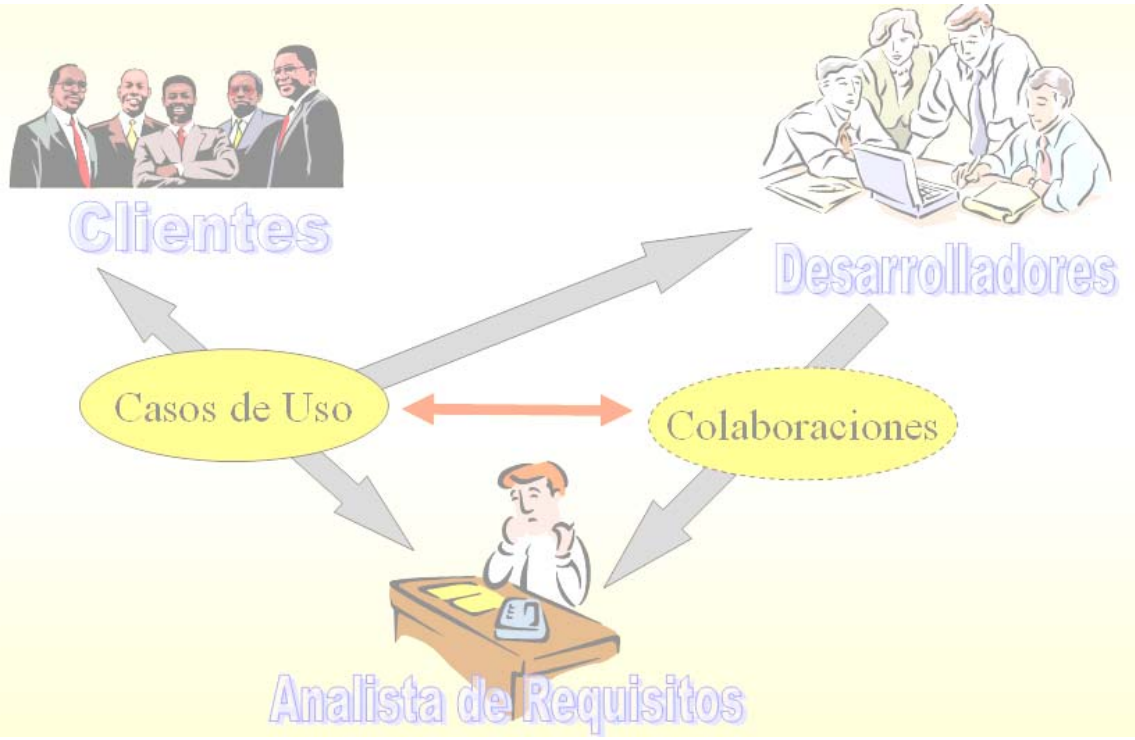
MEMORIA TRABAJO FINAL DE CARRERA

GMODI

GESTIÓN Y MANTENIMIENTO ONLINE DE DEPARTAMENTOS INFORMÁTICOS



Estudiante:

Ramon Balart Llaó
ITIG

Consultor:

Javier Ferro Garcia

ÍNDICE

PRÓLOGO 5

Capítulo 1: Introducción 6

1.1 Justificación del TFC 6

1.2 Contexto en el que se desarrolla el TFC 6

1.3 Punto de partida y aportación del TFC 6

1.4 Objetivos del TFC 7

1.4.1 Pongamos un ejemplo: 7

1.5 Planificación del proyecto 8
1.5.1 Diseño conceptual de las aplicaciones a realizar 9
1.5.2 Diseño conceptual de las bases de datos requeridas 9
1.5.3 Documentación analisi y diseño de las aplicaciones 9
1.5.4 Implementación de las bases de datos del diseño conceptual 9
1.5.5 Implementación del diseño conceptual de las aplicaciones 9
1.5.6 Integración entre las aplicaciones y las bases de datos 9
1.5.7 Introducción de datos para probar la aplicación en producción 9
1.5.8 Depuración de errores en producción 9

1.6. Productos obtenidos 10
1.6.1. Gestión de recambios: 10
1.6.2. Control de stocks: 10
1.6.3 Control y gestión de averías: 10

Capítulo 2: Modelado del negocio 11

2.1 Introducción 11

2.2 Identificación de Roles del Entorno del Negocio 12

2.3 Identificación de los casos de uso del negocio 13

2.4 Identificación de las relaciones entre casos de uso 14
2.4.1 Relaciones de extensión (extend) 14
2.4.2 Relaciones de inclusión (include) 15
2.4.3 Relaciones de generalización 15

2.5 Diagrama de los casos de uso del negocio 15

2.6 Descripción de los Casos de Uso del Negocio más representativos 17
2.6.1 Caso de uso “Alta de usuarios” 17
2.6.2 Caso de uso “Baja de usuarios” 17
2.6.3 Caso de uso “Modificación de usuarios” 18
2.6.4 Caso de uso “Consulta de stock de consumibles” 18

2.7 Futuro caso de uso 19
2.7.1 Caso de uso “Solicitud de pedido por falta de stock mínimo en consumibles” 19

2.8 Casos de uso opcionales a medio plazo 19
2.8.1 Caso de uso “Solicitar recambio” 20
2.8.2 Caso de uso “Solicitar mejora o reparación” 20
2.8.3 Caso de uso “Consultar estado solicitud de recambio” 21
2.8.4 Caso de uso “Consultar estado de mejora o reparación” 21

2.9 Diagrama de paquetes o de componentes 22

2.11 Especificación de clases 23
2.11.1 Introducción a la especificación de clases 23
2.11.2 Clases de entidad 24
2.12 Diagrama conceptual 25
2.13 Diagrama de estructura estática o de clases 26
2.14 Diagrama de secuencia 27
2.15 Diagrama de colaboración 28
2.16 Diagrama de estados 28
Capítulo 3: Presentación de la aplicación 29
3.1 El "login" del administrador 29
3.2 El *Front End* del "Administrador" 30
3.3 Mantenimiento de usuarios: 30
Capítulo 4: Diseño de la aplicación 33
4.1 Patron Value Object 33
4.2 Patron Session Façade 34
4.3 Patron Data Access Object 35
4.4 Patron Modelo Vista Controlador 37
Capítulo 5: Implementación de la aplicación 39
5.1 Software utilizado para la implementación de GMODI 39
5.2 Metodología para la implementación de GMODI 39
5.3 Esquema de la implementación de clases en GMODI 40
5.4 Despliegue de la aplicación GMODI 43
5.5 Comentarios extras de la aplicación GMODI 45
5.5.1 Uso de tags 45
5.5.2 Tratamiento de excepciones 48
5.6 Valoración personal 50

Con la colaboración de la empresa en la que trabajo en el departamento informático,

PLASTICOS CASTELLA

Empresa en la que estamos dedicados a la inyección de materiales termoplásticos, a la construcción de moldes y automatismos para la fabricación de los mismos y además imprimimos y troquelamos nuestras etiquetas de IML en diferentes soportes.

Especiales agradecimientos también a mi consultor Javier Ferro quien ha sido de gran apoyo en momentos en que la necesidad de superación ha sido parte fundamental para llegar al objetivo de este proyecto.

Finalmente deseo dar las gracias a mi familia y a mi pareja por todos los ánimos y apoyo que me han dado en estos duros pero reconfortantes años de estudio, ya que sin ellos gran parte de este esfuerzo no hubiera sido posible.

PRÓLOGO

A grandes rasgos, la finalidad de este proyecto és la de conseguir por un lado la gestión de recambios para los equipos informáticos de la empresa tales como, tonners de impresoras láser y faxes, cartuchos de tinta, scanners y términales de fábrica y finalmente impresoras de impresión térmica para etiquetas de los productos de nuestra fábrica. Por el otro lado tendremos el mantenimiento de los equipos informáticos de la empresa, és decir los PC's de trabajo.

Todo esto en aplicaciones donde los usuarios mediante un portal podran solicitar recambios para sus impresoras, al mismo tiempo que podran solicitar también servicio técnico al departamento informático para las averias que surjan en sus equipos. De este modo se pretende conseguir una lista actualizada de tareas para el departamento informático, la cual estará siempre al dia y con un orden de atención por parte de los técnicos.

El objetivo de este proyecto es conseguir dichas aplicaciones Servidor-Cliente con herramientas del entorno J2EE.
El motivo de escoger J2EE se debe a que es una plataforma que habilita soluciones para desarrollo, uso efectivo y manejo de multicapas en aplicaciones centralizadas en el servidor. Es decir, está basado en la arquitectura del lado del servidor (Server-based), la cual concentra la mayoría de procesos de la aplicación en el servidor o en una parte de este. Esto nos provee de dos ventajas críticas respecto los otros tipos de arquitectura, dichas ventajas son:

- Multiplos Clientes: La arquitectura basada en el servidor requiere una clara separación entre las capas de interfaz o cliente y la capa servidor, en la que se realizan los procesos de la aplicación.
- Operaciones robustas: Ya que esta arquitectura soporta escalabilidad, confiabilidad, disponibilidad y recuperabilidad. Es decir, las aplicaciones pueden ser divididas y distribuidas en múltiples procesadores.

En las siguientes páginas de la memoria se explica en detalle todos los puntos de la aplicación.

Capítulo 1: Introducción

1.1 Justificación del TFC

La motivación que me lleva a desarrollar precisamente este tipo de proyecto es la necesidad diaria de una aplicación de estas características en mi puesto de trabajo, ya que actualmente todo este proceso de control es manual y conlleva a posibles errores en la gestión de parte de mi trabajo.

Tambien cabe mencionar el hecho que el TFC esté enfocado profesionalmente a mi trabajo laboral dará lugar a tener en cuenta hasta el último detalle, a una metodologia más precisa y todo lo próximo posible a la realidad y a las necesidades de una empresa real.

1.2 Contexto en el que se desarrolla el TFC

Tal y como he comentado en el prólogo y en el punto anterior, el desarrollo de este TFC será en un contexto aplicable a la empresa PLASTICOS CASTELLA S.A. y más concretamente, destinado a mi puesto de trabajo en el departamento informático de dicha empresa. Entendiendo por lo tanto que la realización de este tiene una cobertura real de una necesidad de trabajo existente.

1.3 Punto de partida y aportación del TFC

Actualmente existe una pequeña base de datos access con la que gestionamos el recambio de consumibles y otra base de datos, en access también en la que se intenta llevar un control de las incidencias y reparaciones. El problema principal es la costosidad de introducir todos estos datos a mano, ya que a veces el olvido hace que no se mantengan las fuentes de información actualizadas.

Disponemos tambien de un portal en nuestra red interna de la empresa de libre distribución. Se trata exactamente de un gestor de contenidos en php llamado *Joomla*, de libre distribución y costumizado para algunas aplicaciones de nuestra empresa. Seria interesante también la integración entre este y la aplicación fruto de este proyecto.

1.4 Objetivos del TFC

El objetivo principal es la automatización de ciertos procesos de trabajo. De esta manera la empresa reduce costes al mismo tiempo que se obtiene un registro actualizado en todo momento del estado de equipos, consumibles, periféricos y registros de incidencias.

De este modo el simple hecho de pedir un recambio mediante una aplicación correctamente estructurada y programada hará que no solo signifique una solicitud sino que queden registradas al mismo tiempo una serie de acciones que ahorraran trabajo al departamento informático.

1.4.1 Pongamos un ejemplo:

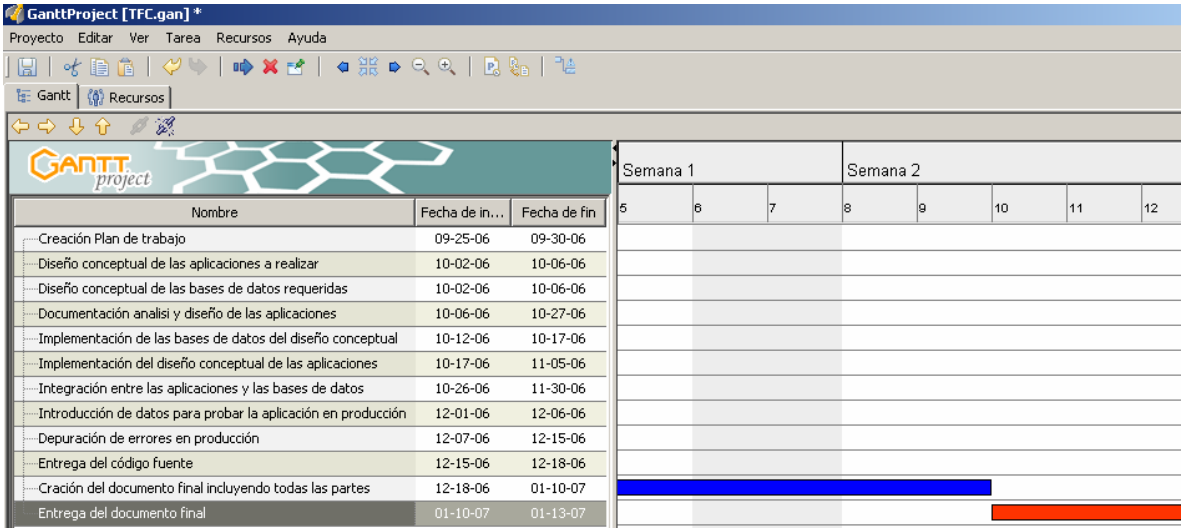
Actualmente un trabajador del departamento de finanzas sol.licita al departamento informático, via telefónica, un tonner para su impresora, facilitando el módelo de esta. En este caso, yo le facilito el tonner, a continuación anoto en una pequeña base de datos en access los datos del recambio y tengo presente para el próximo pedido de consumibles que debo repostar-lo y despues de depositar el tonner usado en su contenedor homologado de reciclaje, anotar-lo para disponer de un albaran en el momento de su recogida con todos los consumibles usados que se llevan.

Con esta aplicación el administrador se conectará al portal, después de introducir su nombre de usuario y procederá a seleccionar el modelo que precise y sol.licitará el recambio del consumible, este proceso se mostrarà la sesión que visualizará el administrador que en este caso corresponderá a los encargados del departamento informático.

Aparantemente este proceso parece que no nos ha liberado trabajo alguno, pero en el siguiente punto observamos con detenimiento todas las ventajas de este ejemplo.

1.5 Planificación del proyecto

Para la realización del *planning* del proyecto he utilizado el GANT PROJECT



Los plazos aconsejables iniciales son los siguientes:

Nombre	Fecha de in...	Fecha de fin
Creación Plan de trabajo	09-25-06	09-30-06
Diseño conceptual de las aplicaciones a realizar	10-02-06	10-06-06
Diseño conceptual de las bases de datos requeridas	10-02-06	10-06-06
Documentación analisi y diseño de las aplicaciones	10-06-06	10-27-06
Entrega de la documentación anterior	10-27-06	10-30-06
Implementación de las bases de datos del diseño conceptual	10-12-06	10-17-06
Implementación del diseño conceptual de las aplicaciones	10-17-06	11-05-06
Integración entre las aplicaciones y las bases de datos	10-26-06	11-30-06
Introducción de datos para probar la aplicación en producción	12-01-06	12-06-06
Depuración de errores en producción	12-07-06	12-15-06
Entrega del código fuente	12-15-06	12-18-06
Cración del documento final incluyendo todas las partes	12-18-06	01-10-07
Entrega del documento final	01-10-07	01-13-07

1.5.1 Diseño conceptual de las aplicaciones a realizar

Esbozo y diseño sobre papel de la estructura y dinámica de la interficie o interficies de las aplicaciones.

1.5.2 Diseño conceptual de las bases de datos requeridas

Esbozo y diseño de los modelos relacionales de las bases de datos que necesitaremos para nuestra aplicación.

1.5.3 Documentación analisi y diseño de las aplicaciones

Deberemos documentar los puntos anteriores y los dos puntos siguientes fruto de la implementación de los diseños conceptuales.

1.5.4 Implementación de las bases de datos del diseño conceptual

Implementaremos las bases de datos que hemos obtenido de los modelos relacionales del punto 1.5.2
La idea inicial es implementar estas bases de datos con *mysql,* que es el gestor de bases de datos que la empresa está empezando a tener en funcionamiento.

1.5.5 Implementación del diseño conceptual de las aplicaciones

Programación del diseño conceptual del punto 1.5.1 mediante el lenguaje de programación J2EE.

1.5.6 Integración entre las aplicaciones y las bases de datos

Programación de las partes que requieren acceso a las bases de datos.

1.5.7 Introducción de datos para probar la aplicación en producción

Introducción de todos los datos necesarios para poner la aplicación en producción y proceder posteriormente a su depuración.

1.5.8 Depuración de errores en producción

Una vez introducidos los datos del punto 1.5.7 testearemos que todo funcione correctamente y depuraremos los errores.

Los otros puntos referentes a las entregas ya quedan claros con la misma descripción.

1.6. Productos obtenidos

A continuación se explican brevemente las aplicaciones principales a las que se desea llegar en la conclusión de este proyecto.

1.6.1. Gestión de recambios:

Solicitud de recambios de tipo consumible, con su registro en una base de datos mediante la cual se facilitará un control aproximado de la previsión para futuros cambios.

1.6.2. Control de stocks:

A partir de la aplicación anterior y con ayuda de otra aplicación, dispondremos de un control de stocks disponibles con stocks mínimos y stocks aconsejables. Además cuando el administrador lo sol.licite le facilitará el pedido de material necesario para cubrir el stock mínimo mediante un correo electrónico por vía de un servidor SMTP.

1.6.3 Control y gestión de averías:

Aplicación mediante la cual los usuarios solicitaran asistencia técnica. Dicha solicitud entrará en un proceso de cola y será atendida por el personal del departamento informático.

Capítulo 2: Modelado del negocio

2.1 Introducción

Para conseguir el objetivo de esta aplicación, he organizado la activadad de este departamento de la empresa por medio de un conjunto de procesos de negocio. Cada uno de estos se caracteriza por una colección de datos que son producidos y manipulados mediante un conjunto de tareas, en las que ciertos agentes, que en este caso son los tecnicos del departamento informático y los trabajadores de la empresa que disponen de equipos informáticos participan de acuerdo a un flujo de trabajo determinado, el cual está sujeto a un conjunto de reglas de negocio, que determinan las políticas y la estructura de la información de dicho departamento. Por lo tanto, la finalidad de este modelado de negocio es describir cada proceso del negocio, especificando sus datos, actividades (o tareas), roles (o agentes) y reglas de negocio.

De este modo en el primer paso del modelado del negocio capturaré los procesos de negocio del departamento bajo estudio a partir de los objetivos principales de este, paso que será crucial debido a que define los límites del proceso de modelado posterior. Teniendo en cuenta que podemos obtener objetivos muy complejos, habrá casos que serán descompuestos en subobjetivos más concretos que a la vez, de ser necesario, podrian ser descompuestos en otra, consiguiendo de este modo una jerarquía de objetivos.

A partir de este punto ilustraré todo el proceso mediante el objetivo del departamento informático de PLÁSTICOS CASTELLA para la aplicación mencionada.

Los objetivos estratégicos del departamento podrían incluir:

- Satisfacer una solicitud de recambio de un trabajador.
  Éste al mismo tiempo puede ser dividido en subobjetivos tales como:
    - Registrar solicitud de recambio.
    - Gestionar almacén de recambios.
    - Realizar pedidos a proveedores de recambios.

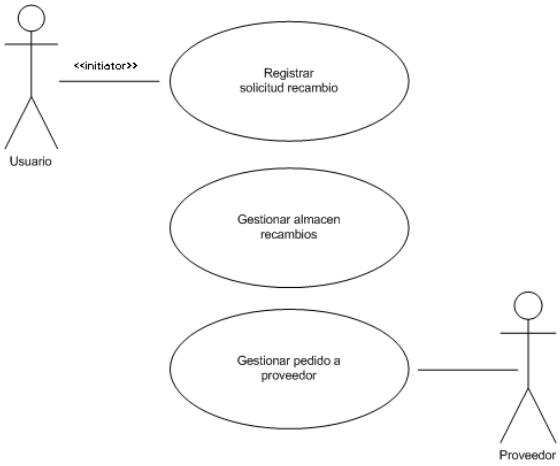
- Satisfacer una solicitud de mejora de equipo o reparación de éste.
  El cual al mismo tiempo puede ser dividido en subobjetivos tales como:
    - Registrar solicitud de mejora o reparación.
    - Gestionar historial del equipo en cuestión.
    - Realizar pedidos a proveedores si es necesario o proceder a su reparación.

2.2 Identificación de Roles del Entorno del Negocio

Una vez identificados los procesos de negocio, es cuando debemos encontrar los agentes involucrados en su realización. Cada uno de estos agentes o también conocidos como actores desempeñan un papel o rol cuando colaboran con otros para llevar a cabo las actividades que conforman dicho caso de uso del negocio. Al tratarse de un caso real, identificaré los roles que son jugados por agentes de la propia empresa (incluyendo trabajadores, departamentos y dispositivos físicos) o agentes externos (como proveedores u otros sistemas).

Para una visión general de los diferentes procesos de negocio del departamento nos centraremos inicialmente en los roles externos al sistema , como son trabajador (ya que al final es el usuario externo del sistema, será como nuestro cliente) y proveedor, claramente externo. Procedo a construir un diagrama de casos de uso del negocio, en el que aparecerá cada proceso del negocio como un caso de uso, mostrando así los límites y el entorno bajo estudio.

En este caso se representa un diagrama de casos de uso UML formado por casos de uso del negocio y actores, donde un trabajador (el usuario del sistema) solicita un recambio (es el que arranca la realización del caso de uso relacionado) mientras que el proveedor simplemente participa en el caso de uso asociado.



2.3 Identificación de los casos de uso del negocio

Como bien sabemos, los casos de uso son procesos autónomos iniciados por un actor o por otro caso de uso. Al mismo tiempo representan las funciones de la aplicación y identifican las entradas y salidas. Debe quedar claro también, que identifican el *que* de éstas funciones y no el *como*.

De este modo se identifican los siguientes casos de uso:

- Login o identificación
- Alta de usuarios
- Baja de usuarios
- Modificación de usuarios
- Alta de equipos y estaciones de trabajo
- Baja de equipos y estaciones de trabajo
- Modificacion de equipos y estaciones de trabajo
- Alta de impresoras y plotters
- Baja de impresoras y plotters
- Modificacion de impresoras y plotters
- Alta consumible
- Baja consumible
- Modificación consumible
- Consulta de stock de consumibles
- Registro de recambios de consumibles
- Registro de reparaciones de equipos
- Solicitud de pedido por falta de stock mínimo en consumibles

Posteriormente se incorporarán los siguientes casos de uso de interactividad con el usuario:

- Solicitar recambio
- Solicitar mejora o reparación
- Consultar estado solicitud de recambio
- Consultar estado de mejora o reparación

2.4 Identificación de las relaciones entre casos de uso

2.4.1 Relaciones de extensión (extend)

Las usaremos cuando un caso de uso base incorpora implícitamente el comportamiento de otro caso de uso en el lugar especificado indirectamente por el caso de uso que extiende al base.

Estas relaciones estan sujetas a una condición cuando existen o bien diferentes flujos de procesos posibles, o bien casos especiales o errores que deben de ser tratados de un modo especial.

● En este proyecto tenemos dos relaciones de extensión entre los casos de uso "Alta de usuarios" y "Baja de usuarios" y "Modificación de usuarios". El motivo de dicha relación de extensión es obvio ya que si no hemos dado de alta a un usuario no lo podremos eliminar ni modificar.

● Por otro lado tenemos otras dos relaciones de extensión entre los casos de uso "Alta de equipos y estaciones de trabajo" y "Baja de equipos y estaciones de trabajo" y "Modificación de equipos y estaciones de trabajo". El motivo de dicha relación es igual al de la anterior.

● Tenmos también otras dos relaciones entre los casos de uso "Alta de impresoras y plotters" , "Baja de impresoras y plotters" y "Modificación de impresoras y plotters" comparten las mismas relaciones de extensión y motivos que los dos casos anteriormente mencionados.

● Cabe mencionar ahora la relación de extensión entre "Alta de equipos y estaciones de trabajo" y "Alta de usuarios", debido a que antes de poner el propietario o responsable de un equipo, éste tiene que ser un usuario dado de alta.

● Igual relación de extensión entre "Alta de impresoras y plotter" y "Alta de equipos y estaciones de trabajo". En este caso al consultar una impresora debe existir el equipo al que está unido, a excepción de las que funcionan con algun JetDirect y no precisen de ningun equipo informático para su funcionamiento.

● Existe otra relación de extensión entre los casos de uso "Alta consumibles", "Baja de consumibles" y "Modificación de consumibles" por igual motivo que los anteriores mencionados.

● Finalmente mencionar que los casos de uso "Alta consumibles" y "Solicitud de pedido por falta de stock mínimos en consumibles" tienen una relación de extensión que a la misma vez extienden del caso de uso "Alta impresoras y plotters" como es obvio.

En la segunda fase del proyecto con interacción con los usuarios tendremos también:

- *Relacion de extensión en "Consultar solicitud de recambio" y "Consultar estado de mejora o reparación" que extienden de los estados de "Solicitud de recambio" y "Solicitud de mejora o reparación" respectivamente como es obvio.*

2.4.2 Relaciones de inclusión (include)

Cuando en un caso de uso base incorpora explícitamente el comportamiento de otro caso de uso en el lugar especificado en el caso base.

Se tratan de un recurso para evitar escribir una misma parte del proceso dentro de diversos casos de uso. Por lo tanto en este caso tendremos tantas relaciones de inclusión como casos de uso derivados del proceso de identificación o "login", para que tanto administradores como usuarios puedan efectuar sus operaciones una vez identificados en el sistema.

2.4.3 Relaciones de generalización

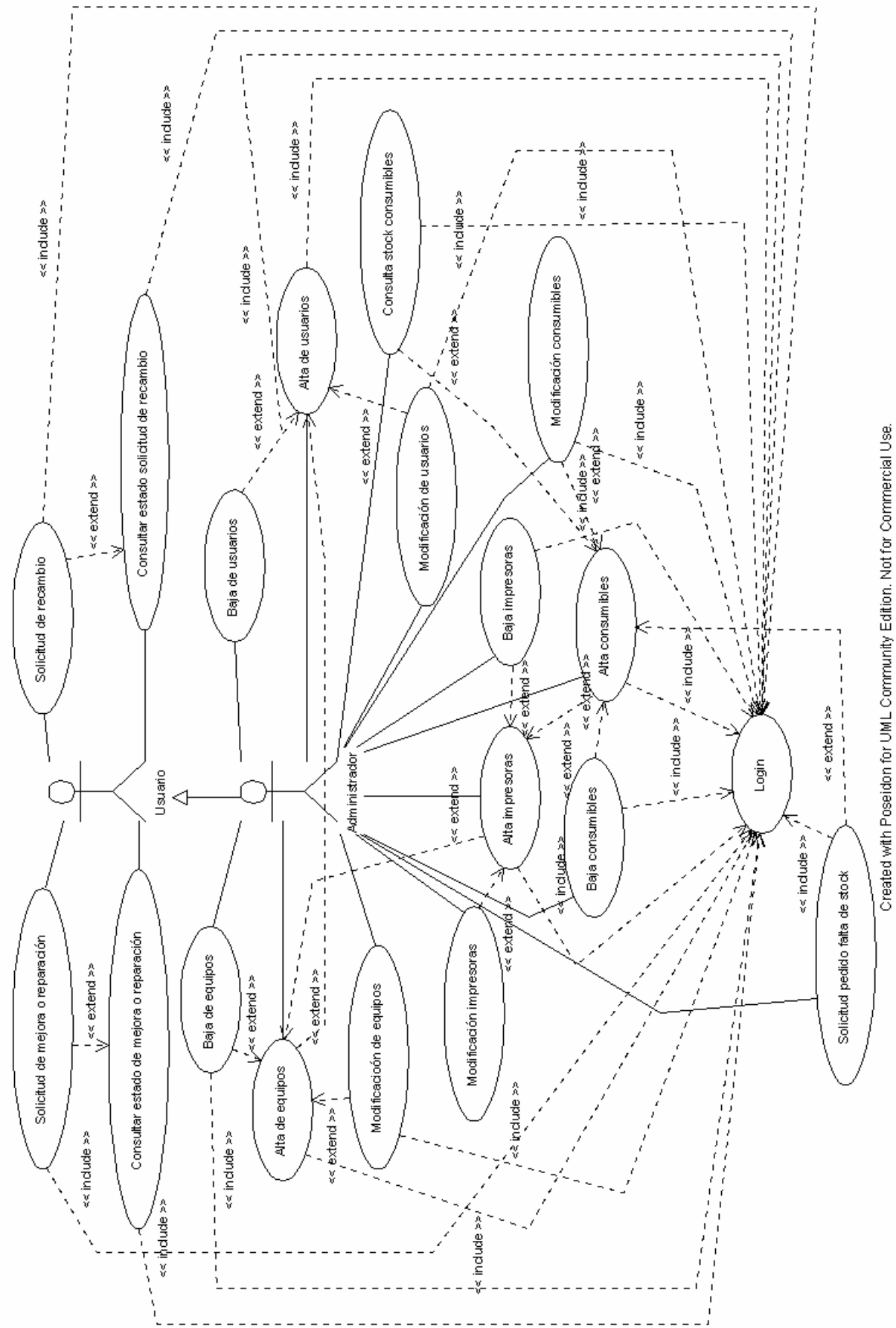
Estas relaciones se identifican cuando el caso de uso hijo hereda el comportamiento y el significado del caso de uso padre, donde el hijo puede agregar o redefinir el comportamiento del padre.

En este proyecto, un caso de generalización es claramente el caso de uso comprovar la clave del login.

2.5 Diagrama de los casos de uso del negocio

A continuación procedemos a la representación del diagrama de los casos de uso teniendo en cuentas los siguientes puntos:

- Las relaciones de generalización entre casos de uso las representaremos como una línea continua con una punta de flecha vacía.
- Las relaciones de extensión las representaremos com una dependencia con la palabra extend.
- Finalmente las relaciones de inclusión las representaremos como una dependencia, usando la palabra include seguido del caso de uso que se quiere incluir.


Usuario
Administrador
Solicitud de recambio
Consultar estado solicitud de recambio
Solicitud de mejora o reparación
Consultar estado de mejora o reparación
Baja de usuarios
Alta de usuarios
Baja de equipos
Alta de equipos
Consulta stock consumibles
Modificación de usuarios
Modificación consumibles
Baja impresoras
Alta impresoras
Alta consumibles
Baja consumibles
Modificacioón de equipos
Modificación impresoras
Login
Solicitud pedido falta de stock
<< include >>
<< extend >>
Created with Poseidon for UML Community Edition. Not for Commercial Use.

2.6 Descripción de los Casos de Uso del Negocio más representativos

En el siguiente paso dentro del modelado del negocio nos introduciremos en cada uno de los casos de uso del negocio identificador, para describirlo en detalle. A continuación, hemos de determinar los agentes internos que juegan un rol en cada caso de uso. Será necesario estudiar la descripción de cada caso y observar el conjunto completo de roles involucrados, tanto externos como internos de la organización que en este caso como ya bien sabemos se trata del departamento informático de PLÁSTICOS CASTELLA.

2.6.1 Caso de uso "Alta de usuarios"

Funcionalidad general	El Administrador podrá dar de alta a los usuarios en la base de datos.
Papel dentro del trabajo del usuario	Es uno de los casos principales del Administrador.
Actores	Administrador.
Casos de uso relacionados	Login
Precondición	El Administrador debe estar registrado y validado por el sistema.
Postcondición	El usuario quedará dado de alta en el sistema.
Descripción	El Administrador entra en la opción de usuarios y dentro de ésta opción entrará en la opción de alta de usuarios para así rellenar los datos y registrarlos en el sistema.

2.6.2 Caso de uso "Baja de usuarios"

Funcionalidad general	El Administrador podrá dar de baja a los usuarios de la base de datos.
Papel dentro del trabajo del usuario	Es uno de los casos de uso ocasionales del Administrador.
Actores	Administrador.
Casos de uso relacionados	Login y Alta de usuarios.
Precondición	El usuario a dar de baja debe de existir en el sistema y el Administrador debe de estar registrado y validado previamente.
Postcondición	El usuario habrá sido dado de baja del sistema.
Descripción	El administrador entra en la opción de usuarios i dentro de ésta pulsará el botón "Eliminar" del usuario que se precise, especificando el motivo de la baja.

2.6.3 Caso de uso "Modificación de usuarios"

Funcionalidad general	El Administrador podrá modificar los datos de un usuario.
Papel dentro del trabajo del usuario	Se trata de otro caso de uso ocasional del Administrador.
Actores	Administrador.
Casos de uso relacionados	Login y Alta usuarios.
Precondición	El Usuario debe de existir en el sistema y el Administrador tiene que estar registrado y validado por el sistema.
Postcondición	Los datos del Usuario quedarán modificados en la base de datos del sistema.
Descripción	El Administrador entra en la opción de usuarios i dentro de ésta pulsará el boton "Modificar" del usuario que se precise y podrá proceder a la modificación de sus datos.

Los anteriores casos de uso són aplicables para el alta, baja y modificación de Equipos, Impresoras y Consumibles, los cuales no he incluido para evitar repeticiones innecesarias en la documentación.

2.6.4 Caso de uso "Consulta de stock de consumibles"

Funcionalidad general	El Administrador podrá visualizar un listado con el stock de los consumibles registrados en el sistema.
Papel dentro del trabajo del usuario	Se trata de un caso de uso principal del Administrador.
Actores	Administrador.
Casos de uso relacionados	Login.
Precondición	El Administrador deberá validarse en el sistema. Además, deberá de haber registrado almenos un consumible para que el listado se pueda efectuar.
Postcondición	El Administrador obtendrá un listado con los stocks de todos los consumibles registrados en el sistema.
Descripción	El Administrador entrará en el menú de consumibles y pulsará el botón "Stock de consumibles" obteniendo así dicho listado.

2.7 Futuro caso de uso

El siguiente caso de uso será operativo en la aplicación a finales de enero, éstos no serán incluidos en la entrega de este trabajo debido a la complejidad de la aplicación y por cuestiones de tiempo de entrega. Pero insisto en el hecho que es de necesaria utilización para la empresa y que estará operativo en breve:

2.7.1 Caso de uso "Solicitud de pedido por falta de stock mínimo en consumibles"

Funcionalidad general	El Administrador podrá solicitar un correo electrónico automático con los consumibles que están por debajo de stock mínimo y las unidades necesarias de cada uno para cubrir el stock aconsejable.
Papel dentro del trabajo del usuario	Es un caso de uso principal del Administrador.
Actores	Administrador.
Casos de uso relacionados	Login.
Precondición	El Administrador deberá validarse en el sistema. Además deberá de haber registrado almenos un consumible para que el proceso se pueda efectuar.
Postcondición	El Administrador recibirá en su correo electrónico un email con los consumibles y unidades necesarias para cubrir el stock aconsejable.
Descripción	El Administrador entrará en el menú de consumibles y pulsará el botón "Solicitud de pedido" obteniendo así dicho correo electrónico.

2.8 Casos de uso opcionales a medio plazo

Los siguientes casos de uso después de reuniones internas del departamento informático de Plásticos Castella, S.A. se ha llegado a la conclusión que no són de necesaria utilización, almenos por el momento, ya que finalmente hemos decido enfocar la aplicación con la finalidad de que sea el administrador del sistema quien la gestione únicamente para un contacto inicial más cercano.
Pues por el momento, és más útil coger las solicitudes por teléfono y que sea el administrador, en éste caso yo, quién la introduzca en el sistema.

2.8.1 Caso de uso "Solicitar recambio"

Donde entrará en juego el rol de usuario en la aplicación

Funcionalidad general	El usuario podrá solicitar un recambio de consumible para su impresora.
Papel dentro del trabajo del usuario	Es uno de los casos de uso principales del Usuario
Actores	Usuario
Casos de uso relacionados	Login.
Precondición	El recambio debe existir y el Usuario validarse en el sistema.
Postcondición	La solicitud queda registrada en el sistema y pasa a manos de un técnico del departamento informático que entregará dicho recambio cuando esté disponible.
Descripción	Cuando un usuario se ha validado en el sistema, éste le mostrará su impresora pudiendo solicitar así alguno de los servicios relacionados.

2.8.2 Caso de uso "Solicitar mejora o reparación"

Funcionalidad general	El usuario podrá solicitar una mejora para su equipo una reparación en caso de averia.
Papel dentro del trabajo del usuario	Es otro de los casos de uso principales del Usuario.
Actores	Usuario
Casos de uso relacionados	Login.
Precondición	El usuario debe validarse en el sistema.
Postcondición	La solicitud queda registrada en el sistema y pasa a manos de un técnico del departamento informático que atenderá la petición. Ademas un historial de registros de averias guardará el histórico de cada máquina.
Descripción	Cuando un usuario se ha validado en el sistema, éste le mostrará su equipo pudiendo solicitar así alguno de los servicios relacionados.

2.8.3 Caso de uso "Consultar estado solicitud de recambio"

Funcionalidad general	El usuario podrá consultar el estado de su solicitud de recambio de consumible.
Papel dentro del trabajo del usuario	Es una caso de uso ocasional del Usuario que extiende del caso de uso Solicitud de recambio.
Actores	Usuario
Casos de uso relacionados	Solicitud de recambio y Login
Precondición	El usuario debe validarse en el sistema y previamente haber hecho una solicitud de recambio.
Postcondición	El sistema mostrará al usuario el estado de su solicitud de recambio, que podrá ser: - Pendiente proveedor - Pendiente entrega - Entregado
Descripción	Cuando el usuario pulse sobre el estado de su solicitud para un recambio en concreto, el sistema le mostrará el estado de éste.

2.8.4 Caso de uso "Consultar estado de mejora o reparación"

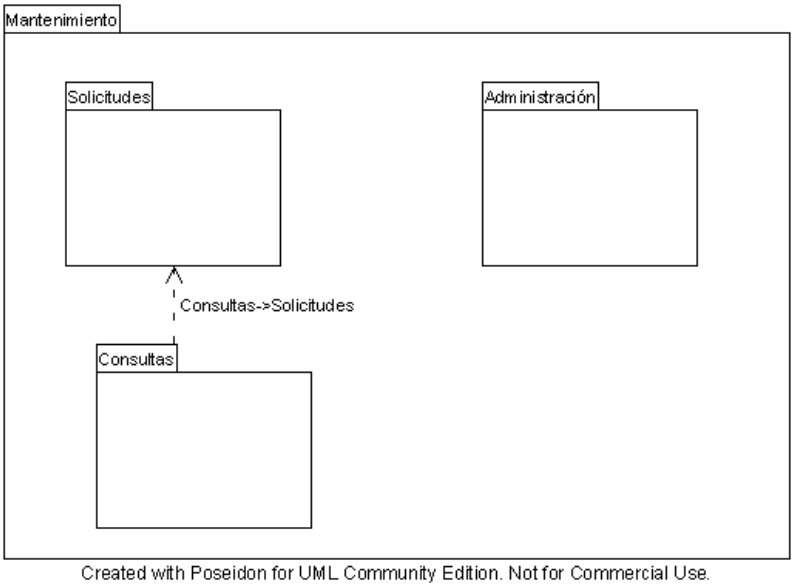
Funcionalidad general	El usuario podrá consultar el estado de su solicitud de mejora de equipo o reparación.
Papel dentro del trabajo del usuario	Es un caso de uso ocasional del Usuario que extiende del caso de uso Solicitud de mejora reparación.
Actores	Usuario
Casos de uso relacionados	Solicitud de mejora o reparación.
Precondición	El usuario debe validarse en el sistema y previamente haber hecho una solicitud de mejora o reparación de su equipo.
Postcondición	El sistema mostrará al usuario el estado de su solicitud de mejora o reparación, que podrá ser: - Mejora aceptada - Mejora rechazada - Pendiente de reparación - En reparación - Reparado
Descripción	Cuando el usuario pulse sobre el estado de su solicitud de mejora o reparación para su equipo, el sistema le mostrará el estado de éste.

2.9 Diagrama de paquetes o de componentes

Cualquier sistema grande se debe dividir en unidades más pequeñas, de modo que las personas puedan trabajar con una cantidad de información limitada, a la vez y de modo que los equipos de trabajo no interfieran con el trabajo de los otros.

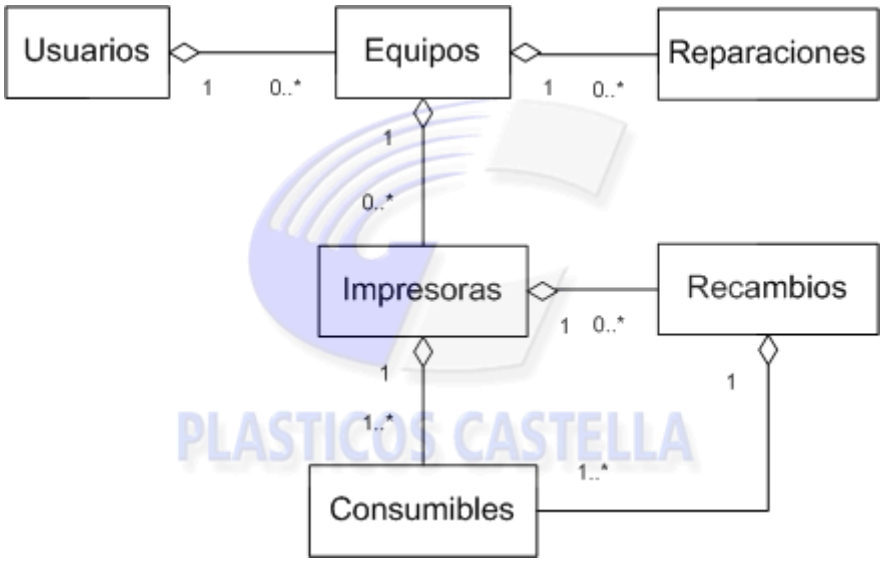
Un paquete es una parte de un modelo. Cada parte del modelo debe pertenecer a un paquete. Pero para ser funcional, la asignación debe seguir un cierto principio racional, tal como funcionalidad común, implementación relacionada y punto de vista común.

Los paquetes ofrecen un mecanismo general para la organización de los modelos/subsistemas agrupando elementos de modelado. Cada paquete corresponde a un submodelo (subsistema) del modelo (sistema). Los paquetes son unidades de organización jerárquica de uso general de los modelos de UML. Pueden ser utilizados para el almacenamiento, el control de acceso, la gestión de la configuración y la construcción de bibliotecas que contengan fragmentos reutilizables del modelo.



2.10 Diagrama Conceptual

Antes de definir el modelo estático o de clases, es necesario definir el Modelo Conceptual, el cual nos muestra los conceptos presentes en el dominio del problema. Un concepto para este caso, en términos de la Programación Orientada a Objetos, es un objeto del mundo real; es decir, es la representación de cosas del mundo real y NO de componentes de *software*. En él no se definen operaciones (o métodos); en este modelo se pueden mostrar los conceptos, los atributos de los conceptos (opcionalmente) y la relación o asociación entre ellos. Informalmente podríamos decir que un concepto es una idea, cosa u objeto. Para descubrirlos debemos analizar los sustantivos en las descripciones textuales del dominio del problema, es decir, de la descripción del sistema, de los requerimientos y de los Casos de Uso.



2.11 Especificación de clases

2.11.1 Introducción a la especificación de clases

En este apartado llevaremos a cabo la especificación de las clases del análisis, distinguiendo de este modo los tres tipos de clases de análisis:

- Una clase **entidad** modela la información de larga vida y su comportamiento asociado. Este tipo de clase suele reflejar entidades del mundo real o elementos necesarios para realizar tareas internas al sistema. También se denominan clase dominio, ya que suelen tratar con abstracciones de entidades del mundo real.
- Una clase **frontera** maneja comunicaciones entre el entorno del sistema y el sistema, suelen proporcionar la interfaz del sistema con un usuario o con otro sistema, en general, por tanto, modelan las interfaces del sistema. Cuando se trata de clases que definen la interfaz con otro sistema se refinarán, durante la fase de diseño, para tener en cuenta los protocolos de comunicación elegidos.
- Una clase **control** modela el comportamiento secuenciado específico de uno o varios casos de uso. Se trata de clases que coordinan los eventos necesarios para llevar a cabo el comportamiento que se especifica en el caso de uso, representan su dinámica.

El modelado de las clases de frontera y de control se acostumbran a hacer en la fase de diseño. Por tanto, en ésta parte del proyecto realizaremos únicamente la identificación de las clases de entidad.

2.11.2 Clases de entidad

Estas clases contendrán los objetos básicos que tal y como ya hemos comentado corresponden al mundo real. Habitualmente se guardan en una base de datos que en nuestro proyecto se realizará con MySQL. Suelen ser independientes de la aplicación y generalmente los objetos de una de estas clases no necesitan conocer nada de las otras. Se comunican a través de los objetos de las clases de control. Los objetos de las clases de entidad solo tienen conocimiento de las cosas que pueden hacer por ellos mismos.

Procederemos por tanto a identificar las primeras clases mediante las cuales se podrán especificar los casos de uso en forma de interacciones. Muchas de estas clases pasarán al diseño y a la implementación.

Inicialmente identificaremos las clases de entidad a partir de los casos de uso y para cada caso indicaremos las clases que se localizan.

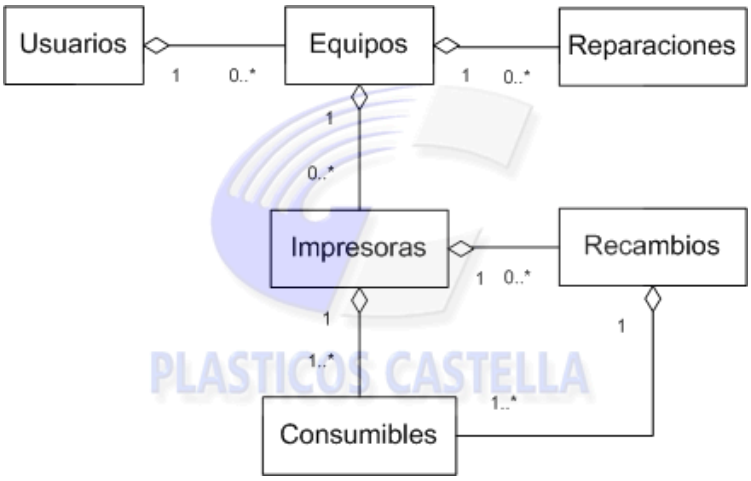
Nº de caso de uso	Nombre	Clases de entidad
1	Login o identificación	Usuarios
2	Alta de usuarios	Usuarios
3	Baja de usuarios	Usuarios
4	Modificación de usuarios	Usuarios
5	Alta de equipos y estaciones de trabajo	Equipos
6	Baja de equipos y estaciones de trabajo	Equipos
7	Modificacion de equipos y estaciones de trabajo	Equipos
8	Alta de impresoras y plotters	Impresoras, Equipos
9	Baja de impresoras y plotters	Impresoras, Equipos
10	Modificacion de impresoras y plotters	Impresoras, Equipos
11	Alta consumible	Consumibles, Impresoras
12	Baja consumible	Consumibles, Impresoras
13	Modificación consumible	Consumibles, Impresoras
14	Consulta de stock de consumibles	Consumibles
15	Solicitud de pedido por falta de stock	Consumibles
16	Solicitar recambio	Usuarios, Consumibles, Recambios
17	Solicitar mejora o reparación	Usuarios, Equipos, Reparaciones
18	Consultar estado solicitud de recambio	Usuarios, Consumibles
19	Consultar estado de mejora o reparación	Usuarios, Equipos, Reparaciones

De este modo la lista inicial de clases de entidad que disponemos és de: Usuarios, Equipos, Impresoras, Consumibles, Recambios y Reparaciones.

Hemos descartado Administrador y Usuario ya que son actores y no son relevantes, tan solo la identificación de los Usuarios será significativa para la aplicación. Es por eso que con una clase Usuarios nos bastará i nos evitará errores en un futuro en el instante de escoger el rol de los usuarios.

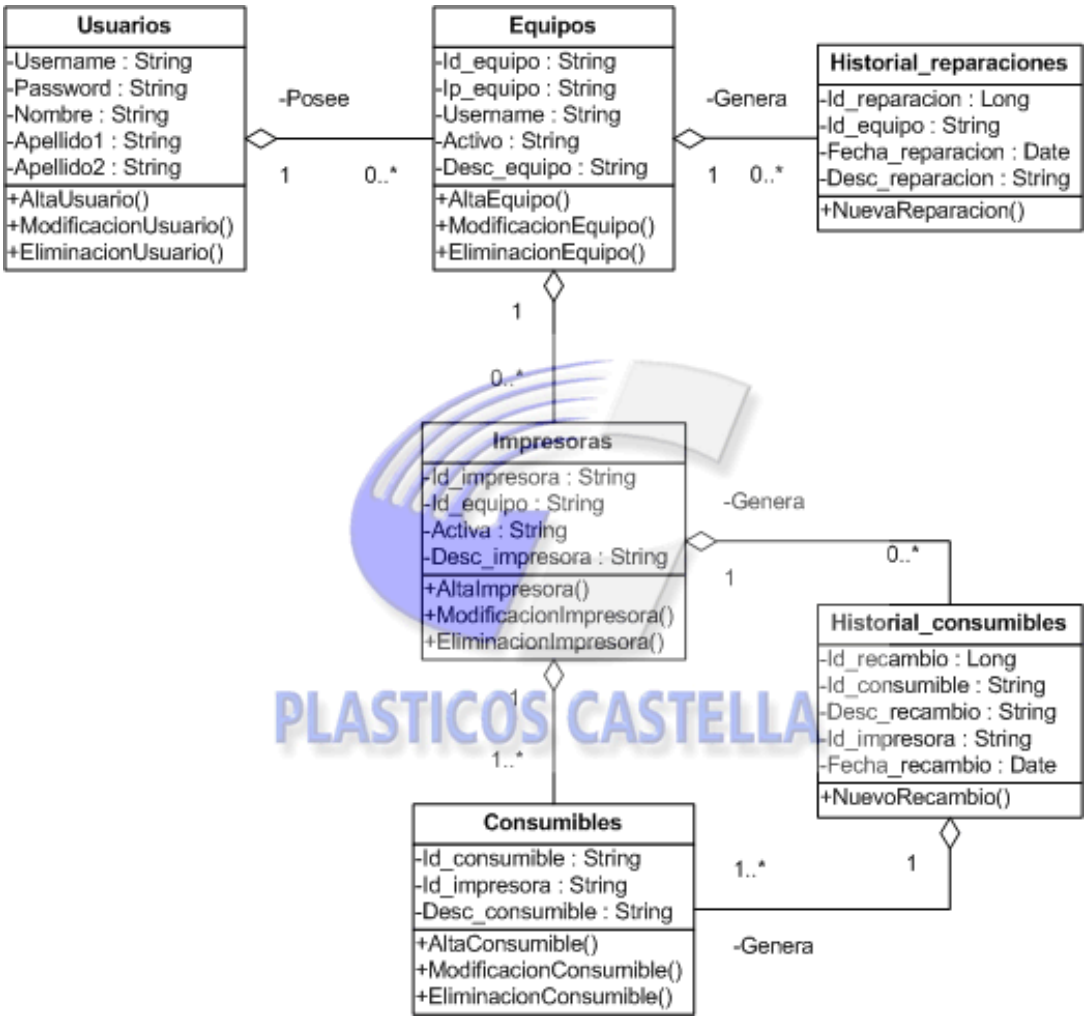
2.12 Diagrama conceptual

Antes de definir el modelo estático o de clases, es necesario definir el modelo conceptual, mostrando así los conceptos presentes en el dominio del problema. En POO, un concepto sería un objeto del mundo real y por lo tanto en él no definiremos operaciones (o métodos) sino que mostraremos los conceptos y las relaciones o asociaciones que existen entre ellos.



2.13 Diagrama de estructura estática o de clases

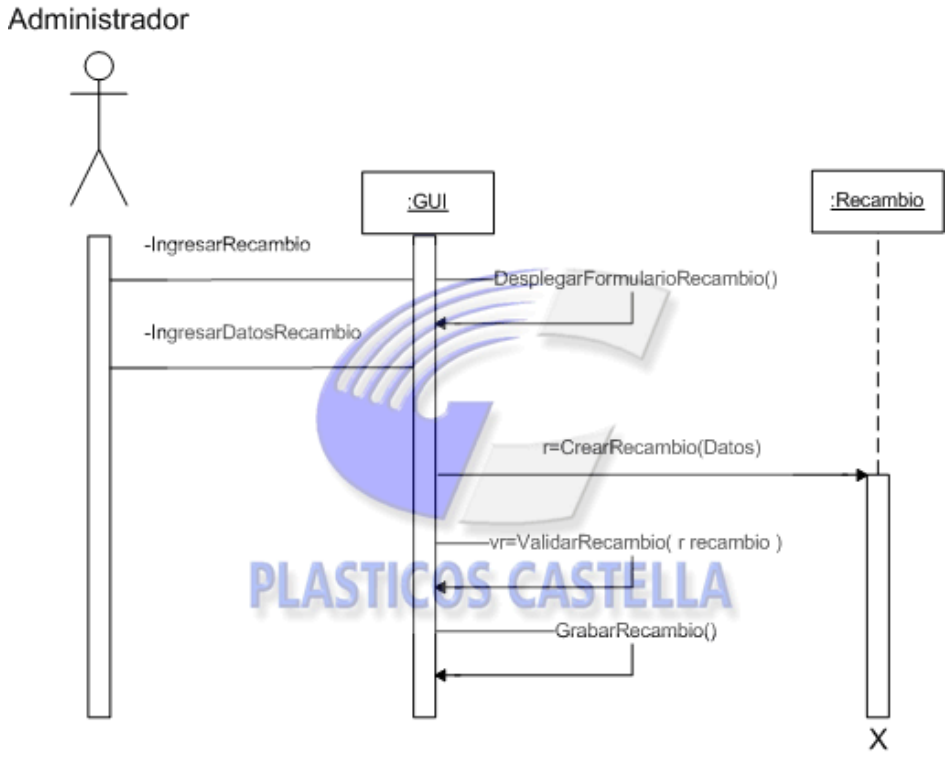
Con éste diagrama pretendemos mostrar una vista de la aplicación en un determinado momento, es decir, en un instante en que el sistema está detenido. Las clases son la plantilla de los objetos, y aquí podemos ver representado a estos con sus atributos o características y su comportamiento o métodos, así como la relación entre ellas.



PLASTICOS CASTELLA

2.14 Diagrama de secuencia

Éste diagrama representa la interacción entre objetos de manera secuencial en el tiempo. Muestra la participación de objetos en la interacción entre sus "lineas de vida" (desde que se instancian) y los mensajes que ellos organizadamente intercambian en el tiempo. El responsable o ACTOR, en nuestro caso el Administrador, es quien inicia el ciclo interactuando inicialmente con la interfaz de usuario: GUI; en seguida se inician todos los objetos que intervienen en el funcionamiento del aplicativo, que en éste primer caso se trata del caso de uso para añadir un movimiento de recambio de consumible en nuestro historial de recambios. Nos damos cuenta que en este diagrama se comienza a observar el comportamiento del sistema a partir de los eventos generados por los actores. Comentar también que aquí se interactúa con instancias, no con clases.



La misma secuencia aplicaremos para el resto de casos de uso de nuestro sistema, tanto el caso de uso de añadir una reparación como los casos de uso de creación, modificación y eliminación de usuarios, equipos, impresoras…

2.15 Diagrama de colaboración

Este tipo de diagrama es útil para identificar objetos, observando así la interacción de un objeto con respecto de los demás. La estructura estática viene dada por los enlaces; la dinámica por el envío de mensajes por los enlaces.

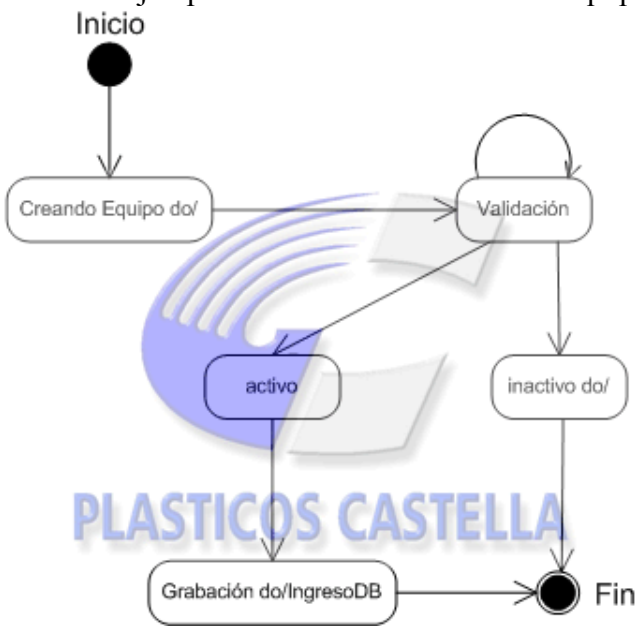
Es decir, representamos una descripción de una colección de objetos que interactúan para implementar un cierto comportamiento dentro de un contexto, describiendo una sociedad de objetos cooperantes unidos para realizar un cierto propósito.

En este caso análizamos desde que un usuario llama al departamento informático solicitando un recambio de consumible hasta que queda registrado en la base de datos y actualizado en el stock de consumibles.

2.16 Diagrama de estados

En este diagrama se muestra la secuencia de los estados de un objeto durante su ciclo de vida. Los estados de los objetos están dados por el valor de su atributos (estados) lo cual cambia sus comportamientos (métodos).

En este caso hemos cogido como ejemplo la creación de un nuevo equipo informático en el sistema:



Capítulo 3: Presentación de la aplicación

Nota: Ésta presentación no incluye las futuras prestaciones del sistema como la *petición de pedido de stock bajo mínimos* ni la parte del usuario que incluye los cosos de:

- Petición de recambios.
- Petición de mejora de su equipo.
- Petición de reparación de su equipo.
- Ver estado de sus peticiones.

3.1 El "login" del administrador

Con esta opción el administrador se identificará ante el sistema facilitando su nombre de usuario y contraseña. El sistema así validará dicho Administrador mostrándole los menús de pantalla correspondientes a la administración del sistema de GMODI.

3.2 El *Front End* del "Administrador"

Una vez validado el Administrador, éste tendrá accesso a los siguientes menús:

3.3 Mantenimiento de usuarios:

Controlará el alta, baja, modificación y listado de usuarios.

En el listado observamos que podemos modificar o eliminar cada uno de los usuarios del sistema (iconos).

Modificación de usuario:

Eliminación de usuario:

Y con el enlace de la parte superior podemos añadir un usuario nuevo:

Para no extendernos de manera repetitiva, la gestión de los siguientes menús utiliza la misma interfície y estilo que los defenidos en el caso anterior de mantenimiento de usuarios:

- Alta, baja, modificación y listado de consumibles.
- Alta, baja, modificación y listado de impresoras.
- Alta, baja, modificación y listado de equipos informáticos.
- Consulta de stocks de consumibles.
- Añadir un nuevo movimiento de recambio en el historial de recambios.
- Listado del historial de recambios.
- Añadir un nuevo movimiento de reparación en el historial de reparaciones.
- Listado del historial de reparaciones.

Capítulo 4: Diseño de la aplicación

En esta parte de la documentación se pretende especificar el diseño arquitectónico de la aplicación basada en JEE. Para ello utilizaremos y justificaremos los patrones arquitectónicos i de diseño que vienen a continuación.

4.1 Patron Value Object

El principio de un Objeto de Valores ("Value Object") es sencillo: agrupar los distintos valores utilizados por el Cliente en un solo método.

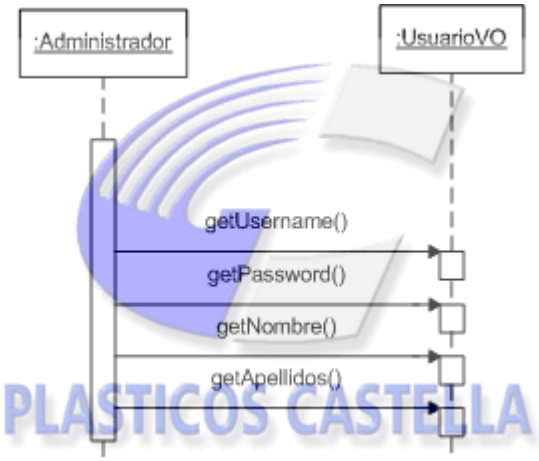
La labor principal de Clientes (JSP/Servlet/Programa de Terminal) para EJB's es invocar métodos presentes en el EJB, sin embargo, como se pudo observar en Interfases Locales estos Clientes no necesariamente residen en el mismo "Application Server/EJB Container", esta residencia trajo como consecuencia el patrón de diseño Value Object.

El llamar método por método presente en distintos EJB's puede presentar una carga substancial sobre la Red y el "Application Server/EJB Container", cada método que es llamado por el Cliente requiere lo siguiente:

- Serializar la información para ser enviada al "EJB Container"
- Deserializar la información para ser procesada en el "EJB Container"
- Revisar parámetros de Seguridad
- Iniciar una Transacción
- Serializar respuesta para ser procesada por el Cliente.
- Deserializar respuesta para utilizarse en el Cliente.

Este mismo proceso siendo llevado acabo 5 o 6 veces consecutivamente, atrevés de un Objeto de Valores ("Value Object") es posible reducir esta carga a una sola llamada.

Estructura del patron en el caso de uso de añadir un equipo, en el que se encapsulan todos los datos requeridos:



PLASTICOS CASTELLA

De esta manera, en el contexto de aplicaciones remotas, se logra una mayor eficiencia, puesto que reduce el envío de mensajes por red. Además, en el contexto de JDBC, permite representar un conjunto de atributos procedentes de uno o varios objetos de dominio, representados por la signatura método (VO) en lugar de método (arg1, arg2, arg3, …).

4.2 Patron Session Façade

Donde las clases Java comunes o EJB encapsulan lógica y datos de negocio, exponiendo sus interfícies y la complejidad de los servicios distribuidos a la capa cliente.

El objetivo de este patron es:

- Proporcionar a los clientes un interfaz sencillo que oculte todas las interacciones complejas entre los componentes de negocio.
- Ocultar al cliente las interacciones y las interdependencias entre los componentes de negocio, permitiendo de esta forma el aumento de flexibilidad y evitar que los cambios en los objetos de negocio repercutan en errores en la vista.
- Evitar la exposición directa de los objetos de negocios a los clientes, para mantener el acoplamiento entre las dos capas al mínimo.
- Centralizar los casos de uso en métodos centralizados sobre una clase.

Aplicándolo conseguiremos usar un session bean (para el caso de aplicaciones EJB) o una clase java común que encapsule la complejidad de las interacciones entre los objetos de negocio participantes en un flujo de trabajo (workflow). El session façade maneja los objetos de negocios y proporciona un servicio de acceso uniforme a los clientes. Es decir, el cliente (JSP en nuestro caso) no tratará con los EJB ni con la complejidad del acceso remoto, ni con JDBC, sino que trabajará con colecciones y Value's Object.

Resumiendo, con Session Façade se obtiene:

- Introducir una capa extra entre el Modelo y el Controlador: Agregando esta capa, se puede modelar los casos de uso de una forma centralizada, logrando también centralizar todo el acoplamiento a esta capa intermedia.
- Expone un interfaz uniforme para el acceso al Modelo.
- Cuando se utiliza con EJB, reduce el número de invocaciones remotas, aumentando la escalabilidad.
- Simplifica la gestión de transacciones y el mantenimiento de los casos de uso.

4.3 Patron Data Access Object

Utilizar un Data Access Object (DAO) nos será muy útil para abstraer y encapsular todos los accesos a la fuente de datos, logrando así desacoplar la lógica de negocios de la lógica de acceso a datos. El DAO maneja la conexión con la fuente de datos para obtener y almacenar datos.

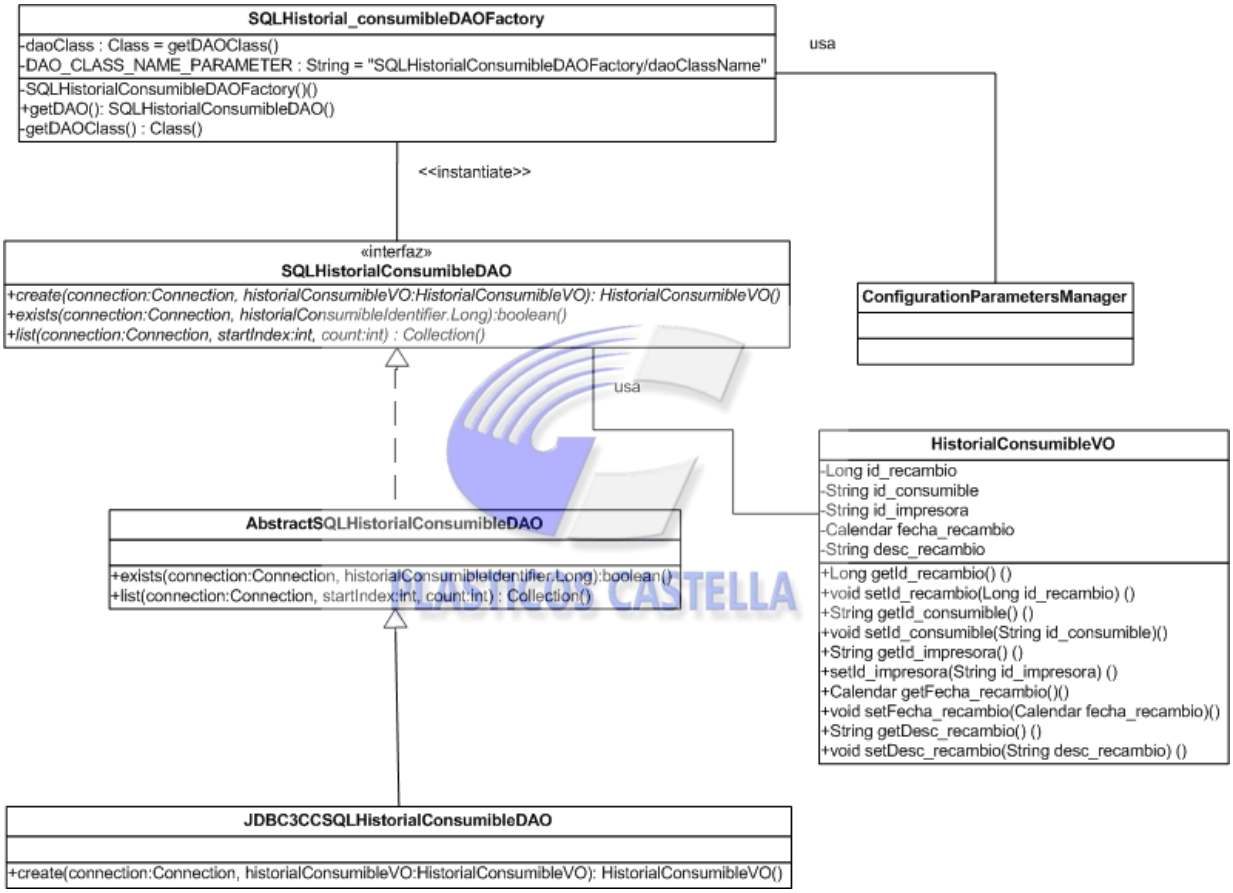
Este patron nos será muy util ya que nos:

- **Permite la Transpariencia**
  Los objetos de negocio puede utilizar la fuente de datos sin conocer los detalles específicos de su implementación. El acceso es transparente porque los detalles de la implementación se ocultan dentro del DAO.
- **Permite una Migración más Fácil**
  Una capa de DAOs hace más fácil que una aplicación pueda migrar a una implementación de base de datos diferente. Los objetos de negocio no conocen la implementación de datos subyacente, la migración implica cambios sólo en la capa DAO. Además, si se emplea la estrategia de factorías, es posible proporcionar una implementación de factorías concretas por cada implementación del almacenamiento subyacente. En este caso, la migración a un almacenamiento diferente significa proporcionar a la aplicación una nueva implementación de la factoría.
- **Reduce la Complejidad del Código de los Objetos de Negocio**
  Como los DAOs manejan todas las complejidades del acceso a los datos, se simplifica el código de los objetos de negocio y de otros clientes que utilizan los DAOs. Todo el código relacionado con la implementación (como las sentencias SQL) están dentro del DAO y no en el objeto de negocio. Esto mejora la lectura del código y la productividad del desarrollo.
- **Centraliza Todos los Accesos a Datos en un Capa Independiente**
  Como todas las operaciones de acceso a los datos se ha delegado en los DAOs, esto se puede ver como una capa que aisla el resto de la aplicación de la implementación de acceso a los datos. Esta centralización hace que la aplicación sea más sencilla de mantener y de manejar.

Aunque en contra nos encontraremos con los siguientes puntos:

- **No es útil para Persistencia Manejada por el Contenedor**
  Como el contenedor EJB maneja los beans de entidad con persistencia manejada por el contenedor (CMP), sirve automáticamente todo el acceso al almacenamiento persistente. Las aplicación que utilizan este tipo de beans no necesitan la capa DAO, ya que el servidor de aplicaciones proporciona de forma transparente esta funcionalidad. Sin embargo, los DAOS aún son útiles cuando se necesita una combinación de CMP (para beans de entidad) y BMP (para beans de sesión, servlets).
- **Añade una Capa Extra**
  Los DAOs crean un capa de objetos adicional entre el cliente y la fuente de datos que necesitamos diseñar e implementar para obtener los beneficios de este patrón. Pero para obtener estos beneficios debemos pagarlos con un esfuerzo adicional.
- **Necesita Diseñar un Árbol de Clases**
  Cuando se utiliza una estrategia de factorías, necesitamos diseñar e implementar el árbol de factorías concretas y el árbol de productos concretos producidos por las factorías. Necesitamos justificar este esfuerzo adicional para ver si merece la pena dicha flexibilidad. Esto incrementa la complejidad del diseño. Sin embargo, podemos elegir la implementación de la estrategia de factorías empezando primero con el patrón **Factory Method**, y luego avanzar hasta el patrón **Abstract Factory** si es necesario.

Estructura del patrón para nuestro caso de uso en el que añadimos un nuevo recambio de consumible en el historial de consumibles:



4.4 Patron Modelo Vista Controlador

El MVC divide una aplicación interactiva en 3 áreas: procesamiento, salida y entrada. Para esto, utiliza las siguientes abstracciones:

- **Modelo (Model)**: Encapsula los datos y las funcionalidades. El modelo es independiente de cualquier representación de salida y/o comportamiento de entrada.
- **Vista (View)**: Muestra la información al usuario. Obtiene los datos del modelo. Pueden existir múltiples vistas del modelo. Cada vista tiene asociado un componente controlador.
- **Controlador (Controller)**: Reciben las entradas, usualmente como eventos que codifican los movimientos o pulsación de botones del ratón, pulsaciones de teclas, etc. Los eventos son traducidos a solicitudes de servicio ("**service requests**" en el texto original) para el modelo o la vista. El usuario interactúa con el sistema a través de los controladores.

Las Vistas y los Controladores conforman la interfaz de usuario. Un mecanismo de propagación de cambios asegura la consistencia entre la interfaz y el modelo. La separación del modelo de los componentes vista y del controlador permite tener múltiples vistas del mismo modelo. Si el usuario cambia el modelo a través del controlador de una vista, todas las otras vistas dependientes deben reflejar los cambios. Por lo tanto, el modelo notifica a todas las vistas siempre que sus datos cambien. Las vistas, en cambio, recuperan los nuevos datos del modelo y actualizan la información que muestran al usuario.

Algunos de sus principales beneficios son:

- Menor acoplamiento
  - Desacopla las vistas de los modelos
  - Desacopla los modelos de la forma en que se muestran e ingresan los datos
- Mayor cohesión
  - Cada elemento del patrón esta altamente especializado en su tarea (la vista en mostrar datos al usuario, el controlador en las entradas y el modelo en su objetivo de negocio)
- Las vistas proveen mayor flexibilidad y agilidad
  - Se puede crear múltiples vistas de un modelo
  - Se puede crear, añadir, modificar y eliminar nuevas vistas dinámicamente
  - Las vistas pueden anidarse
  - Se puede cambiar el modo en que una vista responde al usuario sin cambiar su representación visual
  - Se puede sincronizar las vistas
  - Las vistas pueden concentrarse en diferentes aspectos del modelo

- Mayor facilidad para el desarrollo de clientes ricos en múltiples dispositivos y canales
    - Una vista para cada dispositivo que puede varias según sus capacidades
    - Una vista para la Web y otra para aplicaciones de escritorio
- Más claridad de diseño
- Facilita el mantenimiento
- Mayor escalabilidad

Capítulo 5: Implementación de la aplicación

5.1 Software utilizado para la implementación de GMODI

Para la implementación de GMODI se ha utilizado el siguiente software:

1) Máquina de Java j2re-1_4_2_13 como compilador de la aplicación
2) Eclipse-SDK-3.1.2 como IDE para la implementación del proyecto
3) JBoss-4.0.4.GA como servidor de EJB
4) JBossIDE-1.6.0.GA-ALL como plug-in de Eclipse que da soporte para JBoss
5) Struts 1.3.5 para la implementación del MVC
6) Consola com.jamesholmes.console.struts para Eclipse que nos permite gestión visual de archivos de tipo struts-config.xml de este modo nos será mas facil
7) Mysql-4.1.12a como SGBD para almacenar la base de datos de GMODI
8) Mysql-administrator-1.0.20 para la administración de Mysql

5.2 Metodología para la implementación de GMODI

La idea principal para la elaboración de este proyecto ha sido seguir la estructura de aplicaciones profesionales con J2EE. Es decir, se ha seguido un sistema de empaquetage de los objetos para tener una estructura clara y precisa del programa.

Hemos conseguido de este modo tener separados los diferentes DAO y VO según su tipo de Objeto, como puede ser consumible, impresora, equipo, etc.
Tambien hemos conseguido tener así un paquete con el Façade, otros con los MVC, paquetes de utilidades, de configuración, de control de excepciones, etc.

Resumiendo, con esta técnica de empaquetage ha quedado una aplicación muy clara y transparente, que hace que para el programador sea mas facil de mantener y modificar.

5.3 Esquema de la implementación de clases en GMODI

A continuación se muestra un esquema en forma de árbol de los paquetes y clases definidos para la aplicación, se han marcado con diferentes colores para situarse visualmente en los distintos tipos de objetos i clases:

```
ApplicationResources.properties
PatronesConfig.properties
gmodi\src\com\plasticoscastella
 |_gmodi\src\com\plasticoscastella\gmodi
  |_gmodi\src\com\plasticoscastella\gmodi\patrones
   |_gmodi\src\com\plasticoscastella\gmodi\patrones\facade
        |_GmodiFacade.java
        |_TestGmodiFacade.java
   |_gmodi\src\com\plasticoscastella\gmodi\patrones\model
        |_gmodi\src\com\plasticoscastella\gmodi\patrones\model\consumible
          |_gmodi\src\com\plasticoscastella\gmodi\patrones\model\consumible\dao
                |_AbstractSQLConsumibleDAO.java
                |_JDBC3CCSQLConsumibleDAO.java
                |_SQLConsumibleDAO.java
                |_SQLConsumibleDAOFactory.java
        |_gmodi\src\com\plasticoscastella\gmodi\patrones\model\consumible\vo
                |_ConsumibleVO.java
        |_gmodi\src\com\plasticoscastella\gmodi\patrones\model\desc_equipo
          |_gmodi\src\com\plasticoscastella\gmodi\patrones\model\desc_equipo\dao
                |_AbstractSQLDescEquipoDAO.java
                |_JDBC3CCSQLDescEquipoDAO.java
                |_SQLDescEquipoDAO.java
                |_SQLDescEquipoDAOFactory.java
          |_gmodi\src\com\plasticoscastella\gmodi\patrones\model\desc_equipo\vo
                |_DescEquipoVO.java
        |_gmodi\src\com\plasticoscastella\gmodi\patrones\model\equipo
          |_gmodi\src\com\plasticoscastella\gmodi\patrones\model\equipo\dao
                |_AbstractSQLEquipoDAO.java
                |_JDBC3CCSQLEquipoDAO.java
                |_SQLEquipoDAO.java
                |_SQLEquipoDAOFactory.java
          |_gmodi\src\com\plasticoscastella\gmodi\patrones\model\equipo\vo
                |_EquipoVO.java
        |_gmodi\src\com\plasticoscastella\gmodi\patrones\model\historial_consumible
          |_gmodi\src\com\plasticoscastella\gmodi\patrones\model\historial_consumible\dao
                |_AbstractSQLHistorialConsumibleDAO.java
                |_JDBC3CCSQLHistorialConsumibleDAO.java
                |_SQLHistorialConsumibleDAO.java
                |_SQLHistorialConsumibleDAOFactory.java
          |_gmodi\src\com\plasticoscastella\gmodi\patrones\model\historial_consumible\vo
                |_HistorialConsumibleVO.java
```

|_gmodi\src\com\plasticoscastella\gmodi\patrones\model\historial_reparacion
|_gmodi\src\com\plasticoscastella\gmodi\patrones\model\historial_reparacion\dao
|_AbstractSQLHistorialReparacionDAO.java
|_JDBC3CCSQLHistorialReparacionDAO.java
|_SQLHistorialReparacionDAO.java
|_SQLHistorialReparacionDAOFactory.java
|_gmodi\src\com\plasticoscastella\gmodi\patrones\model\historial_reparacion\vo
|_HistorialReparacionVO.java
|_gmodi\src\com\plasticoscastella\gmodi\patrones\model\impresora
|_gmodi\src\com\plasticoscastella\gmodi\patrones\model\impresora\dao
|_AbstractSQLImpresoraDAO.java
|_JDBC3CCSQLImpresoraDAO.java
|_SQLImpresoraDAO.java
|_SQLImpresoraDAOFactory.java
|_gmodi\src\com\plasticoscastella\gmodi\patrones\model\impresora\vo
|_ImpresoraVO.java
|_gmodi\src\com\plasticoscastella\gmodi\patrones\model\stock_consumible
|_gmodi\src\com\plasticoscastella\gmodi\patrones\model\stock_consumible\dao
|_AbstractSQLStockConsumibleDAO.java
|_JDBC3CCSQLStockConsumibleDAO.java
|_SQLStockConsumibleDAO.java
|_SQLStockConsumibleDAOFactory.java
|_gmodi\src\com\plasticoscastella\gmodi\patrones\model\stock_consumible\vo
|_StockConsumibleVO.java
|_gmodi\src\com\plasticoscastella\gmodi\patrones\model\usuario
|_gmodi\src\com\plasticoscastella\gmodi\patrones\model\usuario\dao
|_AbstractSQLUsuarioDAO.java
|_JDBC3CCSQLUsuarioDAO.java
|_SQLUsuarioDAO.java
|_SQLUsuarioDAOFactory.java
|_gmodi\src\com\plasticoscastella\gmodi\patrones\model\usuario\vo
|_UsuarioVO.java
|_gmodi\src\com\plasticoscastella\gmodi\patrones\struts
|_gmodi\src\com\plasticoscastella\gmodi\patrones\struts\action
|_gmodi\src\com\plasticoscastella\gmodi\patrones\struts\action\administrador
|_ConsumiblesEditAction.java
|_ConsumiblesEditSaveAction.java
|_ConsumiblesListAction.java
|_ConsumiblesNewAction.java
|_ConsumiblesNewSaveAction.java
|_ConsumiblesRemoveAction.java
|_ConsumiblesRemoveSaveAction.java
|_DescEquiposDetailAction.java
|_DescEquiposDetailEndAction.java
|_DescEquiposEditAction.java
|_DescEquiposEditSaveAction.java
|_DescEquiposListAction.java
|_DescEquiposNewAction.java
|_DescEquiposNewSaveAction.java
|_DescEquiposRemoveAction.java
|_DescEquiposRemoveSaveAction.java
|_EditAdminAction.java
|_EquiposEditAction.java
|_EquiposEditSaveAction.java
|_EquiposListAction.java
|_EquiposNewAction.java
|_EquiposNewSaveAction.java
|_EquiposRemoveAction.java

|_EquiposRemoveSaveAction.java
|_HistorialConsumiblesListAction.java
|_HistorialConsumiblesNewAction.java
|_HistorialConsumiblesNewSaveAction.java
|_HistorialReparacionesListAction.java
|_HistorialReparacionesNewAction.java
|_HistorialReparacionesNewSaveAction.java
|_ImpresorasEditAction.java
|_ImpresorasEditSaveAction.java
|_ImpresorasListAction.java
|_ImpresorasNewAction.java
|_ImpresorasNewSaveAction.java
|_ImpresorasRemoveAction.java
|_ImpresorasRemoveSaveAction.java
|_LoginProcessAction.java
|_StockConsumiblesEditAction.java
|_StockConsumiblesEditSaveAction.java
|_StockConsumiblesListAction.java
|_StockConsumiblesNewAction.java
|_StockConsumiblesNewSaveAction.java
|_StockConsumiblesRemoveAction.java
|_StockConsumiblesRemoveSaveAction.java
|_UsersEditAction.java
|_UsersEditSaveAction.java
|_UsersListAction.java
|_UsersNewAction.java
|_UsersNewSaveAction.java
|_UsersRemoveAction.java
|_UsersRemoveSaveAction.java
|_gmodi\src\com\plasticoscastella\gmodi\patrones\struts\action\cliente
|_gmodi\src\com\plasticoscastella\gmodi\patrones\struts\beans
|_ConsumibleForm.java
|_DescEquipoForm.java
|_EquipoForm.java
|_HistorialConsumibleForm.java
|_HistorialReparacionForm.java
|_ImpresoraForm.java
|_LoginForm.java
|_StockConsumibleForm.java
|_UserForm.java
gmodi\src\com\plasticoscastella\gmodi\util
|_DateOperations.java
|_HttpSessionManager.java
|_ValidationSet.java
|_gmodi\src\com\plasticoscastella\gmodi\util\configuration
|_ConfigurationParametersManager.java
|_MissingConfigurationParameterException.java
|_gmodi\src\com\plasticoscastella\gmodi\util\exceptions
|_CalendarMalFormedException.java
|_DataBaseException.java
|_ExceptionFormatMessage.java
|_InstanceException.java
|_InstanceNotFoundException.java
|_InternalErrorException.java
|_ModelException.java
|_gmodi\src\com\plasticoscastella\gmodi\util\sql
|_GeneralOperations.java
|_SimpleDataSource.java

5.4 Despliegue de la aplicación GMODI

Para poder ejecutar la aplicación GMODI únicamente se necesita tener instalado dos aplicaciones:

- JBoss-4.0.4.GA o superior.
- Mysql-4.1.12a o superior.

Ahora solo necesitamos el archivo "gmodi.war" y lo colocamos en la carpeta "deploy" de nuestro JBoss.

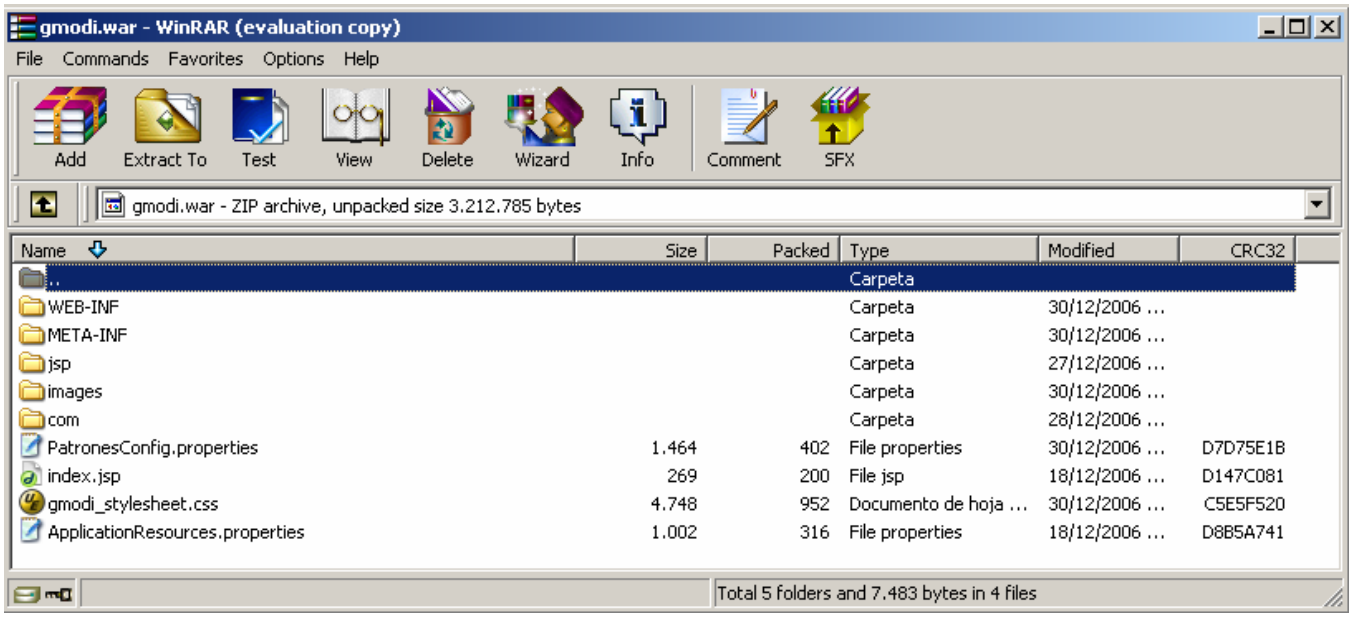
NOTA MUY IMPORTANTE:

GMODI viene configurado para hacer la conexión a la base de datos con el siguiente usuario y contraseña:

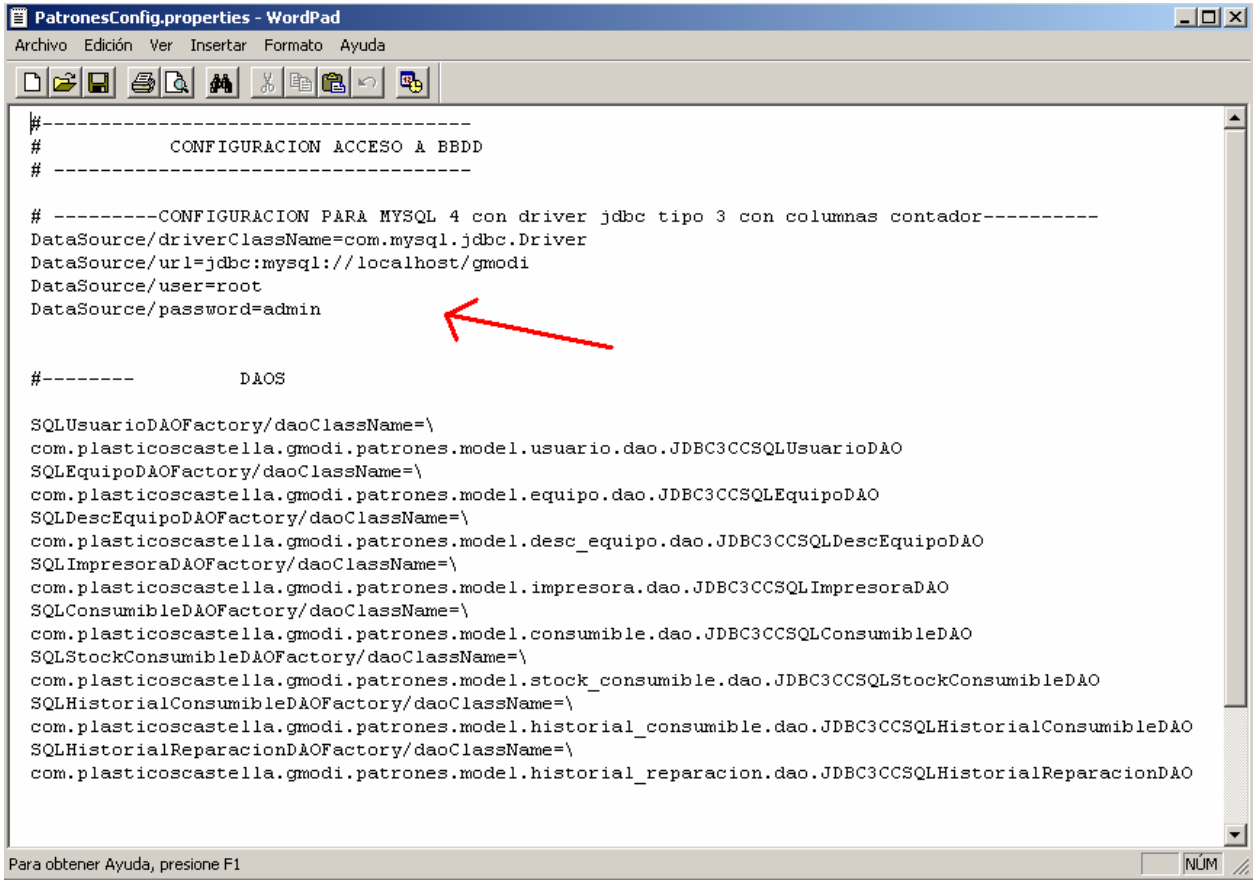
Usuario= "root"
Password = "admin"

Por lo tanto para que la aplicación funcione tenemos dos opciones, o creamos este usuario con permisos en nuestro Mysql o modificamos el fichero de propiedades de conexión a la base de datos de GMODI para Mysql.

Si no se quiere crear el usuario para la conexión, simplemente debemos modificar el fichero de propiedades para la conexión con la base de datos. Para ello, únicamente abriremos el fichero "gmodi.war" con un compresor de archivos, por ejemplo el WinRAR tal y como se muestra en la imagen:



Ahora simplemente editamos el fichero “PatronesConfig.properties” con el usuario y contraseña deseados:



Y ya tenemos configurada la aplicación con nuestro SGBD de Mysql.

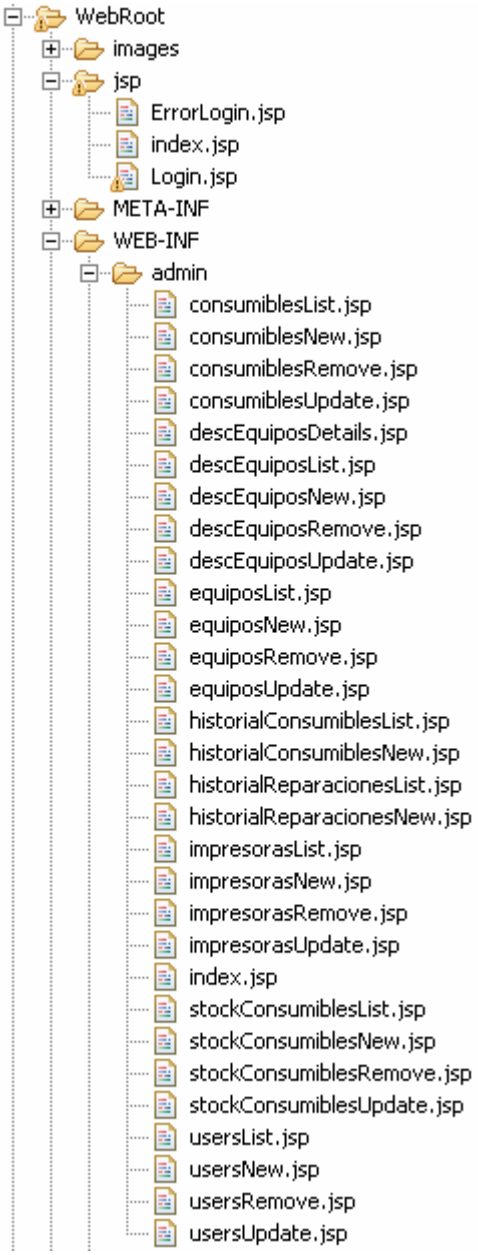
Este fichero de propiedades es muy importante ya que de este modo la configuración de la conexión a la base de datos no se encuentra compilada en las clases de la aplicación y por tanto al ser un fichero de texto plano simplemente con modificarlo la aplicación ya lo carga correctamente.
Si tuvieramos esta configuración en las clases deberiamos modificar el código, compilar y deployar antes de poder-lo utilizar la cual cosa no sería nada óptima ni eficiente.

5.5 Comentarios extras de la aplicación GMODI

5.5.1 Uso de tags

La capa cliente está modelada con Java Server Pages, las cuales manipulan y gestionan los formularios de presentación y modelado de la base de datos.

La estructura de los JSP en GMODI es la siguiente:



Todas estas páginas se gestionan y redireccionan entre ellas gracias al fichero "struts-config.xml" que contiene la definición y ubicación de los form-beans, actions,…
A continuación se muestra solo una parte de este debido al gran volumen que tiene:

```
<!DOCTYPE struts-config PUBLIC "-//Apache Software Foundation//DTD
Struts Configuration 1.1//EN"
"http://jakarta.apache.org/struts/dtds/struts-config_1_1.dtd">
<struts-config>
   <data-sources />
   <form-beans>
   <form-bean name="loginForm"
type="com.plasticoscastella.gmodi.patrones.struts.beans.LoginForm" />
   <form-bean name="userForm"
type="com.plasticoscastella.gmodi.patrones.struts.beans.UserForm" />
   <form-bean name="equipoForm"
type="com.plasticoscastella.gmodi.patrones.struts.beans.EquipoForm" />
   <form-bean name="descEquipoForm"
type="com.plasticoscastella.gmodi.patrones.struts.beans.DescEquipoForm
" />
   <form-bean name="impresoraForm"
type="com.plasticoscastella.gmodi.patrones.struts.beans.ImpresoraForm"
/>
   <form-bean name="consumibleForm"
type="com.plasticoscastella.gmodi.patrones.struts.beans.ConsumibleForm
" />
   <form-bean name="stockConsumibleForm"
type="com.plasticoscastella.gmodi.patrones.struts.beans.StockConsumibl
eForm" />
   <form-bean name="historialConsumibleForm"
type="com.plasticoscastella.gmodi.patrones.struts.beans.HistorialConsu
mibleForm" />
   <form-bean name="historialReparacionForm"
type="com.plasticoscastella.gmodi.patrones.struts.beans.HistorialRepar
acionForm" />
   </form-beans>
   <global-exceptions />

   <global-forwards>
      <forward name="welcome" path="/do/admin" redirect="true" />
   <forward name="login" redirect="false" path="/jsp/Login.jsp"
/></global-forwards>

   <action-mappings>

   <!-- ***************************** CLIENTE
*************************-->

      <action forward="/jsp/index.jsp" path="/default" unknown="true"
/>



   <!-- ************************* ADMINISTRADOR ********************-
->

   <action path="/admin"
type="com.plasticoscastella.gmodi.patrones.struts.action.administrador
.EditAdminAction">
```

```
      <forward name="success" path="/WEB-INF/admin/index.jsp"
redirect="false" />
   </action>

   <action path="/usersList"
type="com.plasticoscastella.gmodi.patrones.struts.action.administrador
.UsersListAction" scope="request">
                  <forward name="success" path="/WEB-
INF/admin/usersList.jsp" />
   </action>

   <action path="/editUser"
type="com.plasticoscastella.gmodi.patrones.struts.action.administrador
.UsersEditAction" name="userForm" validate="false" scope="request">

                  <forward name="success" path="/WEB-
INF/admin/usersUpdate.jsp" />
   </action>

   <action path="/editSaveUser"
type="com.plasticoscastella.gmodi.patrones.struts.action.administrador
.UsersEditSaveAction" name="userForm" validate="true" scope="request"
input="/WEB-INF/admin/usersUpdate.jsp">
                  <forward name="success" path="/WEB-
INF/admin/index.jsp" />
            </action>

   <action path="/removeUser"
type="com.plasticoscastella.gmodi.patrones.struts.action.administrador
.UsersRemoveAction" name="userForm" validate="false" scope="request">

                  <forward name="success" path="/WEB-
INF/admin/usersRemove.jsp" />
   </action>

   <action path="/removeSaveUser"
type="com.plasticoscastella.gmodi.patrones.struts.action.administrador
.UsersRemoveSaveAction" name="userForm" validate="true"
scope="request" input="/WEB-INF/admin/usersRemove.jsp">
                  <forward name="success" path="/WEB-
INF/admin/index.jsp" />
            </action>

   <action path="/newUser"
type="com.plasticoscastella.gmodi.patrones.struts.action.administrador
.UsersNewAction" name="userForm" validate="false" scope="request">
            <forward name="success" path="/WEB-INF/admin/usersNew.jsp"
/>
            </action>

   <action path="/newUserSave"
type="com.plasticoscastella.gmodi.patrones.struts.action.administrador
.UsersNewSaveAction" name="userForm" validate="true" scope="request"
input="/WEB-INF/admin/usersNew.jsp">
                  <forward name="success" path="/WEB-
INF/admin/index.jsp" />
            </action>
```

En estas páginas JSP he hecho uso de los siguientes tags:

```
<%@ taglib prefix="c" uri="http://java.sun.com/jstl/core"  %>
<%@ taglib uri="struts/bean-el" prefix="bean" %>
<%@ taglib uri="struts/html-el" prefix="html" %>
<%@ taglib uri="struts/logic-el" prefix="logic" %>
```

Con ellos he gestionado los atributos de mapeo entre formularios y objetos de clase, tal y como se muestra en los siguientes ejemplos:

```
<td width="20%">Id Reparacion</td>
<td width="80%"><c:out value="${lista.id_reparacion}"/></td>
```

En este ejemplo hemos cogido el valor del atributo de una lista para mostrarlo por el formulario gracias al tag “c:out”

```
<td valign="top">Descripción</td>
<td><html:textarea property="desc_reparacion"
     name="historialReparacionForm" styleClass="campoAzul"/>
```

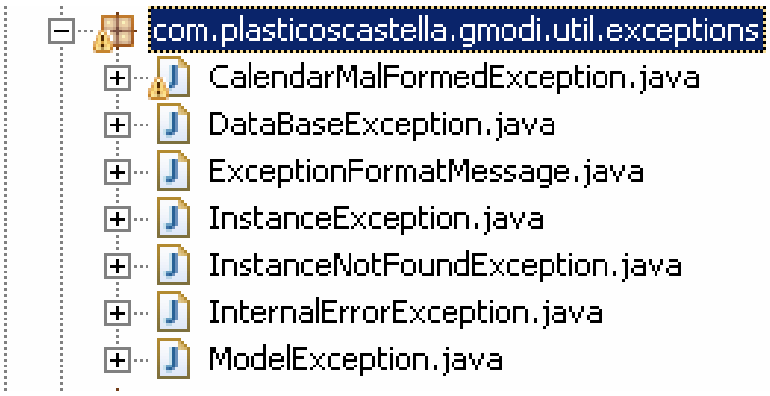
En este otro ejemplo en el que debemos rellenar un formulario gracias al tag “html:textarea” conseguimos que el contenido de la área de texto apunte a un campo determinado de un bean-form determinado, consiguiendo así llenar los datos de un objeto de tipo VO gracias al action form que ejecutamos posteriormente cuando tenemos todos los atributos:

```
<html:form action="newHistorialReparacionSave">
```

El <html:form …> hará que cuando tengamos todos los atributos vaya al “struts-config.xml” a buscar que action form debe ejecutar con estos datos recogidos.

5.5.2 Tratamiento de excepciones

En GMODI he intentado tratar el máximo número de excepciones possibles que puedan suceder, desde las de dentro de las mismas clases como las globales (recogidas en un paquete) que se muestran a continuación:



Además del tratamiento específico de determinadas excepciones controladas en un fichero de propiedades (AplicationResources.properties) en el que se recolectan los mensajes para ser mostrados cuando las excepciones son lanzadas:

```
error.validation.header = Mensaje de error
error.usernamelength = Longitud para el identificador de usuario
incorrecta
error.passwordlength = Longitud para la palabra de paso incorrecta
error.nombrelength = Longitud para el nombre de usuario incorrecta
error.apellido1length = Longitud para el primer apellido no puede se
superior a 15 caracteres
error.apellido2length = Longitud para el segundo apellido no puede se
superior a 15 caracteres
error.activolength = Activo debe ser un caracter, 'Y' o 'N'
error.idequipolength = Longitud para la identificación de equipo
incorrecta
error.ipequipolength = La longitud para la ip del equipo debe ser de
15 digitos contando los puntos
error.idconsumiblelength = Longitud para la identificación del
consumible incorrecta
error.idimpresoralength = Longitud para la identificación de la
impresora incorrecta
error.componentelength = Longitud para el componente incorrecta
error.impresoralength = Longitud para el identificador de impresora
incorrecto
```

5.6 Valoración personal

Llegados a este punto solo me cabe comentar que he obtenido gracias a este trabajo de final de carrera.

Bien, personalmente me ha llenado mucho el hecho que se tratase de una aplicación real de uso profesional para la empresa en la que trabajo. Esto ha hecho que intente dejar atrás las técnicas artesanas de programación e intentar basarme en procesos de ingeniería mucho mas maduros, de este modo conseguir realizar un desarrollo mas productivo y sobre todo que sea de facil absorción para futuros compañeros de mi departamento que puedan necesitar modificar o extender la aplicación.

Soy consciente que todavía falta limar ciertos puntos como el uso de checkbox, combobox,… en los formularios, que hará que el mantenimiento sea mucho mas fácil. En este punto me reitero y no miento cuando digo que estos detalles estarán implementados y en funcionamiento a corto plazo en GMODI para la empresa Plasticos Castella S.A.

No miento tampoco cuando digo que los conocimiento sobre la tecnología J2EE que con los que he partido eran escasos por no decir nulos, cosa que me ha hecho pensar en mas de una ocasión en “tirar la toalla”. Ha sido aquí cuando el papel del consultor me ha servido de gran ayuda ya que Javier Ferro (mi consultor) ha despertado mi motivación respecto el mundo J2EE, usando una técnica que en un principio no entendia ya que ha respondido a casi todas mis preguntas con mas preguntas, pero que después entendí que sus preguntas han sido las que han hecho que me encaminara por el buen camino. Lastima no haber dispuesto de un par de semanas mas para entregar un producto mucho mas maduro.

Pero creo que he conseguido llegar al objetivo de este trabajo de final de carrera con creces y he conseguido superar esta propuesta para introducirnos o profundizar en este apasionado mundo del desarrollo por Internet usando la tecnología Java, concretamente tecnologías usadas ampliamente en la empresa como Servlets, JSPs, EJBs, arquitectura J2EE, JDBC, SGBDs relacionales y JNI. Aunque me queda superar una espinita que no he conseguido por falta de tiempo que es JAAS y JMS pero que espero que muy pronto la habré implantado en GMODI.