Escola Universitària
Politècnica de Mataró

Ingeniería Técnica de Telecomunicación: Especialidad Telemática

APLICACIÓN WEB ONG

CARLOS RODRÍGUEZ SANSÓN
J.M. GABRIEL SOLANILLA

OTOÑO 2009

A J.M. Gabriel por poner orden a tanto caos.

Y a mi familia, sobretodo a mi padre…

Resumen

Este proyecto explica el desarrollo de una aplicación web para una nueva ONG llamada “farts” (fondo de alto rendimiento para la transformación social).
En la misma aplicación se ha implementado una parte de gestión para la organización y otra para la comunicación de los clientes con el sistema. La web se ha desarrollado con servlets y jsp’s en un servidor Tomcat. Para la gestión de los datos se ha elegido MySQL.

Resum

Aquest projecte explica el desenvolupament d’una aplicació web per a una nova ONG anomenada “farts” (fons d’alt rendiment per a la transformació social).
En la mateixa aplicació s’ha implementat una part de gestió per a la organització i un altra per a la comunicació dels clients amb el sistema. La web s’ha desenvolupat amb servlets i jsp’s en un servidor Tomcat. Per a la gestió de les dades s’ha escollit MySQL.

Abstract

This project explores the development of a web application for a new NGO called “farts” (high-yield fund for social transformation).
One module for organization management and another for client communication with the system are deployed within the same application. The website was developed with servlets and jsp's in a Tomcat server. MySQL was chosen for data management.

Índice

1 Introducción

En un principio la idea del proyecto era crear un software de gestión económica y de recursos de una ONG ya existente, aunque lo suficientemente pequeña como para no contar con mucho dinero para contratar personal o los servicios de una empresa que les hiciera una aplicación.

Este proyecto se llama “farts” (fondo de alto rendimiento para la transformación social) y, en resumidas cuentas, trata de recaudar un fondo mediante donaciones puntuales, cuyo importe se ingresa en una cuenta de banca ética que dedica los intereses a fines sociales o para ayudas a organizaciones sin ánimo de lucro.

Al ser donaciones puntuales, se puede conseguir un volumen grande de donaciones. En el momento en que la cantidad de la cuenta empiece a ser grande, con los intereses del banco se pueden empezar a iniciar pequeños proyectos.

Este trabajo se centra sólo en la aplicación web. El marketing y el estudio de la viabilidad va por parte de la ONG.

Para realizar este proyecto se han estudiado las tecnologías más utilizadas y finalmente se ha escogido la opción de Servlets-JSP bajo un servidor Tomcat. Basado en código HTML con posibilidad de inserción de código Java. Por lo tanto por una parte se trabaja en un lenguaje tan conocido HTML y por otra se puede utilizar la gran versatilidad de Java. Se ha escogido esta tecnología también porque no existían proyectos realizados en este lenguaje.

Para la gestión de los datos de la organización se utiliza MySQL. Desde la aplicación principal se podrá interactuar con la base de datos gracias a un conector JDBC, implementado en una clase Java independiente del resto de la aplicación que realiza la conexión y las operaciones pertinentes.

Los componentes (beans) de Java y el objeto “Session” de los Servlets, permiten llevar un control de las sesiones iniciadas en el sistema tanto si son de un donante como de un

administrador. De la misma manera, en el objeto “Request” de los Servlets se insertan los datos sacados de la base de datos para que más tarde se extraigan en los JSP.

Se implementarán también los métodos “JavaMail” y “FileUpload” de Java para el envío de correos electrónicos y para poder adjuntar imágenes respectivamente.

En este proyecto también se explicará como implementar el protocolo de seguridad HTTPS en Tomcat mediante claves certificadas. Y finalmente se mostrarán las diferentes opciones que hay para subir el proyecto a la red, ya que contratando un servicio de “Hosting” no se puede implementar el protocolo de seguridad, por eso también se estudiarán las opciones de “VPS” y “Housing o Colocation”.

2 Definición de requisitos

Según las indicaciones de “farts” la aplicación debería de tener dos partes bien diferenciadas: una parte para la gestión del sistema mediante los administradores y otra para la interacción de los usuarios con las opciones del sistema.

2.1 Parte cliente

El cliente debe entrar en el sistema y poder hacer una donación mínima de 10 euros. Puede hacerla mediante Paypal o con un ingreso en una cuenta bancaria, a la vieja usanza. En principio no se va a crear una pasarela de pago (VTP) ya que es demasiado cara para la organización.

Cuando el usuario haya donado el dinero y haya una confirmación por parte de los administradores del sistema, éste podrá votar el proyecto que más le guste, por lo tanto, los proyectos que salen adelante son elegidos por los usuarios.

También debe de haber una opción para que el usuario pueda pedir el reintegro de la cantidad total donada en el sistema aunque hay pocas posibilidades de que se use ya que la donación tipo va a ser la mínima (10 euros).

Se implementará también una zona de noticias, otra de información personal, una opción de contacto con la administración vía correo electrónico y otra de recuperación de contraseña en el sistema.

2.2 Parte administración

En ésta parte los administradores en primer lugar podrán verificar las notificaciones de donación o de nuevo usuario llegadas por el banco o por Paypal, cotejándolas con los datos insertados en el sistema por cada donante.

De forma similar, los administradores podrán ver los usuarios que han pedido el reintegro de la cantidad donada y la confirmarán cuando éstá se lleve a cabo, haciendo que el usuario pierda su cuenta y ya no pueda entrar en el sistema.

Por otro lado se deberá incluir una opción de información general del sistema; con la cantidad de donantes, la suma total de todas las donaciones del sistema y los votos que lleva cada proyecto en cada tanda de votaciones.

Habrá diferentes opciones de búsqueda de socios: apellido, ciudad, según la cantidad donada, sean de España o de fuera, etc…

También constará un buscador de correos electrónicos de usuarios utilizando distintos tipos de búsquedas

Debe existir una opción de gestión de los proyectos, en la que se podrá crear un proyecto nuevo para la votación o borrar entera la tanda de votaciones.
Por último, se incluye un apartado de gestión de noticias en el que se podrán crear noticias o borrarlas.

3 ¿Qué tecnología utilizar?

Tras haber estudiado el pliego de requisitos de la organización ya se tienen bastante claras las características esenciales de la aplicación. Para elegir la tecnología que se utilizará en el proyecto primero se analizarán las ventajas y desventajas de cada una de las tecnologías más utilizadas.

3.1 Estudio de tecnologías existentes

3.1.1 Lenguaje HTML

Desde el nacimiento de Internet se han publicado sitios web gracias al lenguaje HTML. Es un lenguaje estático para el desarrollo de sitios web (acrónimo en inglés de HyperText Markup Language). Desarrollado por el World Wide Web Consortium (W3C).

Ventajas

- Sencillo, permite describir hipertexto.
- Texto presentado de forma estructurada y agradable.
- Archivos pequeños.
- Despliegue rápido.
- Lenguaje de fácil aprendizaje.
- Lo admiten todos los exploradores.

Desventajas

- Lenguaje estático.
- La interpretación de cada navegador puede ser diferente.
- Guarda muchas etiquetas que pueden convertirse en “basura” y dificultan la corrección.
- El diseño es más lento.
- Las etiquetas son muy limitadas.

3.1.2 Lenguaje Javascript

Es un lenguaje interpretado, no requiere compilación. Fue creado por Brendan Eich en la empresa Netscape Communications. Utilizado principalmente en páginas web. Es similar a Java, aunque no es un lenguaje orientado a objetos. La mayoría de los navegadores en sus últimas versiones interpretan código Javascript.

El código Javascript puede ser integrado dentro de nuestras páginas web. Para evitar incompatibilidades el World Wide Web Consortium (W4C) diseñó un estándar denominado DOM (Document Object Model).

Ventajas

- Lenguaje de scripting seguro y fiable.
- Los script tienen capacidades limitadas, por razones de seguridad.
- El código Javascript se ejecuta en el cliente.

Desventajas

- Código visible por cualquier usuario.
- El código debe descargarse completamente.

3.1.3 Lenguaje PHP

Es un lenguaje de programación utilizado para la creación de sitios web. PHP es un acrónimo recursivo que significa “PHP Hypertext Pre-processor”, (inicialmente se llamó Personal Home Page). Surgió en 1995, desarrollado por PHP Group.

PHP es un lenguaje de script, incrustado en las páginas HTML, interpretado en el lado del servidor y utilizado para la generación de páginas web dinámicas. PHP no necesita ser compilado para ejecutarse. Para su funcionamiento necesita tener instalado Apache o IIS con las librerías de PHP. La mayor parte de su sintaxis ha sido tomada de C, Java y Perl con algunas características específicas de cada una. Los archivos cuentan con la extensión “.php”.

Ventajas

- Muy fácil de aprender.
- Se caracteriza por ser un lenguaje muy rápido.
- Soporta en cierta medida la orientación a objetos, clases y herencias.
- Es un lenguaje multiplataforma: Linux y Windows entre otros.
- Capacidad de conexión con la mayoría de los manejadores de bases de datos: MySQL, PostgreSQL, Oracle, MS SQL Server, entre otros.
- Capacidad de expandir su potencial utilizando módulos.
- Es libre, por lo que se presenta como una alternativa de fácil acceso para todos.
- No requiere definición de tipos de variables ni manejo detallado de bajo nivel.

Desventajas

- Se necesita instalar un servidor web.
- Todo el trabajo lo realiza el servidor y no delega al cliente. Por tanto puede ser más ineficiente a medida que las solicitudes aumenten de número.
- La legibilidad del código puede verse afectada al mezclar sentencias HTML y PHP.
- La programación orientada a objetos es aún muy deficiente para aplicaciones grandes.
- Dificulta la modularización.
- Dificulta la organización por capas de la aplicación.

3.1.4 Lenguaje ASP.NET

Es un lenguaje comercializado por Microsoft, y usado por programadores para desarrollar entre otras funciones, sitios web. ASP.NET es el sucesor de la tecnología ASP, que fue lanzada al mercado mediante una estrategia de mercado denominada .NET.

ASP.NET fue desarrollado para resolver las limitaciones de ASP.

Ventajas

- Completamente orientado a objetos.
- Controles de usuario personalizados.
- División entre la capa de aplicación y el código.
- Facilita el mantenimiento de grandes aplicaciones.
- Incremento de velocidad de respuesta del servidor.
- Mayor velocidad y seguridad.

Desventajas

- Mayor consumo de recursos.

3.1.5 Lenguaje JSP

Es un lenguaje para la creación de sitios web dinámicos, acrónimo de “Java Server Pages”. Está orientado a desarrollar páginas web en Java. JSP es un lenguaje multiplataforma creado para ejecutarse del lado del servidor.

JSP fue desarrollado por Sun Microsystems. Comparte ventajas similares a las de ASP.NET; desarrollado para la creación de aplicaciones web potentes. Posee un motor de páginas basado en los servlets de Java. Para su funcionamiento se necesita tener instalado un servidor Tomcat.

Características

- Código separado de la lógica del programa.
- Las páginas son compiladas en la primera petición.
- Permite separar la parte dinámica de la estática en las páginas web.
- El código JSP puede ser incrustado en código HTML.

Ventajas

- Ejecución rápida.

- Crear páginas del lado del servidor.
- Multiplataforma.
- Código bien estructurado.
- Integridad con los módulos de Java.
- La parte dinámica está escrita en Java.

3.1.6 Lenguaje Ruby

Es un lenguaje interpretado de muy alto nivel y orientado a objetos. Desarrollado en el 1994 por el programador japonés Yukihiro “Matz” Matsumoto. Su sintaxis está inspirada en Phyton, Perl. Ruby es un lenguaje dinámico para una programación orientada a objetos rápida y sencilla. Además es de distribución libre.

Características

- Existe diferencia entre mayúsculas y minúsculas.
- Múltiples expresiones por líneas, separadas por punto y coma “;”.
- Dispone de manejo de excepciones.
- Ruby puede cargar librerías de extensiones dinámicamente si el sistema operativo lo permite.

Ventajas

- Permite desarrollar soluciones a bajo coste.
- Software libre.
- Multiplataforma.

3.2 Tecnología escogida

Finalmente para la realización del proyecto hemos escogido la opción de Servlets y JSP, bajo un servidor Tomcat (servidor web que soporta Java) y apoyado en una base de datos MySQL.

3.2.1 ¿Porqué JSP?

Los Servlets Java son más eficientes, fáciles de usar, más portables, y más baratos que el CGI tradicional y otras muchas tecnologías del tipo CGI.

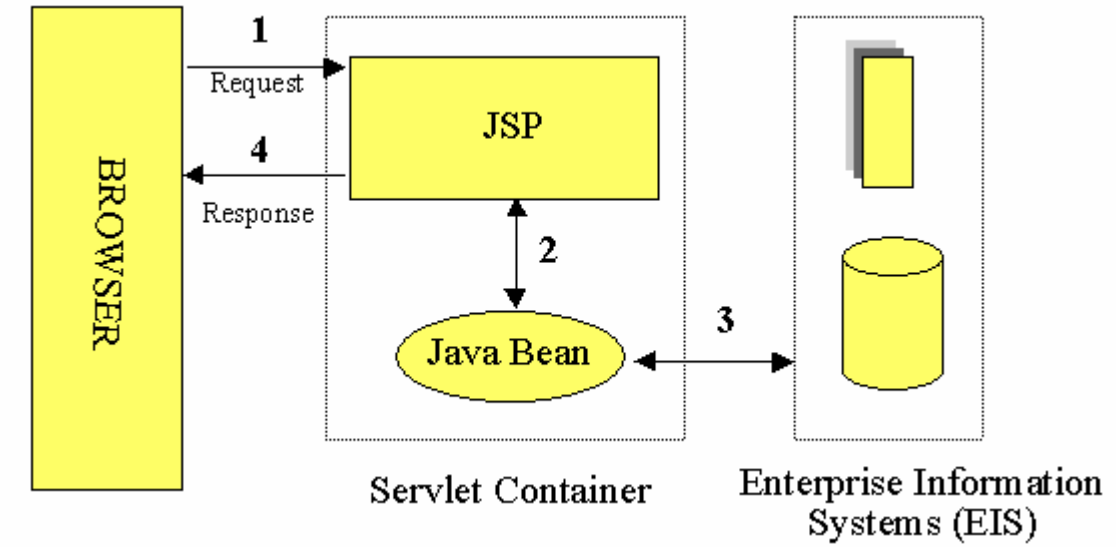


Figura 3.1 Esquema funcionamiento tecnología JSP

Como se ve en la figura anterior, en general, el esquema de funcionamiento de una aplicación bajo Servlets y JSP es muy sencilla. Los usuarios de la aplicación se comunican con ésta mediante objetos “request" y “response”. Las variables que se guardan en éstos objetos tienen poco tiempo de vida (lo que se tarda en mostrar la información al usuario mediante JSP). Si se quieren guardar variables para el control de los usuarios que entran en el sistema, se utilizarán objetos “session” para guardarlas.

Estos objetos “session” permanecen activos con los datos hasta que el usuario cierra la sesión o directamente se cierra el navegador. Los JSP no son más que HTML en el cual se puede insertar fragmentos de código Java.

Para la comunicación de nuestra aplicación con la base de datos se utiliza Java. Para las sesiones se emplean “beans”, donde se guarda la información primordial de cada usuario. Para la comunicación con MySQL se utiliza una clase Java que se dedica exclusivamente a hacer las consultas y a recoger los datos, para ello se tendrá que implementar un conector de Java-MySQL.

3.3 Ventajas de los servlets frente a CGI tradicionales

Eficiencia

Con un CGI tradicional, se arranca un nuevo proceso para cada solicitud HTTP. Si el programa hace una operación relativamente rápida, la sobrecarga del proceso de arrancada puede dominar el tiempo de ejecución. Con los Servlets, la máquina Virtual Java permanece arrancada, y cada petición es manejada por un thread Java de peso ligero, no un pesado proceso del sistema operativo.

De forma similar, en un CGI tradicional, si hay N peticiones simultáneas para el mismo programa, el código se cargará N veces en memoria. Sin embargo, con los Servlets, hay N threads pero sólo una copia de la clase Servlet. Los Servlet también tienen más alternativas que los programas normales para optimizaciones; como cachés de cálculos previos, mantener abiertas las conexiones de bases de datos, etc…

Conveniencia

Lo más importante es la posibilidad de utilizar Java. Aparte, los Servlets tienen una gran infraestructura para análisis automático y decodificación de datos de formularios HTML, leer y seleccionar cabeceras HTTP, manejar “cookies”, seguimiento de sesiones, y muchas otras utilidades.

Potencia

Los Servlets Java permiten fácilmente hacer muchas cosas que son difíciles o imposibles con un CGI normal, éstos pueden interactuar directamente con el servidor web. Ésto simplifica las operaciones que se necesitan para buscar imágenes y otros datos almacenados en situaciones estándar. Los Servlets también pueden compartir los datos entre ellos, haciendo cosas útiles como almacenes de conexiones a bases de datos fáciles de implementar. También pueden mantener información de solicitud en solicitud, simplificando los seguimientos de sesiones y el caché de cálculos anteriores.

Portabilidad

Los Servlets están escritos en Java y siguen el API estándar. Consecuentemente, los servlets escritos, digamos en el servidor I-Planet Enterprise, se pueden ejecutar sin modificarse en Apache, Microsoft IIS o WebStar. Los Servlets están soportados directamente o mediante un plug-in en la mayoría de los servidores web.

Precio

Hay un gran número de servidores web gratuitos o muy baratos que son buenos para el uso de ésta tecnología. Sin embargo, con la excepción de Apache y Tomcat, la mayoría de los servidores web comerciales son relativamente caros.

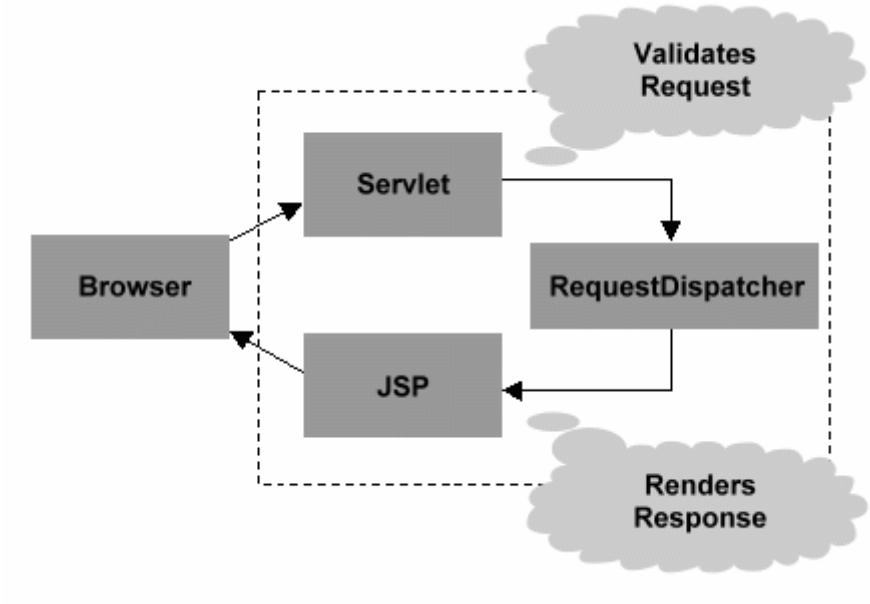


Figura 3.2 Esquema comunicación entre servlets y jsp's

Como se puede apreciar en la imagen anterior se dispone de un método llamado "requestDispatcher", éste envía la aplicación al JSP o Servlet que se indíque en la llamada.

4 Diseño de la aplicación

En este capítulo se explicará detalladamente como se ha llevado a cabo el diseño de la aplicación web, desde las bases de datos hasta la presentación final, pasando por las diferentes opciones de interactuar con los datos gracias a los Servlets y a los JSP.

4.1 Base de datos

Para todo lo que es gestión de datos se utiliza MySQL, por la sencillez y porque es el sistema más utilizado y conocido por el público general. Se utiliza el fichero ".sql" siguiente:

```
drop table if exists Administrador, Donante, Votante, Votacion, Pago, Reintegro, Noticia ;
```

Este fragmento de código se encarga de borrar las tablas a añadir por si existe alguna con el mismo nombre.

```
CREATE TABLE Administrador (
        alias VARCHAR(40) NOT NULL,
        password VARCHAR(15) NOT NULL,
        PRIMARY KEY (alias)
)ENGINE = INNODB;


CREATE TABLE Donante (
        alias VARCHAR(40) NOT NULL,
        password VARCHAR(15) NOT NULL,
        PRIMARY KEY (alias)
)ENGINE = INNODB;
```

Se crean las tablas que controlarán el acceso de la administración y de los usuarios al sistema, con "alias" y "password". "Innodb" es un modo de procesar más rápido las bases de datos que por defecto.

```
CREATE TABLE Votacion (
        id INT(4) NOT NULL auto_increment,
        nombre VARCHAR(100) NOT NULL,
        descripcion VARCHAR(10000) NOT NULL,
        votos INT(8) NOT NULL,
        url VARCHAR(100) NOT NULL,
        PRIMARY KEY (id)
)ENGINE = INNODB;
```

La tabla “Votacion” contiene el nombre, descripción, número de votos y localización de la fotografía adjunta a cada proyecto. Sólo existe una tanda de votaciones en el sistema, por lo tanto, si se crea otra votación se borrarán los datos de los proyectos de anteriores votaciones.

Para la identificación de cada proyecto se utiliza un entero con auto-incremento, este método se utiliza cuando importa más encontrar lo que se busca que la identificación en sí de cada fila de la tabla. El primer proyecto tendría el identificador 1, y así sucesivamente, pero si se borra por ejemplo el proyecto con id=4 y ya se va por el id=21, el siguiente proyecto insertado llevará id=22, el id=4 en este caso se perdería.

```
CREATE TABLE Noticia (
        id INT(5) NOT NULL auto_increment,
        nombre VARCHAR(100) NOT NULL,
        descripcion VARCHAR(10000) NOT NULL,
        url VARCHAR(100) NOT NULL,
        PRIMARY KEY (id)
)ENGINE = INNODB;
```

La tabla “Noticia”·contiene una lista de noticias. Cada una de ellas consta de: nombre, descripción y localización de la foto adjunta a la misma.

```
CREATE TABLE Votante (
        id INT(8) NOT NULL auto_increment,
        nombre VARCHAR(40) NOT NULL,
        apellido1 VARCHAR(40) NOT NULL,
        apellido2 VARCHAR(40) NOT NULL,
        ciudad VARCHAR(40) NOT NULL,
        pais VARCHAR(40) NOT NULL,
        dni VARCHAR(20) NOT NULL,
        email VARCHAR(40) NOT NULL,
        telefono VARCHAR(40) NOT NULL,
        aportado FLOAT(6,2) NOT NULL,
        fecha DATE NOT NULL,
        alias VARCHAR(40) NOT NULL,
        password VARCHAR(15) NOT NULL,
        activado enum('si','no') NOT NULL,
        voto enum('si','no') NOT NULL,
    PRIMARY KEY (id)
)ENGINE = INNODB;
```

La tabla “Votante” contiene todos los datos de los usuarios o donantes del sistema, aparte de los datos personales de cada usuario. El campo llamado “aportado” contiene la cantidad de dinero total donada en la asociación por cada usuario.

El campo “fecha” contiene la fecha de ingreso en el sistema. “alias” y “password” pertenecen a la cuenta que tiene el usuario para entrar en el sistema. Ya que sólo podrán votar los proyectos los socios que hayan sido confirmados por haber hecho una donación.

Con “activado” se sabe si el usuario ha sido confirmado por los administradores. Si el usuario ha pagado la cuota éstos lo activarán y ya podrá entrar al sistema con el “alias” y el “password” que haya escrito en el formulario.

Por último, con “voto” se sabe si el usuario ha votado ya o no en la tanda actual de votaciones. Si ya lo ha hecho no podrá volver a votar hasta que se implemente una nueva tanda de votaciones.

```
CREATE TABLE Pago (
        id INT(8) NOT NULL auto_increment,
        nombre VARCHAR(40) NOT NULL,
        apellido1 VARCHAR(40) NOT NULL,
        apellido2 VARCHAR(40) NOT NULL,
        ciudad VARCHAR(40) NOT NULL,
        pais VARCHAR(40) NOT NULL,
        dni VARCHAR(20) NOT NULL,
        aportado FLOAT(6,2) NOT NULL,
        fecha DATE NOT NULL,
    PRIMARY KEY (id)
)ENGINE = INNODB;
```

La tabla “Pago”controla las nuevas donaciones que hacen los socios ya existentes en el sistema, se indican datos personales del sujeto, la cantidad aportada en la nueva donación y la fecha en la cual se efectuó.

```
CREATE TABLE Reintegro (
        id INT(8) NOT NULL auto_increment,
        nombre VARCHAR(40) NOT NULL,
        apellido1 VARCHAR(40) NOT NULL,
        apellido2 VARCHAR(40) NOT NULL,
        ciudad VARCHAR(40) NOT NULL,
        pais VARCHAR(40) NOT NULL,
        dni VARCHAR(20) NOT NULL,
        email VARCHAR(40) NOT NULL,
        telefono VARCHAR(40) NOT NULL,
        aportado FLOAT(6,2) NOT NULL,
        fecha1 DATE NOT NULL,
        fecha2 DATE NOT NULL,
        descripcion VARCHAR(1000) NOT NULL,
    PRIMARY KEY (id)
)ENGINE = INNODB;
```

“Reintegro” indica a la administración que un usuario quiere recuperar todo el dinero aportado a la organización. Se obtienen todos los datos personales, la fecha de alta y la de realización del reintegro. Finalmente el campo “descripcion” indica la forma como el usuario quiere que se le devuelva el dinero (Paypal, transferencia bancaria, etc…).

```
INSERT INTO Administrador VALUES ('adminfarts', '1234');
```

Por último se inserta un valor a la tabla “Administrador” que permite administrar la aplicación.

4.2 “Beans” Java

La especificación de JavaBeans de Sun Microsystems los define como "componentes de software reutilizables que se pueden manipular visualmente en una herramienta de construcción".

Las convenciones requeridas son:

- Debe tener un constructor sin argumentos.
- Sus propiedades deben ser accesibles mediante métodos “get” y “set” que siguen una convención de nomenclatura estándar.
- Debe ser serializable.

Dentro de un JavaBean se pueden distinguir tres partes:

- Propiedades: Los atributos que contiene.
- Métodos: Se establecen los métodos “get” y “set” para acceder y modificar los atributos.
- Eventos: Permiten comunicarse con otros JavaBeans

En esta aplicación los beans son bastante importantes, ya que son utilizados para las sesiones de los Servlets. Se implementan dos: “Administradores.java” y “Donantes.java”.

Los dos tienen las siguientes características: poder crear, cerrar y comprobar una sesión. “Administradores” crea un objeto del tipo administradores con “alias” y “password”. “Donantes” por el contrario utiliza la información de los usuarios del sistema.

A continuación se presenta el ejemplo de “Administradores”; se implementa el constructor y los métodos “get” y “set”:

```
package beans;
import conexion.*;
import servlets.*;


public class Administradores{
  /*Alias del usuario.*/
  private String alias;
  /*Contraseña del usuario*/
  private String password;


  /*Constructor*/
  public Administradores(String alias, String password){
    this.alias = alias;
    this.password = password;
  }
   public Administradores() {
   }
   public void setAlias (String alias){
    this.alias = alias;
  }

  public String getAlias(){
    return this.alias;
  }


  public void setPassword (String password){
    this.password = password;
  }

  public String getPassword(){
    return this.password;
  }



}
```

4.3 Gestión de datos

Como ya se ha explicado en un capítulo anterior, para la gestión de datos se utiliza una base de datos MySQL con un conector de java (JDBC).

El código utilizado para realizar la conexión se implementa en una clase que sólo tiene métodos para interactuar con la base de datos. De esta forma se separa lo que es la presentación y recogida de información de los formularios con las operaciones en la base de datos. Así la aplicación es más portable y se puede reutilizar la clase para otras tecnologías.

La clase "GestionDatos.java" permite también realizar conexiones a diferentes bases de datos como se ve en la siguiente figura.

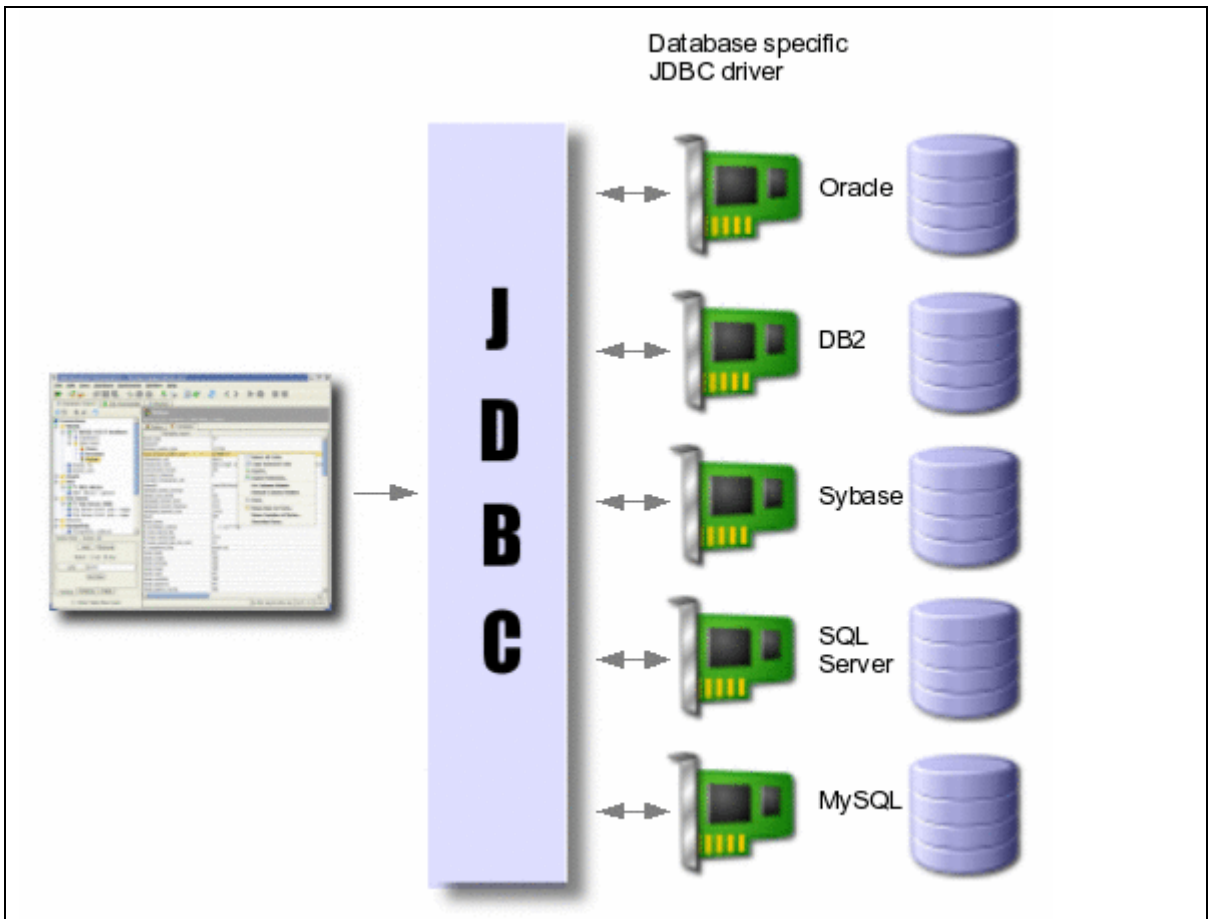


Figura 4.1 Esquema funcionamiento JDBC

4.3.1 Conexión y desconexión

Para la conexión con la base de datos, se utiliza el siguiente conector de la clase GestionDatos, que realiza la conexión junto a un control de posibles errores:

```
public class GestionDatos{

  /**Conexion, contenedor sentencia mysql y objeto contenedor del resultado de la
consulta.*/
  private java.sql.Connection conn;
  private Statement st;
  private ResultSet rs;

  /**Constructor de la clase, donde se crea la conexion sobre la cual se realizaran las
consultas, con nombre usuario y contraseña de la BBDD

   * @return objeto de la clase.
   */
  public GestionDatos(){
    try{
      Class.forName("com.mysql.jdbc.Driver").newInstance();
      conn =
DriverManager.getConnection("jdbc:mysql://localhost/nombreBBDD","usuario","pass");
    }
    catch (SQLException exc){
      exc.printStackTrace();
      System.out.println("Error con la BBDD. Revisar sentencias o datos.");
    }
    catch (Exception exc){
      exc.printStackTrace();
      System.out.println("Error al cargar driver, al establecer la conexion.");
    }
  }
```

Para la desconexión de la “BBDD”, se utiliza el método “cerrar()” de la clase GestionDatos. Se implementa una excepción por si existe algún error:

```
  public void cerrar(){
    try{
      conn.close();
      System.out.println("se cerro");
    }
    catch (Exception exc){
      exc.printStackTrace();
      System.out.println("Error cerrando conexion."); } }
```

4.3.2 Recogida de datos

A continuación se muestra un ejemplo tipo de recogida de datos, “verDonante(alias)”. Primero inserta en una cadena de texto la petición que se quiere hacer a la base de datos, más tarde los datos se recogen en la variable “rs”, éstos se copian posteriormente al objeto que devuelve el método. Se implementa también un control de posibles errores:

```
public Object[] verDonante(String alias){

    System.out.println(alias+" alias que coge para la busqueda del socio ");

  String peticion = "SELECT * FROM Votante WHERE Votante.alias='"+alias+"' ";
  Object[] array = null;
  int i = 0;
  try{
    st =conn.createStatement();
    rs = st.executeQuery(peticion);

    if(rs==null){System.out.println("el rssss es NULLLLLL!!!!!");}

  //15 campos
  array = new Object[15];

  if (rs!=null && rs.next()){

    array[0] = rs.getString(1);//id
    array[1] = rs.getString(2);//nombre
    array[2] = rs.getString(4);//primer apellido.
    array[4] = rs.getString(4);//segundo apellido
    array[4] = rs.getString(5);//ciudad
    array[5] = rs.getString(6);//pais
    array[6] = rs.getString(7);//dni
    array[7] = rs.getString(8);//email
    array[8] = rs.getString(9);//telefono
    array[9] = rs.getString(10);//aportado
    array[10] = rs.getString(11);//fecha
    array[11] = rs.getString(12);//alias
    array[12] = rs.getString(14);//password
    array[14] = rs.getString(14);//activado
    array[14] = rs.getString(15);//voto

  }
  else
    System.err.println("Array con socio ampliado mal creado. No dimension.");
  }
```

```
    catch (SQLException esql){
      esql.printStackTrace();
    System.out.println("Error con la BBDD. Extendiendo los datos de socio seleccionad.");
    }
    catch (Exception edriver){
      edriver.printStackTrace();

      System.out.println("Error con la conexion, el driver, extendiendo los datos del socio
seleccionado.");
    }
    return array;

  }
```

4.3.3 Insertar datos

Se muestra un ejemplo de inserción de datos. Este método crea una nueva fila en la tabla de reintegros. Véase que el identificador auto-incremental se inicializa con NULL. La consulta se ejecuta con “executeUpdate” y posteriormente se implementa un control de posibles errores, si todo sale bien el método devolverá “TRUE” por el contrario “FALSE”.

```
    public void nuevoReintegro(String nombre,String apellido1,String apellido2,String
ciudad,String pais,String dni,String email,String telefono,String aportado,String fecha1,
String fecha2, String descripcion){

    String consulta = "INSERT INTO Reintegro VALUES
(NULL,'"+nombre+"','"+apellido1+"','"+apellido2+"','"+ciudad+"','"+pais+"','"+dni+"','"+e
mail+"','"+telefono+"','"+aportado+"','"+fecha1+"','"+fecha2+"','"+descripcion+"')";

 try{
      st = conn.createStatement();
      int registrosAfectados = st.executeUpdate(consulta);

      System.out.println(registrosAfectados);

      }

    catch (SQLException esql){
      esql.printStackTrace();
      System.out.println("Error con la BBDD.");
    }
    catch (Exception edriver){
      edriver.printStackTrace();
      System.out.println("Error con la conexion, el driver.");
    }}
```

4.3.4 Borrar datos

En el ejemplo se borra la cuenta de entrada en el sistema del usuario que se pasa como parámetro. La petición se inserta en la variable “st”, se ejecuta con “executeUpdate” y si todo sale bien se devuelve “TRUE”. Si salta alguna excepción se devuelve “FALSE”.

```
public void eliminarSesionDonante(String alias){

   String consulta = "DELETE FROM Donante WHERE Donante.alias='"+alias+"' ";


   try{
      st = conn.createStatement();
      int registrosAfectados = st.executeUpdate(consulta);

      System.out.println(registrosAfectados);




   }
   catch (SQLException esql){
      esql.printStackTrace();
      System.out.println("Error con la BBDD.");
   }
   catch (Exception edriver){
      edriver.printStackTrace();
      System.out.println("Error con la conexion, el driver.");
   }

}
```

4.3.5 Actualizar datos

En el ejemplo se muestra la forma de actualizar la cantidad aportada por un usuario. Como en los ejemplos anteriores se debe ejecutar la consulta con “executeUpdate” y si no salta ninguna excepción se devuelve “TRUE”.

```
    public void updateCantidad(String dni, String cantidad){

    String consulta = "UPDATE Votante SET aportado='"+cantidad+"' WHERE
Votante.dni='"+dni+"'";


    try{
      st = conn.createStatement();
      int registrosAfectados = st.executeUpdate(consulta);

      System.out.println(registrosAfectados);

    }
    catch (SQLException esql){
      esql.printStackTrace();
      System.out.println("Error con la BBDD.");
    }
    catch (Exception edriver){
      edriver.printStackTrace();
      System.out.println("Error con la conexion, el driver.");
    }
    }
```

4.3.6 Otras operaciones

Hay otro tipo de operaciones con la base de datos. En el primer ejemplo se busca si existe un usuario en el sistema, en vez de recoger los datos, si éste existe, se recoge un booleano con valor “TRUE”. Si por el contrario éste no existe o salta una de las excepciones controladas se devuelve “FALSE”.

```
public boolean validarDonante(String alias, String password){
    boolean res = false;
 String consulta = "SELECT * FROM Donante WHERE alias='"+alias+"' AND
password='"+password+"'";

   try{
      st = conn.createStatement();
      rs = st.executeQuery(consulta);
       if(rs!=null && rs.next()){
         res = true;
    }}
    catch (SQLException esql){
      esql.printStackTrace();
      System.out.println("Error con la BBDD.");
    }
    catch (Exception edriver){
      edriver.printStackTrace();
      System.out.println("Error con la conexion, el driver.");
    }
    return res;
  }
```

Finalmente se utiliza la función COUNT, para contar el dinero total donado en la organización sumando los valores numéricos de una columna. El valor resultante se inserta en la variable “rs” que más tarde se copia en el objeto que devuelve el método. Está implementado también un control de errores.

```
 String peticion = "SELECT COUNT(*) FROM Votante ";
    Object[] array = null;
    int i = 0;
    try{
      st =conn.createStatement();
      rs = st.executeQuery(peticion);

      if(rs==null){System.out.println("el rssss es NULLLLLL!!!!!");}

    //solo tendremos 1 campo
```

```
    array = new Object[1];

    if (rs!=null && rs.next()){

      array[0] = rs.getString(1);
      System.out.println("array0::::::::::"+array[0]);




    }
    else
      System.err.println("Array con socio ampliado mal creado. No dimension.");
    }
    catch (SQLException esql){
      esql.printStackTrace();


      System.out.println("Error con la BBDD. Extendiendo los datos de socio
seleccionad.");
    }
    catch (Exception edriver){
      edriver.printStackTrace();

      System.out.println("Error con la conexion, el driver, extendiendo los datos del socio
seleccionado.");
    }
    return array;
}
```

4.4 Servlets y JSP's

4.4.1 Cambios en web.xml

Esta es la parte más importante del diseño, dónde se realiza la aplicación en si. El usuario se comunica con el sistema gracias a los JSP, éstos tienen la misma función que el código HTML, pero se le puede insertar código Java. Por otra parte los Servlets son los que mantienen las sesiones, recogen los datos, redireccionan a donde haga falta…

Para que los Servlets sean visibles por Tomcat, hay que configurar el archivo "web.xml" que está en la carpeta "/nuestraAplicacion/WEB-INF/ ".

```
<servlet>
<servlet-name>iniciar</servlet-name>
<description>
Servlet que se encarga de iniciar la portada </description>
<servlet-class>servlets.Portada</servlet-class>
</servlet>
```

En la figura anterior se observa un fragmento del archivo “web.xml” donde se inicializa el Servlet inicial de la aplicación. En “servlet-name” se indica el nombre por el que se le quiere llamar desde el navegador. Y en “servlet-class” la ubicación del Servlet en sí.

```
<servlet-mapping>
        <servlet-name>iniciar</servlet-name>
        <url-pattern>/iniciar</url-pattern>
</servlet-mapping>
```

En “servlet-mapping” se indica el nombre de inicio del Servlet. Esta sección se ha de repetir para cada Servlet que se quiera implementar.

4.4.2 Inicializar un Servlet

A continuación se muestra, de forma detallada, un ejemplo de Servlet, en este caso el que lanza la portada. Se realizan los empaquetados y los “import” que se necesiten, más tarde se implementa el método “GET” de HTML, se establece el sistema de codificación y por último se envía el control de la aplicación al archivo JSP inicial.

```
//se empaqueta nuestro servlet en la carpeta de servlets
//se importan las demas carpetas de la aplicacion
//se importan las librerias necesarias

package servlets;

import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

import beans.*;
import conexion.*;
```

```
// servlet que lanza la portada principal de la aplicación
public class Portada extends HttpServlet {

  //metodo del servlet que responde a un peticion GET
  public void doGet(HttpServletRequest request, HttpServletResponse response)
  throws IOException, ServletException
  {

  // establece ContentType y sistema de codificaciÃ³n
    response.setContentType("text/html; charset=ISO-8859-1");

    // obtiene un PrintWriter para escribir la respuesta
    PrintWriter out = response.getWriter();

    //manda a Portada.jsp y se visualiza

      RequestDispatcher rd = request.getRequestDispatcher("Portada.jsp");
      rd.forward(request,response);

 }
}
```

4.4.3 Portada

Tras arrancar Tomcat si se ha realizado lo descrito en los dos puntos anteriores, al escribir en el navegador: http://localhost/farts/iniciar se iniciará la aplicación como se observa en la figura 4.2.

Figura 4.2 Portada de la aplicación

En la portada sólo se tiene la opción de entrar en la aplicación principal. En ésta hay tres opciones de visualización dependiendo de las sesiones que haya: si hay un donante autenticado, si hay un administrador autenticado o si no hay ninguna sesión en el sistema. Todo esto lo hace un JSP llamado “General.jsp”. El procedimiento de programación es el siguiente:

```
<jsp:useBean id="admin" class="beans.Administradores" scope="session"/>

<jsp:useBean id="donante" class="beans.Donantes" scope="session"/>
```

Figura 4.3 Extracción de usuarios de la sesión

En la figura 4.3 se puede apreciar como se extrae una sesión del sistema: “useBean id” es el identificador con el que se ha insertado la sesión en el sistema (en otro capítulo se verá

el proceso de los inicios de sesión). “Class” se refiere al lugar donde se encuentra el “bean” que inicia la sesión, en este caso se observa como llama a los dos: “Donantes” y “Administradores”, ya que se necesita saber si alguna de las dos clases ha iniciado sesión. Por último en “scope” se señala de donde se quieren extraer los datos, en nuestro caso se extraerán de la “session” del sistema.

Para otras operaciones como puede ser insertar o coger datos de un formulario, se utiliza la variable “request” que tiene un tiempo de vida corto: sólo hasta que se llame al siguiente Servlet o JSP.

Para mostrar un código u otro JSP, se insertan condicionales Java que buscan usuarios en “session”, para ello se utiliza la sentencia “<% %>”, todo lo que vaya dentro de los tantos por cientos se considerará código java:

```
<% if((admin.getAlias()==null) && (donante.getAlias()==null)){ %>
<% } else if(donante.getAlias()!=null) { %>
<% } else if(admin.getAlias()!=null) { %>
```

Figura 4.4 Ejemplos de código Java en JSP

Utilizando el método “get” implementado en los “beans”, se extrae el alias de “session”. Si es nulo significa que no hay nada guardado en “session”, es decir, no hay ningún donante ni administrador con una sesión activa. En caso contrario en la sesión actual hay un donante o un administrador en el objeto “session”.

Opción A: no hay nada en la sesión actual

Esto significa o que nadie se ha autenticado aún , o lo que es peor, es la primera vez que se entra en la aplicación y ni siquiera existe la opción de autenticarse. El código utilizado es el de la figura 4.5 y la vista en el navegador sería la de la figura 4.6.

En el código mostrado a continuación se observa como los enlaces que se quieren mostrar como botones se encuentran dentro de la etiqueta “menu”. Se observa también como existen tres etiquetas llamadas “colOne”, “colTwo” y “colThree”. El código que se

encuentre dentro de ellas se presentará más tarde en la aplicación en la parte izquierda, central o derecha respectivamente.

Por otra parte es importante saber que por ejemplo para mostrar un error no son necesarias tres columnas, por lo tanto las etiquetas de columnas no se indicarían y se insertaría el código en la etiqueta llamada “content”.

```
<% if((admin.getAlias()==null) && (donante.getAlias()==null)){ %>

<div id="header">
<h1><strong>farts</strong></h1>
<h2>fondo de alto rendimiento para la transformacion social</h2>
</div>

<div id="content">

<div id="menu">


                <a href="iniciar">Inicio</a>
                <a href="verNoticias">Noticias</a>
                <a href="Contacto.jsp">Contacta</a>
                <a href="Recuperar.jsp">recupera contraseña</a>

</div>

<div id="colOne">

<h2>Entrada para Socios</h2>



        <form name="input" action="ingresarDonante" method="get">
        <p>Alias:</p>

        <input type="text" name="user">
<p></p>

        <p>Contraseña:</p>

        <input type="password" name="pass">
        <p></p>
        <input type="submit" style="background-color: #ADFF2F" value="Ingresar">

        </form>
```

```
		</div>

		<div id="colTwo" align="center">
			<h2>Si eres nuevo en el sistema...</h2>

<form name="input" action="FormNuevoSocio.jsp" method="get">
		<p>Entra para darte de alta:</p>
		<input type="submit" style="background-color: #ADFF2F" value="Nueva
Donacion">

		</form>

</div>

<div id="colThree">

<h2>Administrador</h2>

		<form name="input" action="ingresarAdmin" method="get">
		<p>Alias:</p>

		<input type="text" name="user">
<p></p>

		<p>Contraseña:</p>

		<input type="password" name="pass">
		<p></p>
		<input type="submit" style="background-color: #ADFF2F" value="Ingresar">

		</form>
</div>
</div>

<div style="clear: both;"> </div>

<div id="footer">

		<p>Copyright &copy; 2010 FARTS . Diseñado por CRS . <a
href="http://www.freecsstemplates.org/">Free CSS Templates</a></p>
</div>

<%}%>
```

Figura 4.5 Primera opción de "General.jsp"

Figura 4.6 Inicio de la aplicación sin sesión iniciada

Si se llega a esta pantalla sin haberse autenticado antes, se tienen varias opciones:

- **Inicio:** Portada principal de la aplicación. (pág.31)
- **Noticias:** Zona de noticias. (pág.39)
- **Contacta:** Contacto con la administración del sistema. (pág.41)
- **Recupera Contraseña:** Formulario para recuperación de contraseña. (pág.44)
- **Entrada para Socios (Ingresar):** Inicio de sesión para socios. (pág.46)
- **Entrada de Administración (Ingresar):** Inicio de sesión para Administradores. (pág.48)
- **Nueva Donación:** Entrada a la zona de nuevo usuario. (pág.48)

Opción B: hay una sesión donante iniciada

El código es muy parecido al anterior, así que directamente se estudia la pantalla resultante en la figura 4.7.

Figura 4.7 Inicio aplicación con sesión de donante iniciada

Las posibles opciones son las siguientes:

- **Info personal:** Zona donde se presenta nuestro resumen de actividad en el sistema. (pág.52)
- **Noticias:** Zona de noticias. (pág.39)
- **Sesión (Cerrar Sesión):** Aparte de una bienvenida para el socio actual, se tiene la opción de cerrar nuestra sesión. (pág.52)
- **Reintegro (Ir):** Zona de confirmación del reintegro de nuestra cuenta en el sistema. (pág.53)
- **Vota :** Zona de votaciones. (pág.54)
- **Cantidad (Ir):** Este paso se debe de realizar para avisar a la administración de que se ha realizado un nuevo pago. (pág.58)
- **Donate:** Si se quiere realizar una nueva donación se pulsa aquí. (pág.49)

Opción C: hay una sesión administrador iniciada

Se estudia la pantalla resultante en la figura 4.8.

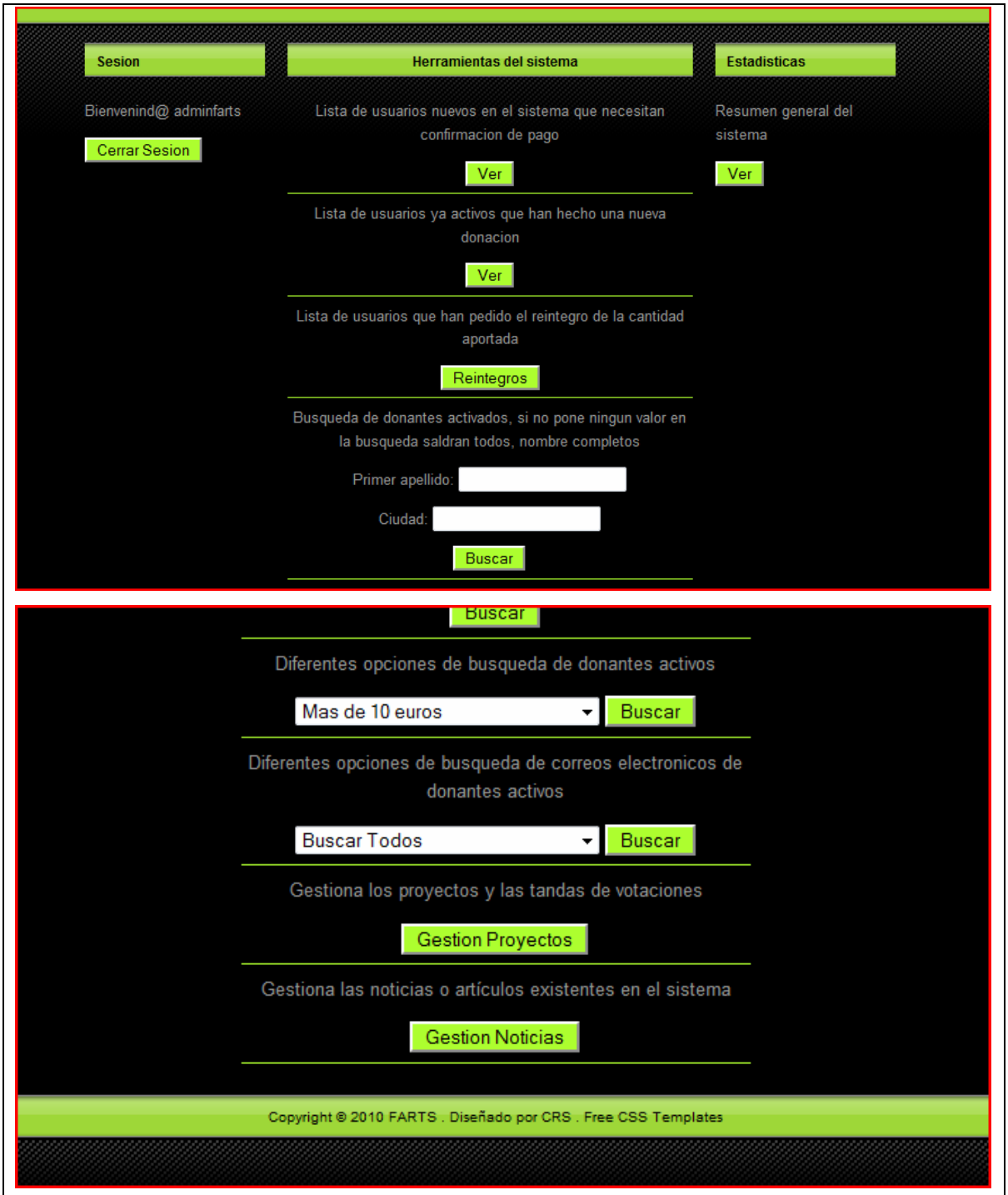


Figura 4.8 Inicio aplicación con sesión de administrador iniciada

Las posibles opciones son las siguientes:

- **Sesión (Cerrar Sesion):** Tras una bienvenida del sistema, se nos da la opción de cerrar nuestra sesión activa. (pág.52)
- **Estadísticas (Ver):** Resumen de estadísticas generales del sistema (pág.58)
- **Ver (lista de usuarios nuevos en el sistema):** Se ve una lista de usuarios nuevos en el sistema, que necesitan la confirmación del primer pago. (pág.59)
- **Ver (lista de nuevos pagos en el sistema):** Se ve una lista de socios que quieren confirmar una nueva donación. (pág.62)
- **Reintegro:** Se ve una lista de usuarios que quieren el reintegro de su cuenta, ésta conlleva una devolución de la cantidad total donada. (pág.63)
- **Buscar :** Se visualiza una lista de los socios del sistema, se puede buscar por primer apellido o por ciudad. Si no se indica nada se visualizaran todos los socios. (pág.63)
- **Buscar (con opciones):** Se visualiza una lista de socios según la opción que se haya buscado (más de 10 euros donados, socios españoles o socios extranjeros). (pág.64)
- **Buscar mails (con opciones):** Se muestra una lista de correos electrónicos según la opción que se haya buscado (más de 10 euros donados, socios españoles, socios extranjeros o todos los correos electrónicos). Gracias a ésto se pueden copiar los correos electrónicos requeridos para su uso en el software de correo propio de la organización. (pág.66)
- **Gestión de proyectos:** Zona de gestión de los proyectos y sus votaciones. (pág.66)
- **Gestión de noticias**: Zona de gestión de noticias. (pág.69)

4.4.4 Noticias

La aplicación dispone de una sección de noticias, en los siguientes gráficos se ve como se recogen los datos de las diferentes noticias y como se presentan al usuario.

```
<% Object[][] array = (Object[][])request.getAttribute("noticias"); %>

<h4>articulos o noticias</h4>
<p> </p>

<% for(int i=0; i<array.length; i++){ %>

<a href='verNoticia?id=<%=array[i][0]%>'> <%=array[i][1]%> </a>
<p>  </p>

<%}%>
```

Figura 4.9 Fragmento de código de "Noticias.jsp"

Figura 4.10 Sección de noticias

En la figura 4.9 se puede observar un fragmento del código de "Noticias.jsp", primero se extrae el objeto noticias del request (que ha sido insertado anteriormente por el Servlet "verNoticias.java" como se puede observar en la figura 4.11).

```
GestionDatos gD = new GestionDatos();
Object[][] array= gD.verNoticias();
gD.cerrar();
request.setAttribute("noticias",array);
RequestDispatcher rd = request.getRequestDispatcher("Noticias.jsp");
rd.forward(request,response);
```

Figura 4.11 Fragmento código "verNoticias.java"

Tras ésto, utilizando iteraciones de Java, se presenta una lista con el nombre de cada noticia. El enlace conduce a "verNoticia.java" y en la url se pasa el "id" de la noticia que se quiere visualizar. El Servlet crea un objeto con la noticia a visualizar, lo inserta en el request y llama a "Noticia.jsp" que representa los datos del objeto "noticia" en el JSP obteniendo un resultado como el mostrado en la figura 4.12.

Figura 4.12 Ejemplo de noticia

4.4.5 Contacto

La aplicación dispone de una zona de contacto con la administración, para el envío del correo electrónico se utiliza la librería de Java llamada “JavaMail”. A continuación se observan los pasos necesarios:

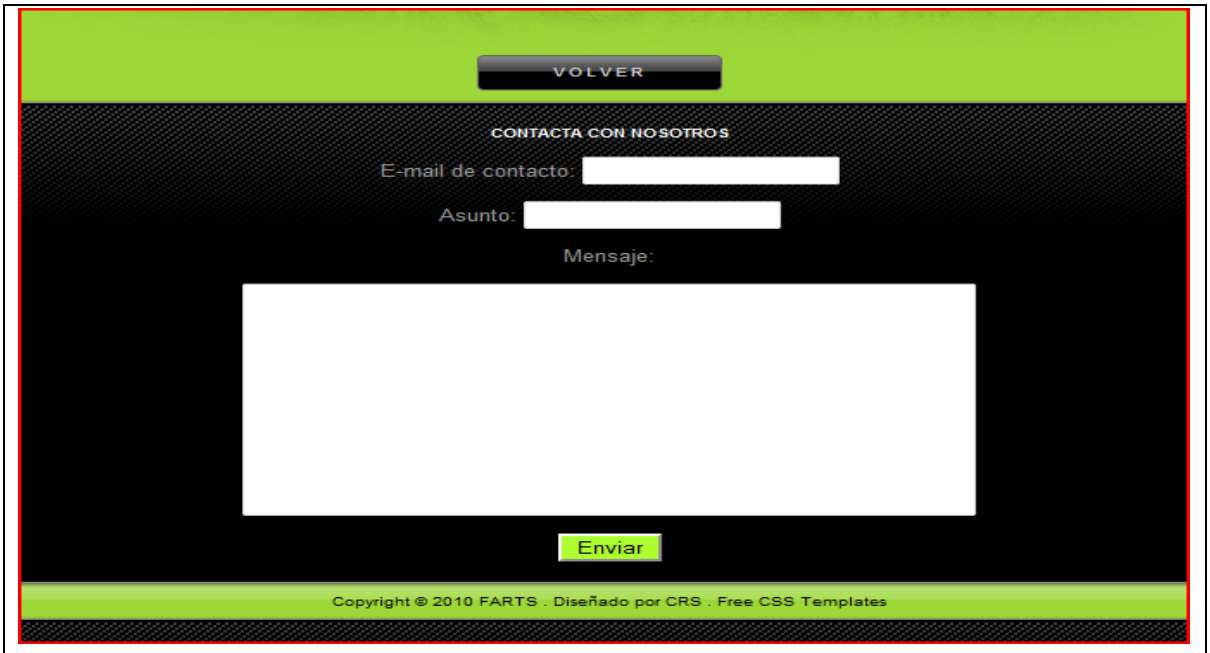


Figura 4.13 Formulario de contacto

En la figura anterior se muestra el resultado de visualización de “Contacto.jsp”, se muestra por pantalla un formulario con tres datos a cumplimentar: e-mail de contacto del usuario, asunto y cuerpo del mensaje.

```
// cogemos campos del correo

String mail = request.getParameter("mail");

String asunto = request.getParameter("asunto");

String cuerpo = request.getParameter("cuerpo");

String body = "#FARTS#  "+mail+" escribio: "+cuerpo;

//envio mail

SendMail envio = new SendMail();

envio.enviar("farts.prat@yahoo.es",asunto,body);
```

Figura 4.14 Llamada al método de envío de correos electrónicos

En la figura 4.14 se puede observar como se extraen los parámetros del request y más tarde se llama al método “enviar” de la clase “SendMail.java”. A este método se le pasan tres parámetros: la dirección de destino (en este caso el correo de la administración, el correo de orígen será siempre el del servidor de correo utilizado), el asunto y el cuerpo del mensaje.

```
public void enviar(String mail,String tema,String cuerpo){

   String username= "farts.prat@yahoo.es";
   String password = "12445678abc";
   Properties props = new Properties();
   props.put("mail.smtp.host", "smtp.mail.yahoo.es");
   props.put("mail.debug", "true");

   Session session = null;
   if (username != null && password != null)
   {
    props.setProperty("mail.smtp.port","25");
    props.put("mail.smtp.auth", "true");

    session = Session.getInstance(props,
    new MyPasswordAuthenticator(username, password));
   }
   else
   {
     session = Session.getDefaultInstance(props, null);
   }

   MimeMessage message = new MimeMessage(session);
   try {
   message.setFrom(new InternetAddress("farts.prat@yahoo.es"));
   InternetAddress[] address = InternetAddress.parse(mail, false);
   message.setRecipients(Message.RecipientType.TO, address);
   message.setSubject(tema);
   message.setText(cuerpo);

   Transport.send(message);

   } catch (AddressException e) {
      e.printStackTrace();
   } catch (MessagingException e) {
      e.printStackTrace();
   } }
```

Figura 4.15 Método “enviar” de “SendMail.java”

En la figura 4.15 se puede observar el funcionamiento de la clase “SendMail.java”, en el caso que se estudia se utiliza el servidor “smtp” de Yahoo, pero se podría extrapolar a cualquier servidor de correo, incluso el proporcionado por la empresa de hosting.

“Properties” se encarga de realizar la conexión con el servidor, “MimeMessage” de rellenar los campos del correo y “Transport” de realizar el envío en sí. Como también se puede observar se utiliza otra clase llamada “MyPasswordAuthenticaction” que realiza la autenticación del nombre y contraseña de usuario de la cuenta de correo que utilizemos. A continuación se observa el código de la clase Java:

```
public class MyPasswordAuthenticator extends Authenticator
{
String user;
String pw;


public MyPasswordAuthenticator (String username, String password)
{
super();
this.user = username;
this.pw = password;
System.out.println(user);
}
public PasswordAuthentication getPasswordAuthentication()
{
return new PasswordAuthentication(user, pw);
}
}
```

Figura 4.16 “MyPasswordAuthentication.java”

Aquí se muestra un ejemplo del mensaje que le llegaría a la organización.

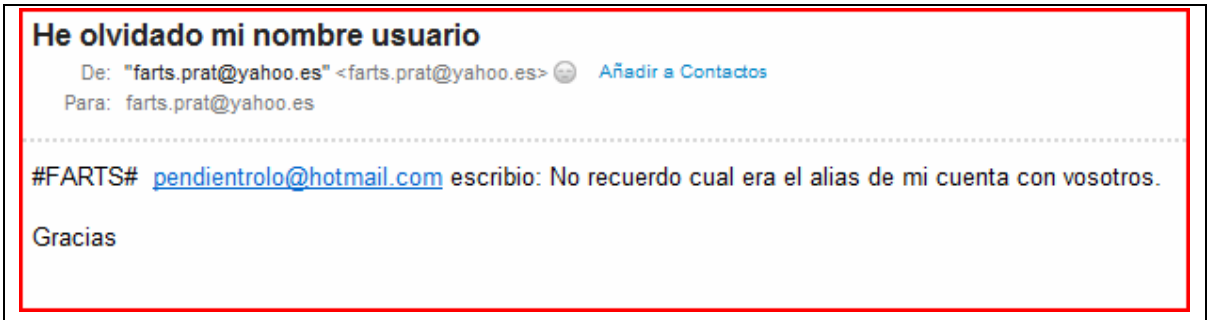


Figura 4.17 Ejemplo correo recibido de contacto por la administración

4.4.6 Recuperar contraseña

El funcionamiento de la recuperación de contraseña es muy parecido al del apartado anterior.

Figura 4.18 Formulario de recuperación de contraseña

En la figura anterior se ve el aspecto de la zona de recuperación de contraseña. Gracias a ésta, un usuario puede recuperar la contraseña de su cuenta si se le ha olvidado. Debe introducir el "alias" que le identifica en el sistema y el dni, si todo es correcto el usuario recibirá en su correo personal la contraseña como se puede ver en la siguiente figura:

Figura 4.19 Ejemplo de correo recibido por un usuario al recuperar contraseña

Si por el contrario el "alias" y/o la contraseña no son válidos, se recibirá un mensaje de error como el siguiente:

Figura 4.20 Error al recuperar contraseña

Como se puede ver, cuando se manda un correo en "Contacto" el destinatario es la cuenta de correo de la propia organización, pero en cuanto a la recuperación de la contraseña, el destinatario será el propio cliente, así que habrá que hacer unos cambios tan simples como variar los parámetros que se pasan al método "enviar" de la clase "SendMail" tal y como puede verse en la siguiente captura de código:

```
SendMail envio = new SendMail();
    String email = (String)array[0];
    String titulo = "FARTS";
    String parrafo = "Su contraseña en el sistema es: "+(String)array[1];
    envio.enviar(email,titulo,parrafo);
```

Figura 4.21 Llamada al método de envío de correos para la recuperación de contraseña

En vez de pasar como destinatario la cuenta de correo de la organización, se pasa el del propio cliente, que se ha recuperado de la base de datos junto a la contraseña. “array[0]” y “array[1]” respectivamente.

4.4.7 Entrada para socios (Ingresar)

En este apartado se explicará como se realiza el inicio de sesión por parte de un usuario del sistema. En el siguiente fragmento de código se explican las acciones que lleva a cabo el Servlet “IngresarDonante.java”.

```
    // se recogeel alias y el password introducido en el formulario,
    // se convierten a minúsculas porque en la base de datos todo figura asi,
    // el alias y el password de la base de datos los introduce el usuario en el
    // formulario de activación de la organización.

    String user = request.getParameter("user");
    String alias = user.toLowerCase();

    String pass = request.getParameter("pass");
   String password = pass.toLowerCase();

    //Crear el objeto que va a realizar la llamada a la BBDD

    GestionDatos gestion = new GestionDatos();

    //se mira si existe ese alias y ese password en la BBDD, en caso negativo habra un
error.

    if(!gestion.validarDonante(alias,password)){

      gestion.cerrar();

      RequestDispatcher rd = request.getRequestDispatcher("ErrorAlias.jsp");
      rd.forward(request,response);
    }



    else
    {
    GestionDatos gD = new GestionDatos();

    // con este metodo se recoge el permiso del usuario, si los administradores
```

```
    // ya lo han confimado por consecuencia de haber llegado la confirmacion
    // de pago el permiso será "SI", en caso contrario "NO"

    Object[] array = gD.verPermiso(alias);
    String permisos = (String)array[0];
    int permiso = permisos.compareTo("si");
    gD.cerrar();

    // si no se tienen permisos se avisara por pantalla

        if(permiso!=0){
       RequestDispatcher rd = request.getRequestDispatcher("ErrorPermiso.jsp");
       rd.forward(request,response);

        }

        else{
      /*cerramos la conexion con la BBDD*/


    // si se tienen permisos se creara la sesion con los datos y seremos conducidos
    // a la zona de socios

        Donantes donante = new Donantes(alias,password);

    //aqui se crea la sesion cogiendo los parametros que le se han introducido en el
formulario html
        HttpSession session = request.getSession(true);

        //se introduce el administrador en la sesion
        session.setAttribute("donante",donante);

        // se redirecciona otra vez a la aplicacion principal, pero esta vez existira una sesion
        RequestDispatcher hj = request.getRequestDispatcher("general");
        hj.forward(request,response);
```

Figura 4.22 Fragmento de código de "IngresarDonante.java"

4.4.8 Entrada para administración (Ingresar)

El inicio de sesión es básicamente idéntico al del apartado anterior, aunque extrapolándolo al bean de Java “Administradores”.

Además, en este caso no es necesaria una validación, sólo que el alias y el password coincidan con alguna de las filas de la tabla “Administrador” de la base de datos MySQL.

4.4.9 Nueva donación

Esta parte sirve para realizar la primera donación y registrarse en el sistema. Se mostrará un formulario como el siguiente:

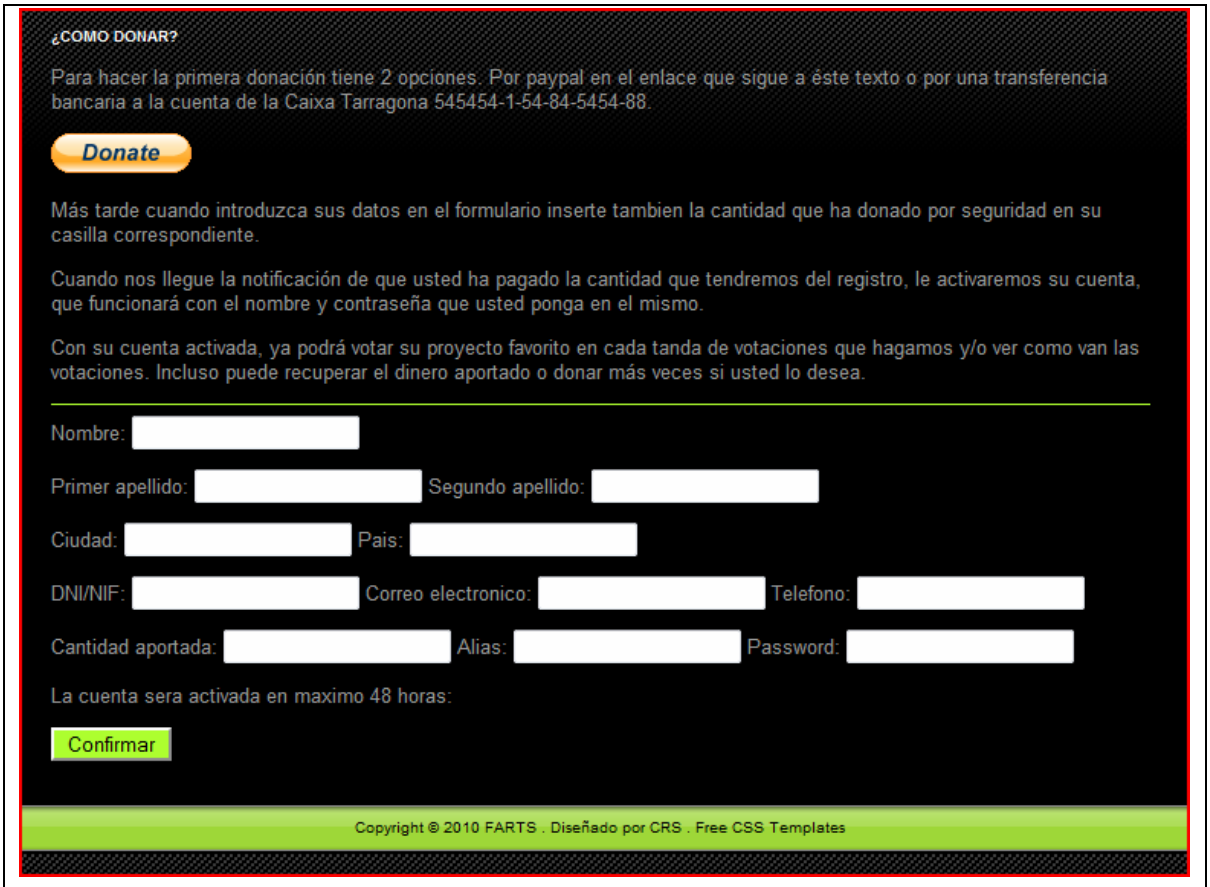


Figura 4.23 Formulario ingreso en la organización

Antes de introducir los datos de registro, se explica como donar una cantidad mínima de 10 euros en la organización. Existen dos opciones: un ingreso bancario en una cuenta como las de toda la vida o utilizar Paypal.

Paypal no cobra nada a los usuarios que lleven a cabo donaciones, los intereses serían cobrados a la cuenta "business" de la organización a la hora de poder cobrar las donaciones.

El proceso de hacer una donación por Paypal es muy fácil, se pulsa el botón de donación y redirecciona directamente a un sitio web seguro de Paypal donde se presentará un formulario para introducir los datos de la cuenta de Paypal y la cantidad que se quiera donar.

El botón que se utiliza en la aplicación es un botón de ejemplo tipo de donación, cuando la organización obtuviera la cuenta "business" solo habría que cambiar el código del botón actual por el suministrado por Paypal. A continuación se ve la página a la que lleva Paypal al pulsar "Donate".

Figura 4.24 Formulario donación de Paypal

Fácilmente se observa como introducir la cantidad que se quiera donar.

Figura 4.25 Inicio sesión Paypal

Si ya se dispone de una cuenta se introducen los datos, si no se debe rellenar el formulario siguiente:

Figura 4.26 Formulario de creación de una cuenta en Paypal

Despues de haber donado la cantidad por Paypal o mediante una transferencia habría que rellenar el formulario de registro de la figura 4.23.

En el parámetro “Cantidad aportada” se deberá indicar la cantidad aportada esta primera vez (por seguridad). El administrador al confirmar la cuenta cotejará la información que le llega del registro con la del aviso por correo electrónico de Paypal o del banco.

Los campos “alias” y “password”, son respectivamente el nombre para entrar en el sistema y la contraseña correspondiente.
Es importante saber que no se podrá entrar en el sistema hasta que el administrador confirme el pago.

4.4.8 Información personal

Cuando el usuario acceda a este apartado, se le presentará un resumen de su actividad en el sistema: los datos personales, fecha de ingreso en la organización, dinero total aportado, etc…

Figura 4.27 Sección de información personal

4.4.9 Sesión (Cerrar Sesión)

Se dispone de un mensaje de bienvenida y de un botón para el cierre de la sesión en el sistema. Para mostrar en el mensaje de bienvenida el “alias” de la persona que haya iniciado la sesión, se utiliza el método del “bean” correspondiente de la “session” de Servlets. Ésto se consigue utilizando el siguiente fragmento de código del archivo “General.jsp”.

```
<p>Bienvenind@ <%= donante.getAlias() %> </p>
```

En el caso que fuese en el apartado de administración, se extraería de “administrador” no de “donante”.

Si se pulsa el botón “Cerrar Sesión”, se redirecciona al Servlet llamado “CerrarSesion.java”, el código utilizado para cerrar la sesión con el sistema es el siguiente:

```
HttpSession session = request.getSession(true);

session.invalidate();

RequestDispatcher rd = request.getRequestDispatcher("general");
rd.forward(request,response);
```

Figura 4.28 Fragmento de código de “CerrarSesion.java”

Si existe sesión, (que siempre será afirmativo, ya que de otra forma no tendríamos acceso al botón de cierre) se coge y se invalida, despues se redirecciona otra vez al inicio de la aplicación.

4.4.10 Reintegro (Ir)

Cualquier socio de la organización tiene derecho a que se le devuelva el dinero que ha donado, este apartado realiza esta gestión.

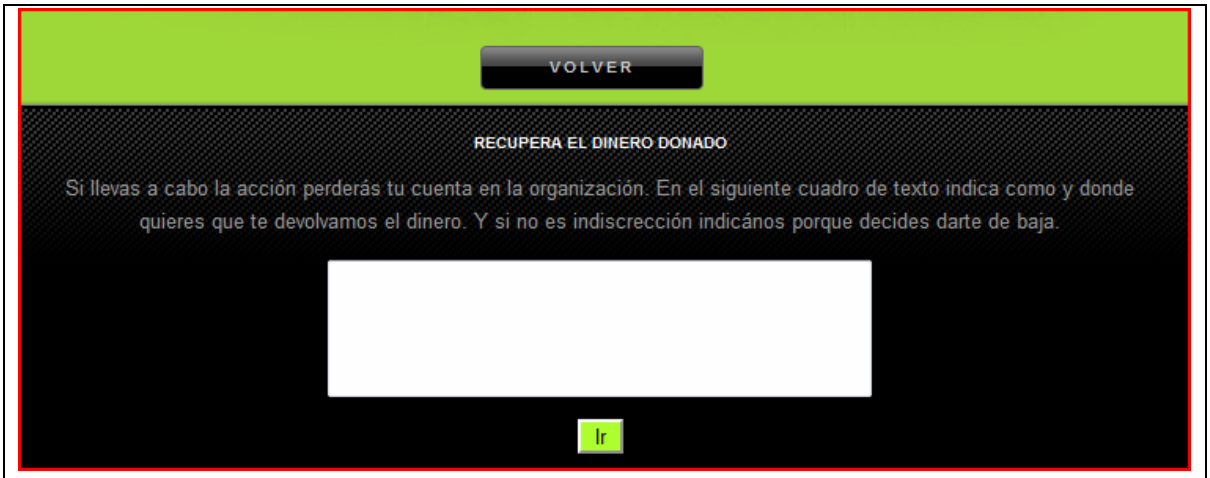


Figura 4.29 Formulario de reintegro de la aplicación

En la figura anterior se observa como se ofrece una opción donde el usuario puede explicar la forma con la que quiere que se le devuelva el dinero (un número de cuenta por ejemplo).

Tras ésto se mostrará por pantalla un mensaje de confirmación de reintegro, ya que si se realiza, aunque se recupere el dinero, la cuenta de entrada al sistema será borrada de la base de datos, y ya no se podrá entrar más a la aplicación a no ser que este usuario se haga otra cuenta en el futuro.

En la figura siguiente se observa como se ofrece una opción para confirmar el reintegro:

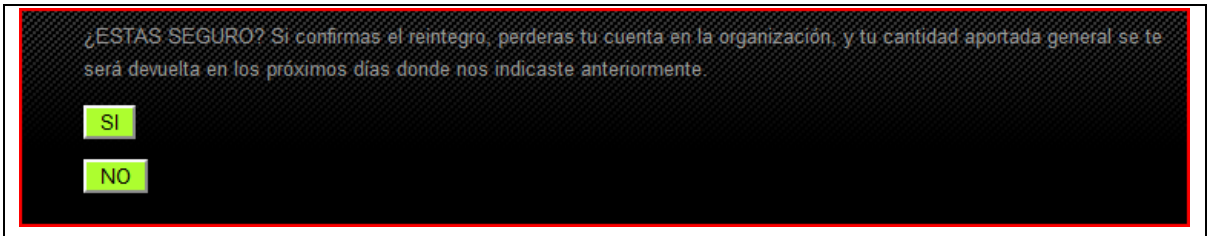


Figura 4.30 Confirmación de reintegro

4.4.11 Vota

Se dispone de una zona de votaciones, el socio podrá votar al proyecto que más le guste en cada tanda de votaciones. El Servlet que lleva a cabo la gestión para el lanzamiento de la pantalla de votaciones se llama “VotarProyecto.java”.

```
GestionDatos gD = new GestionDatos();

Object[][] array= gD.verProyectos();

gD.cerrar();

request.setAttribute("proyectos",array);

RequestDispatcher rd = request.getRequestDispatcher("Proyectos.jsp");
rd.forward(request,response);
```

Figura 4.31 Fragmento de código de “VotarProyecto.java”

En el Servlet anterior se recogen todo los datos de la tabla “Votaciones” de la base de datos MySQL. Se insertan en el “request” y se llama a “Proyectos.jsp” para que los muestre por pantalla. En la siguientes figuras se muestra un fragmento de código de “Proyectos.jsp” y como se visualiza por pantalla.

```
<% Object[][] array = (Object[][])request.getAttribute("proyectos"); %>


        <% for(int i=0; i<array.length; i++){ %>

        <h4>proyecto <%=i%>: <%=array[i][0]%></h4>
        <p> </p>

<img src="<%=array[i][4]%>" alt="imagen sobre el proyecto" width="500" height="400"
>

        <p> </p>
        <%=array[i][1]%>
        <hr size="1" color="#ADFF2F">
        <%}%>
```

Figura 4.32 Fragmento código “Proyectos.jsp”

Primero se recoge la información de cada proyecto, y mediante una iteración se muestran los datos de cada uno, también se mostrará una imagen, para saber donde está, en una columna de la base de datos se recoge la url del archivo en el sistema.

Figura 4.33 Visualización de las votaciones de los proyectos

A continuación se muestra un fragmento de código de "Proyectos.jsp":

```
<form name="input" action="votar" method="get">
  <input  type="hidden" value="<%=alias%>" name="alias">

.
.
.

<% for(int i=0; i<array.length; i++){ %>

        proyecto <%=i%>: <%=array[i][0]%> lleva <%=array[i][2]%> votos      <input
type=radio name="voto" value="<%=i%>">

<p> </p>

<%}%>

        <input type="submit" style="background-color: #ADFF2F" value="Vota!">
<p> </p>
        </form>
```

Figura 4.34 Fragmento código de "Proyectos.jsp"

Se implementa un formulario con “RadioButtons” para poder elegir al proyecto que se vota, pero antes de mostrar los proyectos se hace la llamada al “form” y se pasa también un atributo “hidden” con el alias del usuario que va a realizar la votación (Los atributos hidden sirven para pasar valores que no debe introducir el usuario, valores fijos).

Despues se realiza una iteración para mostrar cada “RadioButton” con sus datos correspondientes, si se marca uno de los proyectos, se pasará como valor el índice que lleva el proyecto en el objeto “array[][]” de votaciones.

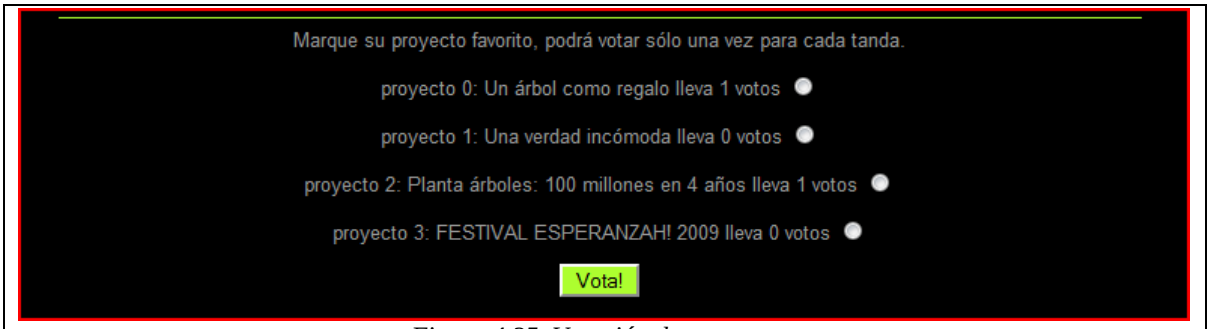


Figura 4.35 Votación de proyecto

El donante marca el proyecto que más le guste. Al votar se le lleva a un Servlet que mira los permisos del usuario, aprovechando que se ha pasado el “alias· del donante mediante el atributo “hidden” del formulario.

Si el usuario aún no había votado en la tanda de votaciones actual, se sumará un voto más al proyecto correspondiente y se cambiarán los permisos del usuario sobre votaciones.
Si el usuario ya había votado en la actual tanda, se le mostrará un mensaje de error en el cual se le comenta que ya ha votado y se le insta a esperar a la siguiente tanda de votaciones.

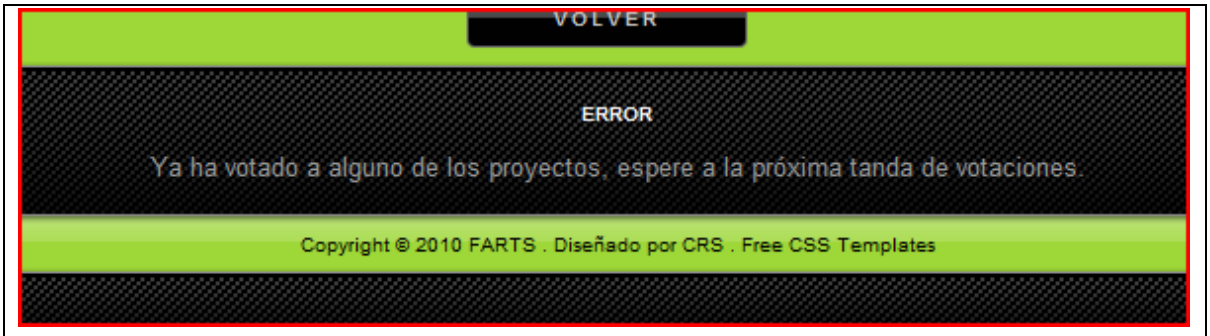


Figura 4.36 Error en votaciones

4.4.12 Cantidad (Ir)

Este apartado está pensado para los socios que quieran hacer otra donación. Despues de hacer la donación con alguna de las opciones que da el sistema, se deberá introducir la cantidad que se ha donado también en la aplicación web, de ésta forma llegaran dos avisos al administrador: uno de Paypal o del banco y otro de la aplicación.

4.4.13 Estadísticas (Ver)

Esta zona sirve un pequeño resumen sobre la actividad de los socios, se muestra el número de socios, la cantidad total donada por los socios y los votos que lleva cada proyecto en las votaciones. Para conseguir esto, se recoge la información de la base de datos y se inserta en el “request” antes de la llamada a “Estadísticas,jsp”. Desde el archivo JSP ya es fácil recoger la información y mostrarla.

Figura 4.37 Estadísticas del sistema

4.4.14 Ver (lista de usuarios nuevos en el sistema)

Esta zona de gestión muestra los datos de los usuarios que han hecho una primera donación en el sistema y aún no han recibido la confirmación para poder entrar.

El administrador podrá confirmar los pagos, editar la cantidad donada en el sistema (Por si no coincide con la aportada realmente) o incluso descartar totalmente la donación (Si pasado un tiempo no llega la confirmación del banco o de Paypal).

```
GestionDatos gestion = new GestionDatos();

Object[][] array = gestion.listar(2);

gestion.cerrar();

request.setAttribute("busqueda",array);
```

Figura 4.38 Fragmento código "ListarSinConfirmación.java"

En la figura anterior se observa un fragmento de código de "ListarSinConfirmacion.java". Se llama a un método de "GestionDatos.java" llamado "listar", el parámetro que se le pasa indica al método que búsqueda debe hacer, en este caso los socios no confirmados. Se puede observar el funcionamiento del método dependiendo del entero que se incrusta en la llamada (Cada número realiza una consulta distinta) en la figura siguiente:

```
public Object [][] listar(int tipo){



   int filasResultSet = 0;
   int i = 0;
   String peticion = null;
   Object[][] array = null;


   if(tipo==1){
   peticion= "SELECT id,nombre,apellido1,apellido2,ciudad,dni,aportado FROM Pago";
   }
   else if(tipo==2){
   peticion= "SELECT id,nombre,apellido1,apellido2,ciudad,dni,aportado FROM Votante
```

```
WHERE Votante.activado='no'";
    }
    else if(tipo==4){
    peticion= "SELECT id,nombre,apellido1,apellido2,ciudad,dni,aportado FROM Votante
WHERE Votante.activado='si' AND Votante.pais='españa'";
    }
    else{
    peticion= "SELECT id,nombre,apellido1,apellido2,ciudad,dni,aportado FROM
Reintegro ";
    }
```

Figura 4.39 Fragmento método "listar" de "GestionDatos.java"

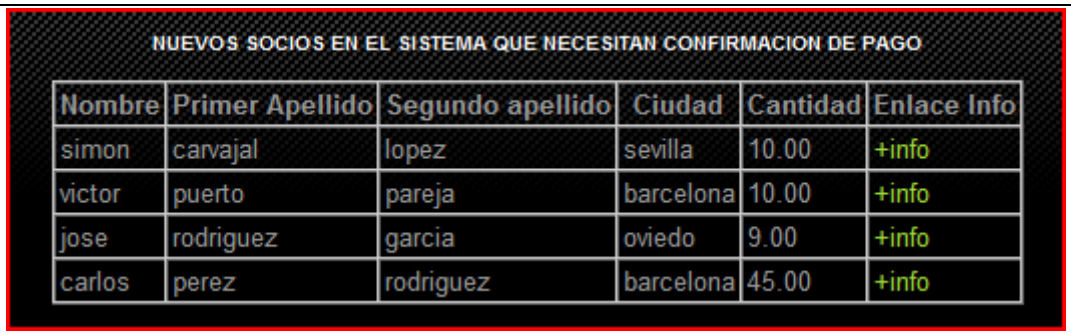
Tras ésto la aplicación presenta en una tabla todos los socios nuevos del sistema. Para ello se utiliza una tabla y un iterador Java para mostrar cada una de las filas del objecto Java.

```
<% Object[][] array = (Object[][])request.getAttribute("busqueda"); %>

<table border=1 cellspacing=0 cellpadding=2>

        <tr><th>Nombre</th><th>Primer Apellido</th><th>Segundo
apellido</th><th>Ciudad</th></th><th>Cantidad</th><th>Enlace Info</th></tr>


<% for(int i=0; i<array.length; i++){ %>

<tr>
                <td> <%=array[i][1] %> </td>
                <td> <%=array[i][2] %> </td>
                <td> <%=array[i][4] %> </td>
                <td> <%=array[i][4] %> </td>
                <td> <%=array[i][6] %> </td>
                <td><a
href='verDonanteSin?id=<%=array[i][0]%>&dni=<%=array[i][5]%>'> +info </a></td>

</tr>
<%}%>
</table>
```

Figura 4.40 Código para listar usuarios en una tabla

En la figura 4.39 se puede observar como se tiene un enlace para ver más detalladamente la información de cada usuario y poder realizar las operaciones pertinentes.

El enlace pulsado lleva insertado en la url los datos del usuario para poder buscar en la base de datos el resto de la información, ésto se ve en la figura 4.38.



NUEVOS SOCIOS EN EL SISTEMA QUE NECESITAN CONFIRMACION DE PAGO

Nombre	Primer Apellido	Segundo apellido	Ciudad	Cantidad	Enlace Info
simon	carvajal	lopez	sevilla	10.00	+info
victor	puerto	pareja	barcelona	10.00	+info
jose	rodriguez	garcia	oviedo	9.00	+info
carlos	perez	rodriguez	barcelona	45.00	+info

Figura 4.41 Tabla listado de socios

La aplicación mostrará la información y las diferentes opciones que se tienen de edición. En la figura 4.42 y 4.43 se observa el código y la visualización del archivo "UsuarioSin.jsp".

```
<% Object[] array = (Object[])request.getAttribute("array"); %>

Nombre: <%=array[1]%> <%=array[2]%> <%=array[4]%>
<hr size="1" color="#ADFF2F">
Ciudad: <%=array[4]%>, Pais: <%=array[5]%>, DNI/NIF: <%=array[6]%>
<hr size="1" color="#ADFF2F">

email: <%=array[7]%>, Telefono: <%=array[8]%>, Aportado: <%=array[9]%>
<hr size="1" color="#ADFF2F">

Fecha ingreso: <%=array[10]%>
<hr size="1" color="#ADFF2F">

<a href='eliminarDon?id=<%=array[0]%>&alias=<%=array[11]%>'> Descartar pago </a>
<p> </p>
<a href='confirmarDon?id=<%=array[0]%>'> Confirmar pago </a>
<p> </p>
<a href='modificarDon?dni=<%=array[6]%>'> Modificar cantidad pago </a>
<p> </p>
```

Figura 4.42 Fragmento código de "UsuarioSin.jsp"

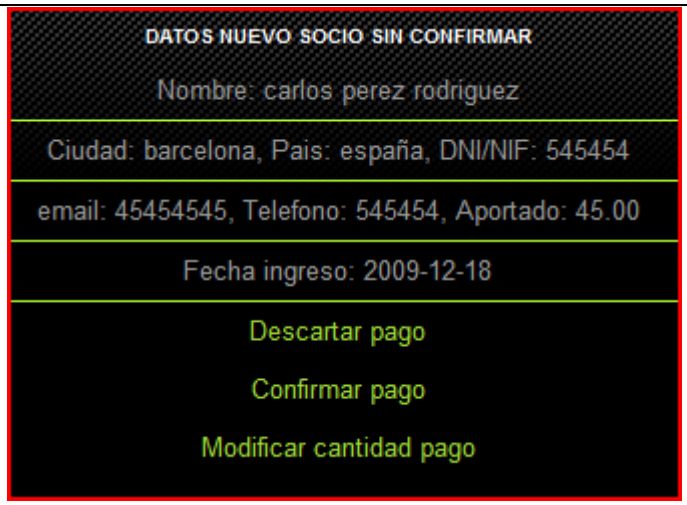


Figura 4.43 Opciones para con una nueva petición de socio

Se disponen de 4 opciones para con el usuario: descartar el pago si pasado un tiempo no llega ningún aviso de pago, confirmar el pago y por lo tanto activar la cuenta de usuario en el sistema y por último, modificar la cantidad si ésta no coincide con la del aviso del banco.

4.4.15 Ver (lista de nuevos pagos en el sistema)

Básicamente se trata de hacer lo mismo que en el capítulo 4.4.14, pero en vez de coger los usuarios sin confirmación de la tabla votantes, se extraen todas las filas de la tabla “Pagos”, ésta contiene los nuevos pagos de socios ya existentes. Se tienen además las mismas opciones: descartar, confirmar y modificar pago (para posibles errores).

4.4.16 Reintegro

Como en el apartado anterior, se muestra una lista con los usuarios que han pedido el reintegro de la cantidad aportada, para ésto se extraen todas las filas de la tabla “Reintegro” de la base de datos. Aunque hay algun cambio con los dos apartados anteriores:

Figura 4.44 Visualización de reintegro por parte de un administrador

En este caso se muestra un dato más: la forma con la que quiere el usuario que le devuelvan el dinero. La única opción que tiene el administrador de la aplicación es confirmar la devolución en la base de datos cuando el dinero llegue al usuario, cuando ésto suceda, los datos y la cuenta del usuario desaparecerán del sistema.

4.4.17 Buscar

Este apartado se utiliza para buscar a socios ya activados en el sistema por dos parámetros de búsqueda: primer apellido y ciudad de residencia. Si no se especifícan ninguno de los dos campos aparecerán en la búsqueda todos los usuarios activos del sistema.

En este caso el administrador no tiene ninguna opción sobre la búsqueda, sólo se le permite la visualización de los datos personales de cada socio. En la siguiente figura se observa un fragmento de código del método de “GestionDatos.java” que realiza la petición a la base de datos. Donde se consulta con una petición distinta dependiendo de los datos que se hayan escrito en el buscador, si no existe ninguna coincidencia se mostrará un error.

```
  public Object [][] buscarActivados(String apellido, String ciudad){

    Object array [][] = null;
     String peticion = null;

     int filasResultSet = 0;
     int i = 0;

     if((apellido=="") && (ciudad=="")){
     System.out.println("No existen coincidencias en la busqueda.");
     peticion = "SELECT id,nombre,apellido1,apellido2,ciudad,dni,aportado FROM
Votante WHERE Votante.activado='si'";
     }
     else if((apellido=="") && (ciudad!="")){
     peticion = "SELECT id,nombre,apellido1,apellido2,ciudad,dni,aportado FROM
Votante WHERE Votante.activado='si' AND Votante.ciudad='"+ciudad+"'";
     }
     else if((apellido!="") && (ciudad=="")){
     peticion = "SELECT id,nombre,apellido1,apellido2,ciudad,dni,aportado FROM
Votante WHERE Votante.activado='si' AND Votante.apellido1='"+apellido+"'";
     }
   else if((apellido!="") && (ciudad!="")){
     peticion = "SELECT id,nombre,apellido1,apellido2,ciudad,dni,aportado FROM
Votante WHERE Votante.activado='si' AND Votante.ciudad='"+ciudad+"' AND
Votante.apellido1='"+apellido+"'";
     }
```

Figura 4.45 Método "buscarActivados" de "GestionDatos.java"

4.4.18 Buscar (con opciones)

El fondo de esta parte de la aplicación es la de buscar socios activos como en el apartado anterior pero con tres opciones de búsqueda: socios que hayan donado más de 10 euros (el mínimo), socios españoles y socios extranjeros.

En la figura 4.46 se observa el funcionamiento de un "option select", según la opción que escojamos se envía un valor diferente al formulario.

En la figura 4.47 se ve un fragmento de código del método que hace la petición a la base de datos.

```
        <p>Diferentes opciones de busqueda de donantes activos</p>
        <form name="input" action="listarOpciones" method="get">
                <SELECT name="tipo">
                <OPTION VALUE="+10">Mas de 10 euros</OPTION>
                <OPTION VALUE="guiris">Fuera de españa</OPTION>
                <OPTION VALUE="spain">Usuarios residentes en España
        </OPTION>
                </SELECT>
                <input type="submit" style="background-color: #ADFF2F"
value="Buscar">
                <br><hr size="1" color="#ADFF2F">
                </form>
```

Figura 4.46 Aplicación de un "option select" en "General.jsp"

```
public Object [][] buscarOpcionesM(String tipo){

    Object array [][] = null;
    String peticion = null;
    int filasResultSet = 0;
    int i = 0;

    if(tipo.compareTo("+10")==0){

    peticion = "SELECT id,nombre,apellido1,apellido2,ciudad,dni,email FROM Votante
WHERE Votante.activado='si' AND Votante.aportado > 10";
    }
    else if(tipo.compareTo("guiris")==0){
    peticion = "SELECT id,nombre,apellido1,apellido2,ciudad,dni,email FROM Votante
WHERE Votante.activado='si' AND NOT Votante.pais='españa'";
    }
    else if(tipo.compareTo("spain")==0){
    peticion = "SELECT id,nombre,apellido1,apellido2,ciudad,dni,email FROM Votante
WHERE Votante.activado='si' AND Votante.pais='españa'";
    }
      else{
    peticion = "SELECT id,nombre,apellido1,apellido2,ciudad,dni,email FROM Votante
WHERE Votante.activado='si'";
    }
```

Figura 4.47 Fragmento método "buscarOpcionesM" de "GestionDatos.java"

4.4.19 Buscar e-mails (con opciones)

Se muestran por pantalla los correos electrónicos de los grupos de socios que hayamos escogido como opción: socios que han donado más de 10 euros, socios españoles, socios extranjeros o todos los socios.

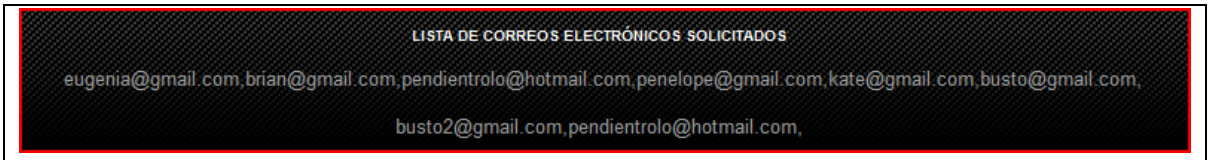


Figura 4.48 Listado de correos electrónicos

4.4.20 Gestión de proyectos

Este apartado sirve al administrador para llevar la gestión de las votaciones de los proyectos del sistema, en la siguiente figura se oberva el aspecto que tiene:

Figura 4.49 Gestión de proyectos

El administrador puede resetear la tabla de votaciones (Cuando se quiera empezar una nueva votación), esta acción implica volver a dar los permisos de voto a todos los usuarios que ya habían votado en la tanda de votaciones que se ha borrado.

Por otro lado se pueden crear proyectos nuevos para las votaciones, aparte de mandar el nombre y la descripción, también se puede adjuntar una fotografía, para que se pueda llevar a cabo se utilizará la librería Java “FileUpload”.

En la siguiente figura (fragmento de código de “GestionProyectos.jsp”) se puede observar que para poder adjuntar un gráfico, el formulario debe utilizar el método “POST” de HTTP y no el método “GET” como hasta ahora se ha utilizado.

```
<form name="input" enctype="multipart/form-data" action="nuevoProyecto"
method="post">
        Rellena los datos para añadir un nuevo proyecto a la votacion:
        <p>  </p>
        Nombre:
        <input type="text" name="nombre">
        <p>  </p>
        Descripcion:
        <textarea name="descripcion" cols="80" rows="8"></textarea>
        <p>  </p>
        Foto:
        <input type="file" name="foto">
<p>  </p>

        <input type="submit" style="background-color: #ADFF2F" value="Añadir">
<p>  </p>
        </form>
```

Figura 4.50 Código formulario utilizando el método POST de HTTP

En la siguiente figura, se observa el código del Servlet llamado “NuevoProyecto.java”, que es el que lleva a cabo la acción de guardar la foto en el directorio que se le haya indicado e insertar la url de ésta en la base de datos. En el código se ofrecen comentarios más específicos:

```
public class NuevoProyecto extends HttpServlet {


  //Método del servlet que responde a una petición POST

 public void doPost(HttpServletRequest request, HttpServletResponse response)
 throws IOException, ServletException
 {

 String nombre=null;
```

```
String fichero=null;
String path=null;
String descripcion=null;
String url=null;

ServletContext application = getServletContext();

try{
  //crea fabrica para los archivos almacenados en el disco

  DiskFileItemFactory factory = new DiskFileItemFactory();

  //crea un nuevo objeto con el que trabajar

  ServletFileUpload upload = new ServletFileUpload(factory);

  // maxima talla permisible (si se excede salta el catch)

  upload.setSizeMax(10000000);

 //se crea la lista de objetos recuperados

  List fileItems = upload.parseRequest(request);

  //obtiene el file enviado

  Iterator j = fileItems.iterator();

  while(j.hasNext()){

  FileItem fi = (FileItem) j.next();

  if(fi.getFieldName().equals("nombre")){

  nombre=fi.getString();

  }

  else if(fi.getFieldName().equals("descripcion")){
  descripcion=fi.getString();

  }
  else{

  //creamos un numero aleatorio para el nombre de cada foto
  //operamos 4 para que sea muchisimo mas dificil que coincida alguno

  int azar1=(int)(Math.random()*100);
  int azar2=(int)(Math.random()*100);
```

```
int azar4=(int)(Math.random()*50);

int azar=azar1*azar2+azar4;
 //indicamos que guardamos el fichero en la carpeta almacen de la raiz de la aplicacion
 path =application.getRealPath("/almacen");
 //ponemos nombre al grafico o foto
 fichero = "foto"+azar+".jpg";
 //guardamos el fichero
 fi.write(new File(path,fichero));
```

Figura 4.51 Código para guardar una imagen en el almacen

4.4.21 Gestión de noticias

Figura 4.52 Gestión de noticias

Básicamente la gestión de noticias es idéntica a la de proyectos, aunque se tiene también la opción de borrar las noticias o artículos de uno en uno. Se pueden borrar todas las noticias de golpe o crear una noticia adjuntándola con una foto tal como pasaba en la gestión de proyectos.

4.5 Diseño gráfico de la web

Sobre el diseño visual de la web, en principio se había pensado realizarlo con "Dreamweaver" a base de etiquetas html. Al tener gran cantidad de archivos JSP con scriplets Java que el programa no reconocía, finalmente se decidió utilizar un archivo ".css" ya diseñado y adaptarlo un poco a lo que queríamos.

Para ello sólo se tiene que incluir las etiquetas "div " correspondientes en los fragmentos de código JSP que se quieran representar en algunas de las partes que nos propone el diseño como vemos en la siguiente figura.

```
<div id="menu">

<a href="general">volver</a>

</div>
```

Figura 4.53 Código para mostrar botones superiores

En este caso se tiene un enlace a la página de inicio de la aplicación, si se inserta dentro de la etiqueta "div menu" se conseguirá que aparezca un botón en la zona superior con las mismas funciones. En el archivo "style.css" se indicará donde está posicionado el enlace a la imagen que representará al botón.

```
<head>
   <meta http-equiv="Content-Type" content="application/xhtml+xml; charset=iso-8859-
1" />
   <title>FARTS</title>
<link rel="stylesheet" href="stylePortada.css">
</head>
```

Figura 4.54 Enlace de archivo JSP con un CSS

En la figura 4.54 se observa como enlazar en el encabezado nuestro archivo JSP con el archivo CSS.

En el sitio www.freecsstemplates.org hay casi 400 diseños distintos de aspecto bastante profesional. Se pueden descargar y utilizar para cualquier tipo de uso (lucrativo o no), siempre que haya un anuncio o un enlace de la web. Para ésta aplicación se descargó el tema llamado “Primitive Element”.

En la siguiente figura se observa el pie de final de página, donde se ha insertado un enlace al sitio web de temas.

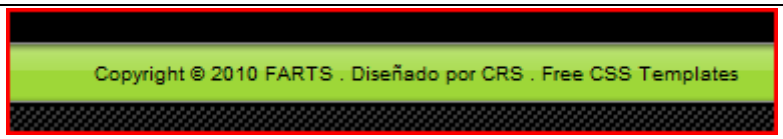


Figura 4.55 Barra de final de página

Para las necesidades actuales del proyecto no se utiliza todo el potencial del diseño, y se estructura la web en tres columnas, siendo la central la más ancha e importante. También se aprovecha el encabezado y la zona superior de botones. Si en un futuro el proyecto avanza lo suficiente como para que haya patrocinadores, anuncios y promociones. El archivo CSS que se ha utilizado puede soportar todo eso como se muestra en la siguiente figura.

Figura 4.56 Vista general de las opciones del CSS

4.6 Protocolo de seguridad: HTTPS (SSL)

En este proyecto también se quería estudiar la opción de dar seguridad a una página web utilizando el protocolo HTTPS, básicamente es idéntico al protocolo HTTP pero con envíos de claves públicas y privadas para asegurar que se está comunicando realmente con el sitio. Si no se reconocen las claves puede ser que haya alguien o algo intentando cazar datos.

El problema de este protocolo de seguridad es conseguir el hosting, como se tienen que hacer configuraciones en ficheros de Tomcat, se nos tiene que proporcionar un servicio dedicado, por lo tanto mucho más caro. Este tema se trata más a fondo en el capítulo de obtención de hosting para la web.

Aun así, para probar el funcionamiento del protocolo en nuestra aplicación, se hará funcionar en nuestro proyecto Tomcat. Para las claves se utilizará la herramienta “Keytool” de Java, no son claves acreditadas por las autoridades de certificados pero para pruebas ya sirven. En el capítulo de obtención de hosting se explica como obtener una clave certificada e insertarla en el sistema (pág.83).

4.6.1 Proceso activación

En primer lugar se tendrá que conseguir un certificado digital. Para generarlo se utilizará la herramienta "Keytool" que nos proporciona Java. Esta utilidad se puede encontrar en el directorio “bin” de la JDK y se utiliza de la siguiente forma desde línea de comandos:

```
keytool  -genkey -alias tomcat -keyalg RSA
```

Una vez iniciado el proceso, se tendrá que contestar a una serie de preguntas que se utilizarán para crear nuestro certificado digital e identificarlo.

El fichero de claves "**.keystore**" que se acaba de generar debería encontrarse en el directorio raíz del usuario actual, es decir:

```
C:\Documents and Settings\MiNombreDeUsuario
```

Ahora se deberá activar el conector para soporte SSL en el fichero "server.xml" que se encuentra en el directorio "conf" de Tomcat. Para ello se tendrá que descomentar el fragmento de código que empieza por: "Define a SSL HTTP/1.1 Connector on port 8444". De esta forma, SSL se activará en el puerto 8444. El puerto por defecto para HTTPS es el 444, por lo que es recomendable cambiarlo si no hay ningún otro servicio que lo utilice. Si se ha cambiado, no hay que olvidarse de modificar también el valor del parámetro "redirectPort" en el resto de conectores que haya definidos.

Se puede indicar la ubicación del fichero de claves si se ha cambiado o si se está utilizando Tomcat como un servicio, indicándolo en la definición del conector SSL con la siguiente línea:

```
keystoreFile="C:\miRuta\.keystore"
```

Si al crear el certificado se ha especificado otra clave diferente de la que Tomcat espera por defecto (*changeit*), se indicará también en la definición del conector SSL de esta forma:

```
keystorePass="nuevoPassword"
```

4.6.2 Forzar https

Con los pasos previos se ha conseguido que la aplicación sea visible tanto por HTTP como a través de HTTPS. Para permitir únicamente el acceso por HTTPS e impedir el acceso a la misma a través de HTTP, se han de incluir las siguientes líneas en el fichero “web.xml” de la aplicación, indicando los directorios o archivos para los que se va a requerir la conexión segura mediante la etiqueta “url-pattern”, en este caso se indica la totalidad de la aplicación:

```
<security-constraint>
 <web-resource-collection>
   <web-resource-name>Entire  Application</web-resource-name>
   <url-pattern>/*</url-pattern>
 </web-resource-collection>
 <user-data-constraint>
   <transport-guarantee>CONFIDENTIAL</transport-guarantee>
 </user-data-constraint>
</security-constraint>
```

Figura 4.57 Marcado de Servlets que se quiere que viajen sobre HTTPS

5 Instalación de la aplicación

Existen bastantes opciones para subir la web a la red, normalmente se utilizará la opción del hosting, pero si se quiere aplicar seguridad SSL se deberá contratar un VPS o un housing. En los siguientes capítulos se explicará como proceder en cada caso.

5.1 Obtención de dominio

Siempre se tiene la opción de contratar el dominio directamente a la empresa que ofrece los servicios de hospedaje, pero también es posible contratar el dominio y más tarde hacer el traspaso a la empresa de hosting.

Se puede obtener un dominio gratuito, el problema es que las extensiones DNS no son muy populares y normalmente obligan a publicitar la empresa.

En el sitio www.midominiogratis.com , se dispone de dominios gratis por tiempo indefinido con contadores de visitas y de usuarios. Los dominios que se pueden registrar son: www.farts.4a2.com , www.farts.tk4.net , www.farts.k25.net y www.farts.km6.net . El problema que se tiene con este tipo de dominios es que no se da una imagen de marca.

Para contratar dominios más populares como “.com” o “.org” hay infinidad de empresas y webs que los ofrecen, con precios y contratos muy variados. Entre todas las opciones que tenemos escogemos la de www.vaxnu.com , que aparte de ofrecer registro de dominios, ofrece servicio de hosting, como se verá más adelante.

En el inicio del portal ya se visualiza un enlace que redirige directamente a la zona de registro de dominios, en la imagen siguiente vemos los precios que tiene cada uno:

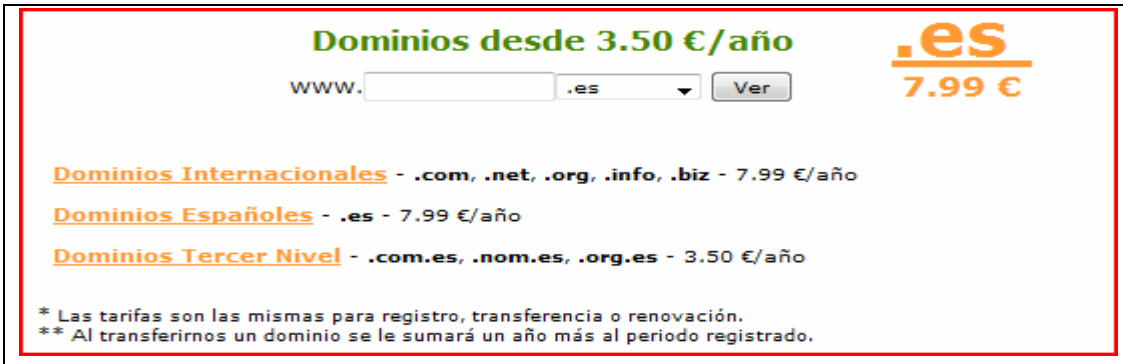


Figura 5.1 Obtención de un dominio en Vaxnu

5.2 Hosting

Si en el primer momento no se quisieran agregar protocolos de seguridad a la web, la opción más barata es la contratación de un servicio de hosting para la aplicación. El problema radica en que para que funcione la aplicación debe haber una máquina Tomcat con las correspondientes librerías, y la empresa Vaxnu ofrece un hosting con todo lo necesario por un precio relativamente bajo. Además, si hay alguna incompatibilidad por las librerías sólo se tendría que mandar vía ftp a la administración del servidor para que la incorporasen.

En el siguiente gráfico se muestra un resumen de las características y el precio del servicio con Tomcat:

PRESTACIONES COMUNES	Micro-Plan NUEVO	Personal	Emprendedor	Avanzado	Profesional	Reseller*
Espacio en Disco	**60 Mb**	**100 Mb**	**400 Mb**	**1 Gb**	**2 Gb**	**10 Gb**
Transferencia Mensual	**0.9 Gb**	**1.5 Gb**	**6 Gb**	**8 Gb**	**15 Gb**	**75 Gb**
Servicios Generales						
Dominios (Sitios Web) + info	1	1	1	1	1	50
Alias de Dominio	Ilimitados	Ilimitados	Ilimitados	Ilimitados	Ilimitados	Ilimitados
Sub-Dominios	0	0	5	5	Ilimitados	Ilimitados
Redireccionamientos	✓	✓	✓	✓	✓	✓
Editor Zona DNS	✓	✓	✓	✓	✓	✓
Cuentas FTP	10	15	50	50	Ilimitados	Ilimitados
Mensajes SMS + info	✓	✓	✓	✓	✓	✓
Tickets de Soporte	✓	✓	✓	✓	✓	✓
Backups Diarios	✓	✓	✓	✓	✓	✓
Estadísticas	✓	✓	✓	✓	✓	✓
Servicios de E-mail						
Cuentas de E-mail	10	15	50	50	Ilimitados	Ilimitados
POP3	✓	✓	✓	✓	✓	✓
IMAP	Solo Linux	Solo Linux	Solo Linux	Solo Linux	Solo Linux	Solo Linux
Webmail	✓	✓	✓	✓	✓	✓
Mailing Lists	✓	✓	✓	✓	✓	✓
Autorespondedores	✓	✓	✓	✓	✓	✓
AntiSpam	✓	✓	✓	✓	✓	✓
AntiVirus	Solo Windows	Solo Windows	Solo Windows	Solo Windows	Solo Windows	Solo Windows
SendGuardian + info NUEVO	Opcional	Opcional	Opcional	Opcional	Opcional	Opcional

Figura 5.2 Ofertas y precios de Hosting en Vaxnu

El servicio de Tomcat se nos ofrece a partir del plan "Emprendedor" y utilizando plataforma Windows (aunque no utilizaramos Tomcat este plan se tendría que contratar igualmente ya que los dos planes anteriores no disponen de la transferencia mensual necesitada):

WINDOWS						
	Contratar >>>	Contratar >>>	Contratar >>>	Contratar >>>	Contratar >>>	Contratar >>>
Windows Server 2003	✓	✓	✓	✓	✓	✓
MS SQL Server	1	1	5	5	Ilimitados	Ilimitados
MS Access	1	1	5	5	Ilimitados	Ilimitados
MySQL Database	1	1	5	5	Ilimitados	Ilimitados
DSNs ODBC	1	1	5	5	Ilimitados	Ilimitados
ASP	✓	✓	✓	✓	✓	✓
.NET (1 y 2)	✓	✓	✓	✓	✓	✓
PHP 5	✓	✓	✓	✓	✓	✓
JAVA TOMCAT + info	-	-	Opcional	Opcional	Opcional	Opcional
WAP	✓	✓	✓	✓	✓	✓
CGI/Perl	✓	✓	✓	✓	✓	✓
Ext. MS FrontPage	✓	✓	✓	✓	✓	✓
Shockwave Flash	✓	✓	✓	✓	✓	✓
Panel Control DNP	✓	✓	✓	✓	✓	✓
CDO SYS	✓	✓	✓	✓	✓	✓
JMail	✓	✓	✓	✓	✓	✓
ASPEmail	✓	✓	✓	✓	✓	✓
ASPJpeg	✓	✓	✓	✓	✓	✓
ASPUpload	✓	✓	✓	✓	✓	✓
ADO.NET	✓	✓	✓	✓	✓	✓
ODBC.NET	✓	✓	✓	✓	✓	✓
Tipos MIME	✓	✓	✓	✓	✓	✓
Protección Hotlink	✓	✓	✓	✓	✓	✓
URLs Protegidas	✓	✓	✓	✓	✓	✓

	Micro-Plan NUEVO	Personal	Emprendedor	Avanzado	Profesional	Reseller*
URLs Protegidas	✓	✓	✓	✓	✓	✓
Tareas Programadas	✓	✓	✓	✓	✓	✓
Aplicaciones Preinstaladas	✓	✓	✓	✓	✓	✓
Páginas de Error	✓	✓	✓	✓	✓	✓
TARIFAS	19€/año	29€/año	37€/semestre	56€/semestre	21€/mes	41€/mes
	Contratar >>>	Contratar >>>	Contratar >>>	Contratar >>>	Contratar >>>	Contratar >>>

+ Alta GRATIS en todos los planes.
* Si necesita más cantidad, el Plan Reseller puede contratarse en múltiplos (x2, x3...). Consúltenos

Figura 5.3 Precios y características de Hosting en Vaxnu

Con el plan emprendedor se disponen de 400 Mb de espacio en disco y 6 Gb de transferencia mensual. El objetivo del proyecto es llegar a un millón de donantes, pero esa cifra no se logra con facilidad por tanto con esas características se tendría bastante durante una buena temporada mientras el volumen de negocio crece.

La aplicación en sí pesa unos 4 Mb, por lo tanto restan 496 Mb para datos de clientes que para empezar es más que suficiente. Para la transferencia mensual de datos se ha hecho una aproximación utilizando una calculadora como la del siguiente enlace http://www.abcdatos.com/webmasters/transferencia.html.

Se han supuesto 100 visitas al día. Realizando distintos modos de visita (Darse de alta como usuario, votar proyectos, leer un par de noticias, etc…) se cargan de media unas 15 páginas. Cada página JSP ocupa de media 100 Kb (contando las páginas con imágenes). Lo que da un total de 5,36 Gb , por lo tanto con los 6 Gb comentados anteriormente sería suficiente para la aplicación.

Resumiendo, el plan emprendedor con dominio incluido tendría un precio de **74 €/año**, añadiendo los **48 €/año** que costaría el servicio Tomcat, hacen un total de **122 €/año.**

Para instalar la aplicación bastará con subir el archivo.sql que inicia la base de datos (viene más detallado en el capítulo de bases de datos (pág.15)) y el archivo.war que lleva todo lo necesario para que un servidor Tomcat haga funcionar la aplicación.

Estructura del war

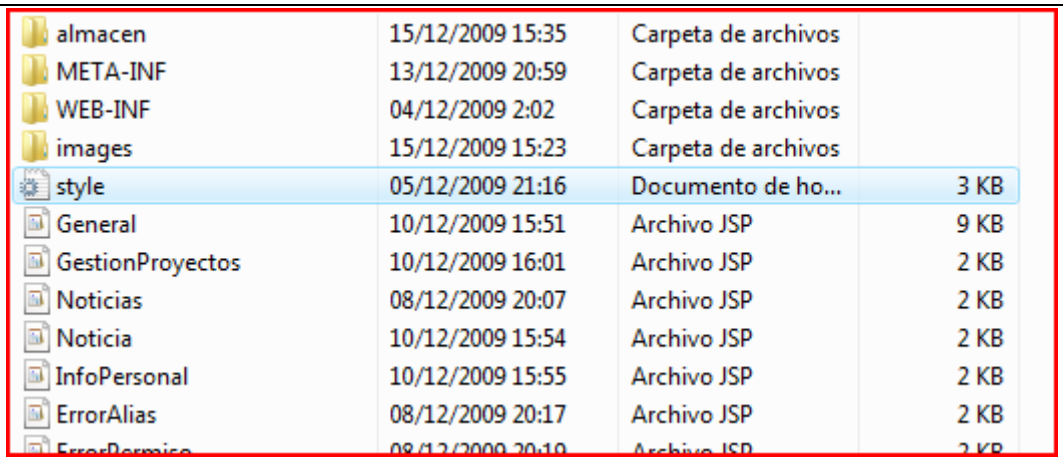


Figura 5.4 Estructura del archivo .war

En la figura anterior se muestra la raíz de un war.

- **almacen:** Es donde se guardan las imágenes que se adjuntan a proyectos y a noticias.
- **META-INF:** Es un directorio con información de la aplicación que se lee cuando el WAR está siendo descomprimido. Es decir, sólo funcionará cuando se esté desplegando la aplicación desde un WAR.
- **WEB-INF:** Se muestra el contenido en el siguiente gráfico.
- **images:** Contiene las imágenes que necesita el CSS para la presentación visual de la web.
- **JSP's y CSS:** aparte de los directorios anteriores en el resto de la raíz se tienen todos los archivos JSP y archivos CSS de la aplicación.

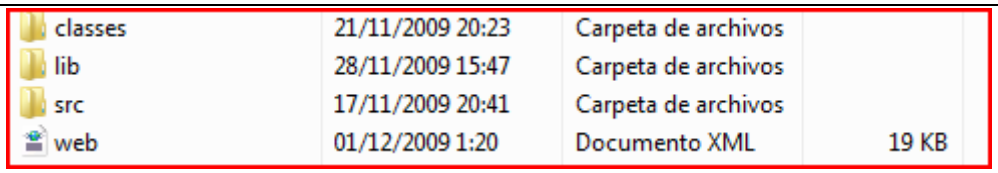


Figura 5.5 Contenido WEB-INF

En la figura 5.5 se muestra el contenido del directorio WEB-INF:

- **classes:** Contiene todos los archivos compilados Java de la aplicación, es decir todos los CLASS, adems está dividido en tres subcarpetas: beans,

conexión y servlets. En donde se coloca cada CLASS en el grupo que le corresponda.

- **src:** Este directorio no es obligatorio, contiene de la misma forma que "classes" en tres subcarpetas, los archivos Java en formato texto por si hay que cambiar o consultar alguna cosa.
- **web.xml:** Archivo donde se configura la aplicación, podemos verlo en detalle en la página 28.
- **lib:** Este directorio contiene todas las librerías adicionales con las que no cuenta Tomcat por defecto. En la siguiente imagen se listan los drivers o librerías que necesitamos.

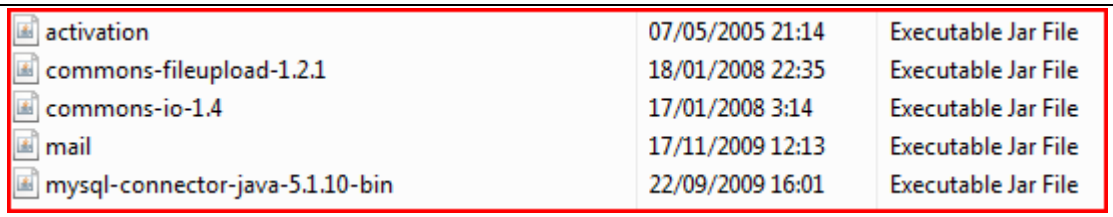


Figura 5.6 Contenido directorio "lib"

Los dos archivos JAR que llevan "commons" en el nombre sirven para poder guardar las imágenes que se adjunten en el sistema. "activatio.jar" y "mail.jar" sirven para autenticar un cliente de correo y enviar un correo respectivamente. Por último "mysql-connector" tiene la función de conectar el proyecto con una base de datos MySQL. En el capítulo de instalación del equipo (pág.94) tenemos más información sobre estos drivers.

5.3 Diferencias entre housing y VPS

La diferencia entre estos dos planes de alojamiento es muy grande y conocerlas de antemano será la clave para decidirnos a la hora de encontrar alojamiento web para el futuro desarrollo del proyecto.

La mayoría de las empresas de hosting ofrecen directamente las dos variantes y cada una de ellas tiene sus propias ventajas y desventajas. En último lugar la decisión se verá afectada por las necesidades de nuestro desarrollo, el precio y la forma en que deseamos manejar nuestra presencia en la web.

La diferencia principal entre un servicio de housing y un servidor dedicado es la propiedad del servidor.

En los planes de servidores dedicados, al cliente se le ofrece el uso exclusivo de un servidor que pertenece al proveedor de hosting mientras que en la modalidad de housing el servidor pertenece al cliente y este servidor es alojado en un espacio físico en las instalaciones del proveedor junto a los demás servidores de la compañía o de otros clientes.

Resumiendo, en la modalidad de servidor dedicado lo que se alquila es el servidor mientras que en la modalidad de housing lo que se alquila es el espacio físico en las instalaciones del proveedor.

Cada uno de estos planes de alojamiento es más barato que la compra de un servidor y el alojamiento particular en su casa o empresa, pero hay ciertas diferencias entre lo que puede hacer o no con el equipamiento en cada uno de estos planes que conviene conocer para evitar problemas en el futuro.

5.4 VPS (Servidor Privado Virtual)

Si se quiere incorporar el servicio de seguridad (SSL), no basta con un hosting, ya que hay que hacer una serie de cambios en la máquina Tomcat y la máquina disponible en el servicio de hosting es dedicada (Misma configuración para todos los usuarios).

La primera de las opciones que se ofrece es la de un servidor virtual dedicado, se utiliza la tecnología VMware para crear múltiples particiones aisladas (VPS’s) en un único servidor físico individual, obteniendo el máximo rendimiento de la máquina.

El VPS permite reservar parte de los recursos para un usuario en exclusiva, dentro de una máquina compartida con otros clientes. Aun así se acerca mucho al concepto de servidor dedicado, ya que los clientes no pueden consumir los recursos asignados a otros usuarios, y el usuario tiene privilegios de administrador, con su propio sistema operativo independiente.

Vaxnu ofrece un servicio con “Plesk Power Pack”, que es un panel de administración que tiene una herramienta para Java Servlets, por lo tanto permite desplegar y gestionar aplicaciones Tomcat y hacer los cambios necesarios en el archivo Server.xml:

PRESTACIONES COMUNES	S Dedicado G1	S Dedicado G2	S Dedicado G4	VPS 128	VPS 256	VPS 512
Procesador	Celeron 3.2 Ghz	Pent Core2 Duo 2.33 Ghz	Xeon QuadCore 2.4 GHz	-	-	-
Memoria RAM	512 Mb	1 Gb	1 Gb	128 Mb	256 Mb	512 Mb
Disco duro	160 Gb	160 Gb	160 Gb	40 Gb	60 Gb	80 Gb
Transferencia mensual	500 Gb	1000 Gb	1500 Gb	100 Gb	200 Gb	300 Gb
Giga Ethernet 10/100/1000	2	2	2	-	-	-
Unidad DVD-ROM	✓	✓	✓	-	-	-
Unidad Disquete	✓	✓	✓	-	-	-
Garantia hardware ilimitada	✓	✓	✓	✓	✓	✓
Direcciones IP	2	2	2	1	1	1
IP inversa personalizable	✓	✓	✓	✓	✓	✓
Reset Remoto (por hardware)	✓	✓	✓	-	-	-
Monitorización y Alertas	✓	✓	✓	Opcional	Opcional	Opcional
OPCIONES						
+ 512 Mb RAM	+ 9.95€/mes	-	-	-	-	-
+ 1 Gb RAM	-	+ 9.95€/mes	+ 9.95€/mes	-	-	-
Espacio Backup FTP + info	+0.50€/mes/Gb	+0.50€/mes/Gb	+0.50€/mes/Gb	+0.50€/mes/Gb	+0.50€/mes/Gb	+0.50€/mes/Gb
SendGuardian + info NUEVO	+ 0.25€/mes	+ 0.25€/mes	+ 0.25€/mes	+ 0.25€/mes	+ 0.25€/mes	+ 0.25€/mes
Panel DirectAdmin (solo Linux)	+ 19€/mes	+ 19€/mes	+ 19€/mes	+ 19€/mes	+ 19€/mes	+ 19€/mes
Panel Plesk 30 Dominios	+ 9€/mes	+ 9€/mes	+ 9€/mes	+ 9€/mes	+ 9€/mes	+ 9€/mes
Panel Plesk 100 Dominios	+ 15€/mes	+ 15€/mes	+ 15€/mes	+ 15€/mes	+ 15€/mes	+ 15€/mes
Panel Plesk 300 Dominios	+ 24€/mes	+ 24€/mes	+ 24€/mes	+ 24€/mes	+ 24€/mes	+ 24€/mes
Panel Plesk Ilimitados Dominios	+ 29€/mes	+ 29€/mes	+ 29€/mes	+ 29€/mes	+ 29€/mes	+ 29€/mes
Plesk Power Pack (Tomcat...)	+ 19€/mes	+ 19€/mes	+ 19€/mes	+ 19€/mes	+ 19€/mes	+ 19€/mes

Figura 5.7 Ofertas y características VPS en Vaxnu

LINUX						
CentOS	✓	✓	✓	✓	✓	✓
SSH telnet	✓	✓	✓	✓	✓	✓
CUOTA DE ALTA	GRATIS	GRATIS	GRATIS	GRATIS	GRATIS	GRATIS
CUOTA MENSUAL	99€*	155€*	199€*	29.95€*	39.95€*	49.95€*
SI OPTA POR PAGO ANUAL	1045.44€ (87.12€/mes)	1636.80€ (136.40€/mes)	2101.44€ (175.12€/mes)	316.27€ (26.36€/mes)	421.87€ (35.16€/mes)	527.47€ (43.96€/mes)
	Contratar >>>	Contratar >>>	Contratar >>>	Contratar >>>	Contratar >>>	Contratar >>>
WINDOWS						
Windows Server 2003	-	✓	✓	-	-	✓
Escritorio Remoto	-	✓	✓	-	-	✓
CUOTA DE ALTA	-	GRATIS	GRATIS	-	-	GRATIS
CUOTA MENSUAL	-	155€*	199€*	-	-	49.95€*
SI OPTA POR PAGO ANUAL	-	1636.80€ (136.40€/mes)	2101.44€ (175.12€/mes)	-	-	527.47€ (43.96€/mes)
	-	Contratar >>>	Contratar >>>	-	-	Contratar >>>
	Servidor G1	Servidor G2	Servidor G4	VPS 128	VPS 256	VPS 512

* DESCUENTO: Si contrata por periodos mayores a un mes (2, 3, 6 ó 12 meses) tendrá hasta un 12% de descuento.
- La disponibilidad de los servidores puede variar. Consúltenos

Figura 5.8 Precios de VPS en Vaxnu

En las dos imágenes anteriores se ven las diferentes ofertas de VPS. El “Plesk Power Pack” cuesta unos **228 €/año** y un certificado SSL **49 €/año**. El VPS más barato para Linux son **416,27 €/año**, por el contrario para Windows sólo hay una opción, aunque ésta sea más potente que la de Linux tiene un precio de **527,47 €/año**.

Por tanto, la opción Linux tiene un precio de **584,27 €/año** y la de Windows **794,47 €/año.** Es importante saber que para servicios de más de tres meses hay descuentos de hasta un **12%.** En este caso no existe problema con la transferencia mensual, ya que se ofrece 100 y 400 Gb respectivamente, y por lo menos al principio estimando unos 1.000 socios, con unos 5 o 6 Gb bastaría.

5.5 Instalar clave certificada en el .keystore

En otro capítulo (pág.72) se explicaba como activar una clave no-certificada en un "keystore" para provar el funcionamiento de HTTPS en Tomcat, pero ahora se explicará como incluir en este "keystore" claves autenticas y certificadas. Se pueden conseguir por empresas que se dediquen a eso como www.verisign.com , pero los mejores precios que podemos encontrar son muy parecidos a los precios a los que te las ofrecen las empresas de hosting, asi que se decide contratar una clave certificada que nos ofrece Vaxnu en sus servicios de VPS o housing.

1. Repetir los pasos descritos en el capítulo anterior en la página 72.

2. Generar el CSR. El fichero resultante de esta operación será lo que se enviará a la CA (autoridad certificadora).

```
$ keytool -certreq -alias autentiaCert -file autentia.csr -keypass
claveDeAutentiaCert -storepass claveDeKeyStore
```

- **-alias:** Nombre del "alias" que hace referencia al certificado creado en el paso anterior.
- **-file:** Nombre del fichero de salida, que más tarde se enviará a la CA.
- **-keypass:** Es la clave con la que se podrá acceder a la clave privada del par de claves creadas.
- **-storepass:** Clave para acceder al "keystore".

3. Enviar el fichero "autentia.csr" a la CA para que devuelva el certificado firmado por ellos. De esta manera la CA acredita el certificado, de forma que el cliente que lo reciba no tendrá duda de que se es quien se dice ser.

4. El último paso es instalar el nuevo certificado firmado por la CA. Para esto se escribirá:

```
keytool -import -alias autentiaCert -keypass claveDeAutentiaCert -file
autentiaCertFirmadoPorCA.pem -storepass claveDeKeyStore
```

El alias es el mismo que se usa en el paso 5.1. Es decir, se está sustituyendo el certificado autofirmado por el certificado firmado por la CA.

Si el certificado de la CA estaba guardado en el "cacerts keystore" hay que añadir la opción "-trustcacerts":

```
keytool -import -alias autentiaCert -keypass claveDeAutentiaCert -file
autentiaCertFirmadoPorCA.pem -storepass claveDeKeyStore -trustcacerts
```

Ahora se podrá ver el contenido del almacén de claves:

```
keytool -list -v -storepass claveDeKeyStore
```

Obsérvese que ahora el emisor de "autentiaCert" es la empresa de certificación.

5.6 Housing

Si se quiere activar el protocolo HTTPS, aparte del VPS, se tiene la opción del housing o colocation. Se trata de llevar a un centro de datos una máquina propia y que ésta actúe como un servidor. La administración de esta máquina puede llevarse a cabo por la empresa o por los mismos desarrolladores de la web.

Para este servicio se ha elegido una empresa con sede en Barcelona llamada www.centrodedatos.com , ya que es importante tener cerca el lugar físico donde va a estar la máquina, para solucionar posibles problemas.

En este caso se dispone de dos opciones: se puede alquilar el espacio físico para una torre o 1U, en este último caso se debería comprar un servidor rackeable.

Figura 5.9 Ofertas Housing (Colocation) en Centro de Datos

En los dos casos se ofrece gratis una transferencia mensual de 1Gb, pero como no es suficiente se debería contratar una transferencia de 10 Gb que tiene un coste de **40 €/mes.**

-Torre

Inicial: Equipo Dell Inspiron 546 (Doble núcleo)……**478 €**

Alta en la empresa de housing……..**40 €**

Mensual: Servicio de housing……...**100 €**

Clave certificada……**49 €**

Total: **408 € + 149 €/mes**

-1U

Inicial: Servidor Dell PowerEdge R200……...**579 €**

Alta en la empresa de housing……..**40 €**

Mensual: Servicio de housing……...**60 €**

Clave certificada……**49 €**

Total: **609 € + 99 €/mes**

La ventaja más importante que se obtiene es que el equipo se configura por los mismos desarrolladores de la web antes de llevarlo al centro de datos, se puede configurar todo lo necesario sin ningún tipo de restricciones, en el siguiente capítulo se explica como preparar un equipo para que pueda soportar la aplicación del proyecto.

Por otro lado, el servicio es mucho más caro que un hosting o un VPS, pero en el precio se incluye el gasto de luz, la tarifa de Internet e incluso se ahorra en posibles alquileres de locales

5.7 Instalar equipo desde cero

No hay grandes diferencias en lo que se refiere a instalar todo el software necesario entre sistemas Unix y Windows, aunque igualmente se explicaran los pequeños detalles que lo diferencian.

5.7.1 MySQL 5.1

Windows

Se accede a la web oficial de MySQL y se descarga la última versión gratuita disponible llamada “MySQL Community Server” http://dev.mysql.com/downloads/mysql/5.1.html .

Ejecutando este archivo el sistema nos muestra un asistente para la instalación del programa:

Figura 5.10 Ventana instalación MySQL

En las sucesivas pantallas que se van mostrando al pulsar "Next" hay que seleccionar los siguientes datos (aunque siempre va a depender de nuestras propias necesidades):

- Detailed Configuration
- Server Machine
- Transactional Database Only
- Se deja todo por defecto
- Decision Support (DSS) OLAP
- Se marca la casilla "Enable TCP/IP Networking", se establece "Port Number" a 4406, se marca la casilla "Add firewall exception for this port" y por último "Enable Strict Mode".
- Se selecciona la opción "Best Support For Multilingualism" para establecer el encoding de la base de datos a UTF-8.
- Se marcan las casillas "Install As Windows Service" e "Include Bin Directory in Windows PATH", dejando el nombre del servicio por defecto.
- Se marca la casilla "Modify Security Settings", estableciendo como usuario "root" y como contraseña "admin" (o la que se quiera).
- Se pulsa "Execute" para que comience el proceso de configuración y cuando finalice ya se puede finalizar.

Para comprobar que la instalación de MySQL se ha hecho correctamente bastará con abrir una consola y teclear “mysql -u root -p”, introduciendo la contraseña establecida anteriormente. El sistema mostrará una pantalla parecida a esta:

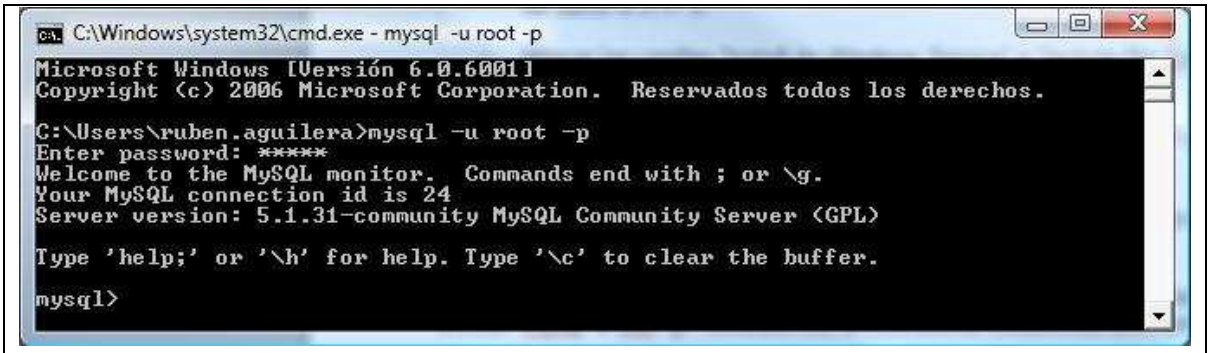


Figura 5.11 Comprobación funcionamiento MySQL

Para darle formato a la base de datos mysql y poder crear tablas hay que introducir los siguientes comandos:

```
Create database nombreBBDD;
```

Se llamará como en el ejemplo: “mysql”

```
Create database mysql;

source bbdd.sql;
```

Con este último comando se consigue cargar el archivo que crea las tablas necesarias.

Linux

Para Instalar MySQL 5.1 Server en un sistema Unix hay que descargar la versión correspondiente de http://dev.mysql.com/downloads/mysql/5.1.html .

Desde un terminal se escribe el siguiente comando:

```
sudo apt-get install mysql-server
```

Para crear una nueva base de datos hay que escribir:

```
mysqladmin create <databasename>
```

5.7.2 Instalar Java y Tomcat

Windows

Se descarga la última version de Java SE Development Kit (JDK) en http://java.sun.com/javase/downloads/index.jsp , ejecutar el archivo e instalar normalmente.

Una vez instalado Java, añadir al "PATH" del sistema la ruta del kit de desarrollo para facilitar la utilización de Java desde el terminal de comandos. Para ello, se siguen los siguientes pasos:

1. Desde una ventana del Explorador de Windows o desde el Escritorio, seleccionar Mi PC y pulsar el botón derecho del ratón.
2. Se elige la opción de "Propiedades", para visualizar las propiedades de sistema.
3. Opciones Avanzadas
4. Pulsar en "Variables de Entorno" para actualizar las variables del sistema. Tras esto se da a "Añadir" o a "Modificar" la variable PATH
5. Si la variable PATH existe:
    - Se selecciona y modifica
    - En el valor de variable, al principio, se introduce el directorio del entorno Java y se separa del resto con un punto y coma: **c:\java\jdk\bin;*resto_de_valores***
6. Si la variable PATH no existe:
    - Seleccionar "Nueva"
    - En nombre de variable introducir el **PATH** y en el valor de variable el directorio del entorno java: **c:\java\jdk\bin**
7. Pulsamos "Aceptar" para guardar los cambios.

Para comprobar que se puede acceder sin ningún problema al compilador de Java y a la máquina virtual, basta con abrir un terminal y escribir:

```
javac -version
java -version
```

Se puede descargar la última versión de Tomcat (La versión 6 para Windows) en http://tomcat.apache.org :

Figura 5.12 Ventana instalación Tomcat

Se ejecuta el fichero descargado, que será del tipo “apache-tomcat-5.5.XX.exe”.

- Lo primero que se visualiza es la pantalla de bienvenida al asistente de instalación de Apache Tomcat, se pulsa el botón “Next”.
- Aparecerá la licencia del producto. Se acepta pulsando el botón “I agree”.
- Posteriormente se muestra la pantalla de selección de componentes de la instalación, seleccionar todas las opciones (En la lista desplegable *Select the type of install*, opción **full**) y pulsar el botón “Next”.
- A continuación aparece el directorio de instalación, se cambia a **c:\java\tomcat**. Pulsar el botón “Next”.

- Después se nos pide el puerto que va a utilizar Apache Tomcat, se cambia al valor **8080** y el nombre del administrador de Tomcat y su contraseña.
- Acto seguido, preguntará donde está la máquina virtual Java (JRE - Java Runtime Environment), por defecto en **c:\java\jre**. Se pulsa el botón "Install" y comenzará la instalación en si.
- Para terminar de instalar Tomcat se pulsa el botón "Finish", dejando marcada la opción "Run Apache Tomcat".
- Una vez instalado Tomcat, en la bandeja del sistema, aparecerá un pequeño icono con el cual se podrá parar, iniciar y configurar el servidor Apache Tomcat.

Si desde el navegador se escribe la dirección **http://localhost:8080** debería aparecer la página de bienvenida de Apache Tomcat.

Figura 5.13 Comprobación funcionamiento Tomcat

Importante: Aunque se visualice la página principal de Tomcat (JSP), para que se puedan ejecutar los servlets hay que añadir al CLASSPATH la ruta de "servlet-api.jar", un driver que se ubica en el directorio "lib" de Tomcat.

Linux

Para instalar la máquina Java:

- Primero se descarga el último JDK de http://java.sun.com/javase/downloads/index.jsp . Al hacer clic en el botón "DESCARGAR" correspondiente a la versión del JDK, se abrirá una página

donde se puede elegir sistema operativo y el idioma. Se debe seleccionar Linux (o Linux 64 si se ha instalado una versión de 64 bits) y marcar la casilla de aceptación de licencia. Después se pulsa en “Continuar”.

- Aparecerán varios paquetes (seguramente 2: un *rpm.bin* y un *.bin*), se descarga el binario *.bin*, por ejemplo: *jdk-6u14-linux-i586.bin.*
- Una vez descargado el archivo binario, se abre una consola en el directorio donde se encuentre, y después se dan los permisos de ejecución con el comando:

```
chmod +x *.bin
```

- Hecho ésto se podrá ejecutar el binario, lo que creará un directorio llamado jdk1.6.0_14, para ello se ejecuta en consola:

```
./jdk-nombre.bin
```

- Se acepta la licencia, y será creado el directorio en la ruta donde se encuentre el binario.
- Se debe copiar el directorio generado al home de nuestro usuario (por ejemplo, /home/java), para ello se ejecuta lo siguiente:

```
mv jdk1.6.0_14 ~/
```

- Ahora lo único que falta es añadir al PATH (variable del entorno que contiene las rutas donde el terminal busca los comandos) el directorio *bin* que hay dentro de la carpeta manejada. Se abre con “gedit· (u otro editor) el archivo ~/.bashrc con el siguiente comando:

```
gedit ~/.bashrc
```

- Se añade al final del PATH (recién instalado puede que el fichero .bashrc no exista aún, si es así, se crea) lo siguiente:

```
PATH=/home/java/.jdk1.6.0_14/bin/:$PATH
```

Ahora ya se dispone de los comandos *javac*, *java*, etc… en nuestro terminal.

Para la instalación de Tomcat:

1. Descargar el fichero apache-tomcat-6.0.20.tar.gz al directorio temporal (/tmp). Este fichero también está disponible para descarga en el sitio Web de descargas de Apache Tomcat 6.x.
2. Se extraen los ficheros de Tomcat en algún directorio de la cuenta. Para ello, una vez situado en el directorio, se ejecuta el comando:

```
tar xvzf /tmp/apache-tomcat-6.0.20.tar.gz
```

3. Se establece la variable de entorno "CATALINA_HOME" para que apunte al directorio donde se ha instalado el servidor Tomcat.
4. Se establece la variable de entorno "JAVA_HOME" para que apunte al directorio donde está instalado Java 1.6 (por ejemplo: home/java/.jdk1.6.0_14/).
5. Se antepone el directorio "${JAVA_HOME}/bin" a la variable de entorno "PATH".

Ahora ya se podrá ejecutar el servidor Tomcat y empezar a probar los primeros Servlets. Para arrancar y detener Tomcat hay que seguir las siguientes instrucciones:

1. El script de arranque se encuentra en el directorio "bin" de la instalación de Tomcat. En este mismo directorio se ejecuta en el terminal: "./startup.sh"
2. Para detener el servidor se debe ejecutar el siguiente comando en la shell desde el mismo directorio: "./shutdown.sh".
3. Para comprobar que la instalación ha sido satisfactoria, se arranca un servidor Tomcat y en el navegador se escribe: http://localhost:8080 . Si se visualiza la página principal de Tomcat, significa que la instalación es correcta.
4. Para compilar servlets, se deberá incluir en la variable de entorno CLASSPATH la ruta al API del Servlet, así como el directorio actual. Esta API es necesaria para compilar un servlet, y está disponible en la instalación de Tomcat: `lib/servlet-api.jar`. Una opción alternativa es compilar con la opción `-classpath`, por ejemplo:

```
javac -classpath ${CATALINA_HOME}/lib/servlet-api.jar:. HolaMundo.java
```

5.7.3 Instalación de librerías y drivers

Hay partes de la aplicación Java-Tomcat que necesitan de librerías o drivers que no estan en las instalaciones básicas.

Lo único que se deberá hacer es añadir la ruta de cada JAR al CLASSPATH del sistema para que el compilador Java sepa donde encontrarlos, se necesitarán cinco “drivers” distintos:

Se necesita un driver para conectar Java con MySQL para que se puedan hacer las peticiones a la base de datos.

En http://dev.mysql.com/downloads/connector/j/5.1.html , se elige el sistema operativo y tras descargarse se busca el archivo con nombre “mysql-connector-java-5.1.10-bin.jar” la ruta al cual es la que se añade al CLASSPATH.

Para el envío y autenticación de correo electrónico se necesitará el API de JavaMail, se descarga de http://java.sun.com/products/javamail/downloads/index.html , se extraen los archivos “activation.jar” y “mail.jar” y se copia la ruta donde se guarden al CLASSPATH.

Finalmente para que se puedan adjuntar imágenes y que estas se guarden en un contenedor de la aplicación, se descargan dos API de Apache Commons de http://commons.apache.org/io/ y http://commons.apache.org/fileupload/ y se extraen respectivamente el archivo “commons-io-1.4.jar” y “commons-fileupload-1.2.1.jar” añadiendo la ruta donde se guarden en el CLASSPATH como en todos los casos anteriores.

6 Presupuesto

Éste sitio web ha sido creado desde cero por lo tanto se contabilizará desde las reuniones previas para discernir los puntos necesarios, hasta los gastos indirectos para la amortización de equipos, ya que el caso tiene que ser lo más parecido a los presupuestos de una empresa de diseño y/o mantenimiento web.

Se cobrarán **30 €/hora** por todo lo que es la creación del sitio en sí, para todo lo que sea gastos indirectos se estudiarán los diferentes casos.

Gastos directos

Se han realizado reuniones previas con el cliente para tener constancia de lo que se requiere. Para esto se utilizaron 5 horas, que resultan un total de **150 €.**

Despues se tuvieron que decidir los puntos importantes de la aplicación y como llevar a cabo el sitio realizando un esquema lógico de la aplicación, se utilizaron 25 horas, por tanto **750 €.**

Para el diseño en sí de la aplicación fueron necesarias 150 horas, que suman un total de **4.500 €.**

Tras el diseño se hicieron una serie de pruebas y cambios para el correcto funcionamiento del sitio, fueron necesarias 15 horas, que resultan **450 €.**

Finalmente se creó un manual de usuario y un informe donde se explican las diferentes maneras de subir el sitio a la red. Fueron necesarias 20 horas, sumando un total de **600 €.**

Gastos indirectos

Se tiene que cobrar la amortización del equipo informático, se cuenta que cada dos años se tiene que cambiar. Por lo tanto por 2 meses de trabajo añadimos a la cuenta **166,7 €,** ya que el equipo tiene un valor de **2.000 €.**

Se ha utilizado “Windows Vista” para la realización del sitio, pero como éste venía incluido con la compra del equipo, no se carga nada a la cuenta final. El cliente dejo claro que todos los informes y cuentas se tenían que presentar en formato “Microsoft Office”, por lo tanto se añade a la cuenta también la amortización del producto. Ésta se estima más o menos en 2 años. El paquete tiene un valor de **99 €** por tanto se añaden a la cuenta **8,25 €.** El resto del software utilizado se distribuye libremente por tanto no se sumará a los gastos.

Como se ha indicado anteriormente el proyecto se ha realizado en 2 meses, y como la empresa no trabaja para más de un cliente a la vez, se cargarán los gastos en alquiler de local, de luz y finalmente de conexión a Internet. El alquiler del local tiene un coste de **800 €,** de luz se tiene un gasto medio de **25 €** y de conexión de Internet se abonan **29,9 €** al mes. Por tanto se cargarán a la cuenta final **1600 €, 50 €** y **59,8 €** respectivamente.

Factura

Reuniones previas	**150**
Esquema lógico de la aplicación	**750**
Diseño	**4.500**
Pruebas	**450**
Manuales	**600**
Amortización equipo	**166,7**
Amortización software	**8,25**
Amortización local	**1600**
Amortización luz	**50**
Amortización conexión a Internet	**59,8**
TOTAL	**8.334,75 €**

Figura 6.1 Presupuesto

7 Conclusión

Al comenzar el proyecto no tenia muy claro si todo los requerimientos previos podían ser realizados por la tecnología de Servlets y JSP's. Pero la realidad es que existen bastantes API pensadas para Tomcat. Y pensando en todo el potencial que tiene Java es una tecnología que podría hacer casi de todo, ya que siempre que tengamos las librerías necesarias podremos insertar cualquier código Java en un archivo JSP.

Uno de los problemas también más difíciles a los que me he enfrentado en este proyecto es saber o intuir que es lo que quiere el cliente. Ya que en este caso la organización no tenía del todo claro lo que querían, y de unas vagas ideas he tenido que construir una aplicación web. Para esto es muy importante discernir los puntos importantes y los que no lo son tanto.

Tras haber acabado todo el diseño, apareció otro problema importante, los usuarios que entren a la web deben intuir el funcionamiento. Para el propio desarrollador es fácil entender como va todo, por eso también viene bien ponerse en la piel de los usuarios y diseñar la web de una forma más intuitiva.

También ha sido importante la realización de la memoria, ya que te acerca un poco más a lo que será el mercado laboral donde los informes y actas son el pan de cada día. Y aunque parezca fácil por haberla creado uno mismo, es complicado diferenciar lo que es importante o no en un mar tan grande de conceptos, programas, pruebas, etc…

Por último también creo que ha sido importante haber trabajado con tal cantidad de programas, archivos, extensiones, aplicaciones, etc… Ya que he conocido muchas formas de trabajar y programas que no había escuchado nunca, y comprenderlos, sobretodo sin una ayuda contínua de libros o manuales lo que ha supuesto todo un reto.

8 Bibliografía

[1] Fundamentos desarrollo web con JSP (Anaya Multimedia)
Jayson Falkner, Ben Galbraith, Romin Irani, Casey Kochner, Meeraj Moidoo Kunnumpurath, Sathya Narayana Panduranga, Krishnaraj Perrumal y John Timney.

[2] MySQL
http://www.mysql.com/ Septiembre 2009

[3] Tomcat
http://tomcat.apache.org/ Septiembre 2009

[4] Java
http://java.sun.com/ Octubre 2009

[5] Servlets
http://java.sun.com/products/servlet/ Octubre 2009

[6] CSS
http://www.w4c.es/divulgacion/guiasbreves/HojasEstilo Noviembre 2009
http://www.freecsstemplates.org/ Noviembre 2009

[7] SSL en Tomcat
http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=SSLenTomcat
Noviembre 2009

[8] JavaMail
http://www.desarrolloweb.com/articulos/2244.php Noviembre 2009

[9] Manejo de certificados con keytool
http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=securitySSLKeytool Noviembre 2009

[10] Conector JDBC

http://www.developer.com/java/data/article.php/4417481/Using-JDBC-with-MySQL-Getting-Started.htm Octubre 2009

http://flanagan.ugr.es/docencia/2005-2006/2/servlets/instalacion.html

Octubre 2009

ANEXO

Contenido del CD

- (CarlosRodriguezSanson)article.doc
- memoria.doc
- (CarlosRodriguezSanson)resum.doc
- (CarlosRodriguezSanson)article.pdf
- memoria.pdf
- (CarlosRodriguezSanson)resum.pdf
- jdk-6u17-windows-i586.exe
- apache-tomcat-6.0.20.exe
- mysql-essential-5.1.42-win42.exe
- BBDD.sql
- farts.war
    - Casilla.jsp
    - Casilla2.jsp
    - ConfirmacionReset.jsp
    - ConfirmacionResetN.jsp
    - Contacto.jsp
    - ErrorA.jsp
    - ErrorAlias.jsp
    - ErrorCantidad.jsp
    - ErrorPermiso.jsp
    - ErrorRec.jsp
    - ErrorVotacion.jsp
    - Estadisticas.jsp
    - FormNuevoSocio.jsp
    - General.jsp
    - GestionNoticias.jsp
    - GestionProyectos.jsp
    - InfoPersonal.jsp
    - Noticia.jsp

- Noticias.jsp
- Portada.jsp
- Proyectos.jsp
- Recuperar.jsp
- Reintegro.jsp
- style.css
- TablaVacia.jsp
- TransitoNuevaDonacion.jsp
- TransitoReintegro.jsp
- UsuarioA.jsp
- UsuarioP.jsp
- UsuarioR.jsp
- UsuarioSin.jsp
- stylePortada.css
- VistaActivados.jsp
- VistaMails.jsp
- VistaReintegros.jsp
- VistaUsuarios.jsp
- VistaUsuariosSin.jsp
- almacen
    - foto148.jpg
    - foto596.jpg
    - foto975.jpg
    - foto1159.jpg
    - foto1994.jpg
    - foto2956.jpg
    - foto2970.jpg
    - foto5596.jpg
- images
    - banner_aerosur.gif
    - banner_aula.gif
    - banner_bankinter.gif
    - banner_bio.gif

        - banner_clicair.gif
        - banner_ecoactiva.gif
        - banner_hp.gif
        - banner_kinder.gif
        - banner_lida.gif
        - banner_mediterrani.gif
        - banner_roca.gif
        - banner_talaris.gif
        - banner_vw.gif
        - img01.gif
        - img1.gif
        - img02.gif
        - img2.jpg
        - img04.gif
        - img4.gif
        - img04.gif
        - img4.gif
        - img5.gif
        - img6.gif
        - img7.gif
        - redyser.gif
        - spacer.gif
        - Thumbs.db
    - META-INF
        - MANIFEST.MF
    - WEB-INF
        - web.xml
        - lib
            - activation.jar
            - commons-fileupload-1.2.1.jar
            - commons-io-1.4.jar
            - mail.jar
            - mysql-connector-java-5.1.10-bin.jar

- classes
  - beans
    - Administradores.class
    - Donantes.class
  - connexion
    - EnviarMail.class
    - Fecha.class
    - GestionDatos.class
    - MyPasswordAuthenticator.class
    - SendMail.class
  - servlets
    - CerrarSesion.class
    - ConfirmarDon.class
    - ConfirmarPago.class
    - ConfirmarReintegro.class
    - Contacto.class
    - EliminarDon.class
    - EliminarNoticia.class
    - EliminarPago.class
    - Estadisticas.class
    - General.class
    - InfoPersonal.class
    - IngresarAdmin.class
    - IngresarDonante.class
    - IntroducirNuevaDonacion.class
    - IntroducirReintegro.class
    - IntroducirSocio.class
    - ListarActivados.class
    - ListarMails.class
    - ListarOpciones.class
    - ListarReintegros.class
    - ListarSinConfirmacion.class
    - ModificarDon.class

        - ModificarDonDef.class
        - ModificarPago.class
        - ModificarPago2.class
        - ModificarPagoDef.class
        - NuevaDonacion.class
        - NuevaNoticia.class
        - NuevoPagos.class
        - NuevoProyecto.class
        - Portada.class
        - Recuperar.class
        - Reintegro.class
        - ResetNoticias.class
        - ResetVotaciones.class
        - VerActivado.class
        - VerDonante.class
        - VerDonanteSin.class
        - VerNoticia.class
        - VerNoticias.class
        - VerReintegro.class
        - Votar.class
        - VotarProyecto.class
- src
    - beans
        - Administradores.java
        - Donantes.java
    - connexion
        - EnviarMail.java
        - Fecha.java
        - GestionDatos.java
        - MyPasswordAuthenticator.java
        - SendMail.java
    - servlets
        - CerrarSesion.java

- ConfirmarDon.java
- ConfirmarPago.java
- ConfirmarReintegro.java
- Contacto.java
- EliminarDon.java
- EliminarNoticia.java
- EliminarPago.java
- Estadisticas.java
- General.java
- InfoPersonal.java
- IngresarAdmin.java
- IngresarDonante.java
- IntroducirNuevaDonacion.java
- IntroducirReintegro.java
- IntroducirSocio.java
- ListarActivados.java
- ListarMails.java
- ListarOpciones.java
- ListarReintegros.java
- ListarSinConfirmacion.java
- ModificarDon.java
- ModificarDonDef.java
- ModificarPago.java
- ModificarPago2.java
- ModificarPagoDef.java
- NuevaDonacion.java
- NuevaNoticia.java
- NuevoPagos.java
- NuevoProyecto.java
- Portada.java
- Recuperar.java
- Reintegro.java
- ResetNoticias.java

- ResetVotaciones.java
- VerActivado.java
- VerDonante.java
- VerDonanteSin.java
- VerNoticia.java
- VerNoticias.java
- VerReintegro.java
- Votar.java
- VotarProyecto.java

ANEXO II

Índice de imágenes

Capítulo 3 **PÁG.**

Capítulo 4

Capítulo 5

Capítulo 6