TRABAJO FIN DE GRADO

GRADO EN INGENIERÍA INFORMÁTICA

MENCIÓN EN INGENIERÍA DEL SOFTWARE

APLICACIÓN WEB PARA CREAR UNA PÁGINA OFICIAL QUE DE SOPORTE A UN JUEGO ONLINE.

Estudiante: Ramiro Serantes Garrido

Dirección:

Dra. Virginia Mato Abad (Facultad de informática de A Coruña)

A Coruña, Septiembre de 2022

Resumen

En este Trabajo de Fin de Grado (TFG) se ha realizado el desarrollo de una aplicación que permite eliminar la necesidad de implementar ciertas funcionalidades dentro de los servidores de Minecraft (aunque es aplicable para todo tipo de juego), para implementarlas en una página web oficial. Este proyecto está motivado por el deseo de liberar la memoria en estos servidores, ya que estos al estar programados en java, es muy valiosa, pudiendo dedicar la liberada a la creación de sistemas más complejos.

Entre muchas funcionalidades que se pueden implementar en la capa web, se ha decidido implementar la gestión de la información de los usuarios/jugadores, una tienda online para que los dueños de los servidores puedan ganar dinero para el mantenimiento y mejora del servidor de Minecraft, un sistema de subastas que permita a los usuarios realizar intercambios entre ellos y un foro que permita la discusión de aspectos relacionados con el juego.

Para el desarrollo del proyecto se ha seguido una adaptación de la metodología de desarrollo ágil Scrum, para la estimación se ha utilizado el método de estimación IFPUG, muy reconocido internacionalmente, se han utilizado tecnologías de código abierto como MySQL, Spring, React, Hibernate…. Y se han empleado herramientas como Intellij, Github, Balsamiq para facilitar el desarrollo del proyecto.

Palabras clave:

- React
- Redux
- REST
- GitHub
- Aplicación web
- Java
- Spring
- Hibernate
- Maven
- IFPUG
- ISBSG
- Balsamiq
- MVC
- Sistemas
- JPA
- Minecraft

Índice General

Índice de Figuras

Capítulo 1

Introducción

En este capítulo se muestran los aspectos básicos para comprender el proyecto que se va a desarrollar sus líneas maestras, la motivación para realizarlo y los objetivos.

1.1 Motivación.

El sector de los videojuegos online multijugador suelen estar siempre en un foco de interés puesto que son una de las mayores formas de entretenimiento de la actualidad y lo seguirán siendo en el futuro. Según las estadísticas de google en Minecraft, en 2020 mensualmente se han encontrado 126 millones de jugadores activos. Esto causa que los servidores de Minecraft, que son llevados por los usuarios (mojang no lleva servidores pese a que es el único motivo por el cual tiene tantos jugadores) tengan la capacidad de mover (y bastante) dinero, ofrecen nuevas formas de juego, como minijuegos dentro de Minecraft, sistemas de trabajos, clases de rol, plugins de ciudades, e.t.c. En estos servidores se puede implementar todo lo que se desee (por eso este juego es prácticamente inmortal) siempre y cuando se sepa programar en java, programar con threads (el problema de estos servidores es que hay que programarlos a bajo nivel en java y no da para muchas más opciones), y el manejo de eventos.

Por ello se desea mejorar la implementación de estos servidores, a través de la creación de una nueva arquitectura con los conocimientos aprendidos durante la carrera universitaria, y la implementación en la capa web de diferentes funcionalidades para liberar la memoria del servidor del juego.

Minecraft al estar programado en java consume bastante memoria, causando que en este tipo de servidores es un recurso vital ya que suele ser bastante limitada, por ello si se abstraen cierta cantidad de funcionalidades a capa web con MVC, podremos liberar memoria y permitir que el servidor la dedique a por ejemplo la creación de sistemas de juego más complejos. (Por este motivo este proyecto es realmente útil, sobre todo en un sector donde se mueve tanto dinero).

La implementación de una nueva arquitectura también es realmente útil, puesto que los servidores de minecraft no tienen un estándar a la hora de programarlos (si se investiga el código fuente de github de varios de ellos, hasta tienen código sql dentro de clases java, es realmente horrible en algunos casos).

1.2 Objetivos.

Debido a que la cantidad de funcionalidades que estos servidores pueden proveer a un jugador/usuario es infinita, se ha escogido la implementación en capa web de cuatro sistemas que deberían estar implementados dentro del servidor de Minecraft, si no fuese por esta abstracción.

Estos cuatro sistemas son:

- **Sistema de usuarios (User System)** : La gestión de los usuarios, es decir, el acceso, creación y modificación de sus datos será implementada en la web.
- **Sistema de foro (Forum System)** : Los usuarios deben poder discutir entre sí los asuntos relacionados con el servidor, como la publicación de nuevas actualizaciones, mini juegos, eventos, e.t.c, por ello se crea un foro para que puedan hacerlo ahí.
- **Sistema de tienda (Shopping System)** : Los dueños del servidor necesitan ganar dinero para el mantenimiento y desarrollo de nuevas funcionalidades, por ello en este sistema se les permite poner elementos que consideren oportunos por dinero. Cabe destacar que existen monedas premium, con las que el usuario realiza las compras, estas monedas se compran con dinero real, el usuario no puede volver a convertirlas a por ejemplo euros, pero puede usarlas dentro del sistema. Es lo más común en este tipo de sistema para permitir a usuarios que no pagan pero juegan la oportunidad de conseguir items de pago comerciando con usuarios que si pagan.
- **Sistema de subastas (Auction System)** : Para permitir un mayor nivel de comercio, se le permitirán a los usuarios poner órdenes para vender sus monedas premium (también se llaman monedas de platino en este sistema), o comprar monedas premium.

1.3 Estructura de la memoria.

La memoria del proyecto estará estructurada en diez capítulos, siendo el primero esta introducción.

- Segundo capítulo: **Tecnologías y herramientas utilizadas**. Se describen todos los elementos usados en la realización del proyecto.
- Tercer capítulo: **Metodología**. Se describe la metodología de desarrollo aplicada.
- Cuarto capítulo: **Análisis**. Se declara una especificación de requisitos formal.
- Quinto capítulo: **Diseño**. Se muestran las decisiones de diseño de las diferentes pantallas de la aplicación , arquitectura y organización.
- Sexto capítulo: P**lanificación y costes**. Se detallan aspectos relacionados con la planificación del proyecto y sus costes asociados.
- Séptimo capítulo: **Implementación**. Se explican las decisiones tomadas en la implementación del proyecto.
- Octavo capítulo: **Pruebas**. Se reúnen todo los tipos de prueba que se han realizado a lo largo del proyecto.
- Noveno capítulo: **Producto Final**. Se muestra como funciona el producto final en la capa web.
- Décimo capítulo. **Conclusiones**.

Capítulo 2

Tecnologías y herramientas utilizadas

Para poder realizar el desarrollo de este proyecto se han utilizado un conjunto de tecnologías y herramientas para facilitar el desarrollo. A lo largo de este capítulo se producirá realizará detalle de cada una.

Esta sección está distribuida en tres partes, el BackEnd, el FrontEnd y las herramientas utilizadas en el desarrollo.

2.1 BackEnd.

El backend se ha creado principalmente usando Java Estandar Edition, MySql, Spring e Hibernate.

2.1.1 MySQL.

MySQL [1] es un sistema de gestión de base de datos relacional, que proporciona la creación de múltiples usuarios con diferentes permisos de accesos a SQL. Es un SGBD de código abierto con gran popularidad usado por grandes aplicaciones como Twitter. Emplea el motor InnoDB.

De entre todas las tecnologías posibles para gestionar la base de datos, se ha escogido MySQL puesto que es de código abierto, requiere pocos recursos, es ampliamente utilizado en muchos sectores y se ha utilizado durante la carrera durante mucho tiempo, aumentando la familiariedad con esta.

Se ha empleado la versión 8.0.27 (La empleada durante la carrera).

2.1.2 Java.

Java es un lenguaje de programación orientado a multiplataforma y a objetos desarrollado por James Gosling y publicado en 1995. Una de las características más importantes de java es que compila a un bytecode y puede ser ejecutado en cualquier máquina virtual de Java, sin importar el software o arquitectura por debajo.

Los principales motivos de su utilización para el desarrollo de la aplicación son, la gran familiaridad con él debido a la carrera, es portátil para cualquier plataforma y tiene una arquitectura que facilita enormemente la construcción de aplicaciones web sobre él.

Se ha utilizado la versión 16.0.2.

2.1.3 Spring.

Spring [2] es un framework de código abierto para el desarrollo de aplicaciones en java. Aplica el patrón de diseño de inversión del control que instancia e inyecta objetos en la aplicación, a los cuales llamará beans.

Las razones por la cuales se ha decidido optar por usar spring, son la familiaridad con este framework debido a la carrera, una configuración sencilla y visual con ficheros de configuración y su gran popularidad y soporte por la comunidad.

Se ha empleado la última versión 2.7.1.

2.1.4 Hibernate.

Hibernate [3] es una herramienta que permite el mapeado de objetos a base de datos, usando el estándar de JPA (Java Persistence API) para el lenguaje java, es de código abierto. Permite mapear objetos de java a entidades con anotaciones y beans.

Se ha optado por esta herramienta puesto que proporciona un formato de consultas a base de datos de forma sencilla y clara, cuenta con un gran soporte en la comunidad y se tiene un gran dominio en esta herramienta debido a su uso durante la carrera.

2.1.5 Maven.

Maven [4] es una herramienta para la gestión de dependencias y construcción de proyectos java. Está basado en un project object model (POM), que es un fichero xml donde se declaran todas las dependencias del proyecto y la forma de construcción del mismo.

Si bien es cierto que tiene una gran capacidad de acoplar plugins al proyecto, se ha optado por el sistema de inclusión de plugins de intellij.

Otra opción sería usar Gradle, que usa un lenguaje específico para la construcción de proyectos, Groovy en vez del xml, pero se ha optado por Maven por la gran familiaridad que se tiene con él al haberse usado durante la carrera.

2.1.6 JUnit.

JUnit [5] es un framework que facilita la construcción y ejecución de pruebas en Java, que permiten evaluar el funcionamiento de todas las funciones involucradas en el sistema.

El motivo principal por el cual se optó por JUnit 5 es que se tiene una gran familiaridad con el framework y es de uso muy popular y extendido.

2.1.7 Pitest.

PiTest [6] proporciona funcionalidades adicionales que trabajan sobre los test creados por JUnit, declarando una serie de mutaciones, alteraciones del código fuente que permitan al código seguir funcionando, que tratan de verificar si los tests que hemos realizado se han dado cuenta de los cambios que ha realizado. Esto favorece enormemente la solidez y confiabilidad en los diseñados e implementados.

2.1.8 ArchUnit.

ArchUnit [7] a la par que PiTest proporciona funcionalidades adiciones sobre los test de JUnit, permitiendo definir un conjunto de reglas a modo de test que verifican las condiciones aplicadas sobre ellas, estas condiciones son aplicadas en su mayoría a verficar si se ha respetado el diseño de la arquitectura o tipos devueltos por interfaces.

2.1.9 Mockito.

Mockito [8] es un framework aplicado a las pruebas unitarias, que permite la creación de objetos de entrada para las funciones que substituyan las dependencias que posean con otras.

La razón por la cual se decidió usar Mockito es debido a su uso durante la carrera y es de código abierto con un gran soporte por la comunidad.

2.1.10 Jacoco.

Jacoco [9] permite realizar un análisis de la cobertura de las pruebas de la aplicación, se puede realizar su configuración a través de ficheros de xml y da reportes visuales acerca de la cobertura alcanzada de forma global en el proyecto.

Se ha decidido su uso debido a que se ha empleado en muchas ocasiones durante la carrera, adquiriendo una gran familiaridad con él y además está integrado con SonarQube (también es muy popular entre la comunidad).

2.1.11 Darkyen's Time Tracker.

Darkyen's Time Tracker [10] permite medir el tiempo programado dentro de intellij, siendo muy útil debido a que integra este resultado de forma inmediata a git, permitiendo tener un seguimiento del tiempo exacto llevado para realizar el proyecto.

2.2 FrontEnd.

Las maquetas del diseño del frontend han sido diseñadas con Balsamiq, mientras que otros aspectos como los diagramas de usuarios se han creado con ayuda de Pencil.

2.2.1 React.

React [11] es una librería JavaScript de código abierto basada en la programación basada en componentes y especialmente diseñada para crear interfaces de usuario con el objetivo de crear páginas SPA (Simple Page Application). Cada componente es una pieza con la que el usuario puede interactuar, se combinan con otros componentes para formar componentes aún mayores y se pueden reutilizar.

Una página SPA es una app web que carga una sola página HTML y realiza su actualización de forma dinámica mientras que el usuario interactúa con ella. Las páginas SPA utilizan AJAX y HTML5 para crear aplicaciones web fluidas.

React es una librería con una gran popularidad, con un gran soporte por parte de la comunidad, proporciona una gran fluidez desde la perspectiva del usuario al tener el DOM virtual.

Se ha utilizado la versión 16.14.0

2.2.2 Redux.

Redux [12] es una librería de JavaScript para la administración del estado de la aplicación en la capa web, se usa frecuentemente con React. Permite gestionar el estado de distintos componentes de la aplicación.

El principal motivo por el cual se ha elegido Redux es por el gran soporte que tiene en la comunidad.

Se ha utilizado la versión 4.0.5.

2.2.2 Plotly.

Plotly [13] es una librería en JavaScript de código abierto construida sobre D3.js y stack.gl. Proporciona un gran número de gráficas, se optó por esta librería puesto que permite implementar gráficas de forma sencilla y otorga una gran cantidad de funcionalidades a la hora de editar estas gráficas.

Se ha usado la versión 1.0.6.

2.3 Herramientas de desarrollo.

Durante el desarrollo de este proyecto se han utilizado diferentes herramientas para la aplicación de buenas prácticas y agilizar el proceso de construcción.

2.3.1 Pencil.

Pencil [14] es una herramienta que proporciona una gran variedad de diseños como mockups, diagramas de usuarios …. Se ha utilizado principalmente para construír los diagramas de historias del usuario.

2.3.2 Dia.

Dia [15] es un software que permite el diseño de diferentes diagramas, es gratuíto y opera en diferentes sistemas operativos, se ha empleado principalmente para crear modelos de bases de datos y entidad-relación. No se ha elegido por algún motivo en especial, solo se necesitaba una herramienta fácil de usar para el diseño de este tipo de diagramas.

2.3.3 Balsamiq.

Balsamiq [16] es un software que permite la creación de pantallas de usuario de forma simple y clara, se ha escogido porque además de cumplir el propósito del diseño de las páginas del usuario, las crea de forma sencilla visualmente sin ir a mucho detalle, permitiendo que el cliente no se haga una idea de que el proyecto está muy avanzando.

2.3.4 SonarQube.

SonarQube [17] es una plataforma de código abierto para la inspección continua de la calidad del código fuente. SonarQube utiliza muchas herramientas de análisis (caja blanca estática) (SpotBugs, Reglas sobre el código….) para obtener una serie de métricas relacionadas a este código fuente, como code smells, vulnerabilidades, e.t.c.

Se ha elegido sobre otras alternativas debido a su integración directa con proyectos de maven y su utilización a lo largo de la carrera.

2.3.5 Github.

Github [18] es una plataforma web de código abierto que permite a sus usuarios almacenar y administrar proyectos de software, permitiendo registrar y controlar los cambios realizados en el proyecto con un control de versiones.

Se ha elegido sobre otras posibles opciones como GitLab, Subversion… Debido a que es considerada como la plataforma más estable y potente, además de tener una gran familiaridad con esta plataforma, por su uso durante muchos años.

2.3.6 Intellij.

Intellij [19] es un entorno de desarrollo integrado (IDE) de código abierto multiplataforma. Permite gestionar y crear proyectos con ayuda de numerosas herramientas que facilitan el proceso de desarrollo, como generación de código automático, auto-save, integración directa con git, un soporte de plugins muy potente que ya te los integra dentro del proyecto, un sistema de organización de paquetes muy bueno.

Se ha elegido por todos los anteriores motivos y además de la gran familiaridad que se tiene con el por su uso durante la carrera. (Además es mi favorito entre todos los IDE's).

Capítulo 3

Metodología

En este capítulo se realizará una descripción de la metodología aplicada en el desarrollo de este proyecto, la cual es una adaptación de la metodología ágil de Scrum.

3.1 Introducción a Scrum.

Scrum es una metodología de trabajo para desarrollo ágil de proyectos, que permite definir unos roles a los cuales se le asignan diferentes responsabilidades y tareas, una serie de artefactos y otorga una estrategia de desarrollo incremental basado en sprints.

Entre las múltiples ventajas que provee esta metodología son:

1. *Al dividir el proyecto en sprints sucesivos permite conocer la velocidad que posee cada equipo a la hora de desarrollar el software, lo que permite estimar cuanto llevaría una historia de usuario.*
2. *Al dividir el proyecto en estos sprints acotados se puede centralizar el esfuerzo en desarrollar una tarea en concreto, mejorando la productividad.*
3. *Al ser un proyecto de una única persona permite eliminar la excesiva necesidad de documentación que presentan otros tipos de metodologías.*
4. *Divide la entrega del producto final en entregas parciales, facilitando la incorporación de cambios dentro de la especificación de requisitos.*

3.1.1 Roles de Scrum.

En Scrum existe una gran variedad de roles, de los cuales se pueden destacar:

1. ***Scrum Master**: Es el encargado de aplicar soluciones a los posibles impedimentos que puedan surgir en cada sprint. No es el líder del equipo, pero se encarga de hacer cumplir las pautas establecidas por Scrum.*
2. ***Development Team**: Es el equipo que va a diseñar y crear el producto final. En este caso el tamaño necesario para realizar el proyecto lo dice el IFPUG.*
3. ***Product Owne**r: Representa a la parte interesada que solicita la creación del proyecto y se encarga de que todos los integrantes del equipo entiendan los elementos presentes en el Product Backlog. También gestiona las historias del usuario.*

3.1.2 Artefactos.

Los artefactos de Scrum son elementos que facilitan la transparencia y el registro de la información asociada a los diferentes tipod de procesos. Deben diseñarse utilizando un lenguaje de negocio. De todos estos artefactos se usarán:

1. ***Sprint Backlog**: Es un conjunto de funciones pertenecientes a una historia de usuario que serán desarrolladas a lo largo de la duración del siguiente sprint. Una vez creadas las historias de usuario en el Sprint Backlog, se detallará como se deben implementar sus requisitos en el sprint. Se dividen en tareas, donde cada miembro del equipo elige la que desee hacer. Una **tarea** es la división de la historia del usuario expresada con lenguaje técnico.*
2. ***Product Backlog**: Es un documento que engloba todo el proyecto. Este documento es un conjunto de todos los requisitos del proyecto organizados por relevancia (en este proyecto son todos igual de relevantes). Solo deberá ser modificado por el Product Owner, conteniendo estimaciones genéricas del proyecto.*
3. ***Historia de usuario**: Son los requisitos del proyecto expresados en un lenguaje técnico desde el punto de vista del cliente. El product Owner es el que se encarga de redactarlo.*

3.1.3 Eventos de Scrum.

Los eventos son una parte vital de la metodología, puesto que crean un patrón y minimizan la necesidad de reuniones indefinidas, permitiendo centrarse en el desarrollo del proyecto, los eventos definen formas de inspección y adaptación de los diferentes aspectos del proyecto.

- **Sprint**: Es una franja de tiempo en la cual se realiza el desarrollo del proyecto, se puede ajustar su duración según se considere necesario.
- **Sprint Planning**: Se trata de un evento anterior a cualquier sprint, en el cual se selecciona la historia de usuario del Product Backlog que se va a realizar durante el sprint. Debe realizarse una estimación de la complejidad y esfuerzo para cada historia de usuario.
- **Daily Scrum**: Reunión diaria con el equipo de desarrollo con el objetivo de la puesta en común de los objetivos alcanzados y los diferentes problemas que han podido surgir.
- **Sprint Review**: Es la primera reunión que se realiza al finalizar un sprint, en la cual se tratan las tareas llevadas a cabo durante este sprint y se revisan.
- **Restrospectiva**: Es complementaria al Sprint Review, donde al terminar un sprint todos los miembros del desarrollo se reúnen para poner en común sus impresiones del trabajo realizado.

3.2 Adaptación de Scrum.

La metodología ágil de Scrum se ha modificado para satisfacer mejor las necesidades del proyecto desarrollado, de entre estas modificaciones se pueden destacar:

1. ***Roles**: En este proyecto todas las responsabilidades recaen en una única persona el autor del trabajo, complementándola en momentos puntuales con la supervisión del director del trabajo.*
2. ***Reuniones**: Al finalizar un sprint se revisa el trabajo realizado durante el tiempo que sea necesario para asegurarse de que se desarrolla el software en una dirección correcta.*
3. ***Daily scrum**: Será la revisión del trabajo realizado en el día anterior.*
4. ***Sprint**: Si bien es cierto que Scrum recomienda que todos los sprints sean constante en el peso, se han organizado de tal forma que estén distribuidos en grupos que presenten funcionalidades similares e insertando sprints más pequeños para aliviar la carga de los sprints más grandes.*

Capítulo 4

Análisis

Este capítulo realizará la descripción de las funcionalidades que debe satisfacer el proyecto para que su desarrollo pueda ser completado. Estas funcionalidades se encontrarán detalladas en la especificación de requisitos y adicionalmente se incluirá el modelo conceptual que poseerá la aplicación.

La creación de la especificación de requisitos es importante realizarla en las etapas tempranas del proyecto, debido a la necesidad de conocer en detalle el alcance global del sistema y las necesidades que debe satisfacer el proyecto.

4.1 Entorno de la Aplicación y descripción general.

El proyecto web desarrollado tendrá como objetivo proporcionar a usuarios que juegan a un servidor de Minecraft funcionalidades adicionales en una capa web, en vez de en el propio juego. La implementación de estas funcionalidades en capa web favorecen la jugabilidad en estos servidores, puesto que al ejecutarse fuera del mismo, permite disponer de más espacio en memoria para realizar otras tareas.

Las funcionalidades adicionales que se pueden aplicar a estos servidores de Minecraft en capa web son muy diversas, por ello se ha tomado la decisión de limitar el tamaño del proyecto a cuatro sistemas :

- **Sistema de Usuarios** : Se deben gestionar los jugadores que participan en el servidor de Minecraft en la capa web.
- **Sistema de Foro** : Los usuarios dentro del servidor de Minecraft deben poder comunicar aspectos relativos al juego en la capa web.
- **Sistema de Tienda** : Los usuarios deben poder comprar elementos del juego proporcionados por la administración en la capa web.
- **Sistema de Subastas** : Los usuarios deben poder comerciar diferentes elementos entre ellos a través de la creación de órdenes de subasta.

4.2 Modelo de Datos.

El modelo de datos permite ofrecer una visión general de la información que será contenida y manejada por la aplicación.
A continuación se muestra el diseño de datos que posee este proyecto. (Figura 4.1)

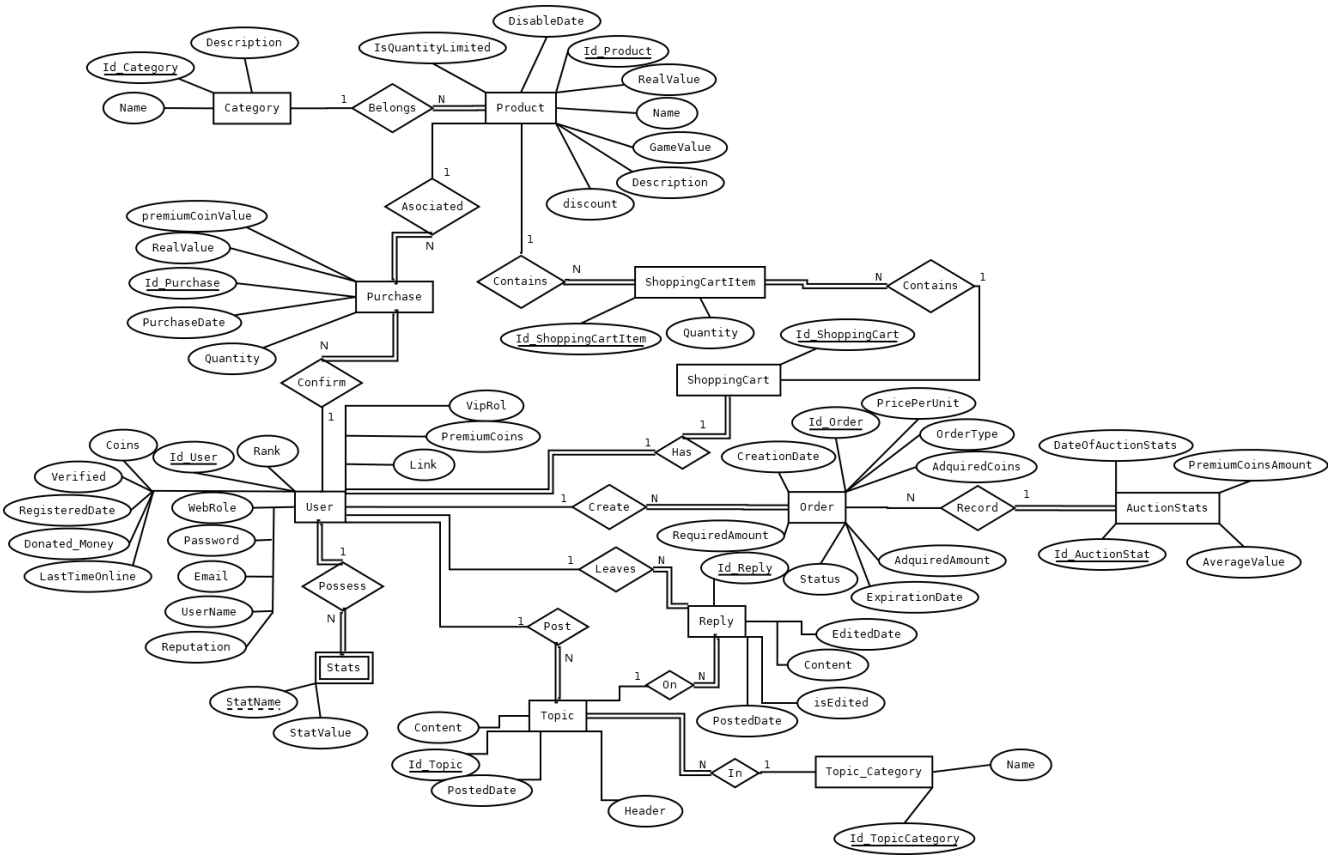

Figura 4.1: Diagrama entidad relación.

A continuación se introduce una breve descripción para entidad y sus correspondientes atributos. Se omitirá la explicación de los id's de cada entidad puesto que son un número generado por la base de datos, excepto por el caso de la entidad débil Stats.

User

Realiza la representación de un jugador en la aplicación web, sus atributos son:

- **Rank**: Los puntos del usuario en el ranking del juego.
- **Coins**: Cantidad de monedas normales que posee el jugador.
- **PremiumCoins**: Cantidad de monedas premium (de pago) que posee el jugador.
- **VipRol**: Rango especial que tiene cada jugador en el juego (pagando premiumCoins).
- **Link**: Dirección web de Mojang donde reside el avatar del usuario.
- **Verified**: Indica si el jugador ha sido verificado por el email.
- **RegisteredDate**: Fecha de registro del usuario.
- **LastTimeOnline**: Fecha de la última conexión del usuario.
- **DonatedMoney**: Cantidad de dinero en euros que ha gastado el usuario en la tienda.
- **WebRole**: Rol que tiene el usuario en la web.
- **Password**: Contraseña asociada al email.
- **Email**: Email del usuario.

- **UserName**: Nombre del usuario en el juego.
- **Reputation**: Reputación del usuario ganada a través de varias acciones.

Stats

Estadísticas que posee el usuario/jugador dentro del juego (fuerza, inteligencia, agilidad, e.t.c), sus atributos son:

- **StatName:** Nombre de la estadística.
- **StatValue:** Valor numérico asociado a esa estadística.

ShoppingCart

Carrito que posee el usuario para incorporar los productos que desee. Solo tiene un id asociado.

ShoppingCartItem

Producto contenido en el carrito del usuario, sus atributos son:

- **Quantity**: Cantidad de unidades del producto que se va a comprar.

Product

Producto de la tienda para ser comprado por los usuarios, sus atributos son:

- **DisableDate**: Día a partir del cual el producto dejará de estar disponible.
- **IsQuantityLimited**: Indica si se pueden comprar más de una unidad de ese mismo producto.
- **RealValue**: Valor del producto en euros.
- **GameValue**: Valor del producto en monedas de platino.
- **Name**: Nombre del producto.
- **Description**: Descripción del producto.
- **Discount**: Descuento aplicado en % al producto.

Purchase

Registro de una compra realizada por el usuario, sus atributos son:

- **Quantity**: Cantidad del producto comprada.
- **RealValue**: Coste del producto en euros.
- **PremiumCoinValue**: Costo del producto en monedas de platino.
- **PurchaseDate**: Fecha en la cual se ha realizado la compra.

Category

Diferentes categorías a las que puede pertenecer los productos, sus atributos son:

- **Name**: Nombre de la categoría.
- **Description**: Descripción de la categoría.

Order

Orden creada por el usuario en el sistema de subastas, sus atributos son:

- **CreationDate**: Fecha de creación de la orden.
- **OrderType**: Tipo de orden, venta o compra.
- **PricePerUnit**: Precio en monedas normales de una moneda de platino.
- **AdquiredCoins**: Monedas normales conseguidas en las órdenes de venta.
- **AdquiredAmount**: Monedas de platino conseguidas en las órdenes de compra.
- **RequiredAmount**: Monedas de platino para transaccionar con otras órdenes.
- **ExpirationDate**: Fecha a partir de la cual no se podrá hacer más transacciones a esta orden.
- **Status**: Estado de la orden, cancelada, expirada, activa, e.t.c.

AuctionStats

Estadísticas de la evolución del precio de las monedas de platino a lo largo del tiempo, sus atributos son:

- **DateOfAuctionStats**: Día en el que se registra los cambios del valor del platino por monedas normales.
- **PremiumCoinsAmount**: Cantidad total de monedas de platino transaccionadas en ese día.
- **AverageValue**: Valor medio por moneda de platino en el día.

Topic

Topic creado por el usuario en el foro, es un inicio de una discusión, sus atributos son:

- **Content**: Contenido del Topic.
- **PostedDate**: Fecha de creación del Topic.
- **Header**: Cabecera del Topic.

TopicCategory

Categorías a las que puede pertenecer un topic, sus atributos son:

- **Name**: Nombre de la categoría del topic.

Reply

Respuestas creadas por usuario con respecto a un topic, sus atributos son:

- **Content**: Contenido de la respuesta.
- **IsEdited**: Indica si la respuesta ha sido editada.
- **PostedDate**: Fecha de la creación de la respuesta.
- **EditedDate**: Fecha de edición de la respuesta.

4.3 Requisitos.

El objetivo principal del análisis es establecer una especificación de requisitos que deberá satisfacer el proyecto. Esta especificación proporciona una visión global de las funcionalidades del proyecto, que permitirá la construcción del mismo.

4.3.1 Actores.

Antes de poder realizar la construcción de la especificación de requisitos es necesario definir el conjunto de actores que van a estar involucrados en el sistema a desarrollar:

- **Usuario no Autenticado**: Usuario que visita la web pero no está autenticado.
- **Usuario Autenticado**: Usuario que ha logrado autenticarse de forma correcta.
- **Usuario Moderador**: Usuario que gestiona el foro.
- **Usuario Administrador**: Usuario que puede gestionar toda la aplicación.

Adicionalmente se ha declarado el actor **Usuario de la Aplicación** que englobará los demás actores presentes en el sistema.
En la siguiente imagen (Figura 4.2) podemos observar un diagrama que especifica el tipo de herencia entre actores. Un Usuario Administrador posee todas las funcionales de un Usuario Moderador, un Usuario Moderador posee todas las funcionalidades de un Usuario Autenticado y este a su vez, posee todas las funcionales de un Usuario de la Aplicación.
Un Usuario no Autenticado posee a su vez las funcionalidades de un Usuario de la Aplicación.

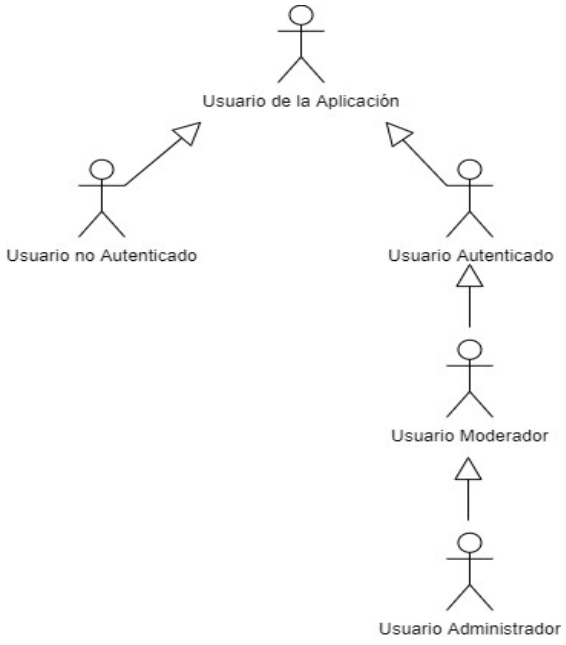

Figura 4.2: Diagrama de herencia de actores.

4.3.2 Historias de usuario.

Tras haber realizado una investigación sobre las necesidades más comunes entre los usuarios que juegan a estos servidores de Minecraft, se ha tomado la decisión de crear cuatro sistemas que engloben un conjunto de necesidades de ámbito similar. Estos sistemas son, Sistema de Usuarios (User System), Sistema de Foro (Forum System), Sistema de Tienda (Shopping System) y el Sistema de Subastas (Auction System). Estos sistemas han sido incluidos en el Product Backlog de Scrum.

Las historias de usuario son los requisitos que deben poseer estos sistemas, explicados de forma breve y concisa. Estas historias deben estar definidas de tal forma que todos los componentes del equipo de desarrollo del proyecto puedan comprender-las fácilmente, además el funcionamiento que representan tiene que estar debidamente acotado.

Las historias de usuario serán implementadas a lo largo de los sprints, a continuación se expone su detalle con sus respectivos diagramas.

4.3.2.1 Historias de usuario. User System.

Como usuario de la aplicación quiero poder autenticarme en mi cuenta, y administrar mis recursos.

Esta historia de usuario hace referencia a la autenticación del usuario y las diferentes acciones que puede realizar el usuario una vez autenticado con respecto a la trata de la información que posee dentro del sistema.

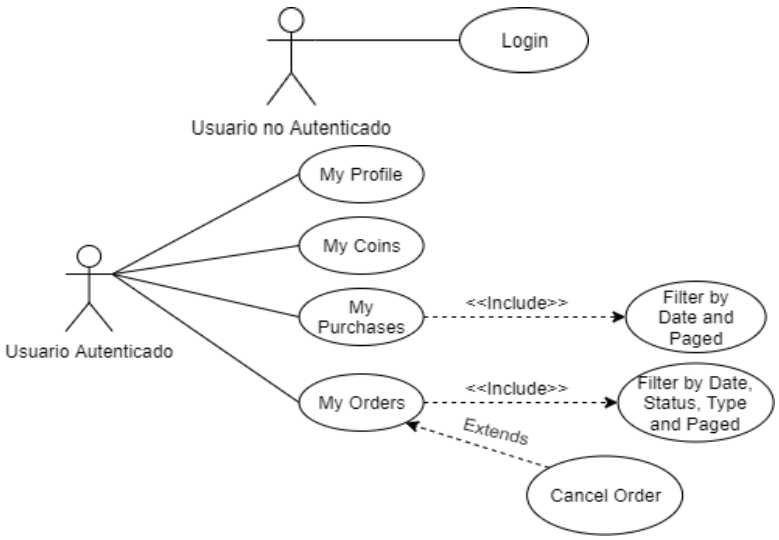

Figura 4.3: Casos de uso de la historia User System.

- **US.01 - Login:** Un usuario puede logearse en la aplicación a través de la introducción de su email y su contraseña. Devolviendo los detalles del perfil del usuario. **Aspectos:**

    1. *El usuario no verá sus puntos de ranking, sino la posición global que ocupa.*
    2. *Ademas de todos sus atributos, tiene que ver el número de comentarios en el forum system que ha realizado.*
    3. *Se tiene que llamar a la Api de Mojang a través del link para complementar la información del usuario.*
    4. *No se mostrará el número asociado al nivel de vip, sino el título asociado a ese nivel. Por defecto los títulos son: vip0, vip1… .*

- **US.02 - Search Coins:** Un usuario logeado puede buscar las monedas que posee dentro de la aplicación. **Aspectos:**

    1. *Las monedas no se pueden representar de la misma forma que estén guardadas, es decir, si tengo 456 monedas normales tienen que ser representadas como 4 monedas de plata y 56 de cobre.*
    2. *Las monedas premium se representarán de la misma que forma que estén guardadas.*

- **US.03 - Search Purchases:** Un usuario logeado puede buscar las compras que ha realizado en la aplicación, siempre paginadas y ordenadas por fecha.

- **US.04 - Search Orders:** Un usuario logeado puede buscar las órdenes que ha realizado a lo largo del tiempo en el Auction System, pero estas siempre tendrán que estar paginadas, ordenadas primero por el estado en el que se encuentran, siendo representadas las activas primero, ordenadas por el tipo de orden y finalmente por la fecha más reciente.

- **US.05 - Cancel Orders:** Un usuario logeado puede cancelar las ordenes que ha creado, aplicándose las siguientes condiciones:

    1. *La orden tiene que estar en un estado de activa.*
    2. *Si se realiza con éxito se aplica una penalización sobre las monedas normales del 5%, ajustable en los ficheros de configuración.*

4.3.2.2 Historias de usuario. Forum System.

Como usuario de la aplicación quiero poder manipular elementos de un foro para poder comunicar mis opiniones de cierto tema a los demás usuarios.

Esta historia de usuario hace referencia a las funciones que dispone el usuario para poder interactuar con otros a través del sistema del foro.

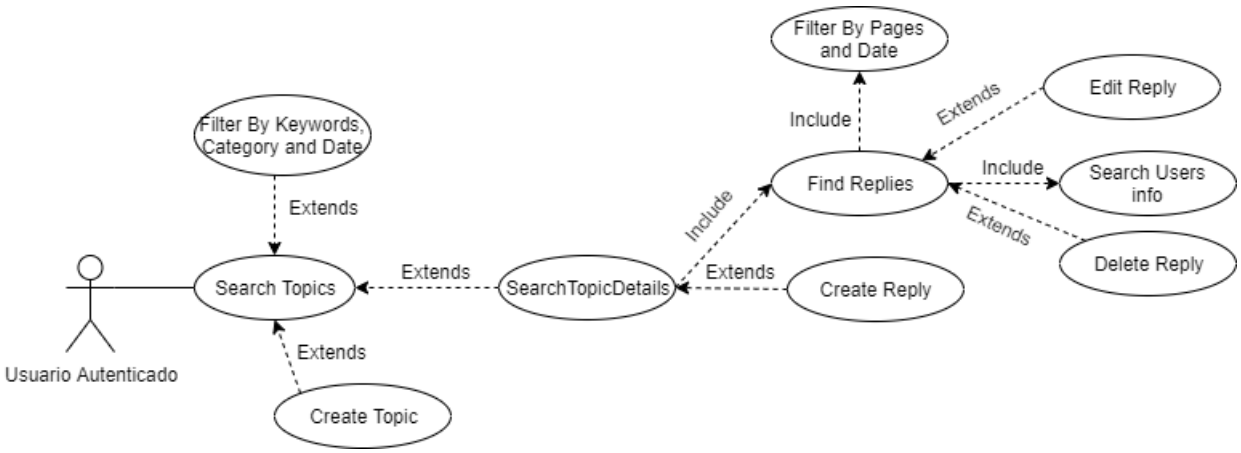

Figura 4.4: Casos de uso de la historia Forum System.

- **FS.01 - Search Topics:** Un usuario desea poder buscar los diferentes topics que se encuentran en la aplicación. Estos siempre se devolverán paginados debido a la posible gran cantidad de topic creados a lo largo del tiempo, ordenados por la fecha de publicación.
  Adicionalmente el usuario podrá buscar con parámetros adicionales, tales como la categoría o palabras clave que estén contenidas en la cabecera del topic. Estos parámetros son opcionales siendo libre de elegir si emplea ambos, ninguno o solo uno de ellos.

- **FS.02 - Create Topic:** Un usuario desea crear un topic en el foro, introduciendo los campos necesarios para su creación.

- **FS.03 - Search Topic Details:** Si el usuario desea más información acerca de un topic en concreto, puede buscar sus detalles, los cuales desplegarán dos secciones:

  1. *El contenido del topic.*
  2. *Las respuestas que presenta cada topic con la información respectiva al usuario que ha creado cada respuesta, mostrándolas paginadas.*
     Adicionalmente se realiza la ordenación por fecha de edición o fecha de creación de las respuestas, tomando como referencia la más reciente de las dos.
     (Esta última operación es realmente complicada).

- **FS.04 - Delete Reply:** Una vez que el usuario ha listado el conjunto de respuestas asociadas a un topic, este puede optar por borrar las respuestas que le pertenezcan, en el caso de que sea administrador o moderador se le permitirá eliminar las de los demás usuarios además de las que ya le pertenezcan.

- **FS.05 - Create Reply:** Al encontrar el contenido del topic, el usuario podrá dejar la respuesta que estime oportuna asociándola a ese topic.

- **FS.06 - Edit Reply:** Al encontrar una respuesta que le corresponda al propio usuario, este podrá modificarla, y debido a los criterios de ordenación de las respuesta, la nueva respuesta modificada pasará al primer lugar en la lista paginada de respuestas.

4.3.2.3 Historias de usuario. Shopping System.

Como usuario de la aplicación quiero poder realizar compras y administrar una tienda online para poder obtener beneficios de ella.

Esta historia de usuario hace referencia a las funciones que dispone el usuario para realizar las compras que desee dentro del sistema y realizar la gestión de las mismas.

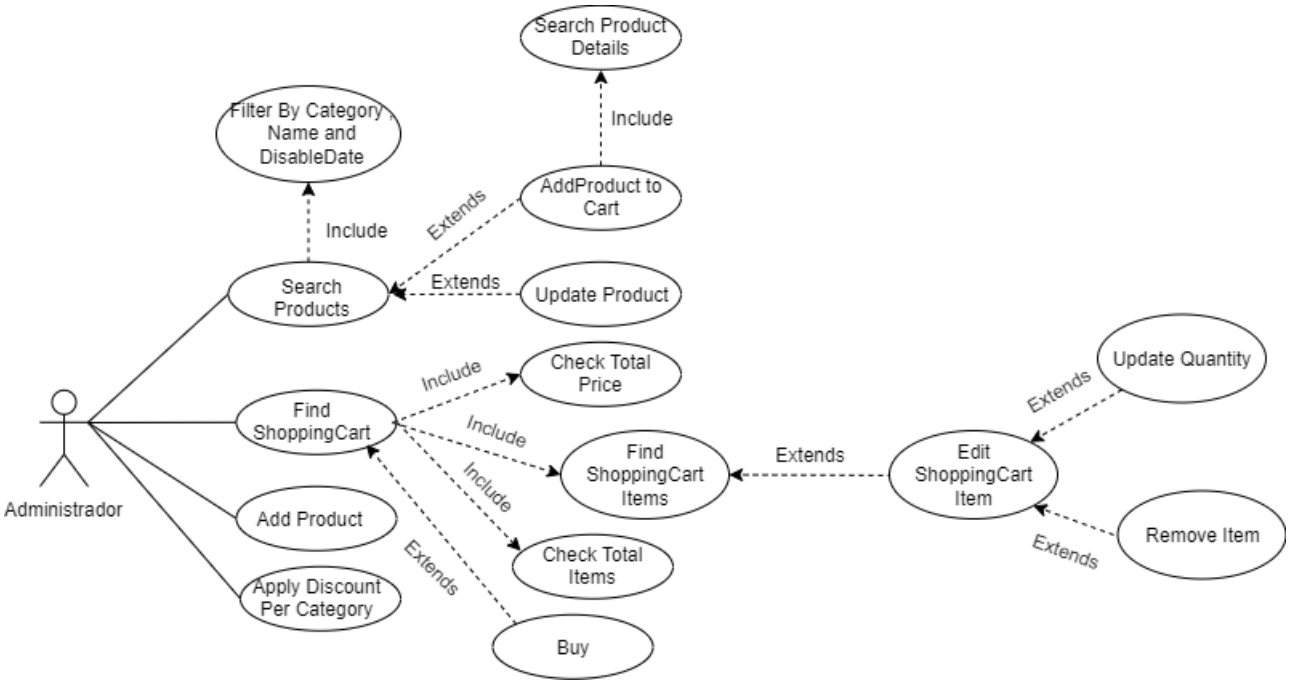

Figura 4.5: Casos de uso de la historia Shopping System.

- **SS.01 - Add Product:** Un administrador desea incorporar al sistema un determinado producto, para ello introduce los campos necesarios, con los siguientes aspectos:

    1. *El producto tiene que aplicar el descuento antes de ser guardado.*
    2. *El usuario solo puede introducir un tipo de precio (en euros o en monedas de platino) por cada producto.*

- **SS.02 - Apply Discount per Category**: Un administrador desea cambiar los descuentos de todos los productos de una categoría (como máximo a 90%), para ello señala la categoría a la que quiere aplicarlo y el nuevo descuento.

- **SS.03 - Search Products:** Un usuario desea buscar todos los productos asociados a una categoría, filtrando por la fecha de des-habilitación (estos no se mostrarán) y ordenados por el nombre del producto.

- **SS.04 - Update Product:** Un administrador al ver la lista de productos por una categoría puede seleccionar un determinado producto para cambiar todos los atributos que posea, una vez más no se pueden introducir el valor de monedas de platino y euros al mismo tiempo.

- **SS.05 - Add Product To Cart:** Un usuario al interesarse por un producto, podrá incorporarlo a su carrito para su posterior compra, esto implica ver los detalles del producto a mayor profundidad (el usuario podrá leer la descripción del producto antes de introducirlo a su carrito). Por defecto se pueden incorporar un máximo de 20 productos al carrito (no repetidos, y ajustable en los ficheros de configuración).

- **SS.06 - Find ShoppingCart:** Un usuario podrá consultar los detalles de su carrito, y ver la evolución de este a lo largo de diferentes operaciones.

- **SS.07 - Check Total Price:** Un usuario al contemplar los detalles de su carrito podrá ver el precio total que va acumulando según los productos y la cantidad de los mismos que va incorporando, este precio está compuesto en euros y monedas de platino.

- **SS.08 - Check Total Items:** Un usuario al contemplar los detalles de su carrito podrá ver el número exacto de productos que tiene en su carrito.

- **SS.09 - Find ShoppingCart Items:** Un usuario al acceder a los detalles de su carrito, se desplegarán los items que tiene almacenados en su carrito, cada uno con su respectivo precio.

- **SS.10 - Edit ShoppingCart Item:** Un usuario al contemplar todos los items de su carrito, podrá modificarlos si lo desea, ya sea a través de eliminar tal elemento o añadirle más cantidad, con una cantidad limitada a 40 (editable en ficheros de configuración), siempre y cuando el producto permita su compra en más de una unidad.

- **SS.11 - Buy:** Finalmente el usuario al ver toda la información de su carrito puede decidir comprar todos los productos que ha ido incorporando, siempre y cuando tenga suficientes monedas de platino para efectuar el pago.

4.3.2.4 Historias de usuario. Auction System.

Como usuario de la aplicación quiero poder realizar subastas de las monedas de platino que poseo y tener la información necesaria para realizar estas operaciones, para poder tener la posibilidad de adquirir monedas de platino con mi desarrollo en el juego.

Esta historia de usuario hace referencia a las funciones que dispone el usuario para realizar diferentes intercambios de objetos con otros usuarios, los objetos abordados en este sistema serán monedas normales y de platino.

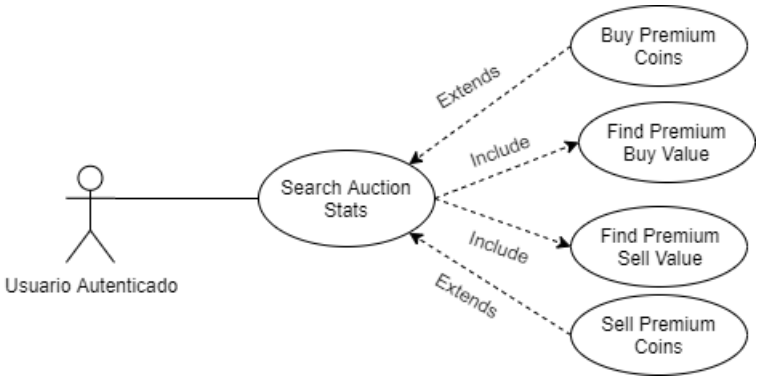

Figura 4.6: Casos de uso de la historia Auction System.

- **AS.01 - Search AuctionStats:** Los usuarios podrán conocer la evolución del precio de las monedas de platino a lo largo del paso de los días, mostrando el valor medio en monedas normales por cada unidad de platino.

- **AS.02 - Find Premium Buy Value:** Los usuarios podrán conocer el precio mínimo que tendrían que pagar en monedas normales en un determinado momento para ser capaz de realizar una compra de al menos una moneda de platino instantáneamente.

- **AS.03 - Find Premium Sell Value:** Los usuarios podrán conocer el precio mínimo que tendrían que pagar en monedas normales en un determinado momento para ser capaz de realizar una venta de al menos una moneda de platino instantáneamente.

- **AS.04 - Sell Premium Coins:** Los usuarios podrán vender sus monedas de platino a cambio de monedas normales. Presentando las siguientes implicaciones:

    1. *Se cobrará una tasa sobre las monedas normales que se consigan a cambio de las monedas de platino vendidas. Esta tasa se decide con el nivel del vip del usuario que pone la orden, siendo menor cuanto mayor nivel de vip se tenga, [10% → 4%] ajustable en el fichero de configuración.*

2. *Solo se podrán realizar transacciones con las órdenes de compra de otros usuarios, completando siempre primero las órdenes de compra de los usuarios con mayor nivel de vip y activas.*
3. *Será posible conseguir más monedas de las del precio medio que he solicitado siempre y cuando existan usuarios que soliciten comprarlas a mayor precio medio.*
4. *Si se produce una transacción, se generan estadísticas, guardándolas en la media de las AuctionStats presentes en el mismo día, es decir, toda transacción para guardar sus estadísticas se engancharán a la del día en el que ocurran.*
5. *El número de ordenes de venta activas al mismo tiempo se verá determinado por el nivel vip del usuario. Ajustable en los ficheros de configuración (Se cuentan también las activas del tipo Compra). [5 → 12 máximas].*
6. *La fecha de expiración de la orden de venta también vendrá determinada según el nivel de vip del usuario. [1 → 6 días a partir del momento de creación de la orden] .*

- **AS.05 - Buy Premium Coins:** Los usuarios podrán tener la oportunidad de comprar monedas de platino de otros usuarios a cambio de mis monedas normales. Esto tiene las siguientes implicaciones:

1. *No se puede cobrar tasas en este caso por conseguir monedas de platino.*
2. *Las transacciones con las diferentes ordenes de venta de otros usuarios, se completarán en orden de nivel vip de los usuarios que venden y estas órdenes tienen que estar activas.*
3. *En este caso no puedo conseguir más monedas de las solicitadas.*
4. *Si se produce una transacción, esta se engancha a las AuctionStats presentes en el mismo día .*
5. *Según el nivel de vip solo se pueden tener un cierto número de órdenes de compra activas al mismo tiempo. Ajustable en los ficheros de configuración. (Se cuentan también las activas del tipo Venta). [5 → 12 máximas).*
6. *La fecha de expiración de la orden de compra también vendrá determinada según el nivel de vip del usuario. (1 → 6 días a partir del momento de creación de la orden).*

Capítulo 5

Diseño

En este capítulo se tratarán de las decisiones tomadas en la creación del diseño de la pagina web y la arquitectura que la soporta.

5.1 Maquetado.

Se han creado con ayuda de la herramienta Balsamiq descrita en el capítulo 2, maquetas que representan la interfaz del usuario. El objetivo de estas maquetas es el diseño de una interfaz sencilla y clara siguiendo las pautas de *User Experience* y *User Interface*.

User Experience (UX) resume la experiencia que ocurre en el usuario al interactuar con la interfaz de la aplicación. *User Interface* (UI) hace referencia a todo lo relacionado al aspecto que presenta la interfaz de la aplicación con la que van a interactuar el usuario.

La maquetación fue diseñada en los momentos iniciales del proyecto, puesto que la metodología de estimación IFPUG descrita en el capítulo 2, requiere la creación de estas para estimar el proyecto en la fase inicial, además proporcionan una visión aún más clara a los integrantes del equipo que realizará el proyecto de las funcionalidades que tendrán que desarrollar.

Las maquetas se agruparán según los sistemas que van a ser implementados por este proyecto. (El diseño de la página principal irá con el User System).

5.1.1 Maquetado. User System.

Todos los usuarios al iniciar en la aplicación verán la interfaz inicial de bienvenida (Figura 5.1), cuyo propósito será proveer a los usuarios un punto de partida para acceder al resto de funcionalidades de la aplicación.

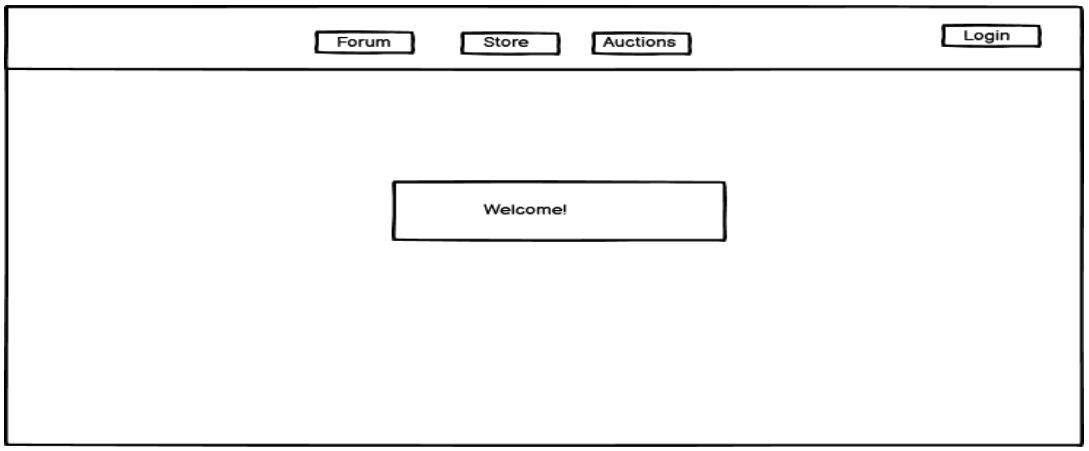

Figura 5.1: Maqueta de bienvenida.

A partir de la cual se le da al usuario la capacidad de autenticarse.

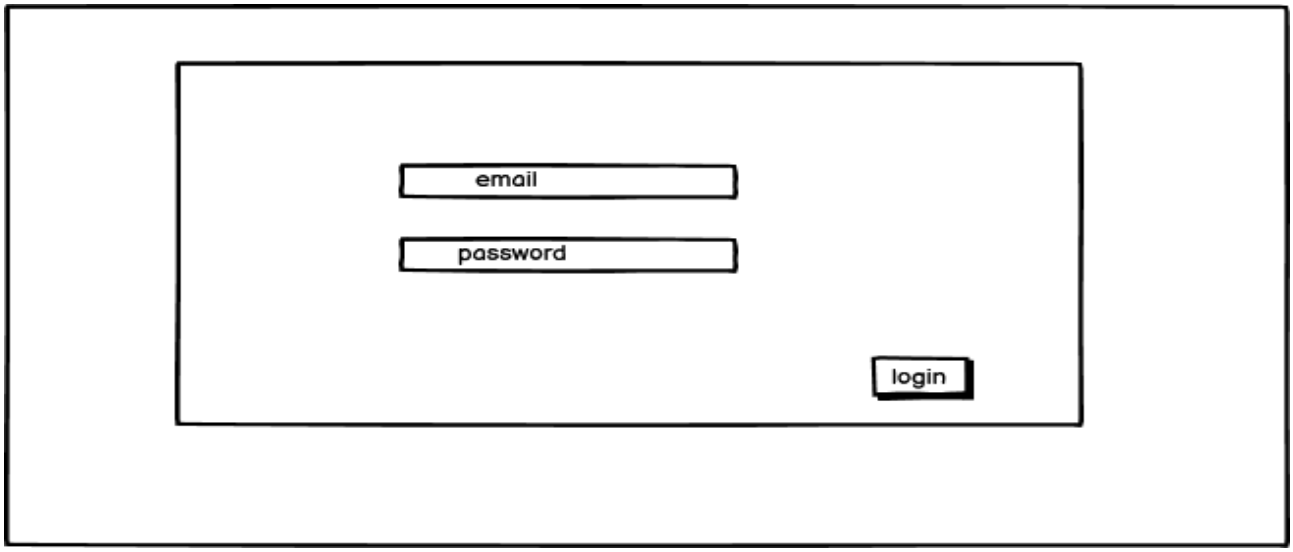

Figura 5.2: Maqueta de Login.

Una vez que el usuario realice su correcta autentificación, podrá acceder a las diferentes funcionalidades que le permiten la gestión de su información, tales como ver sus propias órdenes de subastas creadas, las compras que ha realizado a lo largo del tiempo, las monedas actuales que posee, y consultar su perfil.

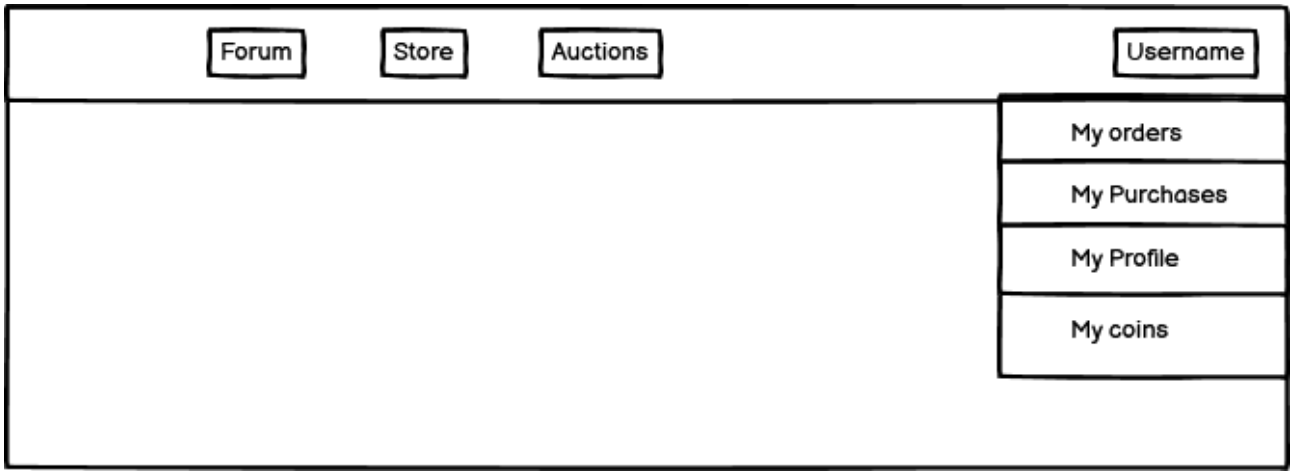

Figura 5.3: Maqueta de bienvenida logeado.

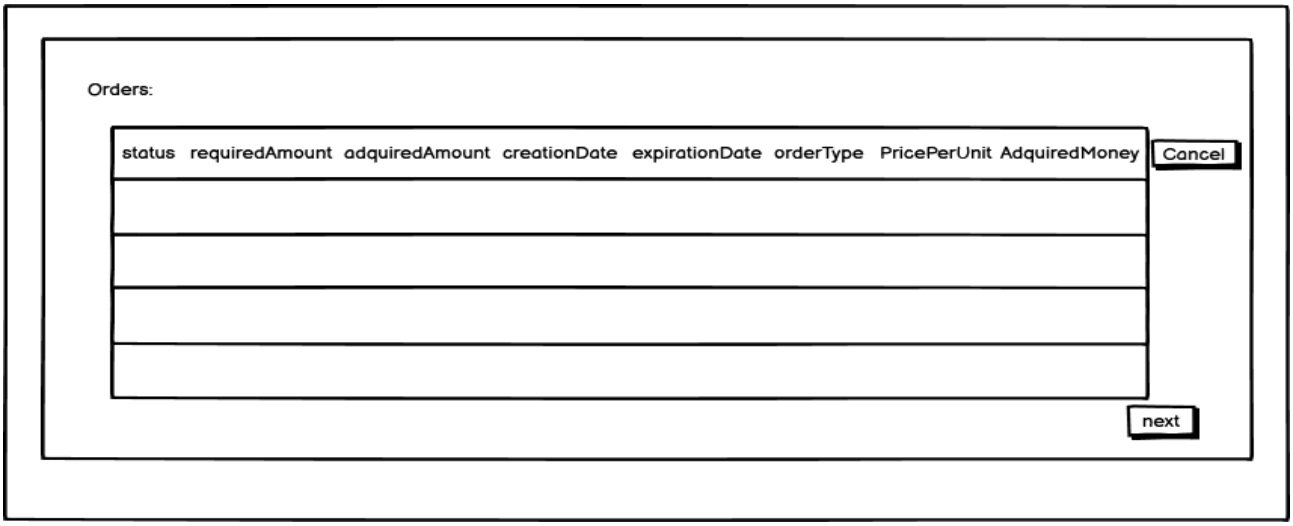

Figura 5.4: Maqueta de ordenes del usuario.

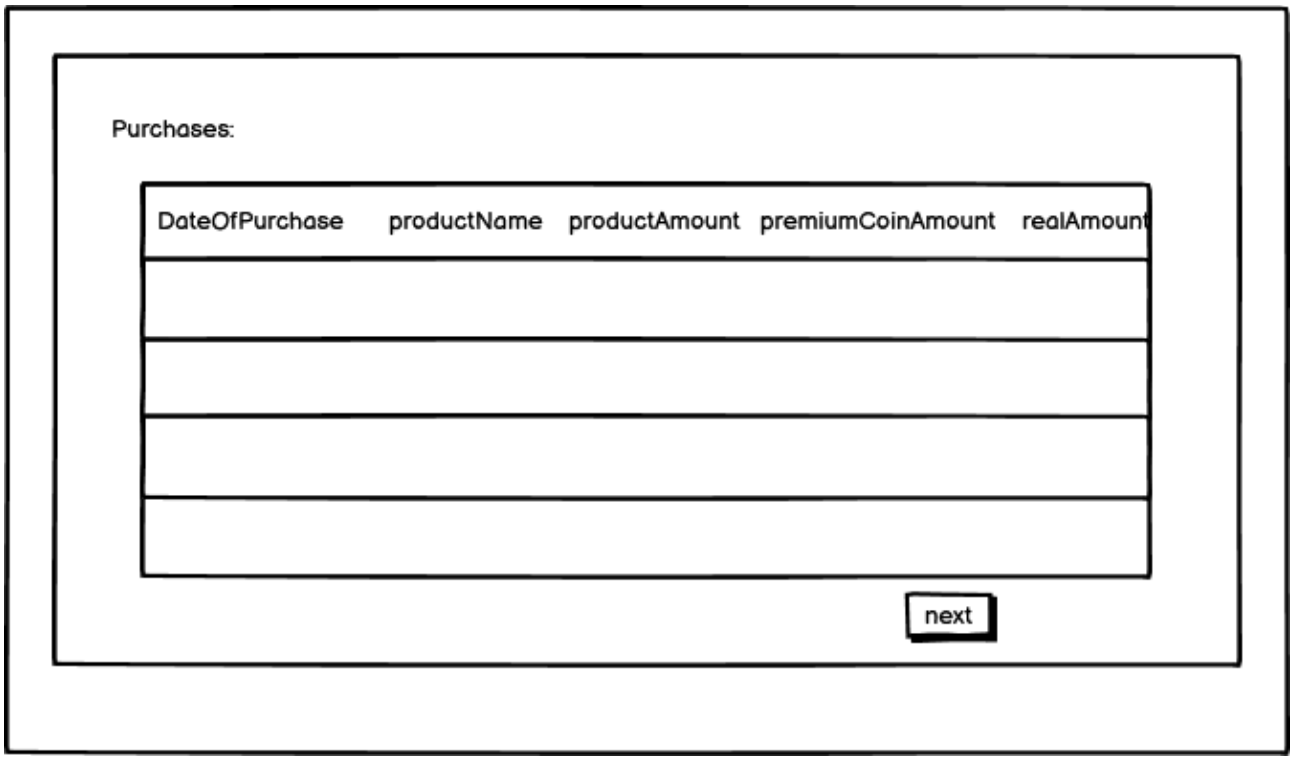

Figura 5.5: Maqueta de compras del usuario.

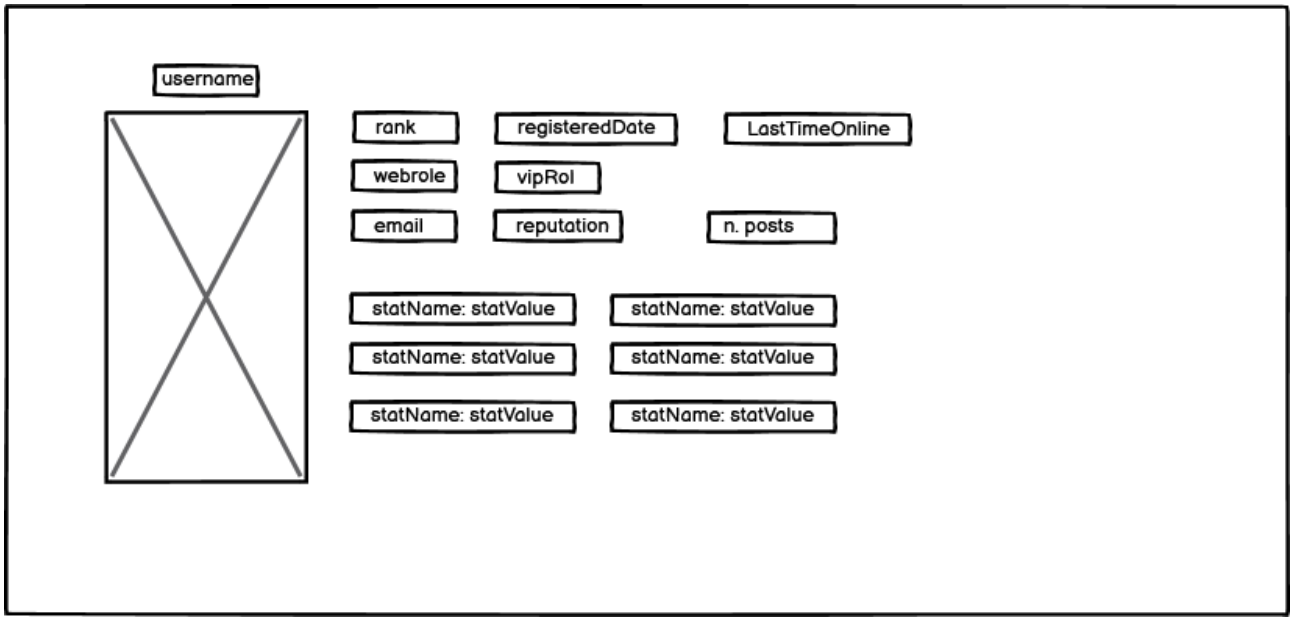

Figura 5.6: Maqueta del perfil del usuario.

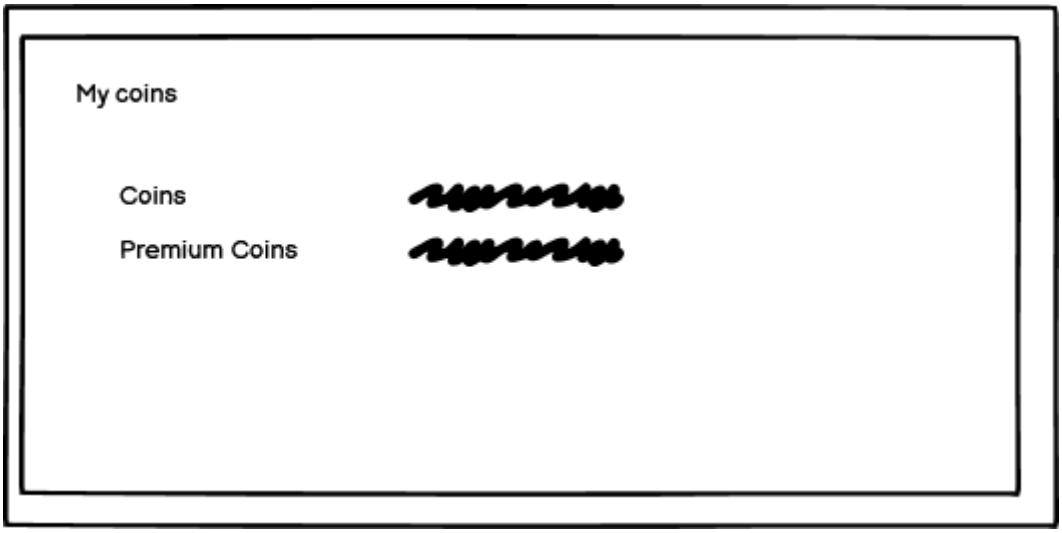

Figura 5.7: Maqueta de las monedas del usuario.

5.1.2 Maquetado. Forum System.

Un usuario puede pulsar en el enlace a foro en la página principal para acceder a este sistema de la aplicación, los botones de New Topic, Edit Reply, Create Reply, Borrar Reply, solo estarán disponibles si el usuario cumple las características establecidas en la especificación de requisitos y esté logeado.

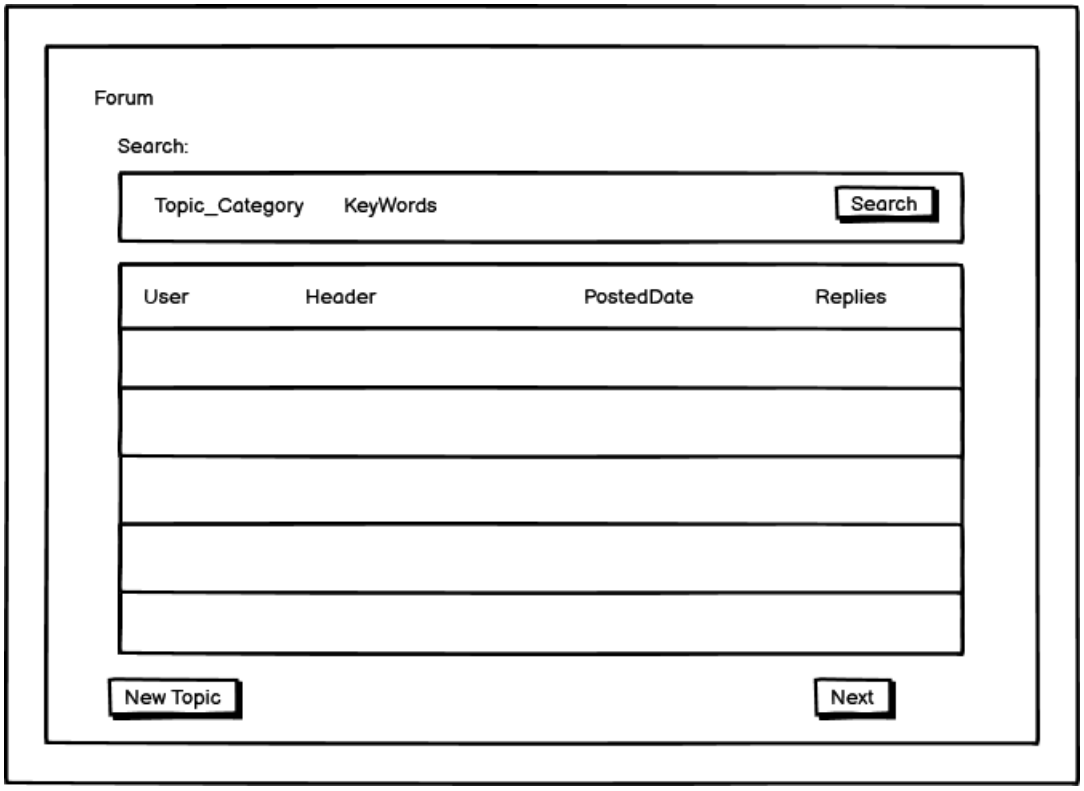

Figura 5.8: Maqueta de búsqueda en el Foro.

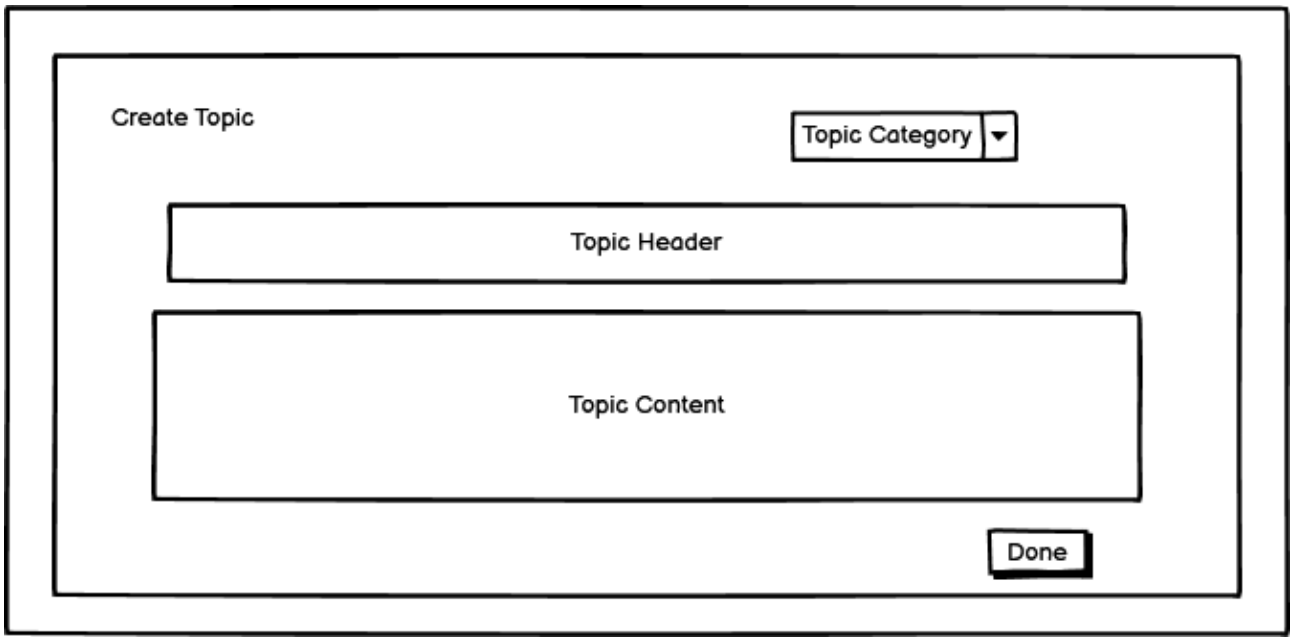

Figura 5.9: Maqueta de creación de un Topic.

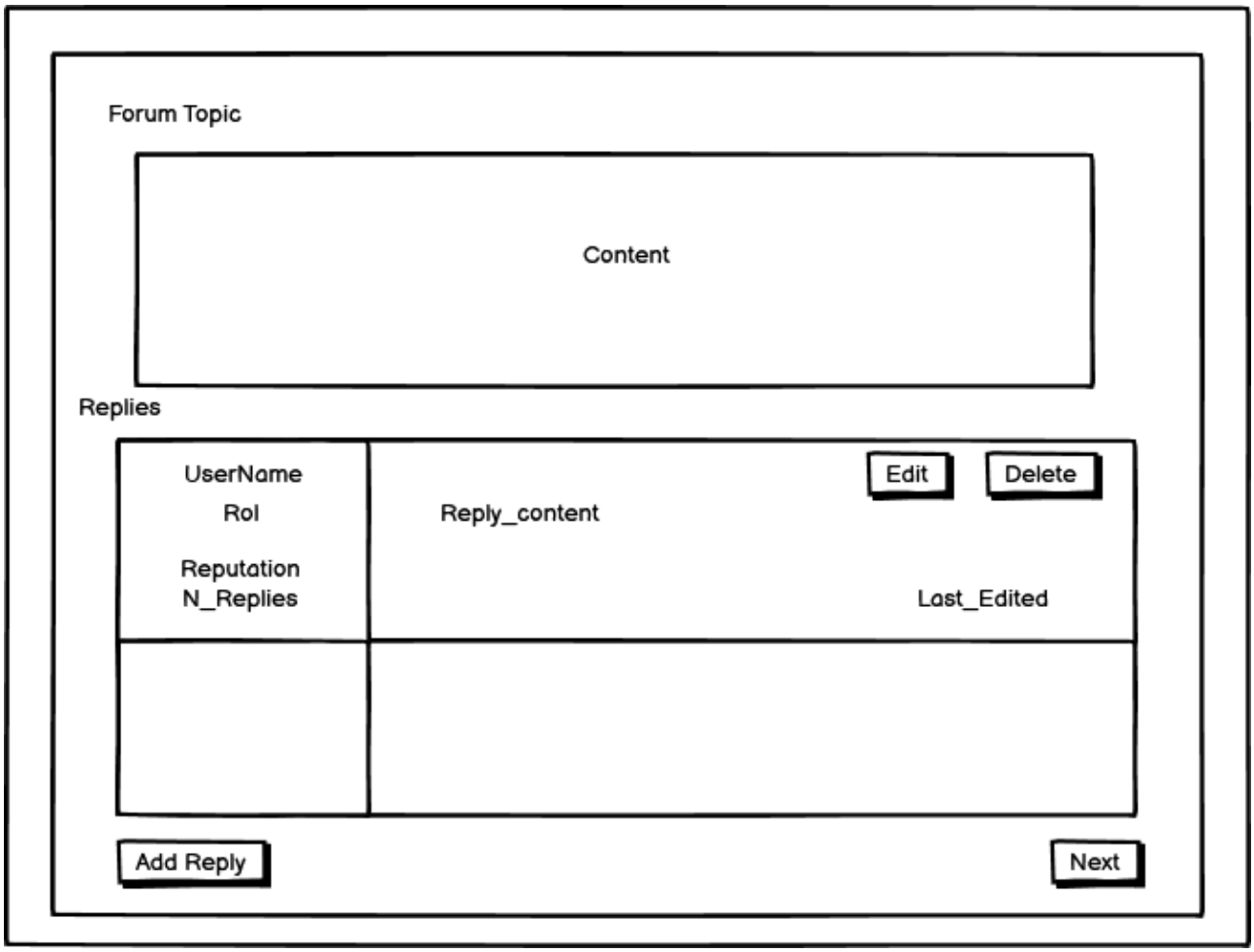

Figura 5.10: Maqueta de detalles de un Topic.

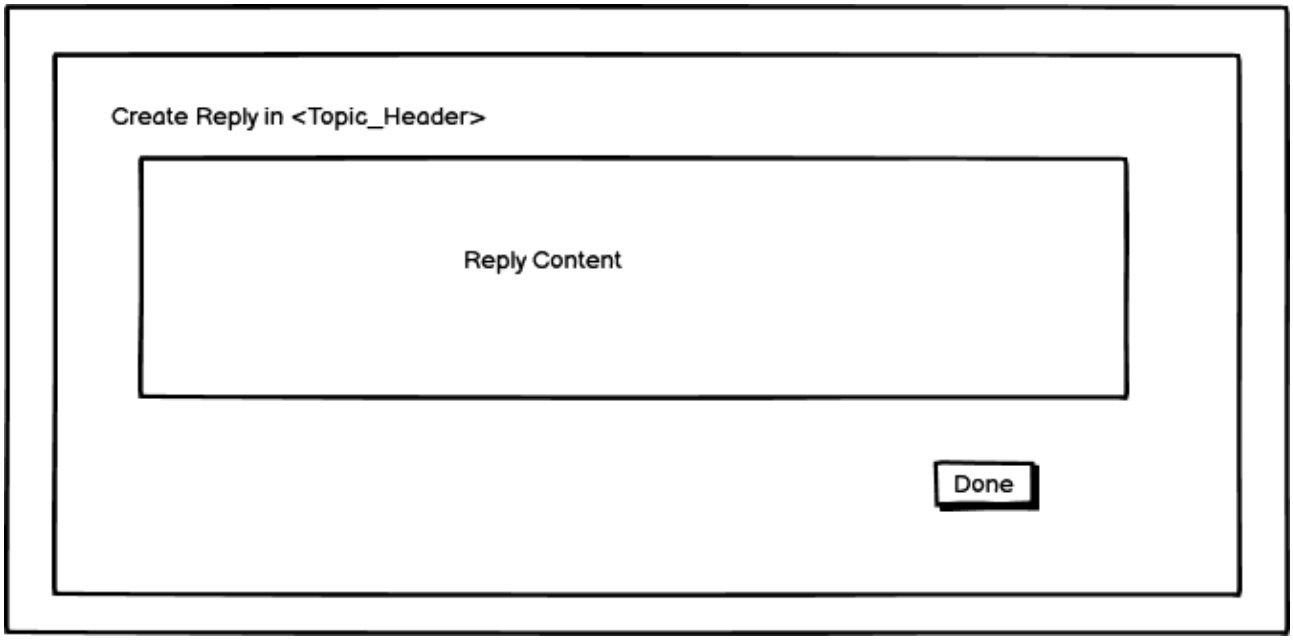

Figura 5.11: Maqueta para crear una Reply en un Topic.

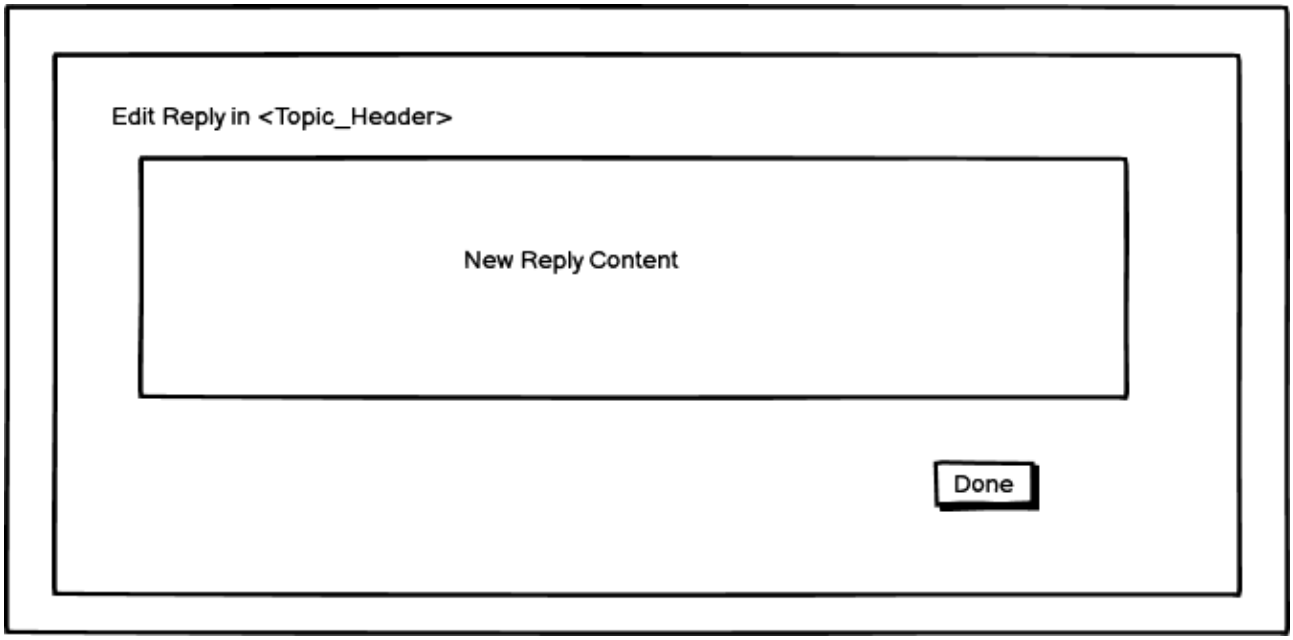

Figura 5.12: Maqueta para editar una Reply en un Topic.

5.1.3 Maquetado. Shopping System.

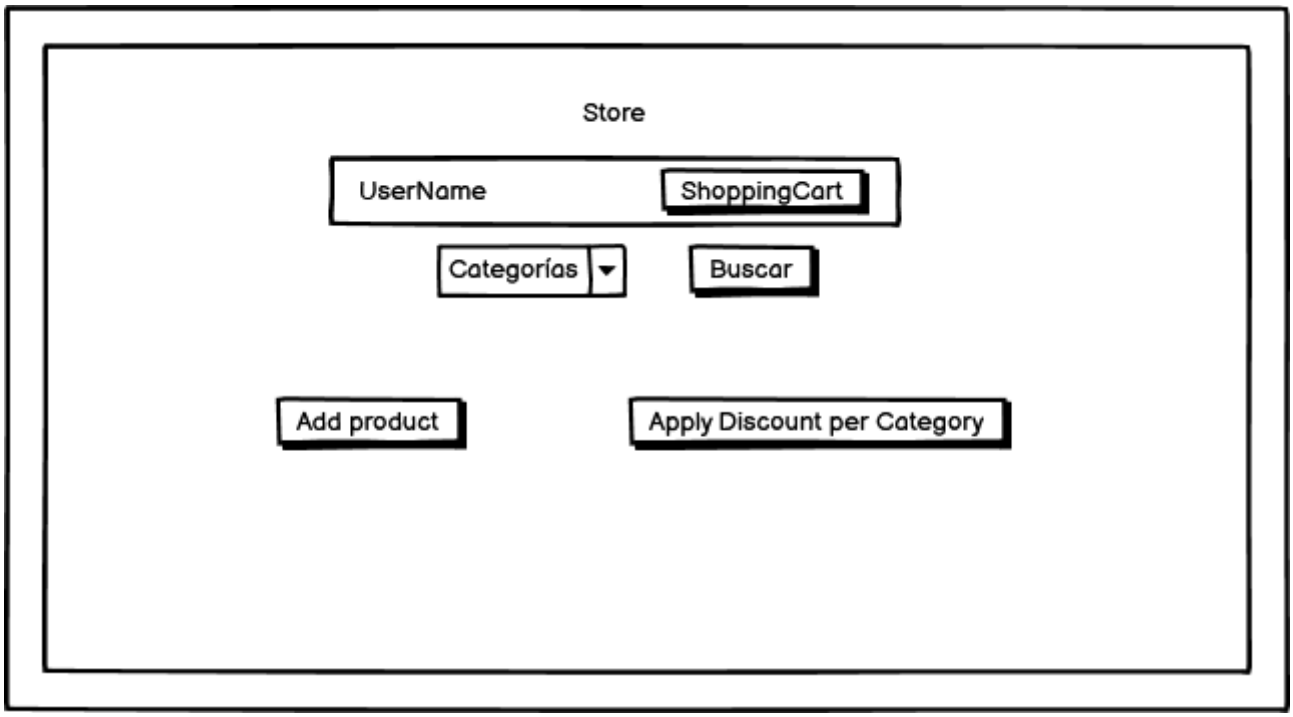

Figura 5.13: Maqueta de la página principal de la tienda.

Los botones de Add Product, Apply Discount per Category, y la barra del usuario solo aparecerán si el usuario cumple con los criterios necesarios de la especificación de requisitos y además esté logeado.

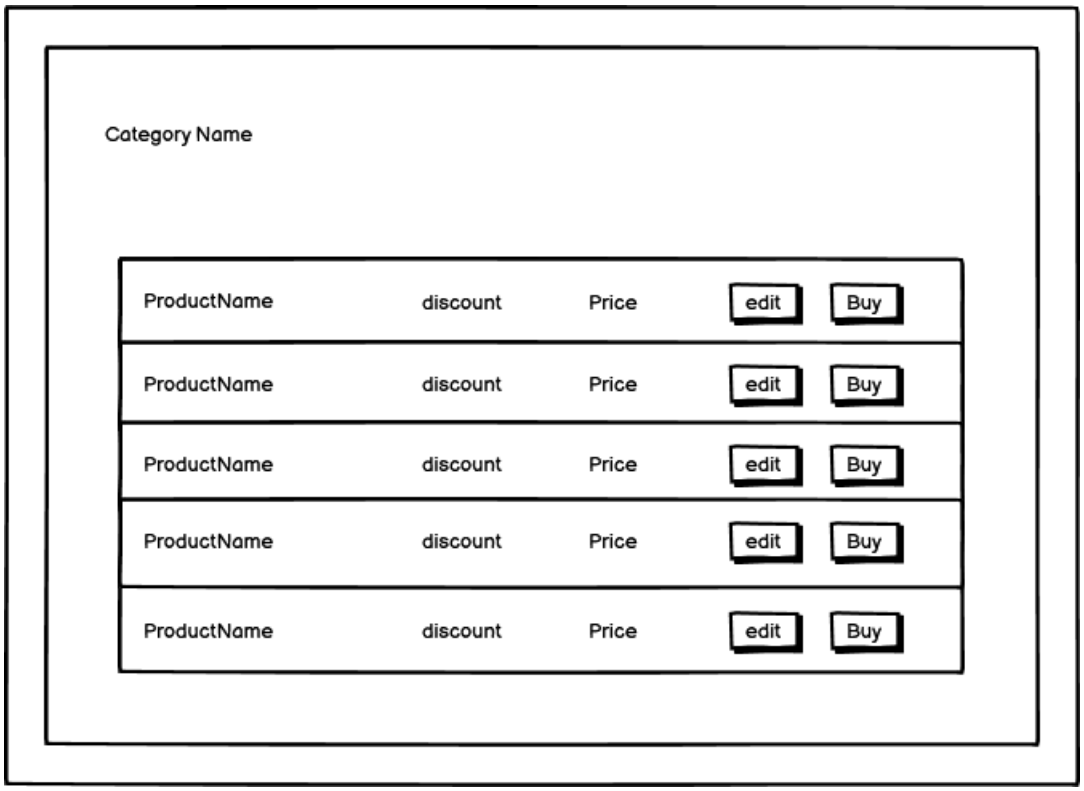

Figura 5.14: Maqueta de búsqueda de productos por categoría.

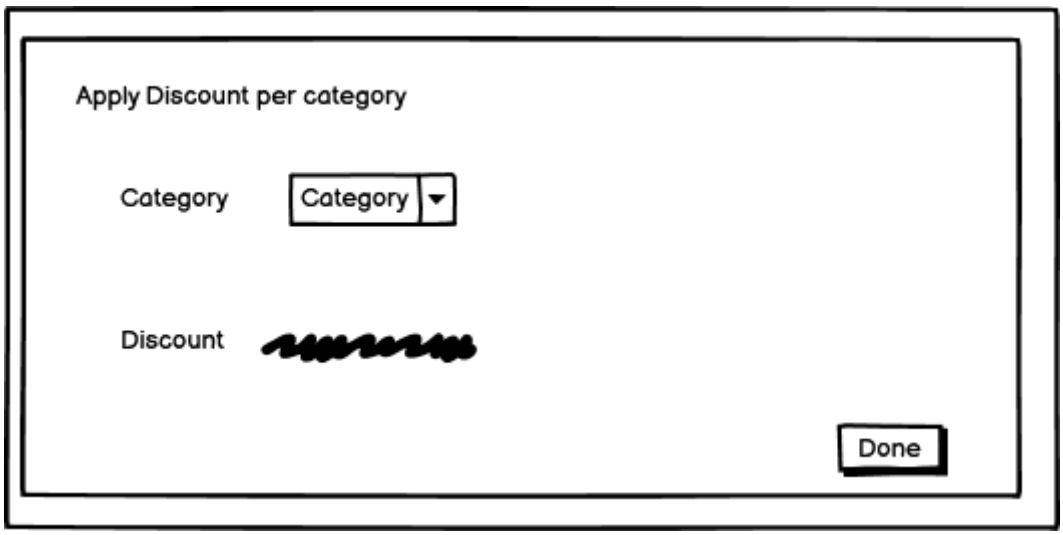

Figura 5.15: Maqueta para aplicar un descuento por categoría.

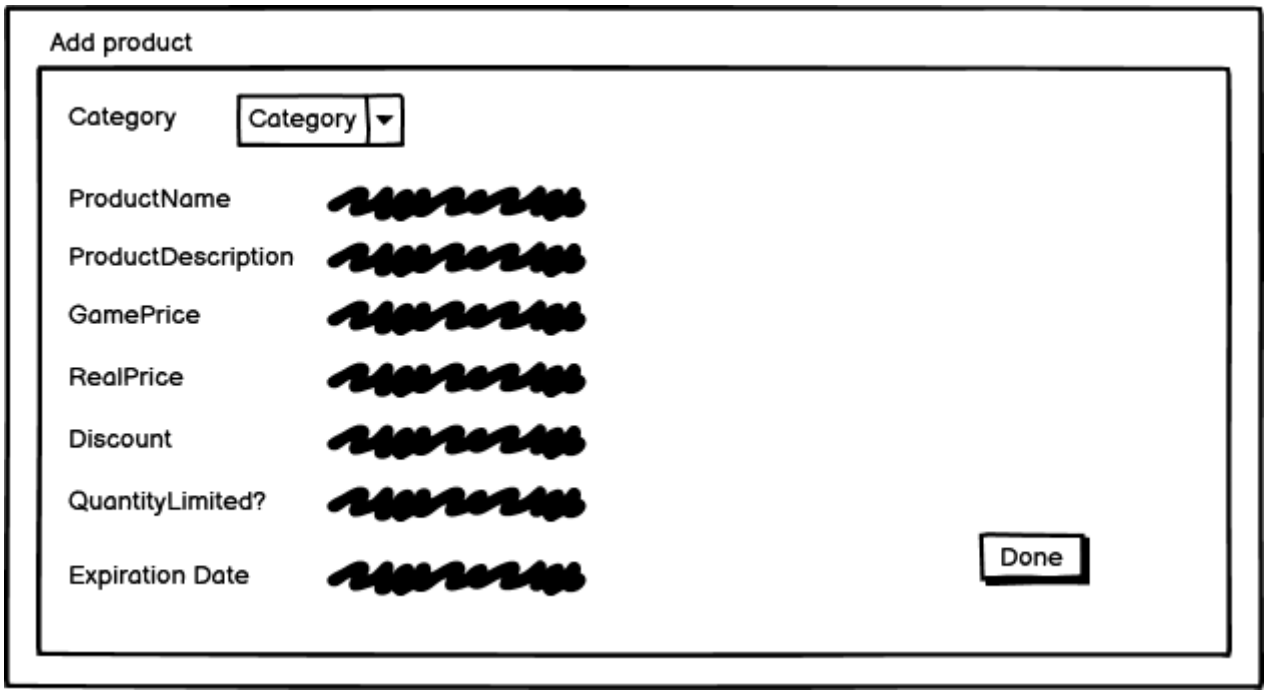

Figura 5.16: Maqueta para añadir un producto a la Tienda.

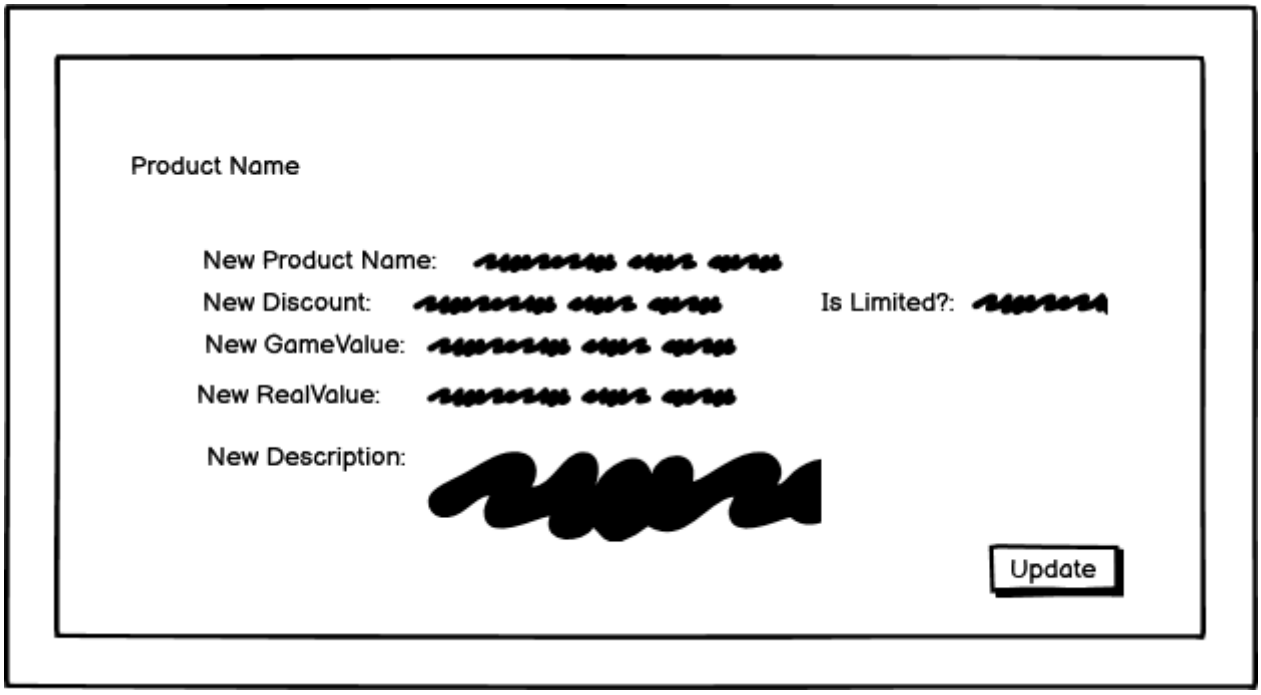

Figura 5.17: Maqueta para actualizar un producto en la tienda.

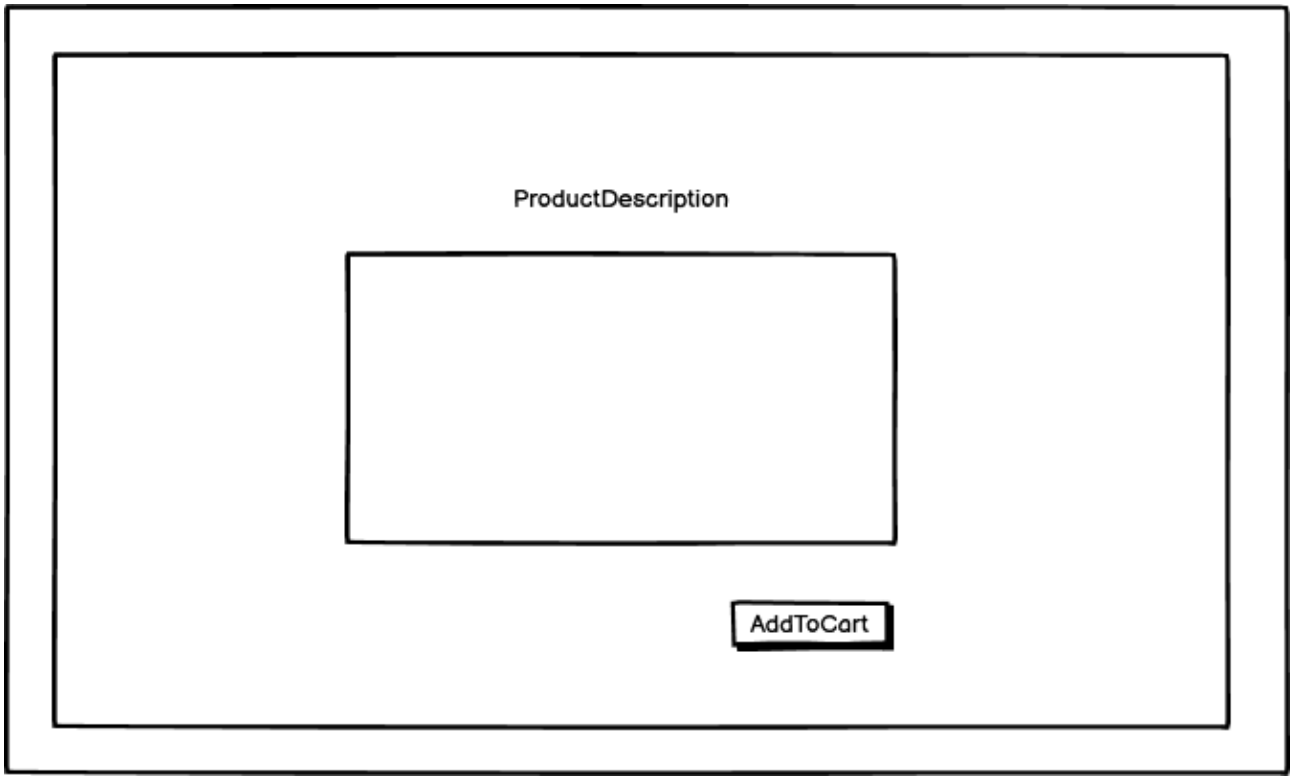

Figura 5.18: Maqueta para poder añadir un producto al carrito.

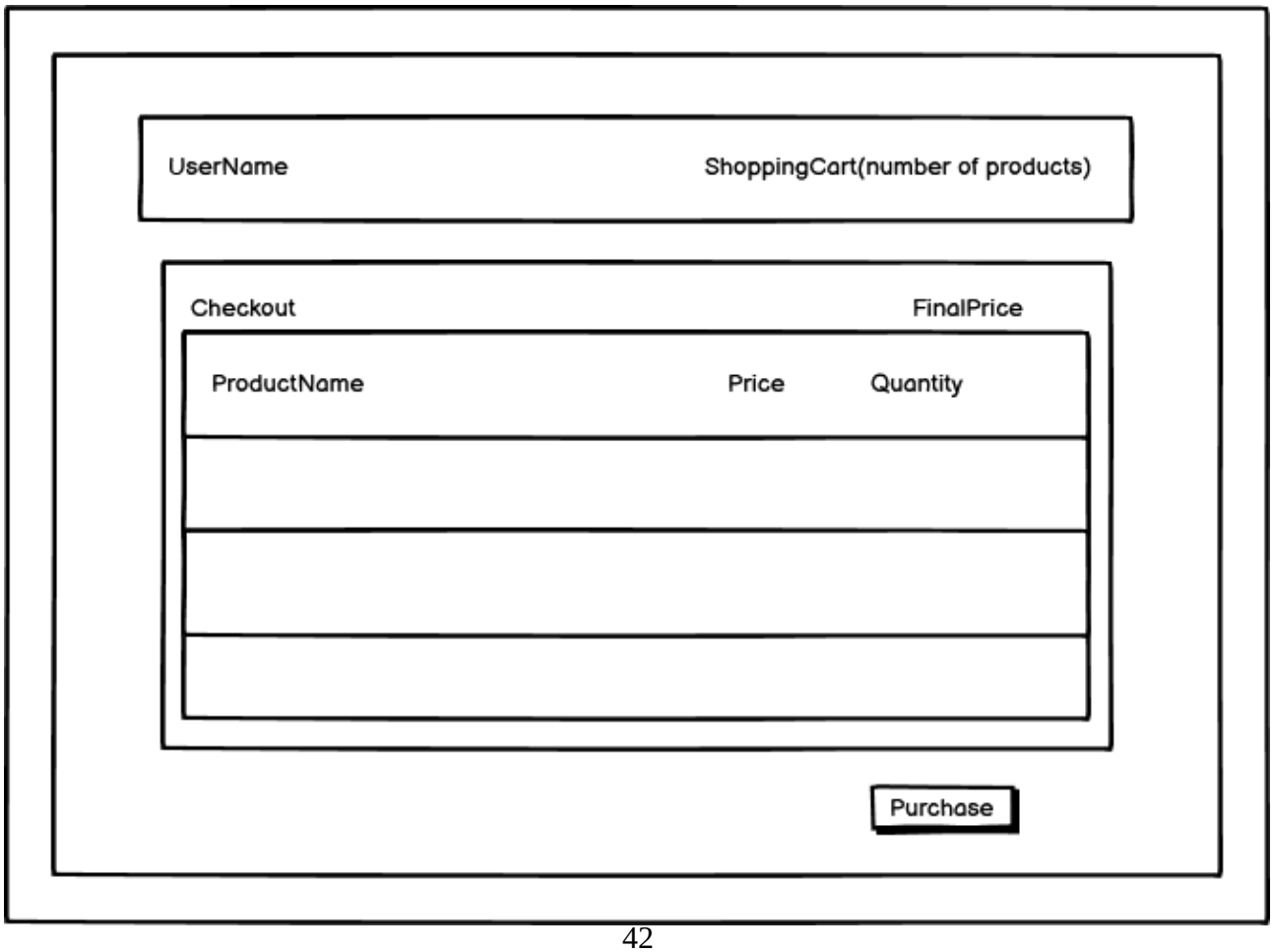

Figura 5.19: Maqueta para poder visualizar todos los detalles del carrito.

5.1.4 Maquetado. Auction System.

Todas las funcionalidades del auction system se encuentran englobadas en una única maqueta puesto que la información que ofrecen al usuario las diferentes funcionalidades de este sistema es necesaria para realizar operaciones tales como compra y venta de platino.

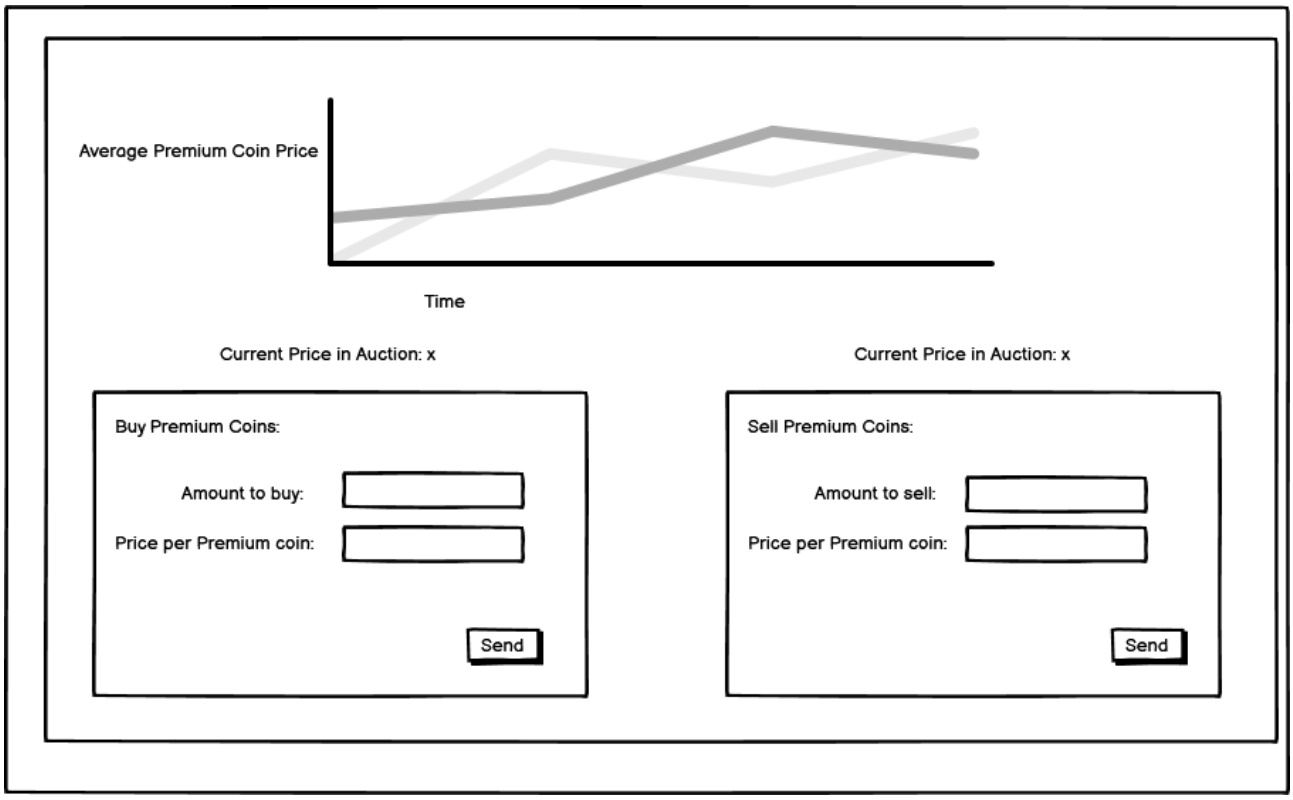

Figura 5.20: Maqueta del sistema de subastas.

5.2 Arquitectura del Proyecto.

A continuación se muestra un diagrama de la arquitectura que tiene este proyecto (Figura 5.20).

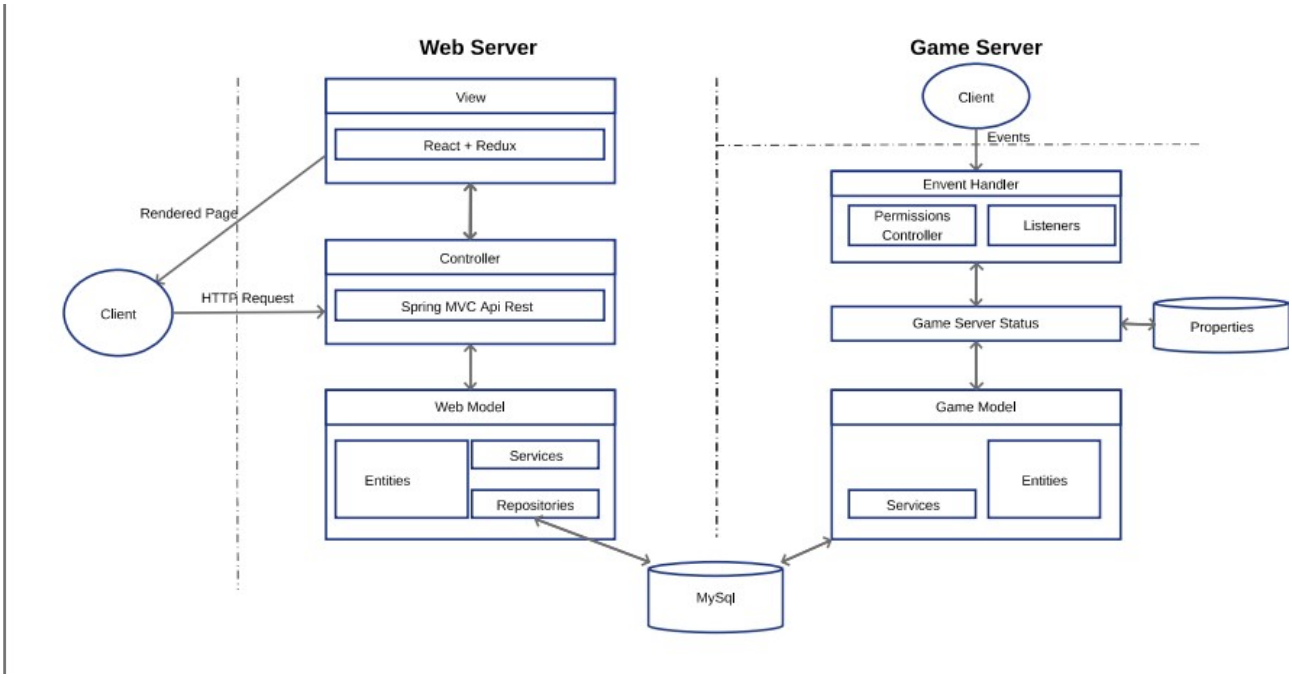

Figura 5.21: Diagrama del sistema.

Esta aplicación presenta dos arquitecturas diferentes, una arquitectura MVC para el lado del servidor de la página web, y una arquitectura nueva para el lado del servidor del juego.

5.2.1 Arquitectura web.

En la capa del servidor web se ha aplicado una arquitectura MVC, donde la capa Controlador realiza el manejo de todas las peticiones realizadas a la aplicación, a su vez que se coordina con la capa Modelo para la preparación de cualquier tipo de información que requiera la capa Vista, quien convertirá a su vez la información recibida a un formato legible por el usuario final.

En los siguientes puntos se detalla con más detenimientos la implementación y las metodologías aplicadas en la creación de estas tres capas.

5.2.2 Arquitectura del Servidor de Minecraft.

En este servidor donde se implementa la capa del juego se ha tenido que diseñar una nueva arquitectura. Los servidores de Minecraft no poseen una arquitectura por defecto o unos patrones de buenas prácticas a seguir, por ello con los conocimientos del diseño de software adquiridos durante

la carrera y un estudio del proceso de ejecución de estos servidores, se ha conseguido el diseño ilustrado en el diagrama (Figura 5.20), explicado a continuación:

1. *Para la gestión de datos recibidos por la capa web del otro servidor se han tomado aspectos similares al patrón MVC en cuanto a la capa Modelo, diseñándola con java a bajo nivel, puesto que los servidores de Minecraft no aceptan JPA o Hibernate.*

2. *Como los datos del usuario son datos compuestos, es decir, la información del usuario es la información que se almacena en la capa web añadiéndole la información del usuario almacenada en una base de datos no relacional (organizada en ficheros Properties), creada de forma automática por Minecraft. Se ha diseñado una capa Game Server Status que se encargue de realizar la composición de esta información (en un PlayerCache), manteniendo-la en memoria, y ofreciendo una serie de operaciones para alterar el estado del servidor de Minecraft, ya sea a través de funciones que modifican el comportamiento interno del estado (como actualizar el estado, encontrar usuarios en el estado e.t.c) o llamadas al servicio del modelo.*

   Un Player Cache es el resultado de la composición de los datos del jugador, siendo un elemento de dos campos, los datos de la base de datos compartida (MySql), y los datos proporcionados por Minecraft (Properties).

3. *En Minecraft de forma genérica el jugador puede interactuar con el mundo de dos formas, o bien por comandos, los cuales se consideran eventos, o bien por reacciones del mundo a la existencia del usuario, los cuales se recogen en los listeners (un listener es un thread que está activo escuchando en el servidor). Debido a la existencia de estos dos tipos de procesos se ha creado una capa llamada EventHandler, la cual reacciona a estos. Tiene un servicio de permisos embebido para que pueda reaccionar de forma rápida a las diferentes situaciones que se pueda encontrar, por ejemplo un usuario lanza un comando (es un evento) que no está permitido, el Event Handler mira el servicio embebido de permisos y cancela el evento diciéndole al usuario que no puede llevar a cabo esa acción.*

   Finalmente, para los eventos del listener, no se pueden cancelar (ya que son un suceso interno del mundo del juego), pero se pueden modificar, por ejemplo si un jugador con nivel vip 2 deseamos que el mundo no le pueda dañar el Game Server Status le comunica al listener los valores del usuario y el listener le da la orden al mundo del juego para que considere al jugador como un objeto invulnerable.

5.2.3 Capa Modelo del servidor web.

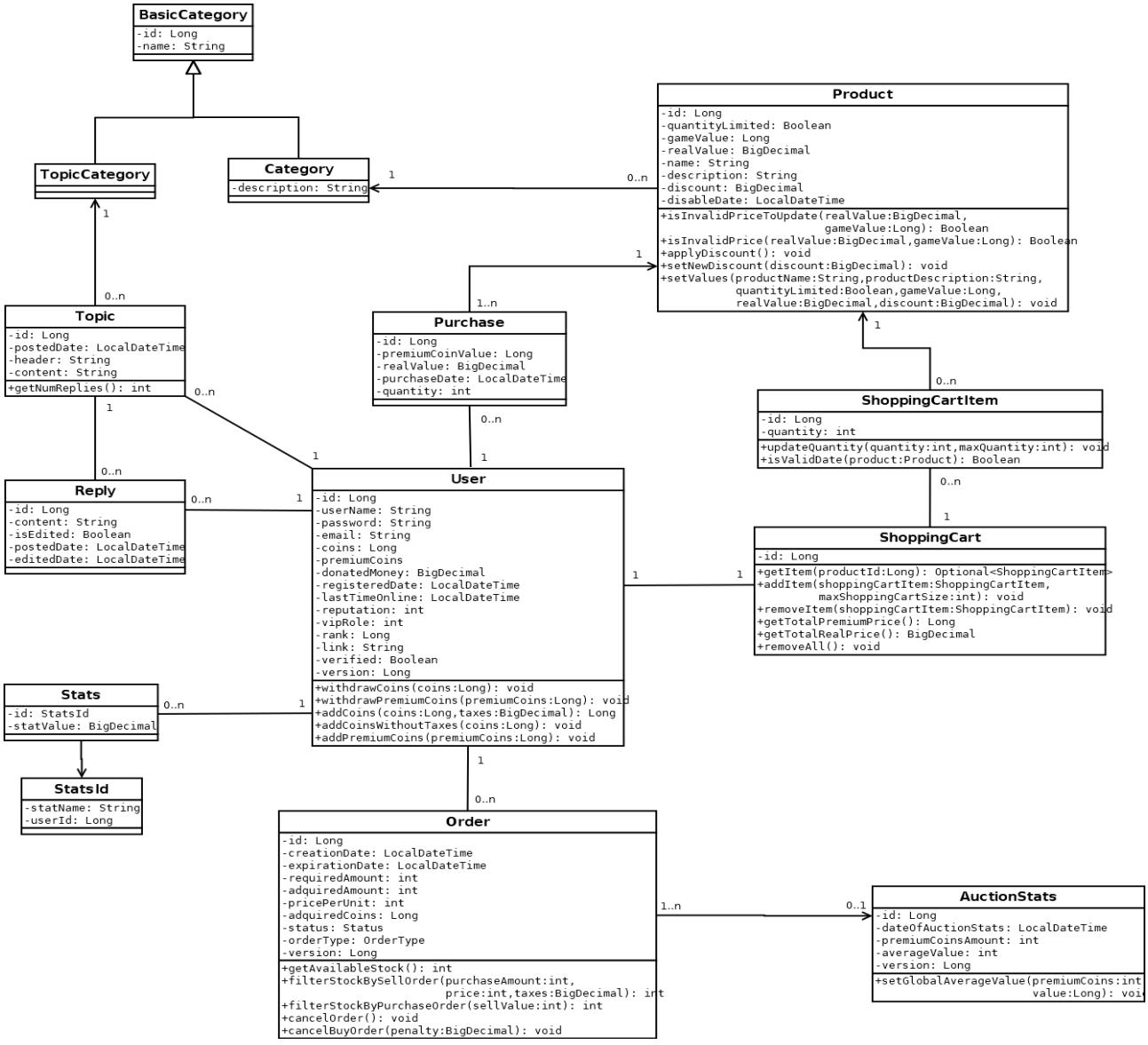

Figura 5.22: Diagrama de entidades persistentes.

La capa Modelo es el nivel más bajo del patrón MVC, ocupada de realizar el mantenimiento y operaciones asociadas a los datos que mantiene la aplicación. Está compuesta por clases que realizan el acceso a datos (los repositorios), de las entidades persistentes (entities) y de un conjunto de servicios que se encargan de realizar las operaciones de lógica de negocio y ofrecen sus operaciones a otras capas con el uso de una interfaz.

Se muestra el diseño de las clases persistentes (Figura 5.21), en las cuales se deben destacar dos aspectos, la direccionalidad de las entidades y las recomendaciones se spring/jpa:

● Direccionalidad:

1. *Si una entidad se relaciona con una línea simple con otra esto indica que se puede navegar de una a otra.*

2. *Si una entidad se relaciona con una línea que termina en punta indica que la entidad de origen (el lado sin punta) puede navegar a la otra, pero la otra no puede navegar de vuelta a la entidad origen. Ej: A → B, Podemos ir a B si estamos en A, pero si estamos en B no podemos ir a A.*

3. *La relación de Stats y StatsId indica que la segunda está contenida en la primera.*

● Recomendaciones de Spring/JPA:

1. *Las clases persistentes tienen un entorno no persistente que permite declarar funcionalidades adicionales sobre el comportamiento de estas clases, marcado este entorno con un @Transient (todos los métodos que tienen las entidades a mayores están contenidos en este entorno).*
   Esto favorece enormemente la construcción y legibilidad de los servicios al subdelegar en estos métodos operaciones de la lógica de negocio que se ejecutan sobre la entidad.

2. *Si se tienen múltiples operaciones que se pueden realizar sobre una entidad, se debe aplicar la técnica de Optimistic Locking, la cual establece que se debe declarar un atributo llamado versión para cada entidad que pueda tener un elevado número de modificaciones que puedan causar conflictos concurrentes (si hay conflicto se hace rollback y se relanza la petición de modificación).*

3. *Spring ofrece una caché de primer nivel donde las entidades quedan en estado "dirty" por ello al identificar las entidades en este estado de forma correcta, no es necesario actualizarlas en la base de datos con un save o update.*

4. *Para todas las entidades que poseen subgrupos o entidades de campos similares, spring declara que se debe crear una superclase sobre la cual hereden las demás clases. (Caso de Topic Category y Category).*

Finalmente, para la entidad Stats, al tratarse de una entidad débil esta tiene que tener su id compuesto embebido en ella StatsId.

5.2.4 Capa Controlador del servidor web.

La capa del controlador es una API REST, responsable de recibir los diferentes tipos de peticiones de los clientes con los controladores REST y maneja la interacción de la capa Modelo con la capa Vista.

La comunicación es creada con el paradigma REST, que define una gran variedad de métodos de acceso que especifican la acción que se realiza sobre un elemento. De entre las cuales se han usado:

- **GET:** Pide la información de un elemento.
- **POST:** Crea un elemento en la aplicación.
- **DELETE:** Elimina un recurso.

Existen muchos otros tipos de métodos como PUT, PATCH, HEAD, OPTIONS, e.t.c, pero para la implementación del sistema solo fueron necesarios los anteriores.

Se ha seguido una política restrictiva, es decir, toda aquella operación no declarada como permitida está bloqueada.

5.2.5 Capa Vista del servidor web.

La capa vista es la encargada de mostrar todos los datos de forma legible al usuario, producir las interfaces donde se llevan a cabo las diferentes llamadas producidas entre el servicio web y el navegador del cliente. Al tratarse de un servicio web, las peticiones lanzadas sobre HTTP comunican la información en formato JSON.

Está compuesta por todas las maquetas de las páginas de la aplicación (src/frontend).

5.2.6 Distribución de paquetes del proyecto.

El paquete base de la aplicación es **com.proyect.tfg**.

- **src.main.resources:** Contiene los ficheros de configuración del sistema

- **src.main.tests:** Contiene todos los ficheros de test de la aplicación, con una estructura idéntica a la de com.proyect.tfg
- **com.proyect.tfg.rest:** Contiene todos los ficheros de los controladores REST de la aplicación, está compuesto por 3 subpaquetes:
    - **com.proyect.tfg.rest.common:** Conjunto de recursos globales que pueden acceder todos los controladores, tales como el JWT, excepciones comunes, e.t.c.
    - **com.proyect.tfg.rest.controllers:** Contiene los controladores para cada uno de los subsistemas de la aplicación

- **com.proyect.tfg.rest.dtos:** Contiene los elementos necesarios para realizar la conversión entre entidades del dominio a elementos no persistentes. Organizados en 4 sub-paquetes según el sistema al que pertenecen.

- **com.proyect.tfg.model:** Contiene todos los ficheros de la capa Modelo de la aplicación, esta compuesto por 4 subpaquetes:
  - **com.proyect.tfg.model.entities:** Contiene todos los paquetes relacionados a las entidades y sus repositorios asociados, por cada entidad y repositorio existe un paquete que los engloba.
  - **com.proyect.tfg.model.util:** Contiene dos clases de utilidad para los servicios, un Block para paginación y un CoinConversor para el cambio del formato de monedas.
  - **com.proyect.tfg.model.exceptions:** Contiene todas las excepciones que pueden ser lanzadas por los diferentes servicios.
  - **com.proyect.tfg.model.services:** Contiene los diferentes paquetes asociados a cada servicio, por cada servicio se almacena en un paquete con su nombre la implementación y la interfaz.

Capítulo 6

Planificación y Costes

En este capítulo se recopila la información del Anexo de Estimación IFPUG, en cuanto a la estimación de la duración, coste y otros aspectos como el seguimiento.

6.1 Planificación inicial.

La estimación de la duración del proyecto, es de 4,45 meses, tal como hemos podido observar en el Apéndice de Estimación IFPUG. Adicionalmente el tiempo total en programación fueron 162h contabilizadas por el time tracker.

Para la realización de la planificación del proyecto es necesario dividirlo en subtareas con objetivos limitados y claros. La metodología de Scrum divide el proyecto en franjas de sprints, sobre cada uno de los cuales se definen las tareas que se van a realizar, ordenadas por prioridad (en este proyecto se le dio la misma cantidad de importancia a todas las tareas). Los Sprints incluyen los siguientes pasos:

1. *Creación del Sprint BackLog, que posee las historias de usuario.*
2. *Dividir las historias en casos de uso y su diseño.*
3. *Implementación del diseño de los casos de uso.*
4. *Pruebas.*
5. *Control de calidad.*

En concreto se ha decidido implementar primero la capa modelo, puesto que es la capa más grande, con más detalle y aspectos a tener en cuenta de la aplicación, también es la que se encarga de toda la lógica de negocio, luego los controladores y finalmente la capa vista.

6.2 Presupuesto.

En el Apéndice de Estimación IFPUG se define el presupuesto como 30829,84 €.

6.3 Seguimiento.

El proyecto como se ha comentado anteriormente se ha dividido en una serie de Sprints (24), la mayoría de ellos se han dedicado a la capa modelo, puesto la gran importancia que tiene en el diseño del servidor web del proyecto. Se han organizado los Sprints de la siguiente forma:

- **Sprints [1-19]:** Capa Modelo.

- **Sprints [19-23]:** Capa Controlador.
- **Sprint [24]:** Capa vista.

A la capa controlador se le ha asignado un sprint por cada sistema que proporciona la aplicación, mientras que la capa vista se le ha asignado un único sprint para realizar la implementación de la representación de la información.

Se han distribuido de esta forma puesto que la capa vista consiste en hacer las mismas operaciones una y otra vez para implementar los casos de uso, no tienen ninguna complejidad extra, solo llevan tiempo. Mientras que la capa de controlador a pesar de no tener que atender a una gran cantidad de detalles como la capa modelo, sigue teniendo cierta complejidad por todas las comprobaciones que tiene que realizar.

A continuación se explican las tareas llevadas a cabo en cada Sprint de forma breve.

6.3.1 Sprint 1.

Este Sprint tiene el objetivo la creación del diseño inicial de la aplicación, su estructura base con todas sus dependencias y la instalación de las herramientas necesarias para la realización del proyecto.

Adicionalmente se crea la base de datos, sus respectivas entidades y los tests asociados al correcto funcionamiento de estas.

6.3.2 Sprint 2.

Este Sprint tiene el objetivo la implementación del caso de uso de buscar las órdenes del usuario a lo largo del tiempo, especificada en los requisitos.

6.3.3 Sprint 3.

Este Sprint tiene el objetivo la implementación del caso de uso de crear una orden de compra de monedas de platino en el sistema de subastas.

6.3.4 Sprint 4.

Este Sprint tiene el objetivo la implementación del caso de uso de poner una orden de venta de monedas de platino en el sistema de subastas.

6.3.5 Sprint 5.

Este Sprint tiene como objetivo recuperar toda la información asociada a la variación del valor medio por moneda de platino a lo largo del tiempo en el sistema de subastas.

6.3.6 Sprint 6.

Este Sprint tiene como objetivo proveer al usuario la información asociada al precio mínimo que tendrá que pagar en monedas normales para poder conseguir al menos una moneda de platino en el momento que cree una orden de compra en el sistema de subastas.

6.3.7 Sprint 7.

Este Sprint tiene como objetivo proveer al usuario la información asociada al precio mínimo que tendrá que ofrecer en monedas normales si desea vender al menos una moneda de platino cuando realice una orden de venta en el sistema de subastas.

6.3.8 Sprint 8.

Este Sprint trata del login del usuario en la aplicación, calculando los datos derivados necesarios para la representación de la información al usuario final de su perfil. A la vez que se establece una petición HTTPS usando el link del perfil del usuario a Mojang para poder recuperar la imagen del usuario. (La api que da Mojang para hacer esto es realmente horrible por eso se tardará en hacer el login.)

6.3.9 Sprint 9.

Este Sprint se centra en conseguir las monedas que tiene el usuario en cierto instante del tiempo en la aplicación, devolviendo las monedas de platino y las monedas normales, estando las monedas normales convertidas en monedas de oro, plata y cobre.

6.3.10 Sprint 10.

Este Sprint recuperará la lista de compras que ha realizado el usuario a lo largo del tiempo en el sistema de tienda.

6.3.11 Sprint 11.

Este Sprint recuperará una lista de todos los productos contenidos dentro de una categoría.

6.3.12 Sprint 12.

Este Sprint se centra en poder aplicar un descuento por categoría, afectando a todos los productos asociados a esta.

6.3.13 Sprint 13.

Este Sprint se centra en poder actualizar todos los campos pertenecientes a un producto.

6.3.14 Sprint 14.

Este Sprint trata sobre la implementación de todos los aspectos que tiene que incluir el carrito, desde recuperar todos los items asociados, editarlos, eliminarlos y obtener el valor final con respecto a todos los items presentes en el carrito.

6.3.15 Sprint 15.

Este Sprint implementa la funcionalidad de comprar todos los productos que están presentes en el carrito.

6.3.16 Sprint 16.

Este Sprint implementa la funcionalidad de buscar topics en el sistema del foro, por categoría y keywords.

6.3.17 Sprint 17.

Este Sprint implementa la funcionalidad de crear un nuevo topic dentro del foro.

6.3.18 Sprint 18.

Este Sprint implementa la funcionalidad de buscar todos los detalles de los topics con sus respectivas respuestas.

6.3.19 Sprint 19.

Este Sprint implementa todas las funcionalidades relativas a editar reply, buscar topicCategories, borrar una reply, y cancelar una orden. Siendo estos los últimos aspectos del proyecto.

También incorpora las pruebas de refuerzo de tests del piTest.

6.3.20 Sprint 20.

Este Sprint implementa todas las funcionalidades asociadas al sistema de usuarios en los controladores.

6.3.21 Sprint 21.

Este Sprint implementa todas las funcionalidades asociadas al controlador del sistema del foro.

6.3.22 Sprint 22.

Este Sprint implementa todas las funcionalidades asociadas al controlador del sistema de subastas.

6.3.23 Sprint 23.

Este Sprint se encarga de realizar todas las funcionalidades asociadas al sistema de tienda, adicionalmente incorpora pruebas de arquitectura y elimina todas las mutaciones restantes detectadas por el piTest.

6.3.24 Sprint 24.

El último Sprint se encarga de implementar todas las funcionalidades de los sistemas en formularios en el frontend.

Capítulo 7

Implementación

En este capítulo se detalla el trabajo realizado durante el desarrollo de la aplicación a través de cada sprint. También se comentarán los aspectos más relevantes de cada sprint.

Se ha usado el plugin de Intellij Darkyen's Time Tracker mencionado en el capítulo 2, para medir el tiempo exacto que se ha pasado programando cada Sprint, el tiempo de programación con este plugin se integra de forma directa a los commits que se realizan en github. Tal y como vemos a continuación. El **Tiempo Total** de cada sprint solo incluye el tiempo programado.

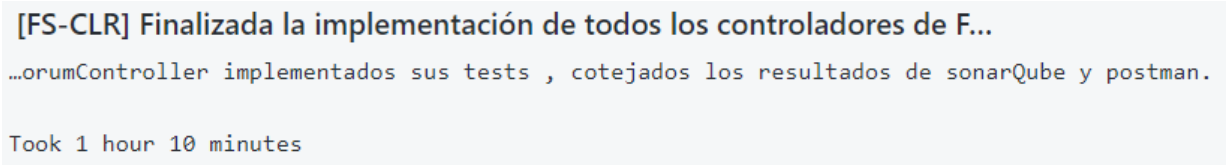

Figura 7.1: Time Tracker en git

7.1 Sprint 1.

Duración: [13 julio 2022 – 21 julio 2022 || Tiempo Total: 13h 43]

Este sprint se ha centrado en el inicio del proyecto, a través del Spring Initializr proporcionado por Intellij, con el cual se pudo generar la estructura básica de la aplicación, y los otros plugins necesarios como jacoco para el coverage, piTest… que fueron incorporados con el soporte de plugins de intellij.

Adicionalmente se centró en crear todas las entidades y testear sus respectivos métodos. (Las metodologías que se han seguido para realizar las pruebas se comentan en el siguiente capítulo en detalle).

Para la creación de estas se realizó la investigación de como mapear entidades débiles en spring, puesto que lo hace de una forma muy singular, y como implementar abstracción en las entidades persistentes para mejorar el código en cuanto a la no repetición de líneas/bloques del mismo.

7.2 Sprint 2.

Duración: [21 julio 2022 – 22 julio 2022 || Tiempo Total: 4h 23 mins]

Este sprint se ha centrado en enseñar al usuario las ordenes que ha realizado a lo largo del tiempo en las subastas, implicando tener que filtrar las órdenes en dos fases:

1. En primer lugar se deben recuperar todas las ordenes del usuario que estén expiradas y que no estén marcadas como tal, esto es lanzar una consulta para saber si alguna de las ordenes tiene su fecha de expiración pasada y en ese caso actualizarlas.

2. Una vez que se han filtrado las posibles expiraciones, se debe mostrar las órdenes por varios criterios, primero ordenando por su estado (primero las activas), luego por el tipo de orden que es (venta o compra) y finalmente sobre estos grupos ordenarlas por fecha de forma descendiente.

7.3 Sprint 3.

Duración: [22 julio 2022 – 28 julio 2022 || Tiempo Total: 7h 30 mins]

Este caso de uso se ha centrado en colocar una orden de compra en el sistema de subastas. Como este caso de uso es realmente complejo se ha dividido en varias partes:

1. *Se realizan diferentes comprobaciones, tales como:*
   - *El usuario tiene que tener suficientes monedas normales para poder hacer la transacción.*
   - *El usuario según su nivel de vip debe tener suficientes espacios para poner más órdenes activas.*
   - *Deben existir suficientes ordenes de venta asociadas a otros usuarios para hacer match, sino se creará una orden por defecto sin hacer matchs.*
2. *Filtrado de datos:*
   - *La lista de órdenes que hacen match son aquellas que son de venta, con un precio por unidad menor o igual del precio por unidad solicitado, ordenadas por el nivel de vip del usuario que ha creado esa orden y que estén activas.*
3. *Cálculos:*
   - *Por todos los elementos presentes en la se toman las monedas de platino suficientes para intentar satisfacer la demanda, si se satisface se termina, en caso contrario se marca en la orden final lo que queda por satisfacer para que futuras órdenes de otros usuarios puedan hacer match.*
   - *Por cada orden se filtra el stock que poseen, se toma lo necesario, marcando las operaciones que se han realizado sobre dicha orden y se paga al usuario propietario de esa orden el precio que ha solicitado aplicando las tasas correspondientes según el nivel vip que posea. Esta operación no se hace en el servicio, sino en el entorno no persistente de las entidades.*
4. *Finalmente se le da al usuario que pone esta orden de venta las monedas de platino conseguidas y se actualizan los valores correspondientes de la orden creada.*

7.4 Sprint 4.

Duración: [28 julio 2022 – 1 agosto 2022 || Tiempo Total: 7h 56 mins]

Este caso de uso se ha centrado en colocar una orden de venta en el sistema de subastas. El proceso es similar a la compra, salvo que se venden monedas de platino, por ello se tendrán que registrar las cantidades de monedas normales adquiridas (con las tasas ya aplicadas), las comprobaciones y los demas procesos son similares.

7.5 Sprint 5.

Duración: [1 agosto 2022 – 2 agosto 2022 || Tiempo Total: 3h 21 mins]

Este caso de uso proporciona al usuario las estadísticas de la evolución del precio medio de las monedas de platino a lo largo de los diferentes días. Esto implica modificar los casos de uso de venta y compra para que cada vez que iteren en las listas de órdenes que han podido realizar un match, y miren si existen estadísticas creadas en ese día. En el caso de que ya existan se engancharán a ella y sino la crearán.

Al crearla o continuar con las estadísticas del día se deberán introducir la cantidad de monedas de platino presentes en la transacción y las monedas normales que se han conseguido en total por esas monedas. Se calculará la media a partir de estos datos, una vez más esta operación no ocurre en el servicio sino en el entorno no persistente de las entidades.

7.6 Sprint 6.

Duración: [2 agosto 2022 – 3 agosto 2022 || Total: 46 mins]

Este sprint tiene como objetivo reciclar unas funcionalidades que ya implementa el caso de uso de comprar platino, en este caso se extrae la orden con precio por unidad mínimo que realizaría match con la orden de compra del usuario que quiere poner esta orden. Esta vez se filtrará como prioridad el precio más bajo.

7.7 Sprint 7.

Duración: [2 agosto 2022 – 3 agosto 2022 || Total: 26 mins]

Este sprint tiene como objetivo reciclar unas funcionalidades que ya implementa el caso de uso de vender platino, en este caso se extrae la orden con precio por unidad mínimo que realizaría match con la orden de venta del usuario que quiere poner esta orden. Esta vez se filtrará como prioridad el precio más bajo.

7.8 Sprint 8.

Duración: [3 agosto 2022 – 5 agosto 2022 || Total: 11h 37 mins]

En este sprint se trata la creación del caso de uso de login para los usuarios. Debido a que Mojang da una referencia a una pagina web renderizada donde se almacena la imagen del avatar del usuario, devuelve un div que hay que filtrar para obtener la imagen del usuario y poder crearla en los ficheros de recursos del proyecto en el caso de que no exista. En la siguiente figura se muestran las dos funciones necesarias para el filtrar y guardar la imagen del avatar.

```
3 usages  ramiroserantes
public File getResourceFile(String name) {
    String path;
    File file;
    try {
        path = Objects.requireNonNull(this.getClass().getResource( name: "/static/" + name + ".png")).getPath();
        file = new File(path);
    } catch (NullPointerException e) {
        return null;
    }
    return file;
}

3 usages  ramiroserantes
protected boolean createMojangProfilePhoto(String link, String userName) throws IOException {

    URL url = new URL(link);
    HttpsURLConnection connection = (HttpsURLConnection) url.openConnection();

    BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(connection.getInputStream()));

    /*Filtramos la entrada para obtener la imagen */
    String string = bufferedReader.readLine().split( regex: ",")[1].trim().split( regex: "\"")[0];

    byte[] imageBytes = javax.xml.bind.DatatypeConverter.parseBase64Binary(string);

   return ImageIO.write(ImageIO.read(new ByteArrayInputStream(imageBytes)), formatName: "png",
            new File( pathname: staticPath + userName + ".png"));

}
```

Figura 7.2: Filtrado de la imagen del perfil.

Una vez creada la imagen se devuelven los demás campos del usuario, incorporando todas las estadísticas que posee el usuario en el juego, y el rango global que posee. En la base de datos se almacena el rango como puntuación (Ej: 190 puntos rank). Si existen 2 usuarios con diferentes rangos, 190 puntos y 200 puntos, al segundo se le devolverá un 1 diciendo que es el primero en el ranking global y al primer usuario se le devolverá un 2 diciendo que es el segundo en el ranking global. En el caso de que múltiples usuarios posean la misma puntuación ambos estarán en la misma posición del ranking.

7.9 Sprint 9.

Duración: [5 agosto 2022 – 6 agosto 2022 || Total: 1h 7 mins]

En este sprint tiene como objetivo devolverle a los usuarios las monedas que poseen en un determinado instante, ya sea porque han llevado a cabo acciones que causan el cambio de este valor, o cuando sea necesario como resultado de algunas operaciones.

Para realizar esta operación se ha creado una clase de conversor donde filtre en el constructor los valores de las monedas normales en oro, plata y cobre, mientras que deje las monedas de platino (premium) intactas.

```
ramiroserantes
public CoinConversor() {}

ramiroserantes
public CoinConversor(Long coins) {
    if(coins != null) {
        this.copperCoins = (int) (coins % 100);
        this.silverCoins = (int) (((coins - this.copperCoins) / 100) % 100);
        this.goldCoins = (int) (coins / 10000);
    }
}
ramiroserantes
public CoinConversor(Long premiumCoins, Long coins) {
    this(coins);
    this.premiumCoins = premiumCoins;
}
```

Figura 7.3: Implementación del conversor de monedas.

Este conversor es realmente importante en todo el proyecto ya que gestiona todas las operaciones realizadas con monedas de la aplicación.

7.10 Sprint 10.

Duración: [7 agosto 2022 – 8 agosto 2022 || Total: 2h 10 mins]

En este Sprint se recupera toda la información relacionada a las compras realizadas por el usuario a lo largo del tiempo dentro de la aplicación. Estas compras se devolverán paginadas (el valor del tamaño de las páginas están en ficheros de configuración para que sea fácil su modificación) y ordenas por la fecha en orden descendiente.

7.11 Sprint 11.

Duración: [8 agosto 2022 – 8 agosto 2022 || Total: 28 mins]

En este sprint se recuperarán todos los productos contenidos dentro de una categoría ordenándolos de forma alfabética por el nombre.

7.12 Sprint 12.

Duración: [8 agosto 2022 – 10 agosto 2022 || Total: 2h 9 mins]

En este sprint se trata la aplicación de un nuevo descuento a todos los productos de una categoría seleccionada, para ello se tendrá que buscar todos los productos y ponerles el descuento solicitado. La operación de modificar el valor monetario de los productos se realizará en el entorno no persistente de la entidad Product, y se mirará que tipo de precio tiene el producto, si son euros o monedas de platino de la aplicación.

```
7 usages  ramiroserantes
@Transient
public void setNewDiscount(BigDecimal discount) {

    if(this.gameValue != null) {

        /* Deshacemos el descuento anterior en el caso del gameValue */
        this.gameValue = BigDecimal.valueOf(this.gameValue).divide(
                BigDecimal.ONE.subtract(this.discount.divide(BigDecimal.valueOf(100)))
                , scale: 2, RoundingMode.DOWN).longValue();
    } else {

        /* Deshacemos el descuento anterior en el caso del realValue */
        this.realValue = this.realValue.divide(
                BigDecimal.ONE.subtract(this.discount.divide(BigDecimal.valueOf(100)))
                , scale: 2, RoundingMode.DOWN).stripTrailingZeros();
    }

    this.discount = discount;
    applyDiscount();

}
```

Figura 7.4: Implementación del nuevo descuento en entorno no persistente.

Como se puede observar en la Figura 7.4 , esta operación no se realiza en el servicio sino en las entidades, filtramos por el tipo del valor y llamamos a otra función para que evalue el nuevo descuento. Finalmente el redondeo se realiza a la baja en las milésimas de decimal 0.005 → 0.

7.13 Sprint 13.

Duración: [10 agosto 2022 – 10 agosto 2022 || Total: 1h 28 mins]

Este Sprint debe implementar la funcionalidad que permite actualizar un producto en todos sus valores, para ello se comprobará la validez de estos campos y se recurre a un método no persistente para que observe los campos que se han introducido, en el caso de que sean nulos se dejará el valor previo y en otro caso se dejará el valor nuevo introducido.

7.14 Sprint 14.

Duración:[10 agosto 2022 – 11 agosto 2022 || Total: 7h 42 mins]

En este sprint se engloba las funcionalidades del carrito puesto que todos los asuntos que tratan están estrechamente relacionados entre sí.
Entre las funcionalidades a ofrecer, permite recuperar el carrito, añadir un item a él, modificar la cantidad de este item, obtener el valor final a pagar y recuperar los items presentes en el mismo.

La mayoría de estas operaciones se realizan en el entorno no persistente de la entidad shoppingCart, mejorando la legibilidad de los servicios.

7.15 Sprint 15.

Duración: [11 agosto 2022 – 11 agosto 2022 || Total: 1h 15 mins]

Este sprint implementa la funcionalidad de comprar los items que hemos metido en el carrito, siempre y cuando tengamos suficientes monedas de platino para realizar el pago. El dinero en euros no se comprobará debido a que habría que realizar una pasarela de pago que dependería de otra aplicación como por ejemplo de la api de PayPal, debido a que el proyecto ya es bastante enorme se recortó esta funcionalidad.

Para cada item comprado se registra una compra con su cantidad asociada y el precio que le ha costado al usuario.

7.16 Sprint 16.

Duración: [11 agosto 2022 – 12 agosto 2022 || Total: 2h 25 mins]

Este Sprint trata sobre la recuperación de los topics del sistema de foro, permitiendo al usuario filtrarlos por categoría o palabras clave presentes en la cabecera del topic. Independientemente de si el usuario elige usar estos dos parámetros de búsqueda, se deben mostrar los topics paginados y ordenados por la fecha de publicación.

7.17 Sprint 17.

Duración: [12 agosto 2022 – 12 agosto 2022 || Total: 39 mins]

En este sprint se incorpora la funcionalidad de crear un topic dentro del foro introduciendo la categoría de topic al que pertenece, su cabecera y el contenido que se desea.

7.18 Sprint 18.

Duración: [12 agosto 2022 – 12 agosto 2022 || Total: 3h 43 mins]

Este sprint realiza la implementación de buscar detalles del topic, es decir se obtendrá el contenido que posee. Adicionalmente obtendrán todas las respuestas asociadas a este con los usuarios que han creado estas respuestas de forma paginada (los datos de los usuarios son los justos y necesarios, no se incorpora su id ni datos privados).

El problema de esta funcionalidad es la necesidad de ordenar las respuestas que se han realizado por la fecha de edición (la cual puede ser nula) o por la fecha de creación eligiendo la mayor de ellas, este campo de ordenación no existe en la base de datos, por ello hay que crear una subconsulta con una query nativa, y realizar la paginación a mano con una count query adicional.

```
public interface ReplyDao extends PagingAndSortingRepository<Reply, Long>{

    1 usage   ramiroserantes
    @Query(value = "SELECT newR.id, newR.content, newR.isEdited, newR.postedDate, newR.editedDate, newR.topicId, newR.userId " +
            "FROM (SELECT r.id, r.content, r.isEdited, r.postedDate, r.editedDate, r.topicId, r.userId, " +
            " COALESCE(GREATEST(r.postedDate, r.editedDate), r.postedDate, r.editedDate) as MaxDate " +
            " FROM Reply r WHERE r.topicId = ?1 ) as newR" +
            " ORDER By newR.MaxDate DESC ",
            countQuery = "SELECT count(*) " +
                    "FROM (SELECT r.id , COALESCE(GREATEST(r.postedDate, r.editedDate), r.postedDate, r.editedDate) as MaxDate " +
                    " FROM Reply r WHERE r.topicId = ?1 ) as newR" +
                    " ORDER By newR.MaxDate DESC}" ,
            nativeQuery = true)
    Slice<Reply> findRepliesByTopic(Long topicId, Pageable pageable);
```

Figura 7.5: Implementación de la ordenación de las replies.

Como se puede observar en la figura (7.5) no es una operación muy sencilla de implementar en el repositorio.

7.19 Sprint 19.

Duración: [12 agosto 2022 – 14 agosto 2022 || Total: 4h 39 mins]

En este Sprint se trata todas las funcionalidades menores relacionadas con el foro, como añadir una respuesta, buscar en categorías del topic, borrar una respuesta, y cancelar una orden. Terminando la capa modelo.

7.20 Sprint 20.

Duración: [14 agosto 2022 – 15 agosto 2022 || Total: 9h 37 mins]

En este Sprint se inicia la construcción de la capa controlador, implementando el login y la obtención de las monedas del usuario. Los controladores consisten en llamar a los servicios para que realicen todas las operaciones y devolver un resultado, como algunos campos que devuelven los servicios son entidades persistentes estos tienen que ser convertidos a dtos, donde ya no estén vinculados a la base de datos.

La autenticación de usuario usa como se vio en la asignatura de Programación Avanzada un web token JWT generado con el id del usuario su rol web y la fecha de expiración en mili segundos.

El problema que presentaron estos controladores fue que en la última versión de spring ha cambiado bastante la forma de filtrar las peticiones, si antes se hacía extendiendo de un Adapter (ya deprecated) ahora se usa un método llamado filterChain que no proporciona un authenticationConfiguration para poder aplicar un filtro de forma fácil.
Por ello se tuvo que investigar las aproximaciones actuales que solucionan este problema.

Se han gestionado todas las excepciones con ExceptionHandlers internacionalizando las respuestas. (Se verá más adelante en ejemplos del PostMan).

7.21 Sprint 21.

Duración: [15 agosto 2022 – 17 agosto 2022 || Total: 8h 4 mins]

Este Sprint consiste en implementar todas las funcionalidades del sistema del foro en su respectivo controlador modificando los valores devueltos a dtos.

7.22 Sprint 22.

Duración: [17 agosto 2022 – 19 agosto 2022 || Total: 4h 5 mins]

Este Sprint implementa todas las funcionalidades asociadas al sistema de subastas en el controlador modificando estos valores a dtos.

7.23 Sprint 23.

Duración: [19 agosto 2022 – 20 agosto 2022 || Total: 16h 52 mins]

En este Sprint se implementa todos los casos de uso del sistema de tienda en los controladores, y adicionalmente se termina de revisar todas las pruebas de la aplicación, haciendo tests de arquitectura y corrección de mutaciones.

7.24 Sprint 24.

Duración: [20 agosto 2022 – 3 septiembre 2022 || Total: 37h 41 mins]

En este Sprint se implementan todos los sistemas en la capa vista, la realización de esta operación es un proceso repetitivo que se puede dividir en diferentes partes:

1. *Llamar al backend para la obtención de información.*
2. *Crear un tipo de acción para la operación realizada hecho.*
3. *Crear una acción que use el tipo de acción.*
4. *Incorporar al estado esta acción.*
5. *Crear un formulario donde se usa los resultados de la acción.*

Es una tarea larga pero sin mucha complejidad una vez que se hace una petición de forma existosa.

Capítulo 8

Pruebas

En este capítulo se comentarán las diferentes pruebas que se han realizado a lo largo del proyecto.

Las pruebas de software son un componente vital en su desarrollo, ya que permiten encontrar errores con respecto a la especificación y mejorar enormemente nuestra confianza en el funcionamiento del software desarrollado. Pero por desgracia no pueden demostrar la ausencia de errores, pero si demostrar que todos los intentos realizados para hacer fallar el software con respecto a su especificación han fallado.

Se han realizado pruebas de unidad, integración, arquitectura, calidad y solidez de tests (a modo de mutaciones). Además se garantiza que los tests comprueben la cantidad mínima posible de código, existiendo una sola comprobación (assert) por test, catalogando cada tipo de test indicando los parámetros introducidos (esto se especifica sobre cada método) y la categoría de test a la que pertenecen (en la documentación de la propia clase).

```
/** Tests para comprobar el correcto funcionamiento de las funcionalidades del servicio de shopping, se
 * probará la mínima cantidad en cada test.
 *
 * - Nivel: Debido a que en los servicios inter-operan diferentes unidades entre sí, son tests de integridad, salvo
 * en funciones propias.
 *
 * - Categoría: Pruebas de caja negra funcionales positivas.
 *
 * - Datos Elegidos:
 *      Se especificará en cada caso la selección de datos.
 */
396 usages  ramiroserantes
@SpringBootTest
@ActiveProfiles("test")
@Transactional
class ShoppingServiceIntegrityTest {
```

Figura 8.1: Ejemplo de documentación de Tests.

8.1 Pruebas de unidad.

Los test de unidad tienen como objetivo comprobar el correcto funcionamiento de los diferentes componentes del sistema de forma individual, esto es, todos aquellos métodos o funciones cuya ejecución no incluya la interacción con otros componentes.

En el caso de los controladores se ha utilizado la librería Mockito, para crear instancias falsas de los componentes de los que depende, permitiendo así recibir elementos de entrada sin depender de esos componentes.

Adicionalmente se han comprobado en estos casos los tipos de datos devueltos con peticiones HTTP a través de PostMan.

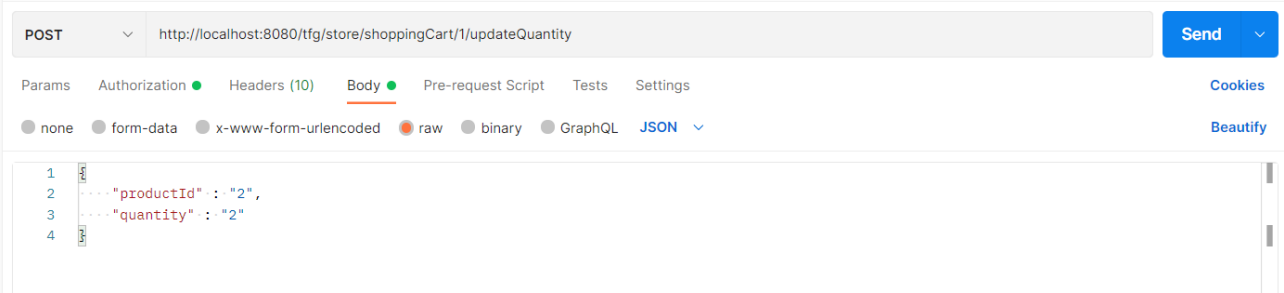

Figura 8.2: Ejemplo de resultados de PostMan.

En determinados tests se han empleado imágenes para comprobar el correcto funcionamiento de funciones que implican la manipulación de estas, como la llamada a Mojang para recibir la skin del usuario.
(src/main/resources/static)

8.2 Pruebas de integridad.

Mientras que los test de unidad se encargan de probar aquellas funciones y métodos de los diferentes componentes que no impliquen interactuar con otros componentes, los test de integridad se encargarán de probar el funcionamiento de las secciones de los componentes en las que si se produzca esta interacción.

Para llevar a cabo esta tarea se ha usado la librería Junit, la cual proporciona un conjunto de librerías que facilitan enormemente la creación y ejecución de pruebas automatizadas.

Para facilitar la independencia de los tests con respecto a la ejecución de la aplicación normal se ha creado otra base de datos diferente done se produzca la ejecución de estos.

8.3 Pruebas de calidad.

Para realizar los test de calidad se empleó SonarQube, una aplicación que permite detectar y ofrecer diferentes tipos de soluciones a los problemas que vaya detectando en el código. Los problemas detectados por SonarQube son muy diversos, como brechas de seguridad dentro del código, malas prácticas de programación (code smells), código repetido, cobertura del código por los test… .

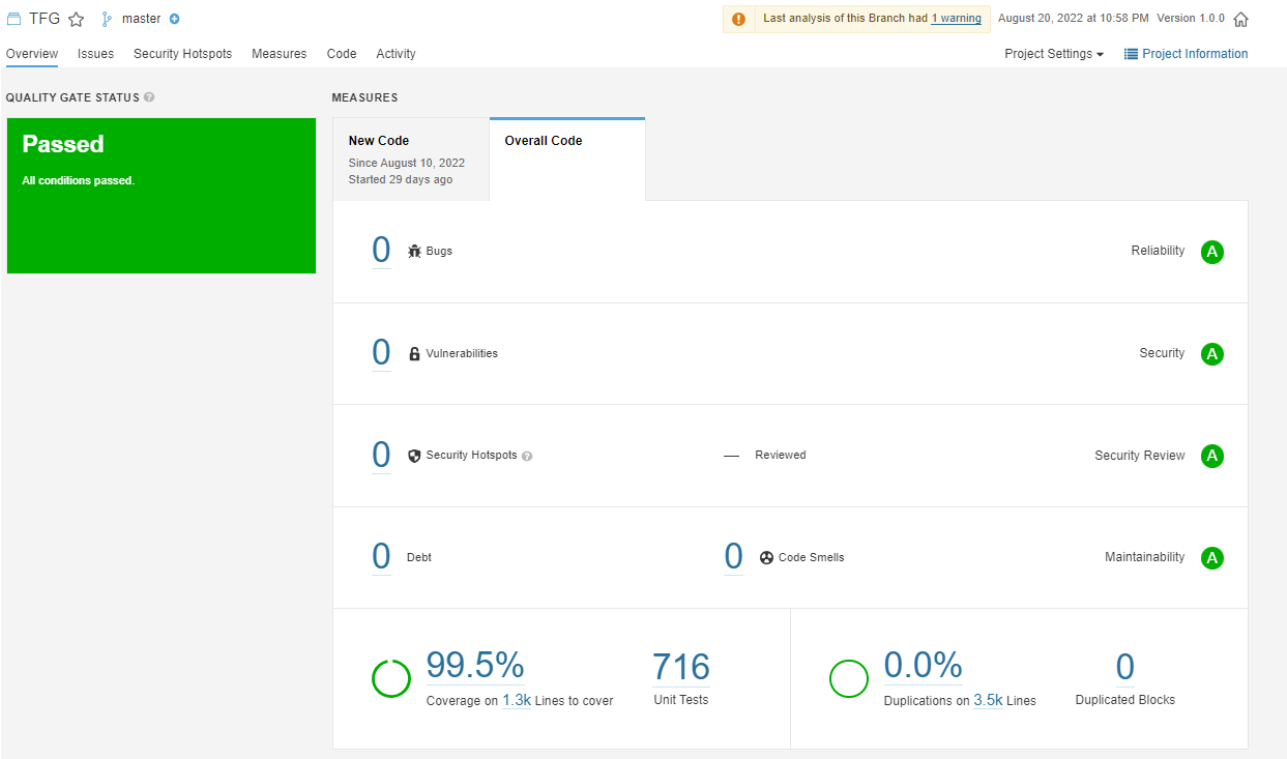

Figura 8.3: Resultados de SonarQube

Cabe destacar que para conseguir un 0% de líneas de código repetidas se ha usado herencia entre clases, y el nùmero de test es tan significativo debido a que se prueba la mínima cantidad de código posible para cada función. SonarQube no diferencia entre test de unidad e integridad, los califica a todos como unidad.

8.4 Pruebas de arquitectura.

Para garantizar que se ha implementado correctamente la arquitectura MVC del sistema se han declarado reglas con ayuda de librería de ArchUnit. Esta librería permite crear tests de forma que ante un conjunto de clases pasadas como parámetro se les pueda aplicar diferentes condiciones.

Las reglas establecidas son las siguientes:

- Las entidades/repositorios no pueden depender de los servicios, de los controladores, de la capa vista.
- Los servicios deben depender de los repositorios/entidades, pero no de los controladores ni de la capa vista.
- Los controladores deben depender de los servicios y los repositorios/entidades, pero no de la vista.
- El útil no depende de nadie.

```java
@Test
void entitiesShouldNotDependOnServicesLayer() {

    noClasses() GivenClasses
            .that().resideInAnyPackage( ...strings: "com.proyect.tfg.model.entities..")
            .should() ClassesShould
            .dependOnClassesThat() ClassesThat<ClassesShouldConjunction>
            .resideInAnyPackage( ...strings: "com.proyect.tfg.model.services..") ClassesSh
            .because( s: "Entities should not depend on servicelayers") ArchRule
            .check(importedClasses);
}

/** Las entidades no deben depender de sus controladores. **/
```

Figura 8.4: Ejemplo de reglas de arquitectura.

8.5 Pruebas con mutaciones.

El proyecto ha logrado cubrir con todos los tests un 99,5% de las líneas del código. Pero la cobertura, tal y como se ha aprendido en la materia de verificación y validación de software, es solo una garantía mínima de calidad, nada nos garantiza que se hayan realizado bien los tests, por ello se ha tomado la decisión de incorporar mutaciones con el plugin piTest.

Una mutación es una alteración del código fuente (ejemplo cambiar una suma por resta, omitir líneas, e.t.c) haciendo que el código siga funcionando. Lo que tratan las mutaciones es verificar si los diferentes test se han dado cuenta de los cambios que se han introducido en el código, en el caso de que no se den cuenta del cambio al ejecutarse, se considerará que la mutacion esta viva, en caso contrario se mata la mutación.

Este sistema es realmente útil para reforzar los tests realizados y aumentar su solidez, ya que con ellas garantizamos que estamos realizando las comprobaciones adecuadas. Adicionalmente provee de reportes automáticos muy útiles para la documentación del sistema, indicando para cada clase las mutaciones vivas y porque están vivas. (En el caso de que una mutación esté viva y no sea necesario eliminarla se puede dejar como está, no todas las mutaciones tienen porque ser matadas).

Mutations

1. replaced return value with null for com/proyect/tfg/model/services/auction_service/AuctionServiceImpl::getDate → KILLED
1. removed call to com/proyect/tfg/model/services/permission_service/PermissionChecker::checkUserExists → KILLED
1. negated conditional → KILLED
1. removed call to com/proyect/tfg/model/entities/order/Order::setStatus → KILLED
1. replaced return value with null for com/proyect/tfg/model/services/auction_service/AuctionServiceImpl::findOrdersByUser → KILLED
1. changed conditional boundary → KILLED
2. negated conditional → KILLED
1. Replaced long multiplication with division → KILLED
2. removed call to com/proyect/tfg/model/entities/user/User::withdrawCoins → KILLED
1. negated conditional → KILLED
1. replaced return value with null for com/proyect/tfg/model/services/auction_service/AuctionServiceImpl::lambda$buyPremiumCoins$0 → KILLED
1. changed conditional boundary → SURVIVED
2. changed conditional boundary → SURVIVED
3. Changed increment from 1 to -1 → KILLED

Figura 8.5: Ejemplo de mutaciones.

Pit Test Coverage Report

Project Summary

Number of Classes	Line Coverage		Mutation Coverage		Test Strength	
23	100%	671/671	97%	344/353	98%	344/352

Breakdown by Package

Name	Number of Classes	Line Coverage		Mutation Coverage		Test Strength	
com.proyect.tfg.model.entities.category	2	100%	17/17	100%	4/4	100%	4/4
com.proyect.tfg.model.entities.order	2	100%	81/81	100%	35/35	100%	35/35
com.proyect.tfg.model.entities.product	1	100%	55/55	100%	32/32	100%	32/32
com.proyect.tfg.model.entities.purchase	1	100%	23/23	100%	7/7	100%	7/7
com.proyect.tfg.model.entities.reply	1	100%	23/23	100%	8/8	100%	8/8
com.proyect.tfg.model.entities.shopping_cart	1	100%	29/29	100%	12/12	100%	12/12
com.proyect.tfg.model.entities.shopping_cart_item	1	100%	24/24	100%	12/12	100%	12/12
com.proyect.tfg.model.entities.topic	1	100%	24/24	100%	8/8	100%	8/8
com.proyect.tfg.model.entities.user	3	100%	105/105	100%	38/38	100%	38/38
com.proyect.tfg.model.exceptions	2	100%	12/12	100%	4/4	100%	4/4
com.proyect.tfg.model.services.auction_service	1	100%	78/78	94%	58/62	94%	58/62
com.proyect.tfg.model.services.catalog_service	1	100%	24/24	100%	18/18	100%	18/18
com.proyect.tfg.model.services.forum_service	1	100%	30/30	100%	17/17	100%	17/17
com.proyect.tfg.model.services.permission_service	1	100%	28/28	100%	18/18	100%	18/18
com.proyect.tfg.model.services.shopping_service	1	100%	47/47	100%	25/25	100%	25/25
com.proyect.tfg.model.services.user_service	1	100%	27/27	80%	12/15	86%	12/14
com.proyect.tfg.model.services.util	2	100%	44/44	95%	36/38	95%	36/38

Figura 8.6: Reporte del piTest.

Como se puede observar en las Figuras 8.6 y 8.5 se ha logrado un sistema de tests muy fuertes, donde podemos estar seguros de que hemos realizado las comprobaciones correctas en cada función testeada del sistema, puesto que se han dado cuenta de la enorme cantidad de cambios realizados por las mutaciones. (Las mutaciones que quedaron vivas no son necesarias eliminarlas)

Cabe destacar que para utilizar el piTest de forma correcta es vital realizar test que realmente verifiquen la mínima cantidad de código posible por cada uno, debido a que las mutaciones realizarán numerosos cambios por cada línea, lo que provoca que si hacemos tests genéricos y grandes no podrán darse cuenta de todos los cambios realizados por estas, y por consecuencia fallarán los tests de solidez.

Capítulo 9
Producto Final

En este capítulo se realizará una muestra de la solución final del proyecto, ilustrando el funcionamiento del mismo. El objetivo del capítulo es proporcionar una guía de la utilización del producto final creado por el proyecto, y describir sus características.

Se mostrará el funcionamiento normal del producto, usando capturas de pantalla para ofrecer un soporte visual que facilite la comprensión de su funcionamiento.

La aplicación es nombrada como **TFG Project** y una vez más se agruparán las funcionalidades proporcionadas por el proyecto en los cuatro sistemas que lo componen.

9.1 Producto Final. User System.

Cuando el usuario se introduce por primera vez en la aplicación (figura 9.1) puede tomar la decisión de logearse o ir a otro sistema de la aplicación pero sin poder acceder a las funcionalidades que requieren estar registrado.

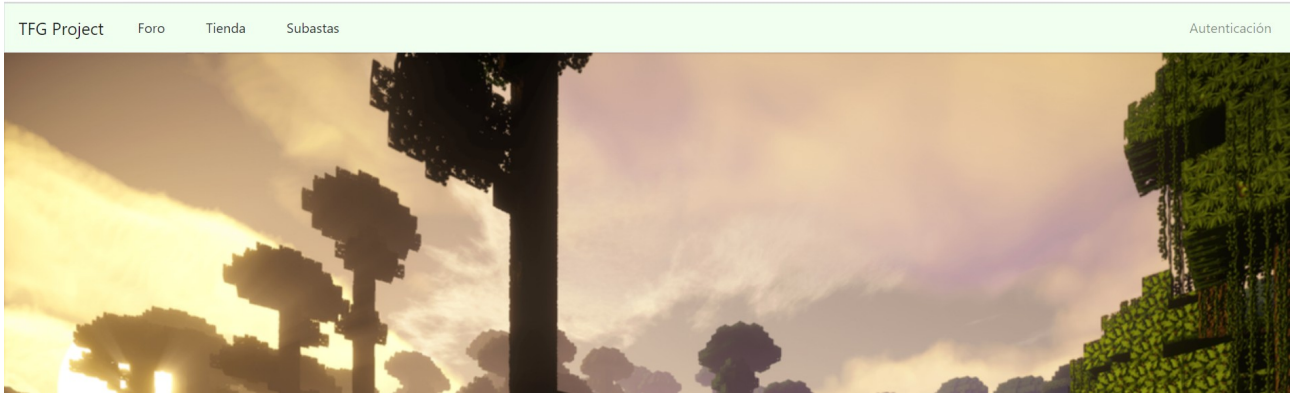

Figura 9.1: Inicio de la aplicación.

Figura 9.2: Login en la aplicación.

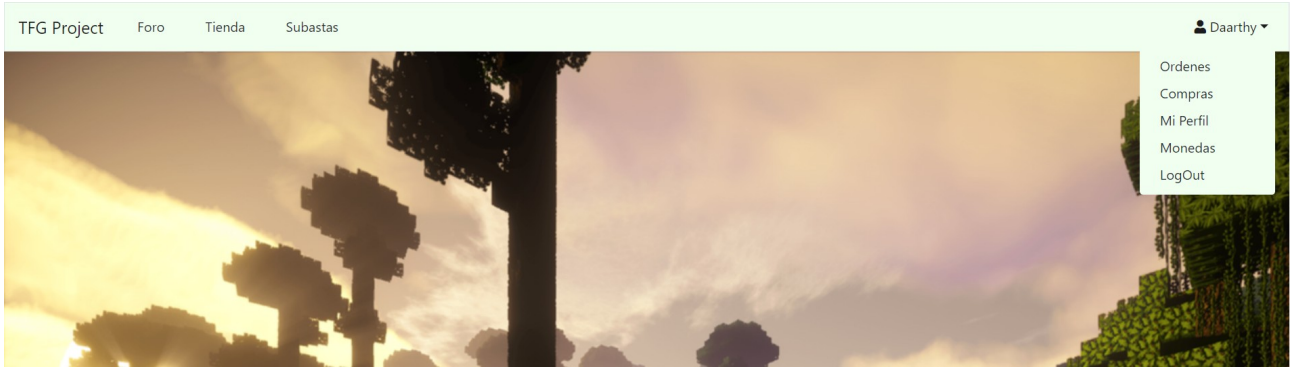

Figura 9.3: Página principal logeado.

Figura 9.4: Página de órdenes del usuario.

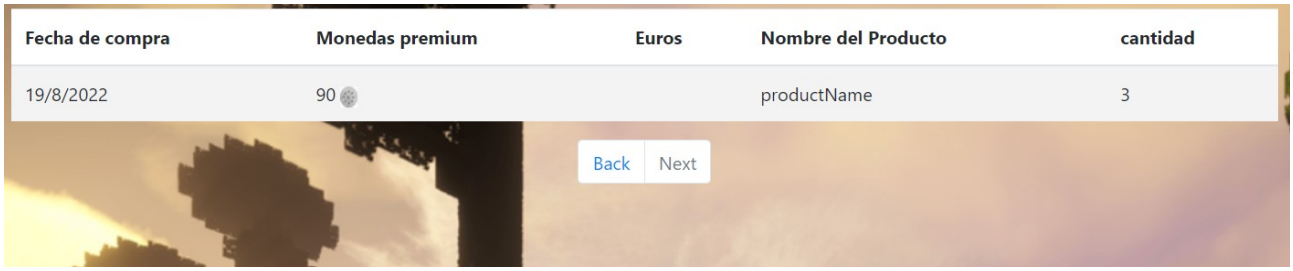

Figura 9.5: Página de compras del usuario.

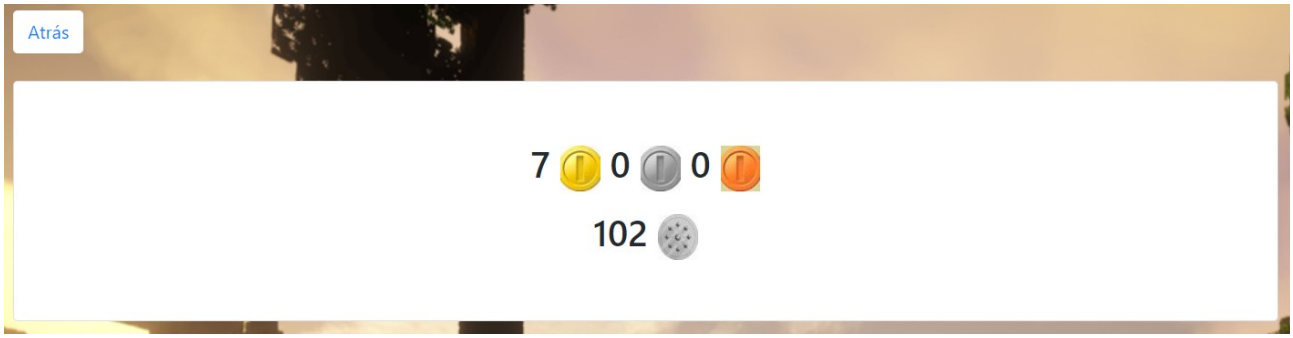

Figura 9.6: Página de monedas del usuario.

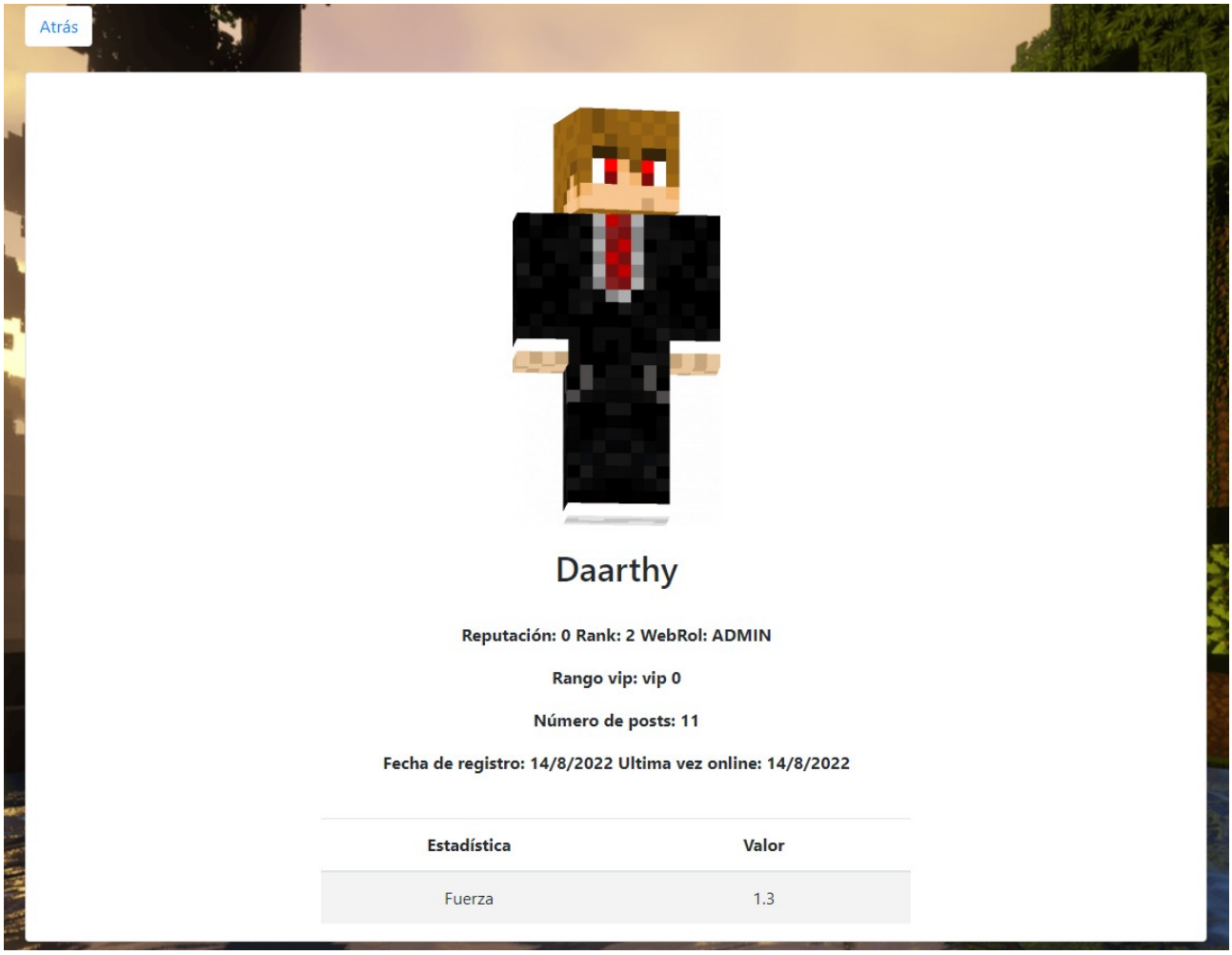

Figura 9.7: Página de datos del perfil del usuario.

Como se puede contemplar en las anteriores figuras, el producto final del sistema del usuario ha sido muy fiel (salvo en detalles del perfil del usuario) al diseño inicial.

9.2 Producto Final. Forum System.

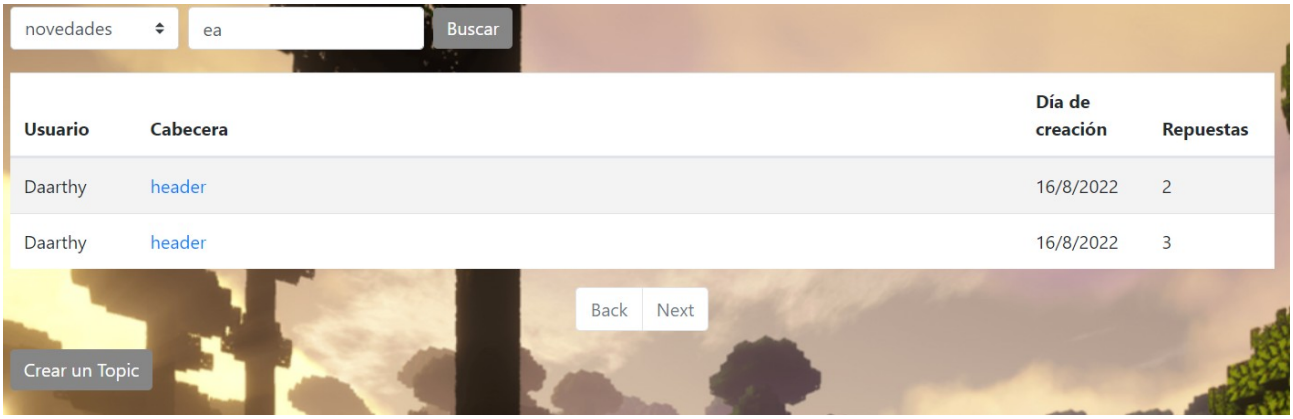

Figura 9.8: Página principal del foro .

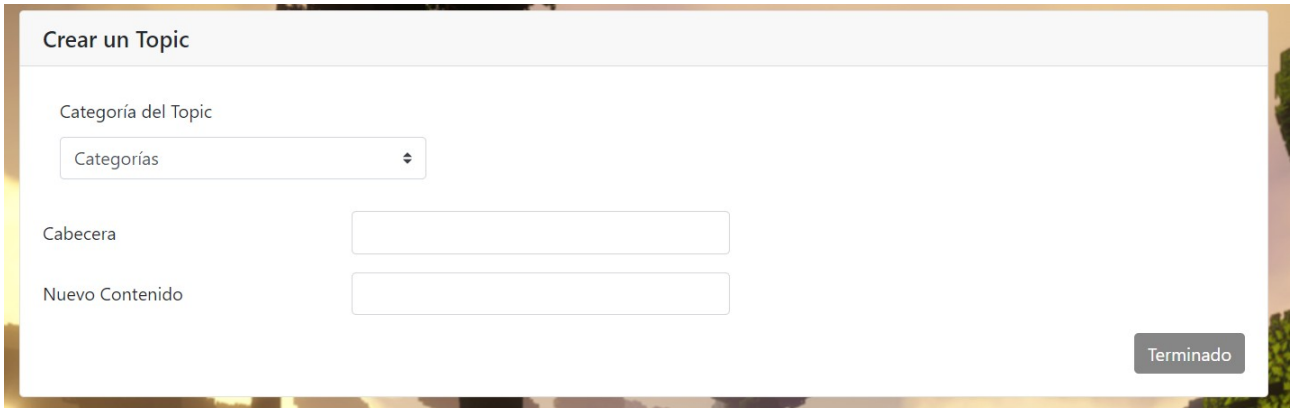

Figura 9.9: Página para crear un topic.

Figura 9.10: Página para crear una respuesta.

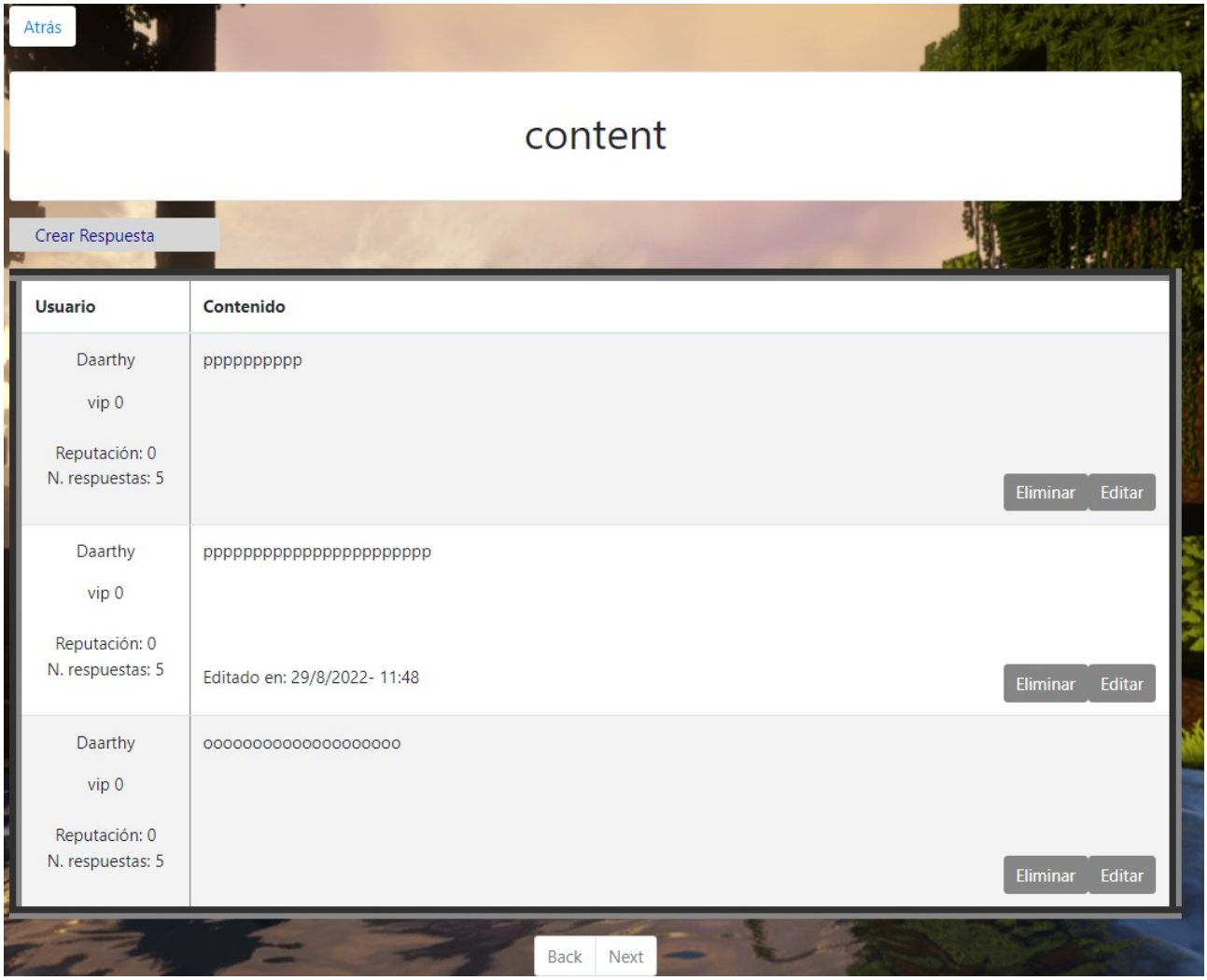

Figura 9.11 : Página de detalles de un topic.

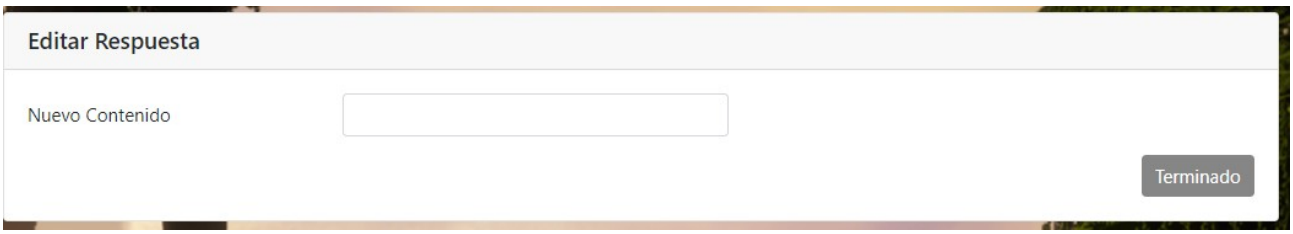

Figura 9.12: Página para editar una respuesta.

9.3 Producto Final. Auction System.

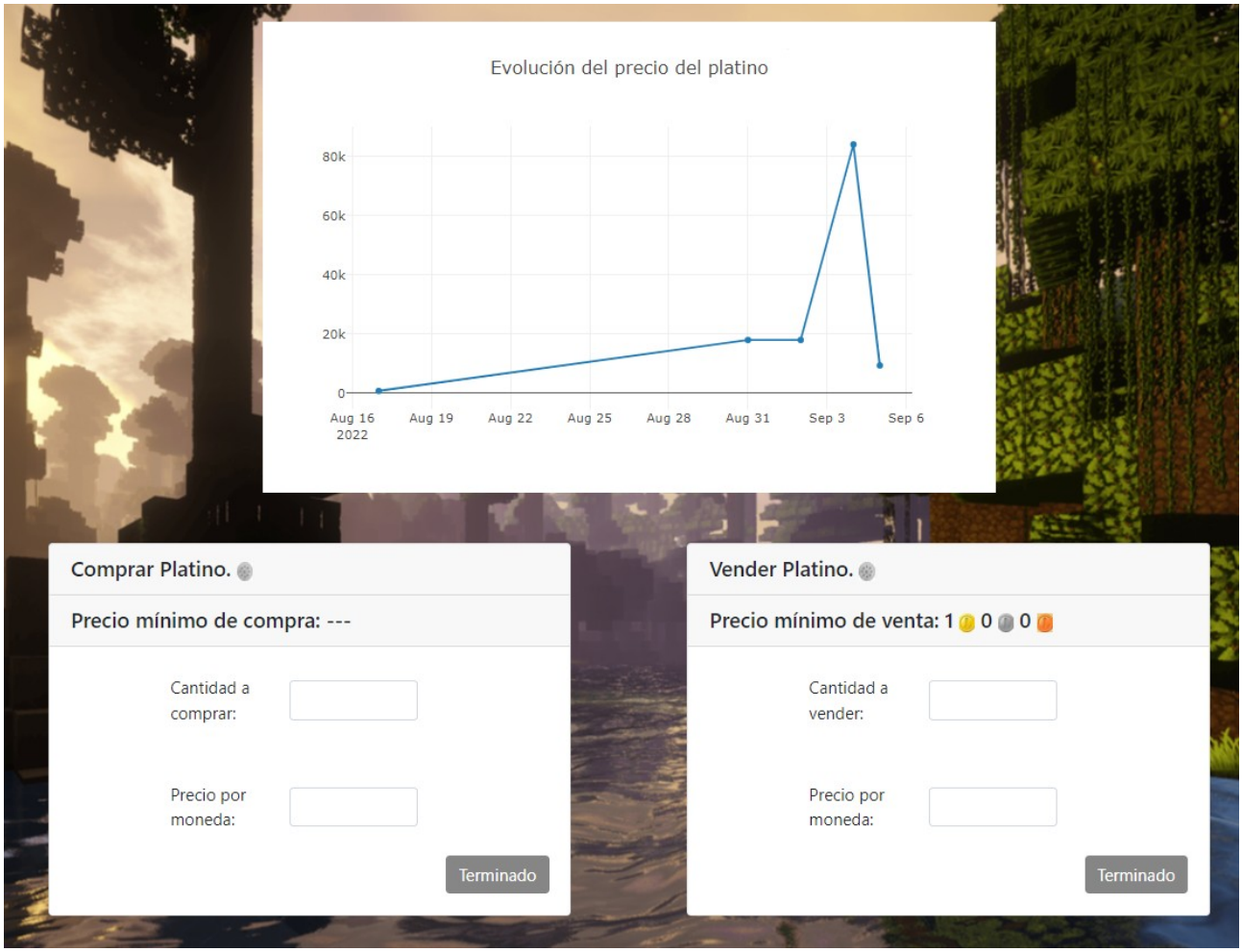

Figura 9.13: Página de subastas.

9.4 Producto Final. Shopping System.

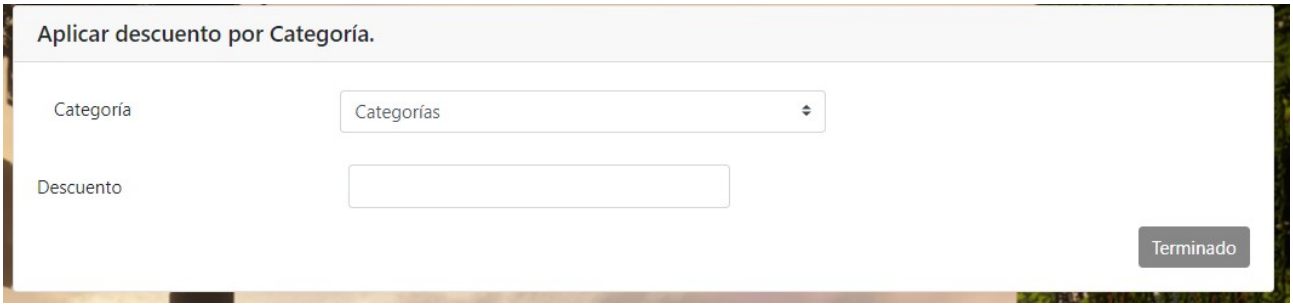

Figura 9.14: Página de descuento por categoría.

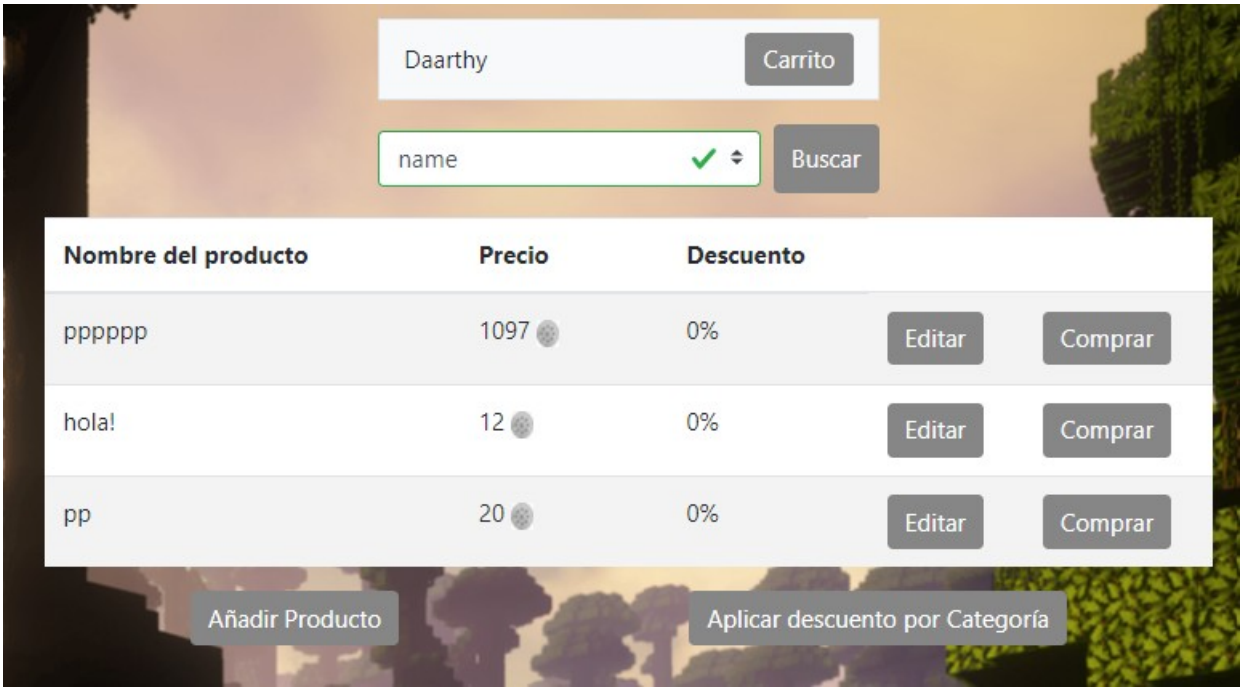

Figura 9.15: Página de la tienda.

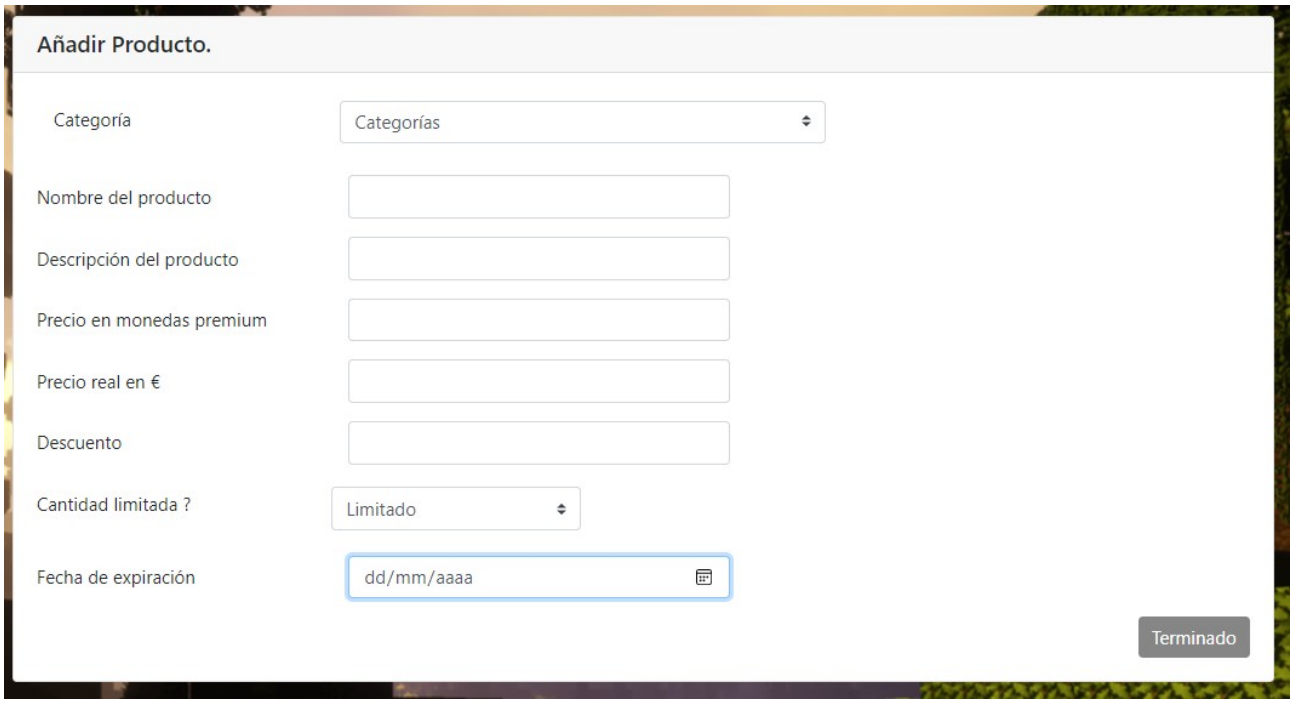

Figura 9.16: Página de añadir un producto.

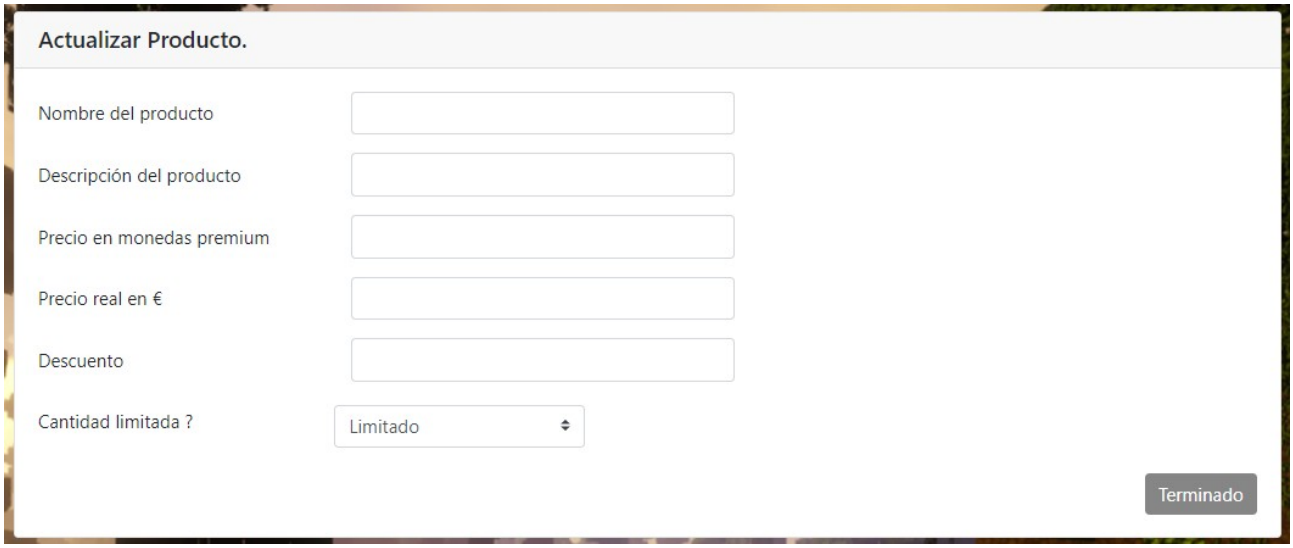

Figura 9.17: Página de actualizar un producto.

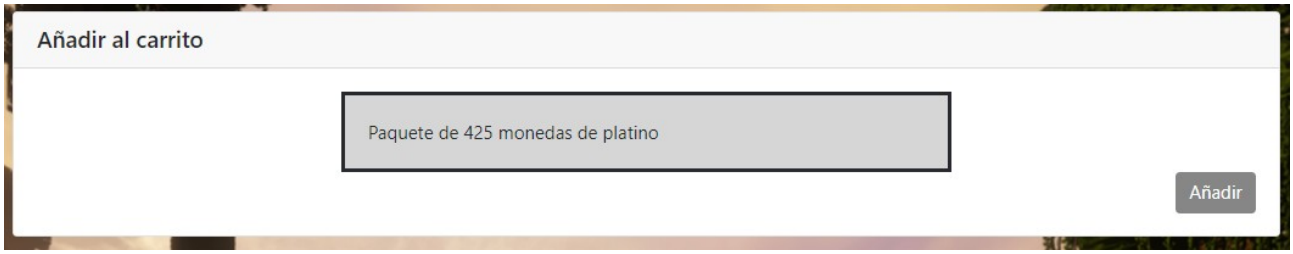

Figura 9.18: Página de añadir un producto al carrito.

Figura 9.19: Página de visualizar el carrito.

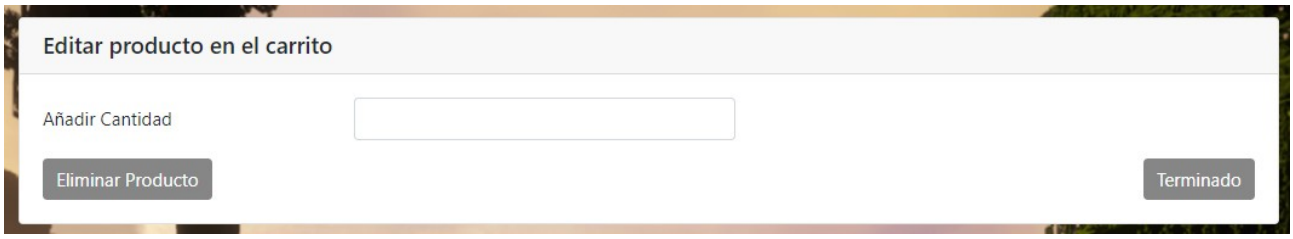

Figura 9.20: Página de editar un producto en el carrito.

Capítulo 10

Conclusiones

Este capítulo final representa una evaluación sobre el desarrollo llevado a cabo en el proyecto y detalla las posibles mejoras aplicables al mismo en el futuro para mejorar el valor del producto final.

10.1 Desarrollo llevado en el proyecto.

Al terminar el proceso de desarrollo de este proyecto se ha realizado la verificación del cumplimiento de los diferentes objetivos propuestos en la especificación de requisitos. Dando lugar a una aplicación web que proporciona una gran variedad de funcionalidades adicionales a los jugadores de un servidor de Minecraft, más concretamente:

- Permite a los usuarios gestionar su información del juego. User System.
- Permite a los usuarios discutir entre ellos diferentes aspectos del juego. Forum System.
- Permite a los usuarios realizar la compra y mantenimiento de una tienda online relacionada con los objetos del juego. Shopping System.
- Permite a los usuarios comerciar entre ellos sus respectivas monedas para favorecer la economía del juego. Auction System.

Alcanzándose objetivos propuestos en el anteproyecto de crear varios sistemas (de forma limitada porque se puede crear una infinidad de funcionalidades a estos servidores de minecraft en capa web) para los jugadores de este tipo de aplicaciones, diseñando de forma adicional una nueva arquitectura que facilite la creación de nuevas aplicaciones de este tipo (ya que antes no existía una similar).

10.2 Lecciones aprendidas.

Durante la realización de este proyecto se han adquirido valiosos conocimientos con respecto a las tecnologías empleadas y reforzadas las lecciones aprendidas durante la carrera. Entre ellos se pueden destacar algunos campos:

- En Spring se ha profundizado en varios aspectos:

    1. *Se ha aprendido a realizar la implementación de entidades débiles.*
    2. *Se ha profundizado en la creación de consultas complejas a bases de datos en los repositorios, con la creación de querys nativas.*
    3. *Debido a que se ha usado la última versión disponible de Spring se han actualizado todos los conocimientos aprendidos en este campo.*

4. *En cuanto a testing, se han aprendido valiosas lecciones relacionadas con la implementación de estos, debido a la gran diversidad de tipos de tests que se han realizado en el proyecto.*

- En React también se han profundizado en varios aspectos:

1. *Se ha adquirido una mayor solidez en el desarrollo de casos de uso con React, debido a que se ha tenido que repetir de forma constante la metodología de implementación que requiere para darles soporte.*
2. *Se ha profundizado en la creación de estadísticas con ayuda de Ploty.*

Finalmente debido a que es un proyecto bastante grande que imita como debería ser un proyecto real desarrollado en una empresa, se ha adquirido una experiencia valiosa, sobre todo porque se han utilizado herramientas y frameworks (Spring y React) bastante demandados dentro de las empresas. Y se han aplicado los conocimientos aprendidos en la materia de Diseño de Software para poder implementar una arquitectura nueva y original que sirva como modelo para este tipo de proyectos.

10.3 Líneas futuras.

Si bien es cierto que se han logrado implementar todos los objetivos propuestos para este proyecto, se le podrían añadir múltiples funcionalidades adicionales para incrementar su valor, como ya se ha mencionado anteriormente en este tipo de proyecto las funcionalidades que pueden ser cubiertas son prácticamente ilimitadas. Algunas de las posibles funcionalidades que se le podrían añadir son:

- Se podrían incorporar más funcionalidades en el sistema del Foro, tales como permitir a un usuario moderador eliminar un Topic de cualquier usuario, cerrar un topic para que ya no admita más respuestas, crear Topics privados visibles unicamente por determinados usuarios que pertenezcan a un grupo…
- En el sistema de Subastas se podría incorporar el intercambio de items del juego de un usuario a cambio de otros items diferentes, o monedas (Esta operación es complicada puesto que se debe acceder a una clase llamada ItemMeta de la base de datos no relacional Properties).
- Se podría incorporar en el sistema del Usuario la posibilidad de ver a todos los amigos que tiene y si están online actualmente dentro del servidor.

Apéndices

Apéndice 1

Estimación IFPUG

Para realizar la estimación del tamaño funcional de un proyecto de software basándose en la metodología del IFPUG [20] serán necesarios los siguientes elementos:

1. El modelo de la base de datos relacional, para poder calcular el tamaño y cantidad de los ILF's (Ficheros Lógicos Internos) y los ELF's (Ficheros Lógicos Externos). [Ya se incorpora en la memoria principal].

2. Un modelo de las diferentes pantallas que estarán presentes en la aplicación, para calcular las entradas/salidas/consultas que se van a realizar. [Ya se incorpora en la memoria principal].

3. Con ayuda de los dos puntos anteriores se realiza el cálculo de puntos función por cada funcionalidad o almacenamientos .

4. Se calcula el cálculo global de puntos función sin ajustar, recopilando la información del punto 3.

5. Se calcula con las métricas establecidas por el ifpug el coste, duración , productividad, e.t.c, con los puntos función totales.

6. Coste Final del proyecto.

3. Estimación del tamaño funcional IFPUG.

Tomando como referencia el modelo de datos relacional, se calcularán los tamaños de los ILF's y ELF's del proyecto, agrupando esta estimación en sistemas por la función que desempeñan.

Debido a que todas las tablas son gestionadas por la aplicación y no se depende de otros sistemas ajenos, no habrán Ficheros Lógicos Externos.

RET = Record Element Type.
DET = Data Element Type.
FTR = File Type Referenced.

==========

Usuarios

Archivo Lógico Interno (User).

RET's = Número de subgrupos presentes en la tabla (2) => Todos los campos se toman como un subgrupo + la entidad débil = 2 RET's.

DET's = Número de campos percibidos por el usuario – Campos creados por técnicas de implementación y no visibles por el usuario = Total (17) – 2 = 15 DET's.

Complejidad ILF (User) = BAJA (2 RET's, 15 DET's).

Salida Externa EO (Consultar Detalles de User).

Debido a los cálculos internos que se deben realizar para llevar a cabo esta función (dando como resultado datos derivados), tales como calcular el ranking y el número de posts, se trata de una consulta externa EO.

FTR's = Número de ficheros lógicos leídos = 3 (User, reply y topic) => 3 FTR's.

DET's = Número de campos de entrada para realizar la consulta (1 [id del usuario]) + Botón (1) + Mensajes de error (0) + Campos mostrados en la salida (12) - Campos mostrados en la salida que coinciden con los datos de entrada(0) =14 DET's.

Complejidad EO (Consultar Detalles de User) = MEDIA (3 FTR's, 14 DET's).

Salida Externa EO (Login).

Debido a que esta función es una consulta externa que implicarán cálculos adicionales en la salida de la consulta, debido a que las monedas se encuentran almacenadas en un formato distinto al que se le mostrará al usuario.

FTR's = Número de ficheros lógicos leídos = 1 (User) => 1 FTR.

DET's = Número de campos de entrada (2) + Botón (1) + Todos los posibles mensajes de error(1) + Campos de salida (2) = 6 DET's.

Complejidad EO (Login) = BAJA(1 FTR, 6 DET's).

Salida Externa EO (Orders).

Debido a que esta función es una consulta hay que calcular de forma adicional la cantidad de monedas que se han conseguido en cada una de las transacciones y añadirlas al resultado final es una Salida externa .

FTR's = Ficheros leídos = 1 (Orders) => 1 FTR.

DET's = Número de campos de entrada (1 UserId) + Botón (1) + Todos los posibles mensajes de error (0) + Campos de salida (8) = 10 DET's.

Complejidad EO (Orders) = BAJA(1 FTR, 10 DET's).

Entrada Externa EI (Cancelar Orden).

Debido a que esta función realiza el mantenimiento de un fichero lógico interno se trata de una entrada externa.

FTR's = Ficheros lógicos mantenidos(Orders) = 1 FTR.

DET's = Campos de entrada(orderId 1) + Botón (1) + Todos los posibles mensajes de error (1) + Campos de salida que no se encuentren en la entrada (1 status = canceled +2 adquiredmoney, adquiredamount) = 6 DET's.

Complejidad EI (Cancelar Orden) = BAJA (1 FTR, 6 DET's).

Consulta Externa EQ (Purchases).

Esta función consiste en mostrar al usuario las diferentes compras que ha realizado en la tienda a lo largo del tiempo, debido a que todas los items en la tienda pueden ser comprados o bien por dinero real o por monedas de platino(cuyo valor de almacenamiento y salida no requieren cálculos adicionales) se considerará una entrada externa.

FTR's = Ficheros lógicos leídos (1) = 1 FTR.

DET's = Todos los posibles mensajes de error(0) + Botón(1) + Campos de entrada(userId) + Campos de salida (5) = 7 DET's.

Complejidad Consulta Externa (Purchases) = BAJA (1 FTR, 7 DET's).

==========

Subastas

Archivo Lógico Interno (Order).

RET's = Número de subgrupos presentes en la tabla = 1 RET's.

DET's = Número total de campos(9) – Campos creados por técnicas de implementación y no reconocibles por el usuario(1 orderId) + Claves foráneas (2) = 10 DET's.

Complejidad ILF (Order) = BAJA(1 RET's, 10 DET's).

Archivo Lógico Interno (AuctionStats).

RET's = Número de subgrupos en la tabla(1 por defecto) = 1RET's.

DET's = Número total de campos(4) – Campos creados por técnicas de implementación y no reconocibles por el usuario(1 auctionStatId) + Cláves foráneas (0) = 3 DET's.

Complejidad ILF(AuctionStats) = BAJA(1 RET, 3 DET's).

Gráfica de subastas (EO).

Esta funcionalidad se considera una consulta externa puesto que los datos representados en la gráfica son la representación directa de los datos ya existentes en la base de datos sin realizar modificaciones. En una escala temporal ya determinada, pero la traducción en formato en monedas la convierte en una en una salida externa.

FTR's = ficheros lógicos leídos (1) = 1 FTR.

DET's = Leemos todas las AuctionStats con sus respectivos valores, su cantidad y la fecha en la que se han realizado para su representación = 3 DET's.

Complejidad EO (Subastas) = BAJA (1 FTR, 3 DET's).

Salida Externa EO (Precio actual de monedas premium en mercado (Compra)).

Debido a que la función consiste en recuperar el precio más bajo disponible en el mercado en el formado de monedas es una salida externa.

FTR's = Ficheros lógicos leídos (1) = 1 FTR.

DET's = Botón (1) + Posibles mensajes de error (0) + Campos de entrada (0) + Campos de salida (1) = 2 DET.

Complejidad EO (Precio actual de monedas premium en mercado(Compra)) = BAJA(1 FTR, 2 DET).

Salida Externa EO (Precio actual de monedas premium en mercado (Venta)).

Debido a que la función consiste en recuperar el precio más bajo disponible en el mercado en el formado de monedas es una salida externa.

FTR's = Ficheros lógicos leídos (1) = 1 FTR.

DET's = Botón (1) + Posibles mensajes de error (0) + Campos de entrada (0) + Campos de salida (1) = 2 DET.

Complejidad EO (Precio actual de monedas premium en mercado(Venta)) = BAJA(1 FTR, 2 DET).

Entrada Externa EI (Vender Monedas Platino).

Esta operación realiza el soporte de un ILF en la aplicación por ello es una entrada externa.

FTR's = Ficheros lógicos mantenidos(1) = 1 FTR.

DET's = Botón (1(Se cuenta unicamente 1 por todos los que haya en este caso hay un mensaje de verificación)) + mensajes de error(0) + campos de entrada (2) + campos de salida(0)* = 3 DET's.

***Esta operación implica crear nuevas ordenes que serán mostradas en el subsistema del usuario a través de la pestaña de ordenes, adicionalmente crearán las transacciones + tasas por vip(todo esto se recupera con las anteriores EQ/EO). Pero todo esto sucede por debajo y no cruza los límites de la aplicación así que no tiene campos de salida, lo mismo en compra.**

Complejidad Entrada Externa (Vender Monedas de Platino) = BAJA (1 FTR, 3 DET's).

Entrada Externa EI (Comprar Monedas Platino).

Esta operación realiza el soporte de un ILF en la aplicación por ello es una entrada externa.

FTR's = Ficheros lógicos mantenidos(1) = 1 FTR.

DET's = Botón (1(Se cuenta unicamente 1 por todos los que haya en este caso hay un mensaje de verificación)) + mensajes de error(0) + campos de entrada (2) + campos de salida(0) = 3 DET's.

Complejidad Entrada Externa (Comprar Monedas de Platino) = BAJA (1 FTR, 3 DET's).

==========

Tienda

Archivo Lógico Interno (Category).

RET's = 0 => 1 Por defecto si no hay subgrupos = 1 RET.

DET's = Número total de campos (3) – Campos existentes únicamente por técnicas de implementación no reconocidos por el usuario (1) + Cláves foráneas (0) = 2 DET's.

Complejidad ILF (Category) = BAJA (1 RET, 2 DET's).

Archivo Lógico Interno (Product).

RET's = 2 RET's debido a que es opcional tener un precio por monedas de platino o euros.

DET's = Número total de campos (8) – Campos por técnicas de implementación no reconocibles por el usuario(3) + Cláve foránea (1) = 6 DET's.

Complejidad ILF (Product) = BAJA (2 RET's, 6 DET's).

Archivo Lógico Interno (Purchase).

RET's = Existen dos subgrupos donde en uno es opcional las monedas de platino = 2 RET's.

DET's = Campos totales (6) – Campos por técnicas de implementación no reconocibles por el usuario(1) + Cláve foránea (1) = 6 DET's.

Complejidad ILF (Purchase) = BAJA(2 RET's, 6 DET's).

Archivo Lógico Interno (ShoppingCart).

Debido a que un ShoppingCart (1 RET, 1 DET) y ShoppingCartItem son lógicamente complementarios en este contexto y shoopingCart solo existe como paso intermedio se contarán como un solo ILF.

RET's = 1 RET.

DET's = Cláves foráneas (3) + Campos (3) – Campos que existen por detalles de implementación (2) = 4 DET's.

Complejidad ILF (ShoppingCart) = BAJA (1 RET, 4 DET's).

Consulta Externa EQ (Obtener las Categorías).

Esta función es una consulta externa debido a que no existe la necesidad de realizar cálculos derivados.

FTR's = Ficheros leídos = 1 FTR.
DET's = Botón (1) + Campos de entrada (0) + Campos de salida (2) = 3 DET's.

Complejidad EQ (Obtener las Categorías) = BAJA (1 FTR, 3 DET's).

Entrada Externa EI (Añadir producto).

Debido a que es una incorporación de datos al sistema y mantiene un ILF se trata de una entrada externa.

FTR's = Ficheros mantenidos = 1 FTR.

DET's = Número de campos totales(8) + Botón(1) + Mensajes de error(1) = 10 DET's.

Complejidad EI (Añadir producto) = BAJA(1 FTR, 10 DET's).

Consulta Externa EQ (Buscar por categoría).

Esta función es una consulta externa debido a que no existe la necesidad de realizar cálculos derivados.

FTR's = Ficheros leídos = 2 FTR.

DET's = Botón (1) + Campos de entrada (1 categoryId) + Campos de salida (4) = 6 DET's.

Complejidad EQ (Buscar por categoría) = MEDIA (2 FTR, 6 DET's).

Entrada Externa EI (Editar Producto).

Esta función es una entrada externa ya que realiza el mantenimiento de un ILF.

FTR's = Ficheros mantenidos = 1 FTR.

DET's = Botón (1) + Campos de entrada (6) + mensajes de error (1) = 8 DET's.

Complejidad EI (Editar Producto) = BAJA (1 FTR, 8 DET's).

Entrada Externa EI (Modificar descuento por categoría).

Esta función es una entrada externa ya que realiza el mantenimiento de un ILF.

FTR's = Ficheros mantenidos = 1 FTR.

DET's = Botón (1) + Campos de entrada (2) + mensajes de error (1) + campos de salida(el descuento modificado y el precio (2)) = 6 DET's.

Complejidad EI (Modificar descuento por categoría) = BAJA (1 FTR, 6 DET's).

Consulta Externa EQ (Buscar detalles del producto)

Se trata de una consulta externa debido a que se muestra la descripción tal cual del producto.

FTR's = Ficheros leídos = 1 FTR .
DET's = Botón (1) + Campos de entrada (productId) + mensajes de error (1) + campos de salida no repetidos (1) = 4 DET's.

Complejidad EQ (Buscar detalles del producto) = BAJA (1 FTR, 4 DET's).

Entrada Externa EI (Añadir Item al carrito).

Esta función es una entrada externa ya que realiza el mantenimiento de un ILF.

FTR's = Ficheros mantenidos = 1 FTR.

DET's = Botón (1) + Campos de entrada (1 productId) + mensajes de error (1) = 3DET's.

Complejidad EI (Añadir Item al carrito) = BAJA (1 FTR, 3 DET's).

Entrada Externa EI (Eliminar Item del carrito).

Esta función es una entrada externa ya que realiza el mantenimiento de un ILF.

FTR's = Ficheros mantenidos = 1 FTR.

DET's = Botón (1) + Campos de entrada (1 productId) + mensajes de error (1) = 3DET's.

Complejidad EI (Eliminar item del carrito) = BAJA (1 FTR, 3 DET's).

Entrada Externa EI (Modificar número de items del carrito).

Esta función es una entrada externa ya que realiza el mantenimiento de un ILF.

FTR's = Ficheros mantenidos = 1 FTR.

DET's = Botón (1) + Campos de entrada (2) + mensajes de error (1) = 4DET's.

Complejidad EI (Modificar número de items del carrito) = BAJA (1 FTR, 4 DET's).

Salida Externa EO (Consultar items del carrito).

Esta función es una salida externa puesto que se muestra la información del carrito e información derivada como el precio total y el número de items.

FTR's = Ficheros leídos = 2 FTR's.

DET's = Botón (1) + Campos de salida (5) + mensajes de error (1) = 7DET's.

Complejidad EO (Consultar items del carrito) = MEDIA (2 FTR, 7 DET's).

Entrada Externa EI (Realizar Compra).

Esta función es una entrada externa debido a que realiza el mantenimiento de varios ILF's de la aplicación.

FTR's = Ficheros mantenidos = 2 FTR's.

DET's = Botón(1) + Campos de entrada(cantidad e id de producto, 2) + campos de salida(solo un mensaje diciendo que la compra se ha realizado con éxito, 0) + todos los posibles mensajes de error(1) = 4 DET's.

Complejidad EI (Realizar Compra) = BAJA (2 FTR's, 4 DET's).

==========

Foro

Archivo lógico interno (Topic).

RET's = Número de subgrupos (0) = Por defecto 1 RET.

DET's = Todos los campos (5) + Cláves foráneas(2) – Campos por técnicas de implementación y no visibles por el usuario(1) = 6 DET's.

Complejidad ILF (Topic) = BAJA (1 RET's, 6 DET's).

Archivo lógico interno (Topic Category).

RET's = Número de subgrupos (0) = Por defecto 1 RET.

DET's = Todos los campos (2) + Cláves foráneas(0) – Campos por técnicas de implementación y no visibles por el usuario(1) = 1 DET's.

Complejidad ILF (Topic Category) = BAJA (1 RET's, 1 DET's).

Archivo lógico interno (Reply).

RET's = Número de subgrupos (0) = Por defecto 1 RET.

DET's = Todos los campos (5) + Cláves foráneas(2) – Campos por técnicas de implementación y no visibles por el usuario(1) = 6 DET's.

Complejidad ILF (Reply) = BAJA (1 RET's, 6 DET's).

Consulta Externa EQ (Buscar Topic category).

Se trata de una consulta externa puesto que no implica realizar ningún cálculo adicional.

FTR's = Ficheros leídos = 1 FTR's.

DET's = Botón (1) + Campos de salida (1) + posibles mensajes de error(0) = 2 DET's .

Complejidad EQ (Buscar Topic category) = BAJA (1 FTR, 2 DET's).

Consulta Externa EQ (Buscar Topic).

Se trata de una consulta externa puesto que no implica realizar ningún cálculo adicional al realizar la busqueda.

FTR's = Ficheros leídos = 3 FTR's.

DET's = Botón (1) + Campos de salida (5) + Campos de entrada (2) +posibles mensajes de error(0) = 8 DET's .

Complejidad EQ (Buscar Topic) = MEDIA (3 FTR's, 8 DET's).

Consulta Externa EQ (Detalles de Topic).

Esta funcionalidad se trata de una consulta externa puesto que no tiene la necesidad de realizar cálculos internos.

FTR's = Ficheros leídos = 3 FTR's.

DET's = Botón (1) + Campos de salida (7) + Campos de entrada (1 ID del topic) +posibles mensajes de error(0) = 9 DET's .

Complejidad EQ (Detalles de Topic) = MEDIA (3 FTR's, 9 DET's).

Entrada Externa EI (Edit Reply).

Debido a que esta funcionalidad realiza el soporte de un ILF de la aplicación se trata de una Entrada Externa.

FTR's = Ficheros Mantenidos = 1 FTR's.

DET's = Botón (1) + Campos de entrada (id de reply + nuevo contenido, 2) + posibles mensajes de error(1) + Campos de salida (0) = 4 DET's .

Complejidad EI (Edit Reply) = BAJA (1 FTR's, 4 DET's).

Entrada Externa EI (Crear Topic).

Debido a que esta funcionalidad realiza el soporte de un ILF de la aplicación se trata de una Entrada Externa.

FTR's = Ficheros Mantenidos = 1 FTR's.

DET's = Botón (1) + Campos de entrada (3) + Posibles mensajes de error (1) = 5 DET's .

Complejidad EI (Crear Topic) = BAJA (1 FTR's, 5 DET's).

Entrada Externa EI (Crear Reply).

Debido a que esta funcionalidad realiza el soporte de un ILF de la aplicación se trata de una Entrada Externa.

FTR's = Ficheros Mantenidos = 1 FTR's.

DET's = Botón (1) + Campos de entrada (Contenido de reply, 1) = 2 DET's .

Complejidad EI (Crear Reply) = BAJA (1 FTR's, 2 DET's).

Entrada Externa EI (Borrar Reply).

Debido a que esta funcionalidad realiza el soporte de un ILF de la aplicación se trata de una Entrada Externa.

FTR's = Ficheros Mantenidos = 1 FTR's.

DET's = Botón (1) + Campos de entrada (id de reply) = 2 DET's .

Complejidad EI (Borrar Reply) = BAJA (1 FTR's, 2 DET's).

==========

4. Estimación Puntos Función.

PFSA => Puntos Función Sin Ajustar.

PFSA (Foro) = 3 ILF's (Complejidad BAJA) * 7 + 2 EQ's (Complejidad MEDIA) * 4 + 1 EQ (Complejidad BAJA) * 3 + 3 EI (Complejidad BAJA) * 4 = 52 PFSA.

PFSA (Tienda) = 4 ILF's (Complejidad BAJA) * 7 + 2 EQ's (Complejidad BAJA) * 3 + 1 EQ (Complejidad MEDIA) * 4 + 7 EI (Complejidad BAJA) * 3 + 1 EO (Complejidad MEDIA) * 5 = 64 PFSA.

PFSA (Subastas) = 2 ILF (Complejidad BAJA) * 7 + 3 EO's (Complejidad BAJA) * 4 + 2 EI (Complejidad BAJA) * 3 = 32 PFSA.

PFSA (Usuario) = 1 ILF (Complejidad BAJA) * 7 + 1 EO (Complejidad MEDIA) * 5 + 2 EO's (Complejidad BAJA) * 4 + 1 EI (Complejidad BAJA) * 3 + 1 EQ (Complejidad BAJA) *3 = 26 PFSA.

PFSA (Totales) = 52 + 64 + 32 + 26 = 174 PFSA.

==========

5. Resultados de la estimación con los PFSA.

El resultado obtenido por el apartado 4 son: 174 PFSA.

Ahora aplicaremos las métricas del ISBSG [21].

Especificación de los parámetros del proyecto:

- **Alcance** => Debido a que se trata de un proyecto que se ha tenido que evaluar desde cero y es nuevo, el alcance es de **Desarrollo**.

- **Plataforma** => El proceso lógico de esta aplicación sería:

Estamos ante una aplicación que va a ser ejecutada en diferentes dispositivos, en sus respectivos navegadores, en cualquier país, por ello tenemos que garantizar que cualquier usuario pueda tener su dispositivo donde pueda ver la información del sistema y realice sus operaciones. Por ello:

- Un usuario en España lanza una operación cualquiera(venta platino) => Se actualizan los valores de este usuario y los valores de los demás usuarios involucrados, realizándose esta operación n veces, para múltiples usuarios, causando que la plataforma sea de **(Mid-Range)**.

- **Lenguaje** => Tenemos oracle, porque tenemos el E-R creado en SQL para realizar todas las consultas, pero la capa de servicios, a la hora de representar la información tendremos que utilizar otro lenguaje de programación. En este caso el lenguaje de desarrollo Java, por ello estamos ante un lenguaje de 3 Generación.

Cálculos de los diferentes valores del proyecto.

Esfuerzo del proyecto = C*(PFSA^E)

Parámetros: 3GL, Mid-Range y Desarrollo. Con la tabla proporcionada por el ISBSG =>
C= 42,48 E=0,744

Por ello: 42,48*(174^0,744) = 1973,11 horas/hombre.

Duración del proyecto = 0,370*Esfuerzo^(0,328)

Parámetros: 3GL, Mid-Range y Desarrollo.

Por ello: 0,370*1973,11^0,328 = 4,45 meses.

Personas necesarias para el proyecto.

1 persona trabaja 20 días al mes y 8h al día =>

Personas = Esfuerzo/(Duración * 20 * 8) = 2,77 personas => 3 Personas. Aprox.

Productividad.

Productividad = Esfuerzo / PF = 1973,11/174 = 11,33 Horas por PFSA.

Velocidad a la que se debe realizar la entrega.

Velocidad entrega = PF / Duración = 174/4,45 = 39,1 PF por mes.

Estimación del esfuerzo en las diferentes etapas del proyecto.

Según el ISBSG, el coste de implementación de las diferentes etapas del proyecto se pueden dividir en:

Planificación	Especificación	Diseño	Construcción	Pruebas	Implantación
9%	11%	15%	43%	16%	6%

Por ello, el **Esfuerzo por fases** con 174 PFSA y Esfuerzo Total 1973,11 horas/hombre :

Planificación = 0,09 * Esfuerzo Total = 177, 57 horas/hombre.
Especificación = 0,11 * Esfuerzo Total = 217,04 horas/hombre.
Diseño = 0,15 * Esfuerzo Total = 295,96 horas/hombre.
Construcción = 0,43 * Esfuerzo Total = 848,43 horas/hombre.
Pruebas =0,16 * Esfuerzo Total = 315,69 horas/hombre.
Implantación = 0,06 * Esfuerzo Total = 118,38 horas/hombre.

Duración en meses por fases :

Planificación = 0,09 * Duración Total = 0,4 meses = 64h
Especificación = 0,11 * Duración Total = 0,49 meses = 78,4h
Diseño = 0,15 * Duración Total = 0,67 meses = 107,2h
Construcción = 0,43 * Duración Total = 1,91 meses = 305,6 h
Pruebas =0,16 * Duración Total = 0,712 meses = 113,92 h
Implantación = 0,06 * Duración Total = 0,267 meses = 42,72 h

==========

6. Coste Final del Proyecto.

Un programador/analista cobra de media 15,625€ (Según google). Y se calcula según el ISBSG como:

Coste = Esfuerzo * CosteMedio = 1973,11 * 15,625 = 30829,84 € .

Bibliografía

[1] Documentación sobre Mysql. Disponible en: MySQL

[2] Documentación sobre Spring. Disponible en: Spring Framework

[3] Documentación acerca de Hibernate. Disponible en: Hibernate. Everything data.

[4] Documentación sobre Maven. Disponible en: Maven – Welcome to Apache Maven

[5] Documentación acerca de JUnit. Disponible en: JUnit 5

[6] Información adicional de Pitest. Disponible en: PIT Mutation Testing (pitest.org)

[7] Información adicional de ArchUnit.
Disponible en: Unit test your Java architecture - ArchUnit

[8] Documentación sobre Mockito. Disponible en: Mockito framework site

[9] Documentación sobre Jacoco.
Disponible en: EclEmma - JaCoCo Java Code Coverage Library

[10] Información sobre el plugin Drakyen's Time Tracker. Disponible en:
Darkyen's Time Tracker - IntelliJ IDEs Plugin | Marketplace (jetbrains.com)

[11] Información complementaria sobre React. Disponible en:
React – Una biblioteca de JavaScript para construir interfaces de usuario (reactjs.org)

[12] Información acerda de Redux. Disponible en:
Redux - A predictable state container for JavaScript apps. | Redux

[13] Material adicional sobre Plotly. Disponible en: plotly · PyPI

[14] Información adicional acerca de Pencil. Disponible en:
Home - Pencil Project (evolus.vn)

[15] Información adicional sobre el software Dia. Disponible en:
Dia Diagram Editor download | SourceForge.net

[16] Documentación sobre Balsamiq. Disponible en: Mockups 3 for Desktop | Balsamiq

[17] Documentación sobre SonarQube. Disponible en:
Code Quality and Code Security | SonarQube

[18] Información adicionar acerca de Github. Disponible en: GitHub

[19] Información adicional sobre Intellij. Disponible en:
IntelliJ IDEA: The Capable & Ergonomic Java IDE by JetBrains

[20] Documentación sobre el IFPUG. Disponible en:
IFPUG – International Function Points Users Group – IFPUG

[21] Documentación sobre el ISGSG. Disponible en: Home - ISBSG