Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Gamificación de un Sistema de Ejercicios Auto-Corregibles de Programación en Java

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Manuel Gregorio

Tutor: Josep Silva Galiana

Curso 2020/2021

Resumen

ASys es un sistema semiautomático de evaluación de ejercicios y exámenes centrado principalmente en el lenguaje Java. El objetivo de esta plataforma es motivar, contribuir y maximizar el aprendizaje de los estudiantes universitarios en lenguajes de programación como el recién mencionado. Para ello, es imprescindible que la aplicación disponga de mecanismos de gamificación que mantengan vivas las ganas de los estudiantes de aprender y de utilizar la plataforma.

Este trabajo analiza los diferentes métodos gamificables existentes con el fin de implementarlos en ASys. A partir de dicho análisis, se ha desarrollado un sistema de monetización interno de la aplicación. Para ello, la realización de ejercicios en la plataforma se premia con monedas que los usuarios pueden gastar en una nueva pantalla que actúa como tienda interna de la aplicación. En esta tienda se pueden comprar diversos objetos, tanto estéticos como de utilidad como pistas en ejercicios. En suma, los objetos comprados por el usuario se pueden visualizar en una pantalla nueva que actúa como un inventario.

El presente documento comienza con un extenso análisis de los distintos métodos de gamificación y prosigue con la definición de los requisitos y funcionalidades extraídas a partir de este análisis. Seguidamente se detalla su diseño e implementación, y termina con una sección que reúne todas las pruebas realizadas sobre la solución final. Durante todo el desarrollo se han empleado las tecnologías que se han utilizado hasta ahora en ASys, es decir, Vue (Javascript, HTML y CSS) para el cliente y Spring Boot (Java) en el servidor.

Palabras clave: ASys, Spring, Vue, HTML, Javascript, CSS, programación, monetización, gamificación, sistema de economía.

Abstract

ASys is a semi-automatic assessment system for exercises and exams mainly focused on the Java language. The objective of this platform is to motivate, contribute and maximize the learning of university students in programming languages like the one just mentioned. For this, it is essential that the application has gamification mechanisms that keep the desire of students to learn and use the platform.

This work analyzes the different existing gamifiable methods in order to implement them in ASys. Based on this analysis, an internal monetization system for the application has been developed. To do this, carrying out exercises on the platform is rewarded with coins that users can spend on a new screen that acts as an internal store for the application. In this store users can buy various objects, both aesthetic and useful, as exercise clues. In sum, once the objects purchased by the user can be displayed on a new screen that acts as an inventory.

This document begins with an extensive analysis of the different gamification methods and continues with the definition of the requirements and functionalities extracted from this analysis. Next, its design and implementation are detailed, and ends with a section that gathers all the tests carried out on the final solution. Throughout the development, the technologies that have been used so far in ASys have been used, that is, Vue (Javascript, HTML and CSS) for the client and Spring Boot (Java) for the server.

Keywords: ASys, Spring, Vue, HTML, Javascript, CSS, programming, monetization, gamification, economy system.

Índice de contenidos

Índice de Figuras

Índice de Tablas

1 Introducción

1.1 Motivación

Hace pocos meses, me surgió la oportunidad de participar en un prometedor proyecto que pretende mejorar la enseñanza de la programación de los estudiantes universitarios. Esto lo consigue mediante la sugerencia de ejercicios que se adecúan al nivel de cada usuario, cuya corrección pretende ser más justa y exacta que los métodos actuales. Entre las posibilidades que me ofrecieron dentro de este proyecto, decidí introducir y mejorar el sistema de gamificación de la aplicación. Afronto este proyecto con la motivación de saber que tengo la posibilidad de aportar mi granito de arena en un sistema que mis compañeros y yo habríamos deseado tener durante todos estos años y, aunque no podamos aprovecharla al máximo nosotros, las futuras generaciones sí que podrán.

1.2 Objetivos

Los objetivos de este TFG son, por un lado, establecer los fundamentos para la implementación de un sistema de monetización en la aplicación ASys como pilar principal de la gamificación y, por otro lado, investigar y determinar qué otros mecanismos de gamificación pueden utilizarse para complementar la monetización de la plataforma. Principalmente, será necesario (1) incorporar un sistema de monedas como recompensa y (2) múltiples productos canjeables por ellas. Estos productos podrán ser tanto estéticos (fotos de perfil, por ejemplo) como funcionales (por ejemplo, pistas en ejercicios) y (3) necesitarán de una tienda para ser comprados y de (4) un inventario para ser almacenados y utilizados. Además, será importante que (5) exista un equilibrio entre el precio de los artículos y la cantidad de monedas ofrecidas como recompensas, siendo esta última mayor en caso de obtener mejores hitos.

Para determinar cómo recompensar a los usuarios y qué elementos ofrecer para canjear, será necesario un extenso análisis de los distintos métodos de gamificación existentes con el fin de (6) extraer las mejores ideas de los sistemas actuales, inventar las propias e integrarlo todo en nuestra aplicación.

1.3 Impacto esperado

Tal como se ha comentado y adelantado en la motivación, ASys es un sistema dedicado a la enseñanza de la programación, donde los usuarios realizan ejercicios acordes a su nivel.

Se espera que la implementación de este sistema de monetización contribuya a la motivación de los usuarios, mejorando su experiencia en la plataforma e incrementando así su

utilización. Además, este nuevo mecanismo de gamificación pretende ser un incentivo para que los estudiantes intenten mejorar las notas obtenidas en los ejercicios. Todo esto se traduce en incrementar la práctica y el aprendizaje de Java, Haskell y futuros lenguajes de programación que se añadan a la plataforma.

Finalmente, la investigación y análisis de diferentes mecanismos de gamificación será de utilidad para implementar nuevos conceptos e ideas en la plataforma en futuros proyectos, los cuales podrán ser compatibles con el sistema de monetización que se desarrollará en este trabajo.

1.4 Metodología

La metodología llevada a cabo en este TFG se ha dividido en tres grandes fases bien diferenciadas. La primera consistió en una extensa formación de las tecnologías utilizadas en ASys. La segunda fase se basó en analizar de forma amplia diferentes aplicaciones que utilizan métodos de gamificación, así como una serie de estudios donde también se ponen en práctica estos mecanismos. De estas ideas, en la tercera y última fase, se determinaron los requisitos a cumplir en el proyecto, se diseñaron y se desarrolló la solución.

Durante la fase de formación, cuya duración alcanza los dos meses, nos centramos en aprender y comprender las tecnologías que utiliza la versión de ASys actual. Para ello, se siguieron una serie de cursos online que se acompañaron con diversos ejercicios de programación.

En la segunda fase del proyecto se analizaron hasta 10 aplicaciones diferentes; la mayoría relacionadas con el aprendizaje de la programación, con el objetivo de determinar los diversos mecanismos de gamificación existentes.

Por último, la tercera fase de este proyecto se llevó a cabo siguiendo un método de trabajo similar a la metodología iterativa [1]. En primer lugar, se especificó un conjunto de requisitos y se diseñó una primera versión a desarrollar que se validó mediante la técnica de prototipado. Durante el desarrollo, se llevaron a cabo una serie de reuniones para evaluar el trabajo realizado hasta el momento, fijar prioridades en las tareas y añadir posibles nuevos requisitos o modificar alguno de los existentes. Finalmente, una vez acabada la implementación, se realizó una batería de pruebas para verificar y validar el correcto funcionamiento del desarrollo de la solución.

1.5 Estructura

La memoria de este trabajo de fin de grado se divide en una serie de epígrafes que explicaremos brevemente a continuación:

En primer lugar, tras la introducción, pasaremos a analizar el estado del arte, estudiando e investigando diversas aplicaciones que poseen sus propios sistemas de gamificación. Tras este análisis, determinaremos los diferentes elementos que se han de implementar en ASys, tanto en este proyecto como en el futuro, para poseer un buen sistema de gamificación basado en la monetización.

Seguidamente, realizaremos una especificación de requisitos con las funcionalidades primordiales que ha de tener el sistema de monetización a implementar. Una vez determinadas las funcionalidades, realizaremos una estimación inicial del esfuerzo y coste de trabajo a realizar.

En tercer lugar, analizaremos el problema que se nos plantea, escogiendo las tecnologías más adecuadas y las funciones y librerías que estas poseen.

A continuación, llevaremos a cabo la fase de diseño, donde determinaremos la arquitectura del proyecto. En este epígrafe, además, detallaremos el modelo de datos, las interfaces a desarrollar, así como el funcionamiento de los objetos que se van a vender en la tienda. Por último, explicaremos el sistema de economía y de precios que tendrá ASys y mostraremos los iconos que se diseñaron para los diferentes objetos.

Tras el diseño, pasaremos a explicar cómo se ha desarrollado la solución propuesta, explicando aquellas partes más relevantes y mostrando el resultado final de la aplicación. En este epígrafe también se comentarán los diferentes problemas encontrados durante la implementación.

En penúltimo lugar, explicaremos cómo implantar el sistema en un entorno de explotación. Aquí se detallarán las configuraciones e instalaciones previas a la puesta en marcha de ASys.

Finalmente, pasaremos a mostrar todas las pruebas llevadas a cabo para verificar el correcto funcionamiento de la aplicación, seguidas de unas pruebas de validación realizadas por usuarios externos. En este epígrafe también mostraremos una serie de pruebas de calidad de código y accesibilidad aplicadas durante todo el desarrollo.

Para concluir el trabajo, sintetizaremos las ideas generales del proyecto implementado y comentaremos si hemos logrado o no los objetivos propuestos en esta introducción. También se compararán las estimaciones de esfuerzo y presupuesto realizadas al comienzo con los valores reales obtenidos. Por último, se detallarán los posibles trabajos futuros que se pueden llevar a cabo en ASys para mejorar aún más el sistema de monetización desarrollado.

2 Estado del arte

En el presente apartado se analizan una selección de aplicaciones del mercado que se asimilan a la nuestra (en términos de aprendizaje de programación a través de la resolución de ejercicios). Nos interesa, además, que estas aplicaciones presenten un mecanismo de gamificación y/o de monetización con monedas virtuales. Puesto que este último mecanismo, orientado a la ganancia de monedas como incentivo, es muy utilizado y desarrollado en múltiples dominios, se analizarán también algunas de estas aplicaciones.

2.1 Codewars

Codewars[1] es un sitio web en el que los usuarios pueden entrenar y mejorar sus habilidades de programación [2] mediante, principalmente, la realización de lo que se denomina "Kata", ejercicios de programación creados por la propia comunidad [3]. En esta web, existen dos métodos diferentes de gamificación: Los rangos y el honor [4].

Los rangos se utilizan para indicar la competencia de los usuarios y, a medida que completen "Katas" obtendrán puntos los cuales serán necesarios para subir de rango. La cantidad de puntos obtenida por cada "Kata" que se complete, varía en función de la dificultad de esta y del rango del usuario [5]. La dificultad de los "Kata" es determinada por la comunidad [6] y se clasifica también mediante los rangos, pero con una escala diferente. En lo que se refiere a los usuarios, la escala de rangos se divide en rangos de nivel maestro (también llamados "Dan") y en rangos "Kyu". Al comienzo, todo usuario empieza en el rango "8 kyu" (hay 8 rangos "Kyu", siendo "1 kyu" el más alto) y su objetivo es completar ejercicios para poder alcanzar los rangos "Dan", cuyo nivel máximo no queda muy claro en la documentación, pues en la sección sobre los rangos [5] existen 8 rangos "Dan" (siendo 8 el más elevado) pero, en la sección *about* [7], hay un apartado visible solo para aquellos usuarios que han iniciado sesión que indica que existe hasta un nivel 4 (ver Figura 1). Sin embargo, en otro apartado de su documentación, siempre referente a los rangos [8], existe una tabla (ver Tabla 1) que indica los puntos necesarios para subir de rango y esta llega hasta "2 dan". Además, si analizamos a los mejores 500 usuarios de la web [9], el máximo rango que podemos encontrar es "2 dan". Como

[1] https://www.codewars.com

contrapartida, en la sección relacionada con las recompensas de honor (tema tratado más adelante), hay premios hasta alcanzar el rango “6 dan” (ver Tabla 2).

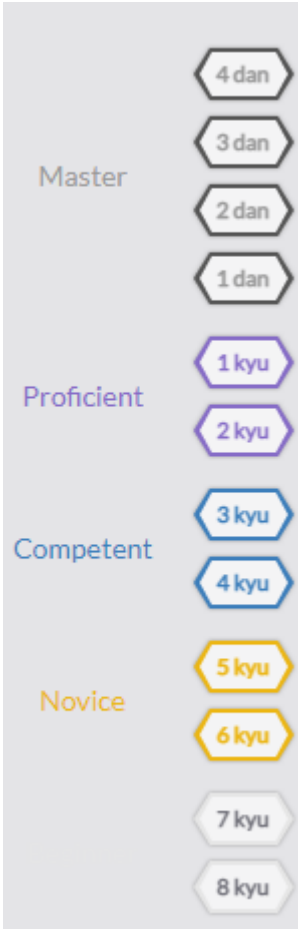


Figura 1. Rangos que pueden tener los usuarios (cuyo máximo rango es "4 dan") según la sección about de Codewars [7].

Tabla 1. Puntuación requerida para subir de rango. Los datos llegan hasta el rango "2 dan" [8].

User's Rank Requirements

User Rank	Required Score
8 kyu	0
7 kyu	20
6 kyu	76
5 kyu	229
4 kyu	643
3 kyu	1,768
2 kyu	4,829
1 kyu	13,147
1 dan	35,759
2 dan	97,225

Tabla 2. Puntos de honor conseguidos al subir de rango [10]. *Aquí los rangos llegan hasta "6 dan"*

Ranking up

Reached Rank	Honor
7 kyu	20
6 kyu	30
5 kyu	45
4 kyu	70
3 kyu	100
2 kyu	150
1 kyu	225
1 dan	450
2 dan	900
3 dan	1,800
4 dan	3,200
5 dan	6,400
6 dan	12,800

En cuanto a los rangos de los "Katas", estos oscilan entre el rango "8 kyu" (dificultad más sencilla) y el rango "1 kyu" (mayor dificultad) y, dependiendo de su rango, otorgarán una determinada puntuación base cuando se complete [8], tal como podemos ver en la Tabla 3. A esta puntuación base se le suma una cantidad porcentual en base al rango que tenga el usuario (ver Tabla 4). Por ejemplo, si un usuario de rango "7 kyu" completa un "kata" de rango "5 kyu" obtendrá 21 puntos más el 30% de 21, por lo que ganará 27,3 puntos. De esta manera, los usuarios que realicen "katas" de rango superior a su rango, serán mejor recompensados. Además, esto permite a aquellos nuevos usuarios que poseen ya un cierto nivel, alcanzar un rango más acorde a sus competencias con mayor rapidez.

Tabla 3. Puntuación base obtenida al completar un "kata" según su rango [8].

Kata Rank Rewards

Kata Rank	Score Awarded
8 kyu	2
7 kyu	3
6 kyu	8
5 kyu	21
4 kyu	55
3 kyu	149
2 kyu	404
1 kyu	1,097

Tabla 4. Porcentaje aproximado de puntuación extra conseguida por completar un "kata" en base al rango de este y al rango del usuario [5].

Kata rank - Your rank	%
...	...
+2	+30%
+1	+12%
0	+5%
-1	+1.7%
-2	+0.3%
-3	+0.09%
...	...

Otro aspecto interesante de los rangos es que cada usuario posee un rango diferente para cada lenguaje de programación en el que ha realizado “Katas” (ver Figura 2). Esto permite conocer las competencias que tiene el usuario en cada uno de estos lenguajes y en cuáles podría plantearse mejorar su nivel. La cantidad de puntos conseguida en cada lenguaje sigue la misma norma utilizada por los usuarios al completar un “Kata” (ver Tabla 3 y Tabla 4) pero con una pequeña diferencia: cuando completas un “Kata” en un determinado lenguaje, obtienes puntos tanto para alcanzar un nuevo rango (puntuación global), así como puntos para ese lenguaje concreto, pero si completas de nuevo ese mismo “Kata” en un lenguaje de programación diferente solo obtendrás puntos en ese nuevo lenguaje y no en tu puntuación global [5].

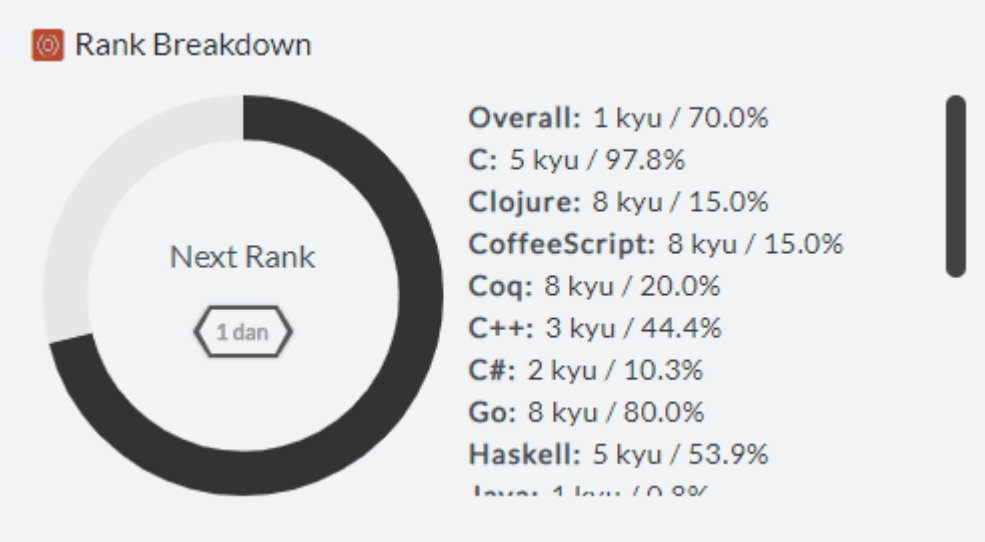


Figura 2. Rango global (el próximo) y propio de cada uno de los lenguajes practicados por el usuario [5].

El segundo método de gamificación consiste en la obtención de honor. El honor representa el nivel de respeto que el usuario se ha ganado de la comunidad gracias a sus habilidades y contribuciones. Para conseguirlo, existen múltiples formas, entre las cuales tenemos [11]:

- Completar katas (ver Tabla 5).
- Alcanzar nuevos rangos (ver Tabla 2).
- Publicar "Katas" en fase beta[2], la aprobación de tus "katas" o votos recibidos a tu "Kata" (los votos negativos restan). Esto se puede apreciar en la Tabla 6.
- Traducir "katas" a otros lenguajes de programación, votar "Katas" o votos recibidos en tus comentarios (los votos negativos restan). Esto y algunas acciones más se pueden observar en la Tabla 7.

[2] Fase en la que la comunidad asegura su calidad y determina su dificultad. Si todo está en orden, se publica el "kata" [6].

Tabla 5 Puntos de honor conseguidos al completar "katas" según su rango [10].

Solving kata

Level	Honor
White (8kyu & 7kyu) *Beginner*	2
Yellow (6kyu & 5kyu) *Novice*	8
Blue (4kyu & 3kyu) *Competent*	32
Purple (2kyu & 1kyu) *Proficient*	128
None *Beta*	4*

* Additional honor based on the level is rewarded once the kata is approved.

Tabla 6. Puntos de honor conseguidos publicando "katas", que te aprueben el tuyo o que lo voten [10].

Authoring Kata

Events on Authored Kata	Honor
Published for beta process	3
Approved as White (8kyu & 7kyu)	3
Approved as Yellow (6kyu & 5kyu)	15
Approved as Blue (4kyu & 3kyu)	75
Approved as Purple (2kyu & 1 kyu)	375
Received an up vote	2
Received a down vote	-2

Tabla 7. Puntos de honor conseguidos por diferentes acciones como traducir "Katas" a otros lenguajes de programación, votar "Katas", determinar su rango en fase beta o incluso votos recibidos a tus comentarios [10].

Other

Contributions	Honor	Description
Translation Approved (White)	4	
Translation Approved (Yellow)	16	
Translation Approved (Blue)	64	
Translation Approved (Purple)	256	
Assessed Beta Kata Rank	1	after a beta kata completion
Assessed Kata Satisfaction	1	after a kata completion
Published Kumite or Fork	2	fork a solution of yours or another user's, or create a new kumite
Kata Solution upvoted	1	your solution gets a 'best practices' or 'clever' upvote
Comment upvoted	1	
Comment downvoted	-1	
Referral Signup (first 5)	3	
Referral Signup (6+)	1	
Added GitHub Account	1	

Conseguir honor, además, te permite ganar ciertos privilegios como, por ejemplo, votar y crear “Katas”, determinar su rango, tener una influencia mayor para determinar si un “Kata” merece ser aprobado o incluso te permite poder aprobarlos (ver Tabla 8)[12].

Tabla 8. Puntos de honor requeridos para desbloquear ciertas acciones denominadas "privilegios" [12].

Privileges

Privilege	Required Honor	Description
Vote Kata	25	Vote on how satisfied you were with a kata
Mark comment as spoiler	50	Mark another's comment as a spoiler
Estimate ranking of own beta kata	75	Estimate on what rank you think your beta kata should be
Assess Rank	100	Vote on what rank you think a beta kata should be
Create Kata	300	Contribute your own kata to the community
Unmark comment as spoiler	500	Unmark another's comment as being a spoiler
Vote 2x power	1,000	Your vote counts 2x towards getting a kata out of beta
Vote 3x power	2,000	Your vote counts 3x towards getting a kata out of beta
Vote 4x power	3,000	Your vote counts 4x towards getting a kata out of beta
Coauthor Kata † / Approve Translation	4,000	Ability to edit other author's kata and approve translations
Resolve comment	5,000	Ability to resolve other's comments
Approve Kata	6,000	Ability to approve a beta kata and assign its rank
Edit locked test cases	10,000	Ability to edit test cases when they're locked

† You can edit a kata created by others if:

- You have contributed to the kata in the past (you do not need to have a coauthoring privilege in this case) **OR**
- You have been granted coauthoring privilege **and** at least one of the following are true:
  - The kata is in beta and waiting for approval
  - The kata is in beta and the author have allowed contribution
  - The kata is approved and unresolved issue/suggestion created over a week ago exists

2.2 Codesignal

Codesignal es una plataforma web que evalúa las habilidades de programación de los desarrolladores, los cuales pueden conseguir certificados, realizar ejercicios de programación de más de 70 lenguajes e incluso pueden entrenar test de programación que se realizan en entrevistas de trabajo de múltiples organizaciones [13].

Una vez iniciada una sesión en el sitio web, el usuario podrá realizar múltiples acciones gracias a una moneda virtual propia de la aplicación denominada “coin” (o “coins” en plural) cuya cantidad está visible en la esquina superior derecha en cualquier pantalla de la aplicación (ver Figura 3). Los “coins” se podrán conseguir realizando satisfactoriamente los diferentes ejercicios propuestos en muchos de los apartados de Codesignal. En la primera pantalla (ver Figura 4) aparecen una serie de recuadros bloqueados como los desafíos diarios (en el ejemplo ya se había desbloqueado) o los desafíos de las empresas que el usuario podrá comprar por 150 monedas (ver Figura 5) [14].

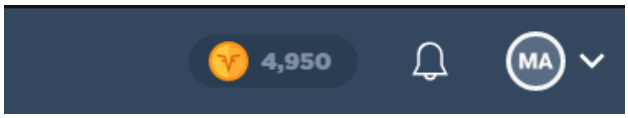


Figura 3. Cantidad de "coins" del usuario. Esto puede visualizarse en la esquina superior derecha de cualquier pantalla del sitio web.

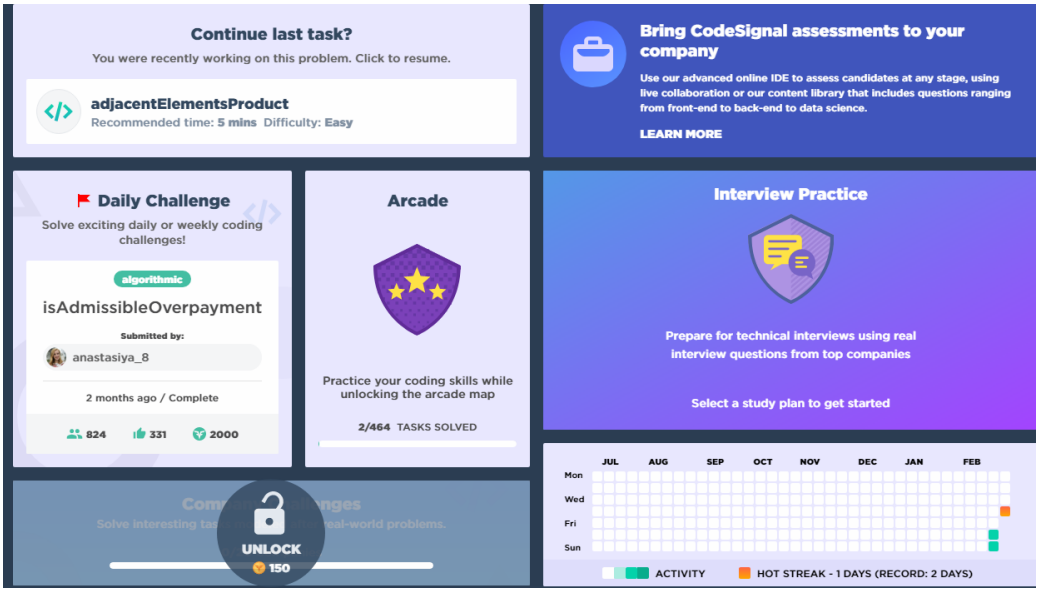


Figura 4. Primera pantalla que aparece al iniciar sesión en Codesignal [14].

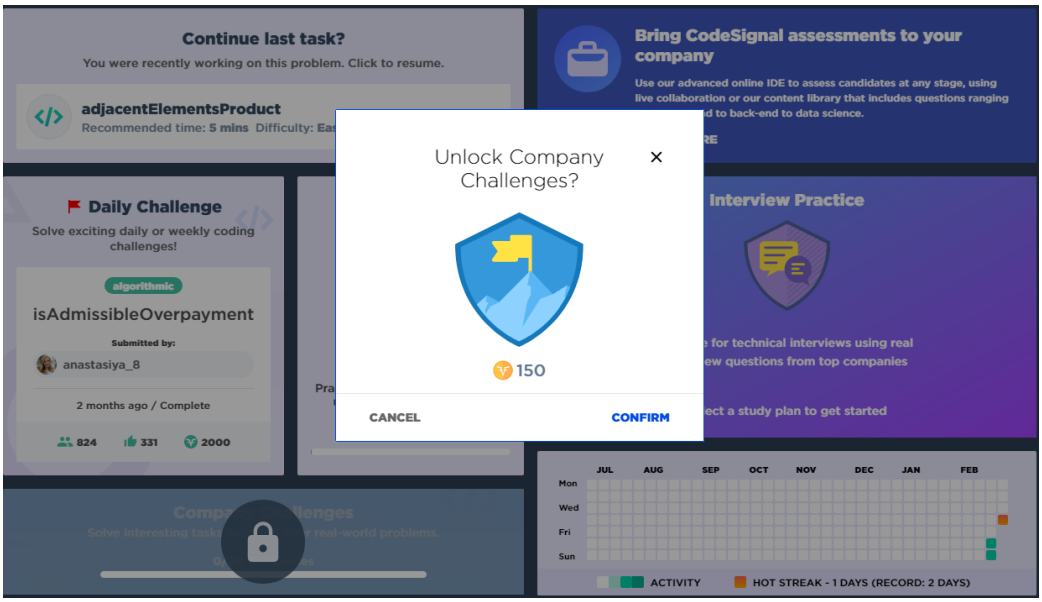


Figura 5. Pantalla de desbloqueo de los desafíos de organizaciones [14].

En la sección de Arcade [15] se encuentran una gran cantidad de ejercicios separados por secciones (ver Figura 6). Cada una de estas secciones se divide en lo que se denomina niveles. Cada nivel está compuesto por una serie de ejercicios, además de poseer un nombre y una ilustración que tienen el objetivo de aportar una pequeña historia (ver Figura 7). Tanto los ejercicios como los niveles son secuenciales, de manera que, para avanzar al siguiente es necesario haber superado el actual. Sin embargo, Codesignal te ofrece la oportunidad de desbloquear el nivel o ejercicio sucesivo (no permite desbloquear cualquiera) pagando una determinada cantidad de monedas (ver Figura 8 y Figura 9).

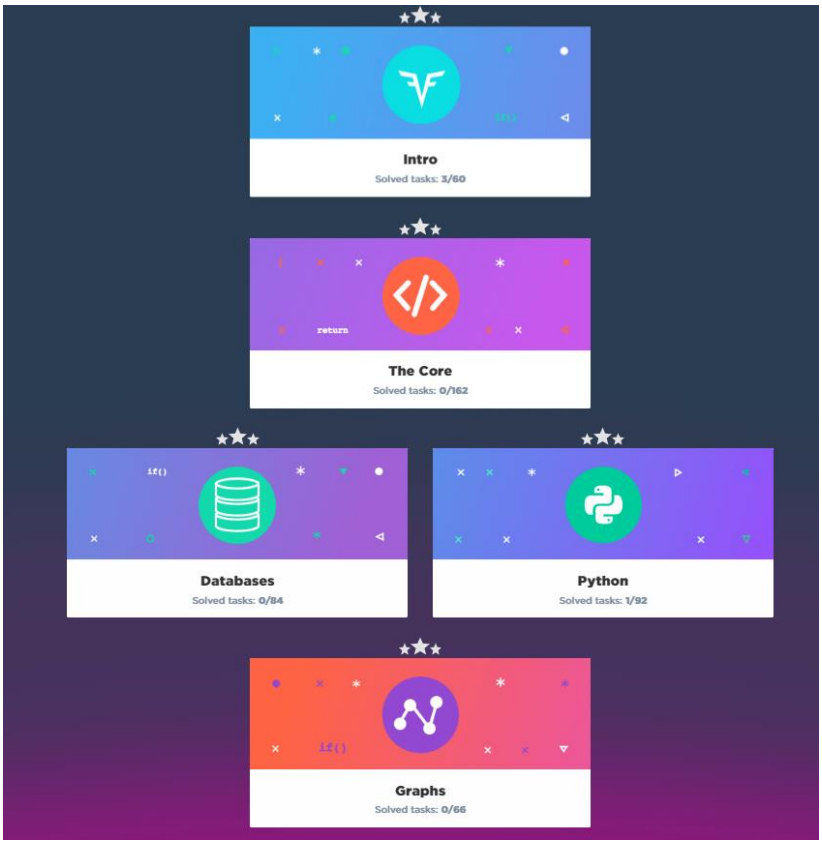


Figura 6. Diferentes secciones de los ejercicios del modo Arcade [15].

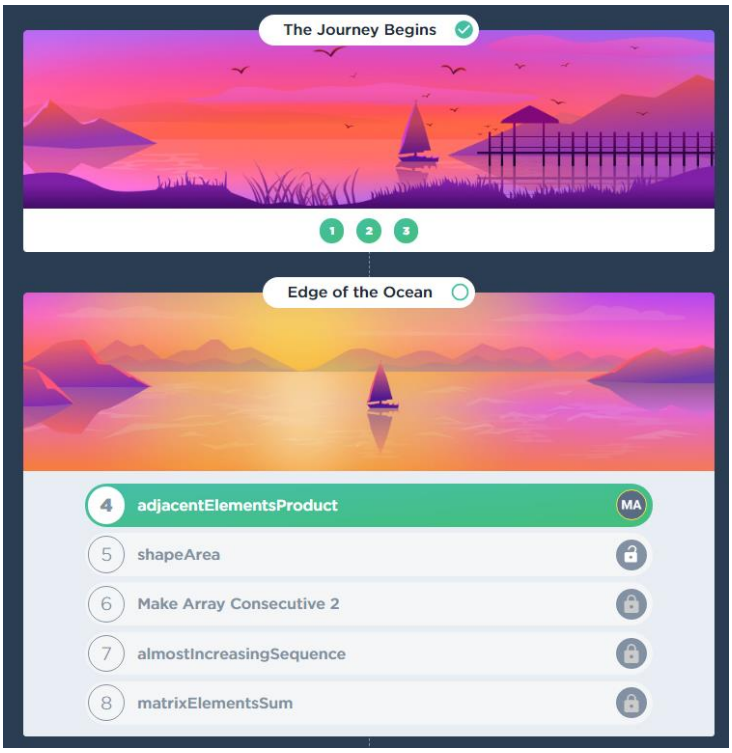


Figura 7. Primeros dos niveles de la sección "Intro". El primer nivel está completado y aparecen los ejercicios del nivel 2 [16].

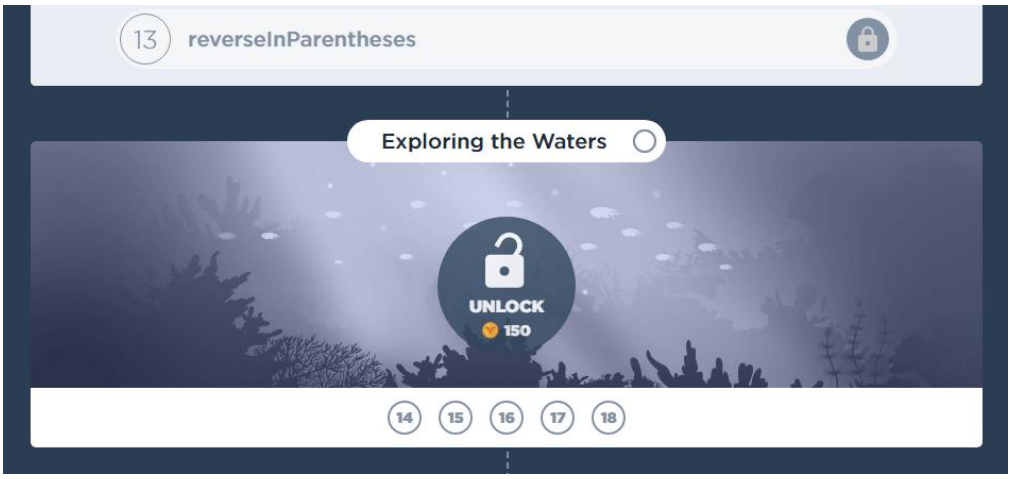


Figura 8. Cantidad de "coins" necesarias para desbloquear el nivel sucesivo [16].

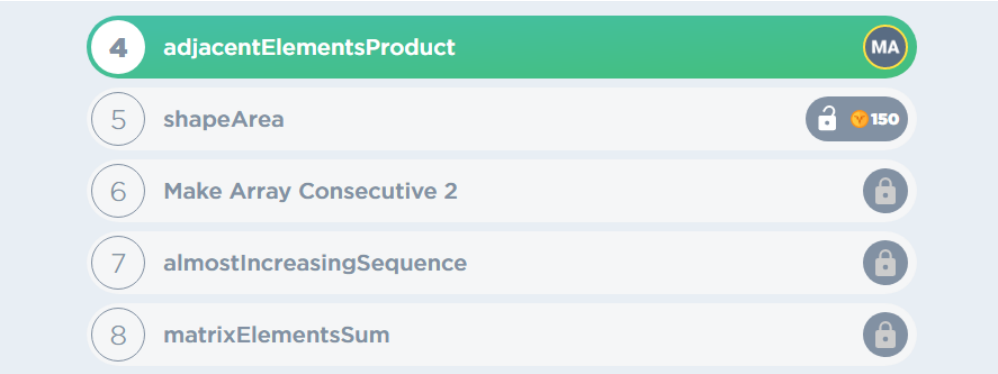


Figura 9. Cantidad de "coins" necesaria para desbloquear el ejercicio sucesivo [16].

Al entrar dentro de un ejercicio, arriba a la izquierda aparece su dificultad y los "coins" de recompensa si se consigue completar (ver Figura 10). Para probar si el código escrito es correcto, se ejecutará una batería de tests que se dividen en gratuitos y ocultos. Los gratuitos ejecutan una serie de casos de prueba muy sencillos y que a veces se repiten entre sí. Por ejemplo, siguiendo el ejercicio del palíndromo de la Figura 10, entre los casos de prueba gratuitos se encuentran dos que, en términos del dominio del problema, son iguales (ver Figura 11 y Figura 12). En cuanto a los test ocultos, para poder ver los casos de pruebas asociados, es necesario pagar 5000 "coins" (ver Figura 13). Estos últimos test son mucho más elaborados; de hecho, en el vigésimo test de este ejercicio se prueba una cadena de caracteres cuya longitud es de alrededor de 100.000 y el sitio web te permite descargar el *json* asociado para visualizarlo (ver Figura 14)[17].

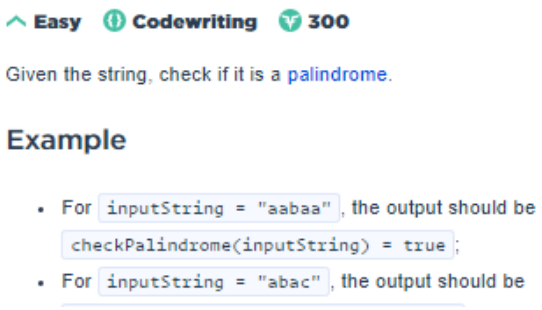


Figura 10. Dificultad de un ejercicio y su recompensa en "coins" por completarlo [17].

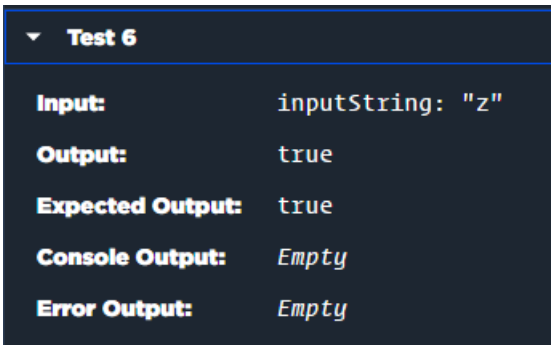


Figura 11. Test gratuito que comprueba si la cadena "z" es un palíndromo [17].

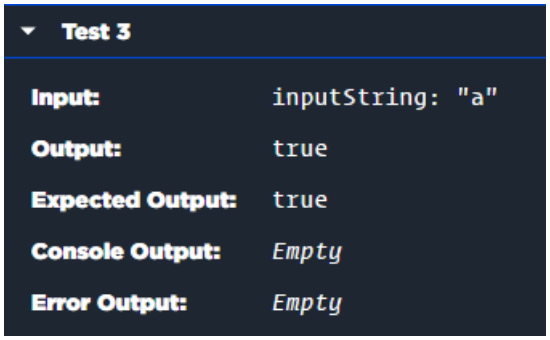


Figura 12. Test gratuito que comprueba si la cadena "a" es un palíndromo [17].

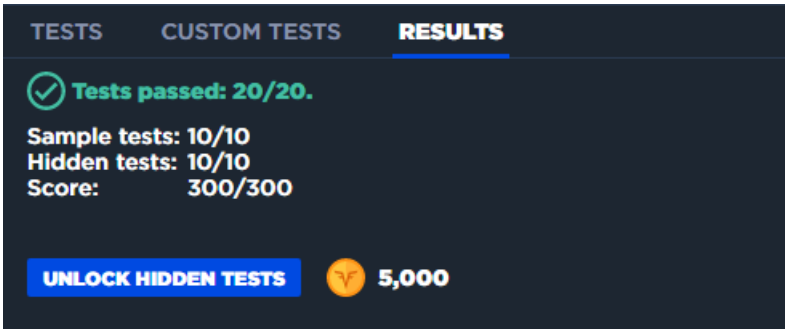


Figura 13. Opción de visionar los test ocultos pagando 5000 "coins" [17].

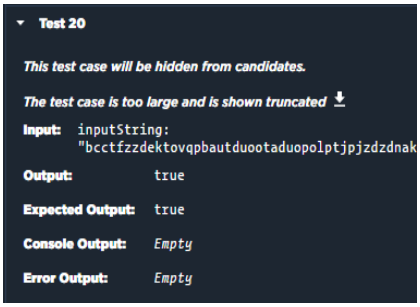


Figura 14. Test oculto cuyo caso de prueba es una cadena de alrededor de 100.000 caracteres que se pueden visualizar si se descarga el json asociado [17].

Moviéndonos a la sección de desafíos de organizaciones [18], encontramos un gran número de estas (algunas pueden apreciarse en la Figura 15). Si entramos en alguna de ellas, como por ejemplo en Dropbox [19], podemos observar que aparecen una serie de desafíos (ver Figura 16) etiquetados con una dificultad y la duración estimada. También aparece la cantidad de "coins" que se obtienen al completar cada uno de ellos. La cantidad recompensada asciende cuanto más difícil y largo sea el desafío, aunque existen desafíos con misma dificultad, pero diferente duración que ofrecen la misma recompensa (desafíos "displayOff" y "loadTimeEstimator" de la Figura 16).

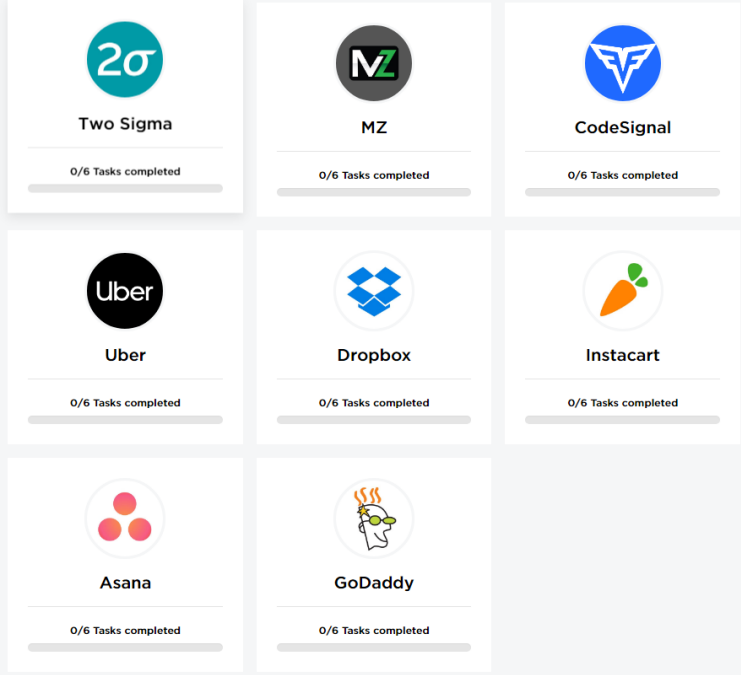


Figura 15. Lista parcial de las organizaciones que tienen desafíos de programación [18].

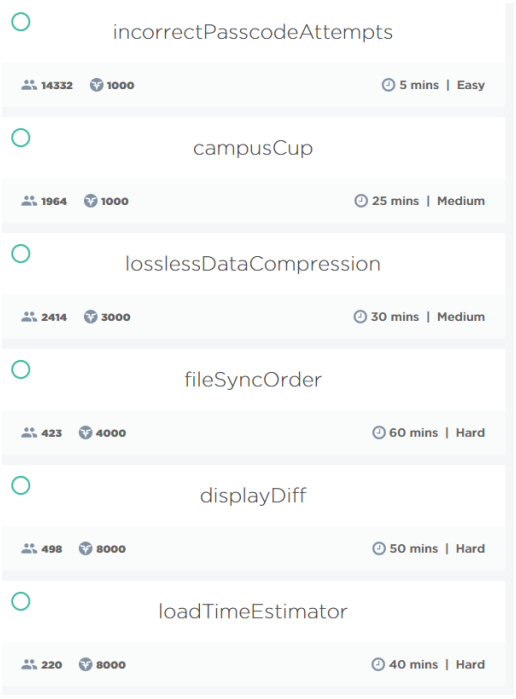


Figura 16. Desafíos de la organización Dropbox [19].

Por otro lado, aquellos desarrolladores interesados en practicar típicas entrevistas de trabajo donde se requiere realizar ejercicios de programación pueden practicar este tipo de ejercicios en la sección “Interview practice”. Para empezar, el usuario deberá elegir un plan de estudios [20] y, posteriormente, podrá realizar ejercicios de diferentes tipos (ver Figura 17) [21]. Si entramos a cualquiera de estos tipos, aparecerá una lista de ejercicios donde solo el primero está bloqueado, pero tenemos la posibilidad de desbloquear cualquier otro pagando 250 “coins”. Cada uno de estos ejercicios otorga una recompensa variable en base a su duración y dificultad (ver Figura 18) [22].

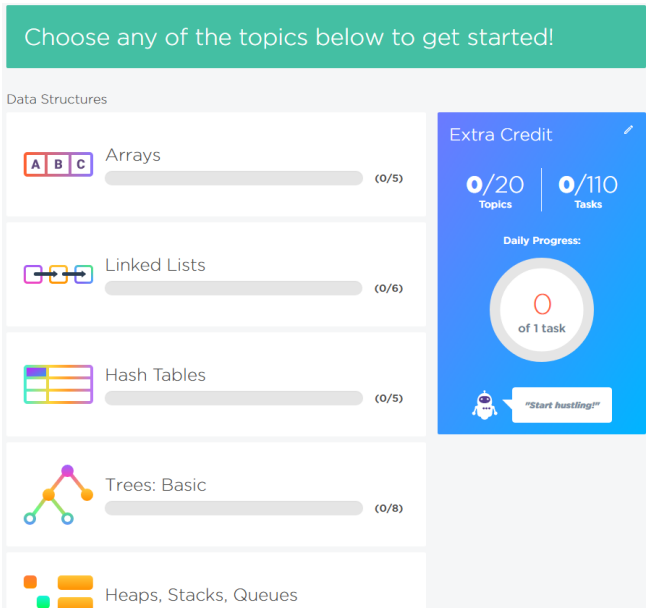


Figura 17. Lista parcial de tipos de ejercicios para practicar entrevistas de trabajo. Esta lista es acorde al plan de estudio llamado "Extra Credit"[21] .

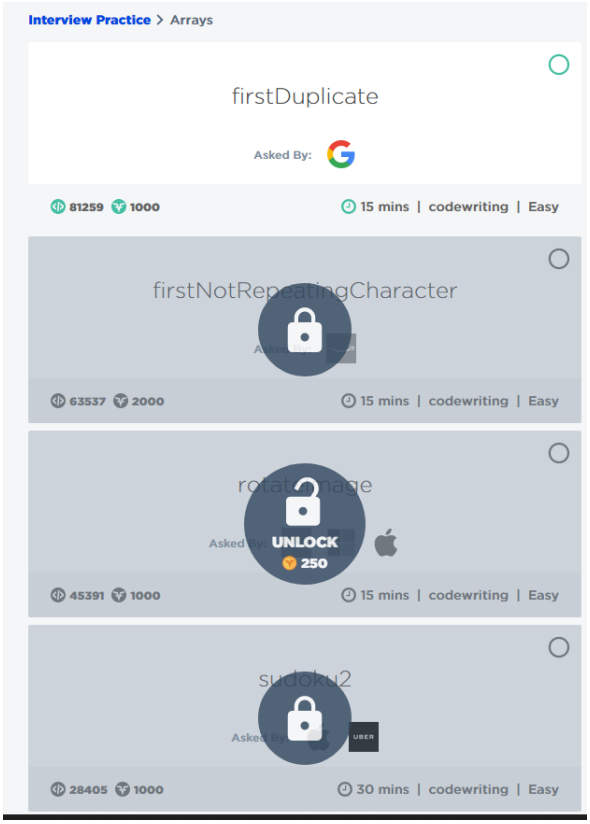


Figura 18. Ejercicios de tipo "Array" donde solo el primero se puede realizar a menos que se pague 250 "coins" para desbloquear cualquier otro [22].

Un último punto para destacar es que el usuario, a medida que utiliza la aplicación, conseguirá diferentes insignias (ver Figura 19) aunque estas no otorgan ningún "coin".

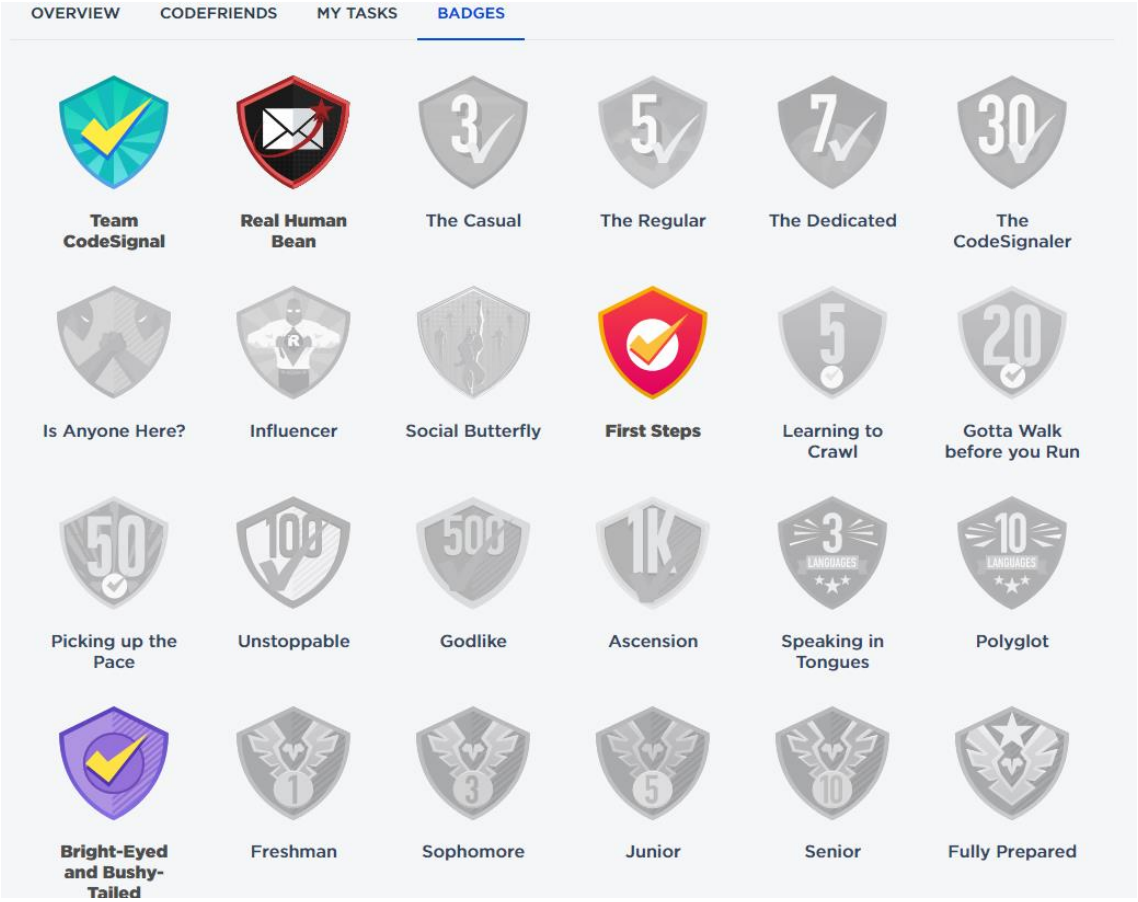


Figura 19. Algunas de las insignias que puede conseguir el usuario en Codesignal.

En conclusión, Codesignal ofrece un sistema que facilita la obtención de "coins" de múltiples maneras y en grandes cantidades, pero su gasto, en general, es muy poco en comparación a lo ganado. El único lugar donde te puedes gastar muchos "coins" es desbloqueando tests ocultos para ver los posibles casos de prueba.

2.3 DataCamp

DataCamp[3] es una compañía con presencia tanto en la web como en dispositivos móviles cuyo objetivo es que los usuarios reciban una educación de calidad y un buen desarrollo de habilidades en lo referente a la ciencia de datos [23].

En esta plataforma es posible realizar ejercicios de programación, principalmente, dentro de cursos, prácticas o proyectos y, al completarlos, el usuario obtendrá puntos de experiencia (XP) como forma de medir el progreso [24]. En la Figura 20 podemos ver un ejercicio, situado dentro de un curso [25], cuya recompensa es de 50 XP. Además, si el usuario no tiene muy claro cómo seguir, puede comprar una pista por 15 XP. La recompensa varía según la dificultad del ejercicio y el valor de la pista crece de forma proporcional, tal como podemos ver en la Figura 21. Tal como se explica en [24], el valor de las pistas no se descuenta de la cantidad de XP que tiene el usuario, sino de la recompensa otorgada (ver Figura 22). Además, si la pista no es suficiente y te rindes, puedes comprar la respuesta, reduciendo la recompensa del ejercicio a 0 XP (ver Figura 23). De esta manera, el usuario nunca perderá su progreso materializado en puntos de experiencia, sino que obtendrá menos recompensa y, por ende, progresará menos si ha obtenido ayuda durante la realización del ejercicio. No existe ninguna forma de gastar esa XP a cambio de algo [24].

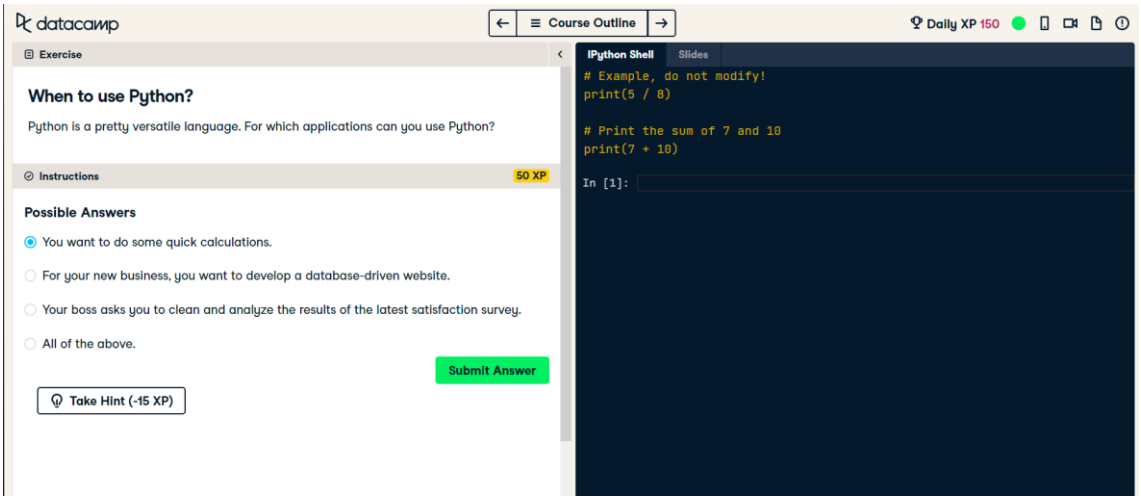


Figura 20. Ejercicio dentro de un curso de DataCamp con 50 XP de recompense [25].

[3] https://www.datacamp.com/

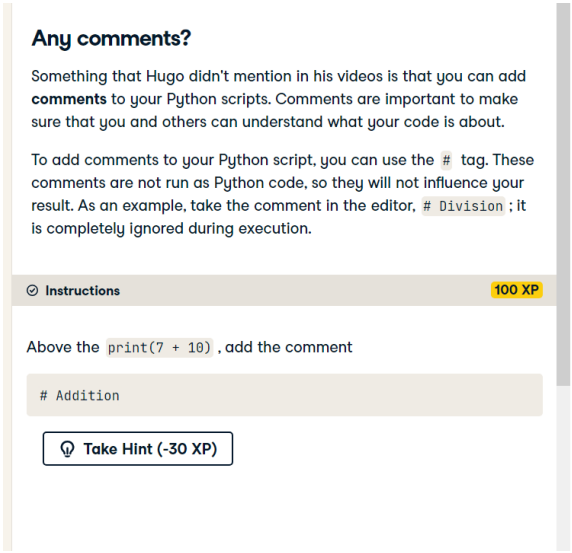


Figura 21. Ejercicio dentro de un curso de DataCamp con 100XP de recompensa [26].

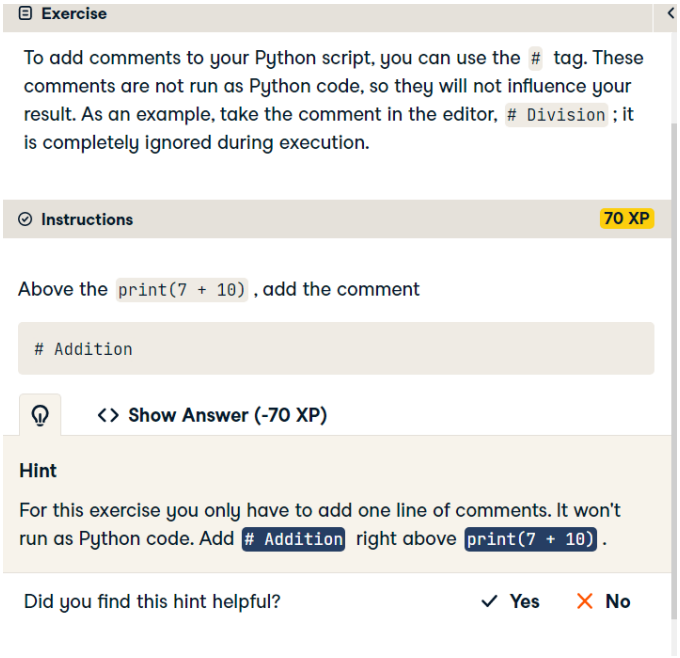


Figura 22. Ejercicio dentro de un curso de DataCamp con una pista comprada [26].

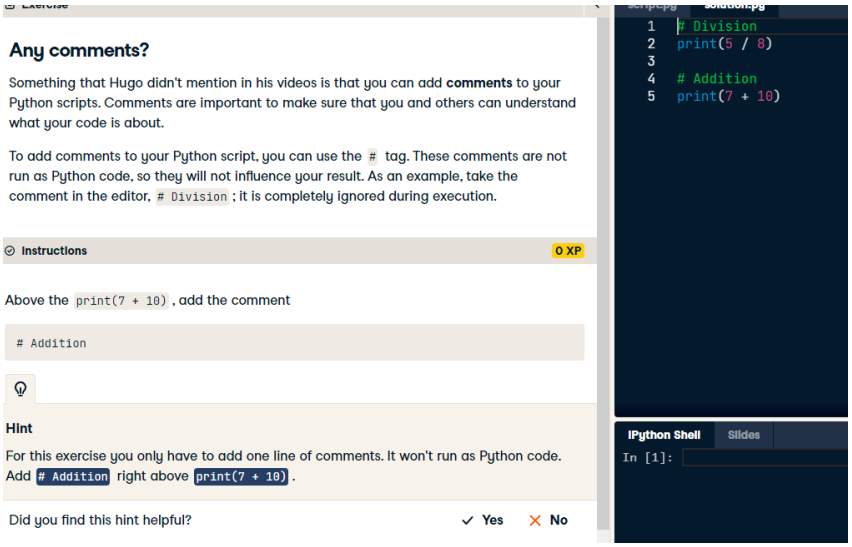


Figura 23. Ejercicio dentro de un curso de DataCamp con solución comprada [26].

En cuanto a los proyectos [27], estos están formados por una serie de ejercicios que tienen unos tests asociados, pero no otorga recompensa alguna completarlos, sino que la XP se obtiene en su totalidad al finalizarlo, tal como podemos ver en la Figura 24.

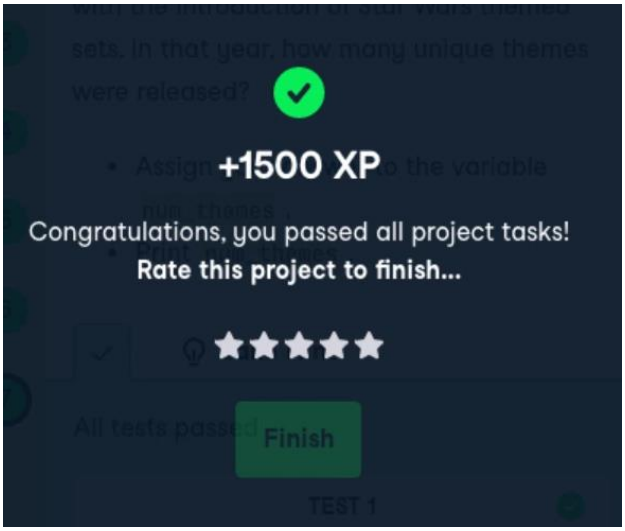


Figura 24. Recompensa de 1500XP por completar un proyecto. Imagen extraída del vídeo situado en [27]. *Para acceder es necesario cerrar sesión en la web.*

En el caso de querer repetir un curso, DataCamp te ofrece la oportunidad de reiniciarlo, perdiendo toda la XP ganada por ese curso hasta el momento (ver Figura 25) [28].

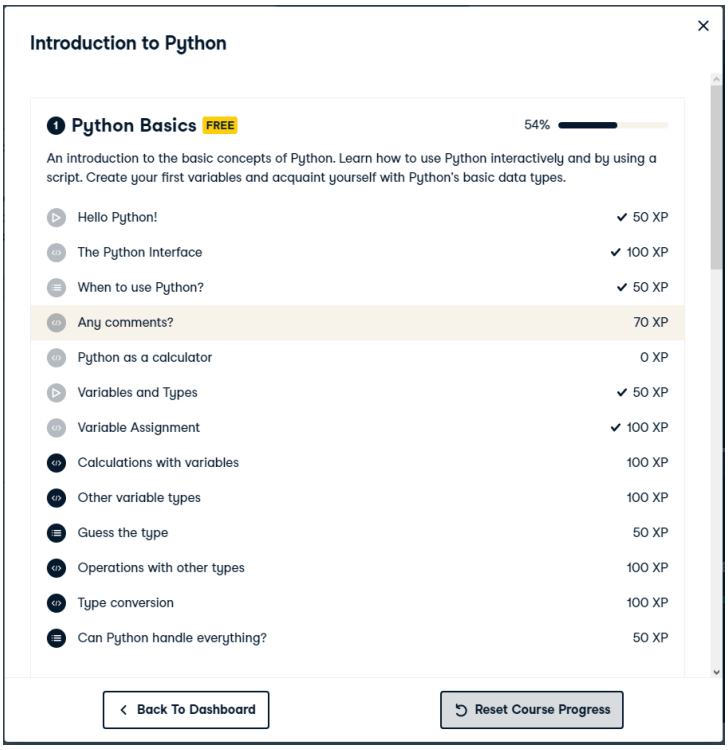


Figura 25. Lista de ejercicios de un curso de DataCamp. Abajo a la derecha se puede reiniciar el curso.

Otra forma de obtener XP es mediante la realización de sesiones de prácticas [29], formadas por una serie de ejercicios y, hasta que el usuario no acierte cinco de ellos, la sesión no finalizará. Una vez terminada, el usuario recibirá 250 XP como recompensa (ver Figura 26).

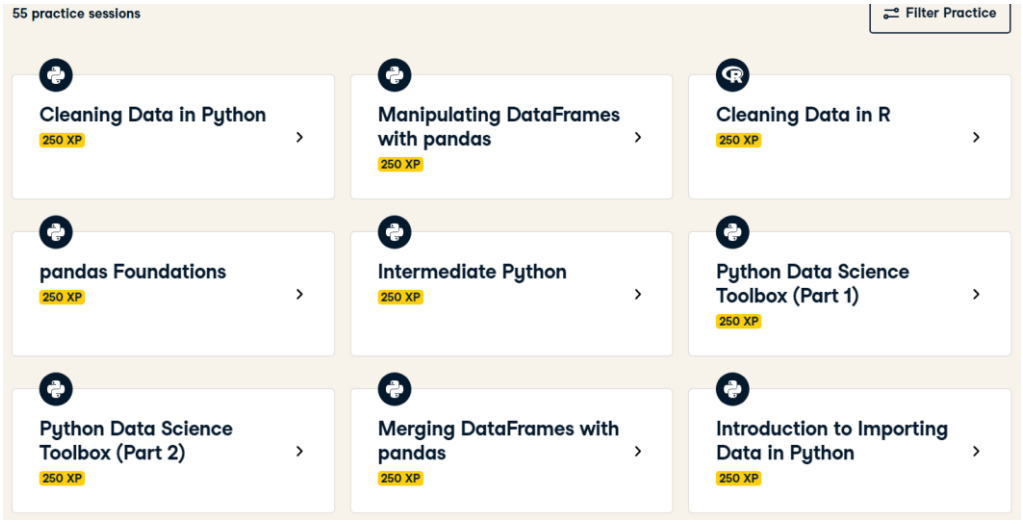


Figura 26. Lista parcial de las sesiones de prácticas de CodeCamp. Completar una sesión otorga 250 XP [29].

Por último, hay que destacar que existe una funcionalidad que consiste en mostrar la cantidad de días seguidos en los que el usuario ha estado de forma activa en DataCamp y ha obtenido, al menos, 250 XP [30].

2.4 Brawl Stars

Brawl Stars es un videojuego para dispositivos móviles, con más de 100 millones de descargas, basado en batallas multijugador en diferentes modos de juego. Para poder participar en estas batallas se utilizan diversos personajes desbloqueables llamados "brawlers" [31]. El objetivo del juego es conseguir el mayor número de trofeos posibles a medida que vas desbloqueando nuevos "brawlers" y mejorando sus niveles. Además, estos personajes tienen sus propios trofeos y, según la cantidad que tengan, un determinado rango. Este videojuego presenta múltiples formas de otorgar recompensas y diversas unidades monetarias: fichas, monedas de oro (también llamadas "coins"), gemas, y puntos estelares. En la Figura 27, podemos ver el menú principal de la aplicación. Arriba a la izquierda, aparecen los trofeos del jugador. Arriba a la derecha los puntos estelares, las monedas de oro y las gemas. Abajo a la derecha, el contador de fichas posibles para ganar en partidas. Abajo a la izquierda, las fichas necesarias para subir de nivel en el pase de batalla (explicado a continuación). Finalmente, arriba en el centro, el rango y trofeos del "brawler" seleccionado. Para poder progresar en el juego existe lo que se denomina "pase de batalla" [32], un sistema de progreso basado en la obtención de niveles que otorgarán recompensas a medida que el usuario juega partidas (ver Figura 28). Además, este pase de batalla posee una sección (parte superior de la Figura 28) cuyas recompensas se pueden obtener solo si pagas una cierta cantidad de gemas y, cada 2-3 meses, el pase de batalla se reinicia con uno nuevo, por lo que habrá que pagar de vuelta si se desean esas recompensas adicionales.

Figura 27. Pantalla principal de Brawl Stars.

Figura 28. Pase de batalla de Brawl Stars con algunas de las recompensas que ofrece.

Las fichas [33] son un elemento fundamental para poder progresar en el pase de batalla, pues ganando fichas, se podrá subir de nivel en el pase. Para conseguirlas, el usuario puede completar misiones del pase de batalla, las cuales pueden ser diarias o de temporada (ver Figura 29). Además, cada cuatro horas cambia algún modo de juego y/o mapa, lo que otorga unas pocas fichas al jugador, lo que motiva a entrar al juego al menos una vez al día (ver Figura 30). Entre estos modos de juegos, también hay mapas creados y votados por la comunidad. En suma, al jugar partidas también puedes obtener fichas, pero la cantidad depende de factores como el resultado (victoria, derrota o empate), si el usuario sube de nivel, si el "brawler" sube de rango o si existe algún tipo de duplicador activado. Sin embargo, es importante destacar que las fichas obtenidas por el resultado no son infinitas, sino que están limitadas por un contador: cada 2 horas y 40 minutos el juego recarga el contador 20 unidades (hasta un máximo de 200) y, cuando un usuario decide jugar, las fichas obtenidas por el resultado se restan al contador. Si el contador está en 0/200, no se obtendrán fichas (ver Figura 27).

Figura 29. Misiones del pase de batalla.

Figura 30. Lista de modos de juego. Entre ellos, hay uno nuevo que acaba de aparecer y si se pincha sobre él, dará fichas como recompensa.

En segundo lugar, tenemos la moneda más valiosa del juego: las gemas [34]. Las gemas solo se pueden obtener mediante el pase de batalla o comprándolas con dinero real (ver Figura 32) se utilizan para comprar múltiples cosas:

- El pase de batalla o niveles de este.
- Aspectos (también llamados “skins”) de “brawlers”. Estos también pueden comprarse con monedas de oro (ver Figura 31).
- Cajas (ver Figura 32).
- Monedas de oro (ver Figura 32).
- Duplicadores de fichas (ver Figura 32).
- Brawlers (ver Figura 31).

Figura 31. Parte de la tienda de Brawl Stars con ofertas diarias y aspectos de "brawlers" para comprar.

Figura 32. Parte de la tienda de Brawl Stars. Con gemas se puede comprar un duplicador de fichas, cajas y oro. También se pueden comprar gemas con dinero real.

Por otro lado, tenemos las monedas de oro ("coins") [35], un recurso utilizado para mejorar de nivel los "brawlers" (ver Figura 33), cuyo precio aumenta cuanto más nivel se necesite; y para, por un lado, comprar objetos que aparecen en la sección "Ofertas diarias" (en esta sección también se regalan objetos); y, por otro lado, para comprar algunos aspectos exclusivos como el de la Figura 31. Para obtener estas monedas es necesario abrir cajas. Estas cajas se obtienen, o bien, como recompensa al subir de nivel en el pase de batalla (Figura 28), o bien, comprándolas en la tienda. Otra forma de conseguir oro es, tal como se explicó antes, comprándolo con gemas (ver Figura 32).

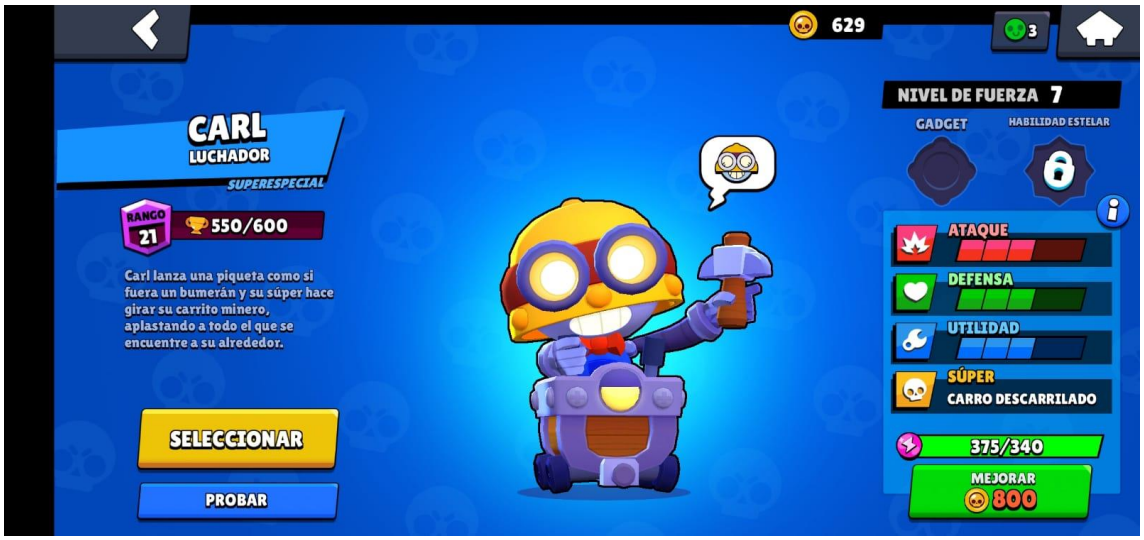


Figura 33. Estadísticas de un "brawler" y botón para subirlo de nivel.

La última moneda presente en el juego se conoce como “puntos estelares” [36]. Estos puntos se utilizan para comprar algunos aspectos de “brawlers” (ver Figura 34) y ciertas cajas que aparecen cada cierto tiempo en la tienda. Los puntos estelares se pueden conseguir subiendo un “brawler” cada cinco rangos (cuánto más alto el rango, mayor la recompensa). Además, cada cierto tiempo, los “brawlers” con más de 500 trofeos perderán una cierta cantidad y obtendrán puntos estelares (ver Figura 35 y Figura 36). Por otro lado, existe una competición llamada “lucha estelar” [37] y consiste en jugar tres partidas diarias en un modo específico y, dependiendo del resultado (victoria, derrota o empate), se ganarán más o menos puntos. Además, si la partida ha ido muy bien y se cumplen ciertas condiciones, se ganará una pequeña cantidad de puntos adicional. Al cabo de dos semanas, en base a la puntuación conseguida, ganarán una cierta cantidad de puntos estelares (ver Figura 37 y Figura 38). En esta competición, los 200 mejores jugadores recibirán una recompensa adicional (ver Figura 39).

Figura 34. Parte de la tienda de Brawl Stars donde se pueden comprar aspectos de "brawlers" con puntos estelares.

Figura 35. Parte del camino de trofeos e información sobre este.

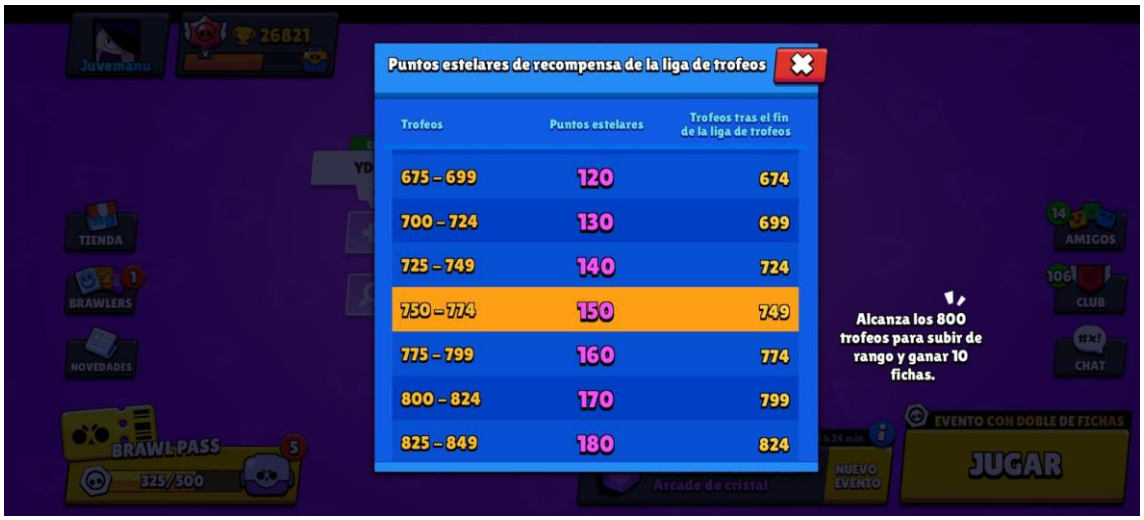


Figura 36. Puntos estelares que se obtienen en el reinicio de trofeos en base a los trofeos que tenga el "brawler".

Figura 37. Clasificación de lucha estelar.

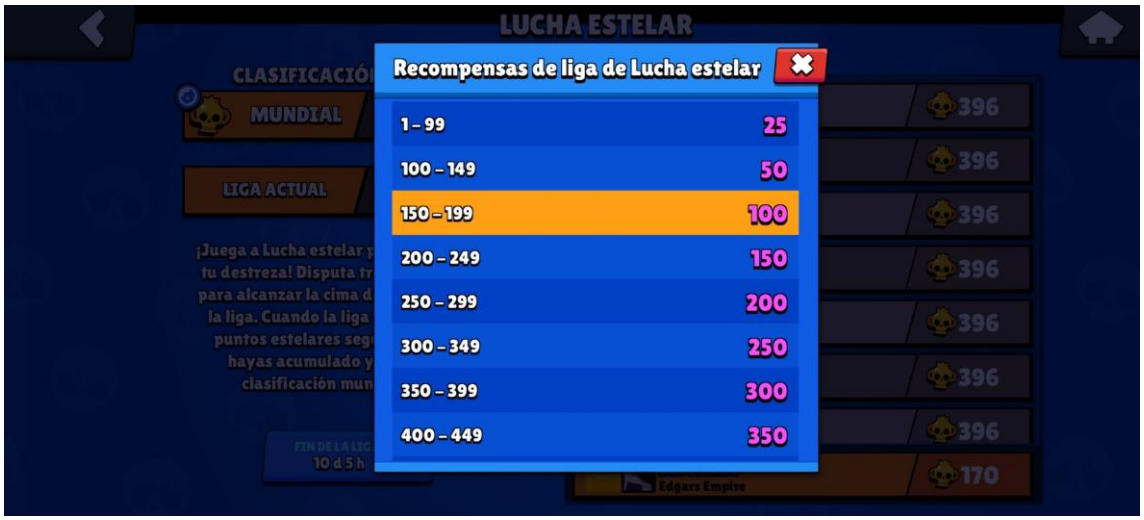


Figura 38. Recompensas en lucha estelar en base a los puntos conseguidos.

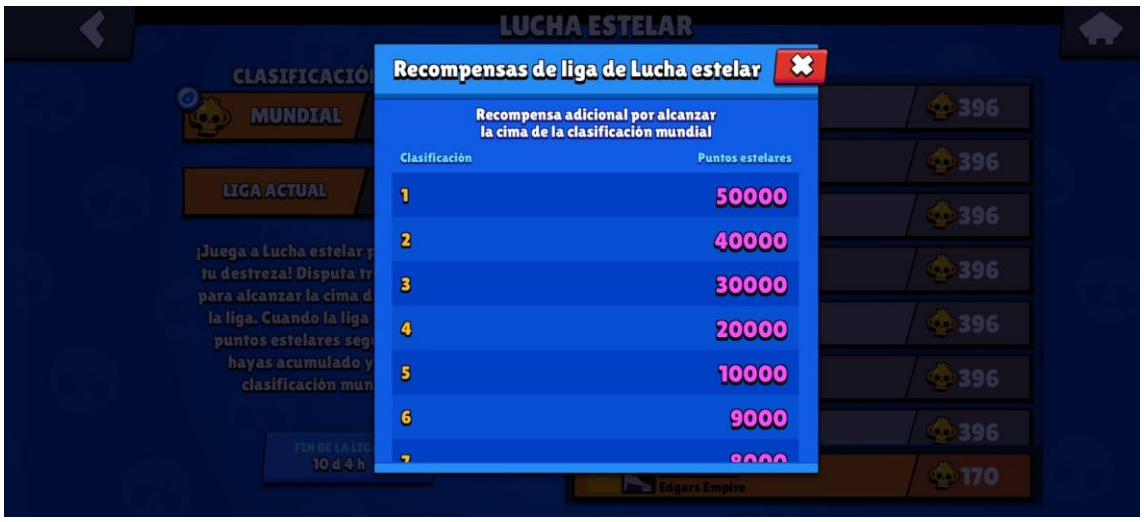


Figura 39. Recompensa adicional en lucha estelar a los jugadores con mejor puntuación.

Finalmente, en el juego tienen lugar diferentes eventos cada cierto tiempo como torneos [38], con puntos estelares y monedas de oro como recompensa, y eventos donde se obtienen más monedas de oro y fichas a la hora de jugar partidas.

2.5 Forest

Forest[4] es una aplicación para dispositivos móviles con más de 6 millones de usuarios. Ha sido galardonada con diversos premios de Google Play y está enfocada a mejorar la productividad y concentración de los usuarios, tratando de reducir el tiempo de utilización del dispositivo móvil. Para conseguir esto, la aplicación invita al usuario a plantar semillas que, con el paso del tiempo, se convertirán en árboles (o arbustos) virtuales y hasta que el árbol no haya crecido, no se podrá salir de la aplicación pues, en dicho caso, el árbol marchitará. Tanto los árboles crecidos como aquellos marchitados aparecerán en un bosque (ver Figura 40) que refleja el esfuerzo y concentración que ha tenido el usuario y es responsabilidad de este mantener un buen bosque y no desconcentrarse utilizando el móvil [39]. Esta aplicación tiene implementado un sistema de monetización formado por dos tipos de unidades monetarias: monedas y cristales de tiempo.

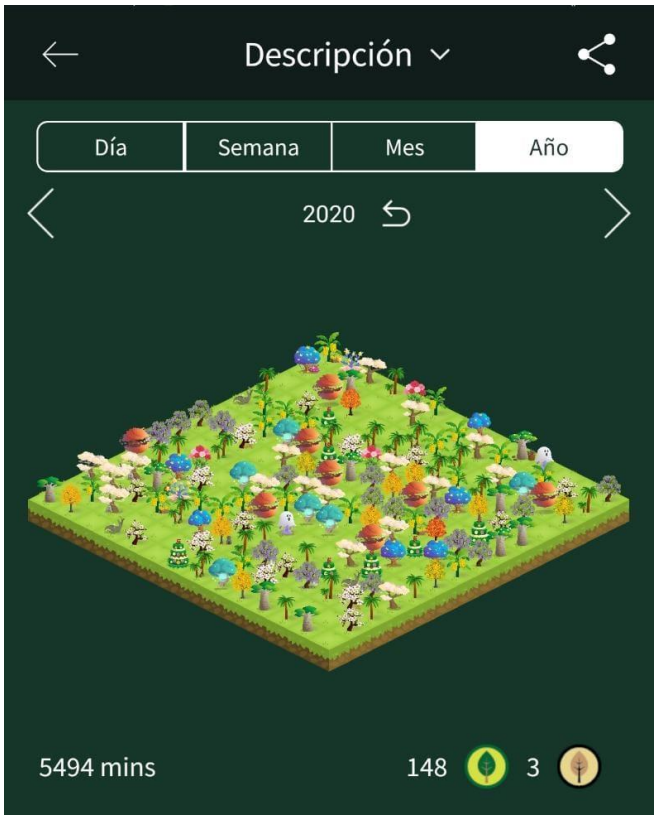


Figura 40. Bosque del año 2020 de un usuario de Forest.

[4] https://forestapp.cc

La unidad monetaria principal son las monedas. Con ellas es posible comprar en la tienda diferentes tipos de árboles y arbustos (ver Figura 41) para tener un bosque más heterogéneo. También es posible comprar diferentes sonidos para activarlos durante el tiempo de concentración (ver Figura 42). Otra utilidad de las monedas es la posibilidad de plantar un árbol en el mundo real a cambio de 2500 de estas (ver Figura 43), hasta un máximo de 5 árboles por usuario [40].

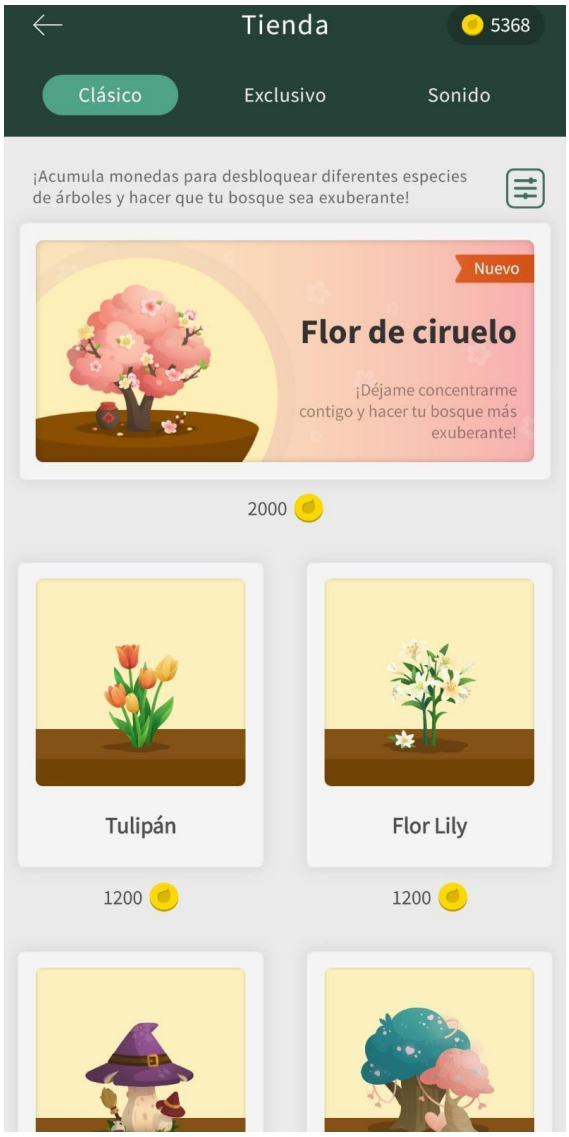


Figura 41. Tienda de Forest. En esta sección se pueden comprar árboles y arbustos.

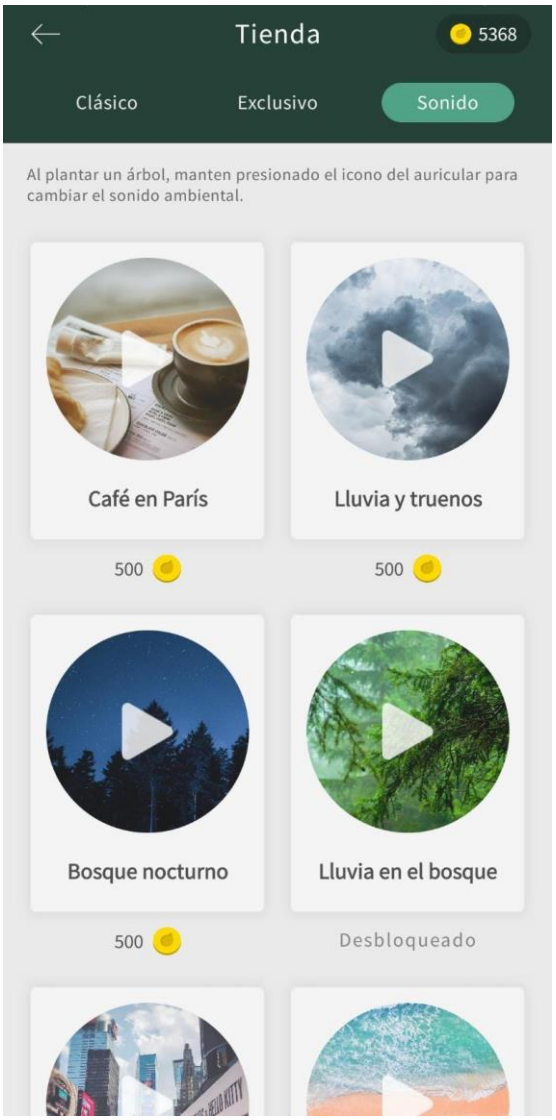


Figura 42. Tienda de Forest. En esta sección se pueden comprar sonidos.

Figura 43. Pantalla que permite plantar árboles reales a cambio de monedas de la aplicación.

En cuanto a la obtención de las monedas, la principal fuente de ingresos consiste en plantar semillas y hacer que crezcan árboles o arbustos. A la hora de plantar, el usuario decide cuánto tiempo quiere permanecer concentrado [41]: a más tiempo concentrado, mayor recompensa de monedas [42]. Por otro lado, otra forma de conseguir monedas consiste en la obtención de logros (ver Figura 44) a medida que utilizas la aplicación, cuya recompensa en monedas (ver Figura 45) depende de la complejidad del logro. Por último, es posible aumentar la obtención de monedas al plantar por medio, o bien, de la visualización de anuncios, o bien, de la compra con dinero real de potenciadores (ver Figura 46).

Figura 44. Lista de logros de Forest.

Figura 45. Recompensa por completar un logro en Forest.

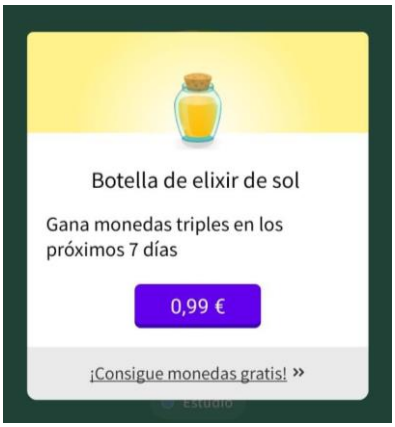


Figura 46. Potenciador que aumenta la obtención de monedas durante 7 días. Se compra con dinero real.

Además de las monedas, existe lo que se denomina "cristal de tiempo", una unidad monetaria que pasa en segundo plano cuya única utilidad es comprar ciertos árboles exclusivos que se podían haber obtenido gratuitamente en eventos temporales ya pasados (ver Figura 47). Se pueden obtener invitando amigos o comprándolos con dinero real.

Figura 47. Árbol exclusivo que cuesta 150 cristales de tiempo.

2.6 ASys. Estado actual

ASys es un sistema de evaluación semiautomático para ejercicios y exámenes de programación en lenguaje Java [43], el cual se ha extendido también al lenguaje funcional Haskell. Para conseguirlo, se ayuda de las capacidades de la reflexión, las cuales permiten, por un lado, evaluar el resultado final producido por el código y, por otro lado, inspeccionar la estructura del código fuente en sí y los recursos del lenguaje utilizados en la implementación realizada [44]. Las principales funcionalidades de ASys [44] son:

1. Identificar los errores de los alumnos y ordenar los exámenes para que se corrijan en un orden en concreto. Esto consigue, por un lado, que errores parecidos se encuentren juntos para agilizar su corrección y, por otro lado, evitar en lo posible efectos subjetivos humanos y negativos al corregir un muy buen examen precedido por uno malo y viceversa.

2. Los ejercicios están divididos en propiedades atómicas de manera que el sistema es capaz de detectar errores en cada una de ellas. Una vez detectado, lo registra junto a una nota y al comentario del profesor para el estudiante. Si se detecta de nuevo el mismo error, el sistema se da cuenta y no vuelve a penalizar por este. Gracias a esto se puede evitar, en suma, la propagación del error que conllevaría a una penalización mayor. Los errores detectados en estas propiedades son corregidos por el sistema y, si ASys no puede, por el profesor (ver Figura 48).

3. Los errores de cada uno de los estudiantes se registran y clasifican. Esto permite al sistema recomendar ejercicios específicos a cada alumno para solucionar sus puntos débiles.

4. Además de la nota y la solución del profesor, la salida obtenida tras realizar ejercicios incluye el código del alumno corregido junto a un informe detallado de los problemas encontrados. Esto se detalla más en la Figura 48, donde observamos cómo la solución del alumno va atravesando una serie de fases y, en cada una de ellas, se recoge la información que se presentará en este informe final. Concretamente, se trata de un diagrama de flujo compuesto por una fase de construcción de ejercicios (y generación de casos de pruebas) y por otras tres fases (compilación, análisis y testing) que componen el camino que debe seguir la solución del alumno.

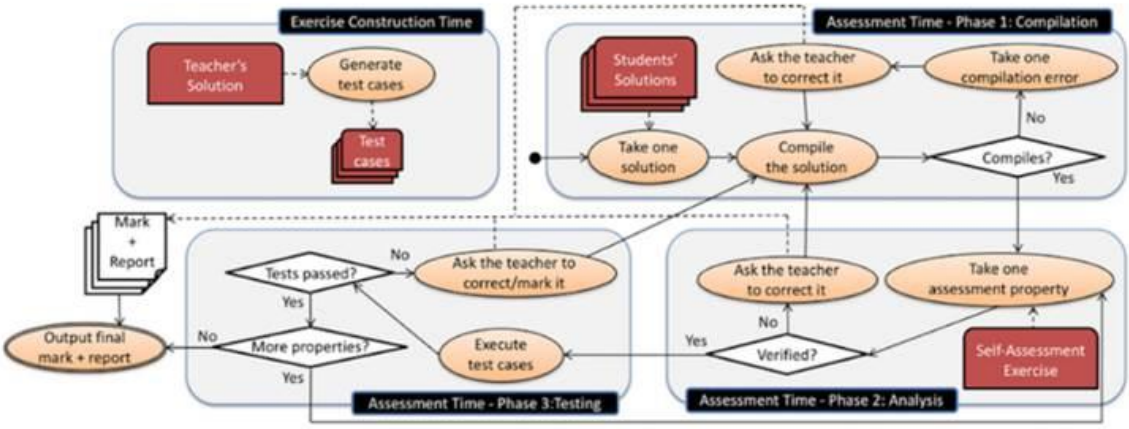


Figura 48. Diagrama de flujo de datos de ASys.

En suma, tal como se puede apreciar en la parte superior izquierda de la Figura 48, cuando el profesor crea un ejercicio y sube su solución, ASys genera automáticamente un conjunto de *tests*, maximizando la cobertura de arco [44].

Por otro lado, en ASys se pueden crear grupos, los cuales pueden utilizar los profesores para crear tareas y publicar anuncios para sus alumnos (ver Figura 49).

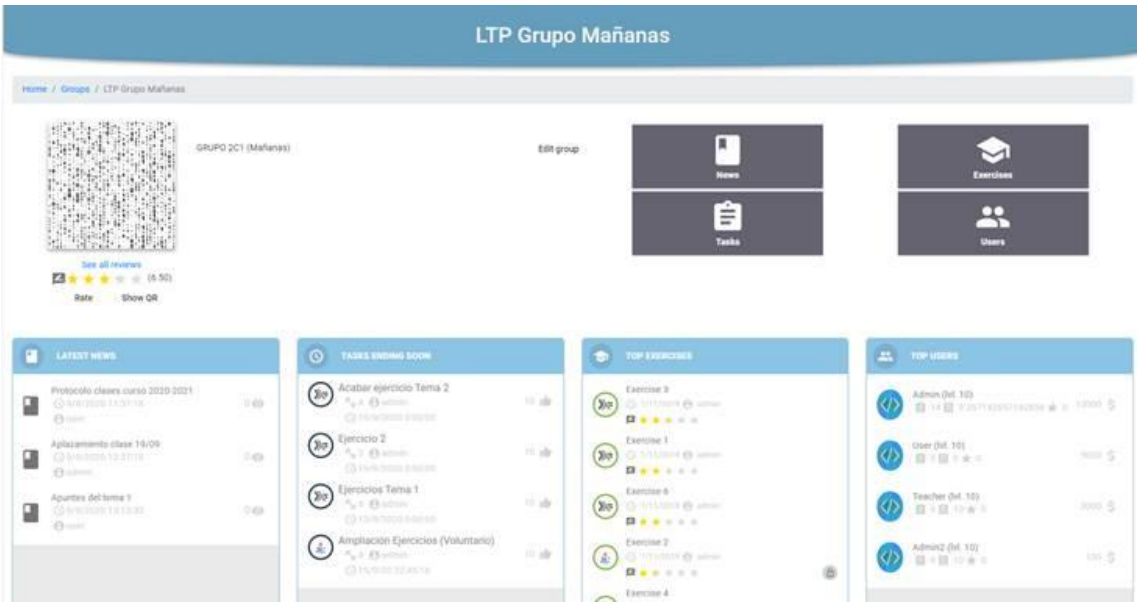


Figura 49. Interfaz de un grupo de ASys. Aparecen las últimas noticias, así como las tareas creadas, los mejores ejercicios y usuarios.

Por último, ASys posee un sistema de *rankings* donde los alumnos pueden medirse frente a otros compañeros en base a ejercicios completados y a calificaciones obtenidas en los

ejercicios (ver Figura 50). Además, cada usuario puede obtener una serie de logros al completar ejercicios o al realizar ciertas acciones en la plataforma (ver Figura 51).

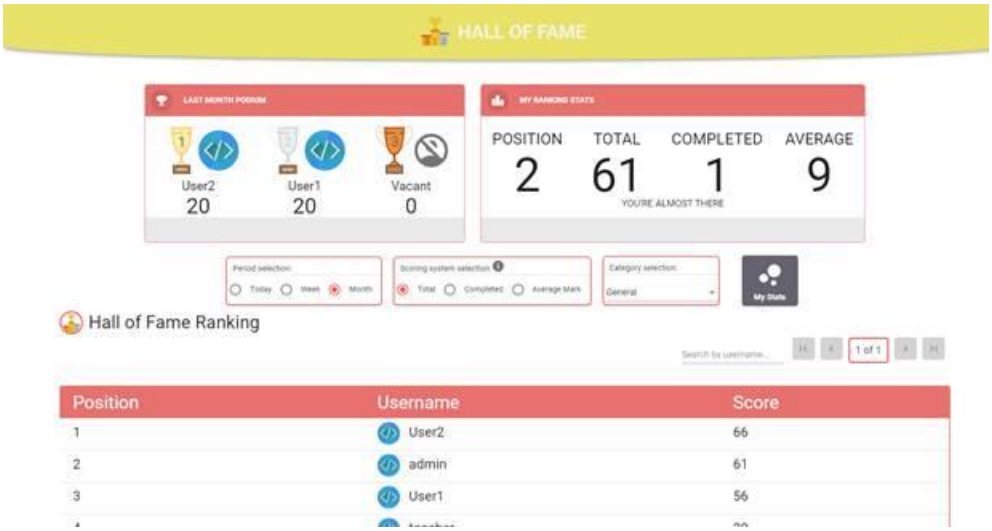


Figura 50. Interfaz de rankings en ASys.

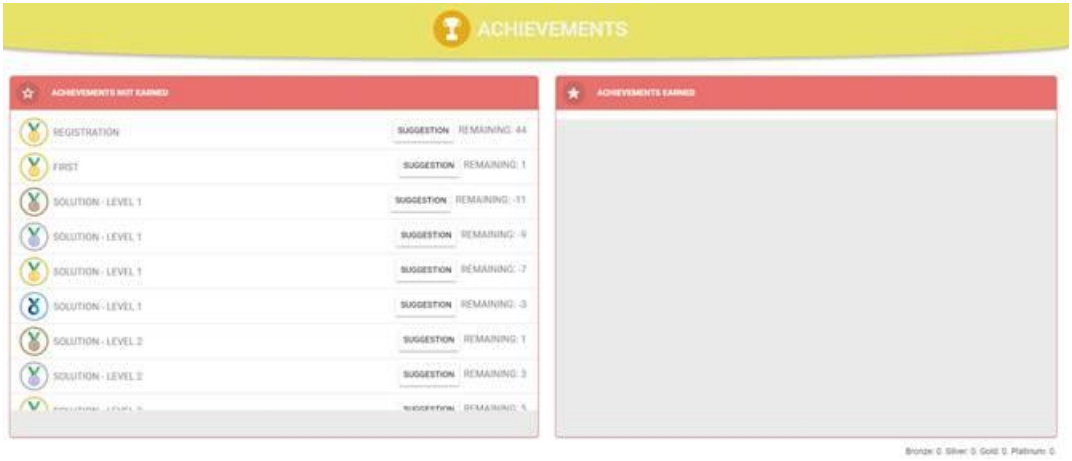


Figura 51. Interfaz de logros de un usuario en ASys.

2.7 Comparación de las aplicaciones

A modo de resumen, en la Tabla 9 podemos ver una comparación de las plataformas de Codewars, CodeSignal, DataCamp, BrawlStars y Forest. Esta tabla será de mucha utilidad para realizar la propuesta a implementar.

Tabla 9. Comparación de funcionalidades relacionadas con la gamificación entre las aplicaciones Codewars, Codesignal, DataCamp, Brawl Stars y Forest.

HERRAMIENTAS					
FUNCIONALIDADES	**Codewars**	**Codesignal**	**DataCamp**	**Brawl Stars**	**Forest**
Sistema de monetización/ gamificación	Solo gamificación	Sí	Solo gamificación. Representa el progreso mediante XP. En los ejercicios se puede utilizar, pero en lugar de gastarse, simplemente reduce su recompensa al completar el ejercicio	Sí	Sí
Más de una unidad monetaria	No tiene monetización, pero hay recompensas con rangos y obtención de honor	-	-	Tiene 4: • Fichas • Monedas de oro • Gemas • Puntos estelares	Tiene 2: • Monedas • Cristales de tiempo
Histórico de movimientos (gastos/ganancias)	-	-	-	-	-
Recompensas por completar un ejercicio	Sí	Sí	Sí, aunque se reduce si utiliza ayuda	Sí, al jugar partidas	Sí, al estar concentrado una determinada cantidad de tiempo
Recompensas de temporada	-	-	-	Cada cierta semana, se reparten puntos estelares en base a ciertas puntuaciones obtenidas	-
Recompensas por entrar en la aplicación	-	-	Puedes acumular una racha de días seguidos conectándote y consiguiendo una pequeña cantidad de puntos. No hay recompensa	Obtienes fichas por cada nuevo modo de juego. Además, cada día se regala algo en la tienda	-

Recompensas por contribuir a la comunidad	Creando, traduciendo y votando ejercicios de programación.	-	-	Hay un modo de juego con mapas creados por la comunidad. Al jugar dichos mapas, puedes votar positiva o negativamente el mapa y eso te otorga algunas fichas. Máximo 3 veces al día	-
Sistema de niveles	A medida que se completen ejercicios, se sube de rango	-	-	A medida que se ganen partidas, se sube de trofeos y de rango con los personajes	-
Desbloqueo de niveles o funcionalidades	Cuanto más honor consigas, desbloqueas privilegios; como por ejemplo poder crear ejercicios.	Pagando monedas se pueden desbloquear desafíos diarios, desafíos de organizaciones, niveles, ejercicios	-	Puedes comprar cajas, personajes o el pase de batalla con gemas (la moneda más preciada en el juego) para avanzar más rápido.	-
Comprar pistas	-	-	Una pista por ejercicio, idéntica para todos los usuarios. Reduce la recompensa obtenida	-	-
Comprar solución	-	-	Sí. La recompensa obtenida se reduce a cero.	-	-
Desbloqueo de elementos estéticos	-	Al comprar los niveles, puedes apreciar mejor los dibujos asociados.	-	Emoticonos y aspectos (*skins*) de personajes	Árboles, arbustos y sonidos.
Logros/Medallas/ Insignias	-	Sí	-	-	Sí
Potenciador/ Duplicador de recompensas	-	-	-	Puedes comprar duplicadores de fichas. También tienen lugar eventos temporales que aumentan las fichas y monedas oro conseguidas al jugar	Puedes comprar con dinero real potenciadores que aumentan las ganancias de monedas. También se pueden ganar monedas visualizando anuncios.

Desafíos especiales con recompensas	-	Desafíos diarios	-	Misiones diarias y de temporada con fichas como recompensas	-
Tabla de clasificaciones o lugar para ver tu progreso	Hay una tabla para ver los mejores usuarios con mayor cantidad de honor	Hay una tabla para ver a los usuarios que más ejercicios hicieron	-	Hay una tabla para ver a los jugadores con más trofeos y otra para ver a los jugadores con más puntos conseguidos en lucha estelar	Hay una tabla para ver a los usuarios que más tiempo han estado concentrados
Pagar para crear grupos o tareas	-	-	-	-	-
Pagar para mantener algo dentro de la app con monedas virtuales (estilo alquiler o subscripción)	-	-	-	Pagar el pase de batalla de la temporada (duración 2-3 meses)	-

2.8 Crítica al estado del arte y propuesta

En general, las webs actuales, orientadas al aprendizaje de la programación, presentan algún que otro mecanismo de gamificación. Codewars, así como Codesignal y DataCamp intentan incentivar al usuario de múltiples maneras para que siga utilizando la aplicación. Además de estas plataformas, existen otras que también persiguen premiar al usuario por los logros que consiguen como, por ejemplo, CodinGame[5] o CodeAcademy[6]; unos sitios web con objetivos similares a las tres webs anteriores, los cuales no se han incluido en el análisis previo, pero sí se han tenido en cuenta. También se han considerado otras páginas web bastante conocidas pero que, sin embargo, no presentan mecanismos de gamificación o no lo tienen muy desarrollado. Entre ellas podemos encontrar HackerRank[7], SoloLearn[8], freeCodeCamp[9] y HackerEarth[10]. Esta carencia de mecanismos de gamificación resulta curiosa pues, desde 2015, empezaron a aparecer indicios de que la utilización de gamificación influye en el rendimiento de forma positiva en estudiantes de informática [43].

En segundo lugar, en cuanto a monetización se refiere, los sitios webs basados en el aprendizaje de la programación carecen, en su mayoría, de esta funcionalidad. Sin embargo, la monetización puede resultar ser una buena herramienta de gamificación, la cual incentiva el uso de la aplicación para obtener monedas que serán utilizadas para desbloquear contenido en la plataforma [44]. En suma, el uso de una moneda de cambio para intercambiar cosas de valor es uno de los 10 principios de gamificación de Mark van Diggelen, 2012, citado en [45]. Por estos motivos, se ha decidido analizar también aplicaciones alejadas de la temática y objetivo de ASys como Forest y Brawl Stars; donde la monetización con monedas virtuales está más desarrollada y se puede extraer más información.

Como consecuencia a lo enunciado en los dos párrafos anteriores, este TFG propone la implementación, en ASys, de un sistema de gamificación para los usuarios cuyo pilar principal será la monetización. Esta funcionalidad se verá apoyada por diferentes elementos utilizados en la gamificación y probados en diversos estudios empíricos, como recompensas, tablas de

[5] https://www.codingame.com/start
[6] https://www.codecademy.com
[7] https://www.hackerrank.com
[8] https://www.sololearn.com
[9] https://www.freecodecamp.org
[10] https://www.hackerearth.com

clasificación, logros e insignias, feedback o retroalimentación, sistema de niveles [46] y desbloqueos [44].

Detallando un poco más la propuesta, la idea se basa en la inserción de dos unidades monetarias: Monedas (Coins en inglés) y Láureas. Al completar ejercicios, tareas, tests, cumplimentar logros o proporcionar *feedback* a los ejercicios, se ganará experiencia (XP), Monedas y Láureas. La experiencia se utilizará para medir el progreso del usuario mediante un sistema de niveles. Las Monedas, por otro lado, se utilizarán para comprar y desbloquear contenido en la plataforma. Entre las cosas que se pueden comprar con esta unidad monetaria destacan, por un lado, elementos estéticos, como por ejemplo avatares o emblemas y, por otro lado, aspectos más funcionales como pistas en ejercicios, potenciadores que aumentan los Coins obtenidos o desbloquear contenidos inicialmente inaccesibles pero que tampoco supongan un bloqueo o impedimento para el usuario. Por otra parte, se considera importante que aquellos estudiantes que logren ciertos logros académicos se vean recompensados mediante una segunda unidad monetaria denominada Láurea. Las Láureas son una recompensa más valiosa y difícil de conseguir (mediante logros avanzados, dieces en ejercicios, tareas específicas o cualquier otra actividad que determine el profesor) pero que ofrecerá objetos más valiosos al ser canjeada. En suma, la mayoría de los elementos canjeables por ambas unidades monetarias se deberían poder adquirir en una nueva pantalla que represente la tienda de la aplicación. Aquí, habrá objetos fijos, así como elementos temporales que irán rotando cada cierto tiempo.

Además, a la hora de implementar estas funcionalidades, es importante tener en cuenta una serie de aspectos importantes:

1. **Priorizar el cuidado a la rapidez**. A la hora de otorgar recompensas (XP, Monedas, Láureas o logros), el premio debería ser mayor para quienes hagan bien el trabajo ante aquellos que lo hagan rápido pero mal. Si se premia de igual manera o con mejores recompensas a aquellos que terminen antes el ejercicio, se generará un ambiente con un tipo de competitividad que no favorece el aprendizaje de los alumnos [47] .

2. **Unidades monetarias balanceadas**. Es de vital importancia crear un buen balance con las unidades monetarias. Es decir, no puede ser muy sencillo conseguirlas porque si no, los alumnos no estarán motivados a conseguirlas, ni tampoco puede ser muy complicado. Esto último es muy importante porque los alumnos podrían dejar de disfrutar de la plataforma y se centrarían en conseguir Monedas y Láureas a toda costa para poder utilizarla, lo cual podría resultar contraproducente en términos de aprendizaje [45]. La idea de Maletz [48], que podemos observar en la Figura 48, está basada en la dificultad de los videojuegos y se puede extrapolar a la dificultad de obtener dinero frente a su gasto en la aplicación.

3. **Histórico de gastos/ganancias**. Esta funcionalidad puede resultar bastante interesante para quienes administren la aplicación ya que pueden realizar un seguimiento del dinero de cada usuario, sus compras y sus ganancias. Esto servirá, por un lado, para facilitar el balance de la economía al ver que objetos se compran más, cuáles menos y, si el gasto es muy superior a la ganancia o viceversa. Por otro lado, servirá para detectar posibles errores que puedan beneficiar o perjudicar injustamente a ciertos usuarios.

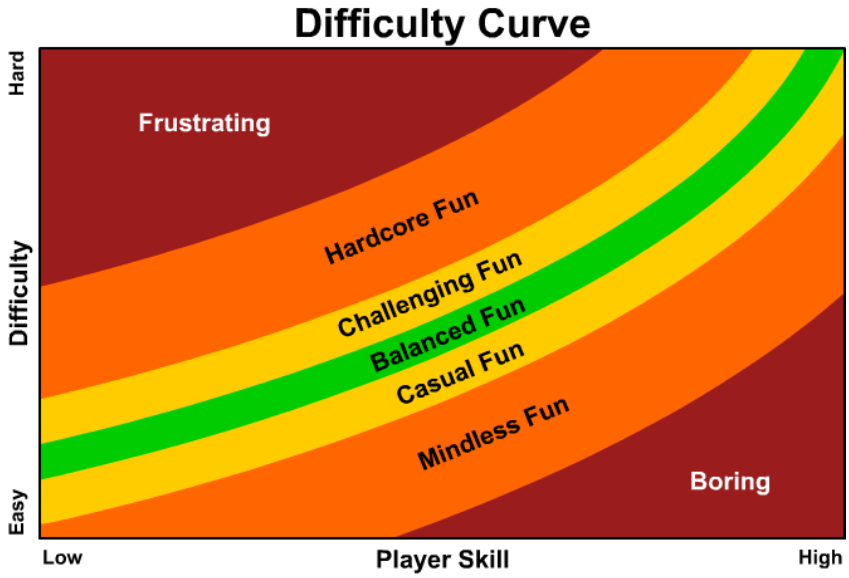


Figura 48. Curva de dificultad de un videojuego [48].

Para concluir, es importante comentar que la propuesta que se ha expuesto no se implementará en su totalidad en este trabajo, puesto que requerirá de diversos años de desarrollo, algo que excede el límite de tiempo y esfuerzo del proyecto, pero que servirá como guía para futuros trabajos. En el presente TFG se comenzará a desarrollar parte de este sistema de monetización, donde las primeras funcionalidades a diseñar e implementar se obtendrán tras una elicitación de requisitos. Además, dentro de la especificación de requisitos se pasará un segundo filtro para determinar qué funcionalidades quedarán como requisitos futuros y cuáles se implementarán en este proyecto.

3 Especificación de Requisitos Software

3.1 Introducción

En esta sección se describen y detallan las condiciones que debe cumplir la aplicación ASys una vez implementado el módulo de la monetización. Este módulo comprende el funcionamiento de la tienda, del inventario, el uso de los objetos comprados en la tienda y la obtención de monedas. Al tratarse de un producto software, para conseguir detallar y describir lo mencionado anteriormente, se va a seguir el estándar IEEE 830 [49].

3.1.1 Propósito

El propósito de esta Especificación de Requisitos Software (ERS) es detallar y especificar qué requerimientos debe cumplir el nuevo módulo de ASys. Esto incluirá requisitos funcionales y no funcionales. Esta documentación está dirigida a profesores, alumnos y a futuros desarrolladores de este sistema.

3.1.2 Ámbito del módulo

En el estado actual de ASys, una vez que completas un ejercicio no obtienes ninguna recompensa. Al principio, tras resolver unos pocos ejercicios, puede resultar insignificante obtener algún premio, pero si el objetivo es motivar al usuario a que utilice de manera periódica la aplicación, es necesario incentivarlo de cierta manera. El módulo de monetización pretende ser el semillero de la gamificación en el sistema ASys. Para comenzar este módulo, es preciso crear una nueva zona que corresponderá a una tienda, donde los usuarios podrán comprar objetos mediante unidades monetarias (Monedas y Láureas). Estos objetos serán, por un lado, estéticos (fotos de perfil, por ejemplo) y, por otro lado, funcionales (pistas en ejercicios, por ejemplo). Sin embargo, una vez comprados los objetos, será necesario poder visualizarlos y utilizarlos y, es por ello, que habrá que desarrollar una zona denominada inventario, específica para este fin. El inventario se incluirá dentro del módulo de usuarios, pues los objetos serán propios de los mismos. Por último, la obtención de las Monedas y Láureas, en este proyecto, se limitará únicamente a la realización de ejercicios. En suma, otras limitaciones del presente trabajo consisten en que la única forma de gamificación será la obtención de unidades monetarias y la tienda será el único lugar donde se podrán gastar.

3.1.3 Definiciones, acrónimos y abreviaturas

- **Back-end:** es la parte que recoge la información procedente del cliente y la procesa con el fin de interactuar con el usuario y la base de datos. Esta parte se integrará en un servidor web.
- **Base de datos:** lugar físico donde se almacena información, permitiendo así la persistencia de los datos.
- **Database**: base de datos en inglés. Ver definición de "Base de datos".
- **ERS**: Especificación de Requisitos Software.
- **Framework**: es una estructura base utilizada como punto de partida para elaborar un proyecto software con objetivos específicos.
- **Front-end:** es la parte del software que visualiza el usuario en el navegador web.
- **IEEE:** Institute of Electrical and Electronics Engineer (Instituto de Ingeniería Eléctrica y Electrónica). Es una asociación mundial de ingenieros dedicada a la estandarización y el desarrollo en ámbitos técnicos.
- **HTTP:** HyperText Transfer Protocol (protocolo de transferencia de hipertexto) [50]. Es el protocolo de comunicación que permite la transición de información en la World Wide Web (WWW).
- **HTTPS:** HTTP seguro.
- **Diseño responsive**: es una técnica de diseño y desarrollo cuyo objetivo es adaptar la visualización de las páginas web al dispositivo que se esté utilizando para visionarlas.

3.1.4 Referencias

[49] "IEEE Recommended Practice for Software Requirements Specifications," *IEEE Std 830-1998*, pp. 1–40, Oct. 1998, doi: 10.1109/IEEESTD.1998.88286.

[50] J. F. Kurose and K. W. Ross, *Redes de computadoras : un enfoque descendente*, 5ª ed. Madrid: Prentice Hall, 2010.

[51] A. Maya Gomis, "Diseño y Desarrollo del sistema ASys con Spring y Vue.js," Valencia, 2019.

[52] J. Nielsen, "10 Usability Heuristics for User Interface Design," Apr. 24, 1994. https://www.nngroup.com/articles/ten-usability-heuristics/#poster (accessed Jul. 18, 2021).

[53] MDN Contributors, "NodeList.prototype.forEach() - Web APIs | MDN," Jun. 15, 2021. https://developer.mozilla.org/en-US/docs/Web/API/NodeList/forEach#browser_compatibility (accessed Jul. 18, 2021).

[54] A. Rady, "Using forEach for IE 11," Nov. 04, 2019. https://rimdev.io/foreach-for-ie-11/ (accessed Jul. 18, 2021).

[55] F. Paredes Borras, "Diseño e implementación de un sistema online de gestión de grupos de programadores," Valencia, 2020.

3.1.5 Visión general del documento

El siguiente apartado de la ERS constituye el resto de esta, siendo constituido por seis subapartados. El primero relaciona los requisitos de un sistema mayor (ASys en este caso) con la funcionalidad del producto descrito en esta especificación de requisitos. Para acompañar este primer apartado, se utilizarán diagramas de bloque, contexto y dominio. En segundo lugar, se definirán las principales funciones del producto a desarrollar, seguido de los actores que participarán y, finalmente, las funciones detalladas de este nuevo módulo representadas mediante casos de uso. Después, se describirán brevemente las principales características de los usuarios de ASys. Seguidamente, se definirán las diferentes restricciones de todo tipo que presenta el desarrollo e implementación de este módulo. Como penúltimo punto, se plantean los requisitos futuros que, por falta de tiempo, se implementarán en proyectos posteriores. Para acabar, se enunciarán, junto a una breve descripción, los atributos de calidad que ha de cumplir el sistema tras implementar este nuevo módulo.

3.2 Descripción general y requisitos

3.2.1 Perspectiva del producto

El módulo software de la monetización forma parte de la aplicación ASys. Su función principal será proporcionar un aliciente a los usuarios para realicen los ejercicios de programación de la mejor forma posible, pues cuanto más complicado sea el ejercicio y mejor se resuelva, mayor será la cantidad de Monedas y Láureas ganadas. Por lo que se podrá comprar más objetos. Una vez comprados los objetos, deberán poder visualizarse y utilizarse en la sección de inventario, incluida en el módulo de usuarios. Es por ello por lo que el módulo de monetización deberá interactuar con el módulo de usuarios. En suma, el módulo de corrección de ejercicios deberá interactuar con el módulo de monetización para comunicarle la nota obtenida y, por ende, la recompensa de Monedas y de Láureas ganadas.

Por otro lado, a pesar de que no se incluyan en el ámbito de este TFG, es interesante mencionar las interacciones entre el módulo de corrección de ejercicios con el módulo de *hall of*

fame y el módulo de insignias, los cuales recibirán también información sobre el ejercicio corregido.

Para visualizar mejor esto, a continuación, se mostrará un diagrama de bloques (ver Figura 49) y, para mejorar el entendimiento del dominio del problema, así como conceptos importantes (entidades que se deben manipular o sobre los que se registra información y eventos que ocurren en el contexto del sistema), se presentará también un diagrama de dominio (ver Figura 50). Otro diagrama importante para poder comprender los límites del sistema a desarrollar es el diagrama de contexto, el cual se puede observar en la Figura 51. Los actores que participan en este último diagrama se pueden ver con más detalles en el apartado Diagrama de actores.

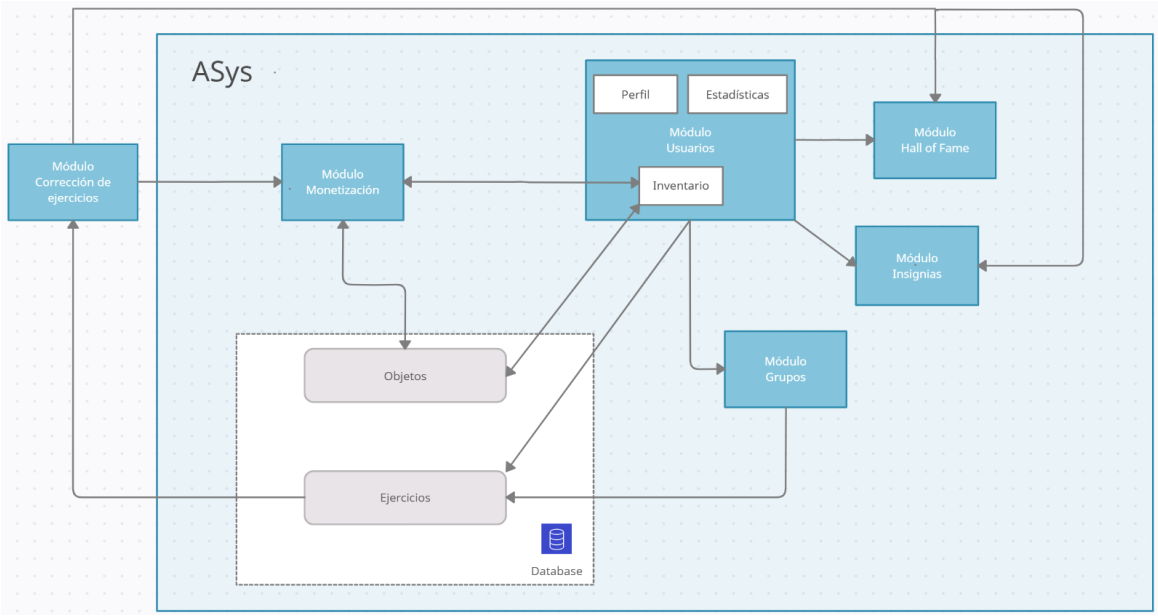


Figura 49. Diagrama de bloques de los principales módulos de ASys que participan en la propuesta de este TFG.

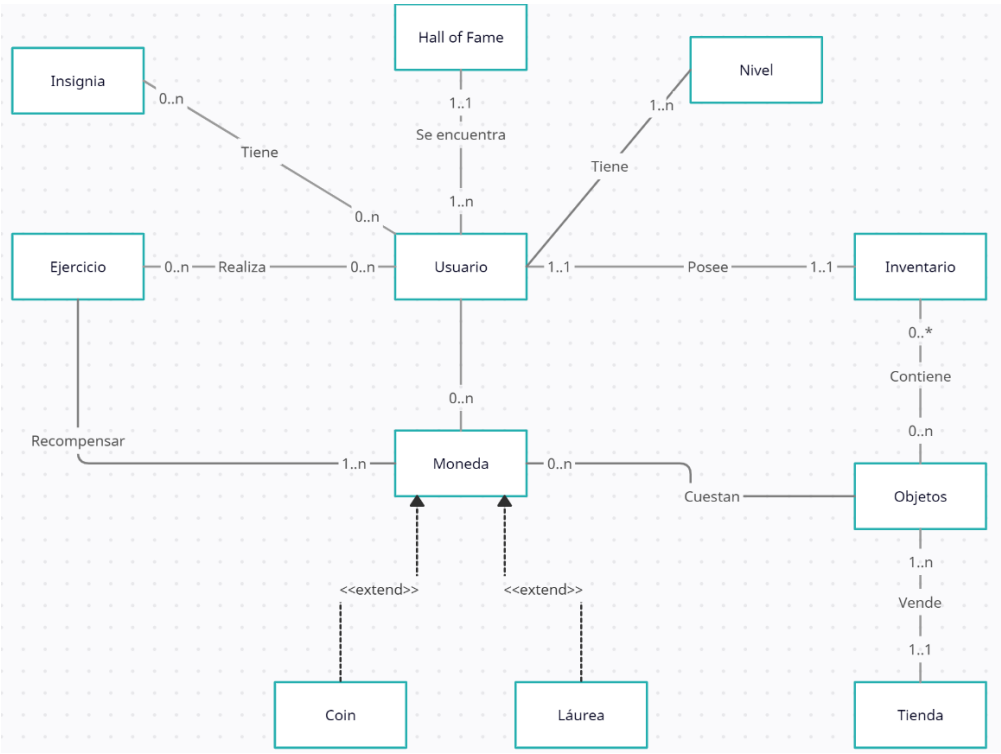


Figura 50. Diagrama de dominio con los principales conceptos que participan en la propuesta de este TFG.

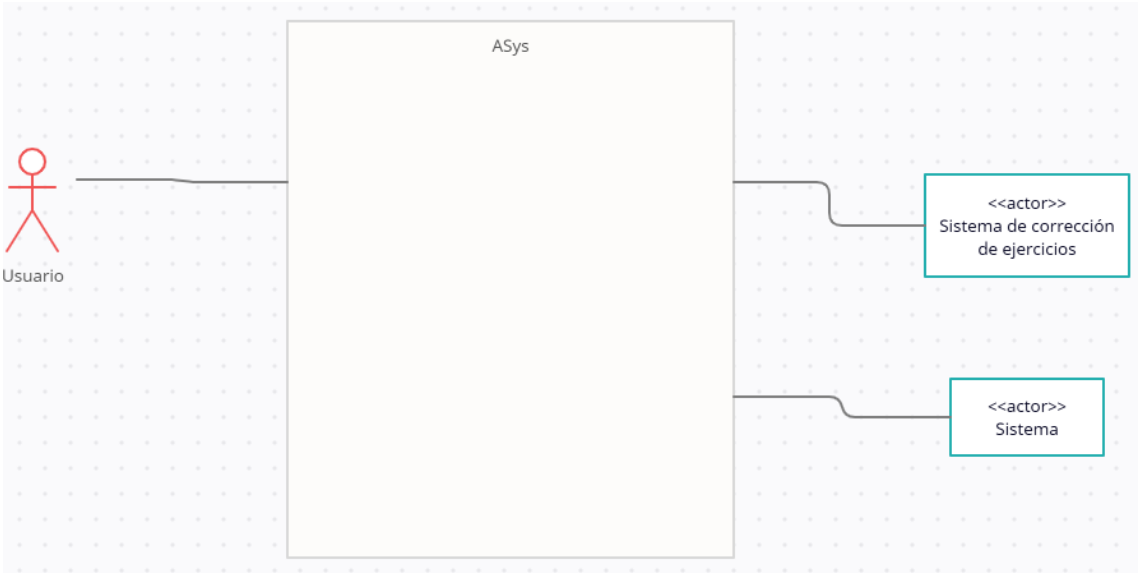


Figura 51. Diagrama de contexto de la propuesta de este TFG.

Por último, a pesar de que no se incluyan ciertas funciones en este proyecto, sí que se definirán en los apartados Funciones del producto y Requisitos futuros, a excepción de todo

aquello relacionado con el módulo del *hall of fame,* puesto que se desarrollará en paralelo a este proyecto, en otro TFG.

3.2.2 Funciones del producto

Se pretende que al final del proyecto el sistema cuente con las siguientes funcionalidades:

- **Gestión de la tienda.** Esta funcionalidad está compuesta por funciones relacionadas con la compra de objetos, el cálculo del precio de estos, y la obtención de objetos, ya sea objetos diarios o permanentes.
- **Gestión del inventario.** Esta funcionalidad se divide a su vez en dos funcionalidades; una más relacionada con la visualización de objetos y otra con el uso de estos.
    - **Visualización de objetos comprados.** Esta funcionalidad comprende aquellas funciones relacionadas con la visualización de los objetos comprados en la tienda.
    - **Utilización de objetos comprados.** Esta funcionalidad está constituida por aquellas funciones relacionadas con la activación y futura consecuencia de la activación de los objetos comprados.
- **Gestión de recompensas.** Esta funcionalidad está relacionada con las funciones encargadas de otorgar recompensas al usuario cuando este realice ejercicios. Además, estas recompensas se han de poder visualizar.

Para poder desarrollar las funcionalidades recién mencionadas y poder ahondar un poco más en ellas, en primer lugar, es necesario definir los diferentes objetos que se van a implementar y utilizar en la aplicación. A continuación, en la Tabla 10 se muestra la lista de objetos que se pretende desarrollar junto a una breve descripción sobre su función dentro de ASys.

Tabla 10 Lista de objetos que se podrán comprar en la tienda.

Nombre objeto	Descripción
Foto de perfil	Es una foto que el usuario tendrá en su perfil y representará su símbolo visual dentro de la aplicación. Esta foto será visible tanto para él como para el resto de los usuarios.
Color para el nombre de perfil	Modifica el color con el que aparece el nombre de perfil del usuario en diferentes sitios de la aplicación. Estos sitios serán principalmente las clasificaciones (o rankings) del *hall of fame* o listas de usuarios en los grupos. Este color será visible tanto para el propio usuario como para los demás.
Duplicador de Monedas temporal	Duplica la cantidad de Monedas ganadas cuando el usuario completa un ejercicio. Se activa y dura un cierto periodo de tiempo.
Triplicador de Monedas temporal	Triplica la cantidad de Monedas ganadas cuando el usuario completa un ejercicio. Se activa y dura un cierto periodo de tiempo.
Pista	Indicio que se solicita durante la realización de un ejercicio. Ayuda al usuario a deducir como realizarlo.
Desbloquea poder pedir pistas en ejercicios	Para poder comprar las pistas y utilizarlas en ejercicio, el usuario tendrá primero que comprar este objeto cuyo objetivo es desbloquear la funcionalidad de pistas en la aplicación. Este objeto no ocupa espacio de inventario.
Aumentar la capacidad máxima de inventario	El inventario tendrá un espacio máximo para almacenar los objetos comprados. Este objeto permitirá aumentar esta capacidad máxima una cierta cantidad.

3.2.2.1 Diagrama de actores

Otro punto importante, antes de comenzar a explayar este documento con los requisitos funcionales, es precisar los diferentes actores que participarán en los casos de uso. Los actores de este proyecto se pueden ver en siguiente diagrama de actores, el cual vendrá acompañado de una definición de cada uno de los actores que aparecen en dicho diagrama (ver Figura 52).

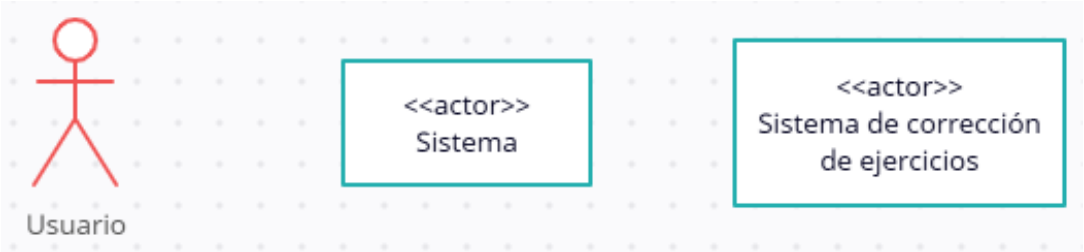


Figura 52. Diagrama de actores.

- **Usuario.** El usuario es cualquier individuo que utiliza la aplicación ASys. Este puede ser un alumno, un profesor o un administrador. Cualquiera de ellos puede utilizar este sistema de gamificación.
- **Sistema de corrección de ejercicios.** Sistema externo que se encarga de corregir los ejercicios completados y enviados por los usuarios de la aplicación. Este actor envía de vuelta a ASys información relativa a la corrección de los ejercicios. A pesar de ser un sistema externo, se posee control sobre él, puesto que ha sido desarrollado por otro alumno hace unos años [51].
- **Sistema.** Como bien indica su nombre, este actor es la propia aplicación ASys, quien se encargará de realizar, internamente, todas las operaciones necesarias para conseguir el correcto funcionamiento de la aplicación.

3.2.2.2 Diagrama de casos de uso y casos de uso

Tras haber definido los objetos a desarrollar y los actores que participarán, podemos dar pie a la elaboración de los casos de uso. A continuación, se presentará un diagrama de casos de uso para cada una de las tres funcionalidades definidas al comienzo de este epígrafe (Funciones del producto). Cada diagrama de casos de usos vendrá acompañado por las tablas de los casos de uso presentes en el correspondiente diagrama. Estas tablas describirán con más detalle las funciones a implementar en la fase de desarrollo o implementación. En primer lugar, se detallará la funcionalidad “Gestión de la tienda” (cuyo diagrama de casos de usos se puede ver en la Figura 53 y en la Figura 54), seguido de la funcionalidad “Gestión del inventario” (su diagrama de casos de uso corresponde a la Figura 55) y, finalmente, se especificará la funcionalidad “Gestión de recompensas” (en la Figura 56 podemos ver el diagrama de casos de uso).

3.2.2.2.1 Gestión de la tienda

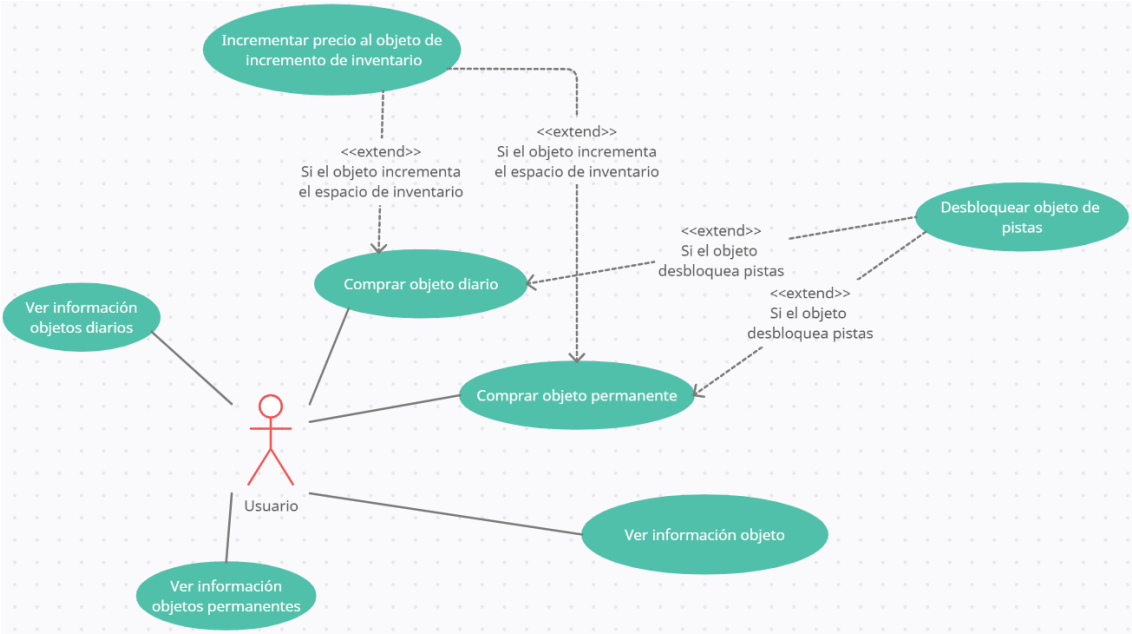


Figura 53. Diagrama de casos de uso de la funcionalidad "Gestión de la tienda". Parte 1

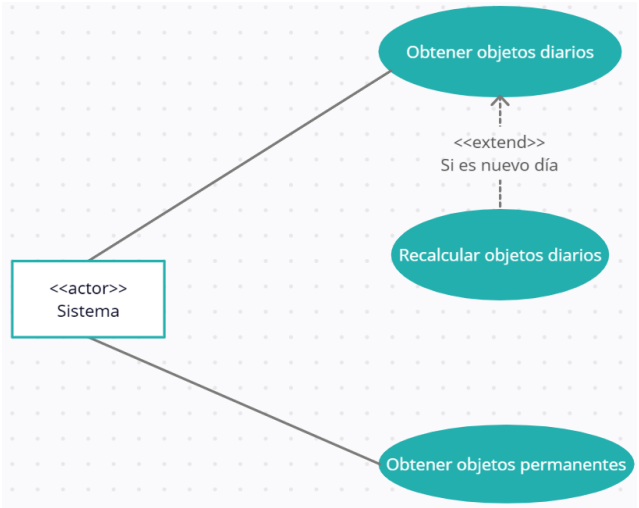


Figura 54. Diagrama de casos de uso de la funcionalidad "Gestión de la tienda". Parte 2.

UC - 1. Comprar objeto diario

Referencia	UC - 1
Nombre	Comprar objeto diario
Descripción	Compra un objeto diario gastando monedas (coins) o láureas. Una vez comprado, no se podrá volver a comprar. Si ocupa espacio de inventario, el espacio de inventario disponible disminuye la cantidad que ocupe. También se decrementa la cantidad de láureas o coins igual a su precio.
Actor	Usuario
Relaciones	Tiene extensión al UC - 3 Tiene extensión al UC - 6
Precondición	El usuario tiene fondos suficientes. El usuario no lo ha comprado aún. Hay espacio suficiente en el inventario.
Comentarios	-

UC - 2. Comprar objeto permanente

Referencia	UC - 2
Nombre	Comprar objeto permanente
Descripción	Compra un objeto permanente gastando monedas (coins) o láureas. Si es un objeto de tipo “contenido”, no se podrá comprar de vuelta. Si ocupa espacio de inventario, el espacio de inventario disponible disminuye la cantidad que ocupe. También se decrementa la cantidad de láureas o coins igual a su precio.
Actor	Usuario
Relaciones	Tiene extensión al UC - 3 Tiene extensión al UC - 6
Precondición	El usuario tiene fondos suficientes. Si el objeto es de tipo “contenido”, no debe haberse comprado aún. Hay espacio suficiente en el inventario.
Comentarios	-

UC - 3. Incrementar precio al objeto de incremento de inventario

Referencia	UC - 3
Nombre	Incrementar precio al objeto de incremento de inventario
Descripción	Incrementa el espacio máximo disponible del inventario la cantidad especificada en el objeto comprado.
Actor	Sistema
Relaciones	Extiende del UC - 1 Extiende del UC - 2
Precondición	-
Comentarios	-

UC - 4. Ver información objetos diarios

Referencia	UC - 4
Nombre	Ver información objetos diarios
Descripción	Muestra información sobre el funcionamiento de los objetos diarios de la tienda.
Actor	Usuario
Relaciones	-
Precondición	-
Comentarios	-

UC - 5. Ver información objetos permanentes

Referencia	UC - 5
Nombre	Ver información objetos permanentes
Descripción	Muestra información sobre el funcionamiento de los objetos permanentes de la tienda.
Actor	Usuario
Relaciones	-
Precondición	-
Comentarios	-

UC - 6. Desbloquear objeto de pistas

Referencia	UC - 6
Nombre	Desbloquear objeto de pistas
Descripción	Habilita la aparición de objetos de pistas en la tienda. Esto implica que ya podrá aparecer como objeto diario y que aparecerá de manera inmediata a comprar entre los objetos permanentes.
Actor	Sistema
Relaciones	Extiende del UC - 1 Extiende del UC - 2
Precondición	-
Comentarios	-

UC - 7. Ver información objeto

Referencia	UC - 7
Nombre	Ver información objeto
Descripción	Muestra información (qué es y su funcionamiento) sobre los objetos.
Actor	Usuario
Relaciones	-
Precondición	-
Comentarios	-

UC - 8. Obtener objetos diarios

Referencia	UC - 8
Nombre	Obtener objetos diarios
Descripción	Obtiene los objetos del día. Estos objetos permanecerán en la tienda durante todo el día hasta las 23:59:59 (hora local del usuario). Aquí puede aparecer cualquier tipo de objeto y solo solo se puede comprar una única vez en el día en esta sección.
Actor	Sistema
Relaciones	Tiene extensión al UC - 9
Precondición	
Comentarios	-

UC - 9. Recalcular objetos diarios

Referencia	UC - 9
Nombre	Recalcular objetos diarios
Descripción	Pasadas las 24 horas, los objetos diarios se volverán a calcular. Un objeto que solo se puede comprar una vez, si un usuario ya lo tiene, no podrá aparecerle. Puede seleccionarse un objeto de pista como objeto diario si estas están desbloqueadas.
Actor	Sistema
Relaciones	Extiende al UC - 8
Precondición	-
Comentarios	-

UC - 10. Obtener objetos permanentes

Referencia	UC - 10
Nombre	Obtener objetos permanentes
Descripción	Obtiene los objetos que siempre estarán disponibles en la tienda. Estos son todos los objetos excepto aquellos que aporten contenido estético (como las fotos de perfil). Si las pistas están desbloqueadas, se obtendrán también.
Actor	Sistema
Relaciones	-
Precondición	-
Comentarios	-

3.2.2.2.2 Gestión del inventario

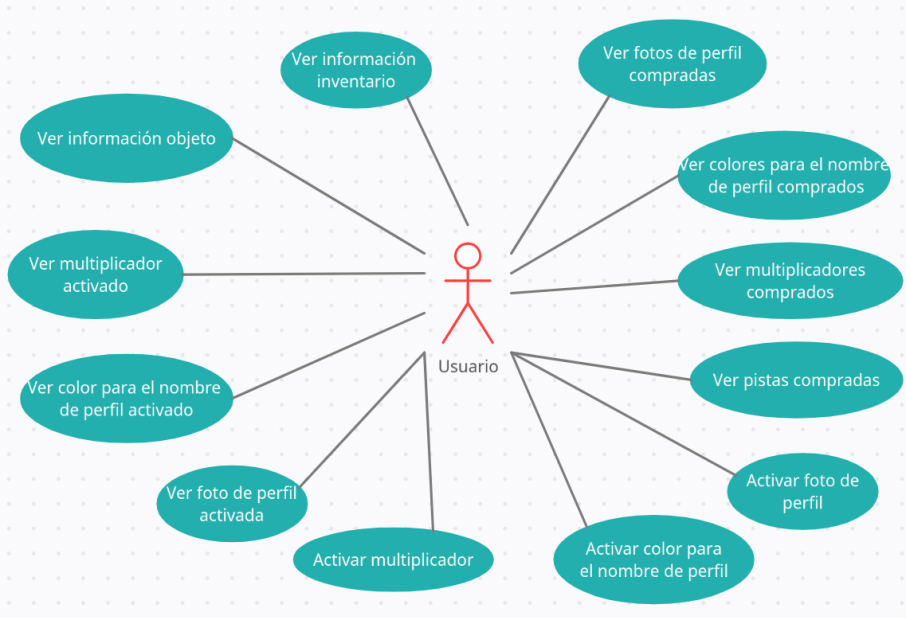


Figura 55. Diagrama de casos de uso de la funcionalidad "Gestión del inventario".

UC - 11. Ver fotos de perfil compradas

Referencia	UC - 11
Nombre	Ver fotos de perfil compradas
Descripción	Muestra la lista de fotos de perfil compradas por el usuario hasta el momento.
Actor	Usuario
Relaciones	-
Precondición	-
Comentarios	-

UC - 12. Ver colores para el nombre de perfil comprados

Referencia	UC - 12
Nombre	Ver colores para el nombre de perfil comprados
Descripción	Muestra la lista de colores para el nombre de perfil comprados por el usuario hasta el momento.
Actor	Usuario
Relaciones	-
Precondición	-
Comentarios	-

UC - 13. Ver multiplicadores comprados

Referencia	UC - 13
Nombre	Ver multiplicadores comprados
Descripción	Muestra la lista de multiplicadores comprados y disponibles para usar por el usuario.
Actor	Usuario
Relaciones	-
Precondición	-
Comentarios	-

UC - 14. Ver pistas compradas

Referencia	UC - 14
Nombre	Ver pistas compradas
Descripción	Muestra la cantidad de pistas compradas y disponibles para utilizar en ejercicios.
Actor	Usuario
Relaciones	-
Precondición	El usuario debe de haber desbloqueado el poder tener pistas.
Comentarios	-

UC - 15. Activar foto de perfil

Referencia	UC - 15
Nombre	Activar foto de perfil
Descripción	Activa la foto de perfil seleccionada. Esta foto aparecerá en las diferentes secciones de la aplicación donde se puedan ver fotos de perfil. La foto de perfil seleccionada podrá ser visualizada por otros usuarios.
Actor	Usuario
Relaciones	-
Precondición	La foto de perfil activada debe pertenecer al usuario.
Comentarios	-

UC - 16. Activar color para el nombre de perfil

Referencia	UC - 16
Nombre	Activar color para el nombre de perfil
Descripción	Activa el color para el nombre de perfil seleccionado. Este color aparecerá en la mayoría de las secciones de la aplicación donde se puedan ver los nombres de perfil (listas de usuarios, clasificaciones). El color para el nombre de perfil seleccionado podrá ser visualizada por otros usuarios.
Actor	Usuario
Relaciones	-
Precondición	El color para el nombre de perfil activado debe pertenecer al usuario.
Comentarios	-

UC - 17. Activar multiplicador

Referencia	UC - 17
Nombre	Activar multiplicador
Descripción	Activa el multiplicador seleccionado. Este durará un tiempo limitado. Al activarse se consumirá, por lo que se restará del inventario del usuario. Por ende, el usuario tendrá un espacio más en el inventario.
Actor	Usuario
Relaciones	-
Precondición	El multiplicador activado debe pertenecer al usuario. No debe haber ningún multiplicador activo en el momento de la activación (solo puede haber uno activado).
Comentarios	-

UC - 18. Ver foto de perfil activada

Referencia	UC - 18
Nombre	Ver foto de perfil activada
Descripción	El usuario podrá ver qué foto del perfil tiene activada.
Actor	Usuario
Relaciones	-
Precondición	-
Comentarios	-

UC - 19. Ver color para el nombre de perfil activado

Referencia	UC - 19
Nombre	Ver color para el nombre de perfil activado
Descripción	El usuario podrá ver qué color para el nombre de perfil tiene activado.
Actor	Usuario
Relaciones	-
Precondición	-
Comentarios	-

UC - 20. Ver multiplicador activado

Referencia	UC - 20
Nombre	Ver multiplicador activado
Descripción	El usuario podrá ver que multiplicador tiene activado y cuánto tiempo le queda.
Actor	Usuario
Relaciones	-
Precondición	Debe haber un multiplicador activo
Comentarios	-

UC - 21. Ver información inventario

Referencia	UC - 21
Nombre	Ver información inventario
Descripción	Muestra información sobre el inventario (su funcionamiento, el espacio disponible y la capacidad máxima).
Actor	Usuario
Relaciones	-
Precondición	-
Comentarios	El inventario posee una capacidad limitada para almacenar objetos, la cual se podrá ampliar comprando un objeto que incremente su capacidad máxima.

3.2.2.2.3 Gestión de recompensas

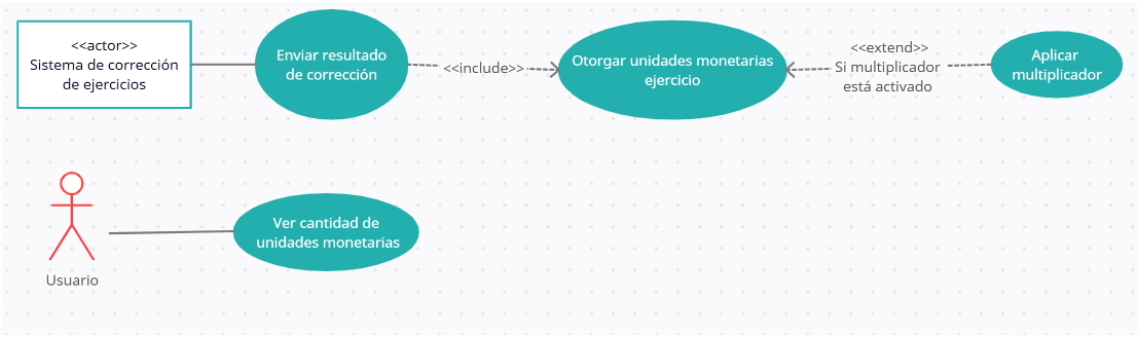


Figura 56. Diagrama de casos de uso de la funcionalidad "Gestión de recompensas".

UC - 22. Ver cantidad de unidades monetarias

Referencia	UC - 22
Nombre	Ver cantidad de unidades monetarias
Descripción	Muestra la cantidad de Coins y Láureas que posee el usuario.
Actor	Usuario
Relaciones	-
Precondición	-
Comentarios	-

UC - 23. Enviar resultado de corrección

Referencia	UC - 23
Nombre	Enviar resultado de corrección
Descripción	Envía el resultado de la corrección de un ejercicio con la nota obtenida, comentarios, información del ejercicio y errores encontrados.
Actor	Sistema de corrección de ejercicios
Relaciones	Incluye al UC - 24 Incluye al UC - 31 Incluye al UC - 32
Precondición	Debe haberse enviado un ejercicio a corregirse.
Comentarios	Este caso de uso ya está implementado. Lo que hay que hacer es añadirle funciones que llamen a los casos de usos indicados en el apartado "Relaciones". Los casos de uso UC - 31 y UC - 32 se encuentran como requisitos futuros.

UC - 24. Otorgar unidades monetarias ejercicio

Referencia	UC - 24
Nombre	Otorgar unidades monetarias ejercicio
Descripción	En base a la recompensa de Monedas del ejercicio, otorga parte de esta al usuario según la nota obtenida. Cuánta mayor sea la nota, mayor será la recompensa. Si el usuario ya había obtenido una nota en el mismo ejercicio anteriormente, entonces solo obtendrá Monedas si ha mejorado su nota. Lo mismo se aplica con las Láureas, aunque estas solo se otorgarán si se ha obtenido la máxima nota.
Actor	Sistema
Relaciones	Es incluido por el UC - 23 Tiene una extensión al UC - 25
Precondición	-
Comentarios	-

UC - 25. Aplicar multiplicador

Referencia	UC - 25
Nombre	Aplicar multiplicador
Descripción	Multiplica la cantidad de Coins obtenidas al completar un ejercicio.
Actor	Sistema
Relaciones	Extiende al UC - 24
Precondición	La recompensa del ejercicio han de ser monedas. El usuario ha de tener un multiplicador activado en el momento de finalizar el ejercicio.
Comentarios	-

3.2.3 Características de los usuarios

En ASys, existen tres roles diferentes [51] (alumno, profesor y administrador). Los conocimientos necesarios para poder utilizar este nuevo módulo no dependen del tipo de rol que posea el usuario, sino de su experiencia previa con videojuegos, y herramientas y aplicaciones con métodos de gamificación como las que se mencionaron en el Estado del arte. Se estima que los alumnos más jóvenes tendrán más experiencia y facilidad al utilizar este nuevo módulo que

los profesores y administradores. Aunque con el pasar de los años, se prevé que esta diferencia irá disminuyendo.

3.2.4 Restricciones

A continuación, se muestra la lista de restricciones que se deben tener en cuenta a la hora de desarrollar el nuevo módulo.

- Es necesario tener acceso a internet para utilizar la plataforma ASys.
- Es preciso disponer de periféricos como un monitor, teclado y ratón para ordenadores o una pantalla táctil para dispositivos móviles y tabletas.
- En el *back-end* se utilizará el lenguaje de programación Java junto al *framework* Spring[11].
- La base de datos utilizada será MySql[12].
- El *framework* utilizado para gestionar la base de datos será Hibernate ORM[13].
- En el *fron-end* se utilizará el lenguaje de programación Javascript junto a la versión 2 del framework Vue.js[14].
- La aplicación contará con un diseño *responsive* que permita la visualización en dispositivos de cualquier tamaño.
- El *front-end* y el *back-end* se comunicarán mediante el protocolo HTTP explicado en [50] .
- La interfaz debe ser usable y sencilla para que pueda ser usada por cualquier usuario, independientemente de su nivel. Para ello, seguirá los principios de Jakob Nielsen, que resume en su blog [52].
- Para mantener la estructura de la plataforma se seguirá la arquitectura cliente/servidor [50].
- Con el fin de evitar tener componentes similares o duplicados, se tienen que reutilizar los componentes ya existentes en ASys.
- La plataforma web podrá ser accedida independientemente del sistema operativo.
- El navegador Internet Explorer no será un navegador apto para utilizar ASys. Esto se debe a que el código existente hasta ahora posee funciones no soportadas por este, como

[11] https://spring.io
[12] https://www.mysql.com
[13] https://hibernate.org
[14] https://vuejs.org/v2/guide/

la función “forEach” (ver Figura 57) [53]. Sin embargo, existen posibles soluciones que podrían solventar este problema de compatibilidad, gracias a la versión 11 de Internet Explorer [54], pero este asunto se deberá solucionar en otro proyecto.

Figura 57. Fragmento de código ya existente en ASys que utiliza la función "forEach", en el fichero braceEditor.vue.

3.2.5 Requisitos futuros

Debido a la duración del proyecto, parte de los requisitos no se recogieron como casos de uso en el epígrafe Diagrama de casos de uso y casos de uso, sino que se han establecido como requerimientos futuros y, por tanto, serán excluidos de este proyecto. Sin embargo, se considera que estos requisitos son necesarios para la aplicación y han de ser implementados en un futuro. Estos casos de uso, a pesar de no poder implementarse en este proyecto, pueden tener alguna relación en el campo “Relaciones”. Estas relaciones pueden ser, tanto con otros casos de uso incluidos en este apartado, como con casos de usos que sí se desarrollarán en el presente TFG.

UC - 26. Ver nivel actual

Referencia	UC - 26
Nombre	Ver nivel actual
Descripción	Muestra el nivel que posee el usuario.
Actor	Usuario
Relaciones	-
Precondición	-
Comentarios	Se incluye en Requisitos futuros.

UC - 27. Ver experiencia actual

Referencia	UC - 27
Nombre	Ver experiencia actual
Descripción	Muestra la experiencia que posee el usuario y la restante para alcanzar el

	siguiente nivel.
Actor	Usuario
Relaciones	-
Precondición	-
Comentarios	Se incluye en Requisitos futuros.

UC - 28. Ver insignias ganadas

Referencia	UC - 28
Nombre	Ver insignias ganadas
Descripción	Muestra la lista de insignias ganadas por el usuario
Actor	Usuario
Relaciones	-
Precondición	-
Comentarios	Se incluye en Requisitos futuros.

UC - 29. Ver insignias no ganadas

Referencia	UC - 29
Nombre	Ver insignias no ganadas
Descripción	Muestra la lista de insignias que aún no posee el usuario.
Actor	Usuario
Relaciones	-
Precondición	-
Comentarios	Se incluye en Requisitos futuros.

UC - 30. Otorgar experiencia

Referencia	UC - 30
Nombre	Otorgar experiencia
Descripción	Otorga una cantidad de experiencia al usuario.
Actor	Sistema
Relaciones	Se especializa al UC - 31 Tiene una extensión al UC - 33 Es incluido por el UC - 34
Precondición	-
Comentarios	Se incluye en Requisitos futuros.

UC - 31. Otorgar experiencia ejercicio

Referencia	UC - 31
Nombre	Otorgar experiencia ejercicio
Descripción	En base a la recompensa de experiencia del ejercicio, otorga parte de esta al usuario según la nota obtenida. Cuánta mayor sea la nota, mayor será la recompensa. Si el usuario ya había obtenido una nota en el mismo ejercicio anteriormente, entonces solo obtendrá experiencia si ha mejorado su nota.
Actor	Sistema
Relaciones	Es incluido por el UC - 23 Se generaliza al UC - 30
Precondición	-
Comentarios	Se incluye en Requisitos futuros.

UC - 32. Comprobar obtención insignia

Referencia	UC - 32
Nombre	Comprobar obtención insignia
Descripción	Comprueba si el usuario cumple los requisitos para desbloquear alguna insignia.
Actor	Sistema
Relaciones	Es incluido por el UC - 23 Tiene una extensión al UC - 34
Precondición	Al usuario le queda alguna insignia por conseguir.
Comentarios	Se incluye en Requisitos futuros.

UC - 33. Subir de nivel

Referencia	UC - 33
Nombre	Subir de nivel
Descripción	Incrementa el nivel del usuario tantas unidades como "metas" de experiencia se alcancen. La "meta" de experiencia es la cantidad de experiencia que le queda al usuario para subir al siguiente nivel.
Actor	Sistema
Relaciones	Extiende al UC - 30
Precondición	No se ha alcanzado el nivel máximo
Comentarios	Subir más de un nivel a la vez podría ocurrir cuando el usuario es nivel bajo y realiza alguna actividad que otorga mucha experiencia. Se incluye en Requisitos futuros.

UC - 34. Otorgar insignia

Referencia	UC - 34
Nombre	Otorgar insignia
Descripción	Otorga la insignia ganada al usuario
Actor	Sistema
Relaciones	Incluye al UC - 30 Extiende al UC - 32
Precondición	-
Comentarios	Se incluye en Requisitos futuros.

Además, también se postergará la incorporación de un objeto para vender en la tienda. Este objeto y su descripción se muestran a continuación:

- **Insignias especiales**: Insignias que solo se pueden obtener si se compran en la tienda. Deben ser caras en comparación al resto de objetos, ya que su objetivo es demostrar que el usuario posee muchas Monedas o Láureas, ya sea porque ha realizado con éxito muchos ejercicios o porque lleva mucho tiempo usando la aplicación. Es otra forma de demostrar el prestigio.

3.2.6 Atributos del Sistema

Hasta ahora, ASys se ha desarrollado respetando una serie de requisitos no funcionales [55], los cuales han de respetarse también en el presente proyecto. A continuación, se presentan los atributos de calidad que ha de cumplir nuestro sistema:

- **Disponibilidad**: El sistema debe estar siempre disponible.
- **Escalabilidad**: El sistema debe poder funcionar sin problema ante un número elevado de usuarios.
- **Mantenibilidad**: Como consecuencia a las necesidades evolutivas, perfectivas o correctivas del software, este debe ser fácil de modificar.
- **Eficiencia**: Las peticiones al servidor de ASys no deben exceder en ningún caso los 5000 milisegundos, siendo el límite normal de respuesta de 1000 milisegundos.
- **Fiabilidad:** El sistema ha de ser capaz de tolerar los fallos que se presenten y proporcionar información adecuada para cada caso.
- **Seguridad**: Solo se permitirán peticiones autentificadas y con protocolo HTTPS.

4 Estimación de Esfuerzo y Coste

En este epígrafe llevaremos a cabo una estimación del esfuerzo y coste de trabajo que conllevará este proyecto mediante el uso de puntos de función [56]. Los puntos de función nos permiten valorar los diferentes casos de uso que hemos definido en la Especificación de Requisitos Software. Concretamente, tendremos solamente en cuenta aquellos casos de uso que sí van a desarrollarse en este TFG y, por tanto, no se incluyeron en el apartado de Requisitos futuros.

Para realizar esta estimación haremos uso del método COSMIC, un sistema de estimación que ya se había utilizado en proyectos anteriores de ASys y que es independiente de la tecnología y metodología utilizada [57].

4.1 Valoración mediante puntos de función

Mediante el uso de puntos de función recién mencionados, se valorarán los movimientos de datos generados por los diferentes procesos a analizar. Según [58], existen cuatro movimientos de datos distintos:

- **Entrada.** Movimiento de un grupo de datos desde el usuario hacia el *software*.
- **Salida.** Movimiento de un grupo de datos desde el *software* hacia el usuario.
- **Lectura.** Movimiento de un grupo de datos desde un almacenamiento persistente (en nuestro caso, una base de datos) hacia el *software*.
- **Escritura.** Movimiento de un grupo de datos desde el *software* hacia un almacenamiento de datos persistente.

En el siguiente apartado contabilizaremos los puntos de función de COSMIC (CFP) para cada caso de uso definido en la ERS y no incluido como requisito futuro. Cada movimiento de un grupo de datos se considera 1 CFP.

4.2 Valoración de la funcionalidad: Gestión de la tienda

4.2.1 UC - 1. Comprar objeto diario

- **Entrada**: Identificador del objeto e identificador del usuario.
- **Salida:** Información del usuario actualizada (con el nuevo objeto y dinero restado).
- **Lectura:** Datos del usuario.
- **Lectura:** Datos del objeto.

- **Escritura:** Usuario con el nuevo objeto y dinero actualizado.

Valoración del caso de uso: 5 CFP

4.2.2 UC - 2. Comprar objeto permanente

- **Entrada**: Identificador del objeto e identificador del usuario.
- **Salida:** Información del usuario actualizada (con el nuevo objeto y dinero restado).
- **Lectura:** Datos del usuario.
- **Lectura:** Datos del objeto.
- **Escritura:** Usuario con el nuevo objeto.

Valoración del caso de uso: 5 CFP

4.2.3 UC - 3. Incrementar precio al objeto de incremento de inventario

- **Entrada**: Identificador del objeto e identificador del usuario.
- **Salida:** Nuevo precio del objeto.
- **Lectura:** Cantidad de incrementos de inventario comprados.
- **Lectura:** Datos del objeto.

Valoración del caso de uso: 4 CFP

4.2.4 UC - 4. Ver información objetos diarios

Al tratarse de información estática y no almacenada en nuestra base de datos, no es necesaria ninguna consulta ni ninguna transferencia de datos con el almacenamiento persistente de ASys.

- **Salida:** Información de objetos diarios.

Valoración del caso de uso: 1 CFP

4.2.5 UC - 5. Ver información objetos permanentes

Al tratarse de información estática y no almacenada en nuestra base de datos, no es necesaria ninguna consulta ni ninguna transferencia de datos con el almacenamiento persistente de ASys.

- **Salida:** Información de objetos permanentes.

Valoración del caso de uso: 1 CFP

4.2.6 UC - 6. Desbloquear objeto de pistas

- **Entrada**: Identificador del usuario.

- **Salida:** Lista de objetos de categoría Pista.
- **Lectura:** Datos del usuario.
- **Lectura:** Lista de objetos de categoría Pista
- **Escritura:** Usuario con las pistas habilitadas.

Valoración del caso de uso: 5 CFP

4.2.7 UC - 7. Ver información objeto

Puesto que cuando se accede al inventario o a la tienda, hay un movimiento de datos que obtiene el objeto con todos los datos de este, la información requerida por este caso de uso se encuentra en el *software*, por lo que el único movimiento de datos necesario será uno de salida:

- **Salida:** Información del objeto.

Valoración del caso de uso: 1 CFP

4.2.8 UC - 8. Obtener objetos diarios y UC - 9. Recalcular objetos diarios

Estos casos de uso se estimarán juntos puesto que el UC-9 no tiene sentido estimarlo de forma aislada.

- **Entrada**: Identificador del usuario.
- **Salida:** Lista de objetos diarios.
- **Lectura:** Lista de objetos diarios.
- **Escritura:** Objetos diarios del usuario actualizados. Esta operación solo será necesaria si los objetos diarios obtenidos en la lectura no corresponden al día actual. Sin embargo, para estimar este caso de uso, consideraremos este peor caso.

Valoración de los casos de uso: 4 CFP

4.2.9 UC - 10. Obtener objetos permanentes

- **Entrada**: Identificador del usuario.
- **Salida:** Lista de objetos permanentes.
- **Lectura:** Lista de objetos permanentes.

Valoración de los casos de uso: 3 CFP

4.3 Valoración de la funcionalidad: Gestión del inventario

4.3.1 UC - 11. Ver fotos de perfil compradas

- **Entrada**: Identificador del usuario.
- **Salida:** Lista de fotos de perfil compradas por el usuario.
- **Lectura:** Lista de fotos de perfil compradas por el usuario.

Valoración de los casos de uso: 3 CFP

4.3.2 UC - 12. Ver colores para el nombre de perfil comprados

- **Entrada**: Identificador del usuario.
- **Salida:** Lista de colores para el nombre de perfil comprados por el usuario.
- **Lectura:** Lista de colores para el nombre de perfil comprados por el usuario.

Valoración de los casos de uso: 3 CFP

4.3.3 UC - 13. Ver multiplicadores comprados

- **Entrada**: Identificador del usuario.
- **Salida:** Lista de multiplicadores comprados por el usuario.
- **Lectura:** Lista de multiplicadores comprados por el usuario.

Valoración de los casos de uso: 3 CFP

4.3.4 UC - 14. Ver pistas compradas

- **Entrada**: Identificador del usuario.
- **Salida:** Cantidad de pistas compradas por el usuario.
- **Lectura:** Cantidad de pistas compradas por el usuario.

Valoración de los casos de uso: 3 CFP

4.3.5 UC - 15. Activar foto de perfil

- **Entrada**: Identificador del usuario e identificador del objeto (foto de perfil).
- **Salida** Información del usuario actualizada (con la nueva foto de perfil).
- **Lectura:** Datos del objeto (foto de perfil).
- **Lectura:** Datos del usuario.
- **Escritura:** Usuario con la nueva foto de perfil.

Valoración de los casos de uso: 5 CFP

4.3.6 UC - 16. Activar color para el nombre de perfil

- **Entrada**: Identificador del usuario e identificador del objeto (color para el nombre de perfil).
- **Salida:** Información del usuario actualizada (con el nuevo color para el nombre de perfil).
- **Lectura:** Datos del objeto (color para el nombre de perfil).
- **Lectura:** Datos del usuario.
- **Escritura:** Usuario con el nuevo color para el nombre de perfil.

Valoración de los casos de uso: 5 CFP

4.3.7 UC - 17. Activar multiplicador

- **Entrada**: Identificador del usuario e identificador del objeto (multiplicador).
- **Salida:** Información del usuario actualizada (con el multiplicador activado pero eliminado de su inventario).
- **Lectura:** Datos del objeto (multiplicador).
- **Lectura:** Datos del usuario.
- **Escritura:** Usuario con el multiplicador activado pero eliminado de su inventario.

Valoración de los casos de uso: 5 CFP

4.3.8 UC - 18. Ver foto de perfil activada

Puesto que cuando se activa una foto de perfil, hay un movimiento de datos que actualiza la información del usuario con esta nueva foto de perfil y, por otro lado, cuando se inicia sesión en la aplicación se obtienen los datos del usuario actualizados, la información requerida por este caso de uso se encuentra en el *software*, por lo que el único movimiento de datos necesario será uno de salida:

- **Salida:** Foto de perfil activada.

Valoración de los casos de uso: 1 CFP

4.3.9 UC - 19. Ver color para el nombre de perfil activado

Puesto que cuando se activa un color para el nombre de perfil, hay un movimiento de datos que actualiza la información del usuario con este nuevo color para el nombre de perfil y, por otro lado, cuando se inicia sesión en la aplicación se obtienen los datos del usuario actualizados, la información requerida por este caso de uso se encuentra en el *software*, por lo que el único movimiento de datos necesario será uno de salida:

- **Salida:** Color para el nombre de perfil activado.

Valoración de los casos de uso: 1 CFP

4.3.10 UC - 20. Ver multiplicador activado

Puesto que cuando se activa un multiplicador, hay un movimiento de datos que actualiza la información del usuario con este nuevo color para el nombre de perfil y, por otro lado, cuando se inicia sesión en la aplicación se obtienen los datos del usuario actualizados, la información requerida por este caso de uso se encuentra en el *software*, por lo que el único movimiento de datos necesario será uno de salida:

- **Salida:** Multiplicador activado con el tiempo restante.

Valoración de los casos de uso: 1 CFP

4.3.11 UC - 21. Ver información inventario

- **Entrada:** Identificador del usuario.
- **Salida:** Información del inventario.
- **Lectura:** Cantidad de aumentos de inventario comprados.
- **Lectura:** Cantidad de objetos que posee el usuario y que ocupan espacio de inventario.

Valoración de los casos de uso: 4 CFP

4.4 Valoración de la funcionalidad: Gestión de recompensas

4.4.1 UC - 22. Ver cantidad de unidades monetarias

Puesto que cuando se compra un objeto, hay un movimiento de datos que actualiza la información del usuario (dinero incluido) y, por otro lado, cuando se inicia sesión en la aplicación se obtienen los datos del usuario actualizados, la información requerida por este caso de uso se encuentra en el *software*, por lo que el único movimiento de datos necesario será uno de salida:

- **Salida:** Cantidad de Monedas y Láureas que posee el usuario.

Valoración de los casos de uso: 1 CFP

4.4.2 UC - 23. Enviar resultado de corrección

- **Entrada:** Resultado de la corrección del ejercicio.
- **Escritura:** Resultado de la corrección del ejercicio.

Valoración de los casos de uso: 2 CFP

4.4.3 UC - 24. Otorgar unidades monetarias ejercicio y UC - 25. Aplicar multiplicador

Estos casos de uso se estimarán juntos puesto que el UC-34 no tiene sentido estimarlo de forma aislada.

- **Entrada:** Identificador del usuario e identificador del ejercicio.
- **Salida:** Dinero (Monedas y Láureas) actualizado.
- **Lectura:** Datos del usuario.
- **Lectura:** Datos del ejercicio.
- **Lectura:** Datos multiplicador.
- **Escritura:** Usuario con el dinero actualizado.

Valoración de los casos de uso: 6 CFP

En la Tabla 11 podemos ver un resumen del recuento recién realizado, donde podemos ver que el número total de puntos de función es de 74.

Tabla 11. Resumen del recuento de puntos de función (CFP) de los casos de uso.

CASO DE USO	MOVIMIENTOS DE DATOS				CFP TOTAL
	Entrada	Salida	Lectura	Escritura	
UC - 1. Comprar objeto diario	1	1	2	1	5
UC - 2. Comprar objeto permanente	1	1	2	1	5
UC - 3. Incrementar precio al objeto de incremento de inventario	1	1	2	0	4
UC - 4. Ver información objetos diarios	0	1	0	0	1
UC - 5. Ver información objetos permanentes	0	1	0	0	1
UC - 6. Desbloquear objeto de pistas	1	1	2	1	5
UC - 7. Ver información objeto	0	1	0	0	1
UC - 8. Obtener objetos diarios y UC - 9. Recalcular objetos diarios	1	1	1	1	4
UC - 10. Obtener objetos	1	1	1	0	3

permanentes					
UC - 11. Ver fotos de perfil compradas	1	1	1	0	3
UC - 12. Ver colores para el nombre de perfil comprados	1	1	1	0	3
UC - 13. Ver multiplicadores comprados	1	1	1	0	3
UC - 14. Ver pistas compradas	1	1	1	0	3
UC - 15. Activar foto de perfil	1	1	2	1	5
UC - 16. Activar color para el nombre de perfil	1	1	2	1	5
UC - 17. Activar multiplicador	1	1	2	1	5
UC - 18. Ver foto de perfil activada	0	1	0	0	1
UC - 19. Ver color para el nombre de perfil activado	0	1	0	0	1
UC - 20. Ver multiplicador activado	0	1	0	0	1
UC - 21. Ver información inventario	1	1	2	0	4
UC - 22. Ver cantidad de unidades monetarias	0	1	0	0	1
UC - 23. Enviar resultado de corrección	1	0	0	1	2
UC - 24. Otorgar unidades monetarias ejercicio y UC - 25. Aplicar multiplicador	1	1	3	1	6
CFP TOTAL	16	22	25	9	72

4.5 Estimación del esfuerzo y presupuesto

Una vez obtenido el total de CFP, es necesario estimar el esfuerzo requerido para poder desarrollar todo lo propuesto. Para ello, es preciso determinar cuántos CFP se estima realizar en un mes y cuál es el coste de cada uno de ellos. En el desarrollo de otros módulos de ASys [59] se consideró, en un comienzo, que en un mes se pueden realizar 23 CFP donde realizar cada uno de ellos cuesta alrededor de 782€. Sin embargo, en lugar de utilizar esta referencia, nos guiaremos por su apartado referente al coste real en el apartado de conclusiones. Aquí muestran en una tabla (ver Tabla 12) el coste total de un proyecto de 28 CFP desarrollado en 8 meses de trabajo. Cinco de estos meses se dedicaron al diseño y desarrollo de la solución, pero no fueron a jornada completa sino a jornada reducida. Además, uno de estos meses no se realizó trabajo efectivo. Sin embargo, el programador encargado del proyecto ha sido pagado como si de una jornada completa se tratase, incluido el mes no trabajado. Suponiendo que esta jornada reducida

es de 20 horas semanales, el sueldo de un programador correspondería a un sueldo de media jornada, es decir 950€ (la mitad de 1.900€). Teniendo esto en cuenta, el coste de un programador a media jornada durante 8 meses reduciría el coste total de 20.409€ a unos 12.800€ aproximados, aunque hay que restarle el mes no trabajado (lo que incluye nómina y gastos de suministro), por lo que la estimación queda en 11.650€. Aunque solo cinco meses fueron los dedicados al diseño, desarrollo y pruebas, mientras que los dos primeros se destinaron a la formación. Si consideramos los últimos 5 meses del proyecto y descartamos los costes de la formación (incluyendo suministros, nómina y precio del curso), nos quedaría un coste de 9.150€ para realizar 28 CFP. Por lo tanto, desarrollar 1 CFP debería costar aproximadamente:

$$\frac{9.150\ €}{28\ CFP} \approx \mathbf{326,78\ €\ /\ CFP}$$

Por otro lado, si se tardaron 5 meses para diseñar, desarrollar y probar el proyecto a media jornada, es como si se hubiera tardado la mitad a jornada completa. En consecuencia, el tiempo necesario para realizar 1 CFP es de:

$$\frac{2{,}5\ meses * 4\ \frac{semanas}{mes} * 40 \frac{horas}{semana}}{28\ CFP} = \frac{400\ horas}{28 CFP} \approx \mathbf{14,29\ horas/CFP}$$

O bien,

$$\frac{28\ CFP}{2{,}5\ meses} \approx \mathbf{11,2\ CFP/mes}$$

Tabla 12. Costes reales asociados a los recursos utilizados. Tabla extraída de [59].

Recurso	Coste (mes)	Coste (fijo)
Programador	1.900 euros	
Usuario de pruebas de aceptación (3 días)		350 euros
Usuario de pruebas 2 (externo) (1 día)		60 euros
Usuario de pruebas 3 (externo) (1 día)		60 euros
Usuario de pruebas 4 (externo) (1 día)		60 euros
Licencias (IDE, Office 365, SO…)	10 euros	700 euros
Cursos de formación y libros (Vue y Spring)		200 euros
Portátil (Intel i7, 12 GB, 512 GB)		1.199 euros
Monitor externo, cableado y otros accesorios		500 euros
Otros gastos (suministros, materiales…)	200 euros	
Total 8 meses	20.409 euros	

Para llevar a cabo nuestra estimación de esfuerzo y coste utilizaremos las cifras recién calculadas, aunque es importante destacar, por un lado, que en nuestro caso trabajaremos a jornada completa, por lo que los costes mensuales (suministros, licencias, etc.) serán inferiores

y, por otro lado, que las cifras calculadas se basan en un único proyecto. Serían necesarios muchos más proyectos para ajustar las cifras obtenidas a un valor más real.

Respecto a la estimación de esfuerzo, si cada CFP equivale a 14,29 horas de trabajo y que nuestro proyecto posee 72CFP, necesitaremos:

$72\ CFP * 14{,}29 \frac{horas}{CFP} * \frac{1 mes}{160\ horas}$ **= 6,43 meses de trabajo,** lo que equivale a **1028,28 horas.**

En cuanto al coste necesario para desarrollar 72 CFP, teniendo en cuenta que cada CFP cuesta 326,78€, es de:

$72\ CFP * \frac{326{,}78€}{CFP}$ = **23.528,16 €**

A modo de resumen, según las estimaciones realizadas necesitaremos casi 6 meses y medio de trabajo (6,43 meses) y 23.528€ para llevar a cabo nuestra propuesta.

5 Análisis del problema

En este apartado se procederá a analizar y definir las diferentes tecnologías que se van a utilizar para el desarrollo del proyecto. Comenzaremos con las tecnologías del lado del cliente, seguidas de las tecnologías de lado del servidor. Finalmente, terminaremos este análisis tratando tecnologías más variadas, empezando por Git, que se utilizará en ambos lados (cliente y servidor) y, posteriormente, veremos tanto herramientas de edición de imágenes como páginas web con imágenes sin derechos de autor.

5.1 Tecnologías en el cliente

Actualmente, el lado del cliente de ASys está construido con tres tecnologías: HTML, CSS y Vue.js. Estas tres tecnologías se mantendrán para el desarrollo de la solución de este TFG. A continuación, se presentarán con más detalle.

5.1.1 Vue.js

Vue.js[15], también conocido como Vue, es un *framework* para el desarrollo de interfaces de usuario. Este *framework* trabaja con JavaScript y es una alternativa a los principales *frameworks* de desarrollo web como Angular o React, presentando ciertas ventajas de rendimiento y simplicidad frente a sus competidores [60]. La principal librería de Vue se centra en la capa de visualización, pero dispone de un gran número de librerías de apoyo en constante actualización que le permiten desarrollar sofisticadas aplicaciones basadas en *Single Page Applications (SPA)* [61]. Para conseguir esto, en Vue hay que crear ficheros con la extensión ".vue", los cuales se denominan componentes. Los componentes son **instancias reutilizables** de Vue con un nombre y permiten separar el comportamiento de la aplicación en múltiples ficheros, cada uno con una plantilla HTML con sus estilos CSS propios, acompañados de datos y lógica local [62], [63]. A continuación, en la Figura 58 se muestra el ciclo de vida de una instancia Vue, es decir, de cada componente de Vue.

[15] https://es.vuejs.org

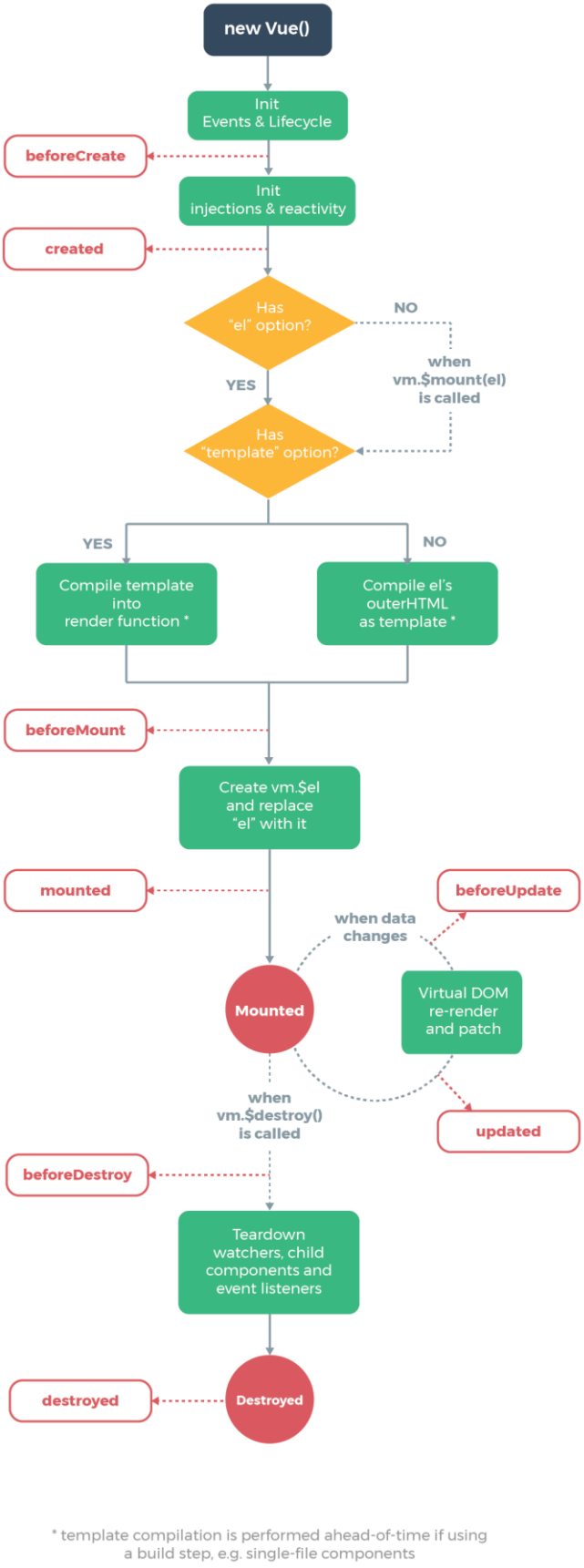


Figura 58. Ciclo de vida de una instancia Vue.

Por otro lado, cuando una aplicación comienza a crecer en tamaño, resulta complicado gestionarla ya que los diferentes componentes tienen muchas dependencias entre sí, pues poseen datos comunes (el conjunto de datos de cada componente se denomina "estado") y que termina siendo tedioso enviarlos de un componente a otro cuando empiezan a existir un gran número de estos y, es por ello, que es importante tener en cuenta la librería de Vuex (ya en uso en ASys). Además de ser una librería, actúa para Vue como un patrón de administración de estado (*state managment pattern*), extrayendo los estados de los diferentes componentes en un patrón Singleton global, centralizando la gestión de los datos [64]. De forma resumida, Vuex posee "acciones", funciones encargadas de conectar con el back-end y/o para llamar a las "mutaciones". Estas son funciones cuyo objetivo es modificar el "estado". El estado de Vuex contiene datos que son compartidos y utilizados por diversos componentes, quienes acceden a este para leer dichos datos (ver Figura 59).

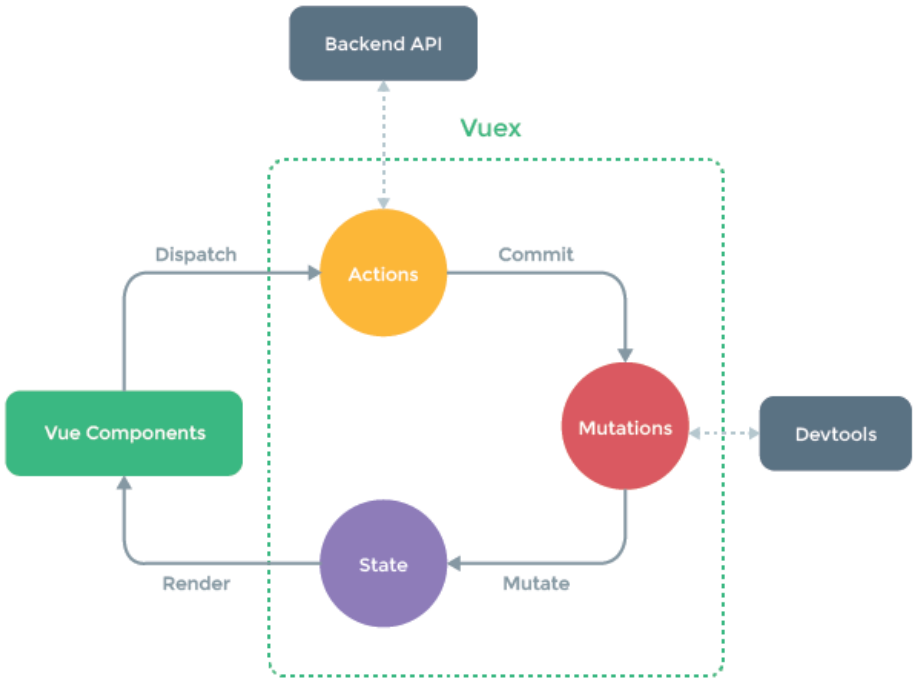


Figura 59. Diagrama de funcionamiento de Vuex.

Otra librería fundamental para construir SPA es Vue Router[16]. Su principal objetivo es facilitar el enrutamiento a las diferentes páginas de ASys. Esta librería será muy útil para encaminar las posibles nuevas páginas que se añadan.

En suma, en el sistema actual se está empleando una librería que es interesante tener en cuenta a la hora del desarrollo, denominada Vue Material[17]. Esta librería proporciona una serie de componentes prediseñados (botones, diálogos, etc.) e iconos basados todos en Material Design[18].

Por último, y no menos importante, ASys cuenta con un sistema de internacionalización que permite tener la aplicación traducida en múltiples idiomas. Esto se consigue gracias al plugin de l18n[19]. Cada idioma tiene dedicado un fichero con todos los elementos que requieren de traducción. Actualmente, solo se encuentra en castellano y en inglés, pero se puede ampliar añadiendo simplemente un fichero dedicado al nuevo idioma.

5.1.2 HTML y CSS

HTML (*HyperText Markup Language* o Lenguaje de Marcas de Hipertexto en castellano) es el componente más básico de la web, cuya función es definir la estructura y significado de esta. Para presentar esta información utiliza unas "marcas" denominadas "etiquetas" [65]. Actualmente, se está utilizando la quinta versión de este lenguaje, el cual permite añadir archivos multimedia a la web y es compatible con diseños adaptativos, entre otras cosas [66].

Para complementar HTML y otorgarle un estilo y una mejor presentación, se utiliza el lenguaje de estilos CSS (*Cascading Style Sheets del* inglés, Hojas de Estilo en Cascada). Este es un lenguaje de estilos, permitiendo otorgar, de manera selectiva, estilos a los diferentes elementos de la página web [67]. CSS, además, posee un módulo muy útil para facilitar la distribución y alineamiento de los elementos en una interfaz, que se denomina Flexbox [68]. Sin embargo, en ASys no se utiliza CSS a secas, sino que se complementa con una extensión de este denominada Sass[20], un lenguaje de estilos que otorga más funcionalidades al CSS convencional.

---

[16] https://router.vuejs.org
[17] https://vuematerial.io
[18] https://material.io/
[19] https://kazupon.github.io/vue-i18n/
[20] https://sass-lang.com

5.2 Tecnologías del servidor

El lado del servidor de ASys actual se ha desarrollado con el lenguaje Java, gracias al apoyo del *frameworks* de Spring, quien colabora junto a Hibernate y JPA para acceder a una base de datos MySQL. Por otro lado, también es importante considerar un sistema para almacenar imágenes asociadas a los objetos que se pretenden introducir en este TFG. Estas tecnologías mencionadas se considerará mantenerlas para el desarrollo de la solución de este proyecto. A continuación, se presentarán con más detalle.

5.2.1 Spring

Uno de los pilares más importantes del servidor de ASys es Spring[21], un *framework* de desarrollo de aplicaciones de código abierto para el lenguaje de programación Java. Este entorno de trabajo simplifica el desarrollo de las tareas esenciales del sistema, lo que permite al programador centrarse en tareas más específicas. Esto lo consigue gracias a múltiples servicios que ofrece. Entre ellos, destaca la inversión de control mediante la inyección de dependencias, esto es, las dependencias de los objetos se "inyectan" a estos durante su creación por Spring [69, pp. 16–25]. Relacionado con esta funcionalidad, existe una anotación denominada *Autowired* [69, pp. 69–73], la cual le permite a Spring inicializar automáticamente los objetos marcados con dicha anotación. Por defecto, se inicializarán siguiendo el patrón de diseño *Singleton* [69, pp. 56–57], pero se podría cambiar al patrón *Prototype.*

Por otro lado, otra característica de Spring muy importante empleada en ASys es la implementación del servicio REST mediante anotaciones específicas de este *framework*, como *@RestController* y *@RequestMapping*, para asignar rutas específicas al controlador, así como las anotaciones *@GetMapping* y *@PostMapping* para realizar peticiones *GET* y *POST* respectivamente [70, Ch. 6] .

Por último, en el anterior párrafo se mencionó un controlador. Esto se debe a que Spring utiliza una estructura de Modelo-Vista-Controlador (MVC) [69, pp. 11–12], la cual se detallará en el apartado de Diseño de la solución.

[21] https://spring.io/why-spring

5.2.2 Hibernate, JPA y MySQL

Para gestionar el acceso a la base de datos de ASys, se utiliza Hibernate ORM[22], un *framework* de "Mapeo Objeto/Relacional". Esto permite transformar tanto los objetos como sus atributos de un lenguaje orientado a objetos como Java a tablas de un esquema relacional. En este caso, se está utilizando una base de datos MySQL [23]. Este entorno de trabajo se complementa con la *Java Persistence API* (JPA) de Spring. El uso de estas dos tecnologías facilita el acceso a los datos, mediante el uso de anotaciones y métodos específicos. Entre ellas, se destacan las anotaciones de Hibernate capaces de generar un modelo de dominio [71, Ch. 2] y la definición de consultas de JPA [72, Ch. 6.3], que evitan el tener que escribir código SQL o HQL (lenguaje de consultas propio de Hibernate) en un gran número de casos.

5.2.3 MinIO

MinIO[24] es un servidor de ficheros que se está utilizando actualmente para almacenar ejercicios, sus respectivas instrucciones y las soluciones entregadas por los alumnos. Puesto que la propuesta de este proyecto es introducir objetos a comprar, muchos de ellos con elementos gráficos por su naturaleza, como las fotos de perfil, este servidor resultará muy útil para almacenar dichas imágenes.

5.3 Otras tecnologías y fuentes de datos

En este apartado se detallarán herramientas no relacionadas únicamente con el cliente o el servidor de ASys, sino que pueden ser útiles para ambos. También se comentarán algunas fuentes de datos de imágenes sin derechos de autor.

5.3.1 Git

Para poder registrar los cambios realizados en un archivo o un conjunto de ellos a lo largo del tiempo, es necesario utilizar un Sistema de Control de Versiones (VCS). En ASys, se utiliza Git, un Sistema de Control de Versiones Distribuido (DVCS) que se diferencia del resto de VCS mediante su forma de manejar datos, utilizando un conjunto de copias instantáneas en lugar de una lista de cambios entre versiones [73]. Junto a Git, se utiliza un repositorio remoto situado en un servidor privado de GitLab: https://kaz.dsic.upv.es/git/

[22] https://hibernate.org/orm/
[23] https://www.mysql.com
[24] https://min.io

5.3.2 Páginas web con imágenes sin derechos de autor

Tal como se explicó brevemente en el apartado de MinIO, al utilizar imágenes para los objetos, será necesario crearlas o buscarlas en internet. Si se elige la segunda opción, es fundamental que estas imágenes sean libres de derechos de autor. Algunas páginas con repositorios de imágenes que cumplen este requisito son:

- https://pixabay.com/es/
- https://www.pexels.com/es-es/

5.3.3 Photoshop y Illustrator

Al utilizar imágenes en internet, como se explica en el epígrafe Páginas web con imágenes sin derechos de autor, es posible que sea necesario modificarlas o, directamente, crear otras desde cero puesto que no se ha encontrado ninguna que satisfaga nuestras necesidades. Es por ello por lo que es importante considerar programas relacionados con la edición de imágenes como Adobe Photoshop[25] o Adobe Illustrator[26].

[25] https://www.adobe.com/es/products/photoshop.html
[26] https://www.adobe.com/es/products/illustrator.html

6 Diseño de la solución

6.1 Arquitectura del sistema

Tal como se ha adelantado en diferentes apartados anteriores, ASys es un sistema ya existente, con una arquitectura ya definida. Esto quiere decir que la arquitectura que se va a emplear en este proyecto de final de grado será la misma ya construida.

ASys está basado en una arquitectura cliente-servidor, es decir, un programa cliente y un programa servidor interactúan enviándose mensajes entré sí a través de Internet. El programa cliente es quien solicita los servicios, los cuales recibe el programa servidor para poder, posteriormente, devolver al programa cliente las respuestas a estos servicios [50].

Especializándonos un poco más a nuestro sistema, tenemos el lado del cliente formado, por un lado, por la aplicación web y, por otro lado, por el sistema de corrección de ejercicios. El cliente, a su vez, realiza peticiones REST al lado del servidor, quien recibe dichas peticiones, las procesa, se comunica con la base de datos o servidor MinIO. Estos devuelven una respuesta, la cual se transmite, a su vez, hacia el lado del cliente (ver Figura 60).

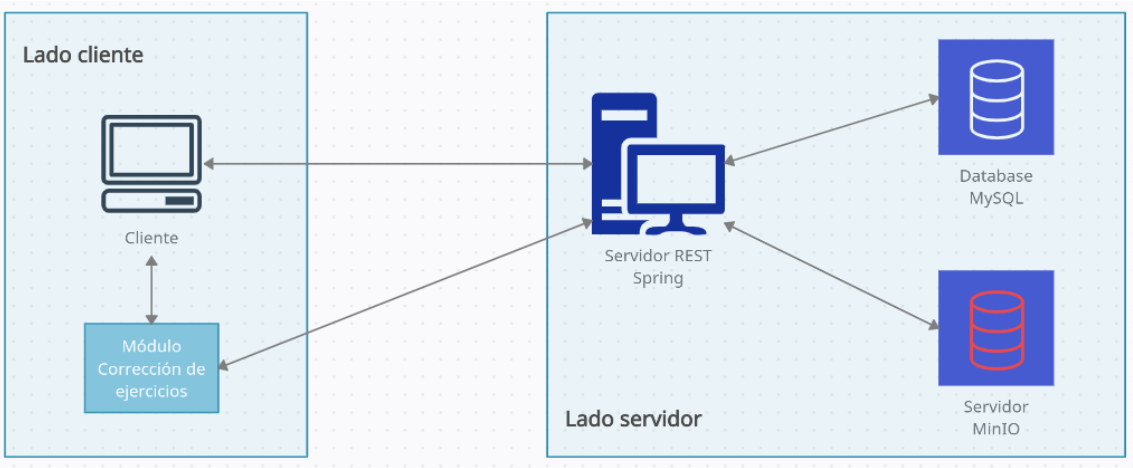


Figura 60. Esquema simplificado de la arquitectura de ASys.

6.2 Diseño detallado

6.2.1 Lado del servidor

El servidor de ASys está desarrollado principalmente en Spring, tal como se mencionó en el epígrafe anterior (Tecnologías del servidor). Además, en el apartado donde se explica Spring se anticipa que este utiliza internamente una arquitectura Modelo-Vista-Controlador (MVC). El la Figura 61 es posible apreciar la arquitectura típica de una API Spring, donde

podemos ver que cuando llega una petición REST a una URL del servidor, Spring maneja una serie de procesos para poder seleccionar el controlador implementado por el programador más adecuado para la situación. Este controlador ejecutará la lógica asociada a lo que se ha pedido, la cual accederá al modelo para poder modificar y/o consultar los datos [74].

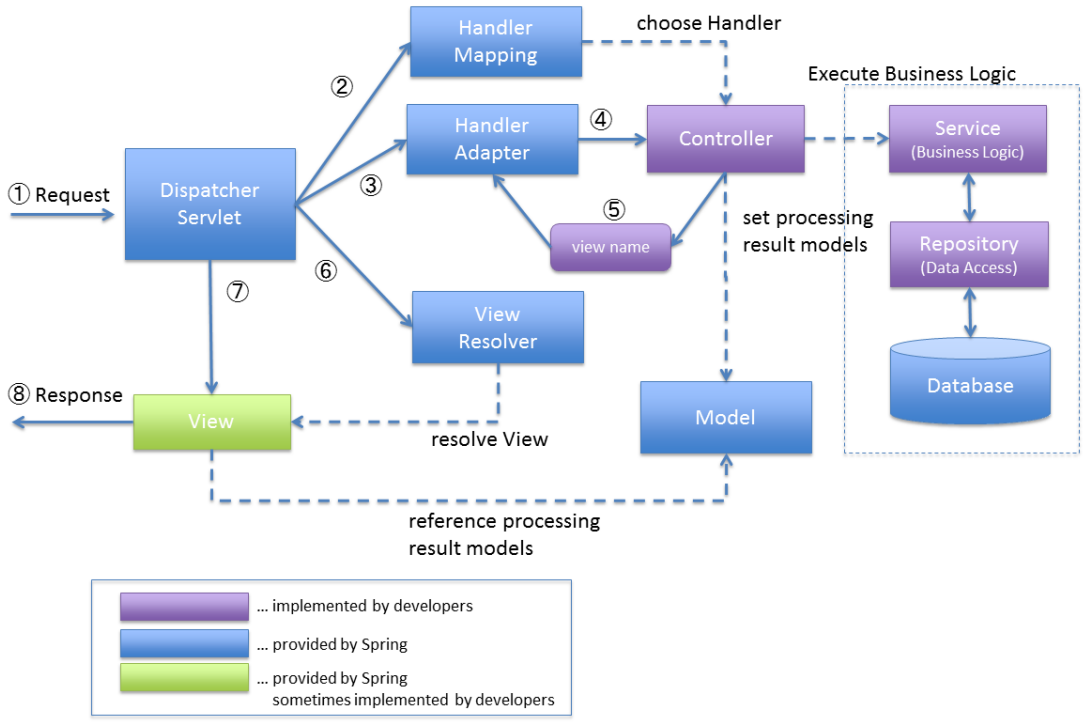


Figura 61. Arquitectura interna de una API Spring [74].

Centrándonos un poco más en la parte implementada por los programadores, podemos considerar los controladores, los servicios y la capa de acceso a datos como diferentes capas. Los primeros pueden verse como la capa de acceso para el lado cliente a los servicios del servidor. Los servicios son las diferentes operaciones o tareas que realizar a nivel de lógica. Estos servicios, muy seguramente utilizarán funciones de la capa de acceso a datos para poder insertar, modificar, eliminar o consultar los datos almacenados en la base de datos. O bien, accederán al servidor de recursos. La Figura 62 muestra de forma gráfica la arquitectura recién explicada.

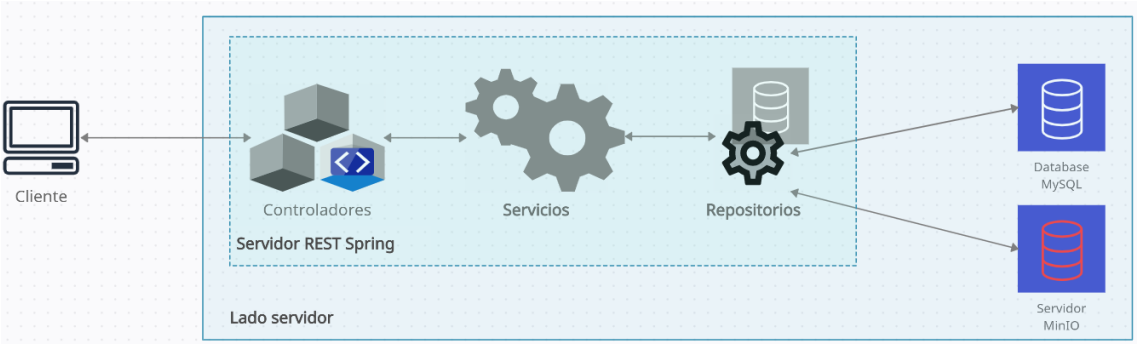


Figura 62. Arquitectura simplificada de lado del servidor de ASys.

Por último, es importante destacar que al ser el módulo de monetización un nuevo módulo en la aplicación, se tendrán que crear nuevos controladores, servicios y repositorios, así como nuevas tablas en la base de datos. Sin embargo, también hay que tener en cuenta que se aprovecharán y modificarán estructuras y componentes ya existentes.

6.2.1.1 Base de datos

Respecto a la base de datos, tal como se explicó en apartados anteriores (incluido el anterior a este), ya existe una base de datos creada en MySQL, pero será necesario crear nuevas tablas, relaciones y añadir columnas a algunas tablas ya existentes. A continuación, en la Figura 63 se muestran las tablas y relaciones que participarán en este proyecto. El esquema mostrado se ha obtenido mediante la plataforma que administra la base de datos MySQL: phpMyAdmin. Puede apreciarse ligeramente que los extremos de las relaciones pueden ser un punto fino, lo que indica una cardinalidad máxima de *1*, o un punto grueso, indicando una cardinalidad máxima de *N*.

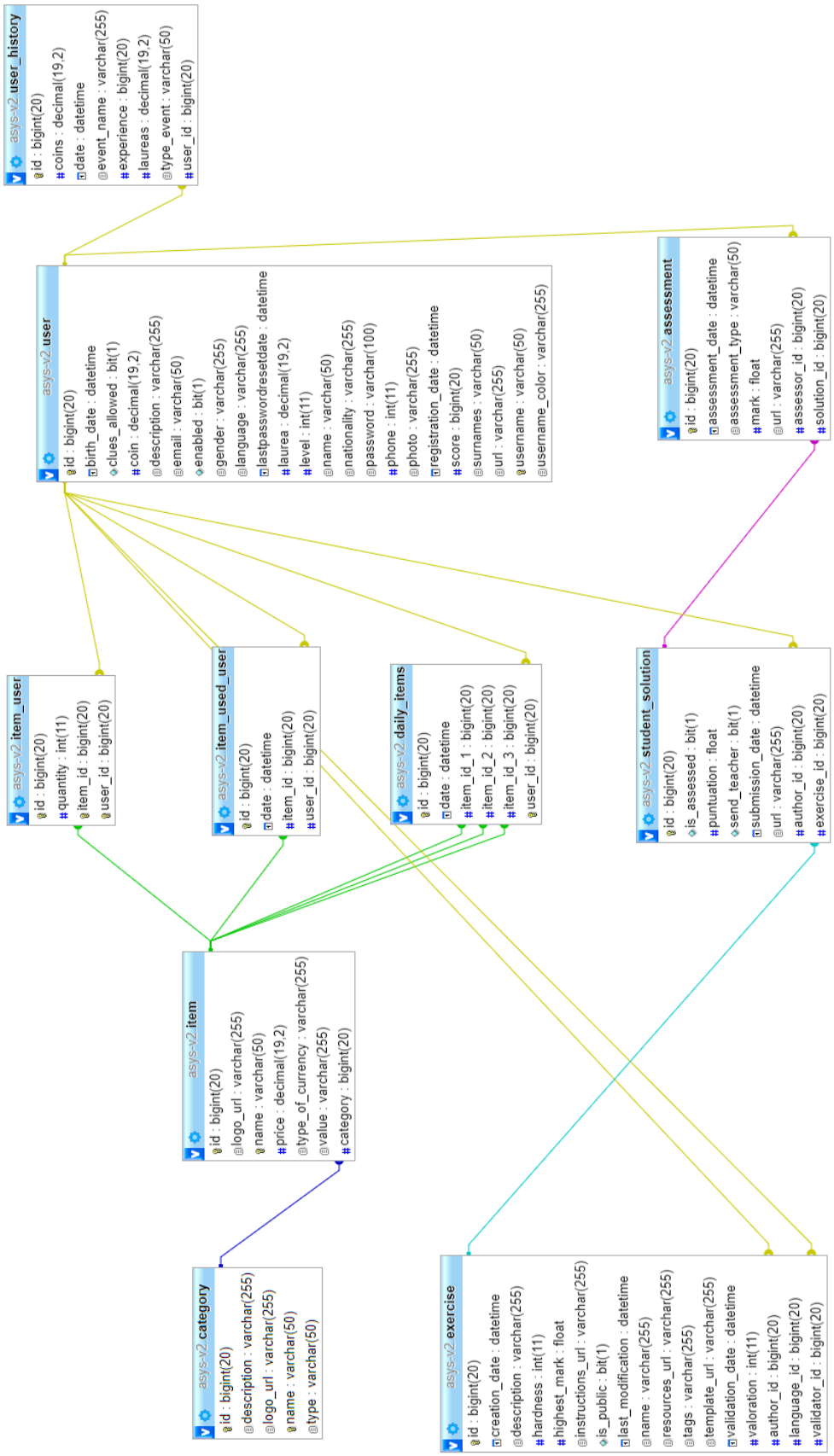


Figura 63. Fragmento del modelo relacional de ASys, con todas las tablas y relaciones que participarán en este proyecto.

6.2.1.1.1 Tablas creadas

Entre las tablas de la Figura 63, las que se van a **crear** son:

- **Item.** Esta tabla servirá para almacenar todos y cada uno de los objetos que se van a incorporar a ASys, presentes en la Tabla 10. Aquí destacamos los atributos de precio del objeto en la tienda y el tipo de unidad monetaria que indicará si se compra con Monedas o Láureas.

- **Category.** Esta tabla representa las diferentes categorías de los objetos. Se incorpora esta tabla para respetar las tres formas normales. La información que almacena sirve para clasificar los objetos y poder diferenciarlos a la hora de aplicar operaciones lógicas. Para ello, esta tabla posee dos atributos importantes: un atributo que indica el nombre y otro el tipo. Esto se debe a que cada categoría posee un nombre ("Foto de perfil", "Multiplicador", etc.) que sirve para realizar operaciones propias de cada una de estas categorías. Por otro lado, estas categorías pueden poseer comportamientos comunes y es necesario un atributo (atributo "tipo") que facilite este reconocimiento. Existen cuatro valores diferentes para este campo:

    - **Activable.** Esto indica aquellas categorías de objetos que solo se compran una vez. Estos objetos podrán activarse en el inventario y no se consumirán. Entre ellos están las fotos de perfil y los colores para el nombre de perfil.

    - **Consumible.** Los objetos clasificados en una categoría de este tipo podrán comprarse las veces que se quiera, pero cada vez que se utilice, se gastará y será necesario comprar otro. Los objetos englobados en este tipo de categoría son los diferentes multiplicadores y las pistas.

    - **Contenido.** Los objetos con una categoría de tipo "Contenido" solo podrán comprarse una única vez en la tienda y poseen un efecto permanente: desbloquear un cierto contenido en la aplicación al usuario que lo compre. Aquí encontramos el objeto que desbloquea el poder pedir pistas en la aplicación.

    - ***Upgrade* o Mejora.** Este tipo de categorías tienen un comportamiento parecido a las categorías de tipo "Contenido", pues habilitan de forma permanente un determinado contenido en la aplicación. La diferencia es que los objetos que posean este tipo de categoría podrán comprarse todas las veces que el usuario desee.

- **Daily_Items.** Cada fila de esta tabla representa los objetos diarios de un usuario presentes en la tienda. Es importante destacar que posee un atributo para cada objeto y, además, un atributo que indica la fecha a la que pertenecen esos objetos diarios; es decir, si la fecha es diferente al día actual, esos objetos ya no son útiles y será necesario recalcularlos.

- **Item_User.** Puesto que la relación que tienen las tablas "Item" y "User" es de "muchos a muchos", surge la necesidad de crear una tabla intermedia. Esta tabla es la encargada de representar qué objetos posee el usuario y equivaldrá al inventario de este. De esta tabla es importante destacar que se almacena el número de cada objeto que posee el usuario. Cuando se compra un objeto, si este no existe, se creará una nueva entrada, mientras que, si ya existiera, se incrementaría el campo "quantity". Muy útil sobre todo para objetos con categoría de tipo "Consumible" o "Upgrade".

- **Item_Used_User.** Esta tabla, al igual que la tabla "Item_User" sirve para permitir una relación "muchos a muchos" las tablas "Item" y "User" aunque con objetivo diferente. La finalidad de esta tabla es registrar los objetos que utilice el usuario (multiplicadores y pistas) para poder, por un lado, tener un seguimiento de los objetos usuarios por cada usuario y, por otro lado, para poder determinar si multiplicador está activo o no gracias al atributo fecha.

- **User_History.** Tal como se concluye en el epígrafe de Crítica al estado del arte y propuesta, tener un seguimiento de los fondos monetarios de los usuarios dentro de ASys es algo importante y necesario. Es por ello por lo que el objetivo de esta tabla es ser un histórico de las Monedas y Láureas ganadas y gastadas por los usuarios. Esta tabla también aprovecha para registrar la experiencia ganada. Un atributo muy importante y que permite realizar este seguimiento es el campo fecha, pues se encarga de registrar el momento en el que se ha realizado la acción. Además, esta tabla puede emplearse para determinar si un usuario ha comprado un objeto diario ese día o no, algo muy importante, ya que según el UC - 1. Comprar objeto diario, un objeto en esta sección no puede comprarse más de una vez en el mismo día.

6.2.1.1.2 Tablas que ya existían

El resto de las tablas de la Figura 63 **ya existen** en ASys pero es necesario tenerlas en cuenta, ya sea por una serie de atributos que poseen, o bien, por algunos atributos que será preciso añadir:

- **User.** La tabla de los usuarios es de las más importantes, pues toda la monetización gira a entorno a estos. La mayoría de las tablas que participan en este proyecto se relacionan con la tabla *User*. A esta tabla se le añadieron dos atributos: *Coin* y *Laurea*. Su nombre es bastante intuitivo, pues sirven para indicar la cantidad de Monedas y Láureas que posee el usuario. También se añadió el atributo *Username_color* para gestionar el color que este tiene seleccionado para el nombre de perfil.

- **Exercise.** La tabla de ejercicios no posee ningún cambio, pero es importante tener en cuenta el atributo *hardness* que representa la dificultad del ejercicio en cuestión, un factor que se utilizará para determinar la recompensa de los ejercicios.

- **Assessment (corrección en castellano).** Esta tabla sirve para indicar la nota obtenida por el alumno al resolver un ejercicio. Este dato es muy importante también para determinar la recompensa que se le va a otorgar.

- **Student_solution.** Esta tabla relaciona las dos anteriores y, del mismo modo, también es importante para el cálculo de la recompensa.

6.2.2 Lado del cliente

El lado del cliente está desarrollado principalmente en Vue.js y, tal como se analizó en su epígrafe dedicado, este *framework* posee una arquitectura formada por vistas (parte HTML y CSS encargada de mostrar la información) y componentes (estructuras reutilizables situadas dentro de las vistas). Además, Vue posee una serie de librerías que utiliza ASys y que serán muy útiles para el desarrollo de nuestro proyecto.

Entrando un poco más en detalle, el código de nuestro sistema está formado por vistas que conforman las diferentes páginas web de ASys. Estas vistas son, en términos teóricos, componentes de Vue dentro de los cuales se utilizan otros componentes en la sección de código HTML para mostrar la información, pero también posee cierto código en Javascript. Estos componentes son también archivos con la extensión ".vue" pero tienen como objetivo ser estructuras reutilizables en las diferentes vistas con una funcionalidad única. A su vez, estos componentes, incluidas las vistas, se comunican con el store de la librería de *vuex* para poder

gestionar el "estado"[27]. Cada vez que se requiera actualizarlo o se necesite consultar el servidor, los componentes llaman a las "acciones" del store. Para realizar peticiones al servidor, las acciones se comunicarán con una capa de servicio encargada justamente de realizar peticiones HTTP. Una vez obtenida la respuesta, las acciones ejecutarán las "mutaciones" correspondientes para actualizar el estado. Cabe mencionar que tanto el *store* como los servicios se particionan en diferentes módulos, uno para cada funcionalidad (ejercicios, grupos, usuario, etc). Por otro lado, para navegar a través de las diferentes vistas, estás utilizarán la librería de Vue Router. En la Figura 64 podemos observar de forma gráfica lo recién explicado.

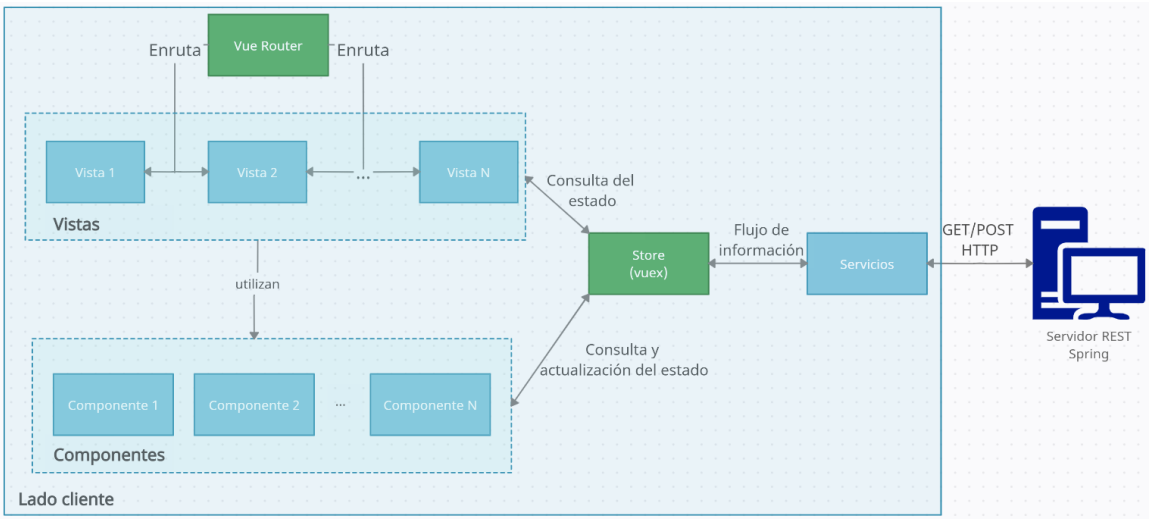


Figura 64. Arquitectura simplificada del lado del cliente de ASys.

En nuestro caso utilizaremos la arquitectura ya existente, donde tendremos que modificar ficheros ya existentes, como el *store* del módulo de usuario para gestionar el inventario, el archivo de rutas (librería Vue Router) o de ciertos componentes para añadir más información. También habrá elementos que no se modificarán pero que si se utilizarán en este proyecto con el fin de reutilizar la mayor cantidad de código posible. Sin embargo, puesto que vamos a desarrollar funcionalidades nuevas como pueden ser la tienda o el inventario, la mayor parte del código a escribir será en componentes nuevos, para representar los objetos, y un nuevo módulo del *store* y servicios, dedicados a la tienda.

[27] Las vistas también se comunican con el store, pero para realizar principalmente consultas de datos que serán útiles a múltiples componentes de la vista.

6.2.2.1 *Interfaces externas*

Las interfaces que se han desarrollado constituyen uno de los pilares de este proyecto, debido a que se tendrán que realizar desde cero en su mayoría. Para poder diseñarlas, se ha utilizado la técnica de prototipado con la elaboración de *mockups* con el fin de visualizar el trabajo deseado y validar los requisitos detallados en la Especificación de Requisitos Software. Una vez validados, estos *mockups* se utilizarán como guía a la hora de implementar la solución en el lado del cliente. Es importante recalcar que estos prototipos muestran una versión preliminar del producto que sirve principalmente para validad los requisitos con el usuario [75], por lo que la versión final de este no tiene por qué ser del todo fiel a esta primera versión de prototipos. La aplicación utilizada para su elaboración y diseño ha sido la página web https://app.moqups.com, la cual permite elaborar *mockups* sencillos de forma gratuita hasta un cierto número de elementos. Esta web se suele recomendar a los estudiantes de tercero y cuarto de carrera en la UPV en las asignaturas dedicadas a este tema. A continuación, se mostrarán los prototipos realizados.

6.2.2.1.1 Tienda

Esta pantalla (ver Figura 65) representa la futura tienda de ASys. Aquí los usuarios podrán comprar los diferentes objetos mencionados en la Tabla 10. En la parte superior aparecerán los objetos diarios que se reiniciarán cada veinticuatro horas, mientras que en la parte inferior aparecerán los objetos permanentes que siempre estarán disponibles. Por otro lado, se añadirá un acceso directo a la tienda en el menú lateral.

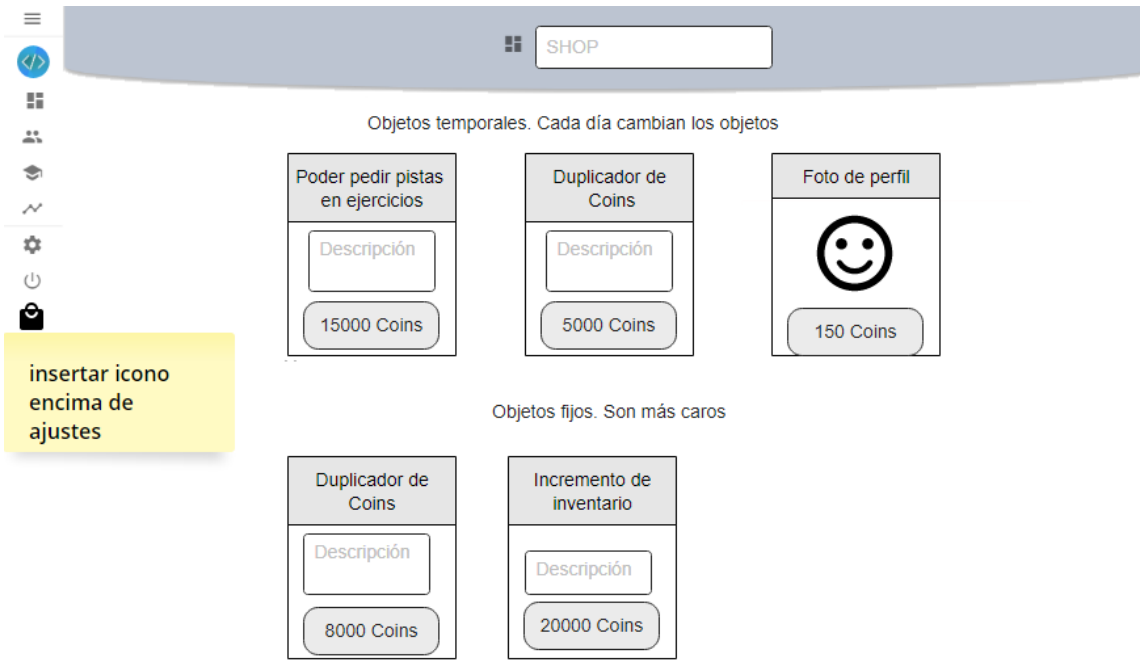


Figura 65. Mockup de la primera versión de la tienda de ASys.

Tras la validación con el usuario, la interfaz no sufrió cambios.

6.2.2.1.2 Inventario

Una vez comprados los objetos, se requiere una pantalla para poder visualizarlos y utilizarlos. Esta pantalla es la del inventario, la cual, en un comienzo, pretendía mostrar todos los objetos en una única pantalla, tal como podemos observar en la Figura 66 .

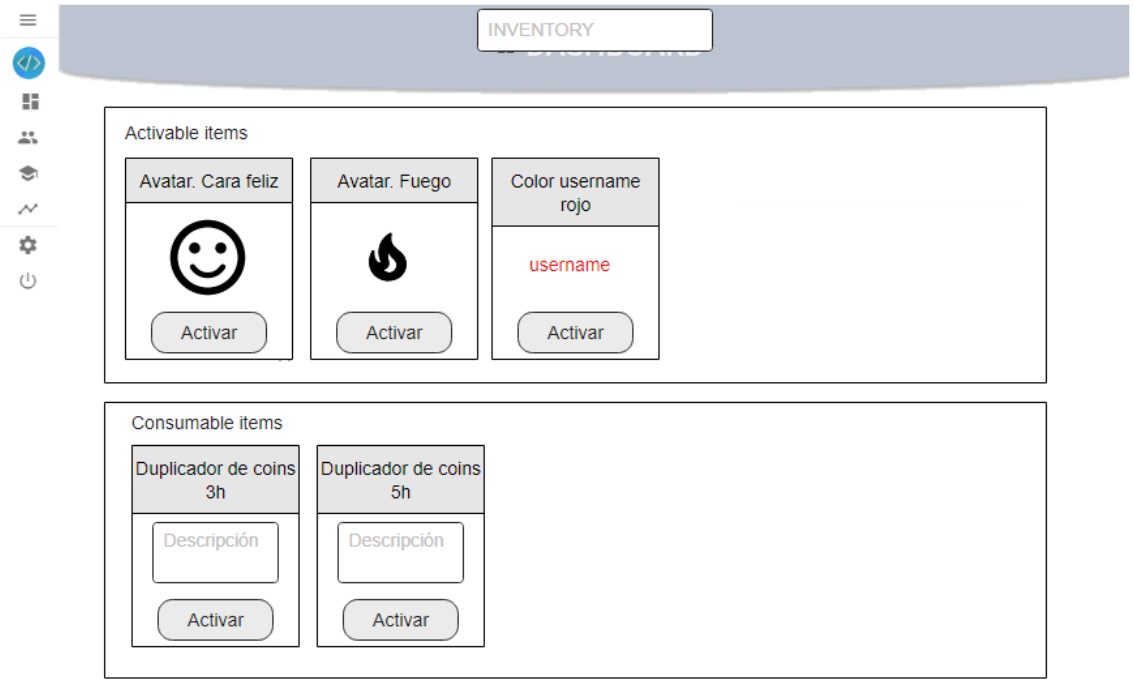


Figura 66. Mockup de la primera versión del inventario de ASys.

En este caso, el usuario sugirió que se podría crear una pantalla previa que dividiese los objetos en categorías con el objetivo de reducir el caos pues, en futuro cuando se introduzcan decenas de objetos más, la pantalla de la Figura 66 se convertiría en algo no escalable debido a su tamaño. De manera que se propuso una segunda versión que podemos apreciar a continuación, en la Figura 67.

Figura 67. Mockup de la segunda versión del inventario de ASys.

A su vez, cada botón redirigirá al usuario a una pantalla con los objetos de esa categoría. Por ejemplo, en la Figura 68 podemos ver un ejemplo de la pantalla que vería el usuario si clicase en el botón "Ver" en la categoría de avatares o fotos de perfil.

Figura 68. Segunda parte del mockup de la segunda versión del inventario de ASys. Sección donde el usuario puede activar su foto de perfil.

6.2.2.1.3 Objetos

Tanto en la interfaz de la tienda como en el inventario puede apreciarse la primera propuesta gráfica de los objetos. Esta primera versión no le convencía al usuario y comentó que los objetos podrían tener todo un logo en lugar de una descripción, y que esta se mostrase mediante un botón. En la Figura 69 se muestra cómo se ha acordado finalmente la interfaz de los objetos, tanto de la tienda como del inventario.

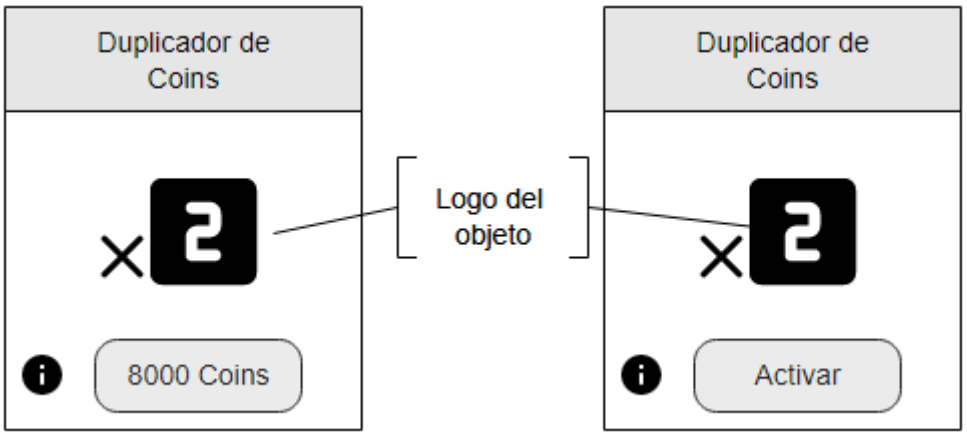


Figura 69. Mockup de la segunda versión de un objeto de ASys.

6.2.2.1.4 Recompensas de los ejercicios

En ASys existen numerosas pantallas donde un usuario puede seleccionar un ejercicio y comenzar a solucionarlo. En cada una de estas, es necesario indicar la cantidad de Monedas y Láureas puede ganar si lo resuelve. Empezando por el menú principal de los ejercicios, aquí aparecen múltiples secciones con listas de ejercicios donde es necesario indicar su recompensa (ver Figura 70).

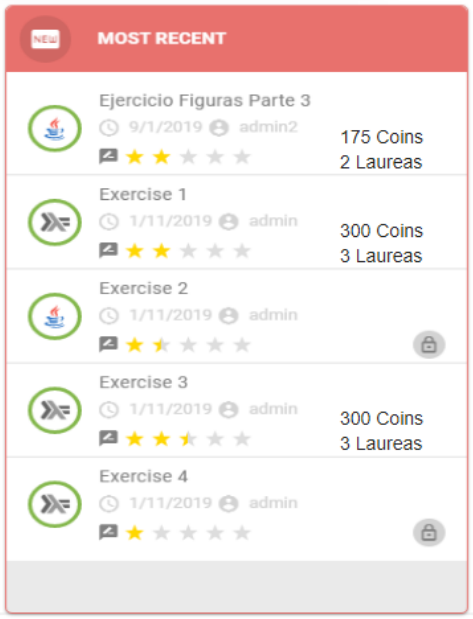


Figura 70. Interfaz de la lista de ejercicios situada en la pantalla dedicada a estos tras añadir las recompensas de estos.

Esta lista de ejercicios es visible también desde la pantalla principal de la aplicación, por lo que también será necesario mostrar las recompensas en este lugar. Puesto que la pantalla principal corresponde al *dashboard* del usuario, se ha considerado un sitio idóneo para situar el botón para acceder al inventario (ver Figura 71).

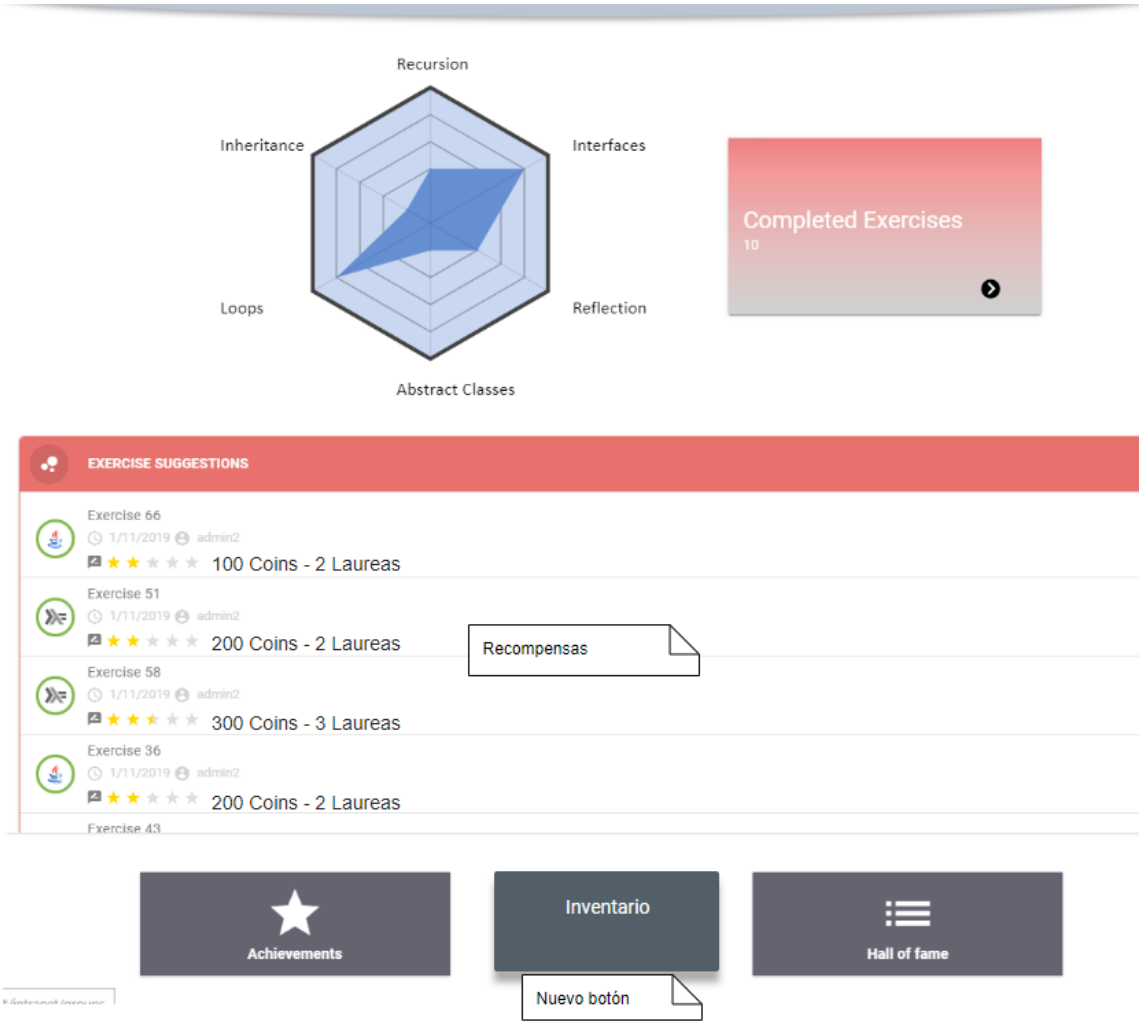


Figura 71. Interfaz del dashboard del usuario tras introducir las recompensas a los ejercicios y el botón para acceder al inventario.

Otra forma de seleccionar los ejercicios es accediendo al buscador de estos. Aquí también habrá que añadir la recompensa que otorgan los ejercicios filtrados (ver Figura 72).

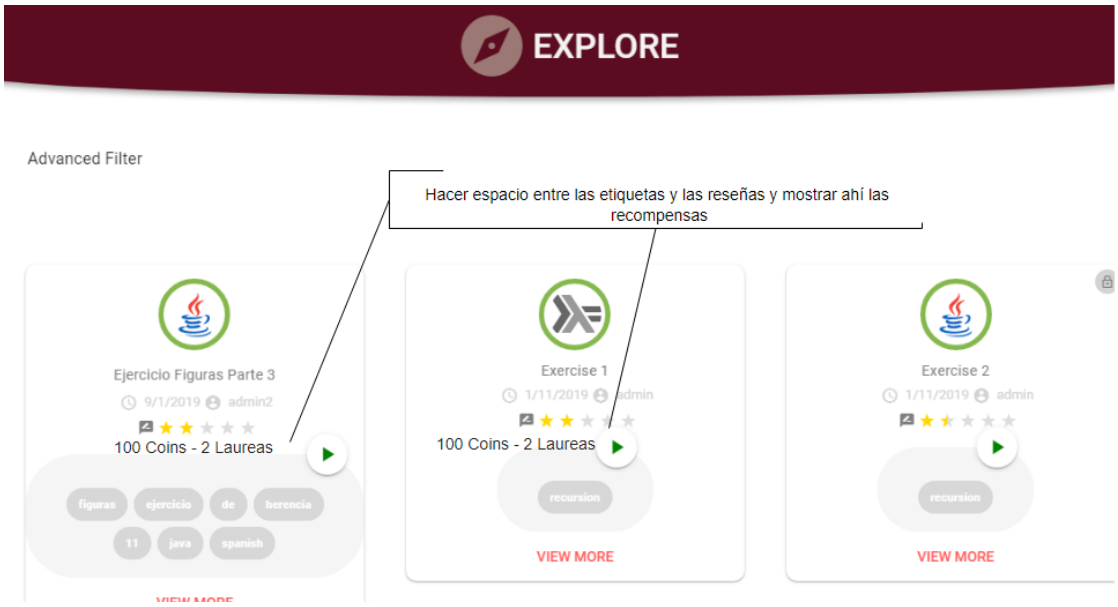


Figura 72. Interfaz de búsqueda de ejercicios a la que se le ha añadido la recompensa de estos.

Una vez seleccionado un ejercicio, aparecerá un diálogo con la información de este. Aquí también será necesario indicar que recompensa ofrece (ver Figura 73). Cabe mencionar que, en el momento de este análisis y diseño, esta interfaz está sufriendo ciertos problemas de visualización de alguno de sus elementos.

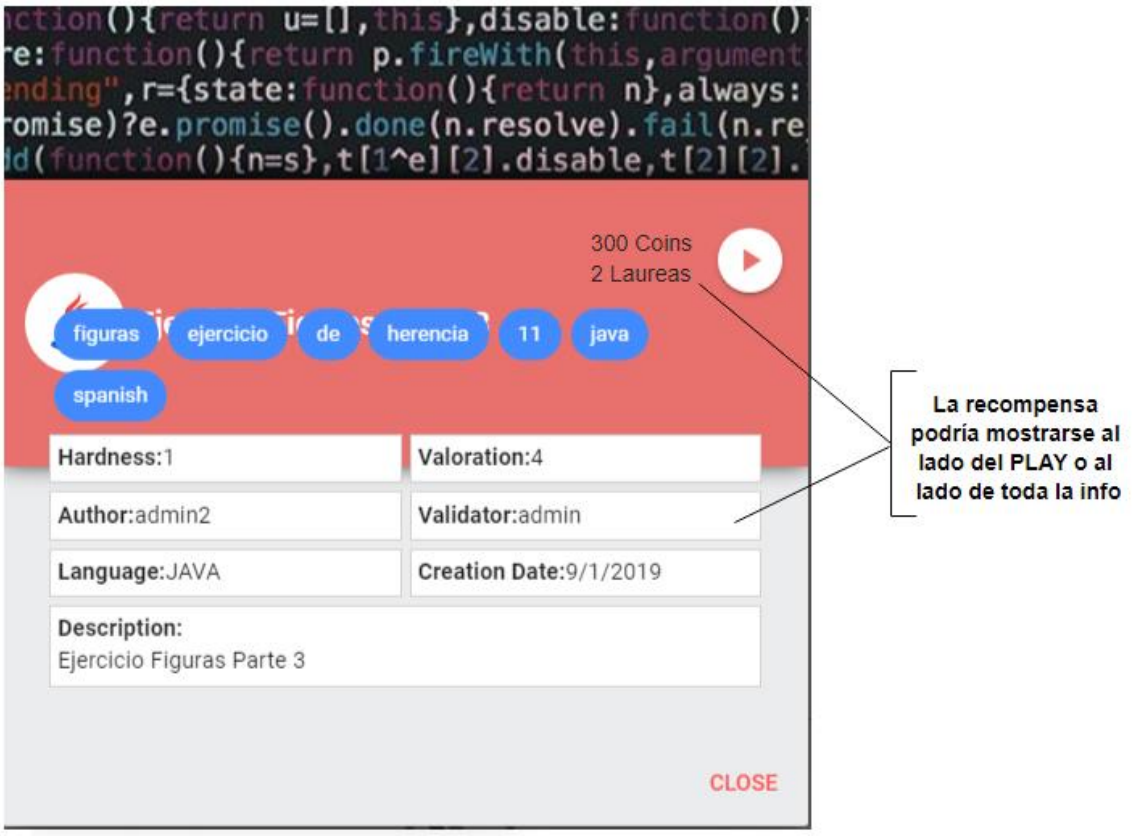


Figura 73. Interfaz del diálogo de información de un ejercicio a la que se le ha añadido dos posibles lugares para situar la recompensa del ejercicio.

Finalmente, en la pantalla dedicada al ejercicio también debería aparecer la recompensa de este, tal como podemos ver en la Figura 74.

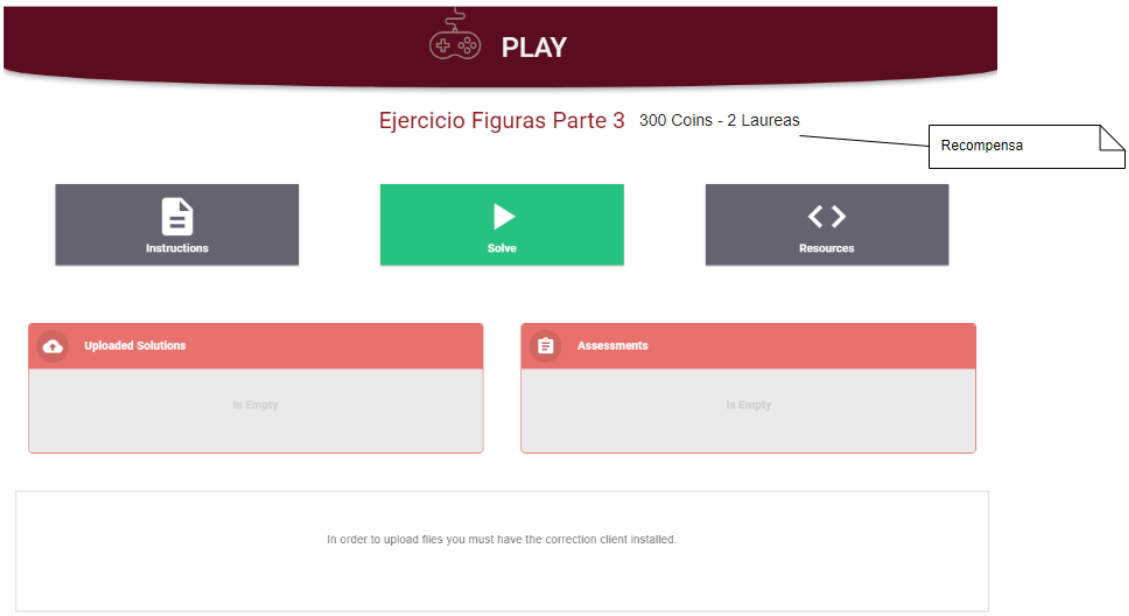


Figura 74. Interfaz de un ejercicio a la que se le ha añadido la información asociada a la recompensa que ofrece.

Todas estas interfaces han sido validadas por el usuario quien únicamente ha comentado que sería buena idea mostrar los iconos de las Monedas y Láureas (cuando se decidan) en lugar de su nombre.

6.2.2.1.5 Cantidad de Monedas y Láureas

Por último, pero no menos importante, es decidir dónde mostrar la cantidad de Monedas y Láureas que posee el usuario en cada momento. Para esto, se le propusieron dos alternativas al usuario. La primera consistía en mostrar esta cantidad en la barra del menú lateral (ver Figura 75), mientras que la segunda situaba esta información arriba a la derecha (ver Figura 76).

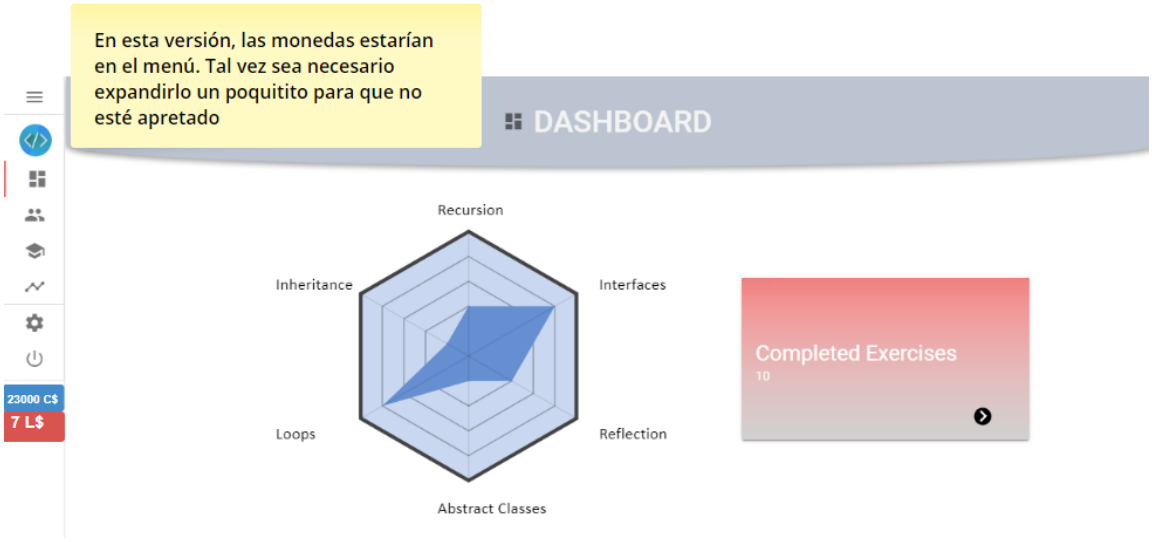


Figura 75. Mockup de una pantalla de ASys con la cantidad de Monedas y Láureas en el menú lateral.

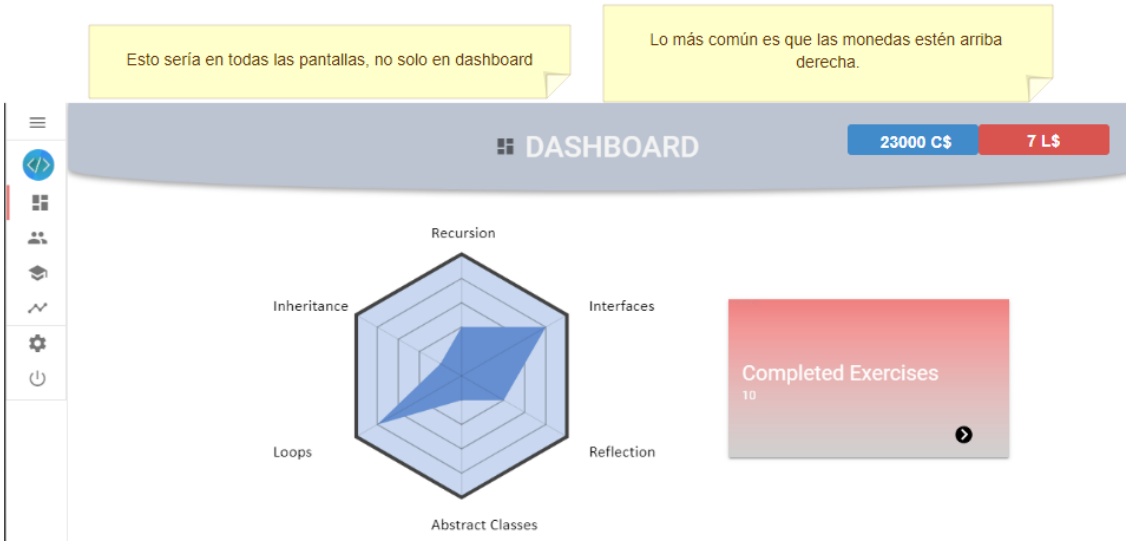


Figura 76. Mockup de una pantalla de ASys con la cantidad de Monedas y Láureas arriba a la derecha.

Tras discutirlo con el usuario y teniendo en cuenta que muchas aplicaciones (las que se trataron en el Estado del arte, entre ellas) las sitúan arriba a la derecha, esta también ha sido nuestra opción elegida.

6.2.2.2 Funcionamiento de los objetos de la tienda

Una vez establecidos los objetos a vender, es necesario precisar, por un lado, su precio en la tienda y, por otro lado, las recompensas que se van a otorgar a los usuarios al completar un ejercicio en la aplicación. La elaboración de un sistema de economía dentro de ASys es algo

fundamental, pues ha de ser equilibrado y, en el caso de que no lo sea o deje de serlo en algún momento,

6.2.3 Objetos y unidades monetarias

Los objetos de la tienda y las unidades monetarias se pueden explicar, en parte en el epígrafe destinado al diseño del servidor y, en otra parte, en el epígrafe asociado al diseño del cliente; sin embargo, estos constituyen el principal pilar del presente TFG y se ha considerado oportuno dedicar un epígrafe separado a su diseño y lógica a implementar.

6.2.3.1 Funcionamiento de los objetos de ASys

Una vez definidos los objetos a vender en la Tabla 10, y extraída a "Requisitos futuros" la implementación de las insignias especiales es preciso determinar cómo debe funcionar la lógica del resto de objetos. Para ello, tal como se mostró recientemente en el apartado Base de datos, se creará una tabla "Item" para los objetos. Esta tabla poseerá una columna llamada "valor". Aquí se almacenará un determinado valor dependiendo de la categoría del objeto que se utilizará ya sea, cuando se compre o cuando se active el objeto. Tanto lo que se introduzca en el campo "valor" o "value" como este comportamiento se reflejan en la Tabla 13. También puede observarse que para algunos objetos se consideró más de una alternativa posible para gestionar la lógica del objeto.

Tabla 13. Funcionamiento de los objetos de ASys

Nombre	**Campo "value"**	**Lógica de activación y/o método para ver cuál es activo o ver su cantidad.**		**Comentarios**
		Opción elegida	Opción 2	
OBJETOS ACTIVABLES				
Foto de perfil	Enlace a la imagen del servidor de MinIO	El enlace se pone en un atributo de la tabla "User" llamado "photo", el cual ya existía, pero no se utilizaba. Cuando se tenga que mostrar la foto de perfil, se accede a ese atributo.	-	-
Color para el nombre de perfil	Una cadena de caracteres con el valor del color en hexadecimal	La tabla "User" tendrá un atributo llamado "usernameColor". Cuando se tenga que mostrar el color activado, se accede a ese atributo.	-	-

OBJETOS CONSUMIBLES				
Duplicador de Monedas temporal	Duración en horas del duplicador	Crear una entrada en la tabla "Item_Used_User" cuando se utilice. Cuando se quiera comprobar que se ha utilizado habrá que consultar la columna asociada a la fecha.	-	Inicialmente, se venderán duplicadores de tres y cinco horas.
Triplicador de Monedas temporal	Duración en horas del triplicador	Crear una entrada en la tabla "Item_Used_User" cuando se utilice. Cuando se quiera comprobar que se ha utilizado habrá que consultar la columna asociada a la fecha-	-	Inicialmente, se venderán triplicadores de cinco horas.
Pista	Cantidad de pistas que otorga su compra	Consultar la cantidad que posee el usuario consultando la tabla "Item_User".	Un atributo en la tabla "User" que indique cuántas posee.	Este objeto puede venderse en grupos o *packs*. Inicialmente, se venderán en grupos de uno, tres y cinco pistas.
OBJETOS QUE DESBLOQUEAN CONTENIDO				
Desbloquea poder pedir pistas en ejercicios	Nada	Un atributo en la tabla "User" que indique si lo tiene o no	Comprobar si el usuario posee el objeto consultando la tabla "Item_User".	-
OBJETOS DE MEJORAS				
Aumentar la capacidad máxima de inventario	Cantidad que incrementa	Consultar la cantidad que posee el usuario consultando la tabla "Item_User". En el lado cliente, mantener actualizada una variable en el store que indique esta capacidad máxima	Un atributo en la tabla "User" que indique su capacidad máxima de inventario	Este objeto puede venderse en grupos o "packs". Inicialmente, se venderán en grupos de uno, tres y cinco incrementos. Se aplica el mismo método para determinar el número de espacios libres de inventario

Para gestionar la lógica de los objetos Pista, Desbloquea poder pedir pistas en ejercicios y Aumentar capacidad de inventario, fue necesario tomar la decisión de añadir campos redundantes o consultar la información existente en el modelo de datos. Cada alternativa tiene sus pros y contras para cada objeto.

6.2.3.1.1 Desbloquea poder pedir pistas en ejercicios

Para poder comprar pistas en la tienda y utilizarlas, es necesario comprar este objeto. Una vez comprado, en la tienda deberán aparecer las **pistas** en la zona de objetos permanentes. También, cuando termine el día y se recalculen los objetos diarios, las **pistas** deberían ser una opción de compra si el objeto **Desbloquea poder pedir pistas en ejercicios** se ha comprado.

Si se añade un campo booleano en la tabla "User", las operaciones se realizarían simplemente teniendo en cuenta ese booleano. En caso de compra de **Desbloquea poder pedir pistas en ejercicios**, se detectaría el cambio a *true* de ese booleano y automáticamente se pedirían los objetos **Pista** al servidor. Por otro lado, hay que tener en cuenta que, al comprar, hay reconocer el objeto para poder poner a *true* el nuevo atributo

En caso de no añadir este objeto, habría que estar "escuchando" la lista de objetos del usuario, de manera que cuando se compre el objeto **Desbloquea poder pedir pistas en ejercicios** se pidan las **pistas.** Por otro lado, cada vez que se pidan los objetos diarios y objetos permanentes a la tienda, habría que consultar la lista de objetos del usuario en busca de este objeto. Por otro lado, como ventaja, no habría que reconocer el objeto al comprarlo.

La **decisión** tomada en este caso es que es mejor utilizar un **nuevo atributo** en la tabla "User" llamado "isCluesAllowed", puesto que existen más pros que contras para esta alternativa. Respecto a este nombre, es cierto que sería mejor llamarlo "areCluesAllowed", pero Spring detecta como *getter* de un atributo booleano aquel que comienza con el prefijo -is.

6.2.3.1.2 Pista

La cantidad de **pistas** se tiene que visualizar cuando se accede al inventario y cuando se realice un ejercicio.

Si se añade un atributo, cada vez que se **compren** pistas, se debe incrementar también ese atributo. Cuando se **usen** pistas, ese atributo debe decrementar su valor. Esto se hace también con la tupla situada en "Item_User". Es decir, con este campo adicional, se duplican las operaciones a realizar (sin tener en cuenta la lógica necesaria para reconocer el item y actuar en consecuencia), pero te ahorras la consulta a la lista de "Item_User" para buscar las **pistas.**

Tal como se adelantó en la Tabla 13, la **decisión** tomada consiste en que se realizará la consulta oportuna a la tabla "Item_User". puesto que esta consulta no hay que realizarla muchas veces.

6.2.3.1.3 Aumentar la capacidad máxima de inventario

Para este caso hay que tener en cuenta que no solo hay que conocer la **capacidad máxima** de objetos que se pueden tener, sino que también hay que conocer el **número de objetos que ocupan espacio** de inventario que posee el usuario. Al igual que las pistas, existe tanto la alternativa de utilizar **nuevos campos** en la tabla "User" (uno para la capacidad máxima y otro que indique el número de objetos que posee), como la alternativa de **consultar la tabla "Item_User"** para obtener los datos requeridos. Sin embargo, ambas alternativas poseen ciertas desventajas adicionales. Esto se debe a que estos datos serán accedidos y escritos más veces; en concreto, cada vez que se acceda a la tienda, se compre un objeto o se acceda al inventario, será necesario acceder a estos datos y cuando se compre o se gaste un objeto (de categoría consumible) será necesario modificarlos.

Por lo tanto, si se utilizasen dos atributos en la tabla "User" sería necesario actualizar esos atributos más a menudo, lo que implicaría, no solo que habría datos redundantes, sino que también habría que realizar más operaciones duplicadas, pues también hay que actualizar la tupla de la tabla "Item_User". Por otro lado, realizar constantemente consultas a la base de datos tampoco resulta viable. Es por ello por lo que se ha decidido optar por una **alternativa mixta**. Por el lado del servidor, se realizarían estas consultas, pues solo se harían cuando se compra un objeto que ocupa espacio de inventario. Por el lado del cliente, en cambio, se propone utilizar un par de variables situadas en el *store* de Vuex: una para la capacidad máxima y otra para el número de objetos que ocupan espacio en el inventario. Estas variables se calcularán la primera vez que se necesiten y el resto de las veces simplemente se consultarán. Y cuando se compre o gaste un objeto se actualizarán. Esta **solución** ofrecerá eficiencia y evitará redundancia en el modelo de datos.

6.2.3.2 Sistema de economía de ASys

Para elaborar el sistema de economía de ASys que vamos a proponer a continuación, hemos utilizado como punto de partida las diferentes aplicaciones tratadas en el epígrafe Estado del arte Estado del arte. Para llevarlo a cabo, se ha tenido en cuenta lo siguiente:

- La dificultad de los ejercicios en el sistema actual oscila entre 1 y 5, siendo 1 la dificultad más sencilla y 5 la más compleja.

- Los usuarios poseen un nivel que se encuentra entre el 1 y el 100, siendo el nivel 1 el más bajo y 100 el más alto.
- La base de la que se partirá y de la cual se realizarán todos los ajustes es que un ejercicio de dificultad 1 otorgará 100 Monedas.

6.2.3.2.1 Recompensa

En este apartado se explica el funcionamiento del mecanismo de recompensas a la hora de realizar ejercicios en ASys. En primer lugar, se detallará el sistema de recompensas de Monedas. En este apartado, para simplificar la explicación, se supondrá que el usuario ha sacado la máxima nota (10 sobre 10) en los ejercicios. En segundo lugar, se hablará sobre cuántas Láureas se entregarán al usuario al completar estos ejercicios. Finalmente, se dedicará otro subapartado para definir la proporción de recompensa otorgada en base a la nota obtenida en cada ejercicio.

6.2.3.2.1.1 Monedas

Respecto a la recompensa de Monedas de los ejercicios existen dos variantes, recompensa fija o variable. A continuación, se analizarán ambas alternativas, siendo la recompensa variable la alternativa por la que nos hemos decantado.

Si fuese una recompensa **fija** quedaría una fórmula de este estilo:

$$Monedas = \text{Dificultad\_Ejercicio} * \text{X}$$

Donde X es la recompensa que otorgaría un ejercicio cuyo nivel de dificultad sea 1. De manera que, si un ejercicio de esta dificultad otorga las recompensas quedarían tal como se muestra en la Tabla 14:

Tabla 14. Recompensas fijas de Monedas según la dificultad del ejercicio.

Dificultad del ejercicio	Recompensa en Monedas
1	100
2	200
3	300
4	400
5	500

El problema de esta alternativa es que un usuario más avanzado, que ya se encuentre en niveles superiores (por ejemplo, nivel 24, nivel 48 o nivel 76) le resultará muy sencillo y más rápido resolver ejercicios de niveles de dificultad bajos, de manera que podría aprovecharse de esto y resolver muchos ejercicios de este tipo para ganar dinero, cuando podría estar haciendo

ejercicios con los que podría estar aprendiendo. Una posible solución a este problema, manteniendo recompensas fijas, sería ir incrementando la diferencia que existe entre cada recompensa. Esto se puede ver a continuación en la

Tabla 15. Recompensas fijas de Monedas según la dificultad del ejercicio. Puede apreciarse como se cuatriplican cada nivel de dificultad adicional.

Dificultad del ejercicio	Recompensa en Monedas
1	100
2	400
3	1600
4	6400
5	25600

Ahora, la diferencia entre la recompensa de un ejercicio de una determinada dificultad y la de otra superior (o inferior) es de cuatro veces más (o menos). Esta alternativa, posee la desventaja de que la diferencia entre resolver un ejercicio de dificultad 1 y dificultad 5 es muy elevada, pues complicará bastante el sistema de precios de los objetos que veremos más adelante.

Por otro lado, existe la opción de hacer un sistema de recompensas variable, donde influya tanto la dificultad del ejercicio como el nivel del usuario:

$$Monedas = (Dificultad\_Ejercicio * X) * (\frac{Dificultad\ Ejercicio}{Y})$$

Donde X es la recompensa que otorgaría un ejercicio cuyo nivel de dificultad sea 1, mientras que Y es un valor entre 1 y 5 según el nivel del usuario (ver Tabla 16).

Tabla 16. Valor de Y según el nivel del usuario.

Nivel de usuario	Y
1 al 20	1
21 al 40	2
41 al 60	3
61 al 80	4
81 al 100	5

Asimismo, podemos asociar el valor de cada Y a la dificultad del ejercicio correspondiente. Por ejemplo, a un usuario de nivel 24 le corresponde un valor 2 de Y, de

manera que este usuario, si realiza ejercicios de dificultad 2, estaría realizando ejercicios acordes a su nivel. De forma que nos quedaría la Tabla 17:

Tabla 17. Recompensas de Monedas variable que depende del nivel del usuario y la dificultad del ejercicio.

Dificultad del ejercicio	Y = 1	Y = 2	Y = 3	Y = 4	Y = 5
1	100	50	33	25	20
2	400	200	133	100	80
3	900	450	300	225	180
4	1600	800	533	400	320
5	2500	1250	833	625	500

De esta manera, un usuario de nivel bajo obtendrá más Monedas si completa ejercicios de dificultad elevada, mientras que un usuario más avanzado, con un nivel superior, si realiza ejercicios de dificultad baja, conseguirá menos Monedas. De esta forma, se reducirá la tentación del usuario de realizar ejercicios de dificultad inferior con la finalidad de ganar más Monedas. En amarillo se marca la recompensa que obtendrá el usuario si realiza ejercicios con una dificultad acorde a su nivel. Sin embargo, este sistema tiene una pequeña desventaja: un usuario, a medida que va ascendiendo de nivel en ASys verá como las recompensas de los ejercicios se van reduciendo poco a poco, pero hay que tener en cuenta los siguientes puntos:

1. ASys te intenta recomendar siempre ejercicios acordes a tu nivel y habilidad.
2. Lo normal es que un usuario realice ejercicios de la dificultad recomendada, por lo que ganará habitualmente una cantidad de Monedas marcada en amarillo en la tabla anterior. Por ejemplo, un usuario de nivel 45 (Y = 3), ganará de forma habitual 300 Monedas. Del mismo modo, un usuario cuyo valor de Y ya sea 5, generalmente realizará ejercicios de dificultad 5, por lo que ganará 500 Monedas por ejercicio. Es decir, cuánto más suba de nivel de un usuario, a pesar de que vea reducir las recompensas, en realidad estará ganando mucho más que antes.

En consecuencia, este último método es la alternativa escogida.

6.2.3.2.1.2 Láureas

Las Láureas, tal como se ha explicado en apartados anteriores, es una recompensa más rara y preciada que las Monedas, permitiendo comprar objetos más valiosos. Esta unidad monetaria solo se podrá conseguir si se obtiene la máxima nota (10) en un ejercicio. A

diferencia de las Monedas, el completar con una nota de 10 un ejercicio de una dificultad inferior a la que le correspondería, no recibirá Láureas.

En cuanto a la cantidad de Láureas a otorgar, este número será fijo y dependerá de la dificultad del ejercicio. En la Tabla 18 podemos ver cuantas Láureas se entregarán en cada caso.:

Tabla 18. Recompensa fija de Láureas al completar un ejercicio con la máxima nota.

Dificultad del ejercicio	Y = 1	Y = 2	Y = 3	Y = 4	Y = 5
1	1	0	0	0	0
2	2	2	0	0	0
3	3	3	3	0	0
4	4	4	4	4	0
5	5	5	5	5	5

6.2.3.2.1.3 Proporción de recompensas en base a la nota

En un principio, se había pensado recompensar al usuario según la siguiente fórmula:

$$Recompensa\ final = Recompensa\ del\ ejercicio * (\text{Max}(0, \frac{Nota\ obtenida - Última\ mejor\ nota}{10}))$$

De esta manera, si un usuario repite el ejercicio, solo ganará Monedas si mejora su mejor nota en el mismo. Esta idea también se aplica a la obtención de Láureas (solo consigues Láureas una vez por ejercicio). Además, la recompensa final del ejercicio crece de forma constante a medida que la nota se acerca a 10. Sin embargo, este método no terminaría de motivar a los usuarios a alcanzar la máxima nota si estos no lo consiguen a la primera, pues al obtener una calificación de 9 ya estarían obteniendo el 90% de las Monedas y, a lo mejor, les convendría realizar un ejercicio nuevo para obtener más Monedas de forma rápida (a menos que quieran conseguir Láureas; en tal caso, sí que estarían interesados en obtener un 10). Es por ello por lo que se propone un sistema de recompensas cuyo porcentaje de Monedas entregadas escala exponencialmente con la nota obtenida, siguiendo una curva X^2, siendo X la nota obtenida. Esto puede verse mejor en la Tabla 19:

Tabla 19. Porcentaje de la recompensa en Monedas que se entregaría al usuario según la nota que consiga en el ejercicio.

Nota obtenida	Recompensa (%)
0	0
1	1
2	4
3	9
4	16
5	25
6	36
7	49
8	64
9	81
10	100

Por lo tanto, la fórmula resultante será:

$$Recompensa\ final = Recompensa\ del\ ejercicio * Max\ (0, \frac{Nota\ obtenida^2 - Última\ mejor\ nota^2}{100})$$

Por otro lado, con el objetivo de premiar aquellos usuarios que obtengan la nota máxima en su primer intento, se otorgará, en estos casos, una bonificación de un 10% de Monedas y 1 Láurea (siempre y cuando pueda conseguirlas).

6.2.3.2.2 Precios

Una vez detallados los métodos para obtener las Monedas y Láureas, es preciso determinar el precio que tendrán los diferentes objetos de la Tienda de ASys, teniendo en cuenta las decisiones recién tomadas.

6.2.3.2.2.1 Fotos de perfil

Con el fin de encontrar un sistema equilibrado, la idea es que existan cinco categorías de fotos de perfil definidas en base a su precio, de manera que habrá fotos de perfil asequibles fácilmente para cualquier usuario, pues estarán centradas en los usuarios de menor nivel, pero además habrá fotos de perfil más caras, que no serán un problema para aquellos usuarios más avanzados, que realicen ejercicios con mayor dificultad y mayores recompensas. Esta idea se asemeja a la tienda que posee Brawl Stars (analizado en el estado del arte), donde existe una gran variedad de aspectos (o *skins*) a diferentes precios, siendo las más caras las más interesantes. En nuestro caso, se ha considerado que un usuario que realice con éxito cinco ejercicios acordes a su nivel ganará suficientes Monedas para comprarse una foto de perfil. En la Tabla 20 se muestra el precio que tendrán las diferentes fotos de perfil. En las dos primeras

columnas se muestra la recompensa que obtendría un usuario al realizar ejercicios de dificultad acorde a su nivel, definidas por su valor Y (nivel del usuario escalado al rango 1-5). Por último, es importante puntualizar que un usuario podrá visualizar y comprar cualquier foto de perfil, independientemente del nivel que posea.

Tabla 20. Precios en Monedas de las fotos de perfil.

Y	Recompensa de un ejercicio	Precio foto de perfil (Monedas)
1	100	500
2	200	1000
3	300	1500
4	400	2000
5	500	2500

6.2.3.2.2.2 Colores para el nombre de perfil

Los colores para el nombre de perfil siguen la misma lógica que las fotos de perfil. A continuación, en la Tabla 21 se muestra su tabla de precios:

Tabla 21. Precios en Monedas de los colores para el nombre de perfil.

Y	Recompensa de un ejercicio	Precio colores para el nombre de perfil (Monedas)
1	100	500
2	200	1000
3	300	1500
4	400	2000
5	500	2500

En este momento, es preciso comentar que si se hubiese elegido la segunda alternativa de sistema de recompensa fijo, ahora estaríamos obligados a definir precios variables de fotos de perfil y colores de nombres de perfil, complicando mucho la lógica de la aplicación. Eligiendo el sistema de recompensas variable definido anteriormente, nos evitamos estos problemas.

6.2.3.2.2.3 Duplicadores de Monedas

Para establecer un precio a los duplicadores se ha escogido como base el duplicador de Monedas de 3 horas de duración. Este duplicador, para que resulte rentable, ha de costar una cantidad de Monedas que un usuario pueda recuperar en una hora de trabajo obteniendo buenas

notas. De manera que sea un aliciente para ponerse a trabajar, pero sin la necesidad de "correr" para poder rentabilizarlos y permitiendo pausas y reflexiones de por medio. Para ello, se ha supuesto que la duración media de los ejercicios es de 20-30 minutos (esta media se podrá ajustar en el futuro). Tomando 30 minutos como referencia, en una hora un usuario habrá realizado dos ejercicios. Estos ejercicios serán de una dificultad, generalmente, acorde al valor Y del usuario, por lo que cada vez esta recompensa será mayor. Esto último implica que el coste de los duplicadores será variable. Además, la cantidad de monedas obtenidas será el doble. Un duplicador de 3 horas costará:

$$Coste\ duplicador\ de\ monedas\ de\ 3\ horas = 2 * (2 * 100 * Y)$$

En esta fórmula destacamos que el 2 dentro del paréntesis es la cantidad de ejercicios que se realizan en una hora y 100 es la recompensa que obtendría un usuario con valor de Y igual a 1. Mientras que el 2 fuera del paréntesis se debe a que un duplicador duplica las monedas conseguidas.

Puesto que existen también duplicadores de 5 horas, si el precio fuese proporcional, este costaría 1,66 veces más, pero para que salga rentable, se hará un descuento llegando a costar solo 1,5 veces más. De esta manera, en la Tabla 22 podemos ver los precios resultantes:

Tabla 22. Precios en Monedas de los duplicadores de Monedas.

Y	Recompensa de un ejercicio	Precio duplicador de monedas de 3 horas (Monedas)	Precio duplicador de monedas de 5 horas (Monedas)
1	100	400	600
2	200	800	1200
3	300	1200	1800
4	400	1600	2400
5	500	2000	3000

6.2.3.2.2.4 Triplicadores de Monedas

De forma análoga a los duplicadores de Monedas, los triplicadores también poseerán un precio variable, pero con ciertas diferencias. En primer lugar, costarán Láureas en lugar de Monedas. En segundo lugar, se considerará que si un triplicador de Monedas dura 5 horas, será necesario realizar una cantidad de ejercicios, con la máxima nota, equivalentes al doble de duración: 10 horas en este caso. Por lo tanto, si cada ejercicio de media se realiza en 30 minutos, realizar 20 ejercicios con la máxima nota otorgarán la cantidad suficiente de Láureas para comprar un triplicador de 5 horas. La fórmula es la siguiente:

$$Coste\ triplicador\ de\ monedas\ de\ 5\ horas = 2\ * duracion(horas) * \frac{2\ ejercicios}{hora} * 1 * Y)$$

Hay que destacar que el valor 1 de la fórmula es la cantidad de Láureas que consigue un usuario con valor 1 de Y al sacar un 10 en un ejercicio de dificultad 1. Finalmente, la Tabla 23 muestra como han quedado los precios:

Tabla 23. Precios en Láureas de los triplicadores de Monedas.

Y	Recompensa de un ejercicio (Láureas)	Precio triplicador de monedas de 5 horas (Láureas)
1	1	20
2	2	40
3	3	60
4	4	80
5	5	100

6.2.3.2.2.5 Pistas

La idea de las pistas es que cuesten un tercio del valor que otorgaría una recompensa de un ejercicio de dificultad acorde al valor de Y del usuario. Además, estas pistas se pueden vender en grupos de 3 y de 5, por lo que recibirán un descuento. El grupo de 3 pistas tendrá un descuento de un 10% y el grupo de 5 pistas un descuento del 15%. El resultado de los precios podemos verlo en la Tabla 24:

Tabla 24. Precios en Monedas de las pistas.

Y	Recompensa de un ejercicio	Precio de 1 Pista (Monedas)	Precio de un grupo de 3 Pistas (Monedas)	Precio de un grupo de 5 Pistas (Monedas)
1	100	33	90	142
2	200	67	180	283
3	300	100	270	425
4	400	133	360	567
5	500	167	450	708

6.2.3.2.2.6 Desbloqueo poder pedir pistas

Al comienzo, el usuario no podrá comprar pistas, de manera que el precio de este objeto no puede ser excesivamente elevado porque si no el usuario podría bloquearse, produciéndose algo parecido a un *deadlock* [76]. Sin embargo, hay que tener en cuenta que ASys te recomienda ejercicios en base a tu nivel, por lo que, por lo general, un usuario no estará pidiendo pistas

continuamente. Aun así, se le otorgarán tres pistas gratis al registrarse en la aplicación. Puesto que será un objeto al que el usuario aspirará relativamente pronto, no es necesario crear ningún precio variable, sino que consideraremos que el usuario se encuentra con un valor 1 de Y. Por consiguiente, la mayoría de las recompensas que obtendrá serán de 100 Monedas, pues serán los ejercicios que hará en su mayoría. Teniendo todo esto en cuenta, se ha decidido que costará 1000 Monedas, lo equivalente a realizar 10 ejercicios de la dificultad más sencilla.

6.2.3.2.2.7 Incremento capacidad máxima de inventario

Con el fin de aportar progreso a ASys, la capacidad máxima de inventario se podrá incrementar comprando este objeto, cuyo precio incrementará cada vez que se compre una unidad de este. La fórmula elegida es:

$$Coste = Precio\ Base + Precio\ Base * 0{,}3 * Nº\ de\ incrementos\ de\ invetario\ comprados$$

Esta fórmula ofrece un crecimiento lineal del precio, de forma que, a medida que el usuario utiliza la aplicación y compra más espacios de inventario, el precio aumentará ligeramente. Para esta fórmula, se ha elegido un precio base de 200 Monedas y un factor multiplicativo de 0,3 con el fin de hallar un equilibrio para que los usuarios de nivel bajo puedan comprar fácilmente estos incrementos al comienzo y, por otro lado, para que los usuarios que ganan más Monedas de recompensa no les resulte ni caro ni barato. Además, al igual que las pistas, este objeto se venderá en grupos de 3 y de 5, con descuentos del 10% y 15% respectivamente. A continuación, en la Tabla 25 se muestran los precios resultantes:

Tabla 25. Precios en Monedas de los incrementos de la capacidad máxima de inventario.

Nº de aumentos comprados	Precio de 1 incremento de inventario (Monedas)	Precio de 3 incrementos de inventario (Monedas)	Precio de 5 incrementos de inventario (Monedas)
0	200	702	1360
1	260	864	1615
2	320	1026	1870
3	380	1188	2125
4	440	1350	2380
5	500	1512	2635
6	560	1674	2890
7	620	1836	3145
8	680	1998	3400
9	740	2160	3655
10	800	2322	3910
11	860	2484	4165
12	920	2646	4420
13	980	2808	4675
14	1040	2970	4930
15	1100	3132	5185
16	1160	3294	5440
17	1220	3456	5695
18	1280	3618	5950
19	1340	3780	6205
20	1400	3942	6460
21	1460	4104	6715
22	1520	4266	6970
…	…	…	…
50	3200	8802	14110

6.2.4 Diseño gráfico

Una vez definidos los objetos que se van a vender, su precio y las diferentes unidades monetarias que se van a utilizar es el turno de diseñar los iconos o logos de cada uno de estos objetos (recordemos que en el epígrafe Interfaces externas se ha acordado que todos los objetos tendrán un logo asociado). En primer lugar, trataremos los iconos de las Monedas y las Láureas y, posteriormente, los logos elegidos para cada uno de los objetos que se venderán en la tienda de ASys.

Es importante mencionar que todos los iconos y logos que veremos a continuación, si no se argumenta lo contrario, se han extraído de la página web https://pixabay.com/es/ y no poseen derechos de autor. Sin embargo, muchos de estas imágenes será necesario modificarlas ligeramente en Photoshop o Illustrator para que quede una imagen cuadrada y no rectangular.

Esto se hace con el objetivo de evitar posibles deformidades a la hora de insertarlas en la aplicación.

6.2.4.1 Unidades monetarias

6.2.4.1.1 Monedas

Para las Monedas, se propusieron al usuario tres tipos de iconos. Los iconos propuestos no pretenden ser una versión final de cada alternativa, pero si una guía que facilitase su elección. Los tipos de iconos propuestos son: una bolsa de monedas (ver Figura 77), un montón de monedas (ver Figura 78) o una moneda única (ver Figura 79).

Figura 77. Propuesta icono para las Monedas o Coins. Saco de monedas.

Figura 78. Propuesta icono para las Monedas o Coins. Montón de monedas

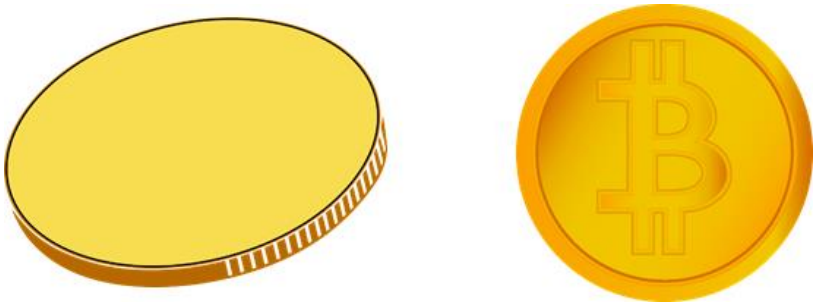


Figura 79. Propuesta icono para las Monedas o Coins. Moneda única.

Por otro lado, también había que decidir el logo que llevaría esta unidad monetaria. Las ideas propuestas al usuario fueron: **i)** un icono de una "**C**" parecida a que se encuentra en la Figura 77, o bien, **ii)** Un icono de un ordenador, puesto que la temática de ASys en la programación.

Finalmente, se decidió con el usuario que el icono utilizado será un montón de monedas cuyo logo sería la primera opción propuesta. El montón elegido ha sido el de la Figura 80 y, gracias a la ayuda de Adobe Illustrator se pudo crear el icono final de las Monedas, que podemos ver en la Figura 81.

Figura 80. Icono de un montón de monedas elegido para ser el icono de las Monedas o Coins.

Figura 81. Icono final de las Monedas o Coins.

6.2.4.1.2 Láureas

Respecto a las Láureas, se propuso un icono de una corona triunfal como la que podemos ver en la Figura 82. Sin embargo, a pesar de que el verde es el color de estas coronas, se quería hacer más énfasis en el valor de esta unidad monetaria. Es por ello por lo que se modificó con Photoshop la Figura 82 y se pintó de color violeta tal como podemos ver en la Figura 83. Esto se debe a que este color siempre se ha asociado a la realeza y al poder [77].

Figura 82. Primera propuesta de icono para las Láureas.

Figura 83. Icono final de las Láureas.

6.2.4.2 Objetos

6.2.4.2.1 Fotos de perfil

Inicialmente, estarán disponibles para la compra 25 fotos de perfil, con el fin de dar cierta variedad en la aplicación. Puesto que según la Tabla 20 existirán cinco precios diferentes para las fotos de perfil, se ha decidido dividir las imágenes elaboradas en cinco grupos de cinco:

- Cinco fotos de perfil que valdrán **500** Monedas (ver Figura 84).
- Cinco fotos de perfil que valdrán **1000** Monedas (ver Figura 85).
- Cinco fotos de perfil que valdrán **1500** Monedas (ver Figura 86).
- Cinco fotos de perfil que valdrán **2000** Monedas (ver Figura 87).

- Cinco fotos de perfil que valdrán **2500** Monedas (ver Figura 88).

Figura 84. Fotos de perfil que costarán 500 Monedas.

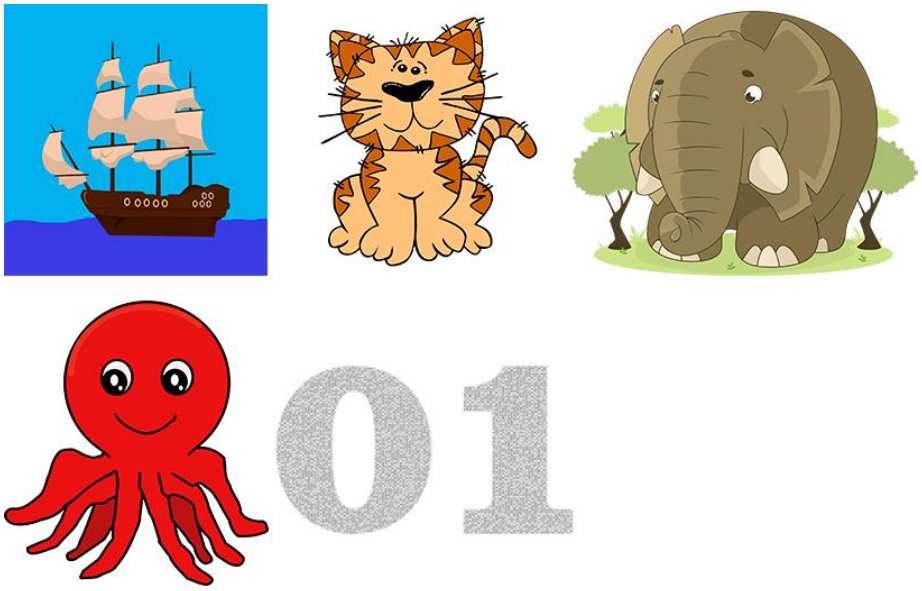


Figura 85. Fotos de perfil que costarán 1000 Monedas.

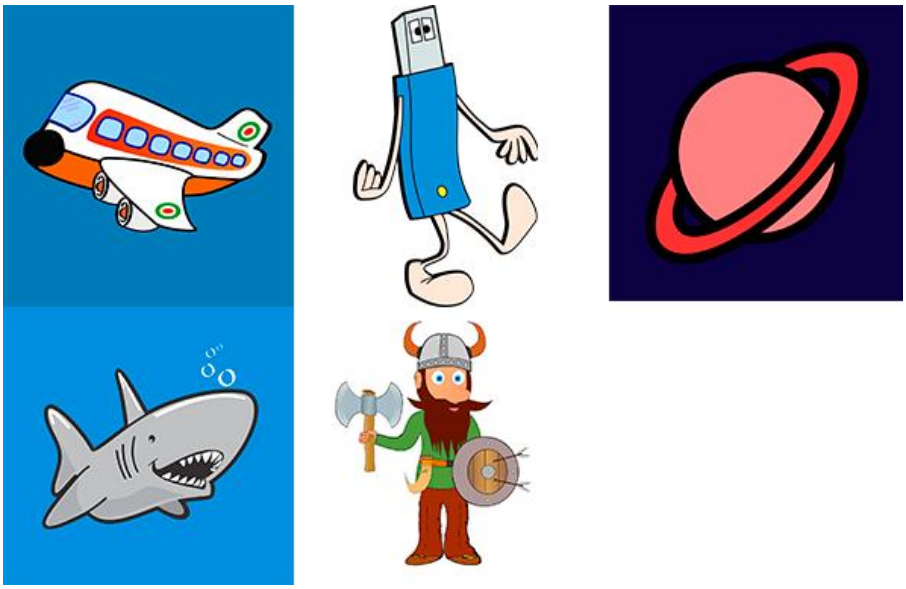


Figura 86. Fotos de perfil que costarán 1500 Monedas.

Figura 87. Fotos de perfil que costarán 2000 Monedas.

Figura 88. Fotos de perfil que costarán 2500 Monedas.

En suma, habrá una foto de perfil por defecto y que poseerán todos los usuarios. Esta la podemos ver en la Figura 89.

Figura 89. Foto de perfil por defecto.

6.2.4.2.2 Colores para el nombre de perfil

Con el mismo propósito de las fotos de perfil de dar variedad, en este proyecto se introducirán siete colores para el nombre de perfil. Para cada uno de estos colores, se ha realizado (elaboración propia) un logo que es el que poseerá el objeto en la aplicación. Acordes con la Tabla 21, estos colores se dividirán en los siguientes grupos:

- Un color (rosa) para el nombre de perfil que costará 500 Monedas (ver Figura 90).

- Dos colores (celeste y verde) para el nombre de perfil que costarán 1000 Monedas (ver Figura 91).
- Dos colores (naranja y rojo) para el nombre de perfil que costarán 1500 Monedas (ver Figura 92).
- Un color (azul) para el nombre de perfil que costará 2000 Monedas (ver Figura 93).
- Un color (violeta) para el nombre de perfil que costará 2500 Monedas (ver Figura 94).

Figura 90. Color para el nombre de perfil que costará 500 Monedas.

Figura 91. Colores para el nombre de perfil que costarán 1000 Monedas.

Figura 92. Colores para el nombre de perfil que costarán 1500 Monedas.

Figura 93. Color para el nombre de perfil que costará 2000 Monedas.

Figura 94. Color para el nombre de perfil que costará 2500 Monedas.

En suma, de forma análoga a las fotos de perfil, habrá un color para el nombre de perfil por defecto y que poseerán todos los usuarios pues, por motivos obvios, es necesario que exista siempre uno. Este es el color negro su logo lo podemos ver en la Figura 95.

Figura 95. Color para el nombre de perfil por defecto.

6.2.4.2.3 Duplicadores y Triplicadores de Monedas

Los logos de los multiplicadores de Monedas se han elaborado mediante Photoshop. El logo de los duplicadores podemos verlo en la Figura 96 y el de los triplicadores en la Figura 97.

Figura 96. Logo de un duplicador de Monedas.

Figura 97. Logo de un triplicador de Monedas.

En suma, puesto que se decidió organizar el inventario en categorías (ver Figura 67), se elaboró un logo que representa los multiplicadores en su conjunto (ver Figura 98).

Figura 98. Logo de los multiplicadores de Monedas

6.2.4.2.4 Pistas

Para las pistas se utilizó un icono de una lupa (ver Figura 99). Aunque, como se venden en grupos de uno, tres o cinco, se modificó la imagen con Photoshop, quedando como resultado los tres logos de la Figura 100 .

Figura 99. Logo base para las pistas.

Figura 100. Logos de los objetos de pista.

6.2.4.2.5 Desbloquea poder pedir pistas en ejercicios

Para este objeto, también se utilizó la Figura 99 como base y se desarrolló un logo como el de la Figura 101. Además, mientras que este objeto no se compre, en la pantalla principal del inventario (ver Figura 67) aparecerá el logo de la Figura 102.

Figura 101. Logo del objeto Desbloquea pistas para ejercicios.

Figura 102. Logo que indica que no se han desbloqueado las pistas.

6.2.4.2.6 Aumento de capacidad máxima de inventario

Al igual que las pistas, para los aumentos de capacidad máxima de inventario se partió de una imagen base (ver Figura 103) y se realizó un logo específico para cada incremento (un incremento, tres y cinco), modificando ligeramente los colores de la imagen, tal como podemos ver en la Figura 104.

Figura 103. Logo base para los aumentos de la capacidad máxima de inventario.

Figura 104. Logos de los objetos de aumento de la capacidad máxima de inventario

7 Desarrollo de la solución

En este apartado procederemos a explicar y detallar la implementación del módulo de monetización, con la tienda y el inventario, así como el sistema de recompensas. En primer lugar, detallaremos cómo se ha procedido con el desarrollo del lado del servidor, donde se preparó la capa de persistencia, la capa de servicios y la capa de los controladores. Estas tres capas se encargarán de dar el soporte necesario a las peticiones entrantes del lado del cliente. Este otro lado será el segundo capítulo de este epígrafe. Aquí se detallarán los diferentes componentes creados y modificados, así como la forma de mostrar los datos y de gestionarlos gracias a la librería de Vuex. Sendos lados (servidor y cliente) seguirán una estructura compuesta por diferentes apartados, cada uno tratando la construcción de un elemento importante del proyecto (objetos diarios, compra de un objeto, utilización de un objeto, etc.). Finalmente, se comentarán algunos problemas que se detectaron durante el desarrollo de esta solución.

7.1 Lado del servidor

7.1.1 Modelo de datos y repositorios

Uno de los primeros pasos llevados a cabo fue la creación de las diferentes clases en la capa de acceso a datos. Concretamente, se creó una clase para cada tabla propuesta en el epígrafe Base de datos:

- Item
- ItemUser
- ItemUsedUser
- Category
- UserHistory
- DailyItems

A cada una de estas clases, para que sean legibles por el *framework* de Hibernate, se le añadieron diversas anotaciones de la librería *javax.persistence* Como ejemplo, podemos ver la clase “ItemUser” en la Figura 105, donde se le añaden las *Java Annotations* @Entity y @Table en la cabecera de la clase para indicar que esa clase será tratada como una entidad y que su tabla se llamará “item_user” (aunque aparezca en mayúsculas, se creará en minúsculas. Sin embargo,

en el texto nos referiremos a las tablas con la primera letra de cada palabra que la compone en mayúsculas). Además, se añade una restricción de unicidad conjunta para los atributos "item_id" y "user_id", de forma que no podrá haber dos tuplas con el mismo usuario y objeto en la tabla. Por otro lado, cada atributo de la clase se etiquetó con la etiqueta @Column para indicar que será una columna en la tabla (la propiedad "name" es opcional si se quiere dejar el mismo nombre que el del atributo de la clase Java). Además, el atributo que será la clave primaria se señala con la anotación @Id acompañada de @GeneratedValue para que este "id" se genere de forma automática siguiendo una determinada estrategia que, si no se indica, será aquella que mejor se adapte al SGBD que se esté utilizando[78]. En suma, también existen anotaciones como @NotNull para indicar que ese valor no puede ser nulo y @ColumnDefault que indica el valor por defecto que tendrá ese atributo determinado si no se le indica durante la creación del objeto. Por último, las anotaciones @ManyToOne y @JsonBackReference tienen que ver con las relaciones con otras clases y se tratarán en un subapartado dedicado.

```java
@Entity
@Table(
        name = "ITEM_USER",
        uniqueConstraints = {@UniqueConstraint(columnNames = {"ITEM_ID", "USER_ID"})}
)
public class ItemUser implements Serializable {

    private static final long serialVersionUID = 5199627630019784932L;

    @Id
    @Column(name = "ID")
    @GeneratedValue
    private Long id;

    //@JsonBackReference
    @JoinColumn(name = "ITEM_ID")
    @ManyToOne(fetch = FetchType.EAGER)
    private Item item;

    @JsonBackReference
    @JoinColumn(name = "USER_ID")
    @ManyToOne(fetch = FetchType.EAGER)
    private User user;

    @Column(name= "QUANTITY")
    @ColumnDefault("1")
    @NotNull
    private Integer quantity;
```

Figura 105. Clase ItemUser.

Al igual que en "ItemUser", se siguió un procedimiento parecido para cada una de las nuevas clases. Una vez creadas estas clases, se procedió con la creación de una interfaz para cada una de ellas que actúa como acceso a los datos de la base de datos. Siguiendo con el

ejemplo de “ItemUser”, en la Figura 106 podemos ver su interfaz de acceso a datos. Aquí destacamos que hereda de *JpaRepository* y posee ciertos métodos que se crearon para cubrir ciertas necesidades que han ido surgiendo durante el desarrollo. Algunas de estas se mencionarán en apartados posteriores. Aun así, podemos aprovechar y comentar que estos métodos pueden ser de dos tipos:

1. Con la anotación **@Query.** Esta alternativa permite realizar consultas de todo tipo, escribiendo la consulta SQL como parámetro.
2. Mediante “**Query Methods”** [79] de Spring Data JPA. Esta solución permite evitar escribir la consulta SQL, pues el propio nombre del método es la consulta. Esta alternativa resulta muy útil cuando la consulta que se quiere hacer es sencilla.

```
@RepositoryRestResource(exported = false)
public interface ItemUserRepository extends JpaRepository<ItemUser, Long>, JpaSpecificationExecutor<ItemUser> {

    @Query("SELECT iu.quantity " +
            "FROM ItemUser iu, Item i " +
            "WHERE iu.user = :user " +
                "AND iu.item = i " +
                "AND i.name LIKE 'database.item.oneInventorySpace.name'")
    Integer inventoryCapacity(@Param("user") User user);

    @Query("SELECT SUM(iu.quantity) " +
            "FROM ItemUser iu, Item i, Category c " +
            "WHERE iu.user = :user " +
            "AND iu.item = i " +
            "AND i.category = c " +
            "AND c.type <> com.asysweb.model.enums.TypeCategory.CONTENT " +
            "AND c.type <> com.asysweb.model.enums.TypeCategory.UPGRADE"
    )
    Integer numberOfItems(@Param("user") User user);

    ItemUser findByUserAndItem(User user, Item item);

}
```

Figura 106. Clase ItemUserRepository. Actúa como interfaz de acceso a los datos de la base de datos.

Otros métodos que proporciona esta clase provienen de la clase heredada *JpaRepository* y están relacionados con las operaciones CRUD.

7.1.1.1 Relaciones entre clases y problemas encontrados

Una vez creadas las clases correspondientes, había que relacionarlas entre ellas y con clases ya existentes como la clase “User”. Para ello, se crearon los atributos correspondientes en cada una de las clases siguiendo el modelo de datos definido en el epígrafe Base de datos. Para que Hibernate reconozca las asociaciones es necesario añadir, según corresponda, la anotación @OneToOne (uno a uno), @OneToMany (uno a muchos), @ManyToOne (muchos a uno) o

@ManyToMany (muchos a muchos). Esta última anotación puede evitarse y se recomienda también puesto que es más natural utilizar una clase intermedia ("ItemUser" es una de ellas) y emplear anotaciones @OneToMany y @ManyToOne para implementar una relación muchos a muchos [80]. En el caso de que no se utilizasen estas anotaciones en alguno de los dos lados de la asociación, se estaría creando una relación unidireccional, lo cual puede no interesar en ciertos casos; según la documentación oficial de Hibernate, las asociaciones unidireccionales @OneToMany son ineficientes [81] y las @OneToOne unidireccionales carecen de sentido lógico [82].

Por los motivos enunciados recientemente, en un comienzo se intentó implementar asociaciones bidireccionales para todos los casos, pero esto ocasionaba una *Infinite recursion (StackOverflowError).* Esto ocurre cuando Spring intenta convertir un objeto JPA que tiene una asociación bidireccional en un JSON mediante la librería *Jackson.* Esta librería es la que se usa en el proyecto para hacer este tipo de conversiones. En otras palabras, al tener una relación bidireccional entre dos objetos A y B, sucede que A tiene a B como atributo, pero a su vez, B tiene a A como atributo, lo que termina convirtiéndose en una recursión infinita. Para solucionar este problema, en [83] se exponen diversas soluciones que podemos agrupar en cuatro grupos:

1. @JsonManagedReference y @JsonBackReference
2. @JsonIdentifyInfo
3. @JsonIgnore y @JsonIgnoreProperty
4. @JsonView o un *serializer/deserializer* personalizado

A continuación, pasaremos a explicar brevemente en qué consiste cada alternativa y comentaremos qué decisión hemos tomado para cada asociación. Pero antes de comenzar este análisis, es importante recalcar que este problema y estas soluciones afectan al lado cliente, pues es quien recibe la información mediante JSON. En el lado del servidor, no afectará en nada.

7.1.1.1.1 @JsonManagedReference y @JsonBackReference

Estas anotaciones se colocan encima del atributo. Esta solución **omite** durante la serialización uno de los dos lados de la relación: aquel que lleve la anotación @JsonBackReference [84]. Si es una relación 1 a 1, es posible elegir el lado que queremos mostrar y cual no nos importa ocultar. Si es una relación 1 a muchos (supongamos la asociación "Category" con "Item"), se omite el lado que tiene el "uno" (@JsonBackReference no se puede colocar sobre una colección). Esto es un problema en ciertos casos, como con "Item" y "Category" puesto que no podríamos determinar la categoría del objeto (o "item") en el lado

cliente, o “Item” con “DailyItems”. En este segundo ejemplo, “DailyItems” es quien posee @JsonBackReference en sus atributos de tipo “Item”, por lo que no se podrían leer los objetos diarios en la parte cliente, que es la que recibe el objeto serializado.

Sin embargo, existen casos donde esta solución nos puede servir, como es el caso mostrado anteriormente en la Figura 105 con el atributo “user” de “ItemUser”, puesto que en el lado del cliente no parece ser una información valiosa. En la Figura 107 se muestra cómo se utiliza @JsonManagedReference desde la clase “User”. De forma análoga, se hizo lo mismo con las relaciones “User- UserHistory”, “User-ItemUsedUser” y “User-DailyItems”.

```
@JsonManagedReference
@OneToMany(
        mappedBy = "user",
        fetch = FetchType.LAZY,
        cascade = {CascadeType.PERSIST,CascadeType.REFRESH,CascadeType.MERGE,CascadeType.DETACH}
)
private List<ItemUser> itemsUser;

@JsonManagedReference
@OneToMany(
        mappedBy = "user",
        fetch = FetchType.LAZY,
        cascade = {CascadeType.PERSIST,CascadeType.REFRESH,CascadeType.MERGE,CascadeType.DETACH}
)
private List<ItemUsedUser> itemsUsedUser;

@JsonManagedReference
@OneToOne(
        mappedBy = "user",
        fetch = FetchType.EAGER,
        cascade = {CascadeType.PERSIST,CascadeType.REFRESH,CascadeType.MERGE,CascadeType.DETACH}
)
private DailyItems dailyItems;

@JsonManagedReference
@OneToMany(
        mappedBy = "user",
        fetch = FetchType.LAZY,
        cascade = {CascadeType.PERSIST,CascadeType.REFRESH,CascadeType.MERGE,CascadeType.DETACH}
)   private List<UserHistory> userHistory;
```

Figura 107. Utilización de @JsonManagedReference en la clase "User".

7.1.1.1.2 @JsonIdentifyInfo

Esta anotación se coloca encima de la cabecera de la clase. Gracias a esto, puedes serializar ambos lados de la relación, pero hay que tener cuidado. Este comportamiento, en relaciones 1 a 1 funcionará bien. Por ejemplo, en la relación “DailyItems” - “User”, un objeto de tipo “User” podrá acceder a su atributo de tipo “DailyItems” y viceversa. Pero si “DailyItems” también tiene relación con “Item” (muchos a uno) y este también posee @JsonIdentifyInfo para

evitar la *Infinite Recursion*, pasarán cosas que no queremos; un usuario podrá acceder a su atributo "dailyItems" pero los "items" de este no se verán correctamente.

En [85] se explica que solo la primera instancia de un objeto se serializa completamente. El resto se referenciarán mediante un *id*. Por ejemplo, una lista de "ItemUser" con esta anotación, al serializarla, quedaría como el pseudocódigo de la Figura 108.

Figura 108. Pseudocódigo JSON resultado de serializar "ItemUser" con @JsonIdentifyInfo.

Con esta alternativa, resulta imposible tratar con clases que se sabe que se listarán, por lo que se decidió no utilizar esta alternativa.

7.1.1.1.3 @JsonIgnore y @JsonIgnoreProperties

Estas dos anotaciones se utilizan para ignorar una propiedad al serializarse y deserializarse con ligeras diferencias. Por un lado, @JsonIgnore se usa a nivel de miembro o método y se puede poner encima del atributo en cuestión, encima del *get* o encima del *set* [86]. El resultado será el mismo. Aunque este se puede combinar con @JsonProperty para poder serializar, pero no deserializar o viceversa [87]. En cambio, @JsonIgnoreProperties se añade encima de la cabecera de la clase y se añaden los atributos que se quieren ignorar en la serialización y deserialización [86]. Puesto que para lo que necesitamos no influyen estas diferencias, se optó por @JsonIgnoreProperties para tener los atributos ignorados de una clase en un mismo sitio. Esta solución se optó para el resto de las asociaciones no mencionadas en el apartado previo (@JsonManagedReference y @JsonBackReference), donde se ignoró uno de los lados de la asociación en base a nuestras necesidades. Un ejemplo de esto puede verse en la Figura 109 donde en la clase "Item" se ignoran en la serialización y deserialización los atributos de tipo "ItemUser" y "ItemUsedUser".

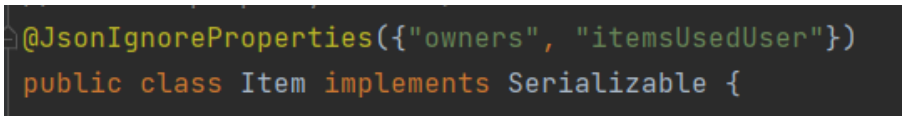


Figura 109. Utilización de @JsonIgno2reProperies en la clase "Item".

Por otro lado, la asociación “Item-DailyItems”, a pesar de poder aplicar esta solución, se convirtió en una relación unidireccional, donde solo “DailyItems” tiene acceso a “Item” y no al revés. Esto se debe a que la eficiencia no se reduce al ser una relación @ManyToOne (muchos a uno) y que resulta un dato “inútil” conocer qué “ítems” son objetos diarios.

7.1.1.1.4 Otras alternativas

Además de las soluciones enunciadas anteriormente, existen un par de alternativas que poseen poca documentación, complicando su implementación, pero estaría bien considerarlas para un futuro:

- @JsonView
- Crear un Serializer y un Deserializar personalizado

7.1.1 Controladores y servicios

Para la implementación de esta solución se creó un nuevo controlador (“ShopRestController”) encargado de gestionar las peticiones relacionadas con la tienda (petición de objetos, compra de estos, etc.). También se añadieron nuevos métodos al controlador “UserRestController” relacionado con la utilización de los objetos.

Por otro lado, para cada clase creada del modelo de datos se creó una clase en la capa de servicios encargada de gestionar la lógica relacionada con estas clases (ver Figura 110).

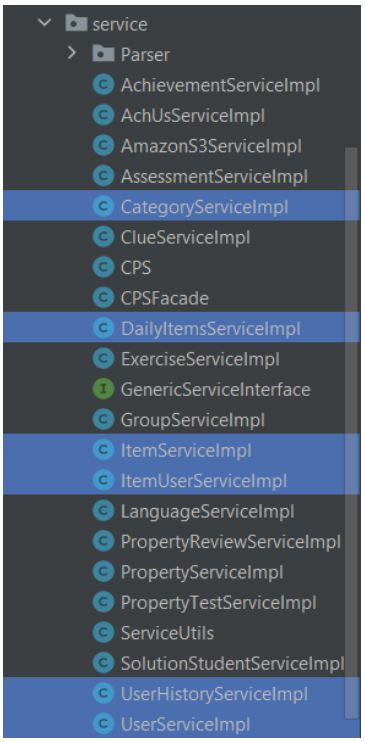


Figura 110. Capa de servicio. En azul, las nuevas clases creadas.

Antes de comenzar a explicar el desarrollo de los servicios, es importante remarcar que siempre se ha intentado reutilizar al máximo el código, evitando el código duplicado. Esto se ha conseguido mediante diversas refactorizaciones con técnicas como *extract method.* Además, se ha cuidado el nombre de las variables y funciones [88], siendo estas últimas pequeñas tal como se recomienda en [89]. Por último, los valores utilizados en fórmulas, los hemos extraído y convertido en constantes, con el fin de tener un código más mantenible (en la Figura 111 podemos ver un ejemplo).

```
public class ItemServiceImpl implements GenericServiceInterface<Item, Long>{

    private static final double EXTRA_PRICE_PERMANTENT_ITEMS = 1.25;
    private static final double EXTRA_PRICE_DUPLICATOR = 1.5;
    private static final double EXTRA_PRICE_INVENTORY_UPGRADE = 0.3;
    private static final double DISCOUNT_PRICE_3_ITEMS = 0.1;
    private static final double DISCOUNT_PRICE_5_ITEMS = 0.15;


    private static final int DEFAULT_NUMBER_OF_RETURN_TOKENS = 3;
    private static final int BASE_PRICE_INVENTORY_UPGRADE = 200;
```

Figura 111. Valores utilizados en fórmula declarados constantes.

7.1.1.1 Objetos diarios

El método principal se puede ver en la Figura 112. Cada vez que se soliciten los objetos diarios, el servidor buscará si el usuario ya tiene objetos diarios y si estos están actualizados al día actual y, en caso contrario, se recalculan. Para ello se llama al método *getNewDailyItems*. Este método elegirá al azar tres nuevos objetos de una lista obtenida consultando la base de datos. Esta lista de objetos contendrá cualquier posible objeto que pueda comprar el usuario, esto es, cualquier objeto de categoría de tipo "Consumable" y "Upgrade", así como aquellos objetos de categoría de tipo "Activable" y "Content" que no pertenezcan al usuario. En suma, se descartarán los objetos de pistas si estas no están desbloqueadas. En la Figura 113 destacamos estas consultas SQL. Finalmente, en cualquiera de los casos en los que se obtengan los objetos diarios, antes de enviarlos al cliente, se ajustarán sus precios de acuerdo con el sistema de economía, concretamente el epígrafe Precios (ver Figura 114).

```
public DailyItems getDailyItems(Long userId) {
    DailyItems dailyItems = this.findByUserId(userId);

    if (dailyItems == null) {
        dailyItems = new DailyItems();
        this.getNewDailyItems(dailyItems, userId);
    } else {
        Calendar date = dailyItems.getDate();

        Calendar today = Calendar.getInstance();
        today.set(Calendar.HOUR_OF_DAY, 0);
        today.set(Calendar.MINUTE, 0);
        today.set(Calendar.SECOND, 0);
        today.set(Calendar.MILLISECOND, 0);

        if(date.before(today)) {
            this.getNewDailyItems(dailyItems, userId);
        } else {
            this.setDailyItemsPrice(dailyItems);
        }
    }
    return dailyItems;
}
```

Figura 112. Método getDailyItems que obtiene los objetos diarios. Se encuentra en DailyItemsServiceImpl.java

```
@Query(
        "SELECT i " +
        "FROM Item i " +
        "WHERE i.id NOT IN " +
                "(SELECT i2.id " +
                "FROM Item i2, ItemUser iu, Category c " +
                "WHERE iu.item = i2 AND iu.user = :userId " +
                "AND i2.category = c " +
                "AND c.type <> com.asysweb.model.enums.TypeCategory.CONSUMABLE " +
                "AND c.type <> com.asysweb.model.enums.TypeCategory.UPGRADE)"
)
List<Item> getItemsCanPurchaseUserNoClues(@Param("userId") User user);


@Query(
        "SELECT i " +
        "FROM Item i " +
        "WHERE i.id NOT IN " +
                "(SELECT i2.id " +
                "FROM Item i2, ItemUser iu, Category c " +
                "WHERE iu.item = i2 AND iu.user = :userId " +
                    "AND i2.category = c " +
                    "AND c.type <> com.asysweb.model.enums.TypeCategory.CONSUMABLE " +
                    "AND c.type <> com.asysweb.model.enums.TypeCategory.UPGRADE) " +
            "AND i.id NOT IN " +
                "(SELECT i3.id " +
                "FROM Item i3, Category c2 " +
                "WHERE i3.category = c2 " +
                    "AND c2.name = 'Clue') "
)
List<Item> getItemsCanPurchaseUser(@Param("userId") User user);
```

Figura 113. Consultas que obtienen los objetos que podrían ser objetos diarios del usuario. Se encuentran en ItemUserRepository.java

```java
protected void setPriceOfItem(Item item, User user) {
    BigDecimal newPrice = new BigDecimal( val: 0);
    int inventoryCapacity = userService.getUserInventoryCapacity(user);
    String itemCategoryName = item.getCategory().getName();

    switch (itemCategoryName) {
        case AVATAR:
        case USERNAME_COLOR:
        case CONTENT_TO_UNLOCK:
            return;

        case DUPLICATOR:
            newPrice = this.setPriceOfDuplicatorItem(item, user);
            break;

        case TRIPLICATOR:
            newPrice = this.setPriceOfTriplicatorItem(item, user);
            break;

        case CLUE:
            newPrice = this.setPriceOfClue(item, user);
            break;

        case INVENTORY_UPGRADE:
            newPrice = this.setPriceOfUpgradeInventoryItem(item, inventoryCapacity);
            break;
    }

    item.setPrice(newPrice);
}
```

Figura 114. Método setPriceOfItem. Es el encargado principal de establecer el precio de un objeto para un usuario.

En el caso de que sea el propio cliente quien solicite la renovación de los objetos diarios, se llamará directamente a *getNewDailyItems*.

7.1.1.2 Objetos permanentes

Para la obtención de estos objetos, simplemente se pedirá a la base de datos todo objeto que no sea de categoría con tipo “Activable” y, en caso de que no estén desbloqueadas las pistas, estas tampoco se pedirán. Una vez obtenidos los objetos, se calculará su precio de la misma forma que con los objetos diarios (reutilizando el método *setPriceOfItem*) y, posteriormente, se le sumará un 25% adicional de coste.

7.1.1.3 Comprar un objeto

Cuando un usuario compra un objeto, antes de proceder con la compra se contrasta cierta información con la base de datos. Si el objeto comprado está en la sección de objetos diarios, se siguen los pasos del diagrama de flujo de la Figura 115, mientras que si es un objeto permanente los pasos a seguir podemos verlos en la Figura 116. Podemos notar que la única diferencia entre sendos diagramas son los elementos de decisión de color naranja.

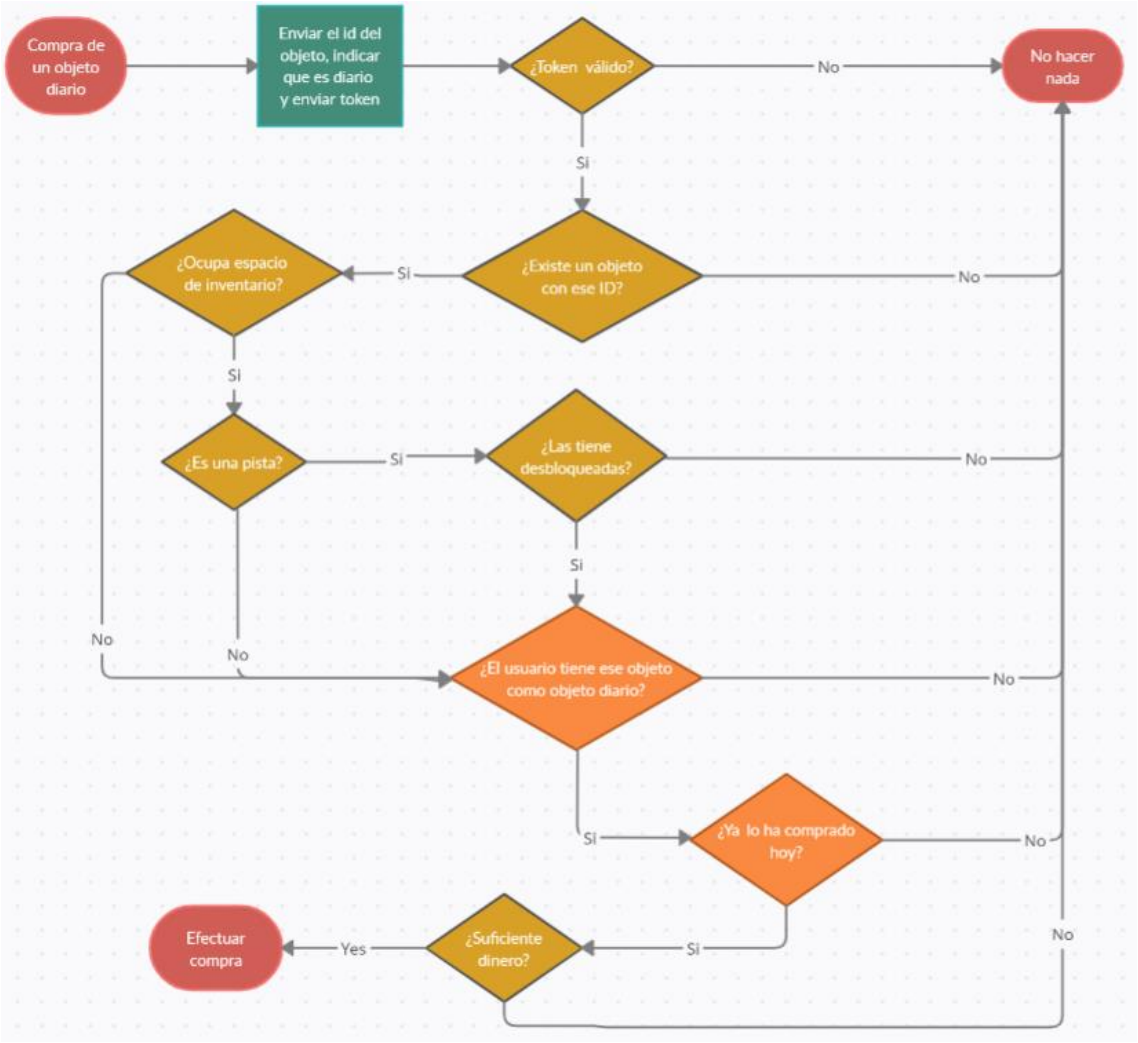


Figura 115. Comprobaciones realizadas antes de proceder con la compra de un objeto diario.

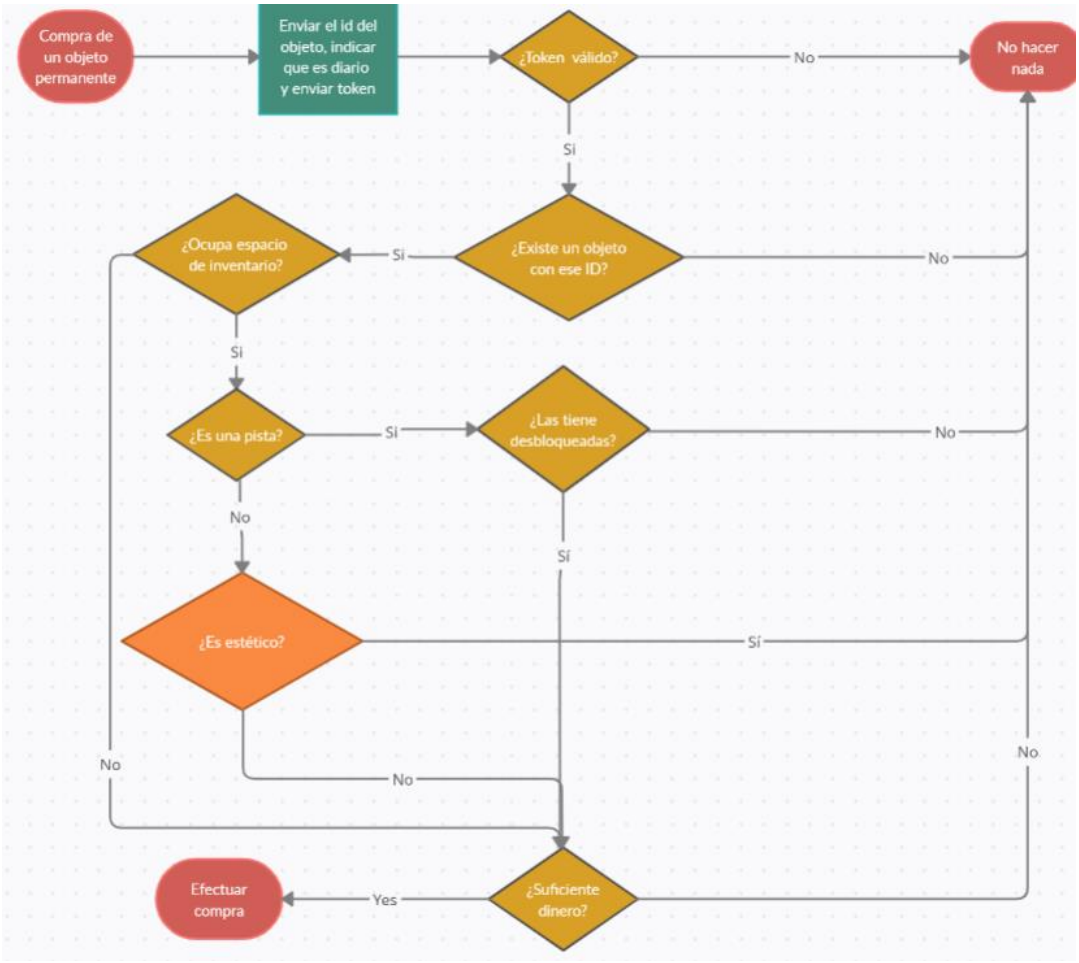


Figura 116. Comprobaciones realizadas antes de proceder con la compra de un objeto permanente.

Una vez realizadas las comprobaciones correspondientes, si todo es correcto, se modificará o creará una entrada en la tabla "item_user" (dependiendo de si el objeto es nuevo o ya poseía uno), se registrará una nueva entrada en "user_history" y se restará el importe del objeto. Este importe se habrá obtenido mediante el método *setPriceOfItem* (ver Figura 114). Por último, si el objeto comprado es "Desbloquea poder pistas en ejercicios" se pondrá a *true* el atributo "isCluesAllowed".

7.1.1.4 Usar un objeto

De forma parecida a la compra de objetos, antes de llamar al código encargado de activar al objeto en cuestión, se contrasta la información recibida con la base de datos. Para ello, se reutiliza en lo posible el código utilizado para las comprobaciones durante la compra y se añaden otros métodos encargados de comprobar las nuevas situaciones planteadas. Las comprobaciones necesarias para activar una foto de perfil están en el diagrama de flujo de la Figura 117, mientras que para activar un color para el nombre de perfil tenemos el diagrama de

la Figura 118 y, en la Figura 119, tenemos el diagrama de flujo con las comprobaciones a realizar si se activa un multiplicador.

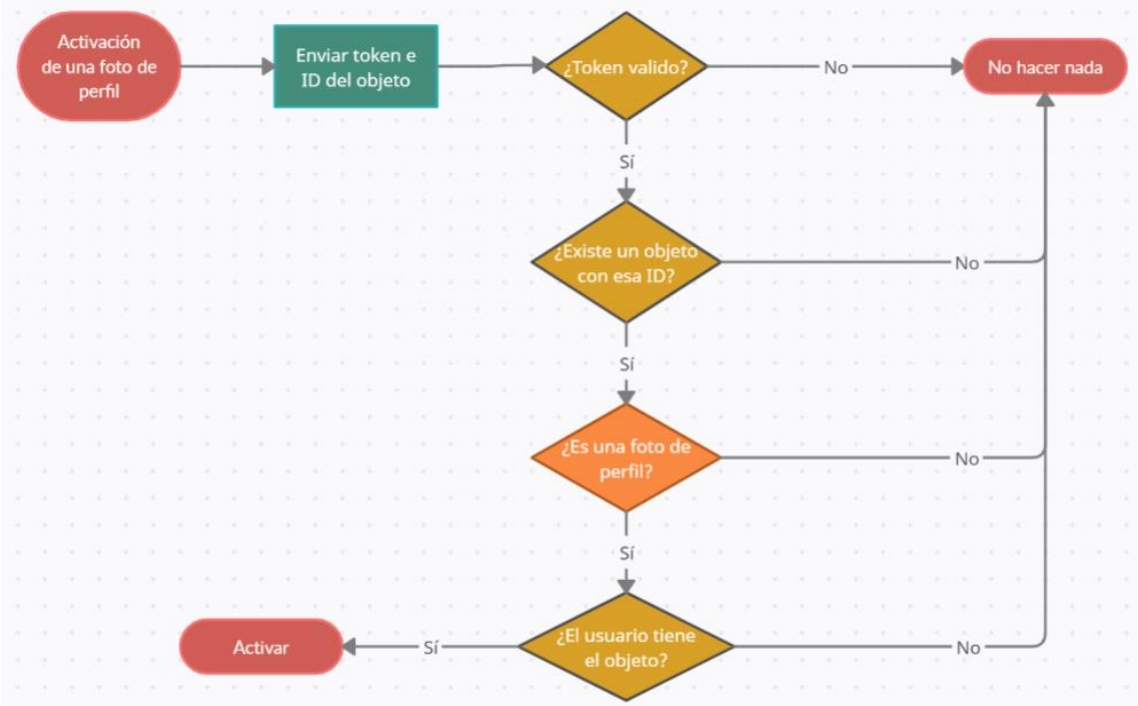


Figura 117. Comprobaciones realizadas antes de proceder con la activación de una foto de perfil.

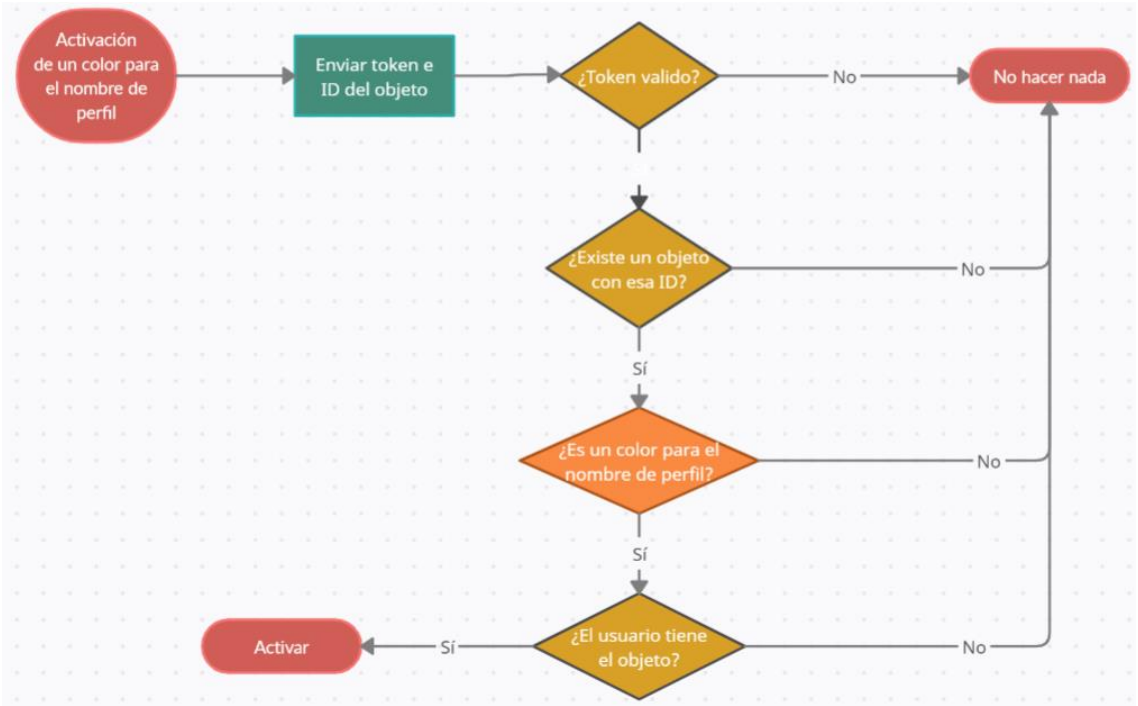


Figura 118. Comprobaciones realizadas antes de proceder con la activación de un color para el nombre de perfil.

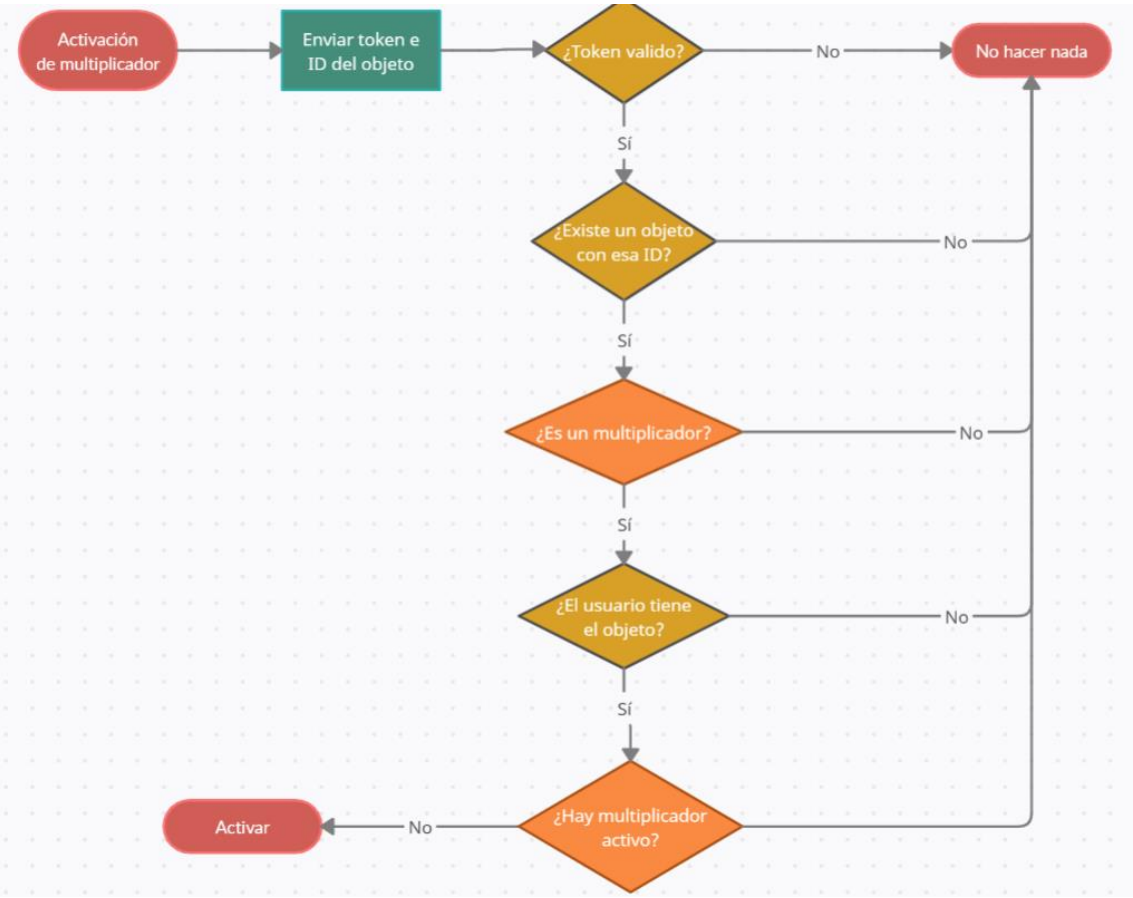


Figura 119. Comprobaciones realizadas antes de proceder con la activación de un multiplicador.

Cabe destacar que, mientras que para activar un elemento estético solo es necesario llamar al *setter* correspondiente, mientras que para utilizar un multiplicador hay que crear una entrada en la tabla "Item_Used_User" y reducir en uno el atributo "quantity" de la tupla de la tabla "Item_User".

7.1.1.5 Devolver un objeto

Esta funcionalidad no viene reflejada en la Especificación de Requisitos Software**¡Error! La autoreferencia al marcador no es válida.**, puesto que es el resultado de la validación con el usuario en las Pruebas. Al devolver un producto, se realizan ciertas comprobaciones al igual que cuando se compra o activa un objeto. Este objeto solo puede ser un elemento estético, por lo que se elimina la tupla de "Item_User" en la base de datos, se registra la devolución en "User_History" y se devuelve el importe pagado al usuario.

7.1.1.6 Recompensar al usuario

Lamentablemente, durante el desarrollo del proyecto, el módulo de ejercicios sufría ciertos errores que impedían la correcta y completa implementación de la funcionalidad tratada

en este apartado. Es por ello por lo que se decidió preparar la función encargada de otorgar la recompensa al usuario y dejar, como trabajo futuro, la implementación completa de esta funcionalidad. Para poder probar este método, se simula el envío y corrección de un ejercicio y se le pasa directamente la nota obtenida y la última mejor nota.

El comienzo de este método puede verse en la Figura 120, donde se comprueba si se ha mejorado la nota o si es la primera vez que se realiza el ejercicio. Posteriormente, se obtiene la recompensa que debería ofrecer el ejercicio, la cual se calcula conociendo su dificultad y el nivel que posee el usuario (ver Figura 121). Conociendo esta nota, se calcula la recompensa de Monedas (ver Figura 122). Ambos cálculos se realizan en sus propios métodos con el fin de aumentar la cohesión y facilitar el mantenimiento posterior en el caso se quisiese modificar la fórmula.

```java
// TODO: When experience is added, add a experienceReward parameter
public void rewardUser (User user, int exerciseHardness, Double bestMark, Double lastMark) {
    boolean firstTime = false;
    if (bestMark == null) {
        bestMark = 0.0;
        firstTime = true;
    }
    double difference = lastMark - bestMark;
    if (difference <= 0 ) { // There is no reward
        return;
    }

    double exerciseCoinReward = exerciseService.getExerciseCoinReward(user, exerciseHardness);
    double coinReward = this.calculateCoinReward(exerciseCoinReward, bestMark, lastMark, firstTime);
```

Figura 120. Primera parte del método rewardUser. Comprueba si el usuario merece una recompensa y la calcula. Se encuentra en ItemServiceImpl.java

```java
protected double getExerciseCoinReward (User user, int exerciseHardness) {
    double userLevelScaledToHardness = userService.getUserLevelScaledToHardness(user);
    double reward = (exerciseHardness * BASE_COIN_REWARD) * (exerciseHardness/userLevelScaledToHardness);
    return Math.floor(reward);
}

protected double getExerciseLaureaReward(User user, int exerciseHardness, boolean firstTime) {
    int userLevelScaledToHardness = userService.getUserLevelScaledToHardness(user);
    if (userLevelScaledToHardness > exerciseHardness) {
        return 0;
    }
    if (firstTime) {
        return exerciseHardness + 1; // Extra Laurea
    }
    return exerciseHardness;
}
```

Figura 121. Métodos getExerciseCoinReward y getExerciseLaureaReward. Se encargan de obtener la recompensa de un ejercicio en base a su dificultad y el nivel del usuario. Se encuentran en ExerciseServiceImpl.java

```java
private double calculateCoinReward(double exerciseCoinReward, double bestMark, double lastMark, boolean firstTime) {
    double difference = Math.pow(lastMark,2) - Math.pow(bestMark,2);
    double percentage = Math.max(0, difference/100);
    if (difference == 100 && firstTime) {
        percentage += 0.1; // Extra 10%
    }
    return Math.floor(exerciseCoinReward * percentage);
}
```

Figura 122. Método calculateCoinReward. Se encarga de obtener la recompensa de Monedas que recibirá el usuario. Se encuentra en ItemServiceImpl.java

La segunda parte de *rewardUser* puede verse en la Figura 123. Aquí se comprueba si el usuario posee un multiplicador activo para duplicar o triplicar la recompensa ganada. También se entregan las Láureas correspondientes si la nota obtenida ha sido un 10.

```
StringHolder itemCategory = new StringHolder();
ItemUsedUser lastMultiplier = this.getLastMultiplier(user, itemCategory);

if (lastMultiplier != null) { // The user used at least one multiplier
    Calendar dateOfUse = lastMultiplier.getDate();
    Calendar dateOfEnding = Calendar.getInstance();
    dateOfEnding.setTimeInMillis(dateOfUse.getTimeInMillis());

    dateOfEnding.add(Calendar.HOUR_OF_DAY, Integer.parseInt(lastMultiplier.getItem().getValue()));
    Calendar now = Calendar.getInstance();

    /* TODO: Multiply the reward only if the user started the exercise X hours before (like 1h).
    /* TODO: Otherwise, the user could finish a lot of exercises and doesn't send its until activate the multiplier.
    */
    if (now.compareTo(dateOfEnding) < 0) { // The multiplier still valid
        if (itemCategory.value.equals("Duplicator")) {
            coinReward *= 2;
        } else { // The multiplier is a triplicator
            coinReward *= 3;
        }
    }
}

if (lastMark == 10) {
    double exerciseLaureaReward = exerciseService.getExerciseLaureaReward(user, exerciseHardness, firstTime);
    user.addLaureas(new BigDecimal(exerciseLaureaReward));
}
user.addCoins(new BigDecimal(coinReward));
userRepository.save(user);
```

Figura 123. Segunda parte del método rewardUser. Se comprueba si hay un multiplicador activo y se entrega la recompensa. Se encuentra en ItemServiceImpl.java

Podemos destacar que hay una serie de *TODO* en el método que se tendrán que implementar cuando las funcionalidades correspondientes estén desarrolladas.

7.2 Lado del cliente

Antes de comenzar a desarrollar la tienda y el inventario, se activó la propiedad “strict” del *store*. Al activarla, en el caso de que se modifique el *state* del store a través de cualquier función que no sea una mutación, saldrá un aviso en el navegador. El uso de esta propiedad es una buena práctica, porque permite al desarrollador estar seguro de que lo que modifica el *state* son las mutaciones y nada más. Sin embargo, es importante que, una vez se pasa a producción, se desactive esta propiedad[90].

7.2.1 Tienda

Para el desarrollo de la tienda en el lado del cliente se creó un módulo adicional en el store (*shop.js*), así como su respectivo módulo en la capa de servicios encargada de comunicarse con el servidor. La pantalla final de la tienda se puede ver en la Figura 124 y en la Figura 125. Para poder realizarla, se implementó la vista correspondiente. Esta vista se encarga de conseguir los objetos diarios, su temporizador y los objetos permanentes. A continuación, se explicará

cómo se obtiene cada una de estas cosas. La obtención y cálculo de estos datos solo se realiza si el usuario entra en la tienda.

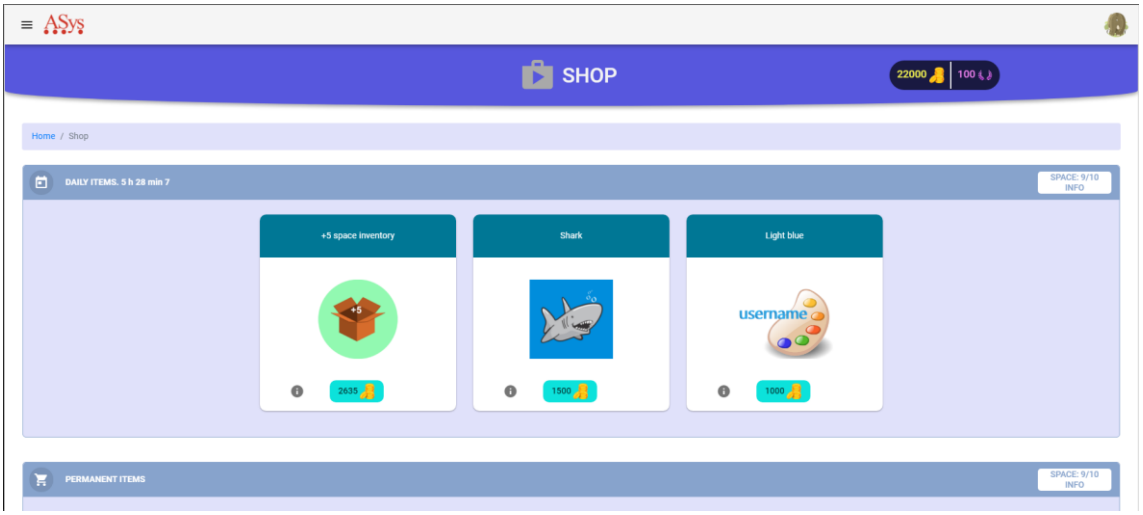


Figura 124. Primera parte de la pantalla de la tienda.

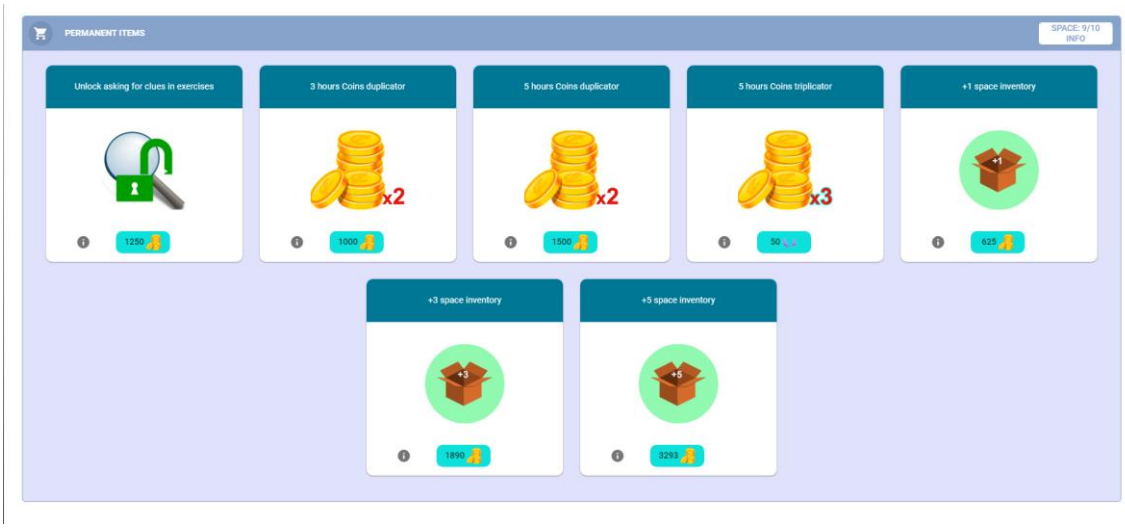


Figura 125. Segunda parte de la pantalla de la tienda.

7.2.1.1 Objetos diarios

Para implementar esta funcionalidad, se crearon variables en el *store* encargadas de almacenar los objetos diarios y su fecha. De esta manera, cuando el usuario accede a la tienda, los objetos diarios solo se solicitan si no están almacenados o ya no son del día actual. Si están almacenados, se comprueba la fecha y, si esta es del día actual, simplemente se muestran (ver Figura 126). Esto permite ahorrar bastantes consultas a la base de datos.

```
async setDailyItems() {
 // Check store
 let dailyItemsStore = this.dailyItems

 if (dailyItemsStore) {
   // First time user login into the app and enters the shop
   await this.$store.dispatch("shop/GET_DAILY_ITEMS_FROM_DATABASE", this.profile.id)
 } else {
    let dailyItemsDateStore = new Date(dailyItemsStore.date)

    // If the dailyItemsDateStore is earlier than today, is necessary to reset the dailyItems.
    if (dailyItemsDateStore.getTime() - this.todayAtMidnightInMillis < 0) {
      this.resetDailyItems();
    }
 }
```

Figura 126. Función setDailyItems. Se encarga de obtener los objetos diarios del usuario.

Como se podrá haber visto en la Figura 124, los objetos diarios poseen un temporizador que, una vez que termine, se reiniciarán los objetos diarios. Este temporizador utiliza la hora del servidor y se reinicia cuando sean las 00:00 en el reloj del servidor. La fecha que se usa para comparar, también se obtiene del servidor. Es decir, en ningún momento se realiza un *new Date()* en el lado cliente (sí que se utiliza pero con el tiempo del servidor en milisegundos como parámetro), de manera que no es posible modificar la hora de un dispositivo para intentar "hacer trampas". Los métodos encargados de mostrar el temporizador se encuentran en la Figura 127.

```
showTimer(distance) {
    if (distance < 0) {
      // Reset the timer and dailyItems
      clearTimeout(this.timeout);
      this.resetDailyItems()
      this.resetTimer();
      return;
    }

    const hours = Math.floor((distance % this.day) / this.hour);
    const minutes = Math.floor((distance % this.hour) / this.minute);
    const seconds = Math.floor((distance % this.minute) / this.second);

    this.timer = `${hours} h ${minutes} min ${seconds}`
    this.timeout=setTimeout(this.showTimer, 1000, distance-1000)
  },
  setTimer() {
    this.endDate = new Date(this.tomorrowAtMidnightInMillis);
    const now = new Date(this.serverTimeInMillis);
    const distance = this.endDate - now;
    this.timeout=setTimeout(this.showTimer, 1000, distance)
},
async resetTimer() {
  await this.getAllServerTimes();
  this.setTimer();
```

Figura 127. Métodos encargados de mostrar el temporizador de los objetos diarios

Por último, es importante mencionar que cuando se sale de la tienda, se ejecuta la función *delete* propia del ciclo de vida de un componente de Vue y se realiza un *clearTimeout* para detener el temporizador.

7.2.1.2 Objetos permanentes

Al igual que con los objetos diarios, en el *store* una variable encargada de almacenarlos, de manera que solo se solicitan si esta variable está vacía.

7.2.1.3 Comprar un objeto

Para poder comprar un objeto, se realizan ciertas comprobaciones en el lado del cliente y si no se cumple alguna, no dejará comprarlo indicando el motivo. Por un lado, comprobará que el usuario posee espacio de inventario suficiente siempre y cuando el objeto ocupe espacio

(Figura 128). Por otro lado, se comprobará si hay fondos suficientes para comprar el objeto (ver Figura 129).

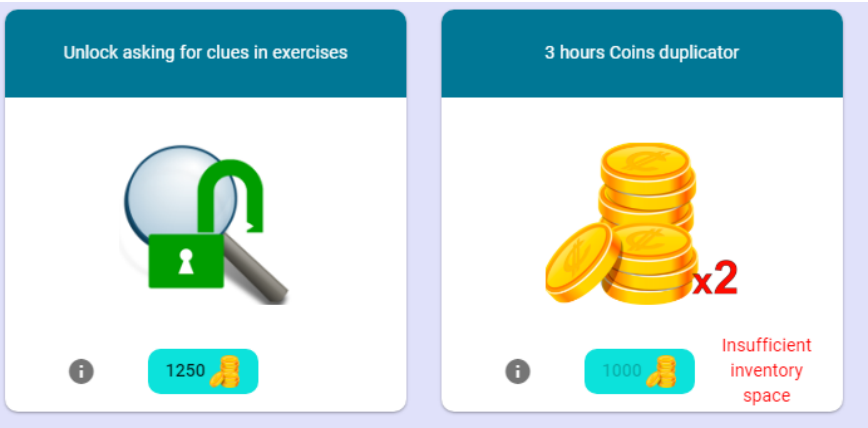


Figura 128. Mensaje de espacio de inventario insuficiente para aquellos objetos que ocupen espacio de inventario.

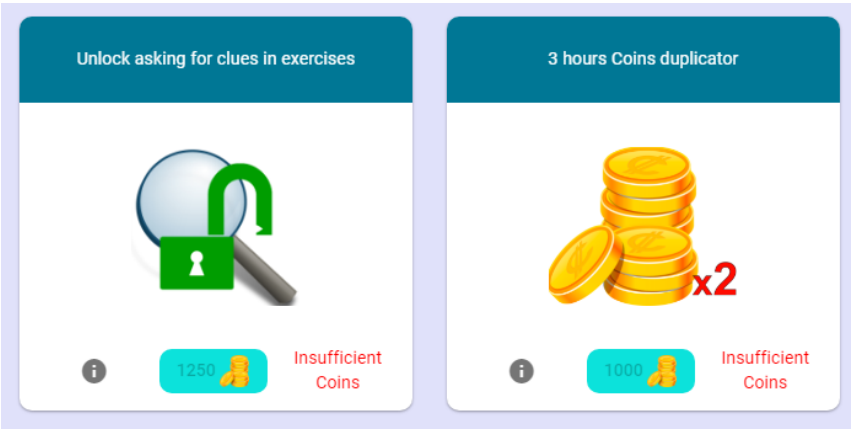


Figura 129. Mensaje de fondos insuficientes.

Una vez comprado el objeto, si todo ha ido de manera correcta en el servidor, se actualizará la variable que contiene toda la información del usuario, llamada "profile" que se encuentra en el store en un fichero user.js. Además, se actualizarán las variables que se encargan de gestionar el espacio del inventario establecidas en la Tabla 13 y detalladas en el epígrafe 6.2.3.1.3 Aumentar la capacidad máxima de inventario. Esta función (ver Figura 130), en el caso de que se compre un objeto de incremento del espacio de inventario, ejecutará otra función encargada de incrementar su precio. Este cálculo se hace en el cliente para evitar realizar una petición al servidor, puesto que no requiere mucha carga la operación y sirve para que el precio se muestre correctamente. Si no se actualizase la vista de este precio, al comprar otro objeto de este estilo, el usuario vería como se le quita más dinero de lo que esperaba, puesto que el servidor sí que hace bien el cálculo.

```
['UPDATE_INVENTORY_CAPACITY_AND_NUMBER_OF_INVENTORY_ITEMS'] : ({commit, dispatch, state}, item) => {
    let categoryName = item.category.name;
    switch (categoryName) {
        case "InventoryUpgrade":
            commit('ADD_INVENTORY_CAPACITY', Number(item.value))
            dispatch('shop/INCREASE_PRICE_OF_INVENTORY_UPGRADE_ITEMS', state.inventoryCapacity, { root: true })
            break;
        case "ContentToUnlock": // Do nothing
            break;
        case "Clue":
            commit('ADD_NUMBER_OF_INVENTORY_ITEMS', Number(item.value))
            break;
        default:
            commit('ADD_NUMBER_OF_INVENTORY_ITEMS', 1)
            break;
    }
},
```

Figura 130. Función encargada de actualizar las variables que gestionan el espacio del inventario.

Una vez realizadas estas operaciones, es necesario marcar el objeto como "vendido" en el caso de que no se pueda seguir comprando. En cuanto a los objetos permanentes, cada uno de ellos comprueba si es de un tipo diferente a "Consumable" o "Upgrade" y, en tal caso, comprueba si se encuentra en la lista de "ItemUser" (ver Figura 131).

```
isPurchased() {
  if (this.profile.itemsUser.length !== 0) {
    if (this.item.category.type !== 'CONSUMABLE' && this.item.category.type !== 'UPGRADE') {
      let found = this.profile.itemsUser.find(itemUser => itemUser.item.name === this.item.name)
      return Boolean(found)
    }
  }
  return false
```

Figura 131. Comprobación de un objeto permanente para saber si se ha comprado o no.

Por otra parte, para determinar si un objeto diario ha sido comprado, cada objeto comprobará si en la lista de "UserHistory" se encuentra ese objeto comprado el día actual. Esta lista de "UserHistory" se obtiene solicitándola al servidor cuando se entra a la tienda (siempre y cuando no esté vacía) y almacena en una variable del *store* llamada "dailyItemsPurchasedToday" (ver Figura 132). Sin embargo, cuando se compra un objeto diario, como se actualiza la variable "profile", sabemos que el último elemento de esta lista es el objeto diario recién comprado, por lo que lo extraemos y lo añadimos a "dailyItemsPurchasedToday", evitando así una consulta adicional al servidor. En la Figura 133

podemos ver como se realiza esto en la última llamada, siendo resp.data el nuevo valor de "profile".

```
isPurchased() {
  if (this.dailyItemsPurchasedFromToday) {
    let found = this.dailyItemsPurchasedFromToday.find(userHistory =>
        userHistory.eventName === this.item.name
        && (userHistory.typeEvent === "SPENDING_DAILY_ITEM"
          || userHistory.typeEvent === "SPENDING_OTHER")
        )
    return Boolean(found)
  } else return false
```

Figura 132. Comprobación de un objeto diario para saber si se ha comprado o no.

```
['BUY_DAILY_ITEM']: ({dispatch, commit}, {item}) => {
    service.shop.buyDailyItem(item.id)
        .then(resp => {
            dispatch('user/SET_PROFILE', resp.data, {root: true})
            dispatch('user/UPDATE_INVENTORY_CAPACITY_AND_NUMBER_OF_INVENTORY_ITEMS',item, {root: true})
            commit('ADD_ITEM_TO_DAILY_ITEMS_PURCHASED_FROM_TODAY', resp.data.userHistory[resp.data.userHistory.length - 1])
        })
},
```

Figura 133. Acción del store que se encarga de comprar un objeto diario.

7.2.2 Inventario

Para implementar el inventario no se creó ningún módulo del *store* ni de la capa de servicio, sino que se utilizó parte del módulo de la tienda y parte del módulo de usuario. Para llegar al inventario se añadió un botón en la pantalla del *dashboard* (ver Figura 134). La pantalla principal del inventario se realizó creando una nueva vista que podemos ver en la Figura 135. Aquí el usuario podrá ver la cantidad de objetos que posee de cada tipo y, exceptuando las pistas, podrá ver con más detalle los diferentes objetos comprados.

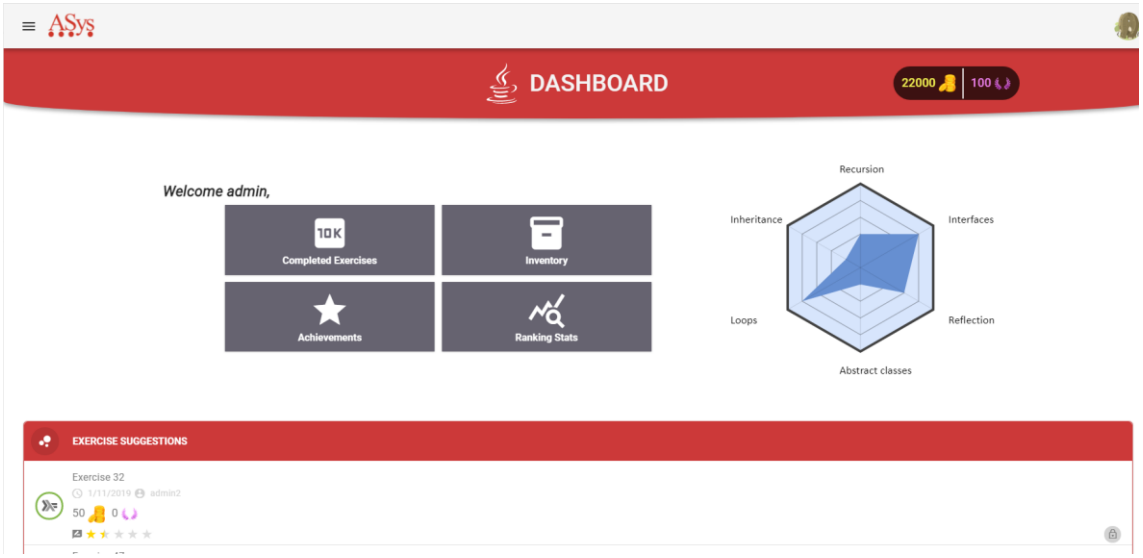


Figura 134. Pantalla del dashboard con un botón para acceder al inventario.

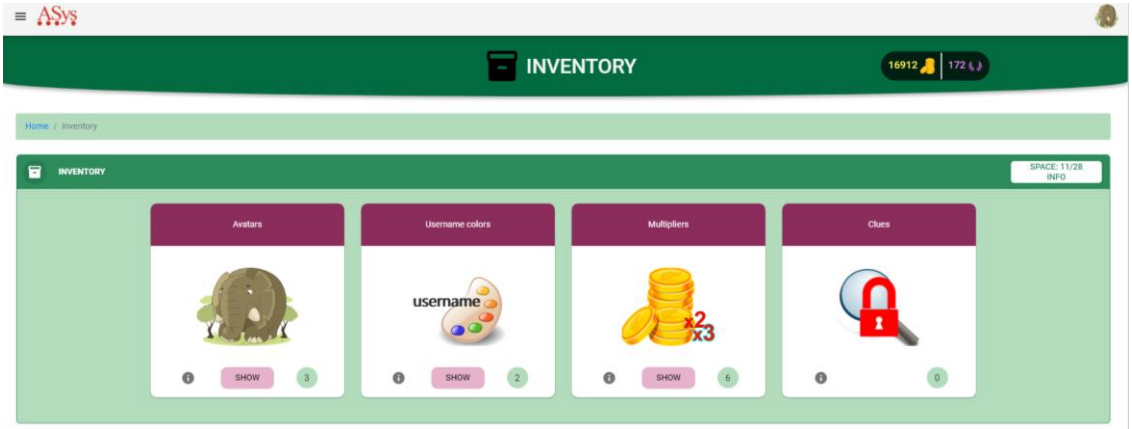


Figura 135. Pantalla principal del inventario.

En esta pantalla destacamos que las categorías mostradas no corresponden exactamente a las categorías definidas en el servidor, pues en "Multipliers" se encuentran tanto los duplicadores como los triplicadores de Monedas. Además, el logo de las fotos de perfil y el color del nombre de perfil no es estático, sino que depende del que tenga activado el usuario en ese momento. Para conseguir esto, se ejecuta una función computada que obtiene categorías personalizadas a partir de las que existen en el servidor, solicitadas previamente y almacenadas en el *store* del usuario. Ahora bien, si modificamos directamente estas categorías nos aparecerá un aviso debido al modo *strict* activado, pues estamos modificando el *store* sin mutaciones. Si intentamos utilizar la sintaxis *spread* de Javascript para clonar el array, tampoco funcionará, puesto que se trata de un array de objetos los cuales tienen a su vez otras referencias. Es por ello

por lo que se optó por utilizar la función *clone* de la librería *rfdc* [91]. Esto solucionó nuestro problema.

7.2.2.1 Uso de objetos

Para activar los objetos hay que entrar en su sección dedicada.

7.2.2.1.1 Fotos de perfil y colores para el nombre de perfil

La pantalla para ver las fotos de perfil compradas se puede ver en la Figura 136, mientras que la de los colores para el nombre de perfil se encuentra en la Figura 137. Aquí el usuario puede activar la foto o color que desee y posea. Este se situará en el primer lugar de la lista. Además, al activarse, el botón del objeto se desactiva. Por otra parte, tanto la nueva foto de perfil actualizada como el nuevo color se mostrarán en todas las partes donde sea necesario.

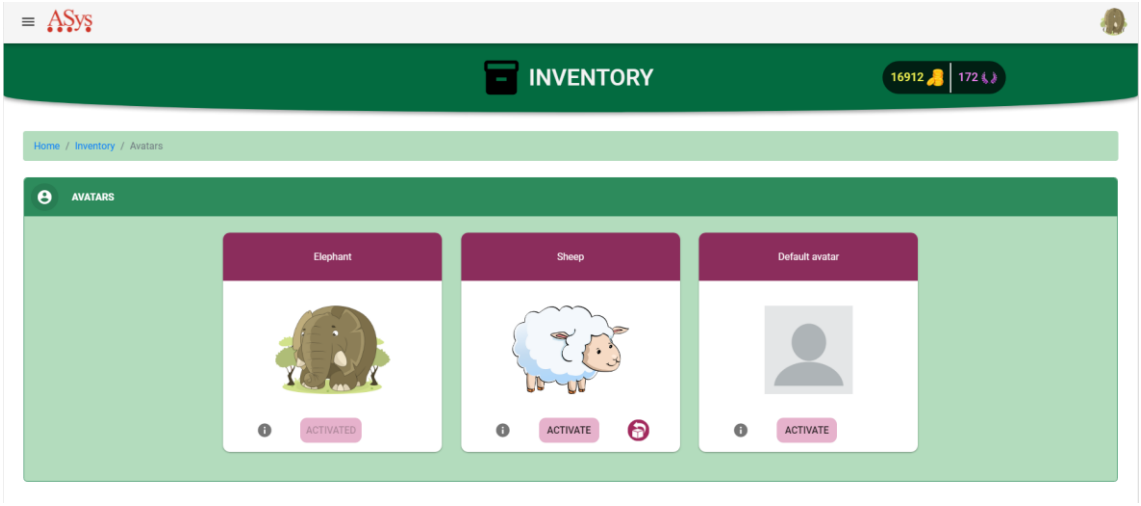


Figura 136. Pantalla del inventario donde se pueden ver las fotos de perfil compradas y activar la que se desee.

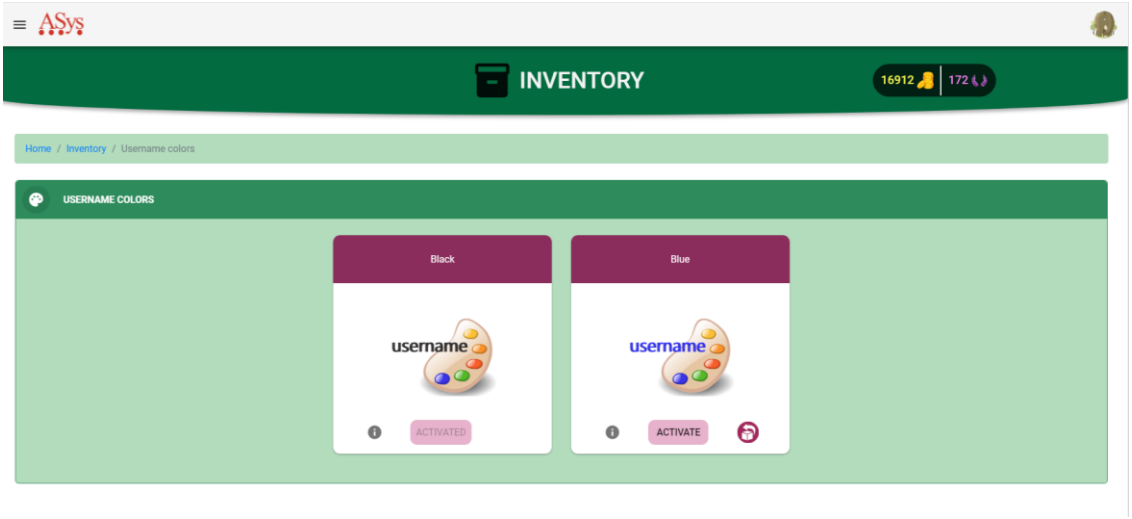


Figura 137. Pantalla del inventario donde se pueden ver los colores para el nombre de perfil comprados y activar el que se desee.

7.2.2.1.2 Multiplicadores

La pantalla de los multiplicadores es análoga a las dos recientemente mostradas (ver Figura 138) aunque aquí, cuando se activa un multiplicador se ejecuta una función parecida al temporizador de los objetos diarios y se muestra un temporizador como el de la Figura 139. Además, se actualizan las variables que gestionan el inventario, pues el usuario ahora posee un objeto menos, y el multiplicador se almacena en una variable del *store* con el objetivo de evitar consultas a la base de datos para obtenerlo en futuras ocasiones.

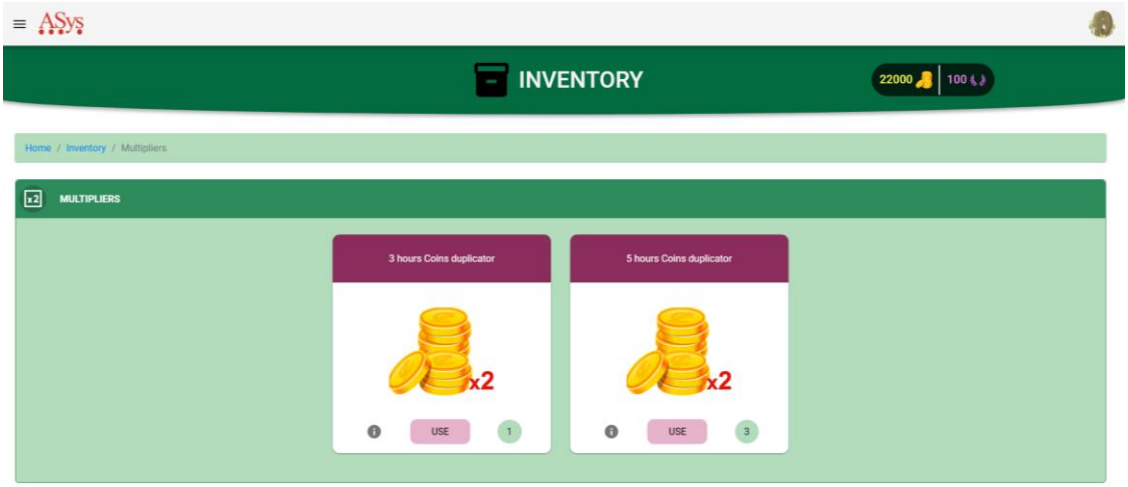


Figura 138.Pantalla del inventario donde se pueden ver los multiplicadores comprados y activar el que se desee.

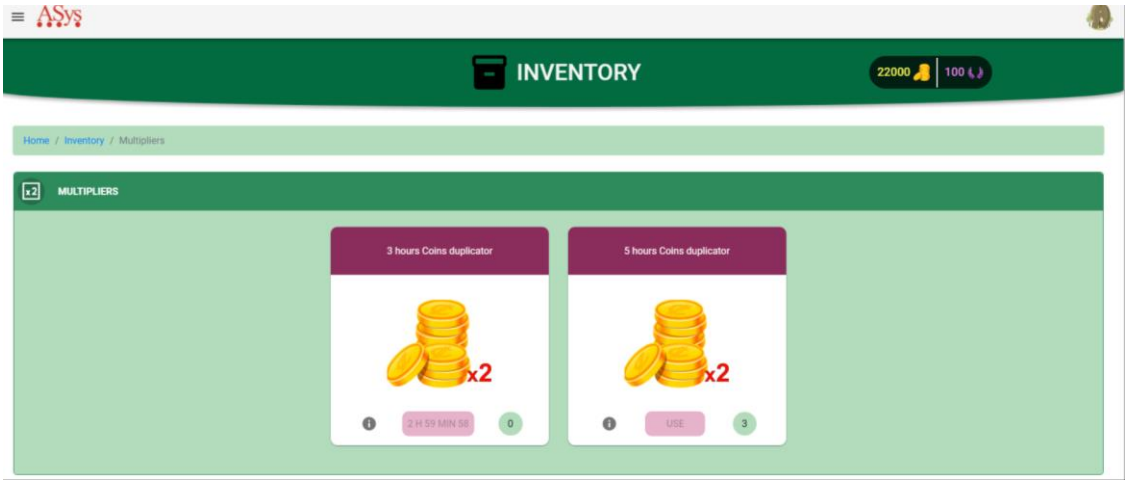


Figura 139. Pantalla del inventario donde se pueden ver los multiplicadores comprados. Ahora mismo hay uno activado.

7.2.2.2 Devolver un objeto

Tal como se explicó anteriormente, esta funcionalidad se añadió durante la fase de pruebas tras la sugerencia de un usuario. Como se puede observar en la Figura 136 y en la Figura 137, al lado del botón de activar, hay un botón que sirve para devolver el objeto. En la Figura 140 se puede ver con más detalle. En esta figura, no aparece el cursor debido a la configuración de la herramienta empleada para capturar la pantalla, pero está situado sobre el botón. Para hacer funcionar esto, se utiliza una variable del *store* consultada previamente al servidor: "numberOfRemainingTokens". Para determinar si se puede devolver o no, se comprueba que esta variable sea mayor que cero. En caso de llegar a cero o el objeto es un objeto por defecto, el botón no aparecerá.

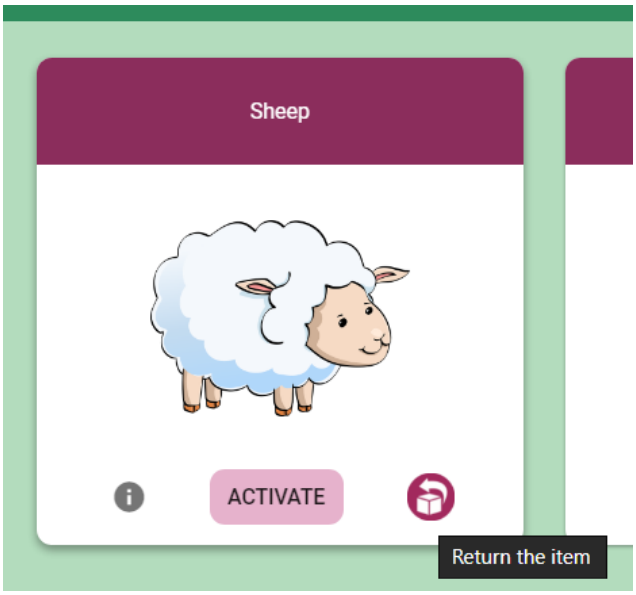


Figura 140. Botón de para devolver el objeto.

7.2.3 Objetos

Los elementos que representan los objetos, aunque parezcan todos iguales (o casi), en realidad son diversos componentes que siguen una estructura peculiar, resultado de un análisis y reflexión. Esto se debe a que dependiendo del tipo de objeto que sea (objeto de la tienda en sección objeto diario, sección objeto permanente, objeto de inventario, etc.) tendrá un comportamiento u otro y tendrá unos elementos HTML u otros. Y, con el fin de reutilizar la mayor cantidad de código, se investigaron diversas soluciones. En [92] se comentan posibles alternativas, donde destacan el uso de *slots* y el uso de *extends*.

7.2.3.1 Slots

Los slots [93] se pueden ver como un sistema jerárquico donde el componente padre añade esta etiqueta en su plantilla HTML y le asigna un nombre. Esa etiqueta será sustituida por otra plantilla HTML implementada por las clases hijas. Esto se puede ver mejor con el pseudocódigo de la Figura 141, donde "BaseItem.vue" declara el *slot* y "Avatar.vue" y "Duplicador.vue" lo utilizan. Al utilizarlo, tanto como "Avatar.vue" como "Duplicador.vue" tendrán la estructura de "BaseItem.vue" pero, en lugar de tener un slot, tendrán un botón. Y cada botón llamará una función propia de cada componente. Por último, algo a tener en cuenta con esta alternativa es que padre e hijos no comparten variables.

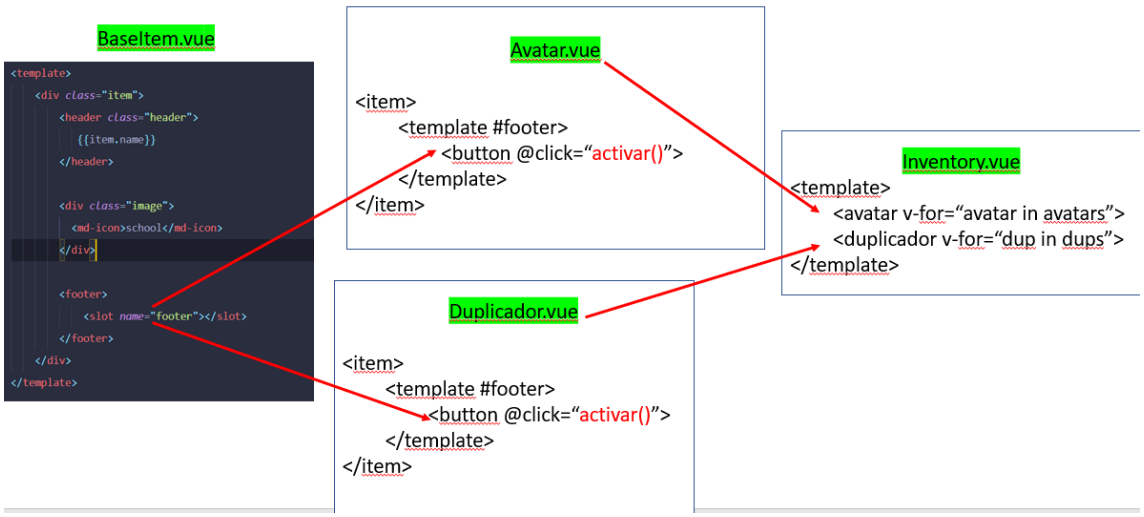


Figura 141. Pseudocódigo que utiliza slots.

7.2.3.2 Extends

El uso de *extends* [94] es una técnica bastante útil, pues permite reutilizar por completo el código del padre y añadir nuevas funciones. Además, en el código del padre se puede hacer referencia a variables y funciones definidas únicamente en el código del hijo. Aunque con esto último hay que tener cuidado, puesto que, si un hijo accede a ese trozo de código por error, saltará un error. Pero ¿y por qué debería ocurrir esto si cada hijo implementa lo que necesita? Esto se debe a que la desventaja que posee esta solución es que una vez que heredas con *extends*, no puedes seguir añadiendo elementos HTML, por lo que estarás obligado a añadir algún que otro condicional en el código del padre.

7.2.3.3 Decisión

Tras analizar ambas alternativas, se determinó que ambas poseen potencial y que no son soluciones excluyentes. Por lo que se decidió utilizar una estructura híbrida, tal como podemos ver en el pseudocódigo de la Figura 142.

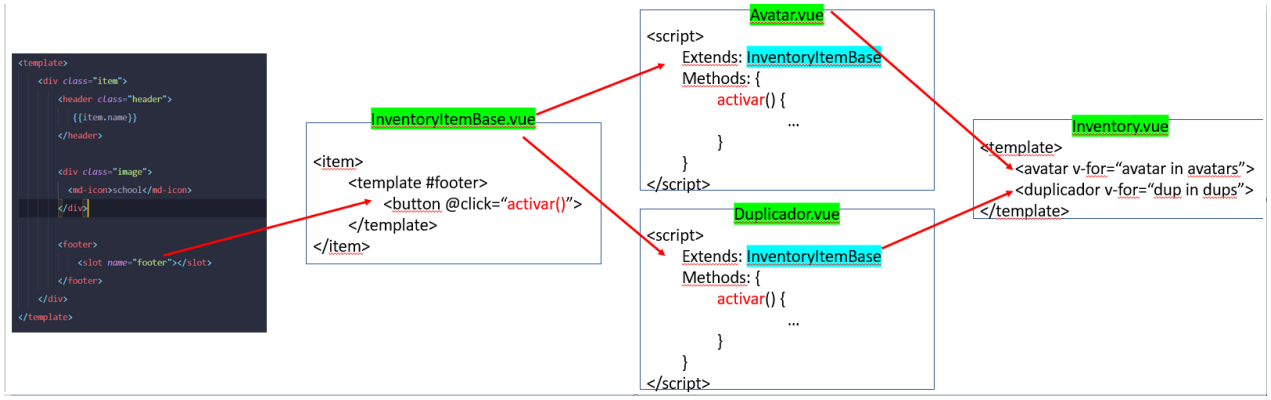


Figura 142. Pseudocódigo que utiliza slots y extends.

La estructura resultante podemos verla en la Figura 143. Los nombres de los componentes siguieron la guía de estilo de Vue "Tightly coupled component names" [95], aunque los componentes del último nivel del inventario se suprimió el prefijo del padre para evitar tener nombres extremadamente largos.

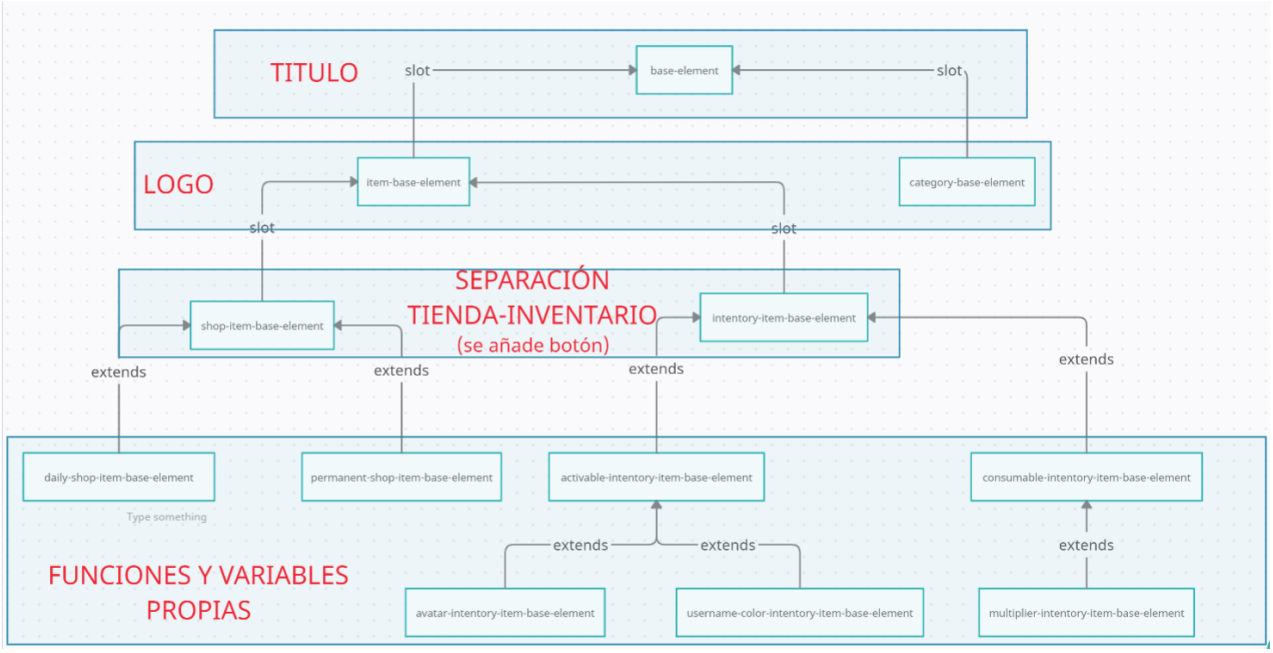


Figura 143. Estructura de los objetos de ASys en el lado cliente.

7.2.4 Carga de logos

Para cargar los logos situados en el servidor de MinIO, se realiza una petición GET. Una vez recibido el logo, este se almacena en la base de datos del navegador propia de HTML, IndexedDB. La próxima vez que se solicite este logo, primero se comprueba si ya existe en esta base de datos y si está, evitamos realizar una petición al servidor. Esto se utiliza tanto para los objetos de la tienda como para los objetos del inventario. También se utiliza para obtener los logos de las categorías mostradas en el menú principal del inventario (ver Figura 135), las cuales son cambiantes y requieren una consulta particular. Por este motivo se realizó una separación en la estructura de objetos y categorías (ver Figura 143).

Por otra parte, estas funciones también se utilizan para determinar la foto de perfil de los usuarios en las diferentes partes de la aplicación, pero aquí, inicialmente, surgía un problema: para cargar los avatares en los distintos lugares de la aplicación, era necesario modificar el código de una serie de componentes. En algunos de estos, los avatares pueden ir variando durante la vida del componente antes de su destrucción. Por lo tanto, es necesario recalcular el avatar cuando pase esto, pero es operación utiliza **promesas** de JavaScript y hay que realizarla de manera asíncrona. El problema es que las propiedades computadas de Vue no funcionan

correctamente con operaciones asíncronas, pues lo que devuelven es una promesa y no un valor. Las alternativas frente a esta situación son dos: utilizar la librería *asyncComputed* o usar *watchers.*

7.2.4.1 AsyncComputed

La librería de *asyncComputed* [96] ha sido desarrollada por un usuario con el fin de solucionar el problema que poseen las propiedades computadas. Esta alternativa solventa la problemática de una manera muy sencilla: añadiendo el código y clasificarlo *asyncComputed* (ver Figura 144).

```
asyncComputed: {
    getAvatar () {
      return this.getImageURL(this.$store.state.user.profile.photo)
    },
```

Figura 144. Ejemplo de código utilizando asyncComputed.

7.2.4.2 Watchers

Los *watchers* son una alternativa nativa de Vue y también sirven para solventar el problema con la contrapartida de que requieren más código (ver Figura 145).

```
data() {
    return {
      avatarURL: ""
    };
  },

created() {
    this.getImageURL(this.$store.state.user.profile.photo)
        .then(logo => this.avatarURL=logo)
    },
watch: {
    '$store.state.user.profile.photo': function(val) {
      this.getImageURL(val)
        .then(logo => this.avatarURL=logo)
    }
  }
```

Figura 145. Ejemplo de código utilizando watchers.

7.2.4.3 Decisión

Ambas alternativas consiguen solucionar el problema. La librería de asyncComputed permite tener un código más limpio que la solución nativa de los watchers. Sin embargo, en este foro [97], el creador de Vue y otros usuarios discuten sobre la librería *asyncComputed* y parece ser que no es la mejor solución, aunque esta discusión empezó en 2016 y ahora el paquete puede encontrarse en un estado más maduro. Otro punto a tener en cuenta es que, si usamos los

watchers, al ser una solución nativa, estamos seguros de que se mantendrán actualizados y no quedarán obsoletos. AsyncComputed, en cambio, al ser una librería no mantenida por los creadores de Vue, podría quedarse obsoleta con más probabilidad. Finalmente, se eligió utilizar los *watchers.*

7.2.5 Estructura de las vistas

Con el fin de reutilizar la mayor cantidad de código, en las vistas del inventario, al tener toda una estructura parecida, se utilizaron *slots* para la vista de las fotos de perfil, colores para el nombre de perfil y multiplicadores.

7.2.6 Monedas y Láureas

Tal como se puede apreciar en las diferentes figuras que muestran las diversas pantallas implementadas, arriba a derecha aparece la cantidad de Monedas y Láureas que posee el usuario. Esto se hizo modificando el componente ya existente “svg-toolbar-ar.vue”. Esto aparece en todas las pantallas de ASys.

7.2.7 Recompensas

Se añadieron las recompensas de los ejercicios en las diferentes pantallas de ASys. Como ejemplo, mostraremos el menú principal de los ejercicios (ver Figura 146), el diálogo de ejercicios (ver Figura 147), el menú de un ejercicio (ver Figura 148) y el buscador de ejercicios (ver Figura 149).

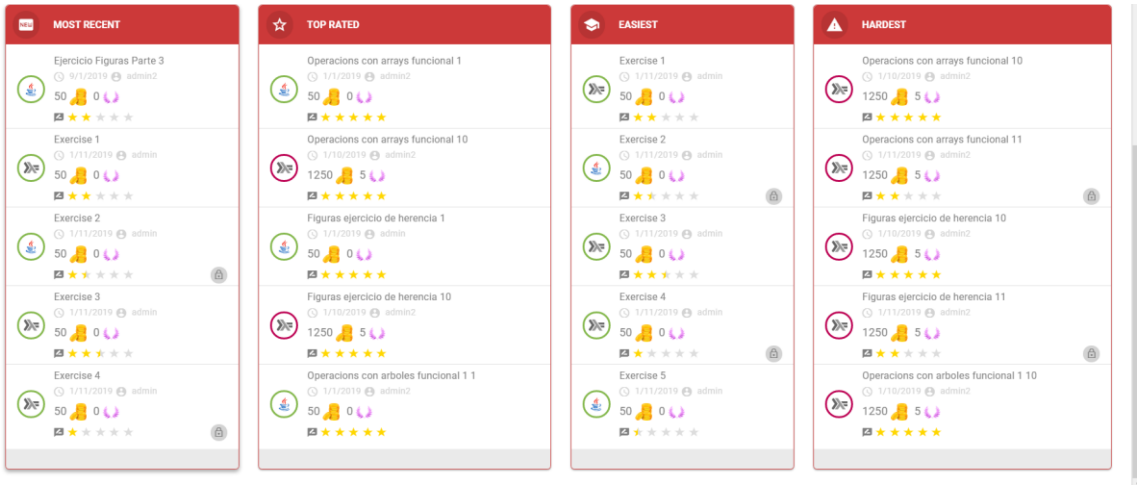


Figura 146. Menú principal de ejercicios con la recompensa de cada uno de ellos.

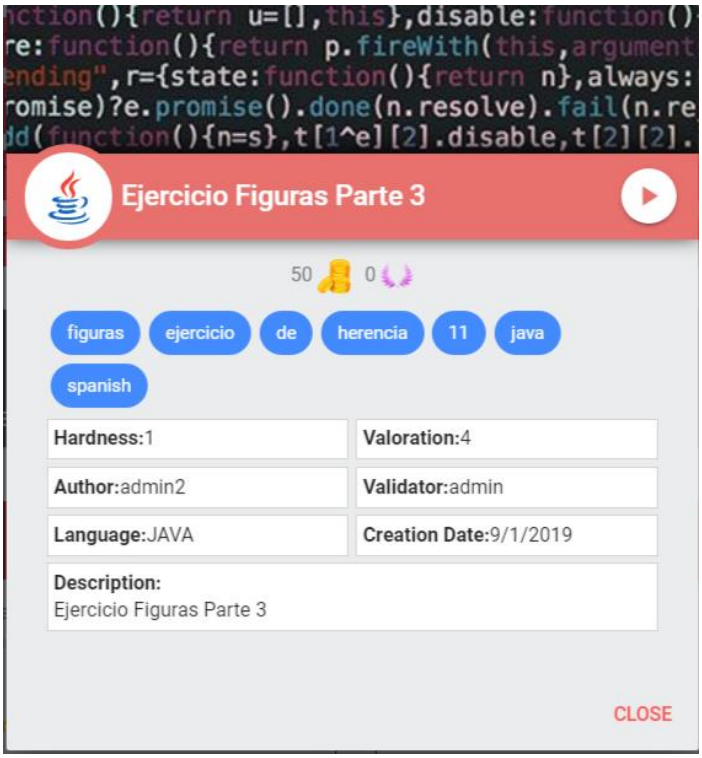


Figura 147. Diálogo de información de un ejercicio con su recompensa.

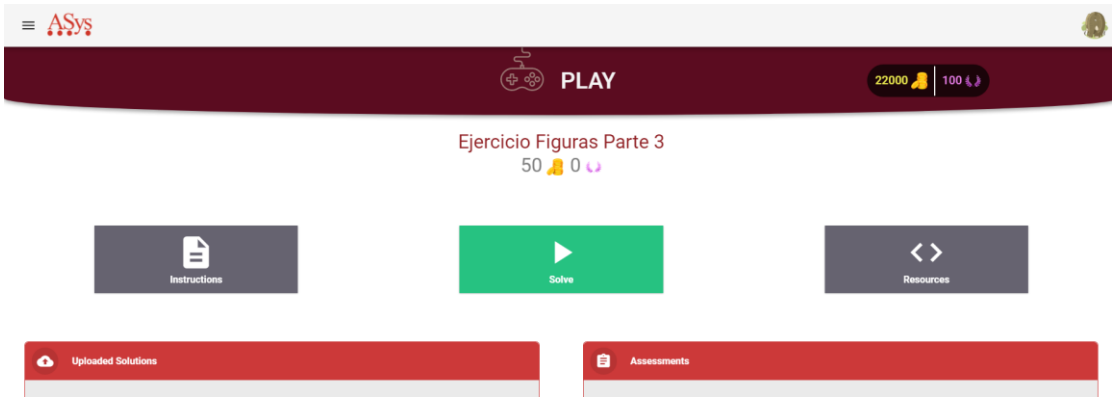


Figura 148. Menú de un ejercicio con su recompensa.

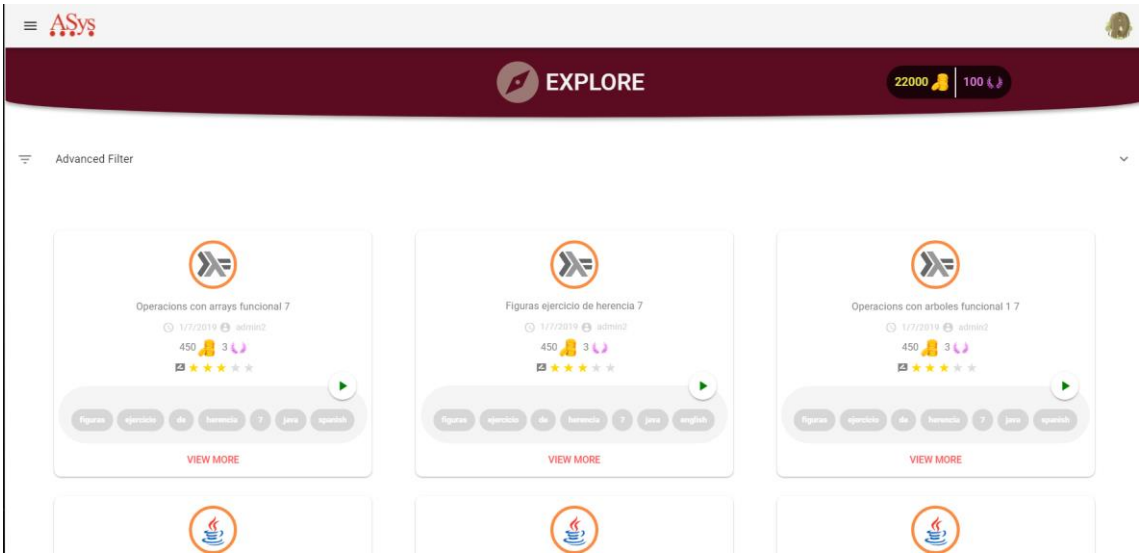


Figura 149. Buscador de ejercicios con la recompensa de cada uno de ellos.

Es importante mencionar que hay diversos componentes implicados que muestran esta información y, para ahorrar código, se realizó un *mixin* [98] con las funciones a utilizar (ver Figura 150).

```
export const exerciseRewardMixin = {
    methods: {
        getExerciseCoinReward(hardness) {
            let baseReward = this.$store.state.exercise.baseCoinReward;
            let userLevelScaledToHardness = this.$store.state.user.userLevelScaledToHardness;
            return Math.floor((hardness * baseReward) * (hardness/userLevelScaledToHardness));
        },
        getExerciseLaureaReward(hardness) {
            let userLevelScaledToHardness = this.$store.state.user.userLevelScaledToHardness;
            if (userLevelScaledToHardness > hardness) {
                return 0;
            }

            return hardness
        }
    },
}
```

Figura 150. Mixin con las funciones encargadas de calcular la recompensa de los ejercicios.

7.2.8 Diálogos

En este proyecto se incluyeron diversos diálogos informativos (ver Figura 151) y de confirmación (ver Figura 152) y todos ellos siguieron la estética ya presente en ASys.

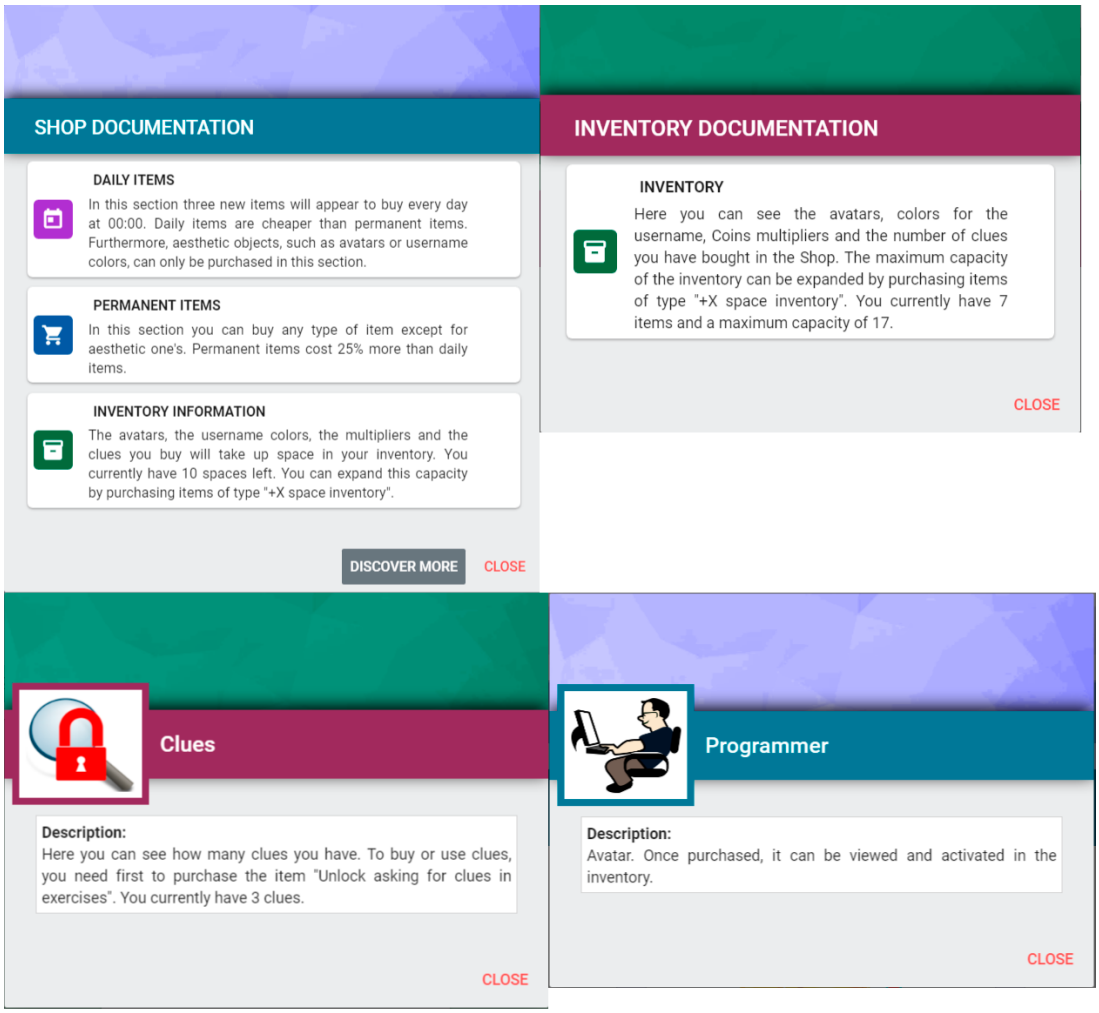


Figura 151. Diálogos informativos.

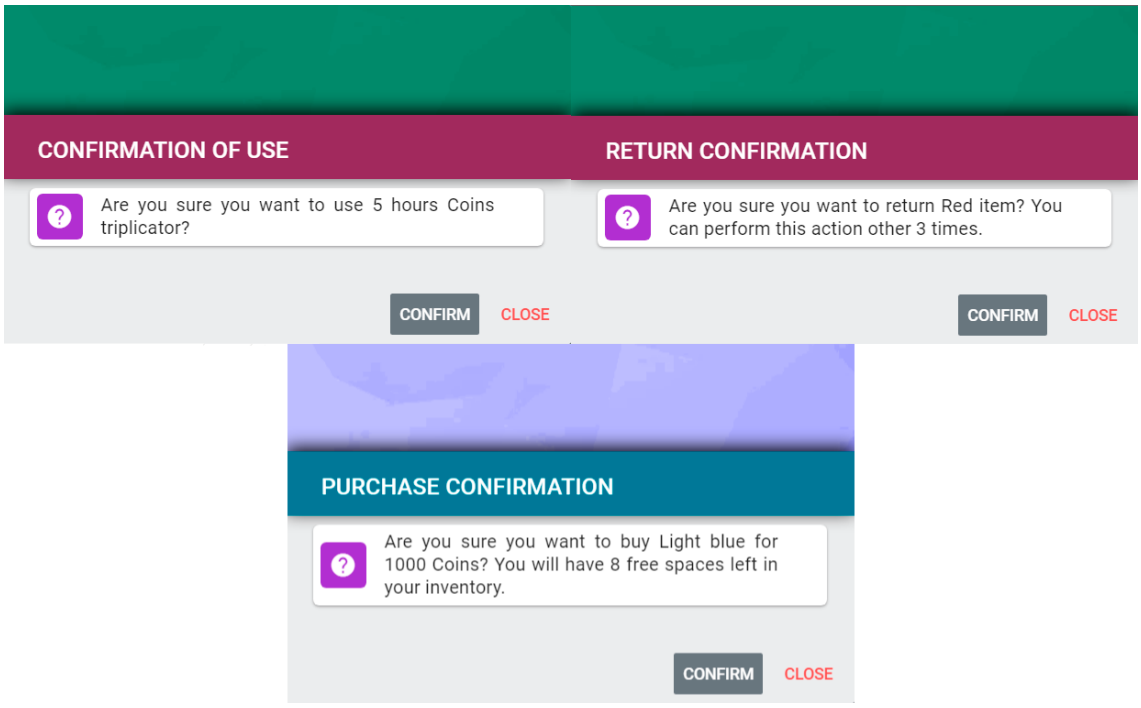


Figura 152. Diálogos de confirmación

7.2.9 Limitación de los colores para el nombre de perfil

Como resultado de las Pruebas de aceptación, se decidió añadir una opción en el menú de administración que limitase los colores para el nombre de perfil solamente al ranking de la aplicación (ver Figura 153). Si se desactiva esta opción, los colores aparecerán en más páginas de ASys. Actualmente esta acción solo puede realizarla un administrador y afectaría a todos los usuarios, pero en un futuro se pretende que esto sea personalizable para cada usuario y el código empleado para bloquear/desbloquear esta funcionalidad se podrá reutilizar para conseguir este fin.

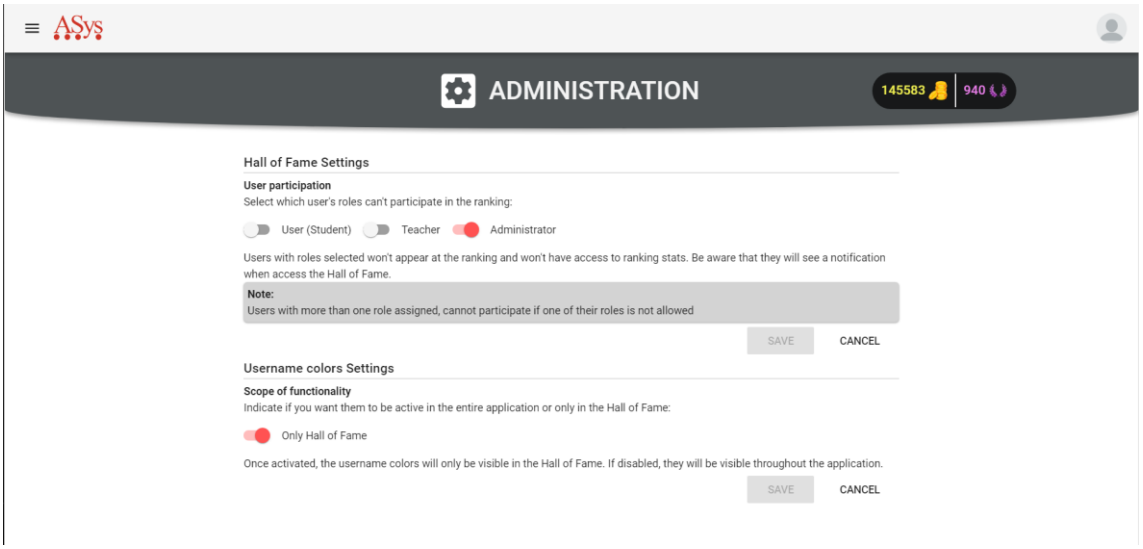


Figura 153. Menú de administración con la opción de limitar el uso de los colores para el nombre de perfil.

7.2.10 Documentación de la monetización

Tanto si se utiliza el botón “DISCOVER MORE” del diálogo informativo de la tienda (ver Figura 151) como si se clica sobre la Monedas o Láureas arriba a la derecha, se redirigirá a una nueva pantalla dedicada a la documentación de la monetización de ASys. Aquí se explica, en primer lugar, las unidades monetarias y, en segundo lugar, el sistema de ganancias de Monedas resumido del epígrafe Recompensa. El resultado final de esta pantalla se encuentra en la Figura 154, en la Figura 155 y en la Figura 156. Esta pantalla se desarrolló como resultado de las Pruebas de aceptación.

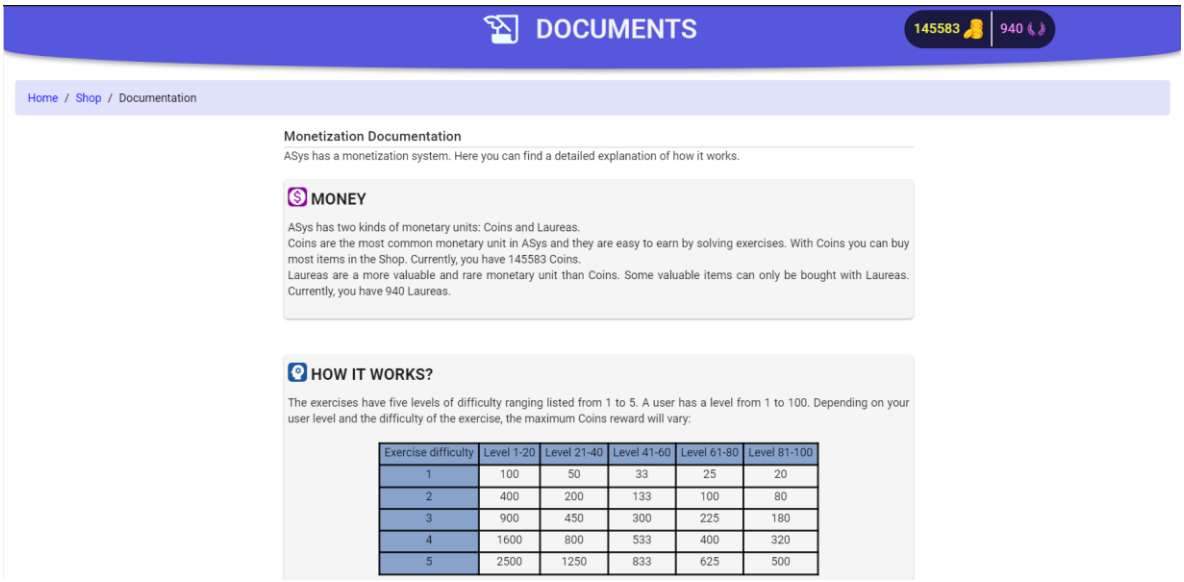


Figura 154. Primera parte de la página de documentación de la monetización.

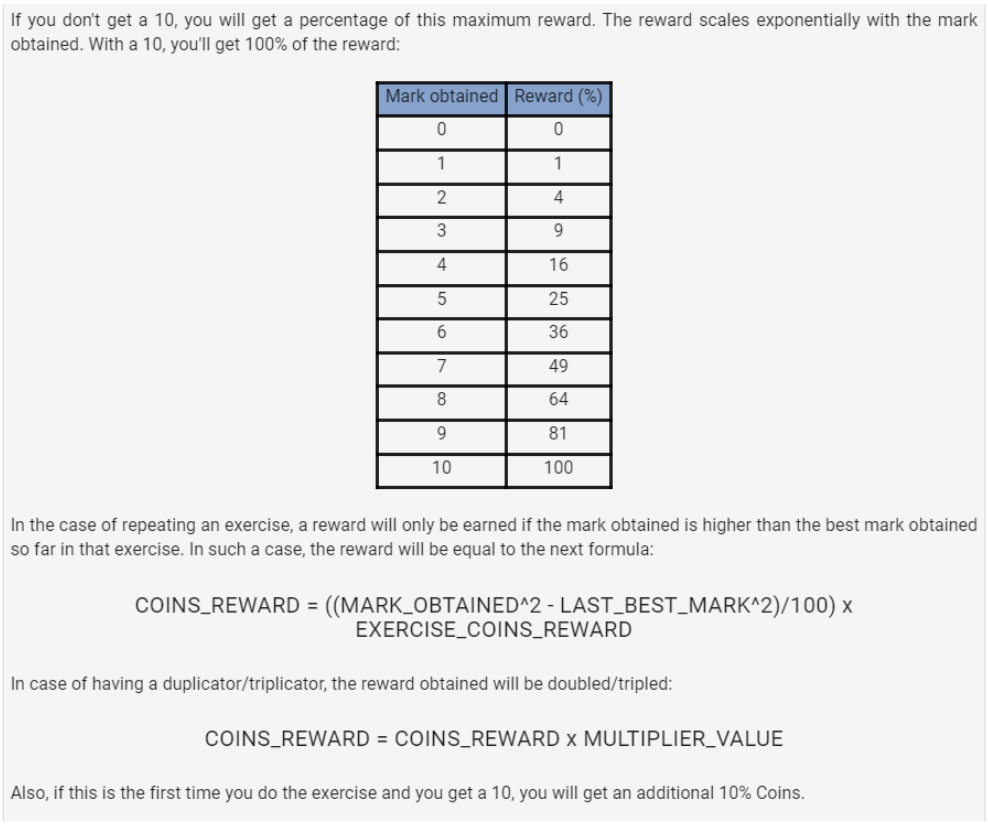
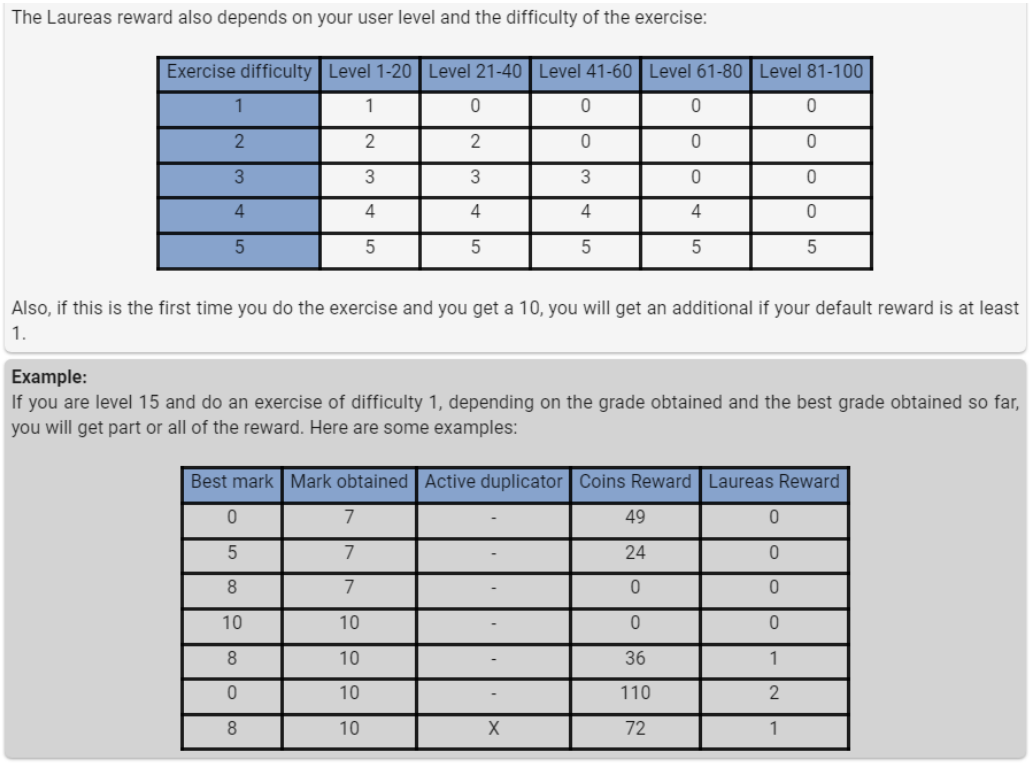


Figura 155. Segunda parte de la página de documentación de la monetización.



The Laureas reward also depends on your user level and the difficulty of the exercise:

Exercise difficulty	Level 1-20	Level 21-40	Level 41-60	Level 61-80	Level 81-100
1	1	0	0	0	0
2	2	2	0	0	0
3	3	3	3	0	0
4	4	4	4	4	0
5	5	5	5	5	5

Also, if this is the first time you do the exercise and you get a 10, you will get an additional if your default reward is at least 1.

Example:
If you are level 15 and do an exercise of difficulty 1, depending on the grade obtained and the best grade obtained so far, you will get part or all of the reward. Here are some examples:

Best mark	Mark obtained	Active duplicator	Coins Reward	Laureas Reward
0	7	-	49	0
5	7	-	24	0
8	7	-	0	0
10	10	-	0	0
8	10	-	36	1
0	10	-	110	2
8	10	X	72	1

Figura 156. Tercera parte de la página de documentación de la monetización.

7.2.11 Reutilización de componentes

Durante el desarrollo de este proyecto hemos procurado reutilizar, en lo posible, los componentes ya existentes en ASys. Entre ellos, destacamos el componente “list-tile-ar.vue”. Este es el contenedor que encapsula todos los componentes de los objetos en grupos (objetos diarios, objetos permanentes, inventario, etc). También hemos aprovechado el botón que posee este componente arriba a la derecha (insertado por un compañero que trabajaba en paralelo a este proyecto) para mostrar información adicional al usuario. Otro elemento reaprovechado ha sido la barra de migas, presente tanto en la tienda como en el inventario, un elemento ya existente en otros apartados de la aplicación y añadida a nuestro proyecto como resultado del proceso de validación de usuario en las Pruebas de aceptación realizadas.

7.2.12 Internacionalización

Durante todo el desarrollo se ha empleado la librería *l18n* de Vue por lo que todas las pantallas se pueden traducir del inglés al castellano y viceversa si se cambia el idioma de la aplicación. Además, el código está preparado para añadir nuevas lenguas.

7.2.13 Responsive

Todas las pantallas visualizadas en las figuras anteriores se mostraron desde la vista de un ordenador, pero están preparadas para ser vistas desde cualquier pantalla de cualquier tamaño y en cualquier navegador.

7.3 Problemas encontrados

7.3.1 LocalStorage

Durante el comienzo del proyecto se intentó usar el *localStorage* del navegador para ahorrar llamadas a la base de datos del servidor y hacer todo más eficiente. Otro motivo por el cual se usaba es que ya se está utilizando con el *profile* del usuario (ahora mismo el *localStorage* tiene al usuario serializado). Sin embargo, tras unos días trabajando con él, nos dimos cuenta de que cualquiera puede modificar el contenido del *localStorage* y esto es grave. Un usuario podría, por ejemplo:

Modificar su *id* y actualizar la página. Ahora Vuex en el *store* tiene el nuevo id del usuario y actuaría como si fuese otro. En nuestro ámbito, podría modificar sus objetos diarios o ponerse objetos que no tiene.

Afortunadamente nuestras peticiones se contrastan con la base de datos, pero habría que añadir cierta seguridad adicional en el lado del servidor que compruebe anomalías. Otra opción sería encriptar los datos del *localstorage*. O tal vez, sea mejor suprimir cualquier dato en el *localstorage*. En [99] se comentan diferentes soluciones que habría que analizar en el futuro. En nuestro caso, hemos decidido evitar utilizar el *localStorage* todo lo posible para garantizar mayor seguridad.

7.3.2 Sistemas de recompensas a medias

Tal como se explicó en el subapartado Recompensar al usuario159, debido a problemas ajenos a este proyecto, no se pudo completar al 100% el sistema de recompensas, pero sí se ha preparado todo lo necesario para que cuando el problema sea solventado, simplemente habrá que llamar al método encargado de recompensar al usuario.

8 Implantación

En este apartado se explica como implantar y configurar la presente solución en un entorno de explotación. En primer lugar, se detallarán los pasos de configuración previos a la instalación del proyecto. En segundo lugar, se explicará cómo proceder con la instalación recién mencionada.

8.1 Configuración

Para llevar a cabo la configuración de ASys es necesario seguir los pasos que se detallan a continuación.

8.1.1 Configuración de ASys Web Client

Primero, configuraremos y prepararemos el lado cliente de ASys en una serie de pasos:

1. En primer lugar, hay que situarse en el directorio del proyecto de ASys y ejecutar en una terminal el comando “npm install”. Una vez lanzado, se descargarán todas las dependencias del proyecto.

2. En segundo lugar, buscaremos el fichero “connection.js” en el directorio “src/services”. En este fichero habrá que inicializar la variable “baseURL” con el valor de la URL en la que se encuentra el servidor web. Si el servidor se encuentra en la propia máquina, un posible valor podría ser “http://localhost:8080/asysweb”, aunque si localhost no funcionase, se recomienda utilizar la IP privada de la propia máquina de la red a la que se encuentre conectada.

3. De manera similar al paso 2, buscaremos el fichero “index.js” en el directorio “src/store/” y le asignaremos a la variable “apiPath” el mismo valor asignado a la variable “baseURL”.

4. A continuación, construiremos el proyecto con el comando “npm run build” ejecutado en una terminal. Es importante ejecutarlo en el directorio del proyecto de ASys, al igual que en el paso 1. Este paso generará una carpeta con el nombre “dist”.

5. Finalmente, la carpeta “dist”, generada en el paso previo, habrá que moverla al servidor de la aplicación web.

8.1.2 Configuración de ASys Web Server

Una vez configurado el lado cliente, pasaremos a configurar el lado del servidor:

1. El primer paso para realizar consiste en abrir el fichero "application.yml". Aquí realizaremos un gran número de configuraciones. Su directorio es "src/main/resources/".

2. Para configurar la base de datos, dentro de "application.yml" buscaremos las propiedades "spring/datasource" e insertaremos el usuario y constraseña de la base de datos a la que queramos acceder.

3. Siempre dentro del mismo fichero, configuraremos las CORS [100] para permitir que nuestro cliente pueda acceder al servidor. Para ello, en la propiedad "cors/path1" pondremos la URL a través de la cual queremos recibir peticiones. En nuestro caso, podremos la dirección en la que vayamos a desplegar la aplicación web.

4. Sin cambiar de fichero, buscaremos la propiedad "jpa/hibernate/ddl-auto" y cambiaremos su valor a "none". Esta modificación hará que no se reinicie la base de datos con su contenido cada vez que se reinicie el servidor.

5. Posteriormente, buscaremos el fichero "JavaEmailUtil.java", situado en el directorio "src/main/java/com/asysweb/útil", para configurar el servidor SMTP [101] e insertaremos nuestras credenciales o, en su defecto, es posible utilizar una cuenta de MailTrap[28].

6. Para configurar el servidor de recursos MinIO, tendremos que acceder al archivo "AmazonS3ServiceImpl.java" situado en el directorio "src/main/java/com/asysweb/". Aquí tendremos que inicializar el atributo "minioClient" con las credenciales del servidor y la URL sobre la que se ha desplegado (Por ejemplo: "'http://localhost:9001").

7. En el archivo "pom.xml" verificaremos que se encuentra la propiedad llamada "packaging" con el valor de "war".

8. Por último, tendremos que construir el proyecto, obteniendo un archivo *war* como resultado, el cual utilizaremos para desplegar la aplicación. Este fichero generado habrá

[28] https://mailtrap.io

que renombrarlo con el mismo nombre con el que queramos acceder al servidor mediante la URL.

8.2 Instalación

Antes de desplegar ASys es imprescindible tener instalados cuatro servidores. Concretamente, tenemos que instalar:

- MySQL. Un Sistema de Gestión de Base de Datos relacional.
- MinIO. Un servidor de recursos
- Tomcat. Un servidor web.
- SMTP o Mailtrap. Un servidor de correo.

Una vez descargados, pasaremos a explicar ciertos detalles de instalación de aquellos que necesiten una determinada configuración.

8.2.1 Instalación de MySQl

Una vez descargado el servidor MySQL tenemos que indicar el usuario y contraseña que hemos especificado en la configuración del servidor de ASys. Por defecto, el usuario debería ser root. Una vez realizado este paso, habrá que crear una nueva base de datos e importaremos el script con la base de datos ya generada.

8.2.2 Instalación de MinIO

En segundo lugar, para instalar MinIO será necesario descargar, previamente, el binario correspondiente al SO de la máquina en la que se desplegará ASys. A continuación, extraeremos la carpeta MinIO del proyecto del servidor donde incorporaremos el binario recién descargado. Una vez realizados estos preliminares, tendremos que ejecutar este fichero binario desde la terminal. Para ello, nos situamos en el directorio donde esté ubicado y, dependiendo del sistema operativo, utilizaremos un comando u otro:

En Linux: “./minio server ./ --address “:puerto” ”

En Windows: “ minio.exe server ./ --address “:puerto” ”

Es importante remarcar que el puerto que hay que introducir en el comando es el mismo que hemos introducido en la URL de la configuración previa del fichero “AmazonS3ServiceImpl-java”.

Si es la primera vez que se despliega el servidor, se habrá generado una carpeta denominada ".minio.sys". Esta carpeta posee en su interior la configuración de las credenciales del servidor MinIO. Estas credenciales (accessKey y secretKey) hay que modificarlas para que coincidan con las que se utiliza nuestro servidor. Una vez aplicados los cambios, será necesario reiniciar el servidor de recursos MinIO.

8.2.3 Instalación de Tomcat

El tercer servidor que tendremos que instalar es el servidor de Tomcat. Durante este proceso de instalación, tendremos que añadir a la carpeta "webapps" los proyectos *war* (en el caso del servidor) y *dist* (en el caso del cliente), previamente obtenidos en los pasos de configuración. Una vez añadidos, podremos iniciar el servidor gracias al archivo "startup" situado en la carpeta "bin" del servidor.

Por último, es importante mencionar que se puede utilizar Wampstack[29] para gestionar el servidor Tomcat y el servidor MySQL desde un mismo punto. En este caso, se deberá utilizar la carpeta "htdocs" en lugar de "webapps" para insertar los proyectos *war* y *dist.*

[29] https://bitnami.com/stack/wamp

9 Pruebas

Dentro de un proyecto de Ingeniería del Software no puede faltar una fase dedicada a las pruebas. En esta fase es posible comprobar si el producto realizado cumple con las funcionalidades definidas en la Especificación de Requisitos Software. Además, gracias a las pruebas es posible detectar posibles errores pasados por alto durante la fase de implementación o desarrollo.

Concretamente, se realizaron tres tipos de pruebas. En primer lugar, se realizaron pruebas unitarias, donde se pudieron probar métodos individuales gracias a *Junit4*[30] y *Mockito*[31]. Posteriormente, se realizaron pruebas de integración, con el fin de comprobar la correcta compenetración de los diferentes componentes implementados de forma separada. Finalmente, se realizaron pruebas de aceptación con el tutor de este proyecto para verificar el correcto cumplimiento de los requisitos, las cuales se complementaron con la validación de la aplicación por parte de una serie de usuarios externos a ASys.

En suma, estas pruebas se acompañaron de un seguimiento de la calidad del código gracias a herramientas apropiadas. Esto se realizó con el fin de implementar un código sencillo de mantener, legible y simple.

9.1 Pruebas unitarias

Las pruebas unitarias permiten asegurar el correcto funcionamiento de los diferentes métodos individuales que componen las clases. Estas pruebas, de forma aislada, permiten verificar que los algoritmos de las diversas funciones cumplen con su fin. En caso de que exista algún error, la prueba fallará.

Los métodos que se probaron unitariamente estaban relacionados, principalmente, con los servicios relacionados con los objetos (o "ítems"). Tanto la compra como su uso. Para llevar a cabo estas pruebas, se utilizaron los *framework* de *Junit4* y de *Mockito.* Este último permite simular llamadas a métodos y devolver algo específico, con el fin de asegurarnos de que, si falla algo, saber a ciencia cierta que el culpable es el método que estamos probando. En la Figura 157

[30] https://junit.org/junit4/
[31] https://site.mockito.org

podemos ver cómo se ha realizado el *testeo* unitario del método *buyItem* encargado de gestionar la compra de un objeto. Aquí podemos observar que, antes de ejecutar el método a probar, estamos preparando los *mocks*, los cuales se han declarado al comienzo de la clase (ver Figura 158).

```
@Test
public void test_buyItem_Ok() {
    user.setCoin(new BigDecimal( val: 1000));
    Category category = new Category( name: "Avatar", TypeCategory.ACTIVABLE, description: "");
    Item item = new Item( name: "Car", new BigDecimal( val: 500), TypeOfCurrency.COIN, category, logoURL: "CarLogo", value: "CarLogo");

    parameters.put(UserServiceImpl.USER, user);
    parameters.put(UserServiceImpl.ITEM, item);

    // Mocking methods
    Mockito.when(serviceUtilsMock.checkPurchaseParameters(Mockito.any(String.class), Mockito.any(long.class),
            Mockito.any(TypeHistoryEvent.class))).thenReturn(parameters);
    Mockito.when(userServiceMock.getUserInventoryCapacity(Mockito.any(User.class))).thenReturn(UserServiceImpl.BASE_INVENTORY_CAPACITY);
    Mockito.when(userRepositoryMock.save(Mockito.any(User.class))).thenReturn(user);

    // Call the method to test
    Map<String, Object> result = itemService.buyItem(TEST_TOKEN, TEST_ITEM_ID, TypeHistoryEvent.SPENDING_DAILY_ITEM);

    // Verify Mocking
    Mockito.verify(serviceUtilsMock).checkPurchaseParameters(Mockito.any(String.class), Mockito.any(long.class),
            Mockito.any(TypeHistoryEvent.class));
    Mockito.verify(userServiceMock).getUserInventoryCapacity(Mockito.any(User.class));
    Mockito.verify(userRepositoryMock).save(Mockito.any(User.class));

    // Check results
    assertNull(result.get(UserServiceImpl.ERRORS));
    User userResult = (User) result.get(UserServiceImpl.USER);
    assertEquals(new BigDecimal( val: 500), userResult.getCoin());
    assertEquals( expected: 1, userResult.getUserHistory().size());
    assertEquals( expected: 1, userResult.getItemsUser().size());
    assertEquals(item, user.getItemsUser().get(0).getItem());
    assertEquals(item.getName(), user.getUserHistory().get(0).getEventName());
}
```

Figura 157. Test unitario de buyItem sin errores.

```java
@RunWith(SpringRunner.class)
@SpringBootTest
public class ItemServiceTest {

    @InjectMocks
    ItemServiceImpl itemService;

    @Mock
    ServiceUtils serviceUtilsMock;

    @Mock
    UserServiceImpl userServiceMock;

    @Mock
    UserRepository userRepositoryMock;

    @Mock
    ExerciseServiceImpl exerciseServiceMock;


    private final static String TEST_TOKEN = "wejkfnewjkflewkfkewrfl";
    private final static long TEST_ITEM_ID = 3l;
    private final static String TEST_ERROR_MESSAGE = "Item doesn't exit";
    private User user;
    private Map<String, Object> parameters;

    @Before
    public void setUp() {
        user = new User( username: "testUser",  password: "123",  email: "test@gmail.com", new Authority());
        parameters = new HashMap<>();
    }
```

Figura 158. Declaración de los mocks seguida del método con anotación @Before.

Para este método, además de su correcta ejecución, se probó simulando un fallo a la hora de verificar el objeto comprado (ver Figura 159).

```java
@Test
public void test_buyItem_error() {
    user.setCoin(new BigDecimal( val: 1000));
    parameters.put(UserServiceImpl.USER, user);
    parameters.put(UserServiceImpl.ERRORS, TEST_ERROR_MESSAGE);

    //Mocking
    Mockito.when(serviceUtilsMock.checkPurchaseParameters(Mockito.any(String.class), Mockito.any(long.class),
            Mockito.any(TypeHistoryEvent.class))).thenReturn(parameters);

    // Call the method to test
    Map<String, Object> result = itemService.buyItem(TEST_TOKEN, TEST_ITEM_ID, TypeHistoryEvent.SPENDING_DAILY_ITEM);

    // Verify Mocking
    Mockito.verify(serviceUtilsMock).checkPurchaseParameters(Mockito.any(String.class), Mockito.any(long.class),
            Mockito.any(TypeHistoryEvent.class));

    // Check results
    assertNotNull(result.get(UserServiceImpl.ERRORS));
    assertEquals(TEST_ERROR_MESSAGE, result.get(UserServiceImpl.ERRORS));

    User userResult = (User) result.get(UserServiceImpl.USER);
    assertEquals(new BigDecimal( val: 1000), userResult.getCoin());
    assertEquals( expected: 0, userResult.getUserHistory().size());
    assertEquals( expected: 0, userResult.getItemsUser().size());
}
```

Figura 159. Test unitario de buyItem con errores.

9.2 Pruebas de integración

Las pruebas de integración permiten probar el correcto funcionamiento de los componentes desarrollados una vez ya dentro del sistema. Gracias a estas pruebas podemos verificar que los controladores, servicios y repositorios se comunican correctamente entre sí.

Una vez probado el método de comprar objetos de forma unitaria, se probó también una vez integrado con el sistema. Una vez comprado un objeto, este pasa directamente al inventario del usuario considerando, además, la posibilidad de ocupar espacio de inventario o incluso incrementar su capacidad máxima. Es por ello que se comprobó su correcto funcionamiento comprando una serie de objetos, cada uno de una categoría diferente.

Otras pruebas de integración realizadas fueron la activación de fotos de perfil y colores para el nombre de perfil. Estos cambios deberían verse reflejados en diferentes módulos de ASys ajenos a la tienda y al inventario.

Por último, para comprobar el correcto funcionamiento del sistema de recompensas, debido a que no se pueden realizar ejercicios, se crearon botones temporales en el menú de ejercicios que simulan la finalización de ejercicios de diferentes dificultades y con diferentes notas (ver Figura 160). Además, este mecanismo nos ha permitido probar también si los multiplicadores funcionan de manera correcta o no. Es importante destacar que estos botones

serán suprimidos cuando se solucionen los problemas relacionados con la corrección de ejercicios.

Figura 160. Botones temporales para probar el sistema de recompensas y los multiplicadores.

9.3 Pruebas de aceptación

En las pruebas de aceptación, el producto final es comprobado por el usuario en su propio entorno de explotación con el fin de determinar si este lo acepta tal como está o no.

Estas pruebas se repartieron en cuatro sesiones y se realizaron gracias a la participación de cuatro usuarios:

- **Usuario 1**: Josep Silva. Este usuario es el tutor del presente TFG y uno de los máximos responsables de ASys.
- **Usuario 2**: Un ingeniero informático graduado en la Universidad Carlos III de Madrid especializado en diseño web. Se considera **usuario externo.**
- **Usuario 3**: Carlos Viñals. Desarrollador de la plataforma.
- **Usuario 4**: Un profesor de la universidad experto en el dominio. Se considera **usuario externo**.
- **Usuario 5**: Un profesor de la universidad no experto en el dominio. Se considera **usuario externo**.

Todos los errores y sugerencias de mejoras detectadas durante esta fase de pruebas se recogieron en diversos informes que se encuentran como anexo a este documento, en el epígrafe: Anexos: Informes de pruebas. Todos los problemas recogidos se pudieron solucionar y el resultado final se plasmó en el capítulo Desarrollo de la solución con el fin de mostrar la

versión final de este proyecto en un único punto. Sin embargo, resulta de interés comentar ciertos problemas detectados y esto es lo que haremos a continuación.

9.3.1 Sesión 1

En esta sesión se detectaron principalmente fallos leves de interfaz como textos con erratas, no explicativos o problemas con iconos. Todos estos problemas se solucionaron sin dificultades. Sin embargo, los problemas que destacan son el 18, 19 y 20:

9.3.1.1 Problema 18: inconsistencia en la interfaz del inventario

En un comienzo, la interfaz principal del inventario lucía como en la Figura 161. Aquí surgía un problema de inconsistencia, puesto que la cantidad de los objetos de cada categoría aparecía dentro el botón, mientras que la cantidad de pistas aparecía en grande y sin botón.

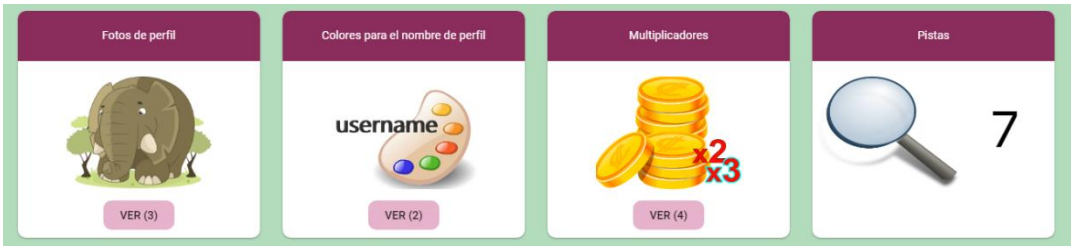


Figura 161. Versión antigua de la página principal del inventario.

Para solucionar este problema se añadió un elemento al lado del botón que mostrase la cantidad y se eliminó el número gigante en las pistas. El resultado puede verse en la Figura 135.

9.3.1.2 Problema 19: icono del candado de las pistas no se entiende

Durante la fase de diseño se diseñaron los candados de la Figura 162. Sin embargo, durante la fase de implementación se probó otra versión de este (ver Figura 163). Finalmente, debido a la problemática que causaron, se regresó al diseño original.

Figura 162. Versión original de los candados.

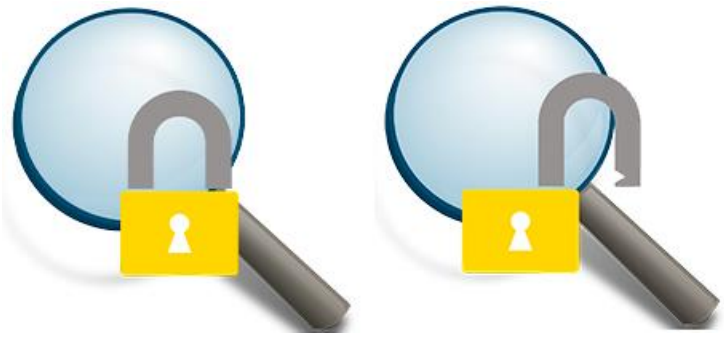


Figura 163. Versión de los candados que se probó durante la implementación.

9.3.1.3 Problema 20: logo de ASys no redirige al Home

Este problema se escapa un poco del ámbito del proyecto, pero debido a la importancia que tiene en una página web y a lo sencillo que es solucionarlo, se decidió solventar el problema.

9.3.2 Sesión 2

Durante esta sesión de pruebas se detectaron únicamente cuatro problemas, donde destacamos el número 22 y el número 23.

9.3.2.1 Problema 22: debería haber una sección de documentación que explicase cómo funciona la monetización

Puesto que existía una pantalla con documentación sobre los rankings, se pudo aprovechar su estructura y componentes para crear esta nueva página cuyo resultado puede verse en el subapartado del desarrollo de la solución Documentación de la monetización. Sería interesante, en un futuro, crear un apartado de documentación exclusivo que contenga información de todo ASys.

9.3.2.2 Problema 23: los diálogos no respetan el formato de la web

Para solucionar este problema se aprovecharon los componentes de diálogos ya existentes para crear los diálogos que podemos ver en el epígrafe Diálogos, situado en el capítulo dedicado al desarrollo de la solución.

9.3.3 Sesión 3

Durante esta sesión de pruebas se detectaron una serie de problemas bastante graves que comentaremos a continuación. También se sugiere una mejora bastante interesante.

9.3.3.1 Problema 29: la fecha-hora para determinar qué ítems diarios se muestran no es la del servidor, sino la del usuario

En un comienzo se utilizaba el constructor *new Date()* en el lado cliente pero nunca se llegó a considerar que el usuario podría estar situado en otro país o simplemente decidiese cambiar de fecha y hora su dispositivo. Este problema se solventó de manera sencilla: tal como se explica en el subapartado de Objetos diarios en el lado del servidor, cada vez que se entra a la tienda se solicita la hora al servidor gracias a una serie de métodos de utilidad situados en una nueva clase en la capa de servicios (ver Figura 164).

```
public Map<String, Long> getAllServerTimesInMillis() {
    Map<String, Long> times = new HashMap<>();

    times.put("timeServer", this.getServerTimeInMillis());
    times.put("todayMidnight", this.getServerTimeTodayMidnightInMillis());
    times.put("tomorrowMidnight", this.getServerTimeTomorrowMidnightInMillis());

    return times;
}

public Long getServerTimeInMillis()  {
    return Calendar.getInstance().getTimeInMillis();
}

private Calendar getServerTimeTodayMidnight() {
    Calendar midnight = Calendar.getInstance();
    midnight.set(Calendar.HOUR_OF_DAY, 0);
    midnight.set(Calendar.MINUTE, 0);
    midnight.set(Calendar.SECOND, 0);
    return midnight;
}

private Long getServerTimeTodayMidnightInMillis() {
    return this.getServerTimeTodayMidnight().getTimeInMillis();
}

private Long getServerTimeTomorrowMidnightInMillis() {
    Calendar tomorrowMidnight = this.getServerTimeTodayMidnight();
    tomorrowMidnight.add(Calendar.DATE,  amount: 1);
    return tomorrowMidnight.getTimeInMillis();
}
```

Figura 164. Métodos de utilidad para obtener la hora del servidor.

Además, este problema de la tienda permitió detectar que ocurría el mismo problema cuando un usuario utilizaba un multiplicador. En este caso, más que una ventaja, podría provocar un descontento del usuario, pues vería como pierde su multiplicador. Este problema se solventó de la misma forma.

9.3.3.2 Problema 31: añadir la opción de eliminar y/o vender objetos

Si dejamos que el usuario pueda eliminar y/o vender objetos previamente comprados, a lo mejor se compra fotos de perfil, los prueba unos días y luego los quita y se compra otras sin necesidad de ampliar el espacio de su inventario. Cuando la foto de perfil cueste menos que

aumentar el espacio, pues le rentará al usuario. Esto nos recordó que en el juego Fortnite posee una dinámica similar: cuando te compras una *skin* (o lo que sea estético) con la moneda de ellos, la cual compras con dinero real, tienes derecho a devolver la *skin* y, además, te devuelven el importe pagado por ella, para que te compres otra cosa. Esto lo puedes hacer un máximo de 3 veces. Esta dinámica nos resultó bastante interesante y fue la que se adoptó finalmente: un usuario puede devolver los objetos estéticos que compre hasta un máximo de tres veces. Estas tres veces, en suma, se le efectuará un reintegro del importe pagado.

9.3.3.3 Problema 34: el usuario puede comprar los ítems que quiera por el precio que quiera

Este problema ocurría porque no se contrastaba toda la información del servidor. Además, se enviaba en la petición POST el JSON del objeto. Para solucionar este problema se efectuaron ciertas modificaciones en el código, añadiendo una serie de comprobaciones en el lado del servidor antes de realizar la acción requerida por el usuario. Estas validaciones se indican en diferentes diagramas de flujo mostrados en los diferentes apartados del epígrafe Controladores y servicios.

9.4 Otras pruebas

Para complementar las diferentes pruebas realizadas para verificar y validar la solución desarrollada, se han utilizado una serie de herramientas con el fin de analizar y revisar la calidad del código y la accesibilidad de las pantallas.

9.4.1 Calidad del Código

Con el fin de reducir el esfuerzo de Mantenimiento de Software y sus costes futuros, es necesario mejorar la calidad del código existente y/o producir nuevo código de calidad. Para ello, hay que llevar a cabo un mantenimiento preventivo, un tipo de mantenimiento que mejora las propiedades del software sin alterar sus especificaciones funcionales [102]. Mejorar la calidad del código con el mantenimiento preventivo permite, además, poder facilitar los otros tipos de mantenimiento: correctivo, adaptativo y perfectivo[103]. En suma, según el estándar ISO-9126, una de sus características de su modelo de calidad es la Mantenibilidad donde una de sus subcaracterísticas es la facilidad de prueba [104]. Es decir, tener un código de calidad no solo facilita su entendimiento y modificación futura, sino que además permite realizar *testing* de una forma más sencilla, por lo que cuanto más mantenible sea, más fácil será realizar pruebas unitarias y de integración.

Para lograr este código de calidad, desde el comienzo del desarrollo de la solución se ha utilizado una herramienta propia del *IDE* de *IntelliJ IDEA* capaz de inspeccionar el código y

detectar anomalías en el mismo antes de compilar [105]. Gracias a esta herramienta es posible solucionar defectos del código en tiempo real, lo que permite ahorrar bastante tiempo ya que el problema se soluciona en el mismo momento en el que se codifica. Además, esta inspección puede lanzarse de forma manual para recoger en un único punto todos los errores y defectos encontrados.

Entre los diferentes aspectos de calidad, en nuestro proyecto se han tenido en cuenta las declaraciones e inicializaciones correctas de variables, una complejidad ciclomática y cognitiva reducida en los diferentes métodos, así como el evitar tener código duplicado. En la Figura 165 podemos ver un ejemplo de errores detectados durante una inspección de código.

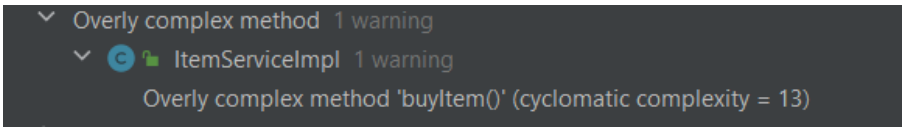


Figura 165. Errores detectados durante una inspección de código.

9.4.2 Accesibilidad web

La accesibilidad de las pantallas de la tienda y el inventario ha sido ha sido otro aspecto valorado en nuestro proyecto. La accesibilidad forma parte de lo que se denomina Usabilidad Universal y consiste en la capacidad de acceso que posee una web para que sus contenidos sean visualizados y utilizados por cualquier usuario, independientemente de sus discapacidades [106]. En [107] se argumenta, además, que una página web puede aportar enormes beneficios, pero solo si es accesible.

Para poder medir la accesibilidad web de las pantallas desarrolladas en este TFG, se ha utilizado la herramienta *Lighthouse* [108] . Para utilizarla hay que instalar su extensión en el navegador *Chrome* y ejecutarla en la página web que deseamos probar y, tras una breve espera, nos indicará con un número del 1 al 100 la puntuación obtenida en accesibilidad web, resultado de una auditoría. Si el puntaje obtenido se encuentra entre 90 y 100, se considerará un resultado positivo.

En nuestro caso, la primera ejecución obtenida rozaba el valor mínimo con un 87 de puntuación (ver Figura 166). Entre los fallos detectados, se encuentra la ausencia del atributo de texto alternativo, *alt*, en las diferentes imágenes mostradas (ver Figura 167).

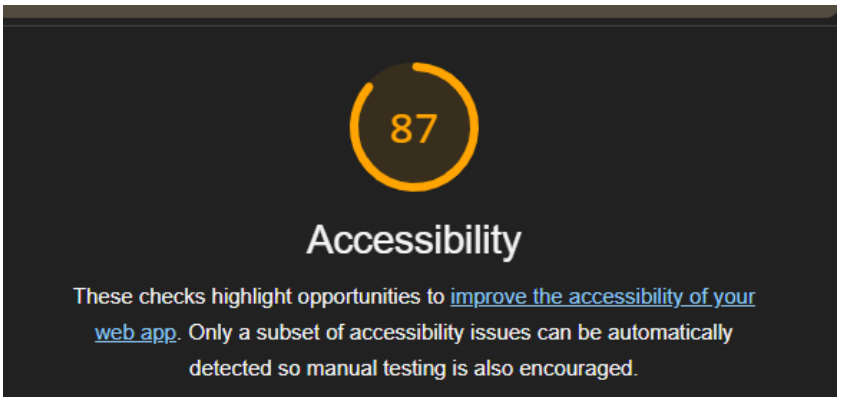


Figura 166. Puntuación de 87 obtenida tras ejecutar Lighthouse la primera vez.

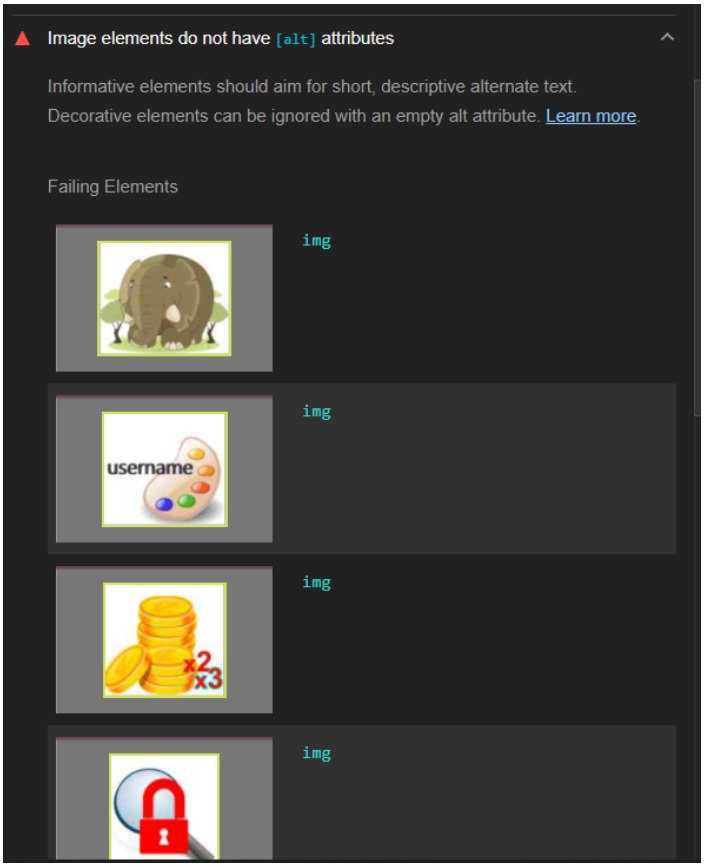


Figura 167. Indicador de ausencia del atributo de texto alternativo, alt, en las imágenes.

Una vez añadido un texto alternativo a estas imágenes (ver Figura 168), se volvió a ejecutar la herramienta, obteniendo una puntuación de 97 sobre 100 (ver Figura 169).

```
<template #logo>
    <img :src="logoURL" :alt="`${$t(`${item.name}`)} logo`">
</template>
```

Figura 168. Inserción del atributo alt en las imágenes.

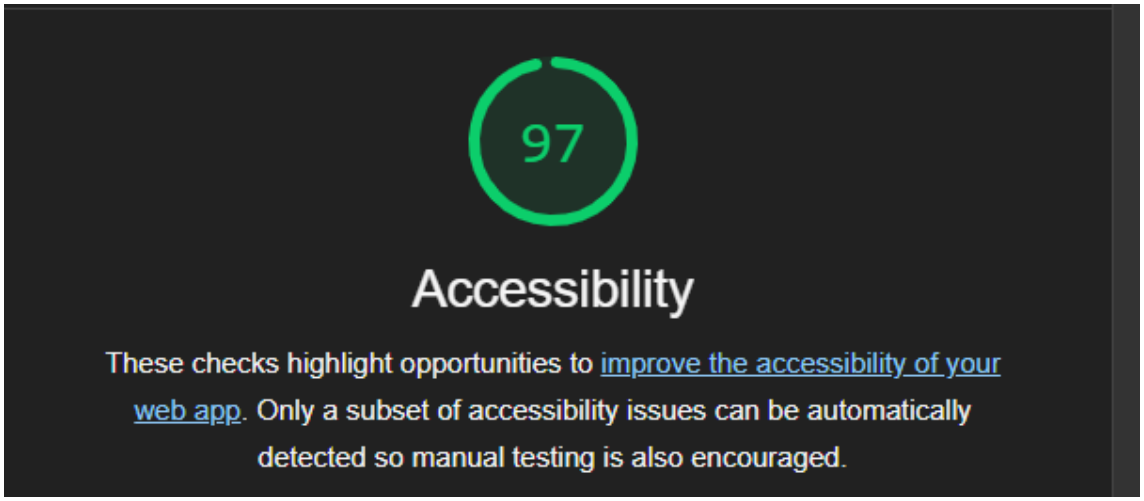


Figura 169. Puntuación de 97 obtenida tras ejecutar Lighthouse tras añadir el atributo alt.

10 Conclusiones

Al comienzo de este proyecto se plantearon una serie de objetivos a cumplir. Ahora que nos encontramos en el desenlace de este trabajo, es el momento de determinar si se han cumplido o no las diferentes metas propuestas.

Durante las primeras semanas del proyecto se procedió con la realización de un vasto análisis sobre el uso de mecanismos de gamificación en diferentes aplicaciones, muchas de ellas relacionadas con la temática de ASys y otras un poco más alejadas de esta, donde las recompensas y la monetización están más evolucionadas. Esta investigación ha permitido reunir **un gran número de ideas y propuestas** a realizar tanto en este proyecto como en trabajos futuros de ASys.

Una vez sintetizadas todas estas ideas y haber definido el ámbito del proyecto, se terminó desarrollado un **sistema de monetización** compuesto por dos **unidades monetarias**, Monedas y Láureas. Ambas se podrán conseguir realizando ejercicios de programación, cuya **recompensa** dependerá de la nota que se obtenga al realizarlos, el nivel del usuario y la dificultad del ejercicio.

Para poder utilizar estas unidades monetarias se desarrolló una nueva pantalla en la aplicación que sirviese como una **tienda de objetos**. Estos objetos son elementos canjeables por el usuario y pueden ser de dos tipos**: cosméticos o de utilidad**. Los primeros son objetos puramente estéticos, como fotos de perfil o colores para que el nombre del perfil del usuario sea de un color diferente al negro. Los objetos de utilidad, en cambio, poseen cierta función en la aplicación. Estos objetos pueden ser pistas para utilizar en ejercicios, multiplicadores de Monedas para incentivar la realización de ejercicios durante un periodo de tiempo continuado o, por otro lado, también existen objetos que otorgan cierto progreso en el juego, como un objeto que desbloquea la funcionalidad de las pistas, inicialmente bloqueada, o un objeto que incrementa la capacidad máxima del inventario del usuario. El **inventario** es la segunda pantalla que se ha implementado en ASys y su función principal es, por una parte, mostrar los diferentes objetos que posee el usuario y, por otra parte, activar el que desee.

Por otro lado, puesto que la recompensa de los ejercicios es variable, en base a factores como la dificultad del ejercicio y el nivel del usuario, los precios de los objetos se establecieron teniendo en cuenta el nivel del usuario, con el objetivo de que **exista un equilibrio** con la economía, independientemente del nivel y experiencia que tenga el usuario. De esta manera, en ningún momento resultará muy sencillo o complicado comprar objetos en la tienda.

En conclusión, los objetivos propuestos se han conseguido con creces. Pero antes de poner fin a este laborioso y enriquecedor trabajo, en el que se han aprendido diversas tecnologías punteras como Vue y Spring, pasaremos a visualizar el esfuerzo y presupuesto reales que se necesitaron para llevar a cabo este proyecto, seguido de una propuesta de trabajos futuros a realizar.

10.1 Esfuerzo y coste real

En el epígrafe Estimación de Esfuerzo y Coste, tal como indica su nombre, se llevó a cabo una estimación del esfuerzo y del coste del proyecto. Ahora que se ha terminado el proyecto, procederemos a comparar las estimaciones realizadas frente a los valores reales.

En cuanto al tiempo dedicado, este ha sido de 7 meses. Los primeros dos se destinaron una formación de diferentes tecnologías, como Vue, Spring, HTML y CSS. Posteriormente, se destinaron cinco meses al diseño, solución y pruebas del proyecto. Durante cuatro de estos cinco meses se trabajó también durante los fines de semana. Esto implica que ha habido 32 días adicionales en los que el programador ha sido pagado un 75% más. Por lo tanto, las horas de diseño, pruebas y desarrollo se quedan en:

$$5\ meses * \frac{40 horas}{semana} * \frac{4 semanas}{1 mes} + \ 4\ meses * \frac{16\ horas}{semana} * \frac{4\ semanas}{1 mes}$$

$$= \ 800\ horas + 256\ horas = \mathbf{1056\ horas}$$

Lo que equivale a:

$$\frac{1056\ horas}{160 \frac{horas}{mes}} = \mathbf{6,6\ meses}$$

Por lo tanto, el tiempo real empleado, sin tener en cuenta la formación es de **6,6 meses** frente a los 6,43 meses estimados al comienzo del proyecto.

Por otro lado, para medir el coste real, hemos elaborado una tabla que refleja todos los costes del proyecto. En la Tabla 26 podemos ver que el coste real del proyecto ha sido de **21.766,67€,** frente a los 23.528€ estimados a inicios del proyecto.

Tabla 26. Coste real de proyecto.

Recurso	Coste mensual (€)	Coste fijo (€)
Programador jornada completa en formación (2 meses)	1900	-
Programador jornada completa (5 meses)	1.900	-
Programador jornada completa fines de semana (32 días)	-	3.546,67
Pruebas. Usuario 1 (3 sesiones)	-	180
Pruebas. Usuario 2	-	60
Pruebas. Usuario 3	-	60
Pruebas. Usuario 4	-	60
Pruebas. Usuario 5.		60
Licencias (Office 365, IDE IntelliJ)	-	700
Cursos de Spring y Vue		200
Libros		500
Portátil (Rayzen 7, 16GB, 512GB)	-	1.200
Monitor externo y otros periféricos	-	500
Otros gastos (suministros, materiales...)	200	-
Total 7 meses	14.700	7.066,67
	21.766,67	

10.2 Trabajos futuros

Si bien la temática principal de este TFG es implementar un sistema de monetización en ASys, otro punto importante de este trabajo se trató al comienzo de este, con la investigación de diferentes ideas y propuestas a realizar para complementar la monetización. Parte de estas posibles nuevas funcionalidades se recogen en la Especificación de Requisitos Software, en el subapartado de Requisitos futuros. Aquí podemos destacar, por un lado, el desarrollo de un **sistema de niveles** que refleje los conocimientos del usuario, y un **sistema de insignias** que se obtengan al cumplir ciertos hitos. Estas insignias serían visibles para el resto de los usuarios. Ambas propuestas pueden unirse sinérgicamente al sistema de monetización actual, otorgando Monedas y Láureas al alcanzar ciertos niveles o al conseguir determinadas insignias. Algunas **insignias**, además, se podrían incorporar a la **tienda** a precios elevados, para aquellos usuarios que posean mucho dinero por haber realizado múltiples ejercicios y sean aficionados al coleccionismo.

Otra propuesta a realizar en un futuro consiste en el desarrollo de una **pantalla** dedicada al **histórico** de compras, obtención de recompensas y utilización de objetos. El trabajo para llevar a cabo esta idea ya está comenzado, pues las tablas de la base de datos que recogen los datos necesarios a mostrar se han creado en el presente TFG. Esta vista resulta bastante interesante sobre todo para los administradores, quienes podrán hacer un seguimiento de las Monedas y Láureas obtenidas y gastadas por el resto de los usuarios y detectar posibles anomalías. Además, con esta vista se podrá determinar si es necesario realizar un ajuste de balance de precios y recompensas.

Finalmente, otro punto interesante a tratar sería la incorporación de este sistema de monetización a los ***rankings*** de la aplicación, los cuales han sufrido una serie de modificaciones positivas paralelamente a este trabajo [59]. Para ello, se podrían dar recompensas monetarias a los usuarios que queden en los primeros puestos diarios, semanales y mensuales de las diferentes tablas de clasificación que ofrece ASys.

Como se puede observar, existen múltiples mecanismos de gamificación que se pueden incorporar a ASys para mejorar la experiencia de usuario, incentivar el uso de la aplicación y motivar a los estudiantes. Todos estos mecanismos pueden crear una bonita sinergia con el sistema de monetización implementado en este trabajo, creando así, un sistema de gamificación robusto y unificado.

11 Bibliografía

[1] S. Martínez, “Metodología iterativa o incremental en la gestión de proyectos,” Dec. 14, 2014. https://www.mundoerp.com/blog/metodologia-iterativa-o-incremental-gestion-proyectos/ (accessed Mar. 30, 2021).

[2] Codewars Developers, “Codewars Docs | Introduction,” Dec. 27, 2020. https://github.com/codewars/docs/blob/master/content/training/getting-started/index.md (accessed Mar. 03, 2021).

[3] Codewars Developers, “Codewars Docs | Kata,” Jan. 14, 2021. https://github.com/codewars/docs/blob/master/content/concepts/kata/index.md (accessed Mar. 03, 2021).

[4] Codewars Developers, “Codewars Docs | Reward and Progress,” Nov. 17, 2020. https://github.com/codewars/docs/blob/master/content/concepts/gamification/index.md (accessed Mar. 03, 2021).

[5] Codewars Developers, “Codewars Docs | Ranks content,” Nov. 17, 2020. https://github.com/codewars/docs/blob/master/content/concepts/gamification/ranks.md (accessed Mar. 03, 2021).

[6] Codewars Developers, “Codewars Docs | Kata beta process,” Jan. 02, 2021. https://github.com/codewars/docs/blob/master/content/concepts/kata/beta-process.md (accessed Mar. 03, 2021).

[7] “Codewars,” *About*. https://www.codewars.com/about (accessed Mar. 03, 2021).

[8] Codewars Developers, “Codewars Docs | Ranks references,” Aug. 08, 2020. https://github.com/codewars/docs/blob/master/content/references/gamification/ranks.md (accessed Mar. 03, 2021).

[9] “Codewars,” *Leaderboard*. https://www.codewars.com/users/leaderboard (accessed Mar. 03, 2021).

[10] Codewars Developers, “Codewars Docs | Honor Rewards,” Oct. 29, 2020. https://github.com/codewars/docs/blob/master/content/references/gamification/honor.md (accessed Mar. 04, 2021).

[11] Codewars Developers, “Codewars Docs | Honor,” Nov. 17, 2020. https://github.com/codewars/docs/blob/master/content/concepts/gamification/honor.md (accessed Mar. 03, 2021).

[12] Codewars Developers, “Codewars Docs | Privileges,” Aug. 08, 2020. https://github.com/codewars/docs/blob/master/content/references/gamification/privileges.md (accessed Mar. 03, 2021).

[13] “CodeSignal,” *Coding Tests and Assessments for Technical Hiring*. https://codesignal.com/ (accessed Mar. 04, 2021).

[14] "CodeSignal," *Portal*. https://app.codesignal.com/ (accessed Mar. 04, 2021).

[15] "CodeSignal," *Code Arcade*. https://app.codesignal.com/arcade (accessed Mar. 04, 2021).

[16] "CodeSignal," *Code Arcade | Intro*. https://app.codesignal.com/arcade/intro (accessed Mar. 04, 2021).

[17] "CodeSignal," *Code Arcade | Intro | CheckPalindrome*. https://app.codesignal.com/arcade/intro/level-1/s5PbmwxfECC52PWyQ (accessed Mar. 04, 2021).

[18] "CodeSignal," *Company Challenges*. https://app.codesignal.com/company-challenges (accessed Mar. 04, 2021).

[19] "CodeSignal," *Company Challenges | Dropbox*. https://app.codesignal.com/company-challenges/dropbox (accessed Mar. 04, 2021).

[20] "CodeSignal," *Interview Practice | Study plan*. https://app.codesignal.com/interview-practice/settings (accessed Mar. 04, 2021).

[21] "CodeSignal," *Interview Practice | Extra Credit Plan*. https://app.codesignal.com/interview-practice (accessed Mar. 04, 2021).

[22] "CodeSignal," *Interview Practice | Extra Credit Plan | Arrays*. https://app.codesignal.com/interview-practice/topics/arrays (accessed Mar. 04, 2021).

[23] "DataCamp," *About*. https://www.datacamp.com/about (accessed Mar. 05, 2021).

[24] B. Scanlan, "DataCamp," *Support | DataCamp's XP and Progress: An Overview*, Nov. 2020. https://support.datacamp.com/hc/en-us/articles/360042332634-DataCamp-s-XP-and-Progress-An-Overview (accessed Mar. 05, 2021).

[25] "DataCamp," *Chapter 1 python basics | Intro to python for data science | When to use Python?* https://campus.datacamp.com/courses/intro-to-python-for-data-science/chapter-1-python-basics?ex=3 (accessed Mar. 05, 2021).

[26] "DataCamp," *Chapter 1 python basics | Intro to python for data science | Any comments?* https://campus.datacamp.com/courses/intro-to-python-for-data-science/chapter-1-python-basics?ex=4 (accessed Mar. 05, 2021).

[27] "DataCamp," *Projects*. https://datacamp.com/projects (accessed Mar. 05, 2021).

[28] R. Cabral, "DataCamp," *Support | How To Reset Course Progress*, Nov. 2020. https://support.datacamp.com/hc/en-us/articles/360001290713 (accessed Mar. 05, 2021).

[29] "DataCamp," *Practice*. https://learn.datacamp.com/practice (accessed Mar. 05, 2021).

[30] Sylar, “DataCamp,” *Support | DataCamp’s Daily Streaks*, Dec. 2020. https://support.datacamp.com/hc/en-us/articles/360014799594 (accessed Mar. 05, 2021).

[31] “Aplicaciones en Google Play,” *Brawl Stars*. https://play.google.com/store/apps/details?id=com.supercell.brawlstars&hl=es&gl=US (accessed Mar. 05, 2021).

[32] “Supercell Support Portal,” *Brawl Pass & Quests*. https://help.supercellsupport.com/brawl-stars/en/articles/brawl-pass-quests-1.html (accessed Mar. 05, 2021).

[33] “Supercell Support Portal,” *Tokens*. https://help.supercellsupport.com/brawl-stars/en/articles/tokens.html (accessed Mar. 05, 2021).

[34] “Supercell Support Portal,” *Gems*. https://help.supercellsupport.com/brawl-stars/en/articles/gems.html (accessed Mar. 05, 2021).

[35] “Supercell Support Portal,” *Coins*. https://help.supercellsupport.com/brawl-stars/en/articles/coins.html (accessed Mar. 05, 2021).

[36] “Supercell Support Portal,” *Star Points*. https://help.supercellsupport.com/brawl-stars/en/articles/star-points.html (accessed Mar. 05, 2021).

[37] “Supercell Support Portal,” *Power Play*. https://help.supercellsupport.com/brawl-stars/en/articles/power-play.html (accessed Mar. 05, 2021).

[38] “Supercell Support Portal,” *Championship Challenge*. https://help.supercellsupport.com/brawl-stars/en/articles/championship-challenge.html (accessed Mar. 06, 2021).

[39] Seekrtech, “Aplicaciones en Google Play,” *Forest ：Mantente Enfocado*. https://play.google.com/store/apps/details?id=cc.forestapp&referrer=utm_source%3Dofficalwebsite%26utm_medium%3Dbutton (accessed Mar. 06, 2021).

[40] D. Zheng, “Forest FAQ 常見問題 - Is there a limit of real trees I can plant? How many can I plant?,” Feb. 2021. http://help.forestapp.cc/en/articles/2209976-is-there-a-limit-of-real-trees-i-can-plant-how-many-can-i-plant (accessed Mar. 06, 2021).

[41] D. Zheng, “Forest FAQ 常見問題 - Set your planting time,” Feb. 2021. http://help.forestapp.cc/en/articles/2203604-set-your-planting-time (accessed Mar. 06, 2021).

[42] D. Zheng, “Forest FAQ 常見問題 - Coin reward formula,” Feb. 2021. http://help.forestapp.cc/en/articles/2328680-coin-reward-formula (accessed Mar. 06, 2021).

[43] J. Vargas-Enríquez, L. García-Mundo, M. Genero, and M. Piattini, “Análisis de uso de la gamificación en la enseñanza de la informática,” in *Actas de las XXI*

Jornadas de la Enseñanza Universitaria de la Informática, 2015, pp. 105–112. Accessed: Mar. 11, 2021. [Online]. Available: http://hdl.handle.net/2117/76784

[44] A. Iosup and D. Epema, "An experience report on using gamification in technical higher education," in *SIGCSE 2014 - Proceedings of the 45th ACM Technical Symposium on Computer Science Education*, Mar. 2014, pp. 27–32. doi: 10.1145/2538862.2538899.

[45] D. Parente, "Gamificación en la educación," *Gamificación en aulas universitarias*, vol. 11, 2016.

[46] J. Hamari, J. Koivisto, and H. Sarsa, "Does Gamification Work? -- A Literature Review of Empirical Studies on Gamification," in *2014 47th Hawaii International Conference on System Sciences*, 2014, pp. 3025–3034. doi: 10.1109/HICSS.2014.377.

[47] L. Hakulinen, T. Auvinen, and A. Korhonen, "Empirical Study on the Effect of Achievement Badges in TRAKLA2 Online Learning Environment," in *2013 Learning and Teaching in Computing and Engineering*, 2013, pp. 47–54. doi: 10.1109/LaTiCE.2013.34.

[48] D. Maletz, "Four Tricks to Improve Game Balance," Sep. 13, 2012. https://www.gamasutra.com/blogs/DavidMaletz/20120913/177683/Four_Tricks_to_Improve_Game_ (accessed Mar. 11, 2021).

[49] "IEEE Recommended Practice for Software Requirements Specifications," *IEEE Std 830-1998*, pp. 1–40, Oct. 1998, doi: 10.1109/IEEESTD.1998.88286.

[50] J. F. Kurose and K. W. Ross, *Redes de computadoras : un enfoque descendente*, 5ª ed. Madrid: Prentice Hall, 2010.

[51] A. Maya Gomis, "Diseño y Desarrollo del sistema ASys con Spring y Vue.js," Valencia, 2019.

[52] J. Nielsen, "10 Usability Heuristics for User Interface Design," Apr. 24, 1994. https://www.nngroup.com/articles/ten-usability-heuristics/#poster (accessed Jul. 18, 2021).

[53] MDN Contributors, "NodeList.prototype.forEach() - Web APIs | MDN," Jun. 15, 2021. https://developer.mozilla.org/en-US/docs/Web/API/NodeList/forEach#browser_compatibility (accessed Jul. 18, 2021).

[54] A. Rady, "Using forEach for IE 11," Nov. 04, 2019. https://rimdev.io/foreach-for-ie-11/ (accessed Jul. 18, 2021).

[55] F. Paredes Borras, "Diseño e implementación de un sistema online de gestión de grupos de programadores," Valencia, 2020.

[56] About Function Point Analysis, "IFPUG." https://www.ifpug.org/about-function-point-analysis/ (accessed Apr. 27, 2021).

[57] “Cosmic Software Sizing,” *The Benefits of COSMIC sizing*. https://cosmic-sizing.org/cosmic-sizing/intro/benefits-of-cosmic/ (accessed Apr. 27, 2021).

[58] “Cosmic Software Sizing,” *Measurement Process*. https://cosmic-sizing.org/cosmic-sizing/intro/measurement-process/ (accessed Apr. 27, 2021).

[59] C. Viñals Guitart, “Diseño y Desarrollo de un Sistema de Ranking para ASys,” Valencia, 2021.

[60] “Vue.js,” *Comparación con otros frameworks*. https://es.vuejs.org/v2/guide/comparison.html#Flexibilidad-y-modularidad (accessed Jul. 25, 2021).

[61] K. Lawson, “Bloomreach,” *What is a Single Page Application*, Jul. 2018. https://www.bloomreach.com/en/blog/2018/07/what-is-a-single-page-application.html#whatssingle-page-application (accessed Jul. 25, 2021).

[62] “Vue.js,” *Conceptos Básicos de Componentes*. https://es.vuejs.org/v2/guide/components.html#Organizacion-de-Componentes (accessed Jul. 25, 2021).

[63] “Vue.js,” *Componentes de un Solo Archivo (Single File Components)*. https://es.vuejs.org/v2/guide/single-file-components.html (accessed Jul. 25, 2021).

[64] “Vuex,” *What is Vuex?* https://vuex.vuejs.org/#what-is-a-state-management-pattern (accessed Jul. 25, 2021).

[65] “HTML: Lenguaje de etiquetas de hipertexto,” *MDN contriubutors*, Jul. 21, 2021. https://developer.mozilla.org/es/docs/Web/HTML (accessed Jul. 25, 2021).

[66] Y. Fernández, “Qué es el HTML5 y qué novedades ofrece,” Jan. 13, 2021. https://www.xataka.com/basics/que-html5-que-novedades-ofrece (accessed Jul. 25, 2021).

[67] MDN Contributors, “CSS,” Jul. 24, 2021. https://developer.mozilla.org/es/docs/Web/CSS (accessed Jul. 25, 2021).

[68] C. Goyier, “A Complete Guide to Flexbox | CSS-Tricks,” Spring 08, 2013. https://css-tricks.com/snippets/css/a-guide-to-flexbox/ (accessed Jul. 25, 2021).

[69] C. Walls and R. Breidenbach, *Spring in Action*. New York: Manning Publications Co, 2005.

[70] C. Walls, *Spring in Action, Fifth Edition [electronic resource]* , 5th edition. Manning Publications, 2018.

[71] V. Mihalcea *et al.*, “Hibernate ORM 5.4.32.Final User Guide.” https://docs.jboss.org/hibernate/orm/5.4/userguide/html_single/Hibernate_User_Guide.htm (accessed Jul. 31, 2021).

[72] O. Gierke, T. Darimont, C. Strobl, M. Paluch, and J. Bryant, "Spring Data JPA," *Reference Documentation*, Jul. 16, 2021. https://docs.spring.io/spring-data/jpa/docs/current/reference/html/#jpa.query-methods (accessed Jul. 31, 2021).

[73] S. Chacon and B. Straub, "Getting Started - What is Git?," in *Pro Git book*, 2nd Edition., Apress, 2014. Accessed: Jul. 31, 2021. [Online]. Available: https://git-scm.com/book/en/v2/Getting-Started-What-is-Git%3F

[74] "Overview of Spring MVC Architecture," *TERASOLUNA Server Framework for Java (5.x) Development Guideline*, Mar. 17, 2017. https://terasolunaorg.github.io/guideline/5.2.1.RELEASE/en/Overview/SpringMVCOverview.html (accessed May 01, 2021).

[75] E. Gottesdiener, *The Software Requirements Memory Jogger: A Pocket Guide to Help Software and Business Teams Develop and Manage Requirements*, First Edition. Salem: GOAL/QPC, 2005.

[76] G. Averbuj, "Economía en Juegos," Jun. 23, 2014. https://es.slideshare.net/starker/economa-en-juegos?from_action=save (accessed Jun. 24, 2021).

[77] J. Beaird, *The principles of beautiful web design*. Victoria: SitePoint, 2007.

[78] ObjectDB Software Developers, "ObjectDB Software," *Auto Generated Values*. https://www.objectdb.com/java/jpa/entity/generated (accessed Apr. 24, 2021).

[79] O. Gierke, T. Darimont, C. Strobl, M. Paluch, and J. Bryant, "Spring Data JPA," *Reference Documentation | Query Methods*. https://docs.spring.io/spring-data/jpa/docs/current/reference/html/#jpa.query-methods (accessed Apr. 24, 2021).

[80] V. Mihalcea *et al.*, "Hibernate ORM 5.4.32.Final User Guide | ManyToMany." https://docs.jboss.org/hibernate/orm/5.4/userguide/html_single/Hibernate_User_Guide.html#associations-many-to-many (accessed Apr. 15, 2021).

[81] V. Mihalcea *et al.*, "Hibernate ORM 5.4.32.Final User Guide | OneToMany." https://docs.jboss.org/hibernate/orm/5.4/userguide/html_single/Hibernate_User_Guide.html#associations-one-to-many (accessed Apr. 15, 2021).

[82] V. Mihalcea *et al.*, "Hibernate ORM 5.4.32.Final User Guide | OneToOne." https://docs.jboss.org/hibernate/orm/5.4/userguide/html_single/Hibernate_User_Guide.html#associations-one-to-one (accessed Apr. 15, 2021).

[83] E. Paraschiv, "Jackson - Bidirectional Relationships | Baeldung," Mar. 01, 2021. https://www.baeldung.com/jackson-bidirectional-relationships-and-infinite-recursion (accessed Apr. 25, 2021).

[84] LogicBig Developers, "Jackson JSON - Using @JsonManagedReference and @JsonBackReference for circular references," Aug. 11, 2020. https://www.logicbig.com/tutorials/misc/jackson/json-managed-reference.html (accessed Apr. 25, 2021).

[85] W. Fitzgerald, “Presentation Jackson 2.0 · FasterXML/jackson-docs Wiki · GitHub,” Aug. 09, 2012. https://github.com/FasterXML/jackson-docs/wiki/Presentation-Jackson-2.0#new-feature-full-handling-of-object-graphs (accessed Apr. 25, 2021).

[86] D. Molinari, “Jackson JSON: difference between @JsonIgnore and @JsonIgnoreProperties annotations | Dede Blog,” Mar. 10, 2015. http://www.davismol.net/2015/03/10/jackson-json-difference-between-jsonignore-and-jsonignoreproperties-annotations/ (accessed Apr. 25, 2021).

[87] D. Molinari, “Jackson: using @JsonIgnore and @JsonProperty annotations to exclude a property only from JSON deserialization | Dede Blog,” Mar. 21, 2015. http://www.davismol.net/2015/03/21/jackson-using-jsonignore-and-jsonproperty-annotations-to-exclude-a-property-only-from-json-deserialization/ (accessed Apr. 25, 2021).

[88] S. McConnell, “Code complete.” Microsoft Press, Redmond, Wash, 2004.

[89] R. C. Martin, “Clean code : refactoring, patterns, Testen und Techniken fur sauberen code.” MITP, Heidelberg, [Germany, 2009.

[90] “Vuex,” *Strict Mode*. https://vuex.vuejs.org/guide/strict.html#development-vs-production (accessed Apr. 25, 2021).

[91] D. Mark Clements, “Really Fast Deep Clone,” Mar. 14, 2021. https://github.com/davidmarkclements/rfdc (accessed May 15, 2021).

[92] A. Gore, “Vue.js Developers,” *Extending Vue.js Components*, Jun. 11, 2020. https://vuejsdevelopers.com/2017/06/11/vue-js-extending-components/ (accessed Apr. 08, 2021).

[93] “Vue.js,” *Slots*. https://vuejs.org/v2/guide/components-slots.html (accessed Apr. 08, 2021).

[94] A. Gore, “Vue.js Developers,” *Extending Vue Component Templates*, Feb. 24, 2020. https://vuejsdevelopers.com/2020/02/24/extending-vuejs-components-templates/ (accessed Apr. 08, 2021).

[95] “Vue.js,” *Style Guide*. https://vuejs.org/v2/style-guide/#Tightly-coupled-component-names-strongly-recommended (accessed Apr. 10, 2021).

[96] B. Fox, “Async computed properties for Vue.js,” Oct. 15, 2020. https://github.com/foxbenjaminfox/vue-async-computed (accessed May 19, 2021).

[97] W. Hilton, E. You, and H. Darkholme, “GitHub,” *asyncComputed · Issue #4083 · vuejs/vue · *, 2016. https://github.com/vuejs/vue/issues/4083 (accessed May 25, 2021).

[98] Mixins, “Vue.js.” https://es.vuejs.org/v2/guide/mixins.html (accessed Jun. 26, 2021).

[99] S. Makwana, "javascript - Make localStorage Data read-Only - Stack Overflow," Oct. 15, 2018. https://stackoverflow.com/questions/52820693/make-localstorage-data-read-only (accessed Mar. 26, 2021).

[100] MDN Contributors, "HTTP | Control de acceso HTTP (CORS)." https://developer.mozilla.org/es/docs/Web/HTTP/CORS (accessed Aug. 18, 2021).

[101] A. Silgado, "¿Qué és el SMTP? Ventajas e inconvenientes de un servidor SMTP," Apr. 25, 2017. https://blog.mailrelay.com/es/2017/04/25/que-es-el-smtp (accessed Aug. 02, 2021).

[102] F. Ruiz González *et al.*, *Mantenimiento del software : conceptos, métodos, herramientas y outsourcing*. Madrid: Ra-ma, 1998.

[103] J. M. Torralba Martínez, "El servicio de mantenimiento de los programas informáticos," in *La gestión de la diversidad: XIII Congreso Nacional, IX Congreso Hispano-Francés*, Jul. 1999, pp. 379–382.

[104] A. Abran and R. Al-Qutaish, "ISO 9126: Analysis of Quality Models and Measures," 2010, pp. 205–228. doi: 10.1002/9780470606834.ch10.

[105] "JetBrains," *IntelliJ IDEA - Code inspections*, Jun. 21, 2021. https://www.jetbrains.com/help/idea/code-inspection.html (accessed Jul. 27, 2021).

[106] B. Shneiderman, *Designing the user interface : strategies for effective human-computer interaction*, 5th ed. Boston [etc: Addison-Wesley, 2010.

[107] D. Stone, C. Jarrett, M. Woodroffe, and S. Minocha, *User interface design and evaluation*. San Francisco: Morgan Kaufmann, 2005.

[108] Google Developers, "Tools for Web Developers | Lighthouse," Apr. 13, 2021. https://developers.google.com/web/tools/lighthouse?hl=es (accessed Jul. 27, 2021).

12 Anexos: Informes de pruebas

12.1 Sesión 1

12.1.1 Informe 1

Informe Pruebas de Aceptación

Sesión	1 de 4	Usuario	Usuario 1: Josep Silva
Fecha	19/06/2021	Externo	No

Observaciones

Tanto el cliente como el servidor API se ejecutan en la misma máquina (portátil personal del desarrollador). Estas pruebas se hicieron en el navegador Chrome en un dispositivo móvil. Y una segunda pasada en Chrome en un Macbook.

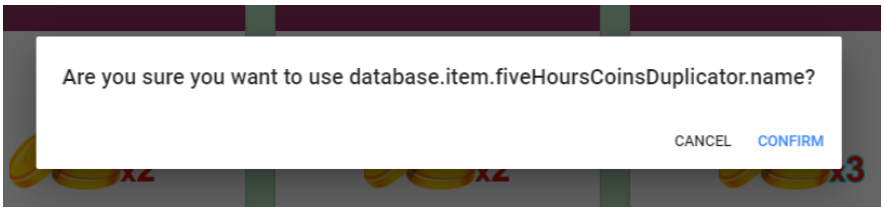
Resultados

Problema 1
Tipo: Mejora
Comentario:
"Falta la barra de migas para poder volver en todas las pantallas del inventario y la tienda"

Problema 2
Tipo: Error / Mejora
Comentario:
"El icono de la cabecera de Inventory no se dibuja al cargar la página como pasa con los demás."

Problema 3
Tipo: Error Responsive
Comentario:
"Click more info" debería ser "Click for more info", pero en la versión movil no cabe, así que debería ser "More Info". En la tienda has puesto INFO. Así que yo dejaría INFO aquí también."
Foto:

Problema 4
Tipo: Bug (Error)
Comentario:
"Al pinchar en "Use" de un multiplicador dice:
"Are you sure you want to use dababase.item.threehours...""
Foto:



Problema 5
Tipo: Error *Responsive*
Comentario:
"El icono de Multipliers es una X. Al meterlo en el círculo parece un botón de cerrar. Hay que cambiar ese icono por otro."
Foto:

Problema 6
Tipo: Mejora
Comentario:
"En castellano, no tiene sentido que las monedas se llamen "coins". Propongo cambiar la palabra "coins" por "oro" en castellano. Eso significa que el mensaje:
"Coins insuficientes" debería ser "Oro insuficiente"
"850 coins" debería ser "850 oro"
etc."

Problema 7
Tipo: Mejora
Comentario:
"En More info Inventory propongo poner el siguiente texto:

"Here you can see the avatars, colors for the username, Coins multipliers and the number of clues you have bought in the Shop. The maximum capacity of the inventory can be expanded by purchasing items of type "+X space inventory". You currently have 17 items and a maximum capacity of 20.""

Problema 8
Tipo: Bug (Error)
Comentario:
"En INFO color hay algunos errores gramaticales. El texto corregido es:

"If you activate this color, your username will be displayed with this color in multiple places of the app. Once purchased, it can be viewed and activated in the inventory.""

Problema 9
Tipo: Bug (Error)
Comentario:
"En X hours Coins duplicator and triplicator hay algunos errores gramaticales. El texto corregido es:

"It doubles the Coins obtained during a certain period of time. Once purchased, it can be viewed and activated in the inventory."
(o it triples en el triplicator)"

Problema 10
Tipo: Bug (Error)
Comentario:
"En INFO en muchos sitios cambiar
"it can be viewed and activated in inventory" por:
"it can be viewed and activated in the inventory""

Problema 11
Tipo: Bug (Error)
Comentario:
"En INFO de objetos diarios:
"En esta sección aparecerán tres nuevos objetos para comprar cada día a las 00:00. Los objetos diarios cuestan un 25% menos que los objetos permanentes. Los objetos estéticos, como las fotos de perfil o colores para el nombre de perfil, solo se podrán comprar en esta sección."

(he quitado "en suma", puesto que lo que le seguía no era una consecuencia de lo anterior)"

Problema 12
Tipo: Bug (Error)
Comentario:
"Info Poder pedir pistas en ejercicio pone "contendido" en lugar de "contenido""

Problema 13
Tipo: Bug (Error)
Comentario:
"Info +X espacio de inventario (le ocurre a los tres items)
"Su precio incrementa" -> "Su precio se incrementa""

Problema 14
Tipo: Bug (Error)
Comentario:
"En Info Duplicador de Coins de 3 horas (y de 5 horas):

"Duplica los Coins conseguidos durante un determinado periodo de tiempo. Una vez comprado, se podrá visualizar y activar en el inventario."

(ponía "periodo tiempo")"

Problema 15
Tipo: Mejora
Comentario:
"El item triplicador de coins debería estar junto al duplicador, y no después de los incrementadores de espacio del inventario."

Problema 16
Tipo: Mejora
Comentario:
"Info X clue (está en todos los items de clues)

"Provide clues in exercises. It is necessary to have the item "Being able to ask for clues in exercises" to be able to use them."
->
"This item contains X clues that you can use when you are solving an exercise. When you use a clue, the system will give you advice about how to solve an exercise. To buy or use

clues, you need first to purchase the item "Unlock asking for clues in exercises".”

12.1.2 Informe 2

Informe Pruebas de Aceptación

Sesión	1 de 4	Usuario	Usuario 2
Fecha	19/06/2021	Externo	Sí

Observaciones

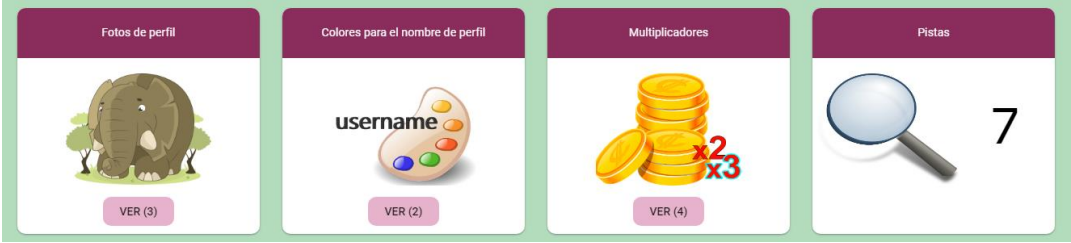
Tanto el cliente como el servidor API se ejecutan en la misma máquina (portátil personal del desarrollador). Se distribuyeron credenciales de acceso al usuario y la dirección de acceso.

Resultados

Problema 17
Tipo: Mejora
Comentario:
“Botón para ir hacia atrás en el inventario. Una barra de migas sirve.”

Problema 18
Tipo: Mejora
Comentario:
“1. En los Avatars, Colors, Multipliers pone su cantidad al lado de show, pero con las pistas, lo estoy poniendo en grande al lado de la lupa.
2. Propongo separar la cantidad de SHOW a un texto con circulito a la derecha de show”

Foto:



Problema 19
Tipo: Mejora
Comentario:
“El candado de las pistas no se nota. Habría que poner otro.”

Problema 20
Tipo: Bug / Mejora
Comentario:
“El logo de ASys debería redirigir a la página principal”

12.2 Sesión 2

12.2.1 Informe 1

Informe Pruebas de Aceptación

Sesión	2 de 4	Usuario	Usuario 1: Josep Silva
Fecha	26/07/2021	Externo	No

Observaciones

Tanto el cliente como el servidor API se ejecutan en la misma máquina (portátil personal del desarrollador).
Maquinas probadas:
Macbook Pro con MacOS BigSur version 11.5
Movil Samsung Galaxy S10

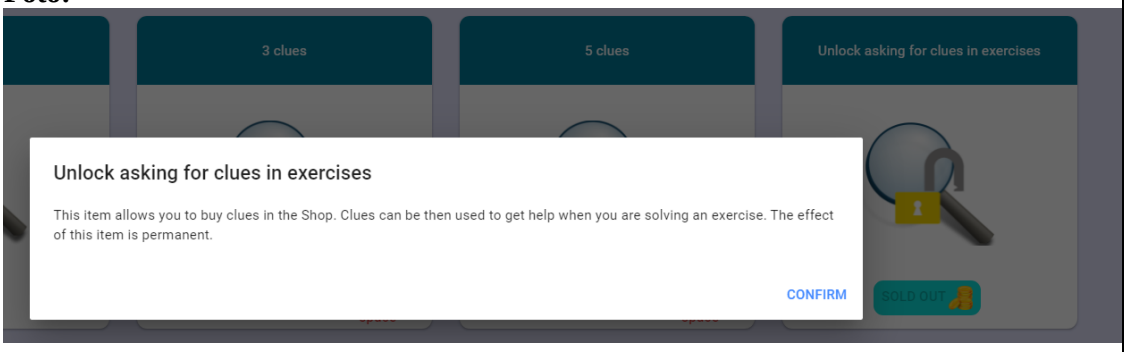
Navegadores probados:
+ Chrome Versión 91.0.4472.164
+ Firefox Versión 0.0.2 (64-bit)
+ Safari Versión 14.1.2

Resultados

Problema 21
Tipo: Mejora
Comentario:
"El avatar de arriba a la derecha debería poder ser clicable y llevarte al menú de avatars"

Problema 22
Tipo: Mejora
Comentario:
"El icono del dinero y las laureas de la cabecera (arriba a la derecha) debería poder ser clicable y llevarte a una página con información que te explique cómo ganar dinero. Concretamente, debería explicar cuánto dinero tienes actualmente y cuánto dinero ganas al resolver un ejercicio en función de tu nivel, el nivel del ejercicio..."

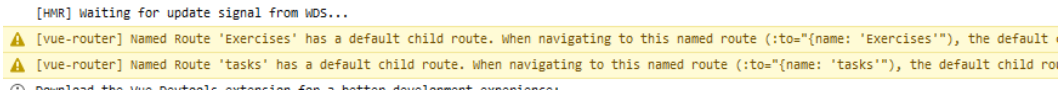
Problema 23
Tipo: Mejora / Error
Comentario:
"Los mensajes de información (popups con la i en las tarjetas y el botón INFO) son muy sosos (blanco y negro) y no siguen el formato de la web."
Foto:



Problema 24
Tipo: Bug (Error)
Comentario:
"Firefox da estos warnings al cargar el inventario:

[vue-router] Named Route 'exercises' has a default child route. When navigating to this named route (:to="{name: 'exercises'"), the default child route will not be rendered. Remove the name from this route and use the name of the default child route for named

links instead. vue-router.esm.js:16
[vue-router] Named Route 'tasks' has a default child route. When navigating to this named route (:to="{name: 'tasks'"), the default child route will not be rendered. Remove the name from this route and use the name of the default child route for named links instead."
Foto:



12.3 Sesión 3

12.3.1 Informe 1

Informe Pruebas de Aceptación

Sesión	3 de 4	Usuario	Usuario 3. Carlos Viñals
Fecha	2/08/2021	Externo	No

Observaciones
Tanto el cliente como el servidor API se ejecutan en la misma máquina (portátil personal del desarrollador). A pesar de ser un desarrollador de la plataforma, se distribuyeron credenciales de acceso al usuario y la dirección de acceso.

Resultados

Problema 25
Tipo: Mejora
Comentario:
"Home / Shop / Documentación
Justificar el texto de los recuadros (en general por lo que veo están alineados a la izquierda en varias páginas)"

Problema 26
Tipo: Bug (Error)
Comentario:
"Home / Shop / Documentación
Dentro de ¿Como funciona? faltan espacios en la descripción en castellano:
Linea 2: ... recompensa.Esta ... Linea 5: ... parte proporcionalde la ..."

Problema 27
Tipo: Mejora / Error
Comentario:
"Home / Shop / Documentación
Esta frase suena confusa:
"Si realizas un ejercicio con recompensa de 100 Monedas y 2 Láureas. Dependiendo del caso, obtendrás una recompensa u otra:"
Parece que indique que no ganas todo (las 100 Monedas y las 2 Láureas) al sacar un 10 a la primera, cuando en realidad se refiere a que tu recompensa depende de tu puntuación."

Problema 28
Tipo: Bug (Error)

Comentario:
"Home / Shop
Al comprar cosas:
¿Seguro que quieres comprar Sombrero de mago por 800 Coins}? <- La llave sobra. En algunos mensajes sale y en otros no"

12.3.2 Informe 2

Informe Pruebas de Aceptación

Sesión	3 de 4	Usuario	Usuario 4
Fecha	02/08/2021	Externo	Sí

Observaciones

Tanto el cliente como el servidor API se ejecutan en la misma máquina (portátil personal del desarrollador. Se distribuyeron credenciales de acceso al usuario y la dirección de acceso.

Resultados

Problema 29
Tipo: Bug (Error)
Comentario:
"La fecha-hora para determinar qué ítems diarios se muestran no es la del servidor, sino la del usuario. Esto permite al usuario cambiar la fecha y ver una selección de ítems distinta."

Problema 30
Tipo: Mejora
Comentario:
"El estado actual del inventario (9 ítems / 10 slots) solo se muestra en el inventario, y no en la tienda. Sería interesante informar al usuario mientras compra, para que no gaste su último hueco en un cosmético cuando quería comprar también una pista."

Problema 31
Tipo: Mejora
Comentario:
"Sería interesante añadir la opción de eliminar y/o vender ítems, ya que es posible que el usuario quiera borrar un avatar antiguo y hacer hueco para otra cosa. Esto es especialmente importante dado el coste de los ítems y de los huecos."

Problema 32
Tipo: Bug (Error)
Comentario:
"La información de la tienda indica que: (1) el precio de los ítems diarios tiene un 25% de descuento con respecto a los permanentes y (2) el precio de los permanentes tiene un 25% añadido con respecto a los diarios. Ambos no pueden ser ciertos, y según los precios que se muestran, parece que (2) es correcto."

Problema 33
Tipo: Error / Mejora
Comentario:
"Los colores para el nombre de perfil pueden causar cierta confusión y problemas de accesibilidad en determinados lugares de la aplicación."

Problema 34
Tipo: Error
Comentario:
"El usuario puede comprar los ítems que quiera por el precio que quiera

Las compras se realizan con una petición al endpoint /asysweb/shop/buy[Daily]Item/N
Además del header de autenticación (Authentication: [blob]), se pasan los datos de la compra en el cuerpo de la petición, en formato JSON. El usuario puede ver la petición con las herramientas de desarrollo de su navegador, y por tanto puede repetirla con un cuerpo distinto. De especial interés para el usuario serían los campos "precio":X (donde X es el precio a restar de tu cuenta; se puede disminuir todo lo que el usuario quiera, e incluso usar un valor negativo para ganar dinero con la transacción) y "value":N (donde N es la cantidad de ítems a comprar, ignorando el límite de inventario).
Todas las peticiones deberían verificarse contra la base de datos, y no confiar en la información enviada desde el cliente.
Así, se puede disminuir el tamaño de las peticiones entre el cliente y el servidor, ya que no tiene sentido enviar el precio desde el cliente cuando el servidor va a tener que consultarlo. Así, una compra solo necesita el id de la compra y el del usuario (derivado del header Authentication) para realizarse.
Es posible que este fallo exista en otros ámbitos, y permita al usuario enviar la calificación deseada para cualquier ejercicio u otras acciones que resultan en la modificación de la base de datos."
etc."

12.3.3 Informe 3

Informe Pruebas de Aceptación

Sesión	3 de 4	Usuario	Usuario 5
Fecha	02/08/2021	Externo	Sí

Observaciones

Tanto el cliente como el servidor API se ejecutan en la misma máquina (portátil personal del desarrollador. Se distribuyeron credenciales de acceso al usuario y la dirección de acceso.

Resultados

Problema 35
Tipo: Mejora
Comentario:
"Como elemento crítico, cambiaría la lista de ejemplos de ejercicios y monedas obtenidas por una tabla."

12.4 Sesión 4

12.4.1 Informe 1

Informe Pruebas de Aceptación

Sesión	4 de 4	Usuario	Usuario 1: Josep Silva
Fecha	21/08/2021	Externo	No

Observaciones

Tanto el cliente como el servidor API se ejecutan en la misma máquina (portátil personal del desarrollador). Estas pruebas se hicieron en el navegador Safari en un Macbook

Resultados

Problema 36
Tipo: Error
Comentario:
"En la documentación de la monetización, propongo modificar el texto de las unidades monetarias al siguiente:

"ASys has two kinds of monetary units: Coins and Laureas.

Coins are the most common monetary unit in ASys and they are easy to earn by solving exercises. With Coins you can buy most items in the Shop. Currently, you have 18448 Coins.

Laureas are a more valuable and rare monetary unit than Coins. Some valuable items can only be bought with laureas. Currently, you have 100 Laureas.""

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica

Universitat Politècnica de València