DESARROLLO DE UNA APLICACIÓN MÓVIL DE AUTOEVALUACIÓN PARA ESTUDIANTES MIR Y SU ADMINISTRACIÓN WEB

TRABAJO FIN DE GRADO
CURSO 2023-2024

AUTOR
ENRIQUE MARTÍNEZ SÁNCHEZ

DIRECTOR
ANTONIO SARASA CABEZUELO

GRADO EN INGENIERÍA DEL SOFTWARE
FACULTAD DE INFORMÁTICA
UNIVERSIDAD COMPLUTENSE DE MADRID

DESARROLLO DE UNA APLICACIÓN MÓVIL DE AUTOEVALUACIÓN PARA ESTUDIANTES MIR Y SU ADMINISTRACIÓN WEB

DEVELOPMENT OF A SELF-ASSESSMENT MOBILE APPLICATION FOR MIR STUDENTS AND ITS WEB ADMINISTRATION

TRABAJO DE FIN DE GRADO EN INGENIERÍA DEL SOFTWARE

AUTOR
ENRIQUE MARTÍNEZ SÁNCHEZ

DIRECTOR
ANTONIO SARASA CABEZUELO

CONVOCATORIA: JUNIO 2024

GRADO EN INGENIERÍA DEL SOFTWARE
FACULTAD DE INFORMÁTICA
UNIVERSIDAD COMPLUTENSE DE MADRID

14 DE MAYO DE 2024

DEDICATORIA

A mis padres y todos mis seres queridos
que me han apoyado en el camino.

AGRADECIMIENTOS

En primer lugar, a mis profesores del grado, que me han permitido desarrollar las habilidades necesarias para poder realizar este trabajo.

A mis compañeros del grado, que me han acompañado en este proceso académico y que me han ayudado siempre que lo he necesitado.

A mi tutor, Antonio Sarasa Cabezuelo, por permitirme desarrollar este TFG y estar siempre disponible ante cualquier duda.

RESUMEN

Desarrollo de una aplicación móvil de autoevaluación para estudiantes MIR y su administración web

Este trabajo tiene como objetivo dar solución a la necesidad presente en los distintos centros de formación de estudiantes de oposición MIR. En estos centros, el profesorado necesita crear baterías de preguntas, basándose en el modelo de examen de oposición y hacerlas llegar al alumnado de una manera simple y que fomente su aprendizaje y autoevaluación.

La aplicación consta de dos partes distintas, una web y otra móvil. La primera, enfocada a la administración por parte de los profesores o médicos, donde se pueden crear distintos temas y preguntas que formarán parte del juego teórico, o también crear y configurar distintos juegos de competición con los parámetros deseados para simular una evaluación específica, además de ver sus resultados. La segunda, brinda la posibilidad a los estudiantes de autoevaluarse mediante la personalización del juego teórico para practicar, o intentar superar los juegos de competición, así como ver sus resultados.

Palabras clave

Aplicación Android, Autoevaluación, Test, MIR, Juego, Medicina, Estudiar, Academia, Practicar, Examen.

ABSTRACT

Development of a Self-Assessment Mobile Application for MIR Students and Its Web Administration

This work aims to address the needs present in various training centers for MIR opposition exam students. In these centers, the teaching staff needs to create question sets based on the opposition exam model and deliver them to the students in a simple way that promotes their learning and self-assessment.

The application consists of two distinct parts, a web and a mobile one. The first, focused on administration by teachers or doctors, where they can create different topics and questions that will be part of the theoretical game, or also create and configure different competition games with desired parameters to simulate a specific evaluation, in addition to viewing their results. The second offers students the chance to self-assess through the customization of the practical game, or try to surpass the competition games, as well as see their results.

Keywords

Android Application, Self-assessment, Test, MIR, Game, Medicine, Study, Academy, Practice, Exam.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

Capítulo 1 - Introducción

En este primer capítulo se presenta, en primer lugar, las distintas razones que justifican la realización de este trabajo, así como los objetivos que se pretenden alcanzar en respuesta a la necesidad a cubrir. Finalmente, se desglosa el plan de trabajo seguido durante la realización del proyecto.

1.1 Motivación

La formación correcta de los estudiantes siempre ha sido motivo de preocupación por los profesionales de la enseñanza, en concreto, por los profesores de estudiantes del ámbito de la medicina. En estos estudiantes, recae una responsabilidad de orden superior, ya que en sus manos se deposita la salud de todas las personas de la sociedad. La manera actual establecida para la evaluación de estos estudiantes, antes de poder ejercer como profesionales médicos, es mediante oposición nacional, la llamada oposición MIR (Médico Interno Residente), a la que se presentaron en 2024 en España un total de 13.990 aspirantes, incrementándose un 10,4% con respecto a las cifras del 2023 [1] .

La constante y creciente necesidad anual de formar a los estudiantes de la oposición, ya sea una vez hayan superado el grado de medicina o durante el mismo, supone un gran reto para las academias y organizaciones encargadas de gestionar esta preparación en un gran número de alumnos. Los estudiantes pasan entre 8 y 10 horas diarias estudiando el temario de lunes a sábado [2] durante el año previo al examen, y tras el contenido teórico, necesitan evaluar los conocimientos adquiridos de manera práctica. Por ello, la creación de aplicaciones que centralicen la creación de preguntas de evaluación por parte de distintos profesores al mismo tiempo, y que las hagan llegar a los alumnos de una manera directa y sencilla, son cruciales para el correcto funcionamiento de los centros de preparación de este tipo de oposición.

Por parte de los estudiantes, es necesario cubrir esta autoevaluación diaria para seguir el progreso de aprendizaje, pero, además, es necesario buscar motivación para realizarla. Para ello, la introducción de la gamificación resulta un beneficio añadido a

este proceso, para los estudiantes, motivando la realización de evaluaciones y para las academias la obtención de resultados globales positivos.

Para cubrir todas estas necesidades, se han desarrollado dos aplicaciones: una web, destinada a distintos administradores, profesores o médicos, que permite la creación de baterías de preguntas del temario y la configuración de distintos juegos, y una móvil Android que posibilita a los estudiantes evaluarse y competir tras el estudio.

1.2 Objetivos

El objetivo principal del proyecto es desarrollar un sistema que mejore la autoevaluación de los estudiantes del MIR, así como la preparación de los contenidos a evaluar por parte de los administradores del sistema. Esto se consigue a través de los siguientes objetivos:

- Implementar funcionalidades para la gestión de la cuenta de usuario, como el registro, inicio de sesión, cierre de sesión, edición y eliminación del perfil.
- Desarrollar un sistema para la administración simultanea por distintos profesores de los contenidos a evaluar, que permita crear un juego teórico con temas y sus tipos de preguntas, además de poder modificarlos, eliminarlos, publicarlos o retirarlos. Igualmente, el sistema debe permitir crear y configurar juegos de competición.
- Visualizar los resultados de las competiciones activas, para que los profesores puedan observar qué grupos de estudiantes están más adelantados que otros y poder adaptar su enseñanza. Igualmente, por parte de los usuarios estudiantes, visualizar la evolución de sus intentos teóricos y los resultados de los juegos de competición, para que puedan llevar un seguimiento de la evolución de su aprendizaje.
- Permitir la autoevaluación de los estudiantes mediante el juego teórico, pudiéndose configurar según la necesidad del usuario. Permitir a los alumnos igualmente acceder a los juegos de competición, preestablecidos por los

administradores para fomentar una competición entre distintos grupos de estudiantes de distintas universidades.

1.3 Plan de trabajo

La estrategia de planificación se ha llevado a cabo mediante distintos hitos. Se realizó una reunión inicial con el tutor para establecer los objetivos de la aplicación y a partir de ahí se elaboró la especificación de requisitos.

Una vez que establecidos los requisitos, se clasificaron en diferentes módulos, y de ellos surgieron los distintos hitos que formarían cada iteración del trabajo. La organización temporal inicial de las iteraciones se muestra en el diagrama de Gantt de la Figura 1 y de la Tabla 1.

Figura 1 Diagrama de Gantt

Etapa	Duración
Especificación de Requisitos	18 Septiembre - 16 Octubre
Gestión de usuarios	17 Octubre - 19 Diciembre
Juego teórico	20 Diciembre - 12 Febrero
Bloques temáticos	13 Febrero - 26 Febrero
Juegos de competición	27 Febrero - 11 Marzo
Gestión de resultados	12 Marzo - 25 Marzo
Jugar	26 Marzo - 29 Abril
Elaboración de la memoria	30 Abril - 14 Mayo

Tabla 1 Etapas temporales del desarrollo de los Hitos

A continuación, se detallarán las fases del proyecto:

Fase 1: Especificación de requisitos

En la primera etapa se idearon las distintas funcionalidades que podría tener la herramienta y se decidieron cuáles de ellas eran las destinadas a formar parte de los casos de uso finales. Estos se agruparon por funcionalidad en distintos módulos, quedando la distribución final de los hitos a realizar.

Fase 2: Gestión de usuarios:

En esta fase se comenzó creando y configurando la base de datos junto con funcionalidades de sesión del usuario en ambas aplicaciones, inicio y cierre. Eliminación del perfil en la parte móvil y la modificación de los datos del usuario en ambas.

Fase 3: Juego teórico:

En esta fase se desarrollaron las funcionalidades de la parte de la administración web de CRUD (*Create, Read, Update y Delete*) referentes al juego teórico, así como la funcionalidad de publicar o retirar cualquier juego visualizado.

Fase 4: Bloques temáticos:

Una vez completada la iteración anterior, en esta fase, continuando en la parte de administración web, se desarrolló la funcionalidad relacionada con el juego teórico único ya creado, como las funcionalidades CRUD de los temas que puede contener y de las preguntas asociadas.

Fase 5: Juegos de competición:

En esta nueva iteración, se completaron las funcionalidades de administración relativas a los juegos de competición, creado, eliminado, modificado y visualizado.

Fase 6 Gestión de resultados:

En esta sexta fase se desarrollaron las funcionalidades relacionadas con la visualización de los juegos de competición activos por parte de los administradores, y de la gráfica de resultados de cada uno de ellos.

Fase 7: Jugar:

En esta nueva iteración, tras completar la parte de la administración web en las anteriores, se han llevado a cabo las funcionalidades de la aplicación móvil de los usuarios estudiantes. Permitir visualizar los juegos disponibles y por iniciar, evaluarse practicando con el juego teórico, jugar al juego de competición y visualizar los resultados de todos juegos, así como las preguntas favoritas. Esta iteración es la última de desarrollo que completa el producto mínimo viable planteado en la idea inicial de la primera fase.

Fase 8: Elaboración de la memoria:

En esta última fase se llevó a cabo la creación y elaboración de la memoria del proyecto.

1.4 Estructura de la memoria

En este apartado se describe la estructura de este documento:

- **Capítulo 1 – Introducción:** motivación para la realización del trabajo, objetivos planteados junto con el plan de trabajo y su estructura.

- **Capítulo 2 - Estado del arte**: descripción de aplicaciones similares a la planteada y sus características principales.

- **Capítulo 3 - Tecnología empleada**: enumeración y explicación de las distintas tecnologías utilizadas durante la ejecución del proyecto.

- **Capítulo 4 - Arquitectura de la aplicación**: descripción de la estructura de las aplicaciones desarrolladas en base a su tecnología, y el modelo de datos sobre el que trabajan junto con una descripción de cada colección.

- **Capítulo 5 - Diseño**: explicación del diseño de la aplicación.

- **Capítulo 6 - Funcionalidades**: descripción de las funcionalidades del proyecto, con partes visuales de las aplicaciones y fragmentos del código fundamental para cada una.

- **Capítulo 7 - Conclusiones y trabajo futuro**: conclusiones del desarrollo de la aplicación y las posibles mejoras y nuevas funcionalidades que podrían ser implementados para completar las actuales u optimizarlas.

1-Introduction

In this first chapter, the various reasons justifying the undertaking of this work are presented first, along with the objectives intended to be achieved in response to the need to be addressed. Finally, the work plan followed during the project's execution is detailed.

1.1 Motivation

The proper training of students has always been a concern for educational professionals, specifically for teachers of students in the field of medicine. These students carry a higher order of responsibility, as the health of all people in society is entrusted to their hands. The current method established for evaluating these students, before they can practice as medical professionals, is through a national examination, known as the MIR (Médico Interno Residente) opposition, to which a total of 13,990 candidates presented themselves in 2024 in Spain, an increase of 10.4% compared to the figures from 2023 [1].

The constant and growing annual need to train opposition students, whether after completing their medical degree or during it, poses a significant challenge for academies and organizations responsible for managing this preparation for many students. Students spend between 8 and 10 hours a day studying the syllabus from Monday to Saturday [2] during the year before the exam, and after the theoretical content, they need to practically assess the knowledge they have acquired. Therefore, the creation of applications that centralize the creation of assessment questions by different teachers at the same time and deliver them to the students in a direct and simple manner, are crucial for the proper functioning of the preparation centers for this type of examination.

For the students, it is necessary to cover this daily self-assessment to follow their learning progress, but it is also necessary to find motivation to carry it out. To this end, the introduction of gamification results in an added benefit to this process, for the students,

motivating the performance of evaluations and for the academies, the achievement of positive overall results.

To meet all these needs, two applications have been developed: a web application, for various administrators, teachers, or doctors, where they can create question banks for the syllabus and configure different games, and an Android mobile application where students can evaluate themselves and compete after studying.

1.2 Goals

The main goal of the project is to develop a system that improves the self-assessment of MIR students, as well as the preparation of the content to be assessed by the system administrators. This is achieved through the following objectives:

- Implement functionalities for managing the user account, such as registration, login, logout, profile editing, and deletion.
- Develop a system for the simultaneous administration by different teachers of the content to be assessed, which allows the creation of a theoretical game with topics and their types of questions, as well as the ability to modify, delete, publish, or withdraw them. Likewise, the system should allow the creation and configuration of competition games.
- Visualize the results of active competitions, so that teachers can observe which groups of students are more advanced than others and can adapt their teaching accordingly. Similarly, for student users, to visualize the evolution of their theoretical attempts and the results of competition games, so that they can track the progress of their learning.
- Allow the self-assessment of students through the theoretical game, which can be configured according to the user's needs. Also, allow students to access the competition games set up by the administrators, to encourage competition among different groups of students from various universities.

1.3 Work Plan

The planning strategy was carried out through various phases. An initial meeting with the tutor was held to establish the objectives of the application, and from there, the specification of requirements was developed.

Once the requirements were established, they were classified into different modules, and from these, the various phases that would form each work iteration emerged. The initial temporal organization of the iterations is shown in the Gantt chart of Figura 1 and Tabla 1.

Next, the phases of the project will be detailed:

Phase 1: Specification of Requirements

In the first stage, the various functionalities that the tool could have been devised, and it was decided which of these were destined to be part of the final use cases. These were grouped by functionality into different modules, with the final distribution of the milestones to be achieved.

Phase 2: User Management

In this phase, the creation and configuration of the database began, along with user session functionalities in both applications, login and logout. Profile deletion in the mobile part and user data modification in both.

Phase 3: Theoretical Game

In this phase, the functionalities of the web administration part of CRUD (Create, Read, Update, and Delete) related to the theoretical game were developed, as well as the functionality to publish or withdraw any displayed game.

Phase 4: Thematic Blocks

Once the previous iteration was completed, this phase continued on the web administration side, developing the functionality related to the already created unique theoretical game, such as the CRUD functionalities of the topics it can contain and the associated questions.

Phase 5: Competition Games

In this new iteration, the administrative functionalities related to competition games were completed, created, deleted, modified, and displayed.

Phase 6: Results Management

In this sixth phase, the functionalities related to the visualization of active competition games by the administrators were developed, and the results view for each of them.

Phase 7: Play

In this new iteration, after completing the web administration part in the previous ones, the functionalities of the mobile application for student users have been carried out. Allow viewing available and upcoming games, evaluate themselves by practicing with the theoretical game, play the competition game, and view the results of all games, as well as favourite questions. This iteration is the last of development that completes the minimum viable product proposed in the initial idea of the first phase.

Phase 8: Report Elaboration

In this final phase, the creation and compilation of the project report were carried out.

1.4 Memory Structure

In this section, the structure of this document is described:

- **Chapter 1 – Introduction:** Motivation for undertaking the work, objectives set out along with the work plan and its structure.
- **Chapter 2 – State of the Art:** Description of applications similar to the one proposed and their main characteristics.
- **Chapter 3 – Technology Used:** Listing and explanation of the various technologies used during the execution of the project.

- **Chapter 4 – Application Architecture:** Description of the structure of the developed applications based on their technology, and the data model they work with along with a description of each collection.
- **Chapter 5 – Design:** Explanation of the application design.
- **Chapter 6 – Functionalities:** Description of the project's functionalities, with visual parts of the applications and snippets of the key code for each.
- **Chapter 7 – Conclusions and Future Work:** Conclusions from the development of the application and the possible improvements and new functionalities that could be implemented to complete the current ones or optimize them.

Capítulo 2 - Estado del arte

En este capítulo se presentan otras aplicaciones similares a la desarrollada en este trabajo. Las tres primeras, forman parte de las academias más importantes y solicitadas a nivel nacional [3], ordenadas de mayor antigüedad a menor. Las dos últimas, añaden al planteamiento anterior gamificación para fomentar el aprendizaje. Todas ellas necesitan del pago de subscripción para poder utilizar todo su contenido, excepto Mirial que es gratuita en su totalidad. El pago por el contenido de las aplicaciones es justificado en cada academia por la estimación de las preguntas con más probabilidades de entrar en el examen cada año, es decir, su modelo de negocio está basado en el correcto funcionamiento de sus estadísticas para la correcta recomendación de preguntas para que practiquen los alumnos cada año.

2.1 MIR Asturias

La academia MIR Asturias [4] fundada en 1988, cuenta con una gran experiencia en la creación de baterías de preguntas de examen para que los alumnos se evalúen y practiquen para la oposición. Esto hace, junto con su plataforma web y móvil de autoevaluación, que sea la academia MIR que mejores resultados obtiene y sea una de las más solicitadas. La plataforma web brinda a sus alumnos numerosas funcionalidades:

- Realizar test personalizados por tipo de preguntas, temario, preguntas más repetidas en exámenes, favoritas, preguntas falladas etc.
- Visualización de los resultados en gráficas de cada intento de la evaluación realizada junto a su configuración, incluso visualización de resultados netos por tipo de pregunta.
- Simulacros de exámenes con orientación de resultado en base al total de alumnos que los realizan.
- Contenido teórico visual con videos de la propia academia.

La aplicación móvil, por el contrario, no brinda toda la funcionalidad anterior, si no que se simplifica a la práctica de exámenes configurables o de simulacro.

2.2 AMIR app

La academia AMIR [5] nacida en 2007, prepara a sus estudiantes para la oposición mediante su plataforma web y móvil de práctica, tras dar el contenido teórico mediante clases online. En este caso ambas aplicaciones ofrecen el mismo contenido a sus usuarios:

- Ver los resultados de los simulacros de exámenes realizados, junto con el percentil respecto al resto de opositores
- Realizar simulacros simplificados o completos según las pautas del tutor.
- Realizar autoevaluaciones configuradas por asignaturas, número de preguntas, preguntas reales MIR, preguntas falladas etc.
- Seguimiento de la evolución individual, con datos desglosados por las asignaturas y gráficas comparativas con el resto de los estudiantes.
- Contenido teórico en video y encuestas a los tutores para mantener la calidad de la enseñanza.
- Seguimiento individualizado de los resultados por los tutores.

2.3 PostMIR app

La academia ProMIR [6], responsable de la aplicación PostMIR, que sin contar con mucha experiencia como las anteriores, es una de las más solicitadas por los nuevos estudiantes de oposición debido al uso de Inteligencia Artificial aplicada a la personalización de los test según el usuario va avanzando en el temario. Algunas de sus funcionalidades más destacadas tanto en web como móvil son:

- Uso de la aplicación divido en fases, calentamiento, construcción, consolidación, competición y calibración. Estas adaptan la funcionalidad de la app dependiendo de la fase en la que te encuentres, aumentando su complejidad progresivamente para facilitar su uso y experiencia al usuario.

- En fases iniciales, cuenta con test adaptados al nivel actual del usuario además de test de práctica con los temas estudiados el mismo día, simulacros de exámenes completos etc.
- Análisis continuo del proceso de aprendizaje para alcanzar el máximo rendimiento en un tiempo mínimo, adaptando los test.
- Contenido teórico en video, así como resultados en gráficos de las netas alcanzadas de aciertos entre otros.

2.4 Mirial

La aplicación Mirial [7], igualmente dedicada a la formación de estudiantes de medicina, pero con gamificación basada en pequeños objetivos y recompensas. Su planteamiento del estudio como un juego es su principal característica. Sus principales funcionalidades disponibles en web y móvil son las siguientes:

- Competir con otros usuarios en directo respondiendo preguntas aleatorias de la oposición, midiendo la agilidad de cada usuario para poder avanzar en el tablero del juego.
- Cuenta con un canal de chat para poder discutir con los usuarios amigos el resultado de las preguntas o dudas del temario.
- Visualizar los resultados, rankings de usuarios y preguntas falladas.
- Brinda premios a los usuarios primeros en los rankings.

2.5 PrepMIR

PrepMIR [8] es una página web de preparación del examen MIR cuya principal característica diferenciadora es que cuenta con un algoritmo patentado basado en el repaso espaciado. Algunas de sus principales funcionalidades son:

- Tiene un sistema de notificaciones basado en inteligencia artificial para no olvidar la práctica de exámenes.

- Elaboración de test con configuración por tema, preguntas más falladas por otros usuarios, tipo de pregunta etc.
- Clasificación de usuarios nacional por universidad y especialidad.
- Elaborar evaluaciones a modo reto para otros usuarios y avanzar en la clasificación realizando retos.
- Visualización de gráficas de resultados.

Capítulo 3 - Tecnología empleada

3.1 Visual Studio Code

Visual Studio Code [9], es uno de los editores de código fuente desarrollado por Microsoft más utilizados. Ofrece numerosas funcionalidades en forma de extensiones, por lo que es muy versátil en cuanto a posibilidades de desarrollo de distintos lenguajes y tecnologías, además de ofrecer funciones de depuración y resaltado del código. Esta gran adaptabilidad, es el motivo fundamental de su elección para ejecutar el resto de las tecnologías de la parte web que se mencionan a continuación.

3.2 NodeJs

NodeJs [10] es un entorno de ejecución de JavaScript en servidor gratuito, construido sobre el motor V8 de Chrome, de código abierto y multiplataforma que da la posibilidad a los desarrolladores de crear servidores, aplicaciones web, herramientas de línea de comandos y scripts. Utiliza un modelo de operaciones de entrada/salida sin bloqueo y orientado a eventos, lo que lo hace eficiente y apto para aplicaciones en tiempo real, permitiendo la ejecución de *endpoints* de manera asíncrona.

3.3 Express

Express [11] es un marco de trabajo para aplicaciones web de Node.js. Ofrece diversas soluciones para programar páginas web mediante el uso de plantillas, como EJS. Express permite configurar rutas con su manejo de peticiones y respuestas HTTP, lo cual facilita la creación de aplicaciones de manera ordenada y escalable.

3.4 JavaScript

JavaScript [12] es un lenguaje de programación interpretado, no necesita ser compilado para su ejecución. Puede ser implementado para el lado del cliente,

añadiendo interactividad y funcionalidad a las páginas web, o del lado del servidor como se mencionó en el punto 3.2. en NodeJs.

Además, se han utilizado las siguientes librerías de este lenguaje:

3.4.1 jQuery

jQuery [13] es una biblioteca de JavaScript que simplifica el manejo del DOM (*Document Object Model*), manipulando los documentos HTML (*HyperText Markup Language*), manejando funcionalidad relacionada con la escucha de eventos, animaciones y Ajax (*Asynchronous JavaScript and XML*), gracias a una API fácil de usar que funciona en la mayoría de los navegadores.

3.4.2 BootStrap

Bootstrap [14] es un kit de herramientas de *front-end* con componentes prediseñados, es decir, un *framework* que facilita el desarrollo web del cliente de manera *responsive*, además es de código abierto.

3.5 HTML5

HyperText Markup Language [15] , es un lenguaje de marcado que permite definir la estructura del documento web mediante etiquetas, de una manera sencilla para los desarrolladores y fácil de interpretar por el navegador. En su última versión introduce una serie de nuevas etiquetas y atributos para ampliar la funcionalidad.

3.6 CSS

Cascading Style Sheets [16], es un lenguaje de estilos utilizado para diseñar páginas web, permitiendo cambiar fácilmente su diseño, tipografía, efectos visuales y colores.

3.7 Android Studio

Android Studio [17] es el entorno de desarrollo integrado (IDE) oficial y desarrollado por Google, enfocado al desarrollo de aplicaciones nativas Android.

Proporciona herramientas de alta calidad como emuladores móviles, documentación de código, depuración y gestión de errores para facilitar el desarrollo de las funcionalidades. Admite código en varios lenguajes, Kotlin, Java y C++, siendo el primero el más recomendado en la actualidad para el desarrollo de aplicaciones nativas Android. Su principal característica para destacar es la interfaz integrada de diseño de las vistas, con componentes nativos.

Los lenguajes que se manejan en este trabajo en este entorno son los siguiente:

3.7.1 Kotlin

Kotlin es un lenguaje de programación orientado a objetos caracterizado por la inferencia de tipos en las variables y diseñado para ser totalmente interoperable con Java. Su principal característica es su control sobre los objetos nulos, ofreciendo mecanismos de seguridad para no obtener excepciones de tipo *nullpointer*. El entorno de desarrollo de Android Studio traduce automáticamente código Java a Kotlin o recomienda cambios de seguridad en el código, facilitando el desarrollo en este lenguaje.

3.7.2 XML

XML [18] *Extensible Markup Language*, es un metalenguaje similar a HTML, utiliza un conjunto de reglas para codificar documentos en un formato que sea compatible con distintas tecnologías. En aplicaciones Android se utiliza para el diseño de las interfaces de usuario.

3.8 Firebase

Firebase [19] es una plataforma adquirida y desarrollada por Google para facilitar el desarrollo de aplicaciones web y móviles. Se encuentra alojada en la nube y facilita la escalabilidad. Los módulos de Firebase utilizados en este trabajo son los siguientes:

3.8.1 Firebase Authentication

Proporciona servicios de autenticación con funcionalidad preparada para registrar a los usuarios de la aplicación y mantener el detalle de la sesión y su estado para los distintos usuarios.

3.8.2 Cloud Firestore

Base de datos documental NoSQL utilizada para almacenar y sincronizar datos de aplicaciones en tiempo real.

3.8.3 Cloud Storage

Servicio de almacenamiento que permite guardar contenido más pesado, como imágenes o videos de las aplicaciones.

Capítulo 4 - Arquitectura de la aplicación

En este quinto capítulo se explica la arquitectura empleada en las aplicaciones desde un punto de vista global, específico para cada una y finalmente el modelo de datos común.

4.1 Estructura del sistema

El sistema está formado por las dos aplicaciones, necesarias para completar este trabajo y con las que se forma la siguiente arquitectura, desde un punto de vista global (Figura 2).

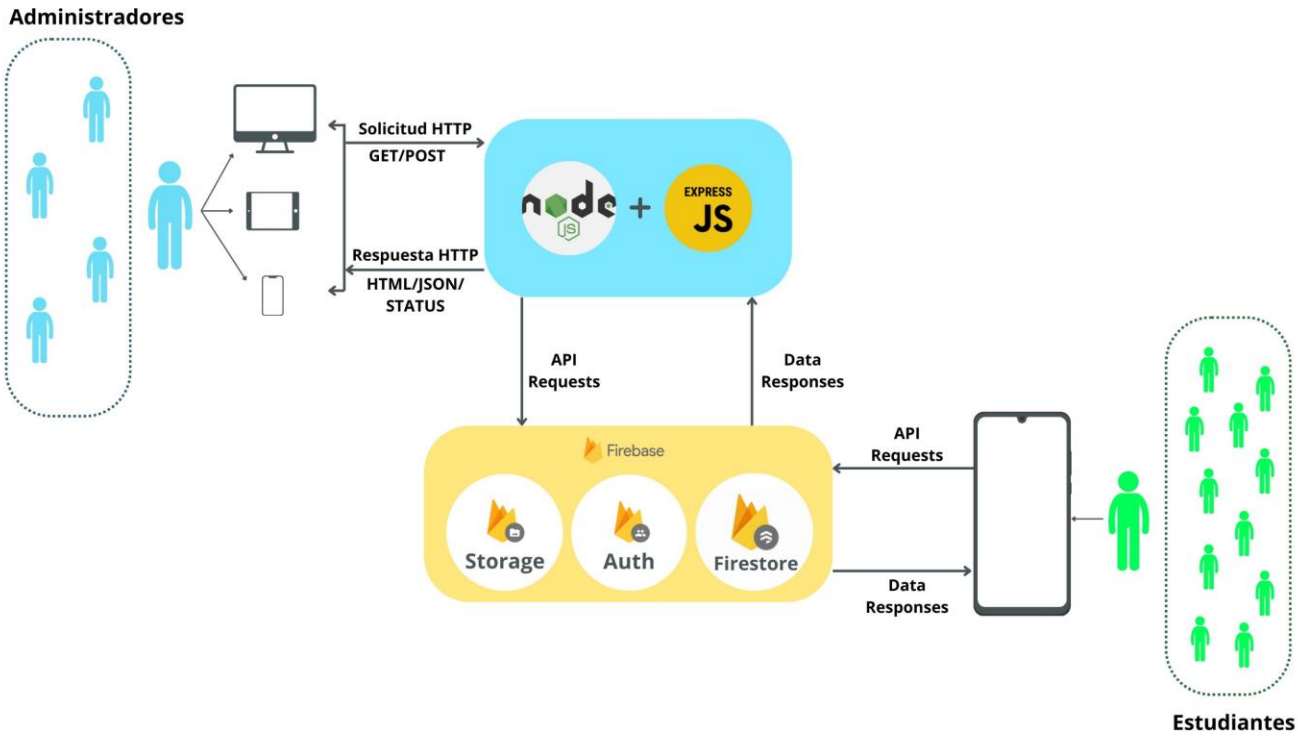

Figura 2 Arquitectura global del sistema

Un grupo reducido de responsables administradores, previamente registrados en el sistema, interactúan con él mediante la aplicación web a través de cualquier

dispositivo con conexión a internet (en el caso de estar ya alojada en un servidor en producción). Se realizan solicitudes http a los distintos *endpoints* del *backend de* Node, el cual obtendrá, si la solicitud es válida, los datos necesarios de Firebase y los devolverá en formato JSON junto al estado de la solicitud o un nuevo renderizado.

Por otro lado, los usuarios estudiantes, haciendo uso de la aplicación nativa en Android, interactúan directamente con el Firebase, sin necesidad de desplegar una nueva arquitectura *backend* ya que está integrada en el propio servicio que ofrece. Se interactúa a través de la API y Firebase se encarga tanto de acceder a la base de datos mediante Firestore Firebase o al almacenamiento de imágenes mediante Firebase Storage.

Igualmente, a la hora de iniciar sesión y registrarse se realizan peticiones desde las aplicaciones y se accede a través de Firebase Authentication en la parte de cliente.

En conclusión, ambas aplicaciones independientemente de su tecnología se comunican a través de la misma base de datos. La estructura individual de cada una se detalla más en profundidad a continuación.

4.2 Estructura de la aplicación web

El tipo de funcionamiento interno que presenta el uso de la tecnología de Node y Express, se caracteriza por el renderizado desde el servidor (*Server Side Rendering*). Esto quiere decir, que cada vez que se necesita actualizar los datos de una vista ya alojada en el cliente, se necesita realizar una nueva petición de actualización, mediante el uso de la funcionalidad que cambia de vista de la aplicación, o simplemente pulsando en el botón del navegador de refrescar la página. De esta manera se carga en el navegador de manera dinámica nuevos contenidos HTML, CSS y JavaScript enviados en bloque desde el servidor.

Por otro lado, una vez renderizada la vista en el cliente, en el caso particular de la aplicación web, se incluye una llamada mediante el JavaScript del navegador a la Api de Google Charts, consiguiéndose mostrar las gráficas de resultados con la información recuperada de Firestore. Esto añade a la estructura planteada un nuevo

renderizado, pero esta vez del lado del cliente (*Client Side Rendering*) como podemos observar en el esquema de la siguiente imagen (Figura 3).

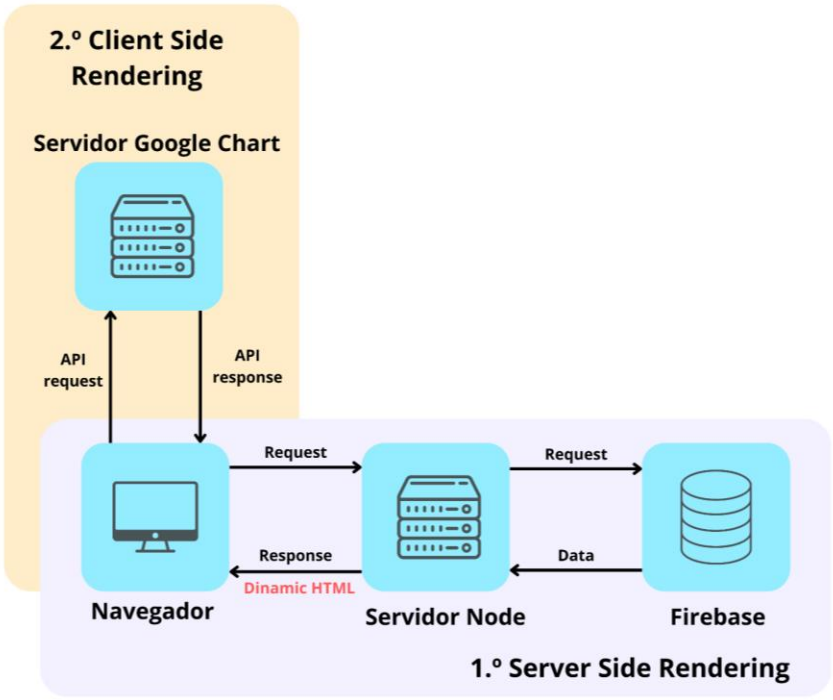

Figura 3 Arquitectura de la aplicación web

Este funcionamiento del sistema plantea un principal inconveniente, un alto número de peticiones de renderizado al servidor. Teniendo en cuenta que la aplicación en su contexto solo será usada por un determinado número pequeño de administradores, no supone un problema real. Pero para evitarlo, se ha utilizado la funcionalidad AJAX de la librería de JQuery, la cual nos posibilita la opción de manejar los datos devueltos por nuestras peticiones al servidor y modificar dinámicamente el HTML de la vista, aunque en la mayoría de las ocasiones es necesario realizar un refresco igualmente tras ello.

4.3 Estructura de la aplicación móvil

La aplicación desarrollada en Kotlin para móviles nativos Android, se ha desarrollado utilizando las prácticas recomendadas por Google para el desarrollo de aplicaciones móviles Android en la actualidad. Estas son, el uso de la arquitectura de programación Model View ViewModel representada en la siguiente imagen (Figura 4) y el uso de argumentos seguros (SafeArgs) para la comunicación entre distintos fragmentos alojados en una actividad principal contenedor, haciendo la aplicación *Single Activity*.

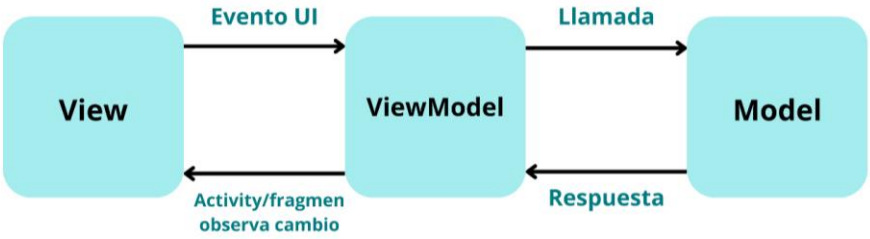

Figura 4 Arquitectura Model View ViewModel

La vista solo es responsable de la funcionalidad relativa a los cambios en la misma y no contiene lógica de acceso a base de datos. Las distintas consultas, se encuentran en objetos que representan el modelo y la interacción entre la vista y el modelo, se realiza a través de objetos ViewModel. Este se establece para cada vista, que además de actuar como puente de las peticiones al desacoplar la lógica de base de datos de los fragmentos, actúa como agente que persiste la información y datos mostrada en los componentes de la vista. Además, realiza una función clave en esta arquitectura, pues la información almacenada, se puede declarar si es necesario como mutable y observar sus cambios desde la vista, para mantener la información dinámicamente actualizada en la aplicación, fomentando la correcta experiencia del usuario.

Hay que diferenciar el uso de dos tipos de ViewModel usados en este trabajo, el simple, que crea una instancia distinta en memoria cada vez que es declarado en una vista, y el que mismas declaraciones comparten instancia internamente. Este último es

el caso del ViewModel de actividad, pues aloja la primera instancia que se crea en la actividad principal, no en el fragmento, es el caso de su uso en el grupo de funcionalidades relacionadas con el perfil.

Por otro lado, la navegación en este tipo de arquitectura se realiza mediante el objeto de navegación NavController de Android Jetpack Navigation Component. Este brinda funcionalidad dedicada al cambio de fragmentos dentro de su contenedor, resultando en una estructura interna que maneja distintas pilas de actividad al asociar este componente con la barra de navegación y sus botones. Cada botón, maneja una *back-stack* distinta como se puede ver en la Figura 5, así cada vez que se cambia entre los grupos de vistas asociadas a cada botón, al regresar, se visualiza el fragmento donde se encontraba el usuario en la navegación de la pila de dicho grupo.

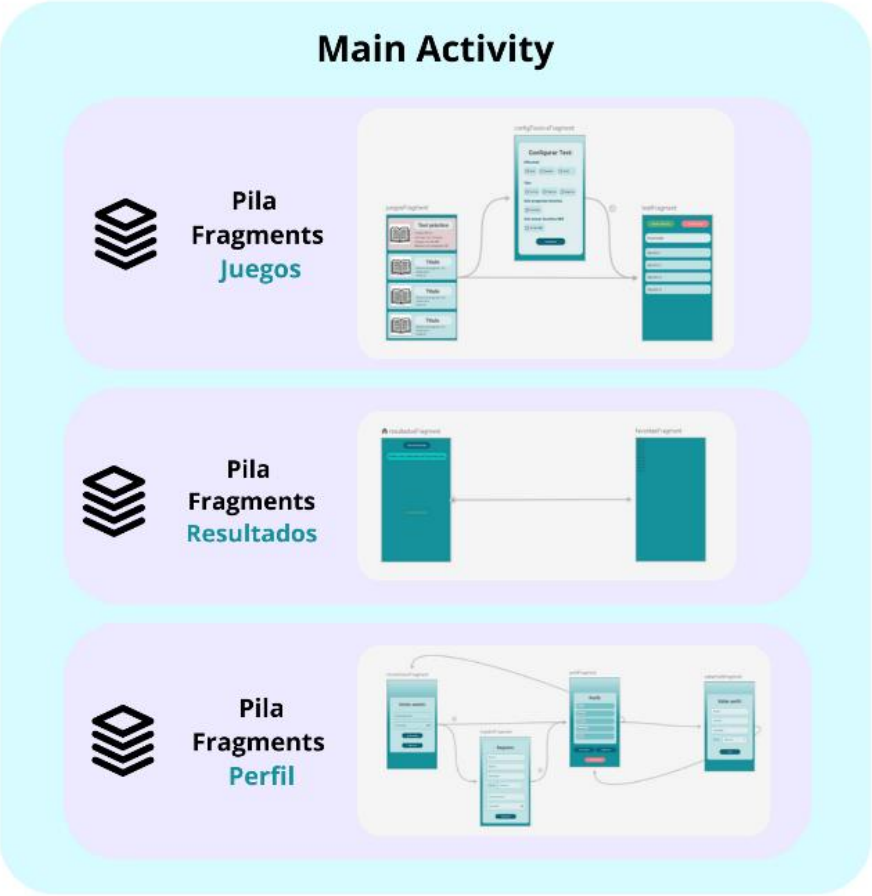

Figura 5 Estructura interna de navegación aplicación móvil

4.4 Modelo de datos

La organización de la información a manejar en el sistema y su almacenamiento ha sido realizada mediante los servicios de Firebase Firestore. Esta es una base de datos no relacional basada en la organización de la información en colecciones que contienen documentos. Además, al usar Firebase Authentication para el registro de los usuarios, se usa la colección del sistema de Authentication mostrado en la Figura 6.

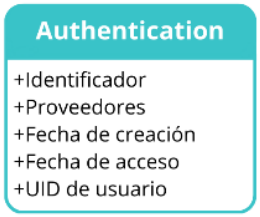

Figura 6 Colección Athentication de Firebase

Para el almacenamiento de los datos referentes a las imágenes, se han seguido dos procesos lógicos. Primero se almacena la imagen en el servicio de Firebase Storage, que mantiene su propio modelo de datos interno para el almacenamiento y segundo, se guarda el enlace creado del tipo URL en el atributo del documento correspondiente, ya que de esta forma se puede recuperar la imagen cuando sea necesario.

Para realizar el diseño de las distintas colecciones y documentos se ha seguido el proceso lógico de extracción de la información mínima necesaria de los requisitos de la aplicación. Como la base de datos es no relacional, la forma natural de representar los datos almacenados sería en forma de arbol, pero al no manejar muchos niveles de profundidad con subcolecciones se simplifica la representación y se ha obtenido el diseño de la Figura 7.

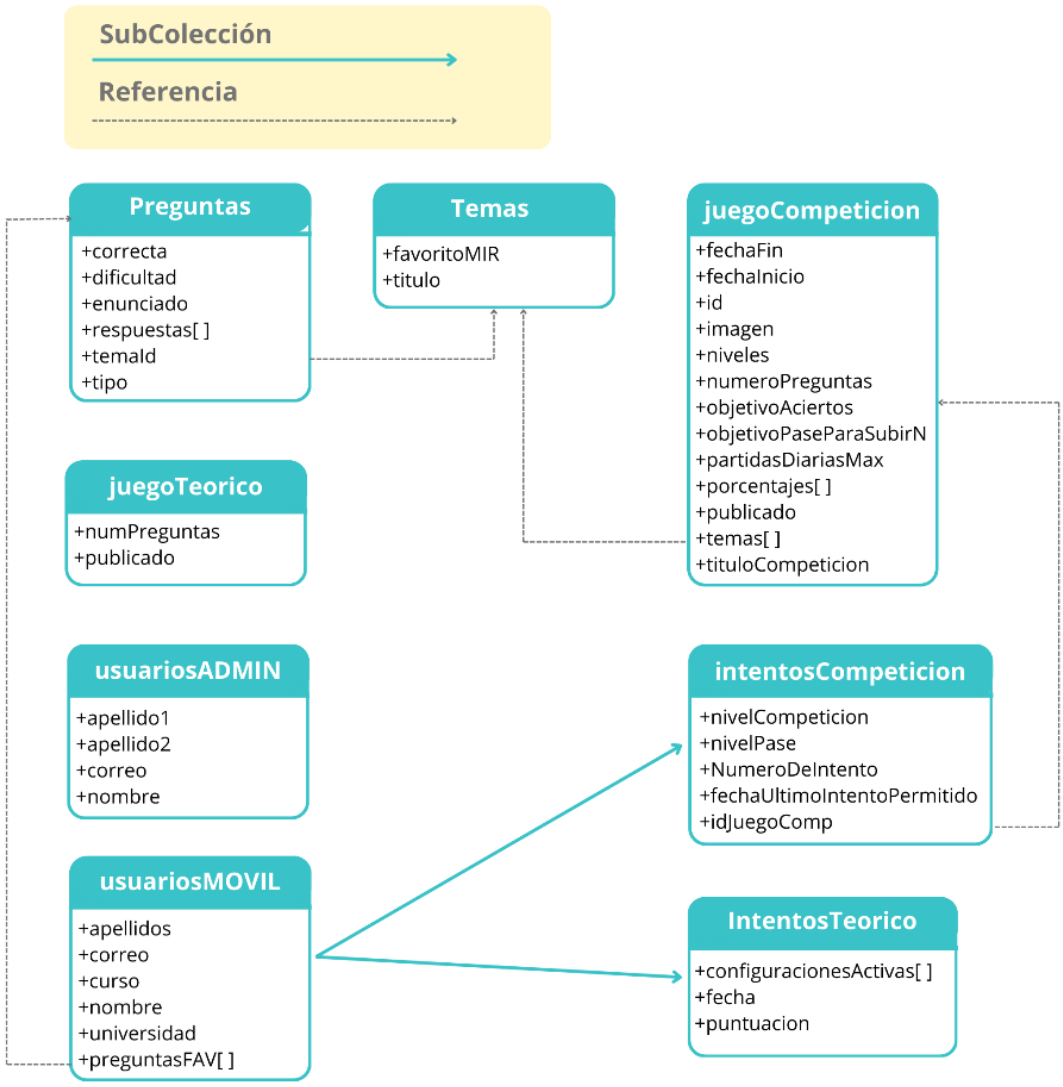

Figura 7 Esquema diseño base de datos no relacional utilizada

Cabe destacar, que durante el desarrollo de las funcionalidades se ha tenido en cuenta mantener la consistencia de las distintas referencias indicadas en la figura anterior para mantener la integridad y evitar inconsistencias.

A continuación, se detallan las distintas colecciones.

4.4.1 Colección Preguntas

La colección de preguntas contiene distintos documentos con la información que se observa en la Figura 8. Es utilizado para almacenar los tres tipos de preguntas, el de la imagen hace referencia a una pregunta de tipo práctica. Para los otros dos tipos de preguntas, es igual, pero sin el campo de la imagen y con dos respuestas para el caso de pregunta negativa. La aplicación web es la que realiza la creación y los usuarios móviles las recuperan para mostrarlas a la hora de realizar los tests.

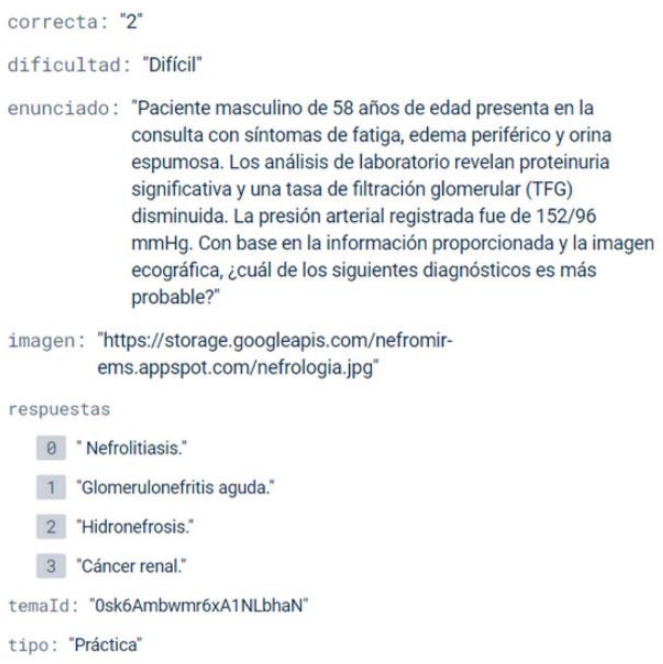

Figura 8 Documento de la colección de Preguntas

4.4.2 Colección Temas

La colección de temas tiene como objetivo almacenar de una manera simple los distintos temas que son creados por los administradores con su título y si son favoritos del MIR. Estos temas son referenciados por distintas preguntas que pertenecen a ese tema. En la siguiente imagen (Figura 9) se observa la estructura de esta colección.

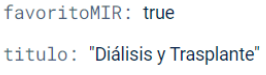

Figura 9 Documento de la colección de Temas

4.4.3 Colección juegoTeorico

Los datos que forman un juego teórico son simplemente su número de preguntas, y su estado publicado, que puede estar false o true para indicar si debe aparecer al usuario o no por motivos de mantenimiento. Este tipo de documento de esta colección se puede observar en la siguiente imagen (Figura 10).

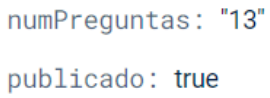

Figura 10 Documento único de la colección de juegoTeorico

4.4.4 Colección juegosCompetición

La colección que representa los distintos juegos de competición almacena en sus distintos documentos información relativa al periodo en el que estará disponible al usuario móvil, además de la imagen opcional que se añade al ser creado por el administrador web. También almacena la información por defecto del juego común a todos de la lógica de subir de nivel, es decir las 5 partidas diarias que puedes jugar (partidasDiariasMax), el objetivo de pases superados en el día para subir de nivel (objetivoPaseParaSubirN) y los 10 niveles a escalar en el ranking del juego (niveles). Igualmente se almacena el número de preguntas de ese juego con el número de preguntas que debe acertar el usuario para considerar el juego ganado y que supere el pase que gastó al iniciarlo. Finalmente, el resto de los campos que definen el juego son su título, las referencias de los temas que entran y los porcentajes de los tipos de preguntas de los temas que deben componer el juego cuando se cree.

Todo esto se puede observar en la siguiente imagen, Figura 11.

Figura 11 Documento de la colección de juegoCompeticion

4.4.5 Colección usuariosADMIN

Los documentos de esta colección están pensados para almacenar la información que representa los datos del administrador (Figura 12). Estos se crean si el registro del usuario es exitoso, cumpliendo la precondición de estar previamente su correo registrado y almacenado por los administradores en la base de datos por motivos de seguridad.

Figura 12 Documento de la colección de usuariosADMIN

4.4.6 Colección usuariosMOVIL

La colección que almacena los distintos datos referentes a un usuario estudiante es usuariosMOVIL, que es creada al registrarse con éxito en la aplicación móvil. Almacena los datos relativos al usuario como su nombre, apellidos, curso, universidad y las referencias a las distintas preguntas que marca como favoritas al realizar los tests (Figura 13).

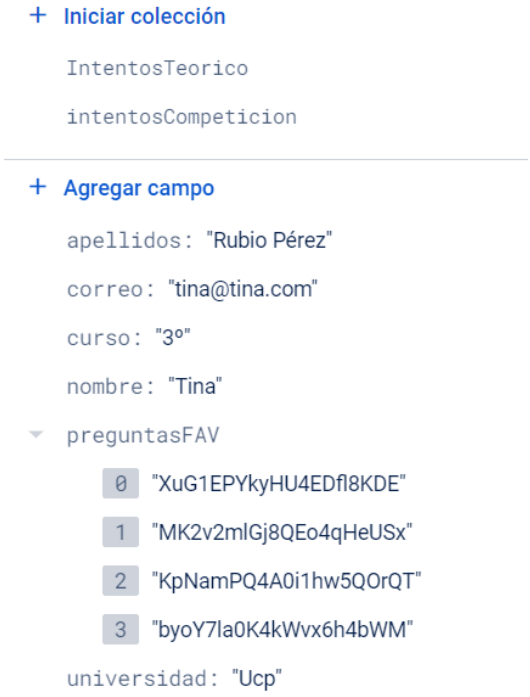

Figura 13 Documento de la colección usuariosMOVIL

Además, cuando el usuario juega al juego teórico o a algún juego de competición se crean las colecciones IntentosTeorico e intentosCompetición para almacenar sus resultados. Se detallan a continuación.

4.4.6.1 Subcolección IntentosTeorico

Esta subcolección representa en documentos cada intento de realizar el juego de práctica teórico por parte del usuario, almacenando la información de la configuración del intento del test, es decir los valores que el usuario marcó para personalizarlo, la fecha y la puntuación obtenida (Figura 14).

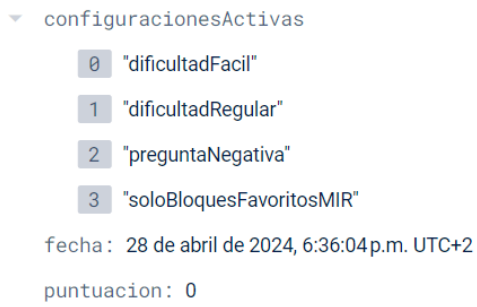

Figura 14 Documento de la subcolección de intentosTeorico

4.4.6.2 Subcolección intentosCompeticion

El siguiente documento (Figura 15) representa la información relativa a un intento por parte de un usuario móvil de realizar un juego de competición específico (idJuegoComp), almacenando el número de intentos del día, el nivel de los 5 pases permitidos ganados y el nivel que se encuentra del total de 10.

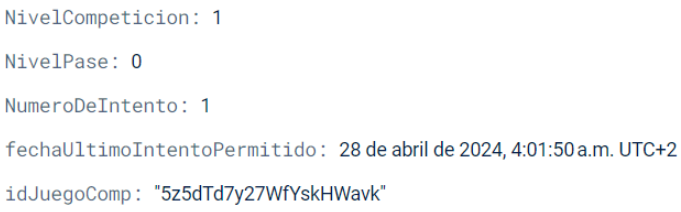

Figura 15 Documento de la subcolección de intentosCompeticion

Capítulo 5 - Diseño

En cuanto al diseño del sistema, hay que diferenciar las dos aplicaciones, ya que al usar tecnologías distintas enfocadas a distintos dispositivos y objetivos el diseño cambia.

5.1 Diseño aplicación web

La interfaz de usuario de la aplicación web está diseñada y creada de manera que sea *responsive* con los distintos tamaños de pantalla. Esto se consigue utilizando las clases que proporciona BootStrap, las cuales están pensadas para esta problemática, como es el caso de la barra de navegación, que se colapsa al superar un determinado ancho de pantalla y se pone en formato móvil. El diseño y colores están adaptados para que la experiencia del usuario sea cómoda y agradable, utilizando colores representativos para cada funcionalidad, mediante clases BootStrap igualmente, como pueden ser la *Danger*, para los botones que indican precaución, coloreándose de rojo, o *Succes* para colorear textos o botones de verde.

Además, la tecnología web empleada, nos permite una alta personalización de las vistas mediante el CSS, con el cual se han aplicado diversos colores y formatos a las clases o elementos HTML para obtener un resultado agradable y sin estar sobrecargado.

Desde el punto de vista de navegación, las acciones son intuitivas, posicionando los botones de cambio de vista en la barra de navegación superior. Además, se añade sombreado *hover* a los elementos botones o filas de las tablas para poder indicar que su pulsación inicia una acción o navegación.

5.2 Diseño aplicación móvil

En cuanto a la aplicación móvil, su diseño se ha desarrollado con los elementos proporcionados por el sistema para manejar las vistas y además con clases de la librería de Google de MaterialComponents para el estilo de los inputs.

Esto se ha llevado a cabo gracias a los XML, editados a través de la interfaz del entorno de desarrollo de Android Studio o en los propios archivos. Se han añadido principalmente botones y tarjetas que cuentan con un formato predefinido que hace que sean fácilmente interpretables por el usuario. Los colores empleados han sido una gama de azules, para adaptar una visualización cómoda al usuario en cualquier condición de luz externa.

Capítulo 6 - Funcionalidades

En este capítulo se describen las funcionalidades de la aplicación divididas por módulos:

- Módulo gestión de usuarios
- Módulo juego teórico
- Módulo bloques temáticos
- Módulo juegos de competición
- Módulo gestión de resultados
- Módulo jugar

6.1 Módulo gestión de usuarios

6.1.1 Registrarse

Se diferencian la funcionalidad implementada para cada aplicación.

6.1.1.1 Administrador web

El usuario administrador, si cumple que se ha sido introducido previamente como administrador por los responsables del sistema, puede registrarse con sus datos en la interfaz web Figura 16 . Cuando se pulsa en el botón de registrar del modal de la vista, el *endpoint* que activa del servidor comprueba el requisito anteriormente comentado y si lo cumple, procede a registrar al usuario, extrayendo sus datos del cuerpo de la solicitud, se registra al usuario en Authentication y se introducen sus datos en su colección de Firebase Firestore, finalmente se activa la sesión de Express con los datos. Este proceso se puede observar con su gestión de errores enviados al cliente en la siguiente Figura 17.

Figura 16 Funcionalidad web Registro

```
router.post('/registrar', (req, res) => {
  const { correo, nombre, apellido1, apellido2, password } = req.body;
  db.collection('usuariosADMIN').where('correo', '==', correo).get()
    .then(snapshot => {
      if (snapshot.empty) {
        res.status(400).json({ success: false, message: 'Correo no autorizado para registro.' });
      } else {
        admin.auth().createUser({
          email: correo,
          emailVerified: false,
          password: password,
          displayName: `${nombre} ${apellido1} ${apellido2}`,
          disabled: false
        })
          .then(userRecord => {
            const docId = snapshot.docs[0].id;
            db.collection('usuariosADMIN').doc(docId).update({
              nombre: nombre,
              apellido1: apellido1,
              apellido2: apellido2,
            })
              .then(() => {
                req.session.user = { correo: correo, nombre: nombre, apellido1: apellido1, apellido2: apellido2 };
                res.status(200).json({ success: true, message: 'Registrado con éxito' });
              })
              .catch(error => {
                res.status(500).json({ success: false, message: 'Error al actualizar el usuario.' });
              });
          })
          .catch(error => {
            res.status(500).json({ success: false, message: 'Error al registrar el usuario.' });
          });
      }
    })
    .catch(error => {
      res.status(500).json({ success: false, message: 'Error al consultar autorización para registro.' });
    });
```

Figura 17 Código funcionalidad registro web

6.1.1.2 Usuario móvil

El usuario móvil puede registrarse a través de la vista Figura 18, donde tras introducir los campos requeridos y necesarios, pulsando el botón Registrarse, desencadena la funcionalidad que comienza en la vista, atraviesa el ViewModel y finalmente registra al usuario en el objeto del modelo (Figura 19), creando el usuario en Authentication e insertando los datos recibidos en la colección correspondiente.

Figura 18 Funcionalidad registrarse móvil

```
fun registerUser(email: String, password: String, nombre: String, apellidos: String,
                 universidad: String, curso: String, onResult: (Boolean, String?) -> Unit) {
    auth.createUserWithEmailAndPassword(email, password)
        .addOnCompleteListener { task ->
            if (task.isSuccessful) {
                val userId = auth.currentUser?.uid ?:
                return@addOnCompleteListener onResult(false, "No se pudo obtener el ID del usuario.")
                val user = hashMapOf(
                    "nombre" to nombre,
                    "correo" to email,
                    "apellidos" to apellidos,
                    "universidad" to universidad,
                    "curso" to curso
                )
                createUserInFirestore(userId, user) { success, firestoreErrorMessage ->
                    onResult(success, firestoreErrorMessage)
                }
            } else {
                onResult(false, task.exception?.message)
            }
        }.addOnFailureListener { e ->
            onResult(false, e.message)
        }
}
```

Figura 19 Código funcionalidad registrarse móvil

6.1.2 Iniciar sesión

Se diferencian la funcionalidad implementada para cada aplicación.

6.1.2.1 Administrador web

El administrador puede iniciar sesión tras registrarse con éxito en el modal que se observa en la Figura 20. El botón de iniciar inicia la funcionalidad en el JavaScript del cliente de inicio de sesión de Firebase para web (Figura 21) y la petición Ajax realiza la verificación del token en el servidor y modifica la sesión de Express con los nuevos datos. Si ocurre algún error se muestra los mensajes correspondientes, manejando el DOM con JQuery.

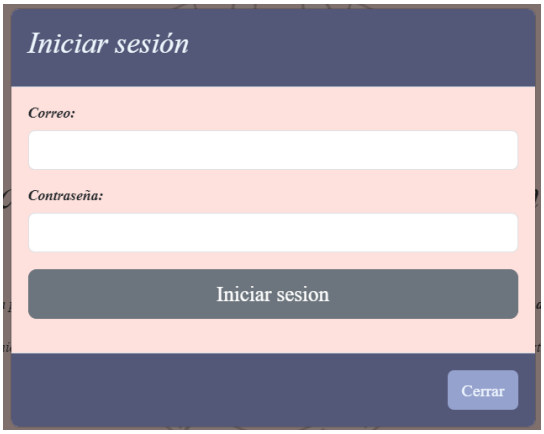

Figura 20 Funcionalidad inicio de sesión web

```
//Gestionamos el envio de datos al iniciar sesion y genera el token de acceso
$('#formIniciarSesion').submit(function (e) {
    e.preventDefault();
    var correo = $('#inputCorreo-is').val();
    var password = $('#inputPasswd-is').val();
    // Con la instancia de firebase del cliente, logeamos con correo y contraseña para obtener
    // las credenciales y mandar el token al servidor
    firebase.auth().signInWithEmailAndPassword(correo, password)
        .then((userCredential) => {
            userCredential.user.getIdToken().then(function (idToken) {
                // Enviamos el token de ID al servidor para su verificación
                $.ajax({
                    url: '/barraNav/verificarToken',
                    method: 'POST',
                    data: { idToken: idToken },
                    success: function (response) {
                        $('#cualquierErrorIni').removeClass('text-danger').addClass('text-success').text(response.message);
                        $('#formIniciarSesion').addClass('d-none');
                        setTimeout(function () {
                            window.location.href = '/';
                        }, 1300);
                    },
                    error: function (xhr, status, error) {
                        $('#cualquierErrorIni').addClass('text-danger').text(xhr.responseJSON.message);
                    }
                });
            });
        })
        .catch((error) => {
```

Figura 21Código funcionalidad inicio de sesión web

6.1.2.2 Usuario móvil

El usuario móvil puede realizar el inicio de sesión mediante la vista de la Figura 22. Cuando se presiona el botón de iniciar sesión se inicia la funcionalidad atravesando el ViewModel y ejecutando el código del modelo (Figura 23), donde se realiza el inicio en Authentication. En caso de éxito se actualiza el estado de la sesión en sharedPreferences y se inicia una navegación hacia el perfil.

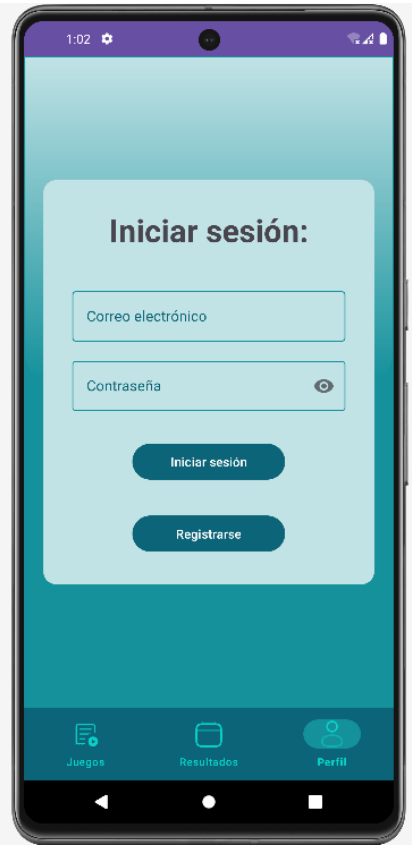

Figura 22 Funcionalidad de inicio de sesión móvil

```
fun signIn(email: String, password: String, onResult: (Boolean, String?) -> Unit) {
    auth.signInWithEmailAndPassword(email, password)
        .addOnCompleteListener { task ->
            if (task.isSuccessful) {
                onResult(true, null)
            } else {
                onResult(false, task.exception?.message)
            }
        }
}
```

Figura 23 Código funcionalidad de inicio de sesión móvil

6.1.3 Cerrar sesión

Se diferencian la funcionalidad implementada para cada aplicación.

6.1.3.1 Administrador web

El cierre de sesión se activa cuando se tiene la sesión iniciada y el administrador pulsa en el botón de cerrar sesión que se observa en la Figura 24. Este acto, activa la lógica JavaScript de cierre de sesión que se muestra en la Figura 25, donde se observa que se cierra la sesión del cliente de Firebase Authentication y además la llamada Ajax, realiza en el servidor la eliminación de la sesión de Express asociada a la cookie de sesión del cliente.

Figura 24 Funcionalidad cerrar sesión web

```
$('#btnCerrarSesion').on('click', function (event) {
    firebase.auth().signOut().then(function () {
        $.ajax({
            type: 'GET',
            url: '/barraNav/cerrarSesion',
            success: function (data) {
                window.location.href = '/';
            },
            error: function (xhr, status, error) {
                alert('Error al cerrar sesión en el servidor. Por favor, inténtalo de nuevo.');
            }
        });
    }).catch(function (error) {
        alert('Error al cerrar sesión en Firebase. Por favor, inténtalo de nuevo.');
    });
});
```

Figura 25 Código funcionalidad cerrar sesión web

6.1.3.2 Usuario móvil

El cierre de sesión en la aplicación Android se realiza pulsando el botón de Cerrar sesión de la vista del perfil, como se observa en la Figura 26. A nivel de código, el flujo se inicia en la funcionalidad del *onClick* del botón de la vista, que a través del ViewModel ejecuta la funcionalidad del modelo que se observa en la Figura 27. Si se completa con éxito en el fragmento se actualiza la variable de sesión de sharedPreferences y se inicia una navegación a la vista de inicio de sesión.

Figura 26 Funcionalidad cierre de sesión móvil

```
fun signOut() {
    auth.signOut()
}
```

Figura 27 Código funcionalidad cierre de sesión móvil

6.1.4 Modificar perfil

Se diferencian la funcionalidad implementada para cada aplicación.

6.1.4.1 Administrador web

El usuario administrador una vez iniciada sesión puede modificar los datos introducidos en su registro, pulsando el botón de Mi perfil de la vista principal y cambiando sus datos actuales por los nuevos en el modal que se abre (Figura 28). Esta acción ejecuta el código del *endpoint* correspondiente (Figura 29), al cual se le aplica el middleware de comprobación del estado de sesión del usuario. Se busca el documento del usuario en Firestore usando el correo asociado, y se actualizan los campos del nombre y apellidos en Firebase. Si hay éxito, también se actualizan estos datos en la sesión del usuario.

Figura 28 Funcionalidad modificar perfil web

```
router.post('/modificarDatosPerfil', middlewareIsLoggedInAndValidated, (req, res) => {
  const { nombre, apellido1, apellido2 } = req.body;
  if (!req.session.user) {
    return res.status(401).json({ success: false, message: 'Usuario no autenticado.' });
  }
  const { correo } = req.session.user;
  const usuariosAdminRef = db.collection('usuariosADMIN').where('correo', '==', correo);

  usuariosAdminRef.get()
    .then(snapshot => {
      if (snapshot.empty) {
        return res.status(400).json({ success: false, message: 'Usuario no encontrado en la base de datos.' });
      }
      const doc = snapshot.docs[0];
      const docId = doc.id;
      const docRef = db.collection('usuariosADMIN').doc(docId);

      return docRef.update({
        nombre: nombre,
        apellido1: apellido1,
        apellido2: apellido2
      })
        .then(() => {
          req.session.user.nombre = nombre;
          req.session.user.apellido1 = apellido1;
          req.apellido2 = apellido2;
          res.status(200).json({ success: true, message: 'Datos de perfil actualizados con éxito.' });
        })
        .catch(error => {
          res.status(500).json({ success: false, message: 'Error al actualizar los datos del perfil.' });
        });
    })
    .catch(error => {
      res.status(500).json({ success: false, message: 'Error al obtener el documento.' });
    });
});
```

Figura 29 Código funcionalidad modificar perfil web

6.1.4.2 Usuario móvil

Para modificar los datos del perfil hay que pulsar en el botón de editar en el perfil del usuario móvil, mostrándose la nueva vista de editar el perfil (Figura 30). Tras rellenar los campos con los nuevos datos se pulsa en el botón de editar, lo que dispara la funcionalidad a través de la arquitectura MVVM presentada anteriormente y se ejecuta el código del modelo (Figura 31) donde se crea un nuevo mapa con los datos recibidos y se inserta en la colección correspondiente.

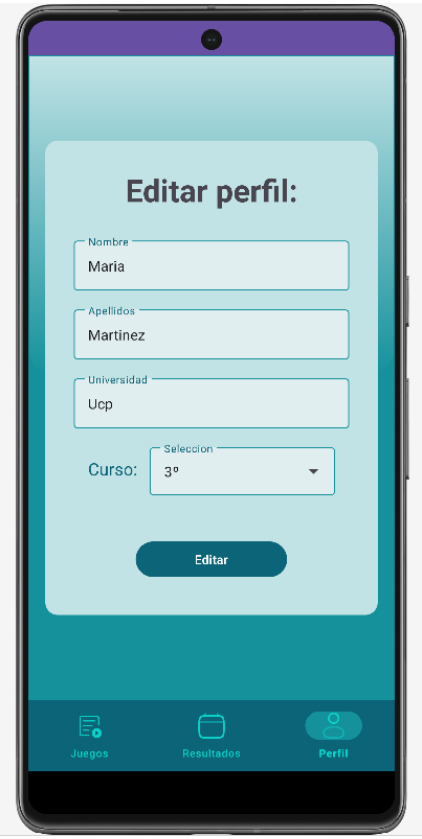

Figura 30 Funcionalidad editar perfil móvil

```
fun updateUserData(nombre: String, apellidos: String, universidad: String,
                   curso: String, onResult: (Boolean, String?) -> Unit) {
    val userId = auth.currentUser?.uid ?: return onResult(false, "Usuario no identificado")
    val userData: Map<String, Any> = hashMapOf(
        "nombre" to nombre as Any,
        "apellidos" to apellidos as Any,
        "universidad" to universidad as Any,
        "curso" to curso as Any
    )
    db.collection( collectionPath: "usuariosMOVIL").document(userId) DocumentReference
        .update(userData) Task<Void!>
        .addOnSuccessListener { it: Void!
            onResult(true, null)
        }
        .addOnFailureListener { exception ->
            onResult(false, exception.message)
        }
}
```

Figura 31 Código funcionalidad modificar perfil móvil

6.1.5 Eliminar perfil

El estudiante usuario móvil, al iniciar sesión con una cuenta registrada, puede eliminar su cuenta pulsando en el botón correspondiente dentro del fragmento de su perfil mostrándose un dialogo de confirmación (Figura 31). Si se procede con la confirmación de eliminación, se ejecuta el código de la Figura 33, eliminándose el usuario de Authentication y de la colección de Firestore correspondiente.

Figura 32 Funcionalidad eliminar perfil móvil

```
fun deleteUserData(onResult: (Boolean, String?) -> Unit) {
    val user = auth.currentUser
    user?.let { firebaseUser ->
        firebaseUser.delete()
            .addOnCompleteListener { task ->
                if (task.isSuccessful) {
                    db.collection( collectionPath: "usuariosMOVIL").document(firebaseUser.uid) DocumentReference
                        .delete() Task<Void!>
                        .addOnSuccessListener { it: Void!
                            onResult(true, null)
                        }
                        .addOnFailureListener { exception ->

                            onResult(false, exception.message)
                        }
                } else {
                    onResult(false, task.exception?.message)
                }
            }
    } ?: run { this: PerfilModel
        onResult(false, "No hay un usuario autenticado.")
    }
}
```

Figura 33 Código funcionalidad eliminar perfil móvil

6.2 Módulo juego teórico

6.2.1 Crear juego teórico

El administrador web con la sesión iniciada, puede crear el juego teórico único en la página web pulsando al botón de crear (Figura 34). Esta acción dispara la apertura de un modal donde se manda introducir el número de preguntas. Una vez creado, se renderiza la página, mostrando esta vez el botón de eliminar. Al confirmar la creación en el modal, se ejecuta el código que se observa en la Figura 35, registrando el juego teórico con su número de preguntas y su estado de publicado a false por defecto.

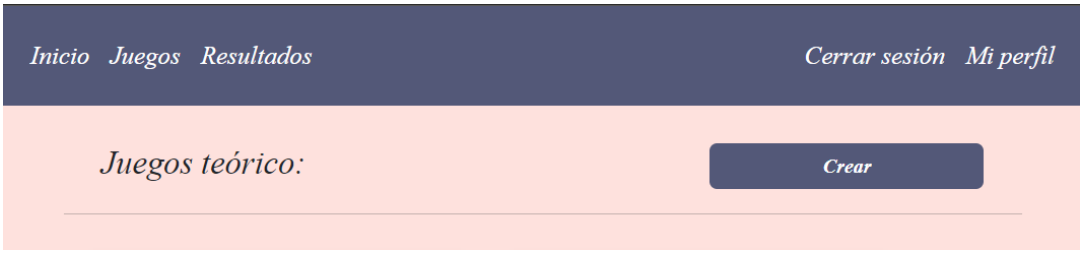

Figura 34 Funcionalidad crear juego teórico

```
router.post('/crearJuegoTeorico', async (req, res) => {
  const numPreguntas = Number(req.body.numpregTeo);
  if (numPreguntas < 10 || numPreguntas > 100 || isNaN(numPreguntas)) {
    return res.status(400).json({ success: false,
      message: 'El número de preguntas debe estar entre 10 y 100.' });
  }
  try {
    let snapshot = await db.collection('juegoTeorico').get();
    if (!snapshot.empty) {
      return res.status(400).json({ success: false,
        message: 'Ya existe un juego teórico en la base de datos y no se permite más de uno.' });
    }
    let docRef = await db.collection('juegoTeorico').add({
      numPreguntas: numPreguntas,
      publicado: false
    });

    if (docRef.id) {
      return res.status(200).json({ success: true,
        message: 'Juego teórico creado con éxito.' });
    } else {
      return res.status(500).json({ success: false,
        message: 'Error al verificar la existencia de juegos teóricos o al crear uno.' });
    }
  } catch (error) {
    return res.status(500).json({ success: false,
      message: 'Error al verificar la existencia de juegos teóricos o al crear uno.' });
  }
});
```

Figura 35 Código funcionalidad crear juego teórico

6.2.2 Modificar juego teórico

El administrador al tener creado el juego teórico único, puede modificarlo pulsando en el botón de editar en la tarjeta que representa el juego en la vista, mostrándose un modal con el número de preguntas a modificar (Figura 36). Una vez se pulsa en guardar cambios del modal se ejecuta el código de actualización del documento (Figura 37) de la base de datos si cumple el requisito de máximo y mínimo de preguntas.

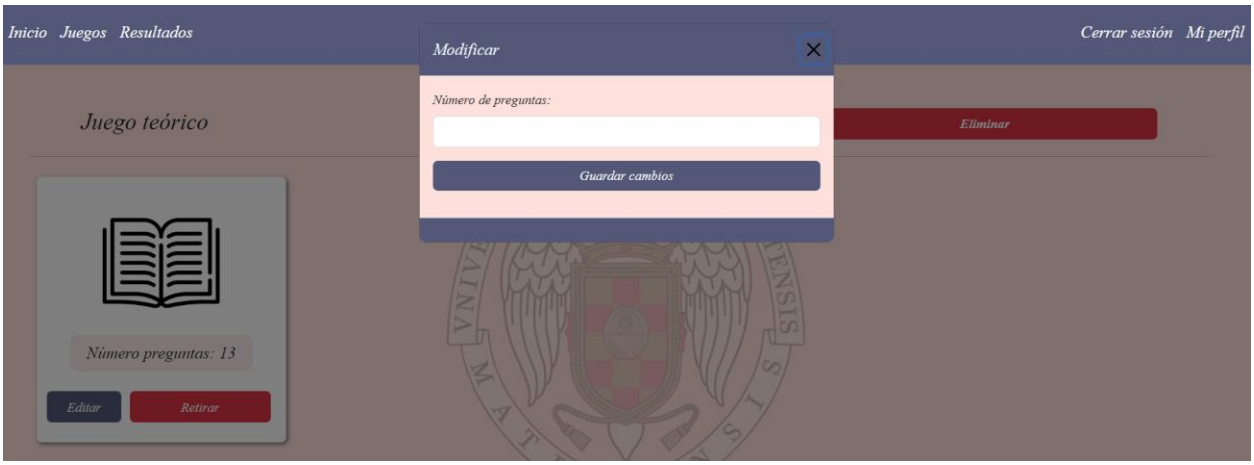

Figura 36 Funcionalidad modificar juego teórico

```
router.post('/modificarTeorico', async (req, res) => {
  const { juegoTeoricoId, numPreguntas } = req.body;

  if (numPreguntas < 10 || numPreguntas > 100) {
    return res.status(400).json({ success: false, message: 'El número de preguntas debe estar entre 10 y 100.' });
  }

  try {
    await db.collection('juegoTeorico').doc(juegoTeoricoId).update({
      numPreguntas: numPreguntas
    });
    res.status(200).json({ success: true, message: 'Número de preguntas actualizado correctamente.' });
  } catch (error) {
    res.status(500).json({ success: false, message: 'Error al actualizar el número de preguntas.' });
  }
});
```

Figura 37 Código funcionalidad modificar juego teórico

6.2.3 Eliminar juego teórico

El administrador con sesión iniciada puede decidir eliminar el juego teórico que almacena los temas y sus preguntas. Esta acción es irreversible, por lo que es de especial cuidado, advirtiéndose en el mensaje de alerta de la confirmación que se muestra en el modal al pulsar en eliminar (Figura 38). Si se confirma la eliminación se ejecuta el *endpoint* correspondiente (Figura 39) donde se elimina el juego teórico, todos los temas y todas las preguntas de sus respectivas colecciones.

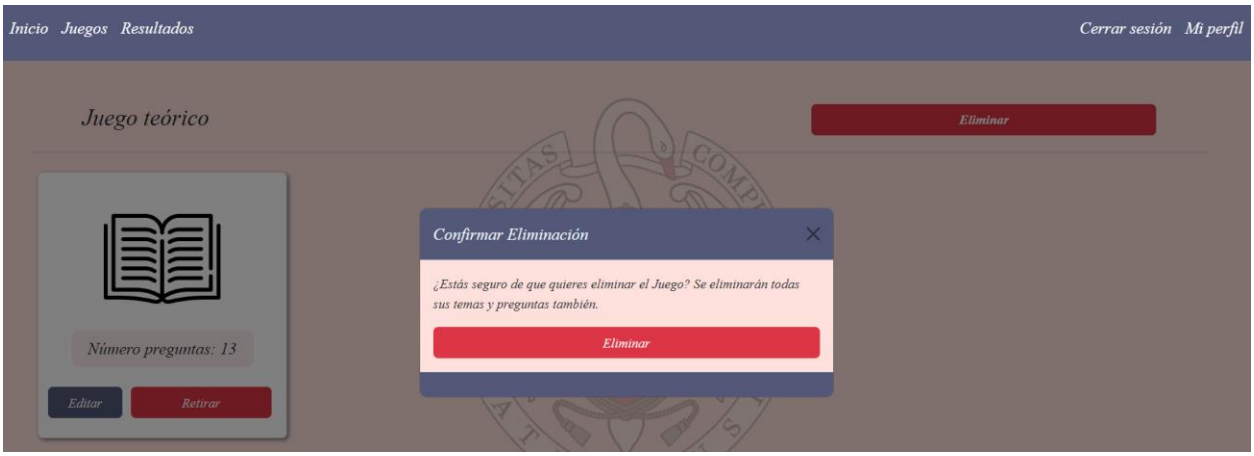

Figura 38 Funcionalidad eliminar juego teórico

```
//eliminar el JuegoTeorico  y las colecciones Temas y Preguntas
router.delete('/eliminarJuegoTeorico/:id', async (req, res) => {
  const juegoTeoricoId = req.params.id;

  try {
    await db.collection('juegoTeorico').doc(juegoTeoricoId).delete();
    const temasSnapshot = await db.collection('Temas').get();
    temasSnapshot.forEach(async (doc) => {
      await db.collection('Temas').doc(doc.id).delete();
    });

    const preguntasSnapshot = await db.collection('Preguntas').get();
    preguntasSnapshot.forEach(async (doc) => {
      await db.collection('Preguntas').doc(doc.id).delete();
    });

    res.status(200).json({ message: 'Juego teórico, temas y preguntas eliminados con éxito' });
  } catch (error) {
    res.status(500).json({ message: 'Error al eliminar el juego teórico y sus dependencias' });
  }
});
```

Figura 39 Código funcionalidad eliminar juego teórico

6.2.4 Listar juegos publicados

Cualquier administrador, tras iniciar sesión puede visualizar en la pestaña de Juegos, la lista de los distintos juegos. En primer lugar, el juego teórico y después los juegos de competición. Se identifica su estado de publicación por el botón que contiene cada tarjeta del juego, rojo o verde que indica retirar o publicar, reflejando así si se encuentra publicado o retirado respectivamente (Figura 40). Esto se consigue para el caso de los juegos de competición, con la ejecución de un bucle que crea tantas tarjetas como juegos recibe la vista EJS paramétrica. Cada tarjeta se crea con la información dinámicamente, insertando el botón que define su estado de publicación (Figura 41) en función a la información del juego analizado en cada iteración del bucle.

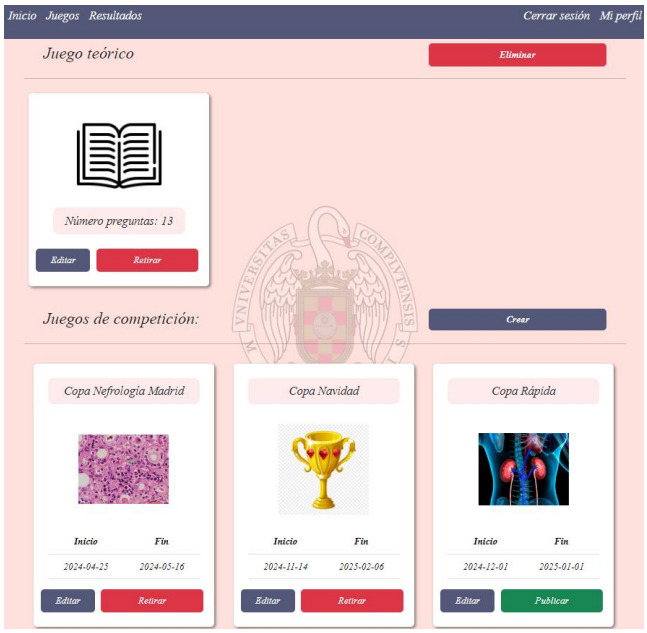

Figura 40 Funcionalidad listar juegos publicados

```
<% if (!juego.publicado) { %>
  <button onclick="abrirModalPublicarJuego('<%= juego.id %>')" class="btn btn-success" style="width: 170px;">Publicar</button>
<% } else { %>
  <button onclick="abrirModalRetirarJuego('<%= juego.id %>')" class="btn btn-danger" style="width: 170px;">Retirar</button>
<% } %>
```

Figura 41 Funcionalidad Listar juegos publicados

6.2.5 Listar juegos sin publicar

La visualización de los juegos sin publicar se completa de igual manera que en el apartado anterior, distinguiéndose en la lista general, generada dinámicamente de la vista de juegos, por el estado de su botón de publicar o retirar dentro de la tarjeta que representa cada juego. En este caso las tarjetas de juegos sin publicar son las que contienen el botón de la Figura 42.

Figura 42 Botón de publicar un juego

6.2.6 Ver juego teórico

Como la información que define el juego teórico solo es el número de preguntas que contendrá y su estado, la información es presentada directamente en la tarjeta BootStrap que se puede observar en la Figura 43. Esto se consigue en el renderizado de la vista EJS como se observa en el código de la Figura 44.

Figura 43 Funcionalidad ver juego teórico

```
<div class="seccionTarjetas">
  <% if (jteorico) { %>
    <div class="card h-100 p-4 m-2 juegoTeoricoCard">
      <!-- Card para juego teorico -->
      <form method="GET" action="/juegos/juegoTeorico" class="juegoTeoricoForm">
        <img src="/images/libro.png" class="card-img-top img" alt="Imagen " />
        <div class="card-body  text-center">
          <h5 class="card-title">
            Número preguntas: <%= jteorico.numPreguntas %>
          </h5>
          <input type="hidden" name="juegoTeoricoId" id="juegoTeoricoId" value="<%= jteoricoId %>" />
        </div>
      </form>
      <div class="row">
        <!-- Botón para abrir el modal de Modificar -->
        <button type="button mb-3" style="width: 90px; margin-right: 10px" class="btn btn-primary"
          data-bs-toggle="modal" data-bs-target="#modalModificar">
          Editar
        </button>
        <!-- Botón para abrir el modal de Publicar, visible si jteorico.publicado es false -->
        <% if (!jteorico.publicado) { %>
          <button type="button" style="width: 170px" class="btn btn-success" id="btnPublicar"
            data-bs-toggle="modal" data-bs-target="#modalPublicar">
            Publicar
          </button>
          <% } %>
            <!-- Botón para Retirar, visible si jteorico.publicado es true -->
            <% if (jteorico.publicado) { %>
              <button type="button" style="width: 170px" class="btn btn-danger" id="btnRetirar"
                data-bs-toggle="modal" data-bs-target="#modalRetirar">
                Retirar
              </button>
              <% } %>
      </div>
    </div>
    <% } %>
</div>
```

Figura 44 Código funcionalidad ver juego teórico

6.2.7 Publicar juego

La publicación de un juego se consigue mediante la pulsación del botón publicar de su tarjeta, que dispara el método definido anteriormente en la imagen de la Figura 41. Al realizarse esta acción, se ejecuta la función de la añadir el id del juego al botón de confirmación y recuperarlo cuando se pulsa para ejecutar el método de la Figura 47 y posteriormente ejecutándose el *endpoint* de la Figura 48. De una manera similar se realiza para el juego teórico.

```
function abrirModalPublicarJuego(juegoId) {
    $('#btnConfirmarPublicacionCompeticion').data('juegoId', juegoId);
    $('#modalPublicarComp').modal('show');
}
```

Figura 45 Método onclick del botón publicar juego de competición

```
//Confirmación de publicación
$('#btnConfirmarPublicacionCompeticion').click(function () {
    var juegoId = $(this).data('juegoId'); //Recuperamos el juegoId del atributo data-
    publicarOretirarJuegoComp(juegoId, true); //true para publicar
});
```

Figura 46 Jquery que maneja la pulsación del botón de confirmación de publicación

```
function publicarOretirarJuegoComp(juegoId, publicar) {
    $.ajax({
        url: `/juegos/statusCompeticion`,
        method: 'POST',
        contentType: 'application/json',
        data: JSON.stringify({ juegoId: juegoId, publicado: publicar }),
        success: function (response) {
            if (publicar) {
                $('#mensajeExitooerrorPublicarComp').removeClass('text-danger').addClass('text-success').text(response.message);
                $('#btnConfirmarPublicacionCompeticion').addClass('d-none');
            }
            else {
                $('#mensajeExitooerrorRetirarComp').removeClass('text-danger').addClass('text-success').text(response.message);
                $('#btnConfirmarRetirarJuegoComp').addClass('d-none');
            }

            setTimeout(function () {
                window.location.reload();
            }, 2000);
        },
        error: function (xhr) {
            if (publicar) {
                $('#mensajeExitooerrorPublicarComp').addClass('text-danger').text(xhr.responseJSON.message);
            }
            else { $('#mensajeExitooerrorRetirarComp').addClass('text-danger').text(xhr.responseJSON.message); }
            setTimeout(function () {
                window.location.reload();
            }, 2000);
        }
    });
}
```

Figura 47 Código de funcionalidad Ajax de publicar juego de competición

```
router.post('/statusCompeticion', async (req, res) => {
  const { juegoId, publicado } = req.body;
  try {
    const juegoRef = db.collection('juegosCompeticion').doc(juegoId);
    const doc = await juegoRef.get();
    if (!doc.exists) {
      return res.status(404).json({ success: false, message: 'Juego de competición eliminado por otro Admin' });
    }

    await juegoRef.update({ publicado });
    res.status(200).json({ success: true, message: 'Juego de competición actualizado con éxito.' });
  } catch (error) {
    res.status(500).json({ success: false, message: 'Error al modificar el juego de competición.' });
  }
});
```

Figura 48 Código de funcionalidad publicar juego competición

6.2.8 Retirar juego

Para retirar un juego, se sigue la misma lógica del punto anterior, pero ahora para cambiar su estado de publicado a false, para ello se ejecuta la funcionalidad del *onClick* del botón de retirar de la Figura 49 y seguido la Figura 50, que usa el mismo método de la Figura 47 vista en el punto anterior.

```
function abrirModalRetirarJuego(juegoId) {
    $('#btnConfirmarRetirarJuegoComp').data('juegoId', juegoId);
    $('#modalRetirarJuegoComp').modal('show');
}
```

Figura 49 Código de funcionalidad del onclick de retirar juego de competición

```
//Confirmación de retirada
$('#btnConfirmarRetirarJuegoComp').click(function () {
    var juegoId = $(this).data('juegoId'); //Recuperamos el juegoId del atributo data-
    publicarOretirarJuegoComp(juegoId, false); //false para publicar
});
```

Figura 50 Código de funcionalidad Jquery de retirar juego de competición

Figura 51 Cotón que activa la funcionalidad de retirar un juego

6.3 Módulo bloques temáticos

6.3.1 Crear bloque temático

El usuario administrador puede crear un tema con su título e indicando si es de los temas favoritos del MIR, mediante la interfaz de la página web en la vista interna del juego teórico pulsando en crear (Figura 52). Esta acción ejecuta el código necesario hasta llegar al *endpoint* de creación del tema, ilustrado en la imagen de la Figura 53. Si no se pudo crear se manejan los errores correspondientes.

Figura 52 Funcionalidad crear bloque temático

```
router.post('/crearTema', async (req, res) => {
  const { titulo, favoritoMIR } = req.body;
  try {
    let docRef = await db.collection('Temas').add({
      titulo: titulo,
      favoritoMIR: favoritoMIR
    });

    if (docRef.id) {
      res.json({ success: true, message: 'Tema creado con éxito.' });
    } else {
      res.status(500).json({ success: false, message: 'No se pudo crear el tema.' });
    }
  } catch (error) {
    res.status(500).json({ success: false, message: 'Error al crear el tema.', error: error });
  }
});
```

Figura 53 Código funcionalidad crear bloque temático

6.3.2 Modificar bloque temático

Los administradores pueden modificar un tema pulsando el botón de modificar de la tabla (Figura 54). Se procede a cambiar en el modal que se abre el título o modificar la selección de la opción que indica si es un tema favorito MIR. Una vez que se abre el modal se rellena con los datos del tema para facilitar la modificación del

usuario. Si se pulsa en el botón de modificar del modal se ejecuta el código de actualización del *endpoint* que se muestra en la Figura 55.

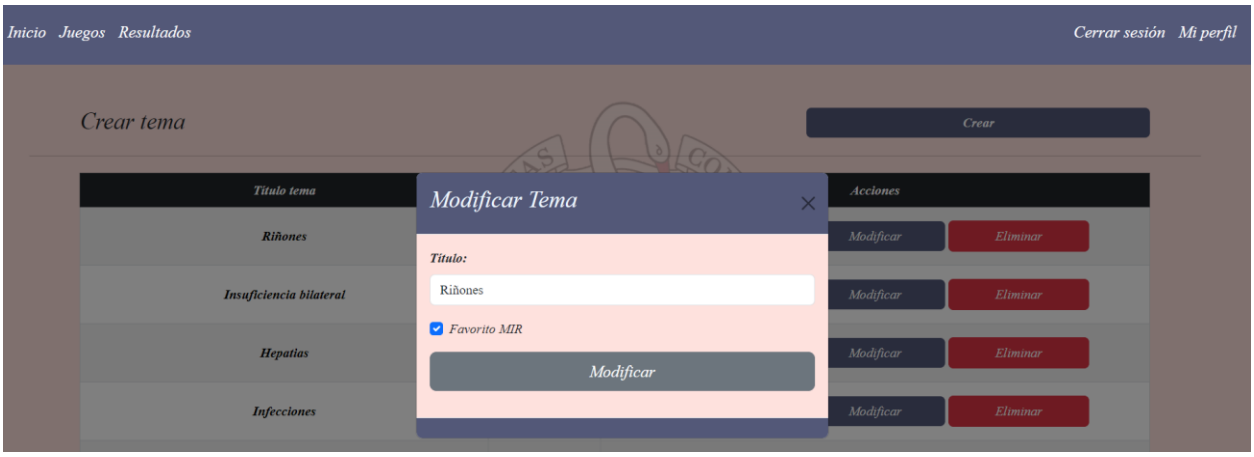

Figura 54 Funcionalidad modificar bloque temático

```
router.post('/actualizarTema', async (req, res) => {
  const { temaId, titulo, favoritoMIR } = req.body;
  try {
    const temaRef = db.collection('Temas').doc(temaId);
    const doc = await temaRef.get();

    if (!doc.exists) {
      return res.status(404).json({ success: false, message: 'Tema eliminado por otro admin' });
    }

    await temaRef.update({
      titulo: titulo,
      favoritoMIR: favoritoMIR
    });
    res.status(200).json({ success: true, message: 'Tema actualizado con éxito.' });
  } catch (error) {
    res.status(500).json({ success: false, message: 'Error al actualizar el tema.' });
  }
});
```

Figura 55 Código funcionalidad modificar bloque temático

6.3.3 Eliminar bloque temático

Los usuarios administradores pueden ejecutar la funcionalidad de eliminar un bloque temático pulsando en el botón *danger* del tema de la tabla de temas, mostrando un modal de confirmación (Figura 56) que, si es aceptado, ejecuta el código

de la Figura 57. Para mantener la integridad de la base de datos es necesario eliminar todas las preguntas asociadas a ese tema e igualmente todas las referencias al tema de los distintos juegos de competición que lo tengan.

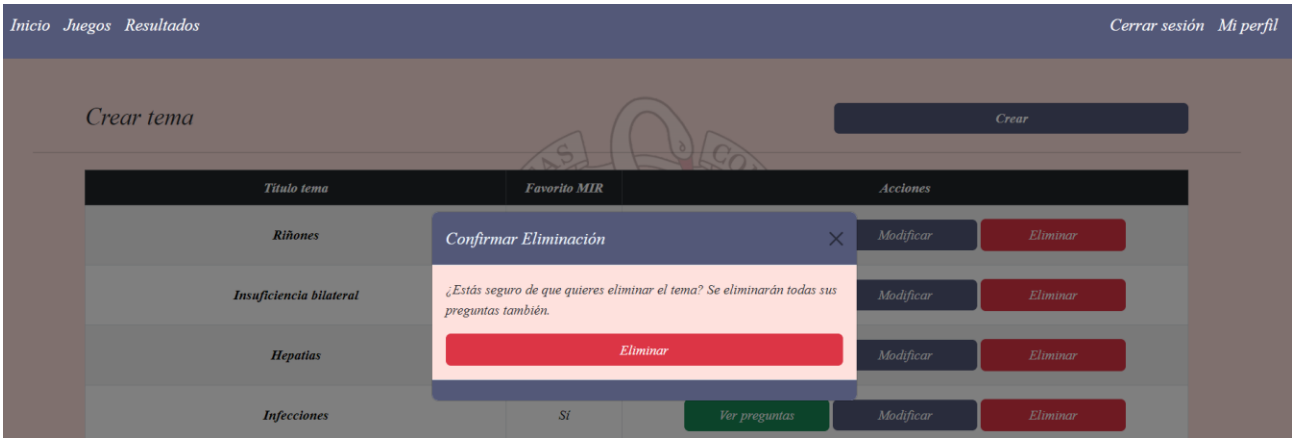

Figura 56 Funcionalidad eliminar bloque temático

```
router.delete('/eliminarTema/:id', async (req, res) => {
  const temaId = req.params.id;
  try {
    //Comienza una transacción batch para eliminar el tema y sus preguntas
    const batch = db.batch();
    const temaRef = db.collection('Temas').doc(temaId);
    //Verificamos si el tema existe antes de intentar eliminarlo
    const tema = await temaRef.get();
    if (!tema.exists) {
      return res.status(404).json({ success: false, message: 'Tema eliminado por otro admin' });
    }
    batch.delete(temaRef);

    //preguntas asociadas al tema
    const preguntasSnapshot = await db.collection('Preguntas').where('temaId', '==', temaId).get();
    if (preguntasSnapshot.empty) {
    } else {
      preguntasSnapshot.forEach(doc => {
        batch.delete(doc.ref);
      });
    }

    //Aplicamos todas las eliminaciones del batch
    await batch.commit();

    //Eliminamos el tema de los juegos de competición
    const juegosSnapshot = await db.collection('juegosCompeticion').where('temas', 'array-contains', temaId).get();
    juegosSnapshot.forEach(async (docJuego) => {
      //Para cada juego encontrado, elimina el temaId del array de temas
      const temasActualizados = docJuego.data().temas.filter(id => id !== temaId);
      await db.collection('juegosCompeticion').doc(docJuego.id).update({
        temas: temasActualizados
      });
    });

    res.status(200).json({ success: true, message: 'Tema, preguntas relacionadas y referencias en juegos eliminadas con éxito' });
  } catch (error) {
```

Figura 57 Código funcionalidad eliminar bloque temático

6.3.4 Crear preguntas de bloque temático

Para poder crear una pregunta de un tema, es necesario estar en la vista de los temas del juego teórico y pulsar sobre alguna fila de la tabla, lo que dispara la apertura de un modal con los tres tipos de formularios posibles. Se selecciona el tipo de pregunta, mostrándose únicamente el formulario correspondiente (Figura 58). Cuando se rellenan los campos requeridos, y se pulsa en crear, se ejecuta el código de recogida de los campos del formulario y se envían al *endpoint* común mostrado en la Figura 59. En este se distingue el tipo de pregunta, se intenta subir el archivo y se almacena su URL en la colección correspondiente junto con los datos de la pregunta.

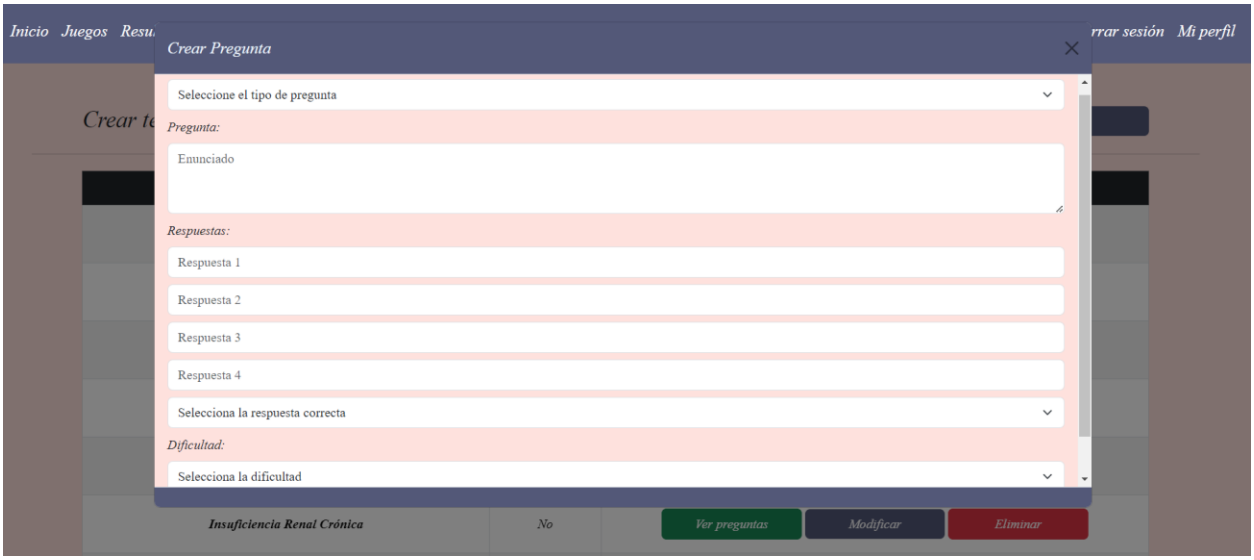

Figura 58 Funcionalidad crear preguntas de bloque temático

```
router.post('/crearPregunta', upload.single('imagen'), async (req, res) => {
  const { dificultad, enunciado, correcta, temaId, tipo } = req.body;
  let respuestas = [];
  if (tipo === "Teórica" || tipo === "Práctica") {
    respuestas = [req.body.respuesta1, req.body.respuesta2, req.body.respuesta3, req.body.respuesta4];
  } else if (tipo === "Negativa") {
    respuestas = [req.body.respuestaNegativa1, req.body.respuestaNegativa2];
  }

  try {
    let imageUrl = null;
    if (req.file) {
      const bucket = storage.bucket();
      const file = bucket.file(req.file.originalname);
      await new Promise((resolve, reject) => {
        const stream = file.createWriteStream({
          metadata: {
            contentType: req.file.mimetype,
          },
        });
        stream.on('error', reject);
        stream.on('finish', async () => {
          await file.makePublic();
          imageUrl = `https://storage.googleapis.com/${bucket.name}/${file.name}`;
          resolve();
        });
        stream.end(req.file.buffer);
      });
    }

    const docRef = await db.collection('Preguntas').add({
      temaId,
      tipo,
      dificultad,
      enunciado,
      respuestas: respuestas.filter(Boolean),
      correcta,
      ...(imageUrl && { imagen: imageUrl }),
    });
    res.status(200).json({ success: true, message: 'Pregunta creada con éxito.' });
  } catch (error) {
    res.status(500).json({ success: false, message: 'Error al crear la pregunta' });
  }
```

Figura 59 Código funcionalidad crear preguntas de bloque temático

6.3.5 Modificar preguntas de bloque temático

Los administradores, tras crear una pregunta, pueden modificarla fácilmente pulsando en el botón de modificar cuando se listan las preguntas de un tema. De esta manera se muestra un modal (Figura 60) con la información de la pregunta cargada y que, al pulsar en enviar, se ejecuta al código del *endpoint* detallado en la Figura 61, muy similar al método del punto anterior, pero esta vez haciendo actualización de una pregunta en concreto. Este *endpoint* es común para los tres flujos de los tres posibles formularios distintos de actualización.

Figura 60 Funcionalidad modificar pregunta de bloque temático

```
router.post('/preguntas/:id', upload.single('imagen'), async (req, res) => {
  const preguntaId = req.params.id;
  const { temaId, tipo, correcta, enunciado, dificultad } = req.body;
  let respuestas = [];
  if (tipo === "Teórica" || tipo === "Práctica") {
    respuestas = [req.body.respuesta1, req.body.respuesta2, req.body.respuesta3, req.body.respuesta4];
  } else if (tipo === "Negativa") {
    respuestas = [req.body.respuestaNegativa1, req.body.respuestaNegativa2];
  }
  let imageUrl = null;
  try {
    const preguntaDoc = await db.collection('Preguntas').doc(preguntaId).get();
    if (!preguntaDoc.exists) {
      return res.status(404).json({ success: false, message: 'Pregunta eliminada por otro admin' });
    }
    if (req.file) {
      const bucket = storage.bucket();
      const file = bucket.file(req.file.originalname);
      await new Promise((resolve, reject) => {
        const blobStream = file.createWriteStream({
          metadata: {
            contentType: req.file.mimetype,
          },
        });
        blobStream.on('error', reject);
        blobStream.on('finish', async () => {
          await file.makePublic();
          imageUrl = `https://storage.googleapis.com/${bucket.name}/${file.name}`;
          resolve();
        });
        blobStream.end(req.file.buffer);
      });
    }

    await db.collection('Preguntas').doc(preguntaId).update({
      temaId,
      tipo,
      dificultad,
      enunciado,
      respuestas: respuestas.filter(Boolean),
      correcta,
      ...(imageUrl && { imagen: imageUrl }),
    });
    res.status(200).json({ success: true, message: 'Pregunta actualizada con éxito.' });
  } catch (error) {
```

Figura 61 Código de funcionalidad modificar pregunta de bloque temático

6.3.6 *Eliminar preguntas de bloque temático*

Para eliminar una pregunta de la lista de preguntas de un tema, hay que pulsar en el botón inferior derecho de la tarjeta que representa la información de la pregunta. Se abre un modal de confirmación como se puede observar en la Figura 62, que, si es confirmado, se ejecuta el código del servidor representado en la imagen de la Figura 63. De esta forma se elimina el documento correspondiente de la colección de preguntas, gestionando las respuestas con sus estados al cliente.

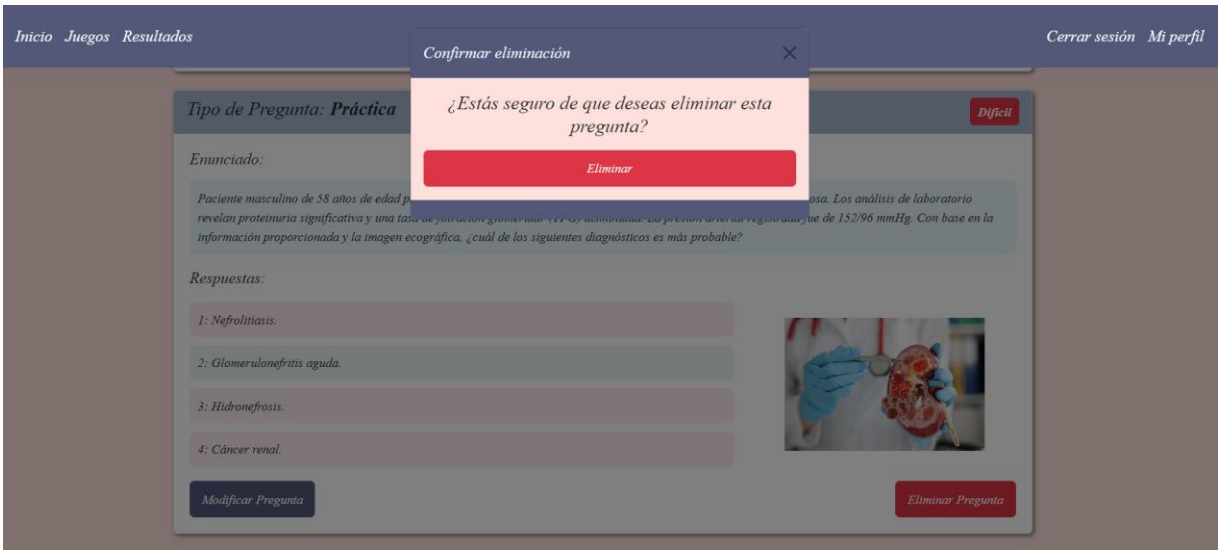

Figura 62 Funcionalidad eliminar preguntas de bloque temático

```
router.delete('/preguntas/:id', async (req, res) => {
  const preguntaId = req.params.id;

  try {
    const docRef = db.collection('Preguntas').doc(preguntaId);
    const doc = await docRef.get();

    if (!doc.exists) {
      return res.status(404).json({ success: false, message: 'La pregunta no existe, probablemente fue eliminada por otro admin' });
    }
    await docRef.delete();
    res.status(200).json({ success: true, message: 'Pregunta eliminada con éxito.' });

  } catch (error) {
    res.status(500).json({ success: false, message: 'Error al eliminar la pregunta' });
  }
});
```

Figura 63 Código de funcionalidad eliminar preguntas de bloque temático

6.3.7 Ver preguntas de bloque temático

Para mostrar la lista de preguntas asociadas a un tema, hay que pulsar en el botón verde de ver preguntas en la fila correspondiente de la tabla de temas. De esta manera, se renderiza una nueva vista rellenada dinámicamente con la información de las distintas preguntas en tarjetas (Figura 64). Cuando se realiza el renderizado se utiliza el *endpoint* de la Figura 65 que recupera toda la información de las preguntas.

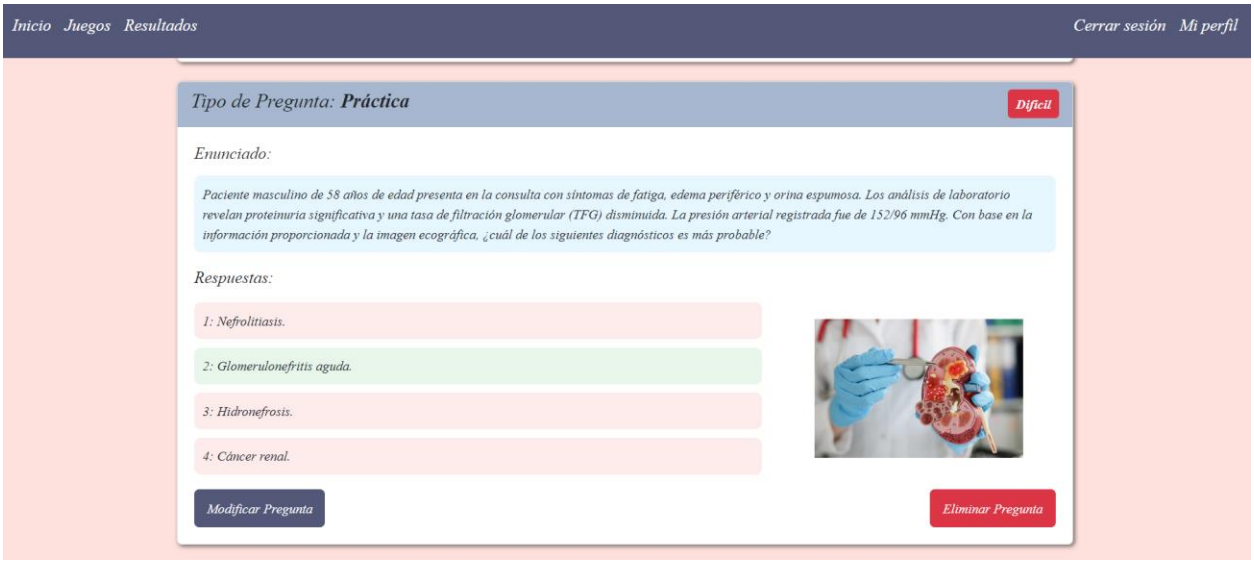

Figura 64 Funcionalidad ver preguntas de bloque temático

```
router.get('/preguntasTema/:id', async (req, res) => {
  const temaId = req.params.id;

  try {
    const preguntasSnapshot = await db.collection('Preguntas')
      .where('temaId', '==', temaId)
      .get();

    const preguntas = preguntasSnapshot.docs.map(doc => {
      return {
        id: doc.id,
        ...doc.data()
      };
    });
    //Renderizamos la vista con las preguntas del tema
    res.render('preguntasTema', { preguntas });
  } catch (error) {
    res.status(500).render('error', { message: 'Error al obtener las preguntas del tema.' });
  }
});
```

Figura 65 Código de funcionalidad ver preguntas de bloque temático

6.4 Módulo juegos de competición

6.4.1 Ver juego competición

A esta funcionalidad se accede pulsando en tarjeta correspondiente del juego de competición de la vista principal de juegos. Se abre una nueva vista de detalles del juego de competición donde se muestra toda su información (Figura 66). Esta funcionalidad hace uso del método del servidor de la Figura 67.

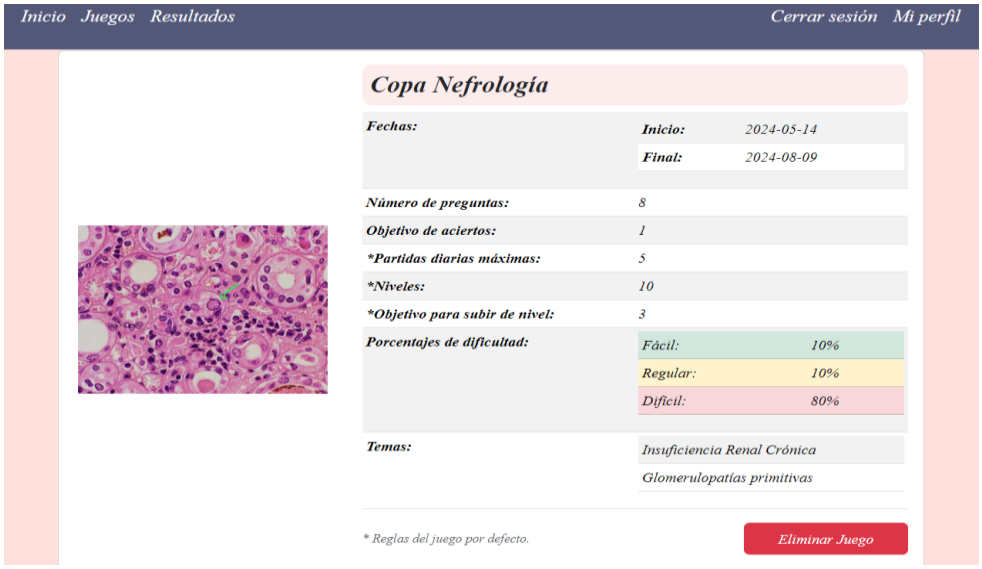

Figura 66 Funcionalidad ver juego de competición

```
router.get('/renderJuegoCompeticion/:idJuegoComp', async (req, res) => {
  const juegoId = req.params.idJuegoComp;
  try {
    const juegoDoc = await db.collection('juegosCompeticion').doc(juegoId).get();
    if (!juegoDoc.exists) {
      return res.status(404).send('El juego no fue encontrado');
    }
    const juego = juegoDoc.data();
    //Promesas para obtener los documentos de temas
    let temasPromises = [];
    juego.temas.forEach(temaId => {
      temasPromises.push(db.collection('Temas').doc(temaId).get());
    });
    //Espera a que todas las promesas se resuelvan
    let temasDocs = await Promise.all(temasPromises);
    let temas = [];
    temasDocs.forEach(doc => {
      if (doc.exists) {
        temas.push(doc.data());
      }
    });
    //Renderiza la vista pasando los datos del juego y los datos completos de los temas
    res.render('detallesJuegoCompetición', { juego: juego, temas: temas, juegoId: juegoId });
  } catch (error) {
    res.status(500).send('Error interno del servidor');
  }
```

Figura 67 Código funcionalidad ver juego de competición

6.4.2 Crear juego de competición

Cualquier usuario administrador puede pulsar en el botón de crear juego de competición de la vista de juegos y rellenar la información requerida del modal que se abre (Figura 68). Esta información debe cumplir que los porcentajes sumen 100, que el número de preguntas del objetivo de aciertos no supere el número de preguntas y que la fecha de fin no sea anterior a la de inicio. Cuando se realiza el envío de este formulario, el método que lo procesa en el servidor es el de la Figura 69, que realiza la subida del archivo imagen y el guardado de los datos en la colección correspondiente.

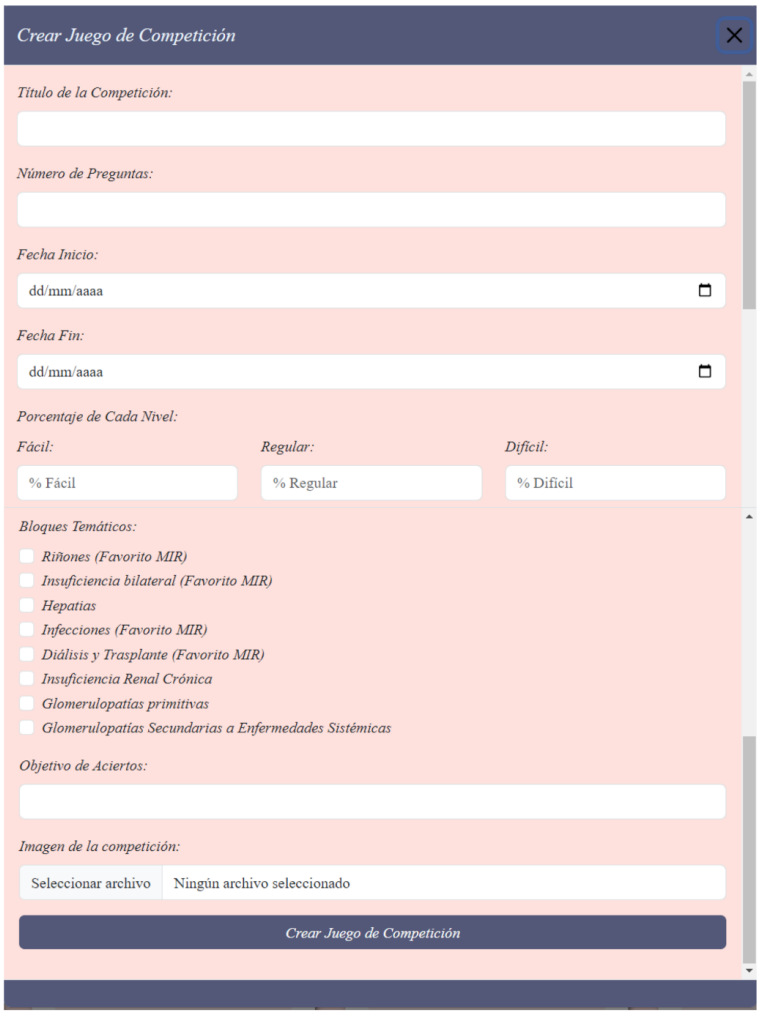

Figura 68 Funcionalidad crear juego de competición

```
router.post('/crearJuegoCompeticion', upload.single('imagen'), async (req, res) => {
  const { tituloCompeticion, numeroPreguntas, fechaInicio, fechaFin, porcentajeFacil,
    porcentajeRegular, porcentajeDificil, objetivoAciertos } = req.body;
  let temasSeleccionados = req.body.temas; //podría ser un array si se seleccionan múltiples temas
  if (!Array.isArray(temasSeleccionados)) {
    temasSeleccionados = [temasSeleccionados];}
  let imageUrl = null;
  if (req.file) {
    const bucket = storage.bucket();
    const file = bucket.file(req.file.originalname);
    try {
      const stream = file.createWriteStream({
        metadata: {
          contentType: req.file.mimetype,
        },
      });
      stream.on('error', (e) => {
        return res.status(500).json({ success: false, message: 'Algo salió mal al subir el archivo' });
      });
      //Se da la orden de terminar el envio de datos por stream, y no se permite escribir más datos
      stream.end(req.file.buffer);
      //Se recive el evento de finalizar, y cuando finaliza continuamos
      await new Promise((resolve) => stream.on('finish', resolve));
      //Hacemos el archivo público y obtener su URL
      await file.makePublic();
      imageUrl = `https://storage.googleapis.com/${bucket.name}/${file.name}`;
    } catch (error) {
      return res.status(500).json({ success: false, message: 'Error al subir la imagen' }); }
  }
  try {
    const docRef = await db.collection('juegosCompeticion').add({
      tituloCompeticion,
      numeroPreguntas: Number(numeroPreguntas),
      fechaInicio,
      fechaFin,
      porcentajes: {
        facil: Number(porcentajeFacil),
        regular: Number(porcentajeRegular),
        dificil: Number(porcentajeDificil),
      },
      objetivoAciertos: Number(objetivoAciertos),
      temas: temasSeleccionados,
      ...(imageUrl && { imagen: imageUrl }), //la imagen solo si existe
      //Datos por defecto
      niveles: 10,
      objetivoPaseParaSubirN: 3,
      partidasDiariasMax: 5,
      publicado: false
    });
    //actualizamos el documento para incluir el ID dentro del mismo
    await db.collection('juegosCompeticion').doc(docRef.id).update({id: docRef.id });
    res.status(200).json({ success: true, message: 'Juego de competición creado con éxito.' });
  } catch (error) {
    res.status(500).json({ success: false, message: 'Error al guardar los datos del juego de competición' }
    );
```

Figura 69 Código funcionalidad crear juego de competición

6.4.3 Modificar juego de competición

De manera similar a la funcionalidad anterior, el usuario administrador puede editar los datos de un juego de competición tras haber sido creado. Estos datos se cargan en el modal que se observa en la Figura 70, y cuando son modificados y guardados, se ejecuta el método del servidor de la Figura 71 que actualiza los campo en la base de datos.

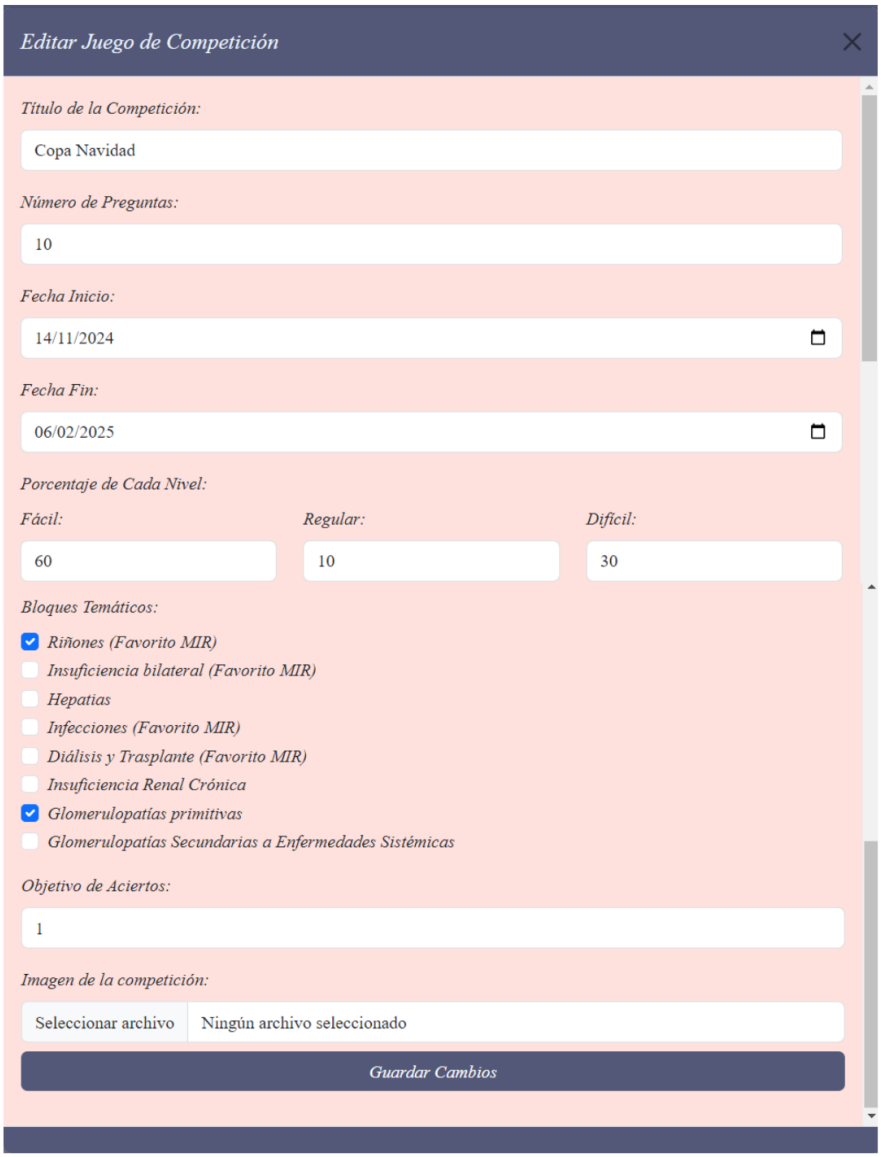

Figura 70 Funcionalidad de modificar un juego de competición

```
router.post('/actualizarJuegoComp', upload.single('imagen'), async (req, res) => {
  const juegoId = req.body.juegoId;
  const { tituloCompeticion, numeroPreguntas, fechaInicio, fechaFin, porcentajeFacil,
    porcentajeRegular, porcentajeDificil, objetivoAciertos } = req.body;
  let temasSeleccionados = req.body.temas;
  if (!Array.isArray(temasSeleccionados)) {
    temasSeleccionados = [temasSeleccionados];
  }
  let imageUrl = null;
  if (req.file) {
    const bucket = storage.bucket();
    const file = bucket.file(req.file.originalname);
    try {
      const stream = file.createWriteStream({
        metadata: {
          contentType: req.file.mimetype,
        },
      });
      stream.on('error', (e) => {
        return res.status(500).json({ success: false, message: 'Algo salió mal al subir el archivo' });
      });
      stream.end(req.file.buffer);
      await new Promise((resolve) => stream.on('finish', resolve));
      await file.makePublic();
      imageUrl = `https://storage.googleapis.com/${bucket.name}/${file.name}`;
    } catch (error) {
      return res.status(500).json({ success: false, message: 'Error al subir la imagen' });
    }
  }
  try {
    //Verificamos si el juego existe antes de actualizarlo
    const juegoRef = db.collection('juegosCompeticion').doc(juegoId);
    const doc = await juegoRef.get();
    if (!doc.exists) {
      return res.status(404).json({ success: false, message: 'Juego de competición eliminado por otro admin' });
    }
    const juegoActualizado = {
      tituloCompeticion,
      numeroPreguntas: Number(numeroPreguntas),
      fechaInicio,
      fechaFin,
      porcentajes: {
        facil: Number(porcentajeFacil),
        regular: Number(porcentajeRegular),
        dificil: Number(porcentajeDificil),
      },
      objetivoAciertos: Number(objetivoAciertos),
      temas: temasSeleccionados,
      ...(imageUrl && { imagen: imageUrl }),
    };
    await juegoRef.update(juegoActualizado);
    res.status(200).json({ success: true, message: 'Juego de competición actualizado con éxito.' });
  } catch (error) {
    res.status(500).json({ success: false, message: 'Error al actualizar los datos del juego de competición' });
```

Figura 71 Código de funcionalidad de modificar un juego de competición

6.4.4 Eliminar juego de competición

La acción de eliminar un juego de competición se realiza en la vista que muestra los detalles de este. Pulsando en el botón inferior derecho de la tarjeta se muestra un modal de confirmación de eliminación (Figura 72). Si es aceptado, se ejecuta el método del servidor mostrado en la Figura 73 para eliminarlo de la base de datos, devolviendo los estados y mensajes correspondientes al cliente.

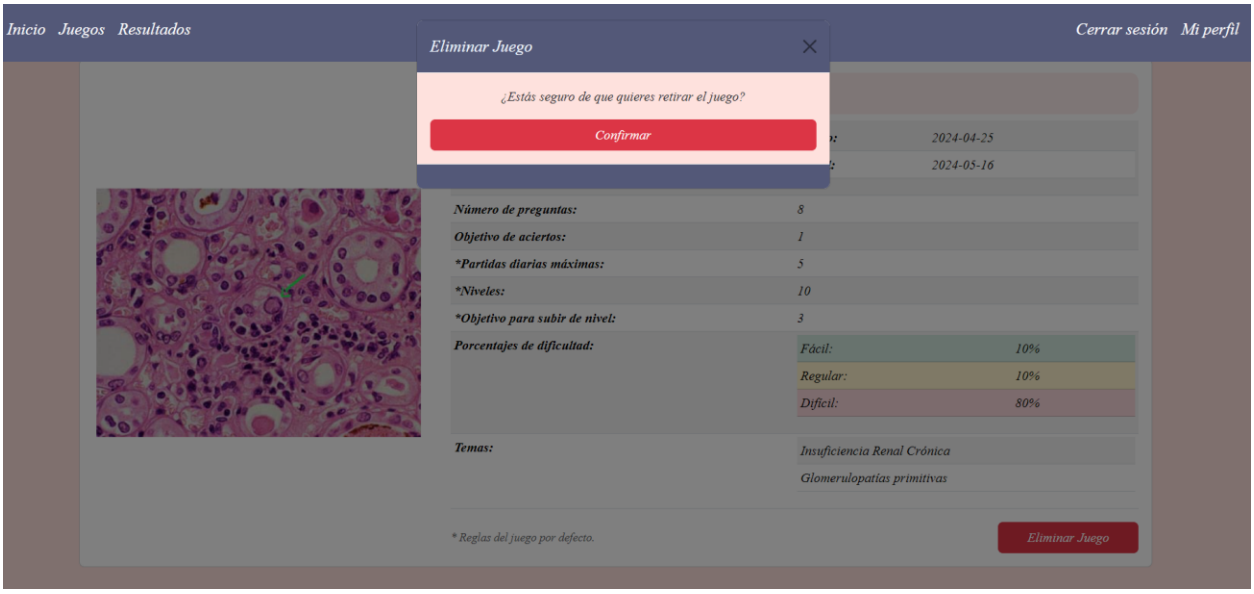

Figura 72 Funcionalidad eliminar juego de competición

```
router.delete('/eliminarJuegoCompeticion/:id', async (req, res) => {
  const juegoId = req.params.id;

  try {
    const juegoRef = db.collection('juegosCompeticion').doc(juegoId);
    const doc = await juegoRef.get();

    if (!doc.exists) {
      return res.status(404).json({ success: false, message: 'Juego de competición ya eliminado por otro admin' });
    }

    //Si el documento existe, se elimina
    await juegoRef.delete();
    res.status(200).json({ success: true, message: 'Juego eliminado con éxito.' });
  } catch (error) {
    res.status(500).json({ success: false, message: 'Error al eliminar el juego de competición.' });
  }
});
```

Figura 73 Código funcionalidad eliminar juego de competición

6.5 Módulo gestión de resultados

6.5.1 Listar juegos activos

Los administradores pueden pulsar la pestaña de resultados y elegir en el selector, mediante la lista de juegos activos que se despliega, cuál es el que más les interesa (Figura 74). Cuando se renderiza esta vista este selector se rellena de manera dinámica con sus opciones, que son parámetro de la vista. La información de la vista es la que se renderiza desde el servidor mediante el método que se observa en la Figura 75, que devuelve la información de los juegos que nos interesa mostrar.

Figura 74 Funcionalidad listar juegos activos

```
router.get('/', async function (req, res, next) {
  try {
    const fechaAct = new Date(); //Fecha actual
    const juegosActivos = [];

    //Traemos todos los juegos de competición
    const snapshotJuegosCompeticion = await db.collection('juegosCompeticion').get();
    snapshotJuegosCompeticion.forEach(doc => {
      const fechaInicio = new Date(doc.data().fechaInicio);
      const fechaFin = new Date(doc.data().fechaFin);
      //Comparamos las fechas para determinar si el juego está activo
      if (fechaInicio <= fechaAct && fechaFin >= fechaAct) {
        juegosActivos.push({
          id: doc.id,
          titulo: doc.data().tituloCompeticion
        });
      }
    });
    res.render('resultados', { juegosActivos: juegosActivos });
  } catch (error) {
    res.render('resultados', { juegosActivos: [] });
  }
});
```

Figura 75 Código funcionalidad listar juegos activos

6.5.2 Ver resultados de juegos de competición activos

Tras seleccionar un juego activo, se muestra una gráfica con los resultados como se observa en la Figura 76. Como se explicó en el capítulo de arquitectura de la parte web, se produce un renderizado en el cliente llamando a la API de Google Charts, la cual renderiza el gráfico específico, si se le pasa la información al método de dibujar la información formateada como la documentación lo requiere. Si el selector del juego cambia se vuelve a dibujar la gráfica con los nuevos datos obtenidos del juego como se observa en la Figura 77. El método de dibujar carga los datos filtrados y formateados con las opciones necesarias en la gráfica, como se observa en la Figura 78.

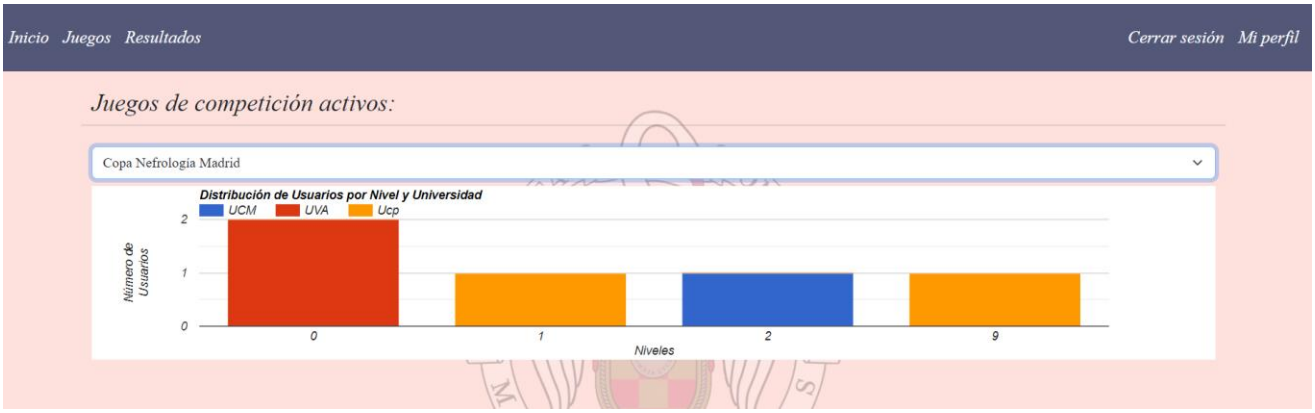

Figura 76 Funcionalidad ver resultados de juego de competición activo

```
router.get('/cargarDatosGrafica/:juegoId', async (req, res) => {
  const {juegoId} = req.params;
  if (!juegoId) {
      return res.status(400).json({ error: 'ID del juego necesario' });
  }

  const datosJuego = [];
  try {
      const usuariosSnapshot = await db.collection('usuariosMOVIL').get();
      for (const doc of usuariosSnapshot.docs) {
          const usuario = doc.data();
          const intentosRef = db.collection('usuariosMOVIL').doc(doc.id).collection('intentosCompeticion');
          const intentoDoc = await intentosRef.doc(juegoId).get();

          if (intentoDoc.exists) {
              datosJuego.push({
                  correoUsuario: usuario.correo,
                  nivelCompeticion: intentoDoc.data().NivelCompeticion,
                  universidad: usuario.universidad
              });
          }
      }

      res.json(datosJuego);
  } catch (error) {
      res.status(500).json({ error: 'Error interno del servidor' });
  }
});
```

Figura 77 Código de funcionalidad ver resultados de juego de competición activo

```
google.charts.load('current', { 'packages': ['corechart'] });

function drawChart(rawData) {
    let dataArray = [['Nivel']];

    if (rawData && rawData.length) {
        dataArray = processStackedChartData(rawData);
    }

    //Convertimos el array de js a DataTable de Google Charts
    var data = google.visualization.arrayToDataTable(dataArray);

    var options = {
        title: 'Distribución de Usuarios por Nivel y Universidad',
        isStacked: true,
        legend: { position: 'top', maxLines: 3 },
        bar: { groupWidth: '75%' },
        hAxis: {
            title: 'Niveles'
        },
        vAxis: {
            title: 'Número de Usuarios'
        }
    };

    const chart = new google.visualization.ColumnChart(document.getElementById("chart_div"));
    chart.draw(data, options);
}
```

Figura 78 Código de la gráfica de funcionalidad ver resultados de juego de competición activo

6.6 Módulo jugar

6.6.1 Listar juegos activos y por iniciar

Los usuarios móviles pueden acceder a la sección de juegos para poder visualizar los juegos publicados disponibles. Aparece el juego teórico de practicar y las competiciones, entre ellas, las que se pueden iniciar y las que aún están por comenzar (Figura 79). Esta acción ejecuta el código del fragmento de juegos, como se observa en la Figura 80, añadiendo la tarjeta del juego teórico en la parte superior y debajo la lista de juegos de competición mediante un *recyclerView* y su adaptador. La información de las tarjetas de las competiciones se añade mediante el *holder* del adaptador de la lista, como se puede ver en la Figura 81.

Figura 79 Funcionalidad listar juegos activos y por iniciar

```
override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
    super.onViewCreated(view, savedInstanceState)


    cardJuegoTeorico = view.findViewById(R.id.cardJuegoTeorico)

    recyclerView = view.findViewById(R.id.recyclerViewJuegos)
    recyclerView.layoutManager = LinearLayoutManager(context)
    juegosAdapter = JuegosAdapter(listOf(), interfazDejuegosFragment: this)
    recyclerView.adapter = juegosAdapter
```

Figura 80 Código fragmento del listado de juegos activos y por iniciar

```
class JuegoViewHolder(itemView: View, private val interfazDejuegosFragment: ICompeticionClickInterface)
    : RecyclerView.ViewHolder(itemView) {
    fun bind(juego: JuegoCompeticion) {
        //bindeo los datos del juego a los views
        itemView.findViewById<TextView>(R.id.tituloCompeticion).text = juego.tituloCompeticion
        itemView.findViewById<TextView>(R.id.numeroPreguntasCompeticion).text =
            "Número de preguntas: ${juego.numeroPreguntas}"
        itemView.findViewById<TextView>(R.id.fechainiPreguntasCompeticion).text =
            "Desde: ${juego.fechaInicio}"
        itemView.findViewById<TextView>(R.id.fechafinPreguntasCompeticion).text =
            "Hasta: ${juego.fechaFin}"

        val imageView = itemView.findViewById<ImageView>(R.id.imageCompeticion)
        Glide.with(itemView.context) RequestManager
            .load(juego.imagen) RequestBuilder<Drawable!>
            .into(imageView)

        itemView.setOnClickListener { it: View!
            interfazDejuegosFragment.onJuegoCOMPClick(juego)
        }


    }
}
}
```

Figura 81Código de la clase holder del adaptador de juegos de competición

6.6.2 Jugar a juego teórico

Los usuarios estudiantes pueden jugar al juego teórico, mostrado como primera tarjeta de color rosa. Una vez pulsado se lleva al usuario a la pantalla de configuración del juego teórico, donde tras pulsar en las opciones que quiere que definan su test de aprendizaje, se lleva al usuario a la ventana del test. Este proceso se puede observar en la Figura 82. En la acción de configurar el juego e iniciar el test, se realiza una navegación con argumentos seguros como se observa en la Figura 83. Además, la vista del test, común a ambos tipos de juegos, determina el juego teórico por el valor por defecto del parámetro del id del juego de competición como se ve en la Figura 84.

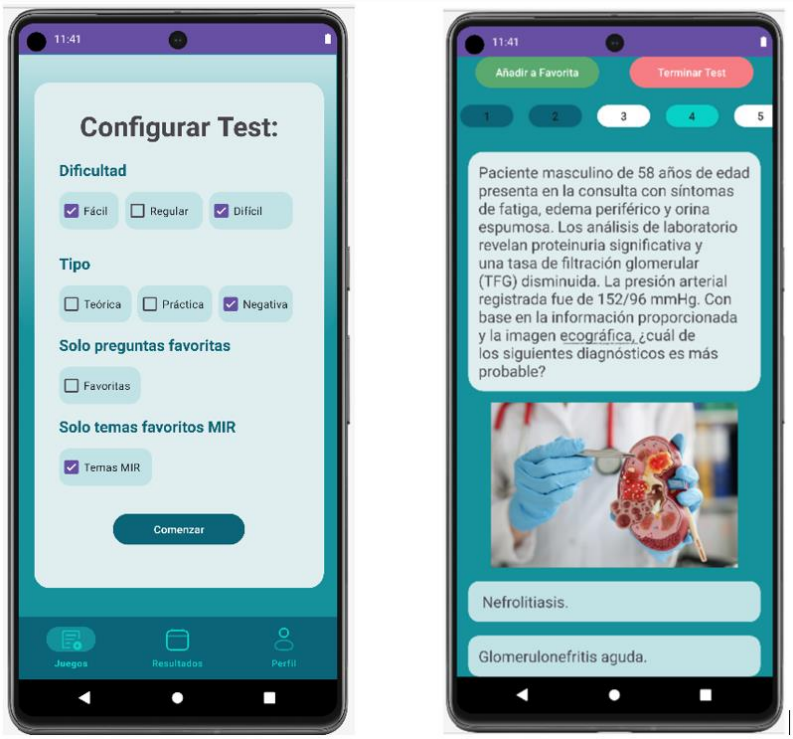

Figura 82 Funcionalidad jugar juego teórico

```
view.findViewById<Button>(R.id.btnIniciarTest).setOnClickListener { it: View!
    if (isOnline(requireContext())) {
        val dificultadFacil = checkBoxFacil.isChecked
        val dificultadRegular = checkBoxRegular.isChecked
        val dificultadDificil = checkBoxDificil.isChecked
        val preguntaTeorica = checkBoxTeorica.isChecked
        val preguntaPractica = checkBoxPractica.isChecked
        val preguntaNegativa = checkBoxNegativa.isChecked
        val soloPreguntasFavoritas = checkBoxPFavoritas.isChecked
        val soloTemasFavoritosMIR = checkBoxFavoritosMIR.isChecked

        if ((dificultadFacil || dificultadRegular || dificultadDificil) &&
            (preguntaTeorica || preguntaPractica || preguntaNegativa)
        ) {

            val action =
                ConfigTeoricoFragmentDirections.actionConfigTeoricoFragmentToTestFragment(
                    dificultadFacilIdArg = dificultadFacil,
                    dificultadRegularIdArg = dificultadRegular,
                    dificultadDificilIdArg = dificultadDificil,
                    preguntaTeoricaIdArg = preguntaTeorica,
                    preguntaPracticaIdArg = preguntaPractica,
                    preguntaNegativaIdArg = preguntaNegativa,
                    PreguntasSoloNegIdArg = soloPreguntasFavoritas,
                    SoloBloquesFavsIdArg = soloTemasFavoritosMIR,

                    )
            findNavController().navigate(action)
        } else mostrarDialogoConfiguracionNoCumpleMinimo()
```

Figura 83 Código funcionalidad configuración de jugar juego teórico

```
val llega =args.idCompeticion
val finishButton = view.findViewById<Button>(R.id.finish_test_button)
if (llega== "EsTeorico") {
    val dificultadFacil = args.dificultadFacilIdArg
    val dificultadRegular = args.dificultadRegularIdArg
    val dificultadDificil = args.dificultadDificilIdArg
    val preguntaTeorica = args.preguntaTeoricaIdArg
    val preguntaPractica = args.preguntaPracticaIdArg
    val preguntaNegativa = args.preguntaNegativaIdArg
    val soloPreguntasFavoritas = args.PreguntasSoloNegIdArg
    val soloBloquesFavoritosMIR = args.SoloBloquesFavsIdArg

    viewModel.cargarPreguntasPorConfiguracion(dificultadFacil,dificultadRegular,dificultadDificil,
        preguntaTeorica,preguntaPractica,preguntaNegativa,soloPreguntasFavoritas,soloBloquesFavoritosMIR)
```

Figura 84 Código del fragmento del test caso juego teórico

6.6.3 Jugar competición

Los usuarios móviles estudiantes pueden intentar jugar a un juego de competición de los visualizados (Figura 85). Cuando se realiza esta acción es probable que aparezca algún aviso como, aún no ha empezado, has superado el límite diario de pases de esta competición, la competición es probable que haya sido eliminada por un administrador, el juego está retirado temporalmente o no se tiene la sesión iniciada. Si se puede realizar el test, la vista es la misma que el juego de competición cambiando su funcionalidad, caso en el que el fragmento recibe el id del juego de competición (Figura 86).

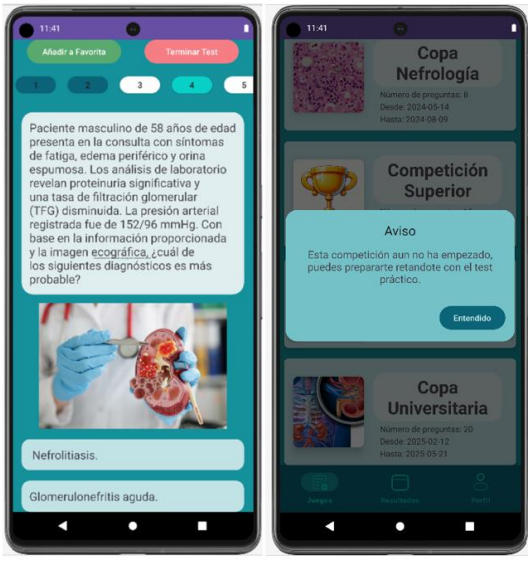

Figura 85 Funcionalidad jugar juego competición

```
} else { // Es competicion
    val JuegoCompID = args.idCompeticion
    //metodo de mi viewmodel de cargar la lista de preguntas del viewmodel segun filtrado
    if (JuegoCompID != null) {
        viewModel.cargarPreguntasPorCompeticion(JuegoCompID)
    }

    finishButton.setOnClickListener { it: View!
        if (isOnline(requireContext())) {
            val puntuacion = viewModel.calcularPuntuacion()
            vistaTestCorregido = true
            //se fuerza un onclik en el boton de la primera pregunta
            firstQuestionButton.performClick()
            finishButton.visibility = View.GONE

            if (JuegoCompID != null) {
                viewModel.obtenerMinAciertosParaSubirNivel(JuegoCompID) { minAciertos ->
                    val incrementa = puntuacion > minAciertos
                    viewModel.registrarIntentoCompeticion(incrementa, JuegoCompID)
                    mostrarDialogoPuntuacionComp(puntuacion, incrementa)
                }
            }

        }else noHayConexion()
    }
}
```

Figura 86 Código funcionalidad fragmento del test, caso juego competición

6.6.4 Ver mis resultados

La funcionalidad de visualizar los resultados de los distintos juegos se realiza en la vista de resultados que se observa en la Figura 87. Esta acción de visualizado ejecuta, en primer lugar, el código de la librería de gráficos añadida (PhilJay:MPAndroidChart), añadiendo al gráfico las puntuaciones de los intentos del juego teórico (Figura 88). Seguido, se observa la lista de tarjetas de los intentos de competición realizados por el usuario y sus parámetros actuales, esto se consigue con un adaptador para tarjetas (Figura 89).

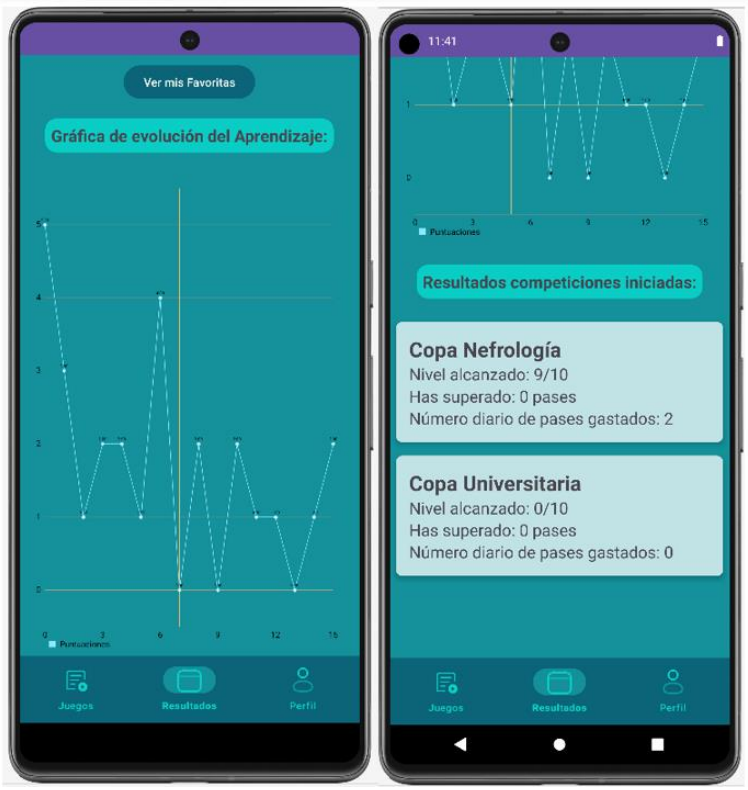

Figura 87 Funcionalidad ver mis resultados

```
private fun actualizarElGrafico(intentos: List<IntentosTeorico>) {
    val entries = ArrayList<Entry>()
    intentos.forEachIndexed { index, intento ->
        intento.puntuacion?.let { Entry(index.toFloat(), it.toFloat()) }
            ?.let { entries.add(it) }
    }
    val dataSet = LineDataSet(entries,  label: "Puntuaciones")
    val lineData = LineData(dataSet)
    chart.data = lineData


    chart.getDescription().setEnabled(false);
    chart.getAxisRight().setDrawLabels(false);

    val xAxis = chart.xAxis
    xAxis.position = XAxis.XAxisPosition.BOTTOM //coloca las etiquetas en la parte inferior
    xAxis.setDrawAxisLine(true) //dibuja la línea del eje X
    xAxis.setDrawGridLines(false) //no dibuja las líneas de la cuadrícula para el eje X

    chart.invalidate() //refresca el gráfico para aplicar los cambios

}
```

Figura 88 Código funcionalidad de ver resultados, caso gráfica intento teórico

```
class ResultadosCompeticionAdapter(var competiciones: List<ResultadosCompeticion>) :
    RecyclerView.Adapter<ResultadosCompeticionAdapter.ResultadosCompeticionViewHolder>() {

    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): ResultadosCompeticionViewHolder {
        val view = LayoutInflater.from(parent.context).inflate(R.layout.card_resultados_competicion, parent,  attachToRoot: false)
        return ResultadosCompeticionViewHolder(view)
    }

    override fun onBindViewHolder(holder: ResultadosCompeticionViewHolder, position: Int) {
        val intento = competiciones[position]
        holder.bind(intento)
    }

    override fun getItemCount() = competiciones.size

    class ResultadosCompeticionViewHolder(itemView: View) : RecyclerView.ViewHolder(itemView) {
        private val tvTituloLaCompeticion: TextView = itemView.findViewById(R.id.tvTituloLaCompeticion)
        private val tvNivelCompeticion: TextView = itemView.findViewById(R.id.tvNivelCompeticion)
        private val tvNivelPase: TextView = itemView.findViewById(R.id.tvNivelPase)
        private val tvNumeroIntento: TextView = itemView.findViewById(R.id.tvNumeroIntento)

        fun bind(competicion: ResultadosCompeticion) {
            tvTituloLaCompeticion.text = competicion.tituloCompeticion
            tvNivelCompeticion.text = "Nivel alcanzado: ${competicion.NivelCompeticion}/10"
            tvNivelPase.text = "Has superado: ${competicion.NivelPase} pases"
            tvNumeroIntento.text = "Número diario de pases gastados: ${competicion.NumeroDeIntento}"
        }
    }
```

Figura 89 Código funcionalidad ver resultados, caso adaptador de tarjetas de intentos de competición

6.6.5 Ver favoritas

Cualquier usuario móvil registrado que haya intentado realizar un juego y haya marcado como favorita alguna pregunta, puede visualizarlas en la sección de resultados pulsando en el botón verde de ver favoritas. De esta manera se lleva al usuario al fragmento donde se encuentra el *recyclerView* de sus preguntas (Figura 90). En la imagen de la Figura 91, podemos ver parte del flujo del código que se ejecuta para recuperar la información de las preguntas favoritas del usuario.

Figura 90 Funcionalidad ver mis favoritas

```
fun obtenerMisPreguntasFavoritasIds(onSuccess: (List<String>) -> Unit, onError: (Exception) -> Unit) {
    val usuarioActual = auth.currentUser
    if (usuarioActual != null && usuarioActual.email != null) {
        db.collection( collectionPath: "usuariosMOVIL") CollectionReference
            .whereEqualTo( field: "correo", usuarioActual.email) Query
            .limit(1)
            .get() Task<QuerySnapshot!>
            .addOnSuccessListener { snapshot ->
                if (!snapshot.isEmpty) {
                    val document = snapshot.documents[0]
                    val preguntasFavIds = document.get("preguntasFAV") as? List<String> ?: listOf()
                    onSuccess(preguntasFavIds)
                } else {
                    onError(Exception("No se encontró el documento del usuario con ese correo"))
                }
            }
            .addOnFailureListener { exception ->
                onError(Exception("Error al acceder a los datos "))
            }
    } else {
        onError(Exception("No hay usuario autenticado o el usuario no tiene email"))
    }
}
```

Figura 91 Código funcionalidad obtener preguntas favoritas del modelo de resultados

Capítulo 7 - Conclusiones y trabajo futuro

7.1 Conclusiones

En este proyecto, se ha desarrollado un sistema compuesto por dos aplicaciones en tecnologías distintas, con el objetivo de satisfacer las necesidades de las academias de estudiantes de oposición MIR.

En la aplicación móvil, se posibilita la autoevaluación de los estudiantes realizando test, similares a los del examen de oposición con los mismos tipos de preguntas. Se ha introducido gamificación para fomentar el aprendizaje y uso de la aplicación móvil, además de la visualización de los resultados para ver la evolución individual.

Por otro lado, se permite la administración de la aplicación móvil mediante la web, ofreciendo la posibilidad a los profesores de las academias de crear los contenidos de las evaluaciones con sus casos de estudio en forma de preguntas.

Finalmente, se ha comprendido la importancia de una buena planificación en la arquitectura del sistema, permitiendo al mismo disponer de distintas tecnologías trabajando en común. Además, en el desarrollo móvil se ha dado importancia a utilizar las recomendaciones actuales de Google para la creación de aplicaciones seguras y robustas, desacoplando el acceso a los datos de las vistas y manejando la comunicación entre los fragmentos con argumentos seguros.

Enlace a los ficheros:

https://drive.google.com/drive/folders/1GMyVHdL1ym34nj2p5ettnPp6muCivhp0?usp=sharing

7.2 Trabajo futuro

A pesar de haber obtenido un resultado de sistema muy parecido al imaginado inicialmente con la especificación de requisitos, se han encontrado una serie de puntos a mejorar en futuras iteraciones.

- Seleccionar temas específicos en la configuración del usuario antes de comenzar a autoevaluarse con el juego teórico.
- Optimización de filtrados a nivel de consulta en el servidor y no en cliente.
- Validaciones de contraseñas y campos de formularios en el servidor, no solo validaciones en el cliente con las clases de Bootstrap.
- Caso de uso no implementado de visualización de resultados del juego teórico para los administradores web.
- Aumentar la información proporcionada en los gráficos.
- Mejorar la interfaz gráfica.
- Posibilidad de aplicar algún modelo de *machine learning* al formar las evaluaciones personalizadas de los estudiantes. Las preguntas se seleccionarían en función de la evolución y punto actual del usuario en la curva de su aprendizaje. Esto podría conseguirse con modelos de aprendizaje automático por refuerzo como el *Deep Q-learning*.
- Añadir búsquedas y filtrados en los listados de los juegos y preguntas.
- Añadir la posición del usuario en un ranking de usuarios global en el resultado de cada juego de competición. De esta manera se trasmitiría mejor al usuario la sensación de estar jugando en una competición.

Chapter 7 - Conclusions and future work

Conclusions of the work and future lines of work.

7.1 Conclusions

In this project, a system consisting of two applications in different technologies has been developed with the goal of addressing the needs of MIR exam preparation academies.

The mobile application enables student self-assessment by conducting tests similar to those in the competitive exam, featuring the same types of questions. Gamification has been introduced to encourage learning and the use of the mobile application, as well as the visualization of results to monitor individual progress.

On the other hand, the mobile application can be administered through the web, offering the possibility for teachers at the academies to create the content of the assessments, with their case studies in the form of questions.

Finally, the importance of good planning in the system's architecture was understood, allowing for the use of different technologies working together. Additionally, in mobile development, importance was given to using Google's recommendations for creating secure and robust applications, decoupling data access from views and managing communication between fragments with safe arguments.

7.2 Future Work

Despite having achieved a system result very similar to what was initially imagined with the requirements specification, several areas for improvement have been identified for future iterations:

- Selecting specific topics in the user settings before beginning to self-assess with the theoretical game.
- Optimization of filtering at the server level rather than on the client.
- Server-side password and form field validations, not just client-side validations with Bootstrap classes.

- Unimplemented use case of displaying results of the theoretical game for web administrators.
- Increasing the information provided in the charts.
- Improving the graphical interface.
- The possibility of applying a machine learning model to form personalized evaluations for students. Questions would be selected based on the user's current point and progression on their learning curve. This could be achieved with reinforcement learning models like Deep Q-learning.
- Adding searches and filters to the listings of games and questions.
- Adding the user's position in a global user ranking to the results of each competitive game, which would better convey to the user the sensation of participating in a competition.

BIBLIOGRAFÍA

[1] «El Español,» [En línea]. Available: https://www.elespanol.com/invertia/observatorios/sanidad/20240119/mir-gana-aspirantes-presentaran-medicos/826167430_0.html.

[2] «Academia MIR,» [En línea]. Available: https://academiamir.com/cuantas-horas-de-estudio-necesito-para-el-mir/.

[3] «Mirial articulos,» [En línea]. Available: https://mirial.es/blog/apuntes-medicina/141-escoger-academia-para-el-mir.

[4] «MIR Asturias,» [En línea]. Available: https://www.curso-mir.com/.

[5] «Academia AMIR,» [En línea]. Available: https://academiamir.com/.

[6] «Academia ProMIR,» [En línea]. Available: https://www.medicapanamericana.com/es/promir/inicio.

[7] «Mirial app,» [En línea]. Available: https://mirial.es/.

[8] «Prepmir app,» [En línea]. Available: https://www.prepmir.es/.

[9] «Visual Studio Code,» [En línea]. Available: https://code.visualstudio.com/.

[10] «Nodejs,» [En línea]. Available: https://nodejs.org/en.

[11] «Express js,» [En línea]. Available: https://expressjs.com/es/.

[12] «JavaScript,» [En línea]. Available: https://es.wikipedia.org/wiki/JavaScript.

[13] «Jquery,» [En línea]. Available: https://jquery.com/.

[14] «BootStrap,» [En línea]. Available: https://getbootstrap.com/.

[15] «HTML,» [En línea]. Available: https://es.wikipedia.org/wiki/HTML5.

[16] «CSS,» [En línea]. Available: https://es.wikipedia.org/wiki/CSS.

[17] «Android Studio,» [En línea]. Available: https://developer.android.com/studio?hl=es-419.

[18] «XML,» [En línea]. Available: https://es.wikipedia.org/wiki/Extensible_Markup_Language.

[19] «Firebase,» [En línea]. Available: https://firebase.google.com/?hl=es.

[20] L. A. Bucki, Word 2013 Bible, John Wiley & Sons, 2013.

[21] CFI, «Cursos de Formación en Informática,» [En línea]. Available: http://cursosinformatica.ucm.es. [Último acceso: 01 06 2019].

[22] «MIR Asturas app,» [En línea]. Available: https://play.google.com/store/apps/details?id=com.cursomir.appmir&hl=es&pli=1.

[23] «Kotlin,» [En línea]. Available: https://kotlinlang.org/.

APÉNDICES

Apéndice A - Especificación de requisitos

En este primer apéndice se presenta la especificación de requisitos inicial de este trabajo, la cual ha servido de guía para su realización. Se detallan los actores y los casos de uso.

Actores

En esta sección se detallan los distintos actores que pueden intervenir en el sistema formado por dos aplicaciones.

- **Usuario móvil no registrado:** este será el primer tipo de usuario que aparece en la aplicación móvil pues es el que aún no se ha registrado y solo tiene acceso a visualizar los distintos juegos y registrarse para poder realizar las evaluaciones.
- **Usuario móvil registrado:** en la aplicación móvil una vez registrado el usuario, podrá tener acceso a la funcionalidad completa de la aplicación, esto implica tener la sesión iniciada.
- **Administrador web:** representa al usuario médico o profesor del centro o academia responsable de la creación de las preguntas del test, utilizando su base de datos de casos de pacientes reales.

Casos de uso

A continuación, se muestra la información relativa a cada caso de uso de los módulos del proyecto. Para cada uno, primero se muestra un diagrama de caso de uso que presenta las funcionalidades de manera resumida y visual. A continuación, se muestran las distintas tablas informativas que detallan las funcionalidades de cada uno de ellos.

Gestión de usuarios

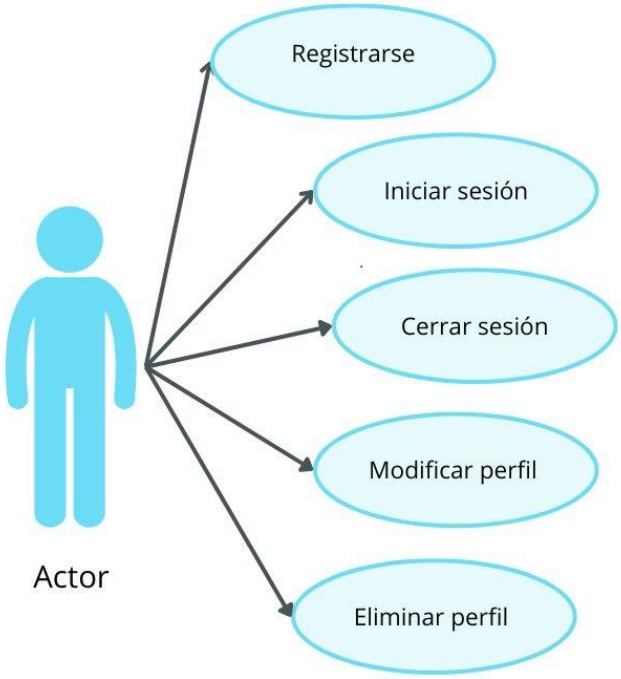

Figura 92 Diagrama de caso de uso Gestión de usuarios

Requisito	**Registrarse**
Id.	2.1.1
Prioridad	Alta
Precondición	El usuario web admin debe estar dado de alta como administrador de la herramienta.
Descripción	El usuario puede registrarse en la aplicación web o móvil. Si es el médico usuario web, su correo debe haber sido introducido en el sistema por los desarrolladores, como administrador, para su registro. Los campos de entrada de Universidad y curso son específicos del usuario móvil no registrado.
Entrada	Nombre, Apellidos, Correo Electrónico, Contraseña, Universidad, Curso

Salida	NA
Secuencia normal	1. El usuario accede a la aplicación, y accede a "Registro" al mostrar la pantalla de inicio. 2. Se le muestra al usuario otra pantalla con los campos necesarios para su registro. 3. El usuario rellena los campos y pulsa en "Crear". 4. El sistema valida los datos. 5. Se muestra la ventana principal de la aplicación.
Postcondición	Se crea la cuenta del usuario en el sistema.
Excepciones	3. Si el correo electrónico no cumple el formato necesario se muestra el mensaje de error correspondiente. 3. Si la contraseña no cumple los requisitos de contraseña se muestra el mensaje de error que corresponda. 3. Si el correo electrónico no existe, se muestra el mensaje de error correspondiente. 3. Si el correo electrónico ya existe en los registros del sistema, se muestra el mensaje de error correspondiente. 3. Si el usuario es web admin y el correo electrónico no está registrado previamente como administrador, se muestra un mensaje para solicitar ser administrador. 3. El usuario pulsa en "Cancelar".
Actores	Admin web, usuario móvil no registrado.

Tabla 2 Requisito: Registrarse

Requisito	**Iniciar sesión**
Id.	2.1.2
Prioridad	Alta
Precondición	El usuario tiene que estar registrado en el sistema.
Descripción	Los usuarios registrados en el sistema pueden iniciar sesión en la aplicación.
Entrada	Correo Electrónico, Contraseña
Salida	-

Secuencia normal	1. El usuario accede a la aplicación, y accede a "Inicio de sesión" al mostrarse la pantalla de inicio. 2. Se le muestra al usuario otra pantalla con los campos necesarios para iniciar sesión. 3. El usuario rellena los campos y pulsa en "Iniciar". 4. El sistema valida los datos. 5. Se muestra la ventana principal de la herramienta.
Postcondición	El usuario ha iniciado sesión
Excepciones	3. Si el correo electrónico no cumple el formato necesario se muestra el mensaje de error correspondiente 3. Si la contraseña no cumple los requisitos de contraseña, se muestra el mensaje de error correspondiente. 3. El usuario pulsa en "Cancelar". 4. Si los datos introducidos no son válidos en el sistema, se muestra el mensaje de error correspondiente.
Actores	Admin web, usuario móvil registrado.

Tabla 3 Requisito: Iniciar sesión

Requisito	**Cerrar sesión**
Id.	2.1.3
Prioridad	Alta
Precondición	El usuario tiene que haber iniciado sesión en la aplicación.
Descripción	Los usuarios que han iniciado sesión pueden cerrar la sesión en la aplicación.
Entrada	-
Salida	-
Secuencia normal	1. El usuario con sesión iniciada en la aplicación pulsa en "Cerrar sesión" en la pantalla principal. 2. El sistema cierra la sesión y muestra la pantalla de inicio.
Postcondición	Se cierra sesión en la aplicación.
Excepciones	-
Comentarios	2. No se puede cerrar sesión.
Actores	Admin web, usuario móvil registrado.

Tabla 4 Requisito: Cerrar sesión

Requisito	**Modificar perfil**
Id.	2.1.4
Prioridad	Baja
Precondición	El usuario tiene que haber iniciado sesión en la aplicación.
Descripción	Los usuarios que han iniciado sesión pueden modificar sus datos personales. Los campos de entrada de Universidad y curso son específicos del usuario móvil registrado.
Entrada	Nombre, apellidos, Universidad, curso.
Salida	-
Secuencia normal	1. El usuario accede a la aplicación, se le muestra la pantalla principal y pulsa en "Modificar usuario". 2. El sistema muestra todos los datos del usuario. 3. El usuario modifica los datos y pulsa en "Modificar". 4. El sistema muestra un mensaje preguntando por la confirmación. 5. El usuario pulsa en "Confirmar". 6. El sistema muestra los datos del usuario que se ha modificado.
Postcondición	Se han modificado los datos del usuario.
Excepciones	3, 5 El usuario pulsa en "Cancelar".
Actores	Admin web, usuario móvil registrado.

Tabla 5 Requisito: Modificar perfil

Requisito	**Eliminar perfil**
Id.	2.1.5
Prioridad	Media
Precondición	El usuario tiene que haber creado una cuenta y tener la sesión activa.
Descripción	El usuario puede eliminar su cuenta creada.
Entrada	Contraseña.
Salida	

Secuencia normal	1. El usuario accede a la aplicación, se le muestra la pantalla principal y pulsa en "Eliminar mi cuenta" 2. El sistema muestra una pantalla de confirmación. 3. El usuario pulsa en "Confirmar y eliminar". 4. El sistema muestra un mensaje de éxito y vuelve a la pantalla principal.
Postcondición	Se ha eliminado la cuenta de usuario.
Excepciones	2. Si la contraseña introducida no es validada en el sistema, se muestra el mensaje de error correspondiente. 3. El usuario pulsa en "Cancelar".
Actores	Usuario móvil registrado.

Tabla 6 Requisito: Eliminar perfil

Juego teórico

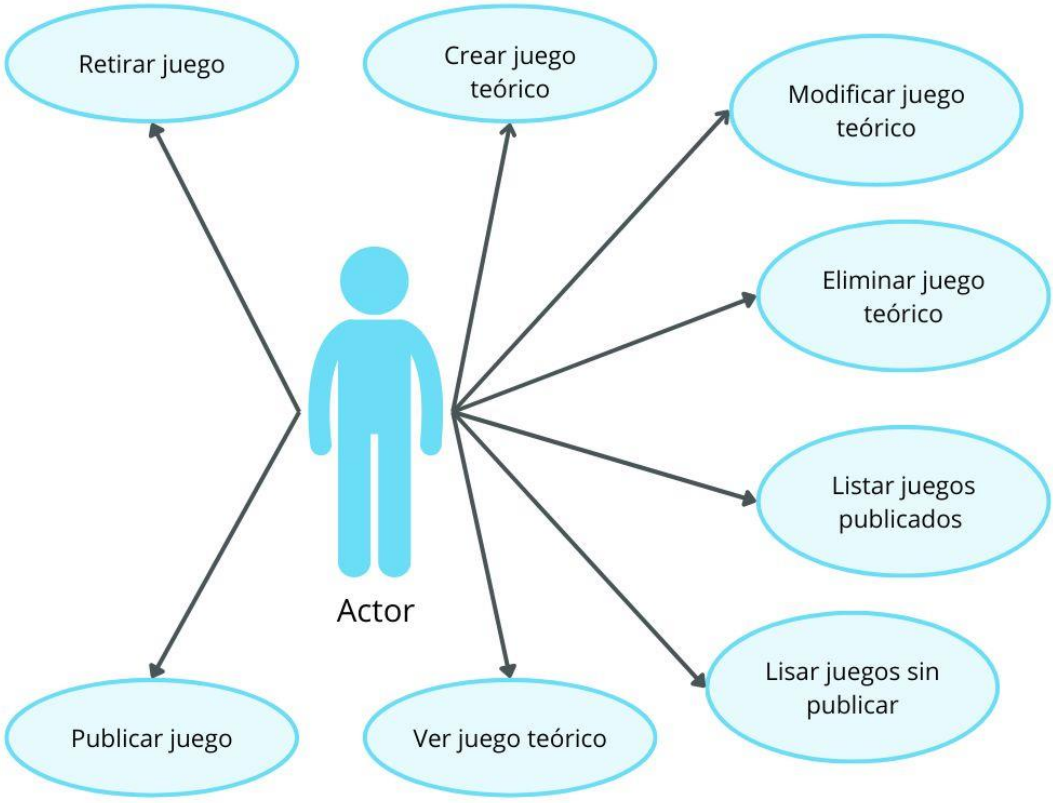

Figura 93 Diagrama de caso de uso Juego teórico

Requisito	**Crear juego teórico**

Id.	2.2.1
Prioridad	Alta
Precondición	El usuario debe estar registrado y con la sesión iniciada.
Descripción	Un usuario admin de la aplicación web puede crear el juego único de tipo teórico sobre Nefrología, introduciendo el número de preguntas por defecto que le saldrán al estudiante cada vez que juegue sin modificar los parámetros de la selección de preguntas.
Entrada	Número de preguntas
Salida	-
Secuencia normal	1. El usuario accede a la aplicación web, se muestra la pantalla principal y pulsa en "Crear juego teórico". 2. La herramienta muestra una pantalla donde introducir el número de preguntas del juego. 3. El usuario introduce el campo requerido por la herramienta y pulsa "aceptar".
Postcondición	El sistema crea en sus registros el juego con sus características.
Excepciones	1 El juego teórico ya ha sido creado por otro admin. 3. El número introducido de preguntas no entra en el rango recomendado.
Comentarios	Una vez creado es visible para el resto de los administradores. Al introducir el número de preguntas durante el juego, estando acotado entre valores recomendados. Las preguntas que mostrará el juego al estudiante serán por selección aleatoria y equilibrada de temas.
Actores	Admin web.

Tabla 7 Requisito: Crear juego teórico

Requisito	**Modificar juego teórico**
Id.	2.2.2
Prioridad	Media
Precondición	Debe haberse creado el juego teórico previamente.
Descripción	El usuario de la aplicación web podrá modificar los parámetros iniciales del juego tras haberlo creado.
Entrada	Número de preguntas.
Salida	

Secuencia normal	1. El usuario accede a la aplicación web, se muestra la pantalla principal con el juego creado y pulsa en "Modificar juego teórico" en el juego. 2. La aplicación muestra una pantalla con el campo de número de páginas para modificar el juego. 3. El usuario introduce el nuevo campo y pulsa "aceptar". 4. Se muestra la pantalla principal de la aplicación.
Postcondición	Se han modificado en los registros del sistema los parámetros del juego
Excepciones	3. El número introducido de preguntas no entra en el rango recomendado.
Actores	Admin web.

Tabla 8 Requisito: Modificar juego teórico

Requisito	**Eliminar juego teórico**
Id.	2.2.3
Prioridad	baja
Precondición	Debe haberse creado el juego teórico previamente.
Descripción	El usuario de la aplicación web podrá eliminar permanentemente el juego con sus bloques y preguntas asociadas tras haberlo creado.
Entrada	-
Salida	-
Secuencia normal	1. El usuario accede en la aplicación web, al juego creado y pulsa en "Eliminar juego teórico" 2. Se muestra una ventana de confirmación. 3. El usuario pulsa en "Aceptar y eliminar" 4. Se muestra la ventana principal de la aplicación.
Postcondición	Se ha eliminado el juego teórico con sus registros asociados de bloques temáticos y preguntas, del sistema.
Excepciones	3. El usuario pulsa en "Cancelar".
Actores	Admin web.

Tabla 9 Requisito: Eliminar juego teórico

Requisito	**Listar juegos publicados**
Id.	2.2.10
Prioridad	Alta

Precondición	Debe existir algún juego creado y publicado en la aplicación.
Descripción	El usuario admin web puede listar todos los juegos creados en la aplicación que se hayan publicado, estén activos o no.
Entrada	-
Salida	Lista de juegos publicados en la aplicación.
Secuencia normal	1. El usuario de la aplicación web pulsa en "listar juegos publicados" 2. Se muestran en una nueva venta todos los juegos publicados. Si hay publicado juego de tipo teórico, aparecerá el primero junto con los de tipo competición, ordenados horizontalmente, por fecha de creación, de más reciente a más antiguo. Mostrando para el juego teórico, título, si está activo y un botón de "Ver juego teórico". Para el juego de competición, muestra título, si está activo o no y las opciones de "Ver juego", "Modificar" y "Eliminar juego".
Postcondición	-
Excepciones	1 Aún no hay juegos publicados que listar.
Comentarios	Estar activo es estar publicado actualmente.
Actores	Admin web

Tabla 10 Requisito: Listar juegos publicados

Requisito	**Listar juegos sin publicar**
Id.	2.2.11
Prioridad	Alta
Precondición	Debe existir algún juego creado y sin publicar en la aplicación.
Descripción	El usuario admin web puede listar todos los juegos creados en la aplicación que no se hayan publicado.
Entrada	-
Salida	Lista de juegos no publicados en la aplicación.
Secuencia normal	1. El usuario de la aplicación web pulsa en "listar juegos no publicados" 2. Se muestran en una nueva venta todos los juegos publicados. Si hay publicado juego de tipo teórico, aparecerá el primero, junto con los de tipo competición, ordenados horizontalmente, por fecha de creación, de más reciente a más antiguo.

	Mostrando para el juego teórico el título y para los de competición el título y fecha, además de las opciones de "Ver juego", "Modificar" y "Eliminar juego" correspondiente.
Postcondición	--
Excepciones	1 Aún no hay juegos creados sin publicar que listar.
Comentarios	Los juegos sin publicar son siempre inactivos.
Actores	Admin web

Tabla 11 Requisito: Listar juegos sin publicar

Requisito	**Ver juego teórico**
Id.	2.2.12
Prioridad	Alta
Precondición	El juego teórico debe estar creado.
Descripción	El usuario puede visualizar los datos del juego teórico, esté publicado o sin publicar.
Entrada	-
Salida	Se visualizan los datos del juego teórico y lista sus bloques temáticos asociados.
Secuencia normal	1. El usuario al listar los juegos pulsa en "Ver juego". 2. Se le muestra al usuario en una nueva ventana los datos del juego teórico de Nefrología, es decir, número de preguntas por defecto, si está activo o no, y opciones para "Publicar" y "Retirar" Igualmente muestra una lista horizontal de los bloques temáticos creados asociados al juego ordenados por nombre, mostrando para cada uno, su nombre, si es favorito MIR o no, y opciones para "Ver preguntas del bloque", "Modificar Bloque" y "Eliminar Bloque"
Postcondición	-
Excepciones	1. Error al cargar los datos del juego.
Actores	Admin web

Tabla 12 Requisito: Ver juego teórico

Requisito	**Publicar juego**

Id.	2.2.18
Prioridad	Alta
Precondición	El juego debe estar creado y en el caso de competición deberá contener los campos de fecha inicio y fin.
Descripción	El usuario de la aplicación web tras crearse el juego teórico con distintos bloques y preguntas asociadas, podrá publicar el juego teórico en la aplicación móvil para los usuarios estudiantes. O bien tras crearse un juego de competición con fechas inicio y fin podrá publicarlo igualmente.
Entrada	-
Salida	-
Secuencia normal	1. El usuario al visualizar el juego teórico o de competición pulsa en la opción de "Publicar juego" 2. Se le muestra una opción de confirmar y pulsa "Continuar". 3. Se muestra la visualización del juego inicial.
Postcondición	El juego es visible para los usuarios móviles.
Excepciones	1 El juego ya está publicado y se muestra el mensaje correspondiente. 1 El juego no cumple los requisitos necesarios para ser publicado de mínimo de preguntas y muestra el mensaje de error correspondiente
Actores	Admin web.

Tabla 13 Requisito: Publicar juego

Requisito	**Retirar juego**
Id.	2.2.19
Prioridad	Alta
Precondición	El juego de cualquier tipo debe estar publicado con sus condiciones para poder retirarse.
Descripción	El usuario de la aplicación web puede retirar la visibilidad del juego teórico de la aplicación móvil.
Entrada	-
Salida	-

Secuencia normal	1. El usuario al visualizar el juego teórico o de competición pulsa en la opción de "Retirar juego" 2. Se le muestra una opción de confirmar y pulsa "Continuar". 3. Se muestra la visualización del juego inicial.
Postcondición	-
Excepciones	-
Actores	Admin web.

Tabla 14 Requisito: Retirar juego

Bloques temáticos

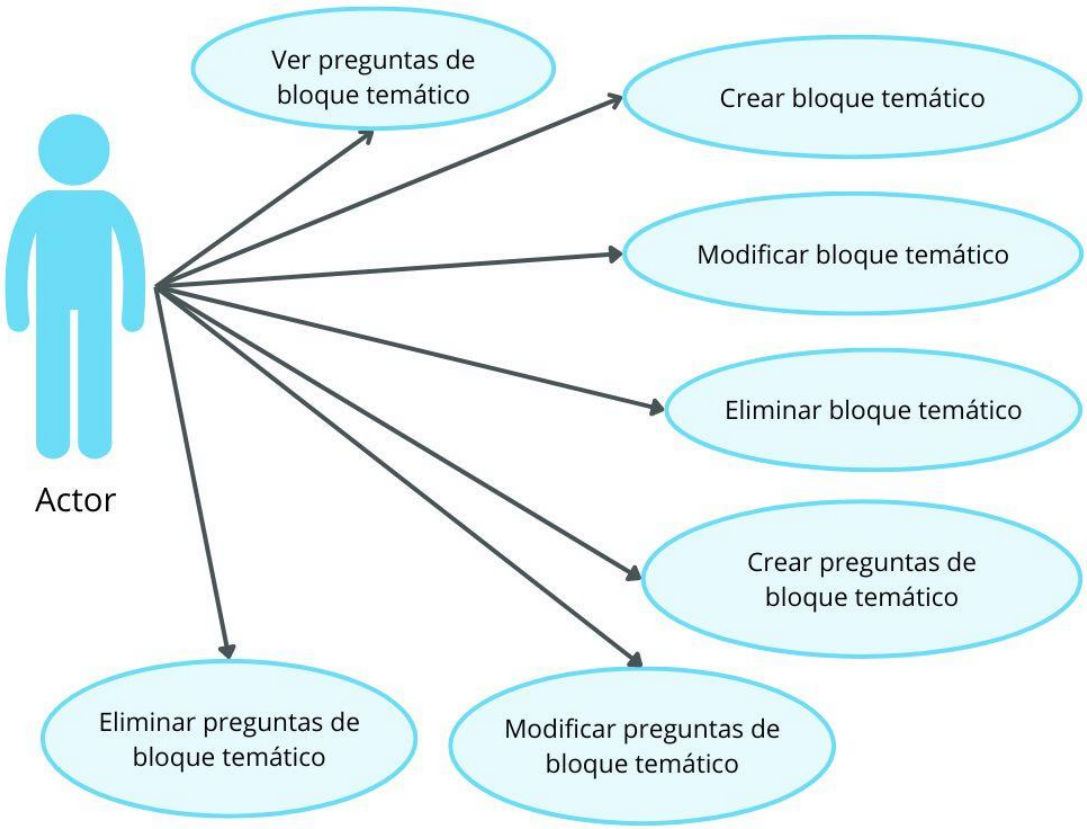

Figura 94 Diagrama de caso de uso Bloques temáticos

Requisito	**Crear bloque temático**
Id.	2.2.4
Prioridad	Alta
Precondición	Debe haberse creado el juego teórico previamente.
Descripción	El usuario médico de la herramienta podrá, tras crear el juego teórico, asignarle un bloque temático (introducido o elegido entre los recomendados por la herramienta) junto con la especificación de si es un bloque de preguntas repetitivas en exámenes MIR (favorito).
Entrada	Nombre del bloque temático, favorito en MIR o no.
Salida	-
Secuencia normal	1. El usuario accede a la aplicación web y al juego teórico creado, se muestra la pantalla principal y pulsa en "Añadir bloque temático" en el juego. 2. La aplicación muestra una pantalla con los campos necesarios para añadir el bloque temático. 3. El usuario introduce los datos requeridos por la herramienta y pulsa "aceptar".
Postcondición	Se ha añadido el bloque temático indicado con sus características al juego teórico.
Excepciones	3. No se han rellenado los campos requeridos y el sistema muestra el mensaje de error correspondiente. 3. Si el bloque temático ya está creado muestra el mensaje correspondiente. 3. El usuario pulsa en "Cancelar".
Comentarios	Una vez creado es visible para el resto de los administradores. Los temas recomendados por el sistema para seleccionar son los oficiales que entran en la oposición del MIR en España: 1. Introducción 2. Líquidos y Electrolitos 3. Fracaso Renal Agudo 4. Insuficiencia Renal Crónica 5. Diálisis y Trasplante 6. Glomerulopatías primitivas 7. Glomerulopatías Secundarias a Enfermedades Sistémicas 8. Nefropatías Tubulointersticiales 9. Enfermedades Vasculares 10. Diuréticos y otros fármacos 11. Infecciones 12. Nefrolitiasis 13. Tumores Renales

Actores	Admin web.

Tabla 15 Requisito: Crear bloque temático

Requisito	**Modificar bloque temático**
Id.	2.2.5
Prioridad	Media
Precondición	Debe haberse creado el bloque temático previamente.
Descripción	El usuario puede modificar los parámetros de creación del bloque temático.
Entrada	Nombre del bloque temático, favorito en MIR o no.
Salida	-
Secuencia normal	1. El usuario accede al juego y pulsa en el bloque creado en "Modificar". 2. La herramienta muestra una pantalla con los campos necesarios para modificar el bloque temático. 3. El usuario introduce los datos requeridos por la herramienta y pulsa "aceptar".
Postcondición	Los datos del bloque han sido modificados en el sistema.
Excepciones	3. El usuario pulsa en "Cancelar".
Actores	Admin web.

Tabla 16 Requisito: Modificar bloque temático

Requisito	**Eliminar bloque temático**
Id.	2.2.6
Prioridad	Baja
Precondición	El bloque temático debe haberse creado en el sistema.
Descripción	El usuario admin web puede eliminar bloques ya creados en el sistema de manera definitiva, eliminando todas sus preguntas creadas asociadas.
Entrada	-
Salida	-

Secuencia normal	1. El usuario accede al bloque temático y pulsa en el botón de "Eliminar bloque". 2. Se muestra una pantalla con el mensaje informativo de la acción a realizar y pregunta si está seguro. 3. El usuario pulsa en el botón "Aceptar y eliminar".
Postcondición	Se ha eliminado del sistema el bloque temático y sus preguntas asociadas.
Excepciones	3. El usuario pulsa en "Cancelar".
Actores	Admin web.

Tabla 17 Requisito: Eliminar bloque temático

Requisito	**Crear preguntas de bloque temático**
Id.	2.2.7
Prioridad	Alta
Precondición	Debe haberse creado el juego teórico y existir un bloque temático al que asignar la pregunta que se crea.
Descripción	Dentro de un bloque temático el usuario crea una pregunta test, asignando una dificultad, fácil, regular o difícil. La pregunta podrá ser de tres tipos (todas a elegir una respuesta entre cuatro): Teórica, con enunciado y respuestas más breves Práctica, un caso práctico clínico, con posibilidad de estar ligado a una imagen. Pregunta negativa, elegir la errónea.
Entrada	Dificultad, Tipo, Enunciado, 4 respuestas, foto opcional, Explicación
Salida	-
Secuencia normal	1. El usuario accede a "Crear pregunta" dentro de un bloque. 2. El usuario rellena los campos y pulsa en "Crear". 3. El sistema valida los datos. 4. Se muestra la ventana principal del juego teórico con los bloques.
Postcondición	Se ha registrado la pregunta en el sistema para el bloque indicado.
Excepciones	2. Los datos introducidos exceden el límite de caracteres para el enunciado o respuestas.
Comentarios	Una vez creada es visible para el resto de los administradores. Todos campos obligatorios, excepto el de la imagen.
Actores	Admin web.

Tabla 18 Requisito: Crear preguntas de bloque temático

Requisito	Modificar preguntas de bloque temático
Id.	2.2.8
Prioridad	Media
Precondición	Debe haberse creado el juego teórico y existir un bloque temático con alguna pregunta que modificar.
Descripción	Para cambiar preguntas con errores la herramienta da la opción de modificar preguntas ya creadas.
Entrada	Dificultad, Tipo, Enunciado, 4 respuestas, foto.
Salida	
Secuencia normal	1. El usuario en la pantalla de visualizar preguntas pulsa en "modificar pregunta". 2. Se muestran en otra pantalla los campos editables de la pregunta. 3. El usuario modifica lo necesario y pulsa en "modificar". 4. El sistema valida los datos. 5. Se muestra la ventana de preguntas del bloque.
Postcondición	El sistema almacena los nuevos campos modificados de la pregunta del bloque correspondiente.
Excepciones	3. Los datos introducidos exceden el límite de caracteres para el enunciado o respuestas.
Actores	Admin web

Tabla 19 Requisito: Modificar preguntas de bloque temático

Requisito	Eliminar preguntas de bloque temático
Id.	2.2.9
Prioridad	Baja
Precondición	Debe existir alguna pregunta creada para eliminar.
Descripción	Una vez creadas preguntas de un bloque el usuario podrá eliminar la pregunta seleccionada del sistema.
Entrada	
Salida	

Secuencia normal	1. El usuario desde la ventana de visualización de preguntas pulsa en "Eliminar pregunta". 2. Se muestra una ventana pidiendo confirmación de seguridad. 3. El usuario pulsa "aceptar y eliminar"
Postcondición	La pregunta ha sido eliminada del sistema.
Excepciones	-
Actores	Admin web

Tabla 20 Requisito: Eliminar preguntas de bloque temático

Requisito	**Ver preguntas de un bloque**
Id.	2.2.13
Prioridad	Alta
Precondición	Debe existir al menos una pregunta creada en el bloque
Descripción	El usuario puede visualizar las preguntas ya creadas por él u otros administradores de la herramienta para el bloque temático en el que se sitúa, pudiendo filtrar por tipo de pregunta, nivel y palabra clave en el enunciado.
Entrada	Opcional tipo de pregunta, nivel y palabra clave.
Salida	
Secuencia normal	1. El usuario accede a "Ver preguntas del bloque" en el bloque correspondiente del juego. 2. Se muestra en una ventana todas las preguntas creadas en ese bloque. Las preguntas se muestran verticalmente, con sus datos respectivos a cada tipo de pregunta y su explicación, ordenadas por tipo de pregunta, y para cada tipo ordenadas por dificultad. Junto a cada pregunta se muestra un campo "Modificar pregunta" y "Eliminar pregunta".
Postcondición	-
Excepciones	1. No hay preguntas creadas aún para el bloque
Actores	Admin web

Tabla 21 Requisito: Ver preguntas de un bloque

Juegos de competición

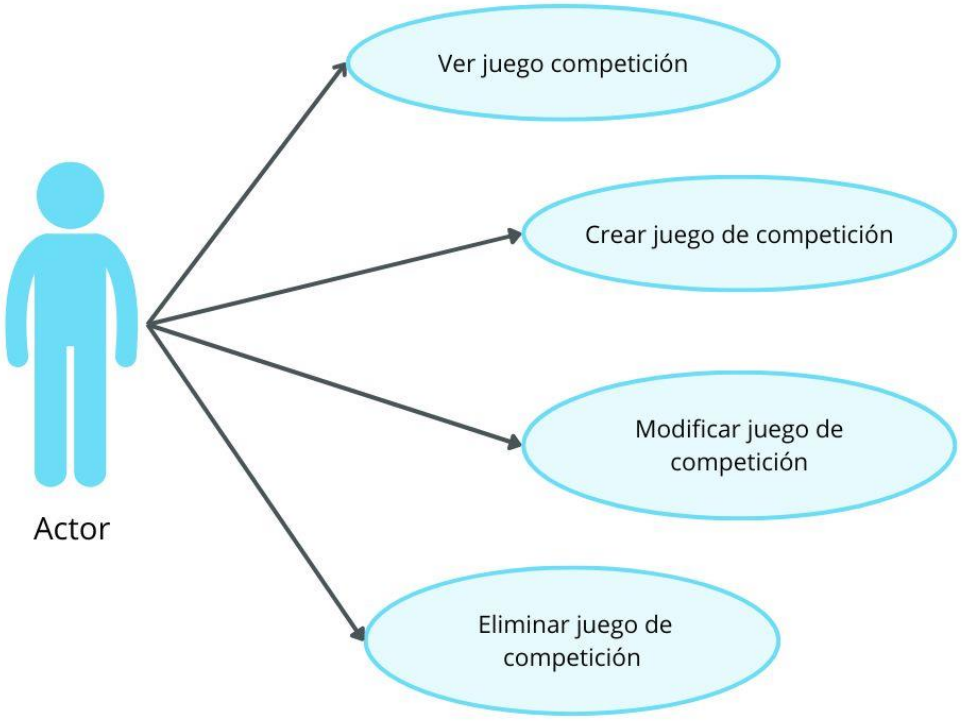

Figura 95 Diagrama de caso de uso Juegos de competición

Requisito	**Ver juego competición**
Id.	2.2.14
Prioridad	Alta
Precondición	El juego debe estar creado en el sistema
Descripción	El usuario puede visualizar los datos de un juego de competición esté publicado o sin publicar.
Entrada	-
Salida	Se visualizan los datos del juego de competición.
Secuencia normal	1. El usuario al listar los juegos pulsa en "Ver juego" del tipo competición. 2. Se le muestra al usuario en una nueva ventana los datos de creación del juego de competición, título, número de preguntas, fecha inicio, fecha de fin, porcentajes de niveles y bloques, objetivo de aciertos y las opciones de "Publicar" y "Retirar".

Postcondición	-
Excepciones	1. Error al cargar los datos del juego.
Actores	Admin web.

Tabla 22 Requisito: Ver juego competición

Requisito	**Crear juego de competición**
Id.	2.2.15
Prioridad	Alta
Precondición	El juego teórico está creado y existe una batería de preguntas necesarias de teoría.
Descripción	El usuario de la herramienta puede crear un juego de competición configurando los parámetros a su gusto, y así el usuario móvil puede poner a prueba lo aprendido jugando en la competición. Estos parámetros serán fijos e inmodificables por el usuario móvil para cada vez que realiza el juego.
Entrada	Título competición, número de preguntas, fecha inicio, fecha fin, porcentajes de niveles a sacar y bloques temáticos a sacar. Objetivo aciertos del total (cuando "se gana el test").
Salida	-
Secuencia normal	1. El usuario accede a la aplicación web, se muestra la pantalla principal y pulsa en "Crear juego de competición" 2. La herramienta muestra una pantalla con los campos de entrada mencionados necesarios para crear el juego. 3. El usuario introduce los campos requeridos por la herramienta y pulsa "aceptar"
Postcondición	El sistema crea en sus registros el juego con sus características.
Excepciones	1 No existe juego teórico o batería de test mínima y el sistema muestra el mensaje de error correspondiente. 3. Faltan campos obligatorios por completar y el sistema muestra el mensaje de error correspondiente. 3. El número de preguntas seleccionadas no es suficiente para el total indicado.
Comentarios	El juego se crea también con parámetros predefinidos, *como el número de partidas diarias máximas (5) , niveles (10) y condiciones para subir de nivel: acertar objetivo en 3 de 5 pases.*
Actores	Admin web

Tabla 23 Requisito: Crear juego de competición

Requisito	Modificar juego de competición
Id.	2.2.16
Prioridad	Media
Precondición	Hay un juego de competición creado para poder modificarlo. Para poder modificar un juego de competición, debe de estar inactivo en el momento de intento de modificación.
Descripción	El usuario de la aplicación web, puede modificar los parámetros de un juego de competición a su gusto, y así modificar el juego a los usuarios móviles finales. Si el juego está en estado sin publicar, el campo de fecha inicio y fin no es obligatorio.
Entrada	Título competición, número de preguntas, fecha inicio, fecha fin, preguntas aleatorias o seleccionadas. Si aleatorias, además, porcentajes de niveles a sacar, bloques temáticos a sacar, Objetivo aciertos (cuando "se gana el test")
Salida	
Secuencia normal	1. El usuario accede a modificar un juego de competición creado. 2. Se muestra una pantalla con los campos del juego modificables. 3. El usuario introduce los nuevos valores en los campos deseados y pulsa "aceptar"
Postcondición	El juego de competición ha sido modificado.
Excepciones	3. El usuario modifica el número de preguntas del test y no existe batería de test mínima por lo que el sistema muestra el mensaje de error correspondiente. 3. Faltan campos obligatorios por completar y el sistema muestra el mensaje de error correspondiente. 3. El número de preguntas seleccionadas no es suficiente para el total indicado.
Comentarios	Para modificar una competición activa, habría que retirarla, pasaría a estar retirada inactiva y ya se podría modificar, publicar y volverse activa con nueva fecha de inicio y fin.
Actores	Admin web.

Tabla 24 Requisito: Modificar juego de competición

Requisito	Eliminar juego de competición
Id.	2.2.17
Prioridad	Media
Precondición	Hay un juego de competición creado para poder eliminarlo. Para poder eliminar un juego de competición, debe de estar inactivo en el momento de intento de eliminación.
Descripción	El usuario de la aplicación web puede eliminar definitivamente un juego de competición del sistema.
Entrada	-
Salida	-
Secuencia normal	1. El usuario al visualizar la lista de juegos publicados pulsa en la opción de "Eliminar juego" en un juego de competición. 2. Se le muestra una ventana de confirmación si todo es correcto. 3. El usuario pulsa en "Confirmar y aceptar"
Postcondición	-
Excepciones	1 El juego no puede ser eliminado al estar en estado Activo para los usuarios móviles, y la aplicación recomienda retirar previamente.
Comentarios	Para eliminar una competición activa, habría que retirarla, pasaría a estar retirada inactiva y ya se podría eliminar del sistema.
Actores	Admin web.

Tabla 25 Requisito: Eliminar juego de competición

Gestión de resultados

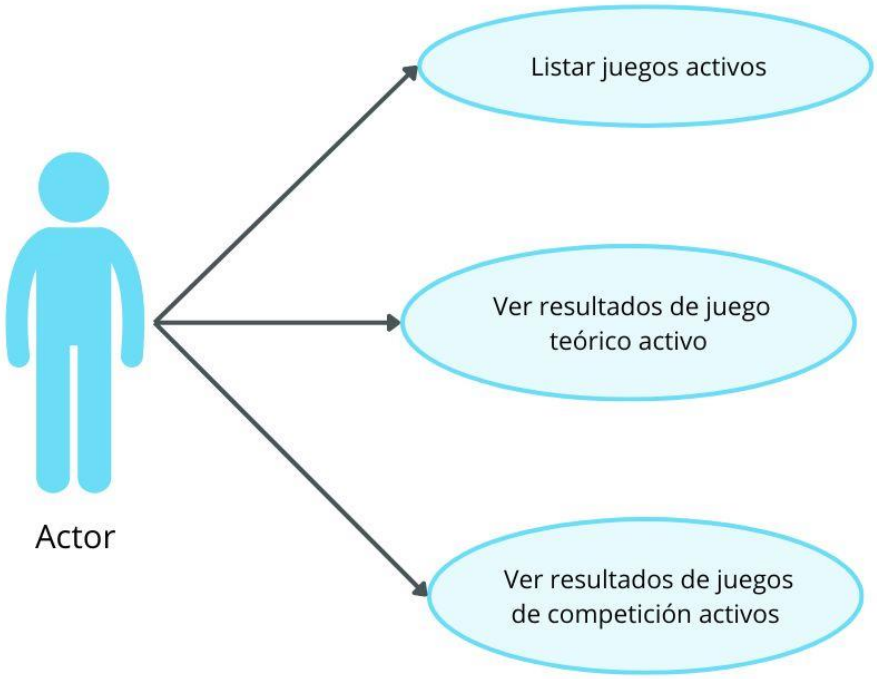

Figura 96 Diagrama de caso de uso Gestión de resultados

Requisito	**Listar juegos activos.**
Id.	2.3.1
Prioridad	Alta
Precondición	Debe existir algún juego creado en activo que mostrar.
Descripción	El usuario admin web accede a ver los juegos disponibles activos para poder acceder a visualizar los resultados de estos.
Entrada	-
Salida	Lista de juegos publicados en activo

Secuencia normal	1. El usuario admin web accede a listar los juegos en activo. 2. Se muestra en otra ventana, si existe en primer lugar el juego teórico en activo, y los juegos de competición en activo ordenados por fecha de inicio. Todos mostrando su título y "Ver resultados".
Postcondición	-
Excepciones	1 No hay juegos en activo para ver
Comentarios	En activo implica estar publicado.
Actores	Admin web

Tabla 26 Requisito: Listar juegos activos.

Requisito	**Ver resultados de juego teórico activo**
Id.	2.3.2
Prioridad	Media
Precondición	El usuario admin web debe haber iniciado sesión, existir algún juego teórico activo y algún usuario haber jugado alguna vez para mostrar datos.
Descripción	El usuario médico para poder mejorar en bloques temáticos puede visualizar un ranking de los temas más fallados en las preguntas de tipo teórico para cada nivel de dificultad.
Entrada	-
Salida	Rankings
Secuencia normal	1. El usuario admin web, accede a "Ver resultados" en el juego teórico al listar los juegos activos. 2. Se muestra en una nueva ventana, un ranking para cada nivel de dificultad, ordenando cada uno los bloques temáticos por mayor porcentaje de fallos obtenido al responder los estudiantes preguntas de ese bloque.
Postcondición	-
Excepciones	2 Aún no hay datos que mostrar.
Actores	Admin web

Tabla 27 Requisito: Ver resultados de juego teórico activo

Requisito	**Ver resultados de juegos de competición activos**
Id.	2.3.3
Prioridad	Media
Precondición	El usuario admin web debe haber iniciado sesión, existir algún juego de competición activo y algún usuario haber jugado alguna vez para mostrar datos.
Descripción	El usuario admin para ver la evolución en la constancia diaria de los estudiantes que juegan en la competición, pudiendo ver la cantidad de estudiantes que hay en cada nivel y observar, a qué altura del juego (día aproximado), los estudiantes de qué universidad dejan de competir.
Entrada	-
Salida	Ranking
Secuencia normal	1. El usuario admin web accede a "ver resultados" de una competición al visualizar los juegos de competición activos. 2. Se muestra en una nueva pantalla el ranking de usuarios, mostrando los 10 niveles, indicando para cada nivel el número de usuarios que hay y la cantidad de estudiantes de cada universidad ordenados de mayor a menor.
Postcondición	-
Excepciones	2 Aún no hay datos que mostrar.
Actores	Admin web

Tabla 28 Requisito: Ver resultados de juegos de competición activos

Jugar

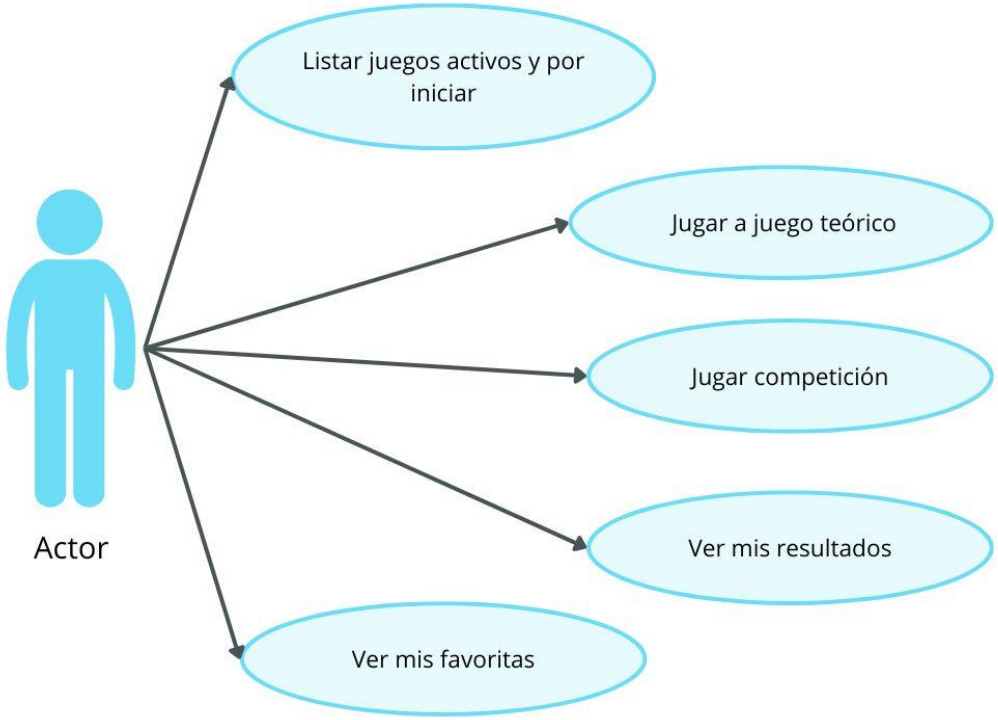

Figura 97 Diagrama de caso de uso de Jugar

Requisito	**Listar juegos activos y por iniciar.**
Id.	2.4.1
Prioridad	Alta
Precondición	Debe existir algún juego creado en activo que mostrar.
Descripción	El usuario accede a ver los juegos disponibles activos y los que próximamente estarán activos, para poder acceder a practicar en el juego teórico o a competir.
Entrada	-
Salida	Lista de juegos activos

Secuencia normal	1. El usuario pulsa en "Ver juegos activos". 2. Se muestra una nueva pantalla donde se despliegan los distintos juegos activos y por iniciar, mostrando el teórico en primer lugar si existe, y el resto de competición ordenados en vertical por fecha de comienzo. Para el juego teórico se muestra el título, número de preguntas por defecto y la opción de "Configurar y Empezar". Para el juego de competición se muestra el título, el número de preguntas y la opción de "Jugar (Comenzar pase)", si se encuentra activo, si no, la fecha de apertura (inicio).
Postcondición	-
Excepciones	1 No hay juegos activos que mostrar 2 Error al cargar los datos de los juegos activos con su correspondiente mensaje de error. 2 Al "jugar" a juego de competición, si aún no está en activo muestra el aviso correspondiente.
Comentarios	Por iniciar: la fecha de inicio aún no se ha alcanzado.
Actores	Usuario móvil registrado, Usuario móvil no registrado

Tabla 29 Requisito: Listar juegos activos y por iniciar

Requisito	**Jugar a juego teórico**
Id.	2.4.2
Prioridad	Alta
Precondición	El juego debe estar creado y publicado
Descripción	El usuario móvil esté registrado o no podrá jugar a "juego teórico" realizando test personalizados, en el momento de jugar, para poder aprender o realizar el juego de competición, visualizando las respuestas tras contestar a todas las preguntas.
Entrada	Dificultad, tipo, solo favoritas, bloques favoritos MIR, usuario
Salida	Se obtiene el número de aciertos.
Secuencia normal	1. El usuario registrado o no, situado en la ventana principal de "listado de juegos activos y por iniciar" pulsa en "Iniciar juego teórico". 2. Se le muestra una nueva pantalla donde se ofrece la opción de personalizar las preguntas del test, multiopción de dificultad de las preguntas, multiopción del tipo de pregunta, marcador solo favoritas y solo bloques favoritos MIR. 3. El usuario modifica los parámetros del test a su gusto y pulsa "iniciar". 4. Se muestra en una nueva ventana el juego cargado con las preguntas de la BBDD seleccionadas aleatoriamente en base a lo

	establecido por el usuario. En la parte superior aparecen los índices de las preguntas para poder redirigir de una a otra, y en el total del resto de la pantalla la pregunta seleccionada con sus opciones de respuestas a seleccionar. Se le da la opción en cada pregunta de marcar con una estrella la pregunta como favorita. 5. El usuario marca una respuesta y la aplicación marca acierto y modifica la pantalla con la siguiente pregunta. 6. El usuario marca todas las respuestas posibles y da a "finalizar test". 7. Se muestra una nueva pantalla con el mismo formato anterior, pero con la puntuación obtenida, y las respuestas correctas resaltadas, junto con una explicación. 8. El usuario pulsa en "Continuar" y vuelve a la pantalla inicial.
Postcondición	Si el usuario estaba registrado, se almacena la fecha de realización y su puntuación en los registros del sistema.
Excepciones	3. El usuario pulsa "cancelar". 5. El usuario al seleccionar la pregunta errónea se muestra el error y se pasa a la siguiente pregunta.
Comentarios	Este resultado si el usuario está registrado se muestra ya en la gráfica del apartado "Ver mis resultados"
Actores	Usuario móvil registrado.

Tabla 30 Requisito: Requisito: Jugar a juego teórico

Requisito	**Jugar competición**
Id.	2.4.3
Prioridad	Alta
Precondición	El usuario debe estar registrado y con la sesión iniciada, el juego de competición debe estar publicado y activo.
Descripción	El usuario registrado puede acceder a un juego de competición en el que podrá realizar un test de nivel personalizado por un médico (admin web). Podrá acceder a él mientras esté activo, una vez al día, y permitirá 5 intentos cada día. Hay 10 niveles en la competición y para subir de nivel hay que pasar 3 de 5 test, llegando al número de aciertos objetivo fijos establecidos por el admin web.
Entrada	-

Salida	Se obtiene el número de aciertos, si se consigue estrella al superar los aciertos al objetivo, y el nuevo nivel y posición en el ranking de usuarios jugando.
Secuencia normal	1. El usuario inicia un juego de competición por primera vez en el día y se muestra en una pantalla el número de aciertos para aprobar un test, y los 5 pases a los 5 test, mostrados como 5 estrellas vacías y su posición en el ranking de usuarios jugando. El usuario pulsa en "iniciar un pase" a un test. 2. Se muestra un test con el formato del juego teórico y el usuario lo finaliza, superando el objetivo de aciertos y finaliza. 3. Se muestra la pantalla de aciertos descrita en el punto 1 con la información actualizada, se rellena una estrella y cambia la posición del ranking. Permite iniciar los cuatro pases restantes. 4. El usuario realiza el resto de los pases y finaliza el intento diario. 5. Se muestra una pantalla con los resultados de los test acertados y mostrando el nuevo nivel obtenido al superar 3 de 5 pases. El usuario pulsa "aceptar". 6. La aplicación redirige a la ventana inicial.
Postcondición	Se obtiene la posición en ranking
Excepciones	1 El usuario ya ha realizado los 5 pases diarios y se muestra un mensaje de error.
Comentarios	Este resultado actualizado se muestra en el apartado "Ver mis resultados".
Actores	Usuario móvil registrado

Tabla 31 Requisito: Jugar competición

Requisito	**Ver mis resultados**
Id.	2.4.4
Prioridad	Alta
Precondición	Se debe haber iniciado algún juego activo teórico o de competición para ver resultados.
Descripción	El usuario móvil registrado, para ver el proceso de evolución en sus aciertos de cada vez que realiza un juego teórico, puede visualizar una gráfica con los aciertos por cada intento realizado hasta el momento. De esta manera puede ver con que configuración jugada no llega a un número de aciertos. Igualmente puede ver a continuación por debajo los resultados actualizados del usuario en cada competición activa que haya participado, para ver el estado en el que se encuentra en ese día y en el ranking de usuarios total, y poder ver cuánto le falta para conseguir el certificado. Finalmente puede ver los Certificados de competiciones obtenidos por llegar al nivel 10 en competiciones pasadas.

Entrada	-
Salida	-
Secuencia normal	1. El usuario móvil registrado accede a "Ver mis resultados" 2. Se muestra una nueva pantalla donde en la parte superior se indican los resultados de evolución del juego teórico mediante una gráfica, con eje Y de aciertos por los intentos del eje X. El usuario puede establecer una línea horizontal el usuario en algún número de aciertos. y ver si la alcanza. El usuario puede pulsar en cada intento del eje X que muestra la fecha, y se muestra en otra ventana la configuración exacta de esa prueba. Seguidamente se muestran los resultados de las competiciones activas, con su título, número de estrellas del pase diario conseguidas y posición en ranking total. Finalmente muestra los certificados obtenidos de competiciones pasadas ya inactivas. Mostrando "certificado en *título de la competición*" y la posición dentro del nivel 10.
Postcondición	-
Excepciones	-
Actores	Usuario móvil registrado

Tabla 32 Requisito: Ver mis resultados

Requisito	**Ver favoritas**
Id.	2.4.5
Prioridad	Alta
Precondición	El usuario registrado debe haber jugado algún juego teórico y haber marcado alguna pregunta como favorita.
Descripción	El usuario de la aplicación móvil para poder aprender bien preguntas que duda puede marcar como favoritas preguntas según juega a un juego teórico para poderlas visualizar completamente más tarde tras finalizar el juego.
Entrada	-
Salida	-

Secuencia normal	1. El usuario accede a "ver favoritas" 2. Se muestra una lista con las preguntas favoritas guardadas, ordenadas por bloques y dentro de cada bloque por dificultad. Para cada pregunta se muestra su enunciado, respuestas, posible foto y su explicación, marcando la respuesta correcta. Igualmente se muestra una estrella para cada pregunta rellena, que puede ser desmarcada para eliminar la pregunta de favoritos. 3. El usuario tras visualizarlas pulsa en "salir" y vuelve a la pantalla de inicio.
Postcondición	-
Excepciones	2. Error al cargar preguntas favoritas.
Actores	Usuario móvil registrado

Tabla 33 Requisito: Ver favoritas

Apéndice B - Manual de usuario

Aplicación web:

-Vista de inicio

Figura 98 Vista de inicio web

Funcionalidad menú sin iniciar sesión (Figura 98).

- Figura 99: El administrador pulsando en registrarse, si ha sido registrado previamente en la base de datos, puede completar el proceso rellenando los campos requeridos en el modal que aparece, de correo, nombre, apellidos y la contraseña. Puede visualizar u ocultar su contraseña pulsando en mostrar. Pulsando en registro si ha completado todos datos correctamente se inicia sesión.
- Figura 100: El administrador puede, previo registro exitoso, iniciar sesión introduciendo su correo y contraseña.

Figura 99 Registro web

Figura 100 Inicio de sesión web

-Vista de juegos

La vista de juegos se divide en la parte de la funcionalidad para el juego teórico (Figura 101) y la funcionalidad de los juegos de competición (Figura 102). En estas figuras se encuentran las acciones que se pueden realizar, ennumeradas para su explicación.

Figura 101 Parte de la vista Juegos relativa a juego teórico

Figura 102 Parte de la vista Juegos relativa a juego competición

A continuación, se enumeran las acciones que el usuario puede realizar relativas al juego teórico que se observan en la Figura 101 :

- **1:** Pulsar en el botón "Inicio" para **acceder a la vista principal de inicio** (Figura 98).
- **2:** Pulsar en el botón "Resultados" para **acceder a la vista de resultados** de juegos de competición activos.
- **3:** Pulsar en el botón "Cerrar sesión" para **cerrar la sesión** actualmente iniciada.
- **4:** Pulsar en el botón "Mi perfil" para poder **visualizar los datos del perfil** y a su vez poder editarlos. (Figura 103)
- **5:** Botón de **eliminar o crear el juego teórico** en función de si se está ya creado o no. Si está creado al eliminar se muestra un modal de confirmación como el de la Figura 104.
- **6: Visualizar los detalles del juego teórico**. Al pulsar en la tarjeta del juego teórico, se accede a la vista de detalles del juego teórico.
- **7:** Botón de **editar el juego teórico**, mostrándose un modal como el de la imagen de la Figura 105, en el que se puede introducir un nuevo número de preguntas para el juego teórico.
- **8:** Botón para poder **publicar o retirar el juego teórico**, y así que los usuarios móviles estudiantes puedan visualizarlo o no. En función de si el juego está retirado o no, cambia para retirarlo o publicarlo, mostrando su modal correspondiente al estilo del que se observa en la Figura 106.

Se continúa enumerando las acciones que se pueden realizar, relativas a los juegos de competición, ilustradas en la Figura 102 :

- **9:** Pulsando en este botón se abre el modal de **crear el juego de competición** (Figura 68), introduciendo sus características, como título, fechas, número de preguntas, temas, foto, etc.
- **10:** Pulsando sobre la tarjeta del juego de competición se accede a una nueva vista para poder **ver los detalles del juego de competición**.

- **11:** Pulsando en **editar el juego de competición** se abre el modal (Figura 107) con los datos del juego de competición a modificar. Introduciendo de nuevo el título, el número de preguntas, las fechas de inicio y final, el porcentaje de los tipos de preguntas, los temas creados que lo componen, la imagen y el número de preguntas que dan por ganado el juego.
- **12:** Igual que el punto 8 pero para juegos de competición.

Figura 103 Modal de ver y editar datos del perfil web

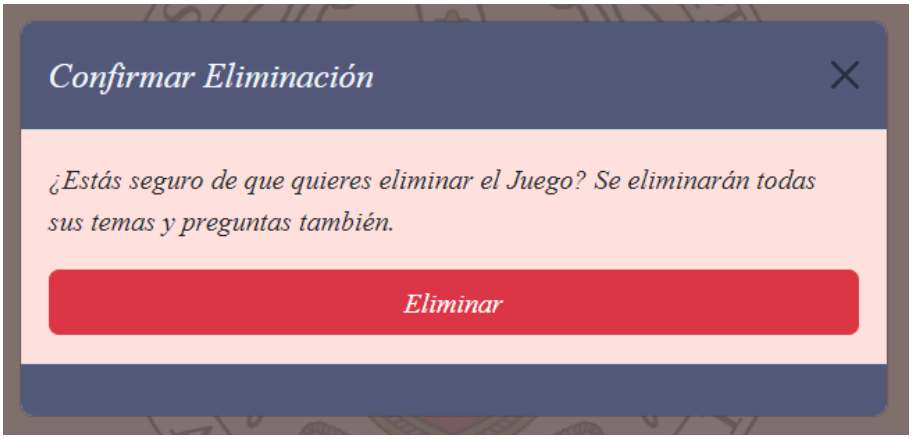

Figura 104 Modal de eliminar juego teórico

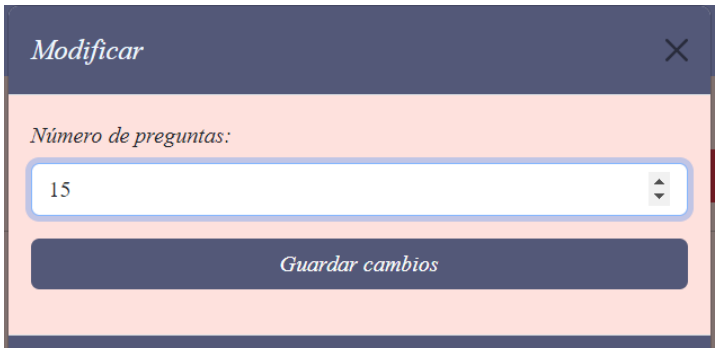

Figura 105 Modal de editar juego teórico

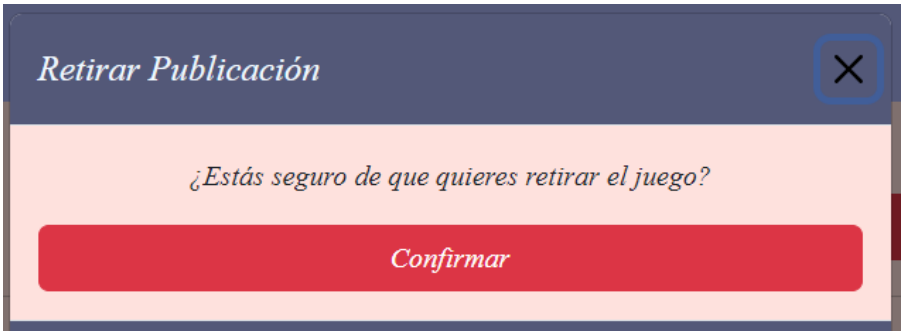

Figura 106 Modal de retirada juego

Figura 107 Modal de editar juego de competición

-Vista detalles de un juego de competición

Pulsando en el botón 10 de la sección anterior se accede a la siguiente vista (Figura 108). En ella se puede visualizar toda la configuración del juego que se ha creado y además pulsar sobre el botón de la sección inferior derecha para eliminarlo. Si se procede a eliminar, se muestra un modal de confirmación (Figura 109).

Figura 108 Vista detalles de un juego de competición

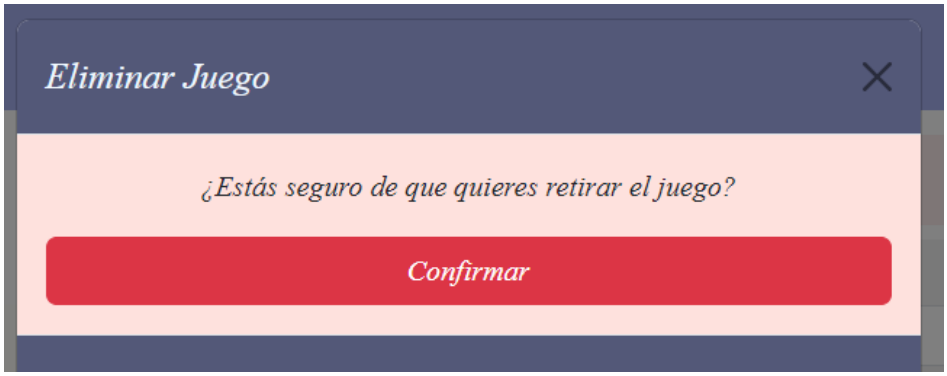

Figura 109 Modal de confirmación eliminación juego de competición

-Vista detalles del juego teórico

Pulsando en el botón 6 de la segunda sección de este apéndice "Vista de juegos" que se observa en la Figura 101, se accede a la vista de detalles del juego teórico. Aquí se pueden visualizar los distintos temas creados. Se indican las distintas acciones que el usuario puede realizar enumeradas en la siguiente imagen Figura 110.

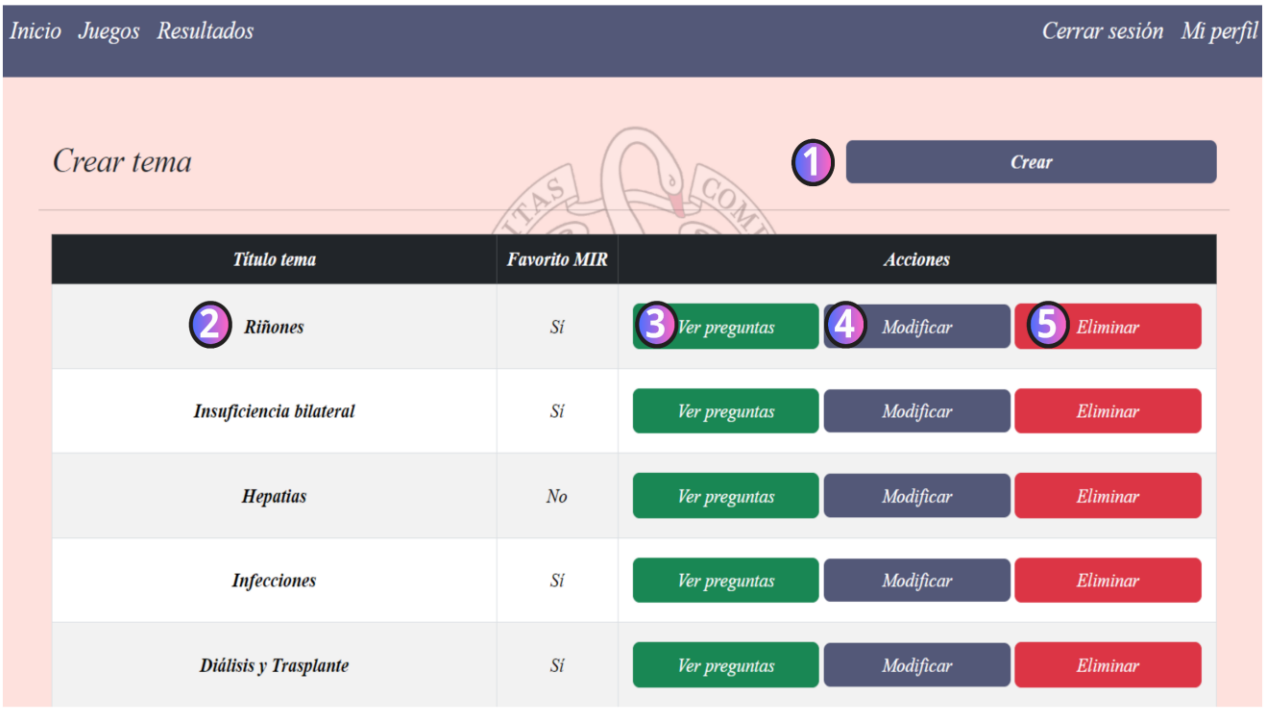

Figura 110 Vista detalles juego teórico

- **1:** Botón para **crear un nuevo tema**. Abre el modal de la Figura 111, donde se puede introducir el título del nuevo tema y si es favorito del MIR.
- **2:** Pulsando en cada fila de la tabla de temas, se puede **crear una pregunta** para ese tema mediante el modal de la Figura 112.
- **3:** Botón para poder **visualizar el listado de preguntas** del tema al que hace referencia. Se abre una nueva vista para el visualizado.
- **4:** Botón para **modificar un tema** a través del modal de la Figura 113.
- **5:** Botón de **eliminar un tema**, eliminando también todas sus preguntas asociadas. Si se pulsa se muestra el modal de la Figura 114.

Figura 111 Modal de crear tema

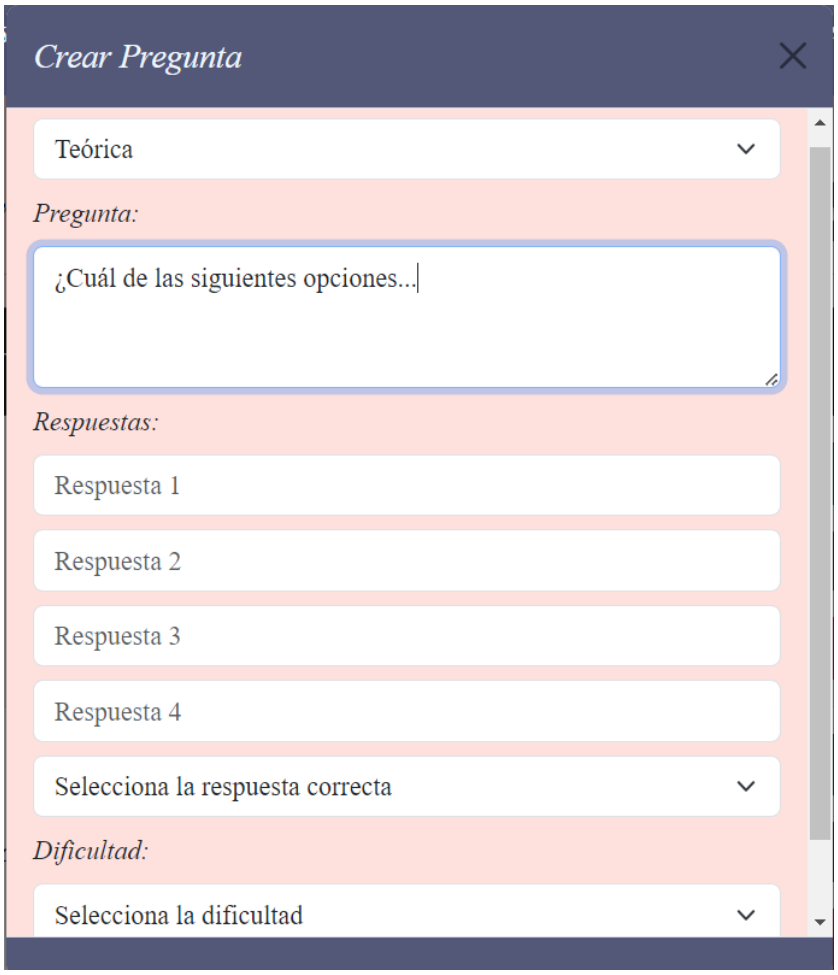

Figura 112 Modal de crear pregunta de un tema

Figura 113 Modal para modificar un tema

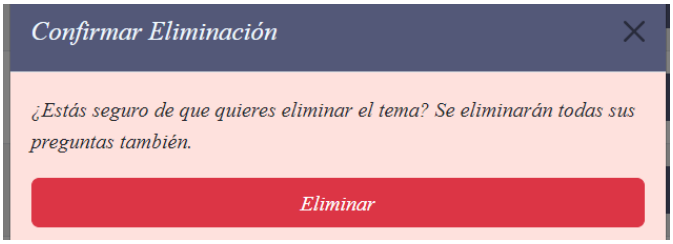

Figura 114 Modal confirmación eliminado de tema

-Vista listado de preguntas de un tema

En esta vista (Figura 115) se muestran verticalmente las tarjetas con la información de las distintas preguntas pertenecientes a un tema. En la parte superior de la tarjeta se observa el tipo de pregunta y su dificultad. En la parte del medio se encuentra el enunciado y las respuestas junto a la posible imagen para las preguntas prácticas. En la parte inferior se muestra un botón a la izquierda para modificar la pregunta (Figura 116) y otro a la derecha para eliminarla (Figura 117), mostrando sus respectivos modales.

Figura 115 Vista listado de preguntas

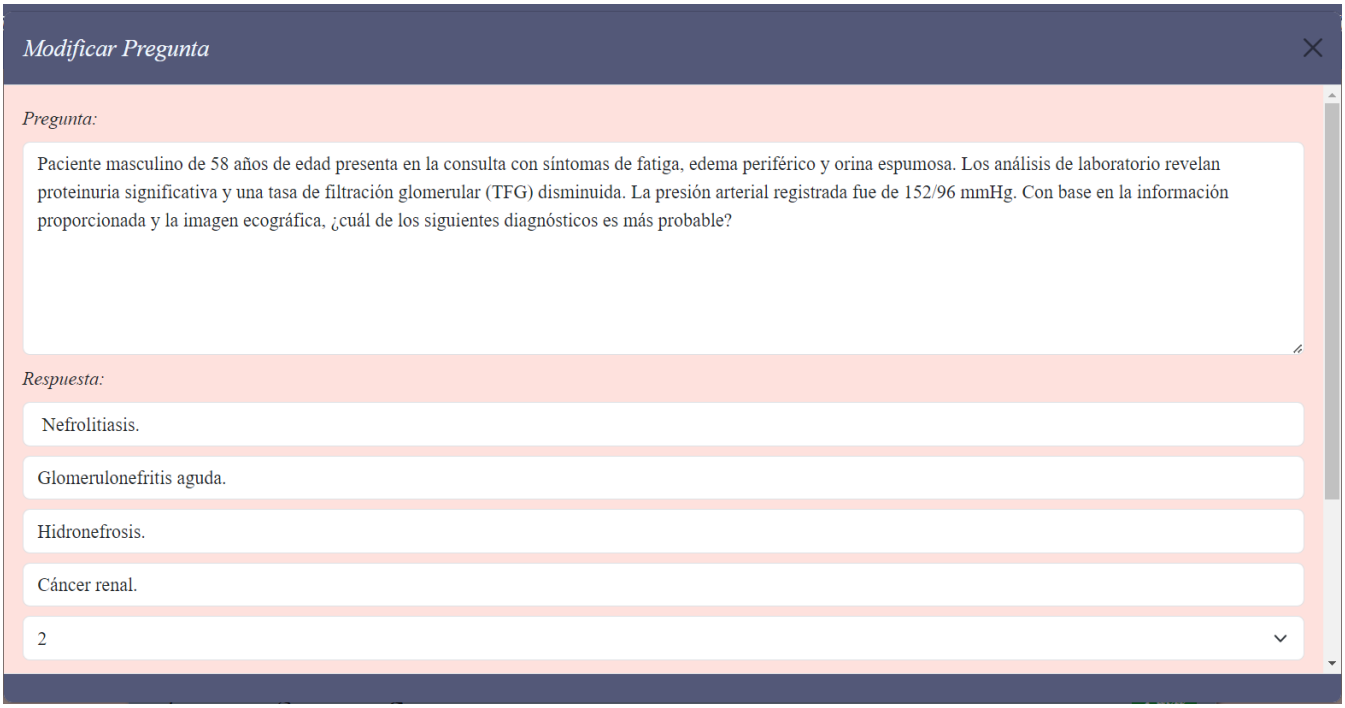

Figura 116 Modal de modificar pregunta

Figura 117 Modal confirmación eliminar pregunta

-Vista de resultados

En esta vista (Figura 118) se pueden visualizar los distintos juegos activos desplegando el selector. Pulsando en uno se muestra la imagen del gráfico correspondiente donde se apilan los grupos de estudiantes por universidad en el eje Y, y en el eje X, se muestran los distintos niveles del juego. Esto permite al usuario conocer en que niveles se encuentran los distintos estudiantes de cada facultad. Al mismo tiempo saber cuántos de esos estudiantes hay en cada universidad para un nivel, se puede ver manteniendo el ratón encima del grupo como se observa en la Figura 119.

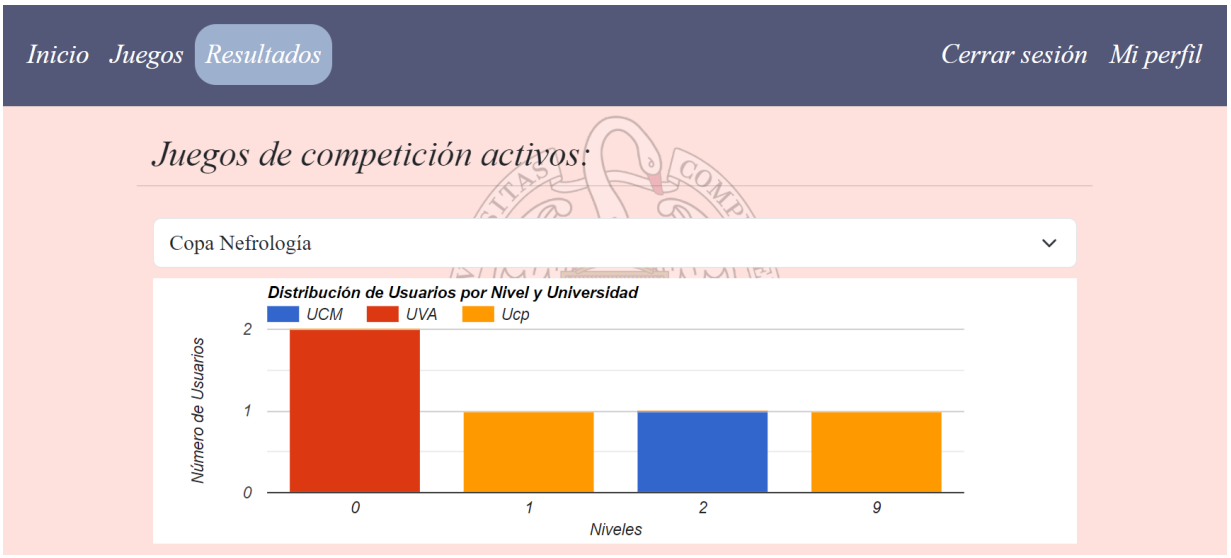

Figura 118 Vista de resultados de juegos de competición

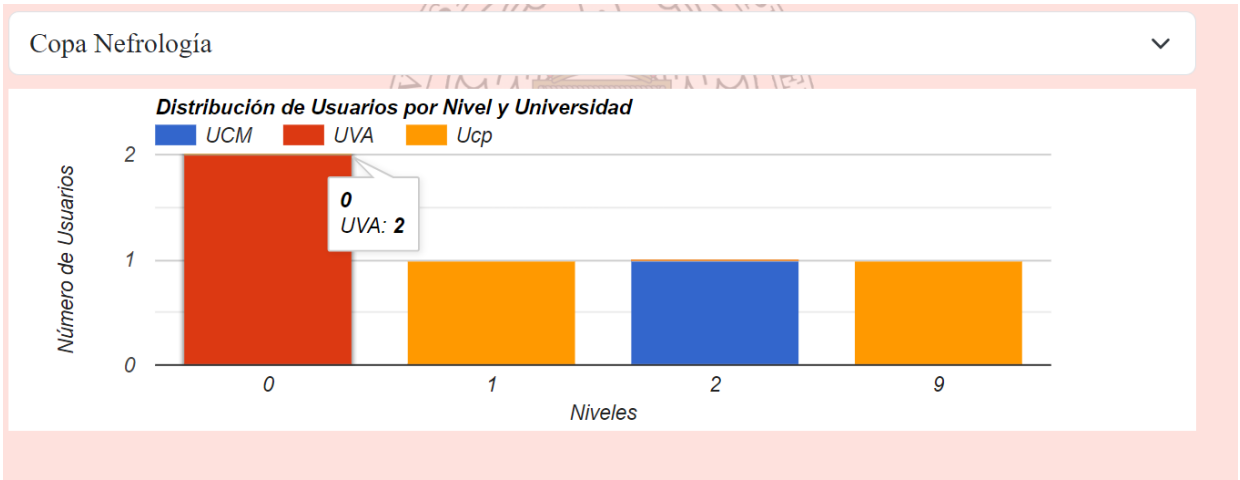

Figura 119 Detalles informativos de la gráfica de resultados web

Aplicación móvil:

-Juegos

Si el usuario no tiene la sesión iniciada, solo podrá realizar en la vista de juegos la visualización de los mismos, pero no realizar ninguna acción sobre ellos, si se intenta acceder a alguno se muestra un mensaje de aviso informando de que hace falta iniciar sesión. El usuario puede ver el título, fecha comienzo y fecha fin de los juegos de competición.

En la vista de juegos con la sesión iniciada el usuario puede realizar las siguientes navegaciones y acciones representadas en la Figura 120. Pulsando en el juego teórico se abre la vista de configuración con los tipos de preguntas y dificultades que formarán el juego de práctica. Si se acepta la configuración se navega a la vista del test teórico, donde se puede contestar a la pregunta actual pulsando sobre la respuesta, cambiar de pregunta con los botones numéricos, añadir la pregunta actual a favoritos y finalmente terminar el test. Los botones númericos de las preguntas, cambian de color según el estado de la pregunta, no contestada, contestada y actual.

Si se inicia un juego de competición la aplicación muestra el mismo formato de test explicado anteriormente pero con la configuración interna aplicada por los administradores.

Una vez pulsado teminar juego, se muestra el mismo test corregido, mostrando los aciertos, fallos y puntuación obtenida. Igualmente se puede navegar por las distintas preguntas pulsando sobre el botón del número que la representa. Los aciertos se muestran en verde y los fallos en rojo. Las reglas del juego de competición son las siguientes:

- Se tiene 5 intentos (pases) para realizar cada juego de competición al día.
- Para subir de nivel en el juego, se necesitan superar 3 de los 5 pases diarios. Todos juegos tienen 10 niveles por donde escalar.
- Si se entra en un día distinto al actual al juego, los pases diarios se renuevan, obteniendo 5 nuevos intentos.

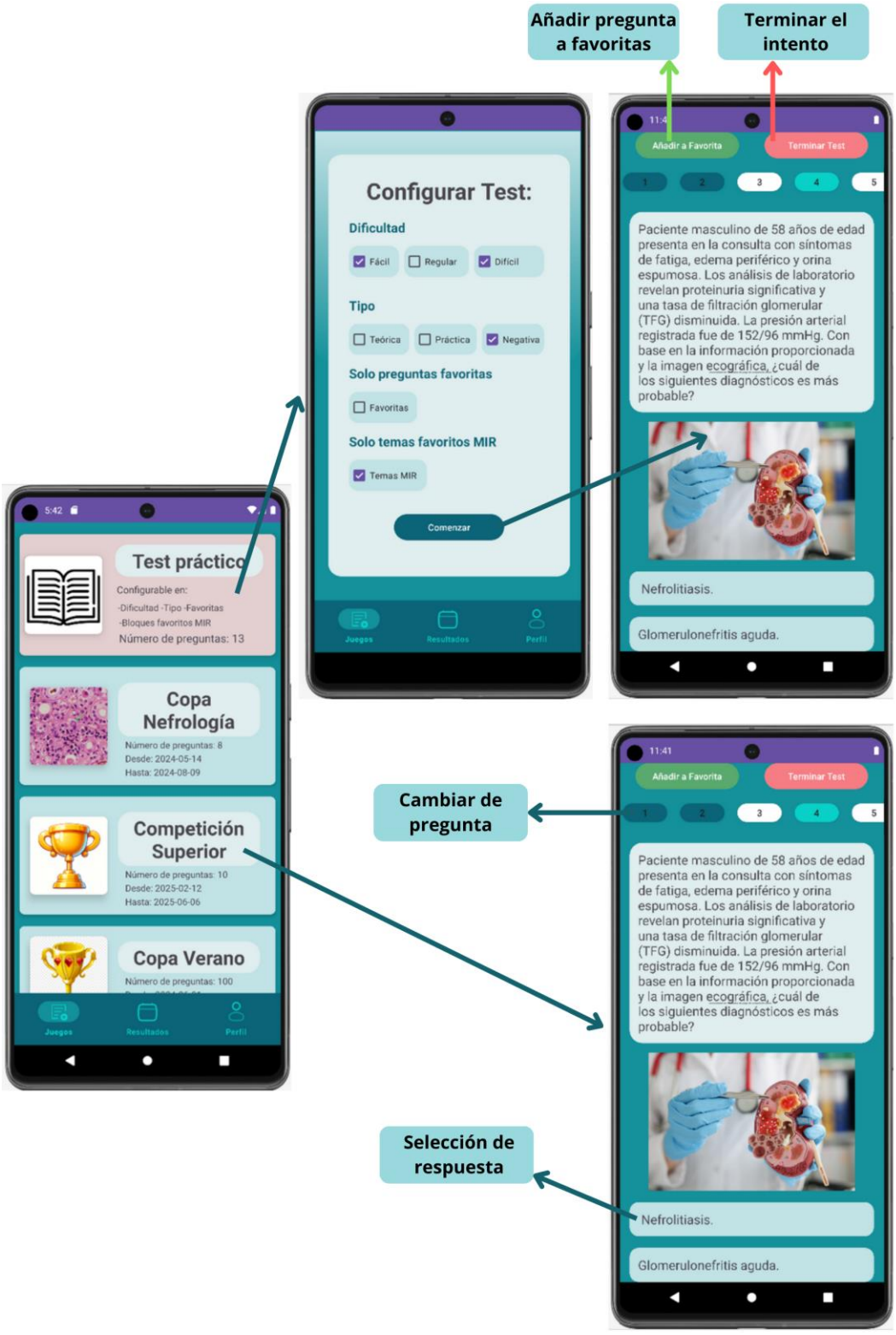

Figura 120 Esquema manual usuario acciones en vista de juegos

-Resultados

El usuario con la sesión iniciada puede pulsar en el botón de la barra de navegación inferior de la aplicación móvil para acceder a la sección de resultados, en ella se encuentra la información relativa a los intentos de los juegos.

En la parte superior se puede ver una gráfica con los resultados del juego teórico, en el eje Y representa la puntuación obtenida en cada intento y el eje X representa cada intento realizado al pulsar en el juego teórico y realizar una autoevaluación.

En la parte inferior, el usuario puede ver los resultados de competiciones iniciadas, su nivel alcanzado para cada una y su información actual de los pases, es decir, el número de ellos que se han realizado en el día y cuántos se han superado.

En la parte superior aparece un botón de visualizar las preguntas que han sido añadidas como favoritas. El usuario puede pulsar este botón iniciando una navegación a la vista de las preguntas favoritas para poder repasarlas.

La vista de las preguntas favoritas muestra el enunciado, las respuestas, la foto si contiene, y la respuesta correcta.

El usuario que no tiene la sesión iniciada, al acceder a esta sección de resultados solo podrá ver los títulos de las secciones y no podrá visualizar ningún resultado de los juegos.

Todo esto que se acaba de mencionar se puede observar en la Figura 121.

Figura 121 Esquema navegación grupo resultados móvil

-Perfil

El usuario puede acceder a la funcionalidad del perfil, pulsando sobre el botón derecho de la barra de navegación inferior.

Si el usuario no tiene la sesión iniciada, entonces podrá iniciar sesión introduciendo su correo y contraseña o registrándose, introduciendo su nombre, apellidos, curso, universidad, correo y contraseña.

Si el usuario se registra o inicia sesión, puede acceder a la visualización de los datos de su perfil, ver su nombre, apellidos, universidad, curso y correo. En la vista del perfil el usuario puede editar los campos de su perfil mencionados, a excepción del correo, así como eliminar su cuenta o cerrar la sesión.

Estas acciones posibles quedan detalladas en la siguiente imagen (Figura 122).

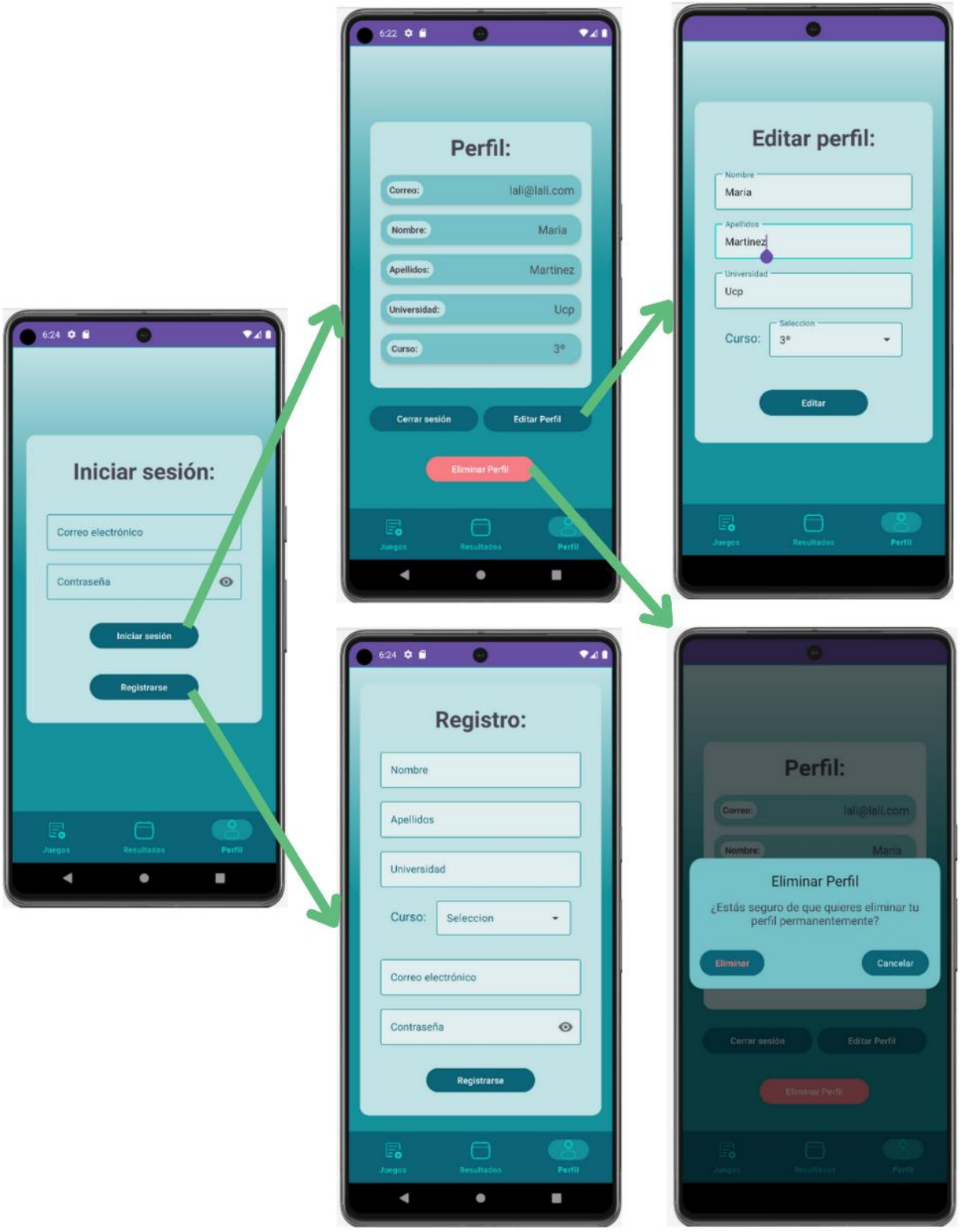

Figura 122 Esquema navegación usuario móvil por perfil