



UNIVERSIDADE DA CORUÑA
FACULTADE DE INFORMÁTICA

Departamento de Tecnoloxías da Información e as Comunicacións

PROXECTO DE FIN DE CARREIRA EN
ENXEÑERÍA TÉCNICA INFORMÁTICA DE XESTIÓN

Diseño e Implementación de una
Aplicación Web Java EE con
Arquitectura MVC

Autor: *Alejandro Martínez Vieites*

Tutor: *Paula Montoto Castelao*

A Coruña, Enero 2010

**Proyecto: Diseño e Implementación de una Aplicación Web Java EE con Arquitectura
MVC**

No está permitida la reproducción total o parcial de este proyecto, ni su tratamiento informático, ni la transmisión de ninguna forma o por cualquier medio, ya sea electrónico, mecánico, por fotocopia, por registro u otros métodos, sin el permiso previo y por escrito del autor del mismo.

A mis padres.

Alejandro Martínez Vieites

Diseño e Implementación de una Aplicación Web Java EE con Arquitectura MVC

Autor: *Alejandro Martínez Vieites*

Tutor: *Paula Montoto Castelao*

Miembros del Tribunal:

Fecha de lectura:

Calificación:

RESUMEN

Título: Diseño e Implementación de una Aplicación Web Java EE con Arquitectura MVC

Autor: Alejandro Martínez Vieites

Tutor: Paula Montoto Castelao

La finalidad de este proyecto consiste en la realización de una aplicación web mediante la tecnología Java EE y su estructuración según el patrón arquitectónico Model-View-Controller (MVC). En concreto, la aplicación a desarrollar elegida ha sido un sistema de gestión de subastas con soporte de valoraciones de usuario.

Puesto que se trata de un proyecto de fin de carrera, este desarrollo tiene una finalidad subyacente, ésta no es otra que el aprendizaje por parte del alumno de las tecnologías y metodologías necesarias para llevar a cabo el desarrollo en cuestión.

La aplicación da soporte al registro de usuarios en el sistema y proporciona un método de comunicación directa entre éstos. Este punto es vital para que se puedan resolver las dudas y/o problemas que pueden surgir en cada subasta y aclarar malentendidos o descripciones de artículos poco concisas.

Los usuarios registrados pueden sacar artículos a subasta pública y pujar por los que saquen los demás de forma que, como en una subasta tradicional, el usuario que haya pujado la cantidad más alta llegada la fecha de cierre de la subasta será el ganador de la misma.

Con el objeto de ofrecer mayor seguridad en las transacciones y evitar al usuario la desconfianza típica en este tipo de transacciones, se ha implementado un sistema de valoraciones a los usuarios. Cuando un usuario gana una subasta, podrá votar al subastador en función del trato recibido y de las condiciones en que haya recibido el artículo ganado. Asimismo, también el vendedor podrá votar al comprador dependiendo del comportamiento de éste a la hora del pago.

Palabras Clave: MVC, Apache Struts, Maven, JUnit, Tomcat, J2EE, JavaEE, JSP, Servlet, JSTL, JNDI, JDBC, Front Controller, Data Access Object, DAO, Value Object, VO, Transfer Object, TO, Tiles, Aplicación Web, CSS, Subasta.

ÍNDICE DE CONTENIDOS

1. INTRODUCCIÓN	1
1.1. DETERMINACIÓN DE LA SITUACIÓN ACTUAL	1
1.2. ALCANCE Y OBJETIVOS	2
1.3. ESTRUCTURA DE LA MEMORIA	3
2. HERRAMIENTAS Y TECNOLOGÍAS UTILIZADAS	6
3. ESTADO DEL ARTE EN LA TECNOLOGÍA UTILIZADA	8
4. ESTUDIO DE VIABILIDAD	10
5. INTRODUCCIÓN AL DESARROLLO REALIZADO	12
6. REQUISITOS DEL SISTEMA	14
6.1. INTRODUCCIÓN	14
6.2. ACTORES	14
6.3. CASOS DE USO	15
6.3.1. Casos de uso relacionados con la cuenta de usuario	15
6.3.2. Casos de uso relacionados con subastas	25
6.3.3. Casos de uso relacionados con administración	36
6.3.4. Casos de usos del sistema	42
6.4. MODELO DE CASOS DE USO	43
7. DISEÑO DE LA APLICACIÓN	48
7.1. ARQUITECTURA GENERAL	48
7.2. SUBSISTEMA QUIENDAMAS	48
7.2.1. Objetivos	48
7.2.2. Arquitectura	48
7.2.2.1. Patrones arquitectónicos	48
7.2.2.2. Estructura de paquetes	49
7.2.3. Modelo	50
7.2.3.1. Clases Persistentes	50
7.2.3.2. Diseño de un DAO	53
7.2.3.3. Interfaces de las Fachadas del Modelo	55
7.2.3.3.1. Fachada de categorías	56
7.2.3.3.2. Fachada de usuarios	57
7.2.3.4. Diseño de una fachada	57
7.2.3.5. Otros aspectos	61
7.2.3.5.1. Thread para cerrar subastas	61

7.2.3.5.2.	Page-by-page iterator	62
7.2.3.5.3.	Custom Transfer Objects	62
7.2.4.	<i>Controlador</i>	64
7.2.4.1.	Patrón Front Controller de Struts	66
7.2.4.2.	Patrón Chain of Responsibility	67
7.2.4.3.	Plugin de categorías	68
7.2.5.	<i>Vista</i>	69
7.2.5.1.	Tiles	69
7.2.5.2.	Custom tags	69
7.2.5.3.	Formularios y validación	70
7.2.5.4.	Internacionalización	70
7.2.5.5.	Application objects	71
8.	IMPLEMENTACIÓN	72
8.1.	LISTA DE DEFECTOS	72
8.2.	SOFTWARE REQUERIDO	72
8.3.	ESTRUCTURA	73
8.4.	INSTRUCCIONES DE COMPILACIÓN	74
9.	PRUEBAS	76
10.	PLANIFICACIÓN Y EVALUACIÓN DE COSTES	78
11.	CONCLUSIONES Y FUTURAS LÍNEAS DE TRABAJO	82
12.	APÉNDICE	84
12.1.	INSTALACIÓN DEL SOFTWARE	84
12.1.1.	<i>Instalación de Maven 2</i>	84
12.1.2.	<i>Instalación de Apache Tomcat</i>	84
12.1.3.	<i>Instalación de MySQL</i>	85
12.1.4.	<i>Variables de entorno</i>	87
12.2.	MANUAL DE USUARIO	88
12.2.1.	<i>Navegación anónima</i>	88
12.2.2.	<i>Creación de subastas y panel de usuario</i>	96
12.2.3.	<i>Administración de la aplicación</i>	104
12.3.	CONTENIDO DEL CD	109
13.	BIBLIOGRAFÍA	110
13.1.	ENLACES DE INTERÉS	111
14.	GLOSARIO	112
15.	ACRÓNIMOS	114

ÍNDICE DE FIGURAS

ILUSTRACIÓN 1 PÁGINA WEB DE EBAY INC	2
ILUSTRACIÓN 2 CASOS DE USO DE USUARIO ANÓNIMO	44
ILUSTRACIÓN 3 CASOS DE USO DE USUARIO REGISTRADO	45
ILUSTRACIÓN 4 CASOS DE USO DE USUARIO PUJADOR	46
ILUSTRACIÓN 5 CASOS DE USO DE ADMINISTRADOR.....	47
ILUSTRACIÓN 6 CASOS DE USO DE SISTEMA	47
ILUSTRACIÓN 7 COMUNICACIÓN ENTRE CAPAS	49
ILUSTRACIÓN 8 ESTRUCTURA DE PAQUETES	50
ILUSTRACIÓN 9 CLASES PERSISTENTES	52
ILUSTRACIÓN 10 EXPLICACIÓN DATA ACCESS OBJECT.....	53
ILUSTRACIÓN 11 DISEÑO DE UN DAO.....	54
ILUSTRACIÓN 12 FACHADA DE CATEGORÍAS	56
ILUSTRACIÓN 13 FACHADA DE USUARIO	57
ILUSTRACIÓN 14 FACHADA DE SUBASTAS	58
ILUSTRACIÓN 15 ACCIONES DE LA FACHADA DE SUBASTAS	59
ILUSTRACIÓN 16 DIAGRAMA DE SECUENCIA DE REALIZAR PUJA.....	60
ILUSTRACIÓN 17 DIAGRAMA DE CLASES DE CERRAR SUBASTAS FINALIZADAS	61
ILUSTRACIÓN 18 DIAGRAMA DE SECUENCIA VALORAR COMPRADOR.....	65
ILUSTRACIÓN 19 DIAGRAMA DE SECUENCIA DE FILTROS DEL CONTROLADOR.....	66
ILUSTRACIÓN 20 DIAGRAMA DE CLASES DEL FRONT CONTROLLER DE STRUTS	67
ILUSTRACIÓN 21 ESTRUCTURA DE DIRECTORIOS.....	73
ILUSTRACIÓN 22 EJEMPLO DE DIAGRAMA DE PRUEBAS PARA UNA FACHADA	77
ILUSTRACIÓN 23 TABLA DE COSTES DEL PROYECTO.....	79
ILUSTRACIÓN 24 DIAGRAMA DE GANTT.....	80
ILUSTRACIÓN 25 PÁGINA PRINCIPAL	89
ILUSTRACIÓN 26 DETALLE DE SUBASTA.....	90
ILUSTRACIÓN 27 PERFIL DE USUARIO.....	91
ILUSTRACIÓN 28 BÚSQUEDA DETALLADA	92
ILUSTRACIÓN 29 RESULTADOS DE LA BÚSQUEDA	93
ILUSTRACIÓN 30 SUBCATEGORÍAS	94
ILUSTRACIÓN 31 FORMULARIO DE INICIO DE SESIÓN.....	95
ILUSTRACIÓN 32 FORMULARIO DE CREACIÓN DE CUENTA.....	96
ILUSTRACIÓN 33 FORMULARIO DE CREACIÓN DE SUBASTA	97
ILUSTRACIÓN 34 BUZÓN DE MENSAJERÍA PRIVADA	98
ILUSTRACIÓN 35 DETALLE DE UN MENSAJE PRIVADO	99
ILUSTRACIÓN 36 FORMULARIO DE ENVÍO DE MENSAJE.....	100

ILUSTRACIÓN 37 FORMULARIO DE EDICIÓN DEL PERFIL DE USUARIO	101
ILUSTRACIÓN 38 FORMULARIO DE CAMBIO DE CONTRASEÑA.....	102
ILUSTRACIÓN 39 INFORMACIÓN SOBRE LAS COMPRAS	103
ILUSTRACIÓN 40 INFORMACIÓN SOBRE LAS VENTAS.....	104
ILUSTRACIÓN 41 MENÚ DE ADMINISTRACIÓN	105
ILUSTRACIÓN 42 GESTIÓN DE CATEGORÍAS.....	106
ILUSTRACIÓN 43 FORMULARIO DE MODIFICACIÓN DE CATEGORÍAS	107
ILUSTRACIÓN 44 USUARIOS VALORADOS NEGATIVAMENTE.....	108
ILUSTRACIÓN 45 HISTORIAL DE OPERACIONES DEL USUARIO	108

ÍNDICE DE TABLAS

TABLA 1 CASO DE USO CREAR CUENTA DE USUARIO	16
TABLA 2 CASO DE USO INICIAR SESIÓN	17
TABLA 3 CASO DE USO CERRAR SESIÓN	18
TABLA 4 CASO DE USO CAMBIAR CONTRASEÑA.....	19
TABLA 5 CASO DE USO EDITAR PERFIL DE USUARIO.....	20
TABLA 6 CASO DE USO VER PERFIL DE USUARIO	21
TABLA 7 CASO DE USO CONSULTAR MENSAJES ENVIADOS	22
TABLA 8 CASO DE USO CONSULTAR MENSAJES RECIBIDOS	22
TABLA 9 CASO DE USO VER MENSAJE PRIVADO	23
TABLA 10 CASO DE USO ENVIAR MENSAJE PRIVADO.....	24
TABLA 11 CASO DE USO BORRAR MENSAJE PRIVADO.....	25
TABLA 12 CASO DE USO CREAR SUBASTA.....	27
TABLA 13 CASO DE USO VER SUBASTA	28
TABLA 14 CASO DE USO REALIZAR PUJA.....	29
TABLA 15 CASO DE USO VER PUJAS ACTIVAS.....	30
TABLA 16 CASO DE USO VER SUBASTAS ACTIVAS.....	31
TABLA 17 CASO DE USO VER ÚLTIMAS SUBASTAS	31
TABLA 18 CASO DE USO BUSCAR SUBASTAS	32
TABLA 19 CASO DE USO BUSCAR SUBASTAS DETALLADO	33
TABLA 20 CASO DE USO VALORAR VENDEDOR	34
TABLA 21 CASO DE USO VALORAR COMPRADOR	36
TABLA 22 CASO DE USO REVISAR USUARIOS VOTADOS NEGATIVAMENTE	37
TABLA 23 CASO DE USO VER HISTORIAL DE USUARIO	37
TABLA 24 CASO DE USO AGREGAR CATEGORÍA	38
TABLA 25 CASO DE USO ELIMINAR CATEGORÍA	39
TABLA 26 CASO DE USO EDITAR CATEGORÍA	40

TABLA 27 CASO DE USO CANCELAR PUJA	41
TABLA 28 CASO DE USO CERRAR SUBASTA.....	42
TABLA 29 CASO DE USO CERRAR SUBASTAS FINALIZADAS	43
TABLA 30 ARCHIVO TOMCAT-USERS.XML.....	84
TABLA 31 GLOBAL DATASOURCE EN SERVER.XML.....	85
TABLA 32 RESOURCE LINK EN CONTEXT.XML	85
TABLA 33 CONTENIDO DEL ARCHIVO .MYSQLDATA	86

AGRADECIMIENTOS

A Paula Montoto Castelao, tutora, por su atención y dedicación.

A mis padres por la paciencia y los ánimos prestados en todo momento.

1. Introducción

1.1. Determinación de la Situación actual

Uno de los factores principales que se deben afrontar a la hora de poner a la venta un producto es el establecimiento de un precio. La cantidad elegida debe, por un lado, compensar suficientemente por la pérdida del bien intercambiado y, por otro, resultar atractiva al comprador de forma que este intercambio llegue a término.

El precio es, *a priori*, sencillo de calcular para algunos productos o servicios, pero puede resultar más complicado cuando el valor de estos va más allá de costos de fabricación o escasez. Sin embargo (y al margen de las estimaciones que se pueden hacer), normalmente el precio que cualquier vendedor persigue para su producto es el máximo que un comprador esté dispuesto a pagar.

Desde la época de las primeras subastas de las que se tiene noticia (Babilonia, año 500 A.C.) hasta la actualidad, éstas han sufrido una gran evolución, como es natural, apareciendo distintos tipos, extendiéndose a diferentes ámbitos del comercio y basándose en diferentes sistemas. La última gran revolución probablemente haya sido la que permitió el salto de esta forma de transacción comercial a la red, a mediados de los noventa.

En esta época se produjo un gran auge de este tipo de sitios, acrecentado por la ~~o d w t d w l c " f g " n c u " r w p v q " e q o ö " s w g " e q o g p | » " g p "~~ la mayoría de este tipo de empresas que habían obtenido un valor irreal en el mercado desaparecieron o fueron absorbidas por otras.

Cuando se habla de subastas en Internet (y de empresas que sobrevivieron a la burbuja), no se puede dejar de mencionar el sitio web por excelencia en este tipo de comercio: eBay (Ilustración 1 Página web de eBay Inc.), un gigante creado en 1995 y que hoy en día cuenta con 88 millones de usuarios activos en todo el mundo.

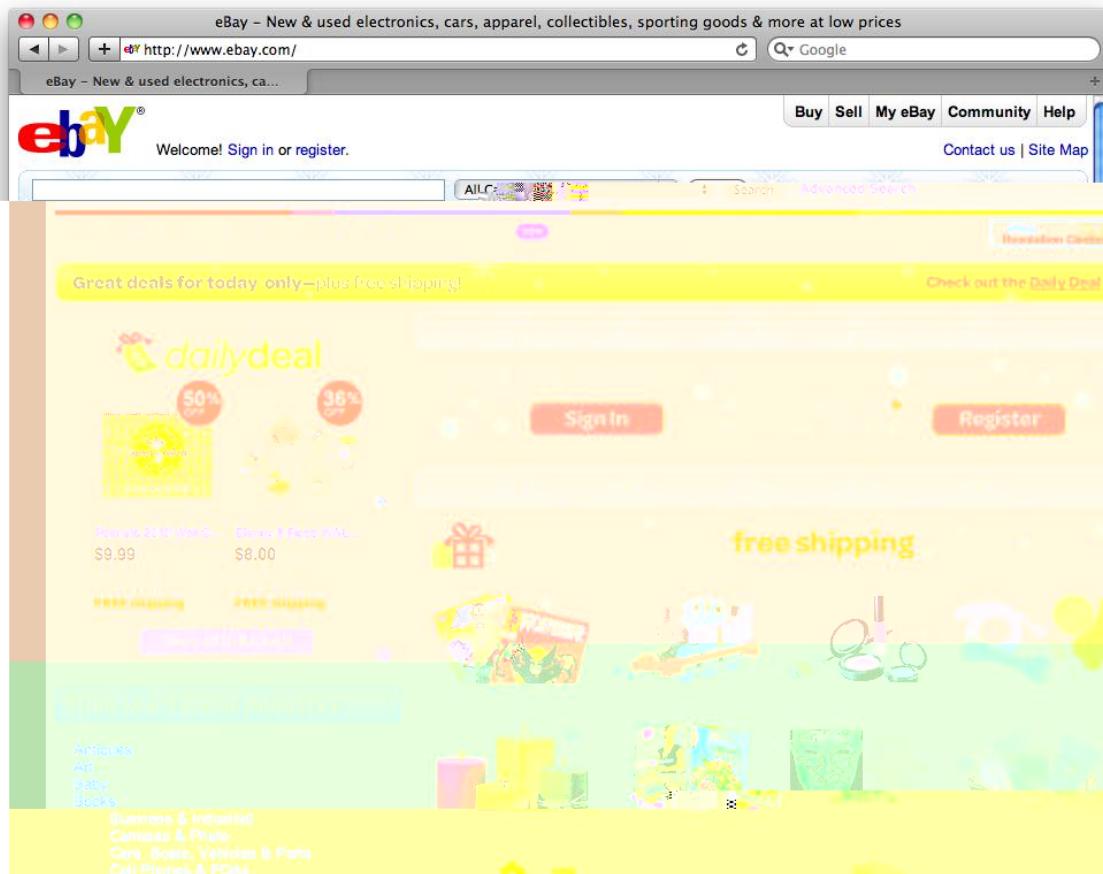


Ilustración 1 Página web de eBay Inc.

1.2. Alcance y Objetivos

El objetivo del sistema desarrollado es facilitar la compra-venta de productos entre personas, ofreciendo un canal común en el que el precio de un artículo está fijado por la demanda del mismo.

Para el correcto funcionamiento del sistema se hace necesario que los usuarios de la aplicación puedan registrarse, ligando así su identidad a las transacciones que realicen. Para el registro de usuarios se necesitará una dirección de correo electrónico válida que identifique a cada uno de forma única. Un sistema de mensajería interna garantizará una comunicación ágil entre vendedores y compradores, de forma que se resuelva rápidamente cualquier duda que pueda surgir sobre un producto o transacción.

Un usuario registrado podrá crear nuevas subastas en el sistema especificando un título, una descripción, un precio inicial, una fecha de finalización y una imagen del artículo subastado. Otros usuarios que estén registrados en el sistema podrán pujar por el artículo, siempre y cuando superen la mayor puja existente hasta el momento. Un usuario no podrá pujar en sus propias subastas para evitar que se inflen los precios artificialmente.

Puesto que el sistema no gestiona el envío y entrega de artículos ni el pago de los mismos, tampoco puede garantizar que no existan usuarios malintencionados que pretendan aprovecharse de los demás. Para evitar esta situación, se implementa un sistema de valoraciones entre los vendedores y los compradores.

Tanto el usuario que vende un artículo como el que lo compra podrán valorar en una escala del 1 al 10 al otro una vez la operación comercial se haya completado. De este modo, si un comprador gana una subasta pero el vendedor no recibe el pago en el tiempo especificado, éste podrá dar una puntuación baja al comprador de forma que los demás usuarios puedan ser advertidos de la actitud de ese comprador. De la misma forma, el comprador podrá dar una baja calificación al vendedor si, por ejemplo, éste no le envía el producto en el plazo estipulado. La valoración que ostenta cada usuario figurará en su perfil personal accesible para todos los usuarios del sistema.

Además de la posibilidad de realizar búsquedas de artículos por las diferentes características aportadas por los vendedores, se define un árbol de categorías en el que deberá encajarse cada producto sacado a subasta. De esta manera, se brinda al usuario una clasificación sencilla y eficaz que le ayudará a encontrar fácilmente lo que busque.

La zona de administración de la aplicación contempla dos funcionalidades: la gestión de las categorías y la supervisión de los votos recibidos por los usuarios. Los administradores serán los encargados de añadir, modificar o borrar las categorías del sistema dependiendo de las necesidades del mismo y de la naturaleza de los artículos subastados. Como complemento a la técnica de valoración, podrá verse qué usuarios reciben peores valoraciones para alertar o expulsar del sistema a los malintencionados.

1.3. Estructura de la memoria

1. **Introducción:** explicación del entorno al que está destinado esta aplicación, los precedentes y lo que se pretende conseguir con ella.

2. **Herramientas y tecnologías utilizadas:** definición de las aplicaciones y tecnologías para llevar a cabo el desarrollo del sistema, comentando los aspectos más relevantes de las mismas.
3. **Estado del arte en la tecnología utilizada:** comparativa entre la tecnología elegida para el proyecto y el resto de tecnologías similares candidatas, explicando brevemente los pros y contras de cada una de ellas.
4. **Estudio de viabilidad:** exposición de los argumentos que se han considerado a la hora de proponer la realización del proyecto justificando su viabilidad.
5. **Introducción al desarrollo realizado:** explicación de la metodología y procesos empleados en el diseño e implementación del proyecto, así como las fases o incrementos del mismo.
6. **Requisitos del sistema:** descripción concisa de las funcionalidades que ofrecerá la aplicación, explicando los actores que intervendrán y los casos de uso a los que se dará soporte.
7. **Diseño de la aplicación:** desarrollo en profundidad de la estructura y arquitectura de la aplicación realizada explicando también los principios y patrones de diseño utilizados.
8. **Implementación:** información sobre el software requerido para el funcionamiento del proyecto así como mejoras que se podrían llevar a cabo.
9. **Pruebas:** exposición de los métodos y herramientas utilizadas para probar el software implementado.
10. **Planificación y evaluación de costes:** explicación del plan de previsión confeccionado para la realización del proyecto y el estudio de costes previstos para la aplicación.
11. **Conclusiones y futuras líneas de trabajo:** conclusiones acerca del resultado final y explicación de las posibles líneas de trabajo futuro: ampliaciones de funcionalidad y mejoras.

12. **Apéndice:** explicación de la instalación y el uso de la aplicación así como relación de contenidos del CD adjunto.
13. **Bibliografía:** relación de fuentes consultadas para la realización del proyecto.
14. **Glosario:** lista y definición de términos utilizados en este documento.
15. **Acrónimos:** explicación de algunas de las siglas empleadas en el documento.

2. Herramientas y Tecnologías Utilizadas

- Apache Maven: herramienta de gestión de proyectos Java pensado para trabajar en red. Proporciona una estructura de directorios común que pretende, de alguna manera, estandarizar la organización de un proyecto (al menos en la organización que lo utilice). Maneja dependencias, incluso las transitivas mediante una configuración sencilla XML y un repositorio. Además, es extensible mediante plugins.
- Apache Struts: framework que facilita el desarrollo de aplicaciones web Java usando el patrón arquitectónico Model View Controller (MVC).
- Apache Tomcat: servidor web que funciona como contenedor de servlets y JavaServer Pages implementando las especificaciones de Sun. Engloba el contenedor de servlets, llamado Catalina y el motor de JSPs, llamado Jasper.
- CSS (Cascading Style Sheets): lenguaje dedicado a definir la forma en que se muestran los datos de un documento HTML, separando así la estructura de un documento de su presentación.
- Java EE: (Java Enterprise Edition): es una plataforma de programación para desarrollar software escrito en Java con arquitectura multinivel distribuida, basado en componentes ejecutados en un servidor de aplicaciones.
- JDBC (Java Database Connectivity): API que permite la ejecución de sentencias contra una base de datos, independientemente del sistema operativo, empleando el dialecto SQL propio del SGDB en cuestión.
- JSP (JavaServer Pages): tecnología que permite la generación dinámica de contenido web mediante la inserción de código java en el documento. También pueden emplearse en forma de etiquetas (*tags*) importadas desde librerías externas o creadas *ad hoc*.
- JSTL (JavaServer Pages Standard Tag Library): componente de Java EE que extiende las características de las JSPs, proporcionando tags que facilitan la construcción del

contenido a mostrar mediante la implementación de las tareas más comunes. Permite simplificar el contenido de las JSPs facilitando el trabajo del diseñador web (que no tiene por qué conocer el lenguaje Java).

- JUnit: framework diseñado para facilitar la implementación de pruebas de unidad para cualquier aplicación Java. Proporciona la posibilidad de probar cada uno de los métodos de una clase, definiendo la salida esperada en función de una entrada.
- HTML (Hypertext Markup Language): lenguaje utilizado para la construcción de páginas web ocupándose de definir la estructura y el contenido de un documento mediante párrafos, imágenes, cabeceras, tablas y otros elementos.
- MagicDraw: herramienta CASE destinada al diseño y modelado de software orientado a objetos con facilidades para trabajo en grupo. Permite la realización de diagramas UML 2.0 de manera sencilla.
- MySQL: sistema de gestión de bases de datos (SGBD) relacional multiusuario y multihilo con licencia GNU GPL para cualquier uso compatible con ésta. Por otro lado está disponible para empresas y uso comercial mediante la compra de una licencia específica.
- NetBeans: IDE para el desarrollo de aplicaciones en cualquier lenguaje gracias a su carácter modular y extensible, aunque está especialmente enfocado a la programación en Java (Java, Java EE, Java ME, EJBs). Tiene un sistema de *plugins* que permite extender aún más su funcionalidad y en las últimas versiones ofrece soporte para proyectos Maven con el sistema base.
- OmniPlan: herramienta de planificación de proyectos que ofrece amplias posibilidades de asignación de tareas y recursos, así como cálculo de costes, establecimiento de horarios, etc. Todo esto es fácilmente exportable en informes y diagramas de Gantt.
- XML (Extensible Markup Language): metalenguaje con capacidad para definir otros lenguajes para necesidades particulares. Propone un estándar para el intercambio de información estructurada entre diferentes plataformas.

3. Estado del Arte en la Tecnología Utilizada

A la hora de realizar una aplicación web existen multitud de tecnologías que pueden ser utilizadas, aunque dependiendo de la naturaleza concreta de la aplicación sea más aconsejable emplear unas u otras. Incluso teniendo definidos los resultados esperados y los procesos que serán necesarios para obtenerlos, el abanico de posibilidades es amplio.

Aunque sería difícil establecer un ranking de popularidad en tecnologías web, sí parece obvio mencionar tres alternativas como las más utilizadas: PHP (PHP Hypertext Preprocessor), ASP (Active Server Pages) y Java EE (Java Enterprise Edition). Para este proyecto se ha decidido emplear la opción de Java EE principalmente por las siguientes razones:

- Facilita y potencia la separación entre la lógica de negocio (modelo) y la entrada y salida de datos (controlador y vista).
- Tiene como pilar fundamental el lenguaje Java, ampliamente difundido.
- El servidor de aplicaciones ofrece una serie de ventajas de bajo nivel (seguridad, conectividad remota, etc.) que no tienen en cuenta el problema a tratar.
- Portabilidad de la aplicación desarrollada a cualquier sistema que pueda instalar una máquina virtual de Java (la mayor parte).
- Escalabilidad de cara al futuro y facilidad a la hora de integrar distintos sistemas por contar con la potencia y flexibilidad de Java detrás.

En el caso de ASP, los principales puntos negativos son la nula portabilidad del software (nativamente, funciona solo en sistemas Microsoft) y la necesidad de adquirir costosas licencias para su utilización.

La opción de PHP es atractiva por su bajo coste (las licencias son libres así como los entornos de desarrollo), también proporciona buenos resultados utilizado en plataformas

LAMP (Linux + Apache + MySQL + PHP) y su curva de aprendizaje es mucho menos pronunciada que la de Java EE. Por el contrario, es más difícil conseguir una buena separación entre modelo y vista, además, tiende a dar lugar a código menos reutilizable y escalable que la opción de Java EE.

En cuanto al SGBD a utilizar, el software de código abierto más popular es MySQL, instalado en más de 6 millones de máquinas actualmente, existen otras opciones de código abierto como PostgreSQL y privativas como Oracle y Microsoft SQL Server (comúnmente utilizado con aplicaciones de tecnología *.Net).

Como IDE, ASP.Net posee Visual Studio, especialmente realizado para la programación en este entorno. PHP y JavaEE, sin embargo, tienen multitud de opciones libres y privativas, muchas de ellas incluso comunes. Los dos entornos más utilizados son Eclipse y NetBeans. En este caso, se ha decidido emplear el segundo por la facilidad de integración con Maven y Tomcat, ya que proporciona soporte total para ambos y facilita el desarrollo y la depuración de la aplicación.

4. Estudio de Viabilidad

A la hora de realizar un estudio de viabilidad, lo primero es tener claro el objetivo del proyecto a desarrollar. Por tratarse de un proyecto de fin de carrera, uno de los objetivos primordiales (además de conseguir la funcionalidad descrita en la fase de inicio) es que el alumno aprenda algunas de las tecnologías de amplia difusión en el entorno empresarial, así como las metodologías que formalizan el desarrollo de un proyecto.

Por tanto, la viabilidad o inviabilidad del proyecto no vendrá determinada solamente por la vertiente del rendimiento económico que se le pueda sacar al mismo, sino por el valor docente que representa para el alumno que lo desarrolla.

Típicamente, el valor de un nuevo proyecto radica en la reducción de costes que provoca en una actividad determinada que ya se está realizando, o bien en el rendimiento económico que ofrecerá el nuevo proyecto en sí. Como se ha dicho, esto no será exactamente así en este proyecto debido a la naturaleza del mismo.

Así pues, tendrá un importante papel el aprendizaje de las tecnologías empleadas en la realización del mismo:

- Diseño mediante MVC y otros patrones.
- Uso de frameworks de programación.
- Uso de gestor de proyectos.
- Programación apoyada en librerías y APIs de terceros.
- Trabajo en un marco de desarrollo iterativo e incremental.

En la vertiente del beneficio económico reportado por la aplicación, si bien está clara la dificultad (e imposibilidad) de competir con las alternativas descritas en la introducción como eBay, se pretende que pueda servir como núcleo sólido en el caso de que se quisiera aumentar la funcionalidad, en un futuro, para poder competir en el mercado.

Otra forma de obtener un rendimiento económico sería mediante la venta de la aplicación para su inclusión como subsistema dentro de otro tipo de aplicación. Algunos ejemplos podrían ser alguno de los mundos virtuales (estilo Second Life) que brindan a sus usuarios la posibilidad de comerciar entre ellos.

5. Introducción al Desarrollo Realizado

La metodología escogida para el desarrollo de esta aplicación ha sido el Proceso Unificado (Unified Process) apoyado en el Lenguaje Unificado de Modelado (UML) cuando se requieren técnicas gráficas que apoyen las explicaciones. Cabe mencionar que el PU es un marco de desarrollo amplísimo, que abarca todo el ciclo de vida de un proyecto y (especialmente siguiendo refinamientos como *Rational Unified Process*) cuenta con multitud de documentos para formalizar todos los aspectos del desarrollo.

En un proyecto como este, por su discreta envergadura, no es necesaria una formalización tan exhaustiva, la cual provocaría grandes retrasos y difícilmente aclararía más el desarrollo, por lo que simplemente se ha seguido la filosofía principal basada en un desarrollo:

- Iterativo e incremental: el PU establece una serie de fases para cada una de las cuales puede haber 1..N iteraciones dependiendo de la complejidad del proyecto y de la fase en sí. El final de cada una de estas iteraciones se corresponde con un incremento en la funcionalidad del sistema.
- Dirigido por los casos de uso: previo paso a definir las iteraciones que se llevarán a cabo, se debe hacer una descripción de casos de uso recabando los requisitos funcionales. Así pues, cada iteración se encargará de un subconjunto de casos de uso.
- Centrado en la arquitectura: dadas las extremas diferencias que se pueden dar entre unos proyectos software y otros, el PU ofrece un marco extensible de trabajo pensado para cubrir todas las características y diseñado para que cada proyecto enfatice, en mayor medida, los aspectos específicos que mejor cubren sus necesidades.
- Enfocado a los riesgos: en el desarrollo de software, un error detectado en un estado tardío es mucho más costoso que uno detectado en las primeras fases

de desarrollo. Por ello, el PU potencia que el desarrollador identifique los principales riesgos en las etapas tempranas.

Las fases del Proceso Unificado son: inicio, elaboración, construcción y transición, cada una de ellas formada, a su vez, por el ciclo de vida clásico: análisis de requisitos, diseño, implementación y pruebas.

La subdivisión en iteraciones que se ha decidido para este proyecto viene dada por dos factores principales:

- Aprendizaje: se ha comenzado el desarrollo por las partes más documentadas y fáciles de comprender para facilitar el aprendizaje de las tecnologías empleadas.
- Funcionalidades: se ha dado preferencia a los aspectos del sistema fundamentales para el funcionamiento del mismo dejando para más adelante los requisitos más accesorios.

Las iteraciones decididas finalmente para la fase de construcción fueron las siguientes:

- Gestión de usuarios y mensajes: centrada en la sesión, perfil de usuario y mensajes privados.
- Gestión de subastas: centrada en creación de subastas, pujas y búsquedas.
- Administración y valoraciones de usuario: implementación de la parte de administración del sistema así como la del sistema de valoraciones de usuarios.

6. Requisitos del Sistema

6.1. Introducción

Una de las características del Proceso Unificado es que está dirigido por los casos de uso y así ha sido, en particular, con este proyecto. Una vez hecha la descripción inicial del sistema se procedió a recopilar los requisitos funcionales mediante un diagrama de casos de uso, así como a detectar los actores que interactuarían con el sistema.

6.2. Actores

Cabe destacar que los actores en un modelo de casos de uso pueden ser usuarios, sistemas externos, subsistemas, etc. Un usuario podría acceder al sistema con diferentes roles (que podría dar lugar a actores distintos) y diferentes usuarios podrían ser el mismo actor si desempeñan el mismo rol en el sistema. A continuación se describen los actores identificados:

- Usuario anónimo: representa al usuario del sistema que no se ha identificado en el mismo, por lo que no tiene una cuenta de usuario asociada y no podrá acceder a la mayoría de funcionalidades.
- Usuario registrado: se corresponde con el usuario del sistema que ha iniciado una sesión (se ha identificado) por lo que está ligado a una cuenta de usuario.
- Usuario pujador: es un usuario identificado en el sistema y que además está participando en una subasta.
- Administrador: rol de usuario con privilegios especiales destinado a tareas de mantenimiento y evitación abusos por parte de los usuarios registrados.
- Sistema: representa al propio sistema en el rol de realizar ciertas funcionalidades automáticas o semi-automáticas.

6.3. Casos de Uso

La descripción de casos de uso de un sistema sirve para dar una especificación detallada de las tareas que éste va a realizar y por quién van a ser realizadas (actores). En ningún caso debe imponer la forma de llevar a cabo estas tareas o los medios a utilizar, puesto que esos son problemas que se deberán abordar más adelante, en las fases de elaboración y construcción.

6.3.1. Casos de uso relacionados con la cuenta de usuario

1. Crear cuenta de usuario

Descripción

Un usuario accede a la aplicación para dejar constancia de sus datos en el sistema y que este pueda identificarlo en sucesivas visitas.

Actores

Usuario anónimo.

Precondiciones

El nombre de usuario e email del usuario no han sido utilizados en el sistema con anterioridad.

Postcondiciones

El nuevo usuario queda registrado y listo para utilizar.

Flujo básico

1. El usuario anónimo accede al formulario de registro.
2. Rellena los datos necesarios para figurar en el sistema y los envía.
3. El sistema valida la información.
4. Se guarda el nuevo usuario.

- | |
|---|
| 5. Se devuelve al usuario al formulario de <i>login</i> . |
|---|

Flujo alternativo

- 3a. El usuario no ha llenado todos los campos obligatorios.
- 3b. El usuario inserta caracteres distintos en los campos: "f g " ñ e q p v e q p v t c u g ° c ö 0
- 3c. El usuario inserta un email con formato inválido.

En todos los casos se devuelve al usuario al formulario de registro señalando los datos obligatorios que no han sido cumplimentados.

Tabla 1 Caso de uso Crear cuenta de usuario

2. Iniciar sesión

Descripción

Un usuario proporciona al sistema su nombre de usuario y contraseña para autenticarse y adquirir el estatus de usuario registrado.

Actores

Usuario anónimo, usuario registrado.

Precondiciones

Es usuario anónimo.

Postcondiciones

Es usuario registrado.

Flujo básico

1. El usuario anónimo accede al formulario de autenticación.

<ol style="list-style-type: none"> 2. Rellena los datos de nombre de usuario y contraseña (opcionalmente el de recordar sesión de usuario) y los envía. 3. El sistema valida la información. 4. Se actualiza la sesión del usuario. 5. Se devuelve al usuario a la página principal.
<p><i>Flujo alternativo</i></p> <ul style="list-style-type: none"> ○ 3a. El usuario no ha llenado todos los campos obligatorios. ○ 3b. El nombre de usuario no existe o la contraseña es incorrecta. <p>En todos los casos se devuelve al usuario al formulario de inicio de sesión señalando los datos erróneos o no cumplimentados.</p>

Tabla 2 Caso de uso Iniciar sesión

3. Cerrar sesión
<p><i>Descripción</i></p> <p>Un usuario registrado finaliza la sesión en el sistema pasando a ser usuario anónimo.</p>
<p><i>Actores</i></p> <p>Usuario registrado, usuario anónimo.</p>
<p><i>Precondiciones</i></p> <p>Es usuario registrado.</p>
<p><i>Postcondiciones</i></p> <p>Es usuario anónimo.</p>
<i>Flujo básico</i>

<ol style="list-style-type: none"> 1. El usuario solicita el cierre de la sesión. 2. El sistema cierra la sesión en el sistema desvinculándola de la cuenta de usuario.
<i>Flujo alternativo</i>

Tabla 3 Caso de uso Cerrar sesión

4. Cambiar contraseña	
<i>Descripción</i>	Un usuario registrado cambia su contraseña de acceso al sistema por otra diferente.
<i>Actores</i>	Usuario registrado.
<i>Precondiciones</i>	
<i>Postcondiciones</i>	La cuenta de usuario tiene ligada la nueva contraseña.
<i>Flujo básico</i>	<ol style="list-style-type: none"> 1. El usuario accede al formulario de cambio de contraseña. 2. Rellena los datos necesarios (contraseña antigua, nueva y nueva repetida) y los envía. 3. El sistema valida la información. 4. Se guarda la nueva clave de usuario. 5. Se devuelve al usuario a la página principal.

Flujo alternativo

- 3a. El usuario no ha llenado todos los campos.
 - 3b. El usuario inserta caracteres dis v k p v q u " g p " n q u " e c o r q u ' e q p v t c u g ° c ö 0
 - 3c. El usuario inserta la contraseña antigua incorrectamente.
- En todos los casos se devuelve al usuario al formulario de cambio de contraseña señalando los datos erróneos o no cumplimentados.

Tabla 4 Caso de uso Cambiar contraseña**5. Editar perfil de usuario***Descripción*

Un usuario registrado actualiza o añade nueva información en su perfil.

Actores

Usuario registrado.

*Precondiciones**Postcondiciones*

El perfil de usuario queda actualizado con los nuevos datos especificados.

Flujo básico

1. El usuario accede al formulario de actualización de perfil que aparecerá cumplimentado con la información ya existente.
2. El usuario añade o modifica cualquiera de los datos mostrados.
3. El sistema valida la información.

- | |
|---|
| <ol style="list-style-type: none"> 4. Se guardan los nuevos datos de perfil. 5. Se devuelve al usuario a la página principal. |
|---|

Flujo alternativo

- o 3a. El usuario no ha llenado todos los campos.
- o 5 d 0 " G n " w u w c t k q " k p u g t v c " e c t c e g v g ö g u " f e q p v t c u g ° c ö 0
- o 3c. El usuario inserta la contraseña antigua incorrectamente.

En todos los casos se devuelve al usuario al formulario de actualización de perfil de usuario señalando los datos erróneos o no cumplimentados.

Tabla 5 Caso de uso Editar perfil de usuario

6. Ver perfil de usuario

Descripción

Un usuario consulta el perfil de otro, accediendo así a una serie de datos de su cuenta como nombre, fecha de nacimiento, valoración en el sistema, etc.

Actores

Usuario anónimo.

Precondiciones

El usuario consultado existe.

Postcondiciones

El perfil consultado se muestra en pantalla.

Flujo básico

1. El usuario solicita acceder al perfil de otro usuario.
2. El sistema valida la petición.
3. El sistema devuelve la información del usuario solicitado.

Flujo alternativo

- o 3a. El nombre de usuario especificado no figura en el sistema: se informa al usuario de que no se encuentra la cuenta especificada.

Tabla 6 Caso de uso Ver perfil de usuario

7. Consultar mensajes enviados

Descripción

Operación para que un usuario obtenga los mensajes que ha enviado hasta el momento y no han sido borrados.

Actores

Usuario registrado.

Precondiciones

Los mensajes enviados por el usuario aparecen en pantalla.

Flujo básico

1. El usuario solicita al sistema la lista de mensajes que ha enviado.
2. El sistema valida la solicitud.
3. El sistema devuelve al usuario una lista con los mensajes que ha enviado.

Flujo alternativo

Tabla 7 Caso de uso Consultar mensajes enviados

8. Consultar mensajes recibidos

Descripción

Operación para que un usuario obtenga los mensajes que ha recibido hasta el momento y no han sido borrados.

Actores

Usuario registrado.

Precondiciones

Postcondiciones

Los mensajes recibidos por el usuario aparecen en pantalla.

Flujo básico

1. El usuario solicita al sistema la lista de mensajes que ha recibido.
2. El sistema valida la solicitud.
3. El sistema devuelve al usuario una lista con los mensajes que ha recibido.

Flujo alternativo

Tabla 8 Caso de uso Consultar mensajes recibidos

9. Ver mensaje privado

Descripción

Se recupera el mensaje especificado y es mostrado en pantalla.
<p><i>Actores</i></p> <p>Usuario registrado.</p>
<p><i>Precondiciones</i></p> <p>El mensaje privado existe en el sistema.</p>
<p><i>Postcondiciones</i></p> <p>El mensaje solicitado se muestra en pantalla.</p>
<p><i>Flujo básico</i></p> <ol style="list-style-type: none"> 1. El usuario solicita un mensaje concreto. 2. El sistema valida la solicitud asegurando que el mensaje solicitado pertenece al usuario solicitante. 3. El sistema devuelve al usuario los detalles del mensaje solicitado.
<p><i>Flujo alternativo</i></p> <ul style="list-style-type: none"> o 2-3a. El mensaje solicitado no pertenece al usuario: se redirige al usuario a la lista de mensajes para que seleccione uno correctamente.

Tabla 9 Caso de uso Ver mensaje privado

10. Enviar mensaje privado
<p><i>Descripción</i></p> <p>Un usuario crea un mensaje privado que otro usuario podrá leer.</p>
<p><i>Actores</i></p> <p>Usuario registrado.</p>

<i>Precondiciones</i>
El destinatario existe en el sistema.
<i>Postcondiciones</i>
Un nuevo mensaje figura en el sistema con los datos especificados.
<i>Flujo básico</i>
<ol style="list-style-type: none"> 1. El usuario accede al formulario de envío de mensaje. 2. El usuario cumplimenta los datos requeridos (destinatario, asunto y texto) y los envía. 3. El sistema valida los datos recibidos. 4. El sistema almacena el mensaje para que sea leído por el destinatario. 5. Se devuelve al usuario a la lista de mensajes privados.
<i>Flujo alternativo</i>
<ul style="list-style-type: none"> ○ 3a. El usuario no ha cumplimentado alguno de los campos: se devuelve al usuario al formulario de envío de mensaje informando del campo no cumplimentado. ○ 3b. El destinatario no existe: se devuelve al usuario al formulario de envío de mensaje informando de que el destinatario no existe en el sistema.

Tabla 10 Caso de uso Enviar mensaje privado

11. Borrar mensaje privado
<i>Descripción</i>
Un usuario borra uno de los mensajes privados recibidos o enviados.
<i>Actores</i>
Usuario registrado.

<i>Precondiciones</i>
El usuario ha enviado o recibido al menos un mensaje.
<i>Postcondiciones</i>
El mensaje seleccionado ya no existe en el sistema.
<i>Flujo básico</i>
<ol style="list-style-type: none"> 1. El usuario solicita el borrado de un mensaje privado. 2. El sistema valida la solicitud asegurando que el mensaje solicitado pertenece al usuario solicitante. 3. El sistema elimina el mensaje especificado.
<i>Flujo alternativo</i>
<ul style="list-style-type: none"> ○ 2a. El mensaje solicitado no pertenece al usuario: se redirige al usuario a la lista de mensajes para que seleccione uno correctamente.

Tabla 11 Caso de uso Borrar mensaje privado**6.3.2. Casos de uso relacionados con subastas**

12. Crear subasta
<i>Descripción</i>
Inserta una nueva subasta en el sistema con los datos especificados por el usuario quedando lista para que el resto de usuarios del sistema participen en ella.
<i>Actores</i>
Usuario registrado.
<i>Precondiciones</i>

Postcondiciones

La nueva subasta figura en el sistema.

Flujo básico

1. El usuario accede al formulario de creación de subastas.
2. El usuario cumplimenta los datos requeridos (título, descripción, categoría, precio inicial, fecha de finalización, imagen) y los envía.
3. El sistema valida la información recibida.
4. Se devuelve al usuario a la página principal.

Flujo alternativo

- 3a. El usuario no ha cumplimentado alguno de los campos:

- < Título
- < Descripción
- < Categoría
- < Fecha de finalización
- < Precio inicial
- < Imagen del producto

Se devuelve al usuario al formulario de creación de subasta indicando los campos requeridos.

- 3b. La categoría especificada no existe en el sistema: se devuelve al usuario al formulario de creación de subasta indicando el error.
- 3c. La fecha de finalización indicada es anterior a la fecha actual: se devuelve al usuario al formulario de creación de subasta indicando el error.
- 3d. El tamaño de la imagen especificada es demasiado grande: se devuelve al usuario al formulario de creación de subasta indicando el error.

Tabla 12 Caso de uso Crear subasta

13. Ver subasta

<i>Descripción</i>

Consulta de la información referente a una subasta en concreto, mostrándose el detalle de ésta como creador, fecha de finalización, pujas hasta el momento, etc.

<i>Actores</i>

Usuario anónimo.
<i>Precondiciones</i>
La subasta existe en el sistema.
<i>Postcondiciones</i>
La información sobre la subasta especificada se muestra en pantalla.
<i>Flujo básico</i>
<ol style="list-style-type: none"> 1. El usuario solicita consultar la información sobre una subasta concreta. 2. El sistema valida la petición. 3. El sistema devuelve la información sobre la subasta especificada.
<i>Flujo alternativo</i>
<ul style="list-style-type: none"> ○ 3a. La subasta especificada no figura en el sistema: se informa al usuario de la inexistencia de la subasta.

Tabla 13 Caso de uso Ver subasta

14. Realizar puja
<i>Descripción</i>
Un usuario realiza una puja a una subasta del sistema optando a conseguir el producto ofertado.
<i>Actores</i>
Usuario pujador.
<i>Precondiciones</i>
Existe la subasta sobre la que se va a realizar la puja.

Postcondiciones

Se añade una puja a la subasta especificada a nombre del usuario que la ha realizado.

Flujo básico

1. El usuario accede al formulario de puja de una subasta concreta.
2. El usuario introduce el valor de la puja a realizar y lo envía.
3. El sistema valida la puja.
4. Se devuelve al usuario a la página con la información de la subasta pujada.

Flujo alternativo

- o 3a. El usuario pujador es el mismo usuario que ha creado la subasta: se devuelve al usuario a la página con la información de la subasta.
- o 3b. El valor de la nueva puja es inferior al mínimo requerido (el de la puja mayor puja + incremento mínimo): se devuelve al usuario a la página con la información de la subasta.
- o 3c. La cantidad no ha sido especificada o el formato es incorrecto: se devuelve al usuario a la página con la información de la subasta.

Tabla 14 Caso de uso Realizar puja

15. Ver pujas activas***Descripción***

Consulta de las pujas (pertenecientes al usuario que hace la consulta) realizadas a subastas que todavía no han finalizado.

Actores

Usuario pujador.

<i>Precondiciones</i>
<i>Postcondiciones</i>
Las pujas realizadas a subastas no finalizadas se muestran en pantalla.
<p><i>Flujo básico</i></p> <ol style="list-style-type: none"> 1. El usuario solicita la lista de pujas que ha realizado y cuyas subastas no han finalizado. 2. El sistema valida la petición. 3. Se devuelve al usuario una lista de pujas.

Tabla 15 Caso de uso Ver pujas activas

16. Ver subastas activas
<i>Descripción</i>
Consulta de las subastas (creadas por el usuario que hace la consulta) que todavía no han finalizado.
<i>Actores</i>
Usuario registrado.
<i>Precondiciones</i>
<i>Postcondiciones</i>
Las subastas creadas y no finalizadas se muestran en pantalla.
<p><i>Flujo básico</i></p> <ol style="list-style-type: none"> 1. El usuario solicita la lista de subastas que ha creado y no han finalizado.

- | |
|---|
| <ol style="list-style-type: none"> 2. El sistema valida la petición. 3. Se devuelve al usuario una lista de subastas. |
|---|

<i>Flujo alternativo</i>

Tabla 16 Caso de uso Ver subastas activas

17. Ver últimas subastas

<i>Descripción</i>

Recupera información de las subastas cuya fecha de finalización está más próxima.

<i>Actores</i>

Usuario anónimo.

<i>Precondiciones</i>

<i>Postcondiciones</i>

Las subastas que están a punto de finalizar se muestran en pantalla.
--

<i>Flujo básico</i>

1. El usuario solicita información sobre las subastas que están a punto de finalizar.
2. El sistema devuelve la información requerida.
3. Se devuelve al usuario a la página con el árbol de categorías.

<i>Flujo alternativo</i>

Tabla 17 Caso de uso Ver últimas subastas

18. Buscar subastas

<i>Descripción</i>
Realiza una búsqueda de subastas según una categoría y una cadena de búsqueda devolviendo un resumen de la información de aquellas que coinciden con el criterio especificado.
<i>Actores</i>
Usuario anónimo.
<i>Precondiciones</i>
<i>Postcondiciones</i>
La lista con las subastas que coinciden con el criterio de búsqueda se muestra en pantalla.
<i>Flujo básico</i>
<ol style="list-style-type: none"> 1. El usuario accede al formulario de búsqueda de subastas. 2. El usuario cumplimenta los datos (cadena de búsqueda y categoría) y los envía. 3. El sistema verifica la información recibida. 4. El sistema muestra la lista de coincidencias.
<i>Flujo alternativo</i>
<ul style="list-style-type: none"> ○ 3a. La categoría especificada no existe: se devuelve al usuario al formulario de búsqueda indicando el campo erróneo. ○ 4a. No se encuentran coincidencias: se informa al usuario de que ninguna subasta coincide con el criterio de búsqueda.

Tabla 18 Caso de uso Buscar subastas**19. Buscar subastas detallado**

<i>Descripción</i>
Realiza una búsqueda de subastas según cualquiera de los campos devolviendo un resumen de la información de aquellas que coinciden con el criterio especificado.
<i>Actores</i>
Usuario anónimo.
<i>Precondiciones</i>
<i>Postcondiciones</i>
La lista con las subastas que coinciden con el criterio de búsqueda se muestra en pantalla.
<i>Flujo básico</i>
<ol style="list-style-type: none"> 1. El usuario accede al formulario de búsqueda detallada de subastas. 2. El usuario cumplimenta los datos y los envía. 3. El sistema verifica la información recibida. 4. El sistema muestra la lista de coincidencias.
<i>Flujo alternativo</i>
<ul style="list-style-type: none"> ○ 3a. La categoría especificada no existe: se devuelve al usuario al formulario de búsqueda indicando el campo erróneo. ○ 4a. No se encuentran coincidencias: se informa al usuario de que ninguna subasta coincide con el criterio de búsqueda.

Tabla 19 Caso de uso Buscar subastas detallado**20. Valorar vendedor**

Descripción

El usuario que ha ganado una subasta asigna una calificación al vendedor, correspondiente al trato recibido por su parte en todo el proceso.

Actores

Usuario pujador.

Precondiciones

El usuario ha ganado una subasta.

Postcondiciones

Existe una nueva calificación asignada al vendedor para la subasta especificada.

Flujo básico

1. El usuario accede al formulario de valoración.
2. El usuario especifica la calificación que ofrece al usuario en la subasta que los relaciona y la envía.
3. El sistema valida la petición.
4. Se devuelve al usuario a la página con información de sus compras.

Flujo alternativo

- 3a. El usuario no ha cumplimentado el campo valoración o es erróneo.
- 3b. El usuario que envía la valoración no es el que ha ganado la subasta.
- 3c. El usuario ya ha sido valorado para esa subasta.
- 3d. La subasta todavía no ha finalizado.

En todos los casos se devuelve al usuario al formulario de valoración indicando el error.

Tabla 20 Caso de uso Valorar vendedor

21. Valorar comprador*Descripción*

El creador de una subasta asigna una calificación al comprador que la ha ganado con la puja máxima de las realizadas.

Actores

Usuario registrado.

Precondiciones

Una subasta del usuario ha finalizado con pujas.

Postcondiciones

Existe una nueva calificación asignada al comprador para la subasta especificada.

Flujo básico

1. El usuario accede al formulario de valoración.
2. El usuario especifica la calificación que ofrece al usuario en la subasta que los relaciona y la envía.
3. El sistema valida la petición.
4. Se devuelve al usuario a la página con información de sus ventas.

Flujo alternativo

- 3a. El usuario no ha cumplimentado el campo valoración o es erróneo.
- 3b. El usuario que envía la valoración no es el que ha creado la subasta.
- 3c. El usuario ya ha sido valorado para esa subasta.
- 3d. La subasta todavía no ha finalizado.

En todos los casos se devuelve al usuario al formulario de valoración indicando el error.

Tabla 21 Caso de uso Valorar comprador

6.3.3. Casos de uso relacionados con administración

22. Revisar usuarios votados negativamente

Descripción

Recuperación de una lista de usuarios registrados cuya valoración media esté por debajo de un umbral determinado.

Actores

Administrador.

Precondiciones

La lista de usuarios con media de valor inferior al umbral se muestran en pantalla.

Flujo básico

1. El usuario solicita la lista.
2. El sistema recupera el umbral especificado en la configuración del sistema.

- | |
|--|
| <ol style="list-style-type: none"> 3. El sistema recupera la lista de usuarios por debajo del umbral. 4. El sistema devuelve la lista de usuarios. |
| <i>Flujo alternativo</i> |

Tabla 22 Caso de uso Revisar usuarios votados negativamente

23. Ver historial de usuario
<i>Descripción</i>

Recuperación del historial de un usuario que contiene las subastas en las que ha participado el usuario hasta el momento junto con la valoración obtenida en ellas.

<i>Actores</i>

Administrador.

<i>Precondiciones</i>

El usuario cuyo historial se va a consultar figura en el sistema.

<i>Postcondiciones</i>

La lista de subastas en las que ha participado el usuario, junto a la valoración obtenida en cada una, se muestra en pantalla.

<i>Flujo básico</i>

1. El usuario solicita el historial de un usuario.
2. El sistema recupera la lista de subastas en las que ha participado el usuario y las muestra.

<i>Flujo alternativo</i>

Tabla 23 Caso de uso Ver historial de usuario

24. Agregar categoría*Descripción*

Permite la creación de una nueva categoría, en cualquier nivel del árbol de categorías del sistema, bajo la cual se podrán clasificar productos

Actores

Administrador.

Precondiciones

De especificarse una categoría padre, ésta existe en el sistema.

Postcondiciones

La nueva categoría se ha agregado al sistema.

Flujo básico

1. El usuario accede al formulario de creación de categorías.
2. El usuario cumplimenta los datos requeridos (nombre, descripción y padre) y los envía.
3. El sistema valida la información recibida.
4. Se devuelve al usuario a la página con el árbol de categorías.

Flujo alternativo

- 3a. La categoría especificada como padre de la nueva no existe en el sistema: se devuelve al usuario al formulario de creación de categorías señalando el campo como inválido.

Tabla 24 Caso de uso Agregar categoría

25. Eliminar categoría*Descripción*

Elimina del sistema alguna de las categorías creadas para la clasificación de subastas siempre que no esté siendo utilizada.

Actores

Administrador.

Precondiciones

La categoría a eliminar no es referenciada por otras ni por subastas.

Postcondiciones

La categoría especificada no existe en el sistema.

Flujo básico

1. El usuario solicita el borrado de una categoría.
2. El sistema elimina la categoría especificada.
3. Se devuelve al usuario a la página con el árbol de categorías.

Flujo alternativo

- o 2a. La categoría especificada no existe en el sistema: se devuelve al usuario al formulario de creación de categorías.
- o 2b. La categoría especificada es padre o clasifica al menos una subasta: se devuelve al usuario al formulario de creación de categorías.

Tabla 25 Caso de uso Eliminar categoría

26. Editar categoría

<i>Descripción</i>
Actualiza la información de una categoría existente en el sistema como el nombre, la descripción o la categoría padre de la misma.
<i>Actores</i>
Administrador.
<i>Precondiciones</i>
La categoría existe en el sistema.
<i>Postcondiciones</i>
La categoría especificada figura en el sistema con la nueva información aportada.
<i>Flujo básico</i>
<ol style="list-style-type: none"> 1. El usuario accede al formulario de modificación de categorías que aparecerá cumplimentado con la información ya existente. 2. El usuario añade o modifica cualquiera de los datos mostrados y los envía. 3. El sistema valida la información recibida. 4. Se devuelve al usuario a la página con el árbol de categorías.
<i>Flujo alternativo</i>
<ul style="list-style-type: none"> ○ 3a. La categoría especificada como padre no existe: se devuelve al usuario al formulario de creación de categorías señalando el campo como inválido. ○ 3b. El usuario no posee privilegios suficientes: se muestra una página de acceso prohibido.

Tabla 26 Caso de uso Editar categoría**27. Cancelar puja**

<i>Descripción</i>
Elimina una de las pujas de una subasta específica de forma que ésta ya no será tenida en cuenta cuando la subasta del artículo termine.
<i>Actores</i>
Administrador.
<i>Precondiciones</i>
Existe la puja a eliminar.
<i>Postcondiciones</i>
La puja especificada ya no figurará en la subasta.
<i>Flujo básico</i>
<ol style="list-style-type: none"> 1. El usuario solicita la cancelación de una de las pujas de una subasta. 2. El sistema cancela la puja. 3. Se devuelve al usuario a la página con la información sobre la subasta.
<i>Flujo alternativo</i>
<ul style="list-style-type: none"> ○ 2a. El usuario no posee privilegios suficientes: se muestra una página de acceso prohibido.

Tabla 27 Caso de uso Cancelar puja

28. Cerrar subasta
<i>Descripción</i>
<i>Actores</i>
Administrador.

Precondiciones

La subasta a cerrar ha sido creada con anterioridad.

Postcondiciones

<i>Precondiciones</i>
<i>Postcondiciones</i>
Todas las subastas cuya fecha indique que han terminado figurarán como cerradas.
<i>Flujo básico</i>

1. El sistema cierra las subastas cuya fecha de finalización sea anterior a la fecha actual.

<i>Flujo alternativo</i>

Tabla 29 Caso de uso Cerrar subastas finalizadas

6.4. Modelo de Casos de Uso

A continuación, se muestran los diagramas de casos de uso. Se ha decidido dividir el modelo en varios diagramas, agrupando los casos de uso según el actor al que corresponden, para así facilitar su entendimiento.

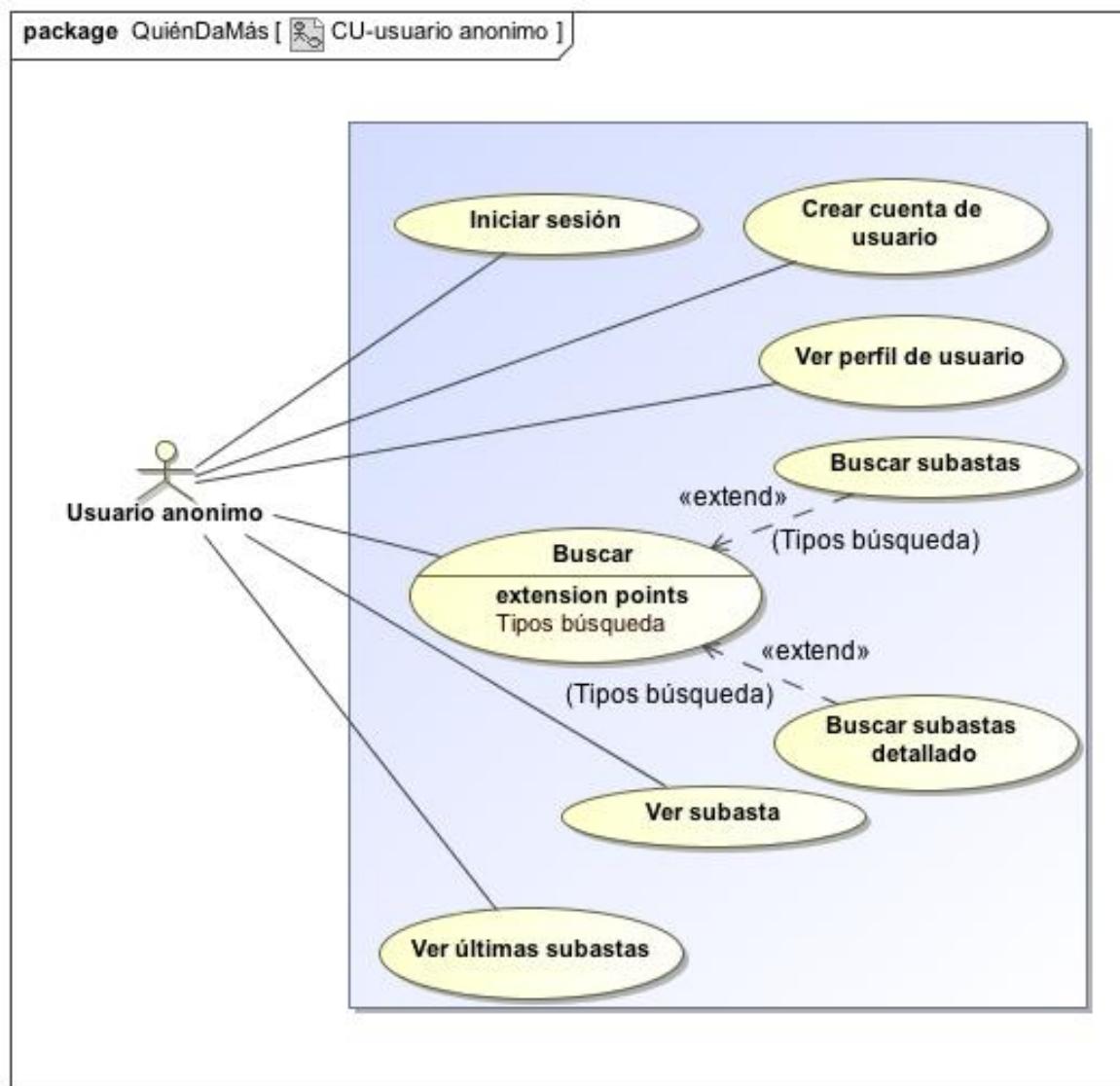


Ilustración 2 Casos de uso de Usuario anónimo

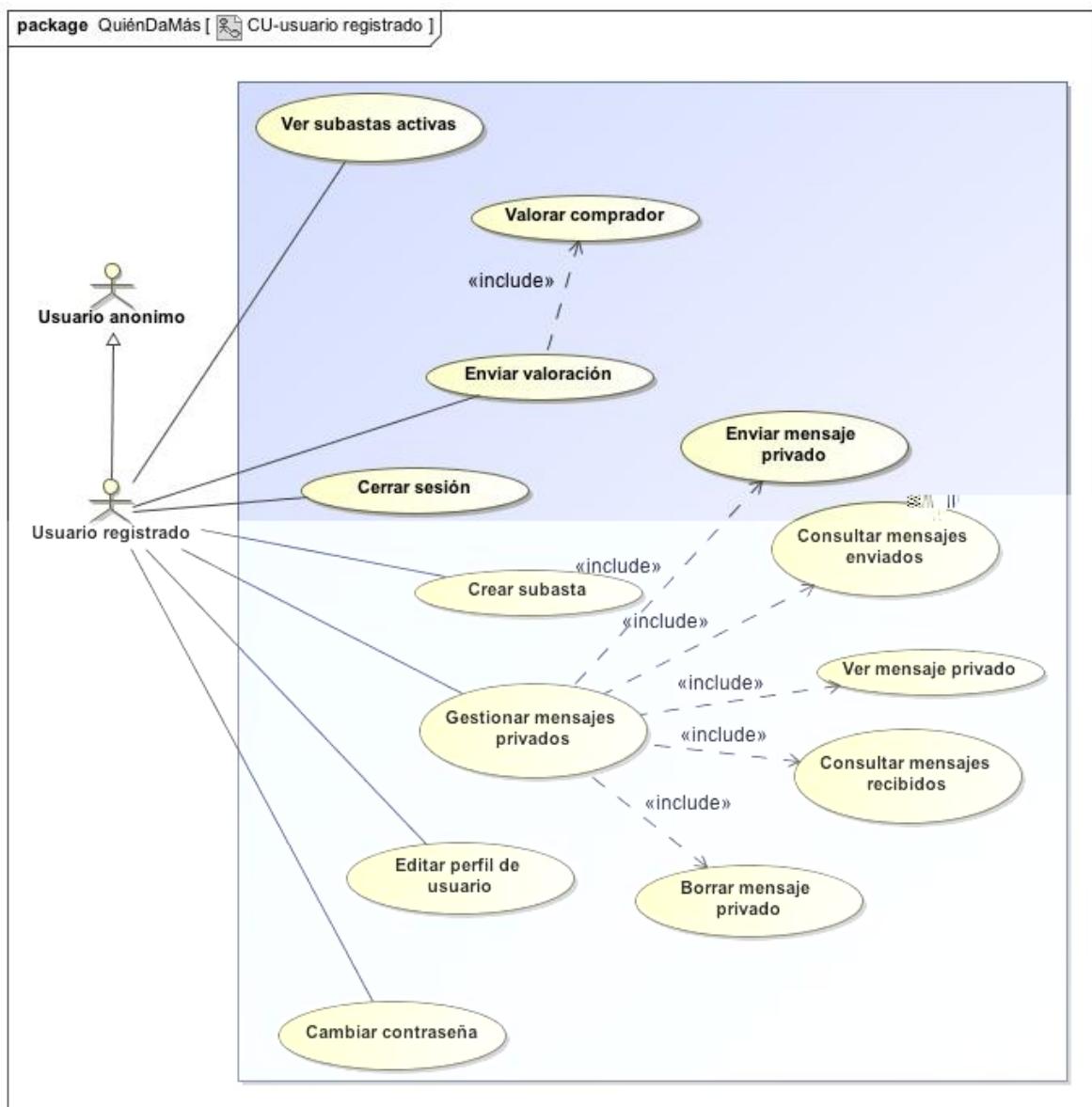


Ilustración 3 Casos de uso de Usuario registrado

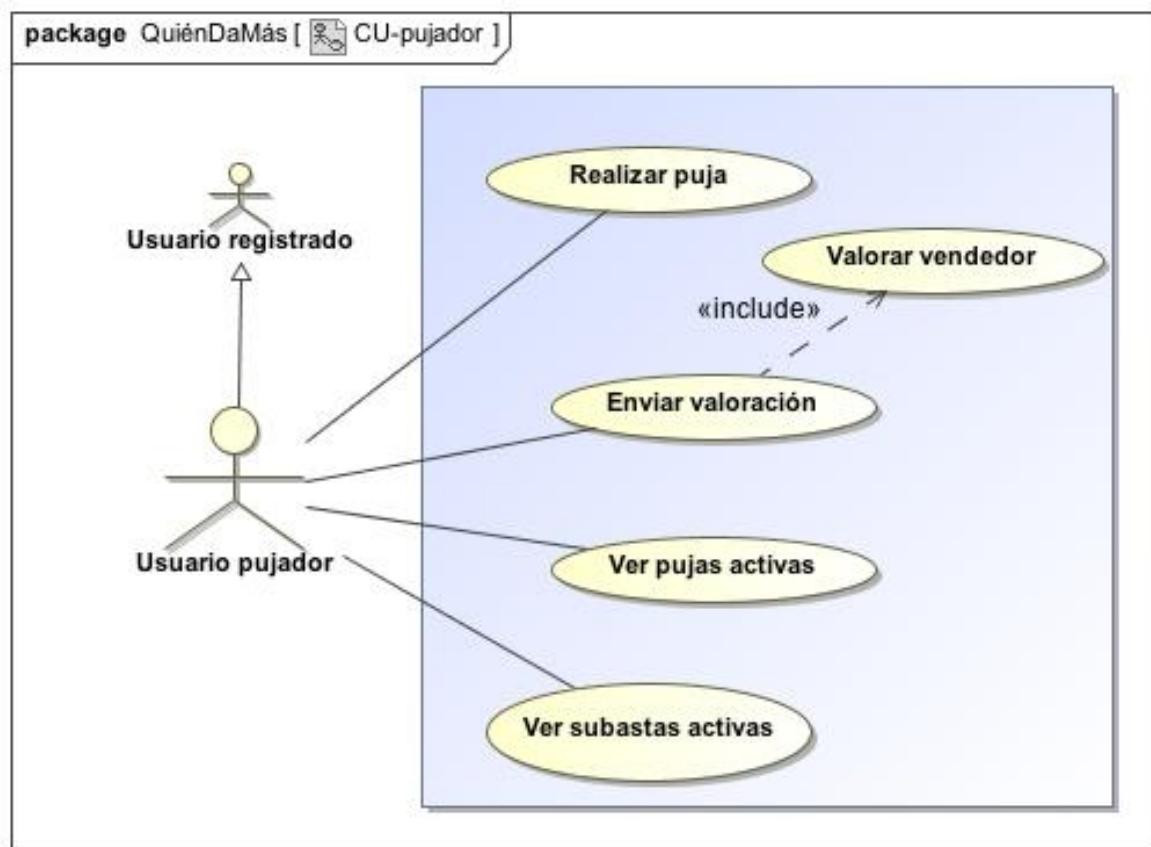


Ilustración 4 Casos de uso de Usuario pujador

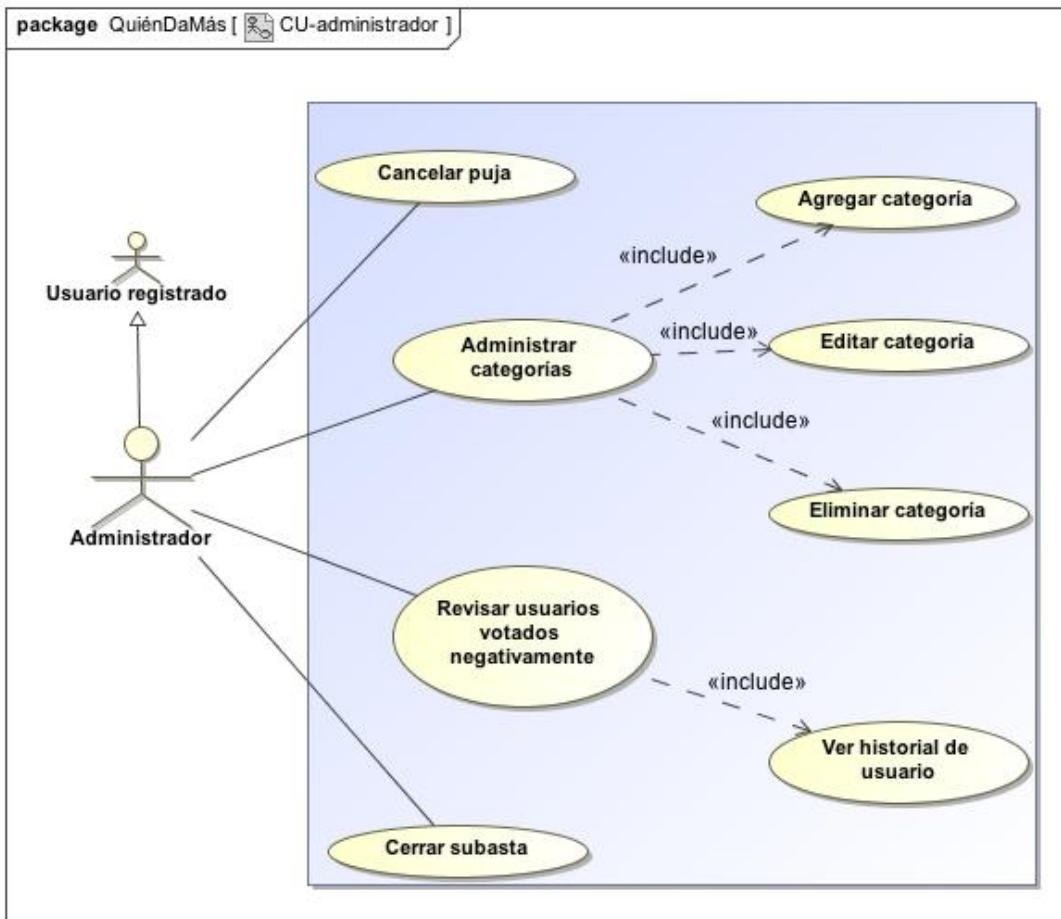


Ilustración 5 Casos de uso de Administrador

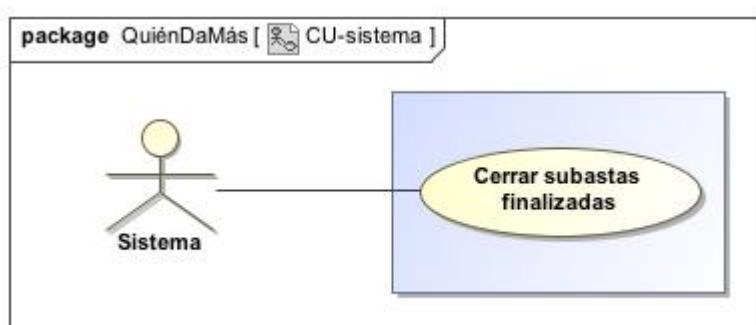


Ilustración 6 Casos de uso de Sistema

7. Diseño de la aplicación

7.1. Arquitectura general

A la hora de plantear la división del proyecto, y considerando el alcance del mismo, se optó por no dividirlo en más de un subsistema, principalmente, porque se considera que las funcionalidades aportadas están fuertemente relacionadas y, en conjunto, forman la base para cualquier sistema de este tipo que se quiera construir.

Sí se podrían considerar subsistemas futuros incrementos como los descritos en 11. Conclusiones y Futuras Líneas de Trabajo.

Se emplean, sin embargo, multitud de librerías de apoyo libres así como las proporcionadas para los ejemplos docentes realizados por profesores de la facultad.

7.2. Subsistema QUIENDAMAS

7.2.1. Objetivos

Puesto que se ha optado por no dividir el proyecto, los objetivos de este subsistema son exactamente los mismos que los que se han mencionado en la introducción y detallado en la descripción y modelo de casos de uso.

7.2.2. Arquitectura

7.2.2.1. Patrones arquitectónicos

La arquitectura de la aplicación se basa en el uso de dos patrones: Layers y Model-View-Controller (Ilustración 7 Comunicación entre capas). Por un lado, la filosofía de uso de *Layers* es la ocultación de la tecnología empleada en una capa de la del resto, así es posible que cambios de versiones del software y librerías que se utiliza en una capa no afecte a las demás.

La estructuración más eficiente cuando se trata de una aplicación web es la basada en tres capas definidas por el patrón Model-View-Controller, como su nombre indica, éstas son:

- Model (modelo): encapsula la lógica de negocio, es independiente de la comunicación con los actores que interactúan con el sistema y se encarga de las tareas de persistencia de la información tratada.
- View (vista): es la capa que sirve de interfaz entre el sistema y el usuario, no tiene por qué ser única ya que un mismo sistema puede necesitar, por ejemplo, una interfaz web y otra *standalone*.
- Controller (controlador): encargado de la comunicación entre vista y modelo, recibe los cambios que el usuario realiza en la capa vista para invocar funcionalidades del modelo y comunica las respuestas generadas.

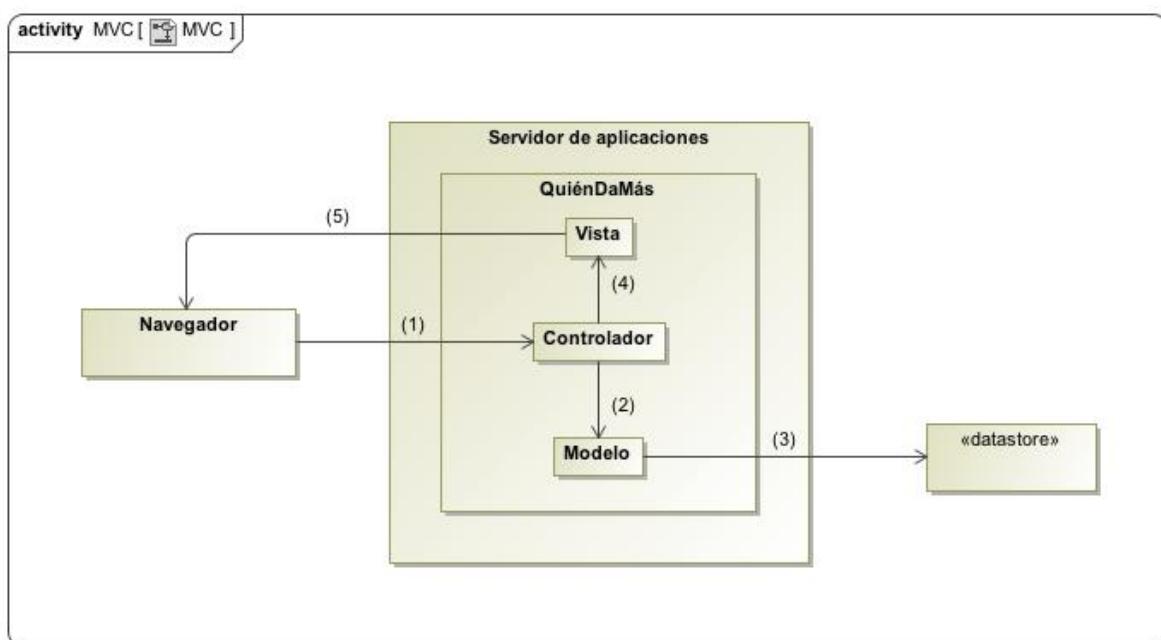
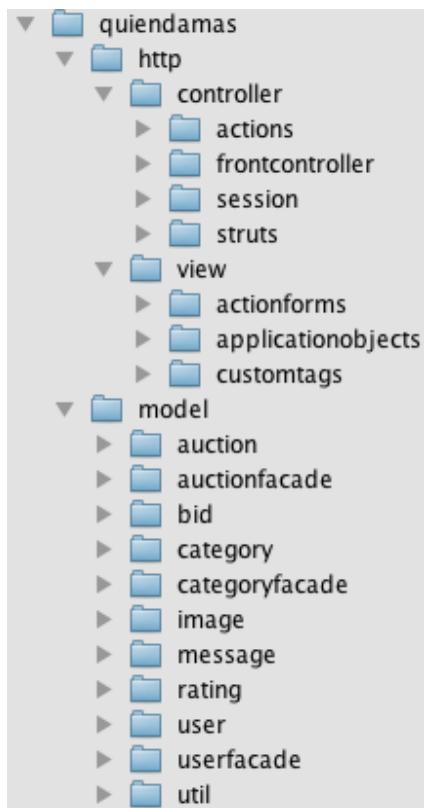


Ilustración 7 Comunicación entre capas

7.2.2.2. Estructura de paquetes

La estructura de paquetes de la aplicación se presenta como reflejo de la estructura Model-View-Controller (Ilustración 8 Estructura de paquetes), de forma que las clases se agrupan ciertos paquetes los cuales a su vez pertenecen a una jerarquía superior de paquetes definida por cada una de las capas descritas por el patrón MVC.

**Ilustración 8 Estructura de paquetes**

- Model (quiendamas.model): paquetes y clases que implementan los Transfer Objects, Data Access Objects, fachadas y factorías (explicado en detalle en 7.2.3 Modelo).
- View (quiendamas.http.view): actionforms de Struts y custom tags creados para facilitar el acceso a los datos por parte del diseñador. También contiene objetos para insertar en los formularios presentados al usuario.
- Controller (quiendamas.http.controller): acciones que procesan la información de los formularios e invocan a operaciones del modelo, clases relacionadas con el Front Controller de Struts como por ejemplo los filtros, gestor de la sesión de usuario y plugins de Struts.

7.2.3. Modelo

7.2.3.1. Clases Persistentes

El diagrama de clases persistentes (Ilustración 9 Clases persistentes) tiene como objetivo la representación visual y global de los *Transfer Objects* de la aplicación,

ofreciendo una visión global de los objetos del modelo y su correspondencia con los del mundo real.

El patrón de diseño *Transfer Object* tiene como finalidad la agrupación de una serie de atributos que corresponden a uno o varios objetos del dominio. De esta forma, cuando se solicita una información, ésta es devuelta encapsulada en un objeto en lugar de requerir la invocación de distintos métodos para la recuperación de todos los atributos.

Cada una de las clases descritas a continuación implementan la interfaz *Serializable*, además, sobrescriben la implementación de los métodos *equals()*, *hashCode()* y *toString()* con el fin de facilitar la codificación de las pruebas.

- CategoryTO: representa una categoría del sistema bajo la cual se clasificarán las subastas. Posee un nombre, una descripción y la categoría padre.
- AuctionTO: agrupa la información relativa a una subasta como el nombre, la fecha en que finaliza, el precio inicial, el usuario que la creó, etc.
- BidTO: recoge la información sobre una puja, siempre figura ligada a una subasta y al usuario que la ha realizado junto con el importe de la misma.
- ImageTO: contiene una imagen correspondiente a una subasta, así como su tipo (*content-type*) y nombre.
- RatingTO: identifica el concepto de valoración a un usuario para una subasta concreta, relacionando estos datos con una calificación específica
- UserTO: representa el concepto de usuario del sistema. Agrupa todos los datos personales del usuario (nombre, apellido, dirección, teléfono, correo electrónico, configuración de su cuenta en el sistema (nombre de usuario, contraseña, fecha de nacimiento, sexo, etc.)
- MessageTO: agrupa los datos de un mensaje privado, tales como el usuario que lo envía y el que lo recibe, la fecha o el contenido del mismo.

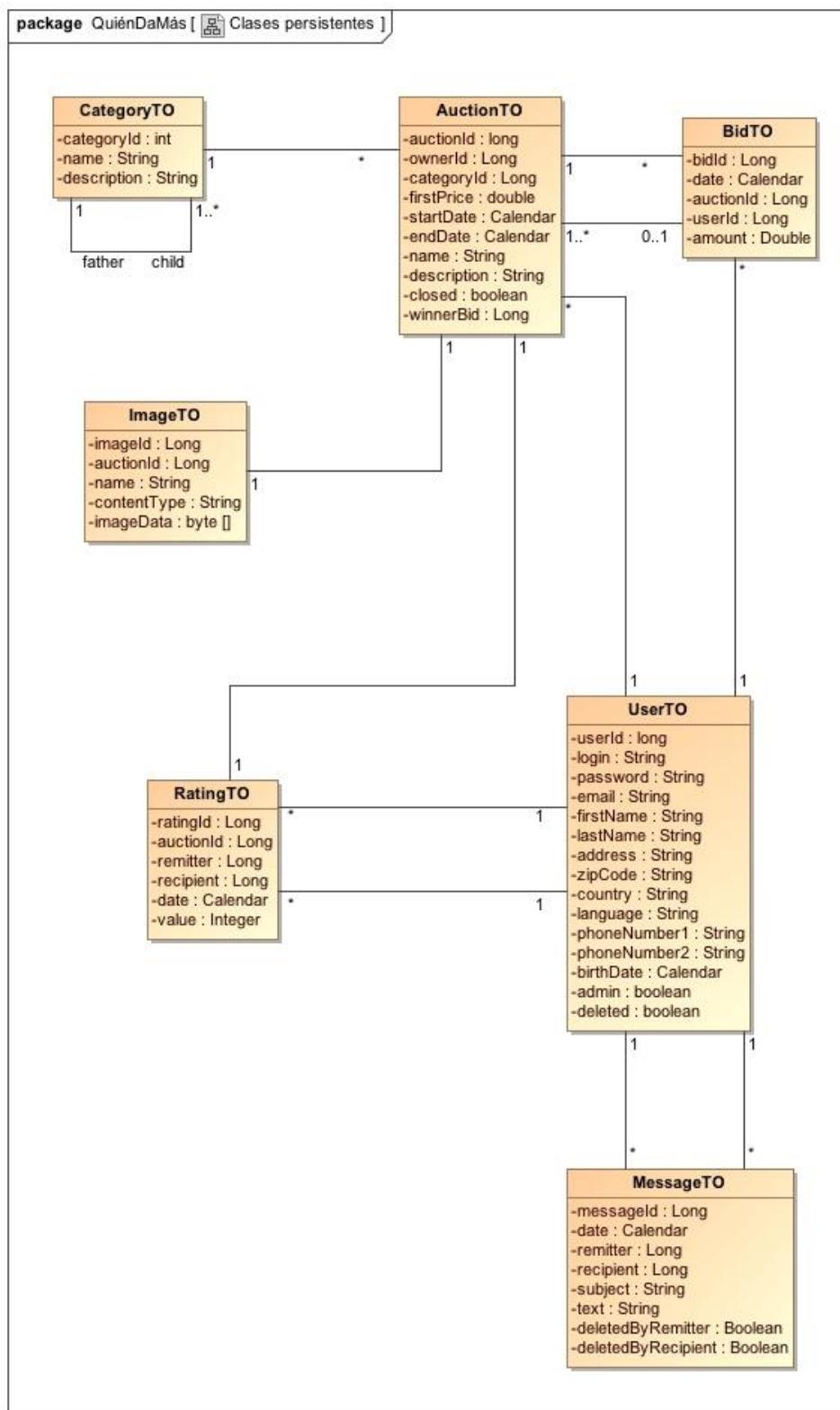


Ilustración 9 Clases persistentes

7.2.3.2. Diseño de un DAO

El patrón *Data Access Object* tiene como principal objetivo proveer una interfaz de acceso a la base de datos (o el mecanismo de persistencia utilizado) basada en una serie de operaciones que no muestran detalles de la misma. Se consigue así un aislamiento entre la lógica de la aplicación y el acceso a datos que garantiza que los cambios realizados en una parte no afectarán a la otra.

Además de la abstracción descrita que proporciona el DAO, la implementación se ha realizado de forma que el código del modelo sea independiente del SGBD mediante el uso conjunto de tres patrones de diseño: *Data Access Object*, *Factory Method* y *Adapter* (Ilustración 10 Explicación Data Access Object).

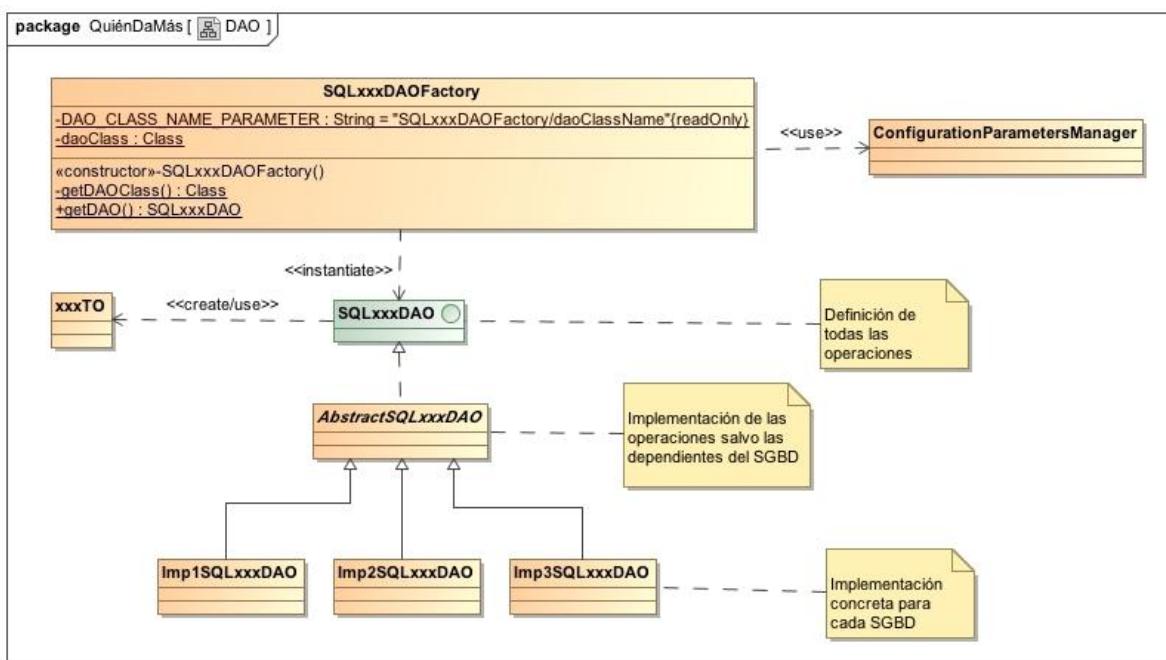


Ilustración 10 Explicación Data Access Object

Mediante la arquitectura descrita, las clases *ImpNSQLxxxDAO* adaptan a cada SGBD, ofreciendo una implementación concreta de las operaciones dependientes del mismo (*create()*). Por otra parte, el uso de la factoría garantiza la posibilidad de cambio de implementación sin necesidad de recompilar el código de la aplicación.

De esta forma, en el momento en que se requiera un DAO, la factoría es la encargada de devolver la implementación que corresponda (especificada de alguna manera en la

configuración de la aplicación) y que a su vez hereda las operaciones comunes a toda base de datos relacional (*AbstractSQLxxxDAO*).

A continuación (Ilustración 11 Diseño de un DAO), como ejemplo de uso del patrón explicado, se presenta en detalle el diseño de uno de los DAO del sistema (*bid.dao*). El resto han sido realizados de manera similar.

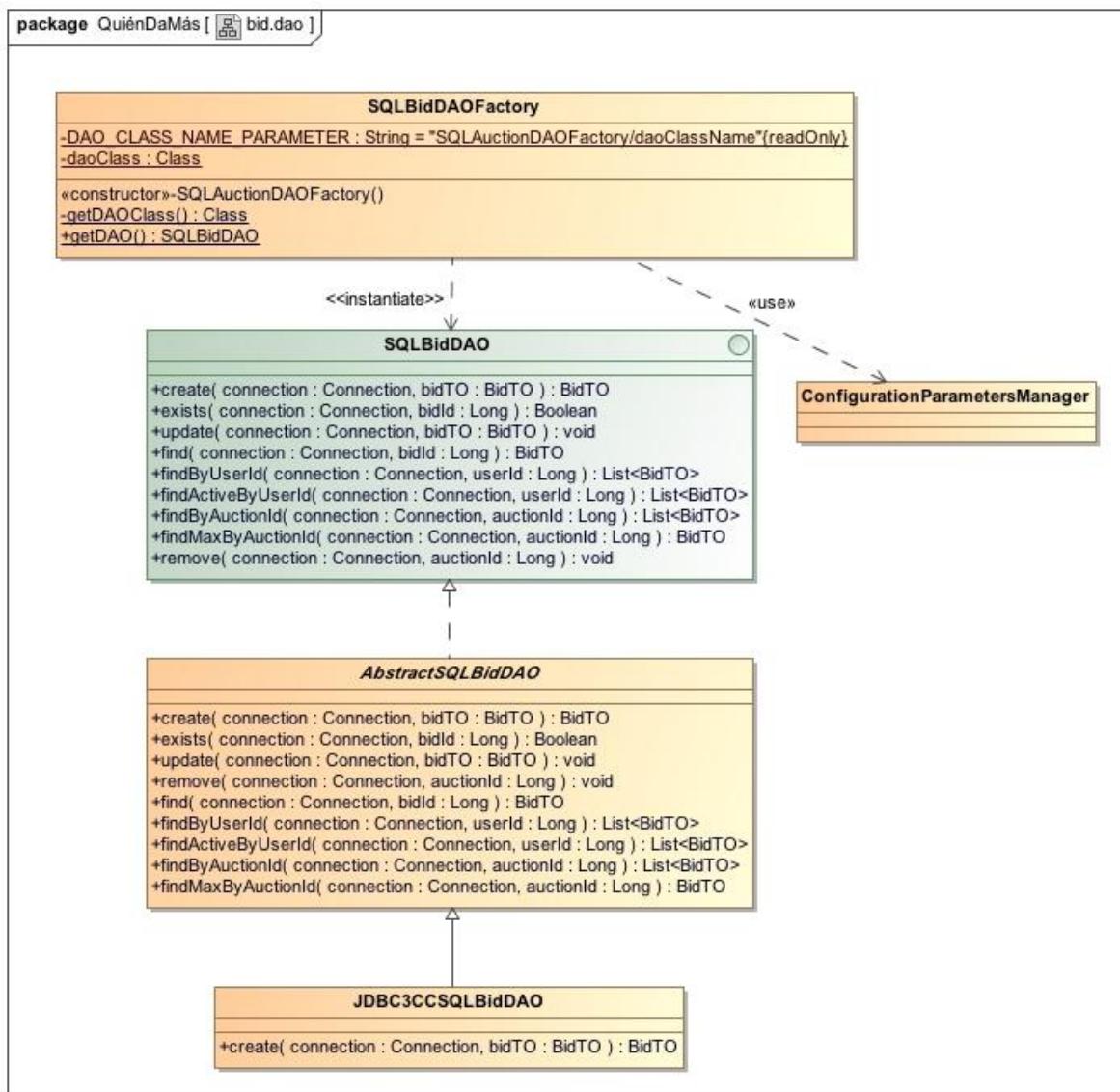


Ilustración 11 Diseño de un DAO

7.2.3.3. Interfaces de las Fachadas del Modelo

Una fachada (Facade) es un patrón que ofrece una interfaz sencilla para el manejo de un subsistema complejo, haciendo así el código externo dependiente de una única clase y no de cada una de las clases del subsistema.

A la hora de agrupar operaciones (casos de uso) en una fachada del sistema, se pueden seguir criterios diferentes pero debe procurarse que las funcionalidades que brinda la fachada estén relacionadas entre sí de forma que se puedan localizar intuitivamente de manera sencilla.

En colaboración con el patrón Facade se emplea también el patrón de diseño Business Delegate, cuya finalidad es ocultar las tecnologías empleadas en la implementación del modelo al resto de la aplicación.

Para esta aplicación se ha decidido emplear tres fachadas para agrupar las funcionalidades relacionadas con: categorías, usuarios y subastas:

- Category facade: casos de uso relacionados con las categorías que clasifican las subastas en el sistema.
- User facade: casos de uso relacionados con la gestión de usuarios y servicio de mensajería privada.
- Auction facade: casos de uso de manejo de subastas y pujas.

En las siguientes imágenes (Ilustración 12 Fachada de categorías, Ilustración 13 Fachada de usuario) se explica de forma detallada la interfaz de las fachadas de categorías y usuarios, incluyendo los *Custom Transfer Objects* (7.2.3.5.3 Custom Transfer Objects) y excepciones de cada una. La fachada de subastas se explicará al completo en el apartado siguiente.

7.2.3.3.1. Fachada de categorías

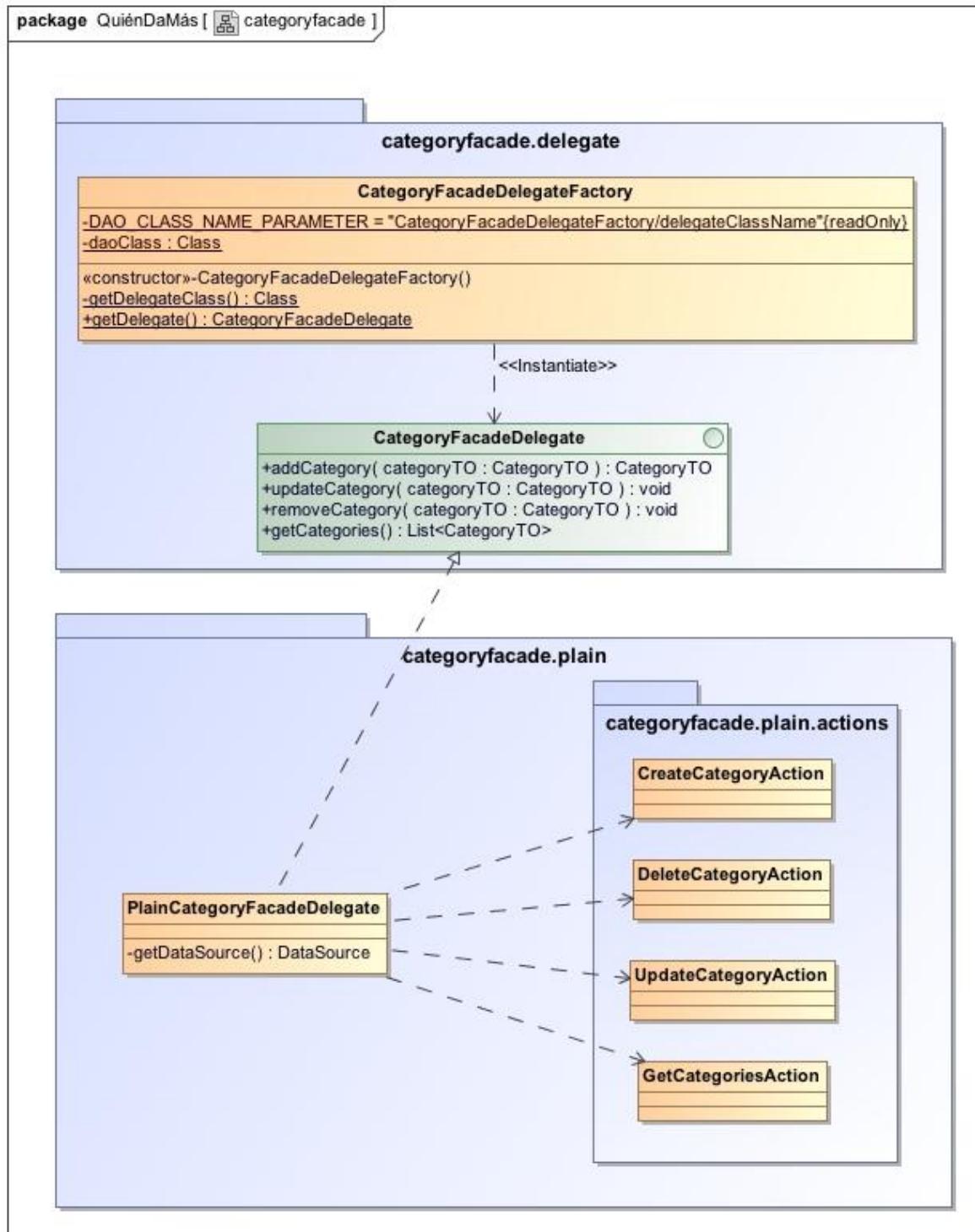


Ilustración 12 Fachada de categorías

7.2.3.3.2. Fachada de usuarios

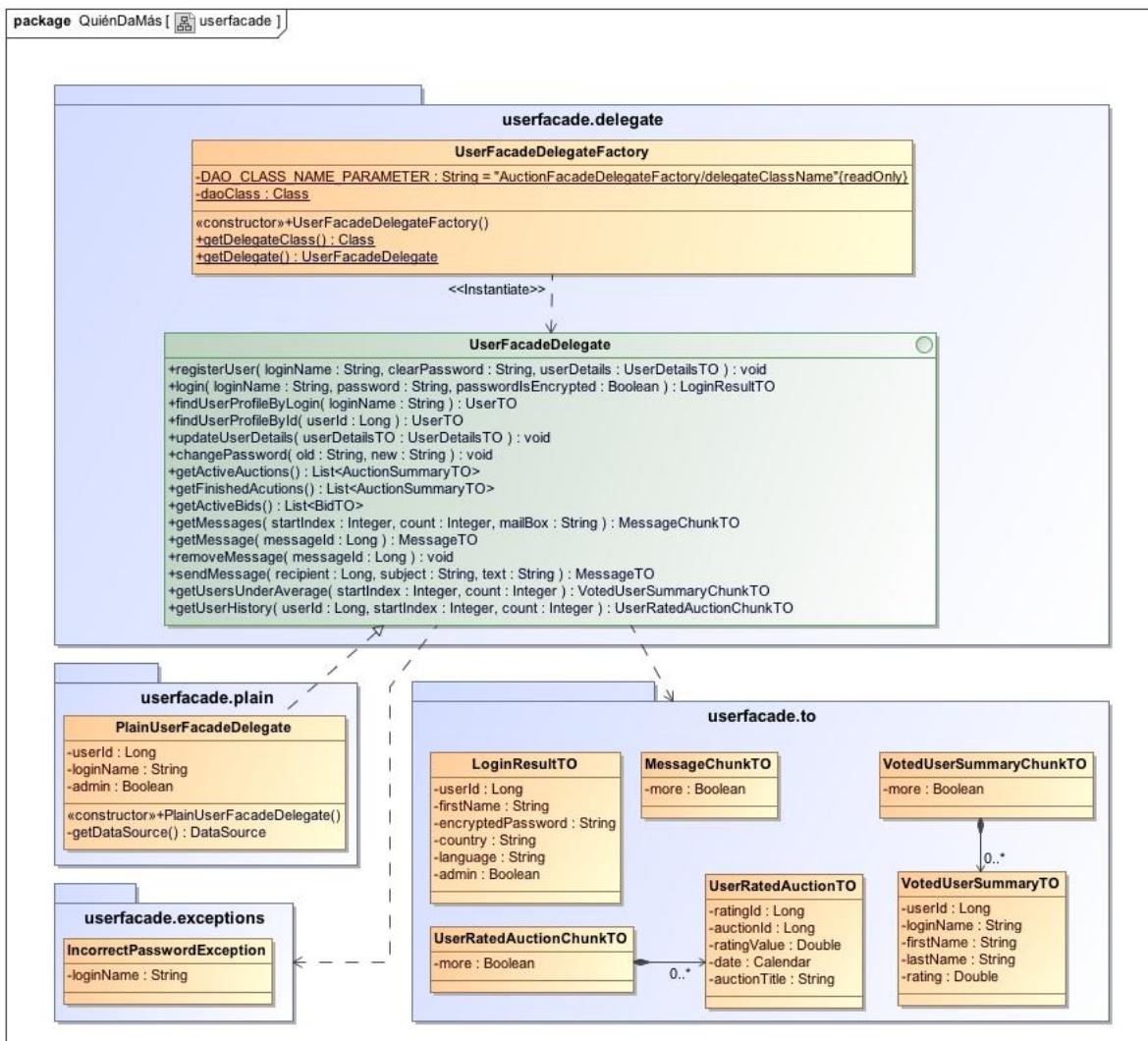


Ilustración 13 Fachada de usuario

7.2.3.4. Diseño de una fachada

En los diagramas de clases siguientes (Ilustración 14 Fachada de subastas, Ilustración 15 Acciones de la fachada de subastas) se detalla el diseño de la fachada de subastas (Auction Facade) incluyendo los *Custom Transfer Objects* que emplea, las excepciones, las acciones asociadas a cada operación con atributos y métodos, etc.

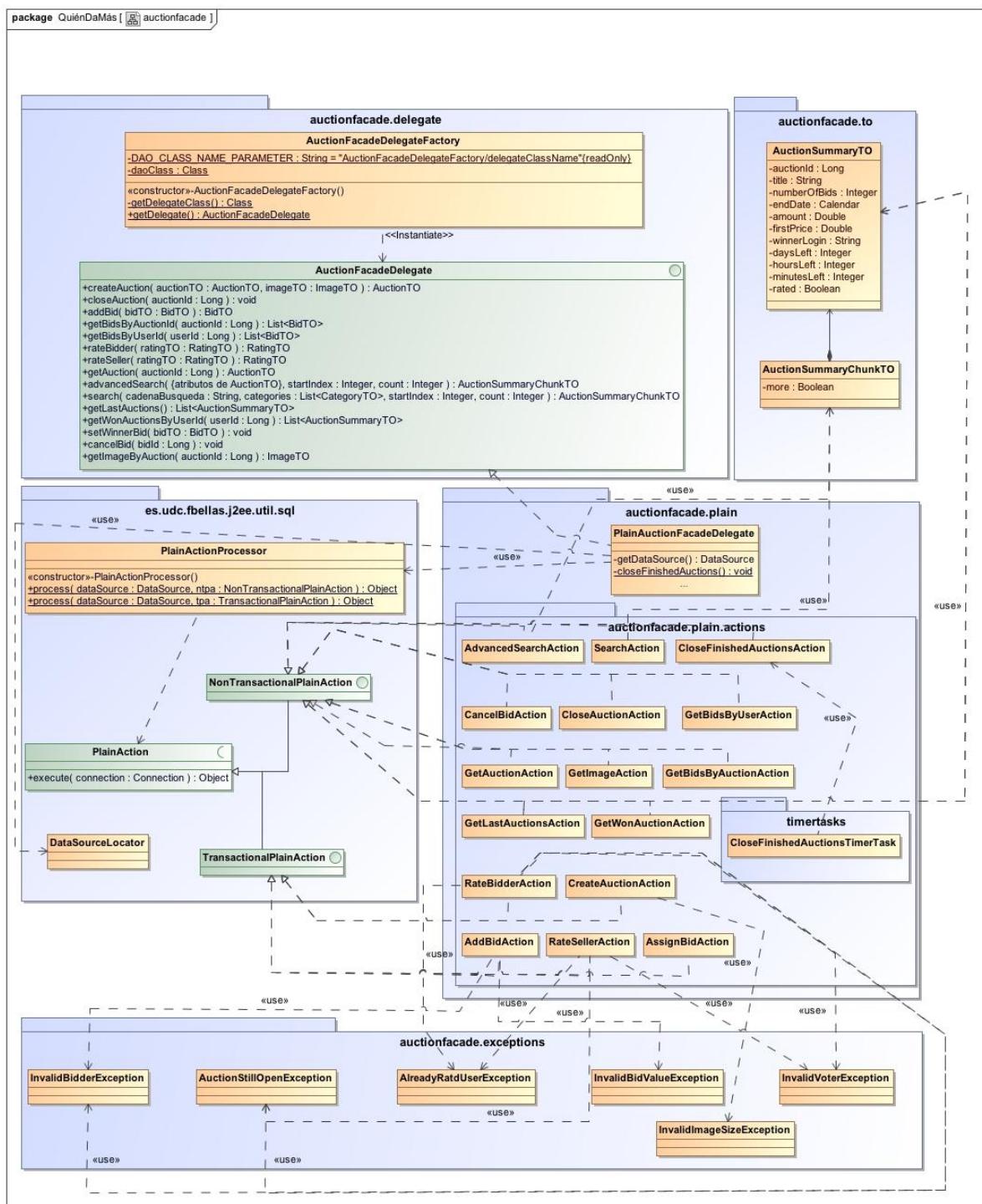


Ilustración 14 Fachada de subastas

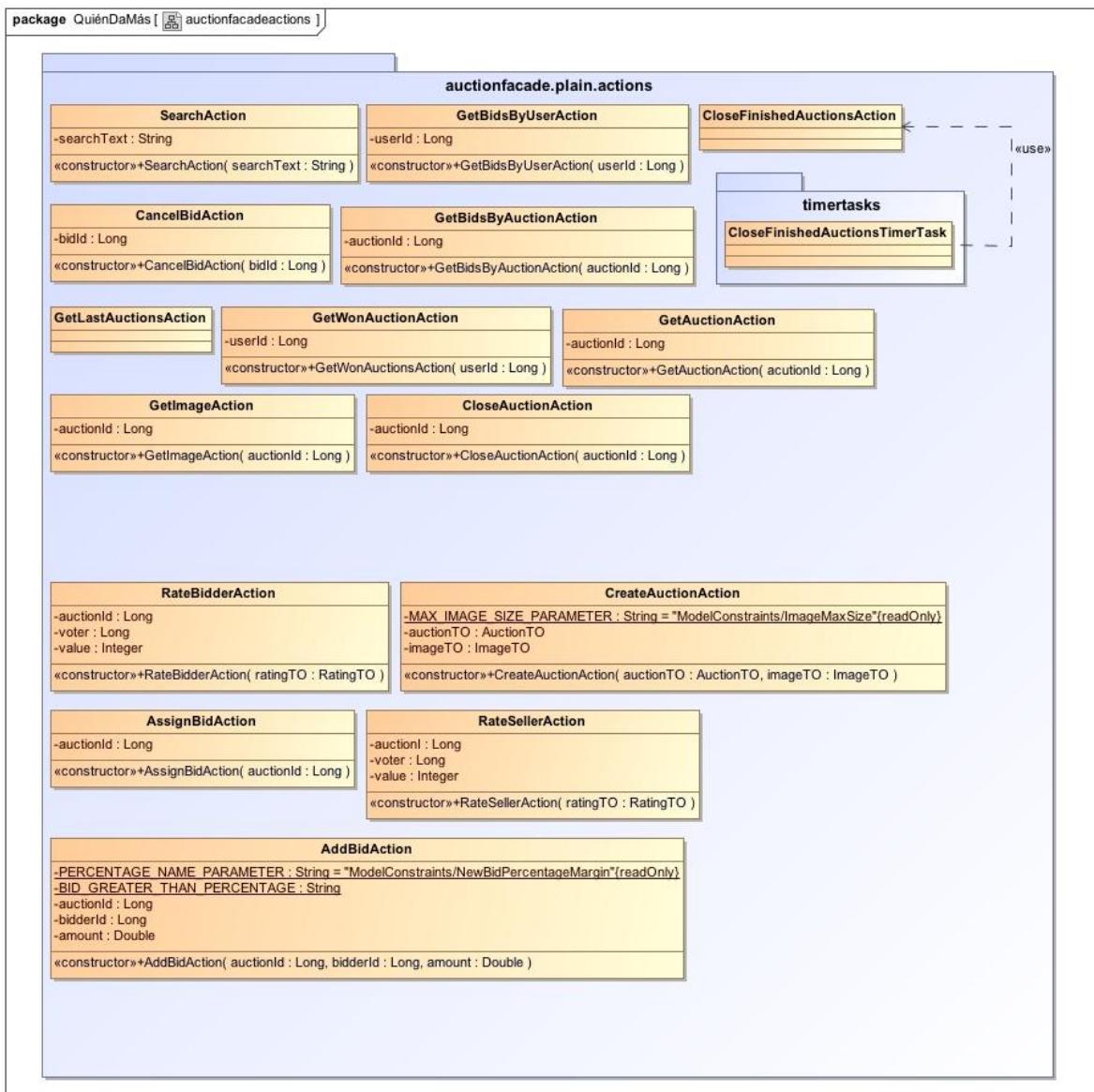


Ilustración 15 Acciones de la fachada de subastas

A continuación se presenta un diagrama de secuencia del caso de uso Realizar puja (Ilustración 16 Diagrama de secuencia de Realizar puja), restringido a la capa modelo, de forma que comienza con la llamada del controlador a la fachada y termina con el retorno al mismo.

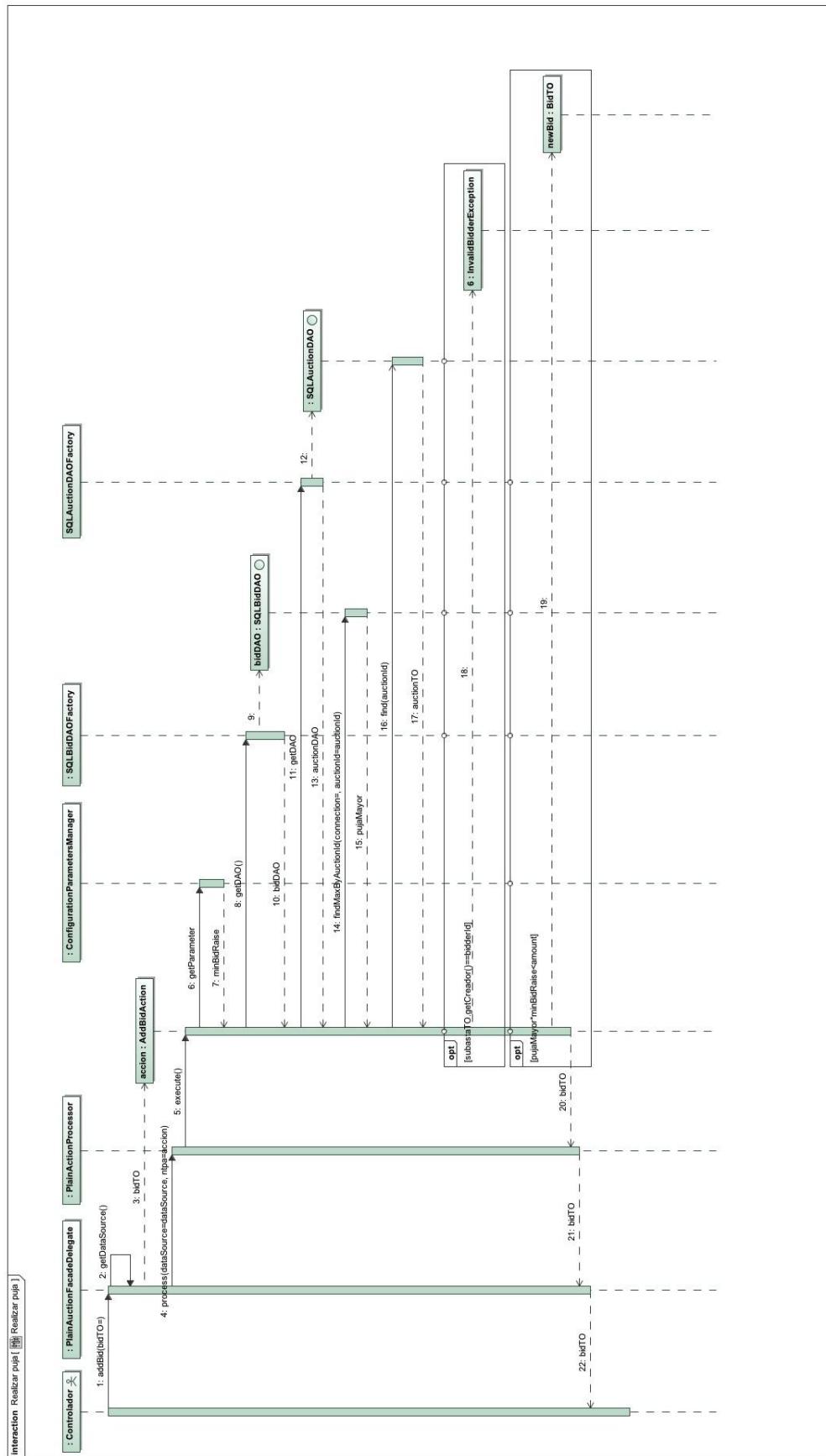


Ilustración 16 Diagrama de secuencia de Realizar puja

7.2.3.5. Otros aspectos

7.2.3.5.1. Thread para cerrar subastas

Como se mencionó en el apartado 6.3.4, el caso de uso Cerrar subastas finalizadas es lanzado de manera automática por el sistema una vez por segundo. Para conseguir este comportamiento se ha recurrido a las clases *Timer* y *TimerTask* (Ilustración 17 Diagrama de clases de Cerrar subastas finalizadas) que pertenecen al API de Java y permiten, de manera sencilla, establecer un hilo en paralelo que ejecute un código en intervalos de tiempo de X milisegundos.

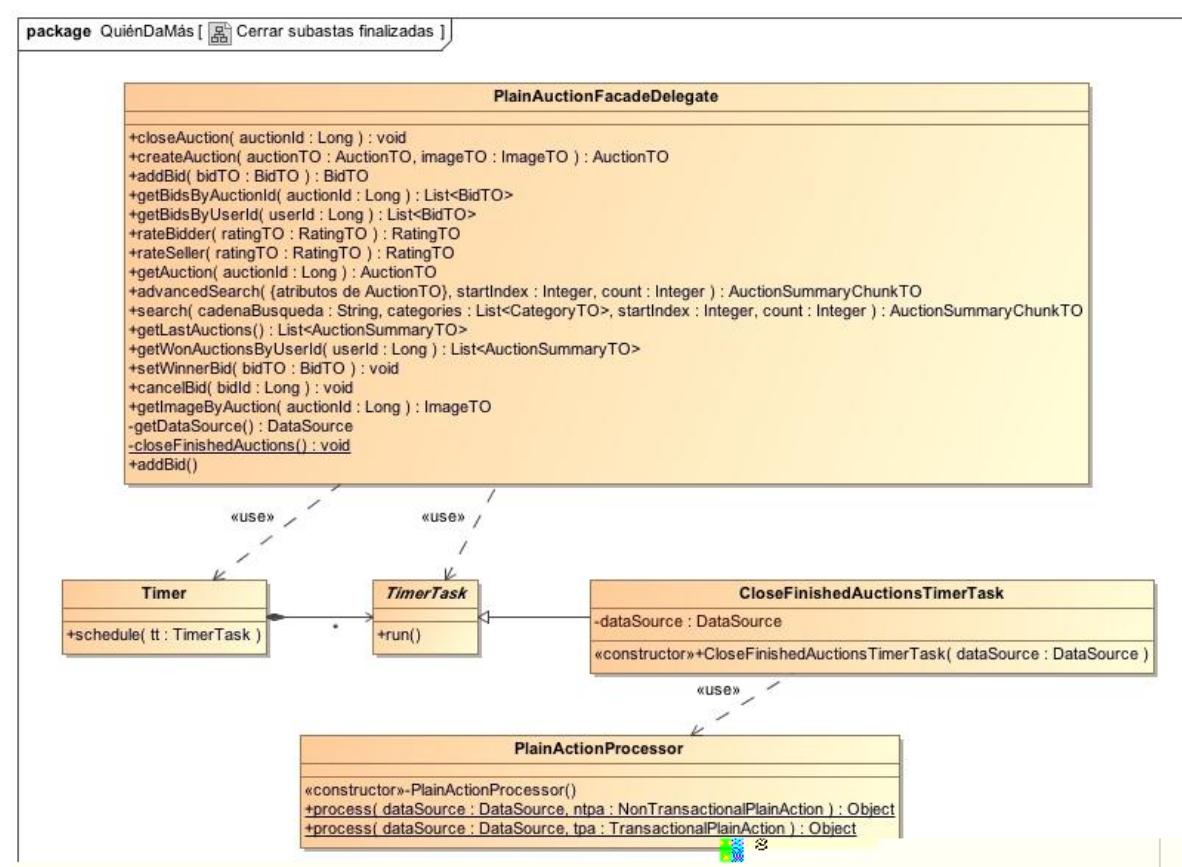


Ilustración 17 Diagrama de clases de Cerrar subastas finalizadas

El lanzamiento de este *thread* es realizado en un bloque estático de la fachada de subastas, por ser a la que pertenece al caso de uso, aunque podría haberse hecho desde otras partes del código que se ejecuten al iniciar. La primera idea fue realizar la llamada desde un plugin de Struts pero se desechó para evitar el solapamiento de las capas Model y Controller.

7.2.3.5.2. *Page-by-page iterator*

En la mayoría de aplicaciones son típicos los casos de uso que retornan N resultados que pueden ir desde cero a varios cientos o miles. Un ejemplo típico son las búsquedas en una base de datos o la generación de informes.

Cuando se realiza una consulta de este tipo, normalmente, el usuario no quiere recibir todo el conjunto de resultados de golpe, sino que es más usable una presentación de los mismos basada en rangos. Además de la usabilidad de cara al usuario, un aspecto más importante si cabe es el rendimiento de la aplicación, que se vería comprometido teniendo que manejar grandes cantidades de TOs procedentes de la base de datos.

Por medio de la utilización de este patrón, se especifica el rango de resultados que se devolverán para cada llamada a la fachada, permitiendo así al usuario navegar adelante y atrás de manera sencilla.

En esta aplicación, se ha considerado necesaria la utilización del patrón Page-By-Page Iterator en los casos de búsquedas de subastas, recuperación de los mensajes de usuario, recuperación de usuarios votados negativamente y recuperación de valoraciones a usuarios.

7.2.3.5.3. *Custom Transfer Objects*

En algunas ocasiones los datos brindados por un Transfer Object son insuficientes o, por el contrario, son demasiados para realizar alguna operación que realmente no los necesita. En ambos casos debe recurrirse a la creación de un Custom Transfer Object que permita la personalización de los datos a emplear.

En el caso de que sean insuficientes, puede ser necesario añadir algunos campos calculados o campos de algún otro TO para facilitar alguna tarea al sistema. En el caso de que los campos sean excesivos para la función a realizar, también debe crearse un TO personalizado para así dotar al sistema de una mayor eficiencia haciendo que maneje únicamente los valores necesarios. Esto último es especialmente importante en los casos de listas de TOs.

En este proyecto, los Custom Value Objects creados son los descritos a continuación:

- *VotedUserSummaryTO*: utilizado en las búsquedas de usuarios valorados, contiene los datos relevantes como el identificador, nombre de usuario y la media de valoración.
- *UserRatedAuctionTO*: se utiliza para la muestra del detalle de las valoraciones recibidas por un usuario apoyándose en datos como el nombre de la subasta en relación a la cual se recibió la valoración, la calificación obtenida o la fecha.
- *AuctionSummaryTO*: contiene los datos que se mostrarán en la búsqueda de subastas. En una búsqueda no es necesario mostrar datos como el subastador o la descripción detallada y sí son necesarios otros calculados como el número de pujas o la puja máxima realizada.
- *LoginResultTO*: empleado para la autenticación de usuarios puesto que no tendría sentido recuperar todos los datos del usuario ya que solo son necesarios la clave encriptada, el identificador, el login, el país, el idioma y el valor del campo admin.

7.2.3.5.4. Excepciones

- *IncorrectPasswordException*: identifica el error producido cuando un usuario introduce una contraseña incorrecta al iniciar sesión.
- *InvalidBidderException*: lanzada cuando el usuario que realiza la puja no puede hacerlo (por ejemplo, cuando un usuario pretende pujar en su propia subasta).
- *AuctionStillOpenException*: lanzada cuando se pretende realizar una operación que requiere que la subasta esté cerrada pero ésta figura todavía abierta.
- *AlreadyRatedUserException*: identifica el error producido cuando se pretende votar a un usuario más de una vez para una misma subasta.
- *InvalidBidValue*: lanzada cuando se puja con un valor inválido (por ejemplo, inferior a la puja máxima).

- *InvalidVoterException*: se produce cuando el usuario que realiza una valoración no está relacionado con el usuario valorado para la subasta especificada.
- *InvalidImageSizeException*: producida cuando el tamaño de la imagen que se pretende ligar a una subasta excede el máximo permitido.

7.2.4. Controlador

La capa Controlador (Controller) se encarga de mantener la separación entre el modelo (Model) y la vista (View), de forma que recibe las peticiones de la vista, consulta al modelo y ofrece una respuesta.

De la misma forma que se explicó la capa modelo mediante el diagrama de secuencia de una operación de la fachada, a continuación (Ilustración 18 Diagrama de secuencia Valorar comprador, Ilustración 19 Diagrama de secuencia de filtros del controlador) se expondrá la secuencia de operaciones (y las clases involucradas) que se lleva a cabo desde que el servidor recibe una petición hasta que se invoca la operación de la fachada correspondiente a la misma.

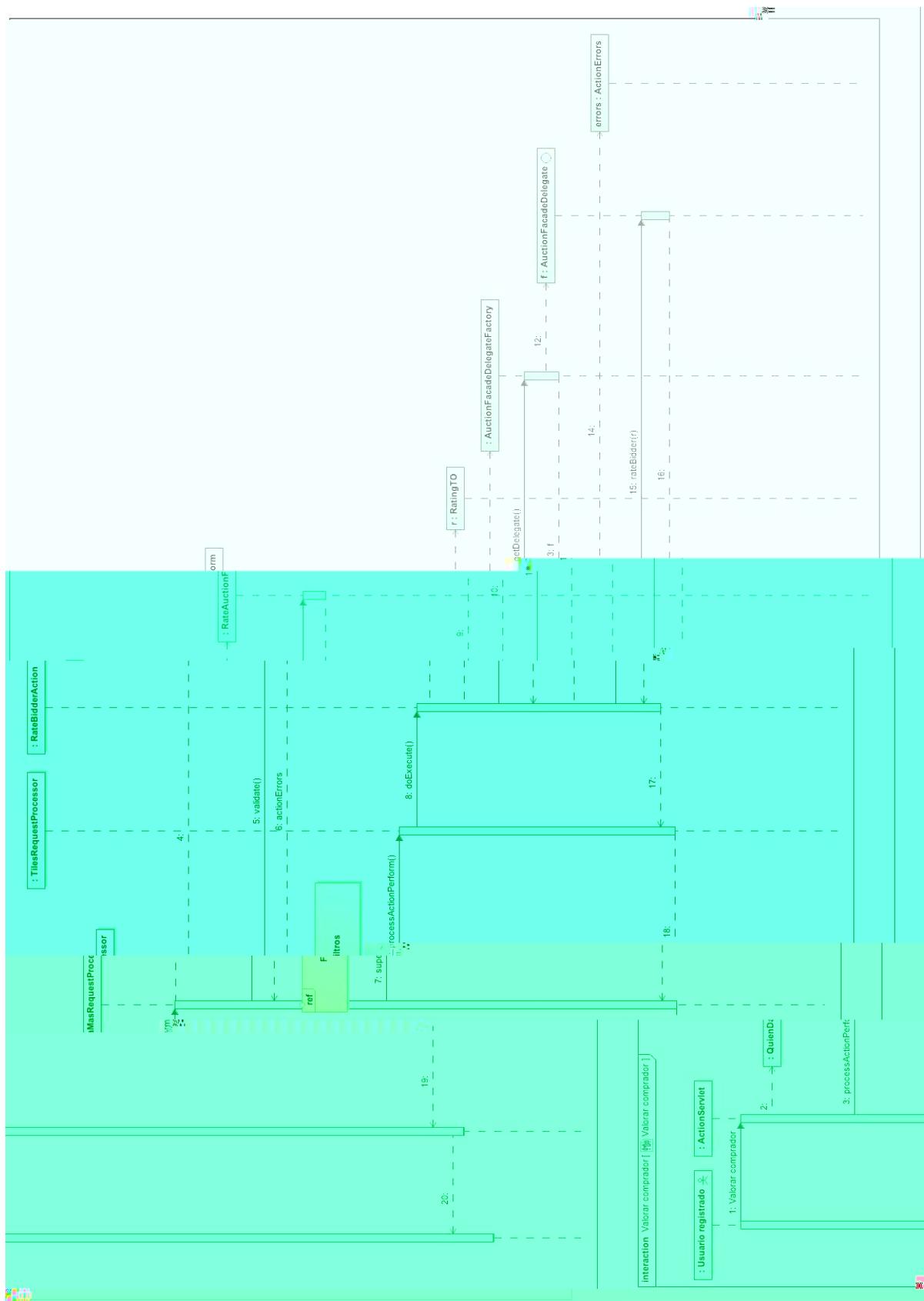


Ilustración 18 Diagrama de secuencia Valorar comprador

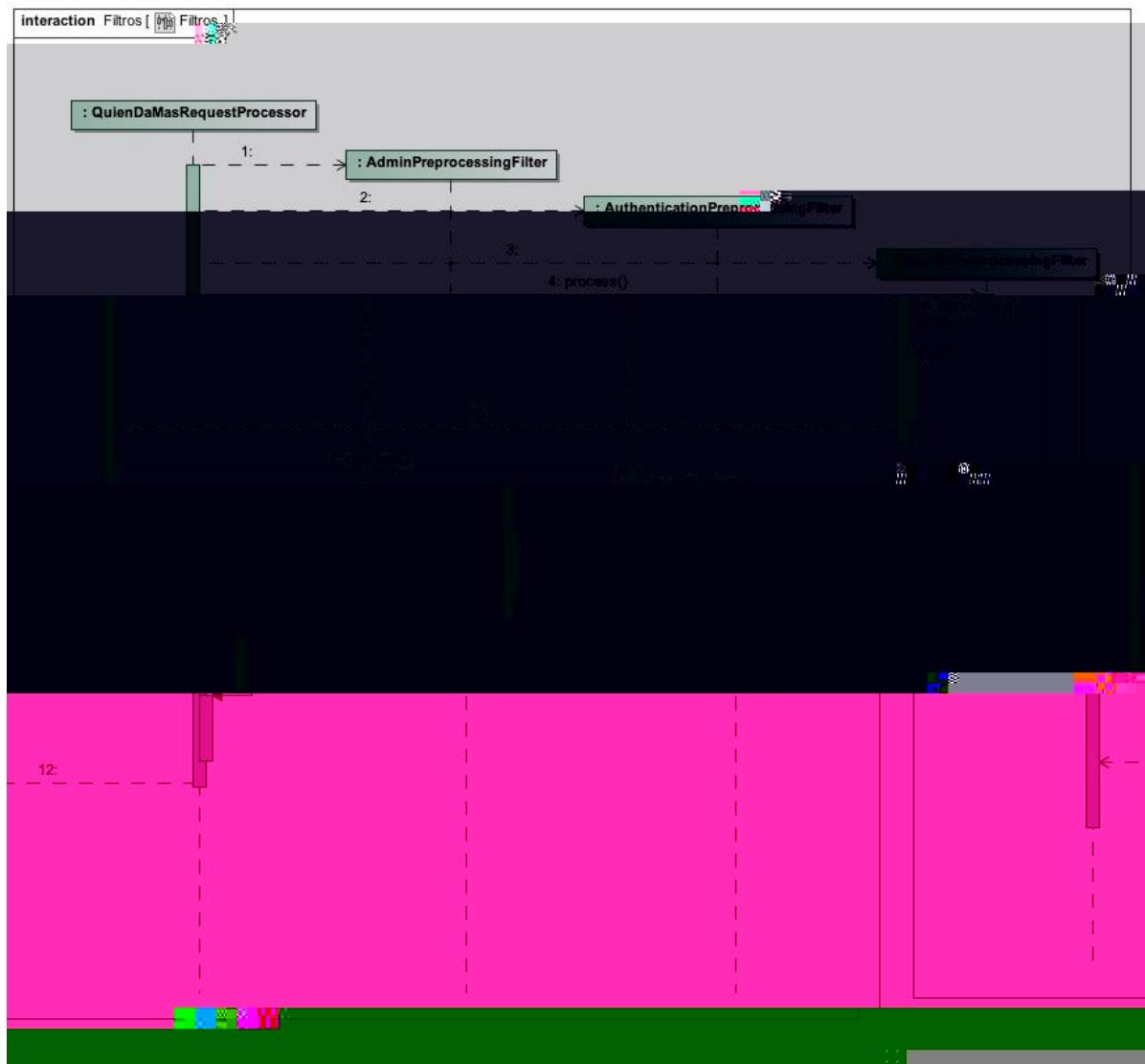


Ilustración 19 Diagrama de secuencia de filtros del controlador

7.2.4.1. Patrón Front Controller de Struts

El patrón *Front Controller* centraliza las peticiones recibidas por la aplicación, redireccionándolas, mediante el *RequestProcessor*, a la acción adecuada (Ilustración 20 Diagrama de clases del Front Controller de Struts). Para esto, en el archivo de configuración *web.xml* se especifica una extensión (.do es la elegida en este caso) con la que deberán terminar todas las URLs que deban ser procesadas por el *ActionServlet*.

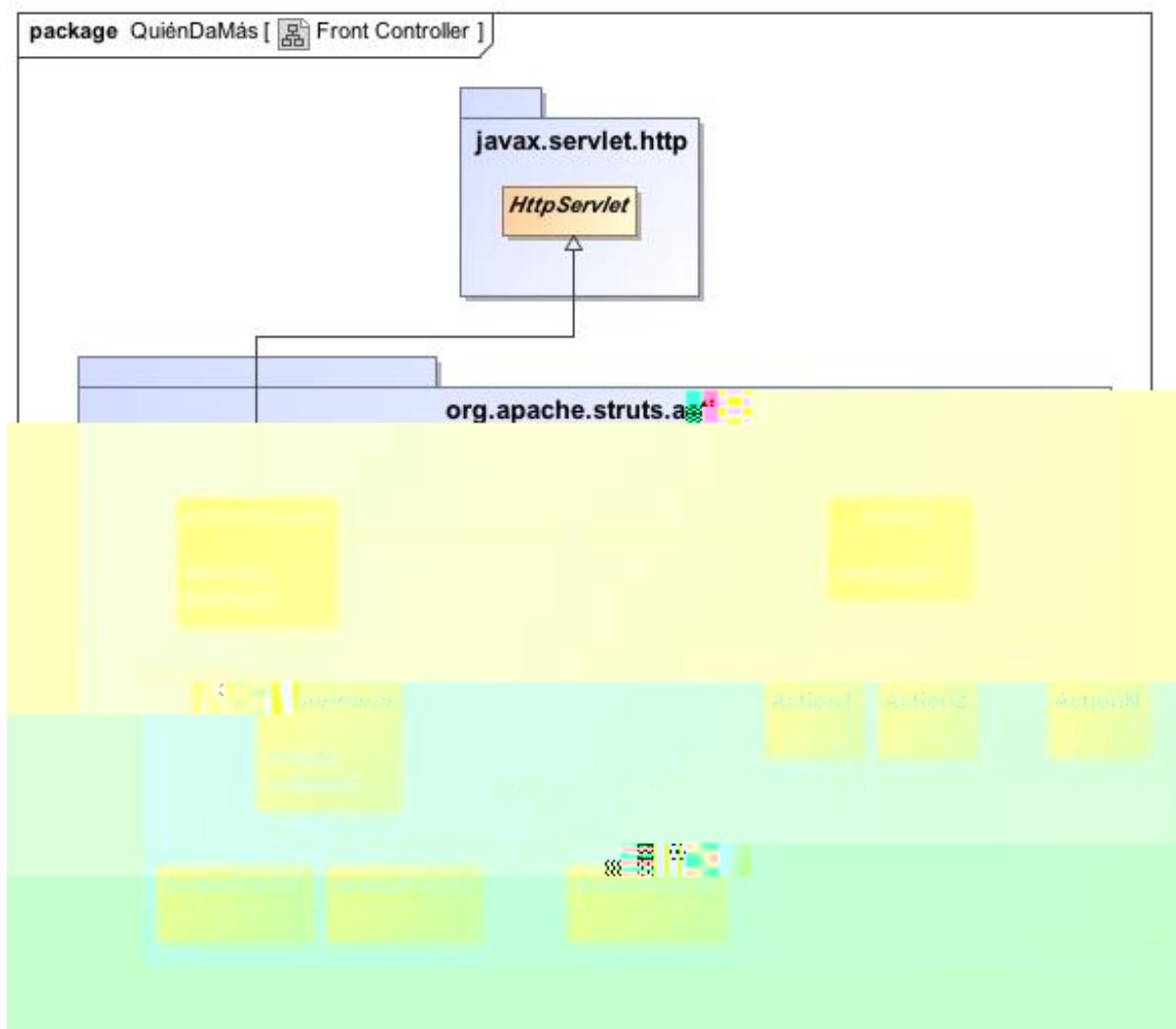


Ilustración 20 Diagrama de clases del Front Controller de Struts

7.2.4.2. Patrón Chain of Responsibility

Como su nombre indica, este patrón consiste en la cobertura de una serie de responsabilidades mediante la implementación separada de cada una de ellas. Así, una vez cumplido un requerimiento, la clase responsable del mismo llamará a la clase responsable del siguiente y así sucesivamente. De esta manera, se construye una cadena de responsabilidad que cubre los requerimientos fijados.

En este sistema, el patrón *Chain of Responsibility* se ha utilizado para la implementación de los filtros que deberán pasar todas las peticiones que recibe la aplicación. Éstos son los detallados a continuación:

- SessionPreprocessingFilter: mantenimiento de la sesión del usuario, comprueba si sigue existiendo, en caso contrario intenta recuperarla mediante *cookies*.
- AuthenticationPreprocessingFilter: verifica que el usuario se ha autenticado en el sistema previamente. Si no lo está, le redirecciona a la página de autenticación.
- AdminPreprocessingFilter: verifica que el usuario posee privilegios de administrador. Si no los posee, le redirecciona a una página de error.

7.2.4.3. Plugin de categorías

Para la gestión de categorías del sistema se ha decidido hacer una caché donde éstas se almacenen para evitar sobrecargar el SGBD con las peticiones de los usuarios. Las categorías se mostrarán en la página principal, en las *combobox* de los formularios de búsqueda y de los formularios de creación de subastas y en la página de administración de categorías, por este motivo, se ha considerado que, desde el punto de vista de la eficiencia, guardar el árbol de categorías en el *application scope* sería una optimización notable.

La clase del *plugin* es *CategoriesPlugin* que extiende de *org.apache.struts.action.PluginIn*. Al iniciar el *plugin*, con una sola consulta a la base de datos se obtienen las categorías del sistema y se ordenan en un árbol de manera jerárquica convirtiendo los *CategoryTO* en *TreeCategoryTO*.

Siempre que se realiza una caché de datos se corre el riesgo de tener inconsistencias en el sistema, en este caso, entre el árbol de categorías en memoria y el contenido de la base de datos. Para minimizar este problema se ha considerado oportuno hacer una recarga del árbol de categorías cada vez que un administrador modifique el estado de las categorías del sistema (adicción, modificación y borrado), ya que, aunque la creación del árbol puede ser costosa con un gran número de categorías, estos cambios no sucederán a menudo.

7.2.5. Vista

7.2.5.1. Tiles

Tiles es un componente de Struts que permite la gestión de plantillas en una aplicación web, de esta forma, se define un marco visual común que cumplirán todas las páginas mostradas sin necesidad de replicación de código ni escritura de m q n g u v q u " ñ k p e en cada una de las páginas JSP del sistema.

La forma en que funciona es creando una página por cada una de las partes que se quiere hacer común al aspecto de la aplicación. Así, para este proyecto se han creado:

- *DefaultLayout.jspx*: plantilla principal con la estructura general de las páginas f g " n c " c r n k e c e k » p " { " u g " j c e g p " n q u " ñ k p e multitud de *Layouts*.
- *DefaultPageTitle.jspx*: contiene la cabecera de la página con el nombre y el título de la aplicación.
- *DefaultFooter.jspx*: contiene el pie de página.

La configuración de las páginas se lleva a cabo con el fichero *tiles-defs.xml*.

7.2.5.2. Custom tags

JSP es una tecnología que permite la inserción de código Java en las páginas HTML, de esta forma se potencia el dinamismo de éstas. Además, proporciona la posibilidad de incluir tags con operaciones comunes para hacer las páginas más legibles a usuarios no familiarizados con el lenguaje (diseñadores). Para dar más potencia a esta tecnología, se permite también la creación de tags personalizados para incluir en las páginas.

Para dar soporte al árbol de categorías diseñado para la gestión de las mismas en las capas View y Controller, se han creado dos *custom tags* que mostrarán dicho árbol de manera amigable al usuario. Por un lado, el tag *PrintTree* lo imprime por pantalla en un documento HTML y por otro *ComboTree* lo imprime dentro de una combobox para utilizar en formularios.

Estos tags personalizados son fácilmente programables extendiendo la clase *javax.servlet.jsp.tagext.TagSupport* e implementando los métodos *doStartTag()* y

doEndTag(), creando el archivo .tld con la descripción de los tags creados e incluyendo la referencia al mismo en la página JSP que lo vaya a usar.

7.2.5.3. Formularios y validación

El framework Struts ofrece un sistema sencillo y potente para la recuperación y validación de los datos introducidos por los usuarios: extendiendo la clase *ActionForm* para cada formulario que necesite el sistema. De esta manera, se crean los atributos en la clase que recogerán los parámetros de la petición HTTP y se implementan los métodos *reset()* y *validate()* para borrar los datos del formulario y validarlos respectivamente.

La relación entre una acción y el formulario que le corresponde se realiza en el fichero *struts-config.xml*, en él, para cada acción se detalla el formulario que le corresponde (previamente definido) y si se invocará la validación del mismo o no.

7.2.5.4. Internacionalización

La internacionalización es un aspecto muy importante para un sistema escalable que debe ser planificado cuidadosamente. A la hora de presentar una aplicación localizada para un país e idioma concreto deben tenerse en cuenta varios puntos:

- 120.51.ET_BT/F2.12T6.10011559378.65Tm ((120.51.ET/B)4(sión)-52(de)4

idiomas de la interfaz se han empleado archivos que relacionan pares (clave, valor). De esta forma, se tiene un archivo para cada idioma al que se quiere dar soporte en el sistema, cada uno de ellos con todas las claves que figuran en el código de las JSP y el valor relacionado con la misma y dependiente de

7.2.5.5. Application objects

Para facilitar la construcción de la capa vista se ha recurrido a los objetos contenidos en el paquete `es.udc.ingamv00.quiendamas.http.view.applicationobjects`:

- *Countries*: relación de países soportados para la internacionalización del sistema, necesarios para ser mostrados en las listas de los formularios.
- *Languages*: relación de idiomas soportados para la internacionalización del sistema, necesarios para ser mostrados en las listas de los formularios.
- *DateCollections*: rangos de fechas que se usan en los formularios como los de creación de usuarios y subastas.

8. Implementación

8.1. Lista de defectos

Como se especificó en los primeros apartados de esta memoria, el proyecto realizado ha tenido una finalidad de aprendizaje para el desarrollador, esto motiva que, en diferentes fases del mismo, surjan problemas novedosos que deben ser analizados para la búsqueda de una solución, con ayuda de ejemplos citados y del tutor. La inexperiencia, en definitiva, sumada a la falta de tiempo, son los mayores problemas que un desarrollador se encuentra, los cuales motivan, en ocasiones, los defectos en la aplicación final.

Quizá un aspecto dejado de lado ha sido la importancia de las cachés. Si bien se ha realizado una caché para el árbol de categorías, en la página principal de la aplicación se presentan las subastas que están a punto de finalizar y podrían ser cacheadas para evitar un gran número de accesos a la base de datos. Por otra parte, esto se realizaría de manera similar a la caché de categorías por lo que no aportaba nada desde el punto de vista del aprendizaje.

Para este sistema, la desventaja más inmediata que un usuario extranjero puede encontrarse es la inexistencia de una internacionalización de categorías. En el momento del diseño no se consideró suficientemente importante por complicar el modelo y en las posteriores fases no hubo tiempo para considerar una adaptación.

8.2. Software requerido

Aunque hay multitud de posibilidades a la hora de escoger un software que proporcione soporte para las tecnologías elegidas, se ha pretendido potenciar el uso de aplicaciones libres y de gran difusión. En este sentido, el software empleado en el desarrollo de esta aplicación y su versión ha sido el siguiente:

- Java SE 1.6.
- Servidor de bases de datos MySQL 5.1.4.

- Servidor de aplicaciones Apache Tomcat 6.0.
- Gestor de proyectos Apache Maven 2.
- Diferentes librerías gestionadas por Maven.

8.3. Estructura

Puesto que se utiliza como gestor de proyecto Maven2, la estructura de directorios (Ilustración 21 Estructura de directorios) en los niveles más altos es la creada por defecto por este software. A niveles más bajos, se ha querido separar en función del patrón MVC.



Ilustración 21 Estructura de directorios

- *src/lib*: librerías webutil-2.2.0 y standardutil-2.2.0 proporcionadas en los ejemplos de la asignatura Integración de Sistemas.
- *src/main/java*: código fuente en lenguaje Java de la aplicación.
- *src/main/resources*: archivos de internacionalización (properties).

- *src/main/webapp/admin*: archivos JSP correspondientes a páginas de administración.
- *src/main/webapp/auction*: archivos JSP correspondientes a páginas de gestión de subastas.
- *src/main/webapp/commontiles*: archivos JSP que conforman la plantilla (Tiles) del sitio.
- *src/main/webapp/css*: archivos de hojas de estilo.
- *src/main/webapp/user*: archivos JSP correspondientes a páginas de gestión de usuarios.
- *src/main/webapp/WEB-INF/Struts*: archivos de configuración de Struts y de su *plugin Tiles*.
- *src/main/webapp/WEB-INF/tlds*: archivo descriptor de la librería de Custom Tags realizada.
- *src/main/sql*: archivos de código SQL para la inicialización de la base de datos desde Apache Maven.
- *src/main/test*: clases de prueba de la aplicación y archivo de parámetros de configuración para la ejecución de pruebas sin JNDI.
- *target*: contiene la aplicación final empaquetada en formato WAR.

8.4. Instrucciones de compilación

El uso de Apache Maven como gestor de proyectos facilita en gran medida la compilación y despliegue de la aplicación realizada. Para la compilación del proyecto es necesario tener instalado y en ejecución el servidor de bases de datos (12.1.3 Instalación de MySQL) y proceder desde una consola en el directorio del proyecto de la siguiente forma:

- Instalación en el repositorio local de las librerías proporcionadas en los ejemplos de la asignatura Integración de Sistemas:

\\$ mvn install:install-file -Dfile=src/lib/standardutil-2.2.0.jar -
DgroupId=es.udc.fbellas -DartifactId=standardutil -Dversion=2.2.0 -
Dpackaging=jar

\\$ mvn install:install-file -Dfile=src/lib/webutil-2.2.0.jar -
DgroupId=es.udc.fbellas -DartifactId=webutil -Dversion=2.2.0 -
Dpackaging=jar

- Inicialización de la base de datos

\\$ mvn sql:execute

- Compilación del código

\\$ mvn compile

- Generación del archivo final para la instalación en el servidor

\\$ mvn package

9. Pruebas

Las pruebas de software consisten en procesos que permiten validar y verificar un sistema, debe hacerse énfasis en las mismas durante las primeras fases del ciclo de vida porque cuanto antes se detecten los errores menos costosa es su corrección.

Debe aclararse que las metodologías de pruebas están orientadas a probar la existencia de errores y no la ausencia de ellos, lo que se persigue es detectar el mayor número de ellos posibles en la fase de diseño/desarrollo para subsanarlos cuanto antes.

En lo que respecta a este proyecto, se han realizado pruebas de unidad de cada una de las operaciones definidas en las tres fachadas del sistema (categorías, subastas y usuarios) con la ayuda de la herramienta libre JUnit, programada y orientada a la realización de pruebas de unidad en lenguaje Java.

Para probar una clase con JUnit se crea otra con un método por cada uno de la clase a probar, estos métodos de test llevarán la anotación `@Test` y contendrán el código necesario para la cobertura de los casos de prueba requeridos. Un método de prueba será superado satisfactoriamente cuando no lance una excepción no esperada y cumpla las aserciones incluidas en su código (`assertEquals()`, `assertFalse *` + . í + 0 "

Con la finalidad de clarificar el código de prueba, se ha definido también una fachada de tests análoga a cada clase de prueba, de forma que sea ésta la que interactúe con los DAOs y no se muestre ese código en los métodos de prueba sino que se realice una simple llamada a la fachada de test (Ilustración 22 Ejemplo de diagrama de pruebas para una fachada).

Como complemento a los tests de unidad, se han realizado unas pruebas poniendo la aplicación accesible a un reducido grupo de personas que se han registrado e interactuado con el sistema. La finalidad de esta última prueba ha sido la validación del sistema como conjunto y el acercamiento a una situación de uso real.

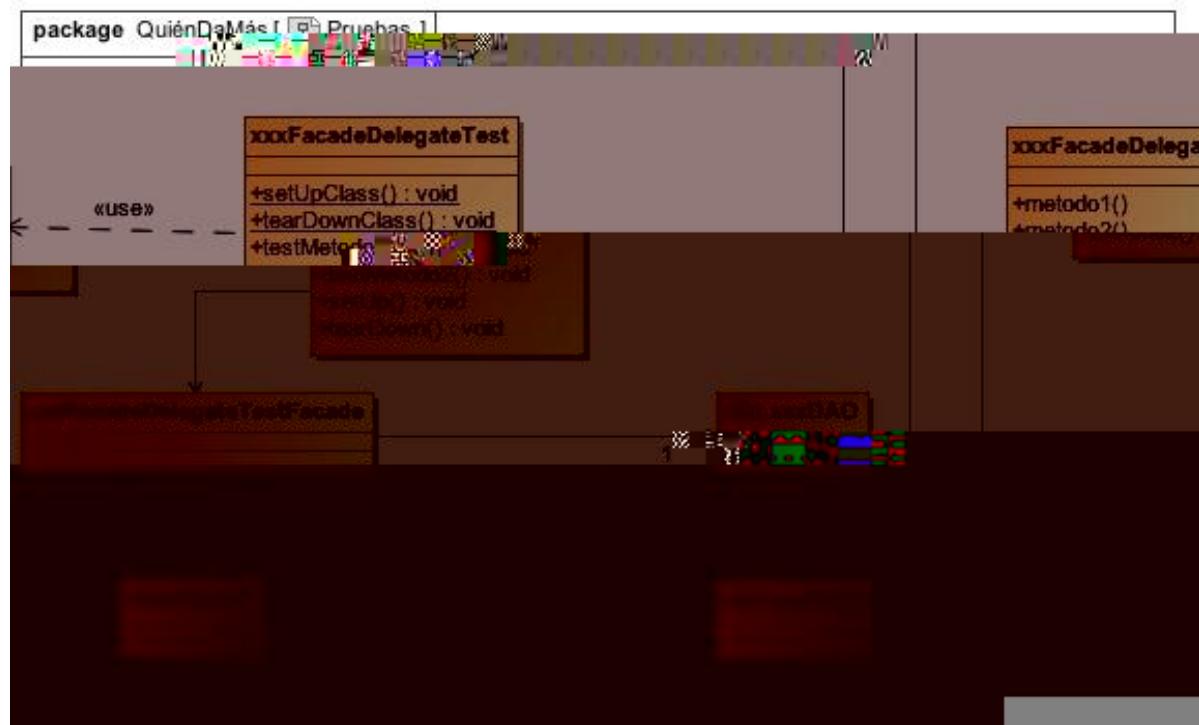


Ilustración 22 Ejemplo de diagrama de pruebas para una fachada

En el fichero *src/test/resources/ConfigurationParameters.properties* se especifica la configuración de DAOs, fachadas y acceso a la base de datos que, en una ejecución normal de la aplicación, se obtendrían mediante JNDI.

Cabe destacar también que para la realización de las pruebas se ha decidido emplear una base de datos distinta a la de la aplicación, de esta forma no interfieren una con otra y las pruebas puedan ser ejecutadas en cualquier momento, a pesar de que la aplicación ya haya sido arrancada con anterioridad y contenga datos de valor.

10. Planificación y Evaluación de Costes

Siguiendo las fases del Proceso Unificado, se ha definido una planificación en tres trimestres distinguiendo los roles de analista y programador. Como se introdujo en el apartado 5 de esta memoria, la fase de construcción de la aplicación se ha dividido en tres iteraciones, cada una de las cuales aporta funcionalidad extra al sistema.

Por tratarse éste de un proyecto de fin de carrera cuya finalidad es el aprendizaje, se dan algunas peculiaridades tanto en la forma de desarrollo como en la planificación. Por un lado, los dos recursos especificados (analista y programador) son en realidad la misma persona, por otro, la fase de transición (en la que se prepara el software para el entorno de producción final) no tiene sentido y se ha empleado en realizar correcciones, mejoras de la capa vista y pruebas extra.

C u k i p c p f q " c n " c p c n k u v c " { " c n " r t q i t c o c f q respectivamente y según la planificación descrita a continuación (Ilustración 23 Tabla de costes del proyecto, Ilustración 24 Diagrama de Gantt), el coste total de desarrollo de la aplicación es de **8.314 p**

Tarea	Inicio	Fin	Duración	Terminada	Dependencias	Coste total	Asignada
0) QuiénDaMás	17/03/09	18/12/09	39s 4d	100%		€ 8.314,00	
0% 1.1	€ 160,00	Análisis			1.1) Consulta de documentación	17/03/09	27/03/09
0% 1.1	€ 160,00	Análisis			1.2) Consulta de documentación	30/03/09	06/04/09

Ilustración 23 Tabla de costes del proyecto

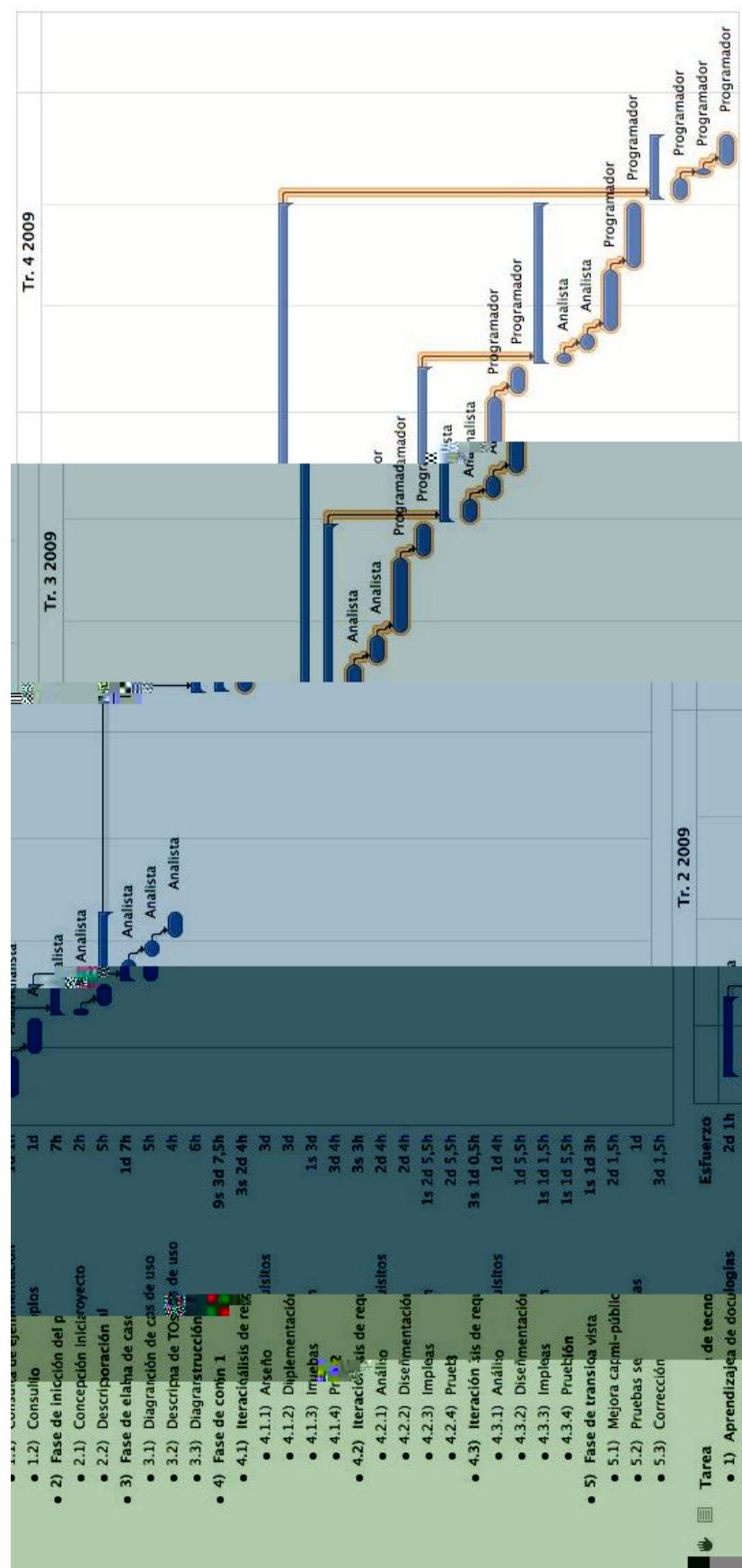


Ilustración 24 Diagrama de Gantt

Se ha considerado conveniente reflejar en el diagrama de Gantt, el hecho de que, después de la fase de elaboración (Abril), se realizó un alto en el desarrollo del proyecto, el cual retrasó el comienzo de la fase de construcción hasta el mes de Julio, debido a la proximidad de las fechas de los exámenes. De no hacer esta aclaración, los resultados de planificación y costes se verían deformados por asignar al desarrollo un tiempo y recursos que no existieron.

11. Conclusiones y Futuras Líneas de Trabajo

El objetivo inicial del proyecto consistía en la realización de una plataforma ágil de compra-venta de artículos por parte de personas con precios fijados por la demanda de los mismos. A este respecto, se ha implementado una aplicación perfectamente funcional que cumple los requisitos funcionales propuestos en la fase de Inicio.

Se ha implementado un sistema basado en el registro de usuarios que puede poner productos a la venta y enviarse mensajes privados. Así mismo, los usuarios pueden buscar y pujar por los artículos que le interesan de manera sencilla con la ayuda de la clasificación por categorías. Finalmente, también se cubrió la funcionalidad de valoraciones de usuarios.

En cuanto a las posibilidades de mejora de cara al futuro para la aplicación desarrollada, sin duda éstas son incalculables. Lo que se ha construido es un sistema base con las funcionalidades básicas para la gestión de subastas, sin embargo, como en la introducción se mencionó, las plataformas que están en el mercado van más allá de este concepto ofreciendo servicios de valor añadido a todos los niveles. Las más inmediatas líneas de trabajo a seguir serían las citadas a continuación:

- Envío email de activación para el registro de nuevos usuarios.
- Posibilidad de recibir notificaciones por email como avisos de eventos
 - * h k p c n k | c e k » p " f g " w p c " x g p v c . " u q d t g r w l c "
- Implementación de una funcionalidad para otorgar privilegios de administrador sin consultas directas a la base de datos.
- E t g c e k » p " f g " w p " c r c t v c f q " f g " õ r t g i w p v c u subastas, de esta forma, los usuarios podrían plantear las dudas que tengan sobre el producto subastado, permaneciendo éstas accesibles al resto de usuarios y evitando la repetición de preguntas por mensajes privados al vendedor.

12. APÉNDICE

12.1. Instalación del Software

Este software ha sido desarrollado en un entorno Unix, con el servidor de aplicaciones Apache Tomcat, el SGDB MySQL y el gestor de proyectos Apache Maven. A continuación se detallan los pasos de instalación del software necesario para la ejecución de la aplicación web:

12.1.1. Instalación de Maven 2

La instalación de Maven no tiene ninguna particularidad, lo único necesario es la descarga del software y desempaquetarlo en el directorio de instalación elegido.

- Desempaquetar el software en el directorio elegido:

```
Š tar ózvf maven2.0.7bin.tar.gz
```

12.1.2. Instalación de Apache Tomcat

Una vez descargado el software Apache Tomcat 6, se procede de la siguiente manera:

- Desempaquetar el software en el directorio elegido:

```
Š tar ózvf apache-tomcat-6.0.14.tar.gz
```

- Otorgar permisos de escritura a las carpetas *conf*, *logs*, *temp*, *webapps* y *work*.
- Modificación del archivo de configuración *conf/tomcat-users.xml*, añadiendo un usuario y rol para la administración del servidor.

```
<tomcat-users>
  <user name="tomcat" password="tomcat" roles="tomcat,manager" />
  <user password="tomcat" roles="manager,admin" username="tomcat" />
</tomcat-users>
```

Tabla 30 Archivo tomcat-users.xml

- Copiar el driver JDBC de MySQL (descargado automáticamente por Maven al repositorio local durante la compilación del proyecto) al directorio *lib*.
- Modificación del archivo *conf/server.xml*, añadiendo la fuente de datos global señalada a continuación:

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

Tabla 31 Global datasource en server.xml

- Modificar el archivo *conf/context.xml* añadiendo la siguiente línea dentro del tag <context>:

```
<ResourceLink name="jdbc/QuienDaMas" global="jdbc/QuienDaMas"
  type="javax.sql.DataSource"/>
```

Tabla 32 Resource link en context.xml

12.1.3. Instalación de MySQL

En primer lugar deberá descargarse el software MySQL, típicamente en un archivo tar.gz que será descomprimido e instalado como usuario *root*:

- Desempaquetar en el directorio elegido:

 \\$ tar -xzvf mysql-5.1.40-linux-i686-glibc23.tar.gz

- Otorgar permisos:

 \\$ chown -R root:root mysql-5.1.40-linux-i686-glibc23

 \\$ chmod Tf 3()] TJ ET BT 1 0 0 1 210149 T19.66 Tm [(c)] TJ ET BT 1 0 0 1 214.44

```
\$ ln -s `pwd`/mysql-5.1.40-linux-i686-glibc23 /usr/local/mysql
```

A continuación, con un usuario normal (ej.: testuser) se continua de la siguiente manera:

- Creación del directorio de bases de datos y fichero de configuración:

```
\$ mkdir /home/testuser/.MySQLData
```

```
\$ vi /home/testuser/.my.cnf
```

```
[mysqld]
datadir=/home/testuser/.MySQLData
```

Tabla 33 Contenido del archivo .MySQLData

- K p k e k c n k | c e k » p " f g " n c u " d c u g u " f g " f c v q u "

```
\$ cd /usr/local/mysql
```

```
\$ scripts/mysql_install_db
```

- Orden de arranque de MySQL:

```
\$ mysqld
```

- Creación de las bases de datos y usuarios:

```
\$ mysqladmin -u root create quiendamas
```

```
\$ mysqladmin -u root create quiendamastest
```

```
\$ mysql -u root
```

```
\$ GRANT ALL PRIVILEGES ON quiendamas.* to quiendamas@localhost
IDENTIFIED BY 'quiendamas', quiendamas@localhost.localdomain
IDENTIFIED BY 'quiendamas';
```

```
\$ GRANT ALL PRIVILEGES ON quiendamastest.* to
quiendamastest@localhost IDENTIFIED BY 'quiendamastest',
quiendamastest@localhost.localdomain IDENTIFIED BY
'quiendamastest';
```

En este momento se cuenta con dos bases de datos (quiendamas y quiendamastes) dos usos fijos que incluyen MySQL y PostgreSQL y una aplicación web.

12.1.4. Variables de entorno

El contenido del fichero .bashrc con las líneas correspondientes a las variables de entorno necesarias para las aplicaciones instaladas sería el siguiente:

```
# J2SE.
JAVA_HOME=/usr/java/jdk1.6.0_02
export JAVA_HOME
PATH=$JAVA_HOME/bin:$PATH

# Maven.
MAVEN_HOME=/usr/local/java/maven-2.0.7
export MAVEN_HOME
PATH=$MAVEN_HOME/bin:$PATH

# Tomcat.
CATALINA_HOME=/home/testuser/tomcat6
Export CATALINA_HOME PATH=$CATALINA_HOME/bin:$PATH

# MySQL.
MYSQL_HOME=/usr/local/mysql
export MYSQL_HOME
PATH=$PATH:$MYSQL_HOME/bin
```

12.2. Configuración de la aplicación

12.2.1. Constantes

Con la finalidad de otorgar mayor flexibilidad a la aplicación, se ha considerado oportuno permitir la configuración de algunos valores empleados en el código. Dicha configuración se realizará desde el fichero *web.xml*, ajustando los parámetros descritos a continuación:

- *NewBidPercentageMargin*: porcentaje en que una nueva puja deberá superar a la anterior para ser registrada en el sistema.
- *ImageMaxSize*: tamaño máximo en kilobytes de las imágenes de las subastas.
- *RatingThreshold*: umbral máximo de valoración para la consulta de usuarios votados negativamente.

12.2.2. Asignación de privilegios de administrador

Puesto que no se ha definido ningún caso de uso para la asignación de privilegios a los usuarios, ésta debe realizarse directamente en la base de datos mediante la siguiente sentencia UPDATE, substituyendo *nombre_de_usuario* por el login del usuario al que se pretenden otorgar los privilegios.

WR F C V G " w u g t u " U G V " c f o k p o m b r e _ d e _ u s u a r i o " Ø Y J G T G " n q i

12.3. Manual de Usuario

12.3.1. Navegación anónima

La página principal de la aplicación (Ilustración 25 Página principal) muestra en la parte superior (junto con la cabecera) un enlace para iniciar sesión en el sistema y, bajo este, el formulario básico de búsqueda. A la izquierda se presenta la lista de categorías bajo las que se pueden clasificar los artículos y a la derecha la lista con las subastas más próximas a finalizar.

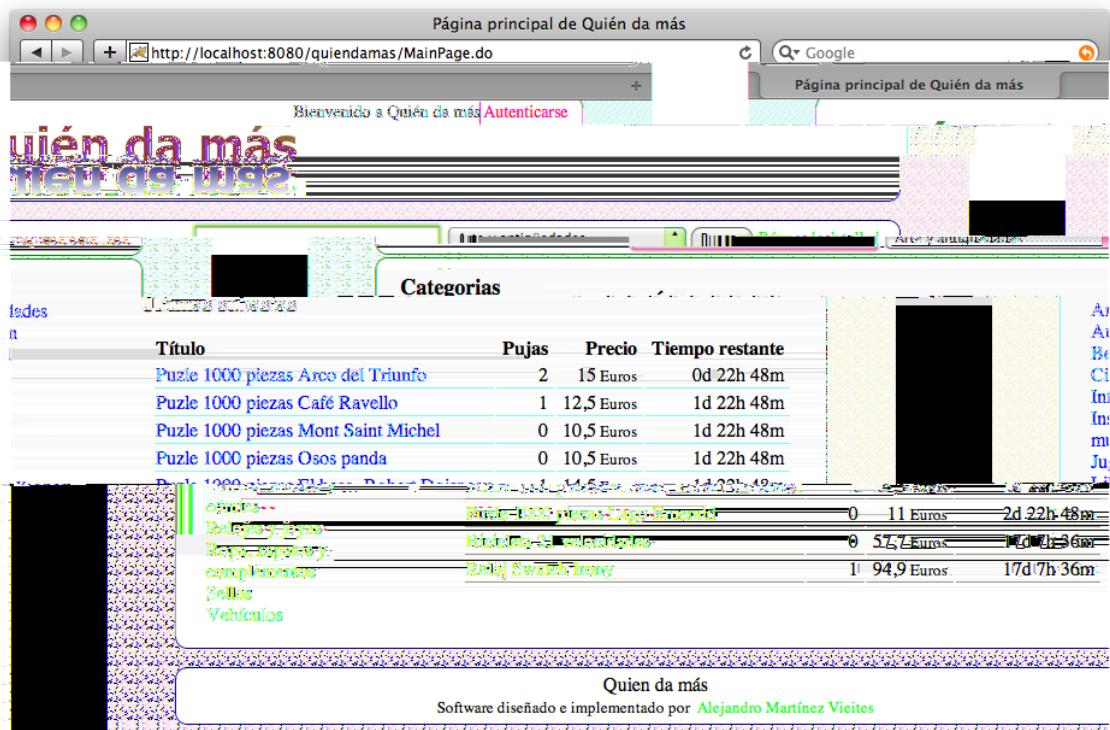


Ilustración 25 Página principal

Pulsando en cualquiera de los títulos de las subastas, se accede a la información detallada de cada una (Ilustración 26 Detalle de subasta), que reúne los datos de la misma, la descripción del producto, las pujas registradas, etc.



Ilustración 26 Detalle de subasta

Siguiendo el enlace del nombre de usuario del vendedor, se accede a la página del perfil de usuario del mismo (Ilustración 27 Perfil de usuario).

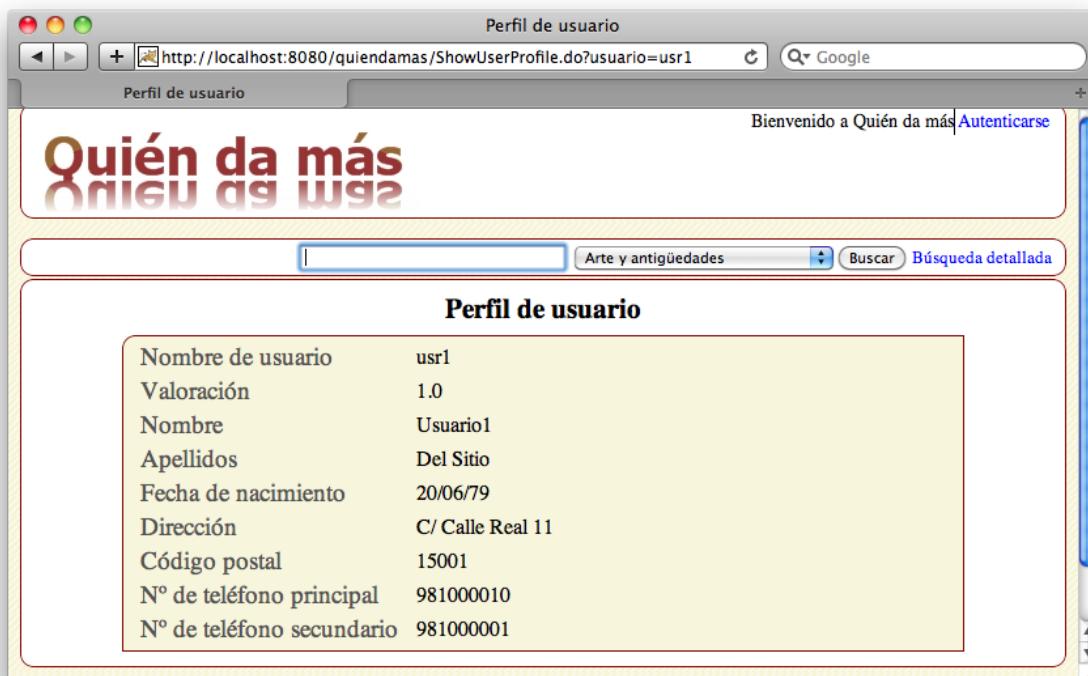


Ilustración 27 Perfil de usuario

Además del acceso directo a una subasta desde la página principal, es posible realizar búsquedas directamente desde el formulario destinado a tal efecto en la parte superior de cualquiera de las páginas de la aplicación. Así mismo, también está disponible un formulario de búsqueda detallada (Ilustración 28 Búsqueda detallada) que permite definir más aspectos de la búsqueda.



Ilustración 28 Búsqueda detallada

Una vez realizada la búsqueda, se presentará una nueva página (Ilustración 29 Resultados de la búsqueda) con la información más importante de cada una de las subastas que cumplen el criterio especificado. En el caso de que el número de resultados sea alto aparecerá en la parte inferior un enlace para consultar el resto de resultados.



Ilustración 29 Resultados de la búsqueda

Desde la lista que figura a la izquierda de la página principal, es posible navegar por el árbol de categorías del sistema (Ilustración 30 Subcategorías) hasta llegar a alguna que no tenga categorías hijas y se muestren las subastas que agrupa.

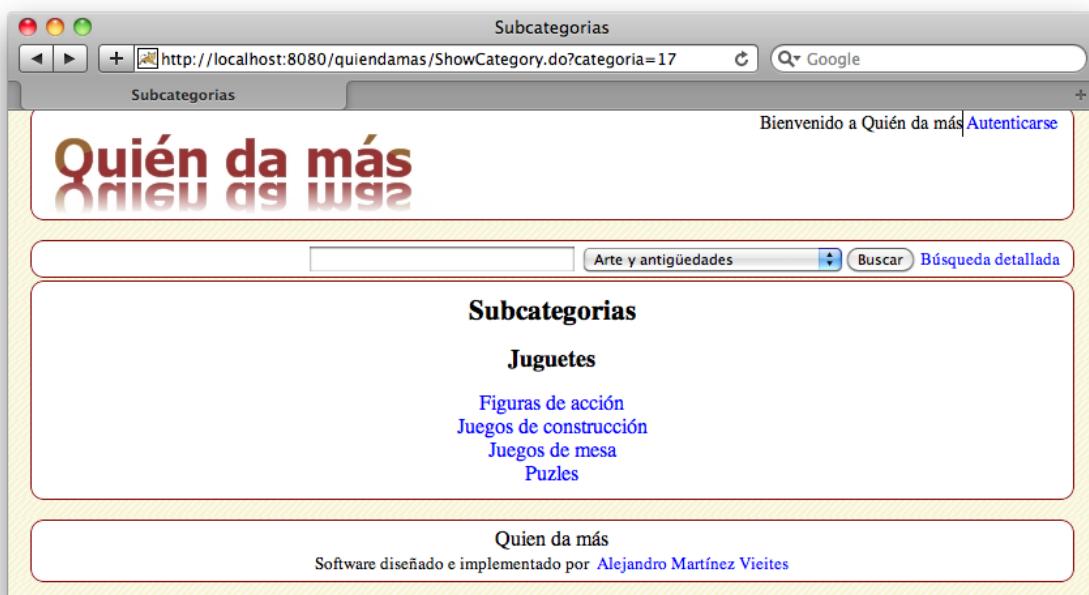


Ilustración 30 Subcategorías

Mediante el citado enlace de autenticación situado en la parte superior de la página, un usuario puede acceder al formulario de inicio de sesión (Ilustración 31 Formulario de inicio de sesión) en el sistema. En caso de no contar con una cuenta de usuario, existe la posibilidad de crear una mediante el enlace de registro que redirige al formulario correspondiente (Ilustración 32 Formulario de creación de cuenta).



Ilustración 31 Formulario de inicio de sesión

The screenshot shows a web browser window with the URL <http://localhost:8080/quierendas/EditUserProfileForRegistration.do?action=1>. The page title is "Formulario de registro de usuarios". The header includes the website logo "Quién da más", a welcome message "Bienvenido a Quién da más Autenticarse", and navigation links "Arte y antigüedades", "Buscar", and "Búsqueda detallada". The main content area is titled "Registro de usuarios" and contains two side-by-side boxes. The left box is titled "Datos personales" and includes fields for Nombre, Apellidos, Dirección correo-e, Dirección, Código postal, N° de teléfono principal, N° de teléfono secundario, Fecha de nacimiento (with dropdown menus for day, month, and year), Idioma (with a dropdown menu for Spanish), and País (with a dropdown menu for Spain). The right box is titled "Datos de la cuenta de usuario" and includes fields for Nombre de usuario, Contraseña, and Confirmar contraseña, followed by a "Registrar" button.

Ilustración 32 Formulario de creación de cuenta

12.3.2. Creación de subastas y panel de usuario

Una vez el usuario se autentica en el sistema, puede acceder al formulario de creación de subastas (Ilustración 33 Formulario de creación de subasta). Será necesario cumplimentarlo indicando todos los campos y fijando la fecha de finalización de la misma según las necesidades particulares de cada usuario y producto.

The screenshot shows a web browser window for 'Crear una subasta' (Create an auction). The URL is <http://localhost:8080/quiendamas/EditCreateAuction.do>. The page title is 'Crear una subasta'. At the top right, there are links: 'Hola usr1', 'Crear subasta', 'Mi QuiénDaMás', and 'Salir'. Below the title, there's a search bar with dropdowns for 'Arte y antigüedades' and 'Buscar', and a link 'Búsqueda detallada'. The main content area is titled 'Crear una subasta' and contains the following fields:

- Título:** A text input field.
- Descripción:** A large text area for the auction description.
- Categoría:** A dropdown menu set to 'Arte y antigüedades'.
- Primera puja:** A text input field followed by 'Euros'.
- Fecha cierre:** A date and time picker showing '1 1 2010 00:00'.
- Seleccionar archivo:** A file selection button with placeholder text 'ningún ar...ccionado'.
- Registrar:** A 'Registrar' button.

At the bottom of the form area, it says 'Quien da más' and 'Software diseñado e implementado por Alejandro Martínez Vieites'.

Ilustración 33 Formulario de creación de subasta

V c o d k ² p " g u v c t ^a " f k u r q p k d n g " g n " g p n c e g " ÷ O k
información de relacionada con su perfil y operaciones en el sistema. La página muestra un menú lateral con distintas opciones y los mensajes privados (Ilustración 34 Buzón de mensajería privada) del usuario.



Ilustración 34 Buzón de mensajería privada

Cuando se pulsa sobre el asunto de uno de los mensajes se accede a su contenido (Ilustración 35 Detalle de un mensaje privado), donde se detalla el asunto, el remitente, el destinatario y el texto del mismo.



Ilustración 35 Detalle de un mensaje privado

Para enviar un mensaje a otro usuario, bastará con pulsar sobre el enlace correspondiente del menú lateral y acceder así al formulario de envío (Ilustración 36 Formulario de envío de mensaje).

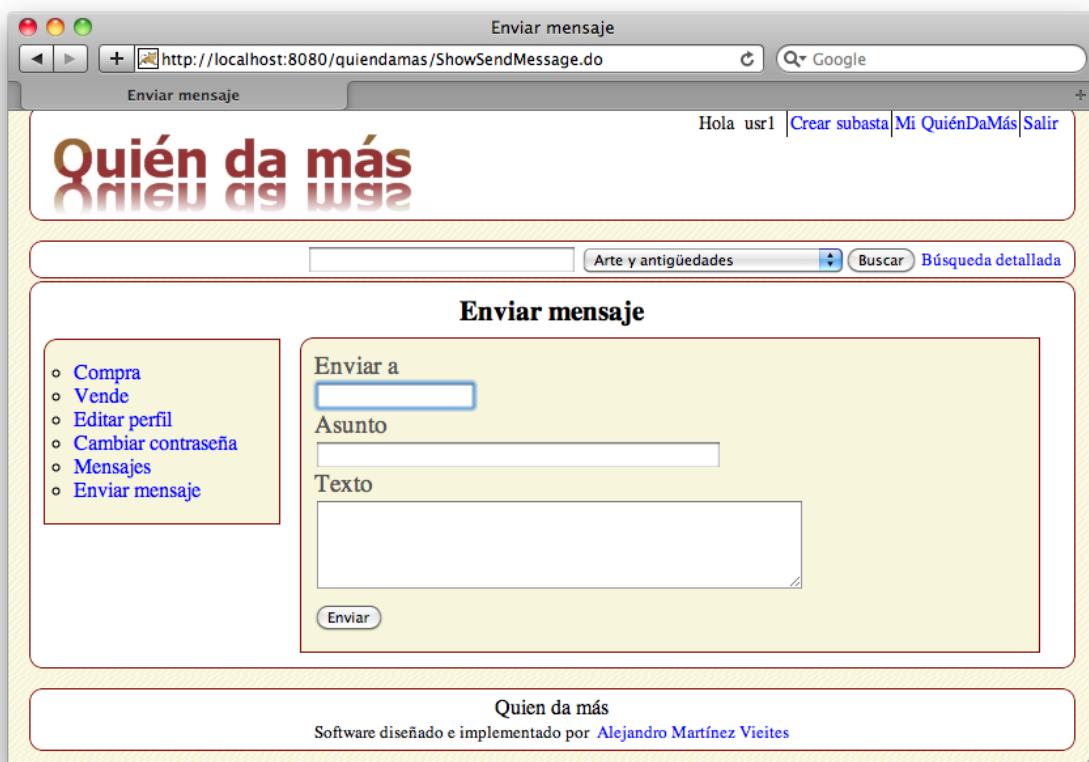


Ilustración 36 Formulario de envío de mensaje

En cuanto a la gestión del perfil de usuario, es posible actualizar los datos del mismo (Ilustración 37 Formulario de edición del perfil de usuario), así como cambiar también la contraseña de acceso al sistema (Ilustración 38 Formulario de cambio de contraseña), mediante dos sencillos formularios.

The screenshot shows a web browser window for the website "Quién da más". The URL in the address bar is <http://localhost:8080/quiendamas/EditUserProfileForUpdating.do?action=...>. The page title is "Formulario de actualización info. registro". The header includes a search bar with "Google", a user greeting "Hola usr1", and links for "Crear subasta", "Mi QuiénDaMás", and "Salir". The main content area has a yellow background and features a sidebar with a red border containing a list of options: "Compra", "Vende", "Editar perfil", "Cambiar contraseña", "Mensajes", and "Enviar mensaje". The main form is titled "Actualización perfil de usuario". It contains fields for "Nombre" (User1), "Apellidos" (Del Sitio), "Dirección correo-e" (usr1@quiendamas.com), "Dirección" (C/ Calle Real 11), "Código postal" (15001), "Nº de teléfono principal" (981000010), "Nº de teléfono secundario" (981000001), "Fecha de nacimiento" (20, 6, 1979), "Idioma" (Español), and "País" (España). A "Actualizar" button is at the bottom.

Ilustración 37 Formulario de edición del perfil de usuario

The screenshot shows a web browser window for the website "Quién da más". The title bar says "Formulario de cambio de contraseña". The address bar shows the URL "http://localhost:8080/quiendamas/ShowChangePassword.do". The page header includes "Hola usr1" and links for "Crear subasta", "Mi QuiénDaMás", and "Salir". The main content area has a yellow background and contains a "Cambio de contraseña" section. On the left, there is a sidebar with a red border containing links: "Compra", "Vende", "Editar perfil", "Cambiar contraseña", "Mensajes", and "Enviar mensaje". The main form section has a red border and contains fields for "Contraseña antigua" (with an input field), "Contraseña nueva" (with an input field), "Confirmar contraseña" (with an input field), and a "Actualizar" button. At the bottom of the page, there is a footer with the text "Quien da más" and "Software diseñado e implementado por Alejandro Martínez Vieites".

Ilustración 38 Formulario de cambio de contraseña

Necesario que el usuario logre iniciar sesión en la aplicación. La sección "Historial de compras" muestra un resumen de las pujas realizadas por el usuario así como de las subastas que ha ganado. Para éstas últimas, existirá la posibilidad de emitir la valoración correspondiente.



Ilustración 39 Información sobre las compras

F g " o c p g t c " e q o r n g o g p v c t k c . " n c " r ª i k p c " o q u
subastas iniciadas por el usuario, tanto las que todavía no han finalizado como las que ya
están cerradas y pueden ser valoradas.



Ilustración 40 Información sobre las ventas

12.3.3. Administración de la aplicación

Cuando el usuario que inicia sesión en el sistema posee privilegios de administrador, es posible acceder al menú de administración (Ilustración 41 Menú de administración) que permitirá gestionar las categorías y revisar los usuarios votados negativamente.



Ilustración 41 Menú de administración

La página de gestión de categorías (Ilustración 42 Gestión de categorías) ofrece una visión general del árbol de categorías del sistema en la zona izquierda y un formulario sencillo a la derecha para añadir más.



Ilustración 42 Gestión de categorías

Siguiendo el enlace de cada una de las categorías es posible acceder a un nuevo formulario de modificación (Ilustración 43 Formulario de modificación de categorías) que ofrece la posibilidad de cambiar el nombre, la descripción y el padre de una categoría específica o incluso borrarla si no existen referencias a la misma.



Ilustración 43 Formulario de modificación de categorías

Siguiendo el enlace de usuarios con baja valoración se accede a una página (Ilustración 44 Usuarios valorados negativamente) con la lista de usuarios que han recibido valoraciones bajas por parte del resto de usuarios.



Ilustración 44 Usuarios valorados negativamente

Para mayor información, pulsando sobre el nombre de usuario es posible acceder al historial de subastas del usuario (Ilustración 45 Historial de operaciones del usuario) para las que ha recibido alguna valoración.

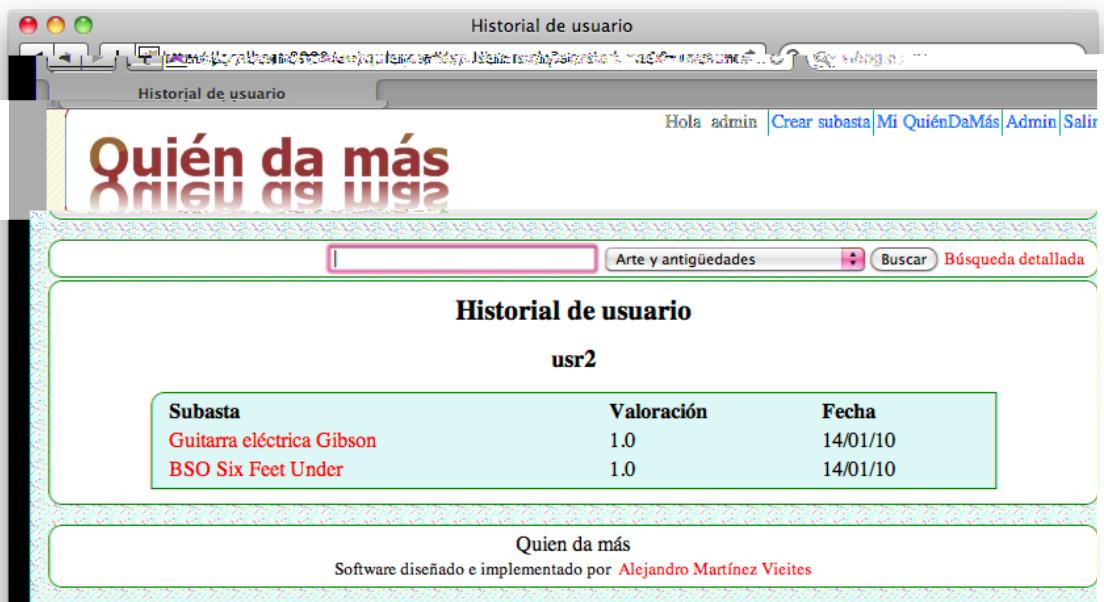


Ilustración 45 Historial de operaciones del usuario

12.4. Contenido del CD

- Proyecto Maven de la aplicación realizada.
- Memoria en formatos .doc, .pdf e imágenes utilizadas.
- Informe detallado de la planificación en formato HTML.
- Proyecto de MagicDraw con todos los diagramas UML realizados.

13. Bibliografía

1. Temario de la asignatura Integración de Sistemas
(<http://www.tic.udc.es/~fbellas/teaching/is-2007-2008/index.html>).
2. Historia de las subastas: <http://en.wikipedia.org/wiki/Auction>
3. Página corporativa eBay: <http://www.ebayinc.com>
4. Información Java EE:
http://en.wikipedia.org/wiki/Java_Platform,_Enterprise_Edition
5. Información ASP.Net: <http://en.wikipedia.org/wiki/ASP.NET>
6. Información LAMP: [http://en.wikipedia.org/wiki/LAMP_\(software_bundle\)](http://en.wikipedia.org/wiki/LAMP_(software_bundle))
7. Patrón TO: http://en.wikipedia.org/wiki/Data_transfer_object
8. Patrón DAO: http://en.wikipedia.org/wiki/Data_access_object
9. Patrón MVC: http://en.wikipedia.org/wiki/Model_View_Controller
10. Patrón adaptador: http://en.wikipedia.org/wiki/Adapter_pattern
11. Patrón factoría: http://en.wikipedia.org/wiki/Factory_method
12. RUP: http://en.wikipedia.org/wiki/IBM_Rational_Unified_Process
13. API de Java: <http://java.sun.com/javase/6/docs/api/>
14. Manual de Maven: <http://maven.apache.org/guides/getting-started/index.html>
15. Manual de Struts: <http://struts.apache.org/1.3.10/index.html>
16. Custom tags:
<http://java.sun.com/developer/technicalArticles/xml/WebAppDev3>

17. Documentación MySQL: <http://dev.mysql.com/doc/refman/5.5/en/>

13.1. Enlaces de Interés

1. IDE NetBeans: <http://netbeans.org>
2. Apache Tomcat: <http://tomcat.apache.org>
3. Magicdraw: <http://www.magicdraw.com/>
4. Omniplan: <http://www.omnigroup.com/applications/omniplan>
5. JUnit: <http://www.junit.org/>
6. Tutorial CSS: <http://www.w3schools.com/css/>

14. Glosario

Analista: persona encargada de investigar, planear, coordinar y recomendar estrategias para acometer el desarrollo por parte del programador.

CASE (Computer Aided Software Engineering): herramientas que, en el marco del desarrollo de software, están destinadas a aumentar la productividad facilitando ciertas tareas y ofreciendo informes sobre la realización de otras.

Contraseña (clave, password): conjunto de caracteres secretos que el usuario proporciona junto con su nombre de usuario (público) para autenticarse en el sistema.

Entorno de desarrollo: entorno de implementación y pruebas de la aplicación.

Entorno de producción: entorno final de funcionamiento de una aplicación donde desempeña el cometido para el que fue desarrollada.

IDE (Integrated Development Environment): aplicación que provee al programador de herramientas "uggtkg" "fg" "tgewtuqu" "gorcswgvcfcu" "*gfkvq" para simplificar la tarea del desarrollo de software.

Java: lenguaje de programación orientado a objetos desarrollado por Sun Microsystems a principios de los 90, con sintaxis basada en la de C.

Lógica de negocio: parte de un sistema que se encarga de las tareas propias de los procesos de negocio que se pretenden resolver en la aplicación.

Programador: persona encargada de escribir, depurar y probar el código fuente de una aplicación informática.

Second Life: mundo virtual en 3D diseñado para que los participantes simulen una realidad alternativa a través de un avatar personalizable capaz de interactuar con el resto.

SQL (Structured Query Language): lenguaje declarativo de acceso y consulta de bases de datos relacionales.

UML (Unified Modeling Language): es el lenguaje gráfico de modelado de sistemas más conocido y empleado en la actualidad. Permite el diseño y la documentación fiel de todas las partes de un sistema, incluidos los aspectos conceptuales.

15. Acrónimos

API: Application Programming Interface.

ASP: Active Server Pages.

CSS: Cascading Style Sheets.

DAO: Data Access Object.

GNU: GNU is Not Unix.

GPL: General Public License.

JDBC: Java Database Connectivity.

JNDI: Java Naming and Directory Interface.

JSP: JavaServer Pages.

JSTL: JavaServer Pages Standard Tag Library.

HTML: Hypertext Markup Language.

LAMP: Linux Apache MySQL PHP.

MVC: Model View Controller.

PHP: PHP Hypertext Preprocessor.

PU (UP): Proceso Unificado (Unified Process).

SGBD: Sistema de Gestión de Bases de Datos.

TO: Transfer Object. También conocido como Data Transfer Object o Value Object.

XML: Extensible Markup Language.

