JUEGOS ONLINE PARA MEJORAR LA ADQUISICIÓN DE COMPETENCIAS EN PATOLOGÍA MÉDICA BUCAL EN ALUMNOS DE GRADO DE ODONTOLOGÍA

TRABAJO DE FIN DE GRADO

ALEJANDRO PLÁZER ARNAZ
DANIEL GARCÍA-VILLARRUBIA LÓPEZ-MENCHERO

Director
Antonio Sarasa Cabezuelo
Rosa María López-Pintor Muñoz

Grado en Desarrollo de Videojuegos
Facultad de Informática
Universidad Complutense de Madrid

Mayo 2023

ONLINE GAMES TO IMPROVE THE ACQUISITION OF SKILLS IN ORAL MEDICAL PATHOLOGY IN THE DENTISTRY DEGREE PROGRAM

FINAL PROJECT DEGREE

ALEJANDRO PLÁZER ARNAZ
DANIEL GARCÍA-VILLARRUBIA LÓPEZ-MENCHERO

Directed by
Antonio Sarasa Cabezuelo
Rosa María López-Pintor Muñoz

Game Development Degree
Faculty of Computer Sciences
Complutense University of Madrid

May 2023

A nuestros profesores, por guiarnos y enseñarnos todo lo necesario para alcanzar este logro tan importante

Agradecimientos

Queremos expresar nuestro más sincero agradecimiento a todas las personas que hicieron posible la realización de este trabajo de fin de grado. En primer lugar, agradecemos a Antonio Sarasa Cabezuelo y Rosa María López-Pintor Muñoz, nuestros tutores, por su paciencia, dedicación y orientación durante todo el proceso de desarrollo.

También agradecemos a nuestras familias, por el apoyo incondicional, por estar siempre ahí en cada paso que hemos dado y creer en nosotros en los momentos más difíciles.

A todos, ¡gracias de todo corazón!

Resumen

Este trabajo tiene como objetivo proporcionar a los alumnos de grado de Odontología de la asignatura Patología Médica Bucal una aplicación online donde puedan realizar una serie de juegos con el objetivo de poner a prueba, de forma amena y entretenida, sus conocimientos en dicha materia. Esta aplicación también permite a los profesores hacer un seguimiento de los alumnos, pudiendo consultar las puntuaciones obtenidas en los distintos juegos. Ello ayudará al profesorado de esta asignatura en la evaluación de los conocimientos adquiridos por el alumno.

El sistema desarrollado está formado por una aplicación para dispositivos Android y sistemas operativos Windows que utiliza la información añadida, actualizada y obtenida de una base de datos.

Palabras clave

Aplicación dispositivos Android, aplicación, medicina bucal, odontología, minijuegos

Abstract

This project aims to give Dentistry degree students of the Oral Medicine subject an online application where several minigames can be played to test their knowledge in the subject. It also allows professors to keep track on the progress of their students and to check how well they did in every minigame.

The developed system consists of an application available for Android devices and Windows operating systems, using information that has been added, updated, and obtained from a database.

Keywords

Android application, application, oral medicine, dentistry, minigames

Índice general

Índice de Imágenes

Capítulo 1: Introducción

A continuación, se presentan los motivos que llevaron a la realización de este trabajo, así como las necesidades que se pretenden cubrir en alumnos y profesores. Posteriormente, se descubrirán los objetivos específicos que abarca el proyecto, la metodología de trabajo que se ha seguido junto con las fases en las que se ha dividido el proyecto y la forma en la que se ha organizado.

1.1. Motivación

El desarrollo de videojuegos para mejorar la adquisición de competencias en educación es una herramienta sumamente útil para mejorar la enseñanza de cualquier materia en comparación con los métodos de enseñanza tradicionales.

La gamificación o ludificación [27] en el aprendizaje proporciona una forma de aprendizaje activo y dinámico que puede mejorar en gran medida el rendimiento de los estudiantes [1] [2]. La aplicación permite a los alumnos aprender de manera autónoma, a su propio ritmo y de una manera más atractiva y divertida, lo que aumenta la motivación y el interés por una asignatura [3]. Varios estudios indican que, en comparación con los métodos tradicionales, los juegos causan menos estrés y ansiedad en los estudiantes de medicina cuando son evaluados [4].

Además, los juegos se pueden diseñar y modificar para adaptarse a las necesidades y estilos de aprendizaje individuales de los estudiantes, lo que puede mejorar aún más el aprendizaje y la retención de conocimientos.

La Medicina Bucal es el área de la odontología que estudia las enfermedades orales y de las estructuras periorales que no aparecen en los dientes. En esta asignatura se estudian diferentes patologías producidas por agentes diversos como las lesiones traumáticas, infecciosas e inmunológicas, entre otras. También en esta materia se tratan lesiones importantes como las potencialmente malignas y el cáncer oral. Por tanto, el número de patologías tratadas es amplio y es complejo conseguir que el alumno adquiera la capacidad de diagnosticar todas ellas.

La asignatura Medicina Bucal en la UCM tiene 6 créditos ECTS. Los créditos ECTS se basan en el trabajo personal del estudiante: horas lectivas, prácticas, horas de estudio y elaboración de trabajos. Un crédito ECTS equivale a 25 horas de trabajo del estudiante. Esto significa que los 6

créditos ECTS de la asignatura suponen un total de 150 horas de trabajo total del alumno. Y de ellos en esta asignatura se dedican un 25% a la actividad didáctica, 50% a la actividad práctica y un 25% al trabajo del alumno que no necesita la presencia de profesor. Cabe señalar que los créditos asignados como prácticos en esta asignatura incluyen, además de las prácticas con pacientes, la realización de talleres de resolución de problemas clínicos/sesiones clínicas, que en otras áreas podrían considerarse dentro de la didáctica, así como los seminarios clínicos monográficos. Ello es debido a que la discusión de los casos clínicos, bien en seminarios o en sesiones clínicas constituyen la piedra angular sobre la que pivota el aprendizaje práctico en la medicina clásica (ya que es imposible obtener pacientes que cursen con cada una de las patologías a estudiar) y que podrían ser consideradas como prácticas no clínicas.

Considerando lo anterior, en esta asignatura la aplicación de juegos que contengan pruebas asociadas a imágenes clínicas de casos reales podría servir como actividad práctica. También, la realización de juegos no necesita de la ayuda del profesor por lo que podría ser también una actividad para desarrollar en las horas de trabajo del alumno que no necesita la presencia del profesorado. Por ello, y dado que los juegos son una actividad atractiva para el alumnado [5], se decidió crear unas actividades de gamificación que permitieran el aprendizaje diagnóstico en Medicina Bucal y que además pudieran ser útiles para la evaluación docente por parte del profesorado.

1.2. Objetivos

El objetivo principal del trabajo es el desarrollo de una aplicación basada en minijuegos para la asignatura Patología Médica Bucal de la Facultad de Odontología de la UCM que ayude a mejorar el rendimiento académico en esta materia mediante un método de aprendizaje más atractivo, personalizado y efectivo.

Este objetivo se particulariza en los siguientes objetivos más específicos:

- Implementar una funcionalidad que permita crear, modificar y eliminar información correspondiente a los usuarios de la aplicación, como son sus datos personales y correo de la universidad y estadísticas en los minijuegos.

- Implementar una funcionalidad que restrinja el acceso a la aplicación solamente a los usuarios registrados en la base de datos. Además, que se restrinja el acceso a ciertas funcionalidades explicadas antes solamente a aquellos usuarios que estén registrados como profesores en la base de datos.
- Implementar una interfaz de usuario que facilite la navegación a las distintas partes de la aplicación y que sea fácil de comprender a simple vista, sin necesidad de explicar las funciones de cada parte.
- Implementar una funcionalidad que permita al usuario modificar los valores del sonido de la aplicación.

1.3. Estructura de la memoria

La memoria del Trabajo de Fin de Grado está estructurada en 8 capítulos y un anexo, que recogen en detalle el proceso de desarrollo del proyecto. A continuación, se detallan los contenidos de cada uno de los capítulos:

- En el capítulo 1 se describen los objetivos que se plantearon para el TFG, así como la metodología que se utilizó para llevarlo a cabo.
- En el capítulo 2 se realiza una revisión de algunas aplicaciones móviles que tienen conceptos o funcionalidades similares a las de este proyecto.
- En el capítulo 3 se presenta una breve descripción de las herramientas que se utilizaron para la elaboración del proyecto.
- En el capítulo 4 se especifican los requisitos de los usuarios que se definieron para esta aplicación.
- En el capítulo 5 se describe en detalle la arquitectura de la aplicación y los modelos de datos del servidor que se implementaron.
- En el capítulo 6 se detalla el proceso de elaboración de las distintas partes de la aplicación, incluyendo la interfaz y los módulos definidos en el capítulo 4.
- En el capítulo 7 se presentan las conclusiones y se proponen posibles extensiones del trabajo en proyectos futuros.
- En el capítulo 8, los dos miembros del equipo describen sus aportaciones personales o grupales en el proceso de elaboración del proyecto.
- El anexo A consiste en una guía de uso para los usuarios de la aplicación, con el fin de

que puedan utilizarla.

1.4. Metodología

El desarrollo del proyecto se ha realizado de acuerdo con un plan de proyecto y a una organización del trabajo que se explica en los siguientes apartados.

1.4.1. Fases del proyecto

Para desarrollar el proyecto, se hizo una planificación que incluyó la toma de requisitos, desarrollo de las aplicaciones y el desarrollo de la memoria. La imagen 1.2 muestra una estimación de la duración de cada periodo, así como la representación visual mediante un diagrama de Gantt en la imagen 1.1.

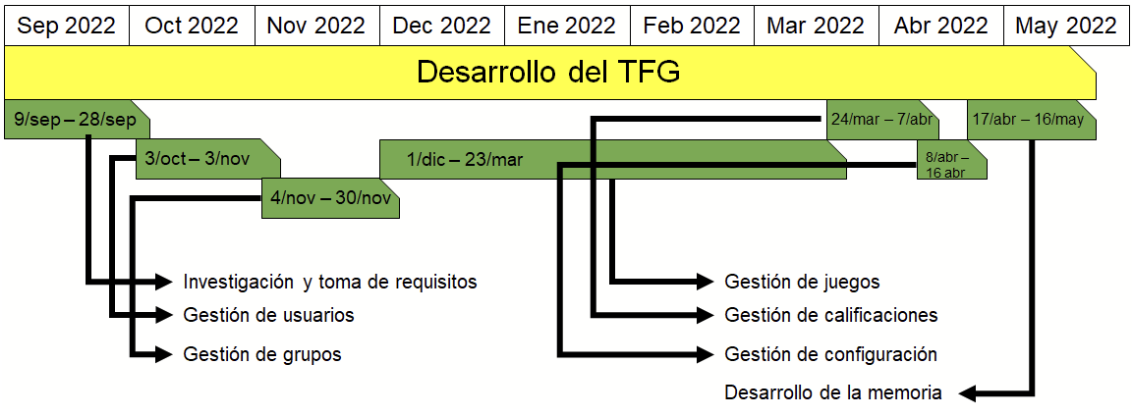


Imagen 1.1: Diagrama de las etapas del desarrollo del proyecto

Etapa	Duración
Investigación y toma de requisitos	9 Septiembre – 28 Septiembre
Desarrollo del módulo de Gestión de usuarios	3 Octubre - 3 Noviembre
Desarrollo del módulo de Gestión de grupos	4 Noviembre - 30 Noviembre
Desarrollo del módulo de Gestión de juego	1 Diciembre – 23 Marzo
Desarrollo del módulo de Gestión de calificaciones	24 Marzo – 7 Abril
Desarrollo del módulo de Gestión de configuración	8 Abril – 16 Abril
Desarrollo de la memoria	17 Abril – 16 Mayo

Imagen 1.2: Etapas en el desarrollo del proyecto

A continuación, se procederá a explicar cada una de las fases en las que se ha segmentado el desarrollo del proyecto

Fase 1: Investigación y toma de requisitos

En esta primera fase se especificaron los requisitos que necesitaba cubrir la aplicación. Se dividió a los usuarios en dos grupos grandes, mediante unos diagramas de casos de uso: alumnos y profesores. También se tomaron en cuenta las necesidades específicas de cada grupo, así como los privilegios de los profesores frente a los alumnos.

Fase 2: Desarrollo del módulo de gestión de usuarios

En la segunda fase se comenzó el desarrollo de la aplicación y la creación del servidor utilizando la herramienta Strapi. En esta fase se crearon las funciones e interfaces de usuario encargadas de gestionar la base de datos y los casos de uso de los usuarios para sus datos personales, como son: registro de usuario, inicio de sesión, cierre de sesión, eliminación de cuenta y modificación de datos de usuario.

Fase 3: Desarrollo del módulo de gestión de grupos

En la tercera fase se comenzó el desarrollo de la parte de la aplicación dedicada a la gestión de grupos. En esta fase se crearon las funciones e interfaces de usuario para poder crear o eliminar un grupo de usuarios, así como poder unirse o meter usuarios a grupos, y sus análogos para expulsar usuarios. En esta fase es donde verdaderamente empieza a existir una diferencia entre usuarios de tipo ‘alumno’ y de tipo ‘profesor’, pues solamente estos últimos tienen permiso de administrar los grupos, con todos sus casos de uso.

Fase 4: Desarrollo del módulo de gestión de juego

La cuarta fase consistió en la creación de los juegos de la aplicación, así como las interfaces de usuario necesarias para cubrir las necesidades de los usuarios para éstos. Se crearon las estructuras necesarias para que la aplicación pudiera dar al jugador la capacidad de elegir el juego a jugar, menús de pausa y configuración en el juego, así como la retroalimentación tras terminar el juego, indicando fallos y aciertos a las preguntas.

Fase 5: Desarrollo del módulo de gestión de competiciones

La fase número cinco corresponde a las posibilidades que da la aplicación de enfrentar a dos grupos de alumnos en una competición mediante un juego, y la forma de puntuar en esta.

Fase 6: Desarrollo del módulo de gestión de configuración

La fase número seis corresponde a los ajustes de control de volumen de los que dispone la aplicación.

Fase 7: Desarrollo de la memoria

La fase número siete corresponde a la escritura de la memoria en la que se describe en detalle aspectos del desarrollo tales cómo las motivaciones, objetivos, metodologías, organización, tecnologías, requisitos y funcionalidades del proyecto en su totalidad.

1.4.2. Organización del trabajo

Durante la realización del proyecto se ha mantenido una metodología de trabajo iterativa incremental. Estas iteraciones han tenido una duración desde una semana hasta 3 meses según la dificultad y extensión de cada hito, terminadas con una reunión en la que se validaba el trabajo realizado en este periodo. También en esta misma reunión se fijaban las tareas y funcionalidades a llevar a cabo durante la siguiente iteración junto con la fecha de finalización de esta.

Debido a que el trabajo está formado por dos personas, se hizo uso de herramientas para poder trabajar en paralelo ambos miembros del grupo utilizando Github [8].

Introduction

The following section presents the reasons that led us to the completion of this work, as well as the needs that are intended to be addressed in students and professors. Subsequently, the specific objectives encompassed by the project will be uncovered, along with the working methodology that has been followed and the phases into which the project has been divided, as well as the organizational structure.

Motivation

The development of video games to enhance skill acquisition in education is an extremely useful tool for improving the teaching of any subject compared to traditional teaching methods.

Gamification in learning provides a way of active and dynamic learning that can greatly enhance students' performance [1][2]. The application allows students to learn autonomously, at their own pace, and in a more engaging, and enjoyable manner, which increases motivation and interest in a subject [3]. Several studies indicate that games compared to traditional methods cause less stress and anxiety in medical students during assessment periods [4].

Furthermore, games can be designed and modified to adapt to individual students' needs and learning styles, further enhancing learning and knowledge retention. Oral Medicine is the area of dentistry that studies oral diseases and perioral structures that are not related to teeth. This subject covers various pathologies caused by different agents, such as traumatic, infectious, and immunological lesions, among others. It also addresses important conditions such as potentially malignant lesions and oral cancer. Therefore, there is a wide range of pathologies to be studied, and it is challenging to ensure that students acquire the ability to diagnose all of them.

The Oral Medical Pathology subject at UCM carries 6 ECTS credits. ECTS credits are based on students' personal work, including lecture hours, practical activities, study hours, and project development. One ECTS credit equals 25 hours of student work. This means that the 6 ECTS credits for the subject correspond to a total of 150 hours of student work. Within these hours, 25% is dedicated to didactic activities, 50% to practical activities, and 25% to student work that does not require the presence of a professor. It should be noted that the credits allocated as practical activities

in this subject include, in addition to patient care, the completion of problem-solving workshops/clinical sessions, which in other areas could be considered part of the didactic activities, as well as specialized clinical seminars. This is because the discussion of clinical cases, whether in seminars or clinical sessions, constitutes the cornerstone of practical learning in classical medicine (since it is impossible to have patients with each pathology to be studied) and could be considered non-clinical practices.

Considering the above, the use of games containing tests associated with clinical images of real cases could serve as a practical activity. Moreover, game-based activities do not require professor assistance, making them suitable for self-directed student work. Therefore, considering the appealing nature of games for students [5], it was decided to create gamification activities that would facilitate diagnostic learning in Oral Medicine and could also be useful for faculty evaluation purposes.

Objectives

The main objective of this work is the development of an application which consists of minigames based on the subject from the degree students in Oral Medicine that would help to improve student performance in the subject using an attractive, customized, and effective way.

This objective could be divided into the following specific objectives:

- Creating a function capable of creating, editing, and deleting the user data, such as personal information, email addresses and minigame performance.
- Creating a function capable of limiting app access to registered users. Certain app sections must only be available to teacher users.
- Creating a user interface which allows easy navigation to every part of the app and is easy to understand.
- Creating a function capable of editing audio levels in the app.

Final Degree Project Report Structure

The final degree project report is structured into 8 chapters and an appendix, which provide a detailed account of the project development process. The contents of each chapter are as follows:

- Chapter 1 describes the objectives set for the final degree project and the methodology used to carry it out.
- Chapter 2 conducts a review of mobile applications that have similar concepts or functionalities to those of this project.
- Chapter 3 provides a brief description of the tools used for the project development.
- Chapter 4 specifies the user requirements defined for this application.
- Chapter 5 provides a detailed description of the application architecture and the server data models that were implemented.
- Chapter 6 details the process of developing the different parts of the application, including the interface and modules defined in Chapter 4.
- Chapter 7 presents the conclusions drawn and proposes possible future extensions of the work.
- Chapter 8 allows the two team members to describe their individual or group contributions to the project development process.
- Appendix A consists of a user guide for the application, aimed at facilitating its utilization.

Methodology

The development of the project has been carried out according to a project plan and a work organization that is explained in the following sections.

Project phases

To develop the project, a planning process included requirements gathering, application development, testing and memory development. The image 1.2 shows an estimate of the duration of each period, as well as the visual representation by using a Gantt chart in image 1.1.

The following is an explanation of each of the phases into which the development of the project

has been segmented.

Phase 1: Research and requirements gathering

This first phase is when app requirements were specified. Users were divided into students and teachers, giving extra privilege to teachers.

Phase 2: User module development

The second phase was the development of the application and the creation of the server using the Strapi tool. During this phase, functions and user interfaces were created to manage the database and the user data, such as the following functions: account creation, log in, log out, account deletion, and user data modification.

Phase 3: Group module development

During this phase, group management started being developed. Functions and user interfaces were created to add or delete groups, join a group, add a user to a group, and kick users from groups. Real differences between users and teachers started in this phase, because only teachers were allowed to manage groups.

Phase 4: Game module development

Minigame development started in this phase. Functions and user interfaces were created to allow minigame selection. Pause and settings menus were added. The option to check minigame performance was added.

Phase 5: Competition module development

During this phase competitions were added, allowing groups to have a duel using a minigame, and later see their results.

Phase 6: Settings module development

Settings to edit audio levels were added during this phase.

Phase 7: Writing the project document

The project document was written to describe certain project aspects such as motivations, objectives, work methods, work organization, technologies, requirements, and functions of the whole project.

Work organization

During the execution of the project, an incremental iterative work methodology has been maintained. These iterations have lasted from one week to three months according to the individual difficulty of each one, ending with a meeting to validate the work done during this period. Also, in this same meeting the tasks and functionalities to be carried out during the next iteration were set, together with the end date of this one.

Since the work is made up of two people, we used GitHub as a version control system so that both members of the team could work in parallel.

Capítulo 2: Estado del arte

En este capítulo se describen algunas aplicaciones con funcionalidades similares al sistema desarrollado en este proyecto.

2.1. Dental School

Aplicación [23] gratuita que ofrece materiales de estudio para estudiantes de odontología, incluyendo notas de conferencias, presentaciones y guías de estudio. La aplicación también incluye un foro de discusión donde los estudiantes pueden interactuar y hacer preguntas.

Además, la aplicación ofrece un directorio de escuelas de odontología en todo el mundo, lo que puede ser útil para aquellos que buscan una institución para estudiar odontología.

2.2. BoneBox - Dental Lite

Aplicación [24] gratuita que ofrece información sobre la anatomía dental, incluyendo modelos 3D de los huesos de la cara y de la boca. La aplicación permite a los usuarios rotar, hacer zoom y acercar los modelos para verlos en detalle, lo que puede ser útil para aquellos que estudian la anatomía dental.

Además, la aplicación incluye información detallada sobre los diferentes huesos, incluyendo los nombres, ubicación y funciones. En general, la aplicación puede ser una herramienta útil para estudiantes de odontología y profesionales dentales que buscan una forma interactiva de aprender sobre la anatomía dental.

2.3. Dental AR

Aplicación [25] que fue desarrollada en Unity3D con Vuforia. Es una herramienta de realidad aumentada (RA) que permite a los alumnos utilizar sus móviles como tarjetas dinámicas para estudiar, practicar o evaluar sus conocimientos en histología oral a su propio ritmo.

2.4. Dr. Doog, could you handle this for me?

Este juego fue diseñado como un juego de rol [1] para valorar la comunicación clínica dentista-paciente (CDPC, por sus siglas en inglés). El juego tiene en cuenta la motivación y eficiencia de los estudiantes para aprender sobre temas de comunicación clínica en el ámbito dental. Consta de 16 escenas que representan escenarios en los que los dentistas se encuentran por primera vez con pacientes dentales. Los jugadores tienen dos opciones para decidir cómo interactuar con los pacientes, lo que lleva a diferentes resultados. Las escenas están diseñadas para que los jugadores valoren cómo tratar con pacientes que sufren dolor extremo o tienen miedo, o pacientes mayores con antecedentes psicosociales.

2.5. Stud2yBuddy

Este juego fue diseñado como un juego de mesa [28] que gamifica la dermatología, haciendo un juego interactivo de aprendizaje y revisión que provee retroalimentación entre pares y capacidad de autoevaluar los conocimientos adquiridos.

2.6. Duolingo

Aplicación [26] educativa popular para móviles de aprendizaje de idiomas, que utiliza un enfoque gamificado para enseñar estos idiomas, presentando el aprendizaje como un juego divertido y desafiante.

La aplicación tiene una serie de lecciones interactivas que incluyen ejercicios de vocabulario, gramática y comprensión auditiva. También hay pruebas de progreso para evaluar el conocimiento del usuario y medir su progreso.

Además, ofrece la capacidad de conectarse con amigos y competir en actividades de aprendizaje de idiomas, como también la posibilidad de configurar recordatorios diarios para practicar un determinado idioma.

Capítulo 3: Tecnología Empleada

En este capítulo se describen las principales herramientas y tecnologías empleadas para desarrollar este proyecto.

3.1. Unity

Unity [6] es un motor de videojuego multiplataforma creado por Unity Technologies, utilizado para crear videojuegos de una amplia variedad de plataformas como PC y Android. Ofrece una amplia gama de herramientas y recursos para el desarrollo de videojuegos, incluyendo un editor visual para crear y editar escenas y soporte para scripts en C#, herramientas para el diseño de interfaces de usuario, sonidos, música y efectos visuales. En específico se empleó la versión estable 2020.3.17f1.

Una de sus características más destacadas es la capacidad que tiene para compilar el mismo proyecto en otras plataformas, lo que significa que los desarrolladores pueden crear un juego para una plataforma y luego compilarlo para que se ejecute en otras plataformas sin tener que reescribir el código por completo.

3.2. Strapi

Strapi [7] es un sistema de gestión de contenido (CMS) de código abierto creado por Strapi Solutions construido con una serie de tecnologías de código abierto como Node.js, MongoDB y React, y desarrollado con JavaScript, cuyo diseño permite a los desarrolladores crear, publicar y administrar contenido en una amplia variedad de aplicaciones y sitios web, incluyendo aplicaciones móviles.

Entre sus características más destacadas están:

- Una arquitectura modular que permite a los desarrolladores personalizar y extender la

plataforma según sus necesidades

- Una API flexible que permite personalizar los endpoints y los tipos de contenido según sus necesidades.
- Seguridad avanzada, incluyendo autenticación y autorización de usuario.
- Una fácil capacidad de integración con otras herramientas de desarrollo como SQL, herramientas de control de versiones como Git, etc.
- Es altamente escalable y se puede utilizar en proyectos grandes y pequeños. La plataforma se puede escalar horizontalmente para manejar grandes volúmenes de tráfico y contenido.

3.3. Postman

Postman [8] es una herramienta de desarrollo de API que permite crear y enviar solicitudes HTTP a una API para obtener una respuesta y verificar su funcionamiento en distintos formatos como JSON, XML y HTML.

3.4. MongoDB

MongoDB [9] es un sistema de gestión de base de datos NoSQL (no relacional) que se centra en la escalabilidad, el rendimiento y la facilidad de uso. A diferencia de otras bases de datos relacionales tradicionales que utilizan tablas y filas, MongoDB utiliza documentos en BSON (JSON binario) para almacenar y organizar datos.

Es especialmente adecuado para aplicaciones web y móviles que necesitan manejar grandes volúmenes de datos y escalar fácilmente en función del crecimiento de tráfico de los usuarios.

3.5. Visual Studio

Visual Studio [10] es un editor de código fuente desarrollado por Microsoft para múltiples sistemas operativos. Cuenta con soporte para depuración, resaltado de sintaxis, finalización inteligente de código, fragmentos y refactorización de código, entre otros. Es el editor por defecto

de Unity.

3.6. iTextSharp

iTextSharp [11] es una biblioteca de código abierto para aplicaciones .NET utilizado para crear, leer y modificar documentos de tipo PDF.

3.7. Newtonsoft

Newtonsoft [12] es una biblioteca de serialización de JSON para aplicaciones .NET que proporciona una serie de funciones y métodos para serializar objetos .NET y tipos anidados en cadenas JSON y viceversa.

3.8. REST Client

REST Client [18] es un cliente que utiliza el protocolo REST (Representational State Transfer) para realizar solicitudes y recibir respuestas a través de la API de Strapi. En este proyecto, se utiliza el protocolo REST tanto en Postman como en el propio proyecto Unity. En Postman se utiliza para probar y realizar solicitudes a la API, mientras que en el proyecto Unity se emplea para poder solicitar y recibir peticiones del servidor Strapi, autenticarse, etc..

3.9. SQLite

SQLite [17] es una biblioteca de enlace dinámico que se incorpora en el servidor Strapi, lo que significa que no requiere una instalación o configuración separada. Esta base de datos se enfoca en la gestión de bases de datos pequeñas y simples, y tiene una amplia compatibilidad con sistemas operativos populares como Windows, macOS, Linux o Android. Al utilizar SQLite, la aplicación puede garantizar una gestión eficiente y segura de los datos de los usuarios para serializar objetos .NET y tipos anidados en cadenas JSON y viceversa.

3.10. GitHub

GitHub [13] es una plataforma en línea para alojar y gestionar proyectos de software utilizando el sistema de control de versiones de Git. Permite a los desarrolladores alojar su código fuente de un proyecto y colaborar con otros desarrolladores en el mismo proyecto de forma paralela. Los proyectos se alojan en un repositorio junto a la documentación y recursos necesarios, y donde los desarrolladores pueden contribuir mediante la creación de *forks*, copias del repositorio original, donde pueden realizar sus caminos y luego solicitar que sean revisados y aceptados en el repositorio original.

Capítulo 4: Especificación de requisitos

Se han definido los siguientes tipos de usuario:

4.1. Actores

Dentro de la aplicación realizada se encuentran los siguientes tipos de usuarios:

- Usuario registrado (alumno): es el usuario que utilizará la aplicación, haciendo uso de las funcionalidades de esta. Sólo podrá hacer las pruebas y editar sus datos de perfil, unirse a grupos de trabajo y consultar sus calificaciones.
- Profesor: son usuarios registrados que además tienen permisos especiales para crear, modificar o eliminar grupos de trabajo, desbloquear pruebas para todos los alumnos de un grupo, organizar competiciones entre grupos o consultar las calificaciones de los usuarios registrados.

4.2. Módulos

Los casos de uso se han agrupado de acuerdo con los siguientes módulos funcionales:

- Módulo de gestión de usuarios
- Módulo de gestión de configuración
- Módulo de gestión de juegos
- Módulo de gestión de grupos
- Módulo de gestión de competiciones

A continuación, se describen cada uno de los módulos.

4.2.1. Módulo de gestión de usuarios

En este módulo están incluidas todas las funcionalidades de los usuarios, tanto alumnos como profesores, con funciones comunes como registrarse, iniciar sesión, y modificar sus datos de usuario. En las imágenes 4.1 y 4.2 se representa el diagrama de casos de uso relacionado con el módulo.

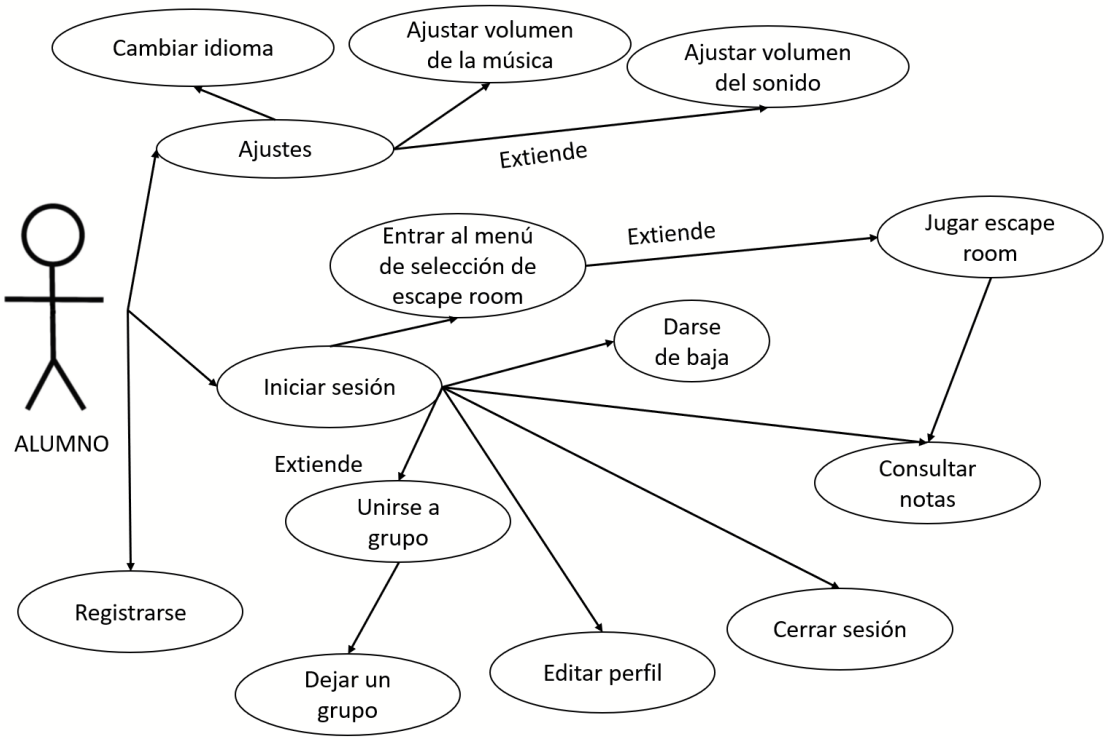


Imagen 4.1: Diagrama de casos de uso de alumno

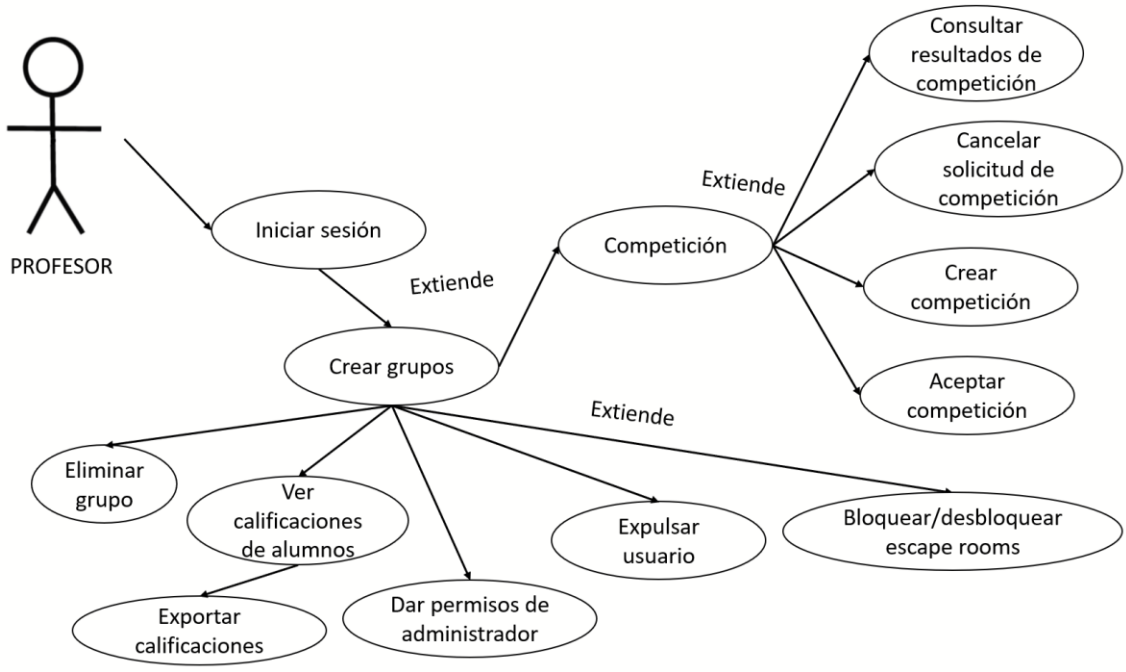


Imagen 4.2: Diagrama de casos de uso de profesor

A continuación, se describe cada caso de uso perteneciente a este módulo.

Requisito	**Registrarse**
Identificador	1.1
Prioridad	Alta.
Precondición	NA.
Descripción	Los usuarios crearán una cuenta en la aplicación introduciendo su nombre, apellidos, nombre de usuario y contraseña
Entrada	Nombre, Apellidos, Nombre de usuario, Contraseña
Salida	Nuevo usuario dado de alta en la aplicación.
Secuencia Normal	1. El usuario accede a la aplicación, y en el menú de inicio, hace clic en “Registrarse”.
	2. Se pasa a otro menú donde se le pide al usuario su nombre, apellidos, nombre de usuario y contraseña para poder registrarse.
	3. El usuario rellena los campos y hace clic en “Registrarse”.
	4. El sistema valida la información.
	5. Su cuenta es creada y avanza al siguiente menú.
Postcondición	El usuario se ha registrado con éxito en la aplicación.
Excepciones	El usuario deja algún campo vacío, en cuyo caso se le pedirá que lo complete.

Actores	Alumno, Administrador

Requisito	**Iniciar sesión**
Identificador	1.2
Prioridad	Alta.
Precondición	Haberse registrado previamente.
Descripción	Iniciar sesión en la aplicación introduciendo nombre de usuario y contraseña
Entrada	Nombre de usuario, Contraseña
Salida	Ninguna
Secuencia Normal	1. El usuario accede a la aplicación, y en el menú de inicio, hace clic en “Iniciar sesión”
	2. Se pasa a otro menú donde se le pide al usuario su nombre de usuario y contraseña para poder iniciar sesión.
	3. El usuario rellena los campos y hace clic en “Iniciar sesión”.
	4. El sistema valida la información.
	5. Se pasa al siguiente menú
Postcondición	NA
Excepciones	Los datos introducidos son incorrectos
Actores	Alumno, Administrador

Requisito	Darse de baja
Identificador	1.4
Prioridad	Media.
Precondición	Haber iniciado sesión
Descripción	Darse de baja de la aplicación y eliminar totalmente la cuenta del usuario.
Entrada	Interacción con botón, Nombre de usuario
Salida	La cuenta del usuario es eliminada del servidor
Secuencia Normal	1. Se accede al menú de perfil desde el menú principal
	2. Se pulsa en un botón para darse de baja
	3. Tras una ventana de confirmación y otra de introducir nombre de usuario para confirmar la acción, se da de baja al usuario con éxito.
Postcondición	NA
Excepciones	NA
Actores	Alumno, Administrador

Requisito	Editar perfil
Identificador	1.5
Prioridad	Media.
Precondición	Haber iniciado sesión

Descripción	Modificar cualquiera de los datos del usuario
Entrada	Nombre, Apellidos, Nombre de usuario, Contraseña
Salida	Ninguna
Secuencia Normal	1. En el menú de perfil se pulsa un botón para editar el perfil
	2. Aparecerá un menú para editar cualquiera de los datos del usuario reemplazando el campo del dato en cuestión por el nuevo dato.
	3. Una vez se termina de editar los datos se pulsa un botón “Guardar” y se vuelve al menú de perfil.
Postcondición	NA
Excepciones	El usuario deja algún campo vacío, en cuyo caso se le pedirá que lo complete.
Actores	Alumno, Administrador

4.2.2. Módulo de gestión de configuración

Este módulo agrupa todas las funcionalidades relacionadas con la configuración de distintas variables del juego, sobre todo relacionadas con el volumen del sonido.

A continuación, se describen cada uno de los casos de uso.

Requisito	**Modificar volumen de la música**
Identificador	2.1

Prioridad	Media.
Precondición	Haber accedido al menú de ajustes
Descripción	Aumentar o disminuir el volumen de la música que suena de fondo durante el juego mediante un slider.
Entrada	Interacción con el slider
Salida	Ninguna
Secuencia Normal	1. Se accede al menú de ajustes
	2. Se ajusta el slider correspondiente hasta que el volumen de la música de muestra sea el preferido por el usuario
Postcondición	NA
Excepciones	NA
Actores	Alumno, Administrador

Requisito	**Modificar volumen de los efectos de sonido**
Identificador	2.2
Prioridad	Media.
Precondición	Haber accedido al menú de ajustes
Descripción	Aumentar o disminuir el volumen de los efectos de sonido que se reproducen durante el juego mediante un slider.
Entrada	Interacción con el slider

Salida	Ninguna
Secuencia Normal	1. Se accede al menú de ajustes
	2. Se ajusta el slider correspondiente hasta que el volumen del efecto de sonido de muestra que sonará sea el preferido por el usuario
Postcondición	NA
Excepciones	NA
Actores	Alumno, Administrador

Requisito	**Cambiar idioma**
Identificador	2.3
Prioridad	Baja.
Precondición	Haber accedido al menú de ajustes
Descripción	Cambiar idioma de la aplicación entre español e inglés.
Entrada	Pulsación de botón
Salida	Ninguna.
Secuencia Normal	1. Se accede al menú de ajustes
	2. Se pulsa en el botón de “Cambiar idioma”.
	3. Se pulsa la bandera del idioma preferido por el usuario y se pulsa en “Aceptar”
	4. El idioma de la aplicación es cambiado con éxito

Postcondición	NA
Excepciones	NA
Actores	Alumno, Administrador

4.2.3. Módulo de gestión de juegos

Este módulo contiene las funcionalidades relacionadas con la interacción del usuario con las distintas pruebas a las que jugará, denominadas escape rooms, como con los menús que agrupan, y el sistema de calificación.

A continuación, se describen los casos de uso que forman este módulo.

Requisito	**Entrar al menú de selección de escape room**
Identificador	3.1
Prioridad	Alta.
Precondición	Haber iniciado sesión
Descripción	Menú que deja seleccionar al usuario cualquiera de las escape rooms disponibles a jugar.
Entrada	Pulsación de botón
Salida	Ninguna.
Secuencia Normal	1. El usuario pulsa “Jugar” en el menú principal
	2. Se pasa al menú de selección de escape room
Postcondición	NA
Excepciones	NA

Actores	Alumno, Administrador

Requisito	**Jugar escape room**
Identificador	3.2
Prioridad	Alta.
Precondición	Que ese escape room haya sido desbloqueado por un administrador o que el usuario sea un administrador
Descripción	Se escoge un escape room desbloqueado para poder jugar a ella
Entrada	Pulsación de botón
Salida	Ninguna.
Secuencia Normal	1. El jugador elige uno de los escape rooms disponibles pulsando su respectivo botón.
	2. Se procede al primer minijuego del escape room que ha elegido.
Postcondición	NA
Excepciones	NA
Actores	Alumno, Administrador

Requisito	**Ver calificaciones**
Identificador	3.3
Prioridad	Alta.
Precondición	Haber jugado al escape room previamente

Descripción	Mostrar las calificaciones de un escape room superado, medido tanto en rapidez como en aciertos.
Entrada	Interacción con botón
Salida	Ninguna
Secuencia Normal	1. El usuario verá sus calificaciones de alguna de las siguientes maneras:
	2. Tras completar esa prueba determinada
	3. Desde un botón de ver calificaciones en el menú principal y posteriormente escogiendo una prueba.
	4. Siendo un administrador, en un botón de ver calificaciones de un usuario concreto de su grupo.
Postcondición	NA
Excepciones	NA
Actores	Alumno, Administrador

Requisito	**Desbloquear/bloquear escape room**
Identificador	3.4
Prioridad	Alta
Precondición	Ser un administrador
Descripción	El administrador configura el escape room para que los alumnos puedan entrar.
Entrada	Pulsación de botón
Salida	NA

Secuencia Normal	1. Se desbloquea uno de los escape room de una de las 2 siguientes maneras:
	2. Pulsando un botón en el menú de grupo para desbloquear/bloquear un escape room concreto para todos los alumnos del grupo
	3. Seleccionando un alumno individual del grupo y desbloqueando/bloqueando el escape room solo para él.
Postcondición	NA
Excepciones	NA
Actores	Administrador

4.2.4. Módulo de gestión de grupos

Este módulo contiene las funcionalidades relacionadas con la gestión de los grupos de usuarios que incorpora la aplicación, y las opciones de administración de estos.

A continuación, se describen los casos de uso que forman este módulo.

Requisito	**Unirse a un grupo**
Identificador	4.1
Prioridad	Alta
Precondición	NA
Descripción	Un usuario se une a un grupo determinado
Entrada	Pulsación de botón, Contraseña del grupo
Salida	Ninguna

Secuencia Normal	1. Desde el menú principal el usuario pulsa en “Unirse a grupo”
	2. El usuario elige un grupo de una lista
	3. El usuario introduce la contraseña del grupo en un campo y le da a “Unirse”
	4. El usuario se une con éxito al grupo.
Postcondición	NA
Excepciones	NA
Actores	Alumno, Administrador

Requisito	**Dejar un grupo**
Identificador	4.2
Prioridad	Alta
Precondición	El usuario está en un grupo
Descripción	El usuario abandona un grupo
Entrada	Pulsación de botón
Salida	Ninguna.
Secuencia Normal	1. El usuario va al menú de grupos
	2. El usuario pulsa el botón de dejar el grupo.
	3. Tras una ventana de confirmación, el usuario abandona el grupo.
Postcondición	NA

Excepciones	NA
Actores	Alumno, Administrador

Requisito	**Crear grupo**
Identificador	4.3
Prioridad	Alta
Precondición	NA
Descripción	El Administrador crea un grupo, llenando el campo de Nombre
Entrada	Nombre del grupo
Salida	Ninguna
Secuencia Normal	1. El Administrador va al menú de Grupos y pulsa en el botón para crear un nuevo grupo
	2. El Administrador pone un nombre de grupo en el campo correspondiente
	3. El Administrador hace clic en “Crear nuevo grupo”
Postcondición	NA
Excepciones	Sí el campo de nombre de grupo se deja vacío o el nombre de grupo ya existe, el grupo no será creado
Actores	Administrador

Requisito	**Eliminar grupo**
Identificador	4.4

Prioridad	Media
Precondición	El administrador está en el grupo
Descripción	El administrador elimina un grupo
Entrada	Nombre del grupo
Salida	Ninguna
Secuencia Normal	1. El administrador va al menú de grupos
	2. Pulsa un botón de eliminar grupo.
	3. Tras ventanas de confirmación e introducir el nombre de grupo para confirmar, el grupo es eliminado con éxito.
Postcondición	NA
Excepciones	NA
Actores	Administrador

Requisito	**Expulsar usuario de grupo**
Identificador	4.5
Prioridad	Media
Precondición	El administrador está en el grupo
Descripción	Un administrador expulsa a un usuario del grupo
Entrada	Pulsación de botón
Salida	NA
Secuencia Normal	1. El administrador va al menú de grupos

	2. Entra en el menú de lista de miembros del grupo
	3. Busca al usuario que desea expulsar y pulsa un botón de expulsar usuario
	4. Tras una ventana de confirmación, el usuario es expulsado con éxito
Postcondición	Ninguna
Excepciones	No se puede expulsar a un administrador
Actores	Administrador

Requisito	**Dar permisos de administrador**
Identificador	4.6
Prioridad	Alta
Precondición	Ambos usuarios interesados están en el grupo
Descripción	Un administrador da a un alumno permisos de administrador y el alumno pasa a ser administrador
Entrada	Pulsación de botón
Salida	NA
Secuencia Normal	1. El administrador va al menú de grupos
	2. Entra en el menú de lista de miembros del grupo
	3. Busca al usuario que desea nombrar administrador y le da a un botón de dar permisos de administrador.
	4. Tras una ventana de confirmación, el alumno es nombrado administrador con éxito

Postcondición	Ninguna
Excepciones	Ninguna
Actores	Administrador

Requisito	**Exportar calificaciones**
Identificador	4.7
Prioridad	Media
Precondición	NA
Descripción	Exportar las calificaciones de los alumnos del grupo o de un alumno individual a un archivo de formato legible
Entrada	Pulsación de botón
Salida	Ninguna.
Secuencia Normal	1. El usuario va a la lista de miembros del grupo
	2. Se selecciona con tics a los alumnos que desea incorporar al archivo exportado
	3. Se pulsa un botón para exportar calificaciones y crear el archivo correspondiente.
Postcondición	NA
Excepciones	NA
Actores	Administrador

4.2.5. Módulo de gestión de competiciones

Este módulo contiene las funcionalidades relacionadas con la gestión de competiciones que permite crear la aplicación para enfrentar a distintos grupos de usuarios.

A continuación, se describen los casos de uso que forman este módulo.

Requisito	**Crear competición**
Identificador	5.1
Prioridad	Alta
Precondición	El usuario está en un grupo
Descripción	El administrador inicia una competición con otro u otros grupos, que puede aceptarla o rechazarla
Entrada	Pulsación de botón
Salida	Ninguna
Secuencia Normal	1. El administrador va al menú de grupos
	2. Va al menú de competiciones
	3. El administrador hace clic en el botón de “Crear competición”
	3. El administrador elige al grupo rival de una lista, y tras una ventana de confirmación, se envía una solicitud a ese grupo.
Postcondición	NA
Excepciones	NA
Actores	Administrador

Requisito	Aceptar competición
Identificador	5.2
Prioridad	Alta
Precondición	El usuario está en un grupo, tiene una solicitud de competición pendiente
Descripción	El administrador recibe una invitación para participar en una competición, que puede rechazar o aceptar
Entrada	Pulsación de botón
Salida	Ninguna
Secuencia Normal	1. El administrador entra en el menú de grupos
	2. Entra en el menú de competiciones
	3. Pulsa el botón de “Solicitudes pendientes” y pulsa un botón para aceptar o rechazar la competición que tiene pendiente.
Postcondición	NA
Excepciones	NA
Actores	Administrador

Requisito	Consultar resultados de competición
Identificador	5.3
Prioridad	Alta
Precondición	El usuario está en un grupo

Descripción	Se consulta la puntuación de los bandos de la competición y quien va ganando o ha ganado la competición sí está ya ha acabado.
Entrada	Pulsación de botón
Salida	Ninguna
Secuencia Normal	1. El usuario va al menú de grupos y posteriormente, al de competiciones.
	2. Pulsa el botón de consultar competiciones.
	3. Se mostrará una lista de todas las competiciones que han tenido lugar en el grupo y el resultado de cada una de ellas.
Postcondición	NA
Excepciones	NA
Actores	Alumno, Administrador

Requisito	**Cancelar solicitud de competición**
Identificador	5.4
Prioridad	Baja
Precondición	NA
Descripción	Cancelar una solicitud enviada de competición
Entrada	Pulsación de botón
Salida	Ninguna
Secuencia Normal	1. El usuario va al menú de grupos y posteriormente, al

	de competiciones.
	2. Pulsa el botón de “Solicitudes pendientes” y pulsa un botón para cancelar la solicitud de competición.
Postcondición	NA
Excepciones	NA
Actores	Administrador

Capítulo 5: Arquitectura de la aplicación

En este capítulo se describe la estructura de la aplicación y su modelo de datos.

5.1. Arquitectura de la aplicación

La aplicación hace uso de una arquitectura de tipo cliente-servidor [14], un modelo de diseño que consiste en una aplicación cliente que envía solicitudes a una aplicación servidor para acceder a datos o recursos compartidos. El servidor procesa las solicitudes y envía las respuestas al cliente, separando funcionalmente ambas partes, aunque estén en la misma máquina y comunicándose a través de una red local o Internet.

Los clientes utilizan la API proporcionada por Strapi para realizar sus solicitudes, lo que les permite consultar un servicio a través de peticiones HTTP [15] a través de un pequeño conjunto de operaciones como son POST, GET, PUT y DELETE y comunicarse con el servidor mediante información en formato JSON. [16].

En la imagen 5.1 se puede ver en forma de esquema una representación gráfica de la estructura de la aplicación.

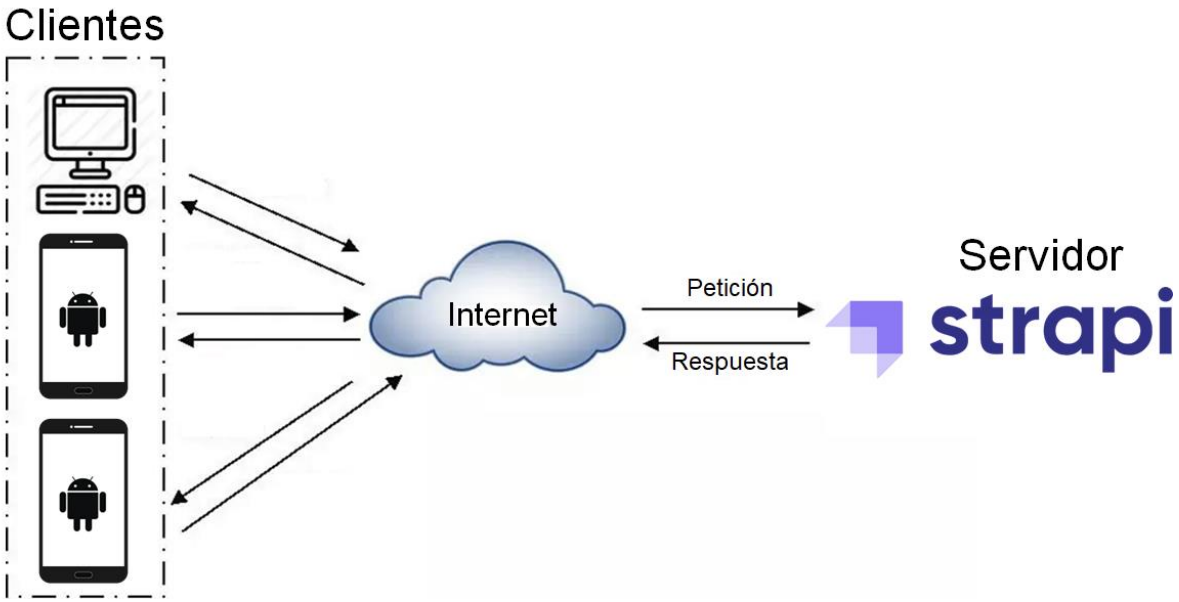


Imagen 5.1: Arquitectura de la aplicación

5.2. Modelo de datos

Para el funcionamiento de la aplicación, se necesitan almacenar varios tipos de datos en la base de datos. Esto incluye información de usuarios, grupos y minijuegos, que son los requisitos específicos de la aplicación. Estos datos luego se utilizan para acciones como autenticar a los usuarios, entre otras, y requieren que los datos estén disponibles constantemente para su uso continuo. Todo esto implica que los datos deben ser almacenados de manera estructurada y persistente en una base de datos. Como varios de los datos tienen relaciones establecidas entre tablas, utilizamos SQLite, una base de datos relacional liviana, que se implementa como una biblioteca en vez de como un servidor independiente. Se eligió SQLite por su capacidad de escalabilidad, útil en proyectos que no tendrán un crecimiento explosivo de datos o una gran cantidad de usuarios concurrentes, por su rendimiento rápido en operaciones de lectura y escritura, y por su disponibilidad, ya que SQLite es una base de datos incorporada y se accede directamente desde el sistema de archivos del servidor de la aplicación y no requiere un servidor de base de datos separado.

Strapi, como CMS (Content Management System), permite la definición de colecciones, las cuales son estructuras de datos que representan los tipos de contenido y sus campos, que siguen una

estructura similar a las tablas de una base de datos relacional. La terminología de ‘colecciones’ se utiliza en Strapi para describir la forma en que los datos se organizan y administran dentro del sistema de gestión de contenido, aunque en SQLite estos datos se organizan en tablas. Por lo tanto, en el contexto de Strapi, se utilizará la terminología ‘Colección’, aunque se trate de tablas en la base de datos.

Al crear un servidor Strapi localmente, la base de datos interna utilizada por Strapi es SQLite. Esta base de datos se almacena localmente, y se utilizará para gestionar los usuarios y las colecciones definidas en Strapi en el entorno de desarrollo. Además, la capacidad de almacenar cambios en el archivo de la base de datos SQLite garantiza la persistencia de datos y permite realizas consultas para interactuar con ellos.

Cada colección de Strapi representa un conjunto de datos relacionados almacenados en la base de datos. Además, Strapi permite establecer relaciones entre colecciones, como una relación uno a muchos, o muchos a muchos, entre otros. Estas relaciones se definen mediante campos de referencia en los tipos de contenido y se emplean para, entre otras cosas, acceder a datos relacionados a través de la API mediante filtros en las URL. Además, las relaciones permiten mantener la integridad referencial, ya que Strapi gestiona automáticamente dicha integridad, asegurando la coherencia de datos. Esto resulta especialmente útil para la gestión de equipos.

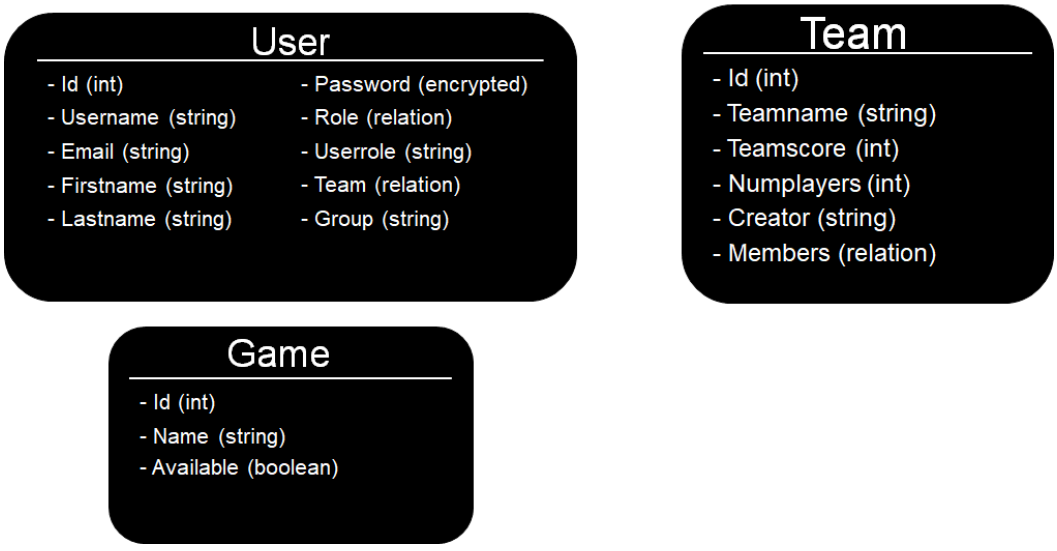


Imagen 5.2: Colecciones de datos Strapi

En la aplicación se emplean archivos serializables [22] JSON para traducir los modelos a código. Estos archivos leen las respuestas HTTP del servidor y transforman la información en datos tangibles de C#, como enteros, decimales, booleanos, matrices, entre otros.

Una vez seleccionado el tipo de base de datos y conocida la información a almacenar, se obtiene un modelo de datos como el visible en la imagen 5.2. Las colecciones utilizadas son las siguientes:

5.2.1. Colección User

La colección 'User' se utiliza para almacenar la información personal de los usuarios registrados en la aplicación. En ella se guardan datos como el nombre de usuario, correo electrónico, contraseña, rol (si es estudiante o profesor), nombre y apellidos, entre otros. Algunos de estos datos, de carácter privado, se protegen mediante la encriptación para evitar su acceso no autorizado. Los datos más importantes de esta colección son las credenciales (contraseña y nombre de usuario o correo electrónico). En esta colección también se definen dos relaciones: una relación uno-a-uno con el rol de usuario (alumno o profesor), y otra relación muchos-a-muchos con el equipo al que pertenezca. En caso de eliminarse este usuario, Strapi automáticamente gestionará que el equipo ya no tenga una relación con el usuario eliminado.

Cada registro en la colección representa a un usuario de la aplicación y está compuesto por un conjunto de atributos que definen sus características. Estos atributos se definen en la estructura de la tabla en la base de datos y se asocian a distintos tipos de datos según su naturaleza. Traduciendo estos atributos mediante los archivos serializables JSON a datos tangibles de C# permite utilizarlos en el código para modificarlos, crearlos o mostrarlos a los usuarios. De esta forma se puede interactuar con la información de la base de datos de una manera sencilla y eficiente.

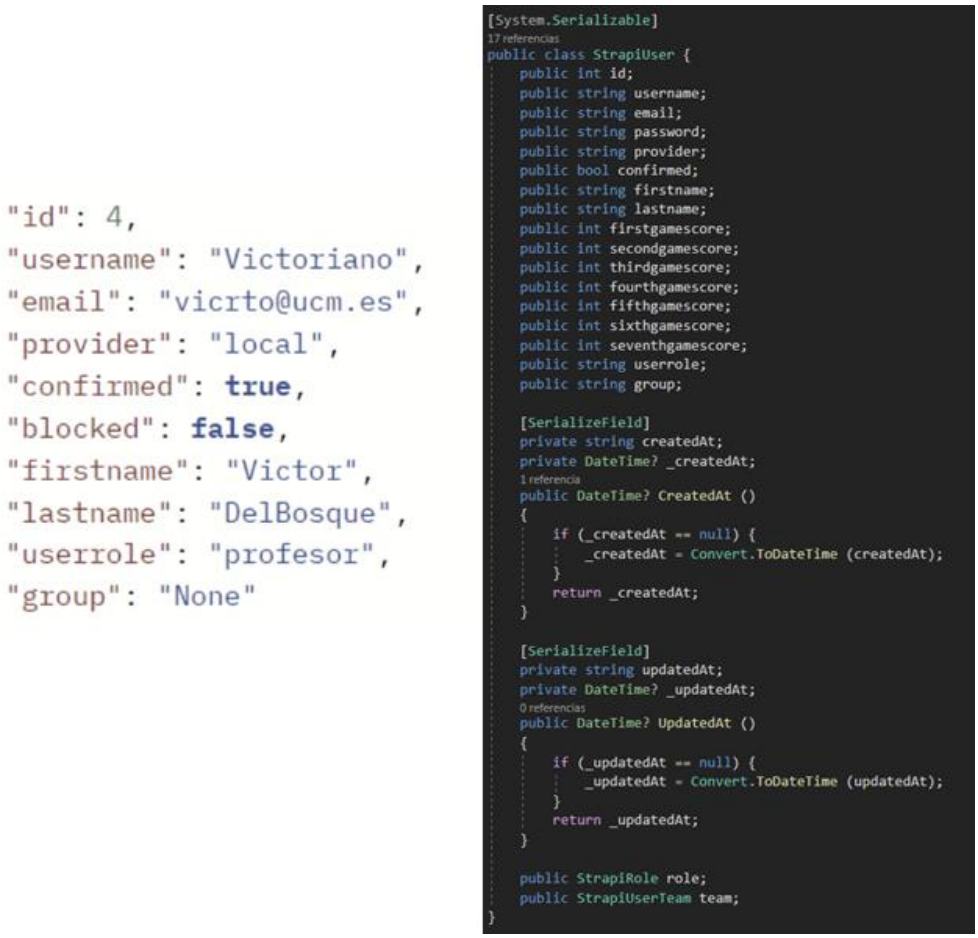


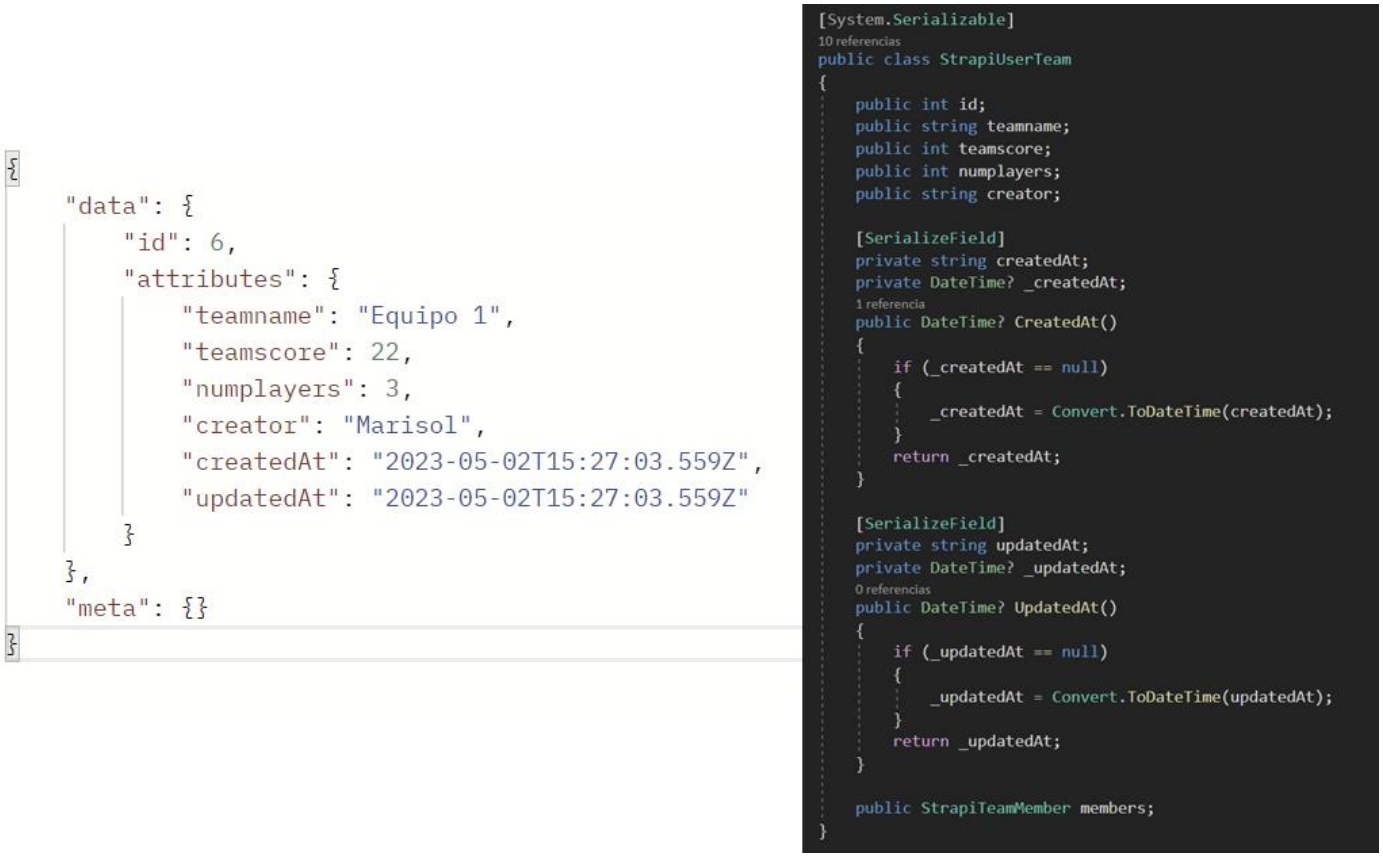
Imagen 5.3: Modelo de datos serializable y respuesta JSON con datos de usuario

5.2.2. Colección Team

La colección ‘Team’ se utiliza para almacenar la información de los equipos de usuarios en la aplicación. En ella se guardan datos como el nombre del equipo, el número de jugadores que lo conforman, el creador del equipo, entre otros.

Un atributo clave de esta colección es la lista de integrantes, que es una relación muchos-a-muchos entre el equipo y uno o varios usuarios. Esto permite filtrar usuarios utilizando estos atributos, en vez de tener que filtrarlos manualmente, lo que consume más recursos. Además, al gestionar automáticamente Strapi la congruencia de datos, eliminar el equipo hará que todos los usuarios queden liberados de tener un equipo.

De manera similar a la colección ‘User’, cada registro en esta colección representa un equipo de la aplicación, conformado por un conjunto de atributos que, empleando los archivos serializables JSON a datos tangibles de C#, permiten su modificación o creación por código.



```
{
    "data": {
        "id": 6,
        "attributes": {
            "teamname": "Equipo 1",
            "teamscore": 22,
            "numplayers": 3,
            "creator": "Marisol",
            "createdAt": "2023-05-02T15:27:03.559Z",
            "updatedAt": "2023-05-02T15:27:03.559Z"
        }
    },
    "meta": {}
}
```

```
[System.Serializable]
10 referencias
public class StrapiUserTeam
{
    public int id;
    public string teamname;
    public int teamscore;
    public int numplayers;
    public string creator;

    [SerializeField]
    private string createdAt;
    private DateTime? _createdAt;
    1 referencia
    public DateTime? CreatedAt()
    {
        if (_createdAt == null)
        {
            _createdAt = Convert.ToDateTime(createdAt);
        }
        return _createdAt;
    }

    [SerializeField]
    private string updatedAt;
    private DateTime? _updatedAt;
    0 referencias
    public DateTime? UpdatedAt()
    {
        if (_updatedAt == null)
        {
            _updatedAt = Convert.ToDateTime(updatedAt);
        }
        return _updatedAt;
    }

    public StrapiTeamMember members;
}
```

Imagen 5.4: Modelo de datos serializable y respuesta JSON con datos de un equipo

5.2.3. Colección Minigame

La colección ‘Minigame’ se encarga de almacenar información relacionada con los diferentes juegos que se ofrecen en la aplicación. En ella se almacena el nombre del minijuego y se registra si está disponible o no para que los estudiantes lo jueguen. Este último dato, un booleano llamado ‘available’ permitiría a los profesores bloquear o desbloquear niveles para que los alumnos los jueguen.

Como con las otras colecciones, esta información se almacena en una base de datos estructurada en forma de tabla. Cada registro representa un minijuego con los dos atributos antes mencionados.

Traduciendo esos atributos a datos tangibles de C#, los profesores podrían modificar el estado de disponibilidad del minijuego de forma cómoda y sencilla desde la interfaz.


```
"data": [
    {
        "id": 1,
        "attributes": {
            "name": "minijuego1",
            "available": true,
            "createdAt": "2023-05-02T15:30:57.561Z",
            "updatedAt": "2023-05-02T15:30:57.561Z"
        }
    },
    {
        "id": 2,
        "attributes": {
            "name": "minijuego2",
            "available": true,
            "createdAt": "2023-05-02T15:31:06.705Z",
            "updatedAt": "2023-05-02T15:31:06.705Z"
        }
    },
    {
        "id": 3,
        "attributes": {
            "name": "minijuego3",
            "available": true,
            "createdAt": "2023-05-02T15:31:15.027Z",
            "updatedAt": "2023-05-02T15:31:15.027Z"
        }
    }
```


Imagen 5.5: Respuestas JSON con datos de minijuegos

Capítulo 6: Implementación de la aplicación

A continuación, se exponen los detalles de la implementación y el diseño de la aplicación, agrupando las funcionalidades según el módulo al que pertenecen:

- Módulo gestión de usuarios
- Módulo gestión de grupos
- Módulo gestión de juego
- Módulo gestión de configuración
- Módulo gestión de competiciones

6.1. Interfaz de usuario

El diseño de IU se hizo tomando en cuenta los principios básicos del diseño de interfaz:

- La interfaz debe ser fácil de entender y predecible, es decir, coherente a través de toda la aplicación para que el usuario no se confunda. [20, p. 55].
- La interfaz debe ser fácil de usar, sin que los usuarios deban pensar demasiado. Todo debería estar como mucho a dos clics de distancia [20, p. 11].
- La interfaz debe ser clara, ofreciendo a los usuarios una retroalimentación clara y en tiempo real cuando interactúan con ella [20, p. 63].

En el diseño de interfaces, la tipografía juega un papel importante en la legibilidad y comprensión de la información presentada. Para ello, se aplican diversos principios para asegurar que la tipografía sea efectiva y coherente con el diseño general de la interfaz.

Uno de los principios clave es el principio de "Mientras más importante es, más notorio es" (Krug, 2014, p. 31). Este principio sugiere que, para destacar información importante, se debe aumentar su tamaño y/o su peso tipográfico. De esta manera, el usuario podrá identificar rápidamente la información que es relevante y que requiere su atención.

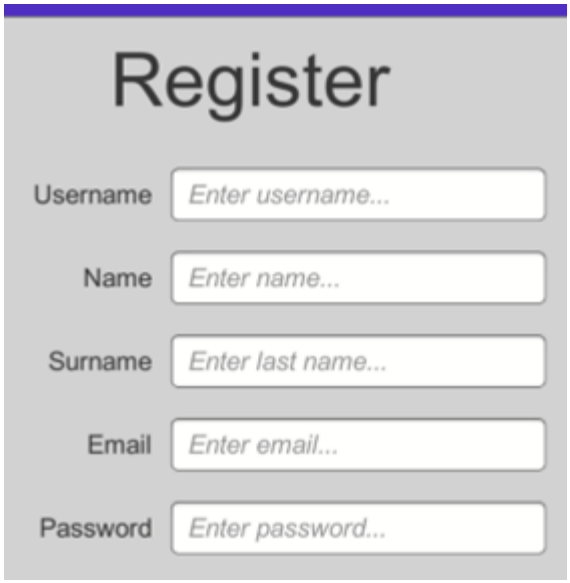


Imagen 6.1: Diferencia de tamaño de fuente entre datos de mayor y menor relevancia

También se utiliza el principio de "anidar" visualmente para mostrar pertenencia (Krug, 2014, p. 32), que implica que elementos relacionados o que pertenecen a un mismo grupo deben estar agrupados visualmente, ya sea mediante el uso de espacios en blanco, líneas o cualquier otro elemento visual que sugiera una conexión entre ellos. De esta manera, el usuario podrá comprender rápidamente la relación entre los distintos elementos presentados en la interfaz.

Imagen 6.2: Los usuarios están anidados en la lista, indicando pertenencia.

En el aspecto visual, se ha tomado la decisión de utilizar dos colores que contrasten entre sí, con el objetivo de evitar que distraigan o interrumpan la armonía de la interfaz. Además, se ha buscado crear un ambiente relajado mediante los valores de contraste entre los tonos, y el valor y/o saturación de los dos colores combinados [19, p. 68 – 69].

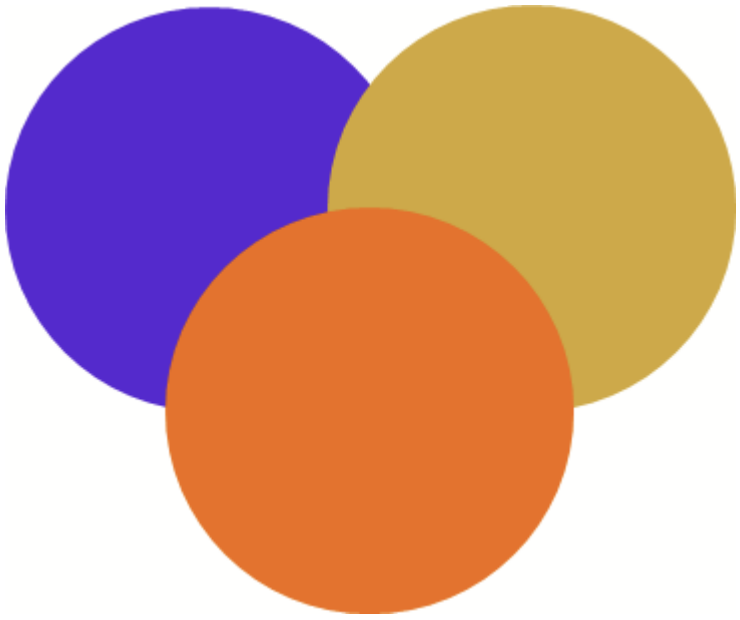


Imagen 6.3: Paleta de colores principales de la interfaz

Para las acciones peligrosas, se ha utilizado un enfoque diferente, utilizando colores que instiguen la duda, con el fin de que los usuarios sean conscientes de las posibles consecuencias negativas de dichas acciones. Esta técnica se utiliza para fomentar la precaución y la atención por parte del usuario [21].

Imagen 6.4: Uso de distintos colores en botones

6.2. Módulo de gestión de usuarios

6.2.1. Iniciar sesión

La aplicación inicia con una vista de inicio de sesión, representado en la imagen 6.5., la cual presenta al usuario dos campos de entrada de texto para introducir su nombre de usuario o correo electrónico y su contraseña. Si las credenciales introducidas son inválidas, se muestra un mensaje de error en la pantalla. En caso contrario, el usuario realiza una petición POST al servidor de Strapi para acceder a la aplicación, lo que se representa en la imagen 6.6.

La lógica detrás de la función de inicio de sesión se muestra en la imagen 6.7. Primero, se obtienen los valores introducidos por el usuario en los campos de entrada de texto. Luego se verifica si los campos no están vacíos. De ser así, se muestra un mensaje de error correspondiente. Caso contrario, se realiza una petición POST al servidor de Strapi con los datos de inicio de sesión del usuario. Si las credenciales son válidas, el usuario es redirigido a la vista principal de la aplicación. Caso contrario, se muestra un mensaje de error al usuario.

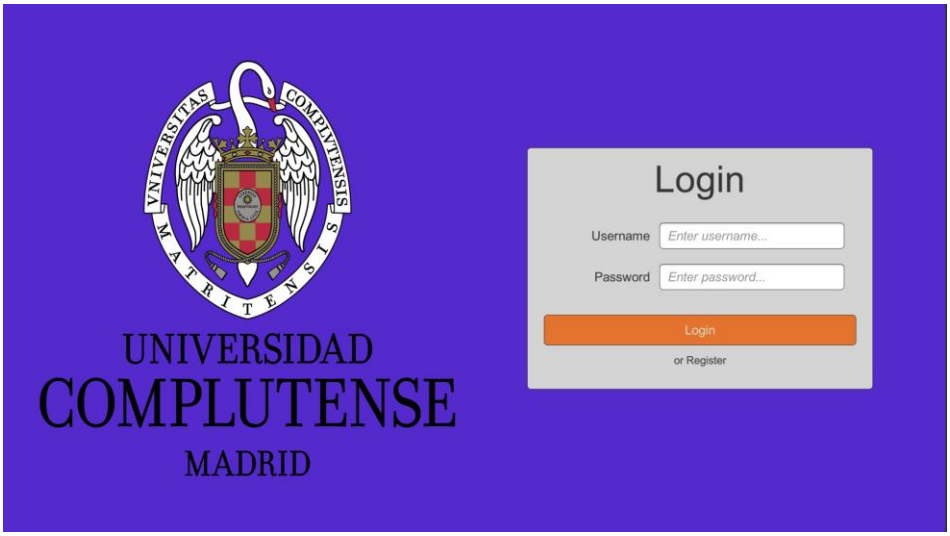


Imagen 6.5: Pantalla de login

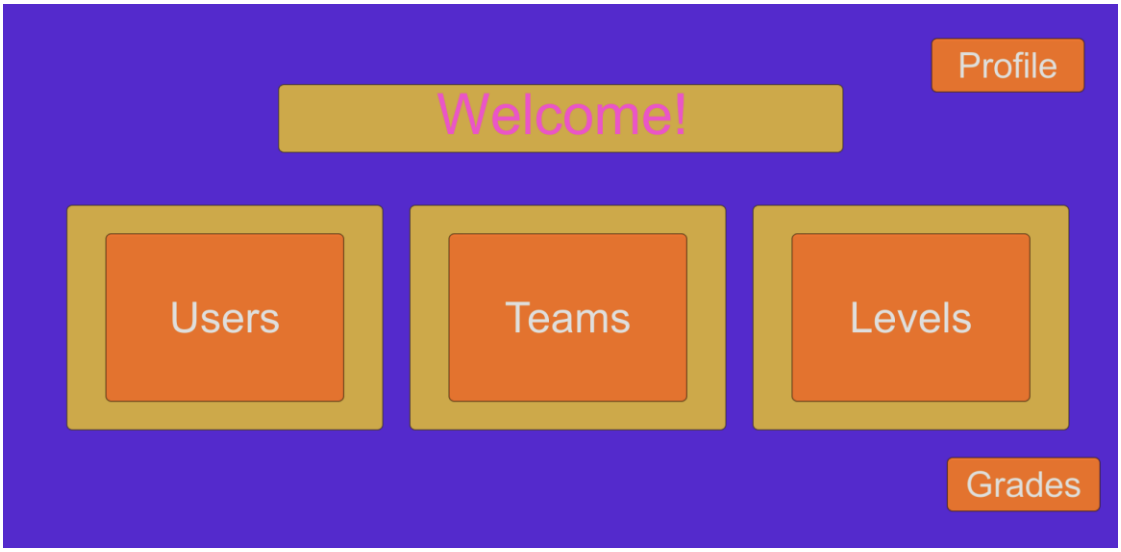


Imagen 6.6: Vista del menú principal

```
public void Login(string username, string password)
{
    OnAuthStarted?.Invoke();

    var jsonObject = new JObject(
        new JProperty("identifier", username),
        new JProperty("password", password)
    );
    var jsonString = jsonObject.ToString();

    string endpoint = "api/auth/local";
    StartCoroutine(PostAuthRequest(endpoint, jsonString));
}
```

```
private IEnumerator PostAuthRequest(string endpoint, string jsonString) {
    AuthResponse response = null;
    OnAuthStarted?.Invoke();
    var request = RestClient.Post<AuthResponse>(BaseURL + endpoint, jsonString).Then(authResponse =>
    {
        response = authResponse;
        OnAuthSuccess?.Invoke(authResponse);
    }).Catch(err =>
    {
        OnAuthFail?.Invoke(err);
        Debug.Log($"Authentication Error: {err}");
    });

    yield return new WaitUntil(() => response != null);
}
```

Imagen 6.7: Fragmentos de código - Login

6.1.2. Registrarse

La aplicación tiene dos opciones de registro para los usuarios no registrados: ‘Register’ y ‘Sign up as teacher’ (imagen 6.8). Al elegir el primero, los usuarios serán dirigidos a un formulario de registro que solicita información personal, como nombre de usuario, nombre y apellidos, correo electrónico y contraseña. En caso de que algún campo obligatorio no se complete, el usuario no podrá proceder con la operación. Los profesores tienen la opción de registrarse como tales seleccionando la segunda opción. Esta opción les pedirá una contraseña secreta adicional.

Si se completan todos los campos correctamente, se enviará una solicitud POST al servidor y se añadirá la información personal del usuario a la tabla de usuarios. Posteriormente, el usuario será dirigido a la pantalla del menú principal. Si el usuario registrado es un profesor, se necesitará hacer una petición PUT adicional para modificar su rol de alumno a profesor. Todos estos métodos se pueden observar en la imagen 6.9.

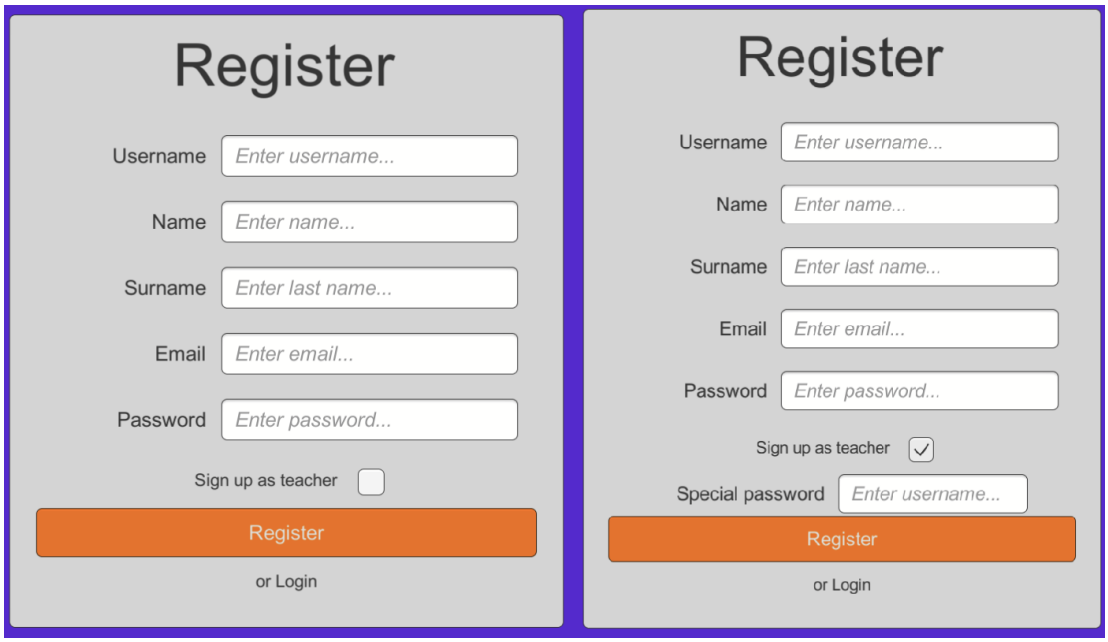


Imagen 6.8: Pantalla de registro de usuario

```
2 referencias
public void Register(string username, string name, string surname, string email, string password, bool isTeacher = false, string specialPassword = "")
{
    if (isTeacher)
    {
        if (specialPassword == TEACHER_SECRET_PASSWORD)
        {
            StartCoroutine(RegisterCoroutine(username, name, surname, email, password, true));
            userIsTeacher = true;
        }
        else Debug.Log("Wrong secret password");
    }
    else
    {
        StartCoroutine(RegisterCoroutine(username, name, surname, email, password, false));
    }
}
```

```
private IEnumerator RegisterCoroutine(string username, string name, string surname, string email, string password, bool isTeacher)
{
    OnAuthStarted?.Invoke();
    var jsonObj = new JObject(
        new JProperty("username", username),
        new JProperty("email", email),
        new JProperty("password", password),
        new JProperty("firstname", name),
        new JProperty("lastname", surname),
        new JProperty("userrole", "estudiante")
    );

    string jsonString = jsonObj.ToString();
    yield return PostAuthRequest("api/auth/local/register", jsonString);

    if (isTeacher)
    {
        yield return ChangeUserRole(profesorRoleID, userID);
    }
    else yield break;
}
```

```
private IEnumerator ChangeUserRole(int roleID, string userID)
{
    string endpoint = $"api/users/{userID}";

    var jsonObject = new JObject();
    jsonObject.Add("role", new JValue(roleID));
    jsonObject.Add("userrole", "profesor");

    string jsonString = jsonObject.ToString();

    yield return PutRequest(endpoint, jsonString);
}
```

Imagen 6.9: Los tres métodos empleados para registrarse

6.1.3. Darse de baja

En la aplicación, se ha implementado una funcionalidad que permite a los usuarios eliminar sus cuentas. Para acceder a esta opción, los usuarios deben ir al menú de edición de perfil, al que se accede desde el menú principal. Ahí (imagen 6.10), los usuarios encontrarán un botón llamado 'Delete Account'. Al hacer clic en este botón, se enviará una solicitud DELETE al servidor que eliminará todos los datos asociados con la cuenta del usuario, incluyendo la información de perfil, los datos de juego y cualquier otra información almacenada en la base de datos del servidor.

Para realizar esta tarea, se ha implementado un método específico en el servidor, como se puede observar en la imagen 6.11. Este método se encarga de recibir la solicitud DELETE y eliminar todos los datos asociados con la cuenta del usuario. Una vez que se completa la eliminación, la aplicación se reinicia, para que el usuario sin registrar pueda entrar con otra cuenta o crear una nueva.

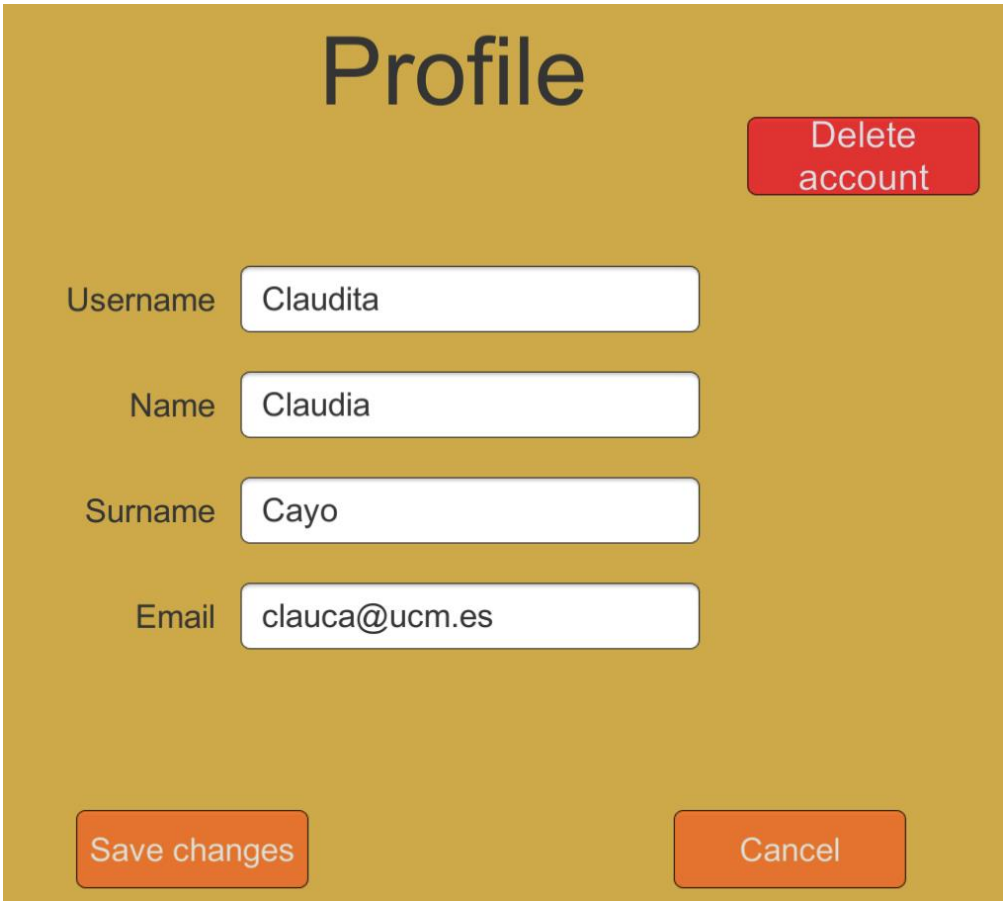


Imagen 6.10: Modificar evento

```
public void DeleteAccount()
{
    OnAuthStarted?.Invoke();
    string endpoint = $"api/users/{authenticatedUser.id}";
    StartCoroutine(DeleteRequest(endpoint));
}
```

```
private IEnumerator DeleteRequest(string endpoint)
{
    string url = BaseURL + endpoint;
    Debug.Log(url);
    RestClient.Delete(BaseURL + endpoint).Then(deleteResponse =>
    {
        Debug.Log($"Successfully deleted data at {endpoint}");
    })
    .Catch(err =>
    {
        Debug.LogError($"Delete error request {err}");
    });

    yield return 0;
}
```

Imagen 6.11: Fragmento de código - DeleteAccount

6.1.4. Editar perfil

En el menú de la aplicación, que se puede observar en la imagen 6.6, los usuarios tienen la opción de modificar su información de perfil personal. Esta opción les permite cambiar detalles como su nombre, dirección de correo electrónico, número de teléfono, entre otros. Para realizar cambios en los datos de perfil, se utiliza una petición PUT, que actualiza los valores en la base de datos del servidor.

El método, que se muestra en la imagen 6.12., se encarga de enviar la petición PUT, que se emplea para enviar los datos de actualización al servidor. A través de esta petición, los usuarios pueden enviar los cambios realizados en su información de perfil. Una vez que la petición PUT es recibida por el servidor, los nuevos valores se actualizarán en la base de datos y estarán disponibles cuando se soliciten al servidor.

```
public void EditProfile(string username, string email, string name, string surname)
{
    OnAuthStarted?.Invoke();
    JObject jsonObject = new JObject();

    if (!string.IsNullOrEmpty(username) && username != authenticatedUser.username)
    {
        jsonObject.Add("username", username);
    }

    if (!string.IsNullOrEmpty(email) && email != authenticatedUser.email)
    {
        jsonObject.Add("email", email);
    }

    if (!string.IsNullOrEmpty(name) && name != authenticatedUser.firstname)
    {
        jsonObject.Add("firstname", name);
    }

    if (!string.IsNullOrEmpty(surname) && surname != authenticatedUser.lastname)
    {
        jsonObject.Add("lastname", surname);
    }

    if (jsonObject.Count > 0)
    {
        string id = authenticatedUser.id.ToString();
        string endpoint = $"api/users/{id}";

        string jsonString = JsonConvert.SerializeObject(jsonObject);

        StartCoroutine(PutRequest(endpoint, jsonString, () =>
            StartCoroutine(GetUpdatedUserData(endpoint))));
    }
    else
    {
        Debug.Log("No changes submitted");
        return;
    }
}
```

Imagen 6.12: Fragmento de código - EditProfile

6.2. Módulo de gestión de grupos

6.2.1. Crear un grupo

La funcionalidad de añadir o eliminar usuarios de un grupo es exclusiva para los usuarios con rol de Profesor. En la interfaz de usuario, esto se logra a través de botones específicos en el menú de gestión de equipo (imagen 6.13). Al hacer clic en estos botones, se selecciona una lista de usuarios a añadir o eliminar del grupo. Posteriormente, se realizan las solicitudes correspondientes al servidor para realizar los cambios necesarios.

Esta funcionalidad se basa en la selección de usuarios y la actualización tanto del equipo involucrado como de los usuarios afectados. La implementación de esta funcionalidad se puede ver en un fragmento de código en la imagen 6.14.

Imagen 6.13: Menú de creación de equipo

```
private IEnumerator CreateStrapiUserTeamCoroutine(string groupName, List<int> members)
{
    int score = 0;
    foreach (int memberId in members)
    {
        int userIndex = Array.FindIndex(users, user => user.id == memberId);
        if (userIndex >= 0)
        {
            score += GetUserTotalPoints(userIndex);
        }
    }

    var teamObject = new JObject(
        new JProperty("teamname", groupName),
        new JProperty("creator", authenticatedUser.username),
        new JProperty("numplayers", new JValue(members.Count)),
        new JProperty("teamscore", new JValue(score)),
        new JProperty("members", JArray.FromObject(members)));

    string jsonString = JsonConvert.SerializeObject(new { data = teamObject });
    string endpoint = "api/teams";

    yield return StartCoroutine(PostGroupRequest(endpoint, jsonString));
    UpdateSelectedPlayersCoroutine(members);
}
```

Imagen 6.14: Fragmento de código - Crear un equipo

6.2.2. Unirse o abandonar un grupo

La aplicación tiene una funcionalidad para que los usuarios con rol de Profesor puedan añadir o eliminar usuarios de un grupo mediante botones en el menú de gestión de equipo. Esto se logra internamente seleccionando una lista de usuarios que se desea añadir o eliminar, haciendo las solicitudes correspondientes en el servidor y actualizando el equipo en cuestión, así como todos los usuarios involucrados en la operación. En la imagen 6.15 se pueden ver los botones correspondientes y en la imagen 6.16 se muestra una parte de la lógica detrás de esta función.

Imagen 6.15: Menú de gestión de grupo

```
private IEnumerator UpdateStrapiUserTeamCoroutine(int teamID, List<int> members, bool addingPlayers) {
    string endpoint = $"api/teams/{teamID}?populate=members";
    yield return GetGroupDataRequest(endpoint);

    int newGroupSize = strapiUserTeam.numplayers;
    if (!addingPlayers) newGroupSize -= members.Count;
    else newGroupSize += members.Count;

    if (newGroupSize < 0) newGroupSize = 0;

    int newScore = strapiUserTeam.teamscore;

    List<int> copy = new List<int>(members);
    for (int i = 0; i < users.Length; i++)
    {
        int id = users[i].id;
        if (members.Contains(id))
        {
            if (!addingPlayers) newScore -= GetUserTotalPoints(users[i]);
            else newScore += GetUserTotalPoints(users[i]);
            copy.Remove(id);
        }
        if (copy.Count <= 0)
            break;
    }
    if (newScore < 0) newScore = 0;

    List<int> copy = new List<int>(members);
    for (int i = 0; i < users.Length; i++)
    {
        int id = users[i].id;
        if (members.Contains(id))
        {
            if (!addingPlayers) newScore -= GetUserTotalPoints(users[i]);
            else newScore += GetUserTotalPoints(users[i]);
            copy.Remove(id);
        }
        if (copy.Count <= 0)
            break;
    }
    if (newScore < 0) newScore = 0;
```

```
    List<int> newMemberList = new List<int>();
    if (!addingPlayers)
    {
        for (int j = 0; j < strapiUserTeam.members.data.Length; j++)
        {
            int id = strapiUserTeam.members.data[j].id;
            if (!members.Contains(id))
            {
                newMemberList.Add(id);
            }
        }
    }
    else
    {
        for (int j = 0; j < strapiUserTeam.members.data.Length; j++)
        {
            int id = strapiUserTeam.members.data[j].id;
            members.Add(id);
        }
        newMemberList = members;
    }

    var teamObject = new JObject(
        new JProperty("numplayers", new JValue(newGroupSize)),
        new JProperty("teamscore", new JValue(newScore)),
        new JProperty("members", JArray.FromObject(newMemberList)));

    string jsonString = JsonConvert.SerializeObject(new { data = teamObject });

    yield return StartCoroutine(PutGroupRequest(endpoint, jsonString));

    string group;
    if (!addingPlayers) group = "None";
    else group = strapiUserTeam.teamname;

    var userObject = new JObject(
        new JProperty("group", group));
    jsonString = JsonConvert.SerializeObject(userObject);
    for (int i = 0; i < members.Count; i++)
    {
        endpoint = $"api/users/{members[i]}";

        if (members[i] == int.Parse(userID))
        {
            StartCoroutine(PutRequest(endpoint, jsonString,
                () => StartCoroutine(GetUpdatedUserData(endpoint))));
        }
        else StartCoroutine(PutRequest(endpoint, jsonString));
    }
}
```

Imagen 6.16: Fragmento de código - Añadir o eliminar usuarios de un grupo

6.2.3. Eliminar un grupo

En el menú de gestión de equipos, los usuarios tienen la opción de eliminar un equipo en su totalidad. Al hacerlo, se realizarán una serie de acciones que incluyen la eliminación de todos los registros asociados a ese equipo en la base de datos y la liberación de todos los usuarios que pertenecían a ese equipo. Una vez que se ha eliminado el equipo, todos los usuarios que formaban parte de este serán marcados como "libres" en la base de datos. Esto significa que estarán disponibles para unirse a otros equipos o para crear un nuevo equipo si así lo desean. Se puede ver una parte del código relevante en la imagen 6.17.

```
public void DeleteGroup(int id)
{
    OnAuthStarted?.Invoke();
    string endpoint = $"api/teams/{id}";

    StrapiUserTeam team = GetGroupByID(id);

    for (int i = 0; i < team.members.data.Length; i++)
    {
        int userId = team.members.data[i].id;
        string userEndPoint = $"api/users/{userId}";

        var userObject = new JObject(
         new JProperty("group", "None"));

        string jsonString = JsonConvert.SerializeObject(userObject);

        StartCoroutine(PutRequest(userEndPoint, jsonString));
    }

    StartCoroutine(DeleteRequest(endpoint));
}
```

Imagen 6.17: Fragmento del código - Eliminar un equipo

6.2.4. Exportar calificaciones

En la aplicación, los profesores pueden descargar las calificaciones de los alumnos en un archivo PDF para tener un registro de su desempeño en los minijuegos. Para hacerlo existe un botón 'Grades' en el menú principal, como se indica en la imagen 6.18, que activa una función de exportación. Al hacer clic en el botón, el sistema ejecutará la función correspondiente, de la cual se puede ver un fragmento en la imagen 6.19.

Esta función acepta una lista de objetos 'Alumno' que contiene información sobre los datos de interés de cada alumno. Utilizando la librería iTextSharp, se crea un archivo PDF en la ruta 'Mis Documentos' con el nombre 'calificaciones.pdf'.

Primero se crea un objeto *Document* con el tamaño de página A4, se establece el escritor PDF en la ubicación especificada del sistema de archivos y se abre el documento. Luego, se agregan metadatos al documento, como el título y creador. Después, se crea una tabla PdfPTable con cuatro columnas que representan el nombre completo, el correo electrónico, la puntuación total y las estadísticas de minijuegos. Se recorre la lista de alumnos y se agregan las celdas correspondientes a cada fila de la tabla. Finalmente, se agrega la tabla al documento y se cierran el documento y el escritor de PDF.

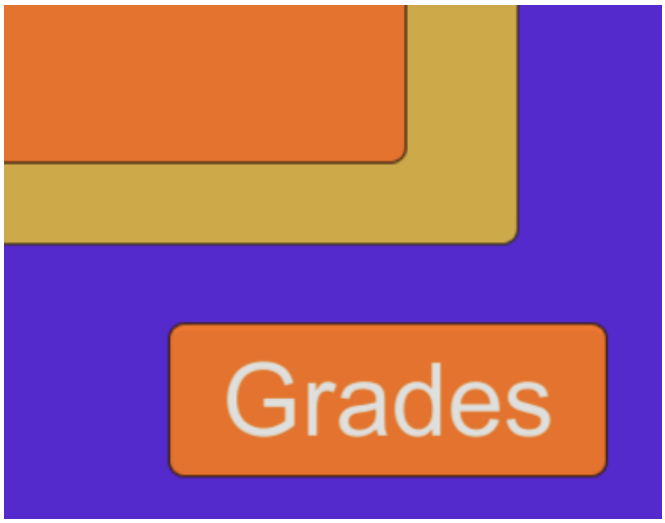


Imagen 6.18: Botón de exportar calificaciones en el menú principal

```
private IEnumerator GetNotesFromServerCoroutine(string endpoint)
{
    yield return GetListOfUsersFromServerCoroutine(endpoint);
    Document document = new Document();
    string filePath = Path.Combine(Environment.GetFolderPath(Environment.SpecialFolder.UserProfile) + "\\Downloads", "notas.pdf");
    PdfWriter.GetInstance(document, new FileStream(filePath, FileMode.OpenOrCreate));
    document.Open();

    iTextSharp.text.Font font = FontFactory.GetFont(FontFactory.HELVETICA_BOLD, 12, BaseColor.BLACK);
    for (int i = 0; i < users.Length; i++)
    {
        string line = "Usuario " + users[i].id + " " + users[i].firstname + " " + users[i].lastname +
            " " + users[i].username + " " + users[i].email + ".\n";

        Chunk chunk = new Chunk(line, font);

        Paragraph paragraph = new Paragraph(chunk);
        document.Add(paragraph);

        chunk = new Chunk("Bloque 1.\n", font);

        paragraph = new Paragraph(chunk);
        document.Add(paragraph);

        line = "Minijuego 1: " + users[i].firstgamescore + ".\n";
        paragraph = new Paragraph(line);
        document.Add(paragraph);
```

Imagen 6.19: Fragmento de código - Método para exportar calificaciones a un PDF

6.3. Módulo de gestión de juego

6.3.1. Código recurrente

En la aplicación existen varios scripts que se utilizan en todos los minijuegos, con mayor o menor importancia en el funcionamiento de la aplicación. Entre estos destacan:

- **StrapiComponent:** es el script más importante de la aplicación. Utiliza la biblioteca Proyecto26, un proyecto de tipo REST Client, un cliente que utiliza el protocolo REST para realizar solicitudes y recibir respuestas a través de la API de Strapi. Con esta biblioteca, el proyecto es capaz de solicitar peticiones de tipo GET, POST, PUT, DELETE, para interactuar con la base de datos, realizar operaciones CRUD (Create, Read, Update, Delete), autenticar usuarios y realizar otras operaciones. Utilizando estos datos de la base de datos de Strapi, la aplicación actualiza los distintos menús y datos de los jugadores, grupos o minijuegos.
- **GameManager:** un script que se utiliza para guardar las puntuaciones de los jugadores en todos los juegos, utilizando un diccionario que guarda los aciertos, fallos y puntos extras en una estructura *Minigame_Score*, cuya clave es el índice del juego. Al terminar el juego, actualiza la base de datos del jugador con su puntuación total en ese juego específico.
- **DragWithMouse:** un script que se utiliza para arrastrar un GameObject por la pantalla utilizando físicas. Sirve tanto para sistemas Windows, utilizando el input del ratón, como para Android, utilizando el input táctil de los móviles.
- **Countdown:** un script que se usa para tener un contador en la parte superior central de la pantalla, indicando cuánto tiempo le queda al jugador. Al llegar a cero, el minijuego actual terminará automáticamente. Siempre que acabe el minijuego, se le darán al jugador hasta 3 puntos adicionales, en función a cuánto tiempo le queda en el contador todavía al terminar.
- **CamerasManager:** un script que utiliza un array de cámaras y permite cambiar la cámara principal para ciertas transiciones o animaciones durante los juegos.
- **MySceneManager:** un script que se usa para cargar una escena específica.
- **PauseMenu:** un script sencillo que maneja la interfaz del menú de pausa, activando o desactivando cuando se pulsa el botón de pausa.
- **TransitionScreenLogic:** tras terminar un minijuego se llama a la escena de transición, que

ejecuta este script, imprimiendo en pantalla los resultados del jugador en el juego que acaba de terminar.

6.3.2. Minijuego 1

El primer minijuego consiste en una habitación con elementos interactivos como cajas y una bola. El objetivo es encestar la bola en las cajas hasta que las plataformas conectadas a estas, visibles en el fondo, estén a la misma altura. Cada bola tiene adjunta una imagen que el jugador puede ver usando el botón de inspección, y las cajas están divididas en tipos de patologías en las que las imágenes se pueden clasificar.

La lógica detrás del juego se encuentra en un script llamado *escapeRoom1_1_logic* que se encarga de manejar varios aspectos del juego. En él se utiliza el CameraManager para alternar entre las cámaras cuando el jugador encesta una bola, cortando a su respectiva torre bajando de altura, y al terminar, se cambia a la cámara de la puerta, mostrando que se ha terminado el juego. También se encarga de comprobar si las respuestas del jugador son correctas o no, y de hacer reaparecer la bola cuando el jugador encesta, así como de elegir la imagen que va adjunta a la bola.

Al llenar todas las cajas o acabarse el tiempo, el jugador habrá completado el objetivo del juego y realiza la transición a la escena de resultados.

```
1 referencia
public void CheckAnswer(string containerName)
{
    string imageType = currentImage.GetComponent<ShowImage>().GetTextureName();
    if (imageType == containerName)
    {
        GameManager.instance.CorrectAnswer();
    }
    else
    {
        GameManager.instance.WrongAnswer();
    }

    Destroy(currentImage);
    CreateImagePrefab();
}

2 referencias
private void CreateImagePrefab()
{
    currentImage = Instantiate(prefab, transform.GetChild(0).position, Quaternion.identity);
    currentImage.GetComponent<ShowImage>().SetPanel(panel);

    // ponemos la textura al panel
    Texture2D texture = null;
    PickRandomTexture(out texture);
    currentImage.GetComponent<ShowImage>().SetTexture(texture);
    picturesSpawned--;
}
```

Imagen 6.20: Fragmento de código

6.3.3. Minijuego 2

El segundo minijuego es un juego de memoria y habilidad visual en el que el jugador debe emparejar una imagen de patología con su correspondiente causa. La pantalla muestra varias imágenes y el jugador controla un cubo con la imagen de una patología en particular. El jugador debe mover el cubo hacia las otras imágenes y tratar de emparejar con la correcta.

El script *escape_room1_2_player* es responsable de la lógica del juego y funciona de la siguiente manera: primero, asigna aleatoriamente las imágenes a los cubos, incluyendo al del jugador, modificando sus texturas. Luego, por cada colisión del jugador con uno de los cubos, el script verifica si la respuesta es correcta o incorrecta. Si la respuesta es correcta, el jugador gana puntos y se le asigna una nueva imagen, y si es incorrecta, pierde puntos. También se maneja el CameraManager para alternar entre la cámara del juego y la de la puerta cuando se responde correctamente o se acaba el juego.

Al terminar el juego, sea por tiempo o tras emparejar todas las imágenes, se pasa a la escena de transición.

```
private void OnCollisionEnter(Collision col)
{
    if(col.gameObject.tag == "image" + (currentQuestion + 1).ToString())
    {
        // We update the player's texture
        GetComponent<Renderer>().material.mainTexture = questions[currentQuestion];
        currentQuestion++;
        GameManager.instance.CorrectAnswer();
        TurnLightOff();
        Destroy(col.gameObject);
    }
    else if (col.gameObject.tag != "wall")
    {
        GameManager.instance.WrongAnswer();
    }
}
```

Imagen 6.21: Fragmento de código de las colisiones

6.3.4. Minijuego 3

El tercer minijuego de Escape Room es un juego de preguntas y respuestas que es manejado

por el script llamado *escaperoom1_3_questions_logic*. Este script tiene la responsabilidad de gestionar toda la lógica del juego. Para hacerlo, recibe dos arrays de respuestas y preguntas, empezando por el índice 0.

La interfaz de usuario es actualizada por el script para mostrar la pregunta actual y recibir la respuesta propuesta por el jugador. Si la respuesta es correcta, el juego avanza a la siguiente pregunta, pero si es incorrecta, un mensaje es mostrado para informar al jugador.

El juego termina cuando se llega al último índice de la última pregunta y se ha contestado correctamente. Durante el juego, el script gestiona la actualización de los elementos de la IU y mantiene un seguimiento de la puntuación del jugador. Si todas las respuestas son correctas, el jugador gana puntos, y si se contesta mal alguna pregunta, pierde puntos.

Al terminar el juego, sea por tiempo o tras responder todas las preguntas, se pasa a la escena de transición.

```
public void OnSubmit()
{

    if (index == questions.Length - 1 && !endgame)
    {
        endgame = true;
        StartCoroutine(EndGame());
    }

    if (!endgame)
        if (checkAnswer(answerInputField.text, index))
        {
            CorrectAnswer();
            Debug.Log("Indice " + index);
        }
        else
        {
            feedbackText.GetComponent<Text>().color = Color.red;
            feedbackText.text = wrongAnswerMessage;
            GameManager.instance.WrongAnswer();
        }
        answerInputField.text = "";
}

1 referencia
public void CorrectAnswer()
{
    StartCoroutine(CorrectAnswerCoroutine());
}
```

Imagen 6.22: Fragmento de código

6.3.5. Minijuego 4

El script escaperoom1_4_questions es responsable de manejar el minijuego 4 de la aplicación. Este minijuego consiste en un cilindro de 5 letras que se obtienen al responder 5 preguntas con 4 posibles respuestas cada una, donde cada respuesta asigna una letra de la A, a la D. El script se encarga de llevar el registro de las preguntas y respuestas, actualizar la interfaz gráfica con las preguntas y respuestas, registrar las respuestas seleccionadas por el usuario, manejar la lógica de aciertos y errores, y actualizar la puntuación del usuario.

El script cuenta con varios métodos, entre los cuales destaca NextQuestion, que comienza el juego y se encarga de actualizar la interfaz con la siguiente pregunta y sus posibles respuestas. Al terminar el juego, sea por tiempo o tras responder todas las preguntas, se pasa a la escena de transición.

```
void Awake()
{
    door.transform.GetComponent<Animator>().enabled = false;

    line1 = gameObject.transform.GetChild(0).GetComponent<TextMeshPro>();
    line2 = gameObject.transform.GetChild(1).GetComponent<TextMesh>();
    line3 = gameObject.transform.GetChild(2).GetComponent<TextMesh>();
    line4 = gameObject.transform.GetChild(3).GetComponent<TextMesh>();
    line5 = gameObject.transform.GetChild(4).GetComponent<TextMesh>();

    NextQuestion();
}

Mensaje de Unity | 0 referencias
private void FixedUpdate()
{
    if (countdown.GetTimeLeft() <= 0.0f && !gameEnded)
    {
        gameEnded = true;
        EndGame();
    }
}

2 referencias
void NextQuestion()
{
    currentQuestion++;
    if (currentQuestion < 5)
    {
        line1.text = question[currentQuestion];
        line2.text = answer1[currentQuestion].Split('|')[0];
        line3.text = answer2[currentQuestion].Split('|')[0];
        line4.text = answer3[currentQuestion].Split('|')[0];
        line5.text = answer4[currentQuestion].Split('|')[0];
    }
    else
    {
        EndGame();
    }
}
```

Imagen 6.23: Fragmento de código

6.3.6. Minijuego 5

El script *escaperoom1_5_player* es utilizado para manejar un minijuego que consiste en una pregunta con tres posibles respuestas, representadas como caminos a elegir. El jugador puede elegir cualquiera de las tres puertas, cuyo valor como respuesta viene indicado en la parte superior, para continuar hasta acabar. El jugador también puede leer la pregunta y ver la imagen adjunta para poder responder.

Destaca el método SetRoom, que utiliza el índice de preguntas para actualizar la interfaz con la pregunta actual y sus posibles respuestas.

Al terminar el juego, sea por tiempo o tras responder todas las preguntas, se pasa a la escena de transición.

```
private IEnumerator SetRoomCoroutine(int currentIndex)
{
    yield return FadeToBlack(1.5f);
    int answersIndex = currentIndex * 3;
    room.transform.GetChild(0).GetChild(0).GetComponent<escaperoom1_5_door>().SetAnswer(answers[answersIndex]);
    room.transform.GetChild(0).GetChild(1).GetComponent<escaperoom1_5_door>().SetAnswer(answers[answersIndex + 1]);
    room.transform.GetChild(0).GetChild(2).GetComponent<escaperoom1_5_door>().SetAnswer(answers[answersIndex + 2]);

    room.transform.GetChild(6).GetComponent<Renderer>().material.mainTexture = textures[currentIndex];
    room.transform.GetChild(7).GetChild(0).GetComponent<TextMeshPro>().text = questions[currentIndex];

    // Child 0 of this GO must always be an empty GO
    room.transform.parent = transform.GetChild(0);
}

1 referencia
private IEnumerator FadeToBlack(float time)
{
    fadeToBlackPanel.SetActive(true);

    fadeToBlackPanel.GetComponent<Image>().color = new Color(0, 0, 0, 0);

    // Gradualmente incrementar la opacidad del color de la imagen del panel
    float timer = 0;
    SoundManager.instance.PlaySound(4);
    while (timer < time)
    {
        float alpha = Mathf.Lerp(0, 1, timer / time);
        fadeToBlackPanel.GetComponent<Image>().color = new Color(0, 0, 0, alpha);
        timer += Time.deltaTime;
        yield return null;
    }

    fadeToBlackPanel.SetActive(false);
}
```

Imagen 6.24: Fragmento de código

6.3.7. Minijuego 6

Este minijuego es una actividad en la que se debe ordenar las letras de dos palabras que se han revuelto por accidente, arrastrándolas a sus posiciones correctas. La tarea se realiza arrastrando las letras a sus posiciones correspondientes en la pantalla.

El script principal se llama *escaperoom1_6*, donde al inicio de la ejecución, en el método OnStart, se ordenan aleatoriamente las letras de las palabras. Utilizando el método SubmitAnswer, se verifica que las cajas estén en su posición correcta, marcando visualmente las que están bien y las que están mal a modo de retroalimentación para el jugador.

Al terminar el juego, sea por tiempo o tras ordenar todas las letras correctamente, se pasa a la escena de transición.

```
void Start()
{
    door.transform.GetComponent<Animator>().enabled = false;

    answer = prompt.Replace(" ", "");
    answerLenght = answer.Length;
    words = prompt.Split(' ');

    int numLetters = words.Sum(w => w.Length);
    int posY = Screen.height / 2;
    int posX = Screen.width / 2;
    int offset = separationVal / 2;

    boxes = new List<GameObject>();

    int totalWordLength = words.Sum(w => w.Length);
    int startX = posX - (totalWordLength / 2) * separationVal + offset;

    int totalWordHeight = words.Length * separationVal * 2;
    int startY = posY + (totalWordHeight / 2) - separationVal - offset;


    for (int i = 0; i < words.Length; i++)
    {
        int wordLength = words[i].Length;
        int wordStartX = startX + (totalWordLength - words[i].Length) / 2 * separationVal;
        float wordStartY = startY - i * separationVal * 2 + (words.Length - 1 - i) * (Screen.height / 2 - startY) / (words.Length - 1);

        // Convertimos la palabra en una lista de caracteres para poder eliminar las letras utilizadas
        List<char> letters = words[i].ToList();

        for (int j = 0; j < wordLength; j++)
        {
            GameObject box = Instantiate(boxPrefab, new Vector3(wordStartX + j * separationVal, wordStartY, 0), Quaternion.identity, transform.GetChild(0).transform);

            // Seleccionamos aleatoriamente una letra de la lista y la eliminamos
            int randomIndex = Random.Range(0, letters.Count);
            char letter = letters[randomIndex];
            letters.RemoveAt(randomIndex);

            box.transform.GetChild(0).GetComponent<TextMesh>().text = letter.ToString();
            boxes.Add(box);
        }
    }
}
```

Imagen 6.25: Fragmento de código

6.3.8. Minijuego 7

Este minijuego toma inspiración del Pasapalabra. Consiste en una serie de preguntas, cada una de las cuales tiene o comienza con una letra determinada del alfabeto, y el jugador puede responderla correcta o incorrectamente, o bien pasar a la siguiente e intentarlo nuevamente en la siguiente vuelta. Tras pasar dos vueltas el juego termina, o bien si se acaba el tiempo, o si se responden todas las preguntas, independientemente de si están correctas o incorrectas.

Este minijuego está formado por dos scripts importantes. El primero, *escape_room1_7_questions* gestiona todo lo relacionado a las preguntas. Lee el archivo JSON con las preguntas y respuestas, y es utilizado por el segundo script, *escape_room1_7_circle*, que gestiona el juego propiamente. Con un índice interno maneja en qué pregunta está el jugador, ofrece la opción de responder o saltar una pregunta, y actualiza la interfaz con la pregunta actual. También verifica que se cumpla cualquiera de los parámetros de fin del juego.

```
private void SpawnCircle()
{
    float angleIncrement = 360f / numberOfLetters;

    for(int i=0;i<numberOfLetters;i++)
    {
        float angle = (270f + i * angleIncrement) * Mathf.Deg2Rad;
        Vector3 position = new Vector3(Mathf.Cos(angle), 0f, Mathf.Sin(angle)) * radius;
        letters[i] = Instantiate(letter, transform.position + position, Quaternion.identity);
        letters[i].transform.SetParent(transform);
        letters[i].transform.GetChild(1).GetComponent<TextMesh>().text = circleLetters[i].ToString();
    }

    transform.Rotate(Vector3.right, 90f);
    LoadLetterMaterial(0);
    letters[activeLetter].GetComponent<Renderer>().material = activeColor;

    // Para las preguntas
    index = 0;

    lettersJSON = questionsManager.GetComponent<escaperoom1_7_questions>().GetLetters();
    hints = questionsManager.GetComponent<escaperoom1_7_questions>().GetHints();
    questions = questionsManager.GetComponent<escaperoom1_7_questions>().GetQuestions();
    answers = questionsManager.GetComponent<escaperoom1_7_questions>().GetAnswers();

    hint.text = hints[index];
    question.text = questions[index];
}
```

Imagen 6.26: Fragmento de código

6.4. Módulo de gestión de configuración

6.4.1. Modificar el volumen del juego

Cuando el usuario pulsa el botón de pausa en el juego, se mostrará una pantalla de pausa que incluye la posibilidad de ajustar el volumen del sonido, como se muestra en la imagen 6.27. En la pantalla de pausa, se mostrará un control deslizante que permite ajustar el volumen del juego, que se representará gráficamente en una barra de progreso. Al mover el control deslizante, se actualiza el valor de una variable que controla el sonido en la escena

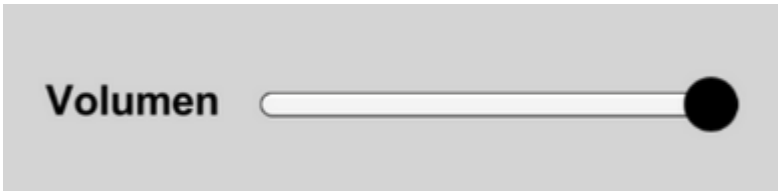


Imagen 6.27: Opción de ajuste de volumen

Capítulo 7: Conclusiones y trabajo futuro

7.1. Conclusiones

En este trabajo se ha desarrollado una aplicación móvil con una arquitectura de cliente-servidor para que los alumnos de la asignatura de Patología Médica Bucal de tercer curso de Odontología de la UCM puedan poner a prueba sus conocimientos mediante juegos interactivos. Del mismo modo, esta aplicación permite que el docente pueda evaluar esta actividad realizada por el alumno, en función de los aciertos y el tiempo consumido en realizarla. Para ello, se implementaron un conjunto de funcionalidades de usuario básicas (registrarse, iniciar sesión, darse de baja, etc.), un conjunto de funcionalidades para los profesores (crear/eliminar grupos, ver las notas de sus alumnos), y los juegos propiamente dichos, basados en ideas aportadas por los profesores de la asignatura, pero elaborados para poder ser utilizados tanto a través de una aplicación móvil como una aplicación de PC.

Todos los ficheros fuente del proyecto se encuentran alojados en el siguiente repositorio de GitHub: https://github.com/Dankgard/Odontogames

7.2. Trabajo futuro

Una buena parte de los objetivos y funcionalidades planteados al comienzo del proyecto se alcanzaron y desarrollaron, pero una parte no se pudo conseguir. Además, durante la realización del proyecto surgieron nuevas ideas a implementar que se proponen a continuación como trabajo futuro:

- **Gestión de competición:** Inicialmente se había planteado un sistema de competiciones entre grupos de usuarios en la aplicación, pero no pudo lograrse. Este sistema consistía en la capacidad de que dos equipos participaran en una competición, donde los alumnos jugarían uno de los juegos y al final el equipo que tuviera la mayor cantidad de puntos (sacada a partir de la cantidad de puntos de todos sus usuarios), sería el ganador.
- **Bloqueo de minijuegos:** Inicialmente se había planteado la posibilidad de una función en la que los profesores pudieran bloquear/desbloquear los minijuegos de la aplicación, pero no llegó a implementarse. Una solución sería usar la base de datos 'Minigame' y consultar por peticiones si el juego está bloqueado, o para bloquearlo, solicitar que se cambie su variable 'available'.
- **Cambiar de idiomas:** Actualmente, la aplicación está hecha únicamente en castellano, pero originalmente se había planteado que pudiera estar también en inglés.
- **Cambiar contraseña:** Actualmente, la funcionalidad de cambiar la contraseña de un usuario en la aplicación no está disponible debido a que requiere de la confirmación de correo electrónico por parte de Strapi. Para poder ofrecer esta opción a los usuarios, se podría expandir el servidor y añadir los plugin necesarios que permitan llevar a cabo esta tarea, o crear una nueva funcionalidad en el servidor que permita enviar una solicitud para cambiar la contraseña.
- **Implementar otros módulos de la materia:** Actualmente, la aplicación cuenta con siete minijuegos que forman parte del mismo módulo de la asignatura, pero se podrían crear más juegos de otros módulos en el futuro. Esto también requeriría algunos cambios en la estructura de las tablas del servidor, para acomodar mejor la creación de nuevos módulos.
- **Posibilidad de descargar las notas de un/unos alumnos específicos:** Actualmente, la aplicación exporta las calificaciones de todos los usuarios alumnos, sin

distinción alguna. Idealmente se elaboraría esta funcionalidad para que se pueda elegir alumnos específicos para exportar sus notas.

Conclusions and future work

Conclusions

For this project a mobile application has been developed with a client-server architecture so that students can test their knowledge by playing through a set of minigames. For this purpose, basic user functions (register, log in, delete account) and professor only functions (add/delete groups, check students results) have been provided. The minigames are mobile app adaptations of games previously played in class.

Project source code can be found on the following GitHub repository: https://github.com/Dankgard/Odontogames

Future work

Although most of the objectives and functionalities proposed at the beginning of the project through the requirements gathering phase have been achieved and developed, some of them were never completed, and new features have been found and are proposed below as future work too:

- **Competition module:** Competitions between groups was an initial idea that was never developed. In these competitions, students from each group would play a minigame to add points to their group global score, and the group with the most points would win in the end.
- **Minigames blocking:** Initially, the possibility of a feature where teachers could block/unblock the mini games in the application was considered, but it was not implemented. One solution would be to use the 'Minigame' database and query for requests to check if the game is blocked, or to block it, request a change in its 'available' variable.
- **Different languages:** The initial idea allowed users to select Spanish language or English language, but English language didn’t make it to the end.
- **Change password:** Currently, the function to change a user's password in the application

is not available as it requires email confirmation from Strapi. To offer this option to users, the server could be expanded, and necessary plugins added to perform this task, or a new function could be created on the server to allow for a password change request to be sent.

- **Extra minigame modules:** The final application has seven minigames from a single subject module, but more modules with extra minigames could be a possibility, by making changes in the server structure.
- **Allowing individual student results to be downloaded:** The final application allows the download of a file containing all student results from the same group, but being able to download individual student results could be a possibility in the future.

Capítulo 8: Aportaciones individuales

En este capítulo se explican las tareas realizadas por cada miembro del equipo en el proyecto.

8.1. Alejandro Plázer Arnaz

Las aportaciones de Alejandro Plázer Arnaz se definen en los siguientes puntos:

Toma de requisitos

Una de las partes más importantes del trabajo fue la búsqueda de requisitos, tarea que consistió en hacer una serie de propuestas de casos de usuario que se desarrollarían en el TFG. El alcance del trabajo se consiguió mediante reuniones en las que ambos miembros del equipo estaban presentes con los tutores, y también se les mandaron varias propuestas hasta llegar a un acuerdo.

Selección de tecnologías

Tras la toma de requisitos fue necesario llevar a cabo una investigación de las tecnologías que se iban a necesitar para desarrollar las funcionalidades acordadas. Esta parte se hizo en conjunto por ambos miembros del equipo, buscando opciones existentes que nos permitieran realizar el trabajo de una forma sencilla y eficaz.

Creación y gestión de la base de datos

Para la creación de la base de datos se estudió conjuntamente la estructura e información a almacenar. También se buscó cómo adaptar esa información al entorno Strapi, así como aprender a usar las herramientas que ofrecía y su API.

Tanto el diseño como el uso del servidor fueron cambiando a lo largo del proyecto mientras surgían eventos inesperados durante el desarrollo de la aplicación.

Diseño e implementación de la aplicación

Los casos de uso se desarrollaron primero con Postman para confirmar que estuvieran bien hechas las estructuras de las bases de datos. Después se crearon scripts serializables que fueran capaces de traducir la información de los archivos JSON respuesta del servidor a datos de C#, para

poder usarlos en el código. También se desarrollaron los métodos para mandar las peticiones y recibir las respuestas del servidor mediante el uso de un protocolo REST Client, usando corrutinas para que las funciones y operaciones se coordinaran entre sí, a modo de no pisarse o ejecutarse con datos desactualizados mientras las peticiones y respuestas seguían ejecutándose.

Respecto a la aplicación, se trabajó en conjunto el diseño de los juegos, pasando luego a programar las funcionalidades planificadas para los minijuegos, el uso de luces, audio, cámaras y animaciones, así como el entorno de las escenas y los elementos de la interfaz.

También se rediseñó la interfaz de usuario, buscando una paleta de colores que no contrastaran mucho entre sí, y se construyeron y diseñaron los distintos elementos de interfaz, como botones, tablas y menús.

Pruebas y corrección de errores

Durante el desarrollo del proyecto fue necesario la realización de pruebas y corrección de errores. Esta parte se hizo de forma conjunta por los dos miembros del equipo, de modo que las funcionalidades iban siendo probadas por el miembro del equipo que no las había desarrollado, con el objetivo de simular el uso que cualquier usuario de la aplicación haría.

Redacción de la memoria

La redacción de la memoria ha sido realizada de entre los dos miembros del equipo bajo la asistencia de los tutores del Proyecto de Fin de Grado.

8.2. Daniel García-Villarrubia López-Menchero

Las aportaciones de Daniel García-Villarrubia se definen en los siguientes puntos:

Toma de requisitos

Entre los dos miembros del grupo se plantearon los requisitos que debía cumplir el proyecto con tal de que cumpliera toda la funcionalidad propuesta. Esta lista de requisitos llegó a una conclusión con ayuda de los tutores mediante una serie de reuniones en las que se acordaban propuestas.

Selección de tecnologías

Tras la toma de requisitos fue necesario realizar una investigación de las tecnologías que se iban a necesitar para desarrollar las funcionalidades acordadas. Esta labor se realizó conjuntamente por ambos miembros del equipo, comparando las opciones existentes que permitiesen trabajar de manera cómoda y eficaz.

Creación y gestión de la base de datos

Para la creación de la base de datos se estudió conjuntamente por ambos miembros del equipo la estructura e información a almacenar. El diseño a su vez fue cambiando a lo largo del desarrollo de la aplicación a medida que se fueron encontrando particularidades que no habían sido previstas.

Diseño de los conceptos de los minijuegos

Tomando como referencia unas ideas de minijuegos propuestas en papel, se diseñaron versiones jugables de ellos, que se fueron adaptando a propuestas de los profesores y a pruebas realizadas de los mismos.

Diseño e implementación de la aplicación

Se trabajó en el desarrollo de gran parte de los módulos de la aplicación, pero sobre todo implementación de minijuegos, opciones de configuración de la aplicación, escenas de transición, sistema de audio de la aplicación e interfaz de usuario.

Pruebas y corrección de errores

Durante el desarrollo del proyecto fue necesaria la realización de pruebas y corrección de errores. Esta parte se hizo de forma conjunta por los dos miembros del equipo, de modo que las funcionalidades iban siendo probadas por el miembro del equipo que no las había desarrollado, con el objetivo de simular el uso que cualquier usuario de la aplicación haría.

Redacción de la memoria

La redacción de la memoria ha sido realizada de entre los dos miembros del equipo bajo la asistencia de los tutores del Proyecto de Fin de Grado.

Bibliografía

[1] Lin CS, Yang CC (2023). Evaluation of a digital game for teaching behavioral aspects of clinical communication in dentistry. BMC Med Educ. PMID: 36721149; PMCID: PMC9889244.

[2] Jia JL, Shen A, Tabata MM, Sarin KY (2020). Gamification improves melanoma visual identification among high school students: Results from a randomized study. Pediatr Dermatol. PMID: 32266730.

[3] Pereira AC, Dias da Silva MA, Patel US, Tanday A, Hill KB, Walmsley AD (2022). Using quizzes to provide an effective and more enjoyable dental education: A pilot study. Eur J Dent Educ. PMID: 34510674.

[4] Zheng M, Ferreira L (2020). Gamification to enhance online learning engagement. J Dent Educ. PMID: 32589759.

[5] Van Gaalen AEJ, Jaarsma SC, Georgiadis JR (2021). Medical Students' Perception of Play and Learning: Qualitative Study With Focus Groups and Thematic Analysis. JMIR Serious Games. PMID: 34319237; PMCID: PMC8367104.

[6] Unity

[7] Strapi

[8] Postman

[9] MongoDB

[10] Visual Studio

[11] iTextSharp

[12] Newtonsoft

[13] GitHub

[14] Arquitectura cliente-servidor

[15] Peticiones HTTP

[16] JSON

[17] SQLite

[18] REST Client

[19] Krause, J (2006). Color for Designers: Ninety-five things you need to know when choosing and using colors for layouts and illustrations. Rockport Publishers

[20] Krug, S (2000). Don't Make Me Think: A Common Sense Approach to Web Usability. New Riders.

[21] Olyslager, P (2017). Call To Action Buttons and the Psychology of Color. Recuperado de: https://www.paulolyslager.com/call-to-action-buttons-psychology-color/

[22] Serialización

[23] Dental School

[24] Bonebox – Dental Lite

[25] Sharmin N, Chow AK, Votta D, Maeda N (2002). Implementing Augmented Reality to Facilitate the Learning of Oral Histology. Healthc Inform Res. PMID: 35576985; PMCID: PMC9117805

[26] Duolingo

[27] Ludificación

[28] Ventre R, Pardoe C, Singhal S, Cripps D, Hough J (2019). Gamification of dermatology: Stud2yBuddy, a novel game to facilitate dermatology revision for final-year medical students. Future Healthc J. PMID: 31572920; PMCID: PMC6752424

Anexo A: Guía de uso

En este anexo se explica cómo hacer uso de la aplicación móvil tanto para usuarios profesores como alumnos.

A.1. Pantalla de inicio de sesión/registro de usuario

La aplicación cuenta con una pantalla de inicio de sesión, como se muestra en la imagen A.1, la cual es accesible para todos los usuarios. Para acceder a la aplicación, el usuario debe proporcionar su nombre de usuario o correo electrónico y su contraseña.

En la imagen A.2 se muestra la pantalla de registro de usuario, en la cual se solicitan los campos necesarios para la creación de una cuenta. Si el usuario es un profesor, deberá seleccionar el botón *Sign up as teacher*, el cual desplegará un campo adicional para ingresar una contraseña secreta requerida. Esto garantiza que sólo los profesores puedan registrarse como tal y acceder a las funcionalidades específicas de su rol en la aplicación..

Imagen A.1: Vista de la pantalla de inicio de sesión

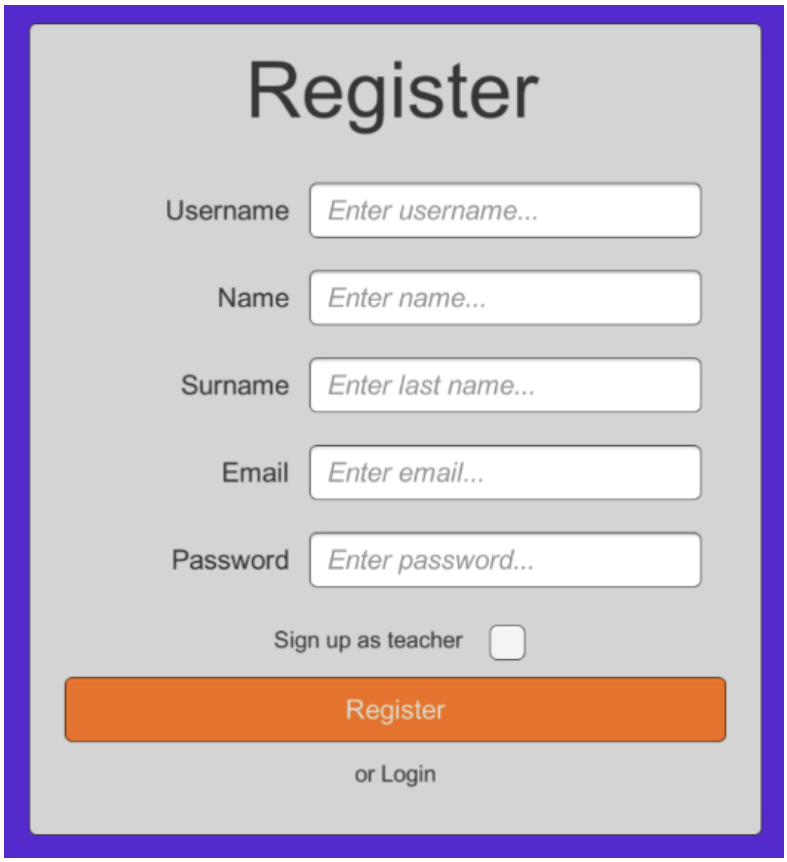


Imagen A.2: Vista de la pantalla de registro

A.2. Menú principal

El menú principal de la aplicación es una pantalla de transición que se muestra al iniciar la aplicación y permite acceder a las distintas secciones de esta. Este menú está representado en la imagen A.3, y cuenta con cuatro botones identificados por un breve texto, los cuales llevan a los menús de usuarios, equipos, minijuegos y perfil del usuario.

Además, si se trata de un usuario profesor, se añade un quinto botón adicional que le permite descargar las notas de los usuarios alumnos en un archivo PDF. De esta manera, el profesor puede revisar y analizar las notas de sus estudiantes de manera rápida.

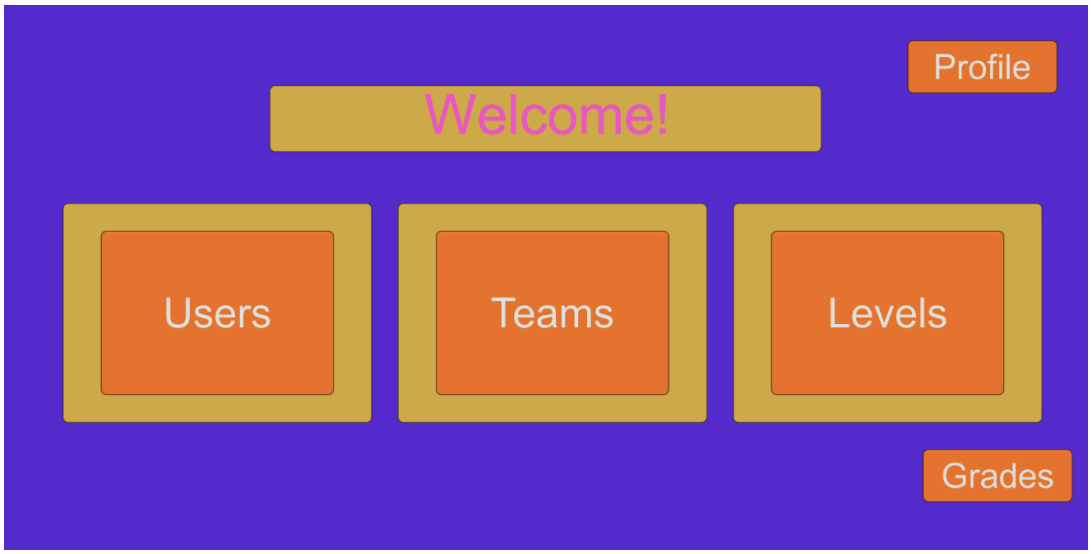


Imagen A.3: Pantalla de menú principal

A.3. Menú de usuarios de la aplicación

El menú de gestión de usuarios de la aplicación es una interfaz gráfica que permite ver un listado de todos los usuarios registrados en la aplicación. La lista incluye información básica de cada usuario, que son su nombre y nombre de usuario, como se ve en la imagen A.4. Al hacer clic en uno de los usuarios, se abre una ventana emergente, como se muestra en la imagen A.5 que muestra información adicional del usuario, como su correo electrónico y equipo al que pertenecen.

Imagen A.4: Listado de usuarios de la aplicación

Imagen A.5: Vista individual de un usuario

A.4. Menú de gestión de grupos

El menú de gestión de grupos se representa en la imagen A.6, que consiste solamente en una lista de todos los grupos que existan en la base de datos. Pulsar en cualquiera de los equipos sacará un mini menú como el de la imagen A.7, que muestra los datos básicos del grupo.

Si el usuario tiene el rol de profesor, tendrá algunas más opciones que el usuario alumno. Para empezar, en el mini menú de equipo, tiene la opción de eliminar el grupo completamente, y añadir nuevos jugadores o eliminar jugadores que pertenezcan al equipo, como se ve respectivamente en las imágenes A.8 y A.9. Finalmente, tienen la opción de crear un nuevo grupo pulsando en el botón + a la izquierda de la pantalla, donde se les pedirá un nombre con un límite mínimo de 3 caracteres, como se ve en la imagen A.10, y luego se seleccionarán a los miembros del grupo, en una pantalla similar a la de la imagen A.8.

Imagen A.6: Listado de grupos

Imagen A.7: Datos del grupo

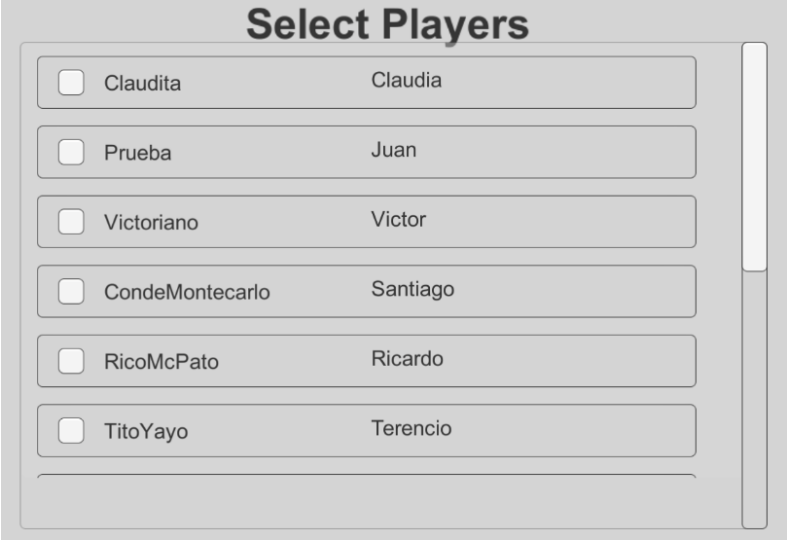


Imagen A.8: Listado de jugadores sin un grupo

Imagen A.9: Vista de usuarios que forman parte del grupo

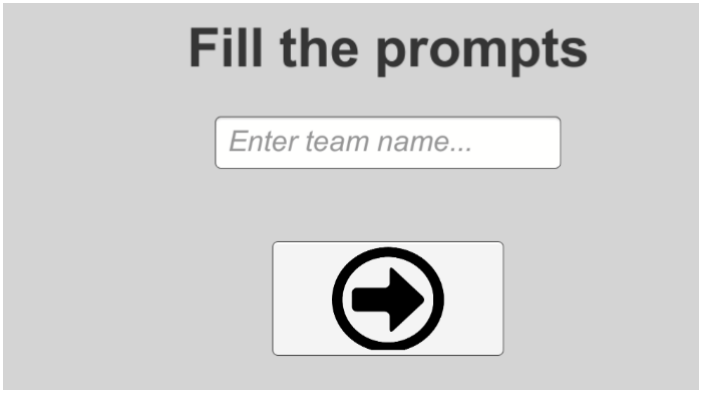


Imagen A.10: Crear un nuevo grupo

A.5. Menú de selección de minijuegos

El menú de selección de minijuegos es una pantalla inicial que sirve como una transición para que el usuario seleccione el juego que desea jugar. En esta pantalla, cada juego está representado por un botón con un título que lo identifica, como se indica en la imagen A.11. Cada botón está vinculado a su respectivo juego, por lo que, al hacer clic en el botón, el usuario será llevado directamente al juego seleccionado.

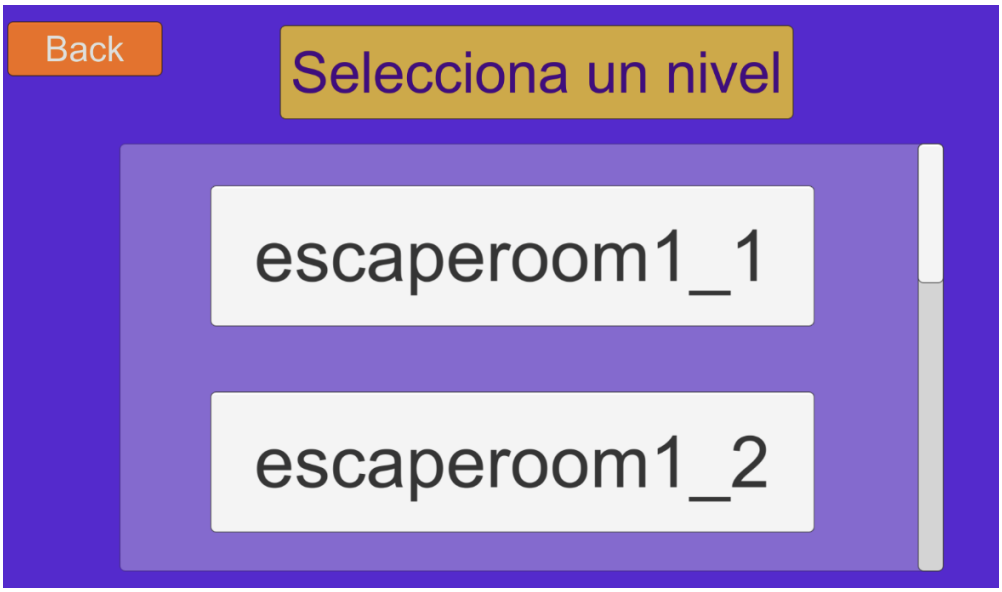


Imagen A.11: Menú de selección de minijuegos

A.5. Menú de edición de perfil

El menú de perfil proporciona al usuario información relevante sobre su cuenta y sus actividades dentro de la aplicación. El usuario puede acceder a esta sección haciendo clic en el botón en la parte superior derecha del menú principal (imagen A.12).

En el menú de perfil, el usuario puede ver sus calificaciones en los minijuegos en los que ha participado, lo que le permite evaluar su progreso y rendimiento en la aplicación (imagen A.14). Además, el usuario puede editar sus datos personales, incluyendo su nombre, dirección de correo electrónico y nombre y apellidos (imagen A.13).

También se ofrece la opción de darse de baja de la aplicación. Si el usuario decide hacerlo, se

eliminarán todos sus datos de las bases de datos del servidor, lo que incluye sus calificaciones y datos personales (imagen A.15).

El menú de perfil, visto en la imagen A.12, permite al usuario ver sus calificaciones en los minijuegos, como se aprecia en la imagen A.14, o poder editar sus datos personales, como se ve en la imagen A.13, donde también puede elegir darse de baja de la aplicación, lo que consistirá en la eliminación de todos sus datos de las bases de datos del servidor.

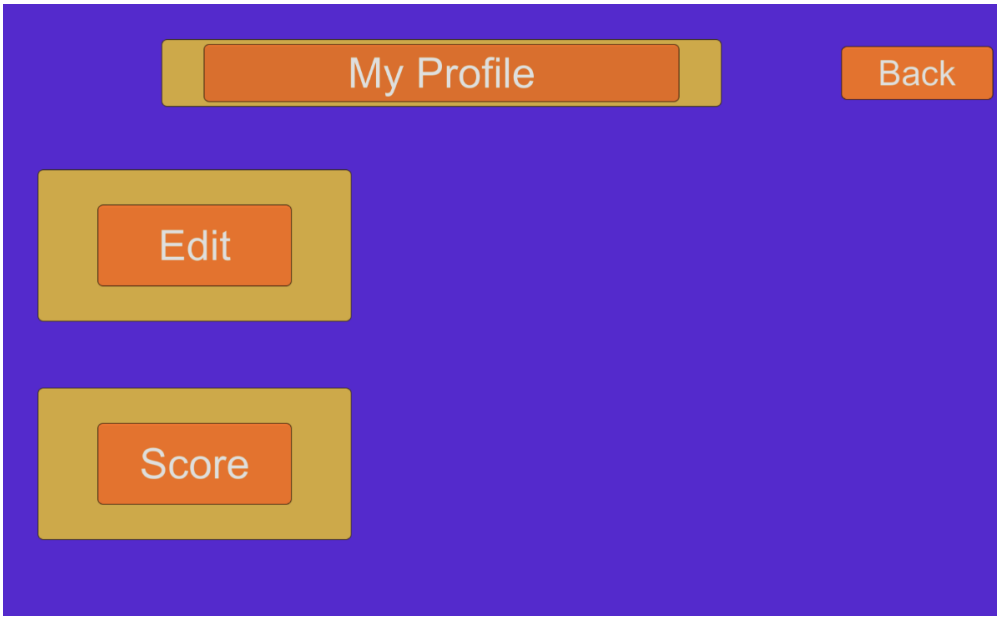


Imagen A.12: Vista del menú de perfil

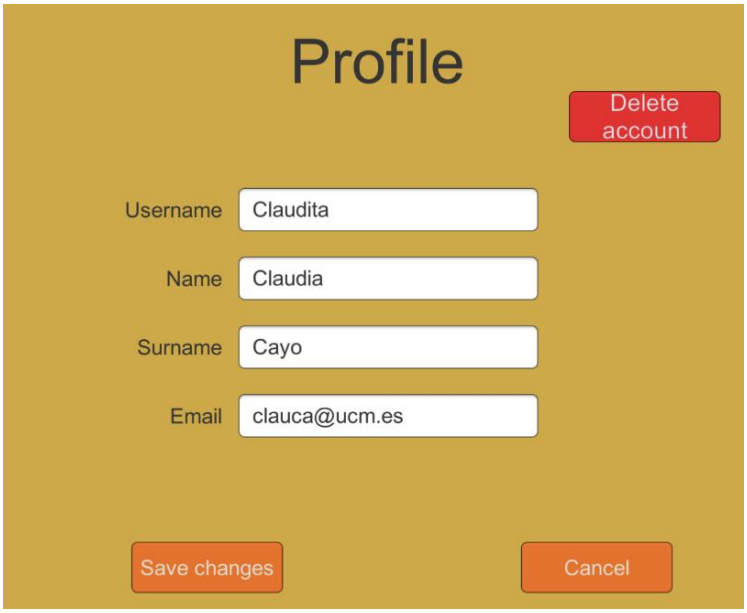


Imagen A.13: Listados

Imagen A.14: Vista de notas del usuario

A.6. Menú de pausa de juego

El menú de pausa es una funcionalidad presente en todos los minijuegos del juego. Se encuentra en la parte superior derecha de la pantalla y está representado por un símbolo específico (imagen A.15). Al presionar este símbolo, se despliega un menú simple (imagen A.16) que ofrece al usuario la opción de ajustar el volumen del juego, así como de continuar o abandonar la partida en curso. Es importante destacar que, si el usuario decide abandonar la partida, no se calificará la puntuación

acumulada hasta ese momento.

Imagen A.15: Ícono del menú de pausa

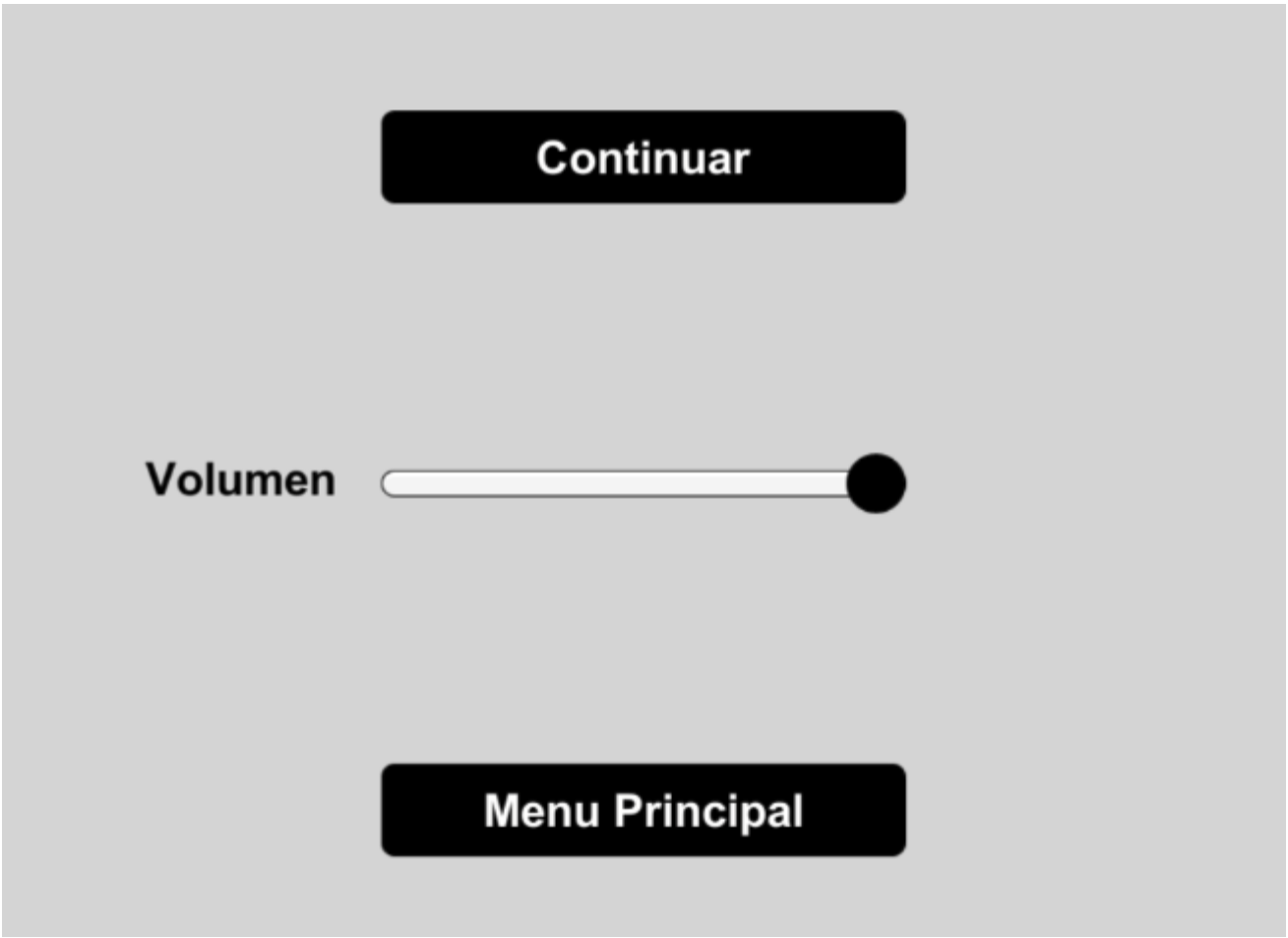


Imagen A.16: Menú de pausa

A.7. Minijuego 1 Bloque 1

Este minijuego, como se muestra en la imagen A.17, consiste en lanzar una bola hacia una de

las cinco cajas, cada una representando una condición médica. Cada bola viene acompañada de una imagen que se puede ver al presionar el botón de inspección (imagen A.18). El objetivo del juego es llenar las cinco cajas, lo que hará que las plataformas en el fondo vayan bajando. En sistemas operativos Windows se utiliza el ratón, y para dispositivos móviles con sistema operativo Android, se usa el dedo para lanzar las bolas. El juego termina cuando todas las plataformas tengan la misma altura.

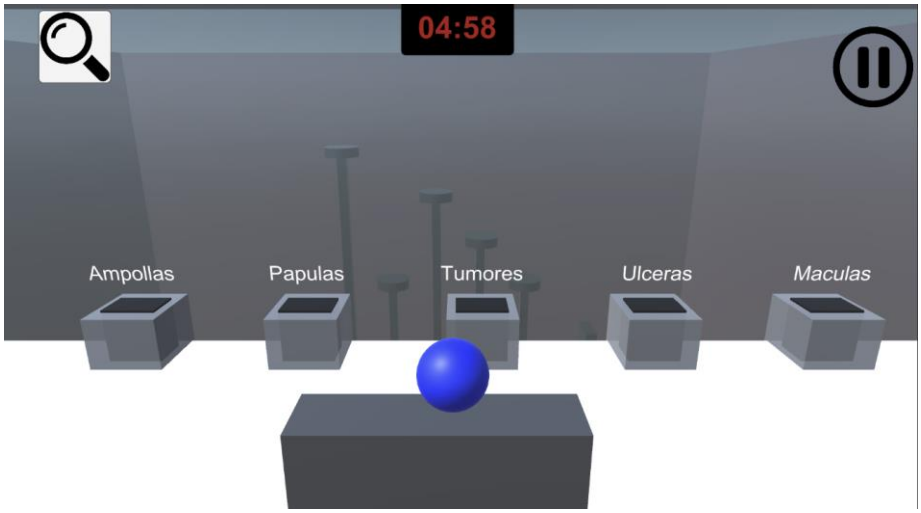


Imagen A.17: Minijuego 1 Bloque 1

Imagen A.18: Botón de inspección

A.8. Minijuego 2 Bloque 1

Este minijuego es una actividad interactiva que requiere del jugador que arrastre una imagen central que representa una patología bucal y la una con su respectiva causa, la cual se encuentra dispersa en diferentes puntos de la pantalla (imagen A.19). La mecánica es muy sencilla, el jugador debe arrastrar el bloque del jugador hasta el bloque que representa su causa correspondiente para que ambos se unan. El juego se considera terminado cuando se han unido todas las parejas de bloques.

En Windows, se utiliza el ratón para arrastrar el bloque del jugador y en Android se utiliza el dedo para el mismo propósito.

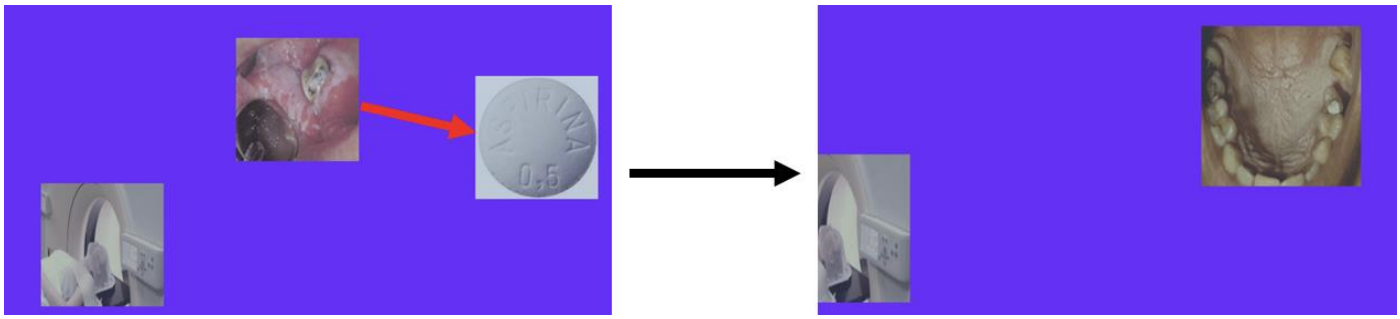


Imagen A.19: Minijuego 2 Bloque 1: el jugador mueve la imagen hacia la respuesta correcta, pasando a tener que encontrar el par de otra patología

A.9. Minijuego 3 Bloque 1

El minijuego presentado en la imagen A.20 es una trivia sobre odontología. El objetivo del juego es responder correctamente una serie de preguntas, lo que permitirá avanzar al siguiente nivel. Sin embargo, si se responde incorrectamente, la puntuación disminuirá y el jugador deberá intentarlo de nuevo. En la imagen A.21 se muestran las respuestas a las preguntas planteadas en el juego

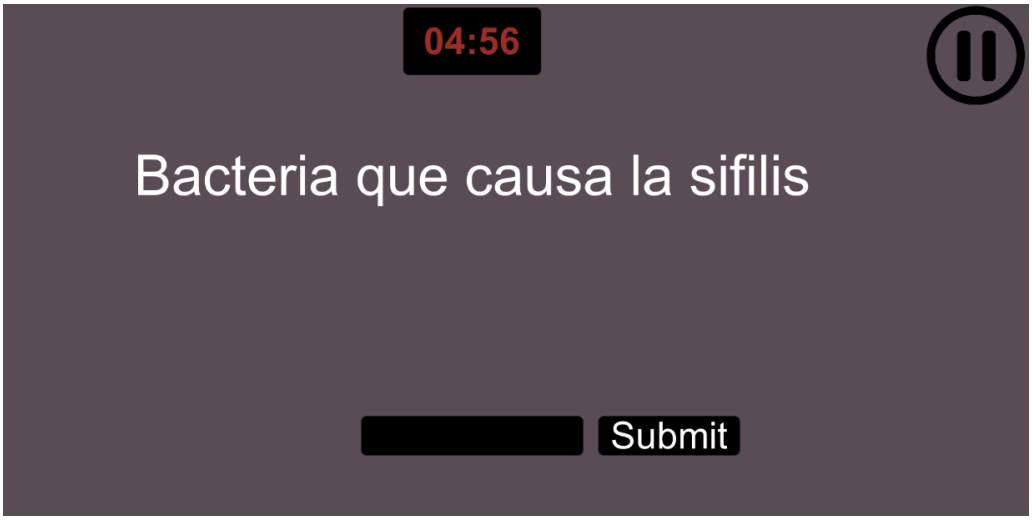


Imagen A.20: Minijuego 3 Bloque 1

Bacteria que causa la sífilis	Treponema Pallidum
Lesión elemental que aparece en la gingivitis necrosante en las papilas	Úlcera
Tipo de estreptococo causante de la escarlatina	B-Hemolítico
Lesión cutánea que aparece en la escarlatina	Erupción
Lesión comisural que aparece en la sífilis terciaria	Rágades
Las úlceras producidas por la tuberculosis son de evolución...	Crónica
Previamente a la gingivitis necrosante se le denominaba en lugar de necrosante...	Ulceronecrotizante
Las lesiones similares a un fresón o fresa	Lengua

de las sífilis, ¿aparecen en qué localización de la cavidad oral?	
Las lesiones orales de la sífilis pueden deberse a la transmisión sexual debida a sexo...	Oral
La gingivitis necrosante se asocia a tabaco, mala higiene y...	Estrés
La infección bacteriana producida por el *S. Aureus* que cursa con erupciones pustulosas agudas de la piel peribucal es el...	Impétigo
La osteomielitis está producida por el...	*Staphilococcus Aureus*

Imagen A.21: Respuestas

A.10. Minijuego 4 Bloque 1

Este minijuego consta de cinco preguntas de opción múltiple que deben ser respondidas correctamente para desbloquear la puerta (imagen A.22). El jugador debe elegir la respuesta correcta de las opciones presentadas y, en caso de acertar, avanzará hacia la siguiente pregunta. Si el jugador falla, se le restarán puntos, pero aún puede terminar el juego incluso si todas las respuestas son incorrectas. El objetivo final es desbloquear la puerta para avanzar al siguiente minijuego.

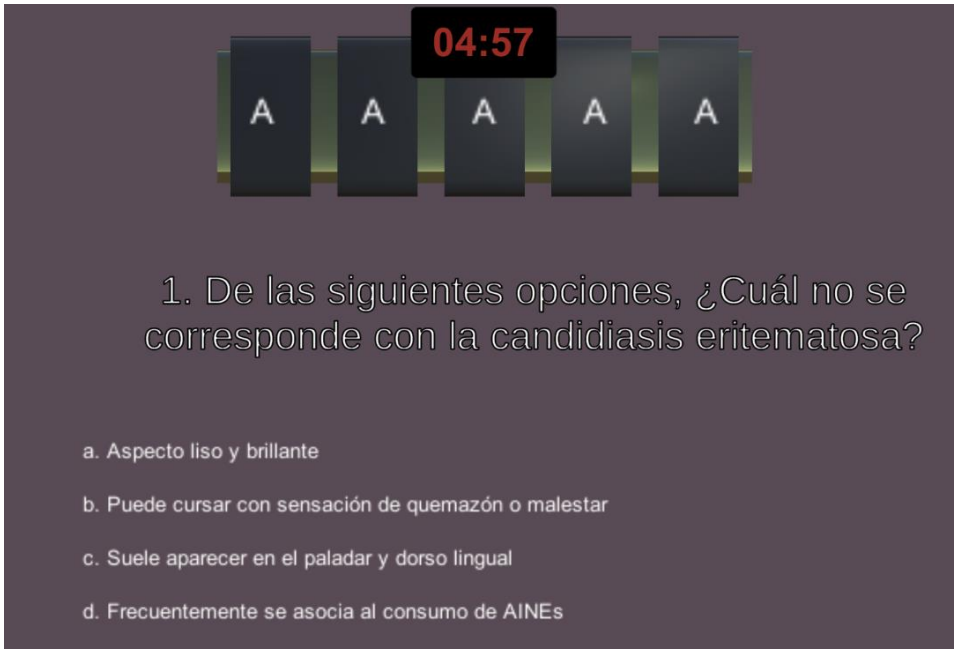


Imagen A.22: Minijuego 4 Bloque 1

A.11. Minijuego 5 Bloque 1

Este minijuego es una dinámica de preguntas con tres posibles respuestas (imagen A.23). El jugador debe elegir una de las tres puertas disponibles, cada una de las cuales representa una posible respuesta. Cada puerta tiene un valor textual en la parte superior que indica la respuesta correspondiente.

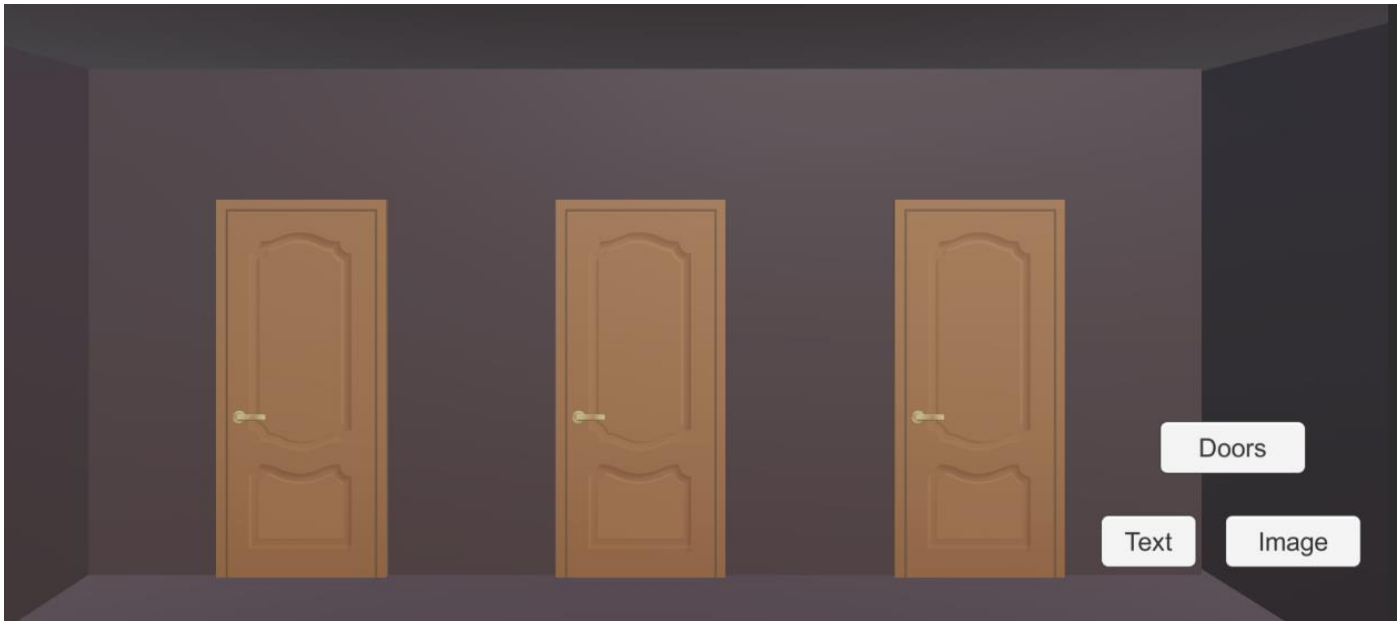


Imagen A.23: Minijuego 5 Bloque 1

Para poder responder a la pregunta, el jugador debe leer la información disponible mediante el uso de los botones de la interfaz la parte inferior derecha de la pantalla, la cual permite ver una imagen relacionada con la pregunta y un texto que plantea la pregunta (imagen A.24). Una vez que se ha seleccionado una de las tres puertas, el jugador puede continuar respondiendo las preguntas hasta el final del juego.

Imagen A.24: Botones del minijuego y sus respectivas vistas

A.12. Minijuego 6 Bloque 1

Este minijuego mostrado en la imagen A.25, es un juego de ordenamiento de letras. El jugador debe arrastrar las letras que se han revuelto por accidente a sus posiciones correctas para formar dos palabras. Una vez que el jugador haya ordenado correctamente las letras y haya formado las dos palabras (imagen A.26), se terminará el juego.

Imagen A.25: Minijuego 6 Bloque 1

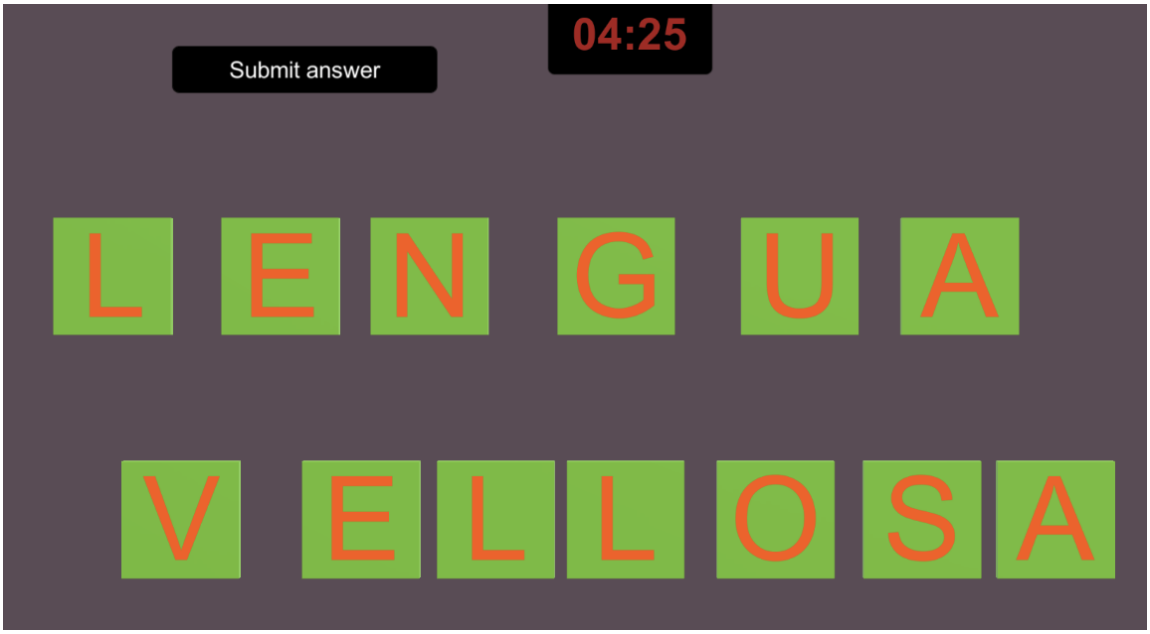


Imagen A.26: Fin del juego

A.13. Minijuego 7 Bloque 1

Este minijuego es un Pasapalabra, que consiste en una serie de preguntas que el jugador debe responder de forma correcta o incorrecta, o bien, pasar a la siguiente. El objetivo es completar todas las preguntas antes de que se den dos vueltas completas a la ruleta (imagen A.27).

El jugador puede avanzar a través de las preguntas que comienzan o contienen una letra de la ruleta y responder a la pregunta correspondiente. Si la respuesta es correcta, se ganará un punto, de lo contrario, se restará un punto. El juego finalizará una vez que todas las preguntas hayan sido respondidas o si la ruleta da dos vueltas completas.

Imagen A.27: Minijuego 7 Bloque 1

A.14. Recorrido completo de la aplicación

Como ejemplo de un recorrido completo de la aplicación, se tiene:

A. **Inicio de sesión:** el usuario inicia sesión con sus credenciales.

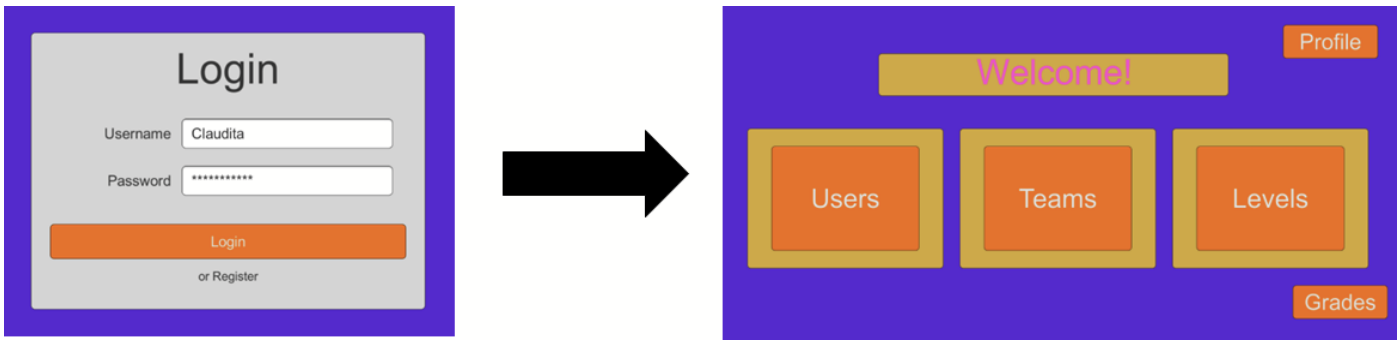


Imagen A.28: Paso A

B. **Navegación al menú de usuarios:** el usuario navega del menú principal al menú de usuarios pulsando el botón correspondiente, y luego examina uno de los usuarios para

eliminarlo, pues se trata de un profesor.

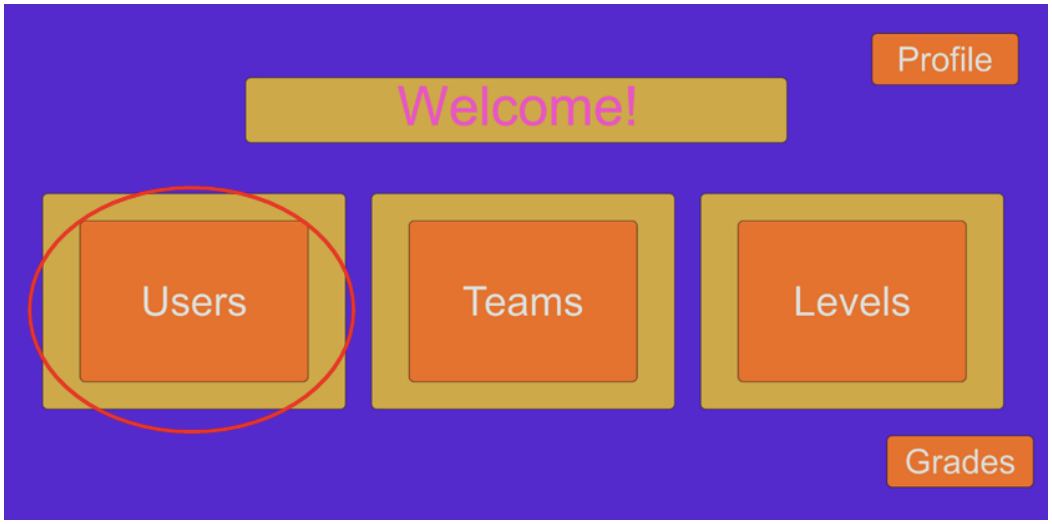


Imagen A.28: Paso B1

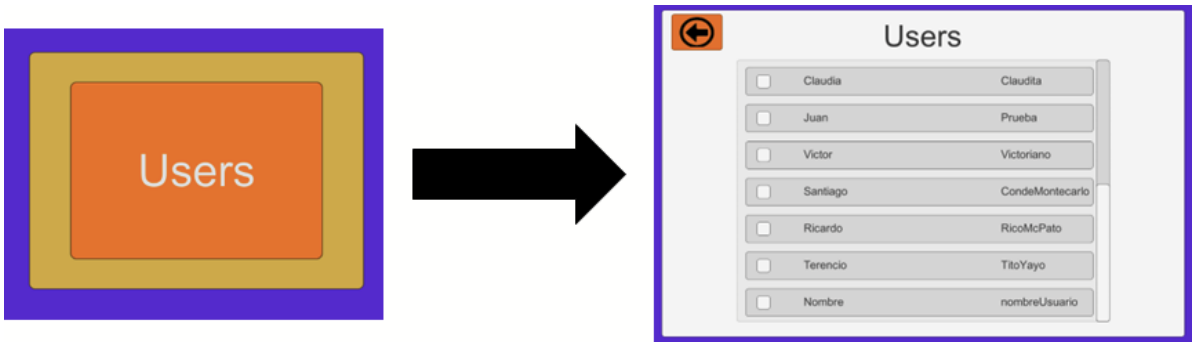


Imagen A.28: Paso B2

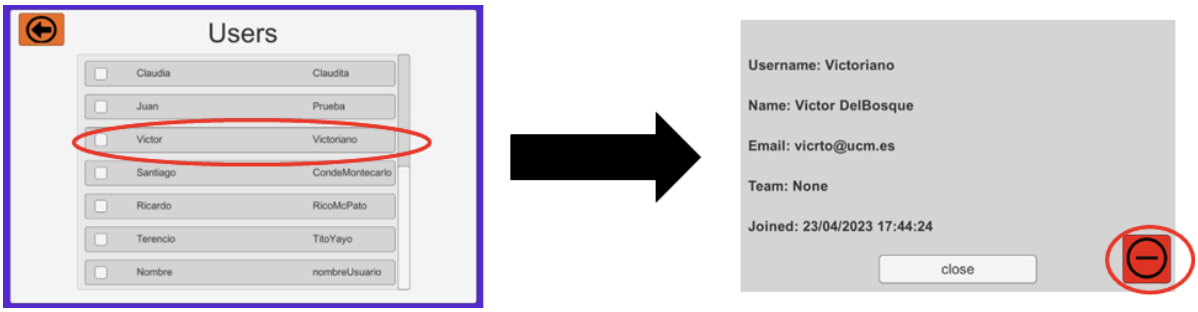


Imagen A.28: Paso B3

C. Navegación al menú de perfil: el usuario navega al menú de perfil, que se encuentra en la parte superior derecha del menú principal.

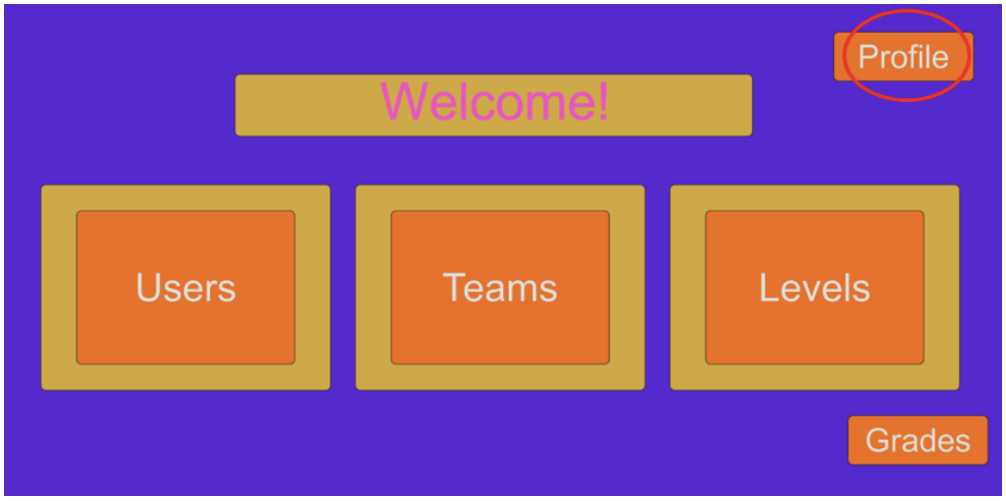


Imagen A.28: Paso C1

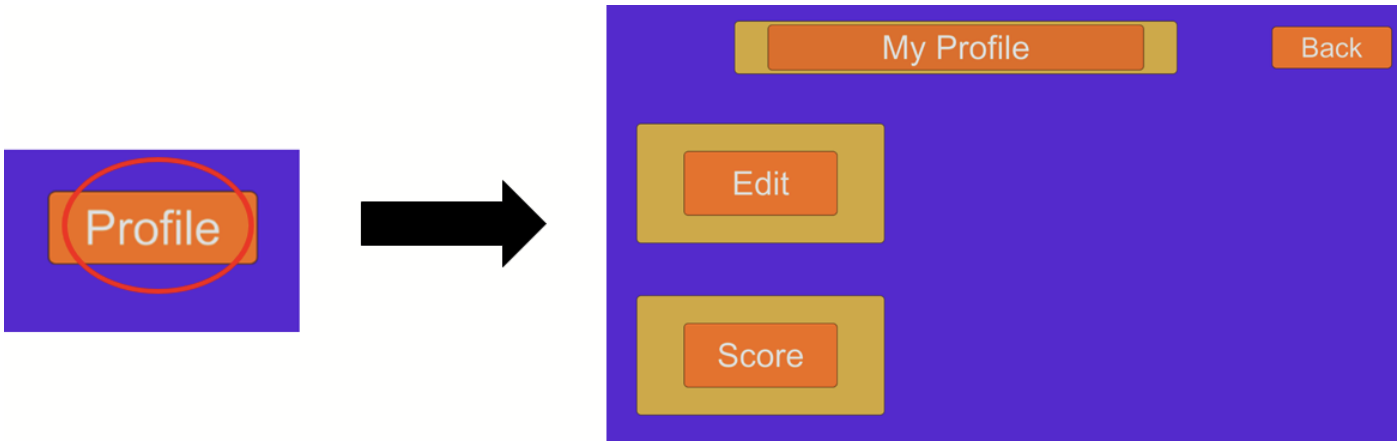


Imagen A.28: Paso C2

D. Vista de notas de usuario: el usuario pulsa el botón score del menú de perfil para ver sus notas en los minijuegos

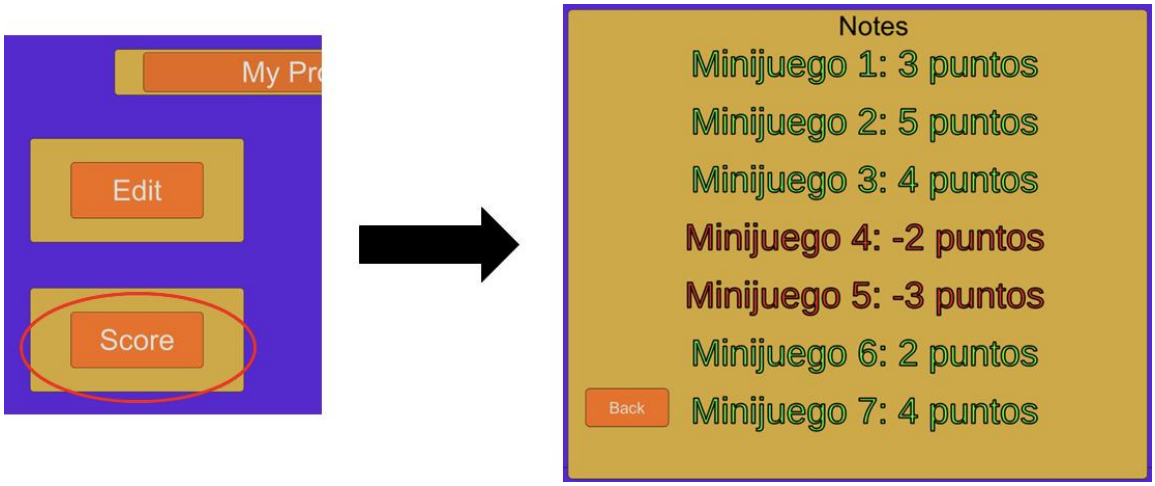


Imagen A.28: Paso D

E. Editar perfil y/o eliminar cuenta: el usuario pulsa el botón de editar perfil, donde puede cambiar sus datos personales o eliminar la cuenta totalmente.

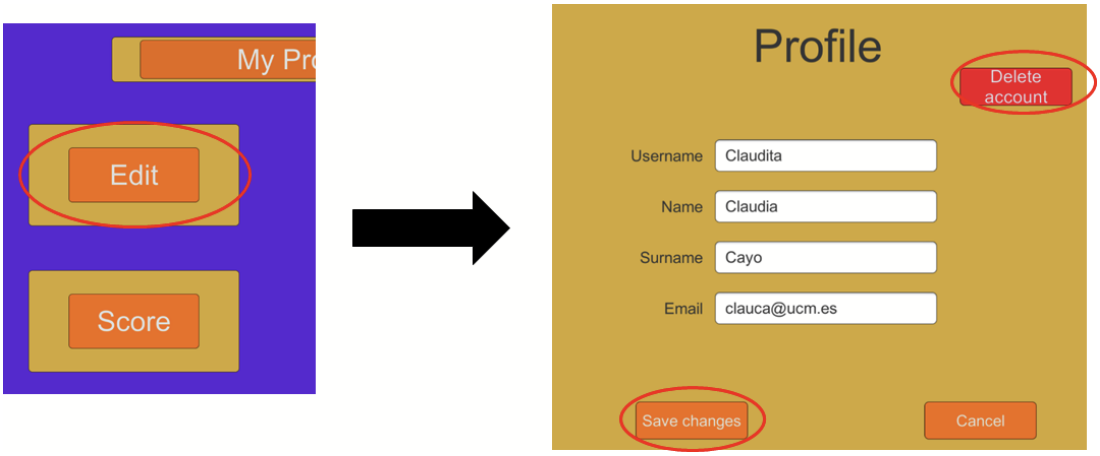


Imagen A.28: Paso E

F. Exportar notas: el usuario profesor pulsa el botón de exportar notas en el menú principal, descargando un archivo PDF con las notas de los alumnos.

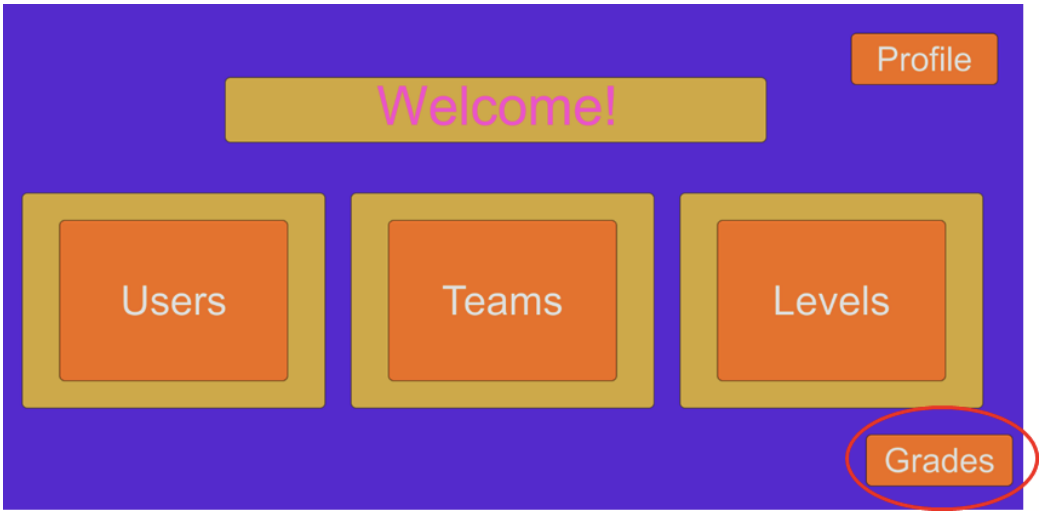


Imagen A.28: Paso F1

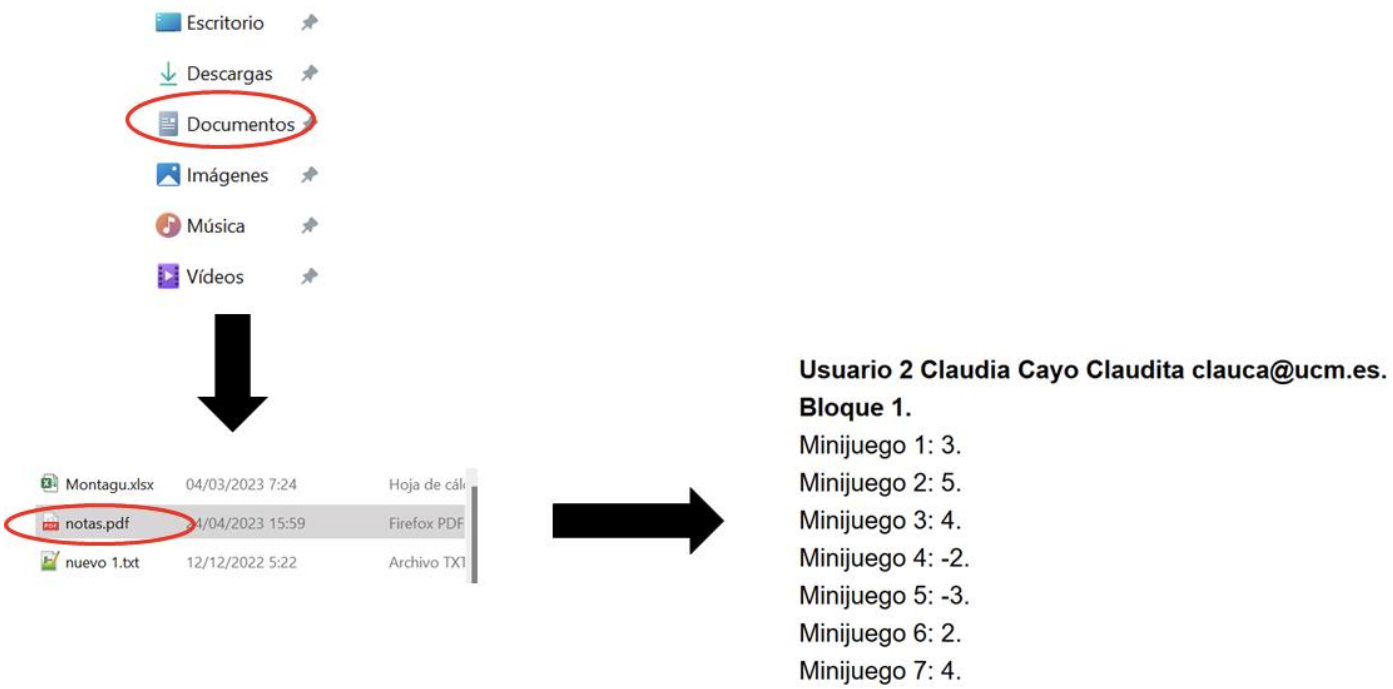


Imagen A.28: Paso F2

G. Selección de minijuego: el usuario pulsa el botón de elegir niveles en el menú principal y llega al menú de niveles, donde puede escoger entre los varios niveles disponibles.

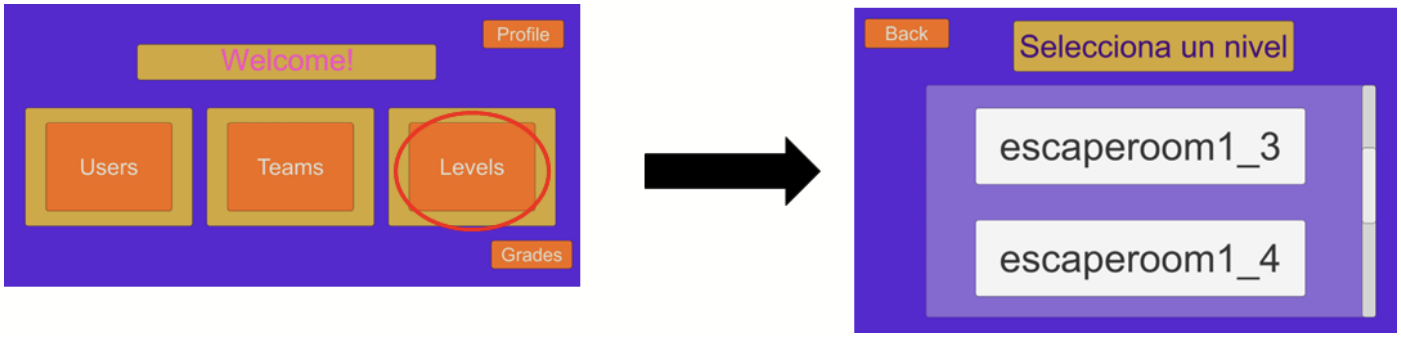


Imagen A.28: Paso G1

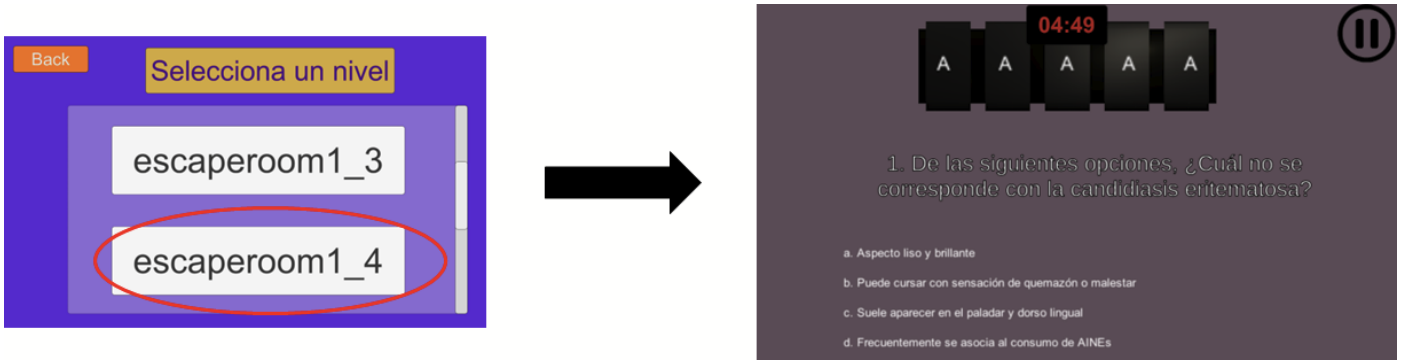


Imagen A.28: Paso G2

H. Abrir el menú de pausa: el usuario pulsa el botón de pausa, deteniendo el contador y abriendo el menú de pausa.

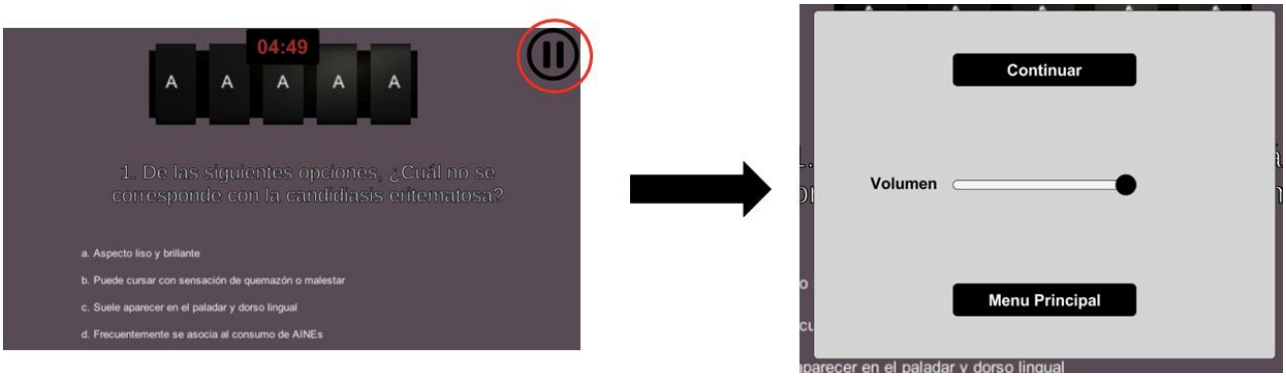


Imagen A.28: Paso H

I. Escena de transición: el usuario termina un minijuego, por lo que pasa a la escena de transición donde puede ver su rendimiento antes de continuar.

Imagen A.28: Paso I