Poscondiciones	Se muestran únicamente los clientes que cumplen las condiciones establecidas mediante los filtros
Actores	Usuario personal
Descripción	Se trata de filtrar el listado de clientes, mostrando solamente aquellos que cumplan con los requisitos marcados en los filtros disponibles a tal efecto
Variaciones	• **Escenario alternativo 1** ○ No existe ningún cliente que mostrar

5.5.2.4.2 Ordenar registros de clientes

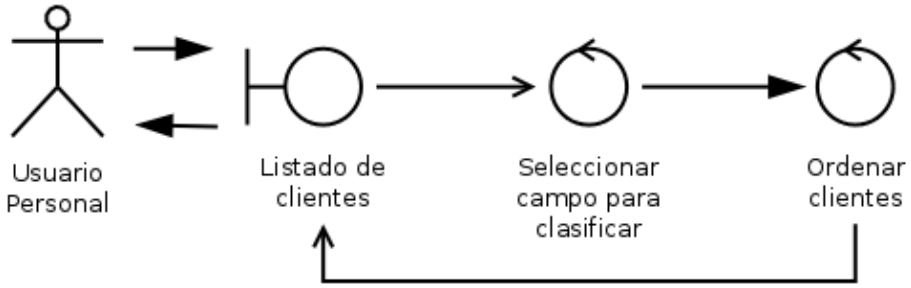


Figura 5.15. Diagrama de Robustez - Ordenar registros de clientes

Ordenar registros de clientes	
Precondiciones	El usuario se encuentra visualizando el listado de clientes
Poscondiciones	Se muestran los clientes ordenados en base al campo seleccionado
Actores	Usuario personal
Descripción	Se trata de ordenar los clientes del listado en base a los campos de la cabecera de cada columna
Variaciones	• **Escenario alternativo 1** ○ No existe ningún cliente que ordenar
Notas	La ordenación será particular para cada tipo de dato de la columna

5.5.2.5 Consultar detalle del cliente

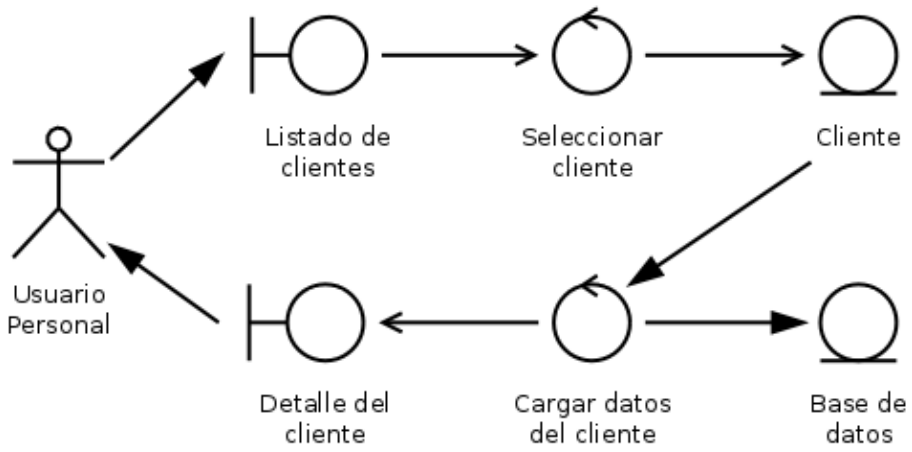


Figura 5.16. Diagrama de Robustez - Consultar detalle del cliente

Consultar detalle del cliente	
Precondiciones	El usuario se encuentra visualizando el listado de clientes
Poscondiciones	Se muestra la ficha completa del cliente seleccionado
Actores	Usuario personal
Descripción	Se trata de seleccionar un cliente de los visualizados en el listado para obtener sus detalles
Excepciones	• **La base de datos no se encuentra disponible** ○ Se informa al usuario con un error descriptivo

5.5.2.6 Ver incidencias del cliente

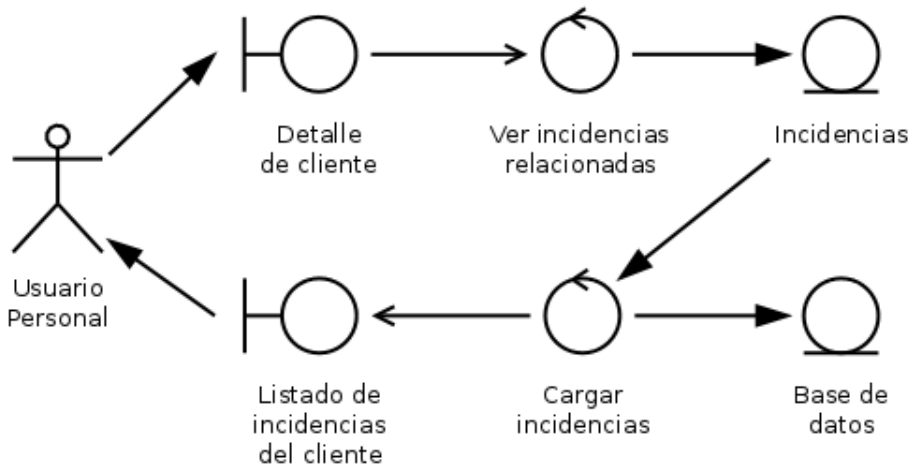


Figura 5.17. Diagrama de Robustez - Ver incidencias del cliente

Ver incidencias del cliente	
Precondiciones	El usuario se encuentra consultando la ficha del cliente
Poscondiciones	Se muestra un listado con las incidencias relacionadas con el cliente en cuestión
Actores	Usuario personal
Descripción	Se trata de, mediante una sola acción, obtener el listado de las incidencias comunicadas por el cliente en cuestión
Variaciones	• **El cliente no ha comunicado ninguna incidencia** ○ Se informa al usuario con una advertencia
Excepciones	• **La base de datos no se encuentra disponible** ○ Se informa al usuario con un error descriptivo

5.5.3 Casos de Uso de la Gestión de Contratos

5.5.3.1 Crear contrato

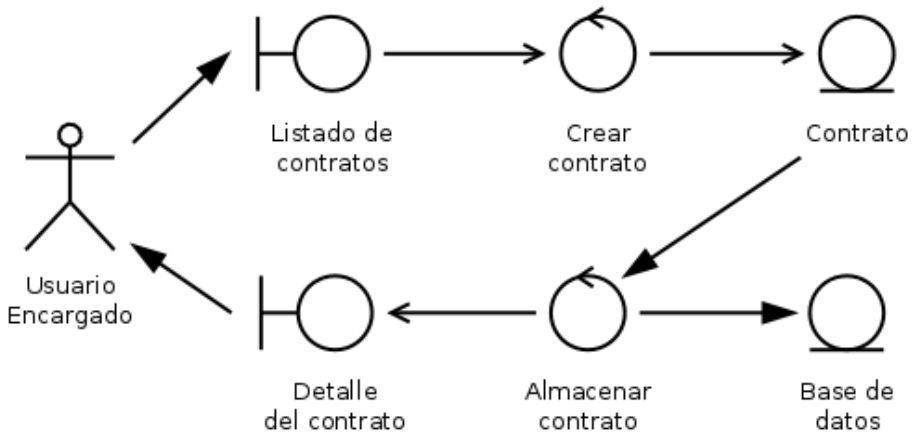


Figura 5.18. Diagrama de Robustez - Crear contrato

Crear contrato	
Precondiciones	El usuario se encuentra visualizando el listado de contratos Existe al menos un cliente dado de alta en el sistema
Poscondiciones	Se almacena un nuevo contrato vinculado a un cliente
Actores	Usuario encargado
Descripción	Se trata de, para un cliente previamente dado de alta en el sistema, vincular un nuevo contrato suscrito por el mismo con la empresa

Excepciones	• **No existe ningún cliente en el sistema**
	  ○ No es posible crear ningún contrato
	• **El formulario de entrada de datos contiene errores de validación**
	  ○ Se informa al usuario con un error localizado
	• **La base de datos no se encuentra disponible**
	  ○ Se informa al usuario con un error descriptivo

5.5.3.2 Modificar contrato

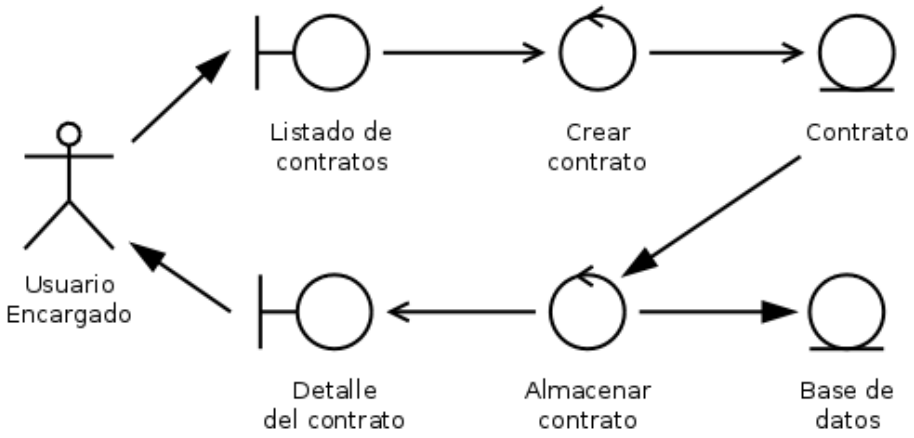


Figura 5.19. Diagrama de Robustez - Modificar contrato

Modificar contrato	
Precondiciones	El usuario se encuentra visualizando el listado de contratos Existe al menos un contrato en el sistema
Poscondiciones	Se muestra el detalle del contrato con los datos actualizados
Actores	Usuario encargado
Descripción	Se trata de modificar las propiedades de un contrato existente en el sistema

Excepciones	• **No existe ningún contrato en el sistema**
	  ○ No es posible modificar ningún contrato
	• **Los nuevos datos no cumplen la validación**
	  ○ Se informa al usuario con un error localizado
	• **La base de datos no se encuentra disponible**
	  ○ Se informa al usuario con un error descriptivo

5.5.3.3 Eliminar contrato

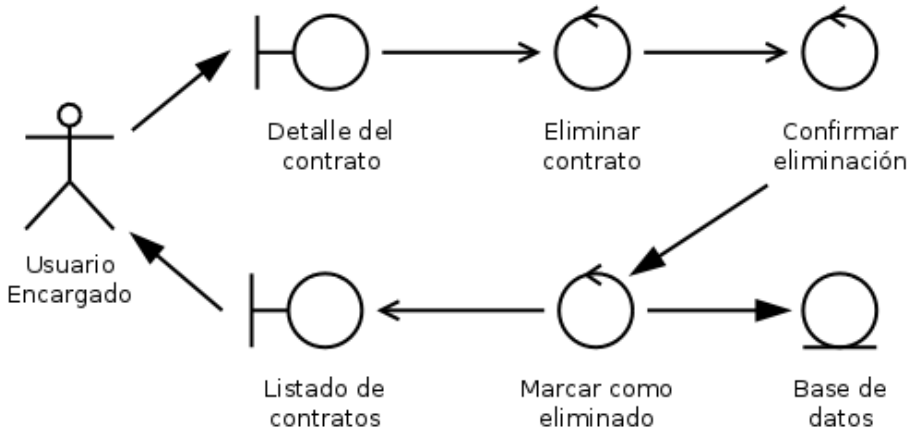


Figura 5.20. Diagrama de Robustez - Eliminar contrato

Eliminar contrato	
Precondiciones	El usuario se encuentra visualizando el listado de contratos Existe al menos un contrato en el sistema
Poscondiciones	El contrato eliminado desaparece del listado de contratos
Actores	Usuario encargado
Descripción	Se trata de eliminar un contrato existente en el sistema

Excepciones	• **No existe ningún contrato en el sistema**
	  ○ No es posible eliminar ningún contrato
	• **La base de datos no se encuentra disponible**
	  ○ Se informa al usuario con un error descriptivo

Notas	El contrato no se elimina completamente de la base de datos. Únicamente se marca como eliminado y ya no aparecerá en la vista de la aplicación

5.5.3.4 Gestionar servicios del contrato

5.5.3.4.1 Incluir servicio

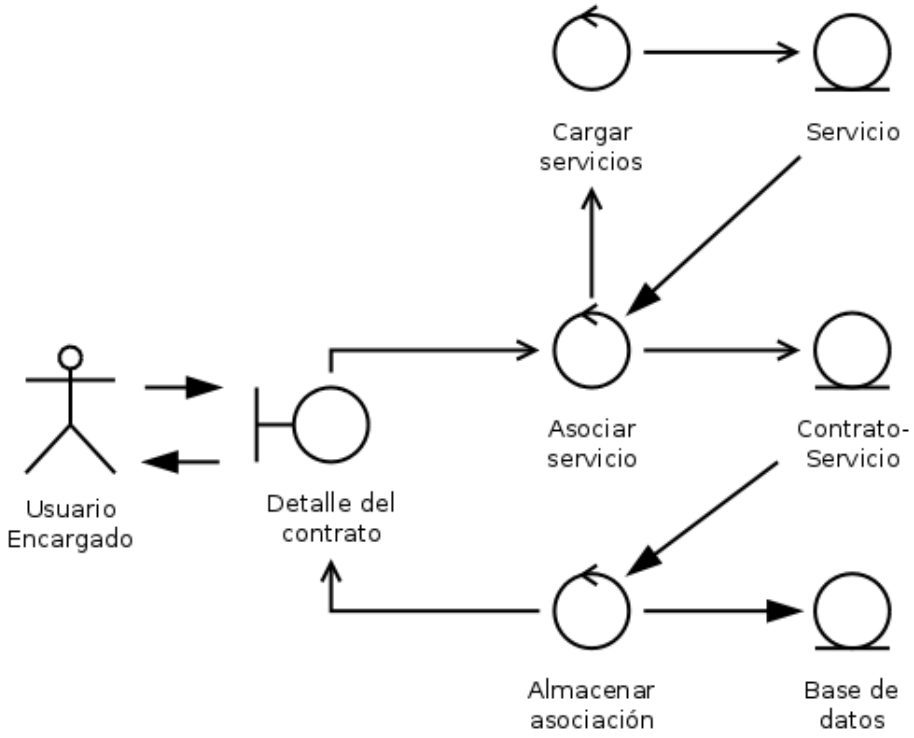


Figura 5.21. Diagrama de Robustez - Incluir servicio

Incluir servicio	
Precondiciones	Existe al menos un contrato en el sistema El usuario se encuentra visualizando el detalle de un contrato El servicio a asociar no se encuentra ya incluido en el contrato
Poscondiciones	El servicio se asocia correctamente al contrato
Actores	Usuario encargado
Descripción	Se trata de asociar un servicio de los ofrecidos por la empresa al contrato en cuestión

Excepciones	• **No existe ningún contrato en el sistema**
	  ○ No es posible incluir ningún servicio
	• **El formulario de entrada de datos contiene errores de validación**
	  ○ Se informa al usuario con un error localizado
	• **El contrato al que se asociará el servicio ya cuenta con el mismo**
	  ○ No es posible incluir dicho servicio en el contrato
	• **La base de datos no se encuentra disponible**
	  ○ Se informa al usuario con un error descriptivo

5.5.3.4.2 Modificar servicio asociado

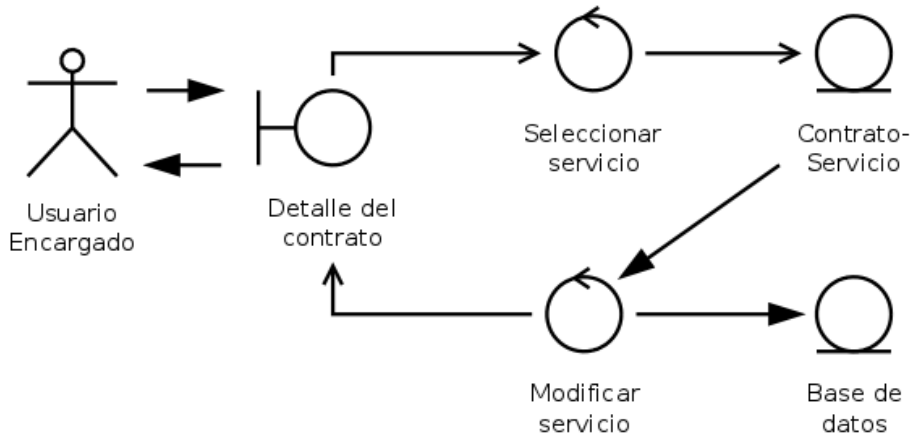


Figura 5.22. Diagrama de Robustez - Modificar servicio asociado

Modificar servicio asociado	
Precondiciones	Existe al menos un contrato en el sistema El usuario se encuentra visualizando el detalle de un contrato El contrato cuenta con al menos un servicio asociado
Poscondiciones	Los datos del servicio del contrato modificados se reflejan correctamente en el detalle del mismo
Actores	Usuario encargado
Descripción	Se trata de modificar las propiedades de un servicio de los asociados a un contrato

Excepciones	• **No existe ningún servicio asociado al contrato**
	  ○ No es posible modificar ningún servicio del contrato
	• **Los nuevos datos no cumplen la validación**
	  ○ Se informa al usuario con un error localizado
	• **La base de datos no se encuentra disponible**
	  ○ Se informa al usuario con un error descriptivo

5.5.3.4.3 Desvincular servicio

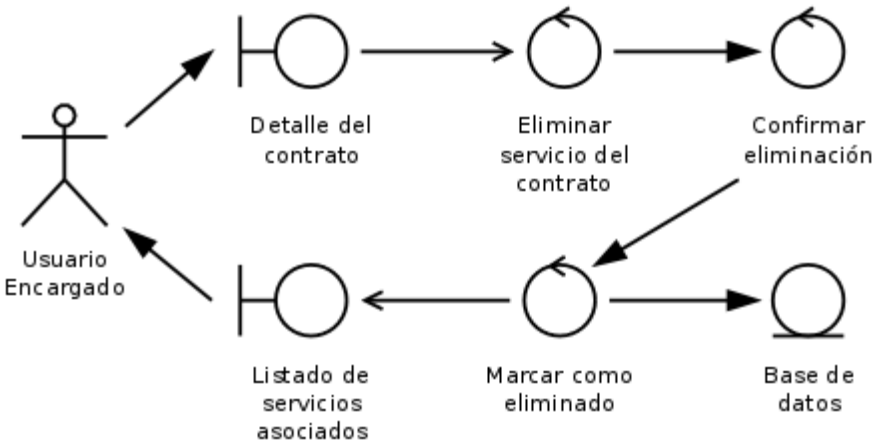


Figura 5.23. Diagrama de Robustez - Desvincular servicio

Desvincular servicio	
Precondiciones	Existe al menos un contrato en el sistema El usuario se encuentra visualizando el detalle de un contrato El contrato cuenta con al menos un servicio asociado
Poscondiciones	El servicio seleccionado desaparece del listado de servicios asociados al contrato
Actores	Usuario encargado
Descripción	Se trata de eliminar el vínculo entre un servicio y un contrato

Excepciones	• **No existe ningún servicio asociado al contrato**
	  ○ No es posible eliminar ningún servicio del contrato
	• **La base de datos no se encuentra disponible**
	  ○ Se informa al usuario con un error descriptivo

Notas	El servicio contratado no se elimina completamente de la base de datos. Únicamente se marca como eliminado y ya no aparecerá en la vista de la aplicación

5.5.3.4.4 Listar servicios asociados

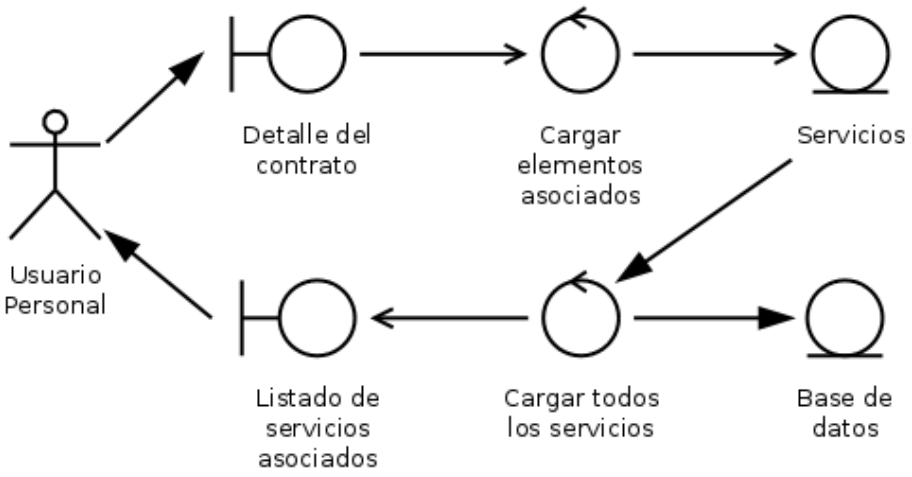


Figura 5.24. Diagrama de Robustez - Listar servicios asociados

Listar servicios asociados	
Precondiciones	Existe al menos un contrato en el sistema El usuario se encuentra visualizando el detalle de un contrato
Poscondiciones	Se muestra un listado con los servicios asociados al contrato
Actores	Usuario personal
Descripción	Se trata de consultar un listado de los servicios asociados al contrato en cuestión
Variaciones	• **Escenario alternativo 1** ○ El contrato no cuenta con ningún servicio asociado
Excepciones	• **La base de datos no se encuentra disponible** ○ Se informa al usuario con un error descriptivo

5.5.3.5 Consultar listado de contratos

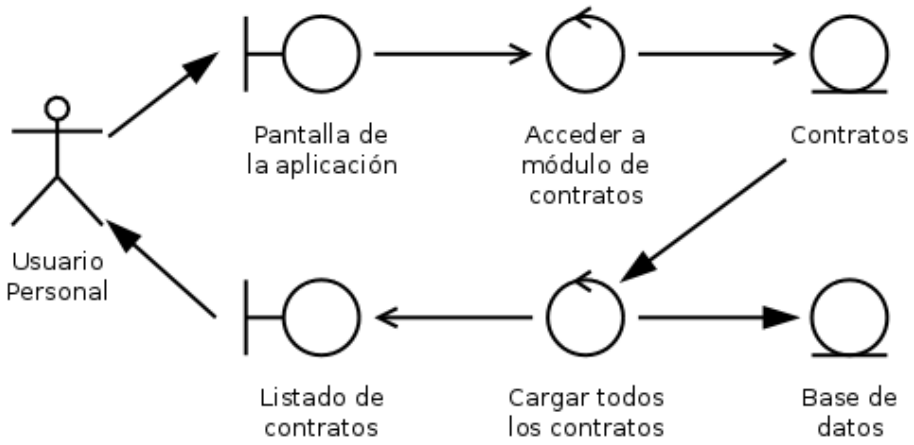


Figura 5.25. Diagrama de Robustez - Consultar listado de contratos

Consultar listado de contratos	
Precondiciones	El usuario se encuentra validado en el sistema y tiene permiso de acceso al módulo de contratos
Poscondiciones	Se muestra un listado con todos los contratos dados de alta en el sistema
Actores	Usuario personal
Descripción	Se trata de consultar un listado con todos los contratos suscritos por clientes almacenados en el sistema
Variaciones	• **Escenario alternativo 1** ○ No existe ningún contrato dado de alta en el sistema
Excepciones	• **La base de datos no se encuentra disponible** ○ Se informa al usuario con un error descriptivo

5.5.3.5.1 Filtrar registros de contratos

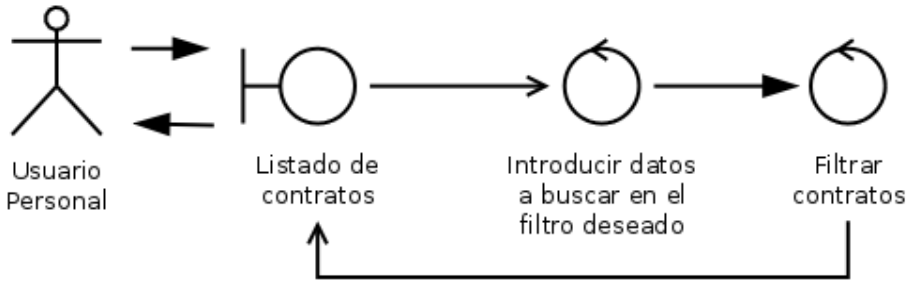


Figura 5.26. Diagrama de Robustez - Filtrar registros de contratos

Filtrar registros de contratos	
Precondiciones	El usuario se encuentra visualizando el listado de contratos
Poscondiciones	Se muestran únicamente los contratos que cumplen las condiciones establecidas mediante los filtros
Actores	Usuario personal
Descripción	Se trata de filtrar el listado de contratos, mostrando solamente aquellos que cumplan con los requisitos marcados en los filtros disponibles a tal efecto
Variaciones	• **Escenario alternativo 1** ○ No existe ningún contrato que mostrar

5.5.3.5.2 Ordenar registros de contratos

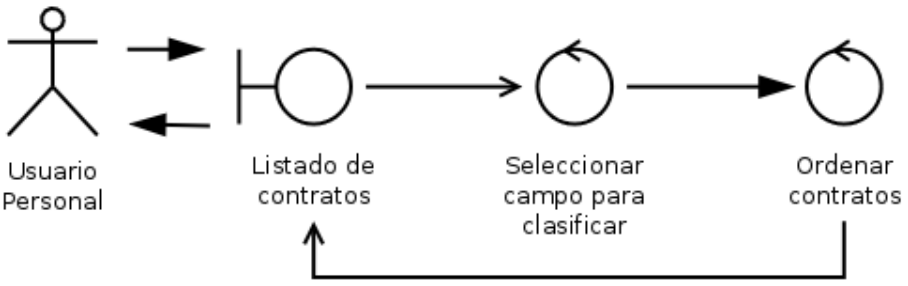


Figura 5.27. Diagrama de Robustez - Ordenar registros de contratos

Ordenar registros de contratos	
Precondiciones	El usuario se encuentra visualizando el listado de contratos
Poscondiciones	Se muestran los contratos ordenados en base al campo seleccionado
Actores	Usuario personal
Descripción	Se trata de ordenar los contratos del listado en base a los campos de la cabecera de cada columna
Variaciones	• **Escenario alternativo 1** ○ No existe ningún contrato que mostrar
Notas	La ordenación será particular para cada tipo de dato de la columna

5.5.3.6 Consultar detalle del contrato

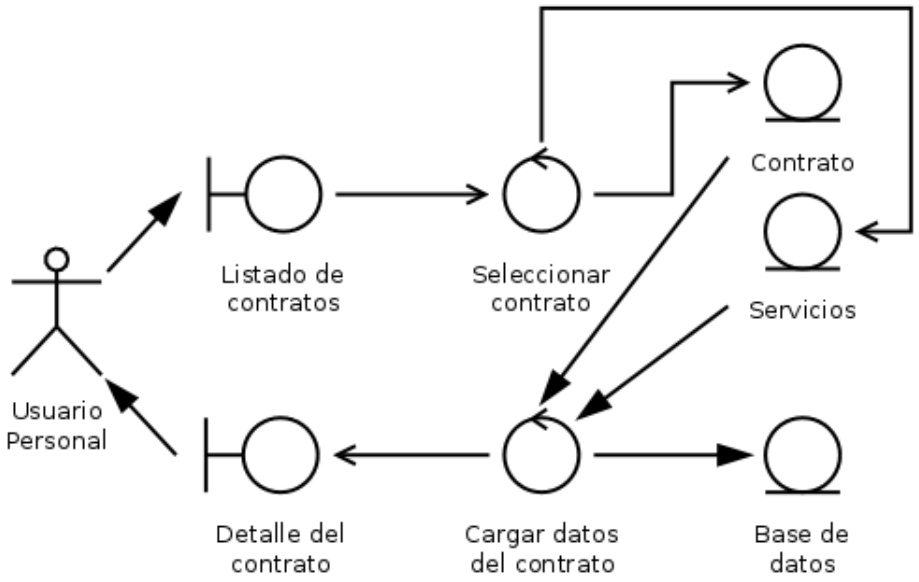


Figura 5.28. Diagrama de Robustez - Consultar detalle del contrato

Consultar detalle del contrato	
Precondiciones	El usuario se encuentra visualizando el listado de contratos
Poscondiciones	Se muestra la ficha completa del contrato seleccionado, incluidos los servicios asociados al mismo
Actores	Usuario personal
Descripción	Se trata de seleccionar un contrato de los visualizados en el listado para obtener sus detalles
Excepciones	• **La base de datos no se encuentra disponible** ○ Se informa al usuario con un error descriptivo

5.5.4 Casos de Uso de la Gestión de Servicios

5.5.4.1 Crear servicio

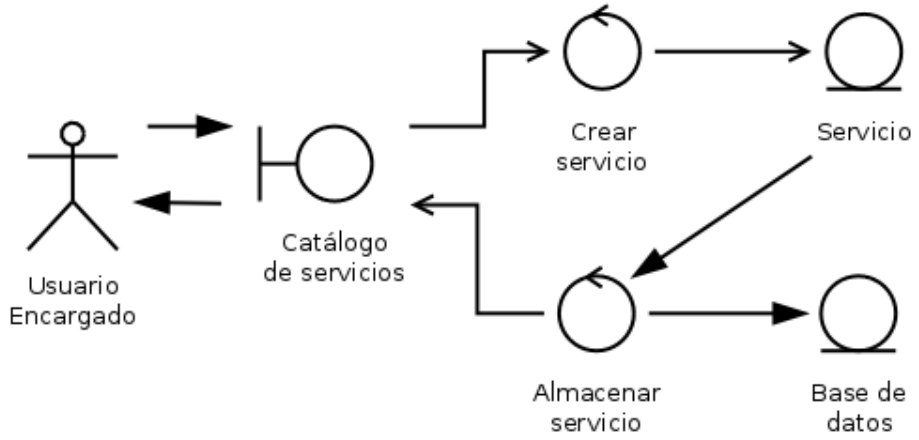


Figura 5.29. Diagrama de Robustez - Crear servicio

Crear servicio	
Precondiciones	El usuario se encuentra visualizando el catálogo de servicios ofrecidos por la empresa
Poscondiciones	Se almacena un nuevo servicio y se muestra en el catálogo
Actores	Usuario encargado
Descripción	Se trata de añadir un nuevo servicio para ofrecer a los clientes

Excepciones	• **El formulario de entrada de datos contiene errores de validación**
	  ○ Se informa al usuario con un error localizado
	• **La base de datos no se encuentra disponible**
	  ○ Se informa al usuario con un error descriptivo

5.5.4.2 Modificar servicio

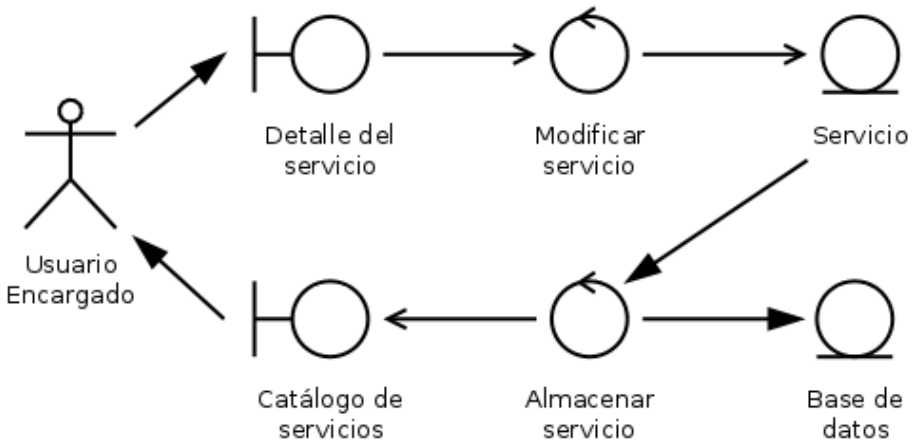


Figura 5.30. Diagrama de Robustez - Modificar servicio

Modificar servicio	
Precondiciones	Existe al menos un servicio en el catálogo El usuario se encuentra visualizando el detalle de un servicio
Poscondiciones	Los datos del servicio modificados se reflejan correctamente en el catálogo de servicios
Actores	Usuario encargado
Descripción	Se trata de modificar las propiedades de un servicio de los ofertados en el catálogo

Excepciones	• **No existe ningún servicio en el catálogo**
	  ○ No es posible modificar ningún servicio del contrato
	• **Los nuevos datos no cumplen la validación**
	  ○ Se informa al usuario con un error localizado
	• **La base de datos no se encuentra disponible**
	  ○ Se informa al usuario con un error descriptivo

5.5.4.3 Eliminar servicio

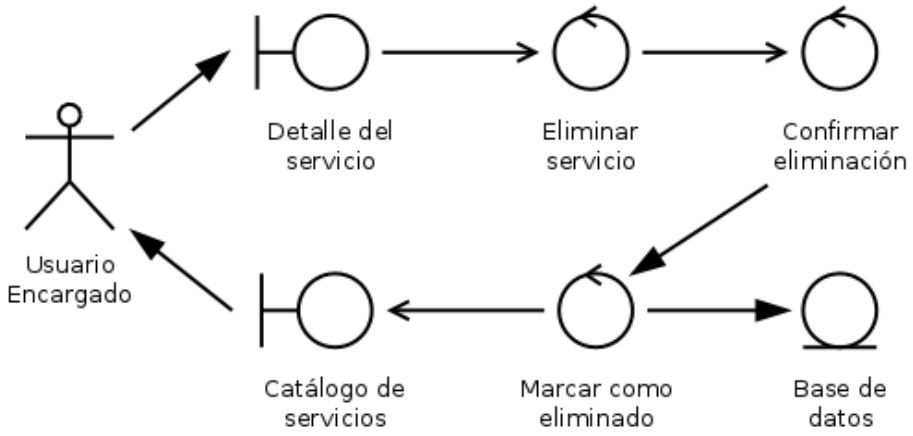


Figura 5.31. Diagrama de Robustez - Eliminar servicio

Eliminar servicio	
Precondiciones	El usuario se encuentra visualizando el catálogo de servicios Existe al menos un servicio en el catálogo
Poscondiciones	El servicio eliminado desaparece del catálogo
Actores	Usuario encargado
Descripción	Se trata de eliminar un servicio de los existentes en el catálogo

Excepciones	• **No existe ningún servicio en el catálogo**
	  ○ No es posible eliminar ningún servicio
	• **La base de datos no se encuentra disponible**
	  ○ Se informa al usuario con un error descriptivo

Notas	El servicio no se elimina completamente de la base de datos. Únicamente se marca como eliminado y ya no aparecerá en la vista de la aplicación

5.5.4.4 Gestionar tarifas del servicio

5.5.4.4.1 Incluir tarifa

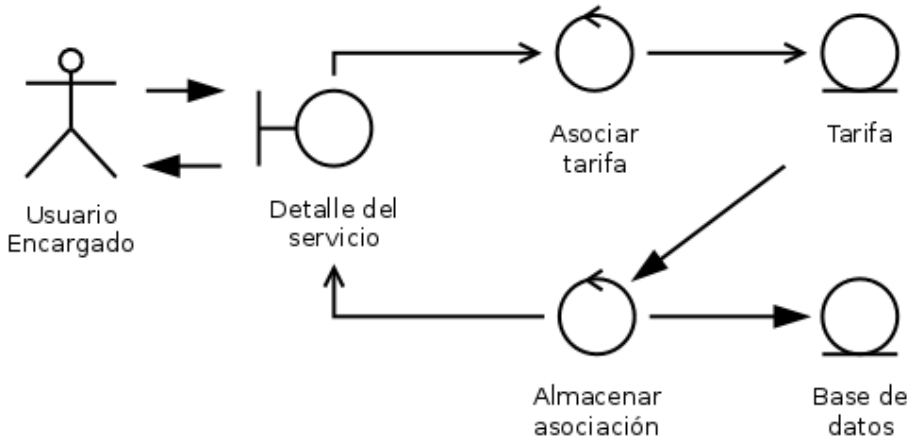


Figura 5.32. Diagrama de Robustez - Incluir tarifa

Incluir tarifa	
Precondiciones	Existe al menos un servicio en el catálogo El usuario se encuentra visualizando el catálogo de servicios
Poscondiciones	La tarifa se asocia correctamente al servicio
Actores	Usuario encargado
Descripción	Se trata de incluir una nueva tarifa para un servicio en particular

Excepciones	• **No existe ningún servicio en el sistema**
	  ○ No es posible incluir ninguna tarifa
	• **Los datos requeridos para la tarifa no cumplen la validación**
	  ○ Se informa al usuario con un error localizado
	• **La base de datos no se encuentra disponible**
	  ○ Se informa al usuario con un error descriptivo

5.5.4.4.2 Modificar tarifa

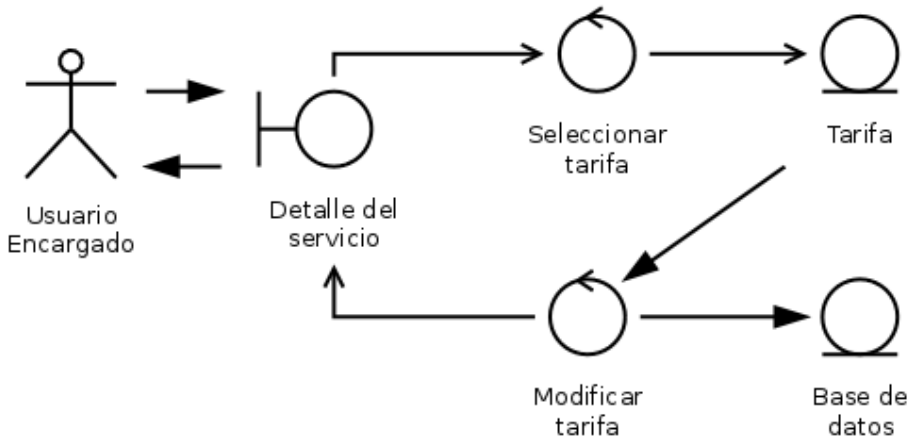


Figura 5.33. Diagrama de Robustez - Modificar tarifa

Modificar tarifa	
Precondiciones	Existe al menos un servicio en el catálogo El usuario se encuentra visualizando el detalle de un servicio El servicio cuenta con al menos una tarifa asociada
Poscondiciones	Los datos de la tarifa modificada se reflejan correctamente en el detalle de la misma
Actores	Usuario encargado
Descripción	Se trata de modificar las propiedades de una tarifa de las aplicables a un servicio en particular

Excepciones	• **No existe ninguna tarifa asociada al servicio**
	  ○ No es posible modificar ninguna tarifa del servicio
	• **Los nuevos datos no cumplen la validación**
	  ○ Se informa al usuario con un error localizado
	• **La base de datos no se encuentra disponible**
	  ○ Se informa al usuario con un error descriptivo

5.5.4.4.3 Eliminar tarifa

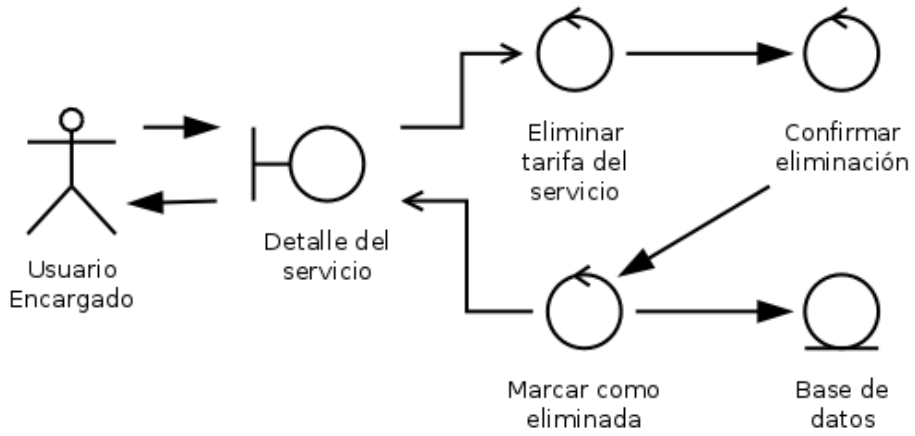


Figura 5.34. Diagrama de Robustez - Eliminar tarifa

Eliminar tarifa	
Precondiciones	Existe al menos un servicio en el catálogo El usuario se encuentra visualizando el detalle del servicio El servicio tiene al menos una tarifa asociada
Poscondiciones	La tarifa seleccionada desaparece del listado de tarifas asociadas al servicio
Actores	Usuario encargado
Descripción	Se trata de eliminar una tarifa de las aplicables a un servicio

Excepciones	• **No existe ninguna tarifa asociada al servicio**
	  ○ No es posible eliminar ninguna tarifa del servicio
	• **La base de datos no se encuentra disponible**
	  ○ Se informa al usuario con un error descriptivo

Notas	La tarifa no se elimina completamente de la base de datos. Únicamente

	se marca como eliminada y ya no aparecerá en la vista de la aplicación

5.5.4.4.4 Listar tarifas

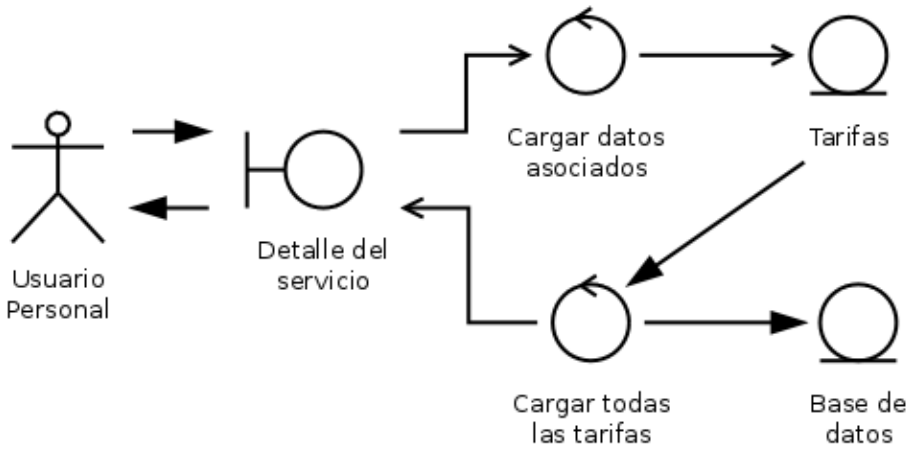


Figura 5.35. Diagrama de Robustez - Listar tarifas

Listar tarifas	
Precondiciones	El usuario tiene permiso de acceso al módulo de servicios El usuario se encuentra visualizando el catálogo de servicios Existe al menos un servicio en el catálogo
Poscondiciones	Se muestra un listado con las tarifas asociadas a cada servicio
Actores	Usuario personal
Descripción	Se trata de consultar un listado con las tarifas asociadas a cada uno de los servicios del catálogo
Variaciones	• **Escenario alternativo 1** ○ El servicio no cuenta con ninguna tarifa asociada
Excepciones	• **La base de datos no se encuentra disponible** ○ Se informa al usuario con un error descriptivo

5.5.4.4.4.1 Filtrar registros

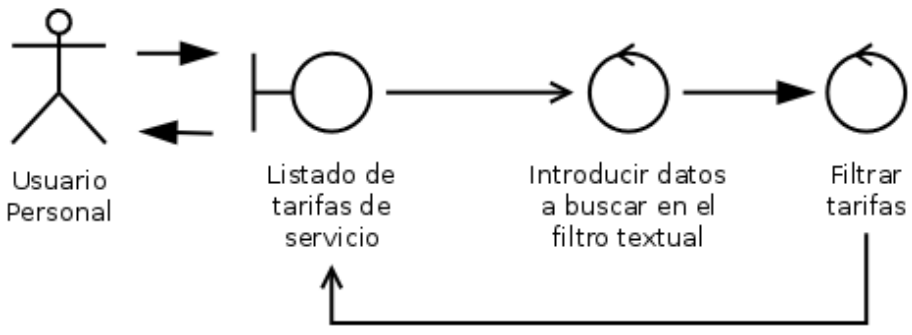


Figura 5.36. Diagrama de Robustez - Filtrar registros de tarifas

Filtrar registros de tarifas	
Precondiciones	El usuario se encuentra visualizando el catálogo de servicios
Poscondiciones	Se muestran, para cada servicio, aquellas tarifas que cumplen las condiciones establecidas en el filtro
Actores	Usuario personal
Descripción	Se trata de filtrar las tarifas aplicables para cada servicio, mostrando solamente aquellas que cumplan con los requisitos marcados en el filtro textual disponible a tal efecto
Variaciones	• **Escenario alternativo 1** ○ No existe ninguna tarifa que mostrar para el servicio

5.5.4.4.4.2 Ordenar registros

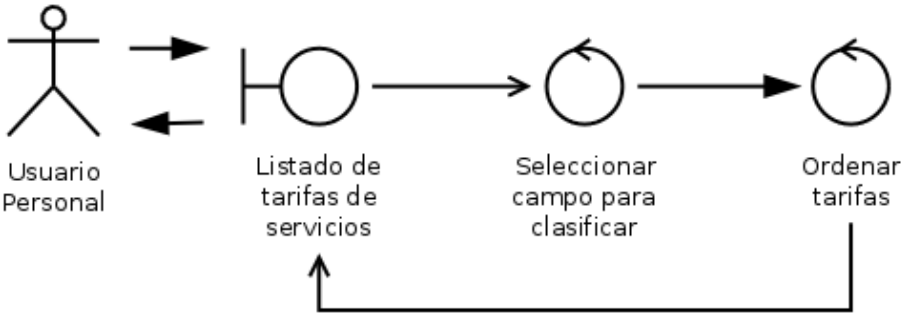


Figura 5.37. Diagrama de Robustez - Ordenar registros de tarifas

Ordenar registros de tarifas	
Precondiciones	El usuario se encuentra visualizando el catálogo de servicios y sus tarifas
Poscondiciones	Se muestran las tarifas del servicio ordenadas en base al campo seleccionado
Actores	Usuario personal
Descripción	Se trata de ordenar las tarifas del catálogo, de forma independiente para cada servicio, en base a los campos de la cabecera de cada columna
Variaciones	• **Escenario alternativo 1** ○ No existe ninguna tarifa que ordenar para el servicio
Notas	La ordenación será específica de acuerdo al tipo de dato de la columna

5.5.4.5 Consultar listado de servicios

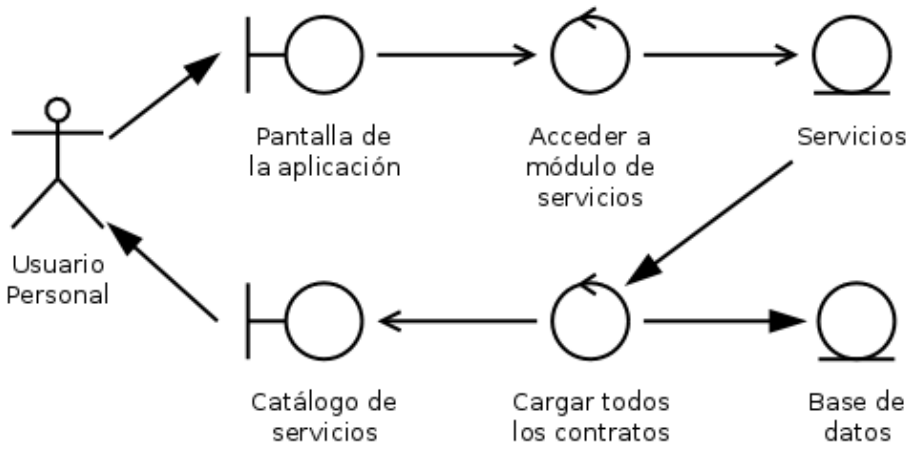


Figura 5.38. Diagrama de Robustez - Consultar listado de servicios

Consultar listado de servicios	
Precondiciones	El usuario se encuentra validado en el sistema y tiene permiso de acceso al módulo de servicios
Poscondiciones	Se muestra un listado con todos los servicios dados de alta en el catálogo del sistema
Actores	Usuario personal
Descripción	Se trata de consultar un listado con todos los servicios ofrecidos por la empresa a sus clientes
Variaciones	• **Escenario alternativo 1** ○ No existe ningún servicio dado de alta en el sistema
Excepciones	• **La base de datos no se encuentra disponible** ○ Se informa al usuario con un error descriptivo

5.5.5 Casos de Uso de la Gestión de Incidencias

5.5.5.1 Crear incidencia

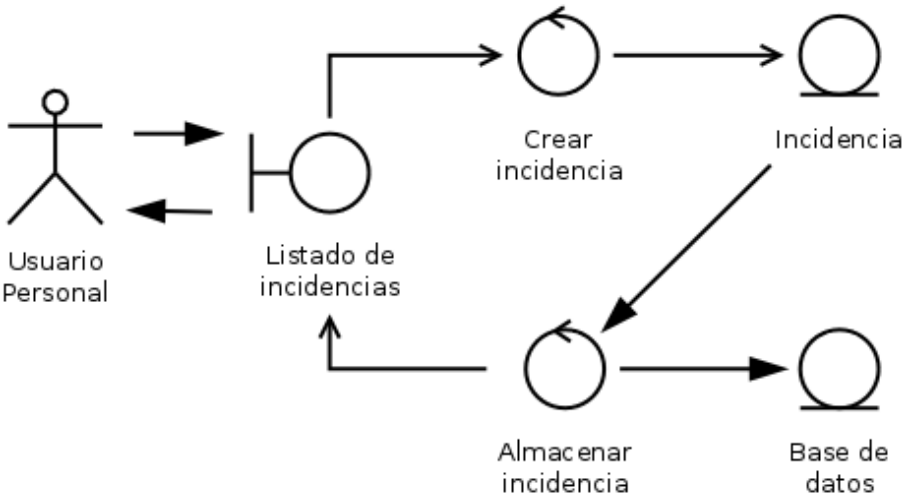


Figura 5.39. Diagrama de Robustez - Crear incidencia

Crear incidencia	
Precondiciones	El usuario se encuentra visualizando el listado de incidencias Existe al menos un cliente que ha suscrito un contrato de servicio
Poscondiciones	Se almacena una nueva incidencia vinculada al contrato de un cliente
Actores	Usuario personal
Descripción	Se trata de almacenar una nueva incidencia comunicada por un cliente que cuenta con un contrato de servicios suscrito con la empresa

Excepciones	• **No existe ningún cliente en el sistema**
	  ○ No es posible crear ninguna incidencia
	• **El cliente no cuenta con ningún contrato suscrito**
	  ○ No es posible crear ninguna incidencia para el cliente
	• **El formulario de entrada de datos contiene errores de validación**
	  ○ Se informa al usuario con un error localizado
	• **La base de datos no se encuentra disponible**
	  ○ Se informa al usuario con un error descriptivo

5.5.5.2 Modificar incidencia

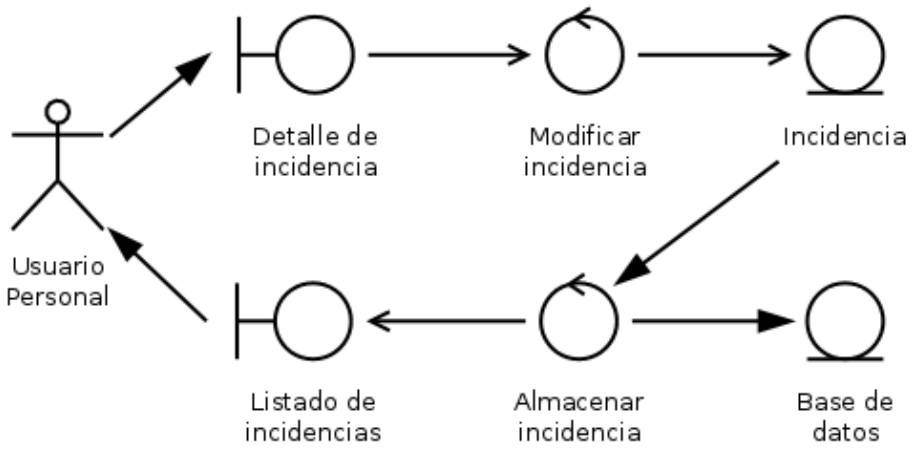


Figura 5.40. Diagrama de Robustez - Modificar incidencia

Modificar incidencia	
Precondiciones	El usuario se encuentra visualizando el detalle de una incidencia
Poscondiciones	Los datos de la incidencia modificada se reflejan correctamente en el detalle de la misma y en el listado de incidencias
Actores	Usuario personal
Descripción	Se trata de modificar las propiedades de una incidencia de las comunicadas por un cliente

Excepciones	• **No existe ninguna incidencia**
	  ○ No es posible modificar ninguna incidencia
	• **Los nuevos datos no cumplen la validación**
	  ○ Se informa al usuario con un error localizado
	• **La base de datos no se encuentra disponible**
	  ○ Se informa al usuario con un error descriptivo

5.5.5.3 Eliminar incidencia

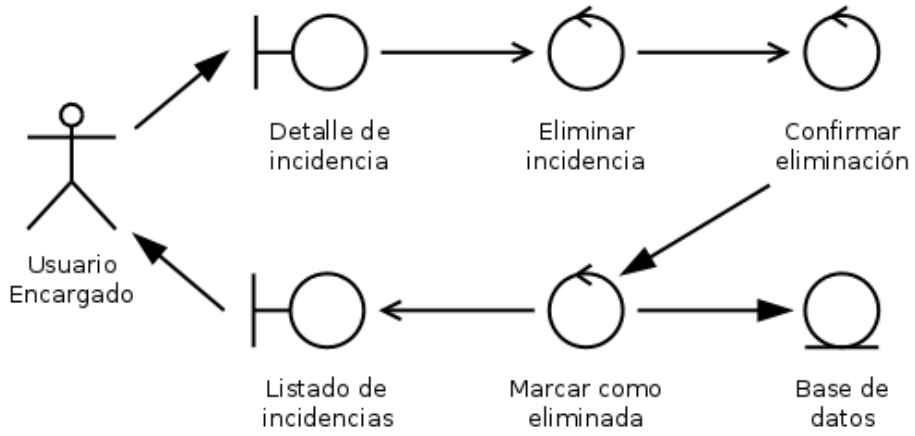


Figura 5.41. Diagrama de Robustez - Eliminar incidencia

Eliminar incidencia	
Precondiciones	El usuario se encuentra visualizando el detalle de una incidencia
Poscondiciones	La incidencia eliminada desaparece del listado de incidencias
Actores	Usuario encargado
Descripción	Se trata de eliminar una incidencia existente en el sistema

Excepciones	• **No existe ninguna incidencia en el sistema**
	  ○ No es posible eliminar ninguna incidencia
	• **La base de datos no se encuentra disponible**
	  ○ Se informa al usuario con un error descriptivo

Notas	La incidencia no se elimina completamente de la base de datos. Únicamente se marca como eliminada y ya no aparecerá en la vista de la aplicación

5.5.5.4 Consultar detalle de incidencia

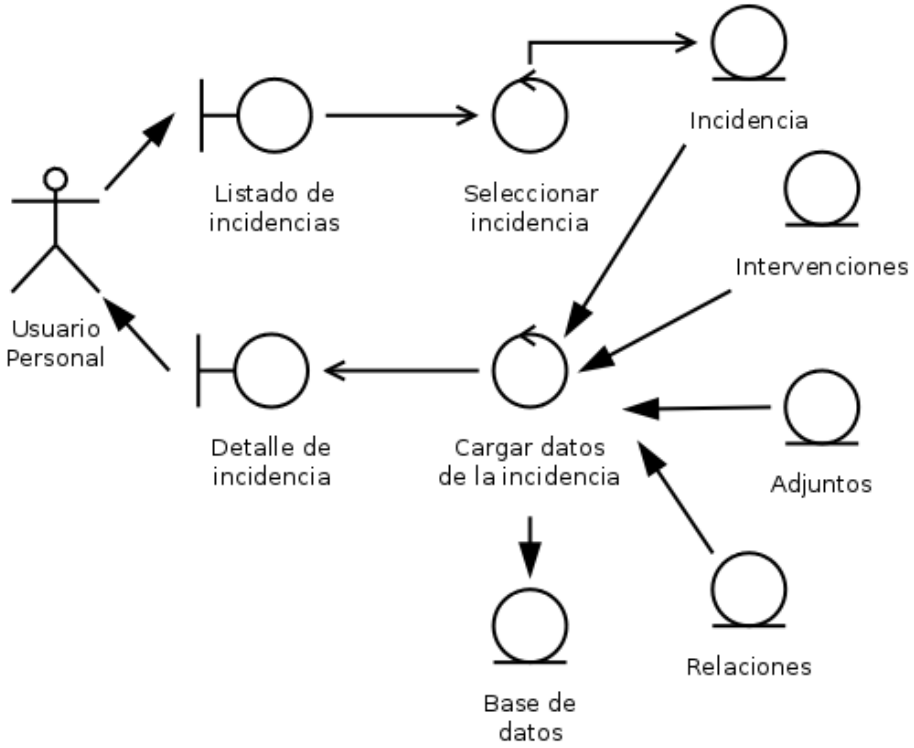


Figura 5.42. Diagrama de Robustez - Consultar detalle de incidencia

Consultar detalle de incidencia	
Precondiciones	El usuario se encuentra visualizando el listado de incidencias
Poscondiciones	Se muestra el detalle completo de la incidencia
Actores	Usuario personal
Descripción	Se trata de seleccionar una incidencia del listado de las existentes para consultar todos los datos relativos a la misma, tanto los descriptivos como asociados (intervenciones, adjuntos y relaciones)
Excepciones	• **La base de datos no se encuentra disponible** ○ Se informa al usuario con un error descriptivo

5.5.5.5 Clonar incidencia

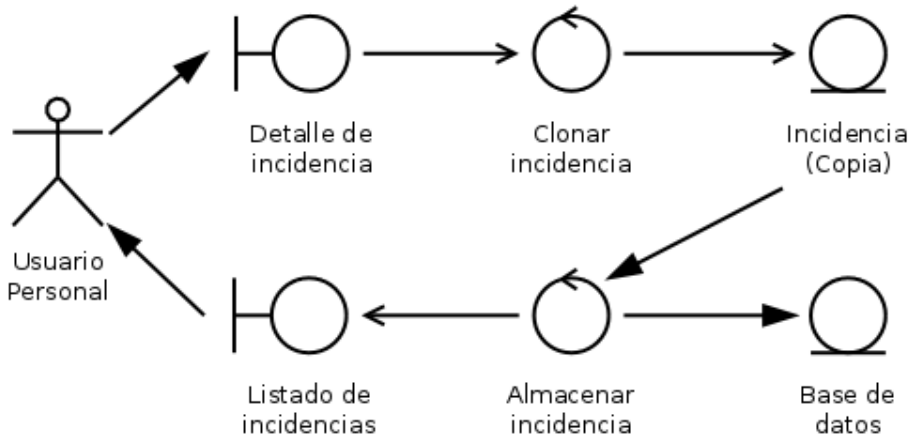


Figura 5.43. Diagrama de Robustez - Clonar incidencia

Clonar incidencia	
Precondiciones	El usuario se encuentra visualizando el detalle de una incidencia
Poscondiciones	Se crea una nueva incidencia en el sistema con los mismos datos descriptivos que la original
Actores	Usuario personal
Descripción	Se trata de crear una incidencia idéntica a otra existente, copiando únicamente sus datos descriptivos. Esto significa que los datos asociados (intervenciones, adjuntos y relaciones) no serán clonados

Excepciones	• **No existe ninguna incidencia en el sistema**
	  ○ No es posible crear una nueva a partir de otra existente
	• **La base de datos no se encuentra disponible**
	  ○ Se informa al usuario con un error descriptivo

5.5.5.6 Generar informe imprimible

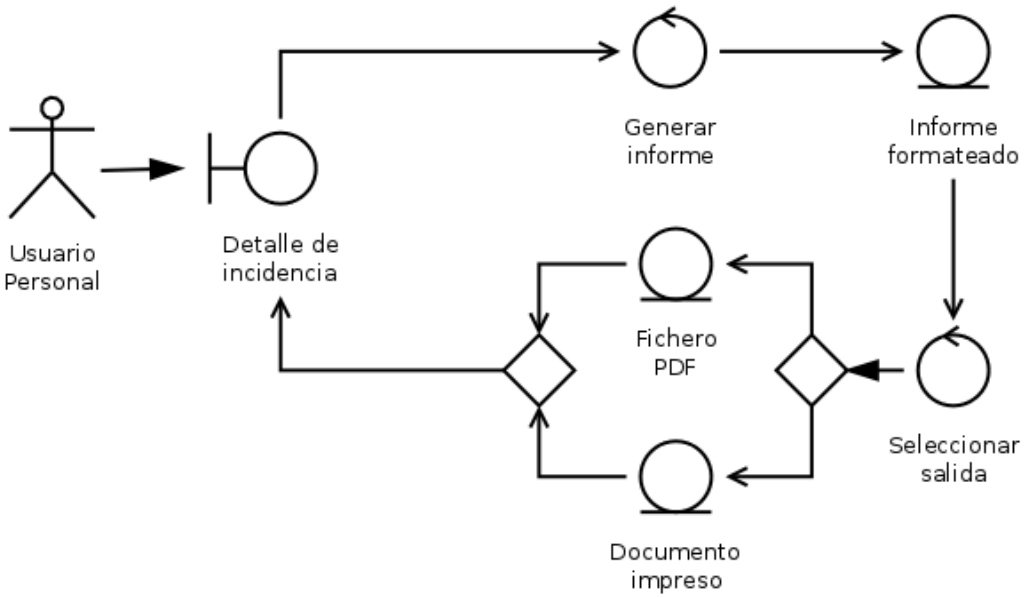


Figura 5.44. Diagrama de Robustez - Generar informe imprimible

Generar informe imprimible	
Precondiciones	El usuario se encuentra visualizando el detalle de una incidencia
Poscondiciones	Se genera un informe en formato imprimible en la salida seleccionada por el usuario
Actores	Usuario personal
Descripción	Se trata de obtener un informe detallado de la incidencia en cuestión en formato adaptado para su impresión. El usuario seleccionará el destino del informe, que podrá ser un fichero PDF o un documento impreso en papel

Variaciones (escenarios secundarios)	• **El usuario selecciona como destino del informe un fichero PDF**
	  ○ Se genera un documento PDF en la ruta especificada
	• **El usuario selecciona como destino del informe una impresora**
	  ○ Se imprime el informe en la impresora seleccionada

Excepciones	• **No existe ninguna incidencia en el sistema**
	  ○ No es posible eliminar ninguna incidencia
	• **La impresora no se encuentra disponible**
	  ○ El informe quedará en cola a la espera de una acción correctiva
	• **La base de datos no se encuentra disponible**
	  ○ Se informa al usuario con un error descriptivo

Notas	Las tareas de generación de PDF e impresión en papel del documento correrán a cargo del navegador y/o del sistema operativo del usuario

5.5.5.7 Listar incidencias

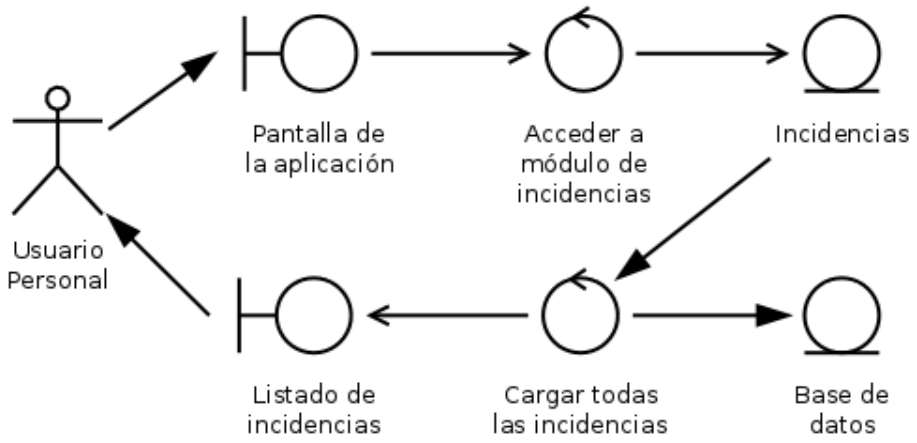


Figura 5.45. Diagrama de Robustez - Listar incidencias

Listar incidencias	
Precondiciones	El usuario se encuentra validado en el sistema y tiene permiso de acceso al módulo de incidencias Existe al menos una incidencia almacenada en el sistema
Poscondiciones	Se muestra un listado con las incidencias dadas de alta en el sistema
Actores	Usuario personal
Descripción	Se trata de consultar un listado de las incidencias comunicadas por los clientes y que están siendo o han sido procesadas
Excepciones	• **La base de datos no se encuentra disponible** ○ Se informa al usuario con un error descriptivo

5.5.5.7.1 Filtrar registros

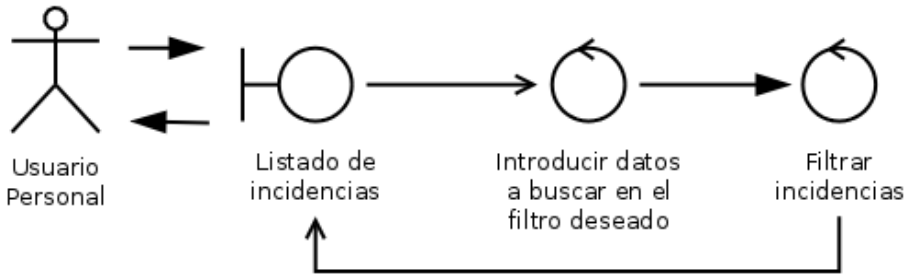


Figura 5.46. Diagrama de Robustez - Filtrar registros de incidencias

Filtrar registros de incidencias	
Precondiciones	El usuario se encuentra visualizando el listado de incidencias
Poscondiciones	Se muestran únicamente las incidencias que cumplen las condiciones establecidas mediante los filtros
Actores	Usuario personal
Descripción	Se trata de filtrar el listado de incidencias, mostrando solamente aquellas que cumplan con los requisitos marcados en los filtros y desplegables disponibles a tal efecto
Excepciones	• **Escenario alternativo 1** ○ No existe ninguna incidencia que mostrar

5.5.5.7.2 Ordenar registros

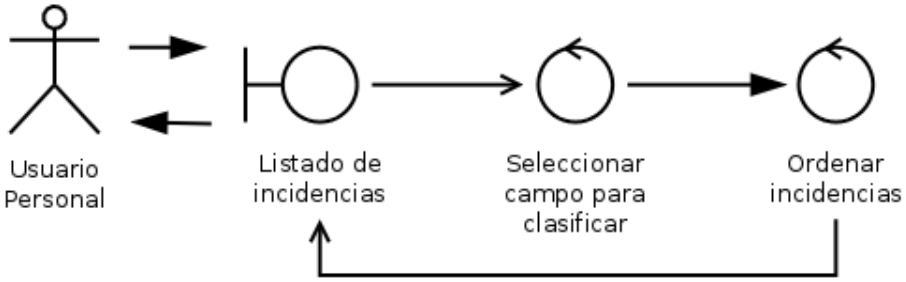


Figura 5.47. Diagrama de Robustez - Ordenar registros de incidencias

Ordenar registros de incidencias	
Precondiciones	El usuario se encuentra visualizando el listado de incidencias
Poscondiciones	Se muestran las incidencias ordenadas en base al campo seleccionado
Actores	Usuario personal
Descripción	Se trata de ordenar las incidencias del listado en base a los campos de la cabecera de cada columna
Variaciones (escenarios secundarios)	• **Escenario alternativo 1** ○ No existe ninguna incidencia que ordenar

Notas	La ordenación será específica de acuerdo al tipo de dato de la columna
	Se aplicará una priorización automática durante la carga del listado de acuerdo a los siguientes niveles
	• **Primer nivel de priorización**
	  ○ Incidencias asignadas al usuario en sesión
	• **Segundo nivel de priorización**
	  ○ Incidencias con mayor prioridad

5.5.5.8 Gestionar intervenciones de la incidencia

5.5.5.8.1 Incluir intervención

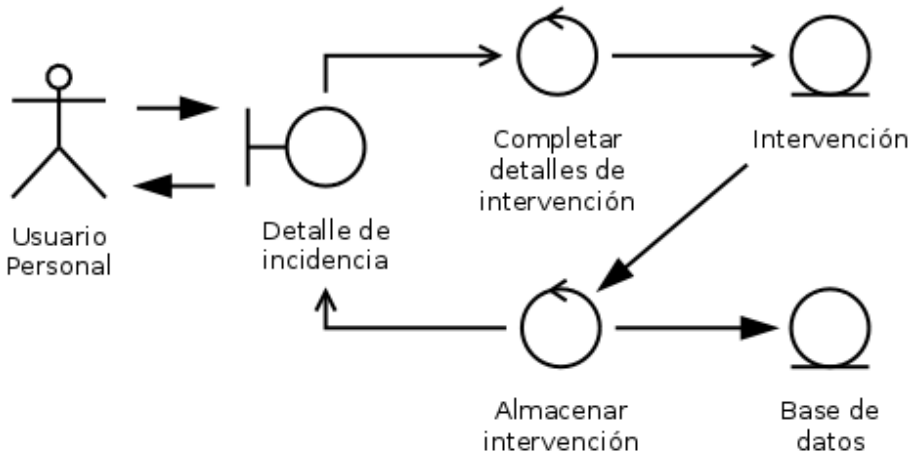


Figura 5.48. Diagrama de Robustez - Incluir intervención

Incluir intervención	
Precondiciones	Existe al menos una incidencia dada de alta en el sistema El usuario se encuentra visualizando el detalle de una incidencia
Poscondiciones	La intervención se anota correctamente en la ficha de la incidencia
Actores	Usuario personal
Descripción	Se trata de incluir una intervención realizada en el procesamiento de la incidencia en cuestión. Para ello, el usuario proporcionará los datos de la intervención y ésta quedará asociada a la incidencia

Excepciones	• **No existe ninguna incidencia en el sistema**
	  ○ No es posible anotar ninguna intervención
	• **Los datos de la intervención no cumplen la validación**
	  ○ Se informa al usuario con un error localizado
	• **La base de datos no se encuentra disponible**
	  ○ Se informa al usuario con un error descriptivo

Notas	El campo *Parte* de la intervención hace referencia al número de albarán que se generó en la intervención. No todas las intervenciones requieren de albarán, por lo que este campo será opcional

5.5.5.8.2 Modificar intervención

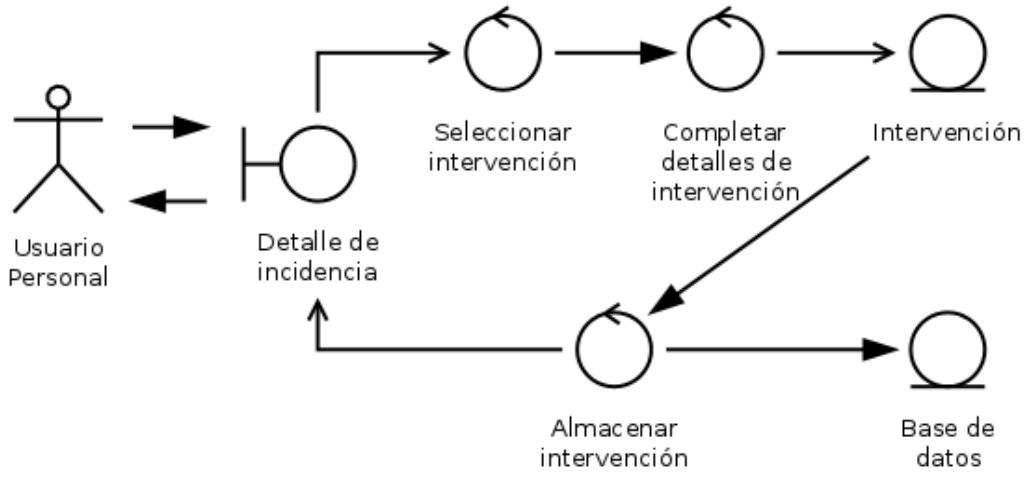


Figura 5.49. Diagrama de Robustez - Modificar intervención

Modificar intervención	
Precondiciones	Existe al menos una intervención en la incidencia a tratar El usuario se encuentra visualizando el detalle de la incidencia
Poscondiciones	Los nuevos datos de la incidencia se presentan correctamente
Actores	Usuario personal
Descripción	Se trata de editar las propiedades de una intervención realizada en una incidencia para actualizarlos o corregir valores incorrectos

Excepciones	• **No existe ninguna intervención asociada a la incidencia**
	  ○ No es posible modificar ninguna intervención
	• **Los nuevos datos no cumplen la validación**
	  ○ Se informa al usuario con un error localizado
	• **La base de datos no se encuentra disponible**
	  ○ Se informa al usuario con un error descriptivo

Notas	El campo *Parte* de la intervención hace referencia al número de albarán que se generó en la intervención. No todas las intervenciones requieren de albarán, por lo que este campo será opcional

5.5.5.8.3 Eliminar intervención

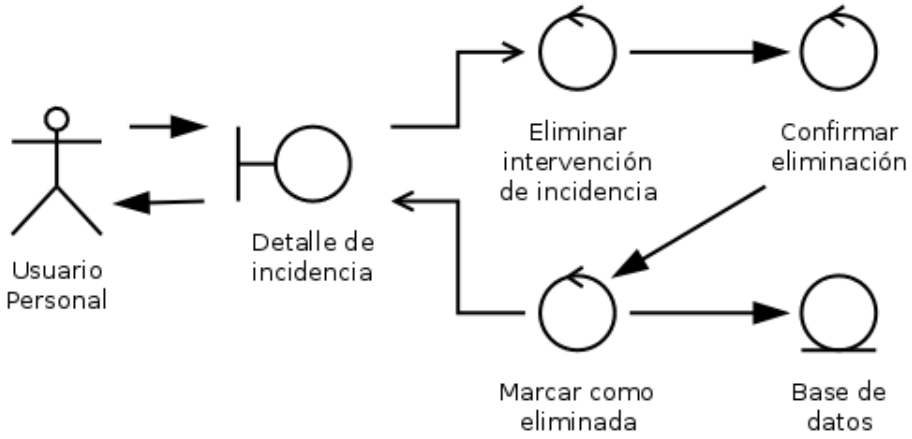


Figura 5.50. Diagrama de Robustez - Eliminar intervención

Eliminar intervención	
Precondiciones	Existe al menos una intervención en la incidencia a tratar El usuario se encuentra visualizando el detalle de la incidencia
Poscondiciones	La intervención desaparece del listado de intervenciones de la incidencia
Actores	Usuario personal
Descripción	Se trata de eliminar una intervención de la que no se precisa para el seguimiento de la intervención

Excepciones	• **No existe ninguna intervención asociada a la incidencia**
	  ○ No es posible modificar ninguna intervención
	• **La base de datos no se encuentra disponible**
	  ○ Se informa al usuario con un error descriptivo

Notas	La intervención no se elimina completamente de la base de datos. Únicamente se marca como eliminada y ya no aparecerá en la vista de la aplicación

5.5.5.8.4 Listar intervenciones

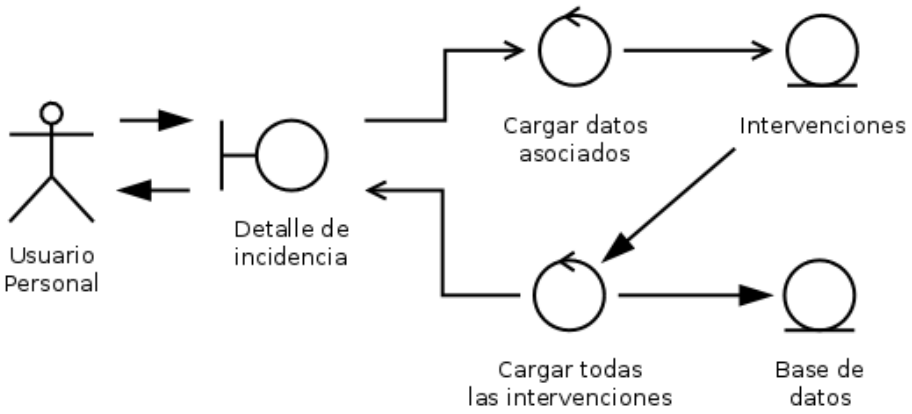


Figura 5.51. Diagrama de Robustez - Listar intervenciones

Listar intervenciones	
Precondiciones	Existe al menos una intervención en la incidencia tratada El usuario se encuentra visualizando el detalle de la incidencia
Poscondiciones	Se muestra un listado con las intervenciones llevadas a cabo en la incidencia
Actores	Usuario personal
Descripción	Se trata de consultar un listado de las intervenciones anotadas en la incidencia durante su tratamiento
Variaciones (escenarios secundarios)	• **Escenario alternativo 1** ○ La incidencia no cuenta con ninguna intervención anotada
Excepciones	• **La base de datos no se encuentra disponible** ○ Se informa al usuario con un error descriptivo

5.5.5.9 Gestionar adjuntos a la incidencia

5.5.5.9.1 Adjuntar fichero

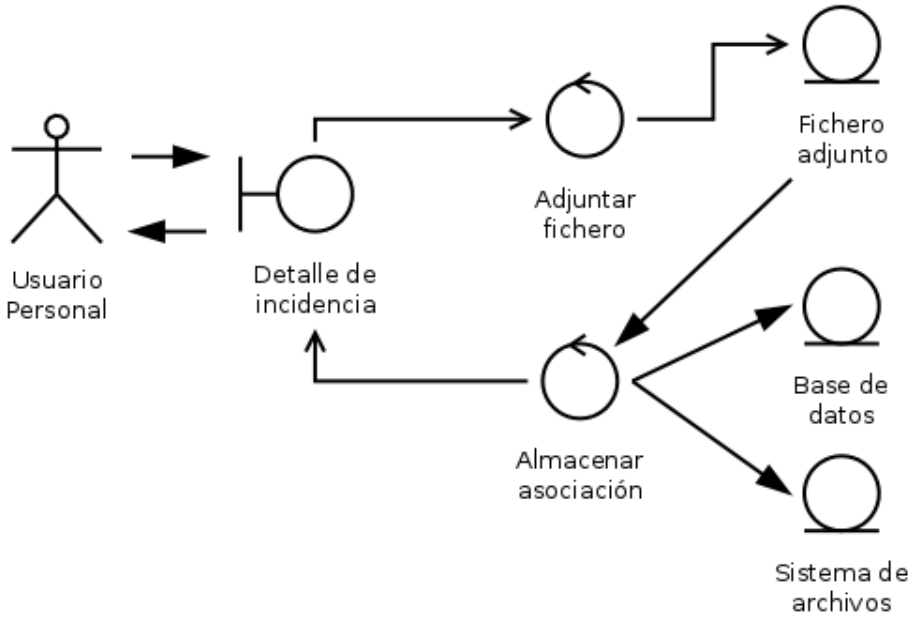


Figura 5.52. Diagrama de Robustez - Adjuntar fichero

Adjuntar fichero	
Precondiciones	Existe al menos una incidencia dada de alta en el sistema El usuario se encuentra visualizando el detalle de una incidencia
Poscondiciones	El fichero adjuntado se almacena en el servidor y se muestra una referencia al mismo en el listado de adjuntos a la incidencia
Actores	Usuario personal
Descripción	Se trata de seleccionar un fichero que resulte de interés para el tratamiento de la incidencia y adjuntarlo a la ficha de ésta. El fichero será almacenado en el servidor y se mostrará un enlace al mismo en la ficha de la incidencia para su posterior consulta

Excepciones	• **No existe ninguna incidencia en el sistema**
	  ○ No es posible anotar ninguna intervención
	• **Los datos del adjunto no cumplen la validación**
	  ○ Se informa al usuario con un error localizado
	• **Se produce un error al almacenar el fichero en el servidor**
	  ○ Se informa al usuario con un error descriptivo
	• **La base de datos no se encuentra disponible**
	  ○ Se informa al usuario con un error descriptivo

Notas	Se aceptarán ficheros de cualquier tipo y tamaño aceptados por el framework. Posteriormente, se restringirán estos criterios de acuerdo a los tipos de ficheros más utilizados y el tamaño medio de los mismos

5.5.5.9.2 Eliminar adjunto

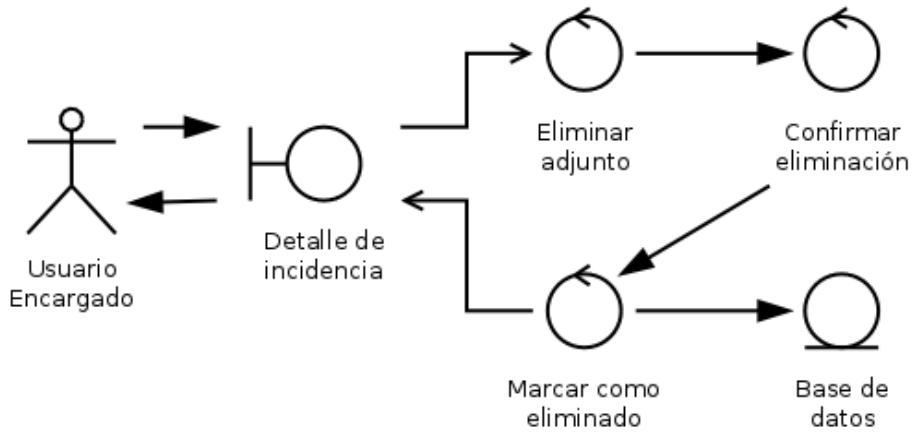


Figura 5.53. Diagrama de Robustez - Eliminar adjunto

Eliminar adjunto	
Precondiciones	Existe al menos un adjunto en la incidencia a tratar El usuario se encuentra visualizando el detalle de la incidencia
Poscondiciones	El fichero seleccionado desaparece del listado de adjuntos a la incidencia
Actores	Usuario encargado
Descripción	Se trata de eliminar un fichero adjunto a la incidencia

Excepciones	• **No existe ningún fichero adjunto a la incidencia**
	  ○ No es posible eliminar ningún adjunto de la incidencia
	• **La base de datos no se encuentra disponible**
	  ○ Se informa al usuario con un error descriptivo

Notas	La referencia al fichero adjunto no se elimina completamente de la base de datos. Únicamente se marca como eliminada y ya no aparecerá en la vista de la aplicación. El fichero en cuestión tampoco se elimina definitivamente del servidor

5.5.5.9.3 Listar adjuntos

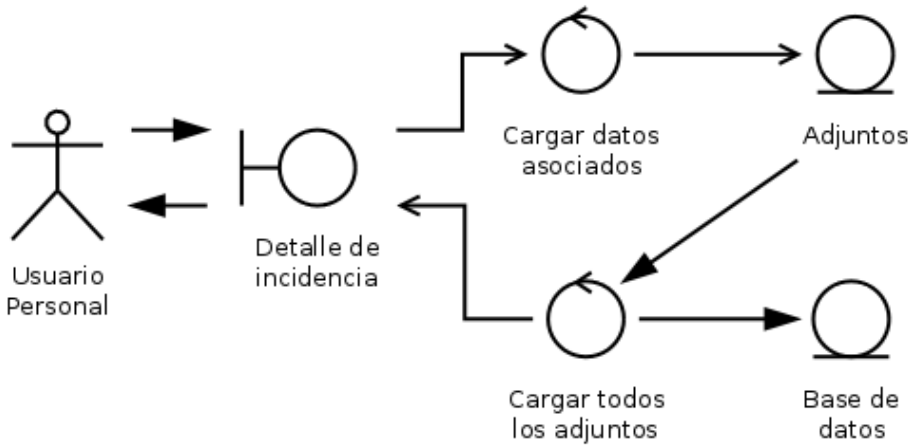


Figura 5.54. Diagrama de Robustez - Listar adjuntos

Listar adjuntos	
Precondiciones	Existe al menos una incidencia en el sistema El usuario se encuentra visualizando el detalle de la incidencia
Poscondiciones	Se muestra un listado con los ficheros adjuntos a la incidencia
Actores	Usuario personal
Descripción	Se trata de consultar un listado con las referencias a los ficheros adjuntos a la incidencia
Variaciones	• **Escenario alternativo 1** ○ La incidencia no cuenta con ningún fichero adjunto durante su tratamiento

Excepciones	• **El servidor de ficheros no se encuentra disponible**
	  ○ Se informa al usuario con un error descriptivo
	• **La base de datos no se encuentra disponible**
	  ○ Se informa al usuario con un error descriptivo

5.5.5.10 Gestionar relaciones de la incidencia

5.5.5.10.1 Relacionar con otra incidencia

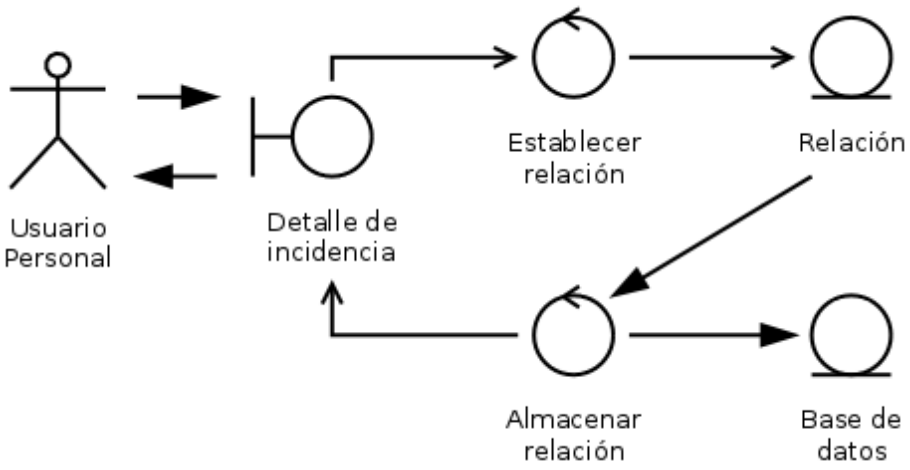


Figura 5.55. Diagrama de Robustez - Relacionar con otra incidencia

Relacionar con otra incidencia	
Precondiciones	Existen al menos dos incidencias dadas de alta en el sistema El usuario se encuentra visualizando el detalle de la incidencia a relacionar
Poscondiciones	Se establece el vínculo determinado entre la incidencia tratada y la relacionada
Actores	Usuario personal
Descripción	Se trata de modelar las relaciones existentes entre incidencias dadas de alta en el sistema. Para ello, se especificará la incidencia a vincular y el tipo de relación que existe entre ambas

Excepciones	• **No existe ninguna incidencia en el sistema a parte de la tratada**
	  ○ No es posible establecer ninguna relación
	• **La relación a establecer es idéntica a una existente entre las incidencias involucradas**
	  ○ Se informa al usuario con un error descriptivo
	• **Los datos de la relación no cumplen la validación**
	  ○ Se informa al usuario con un error localizado
	• **La base de datos no se encuentra disponible**
	  ○ Se informa al usuario con un error descriptivo

5.5.5.10.2 Eliminar relación

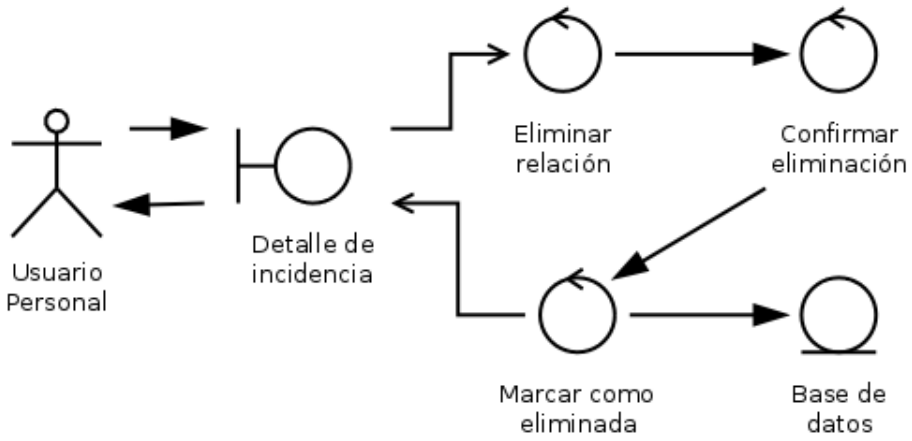


Figura 5.56. Diagrama de Robustez - Eliminar relación

Eliminar relación	
Precondiciones	Existen al menos dos incidencias relacionadas en el sistema El usuario se encuentra visualizando el detalle de la incidencia origen de la relación
Poscondiciones	La relación entre las incidencias desaparece del listado de relaciones de la incidencia en cuestión
Actores	Usuario personal
Descripción	Se trata de eliminar una relación previamente establecida entre dos incidencias existentes en el sistema

Excepciones	• **No existe ninguna relación entre incidencias**
	  ○ No es posible eliminar la relación
	• **La base de datos no se encuentra disponible**
	  ○ Se informa al usuario con un error descriptivo

5.5.5.10.3 Listar relaciones directas

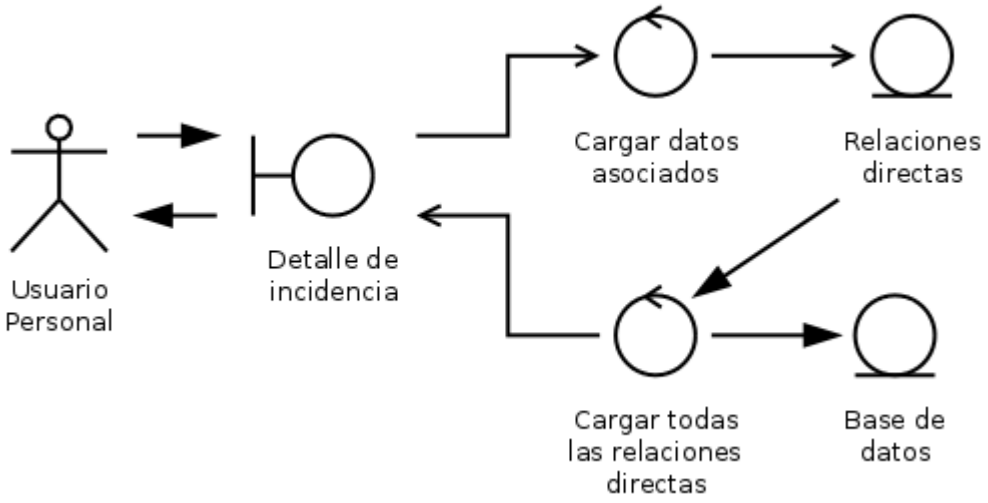


Figura 5.57. Diagrama de Robustez - Listar relaciones directas

Listar relaciones directas	
Precondiciones	El usuario se encuentra visualizando el detalle de una incidencia
Poscondiciones	Se muestra un listado con las relaciones directas de la incidencia en cuestión
Actores	Usuario personal
Descripción	Se trata de consultar un listado con las relaciones en las que la incidencia visualizada actúa como origen
Variaciones	• **Escenario alternativo 1** ○ La incidencia no cuenta con ninguna relación directa
Excepciones	• **La base de datos no se encuentra disponible** ○ Se informa al usuario con un error descriptivo
Notas	Una incidencia *X* tiene una relación directa con una *Y* si el origen de dicha relación se encuentra en la primera

5.5.5.10.4 Listar relaciones inversas

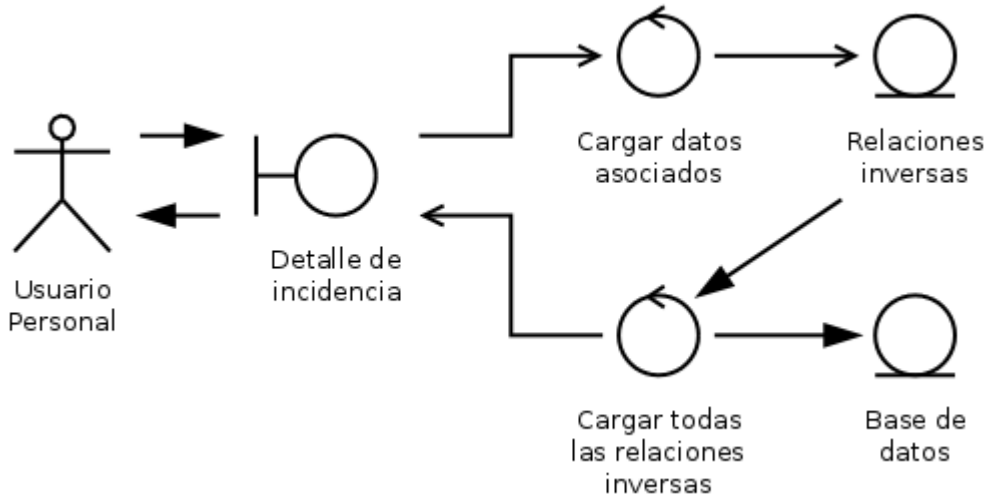


Figura 5.58. Diagrama de Robustez - Listar relaciones inversas

Listar relaciones inversas	
Precondiciones	El usuario se encuentra visualizando el detalle de una incidencia
Poscondiciones	Se muestra un listado con las relaciones inversas de la incidencia en cuestión
Actores	Usuario personal
Descripción	Se trata de consultar un listado con las relaciones en las que la incidencia visualizada es la parte destino de la relación
Variaciones	• **Escenario alternativo 1** ○ La incidencia no cuenta con ninguna relación inversa
Excepciones	• **La base de datos no se encuentra disponible** ○ Se informa al usuario con un error descriptivo
Notas	Una incidencia *X* tiene una relación inversa con una *Y* si el origen de dicha relación se encuentra en la segunda

5.6 Análisis de Interfaces de Usuario

Uno de los acuerdos establecidos con el cliente de este proyecto ha sido el de declinar la responsabilidad del diseño de las interfaces de usuario de la aplicación al departamento de diseño gráfico de la empresa. Es por tanto que el análisis de las mismas no será objeto de estudio en esta sección.

La empresa, siguiendo con los procedimientos y estilos establecidos para este tipo de tareas, diseñó las interfaces de usuario en consonancia con la apariencia general de su sitio corporativo, por lo que la tarea del programador en este aspecto se basará exclusivamente en adaptar el desarrollo a los bocetos creados.

El punto de partida para el esbozo de las interfaces se basará en la siguiente distribución de elementos, común a todas las vistas de la aplicación.

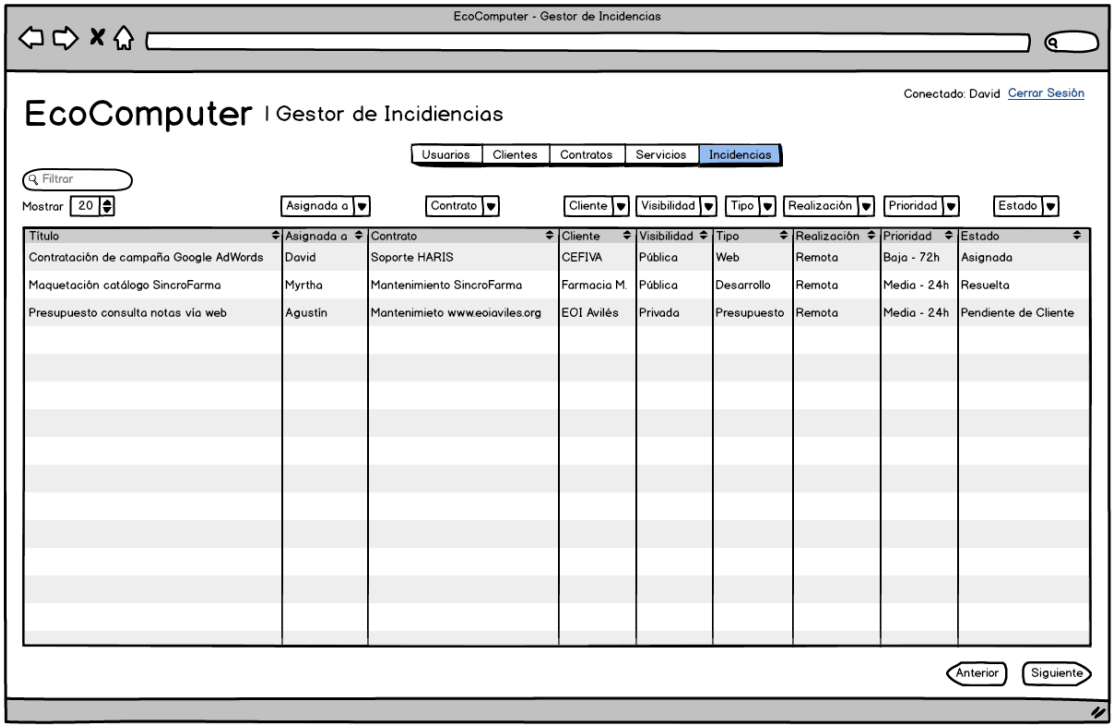


Figura 5.59. Esbozo general de la interfaz de usuario

En el siguiente capítulo se presentan los bocetos recibidos por parte del departamento de diseño.

5.7 Especificación del Plan de Pruebas

Como última fase del análisis del sistema, se presenta a continuación una especificación del plan de pruebas a ejecutar durante el desarrollo del sistema. Esta especificación se divide de acuerdo a los módulos que componen el sistema en su totalidad, especificando para cada ensayo, su fundamento y el resultado esperado de su ejecución.

5.7.1 Pruebas del Módulo de Gestión de Usuarios

Caso de Uso 1.1. Iniciar sesión	
Prueba	**Resultado Esperado**
Iniciar sesión en el sistema con unas credenciales válidas	El usuario es reconocido y se inicia la sesión
Prueba	**Resultado Esperado**
Iniciar sesión en el sistema con unas credenciales no válidas	Se muestra un mensaje de error

Caso de Uso 1.2. Cerrar sesión	
Prueba	**Resultado Esperado**
Cerrar la sesión de usuario en curso	El sistema cierra correctamente la sesión y muestra la pantalla de inicio

5.7.2 Pruebas del Módulo de Gestión de Clientes

Caso de Uso 2.1. Crear cliente	
Prueba	**Resultado Esperado**
Completar todos los datos del formulario correctamente	El cliente es almacenado en el sistema y aparece en el listado de clientes
Prueba	**Resultado Esperado**
Completar incorrectamente el formulario	Se muestran los errores de validación encontrados en cada campo del formulario

Caso de Uso 2.2. Modificar cliente	
Prueba	**Resultado Esperado**
Introducir los nuevos datos del cliente correctamente	La información del cliente es actualizada y se muestra correctamente en su ficha
Prueba	**Resultado Esperado**
Modificar incorrectamente el formulario	Se muestran los errores de validación encontrados en cada campo del formulario

Caso de Uso 2.3. Eliminar cliente	
Prueba	**Resultado Esperado**
Eliminar el cliente y confirmar la acción	El cliente es dado de baja en el sistema y ya no aparece en el listado de clientes
Prueba	**Resultado Esperado**
Eliminar el cliente sin confirmar la acción	La acción se cancela y se retorna a la vista anterior

Caso de Uso 2.4. Consultar listado de clientes	
Prueba	**Resultado Esperado**
Acceder al módulo de gestión de clientes	Se muestra el listado de clientes dados de alta en el sistema

Caso de Uso 2.4.1. Filtrar registros de clientes	
Prueba	**Resultado Esperado**
Introducir los parámetros de búsqueda deseados en el filtro	Se muestran únicamente los clientes cuyos datos contienen los parámetros especificados

Caso de Uso 2.4.2. Ordenar registros de clientes	
Prueba	**Resultado Esperado**
Seleccionar la columna por la que se desea ordenar	Los clientes aparecen ordenados de acuerdo al campo seleccionado

Caso de Uso 2.5. Consultar detalle del cliente	
Prueba	**Resultado Esperado**
Seleccionar el cliente	Se muestra la ficha completa del cliente seleccionado

deseado del listado	

Caso de Uso 2.6. Ver incidencias del cliente	
Prueba	**Resultado Esperado**
Visualizando la ficha del cliente, seleccionar el enlace *Ver Incidencias*	Se muestra el listado de incidencias comunicadas por el cliente seleccionado

5.7.3 Pruebas del Módulo de Gestión de Contratos

Caso de Uso 3.1. Crear contrato	
Prueba	**Resultado Esperado**
Completar todos los datos del formulario correctamente	El contrato es almacenado en el sistema y aparece en el listado de contratos
Prueba	**Resultado Esperado**
Completar incorrectamente el formulario	Se muestran los errores de validación encontrados en cada campo del formulario

Caso de Uso 3.2. Modificar contrato	
Prueba	**Resultado Esperado**
Introducir los nuevos datos del contrato correctamente	La información del contrato es actualizada y se muestra correctamente en su ficha
Prueba	**Resultado Esperado**
Modificar incorrectamente el formulario	Se muestran los errores de validación encontrados en cada campo del formulario

Caso de Uso 3.3. Eliminar contrato	
Prueba	**Resultado Esperado**
Eliminar el contrato y confirmar la acción	El contrato es dado de baja en el sistema y ya no aparece en el listado de contratos
Prueba	**Resultado Esperado**
Eliminar el contrato sin confirmar la acción	La acción se cancela y se retorna a la vista anterior

Casos de Uso 3.4. Gestionar servicios del contrato

Caso de Uso 3.4.1. Incluir servicio	
Prueba	**Resultado Esperado**
Completar correctamente todos los campos del formulario	El servicio es asociado al contrato y se muestra en su ficha
Prueba	**Resultado Esperado**
Completar incorrectamente el formulario	Se muestran los errores de validación encontrados en cada campo del formulario

Caso de Uso 3.4.2. Modificar servicio asociado	
Prueba	**Resultado Esperado**
Introducir los nuevos datos del servicio asociado correctamente	La información del servicio contratado es actualizada y se muestra correctamente en la ficha del contrato
Prueba	**Resultado Esperado**
Modificar incorrectamente el formulario	Se muestran los errores de validación encontrados en cada campo del formulario

Caso de Uso 3.4.3. Desvincular servicio	
Prueba	**Resultado Esperado**
Eliminar el servicio contratado y confirmar la acción	El servicio se desvincula del contrato y no aparece en su ficha
Prueba	**Resultado Esperado**
Eliminar el servicio del contrato sin confirmar la acción	La acción se cancela y se retorna a la vista anterior

Caso de Uso 3.4.4. Listar servicios asociados	
Prueba	**Resultado Esperado**
Consultar el detalle de un contrato con servicios asociados	Se muestra, además de los datos básicos del contrato, un listado con los servicios asociados al mismo
Prueba	**Resultado Esperado**
Consultar el detalle de un contrato sin servicios asociados	Se muestran únicamente los datos básicos del contrato

Caso de Uso 3.5. Consultar listado de contratos	
Prueba	**Resultado Esperado**
Acceder al módulo de gestión de contratos	Se muestra el listado de contratos suscritos por clientes que han sido almacenados en el sistema

Caso de Uso 3.5.1. Filtrar registros de contratos	
Prueba	**Resultado Esperado**
Introducir los parámetros de búsqueda deseados en el filtro	Se muestran únicamente los contratos cuyos datos contienen los parámetros especificados

Caso de Uso 3.5.2. Ordenar registros de contratos	
Prueba	**Resultado Esperado**
Seleccionar la columna por la que se desea ordenar	Los contratos aparecen ordenados de acuerdo al campo seleccionado

Caso de Uso 3.6. Consultar detalle del contrato	
Prueba	**Resultado Esperado**
Seleccionar el contrato deseado del listado	Se muestra la ficha completa del contrato seleccionado, incluyendo sus servicios asociados

5.7.4 Pruebas del Módulo de Gestión de Servicios

Caso de Uso 4.1. Crear servicio	
Prueba	**Resultado Esperado**
Completar todos los datos del formulario correctamente	El servicio es almacenado en el sistema y aparece en el catálogo de servicios
Prueba	**Resultado Esperado**
Completar incorrectamente el formulario	Se muestran los errores de validación encontrados en cada campo del formulario

Caso de Uso 4.2. Modificar servicio	
Prueba	**Resultado Esperado**
Introducir los nuevos datos del servicio correctamente	La información del servicio es actualizada y se muestra correctamente en el catálogo de servicios
Prueba	**Resultado Esperado**
Modificar incorrectamente el formulario	Se muestran los errores de validación encontrados en cada campo del formulario

Caso de Uso 4.3. Eliminar servicio	
Prueba	**Resultado Esperado**
Eliminar el servicio y confirmar la acción	El servicio es dado de baja en el sistema y ya no aparece en el catálogo de servicios
Prueba	**Resultado Esperado**
Eliminar el servicio sin confirmar la acción	La acción se cancela y se retorna a la vista anterior

Casos de Uso 4.4. Gestionar tarifas del servicio

Caso de Uso 4.4.1. Incluir tarifa	
Prueba	**Resultado Esperado**
Completar correctamente todos los campos del formulario	La tarifa es asociada al servicio y se muestra en el catálogo
Prueba	**Resultado Esperado**
Completar incorrectamente el formulario	Se muestran los errores de validación encontrados en cada campo del formulario

Caso de Uso 4.4.2. Modificar tarifa	
Prueba	**Resultado Esperado**
Introducir los nuevos datos de la tarifa del servicio correctamente	La información de la tarifa es actualizada en el servicio y se muestra correctamente en el catálogo
Prueba	**Resultado Esperado**
Modificar incorrectamente el formulario	Se muestran los errores de validación encontrados en cada campo del formulario

Caso de Uso 4.4.3. Eliminar tarifa	
Prueba	**Resultado Esperado**
Eliminar una tarifa asociada a un servicio y confirmar la acción	La tarifa se desvincula del servicio y no aparece en el catálogo
Prueba	**Resultado Esperado**
Eliminar una tarifa asociada a un servicio sin confirmar la acción	La acción se cancela y se retorna a la vista anterior

Caso de Uso 4.4.4. Listar tarifas	
Prueba	**Resultado Esperado**
Consultar el detalle de un servicio con tarifas asociadas	Se muestra, además de la descripción del servicio, un listado con las tarifas aplicables al mismo
Prueba	**Resultado Esperado**
Consultar el detalle de un servicio sin tarifas asociadas	Se muestra únicamente la descripción del servicio

Caso de Uso 4.4.4.1. Filtrar registros de tarifas	
Prueba	**Resultado Esperado**
Introducir los parámetros de búsqueda deseados en el filtro	Se muestran únicamente las tarifas cuyos datos contienen los parámetros especificados

Caso de Uso 4.4.4.2. Ordenar registros de tarifas	
Prueba	**Resultado Esperado**
Seleccionar la columna por la que se desea ordenar	Las tarifas del servicio en cuestión aparecen ordenadas de acuerdo al campo seleccionado

Caso de Uso 4.5. Consultar listado de servicios	
Prueba	**Resultado Esperado**
Acceder al módulo de gestión de servicios	Se muestra el catálogo de servicios ofrecidos por la empresa que han sido incluidos en el sistema

5.7.5 Pruebas del Módulo de Gestión de Incidencias

Caso de Uso 5.1. Crear incidencia	
Prueba	**Resultado Esperado**
Completar todos los datos del formulario correctamente	La incidencia es almacenada en el sistema y aparece en el listado de incidencias
Prueba	**Resultado Esperado**
Completar incorrectamente el formulario	Se muestran los errores de validación encontrados en cada campo del formulario

Caso de Uso 5.2. Modificar incidencia	
Prueba	**Resultado Esperado**
Introducir los nuevos datos de la incidencia correctamente	La información de la incidencia es actualizada y se muestra correctamente en su ficha
Prueba	**Resultado Esperado**
Modificar incorrectamente el formulario	Se muestran los errores de validación encontrados en cada campo del formulario

Caso de Uso 5.3. Eliminar incidencia	
Prueba	**Resultado Esperado**
Eliminar la incidencia y confirmar la acción	La incidencia es dada de baja en el sistema y ya no aparece en el listado de incidencias
Prueba	**Resultado Esperado**
Eliminar la incidencia sin confirmar la acción	La acción se cancela y se retorna a la vista anterior

Caso de Uso 5.4. Consultar detalle de incidencia	
Prueba	**Resultado Esperado**
Seleccionar la incidencia deseada del listado	Se muestra la ficha completa de la incidencia seleccionada, incluyendo sus elementos asociados (intervenciones, adjuntos y relaciones)

Caso de Uso 5.5. Clonar incidencia	
Prueba	**Resultado Esperado**
Seleccionar la opción Clonar en la vista detalle de una incidencia	Aparece una nueva incidencia en el listado del sistema con los mismos datos básicos que la original

Caso de Uso 5.6. Generar informe imprimible	
Prueba	**Resultado Esperado**
Seleccionar la opción *Generar Informe* en la vista detalle de una incidencia y especificar como destino un fichero PDF	Se genera un documento PDF con toda la información de la incidencia en la ruta indicada
Prueba	**Resultado Esperado**
Seleccionar la opción *Generar Informe* en la vista detalle de una incidencia y especificar como destino una impresora conectada al equipo del usuario	Se imprime el informe completo de la incidencia en la impresora seleccionada

Caso de Uso 5.7. Listar incidencias	
Prueba	**Resultado Esperado**
Acceder al módulo de gestión de incidencias	Se muestra el listado de incidencias almacenadas en el sistema, ordenadas del siguiente modo: • Primero, aquellas que han sido asignadas al usuario en sesión • Segundo, aquellas con la prioridad de resolución más alta

Caso de Uso 5.7.1. Filtrar registros de incidencias	
Prueba	**Resultado Esperado**
Introducir los parámetros de búsqueda deseados en el filtro	Se muestran únicamente las incidencias cuyos datos contienen los parámetros especificados

Caso de Uso 5.7.2. Ordenar registros de incidencias	
Prueba	**Resultado Esperado**
Seleccionar la columna por la que se desea ordenar	Las incidencias aparecen ordenadas de acuerdo al campo seleccionado

Casos de Uso 5.8. Gestionar intervenciones de la incidencia

Caso de Uso 5.8.1. Incluir intervención	
Prueba	**Resultado Esperado**
Completar correctamente todos los datos del formulario	La intervención es anotada en la incidencia y se muestra en su ficha
Prueba	**Resultado Esperado**
Completar incorrectamente el formulario	Se muestran los errores de validación encontrados en cada campo del formulario

Caso de Uso 5.8.2. Modificar intervención	
Prueba	**Resultado Esperado**
Introducir los nuevos datos de la intervención correctamente	La información de la intervención es actualizada y se muestra correctamente en la ficha de la incidencia
Prueba	**Resultado Esperado**
Modificar incorrectamente el formulario	Se muestran los errores de validación encontrados en cada campo del formulario

Caso de Uso 5.8.3. Eliminar intervención	
Prueba	**Resultado Esperado**
Eliminar una intervención anotada en una incidencia y confirmar la acción	La intervención se elimina de la incidencia y no aparece en su ficha
Prueba	**Resultado Esperado**
Eliminar una intervención anotada en una incidencia sin confirmar la acción	La acción se cancela y se retorna a la vista anterior

Caso de Uso 5.8.4. Listar intervenciones	
Prueba	**Resultado Esperado**
Consultar el detalle de una incidencia con intervenciones asociadas	Se muestra, además de los datos básicos de la incidencia, un listado con las intervenciones realizadas en la incidencia
Prueba	**Resultado Esperado**
Consultar el detalle de una incidencia con intervenciones asociadas	Se muestran únicamente los datos básicos de la incidencia

Casos de Uso 5.9. Gestionar adjuntos a la incidencia

Caso de Uso 5.9.1. Adjuntar fichero	
Prueba	**Resultado Esperado**
Completar correctamente todos los campos del formulario y seleccionar un fichero con formato y tamaño admitidos por el servidor	El fichero se adjunta a la incidencia y se muestra una referencia al mismo en la ficha de la incidencia
Prueba	**Resultado Esperado**
Completar incorrectamente el formulario y/o seleccionar un fichero con formato y/o tamaño no admitidos por el servidor	Se muestran los errores de validación encontrados en cada campo del formulario y/o en el control de subida del fichero

Caso de Uso 5.9.2. Eliminar adjunto	
Prueba	**Resultado Esperado**
Eliminar el fichero adjunto de la incidencia y confirmar la acción	El fichero se desvincula de la incidencia y no aparece en su ficha
Prueba	**Resultado Esperado**
Eliminar el fichero adjunto de la incidencia sin confirmar la acción	La acción se cancela y se retorna a la vista anterior

Caso de Uso 5.9.3. Listar adjuntos	
Prueba	**Resultado Esperado**
Consultar el detalle de una incidencia con ficheros adjuntos	Se muestra, además de los datos básicos de la incidencia, un listado con los ficheros adjuntos a la misma
Prueba	**Resultado Esperado**
Consultar el detalle de una incidencia sin ficheros adjuntos	Se muestran únicamente los datos básicos de la incidencia

Casos de Uso 5.10. Gestionar relaciones de la incidencia

Caso de Uso 5.10.1. Relacionar con otra incidencia	
Prueba	**Resultado Esperado**
Relacionar correctamente una incidencia con otra existente y no relacionada anteriormente	La relación se establece correctamente y se muestra en la ficha de la incidencia
Prueba	**Resultado Esperado**
Relacionar una incidencia cuando no existe ninguna otra en el sistema	No es posible establecer ninguna relación entre incidencias
Prueba	**Resultado Esperado**
Crear una relación idéntica a una ya existente	Se muestra un error indicando que la relación ya se ha establecido anteriormente
Prueba	**Resultado Esperado**
Completar incorrectamente el formulario	Se muestran los errores de validación encontrados en cada campo del formulario

Caso de Uso 5.10.2. Eliminar relación	
Prueba	**Resultado Esperado**
Eliminar la relación de la incidencia y confirmar la acción	La relación se elimina de la incidencia y no aparece en su ficha
Prueba	**Resultado Esperado**
Eliminar la relación de la incidencia sin confirmar la	La acción se cancela y se retorna a la vista anterior

acción	
Caso de Uso 5.10.3. Listar relaciones directas	
Prueba	**Resultado Esperado**
Consultar el detalle de una incidencia con relaciones directas asociadas	Se muestra, además de los datos básicos de la incidencia, un listado con las relaciones directas de la misma
Prueba	**Resultado Esperado**
Consultar el detalle de una incidencia sin relaciones directas asociadas	Se muestran únicamente los datos básicos de la incidencia

Caso de Uso 5.10.4. Listar relaciones inversas	
Prueba	**Resultado Esperado**
Consultar el detalle de una incidencia con relaciones inversas asociadas	Se muestra, además de los datos básicos de la incidencia, un listado con las relaciones inversas de la misma
Prueba	**Resultado Esperado**
Consultar el detalle de una incidencia sin relaciones inversas asociadas	Se muestran únicamente los datos básicos de la incidencia

Capítulo 6. Diseño del Sistema

6.1 Arquitectura del Sistema

6.1.1 Diagramas de Paquetes

Debido a la extensión del sistema, se presentarán los paquetes del mismo de forma fraccionada. En primer lugar, se muestra una vista general de los paquetes que componen el sistema, presentando únicamente los paquetes de más alto nivel para, posteriormente, detallar el contenido de cada uno de ellos en vistas individuales con mayor profundidad.

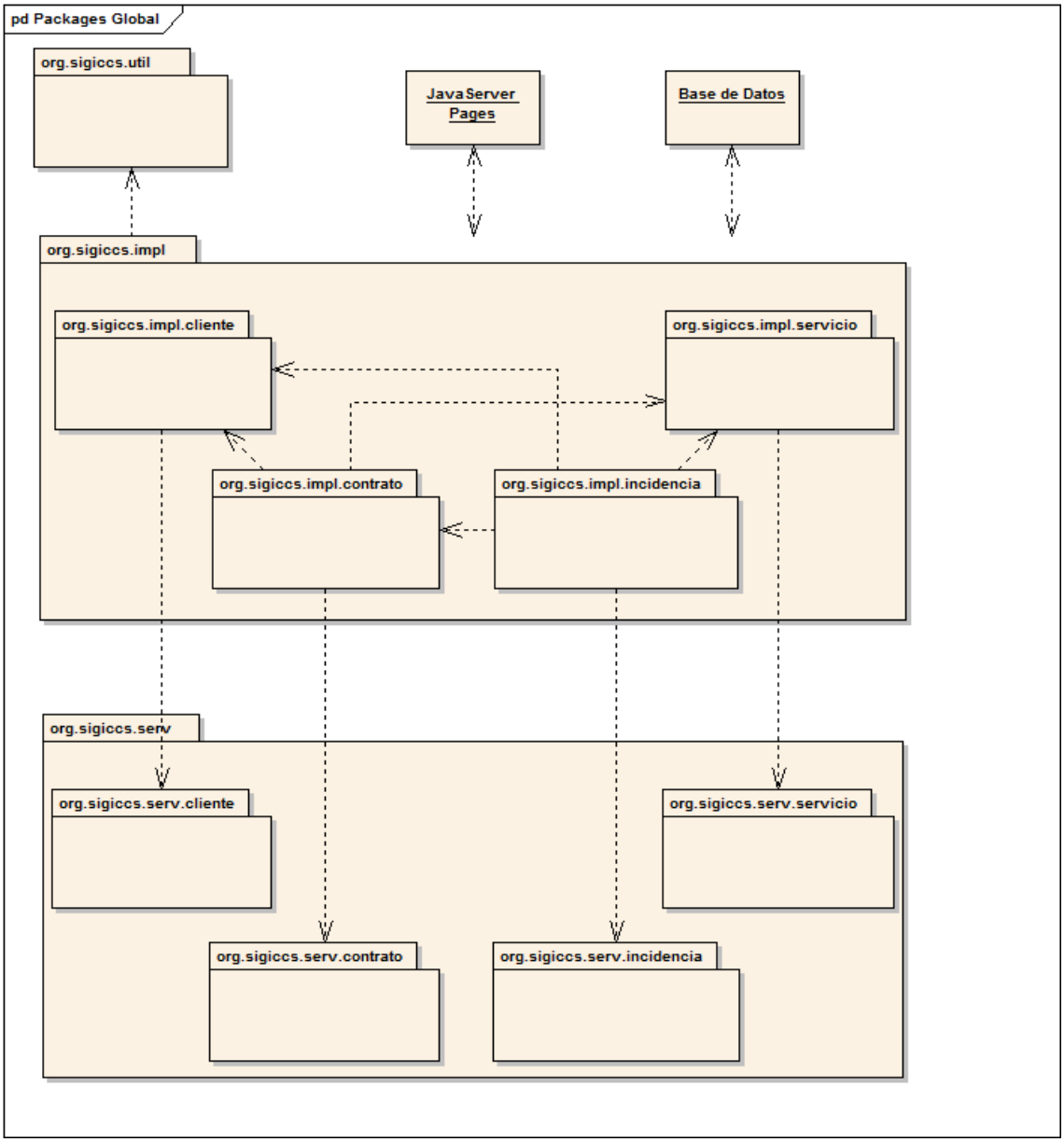


Figura 6.1. Diagrama de Paquetes global del sistema

Como puede observarse, los paquetes de la aplicación se han organizado siguiendo las directrices marcadas por la aplicación del patrón de diseño Modelo-Vista-Controlador. A continuación se ofrece una descripción de cada conjunto de paquetes.

El conjunto **org.sigiccs.serv** agrupa los paquetes contenedores de las clases que implementan los elementos del modelo de dominio, así como las interfaces de las capas de negocio y persistencia del módulo correspondiente.

6.1.1.1 Paquete org.sigiccs.serv.cliente

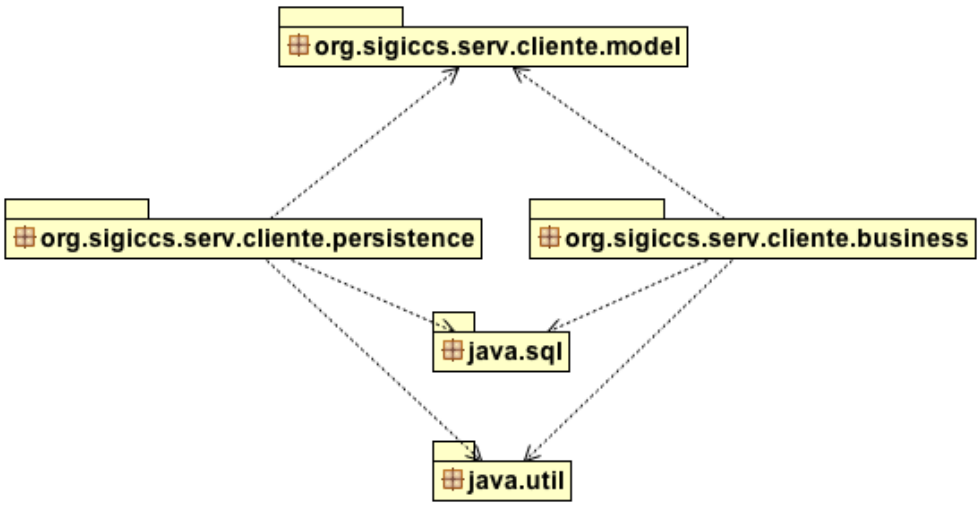


Figura 6.2. Paquete org.sigiccs.serv.cliente

Contiene las interfaces de la capa de negocio y de persistencia, así como las clases que modelan los elementos propios del módulo de gestión de clientes.

6.1.1.2 Paquete org.sigiccs.serv.contrato

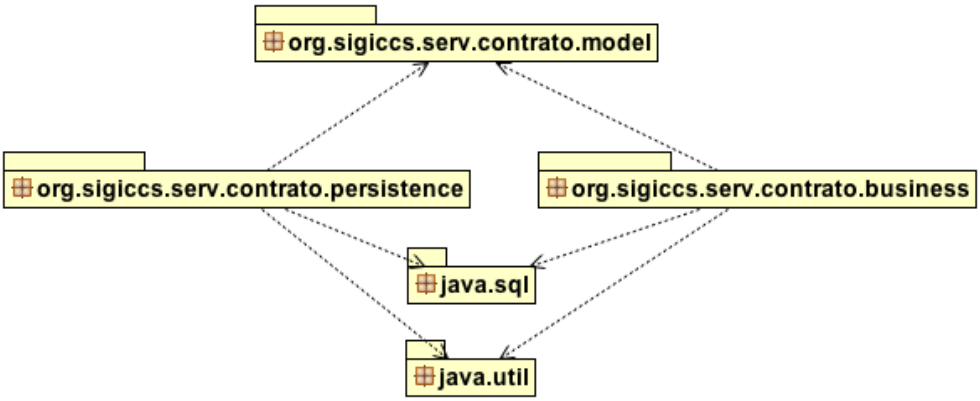


Figura 6.3. Paquete org.sigiccs.serv.contrato

Contiene las interfaces de la capa de negocio y de persistencia, así como las clases que modelan los elementos propios del módulo de gestión de contratos, incluyendo las asociaciones contrato-servicio.

6.1.1.3 Paquete org.sigiccs.serv.servicio

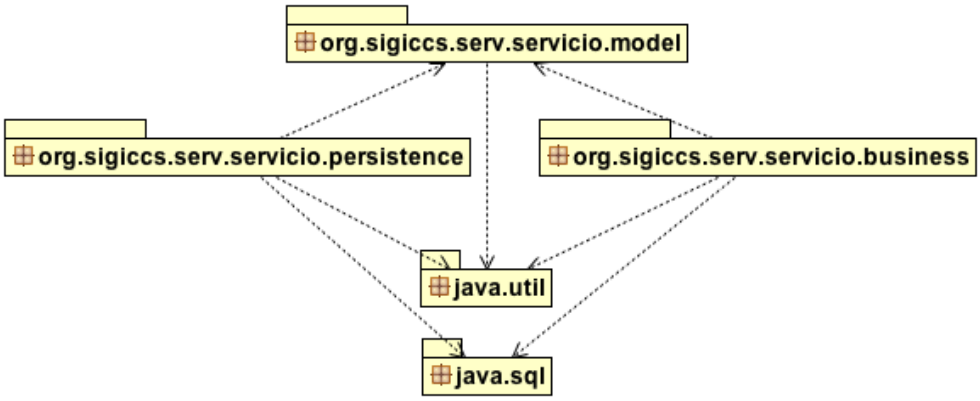


Figura 6.4. Paquete org.sigiccs.serv.servicio

Contiene las interfaces de la capa de negocio y de persistencia, así como las clases que modelan los elementos propios del módulo de gestión de servicios, incluyendo las asociaciones servicio-tarifa.

6.1.1.4 Paquete org.sigiccs.serv.incidencia

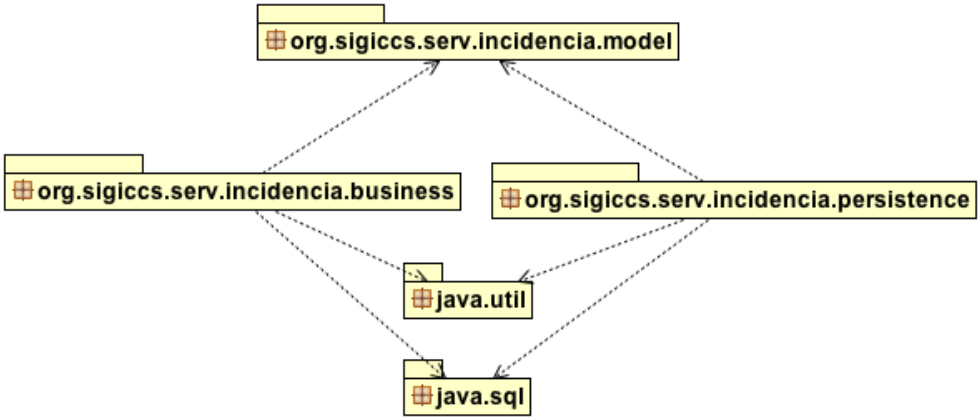


Figura 6.5. Paquete org.sigiccs.serv.incidencia

Contiene las interfaces de la capa de negocio y de persistencia, así como las clases que modelan los elementos propios del módulo de gestión de incidencias, incluyendo las asociaciones incidencia-intervención, incidencia-adjunto, incidencia-relación, incidencia-tipo e incidencia-prioridad.

El conjunto **org.sigiccs.impl** agrupa los paquetes contenedores de las clases que implementan las interfaces definidas en el paquete *org.sigiccs.serv*, así como la implementación de las Actions encargadas de llevar a cabo la ejecución de la acción correspondiente a cada petición, incluyendo los ficheros de validación asociados a cada una de ellas.

6.1.1.5 Paquete org.sigiccs.impl.cliente

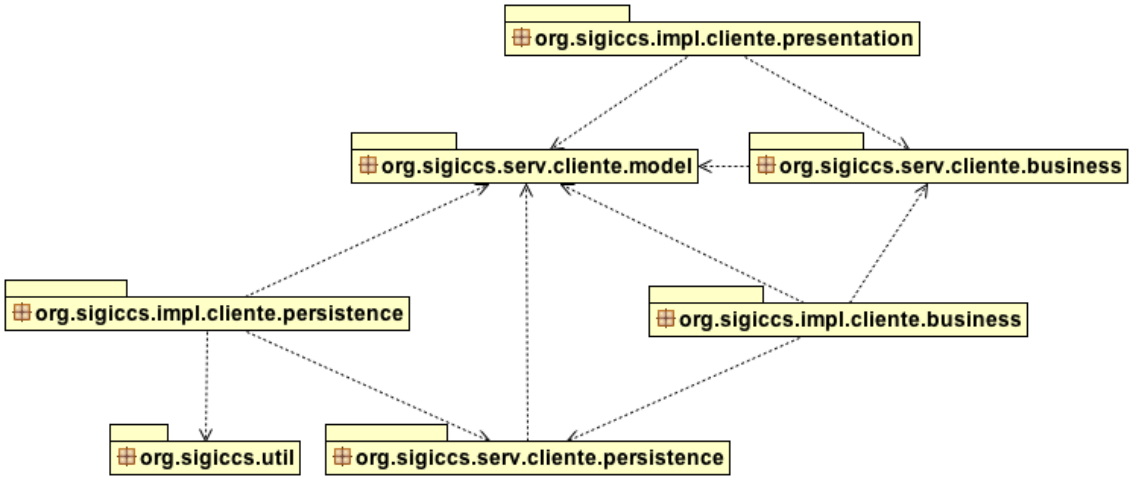


Figura 6.6. Paquete org.sigiccs.impl.cliente

Contiene la implementación de las interfaces de la capa de negocio y de persistencia del módulo de gestión de clientes, así como las Actions encargadas de ejecutar la lógica propia de estas entidades.

6.1.1.6 Paquete org.sigiccs.impl.contrato

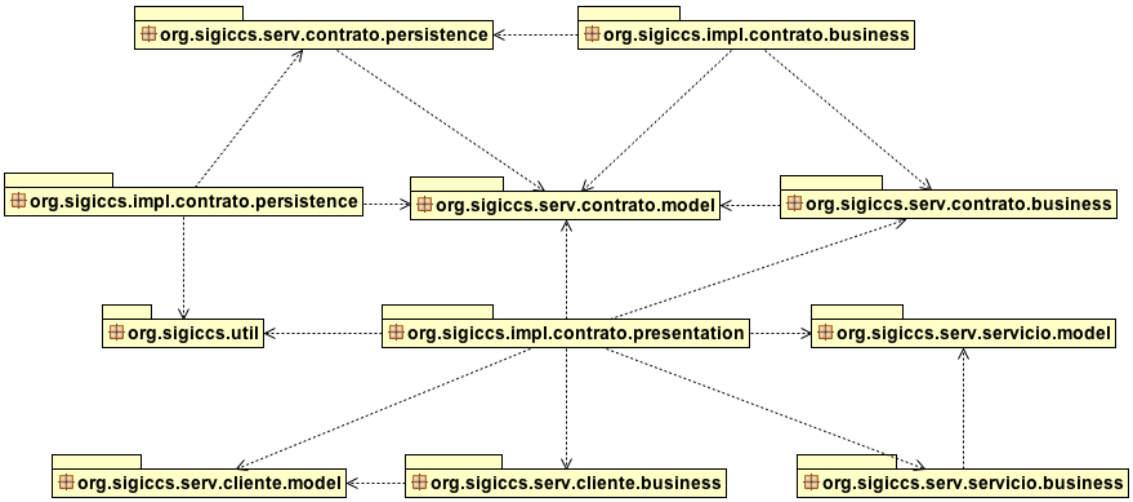


Figura 6.7. Paquete org.sigiccs.impl.contrato

Contiene la implementación de las interfaces de la capa de negocio y de persistencia del módulo de gestión de contratos, así como las Actions encargadas de ejecutar la lógica propia de estas entidades y las relacionadas.

6.1.1.7 Paquete org.sigiccs.impl.servicio

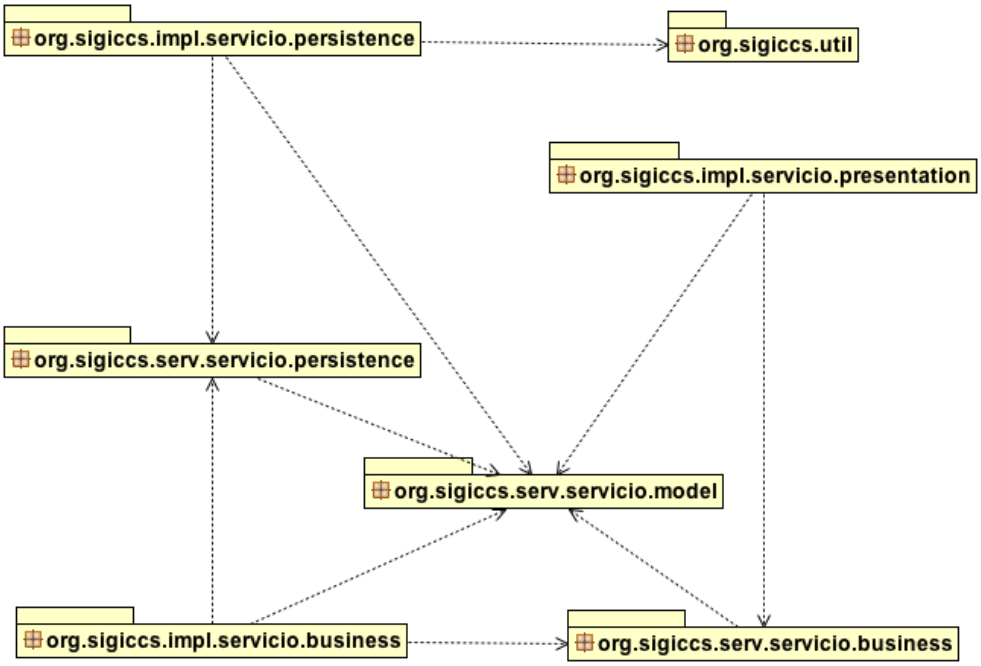


Figura 6.8. Paquete org.sigiccs.impl.servicio

Contiene la implementación de las interfaces de la capa de negocio y de persistencia del módulo de gestión de servicios, así como las Actions encargadas de ejecutar la lógica propia de estas entidades y las relacionadas.

6.1.1.8 Paquete org.sigiccs.impl.incidencia

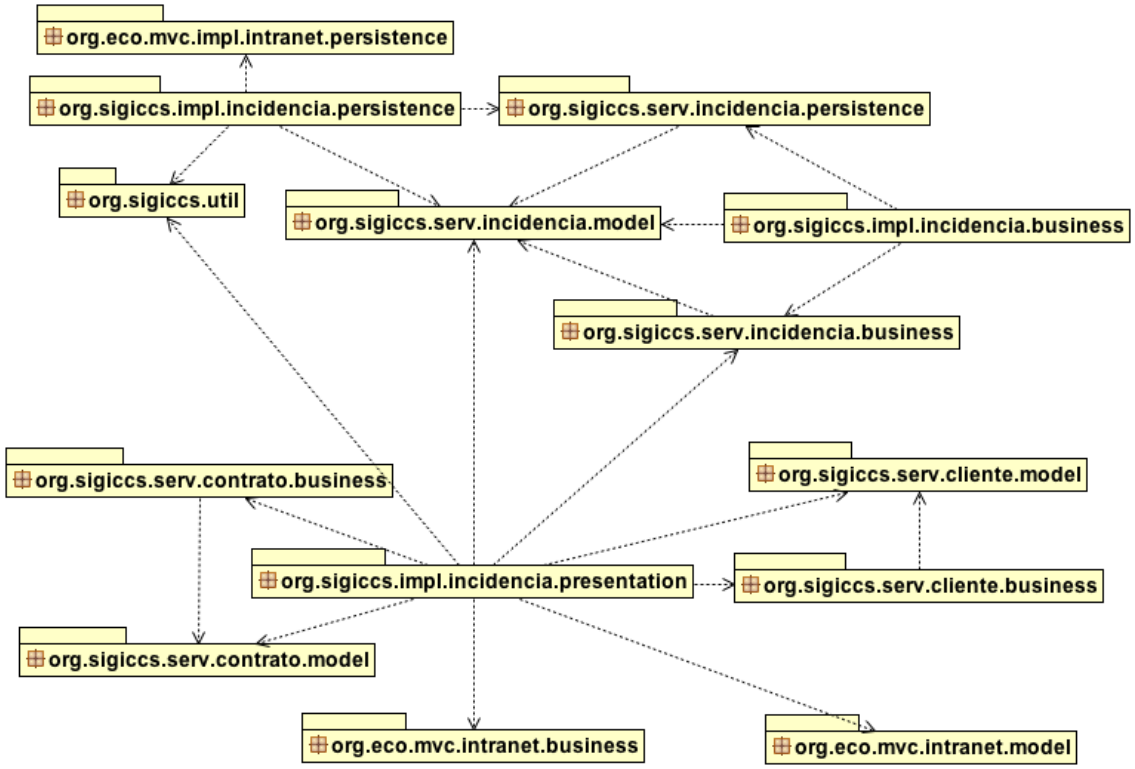


Figura 6.9. Paquete org.sigiccs.impl.incidencia

Contiene la implementación de las interfaces de la capa de negocio y de persistencia del módulo de gestión de incidencias, así como las Actions encargadas de ejecutar la lógica propia de estas entidades y las relacionadas.

6.1.1.9 Paquete org.sigiccs.util

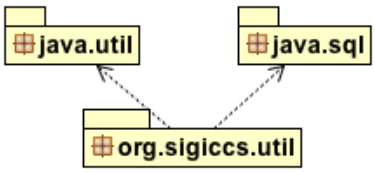


Figura 6.10. Paquete org.sigiccs.util

Finalmente, el paquete **org.sigiccs.util** contiene clases de utilidad empleadas en el resto de la implementación, tales como el gestor de conexiones con la base de datos y un comparador de valores empleado en diversas operaciones complejas de las Actions.

6.1.2 Diagramas de Componentes

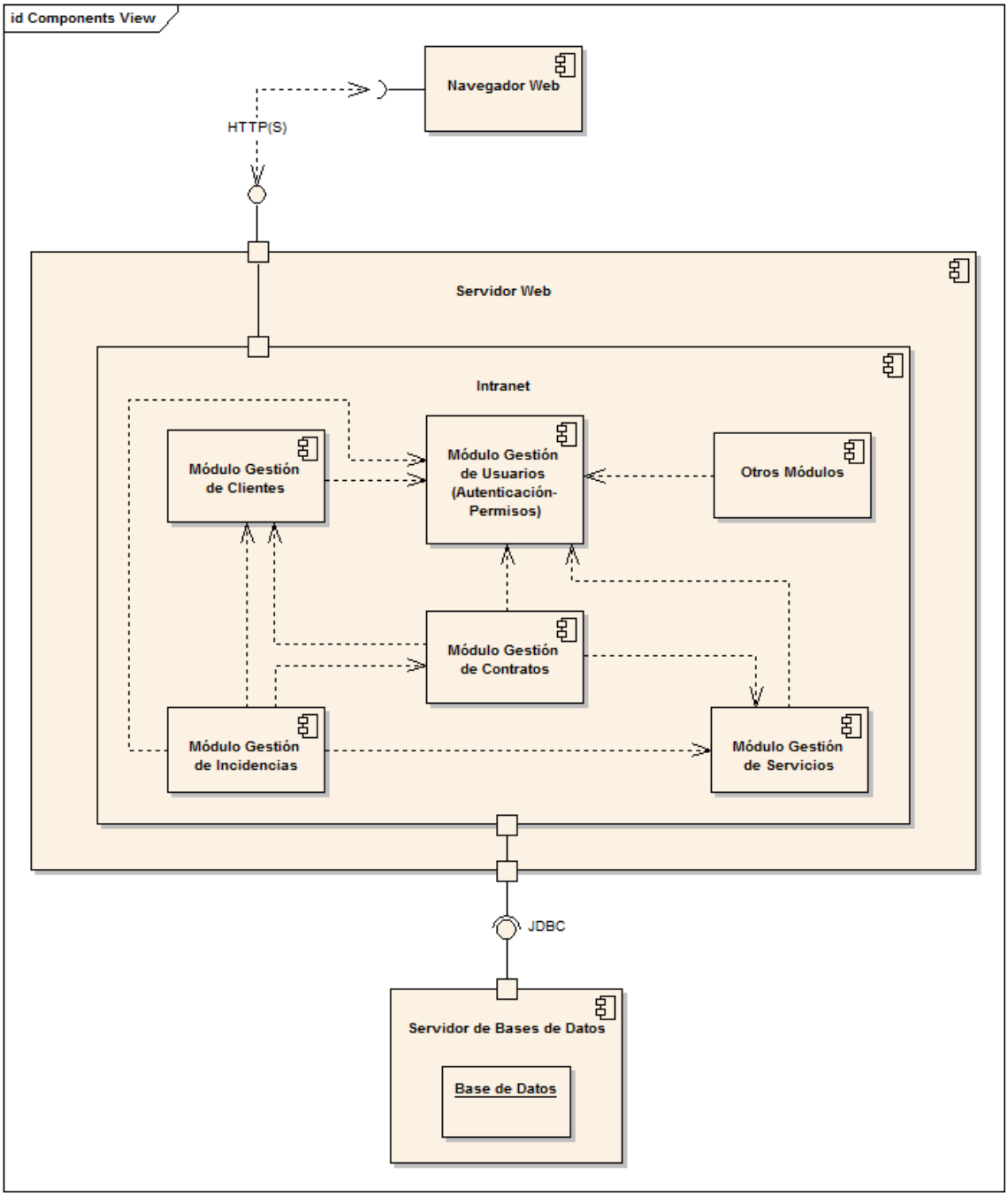


Figura 6.11. Diagrama de Componentes

Este diagrama representa la organización de los componentes lógicos del sistema que, junto con las interconexiones y dependencias marcadas entre ellos, reflejan la arquitectura general planteada.

Todos los módulos que componen la aplicación se encuentran contenidos en la **Intranet**, desplegado todo ello a su vez en un **Servidor Web**, el cual es capaz de comunicarse con el **Servidor de Bases de Datos** mediante **JDBC**. Por otra parte, el servidor web cuenta con un punto de conexión al cual se conectarán los clientes mediante un **Navegador Web**.

6.1.3 Diagramas de Despliegue

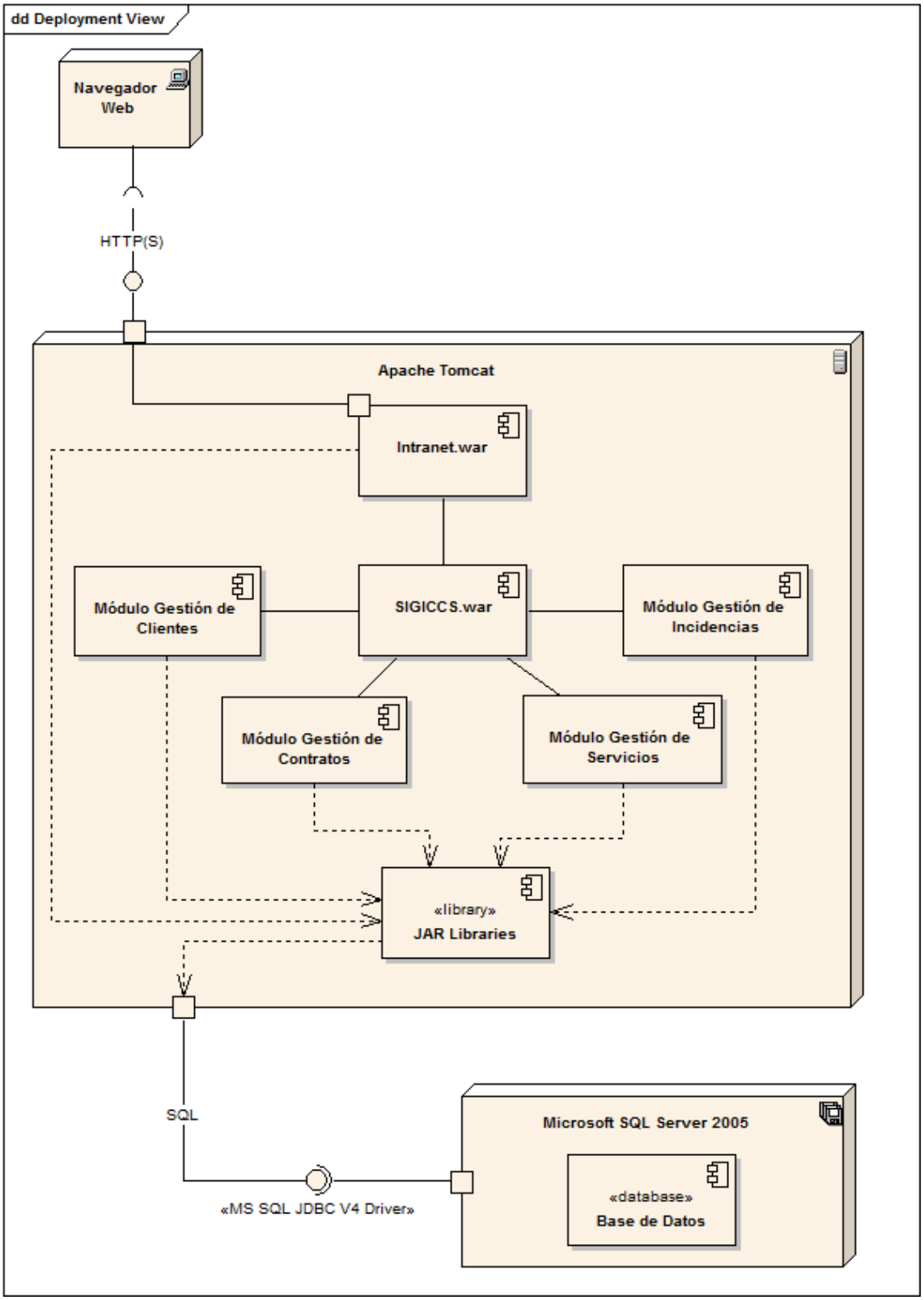


Figura 6.12. Diagrama de Despliegue

El diagrama anterior muestra la estructura de componentes desplegados una vez que el sistema haya sido puesto en funcionamiento. Está compuesto por los siguientes elementos.

6.1.3.1 Apache Tomcat

Actuará como servidor web, alojando el despliegue de las aplicaciones que componen el sistema:

- **Intranet.war**: Aplicación independiente, desarrollada en otro proyecto. Proporciona el soporte para la autenticación de los usuarios.
- **SIGICCS.war**: Aplicación desarrollada en el presente proyecto. Engloba y coordina el conjunto de módulos que la componen (Gestión de Clientes, de Contratos, de Servicios y de Incidencias). Además, en este contenedor se incluyen las librerías específicas de esta parte del sistema.
- **Librerías JAR**: Se trata del conjunto común de librerías que el sistema precisa para trabajar, tales como las correspondientes a Struts2 o las de JDBC.

Es importante recordar que, además de los elementos descritos, el sistema cuenta con otros módulos desarrollados en proyectos independientes, omitidos en el diseño por no resultar relevantes para este sistema.

6.1.3.2 Microsoft SQL Server 2005

Se trata del sistema de gestión de bases de datos utilizado por el sistema para almacenar la información. Se aloja en una máquina independiente de la del servidor web, pero conectadas ambas a la misma red local.

La comunicación con el sistema, mediante lenguaje SQL, se realiza por medio del driver específico para el tipo de base de datos, proporcionado por el fabricante de la misma

6.1.3.3 Navegador Web

Actúa como interfaz de comunicación entre el usuario y la aplicación, comunicando al servidor las peticiones realizadas por el primero y presentando los resultados devueltos por la última.

6.2 Diseño de Clases

6.2.1 Diagrama de Clases

Tal y como se ha estructurado, el proyecto se divide en cuatro módulos diferentes, cada uno de ellos con una funcionalidad determinada. Es por ello que se detallarán los diagramas de clases de cada uno de estos paquetes de forma separada.

Para evitar un exceso de información presentada en esta sección, la descripción detallada de cada una de las clases aquí mostradas puede consultarse en la sección 13.3 del documento.

Para la correcta interpretación de estos diagramas, es necesario haber consultado y comprendido la estructura de paquetes planteada en la sección previa *Arquitectura del Sistema*.

6.2.1.1 Módulo de Gestión de Clientes

6.2.1.1.1 Paquete org.sigiccs.serv.cliente

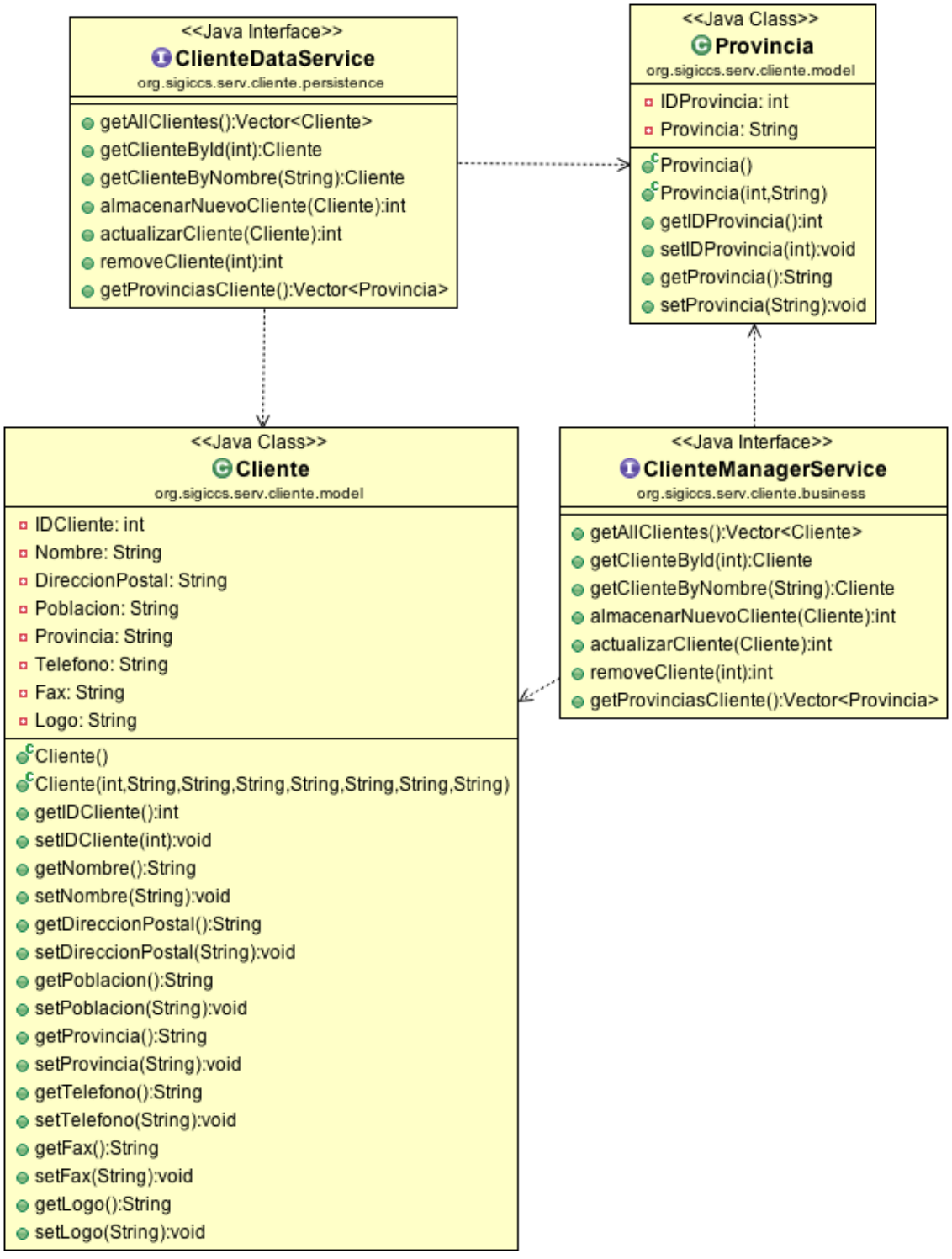


Figura 6.13. Diagrama de Clases - Paquete org.sigiccs.serv.cliente

6.2.1.1.2 Paquete org.sigiccs.impl.cliente

Figura 6.14. Diagrama de Clases - Paquete org.sigiccs.impl.cliente

6.2.1.2 Módulo de Gestión de Contratos

6.2.1.2.1 Paquete org.sigiccs.serv.contrato

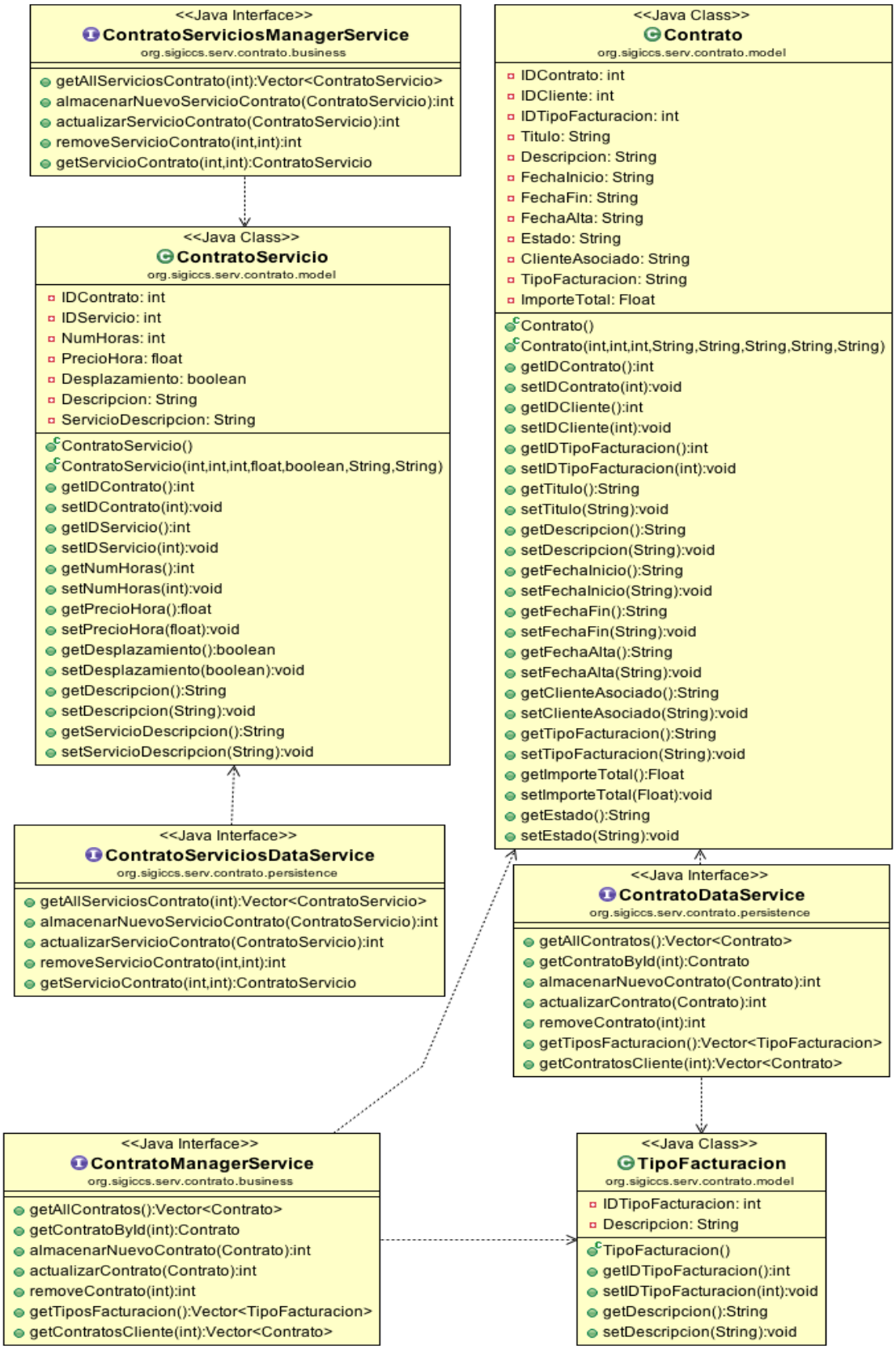
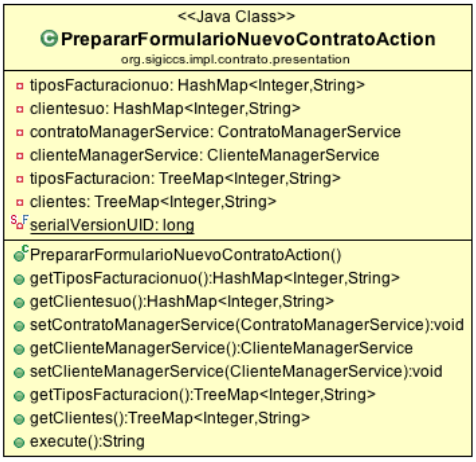
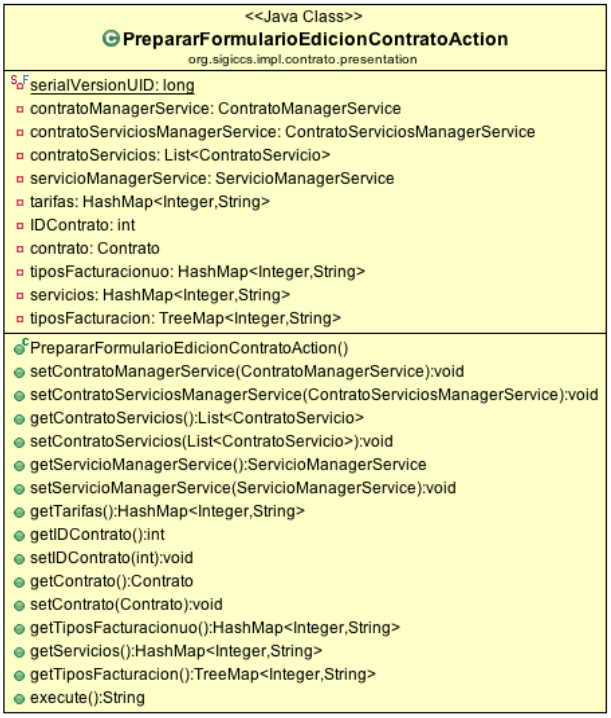
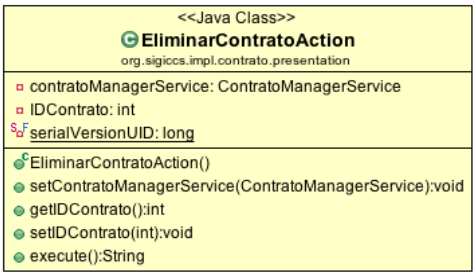


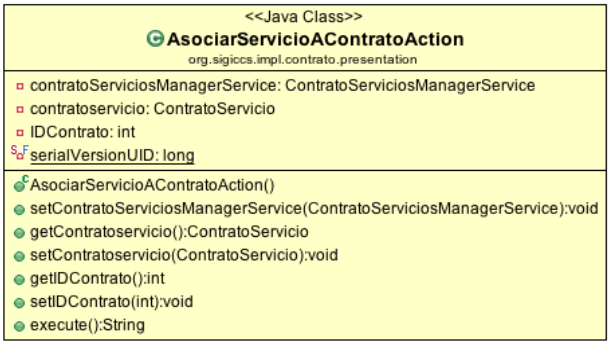
Figura 6.15. Diagrama de Clases - Paquete org.sigiccs.serv.contrato

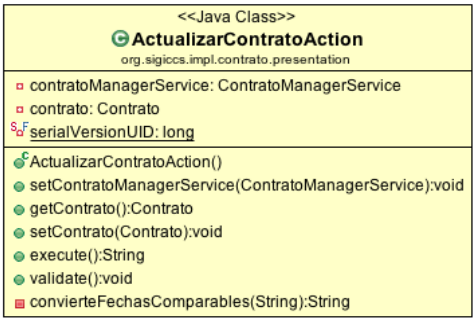
6.2.1.2.2 Paquete org.sigiccs.impl.contrato











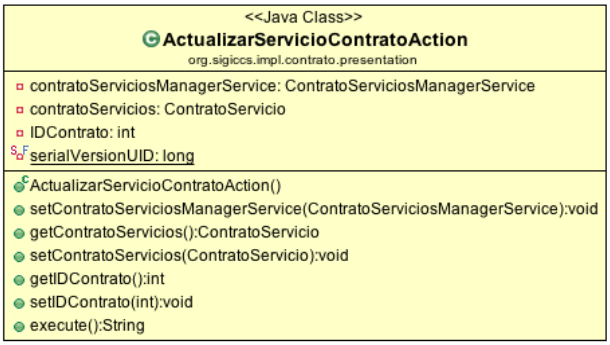


Figura 6.16. Diagrama de Clases - Paquete org.sigiccs.impl.contrato (I)

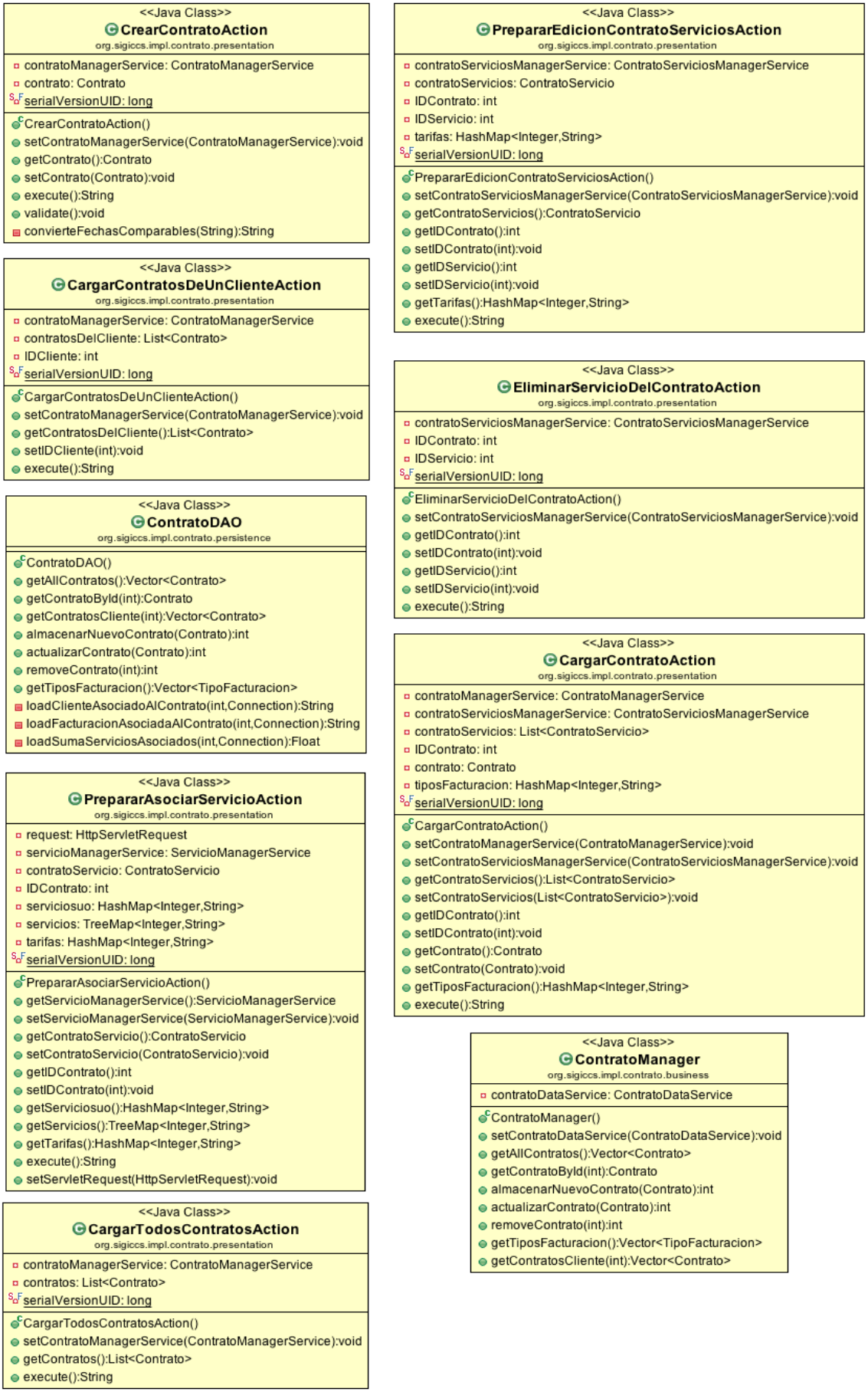

<<Java Class>>
CrearContratoAction
org.sigiccs.impl.contrato.presentation
contratoManagerService: ContratoManagerService
contrato: Contrato
serialVersionUID: long
CrearContratoAction()
setContratoManagerService(ContratoManagerService):void
getContrato():Contrato
setContrato(Contrato):void
execute():String
validate():void
convierteFechasComparables(String):String
<<Java Class>>
CargarContratosDeUnClienteAction
org.sigiccs.impl.contrato.presentation
contratoManagerService: ContratoManagerService
contratosDelCliente: List<Contrato>
IDCliente: int
serialVersionUID: long
CargarContratosDeUnClienteAction()
setContratoManagerService(ContratoManagerService):void
getContratosDelCliente():List<Contrato>
setIDCliente(int):void
execute():String
<<Java Class>>
ContratoDAO
org.sigiccs.impl.contrato.persistence
ContratoDAO()
getAllContratos():Vector<Contrato>
getContratoById(int):Contrato
getContratosCliente(int):Vector<Contrato>
almacenarNuevoContrato(Contrato):int
actualizarContrato(Contrato):int
removeContrato(int):int
getTiposFacturacion():Vector<TipoFacturacion>
loadClienteAsociadoAlContrato(int,Connection):String
loadFacturacionAsociadaAlContrato(int,Connection):String
loadSumaServiciosAsociados(int,Connection):Float
<<Java Class>>
PrepararAsociarServicioAction
org.sigiccs.impl.contrato.presentation
request: HttpServletRequest
servicioManagerService: ServicioManagerService
contratoServicio: ContratoServicio
IDContrato: int
serviciosuo: HashMap<Integer,String>
servicios: TreeMap<Integer,String>
tarifas: HashMap<Integer,String>
serialVersionUID: long
PrepararAsociarServicioAction()
getServicioManagerService():ServicioManagerService
setServicioManagerService(ServicioManagerService):void
getContratoServicio():ContratoServicio
setContratoServicio(ContratoServicio):void
getIDContrato():int
setIDContrato(int):void
getServiciosuo():HashMap<Integer,String>
getServicios():TreeMap<Integer,String>
getTarifas():HashMap<Integer,String>
execute():String
setServletRequest(HttpServletRequest):void
<<Java Class>>
CargarTodosContratosAction
org.sigiccs.impl.contrato.presentation
contratoManagerService: ContratoManagerService
contratos: List<Contrato>
serialVersionUID: long
CargarTodosContratosAction()
setContratoManagerService(ContratoManagerService):void
getContratos():List<Contrato>
execute():String
<<Java Class>>
PrepararEdicionContratoServiciosAction
org.sigiccs.impl.contrato.presentation
contratoServiciosManagerService: ContratoServiciosManagerService
contratoServicios: ContratoServicio
IDContrato: int
IDServicio: int
tarifas: HashMap<Integer,String>
serialVersionUID: long
PrepararEdicionContratoServiciosAction()
setContratoServiciosManagerService(ContratoServiciosManagerService):void
getContratoServicios():ContratoServicio
getIDContrato():int
setIDContrato(int):void
getIDServicio():int
setIDServicio(int):void
getTarifas():HashMap<Integer,String>
execute():String
<<Java Class>>
EliminarServicioDelContratoAction
org.sigiccs.impl.contrato.presentation
contratoServiciosManagerService: ContratoServiciosManagerService
IDContrato: int
IDServicio: int
serialVersionUID: long
EliminarServicioDelContratoAction()
setContratoServiciosManagerService(ContratoServiciosManagerService):void
getIDContrato():int
setIDContrato(int):void
getIDServicio():int
setIDServicio(int):void
execute():String
<<Java Class>>
CargarContratoAction
org.sigiccs.impl.contrato.presentation
contratoManagerService: ContratoManagerService
contratoServiciosManagerService: ContratoServiciosManagerService
contratoServicios: List<ContratoServicio>
IDContrato: int
contrato: Contrato
tiposFacturacion: HashMap<Integer,String>
serialVersionUID: long
CargarContratoAction()
setContratoManagerService(ContratoManagerService):void
setContratoServiciosManagerService(ContratoServiciosManagerService):void
getContratoServicios():List<ContratoServicio>
setContratoServicios(List<ContratoServicio>):void
getIDContrato():int
setIDContrato(int):void
getContrato():Contrato
setContrato(Contrato):void
getTiposFacturacion():HashMap<Integer,String>
execute():String
<<Java Class>>
ContratoManager
org.sigiccs.impl.contrato.business
contratoDataService: ContratoDataService
ContratoManager()
setContratoDataService(ContratoDataService):void
getAllContratos():Vector<Contrato>
getContratoById(int):Contrato
almacenarNuevoContrato(Contrato):int
actualizarContrato(Contrato):int
removeContrato(int):int
getTiposFacturacion():Vector<TipoFacturacion>
getContratosCliente(int):Vector<Contrato>

Figura 6.17. Diagrama de Clases - Paquete org.sigiccs.impl.contrato (II)

6.2.1.3 Módulo de Gestión de Servicios

6.2.1.3.1 Paquete org.sigiccs.serv.servicio

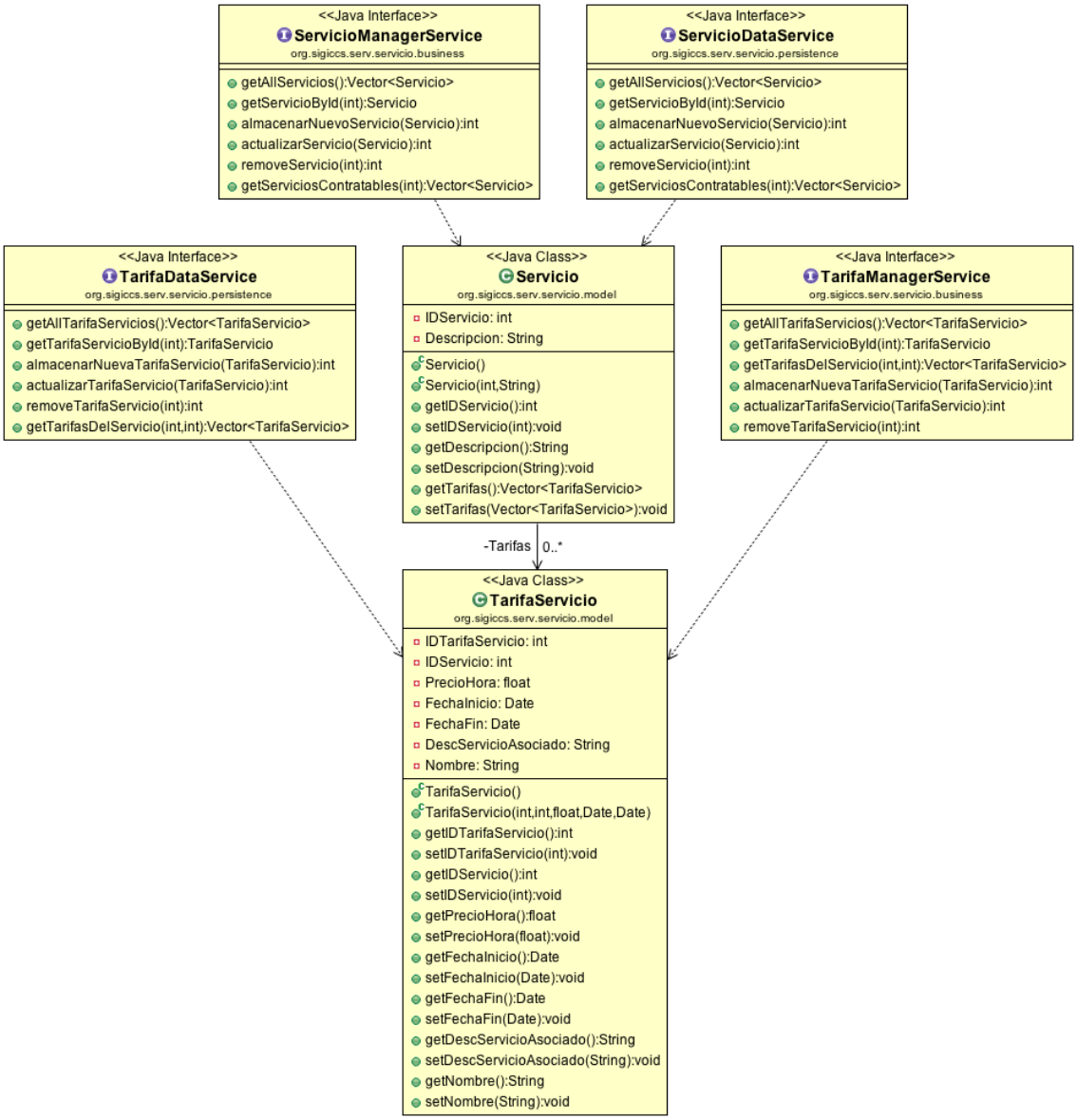


Figura 6.18. Diagrama de Clases - Paquete org.sigiccs.serv.servicio

6.2.1.3.2 Paquete org.sigiccs.impl.servicio

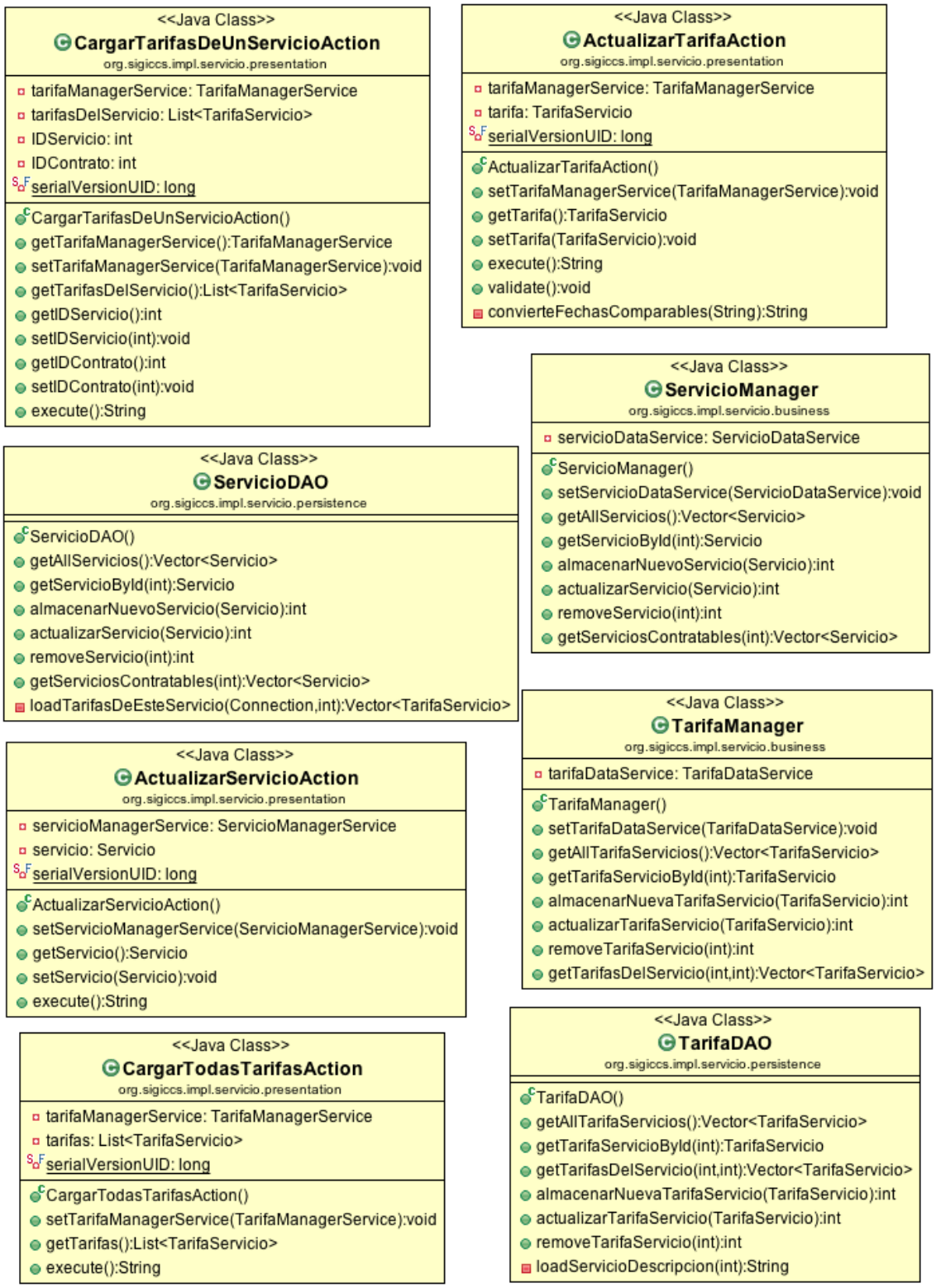


Figura 6.19. Diagrama de Clases - Paquete org.sigiccs.impl.servicio (I)

Figura 6.20. Diagrama de Clases - Paquete org.sigiccs.impl.servicio (II)

6.2.1.4 Módulo de Gestión de Incidencias

6.2.1.4.1 Paquete org.sigiccs.serv.incidencia

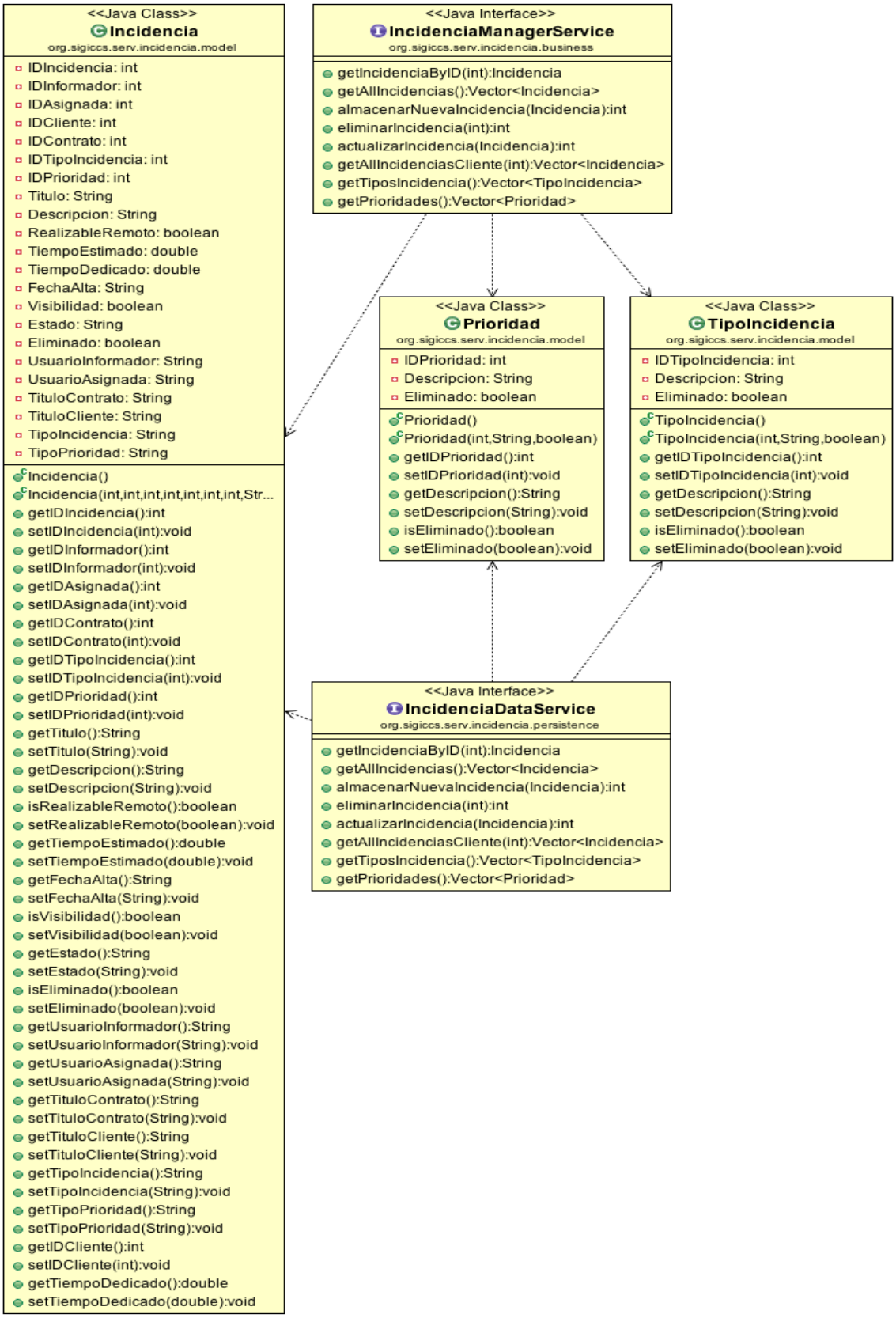


Figura 6.21. Diagrama de Clases - Paquete org.sigiccs.serv.incidencia (I)

Figura 6.22. Diagrama de Clases - Paquete org.sigiccs.serv.incidencia (II)

Figura 6.23. Diagrama de Clases - Paquete org.sigiccs.serv.incidencia (III)

6.2.1.4.2 Paquete org.sigiccs.impl.incidencia

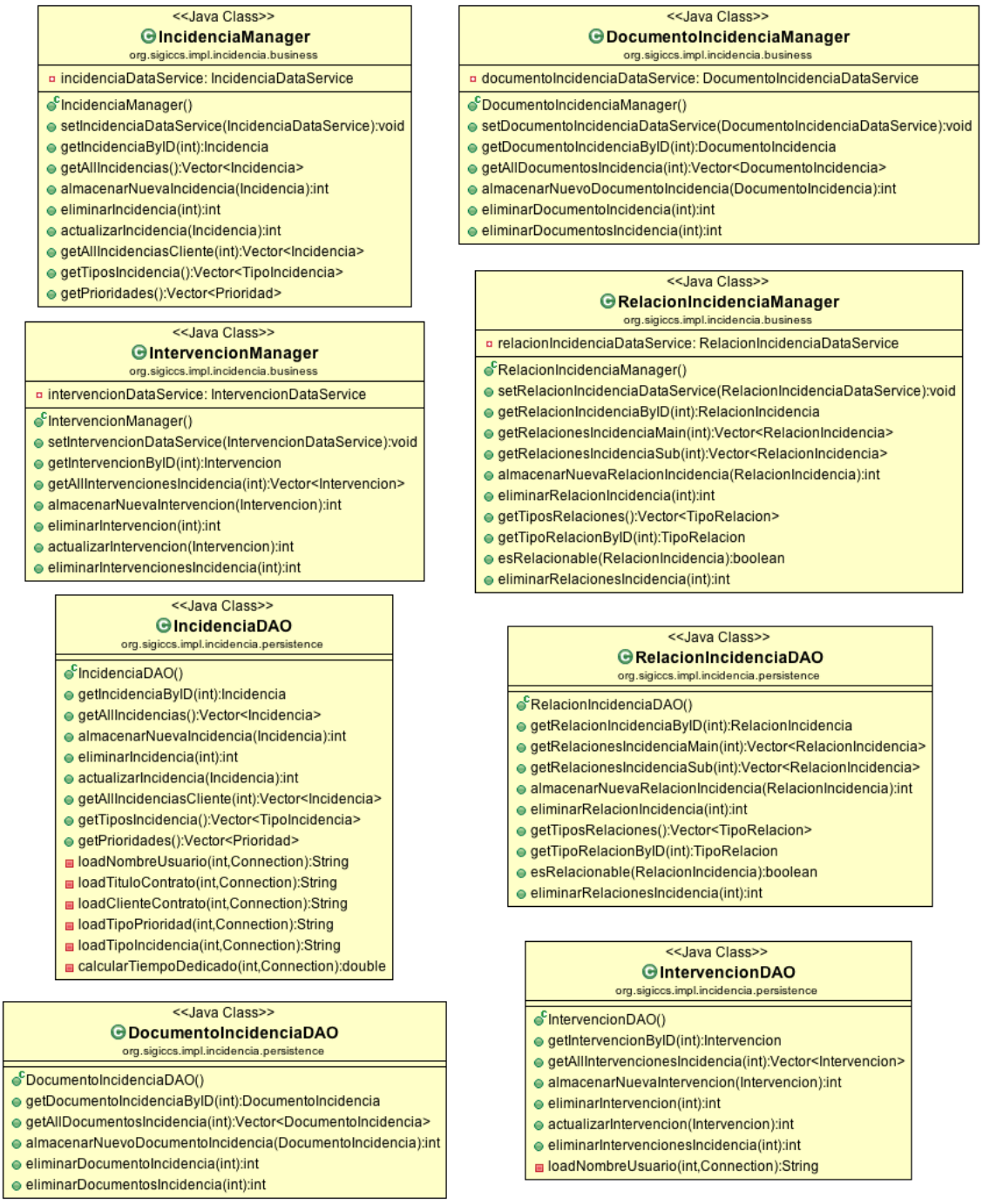


Figura 6.24. Diagrama de Clases - Paquete org.sigiccs.impl.incidencia (I)

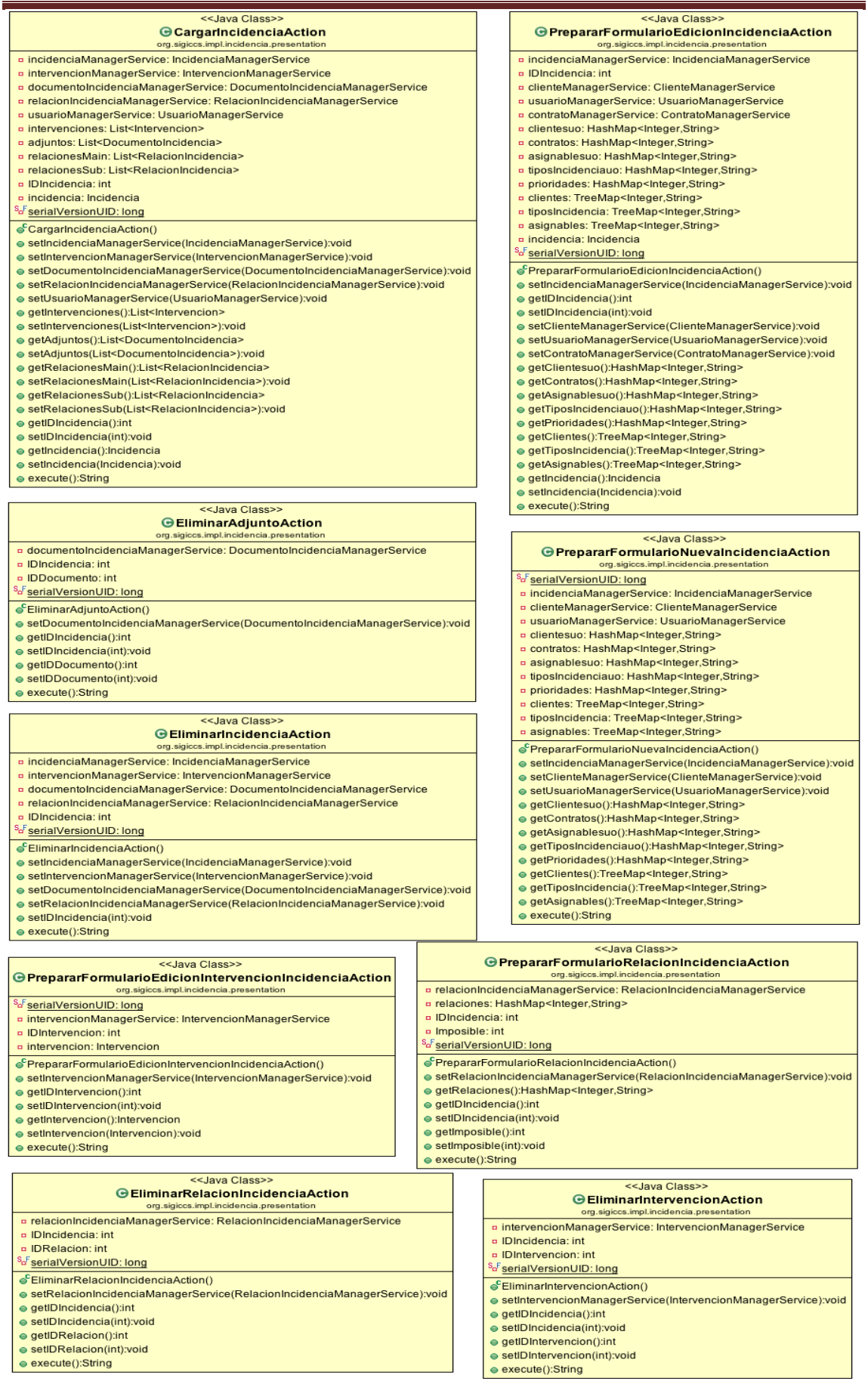

<<Java Class>>
CargarIncidenciaAction
org.sigiccs.impl.incidencia.presentation
incidenciaManagerService: IncidenciaManagerService
intervencionManagerService: IntervencionManagerService
documentoIncidenciaManagerService: DocumentoIncidenciaManagerService
relacionIncidenciaManagerService: RelacionIncidenciaManagerService
usuarioManagerService: UsuarioManagerService
intervenciones: List<Intervencion>
adjuntos: List<DocumentoIncidencia>
relacionesMain: List<RelacionIncidencia>
relacionesSub: List<RelacionIncidencia>
IDIncidencia: int
incidencia: Incidencia
serialVersionUID: long
CargarIncidenciaAction()
setIncidenciaManagerService(IncidenciaManagerService):void
setIntervencionManagerService(IntervencionManagerService):void
setDocumentoIncidenciaManagerService(DocumentoIncidenciaManagerService):void
setRelacionIncidenciaManagerService(RelacionIncidenciaManagerService):void
setUsuarioManagerService(UsuarioManagerService):void
getIntervenciones():List<Intervencion>
setIntervenciones(List<Intervencion>):void
getAdjuntos():List<DocumentoIncidencia>
setAdjuntos(List<DocumentoIncidencia>):void
getRelacionesMain():List<RelacionIncidencia>
setRelacionesMain(List<RelacionIncidencia>):void
getRelacionesSub():List<RelacionIncidencia>
setRelacionesSub(List<RelacionIncidencia>):void
getIDIncidencia():int
setIDIncidencia(int):void
getIncidencia():Incidencia
setIncidencia(Incidencia):void
execute():String
<<Java Class>>
PrepararFormularioEdicionIncidenciaAction
org.sigiccs.impl.incidencia.presentation
incidenciaManagerService: IncidenciaManagerService
IDIncidencia: int
clienteManagerService: ClienteManagerService
usuarioManagerService: UsuarioManagerService
contratoManagerService: ContratoManagerService
clientesuo: HashMap<Integer,String>
contratos: HashMap<Integer,String>
asignablesuo: HashMap<Integer,String>
tiposIncidenciauo: HashMap<Integer,String>
prioridades: HashMap<Integer,String>
clientes: TreeMap<Integer,String>
tiposIncidencia: TreeMap<Integer,String>
asignables: TreeMap<Integer,String>
incidencia: Incidencia
serialVersionUID: long
PrepararFormularioEdicionIncidenciaAction()
setIncidenciaManagerService(IncidenciaManagerService):void
getIDIncidencia():int
setIDIncidencia(int):void
setClienteManagerService(ClienteManagerService):void
setUsuarioManagerService(UsuarioManagerService):void
setContratoManagerService(ContratoManagerService):void
getClientesuo():HashMap<Integer,String>
getContratos():HashMap<Integer,String>
getAsignablesuo():HashMap<Integer,String>
getTiposIncidenciauo():HashMap<Integer,String>
getPrioridades():HashMap<Integer,String>
getClientes():TreeMap<Integer,String>
getTiposIncidencia():TreeMap<Integer,String>
getAsignables():TreeMap<Integer,String>
getIncidencia():Incidencia
setIncidencia(Incidencia):void
execute():String
<<Java Class>>
EliminarAdjuntoAction
org.sigiccs.impl.incidencia.presentation
documentoIncidenciaManagerService: DocumentoIncidenciaManagerService
IDIncidencia: int
IDDocumento: int
serialVersionUID: long
EliminarAdjuntoAction()
setDocumentoIncidenciaManagerService(DocumentoIncidenciaManagerService):void
getIDIncidencia():int
setIDIncidencia(int):void
getIDDocumento():int
setIDDocumento(int):void
execute():String
<<Java Class>>
PrepararFormularioNuevaIncidenciaAction
org.sigiccs.impl.incidencia.presentation
serialVersionUID: long
incidenciaManagerService: IncidenciaManagerService
clienteManagerService: ClienteManagerService
usuarioManagerService: UsuarioManagerService
clientesuo: HashMap<Integer,String>
contratos: HashMap<Integer,String>
asignablesuo: HashMap<Integer,String>
tiposIncidenciauo: HashMap<Integer,String>
prioridades: HashMap<Integer,String>
clientes: TreeMap<Integer,String>
tiposIncidencia: TreeMap<Integer,String>
asignables: TreeMap<Integer,String>
PrepararFormularioNuevaIncidenciaAction()
setIncidenciaManagerService(IncidenciaManagerService):void
setClienteManagerService(ClienteManagerService):void
setUsuarioManagerService(UsuarioManagerService):void
getClientesuo():HashMap<Integer,String>
getContratos():HashMap<Integer,String>
getAsignablesuo():HashMap<Integer,String>
getTiposIncidenciauo():HashMap<Integer,String>
getPrioridades():HashMap<Integer,String>
getClientes():TreeMap<Integer,String>
getTiposIncidencia():TreeMap<Integer,String>
getAsignables():TreeMap<Integer,String>
execute():String
<<Java Class>>
EliminarIncidenciaAction
org.sigiccs.impl.incidencia.presentation
incidenciaManagerService: IncidenciaManagerService
intervencionManagerService: IntervencionManagerService
documentoIncidenciaManagerService: DocumentoIncidenciaManagerService
relacionIncidenciaManagerService: RelacionIncidenciaManagerService
IDIncidencia: int
serialVersionUID: long
EliminarIncidenciaAction()
setIncidenciaManagerService(IncidenciaManagerService):void
setIntervencionManagerService(IntervencionManagerService):void
setDocumentoIncidenciaManagerService(DocumentoIncidenciaManagerService):void
setRelacionIncidenciaManagerService(RelacionIncidenciaManagerService):void
setIDIncidencia(int):void
execute():String
<<Java Class>>
PrepararFormularioEdicionIntervencionIncidenciaAction
org.sigiccs.impl.incidencia.presentation
serialVersionUID: long
intervencionManagerService: IntervencionManagerService
IDIntervencion: int
intervencion: Intervencion
PrepararFormularioEdicionIntervencionIncidenciaAction()
setIntervencionManagerService(IntervencionManagerService):void
getIDIntervencion():int
setIDIntervencion(int):void
getIntervencion():Intervencion
setIntervencion(Intervencion):void
execute():String
<<Java Class>>
PrepararFormularioRelacionIncidenciaAction
org.sigiccs.impl.incidencia.presentation
relacionIncidenciaManagerService: RelacionIncidenciaManagerService
relaciones: HashMap<Integer,String>
IDIncidencia: int
Imposible: int
serialVersionUID: long
PrepararFormularioRelacionIncidenciaAction()
setRelacionIncidenciaManagerService(RelacionIncidenciaManagerService):void
getRelaciones():HashMap<Integer,String>
getIDIncidencia():int
setIDIncidencia(int):void
getImposible():int
setImposible(int):void
execute():String
<<Java Class>>
EliminarRelacionIncidenciaAction
org.sigiccs.impl.incidencia.presentation
relacionIncidenciaManagerService: RelacionIncidenciaManagerService
IDIncidencia: int
IDRelacion: int
serialVersionUID: long
EliminarRelacionIncidenciaAction()
setRelacionIncidenciaManagerService(RelacionIncidenciaManagerService):void
getIDIncidencia():int
setIDIncidencia(int):void
getIDRelacion():int
setIDRelacion(int):void
execute():String
<<Java Class>>
EliminarIntervencionAction
org.sigiccs.impl.incidencia.presentation
intervencionManagerService: IntervencionManagerService
IDIncidencia: int
IDIntervencion: int
serialVersionUID: long
EliminarIntervencionAction()
setIntervencionManagerService(IntervencionManagerService):void
getIDIncidencia():int
setIDIncidencia(int):void
getIDIntervencion():int
setIDIntervencion(int):void
execute():String

Figura 6.25. Diagrama de Clases - Paquete org.sigiccs.impl.incidencia (III)

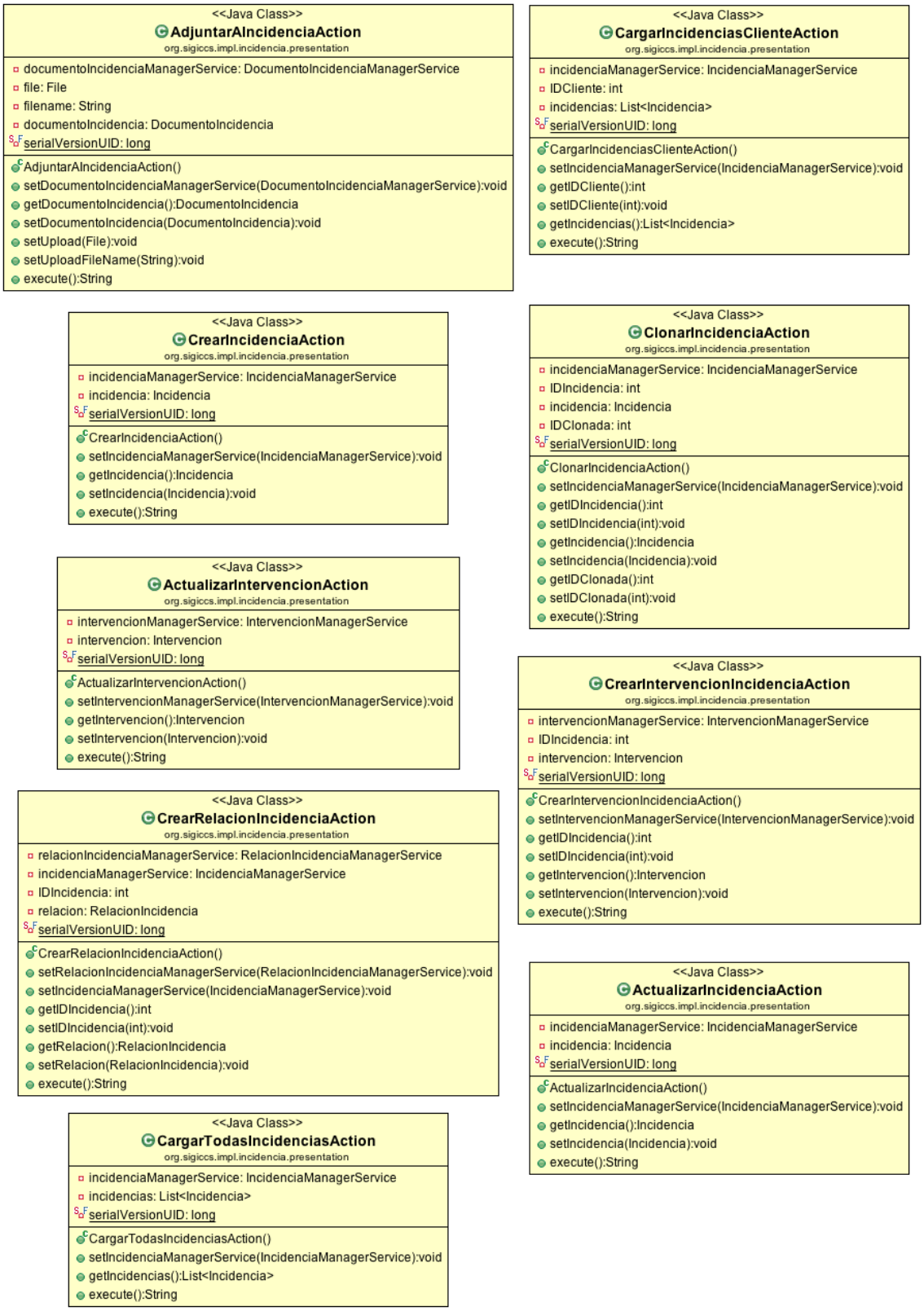


Figura 6.26. Diagrama de Clases - Paquete org.sigiccs.impl.incidencia (II)

6.2.1.5 Paquete org.sigiccs.util

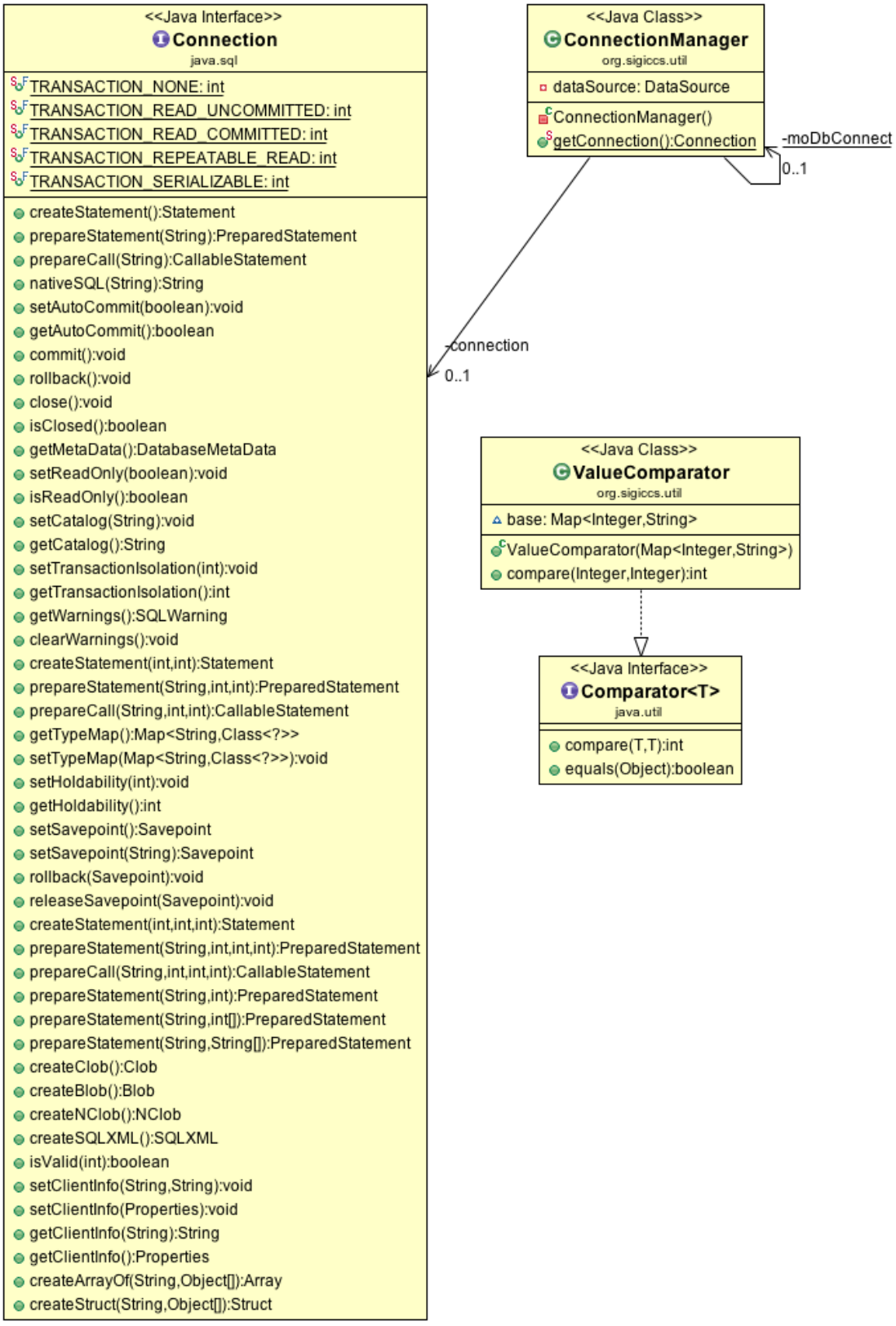


Figura 6.27. Diagrama de Clases - Paquete org.sigiccs.util

6.3 Diagramas de Secuencia

En esta sección se detallan y evolucionan los diagramas de robustez presentados en el análisis del sistema. Sin embargo, debido a que el número de casos de uso es elevado (51) y con el objetivo de no extender excesivamente el contenido de la sección, y de la documentación en general, se sintetizarán las operaciones similares, de modo que se mostrará un diagrama de secuencia representativo de cada operación para la que exista una con idéntico propósito en los demás módulos.

6.3.1 Casos de Uso Alta, Baja y Modificación

Los siguientes diagramas de secuencia muestran el comportamiento del sistema en lo relativo a las operaciones básicas de alta, baja y modificación de clientes, contratos, servicios e incidencias.

6.3.1.1 Caso de Uso Crear Elemento

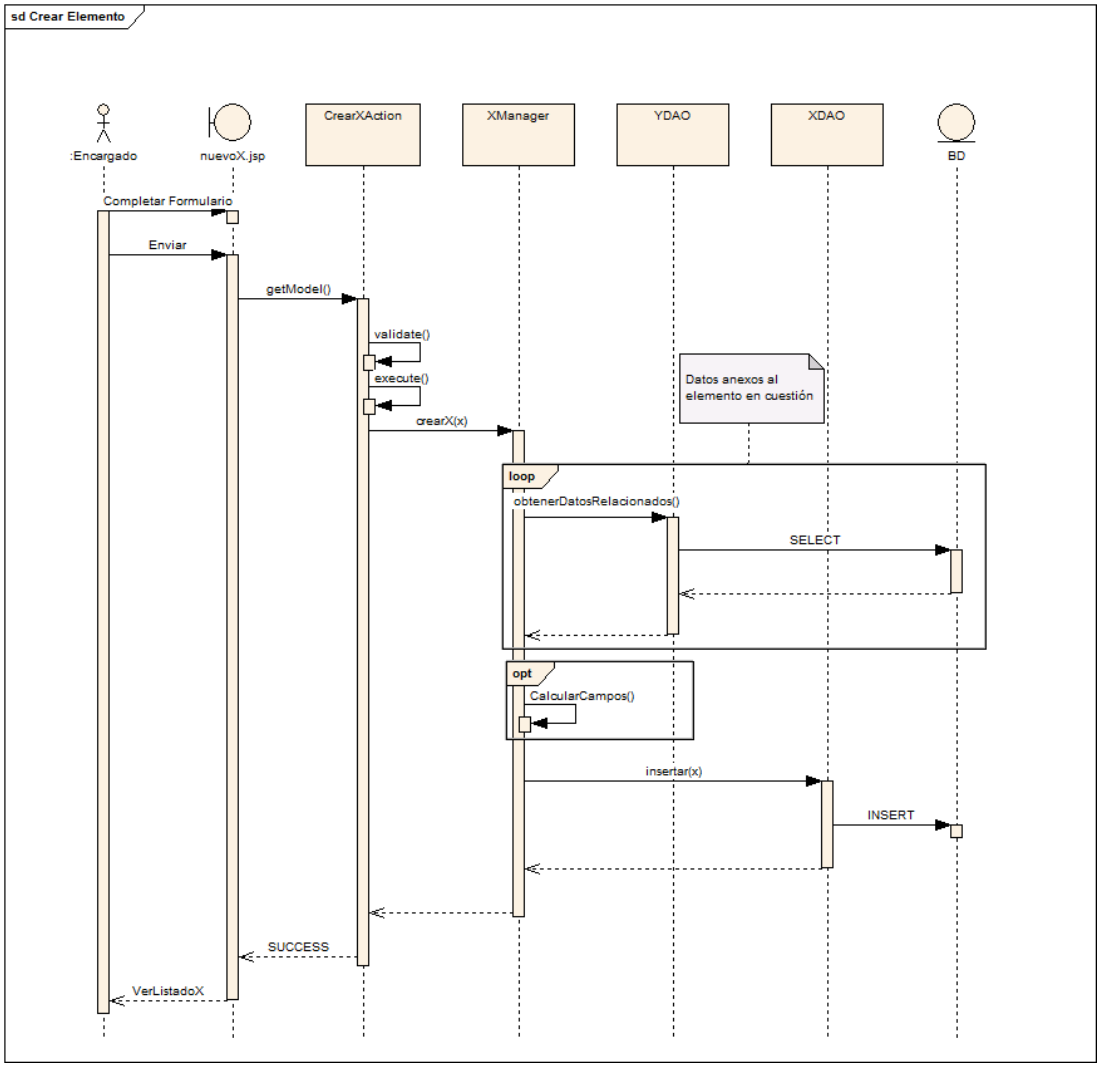


Figura 6.28. Diagrama de Secuencia - Crear elemento

6.3.1.2 Caso de Uso Modificar Elemento

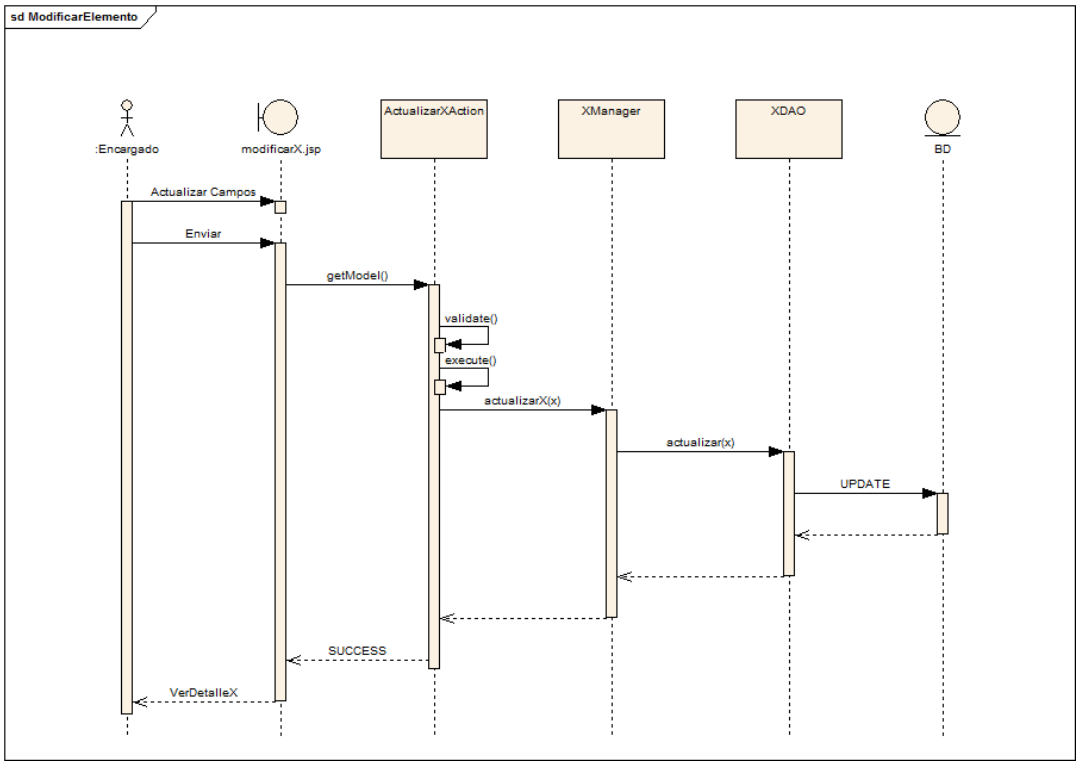


Figura 6.29. Diagrama de Secuencia - Modificar elemento

6.3.1.3 Caso de Uso Eliminar Elemento

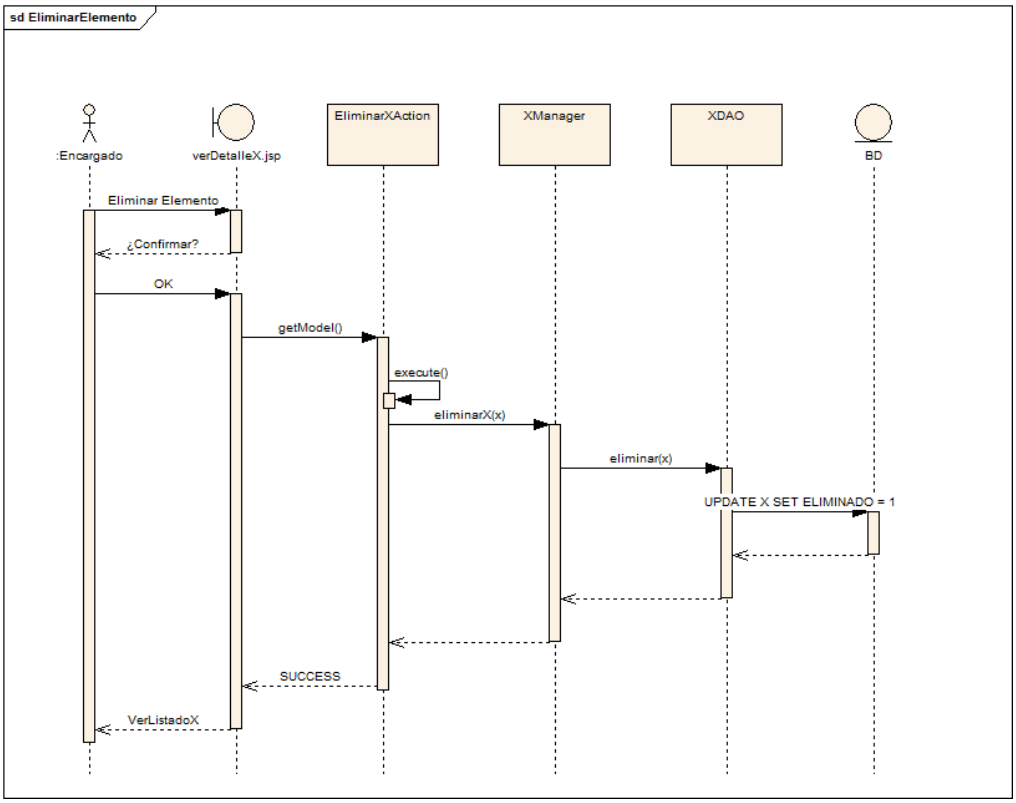


Figura 6.30. Diagrama de Secuencia - Eliminar elemento

6.3.2 Casos de Uso Consulta de Listado y Detalle

Los siguientes diagramas de secuencia muestran el comportamiento del sistema en lo relativo a las operaciones de consulta de listado y detalle de clientes, contratos, servicios e incidencias.

6.3.2.1 Caso de Uso Consultar Listado de Elementos

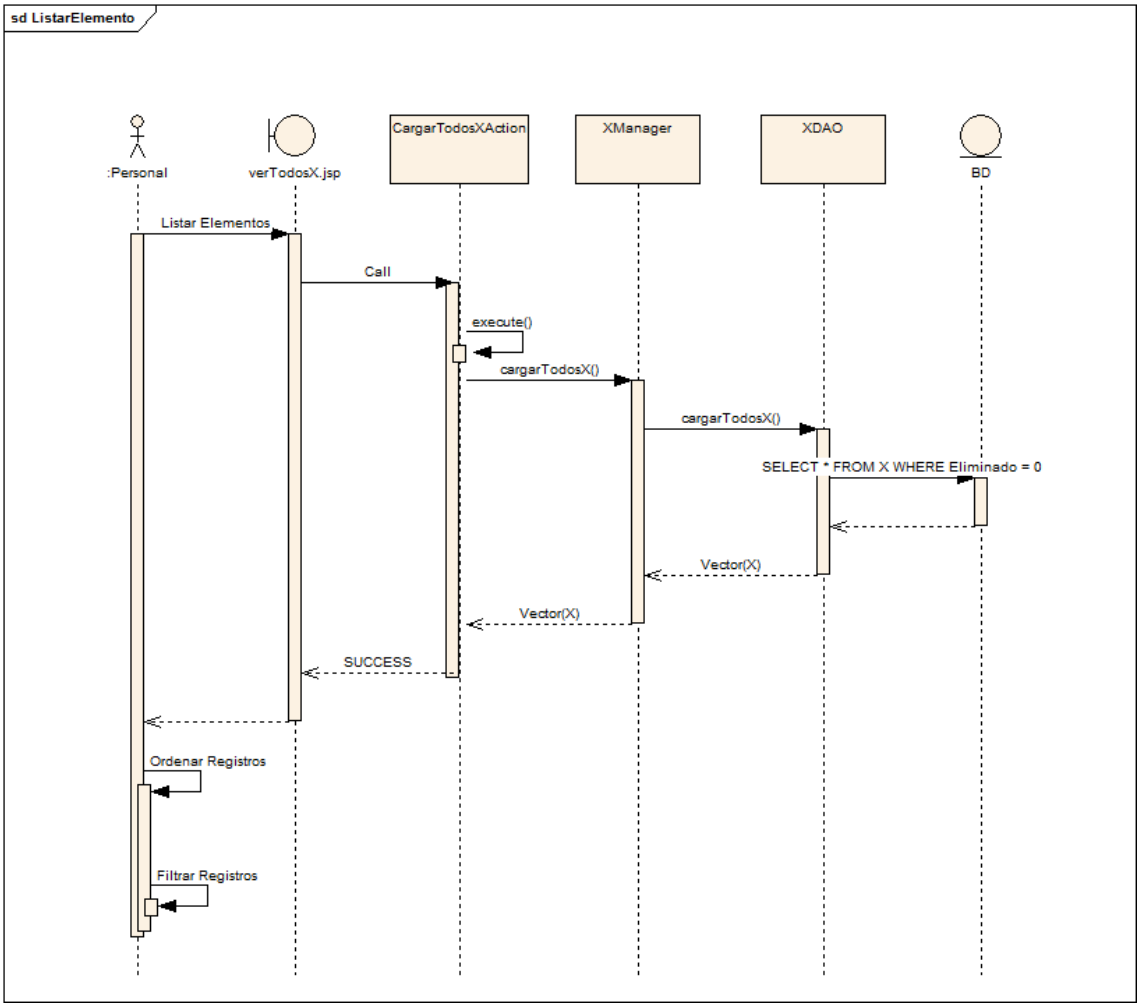


Figura 6.31. Diagrama de Secuencia - Consultar listado de elementos

6.3.2.2 Caso de Uso Consultar Detalle de Elemento

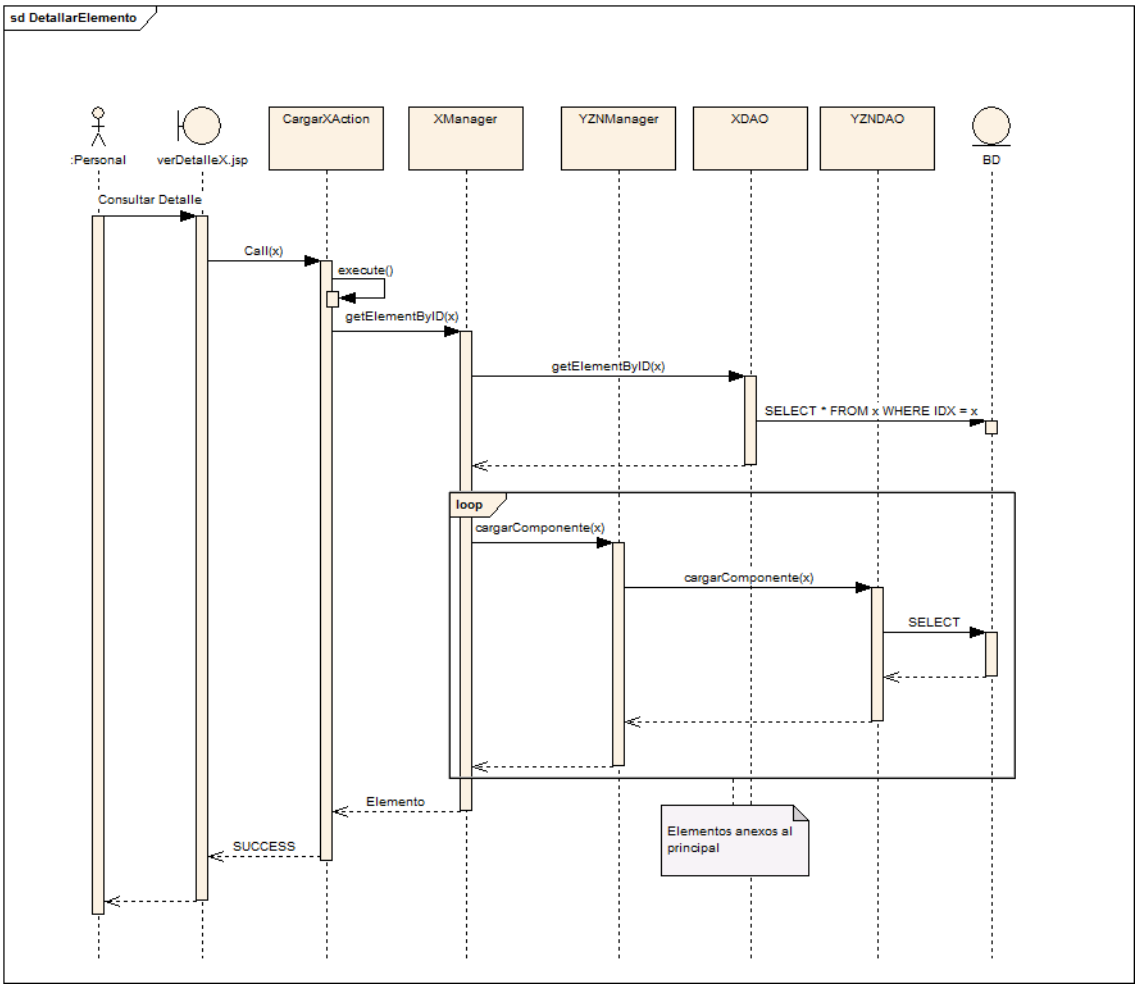


Figura 6.32. Diagrama de Secuencia - Consultar detalle de elemento

6.3.3 Casos de Uso Gestión de Componentes de Elementos

Los siguientes diagramas de secuencia muestran el comportamiento del sistema en lo relativo a las operaciones de gestión de componentes anexos a los elementos cliente, contrato, servicio e incidencia.

6.3.3.1 Caso de Uso Incluir Componente en Elemento

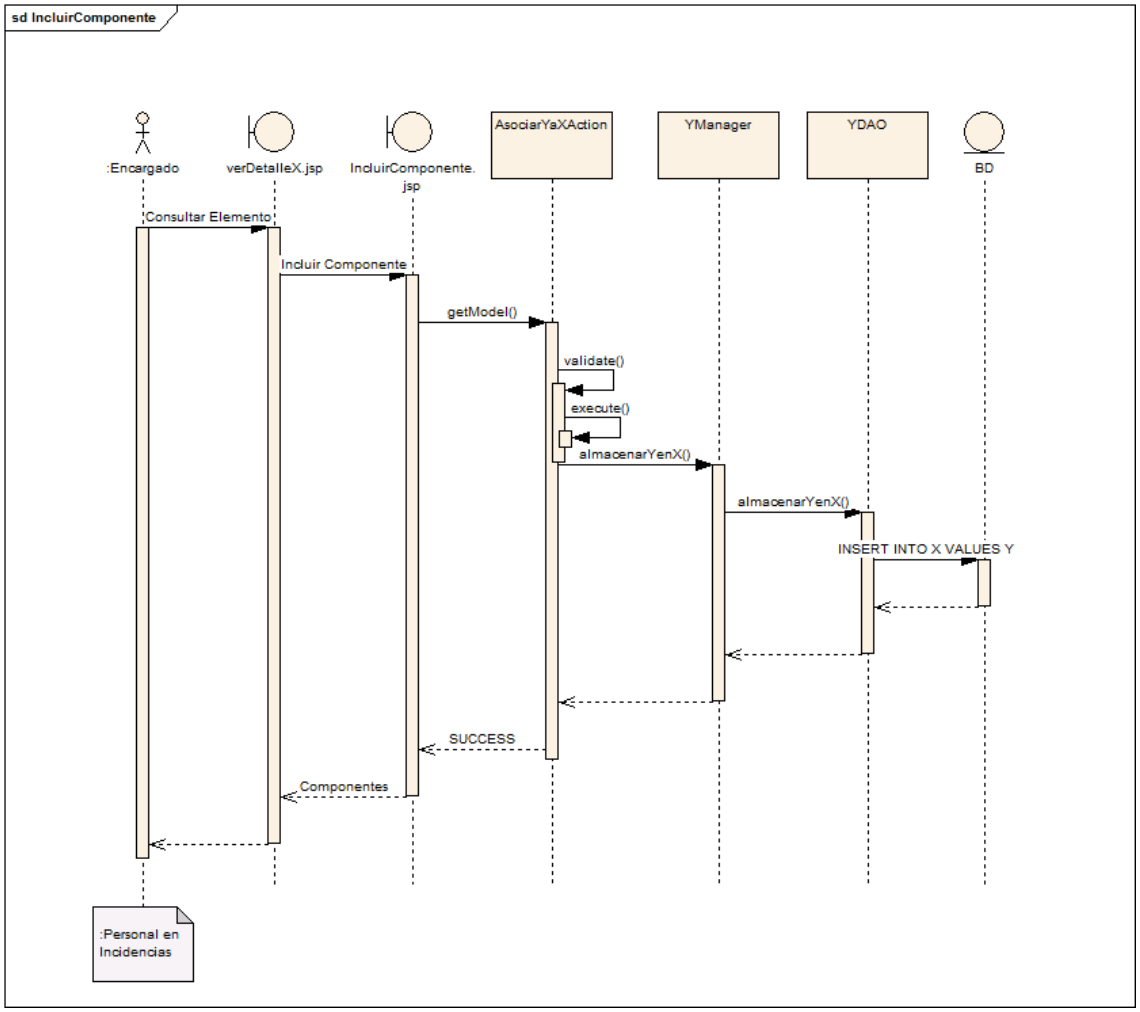


Figura 6.33. Diagrama de Secuencia - Incluir componente en el elemento

6.3.3.2 Caso de Uso Modificar Componente Asociado a Elemento

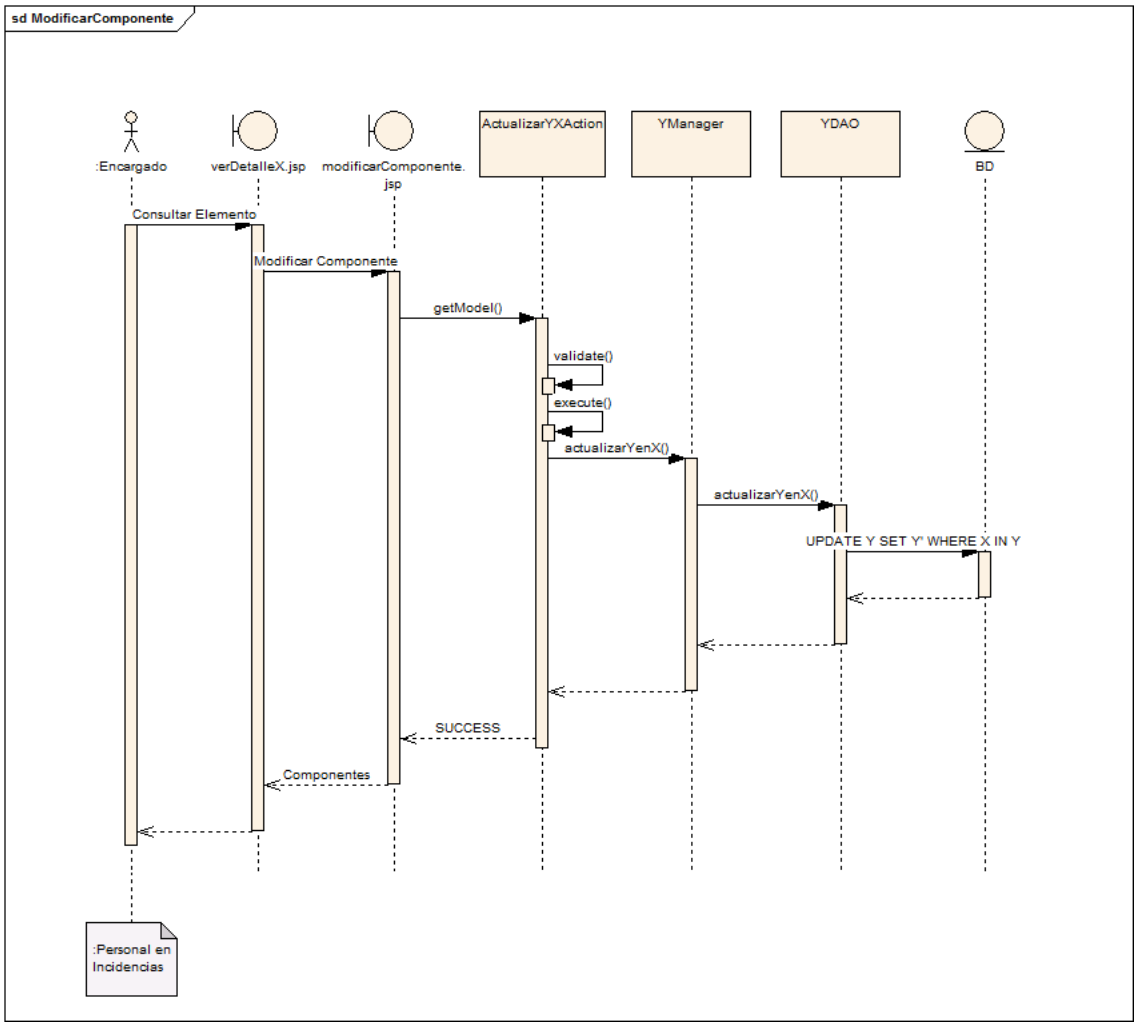


Figura 6.34. Diagrama de Secuencia - Modificar componente asociado al elemento

6.3.3.3 Caso de Uso Eliminar Componente Asociado a Elemento

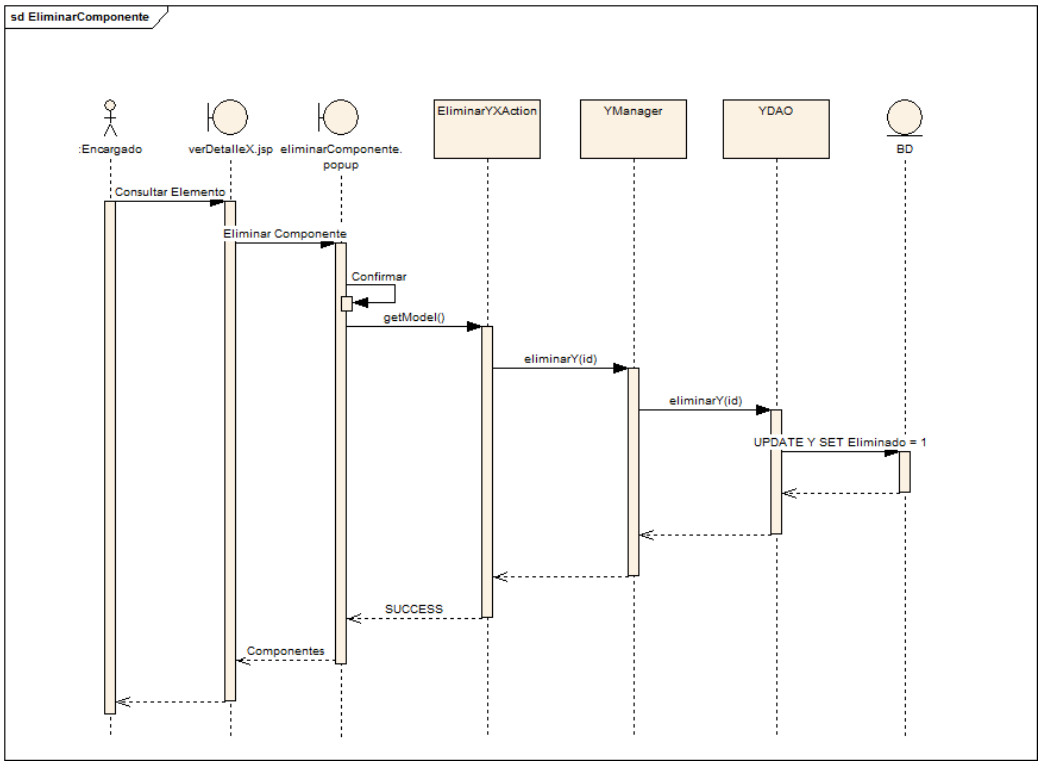


Figura 6.35. Diagrama de Secuencia - Eliminar componente asociado al elemento

6.3.3.4 Caso de Uso Listar Componentes de Elemento

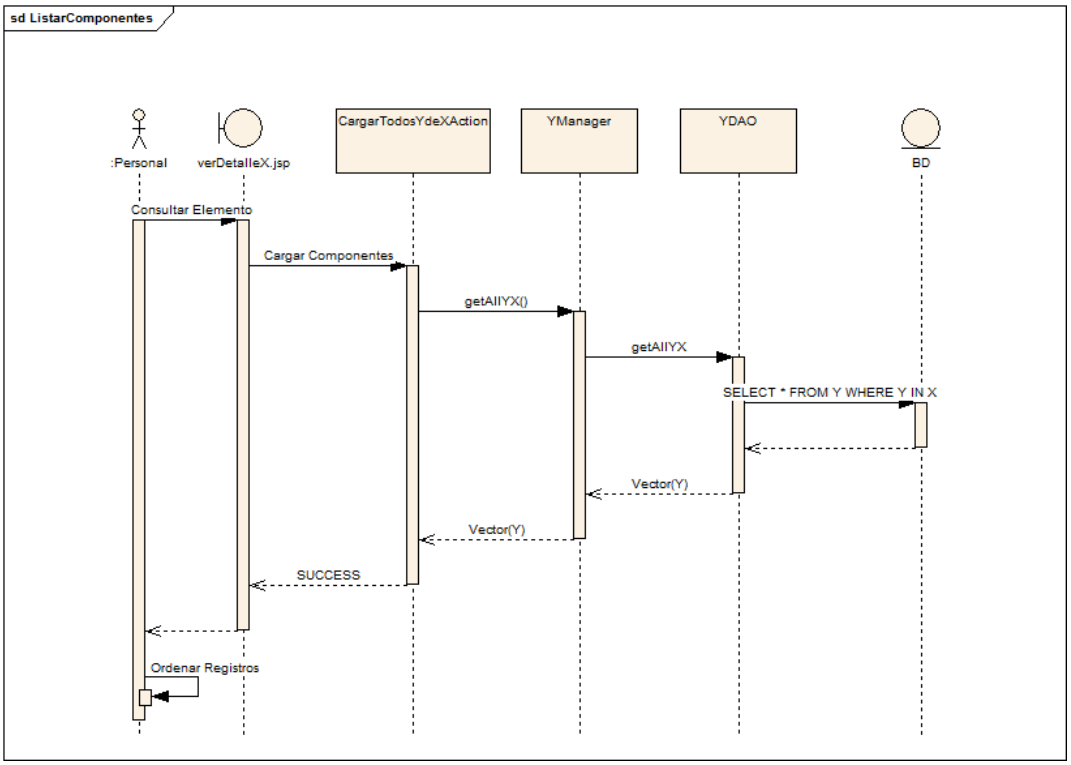


Figura 6.36. Diagrama de Secuencia - Listar componentes del elemento

6.3.4 Casos de Uso Particulares

6.3.4.1 Caso de Uso Clonar Incidencia

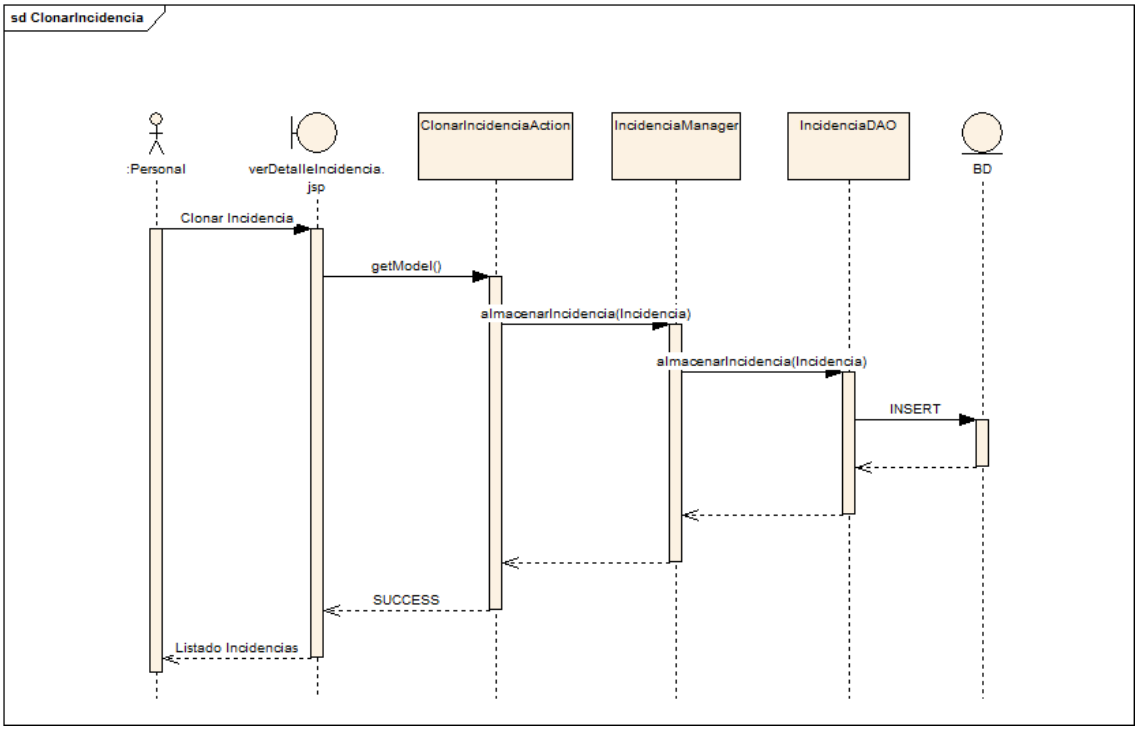


Figura 6.37. Diagrama de Secuencia - Clonar incidencia

6.3.4.2 Caso de Uso Generar Informe de Incidencia

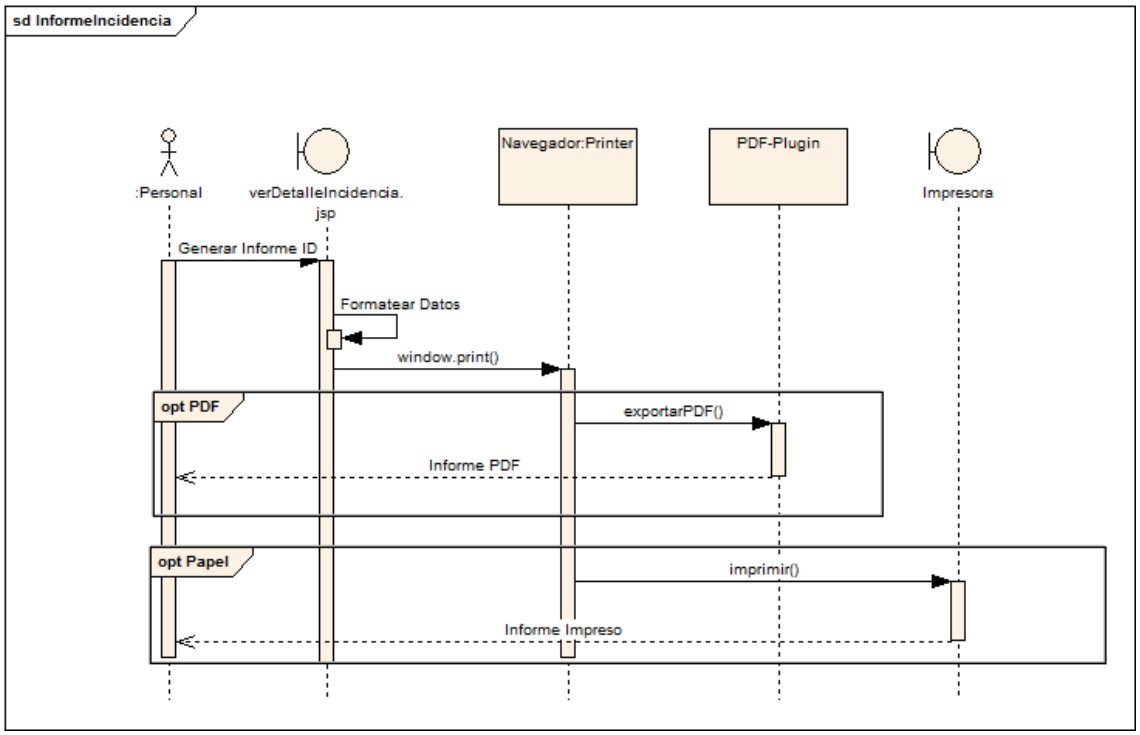


Figura 6.38. Diagrama de Secuencia - Generar informe de incidencia

6.4 Diseño de la Base de Datos

La parte de persistencia de datos del sistema está compuesta por una única base de datos en la que se almacena toda la información necesaria para el funcionamiento de la aplicación.

6.4.1 Descripción del SGBD Usado

El Sistema de Gestión de Bases de Datos empleado es Microsoft SQL Server 2005. Se trata de un requisito impuesto por el cliente, puesto que todas sus bases de datos están construidas sobre este sistema y el empleo de un producto diferente supondría un coste extra para el departamento de sistemas de la empresa.

Este SGBD contempla soporte a transacciones y procedimientos almacenados. Dispone además de un entorno gráfico para su administración y la posibilidad de trabajar en modo cliente-servidor. Gracias al cliente nativo de SQL incorporado es posible ejecutar las sentencias en dicho lenguaje estándar.

6.4.2 Integración del SGBD en Nuestro Sistema

La biblioteca Java DataBase Connectivity (JDBC) empleada en el proyecto dispone de los mecanismos y controladores necesarios para conectarse y trabajar con SQL Server 2005. Para ello, es necesario proporcionar su ubicación de red, identificador de la base de datos y un usuario y contraseña con permisos sobre la misma.

Las conexiones se realizan a través de un *pool* de conexiones, encargado de mantener una serie de conexiones en espera que pueden ser reutilizadas, agilizando así los mecanismos de conexión con la base de datos y evitando sobrecarga en la red, lo cual resulta adecuado en aplicaciones web interactivas.

6.4.3 Diagrama E-R

Por último, se presenta el diagrama entidad-relación que modela el esquema de la base de datos.

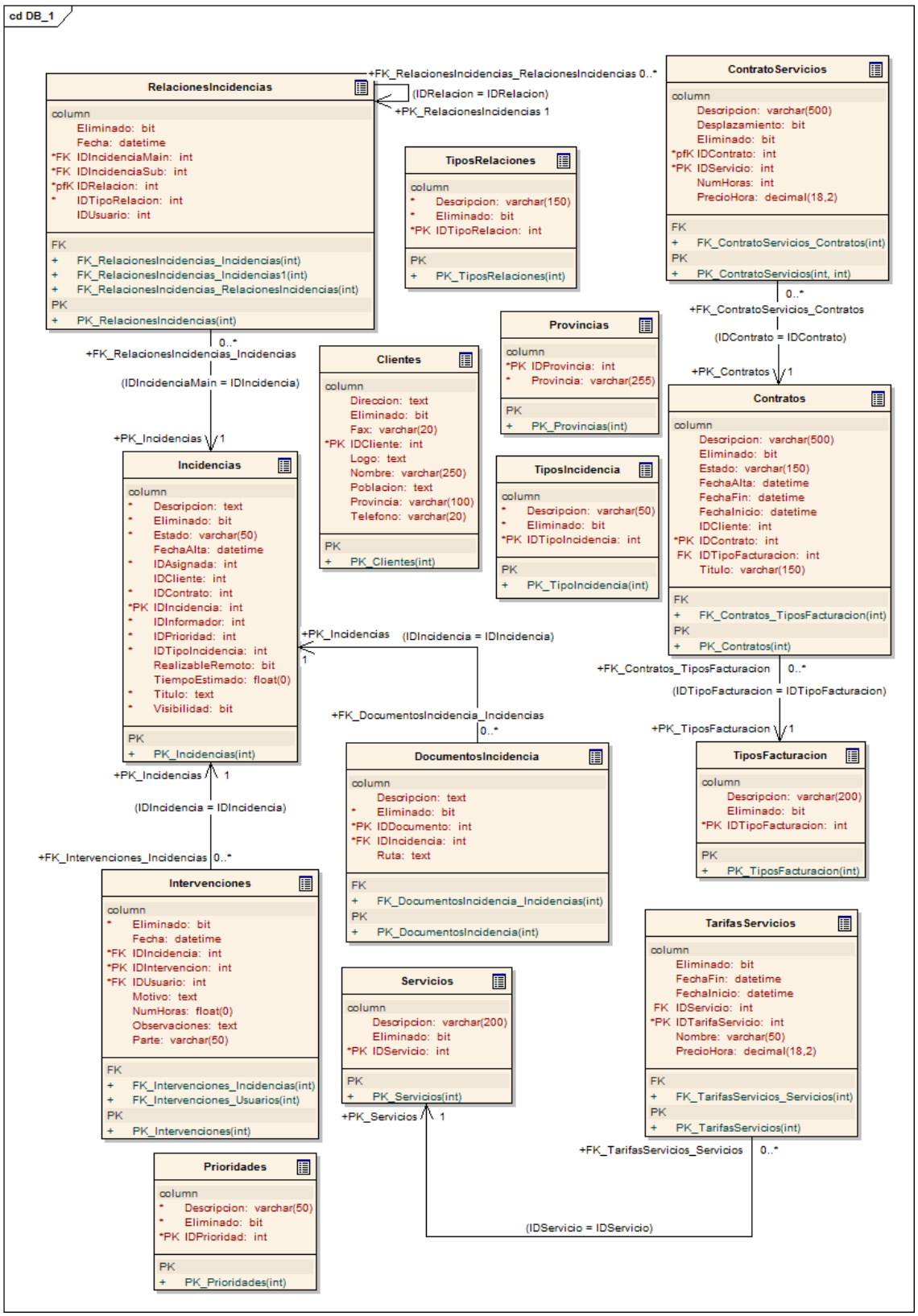


Figura 6.39. Diagrama Entidad-Relación (I)

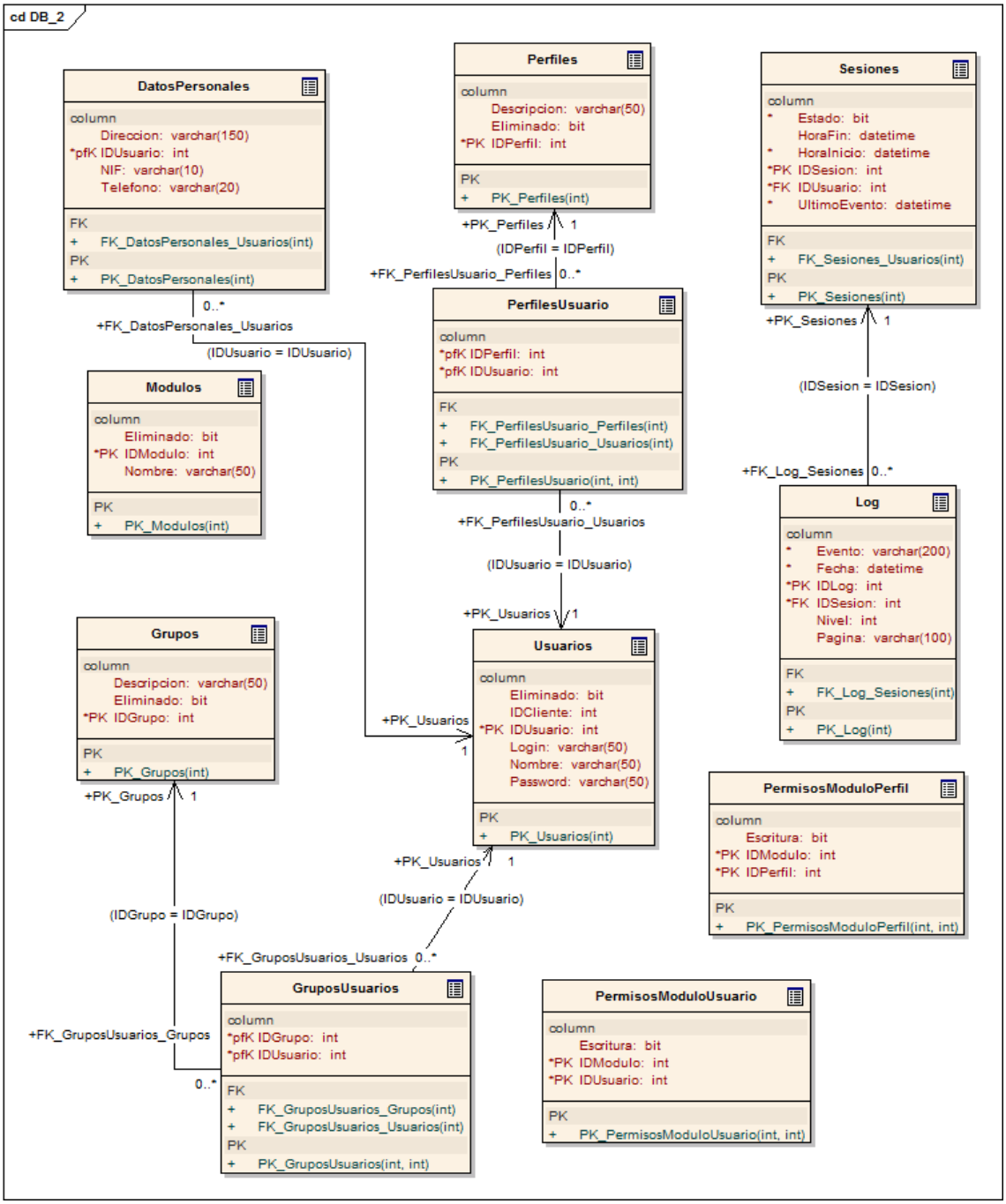


Figura 6.40. Diagrama Entidad-Relación (II)

6.5 Diseño de la Interfaz

A continuación se muestran las capturas que recogen el diseño de la interfaz del sistema, comentando brevemente aquellas en las que se considera oportuno aclarar algún aspecto relacionado.

6.5.1 Interfaz del Módulo de Gestión de Clientes

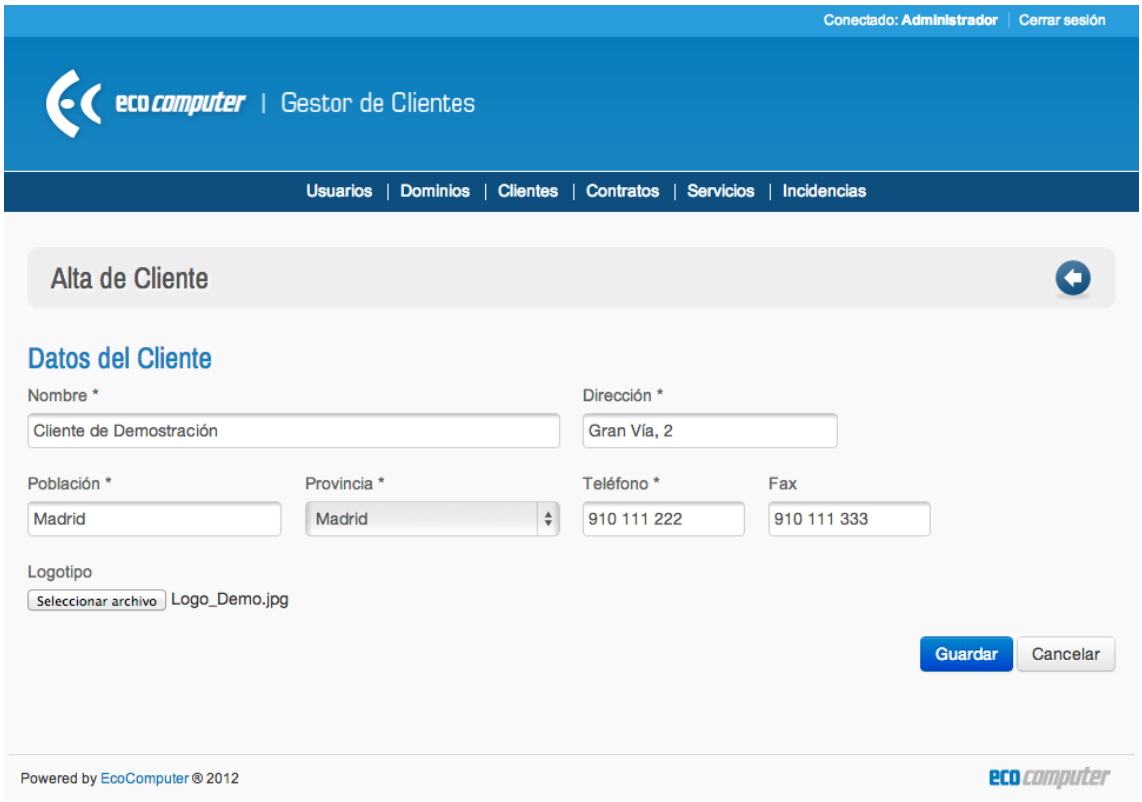


Figura 6.41. Interfaz Gestión de Clientes - Crear cliente

La vista de alta de cliente, similar a sus homólogas del resto de módulos, cuenta con un formulario basado en campos de texto y desplegables. Todos ellos serán validados y los errores encontrados se mostrarán en la parte inferior del componente.

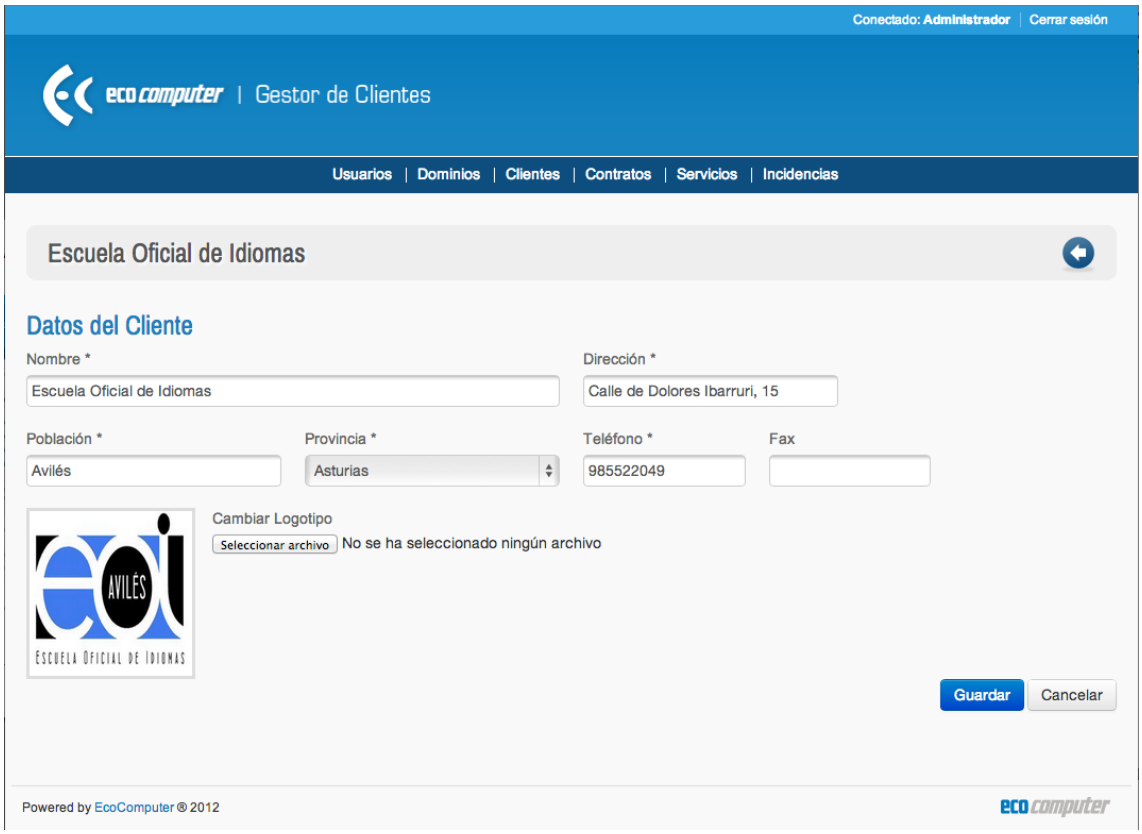


Figura 6.42. Interfaz Gestión de Clientes - Actualizar cliente

De apariencia casi idéntica a la anterior, la interfaz de actualización de elementos existentes cuenta con los datos originales de la entidad a modificar precargados en el formulario. Particularmente, en la actualización de clientes se muestra el logotipo actual del mismo.

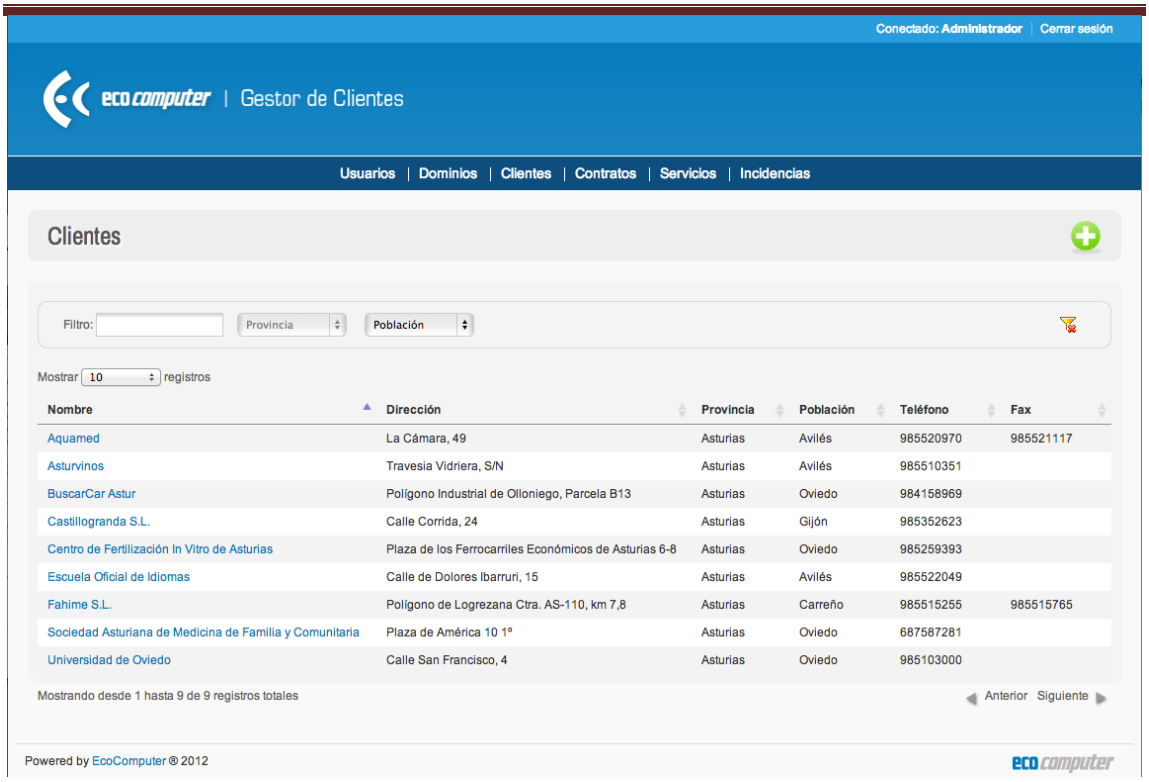


Figura 6.43. Interfaz Gestión de Clientes - Consulta listado de clientes

La vista de listado de clientes pretenderá mostrar de forma clara el conjunto de elementos existentes en el sistema. En el caso de objetos con gran número de campos, como es el caso de los contratos o las incidencias, se deben seleccionar aquellos que resulten más descriptivos para ser mostrados aquí.

Figura 6.44. Interfaz Gestión de Clientes - Detalle de cliente

La ficha del detalle de cliente muestra todos los datos del mismo, incluido su logotipo corporativo. Cuenta además con un acceso directo al módulo de incidencias, donde se mostrarán automáticamente las relativas al cliente consultado.

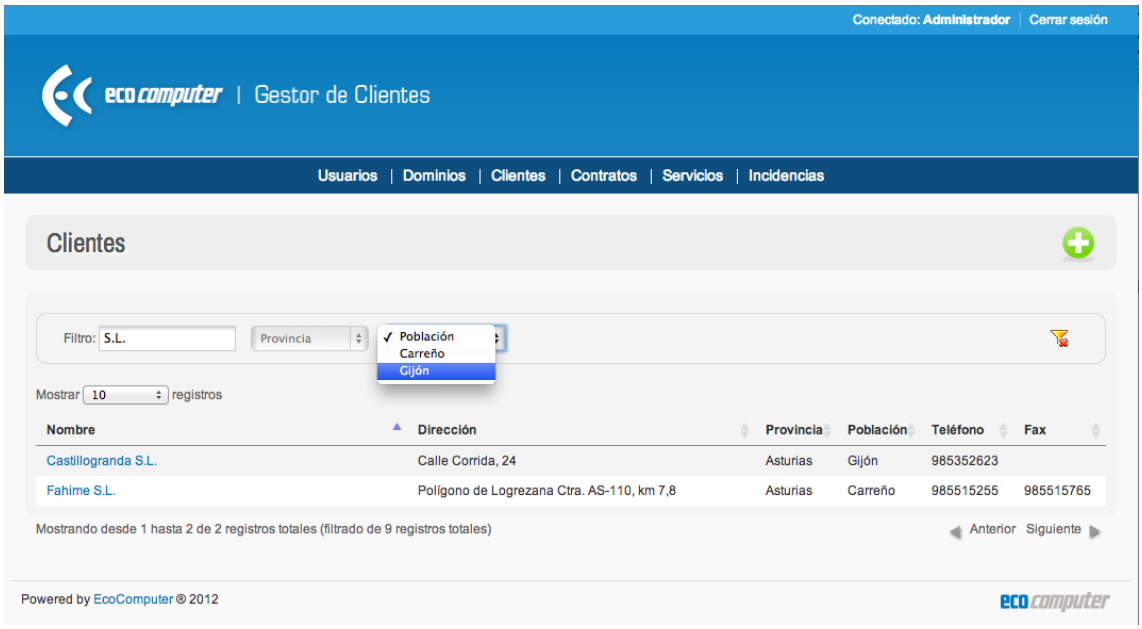


Figura 6.45. Interfaz Gestión de Clientes - Filtrado de registros

El filtrado de registros se comporta de manera inteligente, deshabilitando aquellos filtros que no corresponden a ninguna entidad existente.

6.5.2 Interfaz del Módulo de Gestión de Contratos

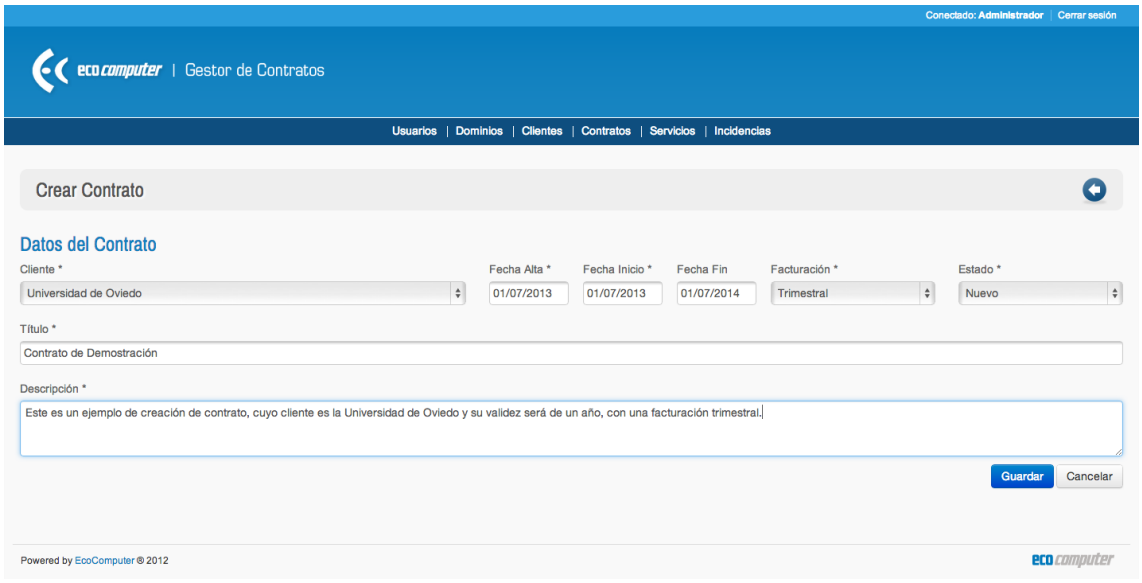


Figura 6.46. Interfaz Gestión de Contratos - Crear-Actualizar contrato

Similar a la ya descrita para el módulo de clientes, la vista de creación y actualización de contratos cuenta con un desplegable destinado a seleccionar el cliente asociado al mismo, así como varios selectores de fecha que muestran un calendario donde elegir la deseada.

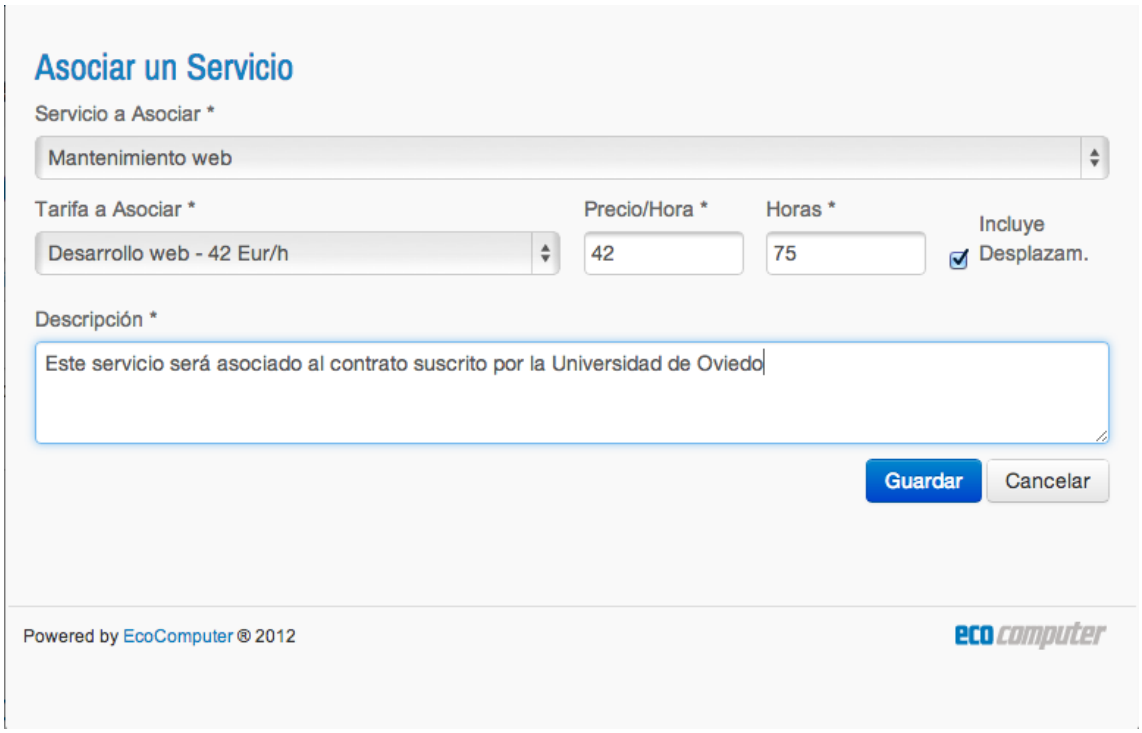


Figura 6.47. Interfaz Gestión de Contratos - Asociar servicio a contrato

Esta vista hace uso de los datos de servicios y tarifas del catálogo para asociar el deseado a un contrato activo.

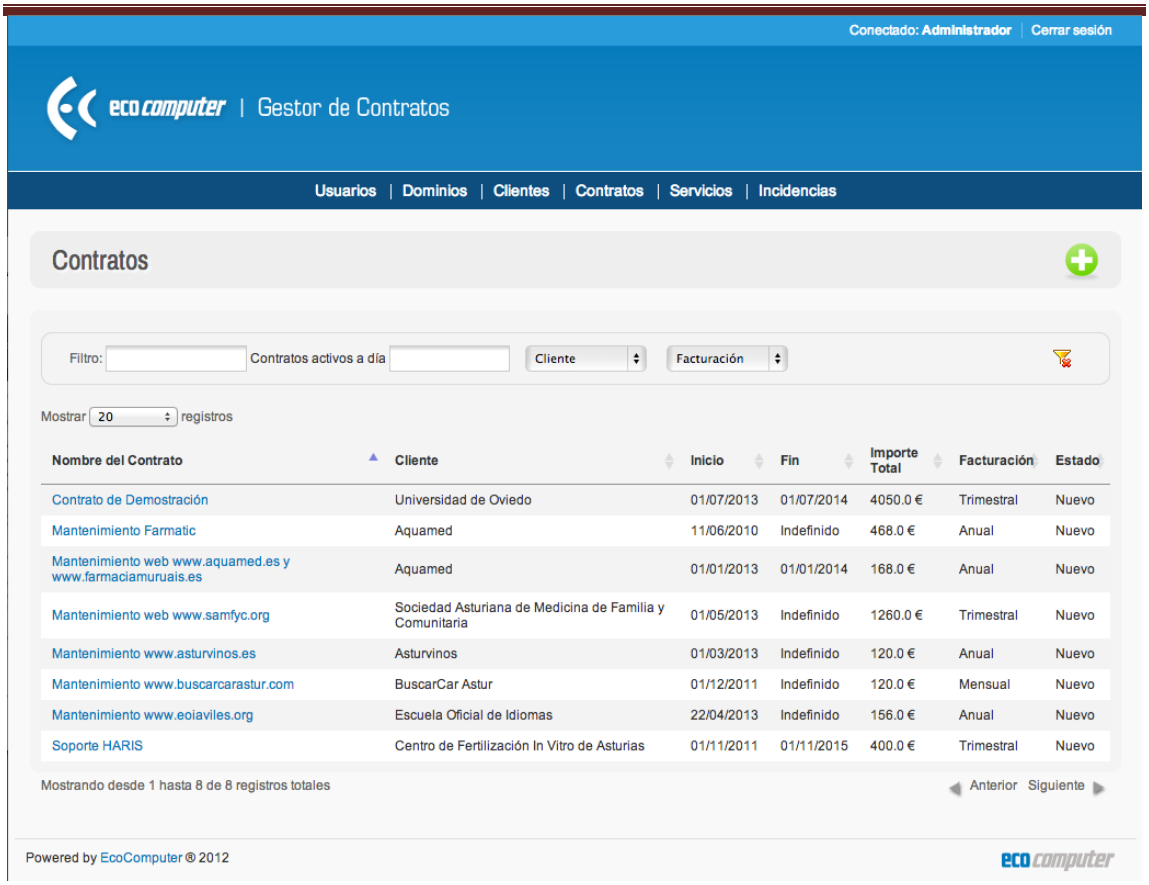


Figura 6.48. Interfaz Gestión de Contratos - Consultar listado de contratos

Figura 6.49. Interfaz Gestión de Contratos - Detalle del contrato

La vista detalle del contrato muestra todos los datos de la entidad, organizándolos en cajas que facilitan su rápida lectura. Además, el listado de servicios asociados al contrato, situado en la parte inferior de la interfaz, ofrece gran cantidad de información de forma clara y sintetizada.

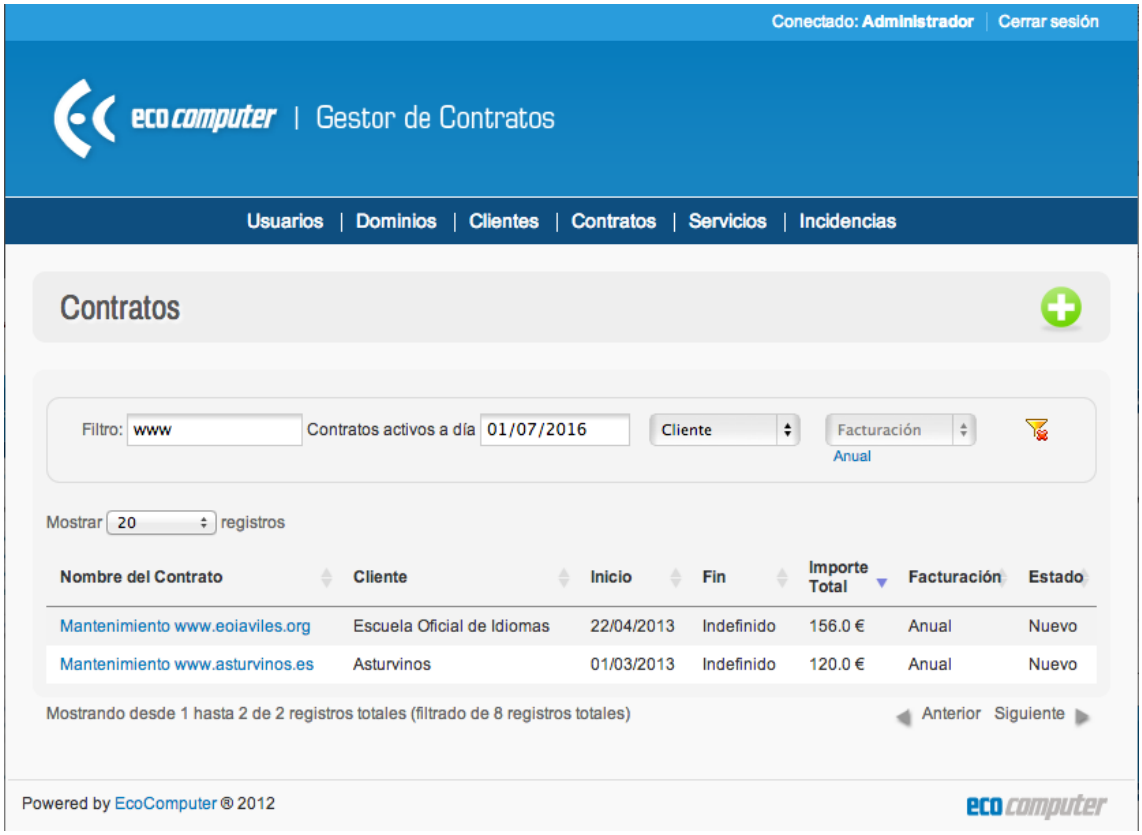


Figura 6.50. Interfaz Gestión de Contratos - Filtrado de registros

El filtro de registros de contratos añade la función de selección en base a una fecha, donde es posible filtrar aquellos contratos que están o han estado activos en un determinado momento.

6.5.3 Interfaz del Módulo de Gestión de Servicios

Figura 6.51. Interfaz Gestión de Servicios - Crear servicio

Se trata de uno de los formularios más simples de la aplicación, pues la descripción es el único dato necesario para dar de alta un nuevo servicio en el sistema.

Figura 6.52. Interfaz Gestión de Servicios - Asociar-Actualizar tarifa

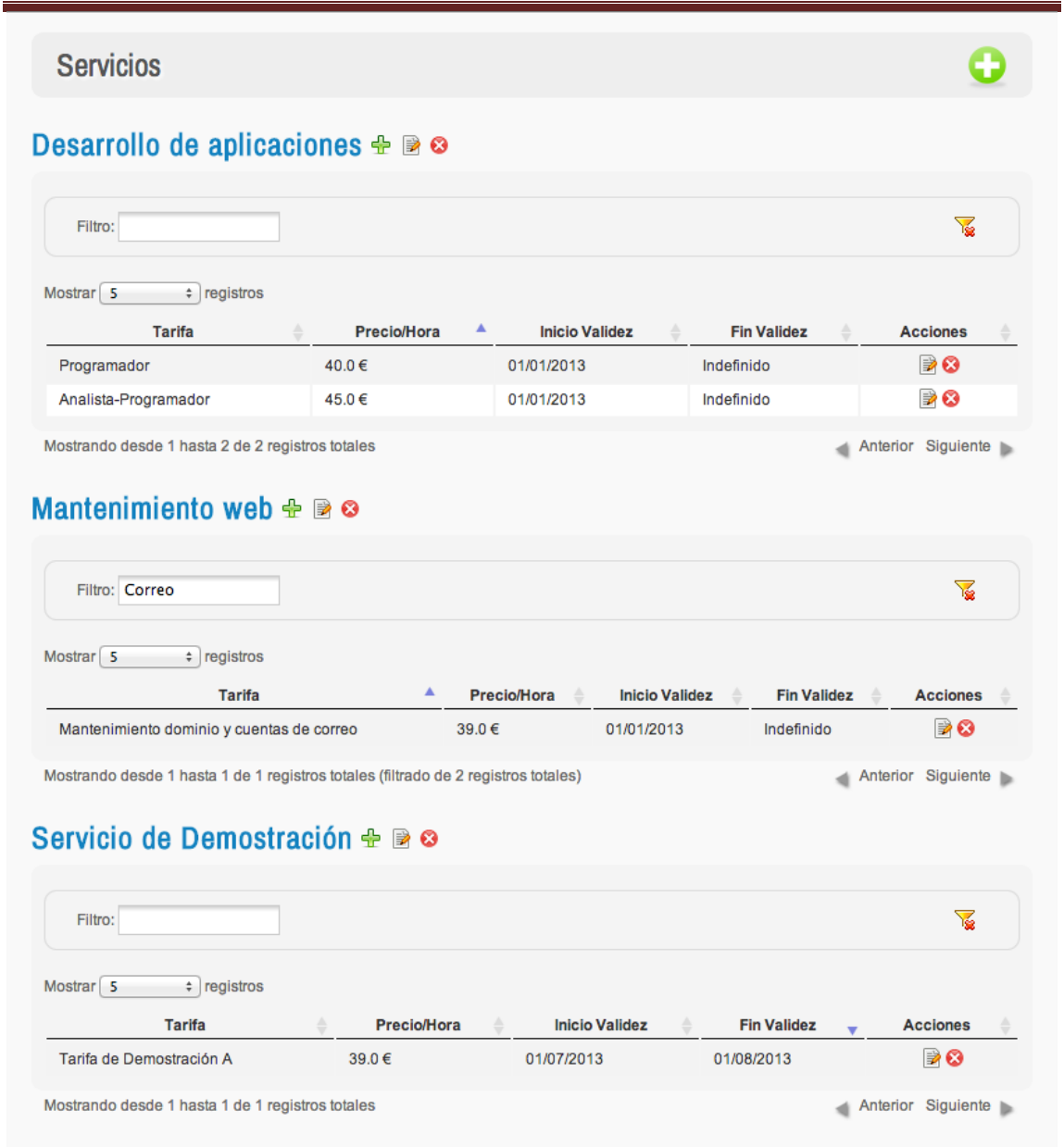


Figura 6.53. Interfaz Gestión de Servicios - Consulta del catálogo y filtrado de registros

A pesar de mostrar una elevada cantidad de información, la estructuración de los contenidos del catálogo de servicios y sus tarifas ha sido diseñada de modo que se facilite su lectura en la medida de lo posible.

6.5.4 Interfaz del Módulo de Gestión de Incidencias

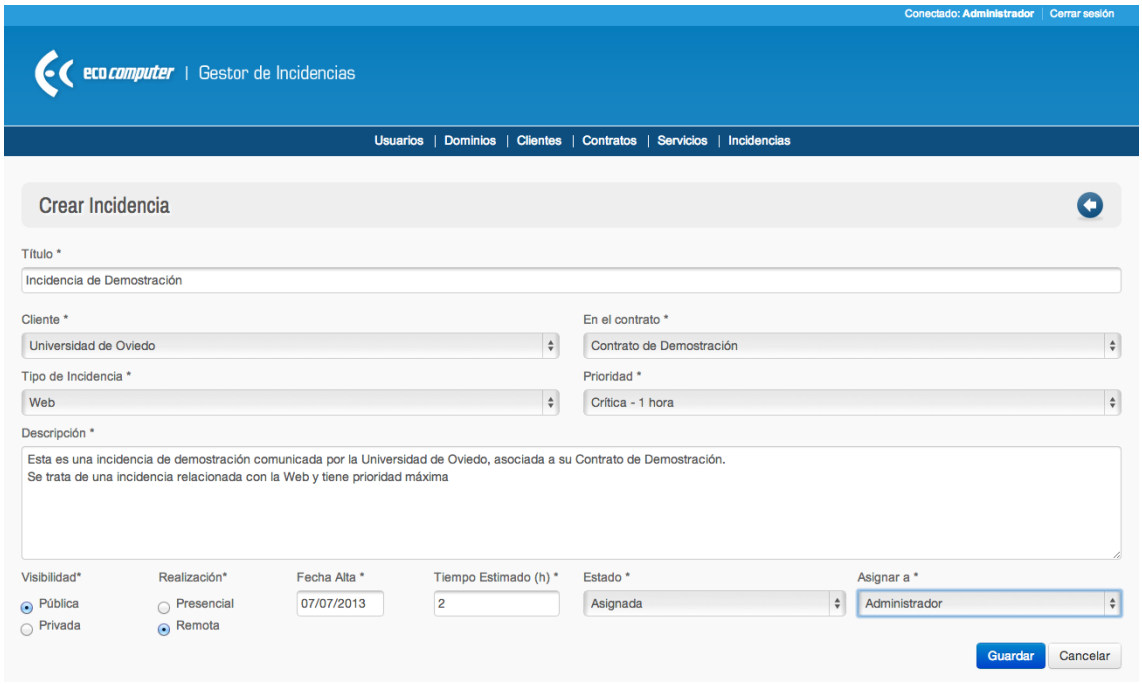


Figura 6.54. Interfaz Gestión de Incidencias - Crear-Actualizar Incidencia

La interfaz de creación y actualización de incidencias es la más completa de las existentes. Se trata de un formulario complejo, con una cantidad considerable de campos a completar, distribuidos de forma intuitiva para el usuario.

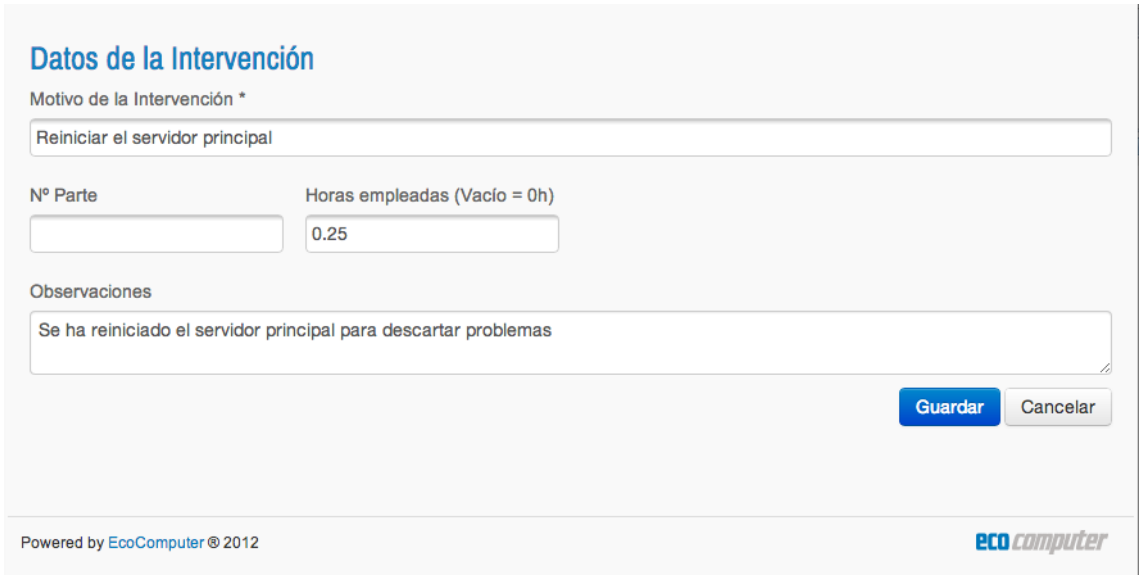


Figura 6.55. Interfaz Gestión de Incidencias - Crear-Actualizar intervención

El elemento más destacable de la vista de creación y modificación de intervenciones es el campo de observaciones, donde se ha provisto un espacio suficiente para anotar todos los datos necesarios.

Figura 6.56. Interfaz Gestión de Incidencias - Adjuntar-Modificar fichero

A la hora de adjuntar un fichero a una incidencia, el formulario mostrado es capaz de mostrar una vista de selección, dependiente del sistema operativo, donde poder acceder al sistema de ficheros del equipo de usuario y seleccionar el que se subirá al servidor.

Figura 6.57. Interfaz Gestión de Incidencias - Relacionar incidencia

Aunque se pensó en listar las incidencias susceptibles de ser relacionadas en un desplegable, fue necesario optar por un campo de texto, debido a que el número de incidencias existentes sería tan elevado que no sería útil mostrar un listado tan extenso.

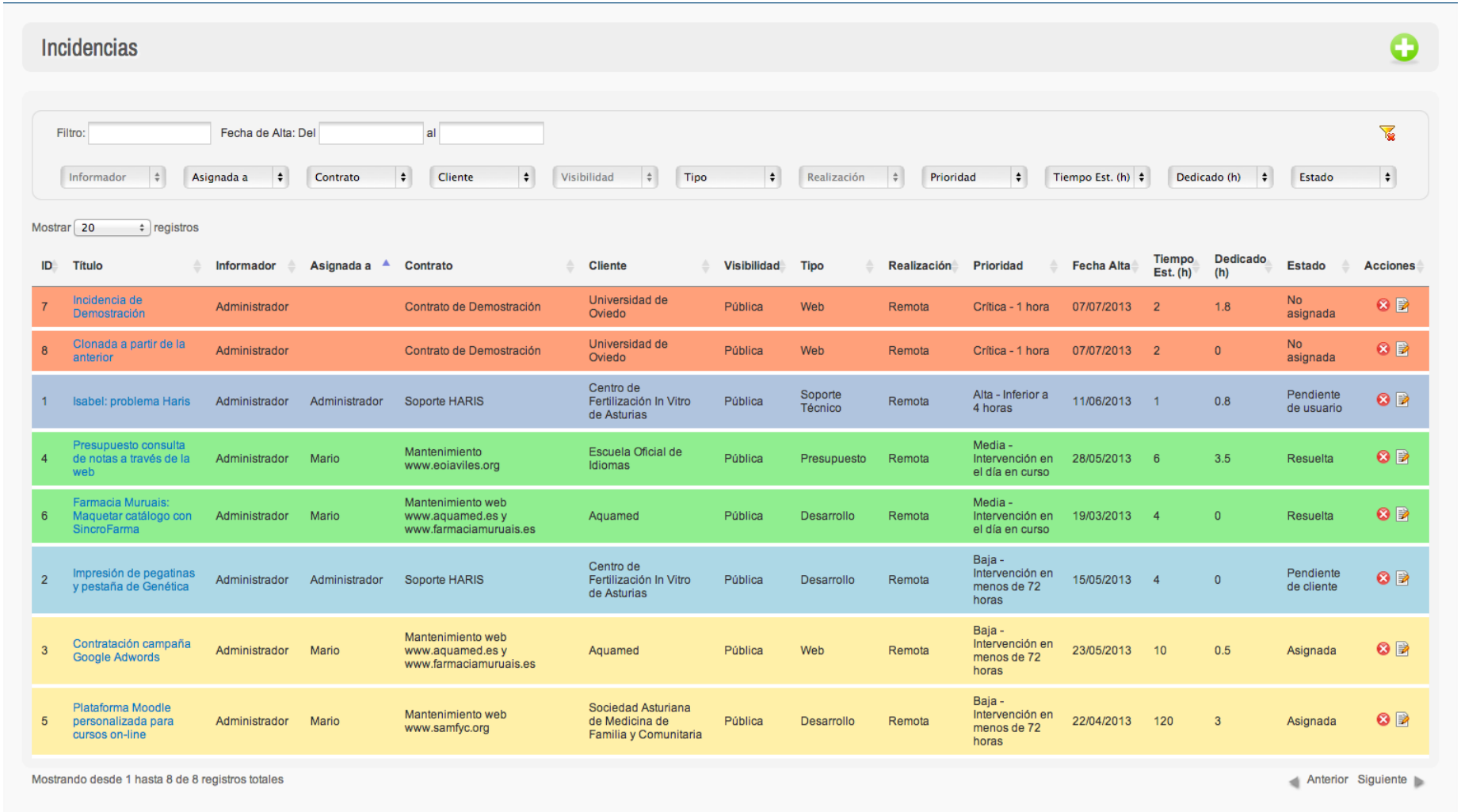


Figura 6.58. Interfaz Gestión de Incidencias - Listado de incidencias

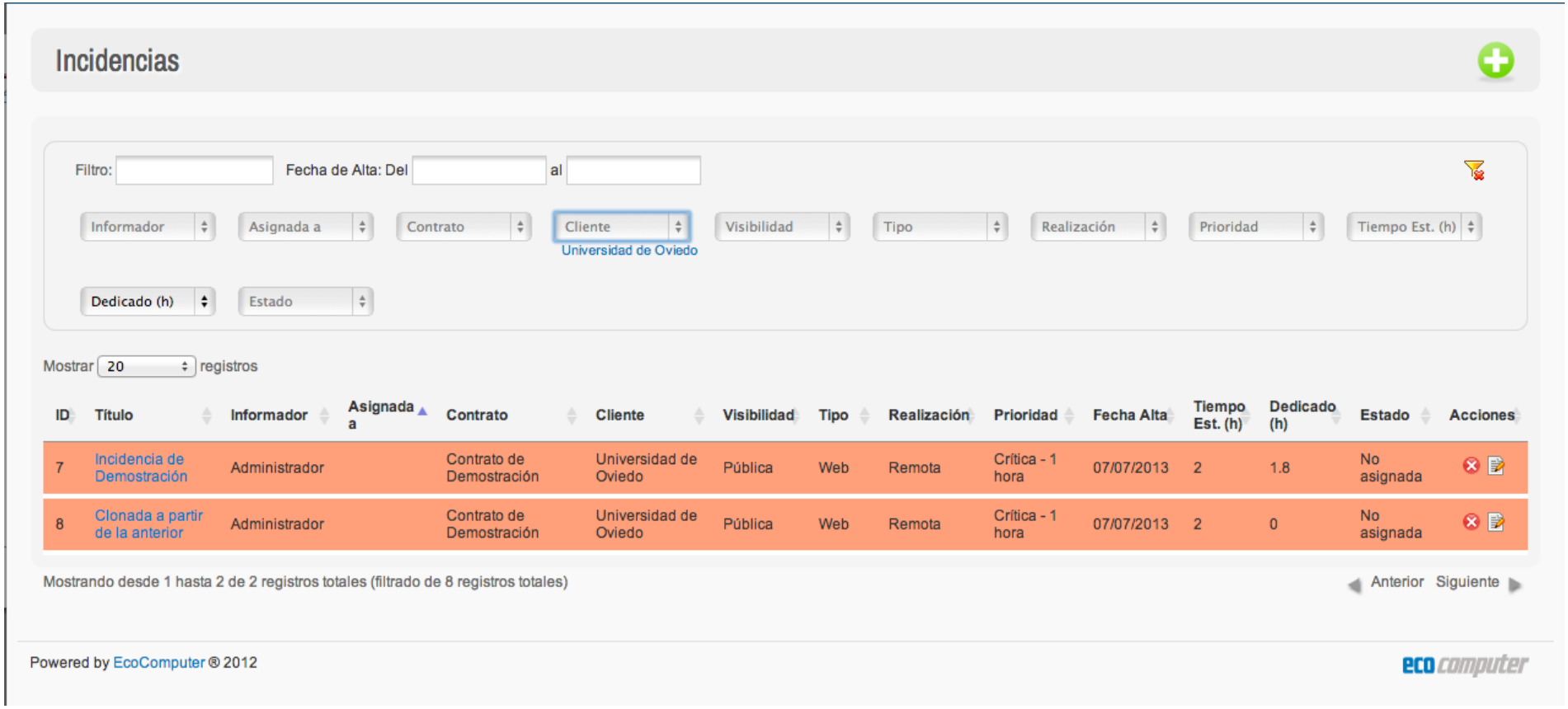


Figura 6.59. Interfaz Gestión de Incidencias - Filtrado y ordenación de registros

De nuevo, el filtro del listado de incidencias se convierte en el más complejo de los existentes. Ha sido necesario distribuir los elementos que lo componen para asegurar su usabilidad y utilidad.

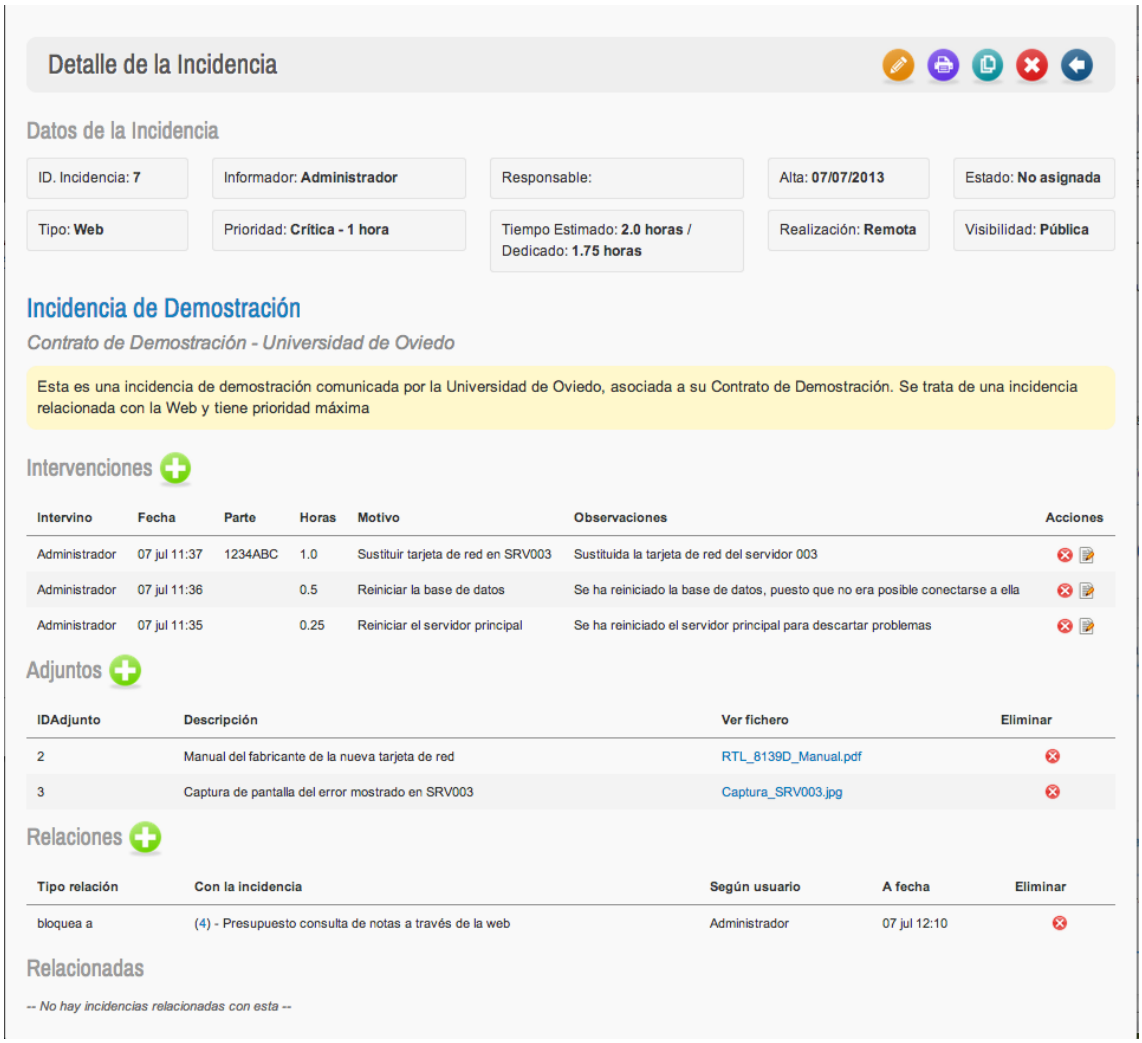


Figura 6.60. Interfaz Gestión de Incidencias - Detalle de incidencia

En esta vista se muestran, además de los datos básicos de la incidencia, todos los elementos anexos a la misma, separados en tres listados diferentes para mejorar su organización.

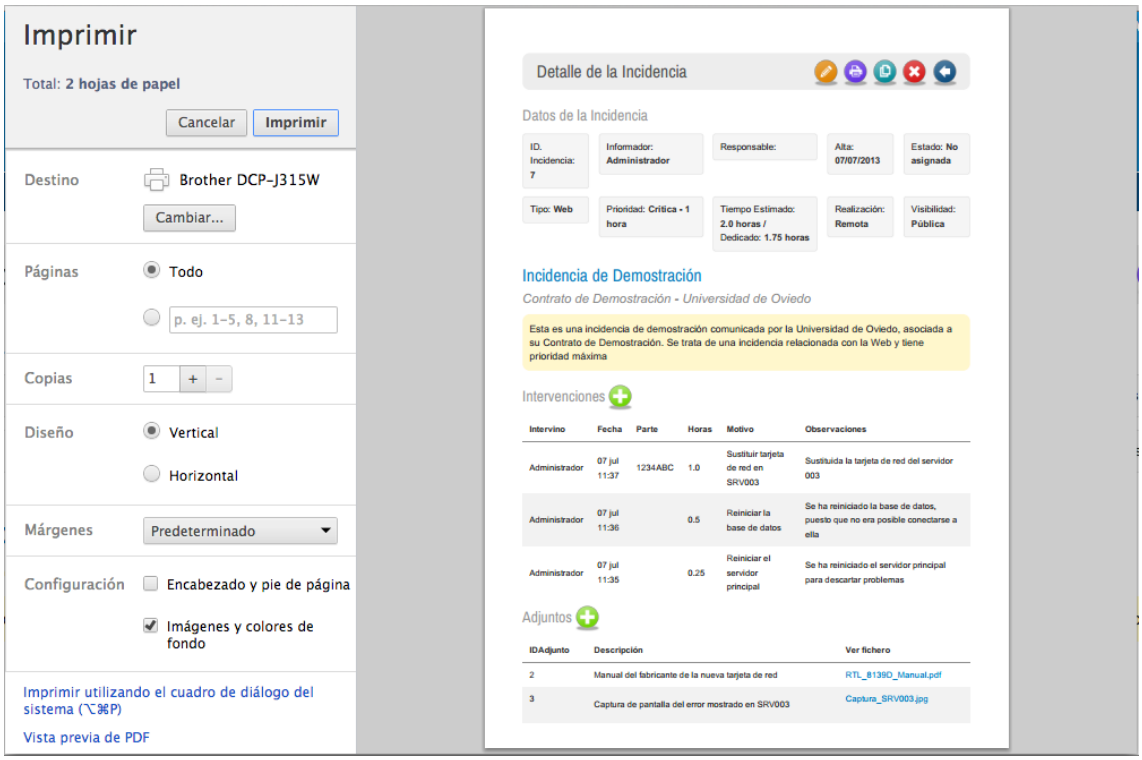


Figura 6.61. Interfaz Gestión de Incidencias - Generar informe imprimible

La función de impresión nativa del navegador es capaz de generar el informe mostrado y exportarlo a un fichero PDF o enviarlo a una impresora instalada en el sistema.

6.6 Especificación Técnica del Plan de Pruebas

6.6.1 Pruebas Unitarias y de Integración

Se llevará a cabo una batería de pruebas unitarias sobre cada una de las clases del modelo encargadas de realizar las operaciones de inserción y consulta de datos en la aplicación. De este modo, se asegurará el buen funcionamiento de las operaciones individuales de cara al proceso de integración de las mismas en el conjunto del sistema.

Por medio de este tipo de pruebas se asegurará el correcto funcionamiento del sistema a medida que se integren los elementos que lo componen, desarrollados y probados individualmente.

Este conjunto de pruebas corresponde con las ya descritas en la fase de análisis, especificando en este punto qué entrada se proporciona al sistema y cuál es el resultado esperado y obtenido en cada caso.

6.6.2 Pruebas de Usabilidad y Accesibilidad

Tal y como se ha mencionado en secciones anteriores, dado que el proyecto se trata de un desarrollo a medida, las preferencias del cliente serán priorizadas sobre los requisitos y recomendaciones de usabilidad y accesibilidad. No obstante, se tratará siempre de asesorar al cliente en este aspecto, buscando el equilibrio entre la funcionalidad requerida y las directrices básicas de usabilidad y accesibilidad. Con esta idea y de cara a intentar mejorar la usabilidad del sistema, se utilizará la evaluación heurística propuesta por el experto Yusef Hassan Montero. Finalmente, se realizará una evaluación de accesibilidad para verificar su conformidad respecto a las WCAG 2.0 del W3C.

6.6.3 Pruebas de Rendimiento

Dado que la aplicación desarrollada está concebida para su uso en una PYME, no se han previsto problemas derivados del rendimiento. La cantidad de potenciales usuarios de la aplicación generarán un escaso nivel de concurrencia y no supondrá un problema necesario de analizar y controlar. Además, la administración de los servidores y sistemas implicados en el proyecto no está dentro del control del desarrollador, si no que es el cliente quien se ocupa de ello.

En cuando al rendimiento del código, el desarrollo se ha realizado pensando en mantener un alto nivel de calidad y limpieza a la vez que se intentaba alcanzar la menor complejidad posible. Las conexiones con la base de datos, el aspecto más susceptible de presentar cadencias en las operaciones, son parte de la llamada Intranet de la aplicación y por tanto no es parte de este proyecto. No obstante, las conexiones se realizan a través de un pool de conexiones, por lo que el rendimiento es óptimo en dicho aspecto.

Capítulo 7. Implementación del Sistema

7.1 Estándares y Normas Seguidos

7.1.1 XHTML 1.1

Acrónimo en inglés de *eXtensible HyperText Markup Language*, hace referencia al lenguaje de marcado concebido como estándar en reemplazo de HTML. Se trata básicamente de HTML expresado como XML válido, siendo más estricto a nivel técnico. Su creación viene motivada por el avance en el proyecto del W3C de lograr una Web semántica, donde la información y su forma de ser representada estén claramente diferenciadas.

Como ventajas principales de XHTML sobre HTML destacan las siguientes:

- Posibilidad de incorporación de elementos de diversos espacios de nombres XML
- El agente de usuario o navegador no precisa de la implementación de heurísticos para detectar qué quiso dar a entender el autor, por lo que el *parser* puede ser simplificado en gran medida.
- Dado que se expresa como XML, es sencillo emplear herramientas creadas para el procesamiento de documentos XML, tales como editores, XSLT, etc.

En el desarrollo del presente proyecto se trata de seguir en la medida de lo posible las normas establecidas por la recomendación, lo cual no será posible en todo momento por la naturaleza sintáctica de Struts 2 y los requisitos impuestos por el cliente de la aplicación, el cual comprende y asume la no adopción completa del lenguaje.

7.1.2 CSS 2

Las hojas de estilo en cascada o *Cascading Style Sheets* (CSS) denotan al lenguaje basado en hojas de estilos empleado para describir el aspecto y formato de un documento cuyo contenido está escrito en lenguaje de marcado. Un resumen de las ventajas que se obtienen con su uso viene dado por las siguientes propiedades:

- Control centralizado de la presentación
- Separación del contenido de la presentación
- Adecuación al dispositivo y propiedades de la conexión
- Mejoras, en algunos casos, en la accesibilidad del documento

Su validación con el estándar será un objetivo a intentar cumplir en este proyecto, tomando en cuenta que esto puede no ser posible llegado un punto del desarrollo en el que se precise el uso de elementos no estándar.

7.1.3 Pautas de Accesibilidad para el Contenido Web (WCAG) 2.0

Las Pautas de Accesibilidad para el Contenido Web (WCAG) 2.0 cubren un amplio rango de recomendaciones para crear contenido Web más accesible. Seguir estas pautas permite crear un contenido más accesible para un mayor número de personas con discapacidad, incluyendo ceguera y baja visión, sordera y deficiencias auditivas, deficiencias del aprendizaje, limitaciones cognitivas, limitaciones de la movilidad, deficiencias del habla, fotosensitividad y combinaciones de las anteriores.

Las WCAG 2.0 suceden a las Pautas de Accesibilidad para el Contenido Web (WCAG) 1.0, que fueron publicadas como Recomendación del W3C en mayo de 1999. Aunque es posible cumplir con las WCAG 1.0 o con las WCAG 2.0 (o con ambas), el W3C recomienda que los contenidos nuevos o actualizados sigan las WCAG 2.0.

7.1.4 Buenas Prácticas de Programación

Secundar las normas establecidas sobre buenas prácticas de programación aporta al desarrollo las propiedades deseadas de mantenibilidad, comprensibilidad y legibilidad, los cuales son principios vitales en la construcción de un sistema de una extensión considerable.

Es por ello que en la implementación de este proyecto se seguirán los principios definidos propios de cada lenguaje, procurando aportar cualidades tales como organización física y lógica de ficheros, uso correcto del lenguaje, identación y espaciado, etc.

7.2 Lenguajes, Frameworks y Componentes Empleados

A continuación se expone un listado con la descripción de los diferentes lenguajes, herramientas y tecnologías que han sido empleados durante la implementación del sistema. Estos elementos son de tipos moderadamente dispares y su utilidad es muy variopinta en algunos casos, por lo que se clasificaran de acuerdo a cómo han sido utilizados de forma particular.

7.2.1 Lenguajes de Programación

7.2.1.1 Java

Java es un lenguaje de programación orientado a objetos desarrollado por Sun Microsystems, hoy propiedad de Oracle, a principios de los años 90. El lenguaje toma gran parte de su sintaxis de lenguajes ya existentes, como C y C++, pero tiene un modelo de objetos más simple, a la vez que prescinde de ciertos componentes de bajo nivel que complican la programación e inducen a errores, como la manipulación de punteros a memoria.

Las aplicaciones Java son compiladas a código intermedio, aunque existe la opción de compilación a código máquina nativo. En tiempo de ejecución, el bytecode es normalmente interpretado o compilado a código nativo para la ejecución.

La implementación original y de referencia del compilador, la máquina virtual (JVM) y las librerías de clases de Java fueron desarrolladas por Sun Microsystems en 1995. Desde entonces, Sun, ahora Oracle, ha controlado las especificaciones, el desarrollo y evolución del lenguaje a través del Java Community Process a la vez que terceros han desarrollado también implementaciones alternativas de estas tecnologías, algunas incluso bajo licencias de software libre.

Java es el lenguaje de programación utilizado en la implementación de toda la lógica del proyecto, complementándolo además con componentes adicionales para alcanzar los objetivos deseados.

7.2.1.2 JavaScript

JavaScript (JS) es un lenguaje de programación interpretado, dialecto del estándar ECMAScript. Se define como un lenguaje orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico.

Se utiliza principalmente del lado del cliente *(Client-Side)*, implementado como parte de un navegador web permitiendo mejoras en la interfaz de usuario y páginas web dinámicas y en bases de datos locales al navegador. Existe una forma de JavaScript del lado del servidor *(Server-Side JavaScript)*, enfocado al uso en aplicaciones externas a la web, como es el caso de

documentos PDF y aplicaciones de escritorio. Todos los agentes de usuario modernos interpretan el código JavaScript integrado en las páginas web y utilizan una implementación del Document Object Model (DOM) para la interacción con el propio código HTML de la página en cuestión.

JavaScript se diseñó con una sintaxis similar a la del lenguaje C, aunque adopta nombres y convenciones del lenguaje de programación Java. Sin embargo Java y JavaScript no están directamente relacionados y tienen semánticas y propósitos diferentes. Cuenta también con influencias de otros lenguajes como Perl, Self o Python. Fue desarrollado originalmente por Brendan Eich de Netscape Communications Corp. en 1995 bajo el nombre de "Mocha", siendo el término "JavaScript" una marca registrada de Oracle Corporation.

Su última versión estable es la 1.8.5 de marzo de 2011, aunque una nueva edición se encuentra en fase de desarrollo e incluirá nuevas características tales como organización en paquetes, espacios de nombres y definición explícita de clases.

7.2.1.3 SQL

SQL (por sus siglas en inglés, *Structured Query Language*) es un lenguaje de acceso a bases de datos que explota la flexibilidad y potencia de los sistemas relacionales y permite así gran variedad de operaciones. Se trata de un lenguaje declarativo de alto nivel que, gracias a su fuerte base teórica y su orientación al manejo de conjuntos de registros, permite una alta productividad en codificación y orientación a objetos. De esta forma, una sola sentencia puede equivaler a uno o más programas que se utilizarían en un lenguaje de bajo nivel.

7.2.2 Frameworks para el Desarrollo

7.2.2.1 Apache Struts

Brevemente descrito con anterioridad, Apache Struts es un framework de código abierto concebido para el desarrollo de aplicaciones web Java EE utilizando y ampliando la API Java Servlet para fomentar la adopción del patrón Modelo-Vista-Controlador (MVC).

En una aplicación web Java EE tipo, el cliente realiza peticiones al servidor mediante un formulario web. La información enviada en dicho formulario es recogida por un Servlet Java, el cual interactúa con una base de datos y genera una respuesta de tipo HTML, que bien puede ser directamente enviada al cliente, o bien será post-procesada por un módulo JavaServer Pages (JSP) que entremezclará código HTML y Java para conseguir un resultado más elaborado. Ambas aproximaciones son funcionales, pero son desaconsejables para proyectos de considerable envergadura, puesto que mezclan la parte de lógica de la aplicación con la parte de presentación, lo que a menudo da lugar a un código con escasa mantenibilidad.

El propósito final de Struts es separar el modelo de la vista y del controlador. Para ello, Struts proporciona el controlador, denominado ActionServlet, y facilita el desarrollo de plantillas para la vista o capa de presentación. El programador es entonces responsable de implementar el código del modelo y definir correctamente la configuración del núcleo de

Struts, el denominado *struts-config.xml*, el cual combina de forma conjunta y ordenada modelo, vista y controlador.

Esta aproximación difiere en gran medida de las dos presentadas. En este caso, las peticiones del cliente son enviadas al controlador en forma de *Actions*, las cuales son definidas en el fichero XML de configuración. El controlador se encargará de analizar la petición recibida e invocar a la *Action Class* correspondiente, la cual interactuará con el código del modelo. La información se intercambia entre el modelo y la vista por medio de un tipo especial de *JavaBean* y un amplio conjunto de etiquetas que permite la lectura y escritura de los contenidos de dichos *Beans* en la capa de presentación, sin hacer uso de código Java incrustado.

La versión empleada en este proyecto es la 2.2.1 por cuestiones de compatibilidad con otros componentes, siendo la más reciente la 2.3.14

7.2.2.2 Spring

Desarrollado inicialmente por Rod Johnson para la publicación de su libro *Expert One-on-One J2EE Design and Development* en octubre de 2002 y publicado bajo la licencia Apache 2.0, se trata de un producto en constante evolución y adaptado a las últimas innovaciones en su campo.

Aunque Spring se define como un conjunto de herramientas para la construcción de aplicaciones Java, existen extensiones para el desarrollo de aplicaciones web en la plataforma Java EE. Spring no impone el uso de ningún tipo de modelo de programación, factor que lo ha hecho popular en la comunidad de desarrolladores Java como alternativa, reemplazo y añadido del modelo Enterprise JavaBean (EJB).

Sus componentes ofrecen una gran variedad de servicios, técnicas y paradigmas de programación, tales como un contenedor de Inversión de Control, programación orientada a aspectos, acceso a datos, gestor transaccional, Modelo-Vista-Controlador, acceso remoto, convención sobre configuración, procesamiento por lotes, autenticación y autorización, administración remota, mensajería y herramientas de pruebas.

En el desarrollo del proyecto se utilizará la versión 2.2.1 distribuida junto con las librerías de Struts con objeto de aprovechar su funcionalidad dedicada a la inyección directa de dependencias.

7.2.3 Componentes y Librerías

7.2.3.1 JavaServer Pages Standard Tag Library (JSTL)

La librería JavaServer Pages Standard Tag Libray o JSTL es un componente de la plataforma de desarrollo de aplicaciones web Java EE. Extiende la especificación de JSP añadiendo una librería de etiquetas JSP empleadas en tareas comunes, tales como el procesado de datos XML, ejecución condicional, bucles, acceso a bases de datos e

internacionalización. JSTL se desarrolla bajo el Java Community Process (JCP), definida en la JSR 52. Su versión 1.2 fue lanzada en mayo de 2006 y es la empleada en este proyecto.

JSTL proporciona un método efectivo de integración de lógica en una página JSP sin recurrir al uso directo de código Java incrustado. Esto favorece la mantenibilidad del código a la vez que fomenta la separación de responsabilidades entre el desarrollo de la lógica de la aplicación y la interfaz de usuario.

7.2.3.2 Object Graph Navigation Language (OGNL)

El Object Graph Navigation Language (OGNL) es un Lenguaje de Expresiones (EL) para Java, desarrollado por OGNL Technology y publicado con licencia de código abierto. Se desarrolló para ser usado como un subconjunto más simple de expresiones que el soportado por el lenguaje Java, permitiendo el uso de *getters* y *setters* para las propiedades de los objetos y la ejecución de métodos de clases Java, además de permitir la manipulación sencilla de arrays.

Su uso se centra en aplicaciones web Java EE por medio de taglibs. Su última versión estable es la 3.0.5, liberada en abril de 2012 y utilizada en el presente proyecto. Actualmente, la versión 4 de OGNL se desarrolla como parte del proyecto Apache Commons.

7.2.3.3 jQuery

jQuery es una librería JavaScript diseñada con el objetivo de simplificar el scripting de código HTML en el lado del cliente. Creada y lanzada en enero de 2006 por John Resig, su uso se ha popularizado con el paso del tiempo, contando en la actualidad con el privilegio de ser empleada en más del 60% de los 10.000 sitios web más visitados. Se trata de software de código abierto, licenciado bajo la MIT License.

Su sintaxis fue diseñada para facilitar la navegación dentro de un documento HTML, permitiendo seleccionar elementos del DOM, crear animaciones, manejar eventos y desarrollar aplicaciones AJAX.

La versión empleada en este proyecto es la 1.9.1 y se utiliza con el propósito de mejorar y extender las posibilidades que JSP ofrece de base.

7.2.3.4 jQuery UI

jQuery UI es una librería JavaScript construida sobre jQuery. Al igual que ésta, jQuery UI es una librería de código abierto, licenciada bajo la MIT License, compartiendo además su autor, John Resig, quien la presentó en septiembre de 2007.

Como principal objetivo tiene el facilitar la construcción de aplicaciones web interactivas. Para ello, provee una capa de abstracción para la interacción a bajo nivel, además de la implementación de animaciones de diversa complejidad, efectos avanzados y un completo catálogo *widgets* altamente personalizables.

Su última versión, la 1.10.2, es la empleada en este proyecto. Es importante recordar que, al reposar sobre jQuery, precisa de la versión 1.6 o superior de ésta para su funcionamiento.

7.2.3.5 DataTables

Al igual que jQuery UI, DataTables es un *plugin* para la jQuery que proporciona controles avanzados para la interacción con tablas HTML de forma flexible y basada en la mejora continua. El proyecto es liderado por Allan Fella, CEO de SpryMedia, empresa británica dedicada al desarrollo de interfaces de usuario, aunque su funcionalidad es extensible por medio de *plugins* desarrollados por terceros que se incorporan al proyecto principal, siendo todos sus componentes ofrecidos bajo licencia de código abierto. Ofrece además soporte comercial a sus usuarios.

Entre sus características más reseñables, destaca el soporte a paginación variable, filtrado *on-the-fly*, ordenación y ocultación de columnas y personalización completa basada en CSS, todo ello definido sobre un modelo de interacción con el DOM no destructivo. Sin embargo, uno de sus puntos débiles es su documentación. A pesar del gran esfuerzo del autor por mantener un amplio catálogo de recursos de consulta, así como un foro de ayuda, resulta, cuando menos, caótico encontrar respuesta a una pregunta o problema que surja en su implementación.

La versión utilizada en el presente proyecto es la 1.9.4, siendo la más reciente en la actualidad. Además, se utiliza el *add-on* ColumnFilterWidgets , desarrollado por Dylan Kuhn para complementarlo y mejorar las funciones de filtrado en las tablas de presentación de datos.

7.2.3.6 Java DataBase Connectivity

Java DataBase Connectivity (JDBC) es una tecnología de acceso a datos de Oracle Corporation. Se trata de una API para Java que define cómo debe acceder un cliente a una base de datos, proporcionando métodos para consultar y modificar el contenido de la base de datos. Está orientada a bases de datos de tipo relacional.

Forma parte del Java Development Kit (JDK) desde su versión 1.1 de febrero de 1997. Su versión más reciente es la 4.1, incluida en la versión 7 de Java y empleada por tanto en este proyecto.

Su utilidad en este desarrollo se limita a la conectividad de la aplicación con la base de datos para realizar las operaciones necesarias sobre la misma. Para ello, se utiliza un driver específico en forma de librería, detallado a continuación.

7.2.3.7 Microsoft JDBC Driver for SQL Server

Se trata de un driver que provee los mecanismos de comunicación entre sistemas JDBC y soluciones SQL Server, SQL Azure y Parallel Data Warehouse (PDW). Es un desarrollo

propietario, ofrecido de forma gratuita a los desarrolladores que estén interesados en utilizarlo en sus proyectos. Se categoriza como driver JDBC de tipo 4 y ofrece conectividad a bases de datos por medio de las APIs estándar de JDBC.

La versión de la que se hace uso en este sistema es la 4.0, con fecha de liberación de junio de 2012.

7.2.3.8 Twitter Bootstrap

Bootstrap hace referencia a una colección de herramientas desarrollada por Twitter y enfocada al desarrollo de sitios y aplicaciones web. Está formada por plantillas HTML y diseños CSS, tales como tipografías, formularios, botones, gráficas, elementos de navegación y de maquetado, además de diversas extensiones JavaScript.

En su versión 2.3.2, se trata del proyecto más popular en la plataforma GitHub en la actualidad y cuenta con implementaciones en importantes sitios web, tales como el de la NASA o el operador de TV por cable americano MSNBC.

Su uso en este proyecto viene dado a través de una librería que lo implementa para su uso en conjunción con el framework Struts 2, denominada Struts2-Bootstrap-Plugin. Esta integración es obra del desarrollador alemán Johannes Geppert y forma parte del repositorio oficial de plugins de terceros de Apache Struts 2. La versión empleada es la 1.5.2, siendo a su vez la más reciente, siendo publicada en noviembre de 2012. Su historia no es demasiado extensa, puesto que se publicó por primera vez en febrero del mismo año, aunque se trata de un desarrollo suficientemente maduro como para considerar su uso.

7.2.3.9 Apache Tiles

Apache Tiles es un framework desarrollado para simplificar el desarrollo de interfaces de usuario en aplicaciones web. Para ello, Tiles permite a los desarrolladores definir fragmentos de una página web que serán incluidos en una página completa en tiempo de ejecución. A este tipo de fragmentos es a lo que se denomina *tiles*.

Además de esto, los *tiles* favorecen la no-duplicación de elementos comunes a diversas páginas, convirtiendo un desarrollo único en una plantilla reutilizable, las cuales dirigen el rumbo de un desarrollo con una apariencia consistente a lo largo de toda la aplicación.

Aunque la versión más reciente es la 2.2.2, la empleada en el proyecto es la 2.0.6, puesto que las versiones posteriores incluyen un agujero de seguridad que desaconseja su uso en entornos de producción.

7.3 Software y Herramientas Usados para el Desarrollo

7.3.1 Ubuntu Server 12.04 LTS

Ubuntu es una distribución Linux que ofrece un sistema operativo mayoritariamente enfocado a ordenadores de escritorio aunque también proporciona soporte para servidores.

Basada en Debian GNU/Linux, Ubuntu concentra su objetivo en la facilidad de uso, la libertad en la restricción de uso, los lanzamientos de versiones regulares y la facilidad en la instalación. Ubuntu es patrocinado por Canonical Ltd., una empresa privada fundada y financiada por el empresario sudafricano Mark Shuttleworth.

La versión más reciente es la 13.10, aunque la empleada en este proyecto ha sido la 12.04 al tratarse de una versión con soporte extendido (LTS), además de tratase de un requisito impuesto por el Departamento de Sistemas del cliente.

7.3.2 Windows Server 2008 R2

Windows Server 2008 es el nombre del sistema operativo de Microsoft enfocado para su uso en servidores. Al igual que Windows 7 , Windows Server 2008 se basa en el núcleo Windows NT 6.1. Entre las mejoras de esta edición, se destacan nuevas funcionalidades para el módulo *Active Directory*, nuevas prestaciones de virtualización y administración de sistemas, la inclusión de IIS 7.5 y el soporte para más de 256 procesadores.

Su uso en este proyecto estará vinculado al sistema de gestión de bases de datos empleado. Nuevamente, se trata de un requisito impuesto por el entorno existente en el cliente, puesto que es el sistema utilizado por el mismo para el despliegue de estos sistemas.

7.3.3 SQL Server 2005

SQL Server 2005 es un sistema de gestión de bases de datos basado en el modelo relacional, que utiliza el lenguaje T-SQL para realizar las consultas pertinentes. Entre sus características cabe destacar el soporte de transacciones, soporte de procedimientos almacenados, la posibilidad de trabajar desde un entorno gráfico, realizar las consultas en modo cliente- servidor, concurrencia de usuarios y la posibilidad de administrar información de otros servidores de base de datos. Su instalación se encuentra restringida a sistemas operativos de la familia Microsoft Windows.

A pesar de encontrarse publicada la versión 2008 e incluso 2012, el empleo de la versión 2005 para el presente proyecto viene impuesto por las necesidades específicas del cliente, quien hace uso de esta revisión concreta del software en sus sistemas.

7.3.4 SQL Server Management Studio

La aplicación SQL Server Management Studio Express es una herramienta empleada para configurar, manejar y administrar todos los componentes de una base de datos Microsoft SQL Server. Soporta tanto la entrada de datos por línea de comandos como a través de las interfaces gráficas que proporciona.

Fue desarrollada por el propio equipo de Microsoft y lanzada conjuntamente con SQL Server 2005. En este proyecto se utiliza la versión 9, subversión Express gratuita, puesto que es la última compatible con la versión del sistema de gestión de base de datos empleado.

7.3.5 Eclipse IDE

Eclipse es un entorno de desarrollo integrado compuesto por un espacio de trabajo o *workspace* y un sistema de *plugins* extensible que permiten su personalización. Está escrito casi íntegramente en Java y soporta el desarrollo en una gran variedad de lenguajes, entre los que se incluye éste. Entre las características relativas al desarrollo en dicho lenguaje, Eclipse incluye un conjunto de herramientas específicas, denominadas *Java Development Tools*, además de un compilador Java incremental y el modelado jerárquico del código de los ficheros fuente. Este diseño permite el uso de avanzadas técnicas de análisis y refactorización de código.

Es la herramienta de la que se ha hecho un uso más intensivo en este proyecto. La versión empleada es la Juno SR 4.1, publicada en septiembre de 2012.

7.3.6 Google Chrome

Chrome es el nombre del navegador web desarrollado por Google en base a distintos componentes y frameworks de código abierto. Parte de la base del proyecto de software libre Chromium. Es por ello que la parte realizada por Google está publicada bajo licencia BSD principalmente.

Son destacables la velocidad y estabilidad del navegador, fundamentados principalmente por su arquitectura de multiprocesamiento y aislamiento de procesos o *sandboxing*. Además de esto, integra por defecto las denominadas *Herramientas para Desarrolladores*, las cuales resultaron de suma utilidad en la construcción del sistema para tareas de depuración y pruebas.

7.3.7 FileZilla FTP

FileZilla es un cliente FTP multiplataforma de código abierto licenciado bajo la LPG-GNU. Ofrece soporte a los protocolos FTP, SFTP y FTPS (FTP sobre SSL/TLS). Cuenta con un gestor de sitios que permite almacenar un listado de sitios y sus datos de conexión, un registro

de mensajes con los comandos enviados y las respuestas recibidas desde el servidor, así como un gestor de ficheros simple que muestra el contenido de los directorios de cliente y servidor.

La versión utilizada ha sido la 3.7, siendo la más reciente hasta la fecha.

7.3.8 SQL Backup & FTP

SQL Backup & FTP ofrece la posibilidad de programar copias de seguridad, bien sean totales o incrementales de una o varias bases de datos, pudiendo incluso enviar una copia a distintos servicios de almacenamiento en la nube, tales como Microsoft SkyDrive, Dropbox o Google Drive, entre otros.

Debido al hecho de no tener permisos totales de acceso al servidor de bases de datos empleado en el sistema, resulta imposible la realización de copias de seguridad completas de la base de datos. Por ello, SQL Backup & FTP fue la solución adecuada para acceder a la base de datos con las credenciales disponibles y descargar una copia total de los datos, tanto de tablas y columnas, como de sus contenidos. La versión utilizada ha sido la 9.0.16.

7.3.9 Sublime Text

Sublime Text es un editor de texto y código fuente desarrollado en C++ y Python, concebido originalmente como una extensión del editor Vim. Se distribuye de forma gratuita, aunque no se trata de software de código abierto, por lo que se requiere una licencia para eliminar el aviso *Versión No Registrada*.

Su uso en este proyecto ha sido fundamentalmente la pre-visualización y edición rápida de ficheros de configuración, entre otros tipos. La versión empleada es la 2.0.1.

7.3.10 Enterprise Architect

La aplicación Enterprise Architect, desarrollada por Sparx Systems, es una herramienta para la creación de diagramas UML. Utiliza la última especificación oficial de UML y dispone de gran cantidad de funcionalidades extra, además de las posibilidades de diseño que ofrece. Se trata de una herramienta comercial.

Incluye soporte para los diagramas de comportamiento, compuestos por casos de uso, actividades, estado, interacción, secuencia y comunicación; además de diagramas estructurales como lo son los de paquetes, clases, objetos, composición, componentes y despliegue. Entre otras características, ofrece la creación automática de código fuente en más de diez lenguajes diferentes partiendo de los diagramas, y viceversa; informes en formato HTML y RTF; manejo de requisitos e integración con entornos de desarrollo como Microsoft Visual Studio o Eclipse.

Enterprise Architect ha sido usado en el desarrollo de este proyecto como herramienta de modelado de los diagramas mostrados en la presente documentación. Concretamente, la versión utilizada de la aplicación ha sido la 6.0, instalada a través de CrossOver 12 para Mac.

7.3.11 DIA Diagram Editor

DIA es una aplicación enfocada a la creación de diagramas estructurales, disponible para Windows, Linux y Mac OS X, publicada bajo licencia GPL.

Se basa en la funcionalidad de Visio, software de similar propósito incluido en el paquete Microsoft Office, soportando diversas clases de diagramas y permitiendo añadir nuevos tipos por medio de la instalación de kits de formas y objetos. La última versión publicada es la 0.97, que a su vez ha sido la empleada en este proyecto como herramienta complementaria a Enterprise Architect, puesto que este conlleva, en ocasiones, a un exceso de complejidad injustificado en el caso de la creación de diagramas sencillos.

7.3.12 Microsoft Office

El paquete Office de Microsoft contiene un conjunto de herramientas ofimáticas, algunas de las cuales han sido empleadas en este proyecto:

- Office Word, utilizado para la redacción de la presente documentación
- Office Excel, empleado para confeccionar algunas tablas de un modo más cómodo
- Office PowerPoint, usado para el desarrollo de la presentación del proyecto
- Office Project, necesario para definir y mantener la planificación
- Office Visio, utilizado para dibujar algunos gráficos y diagramas sencillos

Todos los componentes del paquete se han utilizado en su versión 2011 para Mac, salvo Office Project y Visio, no disponibles para este sistema operativo y utilizados mediante virtualización en Windows en la versión 2007.

7.4 Creación del Sistema

La descripción detallada de las clases implementadas ha sido incluida en la sección 13.3 del documento debido a su extensión y por no resultar imprescindible para la comprensión del proyecto.

Ha de tenerse en cuenta que, con el objetivo de reducir la extensión de la documentación, se han omitido elementos tales como la definición de los atributos en las clases y aquellos métodos comunes a todos los Actions, propios de la arquitectura de Struts, cuya documentación no aporta mayor información respecto a la implementación.

Capítulo 8. Desarrollo de las Pruebas

8.1 Pruebas Unitarias y de Integración

A continuación se perfila el resultado de la ejecución de las pruebas descritas anteriormente. Se muestra, para cada módulo de la aplicación, las pruebas llevadas a cabo sobre las operaciones del mismo y los resultados esperados y obtenidos de su ejecución.

8.1.1 Resultados de las Pruebas del Módulo de Gestión de Usuarios

Caso de Uso 1.1. Iniciar sesión	
Prueba	**Resultado Esperado**
Iniciar sesión en el sistema con unas credenciales válidas	El usuario es reconocido y se inicia la sesión
	Resultado Obtenido
	Se validan correctamente las credenciales del usuario y sus datos se incluyen en la sesión
Prueba	**Resultado Esperado**
Iniciar sesión en el sistema con unas credenciales no válidas	Se muestra un mensaje de error
	Resultado Obtenido
	Aparece un mensaje de error *"Usuario o contraseña no válidos"*

Caso de Uso 1.2. Cerrar sesión	
Prueba	**Resultado Esperado**
Cerrar la sesión de usuario en curso	El sistema cierra correctamente la sesión y muestra la pantalla de inicio
	Resultado Obtenido
	La sesión se destruye en el servidor y se muestra la interfaz de bienvenida

8.1.2 Resultados de las Pruebas del Módulo de Gestión de Clientes

Caso de Uso 2.1. Crear cliente	
Prueba	**Resultado Esperado**
Completar todos los datos del formulario correctamente	El cliente es almacenado en el sistema y aparece en el listado de clientes
	Resultado Obtenido
	La base de datos contiene el nuevo cliente con los datos proporcionados
Prueba	**Resultado Esperado**
Completar incorrectamente el formulario	Se muestran los errores de validación encontrados en cada campo del formulario
	Resultado Obtenido
	Aparece un error descriptivo en color rojo en cada uno de los campos con datos erróneos del formulario

Caso de Uso 2.2. Modificar cliente	
Prueba	**Resultado Esperado**
Introducir los nuevos datos del cliente correctamente	La información del cliente es actualizada y se muestra correctamente en su ficha
	Resultado Obtenido
	La base de datos contiene los datos actualizados del cliente modificado
Prueba	**Resultado Esperado**
Modificar incorrectamente el formulario	Se muestran los errores de validación encontrados en cada campo del formulario
	Resultado Obtenido
	Aparece un error descriptivo en color rojo en cada uno de los campos con datos erróneos del formulario

Caso de Uso 2.3. Eliminar cliente	
Prueba	**Resultado Esperado**
Eliminar el cliente y confirmar la acción	El cliente es dado de baja en el sistema y ya no aparece en el listado de clientes
	Resultado Obtenido
	El cliente es marcado con el *flag* "*Eliminado*" en la base de datos
Prueba	**Resultado Esperado**
Eliminar el cliente sin confirmar la acción	La acción se cancela y se retorna a la vista anterior
	Resultado Obtenido
	Se retorna a la vista anterior de la interfaz sin efectuar cambios

Caso de Uso 2.4. Consultar listado de clientes	
Prueba	**Resultado Esperado**
Acceder al módulo de gestión	Se muestra el listado de clientes dados de alta en el sistema

de clientes	**Resultado Obtenido**
	Aparece el listado de clientes existentes en la base de datos

Caso de Uso 2.4.1. Filtrar registros de clientes	
Prueba	**Resultado Esperado**
Introducir los parámetros de búsqueda deseados en el filtro	Se muestran únicamente los clientes cuyos datos contienen los parámetros especificados
	Resultado Obtenido
	Se muestran solamente los clientes cuyos datos contienen registros coincidentes con los parámetros especificados en los filtros

Caso de Uso 2.4.2. Ordenar registros de clientes	
Prueba	**Resultado Esperado**
Seleccionar la columna por la que se desea ordenar	Los clientes aparecen ordenados de acuerdo al campo seleccionado
	Resultado Obtenido
	El listado de clientes se ordena ascendente o descendentemente, de acuerdo al campo elegido

Caso de Uso 2.5. Consultar detalle del cliente	
Prueba	**Resultado Esperado**
Seleccionar el cliente deseado del listado	Se muestra la ficha completa del cliente seleccionado
	Resultado Obtenido
	Se muestran todos los datos del cliente almacenados en la base de datos

Caso de Uso 2.6. Ver incidencias del cliente	
Prueba	**Resultado Esperado**
Visualizando la ficha del cliente, seleccionar el enlace *Ver Incidencias*	Se muestra el listado de incidencias comunicadas por el cliente seleccionado
	Resultado Obtenido
	La interfaz cambia al módulo de gestión de incidencias y aplica el filtro en base al nombre del cliente, mostrándose únicamente aquellas que están asociadas al cliente en cuestión

8.1.3 Resultados de las Pruebas del Módulo de Gestión de Contratos

Caso de Uso 3.1. Crear contrato	
Prueba	**Resultado Esperado**
Completar todos los datos del formulario correctamente	El contrato es almacenado en el sistema y aparece en el listado de contratos
	Resultado Obtenido
	La base de datos contiene el nuevo contrato con los datos proporcionados
Prueba	**Resultado Esperado**
Completar incorrectamente el formulario	Se muestran los errores de validación encontrados en cada campo del formulario
	Resultado Obtenido
	Aparece un error descriptivo en color rojo en cada uno de los campos con datos erróneos del formulario

Caso de Uso 3.2. Modificar contrato	
Prueba	**Resultado Esperado**
Introducir los nuevos datos del contrato correctamente	La información del contrato es actualizada y se muestra correctamente en su ficha
	Resultado Obtenido
	La base de datos contiene la información actualizada del contrato modificado
Prueba	**Resultado Esperado**
Modificar incorrectamente el formulario	Se muestran los errores de validación encontrados en cada campo del formulario
	Resultado Obtenido
	Aparece un error descriptivo en color rojo en cada uno de los campos con datos erróneos del formulario

Caso de Uso 3.3. Eliminar contrato	
Prueba	**Resultado Esperado**
Eliminar el contrato y confirmar la acción	El contrato es dado de baja en el sistema y ya no aparece en el listado de contratos
	Resultado Obtenido
	El contrato se marca con el *flag "Eliminado"* en la base de datos y ya no aparece en el listado de la aplicación
Prueba	**Resultado Esperado**
Eliminar el contrato sin confirmar la acción	La acción se cancela y se retorna a la vista anterior
	Resultado Obtenido
	Se retorna a la vista anterior de la interfaz sin efectuar cambios

Casos de Uso 3.4. Gestionar servicios del contrato

Caso de Uso 3.4.1. Incluir servicio	
Prueba	**Resultado Esperado**
Completar correctamente todos los campos del formulario	El servicio es asociado al contrato y se muestra en su ficha
	Resultado Obtenido
	La base de datos contiene la información del servicio asociado al contrato y éste aparece correctamente en la ficha del contrato
Prueba	**Resultado Esperado**
Completar incorrectamente el formulario	Se muestran los errores de validación encontrados en cada campo del formulario
	Resultado Obtenido
	Aparece un error descriptivo en color rojo en cada uno de los campos con datos erróneos del formulario

Caso de Uso 3.4.2. Modificar servicio asociado	
Prueba	**Resultado Esperado**
Introducir los nuevos datos del servicio asociado correctamente	La información del servicio contratado es actualizada y se muestra correctamente en la ficha del contrato
	Resultado Obtenido
	La base de datos contiene la información actualizada del servicio asociado y ésta aparece correctamente en la ficha del contrato
Prueba	**Resultado Esperado**
Modificar incorrectamente el formulario	Se muestran los errores de validación encontrados en cada campo del formulario
	Resultado Obtenido
	Aparece un error descriptivo en color rojo en cada uno de los campos con datos erróneos del formulario

Caso de Uso 3.4.3. Desvincular servicio	
Prueba	**Resultado Esperado**
Eliminar el servicio contratado y confirmar la acción	El servicio se desvincula del contrato y no aparece en su ficha
	Resultado Obtenido
	El servicio asociado al contrato se marca con el *flag "Eliminado"* en la base de datos y ya no aparece en la ficha del contrato
Prueba	**Resultado Esperado**
Eliminar el servicio del contrato sin confirmar la acción	La acción se cancela y se retorna a la vista anterior
	Resultado Obtenido
	Se retorna a la vista anterior de la interfaz sin efectuar cambios

Caso de Uso 3.4.4. Listar servicios asociados	
Prueba	**Resultado Esperado**
Consultar el detalle de un	Se muestra, además de los datos básicos del contrato, un

contrato con servicios asociados	listado con los servicios asociados al mismo
	Resultado Obtenido
	Aparece el listado de servicios asociados al contrato en la ficha del mismo
Prueba	**Resultado Esperado**
Consultar el detalle de un contrato sin servicios asociados	Se muestran únicamente los datos básicos del contrato
	Resultado Obtenido
	No se muestra ningún dato sobre los servicios asociados al contrato en su ficha

Caso de Uso 3.5. Consultar listado de contratos	
Prueba	**Resultado Esperado**
Acceder al módulo de gestión de contratos	Se muestra el listado de contratos suscritos por clientes que han sido almacenados en el sistema
	Resultado Obtenido
	Se muestra un listado con todos los contratos dados de alta en el sistema

Caso de Uso 3.5.1. Filtrar registros de contratos	
Prueba	**Resultado Esperado**
Introducir los parámetros de búsqueda deseados en el filtro	Se muestran únicamente los contratos cuyos datos contienen los parámetros especificados
	Resultado Obtenido
	Se muestran solamente los contratos cuyos datos contienen registros coincidentes con los parámetros especificados en los filtros

Caso de Uso 3.5.2. Ordenar registros de contratos	
Prueba	**Resultado Esperado**
Seleccionar la columna por la que se desea ordenar	Los contratos aparecen ordenados de acuerdo al campo seleccionado
	Resultado Obtenido
	El listado de contratos se ordena ascendente o descendentemente, de acuerdo al campo elegido

Caso de Uso 3.6. Consultar detalle del contrato	
Prueba	**Resultado Esperado**
Seleccionar el contrato deseado del listado	Se muestra la ficha completa del contrato seleccionado, incluyendo sus servicios asociados
	Resultado Obtenido
	Se muestra el detalle del contrato, incluyendo su información básica y un listado de los servicios asociados al mismo

8.1.4 Resultados de las Pruebas del Módulo de Gestión de Servicios

Caso de Uso 4.1. Crear servicio	
Prueba	**Resultado Esperado**
Completar todos los datos del formulario correctamente	El servicio es almacenado en el sistema y aparece en el catálogo de servicios
	Resultado Obtenido
	La base de datos contiene el nuevo servicio y sus datos son almacenados correctamente
Prueba	**Resultado Esperado**
Completar incorrectamente el formulario	Se muestran los errores de validación encontrados en cada campo del formulario
	Resultado Obtenido
	Aparece un error descriptivo en color rojo en cada uno de los campos con datos erróneos del formulario

Caso de Uso 4.2. Modificar servicio	
Prueba	**Resultado Esperado**
Introducir los nuevos datos del servicio correctamente	La información del servicio es actualizada y se muestra correctamente en el catálogo de servicios
	Resultado Obtenido
	La base de datos contiene la información actualizada del servicio modificado
Prueba	**Resultado Esperado**
Modificar incorrectamente el formulario	Se muestran los errores de validación encontrados en cada campo del formulario
	Resultado Obtenido
	Aparece un error descriptivo en color rojo en cada uno de los campos con datos erróneos del formulario

Caso de Uso 4.3. Eliminar servicio	
Prueba	**Resultado Esperado**
Eliminar el servicio y confirmar la acción	El servicio es dado de baja en el sistema y ya no aparece en el catálogo de servicios
	Resultado Obtenido
	El servicio se marca con el *flag "Eliminado"* en la base de datos y ya no aparece en el catálogo de la aplicación
Prueba	**Resultado Esperado**
Eliminar el servicio sin confirmar la acción	La acción se cancela y se retorna a la vista anterior
	Resultado Obtenido
	Se retorna a la vista anterior de la interfaz sin efectuar cambios

Casos de Uso 4.4. Gestionar tarifas del servicio

Caso de Uso 4.4.1. Incluir tarifa	
Prueba	**Resultado Esperado**
Completar correctamente todos los campos del formulario	La tarifa es asociada al servicio y se muestra en el catálogo
	Resultado Obtenido
	La base de datos contiene la información de la nueva tarifa y ésta se muestra en el catálogo, asociada al servicio correspondiente
Prueba	**Resultado Esperado**
Completar incorrectamente el formulario	Se muestran los errores de validación encontrados en cada campo del formulario
	Resultado Obtenido
	Aparece un error descriptivo en color rojo en cada uno de los campos con datos erróneos del formulario

Caso de Uso 4.4.2. Modificar tarifa	
Prueba	**Resultado Esperado**
Introducir los nuevos datos de la tarifa del servicio correctamente	La información de la tarifa es actualizada en el servicio y se muestra correctamente en el catálogo
	Resultado Obtenido
	La base de datos contiene la información actualizada de la tarifa del servicio y ésta aparece correctamente en el catálogo
Prueba	**Resultado Esperado**
Modificar incorrectamente el formulario	Se muestran los errores de validación encontrados en cada campo del formulario
	Resultado Obtenido
	Aparece un error descriptivo en color rojo en cada uno de los campos con datos erróneos del formulario

Caso de Uso 4.4.3. Eliminar tarifa	
Prueba	**Resultado Esperado**
Eliminar una tarifa asociada a un servicio y confirmar la acción	La tarifa se desvincula del servicio y no aparece en el catálogo
	Resultado Obtenido
	La tarifa se marca con el *flag "Eliminado"* en la base de datos y ya no aparece vinculada al servicio en el catálogo
Prueba	**Resultado Esperado**
Eliminar una tarifa asociada a un servicio sin confirmar la acción	La acción se cancela y se retorna a la vista anterior
	Resultado Obtenido
	Se retorna a la vista anterior de la interfaz sin efectuar cambios

Caso de Uso 4.4.4. Listar tarifas	
Prueba	**Resultado Esperado**
Consultar el detalle de un servicio con tarifas asociadas	Se muestra, además de la descripción del servicio, un listado con las tarifas aplicables al mismo

	Resultado Obtenido
	Las tarifas asociadas al servicio aparecen junto con la descripción del mismo en el catálogo
Prueba	**Resultado Esperado**
Consultar el detalle de un servicio sin tarifas asociadas	Se muestra únicamente la descripción del servicio
	Resultado Obtenido
	Se muestra la descripción del servicio

Caso de Uso 4.4.4.1. Filtrar registros de tarifas	
Prueba	**Resultado Esperado**
Introducir los parámetros de búsqueda deseados en el filtro	Se muestran únicamente las tarifas cuyos datos contienen los parámetros especificados
	Resultado Obtenido
	Se muestran solamente las tarifas cuyos datos contienen registros coincidentes con los parámetros especificados en los filtros

Caso de Uso 4.4.4.2. Ordenar registros de tarifas	
Prueba	**Resultado Esperado**
Seleccionar la columna por la que se desea ordenar	Las tarifas del servicio en cuestión aparecen ordenadas de acuerdo al campo seleccionado
	Resultado Obtenido
	El listado de tarifas asociadas al servicio se ordena ascendente o descendentemente, de acuerdo al campo elegido

Caso de Uso 4.5. Consultar listado de servicios	
Prueba	**Resultado Esperado**
Acceder al módulo de gestión de servicios	Se muestra el catálogo de servicios ofrecidos por la empresa que han sido incluidos en el sistema
	Resultado Obtenido
	Se muestra, en forma de lista, el catálogo de los servicios dados de alta en el sistema, incluyendo las tarifas asociadas a cada uno

8.1.5 Resultados de las Pruebas del Módulo de Gestión de Incidencias

Caso de Uso 5.1. Crear incidencia	
Prueba	**Resultado Esperado**
Completar todos los datos del formulario correctamente	La incidencia es almacenada en el sistema y aparece en el listado de incidencias
	Resultado Obtenido
	La base de datos contiene la nueva incidencia con los datos proporcionados
Prueba	**Resultado Esperado**
Completar incorrectamente el formulario	Se muestran los errores de validación encontrados en cada campo del formulario
	Resultado Obtenido
	Aparece un error descriptivo en color rojo en cada uno de los campos con datos erróneos del formulario

Caso de Uso 5.2. Modificar incidencia	
Prueba	**Resultado Esperado**
Introducir los nuevos datos de la incidencia correctamente	La información de la incidencia es actualizada y se muestra correctamente en su ficha
	Resultado Obtenido
	La base de datos contiene la información actualizada de la incidencia modificada
Prueba	**Resultado Esperado**
Modificar incorrectamente el formulario	Se muestran los errores de validación encontrados en cada campo del formulario
	Resultado Obtenido
	Aparece un error descriptivo en color rojo en cada uno de los campos con datos erróneos del formulario

Caso de Uso 5.3. Eliminar incidencia	
Prueba	**Resultado Esperado**
Eliminar la incidencia y confirmar la acción	La incidencia es dada de baja en el sistema y ya no aparece en el listado de incidencias
	Resultado Obtenido
	La incidencia se marca con el *flag "Eliminado"* en la base de datos y ya no aparece en el listado de la aplicación
Prueba	**Resultado Esperado**
Eliminar la incidencia sin confirmar la acción	La acción se cancela y se retorna a la vista anterior
	Resultado Obtenido
	Se retorna a la vista anterior de la interfaz sin efectuar cambios

Caso de Uso 5.4. Consultar detalle de incidencia	
Prueba	**Resultado Esperado**
Seleccionar la incidencia	Se muestra la ficha completa de la incidencia seleccionada,

deseada del listado	incluyendo sus elementos asociados (intervenciones, adjuntos y relaciones)
	Resultado Obtenido
	Se muestra el detalle de la incidencia, incluyendo su información básica y los listados de intervenciones, adjuntos y relaciones asociadas a la misma

Caso de Uso 5.5. Clonar incidencia	
Prueba	**Resultado Esperado**
Seleccionar la opción Clonar en la vista detalle de una incidencia	Aparece una nueva incidencia en el listado del sistema con los mismos datos básicos que la original
	Resultado Obtenido
	La base de datos contiene una nueva incidencia, cuyos datos básicos son idénticos a los de la original, a excepción del identificador único

Caso de Uso 5.6. Generar informe imprimible	
Prueba	**Resultado Esperado**
Seleccionar la opción *Generar Informe* en la vista detalle de una incidencia y especificar como destino un fichero PDF	Se genera un documento PDF con toda la información de la incidencia en la ruta indicada
	Resultado Obtenido
	El documento PDF generado aparece en el sistema de ficheros del equipo del usuario y contiene toda la información de la incidencia
Prueba	**Resultado Esperado**
Seleccionar la opción *Generar Informe* en la vista detalle de una incidencia y especificar como destino una impresora conectada al equipo del usuario	Se imprime el informe completo de la incidencia en la impresora seleccionada
	Resultado Obtenido
	El documento impreso contiene toda la información de la incidencia seleccionada

Caso de Uso 5.7. Listar incidencias	
Prueba	**Resultado Esperado**
Acceder al módulo de gestión de incidencias	Se muestra el listado de incidencias almacenadas en el sistema, ordenadas del siguiente modo: • Primero, aquellas que han sido asignadas al usuario en sesión • Segundo, aquellas con la prioridad de resolución más alta
	Resultado Obtenido
	Aparecen, en primer lugar, aquellas incidencias que han sido asignadas al usuario en sesión. En caso de haber más de una, se muestran primero las más prioritarias. Si el usuario en sesión no tiene ninguna incidencia asignada, éstas aparecen simplemente ordenadas por prioridad

Caso de Uso 5.7.1. Filtrar registros de incidencias	
Prueba	**Resultado Esperado**
Introducir los parámetros de búsqueda deseados en el filtro	Se muestran únicamente las incidencias cuyos datos contienen los parámetros especificados
	Resultado Obtenido
	Se muestran solamente las incidencias cuyos datos contienen registros coincidentes con los parámetros especificados en los filtros

Caso de Uso 5.7.2. Ordenar registros de incidencias	
Prueba	**Resultado Esperado**
Seleccionar la columna por la que se desea ordenar	Las incidencias aparecen ordenadas de acuerdo al campo seleccionado
	Resultado Obtenido
	El listado de incidencias se ordena ascendente o descendentemente, de acuerdo al campo elegido

Casos de Uso 5.8. Gestionar intervenciones de la incidencia

Caso de Uso 5.8.1. Incluir intervención	
Prueba	**Resultado Esperado**
Completar correctamente todos los datos del formulario	La intervención es anotada en la incidencia y se muestra en su ficha
	Resultado Obtenido
	La base de datos contiene la información de la intervención y ésta aparece en la ficha detalle de la incidencia
Prueba	**Resultado Esperado**
Completar incorrectamente el formulario	Se muestran los errores de validación encontrados en cada campo del formulario
	Resultado Obtenido
	Aparece un error descriptivo en color rojo en cada uno de los campos con datos erróneos del formulario

Caso de Uso 5.8.2. Modificar intervención	
Prueba	**Resultado Esperado**
Introducir los nuevos datos de la intervención correctamente	La información de la intervención es actualizada y se muestra correctamente en la ficha de la incidencia
	Resultado Obtenido
	La base de datos contiene la información actualizada de la intervención y ésta aparece correctamente reflejada en la ficha detalle de la incidencia
Prueba	**Resultado Esperado**
Modificar incorrectamente el formulario	Se muestran los errores de validación encontrados en cada campo del formulario
	Resultado Obtenido

	Aparece un error descriptivo en color rojo en cada uno de los campos con datos erróneos del formulario

Caso de Uso 5.8.3. Eliminar intervención	
Prueba	**Resultado Esperado**
Eliminar una intervención anotada en una incidencia y confirmar la acción	La intervención se elimina de la incidencia y no aparece en su ficha
	Resultado Obtenido
	La intervención asociada a la incidencia se marca con el *flag "Eliminado"* en la base de datos y ya no aparece en la ficha de la incidencia
Prueba	**Resultado Esperado**
Eliminar una intervención anotada en una incidencia sin confirmar la acción	La acción se cancela y se retorna a la vista anterior
	Resultado Obtenido
	Se retorna a la vista anterior de la interfaz sin efectuar cambios

Caso de Uso 5.8.4. Listar intervenciones	
Prueba	**Resultado Esperado**
Consultar el detalle de una incidencia con intervenciones asociadas	Se muestra, además de los datos básicos de la incidencia, un listado con las intervenciones realizadas en la incidencia
	Resultado Obtenido
	Se muestra, en el detalle de la incidencia, un listado con las intervenciones relacionadas con la incidencia existentes en la base de datos
Prueba	**Resultado Esperado**
Consultar el detalle de una incidencia con intervenciones asociadas	Se muestran únicamente los datos básicos de la incidencia
	Resultado Obtenido
	Se muestran únicamente los datos básicos de la incidencia

Casos de Uso 5.9. Gestionar adjuntos a la incidencia

Caso de Uso 5.9.1. Adjuntar fichero	
Prueba	**Resultado Esperado**
Completar correctamente todos los campos del formulario y seleccionar un fichero con formato y tamaño admitidos por el servidor	El fichero se adjunta a la incidencia y se muestra una referencia al mismo en la ficha de la incidencia
	Resultado Obtenido
	El fichero es vinculado a la incidencia y se muestra una referencia al mismo en el detalle de la incidencia
Prueba	**Resultado Esperado**
Completar incorrectamente el formulario y/o seleccionar un fichero con formato y/o tamaño no admitidos por el servidor	Se muestran los errores de validación encontrados en cada campo del formulario y/o en el control de subida del fichero
	Resultado Obtenido
	Aparece un error descriptivo en color rojo en cada uno de los campos con datos erróneos del formulario

Caso de Uso 5.9.2. Eliminar adjunto	
Prueba	**Resultado Esperado**
Eliminar el fichero adjunto de la incidencia y confirmar la acción	El fichero se desvincula de la incidencia y no aparece en su ficha
	Resultado Obtenido
	El fichero asociado a la incidencia se marca con el *flag "Eliminado"* en la base de datos y ya no aparece en la ficha de la incidencia
Prueba	**Resultado Esperado**
Eliminar el fichero adjunto de la incidencia sin confirmar la acción	La acción se cancela y se retorna a la vista anterior
	Resultado Obtenido
	Se retorna a la vista anterior de la interfaz sin efectuar cambios

Caso de Uso 5.9.3. Listar adjuntos	
Prueba	**Resultado Esperado**
Consultar el detalle de una incidencia con ficheros adjuntos	Se muestra, además de los datos básicos de la incidencia, un listado con los ficheros adjuntos a la misma
	Resultado Obtenido
	Se muestra, además de los datos básicos de la incidencia, un listado con los ficheros asociados la incidencia
Prueba	**Resultado Esperado**
Consultar el detalle de una incidencia sin ficheros adjuntos	Se muestran únicamente los datos básicos de la incidencia
	Resultado Obtenido
	Se muestran únicamente los datos básicos de la incidencia

Casos de Uso 5.10. Gestionar relaciones de la incidencia

Caso de Uso 5.10.1. Relacionar con otra incidencia	
Prueba	**Resultado Esperado**
Relacionar correctamente una incidencia con otra existente y no relacionada anteriormente	La relación se establece correctamente y se muestra en la ficha de la incidencia
	Resultado Obtenido
	La base de datos contiene la información de la relación asociada a la incidencia y ésta aparece correctamente en la ficha de la incidencia
Prueba	**Resultado Esperado**
Relacionar una incidencia cuando no existe ninguna otra en el sistema	No es posible establecer ninguna relación entre incidencias
	Resultado Obtenido
	El formulario de creación de relaciones entre incidencias muestra un error descriptivo en color rojo
Prueba	**Resultado Esperado**
Crear una relación idéntica a una ya existente	Se muestra un error indicando que la relación ya se ha establecido anteriormente
	Resultado Obtenido
	El formulario de creación de relaciones entre incidencias muestra un error descriptivo en color rojo
Prueba	**Resultado Esperado**

Completar incorrectamente el formulario	Se muestran los errores de validación encontrados en cada campo del formulario
	Resultado Obtenido
	Aparece un error descriptivo en color rojo en cada uno de los campos con datos erróneos del formulario

Caso de Uso 5.10.2. Eliminar relación	
Prueba	**Resultado Esperado**
Eliminar la relación de la incidencia y confirmar la acción	La relación se elimina de la incidencia y no aparece en su ficha
	Resultado Obtenido
	La relación se marca con el *flag "Eliminado"* en la base de datos y ya no aparece en la ficha de la incidencia
Prueba	**Resultado Esperado**
Eliminar la relación de la incidencia sin confirmar la acción	La acción se cancela y se retorna a la vista anterior
	Resultado Obtenido
	Se retorna a la vista anterior de la interfaz sin efectuar cambios

Caso de Uso 5.10.3. Listar relaciones directas	
Prueba	**Resultado Esperado**
Consultar el detalle de una incidencia con relaciones directas asociadas	Se muestra, además de los datos básicos de la incidencia, un listado con las relaciones directas de la misma
	Resultado Obtenido
	Se muestra, en la vista detalle de la incidencia, un listado con las referencias a aquellas incidencias con las que la visualizada se relaciona directamente
Prueba	**Resultado Esperado**
Consultar el detalle de una incidencia sin relaciones directas asociadas	Se muestran únicamente los datos básicos de la incidencia
	Resultado Obtenido
	Se muestran únicamente los datos básicos de la incidencia

Caso de Uso 5.10.4. Listar relaciones inversas	
Prueba	**Resultado Esperado**
Consultar el detalle de una incidencia con relaciones inversas asociadas	Se muestra, además de los datos básicos de la incidencia, un listado con las relaciones inversas de la misma
	Resultado Obtenido
	Se muestra, en la vista detalle de la incidencia, un listado con las referencias a aquellas incidencias que guardan relación con la visualizada
Prueba	**Resultado Esperado**
Consultar el detalle de una incidencia sin relaciones inversas asociadas	Se muestran únicamente los datos básicos de la incidencia
	Resultado Obtenido
	Se muestran únicamente los datos básicos de la incidencia

8.2 Pruebas de Usabilidad y Accesibilidad

8.2.1 Resultados de las Pruebas de Usabilidad

A continuación se muestran los resultados obtenidos de la aplicación de la Guía de Evaluación Heurística de Sitios Web. Algunos resultados cuentan con una explicación extendida mostrada posteriormente.

Criterios	¿Cumplido?
Generales	
¿Cuáles son los objetivos del sitio web? ¿Son concretos y bien definidos? ¿Los contenidos y servicios que ofrece se corresponden con esos objetivos?	**SI**
¿Tiene una URL correcta, clara y fácil de recordar? ¿Y las URL de sus páginas internas? ¿Son claras y permanentes?	**SI**
¿Muestra de forma precisa y completa qué contenidos o servicios ofrece realmente el sitio web? El diseño de la página de inicio debe ser diferente al resto de páginas y cumplir la función de 'escaparate' del sitio.	**Parcialmente (1)**
¿La estructura general del sitio web está orientada al usuario? Los sitios web deben estructurarse pensando en el usuario, sus objetivos y necesidades. La estructura interna de la empresa u organización, cómo funciona o se organiza no interesan al usuario.	**SI**
¿El *look & feel* general se corresponde con los objetivos, características, contenidos y servicios del sitio web? Ciertas combinaciones de colores ofrecen imágenes más o menos formales, serias o profesionales.	**SI**
¿Es coherente el diseño general del sitio web? Se debe mantener una coherencia y uniformidad en las estructuras y colores de todas las páginas. Esto sirve para que el usuario no se desoriente en su navegación.	**SI**
¿Es reconocible el diseño general del sitio web? Cuánto más se parezca el sitio web al resto de sitios web, más fácil será de usar.	**SI**
¿El sitio web se actualiza periódicamente? ¿Indica cuándo se actualiza? Las fechas que se muestren en la página deben corresponderse con actualizaciones, noticias, eventos...no con la fecha del sistema del usuario.	**N/A (2)**
Identidad e Información	
¿Se muestra claramente la identidad de la empresa-sitio a través de todas las páginas?	**SI**
El Logotipo, **¿es significativo, identificable y suficientemente visible?**	**SI**
El eslogan o *tagline*, **¿expresa realmente qué es la empresa y qué servicios ofrece?**	**SI**
¿Se ofrece algún enlace con información sobre la empresa, sitio web, 'webmaster',...?	**SI**
¿Se proporciona mecanismos para ponerse en contacto con la empresa? (email, teléfono, dirección postal, fax...)	**N/A (3)**
¿Se proporciona información sobre la protección de datos de carácter personal de los clientes o los derechos de autor de los contenidos del sitio web?	**N/A (3)**

En artículos, noticias, informes... **¿Se muestra claramente información sobre el autor, fuentes y fechas de creación y revisión del documento?**	**N/A (3)**
Lenguaje y Redacción	
¿El sitio web habla el mismo lenguaje que sus usuarios? Se debe evitar usar un lenguaje corporativista. Así mismo, hay que prestarle especial atención al idioma, y ofrecer versiones del sitio en diferentes idiomas cuando sea necesario.	**SI**
¿Emplea un lenguaje claro y conciso?	**SI**
¿Es amigable, familiar y cercano? Es decir, lo contrario a utilizar un lenguaje constantemente imperativo, mensajes crípticos, o tratar con "desprecio" al usuario.	**SI**
Rotulado	
Los rótulos, **¿son significativos?** Ejemplo: evitar rótulos del tipo "haga clic aquí".	**SI**
¿Usa rótulos estándar? Siempre que exista un "estándar" comúnmente aceptado para el caso concreto, como "Mapa del Sitio" o "Acerca de...".	**SI**
¿Usa un único sistema de organización, bien definido y claro? No se deben mezclar diferentes. Los sistemas de organización son: alfabético, geográfico, cronológico, temático, orientado a tareas, orientado al público y orientado a metáforas.	**SI**
¿Utiliza un sistema de rotulado controlado y preciso? Por ejemplo, si un enlace tiene el rótulo "Quiénes somos", no puede dirigir a una página cuyo encabezamiento sea "Acerca de".	**SI**
El título de las páginas, **¿Es correcto? ¿Ha sido planificado?** Relacionado con la capacidad para poder buscar y encontrar el sitio *web*.	**SI**
Estructura y Navegación	
La estructura de organización y navegación, **¿Es la más adecuada?** Hay varios tipos de estructuras: jerárquicas, hipertextual, facetada,...	**SI**
En el caso de estructura jerárquica, **¿Mantiene un equilibrio entre Profundidad y Anchura?**	**SI**
En el caso de ser puramente hipertextual, **¿Están todos los clúster de nodos comunicados?**	**SI**
¿Los enlaces son fácilmente reconocibles como tales? ¿Su caracterización indica su estado (visitados, activos,...)? Los enlaces no sólo deben reconocerse como tales, sino que su caracterización debe indicar su estado, y ser reconocidos como una unidad.	**SI**
En menús de navegación, **¿Se ha controlado el número de elementos y de términos por elemento para no producir sobrecarga memorística?** No se deben superar los 7±2 elementos, ni los 2 o, como mucho, 3 términos por elemento.	**SI**
¿Es predecible la respuesta del sistema antes de hacer clic sobre el enlace? Relacionado con el nivel de significación del rótulo del enlace, aunque también con: el uso de globos de texto, información contextual, la barra de estado del navegador,...	**SI**

¿Se ha controlado que no haya enlaces que no llevan a ningún sitio? Enlaces que no llevan a ningún sitio: Los enlaces rotos, y los que enlazan con la misma página que se está visualizando (por ejemplo enlaces a la "home" desde la misma página de inicio).	**SI**
¿Existen elementos de navegación que orienten al usuario acerca de dónde está y cómo deshacer su navegación? Como *breadcrumbs*, enlaces a la página de inicio, etc. Recuerde que el logo también es recomendable que enlace con la página de inicio.	**SI**
Las imágenes enlace, **¿se reconocen como clicables? ¿Incluyen un atributo 'title' describiendo la página de destino?** En este sentido, también hay que cuidar que no haya imágenes que parezcan enlaces y en realidad no lo sean.	**SI**
¿Se ha evitado la redundancia de enlaces?	**NO (4)**
¿Se ha controlado que no haya páginas "huérfanas"? Páginas huérfanas: que aún siendo enlazadas desde otras páginas, éstas no enlacen con ninguna.	**SI**
Layout de la Página	
¿Se aprovechan las zonas de alta jerarquía informativa de la página para contenidos de mayor relevancia? (como por ejemplo la zona central)	**SI**
¿Se ha evitado la sobrecarga informativa? Esto se consigue haciendo un uso correcto de colores, efectos tipográficos y agrupaciones para discriminar información. Los grupos diferentes de objetos informativos de una página deben ser 7±2.	**SI**
¿Es una interfaz limpia, sin ruido visual?	**SI**
¿Existen zonas en "blanco" entre los objetos informativos de la página para poder descansar la vista?	**Parcialmente (5)**
¿Se hace un uso correcto del espacio visual de la página? Es decir, que no se desaproveche demasiado espacio con elementos de decoración, o grandes zonas en "blanco", y que no se adjudique demasiado espacio a elementos de menor importancia.	**SI**
¿Se utiliza correctamente la jerarquía visual para expresar las relaciones del tipo "parte de" entre los elementos de la página? (La jerarquía visual se utiliza para orientar al usuario)	**SI**
¿Se ha controlado la longitud de página? Se debe evitar en la medida de lo posible el *scrolling*. Si la página es muy extensa, se debe fraccionar.	**SI**
Búsqueda (si es necesario, por la extensión del sitio, incorporar un buscador interno)	
¿Se encuentra fácilmente accesible? Es decir: directamente desde la home, y a ser posible desde todas las páginas del sitio, y colocado en la zona superior de la página.	**SI**
¿Es fácilmente reconocible como tal?	**SI**
¿Permite la búsqueda avanzada? (siempre y cuando, por las características del sitio web, fuera de utilidad que la ofreciera)	**SI**
¿Muestra los resultados de la búsqueda de forma comprensible para el usuario?	**SI**
¿La caja de texto es lo suficientemente ancha?	**SI**
¿Asiste al usuario en caso de no poder ofrecer resultados para una consultada dada?	**SI**

Elementos Multimedia	
¿Las fotografías están bien recortadas? ¿Son comprensibles? ¿Se ha cuidado su resolución?	**Parcialmente (6)**
¿Las metáforas visuales son reconocibles y comprensibles por cualquier usuario? (prestar especial atención a usuarios de otros países y culturas)	**SI**
¿El uso de imágenes o animaciones proporciona algún tipo de valor añadido?	**SI**
¿Se ha evitado el uso de animaciones cíclicas?	**SI**
Accesibilidad (en detalle en la sección posterior)	
¿El tamaño de fuente se ha definido de forma relativa, o por lo menos, la fuente es lo suficientemente grande como para no dificultar la legibilidad del texto?	**SI**
¿El tipo de fuente, efectos tipográficos, ancho de línea y alineación empleadas facilitan la lectura?	**SI**
¿Existe un alto contraste entre el color de fuente y el fondo?	**NO (7)**
¿Incluyen las imágenes atributos 'alt' que describan su contenido?	**SI**
¿Es compatible el sitio web con los diferentes navegadores? ¿Se visualiza correctamente con diferentes resoluciones de pantalla? Se debe prestar atención a: *JavaScript*, *CSS*, tablas, fuentes...	**SI**
¿Puede el usuario disfrutar de todos los contenidos del sitio web sin necesidad de tener que descargar e instalar *plugins* adicionales?	**SI**
¿Se ha controlado el peso de la página? Se deben optimizar las imágenes, controlar el tamaño del código *JavaScript*...	**SI**
¿Se puede imprimir la página sin problemas? Leer en pantalla es molesto, por lo que muchos usuarios preferirán imprimir las páginas para leerlas. Se debe asegurar que se puede imprimir la página (no salen partes cortadas), y que el resultado es legible.	**SI**
Control y Retroalimentación	
¿Tiene el usuario todo el control sobre el interfaz? Se debe evitar el uso de ventanas pop-up, ventanas que se abren a pantalla completa, banners intrusivos...	**Parcialmente (8)**
¿Se informa constantemente al usuario acerca de lo que está pasando? Si el usuario tiene que esperar hasta que se termine una operación, se debe mostrar un mensaje indicándoselo y que debe esperar, con el tiempo de espera estimado o una barra de progreso.	**SI**
¿Se informa al usuario de lo que ha pasado? Por ejemplo, cuando un usuario valora un artículo o responde a una encuesta, se le debe informar de que su voto ha sido procesado correctamente.	**SI**
Cuando se produce un error, **¿se informa de forma clara y no alarmista al usuario de lo ocurrido y de cómo solucionar el problema?** Siempre es mejor intentar evitar que se produzcan errores a tener que informar al usuario del error.	**SI**

¿Posee el usuario libertad para actuar? NO restringir la libertad del usuario: Uso de animaciones que no pueden ser "saltadas", páginas en las que desaparecen los botones de navegación, no impida al usuario poder usar el botón derecho de su ratón...	**SI**
¿Se ha controlado el tiempo de respuesta? Esto tiene que ver con el peso de cada página (accesibilidad) y tiene relación con el tiempo que tarda el servidor en finalizar una tarea y responder. El tiempo máximo que esperará un usuario son 10 segundos	**SI**

Anotaciones a los resultados:

1. La página de inicio difiere con el resto en el contenido central, donde se muestra el formulario de acceso a la aplicación. No se precisan datos sobre los servicios que ofrece el sistema, puesto que se trata de un sitio privado y sus usuarios lo conocen.
2. El contenido actualizable es el generado por los usuarios como resultado del uso de la aplicación, por lo que el requisito no se considera aplicable.
3. Estos elementos se consideran prescindibles por el tipo de aplicación.
4. En algunas ocasiones, los requisitos funcionales del sistema han requerido el uso de enlaces redundantes, puesto que debe ser posible acceder, por ejemplo, al listado de incidencias desde la ficha de un cliente, además de desde el menú principal.
5. En determinadas vistas, como los listados de elementos, la cantidad de información a presentar es elevada, por lo que las zonas en blanco son reducidas. No obstante, en las interfaces menos saturadas, como los formularios, se proporciona una distribución de elementos que favorece la existencia de zonas para descansar la vista del usuario.
6. Por decisión del equipo de diseño, los logotipos de los clientes no se escalan, por lo que es posible que estas imágenes se vean descuadradas en la interfaz.
7. Siguiendo con el patrón de colores establecido por la imagen corporativa de la empresa, el contraste entre el color de la fuente y el fondo no es el adecuado.
8. Se estableció como requisito que, en el caso de acciones de modificación de entidades o inclusión de elementos a las mismas, el formulario habilitado a tal efecto se mostraría en una ventana emergente.

8.2.2 Resultados de las Pruebas de Accesibilidad

A continuación se muestran los resultados obtenidos de la aplicación de los criterios de conformidad de nivel A y AA de las WCAG 2.0 del W3C. Se muestra una breve descripción de aquellos resultados que precisan de ella.

Criterios del Nivel de Conformidad A	Sí	No	N/A
1.1.1 Contenido no textual: Todo contenido no textual que se presenta al usuario tiene una alternativa textual que cumple el mismo propósito.	**X**		
1.2.1 Sólo audio y sólo vídeo grabado. Para contenido sólo audio grabado y contenido sólo vídeo grabado, se cumple lo siguiente, excepto cuando el audio o el vídeo es un contenido multimedia alternativo al texto y está claramente identificado como tal.			**X**
1.2.2 Subtítulos (grabados). Se proporcionan subtítulos para el contenido de audio grabado dentro de contenido multimedia sincronizado, excepto cuando la presentación es un contenido multimedia alternativo al texto y está claramente identificado como tal.			**X**
1.2.3 Audiodescripción o Medio Alternativo (grabado). Se proporciona una alternativa para los medios tempodependientes o una audiodescripción para el contenido de vídeo grabado en los multimedia sincronizados, excepto cuando ese contenido es un contenido multimedia alternativo al texto y está claramente identificado como tal.			**X**
1.3.1 Información y relaciones. La información, estructura y relaciones comunicadas a través de la presentación pueden ser determinadas por software o están disponibles como texto	**X**		
1.3.2 Secuencia significativa. Cuando la secuencia en que se presenta el contenido afecta a su significado, se puede determinar por software la secuencia correcta de lectura.			**X**
1.3.3 Características sensoriales. Las instrucciones proporcionadas para entender y operar el contenido no dependen exclusivamente en las características sensoriales de los componentes como su forma, tamaño, ubicación visual, orientación o sonido.	**X**		
1.4.1 Uso del color. El color no se usa como único medio visual para transmitir la información, indicar una acción, solicitar una respuesta o distinguir un elemento visual.	**X**		
1.4.2 Control del audio. Si el audio de una página web suena automáticamente durante más de 3 segundos, se proporciona ya sea un mecanismo para pausar o detener el audio, o un mecanismo para controlar el volumen del sonido que es independiente del nivel de volumen global del sistema.			**X**
2.1.1 Teclado. Toda la funcionalidad del contenido es operable a través de una interfaz de teclado sin que se requiera una determinada velocidad para cada pulsación individual de las teclas, excepto cuando la función interna requiere de una entrada que depende del trayecto de los movimientos del usuario y no sólo de los puntos inicial y final.	**X**		
2.1.2 Sin trampas para el foco del teclado: Si es posible mover el foco a un componente de la página usando una interfaz de teclado, entonces el foco se puede quitar de ese componente usando sólo la interfaz de teclado y, si se requiere algo más que las teclas de dirección o de tabulación, se informa al usuario el método apropiado para mover el foco.	**X**		

2.2.1 Tiempo ajustable.			**X**
2.2.2 Poner en pausa, detener, ocultar.			**X**
2.3.1 Umbral de tres destellos o menos: Las páginas web no contienen nada que destelle más de tres veces en un segundo, o el destello está por debajo del umbral de destello general y de destello rojo.	**X**		
2.4.1 Evitar bloques: Existe un mecanismo para evitar los bloques de contenido que se repiten en múltiples páginas web.	**X**		
2.4.2 Título de la página. Las páginas web tienen títulos que describen su temática o propósito.	**X**		
2.4.3 Orden del foco. Si se puede navegar secuencialmente por una página web y la secuencia de navegación afecta su significado o su operación, los componentes que pueden recibir el foco lo hacen en un orden que preserva su significado y operabilidad.	**X**		
2.4.4 Propósito de los enlaces (en su contexto). El propósito de cada enlace puede ser determinado con sólo el texto del enlace o a través del texto del enlace sumado al contexto del enlace determinado por software, excepto cuando el propósito del enlace resultara ambiguo para los usuarios en general.	**X**		
3.1.1 Idioma de la página. El idioma predeterminado de cada página web puede ser determinado por software.	**X**		
3.2.1 Al recibir el foco: Cuando cualquier componente recibe el foco, no inicia ningún cambio en el contexto.	**X**		
3.2.2 Al recibir entradas: El cambio de estado en cualquier componente de la interfaz de usuario no provoca automáticamente un cambio en el contexto a menos que el usuario haya sido advertido de ese comportamiento antes de usar el componente.	**X**		
3.3.1 Identificación de errores. Si se detecta automáticamente un error en la entrada de datos, el elemento erróneo es identificado y el error se describe al usuario mediante un texto.	**X**		
3.3.2 Etiquetas o instrucciones. Se proporcionan etiquetas o instrucciones cuando el contenido requiere la introducción de datos por parte del usuario.	**X**		
4.1.1 Procesamiento: En los contenidos implementados mediante el uso de lenguajes de marcas, los elementos tienen las etiquetas de apertura y cierre completas; los elementos están anidados de acuerdo a sus especificaciones; los elementos no contienen atributos duplicados y los ID son únicos, excepto cuando las especificaciones permitan estas características.		**X (1)**	
4.1.2 Nombre, función, valor. Para todos los componentes de la interfaz de usuario (incluyendo pero no limitado a: elementos de formulario, enlaces y componentes generados por scripts), el nombre y la función pueden ser determinados por software; los estados, propiedades y valores que pueden ser asignados por el usuario pueden ser especificados por software; y los cambios en estos elementos se encuentran disponibles para su consulta por las aplicaciones de usuario, incluyendo las ayudas técnicas.	**X**		
Criterios del Nivel de Conformidad AA	**Sí**	**No**	**N/A**
1.2.4 Subtítulos (en directo): Se proporcionan subtítulos para todo el contenido de audio en directo de los multimedia sincronizados.			**X**
1.2.5 Audiodescripción (grabado): Se proporciona una audiodescripción para todo el contenido de vídeo grabado dentro de contenido			**X**

multimedia sincronizado.			
1.4.3 Contraste (mínimo)		**X (2)**	
1.4.4 Cambio de tamaño del texto: A excepción de los subtítulos y las imágenes de texto, todo el texto puede ser ajustado sin ayudas técnicas hasta un 200 por ciento sin que se pierdan el contenido o la funcionalidad.	**X**		
1.4.5 Imágenes de texto.	**X**		
2.4.5 Múltiples vías. Se proporciona más de un camino para localizar una página web dentro de un conjunto de páginas web, excepto cuando la página es el resultado, o un paso intermedio, de un proceso.	**X**		
2.4.6 Encabezados y etiquetas. Los encabezados y etiquetas describen el tema o propósito.	**X**		
2.4.7 Visibilidad del foco. Cualquier interfaz de usuario operable por teclado tiene una forma de operar en la cuál el indicador del foco del teclado resulta visible.	**X**		
3.1.2 Idioma de las partes. El idioma de cada pasaje o frase en el contenido puede ser determinado por software, excepto los nombres propios, términos técnicos, palabras en un idioma indeterminado y palabras o frases que se hayan convertido en parte natural del texto que las rodea.	**X**		
3.2.3 Navegación consistente. Los mecanismos de navegación que se repiten en múltiples páginas web dentro de un conjunto de páginas web aparecen siempre en el mismo orden relativo cada vez que se repiten, a menos que el cambio sea provocado por el propio usuario.	**X**		
3.2.4 Identificación consistente. Los componentes que tienen la misma funcionalidad dentro de un conjunto de páginas web son identificados de manera coherente.	**X**		
3.3.3 Sugerencias ante error. Si se detecta automáticamente un error en la entrada de datos y se dispone de sugerencias para hacer la corrección, entonces se presentan las sugerencias al usuario, a menos que esto ponga en riesgo la seguridad o el propósito del contenido.	**X**		
3.3.4 Prevención de errores (Legales, financieros, de datos). Para las páginas web que representan para el usuario compromisos legales o transacciones financieras; que modifican o eliminan datos controlables por el usuario en sistemas de almacenamiento de datos; o que envían las respuestas del usuario a una prueba.	**X**		

Anotaciones a los resultados:

1. Struts2 genera un conjunto propio de etiquetas automáticamente, por lo que el no cumplimiento de este criterio es inherente a la tecnología empleada.
2. Siguiendo con el patrón de colores establecido por la imagen corporativa de la empresa, el contraste entre el color de la fuente y el fondo no es el adecuado.

Como puede observarse en las tablas anteriores, se han intentado seguir en la medida de lo posible las recomendaciones de las WCAG 2.0 para lograr un nivel de conformidad aceptable. Sin embargo, debido a las herramientas utilizadas para el desarrollo y las preferencias del cliente sobre el diseño de las interfaces, no todas estas normas han podido ser cumplidas en su totalidad, aunque los resultados en este aspecto son bastante

satisfactorios, habiendo cumplido con el 90% de los criterios de nivel A y con el 75% en el caso de los de nivel AA.

8.3 Pruebas de Rendimiento

Tal y como se dijo anteriormente, no ha resultado necesario establecer un plan de pruebas de rendimiento para la aplicación. En lo concerniente a este aspecto, se ha comprobado que los tiempos de respuesta y la experiencia de usuario global son satisfactorios, de acuerdo a los criterios de los usuarios reales de la aplicación, quienes han probado las versiones beta del sistema desplegadas en cada hito del desarrollo.

Capítulo 9. Manuales del Sistema

9.1 Manual de Instalación

En esta sección se describe el conjunto de pasos a seguir para configurar todos los componentes del sistema y elementos externos, logrando así su puesta en marcha.

Se da por supuesto que el entorno cuenta con el hardware necesario para alojar los servicios, así como los sistemas operativos de cada máquina correctamente instalados y configurados, incluidos los parámetros de red, tales direcciones IP, cortafuegos, etc.

9.1.1 Sistema de Gestión de Bases de Datos

9.1.1.1 Instalación

En primer lugar se acometerá la instalación y configuración del SGBD empleado para el almacenamiento de datos, SQL Server 2005 de Microsoft. Los requisitos mínimos del sistema, según el fabricante, son los siguientes:

- Sistema operativo compatible; Windows 2000 SP4, Server 2003 SP1, XP SP2 o superiores
- Equipo con procesador Intel Pentium III (o compatible) a 1 GHz
- 512 MB de memoria RAM
- 525 MB de espacio disponible en disco duro

Se recomienda, como paso previo a la instalación, comprobar que el sistema operativo del servidor se encuentra correctamente actualizado, para lo cual puede hacerse uso de la herramienta nativa *Windows Update*.

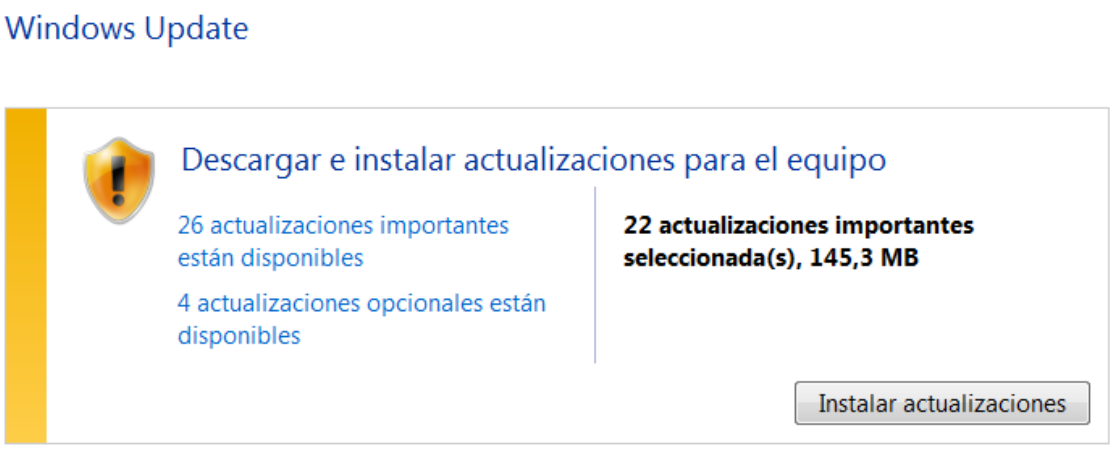


Figura 9.1. Manual de Instalación - Actualizaciones del sistema

El siguiente paso consistirá en ejecutar el instalador, disponible en el directorio *./instalación/sgbd/SQLEXPR_ESN.exe*

Leemos y aceptamos los términos de licencia pulsando *Siguiente.*

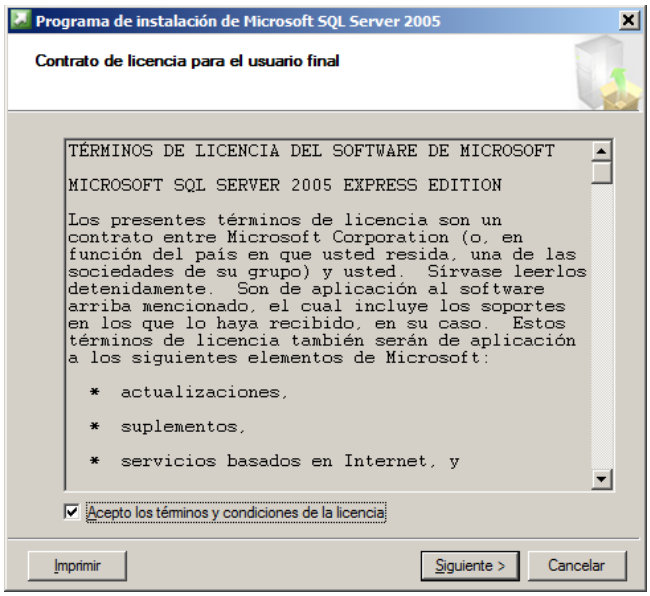


Figura 9.2. Manual de Instalación - Términos de licencia

El instalador se encargará ahora de comprobar las dependencias necesarias e instalar aquellos componentes que fueran necesarios antes de continuar.

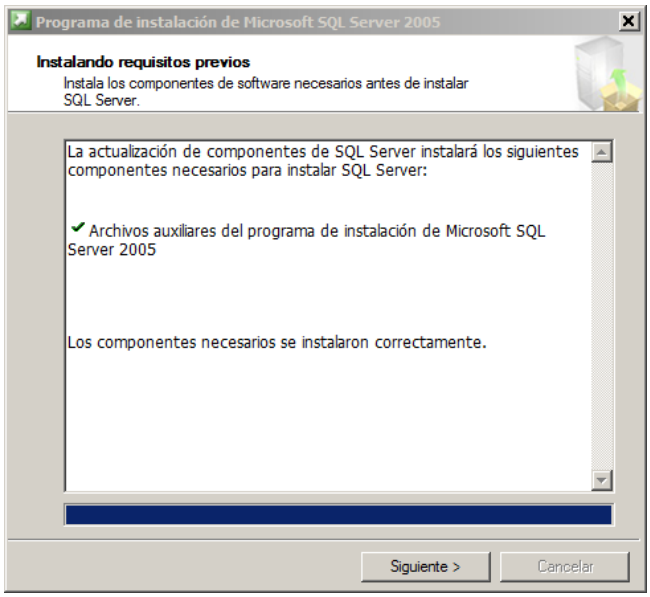


Figura 9.3. Manual de Instalación - Comprobación de dependencias

Será entonces cuando comenzará propiamente la instalación del SGBD. Tras la pantalla de bienvenida del asistente de instalación, en la que pulsaremos *Siguiente*, y la comprobación de requisitos del sistema automatizada, debemos proveer los datos de registro de la aplicación.

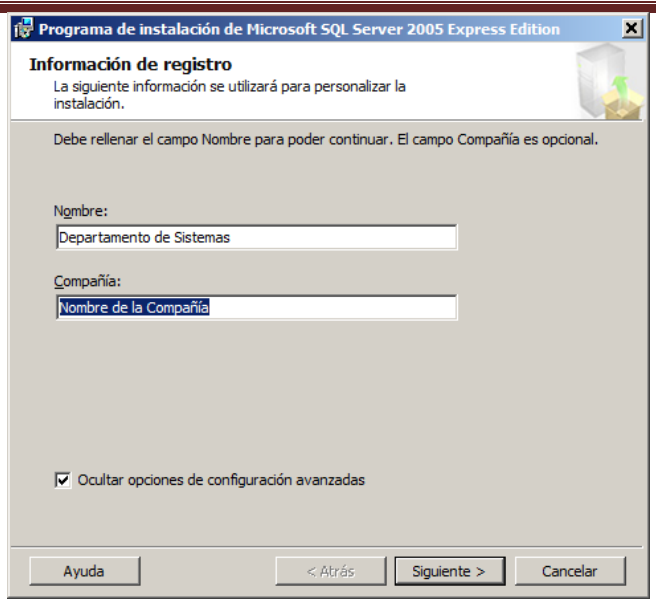


Figura 9.4. Manual de Instalación - Información de registro

Marcamos a continuación qué componentes deseamos instalar. Concretamente, seleccionaremos todos los disponibles mediante la opción *Instalar en la unidad de disco duro local* y continuamos.

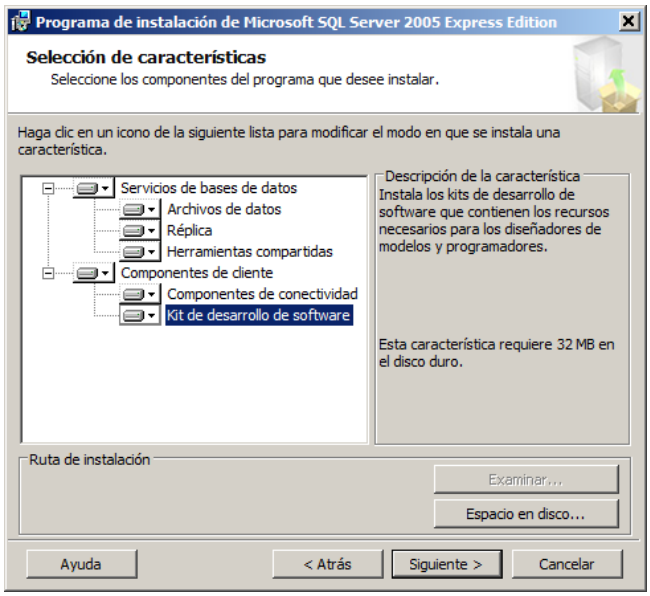


Figura 9.5. Manual de Instalación - Seleccionar componentes a instalar

Especificaremos ahora el nombre para la instancia. En caso de no existir ninguna más en el servidor, podemos hacer uso de la predeterminada. En este caso, ha sido necesario crear una dedicada.

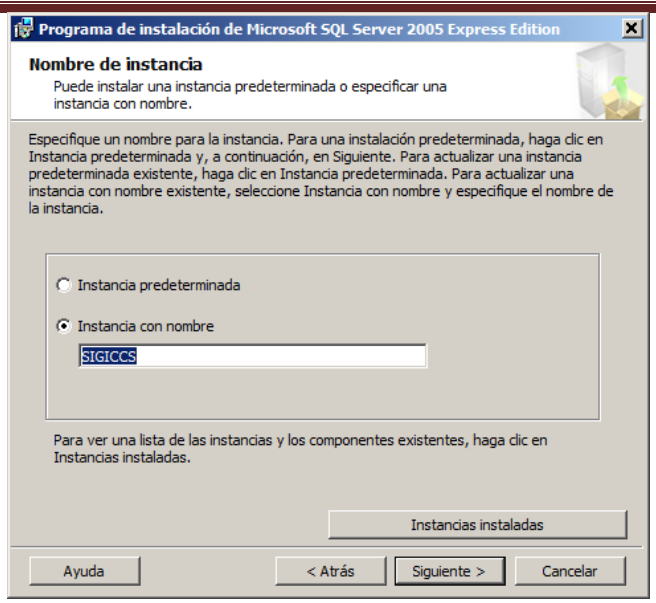


Figura 9.6. Manual de Instalación - Crear instancia de SQL Server

Posteriormente, seleccionaremos el modo de autenticación mixto, mediante el cual podremos conectar con la base de datos por medio del mecanismo de autenticación propio de SQL Server. Para ello, es necesario especificar una contraseña para el usuario predeterminado *"sa"*.

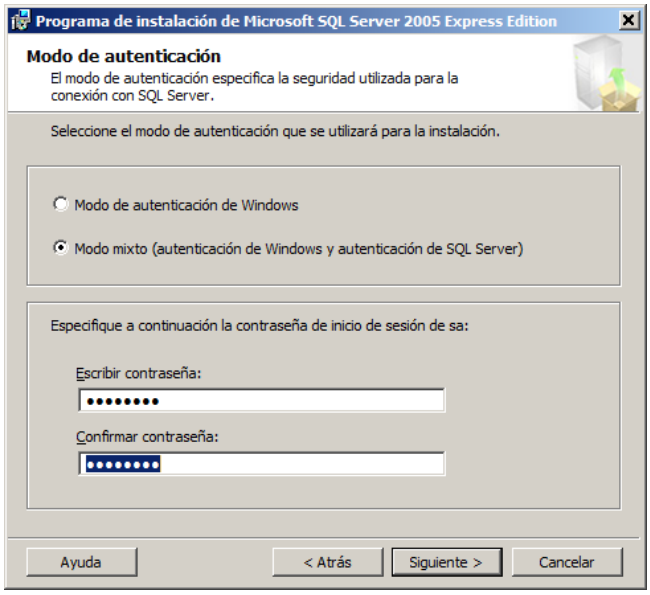


Figura 9.7. Manual de Instalación - Modo de autenticación en SQL Server

Llegado este punto, se muestra un resumen de los componentes a instalar y los parámetros configurados y se solicita confirmación para comenzar la instalación. Pulsamos *Finalizar* y esperamos a que concluya el proceso. Pulsaremos nuevamente en *Finalizar* para concluirlo.

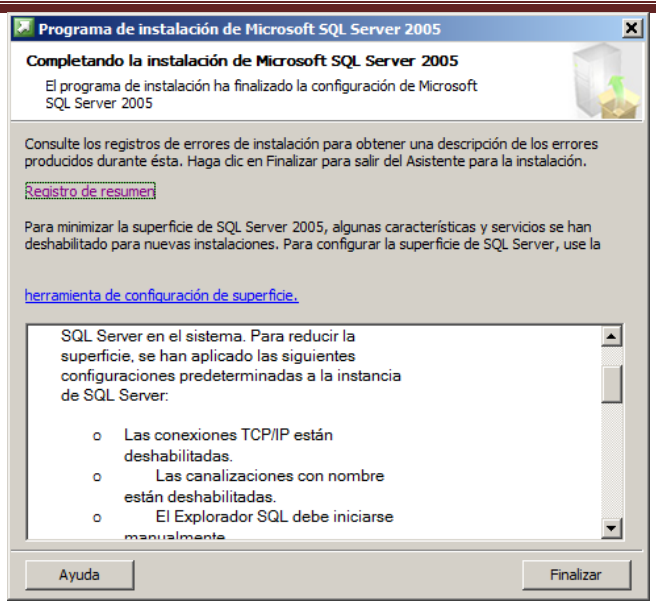


Figura 9.8. Manual de Instalación - Instalación de SQL Server concluida

Como podemos observar en el cuadro de información, las conexiones *TCP/IP* a la base de datos se encuentran deshabilitadas de forma predeterminada. Configuraremos entonces este aspecto para que el servicio acepte conexiones entrantes, necesarias para efectuar la consulta e inserción de datos.

9.1.1.2 Configuración

Para completar la configuración, ejecutaremos el *Administrador de configuración de SQL Server*, situado en *Inicio – Todos los Programas – Microsoft SQL Server 2005 – Herramientas de configuración* y, desplegando el árbol de *Configuración de red de SQL Server 2005*, seleccionamos *Protocolos de <Nombre_de_la_Instancia>*. Configuraremos entonces la propiedad *TCP/IP*.

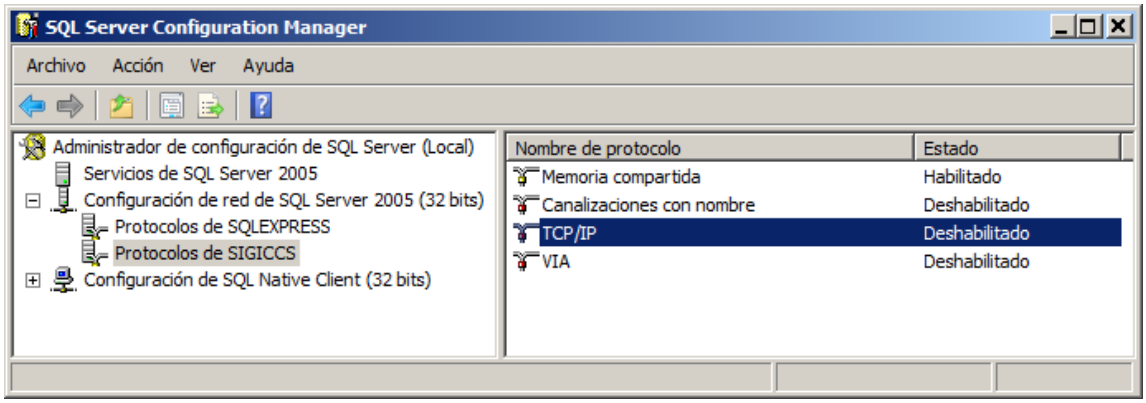


Figura 9.9. Manual de Instalación - Configuración de red de SQL Server

Haciendo clic con el botón derecho del ratón y seleccionando *Propiedades*, estableceremos las siguientes opciones:

- Protocolo: **Habilitado (Sí)**
- Direcciones IP: **IPAll – Puerto TCP (1433)**

Aceptamos y reiniciamos el servicio, seleccionando tal opción en el menú contextual del el nodo *Servicios de SQL Server 2005 – SQL Server (<Nombre_de_la_Instancia>)*.

Realizaremos ahora una breve comprobación para asegurar que el SQL Server se encuentra escuchando conexiones en el puerto especificado. Para ello, nos ayudaremos de la herramienta *SQL Server Management Studio Express*, incluida en la instalación.

Para conectarnos a la instancia recién creada, introducimos los siguientes datos en el diálogo inicial

- Nombre del servidor: **<IP_o_Nombre_del_Servidor>\<Nombre_de_la_Instancia>**
- Autenticación: **Autenticación de SQL Server**
    - Inicio de sesión: **sa**
    - Contraseña: **<contraseña_especificada>**

Figura 9.10. Manual de Instalación - Conexión a SQL Server

Pulsamos *Conectar*. La vista actual muestra el explorador de objetos. Seleccionamos *Nueva Consulta* y ejecutamos la siguiente sentencia T-SQL

```
EXEC xp_readerrorlog 0, 1, N'Server is listening on', N'any', NULL, NULL, 'DESC'
GO
```

Obteniendo la siguiente respuesta

Resultados | Mensajes

	LogDate	ProcessInfo	Text
1	2013-07-11 18:34:10.330	Servidor	Server is listening on ['any' <ipv6> 49656].
2	2013-07-11 18:34:10.330	Servidor	Server is listening on ['any' <ipv4> 49656].
3	2013-07-11 18:34:10.330	Servidor	Server is listening on ['any' <ipv6> 1433].
4	2013-07-11 18:34:10.330	Servidor	Server is listening on ['any' <ipv4> 1433].

Figura 9.11. Manual de Instalación - Comprobación de escucha en SQL Server

Con esta comprobación concluye el proceso de instalación y configuración básica del sistema de gestión de bases de datos empleado en el sistema.

9.1.1.3 Carga de datos

A continuación cargaremos el esquema de la base de datos en la instancia creada. Para ello, nos ayudaremos del *script* de SQL disponible en *./src/sql/initial.sql*

Inicialmente, nos conectamos a la instancia creada mediante la herramienta *SQL Server Management Studio Express*, tal como hicimos en el paso anterior. Una vez hecho esto, seleccionamos *Abrir archivo* en la barra superior y cargamos el *script "initial.sql"*.

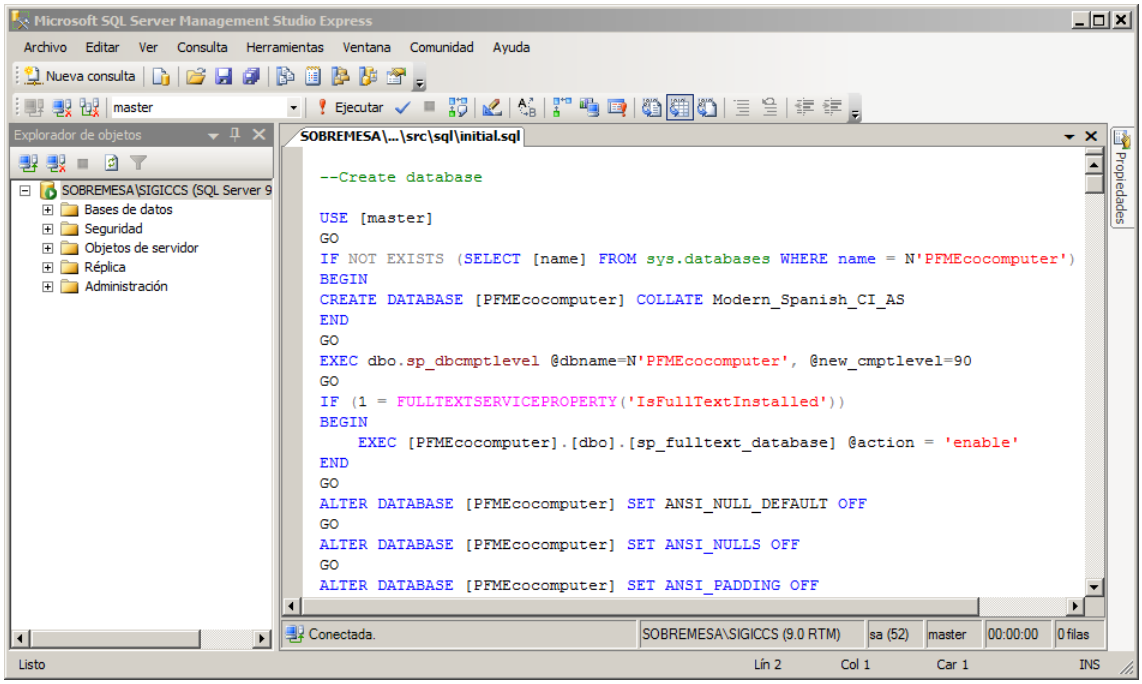


Figura 9.12. Manual de Instalación - Carga de datos

Pulsamos entonces en *Ejecutar*. Si la operación se ejecutó correctamente, veremos el mensaje *"Ejecución de DBCC completada"* en la ventana de resultados y la base de datos estará lista para ser conectada con la aplicación.

9.1.2 Servidor Web

9.1.2.1 Instalación

Para la instalación del servidor web Apache Tomcat se ha provisto un método de actuación simplificado, debido a que:

1. Es necesario modificar gran cantidad de parámetros de configuración y adjuntar librerías requeridas por la aplicación, lo cual podría inducir a errores en el proceso de instalación si se desconoce el sistema.
2. Ya que la instancia de Tomcat puede ser portada copiando únicamente el directorio que contiene sus ficheros, resulta adecuado entregarlo de este modo.

Por todo ello, se puede encontrar la instancia del servidor web completamente configurada en el directorio *./herramientas/tomcat/apache-tomcat.zip*. Para desplegarlo, basta con descomprimirlo en un directorio cualquiera del disco duro del servidor.

Nótese que dicha instancia se encuentra configurada para escuchar peticiones en el puerto 8080, por lo que el cortafuegos del servidor debe permitir las conexiones entrantes a dicho puerto.

Apuntar por último que Tomcat necesita disponer de la máquina virtual de Java instalada en el sistema, cuya instalación y configuración será responsabilidad del administrador del sistema.

9.1.2.2 Configuración

Es necesario recordar que el fichero *context.xml*, situado en el directorio *conf* establece, por defecto, que la conexión con la base de datos se realizará de forma local, es decir, la base de datos se despliega en la misma máquina que el servidor web. De no ser así, es necesario editar el siguiente nodo del fichero XML y sustituir los parámetros en negrita por los establecidos en la instalación y configuración de la base de datos

```
<Resource auth="Container"
driverClassName="com.microsoft.sqlserver.jdbc.SQLServerDriver" maxActive="100"
maxIdle="120" name="jdbc/Ecocomputer" removeAbandoned="true"
type="javax.sql.DataSource"
url="jdbc:sqlserver://IP_DEL_SERVIDOR_DE_BD:PUERTO;DatabaseName=NOMBRE_DE_LA_BD;u
ser=USUARIO;password=CONTRASEÑA;"/>
```

Llegado este punto, tanto el servidor web como la base de datos estarán listos para desplegar el sistema.

9.2 Manual de Ejecución

9.2.1 Inicio y Parada de la Base de Datos

En sistemas Windows, como la gran mayoría de servicios del sistema operativo, es posible iniciar y detener el servicio de SQL Server para arrancar o parar la base de datos. Para ello, haciendo uso de la utilidad *Herramientas de configuración*, elegimos el servicio correspondiente a la base de datos del sistema y, mediante el menú contextual, seleccionamos la opción *Iniciar* o *Detener*, de acuerdo a la acción que se pretende llevar a cabo.

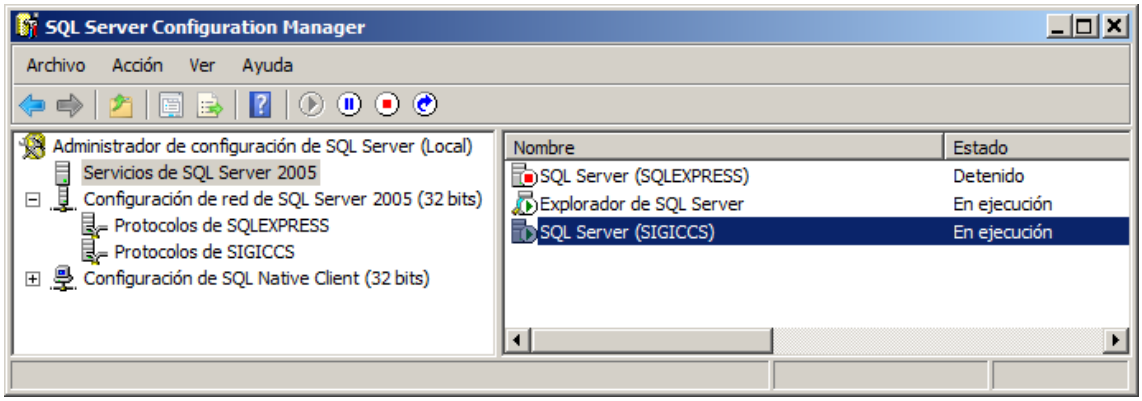


Figura 9.13. Manual de Ejecución - Inicio y parada de la base de datos

9.2.2 Inicio y Parada del Servidor Web

Apache Tomcat integra una serie de *scripts* destinados al inicio y parada del servidor web, denominados *startup.bat – shutdown.bat* y *startup.sh – shutdown.sh* para su uso en sistemas Windows y UNIX respectivamente. Basta con ejecutar el que se precise con los permisos necesarios para arrancar el servicio.

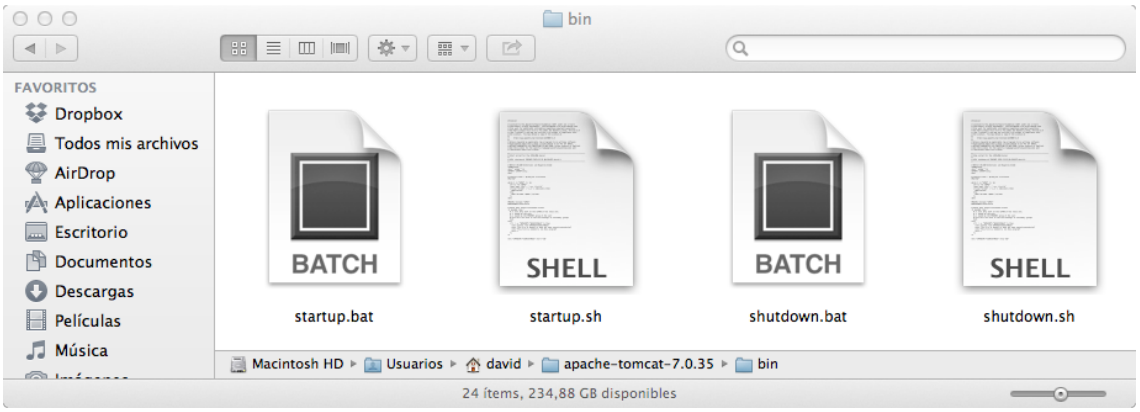


Figura 9.14. Manual de Ejecución - Scripts de inicio y parada del servidor web

9.2.3 Despliegue de la Aplicación

Una vez configurados y desplegados los sistemas anteriores, el despliegue de la aplicación resulta uno de los pasos más sencillos. Basta con copiar los ficheros *intranet.war* y *sigiccs.war*, situados en el directorio *./dist*, a la carpeta *webapps* de Tomcat. Tras unos segundos, el servidor analizará el contenido de los paquetes y los desplegará.

Finalmente, solo quedaría probar que todo funciona como debería, para lo cual accederemos desde un navegador web a la dirección http://IP_DEL_SERVIDOR:8080/Intranet, mostrándose así la interfaz de acceso al sistema.

9.3 Manual de Usuario

Este manual pretende servir de guía para aquellos usuarios que utilicen el sistema por primera vez. Para facilitar su comprensión, se ha estructurado en cuatro secciones, describiendo en cada una de ellas las operaciones que pueden ser llevadas a cabo en el módulo correspondiente de la aplicación.

9.3.1 Operaciones con Usuarios

A pesar de que el módulo de gestión de usuarios cuenta con más funciones, aquí se describen únicamente aquellas que resultan relevantes para el uso de este sistema.

9.3.1.1 Iniciar sesión en la aplicación

Una vez que el usuario conozca sus credenciales y la URL de acceso a la aplicación, se debe acceder a dicha dirección web a través del navegador. Se mostrará entonces una interfaz de bienvenida con el formulario de acceso, donde deben completarse los campos *Nombre de usuario* y *Contraseña*, pulsando finalmente en *Iniciar sesión*.

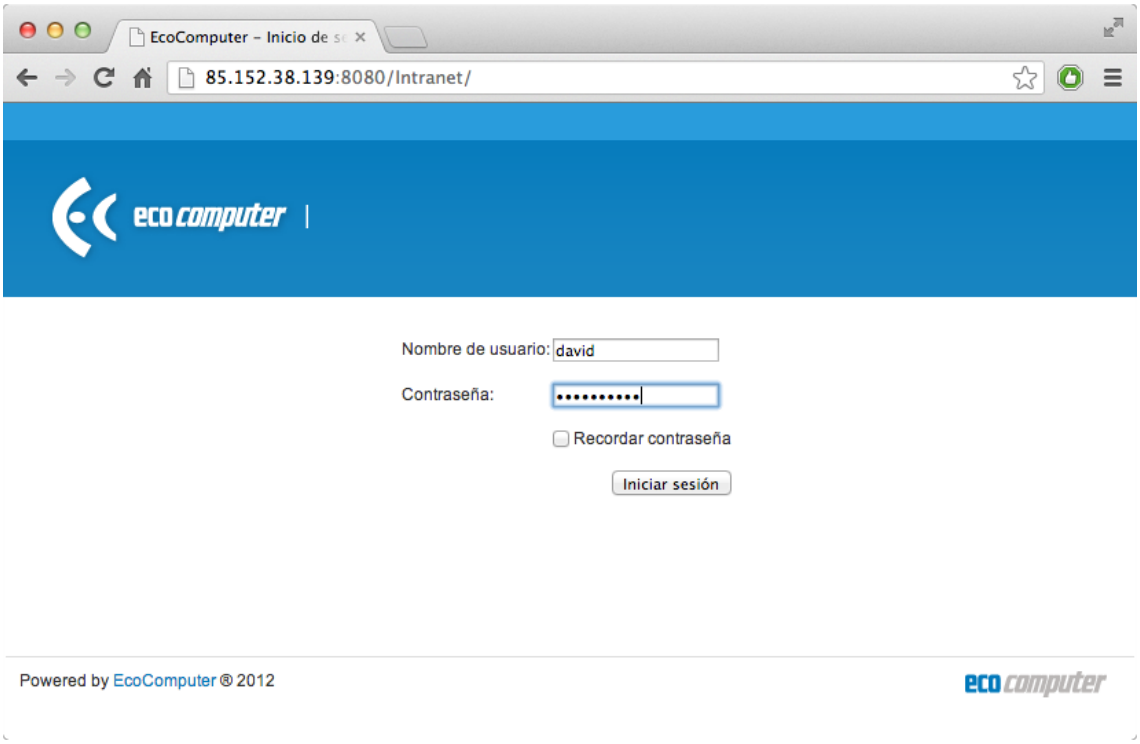


Figura 9.15. Manual de Usuario - Iniciar sesión

Si los datos de acceso son correctos, el sistema mostrará los datos del usuario, tales como nombre real, dirección y formas de contacto.

9.3.1.2 Cerrar la sesión en curso

Para cerrar correctamente la sesión de usuario, se dispone del enlace *Cerrar Sesión* en la parte superior de la interfaz. Una vez pulsado, la sesión terminará y se muestra la interfaz de bienvenida.

Figura 9.16. Manual de Usuario - Cerrar sesión

9.3.2 Operaciones con Clientes

9.3.2.1 Consultar el listado de clientes

Accediendo a la pestaña *Clientes* del menú superior se mostrará el listado de clientes activos en el sistema.

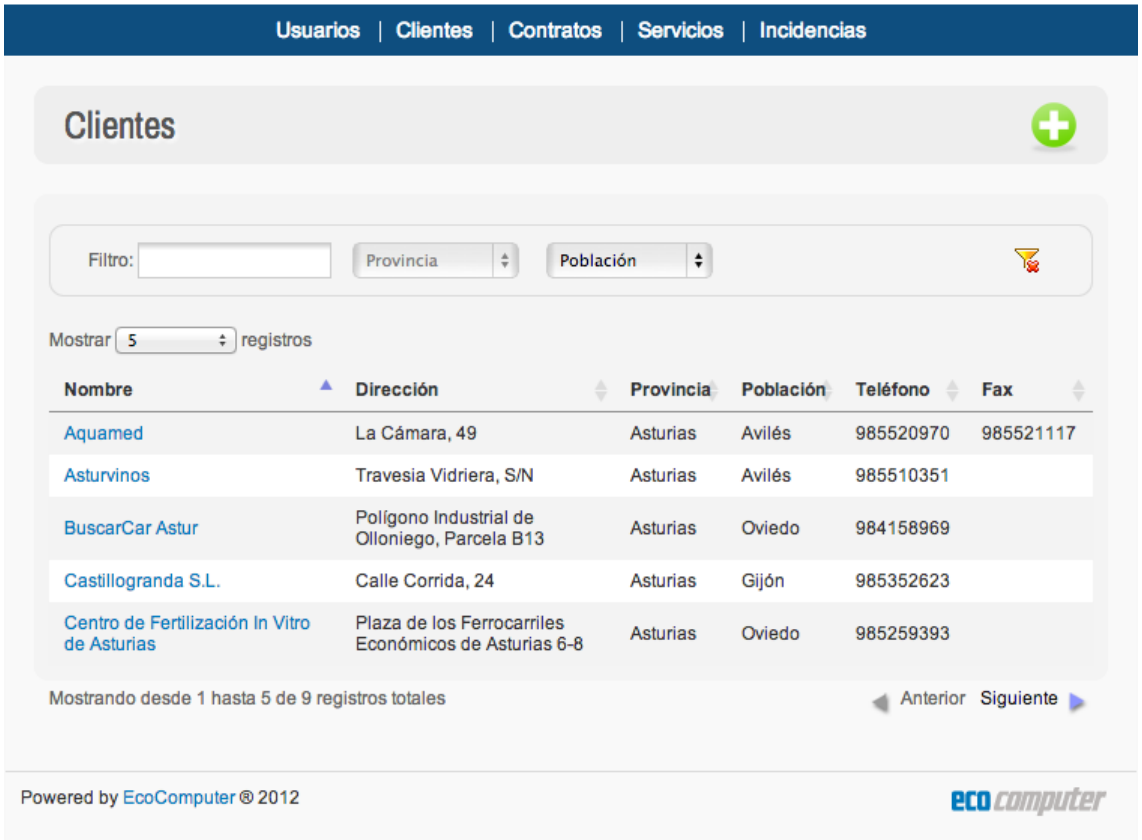


Figura 9.17. Manual de Usuario - Consultar listado de clientes

Es posible que el número de clientes para mostrar sea elevado, por lo cual, se dispone de un selector para la cantidad de registros a mostrar y controles de paginación para consultar el resto de entradas, de haberlas.

Por medio de los controles de la barra de filtros es posible buscar los registros de clientes que cuyos datos cumplan las condiciones marcadas. Pulsando el botón se restablecerá el filtro aplicado.

9.3.2.2 Dar de alta un cliente

Para introducir un nuevo cliente, desde la vista del listado, debemos pulsar el botón . Se mostrará entonces el formulario de entrada de datos que completaremos con la información del nuevo elemento, pulsando finalmente en *Guardar*.

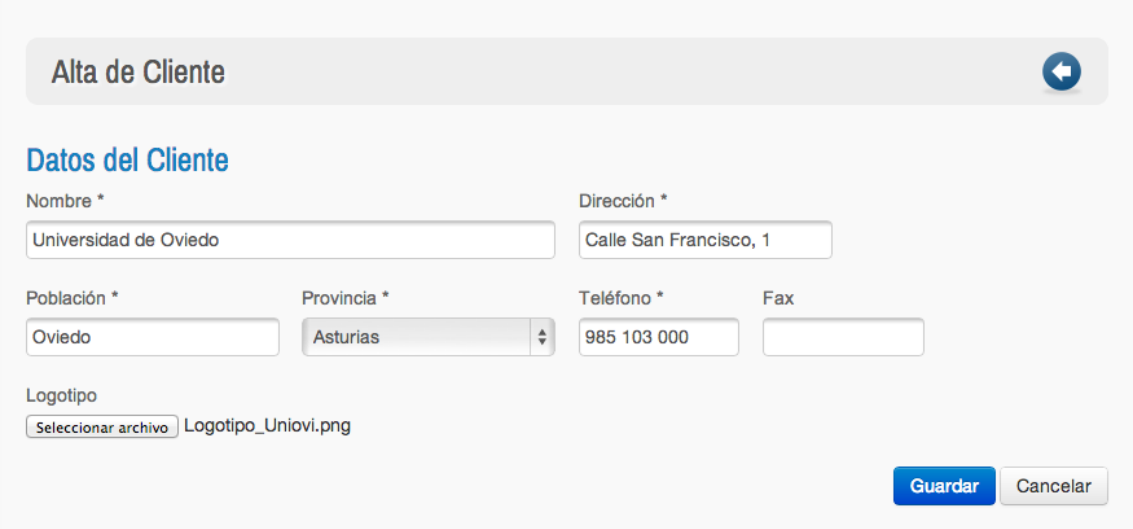


Figura 9.18. Manual de Usuario - Alta de cliente

El nuevo cliente aparecerá entonces en el listado.

Figura 9.19. Manual de Usuario - Nuevo cliente insertado

9.3.2.3 Consultar la ficha de un cliente

Para consultar el detalle de un cliente, basta con seleccionarlo pulsando en su nombre en el listado.

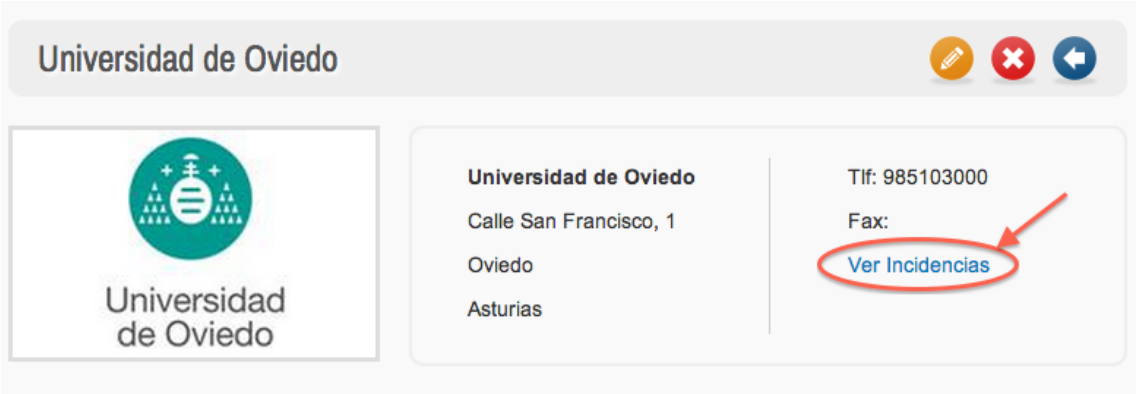


Figura 9.20. Manual de Usuario - Ver ficha del cliente

Desde esta vista es posible además acceder al listado de incidencias vinculadas al cliente en cuestión pulsando en el enlace *Ver Incidencias*.

9.3.2.4 Modificar los datos de un cliente

Para actualizar los datos de un cliente, desde la vista detalle de éste, pulsaremos el botón . De este modo, se muestra el mismo formulario que en el proceso de alta, solo que los datos actuales aparecen precargados. Modificamos los datos necesarios y pulsamos *Guardar*.

Figura 9.21. Manual de Usuario - Modificar cliente

9.3.2.5 Eliminar un cliente

Para eliminar un cliente, basta con pulsar en el botón en la ficha del mismo. Es necesario confirmar la acción mediante el botón *Aceptar* del diálogo que aparecerá a continuación.

9.3.3 Operaciones con Servicios y Tarifas

9.3.3.1 Consultar el catálogo de servicios

Accediendo a la pestaña *Servicios* del menú superior se mostrará el catálogo de servicios disponibles, así como las tarifas asociadas a cada uno de ellos.

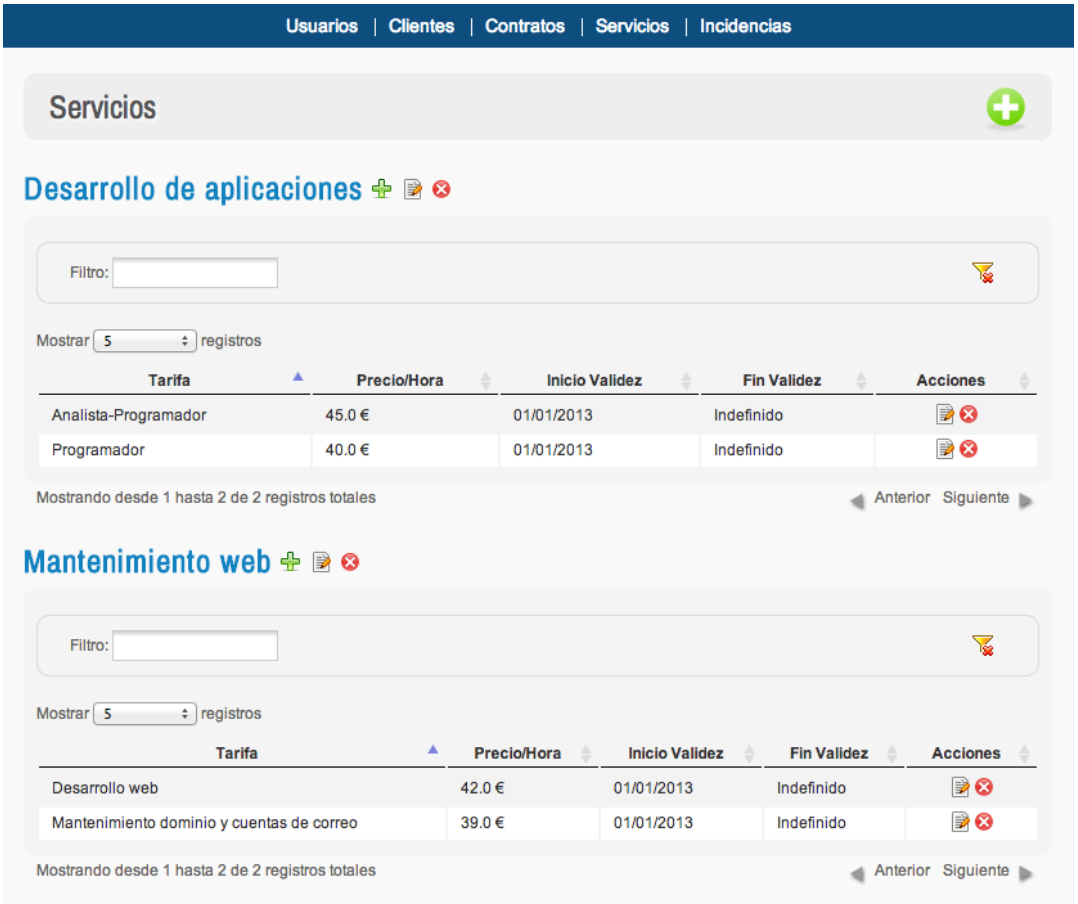


Figura 9.22. Manual de Usuario - Consultar catálogo de servicios

9.3.3.2 Dar de alta un servicio

Para introducir un nuevo servicio, desde la vista del catálogo, debemos pulsar el botón . Se mostrará entonces el formulario de entrada de datos que completaremos con la descripción del nuevo elemento, pulsando finalmente en *Guardar*.

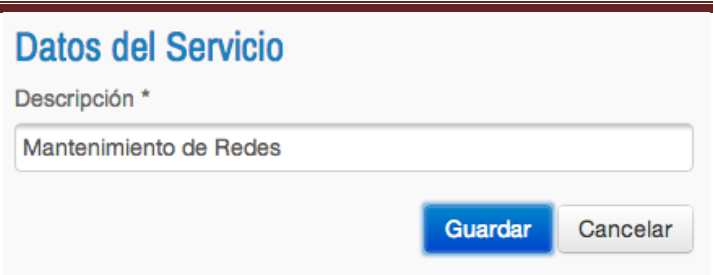


Figura 9.23. Manual de Usuario - Alta de servicio

9.3.3.3 Modificar la descripción de un servicio

La operación de actualización de los datos del elemento se puede llevar a cabo pulsando el botón situado a continuación del nombre del servicio en cuestión.

Figura 9.24. Manual de Usuario - Modificar servicio

9.3.3.4 Eliminar un servicio

Para dar de baja un servicio, basta con pulsar en el botón situado a continuación de su nombre. Es necesario confirmar la acción mediante el botón *Aceptar* del diálogo que aparecerá a continuación.

Figura 9.25. Manual de Usuario - Eliminar servicio

9.3.3.5 Asociar una tarifa a un servicio

Para incluir una tarifa en un servicio, pulsaremos el botón , situado a continuación del nombre de éste.

Figura 9.26. Manual de Usuario - Asociar tarifa al servicio

Se mostrará entonces el formulario de entrada de datos que completaremos con la información de la nueva tarifa, pulsando finalmente en *Guardar*.

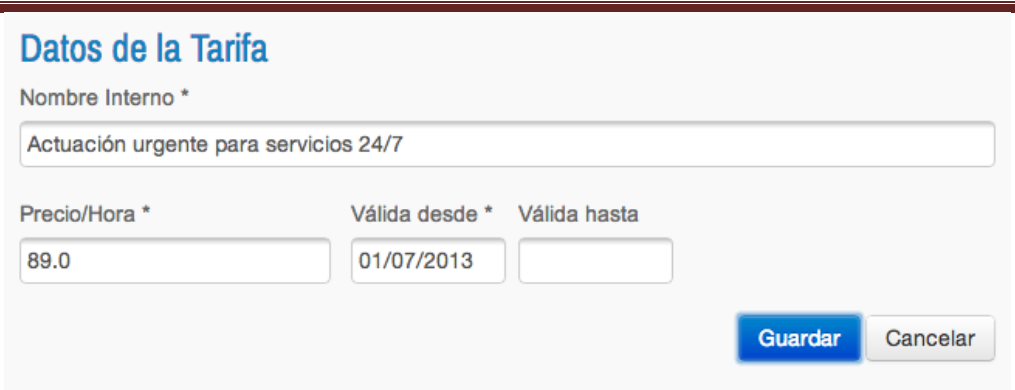


Figura 9.27. Manual de Usuario - Alta de tarifa

La nueva tarifa aparecerá asociada al contrato correspondiente.

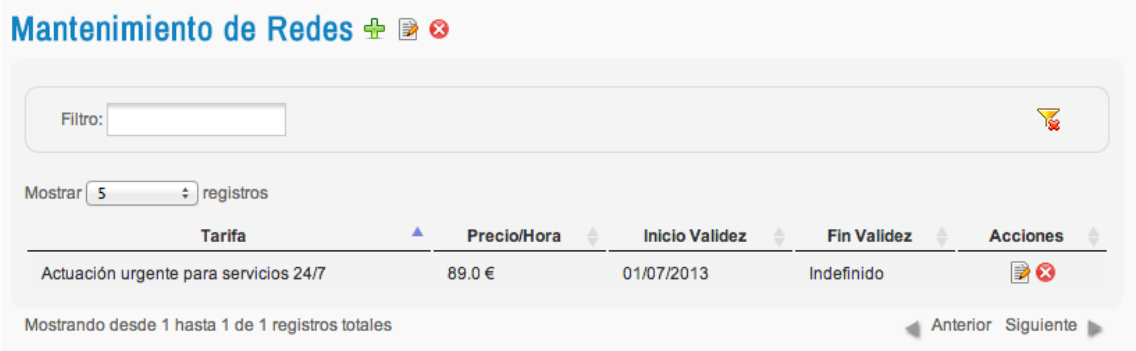


Figura 9.28. Manual de Usuario - Nueva tarifa asociada al servicio

9.3.3.6 Modificar o eliminar una tarifa

Tanto la modificación como la eliminación de tarifas siguen un proceso idéntico al descrito anteriormente para los elementos *Servicio*. Es por ello que serán obviados en este manual.

9.3.4 Operaciones con Contratos

Una vez dados de alta los servicios ofertados y las tarifas asociadas a estos, es momento de gestionar los contratos celebrados con los clientes.

9.3.4.1 Consultar el listado de contratos

Accediendo a la pestaña *Contratos* del menú superior se mostrará el listado de contratos dados de alta en el sistema.

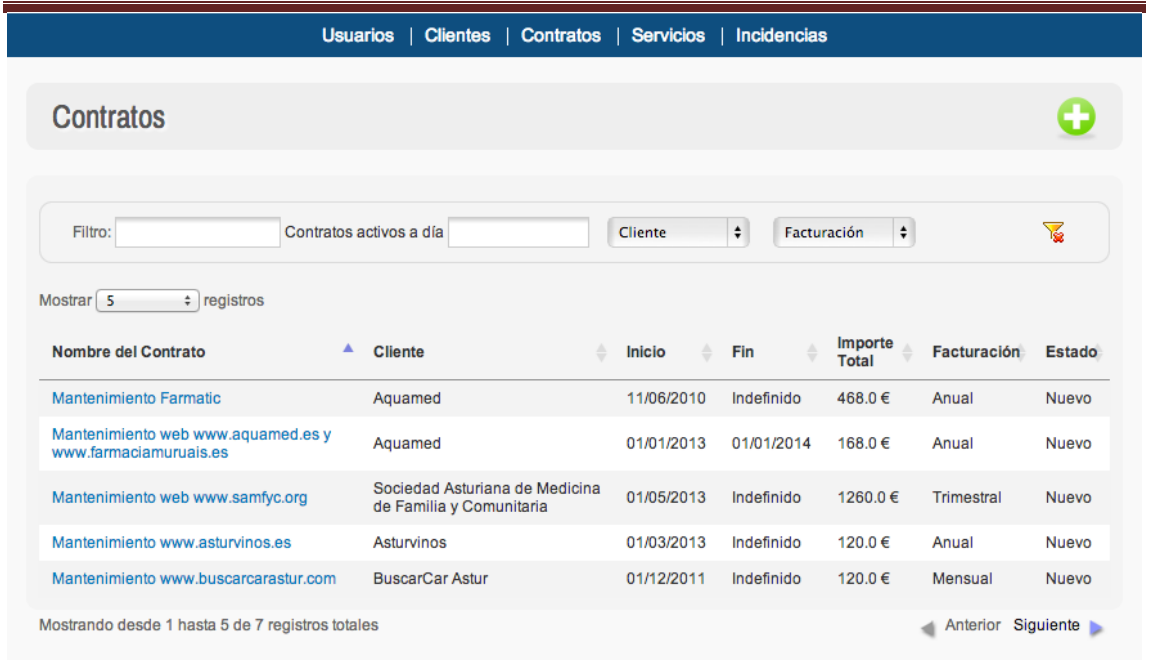


Figura 9.29. Manual de Usuario - Consultar listado de contratos

9.3.4.2 Dar de alta un contrato

Para introducir un nuevo contrato en el sistema, desde la vista del listado, debemos pulsar el botón . Se mostrará entonces el formulario de entrada de datos que completaremos con la información del nuevo elemento, pulsando finalmente en *Guardar*.

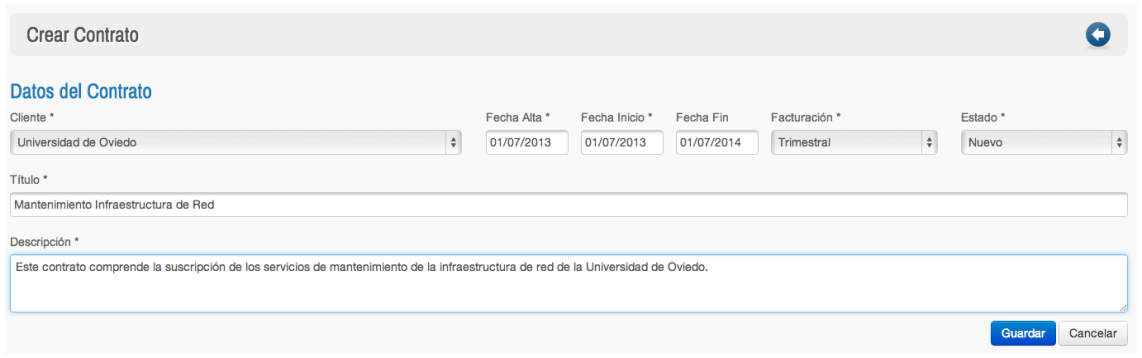


Figura 9.30. Manual de Usuario - Alta de contrato

Figura 9.31. Manual de Usuario - Nuevo contrato insertado

9.3.4.3 Consultar la ficha de un contrato

Para consultar el detalle de un contrato, basta con seleccionarlo pulsando en su título en el listado. Nótese que el contrato no tiene todavía ningún servicio asociado.

Figura 9.32. Manual de Usuario - Ver ficha del contrato

9.3.4.4 Modificar los datos de un contrato

Para actualizar los datos de un contrato, desde la vista detalle de éste, pulsaremos el botón . De este modo, se muestra el mismo formulario que en el proceso de alta, solo que los datos actuales aparecen precargados. Modificamos los datos necesarios y pulsamos *Guardar*.

Figura 9.33. Manual de Usuario - Modificar contrato

9.3.4.5 Eliminar un contrato

Para eliminar un contrato, basta con pulsar en el botón en la ficha del mismo. Es necesario confirmar la acción mediante el botón *Aceptar* del diálogo que aparecerá a continuación.

9.3.4.6 Asociar un servicio al contrato

Para incluir un servicio en el contrato, pulsaremos el botón , situado en la parte inferior de la vista de edición del elemento.

Figura 9.34. Manual de Usuario - Asociar servicio al contrato

Se mostrará entonces el formulario de entrada de datos que completaremos con la información del servicio en cuestión, pulsando finalmente en *Guardar*.

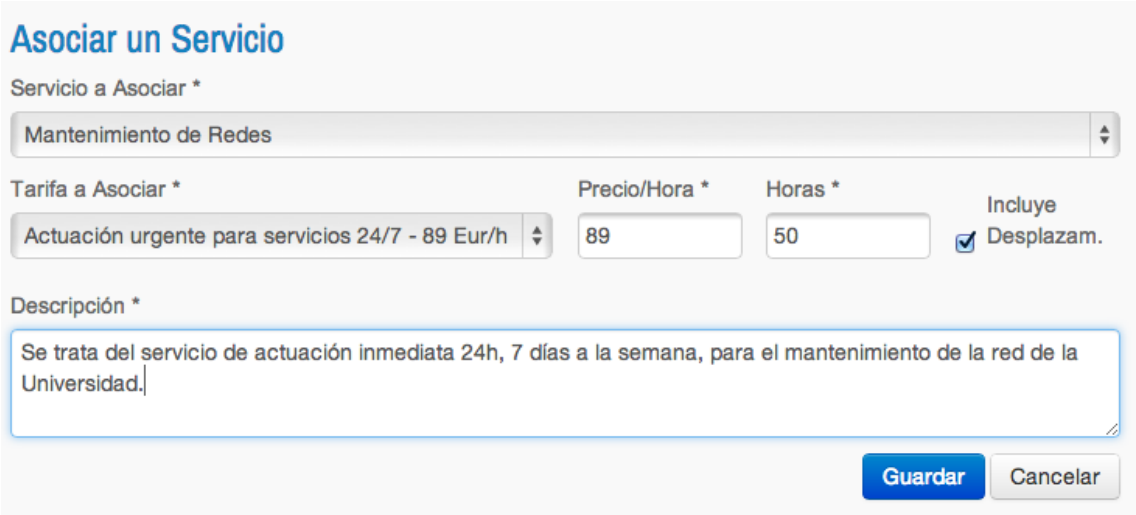


Figura 9.35. Manual de Usuario - Asociación de servicio a un contrato

A la hora de asociar un servicio aplicándole un precio especial, es necesario elegir la opción *Personalizada* en el desplegable de tarifas y especificar manualmente el precio por hora a cobrar.

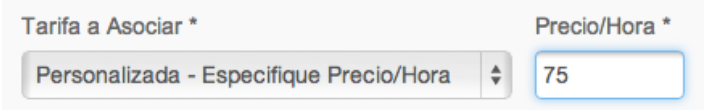


Figura 9.36. Manual de Usuario - Asociación de servicio con tarifa especial

El nuevo servicio asociado aparecerá ahora en la ficha del contrato.

Figura 9.37. Manual de Usuario - Detalle de contrato con servicios asociados

9.3.4.7 Modificar o eliminar una tarifa

Tanto la modificación como la eliminación de contratos y servicios asociados a éstos siguen un proceso idéntico al descrito anteriormente para el elemento *Cliente.* Es por ello que serán obviados en este manual.

9.3.5 Operaciones con Incidencias

Una vez completados los pasos anteriores, el módulo de gestión de incidencias dispondrá de todas las entidades de las que hace uso en sus funciones.

9.3.5.1 Consultar el listado de incidencias

Accediendo a la pestaña *Incidencias* del menú superior se mostrará el listado de incidencias dadas de alta en el sistema.

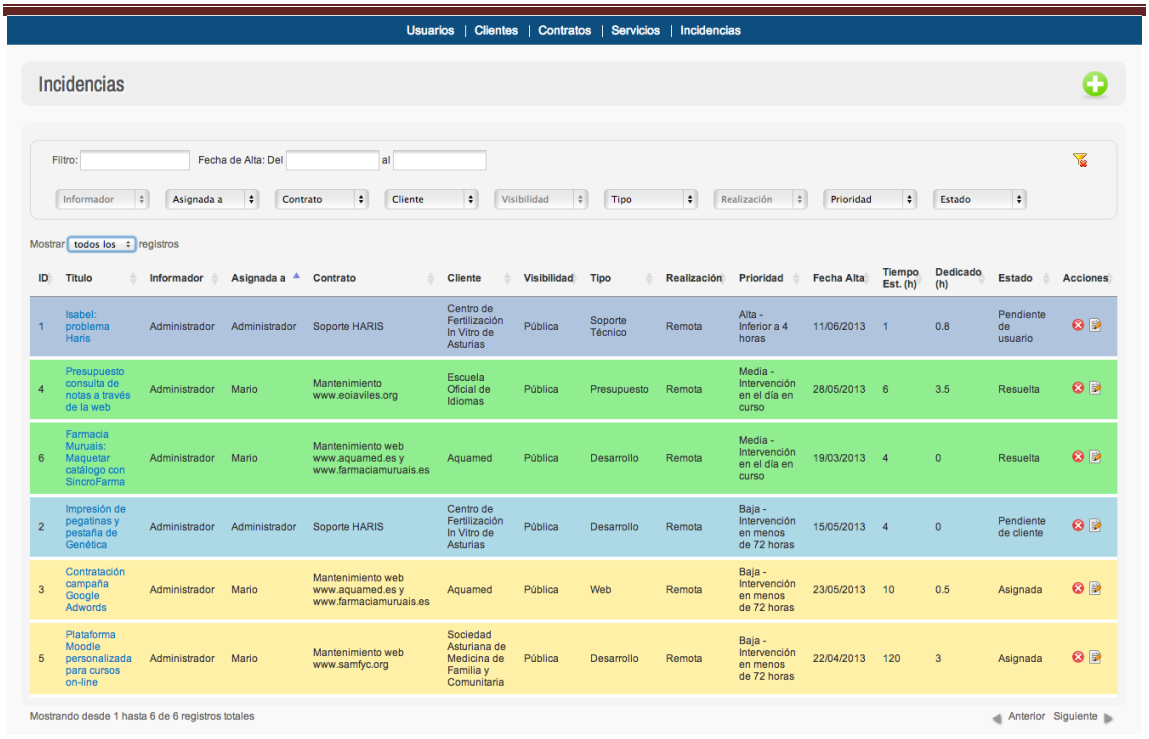


Figura 9.38. Manual de Usuario - Consultar listado de incidencias

9.3.5.2 Dar de alta una incidencia

Para introducir una incidencia en el sistema, desde la vista del listado, debemos pulsar el botón . Se mostrará entonces el formulario de entrada de datos que completaremos con la información del nuevo elemento, pulsando finalmente en *Guardar*.

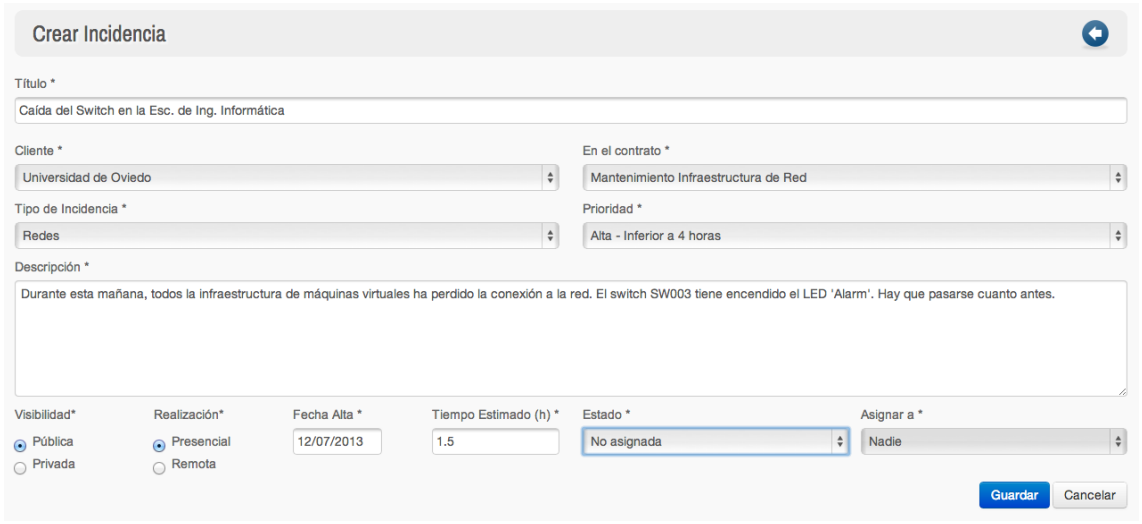


Figura 9.39. Manual de Usuario - Alta de incidencia

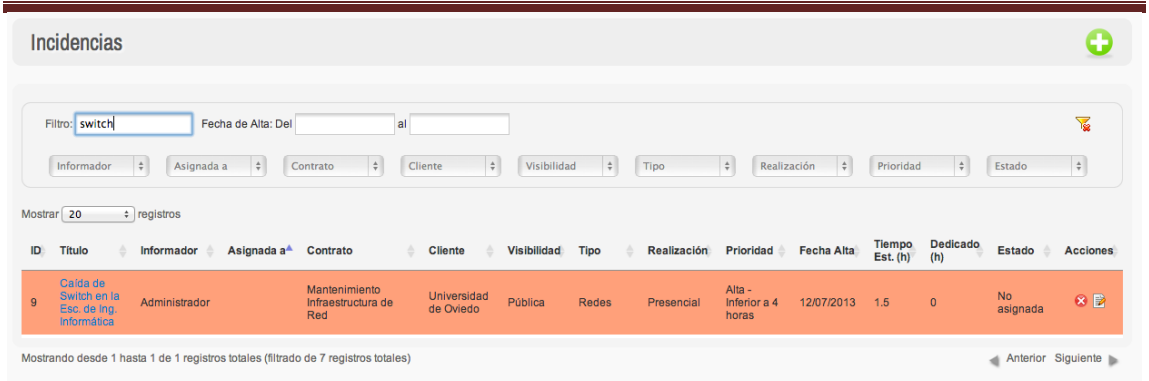


Figura 9.40. Manual de Usuario - Nueva incidencia insertada

9.3.5.3 Consultar la ficha de una incidencia

Para consultar el detalle de una incidencia, basta con seleccionarla pulsando en su título en el listado. Nótese que la incidencia no tiene todavía ningún elemento extra asociado.

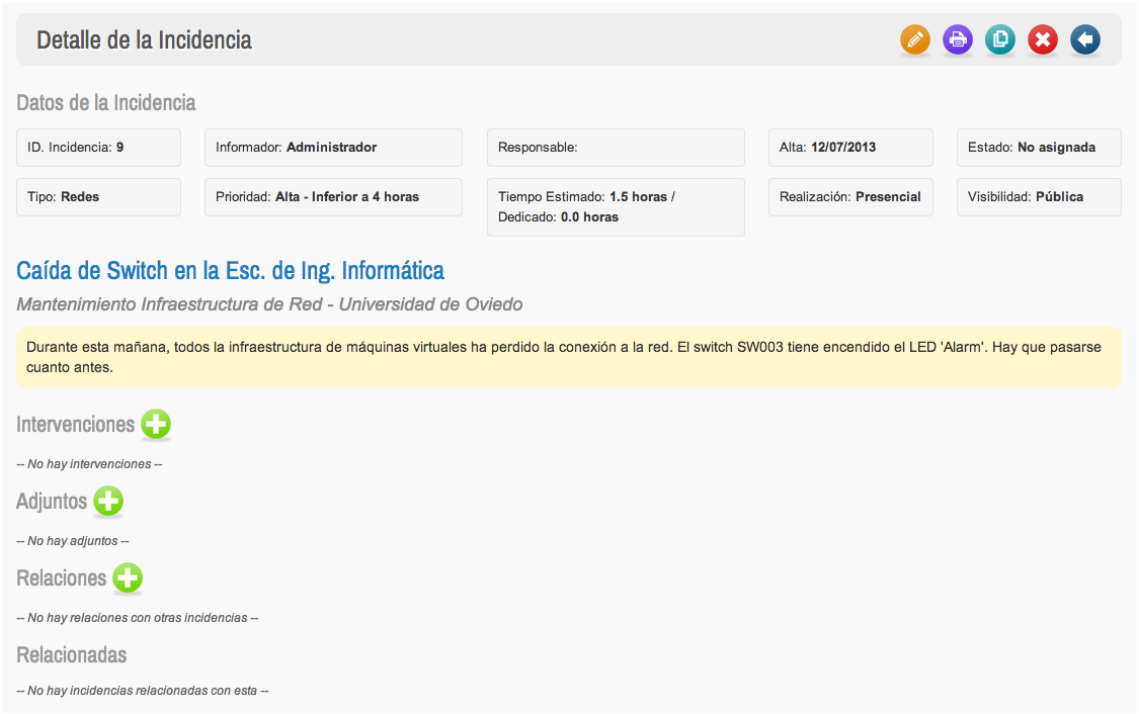


Figura 9.41. Manual de Usuario - Ver ficha de la incidencia

9.3.5.4 Modificar los datos de una incidencia

Para actualizar los datos de una incidencia, desde la vista detalle de ésta, pulsaremos el botón . De este modo, se muestra el mismo formulario que en el proceso de alta, solo que los datos actuales aparecen precargados. Modificamos los datos necesarios y pulsamos *Guardar*.

Figura 9.42. Manual de Usuario - Modificar incidencia

9.3.5.5 Eliminar una incidencia

Para eliminar una incidencia, basta con pulsar en el botón en la ficha de la misma. Es necesario confirmar la acción mediante el botón *Aceptar* del diálogo que se mostrará.

9.3.5.6 Anotar una intervención a una incidencia

Para incluir una intervención en la incidencia, pulsaremos el primer botón , situado en la parte inferior de la ficha de la misma.

Figura 9.43. Manual de Usuario - Anotar intervención a la incidencia

Se mostrará entonces el formulario de entrada de datos que completaremos con la información de la intervención en cuestión, pulsando finalmente en *Guardar*.

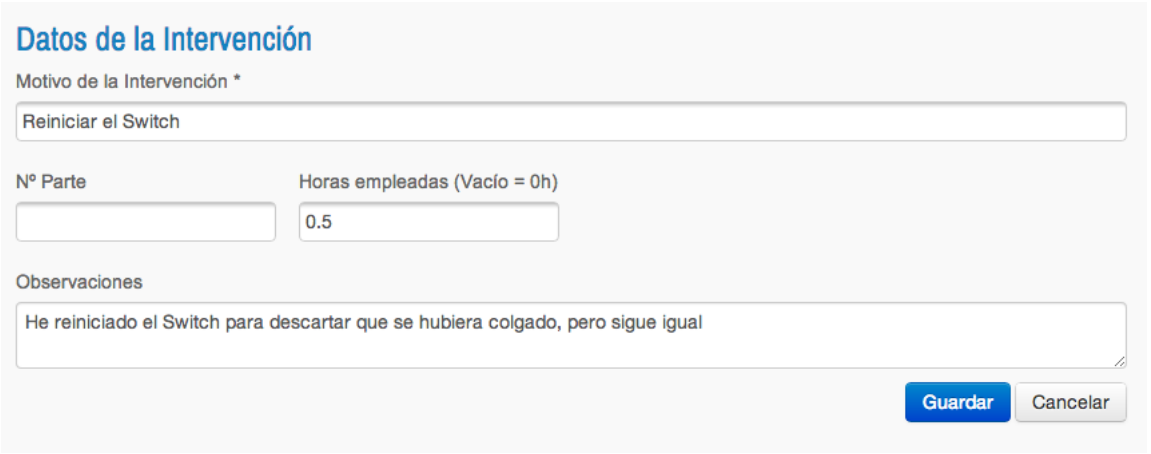


Figura 9.44. Manual de Usuario - Formulario de creación de intervención

Figura 9.45. Manual de Usuario - Intervención anotada a la incidencia

9.3.5.7 Adjuntar un fichero a la incidencia

Para incluir una intervención en la incidencia, pulsaremos el segundo botón , situado en la parte inferior de la ficha de la misma.

Figura 9.46. Manual de Usuario - Adjuntar fichero a la incidencia

Se mostrará entonces el formulario de entrada de datos que completaremos con la descripción del fichero y su ruta dentro de nuestro equipo.

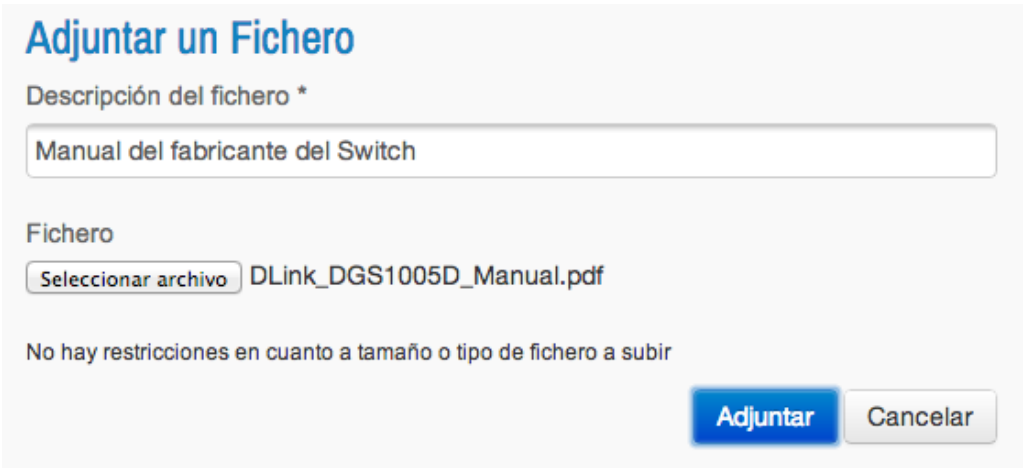


Figura 9.47. Manual de Usuario - Formulario de subida de fichero

Figura 9.48. Manual de Usuario - Fichero adjunto a la incidencia

9.3.5.8 Relacionar la incidencia con otras

Es posible establecer vínculos entre incidencias dependientes o que simplemente, tienen algo que ver. Para ello, pulsaremos el último botón , situado en la parte inferior de la ficha de la incidencia a relacionar.

Figura 9.49. Manual de Usuario - Establecer relación en la incidencia

Se mostrará entonces el formulario de entrada de datos que completaremos con el tipo de relación a establecer y el identificador de la incidencia relacionada, pulsando finalmente en *Guardar*.

Figura 9.50. Manual de Usuario - Formulario de creación de relación

Figura 9.51. Manual de Usuario - Relaciones directas de la incidencia

Una vez establecido el vínculo, es posible acceder directamente a la incidencia relacionada pulsando en el enlace que muestra, entre paréntesis, su identificador.

Adicionalmente, las incidencias que guarden relación con la visualizada serán mostradas en la sección *Relacionadas*.

Figura 9.52. Manual de Usuario - Relaciones recíprocas de la incidencia

9.3.5.9 Modificar o eliminar elementos de la incidencia

Tanto la modificación como la eliminación de intervenciones, adjuntos y relaciones siguen un proceso idéntico al descrito anteriormente para los elementos *Tarifa* y *Servicio.* Es por ello que serán obviados en este manual.

9.3.5.10 Clonar una incidencia

El propósito de clonar una incidencia es ahorrar tiempo a la hora de crear una nueva cuyos datos son similares a una existente. Para ello, pulsando el botón se duplicará la incidencia visualizada. La copia contará con unos datos idénticos a la original, a excepción de los elementos anexos (intervenciones, adjuntos y relaciones) que no se incluyen en la copia.

9.3.5.11 Imprimir la ficha de la incidencia

Es posible imprimir un informe formateado del detalle completo de la incidencia visualizada. Para ello, haciendo uso del botón y posteriormente, del cuadro de diálogo de impresión del navegador, se obtendrá el informe solicitado en el soporte elegido (documento PDF o papel).

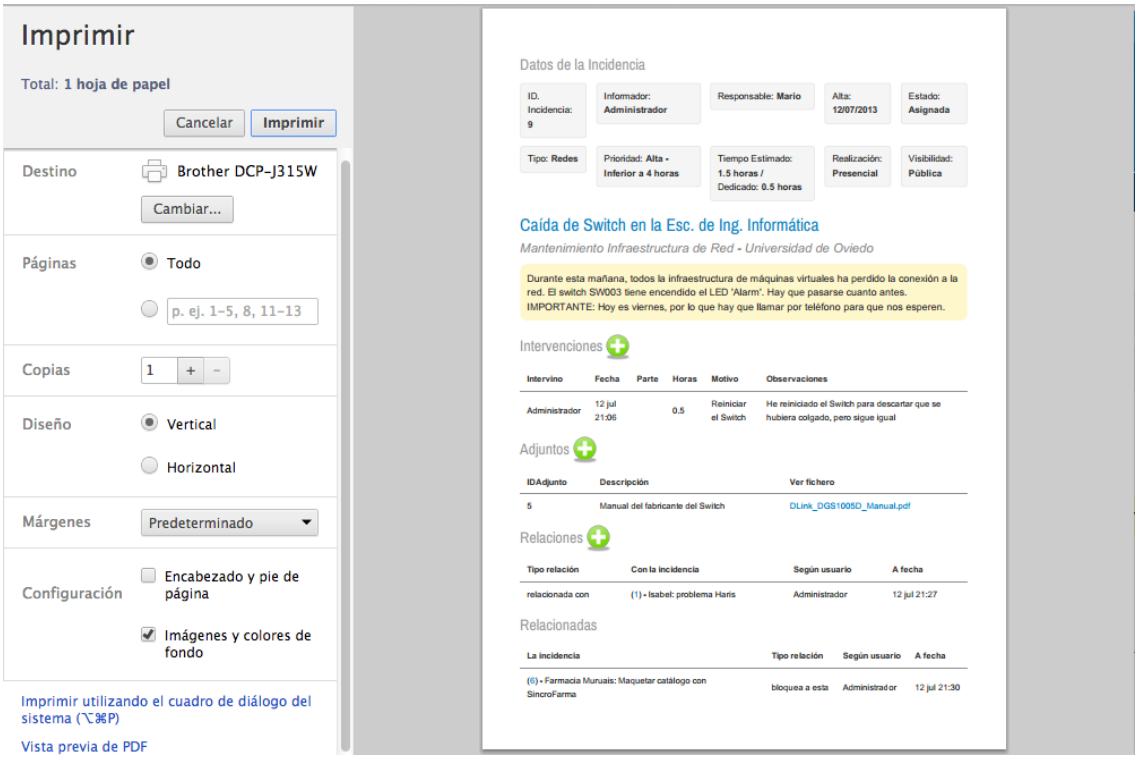


Figura 9.53. Manual de Usuario - Diálogo de impresión del informe en el navegador

9.4 Manual del Programador

Aunque se considera que los diferentes apartados de esta documentación describen suficientemente todos los aspectos de la construcción del sistema, se puntualizan seguidamente, de forma sintetizada, algunos detalles importantes a tener en cuenta de cara a la extensión o modificación de la aplicación desde el punto de vista del programador.

- La **arquitectura del sistema** se basa en el patrón de diseño **Modelo-Vista-Controlador**, separando así **presentación**, **lógica de negocio** y **persistencia**. Es preciso comprender los fundamentos de este patrón arquitectónico para entender adecuadamente su implementación.
- En caso de precisar **extender el modelo y/o funcionalidad**, se recomienda encarecidamente seguir la **jerarquía de clases e interfaces** implementada, procediendo de igual modo con las JSP encargadas de la presentación. Asimismo, es importante declarar correctamente la **inyección de dependencias** de los nuevos elementos en el fichero *applicationContext.xml*.
- La gestión de elementos ***Usuario*** no corresponde con las responsabilidades de este sistema, aunque es necesaria para su funcionamiento. Además de autenticar al usuario en la aplicación, es posible **acceder a sus propiedades** desde cualquier Action, para lo cual se implementa la interfaz *SessionAware* y se invoca al método *session.get("usuario")*.
- Tanto la clase Usuario como todas las asociadas ésta (Perfil, Grupo…) se utilizan desde una **librería Java**, denominada **ecoclasses.jar** e incluida en el directorio *lib* de Apache Tomcat.
- La **conexión con la base de datos** se realiza por medio de un ***pool* de conexiones**, lo que permite optimizar los recursos, reduciendo la carga de la red y consecuentemente, los tiempos de espera entre operaciones. La sección 9.1.2.2 describe la configuración de este elemento.
- Las **JSP** se construyen mediante el uso de plantillas HTML. Se trata de un **framework de *templating***, denominado ***Apache Tiles*** y basado en el patrón Composite. Cada página se compone de varios ***pagelets*** que se corresponden con **plantillas**, **definiciones** y **otras páginas**. En el momento de generar la página, los *pagelets* son **combinados**, dando como resultado el **código HTML** final.

Capítulo 10. Conclusiones y Ampliaciones

10.1 Conclusiones

Al comienzo de este proyecto, la incertidumbre era un aspecto tan poco deseable como presente. El cliente contaba con una necesidad que no había sido satisfecha con diversos productos probados, los cuales cuentan con gran bagaje en lo relativo a su desarrollo e implantación. Partiendo de la especificación del problema en una reunión informal, se comienza a dar forma a la idea del sistema aquí presentado.

Tras decenas de revisiones de requisitos y reuniones con empleados de diferentes departamentos de la empresa, donde se comenzó a experimentar la realidad del trato con el cliente, se consigue establecer un catálogo de requisitos, en principio, cerrado. Esta es la primera de las conclusiones a extraer del proyecto; el proceso de desarrollo de software comienza mucho antes de abrir el entorno de desarrollo.

La fase de análisis y posterior diseño del sistema dio lugar a la aplicación práctica de un concepto inculcado hasta ahora en la teoría: el trabajo colaborativo. Fue necesario cooperar a lo largo del proyecto con diversos equipos de trabajo, tales como diseñadores gráficos, administradores de bases de datos y personal de administración empresarial, además de con otros desarrolladores encargados de la implementación de funcionalidades que coexistirán con las aquí desarrolladas. Nuevamente, de estas participaciones surgieron eventualidades a las que fue necesario adaptarse e incluso asesorar en la toma de decisiones que afectarían colateralmente al sistema desarrollado.

Aún con todo, llegado el punto de hacer balance tras los seis meses que ha durado el proyecto, puede concluirse que los objetivos han sido cumplidos con éxito. El cliente cuenta con un conjunto de herramientas desarrolladas completamente a su medida y que se ajustan a la idea planteada en la primera toma de contacto. Los que serán los usuarios del sistema se consideran satisfechos con la solución ofrecida.

Por parte del autor, se considera una gran satisfacción el haber sido capaz de acometer un proyecto de tal envergadura, enfocado a un cliente real, colaborando con otros compañeros y, aún más importante, estudiando cada decisión de cara a desarrollar un producto, además de funcional, de calidad. Se han estudiado además tecnologías desconocidas hasta la fecha y profundizado en las existentes, todo lo cual conforma una amalgama de conocimientos adquiridos que no habría sido posible obtener en otras circunstancias.

10.2 Ampliaciones

A continuación se describen algunas de las ampliaciones aplicables al sistema desarrollado. Muchas se corresponden con requisitos eliminados de la especificación inicial y el resto son ideas surgidas a posteriori por parte del desarrollador.

10.2.1 Gestión de las Instalaciones e Infraestructura de los Clientes

Consistiría en incluir, para cada cliente, una vista de su infraestructura de sistemas informáticos en modo arbóreo, especificando así su red, servidores, PCs, periféricos, etcétera, con el objetivo de disponer de la trazabilidad completa de cada uno de los elementos, adjuntando su factura de compra o instalación, intervenciones realizadas, mejoras, actualizaciones y licencias asociadas. La siguiente figura muestra un esbozo de esta vista.

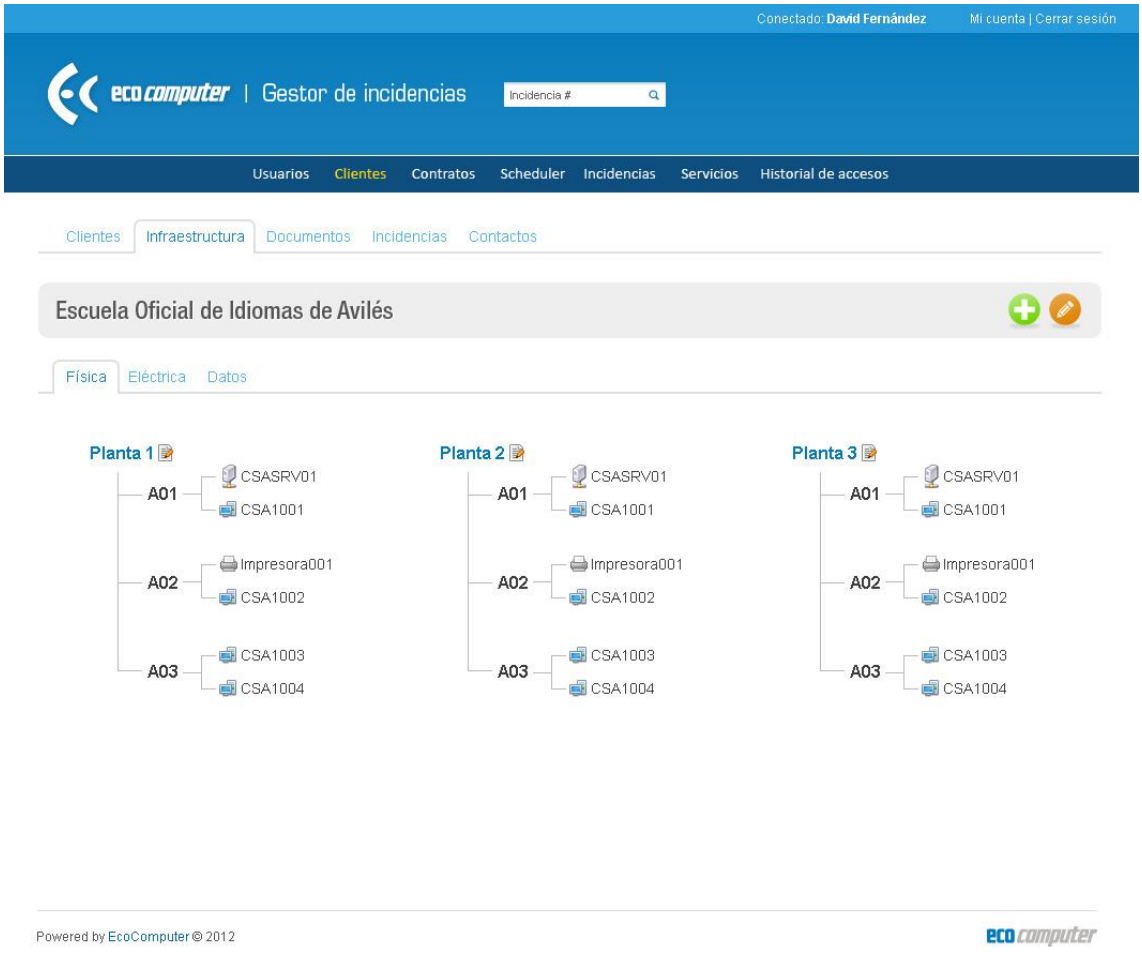


Figura 10.1. Esbozo de la interfaz de gestión de infraestructuras

10.2.2 Rol de Cliente en el Sistema

El objetivo de esta ampliación sería dotar de acceso al sistema a los clientes de la empresa, de modo que tuviera acceso a la información de sus contratos activos. Además, sería posible que el propio cliente, con acceso y permisos especiales en el sistema, comunicara las incidencias vía web, lo que mejoraría aún más los tiempos de respuesta y resolución.

10.2.3 Informes Estadísticos sobre el Tratamiento de Incidencias

Con el objetivo de generar informes de desempeño, sería interesante tener la posibilidad de generar un resumen de información estadística sobre el proceso de resolución de incidencias en la empresa. Para ello, tomando como base los datos de los que se dispone en la aplicación, sería necesario diseñar un conjunto de selectores que permitieran obtener informes personalizados de acuerdo a las variables a monitorizar, tales como estado de la incidencia, cliente, prioridad, tiempo medio de resolución, etc.

10.2.4 Sistema de Notificaciones

Con esta ampliación, el sistema sería capaz de enviar alertas sobre aquellas incidencias en las que los usuarios toman parte, bien por correo, notificaciones *push* e incluso, mediante Twitter, ya que la mayor parte de la plantilla de la empresa utiliza la red social. De este modo y mediante el uso de listas y cuentas privadas, los interesados pueden estar al tanto de cualquier evento (nueva incidencia asignada, actualización de una existente, cambios de estado, intervenciones requeridas, etc.) en tiempo real y sin la necesidad de un sistema dedicado, aprovechando así el elevado nivel de aceptación y uso de la red social.

10.2.5 Sistema Inteligente de Asignación de Incidencias

Una de las propuestas de la directiva de la empresa consistía en la creación de un sistema inteligente encargado de gestionar la asignación de incidencias a los empleados. Para ello, basándose en un aprendizaje autónomo por parte del sistema, de acuerdo a las incidencias ya resueltas, el propio sistema sería capaz de asignar las nuevas incidencias comunicadas al empleado correspondiente, tomando además en cuenta la carga de trabajo que éste tuviera en el día y/o semana en curso.

Al tratarse de un desarrollo de considerable complejidad, se estima conveniente abordarlo en un proyecto individual.

10.2.6 Aplicación para Dispositivos Móviles

Aunque fue uno de los requisitos más activamente comentados al inicio del proyecto, finalmente fue necesario prescindir de esta funcionalidad, debido a que el cliente no contaba con que era necesario haber desarrollado primero el sistema base.

A parte de esto, debido a la heterogeneidad de dispositivos móviles en la empresa, sería necesario desarrollar la aplicación para Android e iOS, por lo que, a pesar de ser una mejora interesante, supondría la necesidad de una cantidad considerable de recursos para su desarrollo, si bien es cierto que sería posible acometerlo sustituyendo el concepto de aplicación dedicada por una adaptación de la interfaz web para dispositivos móviles.

10.2.7 Creación de una Base de Conocimiento de Resolución de Incidencias

Como añadido al sistema, sería de utilidad desplegar un sistema de gestión de conocimiento, basado por ejemplo en un gestor documental ya existente. De este modo, podría crearse un repositorio interno de información con la información relativa a las incidencias ya resueltas, catalogándolas por tipo y sirviendo así de biblioteca de consulta de cara al tratamiento de futuras incidencias.

Capítulo 11. Presupuesto

A continuación se muestran de forma individual cada uno de los presupuestos establecidos para el proyecto.

11.1 Presupuesto del Cliente

Haciendo uso de la estimación de costes del desarrollo del proyecto y aplicando un margen de beneficio del 20%, el presupuesto del cliente es el mostrado en la siguiente tabla.

Para el cálculo de los precios unitarios, se ha tomado como referencia la tarificación oficial de EcoComputer, S.L., que aplica un precio fijo a cada tipo desarrollo.

Ítem	Subítem	Concepto	Total Concepto (€)
01		**Desarrollo de la Aplicación – Tipo: Desarrollo Web**	***21.350,00***
	001	Análisis	3.990,00
	002	Diseño	5.820,00
	003	Implementación	10.490,00
	004	Pruebas	1.050,00
		SUBTOTAL	21.350,00
		IVA (21%)	4.483,50
		TOTAL	***25.833,50***

11.2 Presupuesto de Costes

El siguiente presupuesto muestra el detalle de los costes de desarrollo del proyecto, dividiéndolo en secciones que agrupan conceptos del mismo tipo.

Ítem	Subítem	Concepto	Cantidad	Precio Unitario (€)	Total Concepto (€)
01		**Software**			***402,94***
	001	Sparx Systems Enterprise Architect	1	135,00	135,00
	002	Sistema Operativo Mac OS X 10.8	1	17,99	17,99
	003	Microsoft Office 2011	1	249,95	249,95
	004	Microsoft Windows Server 2008 (*)	1	0,00	0,00
	005	Microsoft SQL Server 2005 (*)	1	0,00	0,00
02		**Hardware**			***447,00***
	001	Amortización Equipos de Desarrollo	6	74,50	447,00
03		**Recursos Humanos**			***13.859,00***
	001	Director de Proyecto (Horas)	20	39,00	780,00
	002	Cuota Empresarial a la Seguridad Social	20	8,75	175
	003	Analista-Diseñador(Horas)	130	29,00	3.770,00
	004	Cuota Empresarial a la Seguridad Social	130	6,80	884,00
	005	Programador (Horas)	330	19,00	6.270,00
	006	Cuota Empresarial a la Seguridad Social	330	6,00	1.980,00
05		**Varios**			***1.761,30***
	001	Alquiler de Oficina	6	200,00	1200,00
	001	Material de Oficina	1	90,00	90,00
	002	Electricidad	6	12,95	77,70
	003	Conexión a Internet y Teléfono	6	65,60	393,60
				TOTAL	**16.470,24**

(*) Elementos proporcionados por el cliente

Capítulo 12. Referencias Bibliográficas

12.1 Libros y Artículos

[Brittain08]. Brittain, Jason; F. Darwin, Ian. "Tomcat: The Definitive Guide". O'Reilly Media. 2008. ISBN 059655494X.

[Brown08] Brown, Daniel; Michael D., Chad; Stanlick, Scott. "Struts 2 in Action". Willey India Pvt. Limited. 2008. ISBN 8177228757.

[Hunter01]. Hunter, Jason. "Java Servlet Programing". O'Reilly Media. 2001. ISBN 0596000405.

[Kogent08] Kogent Solutions Inc. "Struts 2 Black Book". Dreamtech Press. 2008. ISBN 8177228706.

[Newton09] Newton, Dave. "Apache Struts 2 Web Application Development". Packt Publishing Ltd. 2009. ISBN 1847193404.

[Reese00]. Reese, George. "Database Programming with JDBC & Java". O'Reilly Media. 2000. ISBN 1565926161.

[Silberschatz06]. Silberschatz, Abraham; F. Korth, Henry; Sudarshan, S. "Fundamentos de Bases de Datos". McGraw Hill. 2006. ISBN 8448146441.

12.2 Referencias en Internet

[Atwood13] Atwood, J. "Understanding Model-View-Controller" http://www.codinghorror.com/blog/2008/05/understanding-model-view-controller.html. 2008

[Bootstrap13] Twitter Bootstrap front-end. "Bootstrap – A sleek, intuitive and powerful front-end framework for faster and easier web development". 2013

[Conventions99] Oracle TechNetwork. "Code Conventions for the Java Prorgamming Language" http://www.oracle.com/technetwork/java/javase/documentation/codeconvtoc-136057.html. 1999

[CSS11] World Wide Web Consortium. "Cascading Style Sheets Level 2 Revision 1 (CSS 2.1) Specification". http://www.w3.org/TR/CSS2/. 2011

[DotProject07] DotProject. "Main Page – DotProject Wiki". http://docs.dotproject.net/. 2007

[Fella13] Fella, Allan; SpryMedia UI Development. "DataTables – Table plugin for jQuery". http://www.datatables.net/. 2013

[Geppert12] Geppert, J. Bootstrap Plugin for Struts 2. "Apache Struts 2 Plugin Registry – Bootstrap Plugin" https://cwiki.apache.org/S2PLUGINS/bootstrap-plugin.html. 2012

[Hassan13] Hassan Montero, Y. "Guía de Evaluación Heurística de Sitios Web". http://www.nosolousabilidad.com/articulos/heuristica.htm. 2013

[ITIL11] ITIL V3. "Fase 4. Operación de los servicios TI – Gestión de Incidencias" http://itilv3.osiatis.es/operacion_servicios_TI/gestion_incidencias.php. 2011

[JavaEE13] Oracle. Documentación Oficial de Java Enterprise Edition. "Java EE Reference". http://www.oracle.com/technetwork/java/javaee/documentation/index.html. 2013

[JavaSE13] Oracle TechNetwork. "JavaTM Platform, Standard Edition 7". http://docs.oracle.com/javase/7/docs/api/. 2013

[JDBC13] Oracle TechNetwork. "Java SE Technologies – Database Connectivity". http://www.oracle.com/technetwork/java/javase/jdbc/index.html. 2013

[JQuery13] jQuery Foundation. "jQuery – How jQuery Works". http://learn.jquery.com/about-jquery/how-jquery-works/. 2013

[JQUI13] jQuery Foundation. "jQuery UI – About jQuery UI". http://jqueryui.com/about/. 2013

[JSP13] Oracle. Documentación Oficial de JavaServer Pages. "JavaServer Pages Technology". http://www.oracle.com/technetwork/java/javaee/jsp/index.html. 2013

[JSTL13] Oracle TechNetwork. "JavaServer Pages Standard Tag Library". http://www.oracle.com/technetwork/java/index-jsp-135995.html. 2013

[Kuhn13] Kuhn, Dylan. GitHub. "ColumnFilterWidgets – An add-on for the DataTables plugin that creates filtering widgets based on the data in table columns". 2013

[Lotus08] IBM España. Presentación de características y demostraciones de Lotus Notes 8. "IBM Lotus Notes y Domino – España". http://www-01.ibm.com/software/es/lotus/wdocs/notes-domino8/. 2008

[Mantis13] MantisBT. "Mantis Feature List". http://www.mantisbt.org/wiki/doku.php/mantisbt:features. 2013

[MDNJS13] Mozilla Developer Network "Web technology for developers". https://developer.mozilla.org/en-US/docs/Web. 2013

[METRICA13] Ministerio de Hacienda y Administraciones Públicas – Gobierno de España. Documentación de la metodología MÉTRICA VERSIÓN 3. "MÉTRICA V.3 – Portal de Administración Electrónica". http://administracionelectronica.gob.es/?_nfpb=true&_pageLabel=P80029225129365155099 1&langPae=es&detalleLista=PAE_000000432. 2013

[MSBOOKS08] MSDN. "Microsoft SQL Server 2005 Books" http://msdn.microsoft.com/en-us/library/ms130214(SQL.90).aspx. 2008

[MSDEV13] Microsoft SQL Server Developer Center. "Microsoft JDBC Driver for SQL Server". http://msdn.microsoft.com/en-us/sqlserver/aa937724.aspx. 2013

[OGNL13] Apache Software Foundation. "OGNL – Apache Commons". http://commons.apache.org/proper/commons-ognl/. 2013

[OMG13] Object Management Group. "Resources – UML". http://www.uml.org/. 2013

[Pandora13] Ministerio de Hacienda y Administraciones Públicas – Gobierno de España. Descripción del proyecto Pandora. "Gestión de Incidencias Informáticas de la Secretaría de Estado de Administraciones Públicas – Portal de Administración Electrónica". http://administracionelectronica.gob.es/?_nfpb=true&_pageLabel=PAE_PG_CTT_General&langPae=es&iniciativa=151. 2013

[ProjectOpen13] ProjectOpen. "Project-Open – In a nutshell". http://www.project-open.com/en/products/project-open-in-a-nutshell.html. 2013

[Redmine13] Redmine. "Overview – Redmine". http://www.redmine.org/. 2013

[Sparx13] Sparx Systems. "UML Tutorial". http://www.sparxsystems.com/uml-tutorial.html. 2013

[Spring13] Pivotal – SpringSource. "Spring Framework Feature Tour" http://www.springsource.org/features. 2013

[Struts13] Apache Software Foundation. "Apache Struts 2 Documentation" http://struts.apache.org/development/2.x/. 2013

[Tiles13] Apache Software Foundation. "Tiles – Apache". http://tiles.apache.org/. 2013

[Tomcat13] The Apache Software Foundation. "Apache Tomcat". http://tomcat.apache.org/. 2013

[TPoint13] TutorialsPoint. "Java Server Pages Tutorial" http://www.tutorialspoint.com/jsp/index.htm. 2013

[W3C10] World Wide Web Consortium. "XHTML™ 1.1 - Module-based XHTML - Second Edition". http://www.w3.org/MarkUp/. 2010

[WCAG08] World Wide Web Consortium. "Web Content Accesibility Guidelines (WCAG) 2.0" http://www.w3.org/TR/WCAG20/. 2008

Capítulo 13. Apéndices

13.1 Glosario y Diccionario de Datos

- **Apache Struts**: Herramienta de soporte para el desarrollo de aplicaciones web bajo el patrón MVC en la plataforma Java EE.
- **Aplicación Web**: En Ingeniería de Software, se denomina aplicación web a aquellas herramientas que los usuarios pueden utilizar accediendo a un servidor web a través de Internet o de una intranet mediante un navegador web.
- **Cliente**: Persona natural o jurídica que realiza una transacción comercial con un proveedor de bienes o servicios.
- **Contrato**: Acuerdo vinculante establecido entre dos o más personas que regula sus relaciones relativas a una determinada finalidad.
- **CSS:** Acrónimo de Cascading Style Sheets.
- **Diagrama de Gantt:** Gráfico de barras, desarrollado por Henry Gantt en la década de 1910, que ilustra la planificación de un proyecto.
- **Facturación:** En un contrato, la facturación aplicada al mismo define la periodicidad con la que el proveedor de servicios cobrará por la prestación de éstos al cliente.
- **Incidencia**: Cualquier evento que no forma parte del desarrollo habitual de un servicio y que causa, o puede causar, una interrupción o reducción de la calidad del mismo.
- **Intervención:** En la gestión de incidencias, una intervención es una actuación llevada a cabo con el objetivo final de resolver una incidencia activa.
- **JAR:** También denominado librería, es un contenedor de ficheros *.class* de Java, con sus metadatos y recursos asociados.
- **MVC:** Acrónimo de Model-View-Controller.
- **Patrón de diseño de software:** En Ingeniería de Software, se denomina patrón de diseño a una solución genérica aplicable a un problema común en el contexto del diseño de software.
- **Servicio**: Conjunto de actividades que pretenden responder a las necesidades de un cliente.
- **Sesión [de usuario]:** Proceso de intercambio de información entre dos o más sistemas, o bien, entre el usuario y el sistema que manipula. Adicionalmente, una sesión de login se define como el periodo de actividad comprendido entre el comienzo y la finalización de la interacción entre un usuario y un sistema.
- **SGBD:** Acrónimo de Sistema de Gestión de Bases de Datos.
- **SIGICCS:** Acrónimo de Sistema Integrado para la Gestión de Incidencias, Clientes y contratos vía Web.
- **Struts Action:** Petición originada por el cliente de la aplicación desde la interfaz de ésta, que será enviada al controlador para su tratamiento y generación de una respuesta acorde.
- **Tarifa**: Precio que pagan los clientes por la prestación de servicios. En este contexto, una tarifa especifica el precio a cobrar, en euros, por hora-persona trabajada.

- **Twitter Bootstrap**: Colección de herramientas de software libre desarrolladas por Twitter, Inc. enfocadas a la creación de sitios y aplicaciones web compuesta por plantillas de diseño basadas en HTML y CSS.
- **WCAG:** Acrónimo de Web Content Accessibility Guidelines.
- **XHTML:** Acrónimo de eXtensible Hyper-Text Markup Language.

13.2 Contenido Entregado en el CD-ROM

Directorio	Contenido
./ Directorio raíz del CD	Contiene un fichero leeme.txt explicando esta estructura
./SIGICCS	Contiene toda la estructura de directorios del proyecto para desarrollo
./instalacion	Ficheros utilizados para la instalación del proyecto
./documentacion	Contiene toda la documentación asociada al proyecto
./documentacion/img	Directorio que contiene las imágenes utilizadas en la documentación
./documentacion/plan	Contiene el fichero con la planificación del proyecto, el extracto de tareas y el diagrama de Gantt
./documentacion/uml	Ficheros que genera la herramienta con la que se han creado los diagramas UML y de entidad relación
./herramientas	Contiene los ficheros de instalación de las herramientas utilizadas para el desarrollo o puesta en marcha del proyecto
./herramientas/desarrollo	Ficheros de instalación de las herramientas utilizadas en el desarrollo
./herramientas/explotacion	Base de datos, servidor web y herramientas en general

Directorio	Contenido
./ Directorio raíz de "desarrollo"	Contiene los ficheros de proyecto del IDE utilizado.
./build	Contiene los ficheros Java con las clases compiladas
./dist	Contiene el paquete distribuible del proyecto en formato WAR
./doc	Contiene toda la documentación relativa al proyecto generada por Javadoc
./src	Ficheros de código fuente Java y XML de configuración de Struts
./src/sql	Este directorio contiene los scripts de *SQL* que permiten construir e insertar los datos iniciales en la base de datos del proyecto
./WebContent	Este directorio contiene los ficheros que componen la web de la aplicación, además de los ficheros XML de configuración de Spring, Tiles y Struts
./WebContent/css	Contiene las hojas de estilo de la web
./WebContent/images	Contiene las imágenes utilizadas por los ficheros de la web del proyecto.
./WebContent/js	Directorio donde se almacenan los ficheros

	JavaScript utilizados por las JSP
./WebContent/WEB-INF	Almacena todos los elementos de la web visibles únicamente bajo autenticación en el sistema
./WebContent/WEB-INF/lib	Bibliotecas externas JAR necesarias para compilar y distribuir, de las que depende este proyecto.
./WebContent/WEB-INF/pages	Ficheros JSP de la aplicación
./WebContent/WEB-INF/public	Contiene el fichero JSP con la pantalla de bienvenida de la aplicación
./WebContent/WEB-INF/tiles	Plantillas JSP de la capa de tiles de la aplicación

13.3 Descripción Detallada de las Clases

13.3.1.1 Clase ClienteManager

org.sigiccs.impl.cliente.business
java.lang.Object
└ **org.sigiccs.impl.cliente.business.ClienteManager**

All Implemented Interfaces:

ClienteManagerService

---

public class **ClienteManager**
extends Object
implements ClienteManagerService

Clase encargada de establecer la conexión entre la invocación de acciones propias de la gestión de clientes en la capa de presentación y la de persistencia

Constructor Detail

ClienteManager

public **ClienteManager**()

Method Detail

setClienteDataService

public void **setClienteDataService**(ClienteDataService clienteDataService)

Parameters:

clienteDataService - instancia del servicio de acceso a persistencia para las entidades Cliente

---

getAllClientes

public Vector<Cliente> **getAllClientes**()
throws SQLException

Description copied from interface: ClienteManagerService

Busca todos los clientes en el sistema

Specified by:

getAllClientes in interface ClienteManagerService

Returns:

Un objeto Vector con todos los objetos Cliente encontrados

Throws:

SQLException

---

getClienteById

public Cliente **getClienteById**(int ID)
throws SQLException

Description copied from interface: ClienteManagerService

Busca toda la información de un cliente en particular

Specified by:

`getClienteById` in interface `ClienteManagerService`

Parameters:

`ID` - El Identificador del cliente a solicitar

Returns:

Un objeto Cliente con todos los datos del mismo

Throws:

`SQLException`

---

getClienteByNombre

```
public Cliente getClienteByNombre(String nombre)
                           throws SQLException
```

Description copied from interface: `ClienteManagerService`

Busca toda la información de un cliente en base a su nombre

Specified by:

`getClienteByNombre` in interface `ClienteManagerService`

Parameters:

`nombre` - Nombre del cliente a buscar

Returns:

Un objeto Cliente con todos los datos del mismo

Throws:

`SQLException`

---

almacenarNuevoCliente

```
public int almacenarNuevoCliente(Cliente cliente)
                          throws SQLException
```

Description copied from interface: `ClienteManagerService`

Almacena un nuevo cliente en el sistema

Specified by:

`almacenarNuevoCliente` in interface `ClienteManagerService`

Parameters:

`cliente` - El objeto Cliente con todos sus datos

Returns:

1 si el cliente fue insertado correctamente

Throws:

`SQLException`

---

actualizarCliente

```
public int actualizarCliente(Cliente cliente)
                      throws SQLException
```

Description copied from interface: `ClienteManagerService`

Actualiza los datos de un cliente ya existente en el sistema

Specified by:

`actualizarCliente` in interface `ClienteManagerService`

Parameters:

`cliente` - El objeto Cliente con los nuevos datos

Returns:

1 si el cliente fue actualizado correctamente

Throws:

`SQLException`

removeCliente

```
public int removeCliente(int idCliente)
                  throws SQLException
```

Description copied from interface: ClienteManagerService

Marca como eliminado un cliente existente

Specified by:

`removeCliente` in interface `ClienteManagerService`

Parameters:

`idCliente` - Identificador del cliente a eliminar

Returns:

1 si el cliente fue eliminado correctamente

Throws:

`SQLException`

getProvinciasCliente

```
public Vector<Provincia> getProvinciasCliente()
                                        throws SQLException
```

Description copied from interface: ClienteManagerService

Método de utilidad para consultar el listado de provincias para los formularios de los clientes

Specified by:

`getProvinciasCliente` in interface `ClienteManagerService`

Returns:

Un objeto Vector con todos los objetos Provincia encontrados

Throws:

`SQLException`

13.3.1.2 Clase ClienteDAO

org.sigiccs.impl.cliente.persistence
java.lang.Object
  └ **org.sigiccs.impl.cliente.persistence.ClienteDAO**

All Implemented Interfaces:

ClienteDataService

---

public class **ClienteDAO**
extends Object
implements ClienteDataService

Clase encargada de la implementación de todos los métodos declarados en la interfaz que implementa, encargados de efectuar las operaciones de la gestión de clientes en lo relativo a la consulta y modificación de datos

Constructor Detail

ClienteDAO

public **ClienteDAO**()

Method Detail

getAllClientes

public Vector<Cliente> **getAllClientes**()
throws SQLException

Description copied from interface: ClienteDataService

Busca todos los clientes en el sistema

Specified by:

getAllClientes in interface ClienteDataService

Returns:

Un objeto Vector con todos los objetos Cliente encontrados

Throws:

SQLException

---

getClienteById

public Cliente **getClienteById**(int iDCliente)
throws SQLException

Description copied from interface: ClienteDataService

Busca toda la información de un cliente en particular

Specified by:

getClienteById in interface ClienteDataService

Returns:

Un objeto Cliente con todos los datos del mismo

Throws:

SQLException

---

getClienteByNombre

```
public Cliente getClienteByNombre(String nombre)
                           throws SQLException
```

Description copied from interface: ClienteDataService

Busca toda la información de un cliente en base a su nombre

Specified by:

`getClienteByNombre` in interface `ClienteDataService`

Parameters:

`nombre` - Nombre del cliente a buscar

Returns:

Un objeto Cliente con todos los datos del mismo

Throws:

`SQLException`

---

almacenarNuevoCliente

```
public int almacenarNuevoCliente(Cliente cliente)
                          throws SQLException
```

Description copied from interface: ClienteDataService

Almacena un nuevo cliente en el sistema

Specified by:

`almacenarNuevoCliente` in interface `ClienteDataService`

Parameters:

`cliente` - El objeto Cliente con todos sus datos

Returns:

1 si el cliente fue insertado correctamente

Throws:

`SQLException`

---

actualizarCliente

```
public int actualizarCliente(Cliente cliente)
                      throws SQLException
```

Description copied from interface: ClienteDataService

Actualiza los datos de un cliente ya existente en el sistema

Specified by:

`actualizarCliente` in interface `ClienteDataService`

Parameters:

`cliente` - El objeto Cliente con los nuevos datos

Returns:

1 si el cliente fue actualizado correctamente

Throws:

`SQLException`

---

removeCliente

```
public int removeCliente(int idCliente)
                  throws SQLException
```

Description copied from interface: ClienteDataService

Marca como eliminado un cliente existente

Specified by:

removeCliente in interface ClienteDataService

Parameters:

idCliente - Identificador del cliente a eliminar

Returns:

1 si el cliente fue eliminado correctamente

Throws:

SQLException

---

getProvinciasCliente

```
public Vector<Provincia> getProvinciasCliente()
                                        throws SQLException
```

Description copied from interface: ClienteDataService

Método de utilidad para consultar el listado de provincias para los formularios de los clientes

Specified by:

getProvinciasCliente in interface ClienteDataService

Returns:

Un objeto Vector con todos los objetos Provincia encontrados

Throws:

SQLException

13.3.1.3 Clase ActualizarClienteAction

org.sigiccs.impl.cliente.presentation

```
java.lang.Object
  └com.opensymphony.xwork2.ActionSupport
      └org.sigiccs.impl.cliente.presentation.ActualizarClienteAction
```

All Implemented Interfaces:

com.opensymphony.xwork2.Action, com.opensymphony.xwork2.LocaleProvider, Serializable, org.apache.struts2.interceptor.ServletRequestAware, com.opensymphony.xwork2.TextProvider, com.opensymphony.xwork2.Validateable, com.opensymphony.xwork2.ValidationAware

```
public class ActualizarClienteAction
extends com.opensymphony.xwork2.ActionSupport
implements org.apache.struts2.interceptor.ServletRequestAware
```

Action encargado de recoger los nuevos datos de un cliente existente e invocar a la siguiente capa de la arquitectura para hacerlos persistentes

13.3.1.4 Clase CargarClienteAction

org.sigiccs.impl.cliente.presentation

```
java.lang.Object
  └com.opensymphony.xwork2.ActionSupport
      └org.sigiccs.impl.cliente.presentation.CargarClienteAction
```

All Implemented Interfaces:

com.opensymphony.xwork2.Action, com.opensymphony.xwork2.LocaleProvider, Serializable, com.opensymphony.xwork2.TextProvider, com.opensymphony.xwork2.Validateable, com.opensymphony.xwork2.ValidationAware

```
public class CargarClienteAction
extends com.opensymphony.xwork2.ActionSupport
```

Action encargado de consultar los datos de un cliente y pasarlos a la capa de presentación

13.3.1.5 Clase CargarProvinciasClienteAction

org.sigiccs.impl.cliente.presentation

```
java.lang.Object
  └com.opensymphony.xwork2.ActionSupport
      └org.sigiccs.impl.cliente.presentation.CargarProvinciasClienteAction
```

All Implemented Interfaces:

com.opensymphony.xwork2.Action, com.opensymphony.xwork2.LocaleProvider, Serializable, com.opensymphony.xwork2.TextProvider, com.opensymphony.xwork2.Validateable, com.opensymphony.xwork2.ValidationAware

```
public class CargarProvinciasClienteAction
extends com.opensymphony.xwork2.ActionSupport
```

Action encargado de cargar las provincias de España para presentarlas en los formularios de los clientes

13.3.1.6 Clase CargarTodosClientesAction

org.sigiccs.impl.cliente.presentation

```
java.lang.Object
  └com.opensymphony.xwork2.ActionSupport
      └org.sigiccs.impl.cliente.presentation.CargarTodosClientesAction
```

All Implemented Interfaces:

com.opensymphony.xwork2.Action, com.opensymphony.xwork2.LocaleProvider, Serializable, org.apache.struts2.interceptor.SessionAware, com.opensymphony.xwork2.TextProvider, com.opensymphony.xwork2.Validateable, com.opensymphony.xwork2.ValidationAware

```
public class CargarTodosClientesAction
extends com.opensymphony.xwork2.ActionSupport
implements org.apache.struts2.interceptor.SessionAware
```

Action encargado de solicitar a la capa de persistencia el listado de clientes del sistema

13.3.1.7 Clase CrearClienteAction

org.sigiccs.impl.cliente.presentation

```
java.lang.Object
  └com.opensymphony.xwork2.ActionSupport
      └org.sigiccs.impl.cliente.presentation.CrearClienteAction
```

All Implemented Interfaces:

com.opensymphony.xwork2.Action, com.opensymphony.xwork2.LocaleProvider, Serializable, org.apache.struts2.interceptor.ServletRequestAware, com.opensymphony.xwork2.TextProvider, com.opensymphony.xwork2.Validateable, com.opensymphony.xwork2.ValidationAware

```
public class CrearClienteAction
extends com.opensymphony.xwork2.ActionSupport
implements org.apache.struts2.interceptor.ServletRequestAware
```

Action encargado de recoger los datos de un nuevo cliente y hacerlos persistentes

13.3.1.8 Clase CustomImageBytesResult

org.sigiccs.impl.cliente.presentation

```
java.lang.Object
  └org.sigiccs.impl.cliente.presentation.CustomImageBytesResult
```

All Implemented Interfaces:

com.opensymphony.xwork2.Result, Serializable

```
public class CustomImageBytesResult
extends Object
implements com.opensymphony.xwork2.Result
```

Clase de utilidad para consultar el contenido de una imagen

13.3.1.9 Clase EliminarClienteAction

org.sigiccs.impl.cliente.presentation

```
java.lang.Object
  └com.opensymphony.xwork2.ActionSupport
      └org.sigiccs.impl.cliente.presentation.EliminarClienteAction
```

All Implemented Interfaces:

com.opensymphony.xwork2.Action, com.opensymphony.xwork2.LocaleProvider, Serializable, com.opensymphony.xwork2.TextProvider, com.opensymphony.xwork2.Validateable, com.opensymphony.xwork2.ValidationAware

```
public class EliminarClienteAction
extends com.opensymphony.xwork2.ActionSupport
```

Action encargado de invocar la eliminacion de un cliente (marcado)

13.3.1.10 Clase ImageAction

org.sigiccs.impl.cliente.presentation

```
java.lang.Object
  └com.opensymphony.xwork2.ActionSupport
      └org.sigiccs.impl.cliente.presentation.ImageAction
```

All Implemented Interfaces:

com.opensymphony.xwork2.Action, com.opensymphony.xwork2.LocaleProvider, Serializable, org.apache.struts2.interceptor.ServletRequestAware, com.opensymphony.xwork2.TextProvider, com.opensymphony.xwork2.Validateable, com.opensymphony.xwork2.ValidationAware

```
public class ImageAction
extends com.opensymphony.xwork2.ActionSupport
implements org.apache.struts2.interceptor.ServletRequestAware
```

Action encargado de cargar el logotipo de un cliente

13.3.1.11 Clase ContratoManager

org.sigiccs.impl.contrato.business
java.lang.Object
└ **org.sigiccs.impl.contrato.business.ContratoManager**

All Implemented Interfaces:
ContratoManagerService

---

public class **ContratoManager**
extends Object
implements ContratoManagerService

Clase encargada de establecer la conexión entre la invocación de acciones propias de la gestión de contratos en la capa de presentación y la de persistencia

Constructor Detail

ContratoManager

public **ContratoManager**()

Method Detail

setContratoDataService

public void **setContratoDataService**(ContratoDataService contratoDataService)

Parameters:

contratoDataService - instancia del servicio de acceso a persistencia para las entidades Contrato

---

getAllContratos

public Vector<Contrato> **getAllContratos**()
throws SQLException

Description copied from interface: ContratoManagerService

Busca todos los contratos suscritos por clientes que han sido almacenados en el sistema

Specified by:

getAllContratos in interface ContratoManagerService

Returns:

Un objeto Vector con todos los objetos Contrato encontrados

Throws:

SQLException

---

getContratoById

public Contrato **getContratoById**(int ID)
throws SQLException

Description copied from interface: ContratoManagerService

Busca toda la información de un contrato en particular

Specified by:

getContratoById in interface ContratoManagerService

Parameters:

`ID` - El Identificador del contrato a solicitar

Returns:

Un objeto Contrato con todos los datos del mismo

Throws:

`SQLException`

---

almacenarNuevoContrato

```
public int almacenarNuevoContrato(Contrato contrato)
                           throws SQLException
```

Description copied from interface: ContratoManagerService

Almacena un nuevo contrato en el sistema

Specified by:

`almacenarNuevoContrato` in interface `ContratoManagerService`

Parameters:

`contrato` - El objeto Contrato con todos sus datos

Returns:

1 si el contrato fue insertado correctamente

Throws:

`SQLException`

---

actualizarContrato

```
public int actualizarContrato(Contrato contrato)
                       throws SQLException
```

Description copied from interface: ContratoManagerService

Actualiza los datos de un contrato ya existente en el sistema

Specified by:

`actualizarContrato` in interface `ContratoManagerService`

Parameters:

`contrato` - El objeto Contrato con los nuevos datos

Returns:

1 si el contrato fue actualizado correctamente

Throws:

`SQLException`

---

removeContrato

```
public int removeContrato(int idContrato)
                   throws SQLException
```

Description copied from interface: ContratoManagerService

Marca como eliminado un contrato existente

Specified by:

`removeContrato` in interface `ContratoManagerService`

Returns:

1 si el contrato fue eliminado correctamente

Throws:

`SQLException`

---

getTiposFacturacion

```
public Vector<TipoFacturacion> getTiposFacturacion()
                                              throws SQLException
```

Description copied from interface: ContratoManagerService

Consulta el listado de Tipos de Facturacion aplicables a los contratos

Specified by:

getTiposFacturacion in interface ContratoManagerService

Returns:

Un objeto Vector con todos los Tipos de Facturacion disponibles en el sistema

Throws:

SQLException

getContratosCliente

```
public Vector<Contrato> getContratosCliente(int iDCliente)
                                      throws SQLException
```

Description copied from interface: ContratoManagerService

Devuelve todos los contratos suscritos por un cliente en particular

Specified by:

getContratosCliente in interface ContratoManagerService

Parameters:

iDCliente - Identificador del cliente en base al cual se realiza la consulta de contratos suscritos

Returns:

Un objeto Vector con todos los contratos suscritos por el cliente

Throws:

SQLException

13.3.1.12 Clase ContratoServiciosManager

org.sigiccs.impl.contrato.business
java.lang.Object
└ **org.sigiccs.impl.contrato.business.ContratoServiciosManager**

All Implemented Interfaces:

ContratoServiciosManagerService

---

public class **ContratoServiciosManager**
extends Object
implements ContratoServiciosManagerService

Clase encargada de establecer la conexión entre la invocación de acciones propias de la asociación de servicios a contratos en la capa de presentación y la de persistencia

Constructor Detail

ContratoServiciosManager

public **ContratoServiciosManager**()

Method Detail

setContratoServiciosDataService

public void **setContratoServiciosDataService**(ContratoServiciosDataService contratoServiciosDataService)

Parameters:

contratoServiciosDataService - instancia del servicio de acceso a persistencia para las entidades ContratoServicios

---

getAllServiciosContrato

public Vector<ContratoServicio> **getAllServiciosContrato**(int idContrato)
throws SQLException

Description copied from interface: ContratoServiciosManagerService

Busca todos los servicios asociados a un contrato suscrito por un cliente

Specified by:

getAllServiciosContrato in interface ContratoServiciosManagerService

Parameters:

idContrato - Identificador del contrato a consultar

Returns:

Un objeto Vector con todos los objetos ContratoServicio, que representan los servicios contratados por el cliente en ese contrato

Throws:

SQLException

---

almacenarNuevoServicioContrato

public int **almacenarNuevoServicioContrato**(ContratoServicio contratoServicio)
throws SQLException

Description copied from interface: ContratoServiciosManagerService

Asocia un servicio del catálogo a un contrato de servicios

Specified by:

almacenarNuevoServicioContrato in interface ContratoServiciosManagerService

Parameters:

`contratoServicio` - El objeto Servicio contratable

Returns:

1 si el servicio fue asociado correctamente al contrato

Throws:

`SQLException`

---

actualizarServicioContrato

```
public int actualizarServicioContrato(ContratoServicio contratoServicio)
                               throws SQLException
```

Description copied from interface: ContratoServiciosManagerService

Actualiza los parámetros de una suscripción de servicio en un contrato

Specified by:

actualizarServicioContrato in interface ContratoServiciosManagerService

Parameters:

`contratoServicio` - El objeto Servicio asociado al contrato

Returns:

1 si la asociación del servicio se actualizó correctamente

Throws:

`SQLException`

---

removeServicioContrato

```
public int removeServicioContrato(int idContrato,
                                  int idServicio)
                           throws SQLException
```

Description copied from interface: ContratoServiciosManagerService

Marca como eliminado un servicio asociado a un contrato

Specified by:

removeServicioContrato in interface ContratoServiciosManagerService

Parameters:

`idContrato` - Identificador del contrato que contiene el servicio

`idServicio` - Identificador del servicio asociado a eliminar

Returns:

1 si el servicio se desvinculó del contrato correctamente

Throws:

`SQLException`

---

getServicioContrato

```
public ContratoServicio getServicioContrato(int iDContrato,
                                            int iDServicio)
                                     throws SQLException
```

Description copied from interface: ContratoServiciosManagerService

Busca toda la información de una asociación de servicio a contrato en particular

Specified by:

`getServicioContrato` in interface `ContratoServiciosManagerService`

Parameters:

`iDContrato` - Identificador del contrato a consultar

`iDServicio` - Identificador del servicio dentro del contrato

Returns:

Un objeto ContratoServicio con la información del mismo

Throws:

`SQLException`

13.3.1.13 Clase ContratoDAO

org.sigiccs.impl.contrato.persistence
java.lang.Object
└ **org.sigiccs.impl.contrato.persistence.ContratoDAO**

All Implemented Interfaces:

ContratoDataService

---

public class **ContratoDAO**
extends Object
implements ContratoDataService

Clase encargada de la implementación de todos los métodos declarados en la interfaz que implementa, encargados de efectuar las operaciones de la gestión de contratos en lo relativo a la consulta y modificación de datos

Constructor Detail

ContratoDAO

public **ContratoDAO**()

Method Detail

getAllContratos

public Vector<Contrato> **getAllContratos**()
throws SQLException

Description copied from interface: ContratoDataService

Busca todos los contratos suscritos por clientes que han sido almacenados en el sistema

Specified by:

getAllContratos in interface ContratoDataService

Returns:

Un objeto Vector con todos los objetos Contrato encontrados

Throws:

SQLException

---

getContratoById

public Contrato **getContratoById**(int ID)
throws SQLException

Description copied from interface: ContratoDataService

Busca toda la información de un contrato en particular

Specified by:

getContratoById in interface ContratoDataService

Parameters:

ID - El Identificador del contrato a solicitar

Returns:

Un objeto Contrato con todos los datos del mismo

Throws:

SQLException

---

getContratosCliente

```
public Vector<Contrato> getContratosCliente(int iDCliente)
                                    throws SQLException
```

Description copied from interface: ContratoDataService

Devuelve todos los contratos suscritos por un cliente en particular

Specified by:

getContratosCliente in interface ContratoDataService

Parameters:

iDCliente - Identificador del cliente en base al cual se realiza la consulta de contratos suscritos

Returns:

Un objeto Vector con todos los contratos suscritos por el cliente

Throws:

SQLException

---

almacenarNuevoContrato

```
public int almacenarNuevoContrato(Contrato contrato)
                           throws SQLException
```

Description copied from interface: ContratoDataService

Almacena un nuevo contrato en el sistema

Specified by:

almacenarNuevoContrato in interface ContratoDataService

Parameters:

contrato - El objeto Contrato con todos sus datos

Returns:

1 si el contrato fue insertado correctamente

Throws:

SQLException

---

actualizarContrato

```
public int actualizarContrato(Contrato contrato)
                       throws SQLException
```

Description copied from interface: ContratoDataService

Actualiza los datos de un contrato ya existente en el sistema

Specified by:

actualizarContrato in interface ContratoDataService

Parameters:

contrato - El objeto Contrato con los nuevos datos

Returns:

1 si el contrato fue actualizado correctamente

Throws:

SQLException

---

removeContrato

```
public int removeContrato(int idContrato)
                   throws SQLException
```

Description copied from interface: ContratoDataService

Marca como eliminado un cliente existente

Specified by:

removeContrato in interface ContratoDataService

Returns:

1 si el contrato fue eliminado correctamente

Throws:

SQLException

getTiposFacturacion

```
public Vector<TipoFacturacion> getTiposFacturacion()
                                              throws SQLException
```

Description copied from interface: ContratoDataService

Consulta el listado de Tipos de Facturacion aplicables a los contratos

Specified by:

getTiposFacturacion in interface ContratoDataService

Returns:

Un objeto Vector con todos los Tipos de Facturacion disponibles en el sistema

Throws:

SQLException

13.3.1.14 Clase ContratoServiciosDAO

org.sigiccs.impl.contrato.persistence
java.lang.Object
└ **org.sigiccs.impl.contrato.persistence.ContratoServiciosDAO**

All Implemented Interfaces:

ContratoServiciosDataService

---

public class **ContratoServiciosDAO**
extends Object
implements ContratoServiciosDataService

Clase encargada de la implementación de todos los métodos declarados en la interfaz que implementa, encargados de efectuar las operaciones de la gestión de asociaciones de servicios a contratos en lo relativo a la consulta y modificación de datos

Constructor Detail

ContratoServiciosDAO

public **ContratoServiciosDAO**()

Method Detail

getAllServiciosContrato

public Vector<ContratoServicio> **getAllServiciosContrato**(int idContrato)
throws SQLException

Description copied from interface: ContratoServiciosDataService

Busca todos los servicios asociados a un contrato suscrito por un cliente

Specified by:

getAllServiciosContrato in interface ContratoServiciosDataService

Parameters:

idContrato - Identificador del contrato a consultar

Returns:

Un objeto Vector con todos los objetos ContratoServicio, que representan los servicios contratados por el cliente en ese contrato

Throws:

SQLException

---

almacenarNuevoServicioContrato

public int **almacenarNuevoServicioContrato**(ContratoServicio contratoServicio)
throws SQLException

Description copied from interface: ContratoServiciosDataService

Asocia un servicio del catálogo a un contrato de servicios

Specified by:

almacenarNuevoServicioContrato in interface ContratoServiciosDataService

Parameters:

contratoServicio - El objeto Servicio contratable

Returns:

1 si el servicio fue asociado correctamente al contrato

Throws:

`SQLException`

---

actualizarServicioContrato

```
public int actualizarServicioContrato(ContratoServicio contratoServicio)
                                throws SQLException
```

Description copied from interface: ContratoServiciosDataService

Actualiza los parámetros de una suscripción de servicio en un contrato

Specified by:

`actualizarServicioContrato` in interface `ContratoServiciosDataService`

Parameters:

`contratoServicio` - El objeto Servicio asociado al contrato

Returns:

1 si la asociación del servicio se actualizó correctamente

Throws:

`SQLException`

---

removeServicioContrato

```
public int removeServicioContrato(int IDContrato,
                                  int IDServicio)
                           throws SQLException
```

Description copied from interface: ContratoServiciosDataService

Marca como eliminado un servicio asociado a un contrato

Specified by:

`removeServicioContrato` in interface `ContratoServiciosDataService`

Returns:

1 si el servicio se desvinculó del contrato correctamente

Throws:

`SQLException`

---

getServicioContrato

```
public ContratoServicio getServicioContrato(int iDContrato,
                                            int iDServicio)
                                     throws SQLException
```

Description copied from interface: ContratoServiciosDataService

Busca toda la información de una asociación de servicio a contrato en particular

Specified by:

`getServicioContrato` in interface `ContratoServiciosDataService`

Parameters:

`iDContrato` - Identificador del contrato a consultar

`iDServicio` - Identificador del servicio dentro del contrato

Returns:

Un objeto ContratoServicio con la información del mismo

Throws:

`SQLException`

13.3.1.15 Clase ActualizarContratoAction

org.sigiccs.impl.contrato.presentation

```
java.lang.Object
  └com.opensymphony.xwork2.ActionSupport
      └org.sigiccs.impl.contrato.presentation.ActualizarContratoAction
```

All Implemented Interfaces:

com.opensymphony.xwork2.Action, com.opensymphony.xwork2.LocaleProvider, Serializable, com.opensymphony.xwork2.TextProvider, com.opensymphony.xwork2.Validateable, com.opensymphony.xwork2.ValidationAware

---

```
public class ActualizarContratoAction
extends com.opensymphony.xwork2.ActionSupport
```

Action encargado de recoger los cambios en las propiedades de un contrato e invocar a la siguiente capa para persistirlos

13.3.1.16 Clase ActualizarServicioContratoAction

org.sigiccs.impl.contrato.presentation

```
java.lang.Object
  └com.opensymphony.xwork2.ActionSupport
      └
org.sigiccs.impl.contrato.presentation.ActualizarServicioContratoAction
```

All Implemented Interfaces:

com.opensymphony.xwork2.Action, com.opensymphony.xwork2.LocaleProvider, Serializable, com.opensymphony.xwork2.TextProvider, com.opensymphony.xwork2.Validateable, com.opensymphony.xwork2.ValidationAware

---

```
public class ActualizarServicioContratoAction
extends com.opensymphony.xwork2.ActionSupport
```

Action encargado de recoger los cambios en las propiedades de un servicio asociado a un contrato e invocar a la siguiente capa para persistirlos

13.3.1.17 Clase AsociarServicioAContratoAction

org.sigiccs.impl.contrato.presentation

```
java.lang.Object
  └com.opensymphony.xwork2.ActionSupport
      └org.sigiccs.impl.contrato.presentation.AsociarServicioAContratoAction
```

All Implemented Interfaces:

com.opensymphony.xwork2.Action, com.opensymphony.xwork2.LocaleProvider, Serializable, com.opensymphony.xwork2.TextProvider, com.opensymphony.xwork2.Validateable, com.opensymphony.xwork2.ValidationAware

---

```
public class AsociarServicioAContratoAction
extends com.opensymphony.xwork2.ActionSupport
```

Action encargado de recoger los datos del formulario de suscripción de servicio en un contrato e invocar a la siguiente capa para persistirlos

13.3.1.18 Clase CargarContratoAction

org.sigiccs.impl.contrato.presentation

```
java.lang.Object
  └com.opensymphony.xwork2.ActionSupport
      └org.sigiccs.impl.contrato.presentation.CargarContratoAction
```

All Implemented Interfaces:

com.opensymphony.xwork2.Action, com.opensymphony.xwork2.LocaleProvider, Serializable, com.opensymphony.xwork2.TextProvider, com.opensymphony.xwork2.Validateable, com.opensymphony.xwork2.ValidationAware

---

```
public class CargarContratoAction
extends com.opensymphony.xwork2.ActionSupport
```

Action encargado de solicitar los datos de un contrato

13.3.1.19 Clase CargarContratosDeUnClienteAction

org.sigiccs.impl.contrato.presentation

```
java.lang.Object
  └com.opensymphony.xwork2.ActionSupport
      └
org.sigiccs.impl.contrato.presentation.CargarContratosDeUnClienteAction
```

All Implemented Interfaces:

com.opensymphony.xwork2.Action, com.opensymphony.xwork2.LocaleProvider, Serializable, com.opensymphony.xwork2.TextProvider, com.opensymphony.xwork2.Validateable, com.opensymphony.xwork2.ValidationAware

---

```
public class CargarContratosDeUnClienteAction
extends com.opensymphony.xwork2.ActionSupport
```

Action encargado de consultar el listado de contratos suscritos por un cliente determinado

13.3.1.20 Clase CargarTodosContratosAction

org.sigiccs.impl.contrato.presentation

```
java.lang.Object
  └com.opensymphony.xwork2.ActionSupport
      └org.sigiccs.impl.contrato.presentation.CargarTodosContratosAction
```

All Implemented Interfaces:

com.opensymphony.xwork2.Action, com.opensymphony.xwork2.LocaleProvider, Serializable, com.opensymphony.xwork2.TextProvider, com.opensymphony.xwork2.Validateable, com.opensymphony.xwork2.ValidationAware

---

```
public class CargarTodosContratosAction
extends com.opensymphony.xwork2.ActionSupport
```

Action encargado de consultar el listado de todos los contratos dados de alta en el sistema

13.3.1.21 Clase CrearContratoAction

org.sigiccs.impl.contrato.presentation

```
java.lang.Object
  └com.opensymphony.xwork2.ActionSupport
      └org.sigiccs.impl.contrato.presentation.CrearContratoAction
```

All Implemented Interfaces:

com.opensymphony.xwork2.Action, com.opensymphony.xwork2.LocaleProvider, Serializable, com.opensymphony.xwork2.TextProvider, com.opensymphony.xwork2.Validateable, com.opensymphony.xwork2.ValidationAware

---

```
public class CrearContratoAction
extends com.opensymphony.xwork2.ActionSupport
```

Action encargado de recoger los datos del formulario de alta de contrato e invocar a la siguiente capa para persistirlos

13.3.1.22 Clase EliminarContratoAction

org.sigiccs.impl.contrato.presentation

```
java.lang.Object
  └com.opensymphony.xwork2.ActionSupport
      └org.sigiccs.impl.contrato.presentation.EliminarContratoAction
```

All Implemented Interfaces:

com.opensymphony.xwork2.Action, com.opensymphony.xwork2.LocaleProvider, Serializable, com.opensymphony.xwork2.TextProvider, com.opensymphony.xwork2.Validateable, com.opensymphony.xwork2.ValidationAware

---

```
public class EliminarContratoAction
extends com.opensymphony.xwork2.ActionSupport
```

Action encargado de marcar como eliminado un contrato

13.3.1.23 Clase EliminarServicioDelContratoAction

org.sigiccs.impl.contrato.presentation

```
java.lang.Object
  └com.opensymphony.xwork2.ActionSupport
      └
org.sigiccs.impl.contrato.presentation.EliminarServicioDelContratoAction
```

All Implemented Interfaces:

com.opensymphony.xwork2.Action, com.opensymphony.xwork2.LocaleProvider, Serializable, com.opensymphony.xwork2.TextProvider, com.opensymphony.xwork2.Validateable, com.opensymphony.xwork2.ValidationAware

---

```
public class EliminarServicioDelContratoAction
extends com.opensymphony.xwork2.ActionSupport
```

Action encargado de marcar como eliminada la asociación de un servicio a un contrato

13.3.1.24 Clase PrepararAsociarServicioAction

org.sigiccs.impl.contrato.presentation

```
java.lang.Object
  └com.opensymphony.xwork2.ActionSupport
      └org.sigiccs.impl.contrato.presentation.PrepararAsociarServicioAction
```

All Implemented Interfaces:

com.opensymphony.xwork2.Action, com.opensymphony.xwork2.LocaleProvider, Serializable, org.apache.struts2.interceptor.ServletRequestAware, com.opensymphony.xwork2.TextProvider, com.opensymphony.xwork2.Validateable, com.opensymphony.xwork2.ValidationAware

---

```
public class PrepararAsociarServicioAction
extends com.opensymphony.xwork2.ActionSupport
implements org.apache.struts2.interceptor.ServletRequestAware
```

Action encargado de precargar los formularios y campos necesarios para asociar un servicio a un contrato

13.3.1.25 Clase PrepararEdicionContratoServiciosAction

org.sigiccs.impl.contrato.presentation

```
java.lang.Object
  └com.opensymphony.xwork2.ActionSupport
      └
org.sigiccs.impl.contrato.presentation.PrepararEdicionContratoServiciosAction
```

All Implemented Interfaces:

com.opensymphony.xwork2.Action, com.opensymphony.xwork2.LocaleProvider, Serializable, com.opensymphony.xwork2.TextProvider, com.opensymphony.xwork2.Validateable, com.opensymphony.xwork2.ValidationAware

---

```
public class PrepararEdicionContratoServiciosAction
extends com.opensymphony.xwork2.ActionSupport
```

Action encargado de consultar y cargar los datos de una asociación servicio-contrato existente que se pretende modificar

13.3.1.26 Clase PrepararFormularioEdicionContratoAction

org.sigiccs.impl.contrato.presentation

```
java.lang.Object
  └com.opensymphony.xwork2.ActionSupport
      └
org.sigiccs.impl.contrato.presentation.PrepararFormularioEdicionContratoAction
```

All Implemented Interfaces:

com.opensymphony.xwork2.Action, com.opensymphony.xwork2.LocaleProvider, Serializable, com.opensymphony.xwork2.TextProvider, com.opensymphony.xwork2.Validateable, com.opensymphony.xwork2.ValidationAware

---

```
public class PrepararFormularioEdicionContratoAction
extends com.opensymphony.xwork2.ActionSupport
```

Action encargado de consultar y cargar los datos de un contrato existente que se pretende modificar

13.3.1.27 Clase PrepararFormularioNuevoContratoAction

org.sigiccs.impl.contrato.presentation

```
java.lang.Object
  └com.opensymphony.xwork2.ActionSupport
      └
org.sigiccs.impl.contrato.presentation.PrepararFormularioNuevoContratoAction
```

All Implemented Interfaces:

com.opensymphony.xwork2.Action, com.opensymphony.xwork2.LocaleProvider, Serializable, com.opensymphony.xwork2.TextProvider, com.opensymphony.xwork2.Validateable, com.opensymphony.xwork2.ValidationAware

---

```
public class PrepararFormularioNuevoContratoAction
extends com.opensymphony.xwork2.ActionSupport
```

Action encargado de precargar los datos necesarios en el formulario de alta de nuevo contrato

13.3.1.28 Clase DocumentoIncidenciaManager

org.sigiccs.impl.incidencia.business
java.lang.Object
└ **org.sigiccs.impl.incidencia.business.DocumentoIncidenciaManager**

All Implemented Interfaces:

DocumentoIncidenciaManagerService

---

public class **DocumentoIncidenciaManager**
extends Object
implements DocumentoIncidenciaManagerService

Clase encargada de establecer la conexión entre la invocación de acciones propias de la gestión de adjuntos a una incidencia en la capa de presentación y la de persistencia

Constructor Detail

DocumentoIncidenciaManager

public **DocumentoIncidenciaManager**()

Method Detail

setDocumentoIncidenciaDataService

public void **setDocumentoIncidenciaDataService**(DocumentoIncidenciaDataService documentoIncidenciaDataService)

Parameters:

documentoIncidenciaDataService - instancia del servicio de acceso a persistencia para las entidades DocumentoIncidencia

---

getDocumentoIncidenciaByID

public DocumentoIncidencia **getDocumentoIncidenciaByID**(int idDocumento)
throws SQLException

Description copied from interface: DocumentoIncidenciaManagerService

Busca un adjunto a una incidencia en base al identificador del fichero

Specified by:

getDocumentoIncidenciaByID in interface DocumentoIncidenciaManagerService

Parameters:

idDocumento - Identificador del fichero adjunto

Returns:

Un objeto DocumentoIncidencia con todos los datos del mismo

Throws:

SQLException

---

getAllDocumentosIncidencia

public Vector<DocumentoIncidencia> **getAllDocumentosIncidencia**(int idIncidencia)
throws SQLException

Description copied from interface: **DocumentoIncidenciaManagerService**

Busca todos los adjuntos a una incidencia en particular

Specified by:

getAllDocumentosIncidencia in interface DocumentoIncidenciaManagerService

Parameters:

`idIncidencia` - Identificador de la incidencia para la que se buscaran los documentos adjuntos

Returns:

Un objeto Vector con todos los objetos DocumentoIncidencia asociados a la incidencia

Throws:

`SQLException`

almacenarNuevoDocumentoIncidencia

```
public int almacenarNuevoDocumentoIncidencia(DocumentoIncidencia documentoIncidencia)
                                      throws SQLException
```

Description copied from interface: **DocumentoIncidenciaManagerService**

Almacena un fichero adjunto a una incidencia, incluidos sus datos descriptivos

Specified by:

almacenarNuevoDocumentoIncidencia in interface DocumentoIncidenciaManagerService

Parameters:

`documentoIncidencia` - El objeto que representa al adjunto

Returns:

1 si el fichero fue almacenado correctamente

Throws:

`SQLException`

eliminarDocumentoIncidencia

```
public int eliminarDocumentoIncidencia(int idDocumento)
                                throws SQLException
```

Description copied from interface: **DocumentoIncidenciaManagerService**

Marca como eliminado un fichero adjunto a una incidencia

Specified by:

eliminarDocumentoIncidencia in interface DocumentoIncidenciaManagerService

Parameters:

`idDocumento` - El identificador del adjunto a eliminar

Returns:

1 si el adjunto fue eliminado correctamente

Throws:

`SQLException`

eliminarDocumentosIncidencia

```
public int eliminarDocumentosIncidencia(int iDIncidencia)
                                 throws SQLException
```

Description copied from interface: **DocumentoIncidenciaManagerService**

Marca como eliminados todos los adjuntos a una incidencia

Specified by:

`eliminarDocumentosIncidencia` in interface `DocumentoIncidenciaManagerService`

Parameters:

`iDIncidencia` - Identificador de la incidencia a tratar

Returns:

1 si los adjuntos fueron eliminados correctamente

Throws:

`SQLException`

13.3.1.29 Clase IncidenciaManager

org.sigiccs.impl.incidencia.business
java.lang.Object
└ **org.sigiccs.impl.incidencia.business.IncidenciaManager**

All Implemented Interfaces:

IncidenciaManagerService

---

public class **IncidenciaManager**
extends Object
implements IncidenciaManagerService

Clase encargada de establecer la conexión entre la invocación de acciones propias de la gestión de incidencias en la capa de presentación y la de persistencia

Constructor Detail

IncidenciaManager

public **IncidenciaManager**()

Method Detail

setIncidenciaDataService

public void **setIncidenciaDataService**(IncidenciaDataService incidenciaDataService)

Parameters:

incidenciaDataService - instancia del servicio de acceso a persistencia para las entidades Incidencia

---

getIncidenciaByID

public Incidencia **getIncidenciaByID**(int IDIncidencia)
throws SQLException

Description copied from interface: IncidenciaManagerService

Busca toda la información asociada a una incidencia

Specified by:

getIncidenciaByID in interface IncidenciaManagerService

Parameters:

IDIncidencia - El identificador de la incidencia

Returns:

Un objeto Incidencia con todos los datos de la misma

Throws:

SQLException

---

getAllIncidencias

public Vector<Incidencia> **getAllIncidencias**()
throws SQLException

Description copied from interface: IncidenciaManagerService

Busca todas las incidencias dadas de alta en el sistema

Specified by:

`getAllIncidencias` in interface `IncidenciaManagerService`

Returns:

Un objeto Vector con todos los objetos Incidencia encontrados

Throws:

`SQLException`

---

almacenarNuevaIncidencia

```
public int almacenarNuevaIncidencia(Incidencia incidencia)
                             throws SQLException
```

Description copied from interface: IncidenciaManagerService

Almacena una nueva incidencia en el sistema

Specified by:

`almacenarNuevaIncidencia` in interface `IncidenciaManagerService`

Parameters:

`incidencia` - El objeto incidencia con todos los datos proporcionados en su creacion

Returns:

1 si la incidencia fue almacenada correctamente

Throws:

`SQLException`

---

eliminarIncidencia

```
public int eliminarIncidencia(int IDIncidencia)
                       throws SQLException
```

Description copied from interface: IncidenciaManagerService

Marca como eliminada una incidencia existente

Specified by:

`eliminarIncidencia` in interface `IncidenciaManagerService`

Parameters:

`IDIncidencia` - Identificador de la incidencia a eliminar

Returns:

1 si la incidencia fue eliminada correctamente

Throws:

`SQLException`

---

actualizarIncidencia

```
public int actualizarIncidencia(Incidencia incidencia)
                         throws SQLException
```

Description copied from interface: IncidenciaManagerService

Actualiza los datos de una incidencia existente

Specified by:

`actualizarIncidencia` in interface `IncidenciaManagerService`

Parameters:

`incidencia` - El objeto incidencia con los nuevos datos

Returns:

1 si la incidencia fue modificada correctamente

Throws:

`SQLException`

---

getAllIncidenciasCliente

```
public Vector<Incidencia> getAllIncidenciasCliente(int iDCliente)
                                                   throws SQLException
```

Description copied from interface: IncidenciaManagerService

Busca todas las incidencias asociadas a un cliente en particular

Specified by:

getAllIncidenciasCliente in interface IncidenciaManagerService

Parameters:

iDCliente - Identificador del cliente en base al cual se realiza la busqueda

Returns:

Un objeto Vector con todos los objetos Incidencia encontrados

Throws:

SQLException

---

getTiposIncidencia

```
public Vector<TipoIncidencia> getTiposIncidencia()
                                              throws SQLException
```

Description copied from interface: IncidenciaManagerService

Consulta el listado de Tipos de Incidencia existentes

Specified by:

getTiposIncidencia in interface IncidenciaManagerService

Returns:

Un objeto Vector con todos los objetos TipoIncidencia encontrados

Throws:

SQLException

---

getPrioridades

```
public Vector<Prioridad> getPrioridades()
                                    throws SQLException
```

Description copied from interface: IncidenciaManagerService

Consulta el listado de Tipos de Prioridad aplicables a las incidencias

Specified by:

getPrioridades in interface IncidenciaManagerService

Returns:

Un objeto Vector con todos los objetos Prioridad encontrados

Throws:

SQLException

13.3.1.30 Clase IntervencionManager

org.sigiccs.impl.incidencia.business
java.lang.Object
└ **org.sigiccs.impl.incidencia.business.IntervencionManager**

All Implemented Interfaces:

IntervencionManagerService

---

public class **IntervencionManager**
extends Object
implements IntervencionManagerService

Clase encargada de establecer la conexión entre la invocación de acciones propias de la gestión de las intervenciones de una incidencia en la capa de presentación y la de persistencia

Constructor Detail

IntervencionManager

public **IntervencionManager**()

Method Detail

setIntervencionDataService

public void **setIntervencionDataService**(IntervencionDataService intervencionDataService)

Parameters:

intervencionDataService - instancia del servicio de acceso a persistencia para las entidades Intervencion

---

getIntervencionByID

public Intervencion **getIntervencionByID**(int idIntervencion)
throws SQLException

Description copied from interface: IntervencionManagerService

Busca una intervencion en una incidencia en base al identificador de la misma

Specified by:

getIntervencionByID in interface IntervencionManagerService

Parameters:

idIntervencion - Identificador de la intervencion a buscar

Returns:

Un objeto Intervencion con los datos de la misma

Throws:

SQLException

---

getAllIntervencionesIncidencia

public Vector<Intervencion> **getAllIntervencionesIncidencia**(int idIncidencia)
throws SQLException

Description copied from interface: IntervencionManagerService

Busca todas las intervenciones efectuadas en una incidencia en particular

Specified by:

`getAllIntervencionesIncidencia` in interface `IntervencionManagerService`

Parameters:

`idIncidencia` - Identificador de la incidencia en la que se buscara

Returns:

Un objeto Vector con todos los objetos Intervencion encontrados

Throws:

`SQLException`

---

almacenarNuevaIntervencion

```
public int almacenarNuevaIntervencion(Intervencion intervencion)
                               throws SQLException
```

Description copied from interface: IntervencionManagerService

Almacena una intervencion en una incidencia

Specified by:

`almacenarNuevaIntervencion` in interface `IntervencionManagerService`

Parameters:

`intervencion` - El objeto que representa a la intervencion

Returns:

1 si la intervencion fue almacenada correctamente

Throws:

`SQLException`

---

eliminarIntervencion

```
public int eliminarIntervencion(int idIntervencion)
                         throws SQLException
```

Description copied from interface: IntervencionManagerService

Marca como eliminada una intervencion existente en una incidencia

Specified by:

`eliminarIntervencion` in interface `IntervencionManagerService`

Parameters:

`idIntervencion` - Identificador de la intervencion a eliminar

Returns:

1 si la intervencion fue eliminada correctamente

Throws:

`SQLException`

---

actualizarIntervencion

```
public int actualizarIntervencion(Intervencion intervencion)
                           throws SQLException
```

Description copied from interface: IntervencionManagerService

Actualiza las propiedades de una intervencion existente

Specified by:

`actualizarIntervencion` in interface `IntervencionManagerService`

Parameters:

`intervencion` - El objeto intervencion con los nuevos datos

Returns:

1 si la intervencion fue modificada correctamente

Throws:

`SQLException`

eliminarIntervencionesIncidencia

```
public int eliminarIntervencionesIncidencia(int iDIncidencia)
                                     throws SQLException
```

Description copied from interface: `IntervencionManagerService`

Marca como eliminadas todas las intervenciones de una incidencia

Specified by:

`eliminarIntervencionesIncidencia` in interface `IntervencionManagerService`

Parameters:

`iDIncidencia` - Identificador de la incidencia en la que se eliminaran sus intervenciones

Returns:

1 si las intervenciones fueron eliminadas correctamente

Throws:

`SQLException`

13.3.1.31 Clase RelacionIncidenciaManager

org.sigiccs.impl.incidencia.business
java.lang.Object
└ **org.sigiccs.impl.incidencia.business.RelacionIncidenciaManager**

All Implemented Interfaces:

RelacionIncidenciaManagerService

---

public class **RelacionIncidenciaManager**
extends Object
implements RelacionIncidenciaManagerService

Clase encargada de establecer la conexión entre la invocación de acciones propias de la gestión de las relaciones de una incidencia en la capa de presentación y la de persistencia

Constructor Detail

RelacionIncidenciaManager

public **RelacionIncidenciaManager**()

Method Detail

setRelacionIncidenciaDataService

public void **setRelacionIncidenciaDataService**(RelacionIncidenciaDataService relacionIncidenciaDataService)

Parameters:

relacionIncidenciaDataService - instancia del servicio de acceso a persistencia para las entidades RelacionIncidencia

---

getRelacionIncidenciaByID

public RelacionIncidencia **getRelacionIncidenciaByID**(int iDRelacion) throws SQLException

Description copied from interface: RelacionIncidenciaManagerService

Consulta todos los datos de una relación en particular

Specified by:

getRelacionIncidenciaByID in interface RelacionIncidenciaManagerService

Parameters:

iDRelacion - Identificador de la relaicon a consultar

Returns:

Un objeto RelacionIncidencia con todos los datos de la relación

Throws:

SQLException

---

getRelacionesIncidenciaMain

public Vector<RelacionIncidencia> **getRelacionesIncidenciaMain**(int iDIncidenciaMain) throws SQLException

Description copied from interface: **RelacionIncidenciaManagerService**

Consulta todas las relaciones en las que una incidencia es origen de las mismas

Specified by:

getRelacionesIncidenciaMain in interface RelacionIncidenciaManagerService

Parameters:

`iDIncidenciaMain` - Identificador de la incidencia origen

Returns:

Un objeto Vector con todos los objetos RelacionIncidencia encontrados

Throws:

`SQLException`

---

getRelacionesIncidenciaSub

```
public Vector<RelacionIncidencia> getRelacionesIncidenciaSub(int iDIncidenciaSub)
                                                    throws SQLException
```

Description copied from interface: **RelacionIncidenciaManagerService**

Consulta todas las relaciones en las que una incidencia es destino de las mismas

Specified by:

getRelacionesIncidenciaSub in interface RelacionIncidenciaManagerService

Returns:

Un objeto Vector con todos los objetos RelacionIncidencia encontrados

Throws:

`SQLException`

---

almacenarNuevaRelacionIncidencia

```
public int almacenarNuevaRelacionIncidencia(RelacionIncidencia relacionIncidencia)
                                     throws SQLException
```

Description copied from interface: **RelacionIncidenciaManagerService**

Establece una nueva relación para la incidencia

Specified by:

almacenarNuevaRelacionIncidencia in interface RelacionIncidenciaManagerService

Parameters:

`relacionIncidencia` - El objeto que representa a la relación

Returns:

1 si la relación fue establecida correctamente

Throws:

`SQLException`

---

eliminarRelacionIncidencia

```
public int eliminarRelacionIncidencia(int iDRelacion)
                               throws SQLException
```

Description copied from interface: **RelacionIncidenciaManagerService**

Marca como eliminada una relación en particular

Specified by:

eliminarRelacionIncidencia in interface RelacionIncidenciaManagerService

Parameters:

`iDRelacion` - Identificador de la relación a eliminar

Returns:

1 si la relación fue eliminada correctamente

Throws:

`SQLException`

---

getTiposRelaciones

```
public Vector<TipoRelacion> getTiposRelaciones()
                                        throws SQLException
```

Description copied from interface: RelacionIncidenciaManagerService

Consulta el listado de Tipos de Relación entre incidencias existentes

Specified by:

`getTiposRelaciones` in interface `RelacionIncidenciaManagerService`

Returns:

Un objeto Vector con todos los objetos TipoRelacion encontrados

Throws:

`SQLException`

---

getTipoRelacionByID

```
public TipoRelacion getTipoRelacionByID(int iDTipoRelacion)
                                  throws SQLException
```

Description copied from interface: RelacionIncidenciaManagerService

Devuelve un Tipo de Relación en base a su identificador

Specified by:

`getTipoRelacionByID` in interface `RelacionIncidenciaManagerService`

Parameters:

`iDTipoRelacion` - Identificador del tipo de relación a consultar

Returns:

Un objeto TipoRelacion con todos los datos del mismo

Throws:

`SQLException`

---

esRelacionable

```
public boolean esRelacionable(RelacionIncidencia relación)
                       throws SQLException
```

Description copied from interface: RelacionIncidenciaManagerService

Comprueba si una incidencia es relacionable con otra

Specified by:

`esRelacionable` in interface `RelacionIncidenciaManagerService`

Parameters:

`relación` - Atributos de la relación que se pretende establecer

Returns:

true si las incidencias son relacionables. false en caso contrario

Throws:

`SQLException`

---

eliminarRelacionesIncidencia

```
public int eliminarRelacionesIncidencia(int iDIncidencia)
                                 throws SQLException
```

Description copied from interface: `RelacionIncidenciaManagerService`

Marca como eliminadas todas las relaciones directas de una incidencia

Specified by:

`eliminarRelacionesIncidencia` in interface `RelacionIncidenciaManagerService`

Parameters:

`iDIncidencia` - Identificador de la incidencia para la que se eliminaran sus relaciones

Returns:

1 si las relaciones fueron eliminadas correctamente

Throws:

`SQLException`

13.3.1.32 Clase DocumentoIncidenciaDAO

org.sigiccs.impl.incidencia.persistence
java.lang.Object
└ **org.sigiccs.impl.incidencia.persistence.DocumentoIncidenciaDAO**

All Implemented Interfaces:

DocumentoIncidenciaDataService

---

public class **DocumentoIncidenciaDAO**
extends Object
implements DocumentoIncidenciaDataService

Clase encargada de la implementación de todos los métodos declarados en la interfaz que implementa, encargados de efectuar las operaciones de la gestión de adjuntos a las incidencias en lo relativo a la consulta y modificación de datos

Constructor Detail

DocumentoIncidenciaDAO

public **DocumentoIncidenciaDAO**()

Method Detail

getDocumentoIncidenciaByID

public DocumentoIncidencia **getDocumentoIncidenciaByID**(int idDocumento)
throws SQLException

Description copied from interface: DocumentoIncidenciaDataService

Busca un adjunto a una incidencia en base al identificador del fichero

Specified by:

getDocumentoIncidenciaByID in interface DocumentoIncidenciaDataService

Parameters:

idDocumento - Identificador del fichero adjunto

Returns:

Un objeto DocumentoIncidencia con todos los datos del mismo

Throws:

SQLException

---

getAllDocumentosIncidencia

public Vector<DocumentoIncidencia> **getAllDocumentosIncidencia**(int idIncidencia
)
throws SQLException

Description copied from interface: DocumentoIncidenciaDataService

Busca todos los adjuntos a una incidencia en particular

Specified by:

getAllDocumentosIncidencia in interface DocumentoIncidenciaDataService

Parameters:

idIncidencia - Identificador de la incidencia para la que se buscaran los documentos adjuntos

Returns:

Un objeto Vector con todos los objetos DocumentoIncidencia asociados a la incidencia

Throws:

`SQLException`

almacenarNuevoDocumentoIncidencia

```
public int almacenarNuevoDocumentoIncidencia(DocumentoIncidencia documentoIncidencia)
                                      throws SQLException
```

Description copied from interface: DocumentoIncidenciaDataService

Almacena un fichero adjunto a una incidencia, incluidos sus datos descriptivos

Specified by:

`almacenarNuevoDocumentoIncidencia` in interface `DocumentoIncidenciaDataService`

Parameters:

`documentoIncidencia` - El objeto que representa al adjunto

Returns:

1 si el fichero fue almacenado correctamente

Throws:

`SQLException`

eliminarDocumentoIncidencia

```
public int eliminarDocumentoIncidencia(int idDocumento)
                                throws SQLException
```

Description copied from interface: DocumentoIncidenciaDataService

Marca como eliminado un fichero adjunto a una incidencia

Specified by:

`eliminarDocumentoIncidencia` in interface `DocumentoIncidenciaDataService`

Parameters:

`idDocumento` - El identificador del adjunto a eliminar

Returns:

1 si el adjunto fue eliminado correctamente

Throws:

`SQLException`

eliminarDocumentosIncidencia

```
public int eliminarDocumentosIncidencia(int iDIncidencia)
                                 throws SQLException
```

Description copied from interface: DocumentoIncidenciaDataService

Marca como eliminados todos los adjuntos a una incidencia

Specified by:

`eliminarDocumentosIncidencia` in interface `DocumentoIncidenciaDataService`

Parameters:

`iDIncidencia` - Identificador de la incidencia a tratar

Returns:

1 si los adjuntos fueron eliminados correctamente

Throws:

`SQLException`

13.3.1.33 Clase IncidenciaDAO

org.sigiccs.impl.incidencia.persistence
java.lang.Object
  └ **org.sigiccs.impl.incidencia.persistence.IncidenciaDAO**

All Implemented Interfaces:

IncidenciaDataService

---

public class **IncidenciaDAO**
extends Object
implements IncidenciaDataService

Clase encargada de la implementación de todos los métodos declarados en la interfaz que implementa, encargados de efectuar las operaciones de la gestión de incidencias en lo relativo a la consulta y modificación de datos

Constructor Detail

IncidenciaDAO

public **IncidenciaDAO**()

Method Detail

getIncidenciaByID

public Incidencia **getIncidenciaByID**(int iDIncidencia)
throws SQLException

Description copied from interface: IncidenciaDataService

Busca toda la información asociada a una incidencia

Specified by:

getIncidenciaByID in interface IncidenciaDataService

Returns:

Un objeto Incidencia con todos los datos de la misma

Throws:

SQLException

---

getAllIncidencias

public Vector<Incidencia> **getAllIncidencias**()
throws SQLException

Description copied from interface: IncidenciaDataService

Busca todas las incidencias dadas de alta en el sistema

Specified by:

getAllIncidencias in interface IncidenciaDataService

Returns:

Un objeto Vector con todos los objetos Incidencia encontrados

Throws:

SQLException

---

almacenarNuevaIncidencia

```
public int almacenarNuevaIncidencia(Incidencia incidencia)
                             throws SQLException
```

Description copied from interface: IncidenciaDataService

Almacena una nueva incidencia en el sistema

Specified by:

almacenarNuevaIncidencia in interface IncidenciaDataService

Parameters:

incidencia - El objeto incidencia con todos los datos proporcionados en su creacion

Returns:

1 si la incidencia fue almacenada correctamente

Throws:

SQLException

---

eliminarIncidencia

```
public int eliminarIncidencia(int iDIncidencia)
                       throws SQLException
```

Description copied from interface: IncidenciaDataService

Marca como eliminada una incidencia existente

Specified by:

eliminarIncidencia in interface IncidenciaDataService

Returns:

1 si la incidencia fue eliminada correctamente

Throws:

SQLException

---

actualizarIncidencia

```
public int actualizarIncidencia(Incidencia incidencia)
                         throws SQLException
```

Description copied from interface: IncidenciaDataService

Actualiza los datos de una incidencia existente

Specified by:

actualizarIncidencia in interface IncidenciaDataService

Parameters:

incidencia - El objeto incidencia con los nuevos datos

Returns:

1 si la incidencia fue modificada correctamente

Throws:

SQLException

---

getAllIncidenciasCliente

```
public Vector<Incidencia> getAllIncidenciasCliente(int iDCliente)
                                            throws SQLException
```

Description copied from interface: IncidenciaDataService

Busca todas las incidencias asociadas a un cliente en particular

Specified by:

getAllIncidenciasCliente in interface IncidenciaDataService

Parameters:

`iDCliente` - Identificador del cliente en base al cual se realiza la busqueda

Returns:

Un objeto Vector con todos los objetos Incidencia encontrados

Throws:

`SQLException`

---

getTiposIncidencia

```
public Vector<TipoIncidencia> getTiposIncidencia()
                                        throws SQLException
```

Description copied from interface: IncidenciaDataService

Consulta el listado de Tipos de Incidencia existentes

Specified by:

`getTiposIncidencia` in interface `IncidenciaDataService`

Returns:

Un objeto Vector con todos los objetos TipoIncidencia encontrados

Throws:

`SQLException`

---

getPrioridades

```
public Vector<Prioridad> getPrioridades()
                                  throws SQLException
```

Description copied from interface: IncidenciaDataService

Consulta el listado de Tipos de Prioridad aplicables a las incidencias

Specified by:

`getPrioridades` in interface `IncidenciaDataService`

Returns:

Un objeto Vector con todos los objetos Prioridad encontrados

Throws:

`SQLException`

13.3.1.34 Clase IntervencionDAO

org.sigiccs.impl.incidencia.persistence
java.lang.Object
└ **org.sigiccs.impl.incidencia.persistence.IntervencionDAO**

All Implemented Interfaces:

IntervencionDataService

```
public class IntervencionDAO
extends Object
implements IntervencionDataService
```

Clase encargada de la implementación de todos los métodos declarados en la interfaz que implementa, encargados de efectuar las operaciones de la gestión de intervenciones de las incidencias en lo relativo a la consulta y modificación de datos

Constructor Detail

IntervencionDAO

```
public IntervencionDAO()
```

Method Detail

getIntervencionByID

```
public Intervencion getIntervencionByID(int idIntervencion)
                                 throws SQLException
```

Description copied from interface: IntervencionDataService

Busca una intervencion en una incidencia en base al identificador de la misma

Specified by:

getIntervencionByID in interface IntervencionDataService

Parameters:

idIntervencion - Identificador de la intervencion a buscar

Returns:

Un objeto Intervencion con los datos de la misma

Throws:

SQLException

---

getAllIntervencionesIncidencia

```
public Vector<Intervencion> getAllIntervencionesIncidencia(int idIncidencia)
                                                    throws SQLException
```

Description copied from interface: IntervencionDataService

Busca todas las intervenciones efectuadas en una incidencia en particular

Specified by:

getAllIntervencionesIncidencia in interface IntervencionDataService

Parameters:

idIncidencia - Identificador de la incidencia en la que se buscara

Returns:

Un objeto Vector con todos los objetos Intervencion encontrados

Throws:

SQLException

almacenarNuevaIntervencion

```
public int almacenarNuevaIntervencion(Intervencion intervencion)
                               throws SQLException
```

Description copied from interface: IntervencionDataService

Almacena una intervencion en una incidencia

Specified by:

almacenarNuevaIntervencion in interface IntervencionDataService

Parameters:

intervencion - El objeto que representa a la intervencion

Returns:

1 si la intervencion fue almacenada correctamente

Throws:

SQLException

eliminarIntervencion

```
public int eliminarIntervencion(int idIntervencion)
                         throws SQLException
```

Description copied from interface: IntervencionDataService

Marca como eliminada una intervencion existente en una incidencia

Specified by:

eliminarIntervencion in interface IntervencionDataService

Parameters:

idIntervencion - Identificador de la intervencion a eliminar

Returns:

1 si la intervencion fue eliminada correctamente

Throws:

SQLException

actualizarIntervencion

```
public int actualizarIntervencion(Intervencion intervencion)
                           throws SQLException
```

Description copied from interface: IntervencionDataService

Actualiza las propiedades de una intervencion existente

Specified by:

actualizarIntervencion in interface IntervencionDataService

Parameters:

intervencion - El objeto intervencion con los nuevos datos

Returns:

1 si la intervencion fue modificada correctamente

Throws:

SQLException

eliminarIntervencionesIncidencia

```
public int eliminarIntervencionesIncidencia(int iDIncidencia)
                                     throws SQLException
```

Description copied from interface: `IntervencionDataService`

Marca como eliminadas todas las intervenciones de una incidencia

Specified by:

`eliminarIntervencionesIncidencia` in interface `IntervencionDataService`

Parameters:

`iDIncidencia` - Identificador de la incidencia en la que se eliminaran sus intervenciones

Returns:

1 si las intervenciones fueron eliminadas correctamente

Throws:

`SQLException`

13.3.1.35 Clase RelacionIncidenciaDAO

org.sigiccs.impl.incidencia.persistence
java.lang.Object
└ **org.sigiccs.impl.incidencia.persistence.RelacionIncidenciaDAO**

All Implemented Interfaces:

RelacionIncidenciaDataService

---

public class **RelacionIncidenciaDAO**
extends Object
implements RelacionIncidenciaDataService

Clase encargada de la implementación de todos los métodos declarados en la interfaz que implementa, encargados de efectuar las operaciones de la gestión de las relaciones entre incidencias en lo relativo a la consulta y modificación de datos

Constructor Detail

RelacionIncidenciaDAO

public **RelacionIncidenciaDAO**()

Method Detail

getRelacionIncidenciaByID

public RelacionIncidencia **getRelacionIncidenciaByID**(int iDRelacion)
throws SQLException

Description copied from interface: RelacionIncidenciaDataService

Consulta todos los datos de una relación en particular

Specified by:

getRelacionIncidenciaByID in interface RelacionIncidenciaDataService

Parameters:

iDRelacion - Identificador de la relaicon a consultar

Returns:

Un objeto RelacionIncidencia con todos los datos de la relación

Throws:

SQLException

---

getRelacionesIncidenciaMain

public Vector<RelacionIncidencia> **getRelacionesIncidenciaMain**(int iDIncidencia Main)
throws SQLException

Description copied from interface: RelacionIncidenciaDataService

Consulta todas las relaciones en las que una incidencia es origen de las mismas

Specified by:

getRelacionesIncidenciaMain in interface RelacionIncidenciaDataService

Parameters:

iDIncidenciaMain - Identificador de la incidencia origen

Returns:

Un objeto Vector con todos los objetos RelacionIncidencia encontrados

Throws:

SQLException

getRelacionesIncidenciaSub

```
public Vector<RelacionIncidencia> getRelacionesIncidenciaSub(int iDIncidenciaSub)
                                                    throws SQLException
```

Description copied from interface: RelacionIncidenciaDataService

Consulta todas las relaciones en las que una incidencia es destino de las mismas

Specified by:

getRelacionesIncidenciaSub in interface RelacionIncidenciaDataService

Returns:

Un objeto Vector con todos los objetos RelacionIncidencia encontrados

Throws:

SQLException

almacenarNuevaRelacionIncidencia

```
public int almacenarNuevaRelacionIncidencia(RelacionIncidencia relacionIncidencia)
                                     throws SQLException
```

Description copied from interface: RelacionIncidenciaDataService

Establece una nueva relación para la incidencia

Specified by:

almacenarNuevaRelacionIncidencia in interface RelacionIncidenciaDataService

Parameters:

relacionIncidencia - El objeto que representa a la relación

Returns:

1 si la relación fue establecida correctamente

Throws:

SQLException

eliminarRelacionIncidencia

```
public int eliminarRelacionIncidencia(int iDRelacion)
                               throws SQLException
```

Description copied from interface: RelacionIncidenciaDataService

Marca como eliminada una relación en particular

Specified by:

eliminarRelacionIncidencia in interface RelacionIncidenciaDataService

Parameters:

iDRelacion - Identificador de la relación a eliminar

Returns:

1 si la relación fue eliminada correctamente

Throws:

SQLException

getTiposRelaciones

```
public Vector<TipoRelacion> getTiposRelaciones()
                                     throws SQLException
```

Description copied from interface: **RelacionIncidenciaDataService**

Consulta el listado de Tipos de Relación entre incidencias existentes

Specified by:

`getTiposRelaciones` in interface `RelacionIncidenciaDataService`

Returns:

Un objeto Vector con todos los objetos TipoRelacion encontrados

Throws:

`SQLException`

---

getTipoRelacionByID

```
public TipoRelacion getTipoRelacionByID(int iDTipoRelacion)
                                 throws SQLException
```

Description copied from interface: **RelacionIncidenciaDataService**

Devuelve un Tipo de Relación en base a su identificador

Specified by:

`getTipoRelacionByID` in interface `RelacionIncidenciaDataService`

Parameters:

`iDTipoRelacion` - Identificador del tipo de relación a consultar

Returns:

Un objeto TipoRelacion con todos los datos del mismo

Throws:

`SQLException`

---

esRelacionable

```
public boolean esRelacionable(RelacionIncidencia relación)
                       throws SQLException
```

Description copied from interface: **RelacionIncidenciaDataService**

Comprueba si una incidencia es relacionable con otra

Specified by:

`esRelacionable` in interface `RelacionIncidenciaDataService`

Parameters:

`relación` - Atributos de la relación que se pretende establecer

Returns:

true si las incidencias son relacionables. false en caso contrario

Throws:

`SQLException`

---

eliminarRelacionesIncidencia

```
public int eliminarRelacionesIncidencia(int iDIncidencia)
                                 throws SQLException
```

Description copied from interface: **RelacionIncidenciaDataService**

Marca como eliminadas todas las relaciones directas de una incidencia

Specified by:

`eliminarRelacionesIncidencia` in interface `RelacionIncidenciaDataService`

Parameters:

`iDIncidencia` - Identificador de la incidencia para la que se eliminaran sus relaciones

Returns:

1 si las relaciones fueron eliminadas correctamente

Throws:

`SQLException`

13.3.1.36 Clase ActualizarIncidenciaAction

org.sigiccs.impl.incidencia.presentation

```
java.lang.Object
  └com.opensymphony.xwork2.ActionSupport
      └org.sigiccs.impl.incidencia.presentation.ActualizarIncidenciaAction
```

All Implemented Interfaces:

com.opensymphony.xwork2.Action, com.opensymphony.xwork2.LocaleProvider, Serializable, com.opensymphony.xwork2.TextProvider, com.opensymphony.xwork2.Validateable, com.opensymphony.xwork2.ValidationAware

---

```
public class ActualizarIncidenciaAction
extends com.opensymphony.xwork2.ActionSupport
```

Action encargado de recoger los cambios en las propiedades de una incidencia e invocar a la siguiente capa para persistirlos

13.3.1.37 Clase ActualizarIntervencionAction

org.sigiccs.impl.incidencia.presentation

```
java.lang.Object
  └com.opensymphony.xwork2.ActionSupport
      └org.sigiccs.impl.incidencia.presentation.ActualizarIntervencionAction
```

All Implemented Interfaces:

com.opensymphony.xwork2.Action, com.opensymphony.xwork2.LocaleProvider, Serializable, com.opensymphony.xwork2.TextProvider, com.opensymphony.xwork2.Validateable, com.opensymphony.xwork2.ValidationAware

---

```
public class ActualizarIntervencionAction
extends com.opensymphony.xwork2.ActionSupport
```

Action encargado de recoger los cambios en las propiedades de una intervencion e invocar a la siguiente capa para persistirlos

13.3.1.38 Clase AdjuntarAIncidenciaAction

org.sigiccs.impl.incidencia.presentation

```
java.lang.Object
  └com.opensymphony.xwork2.ActionSupport
      └org.sigiccs.impl.incidencia.presentation.AdjuntarAIncidenciaAction
```

All Implemented Interfaces:

com.opensymphony.xwork2.Action, com.opensymphony.xwork2.LocaleProvider, Serializable, com.opensymphony.xwork2.TextProvider, com.opensymphony.xwork2.Validateable, com.opensymphony.xwork2.ValidationAware

---

```
public class AdjuntarAIncidenciaAction
extends com.opensymphony.xwork2.ActionSupport
```

Action encargado de recoger los datos del formulario empleado para adjuntar ficheros a una incidencia e invocar a la siguiente capa para persistirlos

13.3.1.39 Clase CargarIncidenciaAction

org.sigiccs.impl.incidencia.presentation

```
java.lang.Object
  └com.opensymphony.xwork2.ActionSupport
      └org.sigiccs.impl.incidencia.presentation.CargarIncidenciaAction
```

All Implemented Interfaces:

com.opensymphony.xwork2.Action, com.opensymphony.xwork2.LocaleProvider, Serializable, com.opensymphony.xwork2.TextProvider, com.opensymphony.xwork2.Validateable, com.opensymphony.xwork2.ValidationAware

---

```
public class CargarIncidenciaAction
extends com.opensymphony.xwork2.ActionSupport
```

Action encargado de solicitar los datos de una incidencia

13.3.1.40 Clase CargarIncidenciasClienteAction

org.sigiccs.impl.incidencia.presentation

```
java.lang.Object
  └com.opensymphony.xwork2.ActionSupport
      └
org.sigiccs.impl.incidencia.presentation.CargarIncidenciasClienteAction
```

All Implemented Interfaces:

com.opensymphony.xwork2.Action, com.opensymphony.xwork2.LocaleProvider, Serializable, com.opensymphony.xwork2.TextProvider, com.opensymphony.xwork2.Validateable, com.opensymphony.xwork2.ValidationAware

---

```
public class CargarIncidenciasClienteAction
extends com.opensymphony.xwork2.ActionSupport
```

Action encargado de cargar el listado de incidencias asociadas a un cliente

13.3.1.41 Clase CargarTodasIncidenciasAction

org.sigiccs.impl.incidencia.presentation

```
java.lang.Object
  └com.opensymphony.xwork2.ActionSupport
      └org.sigiccs.impl.incidencia.presentation.CargarTodasIncidenciasAction
```

All Implemented Interfaces:

com.opensymphony.xwork2.Action, com.opensymphony.xwork2.LocaleProvider, Serializable, com.opensymphony.xwork2.TextProvider, com.opensymphony.xwork2.Validateable, com.opensymphony.xwork2.ValidationAware

---

```
public class CargarTodasIncidenciasAction
extends com.opensymphony.xwork2.ActionSupport
```

Action encargado de cargar el listado con todas las incidencias existentes en el sistema

13.3.1.42 Clase ClonarIncidenciaAction

org.sigiccs.impl.incidencia.presentation

```
java.lang.Object
  └com.opensymphony.xwork2.ActionSupport
      └org.sigiccs.impl.incidencia.presentation.ClonarIncidenciaAction
```

All Implemented Interfaces:

com.opensymphony.xwork2.Action, com.opensymphony.xwork2.LocaleProvider, Serializable, com.opensymphony.xwork2.TextProvider, com.opensymphony.xwork2.Validateable, com.opensymphony.xwork2.ValidationAware

---

```
public class ClonarIncidenciaAction
extends com.opensymphony.xwork2.ActionSupport
```

Action encargado de leer los datos de una incidencia e invocar las operaciones de persistencia necesarias para duplicarla

13.3.1.43 Clase CrearIncidenciaAction

org.sigiccs.impl.incidencia.presentation

```
java.lang.Object
  └com.opensymphony.xwork2.ActionSupport
      └org.sigiccs.impl.incidencia.presentation.CrearIncidenciaAction
```

All Implemented Interfaces:

com.opensymphony.xwork2.Action, com.opensymphony.xwork2.LocaleProvider, Serializable, com.opensymphony.xwork2.TextProvider, com.opensymphony.xwork2.Validateable, com.opensymphony.xwork2.ValidationAware

---

```
public class CrearIncidenciaAction
extends com.opensymphony.xwork2.ActionSupport
```

Action encargado de recoger los datos del formulario de alta de incidencia e invocar a la siguiente capa para persistirlos

13.3.1.44 Clase CrearIntervencionIncidenciaAction

org.sigiccs.impl.incidencia.presentation

```
java.lang.Object
  └com.opensymphony.xwork2.ActionSupport
      └
org.sigiccs.impl.incidencia.presentation.CrearIntervencionIncidenciaAction
```

All Implemented Interfaces:

com.opensymphony.xwork2.Action, com.opensymphony.xwork2.LocaleProvider, Serializable, com.opensymphony.xwork2.TextProvider, com.opensymphony.xwork2.Validateable, com.opensymphony.xwork2.ValidationAware

---

```
public class CrearIntervencionIncidenciaAction
extends com.opensymphony.xwork2.ActionSupport
```

Action encargado de recoger los datos del formulario de alta de intervencion en una incidencia e invocar a la siguiente capa para persistirlos

13.3.1.45 Clase CrearRelacionIncidenciaAction

org.sigiccs.impl.incidencia.presentation

```
java.lang.Object
  └com.opensymphony.xwork2.ActionSupport
      └org.sigiccs.impl.incidencia.presentation.CrearRelacionIncidenciaAction
```

All Implemented Interfaces:

com.opensymphony.xwork2.Action, com.opensymphony.xwork2.LocaleProvider, Serializable, com.opensymphony.xwork2.TextProvider, com.opensymphony.xwork2.Validateable, com.opensymphony.xwork2.ValidationAware

---

```
public class CrearRelacionIncidenciaAction
extends com.opensymphony.xwork2.ActionSupport
```

Action encargado de recoger los datos del formulario de alta de relación entre incidencias e invocar a la siguiente capa para persistirlos

13.3.1.46 Clase EliminarAdjuntoAction

org.sigiccs.impl.incidencia.presentation

```
java.lang.Object
  └com.opensymphony.xwork2.ActionSupport
      └org.sigiccs.impl.incidencia.presentation.EliminarAdjuntoAction
```

All Implemented Interfaces:

com.opensymphony.xwork2.Action, com.opensymphony.xwork2.LocaleProvider, Serializable, com.opensymphony.xwork2.TextProvider, com.opensymphony.xwork2.Validateable, com.opensymphony.xwork2.ValidationAware

---

```
public class EliminarAdjuntoAction
extends com.opensymphony.xwork2.ActionSupport
```

Action encargado de marcar como eliminada la asociación de un adjunto a una incidencia

13.3.1.47 Clase EliminarIncidenciaAction

org.sigiccs.impl.incidencia.presentation

```
java.lang.Object
  └com.opensymphony.xwork2.ActionSupport
      └org.sigiccs.impl.incidencia.presentation.EliminarIncidenciaAction
```

All Implemented Interfaces:

com.opensymphony.xwork2.Action, com.opensymphony.xwork2.LocaleProvider, Serializable, com.opensymphony.xwork2.TextProvider, com.opensymphony.xwork2.Validateable, com.opensymphony.xwork2.ValidationAware

---

```
public class EliminarIncidenciaAction
extends com.opensymphony.xwork2.ActionSupport
```

Action encargado de marcar como eliminada una incidencia existente en el sistema

13.3.1.48 Clase EliminarIntervencionAction

org.sigiccs.impl.incidencia.presentation

```
java.lang.Object
  └com.opensymphony.xwork2.ActionSupport
      └org.sigiccs.impl.incidencia.presentation.EliminarIntervencionAction
```

All Implemented Interfaces:

com.opensymphony.xwork2.Action, com.opensymphony.xwork2.LocaleProvider, Serializable, com.opensymphony.xwork2.TextProvider, com.opensymphony.xwork2.Validateable, com.opensymphony.xwork2.ValidationAware

---

```
public class EliminarIntervencionAction
extends com.opensymphony.xwork2.ActionSupport
```

Action encargado de marcar como eliminada una intervencion de una incidencia

13.3.1.49 Clase EliminarRelacionIncidenciaAction

org.sigiccs.impl.incidencia.presentation

```
java.lang.Object
  └com.opensymphony.xwork2.ActionSupport
      └
org.sigiccs.impl.incidencia.presentation.EliminarRelacionIncidenciaAction
```

All Implemented Interfaces:

com.opensymphony.xwork2.Action, com.opensymphony.xwork2.LocaleProvider, Serializable, com.opensymphony.xwork2.TextProvider, com.opensymphony.xwork2.Validateable, com.opensymphony.xwork2.ValidationAware

---

```
public class EliminarRelacionIncidenciaAction
extends com.opensymphony.xwork2.ActionSupport
```

Action encargado de marcar como eliminada una relación de una incidencia

13.3.1.50 Clase PrepararFormularioEdicionIncidenciaAction

org.sigiccs.impl.incidencia.presentation

```
java.lang.Object
  └com.opensymphony.xwork2.ActionSupport
      └
org.sigiccs.impl.incidencia.presentation.PrepararFormularioEdicionIncidenciaAction
```

All Implemented Interfaces:

com.opensymphony.xwork2.Action, com.opensymphony.xwork2.LocaleProvider, Serializable, com.opensymphony.xwork2.TextProvider, com.opensymphony.xwork2.Validateable, com.opensymphony.xwork2.ValidationAware

---

```
public class PrepararFormularioEdicionIncidenciaAction
extends com.opensymphony.xwork2.ActionSupport
```

Action encargado de consultar y cargar los datos de una incidencia existente que se pretende modificar

13.3.1.51 Clase PrepararFormularioEdicionIntervencionIncidenciaAction

org.sigiccs.impl.incidencia.presentation

```
java.lang.Object
  └ com.opensymphony.xwork2.ActionSupport
      └
org.sigiccs.impl.incidencia.presentation.PrepararFormularioEdicionIntervencion
IncidenciaAction
```

All Implemented Interfaces:

com.opensymphony.xwork2.Action, com.opensymphony.xwork2.LocaleProvider, Serializable, com.opensymphony.xwork2.TextProvider, com.opensymphony.xwork2.Validateable, com.opensymphony.xwork2.ValidationAware

---

```
public class PrepararFormularioEdicionIntervencionIncidenciaAction
extends com.opensymphony.xwork2.ActionSupport
```

Action encargado de consultar y cargar los datos de una intervencion existente que se pretende modificar

13.3.1.52 Clase PrepararFormularioNuevaIncidenciaAction

org.sigiccs.impl.incidencia.presentation

```
java.lang.Object
  └ com.opensymphony.xwork2.ActionSupport
      └
org.sigiccs.impl.incidencia.presentation.PrepararFormularioNuevaIncidenciaActi
on
```

All Implemented Interfaces:

com.opensymphony.xwork2.Action, com.opensymphony.xwork2.LocaleProvider, Serializable, com.opensymphony.xwork2.TextProvider, com.opensymphony.xwork2.Validateable, com.opensymphony.xwork2.ValidationAware

---

```
public class PrepararFormularioNuevaIncidenciaAction
extends com.opensymphony.xwork2.ActionSupport
```

Action encargado de precargar los datos necesarios en el formulario de alta de nueva incidencia

13.3.1.53 Clase PrepararFormularioRelacionIncidenciaAction

org.sigiccs.impl.incidencia.presentation

```
java.lang.Object
  └com.opensymphony.xwork2.ActionSupport
      └
```

org.sigiccs.impl.incidencia.presentation.PrepararFormularioRelacionIncidenciaAction

All Implemented Interfaces:

com.opensymphony.xwork2.Action, com.opensymphony.xwork2.LocaleProvider, Serializable, com.opensymphony.xwork2.TextProvider, com.opensymphony.xwork2.Validateable, com.opensymphony.xwork2.ValidationAware

---

```
public class PrepararFormularioRelacionIncidenciaAction
extends com.opensymphony.xwork2.ActionSupport
```

Action encargado de precargar los datos necesarios en el formulario de alta de relación entre incidencias

13.3.1.54 Clase ServicioManager

org.sigiccs.impl.servicio.business
java.lang.Object
└ **org.sigiccs.impl.servicio.business.ServicioManager**

All Implemented Interfaces:
ServicioManagerService

---

public class **ServicioManager**
extends Object
implements ServicioManagerService

Clase encargada de establecer la conexión entre la invocación de acciones propias de la gestión de servicios en la capa de presentación y la de persistencia

Constructor Detail

ServicioManager

public **ServicioManager**()

Method Detail

setServicioDataService

public void **setServicioDataService**(ServicioDataService servicioDataService)

Parameters:

servicioDataService - instancia del servicio de acceso a persistencia para las entidades Servicio

---

getAllServicios

public Vector<Servicio> **getAllServicios**()
throws SQLException

Description copied from interface: ServicioManagerService

Busca todos los servicios ofertados por la empresa

Specified by:

getAllServicios in interface ServicioManagerService

Returns:

Un Vector con todos los objetos Servicio encontrados

Throws:

SQLException

---

getServicioById

public Servicio **getServicioById**(int idServicio)
throws SQLException

Description copied from interface: ServicioManagerService

Busca toda la información de un servicio del catálogo en particular

Specified by:

getServicioById in interface ServicioManagerService

Parameters:

`idServicio` - El identificador del servicio que se desea consultar

Returns:

Un objeto Servicio con todos los datos del mismo

Throws:

`SQLException`

---

almacenarNuevoServicio

```
public int almacenarNuevoServicio(Servicio servicio)
                           throws SQLException
```

Description copied from interface: ServicioManagerService

Almacena un nuevo servcio en el catálogo del sistema

Specified by:

`almacenarNuevoServicio` in interface `ServicioManagerService`

Parameters:

`servicio` - El objeto Servicio con todos sus datos

Returns:

1 si el servicio fue almacenado correctamente

Throws:

`SQLException`

---

actualizarServicio

```
public int actualizarServicio(Servicio servicio)
                       throws SQLException
```

Description copied from interface: ServicioManagerService

Actualiza los datos de un servicio ya existente en el catálogo

Specified by:

`actualizarServicio` in interface `ServicioManagerService`

Parameters:

`servicio` - El objeto Servicio con los nuevos datos

Returns:

1 si el servicio fue modificado correctamente

Throws:

`SQLException`

---

removeServicio

```
public int removeServicio(int idServicio)
                   throws SQLException
```

Description copied from interface: ServicioManagerService

Marca como eliminado un servicio de los existentes en el catálogo

Specified by:

`removeServicio` in interface `ServicioManagerService`

Parameters:

`idServicio` - El identificador del servicio a eliminar

Returns:

1 si el servicio fue eliminado correctamente

Throws:

`SQLException`

---

getServiciosContratables

```
public Vector<Servicio> getServiciosContratables(int idContrato)
                                          throws SQLException
```

Description copied from interface: ServicioManagerService

Consulta el listado de servicios que pueden ser asociados a un contrato en particular

Specified by:

`getServiciosContratables` in interface `ServicioManagerService`

Parameters:

`idContrato` - El identificador del contrato en base al cual se realiza la consulta

Returns:

Un objeto Vector con todos los objetos Servicio que cumplen los requisitos para ser asociables al contrato

Throws:

`SQLException`

13.3.1.55 Clase TarifaManager

org.sigiccs.impl.servicio.business
java.lang.Object
└ **org.sigiccs.impl.servicio.business.TarifaManager**

All Implemented Interfaces:
TarifaManagerService

---

public class **TarifaManager**
extends Object
implements TarifaManagerService

Clase encargada de establecer la conexión entre la invocación de acciones propias de la gestión de tarifas asociadas a servicios en la capa de presentación y la de persistencia

Constructor Detail

TarifaManager

public **TarifaManager**()

Method Detail

setTarifaDataService

public void **setTarifaDataService**(TarifaDataService tarifaDataService)

Parameters:
tarifaDataService - instancia del servicio de acceso a persistencia para las entidades TarifaServicio

---

getAllTarifaServicios

public Vector<TarifaServicio> **getAllTarifaServicios**()
throws SQLException

Description copied from interface: TarifaManagerService
Busca todas las tarifas aplicables a los servicios del catálogo del sistema

Specified by:
getAllTarifaServicios in interface TarifaManagerService

Returns:
Un objeto Vector con todos los objetos TarifaServicio

Throws:
SQLException

---

getTarifaServicioById

public TarifaServicio **getTarifaServicioById**(int IDTarifaServicio)
throws SQLException

Description copied from interface: TarifaManagerService
Busca toda la información de una tarifa en particular

Specified by:
getTarifaServicioById in interface TarifaManagerService

Parameters:

`IDTarifaServicio` - El identificador de la tarifa a consultar

Returns:

Un objeto TarifaServicio con todos los datos de la misma

Throws:

`SQLException`

---

almacenarNuevaTarifaServicio

```
public int almacenarNuevaTarifaServicio(TarifaServicio tarifaServicio)
                                 throws SQLException
```

Description copied from interface: `TarifaManagerService`

Almacena una nueva tarifa en el catálogo de servicios

Specified by:

`almacenarNuevaTarifaServicio` in interface `TarifaManagerService`

Parameters:

`tarifaServicio` - El objeto TarifaServicio con todos los datos a almacenar

Returns:

1 si la tarifa fue almacenada correctamente

Throws:

`SQLException`

---

actualizarTarifaServicio

```
public int actualizarTarifaServicio(TarifaServicio tarifaServicio)
                             throws SQLException
```

Description copied from interface: `TarifaManagerService`

Actualiza las propiedades de una tarifa existente

Specified by:

`actualizarTarifaServicio` in interface `TarifaManagerService`

Parameters:

`tarifaServicio` - El objeto TarifaServicio a modificar

Returns:

1 si la tarifa fue actualizada correctamente

Throws:

`SQLException`

---

removeTarifaServicio

```
public int removeTarifaServicio(int idTarifaServicio)
                         throws SQLException
```

Description copied from interface: `TarifaManagerService`

Marca como eliminada una tarifa existente en el catálogo de servicios

Specified by:

`removeTarifaServicio` in interface `TarifaManagerService`

Parameters:

`idTarifaServicio` - El identificador de la tarifa

Returns:

1 si la tarifa se elimino correctamente

Throws:

`SQLException`

---

getTarifasDelServicio

```
public Vector<TarifaServicio> getTarifasDelServicio(int IDServicio,
                                                    int IDContrato)
                                             throws SQLException
```

Description copied from interface: TarifaManagerService

Busca todas las tarifas asociadas a un servicio vinculado a un contrato suscrito

Specified by:

`getTarifasDelServicio` in interface `TarifaManagerService`

Parameters:

`IDServicio` - El identificador del servicio

`IDContrato` - El identificador del contrato que contiene el servicio

Returns:

Un objeto Vector con todos los objetos TarifaServicio encontrados

Throws:

`SQLException`

13.3.1.56 Clase ServicioDAO

org.sigiccs.impl.servicio.persistence
java.lang.Object
└ **org.sigiccs.impl.servicio.persistence.ServicioDAO**

All Implemented Interfaces:

ServicioDataService

```
public class ServicioDAO
extends Object
implements ServicioDataService
```

Clase encargada de la implementación de todos los métodos declarados en la interfaz que implementa, encargados de efectuar las operaciones de la gestión de servicios en lo relativo a la consulta y modificación de datos

Constructor Detail

ServicioDAO

```
public ServicioDAO()
```

Method Detail

getAllServicios

```
public Vector<Servicio> getAllServicios()
                                  throws SQLException
```

Description copied from interface: ServicioDataService

Busca todos los servicios ofertados por la empresa

Specified by:

`getAllServicios` in interface `ServicioDataService`

Returns:

Un Vector con todos los objetos Servicio encontrados

Throws:

`SQLException`

---

getServicioById

```
public Servicio getServicioById(int idServicio)
                         throws SQLException
```

Description copied from interface: ServicioDataService

Busca toda la información de un servicio del catálogo en particular

Specified by:

`getServicioById` in interface `ServicioDataService`

Parameters:

`idServicio` - El identificador del servicio que se desea consultar

Returns:

Un objeto Servicio con todos los datos del mismo

Throws:

`SQLException`

---

almacenarNuevoServicio

```
public int almacenarNuevoServicio(Servicio servicio)
                           throws SQLException
```

Description copied from interface: ServicioDataService

Almacena un nuevo servcio en el catálogo del sistema

Specified by:

`almacenarNuevoServicio` in interface `ServicioDataService`

Parameters:

`servicio` - El objeto Servicio con todos sus datos

Returns:

1 si el servicio fue almacenado correctamente

Throws:

`SQLException`

---

actualizarServicio

```
public int actualizarServicio(Servicio servicio)
                       throws SQLException
```

Description copied from interface: ServicioDataService

Actualiza los datos de un servicio ya existente en el catálogo

Specified by:

`actualizarServicio` in interface `ServicioDataService`

Parameters:

`servicio` - El objeto Servicio con los nuevos datos

Returns:

1 si el servicio fue modificado correctamente

Throws:

`SQLException`

---

removeServicio

```
public int removeServicio(int idServicio)
                   throws SQLException
```

Description copied from interface: ServicioDataService

Marca como eliminado un servicio de los existentes en el catálogo

Specified by:

`removeServicio` in interface `ServicioDataService`

Parameters:

`idServicio` - El identificador del servicio a eliminar

Returns:

1 si el servicio fue eliminado correctamente

Throws:

`SQLException`

---

getServiciosContratables

```
public Vector<Servicio> getServiciosContratables(int IDContrato)
                                          throws SQLException
```

Description copied from interface: ServicioDataService

Consulta el listado de servicios que pueden ser asociados a un contrato en particular

Specified by:

`getServiciosContratables` in interface `ServicioDataService`

Returns:

Un objeto Vector con todos los objetos Servicio que cumplen los requisitos para ser asociables al contrato

Throws:

`SQLException`

13.3.1.57 Clase TarifaDAO

org.sigiccs.impl.servicio.persistence
java.lang.Object
└ **org.sigiccs.impl.servicio.persistence.TarifaDAO**

All Implemented Interfaces:

TarifaDataService

```
public class TarifaDAO
extends Object
implements TarifaDataService
```

Clase encargada de la implementación de todos los métodos declarados en la interfaz que implementa, encargados de efectuar las operaciones de la gestión de tarifas de servicios en lo relativo a la consulta y modificación de datos

Constructor Detail

TarifaDAO

```
public TarifaDAO()
```

Method Detail

getAllTarifaServicios

```
public Vector<TarifaServicio> getAllTarifaServicios()
                                              throws SQLException
```

Description copied from interface: TarifaDataService

Busca todas las tarifas aplicables a los servicios del catálogo del sistema

Specified by:

getAllTarifaServicios in interface TarifaDataService

Returns:

Un objeto Vector con todos los objetos TarifaServicio

Throws:

SQLException

---

getTarifaServicioById

```
public TarifaServicio getTarifaServicioById(int IDTarifaServicio)
                                     throws SQLException
```

Description copied from interface: TarifaDataService

Busca toda la información de una tarifa en particular

Specified by:

getTarifaServicioById in interface TarifaDataService

Parameters:

IDTarifaServicio - El identificador de la tarifa a consultar

Returns:

Un objeto TarifaServicio con todos los datos de la misma

Throws:

SQLException

---

getTarifasDelServicio

```
public Vector<TarifaServicio> getTarifasDelServicio(int iDServicio,
                                                    int iDContrato)
                                             throws SQLException
```

Description copied from interface: TarifaDataService

Busca todas las tarifas asociadas a un servicio vinculado a un contrato suscrito

Specified by:

`getTarifasDelServicio` in interface `TarifaDataService`

Returns:

Un objeto Vector con todos los objetos TarifaServicio encontrados

Throws:

`SQLException`

---

almacenarNuevaTarifaServicio

```
public int almacenarNuevaTarifaServicio(TarifaServicio tarifaServicio)
                                 throws SQLException
```

Description copied from interface: TarifaDataService

Almacena una nueva tarifa en el catálogo de servicios

Specified by:

`almacenarNuevaTarifaServicio` in interface `TarifaDataService`

Parameters:

`tarifaServicio` - El objeto TarifaServicio con todos los datos a almacenar

Returns:

1 si la tarifa fue almacenada correctamente

Throws:

`SQLException`

---

actualizarTarifaServicio

```
public int actualizarTarifaServicio(TarifaServicio tarifaServicio)
                             throws SQLException
```

Description copied from interface: TarifaDataService

Actualiza las propiedades de una tarifa existente

Specified by:

`actualizarTarifaServicio` in interface `TarifaDataService`

Parameters:

`tarifaServicio` - El objeto TarifaServicio a modificar

Returns:

1 si la tarifa fue actualizada correctamente

Throws:

`SQLException`

---

removeTarifaServicio

```
public int removeTarifaServicio(int idTarifaServicio)
                         throws SQLException
```

Description copied from interface: TarifaDataService

Marca como eliminada una tarifa existente en el catálogo de servicios

Specified by:

`removeTarifaServicio` in interface `TarifaDataService`

Parameters:

`idTarifaServicio` - El identificador de la tarifa

Returns:

1 si la tarifa se elimino correctamente

Throws:

`SQLException`

13.3.1.58 Clase ActualizarServicioAction

org.sigiccs.impl.servicio.presentation

```
java.lang.Object
  └com.opensymphony.xwork2.ActionSupport
      └org.sigiccs.impl.servicio.presentation.ActualizarServicioAction
```

All Implemented Interfaces:

com.opensymphony.xwork2.Action, com.opensymphony.xwork2.LocaleProvider, Serializable, com.opensymphony.xwork2.TextProvider, com.opensymphony.xwork2.Validateable, com.opensymphony.xwork2.ValidationAware

---

```
public class ActualizarServicioAction
extends com.opensymphony.xwork2.ActionSupport
```

Action encargado de recoger los cambios en las propiedades de un servicio e invocar a la siguiente capa para persistirlos

13.3.1.59 Clase ActualizarTarifaAction

org.sigiccs.impl.servicio.presentation

```
java.lang.Object
  └com.opensymphony.xwork2.ActionSupport
      └org.sigiccs.impl.servicio.presentation.ActualizarTarifaAction
```

All Implemented Interfaces:

com.opensymphony.xwork2.Action, com.opensymphony.xwork2.LocaleProvider, Serializable, com.opensymphony.xwork2.TextProvider, com.opensymphony.xwork2.Validateable, com.opensymphony.xwork2.ValidationAware

---

```
public class ActualizarTarifaAction
extends com.opensymphony.xwork2.ActionSupport
```

Action encargado de recoger los cambios en las propiedades de una tarifa e invocar a la siguiente capa para persistirlos

13.3.1.60 Clase CargarTarifasDeUnServicioAction

org.sigiccs.impl.servicio.presentation

```
java.lang.Object
  └com.opensymphony.xwork2.ActionSupport
      └org.sigiccs.impl.servicio.presentation.CargarTarifasDeUnServicioAction
```

All Implemented Interfaces:

com.opensymphony.xwork2.Action, com.opensymphony.xwork2.LocaleProvider, Serializable, com.opensymphony.xwork2.TextProvider, com.opensymphony.xwork2.Validateable, com.opensymphony.xwork2.ValidationAware

---

```
public class CargarTarifasDeUnServicioAction
extends com.opensymphony.xwork2.ActionSupport
```

Action encargado de consultar el listado de tarifas aplicables a un servicio determinado

13.3.1.61 Clase CargarTodasTarifasAction

org.sigiccs.impl.servicio.presentation

```
java.lang.Object
  └com.opensymphony.xwork2.ActionSupport
      └org.sigiccs.impl.servicio.presentation.CargarTodasTarifasAction
```

All Implemented Interfaces:

com.opensymphony.xwork2.Action, com.opensymphony.xwork2.LocaleProvider, Serializable, com.opensymphony.xwork2.TextProvider, com.opensymphony.xwork2.Validateable, com.opensymphony.xwork2.ValidationAware

---

```
public class CargarTodasTarifasAction
extends com.opensymphony.xwork2.ActionSupport
```

Action encargado de consultar el listado de todas las tarifas vigentes en el sistema

13.3.1.62 Clase CargarTodosServiciosAction

org.sigiccs.impl.servicio.presentation

```
java.lang.Object
  └com.opensymphony.xwork2.ActionSupport
      └org.sigiccs.impl.servicio.presentation.CargarTodosServiciosAction
```

All Implemented Interfaces:

com.opensymphony.xwork2.Action, com.opensymphony.xwork2.LocaleProvider, Serializable, com.opensymphony.xwork2.TextProvider, com.opensymphony.xwork2.Validateable, com.opensymphony.xwork2.ValidationAware

---

```
public class CargarTodosServiciosAction
extends com.opensymphony.xwork2.ActionSupport
```

Action encargado de consultar el listado de todos los servicios disponibles en el catálogo del sistema

13.3.1.63 Clase CrearServicioAction

org.sigiccs.impl.servicio.presentation

```
java.lang.Object
  └com.opensymphony.xwork2.ActionSupport
      └org.sigiccs.impl.servicio.presentation.CrearServicioAction
```

All Implemented Interfaces:

com.opensymphony.xwork2.Action, com.opensymphony.xwork2.LocaleProvider, Serializable, com.opensymphony.xwork2.TextProvider, com.opensymphony.xwork2.Validateable, com.opensymphony.xwork2.ValidationAware

---

```
public class CrearServicioAction
extends com.opensymphony.xwork2.ActionSupport
```

Action encargado de recoger los datos del formulario de alta de servicio e invocar a la siguiente capa para persistirlos

13.3.1.64 Clase CrearTarifaAction

org.sigiccs.impl.servicio.presentation

```
java.lang.Object
  └ com.opensymphony.xwork2.ActionSupport
      └ org.sigiccs.impl.servicio.presentation.CrearTarifaAction
```

All Implemented Interfaces:

com.opensymphony.xwork2.Action, com.opensymphony.xwork2.LocaleProvider, Serializable, com.opensymphony.xwork2.TextProvider, com.opensymphony.xwork2.Validateable, com.opensymphony.xwork2.ValidationAware

---

```
public class CrearTarifaAction
extends com.opensymphony.xwork2.ActionSupport
```

Action encargado de recoger los datos del formulario de alta de tarifa de servicio e invocar a la siguiente capa para persistirlos

13.3.1.65 Clase EliminarServicioAction

org.sigiccs.impl.servicio.presentation

```
java.lang.Object
  └ com.opensymphony.xwork2.ActionSupport
      └ org.sigiccs.impl.servicio.presentation.EliminarServicioAction
```

All Implemented Interfaces:

com.opensymphony.xwork2.Action, com.opensymphony.xwork2.LocaleProvider, Serializable, com.opensymphony.xwork2.TextProvider, com.opensymphony.xwork2.Validateable, com.opensymphony.xwork2.ValidationAware

---

```
public class EliminarServicioAction
extends com.opensymphony.xwork2.ActionSupport
```

Action encargado de marcar como eliminado un servicio existente

13.3.1.66 Clase EliminarTarifaAction

org.sigiccs.impl.servicio.presentation

```
java.lang.Object
  └ com.opensymphony.xwork2.ActionSupport
      └ org.sigiccs.impl.servicio.presentation.EliminarTarifaAction
```

All Implemented Interfaces:

com.opensymphony.xwork2.Action, com.opensymphony.xwork2.LocaleProvider, Serializable, com.opensymphony.xwork2.TextProvider, com.opensymphony.xwork2.Validateable, com.opensymphony.xwork2.ValidationAware

---

```
public class EliminarTarifaAction
extends com.opensymphony.xwork2.ActionSupport
```

Action encargado de marcar como eliminada una tarifa asociada a un servicio

13.3.1.67 Clase PrepararEdicionServicioAction

org.sigiccs.impl.servicio.presentation

```
java.lang.Object
  └com.opensymphony.xwork2.ActionSupport
      └org.sigiccs.impl.servicio.presentation.PrepararEdicionServicioAction
```

All Implemented Interfaces:

com.opensymphony.xwork2.Action, com.opensymphony.xwork2.LocaleProvider, Serializable, com.opensymphony.xwork2.TextProvider, com.opensymphony.xwork2.Validateable, com.opensymphony.xwork2.ValidationAware

---

```
public class PrepararEdicionServicioAction
extends com.opensymphony.xwork2.ActionSupport
```

Action encargado de consultar y cargar los datos de servicio existente que se pretende modificar

13.3.1.68 Clase PrepararEdicionTarifaAction

org.sigiccs.impl.servicio.presentation

```
java.lang.Object
  └com.opensymphony.xwork2.ActionSupport
      └org.sigiccs.impl.servicio.presentation.PrepararEdicionTarifaAction
```

All Implemented Interfaces:

com.opensymphony.xwork2.Action, com.opensymphony.xwork2.LocaleProvider, Serializable, com.opensymphony.xwork2.TextProvider, com.opensymphony.xwork2.Validateable, com.opensymphony.xwork2.ValidationAware

---

```
public class PrepararEdicionTarifaAction
extends com.opensymphony.xwork2.ActionSupport
```

Action encargado de consultar y cargar los datos de una tarifa de servicio existente que se pretende modificar

13.3.1.69 Clase PrepararFormularioNuevaTarifaAction

org.sigiccs.impl.servicio.presentation

```
java.lang.Object
  └com.opensymphony.xwork2.ActionSupport
      └
org.sigiccs.impl.servicio.presentation.PrepararFormularioNuevaTarifaAction
```

All Implemented Interfaces:

com.opensymphony.xwork2.Action, com.opensymphony.xwork2.LocaleProvider, Serializable, com.opensymphony.xwork2.TextProvider, com.opensymphony.xwork2.Validateable, com.opensymphony.xwork2.ValidationAware

---

```
public class PrepararFormularioNuevaTarifaAction
extends com.opensymphony.xwork2.ActionSupport
```

Action encargado de precargar los datos necesarios para el formulario de alta de una nueva tarifa

13.3.1.70 Interfaz ClienteManagerService

org.sigiccs.serv.cliente.business

All Known Implementing Classes:

ClienteManager

`public interface` **`ClienteManagerService`**

Interfaz que establece la signatura de los métodos que implementará ClienteManager

Method Detail

getAllClientes

`Vector<Cliente>` **`getAllClientes`**`()`
`throws SQLException`

Busca todos los clientes en el sistema

Returns:

Un objeto Vector con todos los objetos Cliente encontrados

Throws:

`SQLException`

---

getClienteById

`Cliente` **`getClienteById`**`(int ID)`
`throws SQLException`

Busca toda la información de un cliente en particular

Parameters:

`ID` - El Identificador del cliente a solicitar

Returns:

Un objeto Cliente con todos los datos del mismo

Throws:

`SQLException`

---

getClienteByNombre

`Cliente` **`getClienteByNombre`**`(String nombre)`
`throws SQLException`

Busca toda la información de un cliente en base a su nombre

Parameters:

`nombre` - Nombre del cliente a buscar

Returns:

Un objeto Cliente con todos los datos del mismo

Throws:

`SQLException`

---

almacenarNuevoCliente

`int` **`almacenarNuevoCliente`**`(Cliente cliente)`
`throws SQLException`

Almacena un nuevo cliente en el sistema

Parameters:

`cliente` - El objeto Cliente con todos sus datos

Returns:

1 si el cliente fue insertado correctamente

Throws:

`SQLException`

---

actualizarCliente

```
int actualizarCliente(Cliente cliente)
              throws SQLException
```

Actualiza los datos de un cliente ya existente en el sistema

Parameters:

`cliente` - El objeto Cliente con los nuevos datos

Returns:

1 si el cliente fue actualizado correctamente

Throws:

`SQLException`

---

removeCliente

```
int removeCliente(int idCliente)
           throws SQLException
```

Marca como eliminado un cliente existente

Parameters:

`idCliente` - Identificador del cliente a eliminar

Returns:

1 si el cliente fue eliminado correctamente

Throws:

`SQLException`

---

getProvinciasCliente

```
Vector<Provincia> getProvinciasCliente()
                                  throws SQLException
```

Método de utilidad para consultar el listado de provincias para los formularios de los clientes

Returns:

Un objeto Vector con todos los objetos Provincia encontrados

Throws:

`SQLException`

13.3.1.71 Clase Cliente

org.sigiccs.serv.cliente.model
java.lang.Object
└ **org.sigiccs.serv.cliente.model.Cliente**

public class **Cliente**
extends Object

Clase del modelo que representa a los objetos Cliente

Constructor Detail

Cliente

```
public Cliente()
```

Cliente

```
public Cliente(int iDCliente,
               String nombre,
               String direccionPostal,
               String poblacion,
               String provincia,
               String telefono,
               String fax,
               String logo)
```

Method Detail

getIDCliente

```
public int getIDCliente()
```

Returns:

El identificador del cliente

setIDCliente

```
public void setIDCliente(int iDCliente)
```

Parameters:

`iDCliente` - Identificador del cliente

getNombre

```
public String getNombre()
```

Returns:

El nombre del cliente

setNombre

```
public void setNombre(String nombre)
```

Parameters:

`nombre` - Nombre del cliente

getDireccionPostal

`public String` **`getDireccionPostal`**`()`

Returns:

La direccion postal del cliente

setDireccionPostal

`public void` **`setDireccionPostal`**`(String direccionPostal)`

Parameters:

`direccionPostal` - Direccion postal del cliente

getPoblacion

`public String` **`getPoblacion`**`()`

Returns:

La poblacion en la que se ubica el cliente

setPoblacion

`public void` **`setPoblacion`**`(String poblacion)`

Parameters:

`poblacion` - Poblacion del cliente

getProvincia

`public String` **`getProvincia`**`()`

Returns:

La provincia en la que se ubica el cliente

setProvincia

`public void` **`setProvincia`**`(String provincia)`

Parameters:

`provincia` - Provincia del cliente

getTelefono

`public String` **`getTelefono`**`()`

Returns:

Telefono principal de contacto

setTelefono

`public void` **`setTelefono`**`(String telefono)`

Parameters:

`telefono` - Telefono de contacto

getFax

`public String` **`getFax`**`()`

Returns:

Numero de fax (opcional)

---

setFax

```
public void setFax(String fax)
```

Parameters:

`fax` - Numero de fax

---

getLogo

```
public String getLogo()
```

Returns:

El identificador de la ruta del logo del cliente

---

setLogo

```
public void setLogo(String logo)
```

Parameters:

`logo` - El identificador de la ruta del logo del cliente

13.3.1.72 Clase Provincia

org.sigiccs.serv.cliente.model
java.lang.Object
  └ **org.sigiccs.serv.cliente.model.Provincia**

public class **Provincia**
extends Object

Clase del modelo que representa al objeto Provincia, asociado al Cliente

Constructor Detail

Provincia

public **Provincia**()

Provincia

public **Provincia**(int iDProvincia,
String provincia)

Method Detail

getIDProvincia

public int **getIDProvincia**()

Returns:

El identificador de la provincia

setIDProvincia

public void **setIDProvincia**(int iDProvincia)

Parameters:

iDProvincia - El identificador de la provincia

getProvincia

public String **getProvincia**()

Returns:

El nombre completo de la provincia

setProvincia

public void **setProvincia**(String provincia)

Parameters:

provincia - El nombre completo de la provincia

13.3.1.73 Interfaz ClienteDataService

org.sigiccs.serv.cliente.persistence

All Known Implementing Classes:

ClienteDAO

`public interface` **`ClienteDataService`**

Interfaz que establece la signatura de los métodos que implementará ClienteDAO

Method Detail

getAllClientes

`Vector<Cliente>` **`getAllClientes`**`()`
`throws SQLException`

Busca todos los clientes en el sistema

Returns:

Un objeto Vector con todos los objetos Cliente encontrados

Throws:

`SQLException`

---

getClienteById

`Cliente` **`getClienteById`**`(int ID)`
`throws SQLException`

Busca toda la información de un cliente en particular

Parameters:

`ID` - El Identificador del cliente a solicitar

Returns:

Un objeto Cliente con todos los datos del mismo

Throws:

`SQLException`

---

getClienteByNombre

`Cliente` **`getClienteByNombre`**`(String nombre)`
`throws SQLException`

Busca toda la información de un cliente en base a su nombre

Parameters:

`nombre` - Nombre del cliente a buscar

Returns:

Un objeto Cliente con todos los datos del mismo

Throws:

`SQLException`

---

almacenarNuevoCliente

`int` **`almacenarNuevoCliente`**`(Cliente cliente)`
`throws SQLException`

Almacena un nuevo cliente en el sistema

Parameters:

`cliente` - El objeto Cliente con todos sus datos

Returns:

1 si el cliente fue insertado correctamente

Throws:

`SQLException`

---

actualizarCliente

```
int actualizarCliente(Cliente cliente)
                throws SQLException
```

Actualiza los datos de un cliente ya existente en el sistema

Parameters:

`cliente` - El objeto Cliente con los nuevos datos

Returns:

1 si el cliente fue actualizado correctamente

Throws:

`SQLException`

---

removeCliente

```
int removeCliente(int idCliente)
           throws SQLException
```

Marca como eliminado un cliente existente

Parameters:

`idCliente` - Identificador del cliente a eliminar

Returns:

1 si el cliente fue eliminado correctamente

Throws:

`SQLException`

---

getProvinciasCliente

```
Vector<Provincia> getProvinciasCliente()
                                  throws SQLException
```

Método de utilidad para consultar el listado de provincias para los formularios de los clientes

Returns:

Un objeto Vector con todos los objetos Provincia encontrados

Throws:

`SQLException`

13.3.1.74 Interfaz ContratoManagerService

org.sigiccs.serv.contrato.business

All Known Implementing Classes:

ContratoManager

`public interface` **`ContratoManagerService`**

Interfaz que establece la signatura de los métodos que implementará ContratoManager

Method Detail

getAllContratos

`Vector<Contrato>` **`getAllContratos`**`()`
`throws SQLException`

Busca todos los contratos suscritos por clientes que han sido almacenados en el sistema

Returns:

Un objeto Vector con todos los objetos Contrato encontrados

Throws:

`SQLException`

---

getContratoById

`Contrato` **`getContratoById`**`(int ID)`
`throws SQLException`

Busca toda la información de un contrato en particular

Parameters:

`ID` - El Identificador del contrato a solicitar

Returns:

Un objeto Contrato con todos los datos del mismo

Throws:

`SQLException`

---

almacenarNuevoContrato

`int` **`almacenarNuevoContrato`**`(Contrato contrato)`
`throws SQLException`

Almacena un nuevo contrato en el sistema

Parameters:

`contrato` - El objeto Contrato con todos sus datos

Returns:

1 si el contrato fue insertado correctamente

Throws:

`SQLException`

---

actualizarContrato

`int` **`actualizarContrato`**`(Contrato contrato)`
`throws SQLException`

Actualiza los datos de un contrato ya existente en el sistema

Parameters:

`contrato` - El objeto Contrato con los nuevos datos

Returns:

1 si el contrato fue actualizado correctamente

Throws:

`SQLException`

---

removeContrato

```
int removeContrato(int idContrato)
            throws SQLException
```

Marca como eliminado un contrato existente

Returns:

1 si el contrato fue eliminado correctamente

Throws:

`SQLException`

---

getTiposFacturacion

```
Vector<TipoFacturacion> getTiposFacturacion()
                                    throws SQLException
```

Consulta el listado de Tipos de Facturacion aplicables a los contratos

Returns:

Un objeto Vector con todos los Tipos de Facturacion disponibles en el sistema

Throws:

`SQLException`

---

getContratosCliente

```
Vector<Contrato> getContratosCliente(int iDCliente)
                              throws SQLException
```

Devuelve todos los contratos suscritos por un cliente en particular

Parameters:

`iDCliente` - Identificador del cliente en base al cual se realiza la consulta de contratos suscritos

Returns:

Un objeto Vector con todos los contratos suscritos por el cliente

Throws:

`SQLException`

13.3.1.75 Interfaz ContratoServiciosManagerService

org.sigiccs.serv.contrato.business

All Known Implementing Classes:

ContratoServiciosManager

`public interface` **`ContratoServiciosManagerService`**

Interfaz que establece la signatura de los métodos que implementa ContratoServiciosManager

Method Detail

getAllServiciosContrato

```
Vector<ContratoServicio> getAllServiciosContrato(int idContrato)
                                                 throws SQLException
```

Busca todos los servicios asociados a un contrato suscrito por un cliente

Parameters:

`idContrato` - Identificador del contrato a consultar

Returns:

Un objeto Vector con todos los objetos ContratoServicio, que representan los servicios contratados por el cliente en ese contrato

Throws:

`SQLException`

---

almacenarNuevoServicioContrato

```
int almacenarNuevoServicioContrato(ContratoServicio contratoServicio)
                                   throws SQLException
```

Asocia un servicio del catálogo a un contrato de servicios

Parameters:

`contratoServicio` - El objeto Servicio contratable

Returns:

1 si el servicio fue asociado correctamente al contrato

Throws:

`SQLException`

---

actualizarServicioContrato

```
int actualizarServicioContrato(ContratoServicio contratoServicio)
                               throws SQLException
```

Actualiza los parámetros de una suscripción de servicio en un contrato

Parameters:

`contratoServicio` - El objeto Servicio asociado al contrato

Returns:

1 si la asociación del servicio se actualizó correctamente

Throws:

`SQLException`

---

removeServicioContrato

```
int removeServicioContrato(int idContrato,
                           int idServicio)
                    throws SQLException
```

Marca como eliminado un servicio asociado a un contrato

Parameters:

`idContrato` - Identificador del contrato que contiene el servicio

`idServicio` - Identificador del servicio asociado a eliminar

Returns:

1 si el servicio se desvinculó del contrato correctamente

Throws:

`SQLException`

---

getServicioContrato

```
ContratoServicio getServicioContrato(int iDContrato,
                                     int iDServicio)
                              throws SQLException
```

Busca toda la información de una asociación de servicio a contrato en particular

Parameters:

`iDContrato` - Identificador del contrato a consultar

`iDServicio` - Identificador del servicio dentro del contrato

Returns:

Un objeto ContratoServicio con la información del mismo

Throws:

`SQLException`

13.3.1.76 Clase Contrato

org.sigiccs.serv.contrato.model
java.lang.Object
└ **org.sigiccs.serv.contrato.model.Contrato**

public class **Contrato**
extends Object

Clase del modelo que representa a los objetos Contrato

Constructor Detail

Contrato

```
public Contrato()
```

Contrato

```
public Contrato(int iDContrato,
                int iDCliente,
                int iDTipoFacturacion,
                String titulo,
                String descripcion,
                String fechaInicio,
                String fechaFin,
                String fechaAlta)
```

Method Detail

getIDContrato

```
public int getIDContrato()
```

Returns:

El identificador del contrato

setIDContrato

```
public void setIDContrato(int iDContrato)
```

Parameters:

`iDContrato` - Identificador del contrato

getIDCliente

```
public int getIDCliente()
```

Returns:

El identificador del cliente que suscribe el contrato

setIDCliente

```
public void setIDCliente(int iDCliente)
```

Parameters:

`iDCliente` - Identificador del cliente que suscribe el contrato

getIDTipoFacturacion

`public int` **`getIDTipoFacturacion`**`()`

Returns:

El tipo de facturacion aplicado al contrato

setIDTipoFacturacion

`public void` **`setIDTipoFacturacion`**`(int iDTipoFacturacion)`

Parameters:

`iDTipoFacturacion` - Tipo de facturacion aplicado al contrato

getTitulo

`public String` **`getTitulo`**`()`

Returns:

El titulo descriptivo del contrato

setTitulo

`public void` **`setTitulo`**`(String titulo)`

Parameters:

`titulo` - Titulo descriptivo del contrato

getDescripcion

`public String` **`getDescripcion`**`()`

Returns:

La descripción detallada del contrato

setDescripcion

`public void` **`setDescripcion`**`(String descripcion)`

Parameters:

`descripcion` - Descripción detallada del contrato

getFechaInicio

`public String` **`getFechaInicio`**`()`

Returns:

La fecha de inicio del contrato

setFechaInicio

`public void` **`setFechaInicio`**`(String fechaInicio)`

Parameters:

`fechaInicio` - Fecha de inicio del contrato

getFechaFin

`public String` **`getFechaFin`**`()`

Returns:

La fecha de expiración del contrato

---

setFechaFin

```
public void setFechaFin(String fechaFin)
```

Parameters:

`fechaFin` - fecha de expiración del contrato

---

getFechaAlta

```
public String getFechaAlta()
```

Returns:

La fecha de alta efectiva del contrato

---

setFechaAlta

```
public void setFechaAlta(String fechaAlta)
```

Parameters:

`fechaAlta` - Fecha de alta efectiva del contrato

---

getClienteAsociado

```
public String getClienteAsociado()
```

Returns:

El nombre del cliente que suscribe el contrato

---

setClienteAsociado

```
public void setClienteAsociado(String clienteAsociado)
```

Parameters:

`clienteAsociado` - Nombre del cliente que suscribe el contrato

---

getTipoFacturacion

```
public String getTipoFacturacion()
```

Returns:

Descripción textual del tipo de facturacion aplicado

---

setTipoFacturacion

```
public void setTipoFacturacion(String tipoFacturacion)
```

Parameters:

`tipoFacturacion` - descripción textual del tipo de facturacion aplicado

---

getImporteTotal

```
public Float getImporteTotal()
```

Returns:

La suma de los importes de los servicios contratados

---

setImporteTotal

```
public void setImporteTotal(Float importeTotal)
```

Parameters:

`importeTotal` - Suma de los importes de los servicios contratados

getEstado

```
public String getEstado()
```

Returns:

El estado actual del contrato

setEstado

```
public void setEstado(String estado)
```

Parameters:

`estado` - Estado actual del contrato

13.3.1.77 Clase ContratoServicio

org.sigiccs.serv.contrato.model
java.lang.Object
└ **org.sigiccs.serv.contrato.model.ContratoServicio**

public class **ContratoServicio**
extends Object

Clase del modelo que representa el vínculo entre un contrato y un servicio asociado a éste

Constructor Detail

ContratoServicio

```
public ContratoServicio()
```

ContratoServicio

```
public ContratoServicio(int iDContrato,
                        int iDServicio,
                        int numHoras,
                        float precioHora,
                        boolean desplazamiento,
                        String descripcion,
                        String servicioDescripcion)
```

Method Detail

getIDContrato

```
public int getIDContrato()
```

Returns:

El identificador del contrato al que se hace referencia

setIDContrato

```
public void setIDContrato(int iDContrato)
```

Parameters:

`iDContrato` - Identificador del contrato al que se hace referencia

getIDServicio

```
public int getIDServicio()
```

Returns:

El identificador del servicio al que se hace referencia

setIDServicio

```
public void setIDServicio(int iDServicio)
```

Parameters:

`iDServicio` - Identificador del servicio al que se hace referencia

getNumHoras

`public int` **`getNumHoras`**`()`

Returns:

El numero de horas contratadas de un servicio en particular

setNumHoras

`public void` **`setNumHoras`**`(int numHoras)`

Parameters:

`numHoras` - Numero de horas contratadas de un servicio en particular

getPrecioHora

`public float` **`getPrecioHora`**`()`

Returns:

El precio por hora del servicio, expresado en Euros

setPrecioHora

`public void` **`setPrecioHora`**`(float precioHora)`

Parameters:

`precioHora` - Precio por hora del servicio, expresado en Euros

getDesplazamiento

`public boolean` **`getDesplazamiento`**`()`

Returns:

El desplazamiento al cliente esta (o no) incluido en el precio

setDesplazamiento

`public void` **`setDesplazamiento`**`(boolean desplazamiento)`

Parameters:

`desplazamiento` - Desplazamiento al cliente (o no) incluido en el precio

getDescripcion

`public String` **`getDescripcion`**`()`

Returns:

La descripción detallada de la asociación

setDescripcion

`public void` **`setDescripcion`**`(String descripcion)`

Parameters:

`descripcion` - Descripción detallada de la asociación

getServicioDescripcion

`public String` **`getServicioDescripcion`**`()`

Returns:

La descripción detallada del servicio, particularizada para el contrato

setServicioDescripcion

```
public void setServicioDescripcion(String servicioDescripcion)
```

Parameters:

`servicioDescripcion` - Descripción detallada del servicio, particularizada para el contrato

13.3.1.78 Clase TipoFacturacion

org.sigiccs.serv.contrato.model
java.lang.Object
└ **org.sigiccs.serv.contrato.model.TipoFacturacion**

public class **TipoFacturacion**
extends Object

Clase del modelo que representa los tipos de facturacion aplicables a los contratos suscritos por los clientes

Constructor Detail

TipoFacturacion

public **TipoFacturacion**()

Method Detail

getIDTipoFacturacion

public int **getIDTipoFacturacion**()

Returns:

El identificador del tipo de facturacion

setIDTipoFacturacion

public void **setIDTipoFacturacion**(int iDTipoFacturacion)

Parameters:

iDTipoFacturacion - Identificador del tipo de facturacion

getDescripcion

public String **getDescripcion**()

Returns:

La descripción textual del tipo de facturacion

setDescripcion

public void **setDescripcion**(String descripcion)

Parameters:

descripcion - Descripción textual del tipo de facturacion

13.3.1.79 Interfaz ContratoDataService

org.sigiccs.serv.contrato.persistence

All Known Implementing Classes:

ContratoDAO

`public interface` **`ContratoDataService`**

Interfaz que establece la signatura de los métodos que implementará ContratoDAO

Method Detail

getAllContratos

`Vector<Contrato>` **`getAllContratos`**`()`
`throws SQLException`

Busca todos los contratos suscritos por clientes que han sido almacenados en el sistema

Returns:

Un objeto Vector con todos los objetos Contrato encontrados

Throws:

`SQLException`

---

getContratoById

`Contrato` **`getContratoById`**`(int ID)`
`throws SQLException`

Busca toda la información de un contrato en particular

Parameters:

`ID` - El Identificador del contrato a solicitar

Returns:

Un objeto Contrato con todos los datos del mismo

Throws:

`SQLException`

---

almacenarNuevoContrato

`int` **`almacenarNuevoContrato`**`(Contrato Contrato)`
`throws SQLException`

Almacena un nuevo contrato en el sistema

Returns:

1 si el contrato fue insertado correctamente

Throws:

`SQLException`

---

actualizarContrato

`int` **`actualizarContrato`**`(Contrato Contrato)`
`throws SQLException`

Actualiza los datos de un contrato ya existente en el sistema

Returns:

1 si el contrato fue actualizado correctamente

Throws:

`SQLException`

removeContrato

```
int removeContrato(int idContrato)
            throws SQLException
```

Marca como eliminado un cliente existente

Returns:

1 si el contrato fue eliminado correctamente

Throws:

SQLException

---

getTiposFacturacion

```
Vector<TipoFacturacion> getTiposFacturacion()
                                    throws SQLException
```

Consulta el listado de Tipos de Facturacion aplicables a los contratos

Returns:

Un objeto Vector con todos los Tipos de Facturacion disponibles en el sistema

Throws:

SQLException

---

getContratosCliente

```
Vector<Contrato> getContratosCliente(int iDCliente)
                              throws SQLException
```

Devuelve todos los contratos suscritos por un cliente en particular

Parameters:

`iDCliente` - Identificador del cliente en base al cual se realiza la consulta de contratos suscritos

Returns:

Un objeto Vector con todos los contratos suscritos por el cliente

Throws:

SQLException

13.3.1.80 Interfaz ContratoServiciosDataService

org.sigiccs.serv.contrato.persistence

All Known Implementing Classes:

ContratoServiciosDAO

`public interface` **`ContratoServiciosDataService`**

Interfaz que establece la signatura de los métodos que implementará ContratoServiciosDAO

Method Detail

getAllServiciosContrato

```
Vector<ContratoServicio> getAllServiciosContrato(int idContrato)
                                                 throws SQLException
```

Busca todos los servicios asociados a un contrato suscrito por un cliente

Parameters:

`idContrato` - Identificador del contrato a consultar

Returns:

Un objeto Vector con todos los objetos ContratoServicio, que representan los servicios contratados por el cliente en ese contrato

Throws:

`SQLException`

---

almacenarNuevoServicioContrato

```
int almacenarNuevoServicioContrato(ContratoServicio contratoServicio)
                                   throws SQLException
```

Asocia un servicio del catálogo a un contrato de servicios

Parameters:

`contratoServicio` - El objeto Servicio contratable

Returns:

1 si el servicio fue asociado correctamente al contrato

Throws:

`SQLException`

---

actualizarServicioContrato

```
int actualizarServicioContrato(ContratoServicio contratoServicio)
                               throws SQLException
```

Actualiza los parámetros de una suscripción de servicio en un contrato

Parameters:

`contratoServicio` - El objeto Servicio asociado al contrato

Returns:

1 si la asociación del servicio se actualizó correctamente

Throws:

`SQLException`

---

removeServicioContrato

```
int removeServicioContrato(int idContrato,
                           int idServicio)
                    throws SQLException
```

Marca como eliminado un servicio asociado a un contrato

Parameters:

`idContrato` - Identificador del contrato que contiene el servicio

`idServicio` - Identificador del servicio asociado a eliminar

Returns:

1 si el servicio se desvinculó del contrato correctamente

Throws:

`SQLException`

---

getServicioContrato

```
ContratoServicio getServicioContrato(int iDContrato,
                                     int iDServicio)
                              throws SQLException
```

Busca toda la información de una asociación de servicio a contrato en particular

Parameters:

`iDContrato` - Identificador del contrato a consultar

`iDServicio` - Identificador del servicio dentro del contrato

Returns:

Un objeto ContratoServicio con la información del mismo

Throws:

`SQLException`

13.3.1.81 Interfaz DocumentoIncidenciaManagerService

org.sigiccs.serv.incidencia.business

All Known Implementing Classes:

DocumentoIncidenciaManager

`public interface` **`DocumentoIncidenciaManagerService`**

Interfaz que establece la signatura de los métodos que implementará DocumentoIncidenciaManager

Method Detail

getDocumentoIncidenciaByID

`DocumentoIncidencia` **`getDocumentoIncidenciaByID`**`(int idDocumento) throws SQLException`

Busca un adjunto a una incidencia en base al identificador del fichero

Parameters:

`idDocumento` - Identificador del fichero adjunto

Returns:

Un objeto DocumentoIncidencia con todos los datos del mismo

Throws:

`SQLException`

---

getAllDocumentosIncidencia

`Vector<DocumentoIncidencia>` **`getAllDocumentosIncidencia`**`(int idIncidencia) throws SQLException`

Busca todos los adjuntos a una incidencia en particular

Parameters:

`idIncidencia` - Identificador de la incidencia para la que se buscaran los documentos adjuntos

Returns:

Un objeto Vector con todos los objetos DocumentoIncidencia asociados a la incidencia

Throws:

`SQLException`

---

almacenarNuevoDocumentoIncidencia

`int` **`almacenarNuevoDocumentoIncidencia`**`(DocumentoIncidencia documentoIncidencia) throws SQLException`

Almacena un fichero adjunto a una incidencia, incluidos sus datos descriptivos

Parameters:

`documentoIncidencia` - El objeto que representa al adjunto

Returns:

1 si el fichero fue almacenado correctamente

Throws:

`SQLException`

---

eliminarDocumentoIncidencia

```
int eliminarDocumentoIncidencia(int idDocumento)
                          throws SQLException
```

Marca como eliminado un fichero adjunto a una incidencia

Parameters:

`idDocumento` - El identificador del adjunto a eliminar

Returns:

1 si el adjunto fue eliminado correctamente

Throws:

`SQLException`

eliminarDocumentosIncidencia

```
int eliminarDocumentosIncidencia(int iDIncidencia)
                           throws SQLException
```

Marca como eliminados todos los adjuntos a una incidencia

Parameters:

`iDIncidencia` - Identificador de la incidencia a tratar

Returns:

1 si los adjuntos fueron eliminados correctamente

Throws:

`SQLException`

13.3.1.82 Interfaz IncidenciaManagerService

org.sigiccs.serv.incidencia.business

All Known Implementing Classes:

IncidenciaManager

`public interface` **`IncidenciaManagerService`**

Interfaz que establece la signatura de los métodos que implementará IncidenciaManager

Method Detail

getIncidenciaByID

`Incidencia` **`getIncidenciaByID`**`(int IDIncidencia)`
`throws SQLException`

Busca toda la información asociada a una incidencia

Parameters:

`IDIncidencia` - El identificador de la incidencia

Returns:

Un objeto Incidencia con todos los datos de la misma

Throws:

`SQLException`

---

getAllIncidencias

`Vector<Incidencia>` **`getAllIncidencias`**`()`
`throws SQLException`

Busca todas las incidencias dadas de alta en el sistema

Returns:

Un objeto Vector con todos los objetos Incidencia encontrados

Throws:

`SQLException`

---

almacenarNuevaIncidencia

`int` **`almacenarNuevaIncidencia`**`(Incidencia incidencia)`
`throws SQLException`

Almacena una nueva incidencia en el sistema

Parameters:

`incidencia` - El objeto incidencia con todos los datos proporcionados en su creacion

Returns:

1 si la incidencia fue almacenada correctamente

Throws:

`SQLException`

---

eliminarIncidencia

`int` **`eliminarIncidencia`**`(int IDIncidencia)`
`throws SQLException`

Marca como eliminada una incidencia existente

Parameters:

`IDIncidencia` - Identificador de la incidencia a eliminar

Returns:

1 si la incidencia fue eliminada correctamente

Throws:

`SQLException`

---

actualizarIncidencia

```
int actualizarIncidencia(Incidencia incidencia)
                  throws SQLException
```

Actualiza los datos de una incidencia existente

Parameters:

`incidencia` - El objeto incidencia con los nuevos datos

Returns:

1 si la incidencia fue modificada correctamente

Throws:

`SQLException`

---

getAllIncidenciasCliente

```
Vector<Incidencia> getAllIncidenciasCliente(int iDCliente)
                                    throws SQLException
```

Busca todas las incidencias asociadas a un cliente en particular

Parameters:

`iDCliente` - Identificador del cliente en base al cual se realiza la busqueda

Returns:

Un objeto Vector con todos los objetos Incidencia encontrados

Throws:

`SQLException`

---

getTiposIncidencia

```
Vector<TipoIncidencia> getTiposIncidencia()
                                  throws SQLException
```

Consulta el listado de Tipos de Incidencia existentes

Returns:

Un objeto Vector con todos los objetos TipoIncidencia encontrados

Throws:

`SQLException`

---

getPrioridades

```
Vector<Prioridad> getPrioridades()
                         throws SQLException
```

Consulta el listado de Tipos de Prioridad aplicables a las incidencias

Returns:

Un objeto Vector con todos los objetos Prioridad encontrados

Throws:

`SQLException`

13.3.1.83 Interfaz IntervencionManagerService

org.sigiccs.serv.incidencia.business

All Known Implementing Classes:

IntervencionManager

`public interface` **`IntervencionManagerService`**

Interfaz que establece la signatura de los métodos que implementará IntervencionManager

Method Detail

getIntervencionByID

`Intervencion` **`getIntervencionByID`**`(int idIntervencion) throws SQLException`

Busca una intervencion en una incidencia en base al identificador de la misma

Parameters:

`idIntervencion` - Identificador de la intervencion a buscar

Returns:

Un objeto Intervencion con los datos de la misma

Throws:

`SQLException`

---

getAllIntervencionesIncidencia

`Vector<Intervencion>` **`getAllIntervencionesIncidencia`**`(int idIncidencia) throws SQLException`

Busca todas las intervenciones efectuadas en una incidencia en particular

Parameters:

`idIncidencia` - Identificador de la incidencia en la que se buscara

Returns:

Un objeto Vector con todos los objetos Intervencion encontrados

Throws:

`SQLException`

---

almacenarNuevaIntervencion

`int` **`almacenarNuevaIntervencion`**`(Intervencion intervencion) throws SQLException`

Almacena una intervencion en una incidencia

Parameters:

`intervencion` - El objeto que representa a la intervencion

Returns:

1 si la intervencion fue almacenada correctamente

Throws:

`SQLException`

---

eliminarIntervencion

`int` **`eliminarIntervencion`**`(int idIntervencion) throws SQLException`

Marca como eliminada una intervencion existente en una incidencia

Parameters:

`idIntervencion` - Identificador de la intervencion a eliminar

Returns:

1 si la intervencion fue eliminada correctamente

Throws:

`SQLException`

---

actualizarIntervencion

```
int actualizarIntervencion(Intervencion intervencion)
                    throws SQLException
```

Actualiza las propiedades de una intervencion existente

Parameters:

`intervencion` - El objeto intervencion con los nuevos datos

Returns:

1 si la intervencion fue modificada correctamente

Throws:

`SQLException`

---

eliminarIntervencionesIncidencia

```
int eliminarIntervencionesIncidencia(int iDIncidencia)
                                throws SQLException
```

Marca como eliminadas todas las intervenciones de una incidencia

Parameters:

`iDIncidencia` - Identificador de la incidencia en la que se eliminaran sus intervenciones

Returns:

1 si las intervenciones fueron eliminadas correctamente

Throws:

`SQLException`

13.3.1.84 Interfaz RelacionIncidenciaManagerService

org.sigiccs.serv.incidencia.business

All Known Implementing Classes:

RelacionIncidenciaManager

`public interface` **`RelacionIncidenciaManagerService`**

Interfaz que establece la signatura de los métodos que implementará RelacionIncidenciaManager

Method Detail

getRelacionIncidenciaByID

`RelacionIncidencia` **`getRelacionIncidenciaByID`**`(int iDRelacion) throws SQLException`

Consulta todos los datos de una relación en particular

Parameters:

`iDRelacion` - Identificador de la relaicon a consultar

Returns:

Un objeto RelacionIncidencia con todos los datos de la relación

Throws:

`SQLException`

---

getRelacionesIncidenciaMain

`Vector<RelacionIncidencia>` **`getRelacionesIncidenciaMain`**`(int iDIncidenciaMain) throws SQLException`

Consulta todas las relaciones en las que una incidencia es origen de las mismas

Parameters:

`iDIncidenciaMain` - Identificador de la incidencia origen

Returns:

Un objeto Vector con todos los objetos RelacionIncidencia encontrados

Throws:

`SQLException`

---

getRelacionesIncidenciaSub

`Vector<RelacionIncidencia>` **`getRelacionesIncidenciaSub`**`(int iDIncidenciaSub) throws SQLException`

Consulta todas las relaciones en las que una incidencia es destino de las mismas

Returns:

Un objeto Vector con todos los objetos RelacionIncidencia encontrados

Throws:

`SQLException`

---

almacenarNuevaRelacionIncidencia

`int` **`almacenarNuevaRelacionIncidencia`**`(RelacionIncidencia relacionIncidencia) throws SQLException`

Establece una nueva relación para la incidencia

Parameters:

`relacionIncidencia` - El objeto que representa a la relación

Returns:

1 si la relación fue establecida correctamente

Throws:

`SQLException`

---

eliminarRelacionIncidencia

```
int eliminarRelacionIncidencia(int iDRelacion)
                        throws SQLException
```

Marca como eliminada una relación en particular

Parameters:

`iDRelacion` - Identificador de la relación a eliminar

Returns:

1 si la relación fue eliminada correctamente

Throws:

`SQLException`

---

getTiposRelaciones

```
Vector<TipoRelacion> getTiposRelaciones()
                                  throws SQLException
```

Consulta el listado de Tipos de Relación entre incidencias existentes

Returns:

Un objeto Vector con todos los objetos TipoRelacion encontrados

Throws:

`SQLException`

---

getTipoRelacionByID

```
TipoRelacion getTipoRelacionByID(int iDTipoRelacion)
                          throws SQLException
```

Devuelve un Tipo de Relación en base a su identificador

Parameters:

`iDTipoRelacion` - Identificador del tipo de relación a consultar

Returns:

Un objeto TipoRelacion con todos los datos del mismo

Throws:

`SQLException`

---

esRelacionable

```
boolean esRelacionable(RelacionIncidencia relación)
                throws SQLException
```

Comprueba si una incidencia es relacionable con otra

Parameters:

`relación` - Atributos de la relación que se pretende establecer

Returns:

true si las incidencias son relacionables. false en caso contrario

Throws:

`SQLException`

---

eliminarRelacionesIncidencia

```
int eliminarRelacionesIncidencia(int iDIncidencia)
                          throws SQLException
```

Marca como eliminadas todas las relaciones directas de una incidencia

Parameters:

`iDIncidencia` - Identificador de la incidencia para la que se eliminaran sus relaciones

Returns:

1 si las relaciones fueron eliminadas correctamente

Throws:

`SQLException`

13.3.1.85 Clase DocumentoIncidencia

org.sigiccs.serv.incidencia.model
java.lang.Object
└ **org.sigiccs.serv.incidencia.model.DocumentoIncidencia**

public class **DocumentoIncidencia**
extends Object

Clase del modelo que representa los documentos adjuntos a una incidencia

Constructor Detail

DocumentoIncidencia

```
public DocumentoIncidencia()
```

DocumentoIncidencia

```
public DocumentoIncidencia(int iDDocumento,
                           int iDIncidencia,
                           String descripcion,
                           String ruta,
                           boolean eliminado)
```

Method Detail

getIDDocumento

```
public int getIDDocumento()
```

Returns:

El identificador del documento adjunto

setIDDocumento

```
public void setIDDocumento(int iDDocumento)
```

Parameters:

`iDDocumento` - identificador del documento adjunto

getIDIncidencia

```
public int getIDIncidencia()
```

Returns:

El identificador de la incidencia a la que se asocia el fichero

setIDIncidencia

```
public void setIDIncidencia(int iDIncidencia)
```

Parameters:

`iDIncidencia` - identificador de la incidencia a la que se asocia el fichero

getDescripcion

`public String` **`getDescripcion`**`()`

Returns:

La descripción del fichero adjunto

setDescripcion

`public void` **`setDescripcion`**`(String descripcion)`

Parameters:

`descripcion` - descripción del fichero adjunto

getRuta

`public String` **`getRuta`**`()`

Returns:

La ruta del fichero en el sistema de archivos del servidor

setRuta

`public void` **`setRuta`**`(String ruta)`

Parameters:

`ruta` - ruta del fichero en el sistema de archivos del servidor

isEliminado

`public boolean` **`isEliminado`**`()`

Returns:

true si el fichero se ha marcado como eliminado. false en caso contrario

setEliminado

`public void` **`setEliminado`**`(boolean eliminado)`

Parameters:

`eliminado` - bit activo si el fichero se marca como eliminado

13.3.1.86 Clase Incidencia

org.sigiccs.serv.incidencia.model

```
java.lang.Object
  └org.sigiccs.serv.incidencia.model.Incidencia
```

```
public class Incidencia
extends Object
```

Clase del modelo que representa las entidades Incidencia

Constructor Detail

Incidencia

```
public Incidencia()
```

Incidencia

```
public Incidencia(int iDIncidencia,
                  int iDInformador,
                  int iDAsignada,
                  int iDCliente,
                  int iDContrato,
                  int iDTipoIncidencia,
                  int iDPrioridad,
                  String titulo,
                  String descripcion,
                  boolean realizableRemoto,
                  double tiempoEstimado,
                  double tiempoDedicado,
                  String fechaAlta,
                  boolean visibilidad,
                  String estado,
                  boolean eliminado)
```

Method Detail

getIDIncidencia

```
public int getIDIncidencia()
```

Returns:

El identificador de la incidencia

setIDIncidencia

```
public void setIDIncidencia(int iDIncidencia)
```

Parameters:

`iDIncidencia` - identificador de la incidencia

getIDInformador

```
public int getIDInformador()
```

Returns:

El identificador del usuario que la dio de alta en el sistema

setIDInformador

`public void` **`setIDInformador`**`(int iDInformador)`

Parameters:

`iDInformador` - Identificador del usuario que la dio de alta en el sistema

getIDAsignada

`public int` **`getIDAsignada`**`()`

Returns:

El identificador del usuario al que se asigna su tratamiento

setIDAsignada

`public void` **`setIDAsignada`**`(int iDAsignada)`

Parameters:

`iDAsignada` - identificador del usuario al que se asigna su tratamiento

getIDContrato

`public int` **`getIDContrato`**`()`

Returns:

El identificador del contrato al que se vincula la incidencia

setIDContrato

`public void` **`setIDContrato`**`(int iDContrato)`

Parameters:

`iDContrato` - identificador del contrato al que se vincula la incidencia

getIDTipoIncidencia

`public int` **`getIDTipoIncidencia`**`()`

Returns:

El identificador del tipo de incidencia

setIDTipoIncidencia

`public void` **`setIDTipoIncidencia`**`(int iDTipoIncidencia)`

Parameters:

`iDTipoIncidencia` - identificador del tipo de incidencia

getIDPrioridad

`public int` **`getIDPrioridad`**`()`

Returns:

El identificador del tipo de prioridad asignada

setIDPrioridad

`public void` **`setIDPrioridad`**`(int iDPrioridad)`

Parameters:

`iDPrioridad` - identificador del tipo de prioridad asignada

getTitulo

```
public String getTitulo()
```

Returns:

El titulo descriptivo de la incidencia

setTitulo

```
public void setTitulo(String titulo)
```

Parameters:

`titulo` - titulo descriptivo de la incidencia

getDescripcion

```
public String getDescripcion()
```

Returns:

La descripción detallada de la incidencia

setDescripcion

```
public void setDescripcion(String descripcion)
```

Parameters:

`descripcion` - descripción detallada de la incidencia

isRealizableRemoto

```
public boolean isRealizableRemoto()
```

Returns:

true si la incidencia se puede resolver de forma remota. false en caso contrario (presencial)

setRealizableRemoto

```
public void setRealizableRemoto(boolean realizableRemoto)
```

Parameters:

`realizableRemoto` - modo de resolución de la incidencia (remoto o presencial)

getTiempoEstimado

```
public double getTiempoEstimado()
```

Returns:

El tiempo, en horas en base 10, estimado para la resolución de la incidencia, con decimales ajustados

setTiempoEstimado

```
public void setTiempoEstimado(double tiempoEstimado)
```

Parameters:

`tiempoEstimado` - El tiempo, en horas en base 10, estimado para la resolución de la incidencia

---

getFechaAlta

```
public String getFechaAlta()
```

Returns:

La fecha en la que la incidencia fue dada de alta en el sistema

---

setFechaAlta

```
public void setFechaAlta(String fechaAlta)
```

Parameters:

`fechaAlta` - fecha de alta de la incidencia

---

isVisibilidad

```
public boolean isVisibilidad()
```

Returns:

true si la incidencia es pública. false si es privada

---

setVisibilidad

```
public void setVisibilidad(boolean visibilidad)
```

Parameters:

`visibilidad` - bit que determina la visibilidad de la incidencia. true si es pública. false si es privada

---

getEstado

```
public String getEstado()
```

Returns:

Estado actual descriptivo de la incidencia (predefinidos)

---

setEstado

```
public void setEstado(String estado)
```

Parameters:

`estado` - estado actual descriptivo de la incidencia (predefinidos)

---

isEliminado

```
public boolean isEliminado()
```

Returns:

true si la incidencia se marco como eliminada. false en caso contrario

---

setEliminado

```
public void setEliminado(boolean eliminado)
```

Parameters:

`eliminado` - bit que marca la incidencia como eliminada

---

getUsuarioInformador

`public String` **`getUsuarioInformador`**`()`

Returns:

El nombre del usuario que dio de alta la incidencia en el sistema

setUsuarioInformador

`public void` **`setUsuarioInformador`**`(String usuarioInformador)`

Parameters:

`usuarioInformador` - nombre del usuario que da de alta la incidencia en el sistema

getUsuarioAsignada

`public String` **`getUsuarioAsignada`**`()`

Returns:

El nombre del usuario al que se asigna la resolucion de la incidencia

setUsuarioAsignada

`public void` **`setUsuarioAsignada`**`(String usuarioAsignada)`

Parameters:

`usuarioAsignada` - nombre del usuario al que se asigna la resolucion de la incidencia

getTituloContrato

`public String` **`getTituloContrato`**`()`

Returns:

Titulo descriptivo del contrato al que pertenece la incidencia

setTituloContrato

`public void` **`setTituloContrato`**`(String tituloContrato)`

Parameters:

`tituloContrato` - titulo descriptivo del contrato al que pertenece la incidencia

getTituloCliente

`public String` **`getTituloCliente`**`()`

Returns:

El nombre del cliente que suscribe el contrato al que pertenece la incidencia

setTituloCliente

`public void` **`setTituloCliente`**`(String tituloCliente)`

Parameters:

`tituloCliente` - nombre del cliente que suscribe el contrato al que pertenece la incidencia

getTipoIncidencia

`public String` **`getTipoIncidencia`**`()`

Returns:

La descripcion del tipo de incidencia (predefinido)

setTipoIncidencia

`public void` **`setTipoIncidencia`**`(String tipoIncidencia)`

Parameters:

`tipoIncidencia` - descripción del tipo de incidencia (predefinido)

getTipoPrioridad

`public String` **`getTipoPrioridad`**`()`

Returns:

La descripción del tipo de prioridad (predefinido)

setTipoPrioridad

`public void` **`setTipoPrioridad`**`(String tipoPrioridad)`

Parameters:

`tipoPrioridad` - descripción del tipo de prioridad (predefinido)

getIDCliente

`public int` **`getIDCliente`**`()`

Returns:

El identificador del cliente que suscribe el contrato al que pertenece la incidencia

setIDCliente

`public void` **`setIDCliente`**`(int iDCliente)`

Parameters:

`iDCliente` - Identificador del cliente que suscribe el contrato al que pertenece la incidencia

getTiempoDedicado

`public double` **`getTiempoDedicado`**`()`

Returns:

El tiempo, en horas en base 10, dedicado a la resolucion de la incidencia, con decimales ajustados

setTiempoDedicado

`public void` **`setTiempoDedicado`**`(double tiempoDedicado)`

Parameters:

`tiempoDedicado` - tiempo, en horas en base 10, dedicado a la resolucion de la incidencia, con decimales ajustados

13.3.1.87 Clase Intervencion

org.sigiccs.serv.incidencia.model

```
java.lang.Object
  └org.sigiccs.serv.incidencia.model.Intervencion
```

```
public class Intervencion
extends Object
```

Clase del modelo que representa las intervenciones realizadas a cada incidencia

Constructor Detail

Intervencion

```
public Intervencion()
```

Intervencion

```
public Intervencion(int iDIntervencion,
                    int iDUsuario,
                    int iDIncidencia,
                    String fecha,
                    String parte,
                    double numHoras,
                    String motivo,
                    String observaciones,
                    boolean eliminado)
```

Method Detail

getIDIntervencion

```
public int getIDIntervencion()
```

Returns:

El identificador de la intervencion

setIDIntervencion

```
public void setIDIntervencion(int iDIntervencion)
```

Parameters:

`iDIntervencion` - identificador de la intervencion

getIDUsuario

```
public int getIDUsuario()
```

Returns:

El identificador del usuario que realizo la intervencion

setIDUsuario

```
public void setIDUsuario(int iDUsuario)
```

Parameters:

`iDUsuario` - identificador del usuario que realizo la intervencion

getIDIncidencia

`public int` **`getIDIncidencia`**`()`

Returns:

El identificador de la incidencia a la que se refiere la intervencion

---

setIDIncidencia

`public void` **`setIDIncidencia`**`(int iDIncidencia)`

Parameters:

`iDIncidencia` - identificador de la incidencia a la que se refiere la intervencion

---

getFecha

`public String` **`getFecha`**`()`

Returns:

La fecha y hora en la que fue llevada a cabo la intervencion

---

setFecha

`public void` **`setFecha`**`(String fecha)`

Parameters:

`fecha` - fecha y hora en la que fue llevada a cabo la intervencion

---

getParte

`public String` **`getParte`**`()`

Returns:

La referencia al parte de trabajo generado en la intervencion (opcional)

---

setParte

`public void` **`setParte`**`(String parte)`

Parameters:

`parte` - referencia al parte de trabajo generado en la intervencion (opcional)

---

getNumHoras

`public double` **`getNumHoras`**`()`

Returns:

El tiempo, en horas en base 10, empleado en la intervencion, con decimales ajustados

---

setNumHoras

`public void` **`setNumHoras`**`(double numHoras)`

Parameters:

`numHoras` - tiempo, en horas en base 10, empleado en la intervencion, con decimales ajustados

---

getMotivo

`public String` **`getMotivo`**`()`

Returns:

El motivo de la intervencion

---

setMotivo

```
public void setMotivo(String motivo)
```

Parameters:

`motivo` - motivo de la intervencion

---

getObservaciones

```
public String getObservaciones()
```

Returns:

Las observaciones adicionales a la intervencion (opcional)

---

setObservaciones

```
public void setObservaciones(String observaciones)
```

Parameters:

`observaciones` - observaciones adicionales a la intervencion (opcional)

---

isEliminado

```
public boolean isEliminado()
```

Returns:

true si la intervencion fue eliminada. false en caso contrario

---

setEliminado

```
public void setEliminado(boolean eliminado)
```

Parameters:

`eliminado` - marca como eliminada la intervencion

---

getUsuarioInterviene

```
public String getUsuarioInterviene()
```

Returns:

El nombre del usuario que realizo la intervencion

---

setUsuarioInterviene

```
public void setUsuarioInterviene(String usuarioInterviene)
```

Parameters:

`usuarioInterviene` - nombre del usuario que realizo la intervencion

13.3.1.88 Clase Prioridad

org.sigiccs.serv.incidencia.model
java.lang.Object
└ **org.sigiccs.serv.incidencia.model.Prioridad**

public class **Prioridad**
extends Object

Clase del modelo que representa las diferentes prioridades con las que se puede tratar una incidencia

Constructor Detail

Prioridad

public **Prioridad**()

Prioridad

public **Prioridad**(int iDPrioridad,
String descripcion,
boolean eliminado)

Method Detail

getIDPrioridad

public int **getIDPrioridad**()

Returns:

El identificador del tipo de prioridad

setIDPrioridad

public void **setIDPrioridad**(int iDPrioridad)

Parameters:

iDPrioridad - identificador del tipo de prioridad

getDescripcion

public String **getDescripcion**()

Returns:

La descripción del tipo de prioridad

setDescripcion

public void **setDescripcion**(String descripcion)

Parameters:

descripcion - descripción del tipo de prioridad

isEliminado

public boolean **isEliminado**()

Returns:

true si el tipo de prioridad ya no está activo. false en caso contrario

setEliminado

```
public void setEliminado(boolean eliminado)
```

Parameters:

`eliminado` - marca como eliminado el tipo de prioridad

13.3.1.89 Clase RelacionIncidencia

org.sigiccs.serv.incidencia.model
java.lang.Object
└ **org.sigiccs.serv.incidencia.model.RelacionIncidencia**

public class **RelacionIncidencia**
extends Object

Clase del modelo que representa las potenciales relaciones entre incidencias

Constructor Detail

RelacionIncidencia

public **RelacionIncidencia**()

RelacionIncidencia

```
public RelacionIncidencia(int iDRelacion,
                          int iDIncidenciaMain,
                          int iDIncidenciaSub,
                          int iDTipoRelacion,
                          String fecha,
                          int iDUsuario,
                          boolean eliminado,
                          String tituloIDMain,
                          String tituloIDSub,
                          String descripcionTipoRelacion,
                          String nombreUsuarioRelaciono)
```

Method Detail

getIDRelacion

public int **getIDRelacion**()

Returns:

El identificador de la relación

setIDRelacion

public void **setIDRelacion**(int iDRelacion)

Parameters:

`iDRelacion` - identificador de la relación

getIDIncidenciaMain

public int **getIDIncidenciaMain**()

Returns:

El identificador de la incidencia origen de la relación

setIDIncidenciaMain

public void **setIDIncidenciaMain**(int iDIncidenciaMain)

Parameters:

`iDIncidenciaMain` - identificador de la incidencia origen de la relación

getIDIncidenciaSub

```
public int getIDIncidenciaSub()
```

Returns:

El identificador de la incidencia destino de la relación

setIDIncidenciaSub

```
public void setIDIncidenciaSub(int iDIncidenciaSub)
```

Parameters:

`iDIncidenciaSub` - identificador de la incidencia destino de la relación

getIDTipoRelacion

```
public int getIDTipoRelacion()
```

Returns:

El identificador del tipo de la relación

setIDTipoRelacion

```
public void setIDTipoRelacion(int iDTipoRelacion)
```

Parameters:

`iDTipoRelacion` - identificador del tipo de la relación

getFecha

```
public String getFecha()
```

Returns:

La fecha y hora de establecimiento de la relación

setFecha

```
public void setFecha(String fecha)
```

Parameters:

`fecha` - fecha y hora de establecimiento de la relación

getIDUsuario

```
public int getIDUsuario()
```

Returns:

El identificador del usuario que estableció la relación

setIDUsuario

```
public void setIDUsuario(int iDUsuario)
```

Parameters:

`iDUsuario` - identificador del usuario que estableció la relación

isEliminado

`public boolean` **`isEliminado`**`()`

Returns:

true si la relación se marcó como eliminada. false en caso contrario

---

setEliminado

`public void` **`setEliminado`**`(boolean eliminado)`

Parameters:

`eliminado` - marca la relación como eliminada

---

getTituloIDMain

`public String` **`getTituloIDMain`**`()`

Returns:

El nombre de la incidencia origen de la relación

---

setTituloIDMain

`public void` **`setTituloIDMain`**`(String tituloIDMain)`

Parameters:

`tituloIDMain` - nombre de la incidencia origen de la relación

---

getTituloIDSub

`public String` **`getTituloIDSub`**`()`

Returns:

El nombre de la incidencia destino de la relación

---

setTituloIDSub

`public void` **`setTituloIDSub`**`(String tituloIDSub)`

Parameters:

`tituloIDSub` - nombre de la incidencia destino de la relación

---

getDescripcionTipoRelacion

`public String` **`getDescripcionTipoRelacion`**`()`

Returns:

La descripción del tipo de relación (predefinido)

---

setDescripcionTipoRelacion

`public void` **`setDescripcionTipoRelacion`**`(String descripcionTipoRelacion)`

Parameters:

`descripcionTipoRelacion` - descripción del tipo de relación (predefinido)

---

getNombreUsuarioRelaciono

`public String` **`getNombreUsuarioRelaciono`**`()`

Returns:

El nombre del usuario que estableció la relación

setNombreUsuarioRelaciono

```
public void setNombreUsuarioRelaciono(String nombreUsuarioRelaciono)
```

Parameters:

`nombreUsuarioRelaciono` - nombre del usuario que estableció la relación

13.3.1.90 Clase TipoIncidencia

org.sigiccs.serv.incidencia.model
java.lang.Object
└ **org.sigiccs.serv.incidencia.model.TipoIncidencia**

public class **TipoIncidencia**
extends Object

Clase del modelo que representa los diferentes tipos de incidencia existentes en el sistema

Constructor Detail

TipoIncidencia

```
public TipoIncidencia()
```

TipoIncidencia

```
public TipoIncidencia(int iDTipoIncidencia,
                      String descripcion,
                      boolean eliminado)
```

Method Detail

getIDTipoIncidencia

```
public int getIDTipoIncidencia()
```

Returns:

El identificador del tipo de incidencia

setIDTipoIncidencia

```
public void setIDTipoIncidencia(int iDTipoIncidencia)
```

Parameters:

`iDTipoIncidencia` - identificador del tipo de incidencia

getDescripcion

```
public String getDescripcion()
```

Returns:

La descripción del tipo de incidencia (predefinido)

setDescripcion

```
public void setDescripcion(String descripcion)
```

Parameters:

`descripcion` - descripción del tipo de incidencia (predefinido)

isEliminado

```
public boolean isEliminado()
```

Returns:

true si el tipo de incidencia ya no existe en el sistema. false en caso contrario

setEliminado

```
public void setEliminado(boolean eliminado)
```

Parameters:

`eliminado` - marca un tipo de incidencia como eliminado

13.3.1.91 Clase TipoRelacion

org.sigiccs.serv.incidencia.model

```
java.lang.Object
  └ org.sigiccs.serv.incidencia.model.TipoRelacion
```

```
public class TipoRelacion
extends Object
```

Clase del modelo que representa los diferentes tipos de relación entre incidencias existentes en el sistema

Constructor Detail

TipoRelacion

```
public TipoRelacion()
```

TipoRelacion

```
public TipoRelacion(int iDTipoRelacion,
                    String descripcion,
                    boolean eliminado)
```

Method Detail

getIDTipoRelacion

```
public int getIDTipoRelacion()
```

Returns:

El identificador del tipo de relación

setIDTipoRelacion

```
public void setIDTipoRelacion(int iDTipoRelacion)
```

Parameters:

`iDTipoRelacion` - identificador del tipo de relación

getDescripcion

```
public String getDescripcion()
```

Returns:

La descripción del tipo de relación (predefinido)

setDescripcion

```
public void setDescripcion(String descripcion)
```

Parameters:

`descripcion` - descripción del tipo de relación (predefinido)

isEliminado

```
public boolean isEliminado()
```

Returns:

true si el tipo de relación ya no existe en el sistema. false en caso contrario

setEliminado

```
public void setEliminado(boolean eliminado)
```

Parameters:

`eliminado` - marca como eliminado el tipo de relación

13.3.1.92 Interfaz DocumentoIncidenciaDataService

org.sigiccs.serv.incidencia.persistence

All Known Implementing Classes:

DocumentoIncidenciaDAO

`public interface` **`DocumentoIncidenciaDataService`**

Interfaz que establece la signatura de los métodos que implementará DocumentoIncidenciaDAO

Method Detail

getDocumentoIncidenciaByID

```
DocumentoIncidencia getDocumentoIncidenciaByID(int idDocumento)
                                               throws SQLException
```

Busca un adjunto a una incidencia en base al identificador del fichero

Parameters:

`idDocumento` - Identificador del fichero adjunto

Returns:

Un objeto DocumentoIncidencia con todos los datos del mismo

Throws:

`SQLException`

---

getAllDocumentosIncidencia

```
Vector<DocumentoIncidencia> getAllDocumentosIncidencia(int idIncidencia)
                                                       throws SQLException
```

Busca todos los adjuntos a una incidencia en particular

Parameters:

`idIncidencia` - Identificador de la incidencia para la que se buscaran los documentos adjuntos

Returns:

Un objeto Vector con todos los objetos DocumentoIncidencia asociados a la incidencia

Throws:

`SQLException`

---

almacenarNuevoDocumentoIncidencia

```
int almacenarNuevoDocumentoIncidencia(DocumentoIncidencia documentoIncidencia)
                                      throws SQLException
```

Almacena un fichero adjunto a una incidencia, incluidos sus datos descriptivos

Parameters:

`documentoIncidencia` - El objeto que representa al adjunto

Returns:

1 si el fichero fue almacenado correctamente

Throws:

`SQLException`

---

eliminarDocumentoIncidencia

```
int eliminarDocumentoIncidencia(int idDocumento)
                          throws SQLException
```

Marca como eliminado un fichero adjunto a una incidencia

Parameters:

`idDocumento` - El identificador del adjunto a eliminar

Returns:

1 si el adjunto fue eliminado correctamente

Throws:

`SQLException`

eliminarDocumentosIncidencia

```
int eliminarDocumentosIncidencia(int iDIncidencia)
                           throws SQLException
```

Marca como eliminados todos los adjuntos a una incidencia

Parameters:

`iDIncidencia` - Identificador de la incidencia a tratar

Returns:

1 si los adjuntos fueron eliminados correctamente

Throws:

`SQLException`

13.3.1.93 Interfaz IncidenciaDataService

org.sigiccs.serv.incidencia.persistence

All Known Implementing Classes:

IncidenciaDAO

`public interface` **`IncidenciaDataService`**

Interfaz que establece la signatura de los métodos que implementará IncidenciaDAO

Method Detail

getIncidenciaByID

`Incidencia` **`getIncidenciaByID`**`(int iDIncidencia)`
`throws SQLException`

Busca toda la información asociada a una incidencia

Returns:

Un objeto Incidencia con todos los datos de la misma

Throws:

`SQLException`

---

getAllIncidencias

`Vector<Incidencia>` **`getAllIncidencias`**`()`
`throws SQLException`

Busca todas las incidencias dadas de alta en el sistema

Returns:

Un objeto Vector con todos los objetos Incidencia encontrados

Throws:

`SQLException`

---

almacenarNuevaIncidencia

`int` **`almacenarNuevaIncidencia`**`(Incidencia incidencia)`
`throws SQLException`

Almacena una nueva incidencia en el sistema

Parameters:

`incidencia` - El objeto incidencia con todos los datos proporcionados en su creación

Returns:

1 si la incidencia fue almacenada correctamente

Throws:

`SQLException`

---

eliminarIncidencia

`int` **`eliminarIncidencia`**`(int iDIncidencia)`
`throws SQLException`

Marca como eliminada una incidencia existente

Returns:

1 si la incidencia fue eliminada correctamente

Throws:

`SQLException`

actualizarIncidencia

```
int actualizarIncidencia(Incidencia incidencia)
                  throws SQLException
```

Actualiza los datos de una incidencia existente

Parameters:

`incidencia` - El objeto incidencia con los nuevos datos

Returns:

1 si la incidencia fue modificada correctamente

Throws:

`SQLException`

---

getAllIncidenciasCliente

```
Vector<Incidencia> getAllIncidenciasCliente(int iDCliente)
                                    throws SQLException
```

Busca todas las incidencias asociadas a un cliente en particular

Parameters:

`iDCliente` - Identificador del cliente en base al cual se realiza la búsqueda

Returns:

Un objeto Vector con todos los objetos Incidencia encontrados

Throws:

`SQLException`

---

getTiposIncidencia

```
Vector<TipoIncidencia> getTiposIncidencia()
                                  throws SQLException
```

Consulta el listado de Tipos de Incidencia existentes

Returns:

Un objeto Vector con todos los objetos TipoIncidencia encontrados

Throws:

`SQLException`

---

getPrioridades

```
Vector<Prioridad> getPrioridades()
                         throws SQLException
```

Consulta el listado de Tipos de Prioridad aplicables a las incidencias

Returns:

Un objeto Vector con todos los objetos Prioridad encontrados

Throws:

`SQLException`

13.3.1.94 Interfaz IntervencionDataService

org.sigiccs.serv.incidencia.persistence

All Known Implementing Classes:

IntervencionDAO

`public interface` **`IntervencionDataService`**

Interfaz que establece la signatura de los métodos que implementará IntervencionDAO

Detail

getIntervencionByID

`Intervencion` **`getIntervencionByID`**`(int idIntervencion)`
`throws SQLException`

Busca una intervencion en una incidencia en base al identificador de la misma

Parameters:

`idIntervencion` - Identificador de la intervencion a buscar

Returns:

Un objeto Intervencion con los datos de la misma

Throws:

`SQLException`

---

getAllIntervencionesIncidencia

`Vector<Intervencion>` **`getAllIntervencionesIncidencia`**`(int idIncidencia)`
`throws SQLException`

Busca todas las intervenciones efectuadas en una incidencia en particular

Parameters:

`idIncidencia` - Identificador de la incidencia en la que se buscara

Returns:

Un objeto Vector con todos los objetos Intervencion encontrados

Throws:

`SQLException`

---

almacenarNuevaIntervencion

`int` **`almacenarNuevaIntervencion`**`(Intervencion intervencion)`
`throws SQLException`

Almacena una intervencion en una incidencia

Parameters:

`intervencion` - El objeto que representa a la intervencion

Returns:

1 si la intervencion fue almacenada correctamente

Throws:

`SQLException`

---

eliminarIntervencion

`int` **`eliminarIntervencion`**`(int IDIntervencion)`
`throws SQLException`

Marca como eliminada una intervencion existente en una incidencia

Returns:

1 si la intervencion fue eliminada correctamente

Throws:

`SQLException`

actualizarIntervencion

```
int actualizarIntervencion(Intervencion intervencion)
                    throws SQLException
```

Actualiza las propiedades de una intervencion existente

Parameters:

`intervencion` - El objeto intervencion con los nuevos datos

Returns:

1 si la intervencion fue modificada correctamente

Throws:

`SQLException`

eliminarIntervencionesIncidencia

```
int eliminarIntervencionesIncidencia(int iDIncidencia)
                              throws SQLException
```

Marca como eliminadas todas las intervenciones de una incidencia

Parameters:

`iDIncidencia` - Identificador de la incidencia en la que se eliminaran sus intervenciones

Returns:

1 si las intervenciones fueron eliminadas correctamente

Throws:

`SQLException`

13.3.1.95 Interfaz RelacionIncidenciaDataService

org.sigiccs.serv.incidencia.persistence

All Known Implementing Classes:

RelacionIncidenciaDAO

`public interface` **`RelacionIncidenciaDataService`**

Interfaz que establece la signatura de los métodos que implementará RelacionIncidenciaDAO

Method Detail

getRelacionIncidenciaByID

`RelacionIncidencia` **`getRelacionIncidenciaByID`**`(int iDRelacion) throws SQLException`

Consulta todos los datos de una relación en particular

Parameters:

`iDRelacion` - Identificador de la relaicon a consultar

Returns:

Un objeto RelacionIncidencia con todos los datos de la relación

Throws:

`SQLException`

---

getRelacionesIncidenciaMain

`Vector<RelacionIncidencia>` **`getRelacionesIncidenciaMain`**`(int iDIncidenciaMain) throws SQLException`

Consulta todas las relaciones en las que una incidencia es origen de las mismas

Parameters:

`iDIncidenciaMain` - Identificador de la incidencia origen

Returns:

Un objeto Vector con todos los objetos RelacionIncidencia encontrados

Throws:

`SQLException`

---

getRelacionesIncidenciaSub

`Vector<RelacionIncidencia>` **`getRelacionesIncidenciaSub`**`(int iDIncidenciaSub) throws SQLException`

Consulta todas las relaciones en las que una incidencia es destino de las mismas

Returns:

Un objeto Vector con todos los objetos RelacionIncidencia encontrados

Throws:

`SQLException`

---

almacenarNuevaRelacionIncidencia

`int` **`almacenarNuevaRelacionIncidencia`**`(RelacionIncidencia relacionIncidencia) throws SQLException`

Establece una nueva relación para la incidencia

Parameters:

`relacionIncidencia` - El objeto que representa a la relación

Returns:

1 si la relación fue establecida correctamente

Throws:

`SQLException`

---

eliminarRelacionIncidencia

```
int eliminarRelacionIncidencia(int iDRelacion)
                        throws SQLException
```

Marca como eliminada una relación en particular

Parameters:

`iDRelacion` - Identificador de la relación a eliminar

Returns:

1 si la relación fue eliminada correctamente

Throws:

`SQLException`

---

getTiposRelaciones

```
Vector<TipoRelacion> getTiposRelaciones()
                                  throws SQLException
```

Consulta el listado de Tipos de Relación entre incidencias existentes

Returns:

Un objeto Vector con todos los objetos TipoRelacion encontrados

Throws:

`SQLException`

---

getTipoRelacionByID

```
TipoRelacion getTipoRelacionByID(int iDTipoRelacion)
                          throws SQLException
```

Devuelve un Tipo de Relación en base a su identificador

Parameters:

`iDTipoRelacion` - Identificador del tipo de relación a consultar

Returns:

Un objeto TipoRelacion con todos los datos del mismo

Throws:

`SQLException`

---

esRelacionable

```
boolean esRelacionable(RelacionIncidencia relación)
                throws SQLException
```

Comprueba si una incidencia es relacionable con otra

Parameters:

`relación` - Atributos de la relación que se pretende establecer

Returns:

true si las incidencias son relacionables. false en caso contrario

Throws:

`SQLException`

---

eliminarRelacionesIncidencia

```
int eliminarRelacionesIncidencia(int iDIncidencia)
                          throws SQLException
```

Marca como eliminadas todas las relaciones directas de una incidencia

Parameters:

`iDIncidencia` - Identificador de la incidencia para la que se eliminaran sus relaciones

Returns:

1 si las relaciones fueron eliminadas correctamente

Throws:

`SQLException`

13.3.1.96 Interfaz ServicioManagerService

org.sigiccs.serv.servicio.business

All Known Implementing Classes:

ServicioManager

`public interface` **`ServicioManagerService`**

Interfaz que establece la signatura de los métodos que implementará ServicioManager

Method Detail

getAllServicios

`Vector<Servicio>` **`getAllServicios`**`()`
`throws SQLException`

Busca todos los servicios ofertados por la empresa

Returns:

Un Vector con todos los objetos Servicio encontrados

Throws:

`SQLException`

---

getServicioById

`Servicio` **`getServicioById`**`(int idServicio)`
`throws SQLException`

Busca toda la información de un servicio del catálogo en particular

Parameters:

`idServicio` - El identificador del servicio que se desea consultar

Returns:

Un objeto Servicio con todos los datos del mismo

Throws:

`SQLException`

---

almacenarNuevoServicio

`int` **`almacenarNuevoServicio`**`(Servicio servicio)`
`throws SQLException`

Almacena un nuevo servcio en el catálogo del sistema

Parameters:

`servicio` - El objeto Servicio con todos sus datos

Returns:

1 si el servicio fue almacenado correctamente

Throws:

`SQLException`

---

actualizarServicio

`int` **`actualizarServicio`**`(Servicio servicio)`
`throws SQLException`

Actualiza los datos de un servicio ya existente en el catálogo

Parameters:

`servicio` - El objeto Servicio con los nuevos datos

Returns:

1 si el servicio fue modificado correctamente

Throws:

`SQLException`

---

removeServicio

```
int removeServicio(int idServicio)
            throws SQLException
```

Marca como eliminado un servicio de los existentes en el catálogo

Parameters:

`idServicio` - El identificador del servicio a eliminar

Returns:

1 si el servicio fue eliminado correctamente

Throws:

`SQLException`

---

getServiciosContratables

```
Vector<Servicio> getServiciosContratables(int idContrato)
                                    throws SQLException
```

Consulta el listado de servicios que pueden ser asociados a un contrato en particular

Parameters:

`idContrato` - El identificador del contrato en base al cual se realiza la consulta

Returns:

Un objeto Vector con todos los objetos Servicio que cumplen los requisitos para ser asociables al contrato

Throws:

`SQLException`

13.3.1.97 Interfaz TarifaManagerService

org.sigiccs.serv.servicio.business

All Known Implementing Classes:

TarifaManager

`public interface` **`TarifaManagerService`**

Interfaz que establece la signatura de los métodos que implementará TarifaManager

Method Detail

getAllTarifaServicios

```
Vector<TarifaServicio> getAllTarifaServicios()
                                        throws SQLException
```

Busca todas las tarifas aplicables a los servicios del catálogo del sistema

Returns:

Un objeto Vector con todos los objetos TarifaServicio

Throws:

`SQLException`

---

getTarifaServicioById

```
TarifaServicio getTarifaServicioById(int IDTarifaServicio)
                                throws SQLException
```

Busca toda la información de una tarifa en particular

Parameters:

`IDTarifaServicio` - El identificador de la tarifa a consultar

Returns:

Un objeto TarifaServicio con todos los datos de la misma

Throws:

`SQLException`

---

getTarifasDelServicio

```
Vector<TarifaServicio> getTarifasDelServicio(int IDServicio,
                                             int IDContrato)
                                        throws SQLException
```

Busca todas las tarifas asociadas a un servicio vinculado a un contrato suscrito

Parameters:

`IDServicio` - El identificador del servicio

`IDContrato` - El identificador del contrato que contiene el servicio

Returns:

Un objeto Vector con todos los objetos TarifaServicio encontrados

Throws:

`SQLException`

---

almacenarNuevaTarifaServicio

```
int almacenarNuevaTarifaServicio(TarifaServicio tarifaServicio)
                          throws SQLException
```

Almacena una nueva tarifa en el catálogo de servicios

Parameters:

`tarifaServicio` - El objeto TarifaServicio con todos los datos a almacenar

Returns:

1 si la tarifa fue almacenada correctamente

Throws:

`SQLException`

---

actualizarTarifaServicio

```
int actualizarTarifaServicio(TarifaServicio tarifaServicio)
                      throws SQLException
```

Actualiza las propiedades de una tarifa existente

Parameters:

`tarifaServicio` - El objeto TarifaServicio a modificar

Returns:

1 si la tarifa fue actualizada correctamente

Throws:

`SQLException`

---

removeTarifaServicio

```
int removeTarifaServicio(int idTarifaServicio)
                  throws SQLException
```

Marca como eliminada una tarifa existente en el catálogo de servicios

Parameters:

`idTarifaServicio` - El identificador de la tarifa

Returns:

1 si la tarifa se elimino correctamente

Throws:

`SQLException`

13.3.1.98 Clase Servicio

org.sigiccs.serv.servicio.model
java.lang.Object
└ **org.sigiccs.serv.servicio.model.Servicio**

public class **Servicio**
extends Object

Clase del modelo que representa a los objetos Servicio

Constructor Detail

Servicio

public **Servicio**()

Servicio

public **Servicio**(int iDServicio,
String descripcion)

Method Detail

getIDServicio

public int **getIDServicio**()

Returns:

El identificador del servicio

setIDServicio

public void **setIDServicio**(int iDServicio)

Parameters:

iDServicio - identificador del servicio

getDescripcion

public String **getDescripcion**()

Returns:

La descripcion textual del servicio

setDescripcion

public void **setDescripcion**(String descripcion)

Parameters:

descripcion - descripcion textual del servicio

getTarifas

public Vector<TarifaServicio> **getTarifas**()

Returns:

Un objeto Vector con todos los objetos Tarifa encontrados. Se utiliza para los formularios dependientes que emplean JSON para recibir los datos

setTarifas

```
public void setTarifas(Vector<TarifaServicio> tarifas)
```

Parameters:

`tarifas` - Un objeto Vector con todos los objetos Tarifa encontrados. Se utiliza para los formularios dependientes que emplean JSON para pasar los datos

13.3.1.99 Clase TarifaServicio

org.sigiccs.serv.servicio.model

```
java.lang.Object
  └org.sigiccs.serv.servicio.model.TarifaServicio
```

```
public class TarifaServicio
extends Object
```

Clase del modelo que representa el vinculo entre un servicio y una tarifa asociada a este

Constructor Detail

TarifaServicio

```
public TarifaServicio()
```

TarifaServicio

```
public TarifaServicio(int iDTarifaServicio,
                      int iDServicio,
                      float precioHora,
                      Date fechaInicio,
                      Date fechaFin)
```

Method Detail

getIDTarifaServicio

```
public int getIDTarifaServicio()
```

Returns:

El identificador de la tarifa

setIDTarifaServicio

```
public void setIDTarifaServicio(int iDTarifaServicio)
```

Parameters:

`iDTarifaServicio` - identificador de la tarifa

getIDServicio

```
public int getIDServicio()
```

Returns:

El identificador del servicio al que es aplicable

setIDServicio

```
public void setIDServicio(int iDServicio)
```

Parameters:

`iDServicio` - identificador del servicio al que es aplicable

getPrecioHora

`public float` **`getPrecioHora`**`()`

Returns:

El precio por hora de la tarifa, expresado en Euros

setPrecioHora

`public void` **`setPrecioHora`**`(float precioHora)`

Parameters:

`precioHora` - precio por hora de la tarifa, expresado en Euros

getFechaInicio

`public Date` **`getFechaInicio`**`()`

Returns:

La fecha de inicio de validez de la tarifa

setFechaInicio

`public void` **`setFechaInicio`**`(Date fechaInicio)`

Parameters:

`fechaInicio` - fecha de inicio de validez de la tarifa

getFechaFin

`public Date` **`getFechaFin`**`()`

Returns:

La fecha de expiracion de la tarifa

setFechaFin

`public void` **`setFechaFin`**`(Date fechaFin)`

Parameters:

`fechaFin` - fecha de expiracion de la tarifa

getDescServicioAsociado

`public String` **`getDescServicioAsociado`**`()`

Returns:

La descripción textual del servicio asociado (particularizada)

setDescServicioAsociado

`public void` **`setDescServicioAsociado`**`(String descServicioAsociado)`

Parameters:

`descServicioAsociado` - descripción textual del servicio asociado (particularizada)

getNombre

`public String` **`getNombre`**`()`

Returns:

El nombre interno de la tarifa

setNombre

```
public void setNombre(String nombre)
```

Parameters:

`nombre` - nombre interno de la tarifa

13.3.1.100 Interfaz ServicioDataService

org.sigiccs.serv.servicio.persistence

All Known Implementing Classes:

ServicioDAO

```
public interface ServicioDataService
```

Method Detail

getAllServicios

```
Vector<Servicio> getAllServicios()
                          throws SQLException
```

Busca todos los servicios ofertados por la empresa

Returns:

Un Vector con todos los objetos Servicio encontrados

Throws:

SQLException

getServicioById

```
Servicio getServicioById(int idServicio)
                  throws SQLException
```

Busca toda la información de un servicio del catálogo en particular

Parameters:

idServicio - El identificador del servicio que se desea consultar

Returns:

Un objeto Servicio con todos los datos del mismo

Throws:

SQLException

almacenarNuevoServicio

```
int almacenarNuevoServicio(Servicio servicio)
                    throws SQLException
```

Almacena un nuevo servcio en el catálogo del sistema

Parameters:

servicio - El objeto Servicio con todos sus datos

Returns:

1 si el servicio fue almacenado correctamente

Throws:

SQLException

actualizarServicio

```
int actualizarServicio(Servicio servicio)
                throws SQLException
```

Actualiza los datos de un servicio ya existente en el catálogo

Parameters:

servicio - El objeto Servicio con los nuevos datos

Returns:

1 si el servicio fue modificado correctamente

Throws:

`SQLException`

removeServicio

```
int removeServicio(int idServicio)
            throws SQLException
```

Marca como eliminado un servicio de los existentes en el catálogo

Parameters:

`idServicio` - El identificador del servicio a eliminar

Returns:

1 si el servicio fue eliminado correctamente

Throws:

`SQLException`

getServiciosContratables

```
Vector<Servicio> getServiciosContratables(int IDContrato)
                                    throws SQLException
```

Consulta el listado de servicios que pueden ser asociados a un contrato en particular

Returns:

Un objeto Vector con todos los objetos Servicio que cumplen los requisitos para ser asociables al contrato

Throws:

`SQLException`

13.3.1.101 Interfaz TarifaDataService

org.sigiccs.serv.servicio.persistence

All Known Implementing Classes:

TarifaDAO

`public interface` **`TarifaDataService`**

Interfaz que establece la signatura de los métodos que implementará TarifaDAO

Method Detail

getAllTarifaServicios

```
Vector<TarifaServicio> getAllTarifaServicios()
                                        throws SQLException
```

Busca todas las tarifas aplicables a los servicios del catálogo del sistema

Returns:

Un objeto Vector con todos los objetos TarifaServicio

Throws:

`SQLException`

---

getTarifaServicioById

```
TarifaServicio getTarifaServicioById(int IDTarifaServicio)
                                throws SQLException
```

Busca toda la información de una tarifa en particular

Parameters:

`IDTarifaServicio` - El identificador de la tarifa a consultar

Returns:

Un objeto TarifaServicio con todos los datos de la misma

Throws:

`SQLException`

---

almacenarNuevaTarifaServicio

```
int almacenarNuevaTarifaServicio(TarifaServicio tarifaServicio)
                            throws SQLException
```

Almacena una nueva tarifa en el catálogo de servicios

Parameters:

`tarifaServicio` - El objeto TarifaServicio con todos los datos a almacenar

Returns:

1 si la tarifa fue almacenada correctamente

Throws:

`SQLException`

---

actualizarTarifaServicio

```
int actualizarTarifaServicio(TarifaServicio tarifaServicio)
                        throws SQLException
```

Actualiza las propiedades de una tarifa existente

Parameters:

`tarifaServicio` - El objeto TarifaServicio a modificar

Returns:

1 si la tarifa fue actualizada correctamente

Throws:

`SQLException`

---

removeTarifaServicio

```
int removeTarifaServicio(int idTarifaServicio)
                  throws SQLException
```

Marca como eliminada una tarifa existente en el catálogo de servicios

Parameters:

`idTarifaServicio` - El identificador de la tarifa

Returns:

1 si la tarifa se elimino correctamente

Throws:

`SQLException`

---

getTarifasDelServicio

```
Vector<TarifaServicio> getTarifasDelServicio(int iDServicio,
                                             int iDContrato)
                                      throws SQLException
```

Busca todas las tarifas asociadas a un servicio vinculado a un contrato suscrito

Returns:

Un objeto Vector con todos los objetos TarifaServicio encontrados

Throws:

`SQLException`

13.3.1.102 Clase ConnectionManager

org.sigiccs.util
java.lang.Object
└ **org.sigiccs.util.ConnectionManager**

public class **ConnectionManager**
extends Object

Clase de utilidad para gestionar el pool de conexiones con la base de datos. Consigue gestionar mejor los recursos, reduciendo los tiempos de espera entre operaciones y el numero de conexiones abiertas.

Method Detail

getConnection

public static Connection **getConnection**()
throws NullPointerException

Returns:

Una conexión con la base de datos a traves del pool.

Throws:

NullPointerException

13.3.1.103 Clase ValueComparator

org.sigiccs.util

```
java.lang.Object
  └org.sigiccs.util.ValueComparator
```

All Implemented Interfaces:

Comparator<Integer>

```
public class ValueComparator
extends Object
implements Comparator<Integer>
```

Clase de utilidad que sobrecarga la operacion de comparación de elementos en un Map

Constructor Detail

ValueComparator

```
public ValueComparator(Map<Integer,String> base)
```

Method Detail

compare

```
public int compare(Integer a,
                   Integer b)
```

Specified by:

compare in interface Comparator<T>