Memoria AuctionHouse

Autores
Breixo Camiña Fernández - breixo.camina@udc.es
David Torres Requeijo - david.torres@udc.es
Iván González Cotelo - ivan.gonzalez.cotelo@udc.es

Programación Avanzada – Curso 2014/2015

Este documento recoge la memoria de la aplicación AuctionHouse. Presentaremos las principales características de nuestro sistema y la arquitectura utilizada.

1. Descripción del sistema

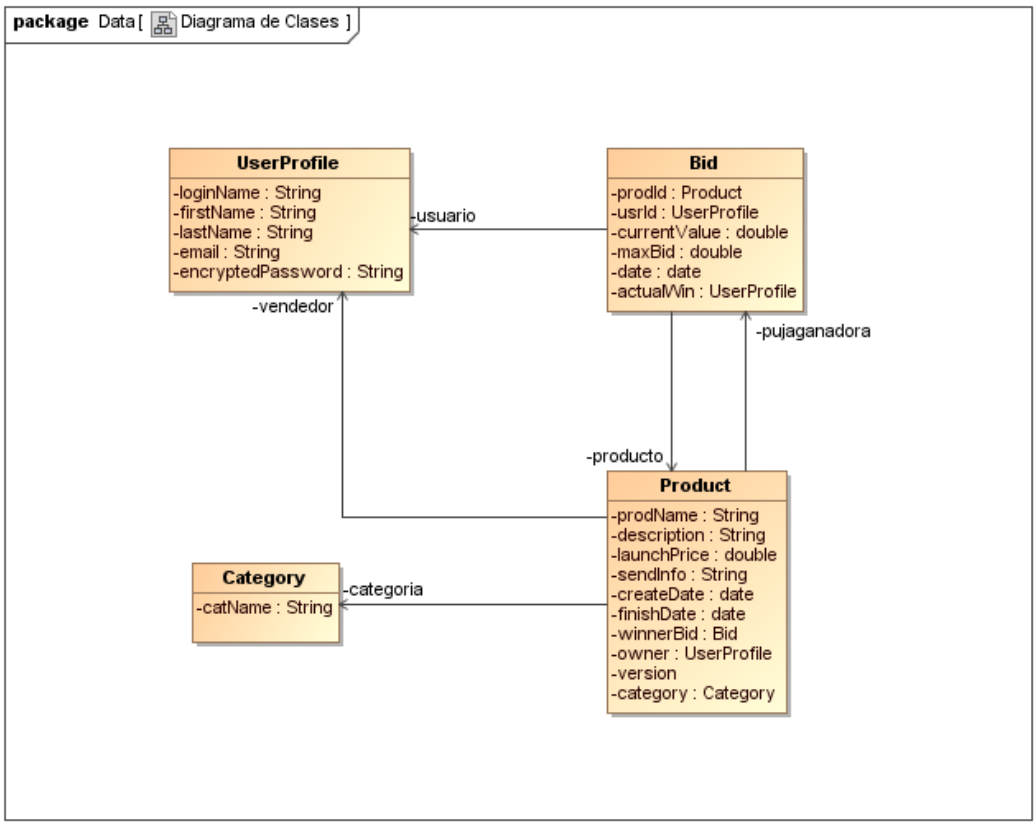
AuctionHouse es una página web de venta de productos mediante el sistema de pujas. La página puede ser visitada por cualquier usuario de la red pero para realizar una puja es necesario que el usuario esté registrado.

2. Arquitectura

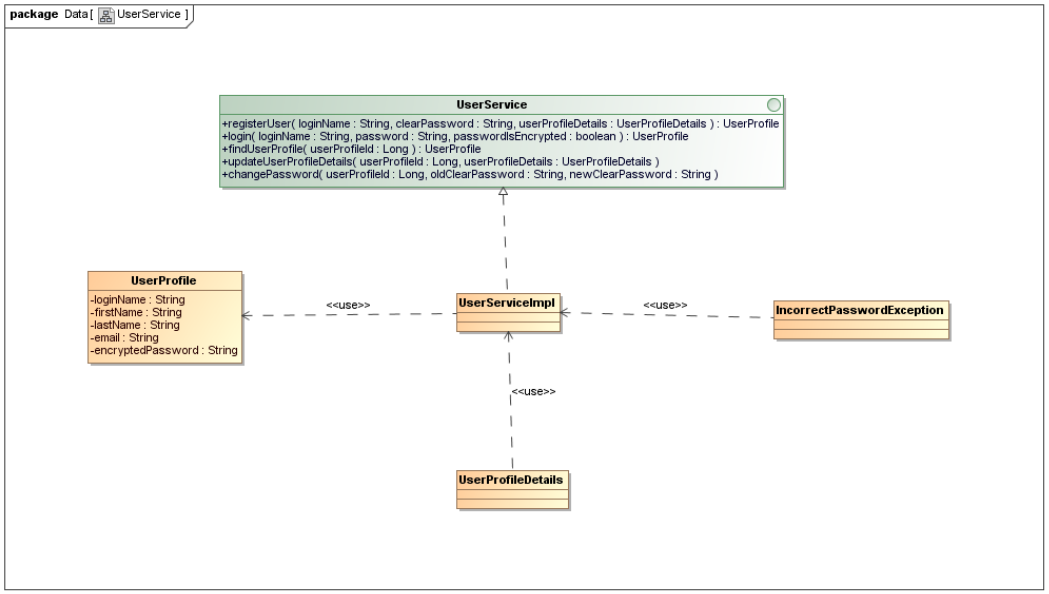
La aplicación está dividida en tres capas: modelo, servicios e IU web.
La capa modelo y la de servicios la componen los ficheros dentro de los paquetes **es.udc.pa.pa007.auctionhouse.model.***, siendo los paquetes correspondientes a los servicios los que llevan la notación ***service** y, finalmente, los ficheros que se pueden encontrar en **es.udc.pa.pa007.auctionhouse.web.*** pertenecen a la capa IU web.
La capa modelo es donde se encuentran las clases persistentes (*UserProfile, Bid, Category y Product*) con sus respectivas implementaciones de persistencia empleando DAOs (BidDao) e Hibernate (BidDaoHibernate). Para conectar la capa de servicios con la capa modelo se usa la implementación de Spring.
Para la capa IU web se implementa usando Tapestry.

3. Diagramas

3.1. Clases persistentes de la Capa Modelo

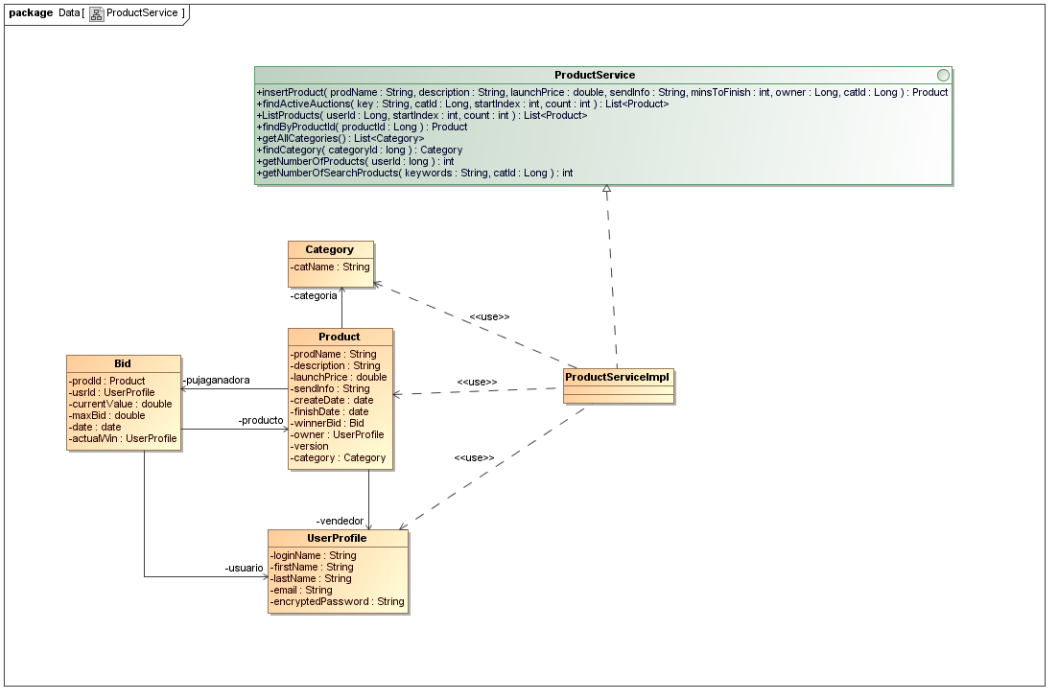


3.2. Interfaz de los servicios de la clase UserProfile



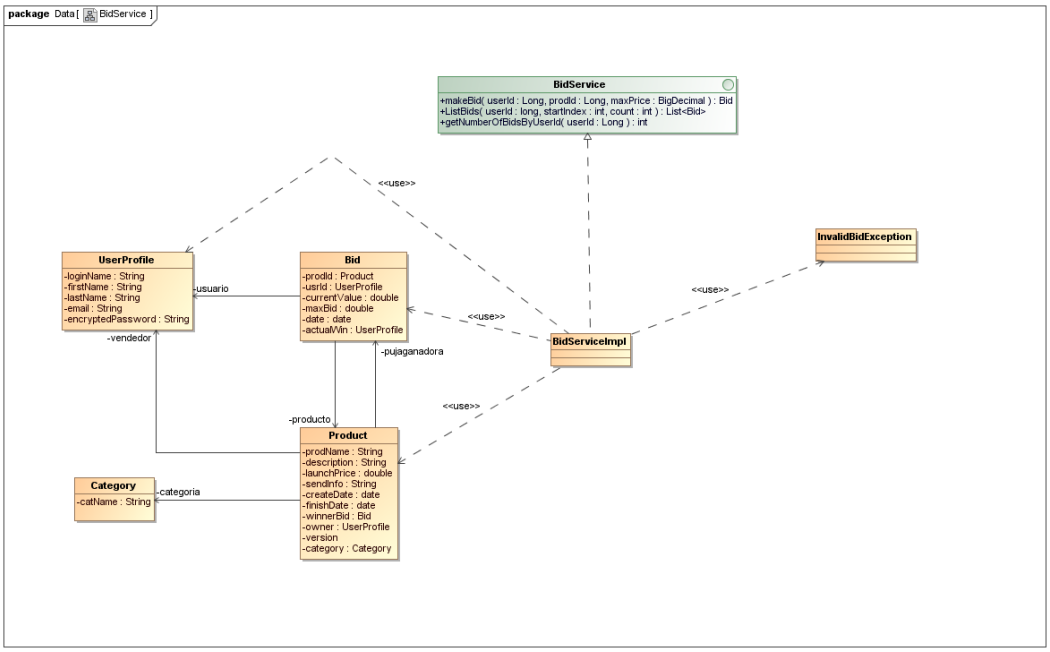
La interfaz UserService nos proporciona los servicios referentes al usuario tales como registrar usuario, loguearse, cambiar contraseña, buscar usuario y actualizar detalles del usuario.

3.3. Interfaz de los servicios de la clase Product



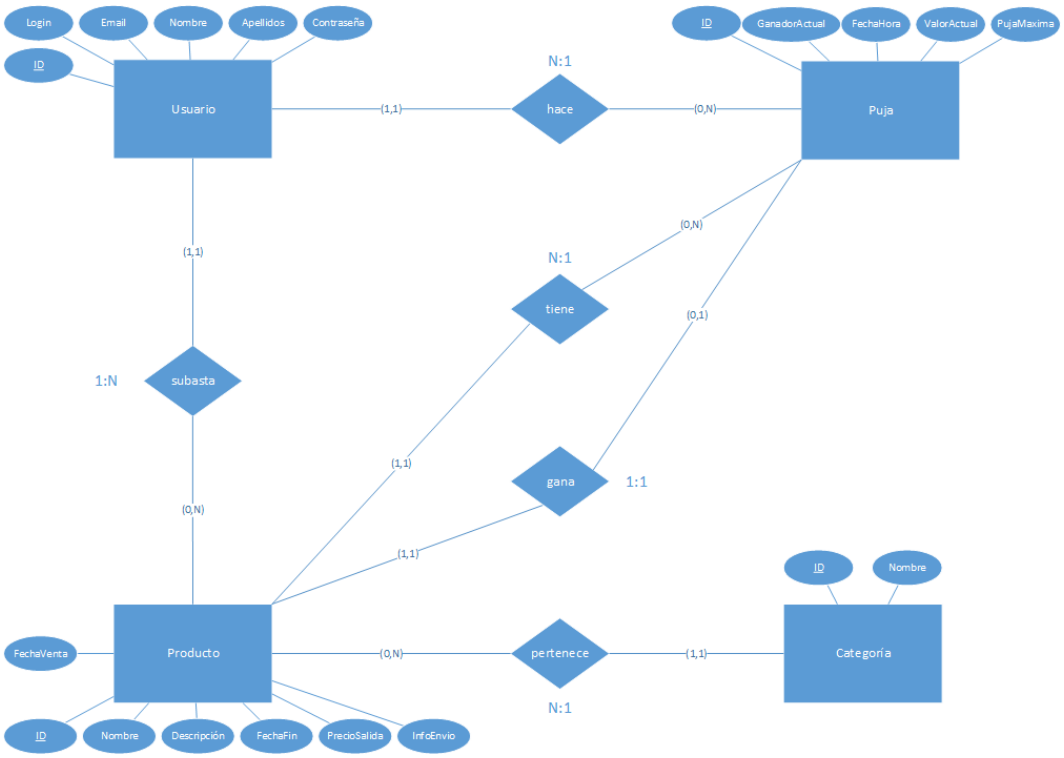
La interfaz ProductService proporciona los servicios referentes a productos tales como añadir producto, buscar productos todavía en venta, listar productos, buscar producto por identificador, mostrar número de productos y mostrar número de resultados de la búsqueda.
También se implementan las funciones buscar categoría y mostrar todas las categorías referentes a Category por la sencillez de estas, aunque se podrían separar en un CategoryService.

3.4. Interfaz de los servicios de la clase Bid



La interfaz BidService proporciona los servicios referentes a pujas: hacer una puja, listar las pujas y número de pujas realizadas por un usuario indicado.

3.5. Modelo Entidad Relación de la BD



A pesar de ser innecesario, para facilitar la creación de nuestra base de datos, hemos empleado este modelo Entidad Relación, que finalmente ha resultado ser muy similar al diagrama de clases persistentes.

4. Interfaz Gráfica

El diseño de la página web está basado en Bootstrap, para minimizar el uso de HTML y CSS (no es la finalidad de la práctica) manteniendo un diseño web Responsive.

Las páginas tienen un diseño básico caracterizado por un header, donde está el menú de acciones del usuario, y un footer con información acerca de los realizadores de la práctica.

Los errores se muestran mediante funciones básicas de Tapestry tanto en formularios como en las páginas de puja, login, inserción de productos y registro de usuarios.

Hemos implementado una funcionalidad opcional, redirigimos a un usuario no autenticado que desea realizar una puja sobre un producto, es decir, primero lo redirigimos hacia la pagina de login, y una vez se autentica se le muestra ya el producto por el que quería pujar ofreciéndole la posibilidad de hacerlo.

5. Trabajos tutelados

5.1. AJAX

Para la realización de AJAX nos hemos centrado en dos casos típicos de AJAX:

- Autocompletado: Al realizar una búsqueda, si escribes un caracter, aparece un desplegable con las primeras 5 coincidencias de artículos disponibles en ese momento, este autocompletado no filtra por categoría. Por ejemplo, si en la búsqueda ponemos **nex**, nos aparecerá como autocompletado **Nexus 5**.

- Recarga de zona: Al recargar páginas que contengan un grid, o bien usando paginación del mismo, en lugar de recargar toda la página, solo recargamos la zona del grid que se necesita, consumiendo la cantidad mínima de recursos posible para satisfacción del usuario.

5.2. Pruebas funcionales contra la interfaz Web

Se han realizado también todas las pruebas funcionales sobre la interfaz web que nos indican en el enunciado, usando para ello Selenium, un entorno de pruebas de software para aplicaciones basadas en la web. En concreto usamos un componente de Selenium llamado WebDriver para interactuar con el explorador y JUnit para comprobar las aserciones.

6. Compilación e instalación de la aplicación

Para arrancar la aplicación, debemos primero arrancar el servicio de **mysql**, bien desde una consola con el comando **mysqld** o activando este servicio de forma automática. A continuación en otro terminal, nos tenemos que situar en la ruta del proyecto y ejecutar **mvn sql:execute package**, este comando ejecutara los archivos de creación de tablas e inserción de datos en la base de datos y generara un archivo war. Copiar dicho archivo **auctionhouse.war** situado en la carpeta **/target** a la carpeta **<<tomcat>>/webapps** y finalmente ejecutar **<<tomcat>>/bin/startup.sh**. Abrir un navegador y escribir la URL **http://localhost:8080/auctionhouse/** y ya podemos navegar en nuestra página.