TFC - J2EE

Proyecto de reparaciones y control de ventas

Naoru Project

Alumno: **David Ferreiro Vilar**

E.T. Informática de Gestión

Consultor: **Jose Juan Rodriguez**

17 de Enero de 2011

1 Dedicatoria

A mis padres, Lourdes y Simón.

Por su preocupación y el tiempo

que han dedicado a sus labores

para poder permitirme el estudio en

una Universidad.

A mi novia, Lidia, por ser paciente

conmigo durante estos 3 largos años

de estudios universitarios. Y por tener

la comprensión de que no le haya podido

dedicar todo el tiempo que quisiera.

A mis profesores, a ellos que son

la base de mi conocimiento

y me han ayudado a llegar donde estoy.

2 Resumen

Este Trabajo de Final de carrera tiene como objetivo la realización del diseño y la implementación de un sistema de consulta de estado de reparaciones y compras realizadas online.

Como profesional del sector de la Informática, no tengo conocimiento de la existencia de ningún tipo de producto similar en el mercado actualmente. Estamos entonces en frente de un proyecto que no tiene ningún otro similar para realizar una comparativa, por la cual cosa, todo el proceso de diseño y creación y tratamiento del prototipo parten des de cero utilizando las nuevas tecnologías J2EE.

Para el desarrollo de esta nueva aplicación, nos centraremos sobre todo en el MVC (Modelo Vista Controlador) y en los principios obligatorios de Usabilidad, Accesibilidad y Diseño Gráfico para una fácil interacción entre el usuario y el programa.

El proyecto Naoru permitirá al cliente saber en todo momento (mientras esté conectado a Internet) el estado de sus equipos llevados a reparar (el estado, precio i otras observaciones) y poder consultar las compras realizadas de una forma sencilla, relajada y segura, proporcionándole todas las garantías y servicios de tal manera cómo si hubiese ido a la tienda a preguntar al dependiente.

El sistema, en su conjunto aprovechará las ventajas que ofrecen las últimas tecnologías J2EE que actualmente dispone el mercado de la programación, con tal de poder proporcionar al cliente una interfaz fácil de usar, intuitivo y con unos exigentes controles de calidad.

El sistema será capaz, mediante un dispositivo móvil o un Ordenador Personal conectados a Internet, de satisfacer las necesidades del cliente a la hora de realizar cualquier tipo de consulta.

3 Índice

4 Introducción

4.1 Estudio Previo Realizado

La idea surgió mientras observando en primera persona una tienda de informática, pude contemplar la mala gestión que llevaban los operarios con los ordenadores arreglados y los que se debían arreglar.

También vi los problemas que tenían tanto el cliente como el vendedor a la hora de mirar si un aparato estaba en garantía, ya que el cliente no tenía la factura y el vendedor no sabía buscarla. Todo eso me hizo pensar en una manera de gestionar los dos problemas, buscarles una solución y a la vez dar otro servicio al cliente.

Mi deseo a la hora de enfocar el proyecto era obtener una aplicación que sirviéndose de tecnologías punteras, de frameworks de reconocido prestigio empleados por la comunidad de desarrolladores java, cumpliera los cometidos exigidos, pero que al mismo tiempo si se quisiera ampliar esas especificaciones, no fuera costoso de implementar.

Por lo tanto el método que debía seguir era crear una arquitectura de la aplicación tan robusta y tan modulable que no se viera limitada por las necesidades funcionales, sino más bien, que cualquier necesidad funcional, tanto presente como futura, fuera fácilmente llevada a la práctica utilizando dicha arquitectura.

Lo más similar que he contrastado en el mercado al producto que voy a desarrollar, es el tema de envíos, en empresas de transporte o en los paquetes azules de correo, donde el cliente puede ver por donde está su paquete o consultar si ya ha sido enviado o recibido.

Evidentemente este proyecto va a ir más allá y podría expandirse mucho más. Se puede suponer un taller mecánico en el que disponen de este programa, no sé si a ustedes les ha pasado que llaman al mecánico para ver cómo está el coche y resulta que aún no lo han mirado, o que el mecánico del taller no sabe si llegó antes un vehículo u otro, con este proyecto tanto clientes como empresas están informados y organizados.

4.2 Objetivos

El objetivo principal de este TFC es profundizar en el uso de la tecnología Java y adquirir conocimientos sobre la arquitectura J2EE mediante el desarrollo del proyecto. Dicho proyecto consiste en la realización del diseño e implementación de un portal Web para una tienda de informática (Por otro lado, este portal podría aplicarse a muchos otros sectores, como talleres mecánicos, electrónicos…). La gestión correrá siempre a cargo de la empresa y el cliente únicamente podrá consultar.

El portal servirá para que cuando una persona solicite a la empresa la reparación de un ordenador (o cualquier otro dispositivo), el cliente esté informado en todo momento de la situación del equipo. De esta manera des de cualquier lugar con acceso a Internet podrá ver si su dispositivo está listo o no, si faltan piezas e incluso el precio de la reparación a medida que se desarrolle. Evidentemente no se trata de que el técnico dedique más horas a modificar esto que a reparar el ordenador, por lo que se diseñará una interfaz muy simple y rápida para que la gestión de todo le tome menos de 30 segundos.

La empresa dará de alta al cliente si este es nuevo o añadirá la reparación (si el cliente ya está dado de alta). Al empezar la reparación, el técnico solo tendrá que cambiar el estado de la reparación de por ejemplo de “En Espera” a “Reparando” o a “Completado” y el precio una vez finalizado. También se podrá tener un comentario u observación que ambas partes verán.

De la misma manera, el vendedor de la tienda, podría añadir las compras que realiza el cliente (en tienda, ya que en el proyecto no se contempla la posibilidad de la compra de componentes online), para así tener los dos un registro de las compras y en qué fechas se realizaron.

4.3 Enfoque y Metodología Seguida

Para el desarrollo de la aplicación se ha intentado seguir el enfoque de una empresa de desarrollo de software. Esto implica que no sólo es importante el resultado final, sino que éste debe haberse acotado previamente de manera clara, debe generarse la documentación adecuada a cada fase y deben alcanzarse la consecución de los objetivos en los plazos establecidos.

Para ello se ha utilizado un ciclo en cascada clásico en el que se han previsto y realizado las siguientes etapas:

- Definición funcional

- Planificación del proyecto

- Análisis

- Diseño

- Implementación

- Pruebas

- Documentación

En paralelo, desde el comienzo del TFC se ha avanzado en el estudio de la arquitectura J2EE y del software necesario para su ejecución. En primer lugar, se ha realizado una breve investigación de las opciones disponibles. Una vez vistas las opciones más adecuadas al proyecto se han realizado algunas pruebas de implementación para evaluarlas. En un principio se empezó con Netbeans i Jboss, pero había problemas de ejecución al meter Struts 2 por lo que se decidió probar otro tipo de software. Finalmente, definidas las principales tecnologías a utilizar, en concreto Java Server Pages, Struts 2, Apache Tomcat e Hibernate, se ha buscado material bibliográfico que permitiese la formación en dichas tecnologías y sirviese de referencia durante la implementación del proyecto.

4.4 Planificación del Proyecto

Teniendo en cuenta el plan de trabajo establecido para las entregas correspondientes del TFC, a continuación se mostrará un diagrama de Gantt y un breve resumen temporal de las fechas establecidas para el correcto seguimiento de la asignatura.

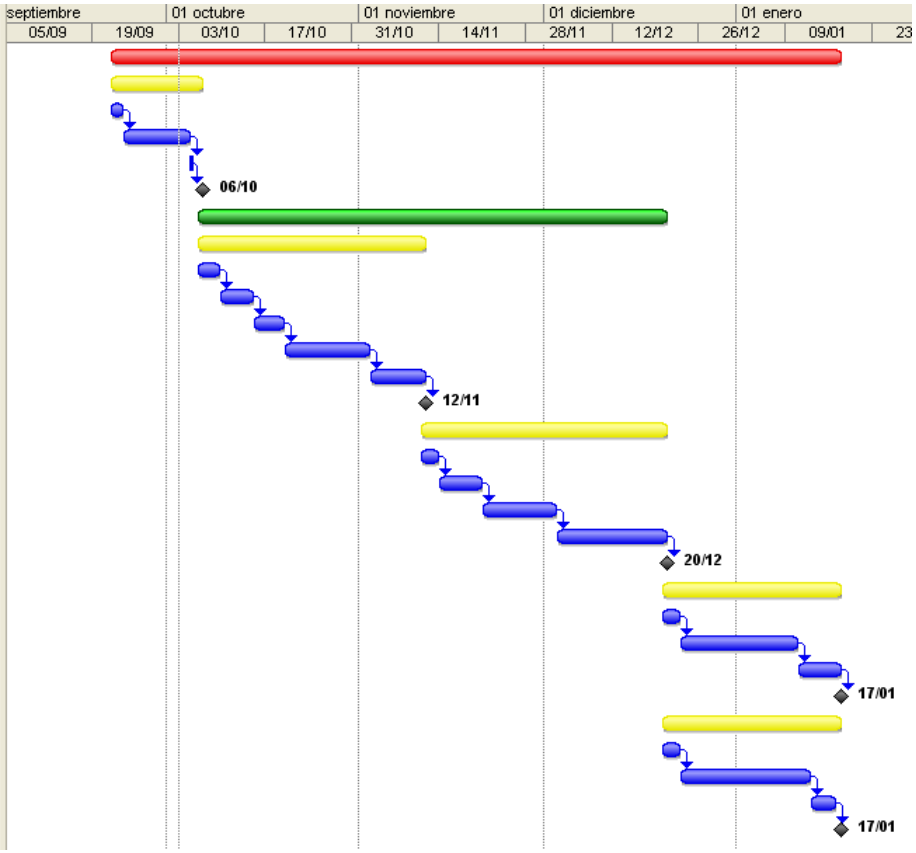
La planificación llevada a cabo tiene como fecha de inicio el día 21/9/2010 (inicio de semestre) y fecha de finalización el 15/01/2011.

Cabe destacar que en el diagrama de Gantt no se tiene previsto ningún día festivo (del 21/9/2010 al 15/01/2011), ya que en mi caso he de dedicar tiempo al trabajo (TFC) tanto laborales como festivos.

Se podrá realizar algún que otro ajuste pero siempre teniendo en cuenta las entregas obligatorias.

1.4.2 - Diagrama de Gantt de seguimiento

	Nombre de tarea	Comienzo	Fin	Predeces
1	**TFC J2EE : Diseño e Implementación del proyecto Naoru**	**mié 22/09/10**	**lun 17/01/11**	
2	**1 - Diseño y desarrollo del "Plan de Trabajo"**	**mié 22/09/10**	**mié 06/10/10**	
3	1.1 - Planteamiento y Estructuración del Plan de Trabajo	mié 22/09/10	jue 23/09/10	
4	1.2 - Desarrollo y Redactado del Plan de Trabajo	vie 24/09/10	lun 04/10/10	3
5	1.3 - Revisión del Plan de Trabajo	lun 04/10/10	mar 05/10/10	4
6	1.4 - Entrega del Plan de trabajo	mié 06/10/10	mié 06/10/10	5
7	**2 - Aprendizaje de Tecnologias J2EE**	**mié 06/10/10**	**lun 20/12/10**	
8	**3 - Analisis funcional y Diseño de la Arquitectura**	**mié 06/10/10**	**jue 11/11/10**	
9	3.1 - Especificaciones	mié 06/10/10	sáb 09/10/10	
10	3.2 - Casos de uso	sáb 09/10/10	vie 15/10/10	9
11	3.3 - Localización de classes	vie 15/10/10	mar 19/10/10	10
12	3.4 - Definición del diseño	mar 19/10/10	mar 02/11/10	11
13	3.5 - Terminar el diseño y revisión	mar 02/11/10	vie 12/11/10	12
14	3.6 - Entrega Estudio y desarrollo + Esudio Usabilidad y Accesibilidad	vie 12/11/10	vie 12/11/10	13
15	**4 - Diseño y implementación del Proyecto Naoru**	**jue 11/11/10**	**lun 20/12/10**	
16	4.1 - Planificación de las funcionalidades y requerimientos del prototipo	jue 11/11/10	sáb 13/11/10	
17	4.2 - Diseño de la estructura principal	sáb 13/11/10	sáb 20/11/10	16
18	4.3 - Sistema de Gestión de Usuarios	sáb 20/11/10	vie 03/12/10	17
19	4.4 - Sistema de Gestión de Reparaciones y Compras	vie 03/12/10	lun 20/12/10	18
20	4.5 - Entrega Implementación del Proyecto Naoru	lun 20/12/10	lun 20/12/10	19
21	**5 - Diseño y desarrollo de la memoria del proyecto**	**lun 20/12/10**	**lun 17/01/11**	
22	5.1 - Planteamiento y Estructuración de la Memoria	lun 20/12/10	mié 22/12/10	
23	5.2 - Desarrollo y redactado de la Memoria	mié 22/12/10	lun 10/01/11	22
24	5.3 - Revisión exhaustiva final de la Memoria	lun 10/01/11	lun 17/01/11	23
25	5.4 - Entrega de la Memoria	lun 17/01/11	lun 17/01/11	24
26	**6 - Diseño y desarrollo de la presentación virtual**	**lun 20/12/10**	**lun 17/01/11**	
27	6.1 - Planteamiento y estructuración de la presentación virtual	lun 20/12/10	mié 22/12/10	
28	6.2 - Desarrollo y elaboración de la presentación virtual	mié 22/12/10	mié 12/01/11	27
29	6.3 - Revisión exhaustiva final de la presentación virtual	mié 12/01/11	dom 16/01/11	28
30	6.4 - Entrega de la presentación virtual	lun 17/01/11	lun 17/01/11	29



5 Análisis

En este apartado avanzaremos en el análisis del problema propuesto, describiendo los actores y los casos de uso más importantes así como la arquitectura a utilizar.

Es importante destacar, que no es una aplicación de facturación, pero bien podría servir en un futuro con las ampliaciones adecuadas.

Además dada su generalización, se podría utilizar para un abanico diverso de empresas y sectores diferentes, des de ordenadores, coches y personas ingresadas en un hospital, todo ello con unas simples modificaciones.

5.1 Diagrama de la Arquitectura

Se ha empleado un modelo vista controlador (MVC implementado en el framework de Struts2), su definición obliga a desarrollar la funcionalidad de Actions, Forms y JSP.

Los Actions se comunican con los Forms o formularios de donde recogen información suministrada por el usuario. Estos Action llaman a la lógica de negocio y estos a los controladores (a la persistencia) que mapea la Base de Datos.

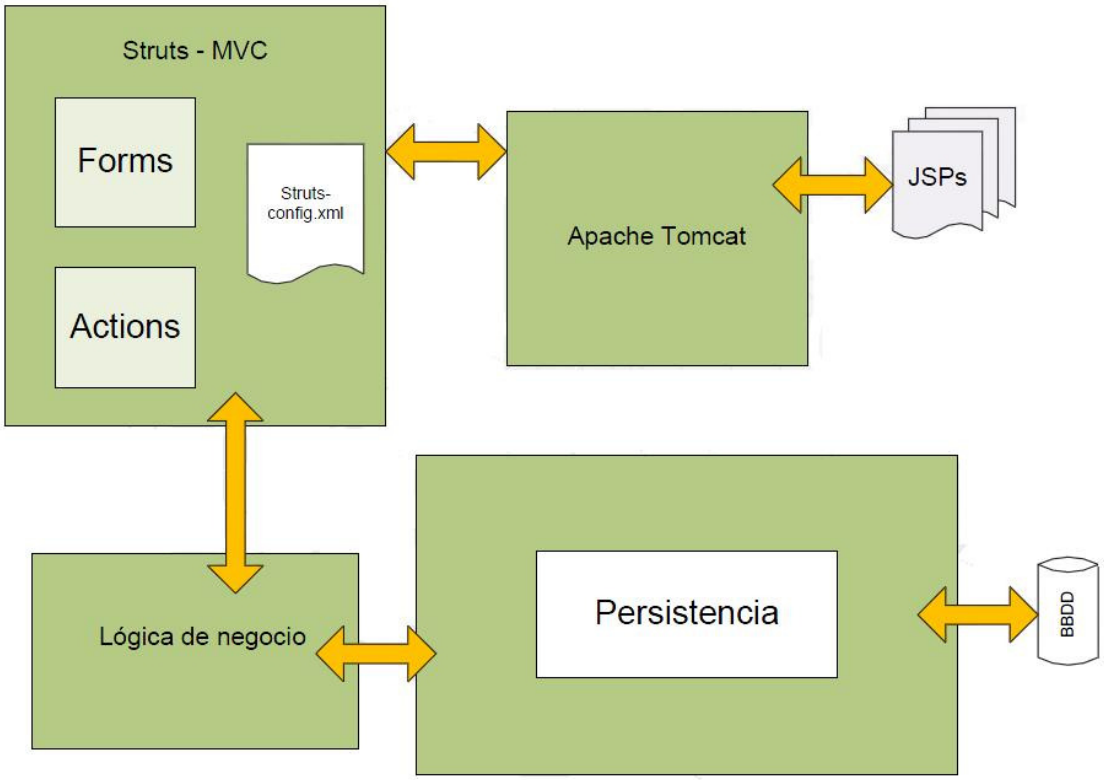


Ilustración 1. Diagrama de arquitectura

5.2 Arquitectura y Complementos

El framework de Struts2 implementa un patrón de software que se llama Modelo Vista Controlador (MVC) en el podemos localizar tres capas principales, la primera es la vista a la que tiene acceso los usuarios, la segunda son los formularios donde se modeliza los datos suministrados o las interacciones que hayan tenido los usuarios. Por último está el controlador que es quién obtiene los datos de los formularios y se los envía a la lógica de negocio y obteniéndose unos resultados que son mostrados a través de la vista. Struts2 no implica que la vista se tenga que realizar con JSP exclusivamente, aunque lo hemos hecho así, se podrían utilizar plantillas de velocity por ejemplo.

En los últimos tiempos han aparecido un número muy amplio de soluciones que encapsulan el acceso y manejo a base de datos (por ejemplo Hibernate, Ibatis, OJB). Hibernate es un potente ORM (Object Relational Modeler), es decir trata de salvar la distancia que hay entre el paradigma de la orientación a objetos con el diseño relacional de Base de Datos, siendo una capa de abstracción logra que el usuario no deje el

paradigma de orientación a objetos en ningún caso ya que Hibernate se encarga de hacer la traducción entre el mundo orientado a objetos y el mundo relacional de base de datos.

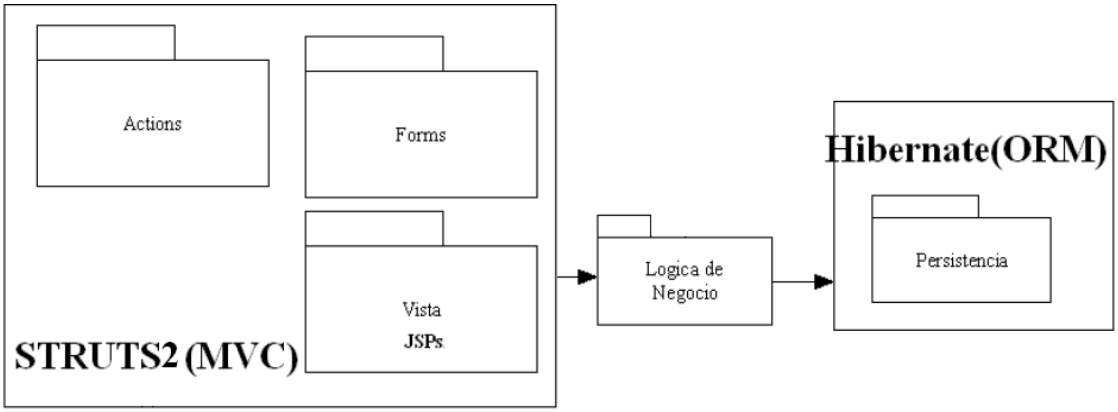


Ilustración 2. Diagrama de arquitectura básico

5.3 Planteamientos Iniciales y Uso de Tecnologías

El presente trabajo trata de crear una arquitectura modulable y adaptativa a nuevas tecnologías y no ciñe su implementación a los requerimientos funcionales ya que una visión demasiado estrecha conllevaría una arquitectura que sería muy poco mantenible y menos aún evolutiva.

El planteamiento del TFC fue crear una serie de capas de abstracción con una funcionalidad bien definida, para que con este orden se obtengan todos los beneficios de un proyecto que pueda evolucionar.

Aunque podía haber utilizado servlets para mostrar JSPs, acceso a base de Datos mediante JDBC, creo que no se hubiera puesto de manifiesto lo que significa J2EE, he preferido mostrar el uso de frameworks que llevan un "largo periodo" en el mercado y que tras varias revisiones se muestran realmente robustos en las tareas para las que han sido diseñados.

Descripción breve de las tecnologías usadas en el TFC

STRUTS2

Struts2, es un framework open-source que implementa el patrón de diseño modelo-vista-controlador (MVC), en la actualidad es un referente en el desarrollo de aplicaciones basadas en la Web. Esta especialmente indicado para aplicaciones con un número importante de páginas, que necesite ser fácilmente mantenido a lo largo del tiempo.

Struts2 al igual que Struts nació como una respuesta a la propuesta de Arquitectura JSP de Modelo 2.

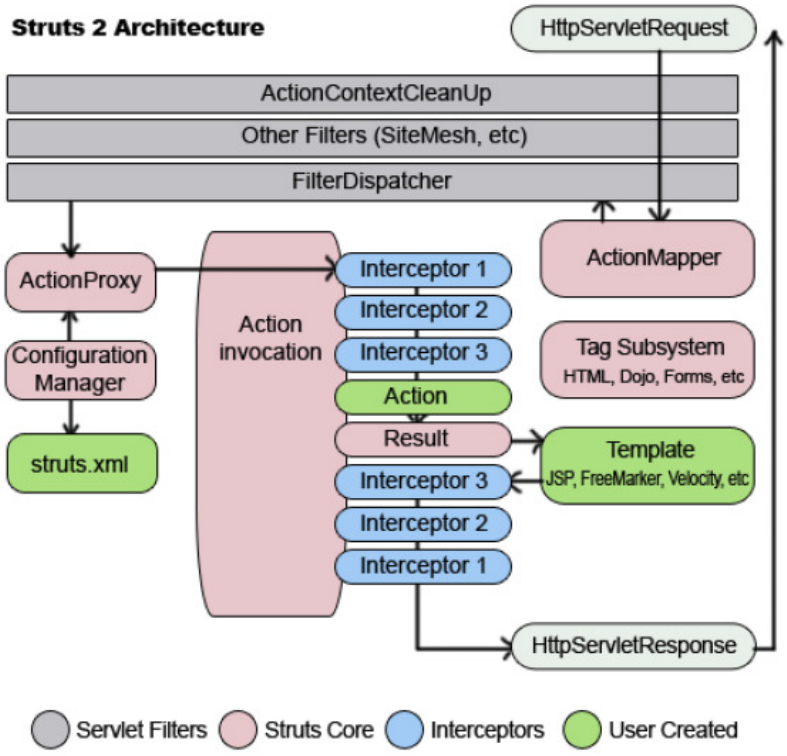


Ilustración 3. Diagrama de la Arquitectura completo de Struts2

Los servlets son el Controlador que se encarga de procesar las peticiones (Request) además de la creación de los beans y otros objetos que se requieran para las JSP's. También poseen toda la lógica que decide, dependiendo de las acciones del usuario, a que pagina JSP hay que redirigirlo.

Con este esquema se logra aislar la lógica del proceso en un lugar y se elimina de las JSP's toda la lógica de proceso que no vaya encaminada a la presentación en exclusiva. Así pues las JSP's son responsables única y exclusivamente de obtener objetos o beans creados con anterioridad en y la extracción de contenido dinámico de servlets para ser insertados dentro de plantillas estáticas.

El beneficio de esta técnica es claro a la vez que simple, se logra la separación de la presentación y el contenido, definiendo diferentes papeles para cada uno de los actores en un desarrollo Web, así pues los diseñadores trabajan sobre esa capa de presentación mientras que los desarrolladores se preocupan en pasar el contenido a esas JSP's. Por consiguiente, esta separación, o compartimentación obtiene sus mayores beneficios en

aplicaciones complejas ya que facilita el trabajo de ambos grupos sin que unos se estorben a los otros.

Hemos hablado de JSP's como capa de presentación y aunque Struts posea un servlet que se encarga de la comunicación entre otras clases y las JSP's se puede añadir otros servlet distintos para utilizar otras tecnologías en la capa de presentación.

VALIDATOR

El Struts2 Validator es otra extensión de Struts2, se emplea para la validación de la información que es introducida por el usuario vía Web, permite el manejo centralizado de la validación de la Web, sin importar el tipo de tecnología que se utilice en la vista.

HIBERNATE

Hibernate es una poderosa herramienta, es un ORM, es decir, un Object Relational Modeler (Modelador Objeto/Relación), provee una avanzada ejecución y habilidad de manejo entre el mundo Relacional (típico de base de datos) y el mundo Orientado a Objetos, moldea el mundo Relacional para dotar a nuestras aplicaciones de un entorno puramente orientado a objetos. Hibernate desarrolla clases persistentes usando el lenguaje Java, extendiendo sus tecnologías incluyendo asociaciones, herencia, polimorfismo, composición, y colecciones.

Para una mejor eficiencia y rapidez de trabajo, se utilizan las annotations para hacer más claro el código y el mapeo, con ellas no ahorramos los XML con los correspondientes mapeados.

TOMCAT

Tomcat es un servidor de aplicaciones open-source. Tomcat implementa las especificaciones de los servlets y de JavaServer Pages (JSP) de Sun Microsystems.

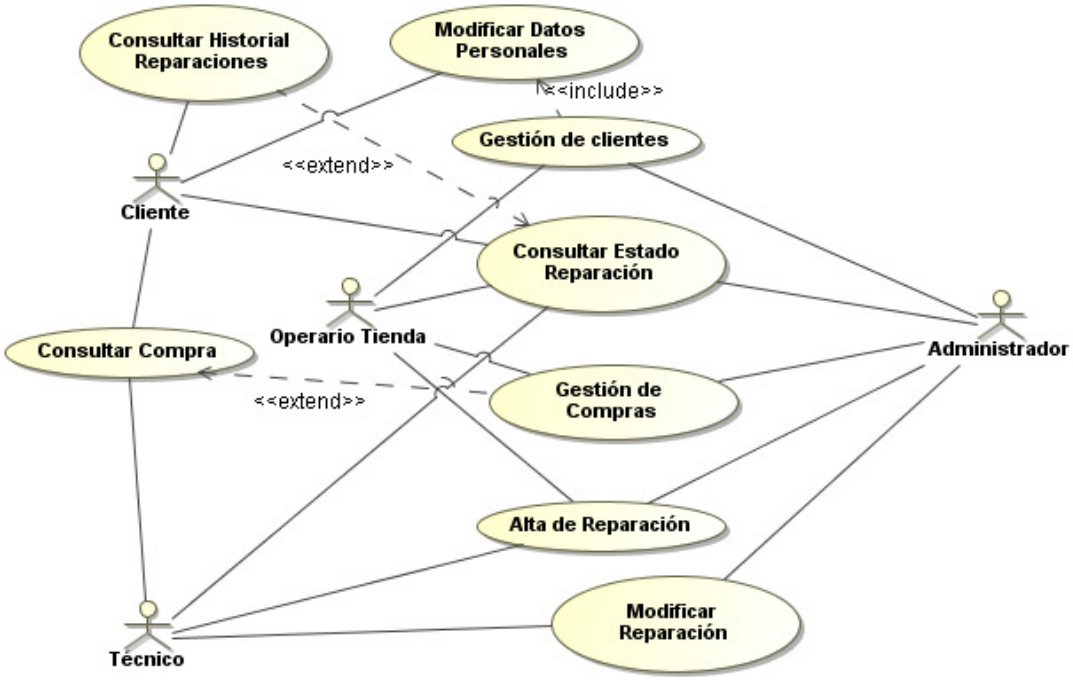
5.4 Actores

Los actores los definimos en el plan de trabajo, pero los veremos de nuevo para refrescar la memoria. Los actores son los siguientes:

- *Operario de Tienda* (vendedor):
  - Éste podrá dar de alta, modificar o consultar clientes (que a la vez son usuarios).
  - Podrá registrar (dar de alta), modificar o consultar las compras que los clientes lo deseen y estén dados de alta
  - También podrán dar de alta las reparaciones y consultar el estado de éstas.
- *Técnico*:
  - Éste podrá consultar las compras realizadas por los clientes, para saber si el artículo que le han entregado esta en garantía o no
  - También podrá modificar el estado de las reparaciones que estén en curso, nunca de una que ya esté cerrada, así como añadir comentarios o el precio de la reparación
  - El técnico al no estar de dependiente (se supone) nunca podrá dar de alta clientes ni compras, aunque eso se podría solucionar si hiciese falta con el siguiente rol.
- *Administrador*:
  - El administrador, como su nombre bien indica, puede hacer lo de los anteriores roles. Por lo tanto, podrá dar de alta, modificar o consultar clientes
  - Dar de alta, modificar o consultar compras
  - Dar de alta, modificar o consultar reparaciones aunque éstas estén cerradas.
- *Cliente*:
  - El cliente podrá consultar sus compras realizadas y el estado de los equipos que tenga en el servicio técnico
  - Consultar el Histórico de las reparaciones hechas
  - También podrá modificar sus datos personales
  - De momento no se contempla que un cliente pueda dar-se de alta por si mismo en el sistema para evitar altas innecesarias.

5.5 Casos de Uso

A partir de la información anterior podemos hacer el diagrama de casos de uso y definir cada uno de ellos. Veamos cómo queda el diagrama:



El primer caso de uso que vamos a analizar no se contempla en el esquema para una mejor legibilidad del diagrama y ya que es algo que todos tienen que hacer antes de poder hacer cualquiera de los otros casos.

Casos de uso	**Acceso al sistema (Login)**
Descripción	El sistema pide el nombre de usuario (login) a todos los usuarios, y su contraseña (password), verifica que este usuario esté dado de alta dentro del sistema i que la contraseña sea correcta
Actores	Administradores, Técnicos, Operarios de Tienda y Clientes
Papel de usuario	Lo utilizan todos los usuarios para acceder a la aplicación
Casos de uso relacionados	NA.
Precondición	El usuario es válido si está dado de alta dentro de la base de datos del sistema y conoce su contraseña actual
Post-condición	El sistema ha validado el usuario i, dependiendo de su perfil, le permite la entrada, mostrándole su menú de opciones correspondiente
Área	
Módulos que intervienen	
Otros comentarios	Esta es la primera pantalla que se muestra al ejecutar el programa 1. El sistema muestra la ventana de login al usuario 2. El usuario introduce su credenciales (nombre de usuario y clave) y pulsa aceptar 3. El sistema verifica que las credenciales existen y son correctas [Flujo Excepcional: el sistema no valida las credenciales del usuario] 4. El sistema comprueba el rol del usuario i se le da acceso como tal.

Casos de uso	**Consulta Historial Reparaciones**
Descripción	Permite al usuario ver el historial de Reparaciones de sus equipos
Actores	Clientes , Administradores, Técnicos y Operarios de tienda
Papel de usuario	El usuario lo usa para ver todas las reparaciones hechas.
Casos de uso relacionados	Acceso al Sistema
Precondición	El usuario se ha identificado correctamente, el usuario es valido i esta dado de alta
Post-condición	Permite al usuario ver todas las reparaciones hechas
Área	
Módulos que intervienen	
Otros comentarios	Permite al usuario ver el historial de reparaciones hechas con los detalles de información, entrega, precio...

Casos de uso	**Gestión de Clientes**
Descripción	Permite al usuario la gestión de los distintos usuarios (Clientes, Operarios de tienda, Técnicos y Administradores) en el mismo.
Actores	Administradores, Operarios de Tienda y Técnicos
Papel de usuario	Le permite al usuario dar de alta y de baja a los usuarios del sistema así como la modificación de sus datos.
Casos de uso relacionados	Modificar datos personales

Precondición	El usuario tiene derechos de ejecución sobre esta funcionalidad. Los distintos usuarios han de haber rellenado el formulario con los datos mínimos requeridos.
Post-condición	Se añade o se borra un usuario, así como la modificación de los datos de los mismos. Posteriormente toda modificación queda registrada en la base de datos del sistema.
Área	
Módulos que intervienen	
Otros comentarios	Los datos quedan siempre almacenados en la base de datos del sistema. Los diferentes roles solo los podrá modificar el administrador ya que la gestión de clientes es igual a la gestión de usuarios de la base de datos.

Casos de uso	**Modificación datos personales**
Descripción	Permite al cliente la modificación de la información del mismo.
Actores	Clientes
Papel de usuario	Permite al cliente modificar la información personal.
Casos de uso relacionados	Gestión de Clientes.
Precondición	El usuario tiene derechos de ejecución sobre esta funcionalidad. El usuario ha de estar dado de alta en la base de datos del sistema y ha de haber hecho login en el sistema.
Post-condición	Queda modificada la información del Cliente. Posteriormente queda registrado en la base de datos del sistema.

Área	
Módulos que intervienen	
Otros comentarios	Los datos quedan siempre almacenados en la base de datos del sistema.

Casos de uso	**Consultar Estado Reparación**
Descripción	Permite al usuario consultar el estado de la reparación
Actores	Clientes , Operarios de Tienda, Administradores y Técnicos
Papel de usuario	El usuario lo usa para consultar el estado de la reparación
Casos de uso relacionados	Acceso al Sistema, Consultar Historial Reparaciones, Modificar Reparación
Precondición	El usuario se ha identificado correctamente, el usuario es válido i esta dado de alta
Post-condición	Permite al usuario ver el estado de la/s reparación/es
Área	
Módulos que intervienen	
Otros comentarios	Los técnicos y los administradores podrán ver todas las reparaciones, mientras que los clientes y los operarios solo verán las de si mismos o el cliente que tengan seleccionado en el caso de los Operarios

Casos de uso	**Consultar Compra**
Descripción	Permite al usuario consultar una compra realizada
Actores	Clientes , Operarios de Tienda, Administradores y Técnicos
Papel de usuario	El usuario lo usa para consultar el estado de la reparación
Casos de uso relacionados	Acceso al Sistema, Gestión Compras
Precondición	El usuario se ha identificado correctamente, el usuario es válido i esta dado de alta
Post-condición	Permite al usuario ver las compras realizadas
Área	
Módulos que intervienen	
Otros comentarios	Los técnicos y los clientes solo pueden consultar las compras (por tema de garantía), mientras que los administradores y los operarios además podrán hacer modificaciones a través de la gestión de compras.

Casos de uso	**Gestión de Compras**
Descripción	Permite al usuario la gestión de las compras realizadas por los distintos usuarios.
Actores	Administradores y Operarios de Tienda
Papel de usuario	Permite al usuario dar de alta y de baja compras así como la modificación de sus datos.
Casos de uso relacionados	Consulta de compras

Precondición	El usuario tiene derechos de ejecución sobre esta funcionalidad. Los distintos usuarios han de haber rellenado el formulario con los datos mínimos requeridos.
Post-condición	Se añade o se borra una compra, así como la modificación de los datos de las mismas. Posteriormente toda modificación queda registrada en la base de datos del sistema.
Área	
Módulos que intervienen	
Otros comentarios	Los datos quedan siempre almacenados en la base de datos del sistema.

Casos de uso	**Alta de Reparación**
Descripción	Permite al usuario la alta de una reparación en el sistema.
Actores	Administradores, Técnicos y Operarios de Tienda
Papel de usuario	Permite al usuario dar de alta una reparación a un cliente concreto.
Casos de uso relacionados	Modificar Reparación, Consulta reparación
Precondición	El usuario tiene derechos de ejecución sobre esta funcionalidad. Los distintos usuarios han de haber rellenado el formulario con los datos mínimos requeridos.
Post-condición	Se añade o se borra una compra, así como la modificación de los datos de las mismas. Posteriormente toda modificación queda registrada en la base de datos del sistema.
Área	
Módulos que intervienen	

Otros comentarios	Los datos quedan siempre almacenados en la base de datos del sistema.

Casos de uso	**Modificar Reparación**
Descripción	Permite al usuario modificar una reparación existente en el sistema.
Actores	Administradores y Técnicos
Papel de usuario	Permite al usuario modificar una reparación de un cliente concreto.
Casos de uso relacionados	Alta de Reparación, Consulta reparación
Precondición	El usuario tiene derechos de ejecución sobre esta funcionalidad. Los distintos usuarios han de haber rellenado el formulario con los datos mínimos requeridos.
Post-condición	Se modifica la reparación con los datos nuevos en la base de datos del sistema.
Área	
Módulos que intervienen	
Otros comentarios	Los datos quedan siempre almacenados en la base de datos del sistema.

5.6 Diagrama de clases

Aunque el modelo puede sufrir algún cambio además de añadir métodos, este será un esquema base para empezar:

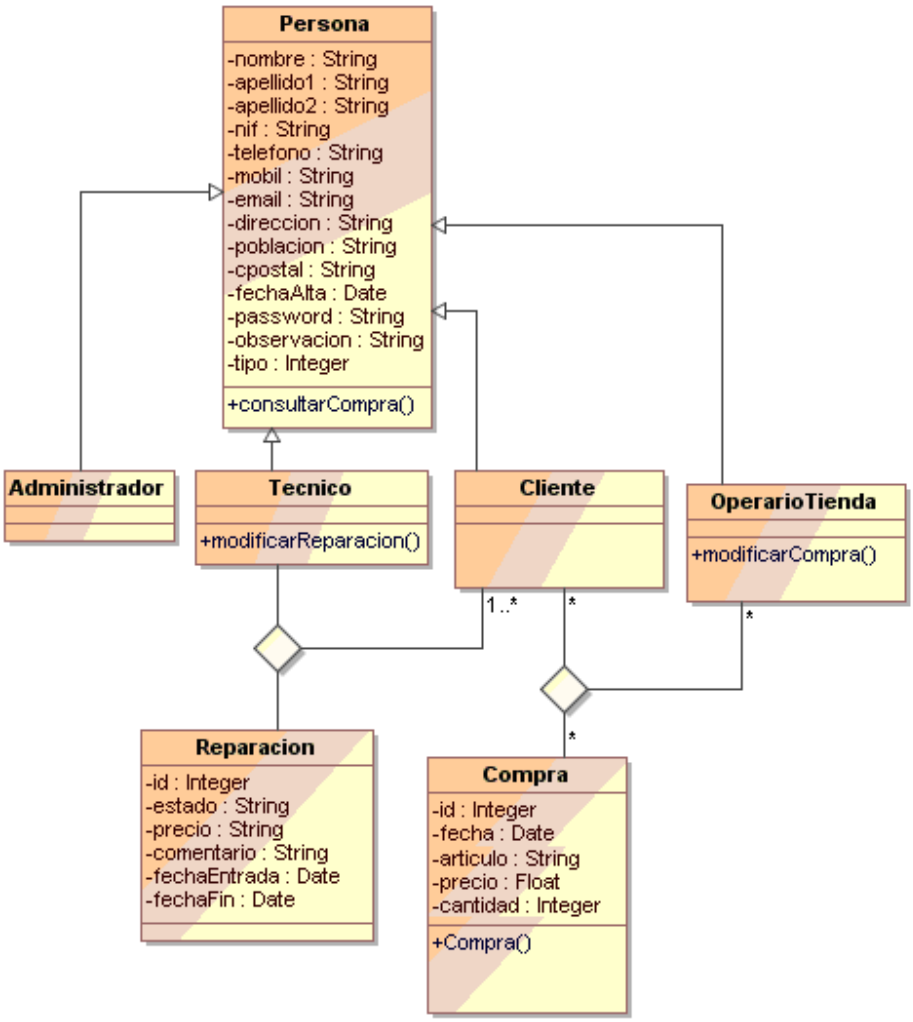


Ilustración 4. Diagrama de clases principal

Creo que todas las clases son suficientemente explícitas. Los atributos que puedan faltar así como los métodos se desarrollaran y buscaran con toda profundidad más adelante.

6 Diseño

Nuestra aplicación tiene que funcionar a través de Internet, con lo cual tendremos que usar una estructura cliente servidor. En el Plan de Trabajo ya comentamos que usaríamos un diseño en capas.

6.1 Programación por Capas

La programación por capas es un estilo de programación en el que el objetivo primordial es la separación de la lógica de negocios de la lógica de diseño; un ejemplo básico de esto consiste en separar la capa de datos de la capa de presentación al usuario.

La ventaja principal de este estilo es que el desarrollo se puede llevar a cabo en varios niveles y, en caso de que sobrevenga algún cambio, sólo se ataca al nivel requerido sin tener que revisar entre código mezclado.

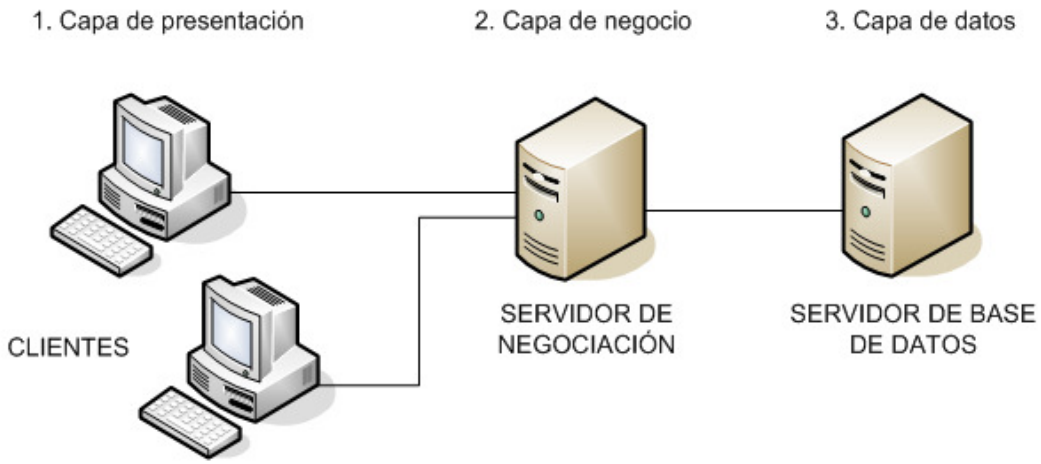


Ilustración 5. Diagrama básico del patrón MVC

- Capa de presentación: es la que ve el usuario, presenta el sistema al usuario, le comunica la información y captura la información del usuario para luego procesar-la. También es conocida como interfaz gráfica y debe tener la característica de ser "amigable" (entendible y fácil de usar) para el usuario. Esta capa se comunica únicamente con la capa de negocio.
- Capa de negocio: es donde residen los programas que se ejecutan, se reciben las peticiones del usuario y se envían las respuestas tras el proceso. Se denomina capa de negocio porque es aquí donde se establecen todas las reglas que deben cumplirse. Esta

capa se comunica con la capa de presentación, para recibir las solicitudes y presentar los resultados, y con la capa de datos, para solicitar al gestor de base de datos almacenar o recuperar datos de él.
- Capa de datos: es donde residen los datos y es la encargada de acceder a los mismos. Está formada por uno o más gestores de bases de datos que realizan todo el almacenamiento de datos, reciben solicitudes de almacenamiento o recuperación de información desde la capa de negocio.

Como nuestro proyecto debe funcionar a través de Internet es lógico que usemos una estructura cliente/servidor.

Tal i como describimos en el Plan de trabajo utilizaremos la estructura antes comentada (por capas). Y aunque no es definitivo, la estructura a seguir será.

Para la capa de presentación usaremos JSP i HTML, el cual se comunicará con la capa de negocio mediante Struts2.

Para la capa de negocio usaremos Tomcat y seguramente usaremos algún patrón, como el Data Acces Object (DAO), el cual se comunicará con la base de dates (capa de datos, con MySQL que dispone de controladores JDBC i cumple con todos nuestros requisitos) mediante Hibernate.

Dada la inexperiencia en este tipo de proyectos, ya que esta es la primera vez que lo realizamos y es normal que vayamos mas perdidos, pero a cambio, después de finalizar el proyecto seremos capaces de repetirlo, esta vez asegurando todo lo que vayamos a realizar, sin ningún quizás y probablemente con increíbles mejoras.

Así pues, el navegador atiende las peticiones del usuario y las JSP crea las páginas dinámicas para que el usuario pueda interactuar, luego struts envía las peticiones a la JSP apropiada.

Mientas hibernate nos facilita el mapeo de las clases a las tablas de la base de datos. En mi caso, usaré las Annotations, que es más fácil que trabajar con un XML con todo el mapeo, de esta manera lo definimos en la propia clase. Este hará que cualquier modificación que hagamos al objeto, la llevemos a cabo a la base de datos de forma transparente.

6.2 Base de Datos

Para poder mantener los datos una vez que la aplicación se ha cerrado, se ha optado por combinar una base de datos relacional para la mayoría de clases de nuestro sistema y una estructura de ficheros para las consultas de los informes, de esta manera las consultas complejas sobre la base de datos solamente se realizarán una sola vez al crear los diversos informes (Si hace falta).

Para diseñar la base de datos se ha utilizado un entorno de diseño que nos proporciona la posibilidad de diseñar gráficamente las tablas, atributos, claves y relaciones construyendo más tarde la consulta SQL que nos proporcionará la creación de todas las tablas, atributos y relaciones…

ENTIDADES Y ATRIBUTOS DEL ESQUEMA E/R :

Persona

nombre , apellido1 , apellido2 ,nif , teléfono , móbil , email , dirección , población , cpostal , fechaAlta , password , observación , tipo

Administrador (entidad subclase de Persona)

nif

Tecnico (entidad subclase de Persona)

nif

Cliente (entidad subclase de Persona)

nif

OperarioTienda (entidad subclase de Persona)

nif

Reparacion

id , estado , precio , comentario , fechaEntrada , fechaFin

Compra

id , fecha

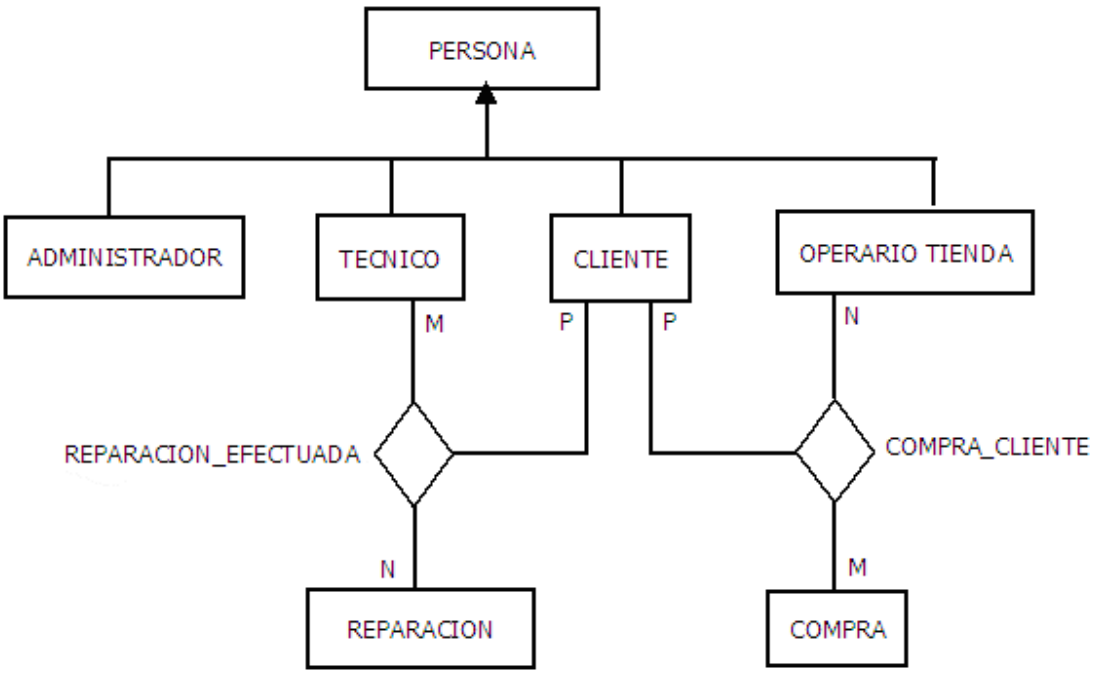


Ilustración 6. Diagrama de la base de datos

TRANSFORMACIÓN AL MODELO RELACIONAL DEL MODELO CONCEPTUAL OBTENIDO EN EL APARTADO ANTERIOR

Persona (nombre , apellido1 , apellido2 ,nif , teléfono , móbil , email , dirección , población , cpostal , fechaAlta , password , observación , tipo)

Administrador (nif)

{ nif } clave forana de Persona (nif)

Tecnico (nif)

{ nif } clave forana de Persona (nif)

Cliente (nif)

{ nif } clave forana de Persona (nif)

OperarioTienda (nif)

{ nif } clave forana de Persona (nif)

Reparacion (id , estado , precio , comentario , fechaEntrada , fechaFin)

Compra (id , fecha)

INTERRELACIONES

ReparacionEfectuada (nifTecnico , nifCliente , id)

{ nifTecnico } clave forana referente a Tecnico (nif)

{ nifCliente } clave forana referente a Cliente (nif)

{ id } clave forana referente a Reparacion (id)

CompraCliente (nifCliente , nifOperarioTienda , id)

{ nifCliente } clave forana referente a Cliente (nif)

{ id } clave forana referente a Compra (id)

{ nifOperarioTienda } clave forana referente a OperarioTienda (nif)

Instrucciones para la Creación de la Base de Datos.

Todas las instrucciones de creación e inserción principales se encuentran en el archivo: Base de Datos.sql

6.3 Diseño Final de las Pantallas

A continuación se muestran unos prototipos de las pantallas que fan a formar parte del proyecto. A continuación se muestra la pantalla de Inicio, lo que verá el usuario cuando acceda a nuestro portal. Las pantallas de usuario están basadas en el usuario Administrador

6.3.1 Pantalla de Inicio

En esta pantalla veremos una simple descripción de lo que es nuestra empresa y como trabajamos. Asi mismo vemos el menuú sencillo de navegación a mano izquierda.

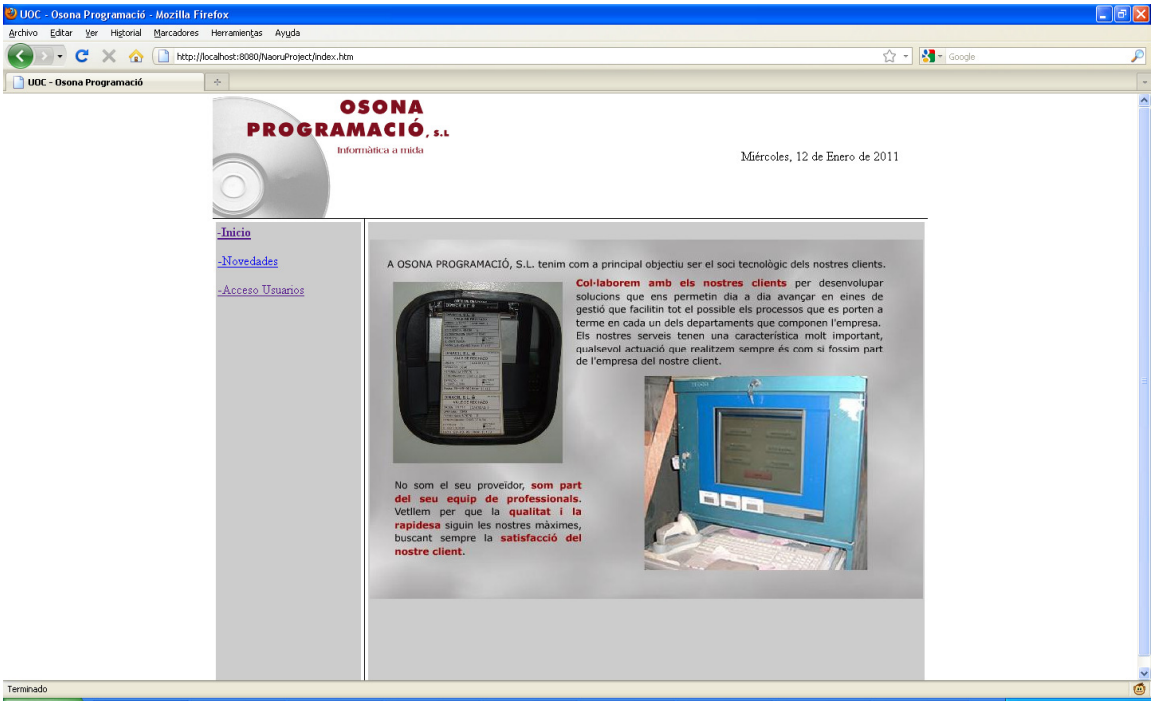


Ilustración 7. Página de Inicio

6.3.2 Pantalla de Novedades

En ella que se podrán ver las novedades que vayan saliendo al mercado o posibles ofertas. De momento en este proyecto no se ha pensado en la implementación de toda la gestión i queda para más adelante.

Ilustración 8. Página de Novedades

6.3.3 Pantalla de Acceso de Usuarios

En ella entraremos el nombre de usuario y el password para poder acceder ya sea como clientes o como cualquiera de los demás roles.

En caso de error se mostraría en la misma ventana un mensaje señalando la naturaleza del error.

Ilustración 9. Pantalla de Login (acceso para usuarios)

6.3.4 Pantalla de Bienvenida

Aunque es una pantalla que no aporta funcionalidad alguna, da la bienvenida a cualquier usuario que entre en el sistema. Pantalla simpática para tratar de acercar el usuario al programa y a la empresa. En un futuro se podría cambiar por la opción que se use más para ahorrar tiempo. De momento como no sabemos que acción harán con más frecuencia, se queda así.

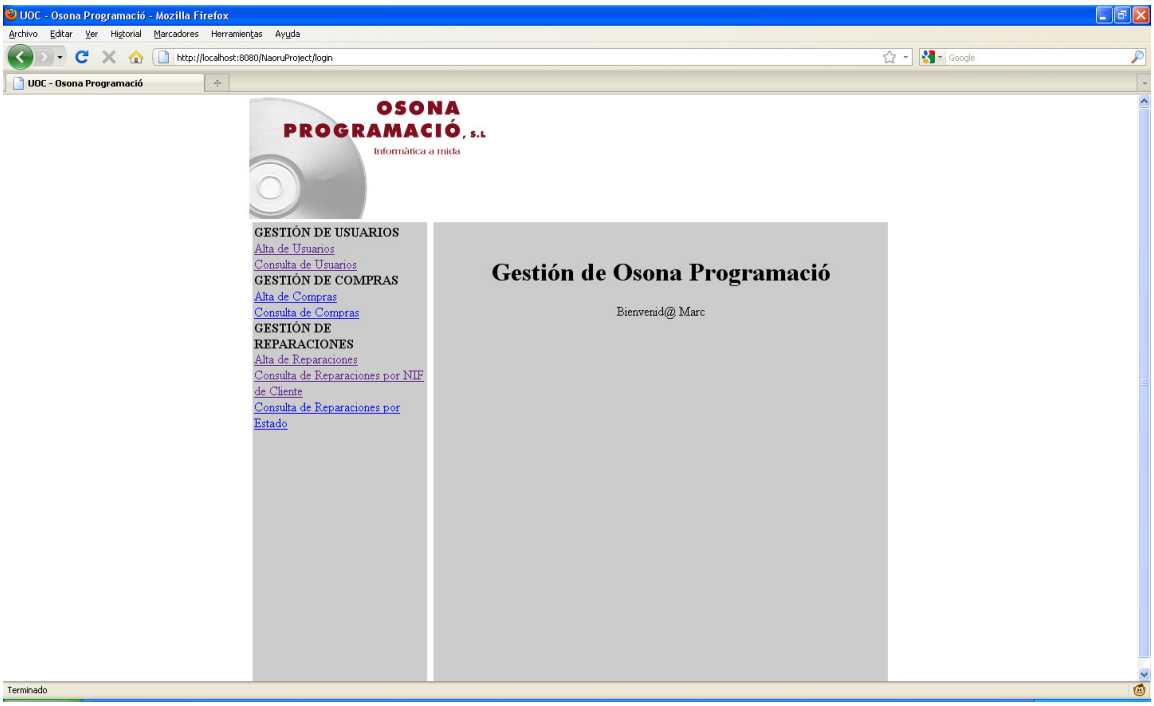


Ilustración 10. Pantalla de bienvenida

6.3.5 Pantalla de Altas de Usuario

En esta pantalla se darán de alta los usuarios, a continuación se muestra una pantalla des del rol de administrador, aunque sea de alta, en el mismo momento de la alta se puede modificar cualquier campo (excepto el nif) y se actualizará a la base de datos, como si un edición se tratara. El usuario cliente no tiene acceso a esta función (ni aunque logre cargar la pagina). Además se controlan diferentes errores, como el correo o los campos mínimos obligatorios.

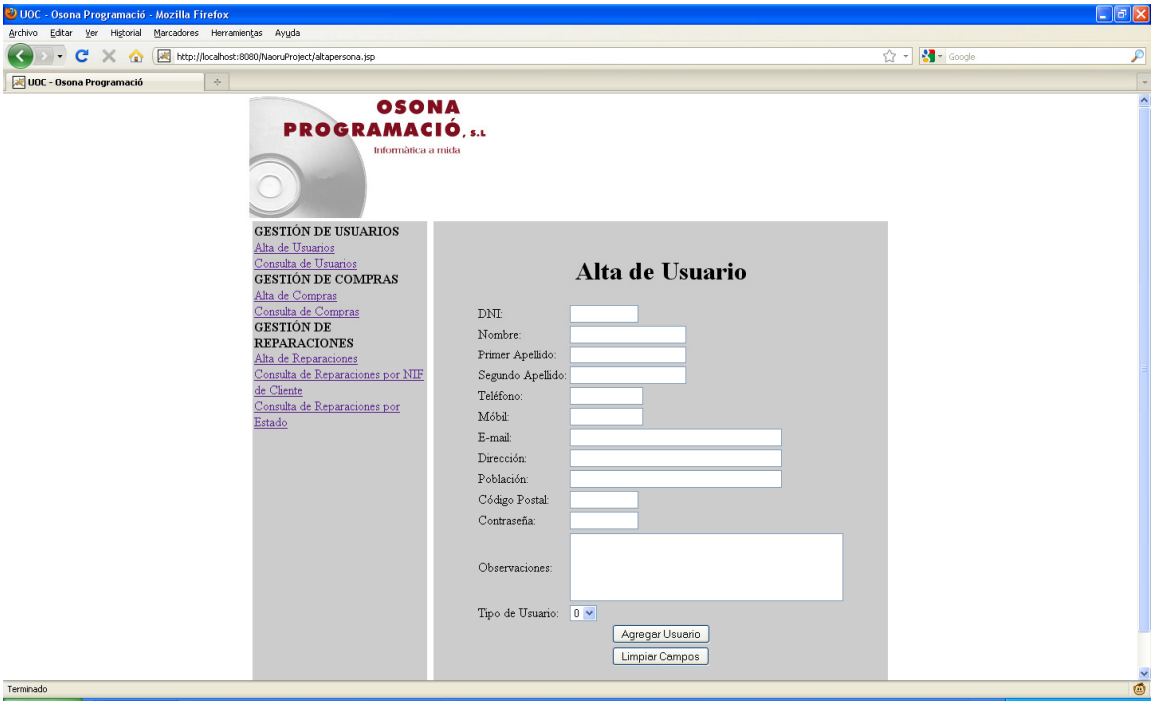


Ilustración 11. Pantalla de Alta de Usuarios

6.3.6 Pantalla de Consulta de Usuarios

Esta pantalla sirve para consultar la información de un cliente y para modificarlo si este tiene permisos. Solo tiene que entrar una parte del nif y se listaran todos los usuarios que tengan esa parte de la cadena (muy útil cuando alguien solo recuerda una parte de él). En caso de no tener permisos únicamente se mostrarían los campos de datos, ni el de editar ni el de eliminar.

Ilustración 12. Pantalla de consulta de Usuarios dados de alta

6.3.7 Pantalla de Alta de Compras

Esta pantalla sirve para dar de alta las compras que realizan los clientes siempre que el usuario tenga permisos.

Como ya se comentó el objetivo no es contabilidad ni nada parecido (de momento), solo tiene utilidad de cara el control de garantías i se podría utilizar para ver los mejores clientes.

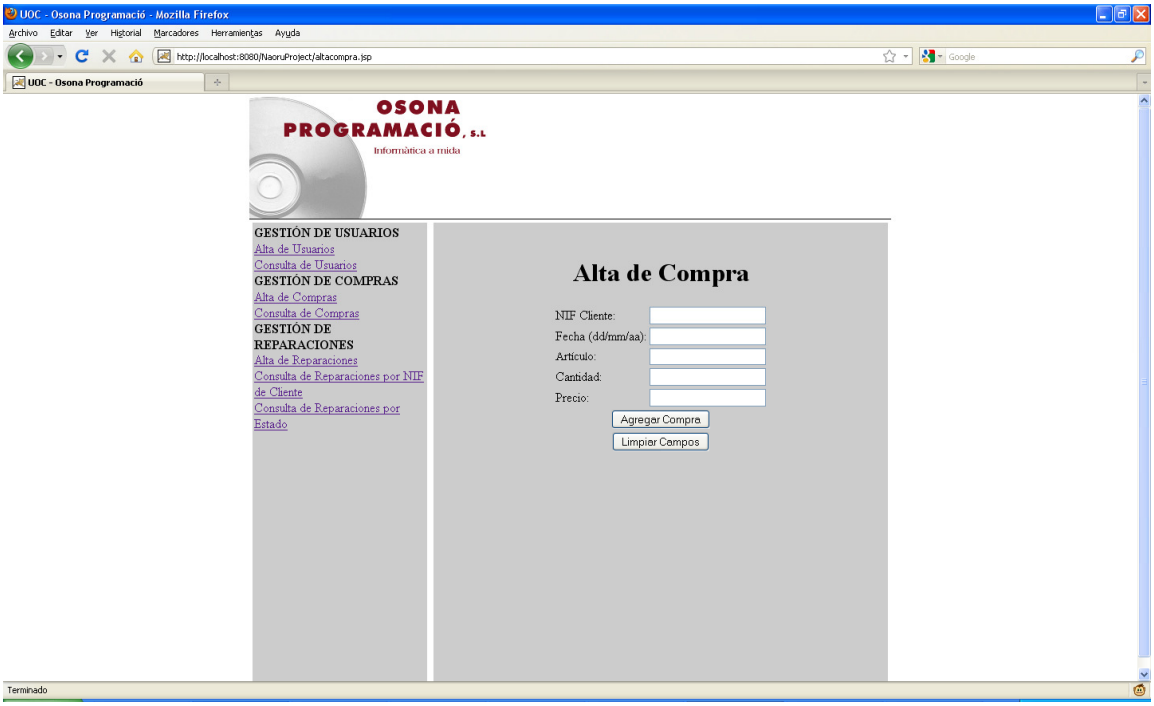


Ilustración 13. Pantalla de Alta de Compras

6.3.8 Pantalla de Consulta de Compras

Esta pantalla sirve para consultar las compras que realizan los clientes siempre que el usuario tenga permisos. En este caso si es un usuario cliente, solo podrá ver las suyas y no tendrá el buscador de clientes.

Si tiene permisos, des de esta misma consulta podría eliminar o modificar la compra. El buscador busca por parte del nif (si el cliente no lo recuerda entero).

En caso de no tener permisos para manipular los datos, tanto la columna Editar como Borrar no estarían visibles.

Ilustración 14. Pantalla de Consulta de Compras

6.3.9 Pantalla de Alta de Reparaciones

Esta pantalla sirve para dar de alta las reparaciones de los equipos de los clientes.

Una vez entrada la información se podrá guardar y entrar una nueva reparación si hiciese falta con los datos en pantalla. Si un usuario se pasa dándole a Agregar Reparación, estará registrando tantas reparaciones como veces haya apretado.

Ilustración 15. Pantalla de Alta de Reparaciones

6.3.10 Pantalla de Consulta de Reparaciones

Esta pantalla sirve para consultar las reparaciones que realizan los Técnicos por el Nif del cliente (o parte de él). En este caso si es un Técnico o Administrador, también podrá modificar la información que sale en ella, para que si el usuario la consulta pueda ver el estado de la misma. También puede borrarla.

Ilustración 16. Pantalla de Consulta de reparaciones por Nif

6.3.11 Pantalla de Consulta de Reparaciones

Pantalla que sirve para consultar las reparaciones según el estado que se elija. De esta manera se pueden ver cuales son las Reparaciones que aun no se han terminado o ni siquiera se han mirado.

Los usuarios con los permisos adecuados podrán como en la otra consulta modificar o eliminar las reparaciones que hagan falta.

Ilustración 17. Pantalla de Consulta de reparaciones por Estado

7 Implementación e Instalación

7.1 Requerimientos de Software

Para poder hacer funcionar el Proyecto basta con 2 elementos, aunque se aconseja tener un tercer elemento por si acaso. Necesitaremos una base de datos, en este caso MySQL Server 5.1, aunque se podría usar cualquier Servidor de base de datos, configurando apropiadamente el hibernate.cfg.xml e introduciendo el JAR con el driver de conexión a la base de datos. Además necesitaremos un Servidor Web, en nuestro caso usamos el Apache Tomcat 6.0, aunque cualquier servidor web, tendría que funcionar para poder hacer el deploy de nuestra aplicación.

Con esto seria posible ya hacer funcionar la aplicación, aunque aconsejamos también tener una Versión J2EE de Eclipse, en mi caso usé la Ganymedes, pero las demás versiones siempre que sean J2EE no tendrían que dar problemas de visualización.

7.2 Desarrollo de la Aplicación

La aplicación esta desarrollada en un 95-100% de su totalidad ahora mismo. El proyecto se podría dividir en 4 grandes partes:

-Gestión de Personas (Usuarios): Alta, baja, listado i modificación de los usuarios, todo ello desarrollado al 100%

-Gestión de Reparaciones: Alta, baja, listado i modificación de las reparaciones, todo ello desarrollado al 100%

-Gestión de Compras: Alta, baja, listado i modificación de las compras, todo ello desarrollado al 100% aunque de manera sencilla

-Login y Roles: El login teniendo en cuenta los usuarios creados están completamente implementado, i la parte de control de roles es completamente funcional (aunque a lo mejor no de la mejor forma, pero funciona bien).

7.3 Desarrollo de la Aplicación

En lo que a decisiones de diseño se refiere no se han tomado más que las estrictamente necesarias. El diseño ha sido muy fiel al original que se propuso como prototipo, cambiado simplemente algunos aspectos de diseño en lo referente a los formularios ya que tienen su propio “estilo”. En lo referente al diseño de la estructura, es exactamente el que ya se propuso.

En cambio, en lo que a implementación se refiere, se han hecho algunos cambios. Des de cambios de Software, en vez de utilizar Netbeans y JBoss se optó por utilizar Eclipse y Apache Tomcat. La decisión (recomendación del tutor) ayudó en sacar el proyecto adelante, ya que en el caso anterior, nos daba problemas el Servidor JBoss, debido a que él mismo tenía librerías que se usaban en el proyecto, cosa que se descubrió después, y fue lo que daba errores al principio.

En lo que a Datos y Estructura de la información, solo se ha variado la estructura de datos de las compras para que sea más fácil su gestión. No es nuestro caso hacer un programa de Contabilidad sino más bien para controlar un poco los mejores clientes y tener las garantías de los productos más caros siempre a mano.

7.4 Consideraciones a la hora de Instalar

Antes de poder empezar a usar nuestra aplicación, hay que crear una base de datos en el MySQL Server. Para eso la aplicación lleva un .sql con todas las sentencias necesarias mas 4 insert into para poder tener los 4 usuarios básicos. El DNI que equivale al nombre de usuario y la contraseña es la misma para facilitar el testeo. En caso de estar encendido se para el Tomcat. Se copia el War dentro de TOMCAT_HOME/webapps/ i se arranca el Tomcat. Para arrancar en la carpeta Bin, encontramos (en nuestro caso: C:\Archivos de programa\Apache Software Foundation\Tomcat 6.0\bin) el archivo tomcat6.exe que iniciará el servidor. Abrimos el navegador que mas nos guste i vamos a la dirección: http://localhost:8080/NaoruProject/

8 Conclusiones

Primero de todo, cabe destacar que los objetivos definidos en el 1.2 se han conseguido. Cuando elegí esta área del TFC sabía que disfrutaría con su diseño y sobre todo con su implementación. Era todo un reto coger el TFC de J2EE dado que solo tengo nociones de Java, todo lo demás lo desconocía pero a la vez me motivaban las posibilidades que este proyecto me podría suponer.

Una vez terminado el proyecto puedo decir que he aprendido muchas tecnologías interesantes y nuevas para mí. El hecho de no tener pautas u objetivos específicos que dan libertad al estudiante son un punto a favor dada la posibilidad de elección que se le da al estudiante, pero a la vez, tiene el inconveniente de no haber una pauta a seguir o un ejemplo con el que hacerse una idea de lo que se espera.

això que el TFC és la culminació dels meus estudis. Tot i les dificultats que he tingut degut a haver d'aprendre noves tecnologies, estic molt orgullós d'haver pogut realitzar un projecte d'aquesta envergadura partint des de zero. Un altre aspecte a remarcar del projecte és que l'estudiant no disposa de gairebé cap pauta ni guió per a la realització del projecte, es disposa de total llibertat per a l'elecció de les tecnologies i tot el referent al TFC. Això crec que fa créixer a l'estudiant com a professional, ja que es tracta d'un projecte quasi real.

9 Bibliografía

Booch, Grady; Rumbaugh, James; Jacobson, Ivar (1999). El lenguaje unificado de modelado. Addison Wesley.
Bauer, Christian; King, Gavin (2005). Hibernate In Action. Manning.
Cockburn, Alistair (2001). Writing Effective Use Cases. Addison-Wesley.
Connolly, Thomas; Begg, Carolyn (2005). Database Systems: A Practical Approach to Design, Implementation, and Management (4ª edición). Addison Wesley.
Gamma, Erich; Helm, Richard; Johnson, Ralph; Vlissides, John (1995). Patrones de Diseño. Addison Wesley.
Larman, Craig (2005). Applying UML and patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development (3ª edición). Prentice Hall.
Mukhar, Kevin; Zelenak, Chris (2006). Beginning Java EE 5 Platform: From Novice to Professional. Apress.
Murach, Joel; Steelman, Andrea (2008). Java Servlets and JSP (2ª edición). Murach.
Pressman, Roger S. (2006). Ingeniería del software: Un enfoque práctico (6ª edición). McGraw-Hill.
Sun Microsystems (2001-2002). "Core J2EE Patterns". Sun Developer Network. http://java.sun.com/blueprints/corej2eepatterns/Patterns/index.html.
Sun Microsystems (2007). The Java EE 5 Tutorial: For Sun Java System Application Server 9.1. Sun Microsystems.
Zambon, Giulio; Sekler, Michael (2007). Beginning JSP, JSF, and Tomcat Web Development: From Novice to Professional. Apress.

También se han utilizado toda la documentación y materiales disponibles en la UOC

Referencias de Internet.

IDE de desarrollo Eclipse.
http://www.eclipse.org/

Información sobre Tomcat, descarga de últimas versiones,…etc
http://tomcat.apache.org/

Información base de datos relacional y descarga de últimas versiones.
http://www.mysql.com/

Tecnología java, paquetes,…etc Sun Microsystems.
http://www.sun.com/

Framework e información de Struts2
http://struts.apache.org/2.x/index.html

Código java, ejemplos,…etc.
http://www.programacion.com/
http://viralpatel.net/blogs/
http://www.roseindia.net/