MEMORIA TFC

TECNOLOGIA J2EE

ALUMNO: AITOR ZUMELAGA CASTILLO.

CONSULTOR: OSCAR ESCUDERO SANCHEZ.

14 DE ENERO DE 2009.

Índice

1.Introducción:

1.1.Descripción del proyecto:

El proyecto consiste en una aplicación para la recogida y gestión de incidencias informáticas y mantenimiento de una base de datos en la que podamos guardar los datos de cada equipo y sus respectivas averías. Esta aplicación será utilizada en el centro de formación profesional de Usurbil (Gipuzkoa) donde actualmente trabajo en el departamento de mantenimiento informático.

Las diferentes incidencias que se pretenden recoger y gestionar tratan sobre:

1.Problemas de Hardware.
2.Problemas de Software.
3.Configuración de equipos nuevos.
4.Problemas de red.
5.Destrozos, robos...

A día de hoy se utiliza una aplicación muy sencilla a la que solo puede acceder el profesorado y el personal de administración.

La idea es realizar una aplicación a la que se pueda acceder a través de la intranet y que los alumnos puedan dar cierto tipo de avisos para evitar los destrozos y robos de ratones etc... que se vienen sucediendo últimamente y de esta manera se pueda identificar más fácilmente a las personas responsables de estos actos vándalos.

En la aplicación actual solo existe la posibilidad de enviar dos tipos de incidencias; averías de software o hardware. Mediante la nueva aplicación se pretende que se puedan especificar más las averías para que así cada uno de los responsables de informática pueda acceder directamente a las averías de cada área. En la actualidad todos los administradores recogemos todas las incidencias y cada cual atiende a las suyas pero para ello tenemos que leer todas las existentes.

Por otra parte también se ve la necesidad de crear una base de datos de los equipos por etiqueta donde se introduzcan todas los equipos con sus características, fecha de adquisición, nombre del distribuidor, garantía, averías y reparaciones

1.2.Participantes:

En este proyecto al tratarse de un TFC individual, no tendremos cliente, director de proyecto... Hay un único actor que participa en el diseño, dirección, gestión e implementación del trabajo que en este caso soy yo.

1.3. Objetivos :

Técnicos:

El objetivo es la creación de una herramienta que se pueda visualizar por medio de un navegador (Firefox, Explorer...). Para ello se cree conveniente la utilización de J2EE, ya que es un sistema muy adecuado para este tipo de aplicaciones. Además no supone el pago de ninguna licencia, tampoco ningún coste añadido a la creación y mantenimiento de la aplicación.

Es también un objetivo en si, el familiarizarme y aprender a utilizar esta tecnología, ya que aunque conozco más o menos la arquitectura y su lógica, nunca he realizado una aplicación J2EE.

Para el desarrollo del proyecto se pretende utilizar *Eclipse* como entorno de desarrollo, *Tomcat* como servidor Web y *MySql* como sistema gestor de bases de datos. Se ha decidido utilizar estas herramientas porque son gratuitas y algunas de las más utilizadas, según el resultado de la investigación en la red y la lectura de diferentes manuales sobre la tecnología anteriormente mencionada.

Para la elaboración de esta aplicación utilizaremos el modelo de patrón MVC. Que separa el modelo, la vista y el controlador.

Funcionales:

Las funcionalidades que debe cubrir el proyecto son:

●Facilidad para introducir los avisos:
Tiene que ser una herramienta sencilla para los usuarios y sin demasiados apartados para rellenar.

●Máxima claridad posible para que los avisos lleguen al administrador correspondiente según el área de responsabilidad de cada uno.

●Acceso de los administradores a una base de datos para comprobar si se repiten el mismo tipo de averías en un equipo concreto, cómo se soluciona una avería etc...

1.4. Planificación:

La planificación del proyecto se hace respetando las fechas de entrega del plan docente para el seguimiento de la evaluación continua:

Pec1 1/10/2008 Plan de trabajo.

-Del 18 de Septiembre al 22 Septiembre:
Decisión de cual es la necesidad que va a cubrir el proyecto en general.

-Del 23 de Septiembre al 26 de Septiembre:
Descripción de las especificaciones que cubrirá el proyecto.

-Del 26 de Septiembre al 1 de Octubre:
Planificación del TFC y entrega del plan de trabajo donde se especifican las características funcionales, técnicas y temporalidad del TFC.

Pec2 5/11/2008 Fase de análisis y diseño.

Definición de la solución que se va a implantar. Se hará entrega de diagramas de clases, clases de uso y el diseño de la base de datos.

-Del 2 al 5 de Octubre:
Casos de uso.

-Del 6 al 12 de Octubre:
Diagramas de Estado y de Secuencia.

-Del 13 al 19 de Octubre:
Diagrama de clases.

-Del 20 al 26 de Octubre:
Diagrama E/R.

-Del 27 de Octubre al 30 de Octubre:
Descripción de vistas.

-Del 31 de Octubre al 5 de Noviembre:
Redacción del documento.

Pec3 17/12/2008 Fase de Desarrollo.

Diseño del prototipo e implementación.

-Del 6 al 9 de Noviembre:
Crear la base de datos e introducir datos.

-Del 10 al 16 de Noviembre:
Crear una inserción sencilla de un dato desde una vista utilizando todas las partes de la aplicación. Vista, Modelo y Controlador.

-Del 17 de Noviembre al 7 de Diciembre:
Ir completando la aplicación sin mirar al diseño hasta que funcione completamente

-Del 8 al 16 de Diciembre:
Crear los estilos mediante CSS y desplegar la aplicación.

Entrega Final 14/01/2009

Entrega de la implementación del proyecto, la memoria,y la presentación.

-Del 17 al 23 de Diciembre:
Ultimas pruebas y finalizar los flecos que puedan quedar.

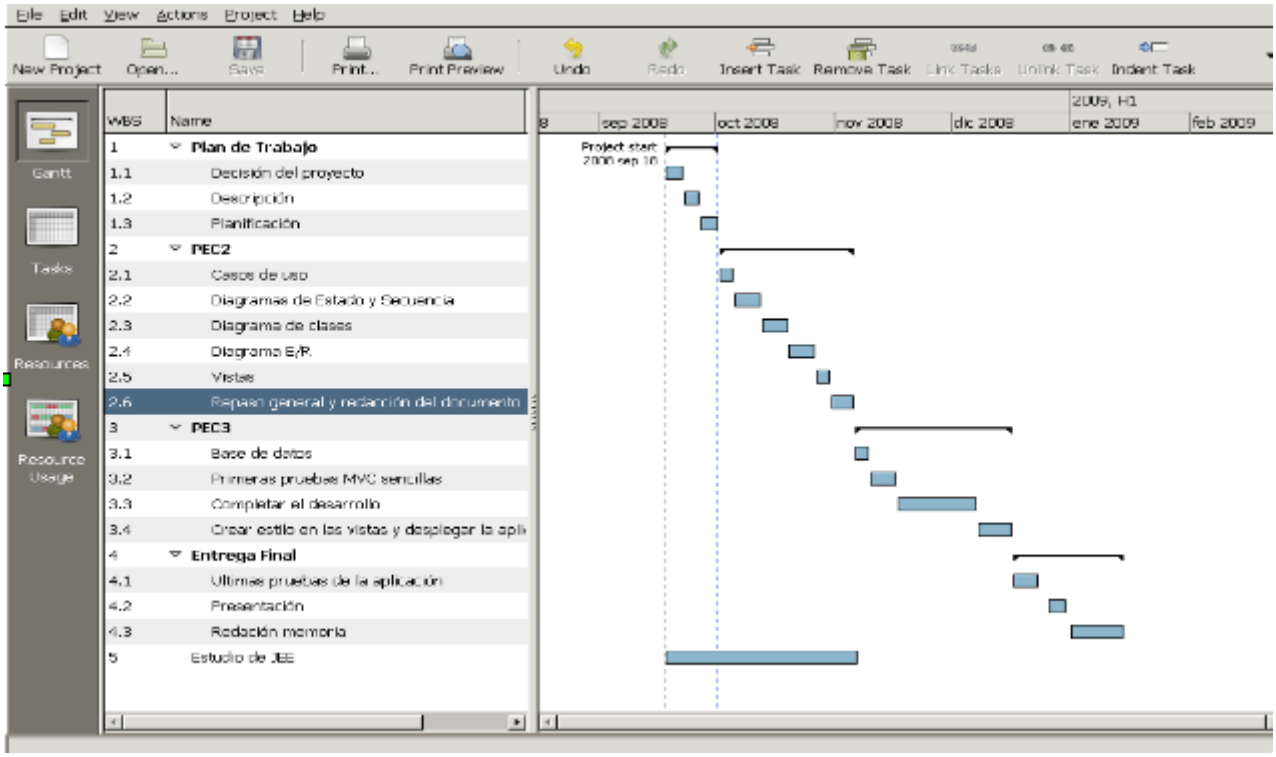
-Del 26 al 30 de Diciembre:
Elaborar la presentación del proyecto.

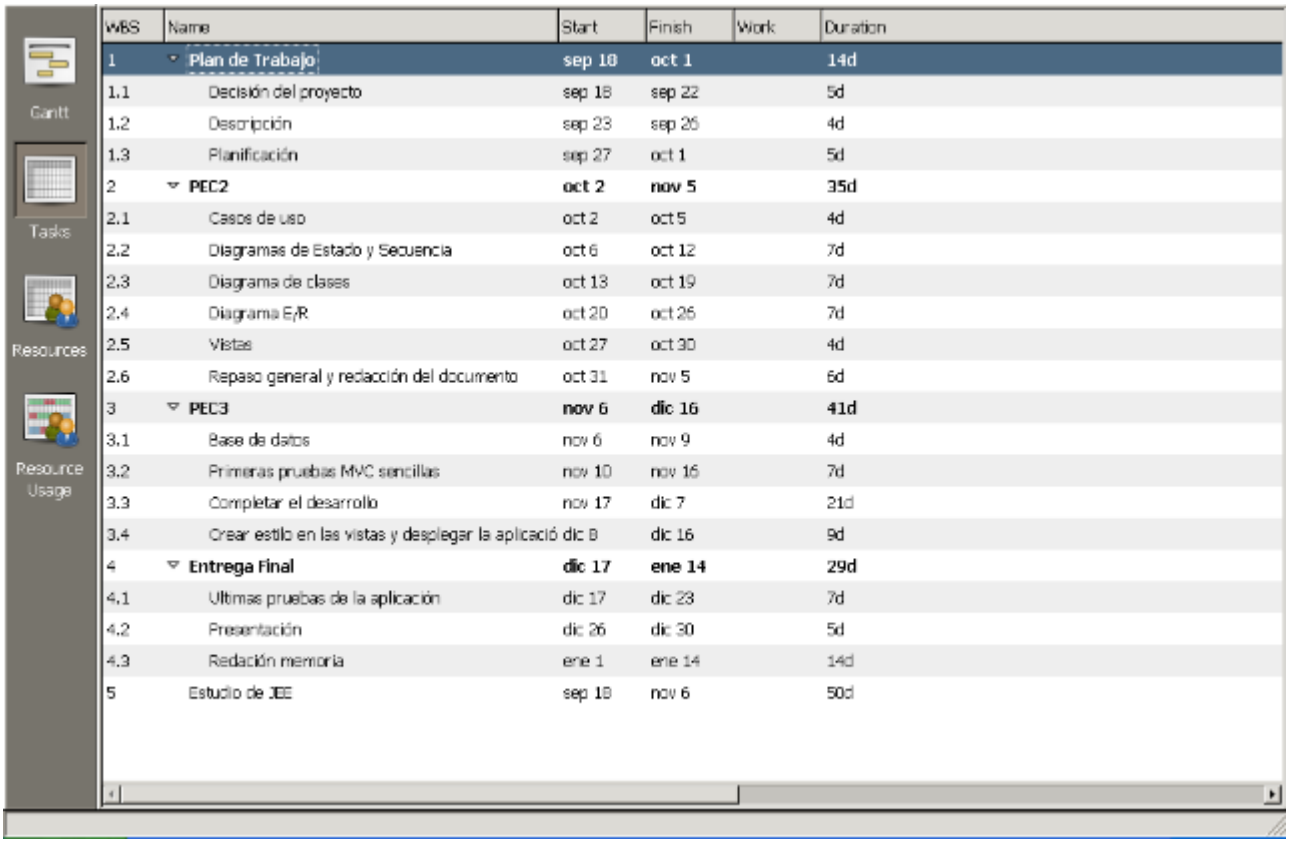
-Del 2 de Enero al 14:
Redactar la memoria y comprimir toda la documentación en un zip.

En la planificación, además de las entregas de las Pec se especifica una fase de aprendizaje de la tecnología J2EE paralela a la realización de las 2 primeras Pec, ya que no conozco mucho esta tecnología y creo necesario ir documentandome y probando con diferentes tutoriales antes de empezar la implementación de la aplicación en si.

En cuanto a la dedicación, se estiman 10 horas semanales. 2 horas al día de Lunes a Viernes, reservando los fines de semana para situaciones extraordinarias en las que las horas dedicadas durante la semana no sean suficientes para llegar al objetivo marcado en esa semana.

A continuación se muestra el diagrama de Gantt con la temporalización del plan de trabajo y las entregas para la realización del TFC J2EE.


File Edit View Actions Project Help
New Project
Open...
Save
Print...
Print Preview
Undo
Redo
Insert Task
Remove Task
Link Tasks
Unlink Task
Indent Task
Gantt
Tasks
Resources
Resource Usage
WBS
Name
2008, H1
sep 2008
oct 2008
nov 2008
dic 2008
ene 2009
feb 2009
Project start: 2008 sep 18
1 Plan de Trabajo
1.1 Decisión del proyecto
1.2 Descripción
1.3 Planificación
2 PEC2
2.1 Casos de uso
2.2 Diagramas de Estado y Secuencia
2.3 Diagrama de clases
2.4 Diagrama E/R
2.5 Vistas
2.6 Repaso general y redacción del documento
3 PEC3
3.1 Base de datos
3.2 Primeras pruebas MVC sencillas
3.3 Completar el desarrollo
3.4 Crear estilo en las vistas y desplegar la apli
4 Entrega Final
4.1 Ultimas pruebas de la aplicación
4.2 Presentación
4.3 Redación memoria
5 Estudio de JEE


Gantt
Tasks
Resources
Resource Usage
WBS
Name
Start
Finish
Work
Duration
1 Plan de Trabajo sep 18 oct 1 14d
1.1 Decisión del proyecto sep 18 sep 22 5d
1.2 Descripción sep 23 sep 26 4d
1.3 Planificación sep 27 oct 1 5d
2 PEC2 oct 2 nov 5 35d
2.1 Casos de uso oct 2 oct 5 4d
2.2 Diagramas de Estado y Secuencia oct 6 oct 12 7d
2.3 Diagrama de clases oct 13 oct 19 7d
2.4 Diagrama E/R oct 20 oct 26 7d
2.5 Vistas oct 27 oct 30 4d
2.6 Repaso general y redacción del documento oct 31 nov 5 6d
3 PEC3 nov 6 dic 16 41d
3.1 Base de datos nov 6 nov 9 4d
3.2 Primeras pruebas MVC sencillas nov 10 nov 16 7d
3.3 Completar el desarrollo nov 17 dic 7 21d
3.4 Crear estilo en las vistas y desplegar la aplicació dic 8 dic 16 9d
4 Entrega Final dic 17 ene 14 29d
4.1 Ultimas pruebas de la aplicación dic 17 dic 23 7d
4.2 Presentación dic 26 dic 30 5d
4.3 Redación memoria ene 1 ene 14 14d
5 Estudio de JEE sep 18 nov 6 50d

1.5.Productos obtenidos:

El producto final obtenido es una aplicación web realizada utilizando la arquitectura J2EE y más concreta mente con la utilización del framework Spring.

Además de la memoria y la presentación los productos obtenidos son:

●**matxurak.war:** contiene los achivos binarios, las librerias necesarias para ejecutar la aplicación y los ficheros xml de configuración.
●**database.sql:** script para la creación e inserción de datos en la base de datos.
●**matxurak:** directorio que contiene los siguientes archivos y directorios:

1. **build.xml** archivo para desplegar la aplicación con ANT.
2. **build.properties** archivo con las propiedades ANT para construir la aplicación Spring.
3. **src** directorio que contiene todas las clases java.
4. **war** directorio que contiene todos los jsp, el servlet, la hoja de estilos y la imagen utilizada en la aplicación.

2.Estructura de la aplicación:

J2EE es una plataforma de programación que tiene como objetivo ejecutar aplicaciones empresariales en arquitecturas distribuidas y de diversos niveles. Para la realización de la aplicación, voy ha utilizar el patrón MVC.

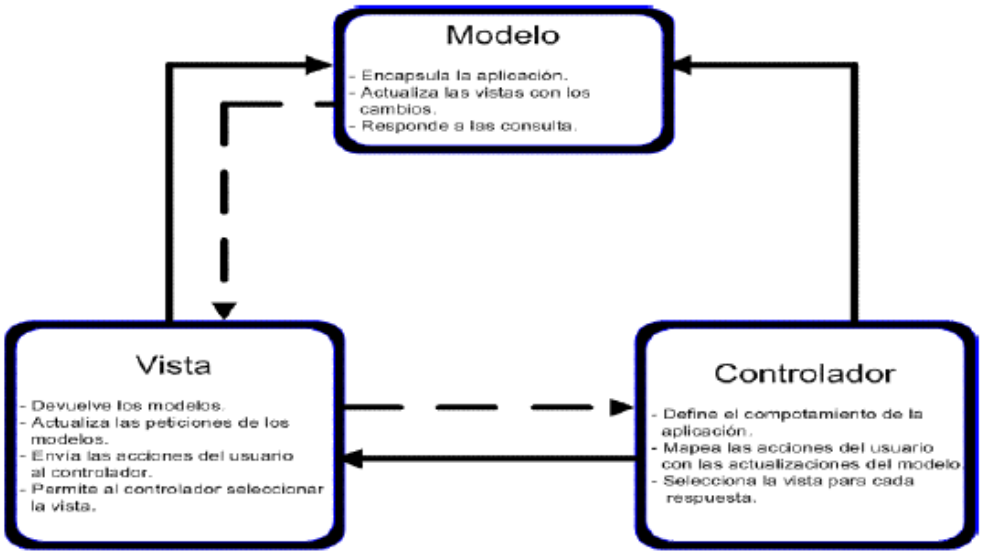
El patrón MVC es un patrón de diseño de software en el cual todo el proceso está dividido en 3 capas, estas capas son el Modelo, la Vista y el Controlador.

El Modelo incorpora la capa del dominio y persistencia, es la encargada de guardar los datos en un medio persistente (ya sea una base de datos, un archivo de texto, registro, etc.). En el modelo es donde se hace el levantamiento de todos los objetos que tu sistema debe de utilizar, es el proveedor de los recursos.

La Vista se encarga de presentar la interfaz al usuario, en nuestro caso será mediante páginas JSP.

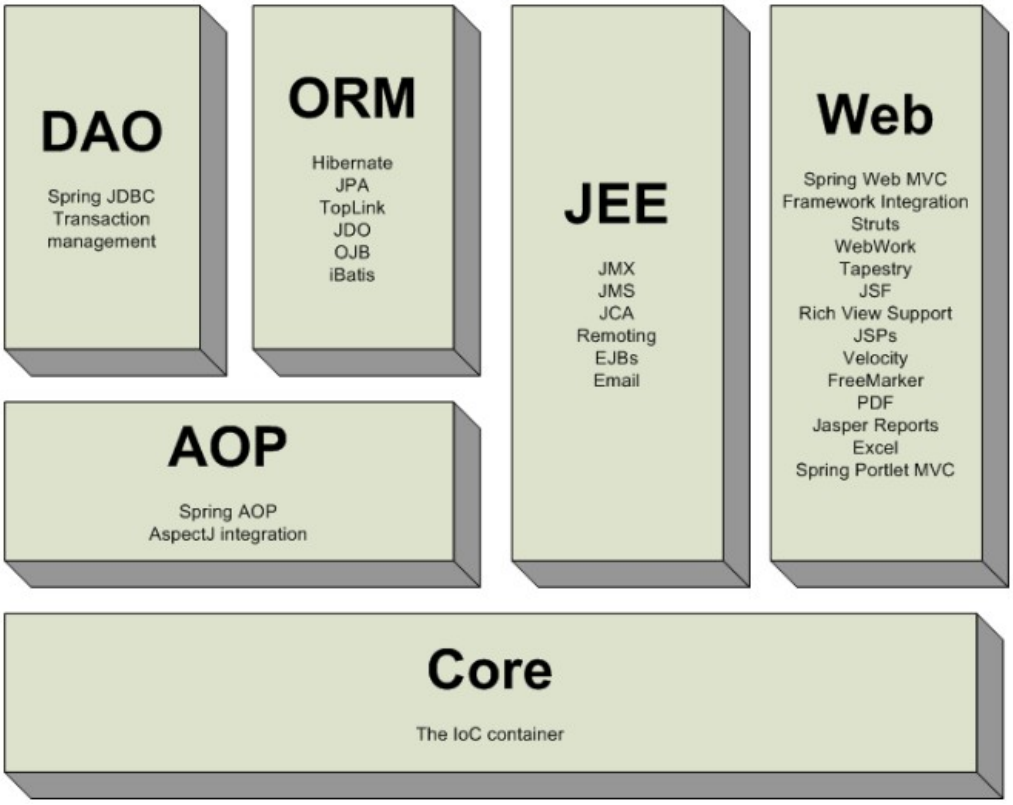
El Controlador es el que controla el flujo de información tal y como su nombre indica . Escucha los cambios en la vista y se los envía al modelo, el cual le devuelve los datos a la vista, es un ciclo donde cada acción del usuario causa que se inicie un nuevo ciclo.

La forma más sencilla de implementar este patrón es pensando en capas, como regla, los accesos a la base de datos se hacen en el modelo, la vista y el controlador no deben de saber si se usa o no una base de datos. El controlador es el que decide que vista se debe de imprimir y que información es la que se envía a dicha vista.



Para la utilización de este patrón he estudiado diferentes frameworks que lo implementan y según la información analizada los dos más utilizados son Struts y Spring.

Tras analizar mucha información he decidido utilizar Spring ya que aunque Struts es un framework muy utilizado y hay más documentación al respecto, Spring se puede integrar fácilmente con otros frameworks existentes y su utilización no impide la utilización de otros. Además es mucho más flexible. En concreto voy ha utilizar SpringMVC y SpringJDBC.



El funcionamiento del framework SpringMVC a grandes rasgos es el siguiente:

Spring tiene implementado un servlet que realiza las tareas de Front Controller, en nuestro caso será *matxura-servlet.xml,* esto significa que cada uno de los request que son realizados por el usuario, pasan a través de este servlet. El nombre que recibe este servlet es Dispatcher Servlet. Y como he indicado anteriormente, todos los request que son realizados por los distintos usuarios pasan por este componente. Los Request llegan al Dispatcher el cual tiene la responsabilidad de delegar a otro componente el procesamiento del request.

Para obtener el nombre del componente que recibirá el request , Spring utiliza lo que se denomina el Handler Mapping , el cual tiene la función de determinar cual sera el Controller que recibirá el request.

El Handler Mapping como indicamos tiene el objetivo de indicar al dispatcher cual sera el componente que debe recibir el request enviado por el usuario. Por lo cual el Dispatcher Servlet , le pregunta a uno o mas Handler Mapping cual sera el Controller que recibirá el Request.

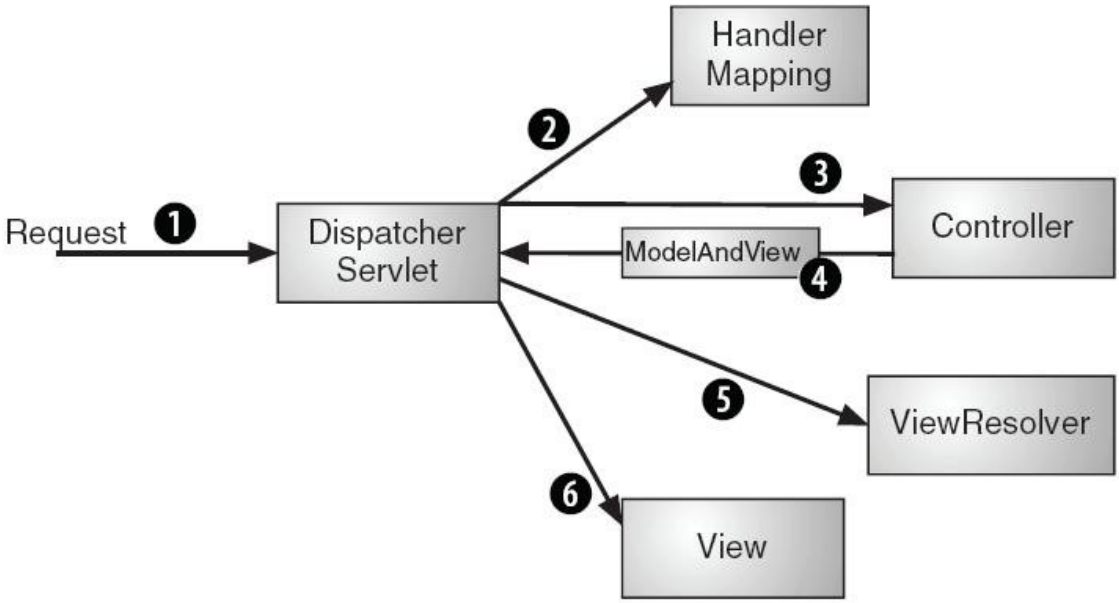
Cuando el Handler Mapping le entrega el nombre del Controller que se hará cargo del request, el Distpacher Servlet le envia el request al Controller. Para poder implementar un Controller sobre Spring es necesario que se cree una clase que herede de los Controller que han sido implementados por Spring.

El Controller recibe el Request , y se construlle un objeto que se denomina ModelAndView, este componente tiene como funcion la de :

- ●Entregar un nombre lógico a la vista que deberá realizar el despliegue del Modelo.
- ●Entregar un nombre lógico al Modelo que esta asociado a este componente.
- ●Inyectar el objeto Model el cual tiene los datos que serán desplegados en la Vista.

Después el objeto ModelAndView regresa al Dispatcher Servlet y este componente delega la responsabilidad del mapping del nombre lógico de la vista, con el componente a utilizar al ViewResolver. El ViewResolver es el encargado de realizar el mapping entre el nombre lógico de la vista y el componente.

Cuando la vista realiza el procesamiento , el dispatcher envía el request de retorno al usuario.

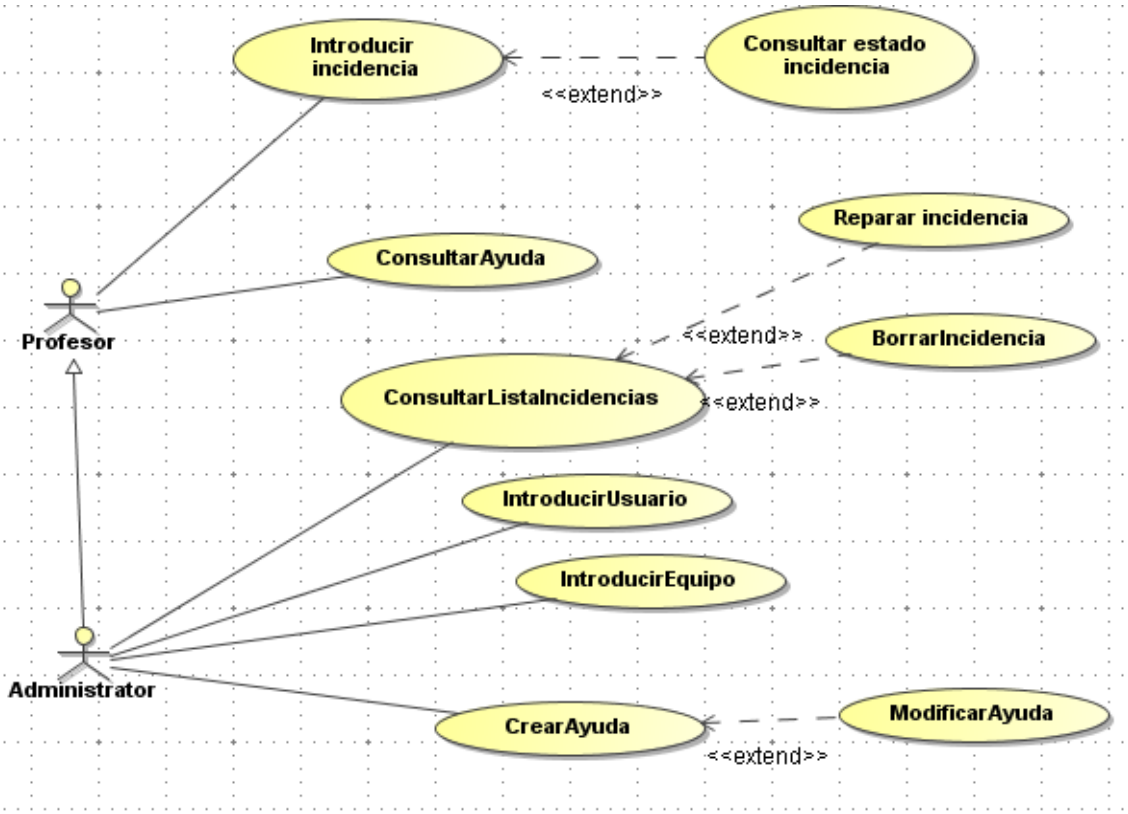


En cuanto a la persistencia de los datos, he utilizado el modulo JDBC de Spring ya que simplifica mucho el acceso a los datos preocupándose por nosotros de abrir y cerrar las conexiones con la base de datos.

3. Análisis:

Hay que tener en cuenta que la aplicación que se va a realizar, estará integrada en la intranet del centro escolar por lo que será esta la que controle el acceso a la aplicación, es decir la opción de poder introducir incidencias o controlarlas se dará en el login de acceso a la intranet por lo que no se considera ningún login especifico para acceder a la aplicación.

3.1Diagrama de Casos de Uso:



Caso de Uso	IntroducirIncidencia
Actores	Profesor y Administrador
Funcionalidad	Introducir una incidencia
Resumen	Permite introducir una incidencia de una avería de un equipo informático en el sistema.
Precondicion	El usuario tiene que tener acceso a la aplicación
Postcondicion	En el listado de incidencias habrá una incidencia más.

Caso de Uso	**ConsultarEstadoIncidencia**
Actores	Profesor y Administrador
Funcionalidad	Consultar el estado de una incidencia.
Resumen	Permite consultar el estado de una incidencia introducida anteriormente.
Precondicion	El usuario tiene que crear la incidencia
Postcondicion	Ninguna, es una simple consulta.

Caso de Uso	**ConsultarAyuda**
Actores	Profesor y Administrador
Funcionalidad	Consultar la ayuda.
Resumen	Consultar la ayuda para solucionar problemas comunes y poderlos soucionar sin introducir incidencias.
Precondicion	El usuario tiene que tener acceso a la aplicación
Postcondicion	Ninguna

Caso de Uso	**CrearAyuda**
Actores	Administrador
Funcionalidad	Crear Ayuda.
Resumen	Permite crear una ayuda para solucionar un problema que se repite con bastante frecuencia para evitar la acumulación de incidencis innecesarias.
Precondicion	El usuario tiene que ser administrador.
Postcondicion	En el listado de ayuda habrá una ayuda más.

Caso de Uso	**ModificarAyuda**
Actores	Administrador
Funcionalidad	Modificar una ayuda.
Resumen	Permite modificar una ayuda porque ha cambiado la forma de solucionarla o no está lo suficientemente bien especificada.
Precondicion	La ayuda existe.
Postcondicion	La ayuda es modificada.

Caso de Uso	**ConsultarListaIncidencias**
Actores	Administrador
Funcionalidad	Consultar las incidencias.
Resumen	Permite consultar las incidencias y su estado actual.
Precondicion	El usuario tiene que ser administrador.
Postcondicion	Ninguna solo es una consulta.

Caso de Uso	**RepararIncidencia**
Actores	Administrador
Funcionalidad	Cambiar el estado de una incidencia a reparado.
Resumen	Cambiar el estado de una incidencia a reparado una vez que se ha solucionado el problema.
Precondicion	El usuario tiene que ser administrador.
Postcondicion	Cambia el estado de la incidencia a reparado.

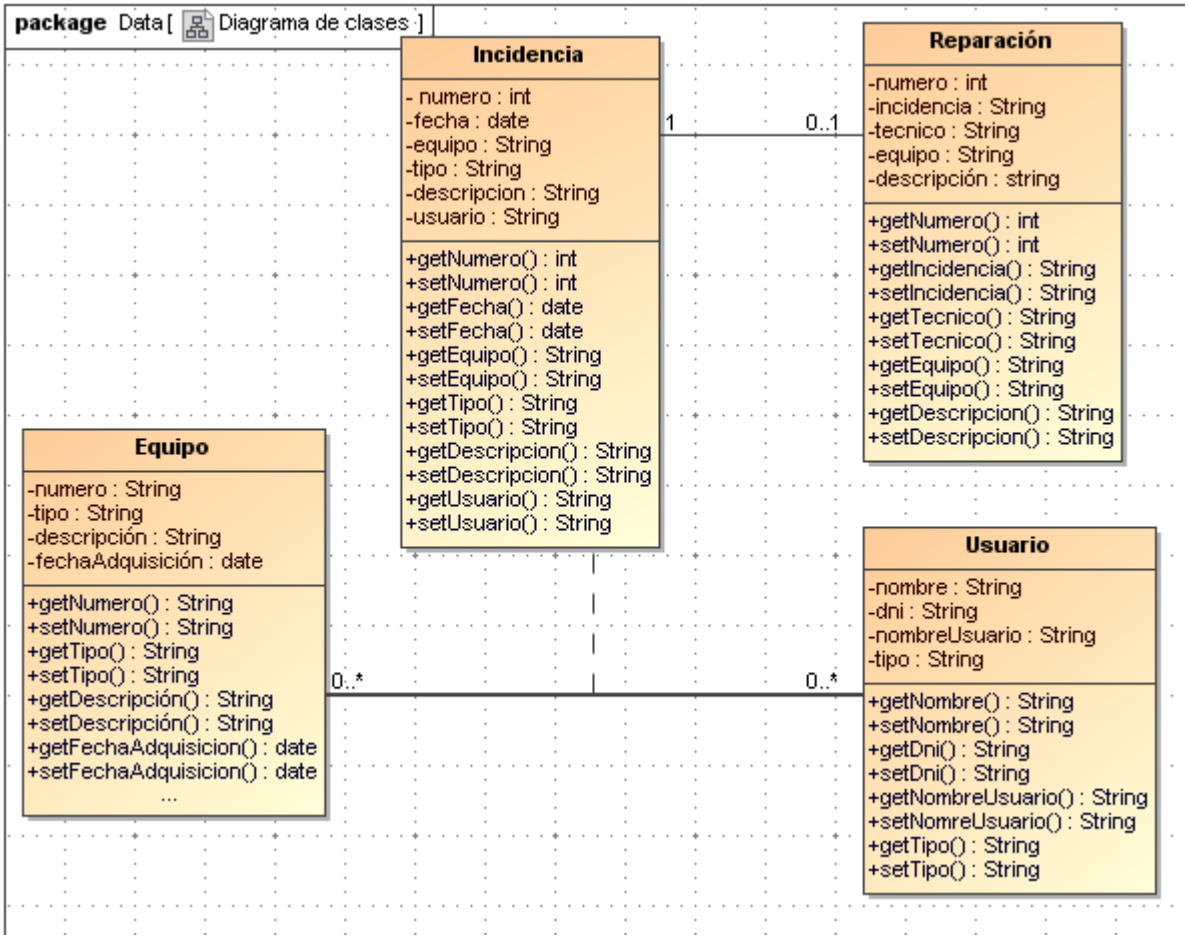
Caso de Uso	**BorrarIncidencia**
Actores	Administrador
Funcionalidad	Eliminar una incidencia.
Resumen	Eliminar una incidencia por diferentes razones (solución en ayudas, repetida...)
Precondicion	El usuario tiene que ser administrador.
Postcondicion	Habrá una incidencia menos en el listado de incidencias.

Caso de Uso	**IntroducirUsuario**
Actores	Administrador
Funcionalidad	Crea un usuario.
Resumen	Crea un nuevo usuario en el sistema.
Precondicion	El usuario tiene que ser administrador.
Postcondicion	Habrá un usuario más .

Caso de Uso	**IntroducirEquipo**
Actores	Administrador
Funcionalidad	Introduce un nuevo equipo.
Resumen	Introducir un equipo nuevo en la base de datos.
Precondicion	El usuario tiene que ser administrador.
Postcondicion	Habrá un equipo más en la base de datos.

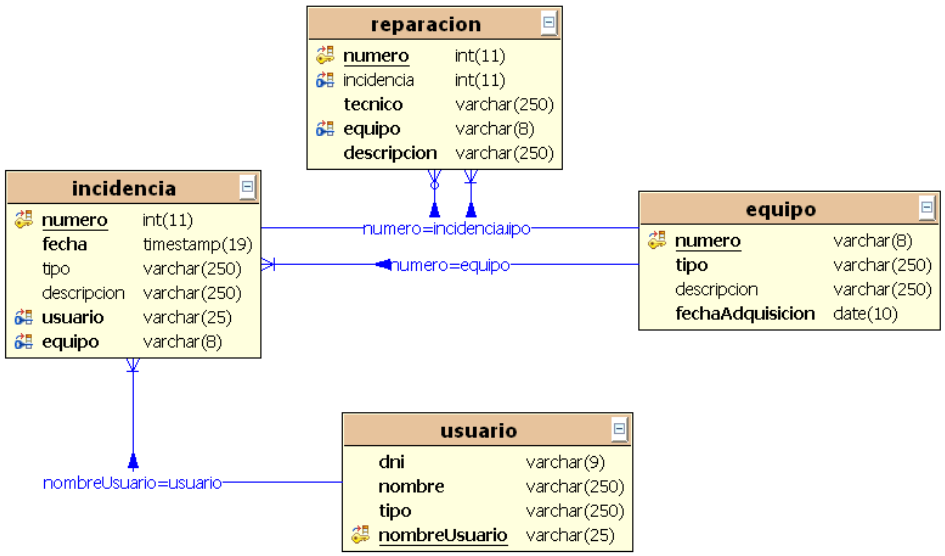
4. Diseño:

4.1.Diagrama de clases:



4.2.Diagrama de Base De Datos:

La base de datos se distribuirá en cuatro tablas. En la tabla ***Usuario*** tendremos los datos de los usuarios y el *nombreUsuario* que será la clave primaria. En la tabla ***Equipo*** guardaremos los datos de los equipos donde numero será la clave primaria. En la tabla ***Incidencia*** guardaremos las incidencias que introduzcan los usuarios y *numero* será la clave primaria además tendremos dos claves foráneas, *usuario* que corresponde a *nombreUsuario* de la tabla ***Usuario*** y *equipo* que corresponde a la clave primaria de la tabla ***Equipo.*** Por último en la tabla ***Reparación*** se guardarán todas las reparaciones de incidencias que se han realizado y tendremos como clave primaria *numero* y como claves foráneas *incidencia* y *equipo* de las tablas **Incidencia** y **Equipo** respectivamente.



Incidencia:

	tipo	descripción
PK numero	integer	Identificador de la incidencia.
fecha	integer	Fecha de la incidencia.
tipo	string	Tipo de incidencia(Software, Hardware...)
descripción	string	Descripción del problema a solucionar.
FK usuario	string	Identificador del usuario.
FK equipo	string	Identificador del equipo.

Reparación:

	tipo	**descripción**
PK numero	integer	Identificador de la reparación.
FK incidencia	integer	Identificador de la incidencia.
técnico	string	Nombre del técnico que ha efectuado la reparación.
descripción	string	Descripción de la solución tomada para la reparación de la incidencia.
FK equipo	string	Identificador del equipo.

Equipo:

	tipo	**descripción**
PK numero	string	Identificador del equipo.
Fecha Adquisición	date	Fecha de adquisición del equipo(importante para la garantía)
tipo	string	Si es un PC, impresora., monitor...
descripción	string	Descripción técnica del equipo(procesador, RAM, pixels...)

Usuario:

	tipo	**descripción**
PK nombreUsuario	string	Identificador del usuario
nombre	string	Nombre de pila y apellidos
tipo	string	Si es profesor, alumno, administrativo...
dni	string	D.N.I del usuario.

PK: Clave primaria.
FK: Clave foránea.

4.3.Herramientas utilizadas:

Para el desarrollo de la aplicación se ha primado la utilización de herramientas gratuitas ya que uno de los objetivos a la hora de realizar mi proyecto era la utilización de estas herramientas además de populares para no tener problemas a la hora de encontrar documentación. Las herramientas que voy a utilizar son las siguientes:

- Java jdk1.6.0_10
- Eclipse Europa como entorno de desarrollo.
- Apache Tomcat 5.5 como servidor ya que se trata de un servidor gratuito donde podré desplegar la aplicación completa.
- MySql como sistema gestor de BBDD. Al igual que las demás herramientas es gratuita y muy popular.
- Apache Ant para desplegar la aplicación en el servidor. Facilita mucho la creación y despliegue del “war” en el servidor.
- Spring framework.
- MagicDrawUML

5.Patrones de diseño

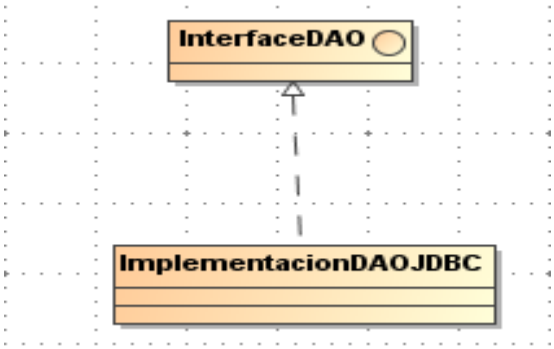
5.1.Data Acces Object DAO:

He utilizado el patrón DAO ya que este oculta completamente los detalles de implementación de la fuente de datos a los clientes. El interface del DAO no cambia cuando cambia la implementación de la fuente de datos, este patrón permite adaptarse a diferentes esquemas de almacenamiento sin que esto afecte a sus clientes o componentes de negocio. Esencialmente, el DAO actúa como un adaptador entre el componente y la fuente de datos, y por lo tanto no tendremos que cambiar estos interfaces aunque cambiemos el SGBD.
En mi caso he utilizado las siguientes interfaces DAO

EquipoManagerDao
IncidenciaManagerDao
ReparacionManagerDao
UsuarioManagerDao

Y sus implementaciónes utilizando JDBC

EquipoManagerDaoJdbc
IncidenciaManagerDaoJdbc
ReparacionManagerDaoJdbc
UsuarioManagerDaoJdbc

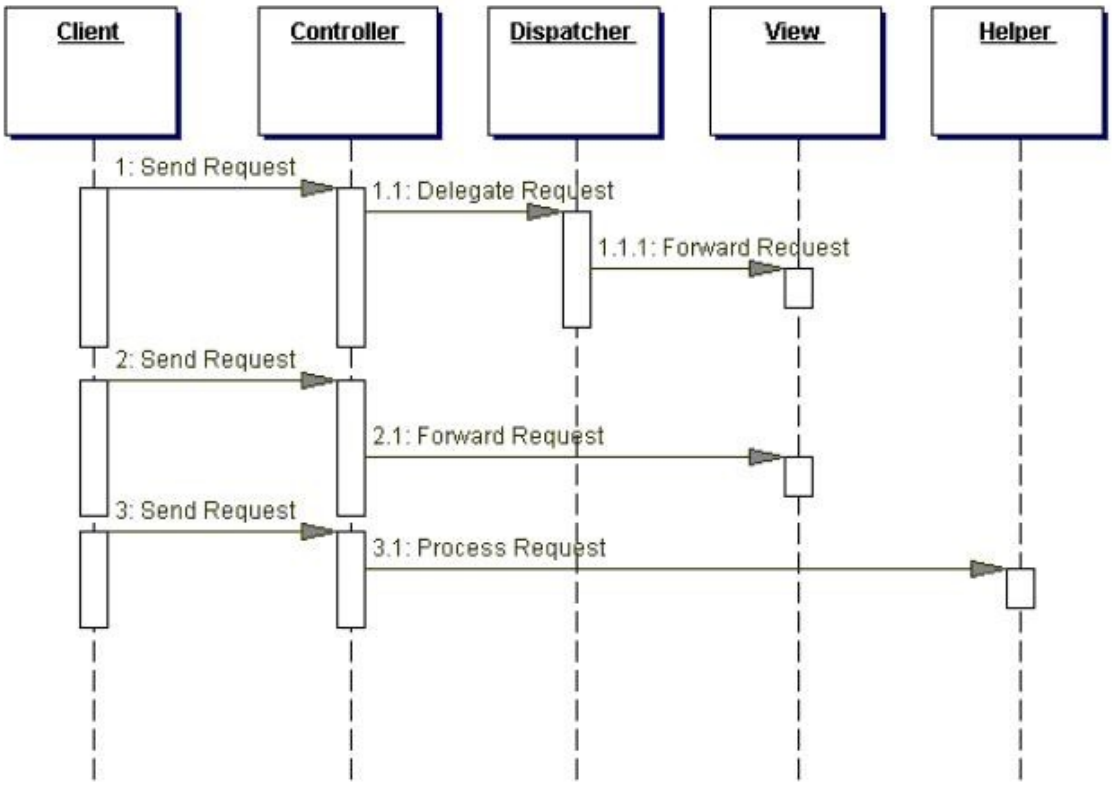


5.2.Front Controller:

Este patrón obliga a que todas las peticiones hechas a nuestra aplicación pasen por un servlet Controlador.

- El controlador proporciona un punto de entrada único que controla y gestiona las peticiones Web realizadas por los clientes.
- Teniendo este único punto de entrada se evita tener que repetir la misma lógica de control en todos los .jsp.
- Normalmente se utiliza junto con un Dispatcher que es el responsable de redirigir el flujo de ejecución hacia el jsp adecuado.

Diagrama de secuencia del patrón Front-Controller



6.**Implementación:**

La implementación del proyecto, la he realizado en diferentes fases, y para ello el punto de partida ha sido el tutorial "*Developing a Spring Framework MVC application step-by-step*" que se puede consultar en la dirección:
http://static.springframework.org/spring/articles/2005/MVC-step-by-step/Spring-MVC-step-by-step-Part-1.html

A partir de este ejemplo he ido modificándolo para al final conseguir desarrollar mi propia aplicación. Lo primero, ha sido crear la base de datos "averias" en mysql. Después implementar toda la estructura necesaria para la introducción de un simple dato desde un formulario en una vista, y una vez conseguido esto ir completando el resto de funcionalidades.

Han sido muchas las dificultades que he encontrado. La más importante ha sido entender el funcionamiento de cada una de las partes de la aplicación.

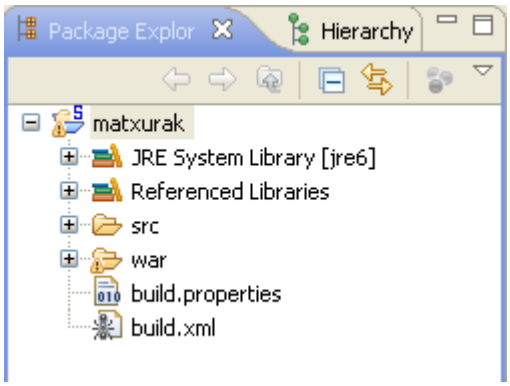
En cuanto a la persistencia de los datos, tras analizar diferentes posibilidades decidí utilizar el JDBC de Spring.

Tampoco conocía las etiquetas JSTL y tras entender su funcionamiento me ha parecido una librería muy sencilla de utilizar y muy práctica para el acceso a los datos.

En el acceso a la aplicación, no se ha diferenciado entre tipos de usuario, se ha desarrollado al completo, es decir se supone que el que accede a la aplicación es administrador, ya que los profesores de momento sólo tendrán acceso a la pantalla de introducción de incidencias.

6.1Estructura de la aplicación:

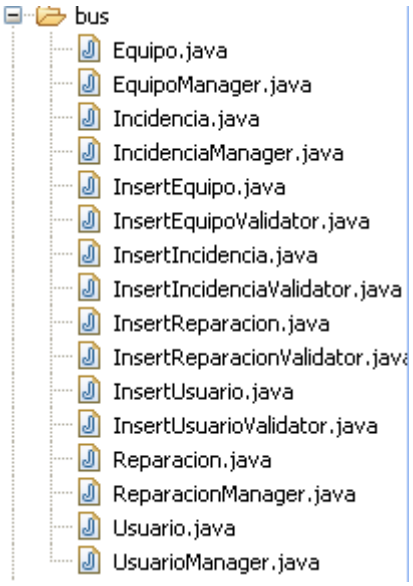
La aplicación se ha estructurado bajo las especificaciones de J2EE, y en concreto las de Spring. La estructura de las carpetas es la siguiente:



Dentro de la carpeta src están todas las clases java de la aplicación, y éstas están divididas en los siguientes paquetes:

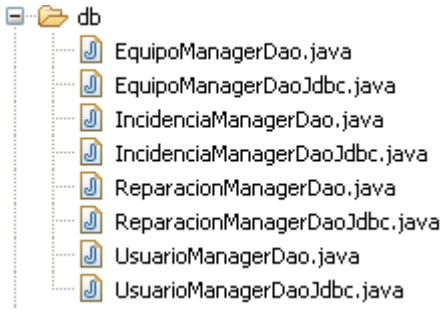
-bus:

Contiene los POJO que conforman la aplicación, los VALIDATOR que se encargan de que los datos introducidos en los formularios sean correctos y los MANAGER que gestionan toda la lógica de negocio.



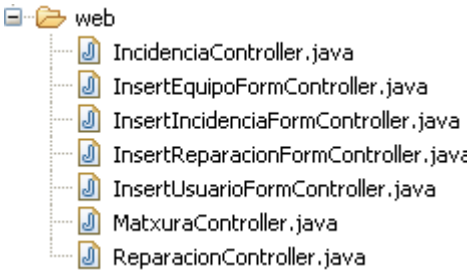
-db:

En este paquete están las clases e interfaces que tienen relación con la persistencia de los datos. Las interfaces Dao y su implementación DaoJdbc..



-web:

Está formada por los CONTROLLER de la aplicación, tanto los que simplemente nos muestran los datos como los FormController que interactúan con los formularios.



En cuanto a la carpeta war, ésta a su vez contiene dos carpetas:

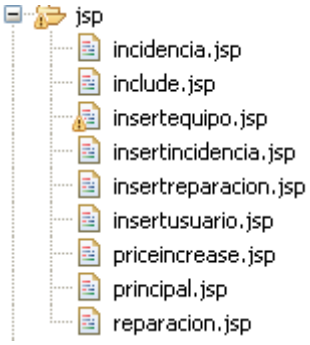
-war:

La carpeta web contiene la hoja de estilos y el logo del centro de enseñanza:

-web:

En cuanto a la carpeta WEB_INF, contiene una carpeta classes y otra jsp. En esta última, están las vistas jsp, excepto index.jsp, que queda fuera del directorio para que no se pueda acceder directamente a las páginas jsp escribiendo la dirección en el navegador. Será index.jsp la que nos redireccionará a principal.jsp al comenzar la aplicación.

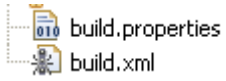
jsp:



En la carpeta classes, además de los paquetes con las clases java, tenemos el archivo log4j.properties que permite introducir los logs en el servidor. Es muy útil para la captura de errores en la fase de implementación. Además, he creado messages.properties para definir los mensajes de error y las cabeceras que se mostrarán en las vistas. De esta forma, si se quiere cambiar algún mensaje haciéndolo aquí conseguimos cambiarlo en todas las vistas.

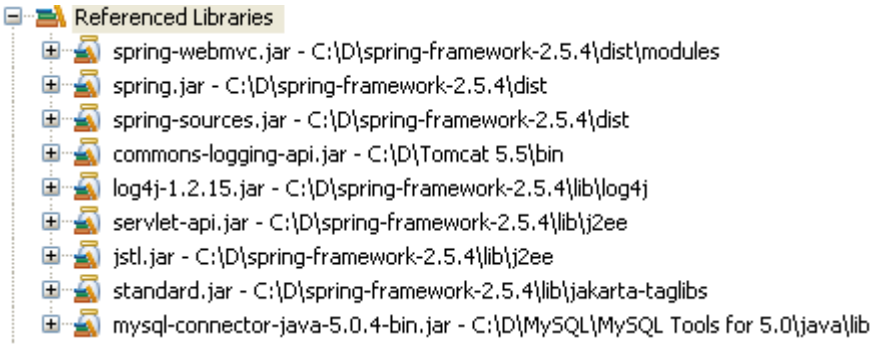
Además de estos ficheros tenemos el dispatcher de Spring matxura-servlet.xml, que es donde se definen todos los beans. El archivo spring.tld que es la jsp tag library y web.xml propio de la arquitectura J2EE.

Por último, en el directorio raíz tenemos los ficheros bu*ild.properties* y *build.xml*
El ".xml" se utiliza para hacer el *build, deploy...* con *ant,* y en el ".properties"tenemos las propiedades del *Apache Tomcat* y del SGBD *mysql*.



6.2.Librerías utilizadas:

Además de las clases java se han utilizado las siguientes librerías para construir la aplicación:



se pueden descargar de:

http://www.springsource.org/download

http://dev.mysql.com/downloads/connector/j/5.0.html

http://tomcat.apache.org/download-55.cgi

7.Instalación de la aplicación:

Para la puesta en marcha de la aplicación necesitamos descargar varios programas:

IDE Eclipse Europa 3.3
http://www.eclipse.org/downloads/

Servidor Apache Tomcat 5.5
http://tomcat.apache.org/download-55.cgi

Sistema gestor de datos Mysql 5.0
http://dev.mysql.com/downloads/

Conector de Java y Mysql
mysql-connector-java-5.0.4-bin.jar
http://dev.mysql.com/downloads/connector/j/5.0.html

Para utilizar *matxura.war* directamente habrá que instalar el Apache Tomcat en la siguiente ubicación:
C:/Archivos de programa/Apache Software Foundation/Tomcat 5.5
Y cambiar en el fichero:
C:\Archivos de programa\Apache Software Foundation\Tomcat 5.5\conf\tomcat-users.xml
la siguiente linea: <user username="admin" password="admin" roles="admin,manager"/>
Ahora solo nos queda pegar *matxura.war* en C:\Archivos de programa\Apache Software Foundation\Tomcat 5.5\webapps
Una vez hecho esto, ejecutamos el script *database.sql* en *mysql.*

Abrimos el navegador y escribimos la siguiente dirección:
http://localhost:8080/matxura
Y nos aparecerá la pantalla principal

Para desplegar la aplicación desde eclipse con *ant*, una vez instalados todos los programas tenemos que configurar el fichero *build.properties* para el *Apache Tomcat* y *Mysql*.

```
appserver.home=”dirección donde hemos instalado el Apache Tomcat”
deploy.path=${appserver.home}/webapps

tomcat.manager.username=”username”

tomcat.manager.password=”contraseña”
tomcat.manager.url=http://localhost:8080/manager

db.driver=com.mysql.jdbc.Driver
db.url=jdbc:mysql://localhost/averias
db.user=”username”
db.pw=”contraseña”
```

También tendremos que comprobar que user y password en el bean datasource de nuestro servlet *matxura-servlet.xml* son validos para la instalación de *mysql*. Si no habrá que modificarlos.

```
<bean id="dataSource"
         class="org.springframework.jdbc.datasource.DriverManagerDataSource">
         <property name="driverClassName">
             <value>com.mysql.jdbc.Driver</value>
         </property>
         <property name="url">
             <value>jdbc:mysql://localhost/averias</value>
         </property>
         <property name="username">
             <value>root</value>
         </property>
         <property name="password">
             <value>root</value>
         </property>
     </bean>
```

Una vez hecho esto, ejecutamos el script *database.sql* en mysql.

Abrimos el navegador y escribimos la siguiente dirección: http://localhost:8080/matxura

Y estaremos en la página principal de la aplicación mostrada en la página anterior.

8.Navegación:

La navegación por la aplicación es muy sencilla, simplemente a partir de la página de inicio clickamos en la opción deseada y accedemos a la siguiente pantalla. Por ejemplo podremos introducir un usuario:

Si dejamos alguno de los datos sin introducir el *validator* nos pedirá que introduzcamos los datos que nos faltan.

Para la introducción de un equipo el funcionamiento es el mismo que para la un usuario. Para introducir una incidencia clikaremos en la opción de introducir incidencia y accederemos a dicha opción.

Aquí tenemos la opción de introducir una incidencia pero para ello tanto el equipo como el usuario tienen que estar dados de alta. Además solo podemos elegir entre los tipos de avería predefinidos para poder hacer una mejor gestión de estos.
Al igual que en las anteriores pantallas todos los campos tienen que contener datos para que sean introducidas en el sistema.
Una vez introducida la incidencia el sistema nos direcciona al listado de incidencias para que veamos como ha quedado registrada.

A partir de esta pantalla podemos volver a la pantalla inicial, podemos introducir una nueva incidencia o reparar alguna de las que tenemos en el listado. A esta última opción también podemos acceder desde la pantalla principal.
Por ultimo podemos acceder a los listados de reparaciones bien desde la pantalla principal o desde la pantalla que nos muestra el listado de incidencias.

AVERIAS

APLICATIVO PARA LA GESTION DE AVERIAS

Listado de Reparaciónes.

NUMERO DE REPARACION	NUMERO DE INCIDENCIA	TECNICO	DESCRIPCION DE LA REPARACION
1	2	Pedro	No estava bien enchufado a la red.
2	1	bngfh	ghghgfdbhgfdbfdd
3	3	fhgbfdhb	bhnfhfdbbfgc

Introducir Reparacion

Inicio

9.Bibliografia

Recursos y tutoriales consultados en la web:

http://static.springframework.org/docs/Spring-MVC-step-by-step/
http://www.jroller.com/afuentes/entry/mvc_en_spring
http://abunodiscentomnes.wordpress.com/2007/07/18/una-pequena-aplicacion-en-spring-paso-a-paso-i/
http://www.springhispano.org/?q=node/21
http://maestric.com/en/doc/java/spring
http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=springMVCdesdeCero
http://java.sun.com/products/jsp/tutorial/TagLibraries17.html

Libros:

●Spring in Action Second Edition.(Craig Walls) *ISBN: 1-933988-13-4*
●Manual de referencia J2EE (Jim Keogh) McGrawHill. *ISBN: 84-481-3980-1*

10.Conclusión

Para concluir el trabajo realizado durante este semestre, quiero mencionar algunos aspectos de cada una de las fases del proyecto:

●Plan de Trabajo: Ha sido la fase más sencilla ya que tenía bastante claro cuál era el proyecto que quería realizar y los objetivos. Además conocía el tiempo del que disponía para el proyecto y por lo tanto hacer la planificación también me resulto sencillo.

●Pec2: Esta fase me ha resultado bastante complicada ya que mi única experiencia con el análisis ha sido a través de la asignatura "Ingeniería del Software". Además no tenía muchos conocimientos de la tecnología J2EE, y por ello los diagramas han sido muy simples. Para completar esta fase he consultado mucha documentación y me he ido formando sobre el funcionamiento del *framework Spring*,y los patrones de diseño.

●Pec3: Ha sido la fase más interesante, a partir del tutorial Spring-MVC-step-by-step he ido modificando las partes hasta conseguir desarrollar mi propia aplicación. Ha sido aquí donde de verdad he entendido el funcionamiento de *Spring,* además me he iniciado en la utilización de la *tag library*.
Algunas partes de la aplicación como el acceso a una incidencia en concreto, o el acceso a todas las incidencias de un equipo han quedado sin implementar, pero el objetivo prioritario que era la introducción y el tratamiento de las incidencias está resuelto. Además se ha añadido la introducción de usuarios y equipos que en un principio no estaba prevista.

Para terminar decir que aunque la carga de trabajo ha sido importante la experiencia ha sido muy enriquecedora ya que mis conocimientos sobre la tecnología J2EE y en concreto del *framework Spring* han mejorado significativamente.