TFC – J2EE

Aplicación Web para la gestión de facturación de una empresa de cerrajería

Sara Gutiérrez Melero
ITIG
Junio de 2012

Consultor: Jose Juan Rodriguez

Índice

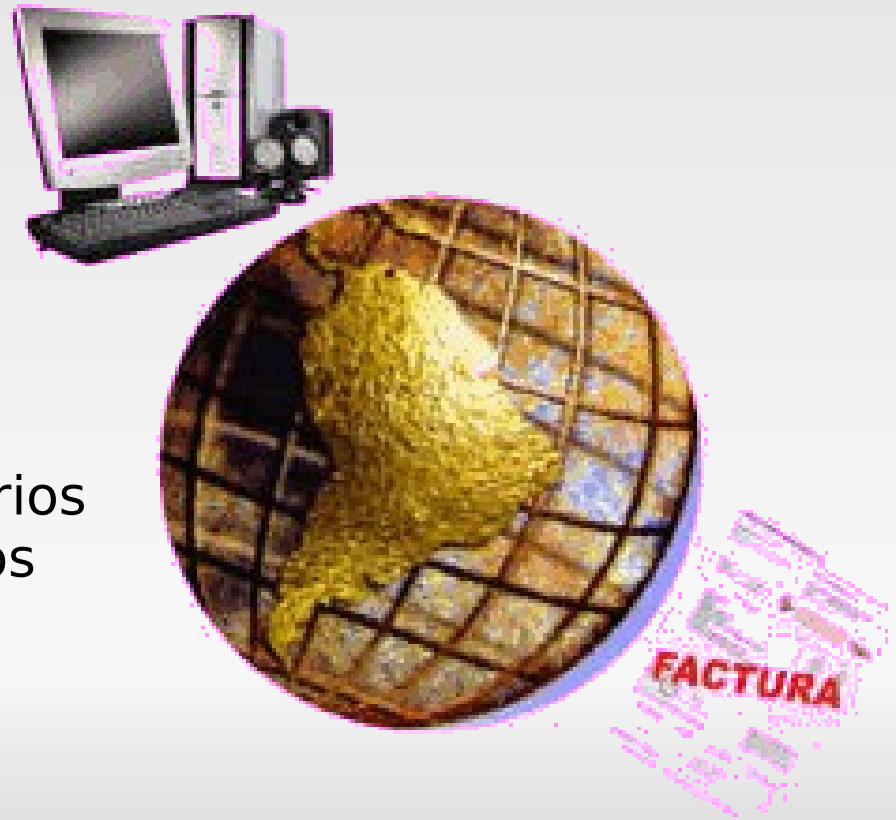
1. Introducción - Objetivos

- Crear una aplicación para una empresa de cerrajería que permita gestionar facturas, presupuestos, clientes y usuarios del sistema.

- Acceder a través de una red como Internet y desde cualquier navegador web.

- Permitir el acceso a los usuarios, previamente autentificados, a la creación y actualización de datos de la empresa.

- Permitir a uno o más administradores introducir, modificar o deshabilitar usuarios del sistema, así como actualizar los datos relativos a la misma empresa.



1. Introducción - Planificación

- PAC 1 – 14/03/12
Documento de Planificación del Proyecto

- PAC 2 – 18/04/12
Documento de Análisis funcional y
Diseño técnico del Proyecto

- PAC 3 – 04/06/12
Documento de Desarrollo e implantación del Proyecto

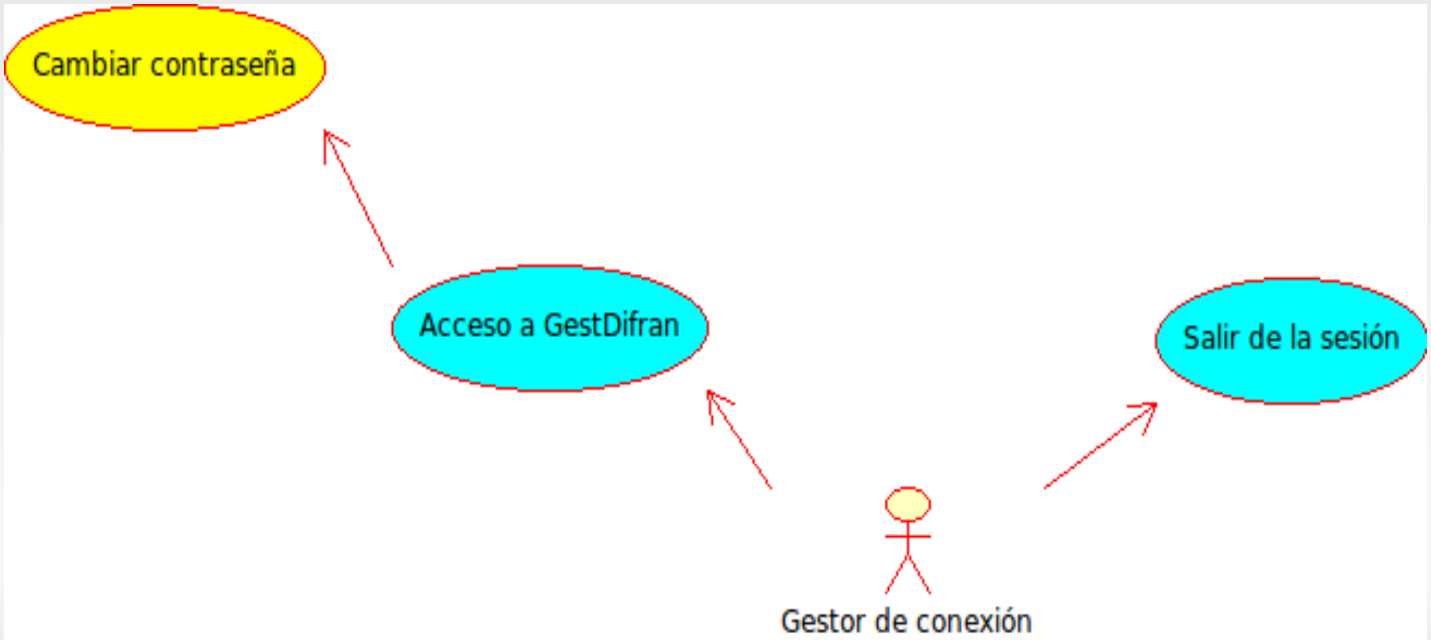
- PAC 4 – 18/06/12
Memoria Final del Proyecto
Presentación Virtual
Producto final

2. Análisis – Requisitos funcionales

La aplicación está compuesta por 4 subsistemas principales:

- **Subsistema de conexión**

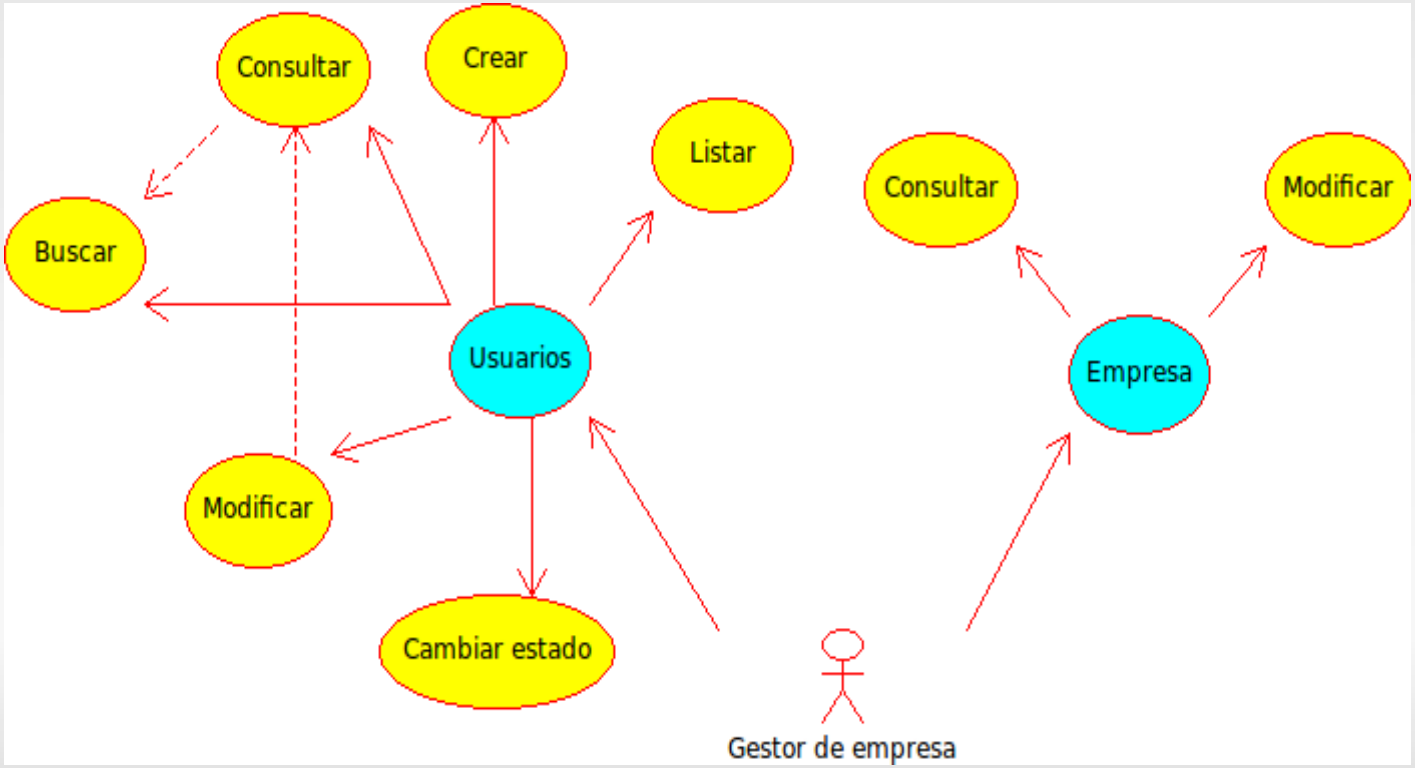
Es el encargado de gestionar los accesos a la aplicación así como las sesiones de usuario.
Un requisito imprescindible para acceder a la aplicación es autentificar-se previamente con un nombre de usuario y contraseña.
Existe la opción, para los usuarios, de cambiar sus contraseñas antes de iniciar sesión.



2. Análisis – Requisitos funcionales

- **Subsistema de gestión de empresa**

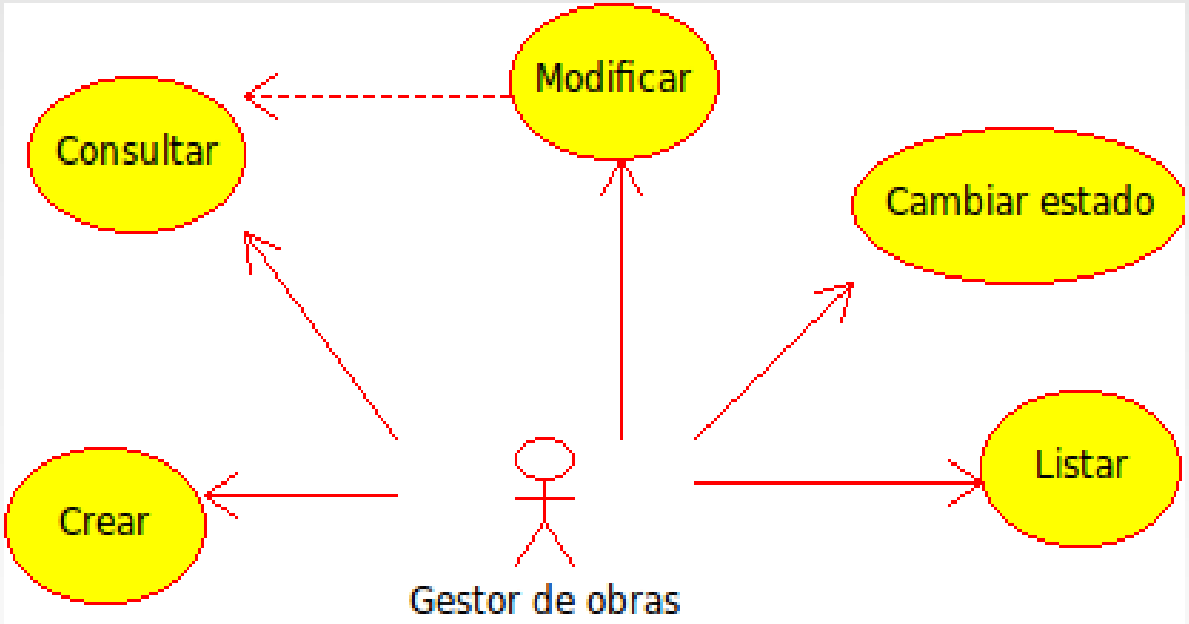
Éste se centra en los administradores de ésta que son los usuarios potenciales de la aplicación.
Mediante una interfaz amigable se podrá consultar y modificar los datos de empresa.
Existirá también una relación de usuarios con la cual se podrán crear, buscar (por NIF), consultar, modificar, activar e inactivar. Si un usuario se inactiva perderá todos los permisos de la aplicación.



2. Análisis – Requisitos funcionales

- **Subsistema de gestión de obras**

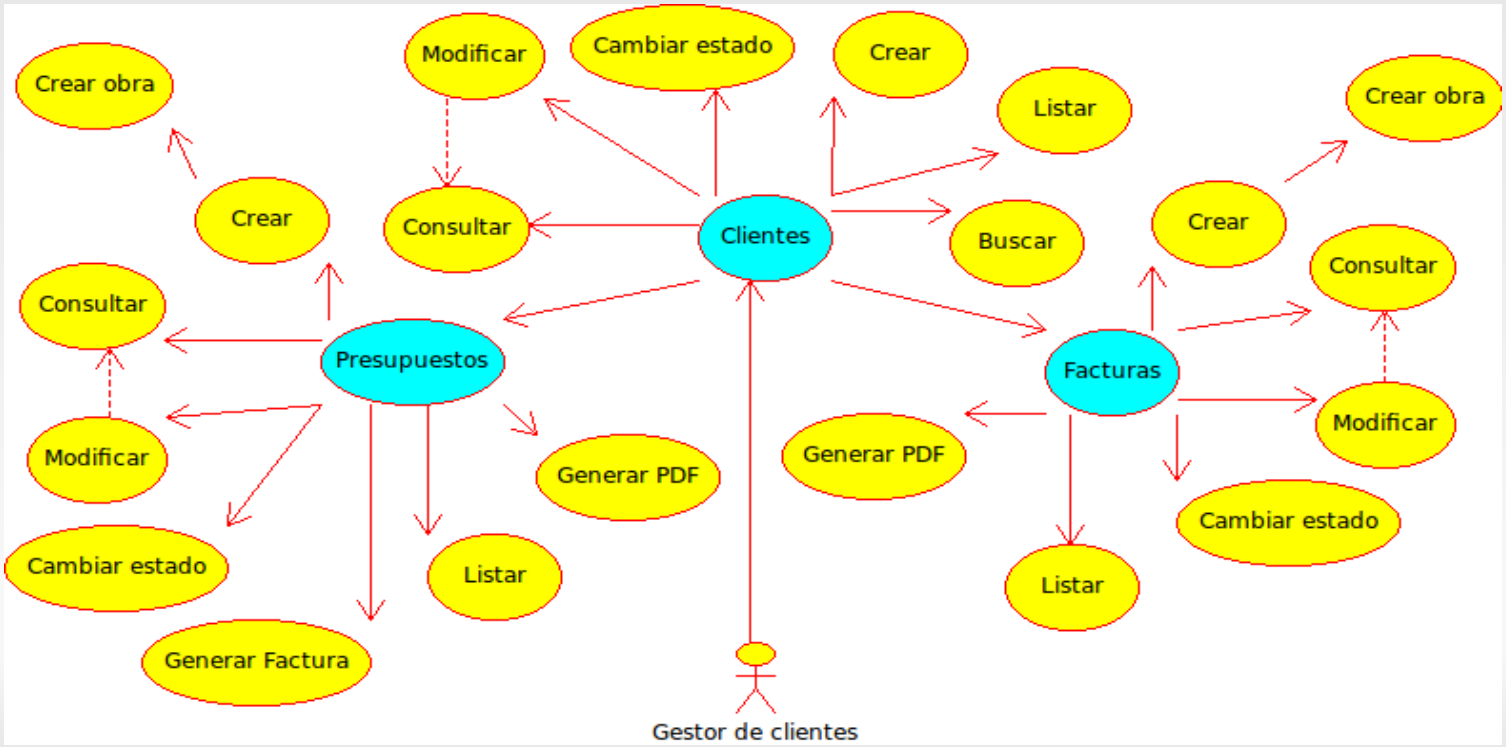
Permite la gestión de las obras de los clientes los cuales han contratado los servicios de la empresa.
Mediante una interfaz amigable se puede crear, consultar, modificar, activar e inactivar cualquier obra.



2. Análisis – Requisitos funcionales

- **Subsistema de gestión de clientes**

Es el subsistema principal de la aplicación. Mediante una interfaz amigable se podrá crear, consultar, buscar (por NIF), modificar, activar e inactivar clientes. Al mismo tiempo se permitirá visualizar las facturas o presupuestos asociados al cliente. Estos también se podrán crear, consultar, modificar, cambiar estado y generar en PDF. De cada presupuesto se podrá generar automáticamente una factura relativa a los trabajos realizados a un cliente, la cual irá asociada a este presupuesto.



2. Análisis - Actores

- En cualquier caso para acceder a la aplicación se necesita una autenticación previa.

- En la aplicación intervienen dos tipos de actores:

Administradores
Tienen permiso absoluto sobre cualquier función de la aplicación.

Empleados
Tienen acceso a la gestión de clientes y obras de la aplicación.

2. Análisis – Requisitos no funcionales

- **Calidad**

Para el correcto funcionamiento de la aplicación es necesario que ésta se mueva en unos márgenes de calidad adecuados, tales como integridad y seguridad, flexibilidad y portabilidad e interoperabilidad.

- **Carga**

Lograr unos parámetros de carga razonables. Podrán conectarse un mínimo de dos usuarios a la vez y se define un tiempo máximo de 10 segundos desde que se hace una petición hasta que se recibe una página de respuesta.

- **Coste**

Realizar una implementación con tecnologías de software libre para lograr un bajo coste de la aplicación.

- **Requerimientos tecnológicos**

Debe ser accesible desde cualquier sistema operativo y navegador web con resolución igual o superior a 800x600 píxeles.

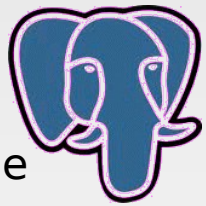
- **Interfaces**

Pantallas intuitivas y sencillas de usar. Información clara y ordenada. En resumen, las interfaces serán amigables y simples.

3. Diseño - Escenario

- **Java SE Development Kit (JDK)**
  Software que requieren las máquinas donde se desarrolle o se compile código en Java. En él se incluyen librerías básicas de Java y la máquina virtual.

- **Apache Tomcat**
  Encargado de tratar las peticiones del cliente y generar respuestas en consecuencia. Servidor libre, de código abierto e implementado en Java, lo que permite que pueda funcionar en cualquier sistema operativo que disponga de máquina virtual Java

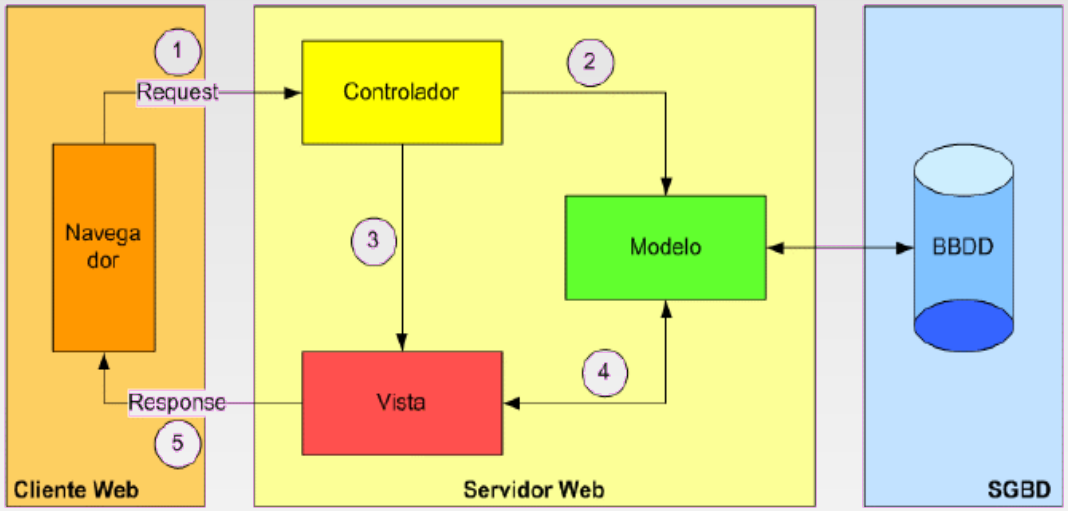


- **PostgreSQL y pgAdminIII**
  Para la creación y administración de la base de datos, se utiliza la herramienta propia del PostgreSQL, como es pgAdminIII. PostgreSQL ofrece una integridad de datos más fuerte que MySQL.

- **Eclipse**
  Plataforma con módulos base preparados para desarrollar en Java y otros extra encaminados a aplicaciones Web. Su uso es muy recomendable y además es una herramienta libre lo que no añade coste alguno al proyecto.

3. Diseño - MVC

Una interesante forma de trabajar es adaptar la aplicación a un patrón que se adapte a nuestras necesidades. El patrón modelo-vista-controlador empleado en el proyecto es una buena manera de modular la aplicación, separando los datos de la aplicación, de la interfaz del usuario y de la lógica de negocio.



Modelo: Se encarga de manipular los datos del programa de forma coherente y ofreciéndolos al programa a medida que los va requiriendo. No debe tener conocimiento ninguna ni del controlador ni de la vista.

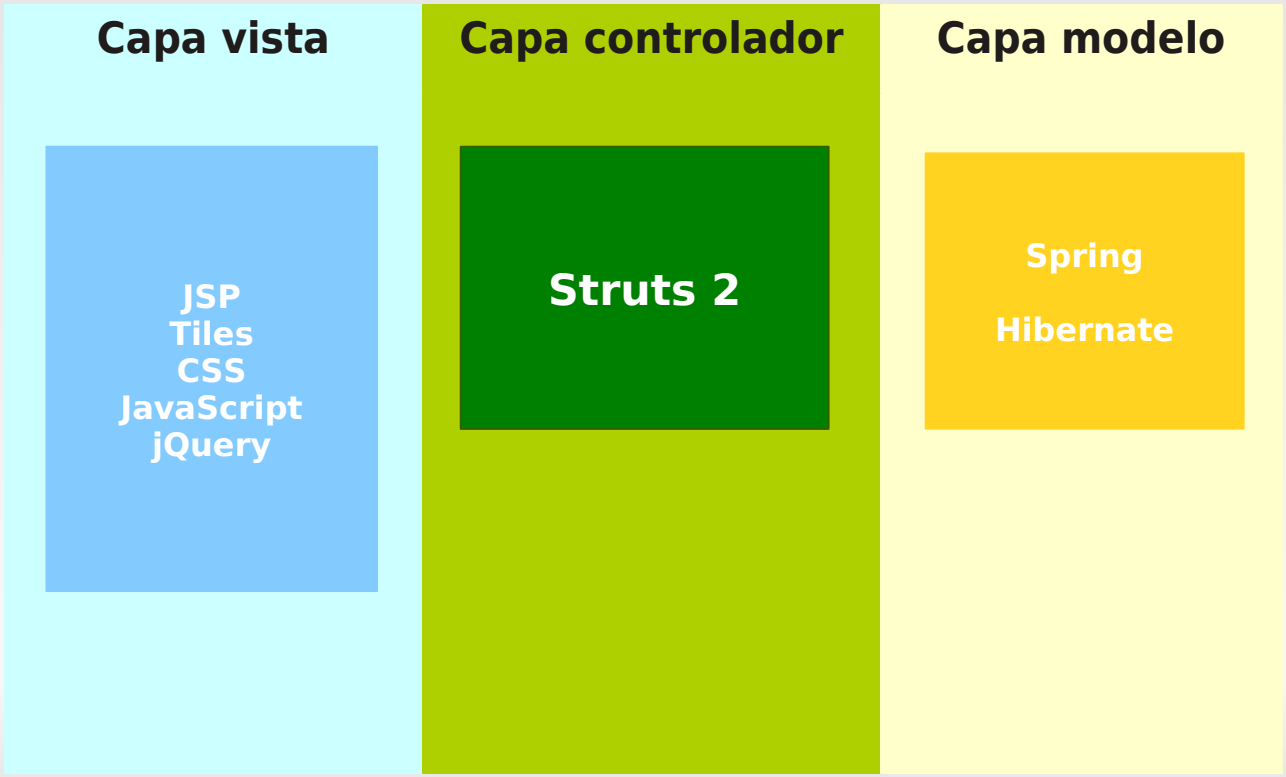
Vista: Representación visual de los datos. Capa mediante el usuario interactuará con la aplicación.

Controlador: Proporciona significado a las órdenes del usuario y crea la relación entre modelo y vista. Se encarga de decidir cuando se debe modificar la base de datos y cuando se deben obtener datos de ella para que los disponga la vista. Además se encarga de la lógica de redireccionamiento de pantallas.

3. Diseño – Arquitectura en capas de la aplicación

El desarrollo de la aplicación GestDifran se ha llevado a cabo siguiendo el patrón Modelo-Vista-Controlador (MVC) definido anteriormente.

En cada una de las capas se ha hecho uso de una o varias tecnologías de J2EE tal y como se puede observar en la siguiente imagen:

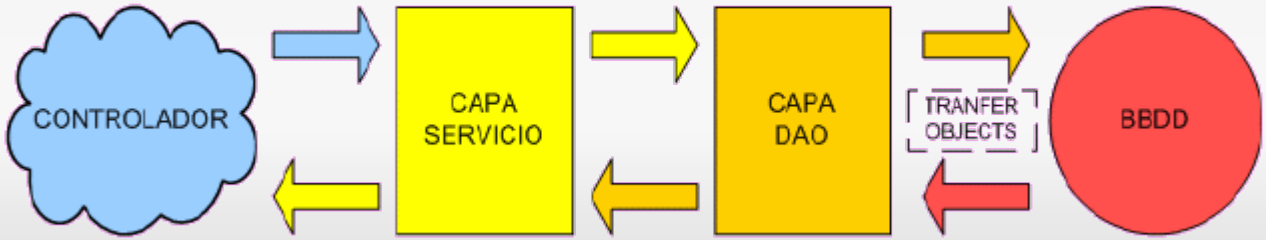


3. Diseño – Arquitectura en capas de la aplicación

CAPA MODELO

- **Spring**
  Spring cuenta con su propio marco de trabajo así como múltiples usos más para aplicaciones Web MVC, pero no es esa la parte que hemos necesitado para este proyecto, sino que nos hemos centrado en la parte de Spring que permite la inyección de dependencias. Muchos desarrolladores de Struts 2 consideran este servicio de gestión de recursos como parte esencial de una aplicación Web bien construida.

- **Hibernate**
  Nos permite convertir objetos en registros o viceversa sin necesidad de implementar complejas estructuras.
  Para realizar este proceso de mapeo se ha hecho uso de anotaciones directamente en las clases Java, aunque también existe otra manera configurando unos archivos que establezcan la relación donde cada fichero simulará una tabla de la base de datos.
  Hibernate también se encarga de llevar a cabo el patrón DAO, que contiene un seguido de clases DAO que serán las encargadas de realizar los accesos a base de datos.



3. Diseño – Arquitectura en capas de la aplicación

CAPA VISTA

- **Java Server Pages (JSP)**
  Por defecto, el marco de trabajo utiliza un tipo de resultados que funciona con páginas JSP para mostrar las páginas de respuesta.
  El lenguaje HTML es el encargado de tratar los temas más visuales de la página junto con una API de etiquetas (**taglibs**) que facilita Struts 2 y que potencian la funcionalidad para crear de forma dinámica páginas Web robustas.

- **Tiles**
  Herramienta que nos permite definir plantillas que se adaptan a nuestras necesidades para la distribución de contenido. Cada plantilla tendrá diversas secciones y cada sección tendrá asignada una jsp.

- **Cascading Style Sheets (CSS)**
  Se hace uso de CSS externas que permiten separar la estructura de una página web de su presentación. Éstas a su vez están validadas por W3C.

- **JavaScript & jQuery**
  Añade pequeñas funcionalidades a nivel de cliente (explorador).
  JavaScript incrementa la funcionalidad a las etiquetas de Struts 2 y aporta eficiencia Jquery, en nuestro caso, facilita la forma de validar un formulario sin tener la necesidad de recargar el documento, por lo que la aplicación consume menos recursos.

3. Diseño – Arquitectura en capas de la aplicación

CAPA CONTROLADOR

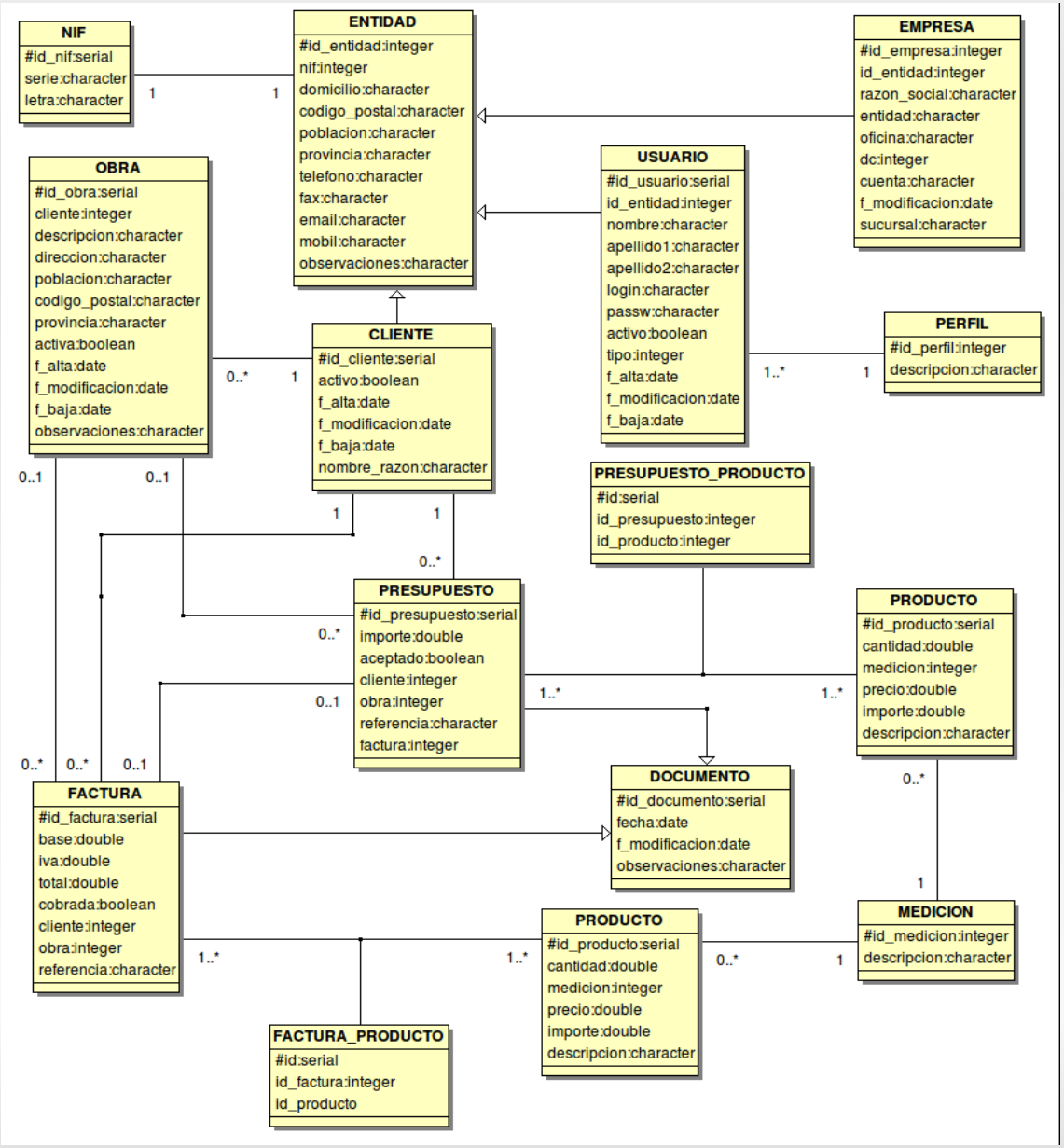
- **Apache Struts 2**
  El corazón de Struts 2 es un filtro, conocido como el StrutsPrepareAndExecuteFilter. Este es el punto de entrada del Framework. A partir de él se lanza la ejecución de todas las peticiones que lo involucran.
  Struts 2 procesa las peticiones usando tres elementos principales:

  - **Interceptores**
    Realizan tareas antes y después de la ejecución de un Action y también pueden evitar que un Action se ejecute.

  - **Acciones**
    Clases encargadas de realizar la lógica de negocio para servir una petición.
    El único requisito para que una clase sea considerada un `Action` es que debe tener un método que no reciba argumentos que regrese ya sea un `String` o un objeto de tipo `Result`. Aunque, también pueden implementar la interface "`com.opensymphony.xwork2.Action`" o extender una clase base que proporciona Struts 2, "`com.opensymphony.xwork2.ActionSupport`" (lo cual nos hace más sencilla su creación y manejo).

  - **Resultados**
    Y, después que un `Action` ha sido procesado se debe enviar la respuesta de regreso al usuario, esto se realiza usando `results`.

3. Diseño – Modelo Relacional



4. Implementación

Algunas de las clases implementadas

- **Login**
Encargada de gestionar los accesos al sistema y los inicios de sesión del usuario.

- **Password**
Encargada de gestionar los cambios de contraseñas.

- **Logout**
Encargada de gestionar las salidas de la aplicación así como los cierres de sesión del usuario.

- **Usuarios, Clientes, Facturas, Presupuestos**
Encargadas de gestionar los listados, las consultas y los cambios de estado. Para las facturas y presupuestos también se encargan de gestionar las modificaciones en los datos.

- **CrearUsuario, CrearCliente, CrearFactura, CrearPresupuesto**
Encargadas de gestionar las creaciones de nuevos objetos.

- **GuardarUsuario, GuardarCliente**
Encargadas de gestionar las modificaciones en los datos.

4. Implementación

PRUEBAS

- Se han realizado pruebas de diseño, navegabilidad, comprobación de enlaces rotos y accesibilidad.
- Validación CSS a través del validador W3C con la versión 3.
- Inserciones de prueba contra la Base de Datos.

CONFIGURACIÓN E INSTALACIÓN

- **Paso 1:** Importar a nuestro IDE el archivo gestdifran.war.
- **Paso 2:** Modificar el archivo “jdbc.properties del .war con el 'username' y 'password' correspondiente.
- **Paso 3:** Crear la BD “gestdifran” en PostgreSQL ejecutando el script “DML_gestdifran.sql” que se encuentra comprimido en el archivo sgutierrezmel_producto.zip.
- **Paso 4:** Ejecutar las instrucciones de inserción para la BD ejecutando esta vez el script “DDL_gestdifran.sql” para realizar una prueba inicial de la aplicación.
- **Paso 5:** Acceder a la aplicación con el nombre de usuario “*sarajoe*” y la contraseña “*123456*” si deseamos tener privilegios de Administrador o con el nombre de usuario “*manu*” y la contraseña “*holamanu*” si simplemente queremos permisos de Empleado.

5. Conclusiones

- La realización de este TFC ha constituido un gran reto para mí. Teniendo en cuenta la inexperiencia absoluta en la arquitectura J2EE y el tiempo limitado del que disponía, creo que el producto final ha sido muy satisfactorio.

- Tras investigar y recopilar información de las diversas tecnologías que componen J2EE y comprender que ésta ofrecía un patrón de diseño para las aplicaciones Web dividido en tareas, concretamente separando las capas de vista, modelo y controlador, decidí realizar la aplicación con el marco de trabajo Struts 2 utilizado para la capa de controlador, el Framework Spring para la creación de objetos e inyección de dependencias en la capa modelo junto con Hibernate y la tecnología JSP para la capa vista por ser simple y poder utilizar en ella tanto código HTML como las librerías de etiquetas de Struts 2 (taglibs).

- He intentado integrar diferentes tecnologías utilizando de cada una sus puntos fuertes. Por ejemplo, podría haber utilizado Struts 2 en lugar de Spring para la creación de objetos, no obstante, Spring es ideal para ello y se integra estupendamente con Struts 2.

- La experiencia obtenida en la realización de este TFC ha sido increíble y motivadora hacia propuestas futuras de proyectos.
  En apenas 4 meses he obtenido conocimientos de tecnologías que jamás había utilizado y ante todo he sido capaz de realizar un producto final al completo pasando por todas sus fases.