

UNIVERSIDAD DE OVIEDO



ESCUELA INGENIERÍA INFORMÁTICA

TRABAJO FIN DE MÁSTER

“Sistema de monitorización y programación de Backups”

Director: Dr. Agustín Cernuda Del Río

Autor: Faustino Alonso Posada

Agradecimientos

A Ecocomputer S.L., en especial a Myrtha y Agustín, por su colaboración e interés en el devenir del proyecto.

Y al Dr. Agustín Cernuda Del Río, director de este trabajo fin de máster.

Resumen

El proyecto aquí presentado consiste en una infraestructura que permita la monitorización por parte de una empresa, de copias de seguridad realizadas por sus clientes. El proyecto ha sido realizado para la empresa Ecocomputer S.L. la cual ofrece a sus clientes servicios de hosting y mantenimiento de sus instalaciones entre otros servicios. Se pretende con este proyecto, controlar que las copias de seguridad de los clientes se realizan de manera adecuada.

La infraestructura desarrollada consiste en una plataforma Web que ofrece resultados sobre las copias realizadas por los clientes. Esta plataforma se alimenta de los datos emitidos por aplicaciones de escritorio instaladas en los ordenadores y/o servidores de los clientes. Resumiendo la funcionalidad básica de cada parte, podemos dividir el proyecto en 3 módulos:

- Aplicación Cliente: Aplicación de escritorio que permitirá programar Backups de bases de datos SQL Server o de directorios de los equipos de los clientes, realizará las copias y almacenará un histórico de resultados.
- Plataforma Web: Permitirá a los usuarios registrados (Personal de la empresa y clientes autorizados) observar los backups realizados y los resultados de los mismos.
- Servicio Web: Será necesario para comunicar los módulos anteriores. La aplicación cliente enviará mediante éste los resultados a la base de datos de la plataforma Web.

De tal manera que el cliente tendrá en su equipo instalado el primer módulo y en este software deberá programar las copias de seguridad que desea realizar de su equipo.

La aplicación Web tendrá una función informativa, mostrando los resultados de los backups realizados. Ésta deberá integrarse con una plataforma de la empresa y hacer uso de su sistema de gestión de usuarios.

El servicio Web será únicamente el canal de comunicación de las aplicaciones de los clientes con la base de datos utilizada por la plataforma Web.

El proyecto se ha visto alterado en cuanto a alcance, debido a modificaciones importantes en los requisitos en una fase avanzada del mismo. El cliente consideró necesarias funcionalidades no contempladas al inicio del proyecto y fue necesario realizar un nuevo diseño para contemplarlas.

La tecnología utilizada para todo el proyecto, ha sido a petición de la empresa libre de licencias y autorizada para su comercialización. Además, el proyecto ha sido principalmente desarrollado en lenguaje Java y para los sistemas de bases de datos se ha hecho uso de SQL Server en el caso de la plataforma y el servicio Web y SQLite para las aplicaciones de escritorio cliente.

Para la realización de pruebas se han utilizado herramientas como JUnit, para el control de pruebas unitarias; y otras como: TAW, HERA, aDesigner . . . para apoyar las pruebas de usabilidad y accesibilidad realizadas con seis sujetos. Éstas se han visto acompañadas de pruebas manuales para la evaluación del sistema.

Palabras clave

Backup,FTP, programador de tareas, SQL Server

Abstract

The developed project is an infrastructure that enables monitoring of backups made by customers. The project was carried out to EcoComputer limited liability company which offers clients hosting, maintenance and other services. This project aims to control customers do their backups properly.

The created infrastructure is a web platform for displaying the results on copies made by customers. This platform is updated by desktop applications installed on computers and / or servers of customers. Summarizing the basic functionality of each part, we can divide the project into 3 modules:

- Client Application: Desktop application that let you schedule backups of SQL Server databases or directories, make copies and store a history of results.
- Web Platform: Allow registered users (employees of the company and authorized customers) to display backups made ??and the results thereof.
- Web Service: It will be necessary to communicate the other modules. The client application sends by this web service, information to the Web platform database.

So the customer will have in his computer installed the first module, and on this software must program the backup that want to perform.

The Web application will have an informative function, showing the results of backups made. This should be integrated with an enterprise platform and make use of that user management system.

The Web service will only make a communication channel for customers to connect to the database used by the Web platform.

The project scope has been modified by significant changes in the requirements, in an advanced stage of development. The customer requested features that were not planned at the beginning of the project and it became necessary to make a new design.

The technology used to develop the project, has been free of licenses and authorized for marketing. In addition, the project has been developed in Java and have used database systems: SQL Server and SQLite.

For testing I used tools like JUnit and TAW, HERA, aDesigner ... to contrast the usability and accessibility tests conducted with six people. In addition, manual tests have been performed to evaluate the system.

Keywords

Backup,FTP, Scheduler, SQL Server

Índice general

1. Introducción	19
1.1. Motivación y justificación	19
1.2. Estudio de la situación previa al proyecto	20
1.3. Objetivos del proyecto	21
2. Aspectos teóricos	23
2.1. Backup (copia de seguridad)	23
2.1.1. Descripción	23
2.1.2. Aplicación en el proyecto	23
2.2. Quartz Scheduler	24
2.2.1. Descripción	24
2.2.2. Aplicación en el proyecto	24
2.3. Lenguaje unificado de modelado	25
2.3.1. Descripción	25
2.3.2. Aplicación en el proyecto	25
2.4. Métrica versión 3	25
2.4.1. Descripción	25
2.4.2. Aplicación en el proyecto	25
2.5. Struts2	26
2.5.1. Descripción	26
2.5.2. Aplicación en el proyecto	26
2.6. SQL Server 2005	27
2.6.1. Descripción	27
2.6.2. Aplicación en el proyecto	27
2.7. Apache Tomcat	27
2.7.1. Descripción	27
2.7.2. Aplicación en el proyecto	27
3. Planificación del proyecto y resumen de presupuestos	29
3.1. Planificación	29
3.1.1. Planificación inicial del proyecto	30
3.1.2. Planificación final del proyecto	31
3.1.3. Justificación de las diferencias	32
3.2. Resumen del presupuesto	33
4. Análisis	35
4.1. Definición del sistema	35
4.1.1. Determinación del alcance del sistema	35
4.2. Requisitos del sistema	35
4.2.1. Obtención de los requisitos del sistema	35
4.2.2. Identificación de actores del sistema	39
4.2.3. Especificación de casos de uso	39
4.3. Identificación de los subsistemas en la fase de análisis	42

4.3.1.	Descripción de los subsistemas	42
4.3.2.	Descripción de los interfaces entre subsistemas	43
4.4.	Diagrama de clases preliminar del análisis	44
4.4.1.	Diagrama de clases	44
4.4.2.	Descripción de las clases	45
4.5.	Análisis de casos de uso y escenarios	51
4.5.1.	Aplicación cliente para Windows	51
4.5.2.	Aplicación Web	54
4.6.	Análisis de interfaces de usuario	55
4.6.1.	Aplicación cliente para Windows	55
4.6.2.	Aplicación Web	56
4.7.	Especificación del plan de pruebas	57
5.	Diseño del Sistema	59
5.1.	Arquitectura del sistema	59
5.1.1.	Diagramas de paquetes	59
5.1.2.	Diagrama de componentes	61
5.1.3.	Diagrama de despliegue	61
5.2.	Diseño de clases	63
5.2.1.	Diagrama de clases	63
5.3.	Diagramas de interacción y estados	68
5.3.1.	Aplicación de escritorio para entornos Windows	68
5.3.2.	Aplicación Web	69
5.4.	Diagrama de secuencia	69
5.5.	Diseño de la base de datos	71
5.5.1.	Descripción de los sistemas de gestión de bases de datos	71
5.5.2.	Integración de lo sistemas de gestión de bases de datos	71
5.5.3.	Diagrama entidad-relación	72
5.6.	Diseño de la interfaz	73
5.6.1.	Aplicación de escritorio para Windows	73
5.6.2.	Aplicación Web	75
5.7.	Especificación técnica del plan de pruebas	76
5.7.1.	Pruebas unitarias	76
5.7.2.	Pruebas de integración y del sistema	76
5.7.3.	Pruebas de usabilidad y accesibilidad	76
5.8.	Decisiones del proyectante	77
5.8.1.	Seguridad en las contraseñas	77
5.8.2.	Código sencillo de mantener	78
5.8.3.	Base de datos sin eliminaciones	78
6.	Implementación del sistema	81
6.1.	Estándares y normas seguidos	81
6.1.1.	XHTML 1.1	81
6.1.2.	CSS 3	81
6.1.3.	Estándares de programación	82
6.1.4.	Modelo-vista-controlador	82
6.1.5.	Patrón de diseño DAO	82
6.2.	Lenguajes de programación	83
6.2.1.	Java	83
6.3.	Herramientas y programas usados para el desarrollo	85
6.3.1.	Eclipse	85
6.3.2.	SQL Server Management Studio Express	85
6.3.3.	Enterprise Architect	85
6.3.4.	Dia	85

6.3.5.	Microsoft Office Project	86
6.3.6.	TeXworks	86
6.4.	Creación del sistema	87
6.4.1.	Problemas encontrados	87
6.4.2.	Descripción detallada de las clases	89
7.	Desarrollo de las pruebas	91
7.1.	Pruebas unitarias	91
7.1.1.	Pruebas del programador de tareas	91
7.2.	Pruebas del sistema	94
7.3.	Pruebas de usabilidad y accesibilidad	95
7.3.1.	Pruebas de usabilidad	95
7.3.2.	Pruebas de accesibilidad	97
8.	Manuales del sistema	109
8.1.	Manual de instalación	109
8.1.1.	Instalación del cliente	109
8.1.2.	Instalación de la aplicación y el servicio Web	110
8.2.	Manual de ejecución	111
8.2.1.	Arrancar el servidor Web	111
8.2.2.	Arrancar la aplicación de escritorio	111
8.3.	Manual de usuario	112
8.3.1.	Aplicación de escritorio cliente	112
8.3.2.	Aplicación Web	116
8.4.	Manual del programador	118
8.4.1.	Aplicación de escritorio para Windows	118
8.4.2.	Aplicación Web	118
9.	Conclusiones y ampliaciones	121
9.1.	Conclusiones	121
9.2.	Ampliaciones	122
9.2.1.	Cliente para sistemas GNU/Linux	122
9.2.2.	Aplicación para dispositivos móviles	122
9.2.3.	Aplicación Web con mayor funcionalidad	122
9.2.4.	Realización de backups de máquinas virtuales	122
9.2.5.	Permitir copias incrementales o diferenciales	123
9.2.6.	Mejoras de usabilidad del producto	123
10.	Apéndices	125
10.1.	Primer análisis del proyecto	125
10.1.1.	Determinación del alcance del sistema	125
10.1.2.	Requisitos funcionales	126
10.2.	Instalación de Microsoft SQLServer 2005 y creación de la base de datos	129
10.2.1.	Servidor de base de datos	129
10.2.2.	Sistema de gestión de base de datos	130
10.2.3.	Base de datos	131
10.3.	Contenido Entregado en el CD-ROM	132

Índice de figuras

2.1. Funcionamiento general de struts2	26
3.1. Planificación inicial del proyecto.	30
3.2. Planificación final del proyecto.	31
4.1. Diagrama de casos de uso de la aplicación cliente en Windows	41
4.2. Diagrama de casos de uso de la aplicación Web	42
4.3. Diagrama relación entre los sistemas implicados en el proyecto.	43
4.4. Diagrama de clases de análisis, aplicación de escritorio	44
4.5. Diagrama de clases de análisis, aplicación Web	44
4.6. Autenticación en el sistema	55
4.7. Creación de una programación de Backup	55
4.8. Resultados de los Backups	56
4.9. Tareas creadas en el sistema	56
5.1. Diagrama de paquetes de la aplicación de escritorio para Windows	59
5.2. Diagrama de paquetes de la aplicación Web	60
5.3. Diagrama de componentes del proyecto	61
5.4. Diagrama de despliegue del proyecto	62
5.5. Diagrama de clases entities	63
5.6. Diagrama de clases del paquete Exec	64
5.7. Diagrama de clases del paquete Util	65
5.8. Diagrama de clases de la aplicación Web	66
5.9. Diagrama de clases del servicio Web	67
5.10. Diagrama de estados de creación del primer backup	68
5.11. Diagrama de estados de la aplicación Web	69
5.12. Diagrama de secuencia en la creación de un Backup	70
5.13. Diagrama Entidad-Relación de la base de datos de la aplicación cliente	72
5.14. Tablas empleadas por la aplicación Web	73
5.15. Diseño del panel de autenticación	73
5.16. Diseño del panel de resultados	74
5.17. Diseño del panel de Log y de Configuración	74
5.18. Diseño del panel de Log y de Configuración	74
5.19. Diseño de la aplicación Web. Resultado de Backups	75
5.20. Diseño de la aplicación Web. Resultado de Tareas	75
5.21. Diseño de la pantalla de conexiones con contraseñas visibles	77
5.22. Diseño de la base de datos en el primer análisis del proyecto	78
7.1. Captura del resultado de JUnit al pasar las pruebas del formateador de CRON	91
7.2. Captura del resultado de JUnit al pasar las pruebas del compresor de directorios	92
7.3. Captura del resultado de JUnit al pasar las pruebas para una copia local	93
7.4. Captura del resultado de JUnit a las pruebas de copia de una base de datos	93

7.5. Captura de la aplicación Web visualizada por distintos navegadores (1) . . .	97
7.6. Captura de la aplicación Web visualizada por distintos navegadores (2) . . .	97
7.7. Captura de la aplicación Web visualizada por un dispositivo móvil	98
7.8. Captura de la herramienta aDesigner, para el estudio de la accesibilidad . .	98
7.9. Captura de la herramienta TAW, para el estudio de la accesibilidad	100
7.10. Captura de la herramienta HERA, para el estudio de la accesibilidad	107
8.1. Configuración de las variables de entorno del sistema	109
8.2. Comprobación de la correcta configuración de Java	110
8.3. Captura de pantalla de registro de usuario	112
8.4. Captura del panel de autenticación	112
8.5. Captura del panel de resultados	113
8.6. Captura del panel de tareas	113
8.7. Captura del panel de creación de un backup programado	114
8.8. Captura del panel de conexiones	115
8.9. Captura del panel de Log	115
8.10. Captura del panel de configuración	116
8.11. Captura del sistema de autenticación Web	117
8.12. Captura de la aplicación Web en la pantalla de tareas	117
10.1. Instalación de Microsoft SQL Server, primer paso	129
10.2. Instalación de Microsoft SQL Server, segundo paso	130
10.3. Instalación del gestor de base de datos	130

Capítulo 1

Introducción

En la actualidad la información tiene en muchas ocasiones más poder que el dinero, es por ello que las empresas sienten la necesidad de mantener sus datos seguros y replicados para evitar desastres. En ocasiones contar con un sistema de creación y mantenimiento de las copias de seguridad no es una tarea trivial. Por ello empresas desarrolladoras de soluciones informáticas se ofrecen a aportar herramientas para facilitar el manejo y almacenamiento de datos a sus clientes.

Este proyecto consiste en una de dichas herramientas. Se pretende abstraer al usuario o cliente de la tarea de acordarse de realizar copias de seguridad de manera periódica, la comprobación de dichas copias y el posterior almacenamiento en un lugar relativamente seguro de las mismas. De esta forma, se puede delegar en dicha herramienta la creación de sistemas de backup y el posterior guardado en servidores independientes de la empresa cliente, si éste lo considera oportuno. Manteniendo a la empresa para la cual se ha desarrollado el proyecto al cargo de controlar la integridad de dichos datos.

1.1. Motivación y justificación

Este proyecto surge de la necesidad de mejorar una infraestructura y política de actuación existente. Ha sido realizado para la empresa Ecocomputer S.L. la cual ofrece a sus clientes un servicio de control sobre las copias de seguridad de los datos de los mismos. El proyecto surge de la necesidad de mejorar la política seguida por la empresa para realizar dicho servicio. Pretendiendo automatizar el proceso y permitiendo, de igual modo, un acercamiento de los clientes a la información manejada en el proceso.

Para entender la necesidad del proyecto, es necesario comprender la situación anterior a su utilización. El proceso seguido se podría resumir de la siguiente forma. Por un lado, los clientes cuentan con diferentes herramientas para la realización de copias de seguridad, en algunos casos éstas se realizan de manera autónoma y en otros casos un empleado es el encargado de realizarlas de manera periódica. Estas herramientas generan unos ficheros de Log donde se recoge la información de la copia y los resultados de la misma. El servicio que ofrece Ecocomputer S.L. consiste en el posterior visualizado de dichos ficheros y la notificación de incidencias a los clientes en caso de ser detectadas en los mismos.

El proceso requiere la participación de personal de ambas partes y el consumo de varias horas de trabajo para realizar un escaneo de los ficheros de resultados emitidos para la localización de incidencias y las razones por las que éstas fueron provocadas.

Se pretende con este proyecto, actualizar todo el proceso para integrar más a los usuarios y ofrecer un mejor servicio minimizando el trabajo requerido por el personal de la empresa solicitante.

En primer lugar se desea automatizar todo el proceso de realización de backups para que el cliente simplemente deba perder tiempo decidiendo cuándo y qué desea copiar. Para ello se presenta un intuitivo programador gráfico similar al programador de tareas del sistema operativo. Permitiendo en el mismo configurar la subida de las copias a un

servidor FTP de manera automatizada.

Por otra parte, se pretende formatear la información recogida en los ficheros de Log para que un usuario sin conocimientos informáticos pueda mantenerse informado del estado de sus copias y obtener información estadística y gráfica de las últimas realizadas. De igual forma, la empresa que proporciona el servicio no deberá consultar largos ficheros de información y podrá consultar rápidamente la información relevante en el mismo portal. Se busca así un doble propósito, por un lado simplificar la información al cliente para lograr su participación y por otro simplificar las tareas de control para el personal de Ecocomputer.

1.2. Estudio de la situación previa al proyecto

Al tratarse de un proyecto para una empresa real, se considera de mayor relevancia estudiar la organización de la misma y sus necesidades reales que el estudio de software o tecnologías que puedan asemejarse al proyecto tratado. Es por ello, que en este apartado se realizará una visión del cliente del proyecto.

Ecocomputer S.L. es una empresa de servicios informáticos con sede en Avilés (Asturias) con un amplio abanico de soluciones a disposición de sus clientes. Cuenta con dos centros de trabajo, en Asturias (con más de 500 metros cuadrados, data center, sala de formación, laboratorio electrónico) y en León. Su zona geográfica habitual cubre Asturias, Castilla y León y Cantabria.

En su plantilla cuenta con 16 empleados que le han permitido desarrollar proyectos muy variados, en algunos casos con una fuerte componente tecnológica y de innovación. Está fuertemente enfocada a la PYME, aunque también trabaja el sector de la administración y la gran cuenta.

Entre los servicios que ofrece destacan el desarrollo de aplicaciones y portales web a medida, hospedaje, registro y alojamiento de dominios para cliente, así como la implantación de sistemas de gestión empresarial. No obstante, también ofrece soluciones de mantenimiento para equipos de los clientes, dentro de este ámbito de negocio es donde se integrará el proyecto tratado.

La situación actual de la empresa en ciertos aspectos de su gestión interna está fundamentada en métodos y mecanismos un tanto anticuados hoy en día. Tratando de buscar una mejora que permita un menor desempeño de sus empleados, una mejor gestión de múltiples clientes y ofrecer soluciones más actuales y completas; surge el proyecto realizado.

Este proyecto será utilizado tanto por los empleados de Ecocomputer S.L. como por sus propios clientes. Fundamentalmente será diseñado pensando en los clientes de la empresa, quienes harán uso del mismo para mantener su información replicada y evitar pérdidas de datos, pero servirá además para que Ecocomputer realice una revisión periódica de las copias realizadas y despreocupe al usuario de dicha tarea.

1.3. Objetivos del proyecto

El proyecto tiene un objetivo claramente definido: aportar una solución automatizada y que requiera menor interacción que la solución implantada hasta el momento. Para conseguir ese propósito los objetivos que se deben cumplir serán los siguientes:

- **Aplicación cliente intuitiva y fácilmente manejable:** Se pretende ofrecer una interfaz amigable que un usuario no especializado utilice de manera cómoda. De nada sirve realizar una aplicación muy completa si su interfaz requiere un período de aprendizaje elevado y los clientes rechazan su utilización.
- **Visualizado de información de manera gráfica:** Se desea formatear la información recogida en los ficheros de Log para facilitar su comprensión y acelerar la lectura de los mismos. Se pretende ofrecer de manera visual y clara el resultado del backup pero sin descuidar u omitir detalles que permitan una mejor comprensión de las anomalías posibles.
- **Programación altamente personalizable:** El sistema ofrecido debe poder ser configurable para adaptarse a los requerimientos de los clientes. Se debe permitir un gran nivel de detalle para que los usuarios puedan programar sus copias de seguridad según sus necesidades.
- **Garantizar la fiabilidad y seguridad del proceso:** Se debe mantener la seguridad de todo el proceso requiriendo autenticación de los usuarios y permitiendo únicamente las acciones para las cuales se tengan permisos. Además, se debe garantizar la fiabilidad asegurando el correcto funcionamiento y la integridad de los datos copiados.
- **Soporte para copias de directorios y bases de datos SQL Server:** Debe permitirse la realización de copias de seguridad de ficheros, directorios completos y bases de datos SQL Server. Permittedose en todos los casos su subida a los servidores FTP, el automatizado del proceso y creándose los registros necesarios para su correcta evaluación posterior.
- **Correcta integración con los servicios prestados por la empresa:** Se debe garantizar en todo momento la correcta integración del proyecto desarrollado con las aplicaciones y la infraestructura actual. De igual manera, las aplicaciones cliente, deben respetar y funcionar de manera conjunta con los procesos y herramientas que los clientes disponen.
- **Uso de herramientas libres de restricciones o licencias:** El proyecto podrá hacer uso de bibliotecas o herramientas disponibles, siempre y cuando no se precise la contratación de una licencia y sean legales para su distribución comercial. De igual manera deberán elegirse los elementos gráficos y demás material a utilizar. No se puede olvidar que el proyecto es una aplicación empresarial y por lo tanto más restrictivo que si únicamente tuviera fines académicos.
- **Proyecto fácilmente mantenible:** Se debe tener en cuenta que el proyecto va a ser utilizado por multitud de clientes y por ello durante la vida del mismo, es más que razonable pensar que serán necesario actualizaciones y retoques para dotar de nuevas funcionalidades, adaptarlo a nuevos requisitos de clientes... Por ello, se considera de importancia realizar un producto fácilmente mantenible y escalable o modificable a las nuevas necesidades que puedan aparecer.

Capítulo 2

Aspectos teóricos

Dado que se trata de un proyecto académico, se considera esta sección importante para detallar algunos aspectos teóricos y demostrar algunos conocimientos adquiridos con la realización del mismo.

2.1. Backup (copia de seguridad)

2.1.1. Descripción

Un backup o copia de seguridad, dentro del ámbito de las tecnologías de la información e informática, es una copia de los datos originales que se realiza con el fin de disponer de un medio de recuperarlos en caso de su pérdida. Cuando se trabaja con información de importancia, es prácticamente una obligación realizar copias de la misma para evitar su pérdida ante situaciones adversas.

Por otra parte, dichas copias deben ser realizadas de manera periódica y rigurosa. Es por ello que se busca la automatización del proceso. Otro aspecto a tener en cuenta es que las copias de respaldo realizadas deben almacenarse en una ubicación física y lógica diferente a los datos originales, de nada vale tener replicados los datos en el mismo dispositivo de almacenamiento si dicho sistema deja de funcionar como debiera.

Dada la importancia de las copias de seguridad los sistemas cuentan con mecanismos para almacenar la situación actual de una base de datos, por citar un ejemplo, permitiendo una restauración de la misma con la copia realizada.

Resumiendo, para que una copia de seguridad sea adecuada debe permitir retomar el punto exacto de un sistema al momento en que fue realizada, debe estar actualizada y debe ser almacenada en un lugar seguro y alejada de los datos de origen.

2.1.2. Aplicación en el proyecto

Para poder entender la documentación de este proyecto, se debe tener en cuenta que en el ámbito del mismo, se utiliza el término «Backup» para hacer referencia a una programación concreta de copias de seguridad. De esta forma un backup será la programación de varias copias que deben realizarse en el mismo momento.

La razón por la que se ha utilizado esta terminología en el proyecto ha sido para mantener un mejor entendimiento con el cliente (en este caso el gerente de Ecocomputer) quien entendía como backup lo anteriormente mencionado. De esta forma las copias de seguridad concretas serán llamadas «tareas» o «jobs» y pertenecerán a los backups o programaciones.

Por otra parte, se desea detallar que el proyecto facilita la realización de copias de respaldo de la manera recomendada. Las copias pueden ser programadas y se realizarán de manera autónoma y periódica, se generaran archivos que permitan una restauración completa y además se facilita el almacenamiento de las copias en distintas ubicaciones permitiéndose la subida a servidores FTP.

2.2. Quartz Scheduler

2.2.1. Descripción

Quartz Scheduler¹ es un proyecto que facilita una interfaz que permite el uso de tareas programadas dentro de un programa.

Centrándonos en Java, engloba unas bibliotecas que permiten la ejecución de clases como hilos independientes y además lanzan los mismos según una programación dada. Se explicará más adelante en este mismo documento el funcionamiento del programador, el cual interpreta una sintaxis similar al administrador de procesos de UNIX «CRON».

Se ha elegido este proyecto para la implementación del cliente para Windows dada la facilidad de uso, la libertad en la licencia, la rigurosidad de la documentación y las opiniones encontradas sobre usuarios satisfechos con el mismo. Dado que nadie en la empresa había afrontado nunca un proyecto que requiriera tal funcionalidad, no se han tenido en cuenta preferencias de los empleados de la misma.

2.2.2. Aplicación en el proyecto

La funcionalidad más importante de la aplicación de escritorio para entornos Windows, es la de permitir una programación de copias de seguridad en el tiempo que permita su realización de manera autónoma y transparente para el usuario.

Se requiere dotar al sistema de un nivel de personalización elevado, pudiendo detallarse una programación diaria, semanal o mensual; una hora concreta a la que realizar las copias, unos intervalos de tiempo entre copias, etc. Dado que hay tantas posibles combinaciones, se consideraba necesario encontrar un formato capaz de definir las. Este problema fue solucionado al leer la documentación de Quartz y conocer su funcionamiento con formato CRON.

Formato CRON

Este formato permite mediante una secuencia de dígitos delimitar cuándo debe lanzarse una tarea. La secuencia se divide en siete campos: segundos, minutos, horas, día del mes, mes, día de la semana y año. La potencia de este formato no radica únicamente en que se utilicen dígitos en cada campo, ya que de esta forma solo podríamos delimitar una fecha concreta. Se permite dejar campos con valores «comodín» para abarcar un conjunto de fechas, horas, etc o bien utilizar limitadores temporales que permiten seleccionar períodos de tiempo abstractos (por ejemplo cada tres sábados).

Con este formato se consigue expresar cualquier programación de una tarea. Y dado que el proyecto de Quartz reconoce dicho formato, se considera muy práctico convertir los valores introducidos por el usuario y permitir así la comunicación con las bibliotecas.

Algunos ejemplos para este formato podrían ser:

- El día 7 de agosto de 2013 a las 12:57 horas: 00 57 12 7 8 * 2013
- Todos los días a las 12:00 horas: 00 00 12 * * * *
- Cada dos horas los lunes: * * */2 * * MON *

¹La página web oficial del proyecto es: <http://quartz-scheduler.org>.

2.3. Lenguaje unificado de modelado

2.3.1. Descripción

También conocido por sus siglas en inglés «UML», es el lenguaje para modelar software más conocido y empleado. Se trata de un lenguaje gráfico para visualizar, definir, detallar, construir y documentar gran parte de los aspectos de un sistema informático. UML cuenta con varios tipos de diagramas, que muestran y especifican diferentes aspectos de lo que se quiere representar.

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

Principalmente se utiliza UML para realizar diagramas de clases que detallan la estructura de un programa realizado con un lenguaje orientado a objetos, como es Java, de manera que se delimiten las clases implicadas, así como las relaciones entre estas.

2.3.2. Aplicación en el proyecto

El lenguaje UML ha sido empleado durante las fases de análisis y diseño de este proyecto, con el fin de definir y documentar todos los aspectos de la arquitectura completa de la aplicación y llegar a un mejor entendimiento de las necesidades de la empresa detallando sobre diagramas la visión de las partes.

Este documento contiene en las secciones correspondientes a Análisis y Diseño, diagramas UML para mejorar la comprensión de la arquitectura y la programación realizada en el proyecto.

Para la realización de los diagramas, se ha utilizado la herramienta Enterprise Architect en su versión 6, software propietario desarrollado por Sparx Systems.

2.4. Métrica versión 3

2.4.1. Descripción

Métrica² 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.

Está basada en el modelo de desarrollo ISO/EIC 12207 (Information Technology ? Software Life Cycle Processes) y en la norma ISO/EIC 15504 SPICE (Software Process Improvement And Assurance Standards Capability Determination).

La aplicación de Métrica Versión 3 permite alcanzar unos objetivos claros, como la correcta definición de sistemas de información, la dotación de un producto software satisfactorio, la mejora de la productividad de los departamentos de sistemas y tecnologías de la información y las comunicaciones, la facilitación de la comunicación entre los diferentes participantes en la producción del software y la consecución de la operación, mantenimiento y uso del producto software.

2.4.2. Aplicación en el proyecto

La versión 3 de la metodología Métrica será la empleada, al menos parcialmente, en la estructura de la presente documentación, organizando por capítulos y secciones determinados según aconseja la metodología. Como se dice, no se cumple exactamente para adecuar el presente documento a las exigencias de un trabajo fin de máster universitario.

²http://administracionelectronica.gob.es/?_nfpb=true&_pageLabel=P800292251293651550991&langPae=es&detalleLista=PAE_000000432

Pese a tratarse de un proyecto no dirigido a una administración pública, que son los organismos objetivo de Métrica, su aplicación se considera un manual de buenas prácticas.

2.5. Struts2

2.5.1. Descripción

Struts 2 es un framework para el desarrollo de aplicaciones web, propiedad de Apache, el cual hace que la implementación de las mismas sea más sencillo, más rápido, y con menos complicaciones. Su carácter de «software libre» y su compatibilidad con todas las plataformas en las que Java Enterprise está disponible lo convierten en una herramienta altamente disponible.

Struts 2 encaja dentro de la capa de presentación implementando el controlador del patrón de diseño MVC (Modelo Vista Controlador), y que podemos configurar de varias maneras; además proporciona algunos componentes para la capa de vista.

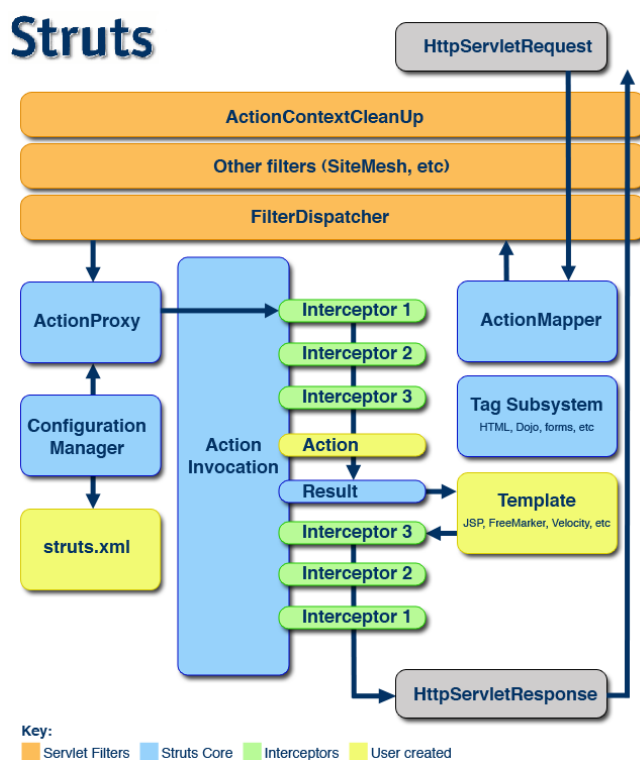


Figura 2.1: Funcionamiento general de struts2

El uso de un framework robusto permite dotar a la aplicación de cierta seguridad aprovechando los mecanismos que éste tiene integrados.

2.5.2. Aplicación en el proyecto

Se ha utilizado el framework Struts2 para realizar la aplicación Web. El uso de éste estuvo condicionado a intereses personales (experiencia previa con el mismo), así como intereses del cliente ya que los módulos con los que integrará este proyecto han sido realizados con el mismo framework.

Por otra parte, dado que el proyecto va a ser mantenido por empleados de la empresa que no tienen experiencia en Java, se ha realizado el proyecto haciendo uso de bibliotecas y frameworks que resulten conocidos por los mismos para facilitar su trabajo.

2.6. SQL Server 2005

2.6.1. Descripción

SQL Server 2005 es un sistema de gestión de bases de datos basado en el modelo relacional, utilizando el lenguaje T-SQL para realizar las consultas pertinentes. Entre sus características cabe destacar el soporte de transacciones, de procedimientos almacenados, la posibilidad de trabajar desde un entorno gráfico, realizar las consultas en modo cliente-servidor, concurrencia de usuarios y la posibilidad de administrar información de otros servidores de base de datos.

Como principal desventaja de este sistema de gestión de bases de datos lanzado en 1989 por Microsoft, nos encontramos que su instalación solo es posible bajo sistemas operativos Windows.

2.6.2. Aplicación en el proyecto

Debido a que es el sistema de gestión de base de datos impuesto por el cliente, se utilizará Microsoft SQL Server como base de datos, para el módulo Web. En ella se almacenará toda la información existente en la aplicación, así como otros datos que no serán utilizados por este proyecto, pero que son empleados por el resto de aplicaciones de la empresa.

La versión del sistema ha sido igualmente impuesto por la empresa cliente, dado que el servidor de despliegue del proyecto, cuenta con aplicaciones en producción funcionando con dicha versión y no se tiene prevista una actualización.

2.7. Apache Tomcat

2.7.1. Descripción

Apache Tomcat consiste en un servidor HTTP para diferentes plataformas (como UNIX, Windows . . .) que implementa el protocolo HTTP/1.1 e introduce el concepto de «sitio virtual». Funciona como un contenedor de servlets e implementa las especificaciones de las JavaServer Pages en su última versión.

Es empleado principalmente para enviar páginas web estáticas o dinámicas en la web a los clientes que las soliciten, ejecutando en el servidor las operaciones necesarias para su correcto funcionamiento. Permite ser instalado de manera local, sin acceso externo a la red, con el fin de probar las aplicaciones antes de ser expuestas en un entorno de explotación real.

Tomcat es mantenido por la Apache Software Foundation y voluntarios independientes. Se trata de software libre gratuito, y los usuarios pueden acceder a su código fuente y binarios en su totalidad, ya que se encuentra bajo una licencia Apache Software License. Fue lanzado al mercado en 1999.

2.7.2. Aplicación en el proyecto

Se trataba de una imposición del cliente la utilización de Apache Tomcat para desplegar tanto la aplicación Web como el servicio Web intercomunicador de módulos. Dado que iba a desplegarse en dicho servidor, para las pruebas y desarrollo se ha utilizado el mismo para evitar problemas de configuración al intercambiar la aplicación una vez desarrollada.

Pese a estar disponible una versión robusta más actualizada, la empresa trabaja con la versión 6 del mismo y es por ello que se ha utilizado esta.

Capítulo 3

Planificación del proyecto y resumen de presupuestos

3.1. Planificación

Es importante destacar que, como en una gran parte de los proyectos software, la planificación inicial de este proyecto no se corresponde enteramente con la realidad. La labor de planificar un proyecto es compleja y requiere de cierta experiencia, hay que tener en cuenta muchos factores que pueden modificar la planificación y contar con un importante plan de riesgos.

A la hora de afrontar este proyecto, no se contaba con ninguna experiencia del mundo laboral y por ello se han cometido errores de principiante. Para comprender mejor algunas de las razones de la mala planificación conviene destacar que la empresa contaba con una jerarquía no muy adecuada para estos casos. Los gerentes de la empresa actuaban de clientes del proyecto, pero éstos no mantenían reuniones con el desarrollador del proyecto, había unas terceras personas intermediarias que hacían llegar las solicitudes. Esto provocó malos entendidos entre las partes y posteriores cambios de planificación.

Por otra parte, durante el proyecto algunas personas de la empresa tuvieron vacaciones, lo que no solo impidió realizar reuniones y ralentizó el proyecto, sino que además las personas encargadas de tratar las dudas no compartían opiniones y se llegó a conclusiones erróneas que posteriormente fueron rectificadas.

Es conveniente destacar además, que el tiempo del proyecto se encontraba fijado desde el inicio y que puesto que había requisitos de poca importancia para la empresa, se han mitigado los retrasos, realizando modificaciones en los requisitos. De esta forma el proyecto ha sido posible en el tiempo previsto con las funcionalidades que deseaba la empresa.

Por citar algún cambio importante, en una primera planificación se contaba con realizar todo el sistema de gestión de usuarios de la aplicación, pero posteriormente se integró con otros proyectos de la empresa aprovechando la infraestructura montada y su gestión de usuarios.

Por todo esto se incluye en el presente apartado dos planificaciones: la inicial y la real, para que puedan ser contrastadas directamente. Se presentan mediante un diagrama de Gantt realizado con la herramienta «Microsoft Project», considerándose una forma representativa y de fácil lectura.

3.1.2. Planificación final del proyecto

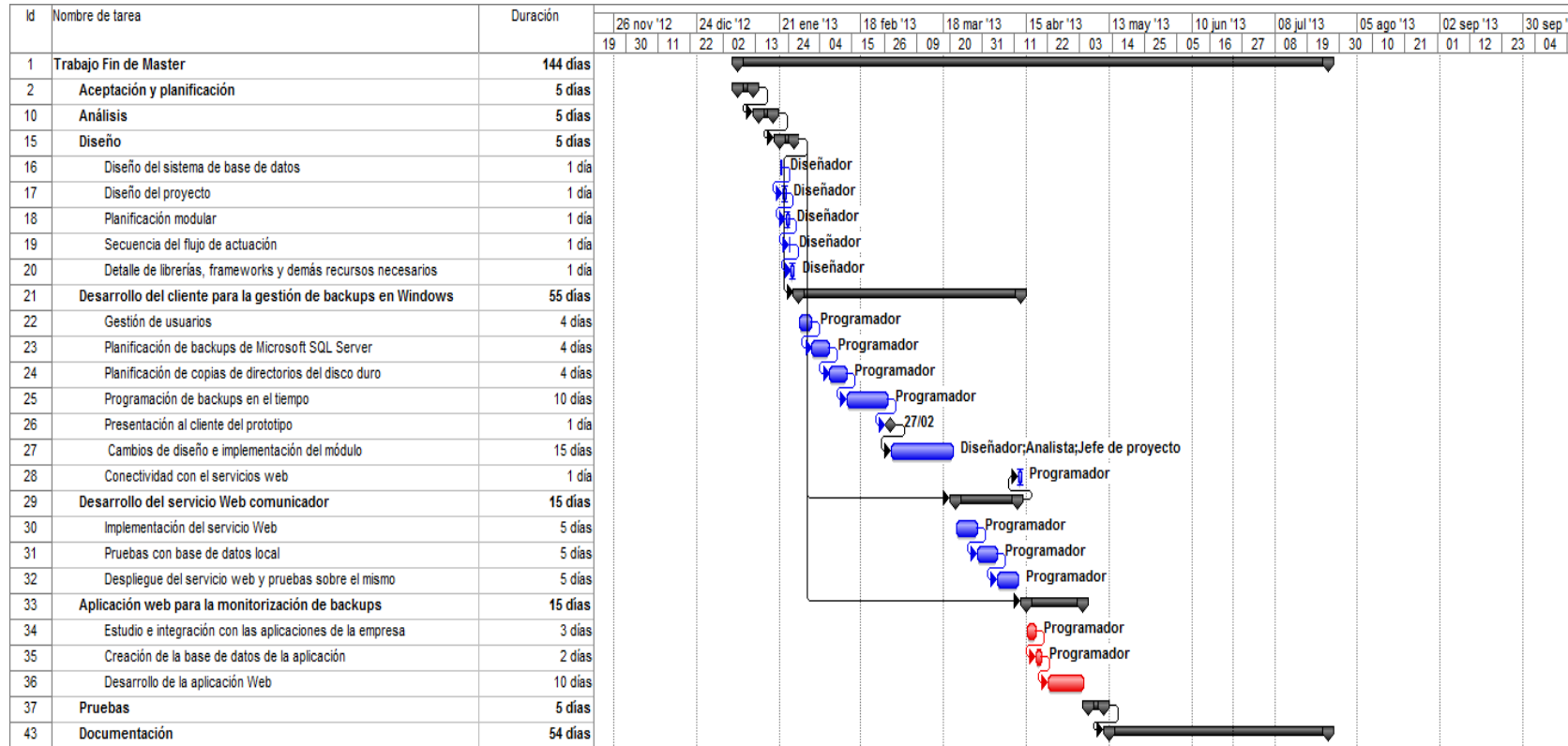


Figura 3.2: Planificación final del proyecto.

La razón de que se haya destinado tan poco tiempo a las fases iniciales del proyecto (análisis y diseño) es que algunas de las tareas necesarias para encarar el mismo habían sido realizadas con anterioridad por miembros de la empresa. Al empezar el proyecto se contaba ya con una especificación de requisitos, un análisis de actores involucrados, decisiones tomadas sobre tecnologías a utilizar . . . Si bien, se realizaron cambios sobre lo anterior, no requiso tanto esfuerzo como si se hubiera realizado desde cero.

3.1.3. Justificación de las diferencias

Pese a comentarse anteriormente en este apartado algunas de las razones que motivaron las diferencias con la planificación inicial, se pretende entrar algo más en detalle en esta sección.

- **Curva de aprendizaje más pronunciada:** Pese a tener experiencia en las tecnologías utilizadas, se ha encontrado dificultad para lograr el resultado deseado. En el caso de la aplicación de escritorio, se había trabajado previamente con frameworks para la realización de componentes e interfaces gráficas, sin embargo, no se habían realizado diseños con tablas ni componentes más complejos. Por otra parte, resultó compleja la parte del programador de backups, ya que nunca antes se había trabajado con algo similar.
- **Cambios de requisitos:** Al iniciar el proyecto, la empresa tenía unos requisitos ya marcados con los que se empezó a trabajar. Sin embargo, dichos requisitos fueron cambiando durante la vida del proyecto. Algunos cambios fueron provocados por el desarrollo de proyectos de manera paralela que finalmente fueron integrados con el aquí tratado, mientras que otros surgieron de ideas aportadas por diferente personal que no se encontraba en la toma de requisitos inicial.
- **Modificación del alcance del proyecto:** Relacionado con lo anterior, nos encontramos con modificaciones de alcance. Mientras se estaba realizando el proyecto que nos ocupa, se realizaban otros con funcionalidades distintas pero destinados a los mismos clientes, por ello se decidió una vez avanzados los mismos, realizar una integración. Este cambio provocó que la gestión de usuarios fuera centralizada para todos los proyectos.

Los retrasos sufridos se vieron reflejados de manera drástica en la planificación real, pese a que la cantidad de horas no varíe demasiado, fue necesario eliminar funcionalidad considerada secundaria para dar cabida a los cambios y retrasos surgidos.

De este modo se desechó:

- **Cliente para entornos GNU/Linux:** Se deseaba en un principio realizar un cliente para entornos GNU/Linux pero se desechó al no ser necesario dada la clientela de la empresa. Por otra parte, se consideró que no tenía demasiado sentido al realizarse únicamente copia de directorios debido a que el sistema de bases de datos elegido (Microsoft SQLServer 2005) es exclusividad del sistema operativo Windows.
- **Módulo de gestión de usuarios:** Tal como se ha comentado, se decidió integrar varios proyectos y por ello, utilizar un único módulo de gestión de usuarios. Se evitó así el desarrollo de dicho módulo pero fueron necesarias tareas de integración con el mismo.
- **Visualizador de resultados en dispositivos móviles:** Pese a no ser un requisito inicial, durante el proyecto surgió la idea de realizar una ampliación orientada a los dispositivos móviles. Debido a la falta de tiempo se desestimó, puesto que los navegadores web de los dispositivos móviles modernos no difieren mucho de los de escritorio, y la aplicación podría ser utilizada con cierta normalidad.

3.2. Resumen del presupuesto

Debido a que el proyecto se ha realizado con y para una empresa de desarrollo informático, se ha creído conveniente realizar el presupuesto de la manera más aproximada posible a los estándares de dicha empresa.

Considérese que la empresa no realiza distinciones entre las diferentes fases del trabajo realizado, si no que el precio por hora es el mismo para todos. Igualmente, en dicho precio se consideran todos los gastos directos e indirectos que el empleado ocasiona durante sus horas de trabajo (sueldo, seguro médico, agua, luz, mantenimiento, uso de los equipos y licencias de software ...).

El número de horas es el estimado para la elaboración del proyecto. Este valor ha sido estimado igualmente por la empresa, ya que como se ha comentado anteriormente, fue el personal de la misma quien realizó el estudio de viabilidad del proyecto.

Concepto	Horas	P. Hora	TOTAL
<i>Desarrollo de la Aplicación</i>	515	39,00 €	20085,00 €
		Subtotal	20085,00 €
		IVA(21 %)	4217,85 €
		TOTAL	24302.85 €

Cuadro 3.1: Tabla de presupuesto del proyecto

Para el cálculo de horas, Ecocomputer S.L. realizó la siguiente estimación temporal. Como se ha dicho anteriormente, se han realizado numerosas modificaciones en los requisitos y alcance del proyecto y la siguiente estimación no coincide con la realidad. No obstante, dado que el plazo del proyecto venía fijado desde el inicio para cumplir los plazos de la Universidad, el presupuesto es acertado para el producto final, ya que pese a cambiar las labores realizadas, el número de horas ha sido equivalente. La segunda tabla muestra una aproximación de horas más realista una vez aplicados los cambios en el proyecto.

Tarea	Horas
<i>Modelado de la base de datos</i>	10h
<i>Tareas de diseño</i>	5h
<i>Aplicación cliente para la gestión de backups en Windows</i>	200h
<i>Aplicación cliente para la gestión de backups en GNU/Linux</i>	60h
<i>Servicio Web para complementar las herramientas de backup</i>	30h
<i>Aplicación Web para la monitorización de backups</i>	110h
<i>Realización de pruebas</i>	30h
<i>Análisis y reuniones en Ecocomputer S.L.</i>	30h
<i>Documentación técnica y manuales de usuario</i>	40h

Cuadro 3.2: Desglose de horas para el cálculo del presupuesto. Versión inicial

Los principales cambios entre las tablas, se deben a la modificación de los requisitos del sistema. Por un lado, fueron necesarios retoques en el diseño de la base de datos, aumentando el tiempo destinado a dicha tarea. De igual forma, las tareas de diseño se vieron aumentadas al ser necesario realizar una revisión y un segundo diseño para recoger los cambios.

Las modificaciones en los requisitos fueron aplicables a la herramienta cliente para plataformas Windows, por lo que el desarrollo de esta también sufrió retrasos. Sin embargo, como se ha comentado anteriormente, estos retrasos fueron compensados con la simplificación de otras tareas de menor importancia y la eliminación de ciertas funcionalidades.

Tarea	Horas
<i>Modelado de la base de datos</i>	15h
<i>Tareas de diseño</i>	15h
<i>Aplicación cliente para la gestión de backups en Windows</i>	290h
<i>Servicio Web para complementar las herramientas de backup</i>	30h
<i>Aplicación Web para la monitorización de backups</i>	50h
<i>Realización de pruebas</i>	30h
<i>Análisis y reuniones en Ecocomputer S.L.</i>	40h
<i>Documentación técnica y manuales de usuario</i>	45h

Cuadro 3.3: Desglose de horas para el cálculo del presupuesto. Versión final

Capítulo 4

Análisis

Tal como se ha comentado anteriormente, el proyecto ha sufrido importantes cambios durante su desarrollo. Dado que dichos cambios afectaron al alcance del mismo y a los requisitos, se presenta en este apartado el análisis final, pero se ha querido incluir el primer análisis en la documentación a modo de apéndice.

4.1. Definición del sistema

4.1.1. Determinación del alcance del sistema

El proyecto que nos ocupa debe cubrir unas necesidades concretas del cliente. Se pretende desarrollar una infraestructura completa que permita realizar copias de seguridad, programar las mismas y llevar un seguimiento de las ya realizadas desde cualquier dispositivo con conexión a Internet.

Para conseguir lo requerido se debe realizar un módulos cliente para entornos Windows, un servicio Web que permita la actualización de los datos almacenados en un servidor y por último, una aplicación Web de visualización de resultados.

Para el caso de la aplicación de escritorio de entornos Windows, debe permitirse la realización de backups de bases de datos Microsoft SQL Server 2005, así como la copia de directorios o ficheros concretos. Debe presentarse además un panel de programación que permita automatizar las copias.

La aplicación Web deberá ser un monitor de resultados de los backups realizados hasta el momento. Deberá estar actualizado y protegido mediante clave de acceso. El sistema de gestión de usuarios será heredado de otras aplicaciones de la empresa, y deberá integrarse con el proyecto realizado.

Por último, para conseguir el correcto funcionamiento de la infraestructura y evitar el acceso directo al sistema de base de datos de la aplicación Web, se deberá realizar un servicio Web que permita actualizar la información sobre los backups realizados por los usuarios.

4.2. Requisitos del sistema

4.2.1. Obtención de los requisitos del sistema

Aplicación cliente

Se requiere una aplicación de escritorio desarrollada en Java que permita programar la ejecución de copias de seguridad de directorios del equipo en el que se encuentre instalada y en ejecución, así como de backups de sistemas de bases de datos Microsoft SQL Server. Se busca una herramienta de sencillo manejo, que realice las funciones de programador de copias de seguridad, ejecutor de las mismas, envíe a servidores FTP de los resultados

(en caso de ser solicitado por el usuario) y actualización de un histórico con los resultados obtenidos en la realización de las mismas.

Dicha aplicación debe estar protegida del personal no autorizado y por ello se requerirá usuario y contraseña para interactuar con la misma. Dicha información debe ser solicitada al usuario en la primera ejecución del software y será almacenada debidamente en la base de datos local.

Código	Nombre requisito	Descripción del requisito
R1.1	Insertar usuario	La primera ejecución de la aplicación debe requerir un registro por parte del usuario, la información será utilizada posteriormente para autenticar al mismo en el sistema y permitir su interacción.
R1.2	Autenticación	La herramienta debe requerir la autenticación del usuario y no permitir la interacción de usuario anónimos.
R2.1	Copias de directorios y de ficheros	La aplicación debe estar dotada de la funcionalidad suficiente que permita realizar copias de seguridad de ficheros y directorios completos del sistema en el que se encuentra en ejecución.
R2.2	Copias de bases de datos Microsoft SQL Server	La herramienta debe permitir la realización de copias de seguridad de sistemas de bases de datos Microsoft SQL Server que se encuentren desplegados en el equipo cliente en el que se encuentra la aplicación en ejecución.
R3.1	Subida a servidores FTP	Se debe permitir que las copias de seguridad sean enviadas a un servidor FTP detallado por el usuario, así como mantener un registro de los resultados del envío.
R4.1	Programación detallada y sencilla	La herramienta debe proporcionar una interfaz que permita la programación de backups en el tiempo. Debe ser altamente personalizable llegando a un nivel de detalle similar al programador de tareas de Microsoft Windows.
R5.1	Conexión con el servicio Web	Los datos registrados por la aplicación sobre las copias de seguridad realizadas, anomalías en su ejecución y envío de resultados a servidores FTP, deben ser recibidas por el servicio Web y almacenadas en la base de datos que consultará la aplicación Web.
R6.1	Ejecución de backups de manera independiente del usuario	Una vez programados las copias de seguridad, llegado el momento de su ejecución no debe ser necesaria la interacción con el usuario, debe realizarse de manera transparente para el mismo y automatizando el proceso.

Cuadro 4.1: Tabla de requisitos de la aplicación de escritorio cliente

En la tabla superior se recogen los **requisitos funcionales** de la aplicación de escritorio cliente, a continuación se detallan algunos requisitos añadidos que debe cumplir el sistema:

1. **Requisitos de usuario:** El usuario debe recordar la información de autenticación introducida en el primer registro de la aplicación. También debe ser consciente de la necesidad de la realización de copias de seguridad y realizar una programación de las mismas eficaz.
2. **Requisitos tecnológicos:** El programa debe integrarse con las herramientas en ejecución de los clientes. Dado que va a ser instalada en diferentes equipos y con diferentes configuraciones, no debe ser restrictiva ni hacer uso de recursos que no puedan ser compartidos. Por ejemplo no deberá hacer uso de puertos del sistema que puedan ser empleados por otras aplicaciones.
3. **Requisitos de usabilidad:** La interfaz debe ser amigable y sencilla de utilizar. Al no estar destinada a personal relacionado con el mundo de la informática, se deben evitar mecanismos de configuración que requieran conocimientos o destrezas tecnológicas.
4. **Requisitos de seguridad:** Las contraseñas almacenadas en la base de datos deberán ser tratadas para evitar su lectura inmediata. La contraseña introducida en el registro será debidamente cifrada siguiendo un algoritmo irreversible, mientras que las contraseñas que permiten la conexión a servidores FTP o a los sistemas de bases de datos Microsoft SQL Server se encontrarán codificadas permitiéndose su recuperación pero evitando su lectura mediante consultas a la base de datos utilizada por la herramienta.
5. **Requisitos de tiempo de respuesta:** La aplicación debe ser capaz de responder a las necesidades programadas por el usuario. Dado que pueden programarse copias de seguridad a nivel de segundos, es importante garantizar la ejecución en tiempo de dichos procesos para cumplir la programación solicitada. Es por ello que la aplicación lanzará diferentes hilos encargados de llevar a cabo procesos independientes. La herramienta será instalada principalmente en servidores con niveles de procesamiento elevados y grandes espacios de memoria, por lo que el uso de hilos no condicionará el rendimiento de la máquina en exceso.

Portal Web

Para no repetir lo mismo que en las secciones previas sobre los cambios producidos en los requisitos, en esta sección se omitirán las opiniones iniciales centrándose la misma en el análisis del producto final.

Se requiere una aplicación Web sencilla y visual, que permita conocer el estado de los backups realizados hasta el momento por cada cliente. Será un sistema de lectura de datos, una monitorización de las copias realizadas y el resultado de las mismas.

Se recogen a continuación, los requisitos que debe cumplir este sistema:

Código	Nombre del requisito	Descripción del requisito
R1.1	Visualización actualizada de los resultados de backups	Debe mostrarse de manera sencilla y actualizada, la información relativa a los resultados de los backups realizados. En caso de error en alguno de ellos debe especificarse la razón del fallo.
R1.2	Visualización actualizada de los resultados de las tareas	Al igual que se muestran los resultados resumidos de los backups, se debe permitir un desglose en tareas para saber cuales fueron realizadas correctamente y cuales no.

continúa en la próxima página

Código	Nombre del requisito	Descripción del requisito
R2.1	Integración con la plataforma Web de la empresa	La aplicación Web debe integrarse con la plataforma de la empresa. Se debe mantener el sistema de gestión de usuarios, permitiendo a un usuario autenticado en la misma trabajar con normalidad en la aplicación Web que nos ocupa.

Cuadro 4.2: Tabla de requisitos de la aplicación Web

Como requisitos no funcionales podríamos detallar los siguientes:

1. **Look and feel:** Se debe respetar la apariencia de la plataforma Web de Ecocomputer. Pese a deber acatar los bocetos realizados por el diseñador gráfico de la empresa, se contaba con relativa libertad. Sin embargo, dado que se va a integrar el proyecto con una plataforma ya creada con otros proyectos, se debe mantener un estilo conjunto para evitar dar sensación de divisiones.
2. **Accesible a todos los navegadores Web:** Siempre que se realiza un proyecto Web se debe tener muy presente el numeroso abanico de navegadores y dispositivos con acceso a Internet que pueden llegar a visitar el sitio. Se pretende realizar una aplicación cumpliendo los estándares, de tal manera que se pueda visualizar en la mayor parte de sistemas.

Web Service

El servicio Web será un sistema sencillo y su única finalidad será comunicar las aplicaciones de los clientes con la base de datos del servidor de la empresa. Se debe garantizar la seguridad de la comunicación, para evitar que cualquiera pueda introducir datos en el sistema conectándose al servicio.

El servicio debe comprobar los credenciales del usuario que se conecta al mismo, para comprobar si pueden almacenarse los datos que transporta. Por otra parte, se debe notificar del éxito o el fracaso en la actualización.

4.2.2. Identificación de actores del sistema

Aplicación cliente

Esta aplicación consiste en una herramienta de programación de backups, esta tarea suele estar destinada a un único individuo dentro de una empresa. Además, los requisitos exigen la autenticación del usuario a la hora de programar nuevos backups o consultar información de los mismos.

Es por ello que solo un actor primario interactuará con este subsistema. Podemos denotarlo como administrador, ya que en muchas ocasiones suele ser el administrador del sistema el encargado de realizar las copias que considere necesarias. La aplicación podrá tener más de un usuario registrado, pero el rol que desempeñen dichos usuarios será el mismo en todos los casos.

Como actor secundario, se encontrará el servicio web encargado de modificar la base de datos del servidor, para mantener actualizada la información del portal web. La herramienta se conectará al mismo para enviar información nueva periódicamente y mantener, de esta forma la plataforma web actualizada.

Portal Web

En este módulo, se distinguen dos usuarios primarios claramente estipulados. En primer lugar un cliente de la empresa y de la aplicación, que se conectará a la misma para conocer el resultado de sus backups. Por otra parte, tendremos un administrador de la empresa que podrá visualizar la información de todos los clientes y así poder asesorar sobre posibles anomalías o procedimientos incorrectos.

Se debe tener en cuenta además, como actor secundario del sistema, el servicio Web encargado de mantener actualizada la base de datos de la cual se nutre la aplicación.

Servicio Web

El servicio Web tendrá como actores los dos módulos que comunica. Pese a trabajar con usuarios, la gestión de los mismos la realizan las otras partes. El servicio únicamente recibe la información de los clientes de escritorio y se la comunica a la base de datos del servidor.

4.2.3. Especificación de casos de uso

Aplicación cliente

La aplicación de escritorio tendrá los siguientes casos de uso:

Código	Caso de uso
C1.1	Registro del usuario en el sistema, al iniciar la aplicación por primera vez, deben solicitarse datos de acceso como usuario y contraseña.
C1.2	Autenticación del usuario para acceder al sistema con sus datos introducidos en el registro.
C2.1	Creación de una tarea. El usuario puede crear una tarea consistente en un origen y un destino de la copia. Esta tarea podrá ser lanzada manualmente una vez creada, o bien, añadida a una nueva programación de backup.
C2.2	Editar una tarea creada. El usuario podrá modificar tareas previamente creadas.
C2.3	Eliminar una tarea. El usuario podrá eliminar tareas previamente creadas, siempre que éstas no pertenezcan a una programación de backup.

continúa en la próxima página

Código	Caso de uso
C2.4	Lanzar una tarea. El usuario podrá lanzar la ejecución de una tarea manualmente, ésta se realizará una única vez en el momento de ser lanzada.
C3.1	Programación de un backup para su realización automatizada y autónoma. El usuario establece cuando desea que se realicen los backups para que se lancen de manera transparente al mismo llegado el momento de su ejecución.
C3.2	Consultar los backups programados. Acceder a la aplicación para listar los backups programados y conocer detalles de alguno de ellos.
C3.3	Eliminar un backup programado. Finalizar la programación de un backup para evitar su ejecución futura.
C4.1	Consultar el registro de Log de la aplicación. Consultar los últimos registros de Log de la aplicación, ordenados por fecha, haciendo uso del visualizador gráfico integrado.
C4.2	Consultar los resultados de los backups realizados. Visualizar la información relativa a los backups ya realizados para conocer sus resultados satisfactorios o erróneos.
C5.1	Agregar información de conexión con una base de datos. Introducir la información de conexión con la base de datos, para posteriormente realizar copias de la misma simplemente referenciándola.
C5.2	Agregar información de conexión con un servidor FTP. Introducir la información de conexión con un servidor FTP, para posteriormente subir las copias al mismo simplemente referenciándolo.
C5.3	Probar la conexión con un servidor FTP. Realizar una prueba de conexión para saber si se encuentran disponibles los servidores FTP introducidos al sistema.
C5.4	Editar la información de una conexión FTP. Modificar los valores introducidos para una conexión.
C5.5	Eliminar la información de una conexión FTP. Eliminar los valores introducidos para una conexión.
C5.6	Probar la conexión con un sistema de base de datos. Realizar una prueba de conexión para saber si se encuentra disponible una base de datos introducida en el sistema.
C5.7	Editar la información de una conexión con un sistema de base de datos. Modificar los valores introducidos para una conexión.
C5.8	Eliminar la información de una conexión con un sistema de base de datos. Eliminar los valores introducidos para una conexión.
C6.1	Actualizar la información del servidor. Conectar con el servicio Web para mantener la información del servidor actualizada.

Cuadro 4.3: Tabla de casos de uso de la aplicación de escritorio cliente

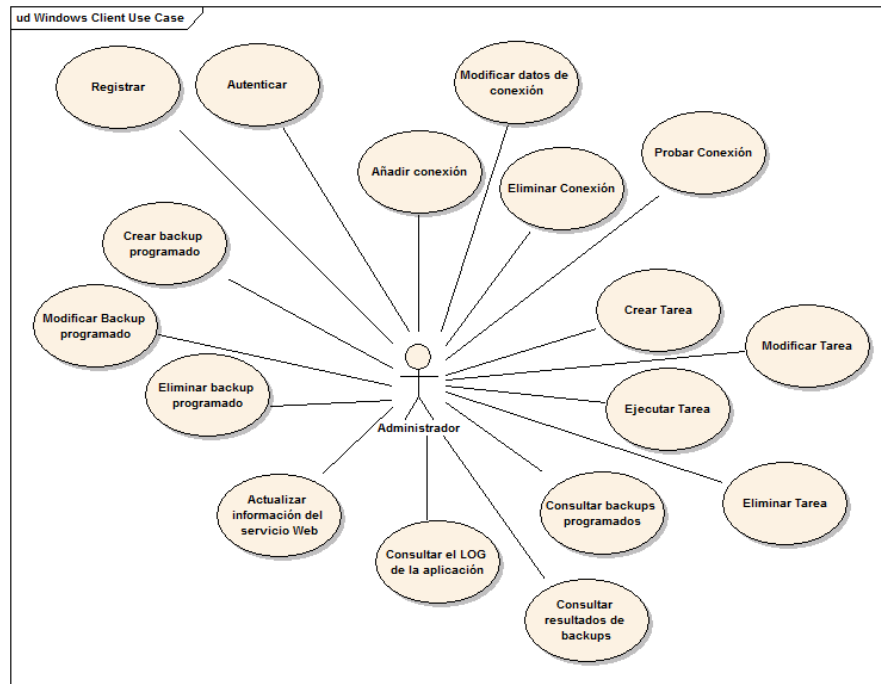


Figura 4.1: Diagrama de casos de uso de la aplicación cliente en Windows

Aplicación Web

La aplicación de Web tendrá los siguientes casos de uso:

Código	Caso de uso
C1.1	Ver información de todos los backups realizados por los clientes. Un usuario con el rol de administrador podrá visualizar los resultados de los backups realizados por todos los clientes del sistema.
C1.2	Ver información detallada de un backup de un cliente. Un usuario con el rol de administrador podrá visualizar los resultados de las tareas pertenecientes a un backup realizado, para conocer en detalle cuales fueron realizadas correctamente y cuales no.
C2.1	Ver información de los backups propios realizados. Un usuario autenticado en el sistema deberá poder visualizar la información relativa a los backups realizados en todos sus equipos.
C2.2	Ver información detallada de un backup propio realizado. Un usuario autenticado en el sistema deberá poder visualizar la información detallada sobre las tareas de un backup realizado, con la finalidad de discernir que procesos se realizaron correctamente y cuales no. Detallando el error producido en este último caso.

Cuadro 4.4: Tabla de casos de uso de la aplicación de Web

Se han eliminado los casos de uso relativos al registro, autenticación, generación de permisos para un usuario . . . debido a que la gestión de usuarios se encuentra realizada y simplemente va a ser integrada con el proyecto que nos ocupa. En los casos de uso iniciales se encontraban numerosas tareas relativas a dicho módulo, ya que en un principio se iba a realizar una gestión de usuario individual para el proyecto que nos ocupa.

Dado que un administrador puede ser además un cliente del sistema, los casos de uso de los clientes son compartidos por el usuario con permisos.

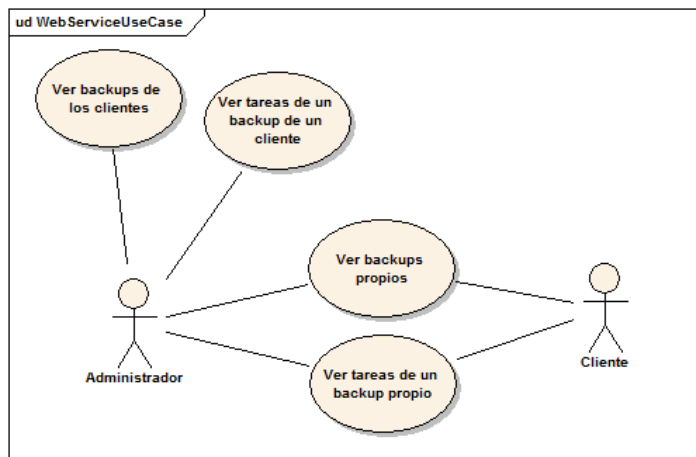


Figura 4.2: Diagrama de casos de uso de la aplicación Web

4.3. Identificación de los subsistemas en la fase de análisis

El proyecto que nos ocupa tiene tres subsistemas claramente definidos desde los requisitos del propio cliente. Tal como se ha venido comentando en las secciones anteriores, contará de una aplicación cliente que debe ejecutarse en todos los equipos que quieran hacer uso de la infraestructura, un servicio web desplegado en los servidores de la empresa ofertadora (Ecocomputer S.L.) y una aplicación web para la visualización el resultado de los backups realizados, que se encontrará igualmente desplegada en los servidores de la empresa proveedora del servicio.

En la fase de diseño se detallará cada subsistema descomponiéndolo en diferentes módulos con finalidades claras. Así, en el cliente de escritorio, por ejemplo será necesario establecer un módulo encargado de la autenticación de los usuarios, otro del registro de nuevos clientes . . .

4.3.1. Descripción de los subsistemas

Aplicación cliente

La aplicación cliente corresponde a una herramienta de escritorio, que permitirá programar copias de seguridad en el tiempo. Por ello, contará con un panel de configuración en el que el usuario detalle los elementos que deben ser copiados (ya sean copias de bases de datos SQL Server o directorios del propio sistema), la información necesaria para subirlos a un servidor FTP, en caso de considerarse necesario, o el directorio donde deben ser almacenadas las copias; así como, la indicación temporal que indique cuándo deben realizarse las mismas.

Por otra parte dicho subsistema contará con un módulo de persistencia implementado sobre una base de datos SQLite (a petición del cliente) que almacenará las programaciones creadas. Por último, la aplicación debe realizar las copias llegado el momento, comprimiendo los directorios para ser enviados a los servidores FTP indicados, o bien, almacenando una réplica en el directorio oportuno.

Todas las operaciones deben quedar registradas en un Log, así como actualizar un historial de resultados de los backups realizados.

Web Service

Se tratará de un servicio web encargado de almacenar la información que le llega de las aplicaciones de escritorio instaladas en los sistemas de los clientes. Su funcionalidad será sencilla pero será necesario garantizar la seguridad en las comunicaciones para evitar almacenar datos incorrectos o aceptar conexiones de sistemas no autorizados.

Portal Web

Deberá realizarse una aplicación Web de consulta que permita visualizar el historial de los backups realizados, así como información de las tareas que los componen. Se deberá integrar dicha aplicación, en una plataforma desarrollada por la empresa, que presenta un módulo de gestión de usuarios de utilidad para el proyecto. Por otra parte, se deberá comprobar si el usuario autenticado tiene permisos de administrador, en cuyo caso debe poder visualizar la información relativa a todos los backups de los clientes.

4.3.2. Descripción de los interfaces entre subsistemas

Las comunicaciones entre los módulos se realizan mediante el servicio Web. Se tenía libertad para implementar un servicio REST o SOAP, pero se ha decidido realizar uno de los últimos, debido a la sencillez de uso utilizando herramientas de generación de código como las que proporciona Eclipse.

De esta forma, las aplicaciones cliente se conectan al servicio SOAP, mediante el WSDL publicado. Así, se le envía al mismo información relativa al usuario que se conecta y los historiales de los backups realizados que no han sido sincronizados hasta ese momento.

El servicio se conectará a la base de datos local de la empresa, que será la consultada por la aplicación Web. De esta forma la información se encuentra actualizada en la Web.

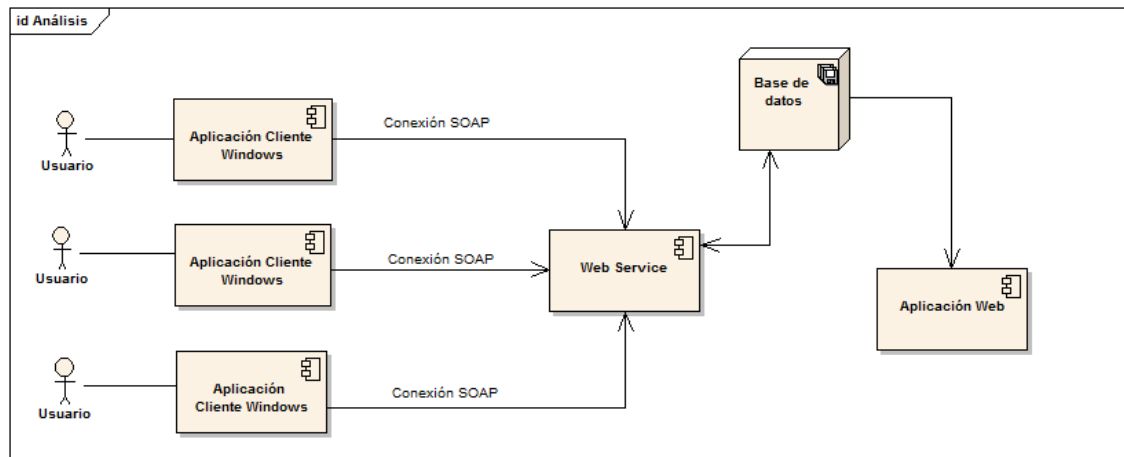


Figura 4.3: Diagrama relación entre los sistemas implicados en el proyecto.

4.4. Diagrama de clases preliminar del análisis

Tal como se ha comentado anteriormente, se han producido importantes cambios durante el desarrollo del proyecto. Por esta razón, las clases identificadas en esta fase del proyecto y las relaciones entre las mismas, no se corresponden al resultado final. El principal cambio, radica en la aplicación de escritorio, en la cual se consideraba al iniciar el proyecto que las tareas pertenecían a una programación de backup y no tenían sentido de manera aislada. Posteriormente, cambios en los requisitos, obligaron a realizar modificaciones en la arquitectura y eliminar el requerimiento de que cada tarea perteneciera a un backup y únicamente a uno.

4.4.1. Diagrama de clases

Aplicación cliente de escritorio para Windows

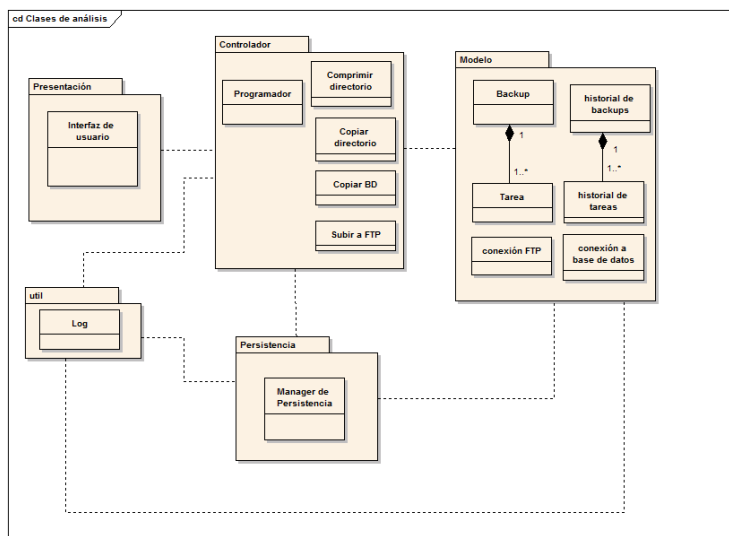


Figura 4.4: Diagrama de clases de análisis, aplicación de escritorio

Aplicación Web

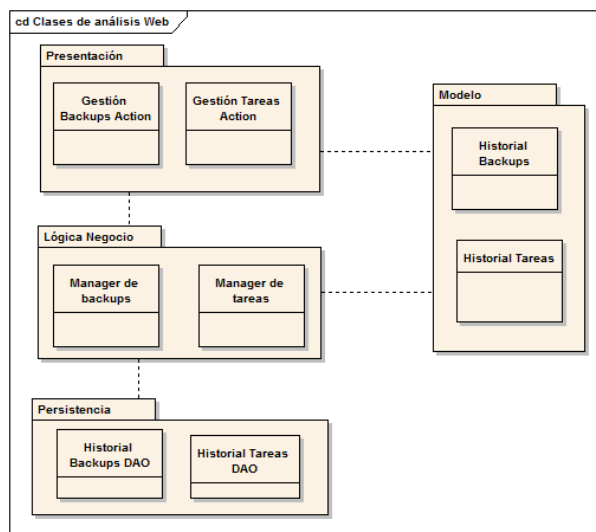


Figura 4.5: Diagrama de clases de análisis, aplicación Web

Servicio Web

Debido a la simpleza del servicio Web no se considera necesario realizar un diagrama de clases para esta fase. Simplemente será necesario tener una clase Serializable que contenga los datos que son transmitidos con las aplicaciones, y un gestor de base de datos para almacenar los mismos.

4.4.2. Descripción de las clases

Aplicación cliente para Windows

Backup
DESCRIPCIÓN
Esta clase representará una programación de copias de seguridad.
RESPONSABILIDADES
Será una clase del modelo sin funcionalidad propia.
ATRIBUTOS PROPUESTOS
<ul style="list-style-type: none"> ▪ Nombre del Backup: Nombre del backup para ser representado. ▪ Momento de Ejecución: Se almacenará en formato CRON la programación del backup. De esta forma se determinará cuándo debe ser lanzado. ▪ Lista de tareas: Se contará con una lista de tareas para realizar, cuando se ejecute el backup se realizarán las copias estipuladas en dicha lista.
MÉTODOS PROPUESTOS
<ul style="list-style-type: none"> ▪ Getters y setters: Puesto que se trata de una clase del modelo, no presentará lógica y únicamente tendrá selectores y modificadores del estado del objeto.

Tarea
DESCRIPCIÓN
Esta clase representará copia de seguridad concreta, marcando un origen y un destino para la misma.
RESPONSABILIDADES
Será una clase del modelo sin funcionalidad propia.
ATRIBUTOS PROPUESTOS
<ul style="list-style-type: none"> ▪ Nombre de tarea: Nombre de la tarea para ser representada. ▪ Origen: Origen de la copia, puede ser una base de datos, un directorio o un fichero. ▪ Destino: Destino de la copia, puede ser un servidor FTP o un directorio local. ▪ Nombre destino: Nombre con el que se quiere almacenar la copia.
<i>continúa en la próxima página</i>

<u>Tarea</u>
MÉTODOS PROPUESTOS
<ul style="list-style-type: none"> ▪ Getters y setters: Puesto que se trata de una clase del modelo, no presentará lógica y únicamente tendrá selectores y modificadores del estado del objeto.

<u>Conexión FTP</u>
DESCRIPCIÓN
Esta clase representará una conexión FTP.
RESPONSABILIDADES
Será una clase del modelo sin funcionalidad propia.
ATRIBUTOS PROPUESTOS
<ul style="list-style-type: none"> ▪ Nombre de la conexión: Nombre de la conexión para ser representada. ▪ Dirección: Dirección del servidor FTP. ▪ Usuario: Nombre de usuario con permisos en el servidor FTP. ▪ Contraseña: Contraseña del usuario anterior.
MÉTODOS PROPUESTOS
<ul style="list-style-type: none"> ▪ Getters y setters: Puesto que se trata de una clase del modelo, no presentará lógica y únicamente tendrá selectores y modificadores del estado del objeto.

<u>Conexión base de datos</u>
DESCRIPCIÓN
Esta clase representará una conexión con una Base de Datos.
RESPONSABILIDADES
Será una clase del modelo sin funcionalidad propia.
ATRIBUTOS PROPUESTOS
<ul style="list-style-type: none"> ▪ Nombre de la conexión: Nombre de la conexión para ser representada. ▪ Dirección: Dirección de la base de datos. ▪ Usuario: Nombre de usuario con permisos en la base de datos. ▪ Contraseña: Contraseña del usuario anterior. ▪ Nombre de la base de datos: Nombre de la base de datos a copiar.
<i>continúa en la próxima página</i>

<u>Conexión base de datos</u>
MÉTODOS PROPUESTOS
<ul style="list-style-type: none"> ▪ Getters y setters: Puesto que se trata de una clase del modelo, no presentará lógica y únicamente tendrá selectores y modificadores del estado del objeto.

<u>Historial de Backups</u>
DESCRIPCIÓN
Esta clase representará los resultados de un backups realizado.
RESPONSABILIDADES
Será una clase del modelo sin funcionalidad propia.
ATRIBUTOS PROPUESTOS
<ul style="list-style-type: none"> ▪ Backup: Backup al que hace referencia el histórico. ▪ Resultado: Resultado del mismo. ▪ Resultado de la copia: Resultado en la realización de la copia. ▪ Resultado del envío: Resultado en la realización del envío.
MÉTODOS PROPUESTOS
<ul style="list-style-type: none"> ▪ Getters y setters: Puesto que se trata de una clase del modelo, no presentará lógica y únicamente tendrá selectores y modificadores del estado del objeto.

<u>Historial de tareas</u>
DESCRIPCIÓN
Esta clase representará los resultados de una tarea realizada.
RESPONSABILIDADES
Será una clase del modelo sin funcionalidad propia.
ATRIBUTOS PROPUESTOS
<ul style="list-style-type: none"> ▪ Tarea: Tarea a la que hace referencia el histórico. ▪ Resultado: Resultado de la misma. ▪ Resultado de la copia: Resultado en la realización de la copia. ▪ Resultado del envío: Resultado en la realización del envío.
<i>continúa en la próxima página</i>

<u>Historial de tareas</u>
MÉTODOS PROPUESTOS
<ul style="list-style-type: none"> ▪ Getters y setters: Puesto que se trata de una clase del modelo, no presentará lógica y únicamente tendrá selectores y modificadores.

<u>Manager FTP</u>
DESCRIPCIÓN
Esta clase realizará las subidas de copias a servidores FTP.
RESPONSABILIDADES
Será la clase encargada de la lógica de negocio para subir un fichero a un servidor FTP.
ATRIBUTOS PROPUESTOS
<ul style="list-style-type: none"> ▪ Conexión FTP: Conexión FTP a la que se debe subir el fichero. ▪ Origen: Fichero que debe ser subido.
MÉTODOS PROPUESTOS
<ul style="list-style-type: none"> ▪ Establecer conexión: Crea una conexión al servidor FTP. ▪ Subir un fichero: Sube el fichero indicado al servidor FTP. ▪ Getters y setters: Métodos selectores y modificadores.

<u>Manager de comprensión</u>
DESCRIPCIÓN
Esta clase se encargará de comprimir los directorios que se vayan a copiar.
RESPONSABILIDADES
Al realizar una copia de un directorio, en caso de querer subir el resultado a un servidor FTP, éste debe ser comprimido; esta clase se encargará de llevar a cabo dicha función.
ATRIBUTOS PROPUESTOS
<ul style="list-style-type: none"> ▪ Origen: Fichero que debe ser comprimido. ▪ Destino: Directorio donde debe ser guardado el resultado de la comprensión.
MÉTODOS PROPUESTOS
<ul style="list-style-type: none"> ▪ Comprimir directorio: Comprime el directorio seleccionado. ▪ Getters y setters: Métodos selectores y modificadores.

<u>Manager de copia de directorios</u>
DESCRIPCIÓN
Esta clase se encargará de realizar las copias de directorios o ficheros.
RESPONSABILIDADES
La responsabilidad de esta clase es la de crear copias de directorios y todo su contenido, así como de ficheros.
ATRIBUTOS PROPUESTOS
<ul style="list-style-type: none"> ▪ Origen: Fichero o directorio que debe ser copiado. ▪ Destino: Directorio donde debe realizarse la copia.
MÉTODOS PROPUESTOS
<ul style="list-style-type: none"> ▪ Copiar directorio: Copia el directorio seleccionado en el destino. ▪ Getters y setters: Métodos selectores y modificadores.

<u>Manager de copia de base de datos</u>
DESCRIPCIÓN
Esta clase se encargará de realizar las copias de sistemas de base de datos Microsoft SQL Server.
RESPONSABILIDADES
La responsabilidad de esta clase es la de conectarse a un sistema de base de datos y ordenar un backup a disco de la base seleccionada.
ATRIBUTOS PROPUESTOS
<ul style="list-style-type: none"> ▪ Conexión con la base de datos: Conexión con la base de datos a copiar. ▪ Destino: Directorio donde debe realizarse la copia.
MÉTODOS PROPUESTOS
<ul style="list-style-type: none"> ▪ Realizar copia: Realiza un backup de la base de datos en el destino. ▪ Getters y setters: Métodos selectores y modificadores.

<u>Clase programadora</u>
DESCRIPCIÓN
Esta clase realizará la integración con el módulo Quartz Scheduler comentado anteriormente en este documento.
RESPONSABILIDADES
La responsabilidad de esta clase es la de interactuar con la biblioteca encargada de lanzar los hilos de la aplicación llegado el momento.
ATRIBUTOS PROPUESTOS
<ul style="list-style-type: none"> ▪ Programador: Objeto Scheduler de la biblioteca de Quartz.
<i>continúa en la próxima página</i>

Clase programadora
MÉTODOS PROPUESTOS
<ul style="list-style-type: none"> ▪ Agregar backup al programador: Añade a la pila de ejecución un backup. ▪ Eliminar un backup del programador: Elimina un backup de la pila de ejecución.

Clase log
DESCRIPCIÓN
Clase encargada de la gestión de un fichero de LOG.
RESPONSABILIDADES
La responsabilidad de esta clase es la de mantener un registro de actividades de la aplicación.
ATRIBUTOS PROPUESTOS
<ul style="list-style-type: none"> ▪ LOG: Ubicación del registro de LOG.
MÉTODOS PROPUESTOS
<ul style="list-style-type: none"> ▪ Escribir entrada de log: Inserta en el LOG una nueva acción o suceso. ejecución.

Las clases relativas a la interfaz gráfica no se han tenido en cuenta en esta fase del análisis pues se estaba a la espera de conocer los bocetos del diseñador gráfico de la empresa para poder maquetar la aplicación y conocer los paneles, diálogos, ventanas ...

Aplicación Web

Para el caso de la aplicación Web se va a realizar un proyecto en capas utilizando el framework Struts2. Además, se utilizará el patrón de diseño conocido como DAO para implementar la persistencia.

No se detallan las clases debido a que se considera la estructura básica de un proyecto con dicho framework que implemente el patrón mencionado. Simplemente se crearán Actions para el correcto funcionamiento de Struts2, las clases del modelo, las clases DAO para las relaciones con base de datos y los managers encargados de la lógica intermedia.

4.5. Análisis de casos de uso y escenarios

En esta sección se describirán los casos de uso identificados anteriormente de forma detallada, a través de sus escenarios. Los escenarios describen las interacciones entre los usuarios y el sistema e incluyen información acerca de los objetivos, expectativas, motivaciones, acciones y reacciones que se llevan a cabo.

4.5.1. Aplicación cliente para Windows

C1.1 Registro de usuario	
Precondiciones	Este caso de uso solo se produce al iniciar por primera vez la aplicación.
Poscondiciones	El usuario quedará registrado y no podrán registrarse más usuarios.
Descripción	Se debe mostrar al usuario un formulario de registro que debe cumplimentar con nombre de usuario y contraseña.
Notas	Se pretendía incluir restricciones de seguridad en la contraseña pero la decisión de la empresa fue prescindir de ellas.

C1.2 Autenticación de usuario	
Precondiciones	Para poder autenticarse en el sistema no debe haber ninguna sesión abierta en el mismo.
Poscondiciones	El usuario quedará autenticado y podrá hacer uso de la herramienta.
Descripción	Para poder hacer uso de las herramientas de la aplicación, se debe iniciar sesión con el nombre de usuario y la contraseña introducidas en el registro.
Excepciones	En caso de no ser correcta la autenticación, el usuario no podrá utilizar la aplicación. Sin embargo, la opción de inicio de sesión siempre estará disponible.

C2.1 Creación de una tarea	
Precondiciones	El usuario ha debido iniciar sesión satisfactoriamente en el sistema.
Poscondiciones	Una nueva tarea será añadida al sistema y se encontrará disponible.
Descripción	Un usuario autenticado, podrá crear una tarea con la finalidad de lanzarla o bien, de añadirla en algún momento a una programación de backup.

C2.2 Edición de una tarea	
Precondiciones	El usuario ha debido iniciar sesión satisfactoriamente en el sistema y el sistema debe contener una tarea que no pertenezca a una programación backup para que ésta, pueda ser modificada.
Poscondiciones	La tarea inicial se verá modificada.
Excepciones	Si la tarea pertenece a una programación de backup, el usuario será notificado de la imposibilidad de modificación.
Descripción	Un usuario autenticado, podrá editar la información de una tarea, cambiando nombre de la misma, origen y destino.

C2.3 Eliminar una tarea	
Precondiciones	El usuario ha debido iniciar sesión satisfactoriamente en el sistema y el sistema debe contener una tarea que no pertenezca a una programación backup para que ésta, pueda ser eliminada.
<i>continúa en la próxima página</i>	

C2.3 Eliminar una tarea	
Poscondiciones	La tarea será eliminada.
Excepciones	Si la tarea pertenece a una programación de backup, el usuario será notificado de la imposibilidad de eliminación.
Descripción	Un usuario autenticado, podrá eliminar una tarea existente.

C2.4 Lanzar una tarea	
Precondiciones	El usuario ha debido iniciar sesión satisfactoriamente en el sistema y el sistema debe contener una tarea que pueda ser lanzada.
Poscondiciones	La tarea será ejecutada.
Descripción	Un usuario autenticado, podrá ejecutar una tarea existente. La copia se realizará en dicho momento pero únicamente se repetirá en caso de que esté incluida en un backup

C3.1 Programación de un backup	
Precondiciones	El usuario ha debido iniciar sesión satisfactoriamente en el sistema y el sistema debe contener una tarea que pueda ser lanzada.
Poscondiciones	La tarea será ejecutada siempre que se cumplan las condiciones programadas.
Descripción	Un usuario autenticado podrá crear una programación de backup añadiendo al menos una tarea previamente creada y seleccionando en un formulario cuándo desea que se ejecute.

C3.2 Consultar backups	
Precondiciones	El usuario ha debido iniciar sesión satisfactoriamente en el sistema y el sistema debe contener algún backup para ser mostrado, de lo contrario se mostrará una tabla sin datos.
Poscondiciones	Se mostrará una tabla con los backups que se encuentran activos.
Descripción	Un usuario podrá ver los backups que se encuentran activos.

C3.3 Eliminar un backup	
Precondiciones	El usuario ha debido iniciar sesión satisfactoriamente en el sistema y el sistema debe contener algún backup que pueda ser eliminado.
Poscondiciones	El backup será eliminado y no se ejecutará más veces.
Descripción	Un usuario autenticado podrá eliminar un Backup, anulando así su programación e impidiendo que vuelva a lanzarse.

C4.1 Consulta del LOG	
Precondiciones	El usuario ha debido iniciar sesión satisfactoriamente en el sistema.
Poscondiciones	El LOG de la aplicación será visualizado por el usuario.
Descripción	Un usuario autenticado podrá consultar las actividades y los sucesos producidos en la aplicación, realizando una consulta del LOG del sistema.

C4.2 Consultar resultados de los backups realizados	
Precondiciones	El usuario ha debido iniciar sesión satisfactoriamente en el sistema y debe haber algún backup con resultados para mostrar, de lo contrario, la tabla aparecerá sin información.
<i>continúa en la próxima página</i>	

C4.2 Consultar resultados de los backups realizados	
Poscondiciones	El usuario podrá visualizar una tabla con los resultados de los backups realizados.
Descripción	Un usuario autenticado podrá consultar los resultados de los backups realizados.

C5.1 y C5.2 Agregar información de una conexión	
Precondiciones	El usuario ha debido iniciar sesión satisfactoriamente en el sistema.
Poscondiciones	El sistema almacenará los datos de una conexión a una base de datos o a un servidor FTP.
Descripción	Un usuario autenticado podrá añadir al sistema los datos de una conexión para posteriormente referenciarla cuando quiera realizar un backup.

C5.3 y C5.6 Probar una conexión	
Precondiciones	El usuario ha debido iniciar sesión satisfactoriamente en el sistema y éste debe contar con información de conexiones que puedan ser probadas.
Poscondiciones	El sistema informará de la disponibilidad de la base de datos o el servidor FTP.
Descripción	Un usuario autenticado podrá comprobar la disponibilidad de una conexión a un sistema de base de datos o a un servidor FTP.

C5.4 y C5.7 Editar una conexión	
Precondiciones	El usuario ha debido iniciar sesión satisfactoriamente en el sistema y éste debe contar con información de conexiones que puedan ser modificadas.
Poscondiciones	El sistema guardará los cambios realizados en la información de conexión de la base de datos o el servidor FTP.
Descripción	Un usuario autenticado podrá editar la información de una conexión a un sistema de base de datos o a un servidor FTP.

C5.5 y C5.8 Eliminar una conexión	
Precondiciones	El usuario ha debido iniciar sesión satisfactoriamente en el sistema y éste debe contar con información de conexiones que puedan ser eliminadas.
Poscondiciones	El sistema eliminará los datos de la conexión seleccionada.
Descripción	Un usuario autenticado podrá eliminar la información de una conexión a un sistema de base de datos o a un servidor FTP.

C6.1 Actualizar información del servicio Web	
Precondiciones	El usuario ha debido iniciar sesión satisfactoriamente en el sistema.
Poscondiciones	El sistema modificará la información de conexión con el servicio Web.
Descripción	Un usuario autenticado podrá modificar la información de conexión del servicio Web.

4.5.2. Aplicación Web

C1.1 Ver información de todos los backups de los clientes	
Actor	Usuario con rol de Administrador
Precondiciones	El usuario ha debido iniciar sesión satisfactoriamente en el sistema con una cuenta con el rol de Administrador.
Poscondiciones	El sistema mostrará la información de todos los backups de todos los clientes
Descripción	Un usuario autenticado con una cuenta de Administrador podrá visualizar la información de todos los clientes.

C1.2 Ver información detallada de los backups de los clientes	
Actor	Usuario con rol de Administrador
Precondiciones	El usuario ha debido iniciar sesión satisfactoriamente en el sistema con una cuenta con el rol de Administrador.
Poscondiciones	El sistema mostrará la información de las tareas de los backups de los clientes
Descripción	Un usuario autenticado con una cuenta de Administrador podrá visualizar la información de las tareas de los backups los clientes.

C2.1 Ver información de todos los backups del cliente	
Actor	Usuario con rol de Cliente
Precondiciones	El usuario ha debido iniciar sesión satisfactoriamente en el sistema con una cuenta con el rol de Cliente.
Poscondiciones	El sistema mostrará la información de todos los backups del cliente
Descripción	Un usuario autenticado con una cuenta de Cliente, podrá visualizar los resultados de sus backups

C2.2 Ver información detallada de los backups del cliente	
Actor	Usuario con rol de Cliente
Precondiciones	El usuario ha debido iniciar sesión satisfactoriamente en el sistema con una cuenta con el rol de Cliente.
Poscondiciones	El sistema mostrará la información de las tareas de los backups del cliente.
Descripción	Un usuario autenticado con una cuenta de Cliente podrá visualizar la información de las tareas de los backups del cliente.

4.6. Análisis de interfaces de usuario

Las interfaces de usuario de este proyecto han sido impuestas por el diseñador gráfico de Ecocomputer S.L. Sin embargo, se ha dado libertad para realizar pequeñas modificaciones que permitieran un sistema más usable y accesible. Tal como se comentará más adelante, se han tenido en cuenta las limitaciones de algunos usuarios para la realización de los interfaces.

Puesto que se cuenta con los diseños realizados por la empresa, no han sido necesarios los bocetos o mockups típicos de las fases de análisis.

4.6.1. Aplicación cliente para Windows

A continuación se presentan las imágenes realizadas por el diseñador gráfico y que se han intentado plasmar con los mínimos cambios posibles en la aplicación realizada.

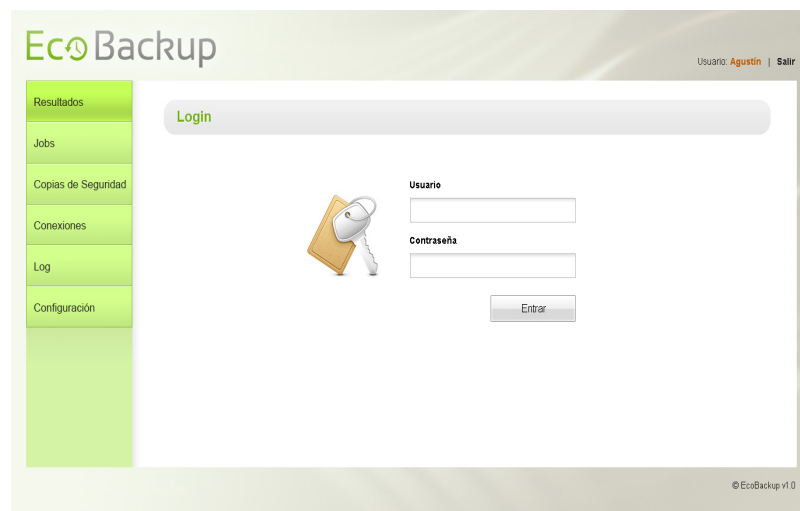


Figura 4.6: Autenticación en el sistema

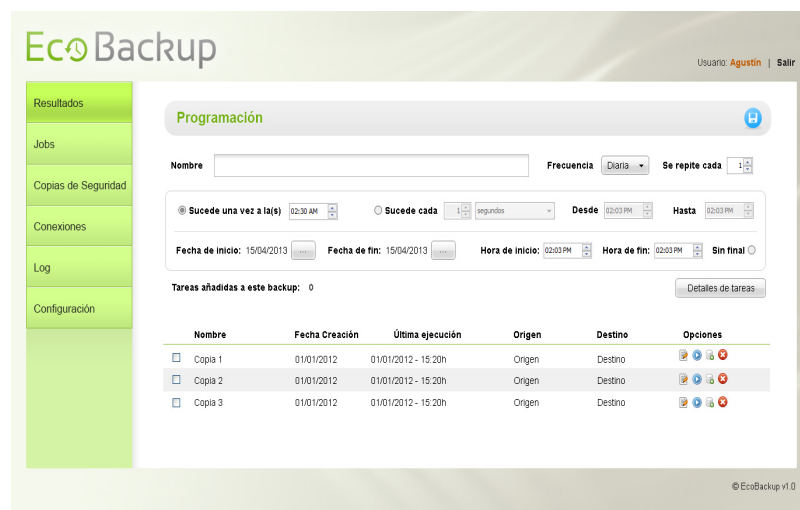


Figura 4.7: Creación de una programación de Backup

Debe tenerse en cuenta que las imágenes que acompañan esta sección, fueron el resultado del análisis posterior al cambio de requisitos y de alcance del proyecto.



Figura 4.8: Resultados de los Backups



Figura 4.9: Tareas creadas en el sistema

Pese a no realizarse un boceto de todas las ventanas que finalmente tiene la aplicación y faltar los diálogos, se tiene una idea general del look and feel global, lo que permite realizar el resto de componentes de la interfaz gráfica siguiendo el modelo.

4.6.2. Aplicación Web

La aplicación Web deberá integrarse en una plataforma de la empresa y por lo tanto presentará una apariencia similar al resto de aplicaciones desplegadas en la misma. No se ha realizado un análisis previo en este aspecto, al considerar tarea del diseñador gráfico la labor, y no requerir demasiado esfuerzo al contar con aplicaciones que deben ser tomadas como modelo de diseño.

Por otra parte, el diseño Web presenta mayor facilidad a la hora de modificar una interfaz gráfica, al contar con hojas de estilo claramente diferenciadas del código. Por esta razón, se dedicó mayores esfuerzos al cliente de escritorio, pues la interfaz de usuario implica la programación de componentes.

4.7. Especificación del plan de pruebas

Dadas las características del proyecto y su utilización en un entorno real, las pruebas cobran aún más importancia. Por otra parte, se tiene la ventaja de que se cuenta con un equipo de probadores y facilita la localización de errores o anomalías. El proyecto será utilizado por el personal de la empresa Ecocomputer S.L. antes de ser lanzado a producción. En esta fase del proyecto, se realizarán pruebas con información real y en entornos reales, lo que sin duda, mejorará los resultados de las pruebas realizadas.

En este apartado se analizarán por encima las pruebas que se realizarán al proyecto, y que serán detalladas más adelante en esta documentación.

- **Pruebas unitarias:** Se realizarán pruebas unitarias utilizando el framework JUnit¹, de la lógica fundamental de la aplicación de escritorio. Se comprobará de esta forma el correcto funcionamiento de los métodos con más funcionalidad, tales como los formateadores de CRON, los encargados de realizar las copias, las conexiones . . .
- **Pruebas del integración:** Estas pruebas serán realizadas el colaboración con la empresa. Se probará que todo funciona como debería una vez desplegado en los sistemas de producción y que la integración es correcta.
- **Pruebas de sistema:** Se realizarán pruebas del sistema una vez desarrollados todos los módulos. Se pretende garantizar con éstas, que todos los módulos trabajan correctamente integrados y el proyecto funciona como es de esperar.
- **Pruebas de usabilidad:** Para este tipo de pruebas, nuevamente se cuenta con la colaboración de Ecocomputer S.L. Serán los propios usuarios finales quienes realicen las pruebas y comenten sus dificultades o posibles cambios que consideren necesarios para una mejor usabilidad del sistema.

Por último, cabe destacar que se creará un entorno lo más similar posible al de despliegue final, para realizar las pruebas durante el desarrollo. Se utilizarán las mismas versiones del gestor de base de datos, el servidor de aplicaciones, máquina virtual de Java . . . Con esto, se pretende evitar problemas una vez desarrollado con alguna incompatibilidad.

Además, no parecía lo más adecuado, pese a contar con el permiso de la empresa, para utilizar el servidor de despliegue y la base de datos final como herramientas para el desarrollo y probado.

¹Más información sobre JUnit en: <http://junit.org/>

Capítulo 5

Diseño del Sistema

5.1. Arquitectura del sistema

5.1.1. Diagramas de paquetes

Aplicación cliente de escritorio para entornos Windows

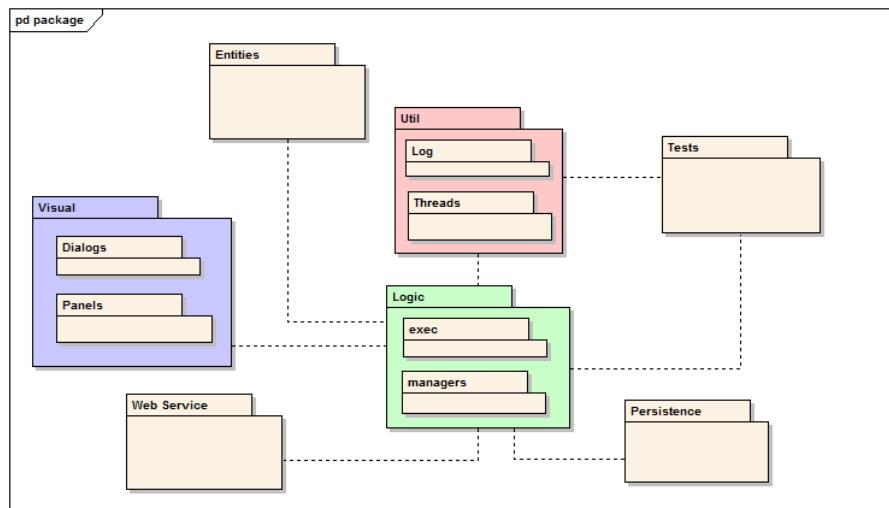


Figura 5.1: Diagrama de paquetes de la aplicación de escritorio para Windows

En el diagrama superior se puede ver la estructura de paquetes que tendrá la aplicación cliente para entornos Windows. A continuación se detalla cada paquete

- **Visual:** Este paquete contendrá todas las clases encargadas de la parte visual de la aplicación. Dentro del patrón arquitectónico Modelo-Vista-Controlador, este paquete haría referencia a la Vista. Dado que se trata de una aplicación con una gran interfaz gráfica, se han creado dos subpaquetes para los paneles y los cuadros de diálogo de la aplicación.
- **Entities:** Si el paquete anterior contenía las clases de la Vista, en este caso se trata del Modelo. Este paquete contendrá las clases del modelo de dominio de la aplicación. Por ello, dentro de este paquete encontraremos clases sin lógica como Backup, Tarea, Historial de backup ...
- **Web Service:** Este paquete contendrá las clases autogeneradas por el entorno para la conexión con un servicio Web SOAP.
- **Util:** Se utilizará un paquete útil para la implementación de elementos concretos que no dependen ni de la lógica de negocio ni de temas más concretos como la persistencia

o la interfaz gráfica. Así, dentro de este paquete se encontrará el subpaquete para la gestión del sistema de LOG y otro subpaquete para la gestión de hilos de la aplicación.

- **Logic:** Este paquete contiene la lógica de negocio propia de la aplicación. Tendrá dos subpaquetes interiores para separar las clases ejecutoras de funcionalidades y los managers encargados de conectar con otros paquetes o realizar grupos de funcionalidades enlazadas.
- **Persistence:** Las clases encargadas de las relaciones con la base de datos SQLite con la que funciona la aplicación, se encontrarán en este paquete. Serán clases que hagan uso de JDBC y cuya única funcionalidad será cargar datos de la base de datos o almacenar los mismos.
- **Tests:** Por último, pero no menos importante, se encuentra el paquete para la realización de pruebas unitarias del código. Como ya se ha mencionado anteriormente, se hará uso del framework específico para esta labor, JUnit.

Aplicación Web

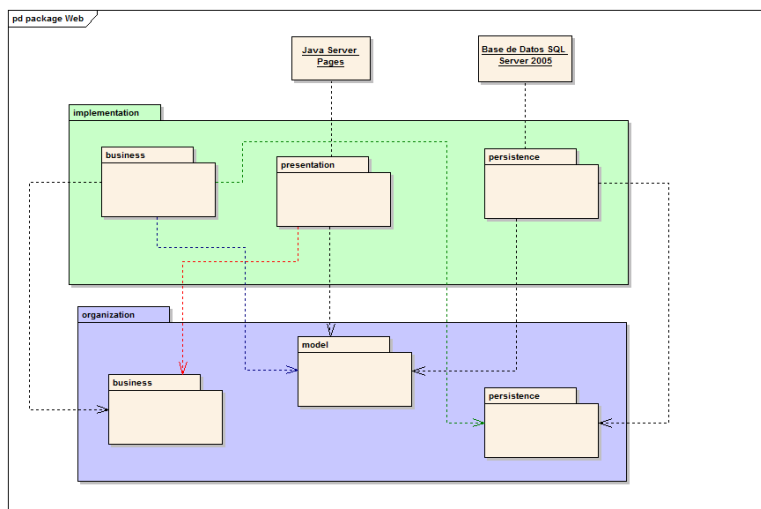


Figura 5.2: Diagrama de paquetes de la aplicación Web

En el diagrama superior se puede ver la estructura de paquetes que tendrá la aplicación Web. Se puede comprobar que hay una división clara entre las clases con lógica y que contienen la implementación y las clases que actúan de interfaces creando la estructura del proyecto.

A continuación se detalla cada paquete:

- **Presentation:** Contiene las clases «Action» que colaboran directamente con las JSP para recoger datos o mostrarlos. Mediante la implementación de la interfaz «ModelDriven» de struts, los datos se transportan en objetos del modelo, de ahí que necesite colaborar con dicho paquete «organization.model».

Para llamar a la siguiente capa de la arquitectura es necesario utilizar los interfaces de objetos contenidos en el paquete «organization.business». Dichas interfaces declaran las firmas de los métodos que necesitan las Action para llevar a cabo las tareas ordenadas por el usuario.

- **Persistence:** Agrupa a los Data Access Object (DAO), que son las clases que suministran el acceso y comunicación al sistema de persistencia, en este caso una base de datos. Implementan los métodos de creación, obtención, actualización y borrado

de datos necesarios para la ejecución de las tareas solicitadas en la capa de lógica de la aplicación.

Dichas operaciones las realiza utilizando los objetos del paquete «organization.model». Los DAO implementan interfaces del paquete «organization.persistence» que dictan los métodos que deben incorporar, y es por eso que existe una relación en el diagrama de paquetes.

- **Business:** Este paquete contiene la lógica de negocio y actúa de intermediario entre el paquete de presentación con los Actions de Struts2 y el paquete de presentación con los objetos DAO.
- **Model:** Este paquete contiene las clases planas que forman el modelo de dominio del sistema. En este caso serán únicamente, historial de backups e historial de tareas.

5.1.2. Diagrama de componentes

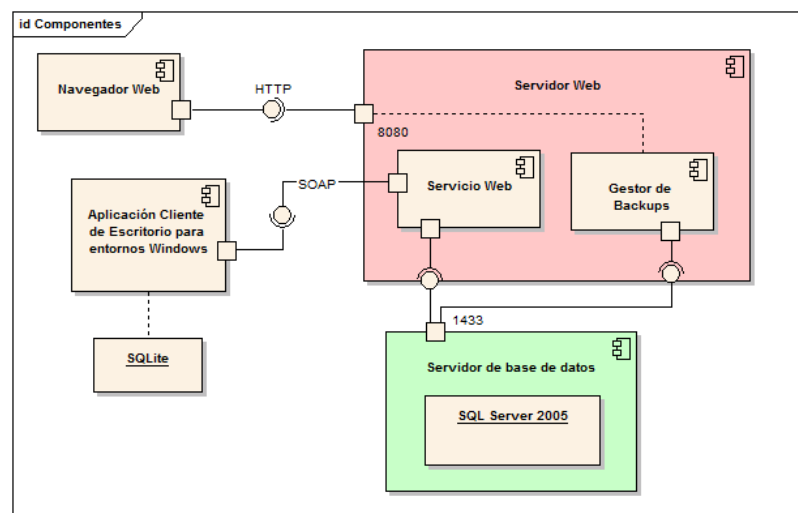


Figura 5.3: Diagrama de componentes del proyecto

Este diagrama de componentes, que tiene como objetivo mostrar las dependencias e interconexiones entre componentes lógicos de un sistema, refleja la arquitectura general planteada.

Se puede apreciar en el diagrama, como se conectan los diferentes módulos del proyecto y los tipos de protocolos o tecnologías empleadas. De esta forma, mediante la tecnología SOAP se conecta la aplicación de escritorio con el servicio Web y éste con la base de datos mediante los drivers de conexión propios de SQL Server. Esta misma interfaz de conexión, es utilizada por la aplicación Web gestora de Backups, a la cual se puede acceder, como es habitual, mediante un navegador Web utilizando el protocolo HTTP.

El servicio Web y la aplicación Web no se encuentran comunicadas, ya que únicamente comparten la base de datos y no intercambian información directamente.

5.1.3. Diagrama de despliegue

En el siguiente diagrama se presenta la estructura de despliegue del proyecto. Se puede diferenciar una división en tres equipos físicos.

Por un lado se encontrarán los ordenadores de los clientes con sistema operativo Microsoft Windows. Para poder disfrutar de todo el proyecto, estos equipos deberán contar con un navegador Web, conexión a Internet, la máquina virtual de Java y el JAR de la aplicación, así como, una base de datos SQLite que es utilizada por la aplicación como sistema de persistencia.

Por otra parte, se encuentran dos servidores de la empresa Ecocomputer S.L. La decisión de utilizar el mismo servidor físico para desplegar el servicio Web y la aplicación Web ha sido interna a la misma, pudiendo ser portado alguno de ellos a un emplazamiento distinto en un futuro si se considerara necesario. Además del servidor de despliegue con un software como Tomcat v6 que actúa de servidor de aplicaciones, se cuenta con otro servidor para el gestor de base de datos SQL Server 2005.

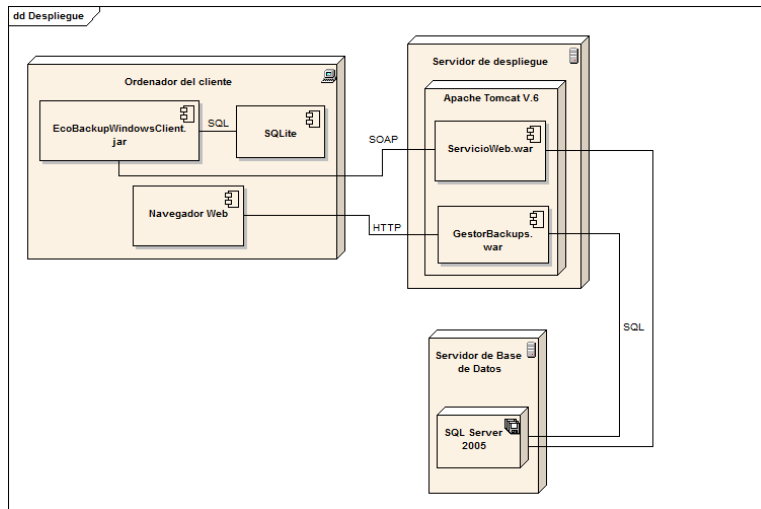


Figura 5.4: Diagrama de despliegue del proyecto

No se detallan las direcciones o nombres DNS de las máquinas, ya que se considera de poca importancia para entender la arquitectura y no parece adecuado revelar información de la empresa.

5.2. Diseño de clases

En esta sección se representan las clases que formarán el proyecto.

5.2.1. Diagrama de clases

Aplicación cliente de escritorio para Windows

Paquete visual

Este paquete, como ya se comentado anteriormente, contiene las clases encargadas de la interfaz gráfica. Dado que son clases llenas de componentes de Swing¹ y que únicamente presentan métodos para la inicialización y manejo de los mismos, no se considera necesario realizar un diagrama de clases de las mismas.

Este paquete contendrá por una parte una clase encargada de representar la ventana de la aplicación, ya que únicamente habrá una ventana y se modificarán los paneles interiores según convenga. Estos paneles se encontrarán en un subpaquete, dada la multitud de clases que se generarán. Por otra parte, los cuadros de diálogo necesarios para notificaciones o pedir información extra al usuario, para realizar una operación, se encontrarán en otro subpaquete.

Paquete Entities

Este paquete contiene las clases del modelo. Se trata de clases planas, sin funcionalidad. En el diagrama inferior, se muestran las clases ausentes de métodos, ya que únicamente contienen selectores y modificadores y se considera aumentar inútilmente la información del diagrama, dificultando su lectura.

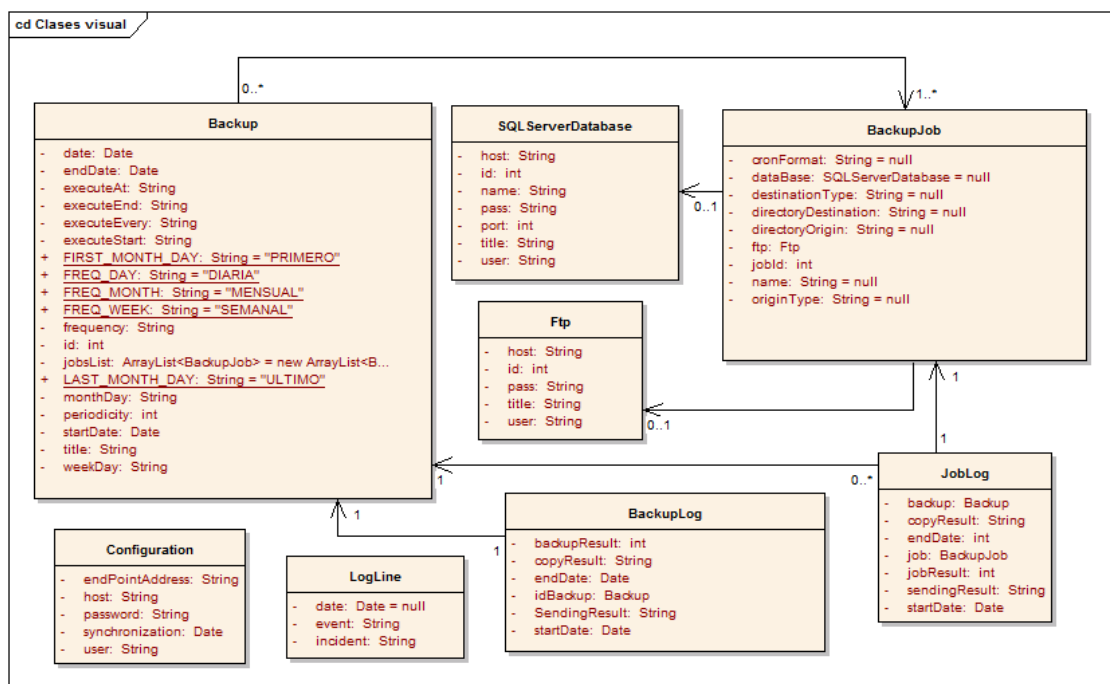


Figura 5.5: Diagrama de clases entities

En el diagrama superior, se pueden ver las clases del modelo, acompañadas de clases igualmente planas, que se requieren para dotar de alguna funcionalidad a la aplicación. Así, aparecen clases como LogLine, para representar una entrada en el LOG del sistema. Configuration, por su parte contiene la información relativa a la comunicación con el servicio Web.

¹Más información sobre Swing en: <http://docs.oracle.com/javase/tutorial/uiswing/start/about.html>

Paquete logic

Este paquete contiene la lógica de negocio propia del sistema. Se encuentra dividido en dos subpaquetes para separar las clases ejecutoras de los manager que actúan de intermediarios entre paquetes y cuya funcionalidad abarca más de una operación lógica.

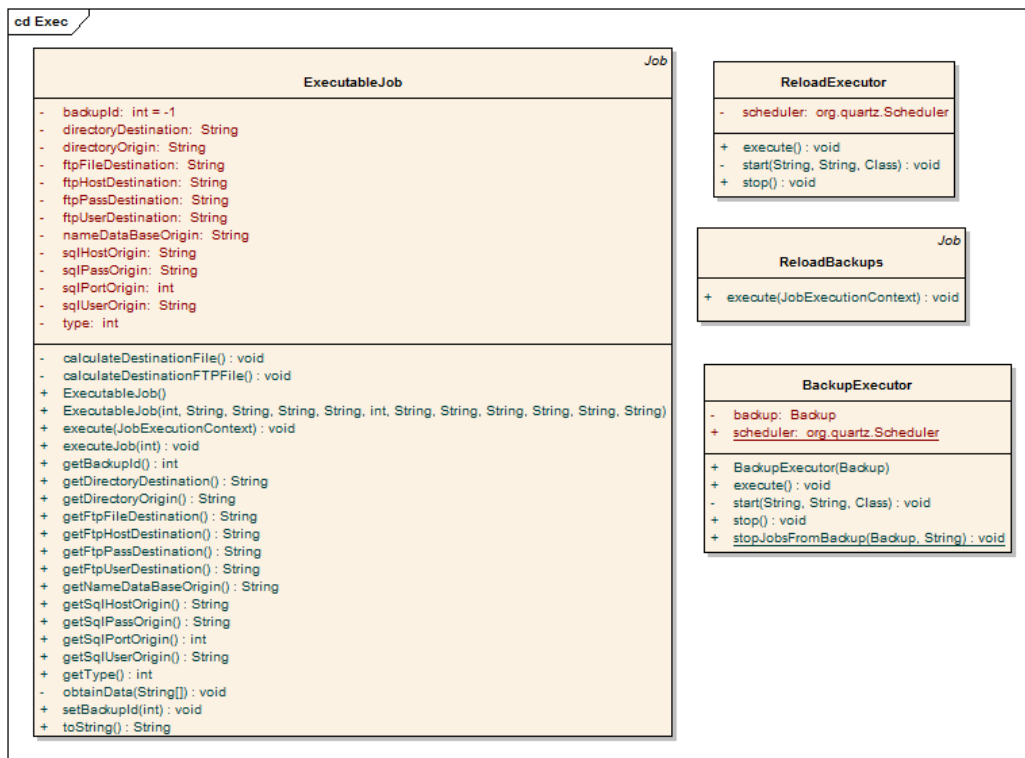


Figura 5.6: Diagrama de clases del paquete Exec

A continuación se detallan las componen este paquete:

- ExecutableJob:** Esta clase representa una tarea ejecutable. Pese a tener en el modelo las tareas con la información relativa al origen y destino de la copia, para trabajar con el programador de hilos, es necesario utilizar una clase un poco diferente. Esta clase, implementa la interfaz «Job»de la biblioteca Quartz Scheduler comentada con anterioridad en este documento.
- BackupExecutor:** Esta clase será la encargada de ejecutar un backup. Su funcionalidad básica será la de recorrer las tareas ejecutables y lanzarlas cuando se cumpla la programación temporal marcada. Para ello, se cuenta con un objeto «Scheduler»de la biblioteca anteriormente comentada. Como funcionalidades que aporta este ejecutor de backups, cabe destacar la parada de un backup en ejecución (lo que supone eliminar el mismo de la pila de ejecución del programador de hilos) y el inicio de un nuevo backup (que supone incluir dicho objeto en la ejecución del scheduler).
- ReloadBackups:** Se cuenta con un tipo de tarea ejecutable adicional, con la finalidad de mantener un refresco periódico de backups. Esta tarea se encargará de mantener en la pila de ejecución del programador de hilos aquellos backups que deban seguir ejecutándose, eliminando de la misma los que deban ser retirados.
- ReloadExecutor:** Se ha explicado anteriormente la tarea encargada del refresco del sistema, pero de nada serviría sin un ejecutor periódico de dicha tarea. Esta funcionalidad se consigue con esta clase. Su única misión es lanzar el actualizador anteriormente comentado periódicamente.

Paquete util

Se presenta a continuación, el diagrama de clases del paquete Util. Este paquete contiene clases que no encajan dentro de la lógica de negocio de la aplicación, pero que son necesarias para cumplir algunas funcionalidades.

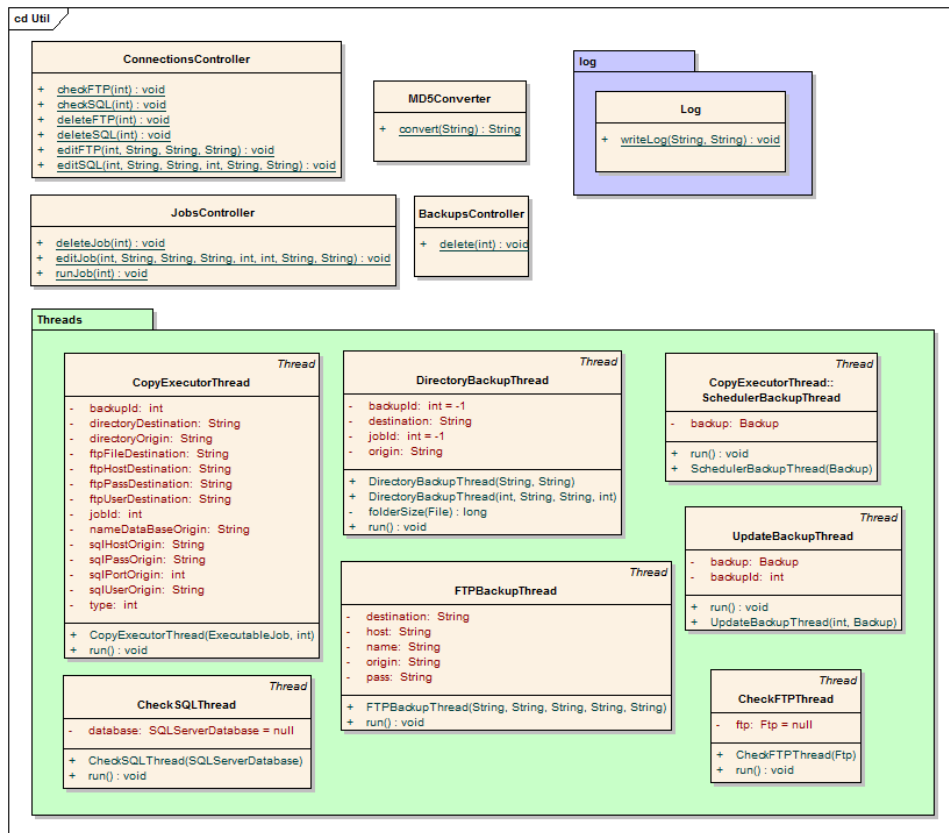


Figura 5.7: Diagrama de clases del paquete Util

A continuación se detallan las componen este paquete:

- MD5Converter:** Dado que se va a trabajar con información sensible como es el caso de contraseñas, se consideraba necesario contar con algún sistema de cifrado de las mismas. Dada la facilidad de implementación y su robustez, se ha decidido implementar el algoritmo de cifrado MD5. Esta clase convierte la cadena en texto plano en la información cifrada.
- Controllers:** Siguiendo el patrón arquitectónico MVC (Modelo-Vista-Controlador), se pretende eliminar la funcionalidad de las clases de visualizado y creación de interfaces. Por esta razón, se cuenta con clases controladoras para las operaciones que requieren algo más que una llamada a una capa inferior.
- Log:** Esta clase presenta una interfaz muy sencilla pero igualmente necesaria. Simplemente contiene un método para incluir un nuevo registro en el LOG del sistema. Se debe indicar el incidente a registrar y la causa del mismo.
- Threads:** Dado que se trata de una aplicación que realiza numerosas actividades en segundo plano, manteniendo la interfaz de usuario operativa para seguir operando, se considera de vital importancia la utilización de hilos. Por esta razón se cuenta con un paquete de este propósito. La única funcionalidad que presentan estas clases es la de encapsular llamadas a capas inferiores, mediante la ejecución de un nuevo hijo del programa.

Paquete webService

Este paquete contiene las clases autogeneradas por el entorno de desarrollo Eclipse a la hora de crear un cliente para un servicio Web SOAP realizado con Axis2. No se presenta diagrama de clases de la fase de diseño porque no es necesario diseñar ni implementar las mismas. Creando el servicio Web, Eclipse se encarga por nosotros de generar las clases necesarias para realizar las llamadas al servicio.

Paquete persistence

Este paquete contiene las clases encargadas de, mediante el framework JDBC, conectarse al sistema de base de datos SQLite y garantizar la persistencia de los datos. Por comodidad de uso, se creará una especie de patrón «Fachada». Se contará con una clase «PersistenceManager» que será la interfaz visible desde el resto de la aplicación, ésta contendrá todas las operaciones de persistencia y delegará en las clases oportunas su tarea. Por ejemplo, contará con un método para obtener un Backup de base de datos, pero delegará la creación de la consulta SQL a la clase «BackupPersistence». Con esto se consigue aparente transparencia para la persistencia pero por otra parte se mantienen separadas las consultas relativas a cada tabla.

Aplicación Web

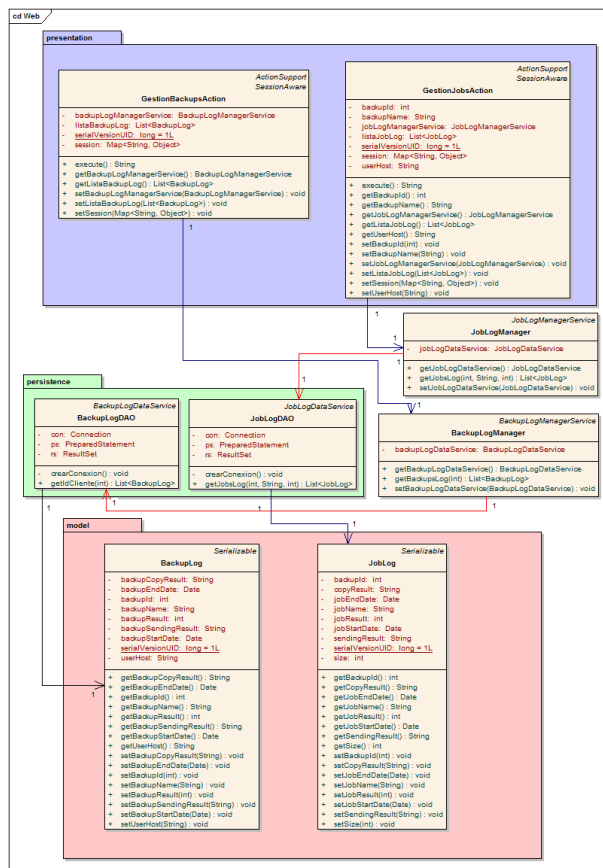


Figura 5.8: Diagrama de clases de la aplicación Web

El diagrama anterior recoge el diseño de la aplicación Web del proyecto. Tal como se ha comentado anteriormente, se utilizará el framework Struts2 y se implementará siguiendo un modelo en capas apoyado sobre el patrón arquitectónico DAO para tratar la persistencia.

El funcionamiento, como el de cualquier proyecto con Struts2 esta fundamentado en los «Actions».

La funcionalidad de cada paquete se encuentra claramente diferenciada. Mientras que el paquete de presentación, contiene los mencionados Actions de Struts2 que desencadenarán en hojas JSP, el paquete de persistencia contendrá únicamente los objetos DAO para recuperar objetos del modelo de base de datos.

Servicio Web

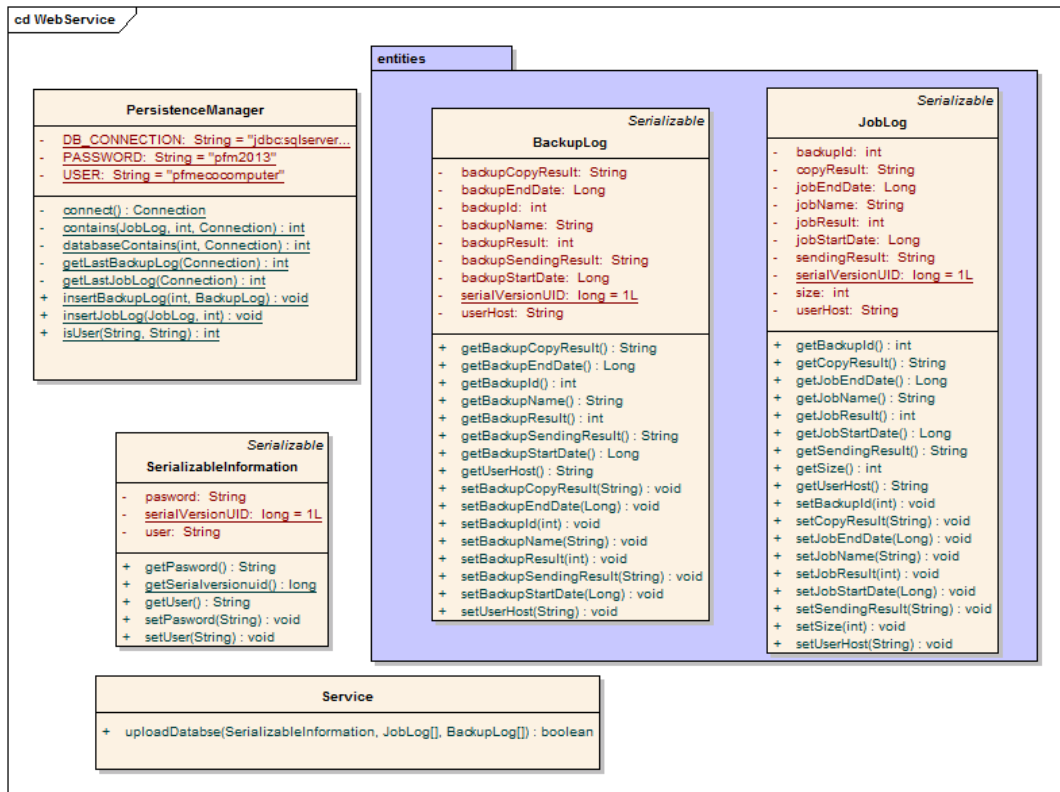


Figura 5.9: Diagrama de clases del servicio Web

En el diagrama superior se muestra el diseño del servicio Web. Se puede observar la simpleza del mismo en el limitado número de clases. Se trata de un servicio que únicamente válida la información de un usuario y en caso de ser correcta actualiza la información de la base de datos.

Se cuenta con una clase gestora de persistencia, la clase propia del servicio que presenta un único método como interfaz, y las clases de tipos de datos. Se han mantenido aisladas las clases de entidades, ya que se trata de la información importante para el servicio.

El funcionamiento del servicio será el siguiente: un cliente se conectará al mismo enviando un objeto de tipo «SerializableInformation», que contendrá la información de autenticación. Por otra parte enviará dos listas con los datos a introducir o actualizar en la base de datos, una para los historiales de los backups y otra para los históricos de las tareas que los forman.

5.3. Diagramas de interacción y estados

A continuación podremos encontrar una serie de diagramas en los que se pretende explicar el funcionamiento interno de alguna de las acciones más importantes del proyecto a lo largo de su ejecución por los diferentes pasos que la comprenden. Mediante los diagramas de interacciones veremos el conjunto de pasos y métodos llamados desde que se inicia una operación hasta que finaliza, además de los elementos implicados en ello. A través de los diagramas de estados se pretende ilustrar qué fases atraviesan los objetos durante una u otra operación. Se muestran los diagramas más representativos del proyecto, ya que algunas operaciones triviales no aportan demasiada información para comprender la arquitectura.

5.3.1. Aplicación de escritorio para entornos Windows

Se van a detallar los pasos necesarios desde que se ejecuta por primera vez la aplicación hasta que se programa el primer backup. Se tomará como ejemplo un backup sobre una base de datos y que suba la copia realizada a un servidor FTP, ya que requiere más pasos y se considera más representativo.

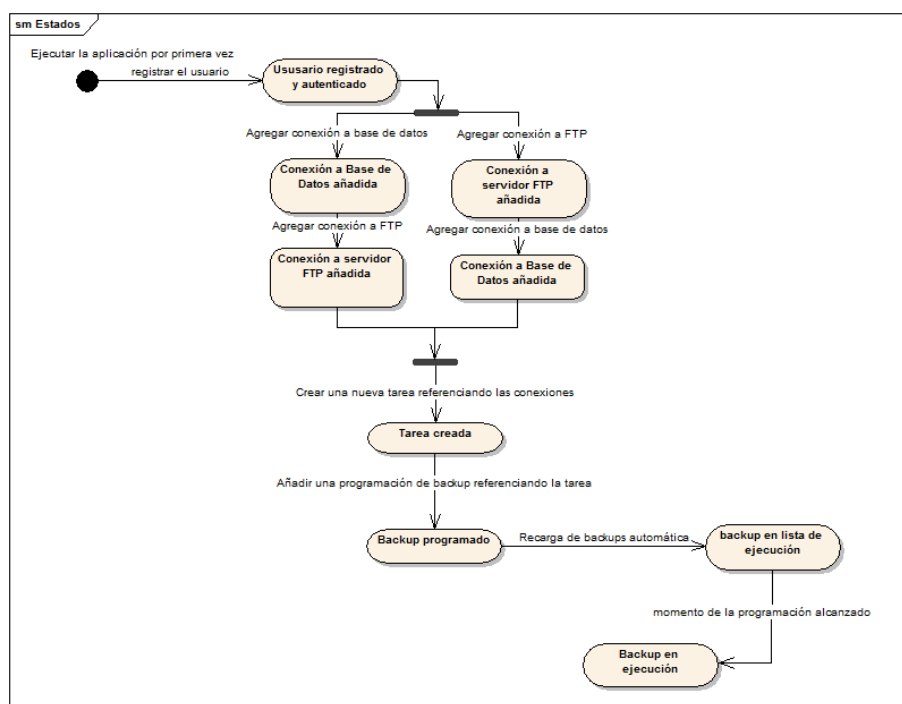


Figura 5.10: Diagrama de estados de creación del primer backup

El diagrama pretende detallar los pasos a seguir para la creación de la primera programación de copias de seguridad.

Lo primero que debe realizar un usuario al iniciar la aplicación por primera vez en un sistema, es registrar un usuario. De esta forma quedará autenticado y tendrá acceso a la herramienta. Para poder crear un Backup, es necesario primero definir tareas, y estas a su vez, en caso de contener referencias a conexiones requieren la creación previa de las mismas.

Una vez que el backup es creado, el sistema lo añade a la lista de ejecución y llegado el momento programado lo ejecuta de manera autónoma. En este diagrama se han prescindido de funcionalidades del sistema como la creación de registros de LOG, la creación de históricos de resultados de las copias y la conexión con el servicio Web para actualizar la información de la base de datos del servidor.

5.3.2. Aplicación Web

Para el caso de la aplicación Web se detallará el proceso requerido para consultar los resultados de un backup y las tareas del mismo. Se considera una tarea sencilla pero que resume toda la funcionalidad de la aplicación.

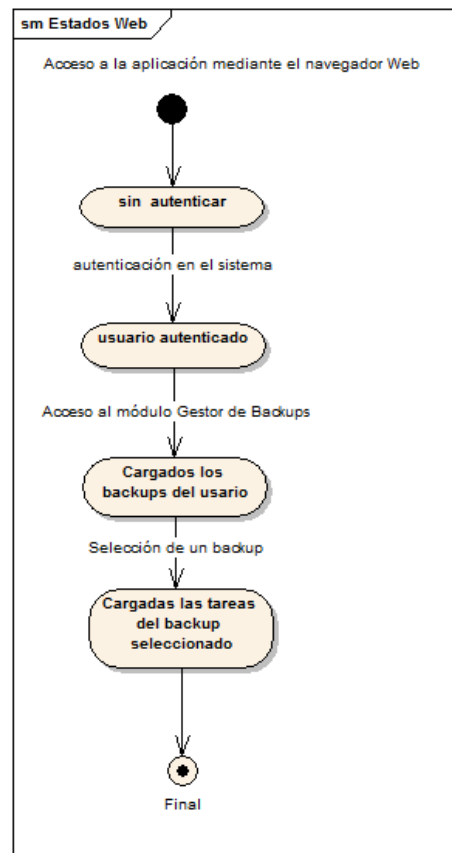


Figura 5.11: Diagrama de estados de la aplicación Web

Se puede comprobar como para poder acceder a la información se debe pasar por un sistema de autenticación. Una vez el sistema reconoce el cliente que está accediendo al sistema le muestra la información relativa a los resultados de los backups realizados en sus equipos. Será el usuario quien deba pedir más información de algún Backup en concreto para conocer el resultado de las tareas del mismo.

5.4. Diagrama de secuencia

Dada la sencillez de la aplicación Web, únicamente se detallará la herramienta de escritorio con un diagrama de secuencia. Se explicará mediante el mismo los pasos llevados a cabo por el sistema desde la creación de un backup hasta que el tiempo de vida del mismo finaliza.

Se han omitido las clases llamadas que generan los nuevos registros de LOG, así como la creación de históricos durante las ejecuciones del Backup, para evitar generar un diagrama con demasiada información y de difícil lectura.

Por otra parte, se ha partido de un usuario autenticado en el sistema. Se recuerda que la aplicación no se encuentra accesible a usuarios anónimos y se debe completar un formulario de usuario y contraseña para realizar cualquier actividad.

Por último, destacar que la ejecución de un Backup no es inmediata, ésta se activa al cumplirse lo programado. Es decir, si un backup es programado para un futuro lejano, éste será añadido a la pila de ejecución del programador de hilos, pero no llegará a ejecutarse hasta llegado el momento concreto de su programación.

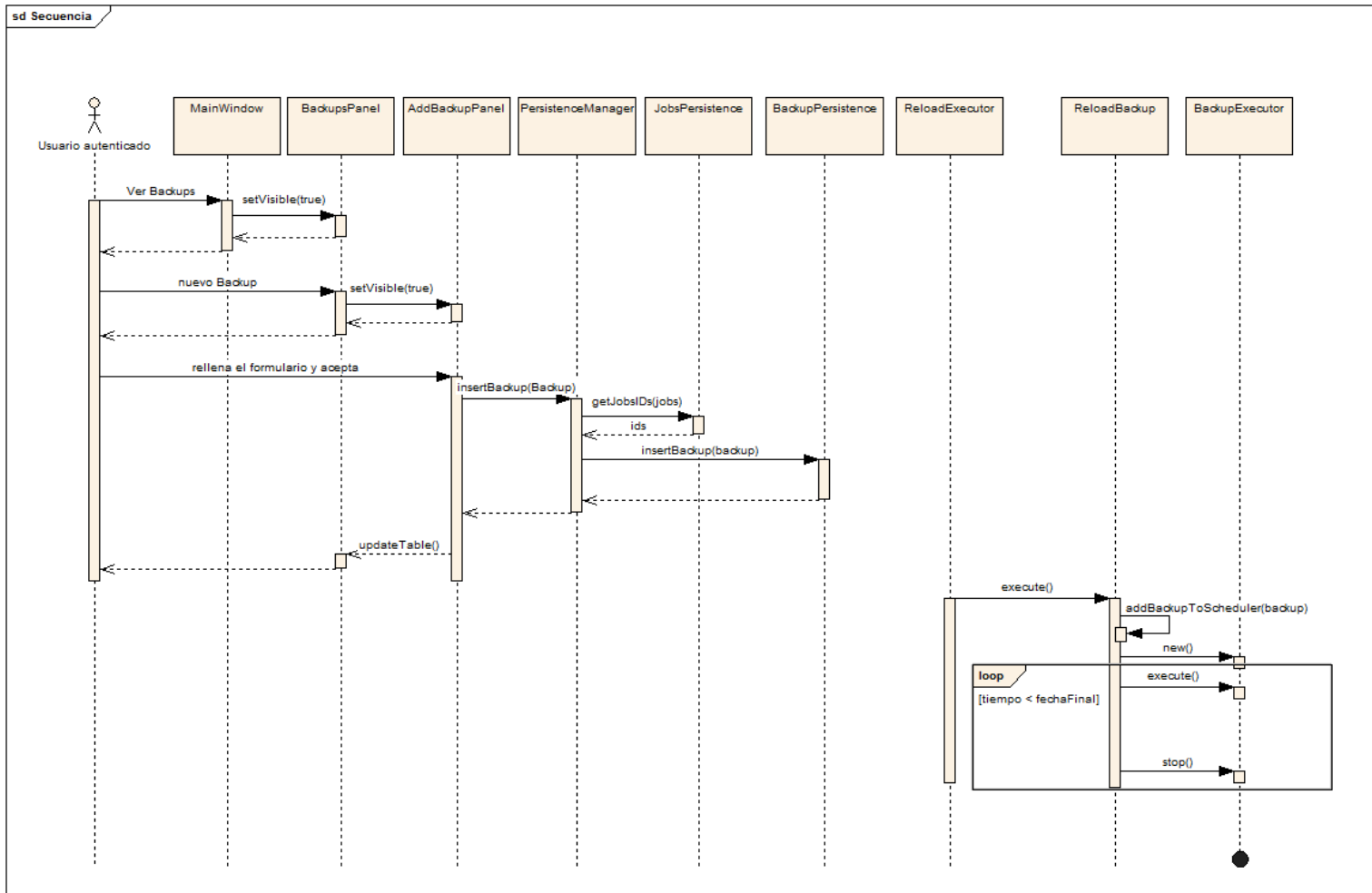


Figura 5.12: Diagrama de secuencia en la creación de un Backup

5.5. Diseño de la base de datos

Este proyecto hace uso de dos sistemas de bases de datos distintos. Por un lado la aplicación cliente trabaja con SQLite, dada su sencillez y su correcto funcionamiento sin requerir recursos o una instalación compleja. Por otra parte, la aplicación Web y el servicio que la mantiene actualizada, hacen uso de Microsoft SQLServer 2005 para almacenar la información

En este apartado únicamente se mencionará la información relativa al proyecto tratado. Como se ha comentado anteriormente, la aplicación se integra en una plataforma con más aplicaciones funcionando. Por esta razón, la base de datos contendrá tablas propias de otros proyectos y que no guardan relación con el que nos ocupa. En este documento, dichas tablas no serán tratadas pese a formar parte del sistema de base de datos utilizado.

5.5.1. Descripción de los sistemas de gestión de bases de datos

SQLite

Tal como se define en su página oficial², SQL es una biblioteca software cuyo código es libre y que dado que no requiere configuración y funciona de manera autónoma y ligera, se ha convertido en un sistema muy utilizado en la actualidad.

Se considera ideal para el proyecto tratado, dado que no requiere instalación o configuración compleja y se puede entregar junto con el JAR a los clientes para que comiencen a utilizar la aplicación al momento.

Microsoft SQL Server 2005

Microsoft SQL Server es un sistema para la gestión de bases de datos producido por Microsoft basado en el modelo relacional. Los lenguajes para consultas soportados son T-SQL y ANSI SQL.

Como principal desventaja frente a sus competidores, se encuentra la limitación de funcionamiento a entornos con operativos de Microsoft. Sin embargo, en las versiones Server de Windows, la integración es muy completa y su instalación no supone ningún esfuerzo.

En este caso, el sistema venía impuesto por la empresa Ecocomputer S.L. Como se ha comentado, la aplicación será integrada en una plataforma con más proyectos y se debe aprovechar la infraestructura existente. Se cuenta con la versión 2005 de Microsoft SQL Server en producción y es por ello que la aplicación debe funcionar con esta versión.

5.5.2. Integración de los sistemas de gestión de bases de datos

Se utilizará JDBC (Java Database Connectivity) para realizar la integración con ambos sistemas de gestión. Se trata de una API que permite la ejecución de operaciones sobre bases de datos desde Java, independientemente del sistema operativo donde se ejecute o de la base de datos a la cual se accede, utilizando el dialecto SQL del modelo de base de datos que se utilice.

Serán necesarios los drivers correspondientes a dichos sistemas de gestión, los cuales pueden descargarse de las páginas oficiales de los productos y son completamente gratuitos y libres para su uso.

Para el caso de SQL Server 2005, se realizará una conexión remota mediante su puerto por defecto (1433). Sin embargo, SQLite debe encontrarse físicamente en el equipo de la aplicación que la utiliza.

²Página oficial de SQLite: <http://www.sqlite.org/>

5.5.3. Diagrama entidad-relación

Aplicación cliente de escritorio para Windows

En la imagen inferior se muestra el diagrama Entidad-Relación de la base de datos que utiliza la aplicación cliente. Puesto que se utiliza un lenguaje orientado a objetos y un sistema de base de datos relacional, se considera esta representación más idónea que otros tipos de representación de modelado de una base de datos.

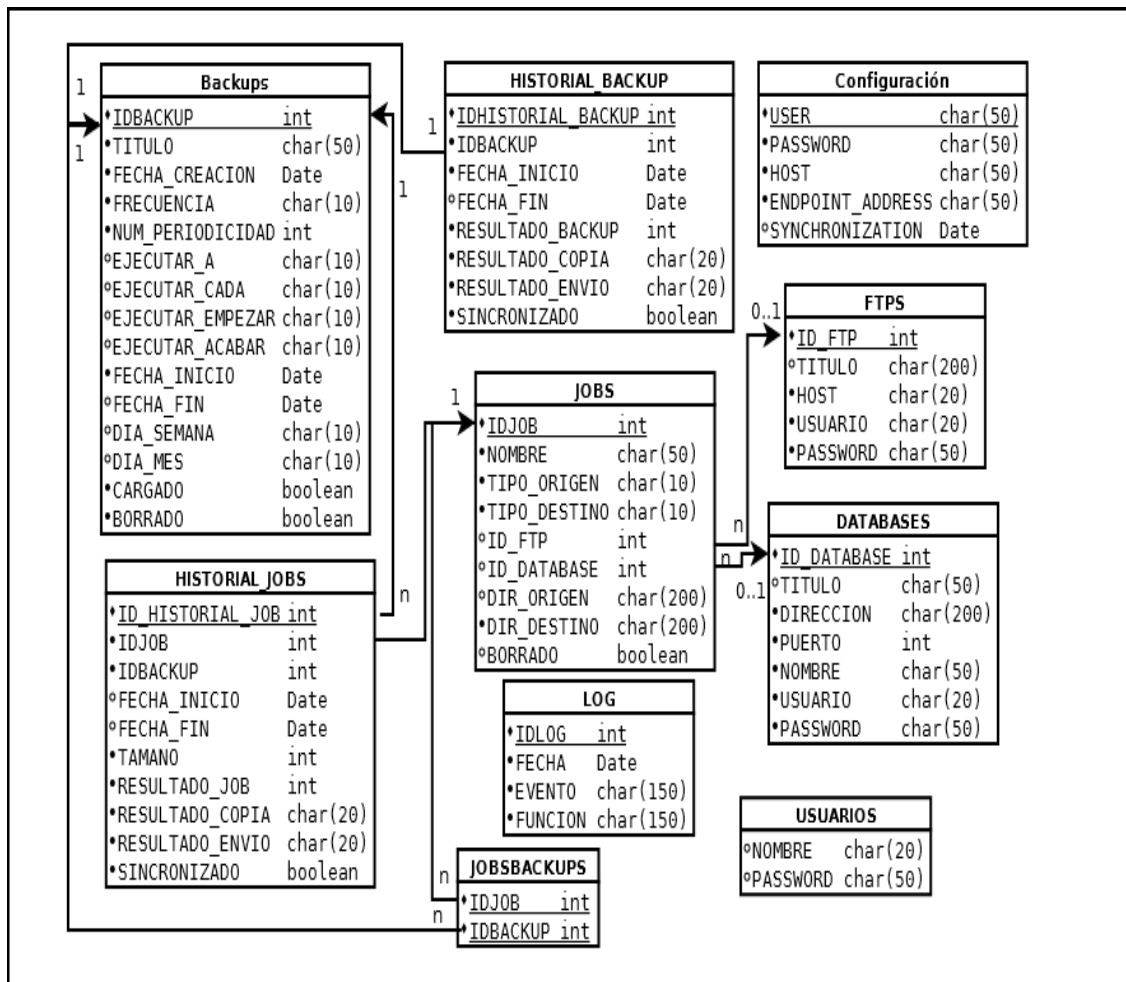


Figura 5.13: Diagrama Entidad-Relación de la base de datos de la aplicación cliente

como puede verse en el modelo, algunas tablas almacenan datos aislados como puede ser el caso de la configuración, los datos de acceso del usuario o el LOG.

Por otra parte se encuentran los datos relacionados de la aplicación que suponen el modelo de negocio. Este diseño es el resultado de varias reuniones con la empresa cliente y varias modificaciones.

No se considera un buen modelo ya que se cuenta con valores nulos en muchos casos, como por ejemplo en la tabla Backup (la que almacena la programación temporal sobre la ejecución de las copias) se cuenta con dos campos: «Ejecutar_A» y «Ejecutar_Cada», de los cuales uno contendrá información y el otro será un valor nulo según el caso. Sin embargo, es la solución acordada con la empresa y no había libertad para su modificación.

Relacionado con esta base de datos y atendiendo a que el cliente no quería que el proyecto eliminara datos antiguos de la base de datos, se recomienda la realización de un Trigger o disparador, que elimine los registros del LOG más antiguos una vez alcanzado un límite admitido.

Aplicación Web

La aplicación Web hace uso de dos tablas propias, tal como se muestra en la imagen inferior. Se debe tener en cuenta además la tabla de usuario que contiene el «ID_Cliente», incluido como clave ajena en ambas tablas.

HISTORIAL_BACKUPS		HISTORIAL_JOBS	
•ID_HISTORIAL_BACKUPS	int	•ID_HISTORIAL_JOB	int
•ID_BACKUP	int	•ID_BACKUP	int
•ID_CLIENTE	int	•NOMBRE_JOB	varchar(50)
•HOST_CLIENTE	varchar(50)	•FECHA_INICIO	Datetime
•NOMBRE_BACKUP	varchar(50)	◦FECHA_FIN	Datetime
•FECHA_INICIO	Datetime	•TAMANO	int
◦FECHA_FIN	Datetime	•RESULTADO_JOB	int
•RESULTADO_BACKUP	int	•RESULTADO_COPIA	varchar(50)
•RESULTADO_COPIA	varchar(50)	◦RESULTADO_ENVIO	varchar(50)
•RESULTADO_ENVIO	varchar(50)	•ID_CLIENTE	int
		◦HOST_CLIENTE	varchar(150)

Figura 5.14: Tablas empleadas por la aplicación Web

Podría buscarse una solución que evitara duplicar en ambas tablas información como es el caso del cliente, para ello deberían relacionarse ambas tablas. Se ha decidido la solución presentada por comodidad en las consultas y para facilitar la información de nuevos datos de manera simple. Es decir, si se desea introducir un nuevo historial de una tarea, no es necesario buscar el historial de backup al que va ligado, simplemente se inserta la información tal como llega en el servicio Web.

5.6. Diseño de la interfaz

5.6.1. Aplicación de escritorio para Windows



Figura 5.15: Diseño del panel de autenticación

La interfaz de usuario de la aplicación de escritorio ya se mostró en la fase de análisis. En ese momento se trataba de imágenes diseñadas con entornos gráficos y se comentó que

podría diferenciarse en algunos aspectos del resultado final de la aplicación. En esta fase del proyecto se pretende diseñar la interfaz de usuario haciendo uso de los componentes reales que se utilizarán. Este diseño será más realista y pese a imitar en todo lo posible en anteriormente mostrado, presenta algunas diferencias.

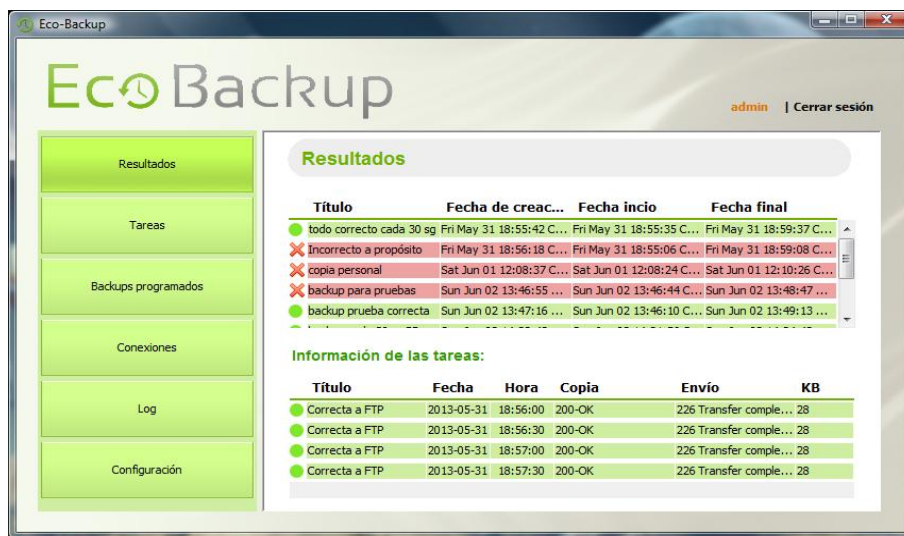


Figura 5.16: Diseño del panel de resultados

Se ha intentado realizar una aplicación usable y accesible en la cual se vea claro el punto en el que se encuentra el usuario. Se ha intentado realizar una aplicación visual, con colorido, pero siempre intentando representar la información con formas o texto además de con colores. Se puede apreciar que los resultados son coloreados según hayan sido realizados con éxito o hayan fracasado. Sin embargo, también se acompaña la información con un pequeño icono cambiante y denotador de la misma información. El icono cuenta además con un texto de ayuda que puede ser visualizado pasando el ratón por encima.

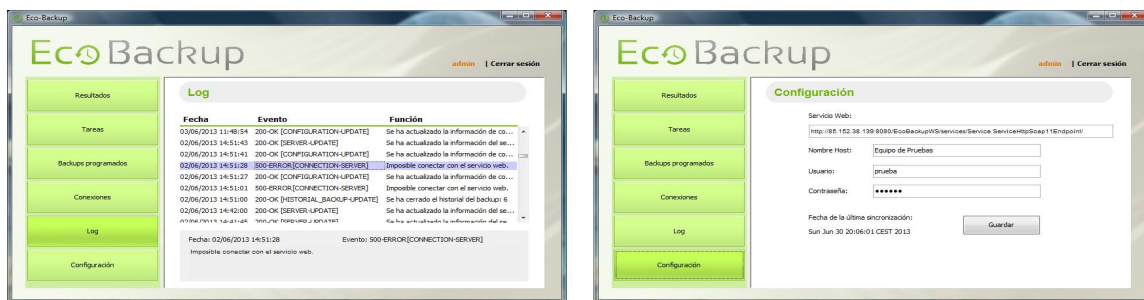


Figura 5.17: Diseño del panel de Log y de Configuración

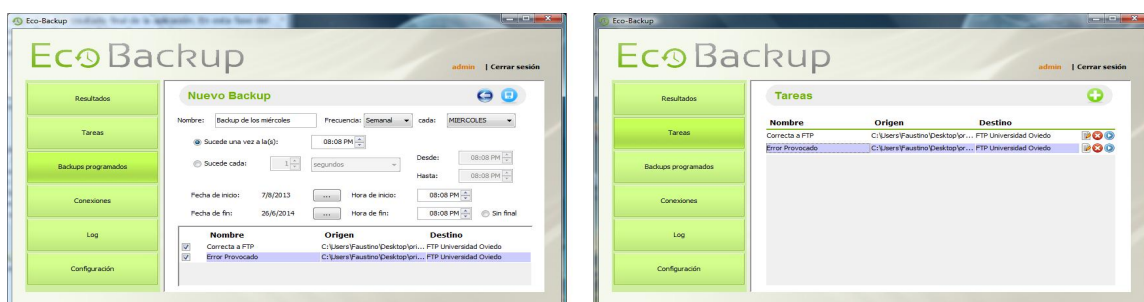


Figura 5.18: Diseño del panel de Log y de Configuración

5.6.2. Aplicación Web

Cliente	Fecha	Hora	Backup	Duración	Tamaño
Familia Cerezo	20/04/2013	10:00h	Farmatic	2 h.	1GB
Familia Cerezo	20/04/2013	10:00h	Farmatic	2 h.	400Mb
Familia Cerezo	20/04/2013	10:00h	Farmatic	2 h.	400Mb
Familia Cerezo	20/04/2013	10:00h	Farmatic	2 h.	400Mb

Figura 5.19: Diseño de la aplicación Web. Resultado de Backups

En las imágenes se pueden ver los diseños realizados por Ecocomputer S.L. para la interfaz de usuario de la aplicación Web. Se ha mantenido el mismo diseño que las aplicaciones que se encuentran desplegadas por la empresa en tan comentada plataforma.

Al igual que los diseños de la aplicación de escritorio, se trata de imágenes para tener como referencia pero no de diseños cerrados. Al realizar las hojas de estilo propias de la aplicación Web, se intentará copiar el diseño aquí presentado en la mayor medida.

Por otra parte, también se han tenido en cuenta temas de usabilidad y accesibilidad tal como será comentado en la sección de pruebas de esta documentación. Como adelanto, comentar que se evita el uso de colores como única forma de representación de información, contando con iconos de diferentes formas que apoyen la misma. Además, se han dotado tales iconos de un texto alternativo que facilite su lectura por navegadores en modo texto o lectores de pantalla.

Tarea	Fecha inicio	Duración	Tamaño	Resultado Backup	Resultado Copia	Resultado Envío
Copia Farmatic	10:00h	1h30min	800Mb	✓	✓	✓
Copia Farmatic	11:30h	30min	200Mb	✓	✓	⊗

Figura 5.20: Diseño de la aplicación Web. Resultado de Tareas

5.7. Especificación técnica del plan de pruebas

5.7.1. Pruebas unitarias

Tal como se ha comentado en el apartado de análisis de este documento, se realizarán pruebas unitarias haciendo uso de la herramienta JUnit. Principalmente se probarán las clases encargadas de la lógica compleja. Se realizarán pruebas para comprobar el correcto funcionamiento del formateador de programaciones a CRON, las clases encargadas de realización de copias, de establecer conexiones . . .

Pese a que el código será intensivamente probado por el cliente (Ecocomputer S.L.) antes de ponerlo en producción y que comiencen a utilizarlo los clientes finales, estas pruebas unitarias permiten la comprobación de cambios y localizar más fallos que un probado convencional, simplemente haciendo uso de la herramienta.

5.7.2. Pruebas de integración y del sistema

Mediante estas pruebas de integración se pretende comprobar que la combinación de cada módulo individual en el total funciona como es esperado. Se realizarán las definidas previamente en la fase de análisis. Habrá que tener en cuenta que las pruebas de visualización de interfaz son más directas y basadas en la experiencia del usuario: se comprobará su buen funcionamiento y eficacia de manera directa, observando la interfaz y los posibles cambios que en ella provoquen las operaciones de los demás módulos.

Por otra parte, dado que se utilizará durante el desarrollo del proyecto un entorno réplica del sistema de producción, el estudio de la integración será inherente al progreso del proyecto. Se utilizarán las mismas versiones que presentan los sistemas de la empresa para evitar problemas de incompatibilidad al migrar el resultado a los servidores de producción.

5.7.3. Pruebas de usabilidad y accesibilidad

Dada la naturaleza del proyecto, y la imposición por parte de Ecocomputer S.L. a implementar unas interfaces de usuario fijadas, no se ha destinado demasiado esfuerzo a garantizar la usabilidad y accesibilidad del proyecto. Sin embargo, se han tenido en cuenta pequeñas recomendaciones o buenas prácticas que no suponen una modificación elevada del diseño propuesto, y que mejoran estos aspectos.

El proyecto está pensado para ser utilizado por usuarios con conocimientos en informática, principalmente administradores de sistemas que suelen ser los encargados de realizar las copias de seguridad en una empresa. Este perfil de usuarios, presenta un gran manejo de las herramientas comunes de interacción con un sistema informático. Pero no por ello se deben descuidar aspectos que faciliten la manipulación de la herramienta.

Para el caso de la aplicación de escritorio se ha intentado permitir la interacción en todo momento haciendo uso únicamente de teclado. Por otra parte, se han trabajado aspectos relacionados con el foco, permitiendo al usuario teclear directamente sobre el primer campo del formulario sin necesitar desplazamientos o clicks de ratón.

Centrando el tema en las pruebas, el proyecto se ha desarrollado con el apoyo de la empresa cliente. Esto permite la recogida de información sobre posibles cambios o dificultades a la hora de utilizar la herramienta. Dado que se presentaron prototipos durante todo el proyecto, se han ido localizando cosas a mejorar, de igual manera, durante todo el proceso.

La aplicación Web, por otra parte, también cuenta con una estructura y un diseño fijado como requisito. No obstante, se realizaran pruebas de accesibilidad siguiendo las Pautas de Accesibilidad de Contenidos Web (WCAG) en su versión primera.

5.8. Decisiones del proyectante

Se ha incluido esta sección en la documentación de este proyecto, para concretar algunas decisiones llevadas a cabo por el proyectante. Algunas de las decisiones tomadas no se consideran la mejor solución posible pero era necesario llegar a un acuerdo con los intereses de la empresa.

5.8.1. Seguridad en las contraseñas

En los requisitos del proyecto se mencionaba que las contraseñas debían encontrarse en texto plano para que en caso de olvido por el usuario, un administrador de Ecocomputer pudiera visualizar la base de datos y conocerla.

Se considera un enorme fallo de seguridad mantener las contraseñas sin cifrar. Por ello se intento pactar la utilización de algoritmos de cifrado «irreversible». De tal manera que la comprobación de contraseñas se hiciera comparando los resultados después del cifrado. Esta solución no es posible dado que se necesita almacenar contraseñas de conexión, las cuales deben ser recuperables para establecer las mismas. Se intentó realizar un tipo de cifrado reversible basado en una palabra clave, sin embargo, esta solución no fue aceptada por Ecocomputer S.L.

La solución a la que se llegó después de negociaciones, fue la siguiente:

- **Cifrado MD5 de la contraseña del usuario.** La contraseña del usuario de la aplicación de escritorio será almacenada cifrada siguiendo el algoritmo MD5. En caso de que el cliente olvide su contraseña un administrador deberá consultar la base de datos y modificar introduciendo un nuevo valor MD5 generado o eliminar el usuario para que la aplicación solicite un nuevo registro.
- **Codificación de las contraseñas reversibles.** Dado que no se llegó a un acuerdo de cifrado de las contraseñas reversibles, se ha decidido codificar las mismas para evitar una lectura sencilla. La solución se considera inaceptable, pero los requisitos del cliente se consideran más importantes que las opiniones personales.

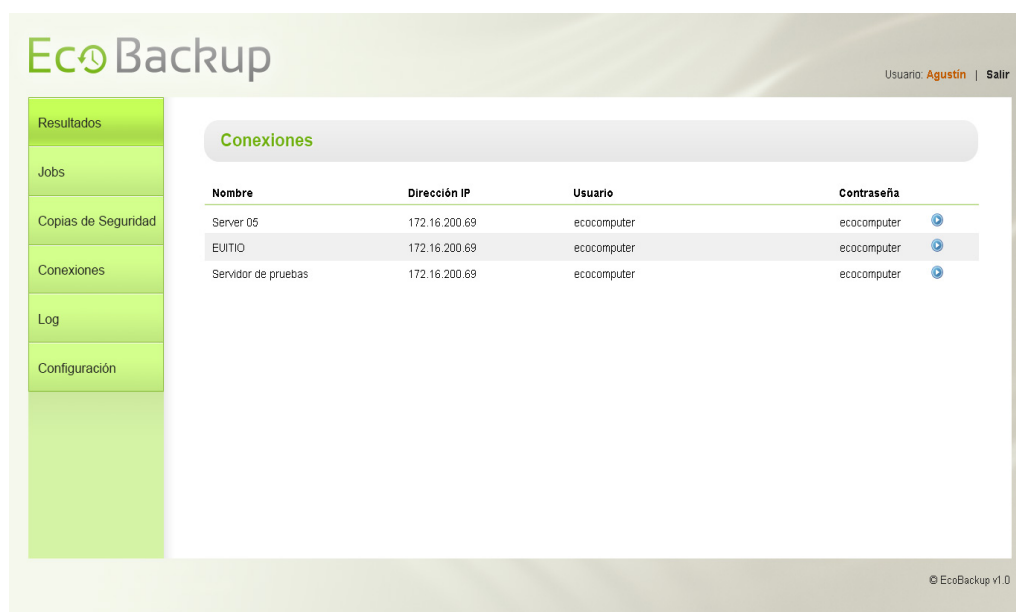


Figura 5.21: Diseño de la pantalla de conexiones con contraseñas visibles

Otra muestra de la poca importancia que se le daba a la seguridad de las contraseñas en este proyecto, podemos verlo en los diseños de la interfaz gráfica elaborados en la fase de análisis. En la imagen que acompaña esta sección, puede verse como en el diseño del

panel de conexiones se muestra la contraseña de los usuarios. No se ha seguido este diseño al considerarlo un fallo importante, y la empresa no ha puesto inconveniente en eliminar dicha información visible.

5.8.2. Código sencillo de mantener

Durante las fases de análisis y diseño se presentaron alternativas al finalmente implantado. Se barajó la posibilidad de utilizar frameworks de persistencia como «Hibernate³».

Hablando con el personal de Ecocomputer S.L. que deberá mantener el proyecto, confesaban no tener experiencia trabajando con Java, ya que siempre habían implementado tecnologías .Net. Reconocían que les presentaría un problema mantener un código en un lenguaje con el que no están acostumbrados y mucho más si se utilizaban frameworks específicos que deberían aprender para entender la estructura del proyecto.

Intentando facilitar las tareas de mantenimiento del proyecto, se han intentado utilizar las tecnologías más comunes y evitar el uso de frameworks que complicaran la comprensión del código.

Para el caso de la aplicación Web, el caso fue distinto. La empresa ya contaba con proyectos realizados con Struts2 y aprovechando algunas ventajas de Spring. Por esta razón, y previa consulta, se decidió utilizar los frameworks mencionados.

5.8.3. Base de datos sin eliminaciones

Otro aspecto de los requisitos iniciales del sistema que provocó intercambio de opiniones, fue el diseño de la base de datos. En la fase de análisis, Ecocomputer S.L. realizó un diagrama de tablas en el que todas presentaban un campo «eliminado», modificando las operaciones de eliminación de información, por actualizadores de ese campo con valor booleano a verdadero.

Desde la perspectiva personal, se considera necesario mantener alguna información importante de cara a posibles análisis o detección de errores. Pero la decisión de no borrar ningún registro de la base de datos supone un aumento considerable del peso de la misma. Por citar un ejemplo, cada operación realizada en la aplicación queda registrada en la tabla de LOG, es importante conservar estos registros, pero de nada valen una vez pasado el tiempo y si la aplicación es utilizada de manera frecuente podremos encontrarnos con miles de registros en pocos días.

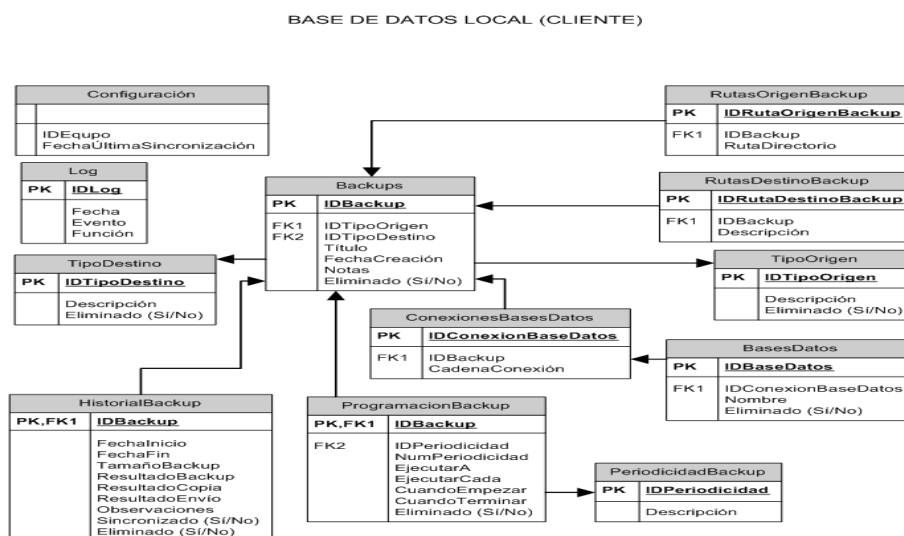


Figura 5.22: Diseño de la base de datos en el primer análisis del proyecto

³Sitio Web de Hibernate: <http://www.hibernate.org/>

Para evitar estos casos, se recomienda el uso de Triggers que se disparen al realizar nuevas inserciones borrando las entradas más antiguas. De esta forma se consigue transparencia para la aplicación y se consigue mantener la información de importancia.

Se acordaron modificaciones en la base de datos, eliminando el campo mencionado de algunas tablas y permitiendo el borrado de registros.

Otro aspecto que se modificó respecto al diseño de la empresa, fue la creación de tablas para mantener los valores que puede presentar un campo. En el diagrama que acompaña la sección se puede ver una tabla «PeriodicidadBackup», cuya funcionalidad es contener los valores: diaria, semanal, mensual y anual.

En el diseño final se ha utilizado restricciones de valores para evitar la creación de tablas del estilo mencionado. Se consigue el mismo efecto, mejorando las consultas y evitando la carga de tablas añadidas.

Capítulo 6

Implementación del sistema

6.1. Estándares y normas seguidos

6.1.1. XHTML 1.1

Acrónimo en inglés de eXtensible Hypertext Markup Language, se trata del lenguaje de marcado pensado para ser estándar en sustitución de HTML. Es solamente la versión XML de HTML, que tiene la misma funcionalidad pero cumple con las especificaciones de XML, más estrictas. Tiene como objeto la obtención de una web semántica, en la que estén claramente separadas la información a mostrar de la forma de mostrarla.

Durante el desarrollo de la aplicación se intenta seguir lo máximo posible las normas establecidas por este estándar. Sin embargo, dado que se utilizan componentes propios de Struts2 que generan código HTML, es probable que no se consiga un marcado que cumpla estrictamente con la norma.

6.1.2. CSS 3

Acrónimo en inglés de Cascading Style Sheets, u «hojas de estilo en cascada», es un lenguaje formal cuya función es la definir la presentación de un documento escrito en HTML o XML. Dichas hojas se pueden aplicar mediante un documento CSS externo llamado desde el archivo asociado, desde una hoja de estilo interna en el mismo documento, o insertando las propiedades CSS directamente dentro de las etiquetas HTML, si bien la opción recomendada por ser la más potente es la primera de ellas.

Aplicando hojas de estilo conseguimos las siguientes ventajas:

- Control centralizado de la presentación de un sitio web, agilizando considerablemente su actualización.
- Posibilidad de que el usuario pueda especificar desde el navegador su propia hoja de estilos para cubrir algunas posibles necesidades como, por ejemplo, aumentar el tamaño del texto o cambiarlo de color.
- Las páginas pueden disponer de diferentes hojas de estilo en función del dispositivo que las lea. Así, la visualización de la web no será la misma desde un ordenador de sobremesa que en un dispositivo móvil, por ejemplo.
- Aumenta la legibilidad del código HTML por contener menos información, además de reducir considerablemente el tamaño del documento.

En este proyecto se han heredado las hojas de estilo presentes en otras aplicaciones de la plataforma Web donde será desplegada la aplicación. De esta forma se mantiene el mismo aspecto y se consigue una mejor integración visual.

Por otra parte, se quiere destacar que el cumplimiento del estándar facilita el correcto visualizado de la aplicación en diferentes navegadores Web y dispositivos.

6.1.3. Estándares de programación

Pese a que no es necesario para la correcta ejecución del código fuente, sí que es muy recomendable aplicar los estándares de cada lenguaje de programación utilizado. Seguir las normas dictadas aporta principalmente mantenibilidad, entendimiento y legibilidad del código. Esto permite que la aplicación sea más fácil de entender y mantener por el propio autor original del código u otros programadores que deban hacerlo posteriormente.

Durante el desarrollo del proyecto se seguirán los estándares propios de cada lenguaje (organización física y lógica de ficheros, correcta declaración de objetos, adecuada indentación y separación ...).

Se han tenido en cuenta buenas prácticas relacionadas con las tecnologías utilizadas. La valoración de dichos consejos ayuda al desarrollo de un mejor código y más fácil de mantener por personas ajenas a su desarrollo.

6.1.4. Modelo-vista-controlador

Este patrón de arquitectura de software se encarga de separar la los datos de la aplicación, la lógica de negocio y la interfaz de usuario que representa la información y recoge las interacciones del usuario. Aplicar este patrón supone aplicar las ideas de reutilización de código y separación de conceptos, lo que facilita el desarrollo y posterior mantenimiento de una aplicación software.

Tal como puede verse en los diagramas de paquetes de la fase de análisis, se ha seguido este patrón arquitectónico en los módulos desarrollados.

6.1.5. Patrón de diseño DAO

Un Data Access Object (Objeto de Acceso a Datos) es un componente de software que suministra una interfaz común entre la aplicación y uno o más dispositivos de almacenamiento de datos, tales como una Base de datos o un archivo.

La utilización del patrón DAO, permite abstraer y encapsular los accesos a un repositorio de datos.

6.2. Lenguajes de programación

6.2.1. Java

Java¹ es un lenguaje de programación orientado a objetos desarrollado por Sun Microsystems a principios de los años 90. El lenguaje en sí mismo toma mucha de su sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros o memoria.

Las aplicaciones Java están típicamente compiladas en un código intermedio, aunque la compilación en código máquina nativo es posible. En el tiempo de ejecución, el bytecode es normalmente interpretado o compilado a código nativo para la ejecución.

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

Java es el lenguaje de programación utilizado en toda la lógica del proyecto que nos atañe. Además de ello, se utilizan componentes adicionales para lograr los objetivos deseados.

Struts

Struts² es una herramienta de soporte para el desarrollo de aplicaciones Web bajo el patrón modelo-vista-controlador para Java. Su utilización permite reducir considerablemente el tiempo de desarrollo, separando automáticamente las capas de interfaz de usuario, lógica de negocio y el acceso a datos. Esto lo realiza mediante Servlets, que se encargan de realizar las operaciones desencadenadas por los Actions del usuario en la interfaz HTML.

Este software libre fue creado originalmente por Craig McClanahan y donado a la Apache Foundation en el año 2000, pero no fue hasta 2005 cuando se convirtió en un proyecto de alta importancia para Apache. La versión empleada en este proyecto es la 2.2.1 por cuestiones de compatibilidad con el resto de software desplegado en la plataforma que se utilizará para el despliegue.

Spring Framework

Spring³ es un conjunto de herramientas para el desarrollo de aplicaciones en Java. Aunque Spring no impone ningún modelo de programación específico, se ha popularizado entre los programadores al ser considerado una alternativa al modelo de Enterprise JavaBeans.

Sus módulos proveen una gran variedad de servicios, técnicas y paradigmas, como un contenedor de Inversión de Control, programación orientada a aspectos, acceso a datos, gestor transaccional, modelo-vista-controlador, framework de acceso remoto, convención sobre configuración, procesamiento por lotes, autenticación y autorización, administración remota, mensajería y herramientas de testado.

Se trata de un software libre desarrollado por Rod Johnson cuya primera versión fue lanzada en 2004. En el desarrollo del proyecto se utilizará la versión 2.2.1 distribuida junto con las librerías de Struts.

Las librerías de Spring se utilizarán principalmente para lograr la inyección directa de dependencias.

¹Descarga de Java: <http://www.java.com/es/download/>

²Sitio Web de Struts: <http://struts.apache.org/>

³Sitio Web de Spring: <http://www.springsource.org/spring-framework>

Java Database Connectivity

Más conocido por sus siglas JDBC, se trata de una API que permite realizar de manera sencilla operaciones sobre una base de datos empleando el lenguaje Java, independientemente del sistema operativo y base de datos a la que se quiera acceder. Una vez establecida la conexión mediante los manejadores pertinentes, se puede realizar cualquier tipo de tarea con la base de datos, siempre y cuando se tengan en la misma los permisos adecuados, utilizando el conjunto de objetos e interfaces que pone a disposición del programador.

Su desarrollo depende de Sun Microsystems puesto que se integra en la Java Development Kit desde su versión 1.1 de febrero de 1997. Su actual versión, JDBC 4.1, está incluida en la Java SE 7 de julio de 2011.

La utilización de las librerías de JDBC en el desarrollo del proyecto se produce en todos los módulos del mismo, ya que al ser independiente del gestor de base de datos, se puede utilizar para conexiones con Microsoft SQLServer y con SQLite de igual manera.

Swing

Swing es una biblioteca gráfica para Java, que incluye componentes o widgets para crear interfaces gráficas de usuario. Una alternativa al uso de Swing, es AWT. Esta biblioteca estaba concebida como una API estandarizada que permitía utilizar los componentes nativos de cada sistema operativo. Sin embargo, la limitación de vinculación entre la aplicación Java y el sistema operativo hace de otras bibliotecas no vinculadas, como Swing, una mejor opción.

Quartz Scheduler

Como ya se ha comentado en anteriores secciones del proyecto, se utilizará la biblioteca mencionada para implementar la programación de ejecución de hilos. Se trata de un proyecto de código abierto que presenta muy buenas críticas entre sus usuarios y que requiere poco tiempo de aprendizaje.

La principal ventaja que se encontró al proyecto es que permite un encapsulamiento elevado de la funcionalidad. Haciendo uso de un formato como el admitido por CRON en Unix, simplemente se requiere una clase que implemente una interfaz concreta de la biblioteca y una programación temporal en el formato indicado.

Datatables

Datatables⁴ es una biblioteca JavaScript basada en JQuery, que permite añadir de manera sencilla, controles a una tabla. Se utilizará para permitir el filtrado de registros de la tabla, la limitación de filas por página, así como permitir su ordenación por columnas.

⁴Sitio Web de Datatables: <https://datatables.net/>

6.3. Herramientas y programas usados para el desarrollo

6.3.1. Eclipse

Eclipse es un entorno de desarrollo multiplataforma utilizado para la realización de proyectos, que da soporte a una gran cantidad de lenguajes de programación, como son C, C++, Fortran, Java, Perl, PHP, Python o Ruby, entre otros, gracias a las herramientas de desarrollo que incorpora y la posibilidad de instalar plugins adicionales.

El entorno de desarrollo integrado emplea módulos que pueden ser activados y desactivados en función de las necesidades de cada usuario, lo que le confiere una ligereza de ejecución considerable. Eclipse provee al programador de una serie de frameworks que facilitan la labor del desarrollador, así como un compilador y herramienta de depurado Java.

Se trata de una aplicación de software libre mantenida por la Eclipse Foundation, fundada en 2003, y su comunidad de usuarios, pese a que inicialmente fue producto de un proyecto de IBM. Su última versión estable, la 4.2 que se conoce bajo el nombre de «Juno», es la empleada en el desarrollo de este proyecto.

Se ha elegido Eclipse frente a otras alternativas como Netbeans, por la experiencia previa del entorno, así como por la integración que presenta con frameworks utilizados.

6.3.2. SQL Server Management Studio Express

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

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

6.3.3. Enterprise Architect

La aplicación Enterprise Architect, desarrollada por Sparx Systems, es una completa herramienta con la que crear todo tipo de diagramas UML. Utiliza la última especificación de UML y dispone de gran cantidad de funcionalidades extra además de las posibilidades de diseño que ofrece. Se trata de una herramienta comercial, aunque los alumnos de la universidad contamos con licencia gratuita.

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

Enterprise Architect ha sido usado en el desarrollo de este proyecto como herramienta de modelaje de algunos diagramas, los cuales se pueden ver en las diferentes secciones de esta documentación.

6.3.4. Dia

Dia⁵ es una aplicación con la que crear diagramas, desarrollada como parte del proyecto GNOME de sistemas Unix, aunque también existe una versión para Windows. Se trata de un programa gratuito de código abierto distribuido bajo la Licencia Pública General de GNU.

⁵Sitio Web de Dia: <https://projects.gnome.org/dia/>

Con Dia se pueden modelar diferentes tipos de diagramas, entre los que se encuentran los de UML, de flujo o incluso de circuitos eléctricos. Gracias al paquete dia2code, Dia también permite generar el esqueleto del código fuente a partir del diagrama UML diseñado. La aplicación fue originalmente creada por Alexander Larsson hasta que James Henstridge tomó su relevo como desarrollador jefe. Actualmente es mantenido por una serie de desarrolladores independientes. Su última versión estable, la 0.97.1, data de enero de 2010.

La herramienta ha sido utilizada para crear los diagramas entidad-relación de los sistemas de bases de datos, dichos diagramas pueden encontrarse en las secciones oportunas de este documento.

6.3.5. Microsoft Office Project

Project se trata de una aplicación desarrollada por Microsoft cuya función es la de administrar proyectos para poder mantener un control sobre el trabajo, la evolución del desarrollo, los plazos temporales y las finanzas, entre otros. Se trata de un software comercial que fue creado para DOS en 1984 a partir de las especificaciones de Alan M. Boyd de Microsoft, aunque fue una empresa de Seattle quien lo desarrolló para que Microsoft comprase los derechos en 1985.

Entre sus tareas se pueden destacar la posibilidad de organizar diferentes tareas, personal dedicado a cada una y el presupuesto previsto; la visualización y gestión del progreso, solucionando problemas del proceso y prediciendo posibles escenarios; y el control de las finanzas de los proyectos pudiendo elaborar presupuestos.

6.3.6. TeXworks

TeXworks es un entorno de edición y trabajo con L^AT_EX. Resulta muy cómodo ya que permite la generación del PDF resultado y mantiene una integración referencial entre ambos documentos.

Se ha decidido elaborar la documentación de este proyecto con esta herramienta y este sistema de composición de textos, debido a que se considera más profesional para documentos largos y con carácter científico.

6.4. Creación del sistema

6.4.1. Problemas encontrados

A continuación se enumeran algunos problemas encontrados durante el desarrollo del proyecto.

Retrasos por cambios en los requisitos

El procedimiento seguido una vez se empezó la implementación, fue la de desarrollar primero la aplicación de escritorio por tratarse del eje central del proyecto y el que más complejidad presenta. Se empezó realizando un prototipo de la interfaz de usuario sin funcionalidad para comprobar que se habían entendido las necesidades del cliente.

En este punto se produjo el siguiente problema. Como representantes de los intereses de Ecocomputer S.L. y encargados del control del proyecto, se encontraban dos personas. En las fechas en las que se organizó la reunión de avance para mostrar el prototipo de interfaz, una de ellas se encontraba de vacaciones y por lo tanto solo se recogieron las opiniones al respecto del otro miembro. La interfaz contaba con la aceptación de dicha persona y pese a realizar algunas pequeñas modificaciones se contó con «luz verde» para seguir desarrollando.

Pasado el tiempo y cuando ya se tenía un prototipo funcional, basado en la interfaz de usuario anteriormente comentada, se procedió a concretar una nueva reunión con la empresa para mostrar los avances. En esta ocasión, el que se encontraba ausente era el miembro que había acudido a la primera reunión. Las conclusiones obtenidas en este nuevo punto del proyecto, fueron que los requisitos del proyecto no se encontraban claros. Al ver la interfaz funcional, la otra persona encargada del proyecto, pensó que no era lo que se necesitaba. Se acordaron multitud de cambios en la interfaz y en la funcionalidad de la herramienta desarrollada. Tanto es así, que se vio modificado hasta el diseño de la base de datos.

Todos estos cambios, al ser sugeridos únicamente por un encargado de la empresa, fueron anotados y se retrasó el proyecto hasta poder realizar una reunión conjunta con los dos directores del proyecto. En esta reunión se acordaron algunos de los cambios anteriormente propuestos y se llegó a la especificación de requisitos que finalmente se implantó.

Todas estas demoras y modificaciones no previstas en la planificación, fueron posibles gracias a la simplificación de otros módulos del proyecto, eliminando funcionalidades que finalmente no consideraban necesarias.

Indisponibilidad de los recursos proporcionados por el cliente

Una vez se comenzó el desarrollo de los módulos Web, se hizo necesario conocer el entorno donde serían desplegados para su puesta en producción. Se pretendía obtener un servidor de pruebas y desarrollo que fuera similar para evitar posibles incompatibilidades en la migración del proyecto.

La empresa no pudo proporcionar un servidor de pruebas y desarrollo, y únicamente facilitó un acceso al servidor de despliegue y producción y al sistema de gestión de bases de datos que utilizan el resto de proyectos.

No se consideró recomendable la utilización de dichos recursos durante la realización del proyecto, por lo que se tuvo que realizar una infraestructura espejo en un equipo personal, con el que trabajar sin riesgo a poner en peligro la integridad de los equipos o los datos de la empresa cliente. El tiempo dedicado a la instalación y configuración de un servidor y un sistema de gestión de bases de datos similares a los presentados por la empresa, provocó retrasos en la planificación del proyecto.

Realización de proyectos integrados en paralelo

Como se ha comentado anteriormente en este documento, este proyecto iba a ser un producto cerrado, sin embargo, los requisitos cambiaron debido a que se estaban desarrollando otros proyectos Web encaminados a los mismos clientes y podría implementarse una plataforma común.

La realización de dicha plataforma se produjo de manera paralela temporalmente al proyecto que nos ocupa. Por esta razón se iban integrando módulos sin probar ni terminar. Dado que como se ha comentado, no se contaba con un servidor de pruebas ni desarrollo, cuando se tenía una versión estable del proyecto se publicaba en los servidores de producción para que lo analizaran en Ecocomputer S.L.

Los problemas encontrados, se basaban en la multitud de versiones y cambios en las partes a integrar. Se contaba con un producto probado y funcional y a la hora de subirlo al servidor de la empresa fallaba la integración, debido a que éste contaba con una nueva versión no contemplada de la gestión de usuarios.

La solución llevada a cabo se basó en una comunicación constante con los desarrolladores de otros módulos. De esta forma se conocían los cambios y podían anticiparse las medidas correctoras para garantizar la integración. Sin embargo, los retrasos en el proyecto al no contar con una solución definitiva de los módulos a integrar fueron constantes.

Problemas de disponibilidad de la empresa

Durante el desarrollo del proyecto surgieron dudas y necesidades que debían ser resueltas por Ecocomputer S.L. Por esta razón se realizaron numerosas reuniones con finalidades distintas.

La problemática que aquí se recoge, tiene que ver con la disponibilidad de la empresa para organizar las mismas. Pese a mostrar un gran interés por el proyecto y presentar todas las facilidades a su alcance, durante el proyecto personal encargado de la supervisión del mismo se vio ausentada debido a períodos de vacaciones u otros compromisos. Estas ausencias provocaron retrasos en el proyecto al paralizar el desarrollo a la espera de concretar requisitos.

Por otra parte, algunos problemas laborales, provocaron retrasos en diseños gráficos. Hay que tener en cuenta que el diseñador además de elaborar los iconos del sistema, realizó bocetos de cómo debía verse la interfaz de usuario y la estructura de los layouts. Por esta razón, la demora en sus diseños produjo retrasos en el proyecto respecto a su planificación.

Como solución, se decidió desarrollar con iconos propios y siguiendo una estructura visual amigable, que permitiera continuar con el desarrollo y presentar prototipos funcionales. De esta forma, una vez realizados los diseños finales, los cambios se podrían aplicar sobre un producto funcional completo.

Desconocimiento del entorno y las tecnologías

Algunas tecnologías empleadas en el proyecto, no se habían utilizado con anterioridad. Este desconocimiento provocó retrasos para comprender y estudiar la tecnología. Un ejemplo claro, dejando a un lado frameworks y bibliotecas utilizadas, fue el sistema de gestión de base de datos impuesto por el cliente, Microsoft SQL Server 2005. Se presentaron dudas respecto a los tipos de dato soportados y las diferencias entre éstos y los datos manejados por el otro sistema utilizado en el proyecto, SQLite. Principalmente, el uso de fechas con hora incluida, ya que en unos casos se representa mediante un tipo de dato y en otros con otro, y debe ser recuperado desde JDBC en objetos Java distintos.

6.4.2. Descripción detallada de las clases

Para describir cada clase se han utilizado comentarios JavaDoc. Dado que se ha generado un elevado número de páginas de información, se ha incluido al final de este documento.

Se ha utilizado un Doclet⁶, para formatear la salida del javadoc a formato PDF y así, poder incluir la información en este documento. La salida habitual de javadoc es HTML, para conseguir la misma información en otros formatos, como en este caso, se deben utilizar herramientas Java encargadas de especificar el contenido y el formato de la salida. Estas herramientas son los llamados Doclets.

⁶Información sobre Doclets: <http://es.wikipedia.org/wiki/Doclets>

Capítulo 7

Desarrollo de las pruebas

En esta sección se detallan las pruebas realizadas durante el proyecto. Se ha realizado las pruebas planificadas y comentadas en este documento en la sección de diseño.

7.1. Pruebas unitarias

Tal como se ha comentado anteriormente en este documento, se han realizado pruebas unitarias del código, haciendo uso del framework JUnit 3. Las pruebas realizadas comprueban el correcto funcionamiento de los métodos con más lógica del proyecto. Estas pruebas ayudaron al correcto desarrollo de las funcionalidades, así como, a localizar fallos en su funcionamiento.

7.1.1. Pruebas del programador de tareas

Formateador de CRON

Se ha realizado una batería de pruebas, para analizar el correcto funcionamiento de la clase encargada de formatear la programación introducida por el usuario en el formulario de la aplicación, y convertirla a una secuencia CRON.

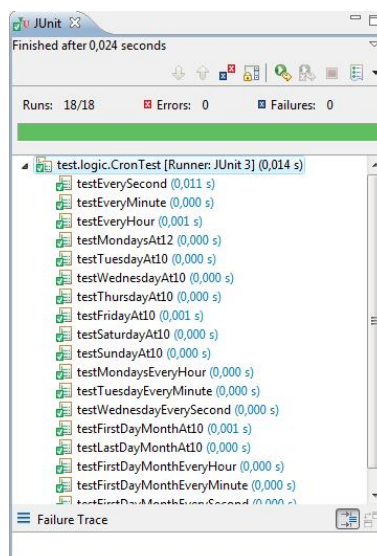


Figura 7.1: Captura del resultado de JUnit al pasar las pruebas del formateador de CRON

Con la intención de realizar un probado completo del módulo, se han creado 18 casos distintos. Se pretende que cada caso de prueba, verifique el correcto formateado de una programación distinta. El número de combinaciones posibles en la programación, es demasiado elevado para probar todos los casos, pero conociendo los valores que pueden

ocasionar más problemas a la hora de ser formateados, se puede crear un subconjunto de pruebas bastante razonable.

Todos los casos de prueba presentan la misma estructura. Dado que el valor formateado se retorna como una cadena de texto, bastará con crear un nuevo backup con la programación que se desea probar y ejecutar el generador del código CRON. Únicamente se debe probar que la cadena resultado coincide con el valor esperado.

En caso de querer realizar pruebas similares, se presenta en la siguiente tabla los 18 casos realizados.

Prueba	Resultado esperado
Cada segundo, todos los días desde las 10:20 hasta las 10:40h	0/01 20-40 10 1/1 * ? *
Cada minuto, todos los días desde las 10:20 hasta las 10:40h	0 20-40/01 10 1/1 * ? *
Cada hora, todos los días desde las 10:20 hasta las 12:20h	0 20 10-12/01 1/1 * ? *
Todos los lunes a las 10:00h	0 00 10 ? * MON *
Todos los martes a las 10:00h	0 00 10 ? * TUE *
Todos los miércoles a las 10:00h	0 00 10 ? * WED *
Todos los jueves a las 10:00h	0 00 10 ? * THU *
Todos los viernes a las 10:00h	0 00 10 ? * FRI *
Todos los sábados a las 10:00h	0 00 10 ? * SAT *
Todos los domingos a las 10:00h	0 00 10 ? * SUN *
Cada hora, los lunes desde las 10:00 hasta las 14:00h	0 00 10-14/01 ? * MON *
Cada minuto, los martes desde las 10:00 hasta las 14:00h	0 00/01 10-12 ? * TUE *
Cada segundo, los miércoles desde las 10:00 hasta las 14:00h	0/01 * 10-12 ? * WED *
El primer día de cada mes a las 10h	0 00 10 1 1/1 ? *
El último día de cada mes a las 10h	0 00 10 L 1/1 ? *
El primer día de cada mes, cada hora desde las 10:00 hasta las 14:00h	0 00 10-14/01 1 1/1 ? *
El primer día de cada mes, cada minuto desde las 10:00 hasta las 12:00h	0 00/01 10-12 1 1/1 ? *
El primer día de cada mes, cada segundo desde las 10:00 hasta las 12:00h	0/01 * 10-12 1 1/1 ? *

Cuadro 7.1: Tabla de casos de prueba utilizados para testear el módulo

Compresor de directorios

Otra funcionalidad que se deseaba probar, es el módulo encargado de comprimir un directorio a un fichero ZIP. Éste será utilizado para permitir la subida de copias de directorios a servidores FTP. La prueba contempla un único caso, dado que la prueba es concreta de una funcionalidad.

El procedimiento de la prueba consiste en comprimir un directorio de pruebas, comprobar que no se produce ninguna excepción en el proceso, asegurar la existencia del fichero comprimido recién creado y eliminar el mismo.

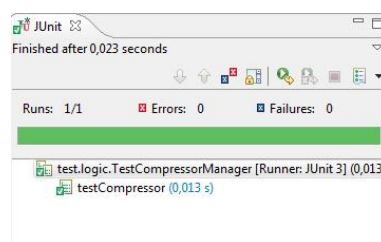


Figura 7.2: Captura del resultado de JUnit al pasar las pruebas del compresor de directorios

Realizador de Backups en local de un fichero o directorio

Se realizarán copias automáticas de un directorio y de un fichero concreto de pruebas, para asegurar que el módulo encargado de realizar las copias en local funciona como debe. La prueba se basará en realizar la copia y asegurar la existencia del fichero o directorio resultado. Adicionalmente se comprobará que el tamaño de origen y destino es el mismo.

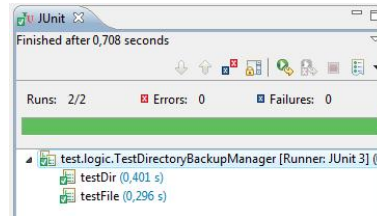


Figura 7.3: Captura del resultado de JUnit al pasar las pruebas para una copia local

Por último, una vez realizada la prueba se eliminarán los ficheros resultado. De esta forma se asegura que al volver a pasar la misma, se comprueba la generación de nuevos resultados y no se validan los anteriormente creados.

Publicador de copias en un servidor FTP

Para probar este módulo, se ha creado una clase de pruebas propia. Sin embargo, no se ha automatizado el proceso haciendo uso de JUnit como en los casos anteriores. Para la realización de esta prueba es necesario comprobar los resultados.

La naturaleza de esta prueba se debe a que no se ha implementado un sistema que descargue ficheros de un servidor FTP, al no ser necesario para la funcionalidad del proyecto. Se puede subir un fichero y comprobar que no se produjeron excepciones, sin embargo no se puede asegurar que el fichero publicado sea correcto, debemos acceder al servidor y descargarlo de manera manual para realizar la comprobación.

Realizador de copias de una base de datos

De manera similar a las pruebas realizadas para copias en local, se ha implementado un sistema de pruebas para copias de base de datos. El método utilizado es similar, se realiza la copia y se comprueba la existencia del fichero .bak resultado.

Por otra parte, se considera necesaria la intervención manual para cargar dicho fichero y comprobar que podemos restaurar el sistema con la copia realizada.

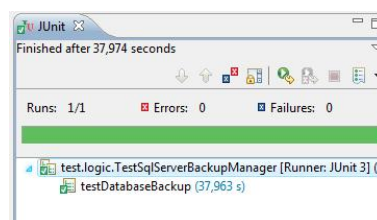


Figura 7.4: Captura del resultado de JUnit a las pruebas de copia de una base de datos

7.2. Pruebas del sistema

A continuación se detallan pruebas de sistema para comprobar la correcta integración de los módulos. Se pretende realizar un guión a seguir que permita comprobar el funcionamiento deseado de todas las capas y subsistemas que componen el proyecto.

- **Inicio de la aplicación por primera vez:** Se debe comprobar que se activa un panel de registro
- **Registrar un nuevo usuario en el sistema:** Comprobar que una vez registrado, el sistema presenta un usuario autenticado en la parte superior derecha de la ventana y se tiene acceso a toda la funcionalidad de la misma. Cerrar la aplicación y comprobar que al volver a lanzar la misma, no se presenta panel de registro.
- **Creación de una tarea:** Crear una tarea con la herramienta y comprobar que al finalizar la creación, aparece ésta disponible en la lista y puede ser editada, eliminada y lanzada. Comprobar además, que al crear un nuevo backup, la tarea creada se muestra disponible para ser añadida.
- **Cerrar sesión en el sistema:** Comprobar que se modifica la información de usuario autenticado en la parte superior derecha de la ventana y se muestra el panel de autenticación al intentar hacer uso del menú vertical.
- **Creación de conexiones:** Crear una conexión a una base de datos y a un servidor FTP, en el panel destinado para ello. Comprobar que al añadir cada una de estas conexiones, se muestran disponibles en la tabla correspondiente, siendo posible su modificación, eliminación y comprobación de disponibilidad. Comprobar además, que al crear una nueva tarea podemos referenciar las conexiones recién creadas.
- **Creación de un nuevo backup para un futuro próximo:** Crear una programación de copia de seguridad para un futuro próximo y añadirle una tarea creada anteriormente. Comprobar que al terminar la programación del mismo, éste aparece en la tabla de backups disponibles en el sistema. Esperar al momento elegido en la programación y comprobar que la copia ha sido realizada. Analizar la información del panel de resultados para comprobar el registro del backup que nos ocupa. Acceder a la aplicación Web y comprobar que la misma información de resultados se encuentra disponible en la misma.
- **Comprobación de persistencia de los datos:** Una vez realizadas las pruebas anteriores, probar a cerrar la aplicación. Lanzar de nuevo la herramienta y comprobar que todos los datos introducidos se muestran accesibles de igual forma que si la aplicación no hubiera sido cerrada.
- **Comprobación del correcto uso del fichero de Log:** Una vez realizadas las pruebas anteriores, comprobar el panel de Log. Asegurarse de que todas las operaciones realizadas aparecen reflejadas en dicho registro con la información oportuna.
- **Comprobación de respuesta a errores:** Realizar un backup con tareas erróneas (de un fichero inexistente, a un servidor FTP inaccesible, de una base de datos con un usuario erróneo . . .) Comprobar que el sistema muestra el fallo en el registro de Log y en los resultados del backup una vez realizado. Asegurar que la aplicación no deja de funcionar por encontrar fallos. Visualizar la información del fallo en la aplicación Web y asegurarse de que el mensaje mostrado coincide con el error producido.

7.3. Pruebas de usabilidad y accesibilidad

7.3.1. Pruebas de usabilidad

Una vez desarrollado el sistema, se han realizado pruebas de usabilidad de todo el proyecto. Para ello se ha contado con la colaboración de seis personas ajenas al desarrollo del mismo. Se deseaba realizar las pruebas con futuros clientes reales de la aplicación, sin embargo, problemas acontecidos en Ecocomputer S.L. imposibilitaron las mismas. Por esta razón se han elegido seis personas de distintas edades y con unos conocimientos informáticos distintos.

Por un lado contamos con dos probadores que podrían ser descritos como «nativos digitales», dos personas de mediana edad que habitualmente no hacen uso de sistemas informáticos y dos ingenieros informáticos con conocimientos en administración de sistemas. Estos últimos, se consideran los más próximos al perfil de los usuarios del proyecto. Se ha decidido buscar parejas de probadores similares para realizar dos pruebas diferenciadas.

Al primer grupo se le entregará el manual de usuario del sistema (contenido en este documento) y se le pasará una hoja de tareas a realizar con el proyecto. De esta forma comprobamos que el tutorial creado es correcto y entendible, y por otra parte que la aplicación utilizable por una persona que disponga del mismo.

El segundo grupo no contará con el manual de usuario. Por otra parte, se realizará una demostración de uso antes de pedir que lleven a cabo las tareas propuestas.

Las tareas a realizar por los dos grupos serán las mismas para conseguir unos resultados comparables. A continuación se enumeran las pruebas a realizar, para posteriormente analizar los resultados obtenidos por los seis sujetos y obtener unas conclusiones sobre las pruebas realizadas.

Tareas a realizar por los probadores del sistema

- **Registro de un usuario en la aplicación cliente:** Se pide crear el usuario «Paco», en la aplicación cliente, con la contraseña «contraseña2013 »y nombrando el equipo como «Pc de pruebas »
- **Cerrar la aplicación y volver a abrirla, e iniciar sesión:** Se pide cerrar la aplicación y volver a abrirla. Una vez relanzada, se debe iniciar sesión en el sistema con el usuario creado anteriormente.
- **Añadir una nueva conexión a un servidor FTP:** Crear la conexión «FTP de pruebas», al servidor «156.35.95.193», con el usuario: «Paco», y contraseña: «Paco13».
- **Probar la conexión creada y modificar su información en caso de ser incorrecta:** Se pide comprobar si la conexión creada anteriormente es correcta y se puede establecer la comunicación con la misma. En caso de no ser así, se debe modificar la información para que el nombre de usuario sea: «Paco13», y contraseña: «PacoFTP13». Comprobar la disponibilidad nuevamente.
- **Crear una nueva tarea:** Se pide crear una tarea nueva que permita la copia del directorio «C:/DirPruebas», al servidor FTP anteriormente comentado. Ejecutar la tarea y comprobar mediante el registro de Log si se ha realizado la copia correctamente. Acceder al servidor FTP y comprobar que se ha subido el fichero creado en la copia. Para acceder, abra con su navegador Web la siguiente dirección: ftp://156.35.95.193 e introduzca los datos de usuario facilitados anteriormente.
- **Crear un nuevo backup:** Se pide crear un nuevo backup que se ejecute todos los días, a las 10 de la mañana y se repita cada media hora hasta las 5 de la tarde. Esta programación debe realizarse hasta el primer día del año próximo. El backup debe lanzar la tarea anteriormente creada, así como realizar una copia del mismo

directorio al destino local: «C:/Resultados/DirPruebas». Cerrar la ventana si cerrar la aplicación para permitir la ejecución del backup.

- **Consultar la información online:** Acceder a la plataforma Web y consultar la información relativa a los backups realizados. Acceder al desglose por tareas de la programación anteriormente creada. (Los datos de acceso a la plataforma Web deben ser entregados con la hoja de pruebas)
- **Modificar los datos de usuario de la aplicación de escritorio:** Modificar los datos de conexión con el servicio Web. Restaurar la ventana de la aplicación, anteriormente cerrada y modificar el usuario de conexión al servicio Web por el siguiente: «Francisco», contraseña: «Francisco2013». Se pide sincronizar con el servicio.

Conclusiones de las pruebas

Las pruebas realizadas han servido para asegurar que el sistema es sencillo de utilizar, aunque se han encontrado algunos patrones de error repetidos en los sujetos de prueba. Se recogen a continuación las dificultades presentadas y las conclusiones obtenidas con el método seguido.

- **Tutorial correcto pero demostración necesaria:** Se considera que el tutorial creado es correcto, ya que permite que los usuarios trabajen con la aplicación. Sin embargo el primer grupo de pruebas, presentó mayores dificultades que el grupo que había presenciado una demostración de uso del proyecto. Se recomienda por lo tanto, que algún responsable de Ecocomputer S.L. realice una demostración o un pequeño cursillo con los clientes finales del producto.
- **Dificultad para comprender que la aplicación sigue ejecutándose pese a no estar visible:** Algunos sujetos de prueba, los que menor experiencia presentan con ordenadores y sistemas informáticos, muestran algunas dificultades para comprender que pese a cerrar la ventana principal de la aplicación, ésta se mantiene en ejecución y se puede volver a mostrar la ventana, haciendo uso del botón situado en la barra de notificaciones del sistema operativo. Se considera necesario recalcar en la demostración o cursillo, que no debe lanzarse una nueva aplicación, cuando una se encuentra en ejecución.
- **Intentos fallidos en la creación de un backup debido a que no se encuentra añadida la tarea previamente:** Un error cometido por varios sujetos de prueba, consistía en la intención de crear un backup antes de añadir la tarea al sistema. Se considera de utilidad, añadir en el panel de programación de un backup, un botón que despliegue un diálogo de creación de tareas, de manera análoga a como se permite el añadido de conexiones a la hora de crear una nueva tarea. Sin embargo, se han respetado las interfaces pedidas por Ecocomputer S.L.
- **Creación incorrecta de una copia en local, debido a la falta de un nombre de fichero destino:** Otro error repetido por algunos sujetos de prueba, fue la creación de copias en local sin establecer un nombre para el fichero o directorio resultado. Un correcto uso de la herramienta implica la selección de un directorio para la copia haciendo uso del selector, pero además, una modificación de la ruta establecida para añadir el nombre con el que se desea almacenar la copia. Se debería establecer un campo adicional para el nombre de la copia, que permitiera una mejor percepción de la necesidad del mismo. Nuevamente, no se ha realizado para respetar las interfaces diseñadas por Ecocomputer S.L.

7.3.2. Pruebas de accesibilidad

Visión con diferentes navegadores gráficos

Dado que se presenta una aplicación Web, y que para acceder a la misma se debe hacer uso de un navegador, será necesario garantizar que la información se encuentra accesible para los navegadores más comunes.

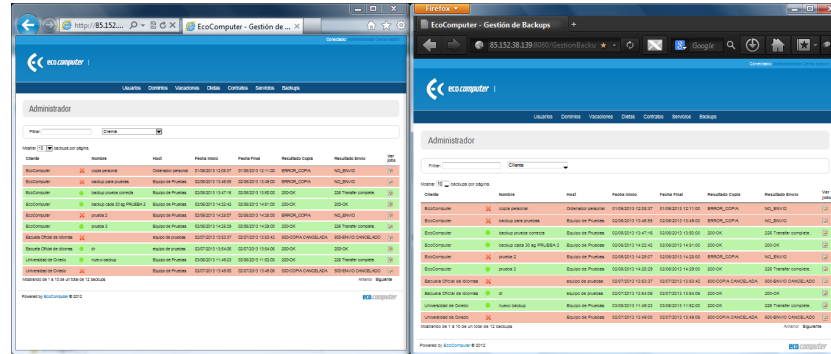


Figura 7.5: Captura de la aplicación Web visualizada por distintos navegadores (1)

Se han utilizado los siguientes navegadores para realizar dicha prueba:

- **Internet Explorer:** Navegador de Microsoft en su versión 9.0.17
- **Firefox:** Navegador desarrollado por Mozilla en su versión 22.0
- **Opera:** Navegador de Opera Software en su versión 12.16
- **Safari:** Navegador desarrollado por Apple en su versión 5.0.4

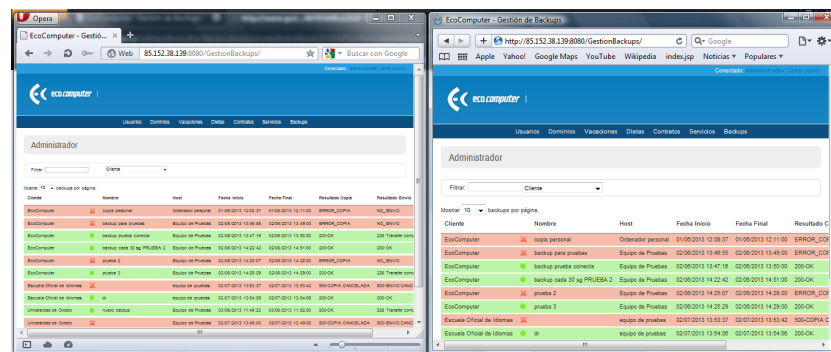


Figura 7.6: Captura de la aplicación Web visualizada por distintos navegadores (2)

En las imágenes que acompañan se puede comprobar que la visualización es correcta en todos los navegadores mencionados.

Se ha realizado una prueba adicional, consistente en probar la visualización desde un dispositivo móvil. Para ello se ha empleado un iPhone 4, con el navegador Safari. Se puede comprobar que la aplicación puede ser utilizada desde este tipo de dispositivos pese a que no se ajusta al tamaño de la pantalla como sería de desear.

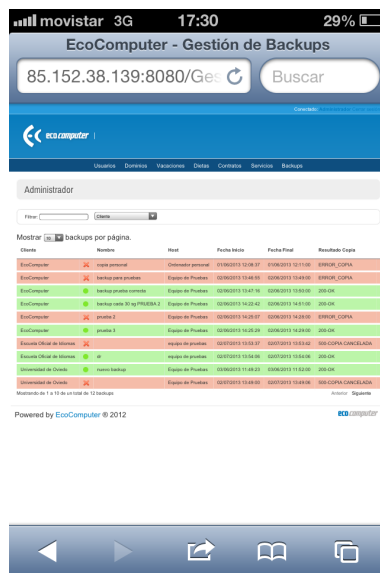


Figura 7.7: Captura de la aplicación Web visualizada por un dispositivo móvil

Utilización de aDesigner

Se ha utilizado la herramienta para el estudio de la accesibilidad, aDesigner, propiedad de IBM. En ella se pudo apreciar la visualización de la página por personas con problemas de visión y distinción de colores.

En la captura que acompaña este texto, se puede comprobar el resultado de la aplicación utilizada. Se visualiza como hay problemas en el tamaño de las fuentes y en el contraste entre algunos colores y otros. Se puede utilizar la aplicación sin necesidad de distinguir correctamente los colores, ya que se presenta información adicional para conocer el resultado del backup.

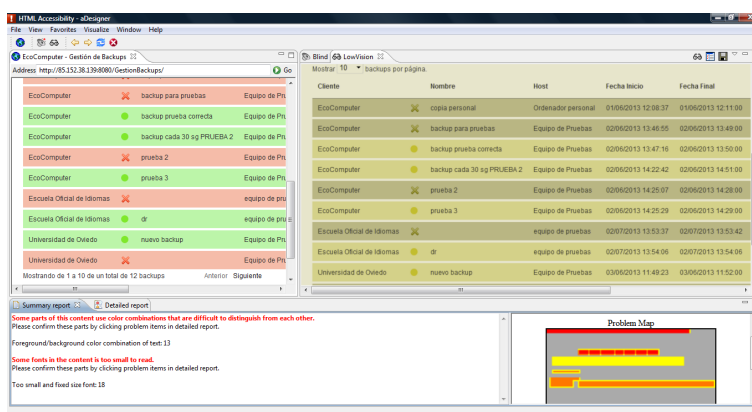


Figura 7.8: Captura de la herramienta aDesigner, para el estudio de la accesibilidad

Por otra parte, aDesigner, permite recorrer la página con un navegador para personas invidentes. La herramienta recorre la página leyendo el texto de la misma. Dado que el uso de estas herramientas requiere cierto entrenamiento, se considera muy dificultosa la interacción con la aplicación por este tipo de usuarios. Sin embargo, se considera bien diseñada para ser leída por estas herramientas.

Los aspectos mejorables serán comentados a EcoComputer S.L. por si admiten cambios de la interfaz de usuario en futuras versiones para mejorar los aspectos accesibles.

Accesibilidad de la aplicación de escritorio

La herramienta de escritorio también debe cumplir unos mínimos de accesibilidad. Por esta razón se han tenido en cuenta los siguientes aspectos para mejorar la interacción con la aplicación.

- **Evitar presentar información únicamente mediante colores:** Se emplean colores para representar el estado de los backups. Sin embargo, estos colores de fondo son acompañados por iconos representativos del mismo estado. De esta forma si una persona no puede distinguir los colores elegidos para la representación, podrá obtener la misma información visualizando los iconos.
- **Imágenes con texto alternativo:** Se utilizan iconos para representar funcionalidades o información. Por esta razón, es importante un uso correcto de textos alternativos. De esta forma si no se entiende una imagen, pasando el ratón por encima, se podrá obtener un texto explicativo.
- **Aplicación accesible mediante teclado:** Se ha desarrollado la herramienta teniendo en cuenta que ésta fuera utilizable haciendo únicamente uso de un teclado. De esta forma, personas con incapacidad para utilizar el ratón, podrá usar la aplicación sin mayores inconvenientes.
- **Evitar el uso de información sonora:** Se ha evitado el uso de información sonora para permitir la utilización de la aplicación por usuarios con problemas auditivos o sistemas sin dispositivos de audio.

Cumplimiento de estándares

Se ha intentado cumplir los estándares recomendados por el W3C. Sin embargo, como se ha comentado, se utilizan componentes de Struts2. Éstos obligan a una sintaxis concreta que en ocasiones, choca con lo estipulado en los estándares. Por citar un ejemplo concreto, el formulario de Struts2 permite la inyección de valores directamente al Action correspondiente, sin embargo para aprovecharse de esta funcionalidad, es necesario establecer como id del campo del formulario el atributo correspondiente. La obligatoriedad de utilizar un identificador específico requiere la repetición de éstos, lo cual no está permitido en el estándar.

Sin embargo, se han corregido los errores que ha sido posible para cumplir lo recomendado por el estándar.

Se ha comprobado si las hojas de estilo o CSS, validaban el estándar del W3C, sin embargo, el resultado fue negativo. No se han realizado las modificaciones oportunas para conseguir su validación, puesto que la hoja de estilo principal se ha heredado de la plataforma Web de Ecocomputer S.L. y se considera que las modificaciones sobre la misma debería realizarlas su creador.

WCAG 1.0

Se ha utilizado la herramienta TAW, para analizar la accesibilidad del sitio Web. Además dado que muchos aspectos no pueden ser probados de manera automática, se han realizado las pruebas manuales con ayuda del cuestionario que se presenta a continuación.

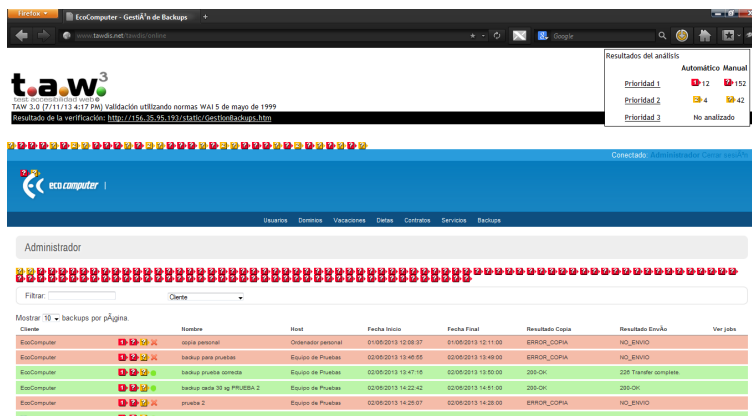


Figura 7.9: Captura de la herramienta TAW, para el estudio de la accesibilidad

Checklist del WCAG 1.0

Puntos de verificación prioridad 1			
En general...	Sí	No	N/A
Proporcione un texto equivalente para todo elemento no textual (Por ejemplo, a través de “alt”, “longdesc” o en el contenido del elemento). <i>Esto incluye:</i> imágenes, representaciones gráficas del texto, mapas de imagen, animaciones (Por ejemplo, <i>GIFs</i> 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		
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		
Identifique claramente los cambios en el idioma del texto del documento y en cualquier texto equivalente (por ejemplo, leyendas).			X
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		
Asegúrese de que los equivalentes de un contenido dinámico son actualizados cuando cambia el contenido dinámico.	X		
<i>continúa en la próxima página</i>			

Puntos de verificación prioridad 1			
Hasta que las aplicaciones de usuario permitan controlarlo, evite provocar destellos en la pantalla.	X		
Utilice el lenguaje apropiado más claro y simple para el contenido de un sitio.	X		
Y si utiliza imágenes y mapas de imagen...	Sí	No	N/A
Proporcione vínculos redundantes en formato texto para cada zona activa de un mapa de imagen del servidor.	X		
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...	Sí	No	N/A
En las tablas de datos, identifique los encabezamientos de fila y columna.	X		
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”)...	Sí	No	N/A
Titule cada marco para facilitar su identificación y navegación.			X
Y si utiliza “applets” y “scripts”...	Sí	No	N/A
Asegure que las páginas sigan siendo utilizables cuando se desconecten o no se soporten los scripts, <i>applets</i> u otros objetos programados. Si esto no es posible, proporcione información equivalente en una página alternativa accesible.		X	
Y si utiliza multimedia...	Sí	No	N/A
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 visual de una presentación multimedia.			X
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

Puntos de verificación prioridad 1			
Y si todo lo demás falla...	Sí	No	N/A
Si, después de los mayores esfuerzos, no puede crear una página accesible, proporcione un vínculo a una página alternativa que use tecnologías <i>W3C</i> , 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. . .	Sí	No	N/A
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	
Cuando exista un marcador apropiado, use marcadores en vez de imágenes para transmitir la información.			X
Cree documentos que estén validados por las gramáticas formales publicadas.		X	
Utilice hojas de estilo para controlar la maquetación y la presentación.	X		
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	
Utilice elementos de encabezado para transmitir la estructura lógica y utilícelos de acuerdo con la especificación.	X		
Marque correctamente las listas y los ítems de las listas.	X		
Marque las citas. No utilice el marcador de citas para efectos de formato tales como sangrías.			X
Asegúrese de que los contenidos dinámicos son accesibles o proporcione una página o presentación alternativa.	X		
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		
Hasta que las aplicaciones de usuario proporcionen la posibilidad de detener las actualizaciones, no cree páginas que se actualicen automáticamente de forma periódica.	X		
<i>continúa en la próxima página</i>			

Puntos de verificación prioridad 2			
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		
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		
Utilice tecnologías <i>W3C</i> cuando estén disponibles y sean apropiadas para la tarea y use las últimas versiones que sean soportadas.	X		
Evite características desaconsejadas por las tecnologías <i>W3C</i> .	X		
Divida los bloques largos de información en grupos más manejables cuando sea natural y apropiado.	X		
Identifique claramente el objetivo de cada vínculo.	X		
Proporcione metadatos para añadir información semántica a las páginas y sitios.		X	
Proporcione información sobre la maquetación general de un sitio (por ejemplo, mapa del sitio o tabla de contenidos).		X	
Utilice los mecanismos de navegación de forma coherente.	X		
Y si utiliza tablas...	Sí	No	N/A
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		
Si se utiliza una tabla para maquetar, no utilice marcadores estructurales para realizar un efecto visual de formato.			X
Y si utiliza marcos (“frames”)...	Sí	No	N/A
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...	Sí	No	N/A

Puntos de verificación prioridad 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		
Asocie explícitamente las etiquetas con sus controles.	X		
Y si utiliza “applets” y “scripts”...	Sí	No	N/A
Para los <i>scripts</i> y <i>applets</i> , asegúrese de que los manejadores de eventos sean independientes del dispositivo de entrada.			X
Hasta que las aplicaciones de usuario permitan congelar el movimiento de los contenidos, evite los movimientos en las páginas.			X
Haga los elementos de programación, tales como <i>scripts</i> y <i>applets</i> , 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
Asegúrese de que cualquier elemento que tiene su propia interfaz pueda manejarse de forma independiente del dispositivo.			X
Para los “scripts”, especifique manejadores de evento lógicos mejor que manejadores de eventos dependientes de dispositivos.			X

Puntos de verificación prioridad 3			
En general...	Sí	No	N/A
Especifique la expansión de cada abreviatura o acrónimo cuando aparezcan por primera vez en el documento.			X
Identifique el idioma principal de un documento.		X	
Cree un orden lógico para navegar con el tabulador a través de vínculos, controles de formulario y objetos.		X	
Proporcione atajos de teclado para los vínculos más importantes (incluidos los de los mapas de imagen de cliente), los controles de formulario y los grupos de controles de formulario.			
<i>continúa en la próxima página</i>			

Puntos de verificación prioridad 3			
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	
Proporcione la información de modo que los usuarios puedan recibir los documentos según sus preferencias (por ejemplo, idioma, tipo de contenido, etc.).		X	
Proporcione barras de navegación para destacar y dar acceso al mecanismo de navegación.	X		
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		
Si proporciona funciones de búsqueda, permita diferentes tipos de búsquedas para diversos niveles de habilidad y preferencias.	X		
Localice la información destacada al principio de los encabezamientos, párrafos, listas, etc.	X		
Proporcione información sobre las colecciones de documentos (por ejemplo, los documentos que comprendan múltiples páginas).		X	
Proporcione un medio para saltar sobre un <i>ASCII art</i> de varias líneas.			X
Complemente el texto con presentaciones gráficas o auditivas cuando ello facilite la comprensión de la página.	X		
Cree un estilo de presentación que sea coherente para todas las páginas.	X		
Y si utiliza imágenes o mapas de imagen...	Sí	No	N/A
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...	Sí	No	N/A
Proporcione resúmenes de las tablas.		X	
Proporcione abreviaturas para las etiquetas de encabezamiento.	X		

continúa en la próxima página

Puntos de verificación prioridad 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 <i>todas</i> las tablas que maquetan texto en paralelo, en columnas de palabras.	X		
Y si utiliza formularios...	Sí	No	N/A
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

Conclusiones

El uso de la herramienta detectó algunos errores fácilmente corregibles, por citar algún ejemplo, no se habían tenido en cuenta los textos alternativos de las imágenes al considerar que el atributo de título podía ofrecer el mismo servicio. Como se comenta, estos errores que implican cambios menores en el código han sido corregidos.

Las soluciones que implican modificaciones en las hojas de estilo proporcionadas por Ecocomputer S.L. no han sido tenidos en cuenta, dado que no se tiene libertad para modificar tales ficheros.

Estado de los puntos de control

Prioridad	Verificar	Bien	Mal	N/A
 P1 HERA WCAG 1.0	4 	1 	--	12 
 P2 HERA WCAG 1.0	9 	10 	1 	9 
 P3 HERA WCAG 1.0	11 	--	2 	6 

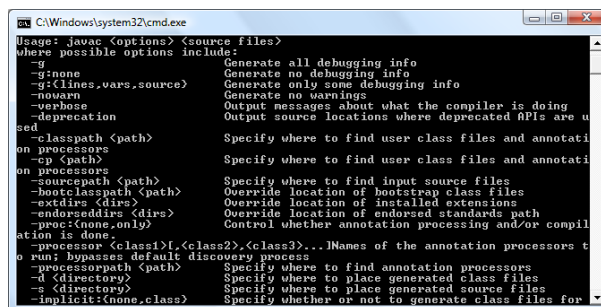
Figura 7.10: Captura de la herramienta HERA, para el estudio de la accesibilidad

Los resultados tras utilizar la herramienta HERA para el estudio de la accesibilidad atendiendo a las WCAG 1.0, pueden verse en la imagen superior.

En esta ventana se deben configurar dos variables de entorno:

- **JAVA_HOME:** Es la variable de entorno que informa al sistema operativo sobre la ruta donde se encuentra instalado Java. En caso de no encontrarse, se debe crear. El valor de la variable deberá ser la ruta donde se ha descargado anteriormente Java, hasta el directorio del JDK.
- **PATH:** Es una variable de entorno del sistema que informa al sistema operativo sobre la ruta de distintos directorios esenciales para el funcionamiento del ordenador. Esta variable debe existir, por lo que únicamente debemos modificar su valor para añadir una nueva ruta. Se debe mantener el valor que presente y concatenar utilizando un punto y coma como separador, la siguiente ruta: «%JAVA_HOME%\bin». Se trata de un equivalente a la ruta utilizada en la variable JAVA_HOME, accediendo al directorio bin.

Se puede comprobar la correcta configuración de Java, abriendo una nueva terminal y ejecutando un comando propio del mismo.



```

C:\Windows\system32\cmd.exe
Usage: javac <options> <source files>
where possible options include:
  -g                Generate all debugging info
  -g:none           Generate no debugging info
  -g:{lines,vars,source}  Generate only some debugging info
  -nowarn           Generate no warnings
  -verbose          Output messages about what the compiler is doing
  -deprecation      Output source locations where deprecated APIs are used
  -classpath <path> Specify where to find user class files and annotation processors
  -cp <path>        Specify where to find user class files and annotation processors
  -sourcepath <path> Specify where to find input source files
  -bootclasspath <path> Override location of bootstrap class files
  -extdirs <dirs>   Override location of installed extensions
  -endorseddirs <dirs> Override location of endorsed standards path
  -proc <none,only> Control whether annotation processing and/or compilation is done.
  -processor <class1>[,<class2>,<class3>...] Names of the annotation processors to run; bypasses default discovery process
  -processorpath <path> Specify where to find annotation processors
  -d <directory>   Specify where to place generated class files
  -s <directory>   Specify where to place generated source files
  -implicit <none,class> Specify whether or not to generate class files for
  
```

Figura 8.2: Comprobación de la correcta configuración de Java

Para la prueba anterior, se ejecutó simplemente la orden «javac »y se comprobó que reconoce el comando, y ofrece la ayuda para indicar como debe ser utilizado. En caso de no estar correctamente configurado, se mostrará un mensaje indicando que no conoce la orden escrita.

8.1.2. Instalación de la aplicación y el servicio Web

El proyecto se ha desplegado en los servidores de Ecocomputer S.L. y configurado para su puesta en producción. Sin embargo, si se deseara montar un sistema de respaldo o cambiar la aplicación de servidor, puede ser de ayuda la siguiente guía de instalación.

Tanto el servicio Web como la aplicación de monitorización se entregan en formato WAR, para ser desplegadas en un servidor de aplicaciones. Sin embargo, debe tenerse en cuenta la integración con módulos ajenos al proyecto y que deben desplegarse en el mismo servidor.

Se recomienda el uso de Apache Tomcat v.6, ya que es el servidor utilizado tanto en las pruebas como en producción, a petición del cliente, y se desconocen incompatibilidades con otros sistemas o distintas versiones.

Por último, es necesario indicar que se debe montar también el sistema de gestión de base de datos, Microsoft SQL Server 2005, que es utilizado por estas aplicaciones. El proceso de instalación, al considerarse demasiado extenso, se ha incluido como apéndice en este mismo documento. Será necesario además modificar el sistema de conexión para que apunte a la nueva dirección de la base de datos, sin embargo, esta tarea es relativa al módulo de gestión de conexiones que queda fuera de este proyecto.

8.2. Manual de ejecución

8.2.1. Arrancar el servidor Web

El proceso de arranque del servidor web es tan sencillo como ejecutar el archivo de lotes «bin\startup.bat»del directorio de Tomcat, siempre y cuando el equipo tenga Java instalado.

Una vez arranque el servidor Tomcat, si se encuentran correctamente ubicados los WAR correspondientes al servicio Web y a la aplicación Web, se encontrarán disponibles para ser utilizados.

8.2.2. Arrancar la aplicación de escritorio

Se entrega la aplicación en un JAR. Este formato es un ejecutable de Java. Para lanzar la aplicación simplemente es necesario ejecutar el siguiente comando en una terminal de Windows:

```
javaw -jar EcoBackup.jar .\clientDataBase.sqlite
```

Para facilitar la tarea y evitar que los usuarios deban recordar el comando, se hace entrega de un fichero .bat que puede ser ejecutado sin necesidad de utilizar comandos de Java.

Es importante destacar, que la base de datos debe encontrarse en el mismo directorio que el JAR y que el .bat, de no ser así, debe modificarse la ruta en este último.

8.3. Manual de usuario

8.3.1. Aplicación de escritorio cliente

Primeros pasos:

La primera vez que se lance la aplicación en un equipo nuevo, se mostrará un panel de registro donde el usuario deberá introducir el nombre y la contraseña con la que desea tener acceso a la aplicación. Además, se solicitará un nombre de equipo con el que hacer referencia al sistema en el que se ejecuta la herramienta.

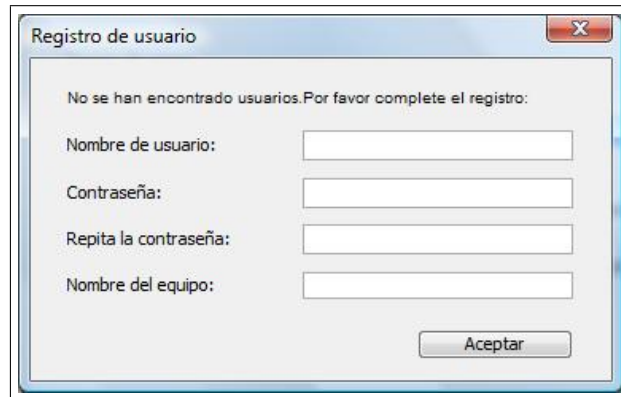


Figura 8.3: Captura de pantalla de registro de usuario

Una vez introducidos los valores, la aplicación no permitirá cambiarlos (se debería borrar la información de la base de datos directamente).

Ya registrado en el sistema, el usuario podrá hacer uso del menú vertical de la parte izquierda y acceder a las distintas funcionalidades de la aplicación.

Debe notarse que además de la ventana mostrada, ha aparecido un icono en el área de notificaciones que nos indica que la aplicación está en uso. Cabe destacar, que cerrar la ventana de la aplicación no acabará con la ejecución del software, ya que éste se seguirá ejecutando hasta que se ordene su cierre en el menú presentado al presionar sobre el icono antes mencionado.

El icono nos presenta un pequeño menú que además de permitir finalizar la ejecución del programa, nos sirve para ver los registros del fichero de Log (ordenados temporalmente) o para restaurar la ventana de interacción si el usuario la ha cerrado con anterioridad o ésta se encuentra oculta o minimizada.

No es recomendable cerrar completamente la aplicación, puesto que es necesario que ésta se encuentre en ejecución en el momento en que deba realizarse un backup.

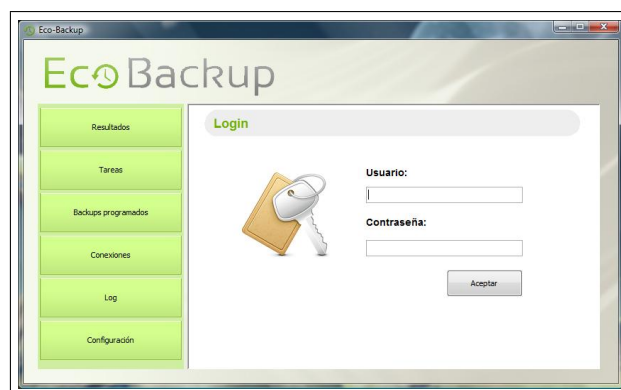


Figura 8.4: Captura del panel de autenticación

Siempre que el programa se inicie con un usuario ya registrado, se presentará la ventana principal y se realizarán las tareas de inicialización, así como, las copias ya programadas

que correspondan. Sin embargo, no se podrá interactuar con la aplicación para crear nuevos programas, realizar una copia instantánea o ver la información de las ya programadas; sin antes introducir el nombre de usuario y la contraseña en el panel de autenticación mostrado.

Explicación de los paneles:

La herramienta cuenta únicamente con una ventana, por ello divide su funcionalidad en distintos paneles accesibles desde el menú vertical izquierdo. A continuación se detallará el objetivo de cada panel y las funcionalidades que permite.

Resultados

El primer botón del menú nos dirige al panel de resultados. En esta sección se pueden ver los resultados de los backups realizados hasta la fecha. Se mostrará una tabla con el nombre y la información de los mismos, notificándose mediante el color de fondo (verde: todo correcto; azul: backup no terminado; rojo: error en alguna tarea) y el icono correspondiente la situación del backup.

Para tener más información sobre un backup concreto, bastará con hacer clic sobre la fila que lo representa. En ese momento se mostrará una tabla en la zona inferior con la información relativa a las tareas de las que se compone el mismo.

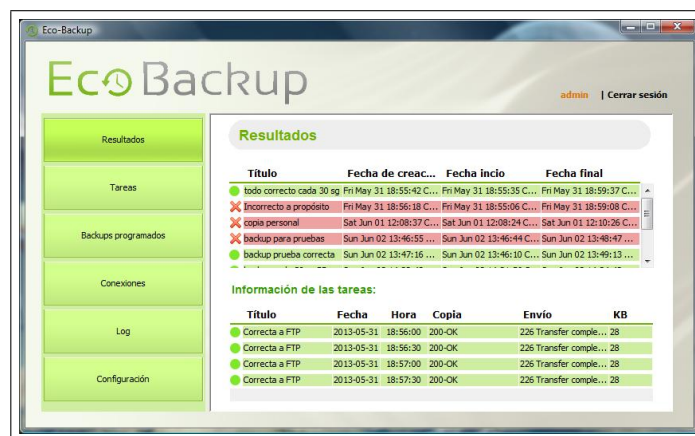


Figura 8.5: Captura del panel de resultados

Tareas

El segundo botón del menú nos permite acceder a la sección de tareas. En este panel podemos consultar las tareas que se encuentran registradas en la aplicación.

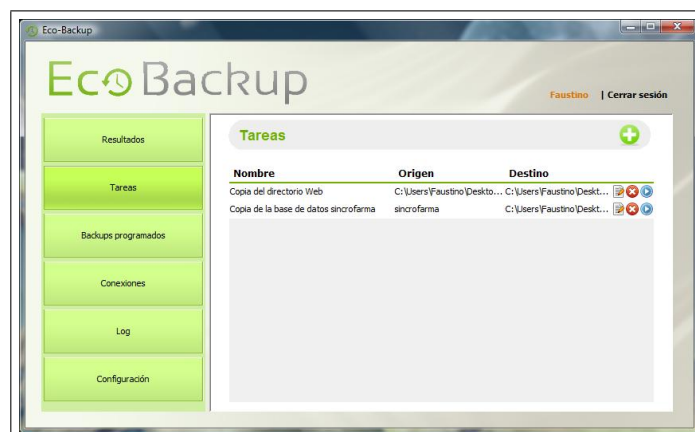


Figura 8.6: Captura del panel de tareas

Al abrir este panel, se mostrará una tabla con las tareas existentes. Acompañando la información de la tarea se encuentran 3 botones. El primero de ellos, permite realizar modificaciones sobre la tarea, el segundo eliminar la misma y el tercero ejecutarla. Es posible realizar una copia inmediata, haciendo uso de esta último botón.

Para crear nuevas tareas se encuentra disponible el botón superior con aspecto de símbolo matemático de suma. Al pulsarlo, aparecerá un cuadro de diálogo que permite seleccionar el tipo de copia que se desea añadir y el origen y el destino de la misma.

Backups programados

En este panel, aparecerá una tabla con los backups que se encuentran activos. Esta tabla contendrá las opciones de modificación y eliminación de manera similar a lo visto en el panel de tareas. Sin embargo, la importancia del mismo radica en la funcionalidad de añadir un nuevo Backup al sistema. Para ello, análogamente con las tareas, haremos uso del botón situado en la parte superior derecha del panel.

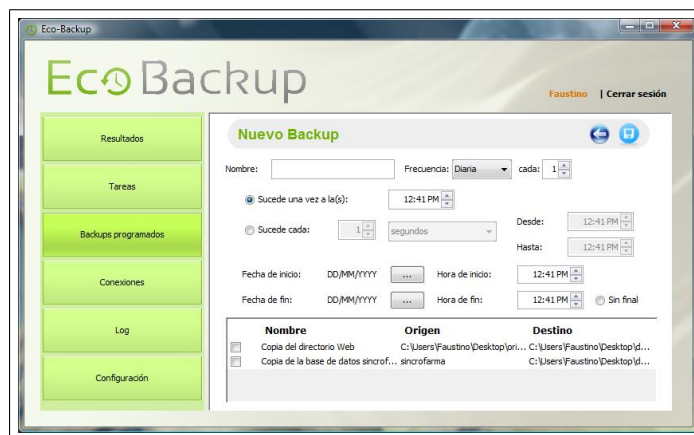


Figura 8.7: Captura del panel de creación de un backup programado

Al seleccionar la creación de un nuevo backup, se hará visible el panel que se muestra en la imagen superior. Este panel presenta un formulario que debe completarse para crear una nueva programación. En la parte inferior del mismo, aparecerá la lista de tareas creadas en el sistema. Para incluir estas a la programación, bastará con marcar el checkbox de las que se deseen.

Una vez completada la programación, se debe guardar utilizando el botón de la esquina superior derecha con apariencia de disco de 3 y medio. Si se desea cancelar la creación, se puede utilizar el botón con forma de flecha situado en la zona superior. En este caso el backup quedaría anulado.

Conexiones

Este panel muestra las conexiones que se han registrado en la aplicación. Dado que se pueden realizar backups de sistemas de bases de datos y publicar resultados en un servidor FTP, se ha decidido crear conexiones para que sea mas sencillo trabajar con los datos de las mismas. Así, se podrán establecer los datos de acceso a una base de datos y luego programar backups de la misma sin necesidad de repetir dicha información y estableciendo únicamente una referencia.

El panel que nos ocupa permite visualizar las conexiones que se encuentran registradas. Se mostrará dividido en dos secciones. Por un lado las conexiones con bases de datos Microsoft SQL Server, y por otro, la información relativa a servidores FTP.

La información de cada conexión se mostrará en una tabla, siguiendo con el modelo utilizado en otros paneles. Nuevamente, la información se verá acompañada de botones para la modificación de una conexión y la eliminación de la misma del sistema. Además, en este caso aparece un nuevo botón cuya funcionalidad es la de probar la conexión.

Será posible realizar una conexión de prueba para comprobar que se encuentra accesible y los datos son correctos.

Por otra parte, de manera análoga a los otros paneles, se utilizarán los botones superiores con el símbolo de suma para añadir nuevas conexiones.



Figura 8.8: Captura del panel de conexiones

En la imagen que acompaña se puede ver el panel mencionado con información relativa a algunas conexiones.

Log

La sección de Log permite visualizar los eventos que han tenido lugar en la aplicación. Se registrarán las actuaciones realizadas por el usuario, así como, los eventos lanzados por el sistema. Para tener más información de un evento concreto, basta con pulsar sobre el mismo y se mostrará en detalle la explicación de la incidencia.

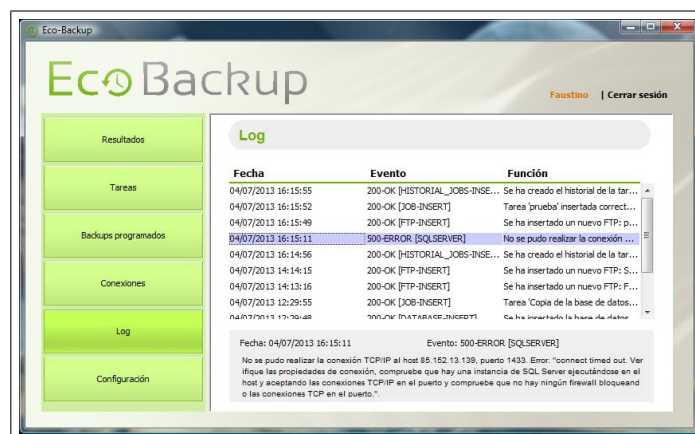


Figura 8.9: Captura del panel de Log

Configuración

El último botón del menú vertical, corresponde al panel de configuración. En el mismo se puede editar la información de sincronización con el servicio Web. En principio no será necesario realizar modificaciones en este aspecto, pero si la empresa necesita modificar la dirección de despliegue del servicio Web, deberá notificar el cambio y que los usuarios actualicen el valor. Además, se permite editar la información de usuario que debe ser la misma que en la aplicación Web.

En caso de realizar una modificación en la cuenta de usuario en la aplicación Web, deberá ser modificada la información en el panel que nos ocupa.

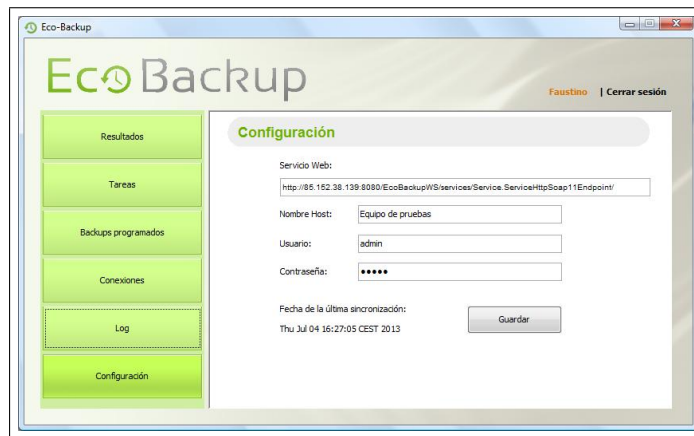


Figura 8.10: Captura del panel de configuración

Creación de un backup complejo

A continuación se explican los pasos a seguir para crear un backup desde cero. Se realizará a modo de ejemplo una copia programada de una base de datos a un servidor FTP. Se considera más complejo en cuanto a número de pasos a realizar, que una copia de un directorio del sistema. Se recuerda que para poder interactuar con la herramienta es necesario autenticarse en la misma en el panel comentado anteriormente.

El primer paso para realizar el backup deseado, será añadir las conexiones al sistema. Para ello accederemos al panel «Conexiones», y mediante los botones de suma añadiremos la información relativa a la conexión con la base de datos que se desea copiar, y al servidor FTP donde se desea almacenar la copia.

El siguiente paso será crear una tarea en el panel correspondiente. Nuevamente hacemos uso del menú vertical y del botón suma para añadir la tarea al sistema. En este caso bastará con referenciar las conexiones creadas en el paso anterior y establecer un nombre para la tarea y otro para la copia. (El segundo debe presentar la extensión .bak en este caso)

Una vez creada la tarea, bastará con acceder al panel de backups y nuevamente con el botón de suma añadir una nueva programación. Se debe rellenar la información relativa al momento en que se desea que sea lanzada la tarea y por último, seleccionar la tarea recién creada en la tabla inferior del panel, dejando marcado su checkbox.

Al guardar la programación de backup, la herramienta mostrará nuevamente el panel con las copias de seguridad activas, entre las cuales debe mostrarse la que acabamos de crear. Otra verificación posible, es consultar el Log del sistema en el panel correspondiente, en el cual debe verse un registro de todos los pasos realizados.

8.3.2. Aplicación Web

La aplicación Web no presenta demasiada funcionalidad y es por ellos trivial de utilizar. Únicamente será necesario acceder a la dirección donde se encuentre desplegada, haciendo uso de un navegador Web. En este tutorial no se presenta la dirección, puesto que por ahora simplemente se conoce la dirección IP del servidor y no tiene un nombre DNS asociado. Será misión de Ecocomputer S.L. dar a conocer la dirección a sus clientes.

Una vez que accedemos a la aplicación Web, se solicita información de autenticación del usuario.

Una vez introducidos los datos correctos, accederemos a la parte privada de la Web. En esta sección se mostrará un menú superior donde debemos acceder a la parte de «Backups». Al entrar se mostrará mediante una tabla la información relativa a los resultados de los backups.

Si el usuario pulsa sobre el botón «Ver Jobs» de alguna fila de la tabla, accederá a la

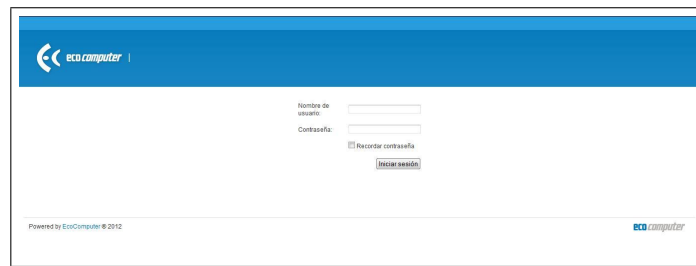


Figura 8.11: Captura del sistema de autenticación Web

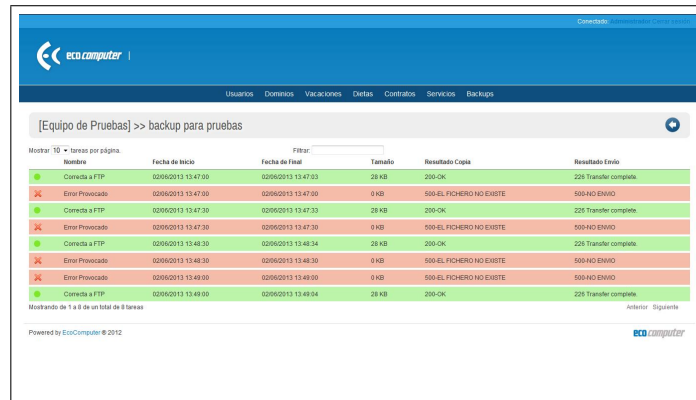


Figura 8.12: Captura de la aplicación Web en la pantalla de tareas

información detallada sobre las tareas de ese backup programado.

Los colores que se han utilizado para denotar el resultado de los backups y las tareas, son los mismos que en la aplicación de escritorio. Azul para los trabajos no finalizados, rojo para los que han terminado con errores y verde para los que han acabado satisfactoriamente.

8.4. Manual del programador

8.4.1. Aplicación de escritorio para Windows

Se ha intentado realizar la aplicación de la manera más clara y mantenible posible, presentando un código comentado y estructurado que permita su comprensión. Sin embargo, se considera importante dejar constancia de algunos aspectos que puedan facilitar la labor de un desarrollador que pretenda realizar ampliaciones o modificaciones. A continuación se detallan algunos aspectos de interés:

- El ejecutor de tareas que presenta la biblioteca utilizada para programar hilos (Quartz Scheduler), no permite seleccionar un objeto con estado. Este software esta pensado para lanzar una clase plana. Por esta razón se ha encontrado una solución compleja pero eficaz. El método «start »del programador admite un flujo de datos que interpreta como nombre de la tarea concreta a ser ejecutada. En ese flujo de datos se encapsulará el objeto backup concreto que deseamos ejecutar. De esta forma en la clase ejecutada podemos obtener el estado del objeto analizando el nombre de la tarea.
- La información requerida por la aplicación Web y que debe enviarse mediante el servicio, no es la misma que la que maneja la aplicación de escritorio. Pese a almacenarse información de resultados de backups y tareas de éstos en ambas bases de datos, las tablas no son iguales.

Por un lado se deben tener en cuenta que los tipos de datos utilizados por SQL Server no son idénticos a los empleados por SQLite. Por otra parte, las tablas del servidor deben contener más campos, que permitan identificar de que cliente y que sistema provienen los datos de resultados. Es importante no confundir los tipos de objetos empleados para el servicio Web y para la aplicación local.

8.4.2. Aplicación Web

En este manual se describen cualquier aspecto que pueda ayudar a otros programadores a ampliar, modificar o entender aspectos de la construcción de nuestra aplicación. No se explica la arquitectura general de una aplicación Struts2 o el funcionamiento de Spring, si no aquellos aspectos particulares del proyecto que pueden provocar confusión.

Tal como se ha comentado en este documento, el proyecto integra con una plataforma Web y aprovecha de ésta la gestión de usuarios. Por esta razón se deben tener en cuenta los siguientes aspectos:

- El módulo integrado se encarga de colocar al usuario en sesión, los Action del proyecto pueden recuperar la información del usuario implementando la interfaz «SessionAware», y utilizando el método `session.get("usuario")`.
- La clase Usuario y asociadas (Perfil, Cliente . . .) se encuentran en la biblioteca Java «ecoclases.jar», incluida dentro del directorio de librerías de Apache Tomcat.
- La biblioteca también contiene lo necesario para gestionar el Log o registro de la aplicación. Para ello basta con declarar la variable

```
Logger log = new Logger(this.getClass().getName(), session);
```

Escribir entradas del registro se hace mediante los métodos `log.info()`, `log.error()` y `log.warning()`, recibiendo como parámetro un String con el mensaje a escribir en la base de datos.

- El módulo también gestiona la comunicación con la base de datos, realizada mediante un pool de conexiones para gestionar mejor los recursos y reducir tiempos de espera innecesarios. La conexión con la base de datos se establece declarando

```
Connection con = ConnectionManager.getConnection();
```

Dejando a un lado esta integración con software ajeno, se presentan algunas recomendaciones para la realización de ampliaciones o modificaciones en el código desarrollado:

- En caso de ser necesaria la creación de nuevos elementos del modelo, se recomienda seguir la misma jerarquía de clases e interfaces empleada (ManagerService, DataService, DAOs ...). Y para aprovechar las ventajas de Spring, se debe declarar la inyección de dependencias necesarias en el fichero «applicationContext.xml».
- Las JavaServer Pages se componen mediante el uso de «tiles», unas bibliotecas Java que se encargan de crear la interfaz como si de un árbol se tratase, compartiendo atributos de padres a hijos para ahorrar código. No es imprescindible su utilización, sin embargo es recomendada al presentarse páginas con una estructura fija y con elementos compartidos. De esta forma, por ejemplo, se evita duplicar el código para la creación del menú principal o de las cabeceras de la página.

Capítulo 9

Conclusiones y ampliaciones

9.1. Conclusiones

Una vez realizado el proyecto, es conveniente analizar los resultados obtenidos y compararlos con los esperados en la fase de análisis.

El proyecto se ajusta a las necesidades del cliente, quien se muestra satisfecho con el producto final. Las herramientas desarrolladas serán puestas en producción cuando Ecocomputer S.L. lo considere oportuno, ya que a su entender facilita su labor a la hora de analizar las copias de seguridad de sus clientes, además de simplificar la realización de las mismas de manera automatizada.

Los retrasos respecto a la planificación inicial se consideran razonables atendiendo a los cambios producidos en los requisitos del cliente. En un proyecto real y que suponga una inversión económica, los requisitos deberán ser firmados en las primeras etapas del proyecto por ambas partes. Por otro lado, se considera que se habrían conseguido subsanar algunos malentendidos, si se hubieran realizado reuniones conjuntas con los gerentes de la empresa. Como se ha comentado con anterioridad, la organización de la empresa colocaba a dos miembros intermediarios entre la comunicación de los gerentes y el desarrollador del proyecto.

A título personal, y teniendo en cuenta que se trata de un proyecto con finalidad académica, se considera una experiencia muy aportadora. La enorme oportunidad de realizar un proyecto para una empresa real, presentó problemas que no se hubieran producido en un proyecto universitario convencional, pero gracias a ello, se cuenta con experiencia en situaciones típicas durante la vida de un proyecto informático.

El proyecto ha servido para conocer algunas tecnologías y profundizar conocimientos aprendidos durante la etapa académica, pero además, para obtener una visión real de la profesión. Se ha obtenido experiencia solventando problemas del trabajo colaborativo con personal de distintos sectores, en la implantación de una metodología de trabajo orientada a la realización de reuniones periódicas de seguimiento . . .

Por otra parte, aunque en algunos aspectos Ecocomputer S.L. tenía claros los requisitos, en otros se contaba con libertad para negociar una solución adecuada. En estas circunstancias, se debía realizar un estudio breve de las alternativas para poder defender una opinión en la reunión posterior. De esta forma se considera que se han mejorado aptitudes relacionadas con la negociación, pues tal como se comenta en este documento, algunas propuestas fueron aceptadas por los clientes al considerarse aportaciones positivas.

9.2. Ampliaciones

Como se ha comentado en las primeras secciones de este documento, el alcance del mismo se vio modificado por diversas circunstancias acontecidas. En un primer estudio se contemplaba la posibilidad de realizar dos módulos adicionales. Estos dos subproyectos podrían realizarse a modo de ampliación. A continuación se detallan éstos y otras propuestas.

9.2.1. Cliente para sistemas GNU/Linux

Se puede desarrollar una herramienta que conecte con el servicio Web y permita realizar copias de seguridad de estos sistemas. No sería necesario realizar un programador de tareas, ya que podría limitarse la funcionalidad a un script capaz de realizar una copia y conectar con el servicio Web, y que éste fuera ejecutado por el gestor de tareas CRON. No tiene sentido realizar copias de sistemas de bases de datos SQL Server, ya que es exclusivo de sistemas con sistema operativo de Microsoft.

En la toma de requisitos inicial del proyecto, se contemplaba el desarrollo de este módulo, sin embargo, Ecocomputer S.L. no cuenta en la actualidad con clientes que trabajen con este tipo de sistemas, por lo que al presentarse retrasos, se eliminó sin mayores inconvenientes.

9.2.2. Aplicación para dispositivos móviles

Como se ha comprobado en las pruebas de usabilidad, la aplicación Web no se adapta a las pantallas de dispositivos móviles. Por esta razón se propone como ampliación, el desarrollo de una aplicación para Android y otra para dispositivos iOS. En los requisitos iniciales, se había contemplado el desarrollo de la versión para móviles con el sistema operativo de Google, sin embargo nuevamente debido a los retrasos, se eliminó del proyecto.

Una alternativa al desarrollo de estas aplicaciones, puede ser el desarrollo de una web adaptable a diferentes tamaños de pantalla, o que detecte cuando es visualizada por un sistema móvil y presentarla con una hoja de estilo más adecuada.

9.2.3. Aplicación Web con mayor funcionalidad

Una vez finalizado el proyecto, se considera que la aplicación Web podría tener mayor funcionalidad. Se podría utilizar la información de la que dispone para generar gráficos y estadísticas que ayuden a una comprensión de los mismos. Dado que las copias se realizan de manera completa, se podría realizar un estudio de incremento entre copias iguales para conocer el progreso o aumento de datos copiados.

Por otra parte, se podría integrar con el servidor FTP que Ecocomputer S.L. disponga para cada cliente, de tal forma que pueda obtener la copia realizada desde la propia aplicación Web de consulta.

9.2.4. Realización de backups de máquinas virtuales

La virtualización de sistemas, es utilizada por Ecocomputer S.L. y por sus clientes, por esta razón se considera una ampliación posible al proyecto, el desarrollo de un nuevo módulo capaz de realizar copias de sistemas virtualizados. Éste debería integrarse con el servicio Web para presentar los datos en la plataforma destinada para ello.

El módulo no requeriría mayor esfuerzo que el estudio de la creación de copias de seguridad con las herramientas de virtualización empleadas. Para el caso de Virtual Box, una aplicación comúnmente utilizada, bastaría con realizar una capa de middleware que permitiera lanzar la funcionalidad de backup de seguridad que ésta presenta.

9.2.5. Permitir copias incrementales o diferenciales

El proyecto permite realizar copias de seguridad completas, sin embargo, puede ser interesante ampliar el mismo para añadir como funcionalidades, las copias incrementales o diferenciales. De esta forma se evita manejar grandes cantidades de datos y se obtienen copias más ligeras.

Se trata de copias menos seguras, al ser necesario mantener la base también almacenada para poder restaurar un sistema. Sin embargo, según el caso concreto puede ser más útil este tipo de backups.

9.2.6. Mejoras de usabilidad del producto

Como se ha visto en las pruebas de usabilidad realizadas, la aplicación de escritorio presenta algunas dificultades de utilización para algunos usuarios. Sería interesante realizar más pruebas y con un grupo mayor de usuarios para concretar que aspectos sería interesante modificar. En este documento se recogen algunas propuestas, pero dado que las pruebas han sido realizadas con una muestra reducida de sujetos y que las interfaces de usuario se consideran impuestas por el cliente, no se han realizado cambios al respecto. Podría realizarse a modo de ampliación, una nueva versión del proyecto con los errores encontrados subsanados.

Capítulo 10

Apéndices

10.1. Primer análisis del proyecto

10.1.1. Determinación del alcance del sistema

El proyecto que nos ocupa debe cubrir unas necesidades concretas del cliente. Se pretende desarrollar una infraestructura completa que permita realizar copias de seguridad, programar las mismas y llevar un seguimiento de las ya realizadas desde cualquier dispositivo con conexión a Internet.

Para conseguir lo requerido se deben realizar dos módulos cliente (uno para entornos Windows y otro para GNU/Linux), un servicio Web que permita la actualización de los datos almacenados en un servidor y por último, una aplicación Web de visualización de resultados.

Para el caso de la aplicación de escritorio de entornos Windows, debe permitirse la realización de backups de bases de datos Microsoft SQL Server 2005, así como la copia de directorios o ficheros concretos. Debe presentarse además un panel de programación que permita automatizar las copias.

El módulo para sistemas GNU/Linux, sin embargo, solo deberá permitir realizar copias de directorios y ficheros concretos, ya que no es posible la instalación de Microsoft SQL Server en estos sistemas. No será necesario implementar un programador de copias, ya que al tratarse de un script con propósito concreto puede delegarse la automatización en CRON.

La aplicación Web deberá ser un monitor de resultados de los backups realizados hasta el momento. Deberá estar actualizado y protegido mediante clave de acceso. Se deberá realizar un sistema de gestión de usuarios que permita otorgar permisos a diferentes cuentas y limite el acceso al contenido no autorizado.

Por último, para conseguir el correcto funcionamiento de la infraestructura y evitar el acceso directo al sistema de base de datos de la aplicación Web, se deberá realizar un servicio Web que permita actualizar la información sobre los backups realizados por los usuarios.

10.1.2. Requisitos funcionales

Aplicación cliente para sistemas Windows

Código	Nombre del requisito	Descripción del requisito
R1.1	Insertar Usuario	La primera ejecución de la aplicación debe requerir un registro por parte del usuario, la información será utilizada posteriormente para autenticar al mismo en el sistema y permitir su interacción.
R1.2	Autenticación	La herramienta debe requerir la autenticación del usuario y no permitir la interacción de usuario anónimos.
R2.1	Copias de directorios y de ficheros	La aplicación debe estar dotada de la funcionalidad suficiente que permita realizar copias de seguridad de ficheros y directorios completos del sistema en el que se encuentra en ejecución.
R2.2	Copias de bases de datos Microsoft SQL Server	La herramienta debe permitir la realización de copias de seguridad de sistemas de bases de datos Microsoft SQL Server que se encuentren desplegados en el equipo cliente en el que se encuentra la aplicación en ejecución.
R3.1	Subida a servidores FTP	Se debe permitir que las copias de seguridad sean enviadas a un servidor FTP detallado por el usuario, así como mantener un registro de los resultados del envío.
R4.1	Programación detallada y sencilla	La herramienta debe proporcionar una interfaz que permita la programación de backups en el tiempo. Debe ser altamente personalizable llegando a un nivel de detalle similar al programador de tareas de Microsoft Windows.
R5.1	Conexión con el servicio Web	Los datos registrados por la aplicación sobre las copias de seguridad realizadas, anomalías en su ejecución y envío de resultados a servidores FTP, deben ser recibidas por el servicio Web y almacenadas en la base de datos que consultará la aplicación Web.
R6.1	Ejecución de backups de manera independiente del usuario	Una vez programados las copias de seguridad, llegado el momento de su ejecución no debe ser necesaria la interacción con el usuario, debe realizarse de manera transparente para el mismo y automatizando el proceso.

Cuadro 10.1: Tabla de requisitos de la aplicación de escritorio cliente para Windows

Aplicación cliente para sistemas GNU/Linux

Código	Nombre del requisito	Descripción del requisito
R1.1	Script de permisos limitados	Se debe realizar una limitación de permisos para evitar que cualquier usuario del sistema pueda lanzar el script.
R2.1	Realización de copias de ficheros	La herramienta debe permitir seleccionar un fichero para realizar una copia del mismo. En caso de no contarse con permisos para acceder al fichero seleccionado se debe permitir la autenticación como otro usuario con permisos.
R2.2	Realización de copias de directorios	La herramienta debe permitir seleccionar un directorio para realizar una copia del mismo. En caso de no contarse con permisos para acceder al directorio seleccionado se debe permitir la autenticación como otro usuario con permisos.
R3.1	Subida a servidores FTP	Se debe permitir que las copias de seguridad sean enviadas a un servidor FTP detallado por el usuario, así como mantener un registro de los resultados del envío.
R4.1	Conexión con el servicio Web	Los datos registrados por la aplicación sobre las copias de seguridad realizadas, anomalías en su ejecución y envío de resultados a servidores FTP, deben ser recibidas por el servicio Web y almacenadas en la base de datos que consultará la aplicación Web.
R5.1	Ejecución de backups de manera independiente del usuario	El script realizado debe ser apto para ser ejecutado por el programador de tareas de Unix.

Cuadro 10.2: Tabla de requisitos de la aplicación de escritorio cliente para sistemas GNU/Linux

Aplicación Web

Código	Nombre del requisito	Descripción del requisito
R1.1	Información privada protegida bajo autenticación	La aplicación Web debe tener un sistema de gestión de usuarios, que permita la autenticación de los mismos para permitir el acceso a su información.
R1.2	Registro de nuevos usuarios	La aplicación Web debe tener un sistema de gestión de usuarios, que permita el registro de nuevos usuarios del sistema.
R1.3	Modificación de permisos de los usuarios	La aplicación Web debe tener un sistema de gestión de usuarios, que permita la modificación de permisos entre éstos. Contando además, con un rol administrador con permisos para realizar todas las operaciones.

continúa en la próxima página

Código	Nombre del requisito	Descripción del requisito
R2.1	Visualización actualizada de los resultados de backups	Debe mostrarse de manera sencilla y actualizada, la información relativa a los resultados de los backups realizados. En caso de error en alguno de ellos debe especificarse la razón del fallo.
R2.2	Visualización actualizada de los resultados de las tareas	Al igual que se muestran los resultados resumidos de los backups, se debe permitir un desglose en tareas para saber cuales fueron realizadas correctamente y cuales no.

Cuadro 10.3: Tabla de requisitos de la aplicación Web

Servicio Web

El servicio Web será un sistema sencillo y su única finalidad será comunicar las aplicaciones de los clientes con la base de datos del servidor de la empresa. Se debe garantizar la seguridad de la comunicación, para evitar que cualquiera pueda introducir datos en el sistema conectándose al servicio.

El servicio debe comprobar los credenciales del usuario que se conecta al mismo, para comprobar si pueden almacenarse los datos que transporta. Por otra parte, se debe notificar del éxito o el fracaso en la actualización.

10.2. Instalación de Microsoft SQL Server 2005 y creación de la base de datos

10.2.1. Servidor de base de datos

En primer lugar será necesario instalar el servidor de base de datos Microsoft SQL Server 2005. Para ello se necesita disponer de un equipo con Windows Server 2003 o superior y cumplir con los siguientes requisitos mínimos:

- Procesador Pentium III o superior a 600 MHz (recomendado superior a 1 GHz)
- 512 MB de memoria RAM (recomendado superior a 1 GB).
- Al menos 2 GB de disco duro.

Antes de comenzar con el proceso de instalación se recomienda tener el sistema operativo totalmente actualizado, mediante el uso de Windows Update. Seguidamente ejecutamos el instalador que comenzará el proceso.

Tras aceptar los habituales términos y condiciones de la licencia y pasar por las primeras pantallas de bienvenida, veremos cómo se ejecuta una comprobación de los requisitos hardware y software del sistema. Si todo está en orden podremos pasar al siguiente paso en el que se nos pedirán nuestros datos:

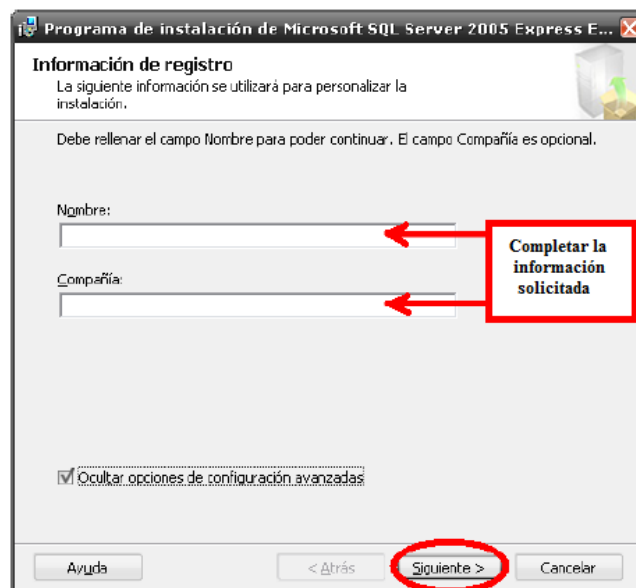


Figura 10.1: Instalación de Microsoft SQL Server, primer paso

Seguidamente deberemos seleccionar qué componentes queremos instalar. Para evitar posibles problemas se aconseja marcar todo.

En el siguiente paso deberemos indicar el nombre de la instancia, que podrá ser el que el usuario desee (en nuestro caso ha sido nombrado PortalEmpleado). Al dar a siguiente deberemos elegir modo de autenticación mixto para que la identificación del usuario no se limite únicamente al entorno Windows y sus usuarios. También deberemos escribir la contraseña para el usuario por defecto a nuestra elección.

El último paso será comprobar que el Firewall del sistema no está bloqueando las conexiones en el puerto de comunicación de la base de datos (por defecto el 1433).

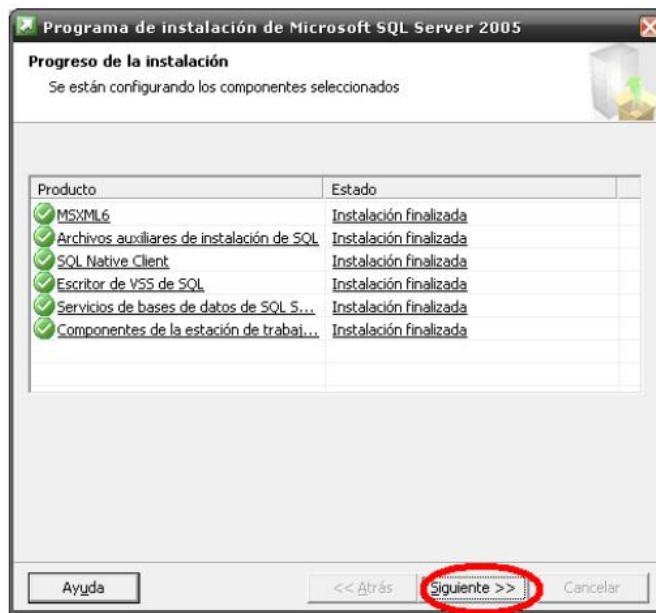


Figura 10.2: Instalación de Microsoft SQL Server, segundo paso

10.2.2. Sistema de gestión de base de datos

Una vez instalado el servidor de base de datos, se necesita una herramienta capaz de gestionar el mismo. El sistema elegido en este proyecto es: Microsoft SQL Server Management Studio Express.

Tras ejecutar el instalador y aceptar las ya comunes primeras pantallas de aceptación de condiciones y bienvenida, se nos pedirá nuevamente los mismos datos de usuario que ya pusimos en la anterior instalación.

En el siguiente paso debemos elegir los componentes a instalar, que por defecto ya está seleccionado el único disponible y necesario. Seguimos aceptando los pasos del instalador y comenzará el proceso:

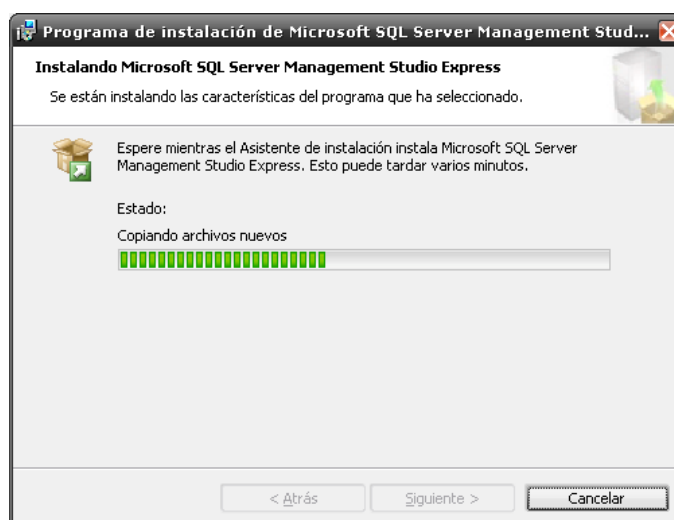


Figura 10.3: Instalación del gestor de base de datos

Una vez instalado tanto el servidor de base de datos como su sistema de gestión se recomienda de nuevo actualizar todo mediante Windows Update, puesto que es probable que exista algún parche de seguridad o rendimiento para ambos paquetes.

10.2.3. Base de datos

Ya con el servidor de base de datos y el SGBD en el equipo procedemos a instalar la base de datos, sus tablas y datos mínimos para poder trabajar en la aplicación. Para ello accedemos a Microsoft SQL Server Management Studio Express y nos pedirá los datos de conexión a la instancia del servidor. Introduciendo los datos de conexión al servidor creado anteriormente, accedemos a la herramienta y podremos empezar a crear la base de datos.

No se detalla el proceso, al considerarlo repetitivo y trivial. La información sobre las tablas que deben crearse y los campos de las mismas se encuentra recogida en este documento. Es posible que sea necesario añadir tablas adicionales que sean utilizadas por el módulo con el que se integra la aplicación desarrollada.

Por último, se deberán introducir datos de configuración en las tablas oportunas. Nuevamente estas tablas no son responsabilidad de este proyecto y deberá consultarse la documentación del proyecto con el que este se integra.

10.3. Contenido Entregado en el CD-ROM

Directorio	Contenido
<i>./ Directorio raíz del CD</i>	Contiene un fichero léeme.txt explicando toda esta estructura. Además, contiene la información relativa al proyectante y director del proyecto.
<i>./Gestión de Backups</i>	Este directorio contiene los ficheros fuente de los tres módulos realizados. Se presentan con una estructura reconocible por el entorno de desarrollo Eclipse, para poder ser importados.
<i>./instalacion</i>	En este directorio se entregan los dos WAR que permiten desplegar el servicio y la aplicación Web en un servidor de aplicaciones. Además, se incluye el JAR de la aplicación de escritorio, la base de datos y un .bat para facilitar su ejecución.
<i>./documentacion</i>	Contiene toda la documentación asociada al proyecto. Se presenta en los siguientes formatos: <i>.TEX</i> y <i>.PDF</i>
<i>./documentacion/javadoc</i>	Este directorio contiene el javadoc generado en formato <i>HTML</i> , para comprender mejor las clases del cliente de escritorio Windows.
<i>./documentacion/uml</i>	Proyecto de Enterprise Architect con los diagramas UML incluidos en esta documentación.

Bibliografía

- [1] Oracle. *Java Platform, Standard Edition 7*. <http://docs.oracle.com/javase/7/docs/api/>, 2013.
- [2] W3C. *Cascading Style Sheets home page* . <http://www.w3.org/Style/CSS/>, 2013.
- [3] W3C. *HTML 4.01 Specification*. <http://www.w3.org/TR/html401/>, 1999.
- [4] Oracle. *Transact-SQL Reference (Database Engine)*. <http://msdn.microsoft.com/en-us/library/bb510741.aspx>, 2007.
- [5] Microsoft MSDN. *HTML 4.01 Specification*. <http://www.w3.org/TR/html401/>, 1999.
- [6] Terracotta. *Quartz Scheduler*. <http://quartz-scheduler.org/>, 2013.
- [7] Terracotta. *Quartz Scheduler, CronTrigger Tutorial*. <http://quartz-scheduler.org/documentation/quartz-1.x/tutorials/crontrigger>, 2013.
- [8] Gamma, Erich y Beck, Kent *JUnit*. <http://junit.org/>, 2013.
- [9] W3Schools *JavaScript Tutorial*. <http://www.w3schools.com/js/>, 2013.
- [10] Hassan, Y. *Guía de Evaluación Heurística de Sitios Web*. www.nosolousabilidad.com/articulos/heuristica.htm, 2008.
- [11] DataTables *DataTables, jQuery Javascript library*. <https://datatables.net/>, 2013.
- [12] Apache Software Foundation. *Apache Tiles* <http://tiles.apache.org/>, 2013.
- [13] DZone. *Struts 2 Tutorials*. <http://www.dzone.com/tutorials/java/struts-2>, 2012.
- [14] *Stack Overflow*. <http://stackoverflow.com/>, 2013.

Documentación de las clases Java

Se ha querido incluir la documentación que presentan las clases a modo de comentarios JavaDoc. Dado que se trata de un elevado número de páginas, se incluye al final del documento y no en la sección destinada para esta finalidad en el capítulo 6. Por otra parte, tampoco se incluyen las clases del proyecto Web ni del servicio, al considerar las primeras demasiado sencillas entendiendo la estructura de Struts2, y en el segundo caso al haber sido prácticamente en su totalidad, generadas con la herramienta de creación de servicios Web de Eclipse.

El orden por el que se encuentra la información de cada paquete es el siguiente:

- **Paquete entities:** Las clases contenidas en este paquete son: Backup, BackupJob, BackupLog, Configuration, Ftp, JobLog, LogLine y SQLServerDatabase.
- **Paquete logic.exec:** Las clases contenidas en este paquete son: BackupExecutor, ExecutableJob, ReloadBackups y ReloadExecutor.
- **Paquete logic.managers:** Las clases contenidas en este paquete son: CompressorManager, CronConverterManager, DirectoryBackupManager, FtpManager, LoadManager, SQLServerBackupManager y WebServiceConnectionManager.
- **Paquete util:** Las clases contenidas en este paquete son: BackupsController, ConnectionsController, JobsController, MD5Converter y SqliteFactory.
- **Paquete util.log:** Este paquete solo contiene la clase encargada del Log de la aplicación.
- **Paquete util.threads:** Las clases contenidas en este paquete son: CheckFTPThread, CheckSQLThread, CopyExecutorThread, DirectoryBackupThread, FTPBackupThread, SchedulerBackupThread, UpdateBackupThread.
- **Paquete visual:** Este paquete contiene las clases: MainWindow y MainUI.
- **Paquete visual.dialogs:** Las clases contenidas en este paquete son: BackupJobsDialog, DatabaseDialog, InformationDialog, RegistrationDialog.
- **Paquete visual.panels:** Las clases contenidas en este paquete son: AddBackupPanel, BackupsPanel, ConfigurationPanel, ConnectionsPanel, EditBackupPanel, JobsPanel, LoginPanel, LogPanel y ResultsPanel. Además, contiene algunas clases privadas encargadas de la maquetación o renderizado de las tablas utilizadas.

El paquete «webservice» no ha sido incluido, al ser las clases generadas por el entorno de desarrollo Eclipse, con su gestor de clientes para servicios Web. Por otra parte, el paquete encargado de la persistencia tampoco ha sido incluido, ya que pese a contar con multitud de métodos, todos presentan una estructura común y una funcionalidad fácilmente reconocible analizando la consulta SQL. Se considera que dichos métodos no necesitan documentación.

Package entities

Class Summary

[Backup](#)

Backup Class, entity that represents a scheduled Backup

[BackupJob](#)

Backup Class, entity that represents a task of a scheduled Backup

[BackupLog](#)

BackupLog Class, entity that represents a Backup result

[Configuration](#)

Configuration Class, configuration system data

[Ftp](#)

FTP Class, entity that represents a FTP connection

[JobLog](#)

JobLog Class, entity that represents a Job result

[LogLine](#)

Log line Class, entity that represents a LOG record

[SQLServerDatabase](#)

SQLServer Database Class, entity that represents a Database connection

entities

Class Backup

```
java.lang.Object
|
+--entities.Backup
```

< [Fields](#) > < [Constructors](#) > < [Methods](#) >

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

Backup Class, entity that represents a scheduled Backup

Version:

1.0 03/04/2013

Author:

Faustino Alonso Posada (EII-Oviedo)

Fields

FIRST_MONTH_DAY

```
public static final java.lang.String FIRST_MONTH_DAY
    {@literal FIRST_MONTH_DAY} It represents first month day
```

FREQ_DAY

```
public static final java.lang.String FREQ_DAY
    {@literal FREQ_DAY} It represents a daily frequency
```

FREQ_MONTH

```
public static final java.lang.String FREQ_MONTH
    {@literal FREQ_MONTH} It represents a monthly frequency
```

FREQ_WEEK

```
public static final java.lang.String FREQ_WEEK
    {@literal FREQ_WEEK} It represents a weekly frequency
```

LAST_MONTH_DAY

```
public static final java.lang.String LAST_MONTH_DAY
    {@literal LAST_MONTH_DAY} It represents last month day
```

Constructors

Backup

```
public Backup(java.lang.String title,  
              java.util.Date date,  
              java.lang.String frequency,  
              int periodicity,  
              java.lang.String executeAt,  
              java.lang.String executeEvery,  
              java.lang.String executeStart,  
              java.lang.String executeEnd,  
              java.util.Date startDate,  
              java.util.Date endDate,  
              java.lang.String weekDay,  
              java.lang.String monthDay)
```

Parameters:

title - String that get name to backup
date - Date Object that represents when backup was scheduled
frequency - String that represents how often backup have to repeat it
periodicity - Integer that represents how many times 'frequency' is repeat it
executeAt - String that represents moment when backup must be executed
executeEvery - String that represents how often backup must be executed
executeStart - String that represents moment when backup must be start
executeEnd - String that represents moment when backup must be end
startDate - Date Object that contains first backup day
endDate - Date Object that contains end backup day
weekDay - String that represents what day of week it must be executed
monthDay - String that represents what day of month it must be executed

Backup

```
public Backup(java.lang.String title,  
             java.util.Date date,  
             java.lang.String frequency,  
             int periodicity,  
             java.lang.String executeAt,  
             java.lang.String executeEvery,  
             java.lang.String executeStart,  
             java.lang.String executeEnd,  
             java.util.Date startDate,  
             java.util.Date endDate,  
             java.lang.String weekDay,  
             java.lang.String monthDay,  
             java.util.List jobsList)
```

Parameters:

title - String that get name to backup
date - Date Object that represents when backup was scheduled
frequency - String that represents how often backup have to repeat it
periodicity - Integer that represents how many times 'frequency' is repeat it
executeAt - String that represents moment when backup must be executed
executeEvery - String that represents how often backup must be executed
executeStart - String that represents moment when backup must be start
executeEnd - String that represents moment when backup must be end
startDate - Date Object that contains first backup day
endDate - Date Object that contains end backup day
weekDay - String that represents what day of week it must be executed
monthDay - String that represents what day of month it must be executed
jobsList - BackupJob objects list about what must be executed

Methods

getDate

```
public java.util.Date getDate()
```

Returns:

Date Object that represents when backup was scheduled

getEndDate

```
public java.util.Date getEndDate()
```

Returns:

Date Object that contains last backup day

getExecuteAt

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

Returns:

String that represents moment when backup must be executed

getExecuteEnd

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

Returns:

String that represents moment when backup must be end

getExecuteEvery

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

Returns:

String that represents how often backup must be executed

getExecuteStart

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

Returns:

String that represents moment when backup must be start

getFrequency

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

Returns:

String that represents how often backup have to repeat it

getId

```
public int getId()
```

Returns:

unique database id

getJobsList

```
public java.util.List getJobsList()
```

Returns:

BackupJob objects list about what must be executed

getMonthDay

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

Returns:

String that represents what month day it must be executed

getPeriodicity

```
public int getPeriodicity()
```

Returns:

Integer that represents how many times 'frequency' is repeat it

getStartDate

```
public java.util.Date getStartDate()
```

Returns:

Date Object that contains first backup day

getTitle

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

Returns:

String that get name to backup

getWeekDay

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

Returns:

String that represents what week day it must be executed

setId

```
public void setId(int id)
```

Parameters:

id - Unique database id

setJobsList

```
public void setJobsList(java.util.List jobsList)
```

Parameters:

jobsList - BackupJob objects list about what must be executed

toString

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

Overrides:

toString in class java.lang.Object

entities

Class BackupJob

```
java.lang.Object
|
+--entities.BackupJob
```

< [Fields](#) > < [Constructors](#) > < [Methods](#) >

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

Backup Class, entity that represents a task of a scheduled Backup

Version:

1.0 03/04/2013

Author:

Faustino Alonso Posada (EII-Oviedo)

Fields

DIR_FTP

```
public static final int DIR_FTP
    {@literal DIR_FTP} It represents a copy of local file or directory and send it to FTP Server
```

DIR_LOCAL

```
public static final int DIR_LOCAL
    {@literal DIR_LOCAL} It represents a local copy of local file or directory
```

SQL_FTP

```
public static final int SQL_FTP
    {@literal SQL_FTP} It represents a copy of Microsoft SQL Server Database and send it to FTP
    Server
```

SQL_LOCAL

```
public static final int SQL_LOCAL
    {@literal SQL_LOCAL} It represents a local copy of Microsoft SQL Server Database
```

Constructors

BackupJob

```
public BackupJob(java.lang.String name,
                 int databaseId,
                 java.lang.String databaseConnectionName,
                 java.lang.String sqlHostOrigin,
                 java.lang.String sqlUserOrigin,
                 java.lang.String sqlPassOrigin,
                 int sqlPortOrigin,
                 java.lang.String nameDateBaseOrigin,
                 int ftpId,
                 java.lang.String ftpTitleDestination,
                 java.lang.String ftpHostDestination,
                 java.lang.String ftpUserDestination,
                 java.lang.String ftpPassDestination,
                 java.lang.String ftpFileDestination)
```

Parameters:

sqlHostOrigin - SQL Server Database address
sqlUserOrigin - SQL Server Database user
sqlPassOrigin - SQL Server Database password of that user
sqlPortOrigin - SQL Server Database port
nameDateBaseOrigin - SQL Server Database name
ftpHostDestination - Destination FTP Server address
ftpUserDestination - Destination FTP Server user
ftpPassDestination - Destination FTP Server password of that user
ftpFileDestination - Destination FTP Server file name

BackupJob

```
public BackupJob(java.lang.String name,
                 int databaseId,
                 java.lang.String databaseConnectionName,
                 java.lang.String sqlHostOrigin,
                 java.lang.String sqlUserOrigin,
                 java.lang.String sqlPassOrigin,
                 int sqlPortOrigin,
                 java.lang.String nameDateBaseOrigin,
                 java.lang.String directoryDestination)
```

Parameters:

sqlHostOrigin - SQL Server Database address
sqlUserOrigin - SQL Server Database user
sqlPassOrigin - SQL Server Database password of that user
sqlPortOrigin - SQL Server Database port
nameDateBaseOrigin - SQL Server Database name
directoryDestination - Destination directory, when do you want to copy result file?

BackupJob

```
public BackupJob(java.lang.String name,  
                java.lang.String directoryOrigin,  
                int ftpId,  
                java.lang.String ftpTitleDestination,  
                java.lang.String ftpHostDestination,  
                java.lang.String ftpUserDestination,  
                java.lang.String ftpPassDestination,  
                java.lang.String ftpFileDestination)
```

Parameters:

directoryOrigin - Origin directory, what do you want to save?
ftpHostDestination - Destination FTP Server address
ftpUserDestination - Destination FTP Server user
ftpPassDestination - Destination FTP Sever password of that user
ftpFileDestination - Destination FTP Server file name

BackupJob

```
public BackupJob(java.lang.String name,  
                java.lang.String directoryOrigin,  
                java.lang.String directoryDestination)
```

Parameters:

directoryOrigin - Origin directory, what do you want to save?
directoryDestination - Destination directory, when do you want to copy result file?

Methods

equals

```
public boolean equals(java.lang.Object obj)
```

Overrides:

equals in class java.lang.Object

getCronFormat

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

Returns:

Scheduled Backup time in CRON format

getDatabaseConnectionName

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

getDatabaseId

```
public int getDatabaseId()
```

getDestinationType

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

getDirectoryDestination

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

Returns:

Destination directory, when do you want to save result file?

getDirectoryOrigin

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

Returns:

Origin directory, what do you want to save?

getFtpFileDestination

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

Returns:

FTP Server destination file name

getFtpHostDestination

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

Returns:

FTP Server address

getFtpId

```
public int getFtpId()
```

getFtpPassDestination

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

Returns:

FTP Server password

getFtpTitleDestination

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

getFtpUserDestination

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

Returns:

FTP Server user

getJobId

```
public int getJobId()
```

Returns:

Unique database job identification

getName

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

getNameDataBaseOrigin

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

Returns:

SQL Server Database name

getOriginType

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

getSqlHostOrigin

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

Returns:

SQL Server Database address

getSqlPassOrigin

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

Returns:

SQL Server Database password of user

getSqlPortOrigin

```
public int getSqlPortOrigin()
```

Returns:

SQL Server Database port

getSqlUserOrigin

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

Returns:

SQL Server Database user

getType

```
public int getType()
```

Returns:

backup job type{'DIR_LOCAL','DIR_FTP','SQL_LOCAL','SQL_FTP'}

hashCode

```
public int hashCode()
```

Overrides:

hashCode in class java.lang.Object

setCronFormat

```
public void setCronFormat(java.lang.String cronFormat)
```

Parameters:

cronFormat - Scheduled Backup time in CRON format

setDatabaseConnectionName

```
public void setDatabaseConnectionName(java.lang.String databaseConnectionName)
```

setDatabaseId

```
public void setDatabaseId(int databaseId)
```

setFtpId

```
public void setFtpId(int ftpId)
```

setFtpTitleDestination

```
public void setFtpTitleDestination(java.lang.String ftpTitleDestination)
```

setJobId

```
public void setJobId(int jobId)
```

Parameters:

jobId - Unique database job identification

setName

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

toString

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

Overrides:

toString in class java.lang.Object

entities

Class BackupLog

```
java.lang.Object  
|  
+--entities.BackupLog
```

< [Constructors](#) > < [Methods](#) >

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

BackupLog Class, entity that represents a Backup result

Version:

1.0 03/04/2013

Author:

Faustino Alonso Posada (EII-Oviedo)

Constructors

BackupLog

```
public BackupLog(int idBackup,  
                java.util.Date startDate,  
                java.util.Date endDate,  
                int backupResult,  
                java.lang.String copyResult,  
                java.lang.String sendingResult)
```

Parameters:

idBackup - Backup identification
startDate - start backup date
endDate - end backup date
backupResult - backup result
copyResult - copy result
sendingResult - sending result

Methods

getBackupResult

```
public int getBackupResult()
```

Returns:

backup result value

getCopyResult

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

Returns:

copy result value

getEndDate

```
public java.util.Date getEndDate()
```

Returns:

end date

getIdBackup

```
public int getIdBackup()
```

Returns:

backup id

getSendingResult

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

Returns:

sending result value

getStartDate

```
public java.util.Date getStartDate()
```

Returns:

start date

setBackupResult

```
public void setBackupResult(int backupResult)
```

Parameters:

backupResult -

setCopyResult

```
public void setCopyResult(java.lang.String copyResult)
```

Parameters:

copyResult -

setEndDate

```
public void setEndDate(java.util.Date endDate)
```

Parameters:

endDate -

setIdBackup

```
public void setIdBackup(int idBackup)
```

Parameters:

idBackup -

setSendingResult

```
public void setSendingResult(java.lang.String sendingResult)
```

Parameters:

sendingResult -

setStartDate

```
public void setStartDate(java.util.Date startDate)
```

Parameters:

startDate -

entities

Class Configuration

```
java.lang.Object
|
+--entities.Configuration
```

< [Constructors](#) > < [Methods](#) >

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

Configuration Class, configuration system data

Version:

1.0 03/04/2013

Author:

Faustino Alonso Posada (EII-Oviedo)

Constructors

Configuration

```
public Configuration()
```

Methods

getEndPointAddress

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

Returns:

EndPoint Address

getHost

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

Returns:

host

getPassword

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

Returns:

password

getSynchronization

```
public java.util.Date getSynchronization()
```

Returns:

last synchronization date

getUser

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

Returns:

user

setEndPointAddress

```
public void setEndPointAddress(java.lang.String endPointAddress)
```

Parameters:

endPointAddress -

setHost

```
public void setHost(java.lang.String host)
```

Parameters:

host -

setPassword

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

Parameters:

password -

setSynchronization

```
public void setSynchronization(java.util.Date synchronization)
```

Parameters:

synchronization - date

setUser

```
public void setUser(java.lang.String user)
```

Parameters:

user -

entities

Class Ftp

```
java.lang.Object
|
+--entities.Ftp
```

< [Constructors](#) > < [Methods](#) >

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

FTP Class, entity that represents a FTP connection

Version:

1.0 03/04/2013

Author:

Faustino Alonso Posada (EII-Oviedo)

Constructors

Ftp

```
public Ftp(int id,  
           java.lang.String title,  
           java.lang.String host,  
           java.lang.String user,  
           java.lang.String pass)
```

Parameters:

id - FTP identification
title - connection name
host - connection address
user - FTP user name
pass - user password

Ftp

```
public Ftp(java.lang.String title,  
           java.lang.String host,  
           java.lang.String user,  
           java.lang.String pass)
```

Parameters:

title - connection name
host - connection address
user - FTP user name
pass - user password

Methods

getHost

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

Returns:

host connection address

getId

```
public int getId()
```

Returns:

FTP identification

getPass

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

Returns:

password FTP user password

getTitle

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

Returns:

connection name

getUser

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

Returns:

user FTP user name

setHost

```
public void setHost(java.lang.String host)
```

Parameters:

host - connection address

setPass

```
public void setPass(java.lang.String pass)
```

Parameters:

password - FTP user password

setTitle

```
public void setTitle(java.lang.String title)
```

Parameters:

title - connection name

setUser

```
public void setUser(java.lang.String user)
```

Parameters:

user - FTP user name

toString

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

Overrides:

toString in class java.lang.Object

entities

Class JobLog

```
java.lang.Object  
|  
+--entities.JobLog
```

< [Constructors](#) > < [Methods](#) >

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

JobLog Class, entity that represents a Job result

Version:

1.0 03/04/2013

Author:

Faustino Alonso Posada (EII-Oviedo)

Constructors

JobLog

```
public JobLog(BackupJob job,  
             int backupId,  
             java.util.Date startDate,  
             java.util.Date endDate,  
             int result,  
             java.lang.String copyResult,  
             java.lang.String sendingResult,  
             int size)
```

Parameters:

job - Backupjob
backupId - Backup identification
startDate - start job date
endDate - end job date
result - job result
copyResult - copy result
sendingResult - sending result
size - copy file size

Methods

getBackupId

```
public int getBackupId()
```

Returns:

backup id

getCopyResult

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

Returns:

copy result

getEndDate

```
public java.util.Date getEndDate()
```

Returns:

end date

getJob

```
public BackupJob getJob()
```

Returns:

job

getResult

```
public int getResult()
```

Returns:

job result

getSendingResult

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

Returns:

sending result

getSize

```
public int getSize()
```

Returns:

copy size

getStartDate

```
public java.util.Date getStartDate()
```

Returns:

start date

setBackupId

```
public void setBackupId(int backupId)
```

Parameters:

backupId - Backup identification

setCopyResult

```
public void setCopyResult(java.lang.String copyResult)
```

Parameters:

copy - result

setEndDate

```
public void setEndDate(java.util.Date endDate)
```

Parameters:

endDate -

setJob

```
public void setJob(BackupJob job)
```

Parameters:

job - Backupjob

setResult

```
public void setResult(int result)
```

Parameters:

job - result

setSendingResult

```
public void setSendingResult(java.lang.String sendingResult)
```

Parameters:

sending - Result

setSize

```
public void setSize(int size)
```

Parameters:

copy - size

setStartDate

```
public void setStartDate(java.util.Date startDate)
```

Parameters:

startDate -

entities

Class LogLine

```
java.lang.Object  
|  
+--entities.LogLine
```

< [Constructors](#) > < [Methods](#) >

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

Log line Class, entity that represents a LOG record

Version:

1.0 03/04/2013

Author:

Faustino Alonso Posada (EII-OVIEDO)

Constructors

LogLine

```
public LogLine(java.lang.String event,  
               java.lang.String incident,  
               java.util.Date date)
```

Parameters:

event - , what happens?
incident - , what it produced?
date - , when it happens?

Methods

getDate

```
public java.util.Date getDate()
```

Returns:

when it happens?

getEvent

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

Returns:

what happens?

getIncident

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

Returns:

what it produced?

setDate

```
public void setDate(java.util.Date date)
```

Parameters:

date - when it happens?

setEvent

```
public void setEvent(java.lang.String event)
```

Parameters:

event - what happens?

setIncident

```
public void setIncident(java.lang.String incident)
```

Parameters:

incident - what it produced?

toString

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

Overrides:

toString in class java.lang.Object

entities

Class SQLServerDatabase

```
java.lang.Object  
|  
+--entities.SQLServerDatabase
```

< [Constructors](#) > < [Methods](#) >

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

SQLServer Database Class, entity that represents a Database connection

Version:

1.0 03/04/2013

Author:

Faustino Alonso Posada (EII-Oviedo)

Constructors

SQLServerDatabase

```
public SQLServerDatabase(int id,  
                        java.lang.String title,  
                        java.lang.String host,  
                        int port,  
                        java.lang.String name,  
                        java.lang.String user,  
                        java.lang.String pass)
```

Parameters:

id - database connection identification
title - connection name
host - database address
port - database port
name - database name
user - database user name
pass - database user password

Methods

getHost

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

Returns:

host database connection address

getId

```
public int getId()
```

Returns:

database connection identification

getName

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

Returns:

database connection name

getPass

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

Returns:

password database user password

getPort

```
public int getPort()
```

Returns:

port database connection port

getTitle

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

Returns:

connection name

getUser

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

Returns:

user database user name

setHost

```
public void setHost(java.lang.String host)
```

Parameters:

host - database connection address

setId

```
public void setId(int id)
```

Parameters:

id - database connection identification

setName

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

Parameters:

database - connection name

setPass

```
public void setPass(java.lang.String pass)
```

Parameters:

password - database user password

setPort

```
public void setPort(int port)
```

Parameters:

port - database connection port

setTitle

```
public void setTitle(java.lang.String title)
```

Parameters:

connection - database user name

setUser

```
public void setUser(java.lang.String user)
```

Parameters:

user - database user name

Package logic.exec

Class Summary

[BackupExecutor](#)

Jobs Executor Class, it is backup scheduler, who execute backup jobs.

[ExecutableJob](#)

Executable job Class represents a backup task.

[ReloadBackups](#)

Reload Backups Class, it is an automatic thread to load new backups and finish old of them.

[ReloadExecutor](#)

Reload Executor Class, it is the ReloadBackups executor.

logic.exec

Class BackupExecutor

```
java.lang.Object
|
+--logic.exec.BackupExecutor
```

< [Fields](#) > < [Constructors](#) > < [Methods](#) >

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

Jobs Executor Class, it is backup scheduler, who execute backup jobs.

Version:

1.0 02/04/2013

Author:

Faustino Alonso Posada (EII-Oviedo)

Fields

scheduler

```
public static org.quartz.Scheduler scheduler
```

Constructors

BackupExecutor

```
public BackupExecutor(entities.Backup backup)
```

Parameters:

backup -

Methods

execute

```
public void execute()
```

Start all BackupJobs that still stopped and are in time.

stop

```
public void stop()
```

Stop all BackupJobs

stopJobsFromBackup

```
public static void stopJobsFromBackup(entities.Backup backup,  
                                     java.lang.String reason)
```

Stop all BackupJobs from a Backup

Parameters:

backup -

reason - why do you stop all of them?

logic.exec

Class ExecutableJob

```
java.lang.Object  
|  
+--logic.exec.ExecutableJob
```

< [Constructors](#) > < [Methods](#) >

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

Executable job Class represents a backup task. What must be do it.

Version:

1.0 03/04/2013

Author:

Faustino Alonso Posada (EII-Oviedo)

Constructors

ExecutableJob

```
public ExecutableJob()
```

ExecutableJob

```
public ExecutableJob(int type,  
                    java.lang.String directoryOrigin,  
                    java.lang.String sqlHostOrigin,  
                    java.lang.String sqlUserOrigin,  
                    java.lang.String sqlPassOrigin,  
                    int sqlPortOrigin,  
                    java.lang.String nameDataBaseOrigin,  
                    java.lang.String directoryDestination,  
                    java.lang.String ftpHostDestination,  
                    java.lang.String ftpUserDestination,  
                    java.lang.String ftpPassDestination,  
                    java.lang.String ftpFileDestination)
```

Methods

execute

```
public void execute(JobExecutionContext arg0)
```

executeJob

```
public void executeJob(int jobId)
```

getBackupId

```
public int getBackupId()
```

getDirectoryDestination

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

getDirectoryOrigin

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

getFtpFileDestination

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

getFtpHostDestination

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

getFtpPassDestination

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

getFtpUserDestination

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

getNameDataBaseOrigin

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

getSqlHostOrigin

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

getSqlPassOrigin

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

getSqlPortOrigin

```
public int getSqlPortOrigin()
```

getSqlUserOrigin

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

getType

```
public int getType()
```

setBackupId

```
public void setBackupId(int backupId)
```

toString

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

Overrides:

toString in class java.lang.Object

logic.exec

Class ReloadBackups

```
java.lang.Object
|
+--logic.exec.ReloadBackups
```

< [Constructors](#) > < [Methods](#) >

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

Reload Backups Class, it is an automatic thread to load new backups and finish old of them.

Version:

1.0 03/04/2013

Author:

Faustino Alonso Posada (EII-Oviedo)

Constructors

ReloadBackups

```
public ReloadBackups()
```

Methods

execute

```
public void execute(JobExecutionContext arg0)
```

logic.exec

Class ReloadExecutor

```
java.lang.Object
|
+--logic.exec.ReloadExecutor
```

< [Constructors](#) > < [Methods](#) >

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

Reload Executor Class, it is the ReloadBackups executor.

Version:

1.0 03/04/2013

Author:

Faustino Alonso Posada (EII-Oviedo)

Constructors

ReloadExecutor

```
public ReloadExecutor()
```

Methods

execute

```
public void execute()
```

Execute ReloadBackups thread every minute

stop

```
public void stop()
```

Stop ReloadBackup thread

Package logic.managers

Class Summary

[CompressorManager](#)

CompressorManager Class, It is used to convert directory in a ZIP file.

[CronConverterManager](#)

CRON Converter Manager Class, it is a CRON format converter.

[DirectoryBackupManager](#)

Directory Backup Manager Class, it makes files or directories copies.

[FtpManager](#)

FTP Manager Class, it upload files to FTP server.

[LoadManager](#)

Load Manager Class, it load Backup information from database.

[SQLServerBackupManager](#)

SQL Server Backup Manager Class, It makes a database backup.

[WebServiceConnectionManager](#)

Web Service Connection Manager Class.

logic.managers

Class CompressorManager

```
java.lang.Object
|
+--logic.managers.CompressorManager
```

< [Constructors](#) > < [Methods](#) >

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

CompressorManager Class, It is used to convert directory in a ZIP file.

Version:

1.0 03/04/2013

Author:

Faustino Alonso Posada (EII-Oviedo)

Constructors

CompressorManager

```
public CompressorManager()
```

Methods

createZipFile

```
public static void createZipFile(java.lang.String in,  
                                java.lang.String out)  
    throws java.io.IOException
```

Parameters:

in - Path to origin directory
out - Path to destination ZIP file

Throws:

java.io.IOException - If something in Input-output stream was wrong

logic.managers

Class CronConverterManager

```
java.lang.Object  
|  
+--logic.managers.CronConverterManager
```

< [Constructors](#) > < [Methods](#) >

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

CRON Converter Manager Class, it is a CRON format converter. It transform to CRON format time information from a scheduled backup

Version:

1.0 03/04/2013

Author:

Faustino Alonso Posada (EII-Oviedo)

Constructors

CronConverterManager

```
public CronConverterManager()
```

Methods

scheduleBackup

```
public java.lang.String scheduleBackup(entities.Backup backup)
```

Parameters:

backup - Scheduled Backup

Returns:

CRON time format information from a scheduled backup

logic.managers

Class DirectoryBackupManager

```
java.lang.Object
|
+--logic.managers.DirectoryBackupManager
```

< [Constructors](#) > < [Methods](#) >

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

Directory Backup Manager Class, it makes files or directories copies.

Version:

1.0 03/04/2013

Author:

Faustino Alonso Posada (EII-Oviedo)

Constructors

DirectoryBackupManager

```
public DirectoryBackupManager()
```

Methods

makeBackup

```
public static boolean makeBackup(java.io.File origin,  
                                java.io.File destination)
```

Parameters:

origin - file or directory, what do you want to save?
destination - file or directory, where do you want to save it?

logic.managers

Class FtpManager

```
java.lang.Object  
|  
+--logic.managers.FtpManager
```

< [Fields](#) > < [Constructors](#) > < [Methods](#) >

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

FTP Manager Class, it upload files to FTP server.

Version:

1.0 03/04/2013

Author:

Faustino Alonso Posada (EII-Oviedo)

Fields

TEMPORAL

```
public static final java.lang.String TEMPORAL  
{@literal TEMPORAL}
```

Constructors

FtpManager

```
public FtpManager()
```

FtpManager

```
public FtpManager(int jobId,
                  int backupId)
```

Parameters:

jobId - backup job identification, who need to upload files?

Methods

uploadDatabaseByFTP

```
public void uploadDatabaseByFTP(java.lang.String hostName,
                                java.lang.String userName,
                                java.lang.String password,
                                java.lang.String databasePath,
                                java.lang.String fileName)
```

uploadFileByFTP

```
public void uploadFileByFTP(java.lang.String hostName,
                             java.lang.String userName,
                             java.lang.String password,
                             java.lang.String filePath,
                             java.lang.String fileName)
```

Parameters:

hostName - FTP Server Address
userName - FTP Server user
password - FTP Server user password
filePath - Origin file path
fileName - FTP Server file name

logic.managers

Class LoadManager

```
java.lang.Object
|
+--logic.managers.LoadManager
```

< [Constructors](#) > < [Methods](#) >

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

Load Manager Class, it load Backup information from database.

Version:

1.0 03/04/2013

Author:

Faustino Alonso Posada (EII-Oviedo)

Constructors

LoadManager

```
public LoadManager()
```

Methods

getBackupsList

```
public java.util.List getBackupsList()
```

Returns:

Backups List with all database information

scheduleBackup

```
public void scheduleBackup(entities.Backup backup)
```

Scheduler a backup

Parameters:

backup - that have to be scheduler

scheduler

```
public void scheduler()
```

Scheduler all load backups

logic.managers

Class SQLServerBackupManager

```
java.lang.Object
|
+--logic.managers.SQLServerBackupManager
```

< [Fields](#) > < [Constructors](#) > < [Methods](#) >

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

SQL Server Backup Manager Class, It makes a database backup.

Version:

1.0 03/04/2013

Author:

Faustino Alonso Posada (EII-Oviedo)

Fields

TEMPORAL

```
public static java.lang.String TEMPORAL
    {@literal TEMPORAL}
```

Constructors

SQLServerBackupManager

```
public SQLServerBackupManager()
```

SQLServerBackupManager

```
public SQLServerBackupManager(int jobId,
                               int backupId)
```

Parameters:

jobId - backup job identification, who need to upload files?

Methods

dbConnect

```
public void dbConnect(java.lang.String db_connect_string,
                      int port,
                      java.lang.String userid,
                      java.lang.String password,
                      java.lang.String temporalFile,
                      java.lang.String dbName)
```

Parameters:

db_connect_string - SQL Server database connection
userid - SQL Server database user
password - SQL Server database password
temporalFile - Temporal File path
dbName - SQL Server database name

dbConnectToFTP

```
public void dbConnectToFTP(java.lang.String db_connect_string,
                           int port,
                           java.lang.String userid,
                           java.lang.String password,
                           java.lang.String temporalFile,
                           java.lang.String dbName)
```

logic.managers

Class WebServiceConnectionManager

```
java.lang.Object
|
+--logic.managers.WebServiceConnectionManager
```

< [Constructors](#) > < [Methods](#) >

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

Web Service Connection Manager Class.

Version:

1.0 03/06/2013

Author:

Faustino Alonso Posada (EII-Oviedo)

Constructors

WebServiceConnectionManager

```
public WebServiceConnectionManager()
```

Methods

uploadWebService

```
public boolean uploadWebService()
```

Updating server database with new data

Returns:

result

Package util

Class Summary

[BackupsController](#)

Backup controller

[ConnectionsController](#)

Connections Controller

[JobsController](#)

Jobs Controller

[MD5Converter](#)

Converter to MD5 (Encrypt password)

[SqliteFactory](#)

SQLite Factory to obtain SQLite Database path

util

Class BackupsController

```
java.lang.Object
|
+--util.BackupsController
```

< [Constructors](#) > < [Methods](#) >

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

Backup controller

Version:

1.0 03/04/2013

Author:

Faustino Alonso Posada (EII-Oviedo)

Constructors

BackupsController

```
public BackupsController()
```

Methods

delete

```
public static void delete(int backupId)
```

Delete backup

Parameters:

backupId -

util

Class ConnectionsController

```
java.lang.Object  
|  
+--util.ConnectionsController
```

< [Constructors](#) > < [Methods](#) >

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

Connections Controller

Version:

2.0 03/06/2013

Author:

Faustino Alonso Posada (EII-Oviedo)

Constructors

ConnectionsController

```
public ConnectionsController()
```

Methods

checkFTP

```
public static void checkFTP(int ftpId)
```

Check FTP connection

Parameters:

ftpId - identification

checkSQL

```
public static void checkSQL(int id)
```

Check SQL Server database connection

Parameters:

id - identification

deleteFTP

```
public static void deleteFTP(int ftp)
```

Delete a FTP connection

Parameters:

ftp - identification

deleteSQL

```
public static void deleteSQL(int id)
```

Delete SQL Server database connection

Parameters:

id - identification

editFTP

```
public static void editFTP(int ftp,
                           java.lang.String title,
                           java.lang.String host,
                           java.lang.String user)
```

Parameters:

ftp - FTP identification
title - connection name
host - FTP address
user - FTP user name

editSQL

```
public static void editSQL(int id,
                            java.lang.String host,
                            java.lang.String title,
                            int port,
                            java.lang.String name,
                            java.lang.String user)
```

Parameters:

id - database identification
host - database address
title - connection name
port - database port
name - database name
user - database user name

util

Class JobsController

```
java.lang.Object
|
+--util.JobsController
```

< [Constructors](#) > < [Methods](#) >

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

Jobs Controller

Version:

1.0 09/05/2013

Author:

Constructors

JobsController

```
public JobsController()
```

Methods

deleteJob

```
public static void deleteJob(int jobId)
```

Delete job

Parameters:

jobId - job identification

editJob

```
public static void editJob(int jobId,  
                           java.lang.String jobName,  
                           java.lang.String originType,  
                           java.lang.String destinationType,  
                           int fpdId,  
                           int databaseId,  
                           java.lang.String originDir,  
                           java.lang.String destinationDir)
```

Parameters:

jobId - job identification

jobName - job name

originType - origin type

destinationType - destination type

fpdId - FTP identification

databaseId - database identification

originDir - origin directory

destinationDir - destination directory

runJob

```
public static void runJob(int jobId)
```

Run job

Parameters:

jobId - job identification

util

Class MD5Converter

```
java.lang.Object  
|  
+--util.MD5Converter
```

< [Constructors](#) > < [Methods](#) >

```
public class MD5Converter  
extends java.lang.Object
```

Converter to MD5 (Encrypt password)

Version:

1.0 03/04/2013

Author:

Faustino Alonso Posada (EII-Oviedo)

Constructors

MD5Converter

```
public MD5Converter()
```

Methods

convert

```
public static java.lang.String convert(java.lang.String clear)
```

Encrypt password to MD5 format

Parameters:

clear - plain text password

Returns:

md5 formatted password

util

Class SqliteFactory

```
java.lang.Object  
|  
+--util.SqliteFactory
```

< [Constructors](#) > < [Methods](#) >

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

SQLite Factory to obtain SQLite Database path

Version:

1.0 03/04/2013

Author:

Faustino Alonso Posada (EII-Oviedo)

Constructors

SqliteFactory

```
public SqliteFactory()
```

Methods

getSqlitePath

```
public static java.lang.String getSqlitePath()
```

Get SQLite path

Returns:

Database path

Package util.log

Class Summary

[Log](#)

Class to control LOG records

util.log

Class Log

```
java.lang.Object
|
+--util.log.Log
```

< [Constructors](#) > < [Methods](#) >

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

Class to control LOG records

Version:

1.0 03/04/2013

Author:

Faustino Alonso Posada (EII-Oviedo)

Constructors

Log

```
public Log()
```

Methods

writeLog

```
public static void writeLog(java.lang.String message,
                             java.lang.String reason)
```

Write a new LOG record

Parameters:

message - what happens?
reason - why it happens?

Package util.threads

Class Summary

[CheckFTPThread](#)

Check FTP Thread, it is used to check FTP connections

[CheckSQLThread](#)

Check SQL Thread, it is used to check database connections

[CopyExecutorThread](#)

Copy Executor Thread, it is used to make a copy

[DirectoryBackupThread](#)

Directory Backup Thread, it is used to throw new directory copy

[FTPBackupThread](#)

FTP Backup Thread, it is used to throw new FTP send

[SchedulerBackupThread](#)

Scheduler Backup Thread, it is used to throw new scheduler backup

[UpdateBackupThread](#)

Update Backup Thread, it is used to throw edited backup

util.threads

Class CheckFTPThread

```
java.lang.Object
|
+-- java.lang.Thread
|   |
|   +-- util.threads.CheckFTPThread
```

All Implemented Interfaces:

java.lang.Runnable

< [Constructors](#) > < [Methods](#) >

```
public class CheckFTPThread
extends java.lang.Thread
```

Check FTP Thread, it is used to check FTP connections

Version:

2.0 11/06/2013

Author:

Faustino Alonso Posada (EII-Oviedo)

Constructors

CheckFTPThread

```
public CheckFTPThread(entities.Ftp ftp)
```

Parameters:

ftp - FTP connection

Methods

run

```
public void run()
```

Overrides:

run in class java.lang.Thread

util.threads

Class CheckSQLThread

```
java.lang.Object
|
+--java.lang.Thread
|
+--util.threads.CheckSQLThread
```

All Implemented Interfaces:

java.lang.Runnable

< [Constructors](#) > < [Methods](#) >

```
public class CheckSQLThread
extends java.lang.Thread
```

Check SQL Thread, it is used to check database connections

Version:

2.0 11/06/2013

Author:

Faustino Alonso Posada (EII-Oviedo)

Constructors

CheckSQLThread

```
public CheckSQLThread(entities.SQLServerDatabase database)
```

Parameters:

database - SQL Server database

Methods

run

```
public void run()
```

Overrides:

run in class java.lang.Thread

util.threads

Class CopyExecutorThread

```
java.lang.Object
|
+-- java.lang.Thread
|   |
|   +-- util.threads.CopyExecutorThread
```

All Implemented Interfaces:

java.lang.Runnable

< [Constructors](#) > < [Methods](#) >

```
public class CopyExecutorThread
extends java.lang.Thread
```

Copy Executor Thread, it is used to make a copy

Version:

2.0 11/05/2013

Author:

Faustino Alonso Posada (EII-Oviedo)

Constructors

CopyExecutorThread

```
public CopyExecutorThread(logic.exec.ExecutableJob job,
                          int jobId)
```

Parameters:

job - ExecutableJob
jobId - job identification

Methods

run

```
public void run()
```

Overrides:

run in class java.lang.Thread

util.threads

Class DirectoryBackupThread

```
java.lang.Object
|
+--java.lang.Thread
|
+--util.threads.DirectoryBackupThread
```

All Implemented Interfaces:

java.lang.Runnable

< [Constructors](#) > < [Methods](#) >

```
public class DirectoryBackupThread
extends java.lang.Thread
```

Directory Backup Thread, it is used to throw new directory copy

Version:

1.0 03/04/2013

Author:

Faustino Alonso Posada (EII-Oviedo)

Constructors

DirectoryBackupThread

```
public DirectoryBackupThread(int jobId,
                             java.lang.String origin,
                             java.lang.String destination,
                             int backupId)
```

Parameters:

jobId - BackupJob Identification
origin - what do you want to save? (path)
destination - where do you want to save? (path)

DirectoryBackupThread

```
public DirectoryBackupThread(java.lang.String origin,
                             java.lang.String destination)
```

Parameters:

origin - what do you want to save? (path)
destination - where do you want to save? (path)

Methods

run

```
public void run()
```

Overrides:

run in class java.lang.Thread

util.threads

Class FTPBackupThread

```
java.lang.Object
|
+--java.lang.Thread
|   |
|   +--util.threads.FTPBackupThread
```

All Implemented Interfaces:

java.lang.Runnable

< [Constructors](#) > < [Methods](#) >

```
public class FTPBackupThread
extends java.lang.Thread
```

FTP Backup Thread, it is used to throw new FTP send

Version:

1.0 03/04/2013

Author:

Faustino Alonso Posada (EII-Oviedo)

Constructors

FTPBackupThread

```
public FTPBackupThread( java.lang.String host,
                        java.lang.String name,
                        java.lang.String pass,
                        java.lang.String origin,
                        java.lang.String destination)
```

Parameters:

host - FTP Server address
name - FTP Server user name
pass - FTP Server user password
origin - what do you want to save?
destination - FTP Server file name

Methods

run

```
public void run()
```

Overrides:

run in class java.lang.Thread

util.threads

Class SchedulerBackupThread

```
java.lang.Object
|
+--java.lang.Thread
|
+--util.threads.SchedulerBackupThread
```

All Implemented Interfaces:

java.lang.Runnable

```
public class SchedulerBackupThread
extends java.lang.Thread
```

Scheduler Backup Thread, it is used to throw new scheduler backup

Version:

1.0 03/04/2013

Author:

Faustino Alonso Posada (EII-Oviedo)

Constructors

SchedulerBackupThread

```
public SchedulerBackupThread(entities.Backup backup)
```

Methods

run

```
public void run()
```

Overrides:

run in class java.lang.Thread

util.threads

Class UpdateBackupThread

```
java.lang.Object
|
+-- java.lang.Thread
|   |
|   +-- util.threads.UpdateBackupThread
```

All Implemented Interfaces:

java.lang.Runnable

< [Constructors](#) > < [Methods](#) >

```
public class UpdateBackupThread
extends java.lang.Thread
```

Update Backup Thread, it is used to throw edited backup

Version:

2.0 11/05/2013

Author:

Faustino Alonso Posada (EII-Oviedo)

Constructors

UpdateBackupThread

```
public UpdateBackupThread(int backupId,  
                           entities.Backup backup)
```

Methods

run

```
public void run()
```

Overrides:

run in class java.lang.Thread

Package visual

Class Summary

[MainUI](#)

Main User Interface: main application class

[MainWindow](#)

Main Window, user interface window

visual

Class MainUI

```
java.lang.Object
|
+--visual.MainUI
```

< [Constructors](#) > < [Methods](#) >

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

Main User Interface: main application class

Version:

1.0 03/04/2013

Author:

Faustino Alonso Posada (EII-Oviedo)

Constructors

MainUI

```
public MainUI()
```

Methods

main

```
public static void main(java.lang.String[] args)
```

main application method

Parameters:

args -

visual

Class MainWindow

```
java.lang.Object
|
+--java.awt.Component
|   |
|   +--java.awt.Container
|       |
|       +--java.awt.Window
|           |
|           +--java.awt.Frame
|               |
|               +--javax.swing.JFrame
|                   |
|                   +--visual.MainWindow
```

All Implemented Interfaces:

java.awt.MenuContainer, java.awt.image.ImageObserver, java.io.Serializable,
javax.accessibility.Accessible, javax.swing.RootPaneContainer,
javax.swing.TransferHandler.HasGetTransferHandler, javax.swing.WindowConstants

< [Fields](#) > < [Constructors](#) > < [Methods](#) >

```
public class MainWindow  
extends javax.swing.JFrame
```

Main Window, user interface window

Version:

1.7 03/06/2013

Author:

Faustino Alonso Posada (EII-Oviedo)

Fields

CONFIGURATION

```
public static final int CONFIGURATION
```

CONNECTIONS

```
public static final int CONNECTIONS
```

JOBS

```
public static final int JOBS
```

LIST_BACKUPS

```
public static final int LIST_BACKUPS
```

LOG

```
public static final int LOG
```

LOGIN

```
public static final int LOGIN
```

RESULTS

```
public static final int RESULTS
```

authenticated

```
public static boolean authenticated
```

Constructors

MainWindow

```
public MainWindow()
```

Methods

getInstance

```
public static MainWindow getInstance()
```

loginUser

```
public void loginUser(java.lang.String user)
```

paintMainPane

```
public static void paintMainPane(int panelName)
```

setAuthenticated

```
public static void setAuthenticated(boolean authenticated)
```

Package visual.panels

Class Summary

[AddBackupPanel](#)

Add new Backup panel

[AddBackupPanel.JobsEditor](#)

[AddBackupPanel.JobsRender](#)

[BackupsPanel](#)

Backups panel

[BackupsPanel.BackupsTableEditor](#)

[BackupsPanel.BackupsTableRender](#)

[ConfigurationPanel](#)

Configuration Panel

[ConnectionsPanel](#)

Connections Panel

[ConnectionsPanel.FTPEditor](#)

[ConnectionsPanel.FTPRender](#)

[ConnectionsPanel.SQLEditor](#)

[ConnectionsPanel.SQLRender](#)

[EditBackupPanel](#)

Edit backup panel

[EditBackupPanel.JobsEditor](#)

[EditBackupPanel.JobsRender](#)

[JobsPanel](#)

Jobs panel

[JobsPanel.JobsEditor](#)

[JobsPanel.JobsRender](#)

[LogPanel](#)

Log panel

[LoginPage](#)

Login panel

[ResultsPanel](#)

Results panel

[ResultsPanel.JobsTableEditor](#)

[ResultsPanel.JobsTableRender](#)

[ResultsPanel.ResultsTableEditor](#)

[ResultsPanel.ResultsTableModel](#)

[ResultsPanel.ResultsTableRender](#)

visual.panels

Class AddBackupPanel

```
java.lang.Object
|
+-- java.awt.Component
|   |
|   +-- java.awt.Container
|       |
|       +-- javax.swing.JComponent
|           |
|           +-- javax.swing.JPanel
|               |
|               +-- visual.panels.AddBackupPanel
```

All Implemented Interfaces:

java.awt.MenuContainer, java.awt.image.ImageObserver, java.io.Serializable,
javax.accessibility.Accessible, javax.swing.TransferHandler.HasGetTransferHandler

< [Constructors](#) > < [Methods](#) >

```
public class AddBackupPanel
extends javax.swing.JPanel
```

Add new Backup panel

Version:

1.0 02/05/2013

Author:

Faustino Alonso Posada (EII-Oviedo)

Constructors

AddBackupPanel

```
public AddBackupPanel(BackupsPanel resultsPanel)
```

This is the default constructor

Parameters:

resultsPanel -

Methods

getJobsList

```
public java.util.ArrayList getJobsList()
```

setJobsList

```
public void setJobsList(java.util.ArrayList jobsList)
```

visual.panels

Class AddBackupPanel.JobsEditor

```
java.lang.Object
|
+-- javax.swing.AbstractCellEditor
|
+-- visual.panels.AddBackupPanel.JobsEditor
```

All Implemented Interfaces:

java.awt.event.ActionListener, java.io.Serializable, javax.swing.CellEditor,
javax.swing.table.TableCellEditor

< [Fields](#) > < [Constructors](#) > < [Methods](#) >

```
public class AddBackupPanel.JobsEditor
extends javax.swing.AbstractCellEditor
implements java.awt.event.ActionListener, javax.swing.table.TableCellEditor
```

Fields

EDIT

```
protected static final java.lang.String EDIT
```


visual.panels

Class AddBackupPanel.JobsRender

```
java.lang.Object
|
+--java.awt.Component
|   |
|   +--java.awt.Container
|       |
|       +--javax.swing.JComponent
|           |
|           +--javax.swing.JLabel
|               |
|               +--visual.panels.AddBackupPanel.JobsRender
```

All Implemented Interfaces:

java.awt.MenuContainer, java.awt.image.ImageObserver, java.io.Serializable,
javax.accessibility.Accessible, javax.swing.SwingConstants,
javax.swing.TransferHandler.HasGetTransferHandler, javax.swing.table.TableCellRenderer

< [Constructors](#) > < [Methods](#) >

```
public class AddBackupPanel.JobsRender
extends javax.swing.JLabel
implements javax.swing.table.TableCellRenderer
```

Constructors

AddBackupPanel.JobsRender

```
public AddBackupPanel.JobsRender(boolean isBordered)
```

Methods

getTableCellRendererComponent

```
public java.awt.Component getTableCellRendererComponent( javax.swing.JTable
table,
color,
java.lang.Object
boolean isSelected,
boolean hasFocus,
int row,
int column)
```

visual.panels

Class BackupsPanel

```
java.lang.Object
|
+--java.awt.Component
|   |
|   +--java.awt.Container
|       |
|       +--javax.swing.JComponent
|           |
|           +--javax.swing.JPanel
|               |
|               +--visual.panels.BackupsPanel
```

All Implemented Interfaces:

java.awt.MenuContainer, java.awt.image.ImageObserver, java.io.Serializable,
javax.accessibility.Accessible, javax.swing.TransferHandler.HasGetTransferHandler

< [Constructors](#) > < [Methods](#) >

```
public class BackupsPanel
extends javax.swing.JPanel
```

Backups panel

Version:

2.0 23/05/2013

Author:

Faustino Alonso Posada (EII-Oviedo)

Constructors

BackupsPanel

```
public BackupsPanel()
```

This is the default constructor

Methods

returnPanel

```
public void returnPanel()
```

visual.panels

Class BackupsPanel.BackupsTableEditor

```
java.lang.Object
|
+--javax.swing.AbstractCellEditor
|
+--visual.panels.BackupsPanel.BackupsTableEditor
```

All Implemented Interfaces:

java.awt.event.ActionListener, java.io.Serializable, javax.swing.CellEditor,
javax.swing.table.TableCellEditor

< [Fields](#) > < [Constructors](#) > < [Methods](#) >

```
public class BackupsPanel.BackupsTableEditor
extends javax.swing.AbstractCellEditor
implements java.awt.event.ActionListener, javax.swing.table.TableCellEditor
```

Fields

EDIT

```
protected static final java.lang.String EDIT
```

Constructors

BackupsPanel.BackupsTableEditor

```
public BackupsPanel.BackupsTableEditor(javax.swing.JTable jTable1)
```

Methods

actionPerformed

```
public void actionPerformed(java.awt.event.ActionEvent e)
```

getCellEditorValue

```
public java.lang.Object getCellEditorValue()
```

getTableCellEditorComponent

```
public java.awt.Component getTableCellEditorComponent( javax.swing.JTable
table,
                                                    java.lang.Object value,
                                                    boolean isSelected,
                                                    int row,
                                                    int column)
```

visual.panels

Class BackupsPanel.BackupsTableRender

```
java.lang.Object
|
+-- java.awt.Component
|   |
|   +-- java.awt.Container
|       |
|       +-- javax.swing.JComponent
|           |
|           +-- javax.swing.JLabel
|               |
|               +-- visual.panels.BackupsPanel.BackupsTableRender
```

All Implemented Interfaces:

```
java.awt.MenuContainer, java.awt.image.ImageObserver, java.io.Serializable,
javax.accessibility.Accessible, javax.swing.SwingConstants,
javax.swing.TransferHandler.HasGetTransferHandler, javax.swing.table.TableCellRenderer
```

< [Constructors](#) > < [Methods](#) >

```
public class BackupsPanel.BackupsTableRender
extends javax.swing.JLabel
implements javax.swing.table.TableCellRenderer
```

Constructors

BackupsPanel.BackupsTableRender

```
public BackupsPanel.BackupsTableRender(boolean isBordered)
```

Methods

getTableCellRendererComponent

```
public java.awt.Component getTableCellRendererComponent( javax.swing.JTable
table,
color,
java.lang.Object
boolean isSelected,
boolean hasFocus,
int row,
int column)
```

visual.panels

Class ConfigurationPanel

```
java.lang.Object
|
+-- java.awt.Component
|   |
|   +-- java.awt.Container
|       |
|       +-- javax.swing.JComponent
|           |
|           +-- javax.swing.JPanel
|               |
|               +-- visual.panels.ConfigurationPanel
```

All Implemented Interfaces:

java.awt.MenuContainer, java.awt.image.ImageObserver, java.io.Serializable,
javax.accessibility.Accessible, javax.swing.TransferHandler.HasGetTransferHandler

< [Constructors](#) >

```
public class ConfigurationPanel
extends javax.swing.JPanel
```

Configuration Panel

Version:

1.0 03/04/2013

Author:

Faustino Alonso Posada (EII-Oviedo)

Constructors

ConfigurationPanel

```
public ConfigurationPanel()
```

This is the default constructor

visual.panels

Class ConnectionsPanel

```
java.lang.Object
|
+-- java.awt.Component
    |
    +-- java.awt.Container
        |
        +-- javax.swing.JComponent
            |
            +-- javax.swing.JPanel
                |
                +-- visual.panels.ConnectionsPanel
```

All Implemented Interfaces:

java.awt.MenuContainer, java.awt.image.ImageObserver, java.io.Serializable,
javax.accessibility.Accessible, javax.swing.TransferHandler.HasGetTransferHandler

< [Constructors](#) >

```
public class ConnectionsPanel
extends javax.swing.JPanel
```

Connections Panel

Version:

1.0 03/04/2013

Author:

Faustino Alonso Posada (EII-Oviedo)

Constructors

ConnectionsPanel

```
public ConnectionsPanel(visual.MainWindow instance)
```

This is the default constructor

Parameters:

instance -

visual.panels

Class ConnectionsPanel.FTPEditor

```
java.lang.Object
|
+--javax.swing.AbstractCellEditor
|
+--visual.panels.ConnectionsPanel.FTPEditor
```

All Implemented Interfaces:

java.awt.event.ActionListener, java.io.Serializable, javax.swing.CellEditor,
javax.swing.table.TableCellEditor

< [Fields](#) > < [Constructors](#) > < [Methods](#) >

```
public class ConnectionsPanel.FTPEditor
extends javax.swing.AbstractCellEditor
implements java.awt.event.ActionListener, javax.swing.table.TableCellEditor
```

Fields

EDIT

```
protected static final java.lang.String EDIT
```

Constructors

ConnectionsPanel.FTPEditor

```
public ConnectionsPanel.FTPEditor(javax.swing.JTable jTable1)
```

Methods

actionPerformed

```
public void actionPerformed(java.awt.event.ActionEvent e)
```

getCellEditorValue

```
public java.lang.Object getCellEditorValue()
```

getTableCellEditorComponent

```
public java.awt.Component getTableCellEditorComponent( javax.swing.JTable
table,
                                                    java.lang.Object value,
                                                    boolean isSelected,
                                                    int row,
                                                    int column)
```

visual.panels

Class ConnectionsPanel.FTPRender

```
java.lang.Object
|
+-- java.awt.Component
|   |
|   +-- java.awt.Container
|       |
|       +-- javax.swing.JComponent
|           |
|           +-- javax.swing.JLabel
|               |
|               +-- visual.panels.ConnectionsPanel.FTPRender
```

All Implemented Interfaces:

```
java.awt.MenuContainer, java.awt.image.ImageObserver, java.io.Serializable,
javax.accessibility.Accessible, javax.swing.SwingConstants,
javax.swing.TransferHandler.HasGetTransferHandler, javax.swing.table.TableCellRenderer
```

< [Constructors](#) > < [Methods](#) >

```
public class ConnectionsPanel.FTPRender
extends javax.swing.JLabel
implements javax.swing.table.TableCellRenderer
```

Constructors

ConnectionsPanel.FTPRender

```
public ConnectionsPanel.FTPRender(boolean isBordered)
```

Methods

getTableCellRendererComponent

```
public java.awt.Component getTableCellRendererComponent( javax.swing.JTable
table,
                                                         java.lang.Object
color,
                                                         boolean isSelected,
                                                         boolean hasFocus,
                                                         int row,
                                                         int column)
```

visual.panels

Class ConnectionsPanel.SQLEditor

```
java.lang.Object
|
+-- javax.swing.AbstractCellEditor
|
+-- visual.panels.ConnectionsPanel.SQLEditor
```

All Implemented Interfaces:

java.awt.event.ActionListener, java.io.Serializable, javax.swing.CellEditor,
javax.swing.table.TableCellEditor

< [Fields](#) > < [Constructors](#) > < [Methods](#) >

```
public class ConnectionsPanel.SQLEditor
extends javax.swing.AbstractCellEditor
implements java.awt.event.ActionListener, javax.swing.table.TableCellEditor
```

Fields

EDIT

```
protected static final java.lang.String EDIT
```

Constructors

ConnectionsPanel.SQLEditor

```
public ConnectionsPanel.SQLEditor( javax.swing.JTable jTable1)
```

Methods

actionPerformed

```
public void actionPerformed(java.awt.event.ActionEvent e)
```

getCellEditorValue

```
public java.lang.Object getCellEditorValue()
```

getTableCellEditorComponent

```
public java.awt.Component getTableCellEditorComponent(javax.swing.JTable
table,
                                                       java.lang.Object value,
                                                       boolean isSelected,
                                                       int row,
                                                       int column)
```

visual.panels

Class ConnectionsPanel.SQLRender

```
java.lang.Object
|
+-- java.awt.Component
|   |
|   +-- java.awt.Container
|       |
|       +-- javax.swing.JComponent
|           |
|           +-- javax.swing.JLabel
|               |
|               +-- visual.panels.ConnectionsPanel.SQLRender
```

All Implemented Interfaces:

java.awt.MenuContainer, java.awt.image.ImageObserver, java.io.Serializable,
javax.accessibility.Accessible, javax.swing.SwingConstants,
javax.swing.TransferHandler.HasGetTransferHandler, javax.swing.table.TableCellRenderer

< [Constructors](#) > < [Methods](#) >

```
public class ConnectionsPanel.SQLRender
extends javax.swing.JLabel
implements javax.swing.table.TableCellRenderer
```

Constructors

ConnectionsPanel.SQLRender

```
public ConnectionsPanel.SQLRender(boolean isBordered)
```

Methods

getTableCellRendererComponent

```
public java.awt.Component getTableCellRendererComponent(javax.swing.JTable
table,
color,
java.lang.Object
boolean isSelected,
boolean hasFocus,
int row,
int column)
```

visual.panels

Class EditBackupPanel

```
java.lang.Object
|-- java.awt.Component
    |-- java.awt.Container
        |-- javax.swing.JComponent
            |-- javax.swing.JPanel
                |-- visual.panels.EditBackupPanel
```

All Implemented Interfaces:

java.awt.MenuContainer, java.awt.image.ImageObserver, java.io.Serializable,
javax.accessibility.Accessible, javax.swing.TransferHandler.HasGetTransferHandler

< [Constructors](#) > < [Methods](#) >

```
public class EditBackupPanel
extends javax.swing.JPanel
```

Edit backup panel

Version:

1.0 03/04/2013

Author:

Faustino Alonso Posada (EII-Oviedo)

Constructors

EditBackupPanel

```
public EditBackupPanel(BackupsPanel resultsPanel,  
                      entities.Backup backup)
```

This is the default constructor

Parameters:

resultsPanel -
backup -

Methods

getJobsList

```
public java.util.ArrayList getJobsList()
```

setJobsList

```
public void setJobsList(java.util.ArrayList jobsList)
```

visual.panels

Class EditBackupPanel.JobsEditor

```
java.lang.Object  
|  
+-- javax.swing.AbstractCellEditor  
|  
+-- visual.panels.EditBackupPanel.JobsEditor
```

All Implemented Interfaces:

java.awt.event.ActionListener, java.io.Serializable, javax.swing.CellEditor,
javax.swing.table.TableCellEditor

< [Fields](#) > < [Constructors](#) > < [Methods](#) >

```
public class EditBackupPanel.JobsEditor  
extends javax.swing.AbstractCellEditor  
implements java.awt.event.ActionListener, javax.swing.table.TableCellEditor
```

Fields

visual.panels

Class EditBackupPanel.JobsRender

```
java.lang.Object
|
+--java.awt.Component
|   |
|   +--java.awt.Container
|       |
|       +--javax.swing.JComponent
|           |
|           +--javax.swing.JLabel
|               |
|               +--visual.panels.EditBackupPanel.JobsRender
```

All Implemented Interfaces:

java.awt.MenuContainer, java.awt.image.ImageObserver, java.io.Serializable,
javax.accessibility.Accessible, javax.swing.SwingConstants,
javax.swing.TransferHandler.HasGetTransferHandler, javax.swing.table.TableCellRenderer

< [Constructors](#) > < [Methods](#) >

```
public class EditBackupPanel.JobsRender
extends javax.swing.JLabel
implements javax.swing.table.TableCellRenderer
```

Constructors

EditBackupPanel.JobsRender

```
public EditBackupPanel.JobsRender(boolean isBordered)
```

Methods

getTableCellRendererComponent

```
public java.awt.Component getTableCellRendererComponent( javax.swing.JTable
table,
color,
java.lang.Object
boolean isSelected,
boolean hasFocus,
int row,
int column)
```

visual.panels

Class JobsPanel

```
java.lang.Object
|
+--java.awt.Component
|   |
|   +--java.awt.Container
|       |
|       +--javax.swing.JComponent
|           |
|           +--javax.swing.JPanel
|               |
|               +--visual.panels.JobsPanel
```

All Implemented Interfaces:

java.awt.MenuContainer, java.awt.image.ImageObserver, java.io.Serializable,
javax.accessibility.Accessible, javax.swing.TransferHandler.HasGetTransferHandler

< [Constructors](#) >

```
public class JobsPanel
extends javax.swing.JPanel
```

Jobs panel

Version:

1.0 03/04/2013

Author:

Faustino Alonso Posada (EII-Oviedo)

Constructors

JobsPanel

```
public JobsPanel()
```

visual.panels

Class JPanel.JobsEditor

```
java.lang.Object
|
+--javax.swing.AbstractCellEditor
|
+--visual.panels.JPanel.JobsEditor
```

All Implemented Interfaces:

java.awt.event.ActionListener, java.io.Serializable, javax.swing.CellEditor, javax.swing.table.TableCellEditor

< [Fields](#) > < [Constructors](#) > < [Methods](#) >

```
public class JPanel.JobsEditor
extends javax.swing.AbstractCellEditor
implements java.awt.event.ActionListener, javax.swing.table.TableCellEditor
```

Fields

EDIT

```
protected static final java.lang.String EDIT
```

Constructors

JPanel.JobsEditor

```
public JPanel.JobsEditor(javax.swing.JTable jTable1)
```

Methods

actionPerformed

```
public void actionPerformed(java.awt.event.ActionEvent e)
```

getCellEditorValue

```
public java.lang.Object getCellEditorValue()
```

getTableCellEditorComponent

```
public java.awt.Component getTableCellEditorComponent( javax.swing.JTable
table,
                                                    java.lang.Object value,
                                                    boolean isSelected,
                                                    int row,
                                                    int column)
```

visual.panels

Class JobsPanel.JobsRender

```
java.lang.Object
|
+-- java.awt.Component
|   |
|   +-- java.awt.Container
|       |
|       +-- javax.swing.JComponent
|           |
|           +-- javax.swing.JLabel
|               |
|               +-- visual.panels.JobsPanel.JobsRender
```

All Implemented Interfaces:

```
java.awt.MenuContainer, java.awt.image.ImageObserver, java.io.Serializable,
javax.accessibility.Accessible, javax.swing.SwingConstants,
javax.swing.TransferHandler.HasGetTransferHandler, javax.swing.table.TableCellRenderer
```

< [Constructors](#) > < [Methods](#) >

```
public class JobsPanel.JobsRender
extends javax.swing.JLabel
implements javax.swing.table.TableCellRenderer
```

Constructors

JobsPanel.JobsRender

```
public JobsPanel.JobsRender(boolean isBordered)
```

Methods

getTableCellRendererComponent

```
public java.awt.Component getTableCellRendererComponent( javax.swing.JTable
table,
color,
java.lang.Object
boolean isSelected,
boolean hasFocus,
int row,
int column)
```

visual.panels

Class LogPanel

```
java.lang.Object
|
+-- java.awt.Component
|   |
|   +-- java.awt.Container
|       |
|       +-- javax.swing.JComponent
|           |
|           +-- javax.swing.JPanel
|               |
|               +-- visual.panels.LogPanel
```

All Implemented Interfaces:

java.awt.MenuContainer, java.awt.image.ImageObserver, java.io.Serializable,
javax.accessibility.Accessible, javax.swing.TransferHandler.HasGetTransferHandler

< [Constructors](#) >

```
public class LogPanel
extends javax.swing.JPanel
```

Log panel

Version:

1.0 15/04/2013

Author:

Faustino Alonso Posada (EII-Oviedo)

Constructors

LogPanel

```
public LogPanel()
```

visual.panels

Class LoginPanel

```
java.lang.Object
|
+--java.awt.Component
|   |
|   +--java.awt.Container
|       |
|       +--javax.swing.JComponent
|           |
|           +--javax.swing.JPanel
|               |
|               +--visual.panels.LoginPanel
```

All Implemented Interfaces:

java.awt.MenuContainer, java.awt.image.ImageObserver, java.io.Serializable,
javax.accessibility.Accessible, javax.swing.TransferHandler.HasGetTransferHandler

< [Constructors](#) > < [Methods](#) >

```
public class LoginPanel
extends javax.swing.JPanel
```

Login panel

Version:

1.0 03/04/2013

Author:

Faustino Alonso Posada (EII-Oviedo)

Constructors

LoginPanel

```
public LoginPanel(visual.MainWindow main)
```

This is the default constructor

Methods

getUserTextField

```
public javax.swing.JTextField getUserTextField()
```

This method initializes userTextField

Returns:

javax.swing.JTextField

visual.panels

Class ResultsPanel

```
java.lang.Object
|
+--java.awt.Component
|   |
|   +--java.awt.Container
|       |
|       +--javax.swing.JComponent
|           |
|           +--javax.swing.JPanel
|               |
|               +--visual.panels.ResultsPanel
```

All Implemented Interfaces:

java.awt.MenuContainer, java.awt.image.ImageObserver, java.io.Serializable,
javax.accessibility.Accessible, javax.swing.TransferHandler.HasGetTransferHandler

< [Constructors](#) > < [Methods](#) >

```
public class ResultsPanel
extends javax.swing.JPanel
```

Results panel

Version:

1.0 03/04/2013

Author:

Faustino Alonso Posada (EII-Oviedo)

Constructors

ResultsPanel

```
public ResultsPanel()
```

Methods

formateDate

```
public java.lang.String formateDate(java.util.Date date)
```

visual.panels

Class ResultsPanel.JobsTableEditor

```
java.lang.Object
|
+--javax.swing.AbstractCellEditor
|
+--visual.panels.ResultsPanel.JobsTableEditor
```

All Implemented Interfaces:

java.awt.event.ActionListener, java.io.Serializable, javax.swing.CellEditor,
javax.swing.table.TableCellEditor

< [Fields](#) > < [Constructors](#) > < [Methods](#) >

```
public class ResultsPanel.JobsTableEditor
extends javax.swing.AbstractCellEditor
implements java.awt.event.ActionListener, javax.swing.table.TableCellEditor
```

Fields

EDIT

```
protected static final java.lang.String EDIT
```

Constructors

ResultsPanel.JobsTableEditor

```
public ResultsPanel.JobsTableEditor(javax.swing.JTable jTable1)
```

Methods

actionPerformed

```
public void actionPerformed(java.awt.event.ActionEvent e)
```

getCellEditorValue

```
public java.lang.Object getCellEditorValue()
```

getTableCellEditorComponent

```
public java.awt.Component getTableCellEditorComponent( javax.swing.JTable
table,
                                                    java.lang.Object value,
                                                    boolean isSelected,
                                                    int row,
                                                    int column)
```

visual.panels

Class ResultsPanel.JobsTableRender

```
java.lang.Object
|
+-- java.awt.Component
|   |
|   +-- java.awt.Container
|       |
|       +-- javax.swing.JComponent
|           |
|           +-- javax.swing.JLabel
|               |
|               +-- visual.panels.ResultsPanel.JobsTableRender
```

All Implemented Interfaces:

```
java.awt.MenuContainer, java.awt.image.ImageObserver, java.io.Serializable,
javax.accessibility.Accessible, javax.swing.SwingConstants,
javax.swing.TransferHandler.HasGetTransferHandler, javax.swing.table.TableCellRenderer
```

< [Constructors](#) > < [Methods](#) >

```
public class ResultsPanel.JobsTableRender
extends javax.swing.JLabel
implements javax.swing.table.TableCellRenderer
```

Constructors

ResultsPanel.JobsTableRender

```
public ResultsPanel.JobsTableRender(boolean isBordered)
```

Methods

getTableCellRendererComponent

```
public java.awt.Component getTableCellRendererComponent( javax.swing.JTable
table,
value,
java.lang.Object
boolean isSelected,
boolean hasFocus,
int row,
int column)
```

visual.panels

Class ResultsPanel.ResultsTableEditor

```
java.lang.Object
|
+-- javax.swing.AbstractCellEditor
|
+-- visual.panels.ResultsPanel.ResultsTableEditor
```

All Implemented Interfaces:

java.awt.event.ActionListener, java.io.Serializable, javax.swing.CellEditor,
javax.swing.table.TableCellEditor

< [Fields](#) > < [Constructors](#) > < [Methods](#) >

```
public class ResultsPanel.ResultsTableEditor
extends javax.swing.AbstractCellEditor
implements java.awt.event.ActionListener, javax.swing.table.TableCellEditor
```

Fields

EDIT

```
protected static final java.lang.String EDIT
```

Constructors

ResultsPanel.ResultsTableEditor

```
public ResultsPanel.ResultsTableEditor( javax.swing.JTable jTable1)
```

Methods

actionPerformed

```
public void actionPerformed(java.awt.event.ActionEvent e)
```

getCellEditorValue

```
public java.lang.Object getCellEditorValue()
```

getTableCellEditorComponent

```
public java.awt.Component getTableCellEditorComponent(javax.swing.JTable
table,
                                                    java.lang.Object value,
                                                    boolean isSelected,
                                                    int row,
                                                    int column)
```

visual.panels

Class ResultsPanel.ResultsTableModel

```
java.lang.Object
|
+-- javax.swing.table.AbstractTableModel
|   |
|   +-- javax.swing.table.DefaultTableModel
|       |
|       +-- visual.panels.ResultsPanel.ResultsTableModel
```

All Implemented Interfaces:

java.io.Serializable, javax.swing.table.TableModel

< [Constructors](#) > < [Methods](#) >

```
public class ResultsPanel.ResultsTableModel
extends javax.swing.table.DefaultTableModel
```

Constructors

ResultsPanel.ResultsTableModel

```
public ResultsPanel.ResultsTableModel(java.lang.Object[][] data,  
                                     java.lang.String[] columnNames)
```

Methods

isCellEditable

```
public boolean isCellEditable(int row,  
                              int column)
```

Overrides:

isCellEditable in class javax.swing.table.DefaultTableModel

visual.panels

Class ResultsPanel.ResultsTableRender

```
java.lang.Object  
|  
+-- java.awt.Component  
|   |  
|   +-- java.awt.Container  
|       |  
|       +-- javax.swing.JComponent  
|           |  
|           +-- javax.swing.JLabel  
|               |  
|               +-- visual.panels.ResultsPanel.ResultsTableRender
```

All Implemented Interfaces:

java.awt.MenuContainer, java.awt.image.ImageObserver, java.io.Serializable,
javax.accessibility.Accessible, javax.swing.SwingConstants,
javax.swing.TransferHandler.HasGetTransferHandler, javax.swing.table.TableCellRenderer

< [Constructors](#) > < [Methods](#) >

```
public class ResultsPanel.ResultsTableRender  
extends javax.swing.JLabel  
implements javax.swing.table.TableCellRenderer
```

Constructors

ResultsPanel.ResultsTableRender

```
public ResultsPanel.ResultsTableRender(boolean isBordered)
```

Methods

getTableCellRenderComponent

```
public java.awt.Component getTableCellRenderComponent(javax.swing.JTable  
table,  
value, java.lang.Object  
boolean isSelected,  
boolean hasFocus,  
int row,  
int column)
```

