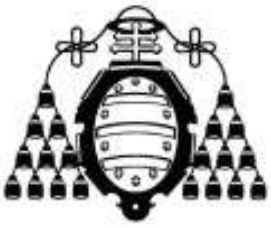
UNIVERSIDAD DE OVIEDO



ESCUELA DE INGENIERÍA INFORMÁTICA

TRABAJO FIN DE MÁSTER

"Sistema de Gestión del Ciclo de Vida de los Riesgos en Proyectos Informáticos"

DIRECTOR: Aquilino Adolfo Juan Fuente

AUTOR: Pablo González Jiménez

Vº Bº del Director del Proyecto

Agradecimientos

No quisiera dejar pasar la admiración, dedicación y agredecimiento por el apoyo que me han brindado las siguientes personas.

- A mi familia, tanto a mi padres José y Pilar que me han brindado la oportunidad de estudiar una carrera, como a mi hermanísima Cristina, que siempre han estado ahí para apoyarme.
- Ma jolie blonde, Madeleine, qui m'a Aidé plus me soutenir tout ce temps dans la distance, elle était aussi mon traducteur, jta.
- A mis compañeros y amigos de China donde realice mis prácticas de empresa, sobre todo a mi jefa María, por todo lo que hizo por mí.
- Como no a mis compañeros de trabajo, amigos de facultad y master por aguantarme las palizas que os daba a dudas y perder el tiempo conmigo en intentar ayudarme en lo máximo posible.
- A mis amigos del colegio e instituto, por alégrame cada día que pasaba con vosotros, que me hacíais desengancharme del proyecto y olvidar los malos ratos pasados
- Mi director de proyecto, Aquilino, por haberme ayudado en todo lo referente al proyecto y por volver a cogerme para realizar un nuevo proyecto
- A mis profesores de Máster por enseñarme a buscar la solución a base del "descubrimiento".
- A Google, como no, por brindarme soluciones a todos los problemas encontrados.
- Y a todos las demás personas que de una manera u otra han contribuido a que pueda presentar el trabajo fin de máster.
- Aunque si hay una persona a la que que agredecer mucho, esa persona soy yo, por lo que espero agradecérmelo en un futuro no muy lejano, tio que bueno eres.

A todos aquellos que me han puesto trabas en algún momento, sin ellos, nunca lo hubiera conseguido, ya que el esfuerzo y las ganas se multiplicaban a cada piedra que me ponían en el camino, seguramente una de las maneras donde más aprenderás en la vida.

出る杭は打たれる。

Resumen

El proyecto implementa una solución para la gestión de los riesgos de los proyectos, se realizara una aplicación web para la identificación, el análisis y el control de los riesgos que pueden producirse en un proyecto. Los objetivos son el aumentar la probabilidad y el impacto de eventos positivos y disminuir la probabilidad y el impacto de eventos negativos para el proyecto.

El sistema contara con diferentes etapas para la planificación de la gestión de riesgos, la identificación de los riesgos, también contara con algunos métodos para realizar tanto el análisis cualitativo como cuantitativo de los riesgos, produciendo una planificación a las posibles respuestas de los riesgos así como un seguimiento y control de los mismos.

Dicha herramienta pretende utilizarse para la sustitución de los documentos que existen en la actualidad, para las asignaturas tanto del Máster de Ingeniería Web como para el Grado de Ingeniería Informática del Software, sobre Gestión de Riesgos, facilitando la labor tanto de los alumnos como del profesor.

Algunas de las tecnologías usadas para la realización del proyecto son utilización del modelo de arquitectura MVC con Struts2 y Spring, donde la vista, iría la lógica de presentación, utilizando para ello JSPs que producen HTML; el controlador que sería el de Struts2 llamaría a sus action class por medio de la peticiones HTTP y por último el modelo con los componentes de acceso a datos utilizando para ello Spring para encapsular los datos JavaBeans y los patrones DAO para almacenar los datos en la BBDD, entre cada capa se comunicaría entre ellas por medio de la aplicación del patrón Façade, utilizando para ello entre las capas interfaces. Una vez registrado los datos en la base de datos estos son tratados en la aplicación y convertidos por medio de una librería llamada IText, en documentos PDF.

Palabras Clave

- Planes de Riesgos
- Identificación de Riesgos
- Análisis de Riesgos
- MVC con Struts2
- Spring
- J2EE
- Sistema de Gestión de Riesgos Informáticos

Abstract

The project implements a solution for the management of project risks, a web application for the identification, analysis and control of risks that can occur in a project. The objectives are to increase the probability and impact of positive events and decrease the probability and impact of negative events for the project.

The system will feature different planning stages for risk management, identification of risks, will also feature some methods to perform both qualitative and quantitative analysis of risks, producing a planning possible response of the risks and monitor and control them.

This tool aims to be used for the replacement of documents that exist today, for the subjects of the Master of Web Engineering as for the Degree of Computer Engineering Software; these subjects are about Risk Management, facilitating the work of both students and teacher.

Some of the technologies used for the project are using the MVC architecture model with Struts2 and Spring, where the view, it would presentation logic, using JSPs that produce HTML; the driver would be to call to Struts2 action class, through HTTP requests and finally the model with data access components using for this Spring, JavaBeans to encapsulate data and DAO patterns to store the data in the DB, each layer would be communicated between them through the application Façade pattern, using the interfaces between layers. Once registered the data in the database they are addressed in the application and converted by a library called IText in PDF documents.

Keywords

- Risks Plans
- Risks Identification
- Risks Analysis
- MVC with Struts2
- Spring
- J2EE
- Computer System Risk Management

Índice General

Índice de Figuras

Capítulo 1. Memoria del Proyecto

1.1 Resumen de la Motivación, Objetivos y Alcance del Proyecto

Después de cursar la asignatura del Máster de Ingeniería Web, Dirección y Gestión de Proyectos Web, el director de proyecto y yo decidimos realizar este proyecto para mejorar la calidad del trabajo sobre riegos, actualmente los informes sobre los planes de riesgos, la identificación de los mismos por medio de un matriz de probabilidad e impactos se realizan por medio de plantillas Word y Excel, se podría agilizar y convertir en una herramienta informática la ejecución de dichos documentos por medio de una aplicación informática que ayudará al profesor y alumno, a uno para una mejor y más rápida exanimación de los alumnos y a los otros una mejor manera y más rápida y eficiente de rellenar los diferentes requerimientos en la asignatura. Por ello me decidí a construir está aplicación que toma un poco de todas las demás asignaturas del Máster en un solo proyecto.

El objetivo es realizar una aplicación web, para desarrollar un gestor de riesgos para proyectos informáticos, debe desarrollar el punto 11 del PMBOK, que es una guía de los fundamentos para la dirección de proyectos, el punto 11 se dedica fundamentalmente a describir los procesos involucrados en la identificación, análisis y control de los riesgos para un proyecto, finalmente con la recogida de los datos en cada proceso se crearán unos documentos PDF, con las diferentes características de cada paso.

1. **Planificar la Gestión de Riesgos—**Es el proceso por el cual se define cómo realizar las actividades de gestión de los riesgos para un proyecto.
2. **Identificar los Riesgos—**Es el proceso por el cual se determinan los riesgos que pueden afectar el proyecto y se documentan sus características.
3. **Análisis (cuantitativo y cualitativo) de los Riesgos —**Es el proceso que consiste en priorizar los riesgos para realizar otros análisis o acciones posteriores, evaluando y combinando la probabilidad de ocurrencia y el impacto de dichos riesgos y analizar numéricamente el efecto de los riesgos identificados sobre los objetivos generales del proyecto.
4. **Informes —** Sería una técnica cuantitativa para el estudio de los riesgos encontrados cuya finalidad es que no se produzcan o que se pueden disminuir su impacto en el proyecto si estos llegan a producirse.

El alcance del proyecto podría llevarnos mucho más tiempo si se considerarán más opciones para el estudio de los riesgos ya que dependen del tipo de riesgo, así como el último punto sobre **Monitorización y Control de los Riesgos**, eso supera un poco la intención de este proyecto, así que el proyecto consistiría en la creación de una aplicación web por medio de una estructura MVC utilizando para ello los Framework de Struts2 para web y Spring para el desarrollo de aplicaciones que nos da soporte a todas las capas de la aplicación.

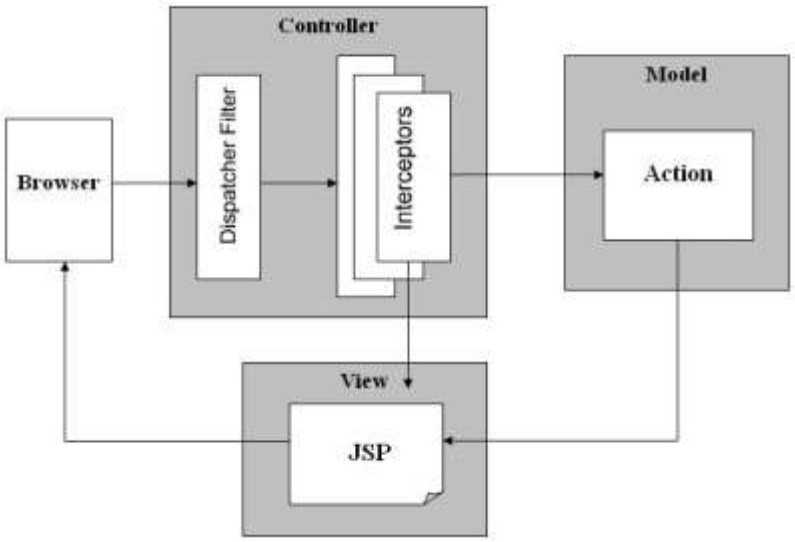


Figura 1.1. MVC con Struts2

Se utilizan diferentes patrones, de diseño como, la separación de responsabilidades por medio de capas:

- Capa de presentación, donde por medio de HTML5, CSS3, JavaScript y JQuery, ofrecemos al usuario las diferentes pantallas para interactuar con la aplicación web.
- Capa de negocio, donde se producen el tratamiento de los datos introducidos por el usuario tanto para tratarlos en la capa de presentación como en la capa de persistencia.
- Capa de persistencia, es lugar donde almacenaremos los datos introducidos por el usuario y devolvemos aquellos que necesitan ser utilizados en ciertos momentos por la aplicación.

MVC que no reemplaza la arquitectura en capas:

- La capa **model**, que representa la realidad
- La capa **controller,** que conoce los métodos y atributos del modelo, recibe y realiza lo que el usuario quiere hacer
- La capa **view**, que muestra un aspecto del modelo y es utilizada por la capa anterior para interaccionar con el usuario.

Estructurales como puede ser el patrón de diseño Façade, que se utiliza para facilitar la modificación de los componentes de los subsistemas, para ello sólo se necesitaría realizar cambios en la interfaz/fachada, y los clientes pueden permanecer ajenos a ello. Además así tendríamos ocultos dichos componentes a los clientes.

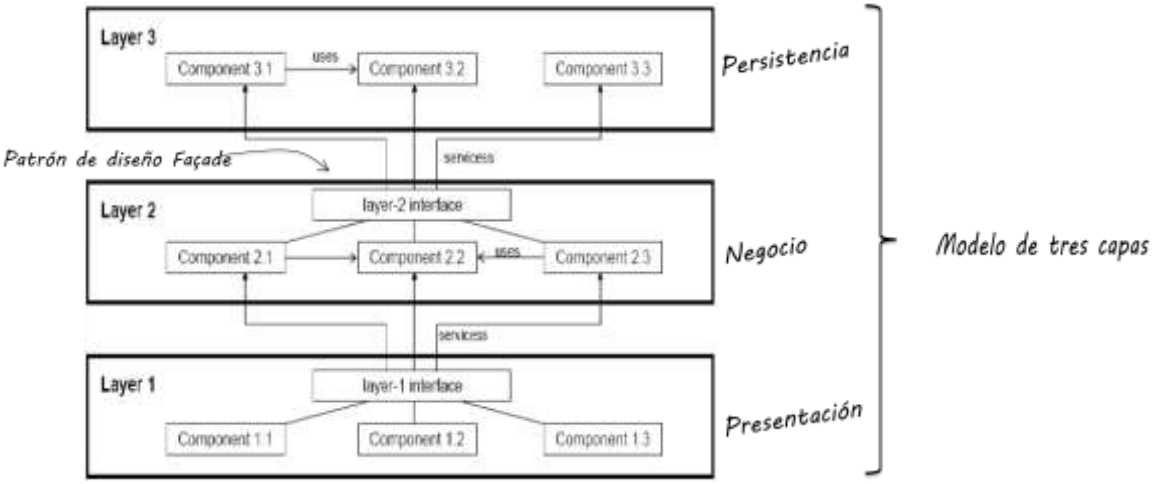


Figura 1.2. Patrones de diseño y estructurales

Otro patrón sería el patrón DAO, que nos proporcionan una interfaz común para el transporte de los datos desde la base de datos hacia la capa de lógica de negocio y viceversa.

Una vez que se tienen identificado todos los riesgos de los formularios que se desarrollan en la aplicación web, se generan unos informes dependiendo del tipo que se halla seleccionado previamente desde la aplicación, para la elaboración de dichos informes en PDF que serán almacenados en el directorio **Home** del usuario, se utiliza la herramienta IText, que es una librería de código abierto que permite crear y manipular los documentos PDF.

1.2 Resumen de Todos los Aspectos

Introducción: En este primer capítulo se centra en dar a conocer el proyecto a la persona que lo lea, explicando el porqué del mismo, que se espera de él y la justificación de las decisiones que he adoptado.

Aspectos Teóricos: Los aspectos teóricos describirán brevemente aquellos conceptos, herramientas y tecnologías que se han utilizado para dar forma a este proyecto.

Planificación del Proyecto y Resumen del Presupuesto: Describirá el plazo de ejecución del proyecto con un diagrama de Gantt donde podrá verse el tiempo asignado a cada tarea, hitos, etc.

Análisis: Este apartado tiene como objetivo la obtención de una especificación detallada del sistema que se han creado, que satisfaga las necesidades del cliente y sirva de base para el posterior diseño del sistema.

Diseño del Sistema: Aquí se definirá la arquitectura del sistema y el entorno tecnológico que le va a dar soporte, además de la especificación detallada de los componentes que forman el sistema.

Con esta información, se generan las especificaciones de construcción concernientes a nuestra aplicación y la descripción técnica del plan de pruebas.

Implementación del Sistema: En la implementación del sistema se describirán los elementos necesarios para la elaboración del sistema (herramientas, lenguajes, programas, etc.) así como todos los conceptos relacionados que tengan que ver con la creación del sistema.

Desarrollo de Pruebas: Se describe el resultado de las pruebas comentadas en los puntos anteriores.

Manuales del Sistema: En este punto, se elaboran los diferentes manuales de instalación y de ejecución, de nuestra aplicación que darán apoyo a las posibles dudas sobre el sistema

Conclusiones y Ampliaciones: Uno de los apartados más importantes para este proyecto, donde se detallarán las conclusiones obtenidas y las posibles ampliaciones, donde seguramente en un futuro se podrá mejorar dicho proyecto para que cumpla con todos los requisitos reales que la empresa pida.

Presupuesto: Se detallarán a continuación el presupuesto obtenido a partir de la medición de los tiempos en una tabla resumen.

Referencias Bibliográficas: Referencias utilizadas a lo largo del desarrollo del proyecto, libros, artículos, revistas, otros proyectos y enlaces Web.

Apéndices: Parte de estructuras principales de código utilizado para crear la aplicación, junto con todo aquello que no tenga cabida en otras secciones y sea considerado de interés.

Capítulo 2. Introducción

2.1 Justificación del Proyecto

Como se comento ya en el capítulo de motivación, la justificación surge para aprovechar el tiempo y no perderlo a la hora de completar pasos a la hora de ya se a realizar el plan de riesgos, la identificación de estos o en análisis, ya que muchos de esos pasos se pueden automatizar y no sería necesario el volver a escribir varias veces lo mismo, pudiendo hacerlo el ordenador por ejemplo.

Otra justificación podría ser que realizando una carrera de informática o una especialidad de la informática en el caso del máster, el tener que seguir trabajando por medio de documentos Excel y Word pudiendo hacerse por medio de una aplicación Web, para mí fue un extra el querer realizar este proyecto.

El proyecto consta de una serie de interfaces diferentes, dependiendo del tipo de usuario que acceda a ellas, hay tres tipo de roles, tendríamos el rol del profesor que sería el Administrador y luego los alumnos que habría dos tipos, uno de ellos podría tener el rol de Manager, que además de crear los planes de riesgo e identificarlos y hacer el análisis sobre ellos, serán los elegidos para poder crear los diferentes equipos para el proyecto con el resto de alumnos que serán el último rol que serán el de simples miembros, que realizan el estudio de los riesgos.

Una vez que cada Manager y cada miembro del equipo, evalúen, identifiquen y analicen los diferentes tipos de riesgos rellenando los diferentes formularios que hay para cada etapa, se crearán sus correspondientes informes en PDF, que podrán ser descargados accediendo al último punto del gestor de riesgos, donde el usuario pulsará sobre aquel informe que desee descargarse, dicho informe se descargará en la sección Home que tenga el usuario asignado en su ordenador.

Podrá repetir dichas acciones las veces que quiera, ya que los datos son almacenados tanto temporalmente en variables de sesión, como a largo plazo en una base de datos, además podrá recuperar viejas versiones que tenga de planes de gestión de riesgos, haciendo que estás recuperen los valores de versiones anteriores a la que está en ese momento y descargarse el informe de esa versión anterior.

2.2 Objetivos del Proyecto

El objetivo del proyecto, es el de desarrollar una aplicación web que supla el sistema actual de gestión de riesgos de esas asignaturas que actualmente se lleva a cabo.

Principalmente para aumentar la probabilidad y el impacto de eventos positivos y disminuir la probabilidad y el impacto de eventos negativos para dichos proyectos.

1. Diferenciando los diferentes roles de personas
2. Posibilidad de recuperación de viejas versiones
3. Sistema para envío por correo de eventos a los usuarios del proyecto
4. Autorelleno en la manera de lo posible de campos en los formularios
5. Gestión de la planificación de los riesgos informáticos
6. Identificación de los riesgos que el usuario crea conveniente
7. Análisis de aquellos riesgos identificados anteriormente que cumplan la condición de ser riesgos con probabilidades altas de producirse utilizando para ello la matriz de impactos y probabilidades, poniendo el usuario el corte que estime oportuno para analizar aquellos que sobrepasen ese límite.
8. Creación de informes sobre la planificación, la identificación y el análisis de riesgos

2.3 Estudio de la Situación Actual

En esta sección deben identificarse y describirse sistemas similares al que se va a desarrollar, estableciendo una comparación entre lo que ofrecen estos sistemas y lo que pretendemos lograr con el proyecto, para de esta forma diferenciar nuestro desarrollo de lo ya existente.

Echando un vistazo por Internet, podemos encontrar gran cantidad de programas que se dedican a la gestión de riesgos los hay de todos los tipos y de todos los grados, una gran parte de ellos, tienen opciones que van mucho más allá de lo que este proyecto contempla, así que solo nos fijaremos en las partes comunes de estos. Tenemos por ejemplo un sistema llamado **@RISK 6** de la empresa **Palisade Corporation**, que hace un montón de cosas espectaculares siempre y cuando estés funcionando con hojas EXCEL (Microsoft, de pago) y tiene en su nueva versión integridad con el Project (también de pago y de Microsoft), hace más bien como de plugin especializado en gestión de proyectos en programas Microsoft, y no deja elegir otro tipo de plataformas. Demasiado amplio, demasiado caro y demasiados requisitos que hay que pagar aparte del programa en sí, si quieres que funcione, cosa que el del proyecto no necesita de otras empresas ya que la mayoría está utilizando software libre para que funcione.

Tenemos software también no muy caro y que no necesitan de terceras personas, como es el caso de **PMP Mobile - Suite Gestión de Riesgos (PMPMSKM)**, es un producto creado por managers bastante intuitivo y fácil de utilizar, genera también documentación con gráficos, está basada en la metodología PMI® reflejada en los estándares de PMBOK® y por supuesto analiza los riesgos de un proyecto, pero solo está disponible para sistemas Android que tengan al menos versión mínima 3.0 de Android (honeycomb, nivel de api 11) para funcionar correctamente. Con nuestro proyecto, podrías utilizar cualquier navegador para utilizar el sistema de la aplicación y no necesariamente para los documentos sería necesario un programa en especial como pudiera ser el Adobe Reader (gratuito) y otros más como Foxit, Nitro PDF e incluso navegadores Web como el Firefox o el Chrome tiene sus propios lectores

Luego tendríamos ya software de peso del estilo de **GxSGSI** y **RBOX** que son soluciones informáticas muy avanzadas, más de lo que en realidad el proyecto abarca, los puntos comunes son muy parecidos, puede que sus gráficos sean más espectaculares, pero la forma de evaluar los riesgos son muy parecidos, incluso en los aspectos técnicos, los tres estamos utilizando las mismas aplicaciones, Unix o Windows con una base de datos MySQL un WebServer de Apache Tomcat y utilización por medio de la Web con los navegadores, pero claro esos gráficos y el modo de presentación de las cosas tienen un precio, que debes pagar mensual, trimestral o anualmente, no es un software que tenga un precio único y puedas utilizarlo hasta que se decida cambiar de producto.

Como mencione anteriormente nuestra aplicación, necesitará de una base de datos que para este caso hemos optado por MySQL (MySQL 5.5.) entre otras cosas, porque es gratuita y porque es la que deja agregar OpenShift al gear de la aplicación para subirla a su servidor, junto a ella hemos agregado por comodidad de administrar la base de datos phpMyAdmin (4.0) y como contenedor Web, se ha escogido Tomcat7 (JBoss EWS 2.0), porque entre otras cosas implementa las especificaciones de los servlets y de JavaServer Pages (JSP), que es lo que

utilizo para la capa de presentación de mi modelo MVC y no menos importante me proporciona un entorno web HTTP para que mi código Java se ejecute, lenguaje, que es en lo que está escrita la aplicación, ya que es el lenguaje que mas conozco y con el que más he trabajado, tanto en la universidad, como en el trabajo, utilizando para ello la herramienta de programación también de código abierto, como es Eclipse, para desarrollar dicha aplicación, sabiendo ya de antemano, que el conjunto de todo ello no me iba a levantar dolores de cabeza, por eso y porque son herramientas gratuitas y de código abierto todas ellas, son mi elección para crear el este proyecto fin de Máster.

2.3.1 Evaluación de Alternativas

2.3.1.1 @Risk 6

Figura 2.1. @Risk 6

@Risk 6 es utilizado como analizador de riesgos, existe tres versiones diferentes, una industrial, otra denominada Pro y la que veremos más en profundidad la Standard. Su principal función es la de la realización de análisis de riesgos, utilizando para ello la simulación para mostrar múltiples resultados posibles mediante una hoja de cálculo Excel indicando en ella la probabilidad que hay de que se produzcan por medio de gráficos, por otro lado también tiene la función de ayudar a la planificación de las estrategias de administración de riesgos, utilizando la simulación Montecarlo[1].

Ventajas	**Inconvenientes**
• Generador de simulación avanzado • Más de 50 funciones de distribución incorporadas • Galería integrada de distribuciones • Permutación de funciones de @RISK • Funciones Compound y Six Sigma • Variedad de gráficos y tablas de resultados • Análisis de sensibilidad y escenarios • Correlación de entradas • Artista de Distribuciones para dibujar distribuciones	• Las simulaciones de @RISK se calculan al 100% dentro de Excel • Solo versiones de 32 bits y 64 bits de Microsoft Windows XP-8 • Integración solo con Microsoft Project • Precio

[1] La simulación Monte Carlo es una técnica matemática computarizada que permite tener en cuenta el riesgo en análisis cuantitativos y tomas de decisiones.

2.3.1.2 PMP Mobile - Suite Gestión de Riesgos

Figura 2.2. PMP Mobile

PMP Mobile es un analizador vectorial de riesgos, está diseñada para apoyar en las tareas relativas a la gestión de riesgos (identificación, seguimiento, generación de documentación...) y está totalmente basada en la metodología PMI® reflejada en los estándares de PMBOK®.

Ventajas	Inconvenientes
• Generación de documentación de proyecto • Exportación e importación de proyectos • Realización de análisis de proyecto • Ecosistema de datos y funcionalidades • Sencillez y accesibilidad	• Sistema solo Android • No hay versiones fura de móviles • Algunas funcionalidades quedan demasiado cortas o escasas

2.3.1.3 GxSGSI

Figura 2.3. GxSGSI

GxSGSI es una herramienta de gestión de riesgos, que permite la identificación y evaluación de las amenazas, vulnerabilidades e impactos, el cálculo del riesgo y otras muchas características más para obtener una implantación de un sistema de gestión de seguridad de la información.

Ventajas	Inconvenientes
• Cumplimiento de norma ISO/IEC 27001:2013 • Multiempresa • Generación de informes • Automatización y realización completo de análisis de riesgos	• Arquitectura nada novedosa • Utilización de programas básicos para su construcción • Opciones fuera del ámbito de este proyecto • Precio

2.3.1.4 RBOX

Figura 2.4. RBOX

RBOX es una solución modular para la Gestión de Seguridad de la Información y Cumplimiento de Estándares, contiene diferentes bloques para su gestión y los resultados del Análisis y la Gestión de Riesgo dependen, principalmente, de un conjunto de datos agrupados en cuatro modelos, como son: modelo de entidades, modelo funcional, modelo de negocio, modelo de gestión.

Ventajas	**Inconvenientes**
• Importación y exportación de datos • Multiempresa • Multiusuario • Control de Estados • Múltiples escenarios: • Customización de Informes	• Arquitectura nada novedosa • Utilización de programas básicos para su construcción • Opciones fuera del ámbito de este proyecto • Precio

Capítulo 3. Aspectos Teóricos

Esta sección está destinada a describir brevemente aquellos conceptos, herramientas y tecnologías existentes que vamos a usar en nuestro proyecto.

3.1 CONCEPTOS

3.1.1 PMBOK

El PMBOK según descripción de **La Guía de los Fundamentos para la Dirección de Proyectos** (Guía del PMBOK®) <<…*es una norma reconocida en la profesión de la dirección de proyectos. Por norma se hace referencia a un documento formal que describe normas, métodos, procesos y prácticas establecidos. Al igual que en otras profesiones, como la abogacía, la medicina y las ciencias económicas, el conocimiento contenido en esta norma evolucionó a partir de las buenas prácticas reconocidas por profesionales dedicados a la dirección de proyectos, quienes contribuyeron a su desarrollo.* >>

El origen tiene lugar en 1996 por el Project Management Institute (PMI) quien publicó por primera vez está guía. Ese documento era el fruto en cierta medida de un trabajo anterior, que comenzó con un libro blanco publicado en 1983 llamado **"Ethics, Standards, and Accreditation Committee Final Report."**[2]. Cuya Segunda edición se publicó en el año 2000.

En 2004, apareció la Tercera edición, se publicó con grandes cambios con respecto a ediciones anteriores. La siguiente edición se publicó en 2009. Y la última edición, la quinta, está escrita en Inglés, que se publicó en 2013, en castellano creo que de momento solo hay traducida hasta la cuarta.

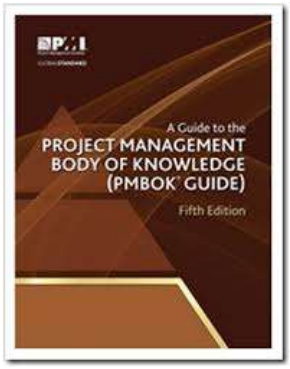


Figura 3.1. PMBOK

[2] A Guide to the Project Management Body of Knowledge, copyright page, edition 2 ISBN 1-880410-12-5, and edition 3 2004 ISBN 978-1-930699-45-8, and edition 4 2008 ISBN 1-933890-51-7

La finalidad principal del PMBOK es identificar, concentrar y publicar las mejores prácticas generalmente aceptadas en la Dirección de Proyectos. << ***Buenas prácticas*** >> significa que se está de acuerdo, en general, en que la aplicación de estas habilidades, herramientas y técnicas puede aumentar las posibilidades de éxito de una amplia variedad de proyectos. Además también proporciona y promueve un vocabulario común en el ámbito de la profesión de la dirección de proyectos, para analizar, escribir y aplicar conceptos de la dirección de proyectos.

En este proyecto solo contaremos con una parte de la guía, concretamente nos centraremos en el capítulo 11, que trata sobre la ***Gestión de los Riesgos del Proyecto****.*

3.1.2 BOEHM

Su origen provine de Barry Boehm, este modelo fue propuesto por 1986 en su artículo "A Spiral Model of Software Development and Enhancement". Básicamente consiste en una serie de ciclos que se repiten en forma de espiral, comenzando desde el centro. Para nuestro proyecto solo nos interesa la etapa referida a la gestión de riesgos.

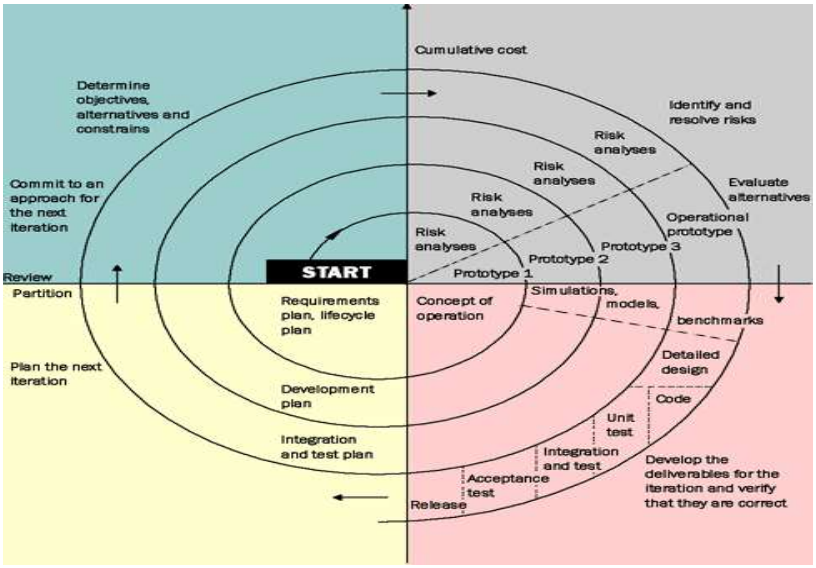


Figura 3.2. Modelo Boehm

Boehm divide la gestión de riesgos en dos fases principales:

1. Valoración de riesgos: Esta fase trata con los trabajos relacionados con la identificación, análisis y priorización de los riesgos.
2. Control de riesgos: Esta fase define las estrategias a aplicar el riesgo, la manera de identificar los indicadores y la monitorización.

Todos los pasos son similares a la metodología de PMBOK, exceptuando la planificación de la Gestión de Riesgos que en el modelo de Boehm, no tiene y que los análisis cualitativo y cuantitativo de los riesgos los elimina, por un análisis de riesgos que sustituye al análisis cualitativo y una priorización de los riesgos sustituyendo al cuantitativo, que es lo que se hará en el proyecto.

3.1.3 Gestión de los Riesgos de un Proyecto

Comenzaremos por la definición de un riesgo, para luego ir adentrándonos en su gestión y porque es tan importante para un proyecto.

La definición de riesgo no es algo que este realmente definido claramente y que sea único para todo el mundo, depende de donde busquemos, podemos encontrar diferentes definiciones para el. Aquí ya que se está siguiendo una guía nos quedaremos con la definición que hace está guía que es << *Un riesgo es un evento o condición incierta que, si sucede, tiene un efecto positivo o negativo en por lo menos uno de los objetivos del proyecto, tales como el alcance, el cronograma, el coste y la calidad*. >>

Estos dos tipos de riesgos, dan lugar a otros termino asociado a ellos, si el riesgo es negativo, se le llama también amenaza, si los riesgos son positivos se les denomina oportunidades. La gestión de los riesgos intenta hacer que los riesgos negativos (amenazas) sean eliminados o evitados por medio de acciones como la transferencia, la mitigación y la aceptación, mientras que los riesgos positivos (oportunidades) deben ser aprovechados por medio de la mejora, la compartición y la aceptación

Para ello el PMBOK incluye un proceso llamado **Gestión de los Riesgos del Proyecto** que su objetivo principal es el aumentar la probabilidad y el impacto de eventos positivos (oportunidades), y disminuir la probabilidad y el impacto de eventos negativos (amenazas) para el proyecto, para llevar a cabo esto y encontrar estos riesgos y tratarlos según su tipo, el PMBOK propone una serie de pasos a seguir, en este proyecto se centrará en principalmente en los cinco primeros pasos, la planificación de la respuesta, será el estudio de los riesgos mediante en este proyecto mediante la creación de informes, ya que existen muchos más técnicas y herramientas que al igual que la monitorización y el control de riesgos puede alargar en exceso este proyecto que de momento no se contemplarán, pero que puede, que un futuro mediante ampliaciones a este proyecto se puedan llegan a dar:

1. **Planificar la Gestión de Riesgos**
2. **Identificar los Riesgos**
3. **Realizar el Análisis Cualitativo de Riesgos**
4. **Realizar el Análisis Cuantitativo de Riesgos**
5. **Planificar la Respuesta a los Riesgos**
6. **Monitorizar y Controlar los Riesgos**

3.1.4 Planificación de la Gestión de Riesgos

El plan de gestión de riesgos describe la manera en que se estructurará y realizará la gestión de riesgos en el proyecto. Una planificación que mejora la probabilidad de éxito de los otros cinco procesos de gestión de riesgos; proporciona los recursos y el tiempo suficientes para las actividades de gestión de riesgos y para establecer una base acordada para evaluar los riesgos. El proceso debe iniciarse tan pronto como se concibe el proyecto y debe completarse en las fases tempranas de planificación del mismo.

El plan de gestión de riesgos incluye lo siguiente:

Metodología: Define los métodos, las herramientas y las fuentes de información que pueden utilizarse para realizar la gestión de riesgos en el proyecto

Herramientas y Tecnologías: Se utilizarán para la gestión de riesgos, al menos una de ellas para cada apartado de la metodología. Entre las cuales se encuentran, revisiones de la documentación, matriz de probabilidad e impacto**,** juicio de expertos, reuniones sobre el estado del proyecto.

Roles y responsabilidades. Define al líder, el apoyo y a los miembros del equipo de gestión de riesgos para cada tipo de actividad del plan de gestión de riesgos, y explica sus responsabilidades.

Presupuesto. Asigna recursos, estima los fondos necesarios para la gestión de riesgos, a fin de incluirlos en la línea base del desempeño de costos y establece los protocolos para la aplicación de la reserva para contingencias.

Calendario. Define cuándo y con qué frecuencia se realizará el proceso de gestión de riesgos a lo largo del ciclo de vida del proyecto, establece los protocolos para la utilización de las reservas para contingencias del cronograma y prevé las actividades de gestión de riesgos que deben incluirse en el cronograma del proyecto.

Categorías de riesgo. Proporciona una estructura que asegura un proceso completo de identificación sistemática de los riesgos con un nivel de detalle coherente, y contribuye a la efectividad y calidad del proceso Identificar los Riesgos.

Definiciones de la probabilidad e impacto. Requerido para asegurar la calidad y credibilidad del proceso. Se adaptan a cada proyecto individual durante el proceso. En la probabilidad se obtienen unas definiciones las cuales van con una determinada probabilidad de que suceda.

Muy Alta 90 %	Objetivos críticos del proyecto están seriamente impactados o no se cumplirán (coste, calendario, calidad o satisfacción del cliente)
Alta 70 %	Objetivos críticos del proyecto que están amenazados
Media 50 %	Algunos objetivos del proyecto pueden verse afectados.
Baja 30 %	Remediables. Los objetivos del proyecto pueden verse afectados, pero sin grandes problemas
Muy Baja 10 %	Fácilmente remediable. Los objetivos del proyecto no serán afectados.

Figura 3.3. Definiciones de Probabilidad

Mientras en la **definición de impactos por objetivo**, se tienen varios objetivos como pueden ser el coste, el tiempo, el alcance y la calidad, a estos objetivos se les asigna un impacto dependiendo de cuanto se aumente o incrementen esos objetivos, para luego realizar con tanto las probabilidades como con los impactos una matriz, la matriz de probabilidades e impacto.

Impacto sobre los objetivos principales *Definirlo sobre amenazas*					
Objetivos	Muy bajo / 5%	Bajo / 15%	Medio / 30%	Alto / 55%	Muy alto / 90%
Coste	Aumento del coste insignificante	Incremento del coste <15%	Incremento del coste entre el 15-30%	Incremento del coste entre el 30-55%	Incremento del coste > 55%
Tiempo	Aumento del tiempo insignificante	Incremento de tiempo <7,5%	Incremento de tiempo entre el 7,5-15%	Incremento de tiempo entre el 15-27,5%	Incremento del tiempo > 27,5%
Alcance	Aumento del alcance insignificante	Áreas de alcance secundarias afectadas	Áreas de alcance principales afectadas	Reducción del alcance inaceptable para el cliente	El elemento terminado del proyecto es efectivamente inservible
Calidad	Aumento del calidad insignificante	Sólo las aplicaciones muy exigentes se verán afectadas	La reducción de la calidad requiere la aprobación del cliente	Reducción de la calidad inaceptable para el cliente	El elemento terminado del proyecto es efectivamente inservible

Figura 3.4. Definiciones de Impacto por Objetivo

Matriz de probabilidad e impacto. Los riesgos se clasifican por orden de prioridad de acuerdo con sus implicaciones potenciales de tener un efecto sobre los objetivos del proyecto anteriormente creado. El método típico para priorizar los riesgos consiste en utilizar una tabla de búsqueda o una Matriz de Probabilidad e Impacto.

Probabilidad	**Muy Alta**	***0,90***	0,05	0,14	0,27	0,50	0,81
	Alta	***0,70***	0,04	0,11	0,21	0,39	0,63
	Media	***0,50***	0,03	0,08	0,15	0,28	0,45
	Baja	***0,30***	0,02	0,05	0,09	0,17	0,27
	Muy Baja	***0,10***	0,01	0,02	0,03	0,06	0,09
			0,05	***0,15***	***0,30***	***0,55***	***0,90***
			Muy Bajo	**Bajo**	**Medio**	**Alto**	**Crítico**
			Impacto				

Figura 3.5. Matriz de Probabilidad e Impacto

Contingencia de planes. Es un tipo de plan preventivo, predictivo y reactivo. Presenta una estructura estratégica y operativa que ayudará a controlar una situación de emergencia y a minimizar sus consecuencias negativas. Como por ejemplo que exista un plan específico de contingencia presupuestaria, por si se produce un sobrecoste en el proyecto, poder tener un plan para esta situación negativa o un plan de contingencia para el tiempo, tareas que no se han diseñado con el tiempo suficiente, que no se conviertan en retrasos para el proyecto.

Formatos de los informes. Definen cómo se documentarán, analizarán y comunicarán los resultados de los procesos de gestión de riesgos.

3.1.5 Identificación de los Riesgos

Identificar los Riesgos es el proceso por el cual se determinan los riesgos que pueden afectar el proyecto y se documentan sus características. Entre las personas que participan, estarán los managers y los demás miembros del equipo de ese proyecto en concreto.

Se pueden utilizar varias técnicas para la identificación de esos riesgos:

Técnicas de recopilación de Información como por ejemplo:

- Tormenta de ideas: El objetivo es realizar una lista de riesgos del proyecto. Se pueden realizar varias, habitualmente con grupos multidisciplinarios de expertos que no forman parte del equipo de proyecto. Se pueden usar las categorías de riesgo (RBS) para dirigir el trabajo. Al final se debe obtener una lista
- Técnicas Delphi: También se pueden usar estas técnicas con el mismo objetivo, la principal diferencia es que usando Delphi, los expertos participan de forma anónima.
- Entrevistas: Se organizan entrevistas con miembros experimentados de equipos de proyecto y con los stakeholders para ayudar a identificar los riesgos.
- Análisis causal: Son técnicas sencillas de identificación de causas para un problema, en este caso un riesgo. En el libro "Técnicas Participativas para la Planeación" (Sánchez Guerrero, 2003), en el capítulo 5, se puede encontrar una descripción detallada de esta

Análisis de listas de control. Esta técnica utiliza el uso de listas de control de elementos del riesgo. También se las conoce como listas de comprobación.

Análisis de supuestos. Cada proyecto y cada riesgo identificado se conciben y desarrollan tomando como base un grupo de hipótesis, escenarios y supuestos.

Técnicas de Diagramación. Del estilo a diagramas de causa y efecto, diagramas de flujo o de sistemas, diagramas de influencias.

Análisis SWOT. Esta técnica examina el proyecto desde el punto de vista de las Debilidades, Amenazas, Fortalezas y Oportunidades (DAFO) del proyecto y de la organización con el objetivo de identificar riesgos adicionales.

Juicio de Expertos. Puede haber personas que tengan gran experiencia en proyectos y que pueden identificar directamente algunos riesgos adicionales al proyecto.

En la lista de riesgos de **salida** que tendremos:

- **Identificador**: Identifica al riesgo, debe ser único para cada código.
- **Nombre**: Puede ser un nombre corto o una descripción muy breve.
- **Descripción**: Es una descripción del riesgo en la que se define todo lo que se conoce sobre la manera en que afecta al proyecto en ese momento.
- **Clase**: (*opcional*) Se trata de decir si es un riesgo de producto, de proceso o de proyecto.

- **Respuesta potencial**: Durante la identificación del riesgo también se identifican algunas posibles respuestas y estrategias que serán recogidas aquí.

Identificar los Riesgos es un proceso iterativo debido a que se pueden descubrir nuevos riesgos o pueden evolucionar conforme el proyecto avanza a lo largo de su ciclo de vida. Este proceso de identificación de riesgos nos llevará al proceso de análisis de riesgos.

3.1.6 Análisis de los Riesgos

El análisis de riesgos evalúa los riesgos identificados en la fase anterior para determinar la probabilidad de que ocurran, el impacto del riesgo, el impacto acumulativo de múltiples riesgos y la prioridad de cada riesgo. El análisis de riesgos es un proceso de apoyo a los procesos de planificación de proyectos, y depende de la identificación de los riesgos del proyecto.

Se definen dos fases de análisis:

Análisis cualitativo: Donde básicamente se caracteriza el riesgo y se calcula su probabilidad, su impacto y su importancia. Además se hace un primera priorización.

Análisis cuantitativo: Donde se hace una valoración cuantitativa del alcance de ciertos riesgos seleccionados por su especial importancia con el objetivo de calcular su impacto económico en el proyecto y poder, entre otras cosas, evaluar las reservas necesarias para los planes de contingencia de los riesgos.

En nuestro proyecto seguiremos lo que dicta el Boehm, que sólo hace el equivalente del análisis cualitativo, posteriormente se hace una priorización y se realizan los análisis cuantitativos sólo para aquellos riesgos priorizados en que sea necesario en función de las estrategias. Esta valoración se hace durante la planificación y la resolución de riesgos.

Durante la **evaluación** se debe obtener una mejor comprensión del riesgo. Se cuantifican, en lo posible, los siguientes conceptos:

Categorización de Riesgos

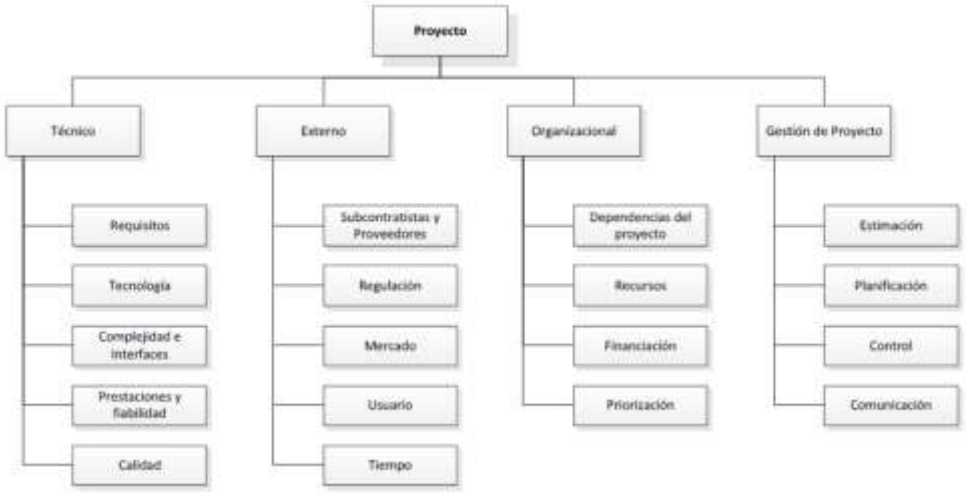


Figura 3.6. Categorías de un Riesgo

Evaluación de la Probabilidad e Impacto de los riesgos

Para cada riego identificado se evalúa la probabilidad de ocurrencia del riesgo y el impacto posible del riesgo sobre el proyecto.

La evaluación de la probabilidad se hace de acuerdo a una escala de rangos que se define en la matriz de probabilidad e impacto anteriormente dibujada. Estas escalas suelen definir rangos del mismo tamaño y se toma como valor del rango el valor medio.

La evaluación del impacto se hace también de acuerdo a una escala de rangos que se define en la tabla de escalas de impacto. Habitualmente estas escalas no son lineales, suelen ser escalas logarítmicas. De ese modo se deja mucho espacio para los riesgos de menor importancia, mientras que a partir de un cierto valor todos los riesgos pasan a ser importantes.

Priorización de los Riesgos

La priorización de riesgos produce una lista ordenada de elementos de riesgo identificados y analizados. Usando las técnicas habituales se realiza una priorización de manera que algunos riesgos, por su importancia, requerirán un seguimiento más intenso. Los riesgos que no se priorizan se mantienen en una lista de riesgos potenciales (no priorizados) para futuras revisiones.

Matriz de Probabilidad e Impacto

Una vez evaluada su probabilidad e impacto en el proyecto, se procede a calcular su importancia mediante una matriz de probabilidad e impacto de manera que el riesgo se priorice para posteriores estudios y evaluaciones o directamente para ser monitorizado en el proyecto.

Juntas, la definición de las escalas de impacto y la matriz de probabilidad e impacto, definirán cuáles son los puntos de corte para decidir que un riesgo debe ser tenido en cuenta, desestimado o ser objetivo de una supervisión durante el desarrollo. Al mismo tiempo definen también cuáles son estos valores para los objetivos principales del proyecto.

La ***salida*** que dará el resultado de la identificación como del análisis será la:

Obtención de una lista de riesgos priorizada por su importancia. Además, se obtiene una lista de riesgos potenciales cuya importancia ha resultado baja pero que se mantiene para ser reestudiada cada cierto tiempo en busca de posibles cambios de las condiciones que hagan emerger nuevos riesgos importantes.

Lista de riesgos con ciertas características como la siguiente, aquellos que superen el valor de corte de la priorización, son riesgos que pueden afectarnos en el proyecto con un alto índice de impacto sobre el:

ID	Nombre	Responsable	Probabilidad	Impacto				Impacto	0,49	Response	Notes
				Presup.	Planific.	Alcance	Calidad		Priorización		
1	Aprendizaje de nuevo personal	Todos	Alta	Muy Bajo	Muy Bajo	Medio	Medio	0,21		Tener a alguien del proyecto como tutor del nuevo personal, para garantizar el aprendizaje rápido y seguro	
2	Baja de trabajadores	RRHH	Muy Alta	Medio	Alto	Medio	Medio	0,50		Tener una lista con posibles nuevos empleados que sustituyan a los de la baja	
3	Bajo rendimiento del personal	Jefe Proyecto	Baja	Muy Bajo	Alto	Crítico	Alto	0,27		Tener hitos que deberán cumplir los empleados	Llevar un listado con las tareas que ha hecho cada empleado
4	Caducidad prematura de la tecnología usada	Desarrollador	Baja	Medio	Medio	Alto	Alto	0,17		Hacer un estudio previo de la tecnología que vamos a usar	
5	Cambio de herramientas durante el proyecto	Jefe Proyecto	Muy Baja	Medio	Medio	Alto	Medio	0,06		Adaptación lo más rápido posible, o tener un plan "b", similar con otras	Tener no solo unas herramientas al principio sino varias por si falla las primeras

Figura 3.7. Identificación de Riesgos

3.1.7 Planificación de la Respuesta a los Riesgos

La Planificación de la gestión de cada riesgo genera un plan para tratar cada riesgo de importancia alta del proyecto. Ayuda, por tanto, a manejar cada elemento de riesgo, incluyendo la coordinación de los planes individuales de elementos de riesgos entre ellos y con respecto al plan general de riesgos.

En el marco del proceso Planificar la Respuesta a los Riesgos, se seleccionan y se acuerdan las respuestas apropiadas, y se incluyen en el registro de riesgos. El registro de riesgos debe escribirse con un nivel de detalle que se corresponda con la clasificación de prioridad y la respuesta planificada. A menudo, los riesgos altos y moderados se tratan en detalle. Los riesgos considerados de baja prioridad se incluyen en una "lista de supervisión" para su monitoreo periódico.

Las técnicas de planificación de la gestión de riesgos más habituales son las siguientes:

- **Estrategias para Riesgos Negativos o Amenazas**
  - **Eliminación del riesgo**
  - **Transferencias del riesgo**
  - **Evitación del riesgo**
  - **Retención del riesgo**
- **Estrategias para Riesgos Positivos u Oportunidades**
  - **Aprovechamiento**
  - **Mejora**
  - **Compartición**
  - **Aceptación**
- **Análisis coste-beneficio:** El análisis coste-beneficio es un procedimiento para formular y evaluar respuestas a los riesgos que sean consistentes en la comparación de costes y beneficios, con el objetivo de que los beneficios sean mayores que los costes
- **Planes de contingencia.** Es un tipo de plan preventivo, predictivo y reactivo
- **Juicio de Expertos:** Puede haber personas que tengan gran experiencia en proyectos y que pueden identificar directamente algunos riesgos adicionales al proyecto.

Se deben obtener las siguientes salidas:

Actualización de las fichas de riesgos con todos los planes de respuesta elaborados.- De toda la lista de riesgos identificados y priorizados se seleccionarán aquellos riesgos de mayor impacto global y se documentarán.

3.1.8 Monitorización de Riesgos

La monitorización del riesgo consiste en controlar el progreso del proyecto en lo relativo a resolución de riesgos durante la fase de desarrollo, tomando las acciones correctoras cuando sea necesario. Esta actividad se lleva a cabo usualmente evaluando los indicadores de los factores de riesgo. Está fase no se llevará a cabo en el proyecto, pero se explica para su conocimiento.

Las técnicas a utilizar podrían ser:

- **Reevaluación de los Riesgos**
- **Auditorías de los Riesgos**
- **Análisis de Variación y de Tendencias**
- **Análisis de Reserva**
- **Reuniones sobre el Estado del Proyecto**

Y como salidas tendríamos:

- **Actualizaciones al Registro de Riesgos**
- **Actualizaciones a los Activos de los Procesos de la Organización**
- **Solicitudes de Cambio**
- **Actualizaciones al Plan para la Dirección del Proyecto**
- **Actualizaciones a los Documentos del Proyecto**

3.2 TECNOLOGÍAS

3.2.1 Struts2

Es una implementación del modelo 2/patrón de diseño MVC que facilita la creación de aplicaciones web en Java. Fue creada por Craig McClanahan y donada a la Apache Software Foundation en el año 2000. El Framework WebWork[3] se dividió de Apache Struts con el objetivo de ofrecer mejoras, al tiempo que conserva la misma arquitectura general del marco original de Struts. En diciembre de 2005, se anunció que WebWork 2.2 cambiara a denominarse Apache Struts 2, que alcanzó su primer lanzamiento completo en febrero de 2007.

Existen dos tipos versiones de Struts: la 1.x y la 2. Struts 1 era el único *Framework* existente de este tipo y en su diseño original había deficiencias en cuanto a flexibilidad y simplicidad de uso. Para solucionar dichas deficiencias se creó y actualizo está versión pasando a la actual versión 2. El problema de esta segunda versión con respecto a la primera es por aquel entonces solo existía este tipo de Framework, ahora para la versión 2 existe mucha más competencia y su renombre ha bajado por culpa entre otros Frameworks como Spring o JSF.

Porque se utiliza este tipo de Frameworks, una razón muy sencilla **Struts 2 hace que el desarrollo de aplicaciones web sea lo más simple posible para los desarrolladores**.

Las características que definen este Framework son:

- Framework basado en Actions.
- Un gran grupo de desarrolladores y comunidad de usuarios.
- Anotaciones y opciones de configuración XML.
- Acciones implementadas como POJOs
- Integración con Spring, SiteMesh y Tiles.
- Integración con el lenguaje de expresión OGNL.
- Object-Graph Navigation Language. Lenguaje de expresión para hacer get y set de propiedades sobre objetos java.
- Integración con Ajax.
- Múltiples opciones de vista (JSF, Freemarker, Velocity and XSLT)
- Plug-ins para ampliar y modificar el Framework

Struts 2 se compone de ciertos elementos como son:

Actions. Son la unidad más básica de trabajo que se asocia con una request HTTP invocada por el usuario.

Interceptors. Los Interceptors son conceptualmente lo mismo que un *filtro HTTP* o la clase Proxy, proveen un camino para suplir pre-processing y post-processing a través

[3] Fue Framework de aplicaciones web, basada en Java y desarrollado por OpenSymphony

del action. Pueden ser secuencializados y ordenados. Tienen acceso al contexto de ejecución del action, a las variables del entorno y propiedades de ejecución.

Pila de valores/OGNL. La pila de valores es un elemento donde se almacenan valores, mientras que OGNL es un camino unificado para acceder a los objetos dentro de esa pila.

Tipo de resultados. El tipo del resultado indica cómo debe ser tratado el resultado que se le regresará al cliente.

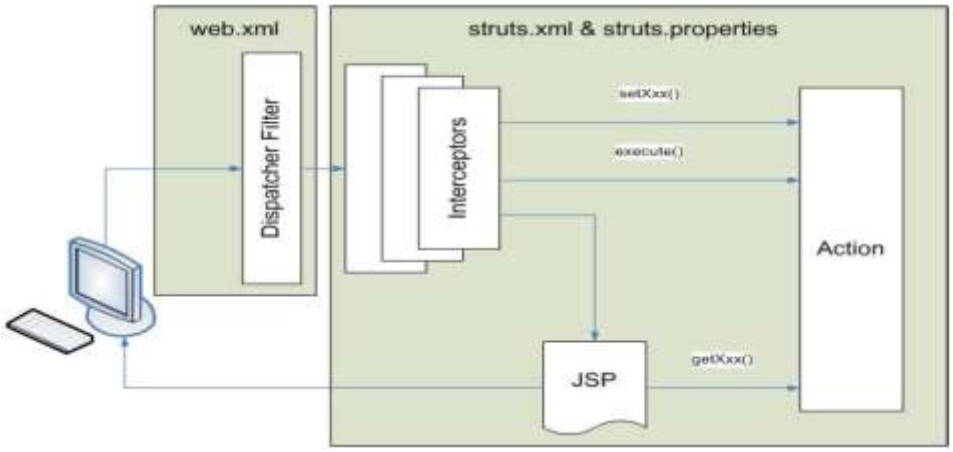


Figura 3.8. Struts2

3.2.2 Spring

La primera versión fue escrita por Rod Johnson, quien lo lanzó junto a la publicación de su libro "*Expert One-on-One J2EE Design and Development*", Spring 1.0 se publicó en marzo del 2004. Entre los años 2004/2005, Spring se popularizo como Framework de desarrollo para aplicaciones Java/J2EE. Spring da soporte a todas las capas de una aplicación.

La estructura de Spring se compone esencialmente de:

- **Contenedor de inversión de control (IoC).** Proporciona una forma consistente de configuración y administración de objetos Java usando la reflexión. El contenedor se encarga de gestionar los ciclos de vida de objetos de los objetos específicos: la creación de estos objetos, llamando a sus métodos de inicialización y configuración de estos objetos mediante el cableado de ellos juntos.
- **Un Framework AOP.** Es un paradigma de programación que tiene como objetivo aumentar la modularidad al permitir la separación de los temas transversales. AOP constituye una base para el desarrollo de software orientado a aspectos.
- **Una capa de abstracción de servicios:** Integración consistente con varios estándares y APIs populares.

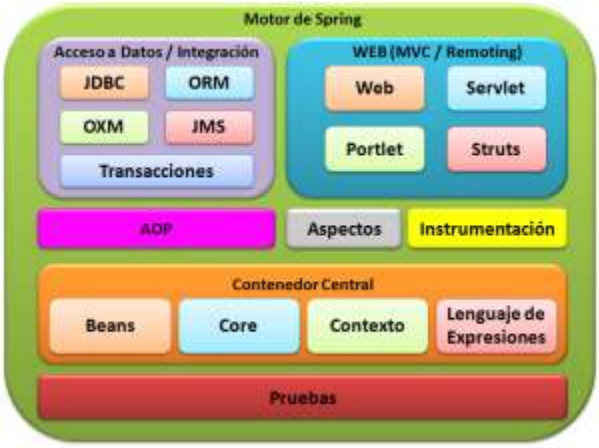


Figura 3.9. Spring

Spring se utiliza para sustituir algunos de los servicios que aportan los servidores de aplicaciones JEE, lo mejor de ello es la de "simplificar" el uso de las tecnologías JEE. Spring propone una estructura consistente para toda la aplicación, está abierta a la integración con múltiples estándares e implementaciones populares como pueden ser: Hibernate, JDO, TopLink, EJB, RMI, JNDI, JMS, Web Services, Struts, etc. Además permite aumentar nuestra productividad evitando al desarrollador la tarea de implementar tareas derivadas de la integración de los componentes de la aplicación.

3.2.3 Patrones de Diseño

Es una solución reutilizable general a un problema común que ocurre dentro de un contexto determinado en el diseño de software. Aceleran el desarrollo de Software y facilitan el mantenimiento. No fueron tal hasta principios de la década de 1990 cuando los patrones de diseño tuvieron un gran éxito en el mundo de la informática a partir de la publicación del libro "*Design Patterns*" escrito por el grupo Gang of Four (**GoF**) compuesto por Erich Gamma, Richard Helm, Ralph Johnson y John Vlisides, en el que se recogían 23 patrones de diseño comunes. En el proyecto se utilizarán diferentes patrones que corresponde con diferentes partes del mismo, por ejemplo:

Para la capa de presentación se está utilizando el patrón de diseño **MVC (Patrón Creacional)** integrando Struts2 en el, el cual consiste en tres partes diferenciadas:

1. **El Controlador (Controller).** Es el Servlet central que recibe peticiones, procesa las URL recibidas y delega el procesamiento a JavaBeans, este Servlet guarda el resultado de procesamiento realizado por JavaBeans en el contexto de la petición, la sesión o la aplicación, además transfiere el control a una JSP que lleva a cabo la presentación de resultados.
2. **El Modelo (Model).** Los JavaBeans (o EJBs para aplicaciones más escalables) juegan el rol de modelo, algunos beans ejecutan lógica, otros guardan datos, pero normalmente: el Servlet controlador invoca un método en un bean lógico y éste

devuelve un bean de datos, a partir de ahí el autor de la JSP tiene acceso a ese bean de datos.

3. **La Vista (View).** En la parte de la vista se ejecutan la JSPs, el Servlet Controlador transfiere el control a la JSP, después de haber guardado en un contexto el resultado en forma de un bean de datos, esa JSP usa *jsp:useBean* y *jsp:getProperty* para recuperar datos y formatear respuesta en una página HTML o XML.

En resumen, los beans o EJBs ejecutan la lógica de negocio y guardan los resultados en las JSPs, que estas proveen la información formateada a los servlets, que estos a su vez coordinan/controlan la ejecución de los beans y las JSPs.

Para la parte lógica se utiliza el patrón **Façade(Patrón Estructural)**, el cual es un patrón de diseño de software de uso común en la programación orientada a objetos. El nombre es por analogía con una fachada arquitectónica. Se aplica cuando se necesita proporcionar una interfaz simple para un subsistema complejo, o cuando se quiera estructurar varios subsistemas en capas, ya que las fachadas serían el punto de entrada a cada nivel.

La principal ventaja del patrón fachada consiste en que para modificar las clases de los subsistemas, sólo hay que realizar cambios en la interfaz/fachada, y los clientes pueden permanecer ajenos a ello. Además, y como se mencionó anteriormente, los clientes no necesitan conocer las clases que hay tras dicha interfaz.

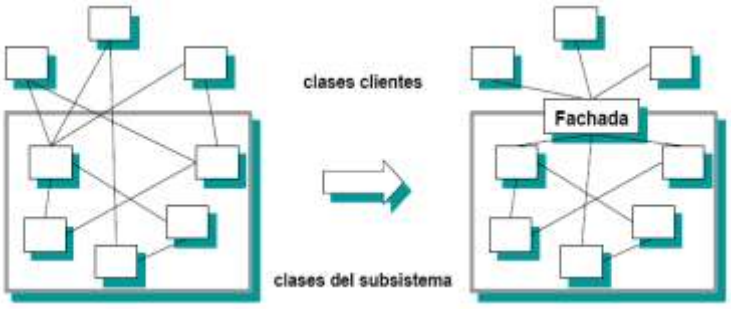


Figura 3.10. Façade

Otro escenario proclive para su aplicación surge de la necesidad de desacoplar un sistema de sus clientes y de otros subsistemas, haciéndolo más independiente, portable y reutilizable (esto es, reduciendo dependencias entre los subsistemas y los clientes).

Y para la parte de persistencia los patrones **DAO**. **DAO** es un objeto de acceso a datos, que proporciona una interfaz abstracta a algún tipo de base de datos u otro tipo de mecanismo de persistencia. Proporciona algunas operaciones de datos específicos sin exponer a los detalles de la base de datos. Cualquier objeto de negocio no requiere conocimiento directo del destino final de la información que manipula.

Los DAO se suelen utilizar en Java para aislar a una aplicación de la tecnología de persistencia Java subyacente, en nuestro caso, la tecnología que estamos usando es ser JDBC, pero podían ser cualquiera de estas: JDO, Enterprise JavaBeans, TopLink, EclipseLink, Hibernate, iBATIS, o cualquier otra tecnología de persistencia. Usando DAOs significa que la tecnología subyacente puede ser actualizada o cambiada sin cambiar otras partes de la aplicación.

3.3 HERRAMIENTAS

3.3.1 APACHE TOMCAT

En realidad son dos herramientas distintas, pero que todo el mundo los utiliza juntos ya que se necesitaban uno del otro, el Tomcat era un contenedor de Servlets mientas que el Apache era solamente un Servidor Web, ahora Tomcat también funciona como un servidor de aplicaciones por sí solo pero Apache es incapaz de ejecutar el contenido dinámico de algunas páginas y ahí es donde entra el Tomcat quien facilita la ejecución de esas aplicaciones ya sean Servlets o JSP. Ambos se pueden conectar por medio de conectores que permiten que el Apache redireccione las peticiones hacía Tomcat para que este se encargue de mostrar un sitio en particular.

Apache Tomcat fue desarrollado bajo el proyecto Jakarta en la Apache Software Foundation. Esta herramienta implementa las especificaciones de los Servlets y de JavaServer Pages (JSP) de Sun Microsystems. Nuestro proyecto está utilizando la versión 7.

3.3.2 MYSQL

El software MySQL proporciona un servidor de base de datos SQL (Structured Query Language) muy rápido, multi-threaded, multi usuario y robusto. El servidor MySQL está diseñado para entornos de producción críticos, con alta carga de trabajo así como para integrarse en software para ser distribuido.

Desde enero de 2008 formaba parte de Sun Microsystems pero en abril de 2009 Oracle Corporation se hizo cargo al adquirir la empresa con los derechos de MySQL como software libre en un esquema de licenciamiento dual. La versión que utilizamos en el proyecto es la 5.5.

3.3.3 JUNIT

JUnit es un conjunto de librerías, creadas por Erich Gamma y Kent Beck, se utilizan expresamente en programación para hacer pruebas unitarias de aplicaciones Java.

JUnit es un conjunto de clases que permite realizar ejecución de clases Java de manera controlada, para poder evaluar si el funcionamiento de cada uno de los métodos de la clase se comporta como se espera. Si la clase cumple con la especificación, entonces JUnit devolverá que el método de la clase pasó exitosamente la prueba; en caso de que el valor esperado sea diferente al que regresó el método durante la ejecución, JUnit devolverá un fallo en el método correspondiente. Se está utilizando la versión 4.

3.3.4 JMeter

Es una aplicación de escritorio de código abierto, una aplicación 100% de Java diseñado para cargar el comportamiento de pruebas funcionales y medir el rendimiento. Originalmente se diseñó para probar aplicaciones Web, pero se ha ampliado desde entonces a otras funciones de prueba.

Puede ser utilizado para probar el rendimiento tanto de los recursos estáticos y dinámicos, como los archivos estáticos, para el proyecto la se utiliza para probar algunos objetos de Java y para la bases de datos. JMeter puede ser utilizado para simular una carga pesada en un servidor, de red o un objeto para poner a prueba su resistencia o para analizar el rendimiento total en virtud de diferentes tipos de carga.

Además, JMeter puede ayudar a probar la aplicación de regresión por lo que le permite crear scripts de prueba, con afirmaciones para validar que la aplicación devuelve los resultados esperados. Para una máxima flexibilidad, JMeter permite crear estas afirmaciones utilizando expresiones regulares.

3.3.5 Métrica V3

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 propia 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).

Sus procesos principales son los siguientes:

- Planificación de sistemas de Información (PSI).
- Desarrollo de sistemas de Información (DSI). Debido a su complejidad, está a su vez dividido en cinco procesos:
  - Estudio de Viabilidad del sistema (EVS).
  - Análisis del sistema de Información (ASI).
  - Diseño del sistema de Información (DSI).
  - Construcción del sistema de Información (CSI).
  - Implantación y Aceptación del sistema (IAS).
- Mantenimiento de sistemas de Información (MSI).

Este documento está basado en la plantilla estándar hecha por la EUITIO (ahora EII) que está desarrollada usando la metodología METRICA V3, pero de forma adaptada, contemplando solo aquellos apartados que se han considerado más adecuados para un Trabajo fin de Máster.

3.3.6 UML

UML es una especificación de notación orientada a objetos. Se basa en las anteriores especificaciones Empezó como una consolidación del trabajo de BOOCH, RUMBAUGH y COAD-YOURDON creadores de tres de las metodologías orientadas a objetos más populares. Divide cada proyecto en un número de diagramas que representan las diferentes vistas del proyecto. Estos diagramas juntos son los que representa la arquitectura del proyecto.

En 1996, el Object Management Group (OMG), un pilar estándar para la comunidad del diseño orientado a objetos, publicó una petición con propósito de un metamodelo orientado a objetos de semántica y notación estándares. UML, en su versión 1.0, fue propuesto como una respuesta a esta petición en enero de 1997.

Se usarán para modelar el sistemas de software, UML ofrece varios diagramas divididos en grupos, en los cuales poder modelar sistemas, se utilizarán aquellos UML según sean necesarios, predominaran los diagramas de clases y de casos de uso.

- Diagramas de Estructura.

  Diagramas de Clases para modelar la estructura estática de las clases en el sistema.
  Diagramas de Componentes para modelar componentes.
  Diagrama de Estructura Compuesta que muestra la estructura interna de una clase y las colaboraciones que esta estructura hace posible.
  Diagrama de Despliegue para modelar la distribución del sistema.
  Diagramas de Objetos para modelar la estructura estática de los objetos en el sistema
  Diagrama de Paquetes que representa las dependencias entre los paquetes que forman un modelo.
- Diagramas de Comportamiento

  Diagramas de Actividad para modelar el comportamiento de los Casos de Uso, objetos u operaciones.
  Diagramas de Estado para modelar el comportamiento de los objetos en el sistema.
  Diagramas de Casos de Uso para modelar los procesos “business”.
- Diagramas de Interacción

  Diagrama de Comunicación moldea las interacciones entre objetos o partes en términos de mensajes secuenciados.
  Diagrama de Interacción es similar al de actividad, pero con la diferencia de que puede contener interacciones o diagramas de secuencia.
  Diagramas de Secuencia para modelar el paso de mensajes entre objetos.

3.3.7 iText

IText es una librería de código abierto que permite crear y manipular documentos PDF. Permite a los desarrolladores que buscan mejorar web -y otras aplicaciones utilizar está herramienta para la generación de documentos PDF dinámicos y / o su manipulación.

Fue escrita por Bruno Lowagie, Paulo Soares, y otros; está distribuida bajo la "Affero General Public License".

La utilización se produce porque los datos que se crean pueden ser escritos a un fichero PDF, aunque también se puede realizar desde un servlet a un navegador web. Es capaz de rellenar formularios, mover páginas de un PDF a otro, y un sinfín de cosas. Estas extensiones son a menudo mutuamente excluyentes. Una clase te permite rellenar en formularios, mientras una clase diferente e incompatible hace posible copiar páginas de un PDF a otro. La versión que se está utilizando es la 5.5.3.

3.3.8 JavaMail

La API de JavaMail ofrece clases que modelan un sistema electrónico. El paquete javax.mail define las clases que son comunes a todos los sistemas de correo. El paquete javax.mail.internet define clases que son específicos de los sistemas de correo basados en estándares de Internet, tales como MIME, SMTP, POP3 e IMAP. El API JavaMail incluye el paquete javax.mail y sub-paquetes.

JavaMail se está utilizando para el envío de correos electrónicos directamente desde la aplicación Java. Como por ejemplo de informar a los miembros de un proyecto de alguna noticia o evento relacionado con eso mismo proyecto o para el envío de cualquier otra información que el Manager o el Administrador consideren relevante informar. Se está utilizando la versión 1.5.1.

Capítulo 4. Planificación del Proyecto y Resumen de Presupuestos

Uno de los temas importantes a la hora de proyectar es el tiempo que va a llevar a cabo hacerlo, ya que todo corre alrededor de ello. Para ello se utiliza el programa denominado Microsoft Project, el cual es un software de administración de proyectos diseñado, desarrollado y comercializado por Microsoft, para dar soporte a la creación de proyectos en el desarrollo de planes, asignación de recursos a tareas, dar seguimiento al progreso, administrar presupuestos y analizar cargas de trabajo. Se eligió este software no libre, ya que es el más familiar con respecto a su uso, ya que se ha utilizado anteriormente, para otros proyectos.

4.1 Planificación Inicial

La duración del proyecto como se observará más adelante en el diagrama de Gantt será de unos 148.75 días, su fase inicial comenzó el 5 de mayo de 2014 y se tiene como fecha de finalización el 2 de febrero de 2015. Fecha escogida según calendario oficial del Máster para la presentación de la memoria del TFM.

El número de personas que trabajara en el proyecto y las cuales se dividirá el trabajo dependiendo de la capacidad y el rol de cada unió de ellos serán:

- 1 Director de Proyecto
- 1 Analista
- 1 Ing. de Diseño
- 1 Tester
- 1 Documentalista
- 1 Técnico
- Yo, Pablo González

Además de agregar como recursos a los trabajadores anteriormente citados estos serán asignados a sus diferentes tareas, además de otros recursos como son los materiales, todo ello se podrá ver en el archivo .mpp, que se encuentra en el proyecto.

La planificación se dividió en varias etapas, las cuales se desarrollaron del siguiente modo.

Se comenzó por un estudio inicial del proyecto, el cual contuvo unos primeros pasos como son el decálogo de redacciones de documentos académicos, problemas que podían surgir y sus posibles soluciones a esos problemas del proyecto, un planteamiento inicial para el desarrollo del proyecto, en el cual se diferenciarían las diferentes fases, las herramientas que se utilizarían y el sistema propiamente dicho.

Un apartado de búsqueda de información, donde recopilaríamos y almacenaríamos esa información durante la mayor parte del proyecto, donde se reservaría un tiempo también a leer y comprender dicha información.

Este mismo capítulo tiene su parte de tiempo en el cronograma, con la construcción de esta planificación y su presupuesto inicial. El cual ha tenido pequeños cambios tanto en el planteamiento como en el presupuesto final.

La administración del proyecto que se encarga de todos los temas del proyecto en sí, como son el análisis, el diseño o la implementación e instalación de los programas para la ejecución del proyecto.

La construcción de esta documentación, con la memoria, el sistema, las pruebas, los manuales, las conclusiones finales y sus ampliaciones, el presupuesto, las referencias bibliográficas, los apéndices y anexos. Después se reservo tiempo para la revisión de este mismo documento, así como tiempo para la subida y de ficheros y entrega de documentación a la administración.

Como última planificación sería la creación de la presentación, videos e instalación del equipo y demás imprevistos para la defensa del trabajo fin de Máster.

Id	Nombre de tarea	Duración	Comienzo	Fin	Predec
1	**1 Planificación TFM**	**148.75 días?**	**lun 05/05/14**	**lun 02/02/15**	
2	**1.1 Estudio de Carácter General**	**4.25 días**	**lun 05/05/14**	**dom 11/05/14**	
10	**1.2 Búsqueda de Información**	**138.75 días?**	**sáb 24/05/14**	**dom 01/02/15**	
13	1.3 Planificación	5 horas	lun 12/05/14	mié 14/05/14	9
14	1.4 Presupuesto	6 horas	mié 14/05/14	vie 16/05/14	13
15	1.5 Presupuesto y Planificación	0 días	sáb 15/11/14	sáb 15/11/14	
16	**1.6 Administración del Proyecto**	**93.63 días**	**vie 16/05/14**	**vie 14/11/14**	**14**
17	**1.6.1 Análisis**	**39.13 días**	**vie 16/05/14**	**sáb 19/07/14**	
18	1.6.1.1 Definición Sistema	16 días	vie 16/05/14	sáb 14/06/14	
19	1.6.1.2 Requisitos del Sistema	9 días	vie 20/06/14	sáb 05/07/14	
20	1.6.1.3 Identificación Subsistema	9 días	sáb 05/07/14	sáb 19/07/14	19
21	**1.6.2 Diseño**	**23.25 días**	**sáb 19/07/14**	**dom 21/09/14**	
22	1.6.2.1 Arquitectura del Sistema	19 días	sáb 19/07/14	lun 15/09/14	17
23	1.6.2.2 Diseño de Clases	8 días	mié 13/08/14	dom 21/09/14	
24	1.6.2.3 Interfaces	5 días	sáb 16/08/14	sáb 20/09/14	
25	**1.6.3 Implementación e Instalación**	**31.25 días**	**lun 22/09/14**	**vie 14/11/14**	
26	1.6.3.1 Código	22 días	lun 22/09/14	sáb 01/11/14	21
27	1.6.3.2 Pruebas	18 días	mié 15/10/14	vie 14/11/14	
28	1.6.3.3 Instalación	5 días	mié 05/11/14	vie 14/11/14	
29	**1.7 Documentación**	**47.38 días**	**sáb 15/11/14**	**sáb 31/01/15**	
30	**1.7.1 Memoria**	**3.13 días**	**sáb 15/11/14**	**mar 18/11/14**	
39	1.7.2 Sistema	13 días	mar 18/11/14	mié 10/12/14	30
40	1.7.3 Pruebas	9 días	mié 10/12/14	vie 26/12/14	39
41	1.7.4 Manuales	7 días	vie 26/12/14	dom 04/01/15	40
42	1.7.5 Conclusiones y Ampliaciones	2 días	dom 04/01/15	sáb 10/01/15	41
43	1.7.6 Presupuesto	2 días	sáb 10/01/15	dom 11/01/15	42
44	1.7.7 Referencias Bibliográficas	2 días	dom 11/01/15	vie 16/01/15	43
45	1.7.8 Apéndices	2 días	vie 16/01/15	dom 18/01/15	44
46	1.7.9 Anexos	2 días	dom 18/01/15	vie 23/01/15	45
47	1.7.10 Revisión	5 días	vie 23/01/15	sáb 31/01/15	46
48	1.7.11 Subida de Ficheros	2 horas	sáb 31/01/15	sáb 31/01/15	47
49	**1.8 Defensa**	**2 días**	**sáb 31/01/15**	**dom 01/02/15**	
50	1.8.1 Creación de la Presentación	16 horas	sáb 31/01/15	dom 01/02/15	29
51	1.9 Final de Proyecto	0 días	lun 02/02/15	lun 02/02/15	

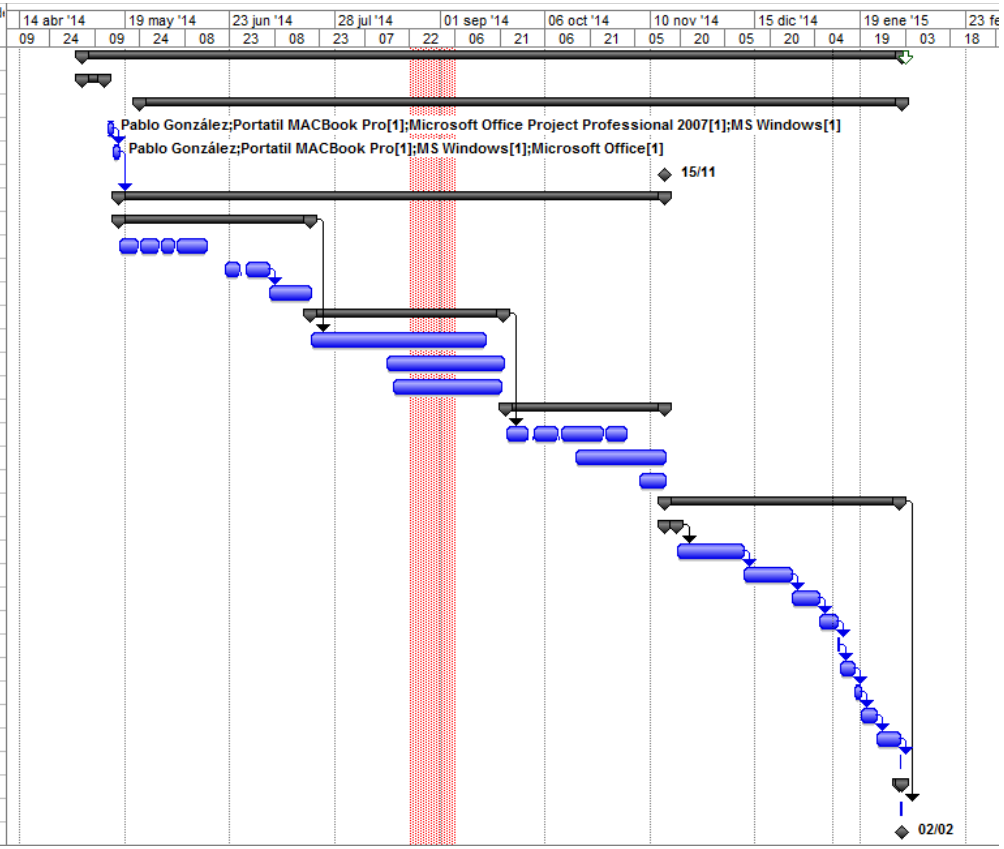


Figura 4.1. Planificación Inicial

4.2 Resumen del Presupuesto Inicial

Este es un resumen para el cliente, sobre el presupuesto de la ejecución del proyecto, de forma que pueda valorar si decide proseguir o no con ello.

Ítem	SubItem	Concepto	Cantidad	Precio Unitario	Factor de Amortización	TOTAL
1		**Materiales**				
		Hardware				**342,11 €**
		Software				**102,79 €**
		Comunicaciones				**141,48 €**
		Formación				**350,00 €**
		Mobiliario				**104,25 €**
2		**Desarrollo de la Aplicación**				
		Estudio Inicial				**237,50 €**
		Análisis				**1.390,00 €**
		Diseño				**744,00 €**
		Implementación				**572,00 €**
		Pruebas				**450,00 €**
		Instalación				**120,00 €**
		Documentación				**1.089,74 €**
3		**Otros**				**2.980,00 €**
					Subtotal	**8.623,87 €**
					Beneficio (21,00%)	**1.811,01 €**
					Total sin IVA	**10.434,88 €**
I.V.A. es un 21%					*Total con IVA*	**12.626,21 €**

Figura 4.2. Presupuesto Inicial

Capítulo 5. Análisis

Este apartado contendrá toda la especificación de requisitos y toda la documentación del análisis de la aplicación, a partir de la cual se elaborará posteriormente el diseño.

5.1 Definición del Sistema

5.1.1 Determinación del Alcance del Sistema

El sistema que se ha desarrollado se utilizará como una aplicación Web en principio para el uso en asignaturas del Grado y del Máster de la EII, con lo que su uso fundamental está destinado a los alumnos.

Como se ha descrito anteriormente, este sistema será usado como medio para la Gestión de Riesgos en Proyectos de Informática, contará entre las acciones a desarrollar:

1. La Planificación de la Gestión de riesgos
2. Identificación y Análisis de Riesgos
3. Planificación de la Respuesta a los Riesgos

Quedaría fuera del desarrollo, La Monitorización y el Control de los Riesgos, por ir más allá de lo que se planteo al principio de la ejecución del proyecto.

Aparte de lo que es el objetivo del sistema central, se contará también, con:

1. Sistema para la identificación de usuarios/roles
2. Sistema de sesiones para usuarios
3. Sistema para el envió de emails de información

Se contará con tres diferentes roles, Administrador, Manager y demás miembros.

El Administrador es quien crea los Managers y los proyectos, se comienza por estos porque son el recurso principal a los que va dirigidos (alumnos), si fuera una empresa real, los proyectos sería lo primero y a estos se les asignarían los recursos de la empresa que serían sus trabajadores, en nuestro caso yo lo veo como que tenemos a los alumnos y a estos hay que asignarles proyectos. El trabajo del Administrador terminará ahí.

Los Managers son los responsables de asignar nuevos miembros del equipo al proyecto, al igual que estos últimos, deberán rellenar su correspondiente gestión de riesgos con los objetivos planteados anteriormente.

Los miembros del equipo solo tendrán la opción para crear sus propias gestiones de riesgos individuales.

5.2 Requisitos del Sistema

5.2.1 Obtención de los Requisitos del Sistema

5.2.1.1 Requisitos Funcionales

Código	Nombre Requisito	Descripción del Requisito
Sistema		
R1.1	Atender Peticiones	El sistema deberá tener una entrada estándar formado por un Servicio Web que atenderá las solicitudes que le lleguen.
R1.2	Crear Interfaz Gráfica de Identificación	El sistema presentará una interfaz gráfica para poder loguearse los diferentes usuarios.
R1.2.1	Comprobar Usuarios	El sistema deberá comprobar si el nombre de usuario y la contraseña se encuentran almacenados en la BBDD.
R1.2.2	Diferenciar Roles/Usuarios	El sistema tendrá una serie de interceptores, que identificarán según el login que se introduzca, el tipo de usuario que se está introduciendo en sesión.
R1.3	Crear Sesiones Individuales	El sistema será capaz de crear y eliminar sesiones para cada usuario que se conecte.
R1.4	Crear Interfaces	Creará interfaces según el tipo de usuario que se encuentre en sesión, las cuales contendrán diversa información para la realización de las acciones de cada rol.
R1.4.1	Insertar Títulos e Imágenes en las Interfaces Gráficas de Usuarios	El sistema deberá presentar la aplicación con varias interfaces gráficas, en primer lugar la inicial desde donde se podrá navegar hacía las demás, cada una de ellas tendrá un título descriptivo y un logotipo común, cada una de ellas se podrá cerrar su sesión mediante una opción de salida.
R1.4.2	Interfaz Ayuda	El sistema contará con una sección para ayuda personalizada, diferente para cada usuario, donde explicará las acciones que este puede realizar.
R1.5	Crear Consultas	El sistema deberá encargarse de crear las consultas de exploración de los datos almacenados en la base de datos para obtener distintas informaciones que la aplicación necesite.
R1.5.1	Almacenar Consultas	El sistema deberá encargarse de almacenar la información en la base de datos cuando la aplicación lo requiera.
R1.5.2	Consultar Consultas	El sistema deberá poder cargar la información que se encuentra en la base de datos cuando la aplicación lo requiera.
R1.6	Crear Variables de Sesión	El sistema deberá poder crear variables en sesión para alojar objetos que puedan estar a disposición en otras partes de la aplicación.

R1.6.1	Almacenar Variables de Sesión	El sistema deberá poder crear variables en sesión para alojar objetos que puedan estar a disposición en otras partes de la aplicación.
R1.6.2	Utilizar Variables de Sesión	El sistema deberá poder utilizar las variables en sesión que puedan estar a disposición de la aplicación.
R1.7	Emitir Mensajes	Emitirá mensajes OK, la acción se realice correctamente, en caso de KO de la acción mostrará mensajes de error.
Administrador		
R1.8	Rellenar Formularios Acceso Proyecto/Managers	Deberá rellenar los formularios de registro tanto de proyectos como de managers, que será el primer y segundo paso para la creación del gestor de riesgos de proyectos.
R1.8.1	Crear Nuevos Manager/Proyectos	Después de recibir la información de los formularios, creará con esa información nuevos usuarios tipo managers o proyectos, que serán almacenados para su posterior utilización por parte de la aplicación.
R1.8.1.1	Enviar Correo	El Administrador cuando crea los diferentes managers y proyectos, se envía un email a los interesados haciéndoles saber sus datos de acceso y que proyecto le corresponde.
R1.8.1.2	Almacenar Managers y Proyectos	Una vez creado los diferentes managers o proyectos, estos son enviados por el Servicio Web, hacía el servidor, donde son almacenados en la base de datos.
R1.9	Recuperar Información Proyectos/Managers	Se recuperan los registros tanto de los proyectos, como de los managers que estén en la base de datos para poder modificarlos.
R1.10	Recuperar Datos Administrador	Se recupera y muestra, la información de acceso y personal del Administrador, que se encuentra almacenada en la base datos, este puede modificarla.
R1.8 **R1.9** **R1.10**	Enviar/Recibir Solicitudes	Deberá poder enviar/recibir los datos al sistema para que este los inserte/obtenga adecuadamente de la base de datos.
R1.8 **R1.9** **R1.10**	Convertir Datos	Al recuperar la información, se deberá plasmar esa información que contiene en diferentes tablas donde el Administrador podrá cambiar los datos que desee.
Managers		
R1.11	Rellenar Formularios Acceso Miembros	Deberá rellenar los formularios de registro de los miembros del proyecto, que será el tercer paso para la creación del gestor de riesgos de proyectos.
R1.11.1	Crear Nuevos Miembros	Después de recibir la información de los formularios, creará con esa información nuevos usuarios tipo miembros, que serán almacenados para su posterior utilización por parte de la aplicación.

R1.11.1.1	Enviar Correo	El Administrador cuando crea los diferentes miembros, se envía un email a los interesados haciéndoles saber sus datos de acceso y que proyecto le corresponde.
R1.11.1.2	Almacenar Miembros	Una vez creado los diferentes miembros, estos son enviados por el Servicio Web, hacía el servidor, donde son almacenados en la base de datos.
R1.12	Recuperar Información Proyecto/Miembros	Se recuperan los registros tanto de los proyectos, como de los miembros del proyecto que es Manager.
R1.13	Recuperar Datos Manager	Se recupera y muestra, la información de acceso y personal del Manager, que se encuentra almacenada en la base datos, este puede modificarla.
R1.14	Crear Gestión de Riesgos	El Manager podrá crear los diferentes pasos de la gestión de riesgos de proyectos.
R1.14.1	Almacenar Datos para la Gestión de Riesgos	El sistema será capaz de almacenar aquellos datos de los pasos referentes a la gestión de riesgos.
R1.14.2	Recuperar Datos para la Gestión de Riesgos	El Manager será capaz de recuperar si existen, en la base de datos, aquellos datos de los pasos referentes a la gestión de riesgos.
R1.14.3	Elegir Versión Planes de Riesgo	El Manager dará la opción al Manager de que elija con qué tipo de versión quiere trabajar de la que se encuentran almacenadas en la base de datos.
R1.14.4	Crear Documentos PDF	El sistema será capaz de crear documentos PDF con los datos almacenados en la base de daros sobre la gestión de riesgos.
R1.12 **R1.13** **R1.15**	Convertir Datos	Al recuperar la información, se deberá plasmar esa información que contiene en diferentes tablas donde el Administrador podrá cambiar los datos que desee.
Miembros		
R1.16	Recuperar Datos Miembro	Se recupera y muestra, la información de acceso y personal del Manager, que se encuentra almacenada en la base datos, este puede modificarla.
R1.17	Crear Gestión de Riesgos	El Manager podrá crear los diferentes pasos de la gestión de riesgos de proyectos.
R1.17.1	Almacenar Datos para la Gestión de Riesgos	El sistema será capaz de almacenar aquellos datos de los pasos referentes a la gestión de riesgos.
R1.17.2	Recuperar Datos para la Gestión de Riesgos	El Manager será capaz de recuperar si existen, en la base de datos, aquellos datos de los pasos referentes a la gestión de riesgos.
R1.17.3	Elegir Versión Planes de Riesgo	El Manager dará la opción al Manager de que elija con qué tipo de versión quiere trabajar de la que se encuentran almacenadas en la base de datos.
R1.17.4	Crear Documentos PDF	El sistema será capaz de crear documentos PDF con los datos almacenados en la base de daros sobre la gestión de riesgos.
R1.16	Convertir Datos	Al recuperar la información, se deberá plasmar esa

R1.17		información que contiene en diferentes tablas donde el Administrador podrá cambiar los datos que desee.

5.2.1.2 Requisitos Tecnológicos

Código	Nombre Requisito	Descripción del Requisito
R1.1	Elección de lenguaje	Como plataforma de programación de la aplicación será J2EE.
R1.2	Frameworks	Los Frameworks utilizados son Struts2 con Spring.
R1.3	Desarrollo abierto	El desarrollo será de código abierto u open source, por tanto el código y la documentación deberán estar comentados de la mejor manera posible para facilitar la tarea a los posibles desarrolladores posteriores.
R1.4	Los datos almacenados en MySQL	Deberá almacenar los datos en una base de datos MySQL versión 5.5.
R1.5	Sistema operativo	El sistema operativo deberá ser de la familia de Windows o Linux.

5.2.1.3 Requisitos Seguridad

Código	Nombre Requisito	Descripción del Requisito
R1.1	Encriptado de datos	Los datos sensibles, como pueden ser contraseñas o correos electrónicos se encuentran encriptados en la base de datos.

5.2.1.4 Requisitos de Tiempo de Respuesta

Código	Nombre Requisito	Descripción del Requisito
R1.1	Tiempo de ejecución	El programa deberá almacenar y recuperar los registros en un tiempo razonable, según el tamaño de las consultas.
R1.2	Fin de sesión	El tiempo máximo de expiración de sesión son 30 minutos.

5.2.2 Identificación de Actores del Sistema

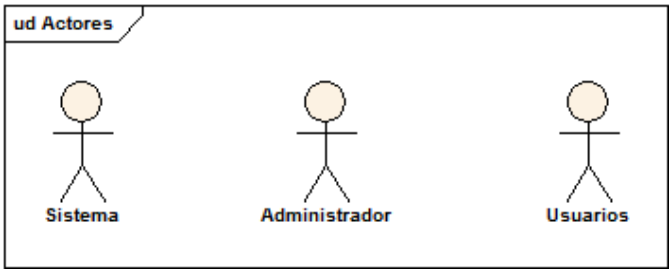


Figura 5.1. Actores del Sistema

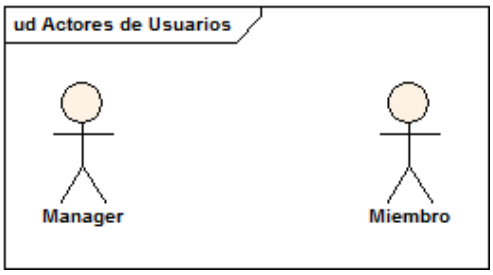


Figura 5.2. Actores de los Usuarios

5.2.3 Especificación de Casos de Uso

5.2.3.1 Sistema

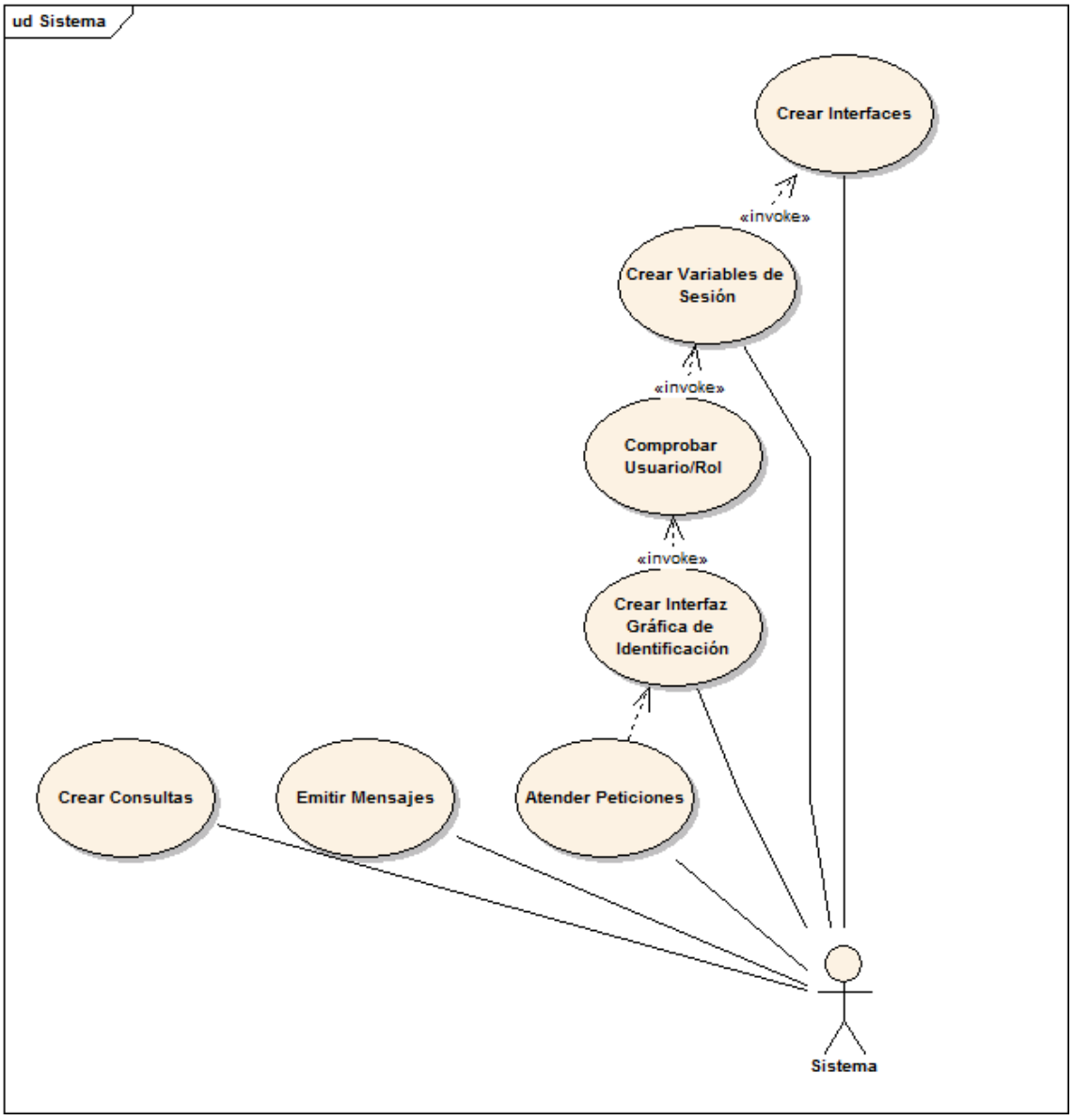


Figura 5.3. Casos Uso Sistema

Nombre del Caso de Uso
Atender Peticiones
Descripción
El sistema debe atender las diferentes solicitudes que le llegan por medio del Servicio Web y darles respuesta.

Nombre del Caso de Uso
Crear Interfaz Gráfica de Identificación
Pre-condición
Sistema debe estar en CU: Atender Peticiones
Descripción
El sistema después de recibir una petición creará la interfaz de acceso al gestor, con una pantalla de login.

Nombre del Caso de Uso
Comprobar Usuario/Rol
Pre-condición
Sistema debe estar en CU: Crear Interfaz Gráfica de Identificación
Descripción
Después de introducir los datos en la pantalla de login, se comprueba en BBDD, si existe ese usuario y se determina en el sistema con los datos recibidos su rol.

Nombre del Caso de Uso
Crear Variables de Sesión
Pre-condición
Sistema debe estar en CU: Comprobar Usuario/Rol
Descripción
Una vez comprobado el usuario y determinado el rol, se agrega esos datos en una variable de sesión. Caso General: Los datos que legan por medio de otros casos de uso son almacenados en diferentes variables de sesión.

Nombre del Caso de Uso
Crear Interfaces
Pre-condición
Sistema debe estar en CU: Crear Variables de Sesión
Descripción
El sistema crea interfaces dependiendo de los datos de sesión, donde determina por el rol cual debe presentar.

Nombre del Caso de Uso
Emitir Mensajes
Descripción
El sistema emite mensajes dependiendo de si la acción ha sido OK, responderá con mensajes correctos, si ha sido KO, será un mensaje de error.

Nombre del Caso de Uso
Crear Consultas
Descripción
El sistema deberá encargarse de crear las consultas de exploración a la base de datos para obtener aquellos datos que se le pida.

5.2.3.2 Administrador

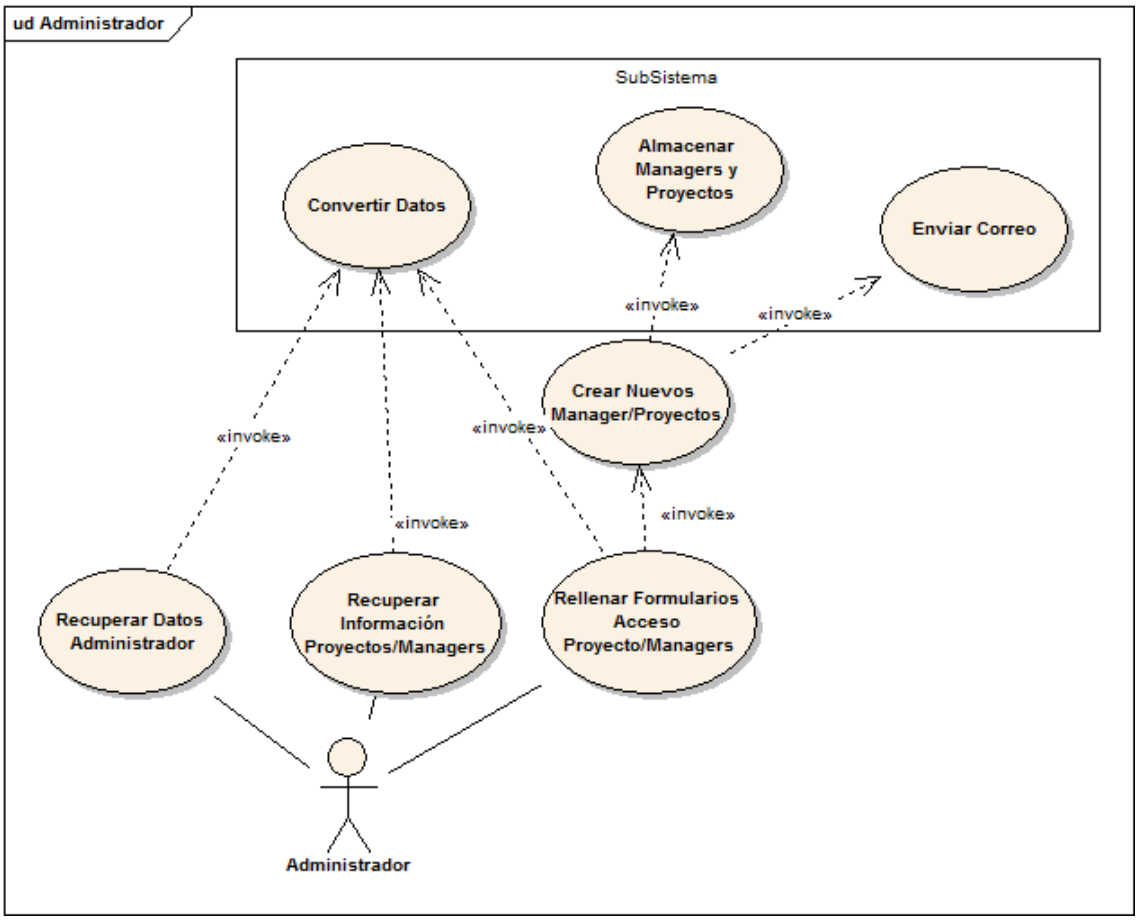


Figura 5.4. Casos de Uso Administrador

Nombre del Caso de Uso
Recuperar Datos Administrador
Descripción
El Administrador podrá recuperar los datos que se encuentran almacenados en la aplicación, para verlos y modificarlos.

Nombre del Caso de Uso
Recuperar Información Proyectos/Managers
Descripción
El Administrador podrá ver todos los proyectos y managers que se encuentran almacenados en la BBDD, y podrá hacer modificaciones sobre ellos.

Nombre del Caso de Uso
Rellenar Formularios Acceso Proyecto/Managers
Descripción
El Administrador es el encargado de rellenar los formularios para la creación de nuevos managers y proyectos.

Nombre del Caso de Uso
Convertir Datos
Pre-condición
Sistema debe estar en CU: Recuperar Datos Administrador o Recuperar Información Proyectos/Managers o Rellenar Formularios Acceso Proyecto/Managers
Descripción
Los datos recuperados de la BBDD, son tratados para que sean presentados al Administrador, en su interfaz.

Nombre del Caso de Uso
Crear Nuevos Manager/Proyectos
Pre-condición
Sistema debe estar en CU: Rellenar Formularios Acceso Proyecto/Managers
Descripción
El Administrador es el encargado de creador los managers y proyectos de la aplicación.

Nombre del Caso de Uso
Almacenar Managers y Proyectos
Pre-condición

Sistema debe estar en CU: Crear Nuevos Manager/Proyectos
Descripción
El subsistema que es llamado desde el Administrador, se encarga de almacenar los diferentes datos que son suministrados por este.

Nombre del Caso de Uso
Enviar Correo
Pre-condición
Sistema debe estar en CU: Crear Nuevos Manager/Proyectos
Descripción
Una vez que el Administrador crea los managers o los proyectos, el subsistema invocado por el Administrador envía los correos a las personas correspondientes.

5.2.3.3 Manager

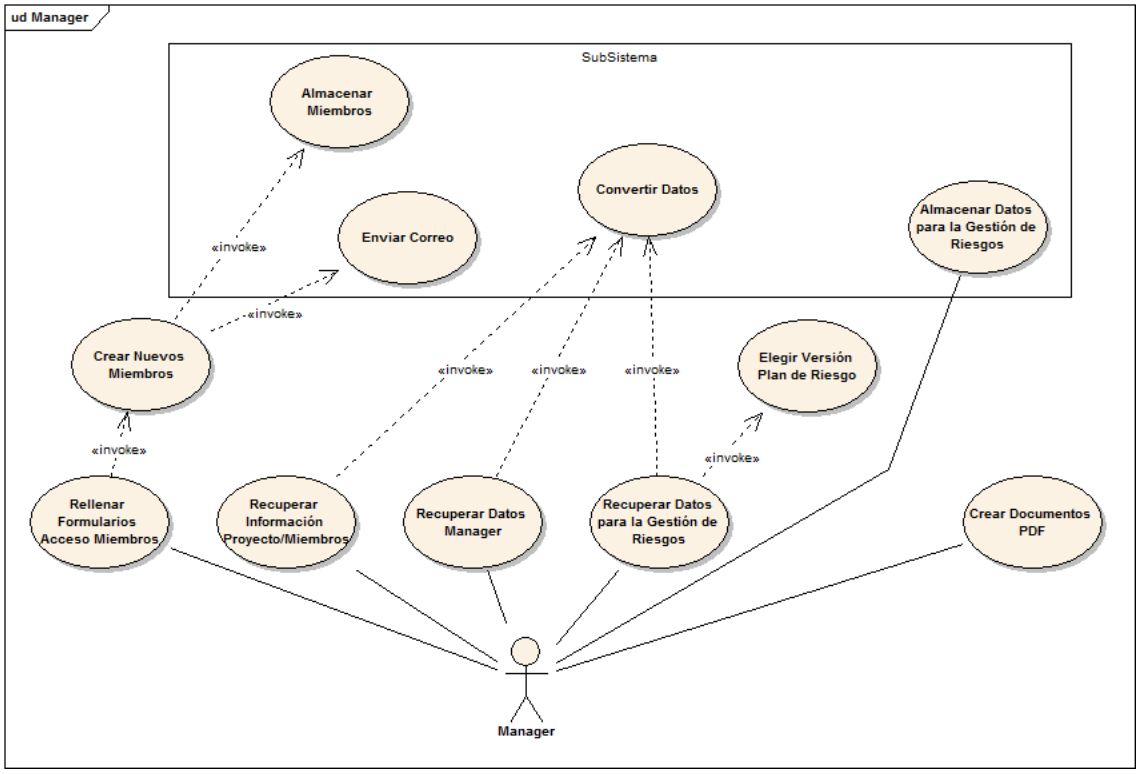


Figura 5.5. Casos de Uso Manager

Nombre del Caso de Uso
Rellenar Formularios Acceso Miembros
Descripción
El Manager será el encargado de rellenar el formulario de los datos de acceso de los demás miembros del proyecto.

Nombre del Caso de Uso
Recuperar Información Proyecto/Miembros
Descripción
El Manager podrá recuperar la información sobre el proyecto que tiene asignado así como los miembros de este.

Nombre del Caso de Uso
Recuperar Datos Manager
Descripción
El Manager podrá recuperar sus propios datos y también podrá modificarlos.

Nombre del Caso de Uso
Recuperar Datos para la Gestión de Riesgos
Descripción
El Manager podrá recuperar si existen los datos de su gestión de riesgos del proyecto.

Nombre del Caso de Uso
Almacenar Datos para la Gestión de Riesgos
Descripción
El Manager a través de un subsistema podrá almacenar los datos introducidos en el objetivo del proyecto que es el gestor de riesgos.

Nombre del Caso de Uso
Crear Documentos PDF
Descripción
El Manager una vez terminados algunos de los pasos del gestor de riesgos podrá crear documentos PDF de ese paso en concreto.

Nombre del Caso de Uso
Crear Nuevos Miembros
Pre-condición
Sistema debe estar en CU: Rellenar Formularios Acceso Miembros
Descripción
El Manager será el encargado de crear los nuevos miembros de su proyecto.

Nombre del Caso de Uso
Almacenar Miembros
Pre-condición
Sistema debe estar en CU: Crear Nuevos Miembros
Descripción
El Manager por medio de un subsistema podrá almacenar los datos introducidos de los miembros.

Nombre del Caso de Uso
Enviar Correo
Pre-condición
Sistema debe estar en CU: Crear Nuevos Miembros
Descripción
El Manager una vez creado los diferentes usuarios miembro, un subsistema enviara un correo a dichos miembros, con sus datos de acceso y el proyecto asignado.

Nombre del Caso de Uso
Convertir Datos
Pre-condición
Sistema debe estar en CU: Recuperar Datos para la Gestión de Riesgos o Recuperar Datos Manager o Recuperar Información Proyecto/Miembros
Descripción
Un subsistema invocado por el Manager será el encargado de recibir los datos de este y proceder a realizar con ellos la acción designada con la consiguiente respuesta a la interfaz del Manager.

Nombre del Caso de Uso
Elegir Versión Plan de Riesgo
Pre-condición
Sistema debe estar en CU: Recuperar Datos para la Gestión de Riesgos
Descripción
El Manager podrá elegir que versión del plan de riesgos quiere utilizar, si es que existiera más de una almacenada.

5.2.3.4 Miembro

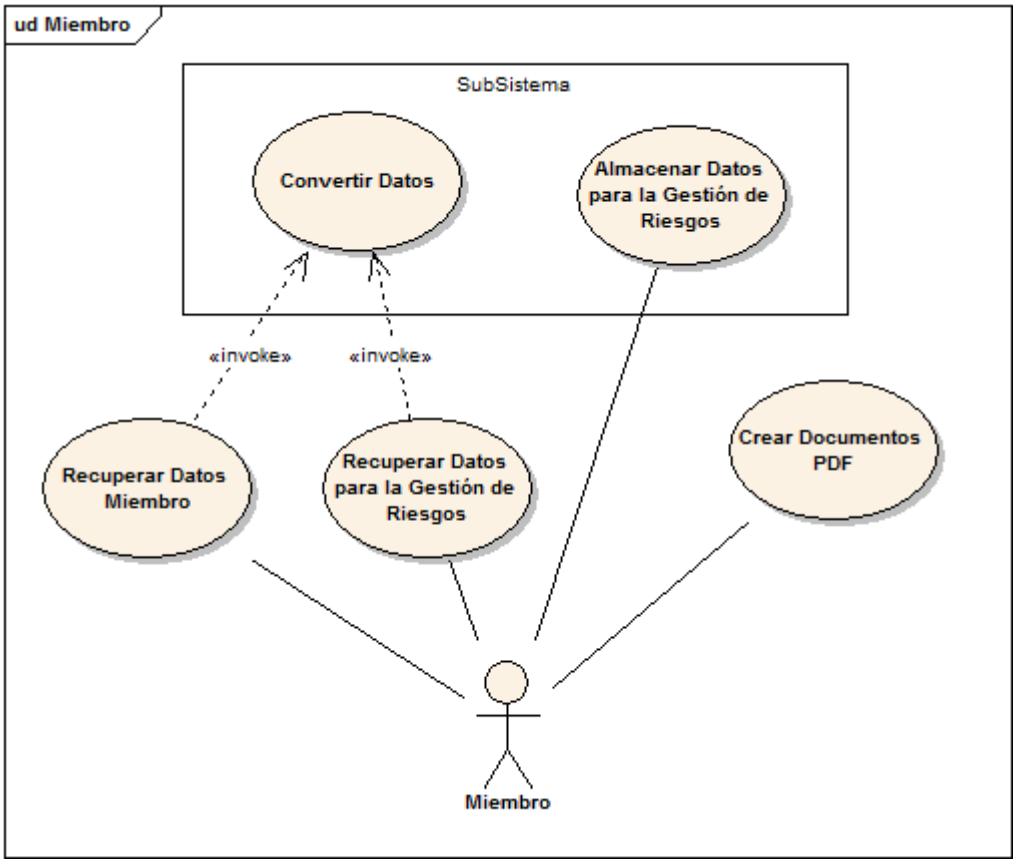


Figura 5.6. Casos de Uso Miembro

Nombre del Caso de Uso
Recuperar Datos Miembro
Descripción
El miembro del proyecto podrá, ver sus datos almacenados en la aplicación y modificarlos en consecuencia.

Nombre del Caso de Uso
Recuperar Datos para la Gestión de Riesgos
Descripción
El miembro podrá recuperar los datos sobre su gestión de riesgo, siempre que tengan datos almacenados en la BBDD.

Nombre del Caso de Uso
Crear Documentos PDF
Descripción
El miembro una vez terminados algunos de los pasos del gestor de riesgos podrá crear documentos PDF de esos pasos en concreto.

Nombre del Caso de Uso
Almacenar Datos para la Gestión de Riesgos
Descripción
El miembro por medio de un subsistema podrá almacenar los datos introducidos de los miembros.

Nombre del Caso de Uso
Convertir Datos
Pre-condición
Sistema debe estar en CU: Recuperar Datos Miembro o Recuperar Datos para la Gestión de Riesgos
Descripción
Un subsistema invocado por el miembro será el encargado de recibir los datos de este y proceder a realizar con ellos la acción designada con la consiguiente respuesta a la interfaz del Manager.

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

5.3.1.1 Subsistema de Identificación

En este primer subsistema, es donde se realiza la búsqueda y comprobación de la entrada de usuarios a la aplicación y dependiendo su rol se le asigna una interfaz (**login**), todo esto se consigue por medio de los filtros de Struts2 con el empleo de interceptores que provienen de Xwork framework y que identifican el usuario y actúan en consecuencia:

- **Administrador.** Si el rol del usuario identificado es el Administrador, este tendrá una interfaz propia con opciones diferentes a los demás roles, como son la creación de proyectos y managers, se le mostrará también cuantos proyectos y managers existen y la posibilidad de modificarlos y añadir nuevos proyectos a managers.
- **Manager.** Si el rol identificado es el de Manager, tendrá también su propia interfaz, donde podrá crear nuevos miembros al proyecto que tenga asignado, se le mostrará también que tipo de proyecto tiene y cuáles son sus integrantes. Además tendrá la opción como el miembro de generar su Gestión de Riesgos.
- **Miembro.** Si el rol es este, su cometido es el de generar la Gestión de Riesgos del Proyecto, al igual que los managers.

Todos tienen en común su propia zona, sobre información de datos de acceso a la aplicación y una ayuda personalizada según el rol.

También una vez que todos los usuarios se encuentren logueados dentro de la aplicación, está tendrá su propio método de **unlogin**, donde se eliminarán las variables que se encuentren en sesión y saldrá de la aplicación.

5.3.1.2 Subsistema de Creación y Validación

Este subsistema, será utilizado por todos los roles de la aplicación, en distintas acciones.

Administrador. Es utilizado a la hora de rellenar los formularios de la creación de los managers y de los proyectos, se ayuda del Framework Spring utilizando el **BeanFactory** que es una implementación del patrón factoría que permite crear Beans, los cuales van con la información de los formularios dependiendo del objeto Bean, estos son validados previamente, por el validador del Framework de Struts2, los campos introducidos se validan a través de un XML que contiene algunas reglas definidas que deben pasar los campos introducidos por él, una vez validados son enviados por las diferentes interfaces hasta su almacenamiento en la BBDD.

Manager. El Manager también tiene el formulario para la creación de los miembros que sigue el mismo ejemplo que el del Administrador. Además contiene también otros formularios como son los de los Planes de Riesgo, Identificación y Análisis de Riesgos y la Respuesta a los Riesgos, todos ellos de la Gestión de Riesgos del Proyecto. Algunos de los datos son válidos por las reglas, convertidos en su Bean correspondiente y transportados hasta la BBDD.

Miembro. Para este tipo de usuario, solo utiliza este subsistema para la creación del Plan de Gestión, de la Identificación/Análisis y la Respuesta de los Riesgos. Se validan aquellos datos que tiene reglas, se encapsula en su Bean y se transporta hasta la BBDD.

Además como paso común a todos ellos, cuando se muestra o se actualiza la información de cada uno de los roles en su espacio personal, donde aparecen los datos de acceso a la aplicación sigue, el proceso de validación, de los nuevos datos introducidos, con las reglas para cada campo.

5.3.1.3 Subsistema Internacionalización

La aplicación cuenta con un subsistema para internacionalizar el texto por medio de archivos porperties que son llamados desde la clase de internacionalización, lo que hace básicamente es recuperar un mensaje de esos ficheros de recursos.

Utilizamos para ello el Global Resource Bundle, se configura en el struts.xml

```
<constant name="struts.custom.i18n.resources" value="global" />
```

Para que se tenga acceso a el en todo momento, luego referenciamos las etiquetas desde las JSPs o el código del modo:

```
<h2><s:text name=" welcome.title"/></h2>
```

Donde en nuestro archivo properties, definimos dicho atributo name, podemos tener tantos properties como idiomas utilicemos. Dichos archivos tienen que seguir la nomenclatura siguiente:

global_en.properties → Inglés; global_es.properties → Castellano

1. **global.** Nombre que se le da en la constante en el archivo de struts.xml
2. **_en.** Es el idioma con el que se va asociar el contenido del texto de su interior
3. **.properties.** Es la extensión de la archivo.

El idioma en que se mostraran los textos se basa en el atributo LOCALE que viaja en la request. Dependiendo del LOCALE, se cogerá un fichero u otro de recursos.

```
welcome.title=Life Cycle´s System Management of the Risks in IT
Projects
```

5.3.1.4 Subsistema Creación PDF

Este subsistema, se basa en la construcción de documentos PDF, gracias a la librería iText.

Solo lo tienen disponibles aquellos usuarios que tengan, como objetivo la realización del Proyecto de Gestión de Riesgos, que para este caso será para los roles de Manager y Miembro. Una vez que hayan terminado con algunos de los pasos anteriores, como son Plan de Gestión de Riesgos, Identificación/Análisis de Riesgos o Respuesta de a los Riesgos, se habilita la última opción que es la creación de la documentación de los informes de los datos que haya, según los pasos anteriores. Dichos documentos son la prueba de la realización de los estudios de los riesgos y el entregable de los miembros y del Manager, de los posibles riesgos que conllevaría ese proyecto informático.

5.3.1.5 Subsistema Envío Correo

Este subsistema está pensado para el envío por correo de información relevante sobre eventos que se produzcan en los proyectos. Por ejemplo cuando se crea una nueva cuenta Manager por parte del Administrador, el sistema según los datos facilitados por el Administrador, contacta por medio de un correo electrónico con ese Manager, enviándole los datos de acceso a la aplicación.

Cuando se crea el proyecto por parte del Administrador, se manda otro correo al Manager para que este sepa que ya tiene creado el proyecto y puede comenzar a introducir los nuevos Miembros a el.

Una vez que el Manager es creado y este tiene asignado un proyecto, puede comenzar a añadir nuevos miembros, cada vez que añada estos nuevos miembros, también a estos se les mandará un correo para que sepan sus datos de conexión a la aplicación.

Está pensada la estructura para que en futuras ampliaciones se puedan utilizar diversos sistemas de comunicación como podrían ser el envío de más y diferentes mensajes de eventos del proyecto y no solo por correo, sino también por los sistemas como SMS, mensajería interna, WhatsApp o diferentes redes sociales, Facebook, Twitter…

5.3.1.6 Subsistema Operaciones en BBDD

El subsistema de persistencia, será el encargado de todo lo que tenga que ver con el modo de almacenar los datos en la base de datos, ya sea guardar, borrar, insertar o recuperar datos, se realice siguiendo las órdenes del subsistema de comunicación.

Desde las diferentes interfaces de los modelos de servicio le llegan los datos y allí se escoge el servicio que han solicitado desde el Action, ese método lleva la información desde allí a este subsistema. El subsistema de operaciones según el servicio que le han solicitado, crea las conexiones con la base de datos, en nuestro caso se utiliza JDBC, se valida que ese usuario existe en la base de datos con los permios necesarios y por medio de las sentencias de SQL que tiene asignadas cada método que utiliza un PreparedStatement, se inserta, se borra, se selecciona o se actualiza, según el servicio escogido.

5.3.2 Descripción de los Interfaces entre Subsistemas

Para la comunicación de todas las partes del proyecto, se ha utilizado el modelo MVC junto a Struts2. El cual consta de lo siguiente:

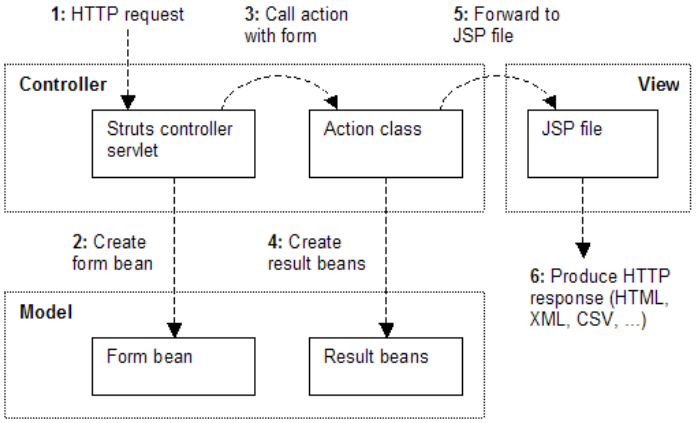


Figura 5.7. MVC con Struts2

El Controlador que en nuestro caso está implementado por un **dispatch server filter**, construido como una pila de interceptores, recibe las peticiones, así procesa las URLs recibidas y delega su procesamiento a los **JavaBeans**, este guarda el resultado de procesamiento realizado por los JavaBeans en el contexto de la petición, la sesión o la aplicación, dependiendo de lo que estemos usando, transfiere luego el control a la JSP que lleva a cabo la presentación de resultados.

La parte del Modelo utilizamos los **Actions** que se asocia con la **request HTTP** invocada por el usuario. El **Action** necesita acceder a los valores desde la request como desde los datos del form. Struts2 se basa en el concepto de la encapsulación de los datos por medio de JavaBeans, diferentes según el modelo que estemos utilizando, algunos de ellos ejecutan lógica y normalmente otros guardan los datos, ya sea por haberlos recogido de la BBDD para mostrarlos o al crearlos desde las JSPs.

El Action antes de retornar un resultado, a veces, necesita procesar la ejecución, necesita acceder a diferentes objetos como pueden ser – **business objects**, **data access objects** u otros

recursos. Con Struts2 se emplea la técnica llamada **dependency injection** es implementada por el constructor **injection**, **interface injection** y **setter injection**, esto significa que para tener objetos disponibles para un action, se necesita tener solamente un setter, además hay objetos que no maneja el framework de Spring, como es:

El objeto **HttpServletRequest**. Estos son manejados por la combinación de **setter injection** e **interface injection**. Con este objeto se comunica el Action con la vista, esto se puede realizar por la implementación de un interface "**aware**", aunque también utilizamos en algunos casos las tag libraries o **JSTL**, cuyo acceso a los datos es mucho más fácil. Ambos métodos hacen directamente accesibles las propiedades del Action una vez este sea añadido al **Value Stack**. Para esto es necesario crear los getters sobre el action que permitan el acceso a los objetos que se necesitan para acceder.

La parte de la vista es ejecutado por JSPs, el Servlet Controlador transfiere control al JSP después de haber guardado en un contexto el resultado en forma de un Bean de datos, esta JSP usa **jsp:useBean** y **jsp:getProperty** para recuperar los datos y formatearlos en una respuesta en HTML.

Aparte tenemos estructurada la aplicación por medio del **modelo de Brown de ncapas**, presentación, negocio y persistencia, la comunicación entre cada capa la estamos realizando por medio de la aplicación del patrón **Façade**, donde en cada separación de capa se comunica una con otra por medio de una interfaz, la cual lleva en su interior los diferentes métodos que se necesitan en la aplicación de los **execute** de los Actions y que requieren una veces llegar hasta la ejecución de dicho Action a la capa de negocio y otras veces se requieren para almacenar o recuperar valores en la capa de persistencia.

5.4 Diagrama de Clases Preliminar del Análisis

5.4.1 Diagrama de Clases

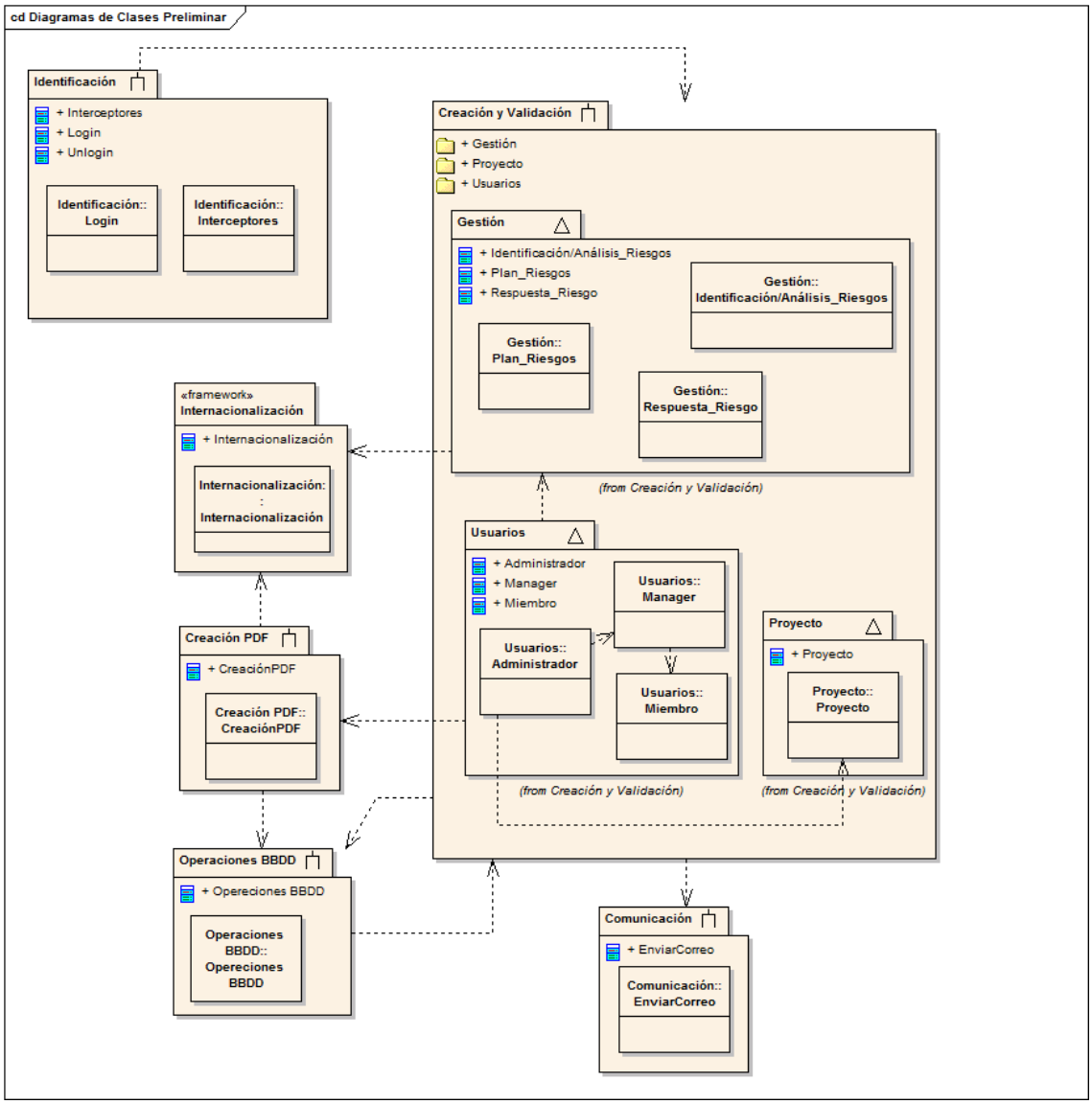


Figura 5.8. Diagrama de Clases Preliminar

5.4.2 Descripción de las Clases

5.4.2.1 Subsistema Identificación

Nombre de la Clase
Login
Descripción
Es la pantalla inicial donde los usuarios tienen que loguearse, si ese usuario existe, dependiendo del rol que tenga se le enviara a un pantalla o a otra.
Responsabilidades
Comprobar que la introducción del usuario y de la contraseña se corresponde con algunos de los actores que se encuentran almacenados en la base de datos, como usuarios de la aplicación.
Atributos Propuestos
usuario: Cadena donde se guardara el nombre de usuario **pass:** Cadena donde se guardara el password **rol:** Cadena con el rol que desempeña ese usuario **lenguaje:** Cadena donde se guardara el idioma del usuario **request:** Objeto HttpServletRequest que nos proporciona acceso a objetos de sesión **userService:** Atributo para comunicarnos con la interfaz de usuario de la capa de negocio
Métodos Propuestos
login: Método de logueo de los diferentes usuarios de la aplicación comprobándose que existan en la base de datos. **unlogin**: Método para salir de sesión en la aplicación y limpiar las variables de la sesión. **setServletRequest:** Setter que establece la solicitud al objeto HTTP en las clases de la aplicación **getUserService:** Getter para establecer la interfaz para los servicios aplicados al modelo de usuarios **setUserService:** Setter para establecer la interfaz para los servicios aplicados al modelo de usuarios

Nombre de la Clase
Interceptores
Descripción
Clase para implementar los interceptores, los cuales actuarán como un filtro HTTP o un Proxy, proveerán un camino para suplir el pre-processing y post-processing a través del action
Responsabilidades
Hay que asegurarse al crear los usuarios que estos tienen definido un rol.
Atributos Propuestos
user: Objeto del modelo de tipo Usuario, donde se alojan los datos de los usuarios **inv:** Objeto ActionInvocation que nos proporciona acceso al ActionContext, donde recuperar de la sesión el usuario
Métodos Propuestos
loginInterceptor: Método que dependiendo del rol que tenga el usuario recuperado de la sesión, este será transportado a su pantalla correspondiente **sesionInterceptor**: Método que cuando la sesión de la aplicación halla caducado se pondrán las variables a null, para que vuelva al inicio que es la pantalla de login

5.4.2.2 Subsistema Creación y Validación

Nombre de la Clase
Plan_Riesgo
Descripción
Es la clase donde se crea el objeto bean de la planificación de los riesgos
Responsabilidades
Debe de ser un usuario existente en BBDD y que su rol sea de Manager o Miembro
Atributos Propuestos
request: Objeto HttpServletRequest que nos proporciona acceso a objetos de sesión **infoService:** Atributo para comunicarnos con la interfaz de gestión de la capa de negocio **application:** Mapa para almacenar objetos que puedan utilizarse en otra parte de la aplicación **user:** Objeto del modelo de tipo Usuario, donde se alojan los datos de los usuarios **info:** Objeto del modelo de tipo Info donde se alojan los datos de los planes de riesgo
Métodos Propuestos
load: Método que carga los datos introducidos sobre el plan de gestión de riesgos en la BBDD **save**: Método que salva los datos introducidos sobre el plan de gestión de riesgos en la BBDD **setServletRequest:** Setter que establece la solicitud al objeto HTTP en las clases de la aplicación **getInfoService:** Getter para establecer la interfaz para los servicios aplicados al modelo de info que se utiliza para la información de los riesgos **setInfoService:** Setter para establecer la interfaz para los servicios aplicados al modelo de info que se utiliza para la información de los riesgos **setApplication:** Setter que establece el mapa de propiedades de la aplicación.

Nombre de la Clase
Iden_Analisis_Riesgo
Descripción
Es la clase donde se crea el objeto bean de la identificación/análisis de los riesgos
Responsabilidades
Debe de ser un usuario existente en BBDD, que su rol sea de Manager o Miembro y tener completado el plan de riesgos.
Atributos Propuestos
request: Objeto HttpServletRequest que nos proporciona acceso a objetos de sesión **infoService:** Atributo para comunicarnos con la interfaz de gestión de la capa de negocio **application:** Mapa para almacenar objetos que puedan utilizarse en otra parte de la aplicación **user:** Objeto del modelo de tipo Usuario, donde se alojan los datos de los usuarios **info:** Objeto del modelo de tipo Info donde se alojan los datos de la identificación y análisis de riesgos
Métodos Propuestos
load: Método que carga los datos introducidos sobre la identificación y análisis de riesgos en la BBDD **save**: Método que salva los datos introducidos sobre la identificación y análisis de riesgos en la BBDD **setServletRequest:** Setter que establece la solicitud al objeto HTTP en las clases de la aplicación **getInfoService:** Getter para establecer la interfaz para los servicios aplicados al modelo de info que se utiliza para la información de los riesgos

setInfoService: Setter para establecer la interfaz para los servicios aplicados al modelo de info que se utiliza para la información de los riesgos **setApplication:** Setter que establece el mapa de propiedades de la aplicación.

Nombre de la Clase
Respuesta_Riesgo
Descripción
Es la clase donde se crea el objeto bean de la respuesta a los riesgos
Responsabilidades
Debe de ser un usuario existente en BBDD, que su rol sea de Manager o Miembro, tener completado el plan de riesgos y la identificación y análisis de riesgos
Atributos Propuestos
request: Objeto HttpServletRequest que nos proporciona acceso a objetos de sesión **infoService:** Atributo para comunicarnos con la interfaz de gestión de la capa de negocio **application:** Mapa para almacenar objetos que puedan utilizarse en otra parte de la aplicación **user:** Objeto del modelo de tipo Usuario, donde se alojan los datos de los usuarios **info:** Objeto del modelo de tipo Info donde se alojan los datos de la respuesta a los riesgos
Métodos Propuestos
load: Método que carga los datos introducidos sobre la respuesta a los riesgos en la BBDD **save**: Método que salva los datos introducidos sobre la respuesta a los riesgos en la BBDD **setServletRequest:** Setter que establece la solicitud al objeto HTTP en las clases de la aplicación **getInfoService:** Getter para establecer la interfaz para los servicios aplicados al modelo de info que se utiliza para la información de los riesgos **setInfoService:** Setter para establecer la interfaz para los servicios aplicados al modelo de info que se utiliza para la información de los riesgos **setApplication:** Setter que establece el mapa de propiedades de la aplicación.

Nombre de la Clase
Administrador
Descripción
Una vez introducido y comprobado que el usuario y la contraseña corresponde a algún usuario de la BBDD y que este tiene el rol de Administrador, se le presentara una interfaz gráfica con las opciones propias para el Administrador
Responsabilidades
Las responsabilidades del Administrador son: 1. Crear Managers 2. Crear Proyectos para esos managers
Atributos Propuestos
login: Cadena con el login del Administrador **email:** Cadena con el correo electrónico **password:** Cadena con la contraseña de acceso **language:** Idioma del Administrador **admin:** Booleano para identificar a los usuarios con rol administradores
Métodos Propuestos
setter y getter de los atributos

Nombre de la Clase
Manager
Descripción
Una vez introducido y comprobado que el usuario y la contraseña corresponde a algún usuario de la BBDD y que este tiene el rol de Manager, se le presentara una interfaz gráfica con las opciones propias para el Manager
Responsabilidades
Las responsabilidades del Manager son: 1. Crear Miembros para el proyecto 2. Crear Gestión de Riesgos • Planes de Gestión de Riesgos • Identificación y Análisis de Riesgos • Respuesta a los Riesgos • Informes cada uno de los pasos anteriores
Atributos Propuestos
login: Cadena con el login del Manager **email:** Cadena con el correo electrónico **password:** Cadena con la contraseña de acceso **language:** Idioma del Manager **Manager:** Booleano para identificar a los usuarios con rol managers **idProyecto:** Long con el identificador del proyecto que se le ha asignado
Métodos Propuestos
setter y getter de los atributos

Nombre de la Clase
Miembro
Descripción
Una vez introducido y comprobado que el usuario y la contraseña corresponde a algún usuario de la BBDD y que este tiene el rol de miembro, se le presentara una interfaz gráfica con las opciones propias para el miembro
Responsabilidades
1. Crear Gestión de Riesgos • Planes de Gestión de Riesgos • Identificación y Análisis de Riesgos • Respuesta a los Riesgos • Informes cada uno de los pasos anteriores
Atributos Propuestos
login: Cadena con el login del miembro **email:** Cadena con el correo electrónico **password:** Cadena con la contraseña de acceso **language:** Idioma del miembro **miembro:** Booleano para identificar a los usuarios con rol miembros **idProyecto:** Long con el identificador del proyecto que se le ha asignado
Métodos Propuestos
setter y getter de los atributos

Nombre de la Clase
Proyecto
Descripción
Los proyectos son creados por el Administrador y este los asigne a un Manager para que los organice, creando a su vez miembros para ese proyecto, cada proyecto será el objetivo de la gestión de riesgos
Responsabilidades
Debe de existir un Manager creado previamente por el Administrador para que se le asigne al proyecto
Atributos Propuestos
id: Long con el número de identificación del proyecto **nombre:** Cadena con el nombre del proyecto **Manager:** Cadena con el Manager que lleva a cabo dicho proyecto **email:** Cadena con el email del Manager para infórmale de eventos que sucedan en el proyecto **fecha:** Cadena con la fecha de creación del proyecto **paso:** Integer con el número del paso por el que va el proyecto
Métodos Propuestos
setter y getter de los atributos

5.4.2.3 Subsistema Internacionalización

Nombre de la Clase
Internacionalizar
Descripción
Está clase se utilizará para comprobar tanto en los correos, como en la creación de PDF, como en otros documentos, el idioma que el usuario ha elegido y se encuentra en la BBDD de datos almacenado, para presentarle los datos en ese idioma
Responsabilidades
Debe existir documentos a los que escribir con el idioma del usuario que tenga preestablecido, al igual que debe existir ese usuario con uno de los dos idiomas que se contemplan
Atributos Propuestos
userDataService: Atributo para comunicarnos con la interfaz de usuarios de la capa de persistencia **properties:** Objeto Properties donde se encuentran las diferentes cadenas de texto en ambos idiomas **user:** Objeto del modelo de tipo Usuario, donde se alojan los datos de los usuarios ***ENGLISH*:** Cadena Final para la identificación de los usuarios con idioma en inglés ***SPANISH*:** Cadena Final para la identificación de los usuarios con idioma en castellano
Métodos Propuestos
getDocumento: Getter para obtener el documento que internacionalizar **setDocumento:** Setter para establecer el documento que internacionalizar **getUser:** Getter para obtener el usuario **setUser:** Setter para establecer el usuario **verIdioma:** Método para recoger el idioma del usuario que quiere la internacionalización **getUserDataService:** Getter de la interfaz entre el negocio y la persistencia de datos de usuarios **setUserDataService:** Setter de la interfaz entre el negocio y la persistencia de datos de usuarios

5.4.2.4 Subsistema Creación PDF

Nombre de la Clase
CrearPDF
Descripción
Clase para la creación de los documentos pdf según los datos facilitados y extraídos desde la BBDD de la aplicación
Responsabilidades
Deben existir documentos con datos para la creación de los PDF
Atributos Propuestos
nProyecto: Cadena con el nombre del proyecto **fecha:** Cadena con la fecha de realización del proyecto **lenguaje:** Cadena con el idioma que se desea el informe **version:** Cadena con la versión del proyecto **autor:** Cadena con el nombre del autor del informe **titulo:** Cadena con el título que se le quiere dar al documento PDF **info:** Objeto con los datos recogidos de la BBDD con la información de los planes de riesgo **iden:** Objeto con los datos recogidos de la BBDD con la información de los riegos identificados y analizados **resp:** Objeto con los datos recogidos de la BBDD con la información de la respuesta a los riegos **file:** Cadena con el nombre que se le quiere dar al fichero PDF que se va a crear
Métodos Propuestos
createPdf: Método para la creación de los pdf **createPath**: Método para crear el path donde se almacenará el PDF

5.4.2.5 Subsistema Enviar Correo

Nombre de la Clase
EnviarCorreo
Descripción
Clase con la cual se informara a al Administrador, managers y miembros de la aplicación de sus datos de acceso, además, a los Manager y miembros también se les mandará datos de eventos relacionados con información sobre el proyecto
Responsabilidades
Debe existir los usuarios, sus cuentas de correo deben ser correctas, no puede estar caído el servidor de correo y debe de haberse producido un evento de creación de nuevos usuarios eventos en el proyecto para que se produzca el envió de correo
Atributos Propuestos
session: Objeto de tipo Session con los parámetros iniciales para el envío de correos **lenguaje:** Cadena con el idioma que se desea el informe **userDataService:** Atributo para comunicarnos con la interfaz de usuarios de la capa de persistencia
Métodos Propuestos
init: Método de carga de parámetros iniciales para el envío de correos **send:** Método de la interfaz entre presentación y negocio para el envió de correo **makeJavamail:** Método para la creación de Javamail **getUserDataService:** Getter de la interfaz entre el negocio y la persistencia de datos de usuarios

setUserDataService: Setter de la interfaz entre el negocio y la persistencia de datos de usuarios

5.4.2.6 Subsistema Operaciones BBDD

Nombre de la Clase
OperacionesBBDD
Descripción
Clase donde se encuentran las operaciones de inserción, actualización y borrado de los diferentes métodos utilizados para de la aplicación para con sus datos
Responsabilidades
Debe existir una BBDD, con una conexión, un usuario existente en esa BBDD, donde poder acceder y realizar las acciones oportunas con los permisos necesarios dentro de ella
Atributos Propuestos
ps: Objeto PreparedStatement **rs:** Objeto ResultSet **con:** Objeto Connection
Métodos Propuestos
getProyecto: Getter para obtener el proyecto de la base de datos **setProyecto:** Setter para establecer el proyecto de la base de datos **getUser:** Getter para obtener el usuario de la base de datos **setUser:** Setter para establecer el usuario de la base de datos **getRiesgo**: Método que obtiene los datos introducidos desde la aplicación de la gestión de riegos en la BBDD **setRiesgo:** Método que establece los datos introducidos desde la aplicación de la gestión de riegos en la BBDD **getConnection:** Getter que devuelve la conexión de la BBDD

5.5 Análisis de Casos de Uso y Escenarios

5.5.1 Caso de Uso 1

Identificarse en el Sistema	
Precondiciones	El usuario debe de existir previamente dentro de la base de datos
Poscondiciones	El usuario debe estar validado y con un rol asignado
Actores	Iniciado por un usuario de cualquier tipo de la aplicación, finalizado por un usuario con el rol asociado a la información de logon introducido
Descripción	El usuario 1. Ejecuta o se dirige a la dirección Web de la aplicación 2. El sistema muestra la interfaz gráfica de logon 3. El usuario introduce su nombre y su clave de usuario 4. El sistema valida la información introducida y la compara con la que tiene almacenada en la base de datos 5. El usuario entra correctamente en el sistema 6. Fin
Variaciones (escenarios secundarios)	• **Escenario Alternativo 1**: El identificador para poder entrar en el sistema es erróneo. o Volver al paso 2 del escenario principal, borrando la información del usuario y la clave. • **Escenario Alternativo 2**: Algunos de los campos introducidos para poder entrar en el sistema es erróneo. o Volver al paso 3 del escenario principal, se le informa en la misma pantalla de que alguno de sus campos no cumple con la validación
Excepciones	• **La clave o usuario para conectarse es incorrecta:** Puede ser correcta pero que se le haya cambiado el nombre de la clase y el Servicio Web no pueda comprar claves, o que de verdad no coincidan o Notificar el error por medio de un mensaje • **La conexión con el servidor no está disponible:** No se puede conectar y llevar los datos a la base de datos o Notificar el error por medio de un mensaje • **La base de datos no está disponible**: No se pueden comparar el nombre de usuario ni la clave. o Notificar el error por medio de un mensaje • **La base de datos produce un error en la consulta**: Error la consulta contiene algún tipo de error o Notificar un error asociado al problema encontrado
Notas	En caso de errores, por lo que sea, se debe mostrar un mensaje de error, añadiendo en la medida de lo posible, una causa del mismo

5.5.2 Caso de Uso 1.2

Recuperar Contraseña	
Precondiciones	El usuario debe de existir previamente dentro de la base de datos
Poscondiciones	
Actores	Iniciado por un usuario de cualquier tipo de la aplicación, finalizado por un usuario existente en la base de datos
Descripción	El usuario 1. Ejecuta o se dirige a la dirección Web de la aplicación 2. El sistema muestra la interfaz gráfica de logon 3. El usuario elige la opción de contraseña olvidada 4. El usuario introduce el correo con el que se registro 5. El usuario es encontrado en la base de datos 6. Se recuperan los datos y se envían al correo facilitado 7. Se informa al usuario de que se han enviado a su correo los datos 8. Fin
Variaciones (escenarios secundarios)	• **Escenario Alternativo 1**: El correo para poder recuperar los datos del usuario es erróneo. o Volver al paso 4 del escenario principal, borrando la información del usuario y la clave. • **Escenario Alternativo 2**: Algunos de los campos introducidos para poder entrar en el sistema es erróneo. o Volver al paso 4 del escenario principal, se le informa en la misma pantalla de que alguno de sus campos no cumple con la validación
Excepciones	• **La clave o usuario para conectarse es incorrecta:** Puede ser correcta pero que se le haya cambiado el nombre de la clase y el Servicio Web no pueda comprar claves, o que de verdad no coincidan o Notificar el error por medio de un mensaje • **La conexión con el servidor no está disponible:** No se puede conectar y llevar los datos a la base de datos o Notificar el error por medio de un mensaje • **La base de datos no está disponible**: No se pueden comparar el nombre de usuario ni la clave. o Notificar el error por medio de un mensaje • **La base de datos produce un error en la consulta**: Error la consulta contiene algún tipo de error o Notificar un error asociado al problema encontrado
Notas	En caso de errores, por lo que sea, se debe mostrar un mensaje de error, añadiendo en la medida de lo posible, una causa del mismo

5.5.3 Caso de Uso 2

Crear/Almacenar Manager y Proyecto	
Precondiciones	El Administrador único que puede crear los Manager y proyectos debe estar validado con el rol perteneciente a la interfaz gráfica donde se encuentra
Poscondiciones	Debe existir un nuevo registro con identificador único en el sistema de almacenamiento, uno por cada creación que se haga
Actores	Iniciado y terminado por el Administrador
Descripción	El Administrador: 1. Accederá a la interfaz gráfica del rol validado 2. Creará nuevo Manager a través del formulario habilitado para ello 3. Rellenará la información necesaria para confeccionar el registro del nuevo Manager 4. Enviará el formulario al sistema para que este lo almacene como un nuevo usuario para la aplicación 5. Enviar Correo al Manager creado, con los datos de acceso 6. Se le presenta nueva ventana para asignarle un proyecto al Manager creado 7. Rellenará la información que aún no está, necesaria para confeccionar el registro del nuevo proyecto 8. Enviará el formulario al sistema para que este lo almacene como un nuevo proyecto para la aplicación 9. Enviar Correo al Manager con los datos del proyecto creado 10. Recibirá un mensaje del éxito de la operación 11. Fin
Variaciones (escenarios secundarios)	• **Escenario Alternativo 1**: En el paso 4 y paso 8, si el validador encuentra algún campo mal introducido o vacio se lo hace saber, no dejará enviarlo a la BBDD hasta que lo corrija. • **Escenario Alternativo 2**: En el paso 6, si no se quiere crear un proyecto, se puede dejar al Manager sin asignarle ninguno, se puede salir y rellenar el proyecto más tarde. ○ Ir al paso último número 11 • **Escenario Alternativo 3**: En el paso 5 y paso 9, si se produce algún error en el envío del correo por el servidor de correo preestablecido, se le comunica al usuario que no se ha enviado el correo, pero el proceso continúa.

Excepciones	• **La clave o usuario para conectarse es incorrecta:** Puede ser correcta pero que se le haya cambiado el nombre de la clase y el Servicio Web no pueda comprar claves, o que de verdad no coincidan
	  ○ Notificar el error por medio de un mensaje
	• **La conexión con el servidor no está disponible:** No se puede conectar y llevar los datos a la base de datos
	  ○ Notificar el error por medio de un mensaje
	• **La base de datos no está disponible**: No se pueden comparar el nombre de usuario ni la clave.
	  ○ Notificar el error por medio de un mensaje
	• **La base de datos produce un error en la consulta**: Error la

	consulta contiene algún tipo de error o Notificar un error asociado al problema encontrado
Notas	En caso de errores, por lo que sea, se debe mostrar un mensaje de error, añadiendo en la medida de lo posible, una causa del mismo

5.5.4 Caso de Uso 2.1

Visualizar Proyectos/Managers	
Precondiciones	El Administrador es el único que puede ver todos los Manager y proyectos que han sido creados, mientras que el Manager, solo podrá ver aquellos proyectos a los que se le ha asignado y los miembros que se encuentran en el. Ambos usuarios deben estar validados con el rol perteneciente a cada una de la interfaz gráfica donde se encuentra. Debe existir al menos un proyecto un Manager o un miembro, para poder visualizar algo.
Poscondiciones	
Actores	Iniciado y terminado por el Administrador o Manager
Descripción	El Administrador/Manager: 1. Accederá a la interfaz gráfica del rol validado 2. Acceder a la pestaña para visualizar los datos 3. El sistema realizara peticiones sobre los proyectos/managers/miembros que se encuentren en la base de datos 4. El servidor le responderá recogiendo los datos 5. El sistema mostrará en la sección de vista por medio de unas tablas en la JSP los datos El Administrador: 6. Podrá escoger en eliminar los proyecto o managers o modificarlos 7. Recibirá un mensaje del éxito de la operación 8. Fin
Variaciones (escenarios secundarios)	• **Escenario Alternativo 1**: En el paso 5, no se encuentran datos. o Se termina ir paso 8 del escenario principal ambos usuarios • **Escenario Alternativo 2**: En el paso 6, se mostrara una nueva pantalla para el Administrador si este ha decidido modificar los datos de algún proyecto o Manager o Si todo va bien se mostrará un mensaje de éxito sino un mensaje de error, ir paso 8 del escenario
Excepciones	• **La clave o usuario para conectarse es incorrecta:** Puede ser correcta pero que se le haya cambiado el nombre de la clase y el Servicio Web no pueda comprar claves, o que de verdad no coincidan o Notificar el error por medio de un mensaje • **La conexión con el servidor no está disponible:** No se puede conectar y llevar los datos a la base de datos o Notificar el error por medio de un mensaje • **La base de datos no está disponible**: No se pueden comparar el nombre de usuario ni la clave.

	○ Notificar el error por medio de un mensaje • **La base de datos produce un error en la consulta**: Error la consulta contiene algún tipo de error ○ Notificar un error asociado al problema encontrado
Notas	En caso de errores, por lo que sea, se debe mostrar un mensaje de error, añadiendo en la medida de lo posible, una causa del mismo

5.5.5 Caso de Uso 3

Crear/Almacenar Miembro	
Precondiciones	El Manager es el único que puede crear los miembros, debe estar validado con el rol perteneciente a la interfaz gráfica donde se encuentra
Poscondiciones	Debe existir un nuevo registro con identificador único en el sistema de almacenamiento, uno por cada creación de miembro que se haga
Actores	Iniciado y terminado por el Manager
Descripción	El Manager: 1. Accederá a la interfaz gráfica del rol validado 2. Creará nuevo miembro a través del formulario habilitado para ello 3. Rellenará la información necesaria para confeccionar el registro del nuevo miembro 4. Enviará el formulario al sistema para que este lo almacene como un nuevo usuario para la aplicación 5. Enviar Correo al miembro creado, con los datos de acceso 6. Recibirá un mensaje del éxito de la operación 7. Fin
Variaciones (escenarios secundarios)	• **Escenario Alternativo 1**: En el paso 4, si el validador encuentra algún campo mal introducido o vacio se lo hace saber, no dejará enviarlo a la BBDD hasta que lo corrija. • **Escenario Alternativo 3**: En el paso 5, si se produce algún error en el envío del correo por el servidor de correo preestablecido, se le comunica al usuario que no se ha enviado el correo, pero el proceso continúa.
Excepciones	• **La clave o usuario para conectarse es incorrecta:** Puede ser correcta pero que se le haya cambiado el nombre de la clase y el Servicio Web no pueda comprar claves, o que de verdad no coincidan ○ Notificar el error por medio de un mensaje • **La conexión con el servidor no está disponible:** No se puede conectar y llevar los datos a la base de datos ○ Notificar el error por medio de un mensaje • **La base de datos no está disponible**: No se pueden comparar el nombre de usuario ni la clave. ○ Notificar el error por medio de un mensaje • **La base de datos produce un error en la consulta**: Error la consulta contiene algún tipo de error ○ Notificar un error asociado al problema encontrado
Notas	En caso de errores, por lo que sea, se debe mostrar un mensaje de error, añadiendo en la medida de lo posible, una causa del mismo

5.5.6 Caso de Uso 4

Obtención y Visualización de Datos para la Gestión de Riesgos	
Precondiciones	El sistema por medio de los usuarios Manager /miembro, debe haber creado algún objeto y haber enviado la petición al servidor
Poscondiciones	Al finalizar tiene que haber un registro en subsistema de operaciones de la base de datos
Actores	Iniciado y terminado por el sistema
Descripción	El sistema: 1. La aplicación del servidor le llega un petición de obtención de datos con información de la gestión de riesgos, desde el sistema del usuario identificado 2. Una vez que le llega la petición del sistema este utiliza las diferentes interfaces de la aplicación y envía la petición al sistema de operaciones de la base de datos 3. Se escoge la operación que se va a realizar en la base de datos 4. Se abre una conexión a la base de datos 5. Se ejecutan las consultas pertinentes sobre la base de datos 6. La base de datos responde, devolviendo los objetos que son requeridos por el sistema 7. El sistema recoge los datos y los lleva devuelta a la capa de presentación, donde por medio de JSPs, muestra al usuario la petición de datos que solicito 8. Fin
Variaciones (escenarios secundarios)	• **Escenario Alternativo 1**: Datos vacios o No se muestra ningún dato en la JSP o Finalizar proceso
Excepciones	• **La clave y usuario para conectarse al servidor es incorrecta:** Puede ser correcta pero que se le haya cambiado el nombre de la clase y el Servicio Web no pueda comprar claves, o que de verdad no coincidan o Notificar el error por medio de un mensaje • **La base de datos no está disponible**: No se pueden comparar el nombre de usuario ni la clave. o Notificar el error por medio de un mensaje • **La base de datos produce un error en la consulta**: Error la consulta contiene algún tipo de error o Notificar un error asociado al problema encontrado
Notas	En caso de errores, por lo que sea, se debe mostrar un mensaje de error, añadiendo en la medida de lo posible, una causa del mismo

5.5.7 Caso de Uso 4.1

Crear/Almacenar Gestión de Riesgos	
Precondiciones	El usuario deberá ser Manager o miembro y debe estar validado con el rol perteneciente a la interfaz gráfica donde se encuentra
Poscondiciones	Debe existir un nuevo registro con identificador único en el sistema de almacenamiento
Actores	Iniciado y terminado por el Manager/miembro
Descripción	El Manager/miembro: 1. Accederá a la interfaz gráfica del rol validado 2. Se dirigirá a la diferentes opciones para la creación de la gestión de riesgos 3. Completará los diferentes campos para la gestión 4. Según vaya rellenando pulsara en cada uno de los botones para almacenar los datos introducidos 5. Estos datos se almacenarán temporalmente en sesión para utilizarlos en pasos posteriores o en futuras visitas 6. La aplicación del servidor le llega un petición para establecer los nuevos datos con información de la gestión de riesgos, desde el sistema del usuario identificado 7. Una vez que le llega la petición del sistema este utiliza las diferentes interfaces de la aplicación y envía la petición al sistema de operaciones de la base de datos 8. Se escoge la operación que se va a realizar en la base de datos 9. Se abre una conexión a la base de datos 10. Se ejecutan las consultas pertinentes sobre la base de datos 11. La base de datos responde, devuelve el control al sistema 12. El sistema comprueba que no ha habido errores durante todo el proceso por alguna excepción, devuelve un mensaje de OK a la capa de presentación por medio de la JSP, para que lo visualice el usuario 13. El usuario recibe una notificación del éxito de la operación 14. Fin
Variaciones (escenarios secundarios)	• **Escenario Alternativo 1**: Si el identificador de la planificación ya existe en el sistema este lo actualiza, si concuerda con el que ya existe en BBDD • **Escenario Alternativo 2**: En el paso 3, si el validador encuentra algún campo mal introducido o vacio se lo hace saber, no dejará enviarlo a la BBDD hasta que lo corrija. • **Escenario Alternativo 3:** Si ocurre alguna excepción tanto desde el transporte por las capas hasta donde se realizan las operaciones de BBDD, como en las propias operaciones y el viaje de vuelta ○ No realizarán ninguna acción y se informa del error al usuario
Excepciones	• **La clave y usuario para conectarse es incorrecta:** Puede ser correcta pero que se le haya cambiado el nombre de la clase y el Servicio Web no pueda comprar claves, o que de verdad no coincidan ○ Notificar el error por medio de un mensaje

	• **La conexión con el servidor no está disponible:** No se puede conectar y llevar los datos a la base de datos
	  ○ Notificar el error por medio de un mensaje
	• **La base de datos no está disponible**: No se pueden comparar el nombre de usuario ni la clave.
	  ○ Notificar el error por medio de un mensaje
	• **La base de datos produce un error en la consulta**: Error la consulta contiene algún tipo de error
	Notificar un error asociado al problema encontrado

Notas	En caso de errores, por lo que sea, se debe mostrar un mensaje de error, añadiendo en la medida de lo posible, una causa del mismo

5.5.8 Caso de Uso 4.2

Creación de Documentos PDF	
Precondiciones	El Manager/miembro debe estar validado con el rol perteneciente a la interfaz gráfica donde se encuentra, deben de haber datos sobre la gestión de riesgos en la base de datos para la creación de los documentos
Poscondiciones	Se crea finalmente un documento en PDF, que será guardado al finalizar su creación en el Home del usuario que lo ha pedido
Actores	Iniciado y terminado por el Administrador
Descripción	El Manager/miembro: 1. Accederá a la interfaz gráfica del rol validado 2. Acceder a la pantalla donde accede a la opción de crear informes 3. Seleccionara que tipo de informes, sobre qué etapa de la gestión de riesgos desea 4. El sistema trasladará la opción de qué tipo de datos de gestión se desea para obtenerlos de la base de datos 5. Una vez que le llega la petición del sistema este utiliza las diferentes interfaces de la aplicación y envía la petición al sistema de operaciones de la base de datos 6. Se escoge la operación que se va a realizar en la base de datos 7. Se abre una conexión a la base de datos 8. Se ejecutan las consultas pertinentes sobre la base de datos 9. La base de datos responde, devolviendo los objetos que son requeridos por el sistema 10. El sistema recoge los datos 11. El sistema envía los datos a la clase encargada de la manipulación y creación de documentos PDF 12. El sistema crea a partir de los datos y la opción determinada por el usuario el documento PDF 13. El sistema almacena dicho documento en el Home del usuario 14. El sistema devuelve un mensaje de confirmación de OK con la dirección donde se ha creado el documento PDF en el ordenador del usuario 15. Fin
Variaciones	• **Escenario Alternativo 1**: Se produce un error en alguno de los

(escenarios secundarios)	pasos intermedios del transporte, operaciones BBDD, creación PDF por las diferentes capas o Se termina la ejecución con un mensaje de error
Excepciones	• **La clave o el usuario para conectarse al servidor es incorrecta:** Puede ser correcta pero que se le haya cambiado el nombre de la clase y el Servicio Web no pueda comprar claves, o que de verdad no coincidan o Notificar el error por medio de una pantalla • **La base de datos no está disponible**: No se pueden comparar el nombre de usuario ni la clave. o Notificar el error por medio de una pantalla • **La base de datos produce un error en la consulta**: Error la consulta contiene algún tipo de error o Notificar un error asociado al problema encontrado
Notas	En caso de errores, por lo que sea, se debe mostrar un mensaje de error, añadiendo en la medida de lo posible, una causa del mismo

5.6 Análisis de Interfaces de Usuario

5.6.1 Descripción de la Interfaz

Este apartado se enseñará el diseño ya de las pantallas definitivas, solo para ver como quedarán en el producto final.

En esta sección se crea la especificación de las interfaces entre el usuario y el sistema a construir, incluyendo todos los diferentes tipos de pantallas que van a existir

5.6.1.1 Interfaz Inicial (Login)

Figura 5.9. Interfaz de Login

La interfaz gráfica de identificación, donde el usuario introducirá su email que identifica a cada usuario como único y su clave, el sistema comparara esos datos con los que tiene almacenados en la base de datos, si son correctos enseñara la interfaz gráfica correspondiente al rol que tiene asignada esa cuenta.

5.6.1.2 Interfaz Administrador (Registro Manager)

Figura 5.10. Interfaz del Administrador (Registro Manager)

La interfaz del **Administrador**. Todas las interfaz tiene común las zonas superiores y inferiores, dependiendo del rol que tenga el usuario se modificará el cuerpo central, apareciendo en el lo designado para ese rol, al igual que para el menú de navegación, que no todos los usuarios tendrán las mismas características y por ello serán diferentes. Comenzando por arriba a la derecha se encuentra que usuario se ha logueado, donde contiene el botón salir, para terminar con la sesión.

Contiene una barra de navegación con las opciones de el **Registro del Manager** que será la opción predeterminada inicial, **Info de Proyectos**, donde aparecen todos los proyectos creados y los managers también creados, con información de ambos y posibilidad de modificarlos, **Cuenta,** lugar donde se pueden modificar los valores de entrada a la aplicación y la **Ayuda** que explica que es lo que puede hacer este usuario en esta aplicación

Inicialmente muestra de entrada una de las opciones como es el dar de alta un nuevo **Manager**, para ello debe rellenar el formulario de la derecha, con los campos que pide. Una vez relleno se registra y el sistema le propondrá el asignarle un nuevo **Proyecto** a ese **Manager** recién creado**.**

A la izquierda está la lista de tareas de todo el proyecto, en amarillo, donde se encuentra actualmente el usuario. Otro de los posibles formatos es que se encuentre tachado, eso significa que esa tarea ya se completo.

Abajo está el pie, que índica las visitas a la página, la fecha completa y la IP desde donde se ha conectado, esto más adelante en las ampliaciones se comentará porque se ha decidido poner.

5.6.1.3 Info Proyectos Administrador

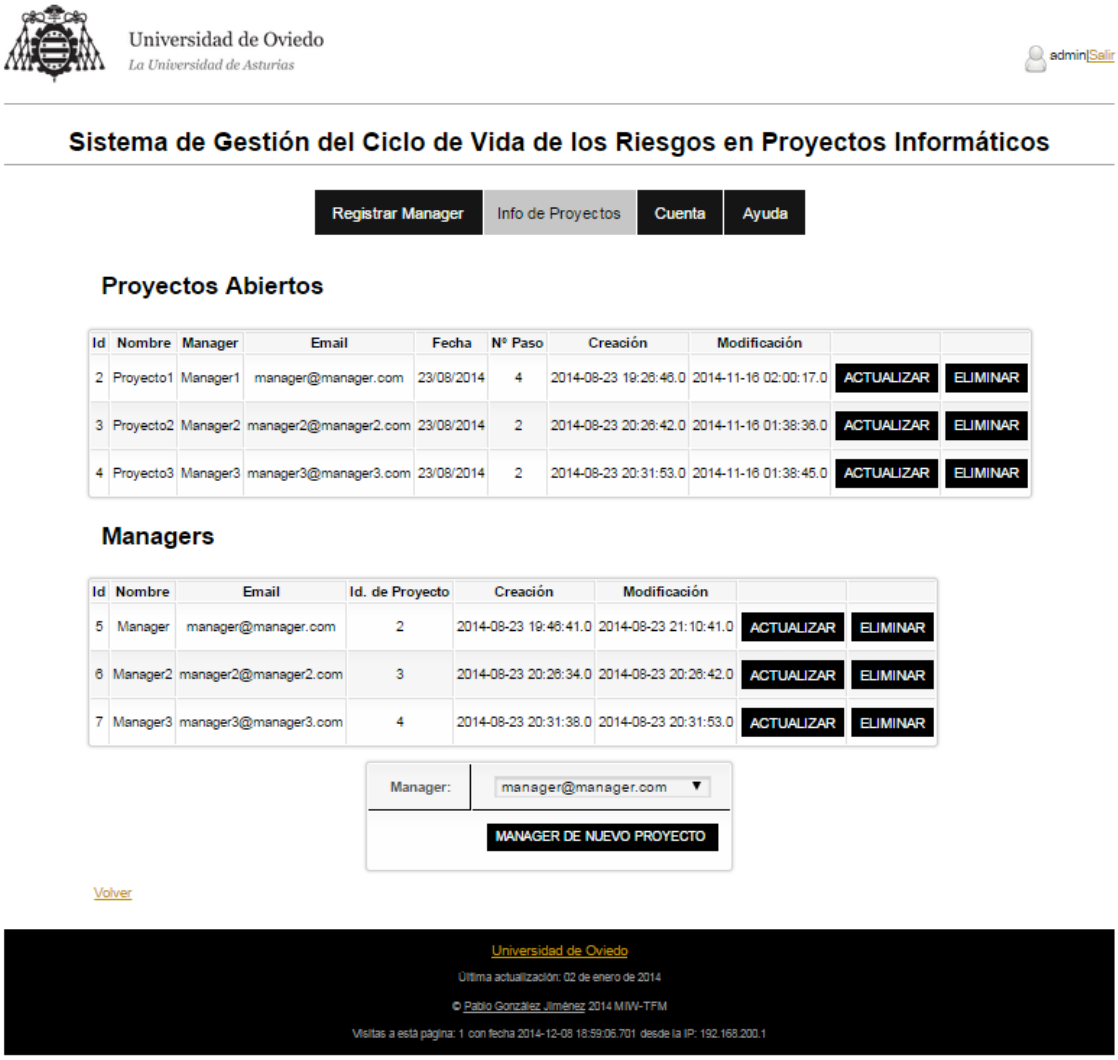


Figura 5.11. Información Proyectos

En la pestaña de Información, deberán aparecer todos aquellos **Proyectos** y **Managers**, que se hubieran creado, tiene la opción de actualizar o eliminar cualquiera de ellos, además, como es posible que al crear el **Manager**, no haya querido asignarle de momento un **Proyecto** ya que no es obligatorio, podrá hacerlo desde aquí, desde el botón **Manager de Nuevo Proyecto.**

5.6.1.4 Cuenta Administrador

Figura 5.12. Cuenta de Administrador

En esta opción el **Administrador** podrá cambiar los datos de acceso como son su login, su email, su contraseña o su idioma, una vez cambiado pulsará en el botón actualizar y cambiara los datos de cuenta.

5.6.1.5 Ayuda Administrador

Figura 5.13. Ayuda Administrador

La pestaña de Ayuda, ofrece un pequeño texto explicativo de lo que puede hacer el Administrador en su parte de la aplicación.

5.6.1.6 Interfaz Manager (Registro Miembro)

Figura 5.14. Interfaz Manager (Registro Miembro)

Esta será la interfaz principal del **Manager**, como se dijo en la del **Administrador** las zonas comunes serán iguales solo cambiaría el cuerpo central, el tipo de usuario logueado y el menú, que ahora la pestaña predeterminada tiene la opción del **Registro del nuevo Miembro.**

A la izquierda, al ser ya completados tanto el registro de este **Manager** como el del **Proyecto** al que está asignado aparecerán tachados y en amarillo constata que el **Manager** está en el punto de la creación de nuevos miembros.

A la derecha tendrá el formulario para completar los diferentes campos que se piden para crear este nuevo miembro al proyecto. El número de proyecto se pondrá automáticamente ya que el **Manager** está referenciado al **Manager**, la contraseña estará por defecto también con el valor "**cambiame**".

El Manager junto a los demás miembros tendrán la opción de crear su **Gestión de Riesgos del Proyecto**, por eso aparece esa nueva opción con un link, en el cuadro de la izquierda.

5.6.1.7 Info Proyectos Manager

Figura 5.15. Info Proyectos Manager

La pestaña común al Administrador como es la información de los proyectos, mostrara en este caso los proyecto de que el usuario es el Manager además del resto de personas que se encuentran en el.

5.6.1.8 Cuenta Manager

Figura 5.16. Cuenta Manager

Al igual que todos los usuarios, podrá actualizar sus datos de acceso desde la pestaña **Cuenta.**

5.6.1.9 Ayuda Manager

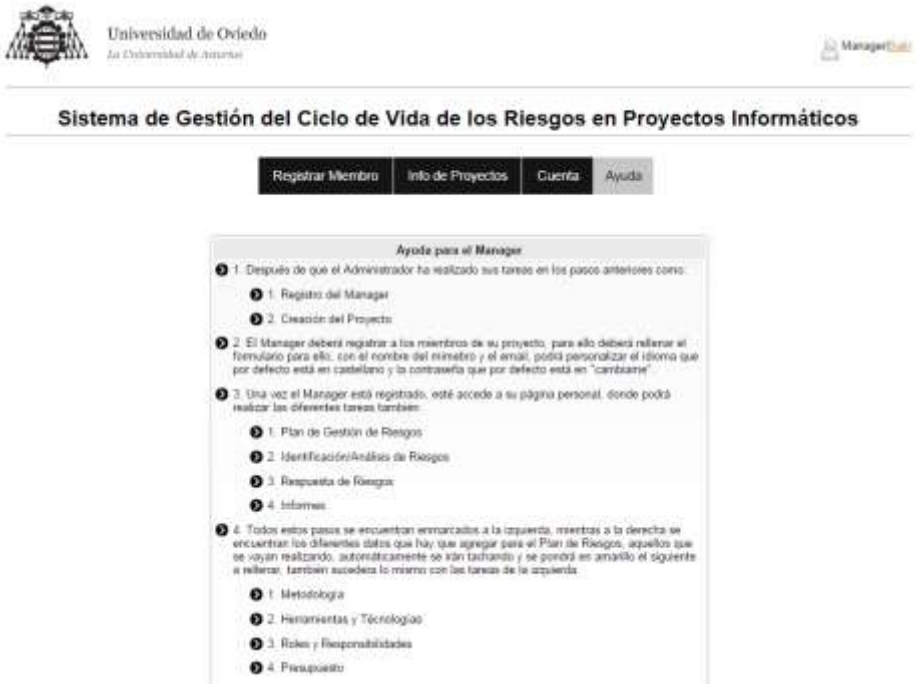


Figura 5.17. Ayuda Manager

La pestaña de Ayuda, ofrece un pequeño texto explicativo de lo que puede hacer el Manager en su parte de la aplicación.

5.6.1.10 Interfaz Inicial Miembro

Figura 5.18. Interfaz Miembro

Esta será la Interfaz inicial del Miembro, donde su tarea única y principal es la de crear la **Gestión de riesgos del Proyecto**, comenzando por el **Plan de Gestión de Riesgos, Identificación/Análisis de Riesgos, Respuesta de Riesgos y Informes.**

En el Plan de Riesgos, a la derecha aparecerán los diferentes campos a rellenar para completar esta etapa, si ya habido algún tipo de plan anterior se podrá ver los cambios realizados de la primera opción **Cambios y Versión**, además de poder cambiar hacía una de las versiones finalizadas anteriormente.

Figura 5.19. Identificación/Análisis de Riesgos

Cuando se tenga la planificación con los campos rellenos, se pasara a la interfaz de la **Identificación y Análisis de Riesgos,** donde se completarán aquellos campos que se necesite.

Figura 5.20. Respuesta a los Riesgos

Una vez identificado y analizados los riesgos, se crearán una respuesta a esos riesgos estudiados, los cuales deberán estar por encima de una nota de corte introducida por el usuario que está haciendo el estudio, si la nota de valor del riesgo es superior, ese riesgo necesitara de una respuesta por que puede producirse.

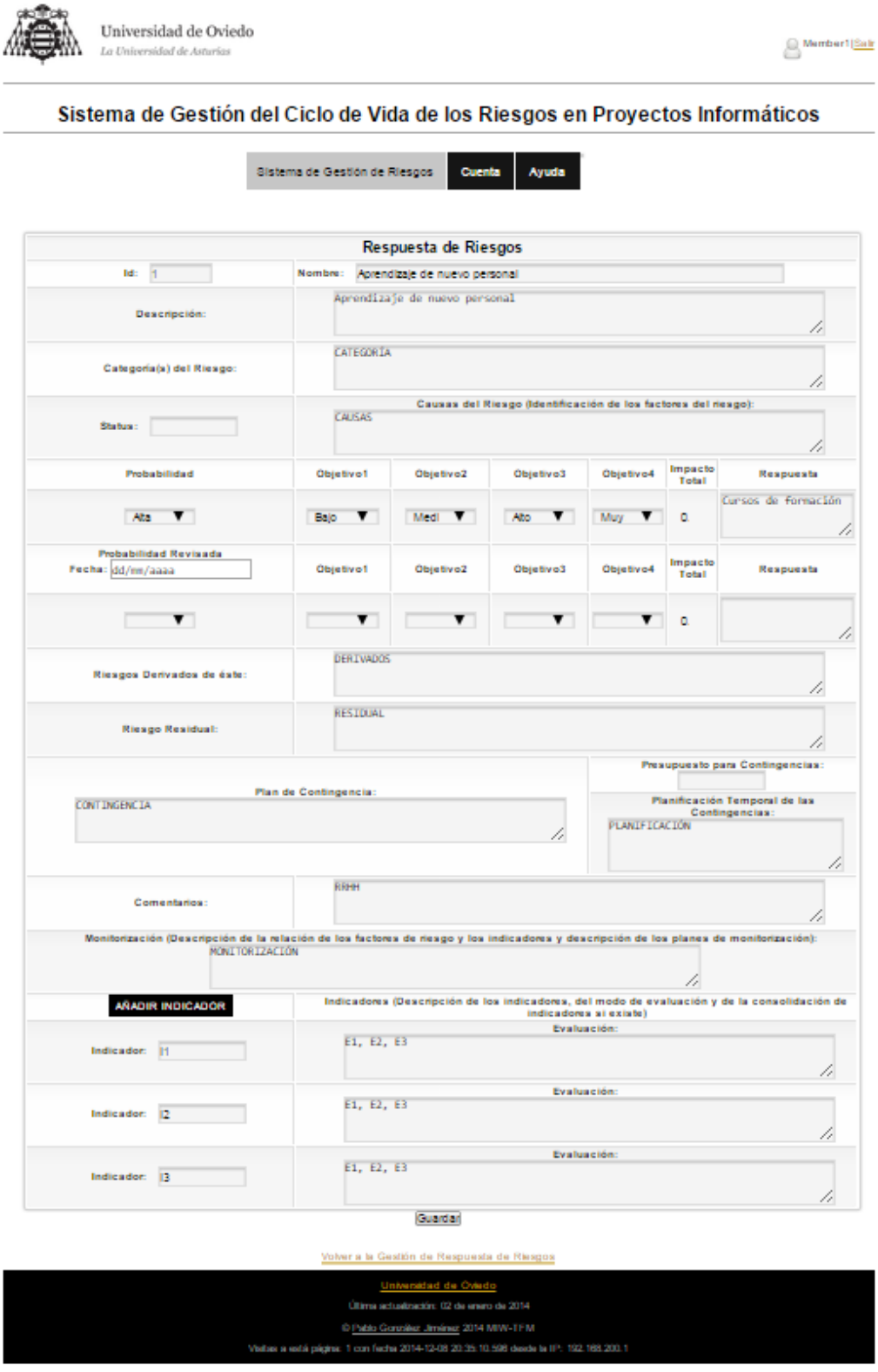


Figura 5.21. Respuesta al Riesgo

Finalmente la opción de **Creación de Informes,** es donde se crearán los documentos PDF, con los datos de cada una de las etapas anteriores.

Figura 5.22. Informes

5.6.1.11 Cuenta Miembro

Al igual que el **Administrador** y el **Manager**, los **Miembros** pueden acceder a sus datos y actualizarlos.

Figura 5.23. Cuenta Miembro

Y también una texto explicativo con las funciones y acciones que pueden realizar el miembro dentro de la aplicación Web, así como una explicación somera de las diferentes funciones a la hora de crear la **Gestión de Riesgos**.

5.6.1.12 Ayuda Miembro

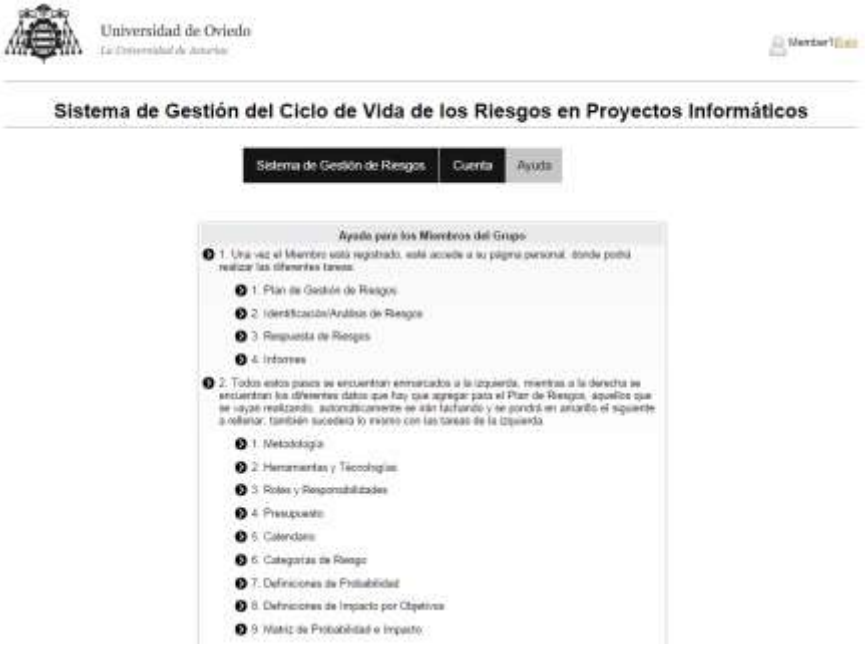


Figura 5.24. Ayuda Miembro

5.6.1.13 Contraseña Olvidada

Como interfaz exterior última, tendremos la interfaz para recuperar la contraseña, que envía un correo electrónico, al usuario registrado en la BBDD, que no se acuerda de su contraseña.

Figura 5.25. Contraseña Olvidada

5.6.2 Diagrama de Navegabilidad

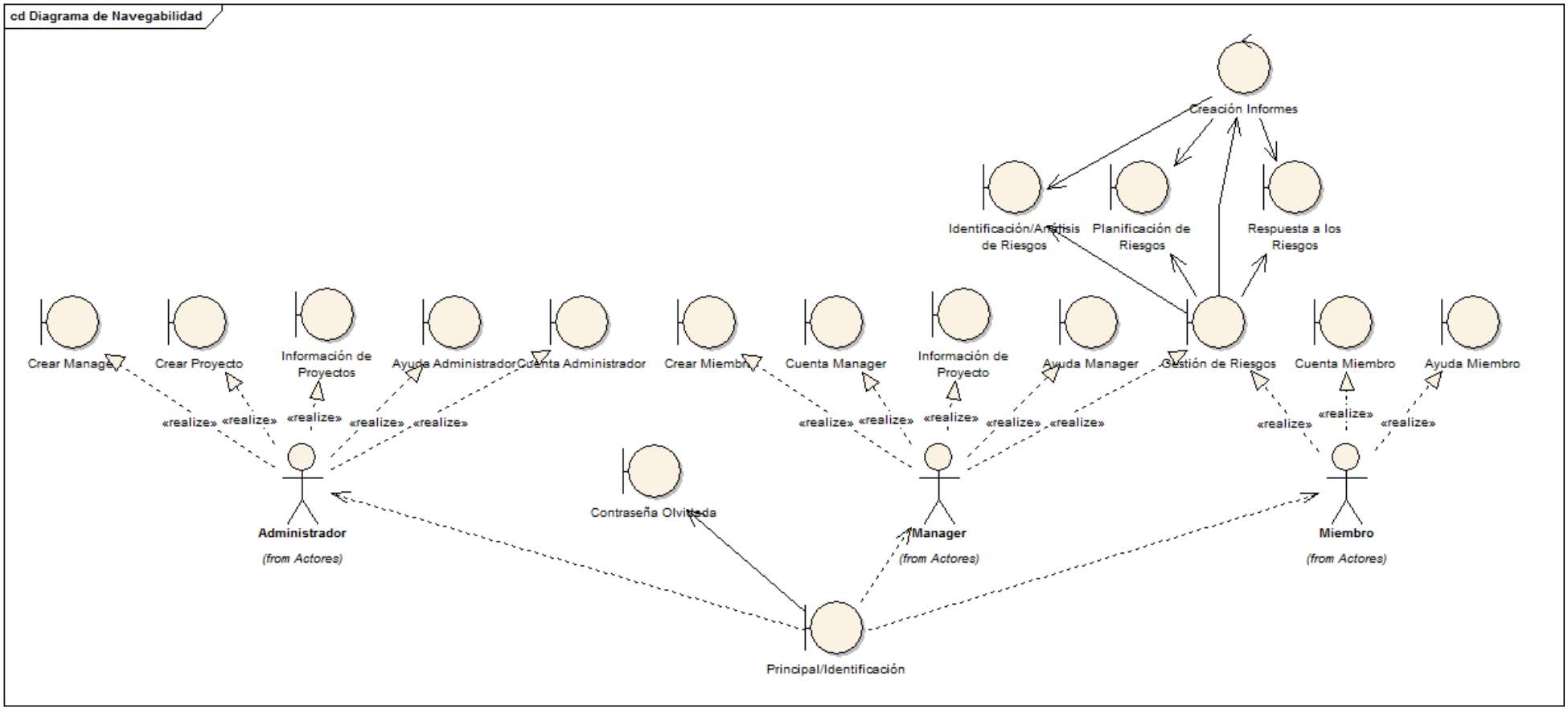


Figura 5.26. Diagrama de Navegabilidad

5.7 Especificación del Plan de Pruebas

En este punto se inicia la especificación del plan de pruebas, el cual sirve como guía para la realización de las diferentes pruebas que se irán realizando, y que permitirán verificar que el sistema cumple con las necesidades establecidas y con las debidas garantías de calidad. Las pruebas contemplarán aspectos tanto de funcionalidad de la aplicación como de aspectos de los usuarios o clientes con el mismo.

Se realizarán tres tipos de pruebas:

- **Pruebas Unitarias**: Una prueba unitaria es una forma de probar el correcto funcionamiento de un módulo de código. Esto sirve para asegurar que cada uno de los módulos funcione correctamente por separado. Luego, con la Prueba De Integración, se podrá asegurar el correcto funcionamiento del sistema o subsistema en cuestión. Para ello se utiliza el JUnit para desarrollar estas pruebas, el cual es un framework de código abierto desarrollado especialmente para crear, ejecutar y hacer reportes de estado de conjuntos de pruebas Unitarias automatizadas hechos en lenguaje Java.

    1. Base de datos
        a. Conexión base de datos
        b. Obtención de datos a la base de datos
        c. Establecimiento/actualización de datos a la base de datos
        d. Borrado de información a la base de datos
    2. Envío de correos electrónicos
    3. Conexión Servicio Web de terceros de OpenShift
    4. Conexión Servicio Web y base de datos de OpenShift
    5. Asociaciones dentro de cada modelo
    6. Creación de documentos de test en PDF

- **Pruebas de Integración**: Las pruebas de integración, se usarán para comprobar que los subsistemas al ensamblarlos juntos funcionan correctamente entre ellos como funcionaban en las pruebas unitarias por separado, para ello las pruebas será ver cómo los diferentes usuarios pueden realizar las acciones que cada uno tiene permiso según su rol, al integrarlos dentro del sistema, los cuales deberían funcionar de la misma manera y correctamente.

- **Pruebas del Sistema**: Las pruebas del sistema son el conjunto de integrar todos los subsistemas que integran la aplicación y probar que funcionan, tanto con la aplicación principal, como con otros sistemas, como puede ser el caso de la aplicación de creación de documentos PDF a partir de los datos creados por los usuarios de la gestión de riesgos que se ha creado aparte y que debe funcionar correctamente como lo hacen en las pruebas de integración de las aplicaciones de los diferentes usuarios, con ello permiten probar el sistema en su conjunto y con otros sistemas que realizan las mismas funciones, verificando así que todas sus especificaciones funcionales y técnicas se cumplen.

- **Pruebas de Usabilidad**: Este tipo de pruebas determinan la satisfacción del cliente con el producto final. Entre las pruebas los objetivos principales son:

    1. Determinar si el Administrador puede completar satisfactoriamente la creación de Manager y proyectos
    2. Determinar si el Manager creado anteriormente puede completar satisfactoriamente la creación de los demás miembros del proyecto
    3. Determinar si un Manager o miembro puede completar satisfactoriamente su parte en el proceso de la gestión de riesgos
    4. Determinar si las diferentes interfaces con que cuenta la aplicación son lo suficientemente intuitivas
    5. Determinar si el sistema de ayuda que se integra con la aplicación ayuda en la resolución de los problemas que le puedan surgir
    6. Determinar si la creación de documentos PDF cumple con su objetivo de facilitar la visión de los riesgos
    7. Determinar si la aplicación requiere modificaciones para que se cumplan los objetivos anteriores

Capítulo 6. Diseño del Sistema

6.1 Arquitectura del Sistema

6.1.1 Diagramas de Paquetes

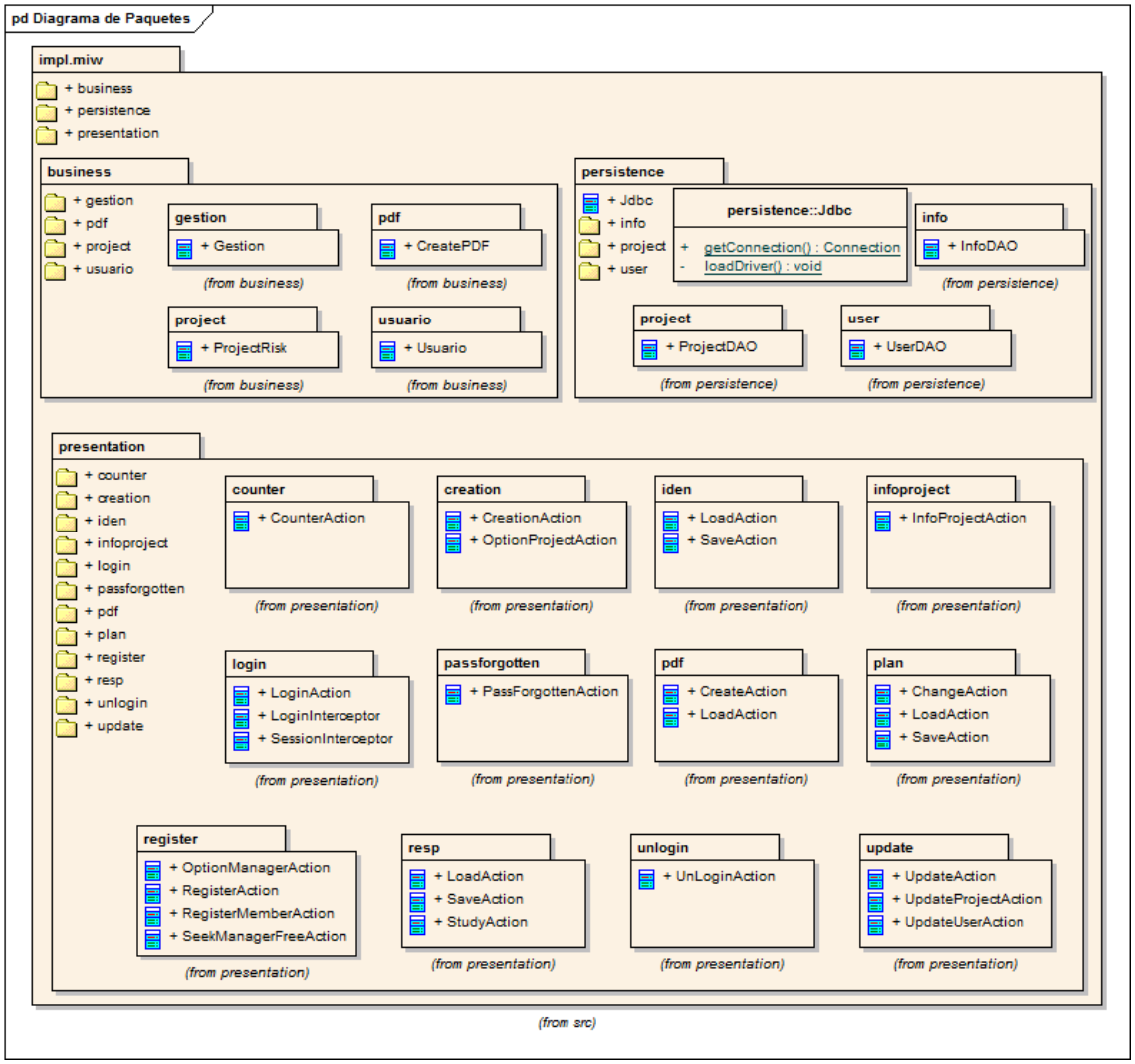


Figura 6.1. Diagrama de Paquetes SRC 1

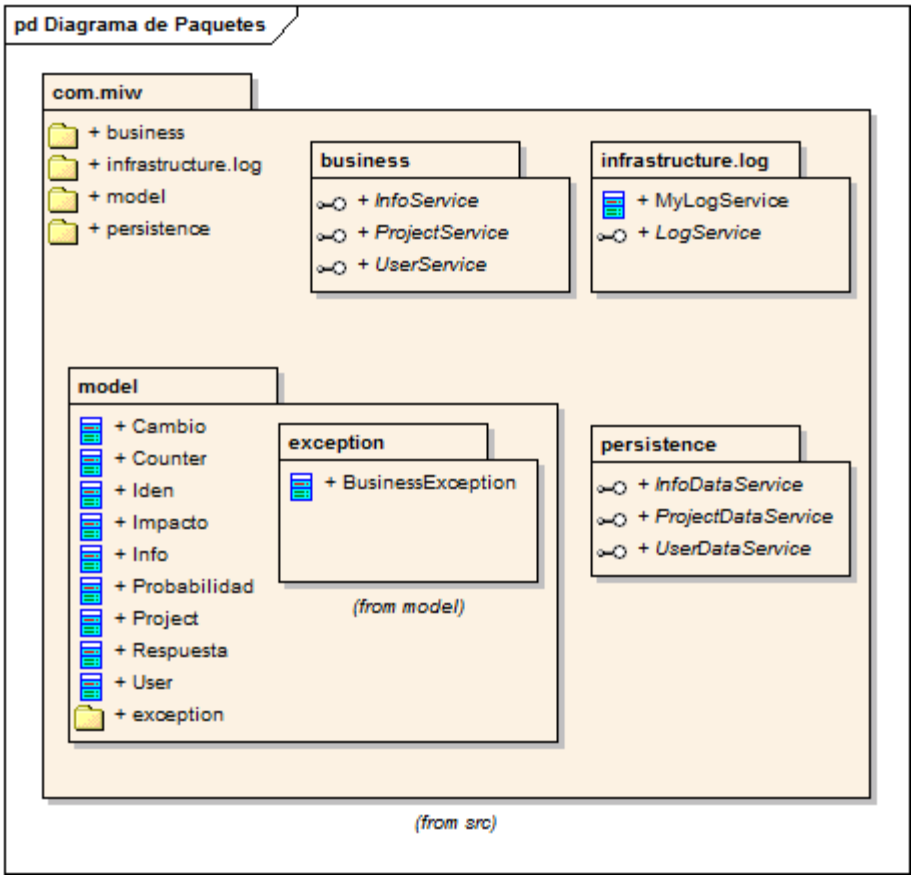


Figura 6.2. Diagrama de Paquetes SRC 2

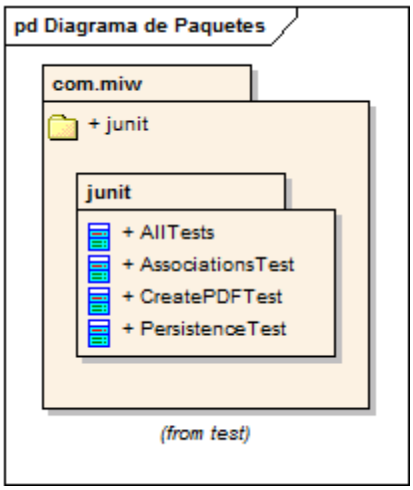


Figura 6.3. Diagrama de Paquetes Test

6.1.1.1 Paquete impl.miw

Descripción de los paquetes del diagrama (que tipo de elementos contiene, para qué sirven los mismos,…).

Paquete com.miw	
Nombre del paquete	**Descripción**
impl.miw.business	Este paquete se encuentra la capa de negocio, que comprende las responsabilidades de la lógica de negocio (o dominio) del sistema, tanto para los bean de los usuarios, de los proyectos, los riegos o la creación de los pdf, el resultado del análisis funcional, como el conjunto de reglas de negocio que abstraen el mundo real. Está capa es independiente de la capa de presentación y viceversa.
impl.miw.persistence	Comprende las responsabilidades de lógica de persistencia de las entidades que maneja el sistema en desarrollo. Aquí es donde se dan las diferentes operaciones que realiza la base de datos con los datos que le llegan de la aplicación: Create, Read, Update, Delete.
impl.miw.presentation	El paquete comprende las responsabilidades de lógica de presentación, todas las acciones de execute de los action del Struts, navegabilidad del sistema, validación de datos de entrada, formateo de los datos de salida, internacionalización, renderizado de presentación, creación de documentos pdfs, creación y actualizaciones de las gestiones de riesgos, proyecto usuarios, login, unlogin, etc.

6.1.1.2 Paquete com.miw

Paquete impl.miw	
Nombre del paquete	**Descripción**
com.miw.business	Este paquete es donde se encuentran las interfaces que conectan la capa de presentación a la capa de negocio, es una de las dos Façade, que tiene la arquitectura. Su función es reducir la complejidad de la aplicación, cada una de las interfaces esta dedica a uno de los objetos Bean que contempla la aplicación, Usuarios; Proyecto; Gestión de los Riesgos.
com.miw.model	En Model, se encuentran los diferentes entidades que usa la aplicación para

	encapsular los datos, que posteriormente van a almacenarse en la BBDD. Además contiene el paquete exception, que contiene una excepción de negocio, para los errores que se produzcan en esa capa.
com.miw.persistence	Este paquete es donde se encuentran las interfaces que conectan la capa de negocio a la capa de persistencia, es la otra Façade que se utiliza en la arquitectura. Su función es reducir la complejidad de la aplicación, cada una de las interfaces esta dedica a uno de los objetos Bean que contempla la aplicación, Usuarios; Proyecto; Gestión de los Riesgos.
com.miw.infrastructure	En este componente es el que da lugar a la capa de infraestructura, que se utiliza para un servicio de log de la aplicación. Contiene una interfaz, donde se redirige todas las llamadas que se hacen desde las demás capas a la clase que está contiene para generar el log correspondiente.

6.1.1.3 Paquete Test

Paquete Test	
Nombre del paquete	**Descripción**
com.miw.junit	Este paquete contiene las clases para producir los Tests de JUnit que prueban las diferentes partes de la aplicación, tiene una clase principal, que llama a las demás para correr los diferentes tests.

6.1.2 Diagramas de Componentes y Despliegue

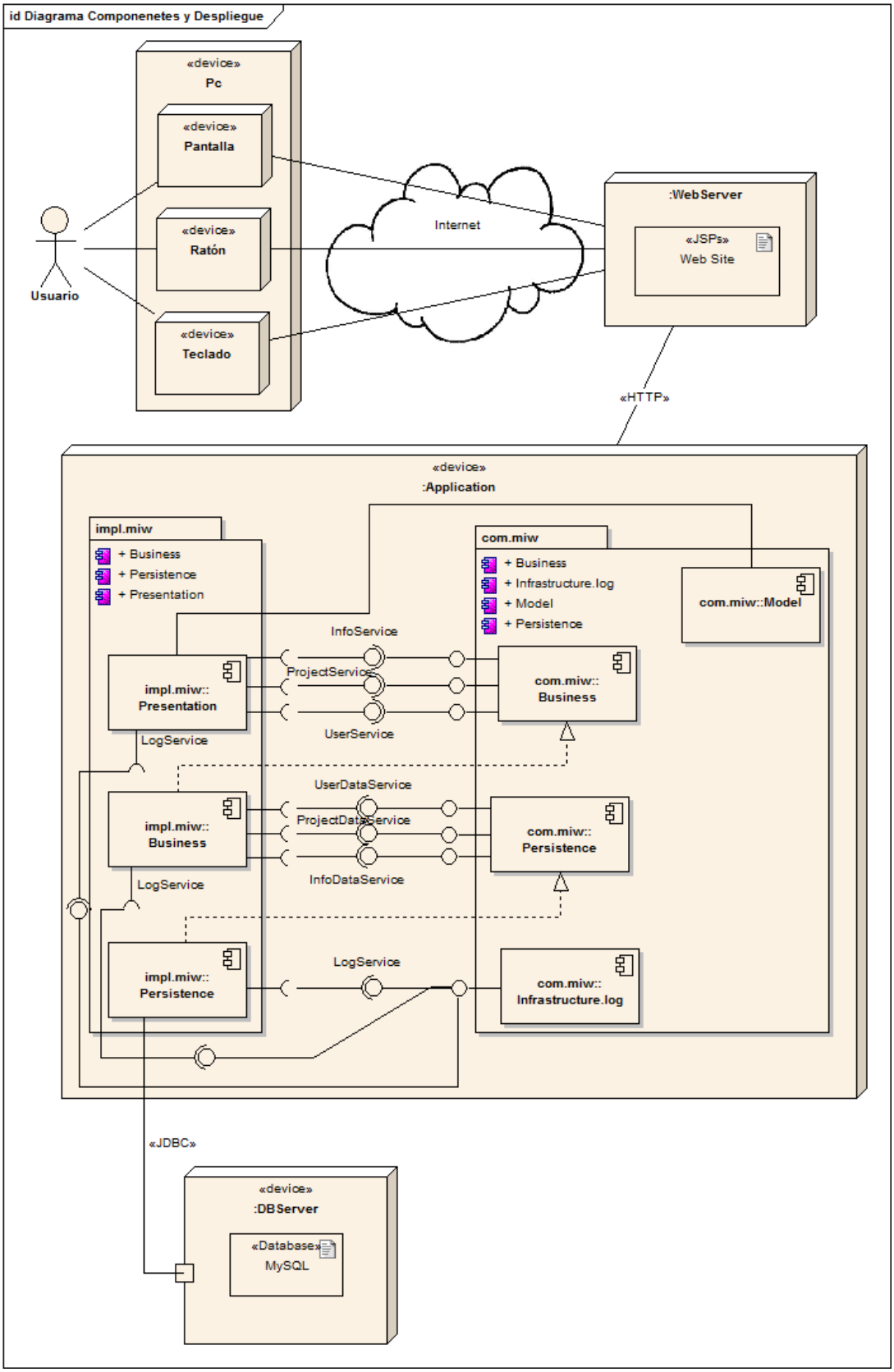


Figura 6.4. Diagramas de Componentes y Despliegue

6.1.2.1 PC

Son los elementos como el ratón, la pantalla o el teclado, elementos que se utilizan, con los que el usuario se conectara, desde donde se encuentre a través de Internet, para acceder a la aplicación alojada en un servidor web.

6.1.2.2 Web Server

El servidor web es donde se alojan todas las JSPs que la aplicación tiene, a partir de las peticiones que recibe por medio del usuario y las que devuelve la aplicación al usuario, todas ellas interactúan por medio del protocolo HTTP, estas JSPs enseñan las funcionalidades que tiene la aplicación y donde el usuario por medio de sus acciones puede invocarlas una vez se halla logueado correctamente en la página, dependiendo del rol que tenga, podrá acceder a unas partes y otras no, al igual que realizar unas acciones y otras no.

6.1.2.3 Application

Es el corazón donde se realiza la mayor parte de la actividad tanto de la **presentación** la cual hace de intermediaria entre el usuario y las acciones que ha elegido este por medio de las JSPs, es el mismo modo por donde la aplicación se comunicará con el resultado de esas acciones presentándoselas al usuario; como de la capa de **negocio** donde se realizan las tareas de la lógica, entre ellas la construcción de los PDFs y todos los demás subsistemas para la obtención o el establecimiento de los proyectos, usuarios y datos de la Gestión de Riesgos; además de la **persistencia**, lugar que desde el negocio le transportará y recogerá dependiendo de las operaciones que tena que hacer sobre este apartado con los diferentes datos que le son transmitidos a su vez desde la capa de presentación o de la misma base de datos, lugar donde almacenara y obtendrá todos aquellos que necesite

6.1.2.4 DBServer

Este elemento es donde finalmente se almacenan o recogen las diferentes tipos de información, que se le está pidiendo por medio de la capa de persistencia y que van dirigidas a dar funcionalidad a las acciones que el usuario ha realizado desde las JSPs.

6.2 Diseño de Clases

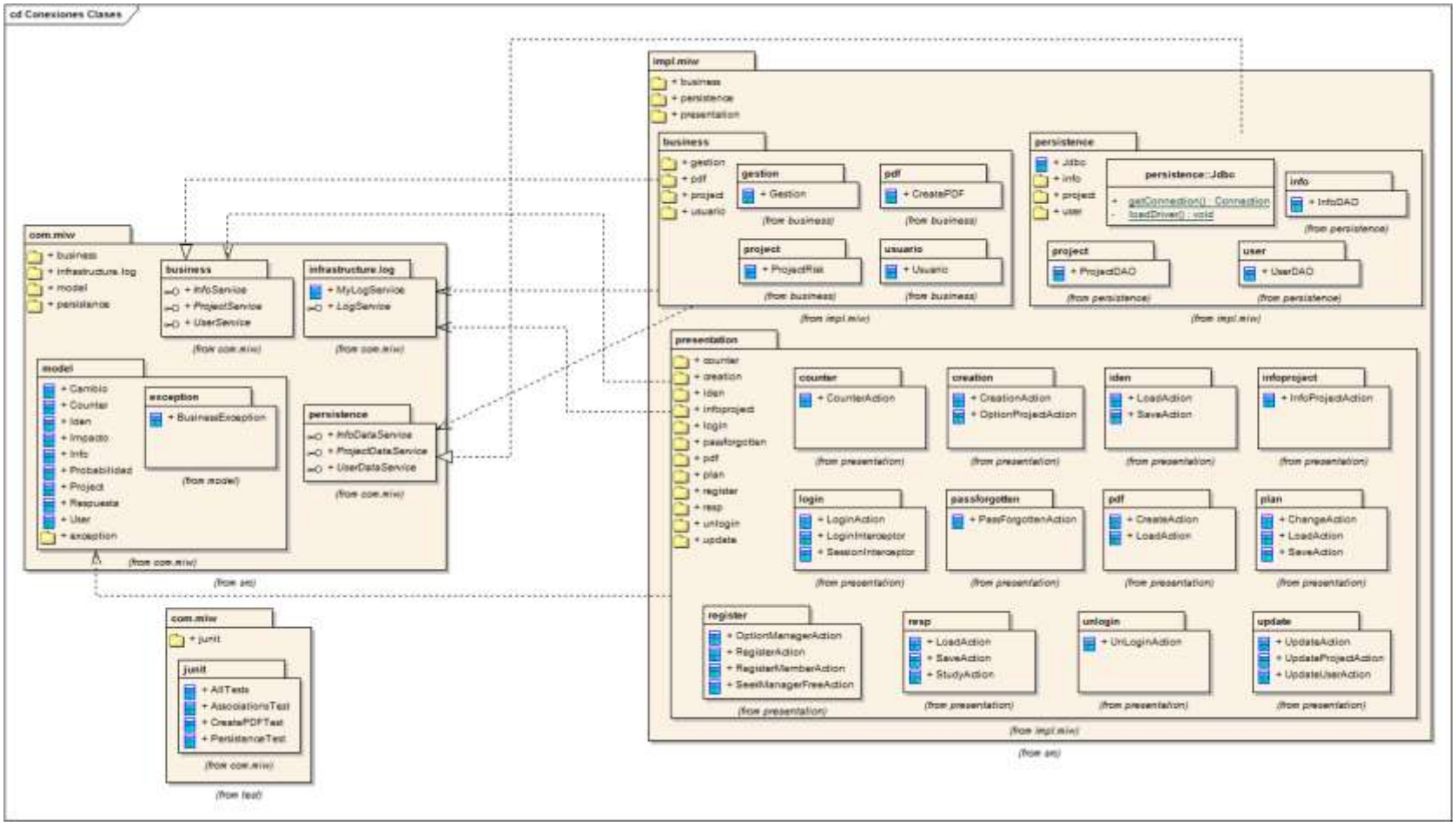


Figura 6.5. Conexiones Clases

6.2.1 Paquete Presentación

6.2.1.1 Diagrama de Clases Identificación

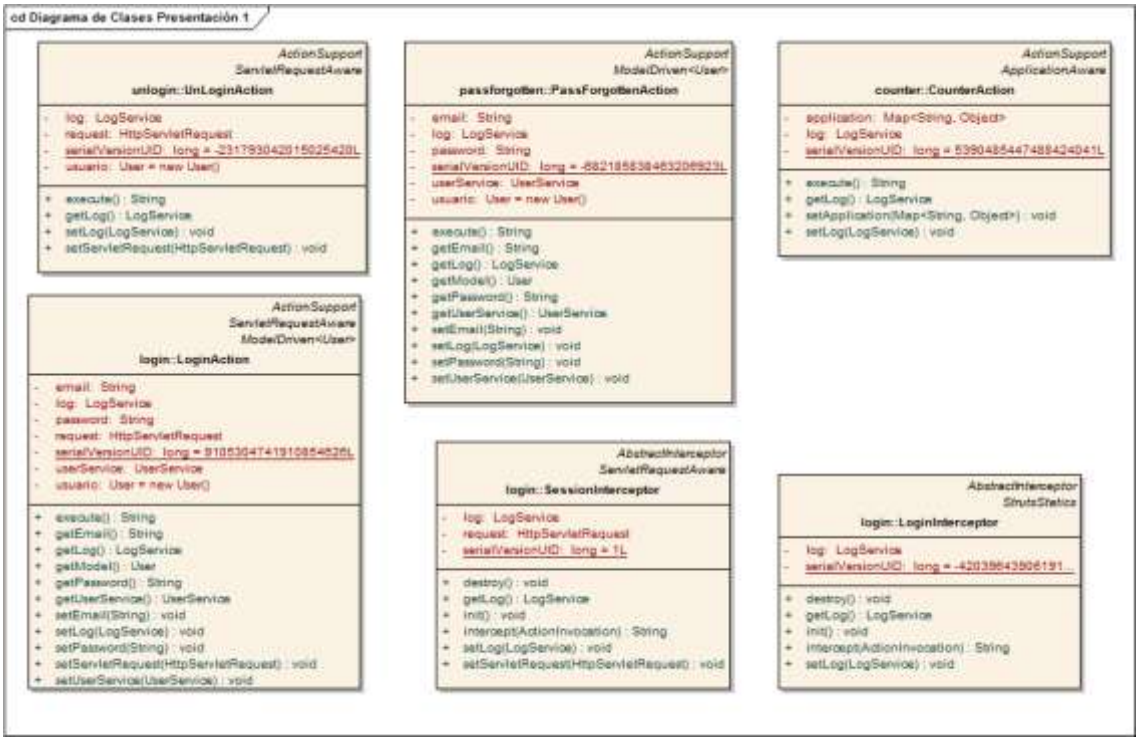


Figura 6.6. Diagrama Clases Presentación 1

6.2.1.2 Diagrama de Clases Identificación

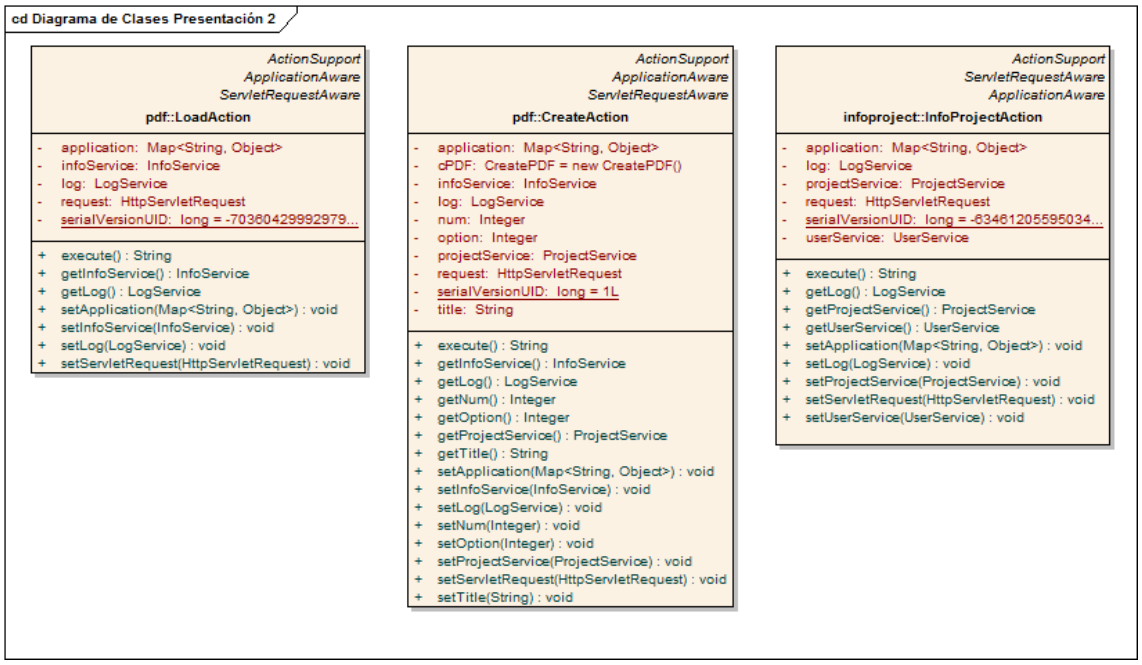


Figura 6.7. Diagrama de Clases Presentación 2

6.2.1.3 Diagrama de Clases Acciones Datos

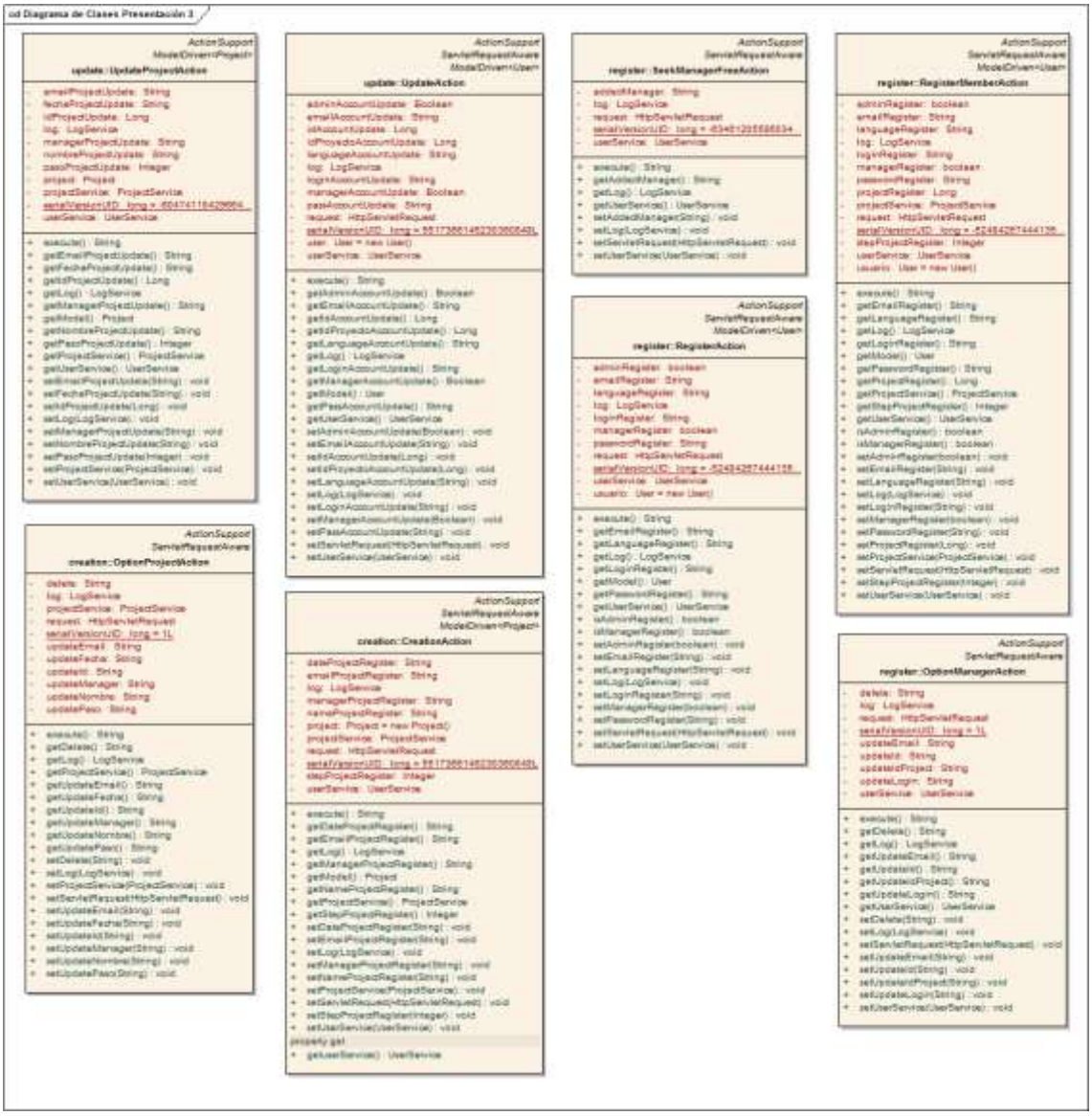


Figura 6.8. Diagrama de Clases Presentación 3

6.2.1.4 Diagrama de Clases Gestión de Riesgos

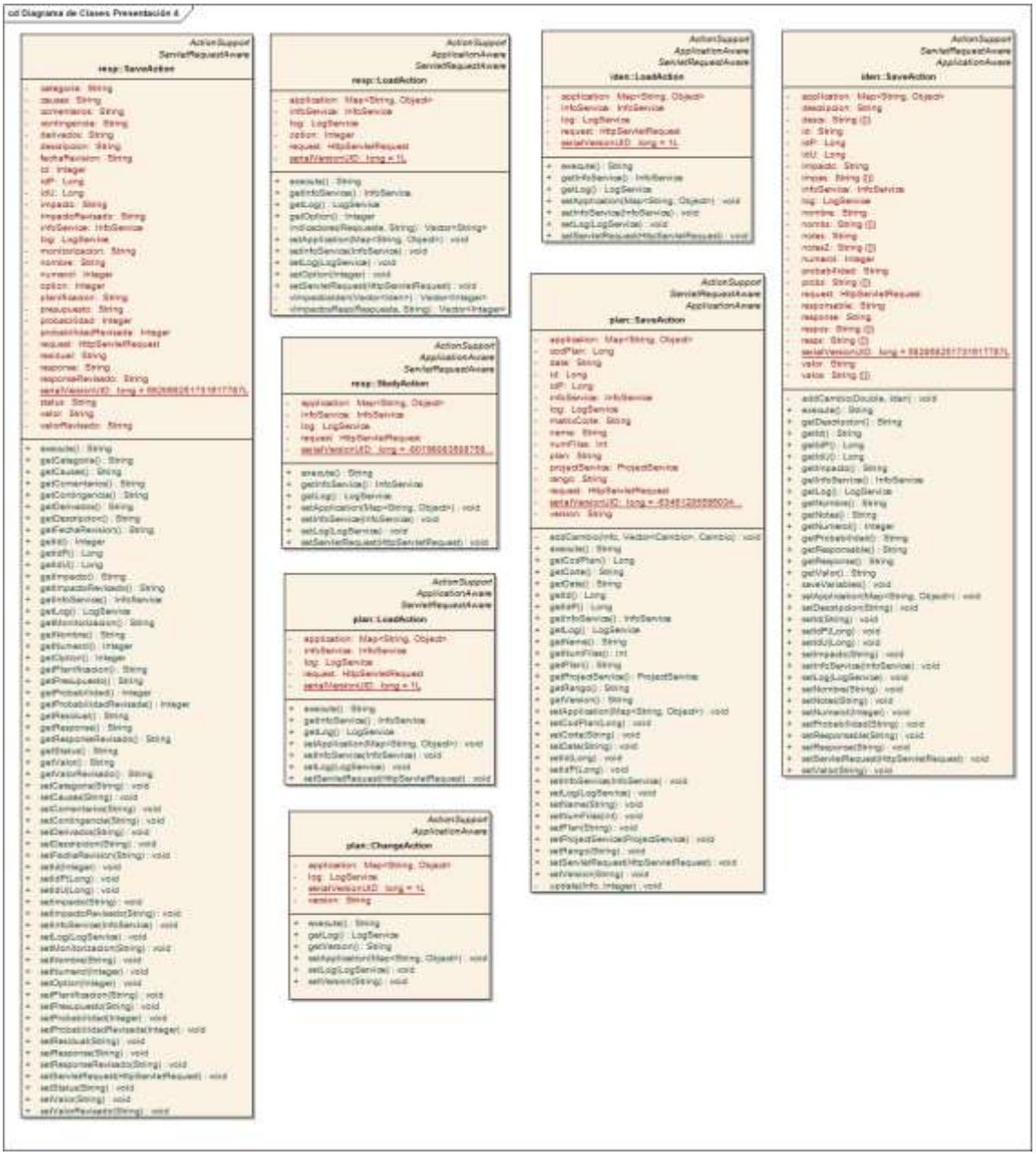


Figura 6.9. Diagrama de Clases Presentación 4

6.2.2 Diagrama de Clases Negocio (impl.miw)

Figura 6.10. Diagrama de Clases Business (impl)

6.2.3 Diagrama de Clases Persistencia (impl.miw)

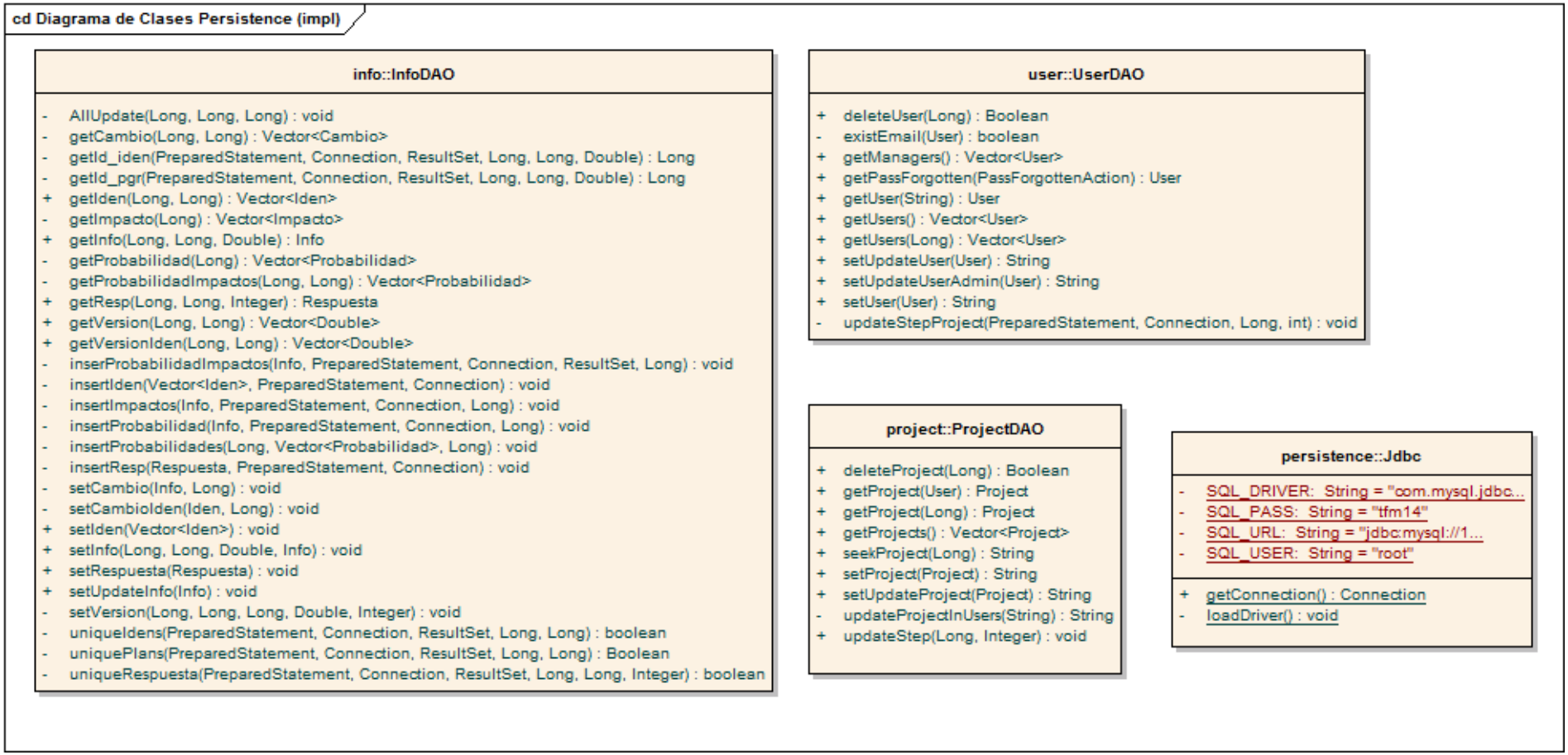


Figura 6.11. Diagrama de Clases Persistencia (impl)

6.2.4 Diagrama de Clases Modelo

Figura 6.12. Diagrama de Clases Modelo

6.2.5 Diagrama de Clases Business (com)

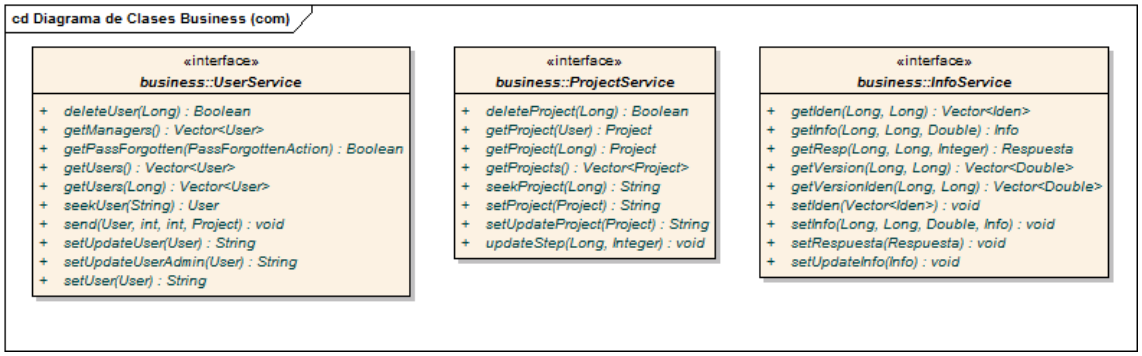


Figura 6.13. Diagrama de Clases Business (com)

6.2.6 Diagrama de Clases Infraestructura

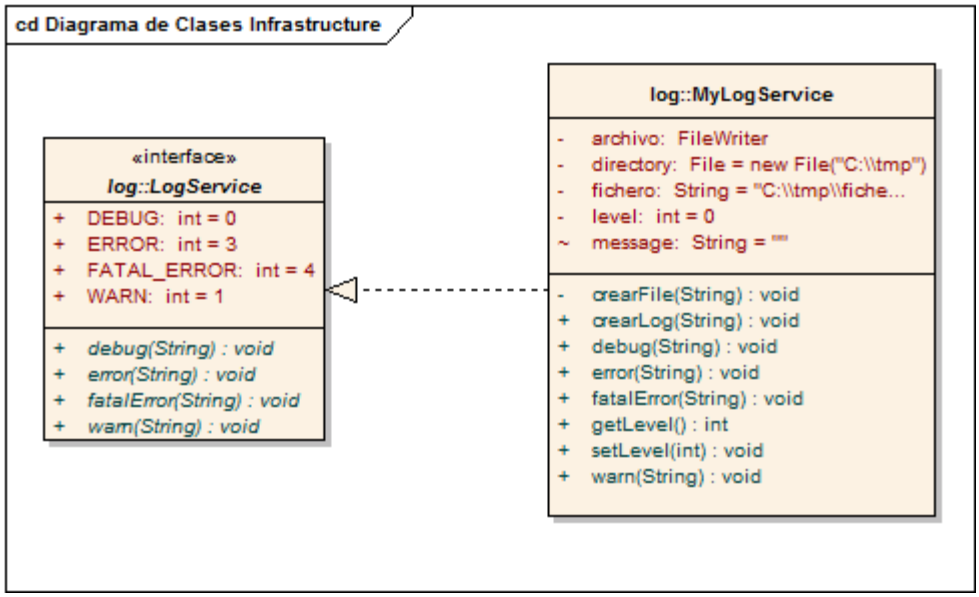


Figura 6.14. Diagrama de Clases Log

6.2.7 Diagrama de Clases Persistencia (com)

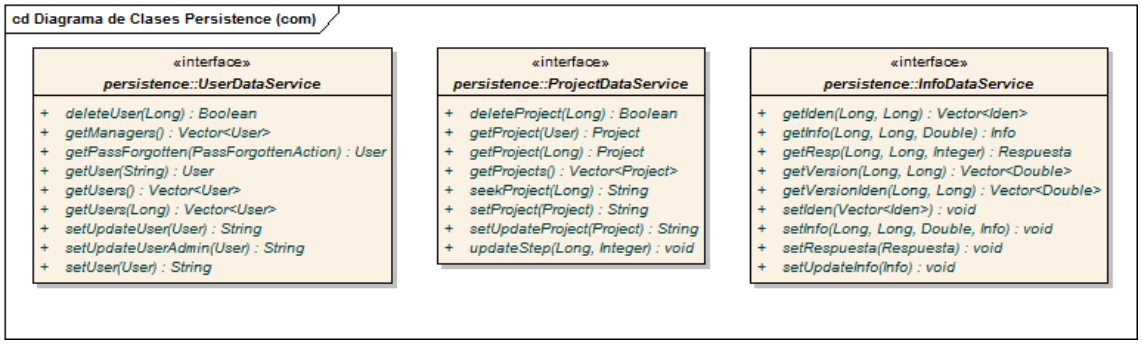


Figura 6.15. Diagrama de Clases Persistencia (com)

6.2.8 Diagrama de Clases Tests

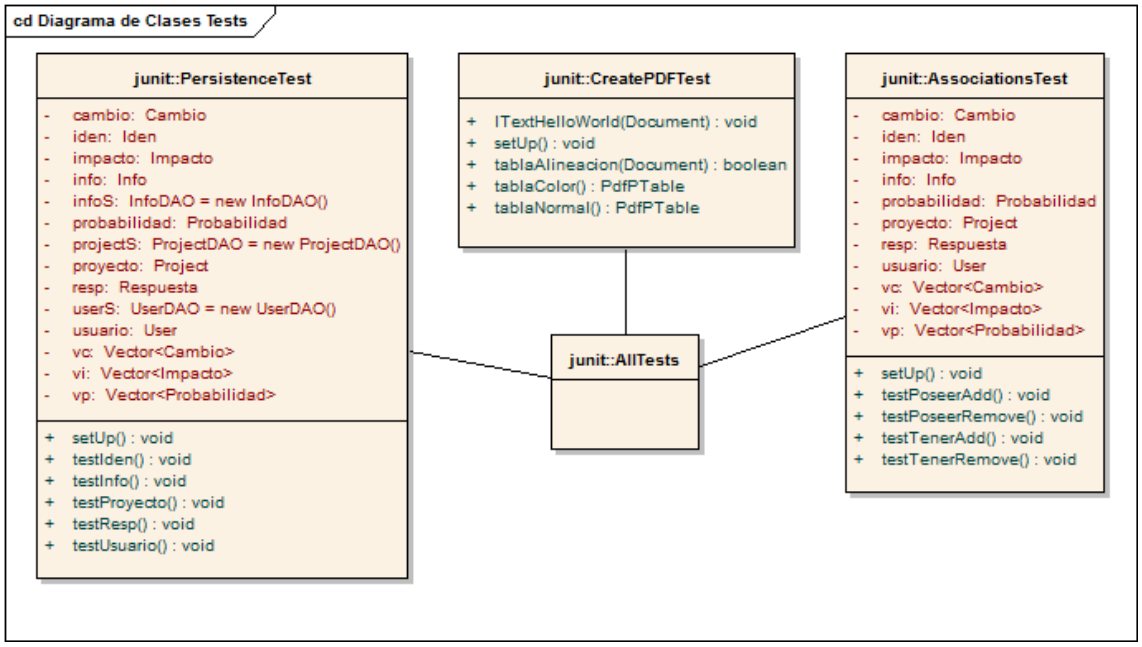


Figura 6.16. Diagrama de Clases Tests

6.3 Diagramas de Interacción y Estados

6.3.1 Caso de Uso 1

6.3.1.1 Diagramas de Interacción Login

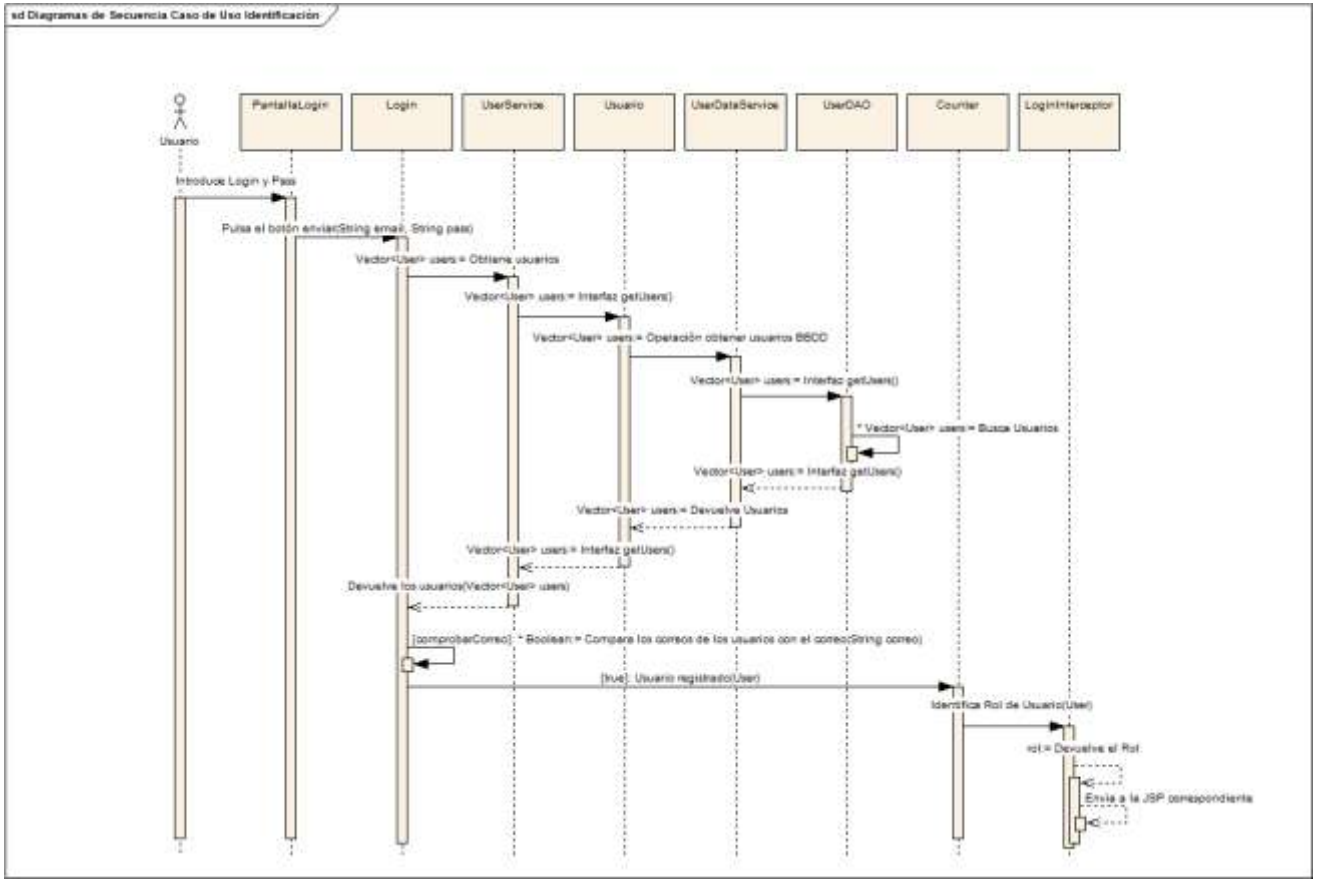


Figura 6.17. Diagrama de Interacción Login

6.3.1.2 Diagramas de Interacción Login Alternativa

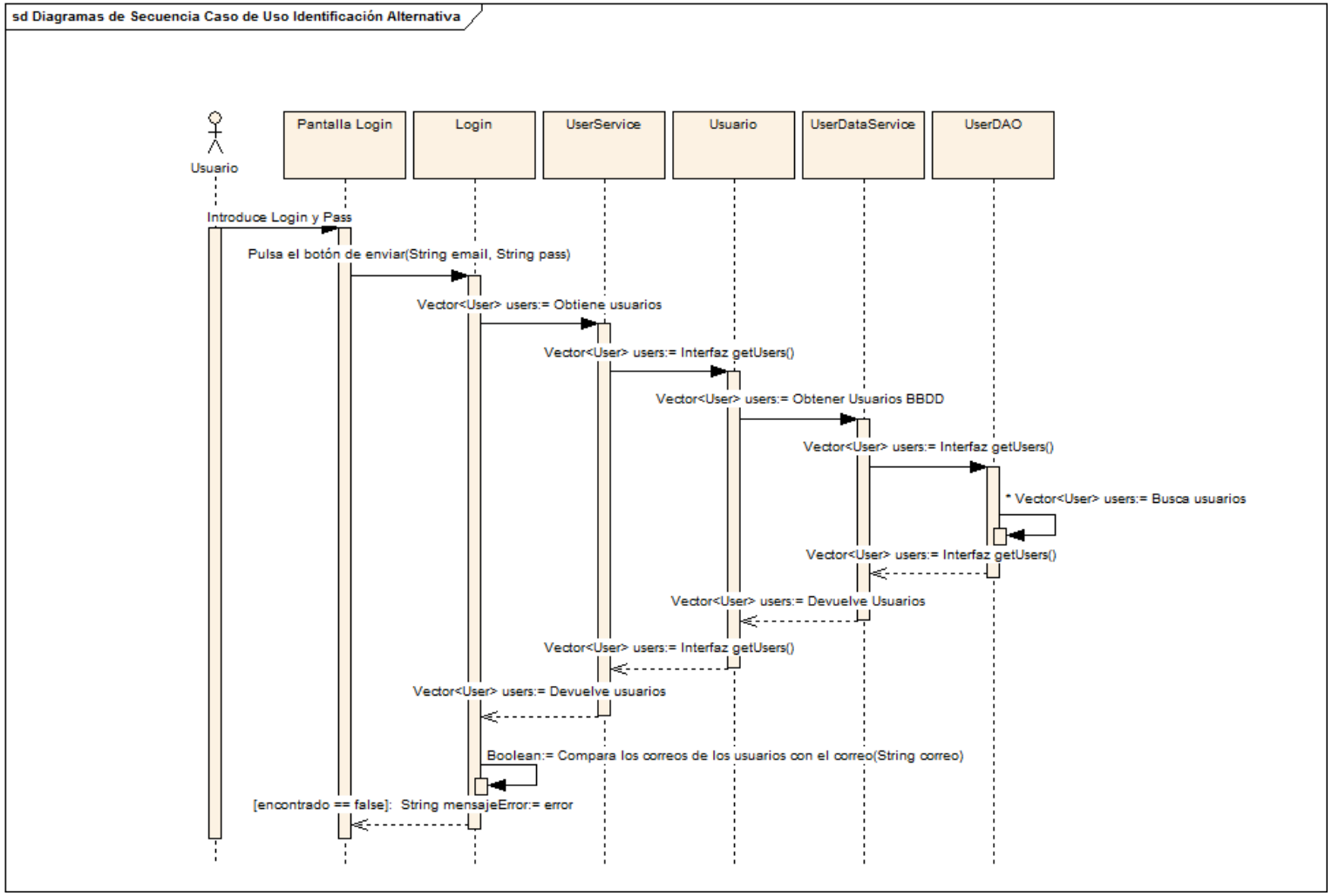

sd Diagramas de Secuencia Caso de Uso Identificación Alternativa
Usuario
Pantalla Login
Login
UserService
Usuario
UserDataService
UserDAO
Introduce Login y Pass
Pulsa el botón de enviar(String email, String pass)
Vector<User> users:= Obtiene usuarios
Vector<User> users:= Interfaz getUsers()
Vector<User> users:= Obtener Usuarios BBDD
Vector<User> users:= Interfaz getUsers()
* Vector<User> users:= Busca usuarios
Vector<User> users:= Interfaz getUsers()
Vector<User> users:= Devuelve Usuarios
Vector<User> users:= Interfaz getUsers()
Vector<User> users:= Devuelve usuarios
Boolean:= Compara los correos de los usuarios con el correo(String correo)
[encontrado == false]: String mensajeError:= error

Figura 6.18. Diagrama de Interacción Login Alternativa

6.3.2 Caso de Uso 1.2

6.3.2.1 Diagramas de Interacción Contraseña Olvidada

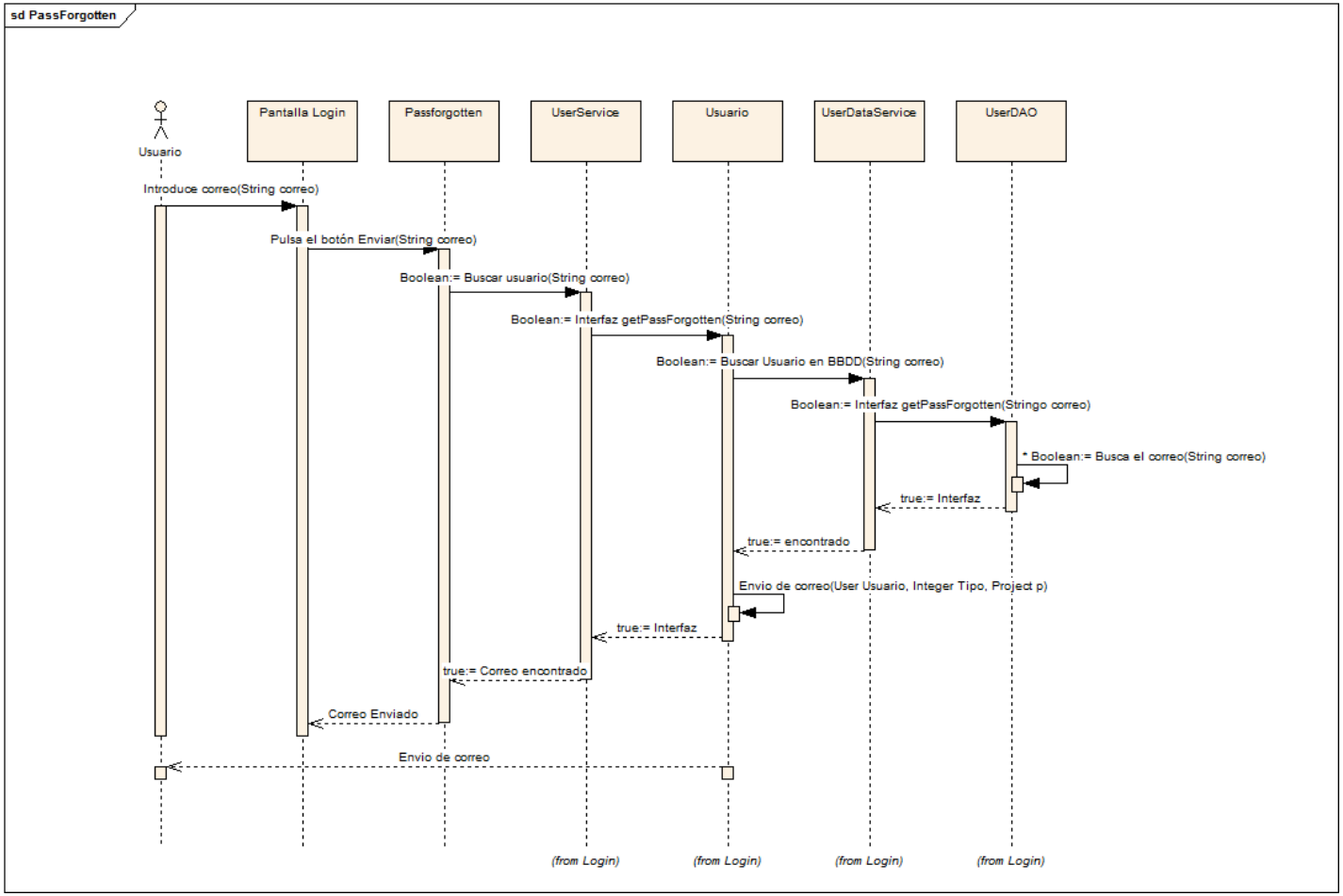


Figura 6.19. Diagrama de Interacción Contraseña Olvidada

6.3.2.2 Diagramas de Interacción Contraseña Olvidada Alternativa

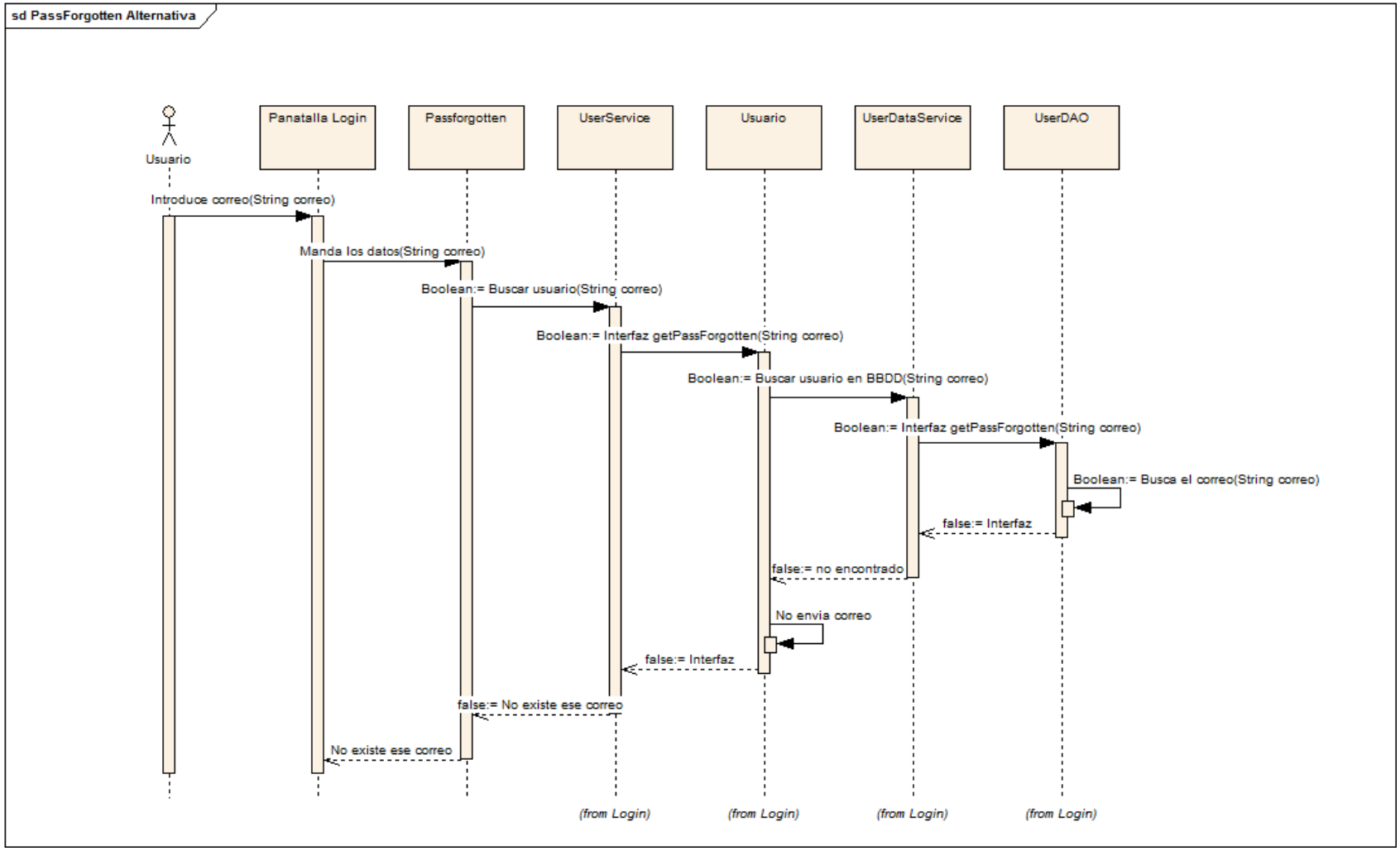

sd PassForgotten Alternativa
Usuario
Panatalla Login
Passforgotten
UserService
Usuario
UserDataService
UserDAO
Introduce correo(String correo)
Manda los datos(String correo)
Boolean:= Buscar usuario(String correo)
Boolean:= Interfaz getPassForgotten(String correo)
Boolean:= Buscar usuario en BBDD(String correo)
Boolean:= Interfaz getPassForgotten(String correo)
Boolean:= Busca el correo(String correo)
false:= Interfaz
false:= no encontrado
No envia correo
false:= Interfaz
false:= No existe ese correo
No existe ese correo
(from Login)
(from Login)
(from Login)
(from Login)

Figura 6.20. Diagrama de Interacción Contraseña Olvidada Alternativa

6.3.3 Caso de Uso 2

6.3.3.1 Registrar Manager y Proyecto

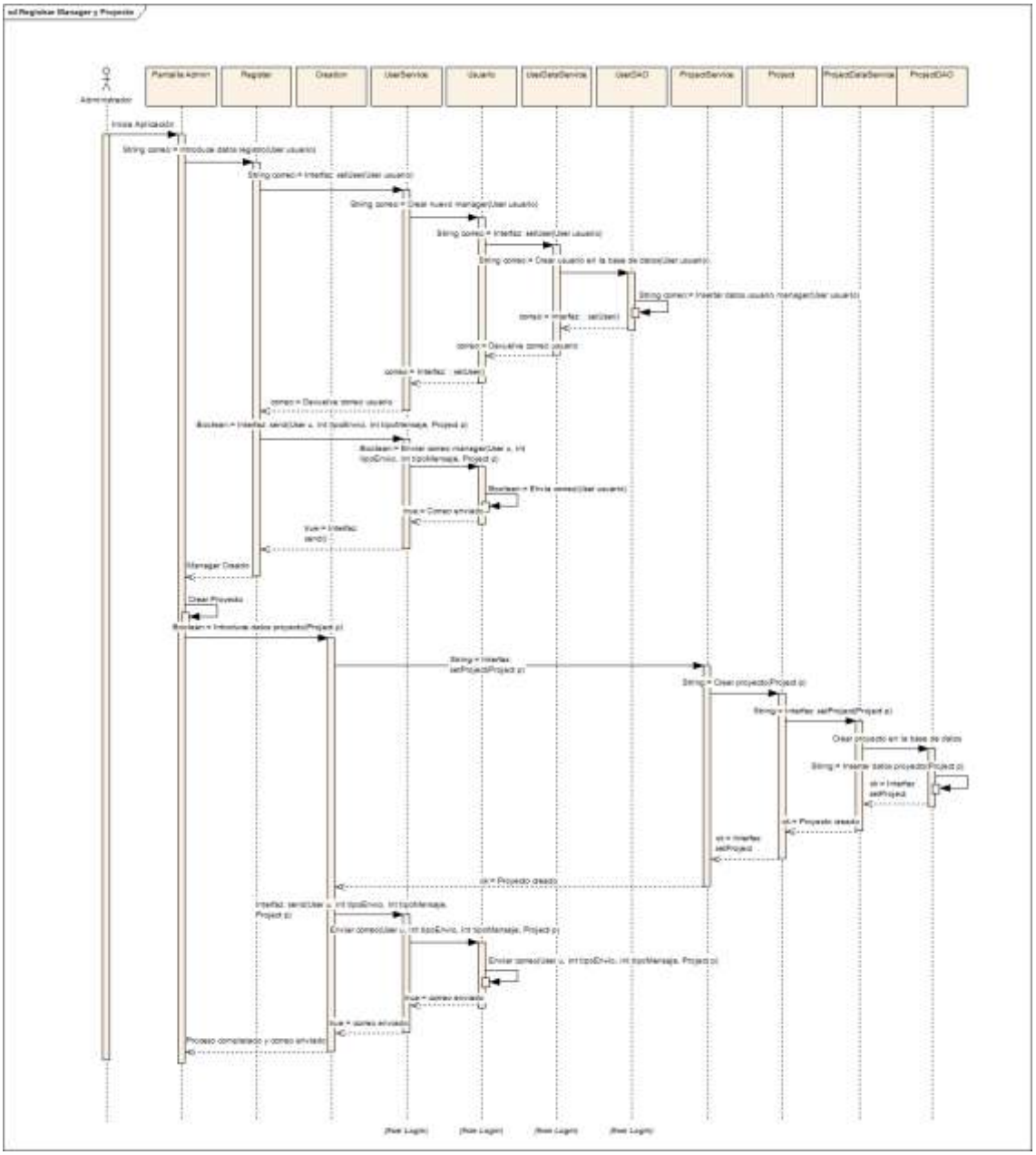


Figura 6.21. Diagrama de Interacción Registro Manager y Proyecto

6.3.4 Caso de Uso 2.1

6.3.4.1 Visualizar Managers y Proyectos

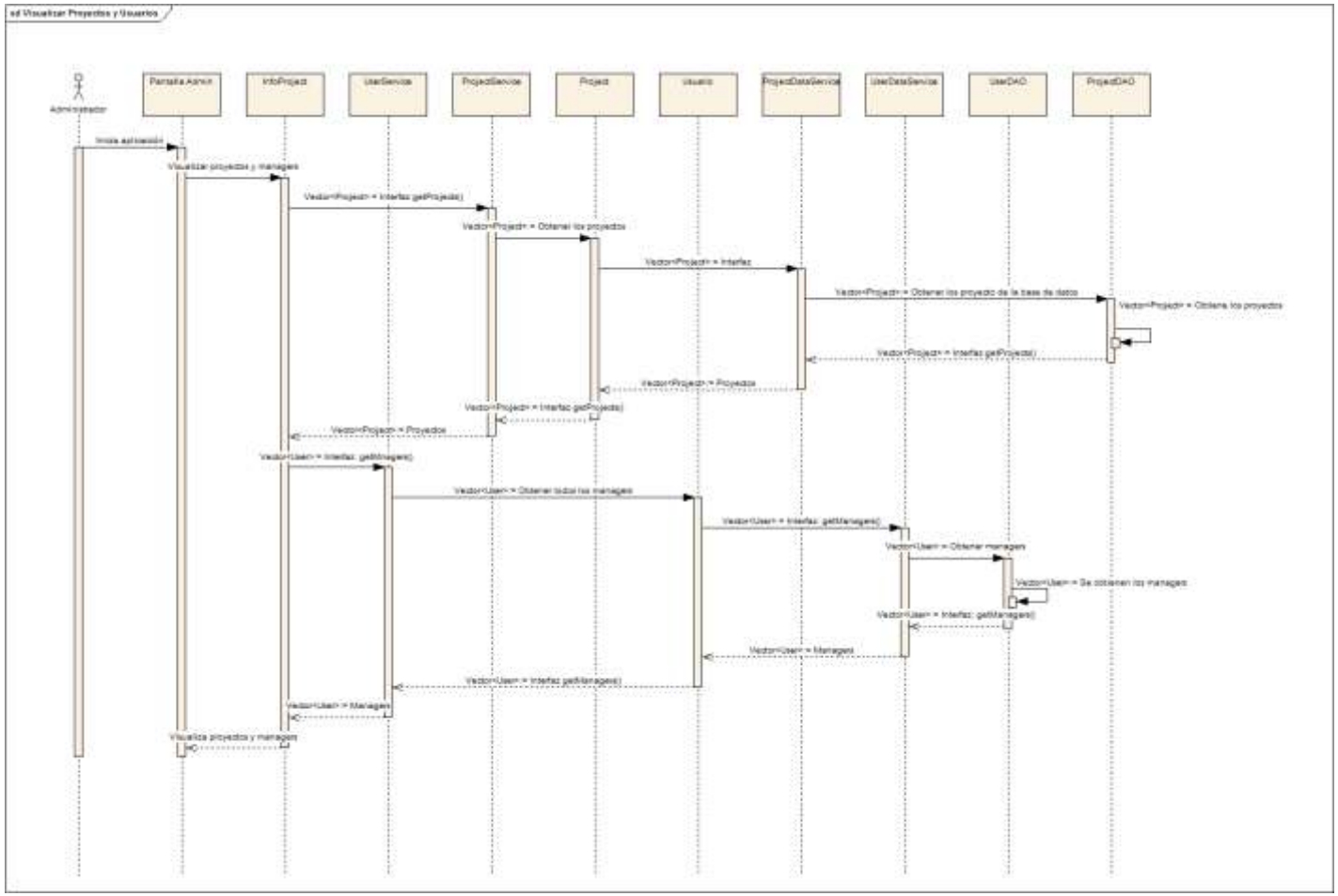


Figura 6.22. Diagrama de Interacción Visualizar Proyectos y Managers

6.3.5 Caso de Uso 4

6.3.5.1 Recuperar y Visualizar Gestión de Riesgos

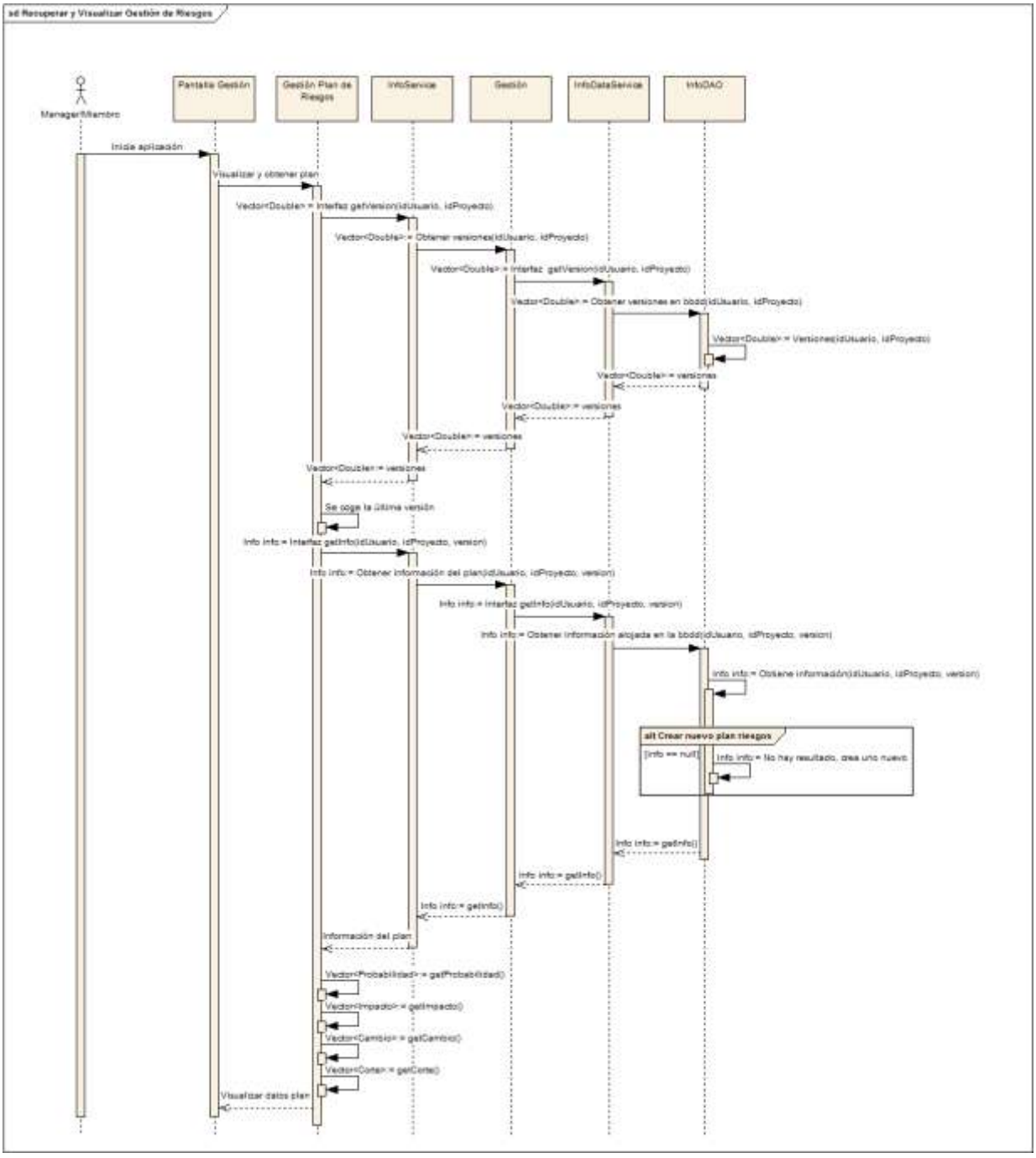


Figura 6.23. Diagrama de Interacción Recuperar y Visualizar Datos Riesgos

6.3.6 Caso de Uso 4.1

6.3.6.1 Crear/Almacenar Gestión de Riesgos

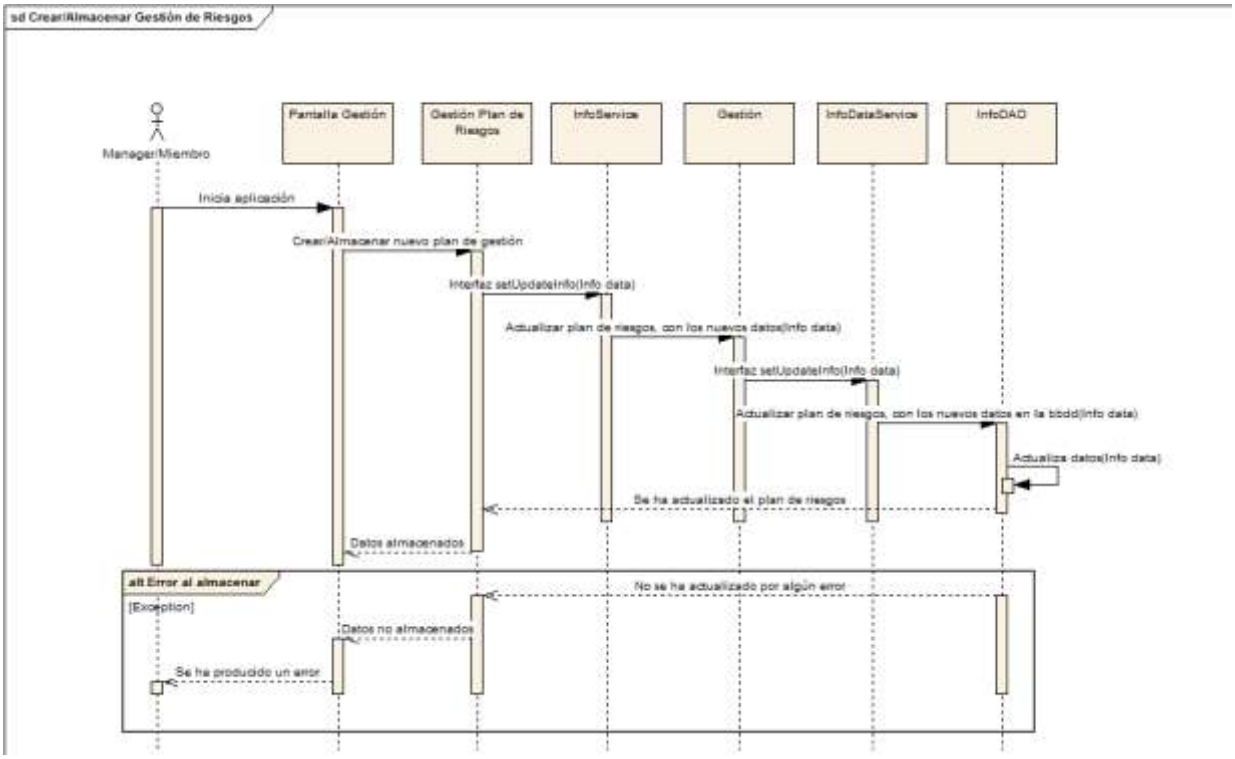


Figura 6.24. Crear/Almacenar Gestión de Riesgos

6.3.7 Caso de Uso 4.2

6.3.7.1 Crear Documentación PDF de la Gestión de Riesgos

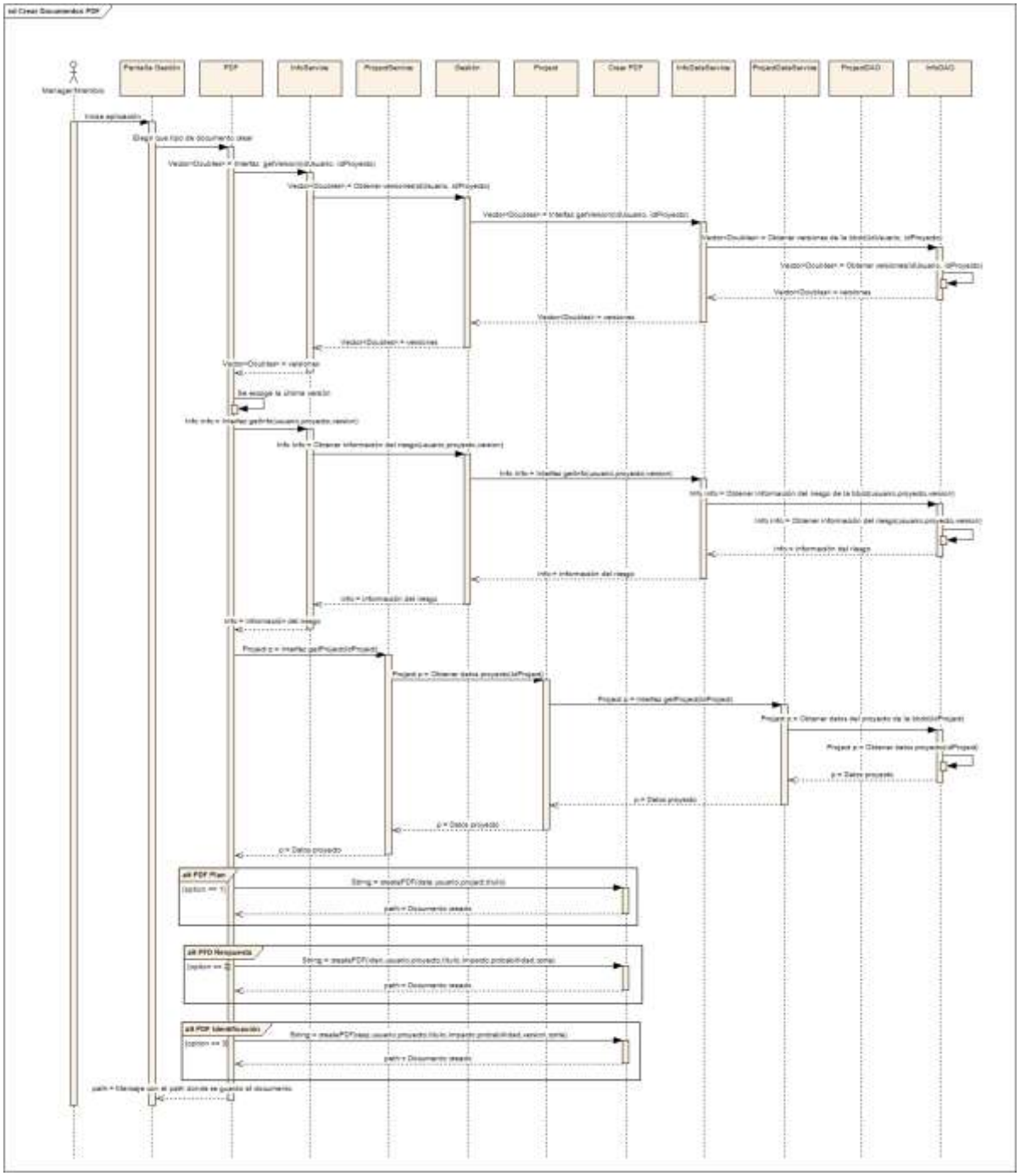


Figura 6.25. Diagrama de Interacción Crear Documentos PDF

6.4 Diagramas de Actividades

6.4.1 Login

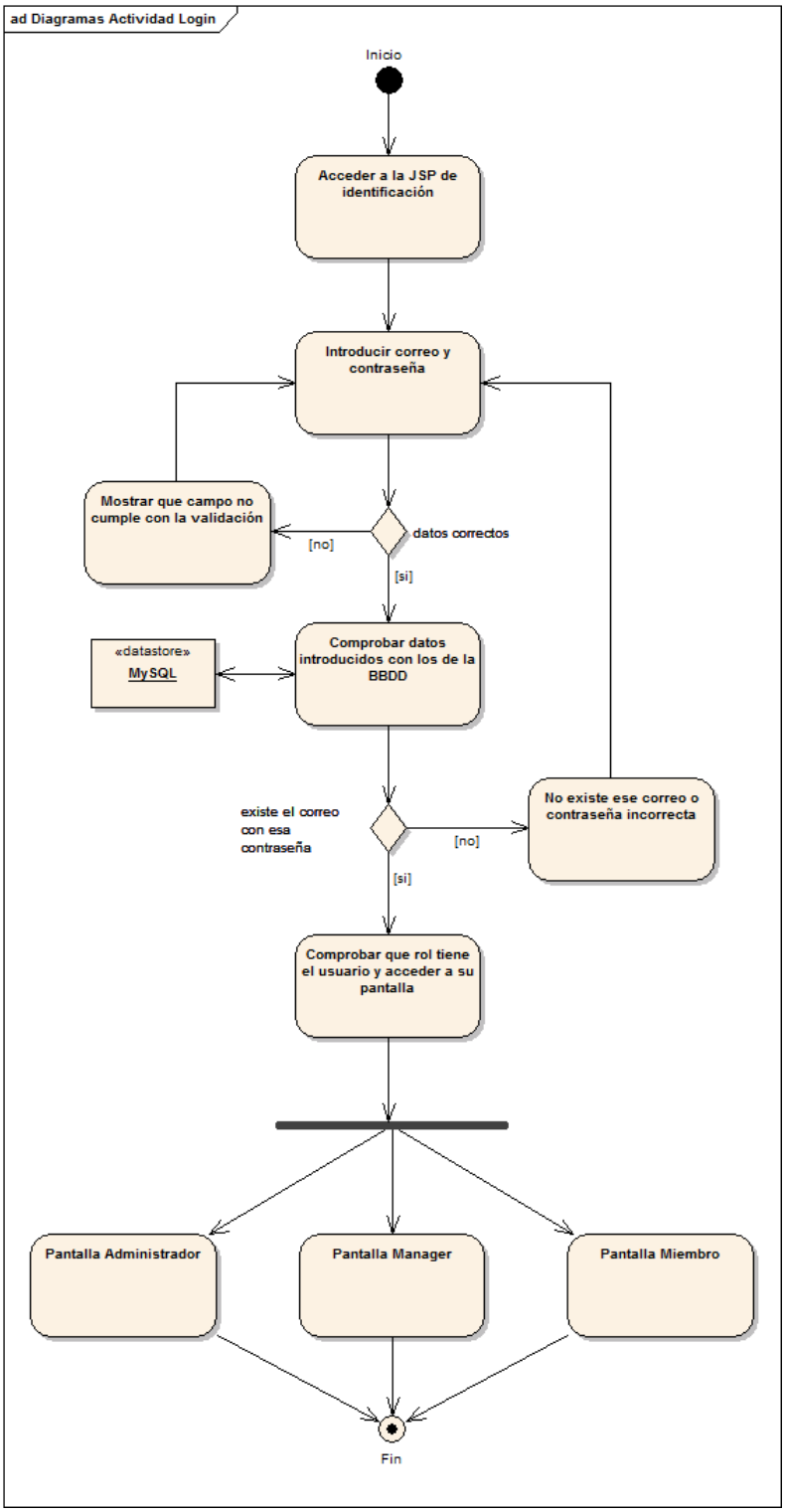


Figura 6.26. Diagrama Actividad Login

6.4.2 Registrar Manager y Proyecto

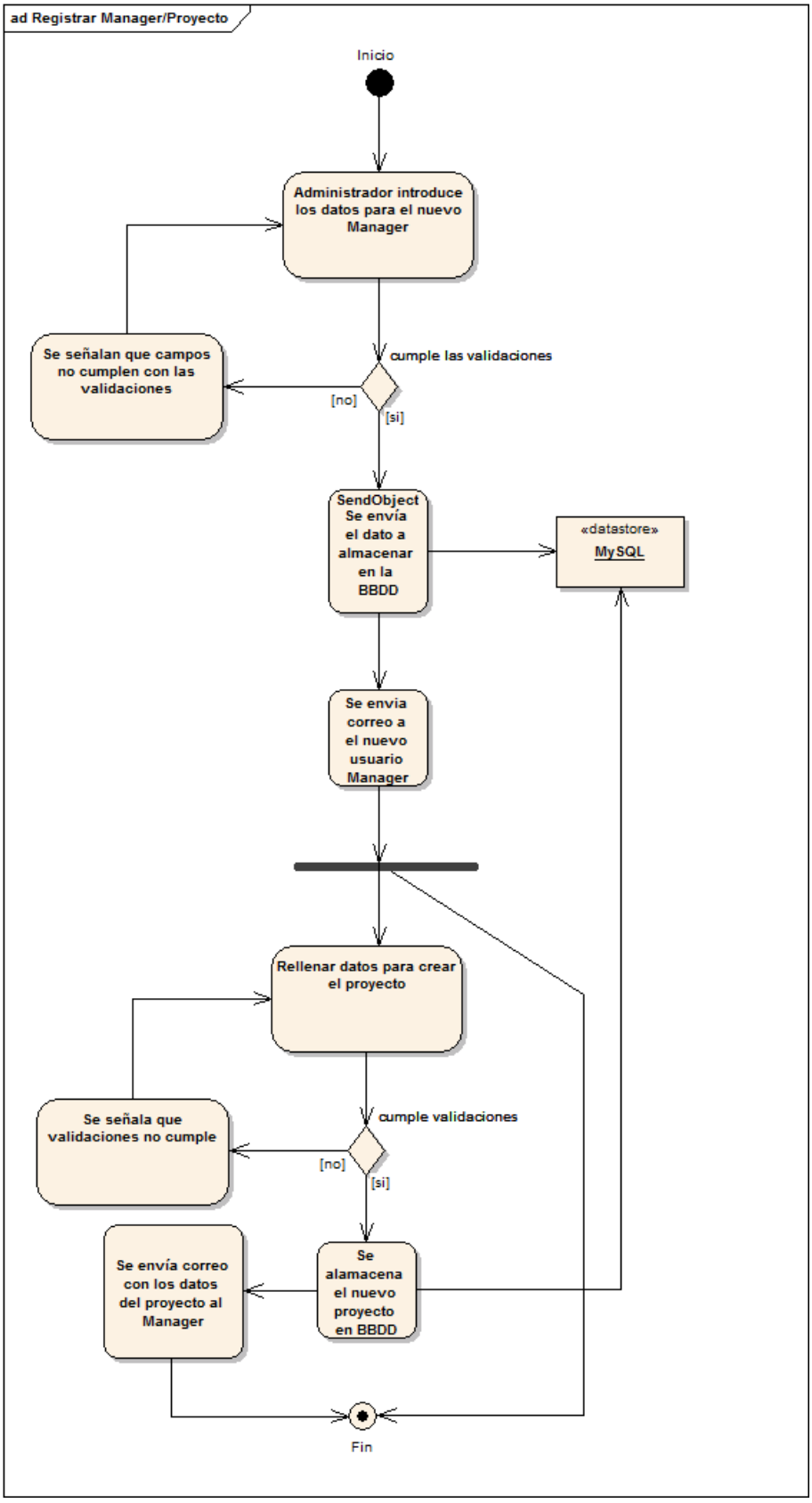


Figura 6.27. Diagrama de Actividad Registrar Manager/Proyecto

6.4.3 Registrar Miembro

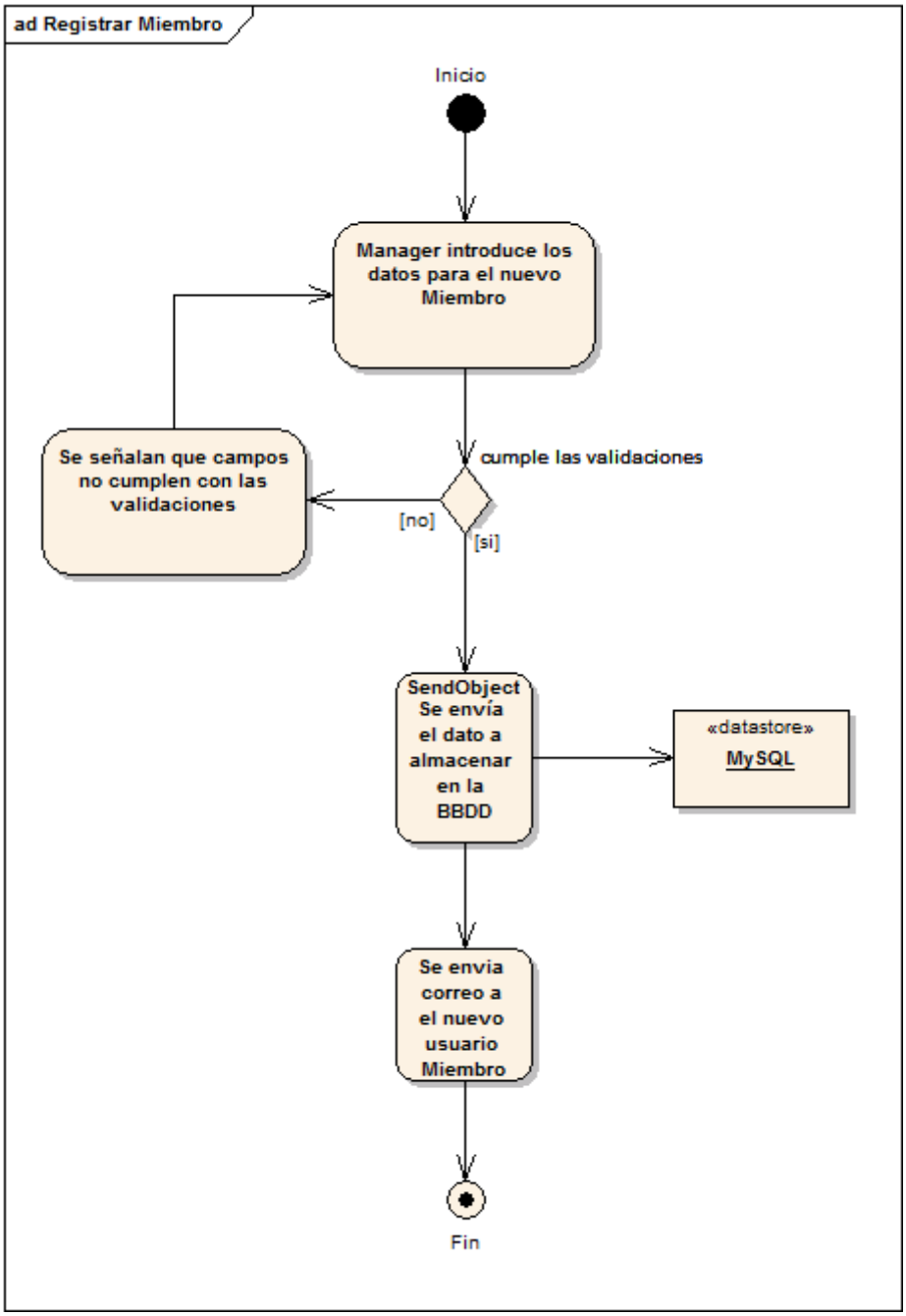


Figura 6.28. Diagrama de Actividad Registrar Miembro

6.4.4 Crear/Almacenar Gestión de Riesgos

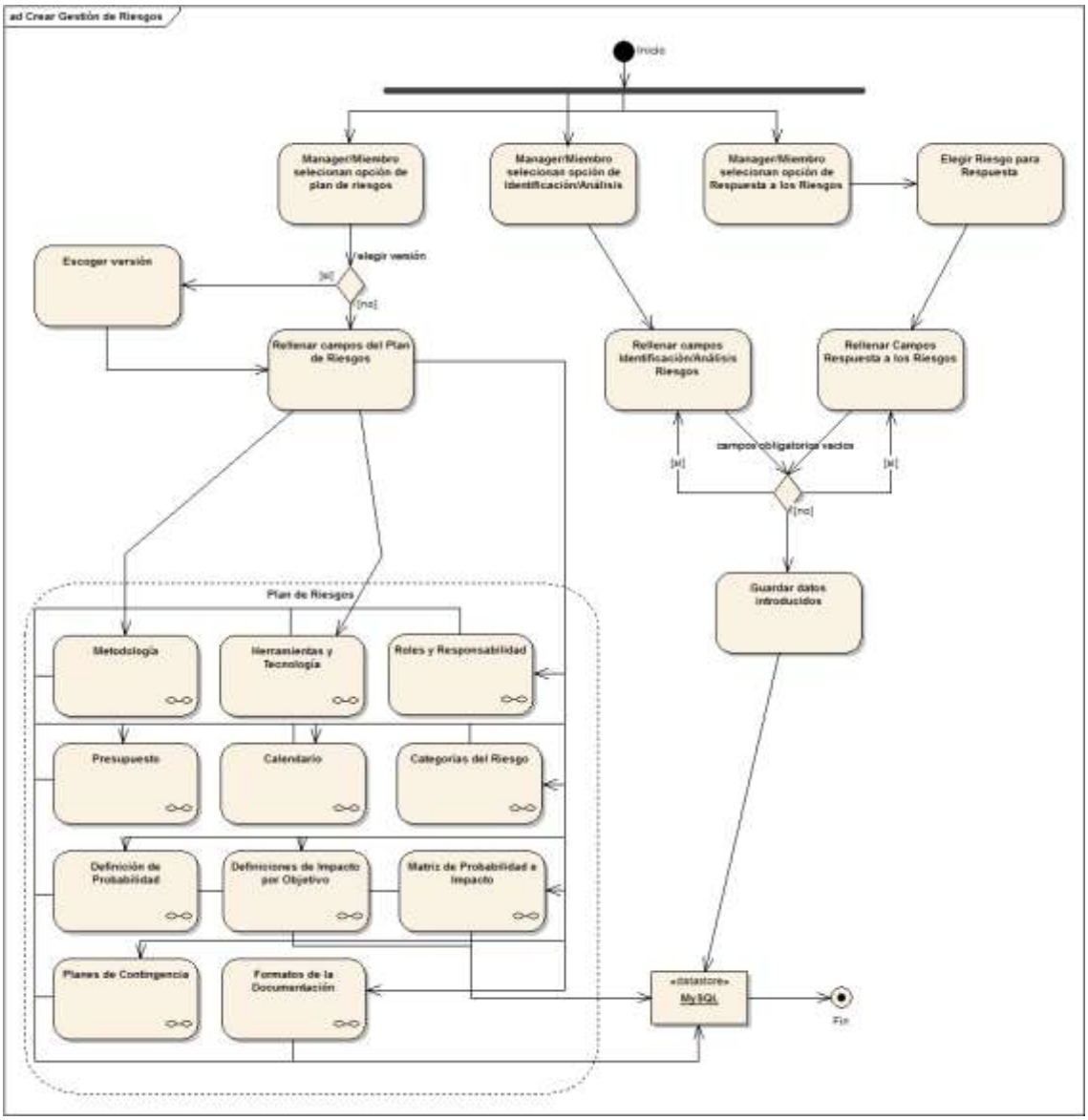


Figura 6.29. Diagrama de Actividad Crear/Almacenar Gestión de Riesgos

6.5 Diseño de la Base de Datos

6.5.1 Descripción del SGBD Usado

El sistema que se ha utilizado para la gestión de la base de datos, es MySQL, el cual es un sistema relacional, multihilo y multiusuario. Tiene licencia GNU GPL, por lo que es de distribución gratuita, a excepción de aquellas empresas que quieran incorporarlo en productos privativos, entonces deberán comprar a la empresa que es en estos momentos es Oracle Corporation, una licencia específica que les permita ese uso.

- MySQL es un sistema de gestión de bases de datos, tanto para tratar datos sencillos y simples como, para tratar una gran cantidad de datos, los sistemas de gestión de bases de datos juegan un papel muy importante en la computación, como aplicaciones autónomas o como parte de otras aplicaciones.

- MySQL es además un sistema de gestión de bases de datos relacionales, ya que almacena datos en tablas separadas en lugar de poner todos los datos en un gran almacén. Esto añade velocidad y flexibilidad. El lenguaje utilizado para acceder a la base de datos es el SQL y está definido por el estándar ANSI/ISO SQL. El estándar SQL ha evolucionado desde 1986 y existen varias versiones. Está base de datos utiliza el "SQL:2003" que se refiere a la versión actual del estándar.

- El servidor de base de datos MySQL es muy rápido, fiable y fácil de usar.

MySQL Server se desarrolló originalmente para tratar grandes bases de datos mucho más rápido que soluciones existentes y ha sido usado con éxito en entornos de producción de alto rendimiento durante varios años. MySQL Server ofrece hoy en día una gran cantidad de funciones. Su conectividad, velocidad, y seguridad hacen de MySQL Server altamente apropiado para acceder bases de datos en Internet

- MySQL Server trabaja en entornos cliente/servidor o incrustados

El software de bases de datos MySQL es un sistema cliente/servidor que consiste en un servidor SQL multi-threaded que trabaja con diferentes bakends[4], programas y bibliotecas cliente, herramientas administrativas y un amplio abanico de interfaces de programación para aplicaciones (APIs).

La versión utilizada para el proyecto se correspondía con la distribución 5.6 pero se han tenido problemas a la hora de subirla a un servidor, como es OpenShift, quienes utilizan una versión anterior, la 5.5 y ha creado algunos problemas que se habían solucionado en la versión siguiente como es el tema del número de timestamp que pueden aparecer en la misma tabla, en la versión 5.5 sólo puede haber una columna TIMESTAMP con CURRENT_TIMESTAMP en DEFAULT u ON UPDATE, yo tengo dos, una para la fecha de creación de los elementos y otra para la modificación. Al final buscando por Internet, se puede hacer un apaño, poniendo uno

[4] Hace referencia al estado final de un proceso.

de los dos valores, el valor inicial de 0, y a la hora de insertar los campos en la tabla, agregar dichos registros con valor NULL, en ese campo en concreto, la funcionalidad final es la misma.

Se ha utilizado para ello el patrón DAO, un patrón de diseño, encargado de encapsular la forma de acceder a la fuente de datos.

Se trata que el software de la capa de negocio se centre en los datos que necesita y no en cómo se realiza el acceso a estos datos. Por consiguiente una migración en el acceso a la forma de datos, supondría una minimización de los cambios en la aplicación.

Inconveniente, cuando se añaden DAOs a una aplicación, la complejidad adicional de usar otra capa de persistencia incrementa la cantidad de código ejecutado, por lo requiere en la mayoría de los casos mucho más tiempo.

A continuación una lista con las clases empleadas:

Nombre de la clase	Descripción
Interfaces	Estas interfaces no deben tener nada que las relacione con una base de datos ni cualquier otra cosa específica del medio de almacenamiento que se está usando. Se ha utilizado el patrón Façade para la separación entre la capa de negocio y la de persistencia.
InfoDataService	Contiene todas las operaciones de los métodos abstractos que especifica qué se debe hacer para obtener y establecer los datos de la gestión de riesgos.
ProjectDataService	Contiene todas las operaciones de los métodos abstractos que especifica qué se debe hacer para obtener y establecer los datos de los proyectos.
UserDataService	Contiene todas las operaciones de los métodos abstractos que especifica qué se debe hacer para obtener y establecer los datos de los usuarios.
Implementaciones	**Para las implementaciones de las interfaces de persistencia, se utiliza como conexión a la base de datos el conector JDBC.**
InfoDAO	Esta es la clase donde se produce la implementación de los métodos de la interfaz, en ella se describe la lógica del comportamiento de los métodos para la obtención y el establecimiento de los datos de la gestión de riesgos.
ProjectDAO	Esta es la clase donde se produce la implementación de los métodos de la interfaz, en ella se describe la lógica del comportamiento de los métodos para la obtención y el establecimiento de los datos

	de los proyectos.
UserDAO	Esta es la clase donde se produce la implementación de los métodos de la interfaz, en ella se describe la lógica del comportamiento de los métodos para la obtención y el establecimiento de los datos de los usuarios.
JDBC	Es la clase donde definimos las opciones de conectividad con la base de datos a través de un archivo properties.

6.5.2 Integración del SGBD en Nuestro Sistema

Para la integración de Sistema de Gestión de Base de Datos en el sistema se utilizará el paquete "persistance" en JAVA, tanto para las interfaces como para las implementaciones, además se utilizará la clase JDBC, para la conectividad a la base de datos, por medio de un archivo *porperties* donde vienen definidas las variables para la conexión.

Lo primero que se necesita para conectarnos con la base de datos es un Driver (o Connector). Para este caso, ya que se utiliza MySQL, se tendrá que utilizar un driver en concreto el que se ha descargado de http://dev.mysql.com/downloads/connector/j/. Una vez descargado, se desempaqueta y se recoge el conector mysql-connector-java-"version"-bin.jar que viene dentro, se hace accesible al proyecto, añadiéndolo al IDE, como una librería externa de tipo "jar".

Una vez que java tiene el *jar* accesible y sabe dónde encontrarlo, lo primero es conectarse con la base de datos y probar su funcionamiento para que sea el correcto, primero se prueba localmente y después se utiliza el servidor de bases de datos que ofrece OpenShift, el cual incluye entre otras cosas una base de datos MySQL 5.5 y la opción de agregar **phpMyAdmin**, que es un programa para gestionar la administración de MySQL a través de una página web. Con este programa se hace más fácil el tema de crear y eliminar Bases de Datos, tablas, borrar, editar y añadir campos a esas tablas, podemos probar a ejecutar cualquier sentencia SQL de nuestras tablas y todo lo que se pueda hacer desde una consola de MySQL podemos realizarlo online sobre nuestra base de datos del servidor. Este programa está bajo licencia GPL la Versión 2.

6.5.3 Diagrama E-R

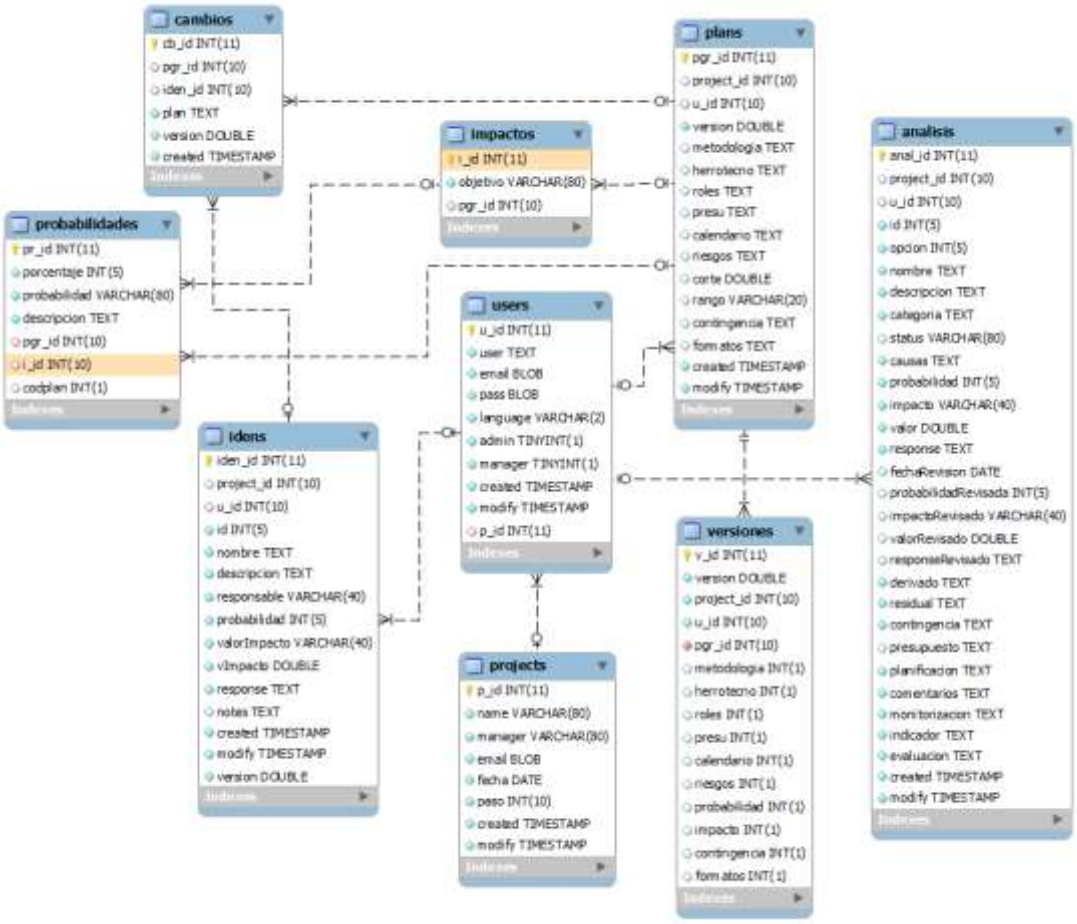


Figura 6.30. Diagrama E-R

6.6 Especificación Técnica del Plan de Pruebas

Procesador de Sistema	Core i7 de Intel de doble núcleo
Velocidad del Procesador	2,9 GHz (Turbo Boost hasta 3,6 GHz)
Memoria de Sistema (RAM)	8 GB SDRAM DDR3 a 1.600 MHz - 2 x 4 GB
Sistema Operativo	Ubuntu con VirtualBox
Software Existente	Cliente de correo estándar (Gmail) Eclipse Java EE IDE for Web Developers. Version: Luna Service Release 1 (4.4.1) Navegador web popular (Chrome, Firefox)

6.6.1 Pruebas Unitarias

Como en todo proyecto, una de las partes fundamentales para que todo funcione tal como se espera, se deben realizar pruebas cada cierto tiempo, según se va avanzando en el proceso, para verificar que lo que uno intenta hacer funciona tal cual lo ha pensado y creado, la mayoría de ellas son las típicas pruebas de, ensayo y error, es un método heurístico, que consiste en probar una alternativa y verificar si funciona y si no es así seguir probando hasta encontrar la solución, se centra sobre todo en problemas específicos para lograr una solución, pero es una técnica no muy adecuada al no ser muy optima y al ser muy costosa.

Para ello se ha dirigido el plan de pruebas sobre herramientas destinadas a ello, a probar que el código hace correctamente lo que tiene que hacer, para ello, se han utilizado unas bibliotecas que se utilizan en programación para hacer pruebas unitarias de aplicaciones Java, la biblioteca JUnit. Además de funcionar para hacer las pruebas unitarias, JUnit es también un medio de controlar las pruebas de regresión, necesarias cuando una parte del código ha sido modificado y se desea ver si el nuevo código cumple con los requerimientos anteriores.

Las pruebas se han realizado, para los puntos anteriormente hablados en el aparatado 5.7 del Análisis, Especificación del Plan de Pruebas.

Las clases que probaremos serán los objetos de modelo, para la asociación de los elementos entre ellos:

```
    private User usuario;
    private Project proyecto;
    private Info info;
    private Probabilidad probabilidad;
    private Vector<Probabilidad> vp;
    private Impacto impacto;
    private Vector<Impacto> vi;
    private Cambio cambio;
    private Vector<Cambio> vc;
    private Iden iden;
    private Respuesta resp;
```

En las pruebas de persistencia, utilizaremos las tres clases DAO:

```
    private final UserDAO userS = new UserDAO();
    private final ProjectDAO projectS = new ProjectDAO();
    private final InfoDAO infoS = new InfoDAO();
```

Y para la prueba de creación de documentos PDF, nos crearemos un test para probar diferentes funciones de la biblioteca ***IText*** que necesitaremos cuando se creen finalmente los documentos.

Todas las pruebas se irán realizando cada vez que se cree una nuevo proceso en la aplicación, para probar su funcionamiento individual, para luego una vez que se tenga toda la documentación probarlo en conjunto.

Cuando se realice una prueba se observara el comportamiento de la misma, siguiendo unas pautas que digan cómo se debe hacer y qué resultado debe salir, para confirmar el OK de la prueba, en caso negativo caso KO, se buscaría el fallo y el porqué se está haciendo así, se comprobaría si es lo correcto y si no se cambiaría debugeando hasta conseguir la finalización marcada en el inicio de la prueba.

6.6.2 Pruebas de Usabilidad y Accesibilidad

Las pruebas de usabilidad tratan de evaluar la aplicación en su funcionamiento real. Debemos realizar un conjunto de pruebas para verificar que las distintas partes del programa se pueden usar adecuadamente. Los objetivos son:

- Mejorar la calidad de la interacción de los usuarios con nuestra aplicación.
- Reducir el tiempo necesario para hacer las distintas tareas en la aplicación y de esta forma aumentar la productividad.

Para realizar estas pruebas utilizaremos lo siguiente:

- **Usuarios**: Cada usuario tendrá un perfil de uso distinto y que tendremos en cuenta a la hora de hacer las pruebas. Para efectuar las pruebas usaremos entorno a 3 usuarios. Se intentara tener constancia si los usuarios lo permiten de la profesión y la edad de los mismos.
- **Lugar de realización**: Al ser un archivo de texto digital, se puede enviar por correo electrónico y al probarlo sobre una aplicación web, que se encuentra subida a un servidor, se puede conectar a la dirección del hosting desde su casa, rellenar el cuestionario y devolverlo.
- **Metodología**: La metodología será un cuestionario, el cual irán una serie de preguntas sencillas y cortas sobre el uso y la accesibilidad de la aplicación web, a las que los usuarios deberán contestar, sus respuestas determinan la satisfacción del cliente con el producto final.
  Una vez in iniciada la prueba su única ayuda sería el sistema de ayuda que puede brindarle la aplicación que está contiene. Después de realizar las tareas se les daría unos

cuestionarios para que rellenasen, uno de carácter general para determinar el tipo de usuario y su nivel de conocimiento informático, otro de actividades guiadas, que se puedan hacer con la aplicación para que nuestros usuarios las hagan y comenten los problemas y dificultades que según su opinión presenta la aplicación, que deberían contestar puntuando de 1 a 3, siendo 1 el valor más bajo referido a la simplicidad y 3 el valor más alto, para un grado alto de complejidad y por último uno de observaciones para que el usuario aporte todo lo que considere oportuno.

6.6.2.1 Diseño de Cuestionarios

6.6.2.1.1 Cuestionario de Evaluación

Se elaborara unos cuestionarios para que los usuarios lo hagan y determinar así distintos aspectos del mismo y de su interacción con la aplicación. Los puntos a tocar son:

- **1º**: **Preguntas de carácter general** a través de las cuales se determine el tipo de usuario y su nivel de conocimiento informático.
- **2º**: **Actividades guiadas** para hacer con nuestra aplicación.
- **3º**: **Batería de preguntas cortas** con los distintos aspectos de la aplicación que se pretendan evaluar.
- **4º**: **Observaciones**, para que el usuario aporte todo lo que considere oportuno de nuestra aplicación.

6.6.2.1.2 Cuestionario para el Responsable de las Pruebas

Además, se desarrollará un cuestionario para que yo, el responsable de las pruebas, pueda anotar distintos aspectos que observe durante la realización de las pruebas.

6.6.2.2 Actividades de las Pruebas de Usabilidad

6.6.2.2.1 Preguntas de carácter general

¿Usa un ordenador frecuentemente?
1. Todos los días 2. Varias veces a la semana 3. Ocasionalmente 4. Nunca o casi nunca
¿Qué tipo de actividades realiza con el ordenador?
1. Es parte de mi trabajo o profesión 2. Lo uso básicamente para ocio 3. Solo empleo aplicaciones estilo Office 4. Únicamente leo el correo y navego ocasionalmente
¿Ha usado alguna vez software como el de esta prueba?
1. Sí, he empleado software similar 2. No, aunque si empleo otros programas que me ayudan a realizar tareas similares 3. No, nunca

¿Qué busca Vd. Principalmente en un programa?
1. Que sea fácil de usar 2. Que sea intuitivo 3. Que sea rápido 4. Que tenga todas las funciones necesarias

6.6.2.2.2 Actividades guiadas

Actividad	1…5	Observaciones
Loguearse		
Utilización de las teclas de acceso rápido		
Utilización de los atajos		
Ayudan los mensajes de estado		
Uso de la ayuda en línea		
Salir de la aplicación		
Administrador		
Crear Manager/Proyecto		
Acceso a la visualización de Proyectos/Managers		
Actualización de datos de acceso		
Eliminación/Actualización de Proyectos/Managers		
Manager		
Crear Miembros		
Manager/Miembro		
Actualización de datos de acceso		
Crear/Almacenar Plan de Riesgos		
Crear/Almacenar Identificación/Análisis de Riesgos		
Crear/Almacenar Respuesta a los Riesgos		
Crear Documentación de la Gestión de Riesgos		

6.6.2.2.3 Preguntas Cortas sobre la Aplicación y Observaciones

Facilidad de Uso	Siempre	Frecuentemente	Ocasionalmente	Nunca
¿Sabe donde está dentro de la aplicación?				
¿Existe ayuda para las funciones en caso de que tenga dudas?				

¿Le resulta sencillo el uso de la aplicación?				
Funcionalidad	**Siempre**	**Frecuentemente**	**Ocasionalmente**	**Nunca**
¿Funciona cada tarea como Vd. Espera?				
¿El tiempo de respuesta de la aplicación es muy grande?				
Calidad del Interfaz				
Aspectos gráficos	**Muy Adecuado**	**Adecuado**	**Poco Adecuado**	**Nada Adecuado**
El tipo y tamaño de letra es				
Los iconos e imágenes usados son				
Los colores empleados son				
Diseño de la Interfaz		**Si**	**No**	**A veces**
¿Le resulta fácil de usar?				
¿El diseño de las pantallas es claro y atractivo?				
¿Cree que el programa está bien estructurado?				
Observaciones				
Cualquier comentario del usuario				

6.6.2.2.4 Cuestionario para el Responsable de las Pruebas

Aspecto Observado	**Notas**
El usuario comienza a trabajar de forma rápida por las tareas	
Tiempo en realizar cada tarea	
Errores leves cometidos	
Errores graves cometidos	
El usuario encuentra alguna dificultad en alguna actividad	

6.6.2.3 Pruebas de Accesibilidad

El grueso de las **pruebas de accesibilidad** será descrito en la parte de desarrollo de las pruebas. A nivel de diseño simplemente puede indicarse qué tipo de pruebas van a realizarse para el programa concreto, que normas *WCAG* van a seguirse u otras consideraciones generales acerca de las mismas, dentro de las posibles vías de actuación. *WAI* propone básicamente tres tipos de revisiones:

- **Revisión preliminar**: Identifica rápidamente problemas de accesibilidad de un sitio *Web*. Todos los procesos de esta revisión también tienen lugar en la siguiente revisión.
- **Evaluación de conformidad**: Permite indicar si un sitio web cumple con los estándares de accesibilidad (Ej.: *WCAG*). Puede encontrarse más información en: www.w3.org/WAI/eval, aunque en esta plantilla se intentarán dar pautas para hacer una evaluación adecuada.
- **Otras evaluaciones**: *WAI* también proporciona otra serie de documentos para evaluar aplicaciones en contextos específicos y para implicar a usuarios discapacitados en la

evaluación. En el primer caso se tratan de documentos complementarios a los anteriores que describen consideraciones para la evaluación de sitios grandes y complejos durante el proceso de desarrollo y mantenimiento, o bien para sitios existentes cuyas páginas son generadas dinámicamente.

6.6.3 Pruebas de Rendimiento

El sistema desarrollado consumirá una determinada cantidad de recursos (memoria y tiempo de proceso) que se procurara que sea el menor posible. Por ello, mediremos la aplicación para ver cuántos recursos consume y si se debe intentar eliminar posibles cuellos de botella en el rendimiento que se puedan presentar en uno o varios subsistemas de la misma.

Las pruebas sobre rendimiento, se utilizará la aplicación JMeter de Apache, la cual permite medir como responde el sistema a un gran número de usuarios y/o a una gran carga de volumen de datos.

Para ello se analizan entre otras cosas, el acceso a la base de datos, una vez dentro de ella se hacen diferentes operaciones sobre las tablas de nuestra base de datos.

Ejemplo, creada expresamente para testearla. Se utilizaran unos 10 usuarios con un periodo de subida de 10 segundos, dentro de un bucle infinito que aproxime cuanto tiempo tarda en completar las acciones

También se probará la carga sobre el Servicio Web de Hosting, se comenzará con peticiones de acceso a la aplicación, concretamente al login que es por donde más tráfico se producirá, se probará también los recorridos por todas las pestañas y el cierre de sesión con el usuario de tipo Miembro en el Servicio Web creado, para ver la duración del tiempo que emplea. Se utilizarán también 10 usuarios con un periodo de subida de 10 segundos, dentro de un bucle infinito.

Capítulo 7. Implementación del Sistema

7.1 Estándares y Normas Seguidos

Para la estructura y la documentación de Java, se ha seguido **el Code Conventions for the Java TM Programming Language (JCC)**, es un documento que contiene los estándares de codificación del lenguaje Java presentado por Sun Microsystems. Controla la codificación tanto de, clases, interfaces, métodos, variables y constantes. El documento, se encuentra en la página oficial de Oracle y se puede consultar desde la siguiente dirección[5].

Para la documentación del código Java, se ha utilizado la herramienta que Oracle, pone a la disposición de los programadores y que está incluida en la aplicación de Eclipse, llamada **Javadoc[6]**.

Para el framework de **MVC** con **Struts2** junto a los 3 tipos de **core** como son los *interceptors, actions* y los *results* se ha seguido la documentación de Apache[7]. Para el framework de **Spring** que se utiliza en conjunto con Struts2[8]. Las etiquetas de las **JSP**, las cuales se utiliza **JavaServer Pages Standard Tag Library (JSTL)**[9] que encapsula las etiquetas simples. Además de la utilización del contenido de **HTML5**[10] y **CSS3**[11] que cumplan con las normas del **W3C[12].**

Para la escritura de las sentencias de MySQL, se ha utilizado el estándar "**SQL:2008[13]**", que es la sexta revisión de la **ISO** y **ANSI**, para el estándar de **SQLN**, que viene integrado en la aplicación de MySQL. Para la encriptación de algunos datos sensibles en la base de datos, se ha utilizado **AES[14]**, que es un esquema de cifrado por bloques adoptado como un estándar de cifrado para criptografía simétrica.

Las normas para la creación de documentos PDF, se ha utilizado lo que dicta el documento gratuito de los creadores de la librería iText[15], mientras que para el envío de correos se ha utilizado el Javadoc[16] que aporta Oracle propietaria de la librería JavaMail, para el envío de correos desde Java.

[5] http://www.oracle.com/technetwork/java/javase/documentation/codeconvtoc-136057.html
[6] http://www.oracle.com/technetwork/java/javase/documentation/index-jsp-135444.html
[7] http://struts.apache.org/docs/guides.html
[8] http://struts.apache.org/docs/spring-and-struts-2.html
[9] http://www.oracle.com/technetwork/java/index-jsp-135995.html
[10] http://www.w3.org/TR/html5/
[11] http://www.w3.org/Style/CSS/current-work
[12] http://www.w3.org/standards/webofservices/description
[13] http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=38646
[14] http://www.movable-type.co.uk/scripts/aes.html
[15] https://leanpub.com/itext_pdfabc/
[16] https://javamail.java.net/nonav/docs/api/

La gestión de riesgos que incluye para este proyecto:

- Plan de Gestión de Riesgos
- Identificación de Riesgos
- Análisis de Riesgos
- Respuesta de los Riesgos

Se ha utilizado el libro de la "***Guía de los Fundamentos para la Dirección de Proyectos (Guía del PMBOK)***" en su edición 4 en versión para castellano, donde el capítulo que hace referencia a los temas tratados anteriormente citados corresponde al El **Capítulo 11, Gestión de los Riesgos del Proyecto**, el cual describe los procesos involucrados en la identificación, análisis y control de los riesgos para el proyecto. También en su versión inglesa que es una versión actualizada "A Guide to the Project Management Body of Knowledge" (PMBOK Guide), fifth Edition, el capítulo utilizado para el proyecto corresponde también al 11.

7.2 Lenguajes de Programación

7.2.1 Java

Es un lenguaje de programación orientado a objetos, desarrollado por Sun Microsystems a principios de los años 90.

El lenguaje Java fue creado teniendo en cuenta cinco objetivos principales:

1) Debería usar la metodología de la programación orientada a objetos.
2) Debería permitir la ejecución de un mismo programa en múltiples sistemas operativos (Independencia de la plataforma).
3) Debería incluir por defecto soporte para trabajo en red.
4) Debería diseñarse para ejecutar código en sistemas remotos de forma segura.
5) Debería ser fácil de usar y tomar lo mejor de otros lenguajes orientados a objetos, como C++.

Será el lenguaje con el que se programe nuestra aplicación, algunas herramientas dentro de la librería de Java como los anteriormente mencionados Javadoc y JUnit, para el proyecto se utiliza la versión de Java 7 Update 45.

7.2.2 JavaScript

Es un lenguaje de programación dinámica, se utiliza en implementaciones que permiten scripts del lado del cliente para interactuar con el usuario, controlar el navegador, se comunican de forma asíncrona, y permite alterar el contenido del documento que se está mostrando.

El lenguaje Javascript es ***opensource***, por lo cualquier persona puede utilizarlo, en el proyecto se utiliza para interactuar con el código de HTML. Básicamente se utiliza para crear acciones en páginas web.

La versión dependerá del navegador, que es quien ejecuta este tipo de lenguaje, la última que hay para al menos FireFox es la 1.8.5. Utilizamos también la librería de JQuery, para interactuar también con los documentos de HTML, la versión utilizada es la 1.4.2.

7.2.3 HTML5

HTML5 es un lenguaje de marcas, que se utiliza para estructurar y presentar contenido para la Web, en el proyecto se utiliza para la elaboración de las JSPs. La versión como índica el nombre es la 5. Su distribución es de código abierto.

7.2.4 CSS3

CSS es un lenguaje de hojas de estilo, usado para describir el aspecto y el formato de un documento escrito en un lenguaje de marcas para el caso que nos ocupa, básicamente para dar estilo a nuestras páginas JSPs escritas con HTML5, con JavaScript, CSS es una tecnología fundamental utilizada por la mayoría de los sitios web para crear páginas web visualmente atractivas, interfaces de usuario para aplicaciones web e interfaces de usuario para muchas aplicaciones móviles.

CSS está diseñado principalmente para permitir la separación de contenido del documento de presentación de documentos, incluyendo elementos tales como el diseño, colores y tipos de letra. La versión utilizada es no es el 3 que índica el nombre, eso índica el nivel o perfil, cada nivel o perfil tiene asignadas unas especificaciones que debería cumplirse.

7.2.5 SQL

El **SQL** es el lenguaje declarativo que se utilizará para acceder a la bases de datos relacional, que permite especificar diversos tipos de operaciones. Una de sus características es el manejo del álgebra y el cálculo relacional, permitiendo efectuar consultas con el fin de recuperar -de una forma sencilla información de interés de una base de datos, así como también hacer cambios sobre ella.

La versión que se utiliza será la que viene dada en el sistema de gestión de base de datos MySQL que es, la versión de 2008, llamada **SQL:2008**, esta es la sexta revisión de los estándares ISO y ANSI, para el lenguaje de consultas SQL.

Como novedad con respecto a la anterior revisión, en está, se permite el uso de la clausula “ORDER BY”, fuera de las definiciones de los cursores, así como, incluir los “trigguers” del tipo “INSTEAD OF” y la posibilidad de añadir la sentencia “TRUNCATE” a l tabla.

7.2.6 XML

El XML proviene de un lenguaje que inventó IBM allá por los años 70. El lenguaje de IBM se llama GML (General Markup Language) y surgió por la necesidad que tenían en la empresa de almacenar grandes cantidades de información de temas diversos.

XML es un Lenguaje de Etiquetado Extensible muy simple, pero estricto que juega un papel fundamental en el intercambio de una gran variedad de datos. Es un lenguaje muy similar a HTML pero su función principal es describir datos y no mostrarlos como es el caso de HTML. XML es un formato que permite la lectura de datos a través de diferentes aplicaciones, sirve entre otras cosas para estructurar, almacenar e intercambiar información.

Se utiliza básicamente para la generación de configuraciones de Struts2 y de Spring. La versión utilizada es la 1.0, y es de distribución abierta.

7.3 Herramientas y Programas Usados para el Desarrollo

7.3.1 Desarrollo

7.3.1.1 Eclipse

Es un entorno de desarrollo integrado de código abierto y multiplataforma, cuya finalidad es la de poder servir al usuario para el desarrollo de lo que el proyecto llama "Aplicaciones de Cliente Enriquecido".

Está escrito principalmente en Java y puede ser utilizado para desarrollar aplicaciones en varios lenguajes de programación, aunque principalmente está pensado para Java. Una gran ventaja de Eclipse, frente a otras soluciones, es la posibilidad incrementar el número de características, mediante los plugins, de esta forma es posible añadirle soporte para lenguajes de programación para los que no fue diseñado expresamente o incluso incluir nuevas funcionalidades.

Eclipse fue desarrollado originalmente por IBM como el sucesor de VisualAge. El desarrollo actual de Eclipse se encuentra en manos de la Fundación Eclipse, una organización independiente sin ánimo de lucro que fomenta una comunidad de código abierto y un conjunto de productos complementarios, capacidades y servicios.

Para el proyecto se ha utilizado la versión Java EE IDE for Web Developers, concretamente el paquete de Luna Service Release 1 (4.4.1).

http://www.eclipse.org/

7.3.1.2 JDK

Es un software que provee de herramientas de desarrollo para la creación de programas en Java. Es un producto de Oracle Corporation dirigido a los desarrolladores de Java. Normalmente siempre viene con el JRE, que es un conjunto de utilidades que permite la ejecución de programas Java. Está conformado por una Máquina Virtual de Java o JVM, un conjunto de bibliotecas Java y otros componentes necesarios para que una aplicación escrita en lenguaje Java pueda ser ejecutada. El JRE actúa como un "intermediario" entre el sistema operativo y Java.

La versión utilizada es la 1.6.0 actualización 31

http://www.oracle.com/technetwork/java/javase/overview/index.html

https://blogs.oracle.com/stevenChan/entry/sun_jre_1_6_0

Algunos programas que se utilizaran del JDK, son:

- **Javadoc**, que se utiliza con el Eclipse, para crear la documentación sobre las clases escritas para el proyecto en Java.

Programas externos que no se incluyen ni en el JDK ni en el JRE:

- **Javamail** 1.4.7, es una expansión de Java que facilita el envío y recepción de e-mail desde código java.

http://www.oracle.com/technetwork/java/javamail/index.html

- **JDBC** 5.1.17, es una API que permite la ejecución de operaciones sobre bases de datos desde el lenguaje de programación Java.

http://www.oracle.com/technetwork/java/javase/tech/index-jsp-136101.html

7.3.1.3 Apache Tomcat

Es un servidor web con soporte de Servlets y JSPs. Tomcat no es un servidor de aplicaciones. El motor de Servlets de Tomcat a menudo se presenta en combinación con el servidor web Apache.

Tomcat puede funcionar como servidor web por sí mismo. En sus inicios existió la percepción de que el uso de Tomcat de forma autónoma era sólo recomendable para entornos de desarrollo y entornos con requisitos mínimos de velocidad y gestión de transacciones. Hoy en día ya no existe esa percepción y Tomcat es usado como servidor web autónomo en entornos con alto nivel de tráfico y alta disponibilidad.

Está escrito en Java y funciona en cualquier sistema operativo que disponga de la máquina virtual Java, fue desarrollado bajo el proyecto Jakarta en la Apache Software Foundation.

La versión utilizada para el proyecto se corresponde con la 7.0.54, y será el servidor web para el archivo ejecutable.

http://tomcat.apache.org/

7.3.1.4 MySQL

El software MySQL proporciona un servidor de base de datos SQL (Structured Query Language) muy rápido, multi-threaded, multi usuario y robusto. El servidor MySQL está diseñado para entornos de producción críticos, con alta carga de trabajo así como para integrarse en software para ser distribuido.

Los usuarios pueden elegir entre usar el software MySQL como un producto Open Source bajo los términos de la licencia GNU General Public License o pueden adquirir una licencia comercial estándar de MySQL AB.

En el proyecto para el archivo ejecutable se usará la versión 5.5, será como se describe, el sistema de gestión de base de datos SQL.

http://dev.mysql.com/

Otra aplicación que se ha utilizado de esta familia se corresponde con:

MySQL Workbench 5.1 OSS, que es una aplicación para el diseño visual de esquemas de bases de datos

http://www.mysql.com/products/workbench/

7.3.1.5 OpenShift

OpenShift es una plataforma de cloud computing como un producto del servicio de Red Hat, comenzando a utilizarse el 4 de mayo de 2011.

El software que ejecuta el servicio bajo el nombre de OpenShift Origen de código abierto, y está disponible en GitHub. Los desarrolladores pueden usar Git para desplegar aplicaciones web en diferentes idiomas en la plataforma.

Será donde subiremos la aplicación para acceder a ella desde cualquier ordenador, está plataforma, nos deja utilizar tanto un contenedor Tomcat 7 para desplegar en ella el archivo .war, que contiene el proyecto del servicio web, así como una base de datos MySQL y un sistema para administrar esta base de datos como es phpMyAdmin.

https://www.openshift.com/

7.3.2 Pruebas

7.3.2.1 JUnit

Es un conjunto de bibliotecas creadas por Erich Gamma y Kent Beck que son utilizadas en programación para hacer pruebas unitarias de aplicaciones Java.

JUnit es un conjunto de clases que permite realizar la ejecución de clases Java de manera controlada, para poder evaluar si el funcionamiento de cada uno de los métodos de la clase se comporta como se espera. Es decir, en función de algún valor de entrada se evalúa el valor de retorno esperado; si la clase cumple con la especificación, entonces JUnit devolverá que el método de la clase pasó exitosamente la prueba; en caso de que el valor esperado sea diferente al que regresó el método durante la ejecución, JUnit devolverá un fallo en el método correspondiente.

La versión que se utiliza es la 4.10, para la documentación de las pruebas unitarias de código Java.

http://www.junit.org/

7.3.2.2 JMeter

Es un proyecto de Apache Jakarta que puede ser utilizado como una herramienta de prueba de carga para analizar y medir el desempeño de una variedad de servicios, con énfasis en aplicaciones web.

Mientras que JMeter es clasificado como una herramienta de "generación de carga", no es una descripción completa de la herramienta. JMeter soporta aserciones para asegurarse que los datos recibidos son correctos, por cookies de hilos, configuración de variables y una variedad de reportes.

En el proyecto se ha utilizado la versión 2.6 y se ha utilizado como herramienta para medir la carga en la base de datos y Servicio Web.

http://jmeter.apache.org/

7.3.3 Visores, Documentación y Navegadores

7.3.3.1 Adobe Reader

Es una familia de programas informáticos desarrollados por Adobe Systems diseñados para visualizar, crear y modificar archivos con el formato Portable Document Format, más conocido como PDF, al comienzo se denomino, Acrobat Reader, a partir de 2010, se le paso a llamar como hoy lo se conoce.

Se encuentra en multitud de idiomas y para una gran cantidad de plataformas, tuvo un duro comienzo batallando con todos los demás visores de texto de diferentes formatos, hasta que en los fines de los años 90, el formato PDF se había convertido en el estándar de facto, y los otros formatos prácticamente desaparecieron.

En el proyecto se utilizará la versión XI del Adobe Reader, para visualizar tanto la documentación referente al proyecto como la creada por este.

http://www.adobe.com/products/reader.html

7.3.3.2 Mozilla Firefox

Es un navegador web libre y de código abierto descendiente de Mozilla Application Suite y desarrollado por la Fundación Mozilla. Es el segundo navegador más utilizado de Internet.

Se utiliza para visualizar páginas web, emplea el motor de renderizado Gecko, el cual implementa estándares web actuales además de otras funciones destinadas a anticipar probables adiciones a los estándares.

Es multiplataforma, su código fuente es software libre, publicado bajo una triple licencia GNU GPL, GNU LGPL o MPL.

Se utilizará la versión 34.0.5, para entre otras cosas acceder a la información en Internet, ver el estado de servidor Apache, acceder a los servicios que ofrece el servidor tanto propio como externo, los cuales ofrecen el acceso a nuestra aplicación Web.

https://www.mozilla.org/en-US/firefox/desktop/

7.3.3.3 Chrome

Es un navegador web desarrollado por Google es un software de código abierto, utiliza como motor de renderizado Blink (un *fork* de WebKit). Está disponible gratuitamente bajo condiciones de servicio específicas.

Es considerado como el navegador más rápido del mundo, sus objetivos principales son:

- Rápido
- Seguro
- Practico
- Estable
- Sentido minimalista

Puede ser instalado en casi cualquier sistema operativo y está disponible en 50 idiomas. La rapidez de Google Chrome se basa principalmente en la capacidad que tiene el navegador de procesar códigos de JavaScript, los cuales son los que se usan en la mayoría de las páginas web.

Al igual que el otro navegador, lo utilizamos para acceder a nuestra capa de presentación del proyecto, donde se encuentran las JSPs. La versión que se ha utilizado es la 39.0.2171.95 m.

7.3.3.4 Microsoft Office

Es una suite de oficina que abarca e interrelaciona aplicaciones de escritorio, servidores y servicios para los sistemas operativos Microsoft Windows y Mac OS X. Microsoft Office fue lanzado por Microsoft en 1989 y fue para que funcionara en un Apple Macintosh, más tarde saco una versión para Windows, en 1990.

La primera versión de Office contenía Microsoft Word, Microsoft Excel y Microsoft PowerPoint. Además, de una versión "Pro" (profesional) de Office que incluía Microsoft Access y Schedule Plus. Con el tiempo, las aplicaciones de Office han crecido sustancialmente y de forma más estrecha con características compartidas, como un corrector ortográfico común, la integración de datos OLE y el lenguaje de secuencias de comandos de Microsoft, Visual Basic para aplicaciones.

De todas esas aplicaciones que cuenta Microsoft Office, en el proyecto se utilizarán las siguientes de la misma versión la 12.0, del año 2007:

- **Microsoft Office Word**, es un procesador de textos, con el cual se hará la documentación del proyecto

http://office.microsoft.com/es-es/word/

- **Microsoft Office Excel**, es un programa de hoja o plantilla de cálculo, que se utiliza para los presupuestos

http://office.microsoft.com/es-es/excel/

- **Microsoft Office Project**, es un software de administración de proyectos, para asistir a administradores de proyectos en el desarrollo de planes, asignación de recursos a tareas, dar seguimiento al progreso, administrar presupuesto y analizar cargas de trabajo

http://www.microsoft.com/project/en-us/project-management.aspx

- **Microsoft Office PowerPoint**, para desarrollar y desplegar presentaciones visuales, se utilizara para la presentación del proyecto al jurado por medio de diapositivas multimedia

http://office.microsoft.com/es-es/powerpoint/

7.3.3.5 Notepad++

Es un editor de texto y de código fuente libre con soporte para varios lenguajes de programación. Solo funciona para Windows.

Se parece al "Bloc de notas" en cuanto al hecho de que puede editar texto sin formato y de forma simple. No obstante, incluye opciones más avanzadas que pueden ser útiles para usuarios avanzados como desarrolladores y programadores.

Se distribuye bajo los términos de la Licencia Pública General de GNU.

La versión que se utiliza será la 5.9.8, que se usará principalmente para editar ficheros de tipo .java, .xml, .sql, para configurar los archivos tanto del software Tomcat como del MySQL.

http://notepad-plus-plus.org/

7.3.4 Sistemas Operativos y Virtualización

7.3.4.1 Microsoft Windows

Es un sistema operativo gráfico desarrollado, comercializado y vendido por Microsoft, quien Bill Gates es el dueño. Se compone de varias familias de sistemas operativos, cada uno de los cuales se adaptan a un determinado sector de la industria de la computación.

La primera version fue creada el 20 de noviembre de 1985, hace 29 años, a esa primera versión se la denomino Windows 1.0. Dicho sistema lo utilizaremos para dar apoyo a los programas que necesitan un sistema de Windows para funcionar.

Para el proyecto utilizaremso la versión 7.

http://windows.microsoft.com/en-us/windows/downloads

7.3.4.2 Ubuntu

Es un sistema basado en el Debian Linux, con Unity como su entorno de escritorio por defecto. Se basa en el software libre, su nombre proviene de la filosofía sudafricana y significa algo así como “humanidad hacia nosotros”

Fue desarrollo por la compañia británica de Canonical Ltd., una empresa propiedad del empresario sudafricano Mark Shuttleworth.

En el proyecto se utilizará este sistema como el principal para desarrollar todo el proyecto, cuyos programas no pueden ejecutarse en este entorno se utilizara el sistema de Windows.

Para el proyecto se utilizará la versión 12.04.

http://www.ubuntu.com/download/desktop

7.3.4.3 VirtualBox

Es un programa de virtualización capaz de instalar en nuestro ordenador sistemas operativos basados en:

- Microsoft Windows
- Linux
- Solaris
- BSD
- IBM OS/2
- Mac OS X
- Otros

El programa ha sido creado por la empresa alemana innotek Actualmente es desarrollado por Oracle Corporation como parte de su familia de productos de virtualización. Este programa lo

utilizaremos para tener nuestro servidor junto con las maquinas clientes listas para ser ejecutadas en cualquier ordenador, ya que lo utilizaremos como un «sistema invitado», dentro de otro sistema operativo «anfitrión», cada uno con su propio ambiente virtual, este programa es gratuito y está en español. La versión utilizada es la versión 4.2.18.

https://www.virtualbox.org/

7.3.5 Diseño

7.3.5.1 Enterprise Architect

Es una herramienta de uso muy sencillo, que aborda el diseño y análisis UML y cubre el desarrollo de software desde la captura de requerimientos a lo largo de las etapas de análisis, diseño, pruebas y mantenimiento. EA es una herramienta multi-usuario, diseñada para ayudar a construir software robusto y fácil de mantener. Además, permite generar documentación e informes flexibles y de alta calidad.

Se ha utilizado la versión 7.5, para la creación de diagramas UML tanto para la parte de análisis, donde se encuentran los diagramas de navegabilidad, los actores, casos de uso y los diagrama de clases preliminares; como para la parte del diseño: diagramas de actividades, diagramas de secuencia, diagramas de clase, diagramas de despliegue y diagramas de paquetes.

http://www.sparxsystems.es/New/products/ea.html

7.3.5.2 Paint

Es un programa simple de dibujo gráfico desarrollado por Microsoft y que se integra dentro del sistema operativo.

La versión de Paint hace uso de la interfaz Ribbon. Cuenta con pinceles "artísticos" compuestos por diferentes tonos de gris y transparencias para dar un resultado más realista. También se ha añadido anti-aliasing a las formas, que pueden cambiar de tamaño libremente hasta que se rastericen cuando se selecciona otra herramienta. Esta versión utiliza el formato de archivo PNG por defecto para guardar imágenes, lo que garantiza máxima calidad y compatibilidad.

En este proyecto se ha utilizado la versión 6.1 incluida en el sistema operativo de Windows 7, para editar y almacenar algunas de las imágenes que se muestran en el proyecto.

http://windows.microsoft.com/es-ES/windows7/products/features/paint

7.3.5.3 VLC Media Player

Es un servidor portátil, libre y de código abierto, multiplataforma reproductor multimedia y streaming de medios de comunicación escrita por el proyecto VideoLAN.

El proyecto VideoLAN se inició originalmente como un proyecto académico en 1996. VLC utiliza para significar Fue pensado para consistir en un cliente y servidor para transmitir vídeos a través de una red de campus. Originalmente desarrollado por estudiantes de la École Centrale de París, en la actualidad se desarrolla por colaboradores de todo el mundo y es coordinado por VideoLAN, una organización sin fines de lucro.

La utilización de este programa, será para la grabación y edición del vídeo para la presentación de la prueba de la funcionalidad del servicio web creado, como opción de seguridad si se diera el caso de que por alguna circunstancia los servidores de OpenShift, estuvieran caidos o el entorno de Eclipse junto al tomcat no funcionaran, se tendría el vídeo para explicar la funcionalidad.

El vídeo de la funcionalidad para el servicio web, ha sido creado utilizando la versión 1.1.7

https://www.videolan.org/vlc/

7.4 Creación del Sistema

7.4.1 Problemas Encontrados

7.4.1.1 Versión BBDD OpenShift y Ubuntu

La versión utilizada en el servicio de OpenShift, que es una aplicación de cloud computing, al igual que el sistema Ubuntu vienen con la versión de MySQL 5.5., el proyecto estaba pensado para utilizar la versión 5.6.

A la hora de importar los datos de la base de datos de nuestro proyecto, por medio de un archivo .sql, obteníamos el siguiente error:

Figura 7.1. Error Importación .sql

El problema se debía a que en la versión 5.5 de MySQL, no se puede tener varios campos TIMESTAMP con el CURRENT_TIMESTAMP en DEFAULT u ON UPDATE.

La solución se encontró en: http://www.mysqltutorial.org/mysql-timestamp.aspx

Donde uno de los dos campos TIMESTAMP debería inicializarse a cero:

```
created TIMESTAMP DEFAULT 0,
modify    TIMESTAMP    DEFAULT    CURRENT_TIMESTAMP    ON    UPDATE
CURRENT_TIMESTAMP
```

Una vez cambiado eso en la descripción de la tabla se procede a la importación de la misma. Esta vez con el resultado esperado:

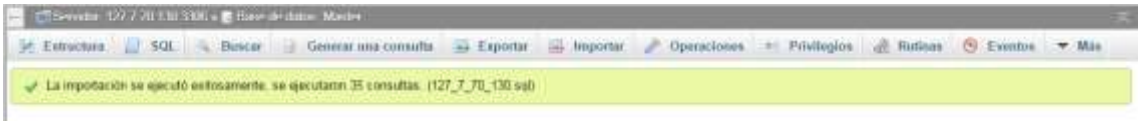


Figura 7.2. Éxito Importación .sql

Además se tuvo que cambiar los **insert** ya que no se actualizaba correctamente ese campo y cada vez que se producía una nueva inserción ese valor para el campo **created**, aparecía todo a ceros y a la hora de recuperar los valores fallaba la aplicación. Por lo que a la hora de los **insert**

se debía añadir el parámetro **created** con el valor *null*, para que este fuera modificado correctamente.

7.4.1.2 Evaluación de las Etiquetas if en Struts2

Al utilizar estas condiciones en los JSP, a veces se requería la comparación con algunos caracteres, el problema es que siempre devolvía *False* esa comparación del estilo:

```
<s:if test="%{#item.type == 'Q'}">
```

En http://stackoverflow.com/questions/1206513/how-to-evaluate-struts-2-sif, la solución que daban era la de cambiar las comillas por comillas simples y utilizar las comillas para el carácter para poder así realizar la comparación correctamente.

```
<s:if test='%{#item.type == ("Q")}'>
```

7.4.1.3 Tablas Dinámicas

Un problema que me tuvo pardo durante bastante tiempo fue la creación de tablas dinámicas, el usuario sería el que finalmente haría cuantas filas debería tener una tabla, con los problemas que conlleva el saber, transportar y almacenar un número indeterminado de datos.

Una parte de la solución que adopte era la utilización de código javascript, para clonar ciertos elementos que se podían repetir, otros se tuvieron que generar completos

http://www.forosdelweb.com/f4/problema-tabla-dinamica-1030075/

https://developer.mozilla.org/es/docs/Trazado_de_una_tabla_HTML_mediante_JavaScript_y_la_Interface_DOM

7.4.1.4 Scriptlets NO

Después de crear algunos JSPs, me di cuenta de que en la mayoría de ellos estaba usando inserciones de código Java dentro de ellos, por lo que leí no es una manera muy elegante de presentar los datos, junto a acciones de lógica en la capa de presentación.

Así que buscando encontré la manera de hacer lo que necesitaba por medio de etiquetas JSTL Java Standard Tag Languages y EL Expresion Languages, son extensiones a la especificación JSP de Sun, que facilitan las cosas y sustituyen la función de los Scriptlets.

El problema vino a la hora de recuperar los objetos que creaba en los action de Struts2 por medio de las interfaces “aware”, los cuales alojaban los objetos para que pudieran estar a disposición en otras partes de la aplicación, yo los utilizaba luego en los JSP, pero no podía acceder a ellos desde las etiquetas, buscando se encontró la solución, por medio de lo siguiente para acceder a los objetos implícitos con la siguiente correspondencia:

${requestScope.test} → **request.getAttribute("test");**
${sessionScope.test} → **session.getAttribute("test");**
${applicationScope.test} → **application.getAttribute("test");**
${pageScope.test} → **pageContext.getAttribute("test");**
${test} → **pageContext.findAttribute("test");**

7.4.1.5 Tildes y ñ no se muestran

Otro problema fue a la hora de almacenar los datos en la BBDD, los datos cuando los almacenaba lo hacía con caracteres extraños, provee todas las soluciones que encontré, desde cambiar parámetros en el Tomcat, en MySQL[17] recodificando los datos en código[18], nada de eso funcionaba.

La solución…estaba insertando los datos en MySQL por medio de variables de tipo BLOB, las cuales son variables binarias y guardaban los datos tal cual le llegan, con esos caracteres raros, ya fuera UTF-8 ó ISO-8859-1, los datos de tipo texto debían de guardarse con el tipo TEXT, para conservar el formato.

http://stackoverflow.com/questions/11624807/what-is-the-differences-between-blob-and-text-in-mysql-data-types

7.4.1.6 Generación de Javadoc sin Codificación

A la hora de generar la documentación de las clases de del Proyecto Java, aparecían errores por contener caracteres como las tildes, ñ y otras que el Javadoc, daba error cuando llegaba a los comentarios de las métodos y clases.

La solución fue utilizar diferentes flags, que se pueden poner a la hora de la creación del Javadoc, para que este interprete correctamente los diferentes símbolos que pueda encontrar y no los trate como errores, por ejemplo poniendo la codificación a UTF-8, la siguiente manera.

Project -> Generate Javadoc -> Next -> en la siguiente ventana, en *Extra Javadoc options* escribir los diferentes comandos para el Javadoc:

```
-encoding UTF-8 -charset UTF-8 -docencoding UTF-8
```

[17] http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=characterencoding-native2ascii
[18] http://programacion.jias.es/2012/09/comprobar-la-codificacion-de-una-entrada-convertir-de-utf-8-iso-8859-1/

7.4.2 Descripción Detallada de las Clases

7.4.2.1 Paquete com.miw.Model

Se mostraran la clase **user** como ejemplo, ya que el resto son modelos para encapsular los datos y son todas muy parecidas, el resto se pueden ver desde la documentación de **JavaDoc** del proyecto.

Class Summary	
Cambio	Modelo de datos para la información de los cambios de versión
Counter	Modelo de datos para la información sobre la conexión
Iden	Modelo de datos para la información de la identificación de riesgos
Impacto	Modelo de datos para los impactos de los riesgos
Info	Modelo de datos para la información de los planes de riesgo
Probabilidad	Modelo de datos para las probabilidades de los riesgos
Project	Modelo de datos para los proyectos
Respuesta	Modelo de datos para la información de la respuesta de los riesgos
User	Modelo de datos para los usuarios, tanto para el admin, Manager, como demás componentes del grupo de proyecto

7.4.2.1.1 Clase User

```
java.lang.Object
  └ com.miw.model.User
```

```
public class User
extends java.lang.Object
```

Modelo de datos para los usuarios, tanto para el admin, Manager, como demás componentes del grupo de proyecto

Author:

Pablo

Constructor Summary	
User()	

Method Summary

java.lang.String	**getCreate**()
java.lang.String	**getEmail**()
java.lang.Long	**getId**()
java.lang.Long	**getIdProyecto**()
java.lang.String	**getLanguage**()
java.lang.String	**getLogin**()
java.lang.String	**getModify**()
java.lang.String	**getPassword**()
java.lang.Boolean	**isAdmin**()
java.lang.Boolean	**isManager**()
void	**setAdmin**(java.lang.Boolean admin)
void	**setCreate**(java.lang.String create)
void	**setEmail**(java.lang.String email)
void	**setId**(java.lang.Long id)
void	**setIdProyecto**(java.lang.Long idProyecto)
void	**setLanguage**(java.lang.String language)
void	**setLogin**(java.lang.String login)
void	**setManager**(java.lang.Boolean Manager)
void	**setModify**(java.lang.String modify)
void	**setPassword**(java.lang.String password)

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

User

public **User**()

Method Detail

getId

public java.lang.Long **getId**()

setId
```
public void setId(java.lang.Long id)
```

getLogin
```
public java.lang.String getLogin()
```

setLogin
```
public void setLogin(java.lang.String login)
```

getPassword
```
public java.lang.String getPassword()
```

setPassword
```
public void setPassword(java.lang.String password)
```

getEmail
```
public java.lang.String getEmail()
```

setEmail
```
public void setEmail(java.lang.String email)
```

getLanguage
```
public java.lang.String getLanguage()
```

setLanguage
```
public void setLanguage(java.lang.String language)
```

isAdmin
```
public java.lang.Boolean isAdmin()
```

setAdmin
```
public void setAdmin(java.lang.Boolean admin)
```

isManager
```
public java.lang.Boolean isManager()
```

setManager
```
public void setManager(java.lang.Boolean Manager)
```

getIdProyecto
```
public java.lang.Long getIdProyecto()
```

setIdProyecto
```
public void setIdProyecto(java.lang.Long idProyecto)
```

getCreate
```
public java.lang.String getCreate()
```

setCreate
```
public void setCreate(java.lang.String create)
```

getModify
```
public java.lang.String getModify()
```

setModify
```
public void setModify(java.lang.String modify)
```

7.4.2.2 Paquete impl.miw.Presentation.Login

De entre los paquetes que contienen las clases relacionadas con el login, se describirá este al completo, el resto puede encontrarse en le JavaDoc del proyecto.

- **Login**
- Unlogin
- PassFogotten

Class Summary

LoginAction	Clase de la capa de presentación para la acción de logueo de los diferentes usuarios de la aplicación comprobándose que existan en la base de datos, extiende de ActionSupport que nos proporciona una implementación por defecto para las acciones más comunes con implementación de una interface “aware” para alojar objetos que puedan estar a disposición en otras partes de la aplicación y del ModelDriven que proporciona un objeto de modelo que se inserta en el ValueStack además de la propia acción
LoginInterceptor	Clase para implementar los interceptores, los cuales actuarán como un filtroHTTP o un Proxy, proveerán un camino para suplir el pre-processing y post-processing a través del action
SessionInterceptor	Clase que crea un interceptor, cuando la sesión de la aplicación halla caducado se pondrán las variables a null con la implementación de una interfaz "aware" para eliminar esas variables de sesión y extiende de AbstractInterceptor

7.4.2.2.1 Clase LoginAction

```
java.lang.Object
  └ com.opensymphony.xwork2.ActionSupport
      └ impl.miw.presentation.login.LoginAction
```

All Implemented Interfaces:

com.opensymphony.xwork2.Action, com.opensymphony.xwork2.LocaleProvider, com.opensymphony.xwork2.ModelDriven<User>, com.opensymphony.xwork2.TextProvider, com.opensymphony.xwork2.Validateable, com.opensymphony.xwork2.ValidationAware, java.io.Serializable, org.apache.struts2.interceptor.ServletRequestAware

---

```
public class LoginAction
extends com.opensymphony.xwork2.ActionSupport
implements          org.apache.struts2.interceptor.ServletRequestAware,
com.opensymphony.xwork2.ModelDriven<User>
```

Clase de la capa de presentación para la acción de logueo de los diferentes usuarios de la aplicación comprobándose que existan en la base de datos, extiende de ActionSupport que nos proporciona una implementación por defecto para las acciones más comunes con implementación de una interface "aware" para alojar objetos que puedan estar a disposición en otras partes de la aplicación y del ModelDriven que proporciona un objeto de modelo que se inserta en el ValueStack además de la propia acción

Author:

Pablo

See Also:

Serialized Form

Field Summary

Fields inherited from interface com.opensymphony.xwork2.Action

`ERROR, INPUT, LOGIN, NONE, SUCCESS`

Constructor Summary

LoginAction()

Method Summary

java.lang.String	**execute**() Método que implementa la ejecución del action de Struts con los datos del request
java.lang.String	**getEmail**()
LogService	**getLog**() Getter de la Infraestructura de Log
User	**getModel**() Getter para obtener el modelo que se inserta en la ValueStack en lugar de la propia acción
java.lang.String	**getPassword**()
UserService	**getUserService**() Getter para la obtención de la interfaz para los servicios aplicados al modelo de usuarios
void	**setEmail**(java.lang.String email)
void	**setLog**(LogService log) Setter de la infraestructura de Log
void	**setPassword**(java.lang.String password)
void	**setServletRequest**(javax.servlet.http.HttpServletRequest httpServletRequest) Setter que establece la solicitud al objeto HTTP en las clases de la aplicación
void	**setUserService**(UserService userService) Setter para establecer la interfaz para los servicios aplicados al modelo de usuarios

Methods inherited from class com.opensymphony.xwork2.ActionSupport

addActionError, addActionMessage, addFieldError, clearActionErrors, clearErrors, clearErrorsAndMessages, clearFieldErrors, clearMessages, clone, doDefault, getActionErrors, getActionMessages, getErrorMessages, getErrors, getFieldErrors, getFormatted, getLocale, getText, getText, getText, getText, getText, getText, getText, getText, getText, getTexts, getTexts, hasActionErrors, hasActionMessages, hasErrors, hasFieldErrors, hasKey, input, pause, setActionErrors, setActionMessages, setContainer, setFieldErrors, validate

Methods inherited from class java.lang.Object
`equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

LoginAction

`public` **`LoginAction`**`()`

Method Detail

getLog

`public LogService` **`getLog`**`()`

Getter de la Infraestructura de Log

Returns:

Log

---

setLog

`public void` **`setLog`**`(LogService log)`

Setter de la infraestructura de Log

Parameters:

`log` -

---

getUserService

`public UserService` **`getUserService`**`()`

Getter para la obtención de la interfaz para los servicios aplicados al modelo de usuarios

Returns:

UserService Objeto que hace referencia a la interfaz

---

setUserService

`public void` **`setUserService`**`(UserService userService)`

Setter para establecer la interfaz para los servicios aplicados al modelo de usuarios

Parameters:

`userService` - objecto interfaz

---

setServletRequest

`public void` **`setServletRequest`**`(javax.servlet.http.HttpServletRequest httpServletRequest)`

Setter que establece la solicitud al objeto HTTP en las clases de la aplicación

Specified by:

`setServletRequest` in interface `org.apache.struts2.interceptor.ServletRequestAware`

Parameters:

`httpServletRequest` - petición del Servlet

---

execute

```
public java.lang.String execute()
```

Método que implementa la ejecución del action de Struts con los datos del request

Specified by:

`execute` in interface `com.opensymphony.xwork2.Action`

Overrides:

`execute` in class `com.opensymphony.xwork2.ActionSupport`

Returns:

String cadena que será procesada en struts.xml

---

getEmail

```
public java.lang.String getEmail()
```

---

setEmail

```
public void setEmail(java.lang.String email)
```

---

getPassword

```
public java.lang.String getPassword()
```

---

setPassword

```
public void setPassword(java.lang.String password)
```

---

getModel

```
public User getModel()
```

Getter para obtener el modelo que se inserta en la ValueStack en lugar de la propia acción

Specified by:

`getModel` in interface `com.opensymphony.xwork2.ModelDriven<User>`

Returns:

User modelo de usuario

7.4.2.3 Paquete impl.miw.Presentation.InfoProject

Class Summary	
InfoProjectAction	Clase de la capa de presentación para información de los proyectos en la aplicación, extiende de ActionSupport que nos proporciona una implementación por defecto para las acciones más comunes con implementación de dos interface "aware" para alojar objetos que puedan estar a disposición en otras partes de la aplicación.

7.4.2.3.1 Clase InfoProjectAction

```
java.lang.Object
  └ com.opensymphony.xwork2.ActionSupport
      └ impl.miw.presentation.infoproject.InfoProjectAction
```

All Implemented Interfaces:

com.opensymphony.xwork2.Action, com.opensymphony.xwork2.LocaleProvider, com.opensymphony.xwork2.TextProvider, com.opensymphony.xwork2.Validateable, com.opensymphony.xwork2.ValidationAware, java.io.Serializable, org.apache.struts2.interceptor.ApplicationAware, org.apache.struts2.interceptor.ServletRequestAware

---

```
public class InfoProjectAction
extends com.opensymphony.xwork2.ActionSupport
implements org.apache.struts2.interceptor.ServletRequestAware,
org.apache.struts2.interceptor.ApplicationAware
```

Clase de la capa de presentación para información de los proyectos en la aplicación, extiende de ActionSupport que nos proporciona una implementación por defecto para las acciones más comunes con implementación de dos interface "aware" para alojar objetos que puedan estar a disposición en otras partes de la aplicación.

Author:

Pablo

See Also:

Serialized Form

Field Summary

Fields inherited from interface com.opensymphony.xwork2.Action

ERROR, INPUT, LOGIN, NONE, SUCCESS

Constructor Summary

InfoProjectAction()

Method Summary

java.lang.String	**execute**() Método que implementa la ejecución del action de Struts con los datos del request
LogService	**getLog**() Getter de la Infraestructura de Log
ProjectService	**getProjectService**() Getter para la obtención de la interfaz para los servicios aplicados al modelo de proyecto
UserService	**getUserService**() Getter para la obtención de la interfaz para los servicios aplicados al modelo de usuarios
void	**setApplication**(java.util.Map<java.lang.String,java.lang.Object> arg0) Setter que establece el mapa de propiedades de la aplicación.
void	**setLog**(LogService log) Setter de la infraestructura de Log
void	**setProjectService**(ProjectService projectService) Setter para establecer la interfaz para los servicios aplicados al modelo de proyecto
void	**setServletRequest**(javax.servlet.http.HttpServletRequest httpServletRequest) Setter que establece la solicitud al objeto HTTP en las clases de la aplicación
void	**setUserService**(UserService userService) Setter para establecer la interfaz para los servicios aplicados al modelo de usuarios

Methods inherited from class com.opensymphony.xwork2.ActionSupport
addActionError, addActionMessage, addFieldError, clearActionErrors, clearErrors, clearErrorsAndMessages, clearFieldErrors, clearMessages, clone, doDefault, getActionErrors, getActionMessages, getErrorMessages, getErrors, getFieldErrors, getFormatted, getLocale, getText, getText, getText, getText, getText, getText, getText, getText, getText, getTexts, getTexts, hasActionErrors, hasActionMessages, hasErrors, hasFieldErrors, hasKey, input, pause, setActionErrors, setActionMessages, setContainer, setFieldErrors, validate
Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait

Constructor Detail

InfoProjectAction

```
public InfoProjectAction()
```

Method Detail

getLog

```
public LogService getLog()
```

Getter de la Infraestructura de Log

Returns:

Log

---

setLog

```
public void setLog(LogService log)
```

Setter de la infraestructura de Log

Parameters:

`log` -

---

getProjectService

```
public ProjectService getProjectService()
```

Getter para la obtención de la interfaz para los servicios aplicados al modelo de proyecto

Returns:

ProjectService Objeto que hace referencia a la interfaz

---

setProjectService

```
public void setProjectService(ProjectService projectService)
```

Setter para establecer la interfaz para los servicios aplicados al modelo de proyecto

Parameters:

`projectService` - objecto interfaz

getUserService

`public UserService **getUserService**()`

Getter para la obtención de la interfaz para los servicios aplicados al modelo de usuarios

Returns:

UserService Objeto que hace referencia a la interfaz

setUserService

`public void **setUserService**(UserService userService)`

Setter para establecer la interfaz para los servicios aplicados al modelo de usuarios

Parameters:

`userService` - objecto interfaz

setServletRequest

`public void **setServletRequest**(javax.servlet.http.HttpServletRequest httpServletRequest)`

Setter que establece la solicitud al objeto HTTP en las clases de la aplicación

Specified by:

`setServletRequest` in interface `org.apache.struts2.interceptor.ServletRequestAware`

Parameters:

`httpServletRequest` - petición del Servlet

execute

`public java.lang.String **execute**()`

Método que implementa la ejecución del action de Struts con los datos del request

Specified by:

`execute` in interface `com.opensymphony.xwork2.Action`

Overrides:

`execute` in class `com.opensymphony.xwork2.ActionSupport`

Returns:

String cadena que será procesada en struts.xml

setApplication

`public void **setApplication**(java.util.Map<java.lang.String,java.lang.Object> arg0)`

Setter que establece el mapa de propiedades de la aplicación. Esto dará acceso a un mapa en el que puedan poner objetos que deben estar a disposición en otras partes de la aplicación.

Specified by:

`setApplication` in interface `org.apache.struts2.interceptor.ApplicationAware`

Parameters:

`arg0` - un mapa de propiedades de la aplicación.

7.4.2.4 Paquete impl.miw.Presentation.Plan

Se mostrará de entre los paquetes que se compone la gestión de riesgos, el paquete del plan de riesgos, los otros planes pueden consultarse en la documentación *JavaDoc*, como son:

- **Plan**
- Iden
- resp

Class Summary	
ChangeAction	Clase de la capa de presentación para la acción de cambio de versión en el plan de riesgos en la aplicación, extiende de ActionSupport que nos proporciona una implementación por defecto para las acciones más comunes con implementación de una interface "aware" para alojar objetos que puedan estar a disposición en otras partes de la aplicación
LoadAction	Clase de la capa de presentación para la acción de carga de la información de los planes de riesgo almacenados en la base de datos, extiende de ActionSupport que nos proporciona una implementación por defecto para las acciones más comunes con implementación de dos interface "aware" para alojar objetos que puedan estar a disposición en otras partes de la aplicación.
SaveAction	Clase de la capa de presentación para la acción de almacenamiento de la información de los planes de riesgo escritos en la aplicación, extiende de ActionSupport que nos proporciona una implementación por defecto para las acciones más comunes con implementación de dos interfaces "aware" para alojar objetos que puedan estar a disposición en otras partes de la aplicación.

7.4.2.4.1 Clase ChangeAction

```
java.lang.Object
  └ com.opensymphony.xwork2.ActionSupport
      └ impl.miw.presentation.plan.ChangeAction
```

All Implemented Interfaces:

com.opensymphony.xwork2.Action, com.opensymphony.xwork2.LocaleProvider, com.opensymphony.xwork2.TextProvider, com.opensymphony.xwork2.Validateable, com.opensymphony.xwork2.ValidationAware, java.io.Serializable, org.apache.struts2.interceptor.ApplicationAware

---

```
public class ChangeAction
extends com.opensymphony.xwork2.ActionSupport
implements org.apache.struts2.interceptor.ApplicationAware
```

Clase de la capa de presentación para la acción de cambio de versión en el plan de riesgos en la aplicación, extiende de ActionSupport que nos proporciona una implementación por defecto para las acciones más comunes con implementación de una

interface “aware” para alojar objetos que puedan estar a disposición en otras partes de la aplicación

Author:

Pablo

See Also:

Serialized Form

Field Summary

Fields inherited from interface com.opensymphony.xwork2.Action

ERROR, INPUT, LOGIN, NONE, SUCCESS

Constructor Summary

ChangeAction()

Method Summary

java.lang.String	**execute**() Método que implementa la ejecución del action de Struts con los datos del request
LogService	**getLog**() Getter de la Infraestructura de Log
java.lang.String	**getVersion**()
void	**setApplication**(java.util.Map<java.lang.String,java.lang.Object> arg0) Setter que establece el mapa de propiedades de la aplicación.
void	**setLog**(LogService log) Setter de la infraestructura de Log
void	**setVersion**(java.lang.String version)

Methods inherited from class com.opensymphony.xwork2.ActionSupport
addActionError, addActionMessage, addFieldError, clearActionErrors, clearErrors, clearErrorsAndMessages, clearFieldErrors, clearMessages, clone, doDefault, getActionErrors, getActionMessages, getErrorMessages, getErrors, getFieldErrors, getFormatted, getLocale, getText, getText, getText, getText, getText, getText, getText, getText, getText, getTexts, getTexts, hasActionErrors, hasActionMessages, hasErrors, hasFieldErrors, hasKey, input, pause, setActionErrors, setActionMessages, setContainer, setFieldErrors, validate
Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

ChangeAction

public **ChangeAction**()

Method Detail

getLog

public LogService **getLog**()

Getter de la Infraestructura de Log

Returns:

Log

---

setLog

public void **setLog**(LogService log)

Setter de la infraestructura de Log

Parameters:

log -

---

setApplication

public void **setApplication**(java.util.Map<java.lang.String,java.lang.Object> arg0)

Setter que establece el mapa de propiedades de la aplicación. Esto dará acceso a un mapa en el que puedan poner objetos que deben estar a disposición en otras partes de la aplicación.

Specified by:

setApplication in interface org.apache.struts2.interceptor.ApplicationAware

Parameters:

arg0 - un mapa de propiedades de la aplicación.

---

execute

```
public java.lang.String execute()
```

Método que implementa la ejecución del action de Struts con los datos del request

Specified by:

`execute` in interface `com.opensymphony.xwork2.Action`

Overrides:

`execute` in class `com.opensymphony.xwork2.ActionSupport`

Returns:

String cadena que será procesada en struts.xml

---

getVersion

```
public java.lang.String getVersion()
```

---

setVersion

```
public void setVersion(java.lang.String version)
```

7.4.2.4.2 Clase LoadAction

```
java.lang.Object
  └ com.opensymphony.xwork2.ActionSupport
      └ impl.miw.presentation.plan.LoadAction
```

All Implemented Interfaces:

com.opensymphony.xwork2.Action, com.opensymphony.xwork2.LocaleProvider, com.opensymphony.xwork2.TextProvider, com.opensymphony.xwork2.Validateable, com.opensymphony.xwork2.ValidationAware, java.io.Serializable, org.apache.struts2.interceptor.ApplicationAware, org.apache.struts2.interceptor.ServletRequestAware

---

```
public class LoadAction
extends com.opensymphony.xwork2.ActionSupport
implements org.apache.struts2.interceptor.ApplicationAware,
org.apache.struts2.interceptor.ServletRequestAware
```

Clase de la capa de presentación para la acción de carga de la información de los planes de riesgo almacenados en la base de datos, extiende de ActionSupport que nos proporciona una implementación por defecto para las acciones más comunes con implementación de dos interface "aware" para alojar objetos que puedan estar a disposición en otras partes de la aplicación.

Author:

Pablo

See Also:

Serialized Form

Field Summary

Fields inherited from interface com.opensymphony.xwork2.Action

`ERROR, INPUT, LOGIN, NONE, SUCCESS`

Constructor Summary

LoadAction()

Method Summary

java.lang.String	**execute**() Método que implementa la ejecución del action de Struts con los datos del request
InfoService	**getInfoService**() Getter para la obtención de la interfaz para los servicios aplicados al modelo de info que se utiliza para la información de los riesgos
LogService	**getLog**() Getter de la Infraestructura de Log
void	**setApplication**(java.util.Map<java.lang.String,java.lang.Object> arg0) Setter que establece el mapa de propiedades de la aplicación.
void	**setInfoService**(InfoService infoService) Setter para establecer la interfaz para los servicios aplicados al modelo de info que se utiliza para la información de los riesgos
void	**setLog**(LogService log) Setter de la infraestructura de Log
void	**setServletRequest**(javax.servlet.http.HttpServletRequest httpServletRequest) Setter que establece la solicitud al objeto HTTP en las clases de la aplicación

Methods inherited from class com.opensymphony.xwork2.ActionSupport

`addActionError, addActionMessage, addFieldError, clearActionErrors, clearErrors, clearErrorsAndMessages, clearFieldErrors, clearMessages, clone, doDefault, getActionErrors, getActionMessages,`

getErrorMessages, getErrors, getFieldErrors, getFormatted, getLocale, getText, getText, getText, getText, getText, getText, getText, getText, getText, getTexts, getTexts, hasActionErrors, hasActionMessages, hasErrors, hasFieldErrors, hasKey, input, pause, setActionErrors, setActionMessages, setContainer, setFieldErrors, validate
Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

LoadAction

```
public LoadAction()
```

Method Detail

getLog

```
public LogService getLog()
```

Getter de la Infraestructura de Log

Returns:

Log

---

setLog

```
public void setLog(LogService log)
```

Setter de la infraestructura de Log

Parameters:

`log` -

---

getInfoService

```
public InfoService getInfoService()
```

Getter para la obtención de la interfaz para los servicios aplicados al modelo de info que se utiliza para la información de los riesgos

Returns:

InfoService Objeto que hace referencia a la interfaz

---

setInfoService

```
public void setInfoService(InfoService infoService)
```

Setter para establecer la interfaz para los servicios aplicados al modelo de info que se utiliza para la información de los riesgos

Parameters:

`infoService` - objecto interfaz

---

setApplication

`public void` **`setApplication`**`(java.util.Map<java.lang.String,java.lang.Object> arg0)`

Setter que establece el mapa de propiedades de la aplicación. Esto dará acceso a un mapa en el que puedan poner objetos que deben estar a disposición en otras partes de la aplicación.

Specified by:

`setApplication` in interface `org.apache.struts2.interceptor.ApplicationAware`

Parameters:

`arg0` - un mapa de propiedades de la aplicación.

---

setServletRequest

`public void` **`setServletRequest`**`(javax.servlet.http.HttpServletRequest httpServletRequest)`

Setter que establece la solicitud al objeto HTTP en las clases de la aplicación

Specified by:

`setServletRequest` in interface `org.apache.struts2.interceptor.ServletRequestAware`

Parameters:

`httpServletRequest` - petición del Servlet

---

execute

`public java.lang.String` **`execute`**`()`

Método que implementa la ejecución del action de Struts con los datos del request

Specified by:

`execute` in interface `com.opensymphony.xwork2.Action`

Overrides:

`execute` in class `com.opensymphony.xwork2.ActionSupport`

Returns:

String cadena que será procesada en struts.xml

7.4.2.4.3 Clase SaveAction

```
java.lang.Object
  └ com.opensymphony.xwork2.ActionSupport
      └ impl.miw.presentation.plan.SaveAction
```

All Implemented Interfaces:

com.opensymphony.xwork2.Action, com.opensymphony.xwork2.LocaleProvider, com.opensymphony.xwork2.TextProvider, com.opensymphony.xwork2.Validateable, com.opensymphony.xwork2.ValidationAware, java.io.Serializable, org.apache.struts2.interceptor.ApplicationAware, org.apache.struts2.interceptor.ServletRequestAware

---

```
public class SaveAction
extends com.opensymphony.xwork2.ActionSupport
implements          org.apache.struts2.interceptor.ServletRequestAware,
org.apache.struts2.interceptor.ApplicationAware
```

Clase de la capa de presentación para la acción de almacenamiento de la información de los planes de riesgo escritos en la aplicación, extiende de ActionSupport que nos proporciona una implementación por defecto para las acciones más comunes con implementación de dos interfaces "aware" para alojar objetos que puedan estar a disposición en otras partes de la aplicación.

Author:

Pablo

See Also:

Serialized Form

Field Summary

Fields inherited from interface com.opensymphony.xwork2.Action

ERROR, INPUT, LOGIN, NONE, SUCCESS

Constructor Summary

SaveAction()

Method Summary

java.lang.String	**execute**() Método que implementa la ejecución del action de Struts con los datos del request
java.lang.Long	**getCodPlan**()
java.lang.String	**getCorte**()
java.lang.String	**getDate**()
java.lang.Long	**getId**()
java.lang.Long	**getIdP**()

InfoService	**getInfoService**() Getter para la obtención de la interfaz para los servicios aplicados al modelo de info que se utiliza para la información de los riesgos
LogService	**getLog**() Getter de la Infraestructura de Log
java.lang.String	**getName**()
int	**getNumFilas**()
java.lang.String	**getPlan**()
ProjectService	**getProjectService**() Getter para la obtención de la interfaz para los servicios aplicados al modelo de proyecto
java.lang.String	**getRango**()
java.lang.String	**getVersion**()
void	**setApplication**(java.util.Map<java.lang.String,java.lang.Object> arg0) Setter que establece el mapa de propiedades de la aplicación.
void	**setCodPlan**(java.lang.Long codPlan)
void	**setCorte**(java.lang.String matrixCorte)
void	**setDate**(java.lang.String date)
void	**setId**(java.lang.Long id)
void	**setIdP**(java.lang.Long idP)
void	**setInfoService**(InfoService infoService) Setter para establecer la interfaz para los servicios aplicados al modelo de info que se utiliza para la información de los riesgos
void	**setLog**(LogService log) Setter de la infraestructura de Log
void	**setName**(java.lang.String name)
void	**setNumFilas**(int numFilas)

void	**setPlan**(java.lang.String plan)
void	**setProjectService**(ProjectService projectService) Setter para establecer la interfaz para los servicios aplicados al modelo de proyecto
void	**setRango**(java.lang.String rango)
void	**setServletRequest**(javax.servlet.http.HttpServletRequest httpServletRequest) Setter que establece la solicitud al objeto HTTP en las clases de la aplicación
void	**setVersion**(java.lang.String version)

Methods inherited from class com.opensymphony.xwork2.ActionSupport
addActionError, addActionMessage, addFieldError, clearActionErrors, clearErrors, clearErrorsAndMessages, clearFieldErrors, clearMessages, clone, doDefault, getActionErrors, getActionMessages, getErrorMessages, getErrors, getFieldErrors, getFormatted, getLocale, getText, getText, getText, getText, getText, getText, getText, getText, getText, getTexts, getTexts, hasActionErrors, hasActionMessages, hasErrors, hasFieldErrors, hasKey, input, pause, setActionErrors, setActionMessages, setContainer, setFieldErrors, validate
Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

SaveAction

public **SaveAction**()

Method Detail

getLog

public LogService **getLog**()

Getter de la Infraestructura de Log

Returns:

Log

setLog

`public void` **`setLog`**`(LogService log)`

Setter de la infraestructura de Log

Parameters:

`log` -

getInfoService

`public InfoService` **`getInfoService`**`()`

Getter para la obtención de la interfaz para los servicios aplicados al modelo de info que se utiliza para la información de los riesgos

Returns:

InfoService Objeto que hace referencia a la interfaz

setInfoService

`public void` **`setInfoService`**`(InfoService infoService)`

Setter para establecer la interfaz para los servicios aplicados al modelo de info que se utiliza para la información de los riesgos

Parameters:

`infoService` - objecto interfaz

getProjectService

`public ProjectService` **`getProjectService`**`()`

Getter para la obtención de la interfaz para los servicios aplicados al modelo de proyecto

Returns:

ProjectService Objeto que hace referencia a la interfaz

setProjectService

`public void` **`setProjectService`**`(ProjectService projectService)`

Setter para establecer la interfaz para los servicios aplicados al modelo de proyecto

Parameters:

`projectService` - objecto interfaz

setServletRequest

`public void` **`setServletRequest`**`(javax.servlet.http.HttpServletRequest httpServletRequest)`

Setter que establece la solicitud al objeto HTTP en las clases de la aplicación

Specified by:

`setServletRequest` in interface `org.apache.struts2.interceptor.ServletRequestAware`

Parameters:

`httpServletRequest` - petición del Servlet

setApplication

```
public void setApplication(java.util.Map<java.lang.String,java.lang.Object> arg0)
```

Setter que establece el mapa de propiedades de la aplicación. Esto dará acceso a un mapa en el que puedan poner objetos que deben estar a disposición en otras partes de la aplicación.

Specified by:

`setApplication` in interface `org.apache.struts2.interceptor.ApplicationAware`

Parameters:

`arg0` - un mapa de propiedades de la aplicación.

---

execute

```
public java.lang.String execute()
```

Método que implementa la ejecución del action de Struts con los datos del request

Specified by:

`execute` in interface `com.opensymphony.xwork2.Action`

Overrides:

`execute` in class `com.opensymphony.xwork2.ActionSupport`

Returns:

String cadena que será procesada en struts.xml

---

getPlan

```
public java.lang.String getPlan()
```

---

setPlan

```
public void setPlan(java.lang.String plan)
```

---

getDate

```
public java.lang.String getDate()
```

---

setDate

```
public void setDate(java.lang.String date)
```

---

getId

```
public java.lang.Long getId()
```

---

setId

```
public void setId(java.lang.Long id)
```

---

getName

```
public java.lang.String getName()
```

---

setName

```
public void setName(java.lang.String name)
```

getIdP
```
public java.lang.Long getIdP()
```

setIdP
```
public void setIdP(java.lang.Long idP)
```

getCodPlan
```
public java.lang.Long getCodPlan()
```

setCodPlan
```
public void setCodPlan(java.lang.Long codPlan)
```

getNumFilas
```
public int getNumFilas()
```

setNumFilas
```
public void setNumFilas(int numFilas)
```

getCorte
```
public java.lang.String getCorte()
```

setCorte
```
public void setCorte(java.lang.String matrixCorte)
```

getVersion
```
public java.lang.String getVersion()
```

setVersion
```
public void setVersion(java.lang.String version)
```

getRango
```
public java.lang.String getRango()
```

setRango
```
public void setRango(java.lang.String rango)
```

7.4.2.5 Paquete impl.miw.presentation.Register

Para el caso de los paquetes relacionados: *register*, *creation* y *update* se mostrará el primero de ellos, el resto se puede ver en la documentación de JavaDoc.

Class Summary	
OptionManagerAction	Clase de la capa de presentación para la acción de actualización o eliminación de los diferentes datos de los usuarios en los proyectos de la aplicación, extiende de ActionSupport que nos proporciona una implementación por defecto para las acciones más comunes con implementación de una interface "aware" para alojar objetos que puedan estar a disposición en otras partes de la aplicación
RegisterAction	Clase de la capa de presentación para la acción de registrar un nuevo Manager sin proyecto asignado en la aplicación, extiende de ActionSupport que nos proporciona una implementación por defecto para las acciones más comunes con implementación de una interface "aware" para alojar objetos que puedan estar a disposición en otras partes de la aplicación y del ModelDriven que proporciona un objeto de modelo que se inserta en el ValueStack además de la propia acción
RegisterMemberAction	Clase de la capa de presentación para la acción de registrar nuevos miembros para los proyectos abiertos en la aplicación, extiende de ActionSupport que nos proporciona una implementación por defecto para las acciones más comunes con implementación de una interface "aware" para alojar objetos que puedan estar a disposición en otras partes de la aplicación y del ModelDriven que proporciona un objeto de modelo que se inserta en el ValueStack además de la propia acción
SeekManagerFreeAction	Clase de la capa de presentación para la acción de búsqueda de Manager sin proyecto asignado en la aplicación, extiende de ActionSupport que nos proporciona una implementación por defecto para las acciones más comunes con implementación de una interface "aware" para alojar objetos que puedan estar a disposición en otras partes de la aplicación

7.4.2.5.1 Clase OptionManagerAction

```
java.lang.Object
  └ com.opensymphony.xwork2.ActionSupport
      └ impl.miw.presentation.register.OptionManagerAction
```

All Implemented Interfaces:

com.opensymphony.xwork2.Action, com.opensymphony.xwork2.LocaleProvider, com.opensymphony.xwork2.TextProvider, com.opensymphony.xwork2.Validateable, com.opensymphony.xwork2.ValidationAware, java.io.Serializable, org.apache.struts2.interceptor.ServletRequestAware

---

```
public class OptionManagerAction
extends com.opensymphony.xwork2.ActionSupport
implements org.apache.struts2.interceptor.ServletRequestAware
```

Clase de la capa de presentación para la acción de actualización o eliminación de los diferentes datos de los usuarios en los proyectos de la aplicación, extiende de ActionSupport que nos proporciona una implementación por defecto para las acciones más comunes con implementación de una interface "aware" para alojar objetos que puedan estar a disposición en otras partes de la aplicación

Author:

Pablo

See Also:

Serialized Form

Field Summary

Fields inherited from interface com.opensymphony.xwork2.Action

ERROR, INPUT, LOGIN, NONE, SUCCESS

Constructor Summary

OptionManagerAction()

Method Summary

java.lang.String	**execute**() Método que implementa la ejecución del action de Struts con los datos del request
java.lang.String	**getDelete**()
LogService	**getLog**()

	Getter de la Infraestructura de Log
java.lang.String	**getUpdateEmail**()
java.lang.String	**getUpdateId**()
java.lang.String	**getUpdateIdProject**()
java.lang.String	**getUpdateLogin**()
UserService	**getUserService**() Getter para la obtención de la interfaz para los servicios aplicados al modelo de usuarios
void	**setDelete**(java.lang.String delete)
void	**setLog**(LogService log) Setter de la infraestructura de Log
void	**setServletRequest**(javax.servlet.http.HttpServletRequest httpServletRequest) Setter que establece la solicitud al objeto HTTP en las clases de la aplicación
void	**setUpdateEmail**(java.lang.String updateEmail)
void	**setUpdateId**(java.lang.String updateId)
void	**setUpdateIdProject**(java.lang.String updateIdProject)
void	**setUpdateLogin**(java.lang.String updateLogin)
void	**setUserService**(UserService userService) Setter para establecer la interfaz para los servicios aplicados al modelo de usuarios

Methods inherited from class com.opensymphony.xwork2.ActionSupport
addActionError, addActionMessage, addFieldError, clearActionErrors, clearErrors, clearErrorsAndMessages, clearFieldErrors, clearMessages, clone, doDefault, getActionErrors, getActionMessages, getErrorMessages, getErrors, getFieldErrors, getFormatted, getLocale, getText, getText, getText, getText, getText, getText, getText, getText, getText, getTexts, getTexts, hasActionErrors, hasActionMessages, hasErrors, hasFieldErrors, hasKey, input, pause, setActionErrors, setActionMessages, setContainer, setFieldErrors,

validate
Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

OptionManagerAction

`public` **`OptionManagerAction`**`()`

Method Detail

getLog

`public` `LogService` **`getLog`**`()`

Getter de la Infraestructura de Log

Returns:

Log

---

setLog

`public void` **`setLog`**`(LogService log)`

Setter de la infraestructura de Log

Parameters:

`log` -

---

getUserService

`public` `UserService` **`getUserService`**`()`

Getter para la obtención de la interfaz para los servicios aplicados al modelo de usuarios

Returns:

UserService Objeto que hace referencia a la interfaz

---

setUserService

`public void` **`setUserService`**`(UserService userService)`

Setter para establecer la interfaz para los servicios aplicados al modelo de usuarios

Parameters:

`userService` - objecto interfaz

---

setServletRequest

`public void` **`setServletRequest`**`(javax.servlet.http.HttpServletRequest httpServletRequest)`

Setter que establece la solicitud al objeto HTTP en las clases de la aplicación

Specified by:

`setServletRequest` in interface `org.apache.struts2.interceptor.ServletRequestAware`

Parameters:

`httpServletRequest` - petición del Servlet

---

execute

`public java.lang.String **execute**()`

Método que implementa la ejecución del action de Struts con los datos del request

Specified by:

`execute` in interface `com.opensymphony.xwork2.Action`

Overrides:

`execute` in class `com.opensymphony.xwork2.ActionSupport`

Returns:

String cadena que será procesada en struts.xml

---

getUpdateId

`public java.lang.String **getUpdateId**()`

---

setUpdateId

`public void **setUpdateId**(java.lang.String updateId)`

---

getUpdateLogin

`public java.lang.String **getUpdateLogin**()`

---

setUpdateLogin

`public void **setUpdateLogin**(java.lang.String updateLogin)`

---

getUpdateEmail

`public java.lang.String **getUpdateEmail**()`

---

setUpdateEmail

`public void **setUpdateEmail**(java.lang.String updateEmail)`

---

getUpdateIdProject

`public java.lang.String **getUpdateIdProject**()`

---

setUpdateIdProject

`public void **setUpdateIdProject**(java.lang.String updateIdProject)`

---

getDelete

`public java.lang.String **getDelete**()`

---

setDelete

`public void **setDelete**(java.lang.String delete)`

7.4.2.5.2 Clase RegisterAction

```
java.lang.Object
  └ com.opensymphony.xwork2.ActionSupport
      └ impl.miw.presentation.register.RegisterAction
```

All Implemented Interfaces:

com.opensymphony.xwork2.Action, com.opensymphony.xwork2.LocaleProvider, com.opensymphony.xwork2.ModelDriven<User>, com.opensymphony.xwork2.TextProvider, com.opensymphony.xwork2.Validateable, com.opensymphony.xwork2.ValidationAware, java.io.Serializable, org.apache.struts2.interceptor.ServletRequestAware

---

```
public class RegisterAction
extends com.opensymphony.xwork2.ActionSupport
implements org.apache.struts2.interceptor.ServletRequestAware,
com.opensymphony.xwork2.ModelDriven<User>
```

Clase de la capa de presentación para la acción de registrar un nuevo Manager sin proyecto asignado en la aplicación, extiende de ActionSupport que nos proporciona una implementación por defecto para las acciones más comunes con implementación de una interface "aware" para alojar objetos que puedan estar a disposición en otras partes de la aplicación y del ModelDriven que proporciona un objeto de modelo que se inserta en el ValueStack además de la propia acción

Author:

Pablo

See Also:

Serialized Form

Field Summary

Fields inherited from interface com.opensymphony.xwork2.Action

ERROR, INPUT, LOGIN, NONE, SUCCESS

Constructor Summary

RegisterAction()

Method Summary

java.lang.String	**execute**() Método que implementa la ejecución del action de Struts con los datos del request
java.lang.String	**getEmailRegister**()
java.lang.String	**getLanguageRegister**()
LogService	**getLog**() Getter de la Infraestructura de Log
java.lang.String	**getLoginRegister**()
User	**getModel**() Getter para obtener el modelo que se inserta en la ValueStack en lugar de la propia acción
java.lang.String	**getPasswordRegister**()
UserService	**getUserService**() Getter para la obtención de la interfaz para los servicios aplicados al modelo de usuarios
boolean	**isAdminRegister**()
boolean	**isManagerRegister**()
void	**setAdminRegister**(boolean adminRegister)
void	**setEmailRegister**(java.lang.String emailRegister)
void	**setLanguageRegister**(java.lang.String languageRegister)
void	**setLog**(LogService log) Setter de la infraestructura de Log
void	**setLoginRegister**(java.lang.String loginRegister)
void	**setManagerRegister**(boolean managerRegister)
void	**setPasswordRegister**(java.lang.String passwordRegister)
void	**setServletRequest**(javax.servlet.http.HttpServletRequest

	`httpServletRequest)` Setter que establece la solicitud al objeto HTTP en las clases de la aplicación
`void`	**`setUserService`**`(UserService userService)` Setter para establecer la interfaz para los servicios aplicados al modelo de usuarios

Methods inherited from class com.opensymphony.xwork2.ActionSupport
`addActionError, addActionMessage, addFieldError, clearActionErrors, clearErrors, clearErrorsAndMessages, clearFieldErrors, clearMessages, clone, doDefault, getActionErrors, getActionMessages, getErrorMessages, getErrors, getFieldErrors, getFormatted, getLocale, getText, getText, getText, getText, getText, getText, getText, getText, getText, getTexts, getTexts, hasActionErrors, hasActionMessages, hasErrors, hasFieldErrors, hasKey, input, pause, setActionErrors, setActionMessages, setContainer, setFieldErrors, validate`
Methods inherited from class java.lang.Object
`equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

RegisterAction

`public` **`RegisterAction`**`()`

Method Detail

getLog

`public LogService` **`getLog`**`()`

Getter de la Infraestructura de Log

Returns:

Log

---

setLog

`public void` **`setLog`**`(LogService log)`

Setter de la infraestructura de Log

Parameters:

`log` -

---

getUserService

`public UserService **getUserService**()`

Getter para la obtención de la interfaz para los servicios aplicados al modelo de usuarios

Returns:

UserService Objeto que hace referencia a la interfaz

---

setUserService

`public void **setUserService**(UserService userService)`

Setter para establecer la interfaz para los servicios aplicados al modelo de usuarios

Parameters:

`userService` - objecto interfaz

---

setServletRequest

`public void **setServletRequest**(javax.servlet.http.HttpServletRequest httpServletRequest)`

Setter que establece la solicitud al objeto HTTP en las clases de la aplicación

Specified by:

`setServletRequest` in interface `org.apache.struts2.interceptor.ServletRequestAware`

Parameters:

`httpServletRequest` - petición del Servlet

---

execute

`public java.lang.String **execute**()`

Método que implementa la ejecución del action de Struts con los datos del request

Specified by:

`execute` in interface `com.opensymphony.xwork2.Action`

Overrides:

`execute` in class `com.opensymphony.xwork2.ActionSupport`

Returns:

String cadena que será procesada en struts.xml

---

getLoginRegister

`public java.lang.String **getLoginRegister**()`

---

setLoginRegister

`public void **setLoginRegister**(java.lang.String loginRegister)`

---

getPasswordRegister

`public java.lang.String **getPasswordRegister**()`

---

setPasswordRegister

`public void **setPasswordRegister**(java.lang.String passwordRegister)`

getLanguageRegister
`public java.lang.String` **`getLanguageRegister`**`()`

setLanguageRegister
`public void` **`setLanguageRegister`**`(java.lang.String languageRegister)`

getEmailRegister
`public java.lang.String` **`getEmailRegister`**`()`

setEmailRegister
`public void` **`setEmailRegister`**`(java.lang.String emailRegister)`

isAdminRegister
`public boolean` **`isAdminRegister`**`()`

setAdminRegister
`public void` **`setAdminRegister`**`(boolean adminRegister)`

isManagerRegister
`public boolean` **`isManagerRegister`**`()`

setManagerRegister
`public void` **`setManagerRegister`**`(boolean managerRegister)`

getModel
`public User` **`getModel`**`()`

Getter para obtener el modelo que se inserta en la ValueStack en lugar de la propia acción

Specified by:

`getModel` in interface `com.opensymphony.xwork2.ModelDriven<User>`

Returns:

User modelo de usuario

7.4.2.5.3 Clase RegisterMemberAction

```
java.lang.Object
  └ com.opensymphony.xwork2.ActionSupport
      └ impl.miw.presentation.register.RegisterMemberAction
```

All Implemented Interfaces:

com.opensymphony.xwork2.Action, com.opensymphony.xwork2.LocaleProvider, com.opensymphony.xwork2.ModelDriven<User>, com.opensymphony.xwork2.TextProvider, com.opensymphony.xwork2.Validateable,

com.opensymphony.xwork2.ValidationAware, java.io.Serializable, org.apache.struts2.interceptor.ServletRequestAware

---

```
public class RegisterMemberAction
extends com.opensymphony.xwork2.ActionSupport
implements org.apache.struts2.interceptor.ServletRequestAware,
com.opensymphony.xwork2.ModelDriven<User>
```

Clase de la capa de presentación para la acción de registrar nuevos miembros para los proyectos abiertos en la aplicación, extiende de ActionSupport que nos proporciona una implementación por defecto para las acciones más comunes con implementación de una interface "aware" para alojar objetos que puedan estar a disposición en otras partes de la aplicación y del ModelDriven que proporciona un objeto de modelo que se inserta en el ValueStack además de la propia acción

Author:

Pablo

See Also:

Serialized Form

Field Summary

Fields inherited from interface com.opensymphony.xwork2.Action

ERROR, INPUT, LOGIN, NONE, SUCCESS

Constructor Summary

RegisterMemberAction()

Method Summary

java.lang.String	**execute**() Método que implementa la ejecución del action de Struts con los datos del request
java.lang.String	**getEmailRegister**()
java.lang.St	**getLanguageRegister**()

ring	
LogService	**getLog**() Getter de la Infraestructura de Log
java.lang.String	**getLoginRegister**()
User	**getModel**() Getter para obtener el modelo que se inserta en la ValueStack en lugar de la propia acción
java.lang.String	**getPasswordRegister**()
java.lang.Long	**getProjectRegister**()
ProjectService	**getProjectService**() Getter para la obtención de la interfaz para los servicios aplicados al modelo de proyecto
java.lang.Integer	**getStepProjectRegister**()
UserService	**getUserService**() Getter para la obtención de la interfaz para los servicios aplicados al modelo de usuarios
boolean	**isAdminRegister**()
boolean	**isManagerRegister**()
void	**setAdminRegister**(boolean adminRegister)
void	**setEmailRegister**(java.lang.String emailRegister)
void	**setLanguageRegister**(java.lang.String languageRegister)
void	**setLog**(LogService log) Setter de la infraestructura de Log
void	**setLoginRegister**(java.lang.String loginRegister)
void	**setManagerRegister**(boolean managerRegister)
void	**setPasswordRegister**(java.lang.String passwordRegister)
void	**setProjectRegister**(java.lang.Long projectRegister)

void	**setProjectService**(ProjectService projectService) Setter para establecer la interfaz para los servicios aplicados al modelo de proyecto
void	**setServletRequest**(javax.servlet.http.HttpServletRequest httpServletRequest) Setter que establece la solicitud al objeto HTTP en las clases de la aplicación
void	**setStepProjectRegister**(java.lang.Integer stepProjectRegister)
void	**setUserService**(UserService userService) Setter para establecer la interfaz para los servicios aplicados al modelo de usuarios

Methods inherited from class com.opensymphony.xwork2.ActionSupport

addActionError, addActionMessage, addFieldError, clearActionErrors, clearErrors, clearErrorsAndMessages, clearFieldErrors, clearMessages, clone, doDefault, getActionErrors, getActionMessages, getErrorMessages, getErrors, getFieldErrors, getFormatted, getLocale, getText, getText, getText, getText, getText, getText, getText, getText, getText, getTexts, getTexts, hasActionErrors, hasActionMessages, hasErrors, hasFieldErrors, hasKey, input, pause, setActionErrors, setActionMessages, setContainer, setFieldErrors, validate

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

RegisterMemberAction

public **RegisterMemberAction**()

Method Detail

getLog

public LogService **getLog**()

Getter de la Infraestructura de Log

Returns:

Log

setLog

`public void` **`setLog`**`(LogService log)`

Setter de la infraestructura de Log

Parameters:

`log` -

---

getProjectService

`public ProjectService` **`getProjectService`**`()`

Getter para la obtención de la interfaz para los servicios aplicados al modelo de proyecto

Returns:

ProjectService Objeto que hace referencia a la interfaz

---

setProjectService

`public void` **`setProjectService`**`(ProjectService projectService)`

Setter para establecer la interfaz para los servicios aplicados al modelo de proyecto

Parameters:

`projectService` - objecto interfaz

---

getUserService

`public UserService` **`getUserService`**`()`

Getter para la obtención de la interfaz para los servicios aplicados al modelo de usuarios

Returns:

UserService Objeto que hace referencia a la interfaz

---

setUserService

`public void` **`setUserService`**`(UserService userService)`

Setter para establecer la interfaz para los servicios aplicados al modelo de usuarios

Parameters:

`userService` - objecto interfaz

---

setServletRequest

`public void` **`setServletRequest`**`(javax.servlet.http.HttpServletRequest httpServletRequest)`

Setter que establece la solicitud al objeto HTTP en las clases de la aplicación

Specified by:

`setServletRequest` in interface `org.apache.struts2.interceptor.ServletRequestAware`

Parameters:

`httpServletRequest` - petición del Servlet

---

execute

```
public java.lang.String execute()
```

Método que implementa la ejecución del action de Struts con los datos del request

Specified by:

`execute` in interface `com.opensymphony.xwork2.Action`

Overrides:

`execute` in class `com.opensymphony.xwork2.ActionSupport`

Returns:

String cadena que será procesada en struts.xml

---

getProjectRegister

```
public java.lang.Long getProjectRegister()
```

---

setProjectRegister

```
public void setProjectRegister(java.lang.Long projectRegister)
```

---

getLoginRegister

```
public java.lang.String getLoginRegister()
```

---

setLoginRegister

```
public void setLoginRegister(java.lang.String loginRegister)
```

---

getPasswordRegister

```
public java.lang.String getPasswordRegister()
```

---

setPasswordRegister

```
public void setPasswordRegister(java.lang.String passwordRegister)
```

---

getLanguageRegister

```
public java.lang.String getLanguageRegister()
```

---

setLanguageRegister

```
public void setLanguageRegister(java.lang.String languageRegister)
```

---

getEmailRegister

```
public java.lang.String getEmailRegister()
```

---

setEmailRegister

```
public void setEmailRegister(java.lang.String emailRegister)
```

---

isAdminRegister

```
public boolean isAdminRegister()
```

---

setAdminRegister

```
public void setAdminRegister(boolean adminRegister)
```

---

isManagerRegister

```
public boolean isManagerRegister()
```

---

setManagerRegister

```
public void setManagerRegister(boolean managerRegister)
```

---

getStepProjectRegister

```
public java.lang.Integer getStepProjectRegister()
```

---

setStepProjectRegister

```
public void setStepProjectRegister(java.lang.Integer stepProjectRegister)
```

---

getModel

```
public User getModel()
```

Getter para obtener el modelo que se inserta en la ValueStack en lugar de la propia acción

Specified by:

`getModel` in interface `com.opensymphony.xwork2.ModelDriven<User>`

Returns:

User modelo de usuario

7.4.2.5.4 Clase SeekManagerFreeAction

```
java.lang.Object
  └ com.opensymphony.xwork2.ActionSupport
      └ impl.miw.presentation.register.SeekManagerFreeAction
```

All Implemented Interfaces:

com.opensymphony.xwork2.Action, com.opensymphony.xwork2.LocaleProvider, com.opensymphony.xwork2.TextProvider, com.opensymphony.xwork2.Validateable, com.opensymphony.xwork2.ValidationAware, java.io.Serializable, org.apache.struts2.interceptor.ServletRequestAware

---

```
public class SeekManagerFreeAction
extends com.opensymphony.xwork2.ActionSupport
implements org.apache.struts2.interceptor.ServletRequestAware
```

Clase de la capa de presentación para la acción de búsqueda de Manager sin proyecto asignado en la aplicación, extiende de ActionSupport que nos proporciona una implementación por defecto para las acciones más comunes con implementación de una interface "aware" para alojar objetos que puedan estar a disposición en otras partes de la aplicación

Author:

Pablo

See Also:

Serialized Form

Field Summary

Fields inherited from interface com.opensymphony.xwork2.Action

ERROR, INPUT, LOGIN, NONE, SUCCESS

Constructor Summary

SeekManagerFreeAction()

Method Summary

java.lang.String	**execute**() Método que implementa la ejecución del action de Struts con los datos del request
java.lang.String	**getAddedManager**()
LogService	**getLog**() Getter de la Infraestructura de Log
UserService	**getUserService**() Getter para la obtención de la interfaz para los servicios aplicados al modelo de usuarios
void	**setAddedManager**(java.lang.String addedManager)
void	**setLog**(LogService log) Setter de la infraestructura de Log
void	**setServletRequest**(javax.servlet.http.HttpServletRequest httpServletRequest) Setter que establece la solicitud al objeto HTTP en las clases de la aplicación
void	**setUserService**(UserService userService)

	Setter para establecer la interfaz para los servicios aplicados al modelo de usuarios

Methods inherited from class com.opensymphony.xwork2.ActionSupport
addActionError, addActionMessage, addFieldError, clearActionErrors, clearErrors, clearErrorsAndMessages, clearFieldErrors, clearMessages, clone, doDefault, getActionErrors, getActionMessages, getErrorMessages, getErrors, getFieldErrors, getFormatted, getLocale, getText, getText, getText, getText, getText, getText, getText, getText, getText, getTexts, getTexts, hasActionErrors, hasActionMessages, hasErrors, hasFieldErrors, hasKey, input, pause, setActionErrors, setActionMessages, setContainer, setFieldErrors, validate
Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

SeekManagerFreeAction

public **SeekManagerFreeAction**()

Method Detail

getLog

public LogService **getLog**()

Getter de la Infraestructura de Log

Returns:

Log

---

setLog

public void **setLog**(LogService log)

Setter de la infraestructura de Log

Parameters:

log -

---

getUserService

public UserService **getUserService**()

Getter para la obtención de la interfaz para los servicios aplicados al modelo de usuarios

Returns:

UserService Objeto que hace referencia a la interfaz

setUserService

`public void` **`setUserService`**`(UserService userService)`

Setter para establecer la interfaz para los servicios aplicados al modelo de usuarios

Parameters:

`userService` - objecto interfaz

setServletRequest

`public void` **`setServletRequest`**`(javax.servlet.http.HttpServletRequest httpServletRequest)`

Setter que establece la solicitud al objeto HTTP en las clases de la aplicación

Specified by:

`setServletRequest` in interface `org.apache.struts2.interceptor.ServletRequestAware`

Parameters:

`httpServletRequest` - petición del Servlet

execute

`public java.lang.String` **`execute`**`()`

Método que implementa la ejecución del action de Struts con los datos del request

Specified by:

`execute` in interface `com.opensymphony.xwork2.Action`

Overrides:

`execute` in class `com.opensymphony.xwork2.ActionSupport`

Returns:

String cadena que será procesada en struts.xml

getAddedManager

`public java.lang.String` **`getAddedManager`**`()`

setAddedManager

`public void` **`setAddedManager`**`(java.lang.String addedManager)`

7.4.2.6 Paquete impl.miw.presentation.PDF

Class Summary

CreateAction	Clase de la capa de presentación para la acción de creación de PDF de los planes, identificación/análisis y respuesta de riesgos que se encuentran almacenados en la base de datos para crear con ellos los informes, extiende de ActionSupport que nos proporciona una implementación por defecto para las acciones más comunes con implementación de dos interfaces “aware” para alojar objetos que puedan estar a disposición en otras partes de la aplicación.

LoadAction	Clase de la capa de presentación para la acción de buscar todos los datos tanto de los planes, identificación/análisis y respuesta de riesgos que se encuentran almacenados en la base de datos y exponerlos en la aplicación, extiende de ActionSupport que nos proporciona una implementación por defecto para las acciones más comunes con implementación de dos interfaces "aware" para alojar objetos que puedan estar a disposición en otras partes de la aplicación.

7.4.2.6.1 Clase CreateAction

```
java.lang.Object
  └ com.opensymphony.xwork2.ActionSupport
      └ impl.miw.presentation.pdf.CreateAction
```

All Implemented Interfaces:

com.opensymphony.xwork2.Action, com.opensymphony.xwork2.LocaleProvider, com.opensymphony.xwork2.TextProvider, com.opensymphony.xwork2.Validateable, com.opensymphony.xwork2.ValidationAware, java.io.Serializable, org.apache.struts2.interceptor.ApplicationAware, org.apache.struts2.interceptor.ServletRequestAware

---

```
public class CreateAction
extends com.opensymphony.xwork2.ActionSupport
implements org.apache.struts2.interceptor.ApplicationAware,
org.apache.struts2.interceptor.ServletRequestAware
```

Clase de la capa de presentación para la acción de creación de PDF de los planes, identificación/análisis y respuesta de riesgos que se encuentran almacenados en la base de datos para crear con ellos los informes, extiende de ActionSupport que nos proporciona una implementación por defecto para las acciones más comunes con implementación de dos interfaces "aware" para alojar objetos que puedan estar a disposición en otras partes de la aplicación.

Author:

Pablo

See Also:

Serialized Form

Field Summary

Fields inherited from interface com.opensymphony.xwork2.Action

ERROR, INPUT, LOGIN, NONE, SUCCESS

Constructor Summary

CreateAction()

Method Summary

java.lang.String	**execute**() Método que implementa la ejecución del action de Struts con los datos del request
InfoService	**getInfoService**() Getter para la obtención de la interfaz para los servicios aplicados al modelo de info que se utiliza para la información de los riesgos
LogService	**getLog**() Getter de la Infraestructura de Log
java.lang.Integer	**getNum**()
java.lang.Integer	**getOption**()
ProjectService	**getProjectService**() Getter para la obtención de la interfaz para los servicios aplicados al modelo de proyecto
java.lang.String	**getTitle**()
void	**setApplication**(java.util.Map<java.lang.String,java.lang.Object> arg0) Setter que establece el mapa de propiedades de la aplicación.
void	**setInfoService**(InfoService infoService) Setter para establecer la interfaz para los servicios aplicados al modelo de info que se utiliza para la información de los riesgos
void	**setLog**(LogService log) Setter de la infraestructura de Log
void	**setNum**(java.lang.Integer num)
void	**setOption**(java.lang.Integer option)
void	**setProjectService**(ProjectService projectService) Setter para establecer la interfaz para los servicios aplicados al modelo de proyecto
void	**setServletRequest**(javax.servlet.http.HttpServletRequest httpServletRequest)

	Setter que establece la solicitud al objeto HTTP en las clases de la aplicación
void	**setTitle**(java.lang.String title)

Methods inherited from class com.opensymphony.xwork2.ActionSupport
addActionError, addActionMessage, addFieldError, clearActionErrors, clearErrors, clearErrorsAndMessages, clearFieldErrors, clearMessages, clone, doDefault, getActionErrors, getActionMessages, getErrorMessages, getErrors, getFieldErrors, getFormatted, getLocale, getText, getText, getText, getText, getText, getText, getText, getText, getText, getTexts, getTexts, hasActionErrors, hasActionMessages, hasErrors, hasFieldErrors, hasKey, input, pause, setActionErrors, setActionMessages, setContainer, setFieldErrors, validate
Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

CreateAction

public **CreateAction**()

Method Detail

getLog

public LogService **getLog**()

Getter de la Infraestructura de Log

Returns:

Log

---

setLog

public void **setLog**(LogService log)

Setter de la infraestructura de Log

Parameters:

log -

---

getInfoService

public InfoService **getInfoService**()

Getter para la obtención de la interfaz para los servicios aplicados al modelo de info que se utiliza para la información de los riesgos

Returns:

InfoService Objeto que hace referencia a la interfaz

---

setInfoService

```
public void setInfoService(InfoService infoService)
```

Setter para establecer la interfaz para los servicios aplicados al modelo de info que se utiliza para la información de los riesgos

Parameters:

`infoService` - objecto interfaz

---

setApplication

```
public void setApplication(java.util.Map<java.lang.String,java.lang.Object> arg0)
```

Setter que establece el mapa de propiedades de la aplicación. Esto dará acceso a un mapa en el que puedan poner objetos que deben estar a disposición en otras partes de la aplicación.

Specified by:

`setApplication` in interface `org.apache.struts2.interceptor.ApplicationAware`

Parameters:

`arg0` - un mapa de propiedades de la aplicación.

---

getProjectService

```
public ProjectService getProjectService()
```

Getter para la obtención de la interfaz para los servicios aplicados al modelo de proyecto

Returns:

ProjectService Objeto que hace referencia a la interfaz

---

setProjectService

```
public void setProjectService(ProjectService projectService)
```

Setter para establecer la interfaz para los servicios aplicados al modelo de proyecto

Parameters:

`projectService` - objecto interfaz

---

setServletRequest

```
public void setServletRequest(javax.servlet.http.HttpServletRequest httpServletRequest)
```

Setter que establece la solicitud al objeto HTTP en las clases de la aplicación

Specified by:

`setServletRequest` in interface `org.apache.struts2.interceptor.ServletRequestAware`

Parameters:

`httpServletRequest` - petición del Servlet

---

execute

```
public java.lang.String execute()
```

Método que implementa la ejecución del action de Struts con los datos del request

Specified by:

`execute` in interface `com.opensymphony.xwork2.Action`

Overrides:

`execute` in class `com.opensymphony.xwork2.ActionSupport`

Returns:

String cadena que será procesada en struts.xml

---

getOption

```
public java.lang.Integer getOption()
```

---

setOption

```
public void setOption(java.lang.Integer option)
```

---

getNum

```
public java.lang.Integer getNum()
```

---

setNum

```
public void setNum(java.lang.Integer num)
```

---

getTitle

```
public java.lang.String getTitle()
```

---

setTitle

```
public void setTitle(java.lang.String title)
```

7.4.2.6.2 Clase LoadAction

```
java.lang.Object
  └ com.opensymphony.xwork2.ActionSupport
      └ impl.miw.presentation.pdf.LoadAction
```

All Implemented Interfaces:

com.opensymphony.xwork2.Action, com.opensymphony.xwork2.LocaleProvider, com.opensymphony.xwork2.TextProvider, com.opensymphony.xwork2.Validateable, com.opensymphony.xwork2.ValidationAware, java.io.Serializable, org.apache.struts2.interceptor.ApplicationAware, org.apache.struts2.interceptor.ServletRequestAware

---

```
public class LoadAction
extends com.opensymphony.xwork2.ActionSupport
implements            org.apache.struts2.interceptor.ApplicationAware,
org.apache.struts2.interceptor.ServletRequestAware
```

Clase de la capa de presentación para la acción de buscar todos los datos tanto de los planes, identificación/análisis y respuesta de riesgos que se encuentran almacenados en la base de datos y exponerlos en la aplicación, extiende de ActionSupport que nos proporciona una implementación por defecto para las acciones más comunes con implementación de dos interfaces "aware" para alojar objetos que puedan estar a disposición en otras partes de la aplicación.

Author:

Pablo

See Also:

Serialized Form

Field Summary

Fields inherited from interface com.opensymphony.xwork2.Action

ERROR, INPUT, LOGIN, NONE, SUCCESS

Constructor Summary

LoadAction()

Method Summary

java.lang.String	**execute**() Método que implementa la ejecución del action de Struts con los datos del request
InfoService	**getInfoService**() Getter para la obtención de la interfaz para los servicios aplicados al modelo de info que se utiliza para la información de los riesgos
LogService	**getLog**() Getter de la Infraestructura de Log
void	**setApplication**(java.util.Map<java.lang.String,java.lang.Object> arg0) Setter que establece el mapa de propiedades de la aplicación.
void	**setInfoService**(InfoService infoService) Setter para establecer la interfaz para los servicios aplicados al modelo de info que se utiliza para la información de los riesgos
void	**setLog**(LogService log)

	Setter de la infraestructura de Log
void	**setServletRequest**(javax.servlet.http.HttpServletRequest httpServletRequest) Setter que establece la solicitud al objeto HTTP en las clases de la aplicación

Methods inherited from class com.opensymphony.xwork2.ActionSupport
addActionError, addActionMessage, addFieldError, clearActionErrors, clearErrors, clearErrorsAndMessages, clearFieldErrors, clearMessages, clone, doDefault, getActionErrors, getActionMessages, getErrorMessages, getErrors, getFieldErrors, getFormatted, getLocale, getText, getText, getText, getText, getText, getText, getText, getText, getText, getTexts, getTexts, hasActionErrors, hasActionMessages, hasErrors, hasFieldErrors, hasKey, input, pause, setActionErrors, setActionMessages, setContainer, setFieldErrors, validate
Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

LoadAction

public **LoadAction**()

Method Detail

getLog

public LogService **getLog**()

Getter de la Infraestructura de Log

Returns:

Log

---

setLog

public void **setLog**(LogService log)

Setter de la infraestructura de Log

Parameters:

log -

---

getInfoService

public InfoService **getInfoService**()

Getter para la obtención de la interfaz para los servicios aplicados al modelo de info que se utiliza para la información de los riesgos

Returns:

InfoService Objeto que hace referencia a la interfaz

---

setInfoService

`public void` **`setInfoService`**`(InfoService infoService)`

Setter para establecer la interfaz para los servicios aplicados al modelo de info que se utiliza para la información de los riesgos

Parameters:

`infoService` - objecto interfaz

---

setApplication

`public void` **`setApplication`**`(java.util.Map<java.lang.String,java.lang.Object> arg0)`

Setter que establece el mapa de propiedades de la aplicación. Esto dará acceso a un mapa en el que puedan poner objetos que deben estar a disposición en otras partes de la aplicación.

Specified by:

`setApplication` in interface `org.apache.struts2.interceptor.ApplicationAware`

Parameters:

`arg0` - un mapa de propiedades de la aplicación.

---

setServletRequest

`public void` **`setServletRequest`**`(javax.servlet.http.HttpServletRequest httpServletRequest)`

Setter que establece la solicitud al objeto HTTP en las clases de la aplicación

Specified by:

`setServletRequest` in interface `org.apache.struts2.interceptor.ServletRequestAware`

Parameters:

`httpServletRequest` - petición del Servlet

---

execute

`public java.lang.String` **`execute`**`()`

Método que implementa la ejecución del action de Struts con los datos del request

Specified by:

`execute` in interface `com.opensymphony.xwork2.Action`

Overrides:

`execute` in class `com.opensymphony.xwork2.ActionSupport`

Returns:

String cadena que será procesada en struts.xml

7.4.2.7 Paquete com.miw.Business

En este paquete se encuentran las clases donde están las interfaces entre la capa de presentación y la de negocio, se mostrará una de ellas, el resto podrá consultarse en la documentación del **JavaDoc** las cuales son:

- **InfoService**
- ProjectService
- UserService

Interface Summary	
InfoService	Facade.
ProjectService	Facade.
UserService	Facade.

7.4.2.7.1 Interface InfoService

All Known Implementing Classes:

Gestion

`public interface` **`InfoService`**

Facade. Interfaz entre la capa de Presentación y Negocio para datos de riesgos

Author:

Pablo

Method Summary	
`java.util.Vector<Iden>`	**`getIden`**`(java.lang.Long id, java.lang.Long idP)`
`Info`	**`getInfo`**`(java.lang.Long id, java.lang.Long idP, java.lang.Double version)`
`Respuesta`	**`getResp`**`(java.lang.Long id, java.lang.Long idP, java.lang.Integer option)`

java.util.Vector<java.lang.Double>	**getVersion**(java.lang.Long id, java.lang.Long idP)
java.util.Vector<java.lang.Double>	**getVersionIden**(java.lang.Long idU, java.lang.Long idP)
void	**setIden**(java.util.Vector<Iden> vIden)
void	**setInfo**(java.lang.Long id, java.lang.Long idP, java.lang.Double v, Info info)
void	**setRespuesta**(Respuesta resp)
void	**setUpdateInfo**(Info data)

Method Detail

setUpdateInfo

```
void setUpdateInfo(Info data)
                   throws java.lang.Exception
```

Throws:

java.lang.Exception

---

getInfo

```
Info getInfo(java.lang.Long id,
             java.lang.Long idP,
             java.lang.Double version)
             throws java.lang.Exception
```

Throws:

java.lang.Exception

---

setIden

```
void setIden(java.util.Vector<Iden> vIden)
             throws java.lang.Exception
```

Throws:

java.lang.Exception

---

getIden

```
java.util.Vector<Iden> getIden(java.lang.Long id,
                               java.lang.Long idP)
                               throws java.lang.Exception
```

Throws:

java.lang.Exception

setRespuesta

```
void setRespuesta(Respuesta resp)
                  throws java.lang.Exception
```

Throws:

`java.lang.Exception`

getResp

```
Respuesta getResp(java.lang.Long id,
                  java.lang.Long idP,
                  java.lang.Integer option)
                  throws java.lang.Exception
```

Throws:

`java.lang.Exception`

getVersion

```
java.util.Vector<java.lang.Double> getVersion(java.lang.Long id,
                                              java.lang.Long idP)
                                              throws
java.lang.Exception
```

Throws:

`java.lang.Exception`

setInfo

```
void setInfo(java.lang.Long id,
             java.lang.Long idP,
             java.lang.Double v,
             Info info)
             throws java.lang.Exception
```

Throws:

`java.lang.Exception`

getVersionIden

```
java.util.Vector<java.lang.Double> getVersionIden(java.lang.Long idU,
                                                  java.lang.Long idP)
                                                  throws
java.lang.Exception
```

Throws:

`java.lang.Exception`

7.4.2.8 Paquete impl.miw.Business

Del paquete de negocio de la implementación se encuentran varios paquetes, que serán para la lógica de:

- project
- usuario
- gestión
- PDF

Se mostraran las clases de proyectos, la cual sigue el esquema de los usuarios y de los riesgos, y las clases del paquete de la lógica del PDF, por ser algo distinta al resto, el resto puede verse en la documentación del JavaDoc del proyecto.

Class Summary

ProjectRisk	Clase que implementa las operaciones para la capa de Negocio para los proyectos

7.4.2.8.1 Clase ProjectRisk

```
java.lang.Object
  └ impl.miw.business.project.ProjectRisk
```

All Implemented Interfaces:

ProjectService

```
public class ProjectRisk
extends java.lang.Object
implements ProjectService
```

Clase que implementa las operaciones para la capa de Negocio para los proyectos

Author:

Pablo

Constructor Summary

ProjectRisk()

Method Summary

java.lang.Boolean	**deleteProject**(java.lang.Long idProject) Método de la interfaz entre presentación y negocio para eliminar el proyecto
LogService	**getLog**() Getter de la Infraestructura de Log
Project	**getProject**(java.lang.Long idP) Getter de la interfaz entre presentación y negocio para obtener los datos del proyecto por medio de su identificador
Project	**getProject**(User user)

	Getter de la interfaz entre presentación y negocio para obtener los datos del proyecto
ProjectDataService	**getProjectDataService**() Getter de la interfaz entre el negocio y la persistencia de datos del proyecto
java.util.Vector<Project>	**getProjects**() Getter de la interfaz entre presentación y negocio para obtener los datos de los proyectos
java.lang.String	**seekProject**(java.lang.Long idProjectUserUpdate) Método de la interfaz entre presentación y negocio para la busqueda de proyectos por identificador
void	**setLog**(LogService log) Setter de la infraestructura de Log
java.lang.String	**setProject**(Project project) Setter de la interfaz entre presentación y negocio para establecer los datos de los proyectos
void	**setProjectDataService**(ProjectDataService project DataService) Setter de la interfaz entre el negocio y la persistencia de datos del proyecto
java.lang.String	**setUpdateProject**(Project updateProject) Método de la interfaz entre presentación y negocio para la actualización de datos del proyecto
void	**updateStep**(java.lang.Long idP, java.lang.Integer step) Método para actualizar el paso actual del proyecto

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

ProjectRisk

public **ProjectRisk**()

Method Detail

getLog

public LogService **getLog**()

Getter de la Infraestructura de Log

Returns:

Log

setLog

`public void **setLog**(LogService log)`

Setter de la infraestructura de Log

Parameters:

`log` -

getProjectDataService

`public ProjectDataService **getProjectDataService**()`

Getter de la interfaz entre el negocio y la persistencia de datos del proyecto

Returns:

ProjectDataService interfaz entre el negocio y la persistencia

setProjectDataService

`public void **setProjectDataService**(ProjectDataService projectDataService)`

Setter de la interfaz entre el negocio y la persistencia de datos del proyecto

Parameters:

`projectDataService` - interfaz entre el negocio y la persistencia

getProjects

`public java.util.Vector<Project> **getProjects**() throws java.lang.Exception`

Getter de la interfaz entre presentación y negocio para obtener los datos de los proyectos

Specified by:

`getProjects` in interface `ProjectService`

Returns:

Vector vector con información de los proyectos

Throws:

`java.lang.Exception`

setProject

`public java.lang.String **setProject**(Project project) throws java.lang.Exception`

Setter de la interfaz entre presentación y negocio para establecer los datos de los proyectos

Specified by:

`setProject` in interface `ProjectService`

Parameters:

`project` - datos con el proyecto

Returns:

String cadena para el log

Throws:

`java.lang.Exception`

getProject

```
public Project getProject(User user)
                  throws java.lang.Exception
```

Getter de la interfaz entre presentación y negocio para obtener los datos del proyecto

Specified by:

`getProject` in interface `ProjectService`

Parameters:

`user` - usuario de ese proyecto

Returns:

Project proyecto

Throws:

`java.lang.Exception`

---

deleteProject

```
public java.lang.Boolean deleteProject(java.lang.Long idProject)
                                 throws java.lang.Exception
```

Método de la interfaz entre presentación y negocio para eliminar el proyecto

Specified by:

`deleteProject` in interface `ProjectService`

Parameters:

`idProject` - identificación del proyecto

Returns:

Booelan de comprobación para el borrado

Throws:

`java.lang.Exception`

---

setUpdateProject

```
public java.lang.String setUpdateProject(Project updateProject)
                                  throws java.lang.Exception
```

Método de la interfaz entre presentación y negocio para la actualización de datos del proyecto

Specified by:

`setUpdateProject` in interface `ProjectService`

Parameters:

`updateProject` - proyecto a actualizar

Returns:

String valor devuelto para el log

Throws:

`java.lang.Exception`

---

seekProject

```
public                                                  java.lang.String
seekProject(java.lang.Long idProjectUserUpdate)
                             throws java.lang.Exception
```

Método de la interfaz entre presentación y negocio para la busqueda de proyectos por identificador

Specified by:

`seekProject` in interface `ProjectService`

Parameters:

`idProjectUserUpdate` - identificador del usuario del proyecto

Returns:

String cadena para comprobación

Throws:

`java.lang.Exception`

---

getProject

```
public Project getProject(java.lang.Long idP)
                   throws java.lang.Exception
```

Getter de la interfaz entre presentación y negocio para obtener los datos del proyecto por medio de su identificador

Specified by:

`getProject` in interface `ProjectService`

Parameters:

`idP` - identificador del proyecto

Returns:

Project proyecto

Throws:

`java.lang.Exception`

---

updateStep

```
public void updateStep(java.lang.Long idP,
                       java.lang.Integer step)
                throws java.lang.Exception
```

Método para actualizar el paso actual del proyecto

Specified by:

`updateStep` in interface `ProjectService`

Parameters:

`idP` - identificador del proyecto

`step` - paso a actualizar en el proyecto

Throws:

`java.lang.Exception`

Class Summary

CreatePDF	Clase de la capa de negocio para la acción de creación de pdfs según los datos facilitados y extraídos desde la BBDD de la aplicación, para la generación de los informes de los mismos.

7.4.2.8.2 Clase CreatePDF

```
java.lang.Object
  └ impl.miw.business.pdf.CreatePDF
```

```
public class CreatePDF
extends java.lang.Object
```

Clase de la capa de negocio para la acción de creación de pdfs según los datos facilitados y extraídos desde la BBDD de la aplicación, para la generación de los informes de los mismos.

Author:

Pablo

Constructor Summary

CreatePDF()

Method Summary

static java.lang.String	**convertFromUTF8**(java.lang.String s) Método para cambiar el formato UTF-8 a ISO-8859-1
java.lang.String	**createPDF**(Info info, User u, Project project, java.lang.String file) Método para la creación de los pdf para los planes de riesgos
java.lang.String	**createPDF**(Respuesta resp, User u, Project project, java.lang.String file, java.lang.Double version, java.util.Vector<Impacto> impactos, java.util.Vector<Probabilidad> probabilidad, java.lang.Double corte) Método para la creación de los pdf para las respuestas de riesgos
java.lang.String	**createPDF**(java.util.Vector<Iden> iden, User u, Project project, java.lang.String file, java.util.Vector<Impacto> impactos, java.util.Vector<Probabilidad> probabilidad, java.lang.Double corte) Método para la creación de los pdf para los riesgos identificados

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

CreatePDF

```
public CreatePDF()
```

Method Detail

convertFromUTF8

```
public static java.lang.String convertFromUTF8(java.lang.String s)
```

Método para cambiar el formato UTF-8 a ISO-8859-1

Parameters:

`s` - String con la cadena a convertir

Returns:

String con la conversión realizada

---

createPDF

```
public java.lang.String createPDF(Info info,
                                  User u,
                                  Project project,
                                  java.lang.String file)
                           throws java.io.FileNotFoundException,
                                  com.itextpdf.text.DocumentException,
                                  java.io.IOException
```

Método para la creación de los pdf para los planes de riesgos

Parameters:

`info` - objecto Info donde se encuentra la información del plan de riesgos

`u` - objeto usuario

`project` - objeto proyecto

`file` - cadena con el nombre del fichero

Returns:

String con el path del fichero creado

Throws:

`java.io.FileNotFoundException`
`com.itextpdf.text.DocumentException`
`java.io.IOException`

---

createPDF

```
public java.lang.String createPDF(java.util.Vector<Iden> iden,
                                  User u,
                                  Project project,
                                  java.lang.String file,
                                  java.util.Vector<Impacto> impactos,
```

```
java.util.Vector<Probabilidad> probabilidad,
                                java.lang.Double corte)
                          throws java.io.FileNotFoundException,
                                com.itextpdf.text.DocumentException,
                                java.io.IOException
```

Método para la creación de los pdf para los riesgos identificados

Parameters:

`iden` - objecto Iden donde se encuentra la información de la identificación de riesgos

`u` - objeto usuario

`project` - objeto proyecto

`file` - cadena con el nombre del fichero

`impactos` - vector con los impactos

`probabilidad` - vector con las probabilidades

`corte` - entero con el número de corte de los riesgos

Returns:

String con el path del fichero creado

Throws:

`java.io.FileNotFoundException`
`com.itextpdf.text.DocumentException`
`java.io.IOException`

---

createPDF

```
public java.lang.String createPDF(Respuesta resp,
                                User u,
                                Project project,
                                java.lang.String file,
                                java.lang.Double version,
                                java.util.Vector<Impacto> impactos,

java.util.Vector<Probabilidad> probabilidad,
                                java.lang.Double corte)
                          throws java.io.FileNotFoundException,
                                com.itextpdf.text.DocumentException,
                                java.io.IOException
```

Método para la creación de los pdf para las respuestas de riesgos

Parameters:

`resp` - objecto respuesta donde se encuentra la información

`u` - objeto usuario

`project` - objeto proyecto

`file` - cadena con el nombre del fichero

`version` - cadena con la versión

`impactos` - vector con los impactos

`probabilidad` - vector con las probabilidades

`corte` - entero con el número de corte de los riesgos

Returns:

String con el path del fichero creado

Throws:

`java.io.FileNotFoundException`
`com.itextpdf.text.DocumentException`

java.io.IOException

7.4.2.9 Paquete com.miw.Persistence

En este paquete se encuentran las clases donde están las interfaces que comunican la capa de negocio con la de persistencia, se mostrará una de ellas, el resto podrá consultarse en la documentación del **JavaDoc** las cuales son:

- **InfoDataService**
- ProjectDataService
- UserDataService

Interface Summary	
InfoDataService	Facade.
ProjectDataService	Facade.
UserDataService	Facade.

7.4.2.9.1 Interface InfoDataService

All Known Implementing Classes:

InfoDAO

---

public interface **InfoDataService**
Facade. Interfaz entre la capa de Negocio y Persistencia para datos de riesgos

Author:

Pablo

Method Summary	
java.util.Vector<Iden>	**getIden**(java.lang.Long id, java.lang.Long idP)
Info	**getInfo**(java.lang.Long id, java.lang.Long idP, java.lang.Double version)
Respuesta	**getResp**(java.lang.Long id, java.lang.Long idP,

	java.lang.Integer op)
java.util.Vector<java.lang.Double>	**getVersion**(java.lang.Long id, java.lang.Long idP)
java.util.Vector<java.lang.Double>	**getVersionIden**(java.lang.Long id, java.lang.Long idP)
void	**setIden**(java.util.Vector<Iden> vIden)
void	**setInfo**(java.lang.Long id, java.lang.Long idP, java.lang.Double v, Info data)
void	**setRespuesta**(Respuesta resp)
void	**setUpdateInfo**(Info data)

Method Detail

setUpdateInfo

```
void setUpdateInfo(Info data)
                   throws java.lang.Exception
```

Throws:

java.lang.Exception

---

getInfo

```
Info getInfo(java.lang.Long id,
             java.lang.Long idP,
             java.lang.Double version)
             throws java.lang.Exception
```

Throws:

java.lang.Exception

---

setIden

```
void setIden(java.util.Vector<Iden> vIden)
             throws java.lang.Exception
```

Throws:

java.lang.Exception

---

getIden

```
java.util.Vector<Iden> getIden(java.lang.Long id,
                               java.lang.Long idP)
                               throws java.lang.Exception
```

Throws:
java.lang.Exception

---

setRespuesta

```
void setRespuesta(Respuesta resp)
                  throws java.lang.Exception
```

Throws:
java.lang.Exception

---

getResp

```
Respuesta getResp(java.lang.Long id,
                  java.lang.Long idP,
                  java.lang.Integer op)
                  throws java.lang.Exception
```

Throws:
java.lang.Exception

---

getVersion

```
java.util.Vector<java.lang.Double> getVersion(java.lang.Long id,
                                               java.lang.Long idP)
                                               throws
java.lang.Exception
```

Throws:
java.lang.Exception

---

setInfo

```
void setInfo(java.lang.Long id,
             java.lang.Long idP,
             java.lang.Double v,
             Info data)
             throws java.lang.Exception
```

Throws:
java.lang.Exception

---

getVersionIden

```
java.util.Vector<java.lang.Double> getVersionIden(java.lang.Long id,
                                                   java.lang.Long idP)
                                                   throws
java.lang.Exception
```

Throws:
java.lang.Exception

7.4.2.10 Paquete impl.miw.Persistence

En el paquete de persistencia al igual que el de negocio de la implementación, se encuentran los paquetes por cada uno de los Bean que creados con Spring,

- **info**
- project
- user

Se describirá uno de ellos el referente a la gestión de riesgos a demás de la clase, para la conexión de base de datos, el resto podrá consultarse en el Javadoc de la documentación.

Class Summary

InfoDAO	Clase que implementa las operaciones de acceso a la base de datos para las entidades relacionadas con los RIESGOS.

7.4.2.10.1 Clase InfoDAO

```
java.lang.Object
  └ impl.miw.persistence.info.InfoDAO
```

All Implemented Interfaces:

InfoDataService

```
public class InfoDAO
extends java.lang.Object
implements InfoDataService
```

Clase que implementa las operaciones de acceso a la base de datos para las entidades relacionadas con los RIESGOS.

Author:

Pablo

Constructor Summary

InfoDAO()

Method Summary

java.util.Vector<Iden>	**getIden**(java.lang.Long id, java.lang.Long idP)

	Getter para la obtención de las identificaciones de los riesgos
Info	**getInfo**(java.lang.Long id, java.lang.Long idP, java.lang.Double version) Getter para la obtención de los datos del plan de riesgo
Respuesta	**getResp**(java.lang.Long id, java.lang.Long idP, java.lang.Integer op) Getter para la obtención de las respuestas de los riesgos
java.util.Vector<java.lang.Double>	**getVersion**(java.lang.Long id, java.lang.Long idP) Getter para la obtención de las versiones de los plnaes de versiones
java.util.Vector<java.lang.Double>	**getVersionIden**(java.lang.Long id, java.lang.Long idP) Getter para la obtención de las versiones de la identificación de riesgos
void	**setIden**(java.util.Vector<Iden> vIden) Setter para establecer los datos de las identificaciones de los riesgos
void	**setInfo**(java.lang.Long id, java.lang.Long idP, java.lang.Double version, Info data) Setter de la información del riesgo, mediante la identificación del usuario del proyecto y de la versión
void	**setRespuesta**(Respuesta resp) Setter para establecer los datos de la respuesta de riesgos
void	**setUpdateInfo**(Info data) Setter para establecer los datos de actualización de los planes de riesgos en la base de datos

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

InfoDAO

```
public InfoDAO()
```

Method Detail

setInfo

```
public void setInfo(java.lang.Long id,
                    java.lang.Long idP,
                    java.lang.Double version,
                    Info data)
             throws java.lang.Exception
```

Setter de la información del riesgo, mediante la identificación del usuario del proyecto y de la versión

Specified by:

`setInfo` in interface `InfoDataService`

Parameters:

`id` - identificación del usuario

`idP` - identificación del proyecto

`version` - versión

`data` - Datos del riesgo

Throws:

`java.lang.Exception`

---

setUpdateInfo

```
public void setUpdateInfo(Info data)
                  throws java.lang.Exception
```

Setter para establecer los datos de actualización de los planes de riesgos en la base de datos

Specified by:

`setUpdateInfo` in interface `InfoDataService`

Parameters:

`data` - datos del riesgo

Throws:

`java.lang.Exception`

---

getInfo

```
public Info getInfo(java.lang.Long id,
                    java.lang.Long idP,
                    java.lang.Double version)
             throws java.lang.Exception
```

Getter para la obtención de los datos del plan de riesgo

Specified by:

`getInfo` in interface `InfoDataService`

Parameters:

`id` - identificación del usuario

`idP` - identificación del proyecto

`version` - versión

Throws:

`java.lang.Exception`

---

setIden

```
public void setIden(java.util.Vector<Iden> vIden)
             throws java.lang.Exception
```

Setter para establecer los datos de las identificaciones de los riesgos

Specified by:

`setIden` in interface `InfoDataService`

Parameters:

`vIden` - vector con las identificaciones

Throws:

`java.lang.Exception`

---

getIden

```
public java.util.Vector<Iden> getIden(java.lang.Long id,
                                      java.lang.Long idP)
                               throws java.lang.Exception
```

Getter para la obtención de las identificaciones de los riesgos

Specified by:

`getIden` in interface `InfoDataService`

Parameters:

`id` - identificación del usuario

`idP` - identificación del proyecto

Returns:

Vector vector con los datos de la identificación de los riesgos

Throws:

`java.lang.Exception`

---

setRespuesta

```
public void setRespuesta(Respuesta resp)
                  throws java.lang.Exception
```

Setter para establecer los datos de la respuesta de riesgos

Specified by:

`setRespuesta` in interface `InfoDataService`

Parameters:

`resp` - datos de la respuesta de riesgos

Throws:

`java.lang.Exception`

---

getResp

```
public Respuesta getResp(java.lang.Long id,
                         java.lang.Long idP,
                         java.lang.Integer op)
                  throws java.lang.Exception
```

Getter para la obtención de las respuestas de los riesgos

Specified by:

getResp in interface InfoDataService

Parameters:

id - identificación del usuario

idP - identificación del proyecto

Returns:

Respuesta datos de la respuesta de los riesgos

Throws:

java.lang.Exception

---

getVersion

```
public java.util.Vector<java.lang.Double>
getVersion(java.lang.Long id,

java.lang.Long idP)
                                                throws
java.lang.Exception
```

Getter para la obtención de las versiones de los plnaes de versiones

Specified by:

getVersion in interface InfoDataService

Parameters:

id - identificación del usuario

idP - identificación del proyecto

Returns:

Vector vector con las diferentes versiones

Throws:

java.lang.Exception

---

getVersionIden

```
public java.util.Vector<java.lang.Double>
getVersionIden(java.lang.Long id,

java.lang.Long idP)
                                                  throws
java.lang.Exception
```

Getter para la obtención de las versiones de la identificación de riesgos

Specified by:

getVersionIden in interface InfoDataService

Parameters:

id - identificación del usuario

idP - identificación del proyecto

Returns:

Vector vector con las diferentes versiones

Throws:

`java.lang.Exception`

Class Summary

Jdbc	Clase para la configuración de la conexión a MYSQL

7.4.2.10.2 Clase Jdbc

```
java.lang.Object
  └ impl.miw.persistence.Jdbc
```

```
public class Jdbc
extends java.lang.Object
```

Clase para la configuración de la conexión a MYSQL

Author:

Pablo

Constructor Summary

Jdbc()

Method Summary

static java.sql.Connection	**getConnection**() Getter que devuelve la conexión

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

Jdbc

```
public Jdbc()
```

Method Detail

getConnection

```
public static java.sql.Connection getConnection()
                                          throws java.io.IOException
```

Getter que devuelve la conexión

Returns:

Connection conexión

Throws:

`java.io.IOException`

7.4.2.11 Paquete com.miw.infrastructure.Log

Interface Summary

LogService	Capa de Infraestructura.

7.4.2.11.1 Interface LogService

All Known Implementing Classes:

MyLogService

```
public interface LogService
```

Capa de Infraestructura. Interfaz para el Servicio de Log

Author:

Pablo

Field Summary

`static int`	**DEBUG**
`static int`	**ERROR**
`static int`	**FATAL_ERROR**
`static int`	**WARN**

Method Summary

void	**debug**(java.lang.String message)
void	**error**(java.lang.String message)
void	**fatalError**(java.lang.String message)
void	**warn**(java.lang.String message)

Field Detail

DEBUG

static final int **DEBUG**

See Also:

Constant Field Values

WARN

static final int **WARN**

See Also:

Constant Field Values

ERROR

static final int **ERROR**

See Also:

Constant Field Values

FATAL_ERROR

static final int **FATAL_ERROR**

See Also:

Constant Field Values

Method Detail

warn

void **warn**(java.lang.String message)

---

error

void **error**(java.lang.String message)

---

debug

void **debug**(java.lang.String message)

---

fatalError

`void` **`fatalError`**`(java.lang.String message)`

Class Summary

MyLogService	Clase para el Servicio de Log

7.4.2.11.2 Clase MyLogService

```
java.lang.Object
  └ com.miw.infrastructure.log.MyLogService
```

All Implemented Interfaces:

LogService

```
public class MyLogService
extends java.lang.Object
implements LogService
```

Clase para el Servicio de Log

Author:

Pablo

Field Summary

Fields inherited from interface com.miw.infrastructure.log.LogService

DEBUG, ERROR, FATAL_ERROR, WARN

Constructor Summary

MyLogService()

Method Summary

void	**crearLog**(java.lang.String message) Método para la creación del Log
void	**debug**(java.lang.String message) Método que sobreescribe el modo Debug

void	**error**(java.lang.String message) Método que sobreescribe el modo Error
void	**fatalError**(java.lang.String message) Método que sobreescribe el modo Error Fatal
int	**getLevel**()
void	**setLevel**(int level)
void	**warn**(java.lang.String message) Método que sobreescribe el modo Warning

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

MyLogService

```
public MyLogService()
```

Method Detail

getLevel

```
public int getLevel()
```

---

setLevel

```
public void setLevel(int level)
```

---

debug

```
public void debug(java.lang.String message)
```

Método que sobreescribe el modo Debug

Specified by:

debug in interface LogService

Parameters:

message -

---

error

```
public void error(java.lang.String message)
```

Método que sobreescribe el modo Error

Specified by:

error in interface LogService

Parameters:

message -

---

fatalError

```
public void fatalError(java.lang.String message)
```

Método que sobreescribe el modo Error Fatal

Specified by:

fatalError in interface LogService

Parameters:

message -

---

warn

```
public void warn(java.lang.String message)
```

Método que sobreescribe el modo Warning

Specified by:

warn in interface LogService

Parameters:

message -

---

crearLog

```
public void crearLog(java.lang.String message)
              throws java.io.IOException
```

Método para la creación del Log

Parameters:

message -

Throws:

java.io.IOException

7.4.2.12 Paquete com.miw.Junit

Class Summary	
AllTests	Clase para producir Tests de JUnit para probar diferentes partes de la aplicación
AssociationsTest	Clase para probar diferentes asociaciones dentro de cada modelo
CreatePDFTest	Clase de Test para probar la creación de ficheros pdf
PersistenceTest	Clase para probar la persistencia

7.4.2.12.1 Clase AllTests

```
java.lang.Object
  └ com.miw.junit.AllTests
```

```
public class AllTests
```

```
extends java.lang.Object
```
Clase para producir Tests de JUnit para probar diferentes partes de la aplicación

Author:

Pablo

Constructor Summary

AllTests()

Method Summary

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

AllTests

```
public AllTests()
```

7.4.2.12.2 Clase AssociationsTest

```
java.lang.Object
  └ com.miw.junit.AssociationsTest
```

```
public class AssociationsTest
extends java.lang.Object
```
Clase para probar diferentes asociaciones dentro de cada modelo

Author:

Pablo

Constructor Summary

AssociationsTest()

Method Summary

void	**setUp**() Método para inicializar variables
void	**testPoseerAdd**() Método de Test para añadir posesión
void	**testPoseerRemove**() Método de Test eliminación de posesión
void	**testTenerAdd**() Método para agregar pertenencias dentro de otro ya agregado
void	**testTenerRemove**() Método para eliminar pertenencias dentro de otro ya agregado

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

AssociationsTest

```
public AssociationsTest()
```

Method Detail

setUp

```
public void setUp()
            throws BusinessException
```

Método para inicializar variables

Throws:

BusinessException

---

testPoseerAdd

```
public void testPoseerAdd()
                    throws BusinessException
```

Método de Test para añadir posesión

Throws:

BusinessException

---

testPoseerRemove

```
public void testPoseerRemove()
                     throws BusinessException
```

Método de Test eliminación de posesión

Throws:

BusinessException

---

testTenerAdd

```
public void testTenerAdd()
                    throws BusinessException
```

Método para agregar pertenencias dentro de otro ya agregado

Throws:

BusinessException

---

testTenerRemove

```
public void testTenerRemove()
                       throws BusinessException
```

Método para eliminar pertenencias dentro de otro ya agregado

Throws:

BusinessException

7.4.2.12.3 Clase CreatePDFTest

```
java.lang.Object
  └ com.miw.junit.CreatePDFTest
```

```
public class CreatePDFTest
extends java.lang.Object
```

Clase de Test para probar la creación de ficheros pdf

Author:

Pablo

Constructor Summary

CreatePDFTest()

Method Summary

void	**ITextHelloWorld**(com.itextpdf.text.Document document) Método de Test para generar diferentes tipos de documentos pdf
void	**setUp**() Método para inicializar variables

boolean	**tablaAlineacion**(com.itextpdf.text.Document document) Método para probar la generación de tablas alineadas
com.itextpdf.text.pdf.PdfPTable	**tablaColor**() Método para probar la generación de tablas y el cambio de color de las celdas
com.itextpdf.text.pdf.PdfPTable	**tablaNormal**() Método para probar la generación de tablas

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

CreatePDFTest

```
public CreatePDFTest()
```

Method Detail

setUp

```
public void setUp()
           throws BusinessException
```

Método para inicilaizar variables

Throws:

BusinessException

ITextHelloWorld

```
public void ITextHelloWorld(com.itextpdf.text.Document document)
```

Método de Test para generar diferentes tipos de documentos pdf

Parameters:

document -

---

tablaNormal

```
public com.itextpdf.text.pdf.PdfPTable tablaNormal()
                                              throws java.lang.Exception
```

Método para probar la generación de tablas

Returns:

PdfPTable con una tabla simple

Throws:

java.lang.Exception

---

tablaColor

```
public com.itextpdf.text.pdf.PdfPTable tablaColor()
                                              throws java.lang.Exception
```

Método para probar la generación de tablas y el cambio de color de las celdas

Returns:

PdfPTable con una tabla simple con celda con fondo amarillo

Throws:

`java.lang.Exception`

---

tablaAlineacion

```
public boolean tablaAlineacion(com.itextpdf.text.Document document)
                        throws java.lang.Exception
```

Método para probar la generación de tablas alineadas

Returns:

Boolean si se ha creado correctamente una tabla alineada

Throws:

`java.lang.Exception`

7.4.2.12.4 Clase PersistenceTest

```
java.lang.Object
  └ com.miw.junit.PersistenceTest
```

```
public class PersistenceTest
extends java.lang.Object
```

Clase para probar la persistencia

Author:

Pablo

Constructor Summary

`PersistenceTest()`

Method Summary

void	`setUp()` Método setUp para inicializar variables
void	`testIden()` Método de Test para Riesgos Identificados
void	`testInfo()` Método de Test para Planes de Riesgos

void	**testProyecto**() Método de Test para Proyecto
void	**testResp**() Método de Test para Respuesta de Riesgos
void	**testUsuario**() Método de Test para Usuario

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

PersistenceTest

```
public PersistenceTest()
```

Method Detail

setUp

```
public void setUp()
            throws BusinessException
```

Método setUp para inicilizar variables

Throws:

BusinessException

---

testUsuario

```
public void testUsuario()
                  throws java.lang.Exception
```

Método de Test para Usuario

Throws:

java.lang.Exception

---

testProyecto

```
public void testProyecto()
                   throws java.lang.Exception
```

Método de Test para Proyecto

Throws:

java.lang.Exception

---

testInfo

```
public void testInfo()
               throws java.lang.Exception
```

Método de Test para Planes de Riesgos

Throws:

java.lang.Exception

---

testIden

```
public void testIden()
            throws java.lang.Exception
```

Método de Test para Riesgos Identificados

Throws:

java.lang.Exception

---

testResp

```
public void testResp()
            throws java.lang.Exception
```

Método de Test para Respuesta de Riesgos

Throws:

java.lang.Exception

Capítulo 8. Desarrollo de las Pruebas

8.1 Pruebas Unitarias

El proceso de desarrollo de dichas pruebas sobre la aplicación, se realizo para ver el funcionamiento o ver la existencia de posibles problemas o fallos en el funcionamiento del programa. Las diferentes pruebas de asociación, creación de PDF y persistencia se realizaron bajo un servidor local (localhost) por medio del framework de testing JUnit.

8.1.1 Asociación

Prueba 1. Añadir Posesión	

Clases Utilizadas	Prueba de Asociación
	• src
	  • com.miw
	  • exception
	  ▪ BusinessException.java
	  • model
	  ▪ Info.java
	  ▪ Iden.java
	  ▪ Impacto.java
	  ▪ Cambio.java
	  ▪ Probabilidad.java
	• test
	  • com.miw.junit
	  • AllTest.java
	  • AssociationsTest.java

Entrada	1. Se carga el test "*Associations.java*" desde la clase de todos los test "AllTests.java"
	2. Está clase crea y carga con datos las diferentes objetos, para realizar las pruebas
	3. Se ejecuta el test para asociar los datos a los modelos, y se comprueba que
	  a. Contiene el dato correctamente
	  b. El dato que tiene s igual al que se le inserto
	  c. No se encuentran vacios
	  d. Su tamaño es mayor que cero

Salida	Los *assert* devuelven el valor **True**
Resultado	Correcto

Prueba 2. Eliminar Posesión	

Clases Utilizadas	Prueba de Eliminación
	• src
	  • com.miw
	  • exception
	  ▪ BusinessException.java
	  • model
	  ▪ Info.java
	  ▪ Iden.java
	  ▪ Impacto.java
	  ▪ Cambio.java
	  ▪ Probabilidad.java
	• test
	  • com.miw.junit
	  • AllTest.java
	  • AssociationsTest.java

Entrada	1. Se carga el test "*Associations.java*" desde la clase de todos los test "AllTests.java"
	2. Está clase crea y carga con datos las diferentes objetos, para realizar las pruebas
	3. Se ejecuta el test para eliminar los datos a los modelos, y se comprueba que:
	  a. No contiene el dato
	  b. Se encuentra vacio
	  c. Su tamaño es igual que cero

Salida	Los *assert* devuelven el valor **True**
Resultado	Correcto

Prueba 3. Agregar Pertenencia dentro de un Agregado	

Clases Utilizadas	Prueba de Agregación de una Pertenencia
	• src
	  • com.miw
	  • exception
	  ▪ BusinessException.java
	  • model
	  ▪ Info.java
	  ▪ Probabilidad.java
	• test
	  • com.miw.junit
	  • AllTest.java
	  • AssociationsTest.java

Entrada	1. Se carga el test "*Associations.java*" desde la clase de todos los test "AllTests.java" 2. Está clase crea y carga con datos las diferentes objetos, para realizar las pruebas 3. Se ejecuta el test para comprobar que han sido cargados la pertenencia a un dato ya agregado, y se comprueba que: a. Dentro del objeto agregado este contiene la nueva pertenencia

	b. Este nuevo objeto corresponde con el agregado c. No se encuentra vacio, el objeto al que se agrega el nuevo dato d. El tamaño de este es mayor que cero
Salida	Los *assert* devuelven el valor **True**
Resultado	Correcto

Prueba 4. Eliminar Pertenencia dentro de un Agregado	

Clases Utilizadas	Prueba de Eliminación de una Pertenencia a un Dato Agregado
	• src
	  • com.miw
	  • exception
	  ▪ BusinessException.java
	  • model
	  ▪ Info.java
	  ▪ Probabilidad.java
	• test
	  • com.miw.junit
	  • AllTest.java
	  • AssociationsTest.java

Entrada	1. Se carga el test "*Associations.java*" desde la clase de todos los test "AllTests.java" 2. Está clase crea y carga con datos las diferentes objetos, para realizar las pruebas 3. Se ejecuta el test que elimina el objeto que anteriormente pertenecía al dato agregado, se comprueba que: a. Dentro del objeto agregado este no contiene la anterior pertenencia b. Se encuentra vacio, ya que no existe el dato anterior c. El tamaño de este es igual que cero
Salida	Los *assert* devuelven el valor **True**
Resultado	Correcto

8.1.2 Creación PDF

***Prueba 1*. Creación PDF**	

Clases Utilizadas	Prueba de Creación de Documento PDF
	• src
	  • com.miw
	  • exception
	  ▪ BusinessException.java
	• test
	  • com.miw.junit
	  • AllTest.java
	  • CreatePDFTest.java

Entrada	1. Se carga el test "*CreatePDFTest.java*" desde la clase de todos los test "AllTests.java"
	2. Está clase crea y carga diferentes opciones de estructuras para la creación de los documentos PDF
	3. Se ejecuta el test que:
	  a. Añade nuevos párrafos de texto al documento
	  b. Añade párrafos de texto junto a llamadas a funciones que crean formatos de fechas
	  c. Genera la estructura de una tabla dentro del documento
	  d. Genera la estructura de una tabla, con cambios de color en las celdas dentro del documento
	  e. Genera la estructura de una tabla con los elementos alineados dentro del documento
	  f. Se rellena el documento del PDF con todas estas estructuras

Salida	Los *assert* devuelven el valor **True**
Resultado	Correcto

8.1.3 Persistencia

***Prueba 1*. Usuario**	

Clases Utilizadas	Prueba de Inserción y Comprobación de Datos de un Usuario
	• src
	  • com.miw
	  • model
	  ▪ User.java
	  • impl.miw
	  • persistence
	  ▪ UserDAO.java
	• test
	  • com.miw.junit
	  • AllTest.java
	  • PersistenceTest.java

Entrada	1. Se carga el test "*PersistenceTest.java*" desde la clase de todos los test "AllTests.java" 2. Está clase crea un usuario

	3. Se ejecuta el test que prueba que: a. El usuario se ha insertado correctamente y devuelve el mensaje de OK b. El usuario se corresponde con el de la inserción del modelo del usuario c. El idioma se corresponde con el de la inserción del modelo del usuario d. El login se corresponde con el de la inserción del modelo del usuario
Salida	Los *assert* devuelven el valor de los campos, que deben ser iguales al introducido inicialmente
Resultado	Correcto

***Prueba 2*. Proyecto**	
Clases Utilizadas	Prueba de Inserción y Comprobación de Datos de un Proyecto • src • com.miw • model ▪ Project.java • impl.miw • persistence ▪ ProjectDAO.java • test • com.miw.junit • AllTest.java • PersistenceTest.java
Entrada	1. Se carga el test "*PersistenceTest.java*" desde la clase de todos los test "AllTests.java" 2. Está clase crea un proyecto 3. Se ejecuta el test que prueba que: a. El proyecto se ha insertado correctamente y devuelve el mensaje de OK b. Si se busca por medio del usuario correspondiente de este proyecto, devuelve el proyecto c. El proyecto se corresponde con el de la inserción del modelo del proyecto d. El nombre del proyecto se corresponde con el de la inserción del modelo del proyecto e. El Manager se corresponde con el de la inserción del modelo del proyecto f. La fecha se corresponde con el de la inserción del modelo del proyecto
Salida	Los *assert* devuelven el valor de los campos, que deben ser iguales al introducido inicialmente
Resultado	Correcto

***Prueba 3*. Plan de Riesgos**	

Clases Utilizadas	Prueba de Inserción y Comprobación del Plan de Riesgos
	• src
	  • com.miw
	  • model
	  ▪ Info.java
	  ▪ Probabilidad.java
	  ▪ Impacto.java
	  ▪ Cambio.java
	  • impl.miw
	  • persistence
	  ▪ InfoDAO.java
	• test
	  • com.miw.junit
	  • AllTest.java
	  • PersistenceTest.java

Entrada	1. Se carga el test "*PersistenceTest.java*" desde la clase de todos los test "AllTests.java"
	2. Está clase crea un Bean sobre el plan de riesgos
	3. Se ejecuta el test que prueba que:
	  a. El plan de riesgos se ha insertado correctamente y devuelve el mismo plan de riesgos
	  b. Los datos del calendario del plan se corresponde con los de la inserción del modelo del plan
	  c. El título se corresponde con la de la inserción del modelo del plan
	  d. Dentro del objeto del plan este contiene un objeto Cambio
	  d. Dentro del objeto del plan el tamaño del objeto probabilidad es igual a uno
	  e. Dentro del objeto del plan el objeto Probabilidad se corresponde con el insertado inicialmente

Salida	Los *assertEquals* devuelven el valor de los campos, que deben ser iguales al que se le compara Los assertSame devuelven el mismo objeto insertado Los *assertTrue* devuelven **True**
Resultado	Correcto

***Prueba 4*. Identificación de los Riesgos**	

Clases Utilizadas	Prueba de Inserción y Comprobación de la Identificación de Riesgos
	• src
	  • com.miw
	  • model
	  ▪ Iden.java
	  ▪ Cambio.java
	  • impl.miw
	  • persistence
	  ▪ InfoDAO.java
	• test

	• com.miw.junit
	  • AllTest.java
	  • PersistenceTest.java

Entrada	1. Se carga el test "*PersistenceTest.java*" desde la clase de todos los test "AllTests.java"
	2. Está clase crea un Bean sobre la identificación de riesgos
	3. Se ejecuta el test que prueba que:
	  a. El Bean se ha insertado correctamente y devuelve el mismo Bean
	  b. Que la inserción supone que el tamaño de los proyectos debe ser mayor que uno
	  c. Los datos de la descripción se corresponde con los de la inserción del modelo para el Iden
	  d. Las notas se corresponde con los de la inserción del modelo para el Iden
	  e. La respuesta se corresponde con los de la inserción del modelo para el Iden
	  f. El tamaño de los cambios de la identificación de riesgos es mayor que cero
	  g. El contenido de ese cambio corresponde al del Bean

Salida	Los *assertEquals* devuelven el valor de los campos, que deben ser iguales al que se le compara Los assertSame devuelven el mismo objeto insertado Los *assertTrue* devuelven **True**
Resultado	Correcto

***Prueba 5*. Respuesta a los Riesgos**	

Clases Utilizadas	Prueba de Inserción y Comprobación de la Respuesta a los Riesgos
	• src
	  • com.miw
	  • model
	  ▪ Respuesta.java
	  • impl.miw
	  • persistence
	  ▪ InfoDAO.java
	• test
	  • com.miw.junit
	  • AllTest.java
	  • PersistenceTest.java

Entrada	1. Se carga el test "*PersistenceTest.java*" desde la clase de todos los test "AllTests.java"
	2. Está clase crea un Bean de la respuesta a los riesgos
	3. Se ejecuta el test que prueba que:
	  a. El Bean se ha insertado correctamente y devuelve el mismo Bean
	  b. Los datos de la categoría se corresponde con los de la inserción del modelo para la respuesta
	  c. Los comentarios se corresponde con los de la inserción del modelo para la respuesta

	d. El dato de probabilidad corresponde a noventa e. La longitud de los campos de impacto son exactamente cuatro f. El número de indicadores, después de dividirlo por el carácter "@" se corresponde con tres
Salida	Los *assertEquals* devuelven el valor de los campos, que deben ser iguales al que se le compara Los assertSame devuelven el mismo objeto insertado Los *assertTrue* devuelven **True**
Resultado	Correcto

8.2 Pruebas de Integración y del Sistema

Se ejecutan las pruebas funcionales ya diseñadas anteriormente en la parte de Análisis en el apartado 5.5., se anota el resultado que se obtiene, comparándolo con el que se especifica anteriormente.

Caso de Uso 1: Identificarse en el Sistema	
Prueba	**Resultado Esperado**
Identificarse con un usuario existente	El sistema comprueba que el login y la contraseña se encuentran en la BBDD y se accede a la pantalla del rol
	Resultado Obtenido
	El sistema efectivamente posee ese login y esa contraseña y deja acceder al usuario a la aplicación
Prueba	**Resultado Esperado**
Identificarse con un usuario no existente	El sistema no posee el login o la contraseña en la BBDD y se muestra un mensaje notificándolo
	Resultado Obtenido
	El sistema efectivamente muestra un mensaje de error
Prueba	**Resultado Esperado**
Datos no válidos	El sistema comprueba los datos introducidos y si no pasa las validaciones emite un mensaje de error
	Resultado Obtenido
	El sistema efectivamente muestra un mensaje de error

Caso de Uso 1.2: Recuperar Contraseña	
Prueba	**Resultado Esperado**
Envío de correo correcto para recuperar contraseña	El sistema identifica el correo con un usuario y le envía el correo con los datos
	Resultado Obtenido
	El sistema efectivamente envía el correo
Prueba	**Resultado Esperado**
Envío de correo no existente para recuperar contraseña	El sistema no encuentra ese correo, no manda el email y muestra un mensaje notificándolo
	Resultado Obtenido
	El sistema emite un mensaje de que ese usuario no existe con ese correo
Prueba	**Resultado Esperado**
Correo no válido	El sistema comprueba el correo introducido y si no pasa las validaciones emite un mensaje de error
	Resultado Obtenido
	El sistema efectivamente muestra un mensaje de error

Caso de Uso 2: Crear/Almacenar Manager y Proyecto	
Prueba	**Resultado Esperado**
Añadir un nuevo Manager y proyecto	El sistema posee un nuevo usuario de tipo Manager y un proyecto asociado a ese Manager, además el usuario tendrá un correo con sus datos y los del proyecto
	Resultado Obtenido
	El sistema efectivamente posee un usuario y un proyecto más, este asociado al Manager y el usuario los correos con los datos creados anteriormente
Prueba	**Resultado Esperado**
Datos no válidos	El sistema comprueba que todos los datos introducidos cumplen con el sistema de validación y si no emite un mensaje de error en el campo incorrecto
	Resultado Obtenido
	El sistema muestra un mensaje en el campo que no cumple la validación
Prueba	**Resultado Esperado**
Error en el envío	Si se produce un error en el envío del correo con los datos, el sistema deberá informar del error sin dar más detalle
	Resultado Obtenido
	El sistema informa de que el correo no se ha mandado por algún error

Caso de Uso 2.1: Visualizar Proyectos/Managers	
Prueba	**Resultado Esperado**
Visualizar datos	El sistema muestra al usuario Administrador los datos de los proyectos y managers que se encuentran en la aplicación
	Resultado Obtenido
	El sistema efectivamente muestra los datos
Prueba	**Resultado Esperado**
No visualizar ningún dato	El sistema no mostrará ningún dato, sino lo hay
	Resultado Obtenido
	El sistema muestra la ventana, pero sin tablas con información
Prueba	**Resultado Esperado**
Operación eliminar	El sistema debería eliminar/actualizar el dato en la BBDD solicitado por el Administrador y emitir un mensaje
	Resultado Obtenido
	Se eliminan/actualiza y se emite el mensaje
Prueba	**Resultado Esperado**
Operación actualizar	El sistema debería mostrar una nueva ventana donde actualizar los datos
	Resultado Obtenido
	El sistema muestra la ventana

Caso de Uso 3: Crear/Almacenar Miembro	
Prueba	**Resultado Esperado**
Añadir un nuevo miembro	El sistema posee un nuevo usuario de tipo miembro y un proyecto asociado a ese miembro, además el usuario tendrá un correo con sus datos
	Resultado Obtenido
	El sistema efectivamente posee un usuario más y este asociado al proyecto, el usuario recibe el correo
Prueba	**Resultado Esperado**
Datos no válidos	El sistema comprueba que todos los datos introducidos cumplen con el sistema de validación y si no emite un mensaje de error en el campo incorrecto
	Resultado Obtenido
	El sistema muestra un mensaje en el campo que no cumple la validación
Prueba	**Resultado Esperado**
Error en el envío	Si se produce un error en el envío del correo con los datos, el sistema deberá informar del error sin dar más detalle
	Resultado Obtenido
	El sistema informa de que el correo no se ha mandado por algún error

Caso de Uso 4: Obtención y Visualización de Datos para la Gestión de Riesgos	
Prueba	**Resultado Esperado**
Obtener/Visualizar datos	El sistema obtiene y muestra por ventana los datos solicitados del plan de gestión
	Resultado Obtenido
	El sistema efectivamente muestra los datos
Prueba	**Resultado Esperado**
Datos vacios	El sistema no mostrará nada al no obtener datos de la base de datos
	Resultado Obtenido
	El sistema no muestra nada

Caso de Uso 4.1: Crear/Almacenar Gestión de Riesgos	
Prueba	**Resultado Esperado**
Añadir nueva gestión de riesgos	El sistema posee un nuevo registro de gestión de riesgos
	Resultado Obtenido
	El sistema efectivamente posee un nuevo registro
Prueba	**Resultado Esperado**
Actualizar gestión de riesgos	Si el sistema comprueba que ya existe un identificador para esa gestión actualizara el de base de datos con los datos del nuevo registro
	Resultado Obtenido
	El sistema actualiza el plan antiguo con los nuevos datos
Prueba	**Resultado Esperado**
Aparece un error	El sistema permanece sin cambios y muestra un mensaje de

	error
	Resultado Obtenido
	El sistema no almacena ni crea nada y informa por medio de un mensaje de error
Prueba	**Resultado Esperado**
Datos inválidos	El sistema muestra un mensaje de error en el campo incorrecto y no realiza ninguna operación
	Resultado Obtenido
	El sistema muestra un error

Caso de Uso 4.2: Creación de Documentos PDF	
Prueba	**Resultado Esperado**
Crear un nuevo documento PDF	El sistema recoge los datos del informe que se quiere montar, crea la estructura y lo almacena en el disco del usuario mostrándole la ruta del path
	Resultado Obtenido
	El sistema efectivamente crea el PDF correctamente lo almacena en disco y muestra el path
Prueba	**Resultado Esperado**
Error	El sistema informa al usuario de que se ha producido algún error sin dar más información
	Resultado Obtenido
	El sistema muestra mensaje de de error

8.3 Pruebas de Usabilidad y Accesibilidad

8.3.1 Pruebas de Usabilidad

A partir de los cuestionarios que se diseñaron anteriormente y de los procedimientos explicados, mostramos aquí el resultado.

8.3.1.1 Preguntas de carácter general

En negrita estarán las respuestas de los tres diferentes usuarios que se han descrito en el punto 6.6.2. del Diseño que iban a realizar los diferentes cuestionarios.

USUARIO A (PERFIL Técnico)

¿Usa un ordenador frecuentemente?
1. **Todos los días** 2. Varias veces a la semana 3. Ocasionalmente 4. Nunca o casi nunca
¿Qué tipo de actividades realiza con el ordenador?
1. **Es parte de mi trabajo o profesión** 2. Lo uso básicamente para ocio 3. Solo empleo aplicaciones estilo Office 4. Únicamente leo el correo y navego ocasionalmente
¿Ha usado alguna vez software como el de esta prueba?
1. **Sí, he empleado software similar** 2. No, aunque si empleo otros programas que me ayudan a realizar tareas similares 3. No, nunca
¿Qué busca Vd. Principalmente en un programa?
1. Que sea fácil de usar 2. Que sea intuitivo 3. Que sea rápido 4. **Que tenga todas las funciones necesarias**

USUARIO B (Perfil Estudios Obligatorios)

¿Usa un ordenador frecuentemente?
1. **Todos los días** 2. Varias veces a la semana 3. Ocasionalmente 4. Nunca o casi nunca
¿Qué tipo de actividades realiza con el ordenador?
1. **Es parte de mi trabajo o profesión** 2. Lo uso básicamente para ocio 3. Solo empleo aplicaciones estilo Office 4. Únicamente leo el correo y navego ocasionalmente
¿Ha usado alguna vez software como el de esta prueba?

1. Sí, he empleado software similar 2. **No, aunque si empleo otros programas que me ayudan a realizar tareas similares** 3. No, nunca
¿Qué busca Vd. Principalmente en un programa?
1. **Que sea fácil de usar** 2. Que sea intuitivo 3. Que sea rápido 4. Que tenga todas las funciones necesarias

USUARIO C (Perfil sin estudios)

¿Usa un ordenador frecuentemente?
1. **Todos los días** 2. Varias veces a la semana 3. Ocasionalmente 4. Nunca o casi nunca
¿Qué tipo de actividades realiza con el ordenador?
1. Es parte de mi trabajo o profesión 2. **Lo uso básicamente para ocio** 3. Solo empleo aplicaciones estilo Office 4. Únicamente leo el correo y navego ocasionalmente
¿Ha usado alguna vez software como el de esta prueba?
1. Sí, he empleado software similar 2. No, aunque si empleo otros programas que me ayudan a realizar tareas similares 3. **No, nunca**
¿Qué busca Vd. Principalmente en un programa?
1. Que sea fácil de usar 2. **Que sea intuitivo** 3. Que sea rápido 4. Que tenga todas las funciones necesarias

8.3.1.2 Actividades Guiadas

Las actividades generales que realizan todos los usuarios contendrán una nota media de entre todos ellos, luego el caso del Usuario A que hace el papel del Administrador y el Usuario B que hace el papel de Manager, tendrá opciones únicas para ellos.

Las puntuaciones cuanto más bajo sea el número, más sencillo ha sido de realizarlo, como se indico en el apartado del Diseño.

Actividad	Usuarios A,B,C 1…3	Observaciones
Loguearse	1	La pantalla inicial aparece inicialmente con los campos de introducción del login y contraseña
Ayudan los mensajes de estado	1	
Uso de la ayuda en línea	1	

Salir de la aplicación	1	
Recuperar Contraseña	1	Hay veces que sale error en el envío del correo
Usuario A		
Crear Managers/Proyectos	1.5	Se puede crear un Manager sin que tenga este un proyecto?
Visualizar Managers/Proyecto	1	
Modificar Managers/Proyectos	1	
Eliminar Managers/Proyectos	1	Al eliminar el proyecto se elimina el Manager
Usuario B		
Crear Miembros	1	
Crear Plan de Gestión de Riesgos	1.5	No sé que poner en algunos campos
Crear Identificación/Análisis de Riesgos	1.5	No sé que poner en algunos campos
Crear Respuesta a los Riesgos	1.5	No sé que poner en algunos campos
Creación de Informes	1	
Usuario C		
Crear Plan de Gestión de Riesgos	1.7	No entiende que debe poner en algunos campos
Crear Identificación/Análisis de Riesgos	2	No entiende que debe poner en algunos campos
Crear Respuesta a los Riesgos	2	No entiende que debe poner en algunos campos
Creación de Informes	1	

8.3.1.3 Preguntas Cortas sobre la Aplicación y Observaciones

Este cuestionario será respondido individualmente, por cada uno de ellos, marcando con un X, la respuesta que consideren más cercana a su visión sobre la pregunta.

USUARIO A (Perfil Técnico)

Facilidad de Uso	**Siempre**	**Frecuentemente**	**Ocasionalmente**	**Nunca**
¿Sabe donde está dentro de la aplicación?	**X**			
¿Existe ayuda para las funciones en caso de que tenga dudas?		**X**		
¿Le resulta sencillo el uso de la aplicación?		**X**		
Funcionalidad	**Siempre**	**Frecuentemente**	**Ocasionalmente**	**Nunca**
¿Funciona cada tarea como Vd. Espera?		**X**		
¿El tiempo de respuesta de la aplicación es muy grande?			**X**	
Calidad del Interfaz				

Aspectos gráficos	Muy Adecuado	Adecuado	Poco Adecuado	Nada Adecuado
El tipo y tamaño de letra es		**X**		
Los iconos e imágenes usados son	**X**			
Los colores empleados son		**X**		
Diseño de la Interfaz		**Si**	**No**	**A veces**
¿Le resulta fácil de usar?		**X**		
¿El diseño de las pantallas es claro y atractivo?		**X**		
¿Cree que el programa está bien estructurado?		**X**		
Observaciones				

USUARIO B (Perfil Estudios Obligatorios)

Facilidad de Uso	Siempre	Frecuentemente	Ocasionalmente	Nunca
¿Sabe donde está dentro de la aplicación?	**X**			
¿Existe ayuda para las funciones en caso de que tenga dudas?			**X**	
¿Le resulta sencillo el uso de la aplicación?		**X**		
Funcionalidad	**Siempre**	**Frecuentemente**	**Ocasionalmente**	**Nunca**
¿Funciona cada tarea como Vd. Espera?		**X**		
¿El tiempo de respuesta de la aplicación es muy grande?				**X**
Calidad del Interfaz				
Aspectos gráficos	**Muy Adecuado**	**Adecuado**	**Poco Adecuado**	**Nada Adecuado**
El tipo y tamaño de letra es		**X**		
Los iconos e imágenes usados son		**X**		
Los colores empleados son		**X**		
Diseño de la Interfaz		**Si**	**No**	**A veces**
¿Le resulta fácil de usar?				**X**
¿El diseño de las pantallas es claro y atractivo?				**X**
¿Cree que el programa está bien estructurado?		**X**		
Observaciones				
Las pantallas no tienen muchos colores, lo que hace que la aplicación sea muy sería.				

USUARIO C

Facilidad de Uso	Siempre	Frecuentemente	Ocasionalmente	Nunca
¿Sabe donde está dentro de la aplicación?		**X**		

¿Existe ayuda para las funciones en caso de que tenga dudas?			**X**	
¿Le resulta sencillo el uso de la aplicación?			**X**	
Funcionalidad	**Siempre**	**Frecuentemente**	**Ocasionalmente**	**Nunca**
¿Funciona cada tarea como Vd. Espera?	**X**			
¿El tiempo de respuesta de la aplicación es muy grande?				**X**
Calidad del Interfaz				
Aspectos gráficos	**Muy Adecuado**	**Adecuado**	**Poco Adecuado**	**Nada Adecuado**
El tipo y tamaño de letra es	**X**			
Los iconos e imágenes usados son	**X**			
Los colores empleados son	**X**			
Diseño de la Interfaz		**Si**	**No**	**A veces**
¿Le resulta fácil de usar?			**X**	
¿El diseño de las pantallas es claro y atractivo?		**X**		
¿Cree que el programa está bien estructurado?		**X**		
Observaciones				
La ayuda no me aclara que es lo que tengo que poner en ciertos campos que desconozco que es lo que debería ser, no tengo conocimientos sobre la gestión de riesgos y por eso no me resulta sencillo realizar las actividades, pero los pasos que debo seguir están claramente marcados.				

8.3.1.4 Cuestionario para el Responsable de las Pruebas

Aspecto Observado	**Notas**
El usuario comienza a trabajar de forma rápida por las tareas	Si, no tiene perdida alguna, ya que las opciones están bien marcadas, y tiene perfectamente definidos los pasos que debe seguir en cada momento así como los siguientes cuando termine esa tarea.
Sabe localizar rápidamente las diferentes opciones que tiene a su disposición	Todos los usuario saben manejarse en un periodo de tiempo corto, en la aplicación, ayuda bastante a que no es muy extensa y no tiene largos recorridos para llegar a los objetivos, ya que con un par de clics, se puede llegar a finalizar cada paso.
Si cometen un error saben porque se produce	La mayoría de errores son informados al usuario sin dar excesiva información si esos errores forman parte de algún fallo en la programación de la aplicación, si se trata sobre alguna validación por la introducción de un campo incorrecto, saben que lo han escrito mal pero no se les informa de cómo deberían introducirlo para que pasará la validación
Tiempo en realizar cada tarea	El que más tiempo tarda es el Usuario C, ya que desconoce muchas veces que significa lo que está escrito

	y tampoco sabe como que debe rellenar en algunos de los campos de la creación de los riesgos, cosa habitual si se desconocen algunos tecnicismos sobre la gestión de riesgos.
Errores leves cometidos	Alguna faltas de ortografía, algunas internacionalizaciones mal definidas
Errores graves cometidos	Se producía un Null pointer al desaparecer el usuario de la sesión, se ha arreglado haciendo que si pasa se envíe al usuario a la pantalla de login
El usuario encuentra alguna dificultad en alguna actividad	El usuario B y C, no entienden a veces que responder en los formularios de creación de los riesgos.

8.3.2 Pruebas de Accesibilidad

8.3.2.1 Revisión

En la revisión preliminar utilizaremos lo siguiente:

- Navegadores Mozilla Firefox 34.0.5 y Google Chrome 39.0.2171.95 m.
- Escala de grises GrayBit v2.0.1.

Junto a Mozilla Firefox se utilizarán los Plugins:

- FireBug 2.0.7
- WebDeveloper 1.2.5

Y para el Google Chrome:

- Developer Tools

PASO 1. SELECCIÓN DE UN GRUPO DE PÁGINAS REPRESENTATIVO DE LA APLICACIÓN

El nivel deseado de conformidad *WCAG 1.0*, será aplicable al mínimo necesario que para los casos del TFM es un AA.

Web Content Accessibility Guidelines 1.0 (WCAG 1.0) es una recomendación del 5 de mayo de 1999 del W3C que explica cómo hacer el contenido web accesible a las personas con discapacidad. Hasta el momento, WCAG 1.0 es reconocido como el estándar *de facto* a nivel internacional en cuanto a accesibilidad web.

Para ello podemos utilizar la dirección. http://www.w3.org/TR/WCAG10/full-checklist.html donde se encuentra un listado de check para realizar a nuestra página web del proyecto, más adelante se detalla el cuestionario, con los resultados en una plantilla con formato Word.

Para pasar las pruebas de accesibilidad seleccionaremos una muestra de páginas de la aplicación representativa para así no tener que someter a toda la aplicación al proceso de revisión.

La muestra representativa está formada por:

- La página de entrada al sitio

Figura 8.1. Pantalla Inicial

- Página del Plan de Gestión de Riesgos
    - Formulario para la **Definiciones de Probabilidad**, que contiene resultados generados dinámicamente

Figura 8.2. Pantalla con Imágenes Definiciones de Probabilidad

- Gráficos y scripts de la **Matriz de Probabilidad e Impacto**

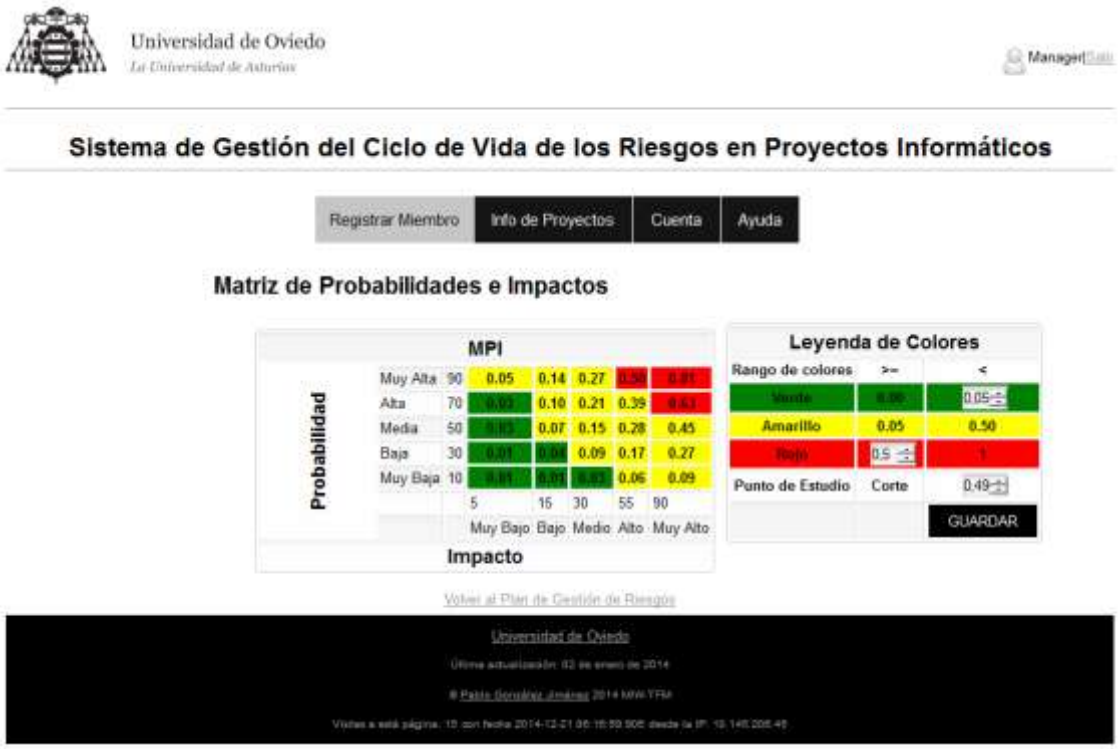


Figura 8.3. Pantalla con Imágenes Matriz de Probabilidad e Impacto

PASO 2. EXAMINAR LAS PÁGINAS USANDO UN NAVEGADOR GRÁFICO

Se desactivan las imágenes y se comprueba cómo queda el aspecto de la aplicación.

Figura 8.4. Pantalla Inicial sin Imágenes

Figura 8.5. Pantalla sin Imágenes Definiciones de Probabilidad

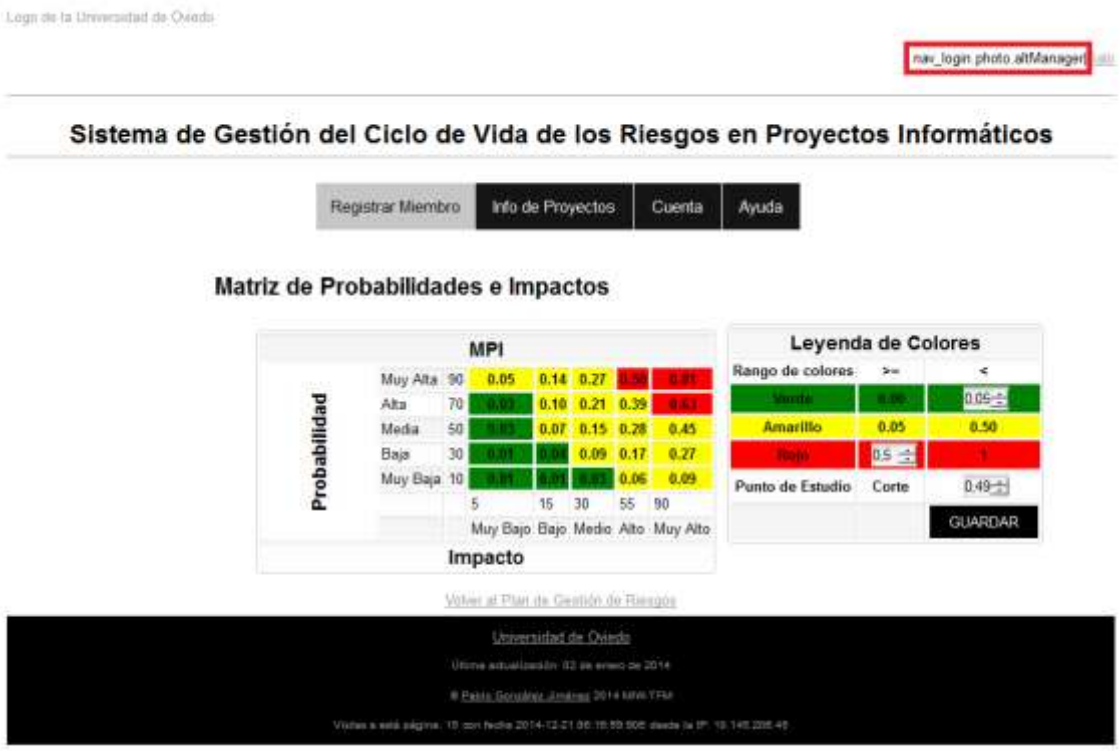


Figura 8.6. Pantalla sin Imágenes Matriz de Probabilidad e Impacto

- Se cambia el tamaño del texto y se comprueba que sigue siendo usable

Figura 8.7. Aumento de Tamaño Pantalla Inicial

Figura 8.8. Aumento Tamaño Definiciones de Probabilidad

Figura 8.9. Aumento Tamaño Matriz de Probabilidad e Impacto

- Se cambia solamente el tamaño de letra de la página para ver cómo se comporta. Cuando se eleva el tamaño de letra por encima de 17 algunos de los elementos comienzan a cambiar de posición, aunque sigue siendo razonablemente legible la página y sus contenidos. En el tamaño máximo de 72 no entra la tabla completa en la pantalla.
- Se prueba a visualizar la página sin sus hojas de estilo *CSS* para asegurarse de que aún es legible y usable.

Figura 8.10. Sin CSS Pantalla Inicial

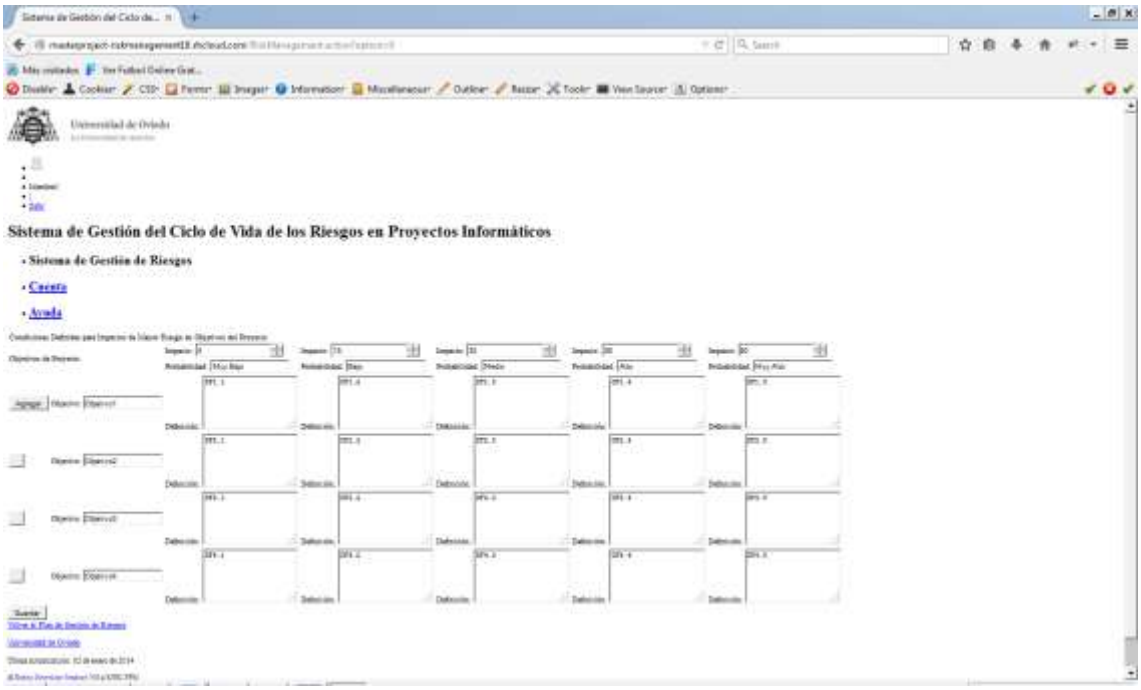


Figura 8.11. Sin CSS Definiciones de Probabilidad

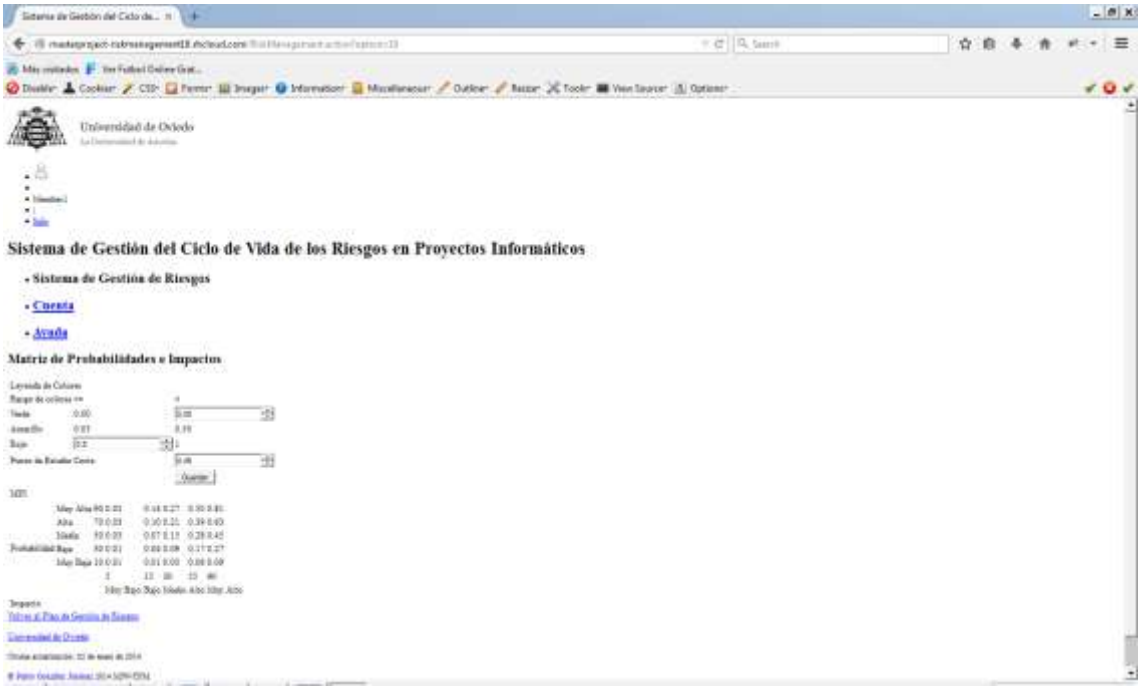


Figura 8.12. Sin CSS Matriz de Probabilidad e Impacto

- Se prueba a visualizar la página usando una escala de grises. Para esto se usará la herramienta web *GrayBit*[19] anteriormente citada.

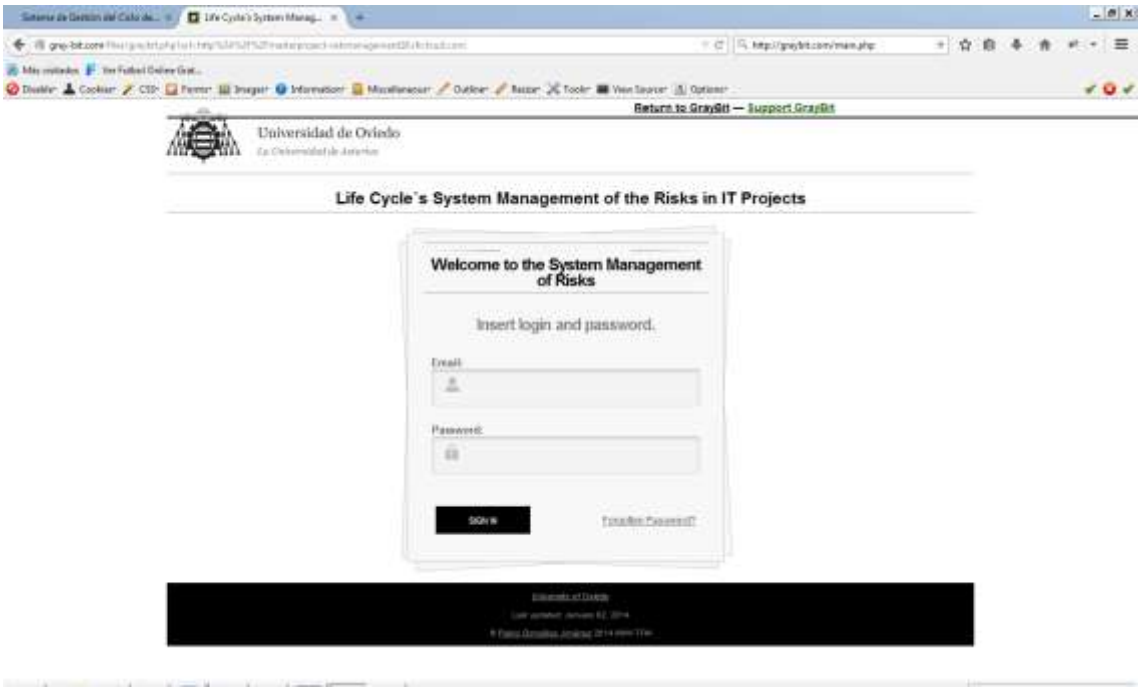


Figura 8.13. Escala de Grises Pantalla Inicial

El resto de páginas no se puede probar.

- Se usa el teclado para navegar a través de los enlaces y controles de formularios, usando *"Tab"* para desplazarse.

Algunos de los campos por crearse dinámicamente o por olvido no carece del atributo "tab" y este pierde la continuidad al pulsar el tabulador por los campos.

PASO 4. UTILIZAR HERRAMIENTAS AUTOMÁTICAS DE EVALUACIÓN DE ACCESIBILIDAD

Este último paso consiste en pasar a la muestra de páginas seleccionadas herramientas de evaluación automática de la accesibilidad como, usaremos dos como son:

- *TAW* (http://www.tawdis.net/taw3/cms/es)
- *HERA* (http://www.sidar.org/hera/index.php.en)

Para las CSS usaremos el validador de W3C:

- W3C (http://jigsaw.w3.org/css-validator)

[19] http://gray-bit.com/main.php

TAW

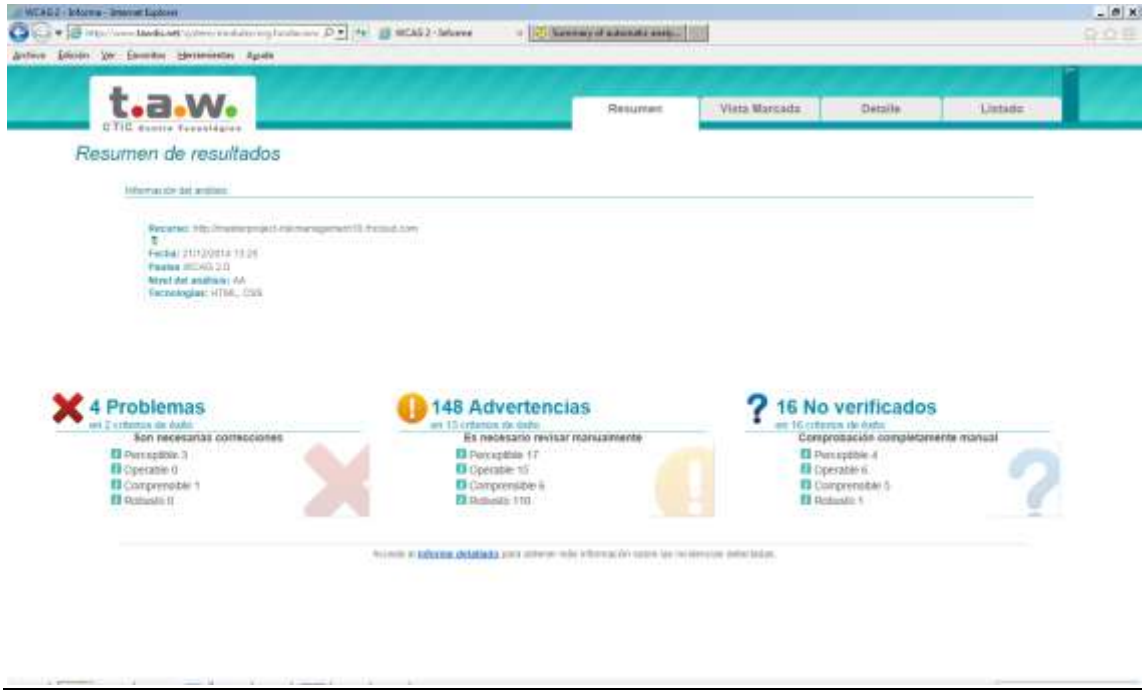


Figura 8.14. Errores TAW

HERA

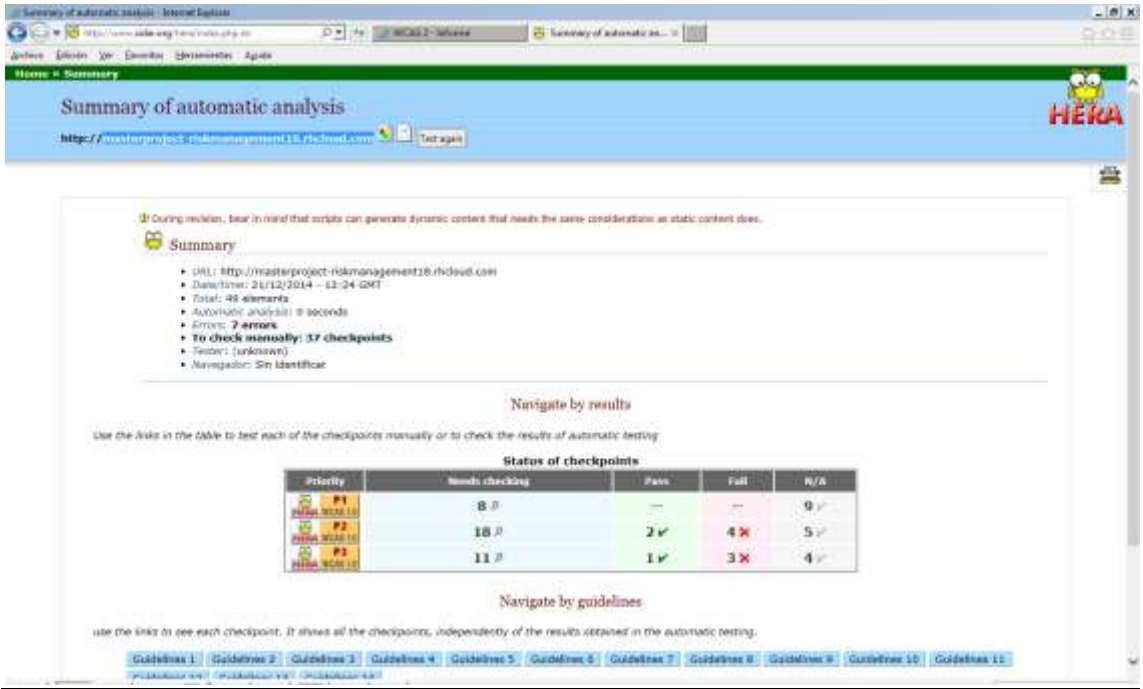


Figura 8.15. Errores HERA

VALIDADOR W3C PARA CSS3

form.css

Figura 8.16. Errores Validador CSS3 para form.css

login.css

Figura 8.17. Errores Validador CSS3 para login.css

style.css

Figura 8.18. Errores Validador CSS3 para style.css

table.css

Figura 8.19. Correcto Validador CSS3 para table.css

PASO 5. RESUMEN DE RESULTADOS

Problemas Encontrados y Soluciones:

Examine Manual:

1. El texto alternativo en algunas imágenes no existe, porque se olvido de incluirlo en el archivo de internacionalización, como fue para el caso de la fotografía para el login.
2. Algunos campos no tienen los "tab" asignados, la mayoría de ellos, porque al crear dinámicamente los datos, no se han puesto al crearlos, se corrige por medio de javascript, creando el atributo "tab", para siga, con la última numeración del último elemento. Además se le asigna a los elementos que no la tenían.

Examine Semiautomático con TAW y HERA:

1. La sección del logo, se repite un mismo identificador para un dos elementos diferentes, se corrige cambiando el valor de uno de ellos.
2. Se está utilizando dos encabezados del mismo nivel seguidos sin contenido entre ellos, se elimina uno de ellos.
3. Se están utilizando etiquetas de presentación cuando se debería utilizar el CSS para ello, se elimina dichas etiquetas y se añade una opción al CSS que las fuentes sean pequeñas con la acción de "*smaller*" en la fuente.
4. Se necesita tener un idioma predeterminado de la página, como se tenía la internacionalización, no se puso, pero se cambia a idioma predeterminado el castellano, realizando los cambios en todos los HTML que se tenga.
5. Se utilizan atributos obsoletos en HTML 4.01., son elementos que crea automáticamente el sistema al transformar por ejemplo los formularios, se soluciona introduciendo la etiqueta ***theme="simple".***
6. Utilización de unidades absolutas en CSS, se tiene la unidad "*rem*" en el tamaño de las fuentes se elimina.

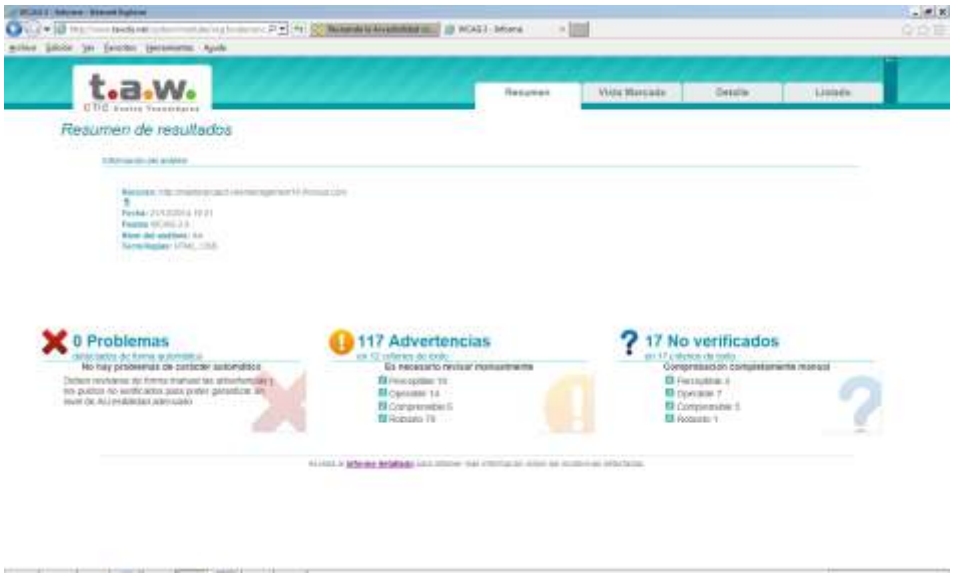


Figura 8.20. Correcto TAW

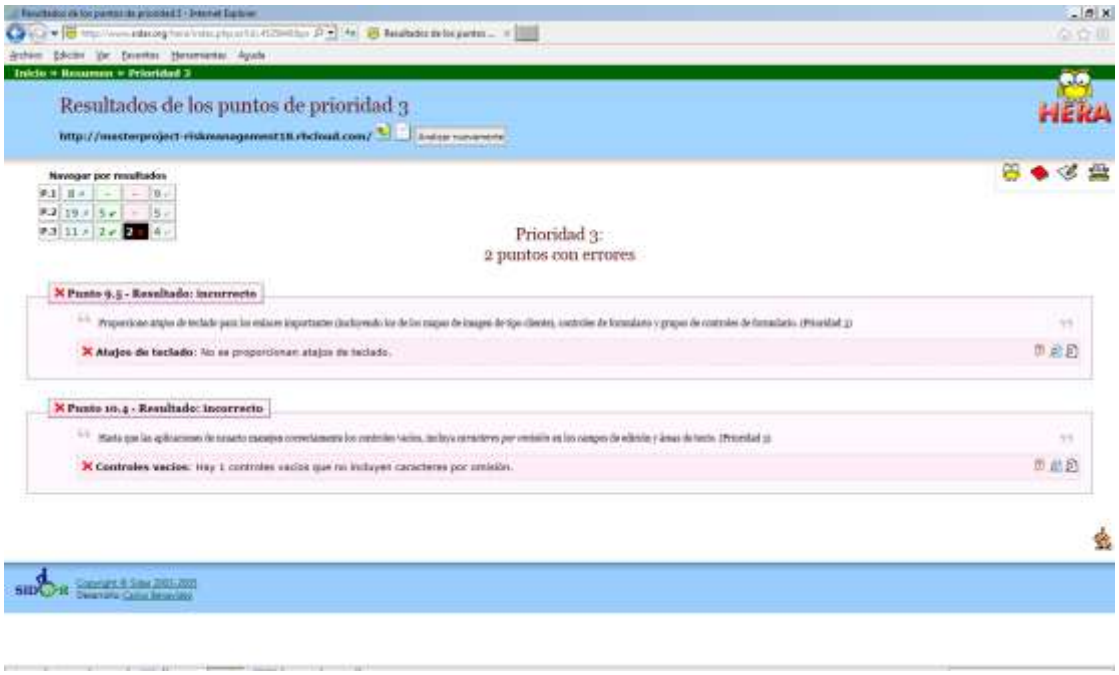


Figura 8.21. Correcto para P.2 HERA

Para futuras ampliaciones se podría trabajar para poder llegar al nivel AAA y al punto de Prioridad 3.

Validación de W3C sobre CSS3

form.css

1. El error que se producía era por aplicar un filtro de Microsoft para IE8, que no son aceptados por el validador. Dicha CSS es correcta si se elimina.

Figura 8.22. Correcto Validador CSS3 para form.css

login.css

1. Mismo error que en la otra CSS, por la cuestión de los filtros, y de que no acepta la posición del atributo en el *background*, se eliminan y se comprueba que se ve igual.

Figura 8.23. Correcto Validador CSS3 para login.css

style.css

1. Uno de los errores era que es que uno de los atributos estaba mal escrito, refiriéndose a la lineación, solo estaba puesto "*align*", cuando debería ser "*text-align*", corregido.
2. Error en los filtro para IE 8, se elimina

Figura 8.24. Correcto Validador CSS3 para style.css

8.3.2.2 Checklist del WCAG 1.0

Puntos de verificación Prioridad 1:

En general (Prioridad 1)	**Sí**	**No**	**N/A**
1.1 Proporcione un texto equivalente para todo elemento no textual (Por ejemplo, a través de "alt", "longdesc" o en el contenido del elemento). *Esto incluye:* imágenes, representaciones gráficas del texto, mapas de imagen, animaciones (Por ejemplo, *GIFs* animados), "applets" y objetos programados, "ascii art", marcos, scripts, imágenes usadas como viñetas en las listas, espaciadores, botones gráficos, sonidos (ejecutados con o sin interacción del usuario), archivos exclusivamente auditivos, banda sonora del vídeo y vídeos.	**X**		
2.1 Asegúrese de que toda la información transmitida a través de los colores también esté disponible sin color, por ejemplo mediante el contexto o por marcadores.	**X**		
4.1 Identifique claramente los cambios en el idioma del texto del documento y en cualquier texto equivalente (por ejemplo, leyendas).			**X**
6.1 Organice el documento de forma que pueda ser leído sin hoja de estilo. Por ejemplo, cuando un documento HTML es interpretado sin asociarlo a una hoja de estilo, tiene que ser posible leerlo.	**X**		
6.2 Asegúrese de que los equivalentes de un contenido dinámico son actualizados cuando cambia el contenido dinámico.	**X**		
7.1 Hasta que las aplicaciones de usuario permitan controlarlo, evite provocar destellos en la pantalla.			**X**
14.1 Utilice el lenguaje apropiado más claro y simple para el contenido de un sitio.	**X**		
Y si utiliza imágenes y mapas de imagen (Prioridad 1)	**Sí**	**No**	**N/A**
1.2 Proporcione vínculos redundantes en formato texto para cada zona activa de un mapa de imagen del servidor.			**X**
9.1 Proporcione mapas de imagen controlados por el cliente en lugar de por el servidor, excepto donde las zonas sensibles no puedan ser definidas con una forma geométrica.			**X**
Y si utiliza tablas (Prioridad 1)	**Sí**	**No**	**N/A**
5.1 En las tablas de datos, identifique los encabezamientos de fila y columna.	**X**		
5.2 Para las tablas de datos que tienen dos o más niveles lógicos de encabezamientos de fila o columna, utilice marcadores para asociar las celdas de encabezamiento y las celdas de datos.		**X**	
Y si utiliza marcos ("frames") (Prioridad 1)	**Sí**	**No**	**N/A**
12.1 Titule cada marco para facilitar su identificación y navegación.			**X**
Y si utiliza "applets" y "scripts" (Prioridad 1)	**Sí**	**No**	**N/A**
6.3 Asegure que las páginas sigan siendo utilizables cuando se desconecten o no se soporten los scripts, *applets* u otros objetos programados. Si esto no es posible, proporcione información equivalente en una página alternativa accesible.			**X**
Y si utiliza multimedia (Prioridad 1)	**Sí**	**No**	**N/A**
1.3 Hasta que las aplicaciones de usuario puedan leer en voz alta automáticamente el texto equivalente de la banda visual, proporcione una descripción auditiva de la información importante de la banda			**X**

visual de una presentación multimedia.			
1.4 Para toda presentación multimedia dependiente del tiempo (por ejemplo, una película o animación) sincronice alternativas equivalentes (por ejemplo, subtítulos o descripciones de la banda visual) con la presentación.			**X**
Y si todo lo demás falla (Prioridad 1)	**Sí**	**No**	**N/A**
11.4 Si, después de los mayores esfuerzos, no puede crear una página accesible, proporcione un vínculo a una página alternativa que use tecnologías *W3C*, sea accesible, tenga información (o funcionalidad) equivalente y sea actualizada tan a menudo como la página (original) inaccesible.			**X**

Puntos de verificación Prioridad 2:

En general (Prioridad 2)	**Sí**	**No**	**N/A**
2.2 Asegúrese de que las combinaciones de los colores de fondo y primer plano tengan el suficiente contraste para que sean percibidas por personas con deficiencias de percepción de color o en pantallas en blanco y negro [Prioridad 2 para las imágenes. Prioridad 3 para los textos].	**X**		
3.1 Cuando exista un marcador apropiado, use marcadores en vez de imágenes para transmitir la información.	**X**		
3.2 Cree documentos que estén validados por las gramáticas formales publicadas.		**X**	
3.3 Utilice hojas de estilo para controlar la maquetación y la presentación.	**X**		
3.4 Utilice unidades relativas en lugar de absolutas al especificar los valores en los atributos de los marcadores de lenguaje y en los valores de las propiedades de las hojas de estilo.		**X**	
3.5 Utilice elementos de encabezado para transmitir la estructura lógica y utilícelos de acuerdo con la especificación.	**X**		
3.6 Marque correctamente las listas y los ítems de las listas.	**X**		
3.7 Marque las citas. No utilice el marcador de citas para efectos de formato tales como sangrías.	**X**		
6.5 Asegúrese de que los contenidos dinámicos son accesibles o proporcione una página o presentación alternativa.		**X**	
7.2 Hasta que las aplicaciones de usuario permitan controlarlo, evite el parpadeo del contenido (por ejemplo, cambio de presentación en periodos regulares, así como el encendido y apagado).			**X**
7.4 Hasta que las aplicaciones de usuario proporcionen la posibilidad de detener las actualizaciones, no cree páginas que se actualicen automáticamente de forma periódica.			**X**
7.5 Hasta que las aplicaciones de usuario proporcionen la posibilidad de detener el redireccionamiento automático, no utilice marcadores para redirigir las páginas automáticamente. En su lugar, configure el servidor para que ejecute esta posibilidad.			**X**
10.1 Hasta que las aplicaciones de usuario permitan desconectar la apertura de nuevas ventanas, no provoque apariciones repentinas de nuevas ventanas y no cambie la ventana actual sin informar al usuario.			**X**
11.1 Utilice tecnologías *W3C* cuando estén disponibles y sean apropiadas para la tarea y use las últimas versiones que sean	**X**		

soportadas.			
11.2 Evite características desaconsejadas por las tecnologías *W3C*.	**X**		
12.3 Divida los bloques largos de información en grupos más manejables cuando sea natural y apropiado.			**X**
13.1 Identifique claramente el objetivo de cada vínculo.	**X**		
13.2 Proporcione metadatos para añadir información semántica a las páginas y sitios.	**X**		
13.3 Proporcione información sobre la maquetación general de un sitio (por ejemplo, mapa del sitio o tabla de contenidos).	**X**		
13.4 Utilice los mecanismos de navegación de forma coherente.	**X**		
Y si utiliza tablas (Prioridad 2)	**Sí**	**No**	**N/A**
5.3 No utilice tablas para maquetar, a menos que la tabla tenga sentido cuando se alinee. Por otro lado, si la tabla no tiene sentido, proporcione una alternativa equivalente (la cual debe ser una versión alineada).	**X**		
5.4 Si se utiliza una tabla para maquetar, no utilice marcadores estructurales para realizar un efecto visual de formato.	**X**		
Y si utiliza marcos ("frames") (Prioridad 2)	**Sí**	**No**	**N/A**
12.2 Describa el propósito de los marcos y cómo éstos se relacionan entre sí, si no resulta obvio solamente con el título del marco.			**X**
Y si utiliza formularios (Prioridad 2)	**Sí**	**No**	**N/A**
10.2 Hasta que las aplicaciones de usuario soporten explícitamente la asociación entre control de formulario y etiqueta, para todos los controles de formularios con etiquetas asociadas implícitamente, asegúrese de que la etiqueta está colocada adecuadamente.	**X**		
12.4 Asocie explícitamente las etiquetas con sus controles.	**X**		
Y si utiliza "applets" y "scripts" (Prioridad 2)	**Sí**	**No**	**N/A**
6.4 Para los *scripts* y *applets*, asegúrese de que los manejadores de eventos sean independientes del dispositivo de entrada.			**X**
7.3 Hasta que las aplicaciones de usuario permitan congelar el movimiento de los contenidos, evite los movimientos en las páginas.			**X**
8.1 Haga los elementos de programación, tales como *scripts* y *applets*, directamente accesibles o compatibles con las ayudas técnicas [Prioridad 1 si la funcionalidad es importante y no se presenta en otro lugar; de otra manera, Prioridad 2].	**X**		
9.2 Asegúrese de que cualquier elemento que tiene su propia interfaz pueda manejarse de forma independiente del dispositivo.			**X**
9.3 Para los "scripts", especifique manejadores de evento lógicos mejor que manejadores de eventos dependientes de dispositivos.			**X**

Puntos de verificación Prioridad 3:

En general (Prioridad 3)	**Sí**	**No**	**N/A**
4.2 Especifique la expansión de cada abreviatura o acrónimo cuando aparezcan por primera vez en el documento.			**X**
4.3 Identifique el idioma principal de un documento.	**X**		
9.4 Cree un orden lógico para navegar con el tabulador a través de vínculos, controles de formulario y objetos.	**X**		
9.5 Proporcione atajos de teclado para los vínculos más importantes (incluidos los de los mapas de imagen de cliente), los controles de		**X**	

formulario y los grupos de controles de formulario.			
10.5 Hasta que las aplicaciones de usuario (incluidas las ayudas técnicas) interpreten claramente los vínculos contiguos, incluya caracteres imprimibles (rodeados de espacios), que no sirvan como vínculo, entre los vínculos contiguos.		**X**	
11.3 Proporcione la información de modo que los usuarios puedan recibir los documentos según sus preferencias (por ejemplo, idioma, tipo de contenido, etc.).	**X**		
13.5 Proporcione barras de navegación para destacar y dar acceso al mecanismo de navegación.	**X**		
13.6 Agrupe los vínculos relacionados, identifique el grupo (para las aplicaciones de usuario) y, hasta que las aplicaciones de usuario lo hagan, proporcione una manera de evitar el grupo.	**X**		
13.7 Si proporciona funciones de búsqueda, permita diferentes tipos de búsquedas para diversos niveles de habilidad y preferencias.			**X**
13.8 Localice la información destacada al principio de los encabezamientos, párrafos, listas, etc.	**X**		
13.9 Proporcione información sobre las colecciones de documentos (por ejemplo, los documentos que comprendan múltiples páginas).			**X**
13.10 Proporcione un medio para saltar sobre un *ASCII art* de varias líneas.		**X**	
14.2 Complemente el texto con presentaciones gráficas o auditivas cuando ello facilite la comprensión de la página.		**X**	
14.3 Cree un estilo de presentación que sea coherente para todas las páginas.	**X**		
Y si utiliza imágenes o mapas de imagen (Prioridad 3)	**Sí**	**No**	**N/A**
1.5 Hasta que las aplicaciones de usuario interpreten el texto equivalente para los vínculos de los mapas de imagen de cliente, proporcione vínculos de texto redundantes para cada zona activa del mapa de imagen de cliente.	**X**		
Y si utiliza tablas (Prioridad 3)	**Sí**	**No**	**N/A**
5.5 Proporcione resúmenes de las tablas.		**X**	
5.6 Proporcione abreviaturas para las etiquetas de encabezamiento.			**X**
10.3 Hasta que las aplicaciones de usuario (incluidas las ayudas técnicas) interpreten correctamente los textos contiguos, proporcione un texto lineal alternativo (en la página actual o en alguna otra) para *todas* las tablas que maquetan texto en paralelo, en columnas de palabras.		**X**	
Y si utiliza formularios (Prioridad 3)	**Sí**	**No**	**N/A**
10.4 Hasta que las aplicaciones de usuario manejen correctamente los controles vacíos, incluya caracteres por defecto en los cuadros de edición y áreas de texto.		**X**	

8.4 Pruebas de Rendimiento

Las pruebas que se han realizado son las explicadas en el apartado del Diseño 6.6.3, con la aplicación de apache JMeter, con los siguientes resultados.

8.4.1 Peticiones HTTP

Para conseguir los resultados hay que hacer lo siguiente:

1. **Grabar** los **requests** que son enviados al servidor no pueden ser grabados aquellos request que sean HTTPS, en nuestro caso no nos afecta.

2. **Cargar** los **requests**, si se ha cargado con éxito, los métodos Web de abajo se deberían rellenar. De lo contrario, un mensaje de error o excepción aparecerá.

3. Seleccionar **Plan de Prueba** en el marco izquierdo, hacer clic derecho, en **Añadir**, se escogen los **Hilos (Usuarios)** y dentro de este **Grupo de Hilos**.

Este componente define el grupo de usuarios que ejecutará la prueba en el servidor. Este componente se configura más tarde cuando se creen las peticiones HTTP.

4. Seleccionar el grupo de hilos creado anteriormente, en el marco izquierdo, hacer clic derecho en **Añadir**, **Elementos de Configuración**, **Valores por defecto para Petición HTTP.**

Este elemento permite configurar los valores predeterminados para los controladores de la petición y siguientes. Esto simplifica la configuración a la hora de enviar varias solicitudes en una prueba JMeter. Se rellenan los siguientes campos:

- Nombre del servidor o la dirección IP:
  masterproject-riskmanagement18.rhcloud.com
- Número de puerto:
  Vacio
- Protocolo:
  HTTP

5. Por último se agregan los Receptores del tipo, Ver Árbol de Resultados, Gráfico de Resultados y Reporte resumen.

8.4.1.1 Acceso al Login

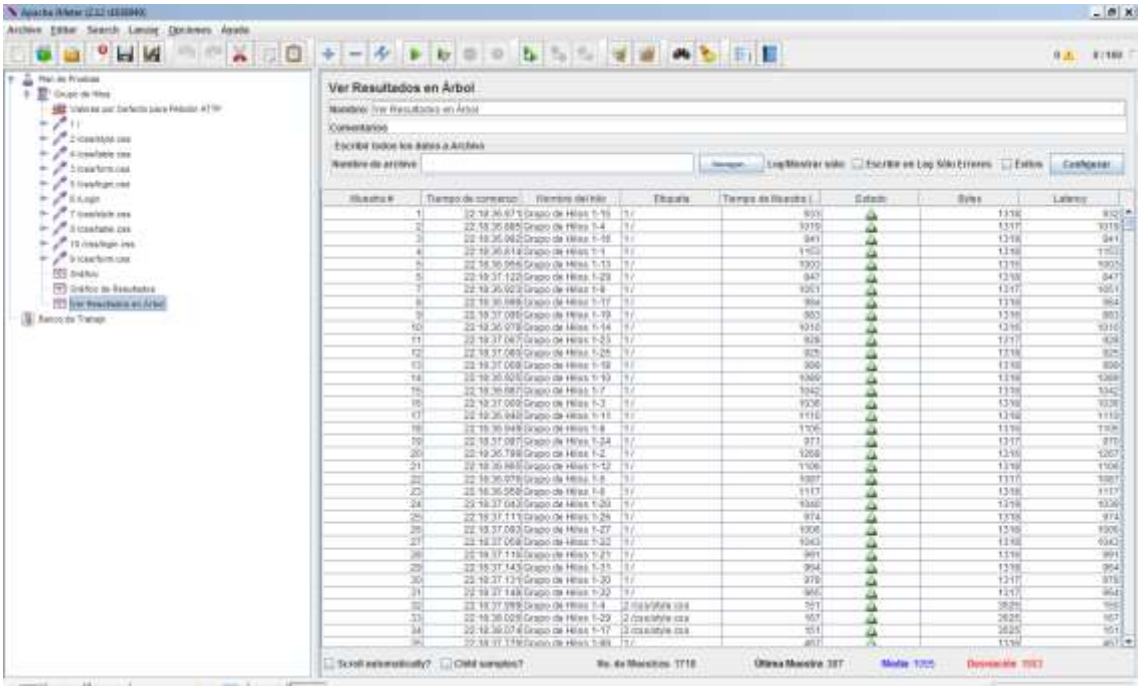


Figura 8.25. Resultados de las Operaciones HTTP para Login

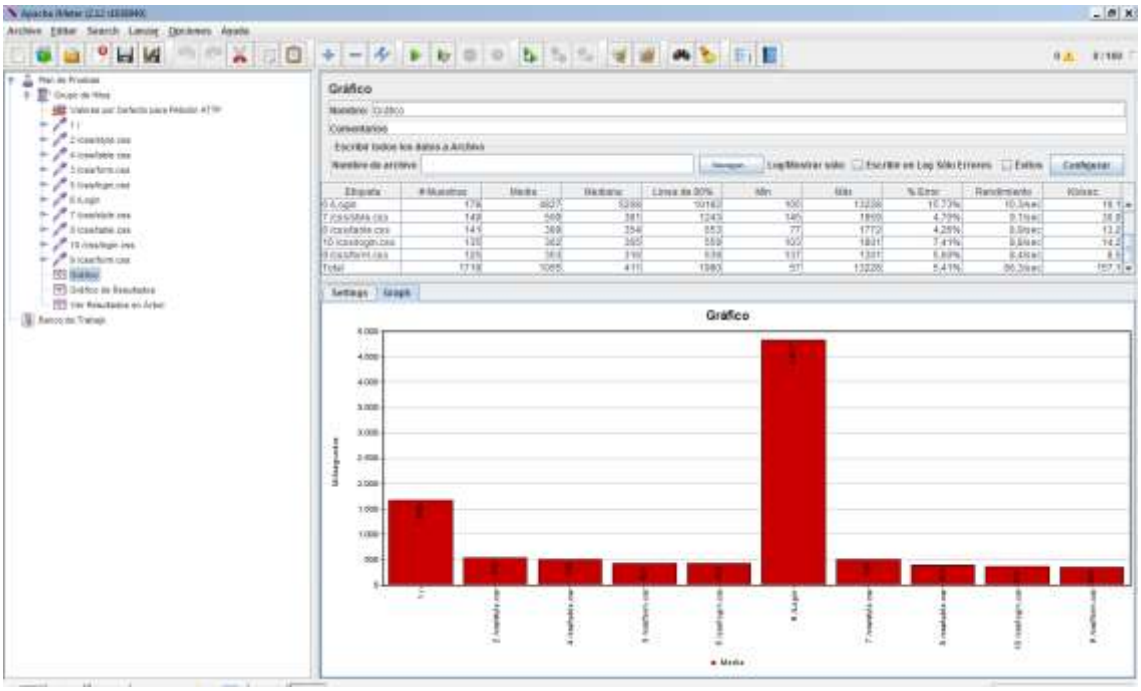


Figura 8.26. Gráfico con los Tiempos de cada Proceso en el Login

Los resultados dicen que con 100 usuarios concurrentes (threads) con periodos de subida de 1 segundo, que trabajan en un bucle infinito pueden invocar unas 86.3 peticiones por segundo. Y el tiempo medio para la petición del Servicio Login es de entre 353-4827 ms que es el tiempo máximo que tarda al cargar la petición Login en sí.

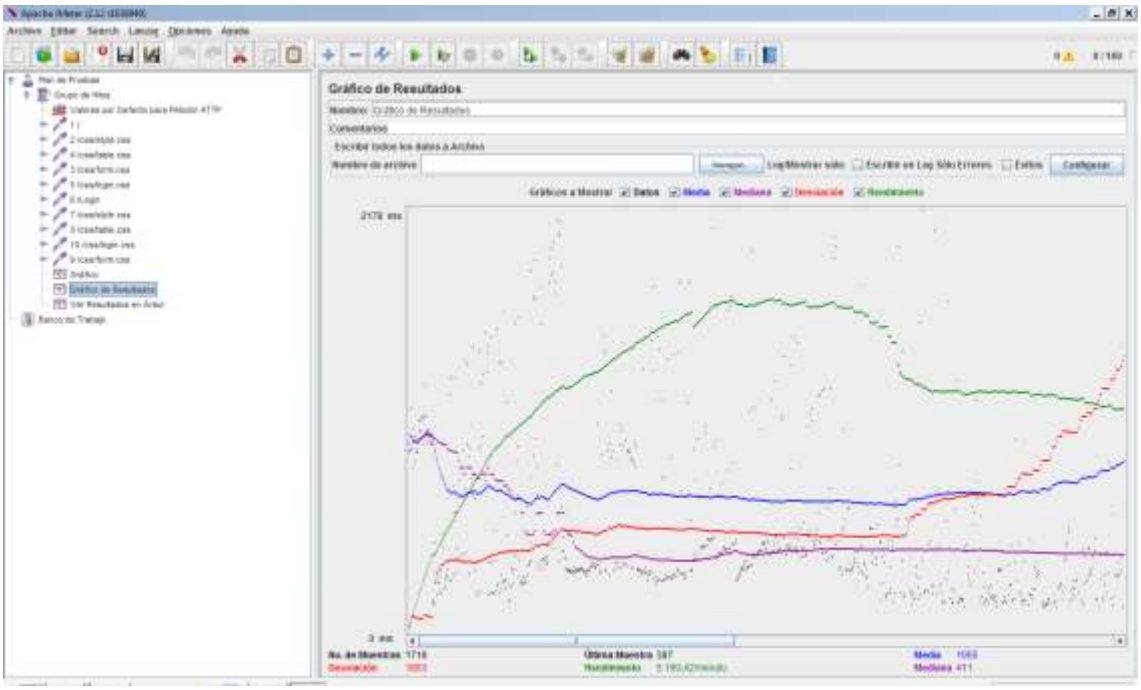


Figura 8.27. Gráficos con los Resultados para el Login

8.4.1.2 Rol Miembro Viajando por la Aplicación

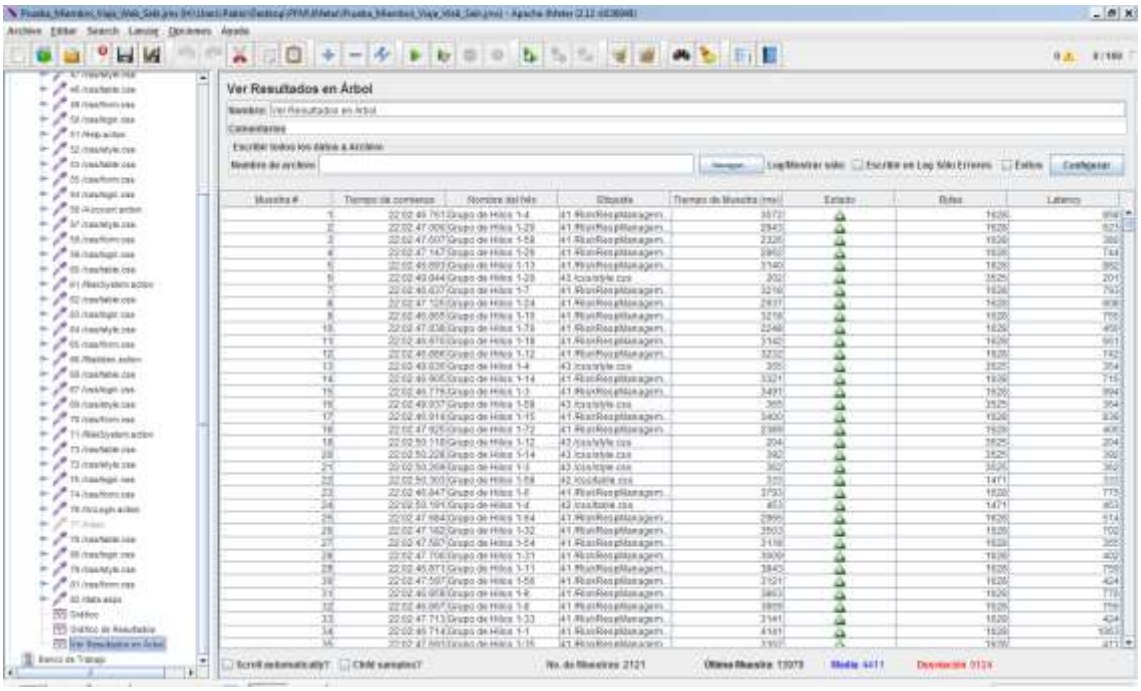


Figura 8.28. Resultados de las Operaciones HTTP a través de la Aplicación de un Miembro

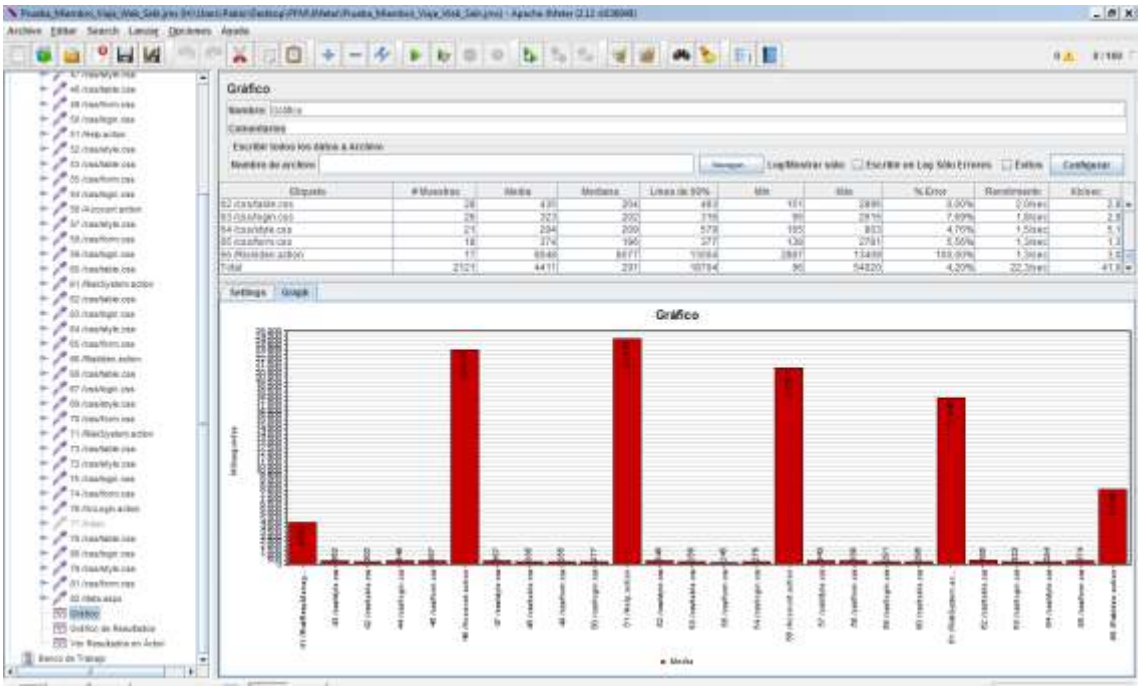


Figura 8.29. Gráfico con los Tiempos de cada Proceso del Viaje por la Web

Los resultados dicen que con 100 usuarios concurrentes (threads) con periodos de subida de 1 segundo, que trabajan en un bucle infinito pueden invocar de todas las acciones que puede realizar un miembro son 22.3 peticiones por segundo en total. Y el tiempo medio para la petición de los diferentes Servicios es de entre 245-24216 ms debido este tiempo tan grande a que me cargue el servidor en un momento dado por las pruebas repetidas, las peticiones sobre los actions son aquellos que tardan más como se puede observar.

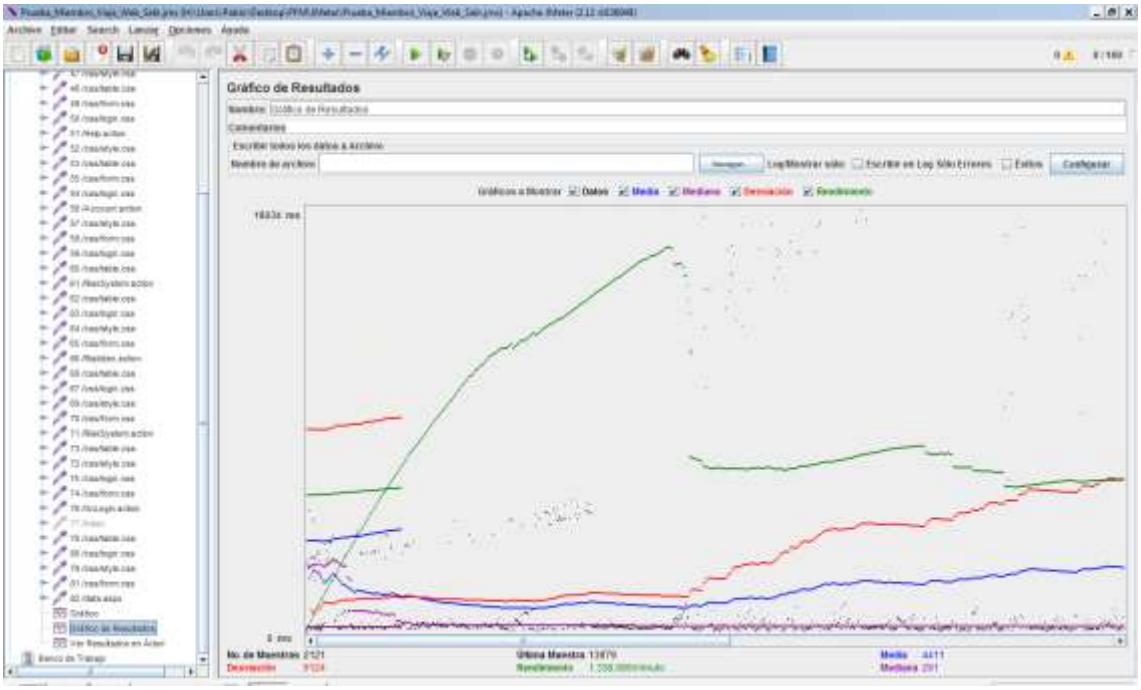


Figura 8.30. Gráficos con los Resultados para el Recorrido por la Web

8.4.2 Peticiones JDBC

Se crea la prueba para peticiones JDBC para operaciones a la base de datos, para realizar pruebas “INSERT”, “SELECT” y “DELETE”, sobre una tabla para testear estas acciones.

Se copia el conector MySQL JDBC en la carpeta lib de la instalación JMeter. El JMeter necesita un driver JDBC adecuado para conectarse a la base de datos.

Se crea la tabla para testearla:

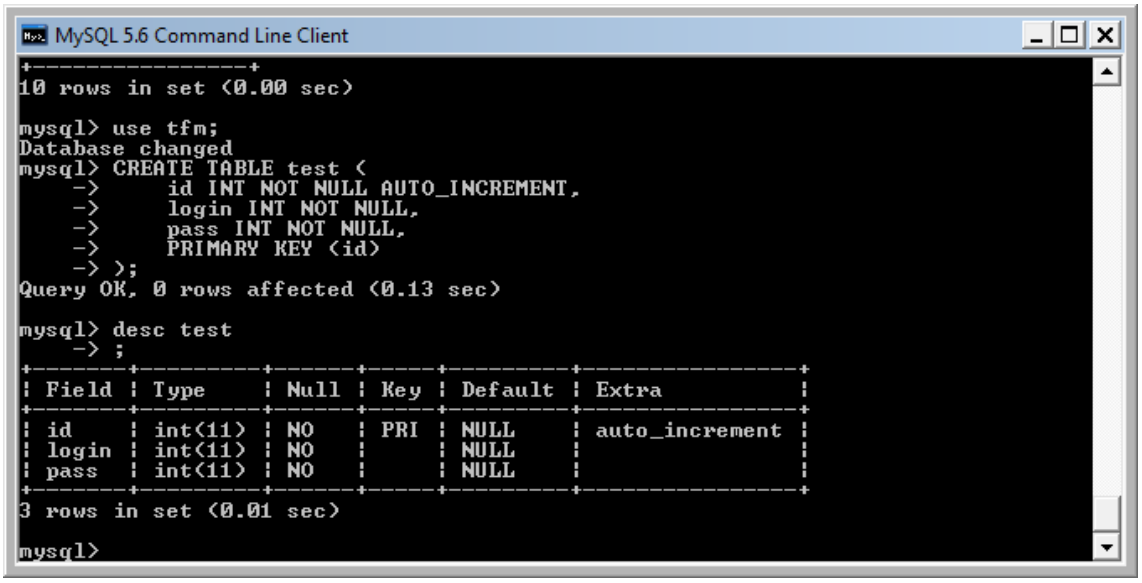


Figura 8.31. Tabla Test Petición JDBC

Se crea un plan de pruebas, se agregan los siguientes componentes de JMeter para el plan de pruebas.

1. **Grupo de Hilos** llamado “JDBC Test”
2. Tres elementos **Muestradores** para cada una de las operaciones (“INSERT”, “SELECT” y “DELETE”), escogiendo la **petición JDBC**
3. **Elemento de Configuración** del tipo de **conexión JDBC**
4. Dos **Elementos de Configuración** del tipo de **Variable aleatoria**
5. Dos **Receptores** del tipo **1. Reporte resumen y 2. Gráfico de resultados**

Configurar los usuarios de la bases de datos. El componente del **Grupo de Hilos**, simula los usuarios de bases de datos.

1. Número de usuarios (hilos)
2. ¿Cuántas veces el usuario enviara la solicitud?

Si selecciona *"Sin fín",* se ejecutará un bucle *while (true) {...}* hasta que se decida detener la prueba.

Configurar la conexión JDBC. Desde el componente **Configuración de la conexión JDBC,** se utiliza para crear conexión JDBC con la base de datos. Con los siguientes datos

1. URL de la base de datos
2. El tipo de controlador JDBC
3. El nombre de usuario
4. La contraseña para conectarse a la base de datos

Este conjunto de conexión se identifica por ***Nombre de Variable****, que será "database"*. Este nombre de variable se utilizará en las **Muestras** de JDBC (peticiones), utilizar esto en **Nombre de Variable Ligada al Pool**, escribiendo en **Nombre de Variable: "**database"

Definir las variables aleatorias que se utilizarán en las instrucciones. Para esta prueba estoy utilizando dos variables *aleatorias:*

1. *login*
2. *pass*

Se configuran las peticiones JDBC, para cada una de las operaciones se añaden los receptores donde se mostraran los resultados y se ejecuta la prueba sobre la base de datos MySQL en local.

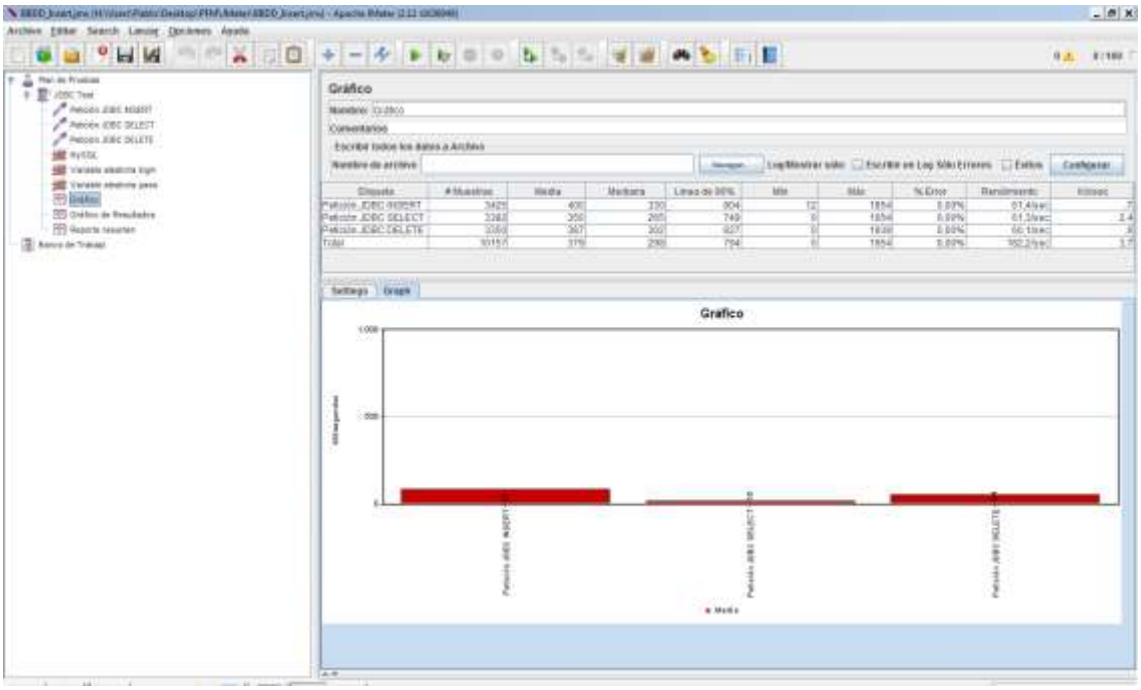


Figura 8.32. Gráfico con los Tiempos de cada Operación en la BBDD

Los resultado que se muestran se han realizado para la prueba con 100 usuarios, con periodos de subida de 10 segundos, trabajando en un bucle infinito realizando a la vez las operaciones de INSERT,SELECT Y DELETE.

Se pueden insertar 61.4 filas en la tabla por segundo. Y la media para insertar una fila es de 400 ms.

Para la operación SELECT, puede realizar 64.3 por segundo y la media de la realiación de una select es de 350 ms.

Para la operación DELETE, se pueden realizar 60.1 por segundo, y la media de cada borrado tarda 387 ms.

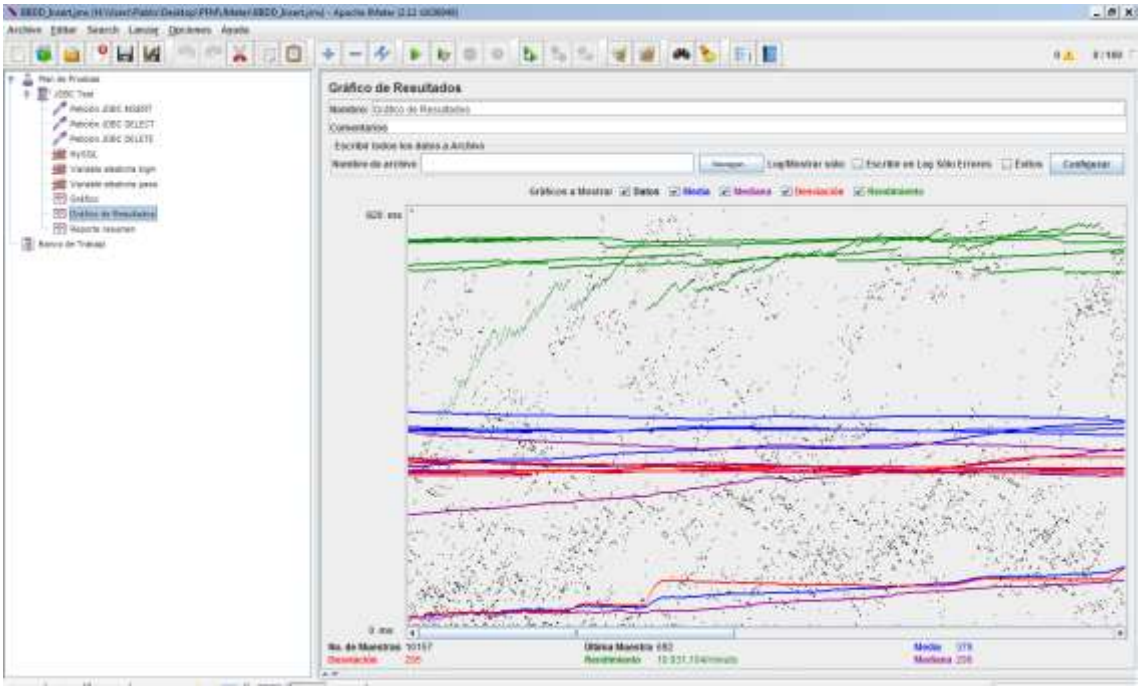


Figura 8.33. Gráficos con los Resultados de las Operaciones en BBDD

Capítulo 9. Manuales del Sistema

9.1 Manual de Instalación

9.1.1 Java

9.1.1.1 JDK y JRE

"Java Development Kit"(JDK), "Standard Development Kit" (SDK) y "Java 2 Standard Edition" (J2SE) son nombres para el mismo componente e incluyen: La API de Java, el JRE, compilador de Java y otras funcionalidades definidas por Sun.

El API ("Application Programming Interface") de Java es un conjunto de clases que es utilizado para generar programas básicos en el lenguaje, estas clases tienen la misma funcionalidad que las funciones/clases estándar utilizadas en otros lenguajes C, C++.

Partiendo de estas clases se generan todos los programas, interfaces y elementos programados en Java, inclusive a partir de estas clases puede definir otras clases específicas que serán utilizadas por su programa o producto.

Una vez que defina sus programas/clases en Java aún es necesario compilarlas, y es esto lo que interpreta el JRE ("Java Runtime Environment").

Es utilizado solo para *ejecutar* ("**Runtime**") programas en Java. Esta situación se da cuando se diseñan alguna interfaces gráficas o aplicaciones en Java. Cabe mencionar que muchos productos que utilizan Java para su interfaz gráfica o instalación *ya incluyen un JRE* para evitar la molestia de instalarlo, aunque muchos productos requieren que lo tenga (no todas porque requiere de un costo a la empresa por ponerlo en su software).

Descarga: JDK

Configuración

Por ejemplo, si va a instalar la actualización JDK 7 versión de actualización 1, la siguiente cadena que representa el nombre del paquete:

```
jdk-7<versión[20]> -linux-i586.tar.gz ←→se convertiría en: jdk-
7u1-linux-i586.tar.gz
```

En el ejemplo anterior, el número de la <versión> es a veces precedido por la letra u, por ejemplo, 7u2, a veces va precedido de un guión bajo, por ejemplo, jdk1.7.0_02.

[20] La notación siguiente se deberá sustituir por el número de versión de actualización adecuada para el JDK

Las diferentes versiones y quien puede instalarlas.

Download File	Architecture	Who Can Install
jdk-7u<*version*>-linux-x64.tar.gz	64-bit	anyone
jdk-7u<*version*>-linux-i586.tar.gz	32-bit	anyone
jdk-7u<*version*>-linux-x64.rpm	64-bit RPM-based Linux	root
jdk-7u<*version*>-linux-i586.rpm	32-bit RPM-based Linux	root

La instalación puede llevarse a cabo utilizando cualquiera de lo siguiente:

Instalación de Oracle Linux JDK utilizando los archivos binarios (.tar.gz) le permiten instalar una versión privada del JDK para el usuario actual en cualquier lugar, sin afectar a otras instalaciones de JDK. Sin embargo, puede implicar pasos manuales para obtener algunas de las características a trabajar (por ejemplo, la -version: opción liberación del mandato java que permite especificar la versión que se utiliza para ejecutar la clase especificada requiere la ruta correcta de la versión del JDK en / usr / jdk).

Instalación de Oracle Linux JDK utilizando paquetes RPM le permite realizar una instalación de todo el sistema del JDK para todos los usuarios, y requiere acceso de root. Plataformas Linux basadas en RPM están basadas en Red Hat y SuSE.

Nota: Al descargar e instalar el Java Development Kit (JDK), Associated Java Runtime Environment (JRE) también se instala.

JDK 7u6 y versiones posteriores incluyen JavaFX SDK (versión 2.2 o posterior). El SDK de JavaFX y Runtime están instalados e integrados en la estructura del directorio JDK estándar.

Instalación JDK en plataformas Linux de 32 o 64-bit

1. Descargue el archivo. Antes de que el archivo pueda ser descargado, debe aceptar el contrato de licencia. El archivo binario puede ser instalado por cualquier persona (no sólo los usuarios root), en cualquier lugar que usted puede escribir. Sin embargo, sólo el usuario root puede instalar el JDK en la ubicación del sistema.
2. Cambie el directorio a la ubicación en la que desea que el JDK que se instale. Mueva el archivo binario tar.gz en el directorio actual.
3. Descomprime el código e instalar el JDK.

```
Versión 64
% tar zxvf jdk-7u<version>-linux-x64.tar.gz
Versión 32
% tar zxvf jdk-7u<version>-linux-i586.tar.gz
```

   Los archivos del kit de desarrollo de Java se instalan en un directorio llamado jdk1.7.0_ <versión> del directorio actual.

4. Elimine el archivo .tar.gz si quiere ahorrar espacio en disco.

Instalación JDK para plataformas Linux de 32 y 64-bit en RPM

1. Descargue el archivo. Antes de que el archivo puede ser descargado, debe aceptar el contrato de licencia.
2. Conviértase en root ejecutando **su** introduzca la contraseña de **superusuario**.
3. Desinstale cualquier instalación anteriores de los paquetes de JDK.

```
# rpm -e <package_name>
```

4. Instale el paquete.

```
Versión 64-bit
# rpm -ivh jdk-7u<version>-linux-x64.rpm
Versión 32-bit
# rpm -ivh jdk-7u<version>-linux-i586.rpm
```

Para actualizar el paquete:

```
Versión 64-bit
# rpm -Uvh jdk-7u<version>-linux-x64.rpm
Versión 32-bit
# rpm -Uvh jdk-7u<version>-linux-i586.rpm
```

5. Elimine el archivo .rpm si quieres ahorrar espacio en disco.
6. Salga del shell de root. No hay necesidad de reiniciar.

9.1.1.2 J2EE (Java 2 Enterprise Edition)

J2EE es un grupo de *especificaciones* diseñadas por Sun (ahora de Oracle) que permiten la creación de aplicaciones:

- Acceso a base de datos (**JDBC**)
- Utilización de directorios distribuidos (**JNDI**)
- Acceso a métodos remotos (**RMI/CORBA**)
- Funciones de correo electrónico (**JavaMail**)
- Aplicaciones Web (**JSP y Servlets**)

Aquí es ***importante notar que J2EE es solo una especificación***, esto permite que diversos productos sean diseñados alrededor de estas especificaciones los cuales algunos necesitan, como por ejemplo el Tomcat.

Descarga: J2EE (Java 2 Enterprise Edition)

Instalación para plataformas Linux

1. En el directorio donde descargó el archivo de distribución.

2. Cambie el permiso del archivo de distribución para que tenga acceso de ejecución:

```
chmod + x distribution_filename
```

   El nombre del archivo de distribución cambia en función de la distribución, la versión y la plataforma, con los formatos:

```
sjas_pe-version-platform.bin     or     j2eeskd-version-
platform.bin
```

3. Corra el programa de instalación.

   Para ejecutar el programa de instalación que utiliza una interfaz gráfica, en el símbolo del sistema, escriba el nombre del fichero de distribución:

```
./distribution_filename
```

   Para ejecutar el programa de instalación que utiliza la interfaz de línea de comando, en el símbolo del sistema, escriba el nombre del archivo de distribución seguido por la opción ***-console***:

```
./distribution_filename -console
```

4. En la página de **Configuración de Administración** (o en la línea de comandos), escriba lo siguiente:
   - Admin User Name-Nombre del usuario que administra el servidor
   - Password-La contraseña del usuario Administrador para acceder al servidor de administración (de 8 caracteres como mínimo)
   - Admin Port-Número de puerto de administración de instancia de servidor predeterminado
   - HTTP Port-Número del puerto para acceder a la instancia de servidor predeterminado
   - HTTPS Port-Número de puerto seguro para acceder a la instancia de servidor predeterminado
5. En la página **Opciones de instalación**, seleccione las opciones que desee.

   Si selecciona la opción "Actualización de la versión anterior", el asistente de actualización comienza después de que se complete la instalación.

6. Ajuste la variable de entorno **PATH** para incluir el directorio del servidor de aplicaciones **install-dir / bin**.
7. Verifique la instalación siguiendo las instrucciones de la "*Guía de inicio rápida Sun Java System Application Server Platform Edition 8.2"*, que se encuentra en **install-dir / docs / QuickStart.html** o en **docs.sun.com**.

9.1.2 MySQL

Descarga del archivo (no necesario)

MySQL de http://dev.mysql.com/downloads/

Descargar la distribución binaria del sitio de MySQL. Aunque aquí se explica la instalación más simple a través de apt-get.

Instalación

1. Para instalar MySQL, se ejecuta el siguiente comando desde una terminal:

```
sudo apt-get install mysql-server
```

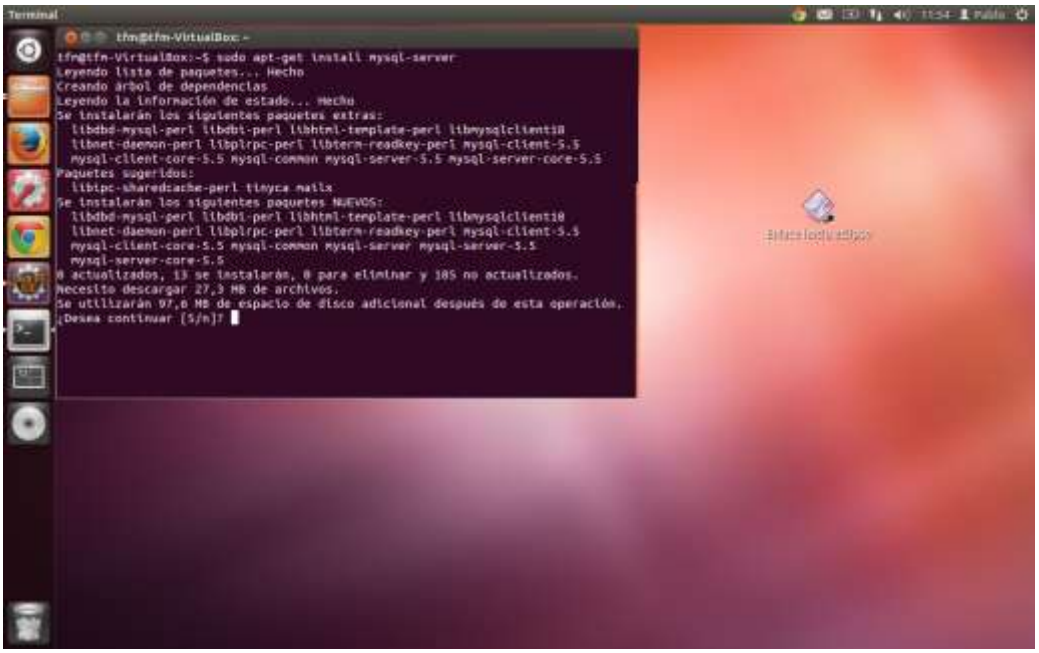


Figura 9.1. Instalación MySQL

2. Durante el proceso de instalación, se le pedirá que introduzca una contraseña para el usuario root de MySQL.

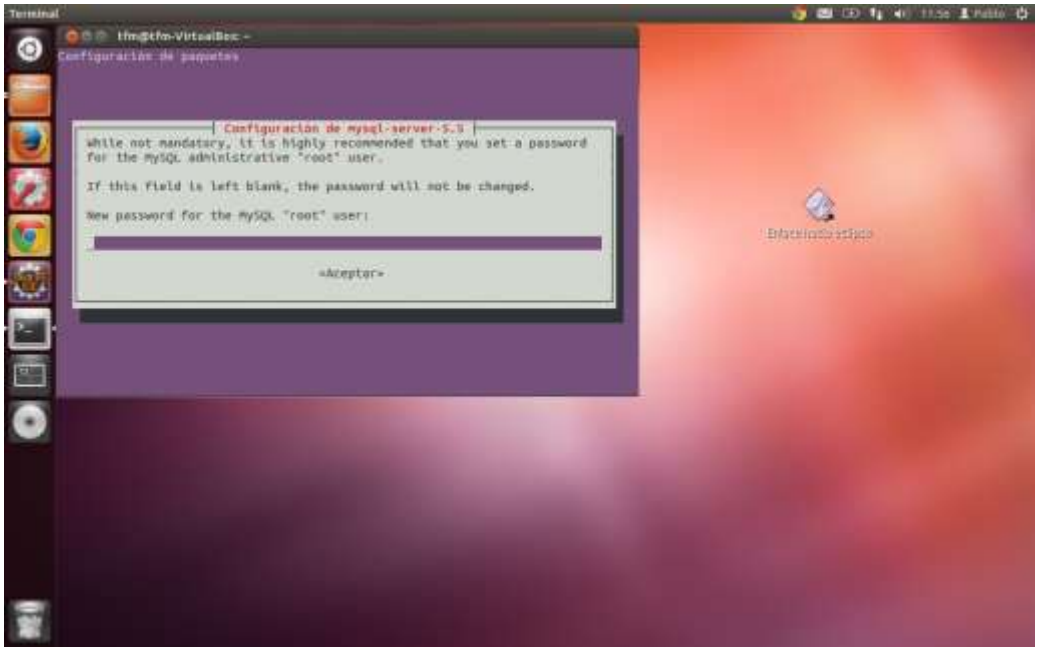


Figura 9.2. Contraseña root MySQL

3. Una vez completada la instalación, el servidor MySQL se iniciará automáticamente. Puede ejecutar el siguiente comando en una terminal para comprobar si el servidor MySQL se está ejecutando:

```
sudo netstat -tap | grep mysql
```

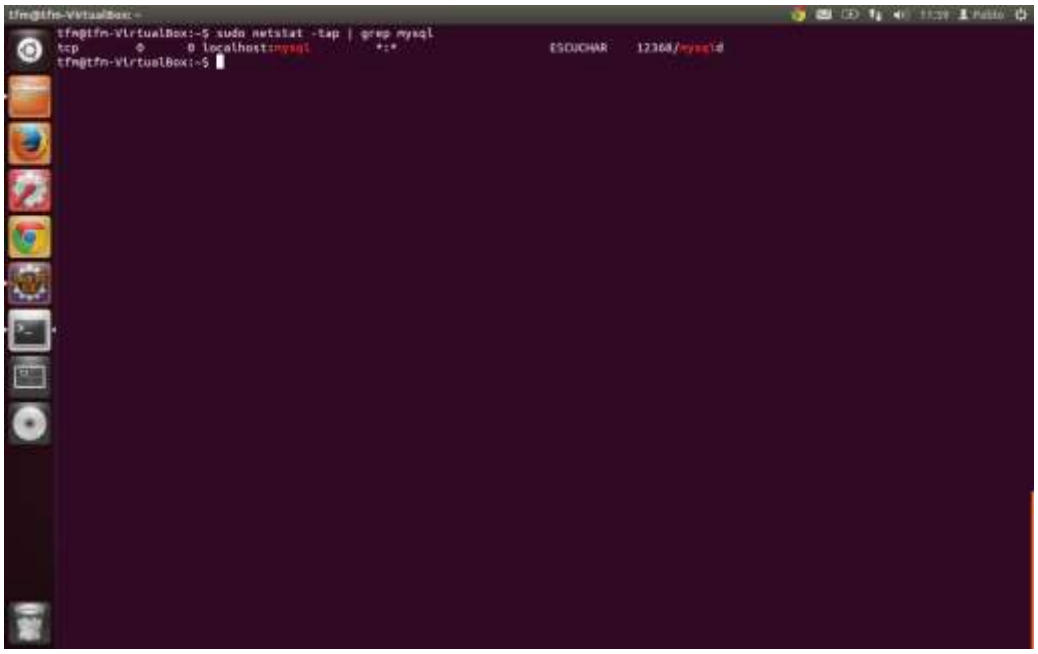


Figura 9.3. Servidor MySQL Arrancado

4. Una vez comprobado que el servidor funciona, entramos por consola dentro de MySQL

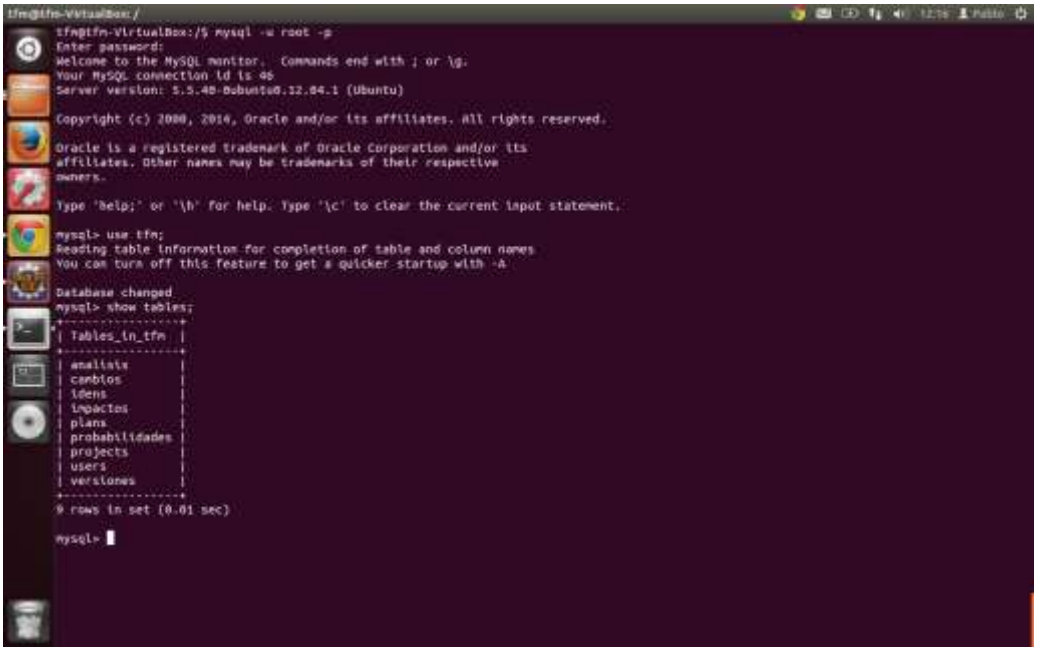


Figura 9.4. Consola MySQL

9.1.3 Tomcat 7.0

Descarga del archivo (no necesario)

Tomcat Apache: http://tomcat.apache.org/download-70.cgi

Descargar la distribución binaria del sitio de Tomcat Apache. Aunque aquí se explica la instalación más simple a través de apt-get.

Instalación

1. Lo primero que tendrá que hacer es actualizar sus listas de paquetes apt-get

```
sudo apt-get update
```

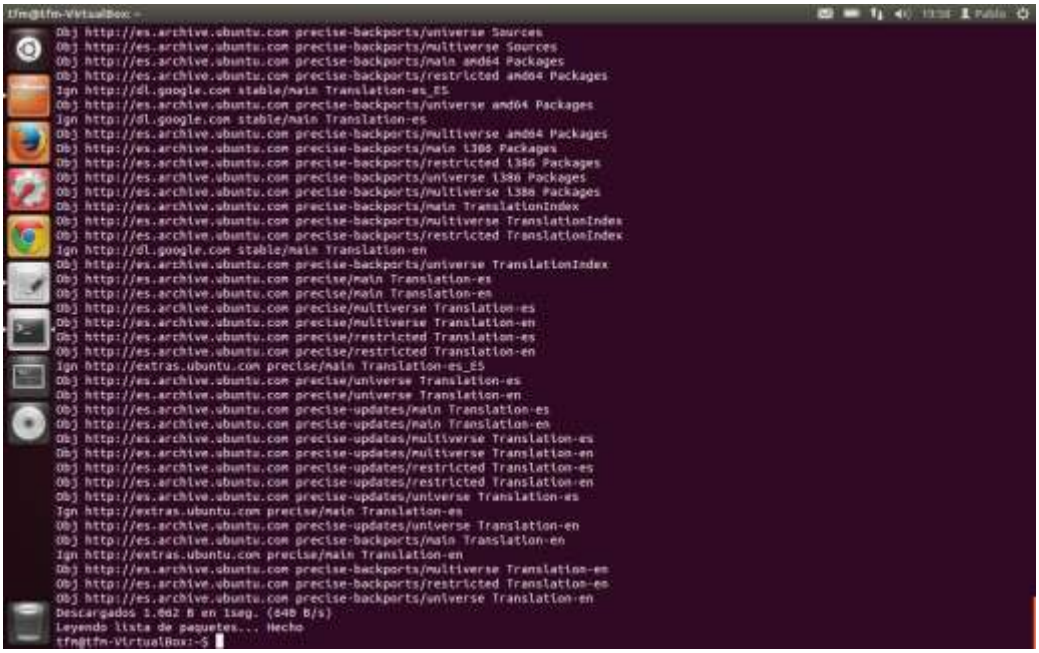


Figura 9.5. Actualizando para Instalar Tomcat7

2. Ahora ya está listo para instalar Tomcat. Ejecute el siguiente comando para iniciar la instalación

```
sudo apt-get install tomcat7
```

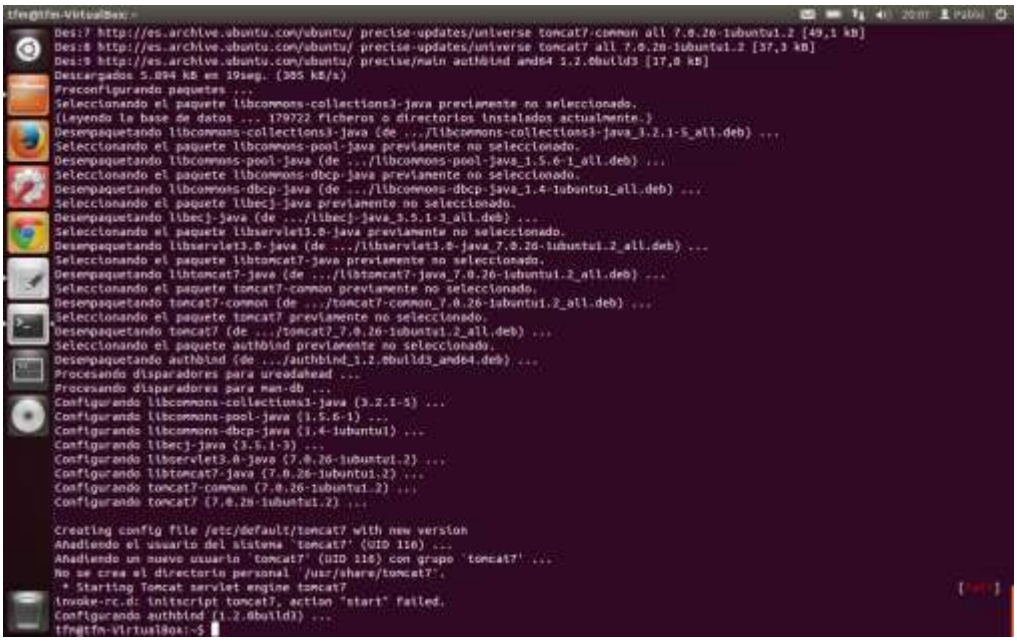


Figura 9.6. Instalando Tomcat7

Conteste sí en el indicador para instalar Tomcat. Esto instalará Tomcat y sus dependencias, como Java, y se creará el usuario tomcat7. También se inicia Tomcat con su configuración predeterminada.

3. Tomcat no está completamente instalado aún, pero se puede acceder a la página de bienvenida por defecto yendo a la dirección IP de ***localhost*** seguido de 8080 en un navegador web

```
http://localhost:8080
```

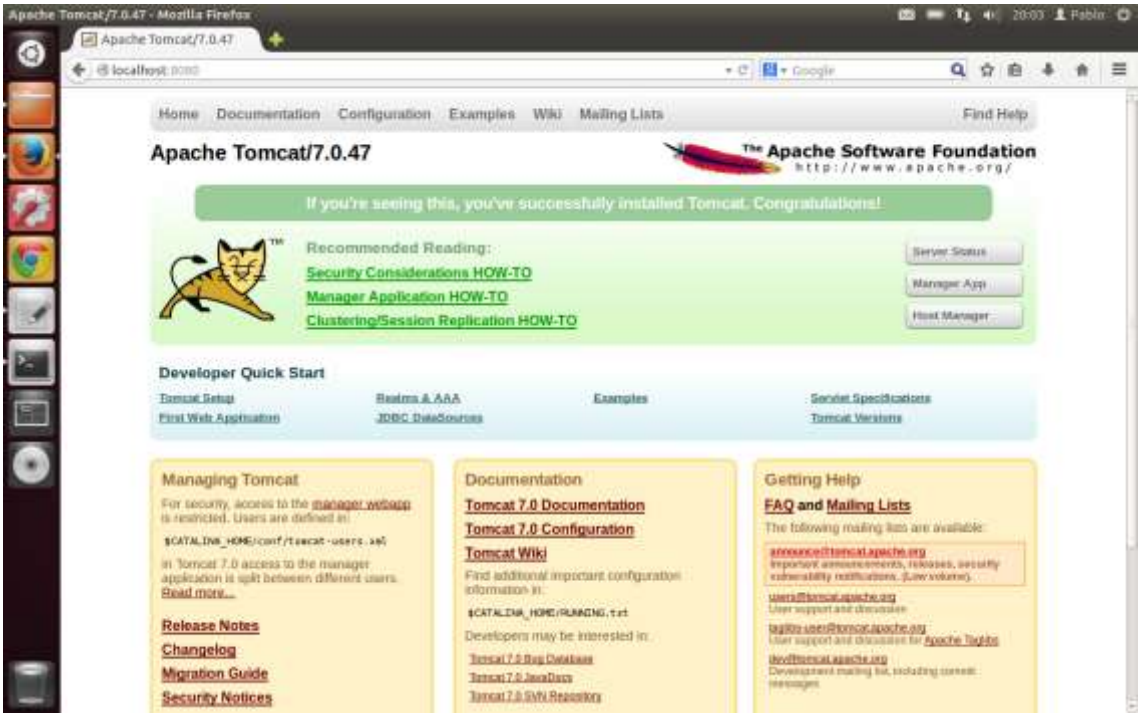


Figura 9.7. Página de Bienvenida Tomcat7

Usted verá la imagen de encima en la página de bienvenida, además de otra información. Ahora vamos a profundizar en la instalación de Tomcat.

Instalación Paquetes Adicionales (No necesaria pero util)

4. Esta sección no es necesario si ya se está familiarizado con Tomcat y que no es necesario utilizar la interfaz de administración web, documentación o ejemplos. Si no es así, por favor continúe.
   Con el siguiente comando, vamos a instalar la documentación de Tomcat en línea, la interfaz web (gerente webapp), y algunos ejemplos de aplicaciones web

```
sudo apt-get install tomcat7-docs tomcat7-admin
tomcat7-examples
```

   Conteste sí en el indicador para instalar estos paquetes.
5. Para utilizar la aplicación web Manager instalado en el paso anterior, hay que añadir una entrada a nuestro servidor Tomcat. Haremos esto editando el archivo *tomcat-users.xml*

```
sudo nano /etc/tomcat7/tomcat-users.xml
```

Este archivo está lleno de comentarios que describen cómo configurar el archivo. Es posible que desee eliminar todos los comentarios entre las dos líneas siguientes, o puede dejarlos si desea hacer referencia a los ejemplos

Figura 9.8. Configurar Usuarios Tomcat7

Usted tendrá que añadir un usuario que puede acceder al Manager-gui y al admin-gui (la interfaz de gestión que hemos instalado en el Paso cuatro). Puede hacerlo mediante la definición de un usuario similar al ejemplo siguiente. Asegúrese de cambiar la contraseña y nombre de usuario si lo desea

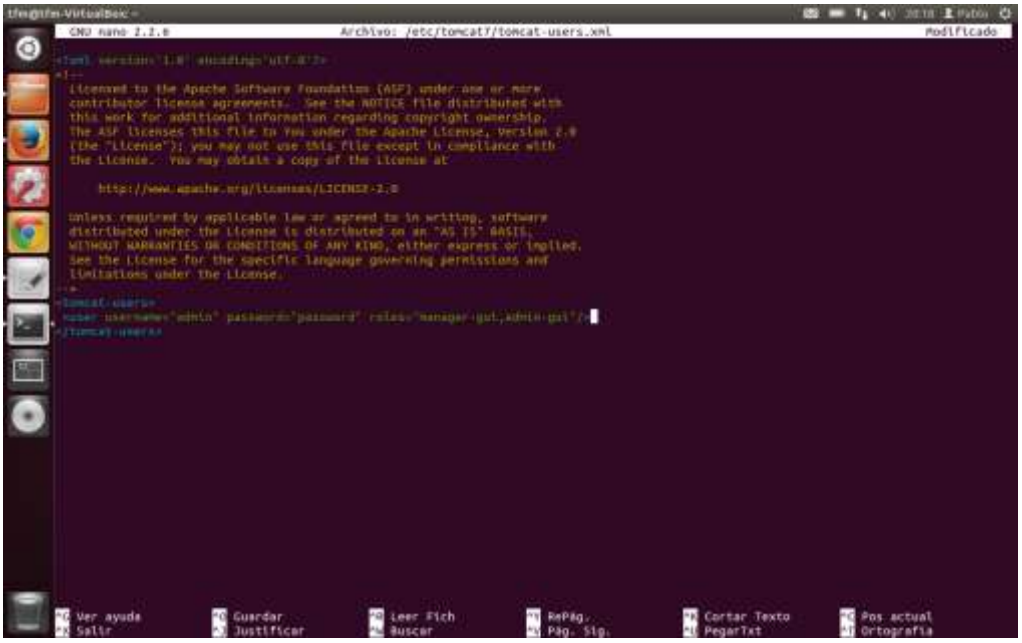


Figura 9.9. Agregar Nuevo Usuario Tomcat7

Guarde y salga del archivo tomcat-users.xml. Para que los cambios hagan efecto, reinicie el servicio de Tomcat

```
sudo service tomcat7 restart
```

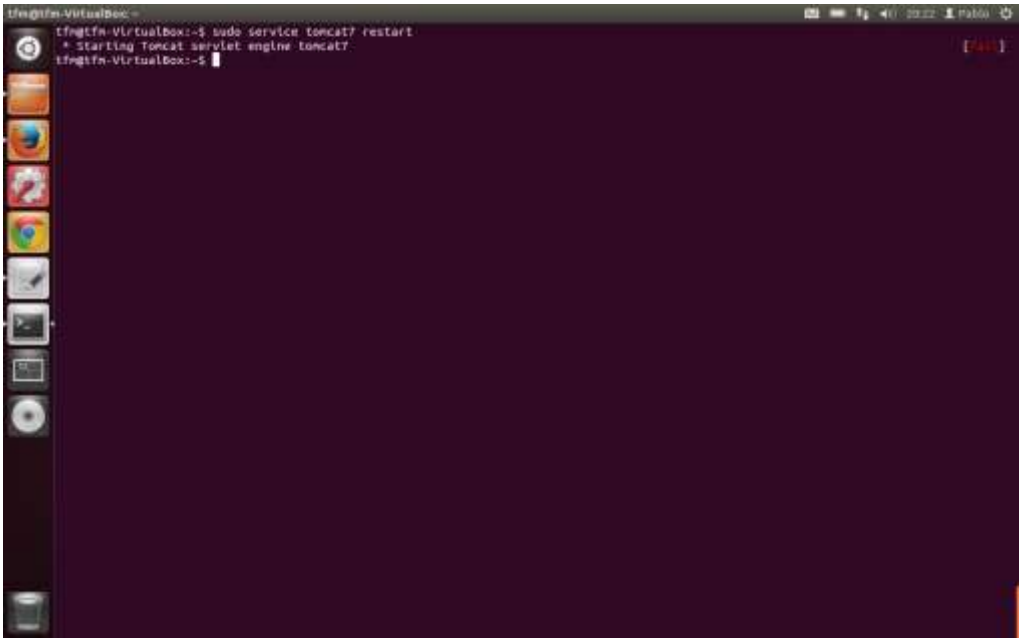


Figura 9.10. Reiniciar Tomcat7

6. Ahora que hemos configurado un usuario Administrador, vamos a acceder a la interfaz de administración web en un navegador web
   Podrá acceder al Administrador de aplicaciones Web, accesible a través del enlace que hay en la página

   ```
   http://localhost:8080
   ```

   O a través de este link

   ```
   http://localhost:8080/Manager/html
   ```

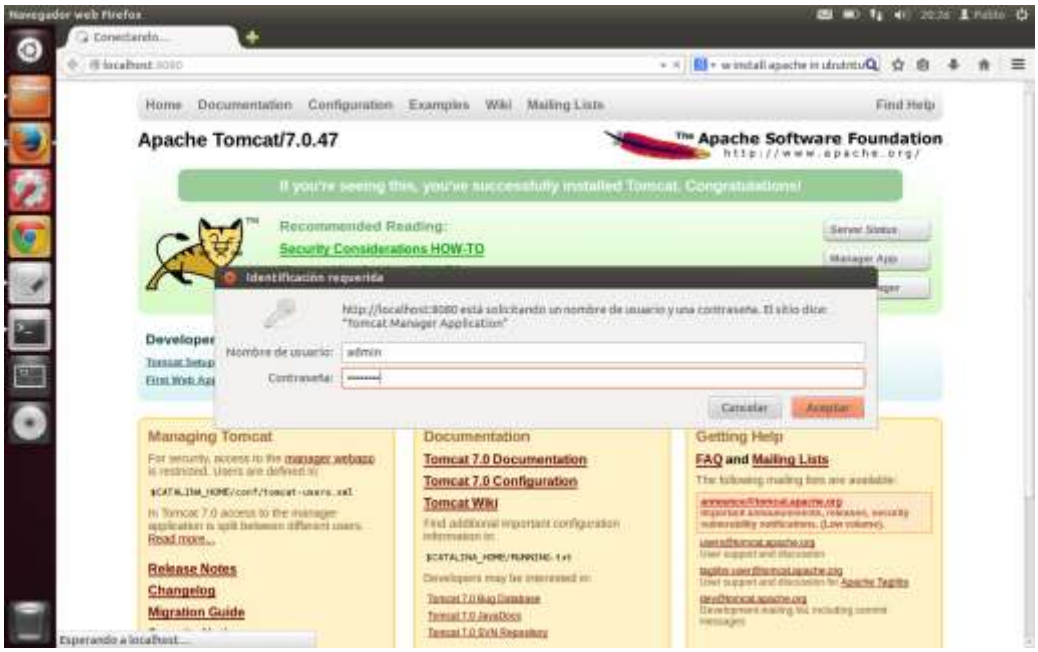


Figura 9.11. Usuario y Contraseña para Acceder

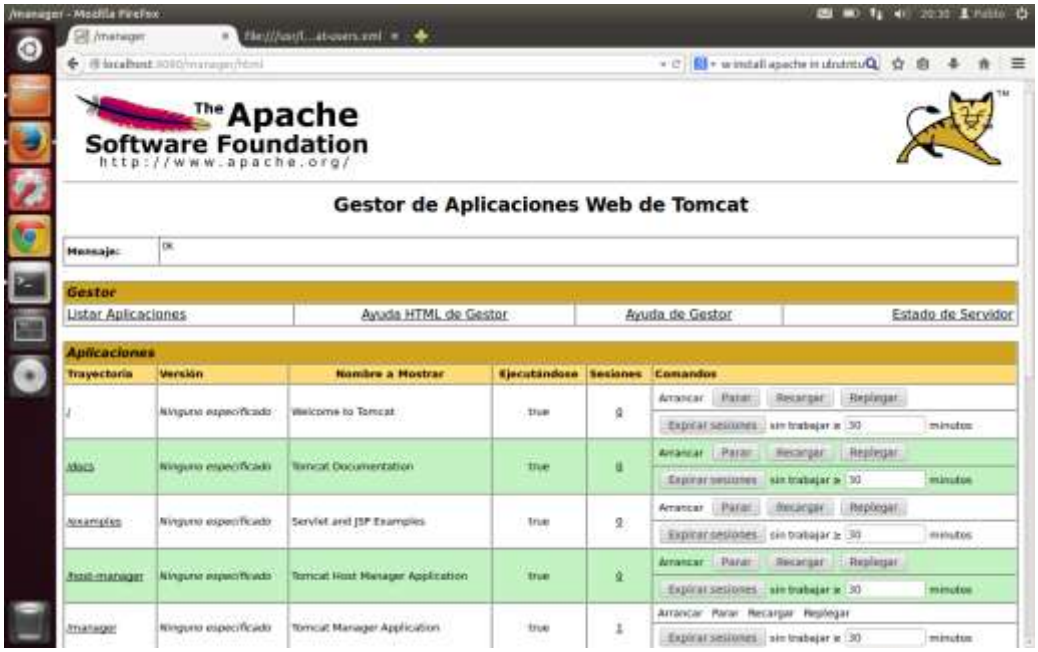


Figura 9.12. Administrador de Aplicaciones Web Tomcat7

El Administrador de aplicaciones Web se utiliza para gestionar las aplicaciones Java. Puede iniciar, detener, recargar, implementar y desplegar aquí. También puede ejecutar algunos diagnósticos sobre sus aplicaciones (es decir, encontrar pérdidas de memoria). Por último, la información sobre el servidor está disponible en la parte inferior de esta página.

9.1.4 VirtualBox

Descarga: https://www.virtualbox.org/wiki/Downloads

Instalación

Después de descargarnos el fichero para OS X (VirtualBox-4.2.18-88780-OSX), lo ejecutamos en Mac donde nos aparecen las diferentes opciones:

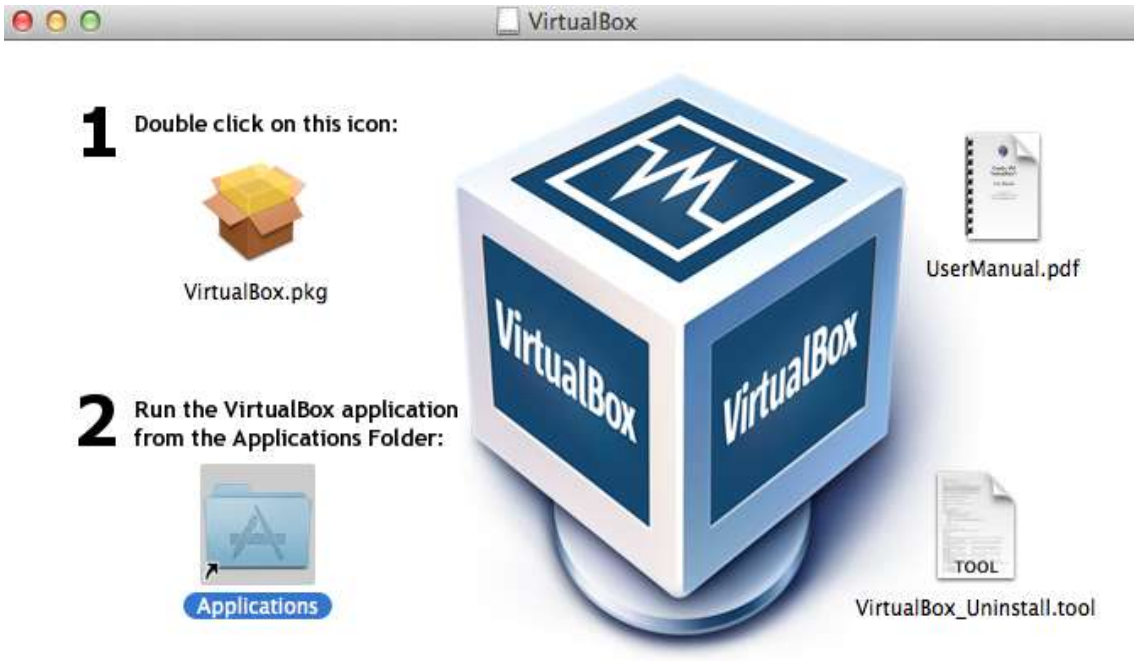


Figura 9.13. VirtualBox

Elegimos ejecutar el fichero VirtualBox.pkg, donde comienza la ejecución del instalador:

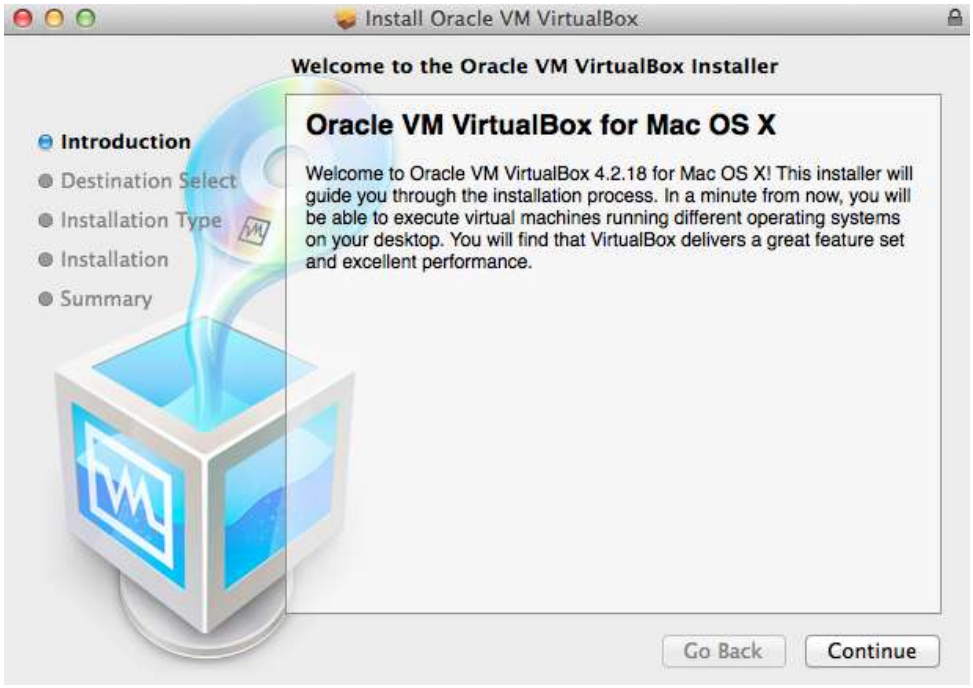


Figura 9.14. Instalación VirtualBox

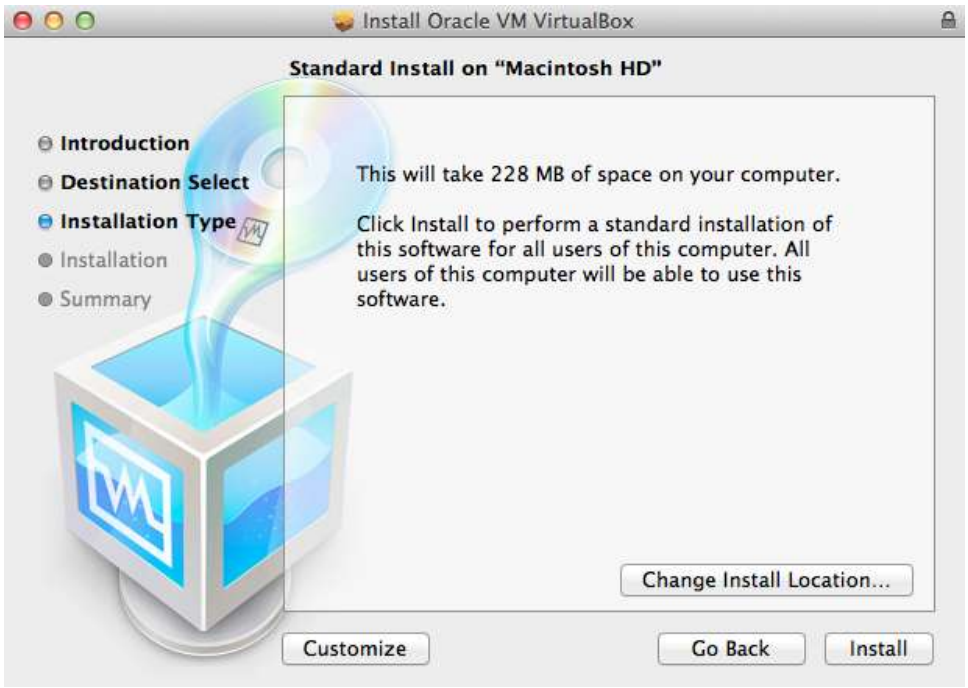


Figura 9.15. Tipo Instalación VirtualBox

Seleccionamos la carpeta destino y nos informa del tamaño de la instalación que se va a producir en el disco.

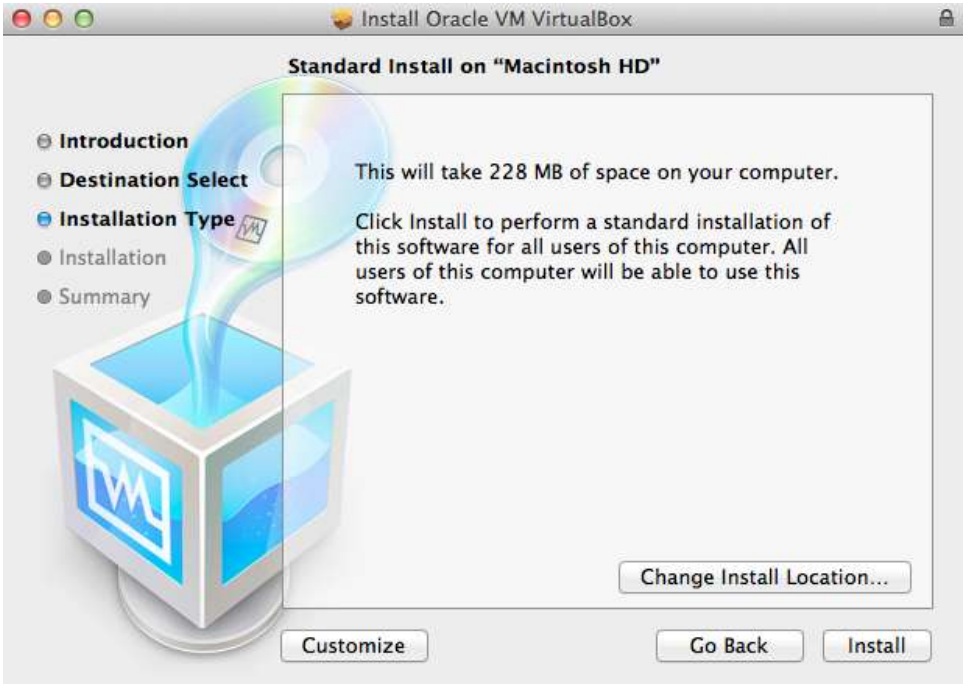


Figura 9.16. Instalando VirtualBox

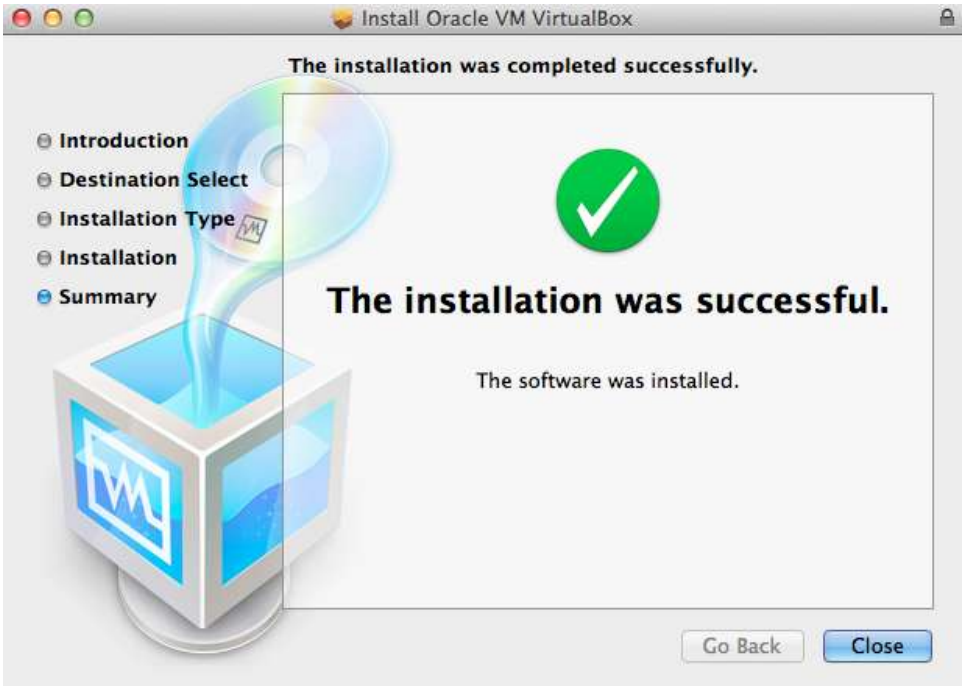


Figura 9.17. Instalación Finalizada del VirtualBox

Una vez que tenemos el VirtualBox instalado comenzamos la configuración e instalación del sistema Ubuntu 64-bit en el emulador.

9.1.5 Máquina Virtual con Ubuntu

Descarga

http://www.ubuntu.com/download

Configuración

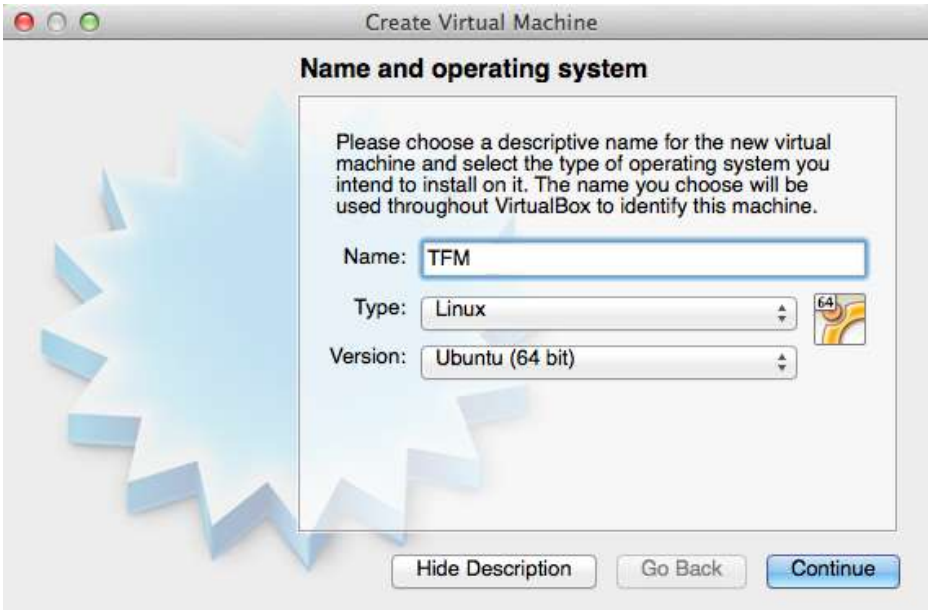


Figura 9.18. Creando Máquina Virtual con Ubuntu

Seleccionamos un nombre, en el caso del ejemplo: **TFM**; un tipo de sistema: **Linux**; y por último una versión de Linux: **Ubuntu (64 bit)**

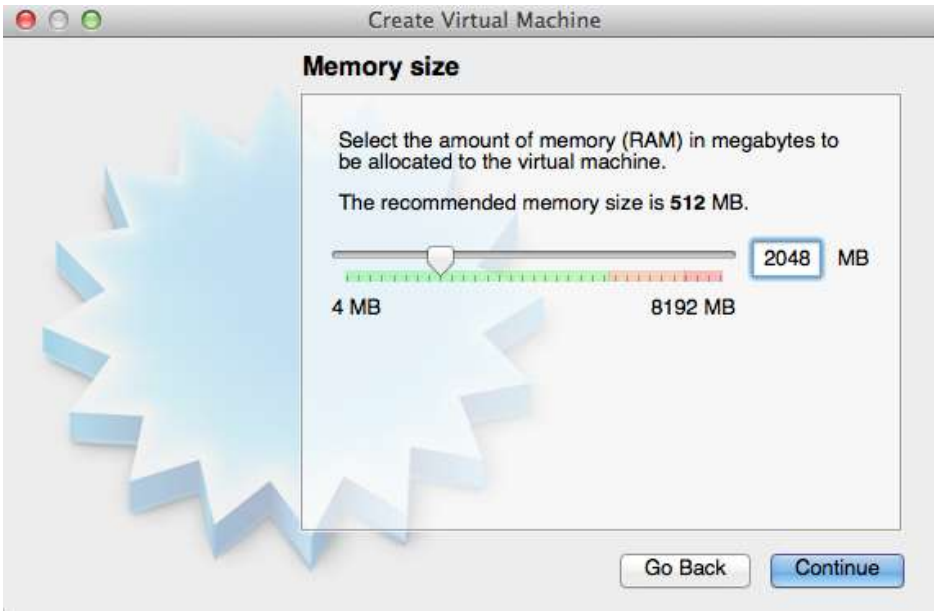


Figura 9.19. Tamaño de Memoria RAM Ubuntu

Seleccionamos en la siguiente ventana el tamaño de la memoria, el recomendado es 512, pero hay memoria de sobra así que la ampliamos hasta 2048.

Seleccionamos en la siguiente ventana, la creación de un disco duro virtual. Utilizaremos unos 30 GB, el recomendado no es suficiente de 8 GB.

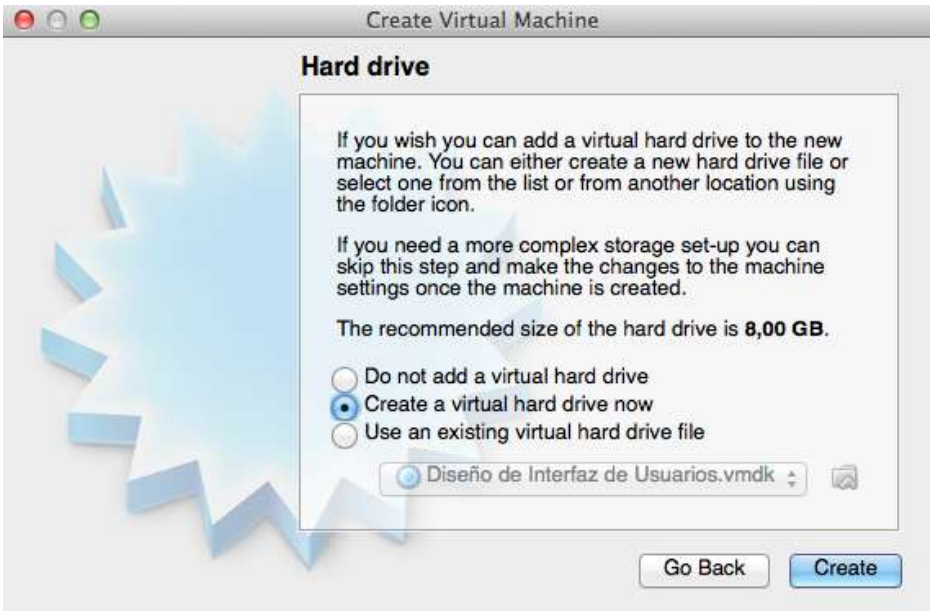


Figura 9.20. Disco Duro Virtual para Ubuntu

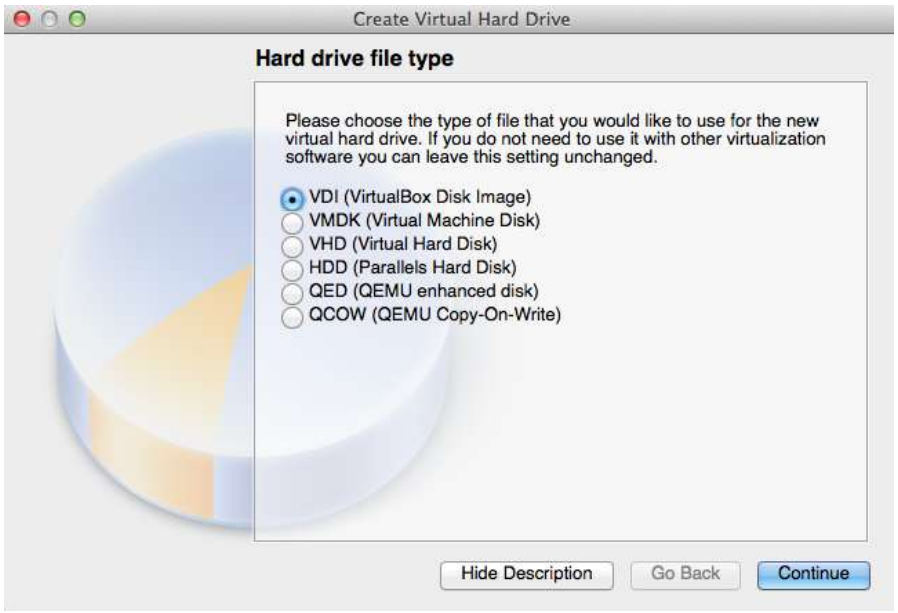


Figura 9. 21. Tipo de Fichero Disco Duro para Ubuntu

El tipo de fichero que utilizaremos, como de momento se va seguir utilizando como máquina virtual está dejamos el que viene por defecto que es el propio de la aplicación VDI.

Figura 9.22. Tipo de Almacenamiento para Ubuntu

Una vez que tenemos la máquina virtual montada para albergar un sistema operativo de Ubuntu, la arrancancamos.

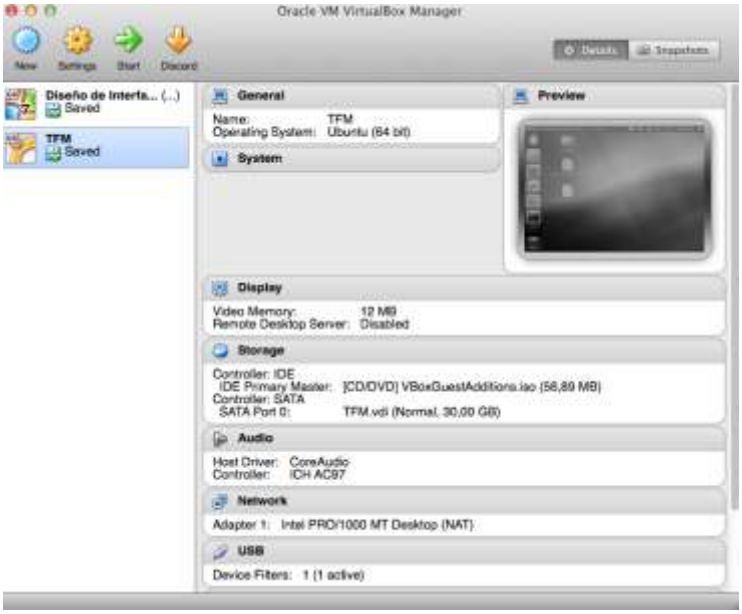


Figura 9 23. Máquina Virtual con Ubuntu

Se nos pregunta si deseamos iniciar con un disco donde se cargue el sistema Ubuntu, buscamos la imagen del sistema operativo Ubuntu que nos hemos descargado anteriormente e iniciamos.

Figura 9.24. Imagen de Ubuntu

9.1.5.1 Ubuntu

Instalación

Una vez que arranca la imagen se comienza con la instalación del sistema operativo Ubuntu

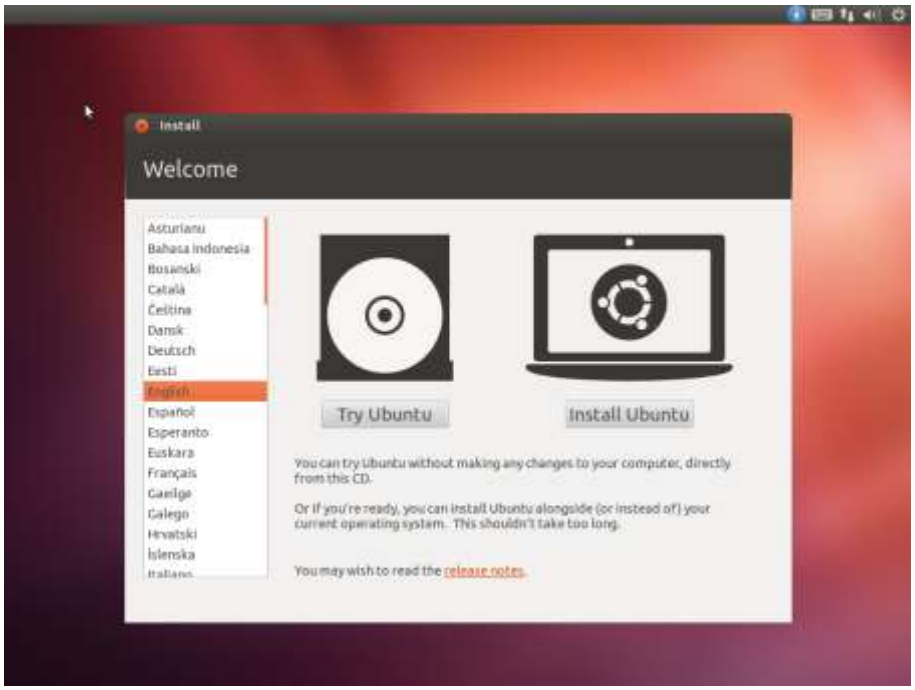


Figura 9.25. Instalar Ubuntu

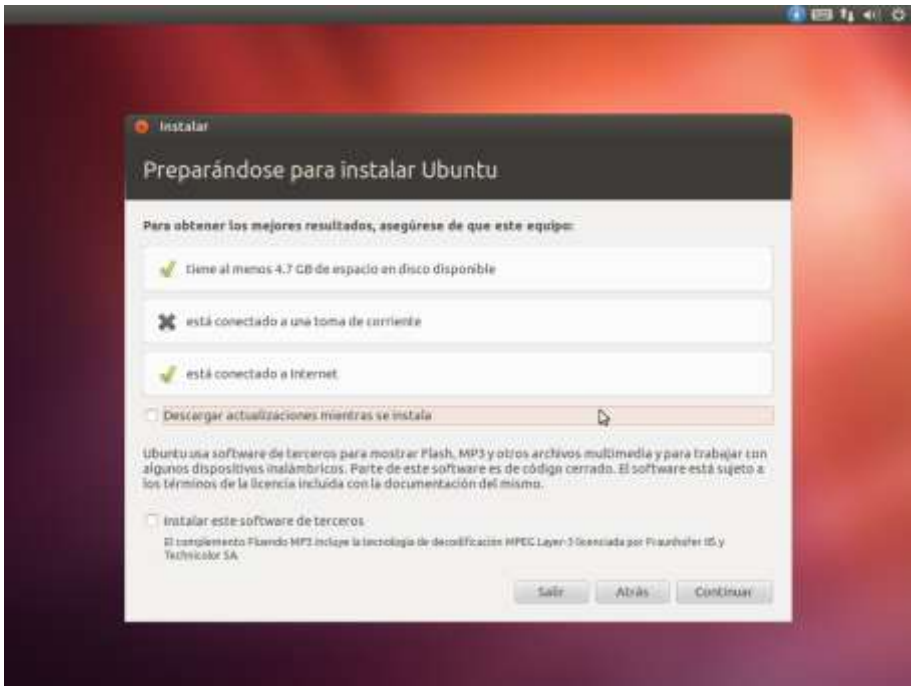


Figura 9.26. Requisitos Ubuntu

Al pulsar sobre el instalador, nos presenta en la ventana siguiente unos puntos con información sobre requisitos que necesita la aplicación.

Figura 9.27. Tipo Instalación Ubuntu

Esta ventana nos muestra el tipo de instalación que podemos realizar, usaremos la primera ya que hemos creado un disco virtual solo para el sistema Ubuntu. Seleccionamos la unidad que está puesta, que es la única que aparece y comenzamos la instalación, del sistema Ubuntu

Figura 9.28. Localización Ubuntu

Rellenamos las diferentes ventanas donde nos pide información.

Figura 9.29. Distribución del Teclado Ubuntu

Figura 9.30. Datos de Acceso Ubuntu

En la última pantalla se nos pide un nombre de usuario y una contraseña que será los que utilicemos para el inicio de sesión.

Una vez relleno estos datos, comenzará con la fase de instalación del resto de ficheros del sistema.

Figura 9.31. Instalando Ubuntu

Por último nos pedirá que reiniciemos el sistema para que se apliquen los cambios realizados y cuando termine nos presentará la pantalla de bienvenida del sistema Ubuntu, donde introduciremos la contraseña previamente escrita anteriormente y ya tendremos listo el sistema Ubuntu para trabajar con el.

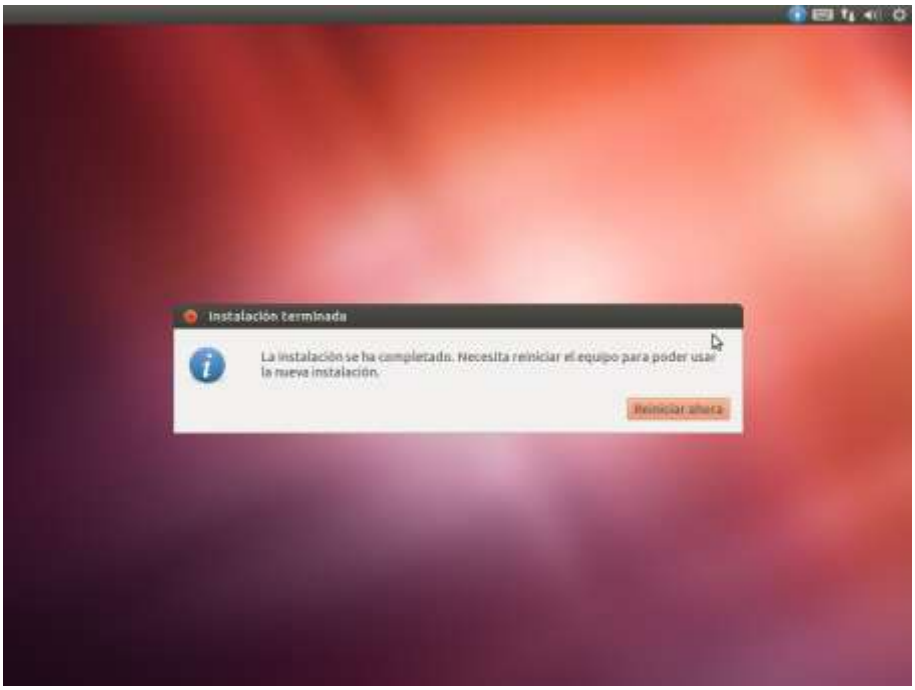


Figura 9.32. Reinicio del Sistema Ubuntu

Figura 9.33. Pantalla de Bienvenida Ubuntu

9.2 Manual de Ejecución

9.2.1 OpenShift

Instalación

Para instalar las herramientas del cliente en Ubuntu requiere **sudoer** o **acceso de root** y comprende cuatro pasos:

Paso 1: Instalación de Ruby

Desde la terminal, ejecute el siguiente comando para instalar Ruby:

```
$ sudo apt-get install ruby-full
```

Se ejecuta el siguiente comando para verificar que Ruby ha instalado correctamente:

```
$ ruby -e 'puts "Welcome to Ruby"'
Welcome to Ruby
```

Paso 2: Instale RubyGems

Ejecute el siguiente comando para instalar RubyGems:

```
$ sudo apt-get install rubygems
```

Paso 3: Instale Git

Ejecute el siguiente comando para instalar el control de versiones Git:

```
$ sudo apt-get install git-core
```

Una vez finalizada la instalación, ejecute el siguiente comando para verificar que Git se ha instalado correctamente:

```
$ git --version
```

Este comando nos devuelve el número de versión de Git que estamos utilizando.

Paso 4: Instalar herramientas de cliente

Cuando el software necesario se ha instalado correctamente, ejecute el siguiente comando para instalar las herramientas de cliente:

```
$ sudo gem install rhc
```

Configuración

Cuando se termine la instalación, abra una ventana de terminal y ejecute:

```
$ rhc setup
```

El asistente de configuración de OpenShift mostrará y solicitara datos que usted debe completetar.

Se le pedirá su nombre de usuario y contraseña OpenShift:

```
Login to openshift.redhat.com: user@example.com
Password: password
```

A continuación se le pregunta si desea generar un token de autorización. Responder que sí , esto hará que se guarde un token en su directorio personal para ser utilizado en las solicitudes posteriores. Cuando se vence, se le pedirá la contraseña de nuevo.

```
OpenShift can create and store a token on disk which allows to you to
access the server without using your password. The key is stored in
your home directory and should be kept secret. You can delete the key
at any time by running 'rhc logout'.
Generate a token now? (yes|no) yes
Generating an authorization token for this client ... lasts about 1
day
```

Después de crear un archivo de configuración, la instalación configurar las claves SSH para que su sistema puede conectarse de forma remota a las aplicaciones, incluyendo el despliegue de sus aplicaciones usando Git:

```
No SSH keys were found. We will generate a pair of keys for you.
Created: C:\Users\User1\.ssh\id_rsa.pub
```

Después de esto se generan las nuevas claves SSH, la clave pública, la id_rsa.pub, debe ser cargado en el servidor OpenShift para autenticar su sistema contra el servidor remoto. Introduzca un nombre que se utilizará para su clave, o dejar en blanco para utilizar el nombre predeterminado. En el siguiente ejemplo se utiliza el nombre predeterminado.

```
Your public ssh key must be uploaded to the OpenShift server to access
code. Upload now? (yes|no) yes
Since you do not have any keys associated with your OpenShift account,
your new key will be uploaded as the 'default' key

Uploading key 'default' from C:\Users\User1\.ssh\id_rsa.pub ... done
```

Después de verificar que el Git está instalado, se le pedirá configurar su dominio si usted no tiene ya uno:

```
Checking for a domain ... none

Your domain is unique to your account and is the suffix of the public
URLs we assign to your applications. You may configure your domain
here or leave it blank and use 'rhc domain create' to create a domain
later. You will not be able to create applications without first
creating a domain.

Please enter a domain (letters and numbers only) |<none>|: MyDomain
Your domain name 'MyDomain' has been successfully created
```

Por último, el asistente de instalación comprueba si existen aplicaciones bajo su dominio. En el siguiente ejemplo, no se han creado aplicaciones. En este caso, el asistente de configuración muestra los tipos de aplicaciones que se pueden crear con los comandos asociados. El asistente de configuración luego se completa mostrando cuantos **Gear** ha usado, junto con el tamaño del **Gear** disponibles para el usuario dado.

```
Checking for applications ... none

Run 'rhc app create' to create your first application.

Do-It-Yourself rhc app create <app name> diy-0.1
JBoss Application Server 7.1 rhc app create <app name> jbossas-7
JBoss Enterprise Application Platform 6.0 rhc app create <app name>
jbosseap-6.0
Jenkins Server 1.4 rhc app create <app name> jenkins-1.4
Node.js 0.10 rhc app create <app name> nodejs-0.10
PHP 5.3 rhc app create <app name> php-5.3
PHP 5.4 rhc app create <app name> php-5.4
Perl 5.10 rhc app create <app name> perl-5.10
Python 2.6 rhc app create <app name> python-2.6
Python 2.7 Community Cartridge rhc app create <app name> python-2.7
Python 3.3 Community Cartridge rhc app create <app name> python-3.3
Ruby 1.8 rhc app create <app name> ruby-1.8
Ruby 1.9 rhc app create <app name> ruby-1.9
Tomcat 6 (JBoss EWS 1.0) rhc app create <app name> jbossews-1.0
Tomcat 7 (JBoss EWS 2.0) rhc app create <app name> jbossews-2.0
Zend Server 5.6 rhc app create <app name> zend-5.6

You are using 0 of 3 total gears
The following gear sizes are available to you: small

Your client tools are now configured.
```

Creando la Aplicación

Se puede utilizar la propia aplicación Web o la herramienta que se instalo anteriormente el RHC, lo haremos con esta última

Utilizando RHC para crear una aplicación con el siguiente comando:

```
$ rhc app create <nombre_app> <web_cartridge_name>
```

Para crear una aplicación con Tomcat7 la denominaremos **miprimerapp**, se utiliza el siguiente comando:

```
$ rhc app create miprimerapp Tomcat7
```

Creando Cambios

OpenShift usa Git para gestionar la implementación de aplicaciones en OpenShift. Para realizar los cambios de código en la máquina local, se comprueba esos cambios primero en local y, a continuación, se debe hacer un **push** de esos cambios en OpenShift. Para ello cada aplicación de OpenShift tiene su propio repositorio Git que sólo usted puede acceder.

Como hemos creado la aplicación desde línea de comandos, RHC descargará automáticamente una copia de ese repositorio (Git llama a esto "clonación") en nuestro sistema local.

Sino desde la consola web, tendrá que indicar a Git donde clonar el repositorio. Busque el Git URL de la página de la aplicación, y luego ejecute en línea de comandos:

```
$ git clone <git_url> <directorio a crear>
```

Subiendo Cambios

Una vez que haya hecho los cambios a su repositorio local, necesita hacer un **add** y luego un **commit** de esos cambios, para cada **commit** necesita de un mensaje que describa el cambio, para estas acciones necesita teclear lo siguiente:

```
$ git add
$ git commit -m "COMENTARIO del cambio"
```

Por último, usted estará listo para subir sus cambios a su aplicación - usted debe hacer un "**push**" de estos cambios con:

```
$ git push
```

9.2.2 MySQL y phpMyAdmin de OpenShift

Una vez que tenemos creada la aplicación podemos acceder a la dirección de la cuenta y verla, en ella podremos agergar sin coste alguno de **Gears,** una base de datos, para nuestra aplicaión usaremos MySQL 5.5.

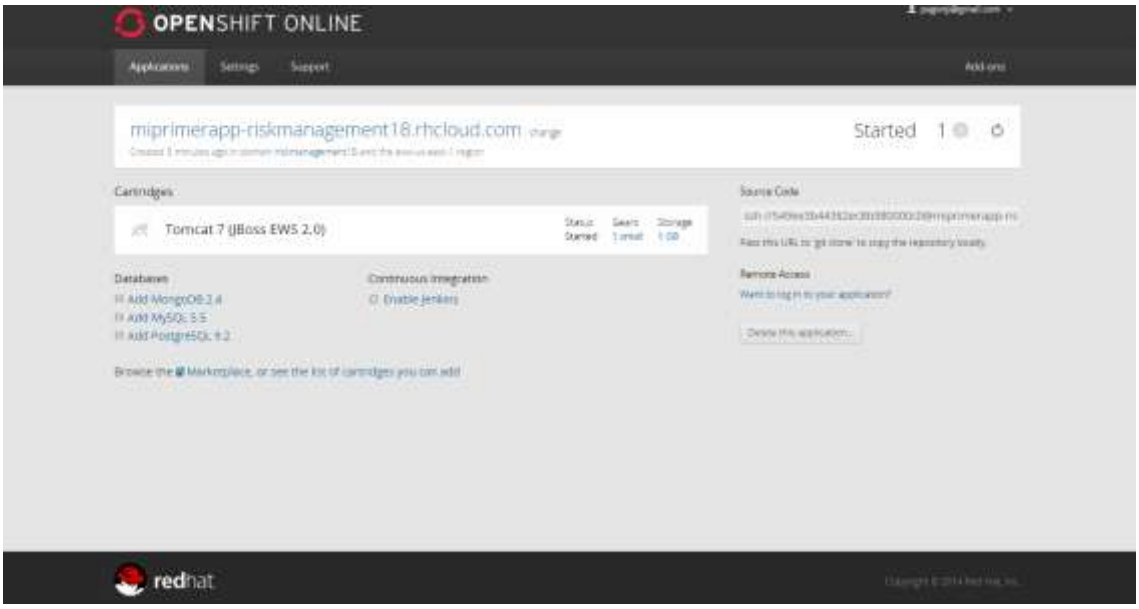


Figura 9.34. Página Aplicación OpenShift

Solo necesitamos pulsar sobre la base de datos que vayamos a añadir a nuestra app.

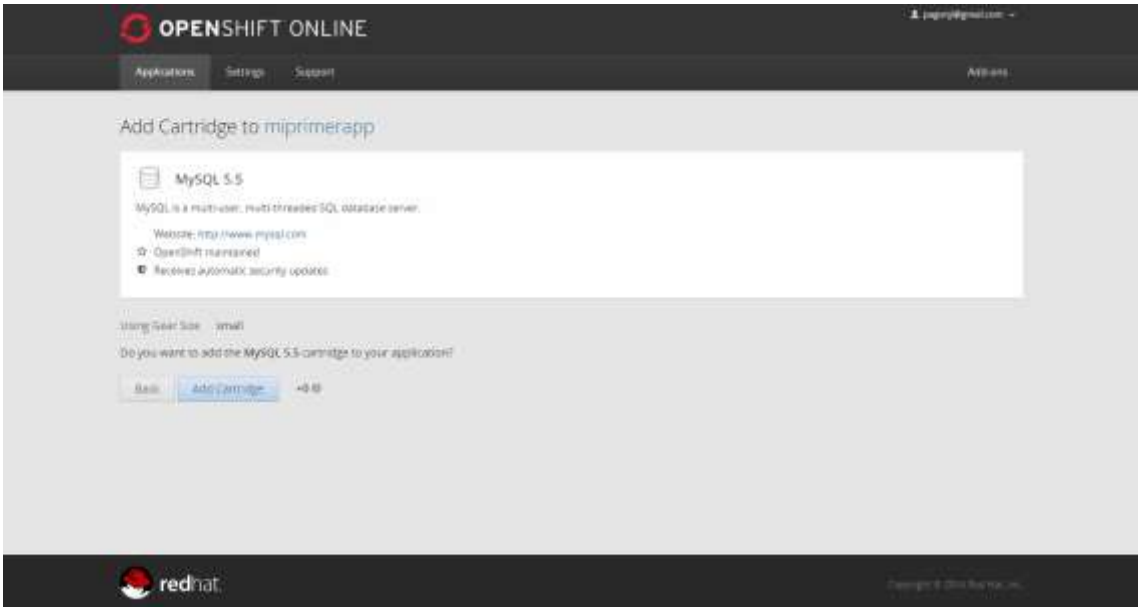


Figura 9.35. Añadir MySQL a APP de OpenShift

Una vez añadida obtendremos los datos de acceso a la misma, como son el usuario root, su contraseña, el nombre de la base de datos y la URL de conexión.

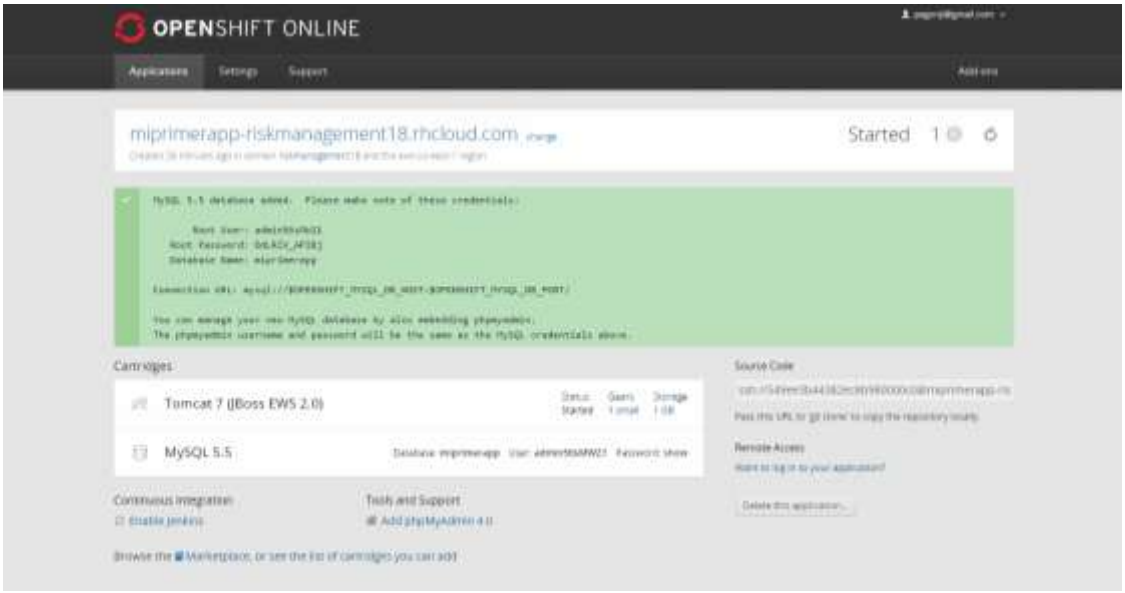


Figura 9.36. Datos de la Base de Datos Creada en OpenShift

Para poder manejar la base de datos, OpenShift nos ofrece también sin coste alguno el Administrador para bases de datos MySQL PHPMyAdmin, como se hiciera antes con la base de datos abajo aparece el enlace para añadirlo a nuestra app.

Los pasos son iguales que para agregar la base de datos

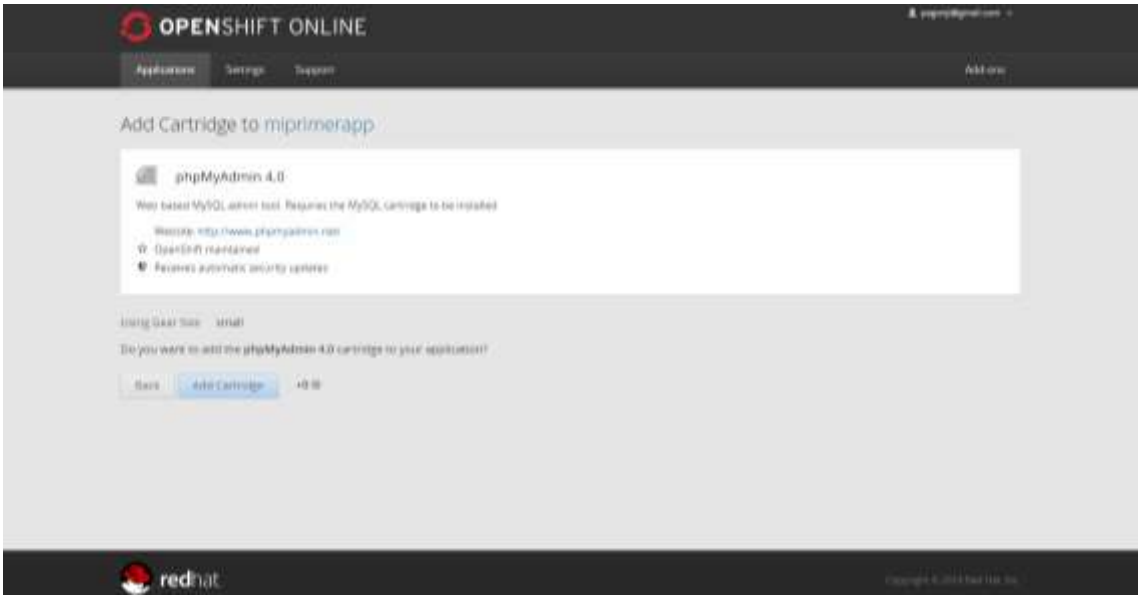


Figura 9.37. Añadir PHPMyAdmin para APP de OpenShift

Una vez agregada, obtendremos los datos para conectarnos al servidor donde está alojada PHPMyAdmin la cual ya contendrá la base de datos de MySQL creado anteriormente para poder administrarla.

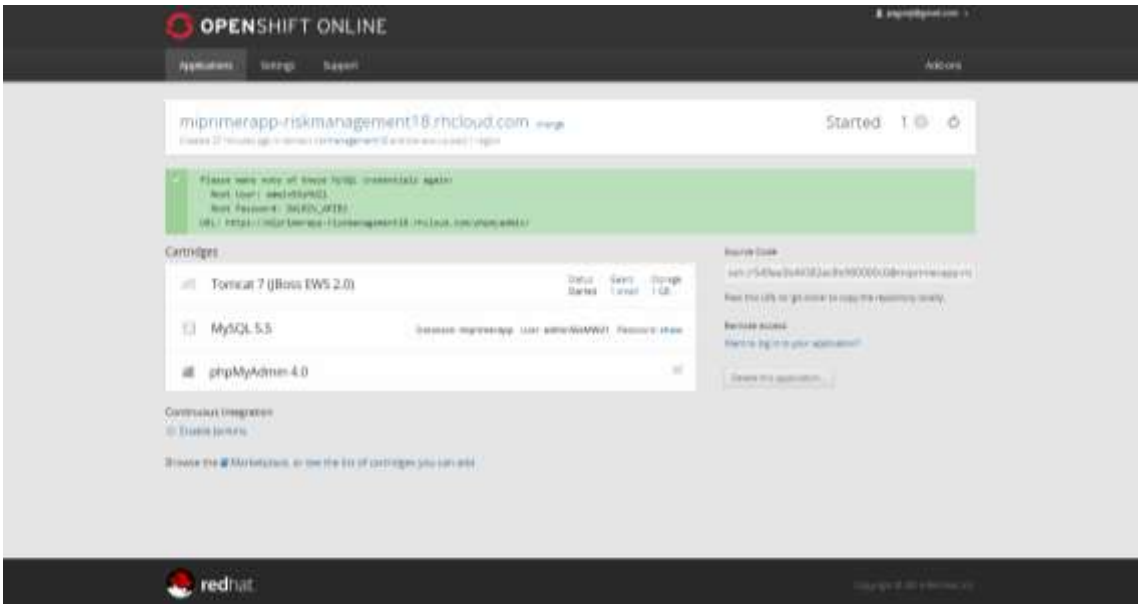


Figura 9.38. Datos de PHPMyAdmin Creada en OpenShift

Con esto ya tendremos todo lo necesario para poder desplegar el fichero **.war**, creado en el entorno de Eclipse, solo necesitaremos alojarlo en la carpeta creada en nuestro directorio local, renombrarla con el nombre de **ROOT.war**, para que la página de inicio nos lleva directamente a nuestra aplicación, subirla al GIT, con los pasos declarados anteriormente y tendremos nuestra aplicación en la nube de OpenShift, donde cualquiera podrá acceder por medio de la URL que aparece en nuestra cuenta de OpenShift.

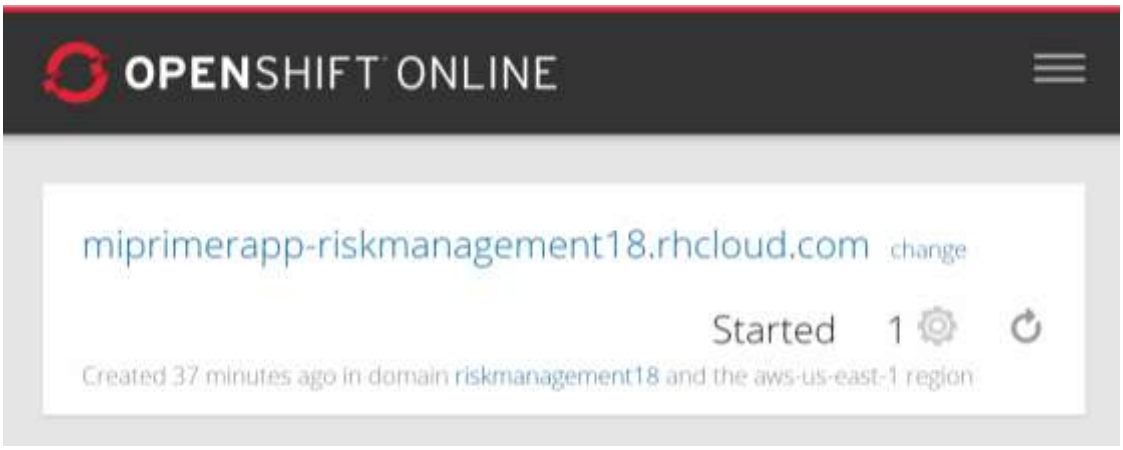


Figura 9.39. URL de nuestra APP

9.3 Manual de Usuario

9.3.1 Aplicación

La aplicación comienza con la pantalla de identificación, dependiendo de qué usuario se logue en la aplicación, podrá realizar unas acciones u otras.

Figura 9.40. Pantalla Inicial

Una vez introducidos los datos de identificación se entrara en la parte privada de cada usuario.

9.3.1.1 Administrador

En la pantalla principal del Administrador, el primer paso sería el completar el registro de un nuevo Manager, quien será el que dirija el proyecto que después deberá registrar también el Administrador.

Figura 9.41. Administrador, Crear Manager

La página contendrá un menú de navegación (recuadro **azul**) por el cual el Administrador podrá desplazarse por las demás páginas. Tendrá un cuerpo donde se realizarán las acciones o/y se visualizarán los resultados de las acciones que el Administrador este realizando.

Tendrá un pie (recuadro **verde**) donde se le indicará desde que IP se está conectando y cuantas veces ha visitado la página en cuestión. Esto estaba pensado para realizar un histórico de las veces que ha iniciado sesión con la fecha, la IP desde donde lo hubiera hecho, se dejará para posibles ampliaciones. Por último en la parte superior derecha contendrá un enlace para terminar la sesión (recuadro **naranja**).

Crear Manager

A la izquierda de la pantalla el Administrador podrá ir viendo el avance por los diferentes puntos por los que se encuentra, aquellos que tienen un fondo amarillo son donde está ahora, los que se encuentran tachados son las acciones que ya ha realizado.

A la derecha está el formulario para registrar el futuro Manager.

Para registrar los managers, el Administrador deberá escribir el nombre de este, el email, el idioma con que se quiere que le lleguen los mensajes y la contraseña tendrá una por defecto que deberá ser cambiada.

Una vez enviado los datos para crear/almacenar el nuevo Manager, se le enviará un correo al a la cuenta del Manager que se utilizo en el registro para que este sepa que los datos de acceso a la aplicación, el siguiente paso sería el de crear un proyecto para ese Manager.

Crear Proyecto

El segundo paso es la creación del proyecto, para el Manager creado anteriormente, se puede observar en el recuadro **rojo**, la acción de "**Registrar Manager**" se encuentra ya tachada por ya se ha realizado, ahora en amarillo se encuentra el "**Registro del Proyecto**".

Figura 9.42. Administrador, Crear Proyecto

Para la creación de los proyectos (recuadro **azul**), el Administrador debería escribir solamente el nombre del proyecto, los demás campos son puestos automáticamente, ya que se recogen

de los datos que el Administrador ha rellenado directamente en el registro del Manager del paso anterior, aunque siempre tendrá la opción de cambiar los valores.

Una vez que se rellena ese campo y se envían los datos para almacenar se le envía un correo al Manager para que se para que proyecto se le ha asignado.

Esta opción de creación de proyecto, se puede dejar sin rellenar y crear solo el Manager, después desde otra ventana como es la de las "**Info de Proyectos**" se puede crear Proyectos para Managers que no tengan.

Info de Proyecto

En esta ventana se encuentra la información de todos los proyectos y todos los Managers que se encuentran almacenados en la aplicación.

Figura 9.43. Administrador, Info de Proyectos

Desde esta sección para los proyecto se puede obtener la información del nombre del proyecto, el Manager que esta al cargo, el email del Manager, el paso en el que se encuentra el proyecto, cuando fue modificado y cuando fue creado el proyecto.Para los Managers se obtiene misma información, como son los nombres, el correo, a que proyecto están destinado y sus fechas de creación y modificación.

Para los dos tipos tanto los Managers, como los Proyectos el Administrador puede actualizar sus datos o eliminarlos mediante el botón habilitado para ello.

Al eliminar el proyecto, se eliminará también el Manager que está a su cargo. Para la actualización se redirigirá al Administrador a otra pantalla para que actualice los datos necesarios.

Figura 9.45. Administrador Actualizar Proyecto

Esta pantalla muestra las opciones para actualizar los diferentes campos del proyecto seleccionado por el Administrador, en ellas puede actualizar los campos del nombre; Manager, email del Manager, fecha o el número del paso donde se encuentra el proyecto.

Figura 9.44. Administrador, Actualizar Manager

Para la actualización de los datos del Manager, el Administrador podrá actualizar los siguientes campos, el nombre del Manager, su correo electrónico y el identificador del proyecto donde es Manager.

Cuenta

Figura 9.46. Administrador, Cuenta

En la sección de "**Cuenta**" el Administrador podrá ver y cambiar sus datos de acceso a la aplicación, así como su login, su email su contraseña y el idioma con el que quiere recibir las notificaciones por correo.

Ayuda

Figura 9.47. Administrador, Ayuda

La sección de ayuda, explica al Administrador cual serán sus posibles tareas dentro de su rol en la aplicación, dicha ayuda sirve para guiar al usuario en alguna tarea que necesite ejecutar dentro de nuestra aplicación. Es una pequeña página como un tutorial o documento de ayuda explicativo de sus funciones dentro de la aplicación.

9.3.1.2 Manager

La página contendrá un menú de navegación (recuadro **rosa**) por el cual el Manager podrá desplazarse por las demás páginas.

Tendrá un cuerpo donde se realizarán las acciones o/y se visualizarán los resultados de las acciones que el Manager este realizando.

Tendrá un pie (recuadro **verde**) donde se le indicará desde que IP se está conectando y cuantas veces ha visitado la página en cuestión. Esto estaba pensado para realizar un histórico de las veces que ha iniciado sesión con la fecha, la IP desde donde lo hubiera hecho, se dejará para posibles ampliaciones.

Por último en la parte superior derecha contendrá un enlace para terminar la sesión (recuadro **naranja**).

Figura 9.48. Manager, Crear Miembro

Crear Miembro

Para crear un nuevo miembro del proyecto, el Manager encargado al loguearse en la página accede directamente al registro de del miembro, tercer paso de la creación del proyecto de gestión de riesgos.

Como se observa en el recuadro **rojo** las acciones de registro del Manager y del proyecto se encuentran tachadas lo que quiere decir que se han completado anteriormente.

Para crear el nuevo miembro del proyecto, se debe completar los campos que se encuentran en el recuadro **azul**, el identificador del proyecto ya se encuentra automáticamente puesto porque es el mismo proyecto del Manager, este deberá cubrir el nombre, el correo electrónico y el idioma, la contraseña por defecto será puesta automáticamente en "*cambiame*".

Info Proyecto

El Manager como el Administrador tiene acceso para que se le muestre los proyectos a los que está destinado, así como los miembros que tiene ese proyecto en concreto, a diferencia del Administrador, el Manager no podrá modificar ni eliminar ni el proyecto ni los diferentes usuarios que lo integran.

Figura 9.49. Manager, Info Proyecto

Datos de Acceso

Figura 9.50. Manager, Cuenta

En la sección de "**Cuenta**" del Miembro podrá ver y cambiar sus datos de acceso a la aplicación, así como su login, su email su contraseña y el idioma con el que quiere recibir las notificaciones por correo.

Ayuda

Figura 9.51. Manager, Ayuda

La sección de ayuda, explica al usuario Manager cual serán sus posibles tareas dentro de su rol, en la aplicación, dicha ayuda sirve para guiar al usuario en alguna tarea que necesite ejecutar dentro de nuestra aplicación. Es una pequeña página como un tutorial o documento de ayuda explicativo de sus funciones dentro de la aplicación.

Creación de Gestión de Riesgos Informáticos

El Manager junto a los demás miembros del proyecto son los que realizarán las diferentes tareas para la realización de la Gestión de riesgos del Proyecto Informático, es el cuarto paso y en el manager se extiende un menú deslizante donde se observan las acciones que podrá desarrollar una vez ingresado en dicha tarea, en la imagen se corresponde con el recuadro **rojo**. Se amplía más adelante en el apartado de Creación Gestión de Riesgos.

Figura 9.52. Manager, Creación de Gestión de Riesgos

9.3.1.3 Miembro

Figura 9.53. Miembro, Pantalla Inicial

La página contendrá un menú de navegación (recuadro **rosa**) por el cual el Miembro podrá desplazarse por las demás páginas.

Tendrá un cuerpo donde se realizarán las acciones o/y se visualizarán los resultados de las acciones que el Miembro este realizando, en el recuadro **rojo** se observan las diferentes acciones dentro de la Creación de la Gestión de Riesgos, mientras que en el cuadro **azul** en este caso aparecen los diferentes tareas para completar la creación de las primeras.

Tendrá un pie (recuadro **verde**) donde se le indicará desde que IP se está conectando y cuantas veces ha visitado la página en cuestión. Esto estaba pensado para realizar un histórico de las veces que ha iniciado sesión con la fecha, la IP desde donde lo hubiera hecho, se dejará para posibles ampliaciones.

Por último en la parte superior derecha contendrá un enlace para terminar la sesión (recuadro **naranja**).

Datos de Acceso

Figura 9.54. Miembro, Cuenta

En la sección de "**Cuenta**" del Miembro podrá ver y cambiar sus datos de acceso a la aplicación, así como su login, su email su contraseña y el idioma con el que quiere recibir las notificaciones por correo.

Ayuda

Figura 9.55. Miembro, Ayuda

La sección de ayuda, explica al usuario Miembro cual serán sus posibles tareas dentro de su rol, en la aplicación, dicha ayuda sirve para guiar al usuario en alguna tarea que necesite ejecutar dentro de nuestra aplicación. Es una pequeña página como un tutorial o documento de ayuda explicativo de sus funciones dentro de la aplicación.

9.3.1.4 Crear Gestión de Riesgos

En la **Creación de la Gestión de Riesgos**, los usuarios de tipo Manager como de tipo Miembro, son los únicos que tienen acceso y pueden realizar dichas tareas, que como se menciono anteriormente que son:

- *Plan de Gestión de Riesgos*
- *Identificación/Análisis de Riesgos*
- *Respuesta a los Riesgos*
- *Informes*

Plan de Gestión de Riesgos

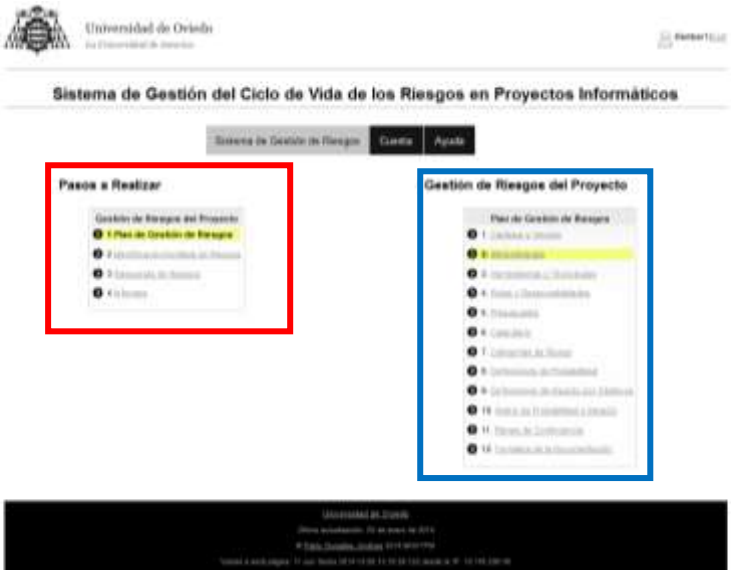


Figura 9.56. Plan de Gestión de Riesgos

En el **Plan de Gestión de Riesgos,** es la primera acción de la Gestión, como se aprecia en el recuadro **rojo**, a la derecha en el recuadro **azul** salen los diferentes campos que se deben rellenar para completar esta tarea.

El primero de ellos, "**Cambios y Versión**", se utiliza para ver los diferentes cambios que se hayan podido realizar sobre la versión de ese Plan de Riesgo, en el también se determina que versión se está utilizando, y si hubiera otras también aparecerían pudiendo cambiar de versión hacia delante o hacia detrás.

El resto de enlaces le llevan a nuevas páginas donde ingresar los datos para esa descripción, alguno de esos datos son requeridos secuencialmente después en el siguiente punto, por lo que se sugiere que se siga el orden según están escritos, aunque se puede realizar en el orden que uno prefiera.

Se mostrará a continuación la imagen de un ejemplo de varias versiones, donde el usuario será capaz de ver los cambios realizados en esa versión y las otras versiones antiguas que se hayan almacenado.

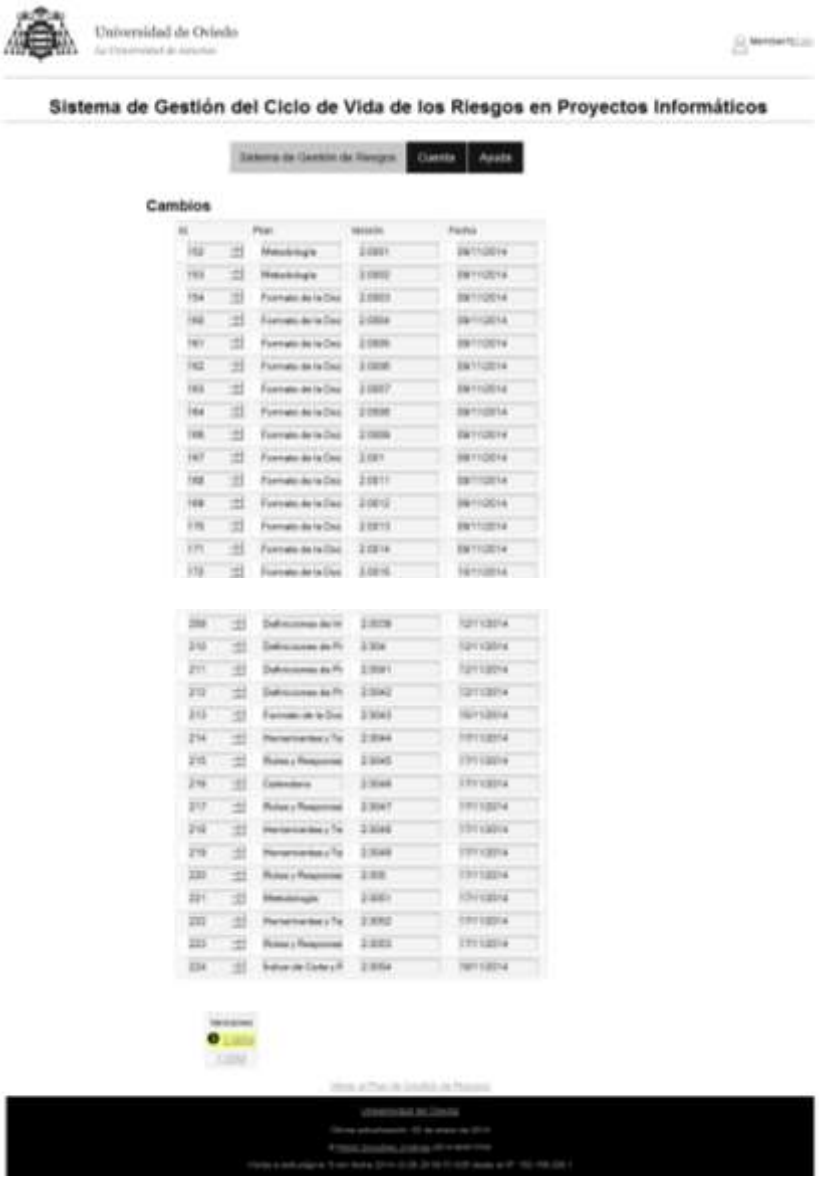


Figura 9.57. Ejemplo de Versiones y Cambio

El Plan de Gestión de Riesgos describe la manera en que se estructurará y realizará la gestión de riesgos en el proyecto.

Los diferentes campos que contiene el Plan de Riesgos es donde se deben introducir las diferentes informaciones en sus campos, un ejemplo sería la **Metodología** que define los métodos, las herramientas y las fuentes de información que pueden utilizarse para realizar la gestión de riesgos en el proyecto

Figura 9.58. Plan de Gestión de Riesgos, Metodología

Tenemos además:

Herramientas y Tecnologías: Se utilizarán para la gestión de riesgos, al menos una de ellas para cada apartado de la metodología. Entre las cuales se encuentran, revisiones de la documentación, matriz de probabilidad e impacto**,** juicio de expertos, reuniones sobre el estado del proyecto.

Roles y responsabilidades. Define al líder, el apoyo y a los miembros del equipo de gestión de riesgos para cada tipo de actividad del plan de gestión de riesgos, y explica sus responsabilidades.

Presupuesto. Asigna recursos, estima los fondos necesarios para la gestión de riesgos, a fin de incluirlos en la línea base del desempeño de costos y establece los protocolos para la aplicación de la reserva para contingencias.

Calendario. Define cuándo y con qué frecuencia se realizará el proceso de gestión de riesgos a lo largo del ciclo de vida del proyecto, establece los protocolos para la utilización de las reservas para contingencias del cronograma y prevé las actividades de gestión de riesgos que deben incluirse en el cronograma del proyecto.

Categorías de riesgo. Proporciona una estructura que asegura un proceso completo de identificación sistemática de los riesgos con un nivel de detalle coherente, y contribuye a la efectividad y calidad del proceso Identificar los Riesgos.

En las **Definiciones de Probabilidad,** se obtienen unas definiciones las cuales van con una determinada probabilidad de que suceda, dicha probabilidad es un impacto que se puede contabilizar numéricamente dependiendo del tipo de si es Muy Alta o Muy Baja y sus termino medios.

Figura 9.59. Plan de Gestión de Riesgos, Definiciones de Probabilidad

Mientras en la **definición de impactos por objetivo**, se tienen varios objetivos como pueden ser el coste, el tiempo, el alcance y la calidad, a estos objetivos se les asigna un impacto dependiendo de cuanto se aumente o incrementen esos objetivos, para luego realizar con tanto las probabilidades como con los impactos una matriz, la matriz de probabilidades e impacto.

Figura 9.60. Plan de Gestión de Riesgos, Definiciones de Impacto por Objetivos

Una vez evaluada su probabilidad e impacto en el proyecto, se procede a calcular su importancia mediante una matriz de probabilidad e impacto de manera que el riesgo se priorice para posteriores estudios y evaluaciones o directamente para ser monitorizado en el proyecto.

Juntas, la definición de las escalas de impacto y la matriz de probabilidad e impacto, definirán cuáles son los puntos de corte para decidir que un riesgo debe ser tenido en cuenta, desestimado o ser objetivo de una supervisión durante el desarrollo. Al mismo tiempo definen también cuáles son estos valores para los objetivos principales del proyecto.

Figura 9.61. Plan de Gestión de Riesgos, Matriz de Probabilidad e Impacto

Identificación/Análisis de Riesgos

Una vez realizado el Plan de Riesgos, se debe realizar la Identificación de Riesgos del proyecto siguiendo una plantilla adaptada al **PMBOK**. Se han agregado algunos ejemplos de posibles riesgos que se pueden producir, para que el usuario pueda adaptarlo a sus necesidades poniendo o quitando los que considere necesarios:

Figura 9.62. Identificación/Análisis de Riesgos, Lista de Posibles Riesgos

La Lista de riesgos identificados, para estar aprobada debe contener los siguientes campos (mínimo):

- Identificación /nombre del riesgo
- Descripción
- Responsable
- Probabilidad
- Impacto en cada uno de los objetivos definidos
- Impacto global sobre el proyecto (Score)
- Nota de corte, color rojo sobrepasa el corte riesgo de estudio, color verde por debajo del corte no es un riesgo prioritario
- Estrategia general usada (Response), esto es Evitar, Aceptar, Mitigar o Transferir. Si los riesgos son positivos, entonces serían Explotar, Compartir, Mejorar y Aceptar

Figura 9.63. Identificación/Análisis de Riesgos

Respuesta a los Riesgos

Figura 9.64. Respuesta a los Riesgos

La Planificación de la gestión de cada riesgo genera un plan para tratar cada riesgo de importancia alta del proyecto. Ayuda, por tanto, a manejar cada elemento de riesgo, incluyendo la coordinación de los planes individuales de elementos de riesgos entre ellos y con respecto al plan general de riesgos.

De toda la lista de riesgos identificados y priorizados se seleccionan los riesgos de mayor impacto global y se documentan, siguiendo una plantilla, donde se deberá rellenar los diferentes campos para completar la respuesta al riesgo.

Figura 9.65. Respuesta al Riesgo

Informes

Una vez completado los diferentes pasos anteriores, se crean a partir de esos datos informes referentes a cada uno de las acciones creadas anteriormente por medio de la generación de informes de PDF.

Figura 9.66. Informes de la Gestión de Riesgos

En el PMBOK hay más técnicas cuantitativas, pero es complicado realizarlas sobre los riesgos porque depende en gran medida del tipo de riesgo y eso supera un poco la intención de este proyecto.

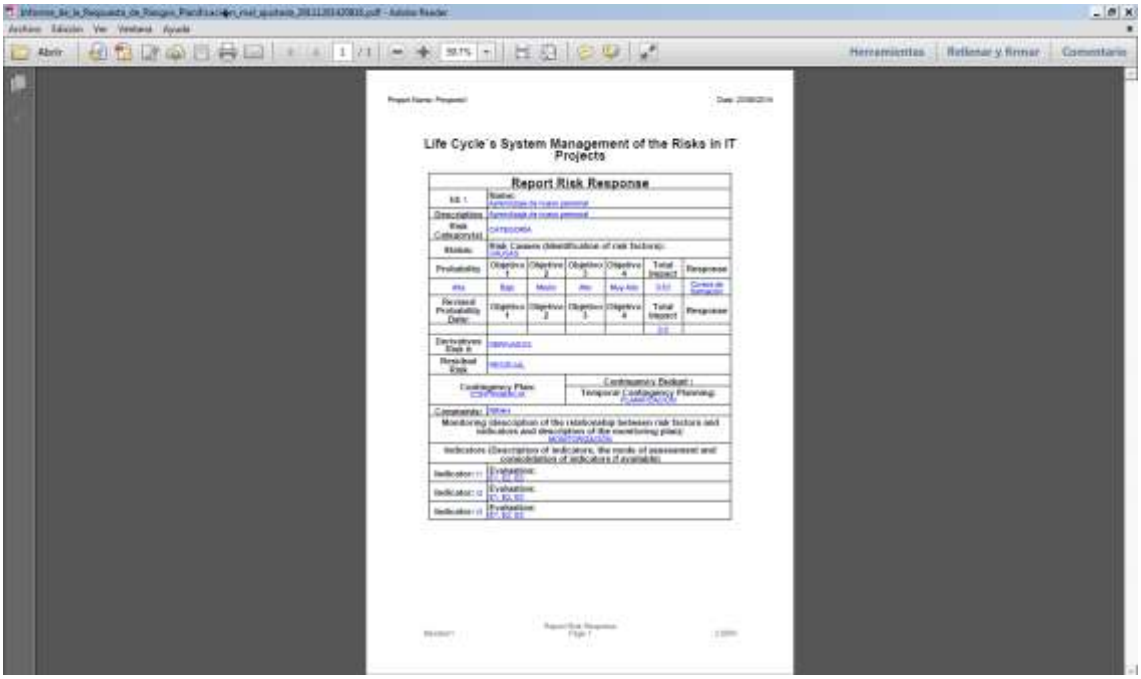


Figura 9.67. Ejemplo de Informe a la Respuesta de Riesgos

Capítulo 10. Conclusiones y Ampliaciones

10.1 Conclusiones

Se ha elaborado finalmente un sistema que en principio es una aplicación web destinada al estudio de los riesgos en proyectos de informática. Que comprende desde la creación de los proyecto y de los manager de los proyecto por parte de un Administrador, la creación de Miembros por parte de ese Manager, donde los Manager y los Miembros pueden confeccionar un plan de riesgos inicial, donde posteriormente se identifican y se analizan los riesgos, una vez identificados y analizados se les da una respuesta y se concluye con la creación de unos informes para cada etapa. Con estos informes se pretende cumplir con los objetivos de aumentar la probabilidad y el impacto de eventos positivos y disminuir la probabilidad y el impacto de eventos negativos para dichos proyectos.

Las tecnologías usadas para la realización del proyecto se ha intentado utilizar las máximas posibles que se han cursado durante el Máster, intentando aplicar cada una de ellas para cada término que se ha necesitado, algunas ocasiones como pueden ser temas de seguridad o bases de datos por tiempo o por conocimiento no se han desarrollado todo lo que uno quisiera.

Con respecto a otros proyectos realizados, este se encuentra bastante bien estructurado gracias en parte a los conocimientos adquiridos en el Máster sobre la utilización de modelos de arquitectura, como puede ser MVC, Struts2 y Spring, la utilizando para la presentación de los datos y recogida de los mismo por medio de JSPs, la organización y el transporte de los datos pasando por las diferentes capas que se han utilizado ayudado en parte por patrones de diseño, yo espero que el a ver trabajado con estas nuevas tecnologías nuevas para mí, me abra nueva expectativas en el campo laboral, donde al menos tendré algunos grados de conocimiento que antes no tenía.

Es una pena que en el proyecto real donde ahora estoy trabajando no se esté utilizando nada de lo anteriormente mencionado, ya que seguramente podría decir claramente que el hacer este tipo de proyecto me hubiera reportado una gran ayuda a desarrollar mejor o más rápidamente mi labor dentro de un proyecto real.

Aunque estoy seguro de que aún me queda mucho por descubrir no solo de la tecnología usada recientemente, sino de la más simple realizada anteriormente, espero y deseo que estos nuevos conocimientos adquiridos me sirvan para tener una buena base para mis futuros trabajos en la Ingeniería tanto informática como de la Web.

10.2 Ampliaciones

Cualquier labor de ampliación que tengamos contemplada en el sistema debe ser descrita aquí, mencionando en qué consiste, cómo ampliará el sistema, qué ventajas nos aporta y porqué no se ha incluido en el sistema diseñado, entre otros aspectos.

10.2.1 Seguridad Encriptada Mejorada

Mejorar el tratamiento de los datos sensibles que los usuarios insertan en la aplicación, con la utilización de certificados de terceras empresas o la utilización de llaves públicas y privadas, las ventajas serían una mayor seguridad a la hora de que nadie espié los datos a través de la red y se puedan hacer con información personal de los usuarios de la aplicación. Un sistema de tales características sería ya en si un proyecto a presentar.

10.2.1.1 Mejora de Recuperación de Contraseña

El sistema actual de recuperación de contraseñas, no es todo lo seguro que debería ser ya que la contraseña viaje por la red sin SSLy cualquiera podría hacerse con ella, el sistema de cifrado en BBDD que si existe, se podría cambiar por un sistema que aunque se pierda la información de la contraseña, no se pueda volver a descifrar una vez cifrada, haciendo que cuando se intente recuperar dicha contraseña se remplazada por una nueva que machaque la actual. No se ha incluido por la misma razón que el punto anterior con respecto al protocolo HTTPS, con respecto al modo de encriptación se eligió desde el principio utilizar las funciones de encriptación propias de MySQL.

10.2.2 Comunicación

Una de las mejoras que podría darse en la incorporación al código es de nuevas formas de comunicación, como podría ser un sistema de Messenger interno, o agregar más sistemas de comunicación por medio de las redes sociales como Facebook, WhatsApp, etc. Esto dará una mayor amplitud a la hora de comunicar nuevas noticias a los miembros del proyecto, al disponer de un mayor número de sistemas. No se ha incluido todas las opciones al sobrepasar el objetivo del proyecto.

10.2.3 Internacionalización

La página sí que tiene todo lo necesario para presentarla en tantos idiomas como *properties* de dichos idiomas tenga, pero la elección de mostrar uno u otro se la ha dejado a elección del navegador, que dependiendo del idioma de este lo muestra en un idioma u otro, una mejora sería tener un acceso más rápido al cambio de idioma introduciendo para ello un botón en la propia aplicación para el cambio más rápido y efectivo sin tener que ir a la configuración del propio navegador. Una vez terminada dicha función, se dejo que el navegador fuera el

responsable del cambio de idioma, después con el tiempo avanzado siempre ocurren mejoras que por falta de eso mismo no se pueden llevar a cabo.

10.2.4 Utilización de Applets

Alguna funcionas pensadas para el proyecto como la elección de la versión del Plan de Riesgo o la elección de directorio donde guardar los PDFs, podría utilizar algún tipo de Applet para realizar dichas funciones, el problema por el cual no se acabaron creando fue que la ejecución de los Applets requiere el uso de la JVM. Con los problemas diversos que esto conlleva por no tener una máquina virtual actualizada, el Applet intentará descargar la última versión si puede, esto requerirá del usuario, tanto permisos que puede que no tenga, como su dedicación, lo cual aumentara el tiempo en que tarda en realizar la acción.

10.2.4.1 Visor Previo PDF

Como la aplicación crea PDFs, por medio de los datos que se encuentran almacenados, se podría insertar una especie de visor de los PDF, sin tener que ir hasta la dirección donde se almacenan los PDFs creados, así la aplicación lo haría por el usuario y le ahorraría tiempo, se encontraron diversos visores, pero requería de ciertas características o permisos que pueden que no todos los usuarios quieran o tengan, así que se prefirió solo de momento descargar el documento en el disco local del usuario.

10.2.5 Almacenamiento de Estadísticas

Se dejo casi listo todo en la versión que existe, para poder tener una especie de historial de conexiones, donde aparezca, tanto las IPs del usuario, como la fecha o el navegador desde que se conecto y las veces que ha visitado cierta página, todo esto por falta de tiempo se ha dejado en el aire para futuras ampliaciones.

10.2.6 Guardado Automático

Una cuestión que me gustaría ampliar sería el tema de una especie de guardado automático de las secciones que el usuario este insertando, como una copia de seguridad de lo que lleva haciendo desde que entro en sesión, algo similar a lo que hacen programas como el procesador de texto de Microsoft Word, el cual pasado un cierto tiempo almacena de forma automática los que lleva escrito. No se ha llevado a cabo pro que no entraba dentro de los objetivos del proyecto.

10.2.7 Accesibilidad y Usabilidad

Tener todas las páginas accesibles en el grado máximo, pudiendo llegar hasta el nivel de conformidad "Triple A" donde se cumplan todos los puntos de las pautas del Contenido Web 2.0 (WCAG 2.0) hasta la prioridad tres. Ser usable para todos los tipos de usuarios, los cuales logren una satisfacción con el uso de la página. La página está usando un nivel "Doble AA", que es el mínimo pedido.

10.2.8 Persistencia

Se ha utilizado el patrón DAO, para el modelo de acceso a los datos, se ha externalizado en un porperties los datos de acceso, pero me hubiera gustado, utilizar otro sistema como por ejemplo, *Ibatis* o *Hibernate* como framework para la persistencia, entre otras cosas para mejorar cuestiones de externacionalización de *queries* o la creación automática de tablas según necesidades, por falta de tiempo no se ha podido profundizar en cualquiera de estas u otras opciones más profesionales a la hora de tratar los objetos de persistencia.

Capítulo 11. Presupuesto

11.1 Presupuesto de Costes

Ítem	SubItem	Denominación	Especificación	Cantidad	Precio Unitario	Factor de Amortización	TOTAL
1		**Materiales**					
1.1		**Software**					
	1	*Office 365 Personal*		3	54,51 €	11,70%	**19,13 €**
	2	*Microsoft Office Project 2007*		1	162,83 €	11,70%	**19,05 €**
	3	*Enterprise Architect - Desktop Edition*		1	110,85 €	11,70%	**12,97 €**
	4	*Microsoft Windows 7 Home Premium 32bits OEM Service Pack 1*		6	73,55 €	11,70%	**51,63 €**
	5	*Ubuntu 12.04.4*		6	0,00 €	11,70%	**0,00 €**
	6	*Oracle VM VirtualBox Manager 4.2.18*		6	0,00 €	11,70%	**0,00 €**
	7	*Apache Tomcat 7.0*		6	0,00 €	11,70%	**0,00 €**
	8	*MySQL 5.6*		6	0,00 €	11,70%	**0,00 €**
	9	*MySQL Workbench 5.1 OSS*		1	0,00 €	11,70%	
	11	*Notepad 5.9.8*		6	0,00 €	11,70%	**0,00 €**
	12	*Eclipse*		6	0,00 €	11,70%	**0,00 €**
	13	*Google Chrome Versión 39.0.2171.65 m*		6	0,00 €	11,70%	**0,00 €**
		Mozilla Firefox 34.0.5		6	0,00 €	11,70%	**0,00 €**
	14	*Acrobat Reader XI*		6	0,00 €	11,70%	**0,00 €**
1.2		**Hardware**					

	1	*MacBook Pro de 13 pulgadas*		1	1.112,00 €	11,70%	**130,10 €**
	2	*Ordenador Sobremesa*		6	276,50 €	11,70%	**194,10 €**
		Verbatim STORE N GO USB 32		6	25,50 €	11,70%	**17,90 €**
1.3		**Comunicaciones**					
	1	*Móvil Samsung SIII*		1	354,55 €	11,70%	**41,48 €**
	2	*Tarifa Móvil*		5	20,00 €		**100,00 €**
1.4		**Formación**		10	35,00 €		**350,00 €**
1.5		**Mobiliario para el equipamiento informático**					
	1	*Escritorio*		6	54,51 €	11,70%	**38,27 €**
	2	*Silla*		6	70,31 €	11,70%	**49,36 €**
	3	*Lámpara*		6	23,69 €	11,70%	**16,63 €**
2		**Desarrollo de la Aplicación**					
	1	*Estudio Inicial*		9,5	25,00 €		**237,50 €**
	2	*Análisis*		34,75	40,00 €		**1.390,00 €**
	3	*Diseño*		23,25	32,00 €		**744,00 €**
	4	*Implementación*		22	26,00 €		**572,00 €**
	5	*Pruebas*		18	25,00 €		**450,00 €**
	6	*Instalación*		5	24,00 €		**120,00 €**
	7	*Documentación*		47,38	23,00 €		**1.089,74 €**
3		**Otros**					
	1	*Dietas*		447	5,00 €		**2.235,00 €**
	2	*Desplazamiento*		149	5,00 €		**745,00 €**

	Subtotal	**8.623,87 €**
	Beneficio (21,00%)	**1.811,01 €**
	Total sin IVA	**10.434,88 €**
	IVA (21,00%)	**12.626,21 €**

Figura 11.1. Presupuesto de Costes

Se utiliza un factor de amortización para calcular el valor que se le ha de cobrar al cliente por los dispositivos utilizados durante la realización del proyecto. Este factor de amortización se calcula en función de la vida útil de un dispositivo, calculada sobre 4 años y la duración total del proyecto.

Variables de entrada

- Duración del proyecto en días son: 149
- 4 años en días: 1273

(Quitando 4 semanas de vacaciones y 17 días festivos por año)

Cálculo del factor de amortización

$$\frac{149}{1273} = 0.117$$

Se halla el porcentaje de ese factor

$$0.117 \times 100 = 11{,}7\ \%$$

Variable de salida

- Porcentaje del precio del dispositivo: 11,7 %

11.2 Presupuesto del Cliente

Ítem	SubItem	Concepto	Cantidad	Precio Unitario	Factor de Amortización	TOTAL
1		**Materiales**				
		Hardware				**342,11 €**
		Software				**102,79 €**
		Comunicaciones				**141,48 €**
		Formación				**350,00 €**
		Mobiliario				**104,25 €**
2		**Desarrollo de la Aplicación**				
		Estudio Inicial				**237,50 €**
		Análisis				**1.390,00 €**
		Diseño				**744,00 €**
		Implementación				**572,00 €**
		Pruebas				**450,00 €**
		Instalación				**120,00 €**
		Documentación				**1.089,74 €**
3		**Otros**				**2.980,00 €**
					Subtotal	**8.623,87 €**
					Beneficio (21,00%)	**1.811,01 €**
					Total sin IVA	**10.434,88 €**
I.V.A. es un 21%					*Total con IVA*	**12.626,21 €**

Figura 11.2. Presupuesto del Cliente

Los **Materiales** necesarios para la realización del proyecto, serán todos aquellos que se necesiten para la elaboración del proyecto, tanto del tipo de **Hardware**, que se utilizará para el desarrollo como para la puesta en producción del servicio, utilizando para ello varios equipos de trabajo, así como material de almacenamiento externo tipo USB de memoria para la realización de Backups, todo ello necesita también de su pertinente **mobiliario** el cual será el mínimo para la realización del proyecto. Cada equipo estará provisto del material necesario de **Software**, para hacer funcionar los diferentes programas que se necesiten para el proyecto, habrá Software que no se necesite pagar licencias por el, y será un objetivo del proyecto el que se consiga la mayoría de Software a coste cero.

Se ha provisto también de la compra de equipo de **comunicaciones**, terminal y tarifa de telefonía, para poder contactar con los proveedores de los servicios secundarios y con los proveedores de los materiales así como para mantener conversaciones con el cliente.

Se llevará a cabo también unos cursos de **formación** dentro del ítem de materiales, para las personas que pudieran utilizar el servicio, para una explicación de la forma de funcionamiento práctico de la aplicación web.

Para el **desarrollo del sistema**, este representa el trabajo humano dedicado al proyecto, el cual integra las diferentes tareas como son la de investigación, con un estudio inicial del problema planteado, así como una posible solución, el análisis, el diseño, la implementación, las pruebas de testing, o la instalación de los diferentes programas, todas estas tareas se encuentran detalladas en la documentación del proyecto, donde se detallan todas estas tareas.

Otros Gastos: Gastos derivados durante el proyecto de dietas y transporte de los trabajadores, no siempre estarán todos, como mínimo habrá tres de ellos, entre otro.

Capítulo 12. Referencias Bibliográficas

Libros y artículos usados de alguna forma durante el desarrollo del proyecto o su documentación.

12.1 Libros y Artículos

[Abad10] Abad, Francisco; Cerdá. "*Consejos para el desarrollo de tu proyecto final de carrera*". 2010.

[Fernández14] Daniel Fernández Lanvin. "*Transparencias: Arquitecturas y Diseño de Sitios Web*". Universidad de Oviedo. 2014

[Gálvez-García06] Gálvez Sergio; García Ignacio. "*Java A Tope: Javamail En Ejemplos*". Universidad de Málaga. 2006. ISBN: 84-690-0697-5.

[Hassan08] Hassan Montero, Y. "Guía de Evaluación Heurística de Sitios Web". http://www.nosolousabilidad.com/articulos/heuristica.htm

[Juan- López14] Aquilino Adolfo Juan Fuente, Benjamín López Pérez. "*Dirección y Planificación de Proyectos Informáticos*", Universidad de Oviedo, 2014.

[Lowagie14] Bruno Lowagie. "*The ABC of PDF with iText*", iText Software, 2014.

[PMBOK08]Project Management Institute. "*Guía de los Fundamentos para la Dirección de Proyectos (Guía del PMBOK®)—Cuarta edición*". Project Management Institute**.** 4º. 2008. ISBN: 978-1-933890-72-2.

[PMBOK13]Project Management Institute. "*A Guide to the Project Management Body of Knowledge (PMBOK® Guide)—Fifth Edition*". Project Management Institute**.** 5th. 2013. ISBN: 978-1-935589-67-9.

[Redondo07] Redondo L.; J. Manuel. "*Ejemplo para la plantilla de PFC*". Universidad de Oviedo. 2007.

[Siles-Mondelo12] Rodolfo Siles, PMP y Ernesto Mondelo, PMP. "*Guía de Gestión de Proyectos para Resultados PM4R*", BID-INDES, 2012.

12.2 Referencias en Internet

Páginas Web consultadas para cualquier aspecto relacionado con el desarrollo del sistema o su documentación.

Accesibilidad

- www.w3.org/WAI/eval
- *TAW* (http://www.tawdis.net/taw3/cms/es)
- *HERA* (http://www.sidar.org/hera/index.php.en)
- http://www.w3.org/WAI/WCAG20/from10/comparison/
- http://www.w3.org/TR/2006/WD-WCAG20-20060427/appendixB.html

BBDD

- **MySQL**
  - https://help.ubuntu.com/12.04/serverguide/mysql.html
  - http://www.e-ghost.deusto.es/docs/TutorialMySQL.html
  - http://sql-info.de/mysql/mysql.html
  - http://www.w3schools.com/sql/sql_foreignkey.asp
  - http://www.w3schools.com/sql/sql_drop.asp
  - http://mysql.conclase.net/curso/index.php
  - http://desarrollandowebsdinamicas.blogspot.com.es/2014/02/agregar-un-campo-una-tabla-mysql.html
  - http://misnotaslinux.blogspot.com.es/2012/03/mysql-crear-clave-principal-con-dos.html
  - http://www.mysqltutorial.org/mysql-timestamp.aspx
  - http://www.thegeekstuff.com/2008/09/backup-and-restore-mysql-database-using-mysqldump/
- **DAO**
  - http://www.proactiva-calidad.com/java/patrones/DAO.html
  - http://www.programacion.com/articulo/catalogo_de_patrones_de_diseno_j2ee__y_ii:_capas_de_negocio_y_de_integracion_243/8
  - http://chuwiki.chuidiang.org/index.php?title=Patr%C3%B3n_DAO

Casos de Uso

- http://en.wikipedia.org/wiki/Use_case
- http://www.gatherspace.com/static/use_case_example.html
- http://www.visualcase.com/tutorials/use-case-diagram.htm
- http://www.wilsonmar.com/1usecase.htm
- http://en.wikipedia.org/wiki/Scenario_(computing)

Diagramas UML

- http://www.agilemodeling.com/artifacts/useCaseDiagram.htm
- http://desarrollandoconocimientoya.blogspot.com.es/p/uml-con-enterprise-architect.html
- http://www.visualcase.com/kbase/use_case_sample.htm
- www.uml.org

- http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=EnterpriseArchitectUML2.x
- http://es.tldp.org/Tutoriales/doc-modelado-sistemas-UML/doc-modelado-sistemas-uml.pdf

Navegabilidad

- www.agilemodeling.com/artifacts/uiFlowDiagram.htm

Diagramas de Paquetes

- http://www.agilemodeling.com/artifacts/packageDiagram.htm
- http://www.sparxsystems.com.au/resources/uml2_tutorial/uml2_packagediagram.html

Diagramas de Despliegue y Componentes

- http://www.agilemodeling.com/artifacts/deploymentDiagram.htm
- http://www.visual-paradigm.com/VPGallery/diagrams/Deployment.html
- http://www.agilemodeling.com/artifacts/componentDiagram.htm
- http://www.visual-paradigm.com/VPGallery/diagrams/Component.html
- http://www.ibm.com/developerworks/rational/library/dec04/bell/

Diagramas de Secuencia

- http://pst.web.cern.ch/PST/HandBookWorkBook/Handbook/SoftwareEngineering/UCDOM_interaction.html

Diagramas de Actividad

- http://dn.codegear.com/article/31863#activity-diagrams
- http://www.agilemodeling.com/artifacts/activityDiagram.htm

Facade

- https://danielggarcia.wordpress.com/2014/03/03/patrones-estructurales-ii-patron-facade/
- http://www.slideshare.net/Anamilena916/patron-fachada-5101584

HTML

- **Tablas**
  - https://developer.mozilla.org/es/docs/Trazado_de_una_tabla_HTML_mediante_JavaScript_y_la_Interface_DOM
  - http://www.forosdelweb.com/f4/problema-tabla-dinamica-1030075/
  - https://hector2c.wordpress.com/2009/07/13/jquery-agregar-y-eliminar-filas-en-una-tabla/
  - http://www.forosdelweb.com/f179/obtener-texto-td-con-jquery-1035146/
  - http://www.w3schools.com/tags/tag_input.asp
- http://www.w3schools.com/jsref/dom_obj_event.asp
- http://www.w3schools.com/jsref/dom_obj_document.asp

JavaDoc

- http://www.oracle.com/technetwork/articles/java/index-137868.html
- http://stackoverflow.com/questions/1319489/how-to-generate-javadoc-documentation-with-umlauts

JMeter

- https://confluence.sakaiproject.org/display/DOC/JMeter+Performance+Testing
- http://jmeter.apache.org/usermanual/build-db-test-plan.html

JUnit

- http://www.vogella.de/articles/JUnit/article.html#unittesting

JSPs

- http://marwanalephn.blogspot.com.es/2009/10/la-creacion-de-sitios-web-dinamicos-en.html
- http://java.ciberaula.com/articulo/introduccion_jstl

OpenShift

- https://developers.openshift.com/en/managing-client-tools.html
- https://developers.openshift.com/en/getting-started-debian-ubuntu.html

PDF

- http://soloinformaticayalgomas.blogspot.com.es/2010/12/generar-un-documento-pdf-desde-java.html
- https://hashblogeando.wordpress.com/2013/07/14/itext-generacion-de-archivo-pdf-en-java/
- http://www.mkyong.com/java/how-to-construct-a-file-path-in-java/
- http://itextpdf.com/book/toc.php

Situación Actual

- http://www.palisade.com/risk/
- http://www.habilissoftware.com/pmpms.html
- http://www.gxsgsi.es/
- http://www.r-box.com.ar/analisis-de-riesgo/

Spring

- http://projects.spring.io/spring-framework/#quick-start

Struts2

- http://www.mkyong.com/tutorials/struts-2-tutorials/
- http://stackoverflow.com/questions/1206513/how-to-evaluate-struts-2-sif
- http://www.javatutoriales.com/2011/06/struts-2-parte-2-ognl.html
- http://www.javatutoriales.com/2011/10/struts-2-parte-3-trabajo-con.html
- http://www.javatutoriales.com/2011/12/struts-2-parte-4-scopes-de-objetos-web.html
- http://www.javatutoriales.com/2012/04/struts-2-parte-5-tipos-de-results.html
- http://www.javatutoriales.com/2012/06/struts-2-parte-6-interceptores.html
- http://www.javatutoriales.com/2013/11/struts-2-parte-7-tags.html
- http://struts.apache.org/docs/guides.html
- http://www.journaldev.com/2237/struts-2-control-tags-example-tutorial
- **Response Download PDF**
  - http://www.mkyong.com/struts2/how-to-get-the-httpservletresponse-in-struts-2/
  - http://itextpdf.com/examples/iia.php?id=173
  - http://stackoverflow.com/questions/3592058/how-to-send-byte-as-pdf-to-browser-in-java-web-application
  - http://stackoverflow.com/questions/11433228/read-pdf-file-and-offer-it-as-download-with-itext

Capítulo 13. Apéndices

13.1 Glosario y Diccionario de Datos

- **Amenaza:** Una amenaza es la posibilidad de que se produzca una determinada vulnerabilidad de forma satisfactoria. Una fuente de amenazas no plantea un riesgo cuando no hay vulnerabilidades que puedan ser 'activadas'.
- **Análisis de riesgos:** Proceso de evaluar riesgos ya identificados para estimar su impacto y probabilidad de ocurrencia.
- **Análisis cualitativo de riesgo:** Evaluación del impacto y la probabilidad de ocurrencia de los riesgos sobre las salidas del proyecto utilizando métodos cualitativos.
- **Análisis cuantitativo de riesgo:** Evaluación matemática de la probabilidad de ocurrencia de cada riesgo y sus consecuencias en las salidas del proyecto.
- **Framework:** Un framework es una estructura en capas que indica qué tipo de programas pueden o deben ser construidos y cómo se interrelacionan. Un framework es generalmente más amplio que un protocolo y más prescriptivo que una estructura.
- **Gestión de riesgos:** aplicación de procedimientos y prácticas en relación a los riesgos que amenazan un proyecto en la organización.
- **GPL**: La GNU GPL (General Public License o licencia pública general) es una licencia que obliga a los licenciatarios a propagar ciertos derechos y libertades en relación al software sobre el que la licencia se aplica. La restricción básica que manda esta licencia es la obligatoriedad de poner a disposición las fuentes de un programa, entendiendo fuentes como la forma en la que es preferible crear modificaciones, a todas aquellas partes que reciban una versión compilada.
- **Impacto:** El impacto es la materialización de un riesgo; una medida del grado de daño o cambio sobre un activo, entendiendo como riesgo la probabilidad de que un evento desfavorable ocurra y que tendría un impacto negativo si se llegase a materializar. Se hará una descripción más detallada acerca de la definición de riesgo en el siguiente apartado.
- **J2EE**: Java 2 Enterprise Edition que es la edición empresarial del paquete Java creada y distribuida por Sun Microsystems. Comprenden un conjunto de especificaciones y funcionalidades orientadas al desarrollo de aplicaciones empresariales.
- **Localhost:** Localhost es el alias que tiene todo ordenador, o router o dispositivo que disponga de una tarjeta de red Ethernet para referirse a sí mismo, el nombre localhost es a su vez un alias de la ip 127.0.0.1 que viene por defecto. Aunque 127.0.0.1 es la dirección lo más comúnmente posible utilizada para el localhost, cualquier dirección IP en la gama 127.*.*.* funciona de manera semejante. También llamada ip de loopback. Esa ip representa a nuestra tarjeta de red.
- **Monitorización y control de riesgos:** monitorizar los riesgos residuales, identificar nuevos riesgos, ejecutar los planes de respuesta de riesgos y evaluar su efectividad a través del ciclo de vida del proyecto.

- **MVC:** El (MVC) es un patrón de arquitectura de software que separa tanto los datos como la lógica de negocio de una aplicación en una interfaz de usuario y es el encargado de gestionar los eventos y las comunicaciones.
- **Riesgo:** un evento no certero o condición que, si ocurriese, tendría un efecto positivo o negativo sobre los objetivos del proyecto. Los riesgos negativos pueden llamarse 'amenazas', y los riesgos positivos 'oportunidades'. Normalmente expresado como impacto y probabilidad.
- **Riesgo de un proyecto:** un riesgo de un proyecto es un evento o condición incierto que, si se produce, tendrá un efecto positivo o negativo sobre al menos un objetivo del proyecto, como tiempo, coste, alcance o calidad.
- **Riesgo residual:** un riesgo que permanece después de que las respuestas de riesgos hayan sido implementadas.
- **Plan de contingencia:** procedimientos operativos específicos y preestablecidos de coordinación, alerta y respuesta ante la ocurrencia de un riesgo.
- **Plan de gestión de riesgos:** documento que describe la estrategia que se va a seguir en el proyecto, y cómo las actividades de gestión de riesgos van a ser organizadas y llevadas a cabo durante la vida del proyecto, es decir, a las actividades relacionadas con la reducción, previsión y control de riesgos, la preparación ante riesgos y la recuperación en caso de desastre. El plan de gestión de riesgos es la salida resultante de la fase de planificación de gestión de riesgos.
- **Planificación de gestión de riesgos:** es la fase del proceso de gestión de riesgo en la que se decide el enfoque y se desarrolla el plan que van a seguir las actividades de gestión de riesgos para un compromiso (plan de gestión de riesgos).
- **Plan de respuesta de riesgos:** un documento detallado de todos los riesgos identificados, incluyendo la descripción, causa, probabilidad de ocurrencia, impacto sobre objetivos, respuestas planificadas, propietarios de riesgos y estado actual. El plan de respuesta contiene las acciones para soportar la estrategia de respuestas, en caso de que un riesgo ocurra.
- **PMBOK®:** guía de fundamentos de dirección de proyectos.
- **Spring:** Spring es un framework ligero para el desarrollo de aplicaciones, propone una estructura consistente para toda la aplicación y facilita un método consistente para *pegar* todos los elementos de la aplicación
- **Struts2:** Es un Framework MVC para la creación de aplicaciones web Java elegantes y modernas. Está diseñado para simplificar el ciclo de desarrollo, desde la construcción, pasando por la implementación y el mantenimiento de las aplicaciones a través del tiempo.

13.2 Contenido Entregado

13.2.1 Contenidos

Descripción del contenido (directorios y para qué sirve cada cosa), descripción de esta documentación y de cualquier material que adicionalmente se entregue en la presentación.

13.2.1.1 Estructura General de Directorios

Directorio	**Contenido**
./ Directorio raíz del CD	Contiene un fichero leeme.txt explicando toda esta estructura.
./ Sistema de Gestión del Ciclo de Vida de los Riesgos en Proyectos Informáticos	Contiene toda la estructura de directorios del proyecto para desarrollo.
./documentacion	Contiene toda la documentación asociada al proyecto.
./documentacion/diagramas	Ficheros que genera la herramienta (Enterprise Architect y MySQL Workbench) con la que se han generado los diagramas UML y de entidad relación.
./documentacion/img	Directorio que contiene las imágenes utilizadas en la documentación.
./documentacion/planificación	Directorio que contiene la imagen de la planificación y el archivo .mpp del Microsoft Project.
./documentacion/presupuesto	Directorio que contiene el archivo .xlsx, con el presupuesto del proyecto.
./presentación	Directorio que contiene la presentación en *Powerpoint* utilizada el día de la defensa del proyecto.
./presentación/vídeo	Contiene el vídeo por si OpenShift falla, se puede tener una visión de la aplicación.
./herramientas	Contiene los ficheros de instalación de las herramientas utilizadas para el desarrollo o puesta en marcha del proyecto.
./herram/desarrollo	Ficheros de instalación de las herramientas utilizadas en el desarrollo
./herram/explotacion	BD, servidor Web y herramientas en general.

13.2.1.2 Estructura de Directorios de "desarrollo"

Directorio	Contenido
./ Directorio raíz de "desarrollo"	Contiene los ficheros de proyecto del IDE utilizado.
./build	Directorio donde se guardan los ficheros compilados.
./conf	Contiene los diferentes ficheros de configuración del proyecto.
./dist	Directorio donde se sitúan los ficheros para la distribución del proyecto. Por ejemplo: los ficheros *.war* o *.ear.*
./doc	Contiene toda la documentación relativa al proyecto.
./lib	Bibliotecas externas necesarias para compilar y distribuir, de las que depende este proyecto.
./src	Ficheros fuente
./src/java	Todos los ficheros *Java*, lógicamente agrupados en los paquetes correspondientes.
./src/sql	Este directorio contiene los scripts de *SQL*
./web	Este directorio contiene los ficheros de la *Web*.
./web/css	Contiene las hojas de estilo utilizadas en las jsps
./web/img	Contiene las imágenes utilizadas por los ficheros de la web del proyecto.
./web/jsps	Contiene las páginas jps utilizadas por la web del proyecto.
./web/scripts	Contiene los scripts sutilizados por la web del proyecto.
./test	Directorio base para todos los ficheros utilizados en la automatización del proceso de prueba.

13.2.2 Código Ejecutable e Instalación

13.2.2.1 OpenShift

Insertando la dirección del servidor donde se encuentra alojada el archivo .war, no es necesario nada más. La dirección es la siguiente:

http://masterproject-riskmanagement18.rhcloud.com/

13.2.2.2 phpMyAdmin

Para gestionar la base de datos utilizaremos como ya se comento phpMyAdmin:

https://masterproject-riskmanagement18.rhcloud.com/phpmyadmin/

Datos de acceso:
Root User: adminaAzyjfz
Root Password: FpJAa8r-w1Mt

13.2.2.3 MySQL

Datos de acceso para la Base de Datos:
Conexión URL: mysql://$OPENSHIFT_MYSQL_DB_HOST:$OPENSHIFT_MYSQL_DB_PORT/
Root User: adminaAzyjfz
Root Password: FpJAa8r-w1Mt
Database Name: masterproject

13.2.3 Ficheros de Configuración

Descripción de todos los ficheros necesarios para poder hacer funcionar la aplicación (ficheros de configuración, ficheros de datos, etc.).

Base de Datos

- **db.properties**: Fichero de configuración para la conexión a las diferentes bases de datos ya sea la local o la de la aplicación de OpenShift.

Deployment Descriptor

- **web.xml**: Es el componente de la aplicación J2EE que describe cómo se debe desplegar (o implantar) la aplicación web.

Hojas de Estilo

- **form.css**: Es un archivo donde se encuentra el lenguaje que definen ciertos atributos para dar forma a la presentación de los documentos con formularios escritos en HTML de la aplicación.
- **login.css**: Es un archivo donde se encuentra el lenguaje que definen ciertos atributos para dar forma a la presentación del documento donde se encuentra el login escrito en HTML de la aplicación.
- **style.css**: Es un archivo donde se encuentra el lenguaje que definen ciertos atributos para dar forma a la presentación de los documentos escritos en HTML de la aplicación.
- **table.css**: Es un archivo donde se encuentra el lenguaje que definen ciertos atributos para dar forma a la presentación de los documentos con tablas escritos en HTML de la aplicación.

Internacionalización

- **####_en.properties**: Fichero de configuración para la internacionalización de las diferentes palabras que se encuentran en la aplicación para el idioma Inglés.
- **####_es.properties**: Fichero de configuración para la internacionalización de las diferentes palabras que se encuentran en la aplicación para el idioma Castellano.

JavaScript

- **matrix.js**: Fichero que contiene scripts para cambiar valores de la matriz de impactos, así como los colores de está dependiendo de los valores almacenados.
- **risk.js**: Fichero que contiene scripts, para agregar filas, eliminar filas eliminar tablas, de los planes de riesgo, así como añadir otros datos tanto a los riesgos identificados como a la repuesta a los riesgos.
- **zebra_deploy.js**: Fichero que contienen scripts para el desplazamientos de ciertas listas de menú y para el cambio de color en forma de zebra para las filas de las tablas.

Spring

- **applicationsContext.xml**: Fichero de la aplicación que usa Spring, donde se definen los diferentes Beans, estos interactúan entre sí para proporcionar los servicios de la aplicación. Los Beans que interactúan son también llamados colaboradores, y se pueden definir utilizando contextos de aplicación. Este es el proceso de "cableado" de los Beans de la aplicación.

Struts2

- **struts.xml**: Contiene la información de configuración que se le va modificando a medida que se desarrollan las acciones. Este archivo se puede utilizar para anular la configuración predeterminada para una aplicación.

Validaciones

- **####-validation.xml**: Con estos archivos se realiza la validación de diferentes campos como puede ser la validación de correo electrónico, la validación de rango de números enteros, campo de validación de formularios, validación expresión, la validación de expresiones regulares y etc. El archivo XML debe ser llamado '[acción de clase]' -validation.xml.

Web application **Ar**chive

- **ROOT.war**: Es un archivo JAR utilizado para distribuir las colecciones que tenemos de las JSPs, servlets, clases Java, archivos XML, librerías de tags que constituyen la aplicación web.

13.3 Índice Alfabético

S

T

U

V

X

13.4 Código Fuente

Está sección contendrá el código fuente de la acción para el Bean de un usuario con las diferentes acciones que puede desarrollar este, la estructura es similar para los otros Bean, como son el de proyecto y el de la gestión de riesgos, no se incluirá todo el contenido del mismo, para no saturar al lector.

Se encuentra habilitado para el visionado de todo el código un repositorio donde poder consultar todo el código, desde la dirección:

https://github.com/elpableras/TFM_PROJECT_RISK_MANAGEMENT

13.4.1 Paquete com.miw.model:

13.4.1.1 Fichero "user.java":

```
package com.miw.model;

/**
 * Modelo de datos para los usuarios, tanto para el admin, manager, como demas
 * componentes del grupo de proyecto
 *
 * @author Pablo
 *
 */
public class User {
    private Long id;
    private String login;
    private String email;
    private String password;
    private String language;
    private Boolean admin;
    private Boolean manager;
    private Long idProyecto;
    private String create;
    private String modify;

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public String getLogin() {
        return login;
    }

    public void setLogin(String login) {
        this.login = login;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    public String getEmail() {
        return email;
```

```
    }

    public void setEmail(String email) {
        this.email = email;
    }

    public String getLanguage() {
        return language;
    }

    public void setLanguage(String language) {
        this.language = language;
    }

    public Boolean isAdmin() {
        return admin;
    }

    public void setAdmin(Boolean admin) {
        this.admin = admin;
    }

    public Boolean isManager() {
        return manager;
    }

    public void setManager(Boolean manager) {
        this.manager = manager;
    }

    public Long getIdProyecto() {
        return idProyecto;
    }

    public void setIdProyecto(Long idProyecto) {
        this.idProyecto = idProyecto;
    }

    public String getCreate() {
        return create;
    }

    public void setCreate(String create) {
        this.create = create;
    }

    public String getModify() {
        return modify;
    }

    public void setModify(String modify) {
        this.modify = modify;
    }
}
```

13.4.2 Paquete impl.miw.presentation:

13.4.2.1 Fichero "LoginAction.java":

```
package impl.miw.presentation.login;

import java.util.Vector;

import javax.servlet.http.HttpServletRequest;

import org.apache.struts2.interceptor.ServletRequestAware;

import com.miw.business.UserService;
import com.miw.infrastructure.log.LogService;
import com.miw.model.User;
import com.opensymphony.xwork2.ActionSupport;
import com.opensymphony.xwork2.ModelDriven;

/**
 * Clase de la capa de presentación para la acción de logueo de los diferentes
 * usuarios de la aplicación comprobandose que existán en la base de datos,
 * extiende de ActionSupport que nos proporciona una implementación por defecto
 * para las acciones más comunes con implementación de una interface "aware"
 * para alojar objetos que puedan estar a disposición en otras partes de la
 * aplicación y del ModelDriven que proporciona un objeto de modelo que se
 * inserta en el ValueStack además de la propia acción
 *
 * @author Pablo
 *
 */
public class LoginAction extends ActionSupport implements ServletRequestAware,
        ModelDriven<User> {

    private static final long serialVersionUID = 9105304741910854626L;
    private final User usuario = new User();
    private UserService userService;
    private HttpServletRequest request;

    private String email;
    private String password;
    private LogService log;

    /**
     * Getter de la Infraestructura de Log
     *
     * @return Log
     */
    public LogService getLog() {
        return log;
    }

    /**
     * Setter de la infraestructura de Log
     *
     * @param log
     */
    public void setLog(LogService log) {
        this.log = log;
    }

    /**
     * Getter para la obtención de la interfaz para los servicios aplicados al
     * modelo de usuarios
     *
     * @return UserService Objeto que hace referencia a la interfaz
     */
    public UserService getUserService() {
        log.debug("Invocado el getUserService de Login");
        return userService;
    }

    /**
```

```
     * Setter para establecer la interfaz para los servicios aplicados al modelo
     * de usuarios
     *
     * @param userService
     *            objecto interfaz
     */
    public void setUserService(UserService userService) {
        log.debug("Invocado el setUserService de Login");
        this.userService = userService;
    }

    /**
     * Setter que establece la solicitud al objeto HTTP en las clases de la
     * aplicación
     *
     * @param httpServletRequest
     *            petición del Servlet
     */
    @Override
    public void setServletRequest(HttpServletRequest httpServletRequest) {
        log.debug("Invocado el setServletRequest de Login");
        this.request = httpServletRequest;
    }

    /**
     * Método que implementa la ejecución del action de Struts con los datos del
     * request
     *
     * @return String cadena que será procesada en struts.xml
     */
    @Override
    public String execute() {
        // Supuestamente, por "dependecy injection" debería
        // tener un: usuario, password
        User user = (User) request.getSession().getAttribute(
                "usuarioRegistrado");
        if (user != null) {
            usuario.setId(user.getId());
            usuario.setLogin(user.getLogin());
            usuario.setEmail(user.getEmail());
            usuario.setPassword(user.getPassword());
            usuario.setAdmin(user.isAdmin());
            usuario.setManager(user.isManager());
            usuario.setLanguage(user.getLanguage());
            usuario.setIdProyecto(user.getIdProyecto());
            request.getSession().setAttribute("usuario", usuario);
            return SUCCESS;
        }
        log.debug("Procesando el execute de Login con ModelDriven");
        log.debug("Id: " + usuario.getId());
        log.debug("Nombre: " + usuario.getLogin());
        log.debug("Email: " + usuario.getEmail());
        log.debug("Password: " + usuario.getPassword());
        log.debug("Admin: " + usuario.isAdmin());
        log.debug("Manager: " + usuario.isManager());
        log.debug("Language: " + usuario.getLanguage());

        // Recuperando los usuarios de la base de datos
        Vector<User> users;
        try {
            users = userService.getUsers();

            for (int i = 0; i < users.size(); i++) {

                // Colocamos el usuario en sesión. En caso de saltar
                // una excepción, no llegaría a colocarse nada.
                if (usuario.getEmail().equals(users.get(i).getEmail())
                        && usuario.getPassword().equals(
                                users.get(i).getPassword())) {
                    request.getSession().setAttribute("usuario", users.get(i));
                    break;
                }
            }
        } catch (Exception e) {
            log.error(e.getClass() + " " + e.getMessage());
            addActionError(getText("user.error") + " "
```

```
                    + getText("password.error"));
            return ERROR;
        }
        return SUCCESS;
    }

    /*
     * Getter y Setter del Request
     */
    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    /*
     * Fin Getter y Setter del Request
     */

    /**
     * Getter para obtener el modelo que se inserta en la ValueStack en lugar de
     * la propia acción
     *
     * @return User modelo de usuario
     */
    @Override
    public User getModel() {
        return usuario;
    }
```

13.4.3 Paquete impl.miw.presentation:

13.4.3.1 Fichero "LoginInterceptor.java":

```
package impl.miw.presentation.login;

import org.apache.struts2.StrutsStatics;

import com.miw.infrastructure.log.LogService;
import com.miw.model.User;
import com.opensymphony.xwork2.ActionContext;
import com.opensymphony.xwork2.ActionInvocation;
import com.opensymphony.xwork2.interceptor.AbstractInterceptor;

/**
 * Clase para implementar los interceptores, los cuales actuarán como un
 * filtroHTTP o un Proxy, proveerán un camino para suplir el pre-processing y
 * post-processing a través del action
 *
 * @author Pablo
 *
 */
public class LoginInterceptor extends AbstractInterceptor implements
        StrutsStatics {

    private static final long serialVersionUID = -4203964390619150335L;
    private LogService log;

    /**
     * Getter de la Infraestructura de Log
```

```
     *
     * @return Log
     */
    public LogService getLog() {
        return log;
    }

    /**
     * Setter de la infraestructura de Log
     *
     * @param log
     */
    public void setLog(LogService log) {
        this.log = log;
    }

    /**
     * Método de destrucción del interceptor
     */
    @Override
    public void destroy() {
        log.debug("Destroy invocado de LoginInterceptor");
    }

    /**
     * Método de inicialización del interceptor
     */
    @Override
    public void init() {
        log.debug("init de LoginInterceptor");

    }

    /**
     * Método de ejecución del interceptor, dependiendo que tipo de usuario
     * entre se le lleva a una acción o a otra en la aplicación
     *
     * @param inv
     *            Invocación de la acción
     * @return String cadena de la acción a procesar
     */
    @Override
    public String intercept(ActionInvocation inv) throws Exception {
        ActionContext ctx = inv.getInvocationContext();
        User usuario = (User) ctx.getSession().get("usuario");
        String uri = ctx.getName().toLowerCase();
        if (usuario == null && !uri.contains("login")) {
            return "login";
        } else if (usuario.isAdmin()) {
            // Página del admin
            return "admin";
        } else if (usuario.isManager()) {
            // Página del manager
            return "manager";
        } else {
            // Página de demás miembros
            return "miembro";
        }
    }
}
```

13.4.4 Paquete impl.miw.presentation:

13.4.4.1 Fichero "SessionInterceptor.java":

```
package impl.miw.presentation.login;

import java.util.Map;

import javax.servlet.http.HttpServletRequest;

import org.apache.struts2.interceptor.ServletRequestAware;

import com.miw.infrastructure.log.LogService;
import com.opensymphony.xwork2.ActionContext;
import com.opensymphony.xwork2.ActionInvocation;
import com.opensymphony.xwork2.interceptor.AbstractInterceptor;

/**
 * Clase que crea un interceptor, cuando la sesión de la aplicación halla
 * caducado se pondrán las variables a null con la implementación de una
 * interfaz "aware" para eliminar esas variables de sesión y extiende de
 * AbstractInterceptor
 *
 * @author Pablo
 *
 */
public class SessionInterceptor extends AbstractInterceptor implements
        ServletRequestAware {

    private static final long serialVersionUID = 1L;
    private HttpServletRequest request;
    private LogService log;

    /**
     * Getter de la Infraestructura de Log
     *
     * @return Log
     */
    public LogService getLog() {
        return log;
    }

    /**
     * Setter de la infraestructura de Log
     *
     * @param log
     */
    public void setLog(LogService log) {
        this.log = log;
    }

    /**
     * Método de destrucción del interceptor
     */
    @Override
    public void destroy() {
        log.debug("Destroy invocado de SessionInterceptor");
    }

    /**
     * Método de inicialización del interceptor
     */
    @Override
    public void init() {
        log.debug("Init de SessionInterceptor");

    }

    /**
     * Setter que establece la solicitud al objeto HTTP en las clases de la
     * aplicación
     *
```

```
     * @param httpServletRequest
     *            petición del Servlet
     */
    @Override
    public void setServletRequest(HttpServletRequest httpServletRequest) {
        log.debug("Invocado el setServletRequest de UnLogin");
        this.request = httpServletRequest;
    }

    /**
     * Método de ejecución del interceptor, si la sesion ha expirado, te
     * devuelve al login
     *
     * @param invocation
     *            Invocación de la acción
     * @return String cadena de la acción a procesar
     * @exception Exception
     */
    @Override
    public String intercept(ActionInvocation invocation) throws Exception {
        ActionContext ctx = invocation.getInvocationContext();
        String uri = ctx.getName().toLowerCase();
        Map<String, Object> session = invocation.getInvocationContext().getSession();

            if (session.isEmpty() && !uri.contains("login")
                    && !uri.contains("counter") && !uri.contains("forgotten")
                    && !uri.contains("passforgotten") && !uri.contains("index")) {
                if (session.containsKey("usuario") &&
request.getSession().getAttribute("usuario") != null) {
                    request.getSession().setAttribute("usuario", "");
                    request.getSession().setAttribute("play", "");
                    request.getSession().setAttribute("saveplan", "");
                }
                return "login"; // session is empty/expired
            }
            return invocation.invoke();
    }
}
```

13.4.5 Paquete com.miw.business:

13.4.5.1 Fichero Interfaz "UserService.java":

```
package com.miw.business;

import impl.miw.presentation.passforgotten.PassForgottenAction;

import java.util.Vector;

import javax.mail.MessagingException;

import com.miw.model.Project;
import com.miw.model.User;

/**
 * Facade. Interfaz entre la capa de Presentación y Negocio para datos de
 * usuarios
 *
 * @author Pablo
 *
 */
public interface UserService {
    public Vector<User> getUsers() throws Exception;

    public String setUser(User user) throws Exception;

    public Boolean getPassForgotten(PassForgottenAction passForgottenAction)
            throws MessagingException, Exception;

    public String setUpdateUser(User user) throws Exception;
```

```
    public Vector<User> getManagers() throws Exception;

    public void send(User user, int optionSend, int i, Project pro)
            throws MessagingException;

    public User seekUser(String email) throws Exception;

    public Vector<User> getUsers(Long idProyecto) throws Exception;

    public Boolean deleteUser(Long idUser) throws Exception;

    public String setUpdateUserAdmin(User updateUser) throws Exception;

    public boolean getManager(Long idUserUpdate, Long idProject) throws Exception;
}
```

13.4.6 Paquete impl.miw.business:

13.4.6.1 Fichero "UserService.java":

```
package impl.miw.business.usuario;

import impl.miw.presentation.passforgotten.PassForgottenAction;

import java.util.Properties;
import java.util.ResourceBundle;
import java.util.Vector;

import javax.mail.Message;
import javax.mail.MessagingException;
import javax.mail.Session;
import javax.mail.Transport;
import javax.mail.internet.InternetAddress;
import javax.mail.internet.MimeMessage;

import com.miw.business.UserService;
import com.miw.infrastructure.log.LogService;
import com.miw.model.Project;
import com.miw.model.User;
import com.miw.persistence.UserDataService;

/**
 * Clase que implementa las operaciones para la capa de Negocio de los usuarios
 *
 * @author Pablo
 *
 */
public class Usuario implements UserService {

    static final String ENGLISH = "EN";
    static final String SPANISH = "ES";
    static final String MANAGER = "Manager";

    private UserDataService userDataService;
    private LogService log;

    private final Properties properties = new Properties();
    private Session session;

    /**
     * Getter de la Infraestructura de Log
     *
     * @return Log
     */
    public LogService getLog() {
        return log;
    }

    /**
```

```
     * Setter de la infraestructura de Log
     *
     * @param log
     */
    public void setLog(LogService log) {
        this.log = log;
    }

    /**
     * Getter de la interfaz entre el negocio y la persistencia de datos de
     * usuarios
     *
     * @return UserDataService interfaz entre el negocio y la persistencia
     */
    public UserDataService getUserDataService() {
        return userDataService;
    }

    /**
     * Setter de la interfaz entre el negocio y la persistencia de datos de
     * usuarios
     *
     * @param userDataService
     *            interfaz entre el negocio y la persistencia
     */
    public void setUserDataService(UserDataService userDataService) {
        this.userDataService = userDataService;
    }

    /**
     * Getter del método de la interfaz entre presentación y negocio para
     * obtener los usuarios
     *
     * @return Vector<User> vector con usuarios
     * @throws Exception
     */
    @Override
    public Vector<User> getUsers() throws Exception {
        log.debug("Entrando en getUsers");
        return userDataService.getUsers();
    }

    /**
     * Setter del método de la interfaz entre presentación y negocio para
     * establecer los usuarios
     *
     * @param user
     *            usuario
     * @return String valor devuelto para comprobar la inserción
     * @throws Exception
     */
    @Override
    public String setUser(User user) throws Exception {
        log.debug("Entrando en setUser");
        return userDataService.setUser(user);
    }

    /**
     * Getter del método de la interfaz entre presentación y negocio para
     * obtener la contraseña
     *
     * @param passForgottenAction
     *            el action con los datos
     * @return Boolean valor booleano de que se ha encontrado el suaurio y su
     *         contraseña
     * @throws Exception
     */
    @Override
    public Boolean getPassForgotten(PassForgottenAction passForgottenAction)
            throws Exception {
        log.debug("Entrando en setPassForgotten");
        User user = userDataService.getPassForgotten(passForgottenAction);

        // Cargar JavaMail
        if (user != null) {
            makeJavamail(user, 0, null);
```

```
        } else {
            return false;
        }
        return true;
    }

    /**
     * Método de la interfaz entre presentación y negocio para el envío de
     * correo
     *
     * @param user
     *            usuario
     * @param i
     *            opción de tipo de envio
     * @param pro
     *            proyecto
     * @throws MessagingException
     */
    @Override
    public void send(User user, int optionSend, int i, Project pro)
            throws MessagingException {
        switch (optionSend) {
        case 1:
            makeJavamail(user, i, pro);
            break;
        case 2:
            // SMS();
            break;
        case 3:
            // mensakeriaInterna();
            break;
        case 4:
            // whatsApp();
            break;
        case 5:
            // facebook();
            break;
        default:
            break;
        }
    }

    /**
     * Método para la creación de Javamail
     *
     * @param user
     *            usuario
     * @param op
     *            opción de tipo de envio
     * @param pro
     *            proyecto
     * @throws MessagingException
     */
    private void makeJavamail(User user, Integer op, Project pro)
            throws MessagingException {
        init();
        switch (op) {
        case 0:
            try {
                MimeMessage message = new MimeMessage(session);
                message.setFrom(new InternetAddress((String) properties
                        .get("mail.smtp.mail.sender")));
                message.addRecipient(Message.RecipientType.TO,
                        new InternetAddress(user.getEmail()));
                if (user.getLanguage().equalsIgnoreCase(ENGLISH)) {
                    message.setSubject(ResourceBundle.getBundle("global_en")
                            .getString("correo.subject_pass"));
                    message.setText(ResourceBundle.getBundle("global_en")
                            .getString("correo.text_pass") + user.getPassword());
                } else {
                    message.setSubject(ResourceBundle.getBundle("global")
                            .getString("correo.subject_pass"));
                    message.setText(ResourceBundle.getBundle("global")
                            .getString("correo.text_pass") + user.getPassword());
                }
                Transport t = session.getTransport("smtp");
```

```
            t.connect((String) properties.get("mail.smtp.user"), "tfm_2014");
            t.sendMessage(message, message.getAllRecipients());
            t.close();
        } catch (MessagingException me) {
            throw (me);
        }
        break;

    case 1:
        try {
            MimeMessage message = new MimeMessage(session);
            message.setFrom(new InternetAddress((String) properties
                    .get("mail.smtp.mail.sender")));
            message.addRecipient(Message.RecipientType.TO,
                    new InternetAddress(user.getEmail()));
            if (user.getLanguage().contains(ENGLISH)) {
                message.setSubject(ResourceBundle.getBundle("global_en")
                        .getString("correo.subject_new"));
                message.setText(ResourceBundle.getBundle("global_en")
                        .getString("correo.text_new_1")
                        + user.getLogin()
                        + "\n"
                        + ResourceBundle.getBundle("global_en").getString(
                                "correo.text_new_2")
                        + user.getEmail()
                        + "\n"
                        + ResourceBundle.getBundle("global_en").getString(
                                "correo.text_new_3")
                        + user.getPassword()
                        + "\n"
                        + ResourceBundle.getBundle("global_en").getString(
                                "correo.text_new_4")
                        + MANAGER
                        + "\n"
                        + ResourceBundle.getBundle("global_en").getString(
                                "correo.text_new_5") + user.getLanguage());
            } else {
                message.setSubject(ResourceBundle.getBundle("global")
                        .getString("correo.subject_new"));
                message.setText(ResourceBundle.getBundle("global")
                        .getString("correo.text_new_1")
                        + user.getLogin()
                        + "\n"
                        + ResourceBundle.getBundle("global").getString(
                                "correo.text_new_2")
                        + user.getEmail()
                        + "\n"
                        + ResourceBundle.getBundle("global").getString(
                                "correo.text_new_3")
                        + user.getPassword()
                        + "\n"
                        + ResourceBundle.getBundle("global").getString(
                                "correo.text_new_4")
                        + MANAGER
                        + "\n"
                        + ResourceBundle.getBundle("global").getString(
                                "correo.text_new_5") + user.getLanguage());
            }
            Transport t = session.getTransport("smtp");
            t.connect((String) properties.get("mail.smtp.user"), "tfm_2014");
            t.sendMessage(message, message.getAllRecipients());
            t.close();
        } catch (MessagingException me) {
            throw (me);
        }
        break;
    case 2:
        try {
            MimeMessage message = new MimeMessage(session);
            message.setFrom(new InternetAddress((String) properties
                    .get("mail.smtp.mail.sender")));
            message.addRecipient(Message.RecipientType.TO,
                    new InternetAddress(user.getEmail()));
            if (user.getLanguage().contains(ENGLISH)) {
                message.setSubject(ResourceBundle.getBundle("global")
                        .getString("correo.subject_create"));
```

```
                message.setText(ResourceBundle.getBundle("global")
                        .getString("correo.text_create_1")
                        + pro.getNombre()
                        + "\n"
                        + ResourceBundle.getBundle("global").getString(
                                "correo.text_create_2")
                        + pro.getFecha()
                        + "\n"
                        + ResourceBundle.getBundle("global").getString(
                                "correo.text_create_4") + pro.getPaso());
            } else {
                message.setSubject(ResourceBundle.getBundle("global")
                        .getString("correo.subject_create"));
                message.setText(ResourceBundle.getBundle("global")
                        .getString("correo.text_create_1")
                        + pro.getNombre()
                        + "\n"
                        + ResourceBundle.getBundle("global").getString(
                                "correo.text_create_2")
                        + pro.getFecha()
                        + "\n"
                        + ResourceBundle.getBundle("global").getString(
                                "correo.text_create_4") + pro.getPaso());
            }
            Transport t = session.getTransport("smtp");
            t.connect((String) properties.get("mail.smtp.user"), "tfm_2014");
            t.sendMessage(message, message.getAllRecipients());
            t.close();
        } catch (MessagingException me) {
            throw (me);
        }
        break;
    }
}

/**
 * Método de carga de parámetros iniciales para el envío de correos mediante
 * Gmail
 */
private void init() {
    properties.put("mail.smtp.host", "smtp.gmail.com");
    properties.put("mail.smtp.starttls.enable", "true");
    properties.put("mail.smtp.port", 587);
    properties.put("mail.smtp.mail.sender",
            "riskmanagementtfm@gmail.com");
    properties.put("mail.smtp.user", "riskmanagementtfm");
    properties.put("mail.smtp.auth", "true");

    session = Session.getDefaultInstance(properties);
}

/**
 * Setter del método de la interfaz entre presentación y negocio para
 * establecer la actualización de usuarios
 *
 * @param user
 *            usuario
 * @return String valor devuelto para el log
 * @throws Exception
 */
@Override
public String setUpdateUser(User user) throws Exception {
    log.debug("Entrando en setUpdateUser");
    return userDataService.setUpdateUser(user);
}

/**
 * Getter del método de la interfaz entre presentación y negocio para
 * obtener los managers
 *
 * @return Vector<User> vector con usuarios managers
 * @throws Exception
 */
@Override
public Vector<User> getManagers() throws Exception {
    log.debug("Entrando en getManagers");
```

```
        return userDataService.getManagers();
    }

    /**
     * Método de la interfaz entre presentación y negocio para la busqueda de
     * usuarios por medio del email
     *
     * @param email
     *            correo
     * @return User usuario
     * @throws Exception
     */
    @Override
    public User seekUser(String email) throws Exception {
        log.debug("Entrando en seekUser");
        return userDataService.getUser(email);
    }

    /**
     * Getter del método de la interfaz entre presentación y negocio para
     * obtener los usuarios por medio del proyecto
     *
     * @param idProyecto
     *            identificación de proyecto
     * @return Vector<User> vector con usuarios
     * @throws Exception
     */
    @Override
    public Vector<User> getUsers(Long idProyecto) throws Exception {
        log.debug("Entrando en getUsers");
        return userDataService.getUsers(idProyecto);
    }

    /**
     * Método de la interfaz entre presentación y negocio para eliminar el
     * usuario
     *
     * @param idUser
     *            identificación de usuario
     * @return Booelan de comprobación para el borrado
     * @throws Exception
     */
    @Override
    public Boolean deleteUser(Long idUser) throws Exception {
        log.debug("Entrando en deleteUser");
        return userDataService.deleteUser(idUser);
    }

    /**
     * Método de la interfaz entre presentación y negocio para la actualización
     * de datos de admin
     *
     * @param updateUser
     *            usuario a actualizar
     * @return String valor devuelto para el log
     * @throws Exception
     */
    @Override
    public String setUpdateUserAdmin(User updateUser) throws Exception {
        log.debug("Entrando en setUpdateUserAdmin");
        return userDataService.setUpdateUserAdmin(updateUser);
    }

    /**
     * Método de la interfaz entre presentación y negocio para la busqueda de
     * usuarios por medio del id
     *
     * @param idUserUpdate
     *            identificador del usuario
     * @param idProject
     *            identificador del proyecto
     * @return Boolean true si existe o false sino no
     * @throws Exception
     */
    @Override
    public boolean getManager(Long idUserUpdate, Long idProject) throws Exception {
```

```
        log.debug("Entrando en getManager");
        return userDataService.getManager(idUserUpdate, idProject);
    }
}
```

13.4.7 Paquete com.miw.persistence:

13.4.7.1 Fichero Interfaz "UserDataService.java":

```
package com.miw.persistence;

import impl.miw.presentation.passforgotten.PassForgottenAction;

import java.util.Vector;

import com.miw.model.User;

/**
 * Facade. Interfaz entre la capa de Negocio y Persistencia para datos de
 * usuarios
 *
 * @author Pablo
 *
 */
public interface UserDataService {
    public Vector<User> getUsers() throws Exception;

    public String setUser(User user) throws Exception;

    public User getPassForgotten(PassForgottenAction passForgottenAction)
            throws Exception;

    public String setUpdateUser(User user) throws Exception;

    public Vector<User> getManagers() throws Exception;

    public User getUser(String email) throws Exception;

    public Vector<User> getUsers(Long idProyecto) throws Exception;

    public Boolean deleteUser(Long idUser) throws Exception;

    public String setUpdateUserAdmin(User updateUser) throws Exception;

    public boolean getManager(Long idUserUpdate, Long idProject) throws Exception;
}
```

13.4.8 Paquete impl.miw.persistence:

13.4.8.1 Fichero "UserDAO.java":

```
package impl.miw.persistence.user;

import impl.miw.persistence.Jdbc;
import impl.miw.persistence.project.ProjectDAO;
import impl.miw.presentation.passforgotten.PassForgottenAction;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.util.Vector;

import com.miw.model.User;
import com.miw.persistence.UserDataService;
```

```
/**
 * Clase que implementa las operaciones de acceso a la base de datos para la
 * entidad USER.
 *
 * @author Pablo
 */
public class UserDAO implements UserDataService {

    /**
     * Getter para la obtención de los usuarios en la base de datos
     *
     * @return Vector vector con los datos de los usuarios
     * @throws Exception
     */
    @Override
    public Vector<User> getUsers() throws Exception {
        // Inicializamos el Vector de retorno.
        Vector<User> resultado = new Vector<User>();
        PreparedStatement ps = null;

        ResultSet rs = null;
        Connection con = null;

        try {
            con = Jdbc.getConnection();

            ps = con.prepareStatement("SELECT @pass:='tfm14';");
            ps.executeQuery();

            ps = con.prepareStatement("select u_id as id, p_id,user,
AES_DECRYPT(email,@pass) as email, AES_DECRYPT(pass,@pass) as pass, admin, manager,
language from users");
            rs = ps.executeQuery();

            while (rs.next()) {
                // Completamos los datos del user en la entidad
                User u = new User();
                u.setId(rs.getLong("id"));
                u.setLogin(rs.getString("user"));
                u.setEmail(rs.getString("email"));
                u.setPassword(rs.getString("pass"));
                u.setAdmin(rs.getBoolean("admin"));
                u.setManager(rs.getBoolean("manager"));
                u.setLanguage(rs.getString("language"));
                u.setIdProyecto(rs.getLong("p_id"));
                // La añadimos al Vector de resultado
                resultado.add(u);
            }

        } catch (Exception e) {
            e.printStackTrace();
            throw (e);
        } finally {
            try {
                ps.close();
                con.close();
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
        // Retornamos el vector de resultado.
        return resultado;
    }

    /**
     * Setter para establecer los datos del usuario
     *
     * @param user
     *            datos del usuario
     * @return String cadena para log
     * @throws Exception
     */
    @Override
    public String setUser(User user) throws Exception {
        // Inicializamos el String de retorno.
        String resultado = "";
```

```
        if (!existEmail(user)) {

            PreparedStatement ps = null;
            Connection con = null;

            try {

                con = Jdbc.getConnection();
                ps = con.prepareStatement("SELECT @pass:='tfm14';");
                ps.executeQuery();

                if (user.getIdProyecto() != null) {
                    ps = con.prepareStatement("INSERT INTO users
(user,email,pass,language,admin,manager,created,p id) VALUES
(?,AES_ENCRYPT(?,@pass),AES_ENCRYPT(?,@pass),?,?,?,?,?);");

                    ps.setString(1, user.getLogin());
                    ps.setString(2, user.getEmail());
                    ps.setString(3, user.getPassword());
                    String language = user.getLanguage();
                    String[] pieces = language.split(" ");
                    language = pieces[0];
                    ps.setString(4, language);
                    ps.setBoolean(5, user.isAdmin());
                    ps.setBoolean(6, user.isManager());
                    ps.setTimestamp(7, null);
                    ps.setLong(8, user.getIdProyecto());

                    ps.executeUpdate();
                } else {
                    ps = con.prepareStatement("INSERT INTO users
(user,email,pass,language,admin,manager,created) VALUES
(?,AES_ENCRYPT(?,@pass),AES_ENCRYPT(?,@pass),?,?,?,?);");

                    ps.setString(1, user.getLogin());
                    ps.setString(2, user.getEmail());
                    ps.setString(3, user.getPassword());
                    String language = user.getLanguage();
                    String[] pieces = language.split(" ");
                    language = pieces[0];
                    ps.setString(4, language);
                    ps.setBoolean(5, user.isAdmin());
                    ps.setBoolean(6, user.isManager());
                    ps.setTimestamp(7, null);

                    ps.executeUpdate();
                }

                if (user.isManager() && user.getIdProyecto() != null) {
                    updateStepProject(ps, con, user.getIdProyecto(), 1);
                } else {

                }

                resultado = "Datos tratados correctamente";

            } catch (Exception e) {
                resultado = "Datos no tratados";
                e.printStackTrace();
                throw (e);
            } finally {
                try {
                    ps.close();
                    con.close();
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }

        } else {
            resultado = "Email";
        }
        // Retornamos el vector de resultado.
        return resultado;
    }
```

```
    /**
     * Método para actualizar el paso de generación de la Gestión de Riesgos en
     * el proyecto actual
     *
     * @param ps
     *            declaración de la qs
     * @param con
     *            conexión
     * @param idProyecto
     *            identificador del proyecto a actualizar
     * @param paso
     *            número de paso
     * @exception SQLException
     */
    private void updateStepProject(PreparedStatement ps, Connection con,
            Long idProyecto, int paso) throws Exception {

        ProjectDAO pDAO = new ProjectDAO();
        pDAO.updateStep(idProyecto, paso);
    }

    /**
     * Método para verificar la existencia del correo del usuario
     *
     * @param user
     *            usuario
     * @return boolean booleano true si existe false sino
     * @throws Exception
     */
    private boolean existEmail(User user) throws Exception {
        boolean resultado = false;
        PreparedStatement ps = null;
        ResultSet rs = null;
        Connection con = null;

        try {
            con = Jdbc.getConnection();

            ps = con.prepareStatement("SELECT @pass:='tfm14';");
            ps.executeQuery();

            ps = con.prepareStatement("select user from users where
email=AES_ENCRYPT(?,@pass)");

            ps.setString(1, user.getEmail());
            rs = ps.executeQuery();

            if (rs.next()) {
                resultado = true;
            }

        } catch (Exception e) {
            e.printStackTrace();
            throw (e);
        } finally {
            try {
                ps.close();
                con.close();
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
        return resultado;
    }

    /**
     * Getter para la obtención de la contraseña
     *
     * @param passForgottenAction
     *            action con todos los datos recibidos de la request
     * @return User usuario con todos sus datos
     * @throws Exception
     */
    @Override
    public User getPassForgotten(PassForgottenAction passForgottenAction)
```

```
            throws Exception {
        // Inicializamos el Vector de retorno.
        User resultado = new User();

        PreparedStatement ps = null;
        ResultSet rs = null;
        Connection con = null;

        try {
            con = Jdbc.getConnection();

            ps = con.prepareStatement("SELECT @pass:='tfm14';");
            ps.executeQuery();

            ps = con.prepareStatement("select user, AES_DECRYPT(pass,@pass) as pass,
AES_DECRYPT(email,@pass) as email, language  from users where
email=AES_ENCRYPT(?,@pass)");

            ps.setString(1, passForgottenAction.getModel().getEmail());
            rs = ps.executeQuery();

            User u = null;
            while (rs.next()) {
                u = new User();
                u.setLanguage(rs.getString("language"));
                u.setPassword(rs.getString("pass"));
                u.setLogin(rs.getString("user"));
                u.setEmail(rs.getString("email"));
            }
            resultado = u;

        } catch (Exception e) {
            e.printStackTrace();
            throw (e);
        } finally {
            try {
                ps.close();
                con.close();
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
        // Retornamos el resultado.
        return resultado;
    }

    /**
     * Setter para actualizar el usuario
     *
     * @param user
     *            datos del usuario que va a actualizarse
     * @return String cadena para el log
     * @throws Exception
     */
    @Override
    public String setUpdateUser(User user) throws Exception {
        String resultado = "";
        PreparedStatement ps = null;
        Connection con = null;

        try {

            con = Jdbc.getConnection();
            ps = con.prepareStatement("SELECT @pass:='tfm14';");
            ps.executeQuery();

            ps = con.prepareStatement("UPDATE users SET user=?,
email=AES_ENCRYPT(?,@pass), pass=AES_ENCRYPT(?,@pass), language=? WHERE  u_id=?");

            ps.setString(1, user.getLogin());
            ps.setString(2, user.getEmail());
            ps.setString(3, user.getPassword());
            String language = user.getLanguage();
            String[] pieces = language.split(" ");
            language = pieces[0];
            ps.setString(4, language);
```

```
            ps.setLong(5, user.getId());

            ps.executeUpdate();

            resultado = "Actualizado";

        } catch (Exception e) {
            resultado = "No Actualizado";
            e.printStackTrace();
            throw (e);
        } finally {
            try {
                ps.close();
                con.close();
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
        return resultado;
    }

    /**
     * Getter para la obtención de los managers
     *
     * @return Vector vector con los datos de los usuarios tipo manager
     * @throws Exception
     */
    @Override
    public Vector<User> getManagers() throws Exception {
        // Inicializamos el Vector de retorno.
        Vector<User> resultado = new Vector<User>();
        PreparedStatement ps = null;

        ResultSet rs = null;
        Connection con = null;

        try {
            con = Jdbc.getConnection();

            ps = con.prepareStatement("SELECT @pass:='tfm14';");
            ps.executeQuery();

            ps = con.prepareStatement("select u_id as id, p_id,user,
AES_DECRYPT(email,@pass) as email, created, modify from users where manager = '1';");
            rs = ps.executeQuery();

            while (rs.next()) {
                // Completamos los datos del user en la entidad
                User u = new User();
                u.setId(rs.getLong("id"));
                u.setLogin(rs.getString("user"));
                u.setEmail(rs.getString("email"));
                u.setIdProyecto(rs.getLong("p_id"));
                u.setCreate(rs.getString("created"));
                u.setModify(rs.getString("modify"));
                // La añadimos al Vector de resultado
                resultado.add(u);
            }

        } catch (Exception e) {
            e.printStackTrace();
            throw (e);
        } finally {
            try {
                ps.close();
                con.close();
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
        // Retornamos el vector de resultado.
        return resultado;
    }

    /**
     * Getter para la obtención de los usuarios por medio de su correo
```

```
     *
     * @param email
     *            correo identificador del usuario
     *
     * @return User usuarios de base de datos
     * @throws Exception
     */
    @Override
    public User getUser(String email) throws Exception {
        User u = null;
        PreparedStatement ps = null;

        ResultSet rs = null;
        Connection con = null;

        try {
            con = Jdbc.getConnection();

            ps = con.prepareStatement("SELECT @pass:='tfm14';");
            ps.executeQuery();

            ps = con.prepareStatement("select user, AES_DECRYPT(email,@pass) as email,
AES_DECRYPT(pass,@pass) as pass, language from users where
email=AES_ENCRYPT(?,@pass);");

            ps.setString(1, email);
            rs = ps.executeQuery();

            while (rs.next()) {
                u = new User();
                u.setLogin(rs.getString("user"));
                u.setEmail(rs.getString("email"));
                u.setPassword(rs.getString("pass"));
                u.setLanguage(rs.getString("language"));
            }

        } catch (Exception e) {
            e.printStackTrace();
            throw (e);
        } finally {
            try {
                ps.close();
                con.close();
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
        return u;
    }

    /**
     * Getter para la obtención de los usuarios por medio del proyecto
     *
     * @param idProyecto
     *            identificador del proyecto
     *
     * @return Vector vector con los usurios de ese proyecto
     * @throws Exception
     */
    @Override
    public Vector<User> getUsers(Long idProyecto) throws Exception {
        // Inicializamos el Vector de retorno.
        Vector<User> resultado = new Vector<User>();
        PreparedStatement ps = null;

        ResultSet rs = null;
        Connection con = null;

        try {
            con = Jdbc.getConnection();

            ps = con.prepareStatement("SELECT @pass:='tfm14';");
            ps.executeQuery();
```

```
            ps = con.prepareStatement("select u_id as id, p_id,user,
AES_DECRYPT(email,@pass) as email, manager, language, created, modify from users where
p_id=?");

            ps.setLong(1, idProyecto);
            rs = ps.executeQuery();

            while (rs.next()) {
                // Completamos los datos del user en la entidad
                User u = new User();
                u.setId(rs.getLong("id"));
                u.setLogin(rs.getString("user"));
                u.setEmail(rs.getString("email"));
                u.setManager(rs.getBoolean("manager"));
                u.setLanguage(rs.getString("language"));
                u.setIdProyecto(rs.getLong("p_id"));
                u.setCreate(rs.getString("created"));
                u.setModify(rs.getString("modify"));
                // La añadimos al Vector de resultado
                resultado.add(u);
            }

        } catch (Exception e) {
            e.printStackTrace();
            throw (e);
        } finally {
            try {
                ps.close();
                con.close();
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
        // Retornamos el vector de resultado.
        return resultado;
    }

    /**
     * Método para elimianr usuarios
     *
     * @param idUser
     *            identificador del usuario a eliminar
     * @return Boolean booleano true si se ha borrado sino false
     * @throws Exception
     */
    @Override
    public Boolean deleteUser(Long idUser) throws Exception {
        // Inicializamos el String de retorno.
        PreparedStatement ps = null;
        Connection con = null;
        ResultSet rs = null;

        try {
            con = Jdbc.getConnection();

            ps = con.prepareStatement("SELECT @pass:='tfm14';");
            ps.executeQuery();

            ps = con.prepareStatement("select user, AES_DECRYPT(email,@pass) as email
from users where u_id=?;");
            ps.setLong(1, idUser);

            String manager = "";
            String email = "";

            rs = ps.executeQuery();
            while (rs.next()) {
                manager = rs.getString("user");
                email = rs.getString("email");
            }

            ps = con.prepareStatement("UPDATE projects SET manager='',
email=AES_ENCRYPT('',@pass) WHERE email=AES_ENCRYPT(?,@pass) and manager=?;");
            ps.setString(1, email);
            ps.setString(2, manager);
            ps.executeUpdate();
```

```
            ps = con.prepareStatement("delete from users where u_id=?;");
            ps.setLong(1, idUser);
            ps.executeUpdate();

            return true;

        } catch (Exception e) {
            return false;
        } finally {
            try {
                ps.close();
                con.close();
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    }

    /**
     * Setter para actualizar el usuario admin
     *
     * @param updateUser
     *            datos del usuario que va a actualizarse
     * @return String cadena para el log
     * @throws Exception
     */
    @Override
    public String setUpdateUserAdmin(User updateUser) throws Exception {
        String resultado = "";
        PreparedStatement ps = null;
        Connection con = null;

        try {

            con = Jdbc.getConnection();
            ps = con.prepareStatement("SELECT @pass:='tfm14';");
            ps.executeQuery();

            ps = con.prepareStatement("UPDATE users SET user=?,
email=AES_ENCRYPT(?,@pass), p_id=? WHERE u_id=?");

            ps.setString(1, updateUser.getLogin());
            ps.setString(2, updateUser.getEmail());
            ps.setLong(3, updateUser.getIdProyecto());
            ps.setLong(4, updateUser.getId());

            ps.executeUpdate();

            ps = con.prepareStatement("UPDATE projects SET manager=?,
email=AES_ENCRYPT(?,@pass) WHERE p_id=?;");
            ps.setString(1, updateUser.getLogin());
            ps.setString(2, updateUser.getEmail());
            ps.setLong(3, updateUser.getIdProyecto());
            ps.executeUpdate();

            resultado = "Actualizado";

        } catch (Exception e) {
            resultado = "No Actualizado";
            e.printStackTrace();
            throw (e);
        } finally {
            try {
                ps.close();
                con.close();
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
        return resultado;
    }

    /**
     * Getter para la obtención de un usuario por su id
     *
```

```
     * @param idUserUpdate
     *            identificador del usuario
     *
     * @return Boolean true si existe false sino no
     * @throws Exception
     */
    @Override
    public boolean getManager(Long idUserUpdate, Long idProject) throws Exception {
        boolean resultado = false;
        PreparedStatement ps = null;
        ResultSet rs = null;
        Connection con = null;

        try {
            con = Jdbc.getConnection();

            ps = con.prepareStatement("SELECT @pass:='tfm14';");
            ps.executeQuery();

            ps = con.prepareStatement("select user from users where u_id=? and p_id=?");

            ps.setLong(1, idUserUpdate);
            ps.setLong(2, idProject);
            rs = ps.executeQuery();

            if (rs.next()) {
                resultado = true;
            }

        } catch (Exception e) {
            e.printStackTrace();
            throw (e);
        } finally {
            try {
                ps.close();
                con.close();
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
        return resultado;
    }
}
```