Resumen

En este trabajo se va a definir una página web de subastas en la que tanto un usuario como un agente inteligente podrá pujar en tiempo real por un artículo. Esta la conformarán tres tipos de subastas, las ascendentes (al alza), las holandesas (a la baja), y las de sobre cerrado (puja única secreta). Por otro lado, las entidades inteligentes capaces de realizar ofertas poseerán distintos comportamientos inteligentes adaptados a cada tipo de puja. Así, estas estrategias se basarán en diferentes parámetros que les atribuye el propietario, como el grado de agresividad.

Para su implementación se ha decidido emplear el framework web Django para la creación de la página web, y la plataforma SPADE para el sistema multiagente. Finalmente, para la comunicación entre ambas partes se usarán peticiones HTTP dirigidas a la extensión Django ReST Framework.

Palabras clave: Sistema multiagente, página web, agentes inteligentes, subastas, inteligencia artificial.

Abstract

During this project an auction website will be done, where not only internet users are able to bid on real time, but intelligent agents also can. Three types of sales are being part of this, clock auctions (upward), dutch ones (down) and sealed bid ones (secret single bid). On other side, different behaviours can be adopted by intelligent entities, depending on the type of bid. Moreover, these strategies are based on different parameters chosen by the owner, such as the aggressivity.

Regarding the development, the web framework Django is chosen for the web creation and the SPADE platform is selected for the multiagent system. Finally, Django ReST Framework extension is being used for the communication between both sides through HTTP requests.

Keywords: Multiagent system, webpage, intelligent agents, auctions, artificial intelligence.

Índice

Índice de figuras

Capítulo I
Introducción

1.1 Introducción

Las subastas han sido un método de intercambio mucho antes de la aparición de los sistemas de comunicación online. Sin embargo, con la llegada de Internet se produce una revolución en el mundo de las subastas. De esta forma, la infraestructura de comunicación permite llegar a millones de nuevos usuarios potenciales. Como resultado, los usuarios pueden pujar por productos desde donde quieran sin tener que asistir personalmente al evento de subasta. Además, el proceso de búsqueda se simplifica enormemente. Al igual ocurre con los vendedores, que se benefician de la visibilidad global de Internet [1].

Ante esta nueva situación, las subastas online se comienzan a popularizar dando lugar a numerosas páginas web conocidas como Ebay[1]. En ellas se empiezan a adaptar los diferentes tipos de subastas existentes, y con ello se desarrollan varias estrategias para enfrentarse a cada una.

En este punto es importante explicar que es y para qué sirven los sistemas multiagente. Se tratan de un grupo de entidades inteligentes proactivas que interactúan entre ellos para resolver un problema concreto. De esta forma, pueden trabajar juntos para alcanzar el mismo objetivo o enfrentarse ya que cada uno busca su propio interés. Además, está siendo empleado en una gran cantidad de áreas como aplicaciones comerciales, control de procesos o comercio electrónico [2].

Como resultado, la propuesta de este proyecto es añadir el uso del sistema multiagente al dominio de las subastas online. Por consiguiente, el objetivo de este trabajo es crear un conjunto de agentes que sean capaces de realizar pujas en una web de subastas, creada para la ocasión en este trabajo, y donde los agentes asuman una serie de comportamientos personalizados que sigan una estrategia predeterminada. Así, cada estrategia estará enfocada a un tipo de subasta y a las preferencias del comprador al cual represente el agente.

Durante este documento se explica todo el proceso que se ha seguido para la creación del trabajo. Comenzando por las fases de estudio y concepción hasta las pruebas de validación y análisis de resultados.

1.2 Motivación

Como se ha comentado en la introducción, actualmente las subastas online son un medio muy común para el intercambio bienes y servicios de una gran diversidad, tanto del mercado mayorista como del mercado de segunda mano.

[1] http://www.ebay.es

Ante esta situación, me planteé la opción de hacer más tecnológico y simple el proceso de puja, evitando tener que estar pendiente de la subasta. Como resultado, decidí crear agentes inteligentes capaces de pujar siguiendo diferentes comportamientos y estrategias en base a lo que el usuario le configure.

Asimismo, para facilitar el proceso he diseñado una página web en la que poder crear subastas de diferentes tipos. En dicha web, podrán participar tanto personas como agentes de software preprogramados para pujar en las subastas, puesto que ofrecerá interfaces diferentes para humanos y agentes inteligentes.

Además, se trata de una gran oportunidad para poner en práctica parte de los conocimientos adquiridos durante el periodo de estudio del Grado en Tecnologías Interactivas. De esta forma, no solo se englobará lo aprendido en las asignaturas de programación, sino que también lo de otras como Inteligencia Artificial, el proyecto de desarrollo web o Diseño de interfaces y experiencia de usuario.

1.3 Objetivos del proyecto

El principal objetivo de este trabajo es el desarrollo tanto de una página web de subastas como de agentes inteligentes con diferentes comportamientos capaces de pujar automáticamente. Así, los usuarios serán capaces de crear sus propias subastas, realizar una puja, al igual que crear un agente que sea capaz de pujar por él.

Como resultado, para poder cumplir con el objetivo principal se deberán cumplir los siguientes objetivos secundarios:

- Analizar el estado del arte actual de los agentes inteligentes y el desarrollo web, valorando las diferentes opciones de implementación.
- Obtener información sobre el mundo de las subastas online, revisando los tipos de subasta.
- Estudiar las fortalezas y debilidades tanto de páginas web similares, como de agentes inteligentes.
- Asegurar la seguridad en la página web, de forma que para poder realizar una acción tengas que estar dado de alta y el sistema de pujas esté protegido.
- Realizar un estudio de las diferentes estrategias existentes sobre subastas para, posteriormente, ser incluidos en los comportamientos de los agentes y decidir cuál es mejor la opción en cada ocasión.
- Probar el correcto funcionamiento del trabajo mediante la creación y uso de pruebas de validación.

1.4 Metodología

En este proyecto se llevará a cabo la metodología Scrum para su desarrollo. De esta forma, contaremos en todo momento con un producto mínimo viable que nos permitirá satisfacer a los clientes iniciales.

Además, con el fin de controlar los cambios realizados en el código y poder retroceder a momentos anteriores se ha decidido utilizar una herramienta de control de versiones. Esto nos permite realizar una buena gestión de las versiones del proyecto.

En el proceso de creación del trabajo se seguirá un orden específico. Así, primero se hace un estudio del estado del arte en el cual se analiza la situación tecnológica. A continuación, se diseña todo lo que posteriormente será implementado. Tras ello, se implementan los diseños en el producto final. Finalmente, se comprueba mediante el uso de pruebas de validación que todo se realiza tal y como había sido planeado.

Todo este punto se explicará más en profundidad en el apartado metodología dentro del desarrollo.

1.5 Estructura del documento

Capítulo I: Introducción

En el primer capítulo se encuentra todo lo referente a la introducción. De esta forma, aparece una breve descripción a modo de resumen del contenido del proyecto, la motivación la cual me ha llevado a su desarrollo y los objetivos que pretendo conseguir durante su creación.

Capítulo II: Estado del arte

Durante este segundo capítulo se va a realizar un análisis teórico de la situación tecnológica. Por consiguiente, se valoran las diferentes opciones para el desarrollo del proyecto tanto por la parte de creación web como por la de inteligencia artificial. Asimismo, también se analizan varios tipos de subastas y las mejores estrategias de apuesta que se pueden llevar a cabo en cada una.

Capítulo III: Desarrollo del proyecto

A continuación, en el tercer capítulo se explica el proceso de creación del trabajo. Esta parte se encuentra dividida en el desarrollo de la página web y en la creación de agentes inteligentes capaces de pujar automáticamente. Así, se muestran tanto la metodología como el análisis de requisitos, los diseños y la implementación.

Capítulo IV: Resultados

Después del desarrollo, en el cuarto capítulo vamos a analizar los resultados obtenidos en el sistema multiagente. En este punto, se descubrirán las mejores estrategias y comportamientos para cada tipo de subasta.

Capítulo V: Conclusiones

Posteriormente, en el quinto capítulo se procede a realizar una reflexión final del trabajo. Como resultado, se muestran los objetivos cumplidos, los resultados obtenidos, y las propuestas de mejora a nuestro proyecto.

Capítulo VI: Anexo

Finalmente, en el sexto capítulo aparecen el manual de usuario y las referencias a las citas y documentos empleados durante la creación del documento.

Capítulo II
Estado del arte

En este capítulo se hace un estudio de la situación tanto en las subastas online como en los sistemas inteligentes y páginas web. Como resultado, se valorarán los distintos tipos de herramientas para su desarrollo y se decidirá cual se implementará.

2.1 Subastas

Durante milenios se han utilizados las subastas en mercados para la determinación de precios sin intermediarios [3]. Así, en esta transacción existen dos roles, los postores o compradores y el vendedor o subastador. En ella, varios compradores compiten por conseguir el producto del vendedor por medio de pujas. Durante su transcurso, el subastador buscará obtener el mayor beneficio, y el postor minimizar el coste de obtención de la compra [4].

2.1.1 Subastas online

Con la llegada de Internet se abrió una ventana de posibilidades en el mundo de las subastas. Para los vendedores implica una revolución ya que sin importar el lugar en el que se encuentren los posibles compradores podrán participar en la puja, solo necesitarán contar con acceso a la red. Por ello, no solo contarán con aquellas personas interesadas en ir al evento, sino que también tendrán acceso a la venta los millones de usuarios diarios de Internet. Al igual ocurre con los compradores que no tendrán que desplazarse de sus hogares para adquirir los productos. Además, cuentan con una gran cantidad de ventas que les permite elegir aquella que más se adecue a sus necesidades.

Asimismo, no solo se trata de un entorno mucho más atractivo e interactivo para comprador y vendedor, sino que también les permite reducir los costos.

De esta forma, las casas de subastas tradicionales obtienen sus ingresos de la venta directa de los artículos, mientras que en las subastas online se cobran tarifas sobre el precio de venta a los vendedores. Por un lado, en las grandes páginas web como Ebay tienen una tasa del 11,5% de la parte inferior a 2000€, y un 2% sobre la parte que supere ese precio. Además, cuenta con un cobro extra por pedido de 5 céntimos si su precio es menor a 10€, y de 35 céntimos si es mayor [5]. Por otro lado, las casas de subastas obtienen entre un 12-25% de la compra en comisiones al comprador, y en determinados casos otro 10% al vendedor [6]. En definitiva, se puede confirmar que el costo es considerablemente inferior en subastas no presenciales.

Sin embargo, no todo son ventajas, también existen algunos inconvenientes como el fraude. Tradicionalmente el comprador realiza un cheque, transacción instantánea o pago en efectivo, y recibe el artículo presencialmente. No obstante, a través de Internet todo es diferente ya que se hace todo en línea.

Cabe destacar que los casos de fraude son escasos debido a diversos factores. En primer lugar, las propias páginas web alientan y ayudan a los afectados a denunciar los casos. En segundo lugar, existe un mecanismo de comentarios y calificación público tanto sobre pujadores como subastadores. Esto permite a los usuarios analizar a quien están comprando o vendiendo antes de realizar una acción. Finalmente, con el paso del tiempo se han creado herramientas de terceros que evitan este tipo de problemas. Así, estas aplicaciones retienen el pago hasta que el comprador recibe el bien en correctas condiciones [7].

2.1.2 Funcionamiento

Ya se han expuesto muchos datos que avalan el uso y alzamiento de las subastas online, pero ¿cómo funcionan realmente? Generalmente existe un precio mínimo, un tiempo de subasta y en algunos casos un “precio de reserva”.

Por lo tanto, el subastador cuando crea la subasta elige una oferta mínima por debajo de la cual no aceptará pujas. Asimismo, el tiempo de subasta consiste en la duración hasta el fin de la misma. La mayoría de los sitios tienen una duración predefinida entre tres y diez días, sin embargo, también existen subastas con un tiempo menor a tres minutos o una hora.

Además, cuando se decide establecer un “precio de reserva”, este se determina de antemano y no se revela a los compradores hasta el fin de la subasta. En el caso de que no sea superado por una oferta, el producto no será vendido. Esto permite al subastador establecer una estrategia en la que se fija un precio inicial bajo para generar interés, impulsando las ofertas y adquiriendo un mayor beneficio. De esta forma, también se le da la oportunidad de observar el entorno a los postores interesados y obtener información con respecto a sus competidores [7].

2.1.3 Subastas ascendentes

Se trata del tipo de subasta tradicional, y por ello la más común. En ellas, los postores realizan una puja que representa una oferta de compra con un precio. Esta tiene que ser superior a la anterior o, en caso de no haber ninguna, al precio mínimo. Finalmente, aquel que haya hecho la última oferta cuando se acabe el tiempo será el ganador de la subasta [3].

Una estrategia a seguir puede tener en cuenta el historial de pujas realizadas en el producto, analizando el interés que existe en él, y el comportamiento del resto de postores, estudiando su agresividad. Finalmente, es muy importante establecer un valor máximo de oferta que no sobrepasar [4].

Económicamente es muy eficiente ya que el postor con la tasación del producto más alta será el que se llevará el producto, pues, estará dispuesto a gastar más dinero y superar al resto. Sin embargo, también posee desventajas como los costos de comunicación en la que en bajos precios o a finales del tiempo de subasta, múltiples compradores pueden pujar simultáneamente [8].

2.1.4 Subastas holandesas

Al contrario que en las subastas ascendentes, en las holandesas el precio va disminuyendo conforme pasa el tiempo. De esta forma, el primer postor en realizar una puja se convierte en el ganador de la subasta y obtiene el producto.

A la hora de abordar este tipo de pujas no existe una estrategia ganadora. Generalmente se hace una relación entre el valor del producto y el resultado de subastas con artículos similares. De esta forma, el planteamiento que se buscará será vencer pagando un precio inferior al valor estimado del artículo [4].

Su principal ventaja es que preserva la privacidad ya que no se revelan informaciones excepto el ganador y su oferta [8]. En este punto, en España se trata del método de venta típico en las lonjas de pescado.

2.1.5 Subastas de sobre cerrado

En este tipo de subastas el comprador solo envía una oferta al subastador. Lo interesante es que esta puja es desconocida para todos los participantes hasta el momento en el que el tiempo de subasta se acaba. En ese momento se descubren todas las ofertas y se vende el artículo al postor con la puja más alta.

Al igual que en el tipo de subastas anterior no existe una estrategia ganadora ya que careces de estímulos con los que interaccionar. Así, el método a seguir será realizar una oferta inferior al precio previsto del producto de manera que se adquiera el mismo.

2.2 Agentes inteligentes

“Los agentes son programas software que actúan flexiblemente en representación de sus propietarios para alcanzar objetivos particulares.” Jennings N. (2001).

Como bien aparece en la cita, un agente inteligente es una entidad autónoma, proactiva y social que desarrolla un comportamiento inteligente para cumplir sus objetivos de forma reactiva y deliberativa. Típicamente siguen un ciclo de Percepción-Deliberación-Acción. Además, actúa en lugar de otros usuarios para obtener unas metas preprogramadas, de forma que puede contar con la ayuda de su propietario o simplemente intervenir por él dependiendo de sus propósitos. En el caso de un agente inteligente que actúa en representación de un ser humano se le llama “agente interfaz”.

2.2.1 Entorno

Con el fin de llevar a cabo su repertorio de acciones, debe analizar el entorno en el que se encuentra y realizar una toma de decisiones en base a ellas. Estos poseen dos características principales que son el no determinismo y el dinamismo [9].

Normalmente, la entidad carece de un control total, sino más bien parcial, sobre el entorno tratándose de entornos no deterministas. En ellos, una acción del agente puede influir y cambiar la situación. De esta forma, el mismo acto realizado ante las mismas circunstancias puede obtener resultados completamente distintos, obteniendo un resultado incierto.

Dentro del dinamismo se distinguen dos escenarios, los entornos dinámicos y los estáticos. Por un lado, los dinámicos cambian debido a procesos externos al agente como puede ser Internet que está en continua evolución. Por otro lado, los estáticos son modificados solamente por la acción de estas entidades.

2.2.2 Lenguaje de comunicación

Para conseguir la comunicación entre los agentes se debe compartir un lenguaje en el que poder coordinarse, cooperar y negociar.

Durante la creación del mismo se deben establecer unas características claras como son la forma, el contenido, la semántica y la fiabilidad. Primero, se busca una forma sencilla y lineal con un contenido simple ya que tiene que ser legible por las personas y el resto de entidades. Además, la semántica debe tener una base teórica con unas reglas y similitud entre el significado y la representación. Finalmente, tiene que permitir una comunicación fiable y segura [9].

2.2.3 Resumen

Como se ha podido ver, todos los agentes inteligentes poseen como propiedades comunes la autonomía, la proactividad, la reactividad y la habilidad social.

De esta forma son entidades que están controlados por una o varias personas, pero que funcionan sin su implicación directa, es decir, son autónomos. También, son proactivos de forma que cumplen los objetivos por iniciativa propia. Además, son reactivos ya que analizan su entorno y responden para obtener las metas establecidas. Asimismo, poseen de un lenguaje común que emplean para comunicarse entre sí, esto es lo conocido como habilidad social.

2.2.4 Sistema multiagente

Un sistema multiagente es aquel en el que varios agentes están en un mismo entorno interaccionando entre sí para cumplir un objetivo común o individual. Así, se explica en la siguiente cita:

“Cuando dos o más agentes son capaces de trabajar de forma conjunta con el objetivo de resolver un problema se habla de un sistema multiagente.” Mas A. (2005) en Agentes software y sistemas multiagente.

Dentro de este sistema se pueden encontrar dos tipos de comportamientos para la resolución del problema. Uno cooperativo en el que los agentes colaboran debido a que tienen los mismos intereses. Y otro competitivo en el que cada uno defiende sus propias metas, enfrentándose así a los del resto. En estos últimos se encontrarán nuestras entidades inteligentes ya que combatirán por ganar las subastas.

2.3 Herramientas para construir sistemas multiagente

Una vez ya se conoce el significado de un agente inteligente se van a analizar diferentes herramientas para la creación de un sistema multiagente. Finalmente, se llegará a una conclusión del mecanismo que se empleará y el motivo.

2.3.1 JACK

Para comenzar se realiza un estudio del método de creación de agentes JACK, que está basado en el lenguaje de programación Java.

Sin embargo, para la programación de los agentes se debe conocer el lenguaje propio de JACK. Este se trata de una especie de extensión de Java que ofrece distintas funciones. Entre ellas se encuentran la definición de nuevas clases, interfaces y métodos para crear agentes y añadirles comportamientos, o los comandos @send y @post que facilitan la comunicación entre ellos. Asimismo, también facilita la gestión de acciones, subtareas y mantenimiento de condiciones [10].

JACK no solo usa un modelo de agente clásico, sino que también emplea un modelo BDI. Permitiendo separar la actividad de seleccionar un plan de la propia acción de ejecutar los activos, de forma que la entidad toma menos tiempo en ejecutar los planes.

Por otro lado, proporciona una serie de capacidades que se trata de comportamientos comunes a varios agentes, es decir, planes y eventos que comparten. Por lo tanto, es viable reutilizar estas capacidades entre distintas entidades.

Además, la plataforma posee una tecnología de serialización de objetos en texto ASCII y de una herramienta para la creación de diagramas de flujo con el razonamiento del agente. De esta forma, es más fácil de entender y analizar el comportamiento del mismo.

Finalmente, cuenta con un entorno de desarrollo gráfico en el que trabajar. Como resultado, permite escribir planes y conectarlos a agentes, administrar la comunicación, compilar, ejecutar, así como depurar y rastrear [11].

2.3.2 JADE

A continuación, se expone el entorno JADE que se encuentra desarrollado completamente en Java. Esta cuenta con una interfaz gráfica que se emplea para el desarrollo, la depuración y la configuración [10].

Además, proporciona una arquitectura simple y potente que se basa en el uso de mensajes asíncronos. También destaca su mecanismo de transporte adaptable en el que se puede elegir entre varios protocolos disponibles como SMTP o HTTP [12].

2.3.3 MADKit

Al igual que el resto de las herramientas vistas, MADKit está implementada en el lenguaje de programación JAVA. Además, los agentes se pueden crear tanto en JAVA como en Scheme, Jess, BeanShell u otro lenguaje que se quiera añadir.

Este se destaca por la utilización de un modelo de organización consistente basado en el Agente, Grupo y Rol. Asimismo, presenta una plataforma modular y escalable con una interfaz gráfica simple y una comunicación basada en el mecanismo P2P [13].

2.3.4 ZEUS

El proyecto ZEUS busca crear sistemas multiagente colaborativos de forma rápida y sencilla para personas inexpertas en este mundo. De esta forma, se persigue que sea una aplicación general y personalizable, basada en la programación visual en lenguaje JAVA [10]. Con el fin de obtener este resultado se ha de separar entre la propia herramienta y la metodología que presenta un conjunto de métodos y principios para el desarrollo de agentes.

Así, se define el desarrollo se basa en cuatro etapas, formando parte las dos primeras de la metodología y las dos siguientes de la herramienta. La primera es el análisis del dominio, donde se espera obtener el modelo de roles de los agentes. En la segunda se diseña la entidad, estableciendo sus necesidades a la hora de realizar su función. La tercera es la implementación del agente en base a lo diseñado anteriormente. Finalmente, la cuarta es el soporte en tiempo de ejecución.

Además, como su objetivo es emplear una programación visual, utiliza instrumentos de monitorización de este tipo. Por lo tanto, posee un Visor de Sociedad para ver los agentes y sus relaciones, una Herramienta de control para modificar su estado remotamente, y un Generador de Informes con las estadísticas del rendimiento [14].

2.3.5 SPADE

Al contrario que el resto, SPADE está basada en el lenguaje de programación Python. Esto le permite poder ejecutarse de manera distribuida en distintas máquinas con distinto sistema operativo. Además, usa un modelo de comunicación fundamentada en XMPP por medio de un servidor de mensajería instantánea.

Con respecto al resto de herramientas posee algunas ventajas. En primer lugar, es una plataforma de código libre y uso gratuito. Asimismo, garantiza confidencialidad y seguridad ya que dispone tanto de autenticación de usuario y contraseña como de una conexión cifrada. Finalmente, también dispone de una interfaz gráfica que muestra toda la información relativa a los agentes, desde su nombre hasta los mensajes recibidos y comportamientos. Así, desde esta se podrá gestionar cualquier actividad del agente, así como sus contactos, su mensajería o presentar una interfaz de aplicación personalizada.

De esta forma, cada agente puede desarrollar un comportamiento previamente establecido que lo diferencie del resto de agentes. Estos son útiles para realizar tareas periódicas, que se ejecuten en bucle, o en una máquina de estados finita entre otros [15].

2.3.6 Elección de la herramienta

Una vez se han visto las características de varias herramientas para la creación de agentes inteligentes, se va a decidir cuál es la que desarrollaremos en este proyecto.

Debido a que se va a contar con uno de los desarrolladores de SPADE como tutor, se ha tomado la decisión de utilizarla. No solo se podrá contar con el tutor para la resolución de dudas, sino que también cumple todos los requisitos para la elaboración del trabajo.

Asimismo, se trata de un mecanismo muy sencillo y rápido para la creación de agentes de uso gratuito y con una buena documentación. Igualmente, posee comportamientos que son adecuados para la implementación de estrategias en las subastas. Todo ello, en una plataforma segura con interfaz gráfica.

2.4 Herramientas para la creación web

En este apartado se debe de estudiar distintas alternativas para la creación de la página web. Para ello, se ha de tener en cuenta que para el sistema multiagente se ha determinado usar SPADE, por ello la herramienta debe de implementar un método para comunicarse por medio de peticiones HTTP.

2.4.1 Node.js

Para comenzar Node.js se trata de un entorno de código abierto basado en el lenguaje de programación JavaScript del lado del servidor. De esta forma, sus aplicaciones constan con distintos módulos que son administradas por NPM, una herramienta de gestión de paquetes.

Su principal característica es que se basa en eventos asíncronos, de manera que el servidor permanece dormido hasta que reciba una petición. Para ello, el receptor debe estar preparado para aceptar datos en cualquier momento ya que el transmisor puede enviarlo sin previo aviso. Todo ello, le hace muy escalable y rápida [16].

Asimismo, permite el acceso directo al sistema de archivos, así como la base de datos o los recursos del sistema operativo. Sin embargo, de forma estándar no provee de ningún mecanismo para vulnerabilidades de inyección de código.

Tampoco posee una herramienta eficiente para analizar el comportamiento y rendimiento del programa por defecto. Al igual que instrumentación de código en tiempo de ejecución, que es clave para la creación de perfiles o detección de malas prácticas de codificación. Todo ello se debe a que se trata de un micro-framework que ofrece solamente la funcionalidad mínima [17].

2.4.2 Laravel

En este caso, Laravel se trata de uno los frameworks full-stack basado en lenguaje PHP más populares. Así, abarca todas las tareas esenciales desde la administración de base de datos hasta la generación HTML.

A nivel estructural, es una mezcla de componentes de Composer. Este es una herramienta de gestión de dependencias PHP que cuenta con miles de paquetes. Como resultado, se encarga de administrar las dependencias en nuestro proyecto. Todo ello, le otorga libertad al desarrollador en cuanto a la elección de paquetes.

Además, emplea en planteamiento Modelo-Vista-Controlador para separar la lógica de la aplicación. De esta forma, el modelo se comunica con la base de datos y permite al usuario manipular la misma. La vista es la representación visual de la página web, mostrando los datos del modelo. Para su construcción, Laravel implementa plantillas blade que simplifican el desarrollo. Finalmente, el controlador se trata del enlace entre estos dos.

En cuanto a la base de datos, cuenta con migraciones, Seeder y el ORM Eloquent. Para modificarla se recurre a las migraciones, que son una especie de control de versiones. Una vez estructurada, se puede rellenar con datos mediante los Seeders. Con el fin de realizar consultas a la base de datos, emplea el ORM Eloquent que utiliza métodos en vez de código SQL. Para ello, se ha de interactuar con un modelo que representa a una tabla de la base de datos [18].

Otras características que incluye son el registro de usuario e inicio de sesión con autenticación y autorización al igual que middlewares para filtrar solicitudes HTTP, y protección contra ataques. Entre estos se encuentra el CSRF, el XSS, la inyección de SQL o la vulnerabilidad de asignación masiva [19].

2.4.3 Django

Al igual que el anterior, Django es un framework full-stack de código abierto que fomenta un desarrollo rápido y un diseño limpio y pragmático. Sin embargo, este se basa en el lenguaje de programación Python. Este fue creado internamente en el Washington Post y se usa actualmente en páginas webs como Pinterest e Instagram.

Además, emplea la arquitectura "*shared-nothing*" que le dota de una gran escalabilidad. De esta forma cada parte es independiente de las demás, pudiendo añadir y reutilizar componentes de otros proyectos simplemente. Asimismo, también cuenta con numerosas extensiones que añaden multitud de funcionalidades por medio del sistema de gestión de paquetes pip. Este funciona de forma similar a la herramienta Composer explicada anteriormente. Una de sus extensiones es Django ReST Framework que permite crear una API web con la que comunicarse por medio de peticiones HTTP.

También utiliza el paradigma Modelo-Vista-Plantilla que se trata de una variante del planteamiento del Modelo-Vista-Controlador. Así, el modelo representa los datos y sirve como medio de comunicación con la base de datos. La plantilla muestra estos datos de forma dinámica gracias al lenguaje de plantillas de Django que se mezcla con HTML. Por su parte, la vista es un enlace entre el modelo y la plantilla, por el cual el usuario cambia el modelo [20].

Por otro lado, relacionado con la base de datos implementa migraciones y un ORM propio. Las migraciones permiten crearla y modificarla por medio de los modelos. Mientras, para comunicarse con ella empleando métodos en vez de código SQL se recurre a el ORM.

A nivel de seguridad se caracteriza por tener una gran robustez. Así, proporciona una forma segura de administración de cuentas y contraseñas, facilitando la creación del inicio de sesión. Igualmente, también facilita protección contra inyección SQL o CSRF.

Para terminar, dispone de otras muchas ventajas como la simplificación de los formularios HTML, la autenticación y los permisos de usuarios, la serialización de datos en JSON o XML, o el sitio de administración que posee por defecto. Asimismo, implementa un sistema de pruebas de validación por defecto. Como resultado se pueden comprobar tanto los elementos visuales como la parte interna del proyecto.

2.4.4 Elección de la herramienta

Ya vistas varias formas de desarrollar la página web, se determina aquella que va a ser elegida para la realización de este trabajo.

En este punto se va a emplear Django no solo por una cuestión de unidad ya que está basada en Python al igual que SPADE, sino que también se vio como una oportunidad para aprender un framework nuevo. Además, cuenta con una extensión muy sencilla tanto de añadir como de implementar para la comunicación con el sistema multiagente. Todo ello, incorporando mucha seguridad, facilidades a nivel de back-end y pruebas de validación por defecto.

Capítulo III
Desarrollo del proyecto

En este capítulo se encuentra el camino seguido para la creación del trabajo. Así, se define desde la metodología empleada hasta los requisitos necesarios, los diseños creados y el desarrollo de la página web y el sistema multiagente.

3.1 Metodología

3.1.1 Metodología ágil Scrum

Durante el desarrollo del proyecto se ha decidido seguir la metodología Scrum ya que nos permitirá trabajar con un producto mínimo viable, es decir, un producto con suficientes características para satisfacer a los clientes iniciales. Gracias a ello, nos aprovecharemos de su simplicidad, inspección, adaptación y de la satisfacción y cercanía con el cliente [21].

Para comenzar se trata de un sistema simple ya que las tareas están identificadas en todo momento, indicando su trabajador, objetivo, el tiempo esperado de creación y el resultado esperado. También destaca su inspección, pues, al estar realizando constantes reuniones y revisiones se pueden detectar fallos y mejorarlos.

Además, el proyecto se encuentra en continua adaptación, pues, gracias a las reuniones con el cliente se pueden modificar las características finales del proyecto en cualquier momento del desarrollo. Esto permite encontrar la satisfacción del cliente ya que forma parte del trabajo y puede ver el proceso de desarrollo. De esta forma, es capaz de elegir que hacer cada sprint y de revisar el trabajo realizado al final de cada uno.

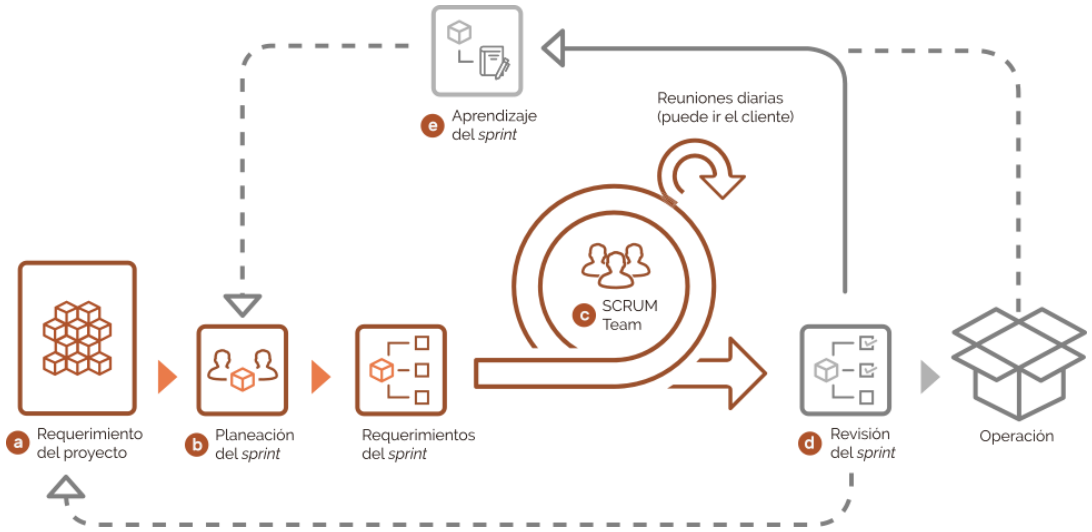

Figura 1: Proceso Scrum. Fuente: https://www.na-at.com.mx/lean-factory

Una vez vistos los motivos por los que he elegido la metodología Scrum, analizaré el procedimiento llevado a cabo.

Primero, se crea una lista de requerimientos u objetivos del proyecto que se deberán de cumplir al final del desarrollo. En ella se encuentran tanto las funcionalidades a llevar a cabo como los distintos diseños y otros aspectos. Tras esto ya es posible la realización de los "*Sprints*", que son cuadros de tiempo fijo durante el cual se crea un producto.

Al inicio de cada Sprint se hace la planificación en la cual se elige qué tareas de la lista de requerimientos se realizarán durante su transcurso. A continuación, en los requerimientos del sprint, a cada miembro del equipo de desarrollo se le asigna las tareas a elaborar. A lo largo del Sprint se hacen reuniones diarias en las que los integrantes informan de su estado con las tareas. De esta manera, comentan los problemas que tienen y la planificación que llevan para desarrollarlas.

Finalmente, cuando acaba un Sprint se da pie a la revisión en la que se muestran los avances establecidos al cliente. Como resultado, obtenemos un aprendizaje del sprint donde el equipo se reúne para valorar los resultados, y reflexionar acerca de los comentarios del cliente y de las mejoras.

En mi caso, he realizado Sprints de siete días de duración de manera que mi cliente era el tutor Javier Palanca. Además, al realizar el trabajo en solitario no se han podido realizar reuniones diarias, y la asignación de tareas ha sido siempre sobre mí mismo.

3.1.2 Herramienta de control de versiones

Las herramientas de control de versiones te permiten asegurarte el tener en todo momento el proyecto versionado. Esto permite controlar los cambios realizados en el código, teniendo la oportunidad de volver a versiones anteriores y de realizar una mejor gestión de las diferentes versiones que se van produciendo en cada sprint.

En este caso, elegí utilizar Git con dos ramas diferentes para el desarrollo del trabajo. Mientras que la rama develop es la que modifico y uso para el desarrollo diario, en la rama main guardo el producto mínimo viable, es decir, el proyecto con los avances que son funcionales y sin errores. De esta forma, al finalizar un avance se realiza una fusión de la rama develop en la main.

3.1.3 Roadmap

En este punto se explica el proceso que se ha llevado a cabo a lo largo del tiempo para la creación del trabajo.

Primero, se hace un estudio del estado del arte en el cual se analiza la situación tecnológica. Así, se valora la etapa actual en el mundo de las subastas, y en el desarrollo web y de inteligencia artificial. También se profundizan en las subastas y sus tipos, estudiando los comportamientos de apuesta que se pueden llevar a cabo en cada una.

A continuación, se diseña todo lo que posteriormente será implementado de manera que podamos plantearnos los posibles errores y anticiparnos a ellos. En este apartado diseñaremos todo a nivel visual como mockups o logotipos, y también a nivel de programación como la base de datos y sus métodos.

Tras ello, se implementan los diseños en el producto final. Comienzo con la parte de desarrollo web ya que será la base para la comunicación con los agentes inteligentes por medio del API. En ella se crean las vistas y los modelos a los que se les aplica la funcionalidad buscada. Además, cuenta con autorización y autenticación para añadirle seguridad. Una vez finalizada, se crea la parte de inteligencia artificial, es decir, los agentes y los diferentes comportamientos que adoptan en cada tipo de subasta.

Finalmente, se realizan los tests para comprobar que todo se realiza tal y como había sido planeado. En este punto existen pruebas de validación tanto para la parte de modelos como para las vistas, el API y el sistema multiagente.

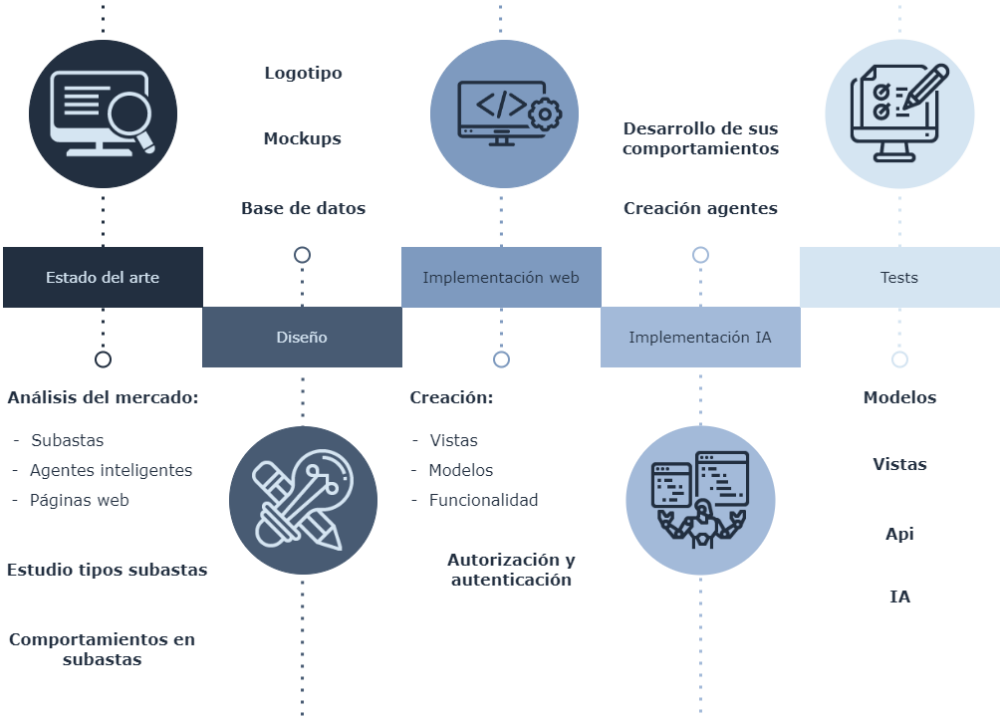

Figura 2: Roadmap

3.2 Análisis de requisitos

La propuesta principal del trabajo es la creación de una página web con tres tipos diferentes de subastas, al igual que un sistema de multiagentes que sean capaces de pujar en la página. Por consiguiente, para la implementación de ambas partes se han de cumplir una serie de requisitos.

En la parte de desarrollo web la condición principal es mostrar los diferentes productos que se encuentran a la venta y hacer que el usuario sea capaz de realizar pujas para poder comprarlos. Dichas pujas serán almacenadas en una base de datos para luego poder analizar los datos de las ventas. Todo ello empleando un diseño intuitivo y de fácil funcionamiento.

Otro punto importante es que deben ser capaces de crear sus propias subastas de distintos tipos. Sin embargo, para poder interactuar con todas estas funcionalidades de la página el usuario debe de estar dado de alta. De forma que cuente con un apartado de perfil en el que se muestre tanto su información personal como sus ventas y subastas ganadas y participadas.

En la parte de inteligencia artificial es necesario conseguir que el usuario pueda crear un agente que sea capaz de pujar por él. Para ello, se ha de diseñar y crear una API ReST que permita la comunicación entre Django y SPADE, que es la plataforma de sistema multiagente elegida para este trabajo.

Asimismo, lo que se busca es crear una serie de comportamientos y estrategias a seguir en función de los distintos tipos de subastas. Para ello se ha de realizar un estudio previo de estrategias a seguir en cada forma de subasta online y un análisis posterior de qué método es más recomendable en cada momento. Como consecuencia, se requiere que el sistema sea escalable, es decir, que sea sencillo incorporar más comportamientos al sistema sin inconvenientes para avances futuros del trabajo.

En ambas partes, es imprescindible que el funcionamiento se adecúe a las reglas de las subastas y del tipo de subasta. Así, en general no podrán pujar en las subastas acabadas, tampoco si el comprador no está dado de alta o el precio de la puja supera al mínimo. Además, en subastas ascendentes tampoco se puede crear una puja si el precio es menor a la puja actual.

3.3 Diseño

3.3.1 Arquitectura del proyecto

Como hemos explicado anteriormente, para elaborar la parte web vamos a trabajar con el framework Django que implementa el paradigma Modelo-Vista-Plantilla[22]. De esta forma, el modelo se encarga del acceso y validación de la información, así como la relación entre los datos y su comportamiento. Por su parte, la plantilla decide como será mostrada la información, y la vista relaciona el modelo y la plantilla, de manera que elige qué información será mostrada y en qué plantilla.

Una vez ya se conoce el funcionamiento del Modelo-Vista-Plantilla se va a explicar que ocurre en el momento que un usuario entre en la dirección o URL. Primero, se accede al mapa de direcciones en el que cada ruta está asociada a una vista. Tras esto la vista, en caso de necesitar datos de la base de datos, los solicita al modelo. Este crea una consulta y, cuando recibe la información, la envía de nuevo a la vista. En este punto, se los transmite a la plantilla que contiene la lógica de presentación. Finalmente, el navegador recibe la página final con todos los datos necesarios y se lo muestra al usuario.

Además, Django se basa en una arquitectura Cliente-Servidor en la que la página web es el cliente, también conocida como el frontend. Para ello, el propio framework tiene integrado un servidor o backend donde está alojada la base de datos. Asimismo, cuenta con una página web de administrador que permite la modificación de los datos de forma sencilla.

Por otro lado, tal y como se ha visto en el estado del arte para la parte de inteligencia artificial se va a utilizar SPADE[30]. Por consiguiente, se ha de utilizar un servidor XMPP en el que cada agente inteligente es un nuevo usuario en el servidor XMPP y se comunican por medio de él. Asimismo, para la comunicación con el backend de la página web se ha utilizado una API ReST que es una extensión de Django llamada Django ReST Framework. A través de este framework el sistema multiagente es capaz de pujar en las subastas.

A continuación, se muestra una imagen a modo resumen de la arquitectura que sigue el trabajo en la parte web y la parte de sistemas multiagentes. Así, la página web se comunica por medio de una arquitectura Cliente-Servidor que usa el Modelo-Vista-Plantilla, y el sistema multiagente necesita de un servidor XMPP y se comunica con los datos mediante la API.

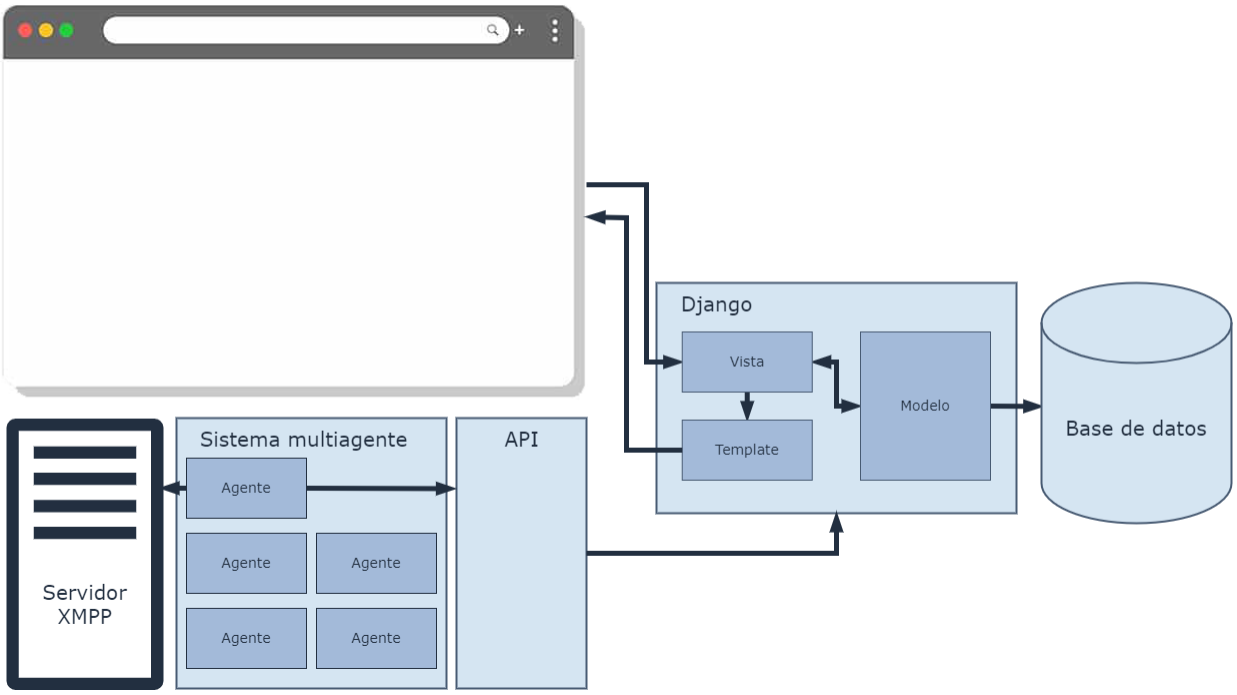

Figura 3: Arquitectura del proyecto

3.3.2 Base de datos

Debido a que Django utiliza un ORM no se va a diseñar la base de datos sino objetos que representan los modelos de datos. Así, se convierten los datos usando programación basada en objetos. Como resultado, cada tabla se representa con un objeto en el framework, y los campos de datos de la clase se corresponden con un campo de la tabla [22] .

En el diseño se muestra elementos propios de Django que no vamos a profundizar, como pueden ser la interfaz del administrador o las sesiones. En cuanto a los modelos utilizados han sido el usuario que ya viene predefinido y los creados por mí. Entre los creados por mi encontramos el producto, la puja, la imagen y el perfil.

Con respecto al producto, contiene el título, la descripción, el tipo de subasta, la cantidad, la medida, el origen, el vendedor, el ganador, y la fecha de inicio y la de fin. También forman parte la puja inicial y la puja final que dependiendo del tipo de subasta tienen una funcionalidad u otra. En relación con el vendedor y el ganador, se corresponden con un usuario predefinido del framework por medio de una relación uno a uno, es decir, cada usuario puede ser muchas veces ganador o vendedor, pero un ganador o vendedor solo se refiere a un producto.

Por otro lado, la puja está conformada por el precio, el producto, el comprador y la fecha de creación. Con el fin de desempeñar esta función, la puja tiene una relación una a muchas con el producto.

Adicionalmente, el modelo de imagen elegido consta con la propia imagen en sí y el producto al que se refiere. Como ya hemos explicado anteriormente, para poder llevarlo a cabo se tiene que aplicar una relación una a muchas.

Finalmente, el perfil añade información al usuario preestablecido de Django. Para ello, posee un campo llamado usuario que tiene una relación uno a uno, en otras palabras, cada usuario tiene un perfil. De esta forma, los datos añadidos son la foto de perfil y la valoración que posee.

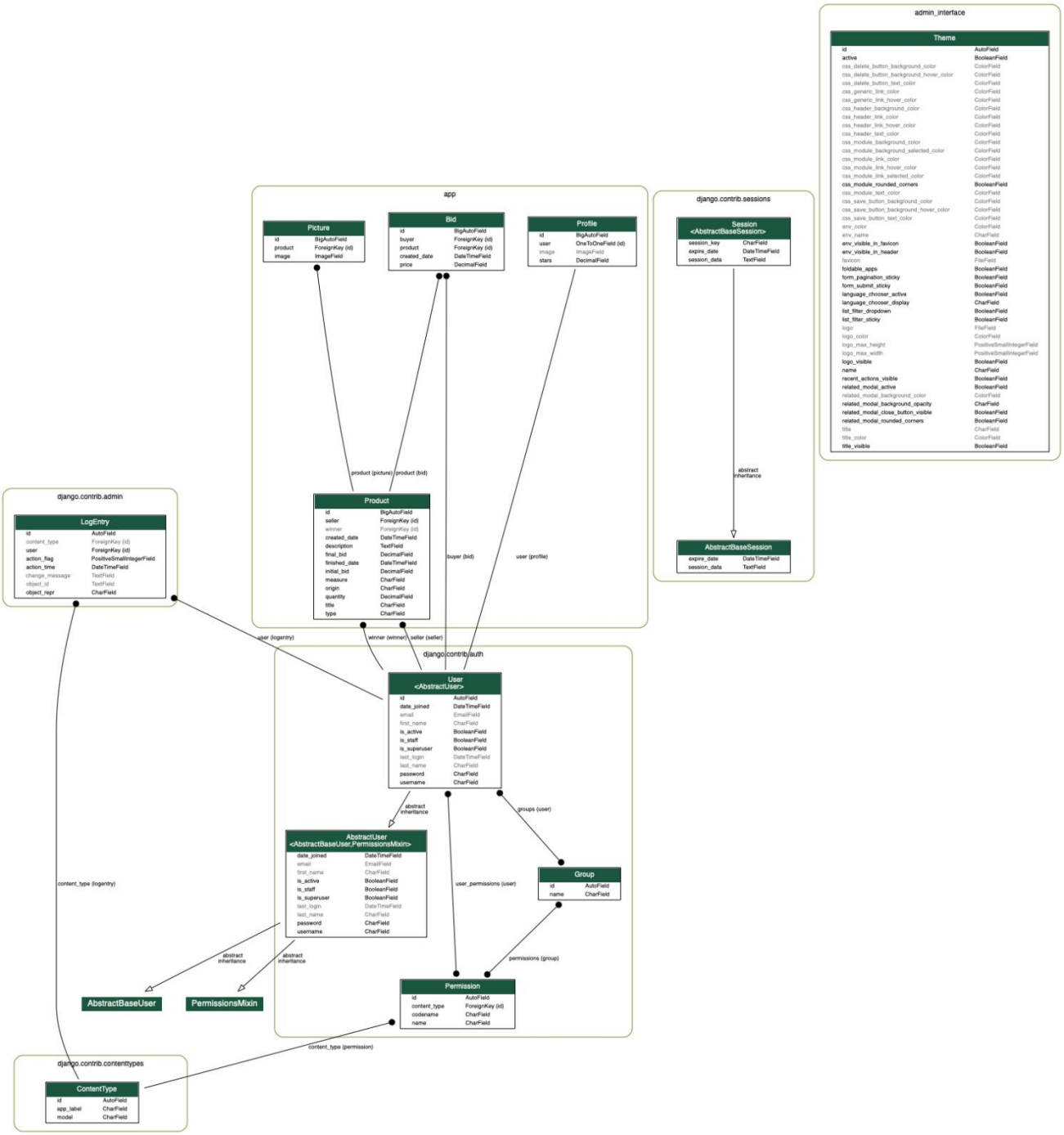

Figura 4; Diseño de la base de datos

3.3.3 Logotipo y Mockups

Antes de comenzar a diseñar hay que elegir cual va a ser la paleta de color que se va a seguir durante todo el trabajo. En este apartado he decidido escoger una serie de tonos azules que abarcan desde el azul oscuro hasta el celeste. Esto se debe a que el color azul transmite confianza y honestidad, dos valores que queremos representar en la página web [23].

Una vez ya contamos con la paleta de colores, se decide crear el logotipo de la página web. Para su elaboración se toma la decisión de utilizar el celeste ya que es visible tanto en fondos claros como oscuros. Al tratarse de una página de subastas online se establece como nombre “4Sale” que significa “En venta”.

Figura 5: Paleta de colores y logotipo

Como se va a crear una página web, se busca que sea accesible tanto para ordenadores como para móviles y tablets. Por ello, se van a realizar los diseños para cada uno de los dispositivos. En estos el principio que se va a seguir es implementar todos los objetivos preservando un estilo sencillo y elegante.

A continuación, se va a mostrar el diagrama de navegación, es decir, la arquitectura de la página web. De esta manera, aparecen las seis vistas diferentes, en la que las vistas de sección y producto son distintas en cada tipo de subasta.

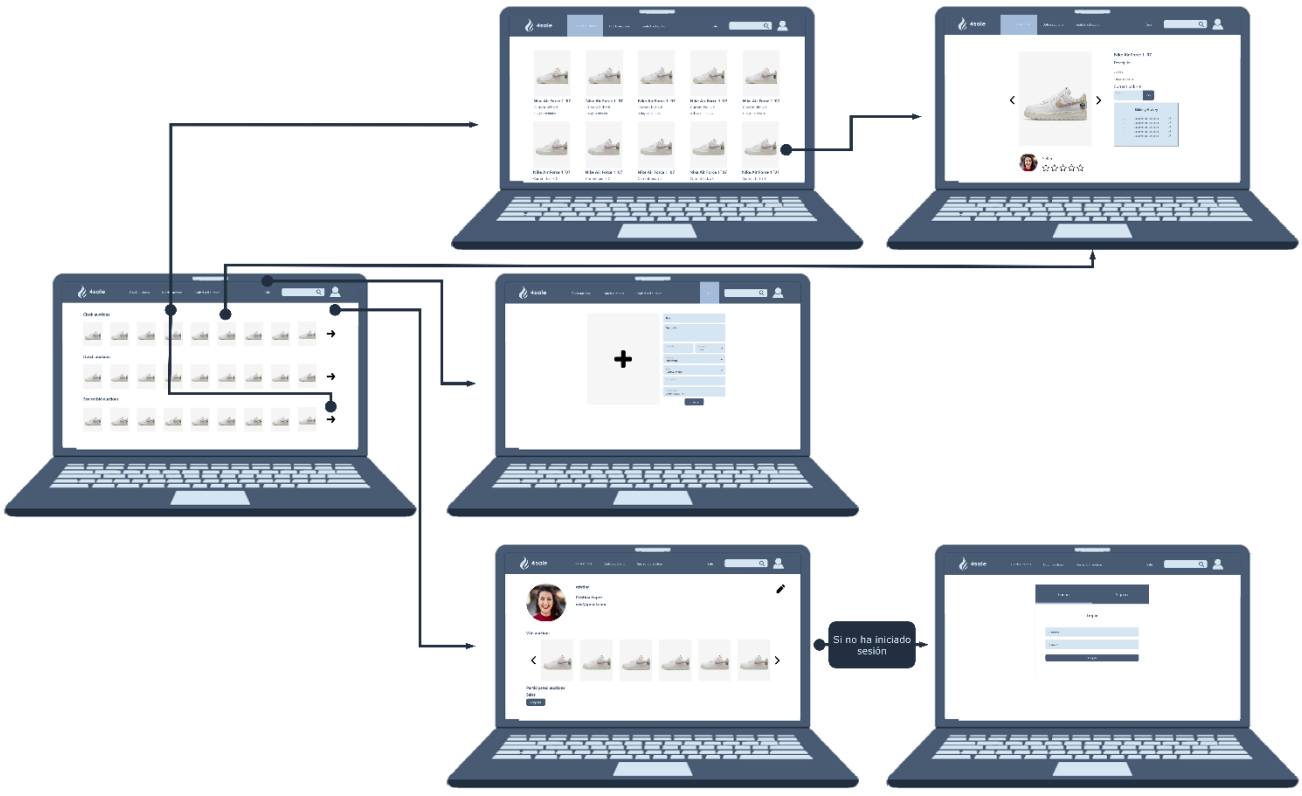

Figura 6: Diagrama de navegación

Comenzando con los mockups se va a comentar la barra de navegación superior que aparece en cada vista del trabajo. En ella están localizados los elementos que más van a usar los consumidores para que hagan el menor número de clicks posibles.

En orden de izquierda a derecha aparece el logotipo que mueve al usuario a la vista principal. Tras esto, se muestran los tres tipos de subastas que existen, todos ellos redireccionan a la página de sección. A continuación, se encuentra un texto de venta que cambia a la vista para crear una puja. Después hay un buscador de productos que localiza las ventas por su título. Finalmente, se representa el icono de un usuario el cual cambia a la página del perfil de usuario si ha iniciado sesión, o si no ha iniciado sesión. un icono que lleva al usuario a la página para iniciar sesión o darse de alta.

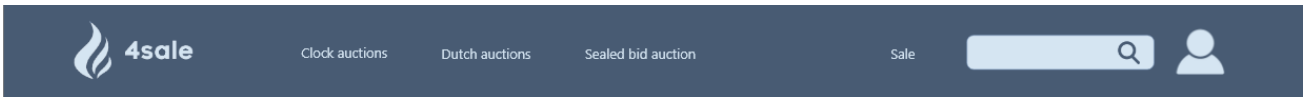

Figura 7: Barra de navegación superior

La pantalla principal que se muestra al entrar en la página web pretende exponer los productos más relevantes para el cliente. Así, se muestran las subastas de cada tipo que se encuentran en activo en orden cronológico descendente, es decir, aquellas que se van a acabar antes. De esta forma, el usuario puede tanto acceder a ese producto en específico como acceder a todas las pujas de este tipo si haces click en la flecha.

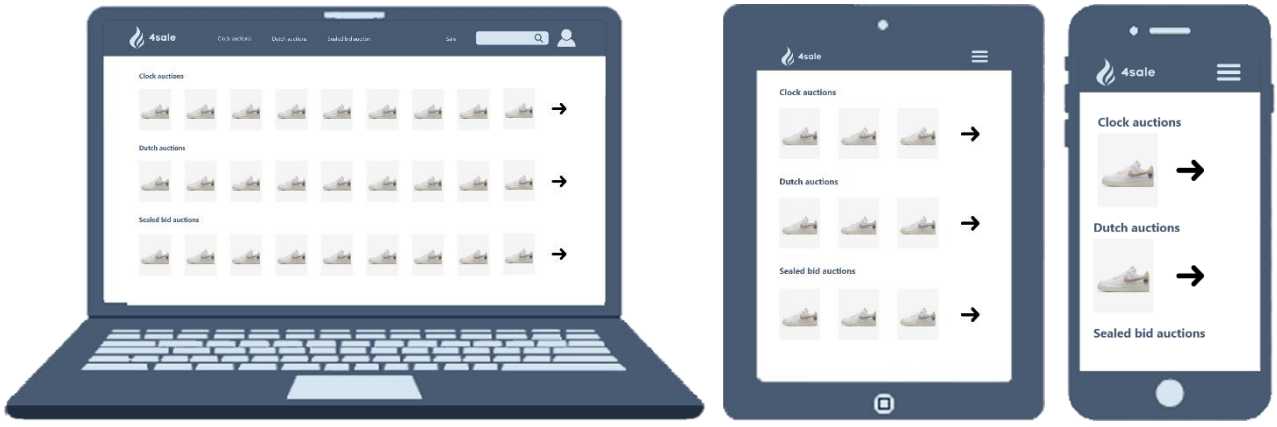

Figura 8: Vista de la página principal

En la vista de la sección se sigue un estilo similar a páginas web estandarizadas donde se venden diferentes productos como Ebay o Nike. Por consiguiente, el cliente estará familiarizado con el diseño de manera que usará la aplicación con facilidad.

Analizando el diseño, se muestran todas las subastas del tipo seleccionado en orden cronológico descendente. Así, cada producto se representa con su imagen, su título, la puja actual y el tiempo restante. Estos elementos tienen un papel muy importante ya que el comprador solo entra a las subastas de productos que les interesa que tienen un precio aceptable.

Finalmente, como existen tres tipos de subastas completamente diferentes la puja actual y el tiempo restante se muestran de forma distinta en la página. Por un lado, en las subastas de sobre cerrado no se muestra la puja actual ya que su objetivo es que no se conozca el precio. Por otro lado, en las subastas holandesas que el precio va disminuyendo en función del tiempo, se expone la puja actual y no el tiempo restante.

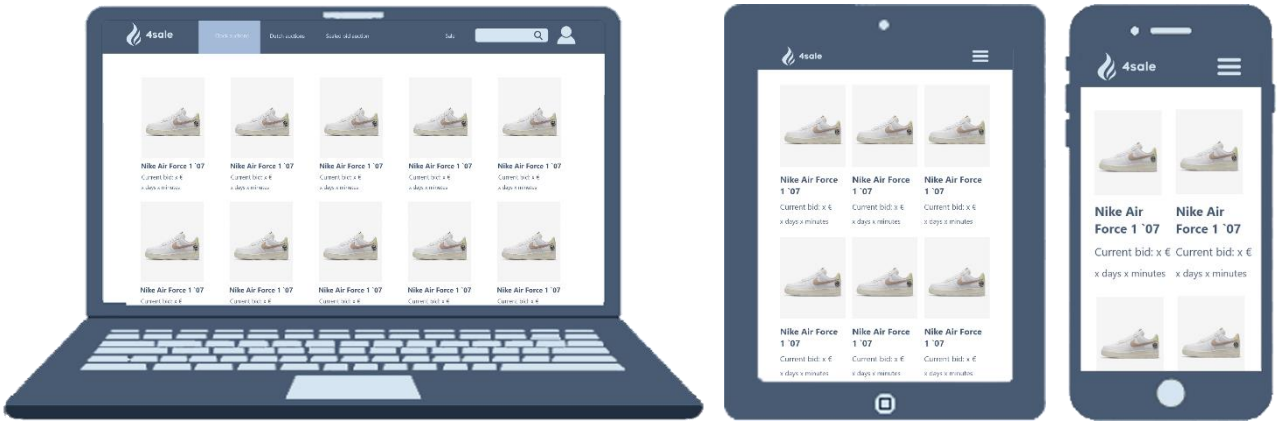

Figura 9: Vista de la página de sección

Desde el comienzo se busca en el diseño promover que el comprador interactúe en la página, es decir, que se sienta seguro a la hora de hacer la puja. Esto se debe a que en esta vista se muestra la función principal del trabajo, la subasta.

En ella podemos encontrar toda la información representativa del producto, desde sus imágenes, título y descripción hasta la foto de perfil y calificación del vendedor. Sin embargo, en este punto el elemento más importante es el campo que permite a los clientes pujar.

Por otro lado, dependiendo del tipo de subasta se despliega una información u otra en la página. De esta forma, en las pujas convencionales se representa tanto el tiempo restante, el precio actual de la subasta como un histórico de pujas ordenadas de mayor pago a menor. En cambio, en las holandesas se muestra solo el precio actual de puja, y en las de sobre cerrado solo el tiempo restante.

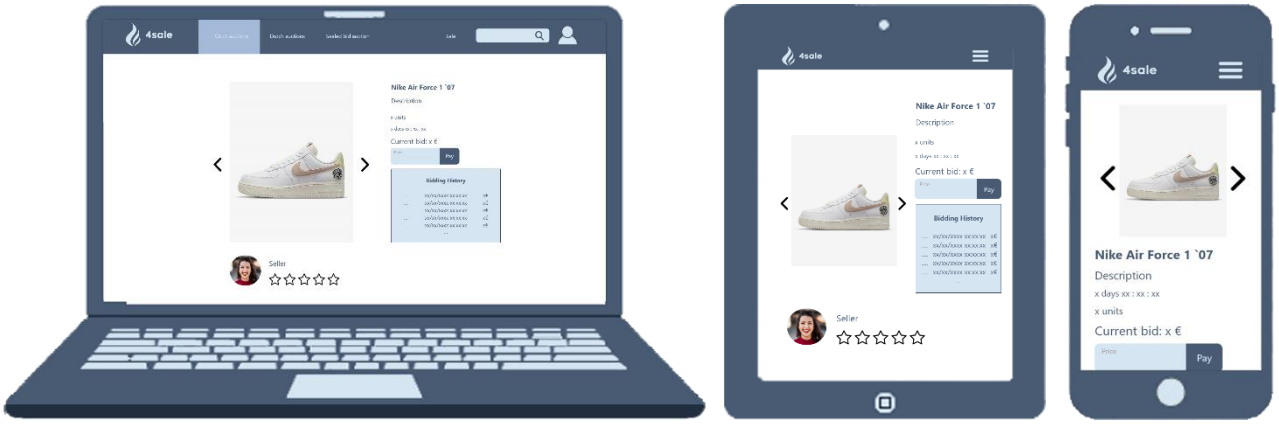

Figura 10: Vista de la página de producto

A continuación, en la página de venta se representa como crear una nueva subasta. Para ello, existe un formulario compuesto por varios campos, así como, por un cuadro donde añadir imágenes.

Entre los campos podemos encontrar el título, la descripción, la cantidad, la medida, el origen, el tipo de subasta, el precio inicial, y la fecha fin. Si se selecciona el tipo de subasta holandesa, se habilita un nuevo campo que es el precio mínimo, y el precio inicial se modifica al precio máximo. De esta forma, la subasta va del precio máximo al precio mínimo hasta la fecha en que se acaba la subasta. En cuanto al cuadro, se transforma en un carrusel donde se muestran las imágenes añadidas, y tiene la posibilidad de añadir o borrar imágenes del carrusel.

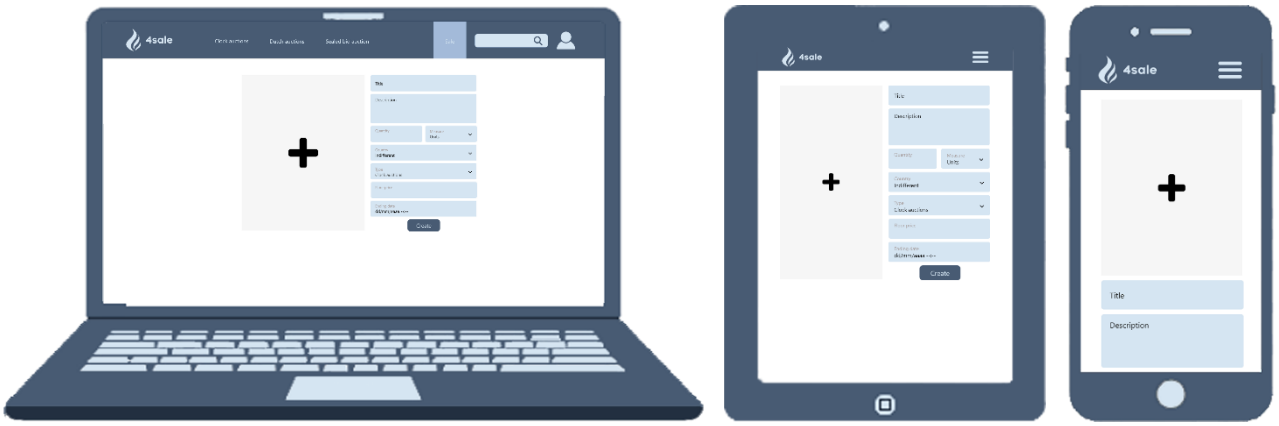

Figura 11: Vista de la página de venta

Tras esto en la página del perfil aparece toda la información con respecto al usuario y su participación en las subastas. Asimismo, también se permite editar todos los campos correspondientes y la foto de perfil.

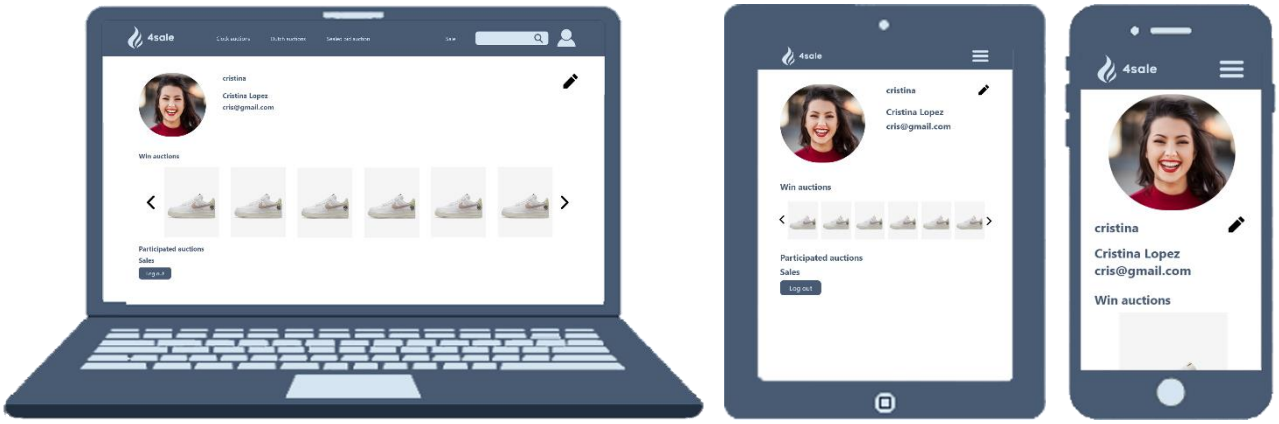

Figura 12: Vista de la página de perfil

Para concluir, la opción de inicio de sesión o darse de alta solo se mostrará en caso de que el consumidor no esté iniciado sesión. Como resultado, se muestra una carta entre la que puedes cambiar entre la opción de registrase y la de ingresar, siendo esta última la que aparece por defecto.

Por un lado, para iniciar sesión se necesitan de los campos del nombre de usuario y la contraseña. Por otro lado, para crear una cuenta se necesita también un nombre, apellido, correo electrónico y una segunda contraseña de confirmación.

Figura 13: Vista de la página de inicio de sesión y creación usuario

3.4 Implementación web

Como ya se ha comentado anteriormente, durante la fase de desarrollo se busca trabajar con un producto mínimo viable. De esta forma, se realiza una creación progresiva en base a pequeños avances en las vistas. Como resultado, se sigue un proceso en el que primero se construye el proyecto de forma global, y tras esto se van añadiendo las diferentes funcionalidades a las páginas.

3.4.1 Front-end

En este apartado se busca implementar todos los mockups realizados en el apartado anterior de manera responsive, es decir, que se adapten al tamaño de la pantalla. Al tratarse Django de un framework que hace uso del paradigma Modelo-Vista-Plantilla, este proceso se elabora en la vista y el template.

Con el fin de construir el diseño se han utilizado varios lenguajes como HTML, CSS o Javascript. En cuanto al HTML, nos permite definir el contenido de la página web, en nuestro caso, también se pueden utilizar diferentes tags propias de Django para la modificación de la vista. Por otro lado, el uso de CSS modifica el estilo de la página web para que tenga la misma forma y colores que el diseño. Con respecto al empleo de Javascript se emplea para modificar la vista ante una acción o un tiempo.

Con el objetivo de simplificar la creación de la web se han utilizado Bootstrap, Flexbox, Fontawesome y SweetAlert2. La combinación entre Bootstrap y Flexbox ha servido tanto para la creación del menú superior y los carruseles de imágenes como para colocar los elementos de la forma buscada en función del tamaño de la pantalla [24] y [25]. Además, para facilitar el uso de iconos se ha recurrido a la librería Fontawesome ya que representa los iconos con formato y características de un texto HTML [26]. Finalmente, SweetAlert2 permite la representación de pop ups personalizados [27].

3.4.2 Modelo de datos

Durante el transcurso de este apartado se aspira a desarrollar los diseños de la base de datos. Para poder hacer uso de los servicios, este framework provee de la conexión a un servidor local.

Como Django emplea una ORM, se crean los distintos objetos que representan cada tabla en los modelos. Como resultado, para poder hacer modificaciones o consultas a la base de datos se tendrán que realizar por medio de estos modelos creados. Además, si se quiere modificar el modelo Django facilita una herramienta llamada migraciones que permite mantener la consistencia de la base de datos.

Como se ha visto en la sección 3.3.2 Base de datos, los modelos que se han creado en esta sección son el producto, la puja, la imagen y el perfil.

3.4.3 API

Como se ha explicado en la arquitectura, en el proyecto se recurre a una API para la comunicación entre la página web y el sistema multiagente. Para ello, Django cuenta con la extensión llamada Django ReST Framework.

Entre las razones por las que está extensión ha sido elegida se hallan su fácil uso y personalización, la autenticación y la serialización. En cuanto a la usabilidad y personalización se destaca que se pueden limitar las vistas a mostrar, solo usando aquellas basadas en funciones más sencillas. Finalmente, la autorización emplea paquetes para OAuth1a y OAuth2 y la serialización admite fuentes de datos ORM y no ORM [28].

De cara a implementarla se deben de crear los serializers, los viewsets y las direcciones. De esta forma, los serializers son una representación del modelo desplegado en formato JSON. Por su parte, los viewsets muestran la información desplegada, así, también pueden restringir el acceso a usuarios no autentificados y elegir los métodos HTTP aceptados. Para concluir, las direcciones relacionan cada ruta con una viewset.

Una vez ya se conoce el funcionamiento se va a explicar que ocurre en el momento en que se accede a la API por medio de la dirección. En primer lugar, se ingresa al mapa de direcciones en el que cada ruta está asociada a un viewset. Tras esto, desde el viewset se solicitan o envían los datos al serializer. Este crea la consulta y, cuando recibe la información, la envía de nuevo. En este punto, el viewset transmite la información al navegador que recibe la página final con todos los datos necesarios y se lo muestra al usuario.

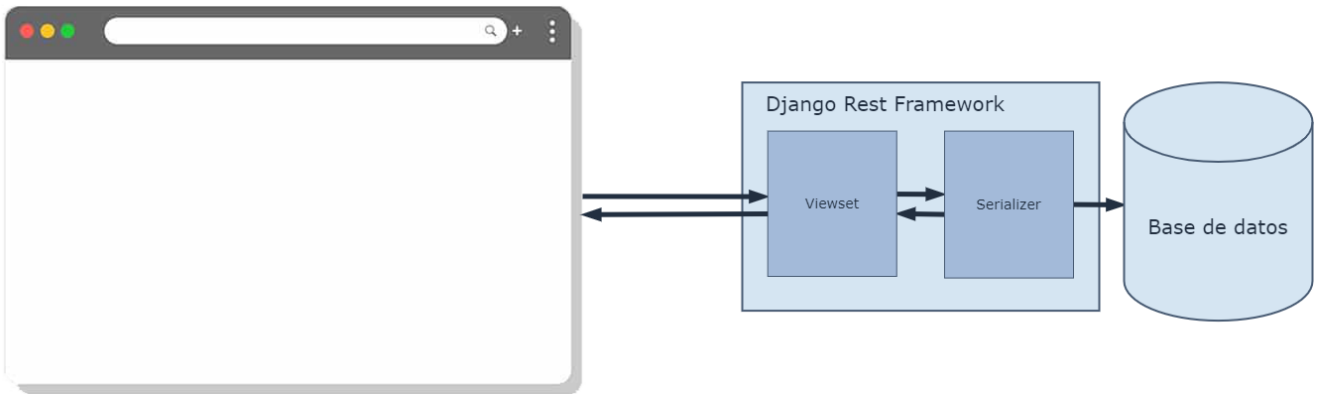

Figura 14: Arquitectura Django ReST Framework

En nuestro caso, se han realizado tres viewsets distintos para conseguir que los agentes inteligentes puedan realizar pujas. Primero, se ha elaborado el UserViewset que permite crear un usuario con el fin de que cada entidad inteligente se corresponda con usuario en la página web. Esto se debe a que no se pueden realizar pujas si no estas dado de alto en la base de datos. Tras esto, se desarrolla un ProductViewset que posibilita la obtención del producto sobre el que se va a realizar la oferta. Por último, se implementa el sistema de puja en el BidViewset, este cuenta con un método con el cual se recibe el precio más alto de la subasta.

Como puede verse en la Figura 15, se representa la vista de un objeto serializado de tipo puja o bid. En dicha imagen se pueden ver los campos del objeto en formato JSON, como por ejemplo son “price”, “created_date”, “product” y “buyer”.

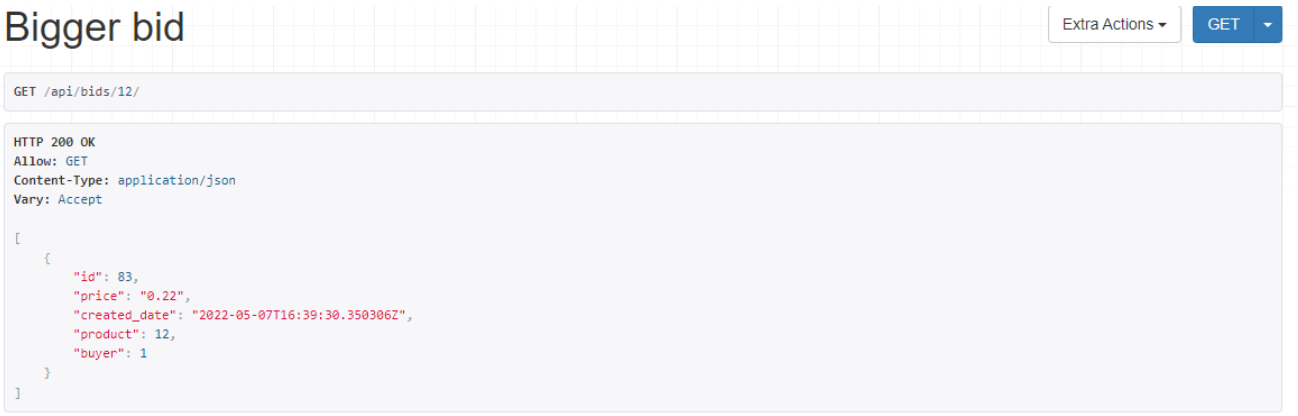

Figura 15: Ejemplo de viewset y JSON de respuesta

3.4.4 Tests

Con el objetivo de realizar una introspección del trabajo realizado y comprobar su correcto comportamiento se ha decidido desarrollar pruebas de validación. Esto se debe a que conforme avanza el desarrollo se ha de verificar que todo tiene un correcto funcionamiento antes de continuar. Para ello se decide crear tres baterías de tests, una de vistas, una de modelos y una de la API. En cada una de ellas, el resultado obtenido debe ser igual a la expectativa esperada.

En el test de las vistas se ha recurrido a la herramienta Selenium, permite la automatización de aplicaciones web con fines de prueba. Así, actúa como si fuese un usuario real y operase en el navegador, permitiendo la interacción con los distintos componentes de la vista [29]. Por consiguiente, en esta prueba comprobaremos que la web responde como se pretende que responda ante los diferentes eventos que se pueden realizar sobre la misma. Por ejemplo comprobar el correcto funcionamiento de la barra de navegación, y de las funcionalidades de la web como la puja, el registro de usuario o el inicio de sesión.

Con respecto a la comprobación de modelos se ha utilizado la librería de tests unitarios de Python unittest. De esta manera, se asegura que cada clase se crea en la base de datos con el formato correcto y, posteriormente, que se obtiene correctamente con todas sus propiedades correctas.

Finalmente, para la revisión de la API también se ha hecho uso de la librería de tests unitarios anterior. Así, se han elaborado tres pruebas de validación, una por viewset. La primera se asegura de que el usuario se cree de forma correcta en la base de datos por medio de la API. A continuación, se verifica que se puede obtener, usando Django ReST Framework, todos los productos o uno en específico. Para concluir, se examina la capacidad de crear pujas y obtener la puja con el precio más alto de un producto.

En todas estas pruebas se revisan los puntos establecidos en el apartado de análisis de requisitos. De esta forma, las exigencias definidas se han de cumplir y, por tanto, han de comprobarse en los tests. Igualmente, entre estas condicione se comprueban también acciones que no se deberían poder realizar en ciertas situaciones como poder pujar sin estar registrado.

3.5 Implementación de los agentes inteligentes

3.5.1 SPADE

SPADE es una plataforma de agentes basada en la comunicación XMPP por medio de un servidor de mensajería instantánea. El utilizar este protocolo nos permite proporcionar un directorio, crear canales de comunicación, y registrar y autenticar a los agentes. Por consiguiente, tras crearse un usuario en el servidor el agente mantiene comunicaciones XMPP con la plataforma de forma automática [30].

De esta forma, a la hora de crear un agente SPADE se necesita de un identificador JID y una contraseña válida para establecer la conexión con el servidor. Este identificador está compuesto por el nombre del agente en la plataforma seguido de una arroba y el dominio del servidor, obteniendo un resultado similar a "*nombreagente@servidor*". En nuestro caso, para la elección de servidor XMPP se ha escogido Openfire ya que tiene soporte para ordenadores Windows.

A continuación, en la figura, se puede observar la forma de crear un agente inteligente en la plataforma. Así, hay que destacar cómo al iniciar la entidad inteligente se devuelve una promesa que espera a que el agente haya terminado de ejecutarse antes de terminarlo, y de esta forma, evitar errores.

```
from spade import agent, quit_spade

class DummyAgent(agent.Agent):
    async def setup(self):
        print("Hello World! I'm agent {}".format(str(self.jid)))

dummy = DummyAgent("your_jid@your_xmpp_server", "your_password")
future = dummy.start()
future.result()

dummy.stop()
quit_spade()
```

Figura 16: Ejemplo de creación de un agente SPADE

3.5.2 Comportamientos

Una vez ya está definido el agente hay que añadirle al menos un comportamiento, es decir, una tarea que ejecuta de forma repetitiva con un objetivo. Para ello SPADE pone a disposición varios comportamientos predefinidos.

Debido a su utilidad a la hora de realizar tareas repetitivas se ha decidido recurrir a comportamientos cíclicos para la elaboración de estrategias de puja. Estos tienen una conducta similar a la de un bucle, pues, carecen de periodo específico.

Como resultado, en los puntos dónde el comportamiento deja de ser funcional se ha añadido el método "*self.kill()*" que se encarga de matarlo. Por ejemplo, se emplea cuando el tiempo de la subasta ha acabado.

Todo comportamiento creado posee un método "*run()*" que se trata del núcleo de la programación. Esto se debe a que en cada iteración del bucle se llama al método, es decir, funciona como el cuerpo del mismo. Además, para inicializar el código se emplea el método "*on_start()*" ya que se ejecuta justo antes de que comience el bucle.

Finalmente, con el objetivo de darle un comportamiento a un agente determinado se debe utilizar el método "*add_behaviour()*" sobre el mismo. Así, para ello se ha de añadir el comportamiento deseado en el primer parámetro.

```
import asyncio
from spade.behaviour import CyclicBehaviour

class MyBehav(CyclicBehaviour):
    async def on_start(self):
        print("Starting behaviour . . .")
        self.counter = 0

    async def run(self):
        print("Counter: {}".format(self.counter))
        self.counter += 1
        if(self.counter >= 5): self.kill()
        await asyncio.sleep(1)


dummy = DummyAgent("your_jid@your_xmpp_server", "your_password")
future = dummy.start()
future.result()

dummy.add_behaviour(MyBehav)
while not dummy.behaviours():
    try:
        time.sleep(1)
    except KeyboardInterrupt:
        break
dummy.stop()
```

Figura 17: Ejemplo de creación de un comportamiento SPADE

Una vez conocido como funciona un comportamiento en SPADE, se desarrollan cuatro diferentes. Todos tienen como objetivo ganar la subasta, estableciendo su precio de puja en base a distintos criterios. En general se basan en cuál es el precio máximo que el cliente quiere pagar para obtener el producto.

La primera estrategia desarrollada se trata de la “Agresiva” que se basa en el grado de agresividad del usuario. Así, su valor varía entre cero y uno, suponiendo el uno una conducta muy agresiva y el cero nada agresiva.

De esta forma, en las subastas ascendentes el agente realizará numerosas ofertas si se posee este tipo de comportamiento. Algo similar ocurrirá en las subastas holandesas en las que la entidad tomará un gran riesgo y esperará a que el precio sea lo más bajo posible. Mientras, en las subastas de sobre cerrado también buscará obtener el producto de la forma más barata posible dependiendo del factor de riesgo que el comprador le haya permitido tomar.

Posteriormente se buscó el planteamiento opuesto, el “Tímido”, que se centra en el grado de timidez. Este también posee un valor que varía entre cero y uno, siendo el uno la actitud más tímida y el cero nada tímida.

Como resultado, la táctica a seguir en las subastas ascendentes se basa en que el agente realizará pocas pujas, actuando de forma poco participativa. Por su parte en las subastas holandesas, se tomará un planteamiento seguro de manera que optará por hacer una oferta con el precio superior a lo esperado por el producto. De igual manera ocurrirá en las subastas de sobre cerrado ya que intentará conseguir la venta sin asumir riesgos, es decir, pagando más de lo esperado.

A continuación, la táctica elegida a desarrollar sería una “Aleatoria” en la que el precio pujado se trata de un número al azar entre el precio mínimo de la subasta y el máximo. Por consiguiente, la entidad puede optar por tomar una actitud agresiva, tímida, intermedia o cualquier otra estudiada.

Por último, se exploró crear una táctica “Informada” en la que el comprador conoce un rango de precios entre los que van a pujar sus contrincantes. Ante esta situación, se decide crear un rango de riesgo que tomará y que variará igualmente entre cero y uno, de forma que uno será arriesgar mucho y cero será no arriesgar.

Por tanto, en las subastas ascendentes comenzará a realizar ofertas en el momento que conozca que el resto de los compradores están llegando a su precio máximo. Además, en las subastas holandesas intentará realizar una puja justo antes de que el primer cliente que conozca haga la suya, obteniendo el producto. Concluyendo, el pago será normalmente superior al del resto en las subastas de sobre cerrado debido a que conoce el rango de precios que van a ser pagados en ella.

Capítulo IV
Resultados

En este capítulo se van a estudiar la eficacia de las estrategias implementadas en la fase de desarrollo mediante pruebas de validación de 2 minutos de duración. De esta manera, se analizarán los mejores comportamientos en cada tipo de subasta por medio de tablas.

4.1 Evaluación de comportamientos en subastas ascendentes

En este apartado se realizará una prueba en la que se comprobará que los compradores con precio máximo de compra más alto son aquellos que ganan las apuestas. Por ello, en este test, que se repetirá en varias ocasiones, se analizarán los resultados de la venta de una subasta ascendente con un precio inicial de 20€.

Asimismo, cuatro agentes con un comportamiento diferente intentarán pujar por el producto. En cada prueba poseerán un precio máximo distinto aleatorio, de manera que se asegura que el tipo de comportamiento no interfiere en el resultado.

PRECIO MAXIMO DE PUJA	GANADOR
TIMIDO	TIMIDO
ARBITRARIO	ARBITRARIO
ARBITRARIO	ARBITRARIO
INFORMADO	INFORMADO
TIMIDO	TIMIDO
ARBITRARIO / INFORMADO	INFORMADO
ARBITRARIO	ARBITRARIO
AGRESIVO / ARBITRARIO	AGRESIVO
TIMIDO	TIMIDO
INFORMADO	INFORMADO
ARBITRARIO	ARBITRARIO
AGRESIVO	AGRESIVO
TIMIDO / ARBITRARIO	TIMIDO
AGRESIVO / TIMIDO	TIMIDO
INFORMADO	INFORMADO
TIMIDO / ARBITRARIO	TIMIDO
TIMIDO	TIMIDO
ARBITRARIO	ARBITRARIO
AGRESIVO / ARBITRARIO	AGRESIVO
INFORMADO	INFORMADO
AGRESIVO	AGRESIVO

Figura 18: Tabla con los resultados de las pruebas de validación

En esta imagen se demuestra que los agentes inteligentes con un precio de compra más alto son aquellos que ganan la subasta. Esto ocurre en el 100% de las ocasiones, de forma que la persona que decida ofrecer un mayor precio siempre será aquella que obtenga el producto, independientemente de la estrategia seguida.

Este hecho nos lleva a pensar que estrategia es la más eficaz en el caso de que todos los agentes posean el mismo precio máximo. Nuestro planteamiento es que la entidad informada es aquella con mayor porcentaje de victoria ya que, en la mayoría de las ocasiones, pujará su máximo de primeras al conocer que el resto también poseen un máximo similar.

Para ello, se llevará a cabo una repetición de una subasta ascendente con un precio inicial de 20€. De esta manera se ha decidido crear cuatro agentes con comportamientos distintos entre sí, y un precio máximo de puja de 30€. Además, la entidad informada contará con un cierto grado de riesgo (seis sobre diez), y conocerá los precios de puja del resto con un rango de 4€ máximo. Así, dado que el resto han decidido pagar 30€, el informado conocerá que han realizado una oferta entre 28€ y 32€.

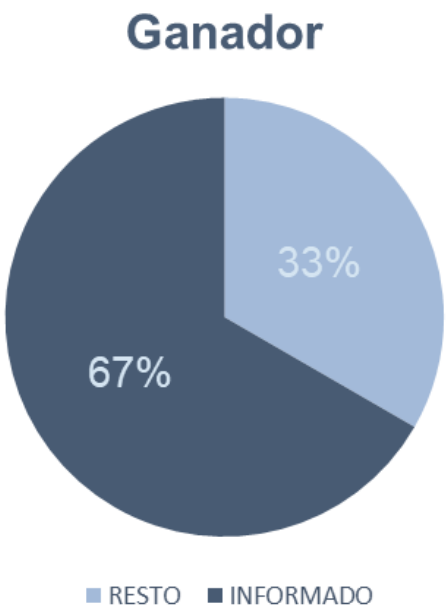

Figura 19: Gráfico con el resultado del test

Como resultado, la gráfica mostrada comprueba que el comportamiento informado es el indicado en caso de que cuenten con el mismo precio máximo. Así, en dos de cada tres casos resultará vencedor de este tipo de subastas. Además, esto se produce tomando un cierto nivel de riesgo, de forma que si se llega a contar con un nivel bajo, el porcentaje de victoria sería mucho mayor.

4.2 Evaluación de comportamientos en subastas holandesas

A continuación, se va a exponer una prueba en la que se demuestra que el precio máximo de compra no es decisivo en este tipo de subastas. De esta forma, se repetirá en varias ocasiones una subasta holandesa con un precio inicial de 50€ y un precio final de 20€.

Así, dos agentes intentarán conseguir el producto, uno con comportamiento agresivo y uno con un comportamiento tímido. Por un lado, el agresivo contará con un precio máximo de puja entre 40€ y 50€, y una estrategia muy arriesgada de forma que busca ganar pagando un precio muy inferior al máximo. Por el otro, el tímido poseerá un precio máximo entre 26€ y 34€, contando con un planteamiento muy seguro y llegando a ofrecer prácticamente su límite.

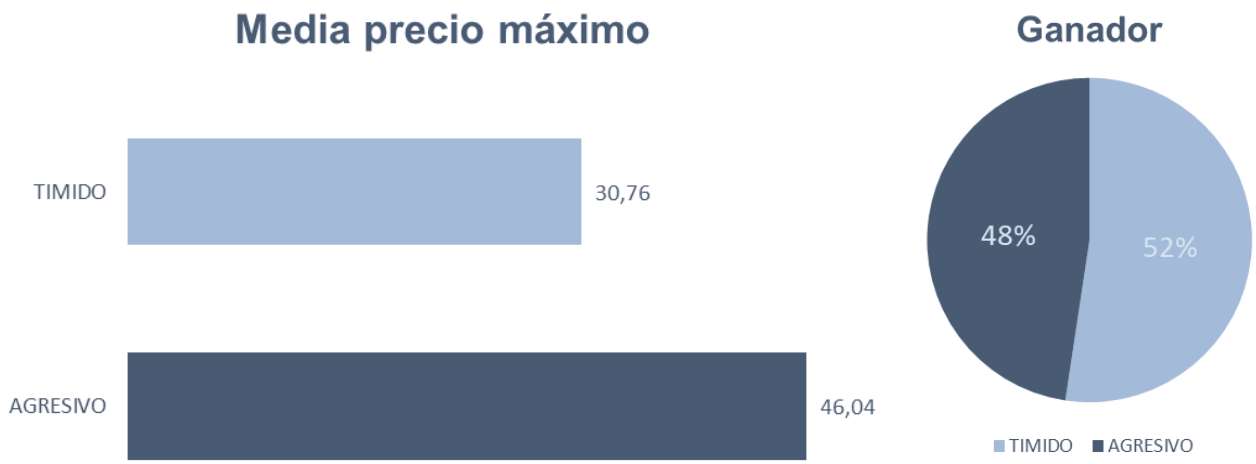

Figura 20: Gráficos de los resultados obtenidos

Como podemos observar en las gráficas, el comportamiento agresivo en la mayoría de las ocasiones no obtiene el resultado esperado debido a que intenta esperarse demasiado para realizar una oferta. Esto demuestra nuestra promesa de que el precio máximo que estes dispuesto a pagar no es definitivo en este tipo de subasta, sino que también influye lo arriesgado o seguro que se desee ser.

4.3 Evaluación de comportamientos en subastas de sobre cerrado

Finalmente se intentará demostrar que la estrategia informada es la mejor opción para este tipo de puja. Para ello, se realizará una prueba de validación en numerosas ocasiones que demuestre nuestra hipótesis. Esta consistirá en una subasta de sobre cerrado con un precio inicial de 20€.

En ella participarán cuatro agentes inteligentes, cada uno con un tipo de comportamiento distinto. Así, aquellos agentes con un planteamiento diferente al informado realizarán una oferta entre 20€ y 30€. Mientras, el enfoque que vamos a analizar contará con un grado de riesgo intermedio, y conocerá los precios de puja del resto con un rango de 8€ máximo. Esto quiere decir que si un agente inteligente ha decidido pagar 27€, el informado conoce que ha realizado una oferta entre 23€ y 31€.

PRECIO PUJADO		
RESTO	CONOCIDO	DIFERENCIA
29	28	-1
23,43	29	5,57
26	28	2
30	30	0
26	27	1
29	28,5	-0,5
23,49	30	6,51
24	25	1
22	22	0
28	28,5	0,5
26	27	1
27	28	1
29	30	1
25	26,5	1,5
30	31,5	1,5
27,79	29,5	1,71
26,08	29,5	3,42
27	27,5	0,5
30	31,5	1,5
26	26	0
29	29	0
MEDIA		1,34333333

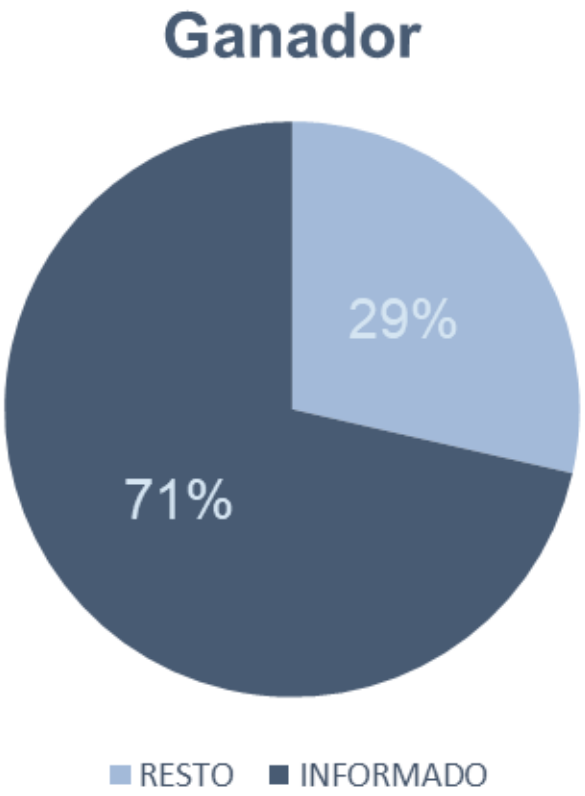

Figura 21: Gráfica y tabla del test llevado a cabo

De nuevo se puede comprobar que nuestro planteamiento estaba en lo correcto. De esta forma, el comportamiento informado casi en la totalidad de veces consigue ganar este tipo de subastas. Asimismo, si se hace un análisis del precio que ha pujado con respecto al resto la diferencia es prácticamente la justa para vencer. Además, esto se produce tomando un nivel de riesgo moderado, si se llega a contar con un bajo riesgo el porcentaje de victoria sería prácticamente perfecto.

Capítulo V
Conclusiones

En este capítulo se van a presentar los resultados del trabajo. Para ello se observarán cuáles son los objetivos que se han conseguido adquirir durante el desarrollo y se analizarán cuáles son los puntos que se pueden mejorar.

5.1 Conclusiones

Como se ha podido comprobar a lo largo del documento, durante la realización del trabajo se han logrado cumplir los objetivos planteados en la introducción.

Para ello se han logrado desarrollar con éxito las dos partes principales del proyecto, tanto la página web de subastas como el sistema de agentes inteligentes con diferentes comportamientos capaces de pujar automáticamente. De esta forma, los usuarios pueden realizar las acciones por su cuenta a través de la página web o crear un agente capaz de hacerlas por él. Entre estas acciones se localizan la posibilidad de crear sus propias subastas y la de efectuar una puja.

Además, con el fin de conseguir este resultado se ha llevado a cabo un proceso de análisis de las tecnologías necesarias, ya que no eran familiares. De esta manera, se realizó un estudio del estado del arte actual de los agentes inteligentes y el desarrollo web, valorando las diferentes opciones de implementación. En este se incluían tanto las fortalezas de cada herramienta como las debilidades incluidas. Igualmente, se obtuvo información sobre el mundo de las subastas online, revisando los tipos de subasta. Finalmente, se realizó un estudio de las diferentes estrategias existentes sobre subastas para, posteriormente, ser incluidos en los comportamientos de los agentes y decidir cuál es la mejor opción en cada ocasión.

Asimismo, también se ha comprobado su correcto funcionamiento por medio de pruebas de validación a lo largo del trabajo y se ha asegurado la seguridad en la página web. Por ello, para poder realizar una acción el usuario debe estar dado de alta. Al igual ocurre con el sistema de pujas y con la API, que se encuentran protegidos en el back-end de la aplicación.

Por otro lado, se ha realizado un estudio de los diferentes comportamientos que pueden adoptar los agentes. De esta manera, podemos comprobar que siguen la estrategia de forma correcta y que, en la mayoría de los casos, el ganador de la subasta es aquel que cuenta con el precio de puja más alto.

Finalmente, ha supuesto una gran oportunidad para exponer los conocimientos recibidos a lo largo del grado e incluso aprender nuevas herramientas y tecnologías. En este punto se han empleado las asignaturas de programación, Inteligencia Artificial, el proyecto de desarrollo web y el Diseño de interfaces y experiencia de usuario. Así como se ha adquirido experiencia en el mundo de los sistemas multiagente y en el framework Python Django, así como sus extensiones.

5.2 Propuestas de mejora

En este punto se debe hacer una reflexión de aquellos elementos que han quedado pendientes, que se plantean como trabajo futuro o que se pueden mejorar. Así, no solo se ofrece un mejor servicio, sino que también añade más funcionalidades y herramientas al proyecto.

De esta forma, en la parte de creación del sistema multiagente sería interesante crear una interfaz gráfica con la que poder interactuar con los agentes. De esta manera, cualquier usuario podría crear una entidad inteligente y añadirle un comportamiento, y no solo aquellos que sepan programar y conozcan el funcionamiento de los mismos. Además, se podrían añadir más estrategias que den un amplio abanico de posibilidades a los clientes.

Finalmente, sería interesante que en vez de elegir el producto por el que quieres que puje, contar con varios vendedores del mismo producto y que el agente compre la cantidad esperada, agregando diversos proveedores. Así, por ejemplo, si quiero obtener diez kilogramos de tomates podría comprarle tres a un vendedor y siete a otro siempre minimizando el coste.

Capítulo VI
Anexo

6.1 Manual de usuario

El proyecto se puede encontrar tanto en el anexo a este TFG como en el siguiente enlace a Github: https://github.com/DavidJmz04/TFG. En él aparece por un lado la parte de desarrollo web por medio de Django, y por otro lado, la parte del sistema multiagente, empleando la herramienta SPADE.

Si se quiere ejecutar la parte web se deben seguir varios pasos. En todos ellos el usuario necesita abrir la consola de comandos de su sistema operativo y encontrarse localizado sobre la ruta raíz del proyecto.

Primero, hay que inicializar el entorno virtual sobre el que se encuentra Django, para lo que se ejecuta el comando "*myvenv\Scripts\activate*" en Windows o "*source myvenv/bin/activate*" en Linux o Mac. Esto nos permite contar con la configuración solo dentro de ese entorno y que no afecte a otros proyectos dentro del mismo ordenador. Después, con el fin de asegurar que se encuentran instaladas todas las dependencias se ejecuta el comando "*pip install -r requirements.txt*" en todos los sistemas operativos. A continuación, por medio de "*python manage.py migrate*" se importan los modelos en la base de datos local. Finalmente, para iniciar el servidor local y utilizar el proyecto se ejecuta "*python manage.py runserver*". Así, una vez realizado este paso, el usuario debe dirigirse a la URL http://127.0.0.1:8000/ de su navegador.

Otras funcionalidades interesantes que pueden ser probadas son la página de administrador, la página del API y la ejecución de las pruebas de validación. A fin de crear un administrador y poder entrar en su página se aplica "*python manage.py createsuperuser*" y se rellena los campos requeridos. Tras esto, puede visualizar la página web entrando a la URL http://127.0.0.1:8000/admin/ y completando los datos del usuario creado anteriormente. Las diferentes consultas a la API se pueden ver y realizar accediendo a http://127.0.0.1:8000/api/. Para concluir, es posible comprobar el correcto funcionamiento de los modelos, las vistas y la API por medio del comando "*python manage.py test*".

Por otro lado, en cuanto a la parte de inteligencia artificial se ha empleado SPADE, y por lo tanto se debe de contar con un servidor XMPP local. En nuestro caso se ha utilizado Openfire con un plugin llamado REST API que nos permite crear usuarios por medio de una petición HTTP [31]. Como resultado, a la hora de crear un agente se llama a una función que crea automáticamente un usuario en el servidor.

De esta manera a la hora de crear un agente inteligente se debe importar la clase BidAgent del archivo agent.py. Posteriormente se ha de inicializar, pasándole como parámetros el JID del usuario XMPP y la contraseña del mismo. Así, se realizaría de la siguiente forma "*BidAgent("nombreagente@servidor", "mi_contraseña")*". Tras comenzar la actividad del agente, habría que añadirle uno de los comportamientos creados en el archivo behaviours.py con la función "*add_behaviour(comportamiento)*".

6.2 Referencias

[1] Klein, Stefan y Keefe, Robert. “The Impact of the Web on Auctions: Some Empirical Evidence and Theoretical Considerations”. En: Evolutionary Computation - EC. 3 (junio de 1998), págs. 5-6. DOI: 10.1080/10864415.1999.11518338.

[2] Botti, Vicente J. y Julián, Vicente J. “Estudio de métodos de desarrollo de sistemas multiagente. Inteligencia Artificial”. En: Revista Iberoamericana de Inteligencia Artificial (2003). ISSN: 1137-3601. URL: https://www.redalyc.org/articulo.oa?id=92501806

[3] Wilson, Robert. “Handbook of Game Theory with Economic Applications”. En: Elsevier (1992), págs. 227-279. ISSN: 1574-0005. DOI: https://doi.org/10.1016/S1574-0005(05)80011-6

[4] Frutos, David. “Sistema multi-agente para la coordinación de UAVs mediante negociaciones concurrentes”. En: Universidad Castilla-La Mancha (2017), págs. 55-56. URL: https://ruidera.uclm.es/xmlui/handle/10578/15435

[5] Tarifa de Ebay. URL: https://www.ebay.es/help/selling/fees-credits-invoices/comisiones-y-tarifas-para-vendedores-particulares?id=4822

[6] “Las casas de subastas ceden comisiones: más ventas pero con menos rentabilidad”. En: El Cronista (2018). URL: https://www.cronista.com/edicionimpresa/Las-casas-de-subastas-ceden-comisiones-mas-ventas-pero-con-menos-rentabilidad-20180809-0060.html

[7] Lucking-Reiley, David. “Auctions on the Internet: What’s Being Auctioned, and How”. En: The Journal of Industrial Economics, Vol. 48, No. 3 (2000), págs. 227-252. URL: http://www.jstor.org/stable/117554

[8] Harkavy, Michael y Tygar, J. D. “Electronic Auctions with Private Bids”. En: USENIX (1998). URL: https://www.usenix.org/legacy/event/ec98/full_papers/harkavy/harkavy.pdf

[9] Delgado, Leandro M. y Monteserin Ariel. “Sistema Multiagente de Negociación Automática Basada en Diálogos Expresivos para Compras en Ámbito Municipal”. En: Universidad Nacional (2015), págs. 32-42. URL: http://www.ridaa.unicen.edu.ar/xmlui/handle/123456789/582

[10] Marchetti, Tulio J. y García, Alejandro J. “Plataformas para Desarrollo de Sistemas Multiagente. Un Análisis Comparativo”. En: Red de Universidades con Carreras en Informática (mayo 2003). URL: http://sedici.unlp.edu.ar/handle/10915/21445

[11] JACK. URL: https://hmong.es/wiki/JACK_Intelligent_Agents

[12] JADE. URL: https://jade.tilab.com/

[13] Mulet, Lluís. “Estudio Interno de la Plataforma Multiagente MadKit”. En: UPV (2006). URL: http://www.gti-ia.upv.es/sma/tools/Magentix/archivos/documentation/report_madkit.pdf

[14] “Agentes inteligentes para la identificación de sistemas en procesos biotecnológicos de tratamiento de residuales”. URL: https://1library.co/article/metodolog%C3%ADa-zeus-metodolog%C3%ADas-dise%C3%B1o-sistemas-multiagente-sma.z3dexl7y

[15] Javier Palanca, Andrés Terrasa, Vicente Julian y Carlos Carrascosa. “SPADE 3: Supporting the New Generation of Multi-Agent Systems”. En: EEE Access, Vol. 8, (2020) págs. DOI: 10.1109/ACCESS.2020.3027357https://riunet.upv.es/bitstream/handle/10251/129903/Frayle -

Integraci%C3%B3n de agentes deliberativos en la plataforma SPADE: Desarrollo de pGomas.pdf?sequence=1

[16] Tacilla, Julio L. “Sistema informático web de gestión de incidencias usando el framework AngularJS y Node.js para la empresa Redteam Software LLC”. En: Universidad Privada Antenor Orrego (2016), págs. 14. URL: https://hdl.handle.net/20.500.12759/3416

[17] Benjamin Barslev, Behnaz Hassanshahi y François Gauthier. “Nodest: Feedback-Driven Static Analysis of Node.js Applications”. En: European Software Engineering Conference (2019). DOI: https://doi.org/10.1145/3338906.3338933

[18] Armel, Jamal. “Web application development with Laravel PHP Framework version 4”. En: Helsinki Metropolia University of Applied Sciences (2014). URL: https://www.theseus.fi/bitstream/handle/10024/74052/Author.pdf

[19] Subecz, Zoltán. “WEB-DEVELOPMENT WITH LARAVEL FRAMEWORK”. En: University, Hungary (2021). DOI: https://doi.org/10.47833/2021.1.CSC.006

[20] Vainikka, Joel. “Full-stack web development using Django REST framework and React”. En: Metropolia University of Applied Sciences (mayo 2018), págs. 13-16. URL: https://www.theseus.fi/bitstream/handle/10024/146578/joel_vainikka.pdf?sequence=1

[21] Rodríguez, César y Dorado, Rubén “¿Por qué implementar Scrum?”. En: Revista Ontare (2015), págs. 10-12. DOI: https://doi.org/10.21158/23823399.v3.n1.2015.1253

[22] Christopher Ireland, David Bowers, Michael Newton y Kevin Waugh. “The International Journal on Advances in Software vol. 2”. En: IARIA (2009), págs. 38-52. ISSN: 1942-2628

[23] Keller, Eva. “Psicología del color: Cómo actúan los colores sobre los sentimientos y la razón”. En: Gustavo Gili (abril de 2004). ISBN: 9788425219771

[24] Bootstrap. URL: https://getbootstrap.com/

[25] Flexbox. URL: https://developer.mozilla.org/es/docs/Learn/CSS/CSS_layout/Flexbox

[26] Fontawesome. URL: https://fontawesome.com/

[27] SweetAlert2. URL: https://sweetalert2.github.io/

[28] Django ReST Framework. URL: https://www.django-ReST-framework.org/

[29] Selenium. URL: https://www.selenium.dev/

[30] SPADE. URL: https://spade-mas.readthedocs.io/en/latest/readme.html

[31] Openfire. URL: https://www.igniterealtime.org/projects/openfire/