# Auctions Service

Este documento recoge la descripción de un caso completo que sirve para ilustrar la aplicación de los conceptos de diseño trabajados en la asignatura.

## Descripción del caso

El caso propuesto está inspirado en una versión simplificada de eBay (https://www.ebay.com/). eBay es una plataforma de subastas y comercio electrónico. En este caso, nos centraremos en la funcionalidad relacionada con las subastas.

Toda la funcionalidad gira en torno a los artículos que se subastan. Los datos básicos del artículo son: número identificativo, título, precio inicial (todos los importes se almacenan en euros) y fecha de fin de la subasta. Los artículos se agrupan en categorías, que se identifican mediante un nombre. Cada artículo se asocia a una única categoría.

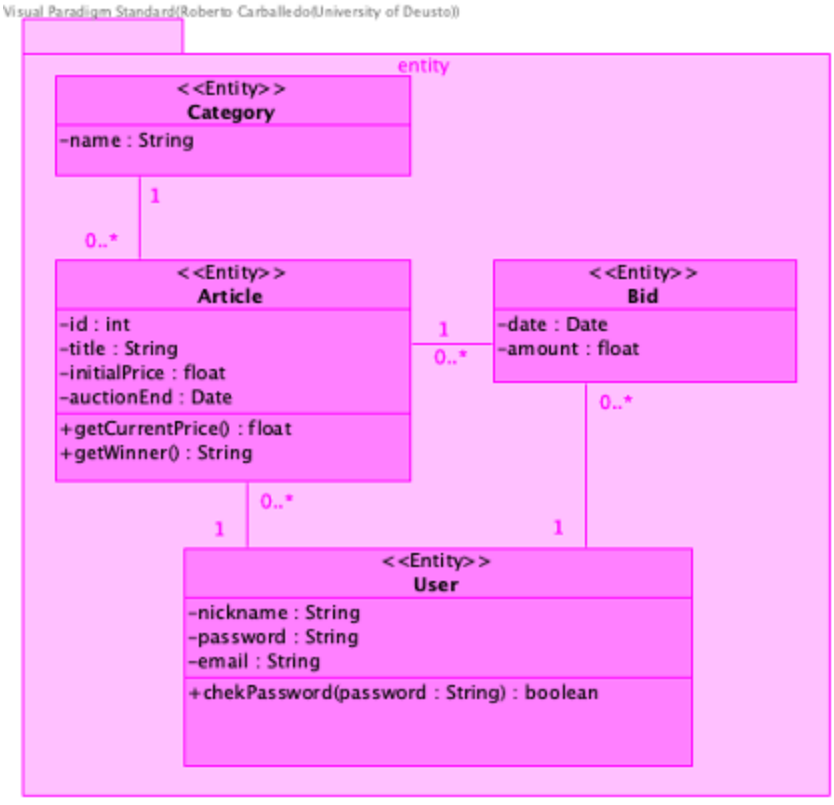


*Ilustración 1: Diagrama de clases de los objetos de dominio*

Cualquier persona puede consultar las categorías, los artículos de una categoría y los detalles de

un artículo, pero es necesario hacer login para pujar por un artículo que está en subasta. El perfil de usuario/a consta de 3 datos: email (utilizado para el login), contraseña y nickname. Además, el perfil también tiene asociados los artículos que una persona ha puesto en subasta así como las pujas realizadas por artículos subastados.

La funcionalidad básica que vamos a diseñar es la puja por artículos subastados. Una puja es una oferta de compra por un artículo subastado. Cada puja se realiza por una persona, en una fecha determinada y lleva asociado un importe, siempre superior al de la puja más alta realizada hasta el momento para dicho artículo.

Cuando finaliza el periodo establecido para una subasta, gana el artículo la persona que haya realizado la puja más alta.

## Consideraciones sobre el de diseño

El sistema de subastas electrónicas propuesto tiene una arquitectura cliente/servidor en la que la comunicación entre cliente y servidor se realiza mediante servicios web REST usando el framework Spring Boot tanto de la parte cliente como de la parte servidora. La persistencia de la aplicación se implementa usando JPA base de datos H2. Por último, el sistema se comunica con un servicio externo (Free Currency Conversion API) que permite realizar conversiones de divisas para mostrar los precios de los artículos en diferentes divisas. La comunicación con el servicio externo también se realiza a través de SB.

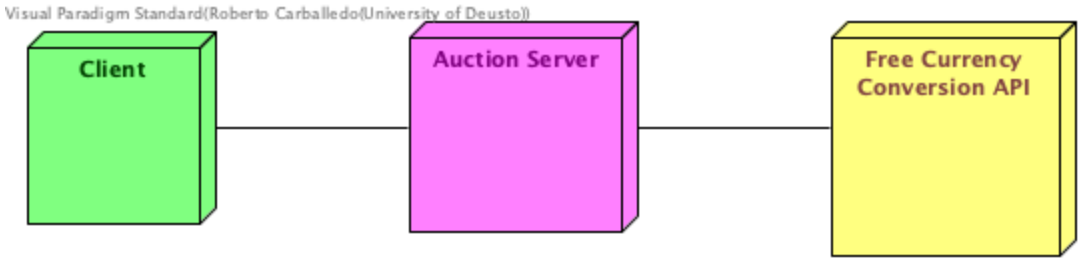


*Ilustración 2: Diagrama de despliegue*

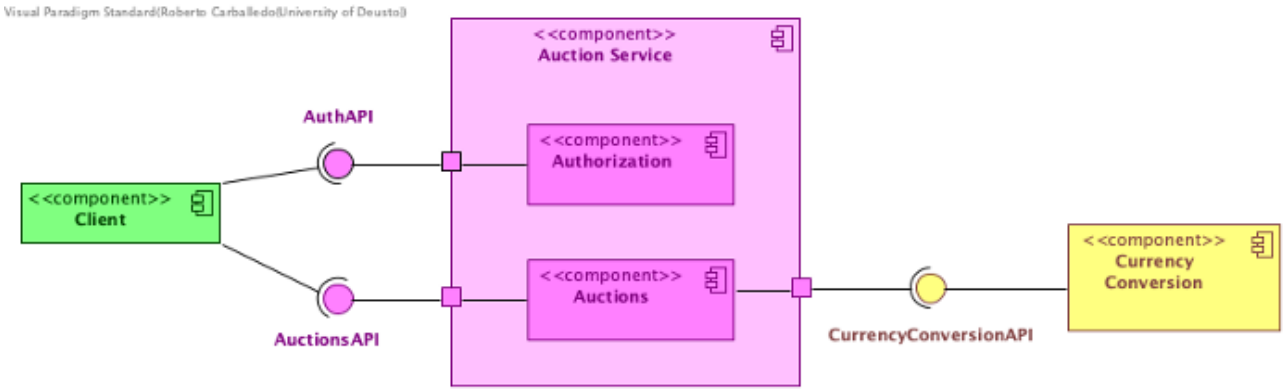


*Ilustración 3: Diagrama de componentes*

El diseño del sistema propuesto integra la mayoría de los patrones de diseño que se revisan a lo largo de la asignatura:

- Patrones del lado servidor:
  - Façade
  - Application Service (AppService)
  - Data Transfer Object (DTO).
- Patrón de acceso a servicios externos:
  - Service Gateway
- Patrón de acceso a datos:
  - Data Access Object (DAO).
- Patrones del lado cliente
  - Client Controller
  - Service Proxy
- Otros patrones:
  - State Management

La funcionalidad que ofrece nuestra versión simplificada de eBay es la siguiente:

- Login y Logout
- Consultar de todas las categorías de artículos
- Consultar de los artículos de una categoría
- Consultar los detalles de un artículo
- Pujar por un artículo subastado

El siguiente diagrama de secuencia muestra la funcionalidad descrita a nivel de componentes.

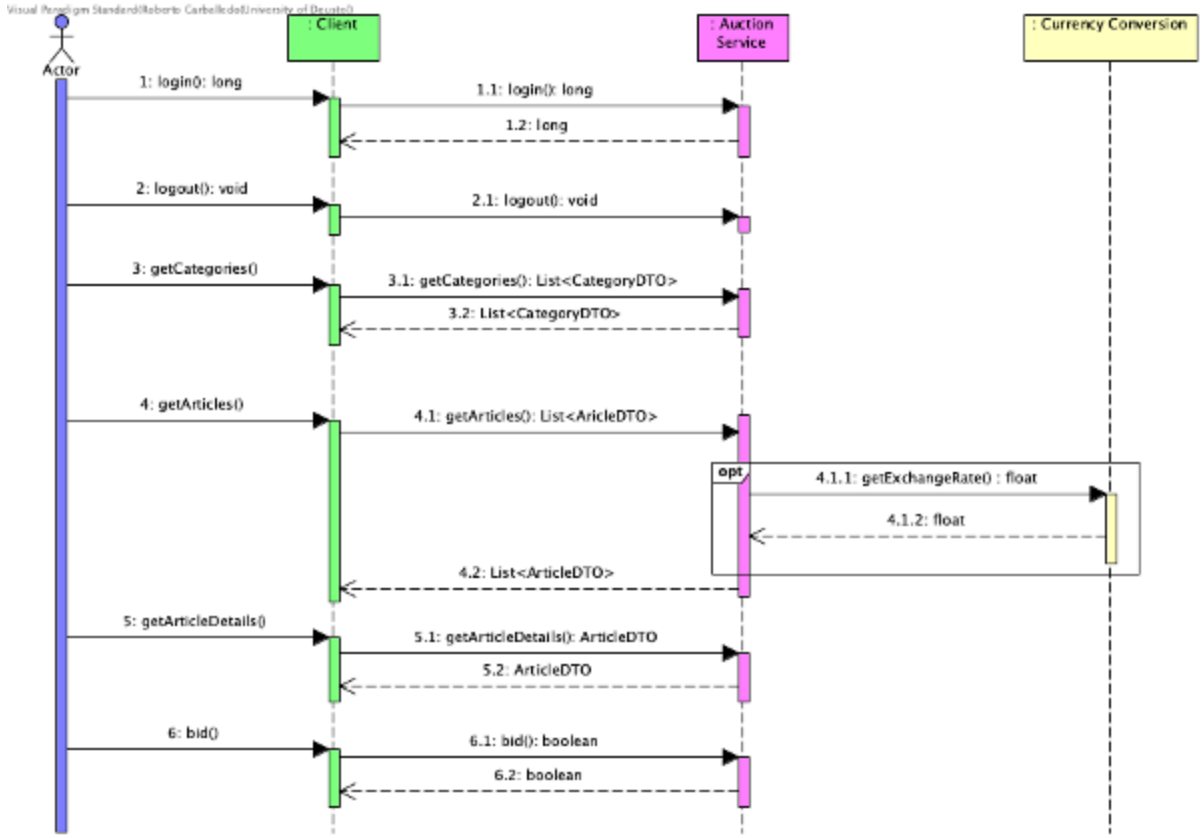


*Ilustración 4: Diagrama de secuencia a nivel de componentes*

## API del Servidor Auctions

El API del Servidor Auctions se ilustra en la siguiente imagen (ver descripción detallada):

- URL de acceso a la documentación Swagger: http://localhost:**8080**/swagger-ui/index.html
- URL de acceso a la documentación Open API: http://localhost:**8080**/v3/api-docs

*(Nota: ajusta el puerto de acuerdo la configuración de inicio de tu servidor en "application.properties")*

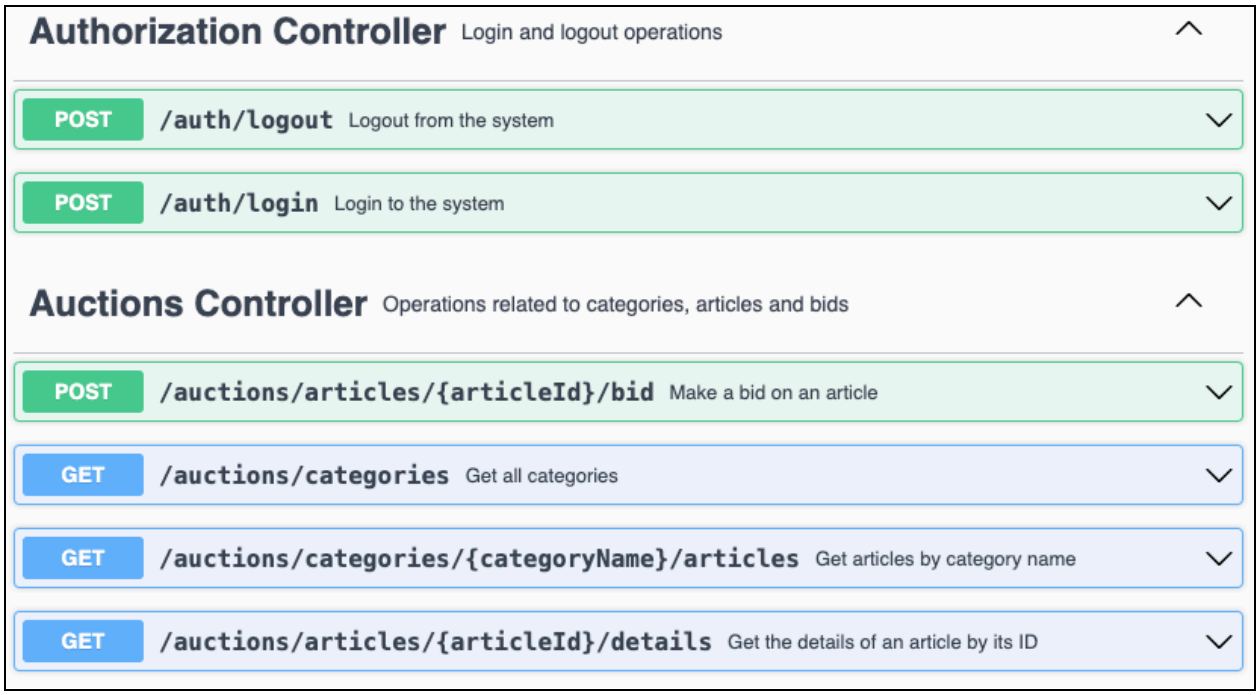


*Ilustración 5: API del Servidor Auctions*

## Consideraciones sobre la implementación

El caso se ha implementado en forma de 2 proyectos:

- **Auctions Server**: contiene el código del servidor. Al igual que en el caso del proyecto en equipo, se han generado 2 versiones del proyecto servidor que van aumentando en complejidad a medida que se incorporan nuevos patrones:
    - **Versión 1** (**https://github.com/rcarball/auctions-server-1**): incluye la estructura base del servidor Auctions. Esta versión implementa los patrones: Façade, Application

Service, Data Transfer Object (DTO) y State Management.

- **Versión 2** (**https://github.com/rcarball/auctions-server-2**): añade a la versión previa la interacción con el servicio externo de conversión de divisas y la persistencia en una base de datos H2. Esta versión implementa los patrones: Data Access Object (DAO) y Service Gateway.

- **Auctions Client** (pendiente)**:** contiene el código del cliente y está implementado como una aplicación web Spring Boot implementada con Thymeleaf. El cliente incorpora los patrones Client Controller y Service Proxy.

## Descripción del escenario "Login"

El proceso de login consiste básicamente en la validación de email y contraseña. Como no queremos manipular las contraseñas en texto plano vamos a utilizar un algoritmo de cifrado tanto en el cliente (antes de enviar la contraseña al servidor) como en el servidor (antes de almacenar la contraseña en la base de datos). En concreto se hace uso del algoritmo SHA-1, y su implementación de Apache Commons. Además, como una medida de seguridad adicional, la clase User NO ofrece método *getPassword()*.

Cuando se realiza la validación de la contraseña, si el resultado es correcto, el servidor genera un token (a partir del timestamps del instante en que se ha realizado el login) que se devuelve a la interfaz gráfica del cliente. Dicho token se envía como parámetro adicional al realizar una puja y sirve para validar que se ha realizado el login previamente. Además, el token se utiliza en el servidor para indexar la información de usuarios/as que han hecho login a modo de caché. De esta forma, se minimizan los accesos a la base de datos y las interacciones entre cliente y servidor. Esta forma de gestionar el estado del servidor representa un caso de Servidor "con estado" (Stateful Server) ya que se recuerda información en el lado servidor para ser utilizada durante la interacción. *No se debe confundir la necesidad de gestionar el "estado de la sesión" con la gestión del "estado de la aplicación". Aquí, nos centramos en mantener el estado de la sesión, lo que significa que el sistema es stateful en ese aspecto. Sin embargo, la gestión del estado de la aplicación puede ser stateful o stateless (Gestión de Estado). En este caso concreto, siendo el caso muy limitado, la gestión del estado tiende a ser stateful porque el objeto User se mantiene en memoria y se utiliza en el escenario de puja.*

## Descripción del escenario "Puja por un artículo"

A continuación, se indican los pasos a seguir para realizar una puja.

1. El proceso se inicia mostrando en la interfaz gráfica la lista completa de categorías.

2. Se selecciona una categoría y se muestra la lista de artículos asociados a dicha categoría que se están subastando. Para cada artículo en subasta se muestra la fecha de fin, el precio inicial, el precio actual (puja más alta) y el número de pujas. A pesar de que todos los importes se almacenan en euros, se puede solicitar una conversión de moneda en cualquier momento.

3. Para pujar es necesario estar logueado/a, por lo tanto, si no se había realizado el login, habrá que hacerlo antes de pujar. Si ya se había hecho login, se indica la cantidad de la puja y la información de la misma se registrará en el servidor central.