

Proyecto TFC

Agregador WEB

Preparado por

Alejandro Miguel Reyero Abad

Usuario UOC

areyero

Versión 1.1 Final

Revisiones

Fecha	Autor	Versión	Descripción
12-03-2007	Alejandro Reyero	PEC1	Documento para la Primera PEC
16-04-2007	Alejandro Reyero	PEC2	Documento para la Segunda PEC
21-05-2007	Alejandro Reyero	PEC3	Documento para la Tercera PEC
18-06-2007	Alejandro Reyero	Entrega Final	Documento Final

Tabla de Contenidos

1	Descripción	4
2	Especificación del Proyecto.....	4
2.1	Tecnología a Utilizar	4
2.2	Especificación de Funcionalidades	4
2.2.1	Alta de Proveedor	4
2.2.2	Baja y Modificación de Proveedor.....	4
2.2.3	Alta de Usuario.....	4
2.2.4	Baja y Modificación de Usuario.....	4
2.2.5	Mapeo de datos de Proveedor.....	4
2.2.6	Añadir Producto a Base de Datos.....	4
2.2.7	Baja y modificación de producto de base de datos	4
2.3	Funcionalidades futuras	4
3	Planificación Temporal	4
4	FASE DE ANALISIS: Actores.....	4
4.1	Administrador principal	4
4.2	Creador de contenidos	4
4.3	Visitante del portal (cliente final).....	4
5	FASE DE ANALISIS: Diagramas de secuencia.....	4
5.1	Mapear Campos	4
5.2	Insertar Producto	4
6	Elección de tecnologías.....	4
6.1	Framework MVC.....	4
6.2	Servidor de Aplicaciones / Web.....	4
6.3	Base de Datos	4
6.4	Entorno de Desarrollo.....	4
7	Diagramas de Clase.....	4
7.1	Diagrama de clases de Base de Datos	4
7.2	Diagrama de Clases de Struts.....	4
7.3	Diagrama de Clases de Librería.....	4
8	Diagrama de Base de Datos	4
9	Implementación de la Seguridad.....	4
10	Uso de la Aplicación e Interfaz	4

10.1	Acceso a la web	4
10.2	Creación de un Proveedor	4
10.3	Añadir un mapeo a un Proveedor	4
10.4	Añadir campos a un Mapeo	4
10.5	Añadir Producto.....	4
11	Glosario.....	4
12	Bibliografía.....	4
12.1	Servidor de Aplicaciones.....	4
12.2	Struts	4
12.3	Netbeans.....	4
12.4	Hibernate.....	4
13	Anexo A: Índice de Imágenes	4
14	Anexo B: Instalación.....	4
14.1	Aplicación web	4
14.2	Base de datos	4
14.3	Ficheros de Configuración	4
14.4	Versiones de productos	4

1 DESCRIPCIÓN

El objetivo de la aplicación será el facilitar y automatizar en la medida de lo posible el trabajo de las personas que mantienen bases de datos de productos en Internet, aunque se puede extender en futuras versiones (y deberá estar preparado para ello) para otros usos como la actualización de la página web con información de terceros en tiempo cuasi-real.

El producto final será un agregador web totalmente parametrizable. ¿Qué significa esto? La idea sería realizar una aplicación web que permitiera recuperar información de otras webs (que no exporten RSS o XML) de forma parametrizable y configurable, sin hacer nada "a mano".

Un ejemplo sencillo sería el de un portal de videojuegos. Este portal necesita mantener una base de datos de juegos a los que se asocia toda la información que se genera y publica: noticias, revisiones, artículos, pantallas...

Dar de alta estos juegos a mano es un trabajo hercúleo, todos tendrán una descripción, un nombre, una compañía desarrolladora, otra editora, precio, edad recomendada, género (rol, primera persona...), plataformas disponibles (Xbox, PC, PS2...), etcétera.

Toda esta información está disponible de forma pública, tanto en las webs de las compañías desarrolladoras como en tiendas online como El Corte Inglés o la FNAC. Estas webs mantienen una estructura similar en sus páginas de productos (normalmente todas se generan de forma dinámica con la misma plantilla HTML para la salida), variando simplemente la información del artículo que presentan, siendo el código HTML el mismo entre productos.

El objetivo del desarrollo será pues desarrollar un sistema que permita mapear textos desde una página web a campos en tablas de una base de datos local.

Por ejemplo, se define un Proveedor Origen: El Corte Inglés.

Para este proveedor se define varios tipos de agregación de productos, uno de ellos "Videojuegos".

Para crear la definición de cómo se toman esos datos se pasa como parámetro, para cada uno de ellos (nombre del juego, compañía, fecha de salida...) el trozo de código HTML que lo precede y el que va a continuación (para poder parsear la página del producto y extraer ese dato) y luego se mapea a una tabla y un campo en esa tabla en la base de datos (se explica en detalle este mismo ejemplo más adelante en este documento).

De esta forma, una vez creado el agregador de "El Corte Inglés" solo se tendrá que pasar como parámetro la URL en la que ellos tienen el producto que se desea añadir a la base de datos y el sistema lo hace todo (se trae la información y la inserta en la base de datos de forma atendida o desatendida).

Esta primera aplicación que se le da al sistema es asíncrona y requiere interacción del administrador (buscar e introducir en El Corte Inglés la URL del

producto de la que “absorber” toda su información), aunque en una segunda fase (fuera del alcance, en principio, de este TFC) se quiere ampliar el programa para que traiga esa información y refresque la base de datos o una página cada X segundos con AJAX (ejemplo las cotizaciones bursátiles en finance.yahoo.com).

2 ESPECIFICACIÓN DEL PROYECTO

2.1 Tecnología a Utilizar

La aplicación será desarrollada usando tecnología J2EE, primando en el diseño e implementación el respecto al paradigma de Orientación a Objetos.

Se busca una independencia total de plataforma, eligiendo una base de datos y un servidor web / de aplicaciones que estén disponibles al menos en Linux (y los principales tipos de Unix) y Windows.

2.2 Especificación de Funcionalidades

La aplicación necesita para funcionar conexión de salida a Internet sin restricciones, además de una base de datos en la que almacenar su configuración y demás datos relacionados con su funcionalidad y otra base de datos (puede ser en la misma que la anterior, en otra instancia en el mismo servidor o incluso en otro servidor con otro gestor de bases de datos diferente) en la que la aplicación introducirá los datos obtenidos de las páginas de terceros.

2.2.1 Alta de Proveedor

Esta funcionalidad permitirá añadir una nueva fuente de datos de la que tomar la información necesaria para dar de alta un nuevo "producto" a la base de datos.

Por ejemplo se podría dar de alta una tienda online de venta de electrodomésticos para tomar de ella información y características de distintos modelos de televisión:

http://www.rebelio.com/es/urwfilter/catalog/do/action/ShowProductDetail/productId/31442/sname/Samsung_LE_32_R_71_B_Negro/index.html

Este mismo producto también se hubiera podido tomar directamente de la página del fabricante:

<http://www.samsung.com/es/products/tv/lcdtv/le32r71bxec.asp?page=Specifications>

En esta opción se da de alta el proveedor dando su nombre, descripción y URL principal.

2.2.2 Baja y Modificación de Proveedor

Funcionalidades básicas de edición y borrado de proveedores.

2.2.3 Alta de Usuario

Pantalla en la que se añaden usuarios y roles a la aplicación. Si la aplicación se integrara dentro de un entorno con usuarios ya existentes se tendría que proveer un mecanismo que permita aprovechar los mismos.

La versión a implementar en el TFC incluirá un único tipo de usuario / administrador, aunque tendrá que estar preparado el sistema de roles para permitir que pueda haber usuarios que puedan añadir nuevos proveedores, otros que tengan acceso a uno o varios proveedores, etc...

2.2.4 Baja y Modificación de Usuario

Funcionalidades básicas de edición y borrado de usuarios.

2.2.5 Mapeo de datos de Proveedor

La funcionalidad principal del programa y la que encierra la mayor parte de la complejidad de la misma.

Se selecciona un proveedor (ej: FNAC) y se procede al mapeo (uno a uno) de los datos a importar en el sistema.

Lo primero que hay que hacer es visualizar (futuras versiones podrían implementar un editor visual) el HTML de un producto (ej: <http://www.fnac.es/dsp/?servlet=extended.HomeExtendedServlet&Code1=2440490334&Code2=174&prodID=631610>). A continuación se elige el primer campo que se quiere mapear (ejemplo, "nombre del juego") y se introduce el código HTML previo y posterior, tanto como sea necesario para que sea unívoco, en el ejemplo:

```
<tr>
  <td class="arial16bold">

</td>

<td rowspan="2" align="center" valign="middle" nowrap
```



```
class="arial10">
```

Luego este campo se asigna a una tabla y campo de la base de datos y se decide el método de inserción (se explica a continuación).

Este proceso se realizará tantas veces como sea necesario para un producto, teniendo en cuenta que todos los datos mapeados contra la misma tabla se insertarán en la misma fila de la base de datos y que un único producto puede provocar la inserción de líneas en distintas tablas (ejemplo: en la tabla de "juegos" y en las tablas intermedias que relacionan "plataformas" o "edades" con la tabla principal).

Habrán además varios tipos de datos a añadir, desde un texto sencillo que va directamente en una columna de una tabla hasta un tipo "selección", en el que aparecería una lista desplegable (por ejemplo, "Plataformas" o "Edades") de la que seleccionar uno (típico para tabla intermedia).

Finalmente añadir que un mismo proveedor puede tener varios mapeos diferentes, por ejemplo, de la FNAC se puede querer importar electrodomésticos, libros y CDs, que tendrían distintos campos e irían en distintas tablas en la base de datos.

2.2.6 Añadir Producto a Base de Datos

Una vez que están dados de alta los proveedores y sus mapeos contra el sistema sólo hay que dar una URL del proveedor en cuestión en la que se encuentra la página con los detalles del artículo. El sistema se conectaría a esa página y presentaría una pantalla editable (para corregir, añadir o borrar textos) en la que se ven todos los datos ya parseados y listos para ser insertados en la base de datos.

2.2.7 Baja y modificación de producto de base de datos

Para dar de baja o modificar un producto una vez que se ha insertado en la base de datos se utilizará un interfaz externo a la aplicación: el del propio gestor de contenidos del portal. Esto es así ya que la aplicación es genérica, e impediría comprobaciones adicionales sobre los datos o modificar o borrar, como podrían ser enlaces a los ID's de los productos a modificar o borrar desde otras tablas (ejemplo: una noticia de videojuegos enlazada a un o unos productos añadidos a la base de datos).

El borrado en cascada en base de datos no se considera como opción por el riesgo enorme de pérdida de datos.

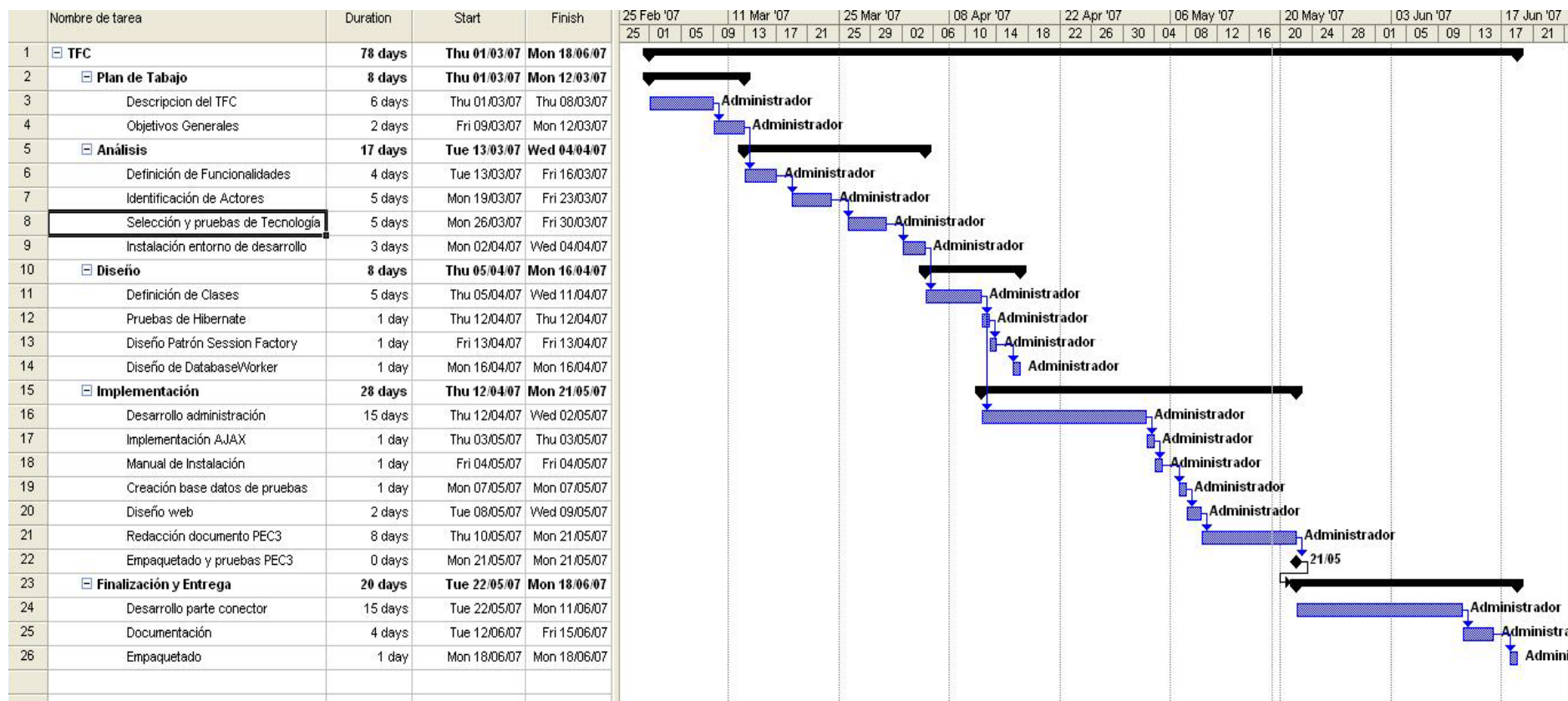
2.3 Funcionalidades futuras

A futuro se puede considerar el añadir nuevas funcionalidades al sistema, como la entrada automática de productos por lotes (ejemplo, todos los productos con la URL X que el parámetro de la URL id vaya de 1000 a 2500), la actualización de una página sin recarga total (AJAX) tomando datos de una página de un tercero sin necesidad de pasar por base de datos, etcétera.

El sistema tendrá que ser lo suficientemente flexible para poder crecer sin problemas derivados de una implementación demasiado apegada a la funcionalidad inicial

3 PLANIFICACIÓN TEMPORAL

Nombre	Duración	Inicio	Fin
TFC	78 Días	01/03/07	18/06/07
Plan de Trabajo	8 Días	01/03/07	12/03/07
Descripción del TFC	6 Días	01/03/07	08/03/07
Objetivos Generales	2 Días	09/03/07	12/03/07
Análisis	17 Días	13/03/07	04/04/07
Definición de Funcionalidades	4 Días	13/03/07	16/03/07
Identificación de Actores	5 Días	19/03/07	23/03/07
Selección y Pruebas de Tecnología	5 Días	26/03/07	30/03/07
Instalación entorno de Desarrollo	3 Días	02/04/07	04/04/07
Diseño	8 Días	05/04/07	16/04/07
Definición de Clases	5 Días	05/04/07	11/04/07
Pruebas de Hibernate	1 Día	12/04/07	12/04/07
Diseño Patrón Session Factory	1 Día	13/04/07	13/04/07
Diseño de DatabaseWorker	1 Día	16/04/07	16/04/07
Implementación	28 Días	12/04/07	21/05/07
Desarrollo Administración	15 Días	12/04/07	02/05/07
Implementación AJAX	1 Día	03/05/07	03/05/07
Manual de Instalación	1 Día	04/05/07	04/05/07
Creación base de datos de Pruebas	1 Día	07/05/07	07/05/07
Diseño Web	2 Días	08/05/07	09/05/07
Redacción documento PEC3	8 Días	10/05/07	21/05/07
Empaquetado y pruebas PEC3	0 Días	21/05/07	21/05/07
Finalización y Entrega	20 Días	22/05/07	18/06/07
Desarrollo parte conector	15 Días	22/05/07	11/06/07
Documentación	4 Días	12/06/07	15/06/07
Empaquetado	1 Día	18/06/07	18/06/07



4 FASE DE ANALISIS: ACTORES

Una vez conocido el alcance de la aplicación se procede a identificar a los actores que intervienen y sus casos de uso.

Se distinguen dos tipos de usuarios de la aplicación de gestión y un cliente final que será el visitante del portal, sin capacidad para modificar los datos que el aplicativo maneja, pero que interactuará con la información incorporada de una forma transparente para él.

4.1 Administrador principal

Este actor es el encargado de definir los proveedores de información y mapearlos al sistema, para que los creadores de contenidos se encuentren el sistema listo para importar datos directamente.

También será el encargado de la gestión de las cuentas de usuario y demás labores administrativas de la aplicación.



Imagen 1: Actor Administrador Principal

4.2 Creador de contenidos

Serán los utilizarán los mapeos y los proveedores de contenidos definidos por el administrador principal. Su labor será la de encontrar los productos a incorporar en los webs de los proveedores de contenidos, seleccionar la URL

de dicha web y comprobar que el mapeo automático que la aplicación realiza está correcto antes de añadir la información a la base de datos. También podrá corregir o ampliar la información de la fuente original y rellenar posibles campos en blanco.

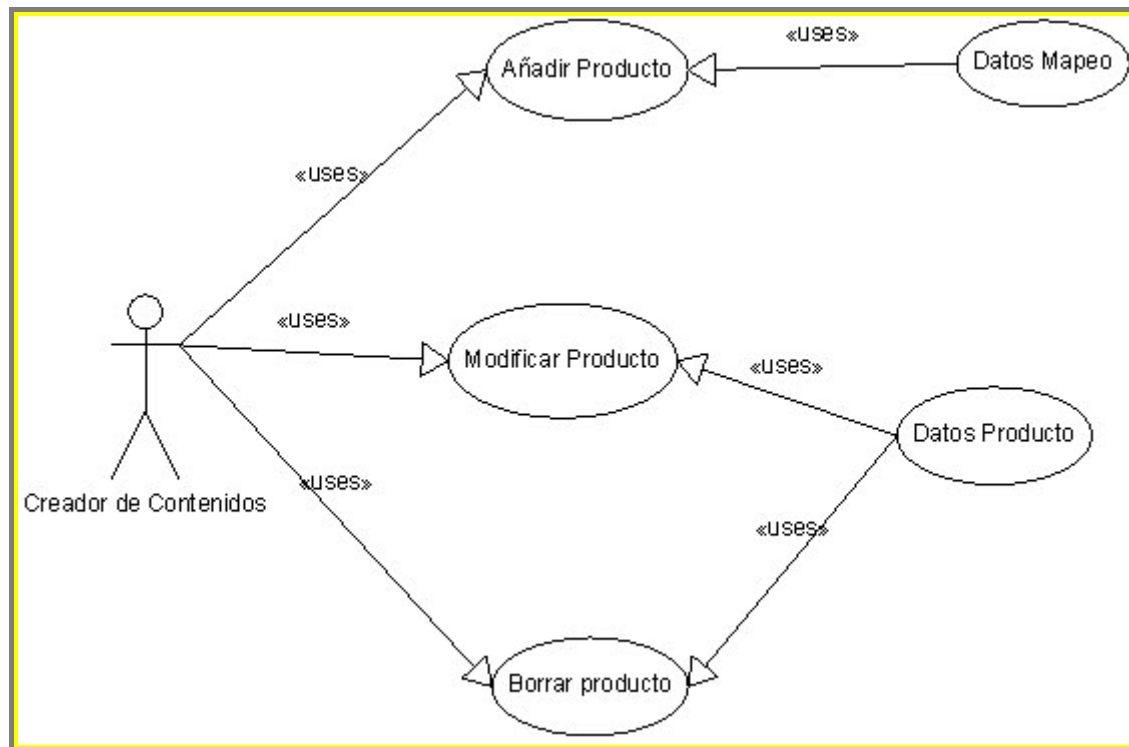


Imagen 2: Actor Creador de Contenidos

4.3 Visitante del portal (cliente final)

El usuario o visitante del portal se encontrará la información en el web tras ser importada y filtrada por la aplicación y el creador de contenidos. Para él la aplicación que no existirá, su interacción con la misma ocurre de forma indirecta al acceder a información que se ha importado.

Se ve en el Diagrama de casos de uso que a los productos no solo se puede hacer referencia en la página que lo describe, también podrá estar en alguna otra relacionada (noticias, revisiones o avances en el caso de videojuegos, por ejemplo).

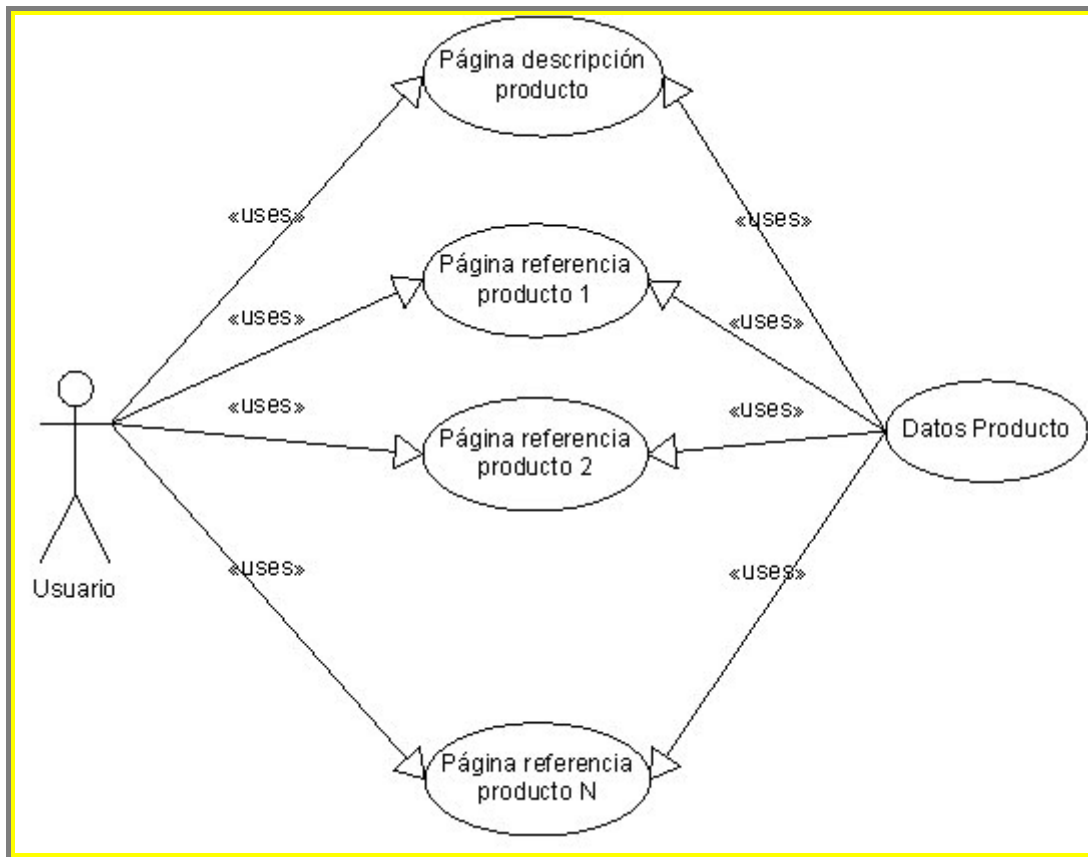


Imagen 3: Actor Usuario

5 FASE DE ANALISIS: DIAGRAMAS DE SECUENCIA

La aplicación a desarrollar incluirá dos funcionalidades especialmente complejas, cuyo funcionamiento se explica a continuación de forma detallada a partir de sus correspondientes diagramas de secuencia.

5.1 Mapear Campos

El proceso de mapeo de campos consiste en añadir nuevas entradas a un mapeo concreto. Por ejemplo, se tiene un mapeo con la sección de perfumería del web El Corte Inglés que ya tiene incorporados los campos "NOMBRE" (denominación del producto), "EMPRESA" (marca que lo comercializa), "PRECIO" y "FAMILIA" (tipo de producto: Colonia, Perfume...). Ahora se desea traer además a la base de datos dos campos más: "BENEFICIOS" (descripción de marketing sobre los supuestos beneficios de productos de estética) y "UTILIZACIÓN" (instrucciones de aplicación).

Detallado, el proceso de añadir un nuevo Detalle de Mapeo (o crear el primero asociado a un mapeo concreto) es el siguiente:

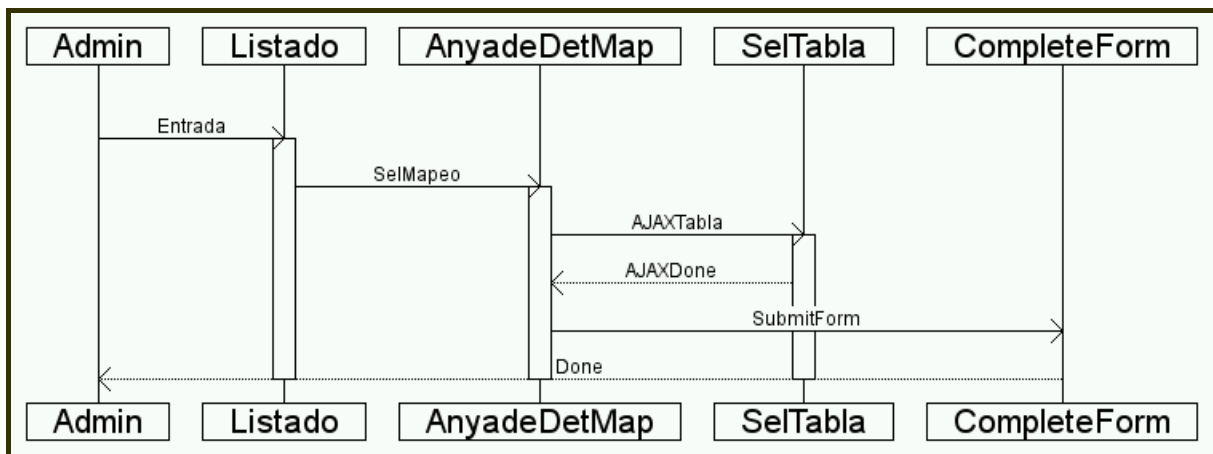


Imagen 4: Proceso de añadir un Detalle de Mapeo

El administrador accede al listado de Mapeos dados de alta, seleccionando "Añadir Campos" en el correspondiente a "Sección Perfumes el corte Inglés". A continuación selecciona en el combo la tabla de la base de datos (acción "AJAXTabla" en el diagrama) en la que se insertará el campo a añadir, que dispara un proceso AJAX para traer las descripciones de las columnas de la misma.

El siguiente paso es seleccionar la columna en la que se insertarán los datos importados, introduciendo el código HTML previo y posterior de la página original de El Corte Inglés que permitirá reconocer el texto a importar de forma automática para todos los productos de esa sección.

5.2 Insertar Producto

Una vez dados de alta los mapeos que compondrán los campos necesarios para dar de alta un tipo de producto, se puede proceder a que sea incorporado en el sistema con la aplicación.

El flujo completo de datos será gestionado por el Action "AnyadeProductoStep2", que recibirá como parámetro un campo hidden llamado "hace" para marcar el paso en el que se encuentra. La secuencia completa gestionada por este Action es la siguiente:

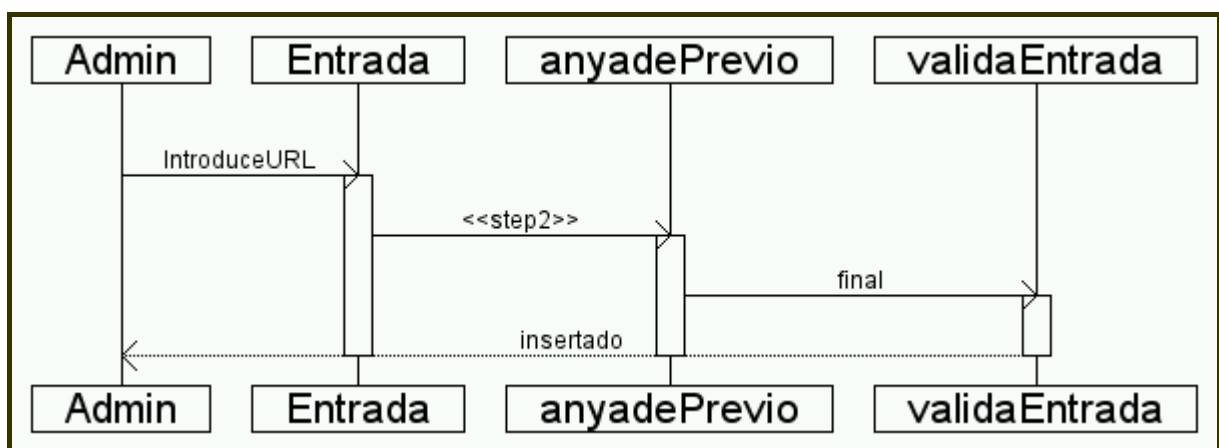


Imagen 5: Secuencia del Action AnyadeProductoStep2

La primera vez que se invoca a la acción el parámetro "hace" estará a null, con lo que el controlador se dirigirá al JSP "anyadeProductoURL" en el que se pide la URL del proveedor seleccionado.

Una vez introducida la URL se pasa a la página en la que se pre-visualiza la información que el sistema ha sido capaz de obtener de la página del producto en el proveedor. La acción que se desata para este paso, "step2", consiste en traerse el código HTML de la URL introducida en el paso anterior, recorrer todos los campos del mapeo, "machearlos" contra el texto obtenido y dejarlo en la sesión para que el JSP "anyadePrevio" se lo presente al usuario.

Si el mapeo ha sido correcto aparecerán ahora todos los campos (sea el número que sea) rellenos, aunque se presenta la información por si es necesario corregir, ampliar o modificar algo.

Como se ve en la secuencia, en caso de que todo esté correctamente mapeado, insertar un nuevo producto en la base de datos es tan simple como localizar el mismo en la web del proveedor, pegar la URL, y aceptar.

La complejidad técnica de la implementación de esta secuencia estriba en varios puntos:

- Para un mapeo no se sabe a priori cuantos campos ni cuales se van a utilizar.
- Hay que traerse el código de la URL remota y parsearlo contra todos los campos del mapeo.
- Hay que controlar que no se inserta en los campos de las tablas mapeadas más información de la que pueden contener (recordatorio: son desconocidos por el sistema así que siempre se trabaja con metadatos JDBC).
- Al querer mostrar al usuario los campos parseados para aceptar correcciones hay que construir un formulario HTML (con Struts) dinámicamente en el que almacenar la tabla en la que se van a insertar los datos, los campos y los valores, para finalmente construir "al vuelo" una sentencia SQL Insert de la que antes de empezar no se conoce nada (ni tabla, ni campos, ni valores).

6 ELECCIÓN DE TECNOLOGÍAS

El proyecto correrá en un servidor cuyo sistema operativo es un Linux CentOS 4.4 (clon del Red Hat Enterprise 4), para el que se eligen piezas de software de libre distribución por coste, rendimiento, documentación y, en algunos casos, acceso al código fuente.

La elección de la arquitectura de la aplicación ha sido relativamente sencilla. Cómo uno de los requisitos principales es la reutilización, separación de código de negocio, presentación y acceso a datos, además de querer a futuro ampliar y que sea escalable, se opta por un Modelo Vista Controlador.

6.1 Framework MVC

Como ya hay muchos frameworks MVC en el mercado se puede utilizar uno de ellos con el fin de optimizar el tiempo de desarrollo. A tal fin se comparan varios de los más utilizados y documentados, pasando a la criba final 3 opciones: Struts, Struts 2 y Spring.

Struts 2 se descarta por no estar disponible una versión final y ser su documentación y ejemplos escasos, además de no haber encontrado ningún entorno de desarrollo o plug-in que facilitara el desarrollo con dicho Framework (además de que no se necesita ninguna de sus novedades para el desarrollo).

Spring ha resultado más tentador por estar basado en el estándar Java Server Faces (JSF) y haber sido creado por el mismo desarrollador de Struts, aunque al final se encuentran los mismos problemas que con Struts 2: nulo soporte en los entornos de desarrollo y escasa documentación y ejemplos.

6.2 Servidor de Aplicaciones / Web.

La elección del servidor de aplicaciones ha venido precedido por la toma de decisión de la tecnología de acceso a datos a utilizar. Las opciones eran tres: Simples objetos DAO que encapsulen los accesos a datos sin mayores complicaciones ni funcionalidades, la utilización de clases de terceros que proporcionen persistencia (a ser posible utilizando el Java Persistence API introducido en J2EE 5.0) o el uso de EJB's.

Si se elige la tercera opción tendría que entrar en escena un motor de EJB's tipo JBoss o similar, pudiendo utilizar un simple motor de Servlets / JSPs en los dos primeros casos.

Como la carga de la aplicación no va a ser muy grande y, en principio, no se necesita tener los servidores de acceso a datos balanceados y comunicados con seguridad (RMI), se descarta el uso de EJB's y se elige como servidor de aplicaciones Tomcat, que, además de ser el más usado, es el que mejor se integra con los entornos de desarrollo disponibles.

Aunque en el desarrollo sólo se utiliza Tomcat (será también quien sirva los elementos estáticos como HTML, CSS, JS o imágenes), también existe la posibilidad de integrarlo con servidores web de terceros, más preparados para servir masivamente elementos estáticos y que además permitirían desacoplar la funcionalidad web en varios servidores, separando físicamente el Tomcat de las máquinas con los servidores web, que serían en todo caso Apache:

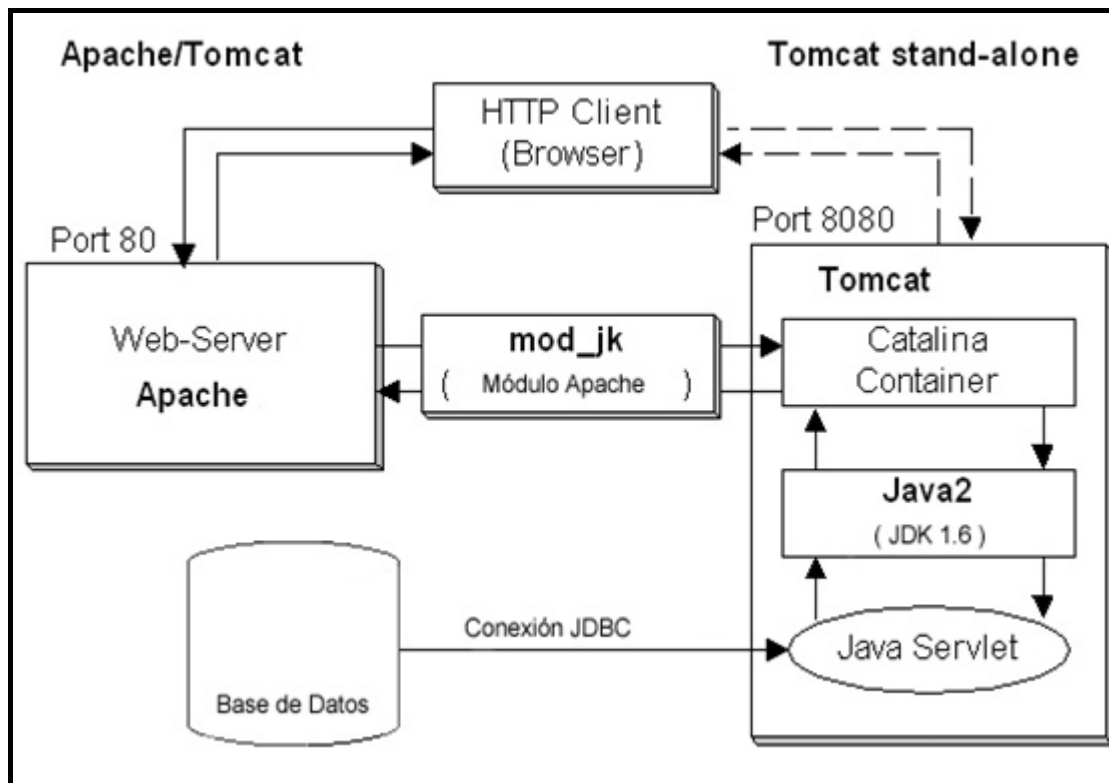


Imagen 6: Esquema Apache / Tomcat

6.3 Base de Datos

La decisión en este caso es sencilla: MySQL. Cumple todos los requisitos necesarios, hay soporte y documentación más que abundante y es la preferida en el entorno en el que va a correr el desarrollo: Linux.

Se decide además utilizar persistencia y usar para la misma Hibernate, que cumple el estándar Java Persistence API. En realidad para la aplicación no sería necesario la utilización de un API de persistencia, dado que la carga de trabajo grande será la del portal para el que se rellene la información a través del aplicativo, pero se utilizará por corrección metodológica y para aprovechar y aprender una nueva tecnología.

6.4 Entorno de Desarrollo

La elección "estándar" en estos momentos es el Eclipse, tremendamente popular y con infinidad de plug-ins, documentación y soporte. Cuando me puse a evaluar las funcionalidades como IDE global para el proyecto me encontré con que para conseguir del eclipse todas las funcionalidades deseadas tenía que decantarme por alguna implementación de pago o recopilar y configurar infinidad de plug-ins, que además llegan a entrar en conflicto entre si.

Las funcionalidades deseadas (y deseables) son las siguientes:

- Modo "Proyecto web" en el que se facilite el desarrollo (con cosas, por ejemplo, como Syntax Highlighting) y edición de fichero no .java, como HTML, CSS o JS.
- Integración, con modo debug a ser posible, del servidor de aplicaciones dentro del entorno de desarrollo, para no tener que hacer un deploy del proyecto para cada prueba.
- Editor y wizards para ficheros especiales de Struts (.XML de configuración, "templates" para clases de negocio...).
- Generador de clases de persistencia, con setters (inserts / updates) y getters (selects) automáticos, para Java Persistence API.
- Editor UML, a ser posible con ingeniería inversa de clases.
- Generador de .WAR (con ANT como motor a ser posible) para hacer deploy de forma sencilla en producción / pre-producción.

Tras buscar, instalar y configurar multitud de plug-ins para el Eclipse me encontré con que un montón de páginas y blogs en los que se hablaba de los mismos recomendaban el uso de Netbeans, sobre todo para proyectos web. Con reticencia por anteriores experiencias con dicho entorno de trabajo (hace años era lento, pesado y se "comía" la máquina) me decidí a instalar la versión 5.5 junto con el pack de desarrollo web y el pack UML, y he de decir que me ha sorprendido gratamente. No sólo soporta completamente los puntos que he mencionado anteriormente sino que es rápido y el debugger de JSP's / Servlets es de lo mejor que he visto hasta ahora.

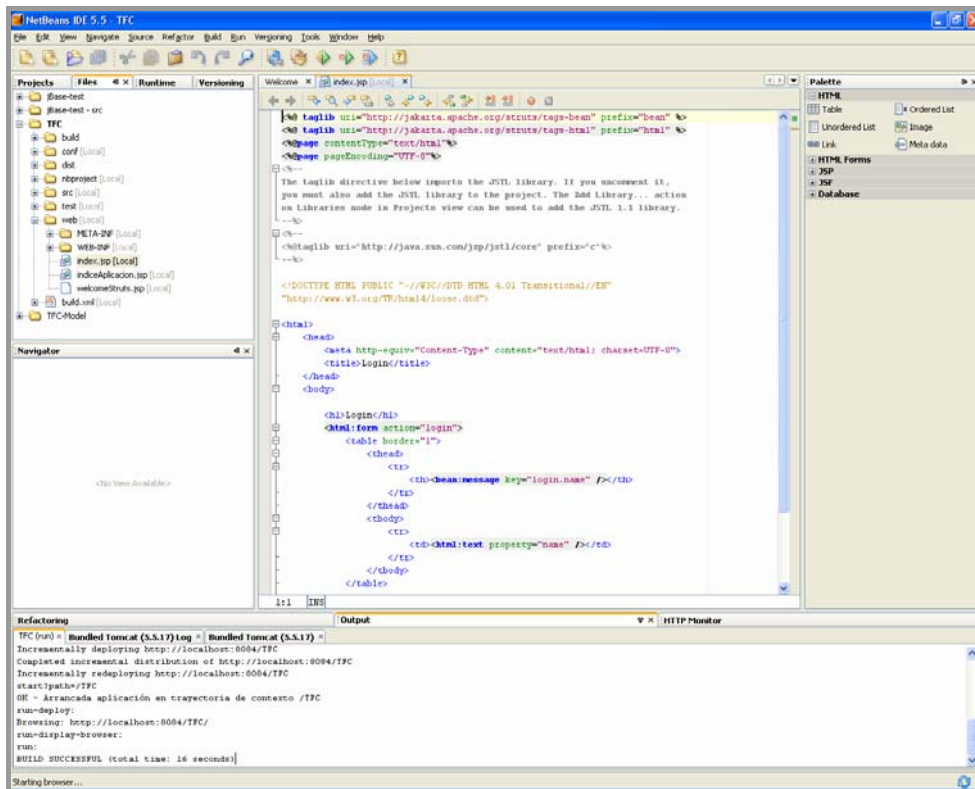


Imagen 7: Aspecto general del Entorno de Desarrollo

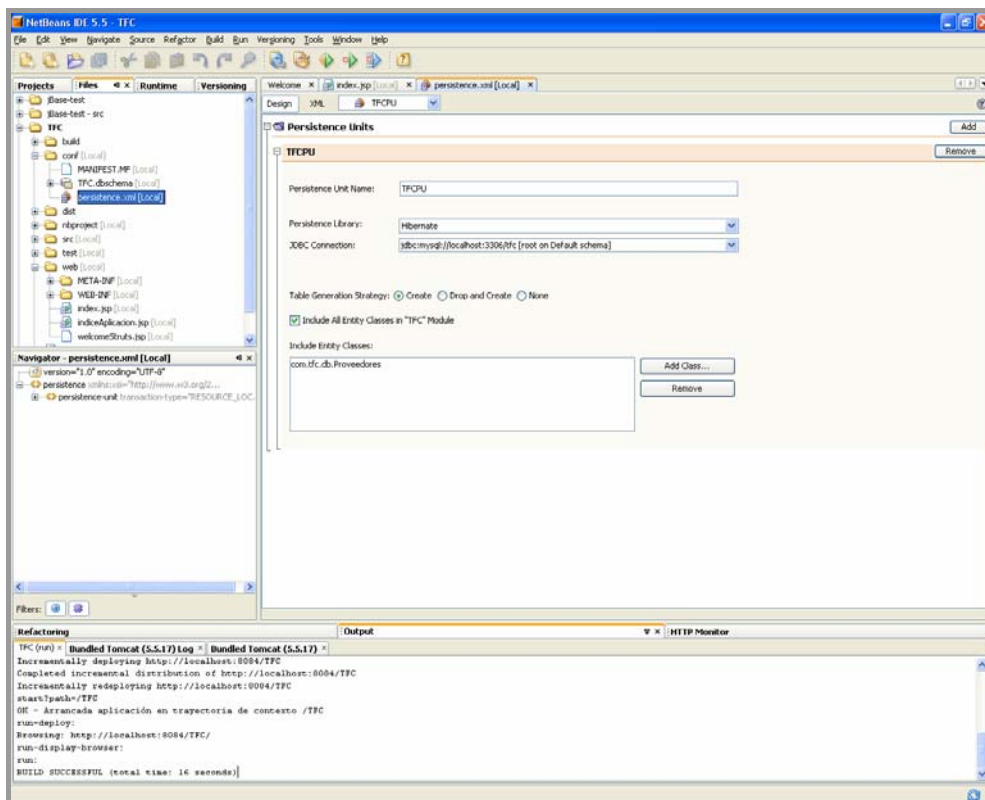


Imagen 8: Editor de Persistence.xml para Hibernate

7 DIAGRAMAS DE CLASE

La aplicación tiene más de 35 clases repartidas entre clases de mapeo de tablas de Hibernate, clases Action, ActionForm y DispatchAction de Struts y clases de apoyo, que se ha dado en llamar de librería. Cada uno de estos tipos de clase está agrupado dentro de paquetes funcionales:

- com.tfc.db
- com.tfc.library
- com.tfc.struts

Como el Diagrama de clases general es absolutamente inmanejable por su tamaño, le acompañan Diagramas separados por grupos funcionales, que se adjuntan además en ficheros JPG de gran tamaño, fuera del documento, para su visionado en detalle (es muy difícil hacerlos visibles en condiciones en el documento). También se adjuntan en formato ETLD/ETLP para su visionado en herramientas UML.

7.1 Diagrama de clases de Base de Datos

Este diagrama no incluye los métodos en las clases, sólo los atributos, para que se pueda leer algo. En la imagen adjunta fuera del documento se puede observar toda la clase en alta resolución.

El paquete de base de datos incluye las clases Hibernate para cada tabla / tipo de query. Estas clases van acompañadas además de ficheros de definición XML (Hibernate Mappings o HBM).

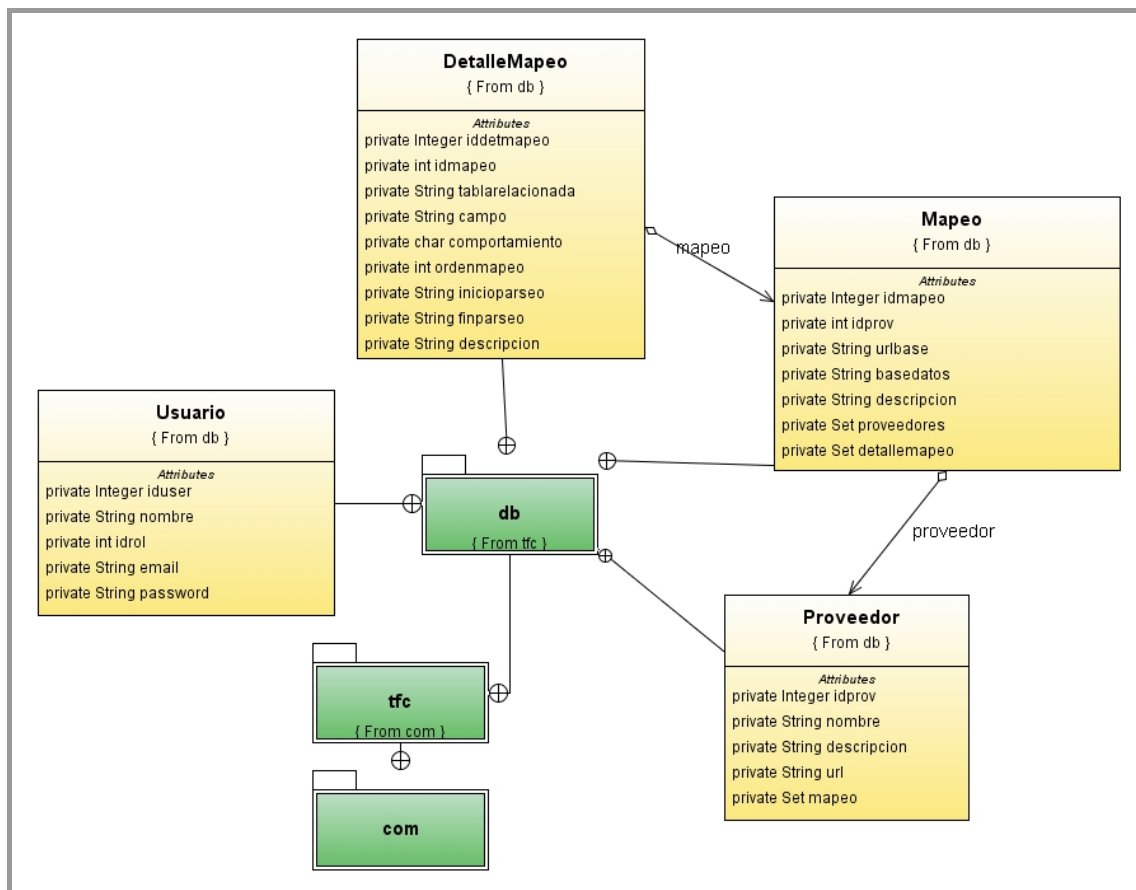


Imagen 9: Diagrama de Clases de Base de Datos

7.2 Diagrama de Clases de Struts

Toda la navegación en el web se realiza utilizando Actions de Struts, con la lógica de negocio en ActionForms, lo que permite la separación total entre las tres capas (la presentación va en JSPs que no incluyen ninguna lógica). Cada tipo de acceso administrativo a las tablas requiere cuatro beans: primero se listan los elementos deseados con los beans ListAction (extends org.apache.struts.action.Action) y ListForm (extends org.apache.struts.action.ActionForm) y la visualización del detalle, edición, e inserción se hará con los beans EditAction (extends org.apache.struts.actions.DispatchAction) y EditForm (extends org.apache.struts.action.ActionForm).

Por ejemplo, aplicado a la administración de Mapeos se tendría:

- MapeosListAction
- MapeosListForm
- MapeosEditAction

- MapeosEDit Form

Cuyas vistas se corresponden con los JSP:

- listaMapeos.jsp
- editaMapeos.jsp

Un caso particular dentro del paquete Struts, que se ha decidido incluir aquí por ser un "Action" en vez de en su lugar "natural" que sería la Librería, es GetTableInfoAJAX.

Este Bean implementa tecnología AJAX: En la edición de Detalles de Mapeo se tienen que obtener dinámicamente el listado de las tablas de la base de datos sobre la que el mapeo trabaja, una vez seleccionada la tabla para la que crear el mapeo hay que obtener los detalles de los campos de la misma para permitir al usuario seleccionar el campo donde insertar la información. Para no traer la metainformación de todas las tablas de la base de datos con la página, ni recargar al seleccionar la tabla, se usa el objeto XMLHttpRequest (Microsoft.XMLHTTP en Internet Explorer) a través de la clase para cargar los datos de los campos en el segundo combo de forma dinámica.

El código JavaScript de invocación a getTableInfoAJAX y uso de XMLHttpRequest o Microsoft.XMLHTTP se encuentra en editaDetalleMapeo.jsp

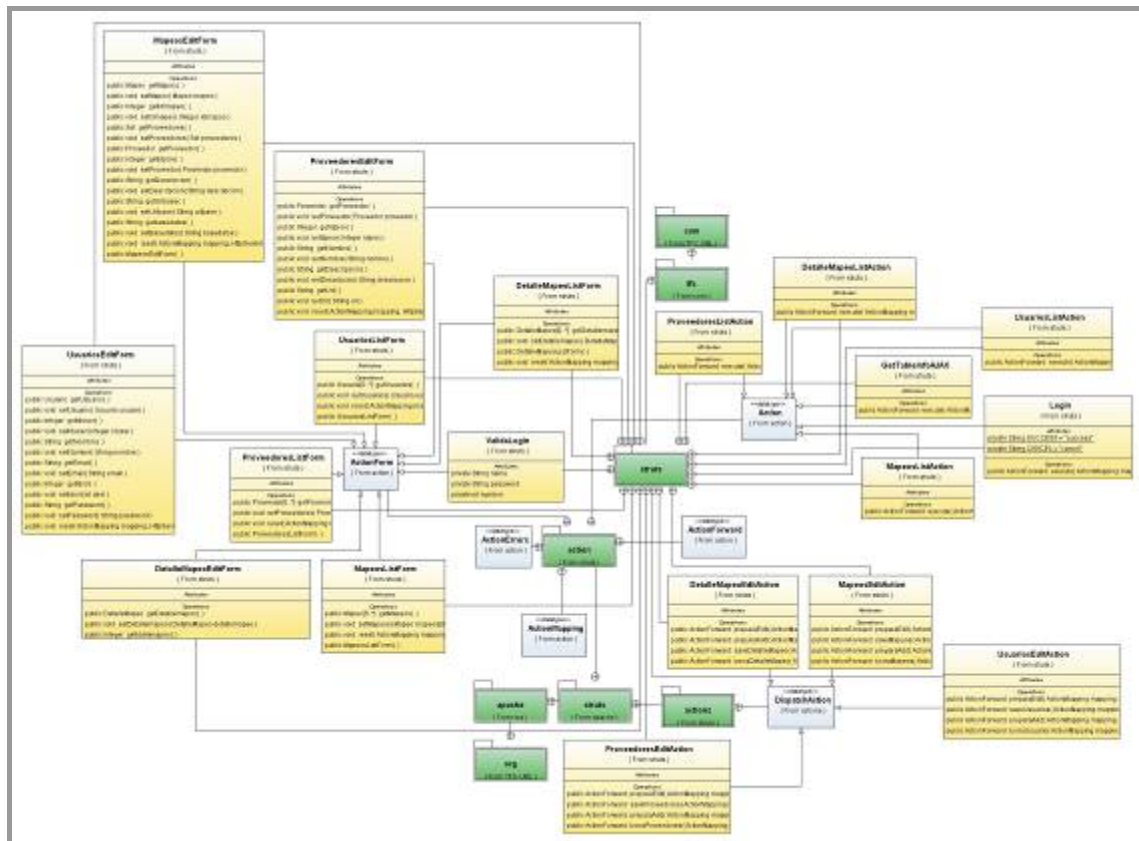


Imagen 10: Diagrama de Clases de Struts

7.3 Diagrama de Clases de Librería

Dentro de este paquete se incluyen las clases de apoyo, en las que se ha encapsulado toda la funcionalidad reutilizable y ajena “per se” a las clases de negocio, de navegación o de mapeo de datos de la base de datos.

La clase **HibernateUtil** implementa el patrón de diseño Session Factory que permite asegurarnos que sólo se usa una instancia de “session” por cada Thread, de esta forma TODOS los accesos a sesiones de Hibernate se hacen a través de este objeto, lo que no facilita el debug al reducir a un único punto de fallo los posibles problemas de conexión, así como evitar reescribir código.

OptionItem es una pequeña clase de apoyo que se utiliza en los DetalleMapeo para construir una ArrayList controlada, comparable y ordenable que será la que “viaje” en la página AJAX sin necesidad de utilizar Reflection o Upcasting para trabajar con sus contenidos.

LibraryManager es la clase que permite no desconectar de la base de datos la sesión Hibernate. Las queries Hibernate devuelven un interfaz a una lista que está directamente ligada al objeto Sesión. Se podría haber utilizado un

mecanismo de cacheado, pero se escapa un poco del alcance del proyecto y esta solución es limpia y eficaz.

DatabaseWorker clase abstracta que encapsula toda la funcionalidad de base de datos necesaria para el listado de tablas y la descripción de los campos de las mismas. Utiliza los metadatos de JDBC, lo que la hace además totalmente independiente de la base de datos utilizada. Para este desarrollo se implementa la funcionalidad de la base de datos en la clase **MySQL**, que extiende la anterior con particularidades de MySQL.

InfoContainer es una clase de apoyo para almacenar los campos necesarios para pintar el formulario de DetallesMapeo una vez parseados de la página original (tabla, campo, ordencampo y textoparseado).

TFCActionServlet es la clase que extiende al ActionServlet e implementa la seguridad de la aplicación. Se apoya en la clase **SessionManager**, cuyo funcionamiento es simple: rechaza peticiones a recursos que intenten ser accedidos por URL directamente y recupera un objeto de sesión con el que se podrían implementar comprobaciones adicionales.

URLParser es la clase encargada de descargar el código HTML de la URL solicitada, con métodos para devolverlo como String.

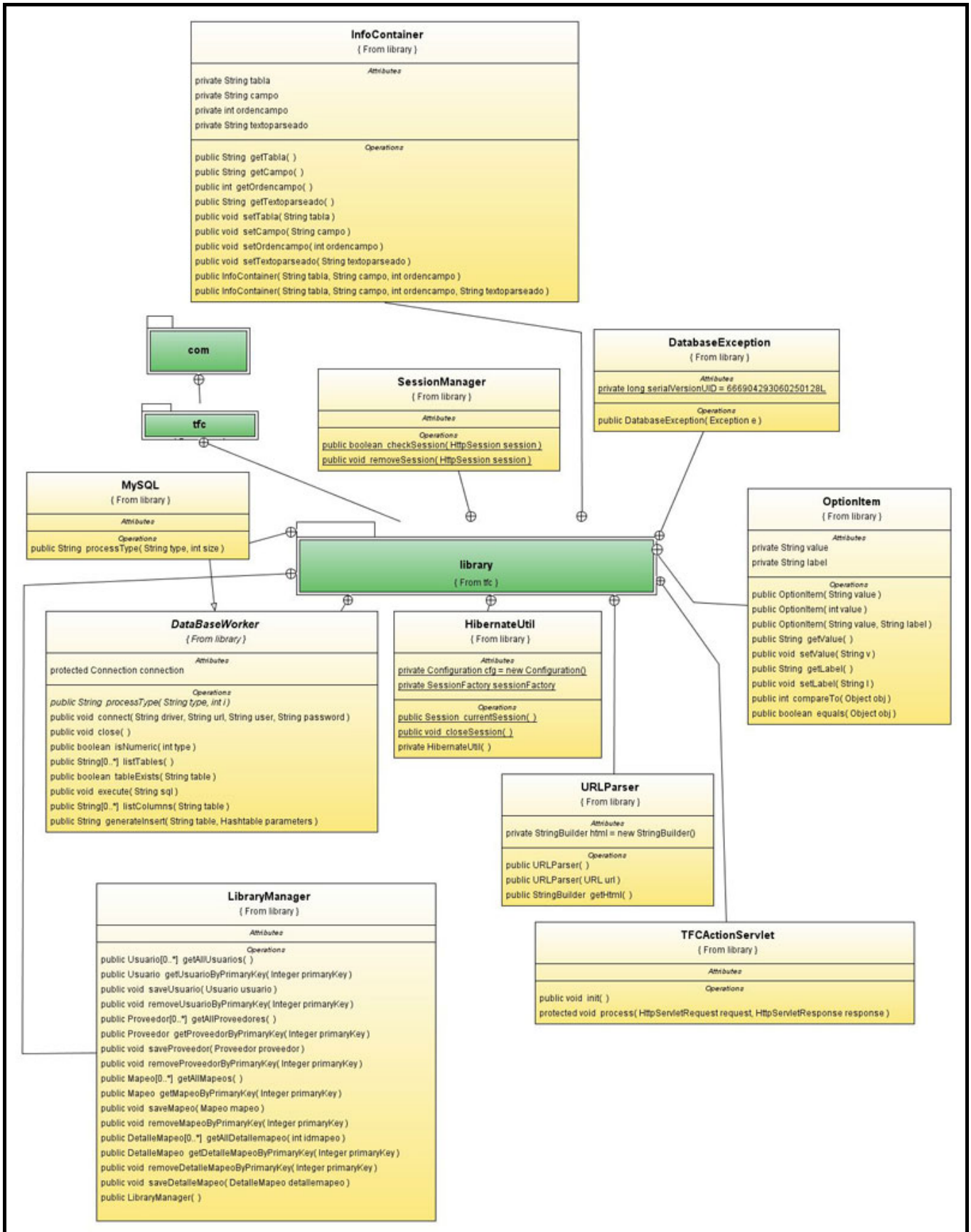


Imagen 11: Diagrama de Clases de Libreria

8 DIAGRAMA DE BASE DE DATOS

En la base de datos hay que dar cabida a la gestión de usuarios, que puede ser delegada, si existe, a la que utilice la intranet en la que la aplicación se integre. Para el proyecto se implementa una seguridad simple, almacenando los parámetros necesarios (usuarios, grupos...) en la propia base de datos. En el próximo punto se explica la implementación de seguridad y cómo sería acoplable dentro de cualquier sistema de forma sencilla.

Las tablas del aplicativo tienen que poder contener la meta-descripción de todos los mapeos de datos, así como el proveedor al que están asociados. Las relaciones entre las tablas son relativamente complejas (al ser totalmente genérico una vez comience el desarrollo no se pueden "cablear" funcionalidades) pero el esquema está compuesto por pocas tablas. Hay que tener en cuenta además que el objetivo final de la aplicación es insertar datos en tablas de terceras aplicaciones, con lo que el esquema final puede ser tan complejo como lo sea el de la base de datos objetivo. Una vez terminado el desarrollo se adjuntará a la memoria un esquema completo de la base de datos contra la que se han hecho las pruebas.

El esquema básico de la aplicación es el siguiente:

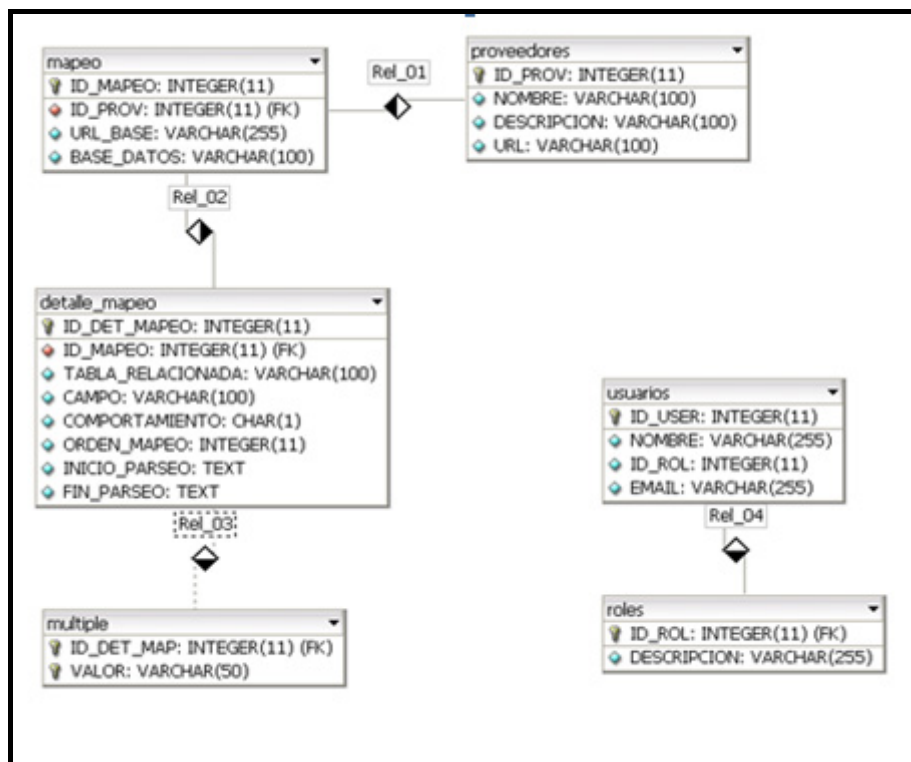


Imagen 12: Diagrama de Base de Datos

NOTA: En la tabla "detalle_mapeo" se incluye un campo llamado "COMPORTAMIENTO" que puede tomar tres valores:

- N → Normal (por defecto).
- M → Múltiple. El campo toma sus datos de una tabla tercera (se implementará si da tiempo en la primera fase).
- L → Limitado. Se presentará una caja de texto vacía del tamaño de la columna "CAMPO".

Para el proyecto se limita el alcance de este modificador de comportamiento por la escasez de tiempo de desarrollo, pero los "beans" de mapeo de base de datos y las clases Hibernate ya lo tienen en cuenta para que su

El script de creación del Esquema se adjunta en el fichero -estructura.sql . También se adjunta el esquema test.sql para la creación de una base de datos con tablas de ejemplo en la que importar productos de otras webs con el agregador.

9 IMPLEMENTACIÓN DE LA SEGURIDAD

El objetivo del agregador no es su funcionamiento como aplicación "stand-alone", ha de integrarse en un entorno de administración de una Intranet, ya que su función es alimentar una base de datos que será explotada desde otros aplicativos.

Para facilitar la integración en sistemas de terceros, a la vez que proveer "de fábrica" un mecanismo de autenticación mínimo, se ha optado por extender el Servlet Action del API de Struts para implementar una validación propietaria, que podrá ser sustituida de forma muy sencilla sin tocar el resto de la aplicación.

En el fichero de configuración del servidor de aplicaciones (web.xml) se sustituye el Action de Struts por el de la aplicación: TFCActionServlet. Este servlet extiende el ActionServlet de Struts, implementando un "process" en el que si la URL es distinta a la de entrada a la aplicación se hace la comprobación de que existe una sesión, usando una clase desarrollada a tal efecto: "SessionManager". En el punto de entrada de la aplicación, el action "Login.java", se crea una sesión en la que se incluye un ID arbitrario (el objetivo no es realizar comprobaciones adicionales, sólo crear la sesión para que TFCActionServlet la encuentre y de OK como logado) tras comprobar contra la base de datos que el usuario y el password son correctos. Si la sesión no existe el usuario no ha entrado por el login, así que se le redirige a la página de entrada.

Si se quisiera integrar la aplicación en alguna intranet sólo habría que modificar el Login.java para que se compruebe contra las tablas de seguridad deseadas y cambiar el TFCActionServlet si se desea que almacene en la sesión algún dato adicional.

10 USO DE LA APLICACIÓN E INTERFAZ

Se ha integrado la aplicación dentro del interfaz de administración de la web TodoJuegos.com , como ejemplo de importación de datos de producto (en este caso videojuegos) de webs como El Corte Inglés, FNAC o Game.es

La aplicación está completamente desvinculada del web en el que se integre y sólo requiere la creación de un esquema de base de datos de nombre TFC (aunque es configurable) y la modificación de la plantilla "include.jsp" , que define el aspecto visual. Esta plantilla está creada para posicionar por CSS los distintos elementos visuales (cabecera, menú y contenido), de tal forma que modificar el aspecto visual o incluso realizar una integración rápida en la web administrativa de cualquier portal sea rápido y sencillo.

10.1 Acceso a la web

La aplicación incluye una sencilla gestión de usuarios, reemplazable por el sistema de seguridad y autenticación del web en el que se integre. No obstante para la demostración de uso y para webs sin backend es necesaria y, aunque no potente, permite el uso seguro de la aplicación desde el primer día:



Imagen 13: Interfaz de Acceso

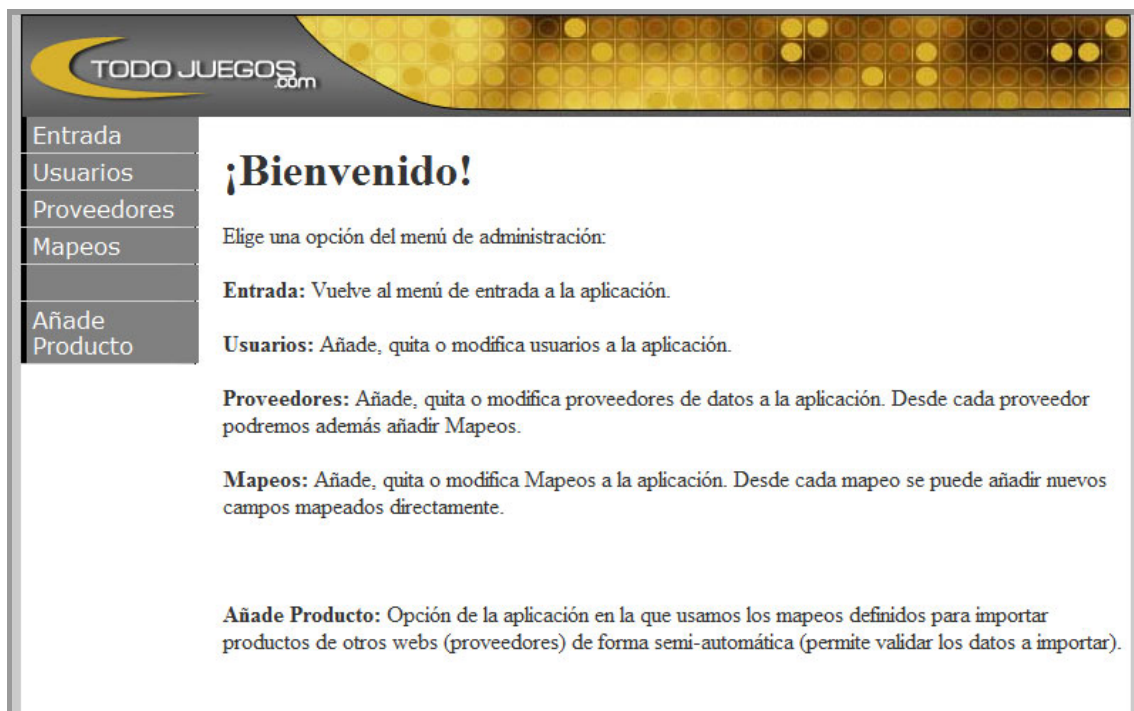


Imagen 14: Pantalla de Inicio

En la instalación por defecto se incluye un usuario "admin" con password "admin". Lo primero que hay que hacer es entrar en la administración de usuarios y cambiar este password:



Imagen 15: Administración de Usuarios

Imagen 16: Edición de Usuarios

10.2 Creación de un Proveedor

Los proveedores serán los portales / tiendas de las que se toman los datos. A cada proveedor se le podrá asignar uno o varios mapeos, por ejemplo se puede tener un mapeo de El Corte Inglés que importe a la base de datos nuevos juegos de ordenador, otro que importe DVDs musicales, etc.

Nombre	Descripción	URL			
El Corte Ingles	Web de El Corte Ingles.	http://www.elcorteingles.es/	Añadir Mapeo	Editar	Borrar

[Añadir Proveedor](#)

Imagen 17: Listado de Proveedores

Se selecciona "Añadir Proveedor" y se introduce "Game.es"

Imagen 18: Edición de Proveedores

Nombre	Descripción	URL			
El Corte Ingles	Web de El Corte Ingles.	http://www.elcorteingles.es/	Añadir Mapeo	Editar	Borrar
Game	Game (antiguo Centro Mail)	http://www.game.es	Añadir Mapeo	Editar	Borrar

[Añadir Proveedor](#)

Imagen 19: Alta de Proveedores

10.3 Añadir un mapeo a un Proveedor

Para importar datos de un proveedor lo primero que hay que hacer es crear un mapeo al que asociar los campos que se quieren importar. Se selecciona "Añadir Mapeo" en la página de proveedores:

Imagen 20: Añadir mapeo a un Proveedor


10.4 Añadir campos a un Mapeo

Una vez está definido lo que se quiere importar (en este caso serán descripciones y datos de juegos de la web "Game.es") se añaden los campos deseados:

Proveedor	Descripción	URL de Base	Base de Datos				
Game	Mapeo de Prueba	http://www.game.es/ficha/ficha.aspx	test	Ver Campos	Añadir Campos	Editar	Borrar

Imagen 21: Añadir Campos a un Mapeo

Se selecciona "Añadir Campos":



Entrada
Usuarios
Proveedores
Mapeos
Añade
Producto

Mapeos

Mapeo:

Descripcion:

Tabla Relacionada:

Campo:

Comportamiento:

Orden Mapeo:

Inicio Parseo:

Fin Parseo:

Imagen 22: Añadir Campos a un Mapeo

Para añadir un mapeo se necesita conocer los tags HTML que preceden y anteceden en cada caso a la información a extraer de la página. En este ejemplo primero se visualiza una página de producto en www.game.es, por ejemplo <http://www.game.es/ficha/ficha.aspx?sku=047733> que se corresponde con el videojuego "God of War II":



Imagen 23: Visualizar página de producto ejemplo

En esta página se localiza todos los datos a extraer, en este caso empezamos por el Nombre del producto. Para ello se necesita conocer los trozos HTML mencionados para la plantilla, para lo que es necesario visualizar el HTML de la página buscando el nombre del producto (God of War II):

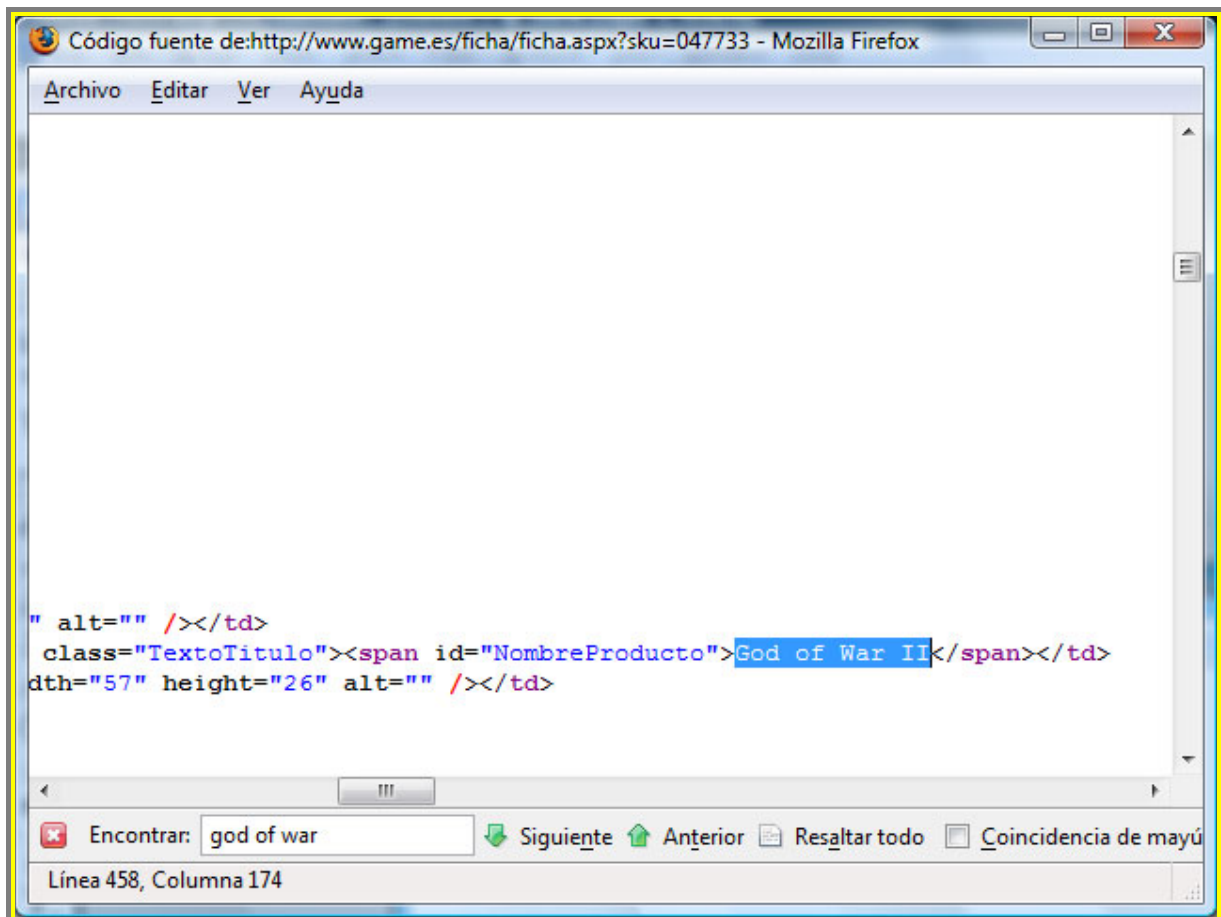


Imagen 24: Visualización de código

El tag que lo precede es "``". Como en el resto del documento se comprueba que ese mismo tag no aparece se puede utilizar para "marcar" al campo "Nombre". El parseador buscará luego a partir del tag de inicio para conocer donde está la primera ocurrencia del tag final y dejar de leer el campo, con lo que en este caso el fin de parseo será "`</td>`".

NOTA: seleccionar la tabla en la que se quieren insertar los datos importados se hace una petición AJAX al servidor para rellenar el combo box "Campo", de tal forma que la navegación del usuario no se vea interrumpida por una recarga de página adicional:

Imagen 25: Añadir campos a un Mapeo

Mapeo	Descripción	Campo	Tabla	Orden	Inicio	Fin	Borrar
Mapeo de Prueba	Nombre del Juego	NOMBRE	juegos	1		</td>	Borrar

Imagen 26: Campos Mapeados

10.5 Añadir Producto

Una vez que todos los campos necesarios para añadir un nuevo producto a la base de datos han sido dados de alta en un Mapeo ya se puede empezar a importar de forma automática a la base de datos. El ejemplo con el que se ha trabajado hasta ahora, Game.es, se encuentra incluido completo en la

instalación del proyecto, para poder probar que todo funciona nada más instalar.

Para empezar la importación de un nuevo producto se selecciona la opción de menú "Añade Producto", que nos presentará una lista de los mapeos existentes en nuestro sistema:



Imagen 27: Añadir Producto

Tras seleccionar el mapeo, en este caso "Mapeo de Prueba" del proveedor Game, se solicita la URL del proveedor en la que está la información del producto. Para el ejemplo se utiliza la del juego de Xbox 360 "Forza Motorsport 2":

<http://www.game.es/ficha/ficha.aspx?sku=043628>



Imagen 28: Introducir la URL del producto

El sistema se conectará a la URL proporcionada y en base a los campos de ese mapeo extrae la información y la presenta para verificación y/o modificación:

The screenshot shows a web application interface for 'TODO JUEGOS .com'. On the left, there is a vertical navigation menu with the following items: 'Entrada', 'Usuarios', 'Proveedores', 'Mapeos', and 'Añade Producto'. The main content area displays a form for adding a product. The form fields are as follows:

- NOMBRE:** Forza Motorsport 2
- DESCRIPCION:** Forza Motorsport 2, continuación del premiado simulador de conducción personalizable Forza Motorsport de Microsoft Game Studio, llegará raudo y veloz a Xbox 360 a finales de año. Abróchate el cinturón para disfrutar de su increíble física de simulación, espectaculares golpes y gráficos hiperrealistas, así como de sus opciones de "tuneado" y personalización con licencia.
- COMPANY:** MICROSOFT
- EDAD:** 3
- GENERO:** CONDUCCION

At the bottom of the form, there are two buttons: 'Submit' and 'Cancel'.

Imagen 29: Producto parseado

Como se observa, en este ejemplo se han mapeado los campos NOMBRE, DESCRIPCION, COMPANY, EDAD y GENERO. De la página de Game se podría extraer mucha más información, como Fecha, precio, etc. Un ejercicio interesante para conocer la corrección de la instalación podría ser añadir algún otro mapeo a la tabla "juegos" de la base de datos "test" (previa modificación de esta para añadir las columnas correspondientes) y comprobar que el mapeo funciona.

Una vez comprobado que todo está correcto se procede a aceptar el formulario y el producto se habrá insertado en la base de datos. Como se puede apreciar con este ejemplo, añadir nuevos productos a una base de datos existente, una vez hecho el trabajo de mapear los campos, es tan sencillo como poner la URL del producto en el proveedor y esperar un par de segundos (dependiendo de la conexión de la máquina en la que esté instalada la aplicación) a que la aplicación descargue y filtre la información.

11 GLOSARIO

Agregador: Aplicación capaz de recopilar información de distintas fuentes de datos e incorporarla al repositorio de un tercero.

AJAX: Asynchronous JavaScript And Xml, técnica de desarrollo web para crear aplicaciones interactivas.

DAO: Data Access Object, patrón para abstraer la forma en la que la aplicación accede a la base de datos.

Framework: Librería que ofrece una serie de interfaces, clases y otras utilidades poniendo a nuestra disposición un marco de trabajo delimitado en el que programar de forma sencilla tareas más complejas.

IDE: Integrated development environment, entorno de desarrollo (para este proyecto se ha usado Netbeans 5.5).

J2EE: Entorno para desarrollar e implementar aplicaciones empresariales basadas en web y con varias capas. La plataforma J2EE consiste en un conjunto de servicios, API y protocolos que proporcionan las funciones necesarias para desarrollar estas aplicaciones.

JavaBean: Clase de Java que puede manipularse mediante herramientas y utilizarse para elaborar aplicaciones. Un componente JavaBeans debe adherirse a determinadas convenciones de propiedades e interfaces de eventos.

JDBC: Java DabaBase Connectivity, interfaz de acceso a base de datos de Java.

JSP: Documento que contiene texto estático y elementos que describen cómo procesar una solicitud para crear una respuesta. Una página JSP se traduce y maneja solicitudes como servlet.

Mapeo: Cada uno de los objetos de datos que son tomados de la página de un tercero para ser incorporado a una base de datos. Cada mapeo ha de tener unos códigos HTML que enmarcan el texto a extraer.

MVC: Modelo Vista Controlador, patrón de arquitectura de software el cual desacopla la lógica de negocio respecto la presentación de los datos.

ORM: Object Relational Mapping, mapeo objetos-relacional, en el proyecto se usa el software libre Hibernate.

Patrón de diseño: expresión de la solución a un problema que es aceptada como la mejor para un escenario concreto.

Request Object: Objeto que contiene datos de página y sesión producidos por un cliente, transmitidos como un parámetro de entrada a un servlet o JSP.

Seguridad: Mecanismo de filtrado que garantiza el acceso exclusivo de clientes autorizados a los recursos de la aplicación.

Security Context: Objeto que encapsula la información de seguridad.

Servlet: Programa de servidor escrito en Java que amplía la funcionalidad de un servidor web, generando contenido dinámico e interactuando con aplicaciones web utilizando el paradigma solicitud-respuesta.

Servlet Engine: Conjunto de procesos que ofrece servicios a un servlet, incluidas la instalación y la ejecución.

Sesión: Objeto utilizado por un servlet o bean de sesión con estado que se utiliza para realizar el seguimiento de la interacción del usuario con una J2EE o aplicación web en varias solicitudes HTTP.

Tag: En documentos SGML (XML, HTML...), un fragmento de texto que describe una unidad de datos o un elemento.

URI: Identificador exclusivo de forma global de un recurso abstracto o físico. Una URL es un tipo de URI que especifica el protocolo de recuperación (http o https para aplicaciones web) y la ubicación física de un recurso (nombre del host y ruta relativa al host).

12 BIBLIOGRAFÍA

12.1 Servidor de Aplicaciones

- Documentación oficial TOMCAT 5.5
 - Documentation Index: <http://tomcat.apache.org/tomcat-5.5-doc/index.html>
 - Release Notes: <http://tomcat.apache.org/tomcat-5.5-doc/RELEASE-NOTES.txt>
 - The Apache Tomcat Connector: <http://tomcat.apache.org/connectors-doc/>
 - Guía de inicio rápido de NetBeans 5.0 para aplicaciones web: http://www.netbeans.org/kb/50/quickstart-webapps_es.html
 - Creating and Running JSP, Servlets and HTML with Netbeans & Tomcat: http://cit.wta.swin.edu.au/cit/subjects/CITP0014/tutorials/netbeans/tomcat/Running_Tomcat_from_Netbeans.html

12.2 Struts

- Documentación Oficial Struts 1.2
 - Documentation Index: <http://struts.apache.org/1.x/index.html>
 - User Guide: <http://struts.apache.org/1.x/userGuide/index.html>
 - FAQs: <http://struts.apache.org/1.x/faqs/index.html>
- Patrones y Struts
 - Design Patterns for the Struts Framework: http://cpd.ogi.edu/seminars04/tyhurstseminar_files/v3_document.htm
 - Struts 1.1 Controller UML diagrams: <http://rollerjm.free.fr/pro/Struts11.html>
- Libros
 - Struts: The Complete Reference, 2nd Edition (Complete Reference Series): http://www.amazon.com/Struts-Complete-Reference-2nd/dp/0072263865/ref=pd_bbs_sr_1/103-7160899-3645428?ie=UTF8&s=books&qid=1179657937&sr=8-1
 - Programming Jakarta Struts, 2nd Edition: http://www.amazon.com/Programming-Jakarta-Struts-Chuck-Cavaness/dp/0596006519/ref=pd_bbs_sr_3/103-7160899-3645428?ie=UTF8&s=books&qid=1179657937&sr=8-3

12.3 Netbeans

- Tutoriales
 - NetBeans Modules and Rich-Client Applications Learning Trail: <http://platform.netbeans.org/tutorials/>
 - Getting Started With the NetBeans IDE Tutorial, Part 1: http://java.sun.com/developer/onlineTraining/tools/netbeans_part1/
- Netbeans + Struts
 - Introduction to the Struts Web Framework: <http://www.netbeans.org/kb/50/quickstart-webapps-struts.html>

12.4 Hibernate

- Documentación Oficial
 - Documentation Overview: <http://www.hibernate.org/5.html>
 - HIBERNATE - Relational Persistence for Idiomatic Java: http://www.hibernate.org/hib_docs/v3/reference/en/html/
- Tutoriales
 - HIBERNATE - Introduction to Hibernate: http://www.allaplabs.com/hibernate/introduction_to_hibernate.htm
 - Persistencia de Objetos Java: El Camino hacia Hibernate: <http://www.programacion.net/java/tutorial/hibernate/>
 - Hibernate Tutorial - The Road to Hibernate: <http://www.gloegl.de/5.html>
- Libros
 - Java Persistence with Hibernate: http://www.amazon.com/Java-Persistence-Hibernate-Christian-Bauer/dp/1932394885/ref=pd_bbs_sr_1/103-7160899-3645428?ie=UTF8&s=books&qid=1179659502&sr=8-1
 - Hibernate in Action (In Action series): http://www.amazon.com/Hibernate-Action-Christian-Bauer/dp/193239415X/ref=pd_bbs_sr_3/103-7160899-3645428?ie=UTF8&s=books&qid=1179659502&sr=8-3
 - Pro Hibernate 3 (Expert's Voice): http://www.amazon.com/Pro-Hibernate-3-Experts-Voice/dp/1590595114/ref=pd_bbs_sr_4/103-7160899-3645428?ie=UTF8&s=books&qid=1179659502&sr=8-4
- Hibernate + Struts
 - First steps using Struts and Hibernate: <http://www.laliluna.de/struts-hibernate-integration-tutorial-en.html>

- Hibernate and Struts:
<http://www.java2s.com/Code/Java/J2EE/HibernateandStruts.htm>
- Hibernate + Netbeans
 - First steps using Struts and Hibernate:
<http://www.netbeans.org/kb/articles/hibernate-javaee.html>

13 ANEXO A: ÍNDICE DE IMÁGENES

Imagen 1: Actor Administrador Principal.....	4
Imagen 2: Actor Creador de Contenidos	4
Imagen 3: Actor Usuario	4
Imagen 4: Proceso de añadir un Detalle de Mapeo	4
Imagen 5: Secuencia del Action AnyadeProductoStep2	4
Imagen 6: Esquema Apache / Tomcat.....	4
Imagen 7: Aspecto general del Entorno de Desarrollo	4
Imagen 8: Editor de Persistence.xml para Hibernate	4
Imagen 9: Diagrama de Clases de Base de Datos.....	4
Imagen 10: Diagrama de Clases de Struts	4
Imagen 11: Diagrama de Clases de Librería	4
Imagen 12: Diagrama de Base de Datos.....	4
Imagen 13: Interfaz de Acceso	4
Imagen 14: Pantalla de Inicio.....	4
Imagen 15: Administración de Usuarios	4
Imagen 16: Edición de Usuarios	4
Imagen 17: Listado de Proveedores	4
Imagen 18: Edición de Proveedores.....	4
Imagen 19: Alta de Proveedores	4
Imagen 20: Añadir mapeo a un Proveedor.....	4
Imagen 21: Añadir Campos a un Mapeo	4
Imagen 22: Añadir Campos a un Mapeo	4
Imagen 23: Visualizar página de producto ejemplo.....	4
Imagen 24: Visualización de código	4
Imagen 25: Añadir campos a un Mapeo	4
Imagen 26: Campos Mapeados.....	4
Imagen 27: Añadir Producto	4
Imagen 28: Introducir la URL del producto	4
Imagen 29: Producto parseado.....	4

14 ANEXO B: INSTALACIÓN

La instalación y puesta en funcionamiento de la aplicación consta de 3 pasos, para los que se provee todo el material necesario.

14.1 Aplicación web

El primero es la instalación de los ficheros de la aplicación, para lo que se provee un WAR (incluido en el fichero "areyero_producto.zip"), llamado "TFC.war", que incluye todo lo necesario para la puesta en funcionamiento de la misma. Este WAR no incluye las librerías de terceros, (Struts, Hibernate, MySQL, etc...), que pueden ser redundantes con la instalación ya existente en el servidor y hacen que el tamaño del mismo se dispare. La descarga de las librerías, en caso necesario, se realizará de las siguientes webs:

- Hibernate Core 3.2: http://sourceforge.net/project/showfiles.php?group_id=40712&package_id=127784&release_id=509407
- Hibernate Annotations 3.2: http://sourceforge.net/project/showfiles.php?group_id=40712&package_id=139933
- Hibernate Entity Manager 3.2: http://sourceforge.net/project/showfiles.php?group_id=40712&package_id=156160
- Hibernate Validator: http://sourceforge.net/project/showfiles.php?group_id=40712&package_id=225206
- Ant 1.6.5: <http://archive.apache.org/dist/ant/binaries/apache-ant-1.6.5-bin.tar.gz>
- Struts 1.2.7: <http://archive.apache.org/dist/struts/struts-1.2.7/struts-1.2.7.zip>
- MySQL Connector/J 5.0: <http://dev.mysql.com/downloads/connector/j/5.0.html>

Se pone también a disposición del usuario otro WAR ("areyero_Full.zip"), que contiene el producto completo con todas las librerías. Este WAR se puede descargar de la siguiente dirección WEB (ocupa 11 MB, demasiado para la aplicación de entrega de trabajos de la UOC):

http://72.232.183.29/areyero_Full.zip

Se adjuntan también los fuentes de los beans y clases de apoyo, en un fichero llamado "TFC-SRC.zip", incluido dentro de "areyero_producto.zip".

La instalación de este WAR depende del servidor de aplicaciones, en el caso de TOMCAT es tan sencillo como dejarlo en el directorio de webapps y reiniciar el servidor, el propio TOMCAT creará los directorios necesarios.

14.2 Base de datos

La estructura de la base de datos, de nombre "tfc", que la aplicación utiliza se adjunta en el fichero "tfc.sql". La tabla "Usuarios" incluye el usuario "admin" de password "admin". También se incluye el proveedor "Game.es" y un ejemplo de mapeo para importar datos de las fichas de videojuegos.

Este fichero de datos está creado por y para una base de datos MYSQL, aunque la adaptación a cualquier otra (Oracle, SQL Server...) sería tan sencilla como cambiar los tipos de campos a los de la nueva base de datos y corregir las diferencias que se encuentren en la sintaxis de los CREATE TABLE.

La instalación por defecto incluye además un fichero "test.sql" con los datos necesarios para importar en la base de datos recién creada un ejemplo de importación de productos de un proveedor externo en una base de datos de nombre "TEST" sobre una tabla llamada "JUEGOS".

Para la puesta en producción de la aplicación este fichero de ejemplo se debe obviar, ya que su uso sólo es interesante como demostración de funcionamiento de la aplicación.

Estos dos ficheros .sql mencionados se encuentran dentro de "areyero_producto.zip".

14.3 Ficheros de Configuración

Los ficheros de configuración "**hibernate.cfg.xml**", incluido en el directorio "WEB-INF\classes" y el "persistence.xml", incluido en "WEB-INF\classes\META-INF" llevan la configuración de acceso a la base de datos, configurada en este caso para "jdbc:mysql://localhost:3306/tfc" y usuario "root" sin password, aunque se puede cambiar para adecuarlo al host, puerto y usuario del sistema en el que se instala.

También hay que modificar los parámetros de acceso a base de datos para las clases que no utilizan persistencia / Hibernate y acceden directamente a través de JDBC (las que tienen que obtener los metadatos de las tablas y las que crean el INSERT dinámico y lo ejecutan). Todas estas clases utilizan el fichero de propiedades "**tfc.properties**" que se encuentra en "WEB-INF\classes".

14.4 Versiones de productos

La aplicación debería funcionar en cualquier servidor de aplicaciones/motor de Servlets que cumpla la especificación 1.4 de J2EE, así como un MySQL 4.1 o superior. Como referencia hacer constar que el proyecto ha sido desarrollado con las siguientes herramientas/servidores:

- Servidor de aplicaciones TOMCAT 5.5.20
- JDK 1.6.0

- Base de Datos MySQL 4.1.22
- Entorno de desarrollo Netbeans 5.5
- Framework MVC: Struts 1.2.9
- API de Persistencia: Hibernate Core 3.2.2