

Memoria Descriptiva

Proyecto Final:

**Máster Profesional especializado en diseño y desarrollo de proyectos Web. MDI
JavaEE + Diseño Web**

Octubre 2013 - Abril 2014

[ShoppingInMadrid]

Oscar Pérez Gómez

Grupo M11

Instructor: Enrique Blasco

Fecha de entrega: 10 de Julio de 2.014

TABLE OF CONTENTS

1. DESCRIPCIÓN DEL PROYECTO	4
1.1 ¿Qué es un servidor web?	5
2. TECNOLOGÍA USADA	6
3. SISTEMA DE LOG: USO DE Log4j	8
4. BASE DE DATOS	10
4.1 Hibernate	10
4.2 Ficheros de configuracion de Hibernate	11
4.3 Esquema de las tablas	12
5. ZONA DE ADMINISTRACION	13
5.1 Seguridad	13
5.1.1 Los riesgos	13
5.2 Solución a estos riesgos	14
5.2.1 Inyección SQL	14
5.2.2 Robo de sesión	16
5.2.2.1 La Autenticación	16
5.2.2.2 La sesión	17
5.3 Panel de administracion	17
5.4 Login	19
5.4.1 Validacion de usuarios e inicio de sesión	19
5.4.2 Cierre de sesión	20
6. ZONA DE ACCESO PUBLICO	21

6.1	Sistema de cookies.....	21
6.2	Paginación.....	22
6.3	Recursos externos usados.....	25
6.3.1	Plugin Masonry	25
6.3.1.1	¿Para que uso este plugin?	25
7.	PROXIMAS MEJORAS	28
8.	GLOSARIO	29
9.	PROBLEMAS SIN RESOLVER	31
9.1	Sistema de multi-lenguaje	31
9.2	Hibernate da un error al recuperar información	31
9.3	Eliminar caché.....	32
10.	SOFTWARE	33
11.	BIBLIOGRAFÍA.....	34

Control de versiones

Autor	Versión	Fecha	Descripción
Oscar Pérez	0.5	18/05/2014	
Oscar Pérez	1.0	27/06/2014	Documentación Final de la versión 1.0 de la web

1. DESCRIPCIÓN DEL PROYECTO

Como finalización del Máster Programación y Desarrollo con Java Enterprise Edition, se requiere la realización de un Proyecto Fin de Máster en donde se debería de plasmar gran parte del temario que se ha visto a lo largo del máster.

Con este motivo voy a realizar una web de temática turística-comercial, y que tendrá dos partes:

- una parte pública donde se mostrara toda la información de la web
- una parte privada (de administración) donde se accederá a través de usuario y clave y donde se añadirá contenido a la web. ([Ver apartado 5.4.1](#))

A grandes rasgos, en la web se mostrara información comercial (tiendas, centros comerciales, grandes almacenes...) de algunas zonas de Madrid. En principio estas zonas son: Sol-Arenal-Mayor, Gran Vía, Fuencarral, Serrano-Goya. En resumen, será un listado de tiendas y está enfocada para los turistas que visitan Madrid.

Por eso, se ha realizado un sistema de multilinguaje, que en principio estará disponible en inglés y español, y el lenguaje por defecto es el inglés.

Lo ideal para este proyecto sería instalarlo en un servidor web (Apache-Tomcat) con acceso a internet. Este servidor tiene que ser capaz de interpretar el código Java que se ha desarrollado para este proyecto.

Se puede ver este proyecto desplegado en internet con los siguientes dominios:

- <http://shoppinginmadrid.info>
- <http://shoppinginmadrid.net>

Quiero comunicar que este proyecto es personal, y por tanto, todo su contenido es particular. Doy permiso a CICE para que use este proyecto como cree adecuado, siempre avisando que los derechos y el autor soy yo.

1.1 ¿QUÉ ES UN SERVIDOR WEB?

Para que un sitio web pueda funcionar correctamente se necesita de varios elementos, en primer lugar es necesario un **Servidor o Web Hosting** que no es más que una computadora con capacidad para estar conectada las 24 horas los 365 días del año a Internet, en este se alberga el sitio web el cual es descargado al equipo que solicita visualizarlo tecleando el dominio o dirección web.

Esto nos lleva al siguiente elemento para que funcione un sitio web que es el dominio, un **Dominio o Dirección Web** es un nombre en la red el cual es amigable y fácil de recordar para los humanos y que tiene una terminación en .com, .net, .org u otros. Lo que hace un dominio es que cuando es tecleado busca en los DNS (sistema de nombres de dominios) la Dirección IP a la que está asociado y redirige al servidor al que apunta el dominio para poder descargar y visualizar la página web solicitada.

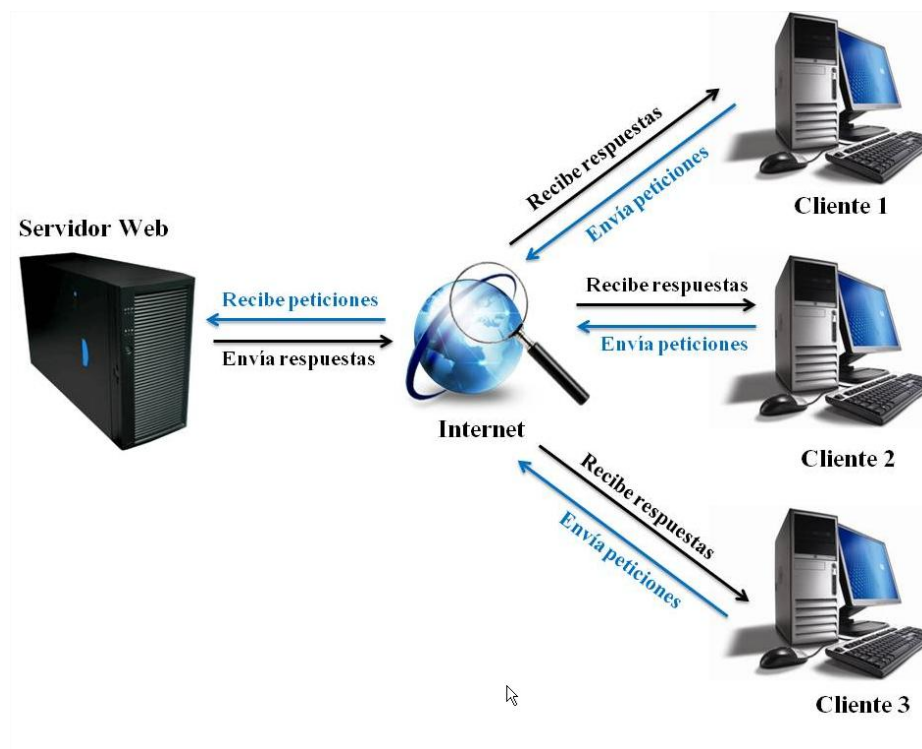


Diagrama de cómo los usuarios se conectan a un servidor web

2. TECNOLOGÍA USADA

Para la realización de esta web hemos usado las siguientes tecnologías:

1. Front-End: HTML5, CSS3, jQuery, JavaScript, AJAX
2. Back-End: Java Enterprise Edition, usando los siguientes Frameworks: Struts e Hibernate
3. Base de datos: MySql

La información está almacenada en una base de datos MySql. El acceso a esta información se realiza mediante el uso de clases Java, usando el framework Hibernate que permite un acceso a base de datos más sencillo.

Las peticiones llegan desde los usuarios que acceden a nuestro website y el servidor Apache-Tomcat es el que se encarga de devolver la información al usuario. El usuario puede pedir esta información haciendo llamadas a un jsp o a un servlet.

Todas las llamadas a la base de datos se realizan desde los servlets, y desde aquí, a través de Hibernate se accede a la base de datos. Si desde un jsp se necesita acceder a la base de datos se llama a una función definida en cualquiera de las clases programadas o se invoca a un servlet.

Uno de las pequeñas complicaciones fue el uso de AJAX para realizar peticiones de base de datos sin que el usuario se diera cuenta de que se realizaban estas peticiones.

El funcionamiento en este caso es el siguiente:

1. Se hace la llamada javascript que ejecuta la petición AJAX. Esta llamada que se realiza se define hacia un servlet
2. El servlet genera la información y la devuelve hacia el jsp, generalmente en formato JSON.
3. El jsp monta la información con lo recibido de la llamada AJAX y lo muestra al usuario.

Vamos a poner un ejemplo ya que creo que será más fácil entenderlo.

1. En el jsp está definido la llamada a la función javascript.

```
<script>
    //cambiamos el valor del input del nombre del área mediante ajax
    $('#idArea').on('change', cambiaSelectAreaByAjax);
</script>
```

La función javascript está definida en el fichero **js/base.js**

```
function cambiaSelectAreaByAjax(){
    var idArea = $("#idArea").val();
    $.ajax({
        url: '/shopMadrid/admin/getAreaAction.do',
        data: 'idArea='+idArea,
        success: modificarInputArea
    });
}
```

2. La función javascript **cambiaSelectAreaByAjax** realiza una llamada ajax a la url: **'/shopMadrid/admin/getAreaAction.do'**.

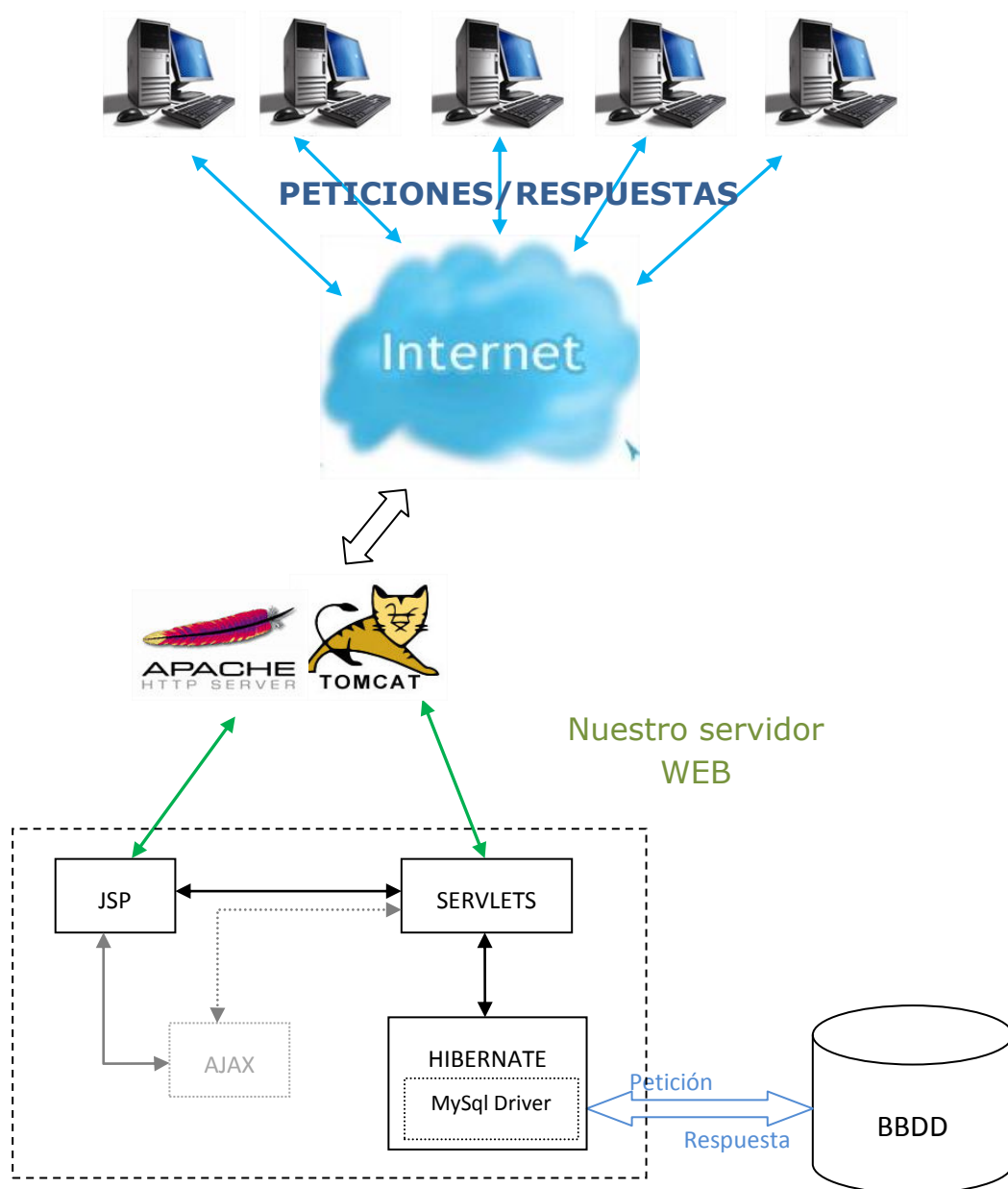
En el fichero **struts-config.xml** hemos definido esta url como:

```
<action input="/" name="Area" path="/admin/getAreaAction" scope="session"
type="com.java.shopping.madrid.servlet.actions.area.AreaGetAction"/>
```

esto lo que quiere decir es que cada vez que llámanos a esta url, se ejecuta el servlet **com.java.shopping.madrid.servlet.actions.area.AreaGetAction**.

3. Este servlet es el que se encarga de realizar la petición a la base de datos y montar la respuesta en formato JSON que llega al final al jsp.

En el siguiente esquema se explica cómo interactúan los distintos componentes y tecnologías usadas.



3. SISTEMA DE LOG: USO DE Log4j

Se está usando la biblioteca Log4j para escribir los logs o mensajes que se producen en tiempo de ejecución.

¿QUÉ ES LA BIBLIOTECA LOG4J?

Log4j es una biblioteca open source desarrollada en Java por la [Apache Software Foundation](#) que permite a los desarrolladores de software elegir la salida y el nivel de granularidad de los mensajes o “logs” (data logging) en tiempo de ejecución y no en tiempo de compilación como es comúnmente realizado. La configuración de salida y granularidad de los mensajes es realizada a tiempo de ejecución mediante el uso de archivos de configuración externos. ([Wikipedia](#))

Este API es totalmente configurable, y se puede configurar mediante un archivo XML o con un fichero en formato Java Properties (clave=valor), generalmente llamado log4j.properties.

En este proyecto se ha usado el fichero log4j.properties.

```
log4j.appender.consola=org.apache.log4j.ConsoleAppender
log4j.appender.consola.target=System.out
log4j.appender.consola.layout=org.apache.log4j.EnhancedPatternLayout
log4j.appender.consola.layout.ConversionPattern=%d{dd MMM yyyy - HH:mm:ss} [%-5p] %c{2} - %m%n

# -- se crea un appender para enviar los mensajes a un archivo
# -- creamos un archivo por cada día
log4j.appender.archivo=org.apache.log4j.DailyRollingFileAppender
log4j.appender.archivo.File=logs/shoppinginmadrid.log
log4j.appender.archivo.DatePattern='yyyy-MM-dd
log4j.appender.archivo.layout=org.apache.log4j.PatternLayout
log4j.appender.archivo.layout.ConversionPattern=%d{dd MMM yyyy - HH:mm:ss} [%-5p] %c{2} - %m%n
log4j.appender.archivo.Append=true

#-- establecemos el nivel de log e indicamos los appender que se usan.
#-- en nuestro caso se escribe en consola
log4j.rootLogger=ERROR, consola
#-- establecemos que para las clases com.java.shopping.madrid, el nivel de log es DEBUG
log4j.logger.com.java.shopping.madrid=DEBUG
```

Con esta configuración, los logs salen por consola, ya que se va a ejecutar en local.

Para la ejecución en un servidor web, he cambiado la línea **log4j.rootLogger=ERROR,consola** por **log4j.rootLogger=ERROR,archivo**, para que los logs de esta aplicación se escriban en un fichero aparte: **log4j.appender.archivo.File=logs/shoppinginmadrid.log**.

La línea de configuración **log4j.appender.archivo.DatePattern='yyyy-MM-dd** nos permite realizar la rotación de los ficheros de logs, es decir, tendremos un fichero de log por cada día.

```
root@ks35315:/var/log/tomcat7# ls -lart
total 500
drwxr-xr-x 9 root    root      4096 Jul  7 12:17 ..
-rw-r--r-- 1 tomcat7 tomcat7  17595 Jul  7 14:28 catalina.2014-07-07.log
-rw-r--r-- 1 tomcat7 tomcat7  17888 Jul  7 14:28 localhost.2014-07-07.log
-rw-r--r-- 1 tomcat7 tomcat7  37674 Jul  7 22:04 localhost_access_log.2014-07-07.txt
drwxr-x--- 2 tomcat7 adm       4096 Jul  8 11:18 .
-rw-r--r-- 1 tomcat7 tomcat7   9301 Jul  8 12:08 localhost.2014-07-08.log
-rw-r--r-- 1 tomcat7 root     304369 Jul  8 12:08 catalina.out
-rw-r--r-- 1 tomcat7 tomcat7   2330 Jul  8 12:08 catalina.2014-07-08.log
-rw-r--r-- 1 tomcat7 tomcat7  55779 Jul  8 12:08 shoppinginmadrid.log
-rw-r--r-- 1 tomcat7 tomcat7  20446 Jul  8 12:09 localhost_access_log.2014-07-08.txt
root@ks35315:/var/log/tomcat7#
```

Como se ve en la imagen anterior, a parte de los ficheros propios del tomcat (localhost, catalina.out..) se ha generado el fichero que hemos indicado **shoppinginmadrid.log**.

Para crear este archivo me he basado en varios ejemplos que he encontrado en internet:

- <http://www.javatutoriales.com/2011/04/log4j-para-creacion-de-eventos-de-log.html>
- http://www.slideshare.net/Emmerson_Miranda/log4j-1215-short-manual

4. BASE DE DATOS

El motor de base de datos utilizado es MySQL.

La conexión entre el código java (netbeans) y MySQL se realiza a través del framework Hibernate.

4.1 HIBERNATE

Hibernate es una herramienta de Mapeo objeto-relacional (ORM) para la plataforma Java (y disponible también para .Net con el nombre de NHibernate) que facilita el mapeo de atributos entre una base de datos relacional tradicional y el modelo de objetos de una aplicación, mediante archivos declarativos (XML) o anotaciones en los beans de las entidades que permiten establecer estas relaciones.

Hibernate es software libre, distribuido bajo los términos de la licencia GNU LGPL.

Como todas las herramientas de su tipo, Hibernate busca solucionar el problema de la diferencia entre los dos modelos de datos coexistentes en una aplicación: el usado en la memoria de la computadora (orientación a objetos) y el usado en las bases de datos (modelo relacional). Para lograr esto permite al desarrollador detallar cómo es su modelo de datos, qué relaciones existen y qué forma tienen. Con esta información Hibernate le permite a la aplicación manipular los datos en la base de datos operando sobre objetos, con todas las características de la POO. Hibernate convertirá los datos entre los tipos utilizados por Java y los definidos por SQL. Hibernate genera las sentencias SQL y libera al desarrollador del manejo manual de los datos que resultan de la ejecución de dichas sentencias, manteniendo la portabilidad entre todos los motores de bases de datos con un ligero incremento en el tiempo de ejecución.

Hibernate está diseñado para ser flexible en cuanto al esquema de tablas utilizado, para poder adaptarse a su uso sobre una base de datos ya existente. También tiene la funcionalidad de crear la base de datos a partir de la información disponible.

Hibernate ofrece también un lenguaje de consulta de datos llamado HQL (Hibernate Query Language), al mismo tiempo que una API para construir las consultas programáticamente (conocida como "criteria").

Básicamente el desarrollador deberá configurar en un archivo XML o mediante annotations donde corresponde un atributo de una clase, con una columna de una tabla. En este proyecto he usado ficheros de configuración xml.

4.2 FICHEROS DE CONFIGURACION DE HIBERNATE

Los ficheros de configuración de Hibernate son los siguientes:

1. **src/java/hibernate.cfg.xml**: En el que ponemos información sobre la base de datos que estamos manejando e incluimos la referencia a los ficheros de mapeo (los *.hbm.xml)

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
  <session-factory>
    <property name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property>
    <property name="hibernate.connection.driver_class">com.mysql.jdbc.Driver</property>
    <property
name="hibernate.connection.url">jdbc:mysql://localhost:3306/shoppingmadrid?zeroDateTimeBehavior
=convertToNull</property>
    <property name="hibernate.connection.username">XXXXXX</property>
    <property name="hibernate.connection.password">XXXXXX</property>
    <mapping resource="com/java/shopping/madrid/models/Markets.hbm.xml"/>
    <mapping resource="com/java/shopping/madrid/models/DepartmentStore.hbm.xml"/>
    <mapping resource="com/java/shopping/madrid/models/Shops.hbm.xml"/>
    <mapping resource="com/java/shopping/madrid/models/Areas.hbm.xml"/>
    <mapping resource="com/java/shopping/madrid/models/Mall.hbm.xml"/>
    <mapping resource="com/java/shopping/madrid/models/Login.hbm.xml"/>
    <mapping resource="com/java/shopping/madrid/models/Stores.hbm.xml"/>
    <mapping resource="com/java/shopping/madrid/models/Streets.hbm.xml"/>
    <mapping resource="com/java/shopping/madrid/models/TypeShop.hbm.xml"/>
  </session-factory>
</hibernate-configuration>
```

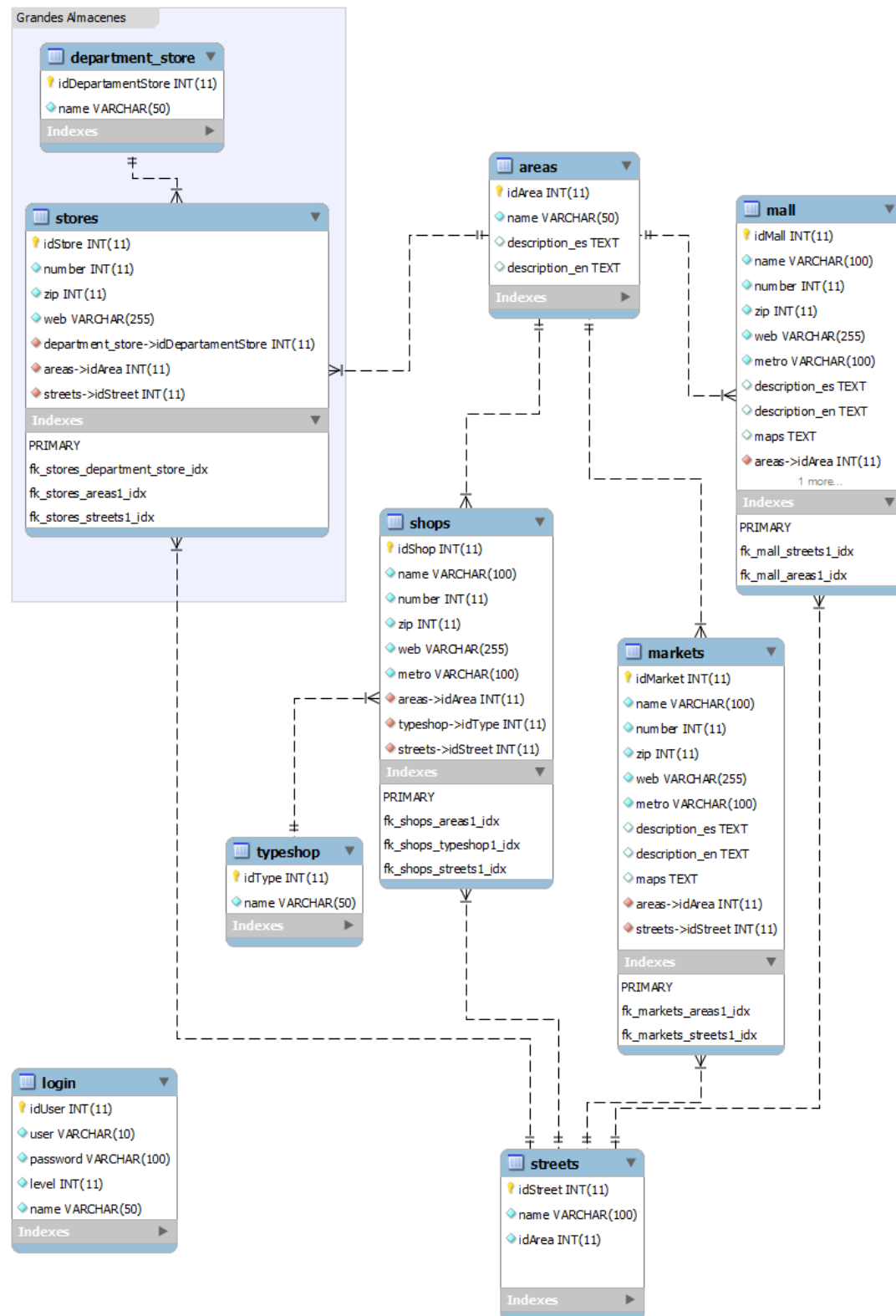
Ejemplo de fichero de configuración: hibernate.cfg.xml

2. **src/java/hibernate.reveng.xml**: Donde se definen las tablas que vamos a usar

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-reverse-engineering PUBLIC "-//Hibernate/Hibernate Reverse Engineering
DTD 3.0//EN" "http://hibernate.sourceforge.net/hibernate-reverse-engineering-3.0.dtd">
<hibernate-reverse-engineering>
  <schema-selection match-catalog="shoppingmadrid"/>
  <table-filter match-name="streets"/>
  <table-filter match-name="departmentstore"/>
  <table-filter match-name="areas"/>
  <table-filter match-name="stores"/>
  <table-filter match-name="markets"/>
  <table-filter match-name="login"/>
  <table-filter match-name="typeshop"/>
  <table-filter match-name="mall"/>
  <table-filter match-name="shops"/>
</hibernate-reverse-engineering>
```

Ejemplo de fichero de configuración: hibernate.reveng.xml

4.3 ESQUEMA DE LAS TABLAS



5. ZONA DE ADMINISTRACION

Antes de entrar a explicar el funcionamiento de la parte de administración quisiera dejar explicados algunos temas que hay que tener en cuenta cuando nos enfrentamos al desarrollo de un proyecto que lleva implícito almacenamiento de información, ya sea en bases de datos u en otros mecanismos.

5.1 SEGURIDAD

Antes de nada vamos a dar una guía de buenas prácticas que es interesante conocer antes de empezar a construir aplicaciones para conseguir un trabajo lo más bueno posible.

- **No dar nunca nada por hecho** ni en cuestiones de seguridad ni en cuestiones del flujo normal de la aplicación. Todo el riesgo que se corra debe ser por parte del usuario (no hay nada que hacer contra eso). Me explico, no supongamos que si le pido el nombre al usuario no me va a poner un número de teléfono. No sé si sería una forma acertada de decirlo pero hay que pensar con pesimismo, en los peores casos, y por remotos que sean pueden ocurrir.
- **La oscuridad no es seguridad.** No poner un botón de acceso a la administración no impide que se pueda acceder a ella. Ocultar nuestro código no debe ser parte de nuestra seguridad.
- **Principio del mínimo privilegio:** El usuario del sistema debe tener únicamente los privilegios que necesita para llevar a cabo su actividad.
- **Fallar de manera segura:** Hay que tratar de manera muy cuidadosa los fallos en la aplicación. Por poner un ejemplo, si se produce un fallo en la aplicación mientras se realizan tareas administrativas no debe seguir iniciada la sesión como administrador. Otro ejemplo, no debe mostrar en un fallo información técnica sobre el mismo al usuario del sistema. Si el usuario sabe datos acerca de nuestro sistema podría tener más fácil la búsqueda de vulnerabilidades.

5.1.1 LOS RIESGOS

Ahora que hemos dado unos pequeños consejos sobre seguridad a la hora de la construcción de aplicaciones ya podemos pasar a explicar los riesgos que hay en las aplicaciones web. No vamos a ponernos a programar nada sin saber los riesgos existentes y el porque de las medidas que se toman para combatirlos.

Voy a nombrar aquí los riesgos que considero más importantes en nuestra aplicación.

- **Inyección SQL:** Consiste en intentar “engañar” al sistema para que realice peticiones contra la base de datos que no son las que han sido programadas y que podrían comprometer gravemente la base de datos o incluso mostrar al atacante toda la información de la misma.

- **Robo de sesión:** Como sabemos HTTP es un protocolo sin estados, lo que significa que las credenciales o información de sesión deberá ir en cada petición; debido a esto dichos datos resultan muy expuestos. Un robo de estos datos podría tener como resultado que alguien se estuviera haciendo pasar por nosotros y realizando acciones con unos privilegios que nos pertenecen. Y tampoco hemos de olvidar que se puede robar la sesión intentando obtener nuestros credenciales de alguna manera (averiguar nuestra contraseña).
- **Acceso a URLs restringidas:** Consiste en la observación de una URL e intentar cambiarla para intentar acceder a otras zonas. Esta es una de las razones por las que la seguridad a través de la ocultación no es efectiva.

5.2 SOLUCIÓN A ESTOS RIESGOS

Ahora que ya hemos identificado y visto en qué consisten los problemas que nos podemos encontrar en las aplicaciones web, o al menos lo más importantes y peligrosos vamos a ir uno por uno explicando cómo los he solucionado.

Lo primero ha sido no implementar un registro online de usuarios en la base de datos. Los usuarios se han guardado directamente en la base de datos.

En este panel de administración, hay dos tipos de usuarios:

- **Administrador: Level 1.** Este usuario puede visualizar/modificar/insertar/borrar cualquier registro de la base de datos
- **Invitado: Level 2.** Este usuario, únicamente puede visualizar la información de la base de datos. En todas las operaciones que haga se le mostrará el mensaje de "Operación realizada correctamente" pero no tendrá efecto sobre la base de datos.

5.2.1 INYECCIÓN SQL

Comenzaremos hablando por el primero de los riesgos de seguridad que comentaba anteriormente. Para hacer un breve recordatorio veíamos que se trataba de intentar comprometer de alguna manera la base de datos de la aplicación.

Para ello, se suelen usar los formularios de las páginas web tratando de cerrar en ellos una consulta SQL e iniciando otra a continuación; dicho esto enviar directamente los datos de los formularios hacia una consulta SQL no es una buena práctica. Veremos en este apartado una serie de buenas prácticas que deberíamos llevar a cabo para evitar la inyección SQL.

Uso de PreparedStatements

En primer lugar una muy buena práctica es cambiar el uso de los Statements por los PreparedStatement. Este tipo de Statement obliga a que la consulta tenga exactamente la estructura que se indica, y no deja que se cambie la estructura de la misma o se cierre una

consulta y se abra otra nueva. También lleva a cabo un escapado de caracteres, se puede poner el texto "O'Donnell" y no habría problema con la comilla simple.

Para hacer una consulta con un PreparedStatement lo primero que se hace es indicar la estructura de la consulta dejando los parámetros marcados con una interrogación "?" y posteriormente se irá indicando que son cada una de esas interrogaciones y su tipo de dato. Y gracias a que se indica el tipo de dato, Java se encarga de poner las comillas que hagan falta. Existen muchos métodos *set* dependiendo del tipo de dato a introducir.

Ejemplo:

```
String sqlInsert = "INSERT INTO AREAS (name) VALUES(?)";
con = mySql.conectarBD();
//Preparamos la sentencia
PreparedStatement pst = con.prepareStatement(sqlInsert);
//Asignamos un parametro a la sentencia Sql
pst.setString(1, name);
//Ejecutamos la sentencia y cerramos el PreparedStatement
pst.execute();
pst.close();
```

También usamos las queries pre-definidas en los ficheros **.hbm.xml** que se crean con el framework Hibernate.

```
<query name="loginUsuario">
    from Login lg where lg.user=:user and lg.password=:password
</query>
```

Definición de la query en uno de los fichero de mapeo .hbm.xml que usa Hibernate

Acceso a la query desde un FormAction,

```
//Conexion con Hibernate y base de datos
SessionFactory sf = HibernateUtil.getSessionFactory();
Session s = sf.openSession();
//Creamos la transición
Transaction tx = s.beginTransaction();
/* comprobamos si la informacion del login es correcta
   usamos una query definida en Usuarios.hbm.xml */
Query q = s.getNamedQuery("loginUsuario");
q.setParameter("user", userForm.getUser());
q.setParameter("password", passMD5);
//usamos .uniqueResult(), aunque tambien podriamos usar .list()
Login result = (Login)q.uniqueResult();
```

El caso es que, sea cual sea el método que usemos, evitaremos la inyección SQL.

5.2.2 ROBO DE SESIÓN

Ya hemos visto el riesgo que tiene el robo de una sesión. Podríamos decir de manera resumida que el peligro está en la exposición de los datos de sesión.

Para solucionar este problema de seguridad hay que atenerse a varios aspectos de la seguridad: la autenticación y la sesión; para cada uno de ellos veremos varios aspectos importantes a cubrir para solucionar problemas con el robo de sesión.

5.2.2.1 LA AUTENTICACIÓN

- El más importante de todos. Usar SSL sobre HTTP (HTTPS) para transferir los datos y asegurarse de que el cifrado cubre los credenciales y el ID de sesión en todas las comunicaciones. De esta manera los datos de sesión de los que hablábamos siguen estando expuestos pero esta vez se encuentran cifrados, por lo que no se pueden usar. Alguien podría pensar: “¿Y si se obtiene la sesión y se rompe la encriptación?”, la respuesta es sencilla, y es que con los medios actuales para cuando hayas conseguido romper la encriptación esa sesión habrá dejado de existir. *(Nota: En este proyecto no he visto la necesidad de usar un servidor seguro, pero siempre hay que tenerlo en cuenta dependiendo del tipo de proyecto y la información que se va a tratar)*
- Posibilitar el bloqueo de autenticación después de un número determinado de intentos fallidos. Esto podría evitar ataques de fuerza bruta intentando averiguar la contraseña del usuario. *(Nota: Tampoco he implementado éste código pero podría ser algo tan fácil como añadir a la sesión el número de intentos fallidos y en el caso de que superen un determinado número, no ejecutar ningún mecanismo de autenticación ni mostrar el formulario de login, únicamente habría que lanzar un timer para desbloquear el inicio de sesión pasado un tiempo).*
- Implementar métodos seguros de recuperación de contraseñas: Es común que se intenten usar estos métodos para intentar ganar acceso a una cuenta del usuario, podemos ver una serie de consejos para implementar estos métodos. Pedir al usuario al menos tres datos o más, obligar a que responda preguntas de seguridad, el uso de un CAPTCHA. La contraseña recuperada deberá generarse de manera aleatoria (con una longitud de al menos 8 caracteres) y enviada al usuario por un canal diferente (E-mail); de esta forma si el atacante consiguió sortear los primeros pasos es difícil que logre sortear el canal usado para transmitir. *(Nota: Tampoco he implementado esta funcionalidad de recuperación de la clave, ya que no se prevé que sea un sistema con pocos usuarios de administración. Lo más seguro es que únicamente hay un usuario que acceda al Panel de Control).*

- Usar un sistema de autenticación simple, centralizado y estandarizado.
- a) El **LoginAction** (com.java.shopping.servlets.admin.actions.login) guarda en sesión los datos del usuario:

```
//creamos la sesión y guardamos los datos del Login  
request.getSession().setAttribute("userLogin", result);  
request.getSession().setAttribute("auth", true);
```

- b) He creado un control.jsp que se incluye en todos los jsp y lo que hace es comprobar que en sesión existe el usuario:

```
//recuperamos el userLogin de la sesión. Se ha guardado en LoginAction  
Login login = (Login)request.getSession().getAttribute("userLogin");  
userName = login.getName();
```

5.2.2.2 LA SESIÓN

Usar los métodos de sesión que nos proporciona el servidor de aplicaciones que estemos usando, y digo esto por las mismas razones que aconsejé usar los métodos de autenticación que proporciona el servidor de aplicaciones. En este caso no estamos refiriendo a la sesión y a las cookies.

- Asegurar que la operación de cierre de sesión realmente destruye dicha sesión. También fijar el periodo de expiración de la sesión); por ejemplo para aplicaciones críticas de 2 a 5 minutos, mientras que para otras aplicaciones más comunes se podría usar de 15 a 30 minutos.

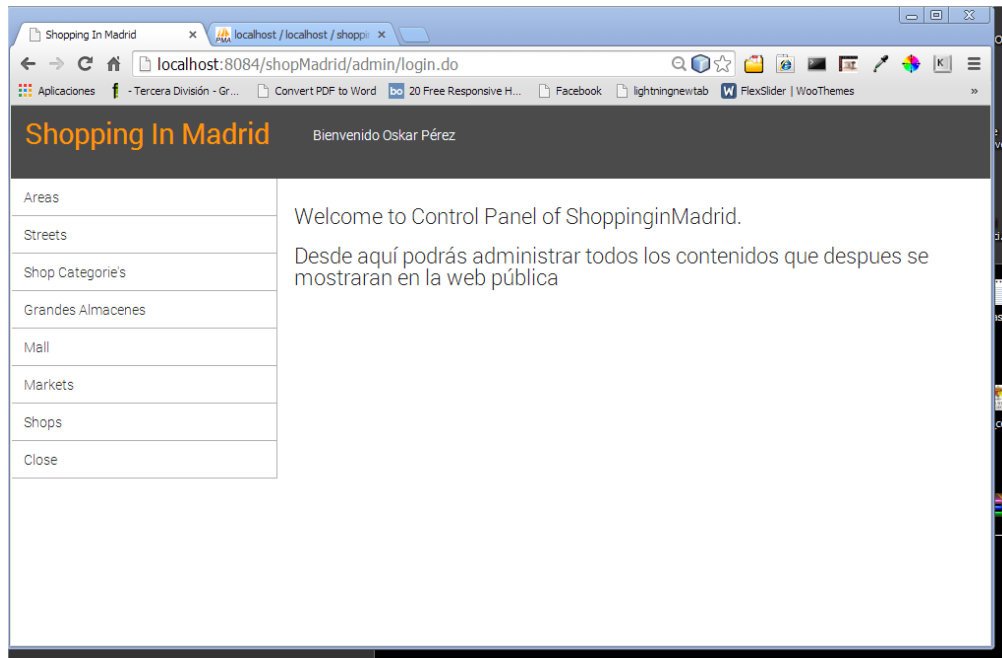
En el descriptor de despliegue (WEB-INF/web.xml) podemos fijar la caducidad de la sesión en minutos.

```
<session-config>  
  <session-timeout>15</session-timeout>  
</session-config>
```

5.3 PANEL DE ADMINISTRACION

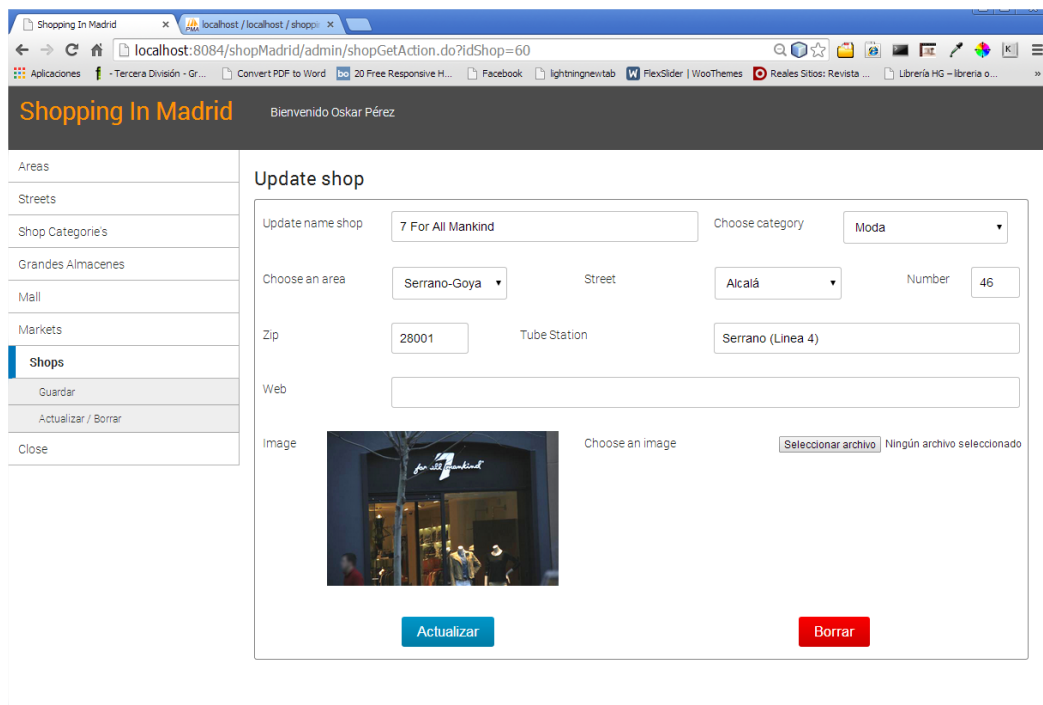
Lo primero que quiero indicar es que el panel no tiene un estilo responsive adaptative. No es adaptativo al tamaño del dispositivo. **Para usar este panel se recomienda usar un ordenador (no está adaptado a móviles y tablets).**

El panel de administración tiene 7 apartados (Áreas, Calles, Categorías de las tiendas, Grandes Almacenes, Centros Comerciales, Mercados, y Tiendas) para poder interactuar y añadir/modificar/borrar información en cada uno de los apartados.



Página de entrada del Panel de Administración

El siguiente pantallazo es un ejemplo de uno de las operaciones que se pueden realizar en este panel de administración.



5.4 LOGIN

La primera página de la zona de administración es un login, que nos pide un usuario y una password para poder acceder al panel de administración.

Esta información se valida contra la tabla login de nuestra base de datos.

La password se almacena en la base de datos cifrada. **Se encripta con el algoritmo MD5.**



Fig. 4.4.1. Formulario de login del Panel de Administración

	idUser	user	password	level	name
<input type="checkbox"/>	2	usuario1	122b738600a0f74f7c331c0ef59bc34c	1	Usuario de test
<input type="checkbox"/>	3	admin	21232f297a57a5a743894a0e4a801fc3	0	Administrador

Tabla login donde se muestra las claves cifradas

5.4.1 VALIDACION DE USUARIOS E INICIO DE SESIÓN

Para acceder a la zona de administración debe de usar esta información:

- Acceso Total: (sólo en el proyecto en local. En el website desplegado en internet este usuario no existe)
 - **Usuario: admin**
 - **clave: admin**
- Acceso Lectura (tanto en local como en internet)
 - **Usuario: usuario1**
 - **clave: usuario1**

Se comprueba que los datos introducidos en el formulario de entrada (Fig. 4.4.1.) son correctos.

La validación se realiza contra la tabla Login de la base de datos y se comprueba si existe el usuario y la password que se ha rellenado en el formulario. Para esta comprobación usamos una query definida en el fichero **Login.hbm.xml**

```
<query name="loginUsuario">
    from Login lg where lg.user=:user and lg.password=:password
</query>
```

Si el usuario esta registrado se crea la sesión y guardamos los datos del Login en ella.

```
request.getSession().setAttribute("userLogin", result);  
request.getSession().setAttribute("auth", true);
```

5.4.2 CIERRE DE SESIÓN

Para el cierre de sesión, lo que se hace es borrar de la sesión las dos variables que se han almacenado cuando se ha creado la sesión

```
request.getSession().setAttribute("userLogin", null);  
request.getSession().setAttribute("auth", false);
```

6. ZONA DE ACCESO PUBLICO

Esta zona es la web pública, la que muestra a los usuarios toda la información que existe en la base de datos.

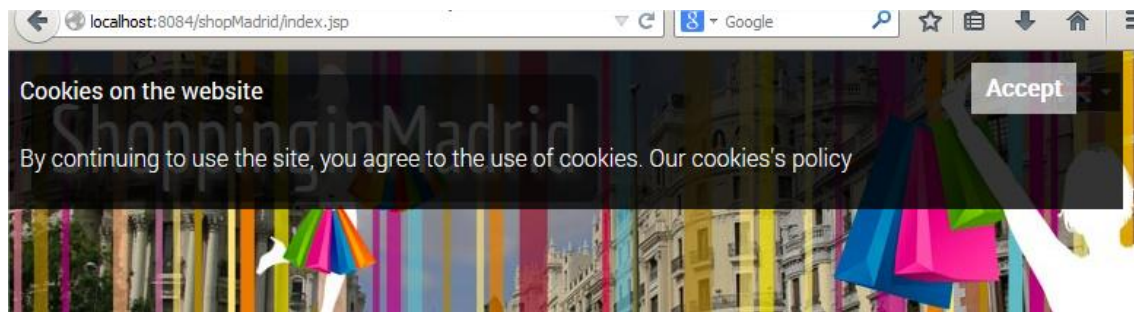
El diseño es responsive adaptative, lo que quiere decir que se ve correctamente en los diferentes dispositivos que usemos para acceder a ella, ya sean ordenadores, tablets, o teléfonos móviles.

6.1 SISTEMA DE COOKIES

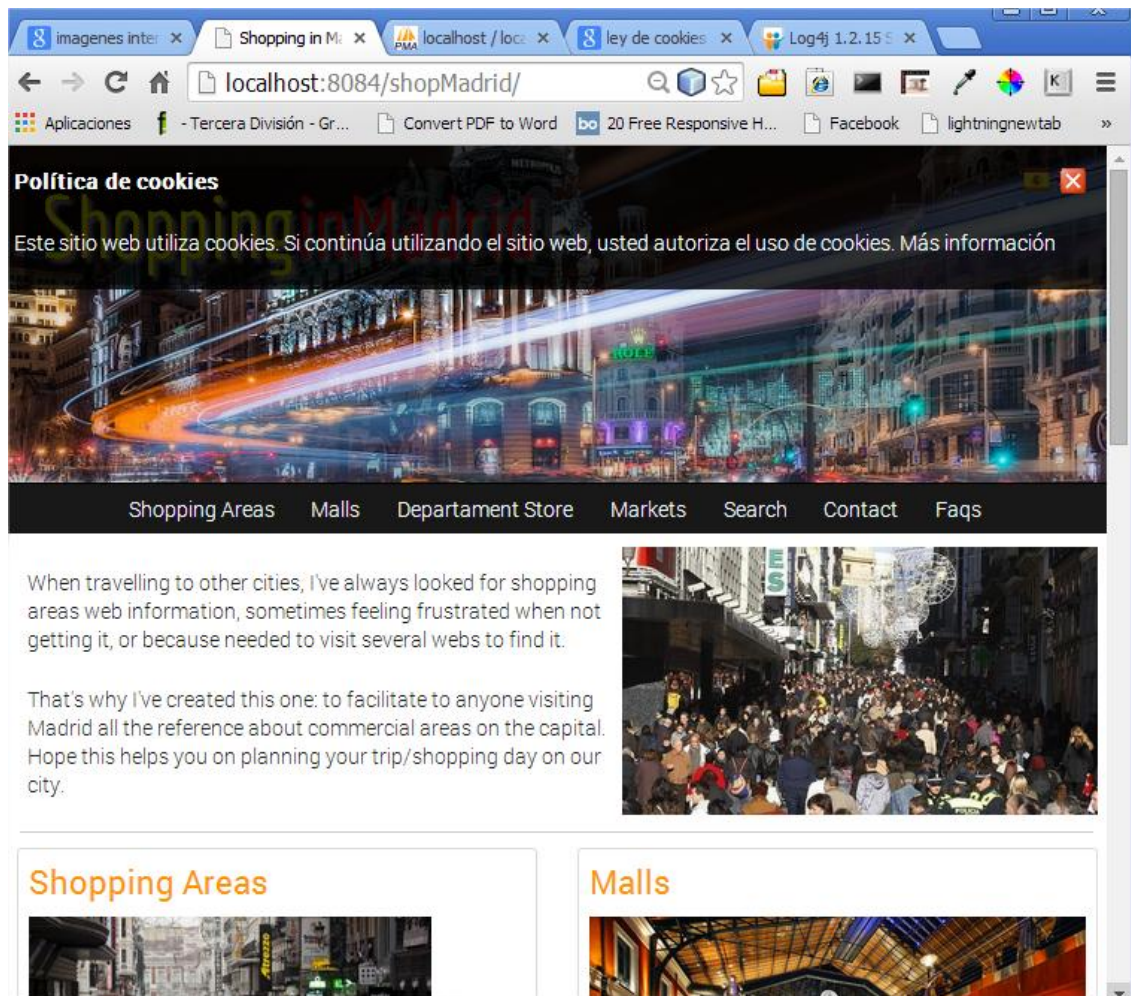
Se ha implementado un sistema de cookies, que se usa para avisar al usuario que se pueden instalar cookies en su ordenador si sigue navegando por nuestro website.

El aviso está situado en la parte superior de la web, y hasta que no se hace clic en el botón "Accept" no se oculta.

La información de las cookies que se muestra es la siguiente:



Nuestro sistema únicamente crea una cookie: ***cookies_mostrar_velo*** que la usamos para mostrar al usuario el mensaje de la Política de cookies. Esta cookie que creamos tiene una vida de unos 15 minutos.



Página index donde se muestra el "velo" de las cookies.

Todas las demás cookies que se puedan añadir, serán las de Google Analytics, Google Maps o proveedores externos de contenidos (Youtube, Flick...)

En esta primera versión lo único que se usa externo son los mapas de Google, y se usará también el sistema de estadísticas de Google Analytics.

6.2 PAGINACIÓN

He desarrollado un sistema de paginación para mostrar el listado de las tiendas. Esta paginación se muestra en los apartados "Shopping Areas" y "Search".

Para ello hay que calcular el número de páginas que tenemos que mostrar. La forma en que calculamos este número es la siguiente:

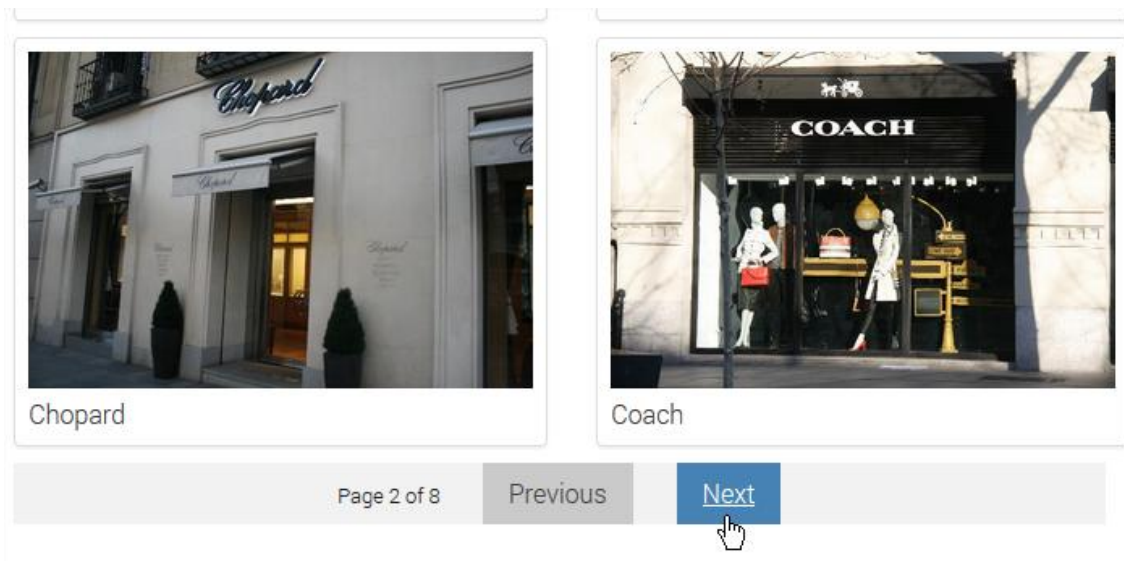
1. Generamos una sentencia SQL para realizar un count(*)
2. Recuperamos con el count(*) el número total de registros

3. Dividimos por el número de objetos que se muestran en la página (esta información está definida en una constante) para saber el número de páginas.

El código java que realiza esta operación es el siguiente:

```
//se calcula el número de páginas
String s_query = "SELECT count(*) FROM Shops" + condicional;
long numTotalObjetos = (Long) s.createQuery(s_query).uniqueResult();
int numPaginas =(int) Math.ceil(((double)numTotalObjetos / (double)Constants.TAMANYO_PAGINA);
```

En la siguiente imagen se muestra la navegación de la paginación de las tiendas.



Hibernate nos permite realizar la paginación ya que dispone de algunas características muy útiles.

Las que hemos usado son las siguientes:

- `setMaxResults`: devuelve un número máximo de resultados
- `setFirstResult`: se define el primer objeto que va a devolver

```
//Indicamos que sólo se devuelvan como máximo tantos objetos como tamaño tiene la página.
q.setMaxResults(Constants.TAMANYO_PAGINA);
//Indicamos cuál es el primer objeto a retornar.
q.setFirstResult( (paginaAMostrar-1) * Constants.TAMANYO_PAGINA);
```

Se puede ver el código completo de la paginación en:

- `com.java.shopping.madrid.servlet.actions.shop.ShopListPublicAction`
- `com.java.shopping.madrid.servlet.actions.shop.ShopSearchAction`

El esquema del funcionamiento de la paginación sería el siguiente:

Los enlaces "Previous" y "Next" hacen una llamada a una función javascript que realiza una petición AJAX para recuperar los datos. En esta llamada hay que pasarle el número de página que queremos recuperar. La llamada es: `goPage(numPage)`

```
function goPage(num){
    $("#numPage").val(num);
    loadShopsPublicByAjax();
}
```

Esta función modifica el valor de un campo oculto donde guardamos el número de la página que queremos mostrar y llama a la función: `loadShopsPublicByAjax()`

Es la función `loadShopsPublicByAjax()` quien realiza la petición AJAX, pasando como parámetros el `idStreet`, `idType` y el `numPage`

```
/* Recuperamos la información de las tiendas */
function loadShopsPublicByAjax(){
    var parametros = {
        "idStreet" : $("#idStreet").val(),
        "idType" : $("#idType").val(),
        "numPage" : $("#numPage").val()
    };
    $.ajax({
        url: '/shopMadrid/shopListPublicAction.do',
        data: parametros,
        type: 'post',
        beforeSend: function () {
            $(".listado").html("<div class='message message-info'>Processing, wait please...</div>");
        },
        success: loadListShops
    });
}

function loadListShops(data){
    var respuesta = data.trim();
    $(".listado").html(respuesta);
    loadMasonry();
}
```

6.3 RECURSOS EXTERNOS USADOS

6.3.1 PLUGIN MASONRY

¿QUÉ ES EL PLUGIN MASONRY.JS?

Esto es lo que indica su página [web](#)

WHAT IS MASONRY?

Masonry is a JavaScript grid layout library. It works by placing elements in optimal position based on available vertical space, sort of like a mason fitting stones in a wall.

Masonry es una biblioteca para implementar un grid en forma de cuadrícula con Javascript. Su acción consiste en la colocación de los elementos en posición óptima sobre la base de espacio vertical disponible, algo así como un albañil cuando coloca una pared.

LICENCIA: Masonry se distribuye bajo [licencia MIT](#)

6.3.1.1 ¿PARA QUE USO ESTE PLUGIN?

Este plugin lo uso en las páginas donde se muestran "celdas", como en el `index.jps`. Y sirve para redimensionar y colocar estas celdas cuando el navegador se redimensiona o se ve la web en dispositivos móviles con una pantalla menor

Shopping Areas



Those are, on our opinion, best shopping places on Madrid.

Malls



Shopping malls accessible by tube.

Department Store



El Corte Inglés

El Corte Inglés S.A., headquartered in Madrid, is the biggest department store group in Europe and ranks fourth worldwide. El Corte Inglés is Spain's only remaining department store chain. You will find a list of department stores in Madrid accessible by tube.

El Rastro



Madrid's El Rastro is the oldest open market. Born around 1740 in the surroundings of Village's slaughterhouse, dedicated overall to second hand items, it takes place every Sunday and festive in a typical neighborhood on the historical center of the capital.

Madrid Municipal Markets



In recent years, a renovation is being developed in historic markets. Those galleries where a closest relationship was established between traders and customers and treatment was far more familiar. Nowadays those spaces are recovering and giving them new usages. This is our route for the most famous.

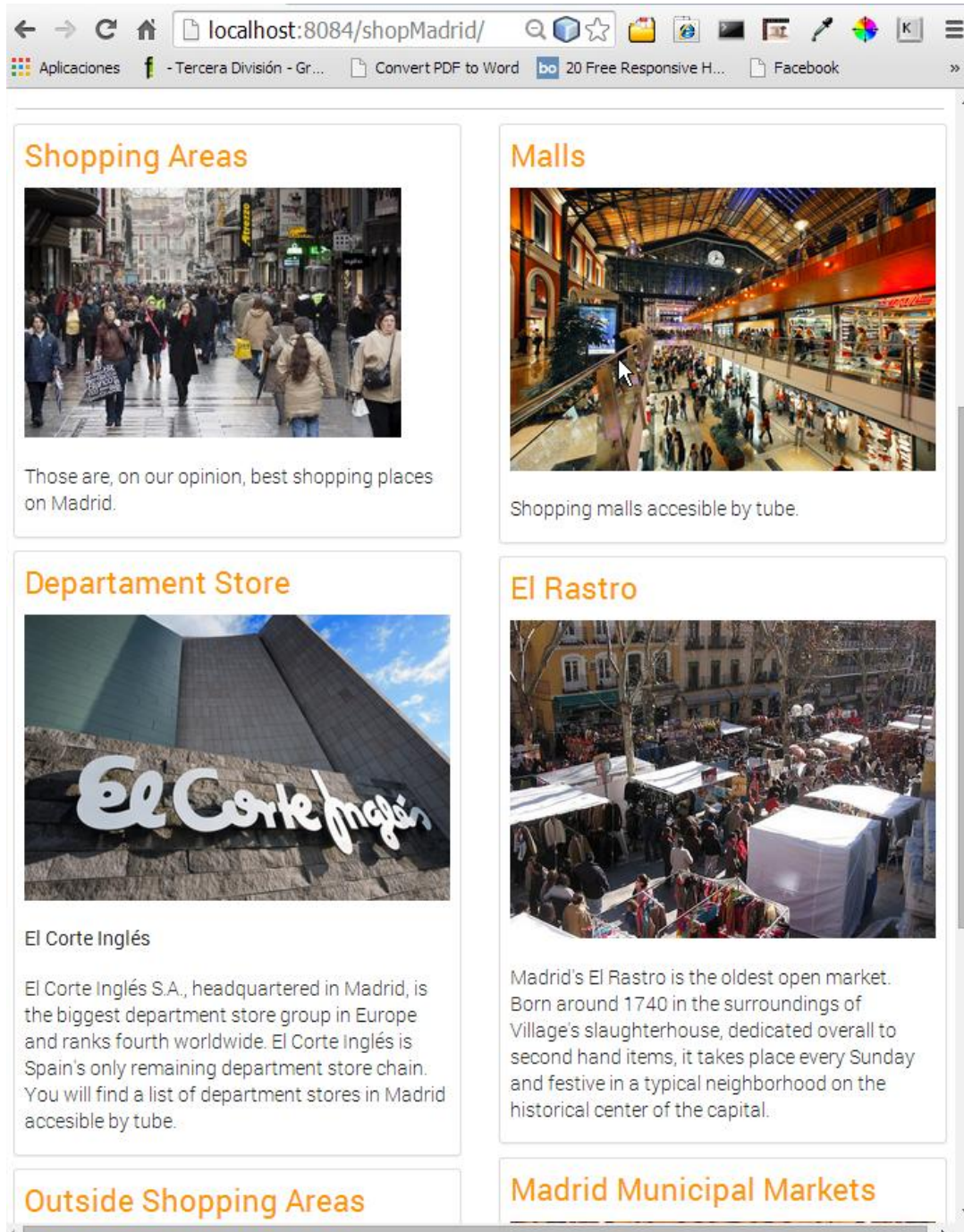
Outside Shopping Areas



Besides you can find other shopping areas outside Madrid all of them can be reached by bus, taxi or public transportation.

Website visto en un ordenador con un ancho de pantalla de 1280px

Si reducimos el tamaño del navegador o se abre el website desde un dispositivo móvil se puede apreciar en la imagen, cómo las "celdas" se han alineado y ordenado en relación al nuevo tamaño del navegador.



Website visto en un ordenador con un ancho de pantalla de 768px

Las siguientes imágenes muestra la web vista en un Smartphone. Se puede comprobar que la información se alinea en una columna automáticamente.

ShoppingInMadrid



When travelling to other cities, I've always looked for shopping areas web information, sometimes feeling frustrated when not getting it, or because needed to visit several webs to find it.


That's why I've created this one: to facilitate to anyone visiting Madrid all the reference about commercial areas on the capital. Hope this helps you on planning your trip/shopping day on our city.




Shopping Areas




For All Mankind



Adolfo Dominguez



Agatha Ruiz de la Prada



Esta funcionalidad que parece tan fácil de realizar con CSS (tan sólo son unos div con float:left) no queda tan bien en la práctica, ya que los elementos no se ajustan y quedan muchos huecos vacíos.

Todo esto lo hace automáticamente este plugin, y por su facilidad de uso y el tamaño que tiene el plugin recomendando el uso de masonry para este tipo de grid.

7. PROXIMAS MEJORAS

1. Quedan por desarrollar dos apartados: Departament Store y Outside Shopping Areas.
2. Una aplicación móvil usando geolocalización.
3. Mejoras según el feed-back de los usuarios

8. GLOSARIO

Hibernate es una herramienta de Mapeo objeto-relacional (ORM) para la plataforma Java (y disponible también para .Net con el nombre de Hibernate) que facilita el mapeo de atributos entre una base de datos relacional tradicional y el modelo de objetos de una aplicación, mediante archivos declarativos (XML) o anotaciones en los beans de las entidades que permiten establecer estas relaciones. Hibernate es software libre, distribuido bajo los términos de la licencia GNU LGPL.

Struts es una herramienta de soporte para el desarrollo de aplicaciones Web bajo el patrón MVC bajo la plataforma Java EE (Java Enterprise Edition).

MySQL es un sistema de gestión de bases de datos relacional, multihilo y multiusuario desarrollado como software libre en un esquema de licenciamiento dual.

HTML5 (HyperText Markup Language, versión 5) es la quinta revisión importante del lenguaje básico de la World Wide Web, HTML. HTML5 especifica dos variantes de sintaxis para HTML: un «clásico» HTML (text/html), la variante conocida como HTML5 y una variante XHTML conocida como sintaxis XHTML5 que deberá ser servida como XML.1 2 Esta es la primera vez que HTML y XHTML se han desarrollado en paralelo.

Al no ser reconocido en viejas versiones de navegadores por sus nuevas etiquetas, se recomienda al usuario común actualizar a la versión más nueva, para poder disfrutar de todo el potencial que provee HTML5.

JavaScript es un lenguaje de programación interpretado, dialecto del estándar ECMAScript. Se define como orientado a objetos,³ basado en prototipos, imperativo, débilmente tipado y dinámico.

Se utiliza principalmente en su forma del lado del cliente (client-side), implementado como parte de un navegador web permitiendo mejoras en la interfaz de usuario y páginas web dinámicas⁴ aunque existe una forma de JavaScript del lado del servidor (Server-side JavaScript o SSJS). Su uso en aplicaciones externas a la web, por ejemplo en documentos PDF, aplicaciones de escritorio (mayoritariamente widgets) es también significativo.

Javascript se diseñó con una sintaxis similar al C, aunque adopta nombres y convenciones del lenguaje de programación Java. Sin embargo Java y JavaScript no están relacionados y tienen semánticas y propósitos diferentes.

Todos los navegadores modernos interpretan el código JavaScript integrado en las páginas web. Para interactuar con una página web se provee al lenguaje JavaScript de una implementación del Document Object Model (DOM).

Tradicionalmente se venía utilizando en páginas web HTML para realizar operaciones y únicamente en el marco de la aplicación cliente, sin acceso a funciones del servidor. JavaScript se interpreta en el agente de usuario, al mismo tiempo que las sentencias van descargándose junto con el código HTML.

jQuery es una biblioteca de JavaScript, creada inicialmente por John Resig, que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web. JQuery es la biblioteca de JavaScript más utilizada. jQuery es software libre y de código abierto, posee un doble licenciamiento bajo la Licencia MIT y la Licencia Pública General de GNU v2, permitiendo su uso en proyectos libres y privativos. jQuery, al igual que otras bibliotecas, ofrece una serie de funcionalidades basadas en JavaScript que de otra manera requerirían de mucho más código, es decir, con las funciones propias de esta biblioteca se logran grandes resultados en menos tiempo y espacio.

Hojas de Estilo en Cascada (Cascading Style Sheets - CSS) es el lenguaje de hojas de estilo utilizado para describir el aspecto y el formato de un documento escrito en un lenguaje de marcas, esto incluye varios lenguajes basados en XML como son XHTML o SVG.

AJAX, acrónimo de Asynchronous JavaScript And XML (JavaScript asíncrono y XML), es una técnica de desarrollo web para crear aplicaciones interactivas o RIA (Rich Internet Applications). Estas aplicaciones se ejecutan en el cliente, es decir, en el navegador de los usuarios mientras se mantiene la comunicación asíncrona con el servidor en segundo plano. De esta forma es posible realizar cambios sobre las páginas sin necesidad de recargarlas, mejorando la interactividad, velocidad y usabilidad en las aplicaciones.

Ajax es una tecnología asíncrona, en el sentido de que los datos adicionales se solicitan al servidor y se cargan en segundo plano sin interferir con la visualización ni el comportamiento de la página. JavaScript es el lenguaje interpretado (scripting language) en el que normalmente se efectúan las funciones de llamada de Ajax mientras que el acceso a los datos se realiza mediante XMLHttpRequest, objeto disponible en los navegadores actuales. En cualquier caso, no es necesario que el contenido asíncrono esté formateado en XML.

Ajax es una técnica válida para múltiples plataformas y utilizable en muchos sistemas operativos y navegadores, dado que está basado en estándares abiertos como JavaScript y Document Object Model (DOM).

9. PROBLEMAS SIN RESOLVER

9.1 SISTEMA DE MULTI-LENGUAJE

Cuando recupero información de la tabla TypeShop

```
CREATE TABLE IF NOT EXISTS `typeshop` (  
  `idType` int(11) NOT NULL AUTO_INCREMENT,  
  `name_es` varchar(50) COLLATE utf8_spanish_ci NOT NULL,  
  `name_en` varchar(50) COLLATE utf8_spanish_ci NOT NULL,  
  PRIMARY KEY (`idType`)  
)
```

dependiendo del lenguaje que está seleccionado en la web, tengo que mostrar el nombre en inglés o español. La función que recuperar la información y crea un select-combo es:

```
public String buildSelectTypeShop(List<TypeShop> listTypeShops, String idioma) {  
    String selectTypeShops = "";  
    String name = "";  
    for (TypeShop a: listTypeShops){  
        if (idioma.equals("en")){ name = a.getName_en(); }  
        else{ name = a.getName_es(); }  
        selectTypeShops += "<option value="+ a.getIdType()+">" + name + "</option>";  
    }  
    return selectTypeShops;  
}
```

¿Como podría recuperar la información sin el if?. En este caso como solo tengo dos lenguajes no hay problema, pero si tuviera varios lenguajes, crear un if para cada uno me parece poco "correcto". ¿Tienes alguna idea de cómo podría mejorarse este tema?

9.2 HIBERNATE DA UN ERROR AL RECUPERAR INFORMACIÓN

Cuando ejecutamos la clase shopping.madrid.servlet.actions.shop.ShopSearch2Action para recuperar la información de la página de búsqueda me da el siguiente error:

*27 jun 2014 - 13:02:42 [ERROR] shop.ShopListPublicAction -
Exception:java.lang.ClassCastException: [Ljava.lang.Object; cannot be cast to
com.java.shopping.madrid.models.Shops*

No he conseguido eliminar este error y tuve que realizar esta operativa de otra manera.

Ahora mismo, el proyecto está funcionando por que en el fichero struts-config.xml asocié la llamada ="/searchShops a la clase ...madrid.servlet.actions.shop.ShopSearchAction.


```
<action name="Shops" path="/searchShops" scope="session"  
        type="com.java.shopping.madrid.servlet.actions.shop.ShopSearchAction"/>
```

Para comprobar el error, lo único que hay que cambiar es la línea 361 del fichero struts-config.xml y cambiar ShopSearchAction por ShopSearch2Action

9.3 ELIMINAR CACHE

Otro problema es que no consigo arreglar es eliminar la caché.

Cuando modifico por ejemplo un Mercado, y cambio la imagen, si voy al directorio donde se guardan las imágenes (web\images\markets) la imagen ha cambiado, pero si vuelvo a cargar la página de "Actualizar Mercados", se sigue mostrando la imagen antigua, una imagen que ya no existe en el directorio.

En todas los jsp's he añadido las siguientes líneas para eliminar la caché:

```
<meta http-equiv="pragma" content="no-cache">  
<meta http-equiv="cache-control" content="no-cache">  
<meta http-equiv="expires" content="0">  
<meta name="expires" content="Wed, 01 Jan 1997 00:00:00 GMT">
```

10. SOFTWARE

Para la realización de este proyecto he usado el siguiente software:

1. Sistemas operativos: Windows 8 y Ubuntu 13.04
2. *NetBeans IDE 7.4*. Entorno de desarrollo integrado libre.
3. XAMPP (tanto en la versión Windows como Linux). Servidor independiente de plataforma, software libre, que consiste principalmente en la base de datos MySQL, el servidor web Apache y los intérpretes para lenguajes de script: PHP y Perl. Aunque NetBeans ya trae incorporado un servidor Apache, he usado XAMPP para crear la base de datos.
4. Prepros App.
5. Sublime. Editor de texto
6. Gimp 2. Editor de imágenes

11. BIBLIOGRAFÍA

1. [Api Java Oracle](#)
2. [Stackoverflow](#)
3. [Wikipedia](#)
4. [Masonry.desandro.com](#)
5. http://www.slideshare.net/Emmerson_Miranda/log4j-1215-short-manual
6. <http://www.javatutoriales.com/2011/04/log4j-para-creacion-de-eventos-de-log.html>
7. Programación Javascript y jQuery. David Sawyer McFarland. Editorial Anaya.
8. Programación HTML5, CSS3 Y Javascript. Julie C. Meloni. Editorial Anaya.
9. HTML & CSS, design and build websites. Jon Duckett
10. Programador certificado Java 2. Curso práctico. 2ª edición. Antonio J. Martin Sierra. Editorial Ra-ma

*For the brave souls who get this far: You are the chosen ones,
the valiant knights of programming who toil away, without rest,
fixing our most awful code. To you, true saviors, kings of men,
I say this: never gonna give you up, never gonna let you down,
never gonna run around and desert you. Never gonna make you cry,
never gonna say goodbye. Never gonna tell a lie and hurt you.*