EasyFCT

Luis Carlos Castañeda Herrera
y Thomas Dragutan Turcan
CFGS Desenvolupament
d'Aplicacions Multiplataforma

INSTITUT PUIG CASTELLAR

Resumen del proyecto

EasyFCT es un proyecto diseñado para transformar la gestión de las prácticas formativas en los centros de Formación Profesional (FP). Su objetivo principal es superar las limitaciones de sistemas actuales como qBid, ofreciendo una solución moderna, eficiente y adaptable a las necesidades específicas de cada centro. Con una interfaz intuitiva y funcionalidades avanzadas, EasyFCT mejora la experiencia de uso y facilita la interacción entre los diferentes usuarios del proceso de Formación en Centros de Trabajo (FCT).

La aplicación está diseñada para gestionar empresas, profesores, alumnos y administradores de forma centralizada. Las empresas podrán registrar ofertas de prácticas, gestionar vacantes y hacer seguimiento del progreso de los estudiantes. Los profesores contarán con herramientas para asignar prácticas a sus alumnos, gestionar sus avances y mantener una comunicación directa con las empresas. Los alumnos tendrán acceso a un panel personalizado donde podrán consultar ofertas, postularse y recibir retroalimentación. Los administradores, por su parte, supervisarán todos los datos y procesos del sistema, garantizando un funcionamiento eficiente.

Con EasyFCT, se busca no solo optimizar los procesos de gestión, sino también crear una plataforma versátil que facilite la comunicación, la asignación y el seguimiento de las prácticas, mejorando significativamente la calidad y la eficacia del sistema actual.

Palabras Claves

1. Gestión de prácticas
2. Aplicación multiplataforma
3. Asignación de prácticas
4. Seguimiento de alumnos

Aquesta obra està subjecta a una llicència de
Reconeixement-CompartirIgual 3.0 Espanya de Creative Commons

Abstract

EasyFCT is a project designed to transform the management of internships in Professional Training (FP) centers. Its main objective is to overcome the limitations of current systems like qBid, providing a modern, efficient, and adaptable solution tailored to the specific needs of each center. With an intuitive interface and advanced features, EasyFCT enhances the user experience and facilitates interaction among the various participants in the Internship Training Program (FCT) process.

The application is designed to centrally manage companies, teachers, students, and administrators. Companies will be able to register internship offers, manage vacancies, and track the progress of students. Teachers will have tools to assign internships to their students, monitor their progress, and maintain direct communication with companies. Students will have access to a personalized dashboard where they can browse internship opportunities, apply, and receive feedback. Administrators, on the other hand, will oversee all data and system processes, ensuring efficient operation.

With EasyFCT, the goal is not only to optimize management processes but also to create a versatile platform that facilitates communication, assignment, and monitoring of internships, significantly improving the quality and efficiency of the current system.

Keyboards

1. Internship management
2. Multiplatform application
3. Internship assignment
4. Student tracking

ÍNDICE

1. Introducción

Las Formaciones en Centros de Trabajo (FCT) son una parte clave en la formación profesional, ya que permiten a los alumnos poner en práctica lo aprendido en clase y les ayudan a prepararse para el mundo laboral. Sin embargo, gestionar estas prácticas suele ser un dolor de cabeza para los centros educativos. Hoy en día, muchos dependen de programas antiguos o soluciones parciales, lo que obliga a usar hojas de cálculo, correos electrónicos o varias herramientas a la vez. Esto genera un trabajo extra y poco eficiente para los profesores.

Un ejemplo claro de estas limitaciones es la plataforma qBid, una herramienta muy extendida que, aunque cumple su función básica, se queda corta. Su interfaz no es muy atractiva ni moderna, y la falta de interactividad hace que usarla sea un poco frustrante. Por ejemplo, encontrar el diario de prácticas no es intuitivo, y la información no siempre está clara, especialmente para los alumnos nuevos. Además, los estudiantes no pueden buscar prácticas por su cuenta, lo que deja toda la carga de asignación en los profesores. También hemos notado que los profesores se quejan a menudo de la gestión y subida de documentos, lo que demuestra que se necesita una herramienta más completa y fácil de usar.

Por todo esto, hemos creado **EasyFCT**, una solución innovadora pensada para mejorar la gestión de las FCT y superar los problemas de las plataformas actuales. EasyFCT es una herramienta moderna, flexible y eficaz, con una interfaz intuitiva y funciones avanzadas que simplifican el proceso y mejoran la comunicación entre todos los involucrados: alumnos, profesores, empresas y administradores.

Las funcionalidades clave que diferencian a EasyFCT incluyen:

- **Autogestión para Alumnos:** Los estudiantes pueden acceder a la plataforma desde el principio, lo que les permite postularse directamente a las ofertas de prácticas. Esto quita trabajo al profesorado en la fase de asignación inicial.
- **Sistema de 'Match' Inteligente:** Hemos incluido una función que calcula cuánto encaja el perfil de un alumno con las habilidades que pide una empresa en su oferta. Así, se facilita una conexión más eficiente y adecuada entre alumnos y prácticas.
- **Gestión Completa de Ofertas para Empresas:** Las empresas tienen un espacio propio donde pueden aceptar o rechazar candidatos, y también ver qué alumnos tienen en su empresa o quiénes se han postulado a sus ofertas.
- **Comunicación Directa Profesor-Administrador:** Creamos un canal directo para que profesores y administradores puedan comunicarse. Esto acelera la resolución de problemas y la corrección de errores, haciendo el proceso más eficiente.

Todo esto se traduce en ventajas claras para los usuarios:

- **Ahorro de tiempo para el profesorado:** Al permitir que los alumnos busquen y se postulen solos, los profesores disponen de más tiempo para centrarse en guiar y apoyar a los estudiantes.
- **Mejor experiencia para el alumno:** Una aplicación más atractiva y fácil de usar hace que los alumnos se sientan más cómodos y motivados, facilitando la gestión de su diario de prácticas y el acceso a oportunidades.
- **Mayor eficiencia y transparencia:** Una comunicación fluida y la centralización de la información ayudan a reducir los errores administrativos y a tener una visión clara del estado de las FCT para todos.

EasyFCT es una **aplicación web** que se adapta a cualquier dispositivo gracias a su diseño *responsive*. Está construida con una clara **separación entre el frontend** (la parte visible para el usuario) **y el backend** (la lógica interna y la base de datos). Esta forma de trabajar nos permite desarrollar ambas partes al mismo tiempo, lo que facilita la integración y asegura que el sistema sea fácil de mantener y de escalar en el futuro.

Este proyecto ha sido desarrollado durante el último curso de Grado Superior de Desarrollo de Aplicaciones Multiplataforma (DAM) y es nuestro **Proyecto de Fin de Ciclo (PFC)**. Su creación surge precisamente de nuestra propia experiencia con las limitaciones de las herramientas actuales que se usan para las FCT.

Todo esto se reflejará en el siguiente documento.

1.1 Contexto y justificación

El sistema actual para la gestión de las Formaciones en Centros de Trabajo (FCT) en los centros educativos, qBid, presenta una serie de limitaciones significativas que complican una administración eficiente del proceso. Entre sus principales inconvenientes, destaca una falta de funcionalidad para ciertos roles y una interfaz poco intuitiva, lo que afecta directamente la experiencia de todos los usuarios.

Desde la perspectiva del **profesorado y la administración**, la plataforma actual carece de la flexibilidad necesaria para gestionar de forma autónoma tareas clave. Por ejemplo, la comunicación entre profesores y administradores es limitada, lo que dificulta la rápida resolución de incidencias o la erradicación de errores en la gestión de expedientes o informes. Además, no se facilita la gestión integral de las ofertas de prácticas, requiriendo en ocasiones procesos externos o manuales. Para el **alumnado**, la plataforma actual no ofrece opciones para buscar prácticas de forma proactiva, dependiendo totalmente de las asignaciones del profesor. Esta dependencia genera ineficiencias y una carga administrativa adicional.

Ante estas deficiencias, se ha desarrollado **EasyFCT**. Este nuevo sistema busca optimizar la gestión de las prácticas en centros de Formación Profesional, ofreciendo una plataforma intuitiva, moderna y fácil de usar. Su diseño es **flexible** en el sentido de que cada rol (alumno, profesor, empresa) puede trabajar de forma más autónoma, reduciendo las dependencias directas entre ellos. Por ejemplo, los alumnos pueden buscar y postularse a ofertas por sí mismos, y las empresas pueden publicar y gestionar sus vacantes directamente. Esto es crucial para digitalizar y simplificar la gestión, optimizar recursos, mejorar la experiencia del usuario y asegurar un seguimiento efectivo del alumnado, aspectos en los que la plataforma actual no cumple las expectativas.

Se espera que EasyFCT contribuya a una mejora sustancial en la calidad de la gestión de las FCT, generando un impacto positivo en la formación de los estudiantes y en la relación con el entorno empresarial. Esto se logra mediante:

- **Mejor adecuación de las prácticas:** El sistema de "match" inteligente permite que los alumnos se postulen a ofertas que realmente se alinean con sus conocimientos y habilidades, lo que facilita encontrar prácticas más significativas y reduce el tiempo de búsqueda.
- **Decisiones más informadas para las empresas:** Las empresas pueden acceder al perfil del alumno antes de aceptar o rechazar una candidatura, lo que les permite seleccionar perfiles que se ajusten mejor a sus necesidades, minimizando el riesgo de incorporar a alguien sin los conocimientos necesarios.
- **Reducción de la carga administrativa y errores:** La autonomía de los roles y la comunicación directa (como la de profesor a administrador) agilizan los

procesos y permiten identificar y solucionar problemas de manera más rápida y eficiente.
- **Claridad y eficiencia en la gestión de ofertas:** Las empresas pueden visualizar sus ofertas activas y las postulaciones recibidas en un mismo lugar, evitando duplicidades y simplificando el seguimiento.

1.2 Objetivos

La creación de EasyFCT persigue un objetivo principal ambicioso: optimizar la gestión integral de las Formaciones en Centros de Trabajo (FCT) y simplificar sus procesos para todos los usuarios implicados. Esto incluye no solo facilitar la búsqueda de prácticas para los alumnos, sino también mejorar la eficiencia y la experiencia general durante el desarrollo de dichas prácticas. Se busca establecer una plataforma que sea una herramienta moderna, intuitiva y eficaz para alumnos, profesores y empresas.

Además de este objetivo fundamental relacionado con el producto, el desarrollo de EasyFCT ha representado un valioso aprendizaje práctico en la gestión integral del ciclo de vida de un proyecto de software de envergadura, desde la fase de análisis hasta la implementación.

Para alcanzar el objetivo general, se han definido los siguientes **objetivos específicos**:

- **A) Establecer un sistema robusto para el registro y gestión de empresas y ofertas de prácticas:**
    - Este objetivo resuelve la dispersión de información y la gestión manual, permitiendo a las empresas **publicar y administrar sus propias ofertas** de forma autónoma.
    - Aporta el beneficio directo de un **catálogo de ofertas centralizado y actualizado**, ofreciendo a las empresas un control total sobre su información y sobre los alumnos postulados a sus vacantes.

- **B) Proveer paneles de usuario personalizados y funcionales para cada rol:**
    - La personalización es crucial para asegurar una **experiencia de usuario adaptada y eficiente**, proporcionando acceso rápido a la información y funcionalidades más relevantes para cada tipo de usuario.
    - Concretamente, el **rol de empresa** puede publicar y revisar sus ofertas, el **rol de profesor** gestiona a sus alumnos y sus asignaciones, y el **rol de alumno** puede buscar, postularse y gestionar sus propias prácticas directamente desde la aplicación.

- **C) Desarrollar funcionalidades básicas para el seguimiento del progreso de las prácticas:**
    - Este objetivo se enfoca en permitir la **digitalización del diario de prácticas**, lo que contribuye a una recolección de datos más estandarizada y un seguimiento básico del avance del alumno durante su periodo formativo.

1.3 Estrategia y metodología del proyecto

Para la gestión y desarrollo de EasyFCT, se optó por una **metodología ágil adaptada, inspirada en los principios de Kanban**, con un fuerte énfasis en la **comunicación directa** y la **flexibilidad**. Dado el tamaño del equipo y la naturaleza del proyecto de fin de ciclo, no se consideró necesario implementar un marco rígido de ceremonias o iteraciones fijas como las que caracterizan a Scrum. En su lugar, se priorizó un flujo de trabajo continuo y una adaptación constante a las necesidades emergentes.

La estrategia se centró en la **visualización del trabajo y la gestión del progreso de manera colaborativa y transparente**. Aunque no se utilizó un tablero Kanban físico o digital con columnas estrictas, las tareas se organizaban y su estado se seguía mediante un **documento compartido en Google Drive**. Este documento servía como un registro centralizado de las funcionalidades pendientes, en desarrollo y completadas, proporcionando una visión clara del avance del proyecto para todos los miembros del equipo.

La **comunicación interpersonal** fue el pilar de la metodología. Las decisiones, los avances y los posibles ajustes en el desarrollo se discutían y acordaban de forma regular y directa entre los miembros del equipo. Esto permitió una **respuesta rápida a los cambios**, la identificación temprana de obstáculos y una asignación dinámica de responsabilidades, asegurando que el trabajo se mantuviera alineado con los objetivos.

Las fases generales del proyecto se abordaron de manera fluida, pero siguiendo una secuencia lógica:

- **Definición de Requisitos y Análisis:** Se realizó un análisis detallado de las necesidades del sistema y de las limitaciones de las plataformas existentes, estableciendo las funcionalidades clave de EasyFCT. Este proceso fue iterativo, permitiendo refinar los requisitos a medida que se profundizaba en el conocimiento del dominio.
- **Diseño y Arquitectura:** Se definieron las bases de la aplicación, incluyendo la arquitectura del frontend y backend, el diseño de la base de datos y la estructura de las APIs.
- **Desarrollo e Implementación:** Se procedió a la codificación de las distintas funcionalidades, trabajando de forma paralela en el frontend y el backend, lo que permitió avances simultáneos.

Para el **control de versiones del código fuente**, se utilizó **GitHub**. Esto permitió una gestión ordenada de los cambios, la colaboración efectiva entre los desarrolladores y la posibilidad de revertir a versiones anteriores si fuera necesario. Las diferentes funcionalidades se desarrollaron en ramas separadas, facilitando la integración de código una vez completadas y verificadas.

1.4 Estudio económico y presupuestario

El desarrollo de EasyFCT, en su fase de Proyecto de Fin de Ciclo (PFC), ha sido concebido y ejecutado bajo un modelo de **coste cero** en lo que respecta a la producción directa y las licencias de software. Esta estrategia ha sido posible gracias a la adopción de herramientas y recursos de **código abierto (Open Source)** y el uso de **infraestructura personal** de los miembros del equipo.

Los principales recursos utilizados sin implicar costes económicos directos han sido:

Software de desarrollo: Para el frontend, se ha empleado **React** dentro del entorno de desarrollo **Visual Studio Code**. Para el backend, se ha utilizado **IntelliJ IDEA** junto con **Spring Boot** para la lógica de negocio y **MySQL** como sistema de gestión de bases de datos. Todas estas herramientas ofrecen versiones comunitarias o licencias que se ajustan a las necesidades de un proyecto académico sin coste.

Hardware: El desarrollo y las pruebas se han realizado íntegramente en los **ordenadores personales** de los miembros del equipo, eliminando la necesidad de adquirir servidores o equipos adicionales dedicados a la producción o al desarrollo.

Herramientas de diseño y gestión: Para la elaboración de diagramas, se han utilizado **MySQL Workbench** y **Modelio**, ambos con opciones de uso gratuito.

La gestión de tareas y el seguimiento del progreso del proyecto se realizaron mediante un **documento compartido en Google Drive** y **comunicación directa** entre los integrantes del equipo.

Recursos de aprendizaje: El proceso de aprendizaje y la resolución de desafíos técnicos se apoyaron en recursos libremente accesibles como **vídeos de YouTube, herramientas de inteligencia artificial (IA)** y la invaluable **colaboración de compañeros y profesores**.

1.4.1. Escenario Futuro y Costes Potenciales

Si EasyFCT evolucionara de un proyecto académico a una **solución en producción real** que requiriera un mantenimiento continuo y escalabilidad, implicaría una serie de costes significativos a considerar. La planificación de estos costes es fundamental para la viabilidad a largo plazo del sistema:

Infraestructura de Alojamiento (Hosting): Sería necesario invertir en servicios de servidor robustos y escalables para almacenar la aplicación y la base de datos, garantizando su disponibilidad y rendimiento. Esto implicaría la contratación de proveedores de **servicios en la nube** (como Amazon Web Services - AWS, Google Cloud, Microsoft Azure, etc.), con costes asociados a la computación, almacenamiento, ancho de banda y bases de datos gestionadas.

Dominio y Certificados de Seguridad: La adquisición y mantenimiento de un **dominio web** propio, así como la implementación de **certificados SSL/TLS** para garantizar la seguridad de las comunicaciones (cifrado de datos), representarían costes recurrentes.

Mantenimiento y Soporte Técnico: Se requeriría un equipo de **programadores** para la resolución de posibles errores, la implementación de actualizaciones, la mejora de funcionalidades y la atención a consultas o problemas de los usuarios (profesores, alumnos, empresas). Esto implicaría salarios para personal cualificado.

Desarrollo continuo: La adición de nuevas funcionalidades y la adaptación a futuras necesidades del mercado o cambios normativos requerirían una inversión constante en recursos humanos y de desarrollo.

Infraestructura física (Opcional): En un modelo de auto alojamiento, se considerarían los costes de un **local físico** para albergar servidores y equipos de trabajo. No obstante, la tendencia actual en el desarrollo de software es hacia soluciones en la nube para reducir esta inversión inicial y operativa.

Licencias y Herramientas Avanzadas: Aunque inicialmente se han usado herramientas gratuitas, en un entorno profesional se podrían considerar la adquisición de **licencias de software** o herramientas premium que ofrezcan mayores capacidades de monitorización, seguridad, análisis de datos o facilidad de manejo para optimizar el flujo de trabajo y la gestión de la aplicación.

Seguridad y Auditorías: Inversiones en soluciones de seguridad más avanzadas y posibles auditorías periódicas para proteger los datos y la infraestructura.

La previsión de estos costes futuros es esencial para evaluar la sostenibilidad y la escalabilidad de EasyFCT como producto comercial o de servicio a largo plazo.

2. Descripción del proyecto

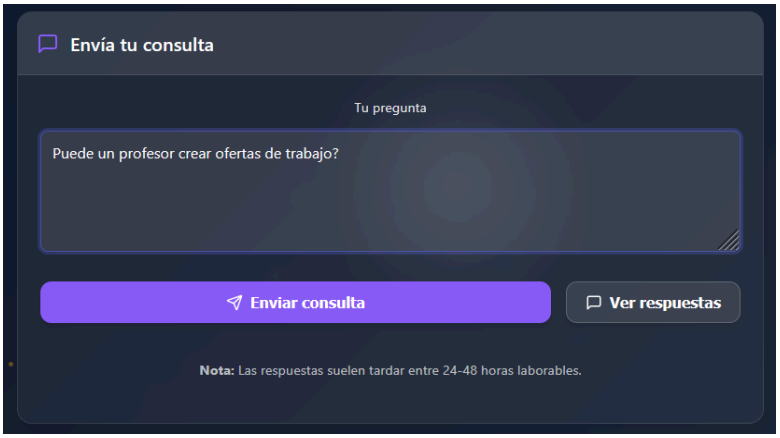
2.1 Análisis de requisitos

Para el desarrollo de EasyFCT se ha realizado un análisis de requisitos enfocado en resolver las principales limitaciones en plataformas actuales como qBid, caracterizadas por una experiencia de usuario poco intuitiva, una distribución ineficaz de funcionalidades entre roles, una escasa flexibilidad operativa y una interfaz básica. Esto afecta directamente a los distintos actores implicados en la gestión de las Formaciones en Centros de Trabajo (FCT): alumnos, profesores, empresas y administradores. En consecuencia, se dificulta una administración fluida, autónoma y eficiente del proceso formativo.

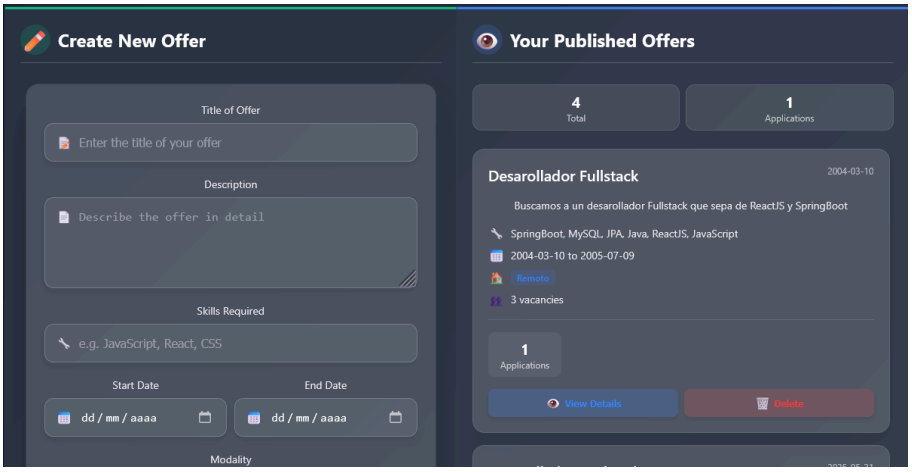
Durante esta fase, se han identificado y definido claramente los distintos roles del sistema, junto con sus necesidades específicas. Esto ha permitido establecer los requisitos funcionales y no funcionales que guían el diseño de EasyFCT, asegurando una solución más equilibrada y efectiva.

Los alumnos requieren mayor autonomía en la gestión de sus prácticas. El aplicativo permite editar su perfil, consultar y postularse a las ofertas disponibles, hacer seguimiento del estado de sus candidaturas y registrar su progreso mediante un diario digital de prácticas accesible y fácil de usar. De este modo, se promueve su participación activa en el proceso formativo y se reduce la dependencia del profesorado para tareas básicas.

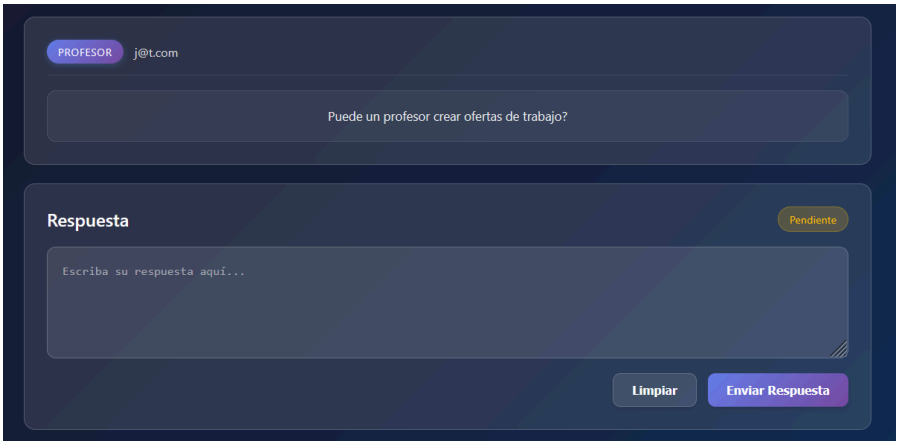
Para los docentes, se identificó la necesidad de reducir la carga administrativa y disponer de una visión centralizada del estado de sus alumnos y de las empresas colaboradoras. Entre sus funciones destacadas se incluyen: registrar nuevos alumnos, restablecer información en caso de errores, y validar el progreso de las prácticas. Además, contarán con un canal directo de comunicación con los administradores del sistema, lo que facilita la gestión de incidencias y mejora los tiempos de respuesta ante cualquier problema.



Las empresas colaboradoras requieren un entorno que les permita actuar con autonomía en la publicación, edición y gestión de sus ofertas. EasyFCT proporciona un espacio dedicado donde podrán crear vacantes, consultar el número de postulaciones recibidas, acceder a los perfiles de los candidatos, aceptar o rechazar solicitudes, y hacer seguimiento de los alumnos asignados. Esta independencia en la gestión les permite participar de manera más activa en el proceso, optimizando el ajuste entre oferta y demanda de prácticas.



Desde el punto de vista administrativo, se identificó la necesidad de disponer de herramientas que faciliten el control general del sistema y la gestión de incidencias. A través de formularios enviados por los profesores, los administradores podrán responder dudas, resolver errores y garantizar el correcto funcionamiento de la plataforma.

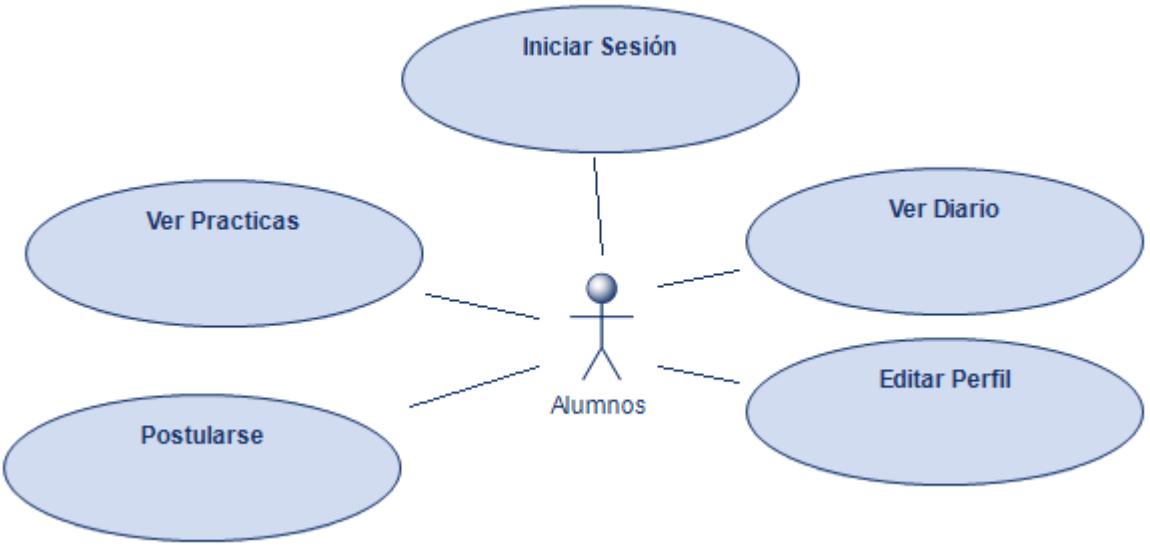


Este análisis de requisitos ha permitido no solo establecer las funcionalidades esenciales que debe ofrecer EasyFCT, sino también definir las características técnicas que un sistema de estas características debe cumplir. Entre ellas destacan la accesibilidad desde cualquier dispositivo, y la escalabilidad del sistema para soportar un creciente número de usuarios y la usabilidad de la interfaz.

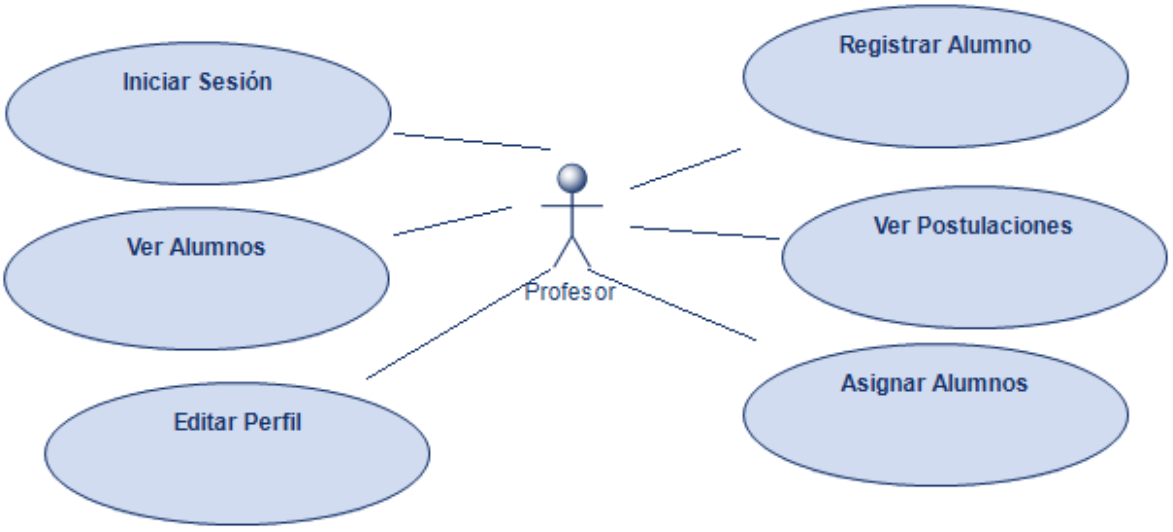
EasyFCT está orientado a cubrir las carencias de sistemas actuales, mejorando la eficiencia, la comunicación y la experiencia de los distintos perfiles implicados en la gestión de las prácticas formativas.

Las funcionalidades de cada rol entonces se categorizaron de la siguiente manera:

Alumnos:
Los alumnos pueden iniciar sesión, básicamente, en una pantalla donde pueden introducir sus credenciales y poder entrar a otras partes de la plataforma. Una vez iniciada sesión, los alumnos tendrán más funcionalidades como poder ver las prácticas ofertadas por las empresas, poder postularse a ellas, la opción de poder editar su perfil, y ver su diario.

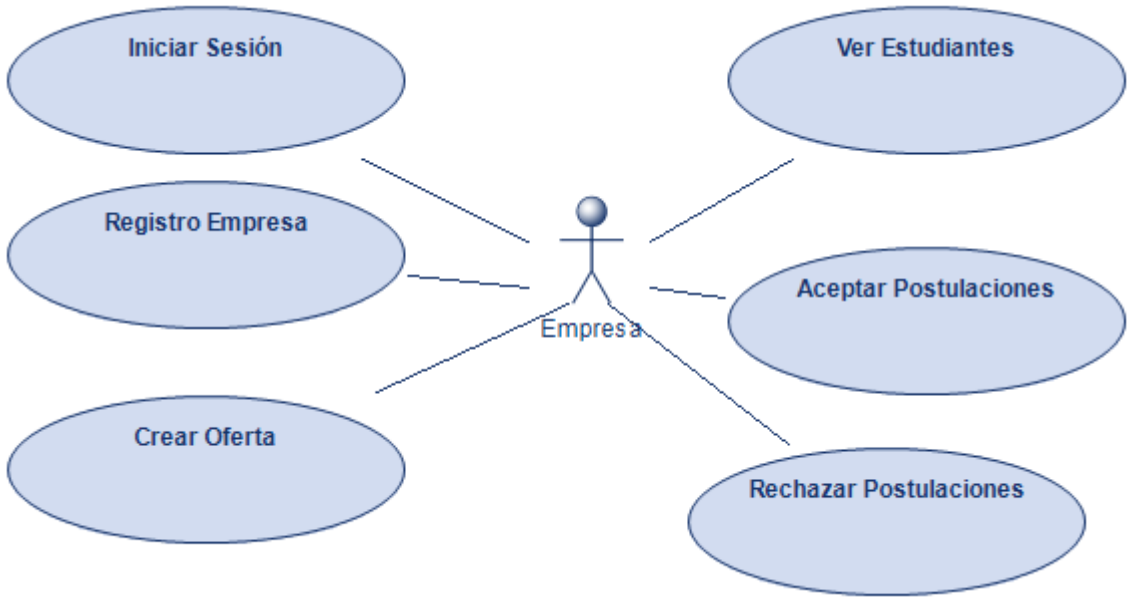


Profesores:
Los profesores pueden establecer su propia cuenta para poder también acceder a las demás funcionalidades, utilizando la previa introducción de credenciales pero esta vez en el login, de esta manera, permitiendo ver alumnos, editar sus perfiles, editar el perfil del profesor mismo, registrar alumnos en el sistema, posibilidad de poner postulaciones y asignar alumnos en las practicas.



Empresas:

Las empresas tienen que también establecer su propia cuenta para poder acceder a las funcionalidades del sistema. Después, utilizar esas mismas credenciales para crear ofertas, aceptar postulaciones, o rechazarlas y listar los estudiantes que estan postulados en la oferta.



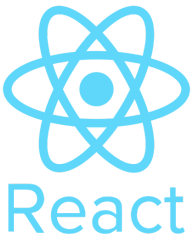
Administradores:
El administrador hereda de todos los previos roles, básicamente tiene control total del sistema, este no se puede registrar, viene establecido previamente por el sistema, no existe un registro concreto de esta entidad. Con sus credenciales, puede entrar en el sistema y realizar cualquier tipo de acción, eliminar profesor, eliminar alumno, eliminar empresa, eliminar oferta... Una funcionalidad que sí tiene es la de contestar reportes que el profesor realiza, ya sean incidencias o dudas.

Foto del diagrama annexo 5

2.2 Tecnologías

Las tecnologías que utilizaremos para el desarrollo de la aplicación son las siguientes:

Para la parte del frontend



Para el desarrollo de la parte visual utilizaremos:

HTML5: Para la maquetación de la aplicación.
CSS: Para mejorar la parte estética de la aplicación.
JavaScript: Para la parte funcional, es decir la funcionalidad de los componentes.

ReactJS en resumen para la utilización de las 3 a la vez.

Para la parte del backend

El framework principal que se va a utilizar en la aplicación web, va a ser completamente en Java, con **Spring Boot**, de manera que respetásemos el patrón MVC, la mayoría para estructurar y gestionar las peticiones HTTP mediante controladores. Recibimos las solicitudes del frontend y crearemos una API REST. Se hará uso también de la de Gmail para enviar o gestionar correos electrónicos para aquellos usuarios

Con **MySQL**, haremos la base de datos relacional para almacenar usuarios, las empresas, ofertas, y las asignaciones. Posteriormente utilizaremos el software de **MySQL Workbench** para explorar la base de datos, sus tablas con sus campos, hacer consultas de prueba, etc...

Las pruebas se harán a través de Postman o el mismo navegador, y las dependencias, librerías y otras tecnologías que se importan estarán gestionadas por **Maven**.

Otras tecnologías

Otras tecnologías que hemos utilizado a lo largo del desarrollo del proyecto es la plataforma de GitHub para poder almacenar nuestro código fuente, y también tener un control de versiones donde, en caso de incidencia podemos volver a una versión anterior de manera rápida.

También a lo largo del proyecto hemos aprendido a hacer un correcto funcionamiento de ReactJS, utilizando **NODE/JS**

3.Desarrollo

3.1 Progresos en el desarrollo del backend

Al iniciar la parte de la backend, planteamos la idea de la necesidad de este backend/API. El motivo es utilizar los servicios y operaciones que hemos diseñado para la gestión de usuarios, ofertas y seguimiento de prácticas, ya que esta plataforma no va ser dirigida a una sola empresa o institución.

Al desconocer sobre cómo se programaban las APIS en general, decidimos documentarnos y preguntar al profesorado sobre cómo funcionaban a nivel básico para así después poder tener una base mínima y empezar pequeñas pruebas. Los primeros días que estuvimos probando el framework en sí, Al principio, empezamos diseñando rutas (que en este caso son endpoints) que mapean a un método/función en concreto que básicamente realizan operaciones aritméticas sencillas más manipulación de Strings para poder después trabajar con ellas, tales como sumas, restas, entre otras operaciones básicas, y la manipulación y concatenación de literales. Eventualmente fuimos mejorando la manera en que se procesan y se reciben los datos a través del protocolo HTTP → GET, donde pasó a ser una respuesta “raw” o cruda, a un formato más claro y ordenado usando el patrón de diseño DTO para escupir los JSON y explotarlos con posterioridad.

A lo que nos referimos:

De recibir una respuesta “Hola mundo”, a un formato DTO que se puede procesar de manera mas ordenada:

```
{
        "id":1,
        "content":"Hola mundo!"
}
```

Se siguio el siguiente tutorial: https://spring.io/guides/gs/rest-service

Todo esto se fue probando con el software de **Postman** que nos permite visualizar todas las peticiones que realizamos.

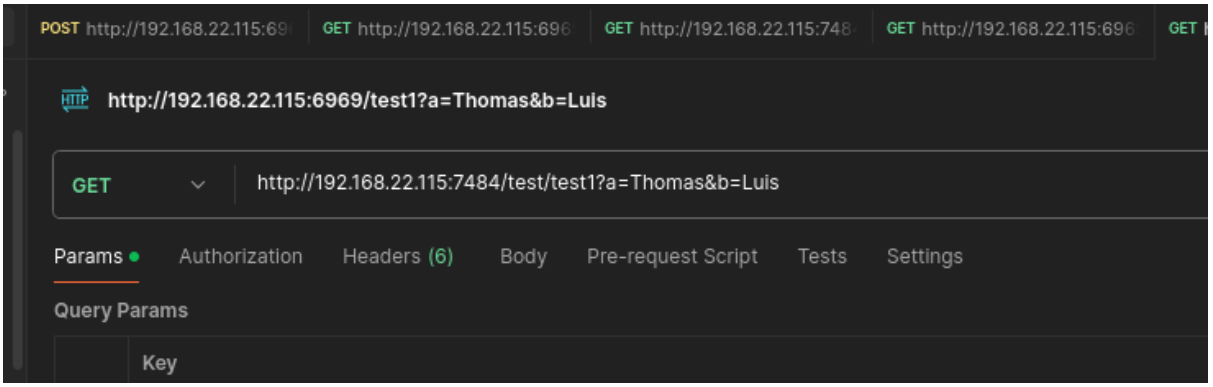
Aquí es donde visualizamos como se realiza la petición GET a traves de un link a nuestra API/Backend.

El metodo que nos devolverá un JSON:

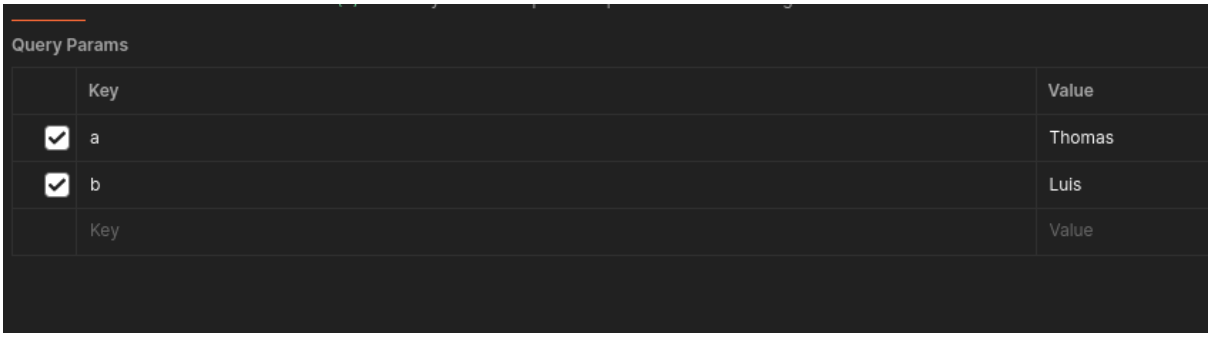
```
        DTONumber response = new DTONumber(a + b);
        return ResponseEntity.ok(response);
    }

    @GetMapping("/test1")  no usages
    public ResponseEntity<DTOString> returnConcat(@RequestParam String a, @RequestParam String b) {
        DTOString response = new DTOString(a + " " + b);
        return ResponseEntity.ok(response);
    }
}
```

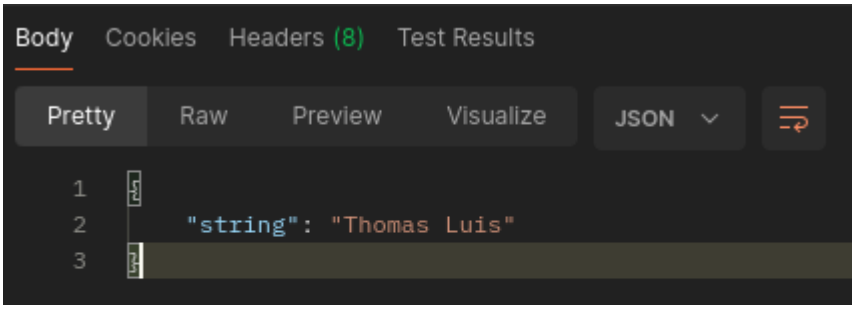
El enlace que se previsualiza para ejecutar la petición en este caso GET



Los parámetros que se piden en la funcion/metodo



Y el resultado de la ejecuion GET:



De esta manera siguiendo el tutorial, conseguimos generar pequeñas peticiones a nuestro gusto y recibirlas en formato JSON para poder explotarlos posteriormente. No se nos presentaba mucha dificultad en este punto.

Una vez se empezó a trastear con los endpoints, decidimos seguidamente empezar con persistencia de datos básica, usando el motor de base de datos **MySQL** y los contenedores **LXC** que Ubuntu nos proporciona para poder instalar el motor y trabajar sobre él, y empezar con pruebas de persistencia de datos. En Windows, las pruebas se realizaron con contenedores en **WSL** (Terminal de linux para Windows) y **MySQL Server** y el **Workbench.**

Estuvimos haciendo lecturas por la documentación oficial de Spring Boot y vimos que el framework incorpora librerías hechas por la comunidad como JPA que nos ahorran la mayoría de consultas SQL para poder trabajar con la base de datos de manera eficiente.

Para las primeras pruebas, creamos una base de datos sencilla con 2 tablas, una que almacena un ***userid*** y sus respectivos campos como el usuario y la contraseña, y otra tabla que almacena, también mediante ***userid***, el estado de las prácticas que el **usuario** está llevando a cabo, que en este caso establecemos 3:

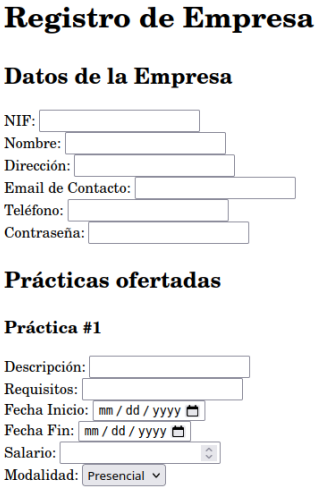
- **Pendiente, En curso** y **Completado**.

Pensamos 2 maneras por la cual pensamos en escribir estos datos, una de ellas consistía en escribir el literal dentro del campo, o la otra es establecer los 3 que definimos anteriormente como únicos que se pueden llegar a insertar en la base de datos. Estuvimos debatiendo los pros y los contras que conllevaba cada propuesta, y concluimos que la manera más óptima para escribir en los campos era con la segunda opción, de establecer un **enumerador** que solamente permite 3 tipos de datos, que son los que se mencionan arriba.

En las fases iniciales se presentaron problemas al desarrollar el endpoint encargado de registrar datos en la base de datos, debido a inconvenientes surgidos al pasar parámetros en la petición **POST**, porque se debía pasar como parámetro de una manera en específico el enumerador. Después de investigar cómo funcionaba, vimos que actuaba de la misma manera que una clase pero de una manera más sencilla, y tiene las expectativas que deseábamos, la limitación de parámetros que solo nosotros queríamos. Una vez hecho esto, y seguidamente persistiendo más en la base de datos, decidimos empezar a implementar nuestro diagrama de entidad relación para tener nuestra base de datos.

En ese momento es cuando empezamos a estructurar las entidades que tenemos como tablas en general, los **create table**...

Sus **claves primarias** y **foráneas**... Permitiéndonos así después desde los controladores persistir. Aún así, durante el proceso, vimos que JPA tiene la opción de crear las tablas automáticamente por nosotros si estructuramos bien la tabla en forma de entidad. Se crearon bocetos de los front-end, con **HTML** y **JavaScript** para poder enviar la información a nuestro backend y realizar demostraciones al profesorado, para después pasarlo al front-end oficial que usamos en nuestra web de gestión. Aqui estan unos ejemplos:



En este momento del desarrollo, tenemos 2 plataformas para testear, un frontend sencillo para realizar peticiones en la base de datos y recibir consultas de inmediato, y Postman para realizar también consultas. Seguidamente se empezó a plantear un sistema de seguridad en la API + enrutamiento la cual se consiguió implementar pero no configurar. Mas adelante, detuvimos el desarrollo de la protección de la API, y empezamos a refactorizar y empezar a usar técnicas más limpias en el desarrollo de un backend, usando propiamente los DTO sin revelar información que no debería ser mostrada como contraseñas, o información personal en general. Una vez realizados los endpoints generales, se empezaron a desarrollar el resto de controladores que nos faltaban de las demás entidades, para asi tener una API medianamente decente y limpia para poderle hacer consultas desde el frontend, y poder escribir tanto a la base de datos, etc. Después, volvimos a intentar a configurar la seguridad de la API, en este caso, con autenticación Bearer, componiendo así de esta manera un string que contiene codificados datos como el id del usuario que está realizando la petición, el correo electrónico más el rol que el usuario tiene, de esta manera podemos saber quien esta realizando peticiones al sistema, identificando tanto al rol y el usuario, y así poder organizar las peticiones, consiguiendo por fin un sistema de verificación básico con Bearer.

3.2 Progreso del desarrollo del frontend

El desarrollo del frontend de EasyFCT se centró en la creación de una interfaz de usuario intuitiva, reactiva y eficiente, que garantizara una experiencia óptima para todos los roles de usuario (administrador, profesor, alumno y empresa). La interacción fluida con la API REST del backend fue una prioridad clave, fundamentada en decisiones tecnológicas robustas y una arquitectura bien definida.

Para la implementación del frontend, se optó por **ReactJS** como librería principal para la interfaz de usuario. Esta decisión se basó en el reciente aprendizaje y la comprensión de React durante las prácticas profesionales, lo que la posicionó como la opción más idónea frente a alternativas como Ionic o Vue.js. React ofrecía un modelo de desarrollo basado en **componentes** que se percibió como superior para la modularidad y reutilización de la UI, esencial en un sistema con múltiples paneles de usuario y funcionalidades.La **facilidad de uso** y la gestión simplificada del **enrutamiento entre páginas** a través de **React Router DOM** también fueron factores determinantes.

Se descartó Ionic debido a problemas recurrentes con la interpretación de algunas de sus etiquetas, lo que dificultaba un control preciso sobre el renderizado y el estilo, mientras que la ganas de innovar y expandir las habilidades del equipo impulsó la elección de React sobre Vue.js. La lógica del frontend se implementó en **JavaScript**, priorizando la familiaridad del equipo y una curva de aprendizaje más sencilla sobre el uso de TypeScript.

La estructura del proyecto de React siguió una organización lógica y modular. El código se dividió en carpetas principales como `src/components` para componentes reutilizables como (`NavTabs`, `ButtonComp`), `src/pages` para las vistas principales de cada sección y rol.. Se gestionó el estilo utilizando **CSS** en archivos separados por componente o sección, lo que ayudó a mantener la modularidad.

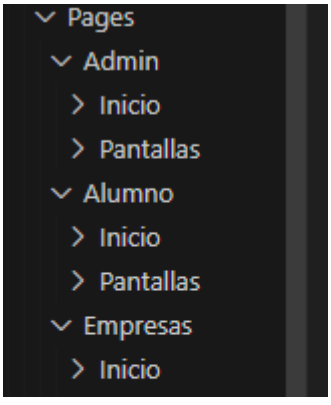


Foto ejemplo carpeta **Pages**

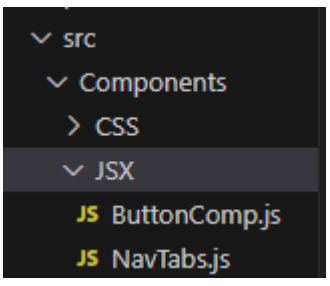


Foto ejemplo carpeta **Components**

Sin embargo, los desafíos iniciales relacionados con la repetición de nombres de clases CSS entre diferentes componentes se abordaron con una mayor rigurosidad en la nomenclatura y la encapsulación. Para la gestión del estado de la aplicación, se hizo un uso extensivo del estado local de React (useState), adecuado para la complejidad del proyecto y la gestión de datos dentro de componentes específicos. Un principio fundamental en la codificación fue el uso sistemático de la palabra clave const para la declaración de variables cuando su valor no iba a ser reasignado. Esta práctica contribuyó significativamente a la claridad y la seguridad del código, previniendo errores de reasignación no intencionados y mejorando la legibilidad del flujo de datos.

La interacción con el backend desarrollado en Spring Boot, a través de su API REST, fue un pilar fundamental. Para realizar las peticiones HTTP, se utilizó la Fetch API nativa de JavaScript, valorando su simplicidad. La gestión de estas peticiones asíncronas y su sincronización con el ciclo de vida de los componentes de React se realizó eficientemente mediante el *hook* useEffect. Este *hook* permitió encapsular las llamadas a la API para cargar datos iniciales al montar un componente o para reaccionar a cambios en las dependencias, asegurando que las peticiones se ejecutaran en el momento adecuado. Un ejemplo del uso para el inicio de sesión es el siguiente:

```
try {
  const response = await fetch(`${API_URL}/auth/userlogin`, {
    method: "POST",
    headers: {
      "Content-Type": "application/json",
    },
    body: JSON.stringify({ email, password }),
  })
```

El sistema de seguridad del frontend se implementó en coherencia con la autenticación Bearer del backend. El token JWT (JSON Web Token) se almacena en el localStorage del navegador, elección que se justificó por su persistencia, permitiendo al usuario mantener la sesión iniciada incluso después de cerrar y reabrir el navegador, lo que mejora la experiencia de usuario. El token se envía en el encabezado Authorization de cada petición HTTP, con el prefijo Bearer, permitiendo al backend identificar al usuario y su rol. La gestión básica de la expiración de la sesión se realiza al navegar a la página EleccionUsuario, donde el token en localStorage se elimina explícitamente mediante localStorage.clear(), forzando una nueva autenticación. El manejo de errores de la API se centralizó en el backend, con el frontend gestionando las respuestas para informar al usuario. La seguridad de las rutas y el acceso a funcionalidades se basa directamente en el token JWT, adaptando la interfaz a los permisos de cada rol.

El diseño del frontend se centró en una interfaz atractiva y funcional. Se integraron elementos visuales como partículas, diseños en movimiento y animaciones sutiles en botones, que contribuyen a una experiencia dinámica. Un ejemplo de ello es el *spinner* de carga implementado para el "Panel de Administración", que mejora la percepción de rendimiento. La adaptabilidad responsive se logró con @media queries en CSS, garantizando la accesibilidad en diversos dispositivos. Los paneles personalizados por rol se implementaron con componentes específicos para cada vista, priorizando la reutilización de componentes clave como NavTabs y ButtonComp para asegurar consistencia y eficiencia. El proceso inicial de diseño se llevó a cabo en Figma, comenzando con un prototipo móvil que, aunque evolucionó drásticamente, sirvió como esqueleto conceptual.

Durante el desarrollo del frontend, se enfrentaron varios desafíos. Uno de los principales fue la comprensión profunda del paradigma de componentes en React, incluyendo el paso de propiedades y la gestión del estado. Los conflictos de estilos CSS entre diferentes archivos o componentes también generaron problemas de renderizado, resueltos mediante una organización más rigurosa del código CSS. La complejidad de JavaScript y el posicionamiento preciso de elementos con CSS requirieron un considerable esfuerzo de aprendizaje y experimentación. La conexión entre el backend y el frontend presentó desafíos, especialmente en el manejo de peticiones y respuestas, lo que se abordó mediante una revisión meticulosa de la coherencia en la estructura de los datos (DTOs). Finalmente, los numerosos cambios y refactorizaciones en el frontend, a menudo para adaptar la interfaz a los datos del backend o por la búsqueda de innovación, demandaron un enfoque sistemático para mantener la armonía visual de las pantallas.

4. Conclusiones

4.1 Conclusiones generales del proyecto

EasyFCT ha demostrado ser una solución robusta y eficiente para la gestión de prácticas formativas en centros de Formación Profesional. Nuestro objetivo principal de superar las limitaciones de sistemas preexistentes, como qBid, se ha logrado con éxito al ofrecer una plataforma centralizada y accesible que aborda directamente los problemas inherentes a la gestión tradicional de FCT.

Antes de la implementación de EasyFCT, la gestión de prácticas implicaba una significativa carga administrativa y una búsqueda a menudo ineficiente para todos los implicados. Con EasyFCT, los profesores han experimentado un notable **ahorro de tiempo**, ya que la plataforma permite a los alumnos buscar y postularse directamente a las ofertas, liberándolos de una parte considerable de la gestión manual de candidaturas y permitiéndoles enfocar sus esfuerzos en el seguimiento académico y pedagógico.

Para los alumnos, la **facilidad de búsqueda y el sistema de "Match" Inteligente** representan una de las funcionalidades más innovadoras. Mediante la comparación de los lenguajes de programación declarados por el alumno en su perfil con los requisitos de las ofertas de las empresas, se calcula un porcentaje de compatibilidad. Esto asegura que los alumnos encuentren ofertas que realmente se ajustan a sus conocimientos y habilidades, reduciendo la frustración de buscar entre opciones irrelevantes y aumentando la probabilidad de encontrar una práctica adecuada que potencie su desarrollo profesional.

Asimismo, la **eficiencia para las empresas** ha mejorado sustancialmente. La capacidad de las empresas para publicar sus propias ofertas y gestionar las postulaciones directamente les otorga un control sin precedentes sobre el proceso de selección. Esto simplifica la identificación y pre-selección de alumnos que mejor se adecúan a los requisitos específicos de su organización, garantizando una mayor calidad en las asignaciones y una mejor adecuación del talento.

En resumen, EasyFCT transforma los procesos manuales y dispersos en experiencias digitales simplificadas. Donde antes los profesores dedicaban innumerables horas a buscar prácticas, los alumnos dependían de búsquedas presenciales o correos electrónicos con respuestas inciertas, y las empresas debían recurrir a centros o profesores para encontrar candidatos, ahora EasyFCT centraliza y automatiza estos procesos. Esto no solo genera un **ahorro de tiempo y una mejora sustancial en la eficiencia**, sino que también optimiza la comodidad y la transparencia para todos los usuarios involucrados en el proceso de Formación en Centros de Trabajo.

4.2 Consecución de objetivos

El desarrollo de EasyFCT ha logrado plenamente la consecución de los objetivos definidos en la fase de planificación del proyecto, impactando positivamente en los procesos de gestión de las Formaciones en Centros de Trabajo (FCT). La aplicación se ha materializado como una **herramienta multiplataforma** robusta para la gestión de FCT, accesible desde diversos dispositivos y ofreciendo una experiencia de usuario consistente, gracias a una interfaz web que se adapta a diferentes tamaños de pantalla, garantizando su usabilidad.

Se ha conseguido el **diseño y desarrollo de una interfaz de usuario moderna e intuitiva**, caracterizada por un diseño atractivo que emplea una paleta de colores neutros (gris, morado y azul), complementada con detalles visuales dinámicos como partículas en movimiento y animaciones sutiles en los botones. El posicionamiento estratégico de los elementos guía al usuario de forma natural, haciendo la navegación intuitiva y la experiencia general más agradable. Este enfoque estético y funcional facilita que los usuarios de todos los roles puedan interactuar con la plataforma de manera eficiente y sin una curva de aprendizaje pronunciada.

Asimismo, se ha logrado la **implementación de un sistema robusto para la gestión de empresas y ofertas de prácticas**. Las empresas disponen ahora de un módulo completo que les permite registrar sus datos y publicar ofertas de prácticas, especificando detalles cruciales como los lenguajes de programación requeridos y la modalidad de la práctica (presencial, remota, híbrida). Esta capacidad de personalización detallada de las ofertas no solo empodera a las empresas, sino que también alimenta directamente el sistema de "match" para los alumnos.

En línea con los objetivos, se han **desarrollado paneles de usuario personalizados para cada rol** dentro del sistema. Cada tipo de usuario —administrador, profesor, alumno y empresa— cuenta con un panel que integra funcionalidades específicas adaptadas a sus necesidades: los alumnos pueden consultar ofertas, postularse y gestionar su perfil; los profesores supervisan a sus alumnos y las asignaciones; las empresas publican ofertas y gestionan candidatos; y los administradores mantienen una visión global y un control total sobre todos los datos y procesos del sistema.

Un logro significativo es la **integración de un sistema de "Match" Inteligente para la asignación de prácticas**. Este sistema compara las habilidades técnicas del alumno, especialmente los lenguajes de programación que domina, con los requisitos de las ofertas de prácticas, calculando un porcentaje de afinidad. Esta funcionalidad no solo agiliza considerablemente el proceso de asignación, sino que también aumenta la probabilidad de éxito de la práctica al asegurar un ajuste óptimo entre el perfil del alumno y las necesidades de la empresa.

Para el seguimiento académico, se han implementado **funcionalidades de seguimiento del progreso del alumno** a través de un diario de prácticas digital. Este permite a los alumnos registrar sus actividades diarias de manera sencilla, simplemente introduciendo lo realizado en cada jornada. A pesar de su diseño directo, esta funcionalidad provee a los profesores de una herramienta estandarizada y efectiva para monitorear el avance de los alumnos, facilitando la identificación rápida de cualquier desviación o la necesidad de proporcionar apoyo adicional.

Finalmente, se ha logrado el **establecimiento de canales de comunicación eficientes** dentro de la plataforma. Se ha implementado un canal de comunicación directo entre administradores y profesores, lo que permite una coordinación efectiva en la gestión global de las FCT y asegura que la información relevante fluya adecuadamente entre estos roles clave.

4.3 Valoración de metodología y planificación

La gestión del proyecto EasyFCT se llevó a cabo utilizando una **metodología ágil adaptada, inspirada en los principios de Kanban**. Esta elección fue crucial para el éxito del proyecto, proporcionando la flexibilidad necesaria para abordar los desafíos y adaptarse a los requisitos cambiantes durante el desarrollo.

- **Implementación de Kanban**: Los principios de Kanban se aplicaron mediante la visualización del flujo de trabajo, lo que permitió al equipo mantener un control claro sobre el estado de cada tarea. Aunque no se utilizaron herramientas de gestión de proyectos específicas como Trello o Jira de forma exclusiva, se empleó **Google Drive como una herramienta colaborativa centralizada para el seguimiento de tareas**. Se organizaron las tareas en documentos que simulaban un tablero Kanban básico, permitiendo una visibilidad constante del progreso.

- **Ventajas de la metodología**: El "flujo de trabajo continuo" inherente a Kanban facilitó una respuesta rápida a las necesidades del proyecto, permitiendo integrar nuevas funcionalidades o ajustar prioridades de manera eficiente sin interrumpir el desarrollo general. Esta flexibilidad resultó invaluable en un proyecto con un ámbito potencialmente amplio y objetivos en evolución.

- **Herramientas de planificación y control de versiones**:
    - **Google Drive para seguimiento de tareas**:
        - **Ventajas**: Ofreció una plataforma de fácil acceso y colaboración en tiempo real, permitiendo a todos los miembros del equipo visualizar las tareas y su estado. Su familiaridad redujo la curva de aprendizaje.
        - **Desventajas**: La falta de funcionalidades específicas de gestión de proyectos (como asignación de roles detallada, dependencias de tareas o generación automática de informes de progreso) requirió una disciplina manual constante para mantener el orden. La visualización a gran escala del proyecto podía ser limitada en comparación con herramientas dedicadas.
    - **GitHub para control de versiones**: GitHub fue fundamental para la gestión ordenada de los cambios en el código fuente y para la colaboración efectiva entre los desarrolladores. Facilitó el trabajo simultáneo en diferentes partes del proyecto, la revisión de código y la resolución de conflictos. A pesar de los desafíos inherentes a la gestión de ramas y fusiones en proyectos colaborativos, GitHub proporcionó la infraestructura necesaria para mantener la integridad del código y un historial de desarrollo claro.
- **Lecciones aprendidas**: La adaptación de Kanban fue apropiada para la naturaleza de este proyecto. Para futuros desarrollos, mantendríamos la flexibilidad de una metodología ágil. Sin embargo, consideraríamos la implementación de una herramienta de gestión de proyectos más robusta (como Trello o Jira) para mejorar la visualización del flujo de trabajo, la asignación de tareas y el seguimiento del progreso, lo que complementaría de mejor manera los principios de Kanban y superaría las limitaciones de Google Drive. También pondríamos un mayor énfasis en la automatización de tests.

4.4 Visión de futuro

La visión de futuro de EasyFCT se centra en la expansión de su alcance y la mejora continua de la experiencia del usuario, priorizando la accesibilidad y la conexión con el entorno profesional. Las funcionalidades más importantes para el futuro son la **integración para dispositivos móviles y con LinkedIn**, ya que permitirían que la aplicación llegara a una audiencia mucho más amplia y diversa. Nuestra prioridad principal sigue siendo **impactar y apoyar a los centros educativos**, ya que son el pilar fundamental en la gestión de las prácticas formativas

1. **Ampliación de funcionalidades**:

    - **Analítica avanzada**: Se prevé la implementación de módulos de analítica avanzada que proporcionarían informes detallados sobre el rendimiento de los alumnos durante sus prácticas y la eficacia de las empresas colaboradoras. Para los **centros educativos**, esto se traduciría en la posibilidad de obtener estadísticas sobre el éxito de las asignaciones de FCT, identificar tendencias en el progreso de los alumnos y evaluar la calidad de las colaboraciones con las empresas. Para las **empresas**, se podrían generar informes sobre la efectividad de sus ofertas, el perfil de los alumnos que postulan y el rendimiento de los estudiantes en prácticas, facilitando la toma de decisiones estratégicas.

    - **Portal exclusivo para empresas con "fusión" de alumnos**: Se concibe un portal más avanzado para empresas que no solo facilitaría la gestión de ofertas, sino que también permitiría una innovadora función de "fusión" o "compartición de alumnos". Esto significaría que, en proyectos colaborativos o de formación específica, las empresas podrían "compartir" o co-gestionar el seguimiento de un alumno entre varias, exponiéndolo a diferentes entornos de aprendizaje y experiencias simultáneamente. Esto aportaría a los alumnos una experiencia de aprendizaje más rica y diversa, y a las empresas la oportunidad de participar en la formación de talento de una manera más colaborativa y estratégica.

2. **Escalabilidad y adaptación a otros sistemas educativos**: Para ampliar el alcance de EasyFCT más allá de las fronteras actuales, es fundamental desarrollar capacidades de **internacionalización y localización**. Esto implicaría que la aplicación pudiera cambiar a diferentes idiomas y se adaptara a las especificidades culturales y legales de diversas regiones. Los desafíos incluirían no solo la traducción de la interfaz, sino también la adaptación a diferentes legislaciones de prácticas formativas, formatos de documentos, normativas de privacidad de datos y estructuras educativas propias de cada país. El diseño modular actual de EasyFCT facilitaría, en cierta medida, esta adaptación.

3. **Optimización de la Experiencia de Usuario**:

    - **Versiones móviles y mejoras continuas de UI/UX**: La implementación de una **versión dedicada para iOS (y potencialmente Android)** es crucial para la accesibilidad y el engagement de los usuarios. Además, se planean mejoras continuas en la interfaz y el rendimiento de la aplicación web. Esto incluiría la optimización de los tiempos de carga de páginas, la mejora de la navegación y la implementación de nuevas visualizaciones de datos en los paneles de usuario para hacer la información más digerible y actionable.
    - **Nuevas pantallas de chat**: La incorporación de **pantallas de chat en tiempo real** para la comunicación entre administradores, empresas y profesores mejoraría significativamente la fluidez comunicativa. Aunque ya existe un canal administrador-profesor, los chats directos con empresas y entre profesores permitirían resolver dudas rápidamente, coordinar acciones y proporcionar feedback instantáneo, superando las limitaciones de la comunicación asíncrona y formal para interacciones más dinámicas.

En conclusión, EasyFCT se ha consolidado como una base sólida para la gestión de FCT. Con las mejoras y expansiones futuras planteadas, se busca no solo mantener su relevancia, sino transformarla en una herramienta líder a nivel más amplio, fomentando una colaboración más estrecha entre centros educativos, alumnos y el mundo empresarial.

5. Glossari

FCT (Formación en centros de trabajo): Periodo de prácticas formativas en empresas, obligatorio para completar los ciclos formativos de Formación Profesional (FP).

Bearer: Es un tipo de token de acceso que otorga acceso a un recurso o servicio a quien lo posee. En esencia, indica que el "portador" del token tiene derecho a acceder a los recursos asociados, sin verificación adicional.

Spring Boot: Framework de desarrollo de aplicaciones en Java que facilita la creación de aplicaciones robustas y escalables.

MySQL: Sistema de gestión de bases de datos relacional, utilizado para almacenar y organizar los datos de la aplicación.

Docker: Herramienta de contenedorización que permite empaquetar una aplicación con todas sus dependencias para su despliegue en diferentes entornos.

LXC: Herramienta de contenedorización similar a docker que permite instalar contenedores minimalistas de cualquier SO, para poder desplegar .

WSL: Es una característica de Windows que permite ejecutar un entorno GNU/Linux directamente sobre Windows, sin necesidad de una máquina virtual. Es especialmente útil para desarrolladores que desean utilizar herramientas de Linux en un sistema Windows.

HTTPS: Es una versión segura del protocolo HTTP. Utiliza cifrado TLS (anteriormente SSL) para proteger la comunicación entre el navegador del usuario y el servidor web, garantizando la confidencialidad e integridad de los datos transmitidos.

HOOK: Herramienta de React.

6. Bibliografía

1. Spring. “Spring Boot Documentation.” Disponible en: https://spring.io/projects/spring-boot. Este framework es el que nos permite hacer toda lógica de negocio.
2. MySQL. “MySQL Documentation.” . Disponible en: https://dev.mysql.com/doc/. Motor de la base de datos.
3. Bearer-auth: https://keepcoding.io/blog/token-bearer-guia-para-desarrolladores/. (Autentificación de token)
4. BCrypt API: https://bcrypt-generator.com/ (Encriptar las contraseñas)
5. Spring Initializr: https://start.spring.io/ (Web para inicializar el proyecto de SpringBoot)
6. Ngrok: https://ngrok.com/ (Para lanzar temporalmente el backend a nivel publico)
7. Stack ovefrlow: https://stackoverflow.com/questions (Para resolver preguntas de programacion)
8. Figma: https://www.figma.com/ (Para el diseño de la web)
9. JWT tutorial: https://www.youtube.com/watch?v=oeni_9g7too (Tutorial para poder entender de que van los JSON Web Tokens para poder utilizarlos posteriormente)
10. Github: https://github.com/ (Almacenamiento del codigo fuente y control de versiones)

Documento con tareas: https://docs.google.com/document/d/1IEQf0d5pBnAwO8fRxOwmEB0Q_xEg8MVQewM0lIrXzK4/edit?tab=t.0

Agradecimientos

Federico Jerez; Gracias a su conocimiento en persistencia de datos SQL, pudimos establecer una base de datos en el primer curso. (Uso de postgres/mysql).

Fernando Porrino; Profesor de programación de primer grado, sin sus fundamentos, muchas de las cosas que hacemos en este segundo de superior, nos hubiera dificultado mucho el desarrollo de este proyecto en general

Yago Morales Ares; Tutor de clase que nos ha ayudado muchísimo en la parte de seguridad del backend y desarrollo del frontal, gracias a tí, hemos aprendido muchísimo sobre el desarrollo FullStack.

David Thomas;

Luis Elia; Gracias a ti, hemos tenido la suerte de estar guiados en esta memoria, con sugerencias, correcciones, y revisiones del trabajo en clase tanto con la parte de servidor y web.

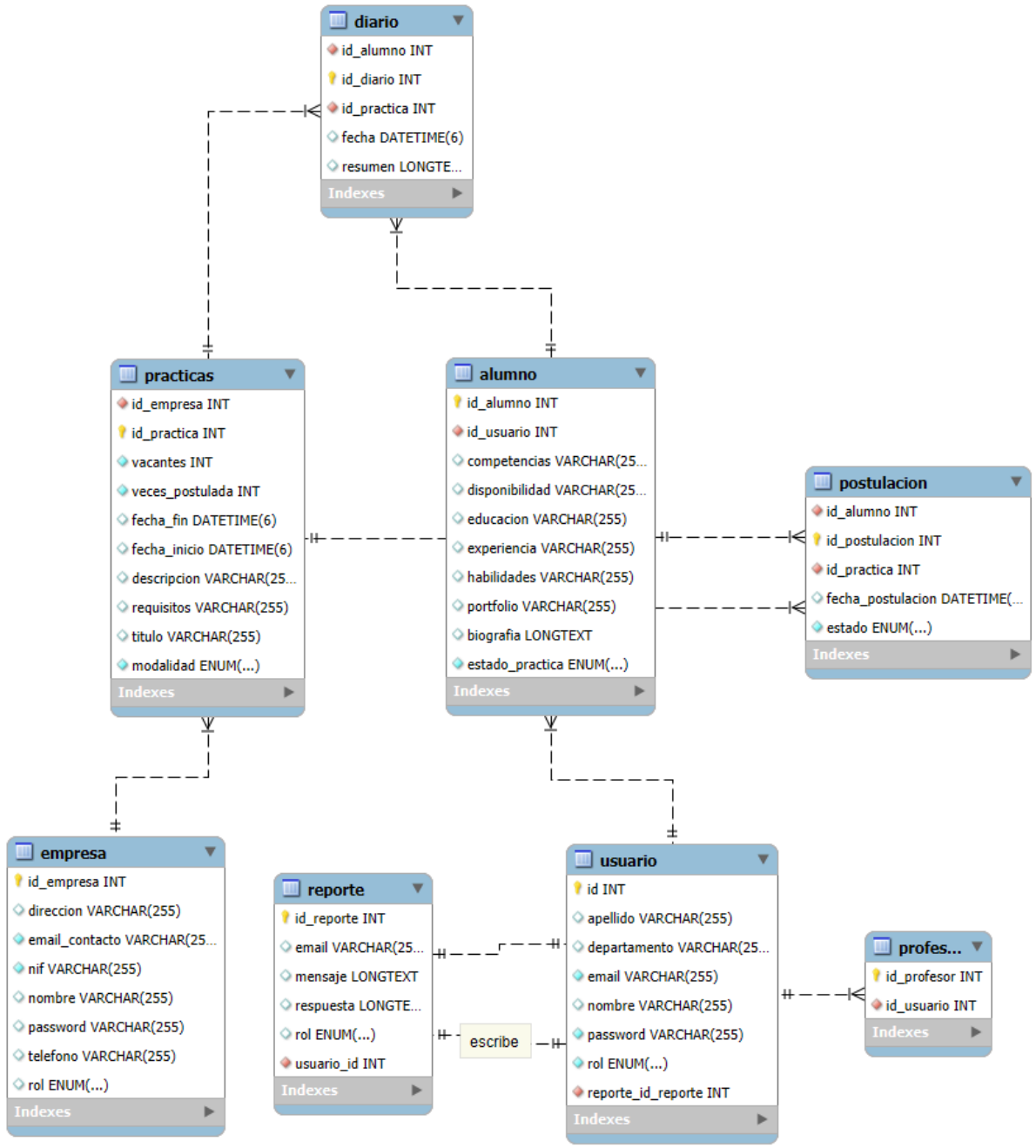
Jordi Hernández; Por las introducciones a las clases de Spring y uso de la librería JPA y persistencia de datos.

Y alumnos de clase;

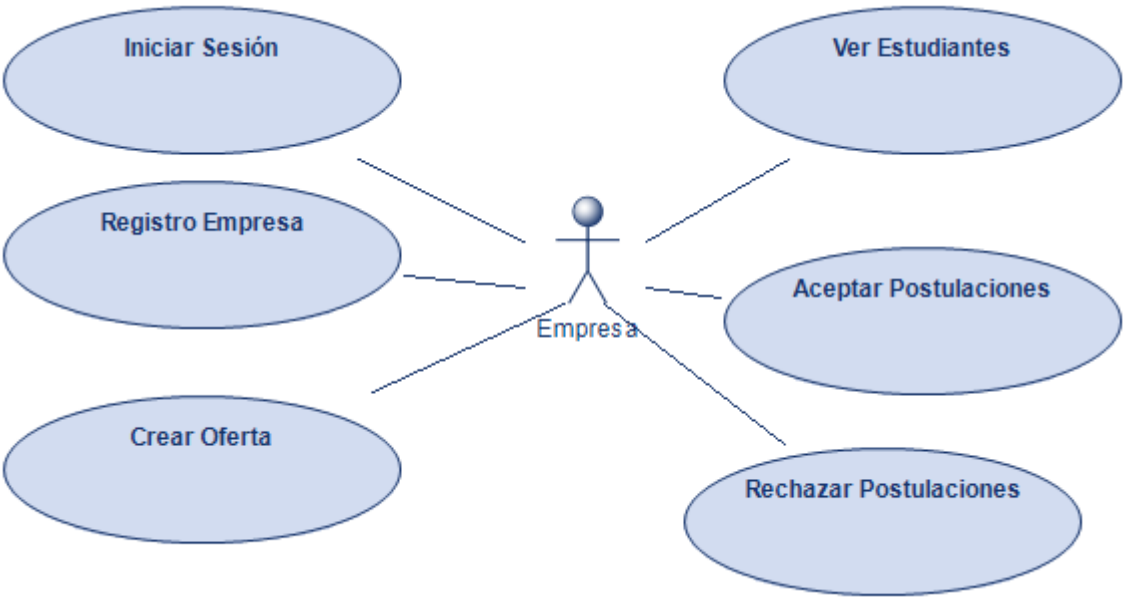
- Benjamin Perez
- Nacho Herrero
- Eloy
- Isma
- Ibra

7. Annexos

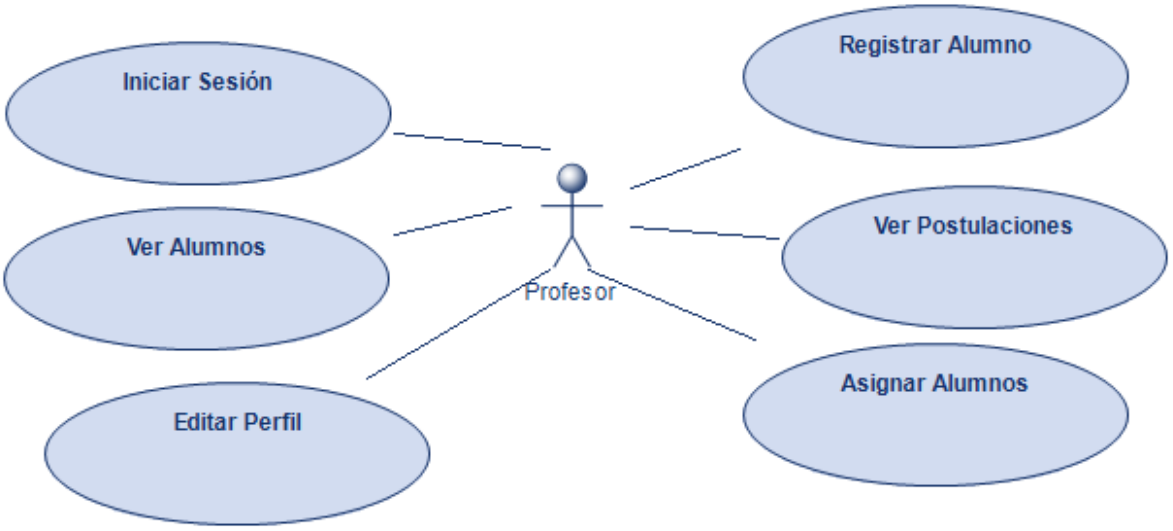
7.1 Diagrama Entidad-Relación



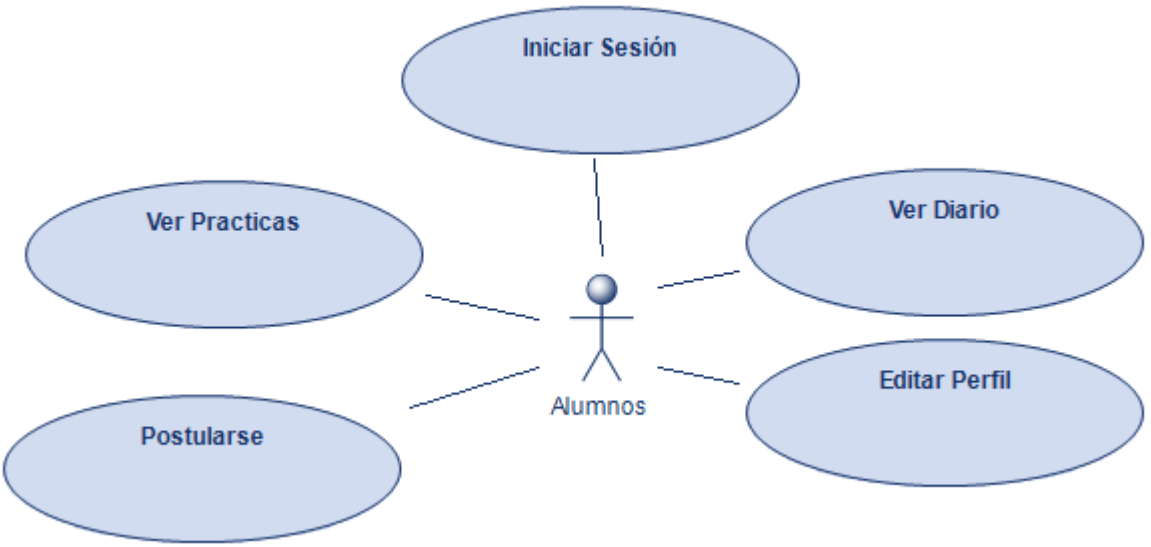
Annexo 1



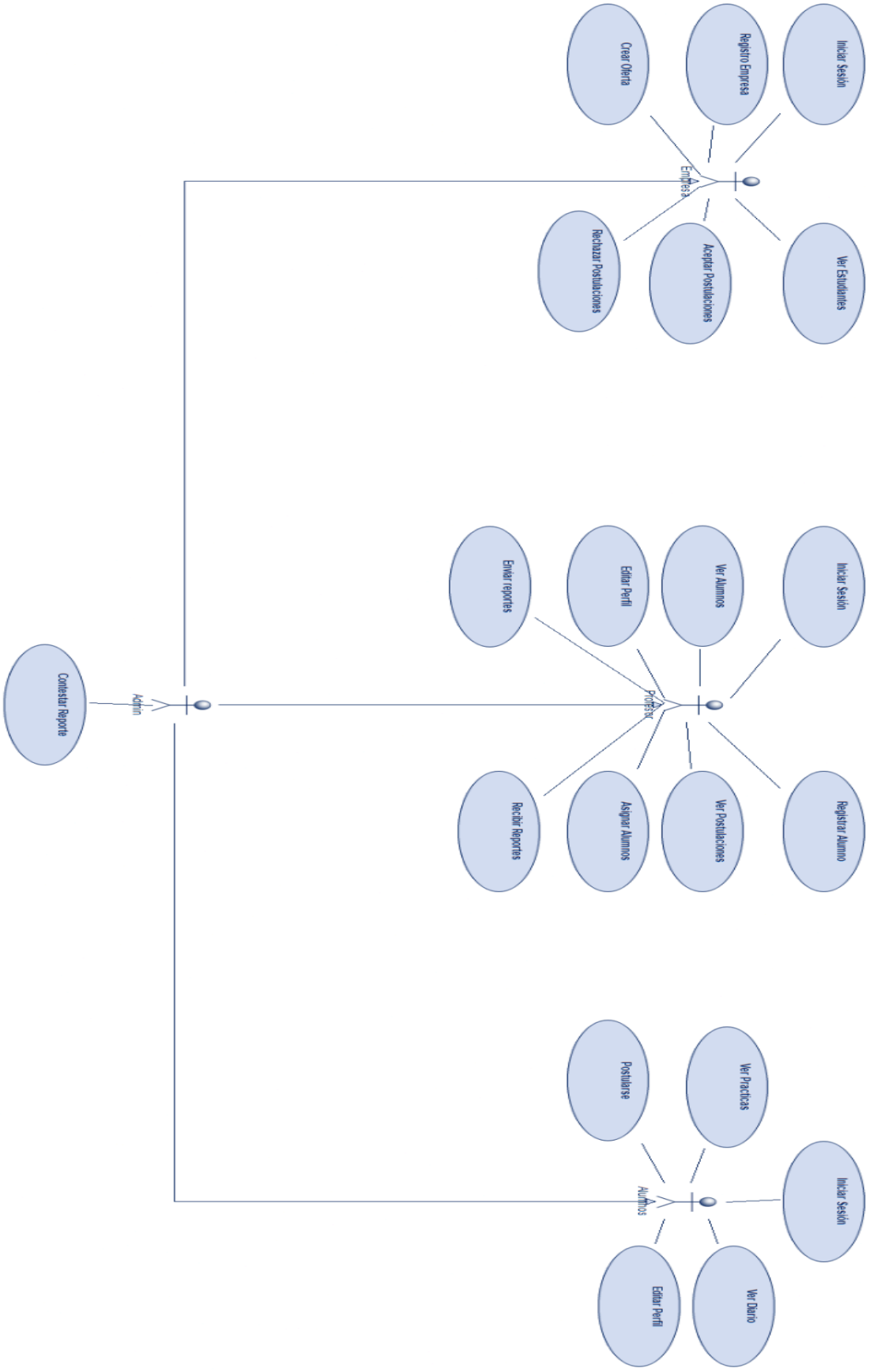
Annexo 2

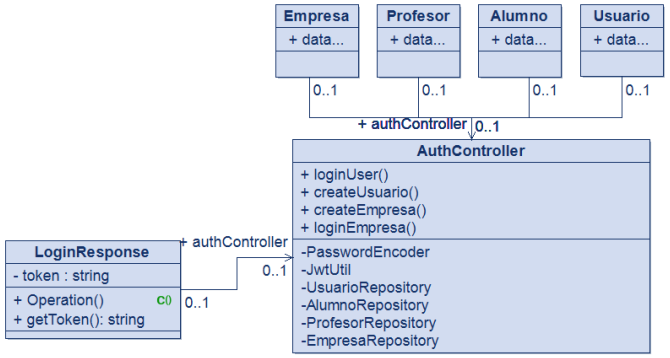


Annexo 3

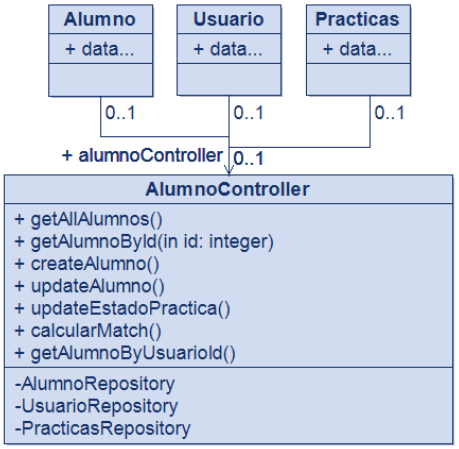

Iniciar Sesión
Ver Practicas
Ver Diario
Alumnos
Editar Perfil
Postularse

Annexo 4

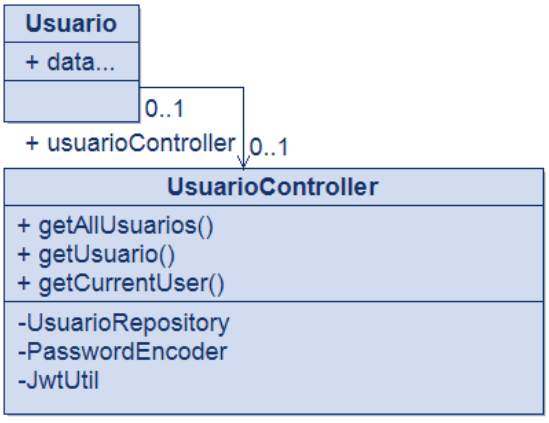

Crear Oferta
Registro Empresa
Iniciar Sesión
Empresa
Rechazar Postulaciones
Aceptar Postulaciones
Ver Estudiantes
Enviar reportes
Editar Perfil
Ver Alumnos
Iniciar Sesión
Contestar Reporte
Admin
Profesor
Recibir Reportes
Asignar Alumnos
Ver Postulaciones
Registrar Alumno
Postularse
Ver Practicas
Alumnos
Iniciar Sesión
Editar Perfil
Ver Diario



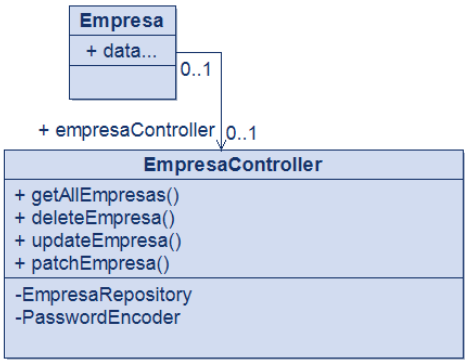
Clase controlador de autentificación de usuarios.



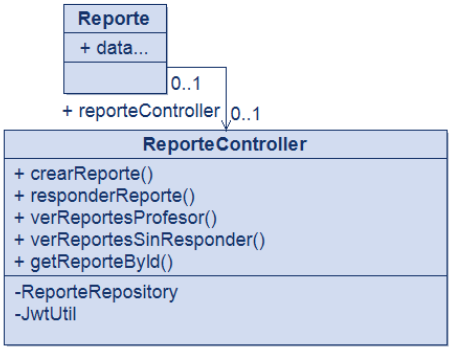
Clase controlador de alumno, responsable de obtener y actualizar alumnos.



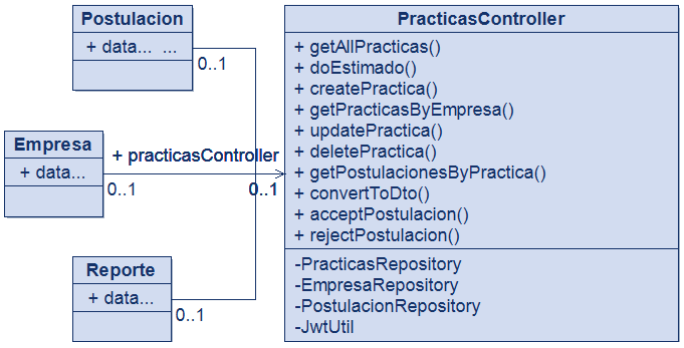
Clase controlador de usuario, responsable de obtener los usuarios, y el logeado actual



Clase controladora de establecer y crear empreas



Clase responsable de establecer un pequeño chat entre pro



Clase controladora de establecer practicas a traves de la empresa

Clase inicializadora del framework de SpringBoot