Gestión de Conservación

Desarrollo para Internet con Tecnología Java

Manuel Fernández López
ETIS

Javier Ferro García

18 de Junio de 2004

RESUMEN

El TFC se basa en implementar un programa de gestión de trabajadores que se aplicará en una empresa de pequeñas dimensiones, la cual está dedicada al mantenimiento de aparatos elevadores. Con este programa de gestión, lo que pretendo conseguir es lo siguiente:

- Poder realizar inserciones de nuevos trabajadores en plantilla o modificar los datos ya existentes, asimismo poder consultar datos referentes a los trabajadores registrados. Personal encargado: administración.
- Realizar nuevas inserciones de vehículos y consultar aspectos sobre el parque de vehículos del centro de trabajo, de esta manera, se podrá determinar el número multas, partes de accidente, antigüedad del vehículo, etc. de un determinado trabajador. Personal encargado: encargado conservación.
- Insertar facturas correspondientes a los teléfonos móviles, consultar sobre aquellas personas que se han excedido en gasto telefónico (30 €). Personal encargado: encargado conservación.
- Podrán realizar las consultas en todas las tablas antes mencionadas el gestor conservación y el encargado conservación.

INDICE

INTRODUCCIÓN

El tema del TFC que he escogido se basa en implementar un sistema de gestión para aplicarlo en una empresa pequeña, dedicada al mantenimiento de ascensores.

Lo que me ha llevado ha escoger este tema, ha sido el hecho de desarrollar mi actividad profesional en una empresa de características similares a la que presento en el TFC. En mi puesto de trabajo tengo una serie de responsabilidades sobre los empleados del centro, he de hacer un seguimiento de estas personas, así como del gasto global de la empresa, de los vehículos de que disponemos y de los percances que éstos puedan sufrir, de los teléfonos móviles que tienen cada uno de los empleados, entre otras funciones.

Para realizar este tipo de seguimiento, dispongo de una hoja de cálculo, de la que me sirvo para posteriormente aplicar los criterios que se seguirán, por ejemplo a la hora del pago de multas, del control del gasto de los teléfonos móviles de los empleados. Otras de las funciones que realizo son la asignación de ropa y material necesario para los trabajadores, gestión de los gastos varios realizados por éstos en diferentes proveedores (ferreterías industriales, suministradoras de repuestos, etc.), auditorias, etc.

El seguimiento de los empleados los realizamos mediante la asignación de un código interno a cada empleado, de esta forma, por ejemplo podemos saber cotejando los albaranes con las facturas oficiales enviadas por los proveedores, que persona ha realizado un determinado gasto.

Estos son los motivos que me han llevado a realizar un sistema informático (sencillo con las funciones más habituales que realizo en el trabajo) para poder disponer de un modo de agrupación de todos los datos que necesito habitualmente en un solo entorno, y así de este modo desarrollar de manera más eficiente mi tarea. Pro otro lado, también me ha atraído desde el principio, el hecho de poder realizar un proyecto orientado a Internet con las posibilidades que esto me ofrece.

JUSTIFICACIÓN Y OBTENCION DE REQUERIMIENTOS

A lo largo del TFC me voy a basar en implementar un programa de gestión de de trabajadores que se aplicará en una empresa de pequeñas dimensiones, la cual está dedicada al mantenimiento de aparatos elevadores.

A continuación paso a detallar las características de la empresa sobre la que voy a trabajar, presentar a los actores que formarán parte de este proyecto y explicar cuales serán sus funciones.

Información Inicial (para describir el dominio).

Se trata de la gestión de varios equipos de trabajo encargados del mantenimiento de aparatos elevadores. Cada trabajador pertenece a un equipo y cada trabajador dispone de un teléfono corporativo y vehículo para desarrollar sus actividades dentro de la organización.

Información Inicial (para describir los guiones).

La empresa se compone de 30 trabajadores, con una capacidad máxima del centro de trabajo para 50 trabajadores.

Guión del Personal de administración.

El personal de administración entrará en el sistema mediante un nombre y clave para poder realizar sus tareas.

Una vez que se ha entrado al sistema, el personal de administración podrá introducir los datos personales de cada nuevo empleado que se incorpora en plantilla (nombre, apellidos, DNI, teléfono particular, dirección y población), le asignará un código interno propio de la empresa para su identificación y el equipo de trabajo al cual va a pertenecer; una vez que se ha validado la información el sistema informará de la nueva incorporación del usuario.

Además de estas tareas el personal de administración también podrá hacer consultas sobre los empleados del centro de trabajo, modificar datos de los empleados (cambio de dirección, número de teléfono, etc.) y borrar empleados (fin de contratos, jubilación, etc.).

Guión del Encargado de mantenimiento.

El encargado de mantenimiento entrará en el sistema mediante un nombre y clave para poder realizar sus tareas.

Una vez dentro del sistema el encargado de mantenimiento asignará a cada trabajador un vehículo de empresa y un teléfono móvil.

De cada vehículo se registrarán sus datos (matricula, año de matriculación, marca y modelo), partes de accidente y multas que pueda tener el empleado que conduce dicho vehículo. También podrá realizar consultas y así conocer que trabajador ha tenido más siniestros en los últimos meses (3, 6, 9 y 12 meses), modificar (asignar un determinado vehículo a otro

trabajador (caso excepcional)) y borrar (cuando un vehículo determinado queda fuera de servicio).
En cuanto a los teléfonos móviles, se registrarán los datos del teléfono (número interno corporativo), la fecha de emisión de la factura y su correspondiente importe, para así poder realizar un seguimiento de gastos de teléfono para cada empleado. También podrá realizar consultas para saber que personas han gastado más teléfono en los últimos meses (3, 6, 9 y 12 meses), modificar (asignar un teléfono a otro trabajador por diversos motivos (baja en la empresa, etc.)) y borrar (será un caso excepcional si roban el teléfono).

Además, el encargado podrá realizar todas las funciones del personal de administración que se ha descrito anteriormente.

Guión del Invitado

El invitado no necesita introducir nombre y clave para acceder al sistema. Solamente podrá realizar consultas sobre los trabajadores (número de empleados que dispone este centro de trabajo, y otros datos), en ningún momento podrá introducir nuevos datos, modificar o borrar.

OBJETIVOS

Al plantear este sistema de gestión de personal, los objetivos que me he propuesto conseguir son los siguientes:

- Desarrollar un entorno usable para las diferentes funciones que ha de realizar, y a su vez que sea funcional (que a los usuarios que utilicen este entorno, les sea sencillo el acceder a las bases de datos, etc.), sin que sea necesario tener que acceder a la base de datos de manera directa, sino que sea el mismo sistema el que realice estas funciones.
- Realizar un entorno portable y actualizable en el futuro según nuestras necesidades sin que ello sea un trastorno a la hora de poder actualizar el software.
- Todos los usuarios del centro de trabajo han de poder acceder a los datos (Negocio) de manera simple y efectiva

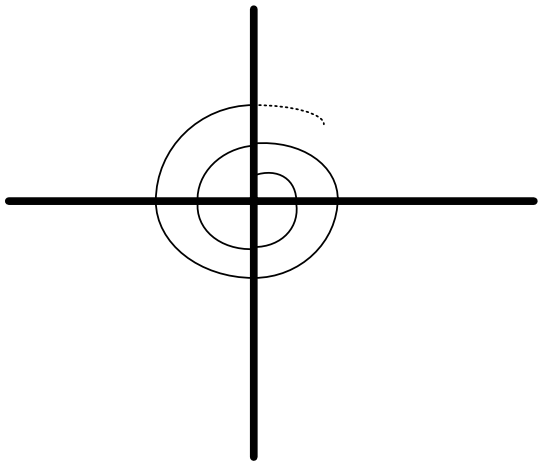
Estos son los motivos por los que he decidido ha realizar este proyecto mediante el modelo de programación que nos define el J2EE, ya que mi idea es separar la aplicación en diferentes capas.

ENFOQUE Y MÉTODO A SEGUIR.

Después de determinar los objetivos que se pretenden con este proyecto, describo a continuación el método a seguir y el ciclo de vida a desarrollar.

Para poder desarrollar este proyecto he utilizado el denominado “ciclo de vida mediante prototipo”, es decir, una vez determinado el estudio y definición del problema, se ha construido un software provisional para poder ver como sería el contenido o apariencia de los resultados de las funciones del futuro software, a continuación, se presentará al usuario final (mi paciente novia) y éste nos dirá cuales son las deficiencias. A partir de esto, se volverá a desarrollar el sistema, se volverá a presentar al usuario, éste nos indicará las funciones necesarias que no están implementadas, y así sucesivamente hasta obtener la versión definitiva.

El desarrollo de todas las versiones del futuro sistema informático hasta llegar al modelo definitivo, se realizan siguiendo el ciclo de vida clásico. La representación esquemática del modelo sería:



PLANIFICACIÓN DEL PROYECTO.

Tareas a realizar.

Las tareas que se derivan de la realización de este producto son las siguientes:

1. Búsqueda de información y Obtención de requerimientos.
    - 1.1 Información sobre los distintos productos a emplear (MySQL, TOMCAT, J2EE, Ant, Structs y EJB).
    - 1.2 Objetivos del TFC.
    - 1.3 Obtención de requerimientos (primera aproximación ciclo de vida).
    - 1.4 Descripción textual de los casos de uso.
2. Análisis del sistema.
    - 2.1 Identificación de las clases de entidades.
    - 2.2 Diagrama estático.
    - 2.3 Diagrama de secuencias.
    - 2b Representación grafica (casos de uso más representativos).
3. Implementación de métodos y programas.
    - 3.1 Implementación Base de datos.
        - 3.1.1. Diseño lógico.
        - 3.1.2. Entidades y atributos.
        - 3.1.3. Tablas obtenidas.
    - 3.2 Configuración servidor Tomcat.
    - 3.3 Conectividad servidor con base de datos.
    - 3.4 Configuración Ant.
    - 3.5 Configuración seguridad.
    - 3.6 Creación de las clases Java y páginas JSP que componen el Proyecto.
4. Redacción memoria y presentación de diapositivas.
    - 5.1 Memoria, estado de la ciencia.
    - 5.2 Memoria, ciclo de vida del proyecto.
    - 5.3 Memoria, manual de instalación del producto.
    - 5.4 Memoria, apéndices con el código fuente.
    - 5.5 Presentación, diapositivas explicativas del TFC.

Planificación Temporal.

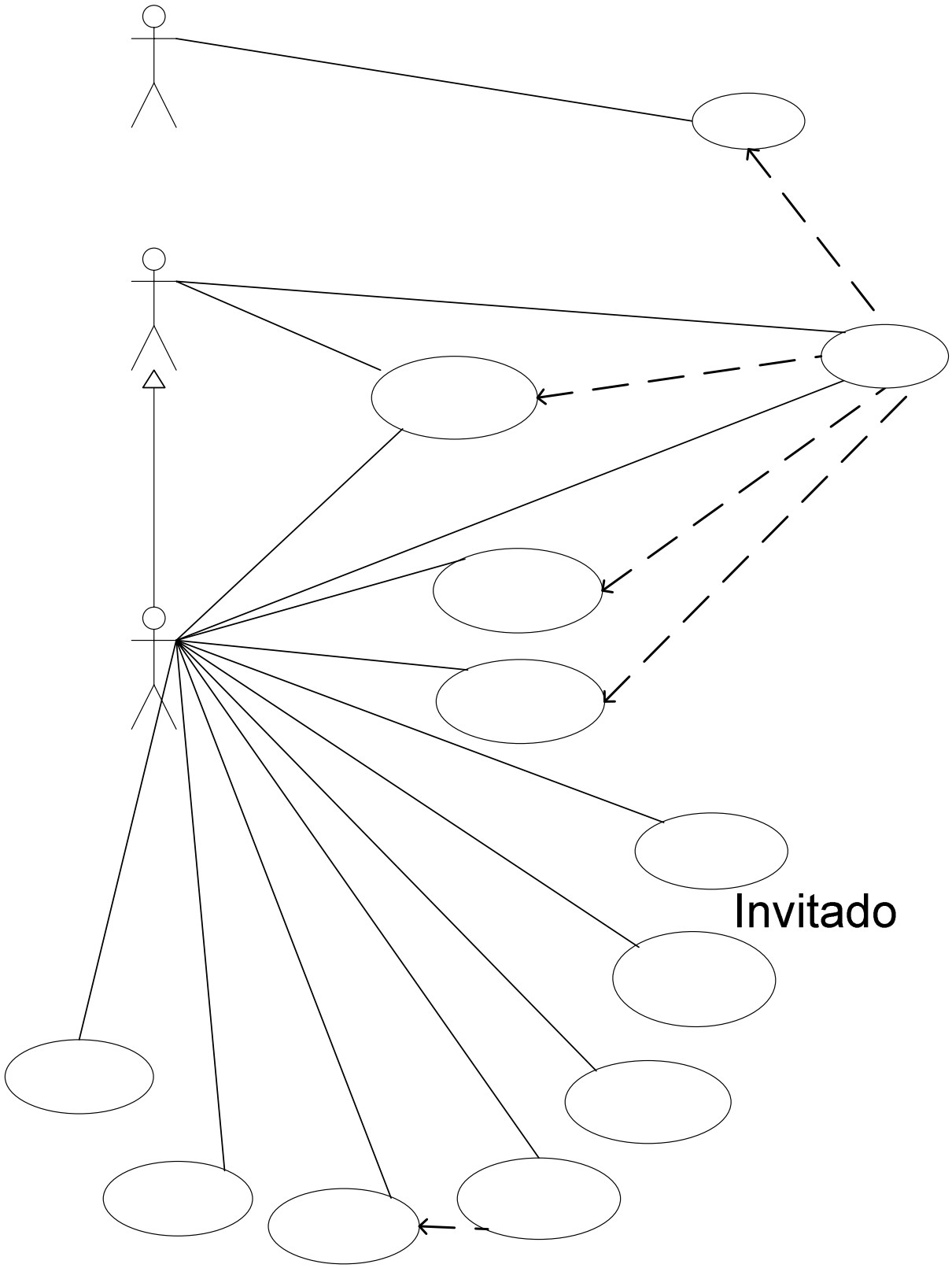
La planificación temporal de tareas contempla todos los trabajos a realizar en el TFC distribuidos temporalmente durante el cuatrimestre.

Semana	Fechas	Actividad	Evento
1	23/02/2004 29/02/2004		**28/02/2004** Encuentro Inicial.
2	01/03/2004 07/03/2004	Definir el proyecto a realizar (Propuestas). Realizar la planificación.	
3	08/03/2004 14/03/2004	Búsqueda de información y obtención de requerimientos. **T1.1** (Productos), **T1.2** (Objetivos).	**08/02/2004** Envío Planificación (PAC1).
4	15/03/2004 21/03/2004	**T1.3 (**Obtención de requerimientos). **T5.1** **(**Redacción de la memoria, estado de la ciencia.)	
5	22/03/2004 28/03/2004	**T1.3 (**Obtención de requerimientos). **T5.1** **(**Redacción de la memoria, estado de la ciencia.)	
6	29/03/2004 04/04/2004	**T1.4** (Descripción textual de los casos de uso). **T5.2** Redacción de la memoria, ciclo de vida a emplear en este proyecto, definición y enfoque.	
7	05/04/2004 11/04/2004	**T2.1 (**Identificación de las clases de entidades). **T2.2** (Diagrama estático).	
8	12/04/2004 18/04/2004	**T2.3 (**Diagrama de secuencias). **T2.4** **(**Representación gráfica de los casos de uso).	**13/04/2004** Envío Análisis del Proyecto PAC2.
9	19/04/2004 25/04/2004	**Revisión y estudio de las etapas Obtención de requerimientos y Análisis.**	
10	26/04/2004 02/05/2004	**T3.1** **(**Implementación Base de datos: Diseño lógico). **T3.1** **(**Implementación Base de datos: Entidades y atributos).	
11	03/05/2004 09/05/2004	**T3.1** **(**Implementación Base de datos: Obtención de tablas). **T3.2** (Configuración Servidor Tomcat).	
12	10/05/2004 16/05/2004	**T3.3** (Conectividad servidor con base de datos). **T3.4** (Configuración Ant). **T3.6** **(**Creación de las clases Java y páginas JSP que componen el Proyecto de manera incremental).	
13	17/05/2004 23/05/2004	**T3.6** **(**Creación de las clases Java y páginas JSP que componen el Proyecto de manera incremental)	**17/05/2004** Envío Diseño del Proyecto PAC3.
14	24/05/2004 30/05/2004	**T3.5** (Configuración seguridad: seguridad SSL para usuario invitado) **T3.6** **(**Creación de las clases Java y páginas JSP que componen el Proyecto de manera incremental).	
15	31/05/2004 06/06/2004	**T3.5** (Configuración seguridad: seguridad acceso a entorno Administrador (mediante Filtrado IP y configuración en servidor Tomcat) y personal administración (configuración en servidor Tomcat) **T3.6** **(**Creación de las clases Java y páginas JSP que componen el Proyecto de manera incremental).	
16	07/06/2004 13/06/2004	**T5.3** Memoria: Manual de instalación del producto	

		Revisiones de la memoria, presentación, etc. **T3.6** **(**Creación de las clases Java y páginas JSP que componen el Proyecto de manera incremental).	
17	14/06/2004 20/06/2004	**T5.4** Memoria: apéndices con el código fuente. **T5.5** Presentación, diapositivas explicativas del TFC. Revisiones de la memoria, presentación, etc. Correcciones de última hora.	**18/06/2004** Envío Final. Memoria, producto y presentación

En la planificación se contempla la carga de trabajo adicional que conllevan las PACs y que se añade a las tareas que he planificado.
Los colores representan los meses que componen el cuatrimestre.

Diagrama de los casos de uso.



Descripción textual de los casos de uso.

Caso de uso: 1. Creación Trabajador
Resumen de la funcionalidad: crea un trabajador nuevo.
Papel dentro del trabajo del usuario: ocasional.
Actores: Administración o **Encargado.**
Casos de uso relacionados: 2. Tratar Trabajador
Precondición: ninguna.
Postcondición: se ha creado un nuevo trabajador.
Proceso normal principal:

1. El personal de administración introduce los datos personales del trabajador: *el nombre, apellidos, DNI, teléfono particular, dirección, población donde reside cada empleado, código interno y el equipo de trabajo al cual va ser asignado.*
2. El sistema almacena los datos del nuevo empleado.
3. El sistema ejecuta el caso de uso 2. Tratar Trabajador

Alternativas de proceso y excepciones:

2a. El trabajador ya existe en el sistema.
2a1. El sistema muestra el listado de trabajadores (sin cambios).

Caso de uso: 2. Tratar Trabajador
Papel dentro del trabajo del usuario: habitual.
Actores: Administración y Encargado.
Casos de uso relacionados: 3. Consulta Trabajador 4. Asignación teléfono a Trabajador 5. Asignación Vehículo a Trabajador
Precondición: ninguna.
Postcondición: ninguna, ya que se trata de una consulta.
Proceso normal principal:

1. El usuario selecciona la consulta.
2. El sistema presenta los siguientes datos de cada trabajador: *nombre, apellidos, dirección, población, código interno de cada trabajador y el equipo al que pertenece.*

Alternativas de proceso y excepciones:

2a. El usuario selecciona editar trabajador.
1a1. El sistema muestra todos los campos del trabajador para su edición.
2b El usuario selecciona borrar trabajador.
2b1. El sistema pide confirmación eliminación del trabajador.
2b11 El usuario confirma eliminación trabajador.
2b111 El sistema elimina el registro y se inicia el paso 2 de nuevo.
2b12 El usuario confirma la **no** eliminación del trabajador.
2b121 El sistema inicia el paso 2 de nuevo.

Caso de uso: 3.Tratar Trabajador
Papel dentro del trabajo del usuario: habitual.
Actores: Invitado.
Casos de uso relacionados:
Precondición: ninguna.
Postcondición: se ha consultado el listado de los trabajadores.
Proceso normal principal:

1. El usuario selecciona la consulta.
2. El sistema presenta los siguientes datos de cada trabajador: *nombre, apellidos, dirección, población, código interno de cada trabajador y el equipo al que pertenece.*

Alternativas de proceso y excepciones:

Caso de uso: 4. Asignación teléfono a Trabajador
Papel dentro del trabajo del usuario: ocasional.
Actores: Encargado.
Casos de uso relacionados: 2. Tratar Trabajador
Precondición: se ha seleccionado un trabajador para asignar teléfono.
Postcondición: se ha asignado teléfono al trabajador.
Proceso normal principal:

1. El sistema ejecuta el caso de uso 2. Tratar Trabajador.
2. El usuario selecciona un trabajador.
3. El sistema pide: *número de teléfono interno a asignar.*
4. El usuario introduce el número de teléfono.
5. El sistema asigna el teléfono al trabajador.
6. El sistema informa de la asignación del teléfono al trabajador.

Alternativas de proceso y excepciones:

3a. El usuario cancela el trabajador seleccionado.
3a1. El sistema cancela el proceso.
5a. El teléfono ya está asignado a un trabajador.
5a1. El sistema no realiza la asignación.

Caso de uso: 5. Asignación Vehículo a Trabajador
Papel dentro del trabajo del usuario: ocasional.
Actores: Encargado.
Casos de uso relacionados: 2. Tratar Trabajador
Precondición: se ha seleccionado un trabajador para asignar vehículo.
Postcondición: se ha asignado vehículo al trabajador.
Proceso normal principal:

1. El sistema ejecuta el caso de uso 2. Tratar Trabajador.
2. El usuario selecciona un trabajador.
3. El sistema pide: *año de matriculación del vehículo, matricula, marca y modelo del vehículo.*
4. El usuario introduce los datos.
5. El sistema asigna el vehículo al trabajador.
6. El sistema informa de la asignación del vehículo al trabajador.

Alternativas de proceso y excepciones:

3a. El usuario cancela el trabajador seleccionado.
3a1. El sistema cancela el proceso.
5a. El vehículo ya está asignado a un trabajador.
5a1. El sistema no realiza la asignación.

Caso de uso: 6. Creación Factura
Papel dentro del trabajo del usuario: habitual.
Actores: Encargado.
Casos de uso relacionados: ninguno
Precondición: ninguna.
Postcondición: se ha creado una factura de un determinado teléfono asignado a un trabajador que corresponde al gasto de un mes.
Proceso normal principal:

1. El sistema pide: *fecha de emisión de la factura telefónica, número de teléfono interno al que corresponde dicha factura y el importe de la factura.*
2. El usuario introduce los datos.
3. El sistema almacena la factura correspondiente en la base de datos.

Alternativas de proceso y excepciones:

1a. El usuario cancela el proceso.
1a1. El proceso acaba.
3a. La factura ya existe en el sistema para la fecha indicada.
3a1. El sistema no realiza la introducción de la factura.
3b. El número de teléfono no existe.
3b1. El sistema no realiza la introducción de la factura.

Caso de uso: 7. Selección Facturas por fecha
Papel dentro del trabajo del usuario: habitual.
Actores: Encargado.
Casos de uso relacionados: ninguno.
Precondición: ninguna.
Postcondición: se presenta el listado de facturas correspondientes a los teléfonos del centro de trabajo para una determinada fecha.
Proceso normal principal:

1. El usuario introduce los meses para procesar las facturas.
2. El sistema presenta las facturas de los trabajadores para los meses que se ha introducido (si se quiere saber el importe de los últimos 3 meses, el actor introduciría 3, para 6 meses 6, 9, hasta los 12 últimos meses), de cada factura se presenta el *número de facturas introducidas, número de teléfono, nombre y apellidos del trabajador así como la media del importe de las facturas presentadas para los meses indicados, además del equipo de trabajo al que pertenece el empleado.*

Alternativas de proceso y excepciones:

1a. El usuario introduce un valor incorrecto para procesar las facturas (por ejemplo un número negativo).
1a1. El sistema muestra las facturas correspondientes al último mes.
1b. El usuario cancela el proceso.
1b1. El proceso acaba y retorna al menú principal.

Caso de uso: 8. Tratar teléfonos
Papel dentro del trabajo del usuario: ocasional.
Actores: Encargado.
Casos de uso relacionados: ninguno.
Precondición: ninguna.
Postcondición: se ha tratado los diferentes usuarios de telefonía móvil dentro de la empresa.
Proceso normal principal:

1. El usuario confirma la consulta.
2. El sistema presenta los siguientes datos de cada teléfono móvil: *número de teléfono, nombre del trabajador, apellidos y código de empleado.*
3. El usuario selecciona un registro.

Alternativas de proceso y excepciones:

1a. El usuario cancela el proceso.
1a1. El proceso acaba.
3a. El usuario cancela el proceso.
3a1. El proceso acaba.
3b. El usuario modifica los datos del registro seleccionado.
4b1. El usuario cancela el proceso.
4b11. El proceso acaba.
4b2. El sistema regraba la modificación en la base de datos.
4b3. El sistema determina que se está asignando un número de teléfono a dos trabajadores.
4b31. El sistema no realiza ningún cambio.

Caso de uso: 9. Consulta de Vehículos por siniestro y multa
Papel dentro del trabajo del usuario: ocasional.
Actores: Encargado.
Casos de uso relacionados: ninguno.
Precondición: ninguna.
Postcondición: se presenta el listado de los vehículos que han tenido multas y partes de accidente.
Proceso normal principal:

1. El usuario confirma la selección.
2. El sistema presenta una lista de los vehículos del centro de trabajo, de cada registro se presenta el *nombre del trabajador, apellidos, matricula, marca, modelo del vehículo, número de multas y número de partes de accidente para este vehículo.*

Alternativas de proceso y excepciones:

Caso de uso: 10. Consulta de Vehículos
Papel dentro del trabajo del usuario: ocasional.
Actores: Encargado.
Casos de uso relacionados: ninguno.
Precondición: ninguna.
Postcondición: se presenta el listado de los vehículos del centro de trabajo.
Proceso normal principal:

3. El usuario confirma la selección.
4. El sistema presenta una lista de los vehículos del centro de trabajo, de cada registro se presenta el *nombre del trabajador, apellidos, matricula, marca y modelo del vehículo*.

Alternativas de proceso y excepciones:

Caso de uso: 11. Tratar Vehículos.
Papel dentro del trabajo del usuario: ocasional.
Actores: Encargado.
Casos de uso relacionados: 9. Consulta de Vehículos por siniestro y multa
Precondición: ninguna.
Postcondición: ninguna.
Proceso normal principal:

1. El sistema ejecuta el caso de uso 9. Consulta de Vehículos por siniestro y multa
2. El usuario selecciona un vehículo.

Alternativas de proceso y excepciones:

2a. El usuario elimina el vehículo seleccionado.
2a1. El sistema desvincula el vehículo del trabajador.
2b. El usuario cancela el proceso.
2b1. El proceso acaba.
2c. El usuario modifica los datos del registro seleccionado (asignar un vehículo a otro trabajador, introducir multas y asignar partes de accidente).
2c1. El sistema regraba la modificación en la base de datos.
2c2. El usuario cancela el proceso.
2c21. El proceso acaba.
2c3. El sistema determina que se está asignando un mismo vehículo a dos trabajadores.
2c31. El sistema muestra un mensaje de error al intentar asignar un vehículo a dos trabajadores y se vuelve al paso 1.
2c4. El usuario selecciona el apartado de multas.
2c41. *El sistema muestra las multas que ha tenido el vehículo seleccionado desde su matriculación: fechas en la que se han producido, tipo de infracción e importe de cada una de la multas*.
2c411. El usuario añade nueva multa, rellenando los campos necesarios.
2c412. El usuario cancela el proceso.
2c4122. El proceso acaba.
2c413. El usuario selecciona una multa del listado presentado.
2c4133. El usuario elimina la multa seleccionada.
2c41333. sistema elimina la multa del sistema.
2c5. El usuario selecciona el apartado de partes de accidente.
2c51. *El sistema muestra los partes de accidente que ha tenido el vehículo desde su matriculación: fechas en la que se han producido los partes de accidente, causa, infractor e importe si es el caso.*
2c511. El usuario añade nuevo parte de accidente, rellenando los campos necesarios.
2c512. El usuario cancela el proceso.
2c5122. El proceso acaba.
2c513. El usuario selecciona un parte de accidente del listado mostrado.
2c5133. El usuario elimina un parte seleccionado.
2c51333. El sistema elimina el parte del sistema.

Caso de uso: 12. Listado Equipos
Papel dentro del trabajo del usuario: ocasional.
Actores: Encargado.
Casos de uso relacionados: ninguno.
Precondición: ninguna.
Postcondición: se presenta el listado de los equipos de trabajo.
Proceso normal principal:

1. El usuario confirma la selección.
2. El sistema presenta una lista de los equipos del centro de trabajo.

Alternativas de proceso y excepciones:

2a. El usuario añade un equipo.

2a1. El sistema añade nuevo equipo de trabajo.

Glosario.

Trabajador: Profesional de mantenimiento de la empresa, dicho trabajador es un técnico cualificado dedicado a tareas de mantenimiento, averías y pequeñas reparaciones en aparatos elevadores.

Equipo de trabajo: Conjunto de profesionales de mantenimiento coordinado por una persona (Jefe de equipo que también es un profesional de mantenimiento), cada profesional de mantenimiento pertenece a un equipo de conservación.

Número de teléfono interno: Número de cinco dígitos que corresponde a la extensión interna de cada teléfono que se asigna a los trabajadores (Por ejemplo 36048,36007, etc.).

Usuario: Persona que interactúa con el sistema informático.

Código de empleado: Número de 5 dígitos que corresponde al código que se asigna a todos los trabajadores, este código identifica a cada uno y mediante el se puede saber los avisos que ha atendido un determinado trabajador (en cada parte de reparaciones se ha de poner el código del técnico que ha atendido el aviso, reparación, etc.).

ANÁLISIS.

Identificación de las clases de entidades.

Comenzaremos por identificar las clases de entidades a partir de los casos de uso. Para cada caso de uso se indican las clases que se encuentran; cuando una clase ya se había identificado en un caso de uso anterior, su nombre va seguido de un asterisco, mientras que las clases que son dudosas van seguidas de un interrogante.

Posibles clases.

- Caso de uso 1: 1.Creación Trabajador.

Trabajador, Nombre, Apellidos, DNI, Teléfono particular, Dirección, Población, Código empleado, Equipo de trabajo.

- Caso de uso 2: 2.Tratar Trabajador.

Usuario?, Trabajador*, Nombre*, Apellidos*, Dirección*, Población*, Código empleado*, Equipo de trabajo*.

- Caso de uso 3: 3.Consulta Trabajador.

Usuario*?, Trabajador*.

- Caso de uso 4: 4.Asignación Teléfono a Trabajador.

Usuario*?, Trabajador*, Número de teléfono interno.

- Caso de uso 5: 5.Asignación Vehículo a Trabajador.

Usuario*?, Trabajador*, Año matriculación vehículo, Matricula, Marca, Modelo, Vehículo.

- Caso de uso 6: 6.Creación Factura.

Fecha emisión factura, Número de teléfono interno*, Importe de la factura, Factura.

- Caso de uso 7: 7.Selección Facturas por fecha.

Usuario*?, Mes?, Factura*, Trabajador*, Nombre del trabajador*, Apellidos*, Importe de la factura*, Número de teléfono interno*, Equipo*.

- Caso de uso 8: 8.Tratar Teléfonos.

Usuario*?, Número de teléfono*, Nombre del trabajador*, Apellidos*, Código de empleado*, Trabajador*.

- Caso de uso 9: 9.Consulta Vehículos por siniestro y multas.

Usuario*?, Vehículo*, Lista de los vehículos del centro de trabajo, Nombre del trabajador*, Apellidos*, Marca*, Matricula*, Modelo del vehículo*, Número de multas, Número de partes de accidente.

- Caso de uso 10: 10.Consulta Vehículos

Usuario*?, Vehículo*, Lista de los vehículos del centro de trabajo, Nombre del trabajador*, Apellidos*, Marca*, Matricula*, Modelo del vehículo*.

- Caso de uso 11: 11.Tratar Vehículos.

Usuario*?, Vehículo*, Trabajador*, Número de Multas*, Fecha multa, Tipo de infracción, Importe multa, Número de partes de accidente*, Fecha parte accidente, Causa, Infractor, Importe parte.

- Caso de uso 12: 12.Listado Equipos

Usuario*?, Lista de Equipos.

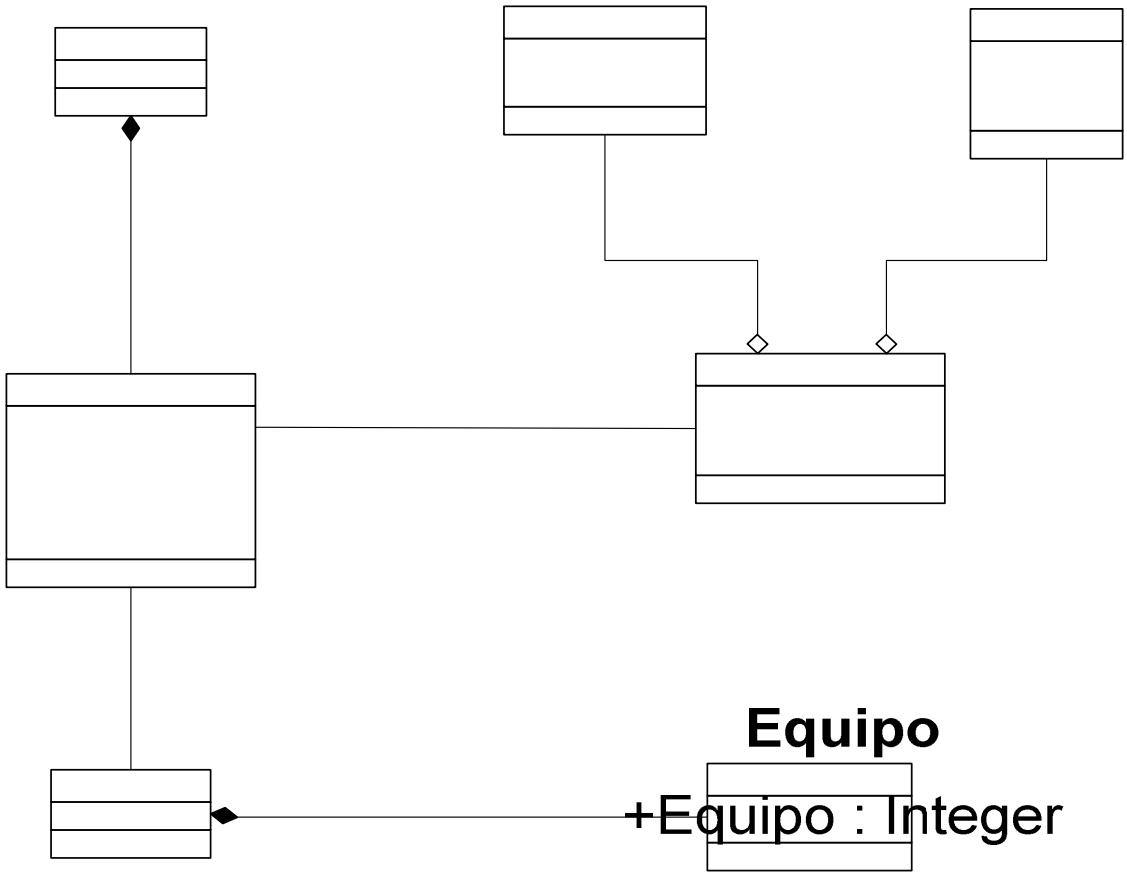
Comentarios.

- La clase *Usuario* se descarta por no tener atributos.
- *Nombre* es un atributo de la clase Trabajador.
- *Apellidos* es un atributo de la clase Trabajador.
- *DNI* es un atributo de la clase Trabajador.
- *Teléfono particular* es un atributo de la clase Trabajador.
- *Dirección* es un atributo de la clase Trabajador.
- *Población* es un atributo de la clase Trabajador.
- *Código empleado* es un atributo de la clase Trabajador.
- *Equipo de trabajo* es una composición de trabajadores.
- *Número de teléfono interno* es un atributo de la clase Teléfono.
- *Año matriculación* es un atributo de la clase Vehículo.
- *Matricula* es un atributo de la clase Vehículo.
- *Marca* es un atributo de la clase Vehículo.
- *Modelo* es un atributo de la clase Vehículo.
- *Fecha emisión factura* es un atributo de Factura.
- *Importe de la factura* es un atributo de Factura.
- *Lista de los vehículos del centro de trabajo* es un subconjunto de objetos de la clase Vehículo.
- *Número de multas* es un subconjunto de objetos de la clase Multa.
- *Número de partes de accidente* es un subconjunto de objetos de la clase Vehículo.
- *Fecha multa* es un atributo de la clase Multa.
- *Tipo de infracción* es un atributo de la clase Multa.
- *Importe multa* es un atributo de la clase Multa.
- *Fecha parte accidente* es un atributo de la clase ParteAccidente.
- *Causa* es un atributo de la clase ParteAccidente.
- *Infractor* es un atributo de la clase ParteAccidente.
- *Importe parte* es un atributo de la clase ParteAccidente.

Atributos de las clases resultantes.

Trabajador: Nombre, Apellidos, DNI, Teléfono particular, Dirección, Población, Código de empleado.
Teléfono: Número de teléfono.
Vehículo: Año de matriculación, Matricula, Marca, Modelo.
Factura: Fecha de emisión factura, Importe de la factura.
Equipo: Equipo.
ParteAccidente: FechaAccidente, Causa, Infractor, Importe.
Multa: FechaMulta, TipoInfracción, Importe.

Diagrama estático.

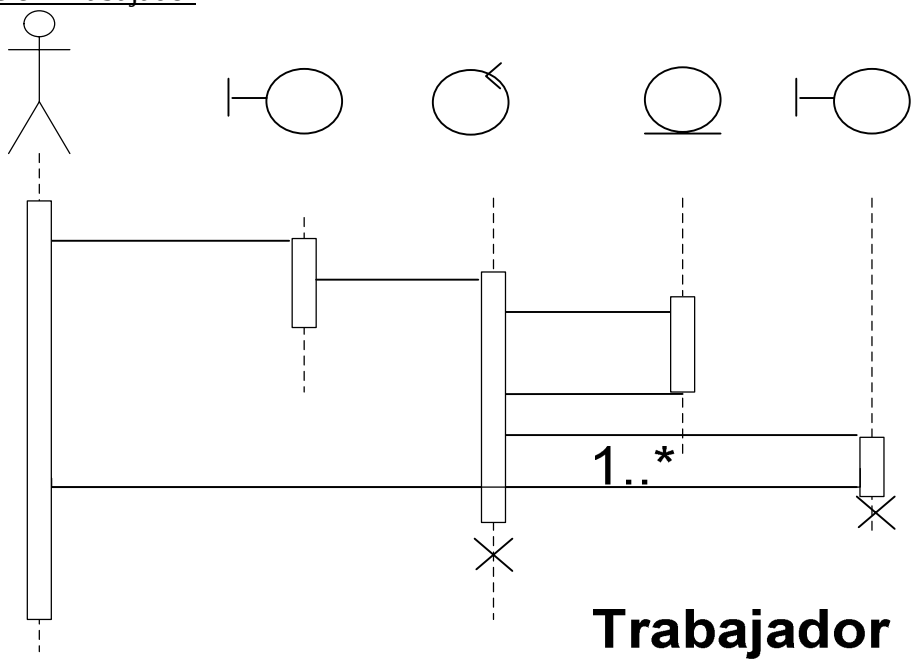


Diagramas de Secuencia.

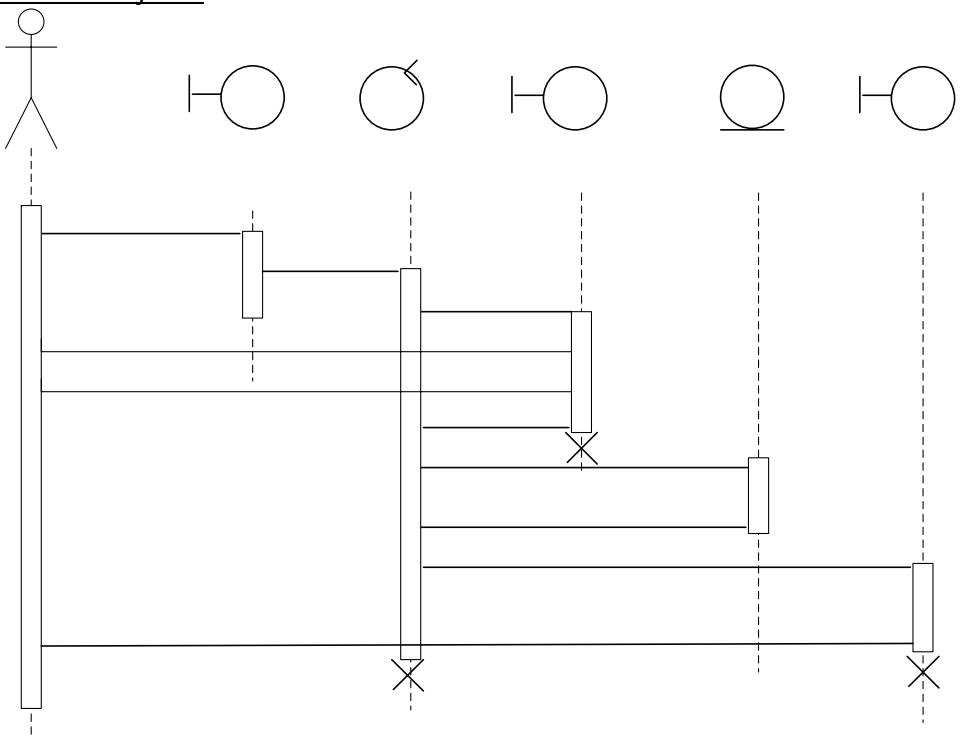
Realización de los diagramas de secuencia para los casos de uso 1. Creación Trabajador y 2.Consulta Trabajador.

1

1. Creación Trabajador

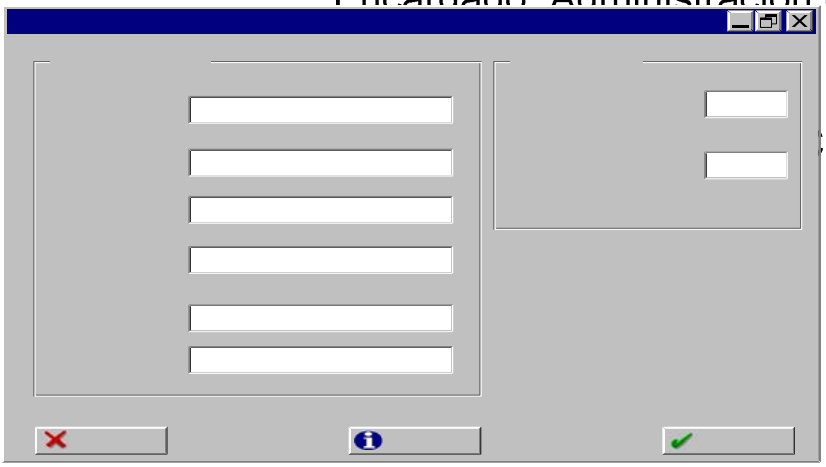


3. Consulta Trabajador

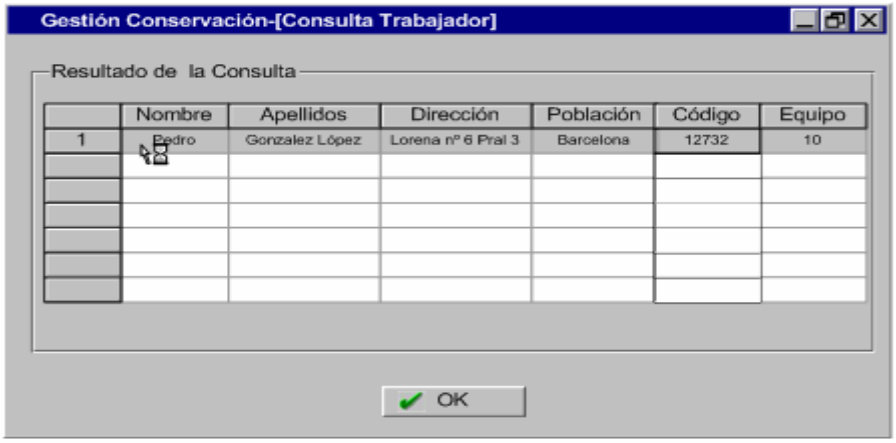


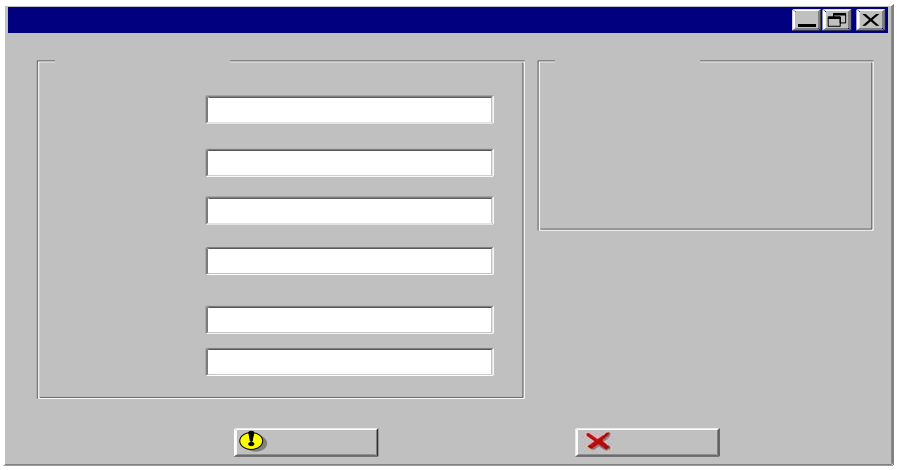
Representación Gráfica (Casos de uso más representativos).

Caso de uso 1: Creación Trabajador.



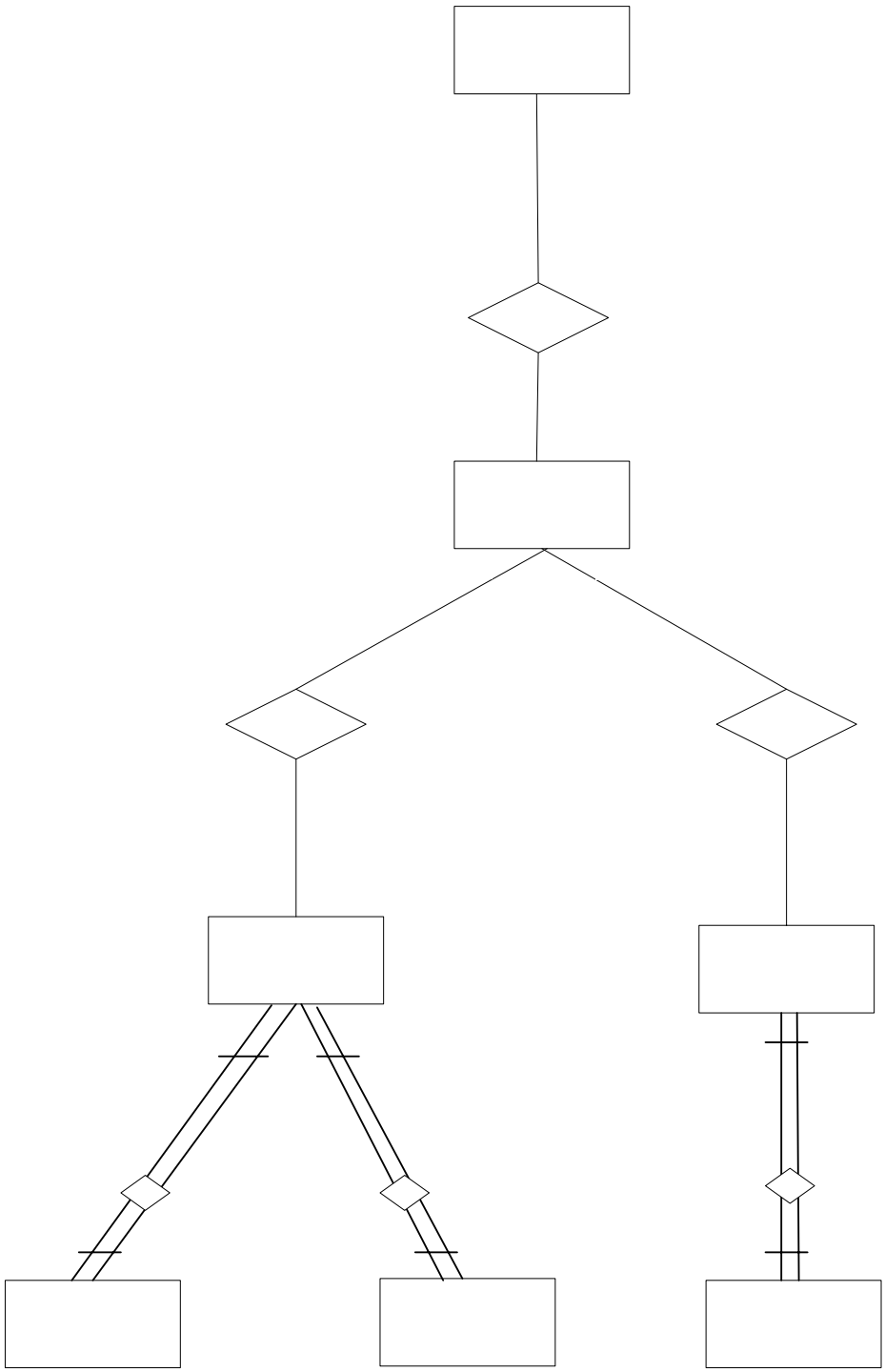
Caso de uso 2: Tratar Trabajador.





DISEÑO.

Esquema conceptual de la base de datos.



Diseño lógico base de datos.

Entidades.

Equipo (NombreEquipo)

Trabajador (Codigo_Empleado, Nombre, Apellidos, DNI, Telefono_Particular, Direccion, Población, Equipo)
{Equipo} Clave foránea a Equipo(NombreEquipo)

Vehiculo (Matricula, Año_Matriculacion, Marca, Modelo, CodigoTrabajador)
{CodigoTrabajador} Clave foránea a Trabajador (Codigo_Empleado)

Multa (Vehiculo, Fecha, TipoInfraccion, Importe)
{Vehiculo} Clave foránea a Vehiculo (Matricula)

ParteAccidente (Vehiculo, Fecha, Causa, Infractor, Importe)
{Vehiculo} Clave foránea a Vehiculo (Matricula)

Telefono (Numero, CodigoTrabajador)
{CodigoTrabajador} Clave foránea a Trabajador (Codigo_Empleado)

Factura (Telefono, Fecha_Emision, Importe)
{Telefono} Clave foránea a Telefono (Numero)

Tablas obtenidas.

EQUIPO		
NombreEquipo	INTEGER	Clave Primaria

TRABAJADOR		
Codigo_Empleado	INTEGER	Clave Primaria
Nombre	CHAR(15)	
Apellidos	CHAR(50)	
DNI	CHAR(9)	
Telefono_Particular	CHAR(9)	
Direccion	CHAR(40)	
Poblacion	CHAR(20)	
Equipo	INTEGER	C. FORANEA (EQUIPO)

VEHICULO		
Matricula	CHAR(10)	Clave Primaria
Año_Matriculacion	INTEGER	
Marca	CHAR(8)	
Modelo	CHAR(10)	
CodigoTrabajador	INTEGER	C. FORANEA (TRABAJADOR)

MULTA		
Vehiculo	CHAR(10)	C. FORANEA (VEHICULO)
Fecha	DATA	Clave Primaria
TipoInfracion	CHAR(50)	
Importe	REAL	

PARTEACCIDENTE		
Vehiculo	CHAR(10)	C. FORANEA (VEHICULO)
Fecha	DATA	Clave Primaria
Causa	CHAR(60)	
Infractor	CHAR(10)	
Importe	REAL	

TELEFONO		
Numero	CHAR(5)	Clave Primaria
CodigoTrabajador	INTEGER	C. FORANEA (TRABAJADOR)

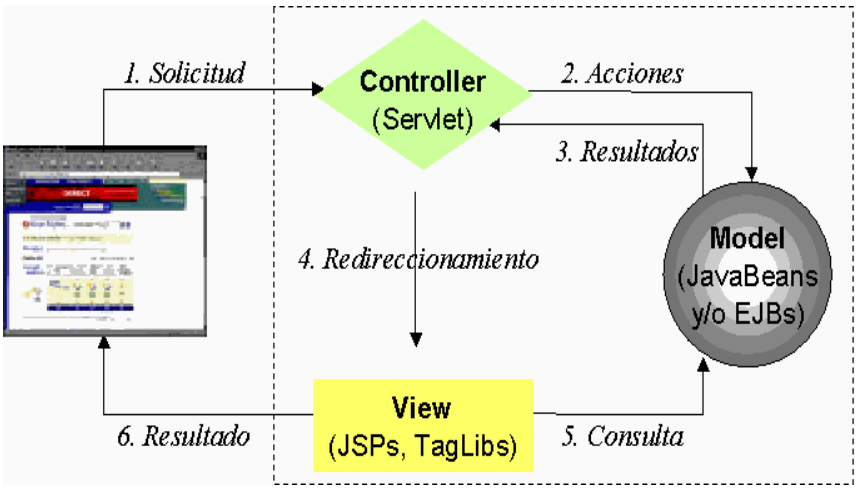
FACTURA		
Telefono	CHAR(5)	C. FORANEA (TELEFONO)
Fecha_Emision	DATA	Clave Primaria
Importe	REAL	

ARQUITECTURA A EMPLEAR.

Para la realización del proyecto, la arquitectura empleada ha sido Model View Controller (MVC), ya que dada la complejidad de éste he considerado que es la mejor opción.

El patrón de arquitectura MVC (Model-View-Controller) es un patrón que define la organización independiente del Model (Objetos de Negocio), la View (interfaz con el usuario u otro sistema) y el Controller (controlador del workflow de la aplicación: "si estoy aquí y me piden esto, entonces hacer tal cosa, si sale bien mostrar esto y si no aquello otro").

Funcionamiento en aplicaciones Web.



El navegador genera una solicitud que es atendida por el Controller (Servlet especializado). El mismo se encarga de analizar la solicitud, seguir la configuración que se le ha programado en su XML y llamar al Action correspondiente pasándole los parámetros enviados. El Action instanciará y/o utilizará los objetos de negocio para concretar la tarea. Según el resultado que retorne el Action, el Controller derivará la generación de interfaz a una o más JSPs, las cuales podrán consultar los objetos del Model a fines de realizar su tarea.

Definición del patrón MVC empleado.

Para este proyecto la aplicación resultante la he descompuesto en tres grandes bloques:

- Controlador
- Modelo
- Vistas

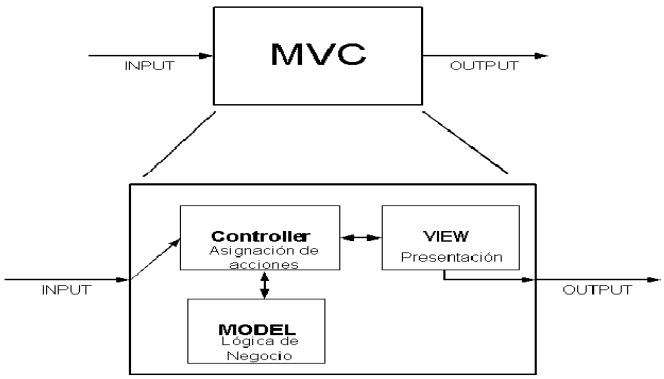
El **controlador** es el encargado de redirigir o asignar una aplicación (un modelo) a cada petición; es decir, cuando el usuario mediante el navegador realiza una petición (por ejemplo ver listado de trabajadores), dicha petición la recibe el Controlador (AdminController, WebController o FrontDeskController en función del menú usuario), una vez detectada la petición se realiza la acción a través de los manejadores (Modelo), en este caso sería el ManejadorTrabajador, dicho Manejador es el que accede a la base de datos (ocultando el JDBC al controlador y demás capas) y retorna al controlador el resultado, es en este momento cuando el controlador presenta el resultado obtenido mediante la vista asignada (en este caso se devuelve mediante una variable de tipo String y está implementado en la SuperClase Controller, dicha variable tiene asociada la página JSP correspondiente), se define la ruta dónde se encuentra la página y se realiza la presentación del resultado.

El **modelo** seria la aplicación que responde a una petición, es la lógica de negocio. Una vez realizadas las operaciones necesarias el flujo vuelve al controlador y este devuelve los resultados a una **vista** asignada.

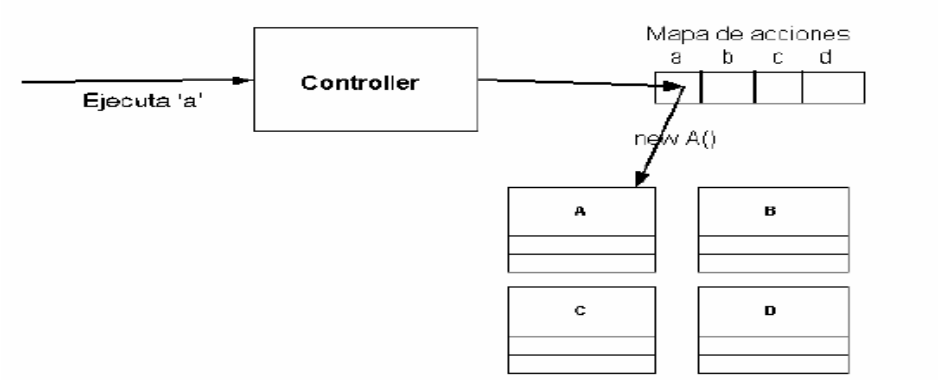
Vemos las diferencias que supone este modelo con los modelos convencionales.

Según el esquema más básico de programa, tenemos una entrada o parámetros que llegan (INPUT), se procesan y se muestra el resultado (OUTPUT).

En el patrón MVC el procesamiento lo llevamos a cabo entre sus tres componentes. El controller recibe una orden y decide quien la lleva a cabo en el modelo. Una vez que el modelo (la lógica de negocio) termina sus operaciones devuelve el flujo, vuelve al controller y este envía el resultado a la capa de presentación.



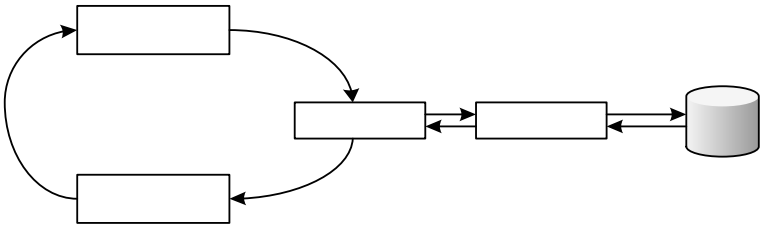
El Controller en cierta forma debe tener un registro de la relación entre ordenes que le pueden llegar y la lógica de negocio que le corresponde (mediante los controles de tipo IF implementados en cada controlador, que actúan como una operadora de teléfono que recibe una petición y une dos líneas). En el siguiente gráfico se representa ese funcionamiento:



Como ventajas obtenemos una separación total entre lógica de negocio y presentación. A esto se le pueden aplicar opciones como el multilenguaje, distintos diseños de presentación,.. etc. sin alterar la lógica de negocio. La separación de capas como presentación, lógica de negocio, acceso a datos es fundamental para el desarrollo de arquitecturas consistentes, reutilizables y más fácilmente mantenibles, lo que al final resulta en un ahorro de tiempo en desarrollo en posteriores proyectos.

Como resultado, nuestra implementación usará los siguientes componentes:

- Objeto Valor (Modelo): Los objetos valor en nuestra aplicación corresponden directamente a las principales entidades de la base de datos.
- Manejador (Modelo): Tenemos que crear una serie de manejadores de negocio que interactúan con la aplicación y la base de datos. Normalmente tenemos un manejador por cada objeto valor en la aplicación, aunque debemos implementar un manejador extra para proveer de requerimientos lógicos que necesitan las múltiples entidades del proyecto, además estos manejadores serán los que accedan a la base de datos.
- Servlet (Controlador): Los servlets dotaran de funcionalidad a la aplicación usando los manejadores, para manipular los objetos valor y así proveer a las páginas JSP con los recursos apropiados.
- JSP (Vistas): Las páginas JSP serán incluidas por los servlets para proveer la vista de la aplicación.



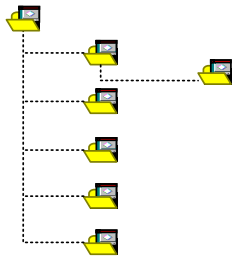
Ejemplo para visualizar listado Equipos.

1. El navegador realiza la petición al servlet específico.
2. El servlet pregunta sobre los contenidos de la petición y determina la acción apropiada a tomar. En este caso será (listarEquipos).
3. El servlet llama al manejador apropiado (ManejadorEquipo) para conseguir la lista de objetos ValorEquipo.
4. El manejador recibe la petición para la lista de objetos ValorEquipo.
5. El manejador ejecuta la conexión con la base de datos mediante el comando SQL apropiado.
6. Como respuesta al comando SQL el manejador recibe el objeto ResultSet procedente del driver JDBC (vía base de datos), representando la lista de registros de la tabla Equipo.
7. El manejador empaqueta el resultado como Colección de objetos ValorEquipo.
8. La colección de objetos ValorEquipo se pasan al servlet.
9. El servlet recibe la colección y la ubica en la página JSP apropiada.
10. La página JSP genera la página HTML apropiada para ser mostrada en el navegador.

10

IMPLEMENTACION.

Estructura de directorio.



- web: Directorio donde guardaremos las páginas HTML y JSP.
- web/WEB-INF: Directorio donde crearemos el fichero web.xml para empaquetar la aplicación.
- etc: Directorio donde guardamos cualquier tipo de archivo que pueda ser necesario.
- DB: Directorio donde guardamos los scripts que nos permitirán crear la base de datos o reconstruir si es necesario.
- docs: Directorio donde Javadoc pondrá las páginas de documentación.
- src: Directorio raíz donde tendremos los ficheros java y su estructura.

El fichero build.xml nos servirá para el programa Ant y así de este modo poder compilar y desarrollar este proyecto.

Puesta a punto de la base de datos.

SCRIPT creación BD.

```
CREATE DATABASE IF NOT EXISTS explotacion;
```

SCRIPT creación usuario BD.

```
grant all on explotacion.* to manuel identified by "redhat";
```

SCRIPT creación tablas de la BD.

```
use explotacion;

DROP TABLE IF EXISTS equipo;
CREATE TABLE equipo (
  NombreEquipo INTEGER(2) NOT NULL UNIQUE,
  PRIMARY KEY(NombreEquipo)
);


DROP TABLE IF EXISTS trabajador;
CREATE TABLE trabajador (
  CodigoEmpleado INTEGER(6) NOT NULL UNIQUE,
  Equipo INTEGER(2) NOT NULL,
  Nombre VARCHAR(15) NOT NULL,
  Apellidos VARCHAR(50) NOT NULL,
  DNI VARCHAR(9) NOT NULL UNIQUE,
  TelefonoParticular VARCHAR(9) NULL,
  Direccion VARCHAR(40) NOT NULL,
  Poblacion VARCHAR(20) NOT NULL,
  PRIMARY KEY(CodigoEmpleado),
  FOREIGN KEY(Equipo) REFERENCES equipo(NombreEquipo) ON DELETE
RESTRICT ON UPDATE RESTRICT
  MATCH FULL
);


DROP TABLE IF EXISTS Vehiculo;
CREATE TABLE Vehiculo (
  Matricula VARCHAR(10) NOT NULL,
  CodigoTrabajador INTEGER(6) NOT NULL UNIQUE,
  Año_Matriculacion INTEGER(4) NOT NULL,
  Marca VARCHAR(8) NOT NULL,
  Modelo VARCHAR(10) NOT NULL,
  PRIMARY KEY(Matricula),
  FOREIGN KEY(CodigoTrabajador)
    REFERENCES trabajador(CodigoEmpleado)
      ON DELETE CASCADE
      ON UPDATE CASCADE
);


DROP TABLE IF EXISTS Multa;
CREATE TABLE Multa (
  Fecha DATE NOT NULL,
  Vehiculo VARCHAR(10) NOT NULL,
  TipoInfracion VARCHAR(50) NOT NULL,
  Importe REAL NOT NULL,
  PRIMARY KEY(Fecha, Vehiculo),
```

```
  FOREIGN KEY(Vehiculo)
    REFERENCES Vehiculo(Matricula)
      ON DELETE CASCADE
      ON UPDATE CASCADE
);


DROP TABLE IF EXISTS ParteAccidente;
CREATE TABLE ParteAccidente (
  Fecha DATE NOT NULL,
  Vehiculo VARCHAR(10) NOT NULL,
  Causa VARCHAR(60) NOT NULL,
  Infractor VARCHAR(10) NOT NULL,
  Importe REAL NOT NULL,
  PRIMARY KEY(Fecha, Vehiculo),
  FOREIGN KEY(Vehiculo)
    REFERENCES Vehiculo(Matricula)
      ON DELETE CASCADE
      ON UPDATE CASCADE
);


DROP TABLE IF EXISTS Telefono;
CREATE TABLE Telefono (
  Numero INT(5) NOT NULL,
  CodigoTrabajador INTEGER(6)NOT NULL UNIQUE,
  PRIMARY KEY(Numero),
  FOREIGN KEY(CodigoTrabajador)
    REFERENCES trabajador(CodigoEmpleado)
      ON DELETE CASCADE
      ON UPDATE CASCADE
);


DROP TABLE IF EXISTS Factura;
CREATE TABLE Factura (
  Fecha_Emision DATE NOT NULL,
  Telefono INT(5) NOT NULL,
  Importe REAL NOT NULL,
  PRIMARY KEY(Fecha_Emision, Telefono),
  FOREIGN KEY(Telefono)
    REFERENCES Telefono(Numero)
      ON DELETE CASCADE
      ON UPDATE CASCADE
);
```

SCRIPT inserción de datos en BD.

Este fichero lo he creado para simulación del sistema.
Fichero Insertar_Datos.sql

Creación de un nuevo contexto.

Creamos el archivo llamado **template.context.xml** en el directorio raíz de nuestro proyecto, creamos también el fichero **web.xml** que situamos dentro de web/WEB-INF.

Puesta a punto de ANT para el proyecto.

Creamos el fichero **build.xml** y lo guardamos dentro del directorio raíz de nuestro proyecto, creamos también el fichero **build.properties** en el que definimos una serie de variables.

Conectividad con la base de datos.

Nuestro proyecto se ha de comunicar con la base de datos MySQL, por tanto, añadiremos una serie de modificaciones en el fichero **template.context.xml** para poder comunicarnos con la base de datos (añadiremos el driver, username, password, URL).

Documento sobre el desarrollo.

En este punto, ya tenemos el entorno listo para desarrollar el proyecto, hemos realizado los siguientes pasos:

- Creación de la estructura de desarrollo (árbol de directorios).
- Creación de la estructura "package" para los archivos fuente.
- Puesta a punto de la Base de Datos (MySQL) mediante el script.
- Creación del fichero **template.context.xml,** archivo que define el contexto para la aplicación.
- Creación del fichero **web.xml, build.xml y build.properties.**
- Ejecutar Ant con la opción context para definir un nuevo contexto e informar a Tomcat de la creación.

Objetos Valor.

En el desarrollo de esta aplicación, se crearán 7 clases que corresponden a las entidades de la base de datos y unos objetos extra para aquellas consultas que combinan diferentes tablas llamados también "Custom Value Object"

ValorSeleccion, ValorSeleccion2, ValorSeleccionVehiculos y ValorParque son Custom Value Objects (son específicos a casos de uso).

- Sólo tienen los atributos necesarios
- Se usan en las fachadas del modelo, pero no en los DAOs
- Las fachadas del modelo no son reusables entre aplicaciones, dado que implementan casos de uso específicos a la aplicación
- Los Custom Value Objects se diseñan para dar soporte a casos de uso específicos.

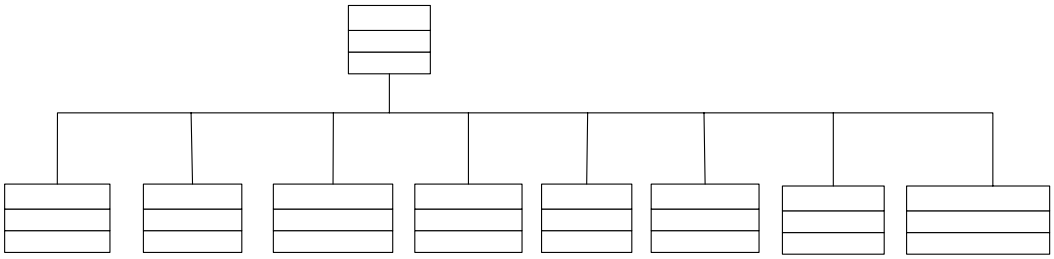
Cuando se han de utilizar:

- Aplicaciones de tamaño medio pueden requerir el uso de Custom Value Objects (además de los Value Objects)
- Lógicamente, sólo tiene sentido definir Custom Value Objects para aquellos casos de uso que requieren atributos de varios objetos del dominio, o existe una diferencia importante entre los atributos que se quiere visualizar y los que tiene el objeto del dominio correspondiente

Manejadores.

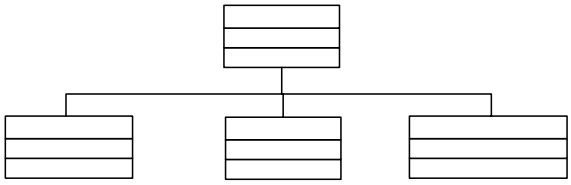
Los manejadores que utilizaremos, serán los que interactúen con la base de datos y la aplicación. Los mensajes que se enviarán a los manejadores serán simples peticiones del tipo: “salvar esto”, “dame eso”, “actualiza esto”. Como respuesta se han de dar los resultados deseados a dicha petición como: “Aquí tienes la colección de objetos”.

Puesto que todos los manejadores necesitarán acceso a recursos comunes, como conexión a base de datos y algunos métodos, es lógico tener una jerarquía de clases como la que muestro a continuación:



Servlets.

Mediante los servlets recogemos la petición del navegador y determinan las acciones apropiadas a tomar. En esta aplicación tendremos una jerarquía de Servlets como la siguiente:



La descripción de cada uno de los servlets:

- Controller: (Controller.java) Este servlet está declarado como abstracto, puesto que no se ha de instanciar directamente, dicho servlet contiene los métodos que son usados por las subclases, incluyendo los métodos estándar de los servlets como: destroy(), init(), y doGet(). Este servlet también contiene los métodos más comunes para la inclusión de nuestras páginas JSP.
- AdminController: (AdminController.java) Este servlet permite aquellas interacciones al sistema para los administradores (crear trabajador, crear equipo, listar equipo, etc.).
- FrontDeskController: (FrontDeskController.java) Este servlet permite interacciones con el personal de administración (listar equipos, añadir trabajador, listar trabajadores,etc).
- WebController: (WebController.java) Este servlet permite las interacciones al sistema para el invitado al sistema (listar equipos).

Seguridad accesos.

Para la implementación de seguridad he elegido dos tipos:

1. Control acceso mediante Login y Password para aquellos usuarios del tipo administrador y personal de administración.
2. Acceso seguro a invitados mediante el protocolo seguro HTTPS (creando el certificado para el servidor Tomcat y configurando el conector SSL en el fichero server.xml).
3. Seguridad Extra para acceso de los administradores mediante AccessAddressFilter.java, dicho control funciona de la siguiente manera: cuando el administrador del sistema introduce el Login y Password, se verifica la dirección IP almacenada en el fichero web.xml si se valida dicha dirección, nos permitirá el acceso al Menú de Administrador, en caso contrario no podremos acceder al sistema aun introduciendo el Login y Password válidos.

ESTADO DE LA CIENCIA.

A continuación, pasaré a describir brevemente y a remarcar aquellos aspectos más ventajosos, de las diferentes tecnologías que voy a emplear en mi TFC.

MySQL.

El servidor de bases de datos MySQL es muy rápido, seguro y fácil de usar, se encuentra en desarrollo constante y ofrece un conjunto rico y útil de funciones. Su conectividad, velocidad y seguridad hacen de MySQL un servidor apropiado para acceder a bases de datos en Internet.

Es un servidor de base de datos relacionales, se trata de un producto Open Source, (se puede usar y modificar). Podemos descargar el software de Internet y usarlo gratuitamente, también se puede estudiar el código fuente y modificar en función de las necesidades.

MySQL procede del mundo Linux, pero se comporta muy bien bajo ambiente Windows, la licencia es gratis para uso no comercial, al utilizar herramientas de programación posee una conexión ODBC mediante la cual se puede acceder a los datos, prácticamente sin cambiar el código fuente.

Es muy rápido al momento de insertar y recuperar información, permite almacenar un elevado número de registros, posee integridad referencial (tablas InnoDB), tiene seguridad a nivel de la aplicación, permitiendo crear usuarios y permisos de acceso.

J2EE.

J2EE es la plataforma que ofrece mejores perspectivas de desarrollo para empresas que quieran basar su arquitectura en productos basados en software libre. Es una especificación, un JSR (JSR-15), que define una plataforma de desarrollo empresarial.

La plataforma J2EE está formada por:

- Especificaciones.
- Test de compatibilidad (J2EE Compatibility Test Suite (CTS))
- Implementación de referencia de J2EE.
- Guías de desarrollo y de prácticas aconsejadas (J2EE BluePrints)

J2EE es la plataforma que está definida por la especificación JSR-151, en ella se definen las pautas, reglas y servicios que han de seguir y ofrecer los diferentes servidores de aplicaciones, las normas generales que han de cumplir los desarrolladores que quieran crear aplicaciones empresariales compatibles con J2EE y los roles que éstos pueden tomar.

El modelo de programación definido por la plataforma de J2EE va encaminado a la creación de aplicaciones basadas en n-capas, una aplicación puede tener cinco capas diferentes: capa cliente, capa de presentación (Servlets y JSP), capa lógica de negocio (EJB), capa de integración (Data Access Objects) y capa de sistemas de información.

J2EE es soporte de múltiples sistemas operativos, ya que es una plataforma basada en el leguaje Java. Ofrece soluciones libres, permite crear arquitecturas completas basadas únicamente en productos de software libre. Las soluciones basadas en J2EE ofrecen características como es rendimiento.

Los contenedores Web son la puerta de las aplicaciones Java a la World Wide Web, capaces de recibir peticiones HTTP y ejecutar las clases de Java indicadas para darles respuesta.

Las aplicaciones Web realizadas en Java pueden tener dos formas: Servlets y páginas JSP (Java Server Pages). Las especificaciones de estas tecnologías forman parte de J2EE. El principal objetivo de estas tecnologías es crear páginas Web dinámicas, es decir, que se generan en el servidor en base a nuestra lógica de negocio. Tanto los Servlets como las páginas JSP aprovechan todas las posibilidades de Java, lo que nos permite poder aprovechar la flexibilidad que ofrece un lenguaje de programación, heredan todas sus ventajas en forma de reutilización, mantenibilidad, extensibilidad, etc.

Apache Tomcat.

Apache Tomcat se ha convertido en la implementación oficial de referencia de Servlets y JSP, lo que lo convierte en el contenedor sobre el que se prueba su adhesión a las especificaciones.

A partir de las nuevas versiones de la serie 4.x, creadas para cumplir las especificaciones 2.3 de Servlets y 1.2 de JSP, Tomcat se ha reescrito entero lo que mejora considerablemente su rendimiento. Quizá siga sin ser la opción más indicada para una aplicación con miles de clientes simultáneos, pero ningún servidor en solitario es adecuado para una carga así. En todos los demás casos su rendimiento es más que aceptable.

Puede integrarse sin demasiados problemas con el servidor Web Apache, y para aplicaciones empresariales grandes con servidores de aplicaciones como JBoss o JOnAS. Asimismo puede integrarse con muchos otros productos con licencias propietarias. Tomcat es el contenedor Web que más se está utilizando en servidores de aplicaciones comerciales.

Muestra de lo estandarizado del uso del Tomcat es su utilización como servidor de Servlets y JSP por parte de los entornos de desarrollo, tanto libres como comerciales.

Ant.

- Herramienta del tipo de make (gnumake, nmake ...)
- Open Source
  - Proyecto Apache Jakarta (http://jakarta.apache.org/ant)
  - Utilizada en otros desarrollos (ej. Tomcat)
- Desarrollada en Java.
- Otras herramientas existentes.
  - Shell-based: Ejecutan comandos específicos del sistema operativo (no reutilizables en diferentes plataformas).
  - Formatos “estrictos” (ej. tabuladores en Makefiles)

- Ant en más portable.
  - Las tareas son ejecutadas por clases Java. Solo requiere una MV Java 1.1 o superior (Reutilizable en diferentes plataformas)
    - Existe una tarea que permite ejecutar comandos basados en el SO sobre el que se esté utilizando.
  - Utiliza ficheros de configuración XML.

XML

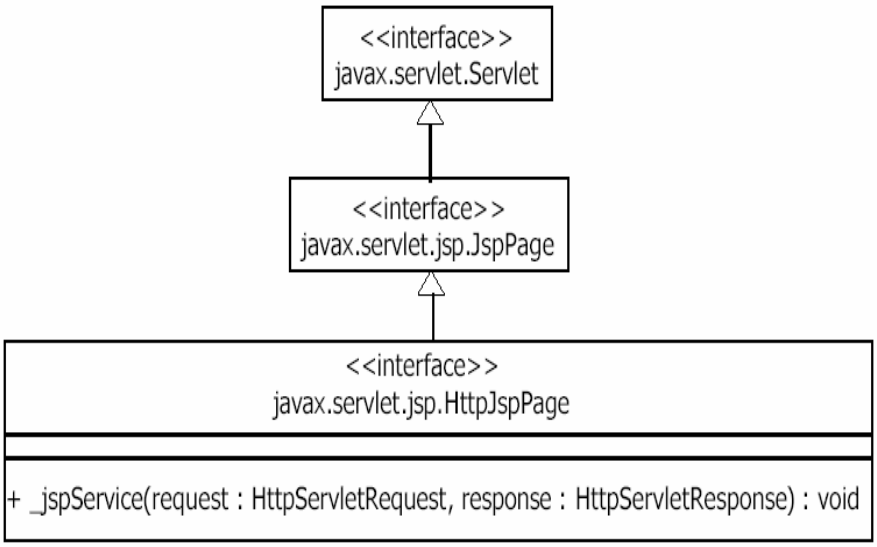
- XML (eXtensible Markup Language) es un lenguaje de tags
- Es extensible
- Permite escribir documentos de texto que expresan datos y no aspecto visual (a diferencia de HTML)
- Campos de aplicación
  - Intercambio de datos entre aplicaciones heterogéneas
  - Configuración de aplicaciones
  - Generación de aspecto visual (ej.: HTML) a partir de los datos
  - Bases de datos
  - ... y muchos otros ...

JSP

¿Qué es JSP?

- A modo de ejemplo, una página JSP que genera HTML
- Tiene el aspecto de una página HTML
- Puede incluir scriptlets (scripts) para generar HTML dinámicamente.
- Típicamente los scriptlets se escriben en Java

En realidad, una página JSP es un tipo especial de servlet (**javax.servlet.jsp** y **javax.servlet.jsp.tagext**) orientado a generar el texto de la interfaz gráfica, invocables por GET y POST



¿Qué ocurre cuando se accede a una página JSP?

- Si es la primera vez, el servidor de aplicaciones genera un servlet (que implementa **javax.servlet.jsp.HttpJspPage**) a partir de la página JSP, lo compila y lo carga en memoria
- Si no es la primera vez, le pasa la petición al servlet (ya compilado y creado en memoria)
- Si la página se ha modificado desde la última compilación, el servidor se da cuenta, genera el nuevo servlet, lo compila y lo carga de nuevo

Comentarios

- Usa **<%= expresión %>** para incluir expresiones Java
  - La expresión es evaluada en tiempo de ejecución y convertida a un **String**
- Usa scriptlets para incluir código Java
  - **<% ... %>**
- Objetos implícitos
  - **request**: **javax.servlet.http.HttpServletRequest**
  - **response**: **javax.servlet.http.HttpServletResponse**
  - **session**: **javax.servlet.http.HttpSession**
  - **out**: **javax.servlet.jsp.JspWriter**
  - Algunos más

CONCLUSIONES

Este proyecto que he realizado me ha servido para poder ver en primera persona como se ha de desarrollar un sistema orientado a Internet de manera fiable, y separando la aplicación en diferentes capas, de manera que sea portable y robusta.

Otro aspecto importante que destaco de este sistema, es la posibilidad de añadir nuevas funcionalidades de manera sencilla (actualizar el controller correspondiente, crear las vistas necesarias y si es el caso añadir manejadores y objetos valor), sin que esto suponga reorganizar toda la estructura.

Por ejemplo, si necesito crear una opción que me borre un registro de la tabla Equipo (que no está implementado), sólo necesitaría modificar el manejador correspondiente al Equipo creando este método, añadiendo la entrada al Manejador de Negocio, actualizando el controller y generando las vistas necesarias mediante las páginas JSP, el resto del sistema permanecería igual.

En cuanto a las herramientas utilizadas, cabe destacar el entorno de programación **Eclipse de IBM** que ha sido una valiosa herramienta, y me ha supuesto un ahorro de tiempo considerable con respecto a otros entornos con los que había trabajado anteriormente, su gran sencillez y la posibilidad de agregarle plugins (como Lomboz J2EE que he utilizado para la edición y corrección de páginas JSP entre otras funcionalidades) han sido de mucha utilidad para el desarrollo de este sistema.

Como aspecto negativo en este desarrollo, ha sido el comportamiento del servidor Apache Tomcat 5.0 que en algunas ocasiones se comportaba de una manera irregular (quizá por la falta de conocimientos acerca de su uso y configuración).

Puedo decir que con este trabajo he descubierto una nueva forma de realizar aplicaciones de manera sencilla, pero que a su vez ofrecen una robustez y portabilidad requeridas para un entrono profesional.

BIBLIOGRAFIA.

- Java 2; Manual Práctico.
  Benjamín Aumaille.
  Ediciones ENI, 2000.

- Programación en JAVA 2.
  John Zukowski.
  ANAYA multimedia, 1999.

- Manual de referencia J2EE.
  Jim Keogh.
  McGraw Hill, 2003.

- Comunicaciones y bases de datos con JAVA a través de ejemplos.
  Jesús Bobadilla/Adela Sancho.
  Ra-Ma, 2003.

- MySQL para Windows y Linux
  César Pérez.
  Ra-Ma 2004

- Apache Tomcat Bible
  Jon Eaves, Rupert Jones and Warner Godfrey.
  Wiley 2003.

- Enginyeria del Programari I
  Benet Campderrich Falgueras, Recerca informatica, SL
  Universitat Oberta de Catalunya /Informàtica.

- Bases de dades I
  Jaime Sistac Planas
  Universitat Oberta de Catalunya /Informàtica 2002.

Enlaces:

-Structs (Implementación del patrón MVC en aplicaciones Web) de Pello Xavier Altadill Izura.

-Tutoriales sobre MySQL en www.mysql-hispano.org

-Tutoriales sobre MVC www.programacion.com

-Otros tutoriales:

Tutorial sobre Ant y páginas JSP de Juan Raposo Santiago Facultad de Informática, Universidad de la Coruña jsr@udc.es

ANEXO

Manual de Instalación.

Utilización del controlador acceso BD (mysql) JDBC en JAVA.

Copiar el archivo `mysql-connector-java-3.0.8-stable-bin.jar` al directorio donde tenemos el gestor mysql.

C:\mysql\>`mysql-connector-java-3.0.8-stable-bin.jar`

Definimos la variable de ambiente.

```
SET CLASSPATH=c:\mysql\mysql-connector-java-3.0.8-stable-bin.jar
```

Ya podemos utilizar el driver para acceder desde cualquier programa JAVA mediante las siguientes sentencias:

```
Connection cn=null;
Class.forName( "com.mysql.jdbc.Driver" );
cn=DriverManager.getConnection(
"jdbc:mysql://127.0.0.1/explotacion","explotacion","explotacion");
```

Utilización del controlador acceso BD(mysql) JDBC para JSP.

Para la utilización del driver en las páginas JSP se tienen que seguir las siguientes instrucciones:

1. Copiar el archivo `mysql-connector-java-3.0.8-stable-bin.jar` en el directorio siguiente: c:\{Directorio dónde tengamos TOMCAT}\common\lib.
2. Con esto, ya hemos terminado la instalación, cualquier página que requiera acceso a la base de datos mediante este driver tendrá soporte.

Puesta en marcha de la base de datos.

En la línea de comandos de MySQL escribimos las siguientes sentencias:

1. mysql> source c:\TFC\db\Crear_BaseDatos.sql
2. mysql> source c:\TFC\db\Crear_Tablas.sql
3. mysql> source c:\TFC\db\Crear_Usuario.sql

Conectividad Base de Datos.

El archivo template.context.xml contiene todos los parámetros que necesitamos para conectarnos a la base de datos:

Ejemplo:

```
<Context displayName="Gestion Conservacion"
        docBase="@context.docbase@"
        path="@context.path@"
        debug="5"
        reloadable="true"
        useNaming="true" >
    <Loader checkInterval="1" />

<!-- Conectividad con la Base de datos -->
<Resource name="jdbc/GestionConservacionDb"
        auth="Container" type="javax.sql.DataSource"/>
    <ResourceParams name="jdbc/GestionConservacionDb">
        <parameter>
            <name>username</name>
            <value>manuel</value>
        </parameter>
        <parameter>
            <name>password</name>
            <value>redhat</value>
        </parameter>
        <parameter>
            <name>driverClassName</name>
            <value>com.mysql.jdbc.Driver</value>
        </parameter>
        <parameter>
            <name>url</name>
            <value>jdbc:mysql://localhost/explotacion</value>
        </parameter>
        <parameter>
            <name>maxActive</name>
            <value>100</value>
        </parameter>
        <parameter>
            <name>maxIdle</name>
            <value>10</value>
        </parameter>
    </ResourceParams>
<!-- Fin conectividad con la Base de datos-->

</Context>
```

Crear el archivo build.propierties

El archivo build.properties contiene el nombre de usuario y login para acceder a la sección manager del Tomcat.

Ejemplo:
# Archivo que utilizará Ant.
# Directorio CATALINA_HOME
catalina.home=c:/uoc/tomcat 5.0
# Configuración para el uso del manager con las tasks del Ant
manager.url=http://localhost:8080/manager
manager.username=manuel
manager.password=redhat

Pasos finales

Fichero web.xml

En el fichero web.xml situado en el directorio \web\WEB-INF\ tenemos la sección siguiente:

```
      <!-- FILTRADO DE DIRECCIONES IP -->
       <init-param>
          <param-name>ip-filter</param-name>
          <param-value>127.0.0.1</param-value>
       </init-param>
       <init-param>
          <param-name>error-page</param-name>
          <param-value>/access_err.html</param-value>
       </init-param>
   </filter>
```

Definimos la dirección IP desde donde nos conectaremos al menú del administrador.

También definimos los roles para los usuarios:

```
<login-config>
       <auth-method>FORM</auth-method>
       <form-login-config>
          <form-login-page>/login.html</form-login-page>
          <form-error-page>/login_err.html
          </form-error-page>
       </form-login-config>
   </login-config>
   <security-role>
       <description>Personal de Administracion</description>
       <role-name>Gestion-staff</role-name>
   </security-role>
   <security-role>
       <description>Administrador de Gestion</description>
       <role-name>Gestion-admin</role-name>
   </security-role>
```

Fichero tomcat-users

En el fichero de configuración de usuarios de Tomcat ubicado en \Tomcat5.0\conf definiremos a los usuarios y sus roles:

```
<?xml version='1.0' encoding='utf-8'?>
<tomcat-users>
  <role rolename="Gestion-staff" description=""/>
  <role rolename="tomcat"/>
  <role rolename="manager"/>
  <role rolename="admin"/>
  <role rolename="Gestion-admin" description=""/>
  <user username="Marta" password="Rodriguez" roles="Gestion-staff"/>
  <user username="Manuel" password="redhat" fullName="Manuel Fernandez Lopez"
roles="Gestion-admin,Gestion-staff"/>
  <user username="Jesus" password="Dolz" roles="Gestion-staff"/>
  <user username="manuel" password="redhat" fullName="Manuel Fernandez Lopez"
roles="admin,manager,tomcat"/>
  <user username="Gaspar" password="Vanrell" roles="Gestion-admin"/>
</tomcat-users>
```

Fichero server.xml

En el fichero server.xml ubicado en \Tomcat5.0\conf definiremos el SSL para el usuario invitado, realizaremos las siguientes modificaciones en dicho archivo.

```
   <!--  Define a SSL Coyote HTTP/1.1 Connector on port 8443 -->
   <Connector port="8443"
              maxThreads="150" minSpareThreads="25" maxSpareThreads="75"
              enableLookups="false" disableUploadTimeout="true"
              acceptCount="100" debug="0" scheme="https" secure="true"
              clientAuth="false" sslProtocol="TLS"
              keystoreFile="c:\uoc\tomcat 5.0\keystore"
              keystorePass="secret"/>
```

El fichero keystore definido en esta sección es el certificado que hemos creado mediante el siguiente comando:

\jdk1.4.1\bin\keytool –genkey –alias tomcat –keyalg RSA –keystore
Como password “secret”
El fichero keystore se copia al directorio \tomcat5. 0

Ejecución de ANT.

Después de hacer todos los pasos anteriores, ya podemos ejecutar las tareas con ANT con el servidor Apache Tomcat funcionando y el motor de base de datos.

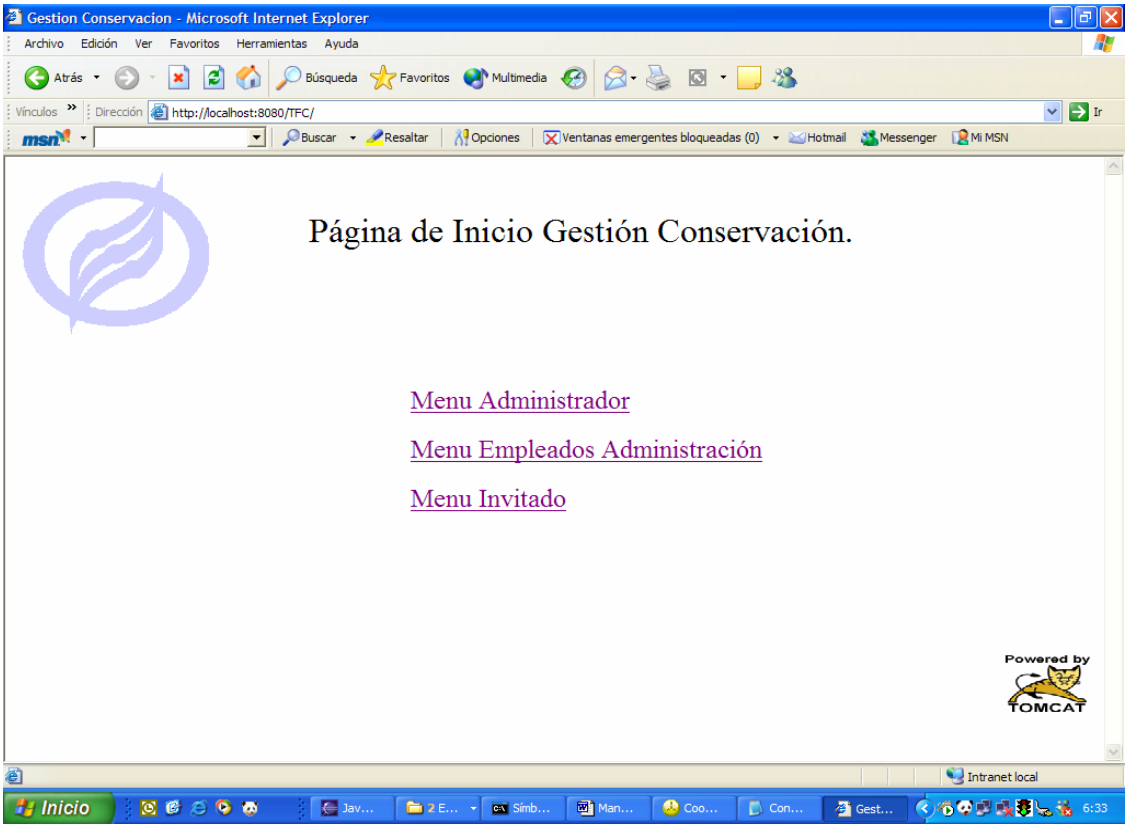
En el directorio \TFC ejecutaremos:

1. ant prepare
2. ant compile
3. ant context

EJEMPLOS DE EJECUCION.

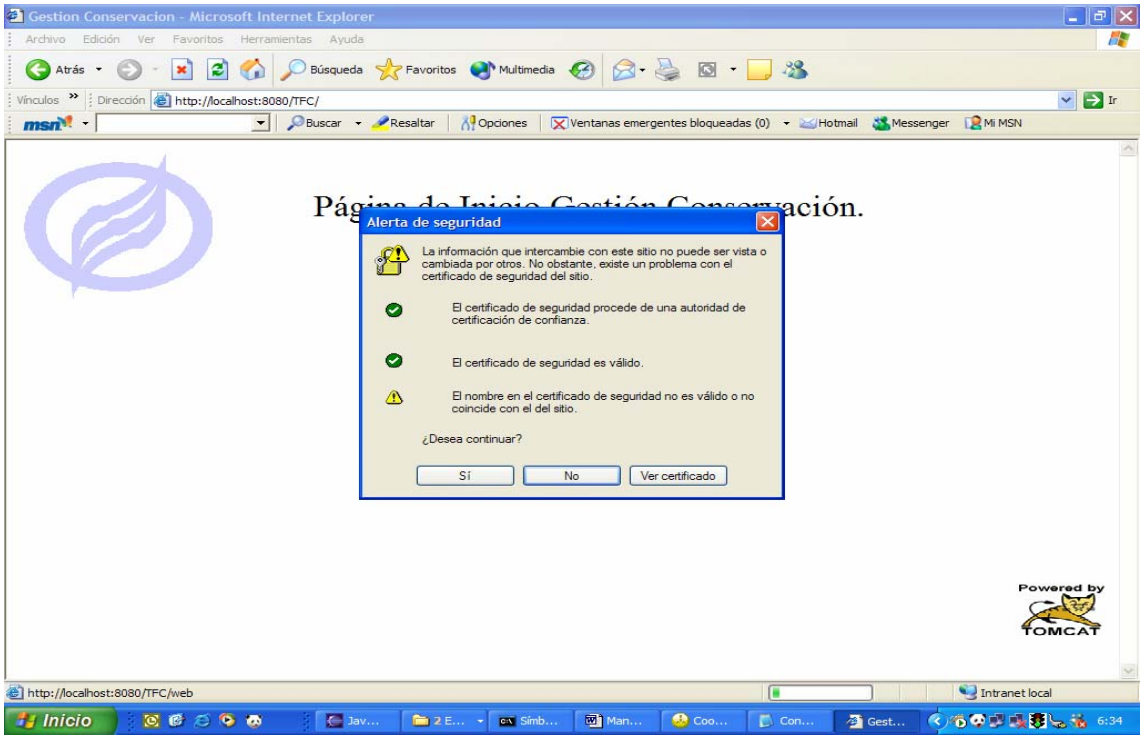
Una vez hecho los pasos anteriores, mediante el navegador ejecutamos:

http://localhost:8080/TFC/



Menú Invitado.

Si seleccionamos el menú invitado entraremos en el servidor seguro tal y como se muestra a continuación:



Comprobación entrada zona segura

Menú Administración.

En este caso si seleccionamos el Menú Empleados Administración, el sistema nos pedirá un login y Password:

Menú Administrador

En este caso al igual que en el caso anterior si seleccionamos el Menú Administrador, el sistema nos pedirá un login y Password, Si el sistema detecta que la IP del equipo es la misma que tenemos configurada en el archivo web.xml, entonces permitirá la entrada, sino nos mostrará un mensaje de error (en este caso se ha configurado la IP 127.0.0.1).