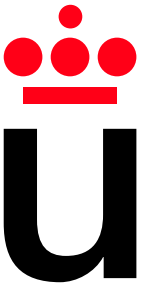

Universidad
Rey Juan Carlos

Escuela Técnica Superior
de Ingeniería Informática

Grado en Ingeniería del Software

Curso 2023-2024

Trabajo Fin de Grado

PLATAFORMA DE CORRECCIÓN DE EJERCICIOS ABIERTOS CON INTEGRACIÓN EN MOODLE

Autor: Brais Cabo Felpete
Tutor: Óscar Soto Sánchez,
Micael Gallego Carrillo

Agradecimientos

A mis padres, por su incansable esfuerzo y dedicación en darme lo mejor cada día, y por ser el pilar fundamental que me ha permitido alcanzar todo lo que he logrado.

A Ingrid, por su ayuda y su apoyo incondicional en cada paso de este viaje.

A mis amigos y compañeros de clase, por los momentos compartidos.

A Sergio, por haber compartido esta aventura a mi lado.

Y, por último, a mis tutores de proyecto, Micael Gallego y Óscar Soto, por su guía y asesoramiento.

Resumen

En esta memoria se documenta el trabajo realizado para el desarrollo de una plataforma de corrección de ejercicios abiertos con integración en Moodle, de adelante PCEAIM, que consiste en un aula virtual realizada usando las últimas tecnologías web.

PCEAIM surge como una aplicación para proporcionar al personal docente una forma rápida y sencilla de realizar exámenes, descargar preguntas, etc... También surge como una base para poder integrar en un futuro distintas herramientas y funcionalidades que permitan aligerar la carga de trabajo de los profesores y mejorar la calidad docente.

La aplicación consiste en un aula virtual implementada como una aplicación web que permite las funciones más comunes de este tipo de aplicaciones de forma que se disponga de un servicio rápido y eficiente a la vez que está disponible a través de un navegador web accesible de forma sencilla para la gran mayoría de los dispositivos.

A lo largo de este documento se explican las características técnicas así como las motivaciones y objetivos a la hora de realizar la aplicación, por lo que se encontrarán las siguientes secciones:

1. Introducción: en esta sección se realizará una breve introducción y explicación en términos generales sobre la aplicación.
2. Objetivos: en esta sección se comentarán los objetivos que se pretenden cumplir con el desarrollo de la aplicación. Para realizar esto se desgranarán los objetivos que buscan los distintos de usuarios de la aplicación a la hora de utilizarla.
3. Tecnologías, herramientas y metodologías: en esta sección se comentarán los aspectos técnicos de la aplicación así como la forma en la que se ha trabajado durante su desarrollo.
4. Descripción informática: en esta sección se comentarán los requisitos que se han seguido a la hora de la implementación, la arquitectura de la aplicación, la implementación del protocolo LTI 1.3 y las pruebas seguidas para asegurar la calidad de la misma.

5. Conclusiones y trabajos futuros: en esta sección se realiza una conclusión sobre el trabajo realizado y se analizan los trabajos que se realizarán en el futuro con el objetivo de seguir incluyendo funcionalidades que aporten valor a los usuarios.

Palabras clave:

- Typescript
- Web
- Bases de datos
- Spring Boot
- Angular
- LTI
- Moodle

Índice de contenidos

1 Introducción

Uno de los principales factores que ha permitido a los humanos convertirse en la especie dominante de la tierra es la transmisión de conocimientos entre las distintas generaciones. Esto permite que el conocimiento de una persona pueda pasar a sus descendientes en un periodo relativamente corto del tiempo por lo que dichos descendientes pueden continuar mejorando y refinando las habilidades y conocimientos que han aprendido para así producir avances tecnológicos, científicos y sociales.

Hoy en día, dicha transmisión de conocimiento se lleva a cabo dentro de aulas, colegios y universidades, donde existen personas dedicadas únicamente a la transmisión de conocimientos a las futuras generaciones.

Dentro de las aulas de estas instituciones es importante que la transmisión de conocimientos se realice de forma adecuada. La necesidad de conocer la eficacia de este proceso ha impulsado a la realización de evaluaciones por parte del personal docente para saber si sus alumnos están adquiriendo los conocimientos esperados.

Teniendo en cuenta lo comentado anteriormente, surge PCEAIM, que permite al profesor crear preguntas, tanto de texto como incluso pequeños fragmentos de código, y realizar la corrección de los ejercicios propuestos para así conocer si la información se ha transmitido correctamente. Además permite incluir un comentario sobre dichas tareas para que los alumnos puedan conocer sus puntos débiles y subsanarlos.

1.1. Contexto y alcance

En el ámbito docente, la tecnología ha demostrado ser valiosa ya que permite hacer más eficientes los procesos de enseñanza y aprendizaje. La aplicación PCEAIM surge por la necesidad de optimizar la gestión del tiempo y los recursos del entorno educativo, proporcionando una forma simple y rápida de manejar los procesos que tradicionalmente han supuesto un mayor consumo de tiempo al personal docente.

El principal objetivo que se persigue con PCEAIM es la mejora de la eficiencia a la hora de administrar evaluaciones y tareas, permitiendo a los educadores hacer una gestión efectiva del tiempo destinado a revisar y calificar las actividades estudiantiles.

Esta aplicación se ha desarrollado pensando en constituir un sistema completo que ofrece soluciones a los desafíos comunes en la educación, como la necesidad de proveer retroalimentación constructiva a los estudiantes de manera oportuna, así como la gestión rápida y eficiente de los historiales académicos.

PCEAIM proporciona funcionalidades para la creación, realización, y calificación de actividades educativas. Esto promueve prácticas de enseñanza más accesibles al ser más sencilla de utilizar que la gran mayoría de las aplicaciones de enseñanza de la actualidad. La aplicación se ha diseñado pensando en la facilidad de uso, asegurando que todo tipo de docentes, incluso aquellos que no presentan un perfil tecnológico, puedan aprovechar sus beneficios sin necesidad de una extensa curva de aprendizaje.

Además, PCEAIM se centra en la retroalimentación que proporciona el profesor al estudiante una vez ha revisado y calificado su tarea. A través de la aplicación, los docentes pueden calificar las tareas y proporcionar comentarios a los alumnos, facilitando la comunicación entre los principales involucrados del proceso de docencia. Esta característica es fundamental para el desarrollo académico de los estudiantes, ya que les permite conocer cuales son los puntos en los que se deben centrar a la hora de mejorar sus habilidades.

2

Objetivos

El propósito fundamental que persigue esta plataforma es el desarrollo de una aplicación web capaz de facilitar y mejorar la labor de los educadores. Con este objetivo, se detallarán los diferentes implicados en la aplicación en las secciones mencionadas a continuación:

- Docentes: desean poder lanzar preguntas a los alumnos de forma rápida y que sean corregidas automáticamente en tiempo real para poder ver los resultados y centrarse más en los conceptos que peor han entendido los alumnos. Sus principales objetivos son los siguientes:
  - Posibilidad de crear preguntas
  - Corrección de preguntas automáticas
  - Conexión a otras plataformas de docencia creadas en Moodle
  - Posibilidad de importar preguntas de otras plataformas a través de archivos de .csv
  - Posibilidad de exportar los resultados de un examen en un archivo .csv
  - Posibilidad de establecer una calificación a las actividades realizadas por los alumnos
  - Posibilidad de crear actividades para la entrega de ficheros por parte de los alumnos
- Equipo de desarrollo: desea que la plataforma sea fiable, usable fácilmente y robusta, además de a prueba de fallos, los cuales se pueden producir en todos los niveles de la aplicación.
- Alumnos: los alumnos desean poder acceder a la plataforma de forma sencilla e intuitiva, además de poder realizar las actividades y comprobar sus calificaciones. Estos objetivos se resumen en los siguientes:

- Acceso a la plataforma a través del aula virtual de su entidad docente.
- Realización de las actividades designadas por el profesor.
- Visualización de las calificaciones en la plataforma de su entidad docente.

3

Tecnologías, herramientas y metodologías

En esta sección se detallan y analizan los lenguajes y tecnologías utilizados, junto con las herramientas y metodologías empleadas para llevar a cabo la realización del proyecto.

3.1. Lenguajes

Esta sección está destinada al análisis de los lenguajes utilizados para la implementación de cada una de las partes que componen la aplicación. Para ello se dividirá en 2 subsecciones:

- *Frontend* (HTML, CSS y TypeScript)
- *Backend* (Java)

3.1.1. *Frontend*

Para realizar la parte de la lógica en el *Frontend* se ha utilizado TypeScript [1]. Este lenguaje es una extensión de JavaScript en el que se añade tipado explícito y estático, además de añadir otras propiedades orientadas a objetos como las clases, los objetos y las interfaces. Esto nos proporciona las ventajas propias de JavaScript de compatibilidad y extensión, en el uso entre los navegadores web, sin perder el enfoque orientado a objetos y el tipado de variables que nos permite tener un desarrollo estructurado y mantenible a lo largo del tiempo. Otro de los

motivos por los que este lenguaje ha sido elegido es debido a que es el propio del *framework* Angular [2], por lo que su integración y facilidad de uso es superior a ningún otro de los lenguajes compatibles con el *framework*.

Figura 1: TypeScript

Respecto al diseño gráfico de la aplicación se ha usado HTML y CSS para la creación de las vistas y las hojas de estilo. Ambos lenguajes nos proporcionan una buena visualización e interacción en los navegadores. Esto se debe a que son ampliamente apoyados por la comunidad y se han constituido como un estándar a la hora de desarrollar la parte gráfica de las páginas web. Además hay que tener en cuenta que tienen completa compatibilidad con Angular lo que facilita y agiliza el desarrollo en gran medida.

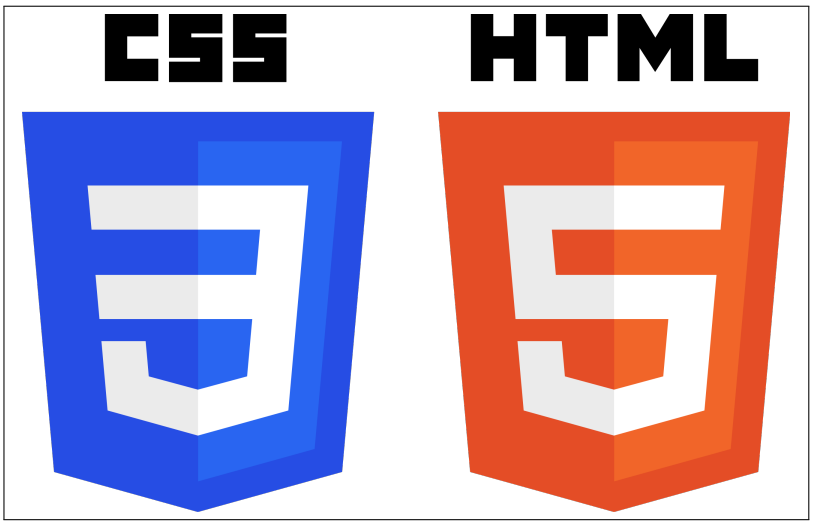

Figura 2: HTML y CSS

3.1.2. *Backend*

Para el desarrollo del *backend* se ha utilizado el lenguaje de programación Java [3]. Este lenguaje es uno de los más empleados y establecidos a lo largo de la historia. Es un lenguaje muy fiable y contrastado lo que asegura que se minimizarán los problemas durante el desarrollo.

La comunidad de Java es inmensa por lo que presenta numerosos *frameworks* y librerías que facilitan y simplifican en gran medida el desarrollo de los proyectos.

Un ejemplo que podemos encontrar de esto en PCEAIM es el uso del *framework* Spring Boot [4], con el que se ha desarrollado toda la parte del *backend* de la que se habla más adelante.

Otro de los puntos a favor por los que ha sido elegido, está también relacionado con la gran comunidad que tiene. Esto se debe a que la cantidad de documentación que se puede encontrar del lenguaje es inmensa y hay gran facilidad para hallar la solución a cualquier problema que se presente durante el desarrollo de una aplicación.

Para la ejecución y compilación de Java se ha decidido utilizar la versión libre de OpenJDK [5] en lugar de su versión privativa. Este uso ha sido motivado ya que la versión libre se ha constituido como la implementación de referencia para Java SE, también se ha tenido en cuenta la naturaleza abierta de la versión libre por lo que es posible que exista una mayor disponibilidad de correcciones y mejoras.

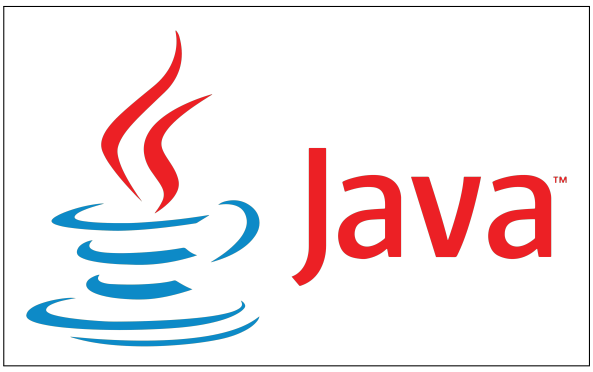

Figura 3: Java

3.2. Tecnologías

En este sección se expondrán las tecnologías utilizadas y los motivos que han llevado a la selección de cada una de ellas. Esta sección está estructurada en 4 subsecciones:

- *Frontend* (Angular)
- *Backend* (Spring Boot, MySQL y OpenCSV)
- Empaquetado en contenedores (Docker y Bash)
- Pruebas (JUnit, Mockito y Selenium)

3.2.1. *Frontend*

Para el desarrollo correspondiente con el *frontend* de la aplicación se ha decidido utilizar Angular en su versión 16.0.4 (la última disponible en el momento en que se empezó el desarrollo del proyecto). Angular es un *framework* desarrollado por Google, ampliamente extendido y utilizado hoy en día para el desarrollo del *frontend* de las aplicaciones web. Está en constante actualización y permite crear de forma muy rápida y sencilla aplicaciones SPA (*Single Page Application*) y conectarlas al servidor.

Una *Single Page Application* [6] es un tipo de aplicación web que se asemeja al concepto de una aplicación de escritorio, donde toda la información se muestra en una única ventana sin ir cargando varias páginas mientras el usuario interactúa con la aplicación web. En este tipo de página todos los ficheros que se van a usar se cargan inicialmente con la primera visita a la página y luego la información mostrada se va actualizando de forma dinámica a medida que el usuario interactúa con la aplicación, sin tener la necesidad de ir cargando los ficheros continuamente.

Este tipo de páginas tiene una desventaja y es que la primera vez que se inicia tiene un mayor tiempo de espera ya que es necesario solicitar todos los ficheros al servidor. Sin embargo presentan una gran ventaja y es que una vez cargados todos los ficheros la experiencia de uso es mucho más fluida, dinámica y elimina las esperas propias de las páginas convencionales en las que había que solicitar continuamente los ficheros al servidor. Uno de los ejemplos más representativos de este tipo de páginas es Gmail, en el que tras un pequeño tiempo de carga la experiencia de uso es totalmente fluida.

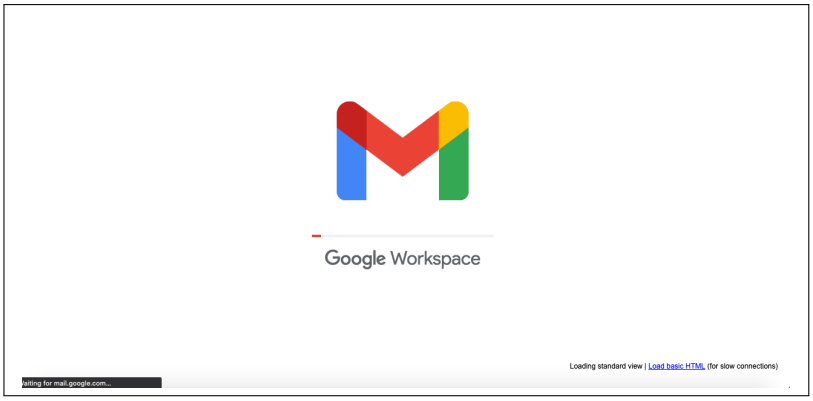

Figura 4: Pantalla de carga Gmail

Angular además permite extender el HTML con etiquetas propias. Esto es posible ya que está basado en componentes reutilizables y no en páginas. Estos componentes se pueden reutilizar a nuestro antojo en cualquier parte de la aplicación lo que favorece la reutilización del código y el uso de buenas prácticas a la hora de desarrollar la aplicación.

Tradicionalmente cuando había que cambiar la información que se mostraba en una página web estábamos obligados a recargar toda la página y volver a solicitar toda la información al servidor. Esto es tremendamente ineficiente ya que si solo se requería cargar una parte específica de la información como el estado de un artículo en el carrito de compra estábamos obligados a solicitar al servidor todos los datos que se iban a mostrar aunque ya supiésemos que su estado no había cambiado. El enfoque basado en componentes de Angular, permite recargar solo aquellas partes de la aplicación que nos interesa en cada momento, lo que evita estar realiazndo peticiones y descargas constantes de datos por parte del usuario.

Angular además tiene bastantes funcionalidades que facilitan el desarrollo frente a la utilización de JavaScript *vanilla* o a JQuery [7]. Una de las más significativas y la que ha motivado en mayor medida la popularización de los *frameworks* de JavaScript es la posibilidad de enlazar variables de la parte lógica con los datos que se muestran en la vista, lo que permite desarrollar de forma fácil formularios en los que el usuario necesita introducir los datos. Otra funcionalidad muy útil y que es propia de Angular es la inyección de dependencias, que aunque es una funcionalidad muy extendida en el desarrollo del *backend* se puede aplicar también al *frontend* con muy buenos resultados. En la inyección de dependencias es el *framework* el encargado de proporcionar todas las dependencias que necesitan los componentes lo que agiliza en gran medida el desarrollo de las aplicaciones.

Por todas las ventajas expuestas anteriormente se ha decidido utilizar Angular para el desarrollo del *frontend* de la aplicación.

Figura 5: Angular

Angular permite ser utilizado tanto con TypeScript como con ECMAScript. La diferencia entre ambos que es que el primero es una versión extendida del segundo, en la que se incluyen todas las características de ECMAScript pero además se implementan algunas nuevas funciones que pueden ayudar a agilizar el desarrollo. Entre estas nuevas características nos encontramos con el tipado de las variables, que permite una mejor localización de los errores al contar con un compilador que puede anticiparse a algunos de los errores que pueden surgir durante el desarrollo. TypeScript además introduce algunos conceptos propios del paradigma orientado a objetos más tradicional lo que puede facilitar el desarrollo sobretodo si estás familiarizado con este paradigma. Por todo esto se ha elegido

utilizar TypeScript para el desarrollo con Angular.

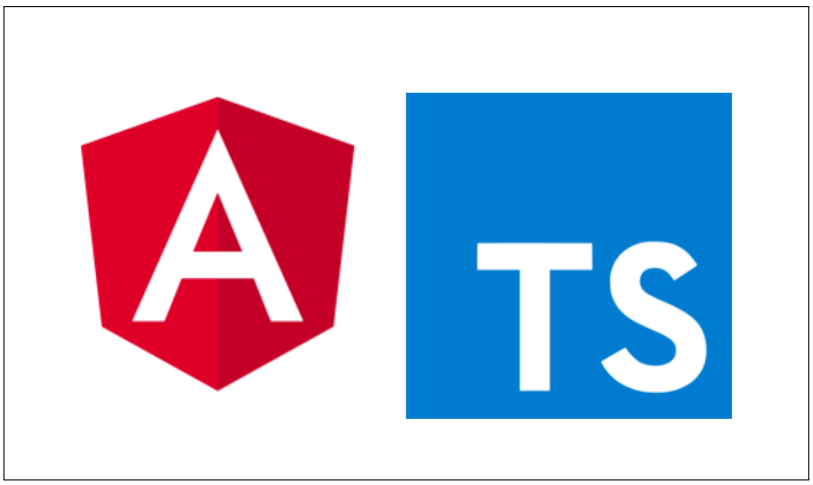

Figura 6: Logos Angular y Typescript

Otra de las funcionalidades de Angular es la posibilidad de complementarlo con librerias que están fuera del alcance de sus funcionalidades básicas. En este desarrollo se ha complementado con la librería de Angular Material [8] en su versión 16.0.4. Esta librería proporciona una serie de componentes gráficos como *Buttons, Cards, Checkboxes y Form fields*. Estos componentes gráficos siguen la normativa del estilo tan característico Material cuyo uso ha sido difundido y fomentado por parte de Google, el cuál está presente por ejemplo en los dispositivos Android y en muchas de sus aplicaciones.

Figura 7: Logo Angular Material

3.2.2. *Backend*

Una de las principales tecnologías que se ha utilizado para el desarrollo del *backend* es Spring Boot en su versión 3.1.0. Spring Boot es un *framework* de Spring [9] que a su vez es un *framework* construido sobre Java. Spring está orientado al desarrollo de aplicaciones empresariales y Spring Boot consigue agilizar el desarrollo de las mismas introduciendo funcionalidades que facilitan en gran medida el desarrollo de una aplicaión. Algunas de las múltiples ventajas que ofrece son su licencia de software libre, sus actualizaciones constantes así como un soporte para todo tipo de bases de datos tanto SQL como NoSQL.

Spring Boot también ofrece la posibilidad de implementar tanto aplicaciones

web como servicios REST. Para este desarrollo se ha decidido utilizar la parte REST que nos proporciona. Esto se ha decido así ya que permite que los datos sean consultados por el *frontend* de la aplicación de forma sencilla mediante peticiones HTTP utilizando algunas de los estándares del protocolo REST para la comunicación entre ambas partes.

Para la integración con las bases de datos se ha utiliado Spring Data [10], que permite la creación de las tablas de la bases de datos a partir de las clases de Java así como la creación de consultas a la base de datos partiendo de los métodos de las clases. También se ha usado JPA (Java Persistence API) [11] para la conexión con las bases de datos relaciones.

Para implementar dichas bases de datos relacionales se ha utilizado MySQL [12] en su versión 8.0.33. Su uso se eligió debido a que es una base de datos de código abierto cuyo uso está muy extendido. Esto ha permitido contar con actualizaciones constantes así como con una gran cantidad de documentación disponible.

Además, también se ha utilizado la librería de OpenCsv [13]. Esta librería se ha usado para conseguir manejar con los archivos .csv relativos a la importación y la exportación de las actividades.

Finalmente, se ha utilizado la librería Java-LTI-1.3 [14] destinada a la integración con plataformas docentes que usan el protocolo LTI para la comunicación con herramientas externas. Esto se ha empleado con el fin de que las plataformas docentes externas puedan registrar usuarios en PCEAIM utilizando un hipervínculo así como para que PCEAIM pueda enviar las calificaciones de los alumnos a las plataformas docentes externas.

Figura 8: Logo Spring Boot y MySql

3.2.3. Empaquetado en contenedores

Uno de los problemas que ha existido históricamente en el desarrollo del software es la diferencia entre los dispositivos de los desarrolladores y los servidores o equipos de producción de las aplicaciones. Esto es debido a que habitualmente los entornos de producción y los de desarrollo son muy distintos y hay que dedicar muchos esfuerzos para que una aplicación se ejecute en ambos entornos. Para solucionar estos problemas se ha decidido empaquetar la aplicación en un

contenedor de Docker [15].

Docker es una plataforma diseñada para facilitar la creación, el despliegue y la ejecución de aplicaciones en contenedores. Los contenedores son entornos ligeros y portátiles que incluyen todo lo necesario para que una aplicación se ejecute de manera independiente. Por lo que facilita su implementación en diferentes entornos de manera consistente y reproducible.

Se ha elegido docker como tecnología de empaquetado en contenedores debido a que es de código abierto y está ampliamente extendida. Esto hace que haya gran cantidad de documentación disponible además de estar actualizándose y mejorándose continuamente. Otra de las ventajas que presenta la utilización de Docker es la existencia de Docker Hub. Docker Hub es el repositorio público de images de Docker mantenido directamente por Docker.

La utilización de Docker se ha complementado con Docker Compose que permite una orquestación entre las diferentes aplicaciones cuando tenemos un proyecto formado por múltiples componentes. También se ha utilizado Bash para el desarrollo de scripts que permitan crear imágenes y subirlas a Docker Hub con un simple comando de consola.

Figura 9: Logo Docker

3.2.4. Pruebas

Para el desarrollo de las pruebas, se ha optado por utilizar el *framework* JUnit [16], el cual facilita la ejecución de test automáticos. Una de las principales funcionalidades del *framework* es la definición de los métodos de prueba mediante el uso de la anotación @Test. Asimismo, facilita y permite la ejecución de funciones previas a cada test mediante el uso de la anotación @BeforeEach. La forma de funcionamiento de JUnit es mediante aserciones, utilizando estas aserciones se puede comprobar si el valor de un cierto campo de un objeto es igual que el valor que se espera que tenga este campo. Además de lo anteriormente mencionado, otra de las grandes ventajas de JUnit es la integración eficaz con los entornos de desarrollo integrado (IDEs) más conocidos, lo que permite que la ejecución de los test se lleve a cabo de forma muy sencilla

Junto a JUnit se ha empleado la librería Mockito [17], la cual facilita la realización de pruebas unitarias sobre las distintas partes del código. Con esta herra-

mienta se pueden *mockear* las respuestas de distintas partes de la aplicación lo que permite centrarse en la función que se está *testeando* sin tener en cuenta los efectos que pueden tener los test en otras funciones de las que se depende. Esto permite aligerar mucho la carga de realización de los test al poder centrarnos en una única funcionalidad.

Para la realización de las pruebas *end-to-end* junto con el *framework* de JUnit se ha empleado el *framework* Selenium [18]. Este nos permite realizar pruebas end-to-end (e2e), es decir de toda nuestra aplicación, sobre los navegadores web. Con este *framework* podemos visitar páginas web, pulsar botones, rellenar formularios y comprobar que las acciones que realizamos sobre la aplicación tengan el efecto esperado.

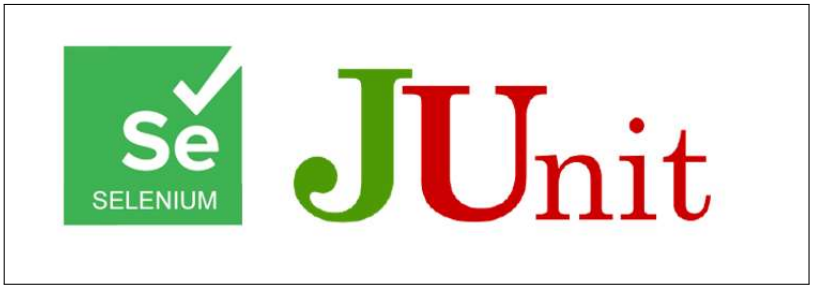

Figura 10: Logos JUnit y Selenium

3.3. Herramientas

En esta sección relativa a las herramientas se hablará sobre aquellas que se han utilizado en el desarrollo del proyecto.

- Gestión de dependencias (Maven y Node Package Manager)
- Gestión de base de datos (MySQL Workbench)
- Control de versiones (Git y GitHub)
- Organización (Trello, Microsoft Outlook y Microsoft Teams)
- Despliegue de la aplicación (Google Cloud Run)
- Integración continua y despliegue continuo (GitHub Actions)

3.3.1. Gestión de dependencias

Para gestionar las dependencias necesarias para el desarrollo de la aplicación se han utilizado 2 tecnologías.

Con el fin de manejar las dependencias relativas al *backend* se ha utilizado Maven [19]. Esta es una herramienta muy consolidada en el mercado, además de ser software libre y estar ampliamente extendida. Cuenta con un inmenso

repositorio público de librerías lo que permite añadir una nueva dependencia a un proyecto de forma rápida. También tiene una alta integración con los IDEs más populares por lo que su uso es muy cómodo a la hora de desarrollar una aplicación. Además de todo esto permite controlar todo el ciclo de vida de un proyecto por lo que también se ha usado para realizar la ejecución de los test y la construcción del proyecto.

Figura 11: Logos de Maven

En cuanto a la administración de dependencias del *frontend* se ha usado Node Package Manager (NPM) [20]. Esta herramienta, de igual forma que Maven, está muy consolidada en el mercado, además de estar ampliamente extendida y contar con una comunidad de usuarios muy grande. También presenta un repositorio público de dependencias gigantesco y una muy buena integración con la consola, lo que permite añadir dependencias a un proyecto de forma muy rápida y fácil. Además de todo esto, ha sido utilizada para construir el *frontend* de la aplicación.

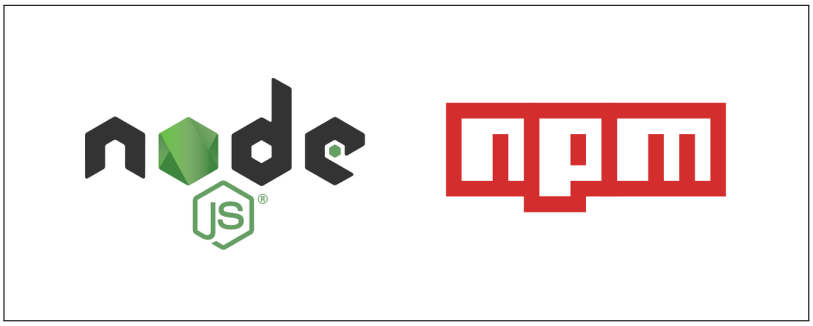

Figura 12: Logos de Node.JS y NPM

3.3.2. Gestión de base de datos

Para gestionar la base de datos se ha decidido usar MySQL Workbench [21], ya que es una herramienta que dedicada a la gestión de la base de datos a través de una interfaz de usuario. Esta herramienta permite la conexión de forma sencilla con las bases de datos locales así como ver la estructura que siguen estas y lanzar *queries* sobre ellas de forma rápida y sencilla. Además de todo esto permite la creación de gráficos a partir de la estructura de tablas y relaciones de la base de datos para poder visualizarla y analizarla de forma más intuitiva y visual.

Figura 13: Logo de MySQL Workbench

3.3.3. Control de versiones

Como herramienta para realizar el control de versiones en este proyecto se ha utilizado Git [22] acompañado de GitHub [23], el cual ha tenido la función de actuar como repositorio remoto en el que se iba guardando todo el trabajo relativo al desarrollo de la aplicación. Además para administrar este repositorio remoto se ha utilizado una versión de GitFlow como modelo de flujo de trabajo (*workflow*). En la versión utilizada existían los siguientes tipos de ramas:

- Rama *master*: es la rama principal del repositorio y aquella en la que va estar el código que se encuentra en producción. Solo se puede actualizar fusionando la rama *develop* o para solucionar algún error encontrado en producción.
- Rama *develop*: es la rama que surge de la ramificación de *master*. Esta solo se vuelve a fusionar con su rama padre cuando tiene todas las características solicitadas. Solo se puede actualizar fusionando las ramas de *features*.
- Ramas de *features*: son las ramas en las que se realiza la mayor parte del desarrollo. Son ramificaciones de *develop* y son creadas para desarrollar una *feature* requerida. Cuando el desarrollo y el *testeo* terminan se fusiona con su rama padre.

Figura 14: Logos de Git y Github

3.3.4. Organización

Para el desarrollo del proyecto se ha seguido una metodología de trabajo ágil apoyada por las herramientas utilizadas para la gestión de las tareas pendientes. Una de estas herramientas ha sido Trello [24], la cual está pensada para llevar a cabo la gestión de un trablero Kanban. En esta herramienta se creaba un tarjeta correspondiente a cada tarea que se tenía que realizar y permitía conocer en cada instante cual era el estado del desarrollo, además de ofrecer la posibilidad de ver las tareas que quedaban por realizar en cada momento.

En lo relativo a planificar las siguientes iteraciones del desarrollo o para mostrar el trabajo realizado por parte del alumno al profesor se ha usado la herramienta Microsoft Teams. Esta herramienta permite realizar llamadas de video y compartir la pantalla del equipo, lo que permite realizar fácilmente una *demo* de una aplicación aunque las personas estén en lugares físico separados.

Para la comunicación rápida y la resolución de dudas se ha usado Microsoft Outlook, el servicio de correo electrónico proporcionado por Microsoft. Esta herramienta permite intercambiar mensajes de forma sencilla lo que ha facilitado en gran medida la comunicación entre el alumno y el tutor.

Figura 15: Logos de Trello, Outlook y Teams

3.3.5. Despliegue de la aplicación

En lo relativo al despligue en producción de la aplicación se utilizó la herramienta Google Cloud Run [25]. Google Cloud Run es un servicio de PAAS (Platform as a service) ofrecido por Google Cloud que proporciona todo lo que necesitamos para el despliegue de un contenedor de Docker. Una de las mayores ventajas que ofrece es su plan gratuito, el cual presenta suficiente capacidad para desplegar aplicaciones sencillas durante un periodo de tiempo considerable, además aunque exista algún problema no se te cobrará por sus servicios, simplemente se cortará el servicio. Además de esto también tiene una buena integración con entornos de CI/CD de lo que hablaremos en el siguiente apartado. Por todo esto y por su facilidad de uso es que se ha elegido Google Cloud Run como herramienta de despliegue.

Figura 16: Logo de Google Cloud Run

3.3.6. Integración continua y despliegue continuo

Hoy en día supone una gran ventaja realizar integración continua y despliegue continuo (CI/CD) en los desarrollos software. Por un lado, permite realizar *tests* de la aplicación de forma automática y sin intervención de un humano, por otro lado permite que el despligue de los cambios *testeados* en una aplicación se suban directamente al servidor de producción sin necesidad de que un humano intervenga. Esto permite reducir en gran medida el esfuerzo de los desarrolladores a la hora de realizar estas tareas repetitivas.

Para realizar esta tarea en el desarrollo de PCEAIM de utilizó la herramienta de GitHub Actions [26]. Esta herramienta proporciona todas las funcionalidades que se requieren para realizar CI/CD sobre las aplicaciones que están siendo desarrolladas. Principalmente se ha elegido esta herramienta ya que se ha utilizado GitHub como repositorio remoto para el código de la aplicación por lo que la integración con GitHub Actions es automática. También se ha tenido en cuenta

la gran comunidad que respalda a la herramienta y la gran facilidad de encontrar documentación de calidad sobre la misma.

Para utilizar esta herramienta solamente hay que añadir un archivo .yml en la carpeta /.github/workflows del proyecto asociado al repositorio poder sacar provecho de ella. Lo primero que se hizo fue definir en que casos se ejecutaría el flujo de trabajo definido. Se tomó la decisión de ejecutar el flujo de trabajo cada vez que se fusionaba una rama con *master* (la rama principal del repositorio) o cada vez que se pusheaba directamente sobre esta rama (al realizar hotfixes).

El flujo definido para realizar CI/CD consiste en los siguientes pasos:

1. Inicialización y configuración: en este paso se inicializan y se configuran los servicios que se van a utilizar durante el flujo de trabajo.

2. Login en Docker y construcción de la imagen: en este paso se realiza el login en la cuenta de Docker y se construye la imagen de la aplicación.

3. Ejecución de la imagen de Docker: se ejecuta la imagen de Docker que se acaba de crear en el servidor de GitHub Actions y se espera a que se esté ejecutando para lanzar sobre ella los test *end-to-end*.

4. *Testeo* de la aplicación: se ejecutan los test unitarios y los test *end-to-end* sobre la aplicación.

5. Publicación de la imagen en Docker Hub: se para la ejecución de la aplicación y se sube la imagen creada a Docker Hub.

6. Despliegue de la aplicación en Google Cloud Run: se instalan las utilidades de Google Cloud, se realiza el login en Google Cloud y se despliega la aplicación sobre el servicio ofrecido por Google Cloud Run.

Figura 17: Logo Github Actions

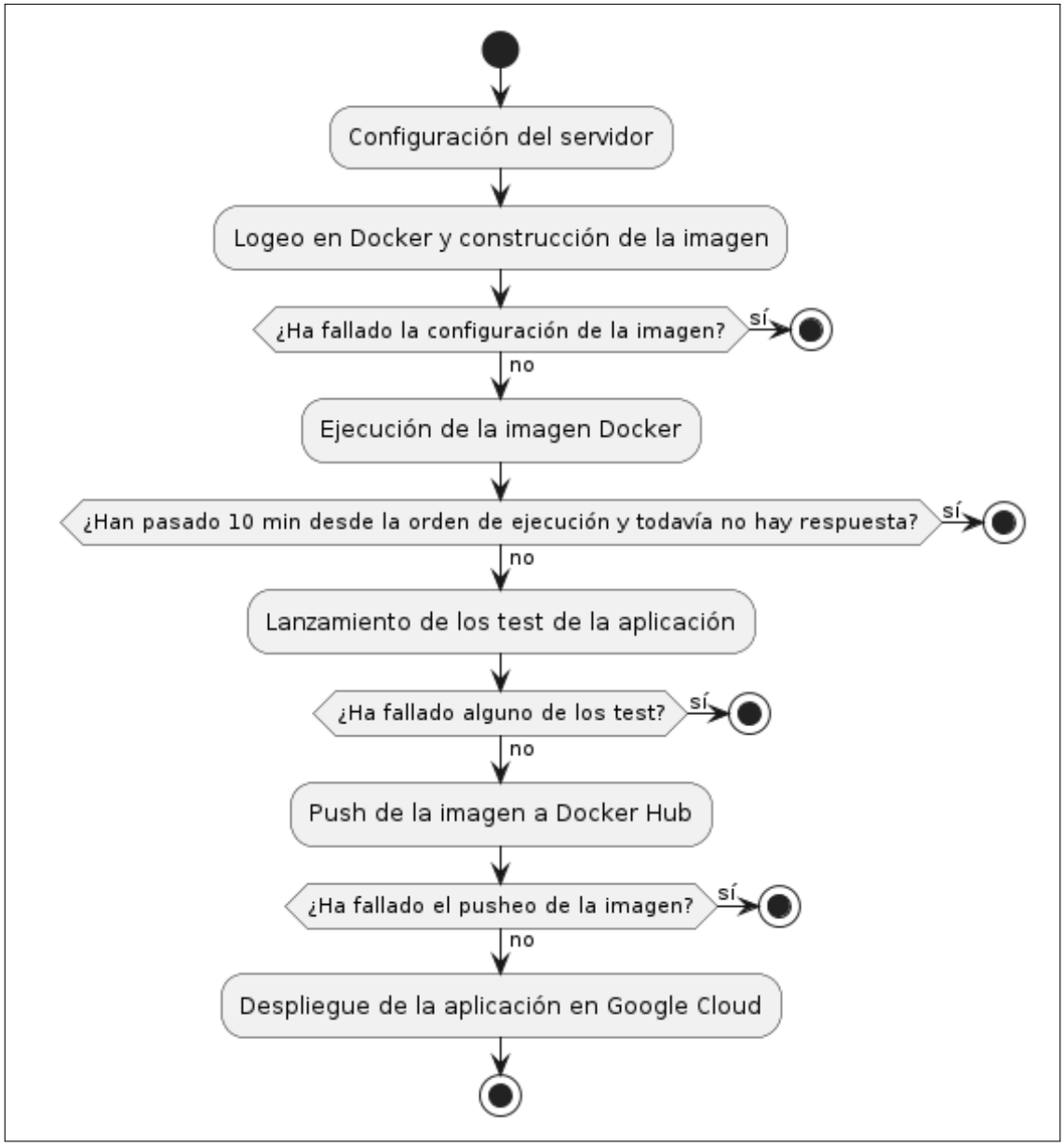

Figura 18: Diagrama de flujo CI/CD

3.4. Metodologías

Para desarrollar esta aplicación se han utilizado técnicas modernas de implementación y desarrollo de software entre las que se encuentra el marco de trabajo Ágil Kanban utilizado junto con un enfoque iterativo e incremental.

El ciclo de vida iterativo incremental es un enfoque de desarrollo de proyectos que se utiliza particularmente en el contexto del desarrollo de software. Este

enfoque implica la división del proyecto en una serie de iteraciones o ciclos, donde cada iteración termina con un producto potencialmente entregable (MVP) el cual puede ser utilizado sin necesidad de terminar todas las iteraciones del desarrollo. Esta forma de llevar a cabo el desarrollo de un proyecto tiene una principal ventaja y no es otra que la aplicación desarrollada, que aunque esté en un estado muy primitivo, puede ser utilizada y explotada en un corto periodo de tiempo sin la necesidad de esperar a que el desarrollo de la misma se dé por concluido.

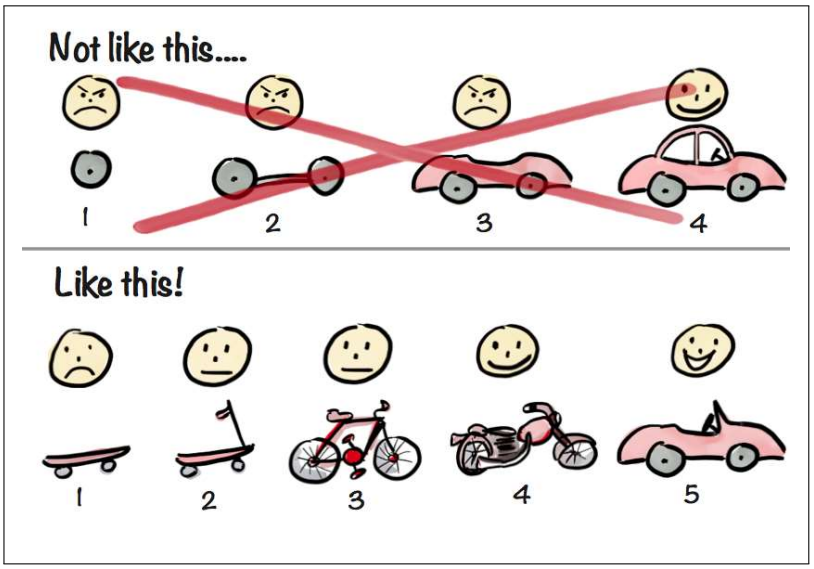

Figura 19: Ejemplo de desarrollo tradicional vs ciclo iterativo e incremental

El desarrollo de PCEAIM ha estado compuesto por 3 iteraciones o incrementos:

1. Aplicación Autónoma: Este fue el primer incremento del desarrollo. En este primer incremento se añadieron todas las funcionalidades presentes actuálmente en la aplicación pero esta no podía comunicarse con otras plataformas docentes de ninguna forma, tampoco podía exportar las actividades a ficheros .csv ni importar actividades de ficheros .csv. En resumen la aplicación funcionaba de forma autónoma sin posibilidad de intercambio de información con otras plataformas.

2. Comunicación con ficheros: Esta fue el segundo incremento del desarrollo. En este incremento se añadió la posibilidad de comunicarse con otras plataformas docentes a través de la importación de actividades de otras plataformas a partir de un fichero .csv y la exportación de las actividades de la plataforma en un fichero .csv.

3. Plaforma LTI: Este fue el último incremento del desarrollo de la aplicación. En este incremento presenta todas las funcionalidades de los incrementos anteriores y además se añadió la posibilidad de configurar la aplicación

como una herramienta externa de otras plataformas docentes. De esta forma las plataformas docentes pueden intercambiar información con PCEAIM a través de un enlace que cumple el protocolo LTI. Esto también ha permitido la posibilidad de enviar las calificaciones de PCEAIM automáticamente a la plataforma docente.

Kanban al ser una metodología ágil se basa en los cuatro valores del Manifiesto Ágil:

- Individuos e interacciones sobre procesos y herramientas.
- Software funcional sobre documentación exhaustiva.
- Colaboración con el cliente sobre negociación contractual.
- Respuesta ante el cambio sobre seguir un plan.

El elemento central de Kanban es el tablero Kanban. Este es una herramienta visual utilizada para representar el flujo de trabajo, se compone de tarjetas que representan las tareas o elementos de trabajo y de columnas que representan el estado de las tareas. Estas columnas, en la versión básica del trablero pueden ser de 3 tipos:

1. *To do*: en esta columna se encuentran las tareas que están pendientes en las cuales aún no se ha empezado a trabajar.

2. *Doing*: en esta columna se encuentran aquellas tareas en las que se está trabajando. El número de tareas que pueden estar en esta columna suele estar limitado.

3. *Done*: en esta columna están aquellas tareas que han sido terminadas, esto incluye que las tareas hayan sido *testeadas*.

Estás columnas más habituales pueden ser modificadas en cualquier momento, de forma que se pueden añadir o eliminar columnas teniendo en cuenta las necesidades del proyecto.

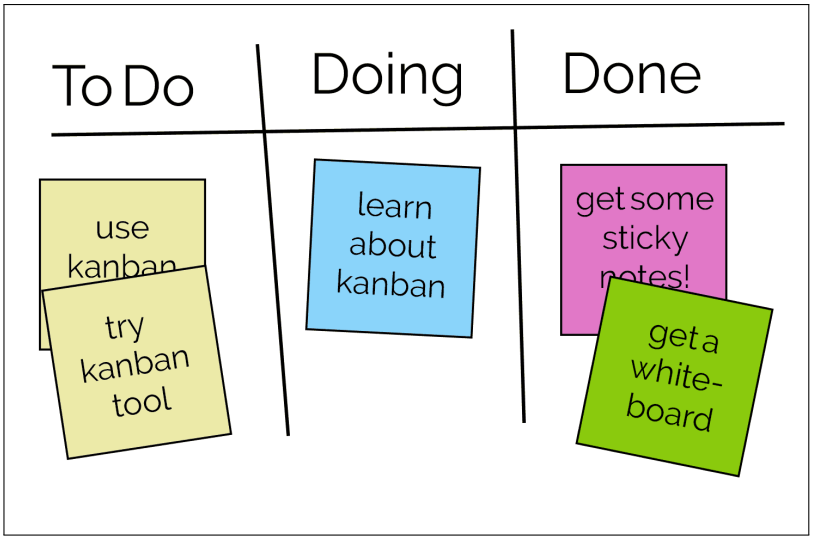

Figura 20: Ejemplo de tablero Kanban

Teniendo en cuenta todo lo mencionado anteriormente cabe destacar que Kanban extiende a los principios del manifiesto Ágil utilizando el tablero Kanban como eje central del desarrollo. Entre las buenas prácticas de trabajo que define Kanban se encuentran:

- Visualización del flujo de trabajo: al utilizar el trablero Kanban para representar visualmente el flujo de trabajo se puede saber en todo momento en que estado se encuentra un proyecto con un vistazo rápido al tablero.
- Limitación del trabajo en curso: otro de los principios claves que se establecen en Kanban es la limitación de las tareas que se encuentran en el estado *doing*. Esta limitación promueve que no haya muchas tareas abiertas al mismo tiempo, lo que promueve que continuamente se vayan terminando tareas y se vaya entregando *working software* de forma continua.
- Priorización basada en el valor: Kanban permite priorizar el trabajo en función del valor de las distintas tareas. Las tarjetas de trabajo relacionadas con tareas que aportan mayor valor suelen recibir una mayor prioridad.

Como se ha comentado anteriormente en esta memoria, para gestionar el tablero Kanban se ha utilizado la herramienta Trello. Además cabe mencionar que a las tres columnas básicas del tablero se le ha añadido una cuarta columna relativa al *testing*. En esta columna se encontraban aquellas tareas que ya estaban desarrolladas pero las cuales aún no habían pasado los *test*.

4

Descripción informática

En esta sección se examinan detalladamente los componentes de alto nivel de la aplicación. Se iniciará discutiendo las historias de usuario que describen las funcionalidades de la aplicación, seguido por un análisis de la arquitectura de la aplicación, así como ciertos detalles de implementación interesantes. La sección concluirá describiendo las pruebas que se han realizado.

4.1. Requisitos

En este apartado se realiza un análisis desde una visión de alto nivel de los requisitos implicados en la aplicación.

En primer lugar, se dará la posibilidad de crear actividades. Estas actividades pueden ser de dos tipos: preguntas cortas y entrega de ficheros.

Para ambos tipos de actividades existen los siguientes apartados modificables:

- Nombre de la actividad.
- Porcentaje de peso de la actividad sobre la calificación final.
- Visibilidad de la actividad.
- Visibilidad de la calificación de la actividad.
- Posibilidad de realizar entregas tardías.
- Fechas de apertura y cierre de la actividad.
- Fichero que sirva a modo de enunciado global.

Además de estos apartados comunes para ambos tipos de actividades existen apartados específicos para cada tipo de actividad. Para las actividades del tipo preguntas cortas se puede establecer el tiempo de realización de la actividad además de añadir preguntas ilimitadas. Estas preguntas deben estar compuestas por un enunciado y una puntuación. Para las actividades de tipo subida de fichero se puede establecer si los alumnos podrán hacer intentos de entrega ilimitados.

Los profesores podrán calificar las actividades que han creado. Para existe la posibilidad de ver fácilmente si el alumno ha entregado o no ha entregado la actividad y en caso de que la haya entregado ver esa entrega.

Como es lógico, los alumnos pueden realizar las actividades creadas por el profesor, con todas las restricciones que los profesores imponen sobre estas actividades. También pueden ver la calificación que han recibido en esa actividad.

Deberá existir una pestaña en la que los alumnos puedan ver todas las calificaciones que han obtenido en una asignatura siempre y cuando estas sean visibles. Los profesores tendrán acceso a esta pestaña y pueden ver todas las calificaciones de todos los alumnos de la asignatura que imparten.

En cuanto al administrador de la aplicación, este tendrá la posibilidad de crear asignaturas, también podrá añadir y eliminar profesores de las asignaturas así como editar la información de las asignaturas y borrarlas.

En la aplicación existirán 2 roles para los usuarios:

- Administrador: encargado de administrar las asignaturas.
- Usuarios: estos usuarios pueden ser tanto alumnos como profesores de las asignaturas.

Para conseguir todo lo anteriormente mencionado se ha implementado una aplicación capaz de llevar a cabo todos los puntos. Esta aplicación está disponible en cualquier tipo de dispositivo ya que se puede acceder a ella a través de los navegadores web modernos (Google Chrome, Mozilla Firefox, Safari, Opera y Microsoft Edge).

Al haber utilizado una metodología ágil para realizar este desarrollo los requisitos se han representado como las historias de usuario que tuvo cada iteración.

Rol Usuarios	**Criterios de aceptación**
Yo como organización **quiero** poder tener diferentes roles de usuario **para** poder usar la aplicación.	El usuario administrador puede crear asignaturas, editarlas y borrarlas. El usuario general puede ser profesor o alumno de las asignaturas.

Creación Cuenta	**Criterios de aceptación**
Yo como usuario general **quiero** poder crear una cuenta **para** poder usar la aplicación.	Deben introducirse obligatoriamente los campos nombre, apellidos, *email* y contraseña. El correo introducido no debe estar asociado a ninguna otra cuenta. Debe solicitarse la repetición de la contraseña para comprobar que el usuario introduce una contraseña que conoce.

Logear Usuario	**Criterios de aceptación**
Yo como usuario general **quiero** poder iniciar sesión en la aplicación con mis credenciales **para** poder utilizar la aplicación.	El usuario solo debe poder logearse cuando proporciona las credenciales correctas. Si el usuario se equivoca al introducir las credenciales de inicio de sesión se debe proporcionar retroalimentación al usuario indicando que se ha equivocado.

Creación de asignaturas	**Criterios de aceptación**
Yo como usuario administrador **quiero** poder crear asignaturas **para** que el personal docente y los alumnos puedan usar la aplicación.	Se debe poder introducir el nombre de la aplicación así como seleccionar los docentes y los alumnos de la asignatura. El nombre introducido para la asignatura debe ser único y no ser igual al de ninguna otra asignatura existente. Un usuario no puede ser seleccionado como alumno y como docente de una misma asignatura al mismo tiempo.

Edición de asignaturas	**Criterios de aceptación**
Yo como usuario administrador **quiero** poder editar asignaturas **para** poder actualizar la información de las asignaturas.	Poder ver la información de la asignatura y modificar los datos con los mismos criterios que para la creación.

Borrar asignaturas	**Criterios de aceptación**
Yo como usuario administrador **quiero** poder borrar asignaturas **para** poder eliminar asignaturas que ya no se imparten.	Si la asignatura no se ha podido borrar por cualquier motivo se debe mostrar activamente al usuario.

Creación de actividades	**Criterios de aceptación**
Yo como usuario general que imparte una asignatura **quiero** poder crear actividades **para** poder realizar exámenes a los alumnos de mis asignaturas.	Se deben poder elegir entre 2 tipos de actividades: subida de fichero y preguntas cortas. En ambos tipos de actividades se deben establecer obligatoriamente los siguientes campos: ▪ Nombre: este campo debe ser distinto al de cualquier otra actividad de la asignatura. ▪ Porcentaje sobre la calificación final: este campo debe ser un número superior a cero y la suma del peso de todas las actividades de la asignatura debe ser menor a 100. ▪ Visibilidad: en caso de que sea *true* los alumnos pueden ver la actividad en caso contrario no pueden verla. ▪ Visbilidad de la calificación: en caso de que sea *true* los alumnos pueden ver la nota de la actividad en caso contrario no pueden verla. ▪ Posibilidad de realizar entregas tardías: en caso de que sea *true* los alumnos pueden realizar la entrega una vez ha terminado el plazo máximo de entrega. ▪ Fechas de apertura y cierre de la actividad: la fecha de apertura debe ser posterior a la fecha actual y la de cierre posterior a la de apertura. De forma opcional se debe poder subir un fichero a modo de enunciado.

Creación de actividades de tipo subida de fichero	**Criterios de aceptación**
Yo como usuario general que imparte una asignatura **quiero** poder crear actividades de tipo subida de fichero **para** poder solicitar ficheros a mis alumnos.	Los mismos criterios que en creación de actividades pero adicionalmente se debe poder seleccionar si los alumnos tendrán la posibilidad de realizar intentos ilimitados.

Creación de actividades de tipo preguntas cortas	**Criterios de aceptación**
Yo como usuario general que imparte una asignatura **quiero** poder crear actividades de tipo preguntas cortas **para** poder realizar preguntas a mis alumnos a través de la aplicación.	Los mismos criterios que en creación de actividades. Obligatoriamente se debe establecer el tiempo de realización de la actividad y este debe ser superior a 1 minuto. Se deben poder añadir un número ilimitado de preguntas cortas. El número de preguntas debe ser superior a uno y para cada pregunta se debe establecer como mínimo un enunciado una puntuación.

Edición de actividades	**Criterios de aceptación**
Yo como usuario general que imparte una asignatura **quiero** poder editar actividades **para** modificar la información de las actividades que he creado..	Se deben cumplir los mismos criterios que para la creación de las actividades.

Importación de actividades	**Criterios de aceptación**
Yo como usuario general que imparte una asignatura **quiero** poder importar un examen a partir de un archivo .csv **para** poder migrar actividades de otras plataformas docentes.	Se deben poder importar ficheros .csv exportados desde plataformas basadas en Moodle. Se deben poder importar únicamente exámenes de preguntas cortas.

Exportación de exámenes	**Criterios de aceptación**
Yo como usuario general que imparte una asignatura **quiero** poder exportar una actividad dell tipo preguntas cortas a un archivo .csv **para** poder migrar actividades a otras plataformas docentes.	Se deben poder exportar ficheros .csv con un formato compatible con las plataformas basadas en Moodle.

Visionado de entregas	**Criterios de aceptación**
Yo como usuario general que imparte una asignatura **quiero** poder ver todas las entregas de una actividad **para** poder conocer el trabajo de los alumnos de la asignatura que imparto.	Se debe indicar el alumno que ha realizado la entrega. Se debe indicar el estado de la entrega, es decir si ha sido entregada o no la actividad. Si la actividad ha sido entregada por el alumno se debe indicar si la entrega ha sido realizada en tiempo o de forma tardía. Debe existir un botón al lado de la actividad para poder descargarla fácilmente, además también debe haber un botón para calificar la actividad.

Calificación de actividades	**Criterios de aceptación**
Yo como usuario general que imparte una asignatura **quiero** poder calificar las entregas de una actividad **para** poder establecer una nota para las actividades entregadas por los alumnos de la asignatura que imparto.	Se debe poder añadir un comentario con retroalimentación además de la calificación. Si el examen es de preguntas cortas se debe poder calificar cada pregunta por separado. La nota establecida para cada pregunta debe ser superior a cero y menor o igual a la nota máxima de la pregunta. La nota de la actividad se calcula automáticamente sobre diez con la nota de las preguntas individuales. Si el examen es de tipo subida de fichero la calificación debe ser mayor a cero y menor a diez.

Edición de calificaciones docente	**Criterios de aceptación**
Yo como usuario general que imparte una asignatura **quiero** poder editar las calificaciones de una entrega realizada por un alumno de la asignatura que imparto **para** poder actualizar las calificaciones que había establecido anteriormente.	Los criterios de aceptación de la edición de calificaciónes deben ser los mismos que para establecer las calificaciones.

Visualización de calificaciones	**Criterios de aceptación**
Yo como usuario general que imparte una asignatura **quiero** poder ver las calificaciones de todas las actividades de los alumnos de mi asignatura **para** poder conocer fácilmente el desempeño de los alumnos.	Se debe mostrar además de las calificaciones un cálculo automático de la nota final del alumno. Además si la nota final es inferior a 5 se debe mostrar que ha suspendido, en caso contrario se debe mostrar que está aprobado.

Realización de actividades	**Criterios de aceptación**
Yo como como usuario general matriculado en una asignatura **quiero** poder realizar las actividades creadas por el docente de la asignatura **para** poder ser evaluado por mi trabajo.	Se debe poder ver el estado de la entrega de forma similar a la que lo hace el docente. Se debe poder ver la entrega que se ha realizado. Si la actividad es de preguntas cortas se debe entregar automáticamente cuando se supera el tiempo máximo de realización.

Visualización de calificaciones alumno	**Criterios de aceptación**
Yo como usuario general que está matriculado en una asignatura **quiero** poder ver las calificaciones de todas las actividades que he entregado **para** poder conocer fácilmente mi desempeño en la asignatura.	Se deben mostrar solo aqueyas calificaciones que no estén ocultas. Se debe mostrar además de las calificaciones un cálculo automático de la nota final del alumno. Además si la nota final es inferior a 5 se debe mostrar que ha suspendido, en caso contrario se debe mostrar que está aprobado.

Conexión LTI	**Criterios de aceptación**
Yo como Como usuario de la aplicación que usa una plataforma docente externa **quiero** poder disponer de conexión lti en la aplicación **para** poder comunicar fácilmente mi plataforma docente externa con la aplicación.	Se debe poder configurar el protocolo LTI a partir de la información proporcionada por la plataforma docente. Se deben poder recibir los datos del usuario que hace la solicitud LTI así como la asignatura y la actividad a la que se quiere acceder. Se deben poder enviar las calificaciones establecidas en la aplicación a la plataforma docente de forma automática.

4.2. Arquitectura y Análisis

Este apartado está destinado al análisis de la arquitectura de la aplicación así como a la comprensión de las diferentes partes que la componen.

La aplicación está formada por un *backend*(API), una base de datos (MySQL), un *frontend*(SPA) y opcionamente una aplicación docente externa LTI a la que se estará conectado.

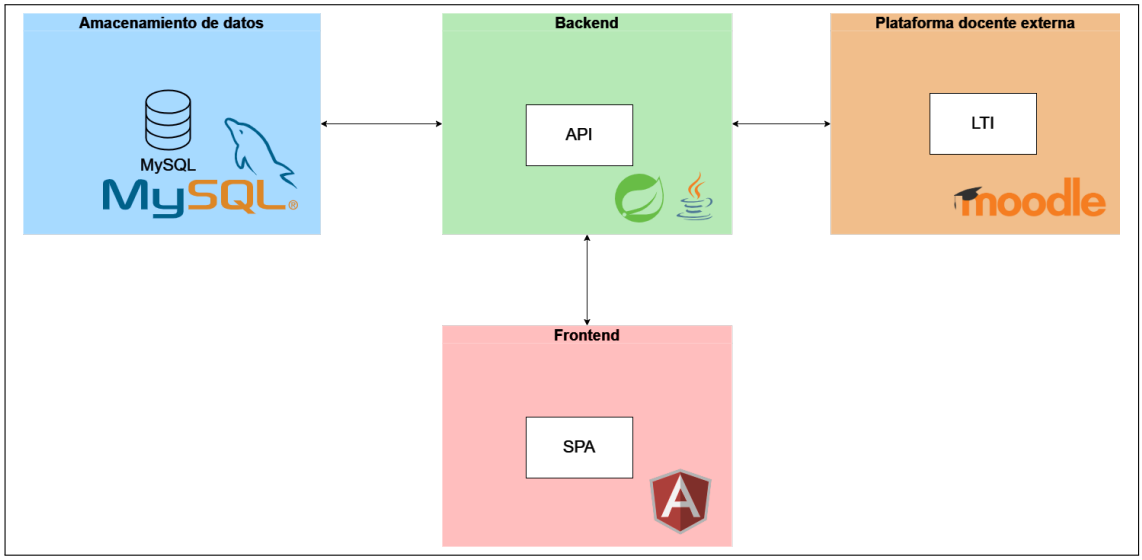

Figura 21: Arquitectura aplicación

Ahora se explicará la interacción entre las distintas partes de la aplicación:

- *Backend*: Es la parte relativa al servidor de la aplicación. Esta parte está compuesta por la la API REST implementada con Spring Boot y Java de la que ya se ha hablado con anterioridad. Se ha decidido implementar con una interfaz a través de API REST ya que así puede ser reutilizada en el futuro si por ejemplo se quiere desarrollar una aplicación móvil. Esta parte se interactúa con las siguientes partes:
  - Plataforma docente externa: esta plataforma le proporciona los datos sobre el usuario que está realizando la solicitud, además de proporcionar un *endpoint* para enviar las calificaciones de los alumnos.
  - Almacenamiento de datos: los datos que se han recogido en el *frontend* son enviados a este componente para que sean persistidos y almacenados.
  - *Frontend*: es la parte con la que interactúa el usuario directamente. Aquí se recogen los datos que introduce el usuario así como las instrucciones del este para devolver la información que desee de la base de datos.
- *Frontend*: es la parte encargada de dirigir la interacción del usuario. Esta parte está compuesta por una aplicación SPA desarrollada en Angular como se ha comentado anteriormente. Esta parte envía la información introducida por el usuario al *Backend* a través de la API REST para que este la persista en la base de datos, también solicita la información que requiere el usuario que está interactuándo con ella.

La aplicación cuenta con distintos tipos de usuarios que pueden relizar varias acciones.

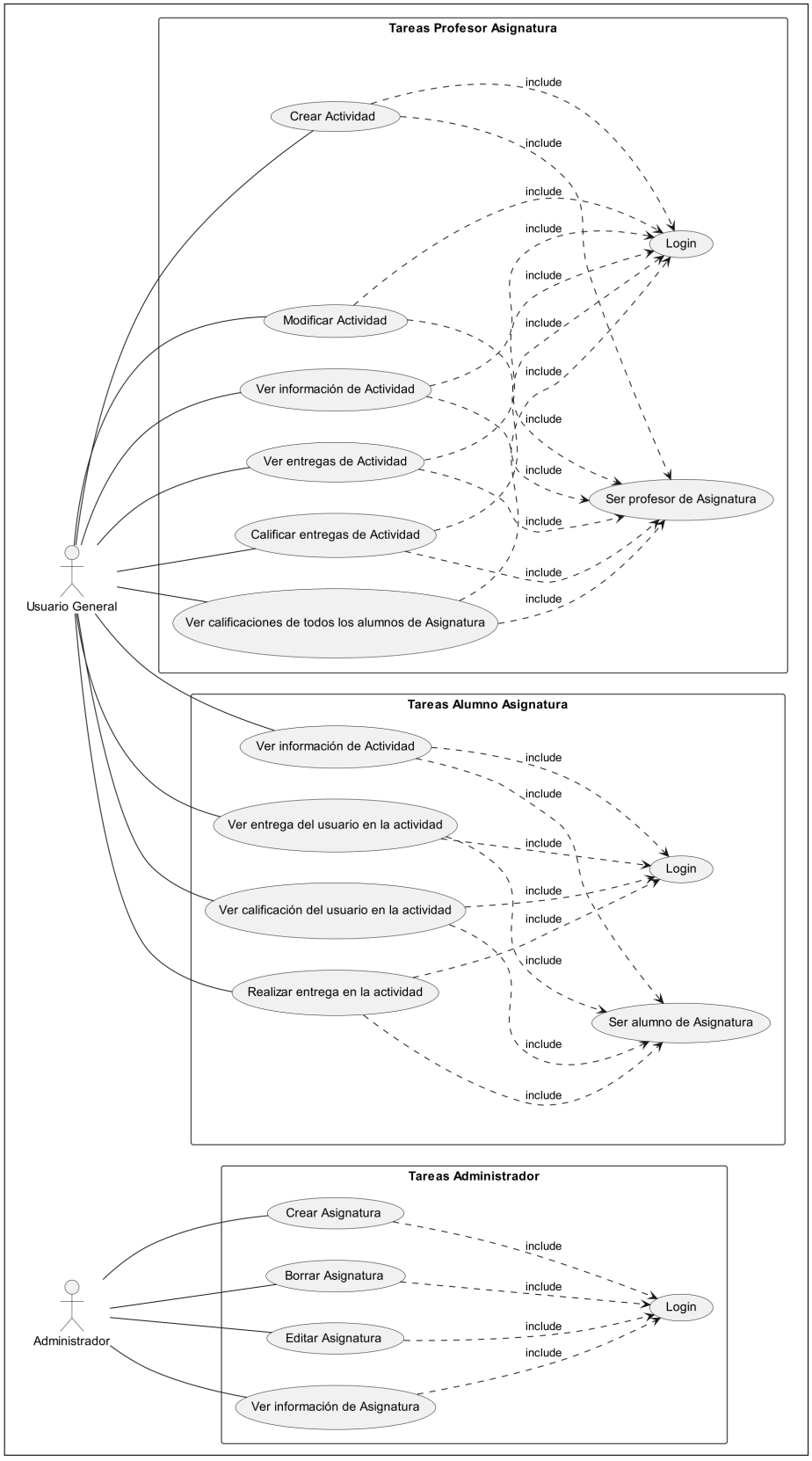

Figura 22: Diagrama casos de uso

4.2.1. *Backend*

Esta parte es una de las más importantes de la aplicación como se ha comentado anteriormente, por lo tanto también es la parte que mayor tiempo de desarrollo ha requerido. En esta sección se explicará tanto la arquitectura que se ha seguido para implementar el *backend* como las características generales de las clases sin ahondar en cada una de ellas.

Respecto a la arquitectura elegida se ha elegido utilizar un diseño basado en el patrón Modelo-Vista-Controlador (MVC) [27]. Este patrón de diseño se utiliza de forma muy habitual en el desarrollo de aplicaciones, en especial en aplicaciones web. Su propósito es separar la lógica de la aplicación en tres componentes distintos para mejorar la modularidad y la mantenibilidad del código. Estos tres componentes son:

- Modelo: Representa los datos de la aplicación. Es decir gestiona los datos y se encarga de la interacción con la base de datos.
- Vista: Se encarga de mostrar los datos al usuario y de proporcionar una interfaz para interactuar con la aplicación. La Vista no realiza ningún cálculo ni manipulación de datos.
- Controlador: actúa como intermediario entre el Modelo y la Vista. Responde a las interacciones del usuario en la Vista. Luego, actualiza el Modelo en caso de que sea necesario y notifica a la Vista para que se actualice la información que debe mostrar.

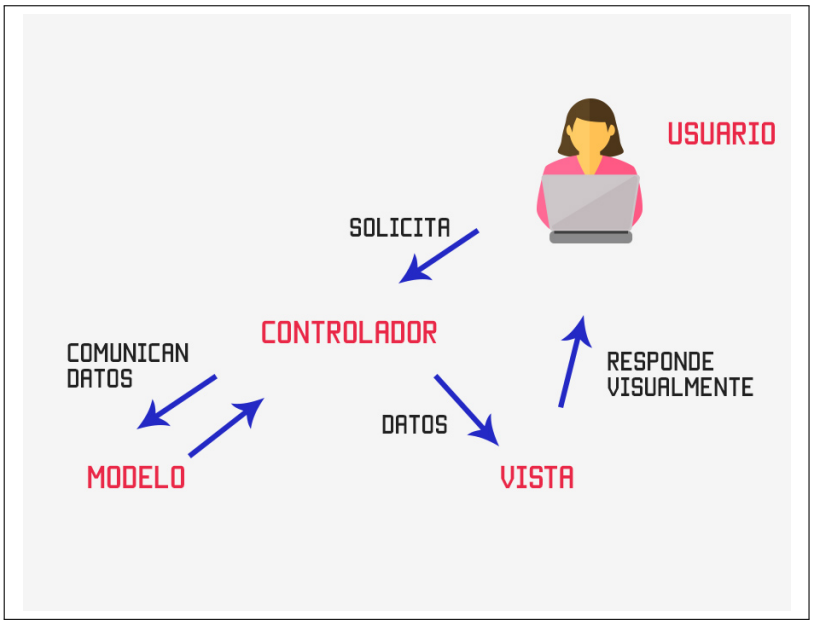

Figura 23: Diagrama del Modelo-Vista-Controlador

Una vez explicada la arquitectura seguida ya se pueden explicar las clases de la aplicación. Respecto a las clases correspondientes al Modelo existen dos

tipos de clases. Por un lado tenemos las clases encargadas de definir que formato tendrán los datos de la aplicación:

- **Exam**: Define que estructura y que propiedades tendrán los datos relativos a los exámenes.
- **User**: Define que estructura y que propiedades tendrán los datos relativos a los usuarios de la aplicación.
- **Subject**: Define que estructura y que propiedades tendrán los datos relativos a las asignaturas.
- **ExerciseUpload**: Define que estructura y que propiedades tendrán los datos relativos a las actividades.

Por otro lado, las clases encargadas de almacenar la información necesaria para la aplicación son:

- **ExamRepository**: Almacena y gestiona la información sobre las actividades.
- **UserRepository**: Almacena y gestiona la información sobre los usuarios.
- **SubjectRepository**: Almacena y gestionala información sobre las asignaturas.
- **ExerciseUploadRepository**: Almacena y gestiona la información sobre las actividades.

Las clases encargadas de la interacción con la aplicación por lo tanto las relativas a la Vista son:

- **AccountController**: Es la clase encargada de gestionar las peticiones relacionadas con las cuentas de los usuarios, por ejemplo recibe las solicitudes para registrar un usuario, editar un usuario y obtener el usuario actual entre otras.
- **AdminController**: Es la clase encargada de gestionar las peticiones relacionadas con las actividades que puede llevar a cabo el administrador de la aplicación, por ejemplo recibe las solicitudes para crear una actividad y editar una actividad.
- **CalificationController**: Es la clase encargada de gestionar las peticiones relacionadas con las calificaciones de una actividad, por ejemplo recibe las solicitudes para establecer una calificación y para ver todas las calificaciones.
- **ExamController**: Es la clase encargada de gestionar las peticiones relacionadas con las actividades, por ejemplo recibe las solicitudes para obtener la información de una actividad, editar una actividad y borrar una actividad entre otras.
- **LoginRestControlller**: Es la clase encargada de gestionar las peticiones relacionadas con el *login* y el *logout* de los usuarios.
- **SubjectControlller**: Es la clase encargada de gestionar las peticiones relacionadas con las asignaturas de la aplicación, por ejemplo recibe las soli-

citudes para obtener la información de una asignatura o saber si un usuario es profesor de una asignatura.
- **UploadController**: Es la clase encargada de gestionar las peticiones relacionadas con las entregas de las actividades.
- **LtiLoginControlller**: Es la clase encargada de gestionar las peticiones relacionadas con el protocolo LTI.

Finalmente, las clases encargadas de implementar la parte correspondiente al Controlador son los servicios. Estas clases son bastante más heterogéneas que las dedicadas al Modelo o a la Vista y también son más numerosas por lo que no se explicarán de forma individualizada. Dichas clases constituyen la lógica de la aplicación y son llamadas por las clases relativas a la Vista y utilizan las clases del Modelo para llevar a cabo las tareas que les corresponden.

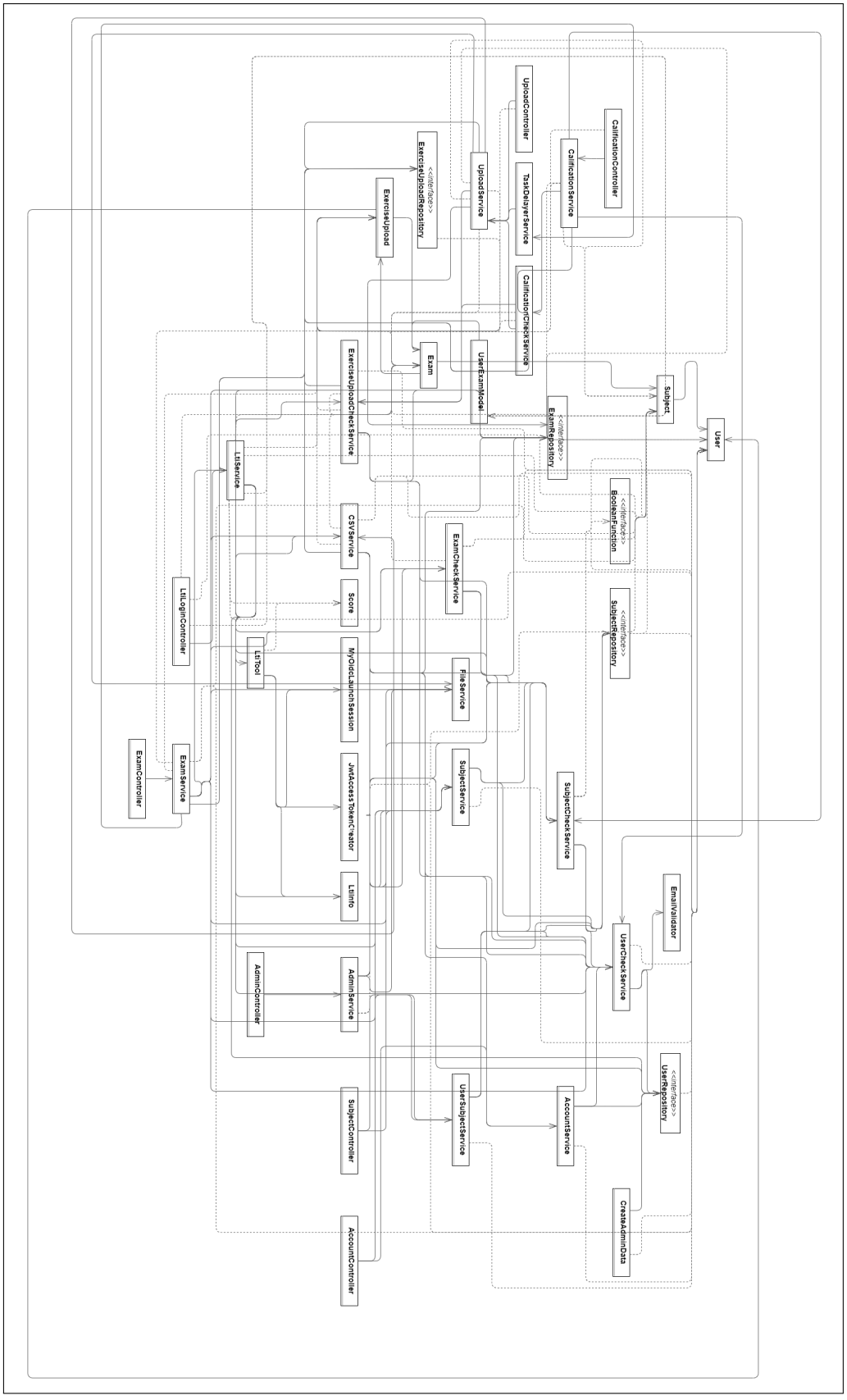

Figura 24: Diagrama de clases del backend

4.2.2. *Frontend*

En esta sección se explicarán las características generales de las clases del *frontend* sin ahondar en cada una de ellas. Como se ha mencionado anteriormente, su implementación está basada en componentes y estos componentes están desarrollados a partir de las siguientes clases:

- **Login**: Componente que permite a un usuario realizar el *login*.
- **Register**: Componente que permite a un usuario registrar una nueva cuenta.
- **Register**: Componente que permite a un usuario registrar una nueva cuenta.
- **Home**: Componente que permite ver todas las asignaturas a las que tiene acceso el usuario.
- **SubjectPage**: Componente encargado de mostrar la información de una asignatura y sus actividades a sus alumnos y profesores.
- **ExamPage**: Componente que muestra toda la información de una actividad al usuario.
- **CreateExam**: Componente que permite a un profesor crear una actividad en para una asignatura que imparte.
- **CreateSubject**: Componente que permite al administrador crear una nueva asignatura.
- **ErrorComponent**: Componente encargado de mostrar un error que se haya producido en la aplicación

En la parte del *frontend* al igual que en el *backend* se han implementado clases de Servicios que son proporcionados a los componentes mediante la inyección de dependencias de Angular. Estos servicios manejan la comunicación con el *backend* y realizan acciones comunes a varios componentes, entre ellos se encuentran:

- **AuthService**: Clase encargada de realizar las peticiones correspondientes a la sesión del usuario así como aquellas para realizar el *login* y *logout*.
- **LocalStorageService**: Clase encargada de guardar en el almacenamiento del navegador la información que le soliciten los componentes.
- **UploadService**: Clase encargada de realizar las peticiones relacionadas con las entregas de las actividades así como de descargar los archivos que proporciona el *backend*.
- **ExamService**: Clase encargada de realizar las peticiones relacionadas con las actividades de las asignaturas.

Cabe destacar que también se han utilizado algunas clases para definir los modelos de datos que utiliza el *frontend*. Aunque no se comentarán estas clases ya que son derivadas de las que aparecen en el *backend*, si que cabe la pena mencionar que se ha utilizado el patrón *Data Transfer Object* (DTO) [28].

El propósito principal del patrón de diseño DTO es encapsular datos y trans-

ferirlos de manera eficiente, ya que a menudo es necesario enviar solo un cierto subconjunto de los datos de un objeto más grande o combinar datos de múltiples objetos en uno solo antes de enviarlos. Un buen ejemplo de este patrón en PCEAIM puede ser el objeto que representa a los usuarios. Este objeto tiene un campo contraseña que no será utilizado por el *frontend* pero no tiene una lista de las asignaturas en las que está matriculado, por lo tanto se crea un DTO con todos los campos del usuario excluyendo la contraseña y añadiendo la lista de las asignaturas.

Uno de los beneficios de este patrón de diseño es que optimiza la comunicación entre el cliente y el servidor al reducir la cantidad de información que se intercambian en las comunicaciones. Otro de los beneficios que presenta es que se aumenta la seguridad de la aplicación al no enviar más datos de los que el cliente necesita, de forma que se controla en cada momento que información se envía y por lo tanto se evita incluir información sensible en las comunicaciones.

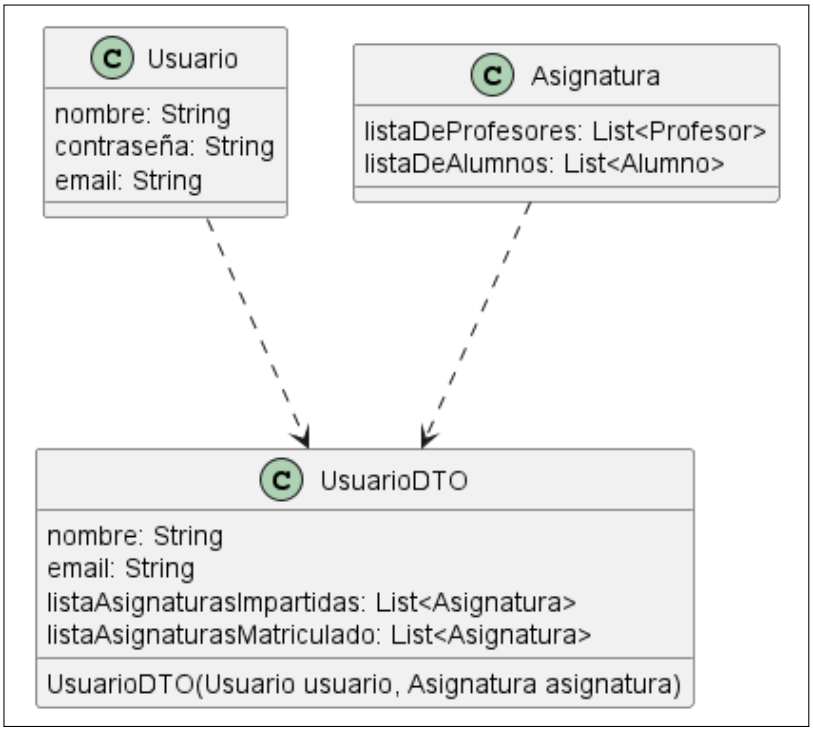

Figura 25: Ejemplo patrón DTO

4.3. Diseño e implementación

En este apartado se realizará un análisis más profundo de algunos aspectos complejos cuya implementación ha sido necesaria durante el desarrollo de la aplicación.

4.3.1. Protocolo LTI 1.3

Durante todo este documento se ha comentado en numerosas ocasiones que PCEAIM implementaba el protocolo LTI. En esta sección se ahondará en la explicación de este protocolo así como en la implementación que se ha seguido en para implementarlo en esta aplicación.

El protocolo LTI (Learning Tools Interoperability) 1.3 [29] es una especificación técnica definida por IMS Global que se utiliza en el campo de la educación para facilitar la interoperabilidad entre sistemas de aprendizaje. LTI 1.3 es una evolución de versiones anteriores y se ha convertido en un estándar ampliamente adoptado en la integración de herramientas y recursos de aprendizaje en plataformas educativas.

LTI 1.3 presenta varias funcionalidades entre las que se encuentran:

- **Inicio de sesión automático**: LTI permite iniciar sesión automáticamente en la plataforma externa si se accede a ella a través de un enlace compatible dentro de la plataforma educativa (si no está registrado en la plataforma externa se le puede registrar automáticamente).
- **Envío de calificaciones a la plataforma educativa**: LTI permite que la plataforma externa envíe automáticamente las calificaciones a la plataforma docente realizando una llamada a un *endpoint* definido por la plataforma docente.
- **Seguridad**: LTI 1.3 utiliza JSON Web Tokens (JWT) para proporcionar una mayor seguridad en las comunicaciones entre el LMS y las herramientas externas. Esto garantiza que la la transmisión de información sea segura y no pueda ser manipulada.

Para poder implementar el protocolo LTI en una aplicación lo primero que necesitamos es generar un par de claves RSA, estas claves son las que permitirán a la plataforma educativa y a la herramienta externa comunicarse a través de datos cifrados.

El primer paso para comunicar dos plataformas con el protocolo LTI es la solicitud de *login* enviada desde la plataforma docente a la herramienta externa. En esta solicitud se envía un parámetro *request* con ciertos datos que se tendrán que utillizar para firmar la solicitud y un parámetro *response* que es la estructura que se utilizará para enviar la solicitud a la plataforma docente.

Una vez recibidos estos datos tenemos que rellenar la solicitud con parte de los datos que ha enviado la plataforma docente y con ciertos datos que nos ha proporcionado la misma plataforma cuando se ha registrado la herramienta externa. De esta forma la plataforma podrá verificar que la solicitud se realiza de forma confiable. Una vez se ha rellenado la solicitud tendremos que enviarla al *endpoint* que ha proporcionado la plataforma docente en el campo *request*.

Una vez se ha envíado la solicitud correctamente y la plataforma docente ha confirmado que el servidor es el adecuado, esta enviará una nueva petición con los datos del usuario que la ha realizado así como información extra que se le ha podido incluir, en el contexto de PCEAIM se ha añadido el nombre de la asignatura y de la actividad a la que se quiere acceder.

Cuando se hayan recibido los datos, lo primero que hay que validar es que estos estén correctamente firmados con la información confidencial que conocen ambas plataformas. En caso de que así sea se extráen los datos enviados por la plataforma docente. En primer lugar se extraén los datos relativos al usuario que realiza la petición. En caso de que este usuario esté registrado en PCEAIM se inicia sesión en su cuenta, en caso contrario se registra al usuario y se inicia sesión. En ambos casos se almacena el identificador asociado al usuario que ha proporcionado la plataforma docente.

La próxima tarea que se realiza es la obtención de la información sobre la asignatura y la actividad a la que se quiere acceder. Una vez se han obtenido estos datos se comprueba que ambas existan en la aplicación. En caso de que el usuario fuera el profesor de la asignatura en la plataforma docente y no estuviera establecido en PCEAIM se establece como profesor de la asignatura en PCEAIM, en caso de que fuera alumno se realiza lo mismo solo que se le matricula como alumno de la asignatura. Finalmente se guarda el enlace al *endpoint* que ha proporcionado la plataforma docente en la petición cuya finalidad es recibir las calificaciones de los alumnos.

Una vez realizado todo lo anterior se genera un token JWT con los datos confidenciales compartidos entre ambas plataformas y se devuelve una respuesta con el estado HTTP 302 indicando que se debe redirigir a la ruta a la que corresponde la actividad en PCEAIM y firmada con el token JWT que se ha generado. En caso de que todo haya funcionado correctamente la plataforma externa redirigirá al usuario a la actividad correspondiente en PCEAIM sin que el usuario haya tenido que realizar ninguna actividad más allá de pulsar un enlace.

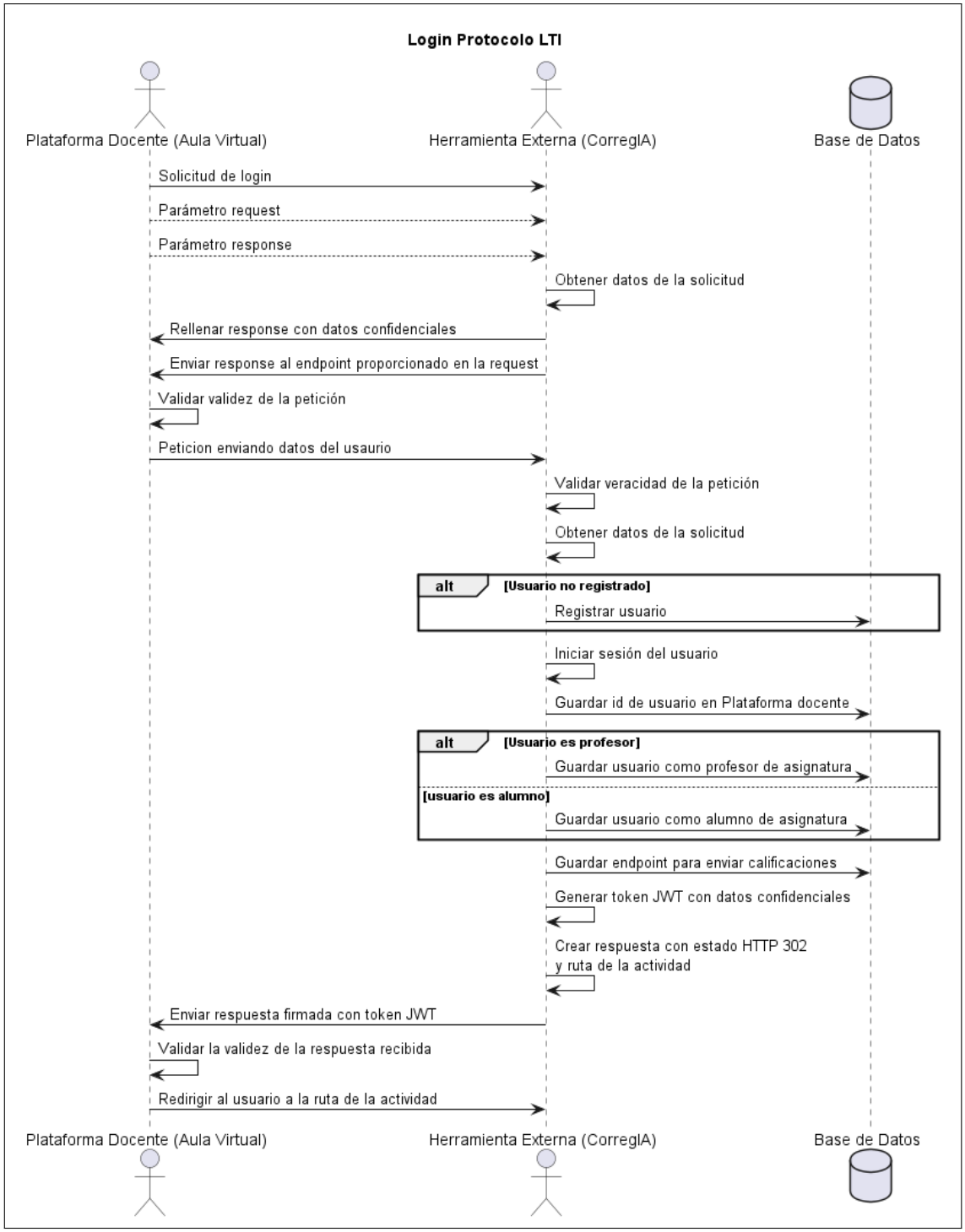

Figura 26: Diagrama de flujo login con LTI

4.4. Pruebas

En todos los desarrollos se debería probar cada funcionalidad que ha sido implementada, ya que en caso contrario la calidad de nuestro producto puede verse lastrada por los errores que pueden ir surgiendo. En los sistemas complejos las pruebas se complican, esto se debe a que suele haber que probar numerosos sistemas de forma independiente así como la forma en la que estos interactúan. Es por este motivo por lo que existen las pruebas unitarias, que comprueban la funcionalidad de partes aisladas del sistema, las pruebas de integración, las cuales prueban la conexión entre diferentes partes de la aplicación, además de las pruebas de extremo a extremo (End to End o E2E), las cuales prueban tido el sistema desde una visión de alto nivel. El tema de las pruebas en los sistemas software fue tratado en profundidad por Mike Cohn en el libro Succeding with Agile [30], proponiendo la clasificación de los esfuerzos de las pruebas que se puede ver en la siguiente figura.

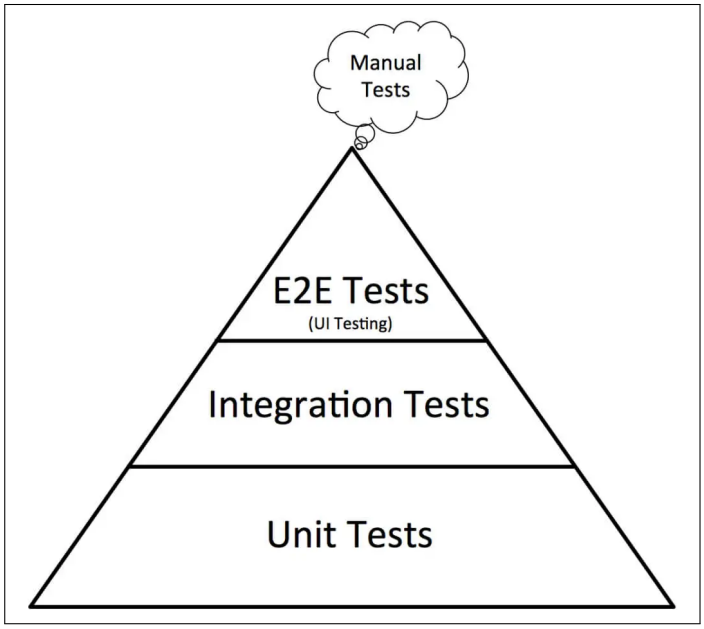

Figura 27: Pirámide de Pruebas

El desarrollo de las pruebas estuvo centrado en ambos extremos de la pirámide, es decir, en las pruebas de extremo a extremo y las pruebas unitarias. Además, después de que cada fase de desarrollo hubiera finalizado, se realizaron pruebas manuales, llevadas a cabo en diferentes navegadores.

Las pruebas unitarias realizadas *testean* distintas partes de la aplicación de forma aislada al resto de las partes. Estas pruebas no necesitan que la aplicación esté en funcionamiento y se pueden ejecutar tanto a través de un IDE. La cobertura de las pruebas unitarias es:

- Creación y edición de usuarios.

- Creación, edición y borrado de asignaturas.
- Creación y edición de actividades de todos los tipos.
- Establecimiento, edición y visualización de calificaciones.
- Entrega de actividades.
- Validación de la estructura de emails de los usuarios.
- Visualización de la información de las asignaturas.
- Obtención de la información de todos los usuarios.
- Obtención de la información del usuario *logeado*.
- Obtención de la información de las actividades de cualquiera de los tipos.
- Obtención de la información de una asignatura específica y de todas las asignaturas.
- Relación de un usuario con una asignatura.

Figura 28: Pruebas unitarias realizadas

También se han realizado pruebas de extremo a extremo utilizando Selenium. Estas pruebas permiten tener una cobertura muy amplia de la aplicación, ya que la prueban a nivel de interfaz simulando distintos procesos que realizaría un usuario de forma similar a la que él ejecutaría las acciones. Las pruebas de extremo a extremo son mucho más lentas que las pruebas unitarias pero permiten probar la aplicación de forma más realista, además de probar de forma indirecta la integración entre los sistemas, motivo por el cual se ha decidido no realizar pruebas de integración.

A continuación se listará la cobertura sobre la interfaz que tienen estas pruebas:

- Registro de un usuario en la plataforma.
- Inició de sesión y cierre de sesión de un usuario.
- Creación de asignaturas.
- Edición de asignaturas.
- Borrado de asignaturas.
- Creación de actividades de ambos tipos.

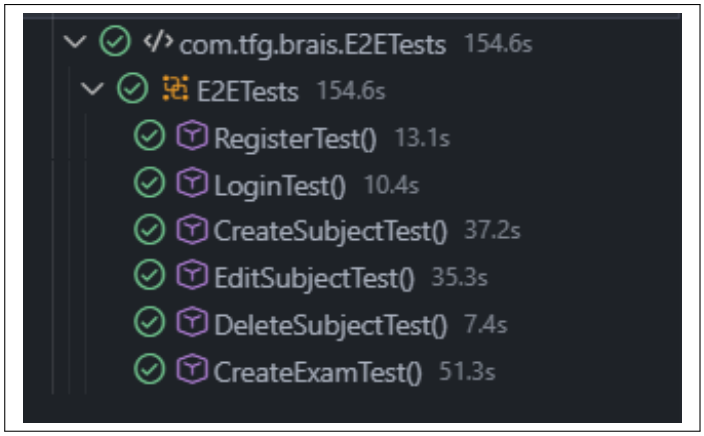

Figura 29: Pruebas extremo a extremo realizadas

5

Conclusiones y trabajos futuros

5.1. Objetivos conseguidos

El proyecto desarrollado es una herramienta muy interesante para el área de la educación, ya que permite agilizar los procesos de calificación de actividades y de comunicación con los alumnos. También es destacable la parte de poder conectar la herramienta con plataformas docentes de forma sencilla, lo que permite que su adopción sea muy rápida y fácil sin una gran barrera para el usuario que la quiera usar.

De cara a la aplicación y a su implementación, hay que tener en cuenta que todo el desarrollo ha estado realizado por una única persona por lo que para ser un proyecto individual si que se puede considerar bastante extenso y laborioso. Sin embargo, se han cumplido satisfactoriamente todos los objetivos que se habían marcado desde un principio.

En primer lugar, se ha conseguido desarrollar una plataforma docente en la que se pueden realizar las funciones necesarias para poder enviar tareas a los alumnos y calificar estas tareas. Además de todo lo anterior la aplicación presenta una interfaz web intuitiva, además de ser *responsive*, pudiendo ser utilizada de forma cómoda desde cualquier dispositivo(desde móviles a ordenadores).

En segundo lugar, es importante considerar que la aplicación dispone de integración automática con otras plataformas docentes al implementar el protocolo LTI. En relación con lo anterior, es crucial destacar el extenso trabajo de investigación que se ha llevado a cabo para comprender como funcionaba esta

tecnología, lo que ha sido bastante complicado debido a la poca documentación que hay disponible.

En tercer y último lugar uno de los puntos claves, sobretodo si se tiene en cuenta que este proyecto tiene fines educativos, es el aprendizaje y utilización de tecnologías no conocidas anteriormente por el autor. Respecto a esto tenemos el ejemplo de la Integración Continua y del Despliegue continuo o los validadores de Angular. Ya que estas tecnologías no eran conocidas por el autor y han sido utilizadas exitosamente para desarrollar la aplicación.

5.2. Trabajos futuros

Hay varias funcionalidades o puntos que sería importante e interesante realizar en un futuro, pero que debido a la falta de tiempo por el plazo tan limitado de entrega no se han podido llevar a cabo. Los puntos en cuestión son los siguientes:

- Tareas en grupo: Se debería introducir la posibilidad de crear actividades grupales y que estos grupos se pudieran crear dentro de la propia aplicación.
- Protocolo OAUTH: Se debería implementar el protocolo OAUTH para que los usuarios pudieran registrarse con sus cuentas de Google de forma automática.
- Editor de código integrado: Sería interesante añadir un editor de código dentro de la propia aplicación para que los alumnos puedan realizar tareas de programación directamente en la aplicación. Este editor podría estar preconfigurado por el docente.
- Incorporación de Inteligencia Artificial: Como se ha ido comentando durante todo el documento, una importante mejora a futuro para la aplicación sería la conexión con inteligencia artificial para la calificación de tareas y retroalimentación sobre las mismas de forma automática sin la necesidad de intervención del docente.

5.3. Conclusiones personales

Este proyecto ha supuesto para mí un primer acercamiento relativamente serio al mundo de la programación. El hecho de haber desarrollado un proyecto semi-real de forma autónoma utilizando las tecnologías más empleadas en el mercado laboral me ha permitido desarrollar una buena capacidad de búsqueda y recolección de información. En relación con a la formar de trabajo, me ha permitido trabajar de forma Ágil y utilizar las metodologías más punteras de desarrollo del software lo que considero una muy valiosa lección para mi futuro laboral.

Bibliografía

[1] Página principal typescript. [Online]. Available: https://www.typescriptlang.org/

[2] Página principal angular. [Online]. Available: https://angular.io

[3] Página principal java. [Online]. Available: https://www.java.com/es/

[4] Página principal spring boot. [Online]. Available: https://spring.io/projects/spring-boot

[5] Página principal openjdk. [Online]. Available: https://openjdk.org

[6] K. Lawson. (2018) What is a single page application? Accessed on 2nd November 2023. [Online]. Available: https://www.bloomreach.com/en/blog/2018/what-is-a-single-page-application

[7] Página principal jquery. [Online]. Available: https://jquery.com

[8] Página principal angular material. [Online]. Available: https://material.angular.io

[9] Página principal spring framework. [Online]. Available: https://spring.io

[10] Página principal spring data. [Online]. Available: https://spring.io/projects/spring-data

[11] Página principal jpa. [Online]. Available: https://jakarta.ee/specifications/persistence/

[12] Página principal mysql. [Online]. Available: https://www.mysql.com

[13] Página principal opencsv. [Online]. Available: https://opencsv.sourceforge.net

[14] Página principal javalti. [Online]. Available: https://github.com/UOC/java-lti-1.3

[15] Página principal docker. [Online]. Available: https://www.docker.com

[16] Página principal junit. [Online]. Available: https://junit.org/junit5/

[17] Página principal mockito. [Online]. Available: https://site.mockito.org

[18] Página principal selenium. [Online]. Available: https://www.selenium.dev

[19] Página principal maven. [Online]. Available: https://maven.apache.org

[20] Página principal npm. [Online]. Available: https://www.npmjs.com

[21] Página principal mysql workbench. [Online]. Available: https://www.mysql.com/products/workbench/

[22] Página principal git. [Online]. Available: https://git-scm.com

[23] Página principal github. [Online]. Available: https://git-scm.com

[24] Página principal trello. [Online]. Available: https://trello.com/es

[25] Página principal google cloud run. [Online]. Available: https://cloud.google.com/run/

[26] Página docuementación github actions. [Online]. Available: https://github.com/features/actions

[27] J. Wilkins. (2021) Mvc architecture – what is a model view controller framework? Accessed on 4th November 2023. [Online]. Available: https://www.freecodecamp.org/news/mvc-architecture-what-is-a-model-view-controller-framework/

[28] F. Salas. (2023) Efficient data transfer in rest apis: A deep dive into the dto pattern with spring boot and mysql. Accessed on 4th November 2023. [Online]. Available: https://blog.stackademic.com/efficient-data-transfer-in-rest-apis-a-deep-dive-into-the-dto-pattern-with-spring-boot-and-mysql-df2bdf1ece74

[29] Página principal lti. [Online]. Available: https://www.imsglobal.org/spec/lti/v1p3

[30] M. Cohn, *Succeeding with Agile*. Addison-Wesley Professional, 2009, páginas 311-317.