
SITIO WEB DE SUBASTAS
Práctica JAVA POJO Tapestry+Spring+Hibernate

Cedrón Santaefemia, Francisco Abel
López Rivas, Nuria

Integración de Sistemas
Área de Ingeniería Telemática
Universidad de A Coruña
Curso 2013/14

Índice

1. Arquitectura global.	1
2. Modelo.	2
2.1. Clases persistentes.	2
2.2. Interfaces de los servicios ofrecidos por el modelo.	3
3. Interfaz gráfica.	7
4. Servicios XML.	11
5. Partes opcionales.	13
5.1. Implementación de AJAX.	13
5.2. Pruebas funcionales contra la interfaz Web.	15
5.2.1. Registro, cambio de contraseña y autenticación con la nueva contraseña.	16
5.3. Realizar una apuesta sobre un producto concreto.	16
5.4. Ejecución de las pruebas funcionales contra la interfaz Web.	17
6. Compilación e instalación.	17
7. Problemas conocidos.	19

1. Arquitectura global.

A continuación se muestran los principales paquetes del diseño.

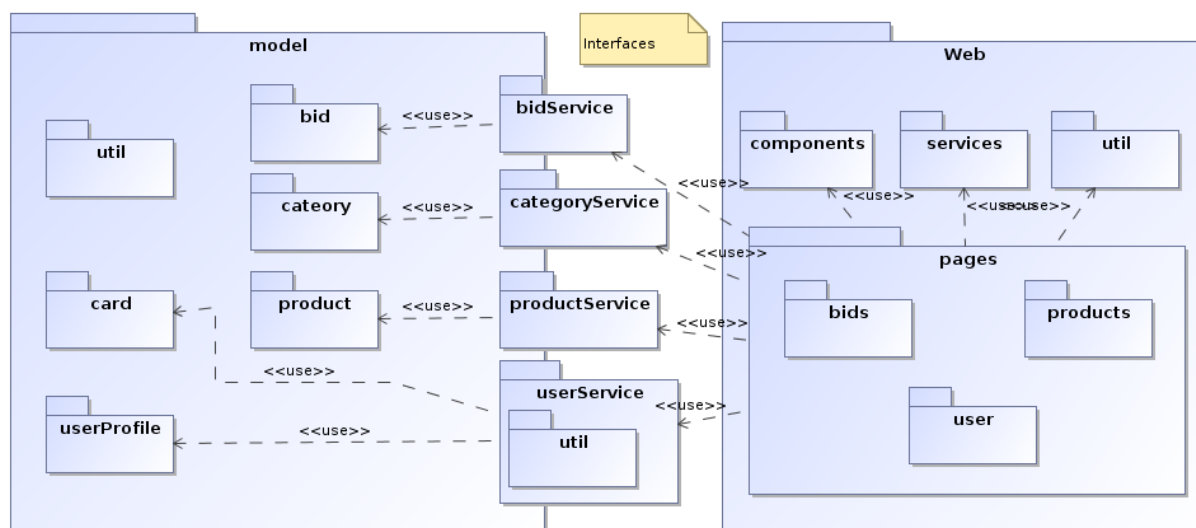


Figura 1: Paquetes del diseño.

Describiendo brevemente la función de cada uno de los paquetes:

1. **model:** Paquete principal de la capa modelo y contenedor de los demás paquetes. Los paquetes situados ligeramente fuera del paquete principal, correspondientes a los servicios, actúan como interfaces entre ambos paquetes generales. Cada entidad persistente dispone de su propio paquete, el cual contiene los DAO's para interactuar con la base de datos.
La mayoría de los paquetes son autodescriptivos mediante su nombre. El paquete *util* contiene el fichero *GlobalNames*, el cual define la ruta al fichero de configuración de Spring.
2. **web:** Paquete principal de la capa web que contiene los siguientes paquetes:
 - *components*: Para los elementos comunes de las páginas.
 - *services*: Contiene las políticas y los filtros que se utilizan en la aplicación web.
 - *util*: Contiene ficheros para la definición de la sesión del usuario, el uso de cookies y las configuraciones para el uso de tablas, paginación y listados en los *.tml*.
 - *pages*: Contiene los paquetes que implementan las páginas que se muestran al usuario.
 - *bids*: Control del formulario de apuestas.
 - *products*: Control para listar productos, mostrar sus detalles y del formulario para añadirlos.

- user: Control de los formularios de registro, autenticación, cambio de contraseña, añadir una tarjeta o modificarla y modificar los datos de registro. Además contiene una página que muestra todos los datos del usuario que se puedan visualizar.

2. Modelo.

2.1. Clases persistentes.

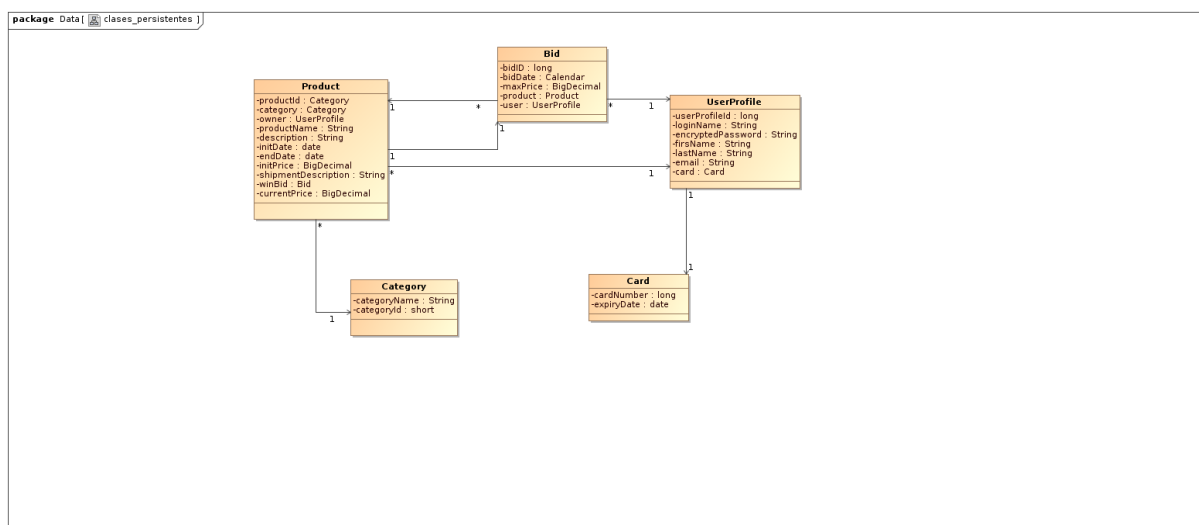


Figura 2: Clases persistentes.

2.2. Interfaces de los servicios ofrecidos por el modelo.

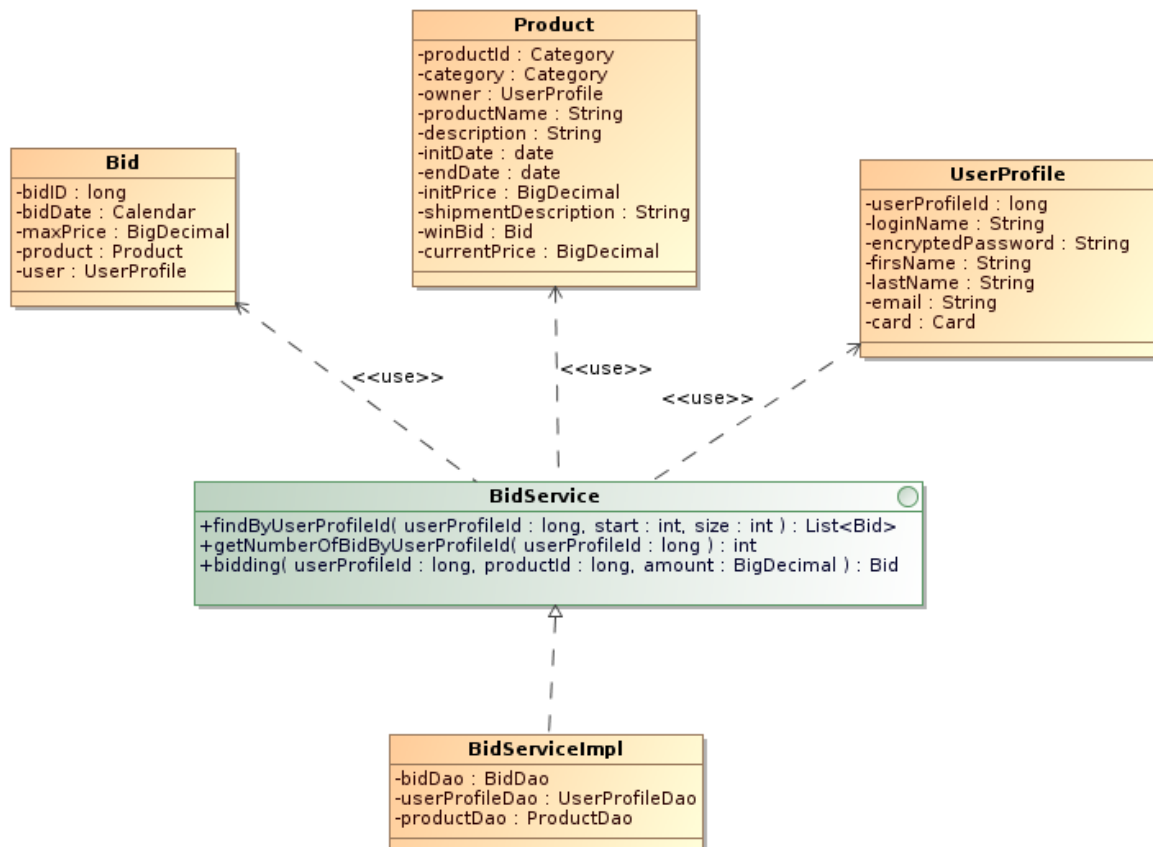


Figura 3: Interfaz del servicio de apuestas.

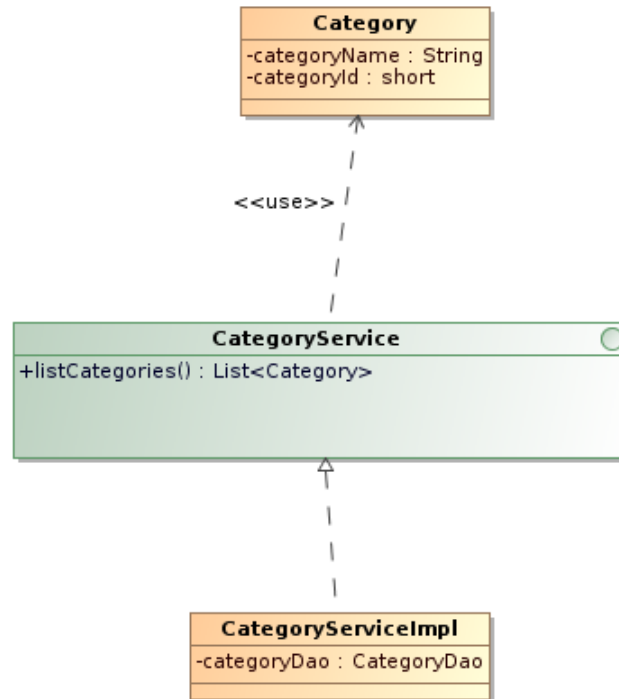


Figura 4: Interfaz del servicio de categoría.

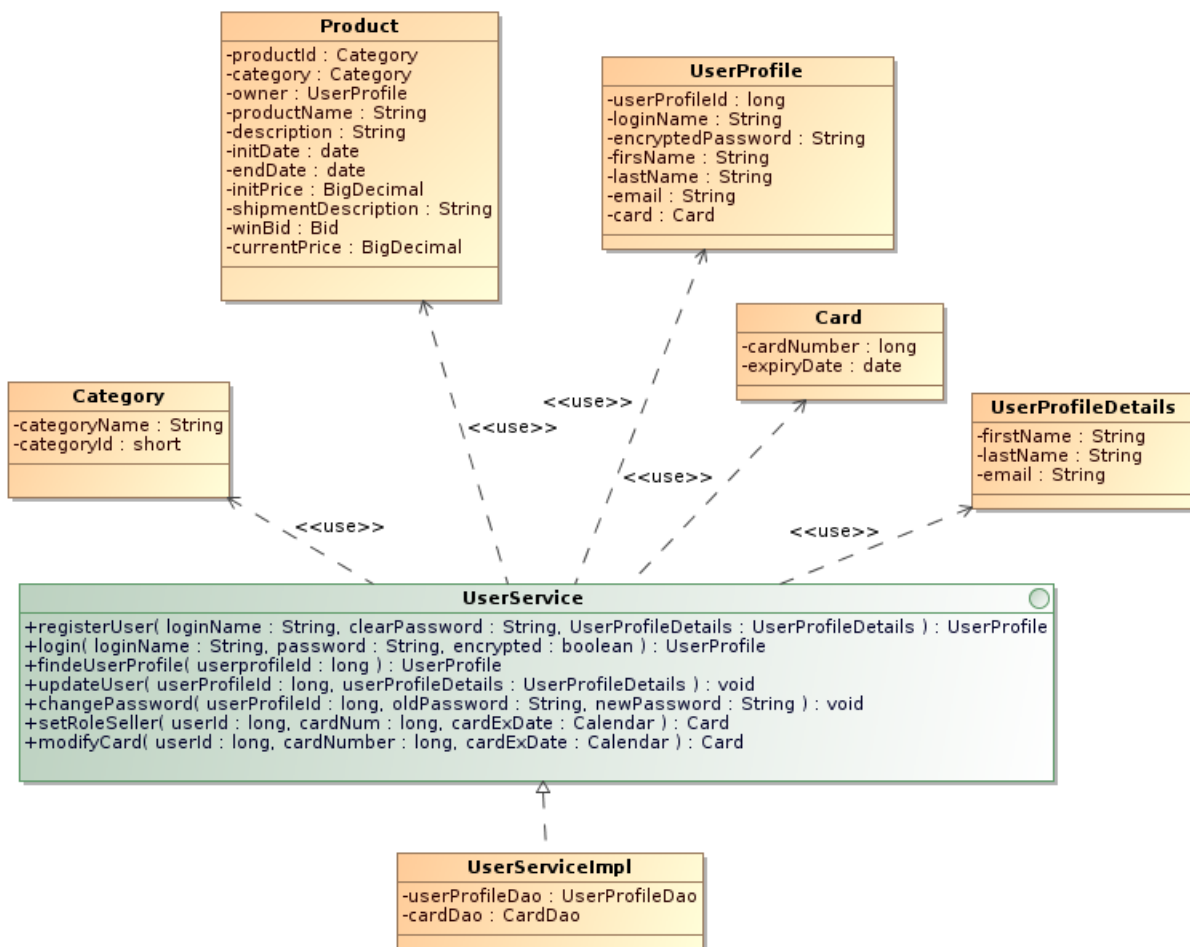


Figura 5: Interfaz del servicio de usuario.

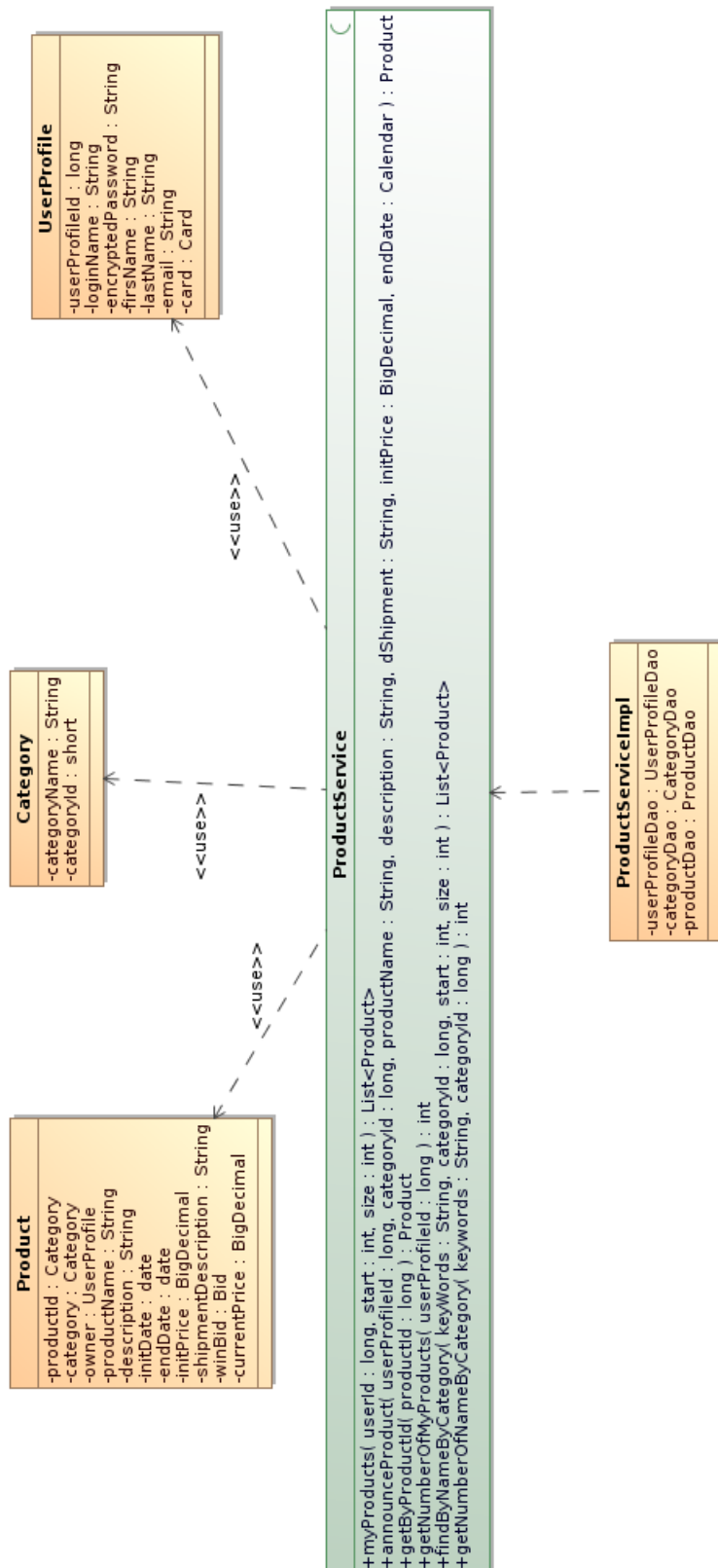


Figura 6: Interfaz del servicio de productos.

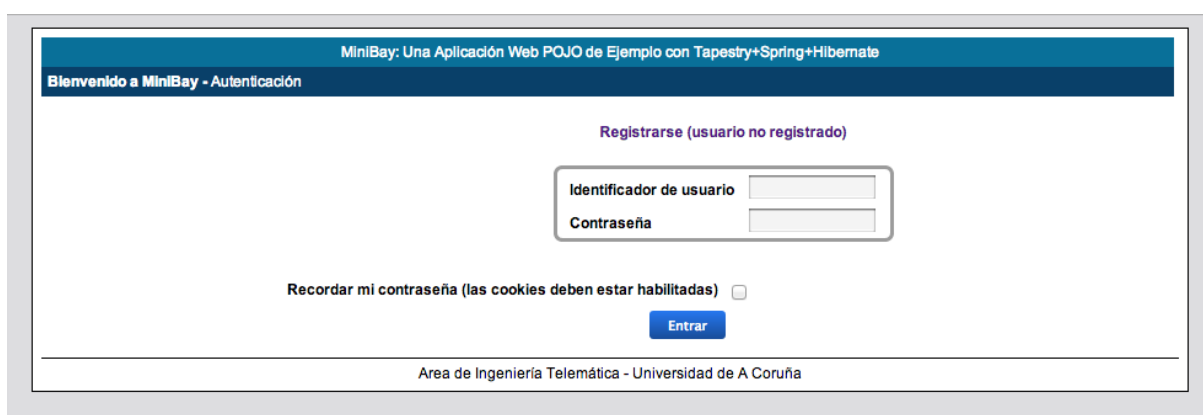
La interfaz lógica de *card* no fue creada ya que su funcionalidad se incluye dentro de la interfaz de *userService*. De las otras 4 interfaces podemos decir:

- *bidService* permite la búsqueda de las pujas de un usuario y la funcionalidad para realizar la propia puja.
- *categoryService* otorga acceso a la lista de categorías.
- *productService* permite buscar productos por palabra clave o por categoría. También implementa la lógica para añadir productos.
- *userService* trata todo lo relacionado con los datos del usuario, incluyendo la creación de una tarjeta de crédito.

3. Interfaz gráfica.

La interfaz gráfica consta de una capa común (Layout) y una serie de páginas que se explican a continuación.

La página inicial consta de una barra de búsqueda, ya que se pueden ver los productos disponibles para pujar tanto si se está autenticado como si no.



MiniBay: Una Aplicación Web POJO de Ejemplo con Tapestry+Spring+Hibernate

Bienvenido a MiniBay - Autenticación

[Registrarse \(usuario no registrado\)](#)

Identificador de usuario

Contraseña

Recordar mi contraseña (las cookies deben estar habilitadas) ☐

[Entrar](#)

Area de Ingeniería Telemática - Universidad de A Coruña

Figura 7: Formulario de autenticación.

Para poder pujar es necesario estar autenticado, a esto se llega pulsando en el enlace “Autenticar” en la barra de menú. En caso de no estar registrado, en la misma página de autenticación hay un enlace que lleva a una página de registro.

Una vez autenticado, se accede a la página principal.



Figura 8: Página principal.

En la parte izquierda de la 8 pueden verse una serie de enlaces a los distintos servicios de la página. El primero de ellos es un enlace a la misma página que se muestra en dicha imagen, siendo útil cuando se quiere volver ahí desde cualquiera de las otras páginas.

El segundo enlace lleva a una página con diferentes tipos de información del usuario.

Figura 9: Cuenta personal del usuario.

En la 9 pueden verse 3 formularios. El primero de ellos muestra la información del

usuario, y al mismo tiempo permite modificarla, escribiendo el nuevo valor en el campo que se desee cambiar y pulsando en “Actualizar”.

El segundo formulario sirve para cambiar la contraseña de acceso al sitio. Por último, el tercero muestra los datos de la tarjeta de crédito propiedad del usuario, y de igual forma que con la información personal, también se pueden modificar estos valores. En caso de que el usuario no haya añadido ninguna tarjeta los campos se mostrarán en blanco, y deberán ser rellenados para añadir una.

El motivo de tener estos 3 formularios aquí es mantener el número de página al mínimo.

El tercer enlace de la figura 8 lleva a un listado de las apuestas realizadas por el usuario.



The screenshot shows the MiniBay web application interface. At the top, a blue header bar contains the text "MiniBay: Una Aplicación Web POJO de Ejemplo con Tapestry+Spring+Hibernate". Below this, a dark blue bar displays "Hola Francisco Abel - Mis apuestas". On the left, a vertical menu lists: Inicio, Mi cuenta, Mis apuestas, Mis productos, Añadir producto, and Salir. The main content area features a search bar, a "Categorías" dropdown, and a "Buscar" button. Below these is a table of bets. The table has six columns: Producto, Usuario ganador, Precio actual, Fecha de la puja, Precio pujado, and Fin de puja. A single row is visible with the data: Motor Nema 17, fcedron, 10.05, 21/02/14 4:47, 100.555, and 31/03/14 4:43. At the bottom, a footer bar reads "Area de Ingeniería Telemática - Universidad de A Coruña".

Producto	Usuario ganador	Precio actual	Fecha de la puja	Precio pujado	Fin de puja
Motor Nema 17	fcedron	10.05	21/02/14 4:47	100.555	31/03/14 4:43

Figura 10: Apuestas realizadas por el usuario.

Los únicos datos que se muestran aquí son: el nombre del producto, el propietario del producto, la fecha en que se realizó la puja y la cantidad pujada. El nombre del producto es a su vez un enlace a otra página que contiene una descripción más en detalle de dicho producto.

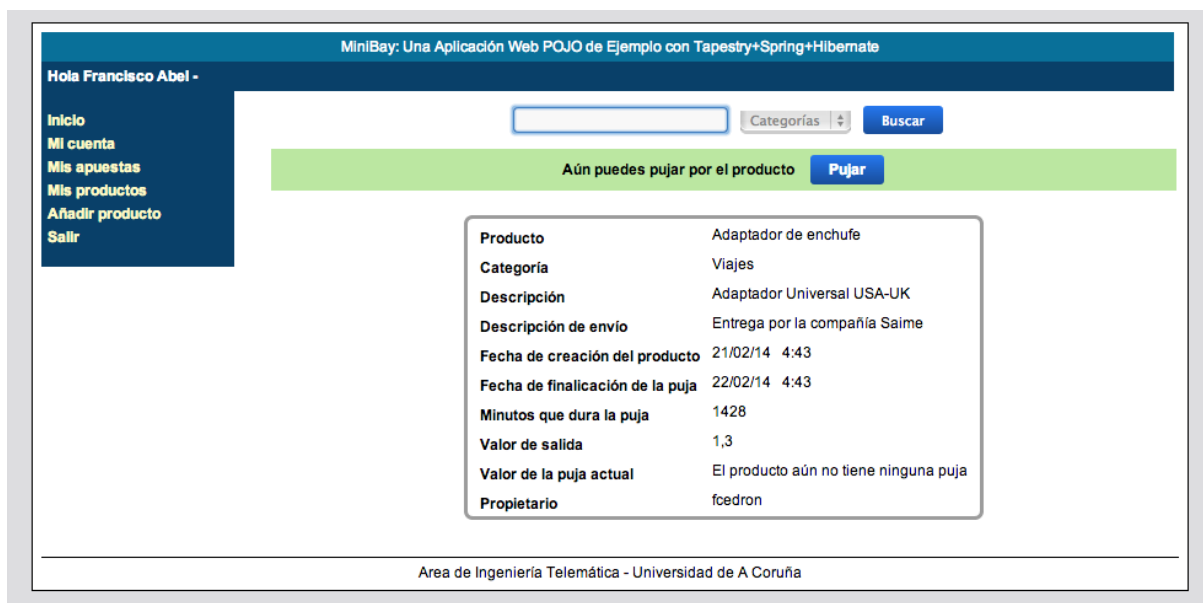


Figura 11: Detalles de un producto.

Como puede observarse en la figura 11 esta página contiene un botón que permite al usuario pujar por el producto que está consultando, además de informales si va ganando o no la puja, y aumentar el valor de ésta si así lo desea.



Figura 12: Información sobre la puja.

El siguiente enlace, siguiendo el orden, lleva a un listado de los productos añadidos por el usuario.

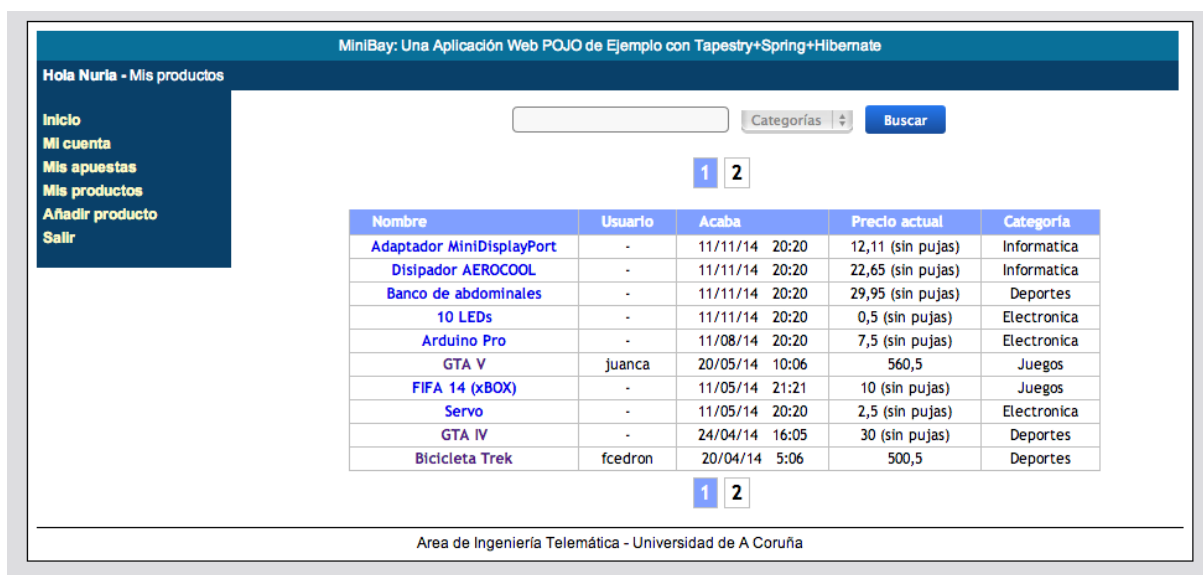


Figura 13: Productos del usuario.

Los detalles que se muestran aquí son: nombre del producto, propietario del producto, fecha y hora de finalización de la puja, valor actual en el proceso de puja y la categoría a la que pertenece. Al igual que en el caso del listado de apuestas, el nombre del producto es un enlace a la página mostrada en la figura 11. En el apartado del valor actual del producto también se muestra si se ha hecho alguna puja por él.

El penúltimo enlace lleva a un formulario para que el usuario pueda añadir productos.

Figura 14: Formulario para añadir productos.

Todo los campos son obligatorios. Una vez añadido el producto, se redirige a la página con sus detalles.

El último enlace cumple la función de “logout”, cerrando la sesión en el sitio.

4. Servicios XML.

Los formatos de la URL's de los servicios XML son los siguiente:

1. Información detallada de un producto:

<http://localhost:8080/minibay/products/xmlproductdetails?productid=X>

X representa el id del producto del cual se quieren ver los detalles.

El formato de salida consta de una etiqueta con los atributos correspondientes a los detalles del producto. Obviando los autodescriptivos, el significado es el siguiente:

- initDate: Fecha de creación del producto.
- endDate: Fecha de fin de subasta.
- initPrice: Precio que pone inicialmente el anunciante del producto.

- dShipment: Descripción del envío del producto.
- currentPrice: Valor actual del producto. Varía según se vayan haciendo pujas.
- winBid: Usuario que va ganando la subasta.

```
<product xmlns:t="http://tapestry.apache.org/schema/tapestry_5_3.xsd"
  xmlns:p="tapestry:parameter">
<t:if test="product">
  name=${product.productName}
  category=${product.category.categoryName}
  owner=${product.owner.loginName}
  initDate=${product.initDate.time}
  endDate=${product.endDate.time}
  description=${product.description}
  initPrice=${product.initPrice}
  dShipment=${product.shipmentDescription}
  currentPrice=${product.currentPrice}
  <t:if test="product.winBid">
    winBid=${product.winBid.user.loginName}
  </t:if>
<p:else>
<errorNoProduct />
</p:else>
</t:if>
</product>
```

2. Búsqueda de productos por palabra clave del nombre:

<http://localhost:8080/minibay/products/XmlListProducts?start=X&size=Y&keywords=Z>

X representa en qué producto se comienza a listar.

Y representa cuántos productos se listarán por página.

Z representa las palabras clave mediante las que se va a realizar la búsqueda.

El formato de salida consta de una única etiqueta que contiene los atributos relevantes del producto mostrado. Habrá una línea por cada producto en la lista.

```
<products total="${total}" show="${show}"
  xmlns:t="http://tapestry.apache.org/schema/tapestry_5_3.xsd">
<t:loop t:source="list" t:value="Product">
  <product
    id="${product.productId}"
    category="${product.category.categoryName}"
    name="${product.productName}"
    seller="${product.owner.loginName}"
    ends="${endDate}"
    minutesToEnd="${minutesToEnd}"
    price="${price}"
  />
</t:loop>
</products>
```


5. Partes opcionales.

5.1. Implementación de AJAX.

En este apartado explicaremos como se ha realizado la parte de optativa que se corresponde con AJAX.

Hemos decidido implementar la funcionalidad de AJAX en la página que muestra los detalles del usuario, la cual se muestra en la figura 9. El motivo de insertarlo aquí fueron los formularios. Si un usuario comete un error en la modificación de alguno de los datos, por ejemplo, si en un cambio de contraseña, al reescribirla no coincide con la primera inserción, o si escribe mal la vieja. En estos casos se usa AJAX para que se muestre el mensaje de error sin necesidad de recargar la página entera.

Además de esto, si se modifican correctamente cualquiera de los datos presentes, solo se recargará el formulario afectado. El siguiente código muestra la implementación de estos casos.

```
Object onSuccessFromChangeCard() {
    okCard = messages.get("ok-card");
    return request.isXHR() ? zoneFormCard.getBody() : null;
}
```

```
Object onFailureFromChangeCard() {
    return request.isXHR() ? zoneFormCard.getBody() : null;
}
```

```
Object onSuccessFromChangePassword() {
    CookiesManager.removeCookies(cookies);
    okPassword = messages.get("ok-pass");
    return request.isXHR() ? zoneFormPassword.getBody() : null;
}
```

```
Object onFailureFromChangePassword() {
    return request.isXHR() ? zoneFormPassword.getBody() : null;
}
```

```
Object onSuccessFromChangeInfo() throws InstanceNotFoundException {
    userService.updateUserProfileDetails(
        userSession.getUserProfileId(), new UserProfileDetails(
            firstName, lastName, email));
    userSession.setFirstName(firstName);
    okInfo = messages.get("ok-info");
}
```

```
return request.isXHR() ? zoneFormInfo.getBody() : null;
}
```

```
Object onFailureFromChangeInfo() {
return request.isXHR() ? zoneFormInfo.getBody() : null;
}
```

Para utilizar esas funciones fue necesario inyectar los componentes “Zone” y “Request” e incorporar los siguientes paquetes al fichero “MyAccount.java”:

```
import org.apache.tapestry5.corelib.components.Zone;
import org.apache.tapestry5.services.Request;

@InjectComponent
private Zone zoneFormCard;

@InjectComponent
private Zone zoneFormPassword;

@InjectComponent
private Zone zoneFormInfo;

@Inject
private Request request;
```

Podemos definir:

Zone: Las “Zone” definen el área de datos actualizable, corresponden a un bloque de la plantilla tml.

Request: Permite controlar la petición http para saber si se trata de AJAX.

En los .tml se hicieron los siguientes cambios:

```
<t:zone t:id="zoneFormInfo" id="zoneFormInfo">
<form t:type="Form" t:id="changeInfo" t:zone="^">

<t:zone t:id="zoneFormPassword" id="zoneFormPassword">
<form t:type="Form" t:id="changePassword" t:zone="^">

<t:zone t:id="zoneFormCard" id="zoneFormCard">
<form t:id="changeCard" t:type="form" t:zone="^">
```

Con eso definimos las zonas que se modificarán dinámicamente.

Este proceso también se definió para el formulario de añadir productos, dando lugar a código similar.

Además de lo anterior, también se añadió una parte de AJAX al utilizar el componente “GridDataSource” para definir las listas de productos y pujas.

En caso de que un usuario no disponga de JavaScript en su navegador, se añadió un aviso en Layout.tml

```
<noscript class="js-required">
<p class="t-red">
    ${message:javascript_required}
</p>
</noscript>
```

5.2. Pruebas funcionales contra la interfaz Web.

Las pruebas funcionales contra la interfaz Web son pruebas que comprueban el correcto funcionamiento de una aplicación Web usando directamente su interfaz Web. Se han implementado dos casos de prueba, los cuales se corresponden con dos secuencias de navegación que representan las funcionalidades que la aplicación ofrece al usuario. Los casos de prueba implementados son:

- Registro, cambio de contraseña y autenticación con la nueva contraseña.
- Realizar una apuesta sobre un producto concreto.

La implementación de estos dos casos de prueba se ha realizado en un nuevo proyecto. La razón por lo que se ha hecho así es porque en el otro proyecto están los test de la capa modelo que se ejecutan con más frecuencia, mientras que los test contra la interfaz no están en ese caso y tardan bastante más en llevarse a cabo. De hecho, si este tipo de test contra la interfaz Web fuese muy numeroso, incluso podría ser inviable ejecutarlos con la frecuencia necesaria.

Para la implementación de los test automáticos de la interfaz web se ha utilizado el componente *WebDriver* de la herramienta *Selenium* junto con JUnit. Estas herramientas abren automáticamente un navegador para ejecutar cada caso de prueba, es decir, ejecutan la secuencia de navegación del caso de prueba y hacen las comprobaciones necesarias para verificar el correcto funcionamiento de la aplicación.

A continuación se detalla la secuencia de navegación realizada en cada caso de prueba.

5.2.1. Registro, cambio de contraseña y autenticación con la nueva contraseña.

La secuencia de navegación utilizada para llevar a cabo el caso de prueba es la siguiente:

1. Acceder a la página principal de la aplicación.
2. Hacer clic en el enlace “Autenticarse”.
3. Hacer clic en el enlace “Registrarse (usuario no registrado)”.
4. Cubrir el formulario de registro.
5. Hacer clic en el botón de “Registrar”.
6. Hacer clic en el enlace “Mi cuenta”.
7. Rellenar el formulario destinado para cambiar la contraseña.
8. Hacer clic en el botón de “Actualizar”.
9. Hacer clic en el enlace “Salir”.
10. Hacer clic en el enlace “Autenticarse”.
11. Cubrir el formulario de inicio de sesión.
12. Hacer clic en el botón de “Entrar”.

5.3. Realizar una apuesta sobre un producto concreto.

Para llevar a cabo esta prueba se usarán los datos que hay en el proyecto de las pruebas funcionales contra la interfaz web dentro de un script de sql. Así pues suponiendo que ya existen esos datos la navegación realizada es la que se cuenta a continuación.

1. Acceder a la página principal de la aplicación.
2. Acceder a la barra de búsqueda para buscar un producto.
3. Pulsar en el botón “Buscar”.
4. Hacer clic en el enlace del producto.
5. Hacer clic en el enlace de “Autenticarse” (con un usuario ya creado).
6. Rellenar el formulario de inicio de sesión.

7. Hacer clic en el botón de “Entrar”.
8. Realizar el paso 2 de nuevo.
9. Realizar el paso 3.
10. Realizar el paso 4.
11. Pulsar sobre el enlace “Pujar”.
12. Rellenar el formulario destinado para pujar con una valor que no sea inferior a la puja mínima.
13. Hacer clic en el botón “Realizar puja”.
14. Pulsar el enlace “Mis apuestas”.
15. Pulsar el enlace sobre la apuesta que se acaba de realizar.

5.4. Ejecución de las pruebas funcionales contra la interfaz Web.

Para poder ejecutar las pruebas automatizadas es necesario tener instalado en el equipo el navegador web Mozilla Firefox y realizar los siguientes pasos:

1. Ejecutar la aplicación de principal en un servidor de aplicaciones.
2. Situar en el directorio del proyecto y ejecutar las sentencias sql para acabar ejecutando los test.

```
$cd minibay-pruebas-interfaz-web
```

```
$mvn sql:execute
```

```
$mvn test
```

6. Compilación e instalación.

Para la compilación de la aplicación se utilizó el entorno *maven*.

En primer lugar será necesario iniciar el servicio *mysql*, ya que de no hacerlo cualquier operación realizada devolvería un error.

```
$mysqld --defaults-file=$HOME/.my.cnf
```

A continuación se hará **\$mvn sql:execute**. Este comando ejecuta los scripts de creación de tablas e inserción de datos. Lógicamente, las bases de datos sobre las que se

va a trabajar deben existir creándolas manualmente antes de ejecutar dicho script. En caso de que ya hubiese tablas creadas en dichas bases de datos, debe tenerse presente que la ejecución de esto borrará dichas tablas y las volverá a crear.

Si se quieren realizar los test unitarios de la capa modelo la orden a ingresar será **\$mvn test**.

Para poder visualizar la página web en un navegador usaremos *apache tomcat*. En primer lugar será necesario ejecutar **\$mvn sql:execute package**. Esto creará un fichero *.war* que habrá que copiar en el directorio “webapps” de tomcat.

A continuación habrá que configurar tomcat para la ejecución de nuestra aplicación. Para ello hay que añadir el nuevo recurso de la base de datos. Dentro del directorio de tomcat editaremos el fichero *conf/server.xml* y añadiremos

```
<Resource name="jdbc/pojo-examples-ds"
    auth="Container"
    type="javax.sql.DataSource"
    driverClassName="com.mysql.jdbc.Driver"
    url="jdbc:mysql://localhost/pojo"
    username="pojo"
    password="pojo"
    maxActive="4"
    maxIdle="2"
    maxWait="10000"
    removeAbandoned="true"
    removeAbandonedTimeout="60"
    logAbandoned="true"
    validationQuery="SELECT COUNT(*) FROM PingTable"/>
```

y en *conf/context.xml*

```
<ResourceLink name="jdbc/pojo-examples-ds" global="jdbc/pojo-examples-ds"
    type="javax.sql.DataSource"/>
```

Estos 2 cambios fueron sacados del fichero “README.txt” y son oportunos para nuestra aplicación. En un entorno de desarrollo real habría que tener cuidado con la configuración del servidor.

Una vez hecho lo anterior, y todavía dentro del directorio de tomcat, ejecutaremos el script *bin/startup.sh* para lanzar el servidor. Si todos los pasos anteriores se realizaron correctamente ahora sería posible abrir nuestra aplicación en un navegador introduciendo la url **localhost:8080/minibay**.

Para detener la ejecución de tomcat bastaría con ejecutar el script “bin/shutdown.sh”.

7. Problemas conocidos.

La interfaz *categoryService* sería prescindible, y su funcionalidad podría integrarse con la interfaz de *productService* ya que es el único lugar donde se hace uso de ella.