Escola d'Enginyeria de Telecomunicació i Aeroespacial de Castelldefels
UNIVERSITAT POLITÈCNICA DE CATALUNYA

TRABAJO FINAL DE GRADO

TÍTULO DEL TFG: Mejoras en el juego de *geocaching* de *Classpip (2)*

TITULACIÓN: Grado en Ingeniería Telemática

Autor: Javier Ramírez Morón

Director: Miguel Valero García

Fecha: 5 de julio de 2023

Título: Mejoras en el juego de *geocaching* de *Classpip (2)*

Autor: Javier Ramírez Morón

Director: Miguel Valero García

Fecha: 5 de julio de 2023

Resumen

Este documento describe el proceso creativo y organizativo de las funciones y los cambios que se han realizado en el entorno de gamificación *Classpip,* creado por la Universidad Politécnica de Cataluña, más concretamente por los alumnos de la EETAC en conjunto con el departamento de Arquitectura de Computadores de esta universidad.

Este entorno de aplicaciones está basado en diversos bloques, los cuáles serán explicados más adelante, el trabajo realizado se centra en específico en el juego de *geocaching*. Esta modalidad de juego, que también veremos más en profundidad en este proyecto, se basa en el uso de dispositivos que cuentan con sistemas GPS con el objetivo de buscar ubicaciones en un mapa con diferentes finalidades, en el caso que se verá a continuación el reto al encontrar los puntos serán una serie de preguntas.

En este proyecto se ha trabajado con la base del juego y con la finalidad de darle al usuario una experiencia mucho más agradable y atractiva, ya que el estado actual del juego es bastante limitado. En resumen, se mostrarán los nuevos pasos de control del juego y de los jugadores además de los pasos en la experiencia del jugador. También se trabaja sobre condensar y redistribuir la arquitectura del juego haciendo uso de las herramientas que el entorno de programación nos ofrece de manera que se consigue un código más entendible y dedicado. Esto aligerará en ciertos momentos el entendimiento de futuros desarrolladores en próximas contribuciones.

Este desarrollo se ha llevado de manera cooperativa con Ángel Franco, debido a que la normativa no permite entregar trabajos de final de grado conjuntamente, se añade esta nota al final de este resumen con el fin de destacar este hecho de cara a la lectura de la memoria de este proyecto ya que ambas compartirán mucho contenido común.

Title: Improvements on *Classpip's geocaching* game module (2)

Author: Javier Ramírez Morón

Director: Miguel Valero García

Date: July 5th 2023

Overview

This document describes the creative and organizational process of the functions and changes that have been made in the gamification environment Classpip, created by the Polytechnic University of Catalonia, specifically by the students of the EETAC altogether with the Department of Computer Architecture of this university.

This application environment is based on various blocks, which will be explained later. The work carried out specifically focuses on the geocaching game. This game mode, which we will also delve into in this project, is based on the use of devices equipped with GPS systems to search for locations on a map for various purposes. In the case we will see below, the challenge upon finding these points will involve a series of questions.

In this project, we have worked on the basis of the game with the aim of providing the user with a much more enjoyable and engaging experience, as the current state of the game is quite limited. In summary, we will present the new game control and player steps, as well as the player experience steps. We also work on condensing and redistributing the architecture of the game, making use of the tools that the programming environment offers us in order to achieve a more understandable and dedicated code. This will facilitate the understanding of future developers in upcoming contributions at certain moments.

This development has been carried out cooperatively with Ángel Franco, as regulations do not allow joint submission of final degree projects. This note is added at the end of this summary to highlight this fact for the reading of the project report, as both will share a lot of common content.

Este proyecto no podría ser posible sin la ayuda de varias personas:

A Miguel y Roc por la tutoría y la constante ayuda que nos han ofrecido, imprescindible a la hora de completar este trabajo. A mis padres y mi hermano Víctor por haberme guiado, cuidado y apoyado desde el primer día. A Ari por tener siempre claro que era capaz de hacerlo aun cuando yo no lo he tenido tan claro y acompañarme todos estos años, y a todos los compañeros que he conocido en toda la carrera, a Eduard por hacer a las veces de mi jefe, profesor y compañero, a Ángela y a mis chicos de Cornellà y por supuesto a Ángel, por ser el otro 50% de mi equipo durante toda la carrera.

Gracias a todos.

ÍNDICE

INTRODUCCIÓN

A día de hoy la tecnología y la vida en sociedad son conceptos prácticamente inseparables. En el mundo laboral, en el hogar, en nuestros ratos de ocio y en la educación... es difícil innovar o plantear nuevas metodologías, nuevos algoritmos que nos faciliten la vida o que nos aporten algo diferencial sin tener en cuenta la tecnología.

Este documento se centrará en el segundo bloque, en el desarrollo de *software* y en una aplicación que en estos últimos 10 años, muchos han experimentado a diferentes escalas del sistema educativo, desde párvulos a universitarios de último año: la gamificación.

El concepto de la gamificación, el cual se profundizará más adelante, no es más que la intersección entre juegos y aprendizaje. La palabra "juego" lleva a cualquiera a pensar en ocio, descanso, tranquilidad, desconexión... es por eso que si se hace uso de mecánicas o dinámicas asociadas a los juegos pero con un fin no recreativo necesariamente, se pueden conseguir objetivos que pueden ser realmente beneficiosos para todos, además de establecer un nuevo paradigma en algunos sectores. En el caso de este proyecto, la gamificación se tratará en su vertiente educativa principalmente.

Llegados a este punto, nace *Classpip* como una iniciativa que trabaja en la gamificación de las aulas desde la *Escola d'Enginyeria de Telecomunicacions i Aeroespacials de Castelldefels* (EETAC), una manera en la que el interés de los alumnos en el mundo de los videojuegos, el aprendizaje y la enseñanza convergen en un punto con el mismo objetivo: seguir motivando y enseñando.

Entrando en detalle, el proyecto *Classpip* consta de diversas partes y bloques. En este desarrollo el punto central es el juego de tipo *geocaching*, este tipo de juegos basan su funcionamiento en la geolocalización: los alumnos buscarán haciendo uso de dispositivos con servicio GPS diferentes localizaciones establecidas previamente por el profesor. En cada punto, los alumnos se encontrarán con retos que deben superar compitiendo entre ellos. Siguiendo este modelo de base, se consigue un *engagement* por parte de los alumnos estableciendo una modalidad competitiva al mismo tiempo que se promueve el aprendizaje.

Se estructurará esta memoria en tres bloques. El primer bloque contendrá los 3 primeros capítulos. En el primer capítulo se trata el trasnfondo y la finalidad de este entorno y se introduce en detalle que es el entorno *Classpip*, explicando cada una de las partes que lo conforman. El segundo capítulo tratará sobre cómo se encuentra este entorno a nuestra llegada y las áreas que se estudiaron de cara a este proyecto. El tercer capítulo muestra una lista de objetivos, esta lista definirá que puntos serán los centrales durante nuestras aportaciones, de manera que al final de este trabajo se puedan evaluar si se han conseguido. En este primer bloque, el contenido es el mismo en ambas memorias. La diferencia está en el capítulo 2 donde cada uno, ha explicado las primeras mejoras que realizó (veáse apartado 2.2.1).

El segundo bloque sigue con los cinco capítulos siguientes. El cuarto capítulo hablará sobre el diseño inicial y el punto de partida de esta colaboración. En el quinto capítulo se hablarán sobre las tecnologías con las que se ha trabajado. El sexto capítulo trabaja las implementaciones y presenta código sobre las mejoras y cambios que se han realizado para conseguir el desarrollo. El séptimo menciona algunas mejoras que nacen fruto de ciertos obstáculos a la hora de hacer las implementaciones. Este bloque finaliza con el octavo capítulo que se enlaza con el primero de este bloque y describe el diseño final de la interfaz y los cambios que ha sufrido respecto al punto inicial. En este bloque, el cuarto, el quinto, el séptimo y el octavo capítulo tienen el mismo contenido en ambas memorias.

En el último bloque se encuentran las evaluaciones del proyecto desde distinto puntos y contienen los capítulos 9, 10 y 11. El noveno y el décimo capítulo hablará sobre pruebas que hemos realizado, primero como hemos comprobado las implementaciones que hemos aportado y posteriormente las pruebas con diferentes usuarios y el *feedback* que se ha recibido, respectivamente. El undécimo capítulo contiene propuestas de mejoras sobre las aportaciones realizadas de cara a futuros colaboradores.

Finalmente, el proyecto cierra con el doceavo y último capítulo que contiene las conclusiones de todo el trabajo de final de grado. El noveno y el décimo capítulo contienen el mismo contenido en ambas memorias.

Todo el trabajo realizado ha sido dividido y compenetrado entre yo, Javier Ramírez Morón y Ángel Franco Martos, de manera que hemos ido avanzando a la par en lo relativo a las aplicaciones del alumno, el profesor y sobremesa de *Classpip* sobre el juego de *geocaching*.

En mi caso, me he focalizado más en la implementación de mapas con geolocalización en tiempo real, el estilizado de las interfaces de usuario y en el diseño final de la interfaz y las nuevas implementaciones, además del despliegue en móvil.

Mi compañero Ángel se ha encargado de la generación y redistribución de las funciones que puede hacer uso el usuario y mejorar errores en la aplicación web de sobremesa donde se generan los juegos, además de la generación de nuevas componentes.

De manera conjunta, hemos trabajado en el diseño de las pantallas y los cambios en las mecánicas del juego que estaban establecidas.

CAPÍTULO 1: CLASSPIP

Este capítulo introduce el concepto en el que se apoya el entorno de trabajo en el que se va a profundizar. Primero, se explica que es el concepto de gamificación para unirlo con la definición de *Classpip* y los diferentes bloques en los que se ha basado su diseño.

1.1 La gamificación: concepto e impacto

La gamificación, anglicismo ampliamente usado que referencia a la ludificación[1], según la Real Academia Española[2] se define como la acción por la cuál algo se *"transforma en un juego o trata de fomentar los aspectos lúdicos de este".* La RAE también contiene una segunda acepción que define la gamificación como *"el proceso por el cuál se aplican técnicas o dinámicas propias del juego a actividades o entornos no recreativos para potenciar la motivación y la participación, o facilitar el aprendizaje y la consecución de objetivos."*

Este proceso genera una gran serie de beneficios allá donde se aplique, principalmente debido a que con esta herramienta se permite estudiar el comportamiento de las personas y estimularlo para mejorar ciertos aspectos de un producto, como bien podría ser una red social o una aula, haciendo más atractivas para aquellos a los que van dirigidas cuales sean las mecánicas establecidas en ciertas actividades. El objetivo principal se puede observar que es la motivación, un estudiante motivado irá a clase con más ganas, un trabajador motivado asistirá a su lugar de trabajo en su día a día con más incentivos y será más productivo, un usuario de una aplicación entrará más a diario para seguir consiguiendo logros a cambio de recompensas.

De esta manera, se consigue atraer, incentivar, motivar y promover un estado de ánimo que en algunos ámbitos como el laboral o estudiantil definen las claves de trabajadores más profesionales o de estudiantes más motivados.

Pivotando hacía *Classpip*, es palpable en la sociedad en la que se desarrolla este entorno que la educación española tiene algunas lacras que afectan a los estudiantes en formación, como la desmotivación, que hace que cada día muchas personas se levanten sin ganas de aprender, acudiendo por obligación a un centro de enseñanza con el mero objetivo de que pasen las horas y poder volver a casa. Esto genera una capa de estudiantes cuyo objetivo es "pasar", conseguir el mínimo y avanzar de curso, sin interiorizar nada, sin aprender nada o lo básico y causando un decremento de la calidad del sistema educativo. Cabe destacar que hay muchos factores que propician este hecho y la responsabilidad no es solo de los centros o de los docentes. En este contexto, no atañe hacer un análisis pormenorizado de la situación del sistema educativo español pero es el punto de partida en el cual el concepto de gamificación puede entrar con muchísima fuerza y mejorar el paradigma en el que millones de estudiantes y docentes se encuentran cada jornada.

Un caso en el que la aplicación de este tipo de herramientas puede ser clave es observar la tasa de abandono escolar en España entre personas de 18 a 24 años

se mantenía en un 13,9% en 2022[3]. Personas que no han completado la segunda etapa de la Educación Secundaria y que no sigue ningún tipo de formación. Si se consigue estimular e incentivar al alumnado es posible reducir esa tasa y mejorar exponencialmente la calidad del sistema educativo.

Si se pretende "gamificar" un aula, hay algunos parámetros que se deben tener en cuenta para conseguir un entorno con éxito, estos son:

- Elecciones
- Progreso
- Social
- Hábitos y lealtad
- Divertido
- Que se comporte como un juego

Aplicado a una aula, el objetivo por parte del docente será qué el alumno aprenda, a través de la creación de juegos con diferentes objetivos dentro del entorno, un profesor puede probar los conocimientos de sus alumnos de una manera mucho más dinámica y divertida de cara a estos últimos. Siguiendo una línea paralela con la metodología tradicional de clases expositivas, ejercicios, examen y evaluación, la gamificación encaja a la perfección en un esquema muy presente a día de hoy.

Otro punto con más desarrollo que hay que precisar es definir una finalidad clara, un objetivo que se pretenda alcanzar a la hora de gamificar un espacio. Es importante y esencial construir un entorno de gamificación siguiendo las claves anteriores pero donde recae gran parte de la responsabilidad del buen funcionamiento de este es en la finalidad para la cuál ha sido diseñado. Esta gran herramienta, si no se diseña correctamente, puede ser contraproducente y generar más problemas que ventajas.

Finalmente, el factor humano juega un papel fundamental. Es necesario hacer un estudio de los usuarios que van a ser participantes del entorno de gamificación en diseño, si esto no se tiene en cuenta, también es posible llegar a puntos no deseados por parte de los participantes o incrementar situaciones de conflicto latentes[4] que estén o se puedan dar como puede ser tanto en una aula entre alumnos como en una oficina entre profesionales de un mismo sector.

Aunque cabe destacar que la gamificación no solo se debe al aprendizaje, la gamificación también puede ser aplicada en los estudios de conducta y cambios de comportamiento de los estudiantes o de profesionales de cualquier ámbito laboral. Un ejemplo de gamificación en el mundo laboral por ejemplo que destaca este último párrafo es visible en la colección de videos *The Fun Theory* de la empresa Volkswagen[5].

1.2 Geocaching: concepto y ejemplos

El *geocaching*[6] es una actividad que se basa en buscar objetos en cualquier lugar a través del uso de dispositivos con servicio GPS. La mecánica es simple de explicar, un usuario esconde un objeto y otro lo busca a través de las coordenadas. El origen de esta práctica lo marcó la celebración por parte de David Ulmer, un usuario de internet de un grupo de noticias que se centraba en los GNSS (Sistemas Globales de Navegación por Satélite), de la derogación de la disponibilidad selectiva.

La disponibilidad selectiva (SA)[7] fue la degradación intencional por parte del gobierno de los Estados Unidos de América de las señales GPS como método de protección contra ataques externos o amenazas militares evitando así una precisión muy exacta de los receptores de señal GPS. Debido a diversos factores, entre ellos que había métodos para evitar esta degradación y la creciente necesidad de un uso incrementado por parte de la población provocó que la administración Clinton derogase el uso de esta práctica el 2 de mayo del año 2000.

David Ulmer decidió proponer un juego a otros miembros del grupo de noticias escondiendo un "tesoro" a los alrededores de Portland (Oregón) y enviando las coordenadas del objeto en si, el 3 de mayo de ese mismo año.
Así, nacería esta práctica, que a día de hoy cuenta con más de 3 millones de jugadores en activo y alrededor de 3,4 millones de tesoros escondidos. Estos datos, cabe destacar, que hablan del *geocaching* como práctica registrada que tiene una web desde septiembre del año 2000 y una aplicación oficial desde el 9 de marzo de 2016. Aun así, son muchas las aplicaciones y proyectos tecnológicos basados en el *geocaching*.

A continuación, se muestra una lista con aplicaciones o proyectos que han implementado el *geocaching* de diferentes maneras[8]:

- **Pokemon Go:** Partiendo del juego original, en este juego para móviles los jugadores se transforman en entrenadores *Pokemon* y buscan capturar a todos los *Pokemon*, estos se encuentran en coordenadas que el jugador puede ver en su móvil para acercarse y capturar al *Pokemon* en cuestión como podemos ver en la figura 1.1.
- **Randonautica:** Esta aplicación móvil de gran popularidad genera coordenadas de manera aleatoria, con una serie de algoritmos, con el objetivo de que los jugadores exploren esa zona y descubran detalles inusuales de sus entornos. La interfaz de usuario se puede observar en la figura 1.2.
- **Ingress:** De los creadores de *Pokemon Go*, este juego se basa en explorar ubicaciones con el objetivo de controlar y capturar portales.
- **Cachly:** A modo de red social, esta aplicación permite buscar y registrar tesoros encontrados y conectar con otros jugadores/exploradores.
- **Zombies, Run!:** Con el objetivo de motivar el ejercicio físico, los jugadores corren siguiendo una narración basada en un apocalipsis zombi, de manera inmersiva, el juego ofrece recompensas virtuales y objetos que se pueden encontrar en cada ruta.

- **WallaMe:** Esta aplicación permite a los usuarios dejar frases y mensajes ocultos en lugares reales, así mismo otros jugadores solo pueden leer esos mensajes con la aplicación una vez lleguen a la misma ubicación.

Fig.1.1*:* IU de *Pokemon Go.* (Fuente: Google Imagenes, *Xataka*)

Fig. 1.2: Aplicación de *Randonautica.* (Fuente: Google Imagenes, *Xataka*)

Estos son algunos de los ejemplos que han usado mecánicas derivadas del *geocaching* por definición y han dado pie a diversos usos, como se puede ver algunos son juegos, pero por ejemplo, si se le echa un vistazo a "*Zombies, Run*" se puede encontrar una relación con un concepto visto anteriormente.

Este es un ejemplo donde la gamificación y el *geocaching* convergen. La actividad física diaria como bien puede ser correr, tiende a ser muy rutinaria y monótona, en cambio, haciendo uso de esta aplicación, se generan una serie de motivaciones e intereses extra en el hecho de salir a correr, mantener un ritmo y cumplir objetivos, y de esta manera se consigue el ejercicio físico tan saludable que la persona quiere ejercer para sí misma.

1.3 ¿Qué es Classpip?

Classpip[9] (2016) es un concepto que nace de la mano de Roc Messeguer y Miguel Valero, dos profesores del departamento de Arquitectura de Computadores de la Universidad Politécnica de Catalunya, más en concreto en la EETAC (*Escola d'Enginyeria de Telecomunicacions i Aerospacials de Castelldefels)*.

A partir de aquí, en colaboración con estudiantes de último año a través de sus trabajos de fin de grado o fin de máster, el entorno se ha ido diversificando y conformando como una herramienta de gamificación muy completa basada en diferentes aplicaciones y funcionalidades[10]. El objetivo que tiene *Classpip* no es otro que el hecho de motivar a alumnos en entornos académicos a seguir motivados y aprendiendo día a día. Esto es conseguible gracias a la implementación de un sistema de recompensas donde el alumno ve de manera tangible su progreso en una asignatura o progreso de evaluación mientras que pone a prueba sus conocimientos adquiridos. A continuación, en la siguiente figura 1.3, es visible el logotipo oficial que lidera este entorno.

Fig. 1.3: Logotipo de *Classpip.* (Fuente: Web Classpip)

El punto principal se encuentra en la diversidad de juegos que contiene el entorno, siendo este un punto fuerte a la hora de dinamizar un aula académica. Por otro lado, el ecosistema *Classpip* también ofrece la posibilidad de ser una herramienta de gestión para el profesorado, pudiendo acceder a diferente información de gran utilidad sobre sus alumnos y los grupos que tienen a cargo.

Además *Classpip* es un entorno de gamificación que se puede extender a cualquier escala educativa o titulación y los métodos que contiene son perfectamente compatibles con la metodología actual docente, en definitiva es una herramienta que solo ofrece ventajas y acompañamiento tanto al equipo docente como al estudiante en su día a día por conseguir los objetivos.

1.3.1 Arquitectura de software

Classpip basa su arquitectura en diferentes módulos que se presentan como aplicaciones móviles o web que se conectan a un servidor. El servidor a su misma vez, gestiona y controla el acceso a los datos de interés y a la información sensible por parte de quién la requiera. En la imagen siguiente, la figura 1.4, se puede observar en detalle la interacción entre los diferentes módulos y las diferentes tecnologías que se detallarán en este apartado.

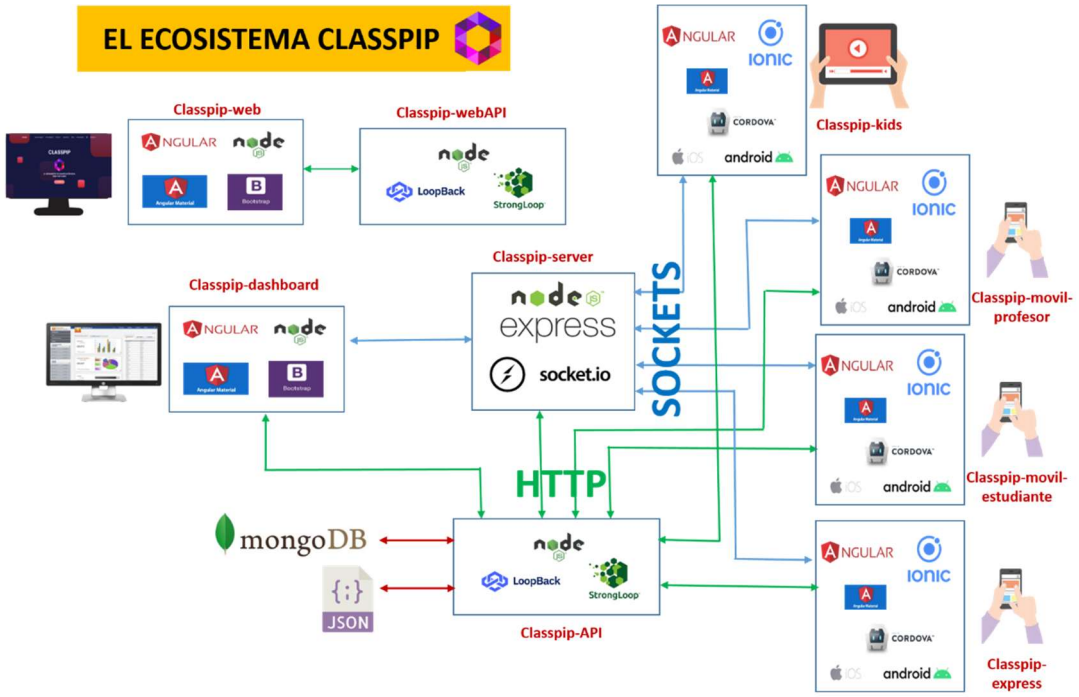

Fig. 1.4*:* Esquema de la arquitectura de *Classpip.* Fuente: *GitHub* de *Classpip*

1.3.1.1 Classpip-server

Partiendo del núcleo del entorno, el servidor se desarrolla utilizando *node.js* y *express*. *Node.js* es uno de los principales *frameworks* para desarrollo *backend* con *Javascript* debido a las facilidades que ofrece a la hora de programar y *express* es un *framework* sobre *node.js* que sirve para desarrollar aplicaciones web, por lo que se usa para generar y obtener una aplicación de servicio. Aunque estas aplicaciones de *backoffice* no son accesibles por los usuarios. En términos de comunicación, utiliza peticiones HTTP para conectarse con la API y sockets con el resto de módulos, con lo que hace uso de *socket.io.*

1.3.1.2 Classpip-API

Basada en el modelo *API-RESTful*, este módulo ofrece al resto de aplicaciones el servicio de acceso a la base de datos y información de interés para cada uno de los módulos y sus respectivas funcionalidades. Debido a ser *REST*, las comunicaciones se dan vía HTTP. Se utiliza *Loopback*, creado por *Strongloop*[11],

debido a que es un *framework* sobre *node.js* al igual que el servidor y cuya finalidad está definida para crear APIs y microservicios aprovechando que se trata de un componente *open-source* privado del tipo mBaaS (mobile-Backend-as-a-Service).

1.3.1.3 Classpip-webAPI

Aunque forme parte del entorno completo de *Classpip*, este módulo conjuntamente con *classpip-web*, no forman parte del conglomerado principal en el que se ha trabajado. Esta API da servicio de backend a la web de Classpip y utiliza las mismas tecnologías que la API principal.

1.3.1.4 Classpip-web

Se trata de una aplicación web. Funciona como un portal de información sobre el proyecto del ecosistema de *Classpip* y ofrece a cualquier persona información, acceso y interacción de algunas de las herramientas del entorno. Esta hecha usando *Angular* y *Node.js*, y usando *Angular Material* y *Bootstrap* para las componentes HTML.

1.3.1.5 Classpip-dashboard

Funciona como una aplicación web que da acceso a la gestión de recursos y funcionalidades a los usuarios que tienen el rol de profesor. Al ser una aplicación web, se ha seguido una hoja de ruta similar que a *Classpip-web* y utiliza las mismas tecnologías. En la figura 1.5, es observable el aspecto de esta app web.

Fig. 1.5*:* Captura de pantalla de la aplicación web *Dashboard* de *Classpip*

1.3.1.6 Classpip-movil-estudiante

A partir de aquí, los siguientes bloques de *Classpip* se tratan de aplicaciones móviles. En este caso las aplicaciones han sido realizadas usando *Ionic* como *framework* de desarrollo y el cuál sigue una estructura dividida en componentes al igual que *Angular*. De hecho, *Ionic* está construido sobre *Angular* pero orientado a aplicaciones en *iOS* y *Android*. En términos de diseño de interfaz, este se basa en HTML y para ello han sido utilizados elementos tanto de *Angular Material* como de la propia librería de *Ionic*.

De cara a generar la aplicación resultante, se ha utilizado *Apache Cordova* para generar los archivos *.apk* en el caso de *Android* y *.ipa* en *iOS*. Como ejemplo visual, la siguiente imagen, la figura 1.6, muestra lo que se encuentra el alumno *e1* cuando accede a la aplicación del alumno, en este caso desde una página web, a pesar de que como se ha explicado la finalidad es que el despliegue de esta aplicación sea para móvil.

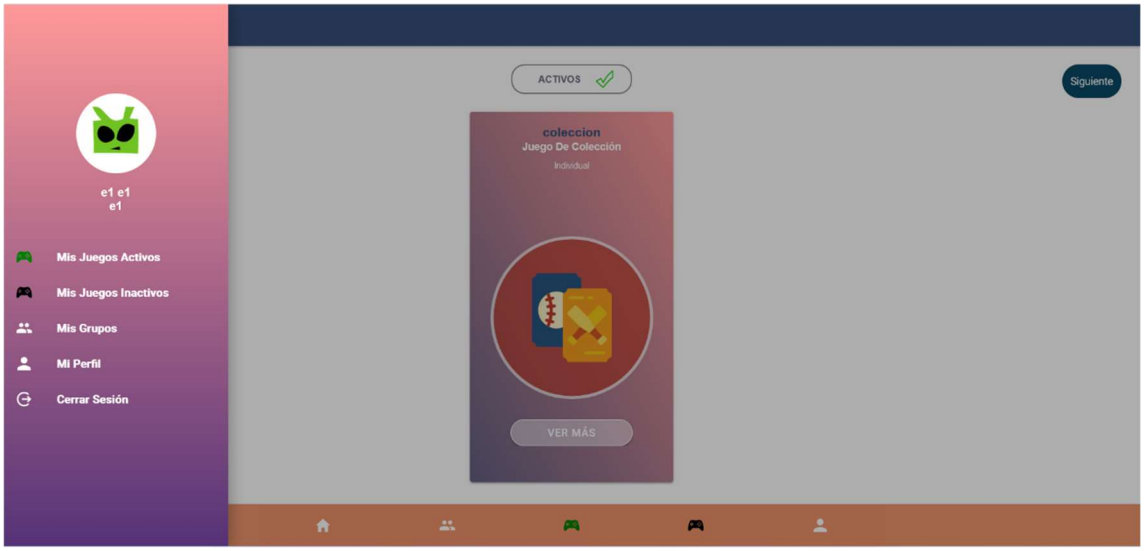

Fig. 1.6: Captura de pantalla de la aplicación móvil del estudiante

Entrando en detalle de *classpip-movil-estudiante*, es una aplicación móvil donde los alumnos hacen uso de los recursos que el profesor genera y diseña desde el *dashboard*. Además también permite a los alumnos hacer algunas gestiones sobre su perfil y consultar información de los grupos en los que está inscritos.

1.3.1.7 Classpip-movil-profesor

Como se puede ver en la figura 1.7, las tecnologías que utilizan todas las aplicaciones móvil de este entorno son las mismas. En este caso se trata de una aplicación complementaria a la aplicación web *classpip-dashboard.* Una aplicación complementaria, tal como dice la palabra, sirve para añadir unas

herramientas extra que faciliten algunas de las funciones que realiza otra aplicación a partir de la cual se extiende.

Esta aplicación se diseña para que el profesor controle y monitorice ciertos juegos o procesos de los alumnos y grupos que tiene a cargo y puede interactuar y/o monitorizar los diferentes juegos o pruebas que genera a través de la aplicación web. Como es posible percibir en la imagen siguiente, la figura 1.7, la interfaz de usuario para los profesores es la misma que en el caso de la aplicación para los alumnos.

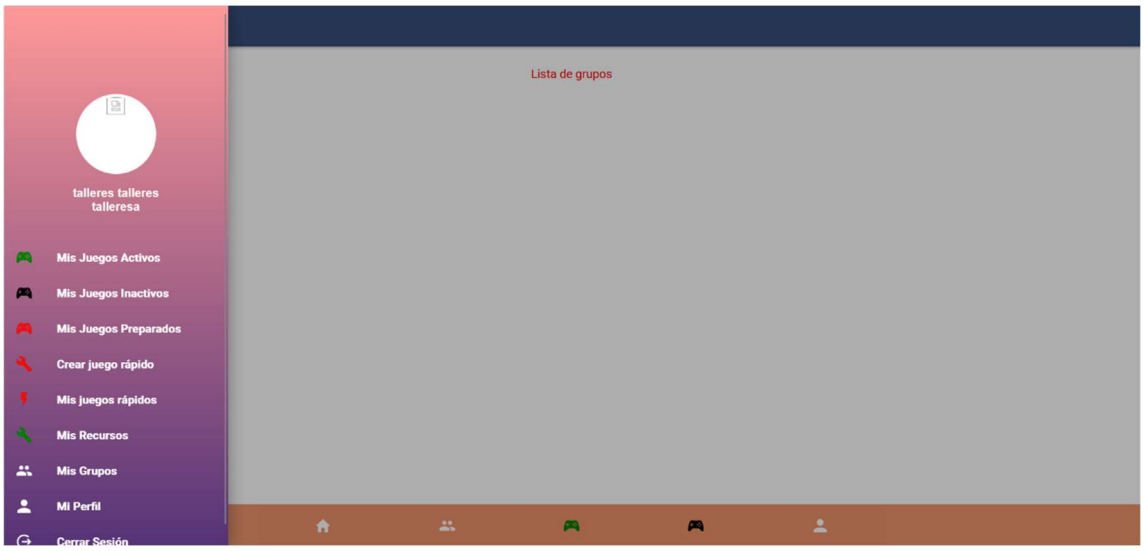

Fig. 1.7: Captura de pantalla de la aplicación móvil del profesor

1.3.1.8 Classpip-kids

Esta aplicación móvil, tiene un enfoque más dedicado al uso de tablets y tiene como *target* los alumnos de menor edad y más pequeños. Los juegos y recursos que se generan para esta aplicación tienen un carácter más visual y infantil como por ejemplo, un juego de creación de cuentos.

1.3.1.9 Classpip-express

Esta aplicación parte de la base de generar una rama del propio entorno pero de una manera más liviana y accesible. Esto en el sentido, de que permite hacer uso de recursos y funcionalidades de *Classpip* sin ser un usuario dado de alta en el entorno *Classpip*.

1.3.2 Definición y descripción de funcionalidades y juegos

Dentro de *Classpip* tenemos diferentes tipos de juegos y funcionalidades que el profesor puede generar:

- Juego de puntos: Los alumnos deberán de realizar una serie de acciones en clase (por ejemplo; buenas notas, participación en clase, etc.) y a cambio de eso, el profesor asigna puntos. Estos puntos se traducen en un ranking donde se puede ver el orden por puntos y además, cuanto mayor sea la cantidad de puntos, más privilegios o premios consigue el alumno.

- Juego de Colección: El profesor define una colección de cromos. Los alumnos deberán realizar méritos para conseguir cromos y completar la colección para una recompensa mayor. Este juego permite además el intercambio de cromos entre alumnos para conseguir completar la colección.

- Juego de Competición: Este juego tiene algunas diferentes modalidades en función del modo de eliminación o victoria. Hay tres tipos: tipo F1, tipo torneo y tipo Liga. El de tipo F1 basa la asignación de premios en función de la posición que se consigue en un ranking. El de tipo torneo funciona como un clásico torneo de *knockout*, en el que si se requiere, se pueden establecer jornadas de "repesca". Finalmente el de tipo Liga, funciona como una liga deportiva al uso, todos contra todos dividido en jornadas donde en cada jornada cada alumno se enfrenta a uno diferente.

- Juego de Cuestionario: El profesor crea un cuestionario con distintas preguntas y tipos de respuesta y los alumnos deben contestar, cuanto más se acierten, mayor es la puntuación final o nota.

- Juego de Avatar: A modo de recompensa, los alumnos pueden editar la imagen con la que son vistos en el entorno *classpip*. En este apartado el alumno puede editar su avatar y modificarlo añadiendo diferentes elementos a una imagen.

- Juego de Geocaching: Haciendo uso de la geolocalización, se asocian ubicaciones físicas reales con preguntas. Así pues, para ganar o puntuar hay que responder las preguntas correctamente con el punto extra de que se debe buscar el punto geolocalizable haciendo uso de una o dos pistas.

- Juego de Votación: En este caso, se basa en generar una votación por parte del profesor hacia los alumnos para decidir cosas en común.

- Juego de Cuestionario de Satisfacción: Siguiendo un esquema similar al juego de cuestionario, este se caracteriza por hacer uso de preguntas abiertas y con el objetivo de obtener un *feedback* por parte del alumnado sobre una actividad, asignatura o juego.

- Juego de Evaluación: Este juego permite la evaluación entre compañeros y/o equipos y consultar la nota final que obtiene uno mismo.

- Control de trabajo en equipo: A través del uso de un formulario, se realizan controles grupales.

- Juego de Coger Turno rápido: Este bien puede servir para asignar un turno en el horario del alumno y el profesor, donde se le ofrece al alumno una lista de turnos disponibles y él estudia la posibilidad de cuál le conviene más en caso de necesitarlo.

- Kids: Juego de cuentos, de memorama o de puzle. Son juegos definidos para un público más infantil donde aprenden jugando a crear historias, unir y asociar imágenes y conceptos y crear y realizar puzles en tablets.

Cada juego tiene dentro sus propias mecánicas y finalidades y cada uno de los juegos se puede generar para ser jugado en modo individual o en equipo, exceptuando el “Control de trabajo en equipo” que, como su nombre indica, es exclusivo para equipos. Mientras los alumnos juegan y aprenden a través del uso de su aplicación móvil, el profesor puede gestionar el inicio o final del juego, el progreso que llevan los alumnos y finalmente la nota o evaluación de los resultados. Esta última también es consultable por parte de los estudiantes.

Cómo se verá en el siguiente capitulo, los puntos fuertes del desarrollo que muestra este trabajo de fin de grado se centrará en las partes de la aplicación web *classpip-dashboard* y la aplicación móvil *classpip-app-estudiante*. En el transcurso de estas mejoras, se buscará también de hacer una propuesta de mejora sobre la aplicación móvil análoga del profesor, *classpip-app-profesor* que encaje con el modelo final resultante de los cambios en la app del alumno.

CAPÍTULO 2: State of Art

State of Art es un concepto utilizado para referirse al punto en el que una persona u organización encuentra un proyecto. En este caso, este fue el punto de partida a la hora de comenzar a pensar, diseñar y desarrollar soluciones o mejoras. Partiendo de esta idea, este desarrollo se basa principalmente en partir de una base que ya ha sido cimentada y buscar mejoras o puntos que requieren de más desarrollo. En este capítulo, se profundiza sobre el *state of art* de *Classpip* y como se encuentran los bloques o módulos en los que hemos trabajado en el momento en el que empezamos a formar parte del proyecto.

En el caso de nuestro desarrollo, quisimos optar por explorar partes del ecosistema *Classpip* donde poner a prueba los conocimientos que recientemente habíamos aprendido y puesto en práctica en asignaturas del grado de Ingeniería Telemática, debido a que en la parte final de la carrera había una parte bastante grande de desarrollo de software y programación como se pueden observar en las asignaturas de DSA y EA como bloques más grandes, aunque no son las únicas. Como segundo factor de decisión, se optó también por evaluar las partes del entorno menos desarrolladas o que tuviesen un potencial de mejoras más grande.

2.1 Estado del proyecto

Las partes más exploradas fueron principalmente el *dashboard*, el móvil del estudiante y el móvil del profesor. En un primer momento, se llega a la conclusión de que el mejor punto de partida es la aplicación web, como punto central de creación de recursos y juegos. Se observa una gran cantidad de recursos y una aplicación muy completa y con mucha diversidad pero al mismo tiempo, se encuentran pequeños fallos que impiden el correcto funcionamiento de algunas de las funciones y recursos. Errores o falta de algunos recursos que no tenían una gran complejidad pero que sí que ofrecían mejoras de gran interés.

En lo relativo a las aplicaciones móvil se encuentra algo similar, aplicaciones completas y con una diversidad importante de opciones y funciones pero, bajo nuestro humilde punto de vista, con un gran margen de mejora en algunas de las partes, como ya se comentará en los siguientes apartados.

Entrando en materia, después de una primera toma de contacto y unas primeras mejoras y correcciones en el *dashboard*, se decide trabajar sobre el juego de tipo *geocaching.*

Este apartado se nos presenta como un juego que hace uso del servidor y la API como *backend* y de la aplicación web *dashboard*, aplicación móvil del estudiante y aplicación móvil del profesor como puntos *frontend* donde cada rol de usuario interactúa con el juego de distinta forma.

2.1.1 Classpip-dashboard

En la parte del *dashboard*, se encuentra el proceso de creación del juego y de los escenarios. Primero, se tendrá que crear el juego. La creación de los juegos se realiza desde los grupos que el profesor tiene. En esta primera etapa, se encuentra un *stepper* que pide *Nombre del juego, Tipo de Juego* y *Modo*. En el momento que se completan estos tres primeros pasos y se selecciona el juego de tipo *Geocaching*, el *stepper* se amplía para poder añadir en los siguientes pasos el escenario, las preguntas, las puntuaciones y un botón para finalizar y guardar el juego.

El escenario no es más que el conjunto de puntos geolocalizables donde los alumnos encontrarán las preguntas haciendo uso de pistas. Es por ello, que antes de continuar con el proceso de crear un juego de tipo *geocaching*, comentamos como es la creación de Escenarios.

Estos escenarios son necesarios para poder generar el juego, y se crean desde la pestaña de Recursos, donde generalmente se pueden ver los recursos que el profesor haya creado anteriormente o se pueden crear más. En el caso de los escenarios, se trata de un formulario, como se puede ver en la figura 2.1, donde se introduce el nombre de la ubicación, las coordenadas geográficas, la pista "difícil" y la pista "fácil".

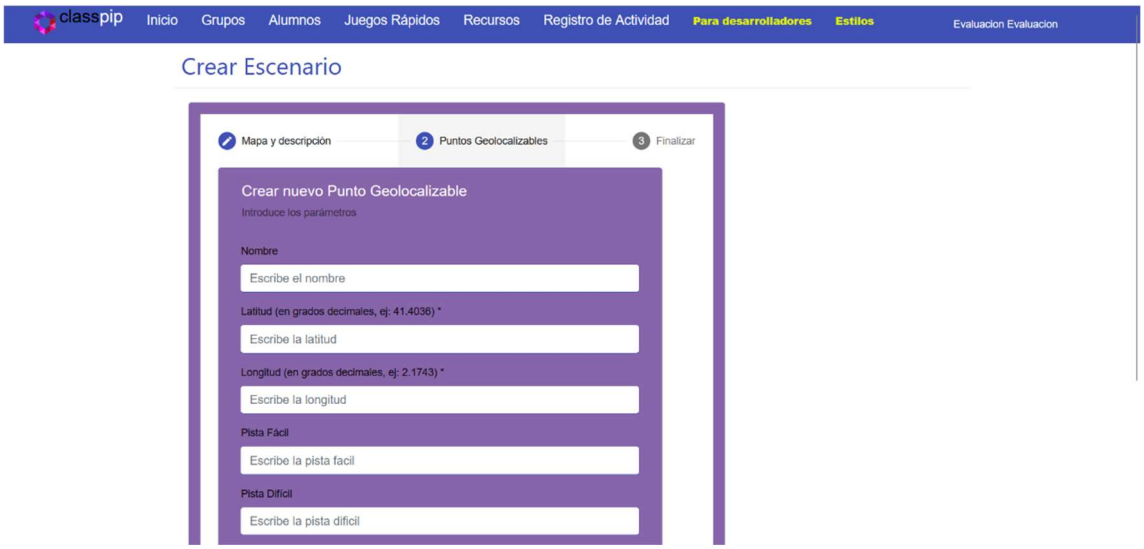

Fig. 2.1: Creación de escenarios en el *Dashboard*

De la misma manera, el profesor tiene que tener una serie de preguntas, que en caso de no tener ninguna, en la misma pestaña de Recursos puede crear Preguntas. Estas pueden ser cargadas en un fichero de tipo JSON o creadas a partir de un *stepper* (ver figura 2.2) donde se introduce el título de la pregunta, que tipo de pregunta es (cuatro opciones, respuesta abierta, verdadero o falso o emparejamiento), la pregunta en si y la temática. Dependiendo el tipo de pregunta, el *stepper* muestra diferente información que introducir pero los siguientes pasos son asignar una imagen a las preguntas, introducir la respuesta

correcta y mensajes de *feedback* al alumno en caso de que acierte o se equivoque.

Fig. 2.2: Creación de preguntas en el *Dashboard*

Partiendo de la base que se tienen ya un escenario y una serie de preguntas creadas, en los siguientes pasos, como se menciona al inicio de este apartado, se selecciona el escenario, las preguntas en función de la cantidad de puntos geolocalizables que haya y se guarda el juego a través de la creación de juegos como se puede ver en la siguiente figura 2.3.

Fig. 2.3: Creación de un juego de tipo Geocaching en *Dashboard*

2.1.2 Classpip-movil-estudiante

Ahora, de cara a las aplicaciones móviles, lo que se ofrece de primeras y centrando el punto de mira en el juego de tipo Geocaching es que en el caso del alumno el juego se basa en un *stepper*.
A continuación, este *stepper* es visible en la siguiente figura 2.4, la cual en orden descendiente muestra los pasos de los que consta el juego y que son detallados en ese mismo orden en cada párrafo.

En un primer punto, se lee una descripción del juego que se va a realizar y en el segundo paso informa del nombre del juego y de los puntos que se otorgan por acierto y por fallo, junto con un botón de Empezar. Si se clica en el botón de empezar, entramos en el 3r punto. Lo que hay por pantalla es la 1a pista, conocida como la pista *dificil* y un botón de Rendirse. El botón de Rendirse da acceso a la 2a pista, conocida como pista *fácil*, ya que es mucho más esclarecedora y en caso de rendirnos, nos informa también de la distancia a la que estamos del punto.

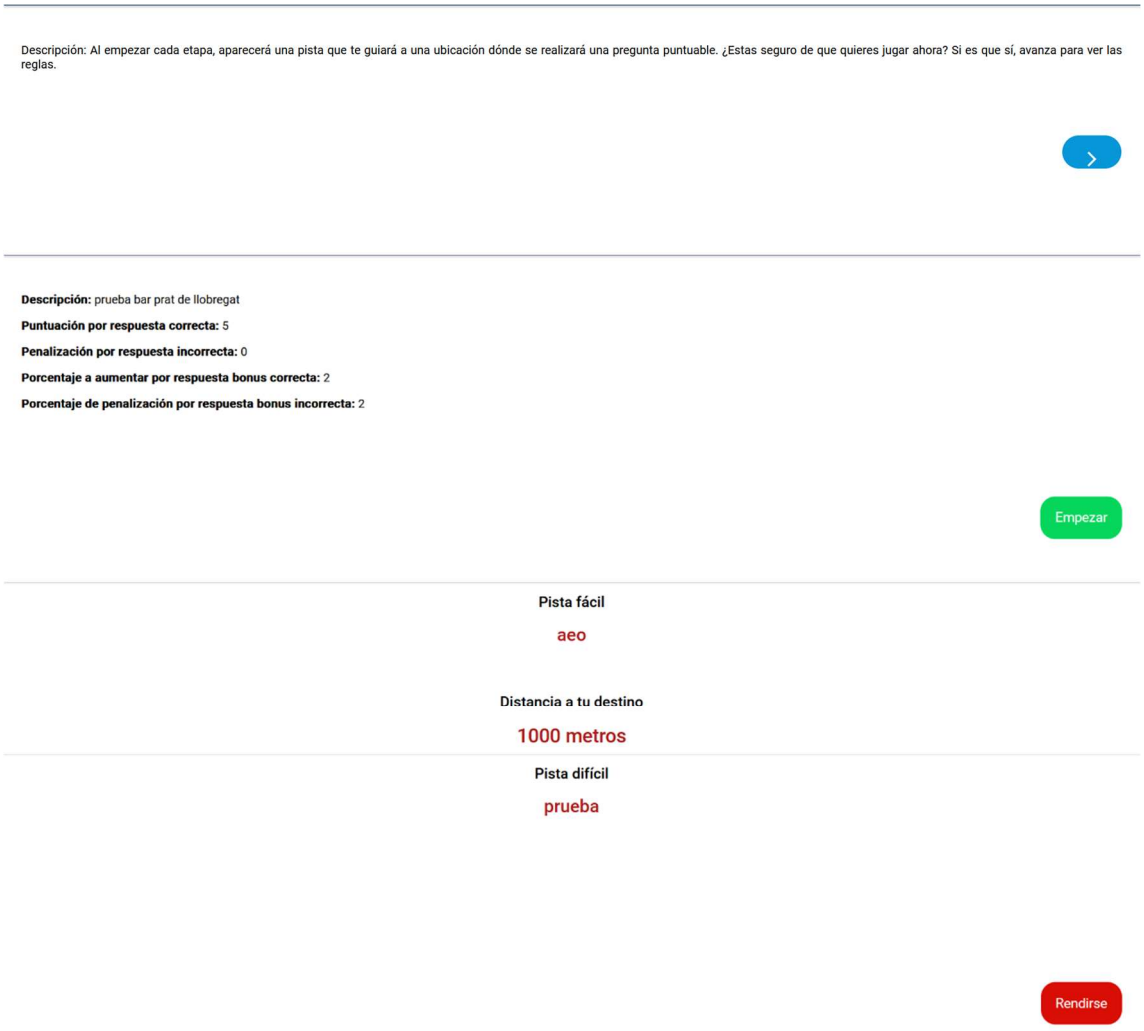

Fig. 2.4: Juego de Geocaching en el móvil del estudiante (Pasos 1-4)

Si el jugador se acerca, tendrá una serie de notificaciones del tipo *popup* en pantalla advirtiéndole cuán cerca está y en el momento que esté en el punto a una distancia muy cercana, se le notifica y se le habilita un botón para Responder Pregunta dando acceso al 4o punto del *stepper*. El siguiente punto se trata de responder las preguntas. Primero, responderá la pregunta básica y en caso de acertarla, el juego le preguntará si quiere responder la pregunta Bonus.

Una vez haya completado el proceso de responder pregunta, el juego le manda al tercer punto del stepper en caso de que hayan más puntos geolocalizables y le muestra los resultados para finalizar del juego en el caso de que no haya más puntos y los haya completado.

2.1.3 Classpip-movil-profesor

En este caso, en la aplicación del profesor no hay nada hecho. Si se intenta clicar sobre un juego activo de *geocaching*, este no responde. En el momento que visualizamos el código del repositorio, se observa que no hay ninguna página o componente del juego de *geocaching* implementada.

2.2 Área de trabajo

Como se ha introducido en este capítulo, en este punto se empiezan a desarrollar pequeñas mejoras para aclimatarnos y hacer, a modo de pequeñas contribuciones, avances en el entorno *Classpip.* En esta situación, hay que hacer una pequeña aclaración que durante estas primeras mejoras y puesta a punto del entorno no habíamos decidido que el juego de *geocaching* sería el plato principal de este trabajo de fin de grado, es por ello que las mejoras aportadas en este punto se podrá apreciar que se suceden en diferentes módulos.

2.2.1 Primeras mejoras

Como ya se comentó anteriormente, el diseño del *dashboard* para el profesor no necesitaba de muchas mejoras más que la implementación de algunos detalles que mejoraran la calidad del producto y así la experiencia de uso del profesor.

Los primeros cambios que realizamos ambos nos fueron asignados por el tutor para empezar a ver como funcionaban los diferentes elementos del *dashboard*:

En el juego de control de trabajo en equipo cada uno de los alumnos deben de enviar una puntuación evaluando a sus compañeros, ésta la recibirá el profesor para poder ver si los alumnos han sido capaces de trabajar bien con dichos compañeros y si se han realizado las tareas equitativamente.

Fig. 2.5: Vista juego Control de Trabajo en Equipo en móvil alumno

En la figura anterior vemos la pantalla que se mostrará al alumno a la hora de responder el control de trabajo en equipo, una vez el alumno decide como repartir sus puntos, la elección se manda al profesor y, una vez todos los alumnos de un grupo hayan mandado sus puntuaciones, el profesor podrá verlas.

Una de las primeras tareas a implementar en la app del *dashboard* fue una herramienta que permitiera al profesor eliminar las votaciones de un determinado alumno, y que éste pueda volverla a realizar, para ello, el primer paso fue estudiar el funcionamiento del proceso de recuento de votos del profesor.

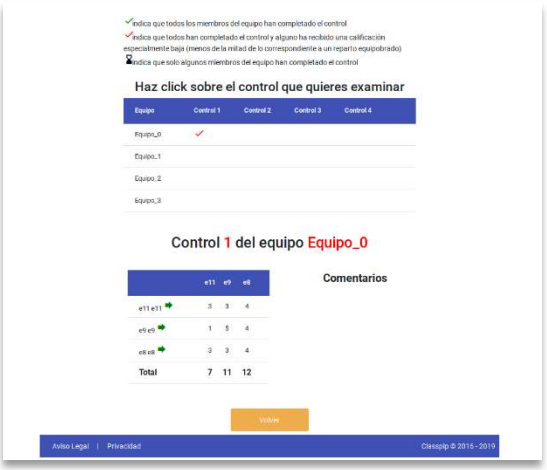

Fig. 2.6: Vista resultados en el *dashboard*

En la figura 2.6, podemos ver como el profesor puede acceder a esa tabla de recuento de los votos en el momento que todos los alumnos de un equipo han

votado, por lo que lo que lo más simple era añadir un botón o algún tipo de seleccionable en cada uno de los votos de los alumnos en un control y éste eliminara el voto, para ello se comenzó añadiendo ese elemento gráfico, como es visible en la figura 2.7.

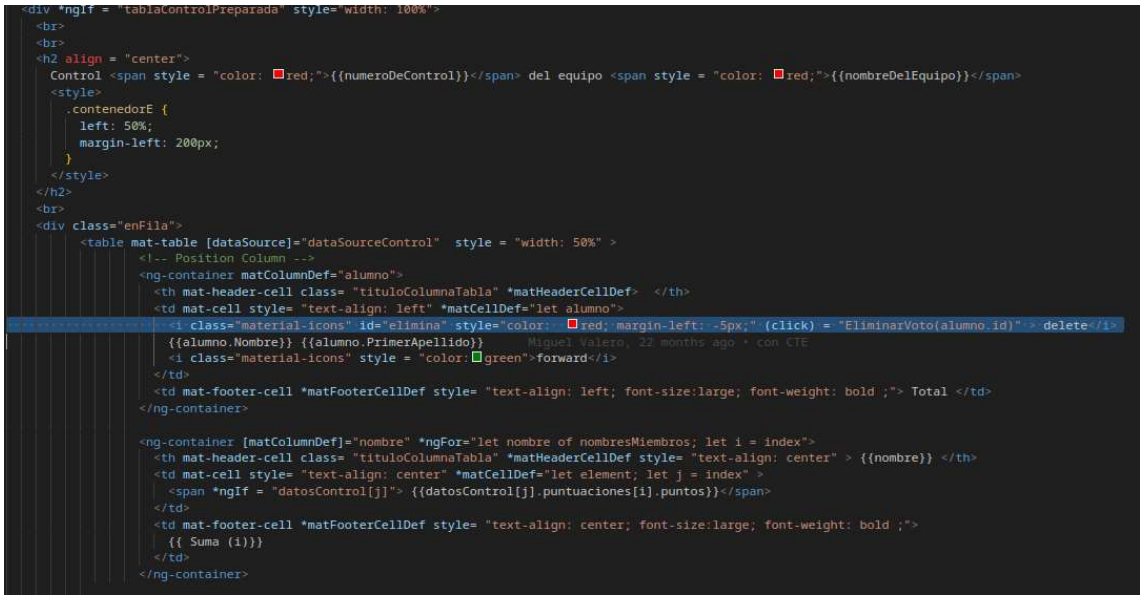

Fig. 2.7: Fragmento de la componente vista donde se muestran las votaciones

Vemos como, en la línea de código destacada en la figura 2.7, se encuentra el elemento en el cual el usuario deberá hacer clic para eliminar el voto, éste es un icono que proporciona la componente de interfaces *Angular Material*, vemos como se le añade la propiedad clic para que ejecute la función *EliminarVoto*, vemos como se le envía el parámetro *alumno.id*, la cual nos permitirá tener localizado cual será el voto a eliminar.

Si observamos la función del Anexo 1, podemos observar que la función se llama *Borra()* en lugar de *EliminarVoto()*, cabe aclarar que en versiones previas el clic del elemento gráfico sí que ejecutaba directamente a la función *EliminarVoto()* pero al comentarlo con el tutor, sugirió que previamente se pidiera al profesor una confirmación de la acción para evitar borrar votos por accidente, por lo que la función *Borra(),* se ha omitido puesto que simplemente ejecuta un pop-up que le consulta al usuario si está seguro de eliminarlo (mediante un aviso *SWAL*, un recurso gráfico que aparece comúnmente en la aplicación). A continuación, veremos cómo funciona la función *Borra()*:

En la primera parte vemos como se procede a localizar el identificador del voto a suprimir (*id*) el cual nos será necesario, puesto que nuestro objetivo es ejecutar una llamada a la API que hace uso de un identificador para eliminarla (*BorrarInscripcionAlumnoJuegoDeControlDeTrabajoEnEquipo*) primero, se trata de encontrar el objeto *Alumno* que ha hecho esa votación, gracias al id del alumno que ha llegado como parámetro a la función se recorren los alumnos que pertenecen al equipo que se estaba mostrando en pantalla y una vez encontremos al alumno que tenga el mismo id que el que nos ha llegado como parámetro procederemos a ejecutar la petición a la api

DameInscripcionAlumnoJuegoDeControlDeTrabajoEn Equipo y guardaremos las inscripciones (votos del alumno) que nos ha proporcionado, al hacer esto, podremos conocer el id de la petición del usuario que queremos borrar para posteriormente ejecutar la petición en la API que la borra.

Habrá un último paso a realizar si queremos que el usuario pueda ser capaz de volver a votar, y es ejecutar la llamada a la API *InscribeAlumnoJuegoDeControlDe TrabajoEnEquipo*, que se realiza al crear los cuestionarios para generar un voto vacío para que el usuario pueda votar.

CAPÍTULO 3: Objetivos y organización

Este capítulo introduce y muestra como nos hemos organizado de una forma efectiva y eficiente para remar en una misma dirección y conseguir los avances para cumplir ciertas cotas dentro del ecosistema *Classpip*.

3.1 Lista de Objetivos

Principalmente, este proyecto nació con el objetivo de mejorar la experiencia del usuario estudiante en el juego de *geocaching*. A raíz de esto, nació una lista de objetivos que se mencionan a continuación:

1. Implementar un mapa para mejorar la selección de puntos al crear un escenario en la aplicación web *dashboard*
2. Implementar nuestro diseño (mapa + componentes del juego) en la app del estudiante
3. Adaptar el código al uso de la librería *leaflet* y entender el uso de mapas y geolocalización dentro del contexto de *Classpip*
4. Implementación de HTTPS
5. Crear una propuesta de trabajo de la app del profesor en función a nuestras aportaciones

Estos objetivos han sido los puntos claves en los que se ha centrado todo el desarrollo y son los que más tiempo nos han requerido. Cada uno de ellos agrupa una cantidad de objetivos más pequeños que en conjunto forman el trabajo que hemos realizado.

3.2 Organización del trabajo

En términos de organización, el trabajo ha sido de codo con codo, hemos desarrollado uno en función del otro y siguiendo una hoja de ruta pactada con el tutor.

Cada dos semanas, realizábamos una reunión donde comentábamos progresos y los pasos a seguir con el tutor, y cada semana nos organizábamos un día para comentar y poner en común los avances de cada uno. A pesar de haber trabajado en implementar distintas cosas, siempre hemos ido desarrollando en el mismo módulo. A continuación, en la siguiente figura 3.1 se puede observar un diagrama de lo que se ha ido trabajando.

Sobre el organigrama, en próximos capítulos se muestra la justificación sobre los tiempos de realización de las tareas que contiene el diagrama. Cabe destacar que desde Octubre del año 2022 hasta Febrero del año 2023 el proyecto sufre cierta ralentización debido a que ambos participantes nos encontrábamos realizando las prácticas. A partir de Febrero de 2023, acabaron nuestras prácticas y tuvimos mucho más tiempo disponible. Eso conjunto con el hecho de que en las primeras fases nos estábamos poniendo a tono con el ecosistema dan pie al resultado de la figura que se muestra en la siguiente página.

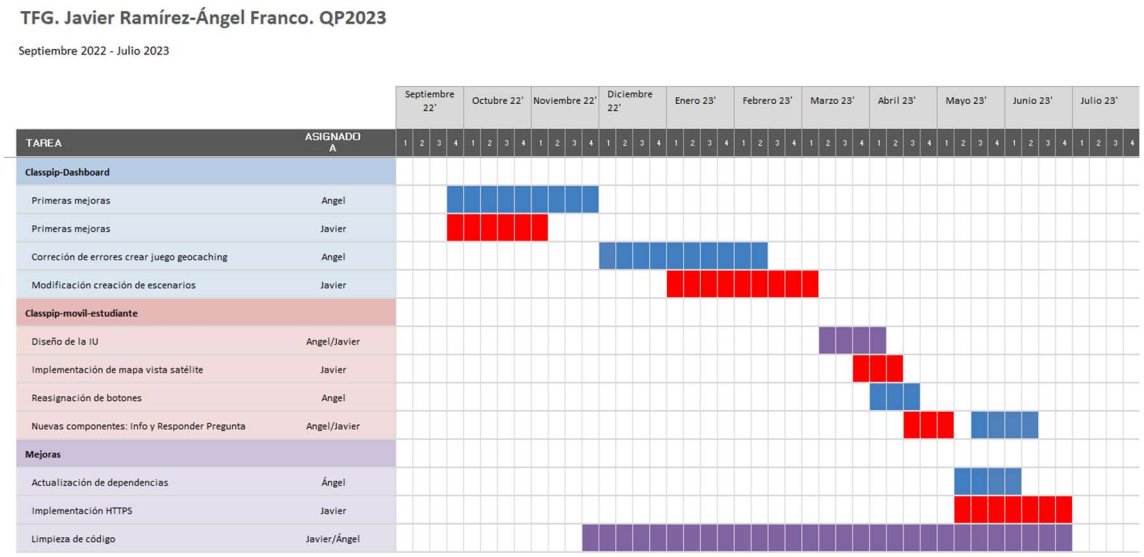

Fig. 3.1: Diagrama de Gantt[12] sobre la organización del proyecto

CAPÍTULO 4: Diseño y redefinición del juego

Este capítulo versará sobre el diseño que se ideó de manera inicial. Este diseño fue ideado entre ambos con una discusión donde el tutor tomó partido. Más adelante, se verán cambios entre nuestro diseño y la implementación final, estos cambios serán también justificados en los siguientes capítulos.

Durante el proceso creativo de cambiar el diseño con un objetivo claro de hacer que la aplicación sea más atractiva y el juego sea más dinámico, se llegó a la conclusión de que el funcionamiento del juego tal y como estaba definido, también debía ser adaptado a este nuevo diseño. De esta manera, la meta es crear este nuevo diseño intentando partir de la base que tenemos, de manera que se pueda reaprovechar parte de la estructura ya existente.

Esto último es una clave que ha sido utilizada como una máxima debido al carácter contributivo y la importancia del trabajo en equipo que muestra *Classpip*. No hay que gastar tiempo en regenerar algo de nuevo, lo importante es mejorar lo existente hasta el máximo punto realizable dentro de las variables características de los entornos de desarrollo de *software* como las capacidades de los equipos humanos o los tiempos de entrega.

4.1 Funcionamiento del juego

A lo largo de este apartado comentaremos cuales son las experiencias de usuario tanto por la parte del profesor como del alumno en el juego de Geocaching.

El juego de Geocaching, como se ha comentado en apartados anteriores (véase la figura 1.2) se basa en la búsqueda de cierta localización en un determinado espacio, *Classpip* implementa una versión de un juego de *Geocaching* adaptada para ser parte de la gamificación de las escuelas, en su caso, el juego de Geocaching se basa en dos mecánicas principales:

- La búsqueda de una localización concreta (designada por el profesor), en la que el alumno contará con una pista que le ayudará a hacerse una idea del lugar al que debe llegar. Es posible que el jugador no sea capaz de encontrar el punto únicamente utilizando la pista designada, lo que le llevaría a bloquearse y puede ser frustrante para un alumno, por lo que el juego permite al profesor introducir también una pista extra. La intención de esta es que sea más esclarecedora que la anterior y haga más evidente cuál es el punto a localizar, aun así, que el alumno recurra a ésta pista extra le supondrá una penalización que se traducirá en la pérdida de un 20% de los puntos que obtendría al acertar la pregunta. Aun utilizando la pista extra, el jugador podría no saber encontrar el objetivo, por lo que cuando el alumno recurre a la opción pista extra no solo recibe la propia pista sino que también se le indica a qué distancia se encuentra del punto

a localizar, esto permite al alumno un último recurso para(de una forma mucho más lenta) buscar el punto tratando de triangular su posición o simplemente mediante el ensayo y error.

- Una vez el alumno haya llegado a la ubicación designada, puede acceder a una pregunta que habrá indicado el profesor en ese lugar, las preguntas son un elemento común en *Classpip* y en el juego de Geocaching se utilizan el mismo tipo de preguntas que los que se usan en los otros juegos, es decir podemos introducir cuatro tipos distintos de preguntas (Cuatro opciones, respuesta abierta, verdadero o falso o emparejamiento). En cuanto el alumno llega al lugar se le muestra un botón "*responder pregunta*" que le manda a ésta. En cuanto el alumno responde se le desbloquea una opción "*responder pregunta bonus*" que le permitirá acceder a otra pregunta en la misma ubicación, la idea es que responder esta otra pregunta suponga asumir un riesgo puesto que fallarla implica perder una cantidad significante de puntos. La pregunta bonus funciona mejorando la puntuación del usuario mediante la siguiente fórmula:

$$P_{punto\ localizable} = P_{pregunta\ básica} + (P_{pregunta\ básica} \cdot P_{pregunta\ bonus} \cdot 0{,}01)$$

Dónde:

- $P_{punto\ geolocalizable}$ es la puntuación final obtenida por el jugador en ese punto.
- $P_{pregunta\ básica}$ es la puntuación obtenida por acertar la pregunta normal.
- $P_{pregunta\ bonus}$ es la puntuación obtenida por acertar la pregunta bonus.

Responder correctamente preguntas aumentará la puntuación total del alumno (en el caso de fallar, si el profesor lo ha considerado así, puede restar y penalizar), cuando el profesor considere oportuno, dará por finalizado el juego y en ese momento, el alumno con más puntuación será el ganador del juego.

Así, el juego de Geocaching supone una herramienta más a la hora de gamificar el ambiente académico utilizando el entorno como un recurso más para el aprendizaje. Para esclarecer algo más el funcionamiento, a continuación se muestra un ejemplo de un caso de uso del juego de *geocaching* a modo de yincana:

Contexto: En una excursión académica, un grupo de alumnos de 1º visita el Zoo de Barcelona.

Creación y preparación del juego (profesor): El profesor procede a crear el escenario que se utilizará en el juego de *geocaching*. Para ello, introduce desde el *dashboard* cuatro puntos localizables diferentes desde el mapa: la zona de los tigres, la zona de los pingüinos, la de los monos y la de los rinocerontes. Para cada uno de ellos ha introducido tanto la pista normal como la extra (esta última hace más evidente dónde está el punto), por ejemplo, para la zona de los pingüinos ha puesto la siguiente pista normal: *"Debes llegar al hábitat donde se encuentra una ave marina, no voladora, que vive principalmente en el hemisferio sur"*. Para la pista extra, el profesor ha introducido lo siguiente: *"El hábitat que buscas contiene un animal comúnmente de pelaje negro y blanco que suele vivir en sitios muy fríos como la Antártida".* Vemos como la pista normal le proporciona una información al alumno que solo le será útil si, por ejemplo, ha atendido en el aula, dónde se le explicó previamente que el pingüino, pese a ser un ave, no puede volar. La pista extra, por otro lado le proporciona una información que es mucho más probable que le haga pensar un pingüino por la imagen que tenemos todos en el pensamiento colectivo.

(Esto da lugar a la dinámica de las posibles opciones a la hora de llegar a un punto localizable, llegar al punto con la pista difícil es lo ideal, pero el alumno deberá tener claros ciertos conceptos para conseguirlo. Encontrarlo con la pista extra debería costarle una cantidad de tiempo menor a costa de una reducción de su puntuación. Y si, ni con la pista extra el usuario puede encontrar el lugar, deberá recurrir a intentar orientarse mediante la información que tiene de la distancia hacia el objetivo, aunque esto, además de hacerle perder puntuación, le consumirá mucho más tiempo y probablemente haga que el resto de alumnos vayan por delante que él y puedan conseguir más puntos en lo que dura el juego).

A continuación, el profesor selecciona las puntuaciones que obtendrán los jugadores al responder las preguntas, por ejemplo, contaremos con que responder una pregunta correctamente proporciona 10 puntos, responder mal restará 5 puntos, las preguntas extras tienen un factor 20 al acertar y 10 al fallar.

Una vez creado el escenario, el profesor asigna el juego de *geocaching* que acaba de crear al grupo de su clase 1ºB (creado previamente). Y clica en el botón de "*activar juego*".

Partida (alumno): Puesto que el profesor ha activado el juego, los alumnos ya deberían de poder jugar al *geocaching*. Los alumnos verán en su móvil una pista normal (por ejemplo, la que se ha indicado anteriormente: *"Debes llegar al hábitat donde se encuentra una ave marina, no voladora, que vive principalmente en el hemisferio sur").* Para ejemplificar el juego vamos a usar dos alumnos diferentes: Anna y Bernat.

- Anna ha decidido no arriesgar utilizando las preguntas bonus e ir exclusivamente a por los puntos y responder las preguntas normales. Al acabar el tiempo de juego, Anna acertó todas las preguntas de los cuatro puntos geolocalizables.
- Bernat optó por una vía mas arriesgada y trató de responder todas las preguntas bonus que pudiera. Acertó las preguntas normales y bonus del primer y segundo punto que encontró, pero a la hora de buscar el tercero tuvo que recurrir a la pista extra para llegar a él, en cuanto llegó acertó la primera pregunta pero falló la extra.

Por lo que, al final sus respectivas puntuaciones serán las siguientes:

- Anna: 40 puntos. (10p por acierto de pregunta normal x 4)
- Bernat: 32 puntos. (10+(10·20·0.01) = 12p por acierto de pregunta normal y bonus x 2) + (10-(10·20·0.01) = 8p por acierto de pregunta normal y fallo en bonus).

Así pues, Anna habría ganado el juego.

4.2 Diseño de la UI

En este apartado se introducirá el nuevo diseño haciendo hincapié en los cambios que hay entre el diseño base del cuál partimos y el diseño que se discutió como punto inicial para las siguientes implementaciones.

4.2.1 Dashboard

Estudiaremos primero el diseño del *Dashboard* en cuanto a los elementos que son necesarios a la hora de crear un juego de *geocaching*, como ya se ha mencionado anteriormente en el apartado 2.1.1, la creación de un escenario para el juego de *geocaching* tiene un diseño muy estándar dentro del resto de menús de la aplicación y utiliza los elementos que ya hemos visto en otros apartados, un "*stepper*"(elemento grafico que actúa como un contenedor de otros elementos que se basa en "*steps*" o "*pasos*"), que los diferentes formularios a rellenar para crear el escenario.

Realmente no encontramos necesario hacer grandes cambios en la interfaz del *dashboard* puesto que era una interfaz bastante funcional, robusta y atractiva pese a su sencillez. Únicamente decidimos realizar algunas mejoras de calidad a la hora de introducir la localización de los puntos en el proceso de *Crear Escenario*, en la pestaña de *Recursos*.

La idea era introducir un botón bajo los campos "*Latitud*" y "*Longitud*" un botón que permita abrir un elemento "*pop-up*" que se despliegue encima del *stepper* que muestre un mapa, éste simplemente serviría para que, en lugar de tener que

recurrir a herramientas externas como *Google Maps* para encontrar el punto geolocalizable, el usuario pueda simplemente hacer clic en el mapa sobre el punto que desee seleccionar, se cierre ese *"pop-up"* y se rellenen automáticamente los campos "*Latitud*" y "*Longitud*" con los valores correspondientes al punto seleccionado.

4.2.2 Móvil estudiante

En este apartado veremos el punto en el que estaba el juego de *geocaching* y que opciones vimos a la hora de cambiar su modo de uso respecto a la versión que aspiramos a conseguir, recordemos que, como hemos podido comprobar en apartados anteriores (véase 1.3.1) *Classpip* contaba con un juego de *Geocaching* completamente funcional pero con un diseño áspero y poco intuitivo.

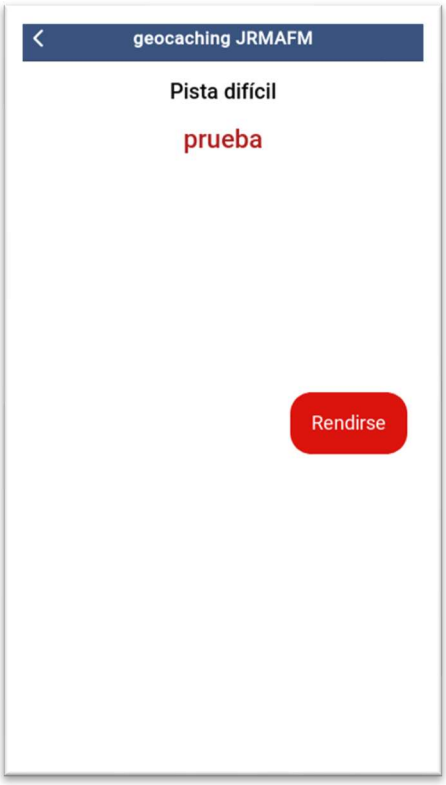

Fig. 4.1: Juego base desplegado en móvil

La figura 4.1 justo anterior muestra la ya comentada anteriormente 2a pantalla del "*stepper*" del juego de *Geocaching*, esta será la que veamos prácticamente durante todo el juego, puesto que será la pantalla que el alumno deberá de utilizar mientras está buscando la localización de los puntos, proceso que normalmente será el que más tiempo lleve al alumno. Consideramos que este diseño no es conveniente para *Classpip* puesto que, teniendo en cuenta que los principales usuarios serán alumnos de colegios, era mejor optar por una interfaz más atractiva y estimulante para ellos, como se puede ver en otros menús de la aplicación (véanse los menús de selección de juegos activos e inactivos, donde estos se muestran en forma de tarjetas que podemos deslizar). Por lo que decidimos que habría que cambiar la idea de utilizar un "*stepper*",el cual le daba

al juego un aire de “formulario a rellenar” o lo hacía parecer mas bien un menú que un juego en sí. En su lugar quisimos optar por un diseño que pusiera el foco en un elemento principal, el mapa. Quisimos que el mapa que indica la posición actual del alumno ocupara el máximo de espacio posible y que sea un elemento persistente en la pantalla del juego.

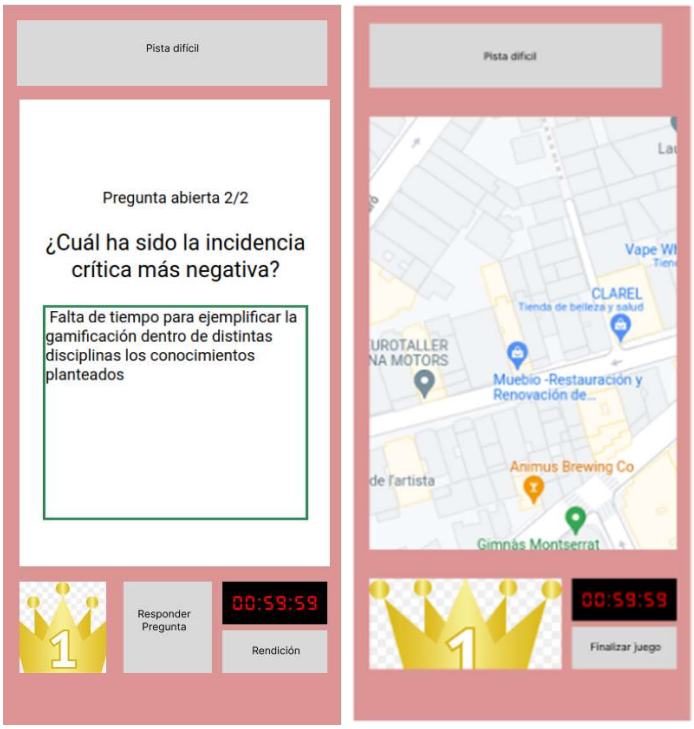

Fig. 4.2[13]: Aplicación vista alumno respondiendo una pregunta (izquierda) y aplicación vista profesor con el mapa (derecha)

En la figura 4.2 se presentan los primeros *mockups* que se desarrollaron en la primera etapa del cambio de la IU. Un *mockup* se traduce como un prototipo de diseño a partir del cual se muestra el producto o la pauta de diseño a seguir. Estas maquetas se generaron a través de la herramienta de diseño web, *Figma*. Queríamos que nuestra interfaz (en la pantalla principal del juego) constara de tres grandes bloques:

- El primero de ellos, el ya comentado mapa. El juego muestra un gran *display* de un mapa con un marcador que indica la posición del alumno y está centrado en ésta, el mapa sirve para que el alumno se pueda hacer una idea del espacio en el que se encuentra y que zonas tiene en sus alrededores (por ejemplo, si la pista indica que el punto está en un parque, el alumno puede ver los parques que están a su alrededor y le podría ayudar).

- En la parte superior de la pantalla, se mostrará una “tarjeta” que contendrá la pista proporcionada por el profesor, este bloque no debería ocupar más de un 20% de la pantalla y se mostrará en todo momento mientras el jugador trata de localizar el punto, para acceder a la pista extra, el alumno podrá pulsar el mismo recuadro de la pista (evitando así tener un botón únicamente para eso, haciendo que el diseño se valga de

menos elementos y sea mínimo) y al hacerlo se le mostrará un aviso para indicarle que su puntuación se reducirá al recurrir a la pista extra, si el alumno acepta, la carta de la pista cambiará y le mostrará la nueva pista y el indicador de distancia hacia el objetivo. La tarjeta cambia de color cuando está en "modo" pista extra para que visualmente sea más evidente y pasado un rato de activarla, el usuario pueda pensar que aún no la ha activado.

- En la parte inferior de la pantalla habrá otra sección (ésta con algo más de peso que la sección superior, con alrededor de un 25% de la pantalla, para así hacer que el mapa tenga un protagonismo mayor en el diseño y ocupe más de la mitad de la interfaz) que contendrá diferentes elementos:

  - Un apartado que muestra directamente la posición del jugador en la tabla de puntuaciones, la idea es que, por ejemplo, si el jugador está en primera posición y cambia su posición, se actualice ese valor. Por otro lado, este apartado no es meramente un *display* de únicamente la posición, el alumno puede pulsar en la sección para que se muestre un "*pop-up*" que contiene el ranking completo, es decir, una tabla que muestra el nombre del resto de alumnos que participan en el juego, así como su puntuación en tiempo real y la etapa en la que se encuentran (esta etapa no es más que el punto geolocalizable en el que se encuentran y da una idea de cuantos puntos geolocalizables ha resuelto cada jugador), ordenados según su puntuación.

  - Un *display* muestra un contador con el tiempo restante del juego, en cuanto llegue a cero, el juego habrá acabado, es importante que el jugador tenga en cuenta este contador para tomar la decisión en cierto momento de seguir buscando el siguiente punto o responder la pregunta bonus. Esto se establece en un principio de diseño con la idea de converger el juego de *geocaching* con una modalidad de examen tradicional.

  - Finalmente, habrá un botón de finalizar el juego para que el usuario pueda abandonar la partida en caso de que se vea imposibilitado para continuar por cualquier factor, en ese caso, su puntuación se reduce a 0, y se indica en el ranking dando un valor a su puntuación de 0,1.

Cabe añadir que en un inicio se hizo un diseño para la aplicación del profesor como se ha podido ver en la figura 4.2. Este diseño se referenciará de cara al diseño final en una propuesta de mejora para futuras contribuciones en un próximo capítulo. Consideramos que debido a la nueva interfaz y redefinición del juego, sería correcto, además que daría lugar a una cantidad de mejoras nuevas en consonancia con esta aplicación para los alumnos.

CAPÍTULO 5: Tecnologías

Antes de seguir avanzando en detalles más técnicos, es preciso hablar del tipo de tecnologías que vamos a utilizar. En el primer capítulo se vieron las tecnologías que usa el entorno *Classpip*. En este capítulo se detallarán cuáles serán trabajadas en las próximas implementaciones.

La herramienta principal que hemos usado ha sido *Visual Studio Code*, y para el control de versiones y aportaciones hemos trabajado con repositorios en *GitHub*. Se destaca también, como herramienta externa para generar el diseño inicial, el uso de la herramienta web *Figma*.

Principalmente, se ha trabajado sobre *Ionic* y *Angular*, en la parte móvil y en la aplicación de escritorio respectivamente.

Ambos *frameworks* trabajan de una manera muy similar, sobre todo debido a qué como ya se ha introducido en el primer capítulo, *Ionic* es un *framework* basado en *Angular.* Por ello explicaremos ahora como funciona *Angular* y como ha sido utilizado. Esta explicación se puede exportar y aplicar también a *Ionic*.

Angular es un *framework* que funciona a través de componentes. Estas componentes pueden tener diferentes salidas y verse en diferentes formatos de cara al usuario final, esto hace referencia a que una componente puede ser una página web entera y dentro de esta, contener más componentes que contengan información dentro de la página.

Si se genera una componente a través de consola, lo que se obtiene es una carpeta con 3 archivos principales. A través de un ejemplo, esto será más entendible, partiremos de que creamos una componente llamada *componente-ejemplo*:

- *componente-ejemplo.component.ts*: *Angular* (y por lo tanto, *Ionic*) están basados en TS. Este archivo será el que contenga toda la lógica necesaria para funcionar.

- *componente-ejemplo.component.html*: Esta componente será la vista del usuario sobre lo que se quiera mostrar y con lo que va a interactuar.

- *componente-ejemplo.component.scss*: Esta componente se puede utilizar para actualizar y modificar estilos y elementos de la componente de vista, la componente HTML.

Cabe destacar, que se pueden generar más archivos que puedan ser necesarios como, siguiendo el ejemplo, podría ser *componente-ejemplo.module.ts*. Este archivo registra como módulo el componente creado, necesario debido a la naturaleza de *Angular* al trabajar sobre módulos. Los módulos a su vez no dejan de ser agrupaciones de componentes y recursos relacionados con el objetivo de mejorar la organización y facilitar la reutilización de código.

Para ilustrar este ejemplo, a continuación se pueden ver las figuras 5.1 y 5.2 que contendrán gráficamente las explicaciones anteriores. En esas imágenes se puede ver lo que sucede cuando generamos una componente.
Para generar una componente se hace uso de la siguiente línea a través de la consola:

ng generate component componente-ejemplo

Dentro de la carpeta de *app*, se generará lo que se puede ver en la figura 5.1 a continuación:

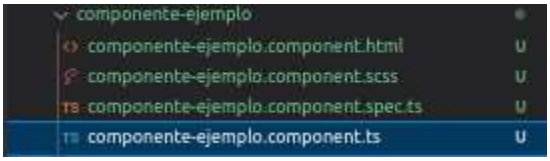

Fig. 5.1: Ficheros del componente *componente-ejemplo*

Y en la imagen siguiente, la figura 5.2, muestra una captura de pantalla de las tres componentes principales de un componente:

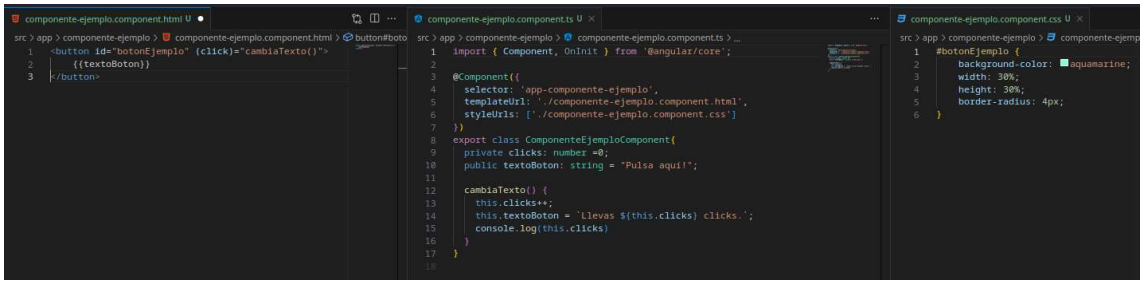

Fig. 5.2: Componentes HTML, TS y SCSS (de izq. a der.) del *componente-ejemplo*

Se puede ver, en la figura 5.1, un archivo del que no hemos hablado, *componente-ejemplo.component.spec.ts*, este archivo se genera al ejecutar la generación de esta componente y es un archivo de test y se utiliza para evaluar el comportamiento y la funcionalidad del componente que se acaba de generar.

Así se podría resumir el funcionamiento de *Angular*, y la parte de interés relevante para próximos apartados. El desarrollo de este proyecto se centra en la modificación, creación y redistribución de diferentes componentes para conseguir los objetivos propuestos.

CAPÍTULO 6: Implementación

Este capítulo tratará en cómo hemos implementado el diseño y como lo hemos adaptado a las expectativas discutidas en el cuarto capítulo sobre el diseño de las interfaces. En este apartado entraremos en un punto mucho más técnico sobre lo que se ha aportado. De cara a comprender como se ha llegado a la vista final del usuario de las aplicaciones del *dashboard* y la app móvil del estudiante, es necesario reparar en este capítulo tanto en este informe como en el del compañero.

6.1 Dashboard

En la aplicación web, siguiendo el hilo del anterior capitulo, no se vió necesaria cambiar o introducir ningún diseño nuevo para la creación de juegos o de recursos.

6.1.1 Implementación del mapa

Se decidió integrar alguna interfaz gráfica que nos permitiera, en lugar de tener que rellenar los dos *inputs* con las coordenadas del punto, hacer el proceso más simple teniendo que hacer clic en un mapa. La idea era implementar un botón bajo los *inputs* de las coordenadas (no queríamos eliminar esos campos, pues en algunos casos concretos podían resultar útiles para los usuarios) y que desplegara algún tipo de página o *pop-up* con el mapa. Allí, el usuario puede hacer clic en un punto del mapa y, desde ese componente, cuando clique un botón "*aceptar"*, se cierre la página o mapa y se actualicen los campos de latitud y longitud con los valores correspondientes al punto seleccionado en el mapa.

Para añadir el mapa, sería necesario utilizar un contenedor que nos permita cargar y mostrar la información que nos otorgaría una API de Base de Datos espaciales. Por lo que, la primera decisión a tomar fue elegir que alternativa queríamos utilizar para esto, en nuestro caso, decidimos utilizar las siguentes opciones:

- ***Leaflet:*** Para la implementación del contenedor del mapa decidimos utilizar *Leaflet*, la principal biblioteca JavaScript de código abierto para mapas interactivos compatibles con dispositivos móviles, permite facilitar el *display* de mapas por casillas (*tiled based maps*), *Leaflet* está diseñada poniendo el foco en el rendimiento y en la facilidad para utilizarse. Es una herramienta muy útil también gracias al hecho de que funciona y es muy fácilmente adaptable a la mayoría de plataformas móviles y de escritorio.

- ***OpenStreetMap***: Por la parte de la API a la que realizar las peticiones para la información de los mapas decidimos optar por *OpenStreetMap* un proyecto colaborativo de código abierto que permite crear mapas editables y gratuitos. Decidimos elegir esta ya que preferíamos utilizar tecnologías de código abierto en lugar de recurrir a sistemas propietarios

cerrados, además el uso de las APIs de *Google Maps* puede suponer cargos económicos si la aplicación utiliza un tráfico importante de peticiones.

Una vez decididas las tecnologías a utilizar, decidimos implementar el uso de los mapas mediante *pop-ups* que se desplegaran encima del formulario de creación de los escenarios, así el usuario no perdería el foco en la creación del recurso, y haría más cómodo el uso de la herramienta si ha de crear un numero elevado de puntos o si ha de rectificar varias veces a la hora de crearlos. El *pop-up* se ejecutará pasando como parámetro un vector de dos elementos de tipo *number* que corresponderá a las coordenadas de latitud y longitud del punto seleccionado, así pues, el *pop-up* mostrará una nueva componente *map-popup.component* ésta únicamente incluirá el contenedor del mapa y un botón de *aceptar* y cerrar el *pop-up*.

Dentro de la componente *map-popup.component,* encontraremos la siguiente función (véase Anexo 2: *initMap()*)

Esta será la función que inicializará el mapa en el *pop-up*. En la componente HTML tendremos un elemento <*div*> con el "id= map", vemos como en la segunda línea del fragmento se referencia a este id para implementar el mapa en ese <*div*>. Como en esta herramienta no se precisará de la ubicación GPS del dispositivo que este utilizando el usuario (en la mayoría de los casos, es posible que el profesor no se encuentre en el mismo espacio donde realizará el juego), se indica explícitamente que el mapa se centre en un punto en concreto, por ahora, se ha indicado que se centre en el Parc Mediterrani de la Tecnologia, observamos también como se introduce en la función *L.tileLayer()* la *url* de la API de *OpenStreetMap*.

Si se estudia el código de la función *selectLocation()* (en detalle en el Anexo 2: *selectLocation()*) se puede observar que en el fragmento de código, en cuanto un usuario hace clic en el mapa, se ejecuta la función *selectLocation()*, la cual añade un marcador al mapa en la posición en la que el usuario ha hecho click y guarda en la variable this.data (que es el vector de números que le ha llegado de la componente principal) las coordenadas del punto con 4 decimales; si es el primer punto que selecciona el usuario, *leaflet* generará el marcador *L.marker().addTo.(this.map)*, dándole los atributos del icono a utilizar.

Finalmente, tenemos la función *onNoClick()* (véase el Anexo 2: *onNoClick()*) que se ejecutará si el usuario hace clic en el botón de salir del mapa, se puede ver como la función devolverá al componente principal las coordenadas que se han almacenado en la variable this.data, en la posición [0] tendremos la latitud y en la [1] la longitud.

El proceso de indicar la posición de un punto geolocalizable en el escenario queda de la siguiente forma, como se puede ver, continua el formulario original pero se añade un botón con el que desplegar el mapa y que el usuario haga clic en la ubicación de interés. En las figuras 6.1 y 6.2 vistas a continuación es visible la anterior explicación.

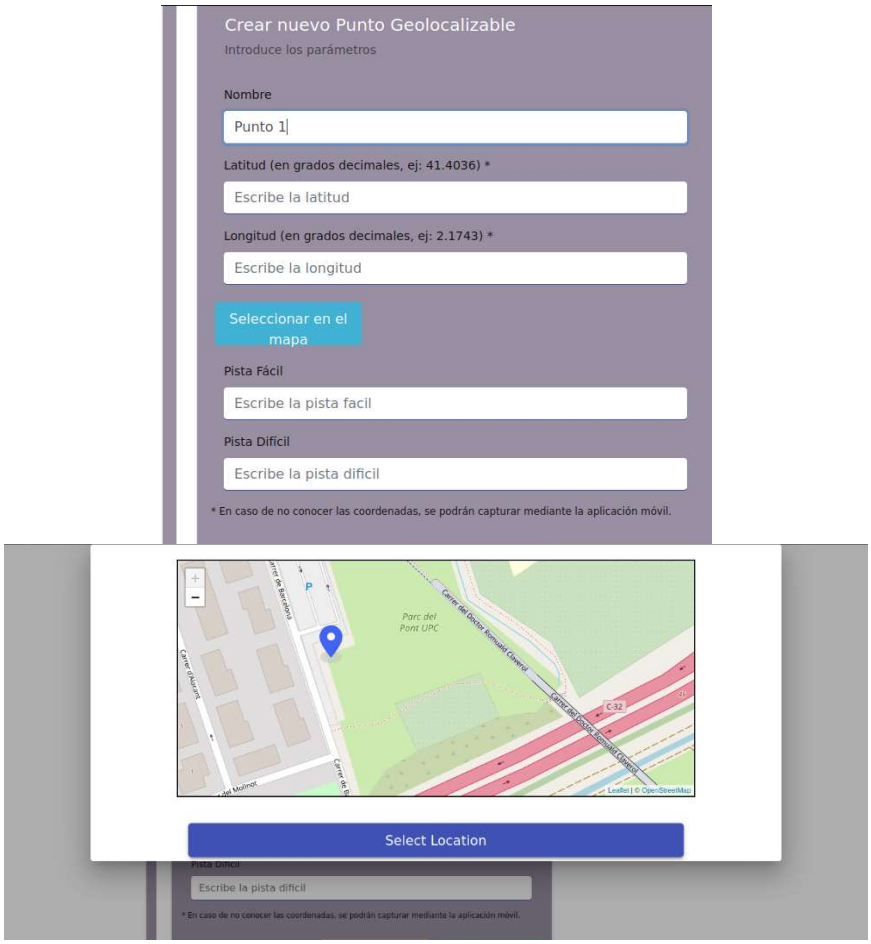

Fig. 6.1 y 6.2: Proceso de creación de punto geolocalizable con el mapa integrado

6.2 Móvil estudiante

En la versión de *Classpip* para los estudiantes de *Ionic* se han realizado otra serie de cambios respecto al juego de *geocaching*, en este subapartado se explicarán cuáles han sido los principales cambios a destacar, siguiendo con los puntos comentados en el anterior apartado del diseño a implementar en la versión móvil, se procedió primero a añadir una vista de mapa en la aplicación como el elemento principal en la pantalla del juego de *geocaching*.

6.2.1 Implementación del mapa

Para la aplicación móvil también se optó por añadir una interfaz basada en *leaflet* y, en un principio, utilizar la misma API de *OpenStreetMap*. Al igual que en el *Dashboard*, se comenzó instalando la librería *Leaflet* al proyecto y, después de solucionar algunos problemas de incompatibilidades generados al implementar la nueva librería, se procedió a añadir el mapa de una forma similar, creando el * * con id= map en el que *Leaflet* mostrará el mapa cargado que reciba de la API.

Para ello, se crea y utiliza la función *initMap()*. Vemos como esta funcion es bastante similar a la implementada a la aplicacion de *Angular* del *dashboard*, aun así, existen algunas diferencias a comentar:

- Se ha dejado como *false* la opción *zoomControl*: Esta opción desactiva los controles de zoom que aparecen en la esquina superior izquierda del mapa. Creímos que, puesto que el juego de *geocaching* se ejecutará en un *smartphone*, el usuario tiene la tendencia a controlar los niveles de zoom haciendo el gesto de pinza y no pulsando en el control del zoom, quitar estos controles nos proporcionaba un diseño más limpio y unificado.

- Cambio en la API de mapas: Se puede observar como ya no se está utilizando la API de OpenStreetMap para la información de los mapas, en las primeras versiones con el mapa implementado en la aplicación de estudiante se siguió utilizando OSM pero nos surgió el problema de que OpenStreetMap no ofrecía mapas con visión de satélite por lo que el usuario vería un mapa que quizás no era tan conveniente en el uso que le queríamos dar, teniendo en cuenta el problema de que Google Maps puede acabar haciendo cargos por el uso de su API, decidimos buscar otra alternativa, Miguel nos recomendó utilizar *Mapbox*, otro proveedor de mapas online, pese a que algunas herramientas que ofrece son de código cerrado, "*Mapbox* abraza con orgullo sus raíces de código abierto. Trabaja abiertamente y libera la mayor cantidad de código posible." Según declaran en su página web. Vemos en la URL de la api como *Mapbox* ofrece mapas satelitales de forma gratuita.

Como en esta implementación sí que requerimos de la localización por GPS. Esto se consigue a través de la siguiente implementación:

```
navigator.geolocation.getCurrentPosition(position => {
  console.log("entro en el getcurrentposition")
  this.lat = position.coords.latitude;
  this.lng = position.coords.longitude;
  console.log(`Estas en ${this.lat}, ${this.lng}`);
  this.setLocation();
});
```

Fig. 6.3: Obtención de la ubicación del jugador

Esta al ejecutarse dará acceso a la web para acceder a la localización del dispositivo, y una vez obtenida, guardará los valores de latitud y longitud del usuario como variables, al actualizar estos valores, se ejecutará la función *setLocation()* que crea un nuevo marcador, en el mapa y, al hacerlo, centra el mapa en ese punto.

La posición que obtenemos en el código anterior se actualiza a través de la función *.watchPosition()*, que se verá en la siguiente figura 6.4. De esta manera, con esta última modificación, la ubicación se transforma en un activo constante y cambiante.

```
this.identificador = navigator.geolocation.watchPosition((position) => {
  this.lat =  position.coords.latitude;
  this.lng =  position.coords.longitude;
  console.log('latitud ' + this.lat);
console.log('longitud ' + this.lng );
},(positionError) => {
  console.log(positionError)
},{
  enableHighAccuracy: true
});
```

Fig. 6.4: *.watchPosition()*

6.2.2 Implementación de la componente de información

Al haber implementado el mapa en *leaflet*, decidimos finalmente eliminar el *stepper* de la interfaz del juego como ya comentamos anteriormente que teníamos pensado en el momento que decidimos el nuevo diseño del juego. Al dejar de lado el *stepper*, había elementos que incluía que hubo que hacer de nuevo, el primero que decidimos renovar era el primer paso del *stepper* que contenía una explicación del funcionamiento del juego para los alumnos, y el segundo, que informaba de las puntuaciones que el profesor había asignado al usuario por acertar y fallar las preguntas tanto normales como extras. Para ello, estuvimos barajando cuales eran las diferentes opciones que teníamos a la hora de renovar esas pantallas, finalmente, nos decidimos por lo siguiente.

Implementar una nueva componente (*juego-de-geocaching-info.component*) que contendrá la información que se mostraba en los dos primeros pasos del *stepper*. Esto nos servirá para poder tener toda la explicación, y datos del juego que mostrar al usuario en un mismo elemento para poder manejarlo con más facilidad. Además, se incluye también un nuevo apartado en esta componente que explica cómo funciona la interfaz por si algún jugador se pierde con el menú y no sabe cómo jugar al juego.

Ahora, una vez tenemos esa componente, decidimos que la mejor forma de implementar la información del juego en la pantalla del juego de *geocaching*, añadiendo a continuación un botón de "*Comenzar*". La idea es que el mapa tenga de inicio el atributo *display* en valor *none*, para que permanezca desactivado, cuando el usuario hace clic en el botón "*Comenzar*" se intercambiará este atributo y la componente de información tendrá el estilo *display=none* mientras que el mapa tendrá *display= block*. Además de ese botón, también hay dos botones más para navegar entre los diferentes *steps*, que son los botones *Siguiente* y *Atrás*.

Además, como consideramos que era posible que un alumno quisiera consultar las asignaciones de las puntuaciones en cualquier momento o se pierda una vez haya comenzado el juego, decidimos introducir también un botón que se muestre en un lateral durante todo el juego para que se puedan consultar en cualquier momento.

Haciendo uso de *ionic lab*, en la figura 6.5 se puede ver el aspecto de la aplicación con el mapa y la componente de información.

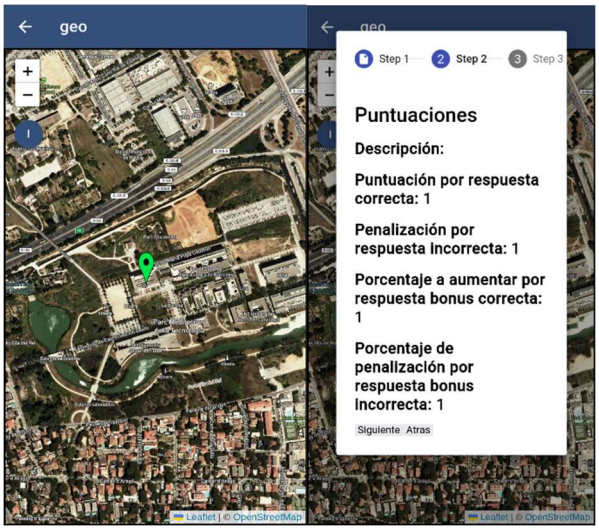

Fig. 6.5: Aspecto de la componente de información en una versión beta

6.2.3 Implementación del ranking

Ángel implementó en el código las funciones del ranking que permitía al usuario ver la tabla de puntuaciones para que el jugador pueda ver los puntos de el resto de jugadores (véase el apartado 6.2.2 de la memoria del compañero). Y posteriormente, hubo que hacerla accesible al jugador mediante la interfaz, para ello recurrimos al uso de un botón y de un *pop-up* de forma similar a los de la componente de información. En la componente HTML de la nueva componente para el ranking (*juego-de-geocaching-ranking.component*) se implementa la tabla.

Como se puede ver en la figura 6.6, y en el siguiente apartado, a partir del uso de las componentes CSS, mejoré los estilos de los diferentes para que tuviesen un diseño similar.

```
.tabla {
    margin: 0%;
    border-collapse: collapse;
    box-shadow: 0 0 20px rgba(0,0,0,0.15);
}
tr :hover {
    background-color: rgb(88, 88, 88);
    box-shadow: 0 0 11px rgba(33,33,33,.2);
}
.tabla thead tr {
    background-color: #233553;
    color: #ffffff;
    text-align: left;
}
.tabla th,
.tabla td {
    padding: 12px 15px;
}
.tabla tbody tr {
    border-bottom: 1px solid #dddddd;
}
```

Fig. 6.6: Fragmento de modificación del estilo de la tabla

6.2.4 Mejoras y estilización en la IU

Se volvieron a diseñar de cero los elementos que contienen las pistas que ve el usuario. Estas ahora utilizan mejor el espacio y pese a ser un elemento que destaca en la interfaz, no ocupan mucho espacio para permitirnos que el usuario ponga el foco en el mapa cuando esté jugando al juego de *geocaching*. Ahora el hacer clic a la pista revelará la pista extra, se mantiene el aviso que notifica al usuario que el utilizar la pista seguramente le hará perder puntos, una vez realizados los cambios, la interfaz de la pista fue cambiada y las pistas difícil y fácil quedan tal cuál muestran las figuras 6.7 y 6.8:

Fig. 6.7: Implementación de la pista difícil

Fig. 6.8: Implementación de la pista fácil

CAPÍTULO 7: Obstáculos y Mejoras

En conjunto con el anterior capítulo y los siguientes, se ha pretendido generar una diferenciación entre las mejoras. Esto se decidió debido a que no todas las contribuciones tenían el mismo carácter o valor. En el capítulo anterior se tratan las mejoras que modifican las funciones y la interfaz de las aplicaciones que hemos trabajado. En este capítulo se comentarán dos bloques que se han estudiado y son necesarios para el correcto funcionamiento del proyecto.

Este capítulo, en definitiva, se centra en mejoras que a pesar de que han sido estudiadas y trabajadas sobre los módulos de desarrollo de este proyecto, se podrían ampliar a otros bloques del entorno de *Classpip*. Sobre todo se aprovecha este apartado para comentar mejoras que han sido necesarias debido a las aportaciones que han sido realizadas.

7.1 Implementación de Modo Seguro (HTTPS)

En este apartado abordaremos la importancia de implementar el protocolo HTTPS en el entorno *Classpip*. Se explorará que es HTTPS, lo fundamental de su implementación y como se puede incluir en este contexto[16].

Teniendo en cuenta el contexto en el que nos encontramos, donde los dispositivos móviles ocuparán la mayor parte del tráfico de la aplicación, el protocolo HTTPS (*Hypertext Transfer Protocol Secure*) es un elemento fundamental para proteger la transmisión de datos.

HTTPS nos proporciona una capa adicional de seguridad gracias al cifrado de la información transmitida. Lo que implica que la comunicación entre el cliente y el servidor que aloje la aplicación web se realiza de forma segura, protegiendo así los datos sensibles de los usuarios.

La protección de los datos ha cobrado una mayor importancia en consecuencia de los avances tecnológicos, que han venido ligados a un aumento cada vez mayor de las amenazas cibernéticas. En el caso específico de las aplicaciones móviles que hacen uso de servicios de ubicación, los datos son especialmente delicados. En el caso de *Classpip*, en el juego de *geocaching* se transmiten ubicaciones donde se encontrarán los alumnos, siendo informaciones altamente sensibles, la exposición a terceros de estos pueden tener graves consecuencias en términos de privacidad y seguridad.

Es por esto que es realmente importante implementar el protocolo HTTPS en aplicaciones que, como esta, utilizan servicios de información personal. Al emplear HTTPS, los datos transmitidos entre el dispositivo del usuario final y el servidor, se encriptan, evitando así que personas no autorizadas puedan interceptar y acceder a información confidencial (utilizando, por ejemplo, ataques *Man-in-the-Middle*). Esto hace que los datos sean confidenciales y evita posibles filtraciones o robo de datos.

HTTPS también implementa técnicas de firma digital para asegurar que los datos se transmiten de forma íntegra[17], es decir, se verifica que los datos no hayan sido modificados o alterados en la transmisión. En el contexto que nos concierne, esto garantiza la precisión y fiabilidad de las informaciones de ubicación, esencial para brindar la experiencia de usuario óptima. Otro aspecto importante que implementa HTTPS es su capacidad para autenticar la identidad del servidor, mediante el uso de certificados digitales emitidos por entidades confiables (autoridades de certificación, a partir de ahora, llamadas comúnmente CA), de esta manera se permite verificar la autenticidad del servidor y prevenir ataques de suplantación. Evitando posibles ataques *phishing* (ataques basados en hacerse pasar por una persona, empresa o servicio de confianza) para garantizar la seguridad e integridad de los datos.

7.1.1 Funcionamiento HTTPS

El funcionamiento de HTTPS se basa en el uso de certificados SSL/TLS (*Secure Sockets Layer/Transport Layer Security*) emitidos por las autoridades de certificación. Cuando un cliente (en este caso, un dispositivo móvil o un ordenador), realiza una solicitud a un servidor web mediante HTTPS, se establece una conexión segura mediante un proceso denominado “apretón de manos” (o *handshake*). Durante ese apretón de manos, se negocian y se intercambian claves criptográficas para establecer una comunicación segura.

Con la ayuda del diagrama de secuencia de la figura 6.3, se presenta el proceso de *handshake* de HTTPS que está basado en los siguientes pasos[19]:

- Mensaje de “hola” del cliente (1): El cliente da inicio a la comunicación al enviar un mensaje *“hola”* al servidor. En ese mensaje se incluye la versión de TLS que admite el cliente, los conjuntos de cifrado compatibles y una cadena de *bytes* conocida como *“cliente aleatorio”*.

- Mensaje de “hola” del servidor (2): En respuesta al “hola” enviado por el cliente, el servidor envía otro mensaje que contiene el certificado SSL del servidor, el conjunto de cifrado que ha elegido de los compatibles y el *“servidor aleatorio”,* otra cadena de *bytes* que genera el servidor.

- Autenticación (3): El cliente verifica el certificado SSL del servidor con la autoridad certificadora que lo emitió. Esto asegura que el servidor sea legítimo y que el cliente esté interactuando con el propietario correcto del dominio.

- Secreto *premaster* (4): El cliente envía otra secuencia de *bytes* aleatorios llamada "secreto *premaster*". Este secreto *premaster* se cifra con la clave pública del servidor y solo puede descifrarse con la clave privada correspondiente. El cliente obtiene la clave pública del certificado SSL del servidor.

- Uso de la clave privada (5): El servidor descifra el secreto *premaster* utilizando su clave privada.

- Creación de claves de sesión (6): Tanto el cliente como el servidor generan claves de sesión utilizando el cliente aleatorio, el servidor aleatorio y el secreto *premaster*. Ambas partes deben obtener los mismos resultados.

- Cliente listo (7): El cliente envía un mensaje cifrado con una clave de sesión llamado "terminado" para indicar que está listo para la comunicación segura.

- Servidor listo (8): El servidor responde con un mensaje cifrado con una clave de sesión llamado "terminado" para indicar que también está listo para la comunicación segura.

(Éste proceso puede ser algo diferente según los métodos de encriptado, por ejemplo, si se utiliza el protocolo Diffie-Hellman se añaden mensajes para calcular las firmas digitales y el intercambio del parámetro DH del cliente).

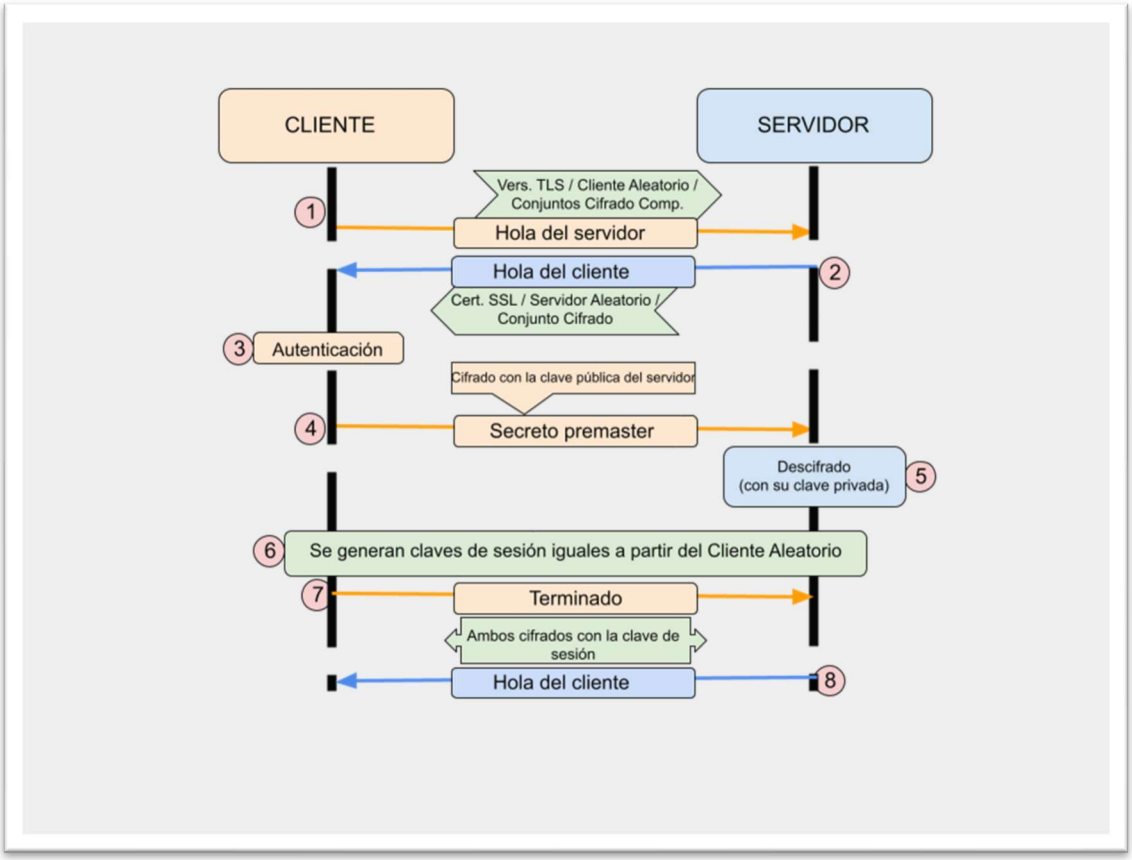

Fig. 7.1: Diagrama de secuencia de un *handshake* HTTPS

Una vez completado el proceso, la conexión se considera segura, ahora, los datos transmitidos en la conexión segura entre cliente y servidor se cifran utilizando las claves de sesión, esto implica que la información tendrá un formato ilegible para cualquier tercero no autorizado que pueda intentar interceptarla.

Además del cifrado de extremo a extremo, HTTPS también garantiza la integridad de los datos transmitidos. Para ello se pueden utilizar varios métodos[20]:

- Hashes criptográficos: HTTPS utiliza funciones de *hash* como SHA-256 para calcular un resumen de los datos transmitidos, el resumen o *hash* es una cadena de caracteres que se genera a partir de los datos y tiene una longitud fija[21]. Cualquier cambio en los datos producirá un *hash* completamente diferente. El cliente y el servidor comparan los hashes calculados para verificar que sean idénticos. Si los hashes coinciden, significa que los datos no han sido modificados.

- Cifrado simétrico: HTTPS utiliza algoritmos de cifrado simétrico para proteger la confidencialidad de los datos transmitidos. Estos algoritmos utilizan una clave compartida entre el cliente y el servidor para cifrar y descifrar los datos. Si un mensaje es modificado durante la transmisión, el descifrado con la clave compartida generará un resultado diferente al esperado, lo que indica que el mensaje ha sido alterado.

- Protocolo de autenticación de mensajes (HMAC): HTTPS también puede utilizar HMAC para verificar la integridad de los mensajes. HMAC es una construcción criptográfica que combina una función *hash* y una clave secreta para generar un valor de autenticación. Tanto el cliente como el servidor utilizan la misma clave secreta para calcular y verificar el HMAC de los mensajes. Si un mensaje ha sido modificado, el valor de autenticación generado será diferente al esperado, lo que indica una alteración.

Por último, otro aspecto clave de HTTPS es la autenticación del servidor, los cuales tienen un papel fundamental en el establecimiento de conexiones seguras y confiables entre un cliente y un servidor, como se ha mencionado anteriormente, están firmados digitalmente por una Autoridad de Certificación (CA) confiable, y así autentica la identidad del titular y asegura las comunicaciones cifradas. Los certificados están compuestos por:

- Identificación del titular: El certificado incluye el nombre del titular, que puede ser un sitio web, una organización o una entidad individual.

- Clave pública: El certificado contiene la clave pública asociada al titular del certificado.

- Firma digital: El certificado está firmado digitalmente por una CA. La firma digital garantiza la autenticidad e integridad del certificado, ya que verifica que el certificado no haya sido modificado y que proviene de una entidad confiable.

Debido a que un certificado debe de ser firmado por una CA, se forman las cadenas de confianza o de certificación, una secuencia de certificados digitales que conecta el certificado del servidor al certificado raíz de confianza del navegador o sistema operativo. Cada certificado en la cadena está firmado por el certificado anterior en la cadena, excepto el certificado raíz, que es auto firmado. Los certificados se obtienen mediante el siguiente proceso:

- Generación de clave pública y privada: El titular del certificado genera un par de claves criptográficas, que consta de una clave privada y una clave pública.

- Solicitud de certificado: El titular envía una solicitud de certificado a una CA, proporcionando información sobre su identidad y la clave pública.

- Verificación y emisión del certificado: La CA verifica la identidad del solicitante y realiza los procesos de validación necesarios. Luego, la CA emite y firma el certificado digital.

- Instalación del certificado: El titular del certificado instala el certificado en su servidor web para que esté disponible durante el *handshake* de HTTPS.

Una vez el servidor web obtiene el certificado, cuando un cliente accede al sitio web, se realiza la validación del certificado, esta validación consta de tres pasos. En la primera se verifica la cadena de confianza asegurándose de que esté firmada por una CA de confianza; después se comprueba que el certificado no haya expirado y que esté dentro del periodo de validez. Por último, el cliente verifica que el nombre de dominio en el certificado coincida con el sitio web al que está accediendo[22].

Fig. 7.2: Diagrama de verificación de certificados HTTPS

Una vez explicados los fundamentos y la importancia de los procesos que aporta HTTPS se podrá ver en el siguiente apartado la implementación de cara al ecosistema *Classpip* y el por qué, ya que, como se entreve actualmente el entorno funciona bajo el protocolo HTTP.

7.1.2 Implementacion de HTTP/SSL en Classpip

En este apartado, el problema central radica en la necesidad de HTTPS por parte de gran cantidad de navegadores y dispositivos para hacer uso de servicios de información sensible, en este caso, para acceder a los servicios de geolocalización (GPS). Este obstáculo se encuentra en las partes finales del desarrollo, en el momento de querer hacer un despliegue de cara a pruebas más complejas. En el momento de ejecutar el juego y hacer pruebas nos percatamos

que en caso de hacer uso de un ordenador, si este ejecuta la aplicación en local, los servicios de ubicación funcionan. Al arrancar el juego, el navegador pide confirmación al usuario sobre dar acceso a los servicios de ubicación a través de una notificación y ya está.

El problema viene en el momento que probamos un despliegue externo, a través de *ionic serve --external*. En ese momento se observa que ni con un móvil ni con otro ordenador que no sea el que ejerce de *host*, son accesibles los servicios de ubicación. Esta situación limita muchísimo el correcto funcionamiento del juego y delimita su puesta a prueba en un entorno completo.
En nuestro caso, Miguel nos puso en contacto con Roc Meseguer, el cual nos ofreció una solución para esta problemática, añadir *certbot* al servidor de producción[18] y que este genere los certificados firmados que nos permitan activar HTTPS. Todo y así, nos encontramos frente a dos problemáticas: la primera de ellas era que no disponíamos de mucho tiempo, estuvimos estudiando la posible implementación del proxy inverso necesario para la instalación, lo cual nos suponía un mayor problema teniendo en cuenta la segunda problemática; necesitábamos acceder y modificar el contenido del servidor alojado en la máquina de producción, lo que implicaba un riesgo añadido al suponer que, si cometíamos algún error, es posible que no tuviéramos tiempo de solucionarlo y dejaríamos todo el entorno sin servicio en producción, ésto lo comentamos con Miguel en una de las últimas semanas y nos puso en contacto con Joana Orpella, otra alumna a la que tutelaba su TFG sobre otro proyecto ajeno a *Classpip*, para tratar de intercambiar información, ya que ella si que estaba utilizando HTTPS y quizá nos podría ayudar con ello, mientras que nosotros le podríamos enseñar nuestro enfoque para obtener la localización del navegador.

Joana nos comentó que ella había descartado la idea de obtener los certificados de CA de confianza y en su lugar había generado ella misma certificados auto firmados que por el momento, le permitían implementar HTTPS de forma provisional para utilizar sus aportaciones al proyecto, por lo que decidimos también utilizar certificados auto firmados para que, pese a que no sea una forma segura de implementar HTTPS (los navegadores incluso muestran una pestaña de advertencia que avisa del riesgo de acceder a páginas que funcionan con certificados auto firmados no seguros), nos podrían al fin permitir acceder a la localización utilizando los sensores GPS del dispositivo móvil, que deberían ofrecernos una mejor precisión que en los dispositivos portátiles. Para ello, generamos primero una clave privada para el certificado utilizando el siguiente comando de OpenSSL:

openssl genrsa -out localhost.key 4096

Donde indicamos que queremos una clave generada mediante RSA de 4096 bits y que se guarde en el mismo directorio donde estamos ubicados con el nombre *localhost.key*.

Una vez obtenida la clave privada, podemos generar el certificado mediante:

openssl req -x509 -new -nodes -key localhost.key -sha256 -days 1024 -out localhost.crt

En el cual le indicaremos que se base en *x.509* (el estándar de certificados de clave pública). La opción *-nodes* indicará que la clave privada no se encriptará y la opción *-key localhost.key* apunta al archivo que contiene la clave privada que utilizará. Utilizaremos SHA-256 para el *hash* de la firma digital que incorpora el certificado y finalmente indicamos también que queremos que el certificado se almacene como *localhost.crt*.
Una vez generamos este certificado, se generó otro para poder implementar HTTPS también en la comunicación con la API, puesto que, al establecer una conexión por HTTPS se bloquean las peticiones HTTP. Una vez configurados los certificados y almacenados dentro de las carpetas del proyecto, se actualizaron las direcciones para los hosts y una vez añadidos los nuevos certificados a los certificados de confianza de nuestro navegador, accedemos por primera vez a la aplicación *Classpip* en modo HTTPS utilizando una línea de ejecución especial:

ionic serve --external --ssl --ssl-key src/assets/ssl/localhost.key --ssl-cert src/assets/ssl/localhost.crt

Donde, además de indicarle con la flag *--external*, que queremos que aloje el servidor en una dirección que no sea localhost, le introduciremos manualmente tanto la clave privada como el certificado. Una vez compilado, accedemos a la aplicación:

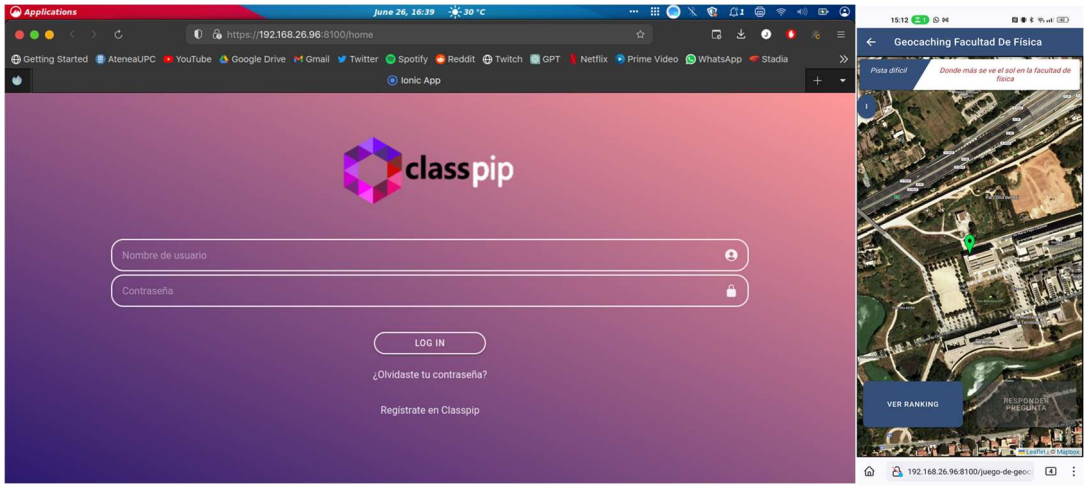

Fig. 7.3: Captura de pantalla de la página de acceso a la app de *Ionic* en modo escritorio y pantalla de juego de *geocaching* desde un dispositivo móvil.

Como se puede ver en la figura inmediatamente anterior, como pese a aparecer un aviso de que el certificado del sitio web está auto firmado por una CA que se ha añadido como excepción manualmente, utiliza SSL. Para acceder a la web desde móvil también ha sido necesario acceder de una forma en concreto y es que para que el mecanismo de seguridad CORS no bloquease la comunicación

entre la aplicación *Ionic* y la API, ha sido necesario acceder desde el navegador *Firefox Nightly* que permite cambiar ajustes que la mayoría de navegadores móviles no permiten como cambiar el valor de la opción *security.fileuri.strict_origin_policy* que permite acceder al recursos de diferentes orígenes.

CAPÍTULO 8: Diseño final

En el cuarto capitulo se mostró una maqueta que enseñaba cuál era nuestro objetivo para la aplicación móvil que al fin y al cabo es la punta de lanza de este desarrollo. Ese *mockup* constituyó el punto de inicio sobre el cuál partir para hacer las diferentes implementaciones y mejoras que ya se han explicado en estos recientes capítulos.

Es por esto que ahora es conveniente comentar y hacer una comparativa entre el diseño final y el *mockup* que como se ve a continuación ha sufrido una evolución considerable.

Se valorará principalmente haciendo comparativas sobre la maqueta de la aplicación del móvil del alumno. En un primer lugar hablaremos de las diferencias y de las funciones o cambios que no se han integrado:

- *Timer*: El temporizador fue una idea inicial, que tal y como se comenta en el punto del cuarto capítulo, servía como un recurso para equiparar el juego con un examen tradicional. Al fin y al cabo, si comparamos ambas herramientas, el objetivo final es el mismo: contestar correctamente una serie de preguntas para conseguir la puntuación más alta. Esta idea de cara al diseño final se desestimó debido a que se encontraron ciertos conflictos sobre como funcionaría. El principal problema que encontrábamos era como implementar el temporizador en un juego cuyo punto central es la búsqueda de puntos y donde además, los jugadores acceden de manera asíncrona al juego. Esto último dio pie a pensar sobre en qué momento el temporizador se activaba y como el profesor podía activarlo o desactivarlo. Se valoró de manera más justa que si la finalidad del juego era la exploración de un medio y el aprendizaje no aportaba algo realmente beneficioso al jugador el uso de este elemento.

- Mapa y componente de información: El mapa y el proceso de responder pregunta aparecían el segundo superpuesto al primero en el momento que se accionaba el botón central. Debido a la importancia que consideramos que tenía el mapa, este fue la primera piedra que se plantó en la aplicación. En un primer momento, cuando ya estaba correctamente implementado vimos que el principal atractivo de la aplicación era el hecho de tener un mapa con el que ver el entorno en el que se juega. El segundo punto fue a raíz de implementar el siguiente cambio, la información del juego. Aquí se sentaron las bases para el diseño resultante. Una vez arrancado el juego, como ya se ha explicado el jugador puede acceder a la información inicial del juego en cualquier momento y se carga en forma de *pop-up* encima del mapa con este y el resto de la interfaz de fondo. Al ver este resultado, nos pareció que era una manera mucho más atractiva a la par que más simple y sencilla que si hubiésemos seguido el diseño inicial.

- Ranking y botones: Siguiendo el punto anterior, los botones fueron redistribuidos y la idea del ranking modificada. La rendición se mantuvo sobre la flecha de la esquina superior izquierda como símbolo de salida del juego, ya que solo en ese caso se utiliza para finalizar el juego (si el juego finaliza de otra manera, el juego se cierra notificándolo al jugador), de esta manera ese botón no era necesario, por lo que la derecha inferior de la pantalla quedaba vacía de contenido. Debido a esto el botón "Responder Pregunta" pasaba a ocupar la parte derecha inferior. Entonces el ranking fue modificado, la izquierda inferior de la pantalla que iba a contener un *display* sobre la posición del jugador en comparación con sus rivales se sustituye por un botón "Ver ranking" que sirve para consultar la posición de todos los jugadores que participan en el juego en una tabla. En medio entre estos dos botones, se situaba un pequeño *display* central donde ahí sí, es visible el número de la posición que ocupa el jugador. De esta manera la parte inferior de la interfaz queda mucho más limpio y con menos contenido.

- Las pistas: Las pistas se podrían considerar que son lo más similar o continuista del diseño inicial. Al fin y al cabo, era un objetivo claro tener las pistas en la parte superior sin nada más, al ser el recurso más importante para el desarrollo del juego.

- Botón de la componente de información: Como ya se ha mencionado, el jugador puede consultar la información en cualquier punto del transcurso de la prueba. Es por ello que se añadió como un pequeño botón en el lateral izquierdo que en el diseño inicial no aparece, ya que de primeras, se quiso ser lo más disruptivo posible con el diseño anterior y queríamos intentar obviar el uso de la información sobre como funciona el juego y los puntos. Esto último se tenía en cuenta por el contexto de juego. En una aula, el profesor antes de realizar cualquier prueba, ejercicio o evaluación realiza una explicación sobre los puntos de interés (tipo de prueba, como funciona, que puntos da cada pregunta, etc.) y por esto, se pensó que era buena idea obviar esta parte. Finalmente, se decide mantener ya que al tratarse de un tipo de prueba donde el profesor no está cerca físicamente o no tiene por qué estarlo, un malentendido o un punto no aclarado puede ser un *handicap* muy decisivo en los resultados que obtengan los alumnos.

En conclusión, los dos puntos fuertes sobre los cambios en el diseño son el uso de componentes como elementos "*pop-up"* y el mapa como fondo activo del juego de manera constante.

Hay que destacar que el primero nace de la naturaleza de *Ionic*. Si queríamos hacer uso de elementos tipo notificaciones "*pop-up*", *Ionic* solo nos permitía introducir texto y realizar alertas pero no podíamos cargar más elementos. Esto nos lleva a crear y hacer uso de componentes para poder mostrar correctamente el ranking o el proceso de responder pregunta.

Por otro lado, como ya se ha dicho anteriormente, el mapa fue una razón puramente estética que nos agradó muchísimo, sobre todo en comparación con el diseño que tenía el juego de *geocaching* cuando lo encontramos.

A continuación en la figura 8.1 se muestra la interfaz de usuario que tiene finalmente el juego de *geocaching* después de nuestras aportaciones.

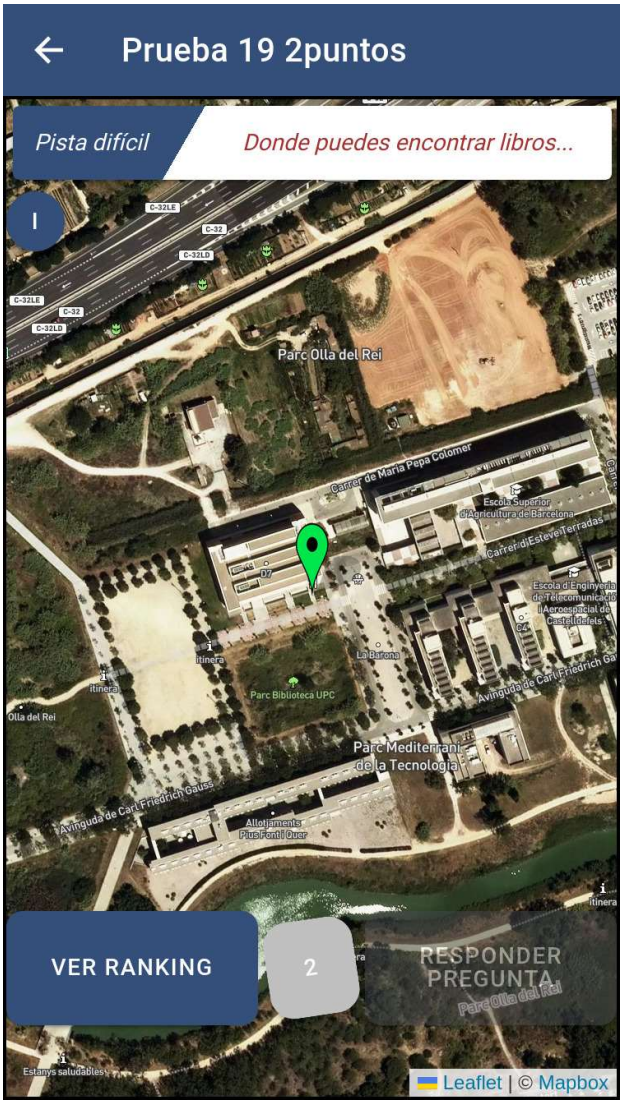

Fig. 8.1: Diseño final de la interfaz del juego de *geocaching* en el móvil del estudiante

CAPÍTULO 9: Pruebas

Durante este capítulo se detallarán los procedimientos que se han seguido para comprobar y probar si las implementaciones han sido exitosas y si han funcionado de la manera esperada. Previo a que los usuarios prueben y evalúen nuestras aportaciones dentro del entorno de *Classpip*, se comentarán de cara a futuras contribuciones para que encuentren en este apartado un apoyo sobre los puntos a la hora de realizar pruebas.

9.1 Punto de arranque

Partiendo de la base de que todo lo que se ha desarrollado es en un entorno frontal, el primer punto era en el momento de comprobar cambios realizados. En ese momento, se tenía que ejecutar este comando a través de la consola o una terminal de cara a ejecutar la API y el servidor en local para el tratamiento y uso de datos:

npm start

Dependiendo de si se ejecuta la aplicación del *dashboard* o la aplicación móvil del estudiante, para ejecutarla se usan comandos distintos. En caso de la aplicación *dashboard* también nos sirve hacer uso del comando anterior, pero al tratarse de una aplicación basada en *Angular* también se puede hacer uso del comando:

ng serve

En caso de la aplicación móvil del estudiante, se puede usar el comando anterior o hacer uso de:

ionic serve

Ambos comandos sirven debido a que como se ha precisado en capítulos anteriores, *Ionic* está construido sobre *Angular*, por lo que hay esa retrocompatibilidad.

9.2 Opciones de configuración avanzadas

Una vez ya tenemos la API, el servidor y el *dashboard* o la aplicación móvil del estudiante arrancadas, puede ser que necesitemos especificar algunas variables para realizar un despliegue más adecuado o que nos interese más.

En el caso de *Ionic*, como ya se ha explicado en capítulos anteriores, permite desplegar la aplicación que se ha creado en móvil o en aplicación web. La aplicación web se puede desplegar tanto en navegador de ordenador como en navegador móvil. De manera automática, si se arranca la aplicación móvil del estudiante, una vez esta lista, se abre automáticamente en una pestaña del

navegador que tengamos predefinido en el ordenador que este ejecutando la aplicación, en el puerto 8100.

En este caso, por ejemplo, el objetivo es que la aplicación móvil se pueda probar y tantear en un dispositivo móvil, es por eso que tenemos que hacer uso de anotaciones *--flags*, en el momento de ejecutar *ionic serve* por consola.

En el caso de que se requiera de usar la aplicación en múltiples dispositivos externos, como puede ser un móvil u otro ordenador, se debe precisar a través de *--external*. Esto hace que el ordenador que ejecuta la aplicación en local, haga de servidor para que se conecten otros dispositivos y puedan cargar la aplicación web a través de una IP y un puerto. En el momento de ejecutar el comando con el flag *--external*, por defecto, designa una IP y un puerto.

Cabe detallar que los dispositivos móviles u otros ordenadores tendrán que encontrarse en la misma red inalámbrica que en la que está conectada el ordenador que ejecuta la aplicación en local.

Si se precisase de hacer uso de una IP concreta o un puerto diferente, se puede usar el flag *--host*. En nuestro caso, también hemos usado el flag *--lab*, esta opción nos da la posibilidad de cargar o emular la interfaz de un teléfono móvil con *Android* y otro al lado con *iOS* y probar las aplicaciones en vista móvil.

Otro punto que se puede tener en cuenta es que si se trabaja con *Ionic*, este por defecto tiene habilitada la opción *livereload*, esta opción lo que hace es que en caso de cambiar algo en el código mientras la aplicación está en ejecución, se recompila y se vuelve a ejecutar. En caso de que no se quiera o no nos interese, se puede deshabilitar a través de *--no-livereload*.

Toda esta información sobre las opciones que tenemos a mano a la hora de ejecutar la aplicación móvil del estudiante son fácilmente accesibles a través de la documentación de *Ionic* oficial[27].

9.3 Ordenadores Linux

En nuestro caso, ambos hemos desarrollado y trabajado con ordenadores con Linux como sistema operativo, trabajando sobre la distribución Linux *Mint*. En esta situación, se nos presentó un obstáculo sobre el uso y ejecución de *ionic serve --external*.

El problema residía sobre el *firewall* del sistema operativo por defecto. En este caso, hay que comprobar siempre que el puerto 8100 este activo o ocupado y que se pueda usar. Se ha detallado el puerto 8100 porque es el puerto que usa *Ionic* por defecto como se ha dicho en el anterior apartado, en caso de usar otro puerto definido por el usuario, esta explicación es igualmente aplicable y se debe realizar la misma comprobación.

En el caso de trabajar con Linux hay que acceder a la gestión de los puertos a través de una terminal. A través de la consola, ejecutamos la siguiente orden:

sudo ufw allow 8100

De esta manera, con permisos de administrador (*sudo*) abrimos el puerto (*allow)* 8100. En caso de querer cerrar el puerto, se utiliza el siguiente comando:

sudo ufw deny 8100

En la siguiente figura, vemos a través de una terminal abierta, los resultados de abrir el puerto 8100.

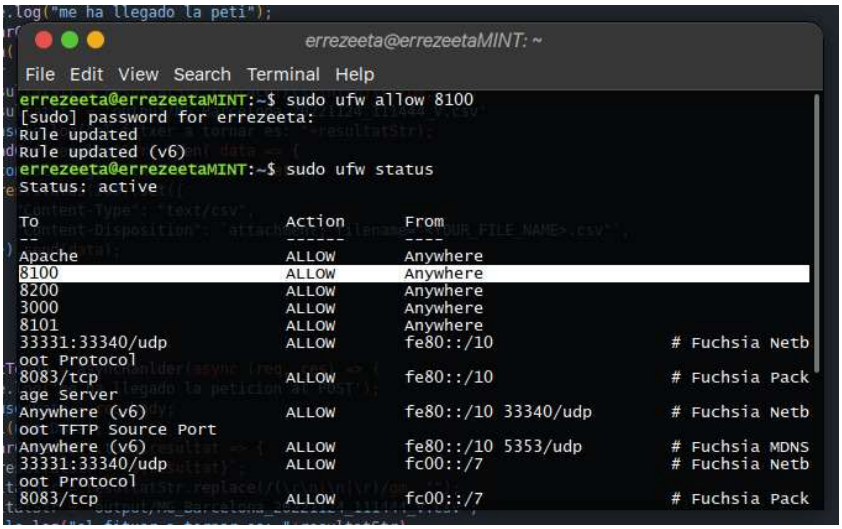

Fig. 9.1: Salida por terminal de la configuración de puertos (*ufw*)

9.4 Obtención de resultados

Una vez se ha tenido en cuenta todos estos factores, no se debería tener ningún problema sobre ver y acceder a la aplicación móvil y ya solo se trata de entrar y acceder a las partes y componentes que han sufrido cambios. Al tratarse de un desarrollo *frontend*, los cambios pueden ser visualizados en la vista directamente. Aun así, se recomienda hacer uso durante el desarrollo de *console.log* y la consola del desarrollador en el navegador de cara a visualizar los procesos y peticiones que se están ejecutando durante la vista.

En el primer caso, los *console.log*, nos pueden servir para imprimir información por consola, como indica el propio nombre y mostrar información de interés que estemos manipulando en ese instante, como si fuesen pequeños puntos de control.

De esta manera, es posible ver todo lo que está pasando y si algunas operaciones en segundo plano, como servicios o peticiones, se dan correctamente. Uno de los puntos clave durante las pruebas ha sido siempre ver si la información se recibe de manera correcta y adecuada. En nuestro caso, el

navegador que se ha utilizado ha sido *Firefox*. Para poder monitorizar la información y las operaciones en segundo plano, accedemos en la pestaña donde se despliegue la aplicación al recurso de la consola del navegador:

Opciones -> Más herramientas -> Herramientas para el desarrollador

o a través del *shortcut:*

Ctrl. + Mayus + I

Si se siguen estos pasos, se pueden verificar que las diferentes implementaciones que hemos aportado tanto mi compañero como yo, funcionan y están implementadas en la aplicación y en el *dashboard*. Durante el desarrollo, cada cambio por mínimo que fuese se comprobaba al momento a través de ejecutar el servidor y la API en local y la aplicación donde se hubiesen realizado los cambios. Esa ha sido la manera de trabajar que hemos considerado más adecuada para avanzar de la manera más eficiente.

9.4.1 Sobre el control de versiones

De cara a trabajar desde los repositorio para clonar el código, en este punto se detalla cómo se ha trabajado con *GitHub,* de cara a posibles dudas que puedan producirse a la hora de realizar pruebas accediendo a nuestro código. En los momentos donde se han producido uniones de las contribuciones (*git merge*) se ha realizado control de versiones y confirmación de estas y se ha repetido el mismo proceso para comprobar que el trabajo conjunto funcionaba correctamente (véase anteriores sendos apartados). Las versiones y cambios que se generaban son fácilmente comprobables en los diferentes repositorios que se han adjuntado en la bibliografía. De esta manera se puede hacer un control sobre las diferentes versiones en las diferentes ramas que se han generado y las diferencias con la versión inicial que se encuentra en el repositorio del ecosistema *Classpip,* también referenciado.

Concluyendo con las pruebas, estas se realizaban una tras otra con el objetivo de aislar el error y atacarlo en concreto. Finalmente, en el momento en el que obteníamos unos resultados que concordaban con lo que se quería realizar con cada implementación, se pasaba al siguiente paso. Una vez se consideraba completa una implementación, se cargaba al repositorio y posteriormente, sobre todo durante el final de las contribuciones, se generaba una rama conjunta donde unir los cambios que se encontraban resguardados en las ramas individuales y repetir el proceso ya definido.

CAPÍTULO 10: Evaluaciones

Durante este capítulo se estudiarán diferentes pruebas y se irá evaluando en base a dos puntos: la experiencia del usuario y la valoración de las aportaciones. El objetivo era conseguir una valoración por parte de usuarios externos al proyecto y así llegar a ver cuáles son los puntos fuertes y débiles del desarrollo realizado.

10.1 Evaluaciones del usuario

Las pruebas se realizaron con la ayuda de nuestro entorno cercano, aprovechando las diferencias de edad para evaluar el nuevo estado del juego. De esta manera, se pretendía conseguir diferentes puntos de vista de diferentes edades ya que, a pesar de que el entorno académico normativo se centra en personas menores de edad y jóvenes adultos, existen también casos de adultos y mayores que deciden retomar estudios. Al ser *Classpip* una herramienta que no tiene un *target* definido en lo que a etapas académicas se refiere, lo vimos como una oportunidad que aportaría riqueza.

Las evaluaciones las agrupamos por grupos de edad; por una parte agruparemos menores de edad (0-18 años), jóvenes adultos (18-35 años) y adultos (+35 años). Teníamos como un objetivo extra intentar conseguir datos sobre pruebas con personas de la tercera edad pero no se pudo contactar con personas cercanas de la tercera edad que tengan unos mínimos conocimientos y manejo básico de tecnologías.

Teniendo en cuenta el problema que tuvimos sobre el uso de servicios de ubicación comentado en el sexto capítulo, la prueba se realizó desde un ordenador. Los usuarios accedían al juego con mínima información por parte de los desarrolladores. El objetivo era que valorasen la interfaz y conseguir conclusiones sobre si la aplicación es intuitiva y accesible. El juego se simuló haciendo un escenario cercano a la ubicación que nos encontrábamos.

- Menores de edad (0-18 años):
  Entienden el juego, con la componente de información se aclaran bastante y con el uso de pistas han adivinado el sitio que se trataba. El diseño generalmente les parece atractivo y agradable, aunque les ha costado un poco de primeras empezar a jugar y entender las mecánicas.

- Jóvenes adultos (18-35 años):
  Les costaba más adaptarse al juego, a pesar de entenderlo con facilidad. En términos de interfaz recibimos quejas sobre la tipografía de la aplicación y un diseño a priori poco vistoso pero dinámico.

- Adultos (+35 años):
  Entienden el juego y las mecánicas pero no les parece intuitivo, se quejan del tamaño de las fuentes principalmente y piden un tamaño mayor de las

letras y del botón de información. Algunos consiguen adivinar las ubicaciones, y otros no. No se aclaran con el uso de las pistas.

10.2 Demostración

Se realiza además una pequeña demostración en forma de video[26] donde se muestran las mejoras del *dashboard* y de la aplicación móvil del estudiante.

En el video, se muestra como un profesor crea un escenario, y crea un juego de *geocaching* haciendo uso del escenario que ha creado. Después de crear el juego, lo activa y pasamos a la aplicación móvil donde el alumno juega al juego que se ha visto creado anteriormente.

En esta demostración se consiguió hacer una emulación de la ubicación y gracias a eso se pudo comprobar ciertas mejoras de una manera mucho más correcta. En el punto que se realiza el video, la implementación de HTTPS no había sido realizada.

Por parte del receptor, la crítica fue positiva y se recibió unas sugerencias sobre adaptar algo más el juego al entorno. En el siguiente apartado, en conjunto con los puntos fuertes y débiles de la aplicación se describirán mejoras realizables mientras que otras estarán representadas directamente en el décimo capítulo donde se comentan posibles futuras mejoras.

10.3 Puntos fuertes y débiles

Como todo desarrollo, con cada mejora que se consigue, se generan nuevas vulnerabilidades o puntos que se deben mejorar.

Después de esto, se consiguió una lista de puntos fuertes y débiles que nacen de los apartados anteriores por parte de los usuarios y la crítica del video, empezaremos con los puntos fuertes:

- El diseño es bastante correcto, en líneas generales se consigue un diseño que atrae mucho más al usuario que el anterior.

- El juego se entiende: Gracias a la componente de información, sin que demos ningún tipo de explicación previa el juego se entiende perfectamente, se considera este un punto importante sobre concluir si el desarrollo ha sido bueno.

- Atrae a los escolares: Por lo que vimos en los menores de edad, les gustó y su implementación sería de interés. El punto de explorar un terreno para adivinar la ubicación es bastante atractivo. Los jugadores a pesar de no poder hacer uso de la geolocalización de manera activa, se empeñaban en intentar adivinar el punto haciendo uso o no de pistas a través de estudiar el mapa en vía satélite.

- El mapa: La implementación del mapa en vía satélite es un éxito. Genera mucha atracción y ayuda a guiarse en detalle a todos los participantes.

- El juego es dinámico, es otro de los factores que se pretendían, que no fuese algo tosco de ver o extraño. Con el diseño actual el juego es bastante más jugable para los usuarios.

Sobre los puntos débiles, la mayoría fueron de carácter concreto. Se centraron sobre todo en competencias que estaban muy relacionadas con el diseño de la interfaz de juego:

- Botón de información: Algunos usuarios se quejaron sobre el hecho de que la componente de información era accesible a través de un botón quizás algo pequeño y que con la vista del mapa y el color de este no era muy vistoso. Este problema de accesibilidad es fácilmente resoluble, aún así cabe destacar que sobre todo se produce debido a que la prueba se realiza en un ordenador con una pantalla de 15,6 pulgadas, esto puede causar que al tener una pantalla grande el botón que en un principio se diseña así para que tenga un carácter discreto en el lateral de la pantalla sea poco visible.

- Tipografía y fuente: La tipografía no acaba de convencer y la fuente se considera algo pequeña también, aunque tengamos en cuenta el factor que se acaba de explicar al final del primer punto débil se puede intentar hacer un poco más grande y vistosa.

- Intuitiva: A pesar de que los usuarios de menor edad consiguen entender con cierta facilidad, los usuarios con mayor edad se quejan de que la aplicación es poco intuitiva y se sienten algo perdidos para situarse a pesar de haber entendido el juego. Demandan algo de más asistencia y algo más de información en pantalla.

- Dispositivo de despliegue: Los usuarios en líneas generales consideran que el despliegue en ordenador de esta aplicación no es muy práctico debido al carácter que tiene el propio juego.

La lectura que sacamos sobre estos problemas se basa sobre todo en lo que respecta el último punto. Las pruebas era realizables también a partir de la vista móvil haciendo uso de *ionic lab* pero aun así seguía situando la aplicación en un ordenador cosa que no interesa de cara a una práctica real. No es práctico realizar este tipo de juegos con un ordenador debido al peso y a estar en constante movimiento con los riesgos que eso tiene (posibles daños a los dispositivos, condiciones ambientales, constante movimiento, entre otros).

Por otro lado, muchos de estos problemas, pasan por desplegar la aplicación en móvil, sobre todo en términos de accesibilidad y visibilidad de la interfaz de la aplicación. Algunas sugerencias como los cambios en tipografía o tamaño de botones son fácilmente mejorables a través de la manipulación de los componentes HTML y CSS.

Finalmente, sobre la demostración que se muestra en video, se consigue una lectura mucho más positiva, ya que en el video, la aplicación se ejecuta en vista móvil. El diseño en móvil gusta y se ve mucho mejor. La sugerencia que se nos plantea es sobre adaptar las distancias según el escenario.

CAPÍTULO 11: Propuestas de futuras mejoras

En este último capítulo, se recoge a través de los desarrollos plasmados en los capítulos previos, una serie de propuestas de mejora para futuras contribuciones y proyectos de otros alumnos y estudiantes que quieran formar parte de este gran proyecto.

Estas propuestas buscan ser lo más completas posibles y ser utilizadas como punto de partida.

11.1 Propuesta classpip-app-móvil-profesor

La aplicación móvil del profesor, como ya se ha visto en el primer bloque de la memoria, sigue un desarrollo y una interfaz muy similar a la aplicación móvil del estudiante. En el caso del módulo del juego de *geocaching*, en un inicio, se definió un diseño en concordancia con el diseño del juego de la aplicación de los estudiantes.

Partiendo del diseño final expuesto en el capítulo 8, la propuesta se basaría en mantener una interfaz similar pero con funcionalidades distintas. Estas serían las siguientes:

- Ranking: El profesor puede verificar la posición de los participantes en cualquier momento. El ranking se podría implementar siguiendo la implementación realizada en el móvil del alumno.

- *Tracking* de ubicaciones: Debido al contexto en el que se desarrolla el juego, se consideró este el punto fuerte de la aplicación del profesor. Mientras los jugadores están jugando, el profesor puede ver la ubicación de todos los alumnos que estén participando al mismo tiempo en su pantalla. Esto añade una capa de seguridad sobre todo para aulas o entornos académicos con menores de edad.

- Botones para empezar o finalizar el juego: El profesor tiene la potestad de activar y desactivar el juego, y creemos que es una buena idea añadir una funcionalidad para empezar una partida y terminarla. Esto, además, podría dar pie a la reutilización de juegos para generar partidas distintas dentro del mismo grupo.

De esta manera se conseguiría que el profesor pueda monitorizar en tiempo real el juego que él ha creado y en caso de ser necesario se pueden estudiar modificaciones en vivo. Una posibilidad más remota sería el hecho de teniendo el mapa y el juego activo, poder añadir puntos a la partida o editar puntos del escenario.

En el punto en el que se encuentra el repositorio de la aplicación móvil del profesor, sería necesario crear una componente de una página para el juego de *geocaching*. Además de esto, sería adecuado también importar y modificar algunos de los servicios y las peticiones a la API que utiliza el juego de

geocaching en la aplicación del estudiante. Gran parte del código puede ser reutilizado y adaptado de parte de la aplicación del estudiante que se ha realizado en nuestro desarrollo. Si se trabaja de esta manera, se puede conseguir tener algo tangible de una manera más directa y a partir de esa base se pueden implementar más mejoras y darle acceso a más información al profesor.

Finalmente, sería conveniente que el profesor pueda acceder a los resultados de cada jugador una vez que ha completado el juego. El profesor podría observar las preguntas y respuestas del jugador y las puntuaciones que ha recibido en cada etapa.

11.2 Implementación de un temporizador en el juego

Aunque bajo nuestro juicio implementar el uso de temporizador en un juego de búsqueda no era algo justo, sí que podría tener una implementación útil de cara a una posible situación del propio juego.

El caso donde es útil es si se produce el empate entre dos jugadores por la victoria. En ese caso, una posible implementación puede ser un temporizador que se active en el momento que el jugador empiece la partida. Este temporizador se puede implementar dentro de la interfaz gráfica o mantenerse como un proceso en segundo plano de cara a no ser un refuerzo negativo en el jugador. En ciertos casos el hecho de tener un temporizador corriendo puede ser un factor de estrés que no es muy conveniente en un entorno de gamificación y le restaría eficacia al juego y al entorno.

Si esta propuesta es implementada, este dato se podría registrar en la tabla del ranking de cara al final de la partida. Trabajando sobre el servicio que usamos para obtener una lista de alumnos que luego se muestra en la tabla del ranking, se puede modificar para que compare en función del registro del tiempo una vez se haya terminado el juego.

11.3 Mejoras en la implementación de HTTPS

En el caso del juego que se presenta en estas aportaciones, se opta por el uso de certificados auto firmados a modo de pequeño parche, como se ha comentado en la introducción del apartado 7.2.2.

Para la implementación de HTTPS en la aplicación para los alumnos de *Classpip*, la mejor solución es recurrir a una herramienta llamada *Certbot*.[23] *Certbot* es una herramienta gratuita de software de código abierto que permite obtener automáticamente certificados de la Autoridad Certificadora *Let's Encrypt* para activar HTTPS en servidores web. *Certbot* está desarrollado por *Electronic Frontier Foundation* (EFF), una fundación sin ánimo de lucro con sede en San Francisco que aboga por la libertad de expresión, privacidad e innovación digital. Tanto *Certbot* como *Let's Encrypt* nos permitirán obtener certificados firmados por una CA de forma totalmente gratuita.

CAPÍTULO 12: Conclusiones

Finalizando este informe, en este apartado se encontrarán las ideas finales que se han extraído de todo el proyecto realizado. Primero se presentarán unas conclusiones en grupo, donde se hablarán de la lista de objetivos y temas relacionados sobre el trabajo realizado entre ambos colaboradores y como nos hemos encontrado trabajando juntos. En un segundo punto, habrá unas conclusiones ya a nivel personal de cada uno, donde se verán puntos difíciles y más dificultosos y la visión individual de las aportaciones realizadas por cada uno de los colaboradores.

El primer apartado es común en ambas memorias ya que es fruto de una puesta en común y una lectura de todo el trabajo por parte de mi compañero y yo. En este apartado, adjuntamos para su revisión en caso de ser necesarios los enlaces a los repositorios donde hemos generado cambios:

- Repositorio sobre las aportaciones a la aplicación móvil: https://github.com/errezeeta/classpip-movil-estudiante-dev
- Repositorio sobre las aportaciones a la aplicación *dashboard*: https://github.com/Angelfm98/classpip-dashboard
- Repositorio sobre las aportaciones a la API: https://github.com/errezeeta/classpip-API

12.1 Conclusiones del grupo de trabajo

Primero realizaremos una visión más técnica de las conclusiones y es que, como esperábamos al empezar a trabajar en el proyecto, realizar una serie de aportaciones a un ecosistema ya previamente trabajado por diversas personas ha supuesto en diversos puntos del trabajo un desafío a la hora de entender cómo funcionaban ciertos algoritmos del sistema o cómo coexisten en el entorno muchos enfoques y estilos distintos a la hora de implementar los diferentes módulos que conforman *Classpip*. Eso nos ha servido también para tratar de ser más conscientes de que, puede qué, nuestras aportaciones hayan de ser modificadas en algún punto por lo que hemos tratado de desarrollarlas de forma que sean fácilmente entendibles, siguiendo un hilo conductor a la hora implementar las diversas funciones añadidas, y tratando de añadir comentarios que ayuden a explicar el funcionamiento de fragmentos del código.

Debido a esto también es destacable la forma en la que nuestro ritmo de implementación de mejoras ha sido exponencial, siendo que, para ambos, las primeras mejoras nos tuvieron ocupados varias semanas en implementarlas aunque, por si mismas, no eran grandes modificaciones en el funcionamiento del entorno pero sí que requirieron que entendiéramos cómo funcionaba, por ejemplo, la gestión de las diferentes páginas en la aplicación *dashboard*, los

elementos estandarizados en la interfaz, el funcionamiento de las peticiones y servicios, etc.

Mientras que durante los últimos meses de desarrollo, ya teníamos muy interiorizado el funcionamiento de todo el entorno y creemos que acabamos este proyecto con un considerable dominio a nivel global de cómo funciona *Classpip* como conjunto. A nivel de los lenguajes y herramientas utilizadas, cabe destacar que ya teníamos previa experiencia en algunas de las tecnologías utilizadas como pueden ser *TypeScript*, HTML, CSS y *frameworks* como *Angular*, así como *Git* y *GitHub*. Aun así, es también verdad que nos hemos visto frente a herramientas con las cuales no habíamos trabajado aún, véase los *frameworks Ionic* y *Loopback*, controles de geolocalización, librerías y algunas API externas, consideramos que ha sido realmente interesante y enriquecedor movernos en este ambiente donde nos enfrentábamos tanto a un sistema como a unas herramientas que en muchas ocasiones escapaban a nuestros conocimientos ya que nos ha puesto en una situación en la que nos forzaba a adaptarnos, estudiarlas, aprender sobre ellas y experimentar, lo que consideramos que es algo especialmente importante en el sector profesional al cual queremos enfocarnos.

El sector del desarrollo de software es un campo con infinidad de opciones y cada día aparecen nuevas y, ante la imposibilidad de conocerlas todas, reside la importancia en el saber adaptarse a nuevas tecnologías sobre la base que ya conocemos.

Al inicio del proyecto, sabíamos que gran parte del resultado que obtuviéramos dependía de nuestra organización y gestión del trabajo como grupo, realizar este trabajo de final de grado en pareja nos ha brindado muchas ventajas como puede ser el reparto de las diferentes tareas y responsabilidades, donde hemos podido tomar una metodología de trabajo en la que cada uno hemos podido trabajar en un aspecto en concreto, profundizando en él y realizando las tareas individuales; para luego también hacer una tarea de integración de nuestros avances poniendo en común los resultados obtenidos, así como los problemas que nos hemos encontrado para: o bien buscar ayuda en el compañero o bien para compartir las soluciones a éste para evitar futuras complicaciones similares.

Por otro lado, el hecho de trabajar en pareja también ha supuesto determinadas situaciones en la que optábamos por enfoques distintos y el hecho de ser solo dos personas nos ha facilitado el hecho de tener que negociar y buscar soluciones que nos resultaran lo más óptimas posibles para los dos.

Entorno a este proyecto, también se definieron unos objetivos que se querían conseguir como grupo, la lista de estos objetivos se puede consultar en el tercer capítulo. Creemos que hemos conseguido satisfactoriamente cumplir estos objetivos. Esta lista fue creada durante la primera mitad del proyecto para focalizar la acción y el progreso en puntos que considerábamos más dificultosos o para concentrar las implementaciones.

Así pues, el único punto que sí somos estrictamente rigurosos no se ha conseguido ha sido la implementación de HTTPS. Esto ha sido debido al

contexto que requería la implementación, se ha conseguido sortear la obligatoriedad de utilizar HTTPS aunque sigue siendo algo necesario para posteriores contribuciones al desarrollo que este correctamente implementado siguiendo la solución que propusimos y que fue propuesta a investigación por parte de Roc.

El resto de objetivos consideramos que nos han ayudado a profundizar y aprender muchísimo como ya hemos comentado como grupo en estas conclusiones y solo queda agradecer a este proyecto el hecho de habernos facilitado y entrenado sobre un desarrollo práctico, desafiante y sobre todo muy interesante.

12.2 Conclusiones personales

Por mi parte, y a modo individual, volviendo a la introducción del trabajo, se esperaba que en mi caso en este proyecto me encargara de las implementaciones de mapas con geolocalización en tiempo real, estilización de interfaces de usuario y diseño de la interfaz del juego y el despliegue en movil. Ahora, una vez realizadas todas las implementaciones y aportaciones, termino mi participación en el proyecto *Classpip* confiando en haber hecho un trabajo que se aproveche en el futuro del ecosistema y que pueda seguir formando parte de este gran proyecto.

En mi opinión, trabajar en *Classpip* ha sido una experiencia más enriquecedora de lo que esperaba, la primera vez que hablé con Miguel, esperaba que durante el TFG el reto residiera más bien en dominar un nuevo lenguaje o *framework*, y pese a que, como siempre sucede, ha habido momentos de leer documentaciones, buscar tutoriales y consultar foros, la verdad es que por mi parte he destinado una parte muy importante del tiempo de trabajo en entender cómo funcionaba el código de otros antes de poder extenderlo yo. Es una situación a la que nunca me había enfrentado ni durante las asignaturas de desarrollo de software de la titulación ni en los puestos donde he trabajado, en los que o bien he gestionado o trabajado en proyectos comenzándolos desde cero, aquí en cambio me he tenido que adaptar y siento que es esto lo más común en escenarios y equipos laborales reales.

Además, siendo sincero, en un principio también esperaba trabajar más en el *backend* del proyecto y siento que el focalizar prácticamente todas las implementaciones en el *frontend* me ha hecho descubrir una parte del desarrollo de software que realmente no conocía y que en parte, menospreciaba de cara a la complejidad y profundidad en la lógica, y teniendo en cuenta que me gustaría tratar de abarcar *Full Stack* en el mundo laboral, siento en parte, que este Trabajo de Final de Grado me sirve también para “cerrar” en parte un aprendizaje del desarrollo en el que hasta ahora, en los proyectos y trabajos que he realizado, me he centrado en las partes de *backend* (sobre todo el desarrollo de las API y parte de Bases de Datos) y *devOps* (“dockerización” y automatizaciones).

Finalmente, hacer una última mención de agradecimiento tanto a Miguel como a Roc, gracias por todo, por ayudarnos siempre que ha sido necesario, las reuniones en horarios fuera de lo común y sobre todo, por dejarnos formar parte de lo que es *Classpip*, una aplicación “de profesores y alumnos, para profesores y alumnos”, a la cual le deseo lo mejor y espero seguir sabiendo de ella.

Bibliografía

A continuación, se adjuntan algunos enlaces de información relativa a ciertos puntos del trabajo o herramientas web que han sido utilizadas.

[1] Ludificación. https://es.wikipedia.org/wiki/Ludificaci%C3%B3n
[2] Definición de ludificación. https://dle.rae.es/ludificar?m=form
[3] La tasa de abandono educativo se mantiene estable en el 13,9% en 2022. 27/01/2023.https://www.educacionyfp.gob.es/prensa/actualidad/2023/01/20230127-aet.html
[4] Problemas de la gamificación. https://www.videojuegosvascos.com/videojuego-industrial/carlos-gonzalez-tardon-dr-psicologia-y-experto-en-gamificacion-aplicar-procesos-de-gamificacion-en-un-ambiente-laboral-problematico-puede-ser-contraproducente/
[5] The Fun Theory. Volkswagen. https://www.youtube.com/watch?v=SByymar3bds
[6] Geocaching. https://es.wikipedia.org/wiki/Geocaching
[7] Disponibilidad Selectiva. https://es.wikipedia.org/wiki/Disponibilidad_selectiva
[8] ChatGPT. https://chat.openai.com/
[9] Web oficial de Classpip. http://classpip.upc.edu:8120/#/home
[10] Repositorio oficial de Classpip. https://github.com/classpip
[11] Strongloop. https://en.wikipedia.org/wiki/Strongloop
[12] Plantilla del diagrama de Gantt del proyecto. https://templates.office.com/es-es/diagrama-de-gantt-simple-tm16400962
[13] Figma. https://www.figma.com
[14] Repositorio del equipo para *classpip-app-movil.* https://github.com/errezeeta/classpip-movil-estudiante-dev
[15] Repositorio del equipo para *classpip-dashboard.* https://github.com/Angelfm98/classpip-dashboard
[16] Kayce Basques. “Por qué el HTTPS es importante”. 07/04/2020 https://web.dev/why-https-matters/
[17] SSL.com Support Team. “What is HTTPS?”. 12/10/2021 https://www.ssl.com/faqs/what-is-https/
[18] leangaurav. “Simplest HTTPS setup: Nginx Reverse Proxy + Letsencrypt + AWS Cloud + Docker”. 29/07/2021 https://leangaurav.medium.com/simplest-https-setup-nginx-reverse-proxy-letsencrypt-ssl-certificate-aws-cloud-docker-4b74569b3c61
[19]“¿Qué ocurre durante un protocolo de enlace TLS?” https://www.cloudflare.com/es-es/learning/ssl/what-happens-in-a-tls-handshake/
[20] Sathya Bandara. “Ensuring Message Integrity with HTTP Signatures”. 25/06/2019. https://technospace.medium.com/ensuring-message-integrity-with-http-signatures-86f121ac9823
[21] Transport Layer Security. https://en.wikipedia.org/wiki/Transport_Layer_Security
[22] Dante Odín Ramírez López, Carmina Cecilia Espinosa Madrigal. “El cifrado web (SSL/TLS).” https://revista.seguridad.unam.mx/numero-10/el-cifrado-web-ssltls

[23] Electronic Frontier Foundation. “about certbot”
https://certbot.eff.org/pages/about
[24] Web para gestión de versiones de dependencias con NPM.
https://www.npmjs.com/
[25] Web para saber la compatibilidad de dependencias a través de NPM.
https://www.npmpeer.dev/
[26] Demostración juego Geocaching Classpip. https://youtu.be/ZWO3gHtvFQU
[27] Documentación sobre opciones de ionic serve
https://ionicframework.com/docs/v5/cli/commands/serve
[28] Repositorio del equipo para classpip-API:
https://github.com/errezeeta/classpip-API

Anexo 1: Función EliminarVoto()

En este anexo, se muestra la lógica detrás de la función citada en el apartado 2.2.1 como un extra de información en caso de querer ser estudiada, aún así se puede acceder en el código de la aplicación *dashboard* en el componente *juego-de-control-de-trabajo-en-equipo-seleccionado-activo.component*

```
1  public async Borra(id: number)
2  {
3      const alumnosEquipo = await this.peticionesAPI.DameAlumnosEquipo (
4      this.equipoAhora.id).toPromise();
5      var alumnoSelecionado: Alumno;
6      for (let i=0, encontrado=false; i<alumnosEquipo.length &&
7      encontrado==false ; i++)
8      {
9              if (alumnosEquipo[i].id==id)
10             {
11                     encontrado=true;
12                     alumnoSelecionado= alumnosEquipo[i];
13             }
14     }
15     console.log("He hecho la busqueda, y voy a ir a buscar el id=
16     "+alumnoSelecionado.id+". Que pertenece a: "+alumnoSelecionado.Nombre);
17     const inscripcion = await
18     this.peticionesAPI.DameInscripcionAlumnoJuegoDeControlDeTrabajoEnEquipo(th
19     is.juegoSeleccionado.id, alumnoSelecionado.id).toPromise();
20     console.log("Me voy a cargar la inscripcion del alumno con id:
21     "+inscripcion[0].alumnoId+" .Que ha hecho la votación con id:
22     "+inscripcion[0].id+" Que porcierto pertenece al juego:
23     "+inscripcion[0].juegoDeControlDeTrabajoEnEquipoId);
24     await
25     this.peticionesAPI.BorrarInscripcionAlumnoJuegoDeControlDeTrabajoEnEquipo(
26     inscripcion[0].id).toPromise();
27     console.log("Ya he borrado la inscripcion del alumno con id:
28     "+inscripcion[0].id+"Ahora voy a crear la inscripción vacía");
29     this.peticionesAPI.InscribeAlumnoJuegoDeControlDeTrabajoEnEquipo( new
30     AlumnoJuegoDeControlDeTrabajoEnEquipo(inscripcion[0].alumnoId,
31     inscripcion[0].juegoDeControlDeTrabajoEnEquipoId))
32     .subscribe();
33 }
```

Fig. A1.1: EliminarVoto()

Anexo 2: Funciones sobre la geolocalización y el mapa

En este segundo anexo, se detallan algunas de las funciones importantes sobre inicializar el mapa y cargarlo en la aplicación, seleccionar una ubicación o punto en el mapa y salir del mapa para la aplicación *dashboard* dentro del componente *map-popup.component.*

```
1  private initMap(): void
2  {
3       this.map = L.map('map',
4       {
5                center: [ 41.2758, 1.9882 ],
6                zoom: 15
7       });
8       const tiles = L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png',
9       {
10               maxZoom: 18,
11               minZoom: 3,
12               attribution: '&copy; <a
        href="http://www.openstreetmap.org/copyright">OpenStreetMap</a>'
13      });
14      tiles.addTo(this.map);
15 }
```

Fig. A2.1: initMap()

```
1  private selectLocation()
2  {
3       this.map.on('click', (e: { latlng: any; }) =>
4       {
5                if (this.first == true)
6                {
7                         var coord = e.latlng;
8                         console.log("Click en "+coord.lat+" y "+coord.lng);
9                         this.selection = new L.Marker([coord.lat, coord.lng], {icon:
10                        this.icon}).addTo(this.map);
11                        if (coord.lat <0)
12                        {
13                                this.data[0] = -Math.abs(coord.lat).toFixed(4);
14                        }
15                        else
16                        {
17                                this.data[0] = coord.lat.toFixed(4);
18                        }
19                        if (coord.lng <0)
20                        {
21                                this.data[1] = - Math.abs(coord.lng).toFixed(4);
22                        }
23                        else
24                        {
25                                this.data[1] = coord.lat.toFixed(4);
26                        }
27                        this.first = false;
28               }
29               else
30               {
31                        var coord = e.latlng;
32                        this.selection.setLatLng(e.latlng);
33                        this.data[0]= coord.lat.toFixed(4);
34                        this.data[1]= coord.lng.toFixed(4);
35               }
36
37      });
38 }
```

Fig. A2.2: selectLocation()

```
onNoClick()
{
      this.dialogRef.close({event: 'Update', data: this.data});
}
```

Fig. A2.3: onNoClick()