UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Informática

# Una aplicación web para la realización de ejercicios basada en gamificación

Trabajo Fin de Grado

Grado en Ingeniería Informática

AUTOR/A: Marin, Delia Andreea

Tutor/a: Vidal Oriola, Germán Francisco

CURSO ACADÉMICO: 2023/2024

# Resum

La gamificació ha transformat nombrosos sectors, inclòs l'educatiu, on s'ha demostrat ser una eina efectiva per a augmentar la motivació i el rendiment dels estudiants mitjançant l'ús d'elements lúdics com punts, nivells i insígnies. Aquest enfocament no sols millora la participació i assistència a classe, sinó que també contribueix a una major equitat en els resultats acadèmics, reduint les disparitats entre els estudiants.

En aquest context, el propòsit d'aquest treball de fi de grau és el desenvolupament d'una aplicació web que permeta tant a professors com a estudiants gestionar i resoldre tasques educatives de forma gamificada, destinades a preparar als estudiants per a diverses avaluacions acadèmiques. L'aplicació busca fomentar la pràctica contínua i la competitivitat sana entre els estudiants, oferint-los la possibilitat de verificar la correcció de les seues respostes i aprendre de les solucions aportades pels seus companys.

L'objectiu principal és crear un entorn interactiu on es puga implementar un sistema de puntuació que reforce els objectius de la gamificació, assegure als estudiants la certesa sobre la correcció de les seues respostes, i millore la col·laboració per a resoldre dubtes acadèmics.

**Paraules clau:** aplicación, web, Spring Boot, React, MySQL, gamificació, educació, Arquitectura per capes, Arquitectura client-servidor

# Resumen

La gamificación ha transformado numerosos sectores, incluyendo el educativo, donde se ha demostrado ser una herramienta efectiva para aumentar la motivación y el rendimiento de los estudiantes mediante el uso de elementos lúdicos como puntos, niveles e insignias. Este enfoque no solo mejora la participación y asistencia en clase, sino que también contribuye a una mayor equidad en los resultados académicos, reduciendo las disparidades entre estudiantes.

En este contexto, el propósito de este trabajo de fin de grado es el desarrollo de una aplicación web que permita tanto a profesores como a estudiantes gestionar y resolver tareas educativas de forma gamificada, destinadas a preparar a los estudiantes para diversas evaluaciones académicas. La aplicación busca fomentar la práctica continua y la competitividad sana entre los estudiantes, ofreciéndoles la posibilidad de verificar la corrección de sus respuestas y aprender de las soluciones aportadas por sus compañeros.

El objetivo principal es crear un entorno interactivo donde se pueda implementar un sistema de puntuación que refuerce los objetivos de la gamificación, asegure a los estudiantes la certeza sobre la corrección de sus respuestas, y mejore la colaboración para resolver dudas académicas.

**Palabras clave:** aplicación, web, Spring Boot, React, MySQL, gamificación, educación, Arquitectura por capas, Arquitectura cliente-servidor

# Abstract

Gamification has transformed numerous sectors, including education, where it has proven to be an effective tool for increasing student motivation and performance through the use of playful elements such as points, levels, and badges. This approach not only improves participation and attendance in class but also contributes to greater equity in academic outcomes, reducing disparities among students.

In this context, the purpose of this thesis is the development of a web application that allows both teachers and students to manage and solve educational tasks in a gamified manner,

aimed at preparing students for various academic assessments. The application seeks to encourage continuous practice and healthy competition among students, offering them the opportunity to verify the correctness of their answers and learn from the solutions provided by their peers.

The main goal is to create an interactive environment where a scoring system can be implemented that reinforces the objectives of gamification, assures students of the accuracy of their responses, and improves collaboration to resolve academic queries.

**Key words:** application, web, Spring Boot, React, MySQL, gamificaton, education, Layered Architecture, Client-Server Architecture

# Índice general

# Índice de figuras

# Índice de tablas

# CAPÍTULO 1
# Introducción

La gamificación es una técnica de aprendizaje que consiste en aplicar a la enseñanza las dinámicas de competencia y recompensa de los juegos, con el objetivo de proporcionar a los alumnos mayor motivación para progresar y obtener mejores resultados en el área de estudio tratado, haciendo que se involucren más en el proceso consiguiendo así un mayor entendimiento y retención de la información. Además de ayudar en el aprendizaje, también puede fomentar el desarrollo de otras habilidades como el trabajo en equipo, la comunicación y la colaboración entre los estudiantes.

## 1.1 Motivación

La gamificación se lleva aplicando mucho tiempo en el mundo empresarial como una forma de motivar a los empleados a través de incentivos y recompensas en el trabajo. Si bien el concepto fue creado en 2002 por *Nick Pelling* [1], un programador de juegos, en las últimas décadas se ha extendido a la educación, centrándose al principio en la mecánica de juegos, como las recompensas y los puntos, y después evolucionando para incluir elementos más sofisticados, convirtiéndose así en una herramienta más efectiva para motivar a los estudiantes.

La aplicación de la gamificación en la educación puede resultar más efectiva que los métodos tradicionales para ayudar a los estudiantes a desarrollar distintas habilidades, mejorar el aprendizaje y aumentar su motivación. Esto se ve respaldado por varios estudios donde se investiga el efecto de aplicar en la educación alguna técnica de juego concreta como puntos, niveles o insignias, donde la mayoría revelan buenos resultados, entre los que cabe destacar tanto un aumento en la asistencia y la participación en el aula, así como una disminución en la diferencia de las notas obtenidas entre los alumnos [2].

Dentro de la gamificación, el uso de la competitividad puede ser una herramienta efectiva para motivar más aún a los estudiantes. Al incorporar elementos de juego competitivos en la gamificación, los educadores pueden motivar a los estudiantes a superarse a sí mismos y a sus compañeros, completando así un mayor número de tareas y mejorando su rendimiento.

En este contexto, la tecnología juega un papel fundamental, ya que permite la creación de plataformas y herramientas digitales que facilitan la aplicación de la gamificación en la educación. En particular, las aplicaciones web ofrecen un amplio abanico de posibilidades para la creación de entornos gamificados, donde los estudiantes pueden interactuar con los contenidos y aprender de manera lúdica, en un ambiente interactivo, visual y personalizado. Además, los datos generados por estas tecnologías permiten a los profesores identificar áreas problemáticas en el aprendizaje y adaptar la enseñanza para mejorar los resultados.

## 1.2 Objetivos

El objetivo de este proyecto es desarrollar una aplicación que permita a profesores y alumnos la gestión de problemas y tareas que no tienen una solución única y están pensadas para la preparación del alumnado de cara a los distintos actos de evaluación de las asignaturas del curso, con el añadido de incentivar a los alumnos a practicar lo máximo posible al verse motivados por cierta competitividad. Otro objetivo es el de ofrecer a los alumnos la posibilidad de comprobar si sus soluciones son correctas al mismo tiempo que pueden obtener un punto de vista diferente a partir de las soluciones de sus compañeros.

Para lograr los objetivos deseados, será fundamental tener en mente durante las fases de diseño, implementación y despliegue de esta aplicación las siguientes cuestiones:

- Incorporar un sistema de puntuación que cumpla con éxito los objetivos de la gamificación.
- Garantizar a los alumnos la eliminación de la incertidumbre a la hora de comprobar la solución propuesta a los distintos problemas.
- Mejorar la colaboración entre los alumnos para la resolución de posibles dudas sobre los contenidos de la asignatura.
- Finalizar un *MVP*[1] del proyecto que permita hacer un uso completo de las funcionalidades principales mediante la correcta realización de todas las operaciones CRUD[2] sobre los elementos principales.

Por otra parte, cabe destacar algunos objetivos de carácter personal que se cumplirán con la finalización de este proyecto:

- Profundización en la creación de servicios *RESTful* al mismo tiempo que se adquiere conocimiento sobre Spring Boot, el framework más utilizado para el principal lenguaje de programación utilizado en la carrera, Java.
- Aplicación de lo aprendido sobre realización de consultas con el lenguaje SQL en un proyecto real utilizando un sistema de gestión de bases de datos relacional.
- Indagar en técnicas de pruebas de software para facilitar el mantenimiento de la aplicación, aplicando cuidadosamente todas las pruebas necesarias.

## 1.3 Metodología

En términos generales, las metodologías de desarrollo de software buscan obtener la mejor organización y eficiencia en la comunicación con los clientes y el desarrollo del producto a través de un equipo de desarrollo. Dado que en este caso el desarrollo es realizado por una única persona y no existe un cliente final, se adaptará y reutilizará principios de alguna de estas.

### 1.3.1. Desarrollo ágil de software

Las metodologías ágiles[3] consisten en la entrega continua de piezas de software funcionales creadas en una serie de iteraciones cortas. A lo largo de estas iteraciones se colabora

[1] Producto viable mínimo. Un producto con suficientes características para satisfacer a los clientes iniciales
[2] Crear, Leer, Actualizar y Borrar.

constantemente con el cliente final para mejorar el producto en cada iteración. De esta manera se consigue una mayor flexibilidad pudiendo realizar cambios en cualquier fase del ciclo de desarrollo del software de manera fácil y rápida, adaptándose así a las nuevas necesidades del cliente que puedan surgir a raíz de la mejor comprensión del producto entregado en cada iteración.

### 1.3.2. Scrum

*SCRUM* [4] es uno de los marcos de trabajo para el desarrollo ágil más utilizado. En *SCRUM* los equipos son pequeños y las iteraciones, llamadas sprints, tienen una duración de entre una y cuatro semanas. Durante estos sprints, los equipos tienen participación en la totalidad de la organización del proyecto, donde cada miembo se encarga de unas tareas determinadas, siendo conscientes del trabajo que están realizando el resto, por lo que tienen una visión y un entendimiento mayor del proyecto. Además, los miembros se comunican diariamente para mantenerse informados y asegurarse de la correcta adaptación y priorización de los requisitos deseados por el cliente, de manera que se entregue un producto de máxima calidad.

Para definir el proceso de trabajo, se establece un conjunto de eventos, roles y artefactos que se definen a continuación.

#### Roles

- **Product Owner**: Es la persona encargada de establecer toda la comunicación con el cliente para definir y priorizar los requisitos utilizando para ello el *Product Backlog*.
- **Scrum Master**: Es la persona encargada de la gestión de personas, asegurando la correcta implementación de los principios y procesos *scrum*.
- **Desarrolladores**: Las personas encargadas de llevar a cabo el ciclo de vida de desarrollo del software durante los *sprints*.

#### Artefactos

- **Product Backlog**: Listado de requisitos con un tiempo estimado y ordenados por la prioridad acordada entre el *Product Owner* y el cliente.
- **Sprint Backlog**: Listado de tareas extraídas del *Product Backlog* y elegidas por los desarrolladores que se deben desarrollar durante el *sprint*.
- **Incremento**: Pieza de software resultado de completar todas las tareas de los *sprint* backlogs anteriores.

#### Eventos

- **Sprint Planning**: La reunión al comienzo del *sprint* en la que los desarrolladores establecen el *Sprint Backlog*.
- **Daily Meeting**: Reunión diaria entre desarrolladores y *Scrum Master* de menos de 15 minutos en la que se explica el trabajo hecho el día anterior, el planeado para ese día y los problemas encontrados.
- **Sprint Review**: Reunión al final del *sprint* en la que el *Product Owner* presenta el incremento al cliente. En base a esta reunión, el *Product Owner* actualizará el *Product Backlog* pertinentemente.
- **Sprint Retrospective**: Reunión anterior al comienzo del siguiente *sprint* en la que se lleva a cabo una evaluación del *sprint* anterior y se propone mejoras para aplicar en el siguiente.

### 1.3.3. Adaptación

En este proyecto, el cliente serán los profesores y alumnos que hagan uso de la aplicación. Por tanto, podemos asumir nosotros todos los roles. De esta manera, podemos llevar a cabo el rol de *Product Owner* definiendo el producto final clara y detalladamente nosotros mismos. Igualmente, podemos ser estrictos con nuestro trabajo y buscar la manera de solucionar los problemas que nos encontremos, haciendo de *Scrum Master* propio. Finalmente, nos encargaremos de desarrollar la aplicación durante los diferentes *sprints*.

Por otra parte, se mantendrán todos los artefactos de la misma manera que en un equipo *scrum* normal, con la diferencia de que todas las tareas serán asignadas a una única persona. Los eventos también se pueden mantener, ya que podemos simular una pequeña reunión dedicando cierto tiempo a evaluar de manera crítica nuestro trabajo, a excepción del *Daily Meeting* ya que no hay nadie a quien debamos explicar nuestro progreso.

#### Jira

Para hacer la gestión del *Product Backlog* y los *Sprint Backlogs* utilizaremos Jira[5]. Se trata de una herramienta que nos permite consultar los datos referentes a nuestro proyecto de forma rápida e intuitiva, con una versión gratuita suficiente para este proyecto. Incluye una plantilla para *Scrum* que permite establecer *sprints* con sus respectivos backlogs. Estos están compuestos por diferentes tipos de entidades estimadas y priorizadas, llamadas incidencias, que permiten organizar y jerarquizar los items que representarán tareas y requisitos. Además, también incluye una hoja de ruta y un tablero con un flujo de trabajo personalizable.

## 1.4 Estructura de la memoria

La memoria se divide en distintos capítulos. A continuación se resumen brevemente los posteriores a este:

- **Capítulo 2. Estudio estratégico:** se realiza un análisis de las diferentes soluciones relacionadas con el aprendizaje y la gamificación existentes en el mundo digital, con el objetivo de facilitar la identificación de requisitos.
- **Capítulo 3. Análisis del problema:** se detalla el listado de los requerimientos acordados para completar el proyecto mediante diferentes técnicas de especificación de requisitos.
- **Capítulo 4. Diseño de la solución:** se exponen las decisiones que se han tomado en cuanto a la arquitectura y tecnologías empleadas para el sistema.
- **Capítulo 5. Desarrollo de la solución:** se entra más en detalle en cuanto al funcionamiento de las tecnologías explicadas anteriormente en nuestra aplicación específica y las diferentes herramientas utilizadas para facilitar el desarrollo.
- **Capítulo 6. Despliegue y mantenimiento:** se expone el proceso necesario para llevar a cabo el mantenimiento y despliegue final de la aplicación.
- **Capítulo 7. Manual de uso:** se expone una demostración del resultado final a través de capturas de pantalla.
- **Capítulo 8. Conclusiones y trabajo futuro:** se evalúan los resultados obtenidos tras completar el trabajo y potenciales mejoras para futuras versiones del sistema.

# CAPÍTULO 2
# Estudio estratégico

Los profesores cada vez hacen más uso de herramientas tecnológicas para gestionar de manera más eficiente sus clases o para hacerlas más amenas y divertidas para sus alumnos. Por ello, podemos encontrar multitud de aplicaciones destinadas no solo a ser usadas en el aula, sino también a mejorar el proceso de aprendizaje de aquellos que lo tengan como un objetivo personal.

## 2.1 Aprendizaje a través de la tecnología

Para extraer los requisitos de nuestra aplicación necesarios para el cumplimiento de los objetivos que nos conciernen, nos centraremos únicamente en aquellas que estén enfocadas al uso en el aula o que contengan algún sistema de recompensa similar al que buscamos.

Por un lado, estudiaremos aplicaciones cuyo uso principal es la creación de tareas por parte de los profesores y la entrega de estas por parte de sus alumnos.

Por otro lado, observaremos aplicaciones gamificadas, que incluyen un sistema de puntuación o recompensas, no necesariamente centradas en el uso en el aula.

### 2.1.1. Hackerrank

Hackerrank es una plataforma de resolución de problemas de programación competitiva utilizada tanto por desarrolladores para preparación de entrevistas de trabajo o competiciones, como por empresas para la contratación de estos.

Como se puede observar en la figura 2.1 dispone de distintos conjuntos de problemas diseñados por habilidad o tiempo de preparación, dentro de los que se pueden filtrar los problemas disponibles por subtemas, dificultad, etc.

Lo interesante de esta plataforma es el método de gamificación que propone, basado en un sistema de puntuación y recompensas que se puede observar en la figura 2.2. Este sistema consiste en la posibilidad de obtener distintas insignias para cada una de las habilidades disponibles. Estas a su vez tienen distintos niveles representados por estrellas que se consiguen ganando puntos al resolver problemas, pudiendo obtener más o menos dependiendo de la dificultad de estos.

También incluye la posibilidad de ver las soluciones propuestas por otras personas en un panel de discusiones así como un leaderboard por tarea, como vemos en la figura 2.3.

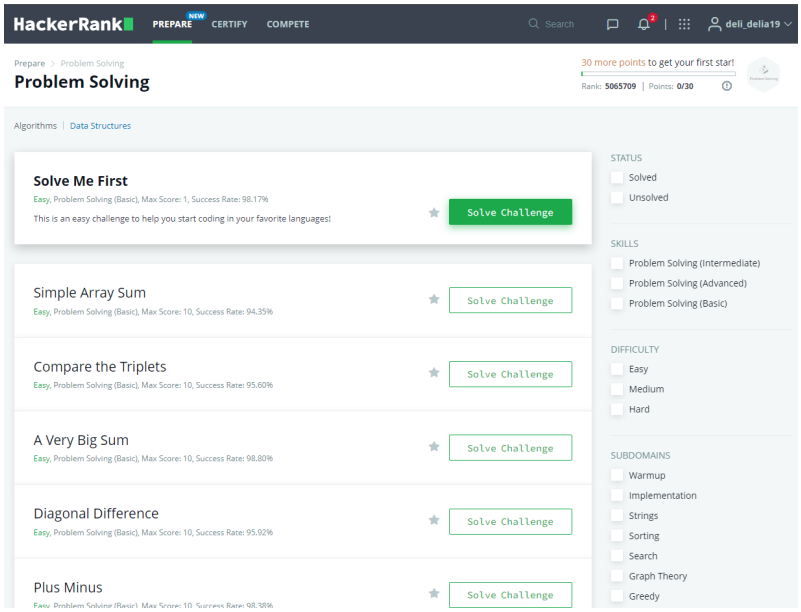



**Figura 2.1:** Pantalla de listado de problemas

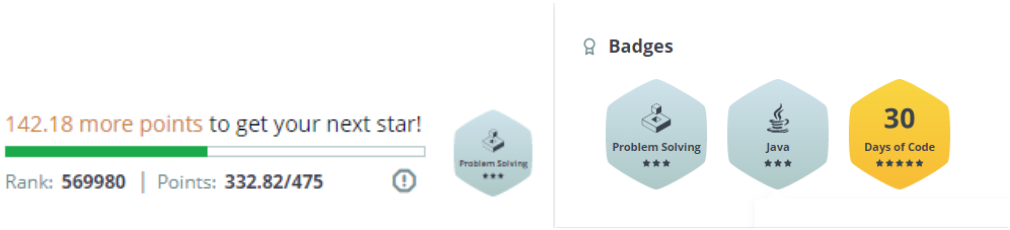



**Figura 2.2:** Sistema de puntuación, estrellas e insignias.

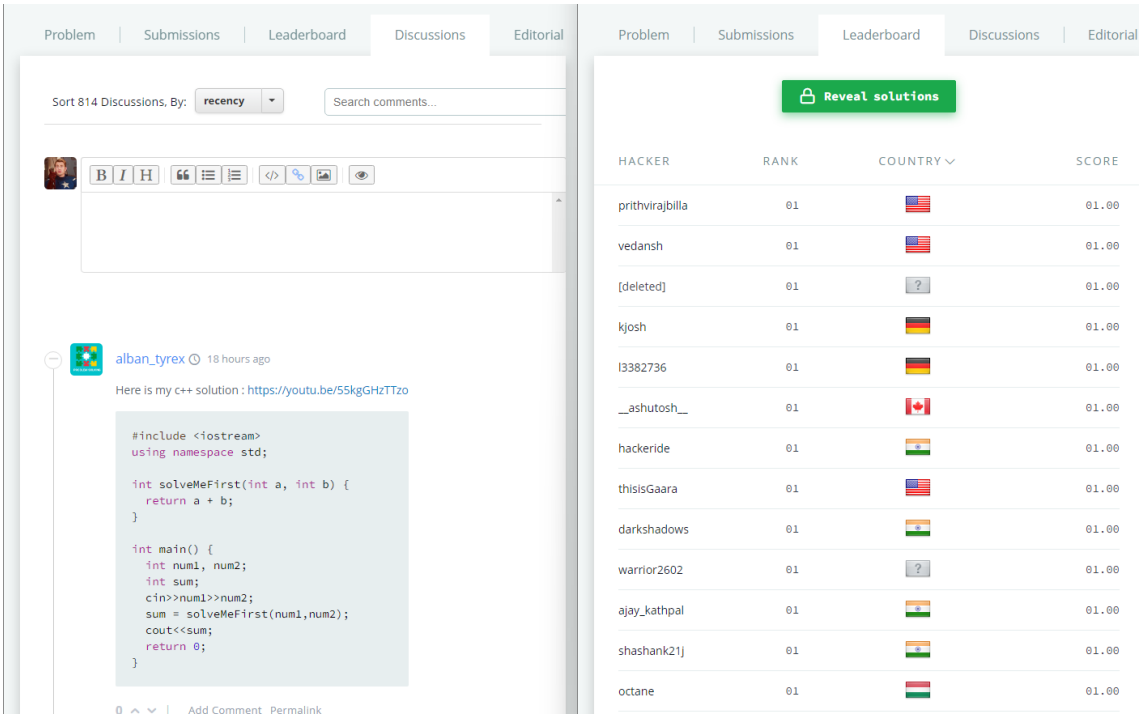



**Figura 2.3:** Panel de discusiones y leaderboard

### 2.1.2. Duolingo para escuelas

Duolingo es una de las aplicaciones más utilizadas para el aprendizaje de idiomas no solo debido a que su diseño interactivo e intuitivo hace divertido el estudio, sino también al componente de gamificación que propone.

Duolingo genera un recorrido de ejercicios a realizar para ganar experiencia y desbloquear nuevos niveles a medida que se avanza en el conocimiento del idioma, ofreciendo además la opción de establecer una meta diaria para mantener la motivación.

La versión para escuelas que vemos en la figura 2.4 permite el uso en el aula de la aplicación, brindando a los profesores la oportunidad de crear objetivos de experiencia a ser completados dada una fecha límite, y hacer un seguimiento del avance de sus alumnos, pudiendo detectar así las necesidades de cada uno. Los alumnos pueden ver el top 3 de alumnos en su clase y sus tareas pendientes.

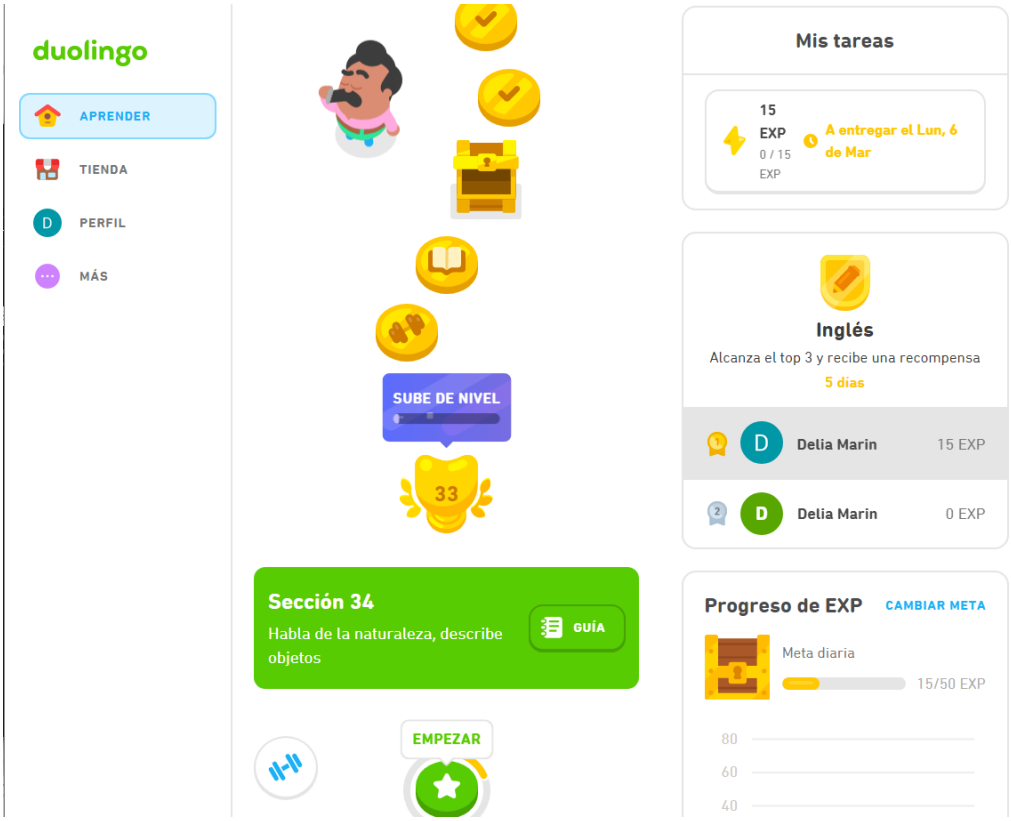



**Figura 2.4:** Pantalla principal del alumno

### 2.1.3. Google Classroom

Google Classroom es una aplicación web que ofrece a los profesores la posibilidad de crear clases en las que interactuar con sus alumnos a través de un tablón, visible en la figura 2.5, donde pueden publicar anuncios y tareas o responder a los alumnos, que también pueden verlo.

Una de las mejores características es la integración con las distintas herramientas de Google, de manera que las tareas se pueden tanto crear, corregir o entregar adjuntando presentaciones, hojas de cálculo, formularios, o cualquier otro tipo de documento de Google Drive con todas las funcionalidades que estas ofrecen como añadir comentarios, etc. En la figura 2.6 tenemos un ejemplo de creación de una tarea.

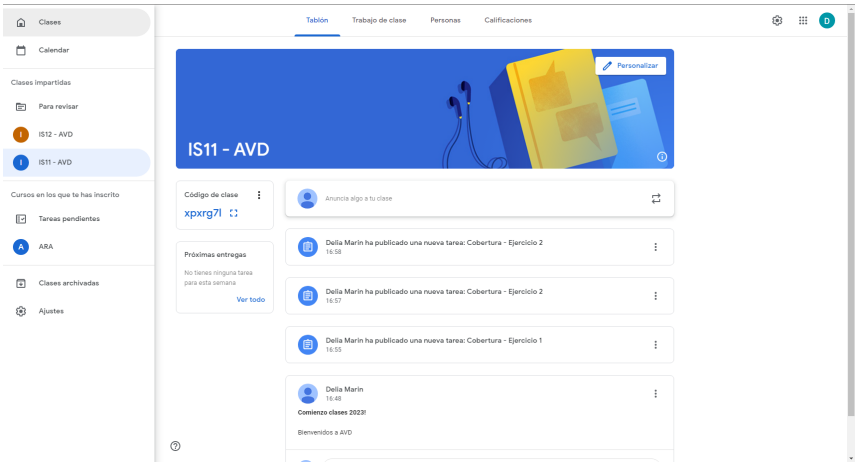



**Figura 2.5:** Pantalla principal. Las vistas del profesor y del alumno son prácticamente idénticas

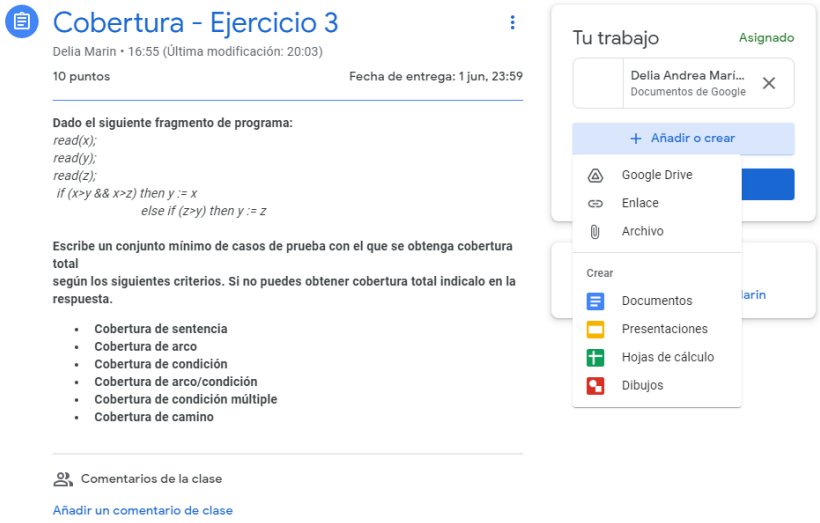



**Figura 2.6:** Pantalla de entrega de tarea

Las tareas se pueden agrupar por temas y, además, los alumnos también pueden filtrarlas en función de si están asignadas, devueltas o sin entregar.

Los profesores, por su parte, pueden asignar una puntuación a cada tarea, enviando comentarios y correciones sobre los documentos adjuntos, así como ver las calificaciones de cada tarea por cada alumno y la media de la clase. Sin embargo, no hay ningún sistema de gamificación.

## 2.2 Análisis de las aplicaciones

Tras analizar las aplicaciones más relevantes en relación a la gamificación y entrega de tareas, podemos encontrar ciertas características que resultan interesantes para introducir en nuestra solución y a los objetivos planteados. También es importante considerar aquellas características que no aportan un valor significativo a nuestra solución y no merecen ser incluidas, así como otras faltantes que sí deberían ser incluidas. En la siguiente tabla se presentan las características seleccionadas, junto con su presencia en las distintas aplicaciones estudiadas y su inclusión en nuestra solución.

| **Característica** | **Hackerrank** | **Google Classroom** | **Duolingo for School** | **Incluida** |
|---|---|---|---|---|
| Tipos de usuarios | Estudiante | Estudiante / Profesor | Estudiante / Profesor | Estudiante / Profesor |
| Filtrado de tareas | ✓ | ✓ | ✓ | ✓ |
| Sistema de puntuación | ✓ | ✓ | ✓ | ✓ |
| Insignias | ✓ | ✗ | ✗ | ✗ |
| Ranking | ✓ | ✗ | ✓ | ✓ |
| Niveles de dificultad | ✓ | ✗ | ✗ | ✗ |
| Colaboración para dudas | ✓ | ✓ | ✗ | ✓ |
| Creación de tareas | ✓ | ✓ | ✓ | ✓ |
| Organización por temas | ✓ | ✓ | ✗ | ✓ |
| Variedad de clases | ✗ | ✓ | ✓ | ✓ |
| Ver solución propia | ✓ | ✓ | ✗ | ✓ |
| Ver otras soluciones | ✓ | ✗ | ✗ | ✓ |
| Ver solución profesor | ✓ | ✗ | ✓ | ✗ |
| Corrección con comentarios | ✗ | ✓ | ✗ | ✓ |
| Correción sobre archivos | ✗ | ✓ | ✗ | ✗ |
| Sistema de monedas | ✓ | ✗ | ✗ | ✗ |
| Calendario | ✗ | ✓ | ✗ | ✗ |
| Estadísticas | ✗ | ✗ | ✓ | ✗ |
| Objetivos diarios | ✗ | ✗ | ✓ | ✗ |
| Logros | ✗ | ✗ | ✓ | ✗ |
| Puntuación entre alumnos | ✗ | ✗ | ✗ | ✓ |

**Tabla 2.1:** Tabla de características concluidas

## 2.3 Propuesta

Para cumplir estos objetivos los requerimientos fundamentales de la aplicación los podemos definir en función de dos tipos distintos de usuario:

- **Profesor**
  Gestionará y aportará los problemas y sus soluciones. Comprobará que las calificaciones asignadas a los alumnos están fundamentadas.
- **Alumno**
  Aportará soluciones a los problemas deseados y evaluará las soluciones de otros compañeros. Podrá tanto realizar preguntas como aportar respuestas a aquellas hechas por otros compañeros.

Por otra parte, cabe destacar las principales vistas de la aplicación que tendrán disponibles los alumnos:

- **Problemas disponibles**
  Listado de todos los problemas disponibles.
- **Preguntas disponibles**
  Listado de todas las preguntas realizadas por los alumnos sobre el temario de la asignatura o los problemas disponibles.
- **Soluciones de otros compañeros**
  Listado de alumnos que han solucionado un problema seleccionado con posibilidad de acceder a la solución propuesta para visualizarla y posteriormente proporcionar una evaluación de la solución.
- **Ranking**
  *Ranking* basado en la puntuación total obtenida por los alumnos del curso actual.

# CAPÍTULO 3
# Análisis del problema

La documentación de los requisitos de software es una de las partes más importantes del ciclo de vida del software. Según el estándar *Iso/Iec/IEEE 29148:2018* [6] permite hacer una estimación de costes, riesgos y plazos, reduce el rediseño posterior y proporciona una base para la mejora del producto, entre otras ventajas. En este capítulo se detallarán los requisitos de la aplicación, haciendo uso de distintas técnicas de especificación de requisitos.

## 3.1 Especificación de requisitos

Para especificar los requisitos de nuestro sistema utilizaremos técnicas y lenguajes de especificación informales y semi-formales. De esta manera, gracias al lenguaje natural se evita la necesidad de un entrenamiento especial para el correcto entendimiento de las funcionalidades del sistema, y mediante técnicas semi-formales se solventan algunos problemas que este supone como la estructuración e identificabilidad pobre, la ambigüedad o el ruido.

### 3.1.1. Límites del sistema

Para representar los límites del sistema y los actores que interactuan con el mismo utilizamos un diagrama de contexto.

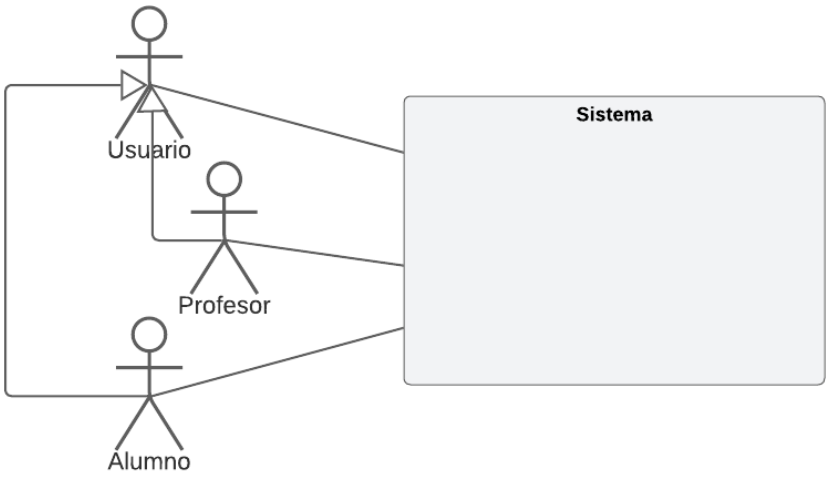



**Figura 3.1:** Diagrama de Contexto

Se observa que el usuario que interactúa con el sistema puede ser de dos tipos: alumno o profesor. Estos tendrán vistas del sistema distintos, pero con muchas funciones en común. Estos dos tipos de usuarios tendrán vistas del sistema distintas, adaptadas a sus necesidades y roles específicos dentro del punto de la aplicación en el que se encuetren, pero compartirán muchas funciones en común, asegurando una experiencia de usuario coherente y eficiente.

### 3.1.2. Modelo de Dominio

Para establecer los conceptos importantes relacionados al contexto del sistema y las relaciones entre estos conceptos utilizamos un modelo de dominio representado con un diagrama de clases UML. Este modelo es crucial para tener una visión base de cómo se estructurarán los datos y cómo interactuarán los diferentes componentes del sistema.

#### 3.1.2.1. Definiciones

A continuación se definen algunos de los atributos y conceptos que contienen las clases del modelo.

| Clase | Atributo | Descripción |
|---|---|---|
| Alumno | Puntuación | La puntuación que el usuario ha obtenido mediante la entrega de soluciones, corrección de soluciones y respuestas correctas a preguntas. |
| Solución | Calificación | Calificación calculada en base a las calificaciones asignadas en las correcciones. |
| Corrección | Calificación | Calificación que el alumno considera asignar a la solución, entre los tres atributos de la clase 'Clasificación' ordenados de mayor a menor. |
| EstadoTarea | Entregada | Únicamente visible por usuarios que son alumnos. Indica que el usuario ya ha propuesto una solución para dicha tarea. |
| EstadoTarea | Pendiente | Indica que la fecha de entrega de la tarea es mayor que la fecha actual y por tanto aún se pueden proponer soluciones. |
| EstadoTarea | Cerrada | Indica que la fecha de entrega de la tarea es menor que la fecha actual y por tanto ya no se pueden proponer soluciones. |
| EstadoPregunta | Resuelta | Indica que la pregunta ya contiene una respuesta marcada como correcta. |
| EstadoPregunta | Pendiente | Indica que la pregunta aún no contiene una respuesta marcada como correcta. |

**Tabla 3.1:** Tabla de definición de atributos

#### 3.1.2.2. Diagrama

En el Modelo de Dominio únicamente nos centraremos en los aspectos relevantes para entender el dominio, por tanto utilizaremos las relaciones y atributos estrictamente necesarios para tener una mayor claridad.

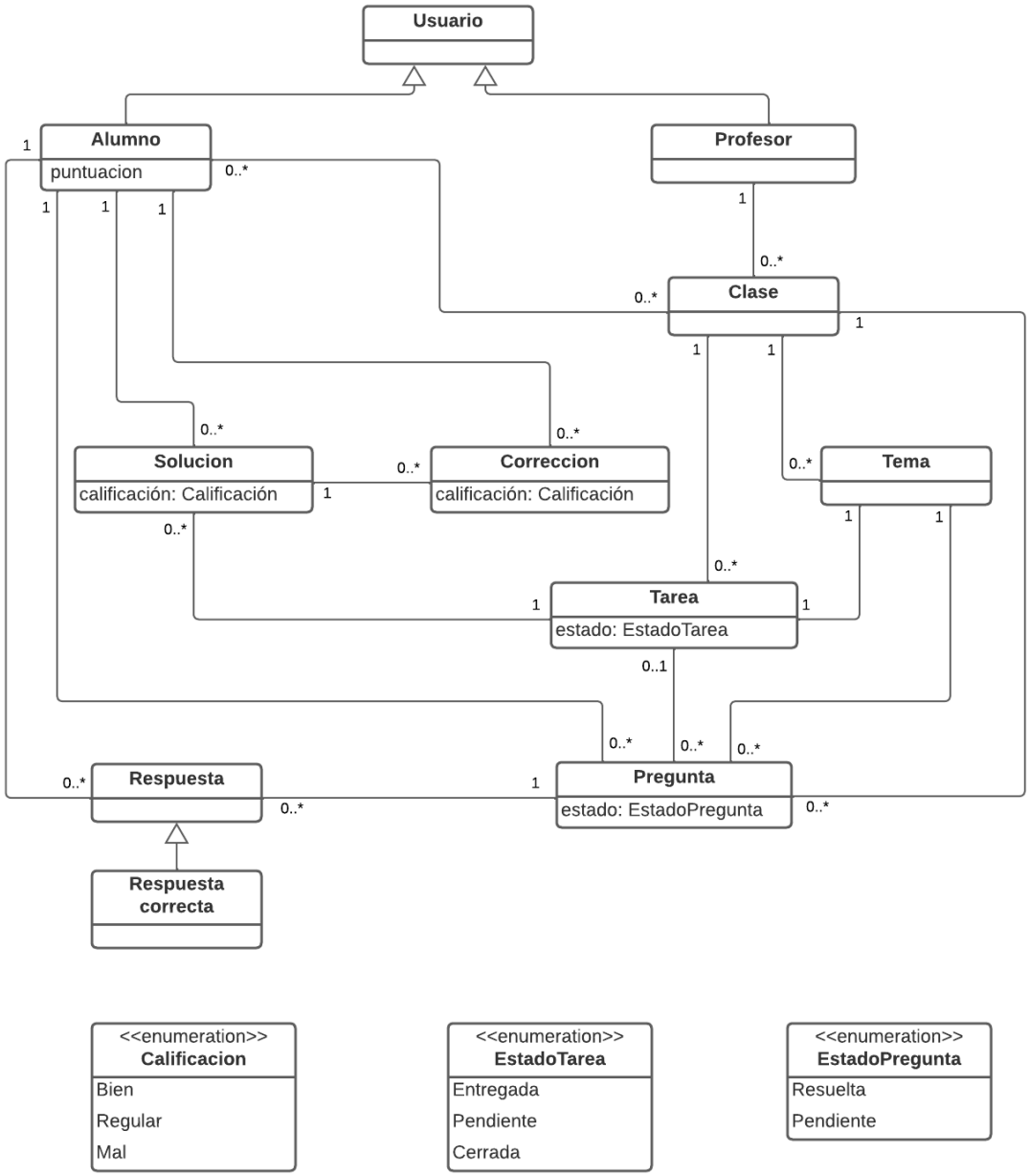



**Figura 3.2:** Modelo de Dominio

### 3.1.3. Diagramas de casos de uso

Los requisitos funcionales que se van a implementar en el sistema se pueden recoger en casos de uso. Los vamos a agrupar por las vistas desde las que se podrán acceder.

#### 3.1.3.1. Diagrama: Autenticación

En primer lugar, los casos de uso referentes a la autenticación.

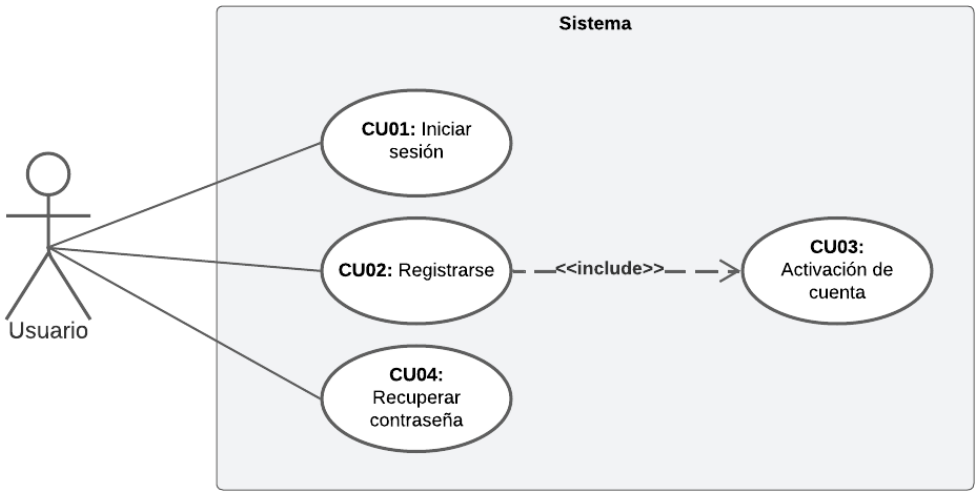



**Figura 3.3:** Diagrama de casos de uso de autenticación

#### 3.1.3.2. Diagrama: Navegación principal

Los casos de uso a los que se puede acceder desde el menú de navegación en cualquier vista de la aplicación en la que se encuentre el usuario.

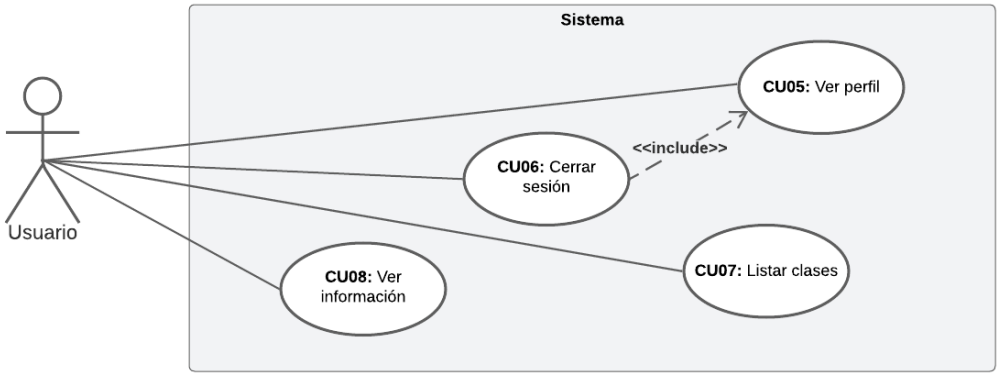



**Figura 3.4:** Diagrama de casos de uso de navegación principal

#### 3.1.3.3. Diagrama: Vista de clases

Los casos de uso disponibles en la vista a la que se accede desde el caso de uso “Listar clase“ en el menú de navegación.

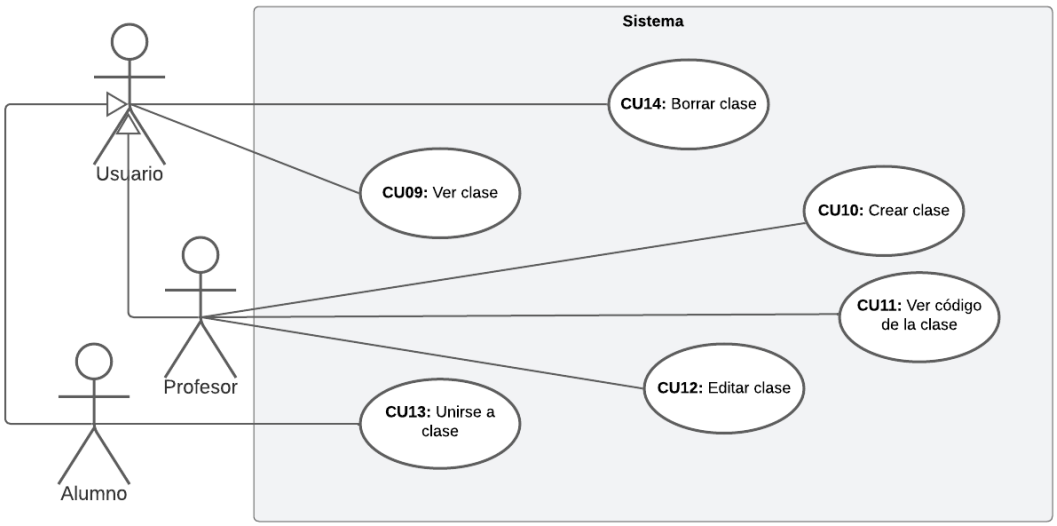



**Figura 3.5:** Diagrama de casos de uso de la vista de clases

#### 3.1.3.4. Diagrama: Vista de una clase

Los casos de uso visibles en la vista de una clase a la que se accede al seleccionarla en el listado de clases de la anterior vista.

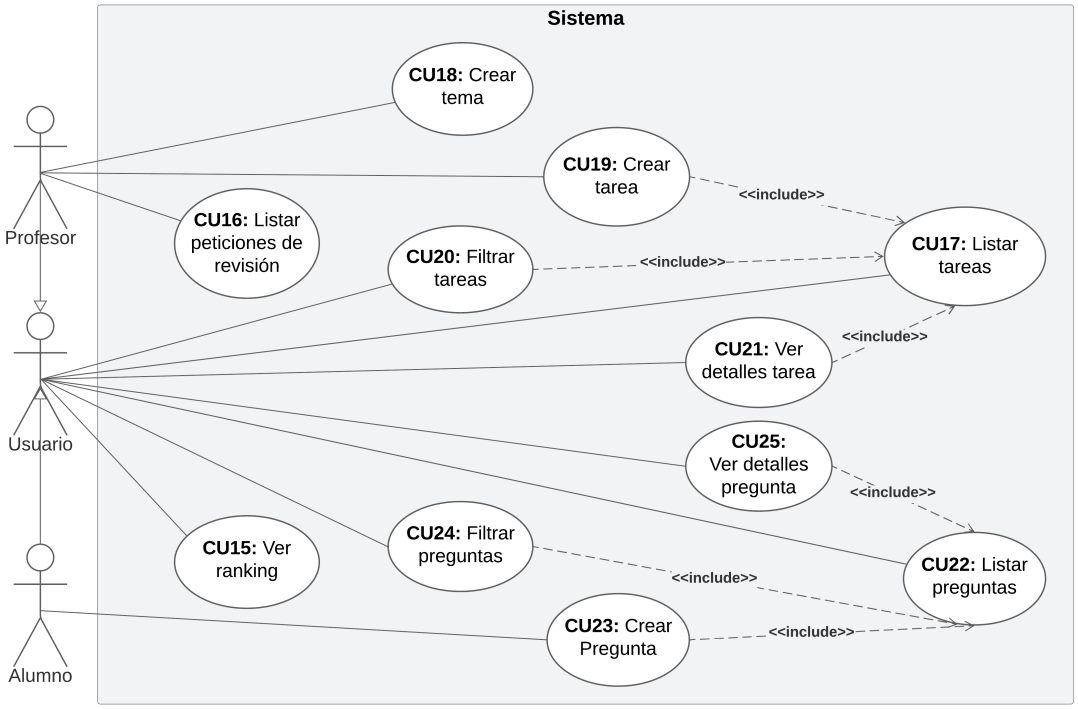



**Figura 3.6:** Diagrama de casos de uso de la vista de una clase

#### 3.1.3.5. Diagrama: Vista de una pregunta

Los casos de uso visibles en la vista de una pregunta, a la que se accede con el caso de uso 'Ver detalles pregunta' al seleccionar una pregunta del listado de preguntas en la vista anterior.

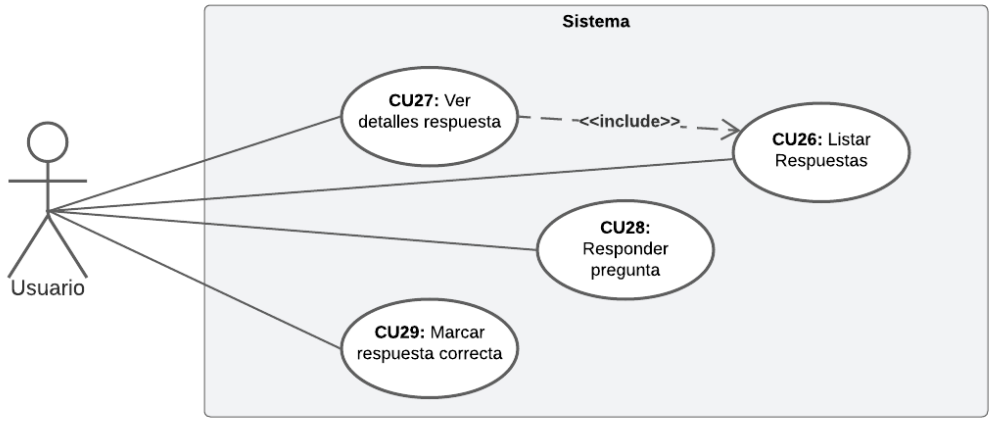



**Figura 3.7:** Diagrama de casos de uso de la vista de una pregunta

#### 3.1.3.6. Diagrama: Vista de una tarea

Los casos de uso accesibles desde la vista de una tarea, a la que se accede con el caso de uso 'Ver detalles tarea' al seleccionar una tarea del listado de tareas en la vista de una clase.

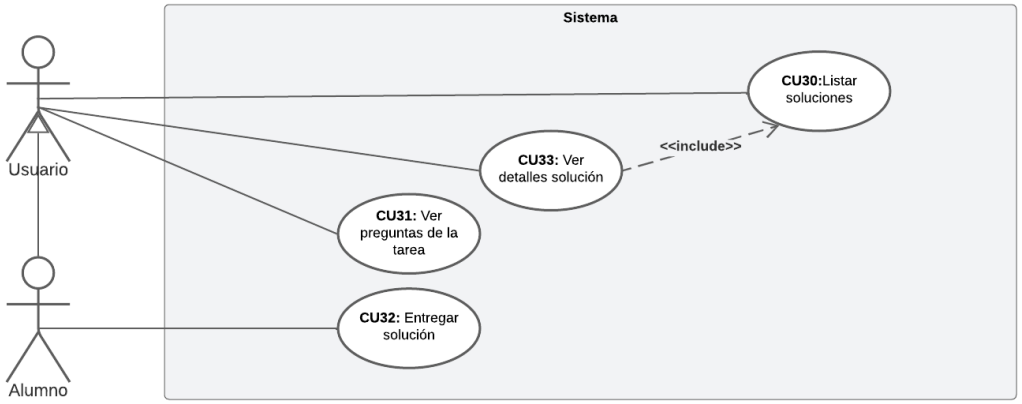



**Figura 3.8:** Diagrama de casos de uso de la vista de una tarea

#### 3.1.3.7. Diagrama: Vista de una solución

Los casos de uso a los que se puede acceder desde la vista de una solución, a la que se accede con el caso de uso 'Ver detalles solución' al seleccionar una solución del listado de soluciones de la vista anterior.

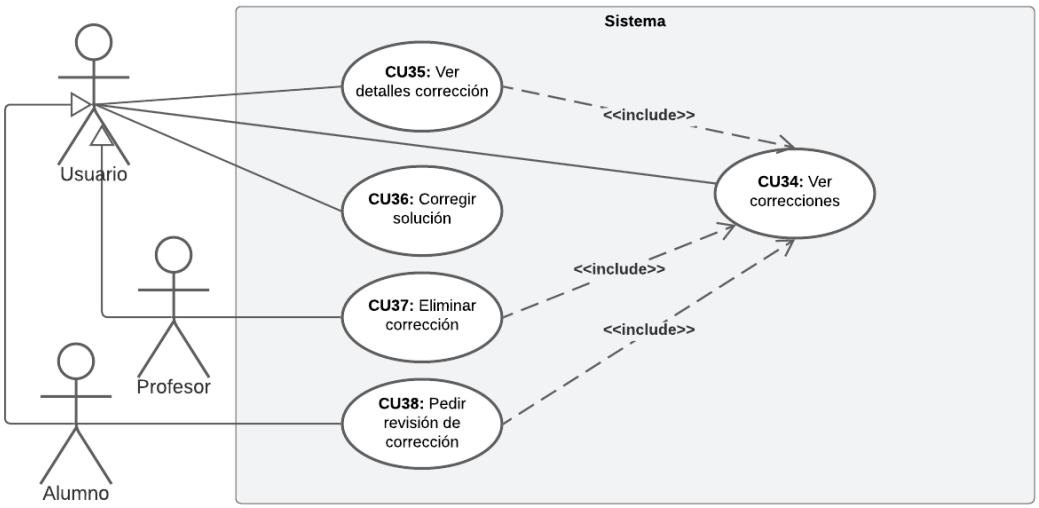



**Figura 3.9:** Diagrama de casos de uso de la vista de una solución

### 3.1.4. Requisitos funcionales

En esta sección se describirá en detalle cada requisito funcional mencionado en los previos diagramas de casos de uso, especificando una descripción y las precondiciones necesarias.

| | |
|---|---|
| **Identificador** | CU01 |
| **Nombre** | Iniciar sesión |
| **Precondición** | No tener una sesión iniciada |
| **Descripción** | El usuario introduce su correo y contraseña. Si la información es correcta, el sistema redirige al usuario a la página de 'Clases' (Tabla 3.8) con la vista correspondiente a su rol. |

**Tabla 3.2:** Caso de Uso 01 : Iniciar sesión

| | |
|---|---|
| **Identificador** | CU02 |
| **Nombre** | Registrarse |
| **Precondición** | No tener una sesión iniciada |
| **Descripción** | El usuario elige si es un profesor o un alumno. A continuación debe introducir nombre y apellidos, un correo perteneciente a la UPV y una contraseña. Además, si se trata de un profesor se le pedirá una clave de la que sólo disponen los profesores. El sistema validará la información y si es correcta se almacenará un nuevo usuario no verificado y se procederá al proceso de verificación (Tabla 3.4). |

**Tabla 3.3:** Caso de Uso 02 : Registrarse

| **Identificador** | CU03 |
|---|---|
| **Nombre** | Verificación de cuenta |
| **Precondición** | Registro realizado |
| **Descripción** | El sistema envía un email con un enlace de verificación al usuario. Una vez este pulse el enlace, su cuenta se activará y podrá iniciar sesión con ella. |

**Tabla 3.4:** Caso de Uso 03 : Verificación de cuenta

| **Identificador** | CU04 |
|---|---|
| **Nombre** | Recuperar contraseña |
| **Precondición** | |
| **Descripción** | El usuario introduce su correo electrónico y el sistema envía un email con su contraseña. |

**Tabla 3.5:** Caso de Uso 04 : Recuperar contraseña

| **Identificador** | CU05 |
|---|---|
| **Nombre** | Ver perfil |
| **Precondición** | Tener una sesión iniciada |
| **Descripción** | Cuando el usuario selecciona su icono de perfil, el sistema despliega un menú donde puede cerrar sesión (Tabla 3.7) y ver información como su correo, nombre y apellidos. En caso de ser un alumno también le muestra su puntuación. |

**Tabla 3.6:** Caso de Uso 05 : Ver perfil

| **Identificador** | CU06 |
|---|---|
| **Nombre** | Cerrar sesión |
| **Precondición** | Tener una sesión iniciada |
| **Descripción** | El sistema termina la sesión y redirige al usuario a la página de inicio de sesión. |

**Tabla 3.7:** Caso de Uso 06 : Cerrar sesión

| **Identificador** | CU07 |
|---|---|
| **Nombre** | Listar clases |
| **Precondición** | Tener una sesión iniciada |
| **Descripción** | El sistema muestra un listado con las clases a las que pertenece. |

**Tabla 3.8:** Caso de Uso 07 : Listar clases

| | |
|---|---|
| **Identificador** | CU08 |
| **Nombre** | Ver información |
| **Precondición** | Tener una sesión iniciada |
| **Descripción** | El sistema muestra una página con información sobre las distintas ocpiones que permite la página y su funcionamiento. |

**Tabla 3.9:** Caso de Uso 09 : Ver información

| | |
|---|---|
| **Identificador** | CU09 |
| **Nombre** | Ver clase |
| **Precondición** | |
| **Descripción** | Al seleccionar una clase en el listado de clases, el usuario es redirigido a la vista de la clase. |

**Tabla 3.10:** Caso de Uso 09 : Ver clase

| | |
|---|---|
| **Identificador** | CU10 |
| **Nombre** | Crear clase |
| **Precondición** | |
| **Descripción** | El usuario introduce el nombre de la clase y el sistema almacena la clase. |

**Tabla 3.11:** Caso de Uso 10 : Crear clase

| | |
|---|---|
| **Identificador** | CU11 |
| **Nombre** | Ver código de la clase |
| **Precondición** | |
| **Descripción** | El sistema muestra al profesor el código correspondiente a la clase seleccionada, que podrá compartir con sus alumnos para que estos se unan. |

**Tabla 3.12:** Caso de Uso 11 : Ver código de la clase

| | |
|---|---|
| **Identificador** | CU12 |
| **Nombre** | Editar clase |
| **Precondición** | |
| **Descripción** | El profesor introduce un nuevo nombre para la clase y el sistema almacena el cambio. |

**Tabla 3.13:** Caso de Uso 12 : Editar clase

| **Identificador** | CU13 |
|---|---|
| **Nombre** | Unirse a clase |
| **Precondición** | |
| **Descripción** | El alumno introduce el código proporcionado por el profesor y el sistema le otorga acceso a la clase. |

**Tabla 3.14:** Caso de Uso 13 : Unirse a clase

| **Identificador** | CU14 |
|---|---|
| **Nombre** | Borrar clase |
| **Precondición** | |
| **Descripción** | El usuario será desvinculado de la clase seleccionada. En caso de que el actor sea un profesor, el sistema eliminará toda la información perteneciente a la clase. |

**Tabla 3.15:** Caso de Uso 14 : Borrar clase

| **Identificador** | CU15 |
|---|---|
| **Nombre** | Ver *ranking* |
| **Precondición** | Tener una sesión iniciada |
| **Descripción** | El sistema muestra una lista con el nombre de todos los alumnos de la clase ordenados por su puntuación de mayor a menor. |

**Tabla 3.16:** Caso de Uso 15 : Ver *ranking*

| **Identificador** | CU16 |
|---|---|
| **Nombre** | Listar peticiones de revisión |
| **Precondición** | Tener una sesión iniciada |
| **Descripción** | El sistema muestra al profesor un listado con todas las correcciones que han sido solicitadas para revisión por otros alumnos con la opción de ser redirigido a la solución y eliminar la corrección (Tabla 3.38). Se mostrará el nombre del alumno corrector, el nombre de la tarea a la que corresponde y la fecha de solicitud. |

**Tabla 3.17:** Caso de Uso 16 : Listar peticiones de revisión

| | |
|---|---|
| **Identificador** | CU17 |
| **Nombre** | Listar tareas |
| **Precondición** | Estar en una clase |
| **Descripción** | El sistema muestra un listado con las tareas que contiene la clase, por defecto ordenadas por fecha de entrega más reciente. Se mostrará el estado de la tarea, el título, el tema al que está relacionado, su fecha de entrega y el número de soluciones propuestas. A los alumnos también se les mostrará el número de correcciones que han realizado. |

**Tabla 3.18:** Caso de Uso 17 : Listar tareas

| | |
|---|---|
| **Identificador** | CU18 |
| **Nombre** | Crear tema |
| **Precondición** | Estar en una clase |
| **Descripción** | El usuario introduce el nombre de un tema y el sistema almacena el tema relacionado a la clase en la que se encuentra. |

**Tabla 3.19:** Caso de Uso 18 : Crear tema

| | |
|---|---|
| **Identificador** | CU19 |
| **Nombre** | Crear tarea |
| **Precondición** | Estar en una clase |
| **Descripción** | El profesor introduce un título, fecha de entrega y tema. Opcionalmente puede introducir una descripción y adjuntar archivos. Finalmente, el sistema almacena la tarea. |

**Tabla 3.20:** Caso de Uso 19 : Crear tarea

| | |
|---|---|
| **Identificador** | CU20 |
| **Nombre** | Filtrar tareas |
| **Precondición** | Estar en una clase |
| **Descripción** | El usuario selecciona el orden de las tareas por fecha de entrega, si quiere ver las tareas de un solo tema o de todos y en qué estados están las tareas que quiere ver. El sistema muestra las tareas con dichos filtros. |

**Tabla 3.21:** Caso de Uso 20 : Filtrar tareas

| | |
|---|---|
| **Identificador** | CU21 |
| **Nombre** | Ver detalles tarea |
| **Precondición** | |
| **Descripción** | El sistema muestra toda la información de la tarea, incluyendo la descripción y los archivos adjuntos. |

**Tabla 3.22:** Caso de Uso 21 : Ver detalles tarea

| | |
|---|---|
| **Identificador** | CU22 |
| **Nombre** | Listar preguntas |
| **Precondición** | Estar en una clase |
| **Descripción** | El sistema muestra un listado con las preguntas que contiene la clase, por defecto ordenadas por fecha de publicación más reciente. Se mostrará el estado de la pregunta, el título, el tema al que está relacionada y el número re respuestas. |

**Tabla 3.23:** Caso de Uso 22 : Listar preguntas

| | |
|---|---|
| **Identificador** | CU23 |
| **Nombre** | Crear pregunta |
| **Precondición** | Estar en una clase |
| **Descripción** | El usuario introduce el título, la descripción, el tema y opcionalmente puede adjuntar archivos y asociar la pregunta a una tarea. El sistema almacena la pregunta. |

**Tabla 3.24:** Caso de Uso 23 : Crear pregunta

| | |
|---|---|
| **Identificador** | CU24 |
| **Nombre** | Filtrar preguntas |
| **Precondición** | Estar en una clase |
| **Descripción** | El usuario selecciona el orden de las preguntas por fecha de publicación, si quiere ver las preguntas de un solo tema o de todos y en qué estados están las preguntas que quiere ver. El sistema muestra las preguntas con dichos filtros. |

**Tabla 3.25:** Caso de Uso 24 : Filtrar preguntas

| | |
|---|---|
| **Identificador** | CU25 |
| **Nombre** | Ver detalles preguntas |
| **Precondición** | |
| **Descripción** | El sistema muestra toda la información de la pregunta, incluyendo la descripción y los archivos adjuntos. |

**Tabla 3.26:** Caso de Uso 25 : Ver detalles preguntas

| | |
|---|---|
| **Identificador** | CU26 |
| **Nombre** | Listar respuestas |
| **Precondición** | |
| **Descripción** | El sistema muestra al usuario un listado con las respuestas que contiene la pregunta, ordenadas por fecha de publicación más reciente. Se mostrará el nombre del alumno y la fecha de publicación. |

**Tabla 3.27:** Caso de Uso 26 : Listar respuestas

| | |
|---|---|
| **Identificador** | CU27 |
| **Nombre** | Ver detalles respuesta |
| **Precondición** | |
| **Descripción** | El sistema muestra toda la información de la respuesta, incluyendo la descripción y los archivos adjuntos. |

**Tabla 3.28:** Caso de Uso 27 : Ver detalles respuesta

| | |
|---|---|
| **Identificador** | CU28 |
| **Nombre** | Responder pregunta |
| **Precondición** | La pregunta no tiene respuesta correcta aún y el usuario no ha respondido ya |
| **Descripción** | El usuario introduce la respuesta y opcionalmente adjunta archivos. El sistema almacena la respuesta. |

**Tabla 3.29:** Caso de Uso 28 : Responder pregunta

| | |
|---|---|
| **Identificador** | CU29 |
| **Nombre** | Marcar respuesta correcta |
| **Precondición** | Ser el creador de la pregunta |
| **Descripción** | El usuario selecciona la respuesta correcta. El sistema ahora muestra esta respuesta como correcta la primera en la lista. |

**Tabla 3.30:** Caso de Uso 29 : Marcar respuesta correcta

| **Identificador** | CU30 |
|---|---|
| **Nombre** | Listar soluciones |
| **Precondición** | Haber entregado una solución o renunciado a la opción de entregar una |
| **Descripción** | El sistema muestra un listado con las soluciones que contiene la tarea, ordenadas por mayor calificación. Se muestra el nombre del usuario que ha propuesto la solución y la calificación actual. |

**Tabla 3.31:** Caso de Uso 30 : Listar soluciones

| **Identificador** | CU31 |
|---|---|
| **Nombre** | Ver preguntas de la tarea |
| **Precondición** | |
| **Descripción** | El sistema muestra un listado de las preguntas relacionadas con la tarea. |

**Tabla 3.32:** Caso de Uso 31 : Ver preguntas de la tarea

| **Identificador** | CU32 |
|---|---|
| **Nombre** | Entregar solución |
| **Precondición** | |
| **Descripción** | El usuario introduce una descripción y opcionalmente adjunta archivos. El sistema almacena la solución. |

**Tabla 3.33:** Caso de Uso 32 : Entregar solución

| **Identificador** | CU33 |
|---|---|
| **Nombre** | Ver detalles solución |
| **Precondición** | |
| **Descripción** | El sistema muestra toda la información de la solución, incluyendo la descripción y los archivos adjuntos. |

**Tabla 3.34:** Caso de Uso 33 : Ver detalles solución

| | |
|---|---|
| **Identificador** | CU34 |
| **Nombre** | Ver correcciones |
| **Precondición** | Haber entregado una corrección, renunciado a la opción o ser el usuario creador de la solución |
| **Descripción** | El sistema muestra un listado con las correcciones que contiene la solución, ordenadas por mayor calificación. Se muestra el nombre del usuario que ha propuesto la solución y la calificación asignada. |

**Tabla 3.35:** Caso de Uso 34 : Ver correcciones

| | |
|---|---|
| **Identificador** | CU35 |
| **Nombre** | Ver detalles corrección |
| **Precondición** | |
| **Descripción** | El sistema muestra toda la información de la corrección, incluyendo la descripción y los archivos adjuntos. |

**Tabla 3.36:** Caso de Uso 35 : Ver detalles corrección

| | |
|---|---|
| **Identificador** | CU36 |
| **Nombre** | Corregir solución |
| **Precondición** | No haber entregado una corrección previamente |
| **Descripción** | El usuario selecciona la clasificación. Si no selecciona "Bien", debe introducir un comentario y opcionalmente archivos adjuntos. El sistema almacena la corrección y recalcula la clasificación que le corresponde a la solución. |

**Tabla 3.37:** Caso de Uso 36 : Corregir solución

| | |
|---|---|
| **Identificador** | CU37 |
| **Nombre** | Eliminar corrección |
| **Precondición** | |
| **Descripción** | El sistema elimina la corrección y recalcula la calificación que le corresponde a la solución. |

**Tabla 3.38:** Caso de Uso 37 : Eliminar corrección

| | |
|---|---|
| **Identificador** | CU38 |
| **Nombre** | Pedir revisión de corrección |
| **Precondición** | Ser el usuario creador de la solución |
| **Descripción** | El sistema almacena la petición de revisión. |

**Tabla 3.39:** Caso de Uso 38 : Pedir revisión de corrección

### 3.1.5. Sistema de puntuación

Como se ha mencionado previamente, los alumnos pueden recibir puntos a través de entregar soluciones, corregir soluciones de otros alumnos o responder correctamente a preguntas. Cada una de estas opciones otorgará distintos puntos y se calculará de distintas maneras. Además, para evitar la utilización indebida de este sistema para obtener puntos, se restará puntuación a soluciones y correcciones erróneas. Las puntuaciones que se muestran en las tablas de las siguientes secciones son estimaciones iniciales, que serán ajustadas de la manera correspondiente tras un análisis de los datos recogidos en el futuro, cuando la aplicación se comience a utilizar.

**Cálculo de la clasificación de una solución**
En primer lugar, es necesario clarificar cómo se calculará la clasificación de una solución. Esto se hará en el momento de cierre de la tarea, es decir, cuando su fecha de entrega límite haya expirado. Se realizará en base a las clasificaciones asignadas por las correcciones que no han sido eliminadas al ser revisadas por el profesor:

- 'Bien': Todas las clasificaciones de las correcciones son Bien.
- 'Regular': Existe alguna corrección que la clasifique como Regular.
- 'Mal': Existe alguna corrección que la clasifique como Mal.

**Respuestas correctas**
En cuanto el alumno creador de la pregunta marque la respuesta que considere correcta como tal, automáticamente se le asignará al alumno que ha respondido 50 puntos más.

**Soluciones**
La puntuación que recibe el alumno que ha entregado la solución se calculará también en el momento de cierre de la tarea, en base a la clasificación de la solución:

| Solución | Puntuación |
|---|---|
| Bien | +100 |
| Regular | +50 |
| Mal | -25 |

**Tabla 3.40:** Tabla de puntuación recibida por soluciones

**Correcciones**
La puntuación que recibe el alumno que ha corregido la solución se calculará también en el momento de cierre de la tarea, en base a la clasificación de la solución y en función de la clasificación de la corrección.

| Solución | Corrección | Puntuación |
|---|---|---|
| Bien | Bien | +25 |
| Bien | Regular | +0 |
| Bien | Mal | +0 |
| Regular | Bien | -15 |
| Regular | Regular | +25 |
| Regular | Mal | +0 |
| Mal | Bien | -30 |
| Mal | Regular | -15 |
| Mal | Mal | +25 |

**Tabla 3.41:** Tabla de puntuación recibida por correcciones

## 3.2 Prototipado de pantallas

El prototipado de todas las pantallas que estarán disponibles en la aplicación está realizado con la herramienta Figma[7]. Todos los casos de uso definidos estan representados por sus identificadores en las secciones de la aplicación que corresponden. En los prototipos se verá representado el acceso a todos los casos de uso por motivos de optimización del prototipo, pero aquellos actores que no puedan acceder a ellos no verán esta representación. Se mostrará agunos de los prototipos más importantes, y los estrictamente necesarios para inferir cómo se deben implementar el resto, pero se pueden ver en su totalidad en el apéndice A
En la figura 3.10 podemos ver los paneles que se verán en las distintas páginas de autenticación correspondientes a cada caso de uso indicado.

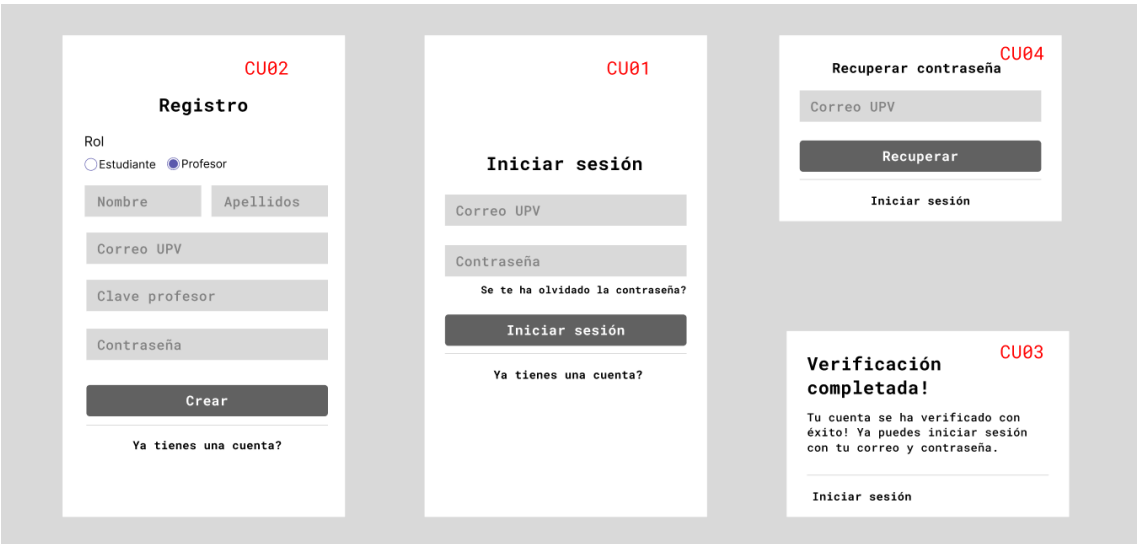



**Figura 3.10:** Prototipo de autenticación

En la figura 3.11 se observa la pantalla correspondiente al caso de uso "Listar clases", donde además se ven representados los casos de uso de la navegación principal, así como los diálogos correspondiente a los casos de uso de la vista de clases.

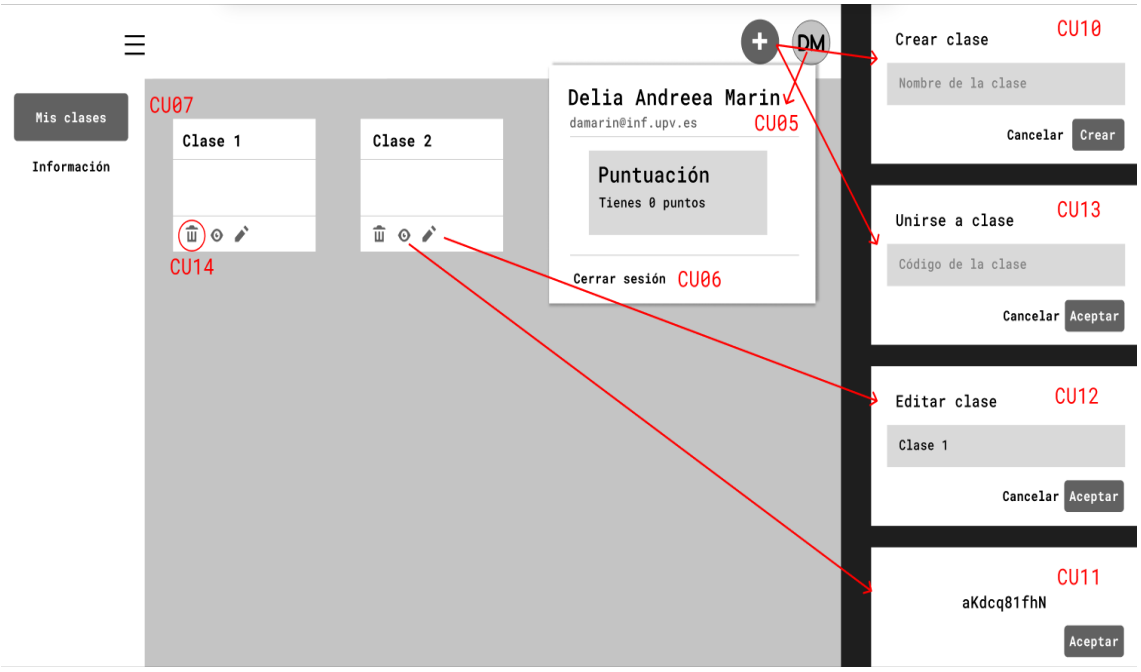



**Figura 3.11:** Prototipo de la vista clases y navegación principal

A continuación se observan los prototipos que representan los casos de uso de la vista de una clase, concretamente los relacionados con las tareas. Aquellos referentes al listado de tareas serán muy similares a los del listado de preguntas. En cuanto al prototipo de la vista de crear una tarea que se puede observar en la figura 3.13 se mantendrá la misma forma tanto para la creación de preguntas, como entrega de soluciones a tareas, respuestas a preguntas o correcciones de soluciones.

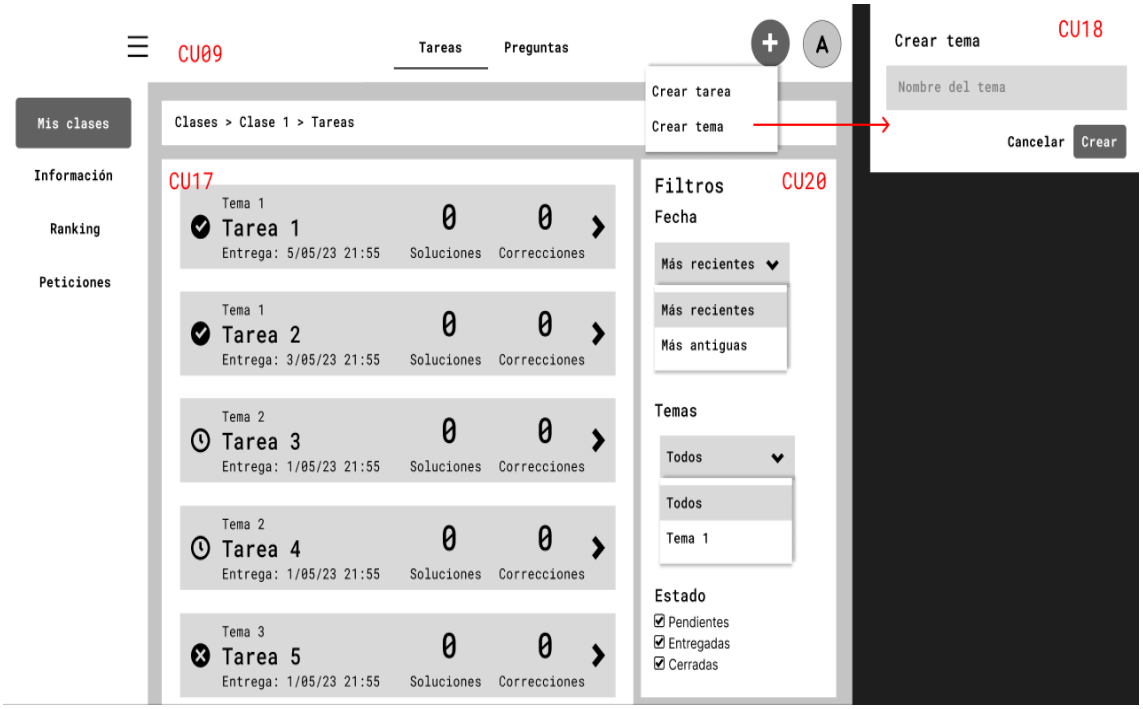



**Figura 3.12:** Prototipo de la vista de listar tareas

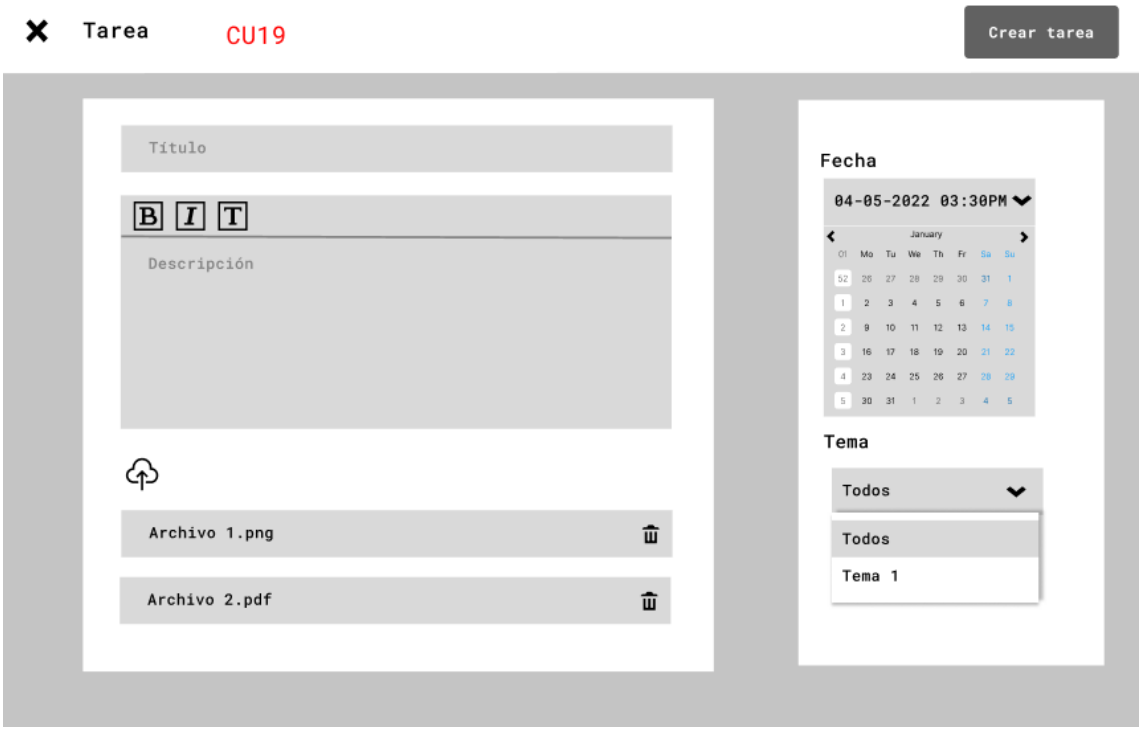



**Figura 3.13:** Prototipo de la vista de crear tarea

En las figuras mostradas a continuación se pueden observar los casos de uso correspondientes a la visualización del *ranking* y del listado de peticiones de revisión.

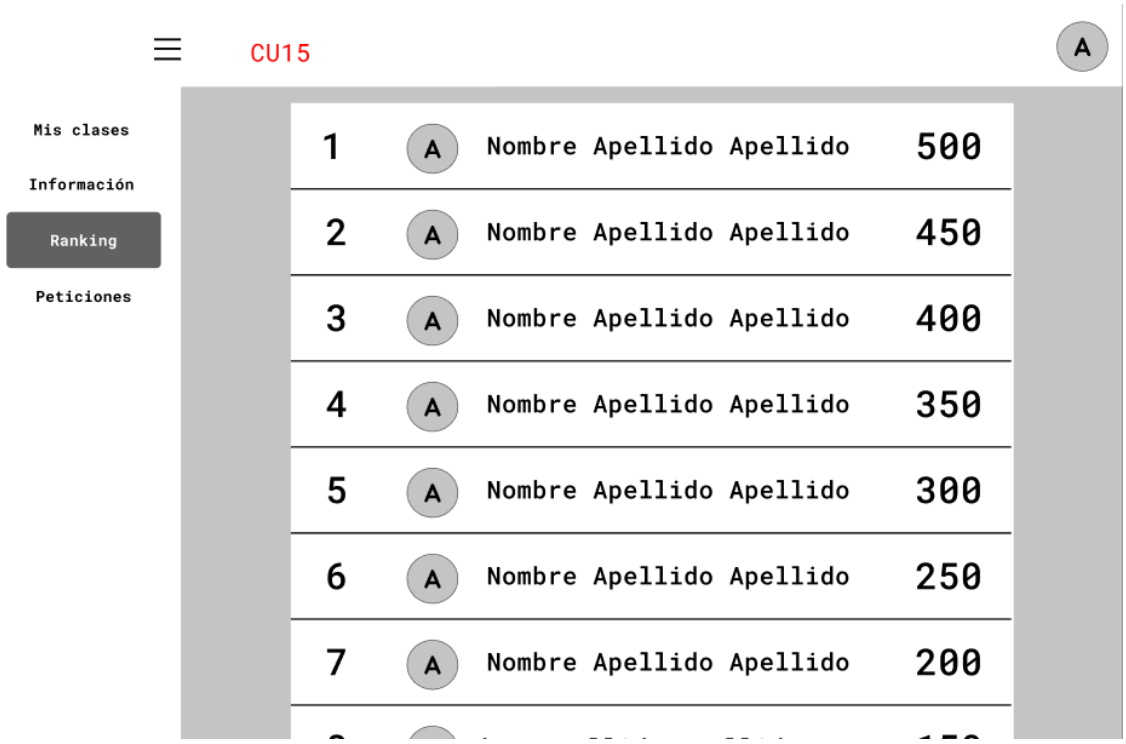



**Figura 3.14:** Prototipo de la vista del *ranking*

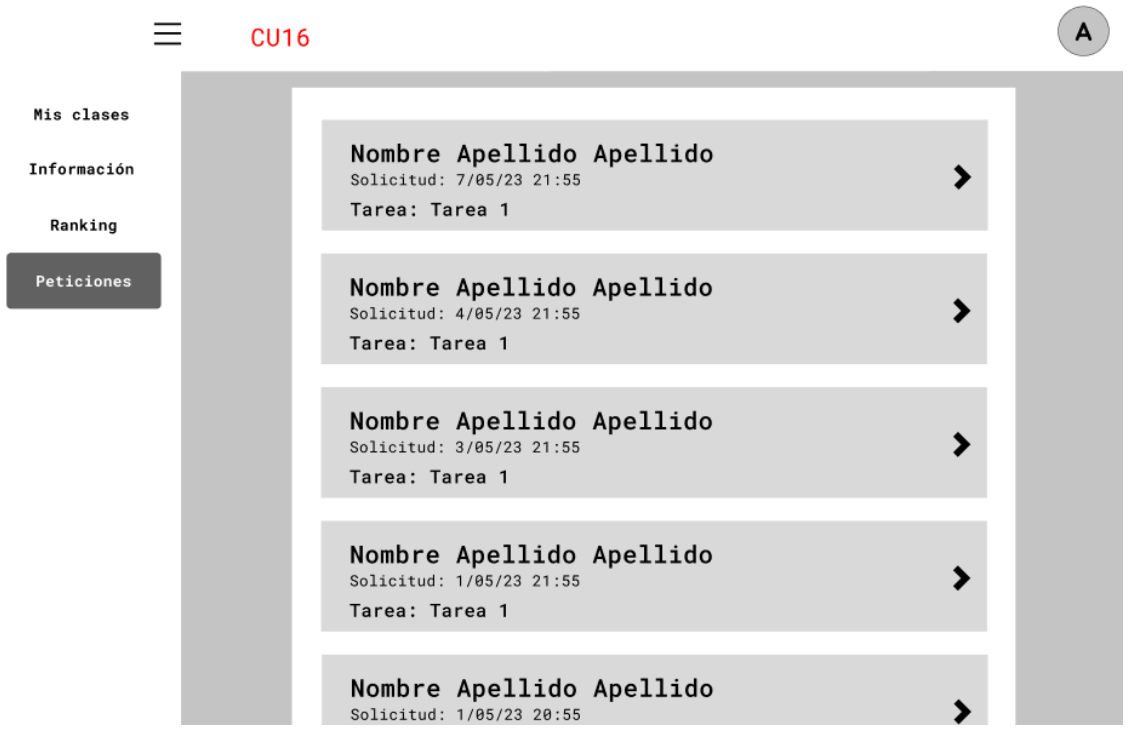



**Figura 3.15:** Prototipo de la vista de peticiones de revisión

En la figura 3.16 se observa la vista de una tarea. La vista de una pregunta será muy similar, listando respuestas en lugar de soluciones. Lo mismo se aplicaría para las soluciones y sus correcciones.

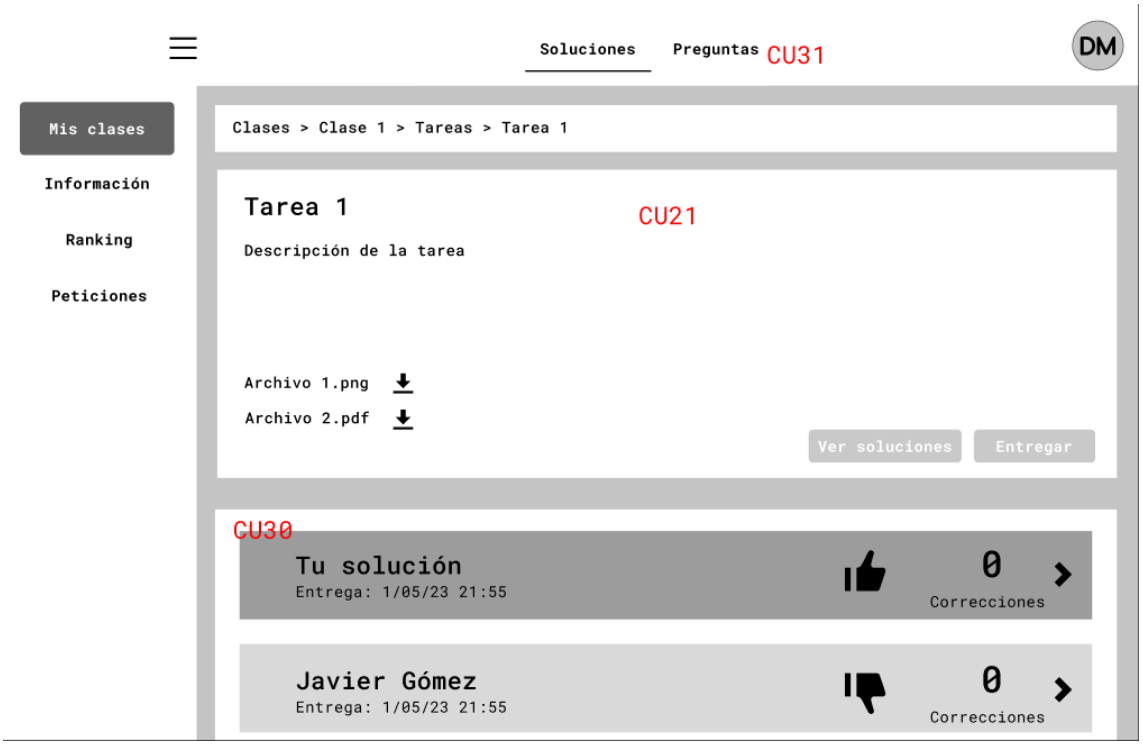



**Figura 3.16:** Prototipo de la vista de una tarea

# CAPÍTULO 4
# Diseño de la solución

*"Elegir la tecnología adecuada es fundamental para el logro y la escalabilidad de su empresa de servicios públicos. Con una gran cantidad de lenguajes de programación, marcos, bibliotecas y bases de datos disponibles, el proyecto de elegir la mejor combinación puede resultar abrumador."*[8]

## 4.1 Tecnología empleada

Una de las prácticas fundamentales en el desarrollo de software es la reutilización de código o reutilización de software. Consiste en el uso de software existente para construir nuevo software de manera más eficiente y rápida [9]. Esto es precisamente lo que nos proporciona el uso de los frameworks. Gracias a la amplia colección de librerías y módulos preconstruidos que ofrecen, permiten mucha más agilidad y evitan errores a la hora de desarrollar y mantener software [10]. Es por eso que se han convertido en la opción preferida por los desarrolladores para desarrollar aplicaciones web modernas y eficientes, y por lo que se ha decidido utilizar en este proyecto también.

En este proyecto, se ha optado por utilizar React en el lado del cliente y Spring Boot para el servidor. Estas dos tecnologías, junto con un gestor de bases de datos como MySQL, proporcionan una combinación robusta que garantiza un software escalable y mantenible.

### 4.1.1. React

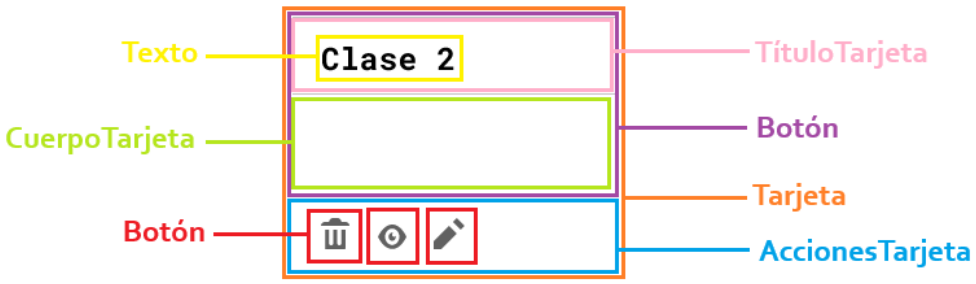



**Figura 4.1:** Componente Tarjeta utilizado para las clases en la vista del profesor

React [11] es una librería de JavaScript de código abierto, utilizada para el desarrollo de front-end, desarrollada y mantenida por Facebook.

Su enfoque está basado en la utilización de los llamados componentes. Se trata de piezas independientes y autónomas, con su propia estructura, estilo y lógica, que se pueden reutilizar en diferentes partes de la aplicación. Esto se ve facilitado por el uso de un modelo de flujo de datos unidireccional. Es decir, cada componente puede tener su propio estado interno, y cuando los datos cambian, React actualiza y renderiza solo los componentes dependientes de dichos datos. Esto proporciona un mayor rendimiento y permite mayor facilidad para comprender la lógica de la aplicación y por lo tanto depurar.

Para poder realizar esto, React utiliza lo que se conoce como el *Virtual DOM*.

#### 4.1.1.1. Virtual DOM

El *Virtual DOM* es una representación en memoria del *Document Object Model (DOM)* real de una página web, que representa una estructura de árbol de datos. Almacenar este árbol en memoria permite que React pueda optimizar cómo se actualizan partes específicas del mismo.

Para ello, cada vez que se produce un cambio en los datos, ocurre lo siguiente:

1. **Cambio del estado:** React genera una nueva versión del *Virtual DOM* tras renderizar los componentes cuyo estado ha cambiado.
2. **Diffing:** Se produce una comparación de la nueva versión con la anterior, identificando qué partes han cambiado.
3. **Re-renderización:** Se calcula la forma más eficiente de actualizar el DOM real en base a las diferencias encontradas, actualizando lo estrictamente necesario.

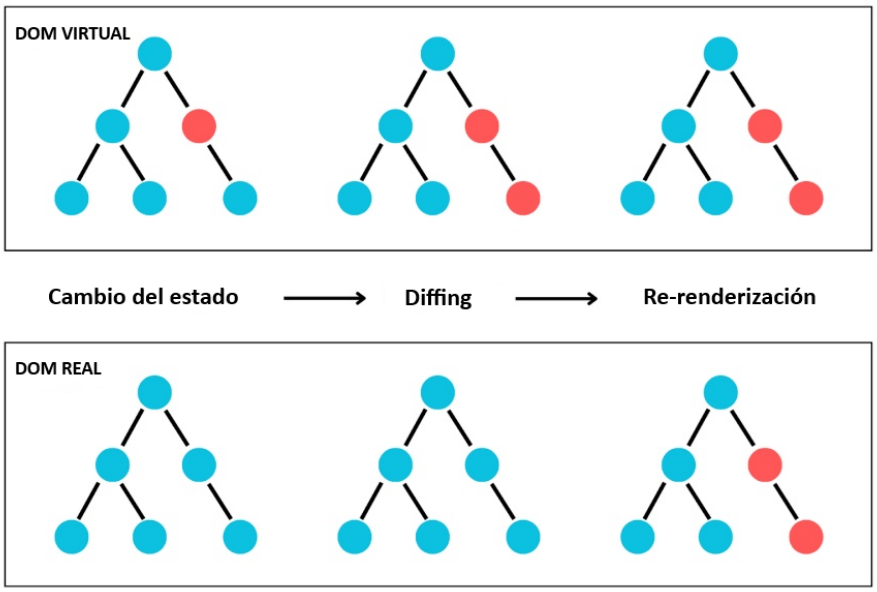



**Figura 4.2:** Proceso del Virtual DOM. Fuente: [12]

Algunas de las ventajas que tiene React sobre otras alternativas son las siguientes:

- Al ser una biblioteca, es mucho más flexible que frameworks como Angular, ya que permite más libertad de elección sobre distintas herramientas y soluciones que se adapten

mejor a las necesidades del proyecto, en lugar de forzar soluciones integradas para implementaciones como el enrutamiento o el manejo de estado, como es el caso de Angular.

- Gracias al patrón de actualización del DOM virtual de React, renderizar la interfaz tras cambios en el estado es más rápido y utliza menos recursos, ofreciendo así un mejor rendimiento que, por ejemplo, la reactividad automática de Vue.
- Está respaldado por una amplia comunidad de desarrolladores y empresas, ofreciendo una mayor cantidad de paquetes, herramientas y recursos de aprendizaje, que facilitan el desarrollo.

### 4.1.2. Spring y Spring boot

Spring es un framework de código abierto para el desarrollo de aplicaciones Java. El principal beneficio que ofrece es su gestión de la infraestructura que permite a los desarrolladores enfocarse más en la lógica de negocio.

Está compuesto por módulos interconectados que tienen distintos propósitos como testing, seguridad, integración de datos, autenticación, etc. Pueden ser utilizados de manera independiente o en conjunto.

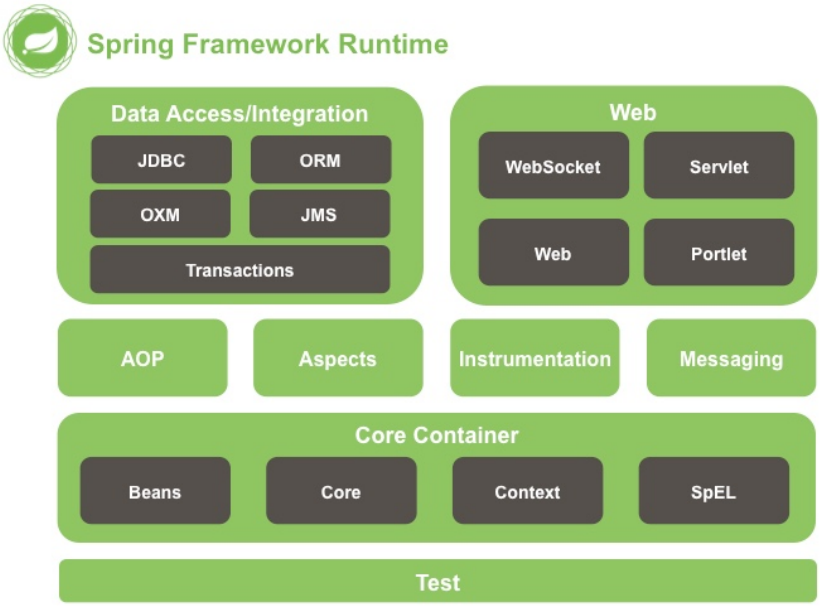



**Figura 4.3:** Módulos de Spring. Fuente: [13]

Otro aspecto importante es el uso de inyección de dependencias. Esto facilita la configuración e integración de los diferentes componentes, los cuales reciben automáticamente las dependencias sin necesidad de buscarlas, consiguiendo menos acoplamiento.

Java Spring Boot [14] es una extensión de Spring que acelera y simplifica el proceso de desarrollo y despliegue de microservicios y aplicaciones web, reduciendo el tiempo de instalación y configuración al ofrece configuraciones predeterminadas y automatizar tareas repetitivas. Para ello hace uso de tres funciones principales:

- **La autoconfiguración.** Se basa en el uso de configuraciones y dependencias predefinidas según las prácticas recomendadas, reduciendo la posibilidad de errores humanos.

Detecta automáticamente los paquetes presentes en el proyecto y los usa para configurar la aplicación de manera apropiada.

- **Starters.** Los "Starters"son un conjunto de dependencias que se pueden añadir al proyecto para habilitar funcionalidades configurando automáticamente valores por defecto y proporcionando todas las dependencias necesarias, simplemente puedes añadiedo el starter al archivo de configuración del proyecto.

- **Aplicaciones autónomas.** Permite crear aplicaciones que se ejecutan de manera independiente sin necesidad de servidores web externos, incorporando servidores como Tomcat o Netty durante el proeso de inicialización. Esto permite iniciar la aplicación en diversas plataformas sin complicaciones.

### 4.1.3. Visual Studio Code

Como IDE de desarrollo para el frontend de la aplicación se ha seleccionado Visual Studio Code, desarrollado por Microsoft. Esta elección se debe a su ligereza y versatilidad, así como a la amplia gama de extensiones disponibles que facilitan el desarrollo con React.

### 4.1.4. IntelliJ

Para el backend de la aplicación, se ha optado por utilizar IntelliJ IDEA de JetBrains, en su versión Ultimate, que ofrece soporte para Spring Boot y bases de datos SQL como MySQL. Este IDE es reconocido por su capacidad para facilitar la escritura de código más limpio y eficiente, gracias a sus avanzadas características de análisis y generación de código, así como de refactorización. Además, proporciona integración con Maven y Gradle, herramientas fundamentales para la gestión de dependencias y automatización de proyectos Java.

### 4.1.5. Base de datos: MySQL

#### 4.1.5.1. Bases de datos relacionales

Al tratarse de una aplicación que consiste principalmente en creación y modificación de muchos objetos relacionados entre sí de distintas maneras, es indispensable utilizar una base de datos relacional ya que estas aportan:

- **Un modelo de datos tabular**. Al organizar los datos en tablas que representan entidades diferentes, donde cada fila es una instancia de esa entidad, se facilita mucho la comprensión y representación clara y estructurada de los datos.

- **Facilidad para CRUD.** Las operaciones CRUD se simplifican mediante el uso de SQL, un lenguaje de consulta poderoso y bien establecido que permite manipular e interactuar con los datos de manera eficiente y segura.

- **Integridad del los datos.** Gestionan la integridad de los datos a través de reglas de integridad y restricciones de clave foránea, asegurando que las relaciones entre los datos sean lógicamente coherentes y que las operaciones no dejen la base de datos en un estado inconsistente. Esto también está respaldado por el soporte de transacciones, que permiten ejecutar múltiples operaciones como una unidad de trabajo coherente.

#### 4.1.5.2. MySQL

MySQL[15] es un gestor de bases de datos relacional que utiliza el lenguaje SQL. Es considerada la base de datos de código abierto más popular[16] y está soportada por Oracle.

El motivo por los que se ha elegido MySQL sobre otras bases de datos relacionales como PostgreSQL es que dispone de un gran equilibro entre los siguientes aspectos:

- **Escalabilidad.** MySQL puede manejar tanto pequeñas como grandes cantidades de datos, lo que es ideal para una aplicación que puede crecer en número de usuarios y en complejidad de datos.
- **Compatibilidad con Spring Boot.** MySQL se integra de manera sencilla con Spring Boot a través de Tomcat, facilitando la configuración y reduciendo el tiempo de desarrollo.
- **Seguridad de datos.** Proporciona robustas opciones de seguridad, incluyendo cifrado de datos y control de acceso basado en roles, muy importante para proteger la información de los usuarios.
- **Alto rendimiento y optimización.** Con características como el almacenamiento en caché de consultas y la optimización de índices, MySQL puede ejecutar operaciones de manera rápida y eficiente, lo cual es esencial para una aplicación interactiva y en tiempo real.

#### 4.1.5.3. Diagrama Entidad-Relación

A continuación se incluirá el diagrama entidad-relación que representa la organización de las distintas entidades así como sus atributos y relaciones necesarias para la implementación de todas las funciones que conciernen a esta aplicación. Este modelo, organizado en torno a relaciones de uno a uno, uno a muchos y muchos a muchos, permite una interacción eficiente y un almacenamiento de datos coherente.


**Figura 4.4:** Leyenda de símbolos diagrama entidad-relación

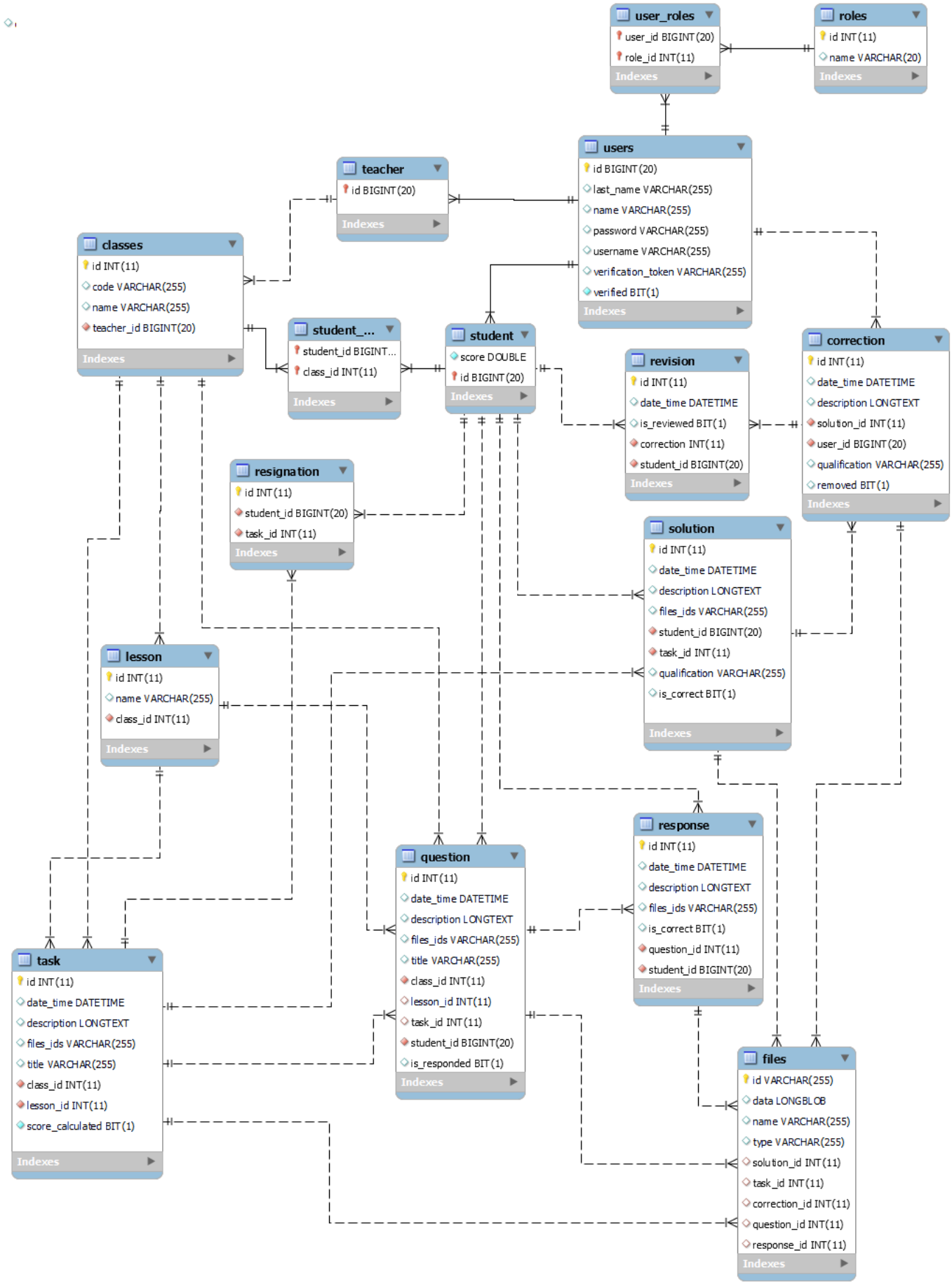



**Figura 4.5:** Diagrama entidad-relación

En el diagrama entidad-relación, podemos observar que se ha implementado una tabla para cada una de las entidades del dominio de nuestra aplicación, además de una tabla para los archivos, las peticiones de revisión y las renuncias (*resignation*) por parte de los alumnos para poder visualizar las soluciones de otros compañeros. Cada una de estas entidades están identificadas mediante uno o varios atributos que actúan como 'calves primarias'. Además, contienen 'claves foráneas' para hacer referencia a los identificadores de aquellas entidades con las que están relacionadas. Estas relaciones pueden implicar que puedn estar relacionadas o bien con una única instancia de la entidad correspondiente, o con muchas. Por ejemplo, una clase puede contener muchas tareas, pero solo un profesor. Es interesante destacar que algunas entidades contienen atributos derivados, que son calculados en base al valor de atributos de la propia entidad u otras, como las puntuaciones de los alumnos y las tareas.

### 4.1.6. Git

Para la gestión del código fuente de la aplicación, se ha utilizado Git, un sistema de control de versiones distribuido, muy útil para la coordinación de proyectos de desarrollo de software, y el projecto se ha alojado en un repositorio en GitHub. A pesar de que se ha dispuesto de un único desarrollador, ha resultado útil para mantener una historia clara y detallada de las modificaciones realizadas en el proyecto. Esto no solo facilita la identificación y corrección de errores al permitir revertir a versiones anteriores del código, sino que también posibilita una organización metodológica del trabajo mediante el uso de ramificaciones (branches).

#### 4.1.6.1. Feature Branch Workflow

En este caso, al ser solo un desarrollador y no tener la preocupación de gestionar una versión en producción, se ha optado por utilizar "Feature Branch Workflow". Se trata de una metodología simple y directa que consiste en una rama principal *master* y una rama *feature* por cada nueva característica o mejora, que se fusiona en la principal una vez satisfecho el desarrollo de dicha funcionalidad.

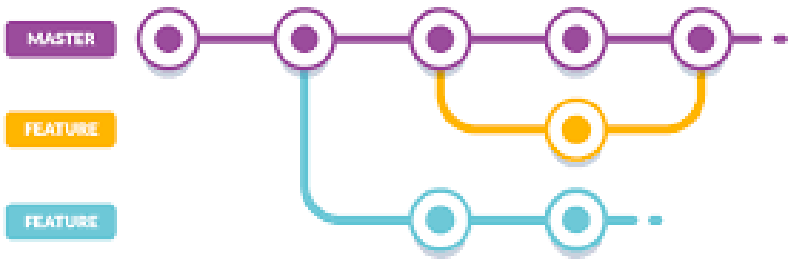



**Figura 4.6:** Diagrama del funcionamiento de Feature Branch Workflow

## 4.2 Arquitectura del sistema

La elección de la arquitectura adecuada es un paso importante en el desarrollo de cualquier sistema de software. En el caso de nuestra aplicación, se ha optado por una combinación de la arquitectura cliente-servidor y la arquitectura por capas, con el objetivo de aprovechar las

fortalezas de cada una y adaptarlas a las necesidades y limitaciones del proyecto. Este enfoque busca optimizar tanto el proceso de desarrollo como las operaciones de mantenimiento y escalado posterior del sistema.

### 4.2.1. Arquitectura cliente-servidor

La arquitectura cliente-servidor es un modelo estructural que distribuye las tareas entre los proveedores de recursos o servicios, conocidos como servidores, y los solicitantes de servicios, llamados clientes. En nuestra aplicación:

- **Cliente (Frontend con React).** El cliente se encarga de presentar la interfaz de usuario y manejar las interacciones del usuario. Este lado de la aplicación se comunica con el backend a través de solicitudes HTTP, solicitando datos y enviando respuestas del servidor para procesar y almacenar.
- **Servidor (Backend con Spring Boot).** El servidor gestiona la lógica de negocio, el acceso a la base de datos, y la autenticación de los usuarios. Utiliza Spring Boot para estructurar la lógica de negocio, manejar las solicitudes del cliente, y comunicar las respuestas apropiadas. La aplicación del servidor también es responsable de interactuar con la base de datos MySQL para guardar y recuperar datos necesarios para las funcionalidades de la aplicación.

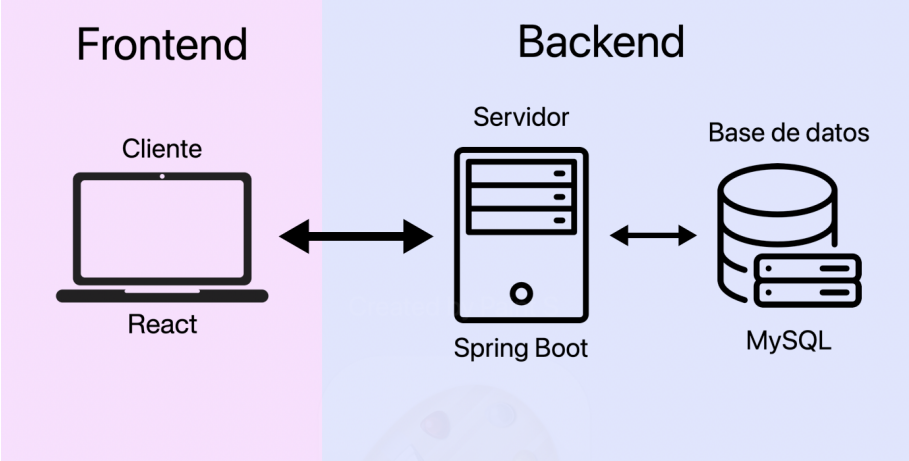



**Figura 4.7:** Diagrama de la arquitectura cliente-servidor

### 4.2.2. Arquitectura por capas

En el backend, nuestra aplicación utiliza una arquitectura por capas, que es una forma de arquitectura de software que divide la aplicación en grupos lógicos, o capas, cada una con una responsabilidad específica. Esta estructura facilita la gestión del código y mejora su mantenibilidad al separar las preocupaciones de cada capa:

- **Capa de Presentación.** Aunque en el contexto del backend esto usualmente se refiere a la capa que interactúa directamente con el cliente, en nuestro caso, esta capa está más orientada a manejar las peticiones y respuestas que se intercambian entre el cliente y el servidor.
- **Capa de Lógica de Negocio.** Contiene la lógica de aplicación esencial para las funciones del negocio. En esta capa se implementan las decisiones sobre cómo procesar los datos y se gestionan aspectos como validación o ejecución de procesos.

- **Capa de Acceso a Datos.** Se encarga de la comunicación con la base de datos. Gestiona la obtención, inserción y actualización de los datos en la base de datos, utilizando repositorios y modelos para abstraer y simplificar estas operaciones.

Esta separación de responsabilidades asegura que cada componente del backend sea independiente, lo que simplifica las pruebas, la depuración y el mantenimiento del sistema.

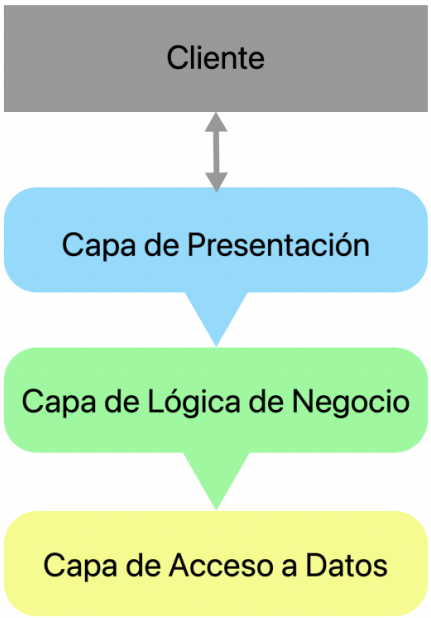



**Figura 4.8:** Diagrama de la arquitectura por capas

# CAPÍTULO 5
# Desarrollo de la solución

En este capítulo se entrará más en detalle en cuanto a las diferentes herramientas de las que se han hecho uso para implementar cada parte de la arquitectura mencionada en el capítulo anterior.

## 5.1 Estructura del proyecto

Es importante tener una estructura de los directorios adecuada para desarrollar un código de calidad, legible e intuitivo. Para ello se han definido los siguientes árboles de directorios.

### 5.1.1. Frontend

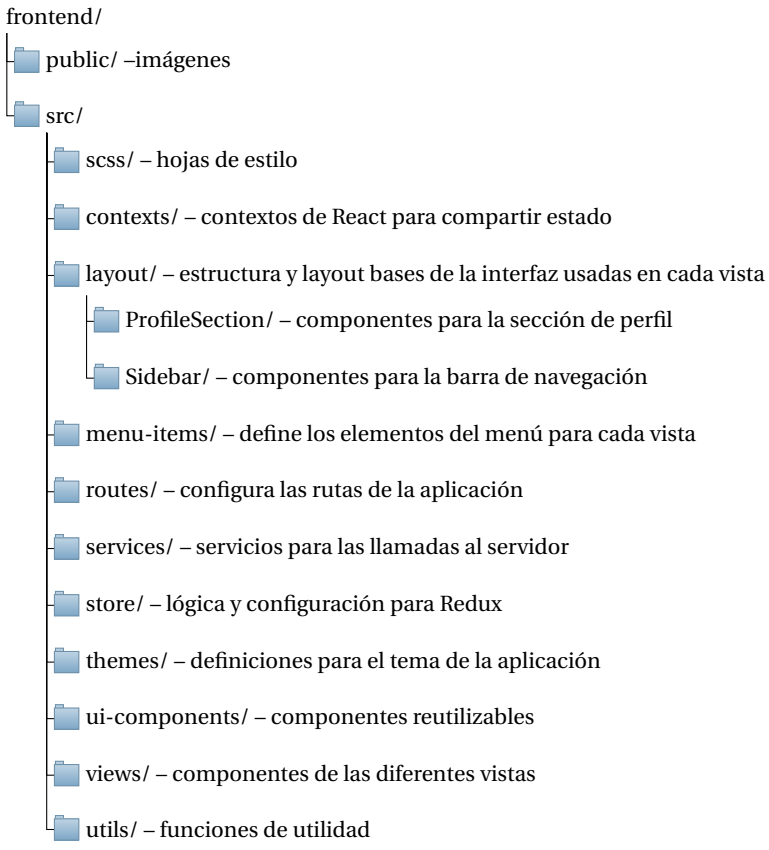



**Figura 5.1:** Estructura de directorios del frontend

En el frontend se han separado en diferentes directorios los componentes relacionados con cada aspecto del que se compone la aplicación, tal como se define en los comentarios de cada directorio en la estructura 5.1.

### 5.1.2. Backend

La estructura del backend se ha desarrollado de manera que facilita la comprensión de una arquitectura segmentada por capas, agrupando los componentes necesarios para cada capa en un directorio diferente. Estos componentes son explicados en más detalle en la siguiente sección de este capítulo.

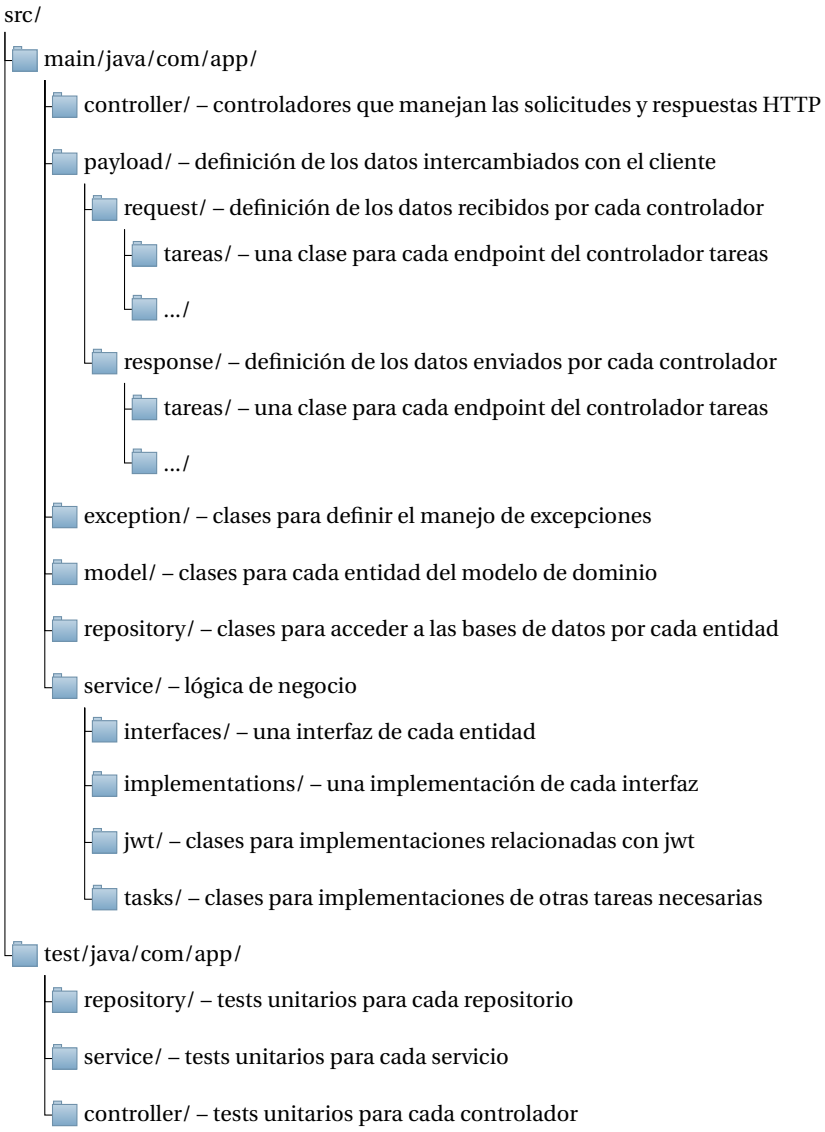



**Figura 5.2:** Estructura de directorios del backend

## 5.2 Codificación del aplicativo

En este apartado se explicarán las diferentes librerías utilizadas para cada capa de la arquitectura del proyecto, así como algunas de las implementaciones llevadas a cabo. Para cada una de las capas del backend, se puede encontrar un ejemplo con un fragmento de código en el Apéndice B,

### 5.2.1. Frontend

#### 5.2.1.1. Diseño: Material UI

Para el diseño de la interfaz de usuario se ha decidido utilizar Material UI, una biblioteca de componentes de React orientada al que implementa los principios del Material Design. Permite la creación de interfaces estéticamente agradables de manera muy sencilla y rápida gracias a sus componentes de UI predefinidos.

#### 5.2.1.2. Manejo de estados: React Redux, React Context y React State

Para la gestión de estados dentro de la aplicación se utilizaron varias herramientas de React:

- React Redux: Ofrece una solución global para el manejo del estado, útil para estados que necesitan ser compartidos entre muchos componentes. El estado se almacena en la *store*, los componentes manipulan el estado a través de unos objetos enviados usando la función *dispatch* y estas acciones son tomadas por unas funciones llamadas reducers, que en base al estado actual devuelven un nuevo estado.

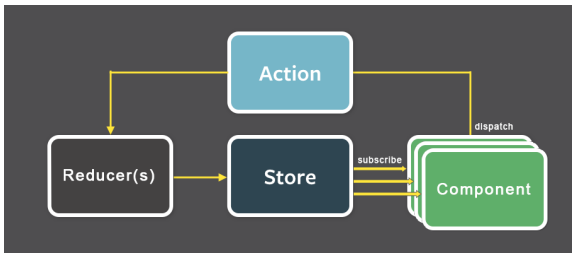



**Figura 5.3:** Diagrama del flujo de React Redux

- React Context: Proporciona una forma de especificar los valores de algunas variables que deben utilizar los *Consumers*, componentes que están envueltos por el componente *Provider*, sin tener que pasar explícitamente un prop a través de cada nivel del árbol manualmente.

- React State: Utilizado para manejar el estado local dentro de los componentes individuales, haciendo que estos sean re-renderizados para mostrar la nueva información cada vez que este estado es actualizado.

#### 5.2.1.3. Comunicación HTTP: Axios

Para la comunicación del cliente con el servidor se ha decidido utilizar Axios, una biblioteca de JavaScript que se utiliza para realizar solicitudes HTTP desde el navegador a través de una API muy fácil de usar. Acelera el desarrollo ya que transforma automáticamente los datos de las solicitudes y respuestas desde o a JSON.

#### 5.2.1.4. Enrutación: React Router

React Router se ha empleado para manejar la navegación en la aplicación ya que facilita la creación de aplicaciones de una sola página (SPA), donde los componentes de la página cambian en respuesta a las interacciones del usuario sin requerir una recarga completa de la página.

### 5.2.2. Capa de Presentación

#### 5.2.2.1. API REST

Una API (interfaz de programación de aplicaciones) establece un contrato de servicio entre aplicaciones, definiendo como estas deben comunicarse entre sí, siendo el cliente la aplicación que utiliza el servicio que el servidor ofrece. Una API REST se basa en un conjunto de principios que permiten al cliente y al servidor comunicarse mediante el protocolo HTTP, utilizando métodos estándares como GET, POST, PUT y DELETE.

En este caso, la API se ha realizado a través de las anotaciones que Spring Boot ofrece para facilitar la configuración y el mapeo de las rutas para cada solicitud.

La idea principal consiste en crear controladores, es decir, las clases que definen todas las solicitudes HTTP que el cliente puede enviar y cómo se procesa la información recibida, a través de la invocación de los servicios implementados en la capa de lógica de negocio. Además, también se encargan de preparar adecuadamente las respuestas que se devuelve al cliente, así como declarar todos los datos requeridos por parte del cliente para poder procesar correctamente la solicitud.

Para declarar que una clase es un controlador en Spring Boot, se utiliza la etiqueta *@RestController*, las rutas se especifican mediante la etiqueta *@RequestMapping* y para declarar qué tipo de solicitud HTTP debe gestionar cada método, se utilizan las etiquetas *@PostMapping*, *@GetMapping*, *@DeleteMapping* y *@PutMapping*, que mapean las operaciones CRUD a los métodos HTTP correspondientes.

#### 5.2.2.2. Autenticación: JWT y Security

Para el sistema de autenticación se ha decidido utilizar JWT (Jason Web Token). Se trata de un estándar que define una manera segura de transmitir información firmada digitalmente como un objeto JSON. En este caso, lo que se codifica es la identidad del usuario, que el cliente utiliza para cada solicitud que hace al servidor. Esto se genera cuando el usuario se identifica y después se utiliza para decidir si posee autorización para acceder a determinados recursos y servicios.

En este caso además, se ha integrado con Spring Security, que es un marco de seguridad, utilizado en este caso para controlar el acceso a recursos en base a roles.

En la aplicación que estamos discutiendo, se definen dos roles principales utilizando Spring Security:

- **ROLE_TEACHER.** Este rol habilita el acceso a aquellas funcionalidades encargadas de cumplir con los casos de uso definidos para los profesores.
- **ROLE_STUDENT.** Este rol habilita el acceso a aquellas funcionalidades encargadas de cumplir con los casos de uso definidos para los estudiantes.

#### 5.2.2.3. Verificación y recuperación: Spring Boot Starter Mail

Para las funciones de verificación de cuentas tras el registro de un nuevo usuario y recuperación de contraseña, se ha implementado un sistema de generación de correo electrónico a través del uso del starter “Spring Boot Starter Mail“ que nos permite utilizar la interfaz ‘JavaMailSender‘ proporcionada por Spring Framework. Esta interfaz abstrae la complejidad involucrada en el uso de JavaMail, la API estándar de Java para envíos de correos electrónicos. A continuación se muestra un ejemplo de utilización de esta interfaz para una implementación sencilla de un método que envía un correo electrónico:

```java
@Service
public class EmailService {

    @Autowired
    private JavaMailSender mailSender;

    public void sendSimpleEmail(String to, String subject, String text) {
        SimpleMailMessage message = new SimpleMailMessage();
        message.setTo(to);
        message.setSubject(subject);
        message.setText(text);
        mailSender.send(message);
    }
}
```

### 5.2.3. Capa de Lógica de Negocio

La capa de lógica de negocio se ha implementado utilizando interfaces que definen los servicios necesarios. Cada una de estas interfaces representa un conjunto de operaciones que se puede realizar sobre una entidad y tienen su correspondiente implementación a través de una clase que utiliza la etiqueta *@Service*. Cada una de estas implementaciones además inyectan el repositorio correspondiente a la entidad sobre la que operan, lo que permite interactuar con la base de datos.

#### 5.2.3.1. Sistema de puntuación: Cron scheduler

Además de clases que operan sobre las entidades de nuestro sistema, también se han implementado otros servicios como los encargados de generar y validar los tokens JWT mencionados anteriormente, así como el servicio necesario para actualizar las puntuaciones de los alumnos, que se explica en esta sección.

Utilizando la anotación @Scheduled, se define un cronograma para ejecutar automáticamente este proceso todos los días a las 2:00 AM, asegurando que las puntuaciones se calculen y actualicen teniendo en cuenta tanto las clasificaciones de las soluciones como las clasificaciones de las correcciones entregadas en aquellas tareas que han alcanzado su fecha límite y que por tanto ya no permiten cambios en las calificaciones de sus soluciones.

El cálculo de la clasificación de la solución se realiza utilizando el servicio correspondiente a la entidad solución, que modifica la clasificación actual por una nueva cada vez que se crea o elimina una corrección nueva, en base a la clasificación establecida por el alumno que ha creado la correción.

Finalmente, el servicio de la entidad respuesta es el encargado de actualizar la puntuación del estudiante que ha entregado una respuesta a una pregunta marcada como correcta.

El cálculo específico de las puntuaciones asignadas se ha realizado siguiendo la lógica descrita en la sección 3.1.5.

### 5.2.4. Capa de Acceso A datos

La capa de acceso a datos en Spring Boot se estructura sobre el marco de trabajo de Spring Data JPA, que simplifica la creación y el mantenimiento de esta capa, facilitando la implementación de repositorios de datos mediante interfaces y permitiendo realizar operaciones sobre la base de datos de una manera más eficiente y menos propensa a errores. Se apoya en el provee-

dor JPA Hibernate para la implementación de estas operaciones, permitiendo una abstracción completa del acceso a la base de datos y la manipulación de entidades.

#### 5.2.4.1. Hibernate

Hibernate se utiliza para la comunicación entre la aplicación Java y la base de datos a través de JPA (Java Persistence API). Esto es posible debido a que proporciona una manera de mapear las entidades a la base de datos relacional mediante anotaciones en la clase Java.

A través de anotaciones, Hibernate permite definir una clase como una entidad de la base de datos y su tabla correspondiente con los respectivos atributos, así como configurar otros aspectos como la especificación de claves primarias, las relaciones entre entidades (muchos-a-muchos, uno-a-muchos o uno-a-uno) y las opciones de carga y cascada.

#### 5.2.4.2. JPARepository

JPARepository es una interfaz de Spring Data JPA que proporciona una gran variedad de métodos comunes para realizar operaciones CRUD sobre una entidad, reduciendo así el tiempo de desarrollo. Además, ofrece la posibilidad de personalizar las operaciones para añadir métodos más complejos a través de consultas escritas en JPQL (Java Persistence Query Language).

### 5.2.5. Tests

En el desarrollo de software, los test o pruebas son procedimientos que ayudan a localizar errores a tiempo, asegurando que la aplicación funciona correctamente y los cambios no afectan al resto de funcionalidades, creando así un software de calidad y mantenible.

#### 5.2.5.1. Tests unitarios

Se han llevado a cabo los llamados tests unitarios, que se centran en comprobar que los métodos y funciones de la aplicación funcionan correctamente de forma individual. Facilitan el desarrollo ágil ya que se pueden ejecutar cada vez que se implementa un nuevo cambio.

Para implementar los tests unitarios con spring boot, se han utilizado JUnit, un framework de pruebas unitarias para Java, y Mockito, un framework de simulación utilizado para simular dependencias de manera que que una clase pueda ser testeada de manera aislada.

Se utilizan las anotaciones *@Mock* para simular instancias de las dependencias que necesita el componente que se está probando, y *@InjectMocks* para crear una instancia del componente probado.

Los métodos que representan un test unitario específico para una funcionalidad del componente se anotan con *Test*. Para implementar cada caso de prueba, se ha seguido el patrón **Arrange-Act-Assert**, que establece el siguiente orden de operaciones:

- **Preparación del ambiente de prueba (Arrange).** Configuración de las instancias simuladas para devolver datos específicos o comportarse de una manera determinada cuando se les invoque.
- **Ejecución del código bajo prueba (Act).** Llamada al método del servicio que se está probando.

- **Verificación de resultados y comportamientos (Assert).** Comprobación de que el método bajo prueba se comporta como se espera, mediante aserciones o verificaciones de interacciones con los mocks.

#### 5.2.5.2. Pruebas end-to-end

Por otra parte, tras finalizar la implementación de cada funcionalidad, tanto en el backend como en el frontend, se han realizado pruebas *end-to-end*, consistentes en los siguientes pasos:

1. Definir criterios de aceptación para cada requisito.
2. Simular el entorno de un usuario final.
3. Manualmente realizar todas las interacciones posibles con la aplicación, verificando que no se produzca ningún resultado inesperado.
4. Resolver los fallos encontrados y volver a realizar la prueba de aceptación

## 5.3 Sprints

En esta sección, se detalla el trabajo realizado durante cada *sprint*, incluyendo los tiempos invertidos, los resultados obtenidos, las pruebas realizadas y los desafíos enfrentados, en caso de haberlos.

Cada *sprint* tuvo una duración de dos semanas, invirtiendo aproximadamente 60 horas por *sprint*.

### 5.3.1. Sprint 1

La mitad del tiempo de este *sprint* se dedicó a montar el repositorio y la infraestructura básica necesaria para correr por primera ver la aplicación. Es decir, se comenzó con la puesta en marcha de la base de datos, la creación del código base de Spring Boot y la generación del proyecto npm para el frontend, así como la interconexión entre todos los componentes.

La otra mitad del tiempo se empleó en la implementación del sistema de autenticación y verificación, tanto en la parte del servidor como el desarrollo de los formularios necesarios en la parte de interfaz de usuario.

Durante este *sprint* se entendió cómo funcionan los diferentes componentes correspondientes a las diferentes capas utilizadas en Spring Boot así como el funcionamiento de las pruebas unitarias. El desconocimiento de estas características fue el principal reto de este *sprint*.

### 5.3.2. Sprint 2

En este *sprint* se invirtieron unas 15 horas en la creación, listado, edición y eliminación de clases, así como la lógica necesaria para que los alumnos se pudieran unir a las clases. Además, 15 horas fueron dedicadas al frontend, para la implementación del *layout* base, con la barra lateral, las pestañas y el menú de perfil. Otras 20 horas fueron dedicadas a la creación, listado y presentación de tareas. El resto de horas fueron invertidas en el desarrollo de las funcionalidades relacionadas con las soluciones de las tareas.

Se simuló el flujo de interacción, realizando todas las operaciones CRUD necesarias sobre las diferentes entidades creadas, comprobando que los cambios se veían reflejados en la base de datos.

El mayor reto en este *sprint* fue mantener el estado de los componentes como el menú y la información del usuario a lo largo de las diferentes pantallas, y establecer en qué lugar de la web se encuentra el usuario dentro de la jerarquía de la clase. Para facilitarlo, se decidió utilizar React redux.

### 5.3.3. Sprint 3

En este *sprint* se dedicaron 15 horas al desarrollo de las funcionalidades referentes a las correcciones y otras 20 a las peticiones de revisión. Finalmente, 25 horas fueron invertidas en generar el sistema de puntuación así como reflejar los cambios de las calificaciones de cada solución correspondientemente en la interfaz.

Además de las pruebas unitarias necesarias para cada caso de uso, se simuló un proceso de interacción entre diferentes alumnos creando y calificando soluciones y correcciones, comprobando que la puntuación se actualizaba de manera correcta y que la información que se mostraba o se ocultaba para cada usuario era la debida.

### 5.3.4. Sprint 4

Este último *sprint* se dedicó a finalizar los requisitos restantes, de manera que se dedicaron unas 10 horas a la creación, listado y presentación de preguntas, y otras 10 horas a las funcionalidades necesarias para crear respuestas, marcarlas como correctas y cambiar la puntuación de manera acorde. Después se emplearon 10 horas en la implementación de los filtros, tanto de tareas como preguntas y se terminó el desarrollo con otras 10 horas para las páginas de *Ranking* e información.

Las 20 horas restantes del *sprint* se dedicaron al despliegue de la aplicación, generando los archivos docker necesarios. Esto fue un gran reto ya que se encontraron diferentes problemas por los que el despliegue final de la aplicación no funcionaba como debía relacionados con las dependencias necesarias y la configuración de la red, pero mediante la búsqueda de información en foros y artículos se pudieron solucionar finalmente.

# CAPÍTULO 6
# Despliegue y mantenimiento

## 6.1 Integración continua

Las versiones resultantes en cada iteración de la metodología ágil *SCRUM* y las entregas de cada característica utilizando la metodología *Feature Branch Workflow* nos facilitan la implementación de la *Integración Continua (CI/CD)*, un enfoque muy beneficioso ya que al integrar y probar cambios de manera regular se minimiza el riesgo de problemas en el desarrollo, disminuyendo nuestro flujo de trabajo de desarrollo y aumentando la calidad del software.

Además, al haber alojado nuestro repositorio en GitHub, podemos hacer uso de las GitHub Actions [17] para llevar a cabo el despliege de *CI/CD*. Esta solución nos permite definir los flujos de trabajo necesarios para preparar y desplegar el ambiente de prueba. Estos flujos se especifican en archivos YAML y permiten configurar *jobs* que compilen el proyecto y ejecuten las pruebas necesarias. Estos flujos son activados mediante los denominados 'eventos de disparos' que especifican cuando se deben ejecutar ciertas acciones, por ejemplo cuando se realiza un *push* hacia la rama *master*. Finalmente, tras la ejecución de las pruebas, presentan dos escenarios posibles:

- Si las pruebas son exitosas, el despliegue de la aplicación a producción se realiza de manera automática.
- Si las pruebas fallan, se bloquea el despliegue y se envían notificaciones al desarrollador para informar de los resultados de las pruebas.

## 6.2 Mantenimiento

El mantenimiento [18] es un aspecto crítico que garantiza la estabilidad del software tras su lanzamiento. Para el mantenimiento continuo de nuestra aplicación:

- Es fundamental mantener la aplicación compatible con las últimas versiones de los navegadores y las actualizaciones de los frameworks y librerías utilizados, y en el caso de nuestra aplicación este aspecto es primordial ya que se han utilizado una amplia variedad de herramientas que además tienen una amplia comunidad y por tanto son continuamente actualizadas. De esta manera prevenimos problemas de seguridad que puedan surgir con versiones obsoletas.
- Debemos implementar nuevas funcionalidades, así como reestructurar la lógica detrás de las funcionalidades ya implementadas para mantener a los usuarios activos gracias a aplicar su feedback en las decisiones tomadas a la hora de realizar cambios.

- Es muy importante identificar y corregir los errores que puedan surgir tras el aumento del uso de la aplicación con el tiempo para mantener la aplicación funcional.

Se analizará y actualizará la aplicación de manera regular para evitar cualquier tipo de problema y agregar nuevas funcionalidades.

## 6.3 Publicación: Docker

La última etapa del desarrollo de la aplicación implica hacer accesible la aplicación a los usuarios a través de Internet. Para lograrlo, utilizaremos Docker[19] para contenerizar la aplicación. Los pasos a seguir son:

1. **Creación del *Dockerfile***. Este archivo se escribe para cada componente, en este caso el frontend en React y el backend en Spring Boot, y define cómo se construye la imagen de docker, especificando las dependencias, el entorno de ejecución y los comandos necesarios para iniciar el servicio.

2. **Creación del *Docker Compose***. Este archivo se utiliza para definir y ejecutar cada uno de los contenedores como un único servicio. Además, especifica cómo estos contenedores deben estar configurados y cómo se interconectan.

3. **Subida de imágenes**. Tras construir las imágenes con el comando *'docker build'*, cada imagen se subirá a *Docker Hub* que es un registro de contenedores que permite que las imágenes estén disponibles para ser descargadas en cualquier entorno de producción, facilitando así la distribución y despliegue de la aplicación.

4. **Despliegue en un servidor de producción**. Se configura el entorno de producción, incluyendo la configuración de la red, la seguridad y la base de datos. Después se utiliza Docker para ejecutar los contenedores según lo definido en el archivo Docker Compose, iniciando los servicios utilizando el comando *'docker-compose up'*.

5. **Configuración de DNS**. Finalmente, se asocia un dominio con la dirección IP del servidor de producción. De esta manera los usuarios pueden acceder a la aplicación utilizando un nombre de dominio fácil de recordar en lugar de una dirección IP.

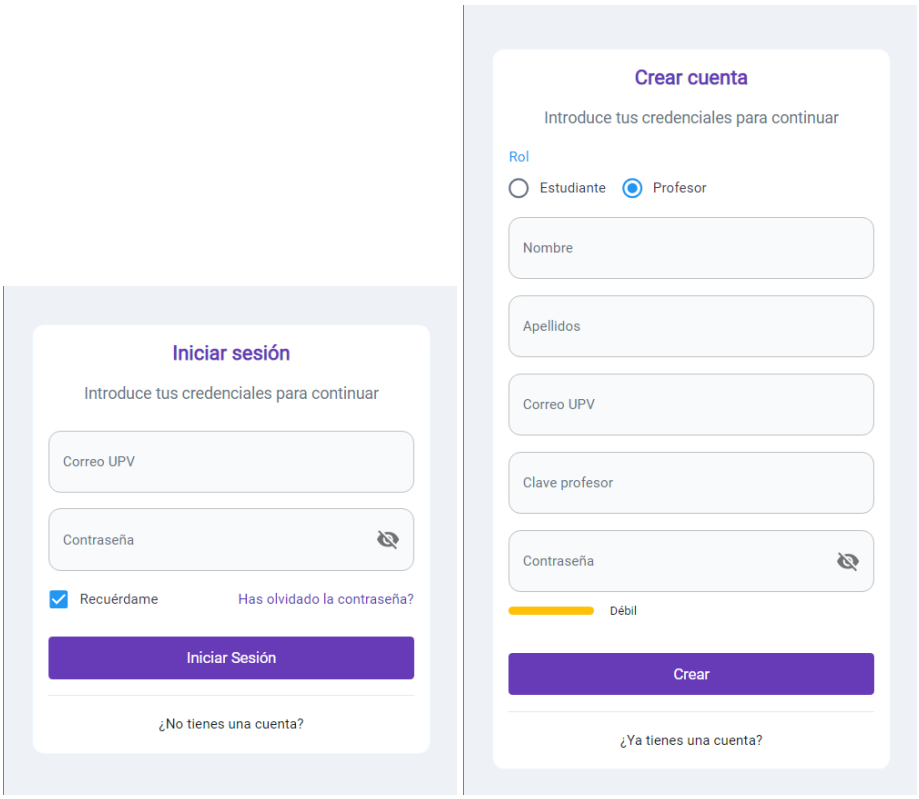
Se puede ver un la implementación de los archivos mencionados en el apéndice C

# CAPÍTULO 7
# Manual de uso

En este capítulo se mostrará el manual de uso resultante del MVP desarrollado.

## 7.1 Autentificación y verificación

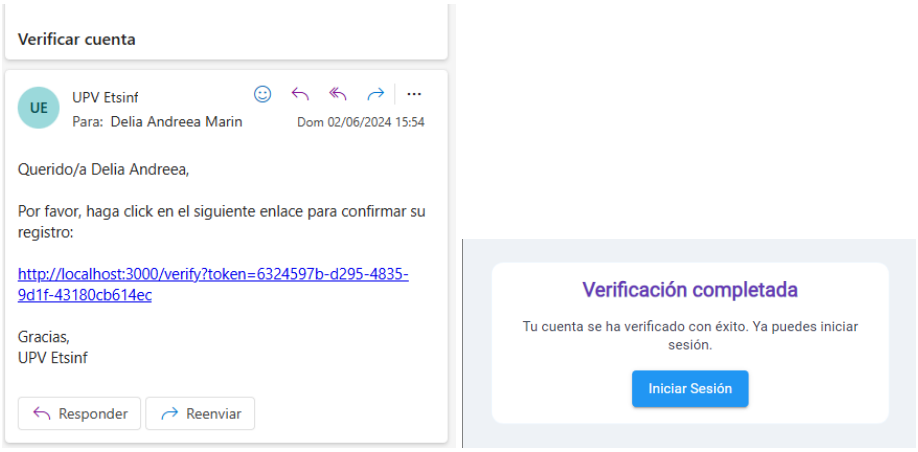
Al iniciar la aplicación, la primera pantalla le pedirá al usuario que inicie sesión. Si el usuario no se ha registrado aún, puede elegir registrarse mediante el botón "¿No tienes una cuenta?". En esta pantalla el usuario deberá elegir si es un profesor o un estudiante e introducir sus datos.




**(a)** Inicio sesión **(b)** Registro

**Figura 7.1:** Autentificación

Una vez hecho esto, recibirá un correo con un enlace que deberá abrir para poder iniciar sesión. Si intenta iniciar sesión sin ser verificado, un nuevo correo con un nuevo enlace será enviado.




**(a)** Email **(b)** Verificación completada

**Figura 7.2:** Verificación

## 7.2 Clases

Una vez el usuario haya iniciado sesión, verá un listado con las clases a las que tiene acceso y tendrá distintas posibilidades dependiendo de si es un profesor o un estudiante.

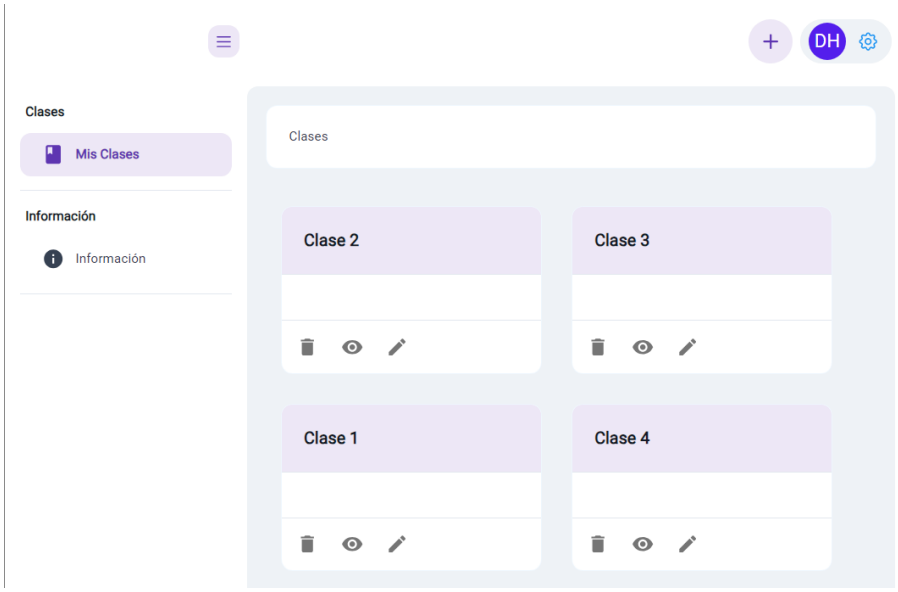



**Figura 7.3:** Listado de clases

Un profesor podrá crear, eliminar y modificar clases, así como ver el código de cada clase, que deberá compartir con sus alumnos para que estos puedan unirse.

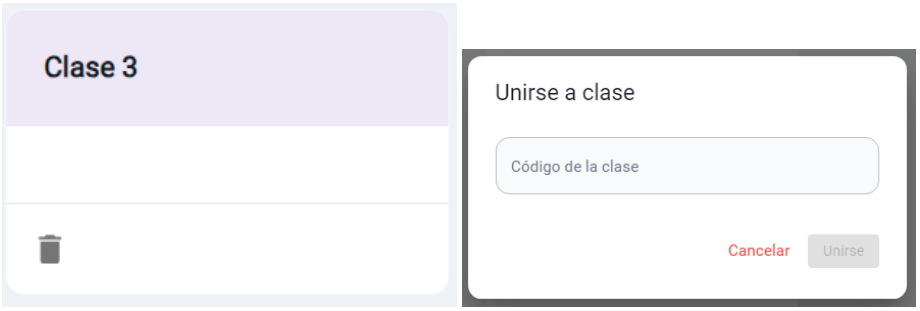
Un alumno podrá unirse a una clase utilizando el código compartido por el profesor o salirse de una clase.

Los usuarios también podrán ver la información de su perfil presionando el icono que contiene las iniciales de su nombre completo.


**(a)** Crear clase **(b)** Eliminar clase

**(c)** Editar clase **(d)** Ver código de la clase

**Figura 7.4:** Clases para profesores




**(a)** Clase en vista alumno **(b)** Unirse a una clase

**Figura 7.5:** Clases para alumnos


**Figura 7.6:** Ver perfil

## 7.3 Tareas

Una vez el usuario seleccione una clase, podrá visualizar el *layout* de la clase, donde podrá listar las tareas de la clase, listar las preguntas de la clase o ver el *ranking* actual de la clase. Si es un profesor, además, podrá acceder a un listado de peticiones de revisión desde el menú, así como crear una tarea o tema.

En el listado de tareas, el usuario podrá ver una vista previa con información de cada tarea. Están ordenadas por fecha y mediante un icono muestran si el usuario ya ha entregado una solución, si está pendiente o si la fecha límite de entrega ha pasado. Se podrán filtrar según estos aspectos y, además, podrá ver el número de soluciones que tiene la tarea y cuántas de ellas ha corregido. Todo ello se puede ver en la figura a continuación.

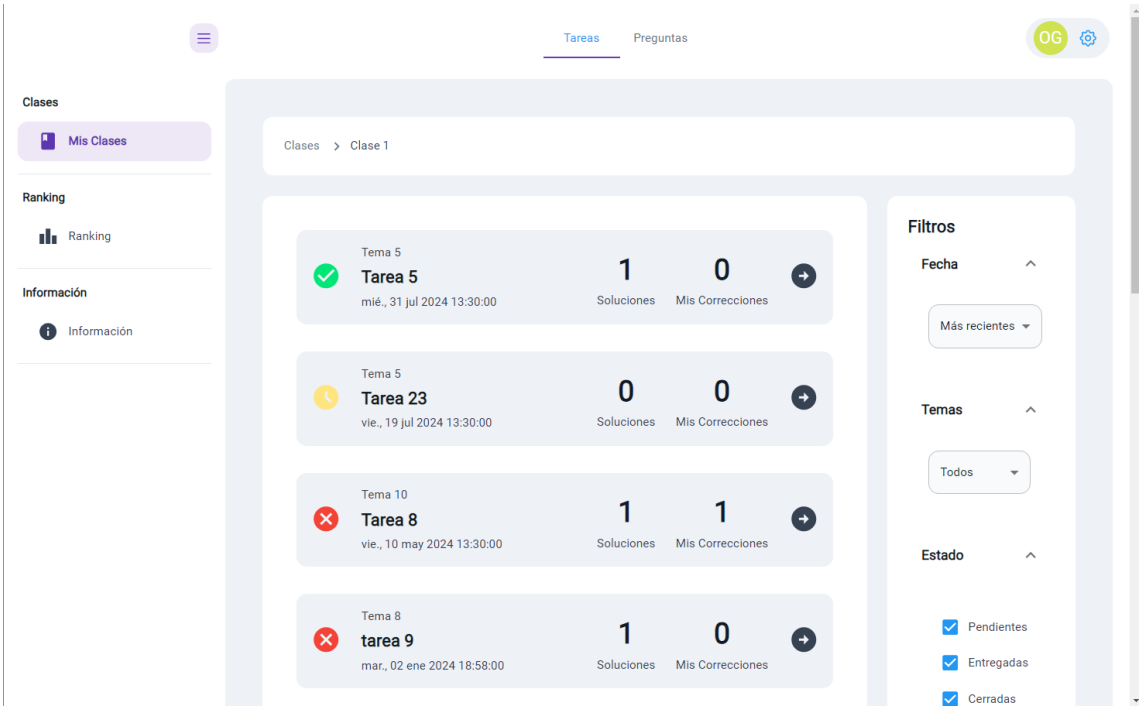



**Figura 7.7:** Listado de tareas

Para crear una tarea, un profesor deberá pulsar sobre el botón con el icono de cruz, y seleccionar 'Crear tarea' (figura 7.8), desde donde aparecerá una ventana de diálogo donde podrá definir la tarea y cargar los recursos necesarios, así como elegir el tema relacionado y la fecha de entrega límite, como se puede ver en la figura 7.9.

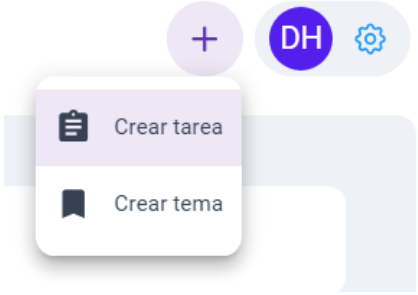



**Figura 7.8:** Opciones para crear del profesor

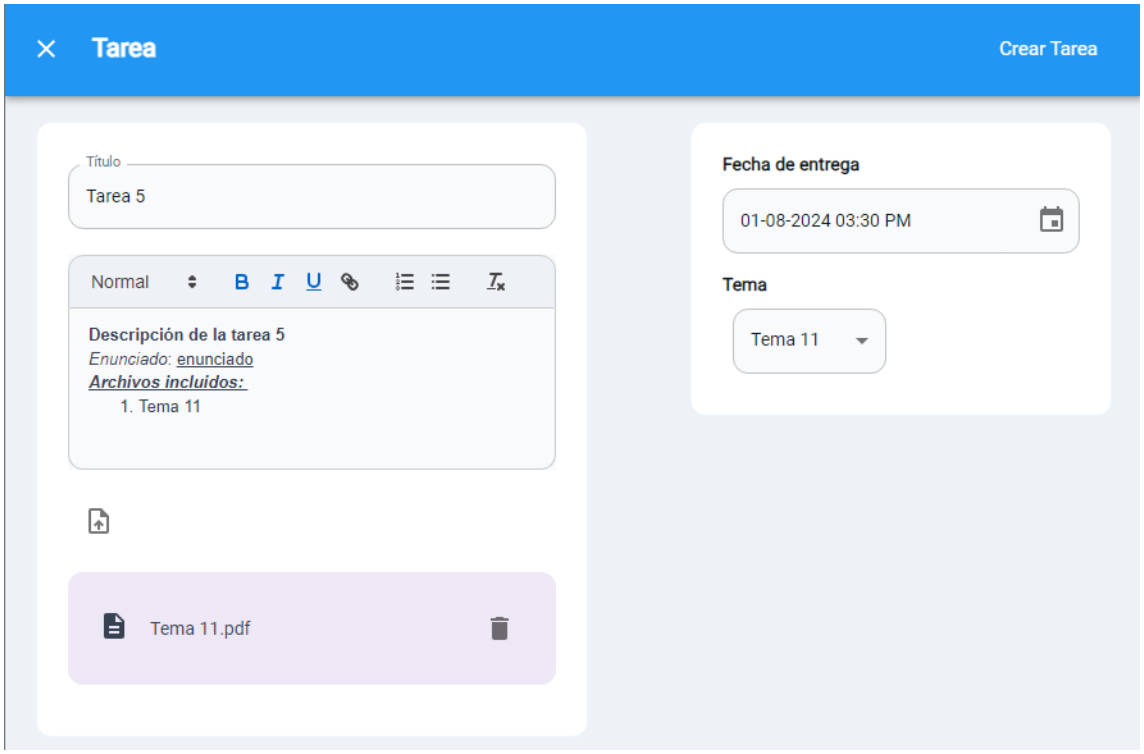



**Figura 7.9:** Crear tarea

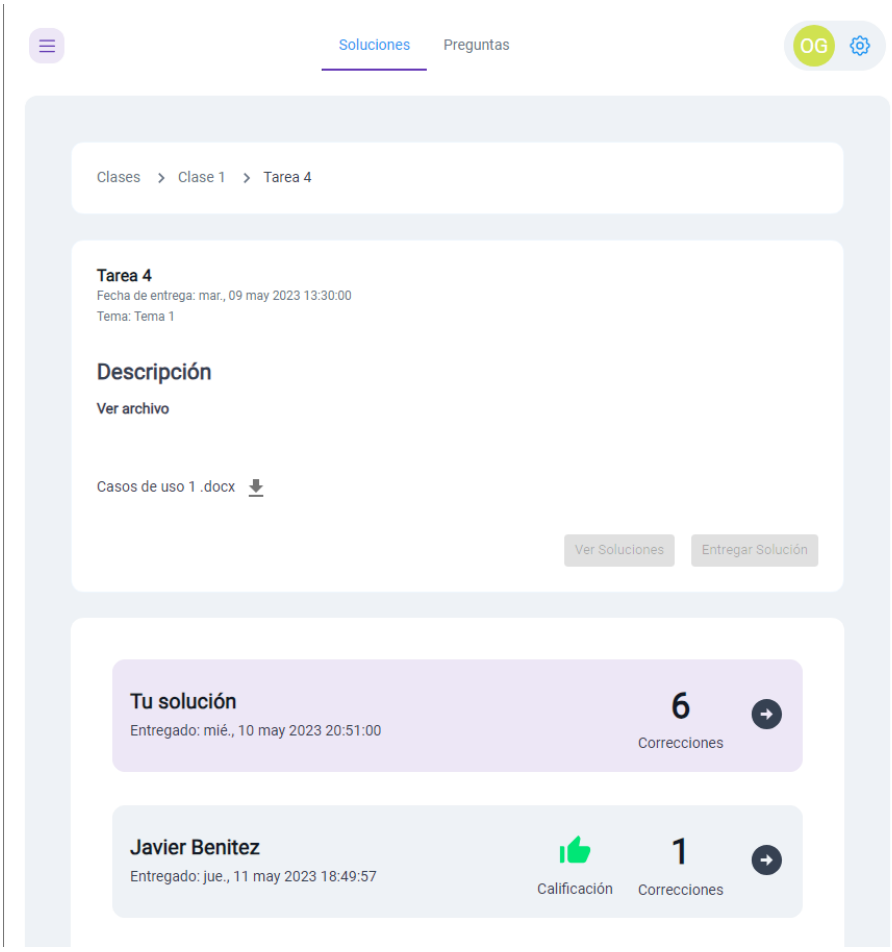



**Figura 7.10:** Ver detalles tarea

Al seleccionar una tarea, se podrán ver todos los recursos asociados a la tarea, así como el listado de soluciones con sus respectivas calificaciones calculadas y el número de correcciones que tienen.Si el usuario ha entregado una solución, verá su solución primera y no podrá entregar más soluciones, además de que podrá ver las soluciones de los demás. Si no ha entregado una solución, para poder ver las soluciones de los demás deberá renunciar al derecho de entregar una solución.Finalmente, podrá acceder a la pestaña de preguntas para listar únicamente las preguntas asociadas a dicha tarea. Todo esto se puede ver en la figura 7.10

## 7.4 Preguntas

Las preguntas son listadas, filtradas y creadas de una manera muy similar a las tareas, con la diferencia de que estas no tienen fecha límite, solo muestran la cantidad de respuestas que tiene una pregunta y si es una pregunta que tiene una respuesta correcta o no.

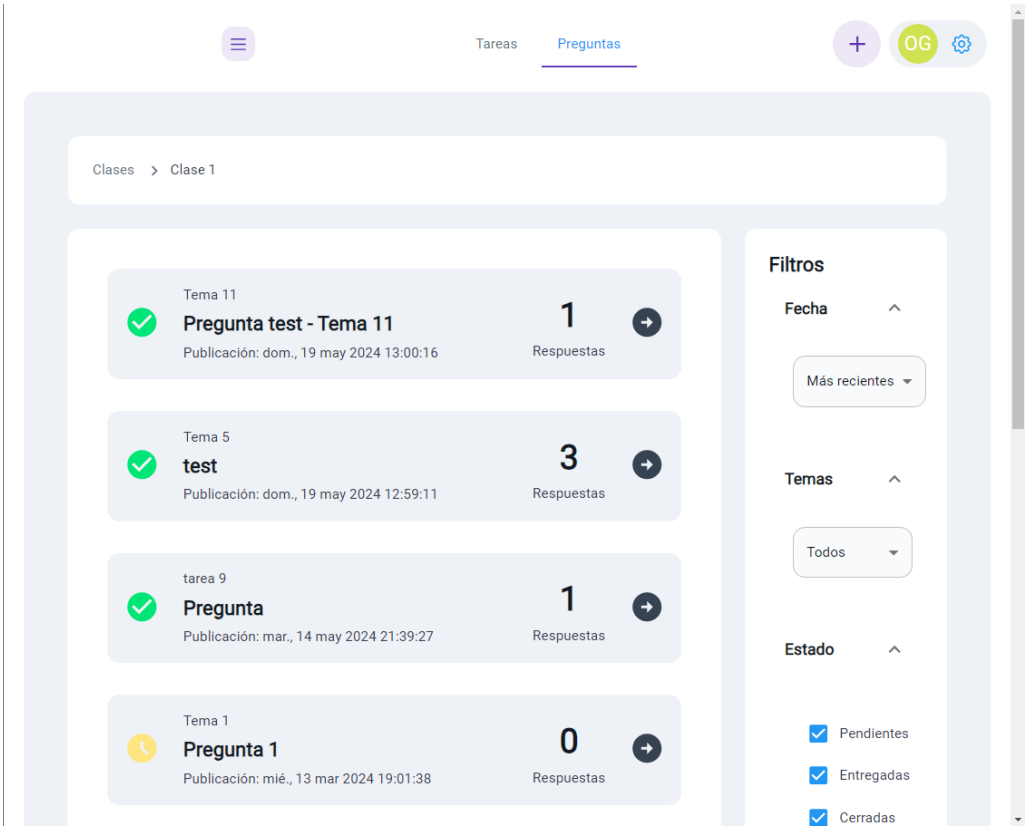



**Figura 7.11:** Listar preguntas

Para acceder a la creación de preguntas, se hace desde el botón con un ícono de cruz en la pestaña de preguntas como se muestra en la figura 7.12. Para crearla, el usuario podrá elegir el tema o la tarea relacionada, pero sólo una de estas opciones, como vemos en la figura 7.13.

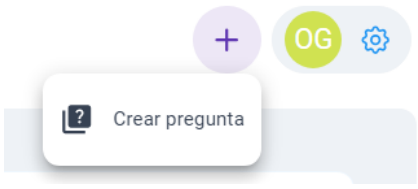



**Figura 7.12:** Opciones para crear del alumno

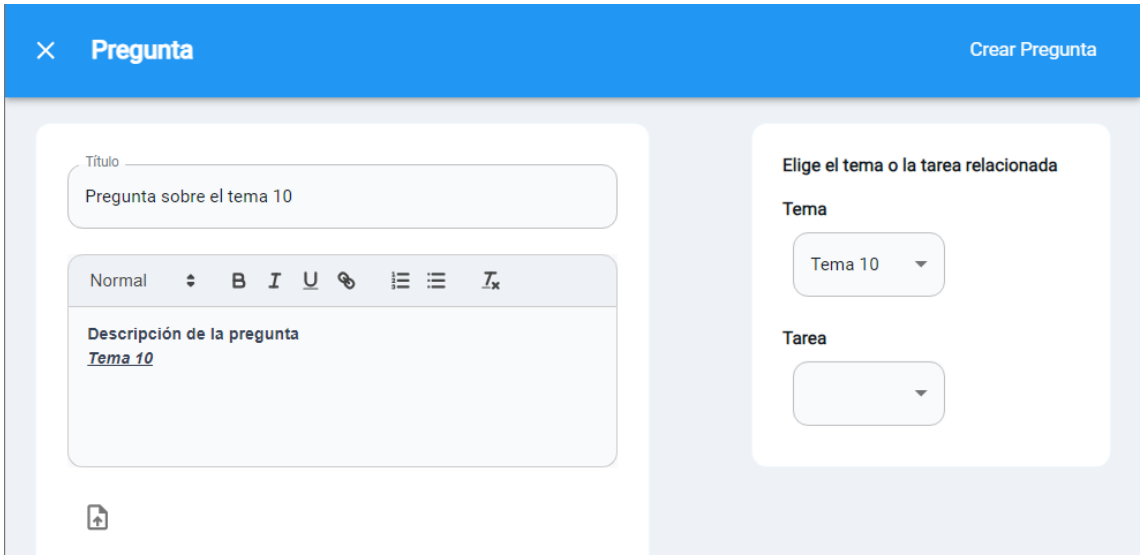



**Figura 7.13:** Crear pregunta

Al acceder a una pregunta, el usuario verá en primer lugar la respuesta correcta, después la suya y después el resto de respuestas, pudiendo desplegar los detalles de cada una al pulsar sobre el botón con un icono de flecha apuntando hacia abajo. Si ya ha respondido o ya hay una respuesta correcta no podrá entregar una respuesta.

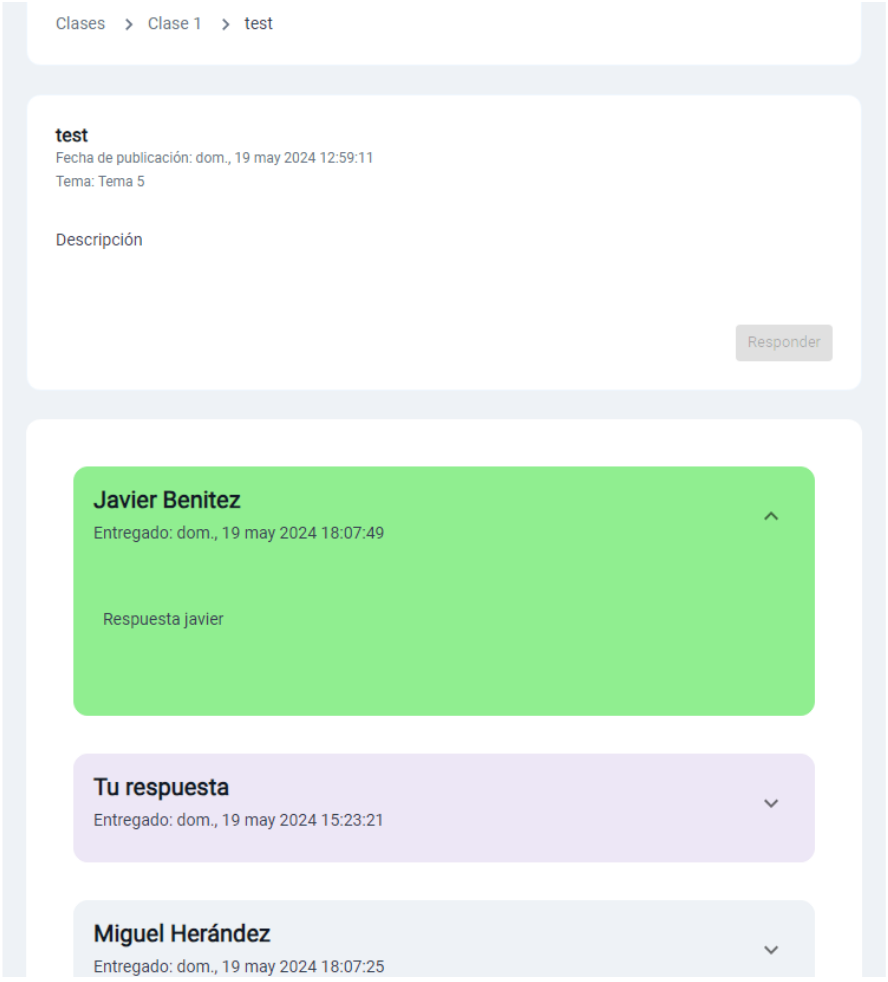



**Figura 7.14:** Ver pregunta

Si el usuario es el creador de la pregunta, podrá marcar una respuesta como correcta si aún no hay ninguna.

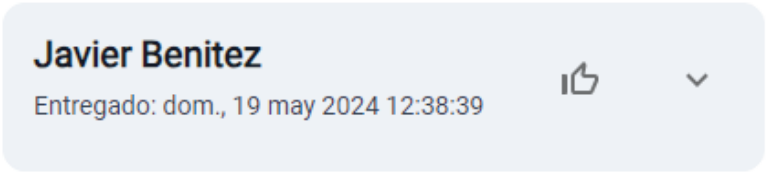



**Figura 7.15:** Marcar respuesta correcta

Tanto la creación de soluciones como de respuestas a preguntas consisten en cargar los archivos necesarios para apoyar la información añadida en la descripción.

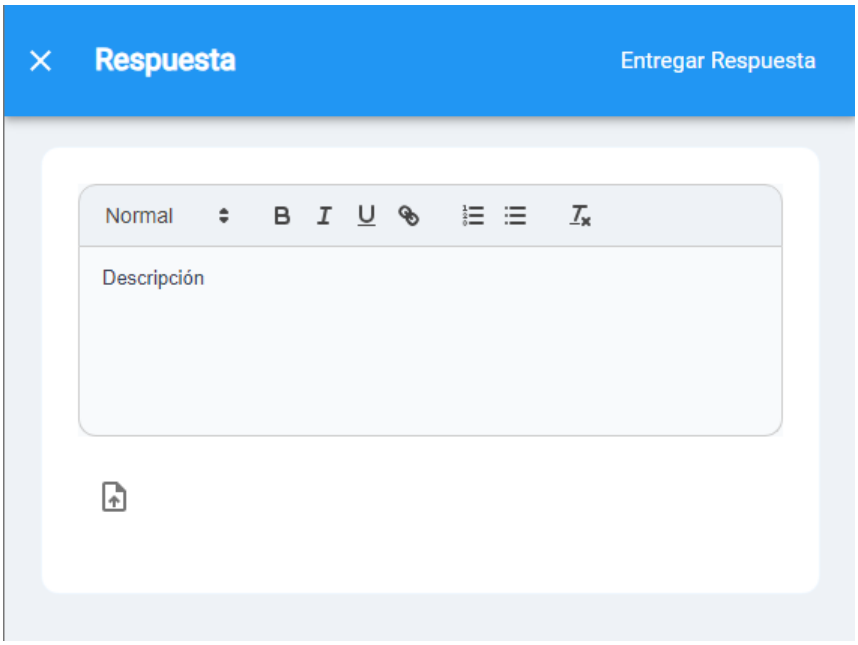



**Figura 7.16:** Entregar respuesta

## 7.5 Soluciones

Al seleccionar una solución, se podrá observar un listado de correcciones y entregar una corrección si no se ha entregado ya. La primera corrección listada será la del usuario que está viendo la aplicación. De igual manera que las respuestas, se podrán desplegar para ver en detalle, y además muestran la calificación elegida por el corrector.

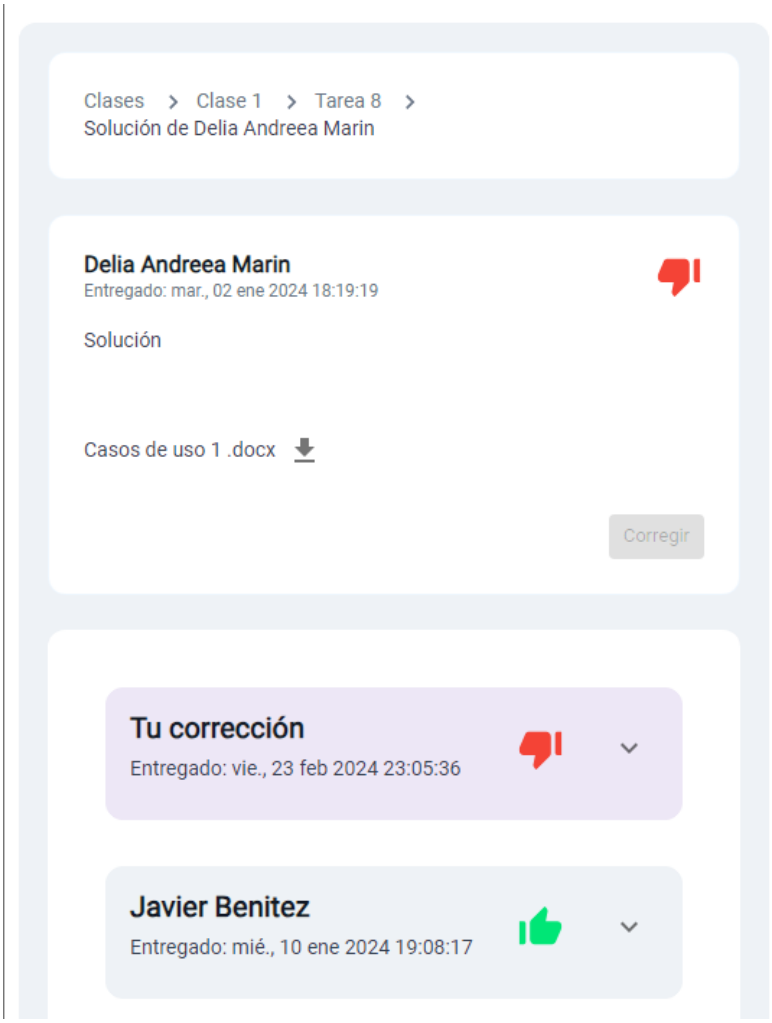
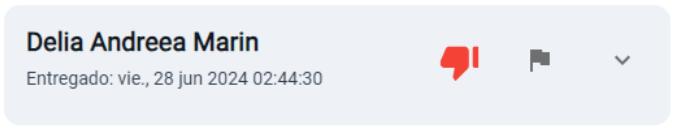
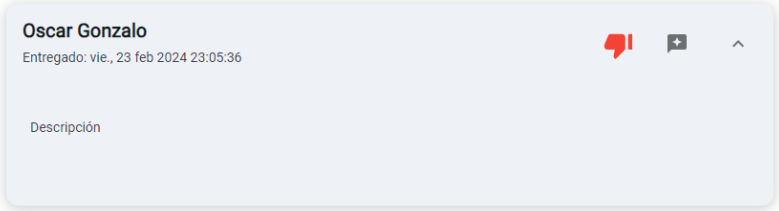



**Figura 7.17:** Ver solución

En caso de ser el usuario que ha creado la solución se podrá hacer una petición de revisión de una corrección, y en caso de ser un profesor se podrá eliminar la corrección, de manera que ya no influirá en la calificación final de la solución, o marcarla como revisada, para que ya no aparezca como pendiente de revisión.




**(a)** Corrección en vista alumno




**(b)** Corrección en vista profesor

**Figura 7.18:** Petición de revisión y revisar corrección

Crear una corrección es igual que crear una respuesta o solución, pero además se debe elegir una calificación.

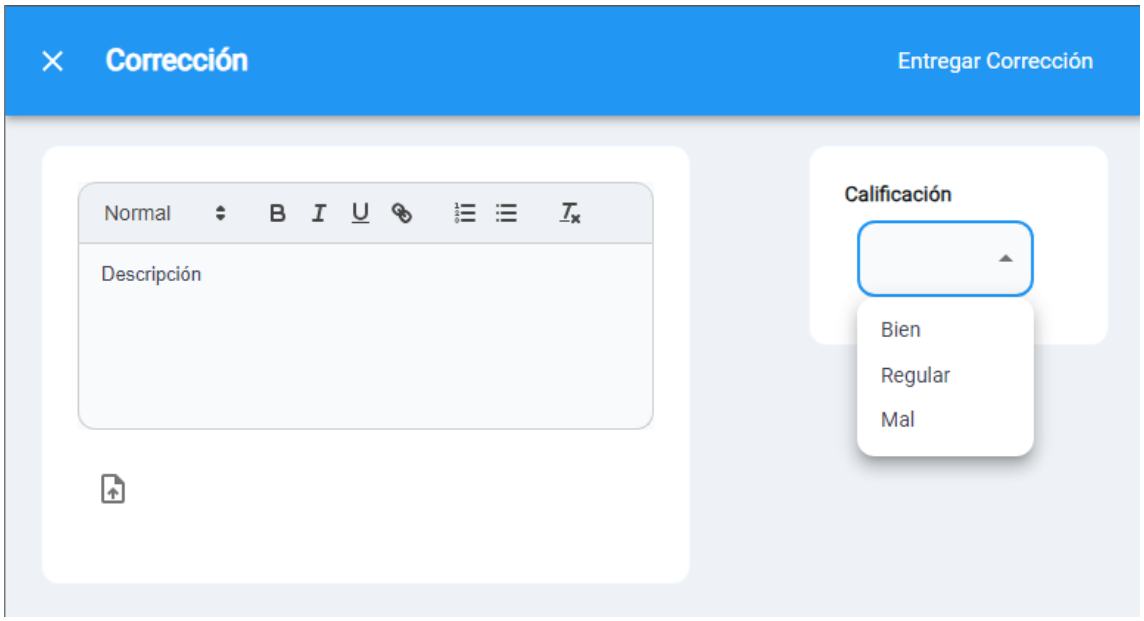



**Figura 7.19:** Crear corrección

## 7.6 Peticiones de revisión

Los profesores podrán ver las peticiones de revisión pendientes de una clase, accediendo desde el menú lateral. Estarán ordenadas por fecha de solicitud más antigua y mostrarán el nombre del usuario que la ha solicitado y la tarea a la que corresponde.

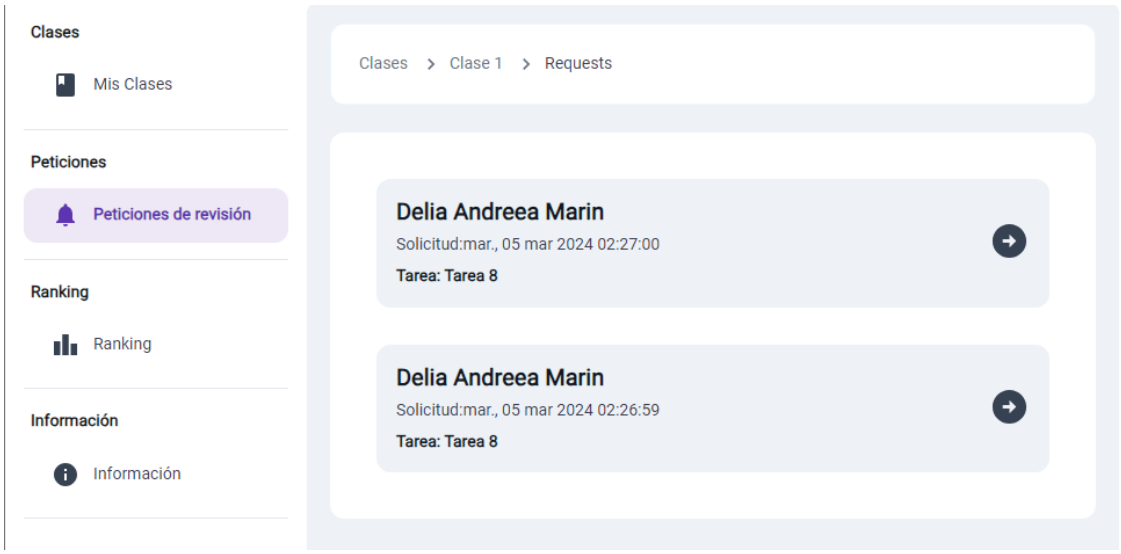



**Figura 7.20:** Listar peticiones de corrección

Al seleccionar una petición de revisión, la aplicación redirigirá al usuario a la solicitud correspondiente, desplazando verticalmente hasta mostrar la corrección que debe ser revisada resaltándola mediante una animación.

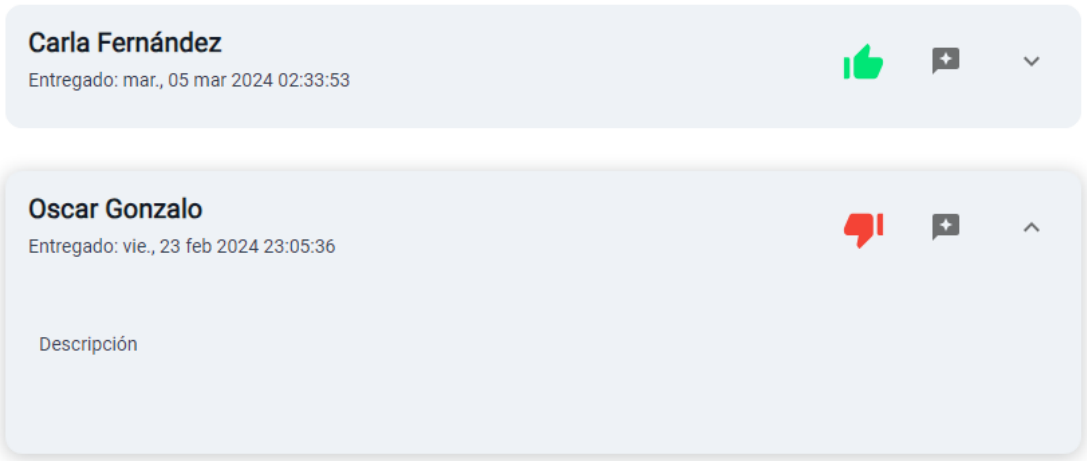



**Figura 7.21:** Revisión pendiente destacadan

## 7.7 Ranking

Todos los usuarios que pertenezcan a una clase podrán ver el *ranking* de la clase, consistente en una lista de alumnos ordenada por puntuación.

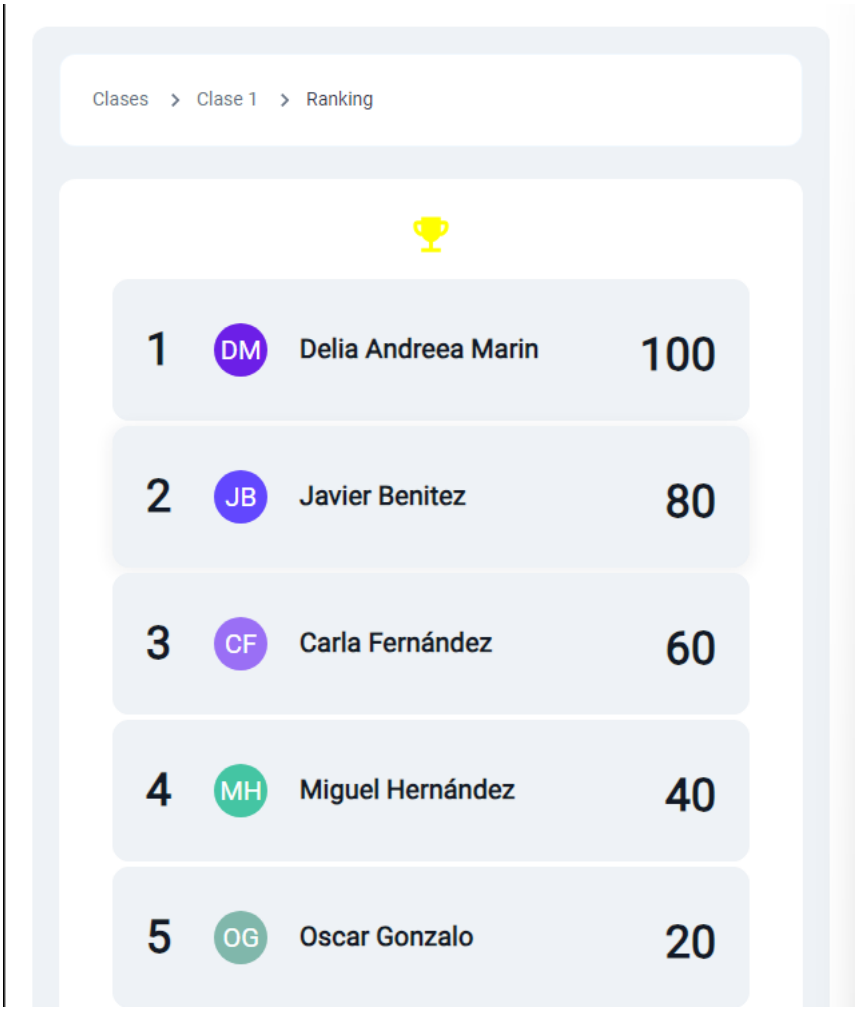



**Figura 7.22:** Ranking

# CAPÍTULO 8
# Conclusiones y trabajo futuro

## 8.1 Relación con los estudios cursados

Es claro que para la realización de este trabajo han sido fundamentales los conocimientos adquiridos en el Grado de Ingeniería Informática, especialmente en las asignaturas de la rama de Ingeniería del Software. A continuación, se explica cómo estos estudios han influido en el desarrollo del proyecto:

Las asignaturas de *Proceso del Software* y *Proyecto de Ingeniería del software* fueron cruciales para elegir una metodología adecuada entre las diversas opciones disponibles. Estas asignaturas facilitaron la evaluación de cada metodología, considerando sus limitaciones e impacto en el proyecto, y finalmente se seleccionó la metodología *SCRUM* como la más adecuada, cuya puesta en marcha se ha visto facilitada al haberse aplicado en *Proyecto de Ingeniería del software*.

Por otro lado, la asignatura de *Análisis y Estudio de Requisitos* permitió identificar y analizar todos los requisitos necesarios para la aplicación. Los conocimientos en *Diseño de Software* sobre patrones de diseño y principios de código limpio han sido aplicados en el proyecto para establecer una arquitectura adecuada y desarrollar un código legible, reusable y mantenible.

Finalmente, los aprendizajes en la asignatura *Bases de datos y sistemas de información* en cuanto a principios de diseño de bases de datos y cómo modelar las relaciones entre datos, ha ayudado a la implementación y diseño de una base de datos eficiente y robusta.

## 8.2 Conclusiones

Al finalizar este proyecto y hacer una retrospectiva sobre los objetivos establecidos inicialmente, podemos afirmar que hemos alcanzado el objetivo principal de desarrollar un MVP funcional para una aplicación web que no solo facilita la colaboración y comunicación entre estudiantes, sino que también integra un sistema de gamificación para aumentar la motivación y participación de los alumnos. Este MVP ha sido implementado efectivamente, permitiendo la realización completa de todas las operaciones CRUD sobre los elementos principales de la aplicación, lo que refleja la correcta aplicación de las técnicas de diseño e implementación aprendidas.

En cuanto a los aspectos técnicos, se ha logrado implementar una base de datos local robusta utilizando tecnologías avanzadas como Spring Boot y SQL. Esta implementación no solo ha permitido almacenar y gestionar de manera eficiente los datos, sino que también ha reforzado los conocimientos adquiridos en la carrera sobre la creación de servicios RESTful y la ejecución de consultas.

La adopción de pruebas automatizadas ha mejorado la eficiencia del proceso de desarrollo, permitiendo identificar problemas con mayor rapidez y precisión. Esto ha permitido no solo garantizar la funcionalidad y fiabilidad de la aplicación, sino también facilitar su mantenimiento continuo. La aplicación de una metodología ágil, *SCRUM*, ha sido fundamental en todo el proceso.

A nivel personal, el proyecto ha sido una oportunidad invaluable para profundizar en el estudio de tecnologías emergentes y consolidar la comprensión teórica mediante su aplicación práctica. El desafío de aprender sobre Spring Boot ha enriquecido significativamente mi perfil profesional, proporcionando una base sólida para futuros proyectos y oportunidades de empleo.

En conclusión, este proyecto no solo ha cumplido con sus objetivos académicos y técnicos, sino que también ha contribuido al desarrollo personal y profesional. Las habilidades y conocimientos adquiridos durante su ejecución serán de gran valor en la continuación de mi carrera en desarrollo de software.

## 8.3 Trabajo futuro

A pesar de que este proyecto se ha enfocado en alcanzar los objetivos y requisitos inicialmente establecidos, a lo largo de su desarrollo han emergido varias ideas y aspectos que podrían ser explorados y potencialmente integrados en futuras versiones del software.

El desarrollo de la interfaz de usuario se ha realizado teniendo en mente su utilización a través de un ordenador, por lo que un aspecto muy ideal a mejorar sería ajustar la interfaz para que sea responsiva e interactiva en diferentes tipos de pantalla, como dispositivos móviles. Esta tarea es interesante realizarla ya que así se aprovecharía al máximo el uso de Material UI, que ofrece muchas facilidades para hacer interfaces responsivas.

Para determinar si un usuario se puede registrar como profesor, la idea implementada es la utilización de un código único que se debe distribuir únicamente a los profesores elegibles y debe ser ingresado durante el proceso de registro. Esto se podría mejorar añadiendo una capa adicional de seguridad, implementando un sistema de aprobación manual donde un administrador debe verificar y aprobar cada registro de profesor. Para ello, se podría añadir un nuevo rol de administrador y desarrollar un panel de control administrativo en la aplicación desde el que gestionar y verificar manualmente las solicitudes de registro.

Actualmente los usuarios se registran y en su perfil únicamente pueden ver su puntuación, nombre y un icono generado automáticamente. En un futuro, se podrían implementar funcionalidades que permitan personalizar sus perfiles, modificando datos personales, añadiendo fotos de perfil u otro tipo de datos. Otra funcionalidad que sería interesante implementar es un sistema de notificaciones por correo en cuanto a cambios en recursos creados por el usuario, como respuestas a sus preguntas o correcciones a sus soluciones.

# Bibliografía


[1] F. Rodríguez and R. Santiago, "Cómo motivar a tu alumnado y mejorar el clima en el aula," *La Rioja: Digital-Text*, pp. 5–19, 2015.

[2] D. Dicheva, C. Dichev, G. Agre, and G. Angelova, "Gamification in education: A systematic mapping study," *Journal of educational technology & society*, vol. 18, no. 3, pp. 75–88, 2015.

[3] K. L. Beck, M. A. Beedle, A. van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries, J. Kern, B. Marick, R. C. Martin, S. J. Mellor, K. Schwaber, J. Sutherland, and D. A. Thomas, "Manifesto for agile software development," 2013.

[4] A. Dominguez, "¿Qué es Scrum?." https://www.albertodominguez.co/que-es-scrum/, 2020. [Accedido 03-mayo-2023].

[5] Jira, "Introduction to Jira Software." https://www.atlassian.com/software/jira/guides/getting-started/introduction#what-is-jira-software. [Accedido 03-mayo-2023].

[6] "Iso/iec/ieee international standard - systems and software engineering – life cycle processes – requirements engineering," *ISO/IEC/IEEE 29148:2018(E)*, pp. 1–104, 2018.

[7] Figma, "Prototyping." https://www.figma.com/prototyping/. [Accedido 04-mayo-2023].

[8] corewave, "How to Choose the Right Technology Stack for Your App." https://medium.com/@corewave/how-to-choose-the-right-technology-stack-for-your-app-251d2ad34eaa, 2024. [Accedido 02-junio-2024].

[9] W. Frakes and K. Kang, "Software reuse research: status and future," *IEEE Transactions on Software Engineering*, vol. 31, no. 7, pp. 529–536, 2005.

[10] D. Riehle, *Framework design: A role modeling approach*. PhD thesis, ETH Zurich, 2000.

[11] React, "React." https://es.react.dev/. [Accedido 02-junio-2024].

[12] A. Ravichandran, "React Virtual DOM Explained in Simple English." https://adhithiravi.medium.com/react-virtual-dom-explained-in-simple-english-fc2d0b277bc5. [Accedido 03-junio-2024].

[13] Spring, "Introduction to Spring Framework." https://docs.spring.io/spring-framework/docs/4.0.x/spring-framework-reference/html/overview.html. [Accedido 03-junio-2024].

[14] S. Boot, "Spring Boot Overview." https://docs.spring.io/spring-boot/index.html. [Accedido 03-junio-2024].

[15] MySQL, “MySQL.” https://www.oracle.com/mysql/. [Accedido 04-junio-2024].

[16] S. Overflow, “Stack overflow developer survey 2022: Most popular technologies - databases.” Stack Overflow Developer Survey: "https://survey.stackoverflow.co/2022/#section-most-popular-technologies-databases", 2022. "[Accedido 04-junio-2024]".

[17] GitHub, “GitHub Actions.” https://github.com/features/actions. [Accedido 05-junio-2024].

[18] I. Standard, “Software engineering–software life cycle processes–maintenance,” *ISO Standard*, vol. 14764, p. 2006, 2006.

[19] I. Docker, “Docker.” https://www.docker.com/what-docker. [Accedido 05-junio-2024].

# APÉNDICE A
# Prototipado de pantallas

En este apéndice se aportan algunos prototipos que representan aquellos casos de uso no cubiertos por los prototipos mencionados en la memoria para un mayor entendimiento de cómo debe ser la interfaz de usuario y a qué caso de uso corresponde cada funcionalidad.

## A.1 Preguntas

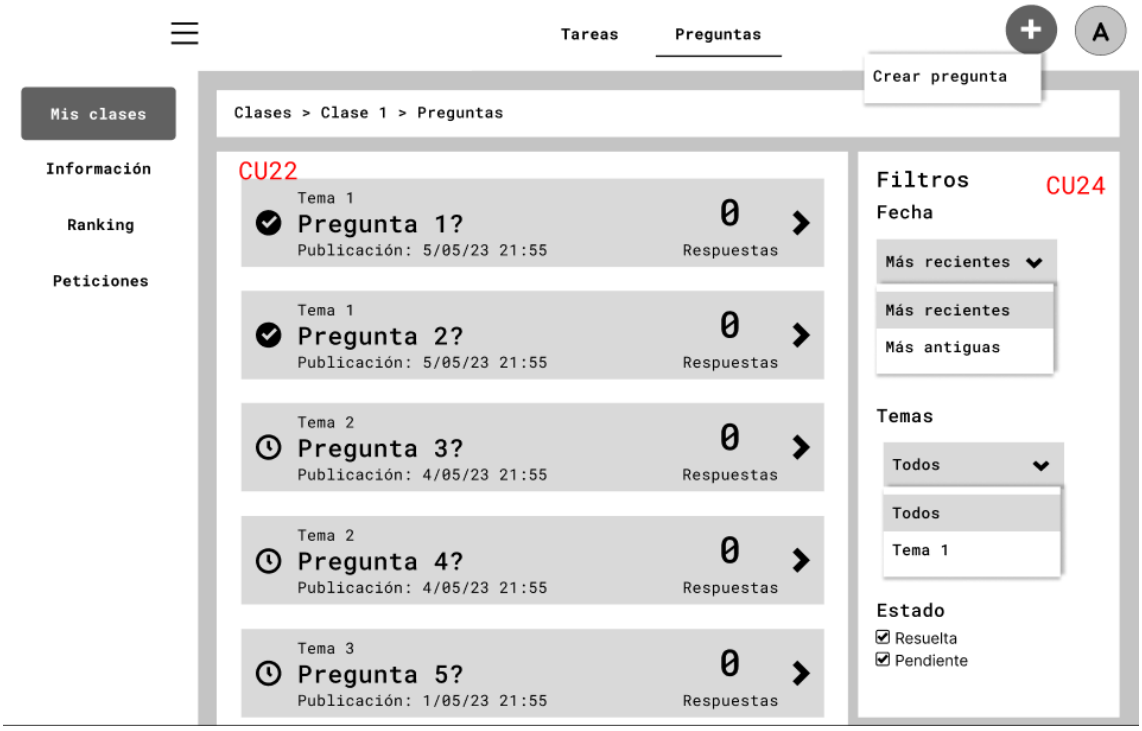



**Figura A.1:** Prototipo de la vista de listar preguntas

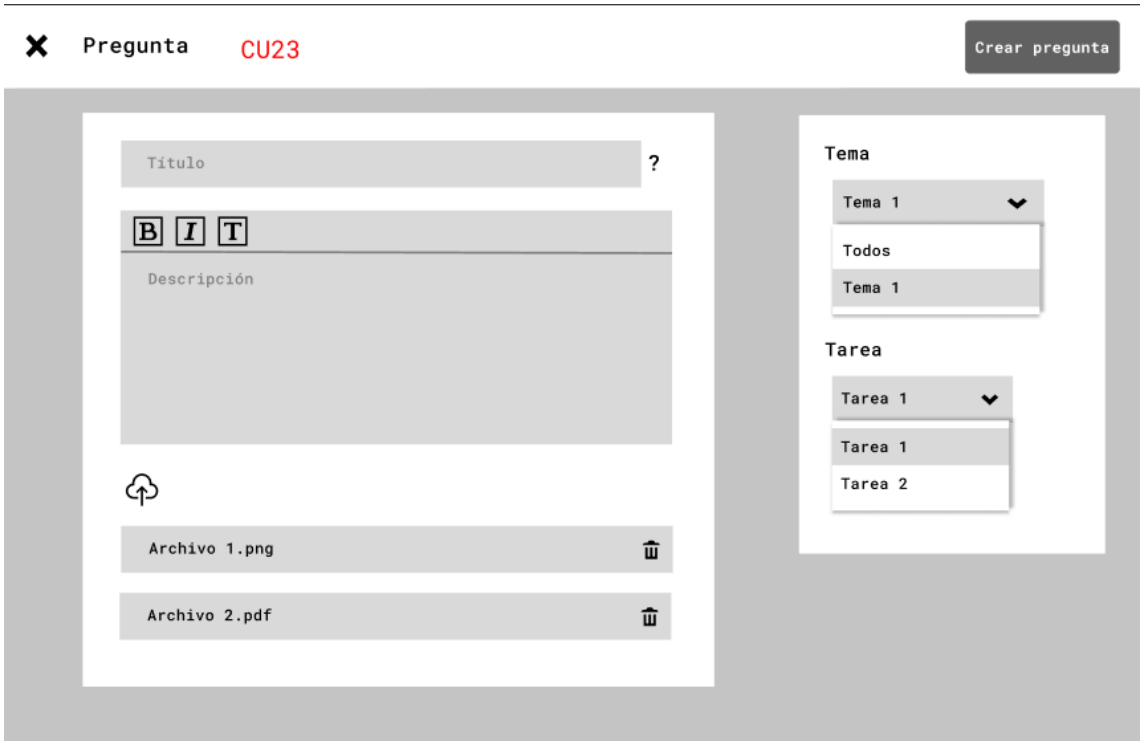



**Figura A.2:** Prototipo de la vista de crear pregunta

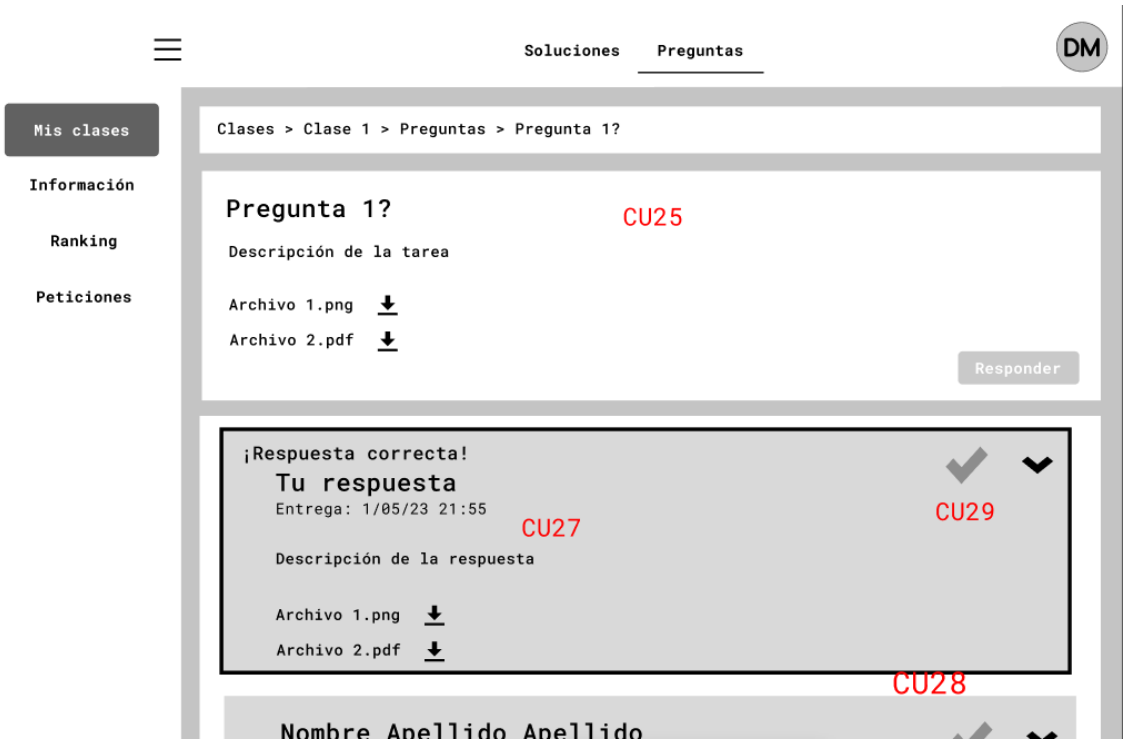



**Figura A.3:** Prototipo de la vista de una pregunta

## A.2 Soluciones

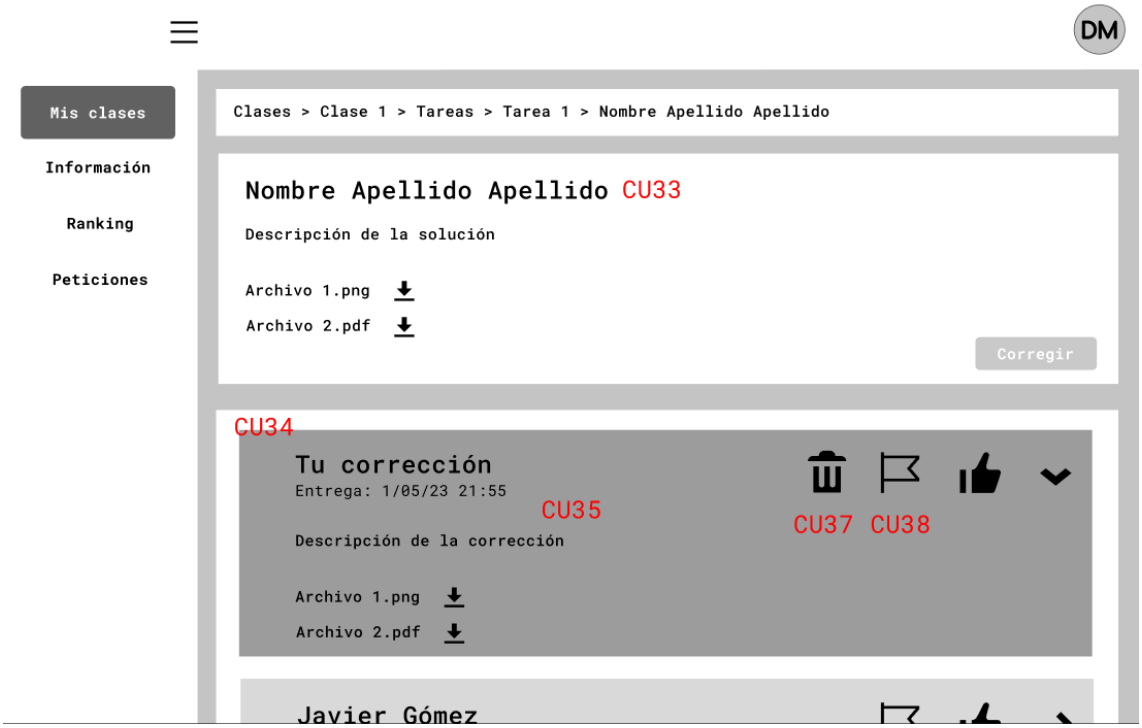



**Figura A.4:** Prototipo de la vista de una solución


**Figura A.5:** Prototipo de la vista de entregar respuesta o solución

## A.3 Correcciones

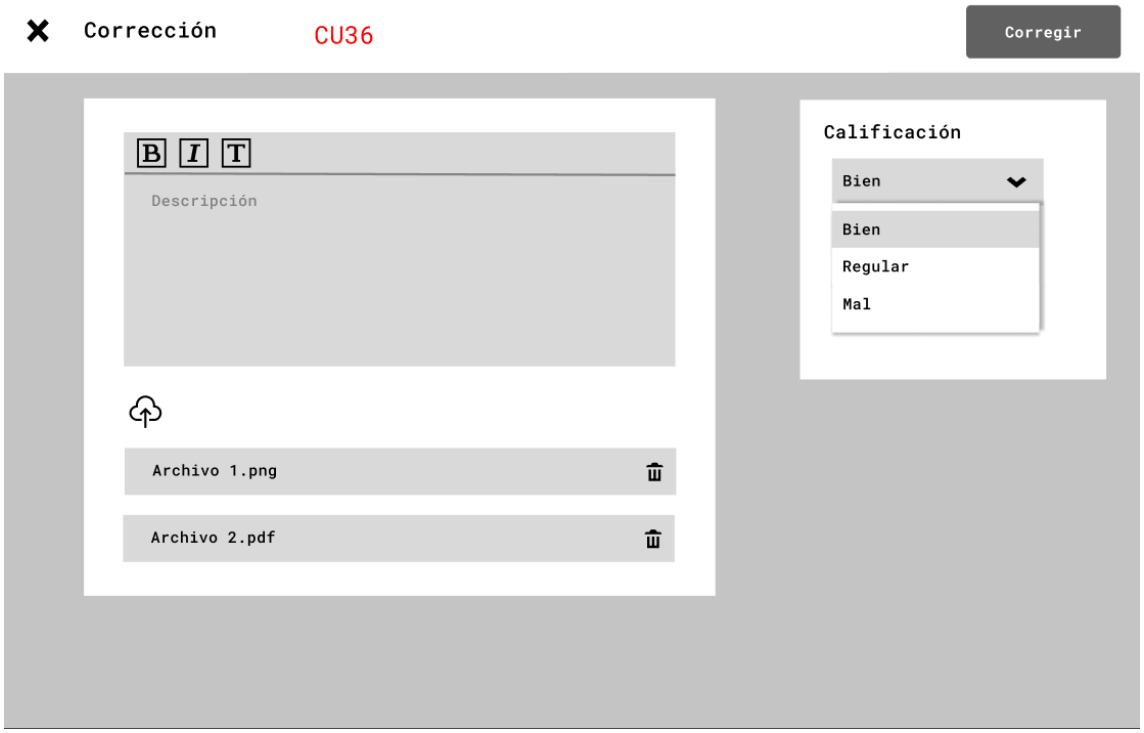



**Figura A.6:** Prototipo de la vista de entregar corrección

# APÉNDICE B
# Fragmentos de código

## B.1 Código de controlador de clases

En este apartado podemos ver algunos de los métodos del controlador utilizado para definir el manejo de aquellos datos relacionados con las clases. Podemos observar el uso de las anotaciones de cada método HTTP, así como las necesarias para definir y configurar el controlador, las rutas, y los roles autorizados.

```
@RestController
@CrossOrigin
@RequestMapping("/class")
public class ClassController {
    @Autowired
    private ClassService classService;

    @Autowired
    private StudentService studentService;

    @Autowired
    private TeacherRepository teacherRepository;
    @Autowired
    private StudentRepository studentRepository;

    @PostMapping("/add")
    @PreAuthorize("hasRole('ROLE_TEACHER')")
    public ResponseEntity<Class> add(@RequestBody AddRequest request){
        Class classObj = new Class(request.getClassName());
        Class finalClassObj = teacherRepository.findById(request.
    getTeacherId()).map(teacher -> {
            classObj.setTeacher(teacher);
            return classService.saveClass(classObj);
        }).orElseThrow(() -> new ResourceNotFoundException("Not found
    Teacher with id = " + request.getTeacherId()));

        return new ResponseEntity<>(finalClassObj, HttpStatus.CREATED);
    }

    @GetMapping("/getClass")
    public ResponseEntity<Class> getClass(@RequestParam Integer classId){
        try {
            Class classObj = classService.findById(classId);
            return ResponseEntity.ok().body(classObj);
        } catch (ChangeSetPersister.NotFoundException ex) {
            return ResponseEntity.status(HttpStatus.NOT_FOUND).build();
        }
    }

```

```
    @DeleteMapping("/classes/{classId}")
    @PreAuthorize("hasRole('ROLE_TEACHER')")
    public ResponseEntity<Void> delete(@PathVariable Long classId) {
        classService.delete(classId);
        return ResponseEntity.noContent().build();
    }

    @PutMapping("/edit")
    @PreAuthorize("hasRole('ROLE_TEACHER')")
    public ResponseEntity<String> updateClass(@RequestBody Class classObj)
    {
        try {
            classService.updateClass(classObj);
            return ResponseEntity.ok().body("Updated correctly");
        } catch (ChangeSetPersister.NotFoundException ex) {
            return ResponseEntity.status(HttpStatus.NOT_FOUND).build();
        }
    }
```

## B.2 Código del planificador de puntuación

En este apartado se muestra la implementación de la clase encargada de actualizar la puntuación de los estudiantes.

```
@Component
public class ScoreScheduler {

    @Autowired
    private TaskService taskService;

    @Autowired
    private StudentService studentService;

    @Scheduled(cron = "0 0 2 * * *")
    public void execute() throws InterruptedException {
        List<Task> taskList = taskService.getTasksByDateBefore(
    LocalDateTime.now());
        for (Task task: taskList){
            if(!task.isScoreCalculated()){
                for (Solution solution : task.getSolutions()){
                    addScoreToSolvingUser(solution.getQualification(),
    solution.getStudent());
                    addScoreToCorrectors(solution.getQualification(),
    solution.getCorrections());
                }
                task.setScoreCalculated(true);
                taskService.saveTask(task);
            }
        }

    }

    private void addScoreToSolvingUser(EQualification qualification,
    Student student){
        double score = student.getScore();
        if(qualification == null) return;
        switch (qualification){
            case GOOD -> score+=100;
            case FAIR -> score+=50;
            case POOR -> score-=25;
        }
```

```
34          student.setScore(score);
35          studentService.saveStudent(student);
36      }
37
38      private void addScoreToCorrectors(EQualification qualification, List<
        Correction> corrections){
39          if(qualification == null) return;
40          for (Correction correction: corrections){
41              User user = correction.getUser();
42              if(user instanceof Student){
43                  Student student = (Student) user;
44                  double score = student.getScore();
45                  switch (qualification){
46                      case GOOD -> {
47                          switch (correction.getQualification()){
48                              case GOOD -> score+=25;
49                              case FAIR -> {}
50                              case POOR -> score-=15;
51                          }
52                      }
53                      case FAIR -> {
54                          switch (correction.getQualification()){
55                              case FAIR -> score+=25;
56                              case GOOD -> {}
57                              case POOR -> {}
58                          }
59                      }
60                      case POOR -> {
61                          switch (correction.getQualification()){
62                              case POOR -> score+=25;
63                              case FAIR -> {}
64                              case GOOD -> score-=15;
65                          }
66                      }
67                  }
68                  student.setScore(score);
69                  studentService.saveStudent(student);
70              }
71          }
72      }
73  }
```

## B.3 Código del modelo de Tarea

En esta sección podemos ver parte de cómo se ha definido la entidad Tarea.

```
1       @Entity
2   @Table(name="task")
3   @JsonIgnoreProperties(value = {"lesson, classObj, files"})
4   public class Task {
5
6       @Id
7       @GeneratedValue(strategy = GenerationType.IDENTITY)
8       private int id;
9       private String title;
10
11      @Column(columnDefinition="LONGTEXT")
12      private String description;
13      @DateTimeFormat(iso=DateTimeFormat.ISO.DATE_TIME)
14      private LocalDateTime dateTime;
15
16      @ManyToOne(fetch = FetchType.LAZY, optional = false)
```

```
    @JoinColumn(name = "class_id", nullable = false)
    @OnDelete(action = OnDeleteAction.CASCADE)
    @JsonIgnore
    private Class classObj;

    @ManyToOne(fetch = FetchType.LAZY, optional = false)
    @JoinColumn(name = "lesson_id", nullable = false)
    @OnDelete(action = OnDeleteAction.CASCADE)
    @JsonIgnore
    private Lesson lesson;

    @OneToMany(mappedBy = "task", cascade = CascadeType.ALL, orphanRemoval
    = true)
    @JsonIgnore
    private Set<FileDB> files = new HashSet<>();

    @OneToMany(mappedBy = "task", cascade = CascadeType.ALL, orphanRemoval
    = true, fetch = FetchType.EAGER)
    @JsonIgnore
    private Set<Solution> solutions = new HashSet<>();

    @OneToMany(mappedBy = "task", cascade = CascadeType.ALL, orphanRemoval
    = true)
    @JsonIgnore
    private Set<Question> questions = new HashSet<>();

    @JsonIgnore
    private boolean scoreCalculated;

    public Task(){

    }
    public Task(String title, String description, LocalDateTime dateTime,
    Class classObj, Lesson lesson){
        this.title = title;
        this.description = description;
        this.dateTime = dateTime;
        this.classObj = classObj;
        this.lesson = lesson;
        this.scoreCalculated = false;
    }

    .... getters y setters....
```

## B.4 Código del repositorio de corrección

En esta sección podemos ver cómo se ha declarado el repositorio correspondiente a la entidad correccion, y además se ha añadido un método personalizado.

```
@Repository
public interface CorrectionRepository extends JpaRepository<Correction,
    Integer> {
    @Query("SELECT c FROM Correction c WHERE c.solution.id = :solutionId
    AND c.removed=false ORDER BY c.qualification ASC")
    List<Correction> findAllBySolutionId(@Param("solutionId")Integer
    solutionId);

}
```

## B.5 Tests del servicio de tareas

En este apartado se muestra la implementación de algunas de las pruebas unitarias utilizadas para probar el servicio encargado de la entidad 'Tarea'.

```
@RunWith(MockitoJUnitRunner.class)
public class TaskServiceTest {

    @Mock
    private TaskRepository taskRepository;

    @InjectMocks
    private TaskServiceImpl taskService;

    @Test
    public void testSaveTask() {
        //Arrange
        Task task = new Task();
        when(taskRepository.save(any(Task.class))).thenReturn(task);

        //Act
        Task savedTask = taskService.saveTask(task);

        //Assert
        verify(taskRepository).save(task);
        assertEquals(task, savedTask);
    }

    @Test
    public void testUserHasAnswered() {
        // Arrange
        Task task = new Task();
        Student student = new Student();
        student.setId(1L);
        Solution solution = new Solution();
        solution.setStudent(student);

        Set<Solution> solutions = Collections.singleton(solution);
        task.setSolutions(solutions);

        // Act
        Optional<Solution> optionalSolution = taskService.hasUserAnswered(1
    L, task);

        // Assert
        assertTrue(optionalSolution.isPresent());
        assertEquals(solution, optionalSolution.get());
    }
    @Test
    public void testIsTaskActive() {
        // Arrange
        Task task = new Task();
        LocalDateTime dateTime = LocalDateTime.now().plusMinutes(5);
        task.setDateTime(dateTime);

        // Act
        boolean isActive = taskService.isTaskActive(task);

        //Assert
        assertTrue(isActive);

    @Test
    public void testUserCorrectionsInTask() {
        // Arrange
```

```
59          Task task = new Task();
60          Solution solution = new Solution();
61          Correction correction1 = new Correction();
62          User user = new User();
63          user.setId(1L);
64          correction1.setUser(user);
65          solution.setCorrections(Collections.singletonList(correction1));
66          task.setSolutions(Collections.singleton(solution));
67
68          // Act
69          int count = taskService.userCorrectionsInTask(1L, task);
70
71          //Assert
72          assertEquals(1, count);
73      }
```

# APÉNDICE C
# Archivos Docker

## C.1 Dockerfile backend

En este apartado podemos ver la definición del *Dockerfile* utilizado para el backend.

```
FROM openjdk:19-alpine AS build
ADD target/backend.jar backend.jar
ENV WAIT_VERSION 2.7.2
ADD https://github.com/ufoscout/docker-compose-wait/releases/download/
$WAIT_VERSION/wait /wait
RUN chmod +x /wait
EXPOSE 8080
```

## C.2 Dockerfile frontend

En este apartado podemos ver la definición del *Dockerfile* utilizado para el frontend.

```
FROM node:14-alpine
WORKDIR /app
COPY package*.json ./
RUN npm install
COPY . .
EXPOSE 3000
CMD ["npm", "start"]
```

## C.3 Docker compose

En este apartado podemos ver la definición del *Docker Compose*.

```
version: "3"
services:
  db:
    image: mysql
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD: mypassword
      MYSQL_DATABASE: database
    ports:
      - "3306:3306"
    networks:
      - my-network
    volumes:
```

```
      - ./init.sql:/docker-entrypoint-initdb.d/init.sql

  app:
    build:
      context: ./backend
      dockerfile: Dockerfile
    ports:
      - "8080:8080"
    networks:
      - my-network
    depends_on:
      - db
    environment:
      WAIT_HOSTS: db:3306
      WAIT_HOSTS_TIMEOUT: 300
      WAIT_SLEEP_INTERVAL: 30
      WAIT_HOST_CONNECT_TIMEOUT: 30
      SPRING_DATASOURCE_URL: jdbc:mysql://db:3306/database?
    allowPublicKeyRetrieval=true&useSSL=false
      SPRING_DATASOURCE_USERNAME: root
      SPRING_DATASOURCE_PASSWORD: mypassword
    command: sh -c "/wait && java -jar backend.jar"

  frontend:
    build:
      context: ./frontend
      dockerfile: Dockerfile
    ports:
      - "3000:3000"
    networks:
      - my-network
    environment:
      API_URL: http://localhost:8080
    command: npm start

  phpmyadmin:
    image: phpmyadmin/phpmyadmin
    container_name: my-phpmyadmin
    environment:
      PMA_HOST: db
      MYSQL_ROOT_PASSWORD: mypassword
    restart: always
    ports:
      - 8081:80
    networks:
      - my-network

networks:
  my-network:
```

# APÉNDICE D
# Objetivos de Desarrollo Sostenible

Este apéndice está dedicado a explicar cómo se relaciona el proyecto con los Objetivos de Desarrollo Sostenible (ODS).

| **Objetivos de Desarrollo Sostenible** | **Alto** | **Medio** | **Bajo** | **No Procede** |
|---|---|---|---|---|
| ODS 1. **Fin de la pobreza.** | | | | X |
| ODS 2. **Hambre cero.** | | | | X |
| ODS 3. **Salud y bienestar.** | | | | X |
| ODS 4. **Educación de calidad.** | X | | | |
| ODS 5. **Igualdad de género.** | | | | X |
| ODS 6. **Agua limpia y saneamiento.** | | | | X |
| ODS 7. **Energía asequible y no contaminante.** | | | | X |
| ODS 8. **Trabajo decente y crecimiento económico.** | | | X | |
| ODS 9. **Industria, innovación e infraestructuras.** | | X | | |
| ODS 10. **Reducción de las desigualdades.** | | X | | |
| ODS 11. **Ciudades y comunidades sostenibles.** | | | | X |
| ODS 12. **Producción y consumo responsables.** | | | | X |
| ODS 13. **Acción por el clima.** | | | | X |
| ODS 14. **Vida submarina.** | | | | X |
| ODS 15. **Vida de ecosistemas terrestres.** | | | | X |
| ODS 16. **Paz, justicia e instituciones sólidas.** | | | | X |
| ODS 17. **Alianzas para lograr objetivos.** | | | | X |

**Tabla D.1:** Tabla de grados de relación del proyecto con los ODS.

Al tratarse de una herramienta enfocada al uso en un entorno educativo el objetivo de desarrollo sostenible que se ve directamente afectado por la aplicación desarrollada es el *4. Educación de calidad*. Al tratarse de una plataforma que ofrece la posibilidad de interactuar y colaborar con el objetivo de conseguir un mayor entendimiento de los contenidos, la calidad del estudio se ve directamente incrementada. Además, gracias a la posibilidad de corrección y resolución de dudas se fomenta el desarrollo de habilidades críticas como la evaluación crítica y la retroalimentación constructiva.

Por otra parte, y también contribuyendo en cierta manera al ODS anterior, la aplicación integra mecanismos para que los profesores revisen y recalculen calificaciones en caso de disconformidad, asegurando así una evaluación académica más justa y garantizando una educación inclusiva y equitativa, lo que se puede relacionar con el ODS *10. Reducción de las desigualdades*. Otro aspecto con el que se contribuye a este ODS es el hecho de proporcionar acceso a recursos educativos a estudiantes que se encuentren en cualquier condición o contexto social, con el único requisito de tener acceso a internet. Además, permite la interacción entre estudiantes de diversos orígenes, lo que puede ayudar a reducir prejuicios y barreras sociales.

Con respecto al ODS *9. Industria, innovación e infraestructuras* la aplicación contribuye a la innovación en el ámbito educativo al tratarse de una herramienta tecnológica que facilita el aprendizaje de una manera más eficente y posibilita diferentes métodos de enseñanza, lo cual también apoya el desarrollo de una infraestructura educativa más robusta y adaptable.

Finalmente, se puede encontrar cierta relación con el ODS *8. Trabajo decente y crecimiento económico* ya que todos los aspectos mencionados anteriormente proveen a los estudiantes con habilidades que les pueden ayudar a adaptarse mejor al mercado laboral, como el pensamiento crítico, la colaboración, o la retroalimentación, promoviendo así el crecimiento económico y el trabajo decente.