PROYECTO FIN DE GRADO

TÍTULO: Plataforma web de becas y ofertas de trabajo para estudiantes y empresas

AUTOR/A: Adrián Barco Barona

TITULACIÓN: Grado en Ingeniería Telemática

TUTOR/A: Pablo Ramírez Ledesma

DEPARTAMENTO: Departamento de Telemática y Electrónica (DTE)

VºBº TUTOR/A

Miembros del Tribunal Calificador:

PRESIDENTE/A: Rafael Delgado López

TUTOR/A: Pablo Ramírez Ledesma

SECRETARIO/A: Aurelio Bergés García

Fecha de lectura:

Calificación:

El Secretario/La Secretaria,

Resumen

Este proyecto aborda la creación de una plataforma web desarrollada con Java, utilizando las tecnologías de Spring-Boot y Vaadin. Su principal objetivo es gestionar becas y ofertas de trabajo o prácticas, para facilitar la interacción entre estudiantes y empresas. Además, se ha llevado a cabo un despliegue mediante contenedores de software para ofrecer una mayor estabilidad y escalabilidad del sistema.

Respecto a Vaadin, merece la pena destacar que se trata de un entorno de trabajo poco conocido e innovador, que ayuda al desarrollador a crear una interfaz de usuario práctica y eficiente mediante el uso de una gran variedad de componentes de usuario. De este modo, uno de los objetivos tecnológicos de este proyecto es aplicar dicha tecnología en un caso real para evaluar sus resultados.

De igual forma, el resto de las tecnologías utilizadas comprenden un conjunto de herramientas ampliamente aceptadas en el ámbito del desarrollo web. Esto incluye a Maria DB como motor de base de datos, que proporciona una estructura sólida y fiable para el almacenamiento de los datos, y a *Git* como software de controlador de versiones, que permite gestionar la colaboración y el seguimiento de cambios en el código.

Por otra parte, hay una serie de consideraciones económicas y normativas que han condicionado el proyecto. Relativo al plano económico, se ha priorizado el uso de recursos de código abierto y tecnologías gratuitas, lo que ha permitido minimizar costes asociados a licencias y herramientas. Este enfoque ha sido fundamental para garantizar la viabilidad financiera del proyecto sin comprometer su calidad.

Por otro lado, en cuanto al factor legislativo, se ha asegurado el cumplimiento de las leyes de protección de datos y privacidad, velando por la seguridad y confidencialidad de la información de los usuarios.

De este modo, en el presente documento se puede encontrar toda la información relativa a la comprensión del desarrollo del proyecto y cómo ha sido definida, construida y probada la aplicación. Al final de la memoria se ofrece un apartado de conclusiones y un anexo donde se completan los manuales de instalación y de uso de la aplicación.

Abstract

This project deals with the creation of a web platform developed with Java, using Spring-Boot and Vaadin as main technologies. Its principal objective is to manage scholarships and job or internship offers, to facilitate interaction between students and companies. In addition, it has been deployed using software containers to offer a greater stability and scalability on the system.

In relation to Vaadin, it is worth noting that it is an unknown and innovative framework, which helps the developer to create a practical and efficient user interface through the use of a wide variety of user components. Therefore, one of the technological objectives of this project is to apply this technology in a real-world case in order to evaluate its results.

By the same token, the other technologies used make up a set of tools that are widely accepted in the area of web development. This includes MariaDB as a database engine, which provides a robust and reliable structure for data storage, and Git as version control software, which allows for collaboration and code change tracking.

On the other hand, there are plenty of economic and regulatory considerations that have conditioned the project. On the economic side, priority has been given to the use of open-source resources and free technologies, which has allowed minimizing costs associated with licenses and tools. This approach has been fundamental to guarantee the financial viability of the project without compromising its quality.

Furthermore, regarding to the legislative factor, compliance with the data protection and privacy laws has been verified, ensuring the security and confidentiality of user information.

Thus, this document contains all the information related to the understanding of the development of the project and how the application has been defined, built and tested. At the end of the report there is a section of conclusions and an annex where the installation and user manuals of the application are completed.

Agradecimientos

A mi familia, que siempre ha estado detrás de mí, apoyándome en cada paso del camino.

A Isabel, que ha sido un apoyo constante en momentos de incertidumbre.

A mis profesores y mentores, en especial a mi tutor, que han sido mis guías y maestros en mi formación académica.

A todos aquellos que dudaron de mí, este proyecto es para ustedes.

Aquí estoy, lo he logrado.

Índice

Índice de imágenes

Índice de tablas

Lista de acrónimos

AOP: Programación Orientada a Aspectos (*Aspect Oriented Programming)*.

CRUD: Crear, Leer, Actualizar y Borrar (*Create, Read, Update and Delete*).

CSS: Hojas de Estilo en Cascada (*Cascading Style Sheets*).

HTML: Lenguaje de Marcas de Hipertexto (*Hypertext Markup Language*).

IDE: Entorno de Desarrollo Integrado (*Integrated Development Environment*).

JAR: Archivo Java (*Java Archive*).

JDBC: Conectividad a Bases de Datos de Java (*Java Database Connectivity*).

JSON: Notación de Objetos de JavaScript (*JavaScript Object Notation*).

MVC: Modelo – Vista – Controlador (*Model – View - Controller*).

ORM: Mapeo Relacional de Objectos (*Object – Relational Mapping*).

POM: Modelo de Objecto para un Proyecto (*Project Object Model*).

REST: Transferencia de Estado Representacional (*Representational State Transfer*).

SQL: Lenguaje de Consulta Estructurada (*Structured Query Language*).

UML: Lenguaje de Modelado Unificado (*Unified Modeling Language*).

URI: Identificador Uniforme de Recursos (*Uniform Resource Identifier*).

XML: Lenguaje de Marcado eXtensible (*Extensible Markup Language*).

YAML: YAML No Es un Lenguaje de Marcado (*YAML Ain't Markup Language*).

1. Introducción

1.1 Motivación

Hoy en día, es un desafío desarrollar una aplicación web práctica y útil mediante un único lenguaje de programación, tanto en el desarrollo de la parte de la lógica de negocio (*backend*), como para la interfaz visible (*frontend*), sin necesidad de hacer uso de lenguajes específicos como HTML, JavaScript o CSS para este último.

Sin embargo, el entorno Vaadin es una nueva solución que permite utilizar Java para el *frontend* y realizar el desarrollo de una aplicación Web utilizando un único lenguaje. Por esta razón, uno de los objetivos de este proyecto es evaluar dicha tecnología en un caso de uso práctico desarrollando una aplicación real de utilidad.

Además, pasar del ámbito académico al laboral o profesional se presenta como un gran reto para infinidad de estudiantes. Esto se debe a factores como la escasa o nula experiencia laboral o la elevada competencia por la alta cualificación que requiere el mercado. Así pues, se realizará una aplicación que permita lidiar con los retos relativos a la inserción laboral de estudiantes y jóvenes profesionales, gestionando becas y ofertas de trabajo, aplicando esta nueva solución tecnológica.

1.2 Descripción general de la aplicación

En el contexto actual del desarrollo de software, la creación de aplicaciones web se ha convertido en una práctica muy normalizada y versátil, disponiendo de una amplia variedad de entornos de trabajo o *frameworks* y lenguajes de programación para llevar a cabo este tipo de proyectos. Dentro de lo mencionado anteriormente, destaca Spring-Boot como una tecnología especialmente útil y eficaz para ello.

Fundamentalmente, Spring-Boot ofrece múltiples ventajas relevantes para la construcción de código. Aparte de facilitar esto último, también simplifica en gran medida tareas relacionadas con la infraestructura y el despliegue de la aplicación. Así, permite que los programadores centren su atención en la esencia del desarrollo de la aplicación, liberándose relativamente los aspectos que han sido comentados con anterioridad.

De este modo, lo que se describe en este documento es el desarrollo de una aplicación web con el objetivo de facilitar la inserción laboral de estudiantes. Lo que se les ofrece es un entorno sencillo y accesible para la parte del entorno gráfico y para la gestión de becas y ofertas de trabajo por parte de los propios usuarios. En este sentido, la plataforma simplifica el proceso tanto para los alumnos como para las empresas interesadas, fomentando así una interacción más fluida y productiva entre ambos.

En consecuencia, se ha optado por tecnologías ya estandarizadas para el desarrollo de código, como Spring-Boot, ya mencionada previamente, y Vaadin, aprovechando las ventajas y funcionalidades que

ofrecen estas herramientas en términos de eficiencia y flexibilidad. Por otro lado, se emplea *Docker* para el despliegue de la aplicación mediante contenedores de software, lo que garantiza un entorno de ejecución homogéneo y fácilmente replicable en otro entorno diferente.

Asimismo, es importante destacar que la aplicación resultante es completamente independiente de la plataforma de uso, lo que significa que puede ser accesible a través de sistemas operativos como Windows, Linux, iOS, etc. Por tal motivo, esta característica contribuye notablemente a su alcance y funcionalidad, permitiendo que una gran cantidad de usuarios puedan beneficiarse de sus servicios.

En resumen, este proyecto tiene como objetivo prioritario el poder poner en funcionamiento una aplicación web creativa y accesible que pueda facilitar la iniciación laboral, sobre todo, a estudiantes que vayan a concluir sus estudios o los hayan finalizado recientemente. Ahora bien, esto se alcanza aprovechando tecnologías avanzadas en el ámbito del desarrollo de software y garantizando su accesibilidad y funcionalidad en diversas plataformas de uso.

1.3 Objetivos

Esta aplicación web es una plataforma en la que las empresas pueden cargar tanto ofertas de becas como ofertas de trabajo para estudiantes, y entrar en contacto con los alumnos que estén interesados y apliquen a ellas. De esta forma, se facilita y se agiliza el contacto entre empresa y estudiante. Los principales objetivos de este proyecto se basan en el empleo de tecnologías que faciliten la tarea al programador a través de componentes de código abierto (también denominados *open source*) y seguros, de una forma flexible y eficiente, con los siguientes objetivos:

- Ayudar a la inserción laboral de estudiantes y culminar su formación académica.
- Ayudar a las empresas a explotar el talento joven.
- Desplegar una aplicación de forma rápida y sin consumir muchos recursos.
- Desarrollar una interfaz gráfica original y de fácil acceso, enteramente con código Java.
- Construir una aplicación segura, en cuanto a protección de datos y autenticación.

1.4 Tecnología usada

Durante el desarrollo de las diferentes partes y artefactos del sistema se han utilizado los siguientes medios de trabajo:

- IDE de desarrollo Eclipse para la construcción de código.
- Herramienta de modelado Visual Paradigm para crear modelos UML.
- Microsoft Word para escribir la memoria.
- Git para llevar un control de versiones.
- Docker como herramienta de despliegue de contenedores de software.
- DB Adminer, como herramienta para administrar el contenido de la base de datos.
- Postman, para realizar peticiones de prueba al API REST.

De esta forma, el sistema final está implementado usando la siguiente tecnología:

- Java
- JSON
- Maven
- Spring-Boot
- REST
- Vaadin
- SQL
- MariaDB

2. Marco tecnológico

2.1 Introducción

El desarrollo web ha estado evolucionando constantemente, hasta llegar al punto de convertirse, actualmente, en una labor fundamental para la creación de soluciones tecnológicas, las cuales están altamente demandadas en el mercado. Sin embargo, esta actividad, aparte de comprender la creación y el mantenimiento de los sitios web, también incluye la introducción de funcionalidades innovadoras o la optimización del rendimiento. En consecuencia, este proceso implica la colaboración entre varios desarrolladores, quienes suelen estar especializados en *back* o en *front*, requiriendo una gran variedad de tecnologías y herramientas para poder completar los requisitos del proyecto solicitado para cada caso.

En primer lugar, hay que partir del concepto de aplicación web, que, en definitiva, es un software que se ejecuta en el navegador web, permitiendo acceder a funcionalidades complejas sin la necesidad de instalarlo o configurarlo. Este software, además, se almacena en grandes servidores de internet de forma permanente, enviando a los dispositivos los datos que se requieren en ese momento específico, dejando una copia temporal en el propio dispositivo.

De acuerdo con todo ello, es importante señalar que la aplicación web propuesta cumple con una función primordial y bastante provechosa para el ámbito académico y laboral. A causa de la progresiva transformación digital, no cesa el surgimiento de nuevas profesiones y demanda de oportunidades profesionales por parte de los estudiantes, por lo que las empresas tienen la necesidad de encontrar ese talento joven. Por esta razón, es necesaria una plataforma eficiente que faciliten la conexión entre ambas partes.

2.2 Estado actual

En la actualidad, existen diversas plataformas y sistemas web que ofrecen una gran variedad de servicios. Sin embargo, muchos de ellos presentan limitaciones en términos de manejo, escalabilidad, seguridad o compatibilidad con otras herramientas. De esta manera, la tendencia actual se basa en la creación de aplicaciones web modernas que brinden una experiencia de usuario fluida, sean fáciles de mantener y, sobre todo, que sean escalables, para poder satisfacer las necesidades comerciales.

No obstante, por norma general, todas estas aplicaciones suelen tener un *backend* y un *frontend* claramente diferenciado, cada uno desarrollado con tecnologías notablemente reconocidas dentro de cada contexto:

- Como ***backend***, se considera la parte lógica del sistema, que recibe y devuelve los datos procesados por las aplicaciones, garantizando la seguridad y el funcionamiento eficiente de éstas últimas. Aquí se incluyen gran variedad de lenguajes de programación como base, tales como Java, Python o JavaScript, entre otros, y diversos frameworks como Django, Spring o Express.js, que utilizan los lenguajes anteriores, respectivamente.
- El ***frontend*** hace referencia a la parte encargada de interactuar con el usuario, la aplicación cliente que se ejecuta en el navegador. HTML, CSS o JavaScript son algunos de los lenguajes más utilizados para este propósito, y en donde no suele contemplarse Java. De esta forma, Vaadin, que no es tan conocido como lo anteriores, va a permitir crear un *frontend* con el mismo lenguaje de programación que el *backend*, unificando un mismo lenguaje para toda la aplicación.

En este contexto, Vaadin ofrece una alternativa innovadora al tradicional enfoque de separación entre frontend y backend, al utilizar el mismo lenguaje, lo que facilita la integración y la comunicación entre los dos componentes. Esto permite crear aplicaciones web modernas, seguras y fáciles de mantener, y que brinden una experiencia de usuario fluida y adaptable. Además, Vaadin proporciona una amplia variedad de componentes y herramientas para facilitar el desarrollo de aplicaciones web, incluyendo soporte para Progressive Web Apps y extensiones para incorporar componentes más complejos.

Por último, cabe destacar algunos de los sistemas ya existentes y que son relevantes en este campo, como las plataformas LinkedIn, Indeed y algunos portales universitarios que ofrecen información sobre becas y oportunidades laborales. A pesar de ello, estos sistemas suelen carecer de funcionalidades específicas para la gestión de becas y ofertas de trabajo dirigidas específicamente a estudiantes y empresas.

2.3 Vaadin: innovación en la integración de Java y web

Vaadin Flow es el entorno utilizado para construir la interfaz de usuario. Desde la perspectiva más general, proporciona una serie de componentes y herramientas con el fin de simplificar el proceso de

creación de una interfaz gráfica sencilla y confiable. A esto hay que añadir que se desarrolla con Java y, como se ha señalado anteriormente, se integra adecuadamente con Spring. Además, ofrece múltiples posibilidades en el diseño mediante elementos como formularios, cajas de texto, botones, etc., fácilmente instanciables y personalizables, lo que será descrito posteriormente.

Lo que propone Vaadin es que, en lugar de hacer uso de HTML, CSS o JavaScript, la interfaz de usuario se construya a partir de componentes UI en Java. Sin embargo, HTML, CSS y JavaScript no se suprimen, sino que se abstraen en una API de Java, para poder seguir utilizando a este último como lenguaje principal. Esta API favorece la sincronización entre los objetos Java del lado del servidor y los elementos HTML del lado del cliente. Así, Vaadin Flow puede gestionar los elementos HTML en el navegador, controlando los atributos, las propiedades y los elementos secundarios de estos últimos, además de realizar invocaciones personalizadas de JavaScript.

Así, las vistas de la aplicación y sus componentes serán los responsables de mostrar y permitir la entrada de datos a la propia aplicación web. Como puede verse en el esquema de la Ilustración 1, los datos suelen almacenarse en un servicio de *backend*, como una base de datos, a la que puede accederse por un API REST.

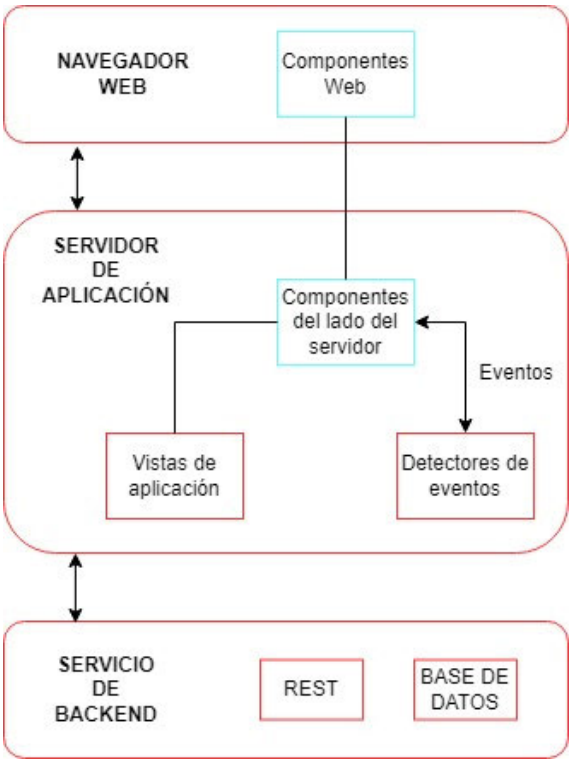

Ilustración 1. Componentes de una aplicación Vaadin Flow

De este modo, para permitir la interacción de los usuarios con los componentes Java, Vaadin Flow gestiona la retransmisión de la actividad del usuario a la aplicación del lado del servidor, que la maneja mediante detectores de eventos. Esto quiere decir que la interacción de un usuario con la interfaz en el

navegador provoca eventos, los cuales se pasan al lado del servidor, donde se manejan mediante referencias a métodos privados, expresiones lambda o clases regulares.

Por otro lado, cada vista tiene una ruta única a través de la cual se puede acceder a ella. De esta forma, el usuario puede interactuar con la aplicación y navegar por las vistas disponibles de dos formas distintas: propiciando eventos referentes a los componentes web de la vista o ingresando en claro la URL específica de una de las visitas para navegar directamente ella. Al generar eventos en la interfaz, como el que genera un usuario al pulsar un botón, se procede a modificar la URL del navegador de forma dinámica o a vincular una URL determinada con ese componente, para que el usuario vea reflejado en la pantalla la navegación entre las vistas.

En cuanto a componentes web, Vaadin incluye una colección diversa y elemental para poder crear la interfaz gráfica. Todos los componentes suelen estar albergados en layouts, que son contenedores visuales que pueden alojar una gran cantidad de componentes de forma organizada. Por tanto, estos elementos primero se instancian y se configuran, y luego se añaden a un layout ya existente. Como layouts imprescindibles se presentan el Layout Vertical (*VerticalLayout*), que dispone los componentes uno detrás de otro verticalmente, y el Layout Horizontal (*HorizontalLayout*), que lo hace análogamente, pero de forma horizontal. Y como componentes habituales destacan los botones (*Button*), las cajas de texto (*TextField*), la barra de navegación (*SideNav*) o las notificaciones (*Notification*).

Ilustración 2. Ejemplo de componente web (Button)

Estos componentes ofrecen métodos para personalizar todos sus atributos o gestionar los posibles eventos que ofrecen, permitiendo establecerlos de una forma sencilla y transparente.

Otro aspecto importante es la apariencia predeterminada que los componentes de Vaadin proporcionan. Esto se debe a que incluye un tema por defecto denominado Lumo, que se basa en un conjunto de propiedades de estilo personalizables y predefinidas, así como variantes oscuras y compactas ya integradas. En consecuencia, la apariencia de todos los componentes de Vaadin se puede personalizar de manera consistente y práctica en toda la aplicación, proporcionando sus propios valores a estas propiedades de estilo. Adicionalmente, el tema Lumo también incluye un conjunto completo de clases de utilidad CSS que se pueden aplicar a elementos HTML y componentes de diseño a través de Java, para cambiar su estilo sin escribir ningún CSS propio.

Además, como Vaadin Flow es un marco del lado del servidor, donde todo el estado de la aplicación, la lógica de negocio y la lógica de la interfaz de usuario permanecen en el servidor, la aplicación nunca va a exponer sus componentes internos al navegador, donde una atacante podría aprovechar las vulnerabilidades. Esto añade una capa de seguridad implícita, aunque debe ir acompañada de buenas prácticas para evitar otras vulnerabilidades, con el fin último de garantizar la seguridad máxima posible.

En definitiva, construir una aplicación con Vaadin permite a un solo grupo de programadores crear todo el entorno, desde el acceso a los datos hasta la interfaz de usuario. Esta independencia reduce significativamente las demoras asociadas con la coordinación con otros equipos expertos, acelerando su puesta en marcha final.

2.4 Otras tecnologías utilizadas en el proyecto

En este proyecto concreto, se ha tratado de emplear una serie de tecnologías innovadoras y eficientes en su conjunto, con el fin de combinarse con Vaadin y facilitar tanto la parte del desarrollo como la parte de interacción del usuario. En consecuencia, la base de todo el sistema se va a fundamentar en:

- **Java:** es el lenguaje de programación que se conforma como la base de todo el proyecto. Fue creado por *Sun Microsystems* en 1995 y, desde entonces, no ha parado de evolucionar hasta convertirse en una parte importante de muchos servicios y aplicaciones.
- **XML:** es un metalenguaje de marcado que permite definir y almacenar datos en un formato legible, con el fin de facilitar su procesado.
- **JSON:** es un formato ligero y sencillo para el intercambio de datos, independientemente del lenguaje de programación, y muy cómodo de interpretar. Estructura los objetos en pares clave – valor, de modo que posibilita la serialización y deserialización de los datos.
- **Spring-Boot:** esta tecnología parte de Spring como entorno de trabajo especializado en el desarrollo de aplicaciones web. Asimismo, este *framework* está constituido por una serie de módulos o componentes, algunos de ellos opcionales, que permiten al desarrollador crear aplicaciones robustas y escalables:
    - **Contenedor Central (*Core Container*):** es un elemento clave, que atiende a la responsabilidad de crear y administrar los objetos y sus dependencias, y de inyectar estas dependencias por medio de anotaciones y configuraciones XML.
    - **Componente Web (*Web Component*):** facilita la implementación de patrones de arquitectura como MVC, la creación de servicios web *RESTful* y la integración de una gran cantidad de estándares web, al proporcionar las características y funcionalidades necesarias para manejar controladores web.
    - **Acceso a datos (*Data Access*):** facilita el acceso a los datos haciendo uso de elementos como ORMs, capas de abstracción sobre JDBC o manejo de transacciones. Esto permite interactuar de manera coherente con una gran variedad de tipos de fuentes de datos.
    - **AOP:** se enfoca en la gestión de transacciones y en los comportamientos que afectan a múltiples puntos de ejecución en una aplicación. De esta manera, mediante anotaciones, Spring desarrolla aspectos de sus aplicaciones de forma sencilla.
    - **Instrumentación (*Instrumentation*):** hace referencia a la capacidad de modificar el comportamiento de las aplicaciones en tiempo de ejecución, para agregar

funcionalidades adicionales, facilitar la depuración y la detección de errores y mejorar el rendimiento.

- **Test*:*** contiene su propio entorno de pruebas para permitir tests unitarios y de integración, simplificando estos últimos para garantizar la calidad y la estabilidad de las aplicaciones.

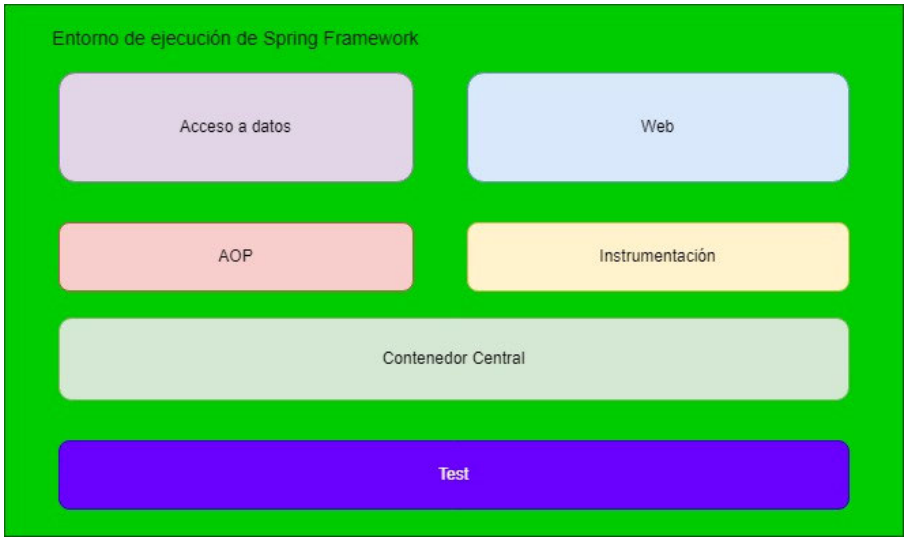

Ilustración 3. Spring Framework Runtime

De acuerdo con todo lo anterior, Spring-Boot integra, además, el servidor de aplicaciones y lo arranca cuando se inicia la aplicación. Esto significa que, adicionalmente, permite distribuir las aplicaciones como imágenes de Docker y configurar el servidor junto con la aplicación.

- **REST:** es un tipo de arquitectura que hace uso de hipermedios, que permiten la navegación y la interacción con los recursos web, para transmitir y representar esta información. Además, al ser un protocolo sin estado, no almacena datos de peticiones en el servidor, cuenta con un conjunto de operaciones predefinidas (conocidas como CRUD) y dispone de URIs que siguen una sintaxis específica.
- **SQL:** es el lenguaje utilizado para gestionar y manejar la información de las bases de datos relacionales. De esta manera, permite la lectura, el borrado y cualquier tipo de manipulación de su contenido.
- **MariaDB:** es un motor de base de datos relacional, de gran rendimiento y estabilidad, que funciona como una extensión de MySQL, y que permite todo tipo de interacciones con la información se almacena en él.
- **Docker:** esta tecnología ha sido empleada para llevar a cabo el despliegue de la aplicación. Se trata de un sistema de organización de contenedores de software que facilita el despliegue y la gestión de aplicaciones, con el fin de garantizar un entorno de ejecución portable y consistente.

Como resultado, estos contenedores se manejan como máquinas virtuales independientes, y tienen la capacidad de ser creados, copiados o importados a otros entornos rápidamente. Para alcanzar esto, Docker promueve la descomposición de las aplicaciones en componentes aislados y les proporciona un entorno de ejecución estable.

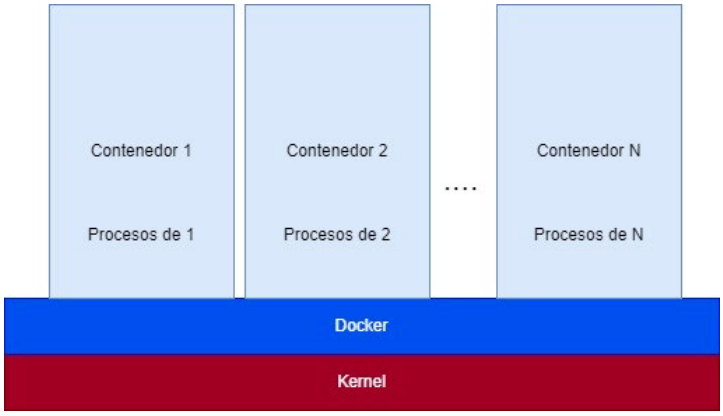

Ilustración 4. Estructura de un entorno con Docker

- **Git:** es un sistema de control de versiones. Se ha optado por utilizar Git para tratar de simplificar y optimizar al máximo la tarea de desarrollo. Esto se ha conseguido llevar a cabo ya que esta tecnología permite controlar los cambios que se introducen en el proyecto, lo que permite retroceder a una versión anterior, desarrollar una rama en paralelo, almacenar la propia aplicación de forma distribuida, etc.

  Además, con Git, se posibilita la idea de trabajo colaborativo entre varios miembros de un mismo equipo, aunque en este caso solo ha habido un desarrollador. De este modo, todos los individuos a cargo de un mismo proyecto pueden seguir los cambios en primera persona y hacer modificaciones independientes al resto.

2.5 Justificación de las decisiones tecnológicas

El primer punto principal que se ha perseguido con el empleo de las tecnologías anteriores era el desarrollo con un lenguaje de programación común, sin depender de tecnologías habituales del *frontend*, como HTML o JavaScript. Así, mediante el lenguaje Java, se consigue implementar tanto el ámbito de la lógica y la infraestructura como el punto de contacto con el usuario.

En primer lugar, Spring-Boot tiene una amplia aceptación en el desarrollo de aplicaciones, ofrece una gran facilidad de configuración y, también, una gran robustez en cuanto a seguridad y rendimiento. Por otro lado, Vaadin, al ser altamente integrable con Spring, y brindar un enfoque sencillo para el desarrollo dinámico de interfaces de usuario web, se constata como una vía original y relativamente novedosa para permitir la interacción de los usuarios con la propia aplicación.

Por consiguiente, Docker ha facilitado considerablemente el despliegue y la gestión de la aplicación, permitiendo modularizar al máximo los componentes que en ella intervienen, y también favorecería una posible migración de la plataforma a la nube, gracias al entorno de ejecución que ofrece, independiente del sistema operativo que lo alberga.

Y finalmente, Git se ha erguido como una herramienta fundamental para guardar todos los cambios realizados, recuperar una versión anterior en caso de haber sido necesario y plantear varias líneas de desarrollo paralelas.

Resumiendo lo planteado, gracias a la combinación de todas estas tecnologías, se ha conseguido construir un entorno fiable para el desarrollo de una aplicación web moderna, escalable y fácil de mantener, satisfaciendo las necesidades que se plantearon inicialmente.

3. Especificaciones y restricciones de diseño

3.1 Introducción

Para llevar a cabo el proyecto, se han tenido que seguir una serie de limitaciones y requisitos, así como consideraciones de usabilidad y accesibilidad, que están detallados a continuación:

3.2 Especificaciones del sistema

El sistema debe cumplir los siguientes requisitos funcionales una vez esté finalizado:

Tabla 1. Requisito funcional 1

RF-1	**Buscador de ofertas**
Descripción	El sistema debe mostrar un buscador de ofertas al iniciar la aplicación

Tabla 2. Requisito funcional 2

RF-2	**Listar ofertas**
Descripción	El sistema debe listar las ofertas que correspondan al filtro seleccionado

Tabla 3. Requisito funcional 3

RF-3	**Información de las ofertas**
Descripción	El sistema debe mostrar la información de la oferta seleccionada, así como un mapa de la ubicación de dónde se ofrece la oferta.

Tabla 4. Requisito funcional 4

RF-4	**Formulario de registro**
Descripción	El sistema debe visualizar un formulario donde los usuarios puedan registrarse o iniciar sesión.

Tabla 5. Requisito funcional 5

RF-5	**Datos de usuario**
Descripción	El sistema debe verificar que los datos introducidos corresponden a un usuario.

Tabla 6. Requisito funcional 6

RF-6	**Funcionalidades de usuario**
Descripción	El sistema debe mostrar las funcionalidades que corresponden al rol del usuario autenticado.

Tabla 7. Requisito funcional 7

RF-7	**Estudiantes solicitantes**
Descripción	El sistema debe mostrar los estudiantes que han solicitado información de una oferta, si el rol es el de empresa.

Tabla 8. Requisito funcional 8

RF-8	**Consultar perfil de estudiante**
Descripción	El sistema permitirá consultar el perfil de un estudiante, si el rol es el de empresa.

Tabla 9. Requisito funcional 9

RF-9	**Contactar estudiante**
Descripción	El sistema permitirá contactar con un estudiante, si el rol es el de empresa.

Tabla 10. Requisito funcional 10

RF-10	**Modificar oferta**
Descripción	El sistema permitirá modificar los datos de una oferta, si el rol es el de empresa.

Tabla 11. Requisito funcional 11

RF-11	**Comprobación de datos modificados**
Descripción	El sistema deberá comprobar si los datos modificados son válidos.

Tabla 12. Requisito funcional 12

RF-12	**Eliminar oferta**
Descripción	El sistema permitirá eliminar una oferta propia, si el rol es el de empresa.

Tabla 13. Requisito funcional 13

RF-13	**Comprobación de eliminación**
Descripción	El sistema deberá comprobar si la oferta se ha eliminado correctamente.

Tabla 14. Requisito funcional 14

RF-14	**Publicar ofertas**
Descripción	El sistema permitirá publicar nuevas ofertas, si el rol es el de empresa.

Tabla 15. Requisito funcional 15

RF-15	**Formulario de oferta**
Descripción	El sistema deberá mostrar un formulario para introducir los datos de la oferta

Tabla 16. Requisito funcional 16

RF-16	**Comprobación de nuevos datos**
Descripción	El sistema debe comprobar si los datos introducidos son válidos.

Tabla 17. Requisito funcional 17

RF-17	Solicitar información
Descripción	El sistema debe mostrar un botón donde solicitar información de la oferta, si el rol es el de estudiante.

Tabla 18. Requisito funcional 18

RF-18	Formulario de solicitud de información
Descripción	El sistema debe mostrar un formulario para solicitar información de una oferta, si el rol es el de estudiante.

Tabla 19. Requisito funcional 19

RF-19	Comprobar datos de formulario
Descripción	El sistema debe comprobar si los datos del formulario son válidos.

Tabla 20. Requisito funcional 20

RF-20	Cancelar solicitud
Descripción	El sistema permitirá cancelar la solicitud de una oferta, si el rol es el de estudiante.

Tabla 21. Requisito funcional 21

RF-21	Perfil de empresa
Descripción	El sistema permitirá ver el perfil de una empresa, si el rol es el de estudiante.

Tabla 22. Requisito funcional 23

RF-22	Modificar datos de usuario
Descripción	El sistema permitirá modificar los datos de su propio perfil de usuario, si el rol es el de empresa o estudiante.

Tabla 23. Requisito funcional 24

RF-23	**Contenido multimedia**
Descripción	El sistema permitirá añadir contenido multimedia al perfil de usuario, si el rol es el de estudiante o empresa.

Tabla 24. Requisito funcional 25

RF-24	**Visualizar contenido multimedia**
Descripción	El sistema permitirá visualizar el contenido multimedia en el perfil de usuario, si el rol es el de estudiante o empresa.

Tabla 25. Requisito funcional 26

RF-25	**Mostrar ofertas propias**
Descripción	El sistema permitirá mostrar todas las ofertas que el usuario ha publicado, si el rol es el de empresa

Tabla 26. Requisito funcional 27

RF-26	**Solicitudes a ofertas**
Descripción	El sistema debe mostrar todas las ofertas sobre las que el usuario ha solicitado información, si el rol es el de estudiante.

Tabla 27. Requisito funcional 28

RF-27	**Visualizar mensajes**
Descripción	El sistema permitirá visualizar los mensajes recibidos, si el rol es el de estudiante o empresa.

Tabla 28. Requisito funcional 29

RF-28	**Enviar mensajes**
Descripción	El sistema permitirá enviar un mensaje a otro usuario de la aplicación, si el usuario es el de estudiante o empresa.

Tabla 29. Requisito funcional 31

RF-29	Eliminar usuario
Descripción	El sistema permitirá eliminar un usuario, si el rol es el de administrador.

Tabla 30. Requisito funcional 32

RF-30	Eliminar oferta
Descripción	El sistema permitirá eliminar una oferta, si el rol es el de administrador.

Tabla 31. Requisito funcional 33

RF-33	Crear usuario
Descripción	El sistema permitirá crear un nuevo usuario, si el rol es el de administrador.

Tabla 32. Requisito funcional 34

RF-31	Cerrar sesión
Descripción	El sistema permitirá cerrar la sesión de un usuario.

3.3 Restricciones de diseño

El sistema debe cumplir los siguientes requisitos funcionales una vez esté finalizado, clasificados por funcionalidades:

- **Seguridad:**

Tabla 33. Requisito no funcional 1

RNF-01	Autenticación
Descripción	Debe existir un proceso de autenticación de usuarios.

Tabla 34. Requisito no funcional 2

RNF-02	Identificación
Descripción	Los usuarios podrán identificarse por medio de un correo electrónico y una contraseña.

- **Accesibilidad:**

Tabla 35. Requisito no funcional 3

RNF-03	Acceso al sistema
Descripción	Se debe acceder al sistema a través de un navegador web.

- **Usabilidad:**

Tabla 36. Requisito no funcional 4

RNF-04	Uso de la aplicación
Descripción	El usuario debe ser capaz de utilizar la aplicación sin ningún problema tras haber consultado el manual de usuario.

Tabla 37. Requisito no funcional 5

RNF-05	Correcto funcionamiento
Descripción	El sistema debe asegurar un funcionamiento adecuado y sin incidencias.

- **Escalabilidad:**

Tabla 38. Requisito no funcional 6

RNF-06	Escalabilidad
Descripción	La aplicación debe ser escalable, es decir, debe permitir futuras actualizaciones sin perder el rendimiento y el funcionamiento alcanzados.

- **Interfaz:**

Tabla 39. Requisito no funcional 7

RNF-07	**Interfaz interactiva**
Descripción	El sistema debe tener una interfaz de usuario interactiva, atractiva e intuitiva.

- **Interoperabilidad:**

Tabla 40. Requisito no funcional 8

RNF-08	**Compatibilidad**
Descripción	El sistema debe ser compatible con cualquier sistema operativo y con todos los navegadores que puedan tener los dispositivos.

- **Mantenibilidad:**

Tabla 41. Requisito no funcional 9

RNF-09	**Mantenibilidad**
Descripción	El sistema debe sobreponerse a cualquier error que pueda producirse

- **Requisitos de información:**

Tabla 42. Requisito no funcional 10

RNF-10	**Datos de autenticación**
Descripción	El sistema ha de almacenar los datos de autenticación de los usuarios que se registran.

Tabla 43. Requisito no funcional 11

RNF-11	**Información de usuario**
Descripción	El sistema debe almacenar la información de los usuarios que se registran.

Tabla 44. Requisito no funcional 12

RNF-12	**Información de ofertas**
Descripción	El sistema debe almacenar la información de las ofertas que se publican.

- **Requisitos de restricción de información:**

Tabla 45. Requisito no funcional 13

RNF-13	**Registro de usuarios**
Descripción	El sistema no puede registrar a dos usuarios con el mismo correo electrónico, ni tampoco con el mismo DNI o CIF.

Tabla 46. Requisito no funcional 14

RNF-14	**Publicación de ofertas**
Descripción	El sistema no puede registrar más de una oferta con el mismo identificador.

- **Requisitos de despliegue:**

Tabla 47. Requisito no funcional 15

RNF-15	**Modo de despliegue**
Descripción	El sistema deberá desplegarse mediante contenedores independientes, utilizando la tecnología de Docker.

Tabla 48. Requisito no funcional 16

RNF-16	**Despliegue**
Descripción	El sistema deberá estar granulado al máximo a la hora de su despliegue, desplegando tantos contenedores como componentes disponga.

4. Descripción de la solución propuesta

En esta sección se detalla todo lo referente al diseño completo del sistema. Su objetivo es indicar y especificar como deberá ser implementado el sistema, cumpliendo con lo descrito en la sección de requisitos anterior.

4.1 Decisiones de diseño

En esta etapa inicial se tomaron una serie de decisiones que marcarían toda la etapa de diseño e implementación:

- Para la parte de lógica de negocio de decide usar Spring-Boot.
- Para la parte de interfaz gráfica se decide utilizar Vaadin, integrado con Spring-Boot.

- La arquitectura general del proyecto se diseña para separar la lógica de negocio en capas, con una capa de presentación (Vaadin sobre Spring-Boot), una capa de negocio (API REST sobre Spring-Boot) y una capa de datos (base de datos).
- Se decide usar una base de datos relacional, MariaDB, para poder manejar grandes cantidades de datos y consultas complejas.
- Se implementa autenticación y autorización utilizando Spring Security, para garantizar que solo los usuarios autorizados puedan acceder a ciertas partes de la aplicación.
- Se decide utilizar RESTful APIs para la comunicación entre la interfaz visible y la capa de negocio, permitiendo una separación clara de responsabilidades y una mayor flexibilidad.
- Se implementan mecanismos de manejo de errores y excepciones para garantizar que la aplicación pueda manejar situaciones inesperadas de manera robusta y segura.
- Se decide utilizar Docker para desplegar la aplicación, mediante scripts automatizados para facilitar el proceso.

4.2 Arquitectura física: Despliegue

Aquí se muestra el diagrama de despliegue del sistema. Los nodos y componentes son:

- El **navegador web** usado por el usuario para enviar peticiones http al servidor.
- El **Docker host**, o equipo que alberga Docker. En este caso sería el PC utilizado para el desarrollo.
- Los **contenedores**, que son cada uno de los servicios independientes del sistema, ejecutándose en componentes individuales para una mayor modularización del mismo.
- **PFG_frontend**, que es la interfaz gráfica desarrollada con Vaadin.
- **PFG_backend**, que es el RESTful API desarrollada sobre Spring-Boot.
- **Maria DB**, la base de datos que almacena la información del sistema.
- **DB Adminer**, administrador básico de bases de datos para facilitar la visualización del contenido de la base de datos.

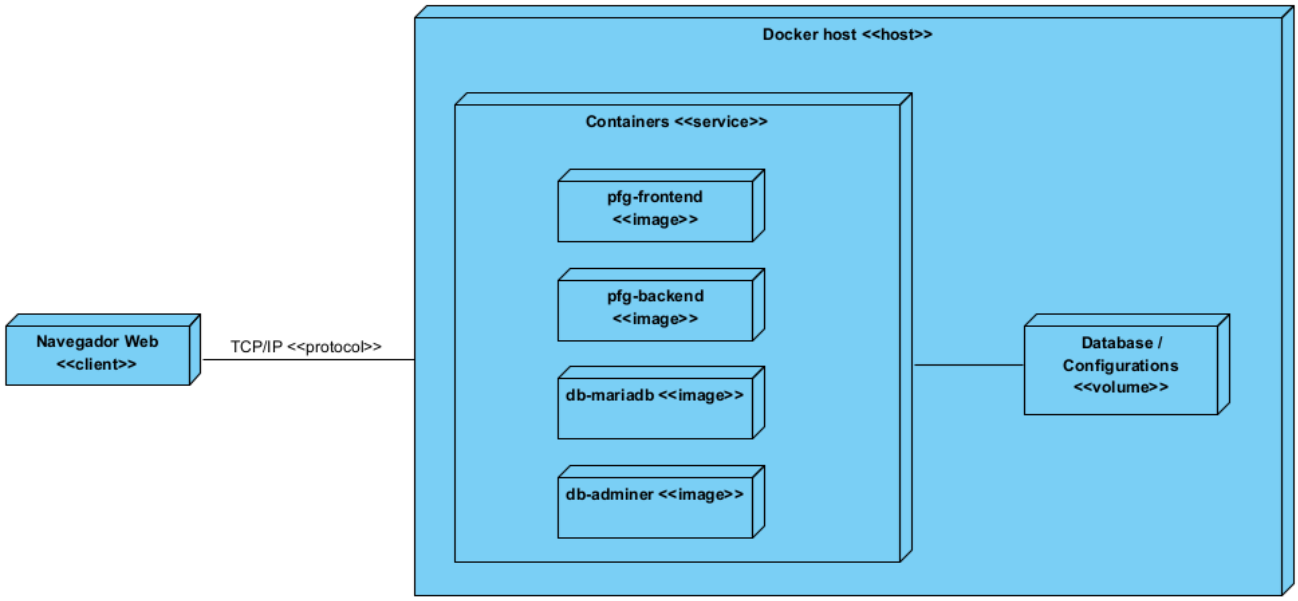

Ilustración 5. Diagrama de despliegue

4.3 Casos de uso del diseño

A continuación, se muestran los diagramas de casos de uso, separados por actores, con el fin de poder facilitar la visualización de este:

- **Usuario invitado:** los casos de uso para el usuario con rol de invitado son los siguientes:

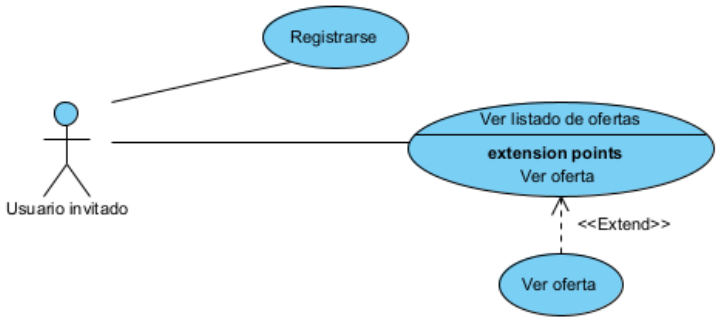

Ilustración 6. Diagrama de casos de uso para usuarios sin registrar

- **Administrador:** los casos de uso para el usuario con rol de administrador son los siguientes:

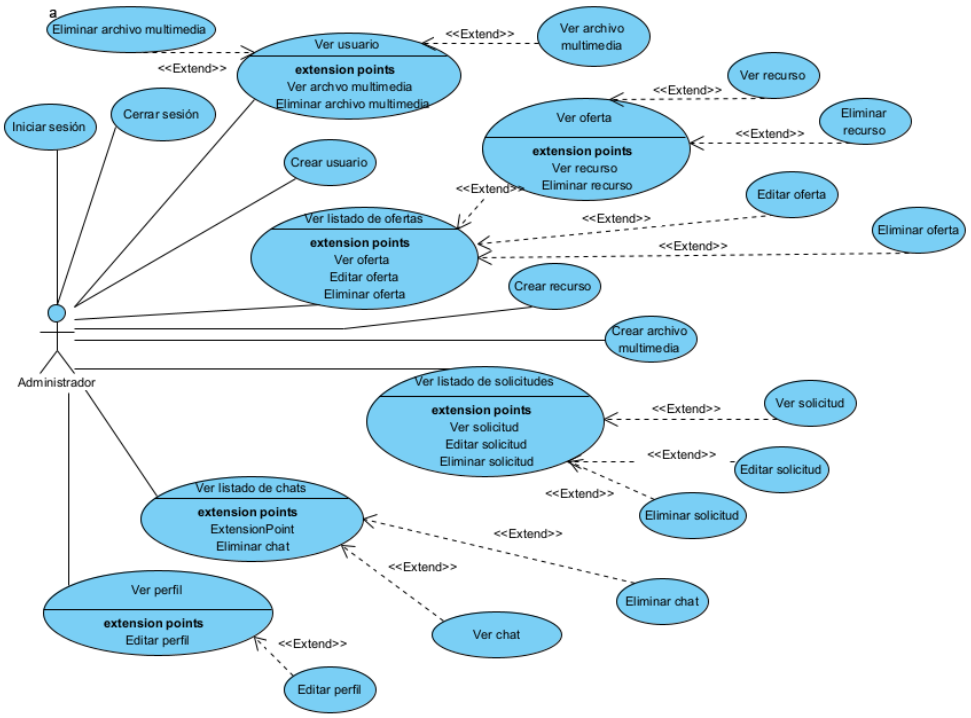

Ilustración 7. Diagrama de casos de uso para Administradores

- **Empresa:** los casos de uso para el usuario con rol de empresa son los siguientes:

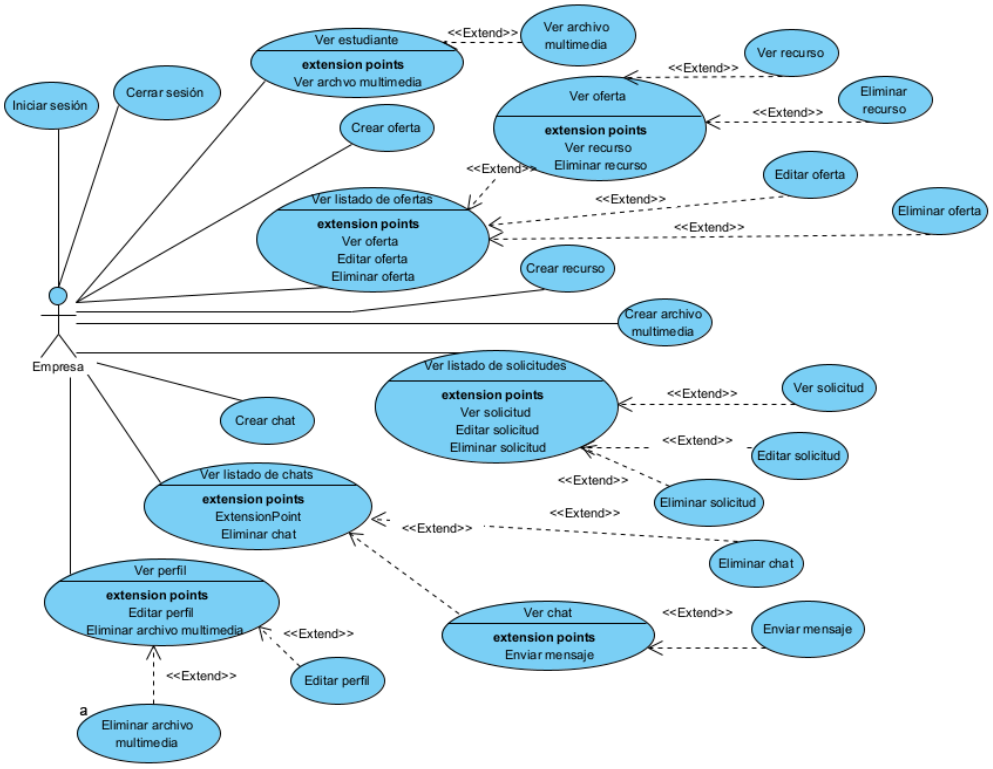

Ilustración 8. Diagrama de casos de uso de Empresas

- **Estudiante:** los casos de uso para el usuario con rol de estudiante son los siguientes:

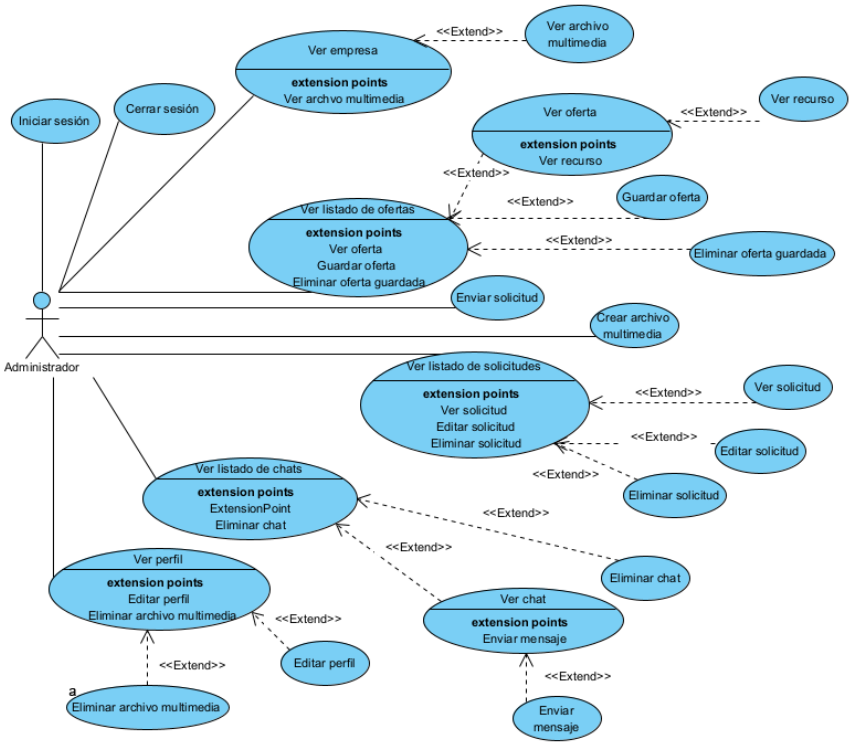

Ilustración 9. Diagrama de casos de uso de Estudiantes

4.4 Diseño de la base de datos

Seguidamente, se presenta el diagrama de diseño de las tablas de la base de datos, con el fin de poder representar de forma visual y clara la estructura y las relaciones entre las entidades. Este diagrama es fundamental para entender cómo se organizan y se relacionan los datos, lo que facilita la comprensión y el manejo de la información almacenada:

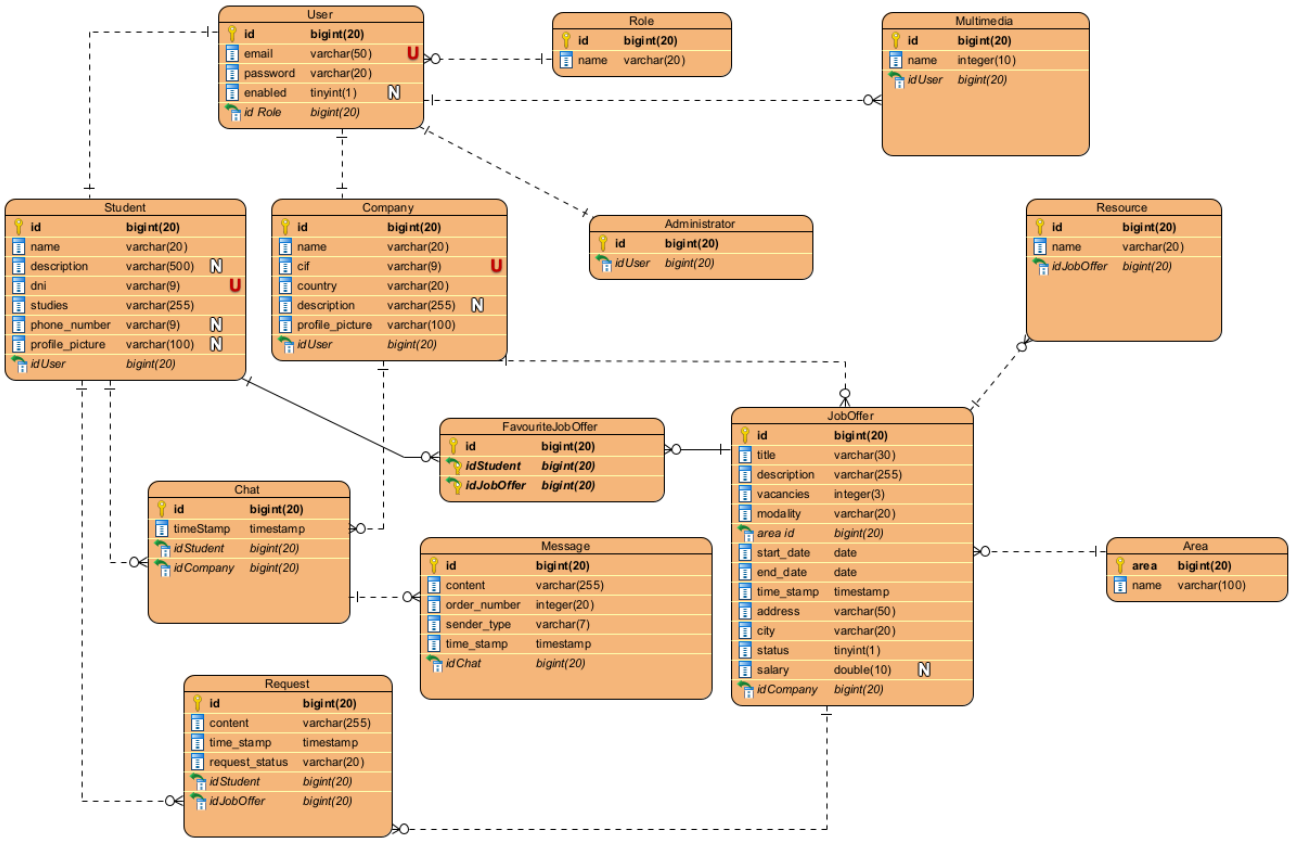

Ilustración 10. Diseño de la base de datos

4.5 Diseño de clases

Dentro del diseño de clases de la lógica de negocio, el contenido se ha separado en paquetes para facilitar la visibilidad de cada una de las clases:

- **Entidades:** paquete de entidades (*entities*), que representan los objetos de negocio de la aplicación:

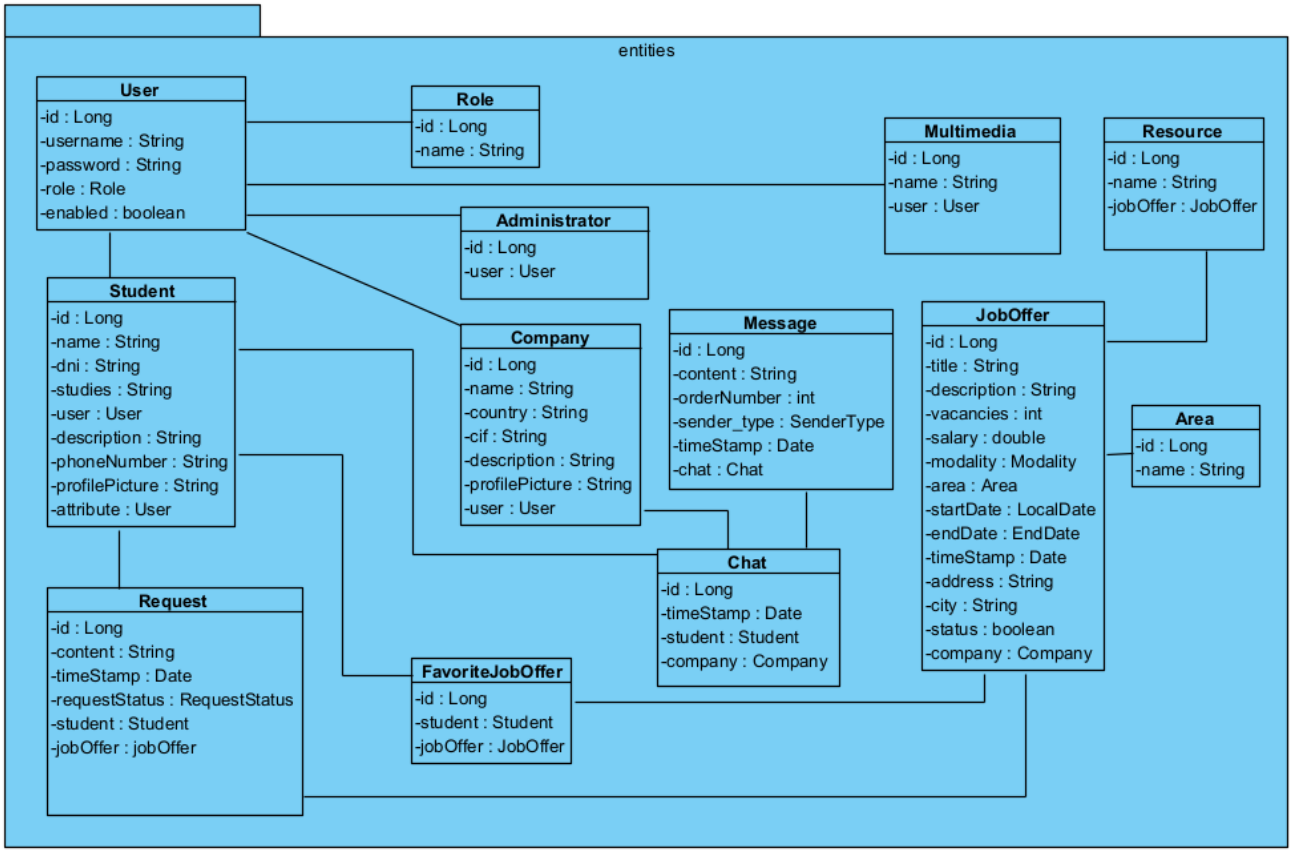

Ilustración 11. Diseño de clases del paquete entity

- **Dtos:** paquete de objetos utilizados para serializar y deserializar los objetos de negocio anteriores:

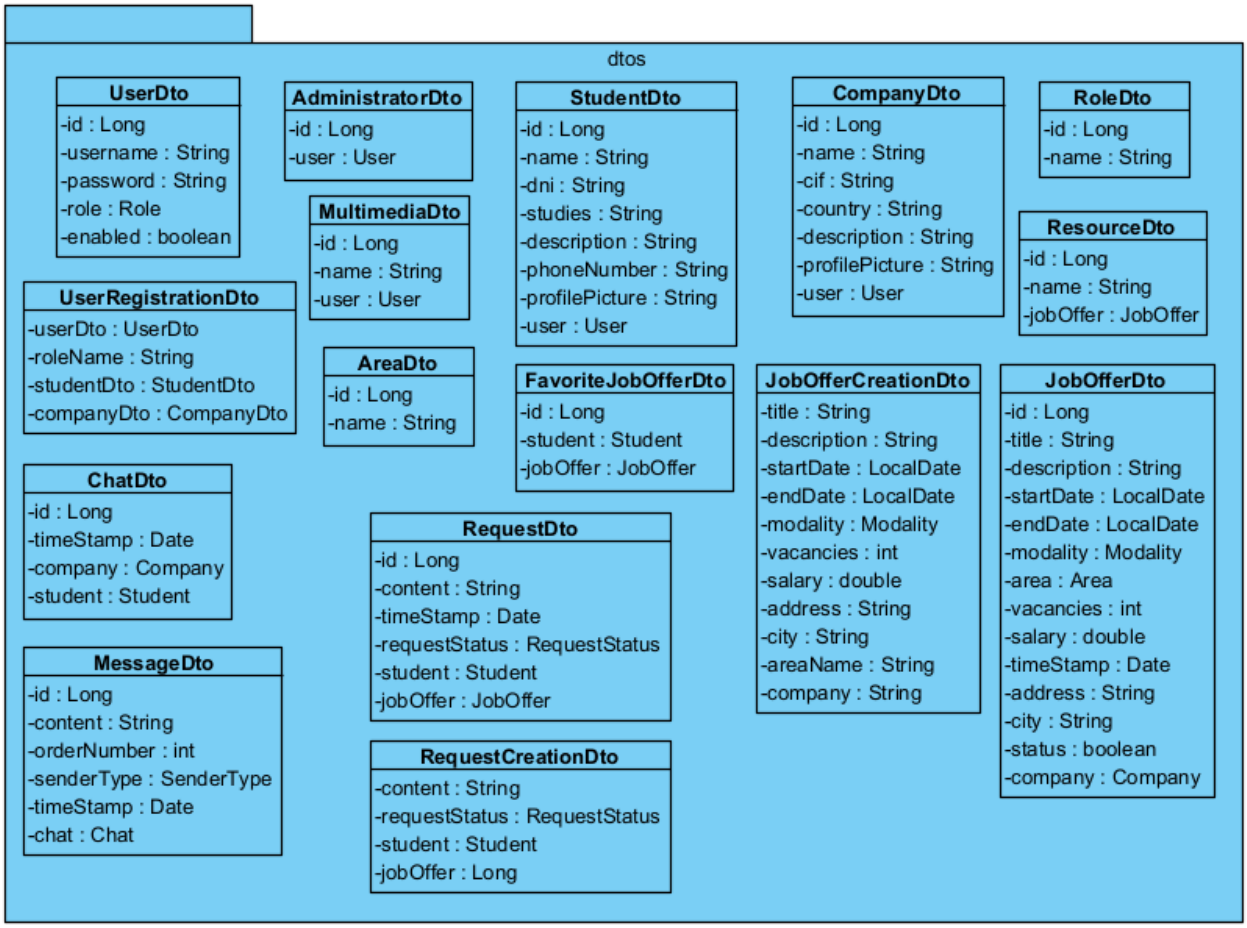

Ilustración 12. Diseño de clases del paquete dtos

- **Repositorios:** paquete de repositorios (*repositories*) de acceso a la base de datos. Todas las clases extienden de *JpaRepository*, lo cual no se indica en el diagrama para facilitar la lectura.

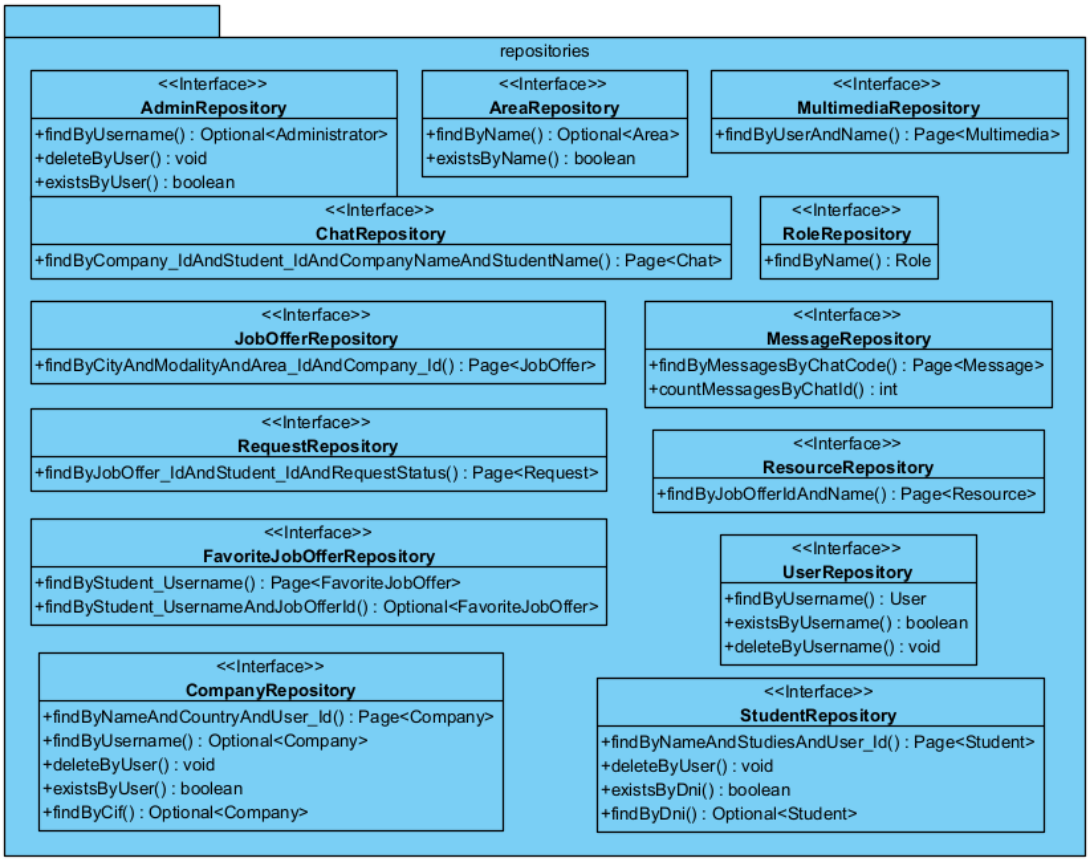

Ilustración 13. Diagrama de clases del paquete repositories

- **Servicios:** paquete de servicios (*services*) de la RESTful API, que encapsulan la lógica de negocio de la aplicación:

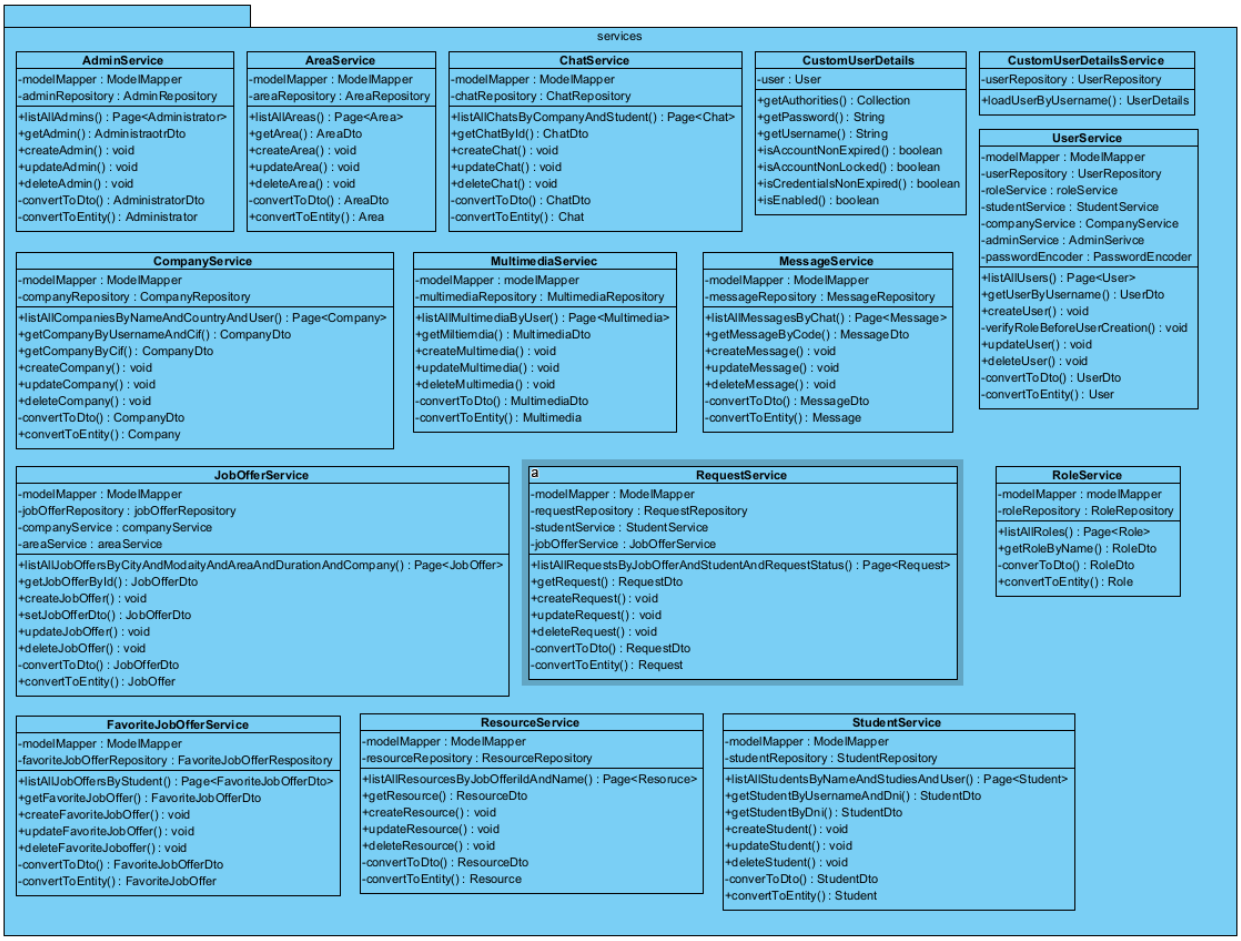

Ilustración 14. Diagrama de clases del paquete services

- **Rest:** paquete de controladores rest, que manejan las solicitudes HTTP:

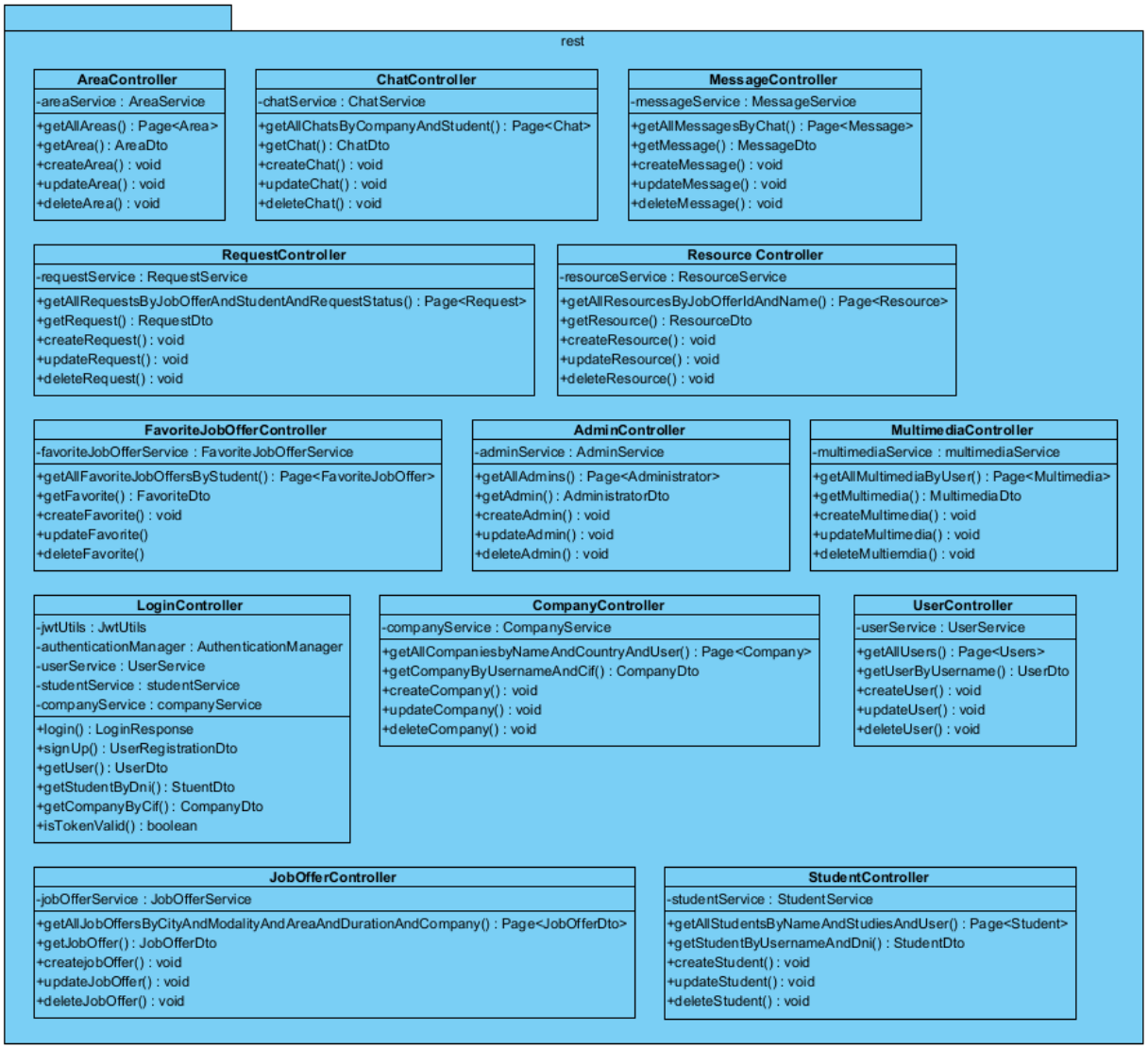

Ilustración 15. Diagrama de clases del paquete rest

- **Config:** paquete que define las clases de configuración de seguridad de la aplicación:

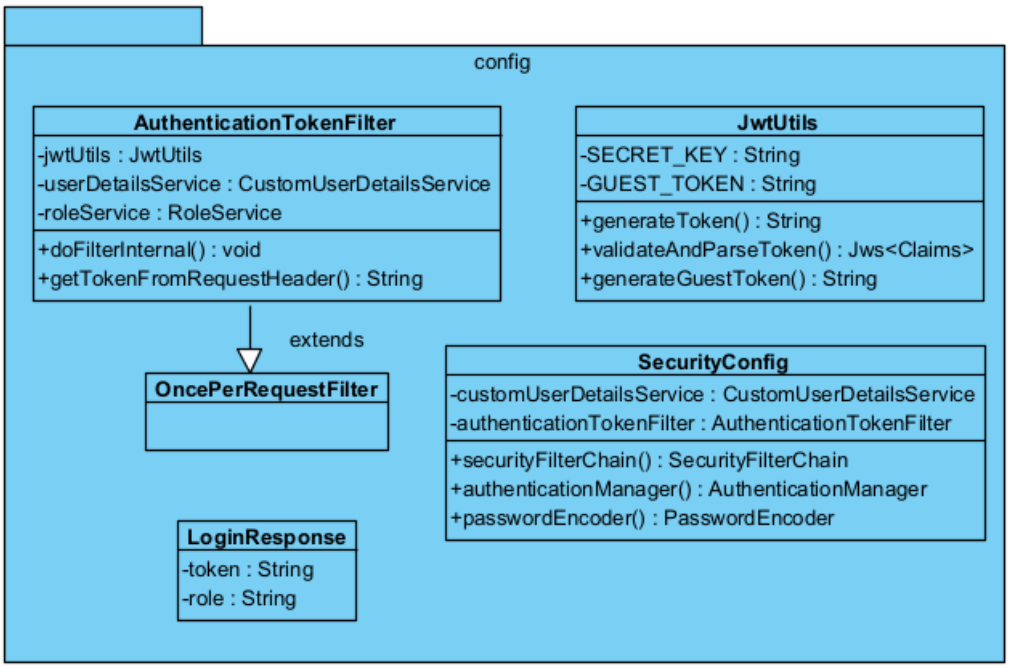

Ilustración 16. Diseño de clases del paquete config

4.6 Diagrama de vistas

Con este diagrama se pretende representar de forma visual cada una de las diferentes vistas de las que podrá disponer la interfaz gráfica. Se señalan las principales interacciones entre ellas, dejando sin señalar las que son comunes a todas para facilitar su legibilidad:

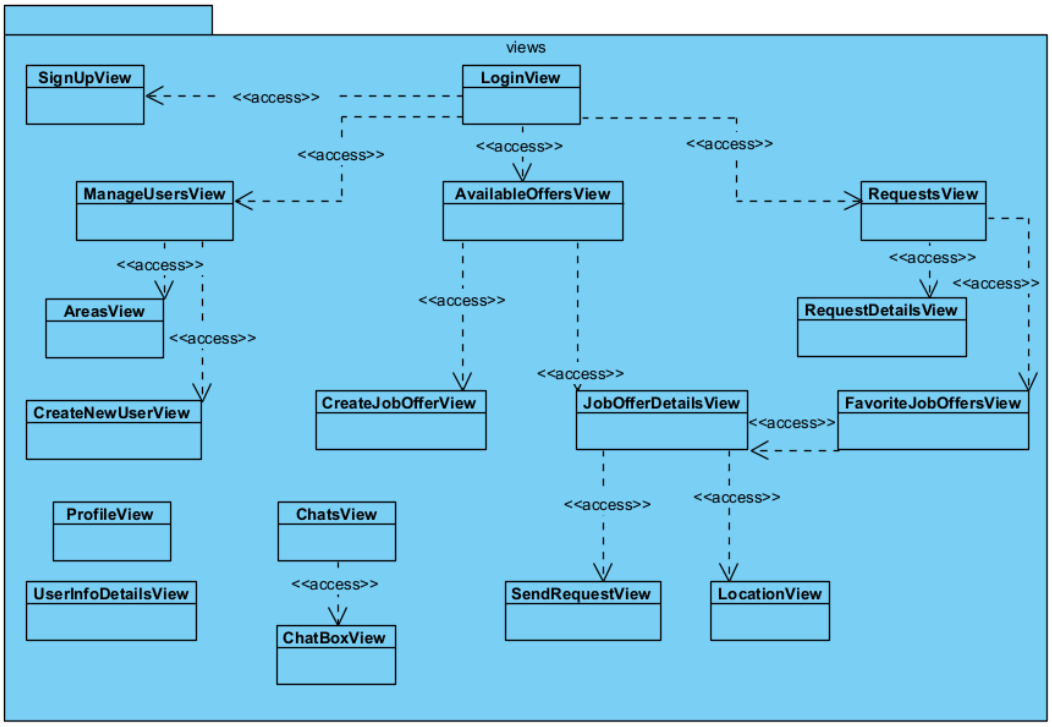

Ilustración 17. Diagrama de las vistas de la interfaz gráfica

4.7 Implementación

Partiendo del diseño anterior, se procede con la implementación completa del sistema. En primer lugar, para el desarrollo de la RESTful API se utilizan las siguientes dependencias:

- **Spring-Boot Starter web:** permite la creación de aplicaciones web RESTful, incluyendo dependencias para Spring MVC, Spring Web y Tomcat.
- **Spring-Boot Starter Validation:** permite la validación de datos en aplicaciones web.
- **MariaDB Java Client:** permite interactuar con bases de datos MariaDB con Java.
- **Lombok:** permite generar los llamados *getters* (posibilitan el acceso al valor de un atributo) and *setters* (proporcionan el valor de un atributo), constructores por defecto y algunos métodos auxiliares más, reduciendo considerablemente las líneas de código empleadas.
- **Hibernate Validator:** permite valor datos en aplicaciones web mediante reglas de validación.
- **ModelMapper:** permite el mapeo de objetos para convertir objetos de presentación en objetos de negocio y viceversa.
- **Springfox Swagger 2:** permite generar automáticamente documentación de API REST basada en el estándar Swagger 2.0.
- **Spring-Boot Starter Security:** permite implementar la autenticación y la autorización en aplicaciones web.
- **Spring Security Core:** proporciona funcionalidades básicas para autenticar y autorizar a usuarios en aplicaciones web.

- **Json Web Token:** permite autenticar y autorizar usuarios en aplicaciones web mediante la serialización de información de autenticación en un token enviado en las peticiones HTTP.

Estas dependencias se añaden fácilmente gracias a Maven, que proporciona un fichero en formato XML, denominado *pom.xml*, donde se señala toda la información relativa del proyecto para su compilación.

Por consiguiente, en la parte de codificación de la RESTful API, hay ocho paquetes claramente diferenciados, definidos en la sección de diseño:

- **Entidades (*entities*):** representa el grupo de clases que hacen referencia a los objetos de negocio de la aplicación, los cuales poseen los campos que representan las propiedades de éstos últimos. De este modo, mediante etiquetas, se incorporan todas las funcionalidades necesarias para la generación de la base de datos, facilitando el proceso de codificación. Principalmente, se hace uso se la librería *Jakarta.Persistance*, destacando etiquetas como *@Entity* o *@Table*, entre otras, para especificar que la clase es una entidad, para especificar la tabla que se utilizará en la base de datos, etc.
- **Dtos:** representa el grupo de clases necesarias para serializar y deserializar las entidades que se emplean para manejar operaciones con la base de datos, generando objetos auxiliares con las mismas propiedades. Además, se ayudan de las etiquetas *@Data* y *@NoArgsConstructor*, ofrecidas por Lombok, que simplifican las líneas de código considerablemente al autogenerar los denominados métodos *getters* y *setters* y los constructores por defecto.
- **Repositorios (*repositories*):** contiene las interfaces que se utilizan para interactuar con la base de datos. Todas ellas extienden de la clase *JpaRepository* (perteneciente a la librería *org.springframewrok.data.jpa.repository*) , que define la persistencia de datos en Java proporcionada por Spring, para facilitar el acceso a los datos. Se ayudan de una serie de etiquetas, como @Query o @Param, para especificar una consulta SQL concreta y para pasar parámetros a estas consultas, respectivamente.
- **Servicios (*services*):** contiene las clases donde se ejecuta la lógica de negocio de la aplicación. Reciben los parámetros del controlador para efectuar las operaciones pertinentes, y le devuelven la respuesta. Además, hacen uso de los repositorios para conectarse con la base de datos, y de *ModelMapper* para hacer mapeos entre *dtos* y entidades para realizar consultas y operaciones. Adicionalmente, utilizan etiquetas como *@Service* o *@Autowired*, para especificar que la clase es un servicio y para inyectar instancias de componentes en un atributo de la clase.
- **Rest:** contiene las clases que manejan las solicitudes HTTP. Define varios métodos donde se ejecutan operaciones CRUD apuntando a la lógica de negocio real y devuelven los correspondientes resultados. Para ello, hacen uso de etiquetas para señalar la funcionalidad de controlador (*@RestController*) o para señalar la funcionalidad CRUD del método *(@GetMapping* o *@PostMapping*). En los controladores también se concreta la seguridad, ayudándose de la etiqueta @EnabledMethodSecurity, para habilitar los mecanismos de seguridad proporcionados por Spring, y de *@PreAuthorize*, para definir la expresión que limita el acceso a los recursos.
- **Config:** este paquete agrupa todas las clases necesarias para configurar la seguridad implementada en la aplicación. Las etiquetas más importantes de las que se sirve para incorporar estas características son *@EnableWebSecurity*, que habilita la configuración de seguridad realizada, y *@Configuration*, señalando a una clase para que sea gestionada por Spring como clase de configuración.

Además, es necesario definir una serie de configuraciones de la aplicación. Estas se definen en el fichero *application.yml*, que un archivo de tipo YAML que contiene pares clave-valor para definir propiedades de la aplicación. Las necesarias para el caso actual son las siguientes, aunque algunas no se incluyen por cadenas de caracteres sin ningún significado específico:

Tabla 49. Propiedades del fichero application.yml del backend

Propiedad	Valor	Uso
server.port	8080	Especifica el puerto donde escucha el servidor web
authentication .secret-key	x	Define la constante *secret-key,* utilizada para encriptar los tokens de usuario
authentication .guest-token	x	Define la constante *guest-token,* que es el token asignado a todos los usuarios invitados
spring.mvc.pathmatch .matching-strategy	ant-path-matcher	Especifica la estrategia de coincidencia de patrones para asignar las solicitudes HTTP a controladores
Spring.datasource.url	jdbc:mariadb:// ${DB_HOST:localhost}:${DB_PORT:3306} /application_data?useUnicode=true &characterEncoding=UTF8 &collation=utf8mb4_general_ci	Define la URL de conexión con la base de datos. Extrae de variables de entorno el valor del nombre del contenedor y del puerto donde se escucha la base de datos
spring.datasource .driver-class-name	org.mariadb.jdbc.Driver	Define la clase del controlador JDBC que se debe emplear para conectarse a la base de datos
spring.datasource.username	${MARIADB_USER:admin}	Define el usuario de acceso a la base de datos, extraído de la correspondiente variable de entorno
spring.datasource.password	${MARIADB_PASSWORD:admin}	Define la contraseña de acceso a la base de datos, extraído de la correspondiente variable de entorno

spring.jpa.database-platform	org.hibernate.dialect.MariaDBDialect	Define el dialecto de Hibernate para la base de datos
spring.jpa.generate-ddl	false	Indica si se deben generar los scripts de creación de la base de datos
spring.jpa.show-sql	true	Activa el *logging* para operaciones con la base de datos
spring.jpa.hibernate.ddl-auto	update	Define la estrategia de creación de la base de datos, actualizando los datos para que coincidan con los definidos en la aplicación
spring.security.user.name	admin	Especifica el nombre de usuario para un usuario por defecto
spring.security.user.password	admin	Especifica la contraseña del usuario por defecto
spring.security.user.role	ADMIN	Especifica el rol asignado al usuario por defecto
logging.level	info / debug	Se establece el nivel de *logging* en *info* para operaciones con la base de datos y genéricas, y en *debug* para cuestiones de seguridad

Con la lógica de negocio y la RESTful API ya implementados, se procede con la creación de la interfaz de usuario. Para ello se hace uso de las siguientes dependencias:

- **Spring-Boot Starter Web** y **Lombok**, descritas anteriormente.
- **Vaadin Spring-Boot Starter:** dependencia de Vaadin específica para Spring-Boot.
- **Okhttp:** permite realizar peticiones HTTP a API REST, manejando todo tipo de funcionalidades relacionadas con ellas.
- **Json:** es una implementación específica de JSON para Java, que permite trabajar con objetos de este formato.
- **MapLibre:** extensión de Vaadin para la creación de mapas interactivos.
- **Vcf Pdf Viewer:** extensión de Vaadin para la creación de componentes visuales que muestran documentos en formato PDF.
- **Video JS Player:** extensión de Vaadin que permite la creación de componentes visuales que muestran vídeos.

Tal y como se ha descrito en la implementación del API REST, estas dependencias se añaden al fichero *pom.xml* para su correcta compilación.

A continuación, se desarrolla la interfaz gráfica, dividiendo el código en tres paquetes bien definidos:

- **Vistas (views):** paquete donde se definen todas y cada una de las vistas que el usuario va a poder visualizar en el navegador. Todas ellas extienden del componente personalizado *CustomAppLayout* (salvo *LoginView* y *SignUpView*), que se describe en el siguiente paquete, para unificar la estructura visual de las vistas en el navegador. Así, las vistas generadas son las siguientes:

Tabla 50. Definición de vistas de la interfaz gráfica

Nombre de la clase	Descripción
LoginView	Muestra el formulario de inicio de sesión
SignUpView	Muestra el formulario de registro para estudiantes y empresas
AvailableOffersView	Muestra el listado de ofertas disponibles y algunos de sus atributos
JobOfferDetailsView	Muestra los detalles de una oferta
LocationView	Muestra en un mapa la localización de la oferta seleccionada
SendRequestView	Específica para estudiantes, muestra el formulario para enviar una solicitud de una oferta
RequestsView	Muestra un listado de solicitudes ya realizadas
RequestDetailsView	Muestra los detalles de una solicitud
UserInfoDetailsView	Muestra los detalles del perfil de un usuario, ya sea estudiante, empresa o administrador
FavoriteJobOffersView	Específica para estudiantes, muestra el listado de ofertas guardadas como favoritas
ChatsView	Muestra el listado de chats de un usuario
ChatBoxView	Muestra una conversación específica entre un estudiante y una empresa
AreasView	Específica para administradores, muestra las áreas ya creadas, permitiendo añadir o eliminar
CreateJobOffersView	Específica de empresas, muestra un formulario para crear una nueva oferta, asignada a la empresa seleccionada

ManageUsersView	Específica de administradores, muestra el listado de usuarios y permite su gestión directa
CreateNewUserView	Específica de administradores, muestra el formulario que permite crear un nuevo usuario
ProfileView	Muestra el perfil del usuario actual

Para adquirir las funcionalidades descritas en la tabla anterior, se hace uso de las siguientes etiquetas o componentes, todas pertenecientes a la librería *com.vaadin.flow.router* de Vaadin:

Tabla 51. Etiquetas y componentes utilizados en las vistas

Etiqueta / Componente	Uso
@Route	Especifica la ruta a través de la cual se puede acceder a la vista
@RouteAlias	Permite especificar un alias, para llegar a una vista a través de más de una ruta
@PageTitle	Permite asignar un título a la pestaña cuando se entra a la ruta correspondiente
BeforeEnterObserver	Interfaz que permite realizar una serie de operaciones antes de construir la vista, permitiendo verificar los permisos que tiene cada usuario, por ejemplo
HasUrlParameter<T>	Permite definir un parámetro en la ruta de una vista

- **Personalizado (custom):** paquete donde se definen todos los componentes que han sido personalizados de algún modo para poder ser reutilizados, con el fin de reducir las líneas de código en la implementación.

Tabla 52. Componentes personalizados

Componente	Uso
CustomAppLayout	Layout personalizado, que extiende de la clase de Vaadin *AppLayout* y que muestra la vista con una interfaz amigable
CustomAvatar	Componente personalizado que extiende de la clase de Vaadin *Avatar*, para mostrar la foto de perfil de los usuarios
CustomBasicDialog	Componente personalizado que extiende de la clase de Vaadin *Dialog*, utilizado para mostrar diálogos de aviso al usuario
CustomChatsGrid	Componente personalizado que extiende de la clase de Vaadin *Grid*, para poder mostrar un listado de chats de forma organizada

CustomFilesGrid	Componente personalizado que extiende de la clase de Vaadin *Grid*, para poder mostrar un listado de chats de forma organizada
CustomHeaderButtonsLayout	Layout personalizado para mostrar los botones de la barra de navegación superior
CustomJobOfferListItem	Layout personalizado que conforma un ítem de los listados de las ofertas con varios componentes para reflejar la información de estas
CustomJobOffersGrid	Componente personalizado que extiende de la clase de Vaadin *Grid*, para poder mostrar un listado de ofertas de forma organizada
CustomNavigationOptionsPageLayout	Layout personalizado para mostrar los botones de navegación entre paginas cuando se muestren listados
CustomNotification	Componente personalizado que extiende de la clase de Vaadin *Notification*, para mostrar las notificaciones en la aplicación de forma estandarizada
CustomNumElementsSelect	Componente personalizado que extiende de la clase de Vaadin *Select*, para mostrar un desplegable con el listado de número de elementos a mostrar
CustomRequestsGrid	Componente personalizado que extiende de Grid, para poder mostrar un listado de solicitudes de forma organizada
CustomSelectAreas	Componente personalizado que extiende de la clase de Vaadin *Select*, para mostrar un desplegable con el listado áreas disponibles
CustomSideNav	Componente personalizado que extiende de la clase de Vaadin *SideNav*, para estandarizar la barra de navegación y los posibles atajos en todas las vistas
CustomSignUpComponent	Layout personalizado que contiene todos los componentes necesarios para conformar el formulario de registro de las credenciales de usuario
CustomSignUpStudent	Layout personalizado que contiene todos los componentes necesarios para conformar el formulario de registro de los detalles de perfil de estudiantes
CustomTextArea	Layout personalizado que contiene todos los componentes necesarios para conformar el formulario de registro de los detalles de perfil de empresas
CustomUsersGrid	Componente personalizado que extiende de *Grid*, para poder mostrar un listado de usuarios de forma organizada

- **Utils:** paquete donde se definen clases auxiliares de ayuda para operaciones concretas. Su utilidad es la siguiente:

Tabla 53. Clases auxiliares

Componente	Uso
ChatListComponent	Objeto serializado para poder crear listas de chats más fácilmente a través del *Grid* personalizado
Constants	Clase donde se definen las constantes utilizadas, con el fin de poder reutilizar etiquetas, mensajes de error, etc
Countries	Enumerado con una lista de países
FileListComponent	Objeto serializado para poder crear listas de archivos más fácilmente a través del *Grid* personalizado
HttpBackendClient	Clase que encapsula la lógica del cliente HTTP destinado a hacer peticiones al backend. Posee los métodos necesarios para hacer las peticiones GET, POST, PUT y DELETE
HttpApiCoordinatesClient	Clase que encapsula la lógica del cliente HTTP destinado a hacer peticiones a la API Here, una API de geolocalización para obtener las coordenadas a partir de una dirección postal
JobOfferListComponent	Objeto serializado para poder crear listas de ofertas más fácilmente a través del *Grid* personalizado
RequestListComponent	Objeto serializado para poder crear listas de solicitudes más fácilmente a través del *Grid* personalizado
UserListComponent	Objeto serializado para poder crear listas de usuarios más fácilmente a través del *Grid* personalizado

Como se ha indicado en la tabla anterior, se ha desarrollado un cliente HTTP para hacer peticiones a una RESTful API específica de geocodificación, la cual se denomina HERE. Esto se debe a la necesidad de obtener las coordenadas geográficas a partir de una dirección, para poder mostrar una localización determinada en un mapa. Así, pues, esta plataforma ofrece este servicio y se ha incorporado a la implementación.

De este modo, se conforma toda la interfaz gráfica. Sin embargo, de igual forma que con el API REST, hace falta introducir una serie de propiedades en el fichero *application.yml*:

Tabla 54. Propiedades del fichero application.yml del frontend

Propiedad	Valor	Uso
server.port	8080	Especifica el puerto donde escucha el servidor web

api-endpoint.backend-hostname	${BACKEND_HOST:localhost}	Extrae el valor de la variable de entorno que indica el nombre del contenedor donde se ejecuta el backend
api-endpoint.backend.port	${BACKEND_PORT:8080}	Extrae el valor de la variable de entorno que indica el puerto del contenedor donde escucha las peticiones
here-api.api-key	${API_HERE_KEY}	Clave del API Here para poder hacer peticiones autorizadas. Se extrae de una variable de entorno
spring.servlet.multipart.max-file-size	20MB	Define el tamaño máximo de archivos admitidos
spring.servlet.multipart.max-request-size	20MB	Define el tamaño máximo de archivos admitidos en peticiones HTTP
vaadin.frontend.hotdeploy	true	Activa el modo producción de Vaadin

Como último paso, se procede a la creación de los paquetes *jar* ejecutables y a la dockerización de la aplicación. Esto es, empaquetar la aplicación y sus dependencias en un contenedor de software independiente. En primer lugar, se debe ejecutar por consola el siguiente comando que ofrece la herramienta Maven:

```
#mvn package
```

Posteriormente, se ejecuta la instrucción que permite la dockerización de la aplicación. Pero para ello, primero es necesario definir un fichero *Dockerfile*, que proporciona las instrucciones para construir una imagen de la aplicación, lista para ser ejecutada en un contenedor:

```
from openjdk:17-slim
COPY target/PFG_Frontend-1.0s.jar /app/PFG_Frontend.jar
WORKDIR /app
CMD java -jar PFG_Frontend.jar
```

Lo que refleja las líneas anteriores es que utiliza la imagen openjdk:17-slim como base, que contiene Java 17 para la ejecución de la aplicación; copia el archivo *jar* ejecutable generado a partir del proyecto; establece el directorio de trabajo; y ejecuta ese paquete ejecutable copiado anteriormente. De este modo, se crearían las dos imágenes de las dos partes de nuestro proyecto: el API REST y la interfaz gráfica. Así podemos ejecutar el comando:

```
#docker build -t pfg-frontend .
```

Los pasos definidos anteriormente habría que ejecutarlos tanto para el *backend* como para el *frontend*. Se ha ejemplificado con el *frontend* únicamente, pero se realiza de igual forma con el *backend*. No se añade para no repetir comandos análogos.

Finalmente, con las imágenes ya generadas, se procede a la ejecución completa del entorno mediante un fichero *docker-compose.yml*. Esto permite definir varios servicios que serán puestos en funcionamiento en contenedores independientes una vez haya sido ejecutado. Por lo tanto, se definen los siguientes servicios:

- **db-mariadb:** servicio que hace referencia a la imagen de Maria DB, que será utilizado para almacenar los datos de la aplicación. Las propiedades más importantes son las siguientes:

Tabla 55. Propiedades del servicio de Maria DB en docker-compose

Propiedad	Valor	Uso
hostname	db-mariadb	Establece el nombre de host del contenedor, para que otros servicios puedan hacer referencia a él
ports	3306:3306	Mapea los puertos entre el host de Docker y el contenedor
environment	-	Definen una serie de variables de entorno como el nombre de la base de datos y las credenciales del usuario por defecto
volumes	-	Monta un volumen para persistir el contenido de la base de datos

- **db-adminer:** servicio que hace referencia a la imagen de Adminer, cuyo uso es para gestionar la base de datos mediante la interfaz gráfica que este proporciona. Tiene que hacer referencia al hostname del contenedor de la base de datos para poder gestionarla.
- **pfg-frontend:** servicio que hace referencia a la imagen que se ha creado anteriormente del frontend.

Tabla 56. Propiedades del servicio de PFG-Vaadin en docker-compose

Propiedad	Valor	Uso
hostname	pfg-client-vaadin	Establece el nombre de host del contenedor, para que otros servicios puedan hacer referencia a él
ports	8080:8080	Mapea los puertos entre el host de Docker y el contenedor
environment	BACKEND_HOST: pfg-server-api	Define como variable de entorno el hostname y el puerto del contenedor del backend, para poder hacer peticiones HTTP adecuadamente.

	BACKEND_PORT: 8080 API_HERE_KEY: -	También define la clave del API Here para que pueda ser modificada de forma externa

- **pfg-backend:** servicio que hace referencia a la imagen que se ha creado anteriormente del backend. Debe hacer referencia al hostname del contenedor de la base de datos para poder conectarse a ella y hacer consultas.

Tabla 57. Propiedades del servicio API-REST en docker compose

Propiedad	Valor	Uso
hostname	pfg-server-api	Establece el nombre de host del contenedor, para que otros servicios puedan hacer referencia a él
ports	8081:8080	Mapea los puertos entre el host de Docker y el contenedor
environment	DB_HOST: db-mariadb DB_PORT: 3306	Define como variable de entorno el hostname del contenedor de la base de datos y el puerto, para poder hacer consultas

Se ha decidido no añadir el contenido completo del fichero *docker-compose.yml* para facilitar la lectura y la compresión del contenido principal. En su lugar, ha sido incorporado al anexo del documento para poder consultarlo.

Para concluir, se procede con la ejecución del fichero *docker-compose.yml* y se comprueba que los contenedores se ponen en funcionamiento de forma correcta.

5. Resultados

5.1 Pruebas unitarias

Estas pruebas son aquellas que permiten comprobar el correcto funcionamiento de pequeños fragmentos de código. De este modo, han sido realizadas durante la fase de codificación del sistema, de forma manual al depurar el código, y con la ayuda de herramientas como Postman. Se prueban todos los métodos con el fin de validar todas las funcionalidades, pero no se llegaron a implementar pruebas unitarias automáticas.

5.2 Pruebas de caja negra

Este tipo de tests se llevan a cabo para probar cómo se comporta un sistema, sin conocer cómo funciona internamente. Así, se realizaron pruebas para los distintos escenarios de cada uno de los casos de uso, resumiendo el resultado en la siguiente tabla.

Hay que aclarar que en las pruebas en las que hay que introducir datos, se han utilizado un caso por cada situación de error, para evitar solapamientos y facilitar la depuración.

Tabla 58. Resultados de la validación de pruebas

Nº	Prueba	Resultado
1	Registro de usuario para todos los roles	El usuario queda registrado con el rol indicado
2	Identificación de usuario para todos los roles	El usuario visualiza las vistas específicas para su rol
3	Ver perfil de usuario para todos los roles	El sistema muestra el perfil adecuadamente
4	Editar datos del perfil para todos los roles	Los datos son actualizados de forma correcta
5	Cargar foto de perfil	El sistema permite la carga de imágenes
6	Cargar foto de perfil con formatos no permitidos	El sistema impide cargar imágenes que no tienen el formato establecido
7	Cargar archivos a perfil	El sistema permite la carga de recursos multimedia
8	Cargar archivos no permitidos	El sistema impide la carga de recursos multimedia no admitidos
9	Ver ofertas	Se permite la visualización de ofertas
10	Filtrar ofertas	El sistema permite realizar un filtrado de ofertas
11	Desactivar ofertas	El sistema permite desactivar ofertas por parte de empresas y administradores
12	Eliminar ofertas	El sistema permite eliminar ofertas a empresas y administradores
13	Ver oferta perteneciente a otra empresa	El sistema impide la visualización de ofertas que no pertenecen al perfil propio de cada empresa
14	Editar oferta propia	Los datos son actualizados correctamente
15	Ver localización de la oferta	El sistema permite la visualización del mapa que muestra la localización

16	Guardar oferta a estudiante	El sistema permite guardar ofertas a los estudiantes
17	Eliminar oferta guardada a estudiante	El sistema permite eliminar la oferta guardada
18	Inscribirse a oferta por estudiante más de una vez	El sistema permite inscribirse a cada oferta por parte de los estudiantes, una sola vez
19	Eliminar oferta a empresas y administradores	El sistema permite eliminar ofertas tanto a empresas como a administradores
20	Ver solicitudes	El sistema permite visualizar las solicitudes
21	Editar solicitud	Los datos son actualizados correctamente
22	Eliminar solicitud	El sistema permite eliminar las solicitudes
23	Actualizar estado de solicitud por parte de empresas	El sistema permite actualizar el estado de las solicitudes a las empresas
24	Ver información de empresa	El sistema permite visualizar el perfil de las empresas
25	Ver información de empresa por otra empresa	El sistema no permite ver el perfil de otra empresa que no sea la propia
26	Ver información de estudiante	El sistema permite ver el perfil de estudiantes
27	Ver información de estudiante por otro estudiante	El sistema no permite ver el perfil de otro estudiante que no sea el propio
28	Crear chats	El sistema permite iniciar un nuevo chat
29	Ver chats	El sistema permite ver chats a los usuarios
30	Ver chat de otros perfiles	El sistema no permite ver chats que no pertenecen al propio perfil
31	Eliminar chats	El sistema permite eliminar los chats
32	Visualizar el contenido de todos los usuarios por administradores	El sistema permite a los administradores visualizar el contenido de todos los perfiles
33	Acceder a contenido de usuarios autenticados por parte de invitados	El sistema no permite a los invitados acceder a contenido de usuarios autenticados

Como se ha podido comprobar, se ha verificado el correcto funcionamiento de la aplicación por parte de una persona externa al proyecto, validando todas y cada una de las funcionalidades que ofrece cada rol de usuario. No se han apreciado errores en el funcionamiento aparentemente, por lo que se concluye el apartado de resultados de forma satisfactoria.

6. Presupuesto

6.1 Costes de hardware

Para el desarrollo del proyecto ha sido necesario un PC sin requisitos mínimos, pero con capacidad suficiente como para soportar un procesamiento adecuado de un entorno virtualizado corriendo varios contenedores de software, entre los que se encuentran un motor de base de datos y el propio servidor web.

De este modo, se ha utilizado un *Dell Precision Tower 6810*, adquirido por un valor de 550€. La vida estimada para este tipo de dispositivos es de unos 4 años, es decir, 48 meses. El proyecto duró 4 meses, y durante toda su duración se utilizó el dispositivo mencionado:

Tabla 59. Costes de hardware

Concepto	**Coste**
Dell Precision Tower 6810	(550€ / 48 meses) * 4 meses = 45,80€
TOTAL	**45,80€**

6.2 Costes de software

Todo el software adquirido para realizar el proyecto es gratuito, a excepción de Microsoft Word. El sistema operativo estaba instalado de fábrica en la computadora:

Tabla 60. Costes de software

Concepto	**Coste**
Microsoft Word 2016	59,99€ (Licencia Word 2016)
TOTAL	**59,99€**

Finalmente se utilizó una licencia de la ETSIST de la Universidad Politécnica de Madrid, por lo que se obtuvo gratuitamente, aunque se mantiene el concepto en el presupuesto.

6.3 Costes de personal

Los costes de personal son una parte fundamental en cualquier proyecto, representando los recursos humanos necesarios para su desarrollo y ejecución. En este caso específico, el personal involucrado abarca solamente un desarrollador cuyos costes son:

Tabla 61. Costes de personal

Concepto	Coste
Desarrollador web	(5 [h/día] * 5 [días] * 16 [semanas] * 15 [€/h]) = 6.000,00€
TOTAL	**6.000,00€**

De esta manera, se logra reconocer el valor del tiempo y el talento dedicados por el desarrollador, contribuyendo a apreciar el impacto humano en el éxito de la aplicación web y la satisfacción de las necesidades del cliente.

6.4 Costes totales

A continuación, se muestra el coste total del proyecto a partir del sumatorio de los costes anteriores:

Tabla 62. Costes totales

Concepto	Coste
Costes de hardware	45,80€
Costes de software	59,99€
Costes de personal	6.000€
TOTAL	**6.105,79€**

7. Manual de usuario

Este manual se ha dividido en dos secciones: Manual de instalación, destinado a los administradores de la aplicación, y Manual de uso, destinado a los propios usuarios.

El primero de ellos proporciona las instrucciones necesarias para instalar y configurar la aplicación en un entorno de trabajo, y el segundo, de cómo utilizar la aplicación, detallando cómo navegar entre las vistas y las diferentes funcionalidades.

7.1 Manual de instalación

Para poner en funcionamiento la aplicación, es necesario cumplir con unos requisitos previos:

- Tener instalado Docker en la máquina local.
- Tener instalado Docker Compose, para poder ejecutar aplicaciones multi-contenedor.
- Tener instalado *make*, para poder ejecutar el script automatizado.
- Tener instalado Maven, para poder empaquetar la aplicación con todas sus dependencias en un *jar* ejecutable.
- Tener instalado *git*, para poder clonar el proyecto.

Así, los pasos de instalación son los siguientes:

1. Abrir un terminal, navegar hasta el directorio que se desee y clonar el repositorio de GitHub mediante el comando:

   ```
   >git clone https://github.com/adrianBBdev/PFG-Jobs4StudentsApp.git
   ```

2. Navegar hasta el directorio donde hemos clonado el proyecto y ejecutar el script que automatiza el empaquetamiento de la aplicación y la generación de las imágenes necesarias para ejecutarlas en conteneodres:

   ```
   >make build
   ```

3. Ejecutar el fichero *docker-compose.yml*, para poner en funcionamiento todos los contenedores que conforman la aplicación, mediante el comando:

   ```
   >docker-compose up -d
   ```

4. Verificar que todos los contenedores que se precisan en el fichero anterior están en ejecución, mediante el comando:

   ```
   >docker ps
   ```

5. Acceder a la aplicación web mediante el navegador, la cual se encuentra en la ruta: http://[IP-acceso]:8080, donde en el caso que nos concierne, la IP de acceso se corresponde con localhost o 127.0.0.1, visualizando lo siguiente:

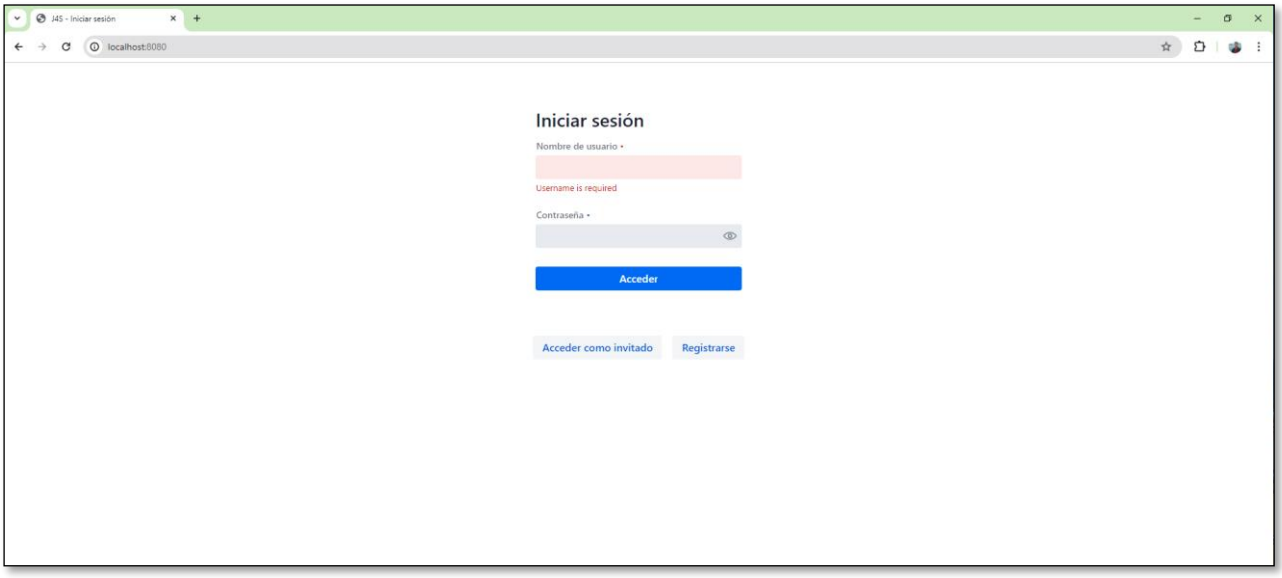

Ilustración 18. Página de inicio de la aplicación web.

7.2 Manual de uso de la aplicación

Este manual va dirigido a los propios usuarios que van a hacer uso de la aplicación, una vez el sistema ya está disponible. Por tanto, el primer paso es acceder a un navegador e introducir la URL de acceso: http://[IP-acceso]:8080, donde la IP de acceso será la dirección IP del servidor donde esté ejecutándose.

7.2.1 Inicio de sesión

La pantalla de inicio de sesión es el primer punto de contacto que el usuario tiene con la aplicación. Al acceder a ella desde el navegador, este se encuentra con la consiguiente vista, que le va a permitir lo siguiente:

- Iniciar sesión introduciendo sus credenciales.
- Acceder al formulario de registro.
- Acceder a la aplicación como invitado, sin previo registro, pudiendo acceder al contenido que ofrece la aplicación web con ciertas limitaciones.

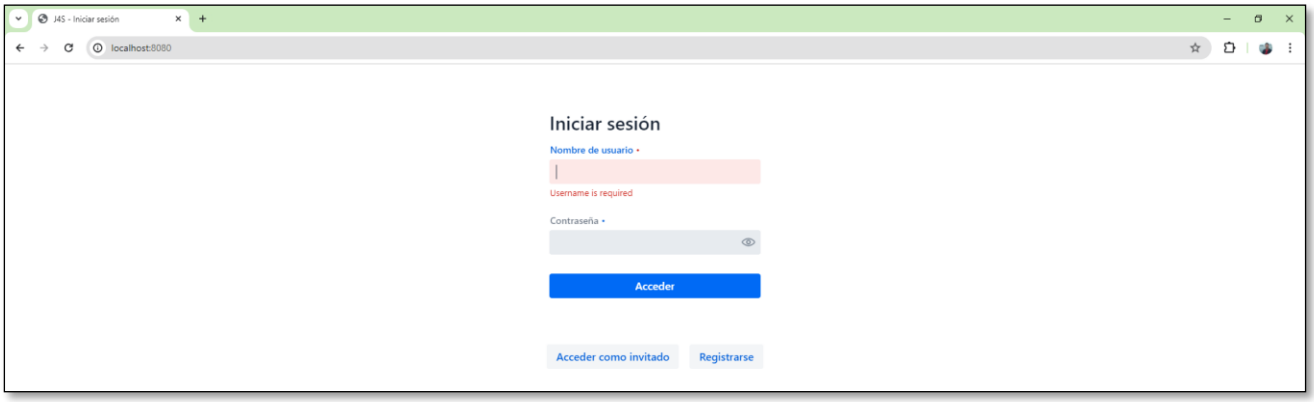

Ilustración 19. Captura de la vista de inicio de sesión

7.2.2 Registro de usuarios

Una vez pulsada la opción de registrarse, el usuario accede al formulario donde tiene que introducir sus credenciales y datos de perfil:

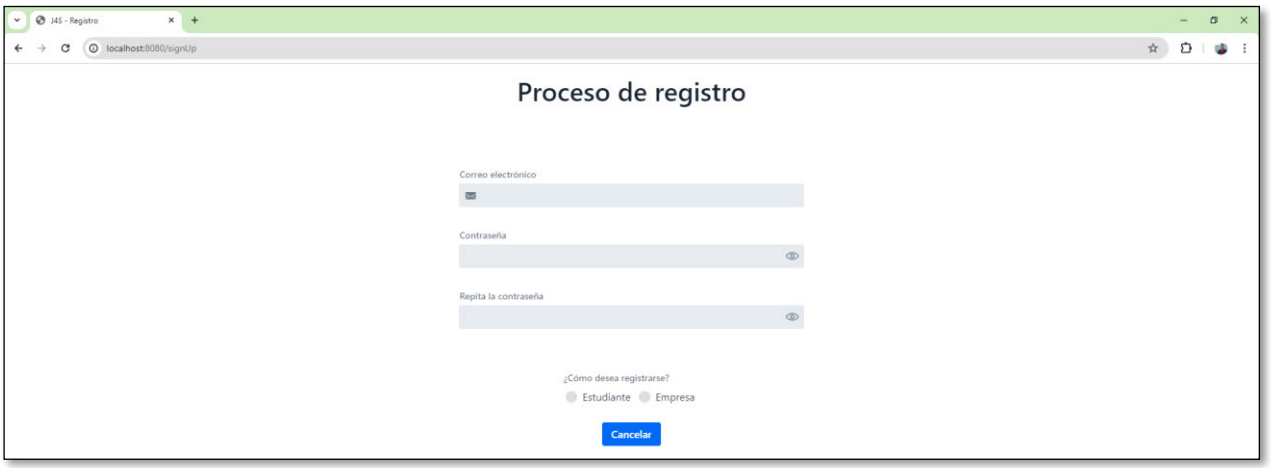

Ilustración 20. Captura de la vista de registro de usuarios (I)

Concretamente, el usuario debe:

- Rellenar los campos de credenciales.
- Seleccionar el tipo de usuario.
- Rellenar el resto de los datos del perfil, condicionado por la selección del tipo de usuario seleccionado previamente.

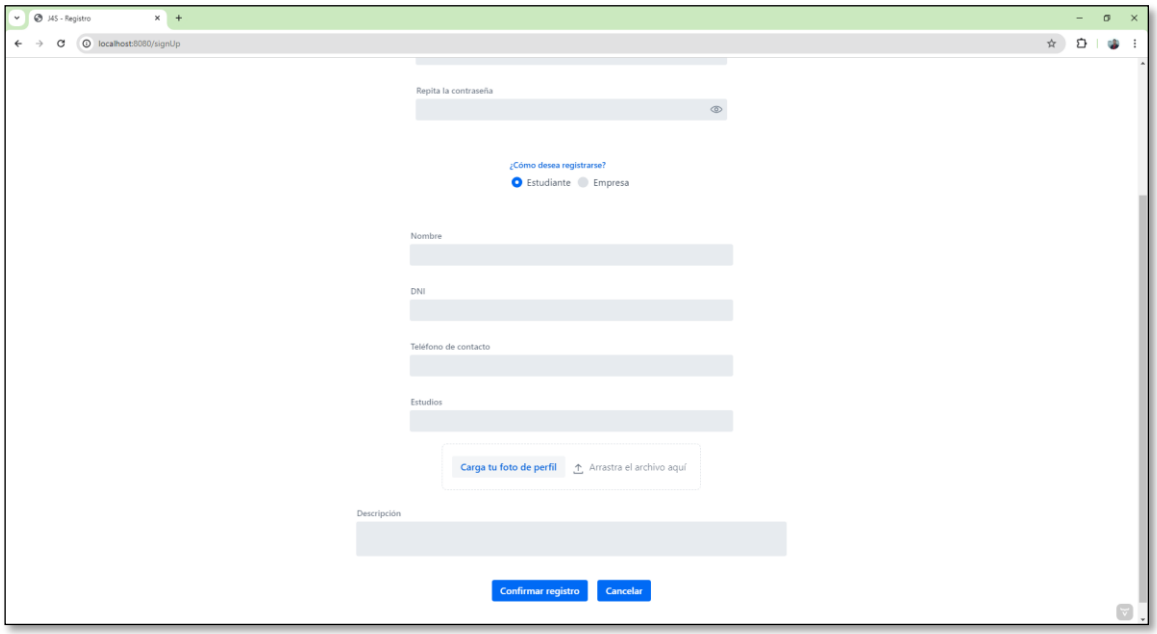

Ilustración 21. Captura de la vista de registro de usuarios (II)

De este modo, si el usuario no rellena correctamente todos los campos, no podrá darse de alta en la aplicación. Adicionalmente, dependiendo de la opción que escoja, entre *Estudiante* y *Empresa*, podrá visualizar los campos que debe rellenar, pues son distintos dependiendo del rol que se seleccione.

7.2.3 Acceso de invitados

Al acceder como invitado, el usuario puede acceder al contenido de la aplicación, pero de forma restringida:

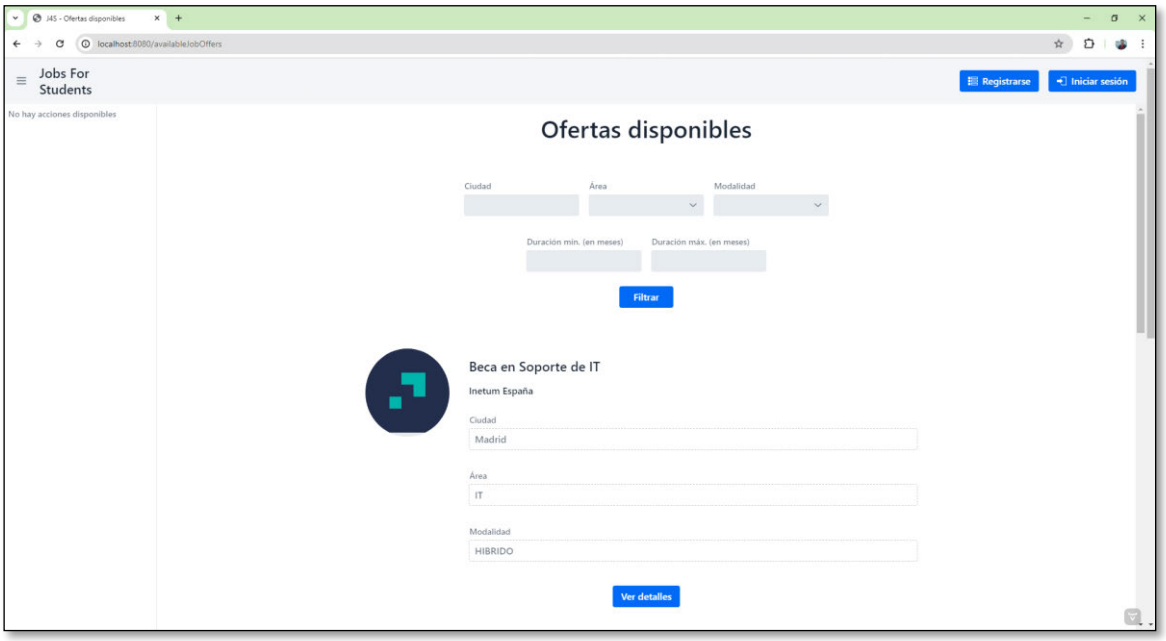

Ilustración 22. Captura de la vista de acceso como invitado

Como se puede observar en la Ilustración 6, la barra de navegación no posee acciones disponibles a realizar, y la vista se reduce a la visualización de las ofertas y sus detalles, para que pueda explorar cómo funciona y qué opciones se ofrecen en la aplicación.

7.2.4 Acceso con credenciales

Al iniciar sesión con credenciales, el usuario podrá acceder a la totalidad del contenido. Sin embargo, la vista del usuario cambiará dependiendo del rol que posea:

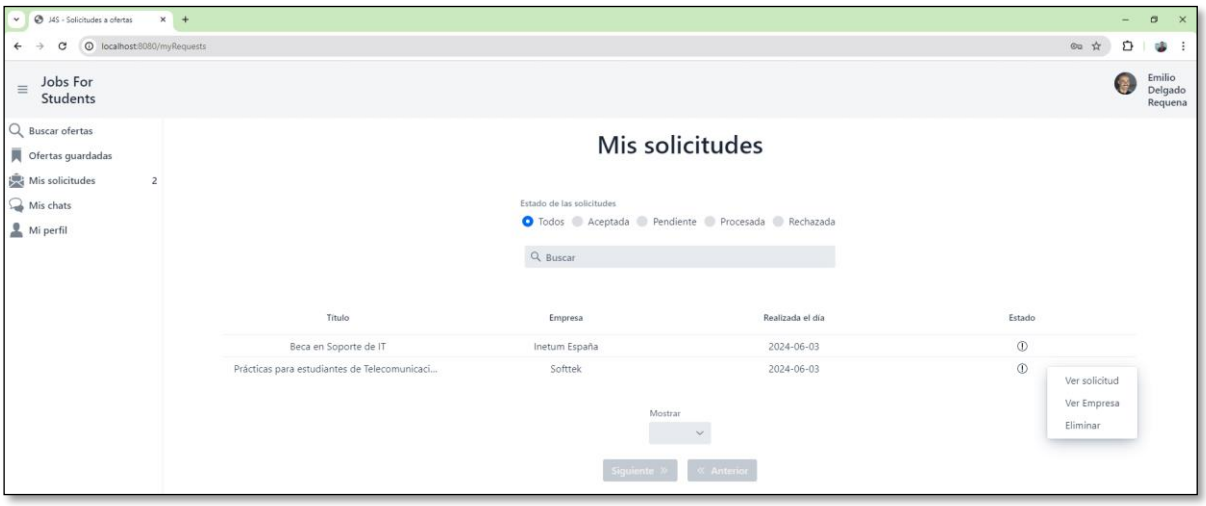

Ilustración 23. Captura de la vista de acceso como estudiante

En este primer caso, a un usuario con perfil de estudiante se le redirige a la página de solicitudes, donde puede ver las inscripciones a ofertas que ya ha realizado y donde se muestran los detalles de éstas. Adicionalmente, tiene la posibilidad de filtrar tanto por el estado de las solicitudes como por el nombre de empresa o de la oferta.

Respecto a la barra de navegación, ya si se pueden ver las acciones que a este tipo de usuario se le permite llevar a cabo, y en la parte superior, aparece su avatar personalizado con su foto de perfil, desde donde puede acceder a su perfil o cerrar la sesión actual y salir.

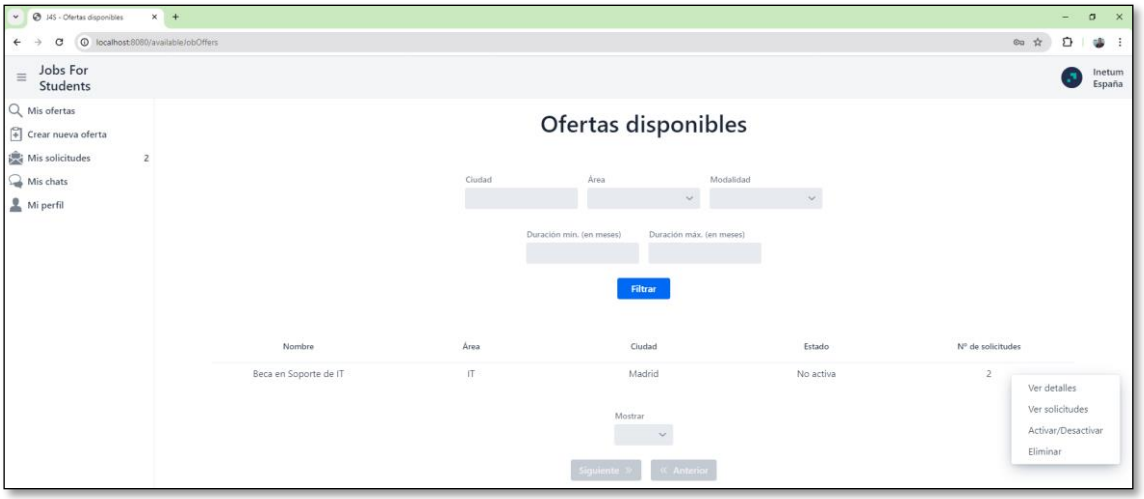

Ilustración 24. Captura de la vista de acceso como empresa

Por consiguiente, un usuario con rol de empresa, lo que podrá ver en un principio será el listado de sus propias ofertas, con sus detalles y posibles filtros para buscar entre ellas. Al igual que el estudiante,

la barra de navegación también dispone de las acciones que puede realizar este tipo de usuario, y una barra superior con el avatar y el submenú de opciones.

Y por último, si inicia sesión un usuario administrador, podrá visualizar lo siguiente:

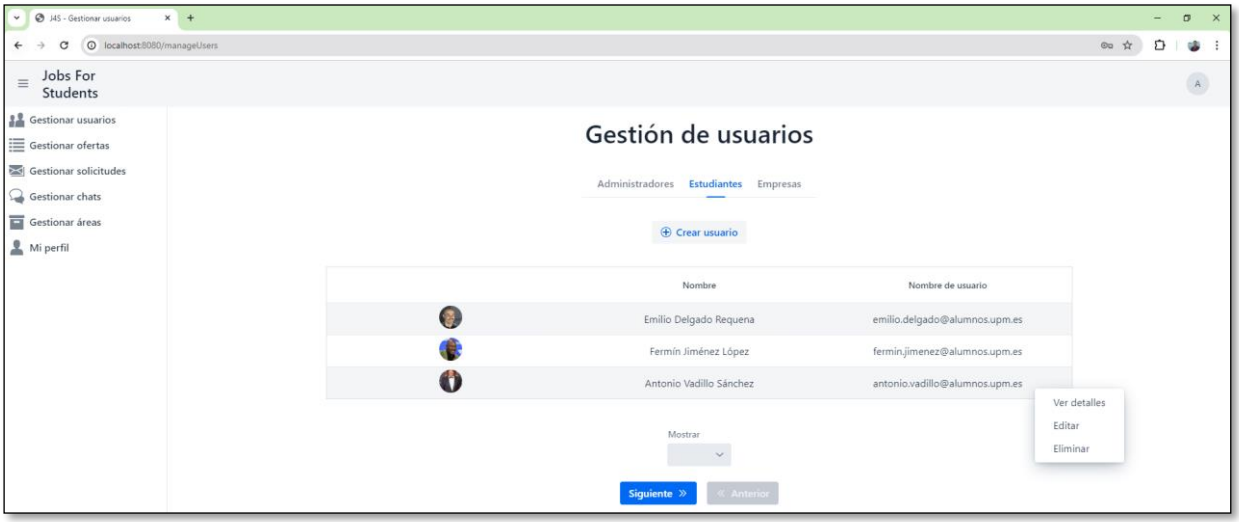

Ilustración 25. Captura de la vista de acceso como administrador

Este tipo de usuario accede directamente a la gestión de los todos los usuarios, pudiendo realizar cualquier tipo de acción sobre ellos, con el correspondiente listado, y clasificado por el rol que tienen asignado. Por otra parte, en la barra de navegación visualiza las acciones específicas que se le permiten, y en la parte superior, el avatar ya no contiene ninguna imagen, sino la letra inicial de su rol.

7.2.5 Vistas adicionales

El resto de las vistas se adjuntan en el anexo del documento, para ejemplificar las acciones que tiene disponibles cada tipo de usuario.

Asimismo, también se debe señalar que la aplicación web se ha desarrollado para que la interfaz gráfica sea híbrida, es decir, que sea accesible tanto desde PCs como desde dispositivos móviles:

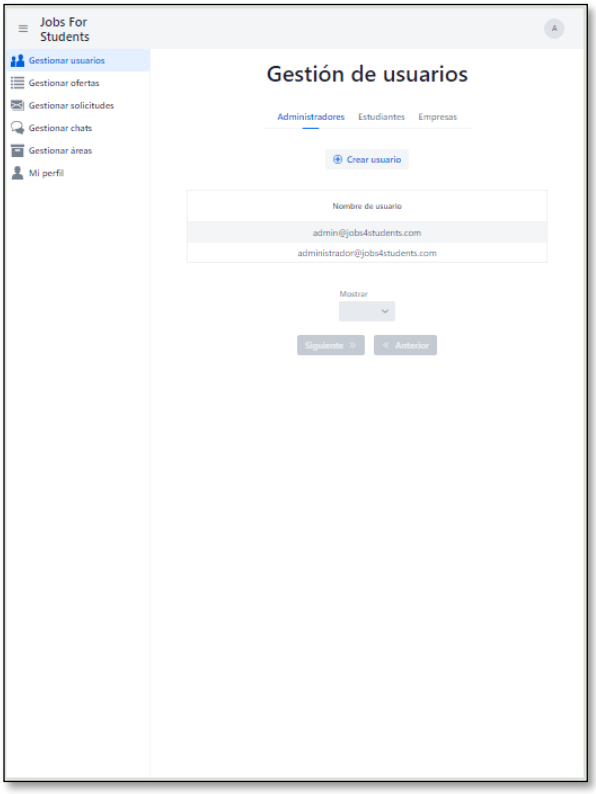

Ilustración 26. Ejemplo I de vista para dispositivos móviles

Como se puede visualizar en la imagen superior, el contenido de la vista se ajusta a las dimensiones de la pantalla y a las características del dispositivo, garantizando al usuario la accesibilidad universal. A continuación, se muestra otro ejemplo, con la barra de navegación cerrada:

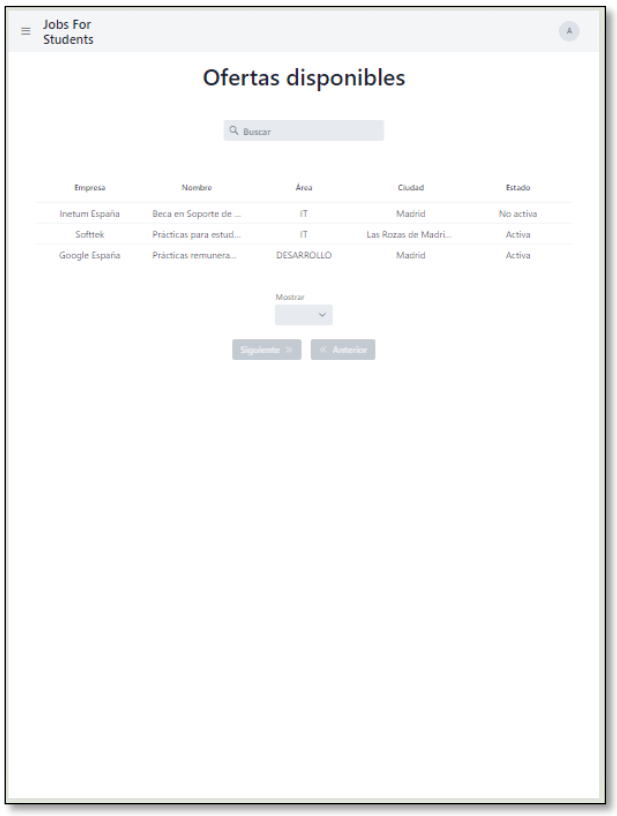

Ilustración 27. Ejemplo II de vista desde dispositivos móviles

8. Impacto del proyecto

La evaluación del impacto de un proyecto es fundamental para comprender su alcance y aportación, más allá de la confección de la parte técnica. En consecuencia, en este apartado se ha explorado el impacto que la aplicación web ha tenido y puede tener en diferentes áreas, centrándose en las implicaciones de ámbito general que afectan a la sociedad:

- **Impacto social:** esta plataforma tiene el potencial de impactar positivamente en la sociedad al facilitar la inserción laboral de estudiantes. Al conectar a estudiantes con oportunidades de becas y empleo, contribuye a mejorar su futuro profesional y su calidad de vida.
- **Impacto en educación:** al proporcionar a los estudiantes acceso a becas y oportunidades de trabajo relacionadas con su campo de estudio, esta plataforma apoya el desarrollo educativo de los estudiantes al tiempo que fomenta el aprendizaje continuo y la adquisición de habilidades relevantes para el mercado laboral.
- **Impacto económico:** favorecer la inserción laboral de estudiantes puede tener un impacto económico significativo al ayudar a reducir la tasa de empleo juvenil y fomentar el crecimiento económico al conectar a empresas con talento joven y motivado.

- **Impacto tecnológico:** el uso de tecnologías como *Spring-Boot*, *Vaadin*, *MariaDB* y *Docker* demuestra un enfoque innovador y eficiente para el desarrollo del proyecto. Esto puede inspirar a otros proyectos a adoptar tecnologías similares y promover el avance tecnológico de la industria.
- **Impacto ambiental:** este proyecto contribuye de forma indirecta a la reducción del uso de recursos físicos y energéticos. Al proporcionar una plataforma digital para la búsqueda de oportunidades laborales y educativas, se pueden evitar desplazamientos innecesarios y generación de residuos asociados, como el papel. Además, al fomentar el teletrabajo y la colaboración remota, la plataforma puede ayudar a disminuir las emisiones de gases de efecto invernadero relacionadas con los desplazamientos diarios al lugar de trabajo. Esto promueve un modelo de trabajo más sostenible y contribuye a la mitigación del cambio climático y la preservación del medio ambiente.

Por otro lado, en cuanto a los Objetivos de Desarrollo Sostenible (ODS), este proyecto participa activamente en el impulso de los siguientes propósitos:

- **ODS 4 – Educación de calidad:** al facilitar el acceso a oportunidades educativas y laborales para estudiantes, se contribuye a promover una educación inclusiva, equitativa y de calidad para todos.
- **ODS 8 – Trabajo decente y crecimiento económico**: al poner en contacto a estudiantes con oportunidades de trabajo, se contribuye a promover el crecimiento económico constante, inclusivo y sostenible, así como el empleo pleno y productivo para todos.
- **ODS 9 – Industria, innovación e infraestructura:** el uso de tecnologías modernas y eficientes en el desarrollo de la plataforma pone de manifiesto el empleo de un enfoque innovador para abordar los desafíos de inserción laboral. Esto puede contribuir al desarrollo de infraestructuras resilientes, capaces de asumir con flexibilidad situaciones contraproducentes, con el fin de promover la industrialización inclusiva y fomentar la innovación.

En resumen, el impacto de este proyecto no solo acentúa su relevancia tecnológica, sino también su potencial transformador en diversos ámbitos. Esto incluye desde el impulso de la inserción laboral y el desarrollo educativo de los estudiantes hasta la promoción de un crecimiento económico inclusivo y la adopción de soluciones innovadoras y sostenibles. De este modo, se demuestra el compromiso con la creación de un futuro más equitativo, próspero y sostenible para todos.

9. Conclusiones

9.1 Conclusiones generales

Una vez se ha completado el desarrollo del sistema, se puede ratificar que se han cumplido los objetivos fijados inicialmente:

- Se ha desarrollado la aplicación enteramente con Java.
- Se ha puesto a prueba el desarrollo de interfaces gráficas con el entorno Vaadin.
- Se ha construido una interfaz gráfica que permite a los usuarios gestionar fácilmente sus perfiles, ofertas, solicitudes y chats.
- La aplicación permite al administrador gestionar el sistema completamente.
- Se ha adoptado un nivel de seguridad mínimo, permitiendo validar las acciones de los usuarios e identificarles en el sistema.
- La aplicación ha sido validada por un cliente externo.

Ahora bien, como análisis final de la solución, se aprecian una serie de puntos positivos y desventajas. Primordialmente, se pone de manifiesto el atractivo del uso de un marco basado únicamente en Java, que ofrece una integración simple con Spring-Boot, y que garantiza una curva de aprendizaje relativamente pequeña, especialmente si ya se ha trabajado con Swing o JavaFX. Vaadin permite crear interfaces de usuario coherentes y personalizables, aunque también es cierto que se deben conocer sus limitaciones y evaluar si es la mejor opción para tu proyecto. Se convierte en una opción viable, sobre todo, para proyectos que requieren una solución rápida y funcional.

En comparación con otras tecnologías que utilizan otros lenguajes, como Angular o JavaScript, la curva de aprendizaje varía, dependiendo de si se tiene conocimiento o no de JavaScript, o se parte de simplemente de Java. No obstante, probablemente proporcionen más posibilidades y comodidad a la hora de la creación de la propia interfaz.

De este modo, el dictamen final se basa en el siguiente resultado:

Tabla 63. Comparación de tecnologías.

Característica	Vaadin	Hilla	Angular	Mejor tecnología
Lenguaje de programación	Java	Java	TypeScript	Vaadin/Hilla
Unificación lenguaje con backend	Sí	Sí, pero se apoya en TypeScript y React	No	Vaadin
Atractivo visual	Bajo	Bajo	Alto	Angular

Integración con Spring-Boot	Integración directa	Integración directa	Integración indirecta	Vaadin/Hilla
Uso generalizado	Bajo	Bajo	Alto	-
Desarrollo de aplicaciones móviles	No	No	Sí	Angular
Curva de aprendizaje	Baja	Baja	Alta	Vaadin/Hilla
Personalización	Fácil	Fácil	Fácil	-
Escalabilidad	Baja	Alta	Alta	Angular

Lo que se puede inferir de la tabla anterior, es que si se busca una integración directa con Spring-Boot y un aprendizaje rápido para saber cómo manejar la tecnología en cuestión, Vaadin es una alternativa factible. Permite crear una interfaz práctica en poco tiempo y sin mucha complejidad para el programador, sin la necesidad de tener que aprender un lenguaje de programación distinto al empleado en la lógica de negocio. Sin embargo, si lo que se necesita es una mayor flexibilidad y escalabilidad, con un amplio abanico de posibilidades en el entorno gráfico, una solución notablemente conocida como Angular u otras tecnologías pueden ser una mejor opción, pero que conlleva un aprendizaje especializado.

A pesar de todo ello, el propósito general del proyecto era la construcción de una aplicación web que facilitase el contacto estudiante-empresa. Así pues, el resultado obtenido es un portal web desarrollado con Vaadin, en el que se pueden llevar a cabo todas las acciones establecidas en la fase de diseño, cumpliendo con los requisitos iniciales.

9.2 Líneas de trabajo futuras

Se podría destacar como una mejora significativa del proyecto la incorporación del protocolo HTTPS con el fin garantizar una comunicación segura mediante la encriptación de los datos, lo que previene diversos tipos de ciberataques como la suplantación de identidad.

Asimismo, se puede mencionar la creación de ficheros de log internos del servidor donde se almacene información de los usuarios relativa a los inicios de sesión y actividad dentro de la aplicación. De este modo, los administradores y desarrolladores tienen una mayor visibilidad acerca de posibles errores de ejecución de la aplicación.

En cuanto a la interfaz gráfica, se puede enriquecer el contenido visual a través del uso de una gama de colores más amplia o de fuentes de texto más complejas.

También se puede incluir una aceptación de más tipos de recursos multimedia para cargar en la plataforma. Actualmente, solo se pueden seleccionar imágenes JPEG o PNG, documentos PDF o vídeos MP4 o MOV. Si se aumenta este listado, se ofrecen más posibilidades a los usuarios.

Y respecto a funcionalidades concretas, mejorar la presentación de la localización de las ofertas de trabajo, añadiendo la opción de poder ver la mejor ruta desde el domicilio a la localización de la oferta en el mapa, lo que resultaría práctico para el usuario.

10. Referencias

[1] Spring. Spring Boot. Online: https://spring.io/projects/spring-boot. Consultado en febrero, 2024.

[2] Softtek. Autenticación de APIs basada en tokens con Spring y JWT. Online: https://blog.softtek.com/es/autenticando-apis-con-spring-y-jwt. Consultado en marzo, 2024.

[3] Arquitectura Java. ¿Qué es Spring Boot? Online: https://www.arquitecturajava.com/que-es-spring-boot/. Consultado en febrero, 2024.

[4] CampusMVP. ¿Qué son Spring framework y Spring Boot? Online: https://www.campusmvp.es/recursos/post/que-son-spring-framework-y-spring-boot-tu-primer-programa-java-con-este-framework.aspx. Consultado en marzo, 2024.

[5] Docker. Docker overview. Online: https://docs.docker.com/get-started/overview/. Consultado en marzo, 2024.

[6] Sysarmy – El blog de quienes dan soporte. ¿Qué es Docker y que soluciones brinda? Online: https://sysarmy.com/blog/posts/que-es-docker/. Consultado en marzo, 2024.

[7] Red Hat. ¿Qué es Docker y cómo funciona? Online: https://www.redhat.com/es/topics/containers/what-is-docker. Consultado en marzo, 2024.

[8] Vaadin. Vaadin Docs. Online: https://vaadin.com/docs/latest/ . Consultado en abril, 2024.

[9] Naciones Unidas. Objetivos y metas de Desarrollo sostenible. Online: https://www.un.org/sustainabledevelopment/es/objetivos-de-desarrollo-sostenible/ . Consultado en febrero, 2024.

[10] Hostinger. ¿Qué es git y cómo empezar a usarlo? Online: https://www.hostinger.es/tutoriales/que-es-github . Consultado en febrero, 2024.

[11] Here. Here Geocoding & Search. Online: https://www.here.com/platform/geocoding. Consultado en mayo, 2024.

11. Anexo

11.1 Anexo 1 – Fichero *docker-compose.yml*

```
version: '3.7'

services:

  db-mariadb:
    image: mariadb
    hostname: db-mariadb
    container_name: db-mariadb
    restart: always
    ports:
      - 3306:3306
    environment:
      MARIADB_ROOT_PASSWORD: root
      MARIADB_USER: admin
      MARIADB_PASSWORD: admin
      MARIADB_DATABASE: application_data
    volumes:
      - ./.db:/var/lib/mysql

  db-adminer:
    image: adminer
    container_name: db-adminer
    restart: always
    environment:
      ADMINER_DEFAULT_SERVER: db-mariadb
    ports:
      - 9000:8080
    depends_on:
      - db-mariadb

  pfg-frontend:
    image: pfg-frontend
    container_name: pfg-frontend
    hostname: pfg-client-vaadin
    restart: always
    ports:
      - 8080:8080
    environment:
      BACKEND_HOST: pfg-server-api
      BACKEND_PORT: 8080
  pfg-backend:
```

```
image: pfg-backend
container_name: pfg-backend
hostname: pfg-server-api
restart: always
ports:
  - 8081:8080
environment:
   DB_HOST: db-mariadb
   DB_PORT: 3306
```

11.2 Anexo 2 – Vistas adicionales

- Vistas comunes a todos los usuarios:

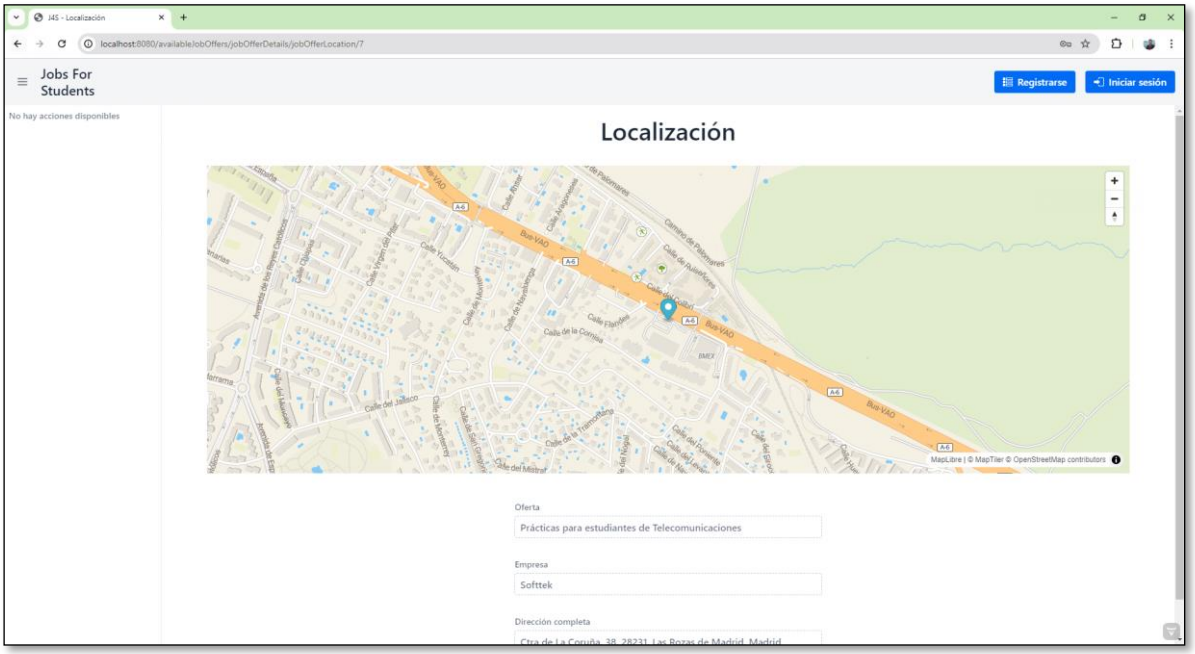

Ilustración 28. Captura de la vista de localización de una oferta

- Vistas de usuarios con rol de invitado:

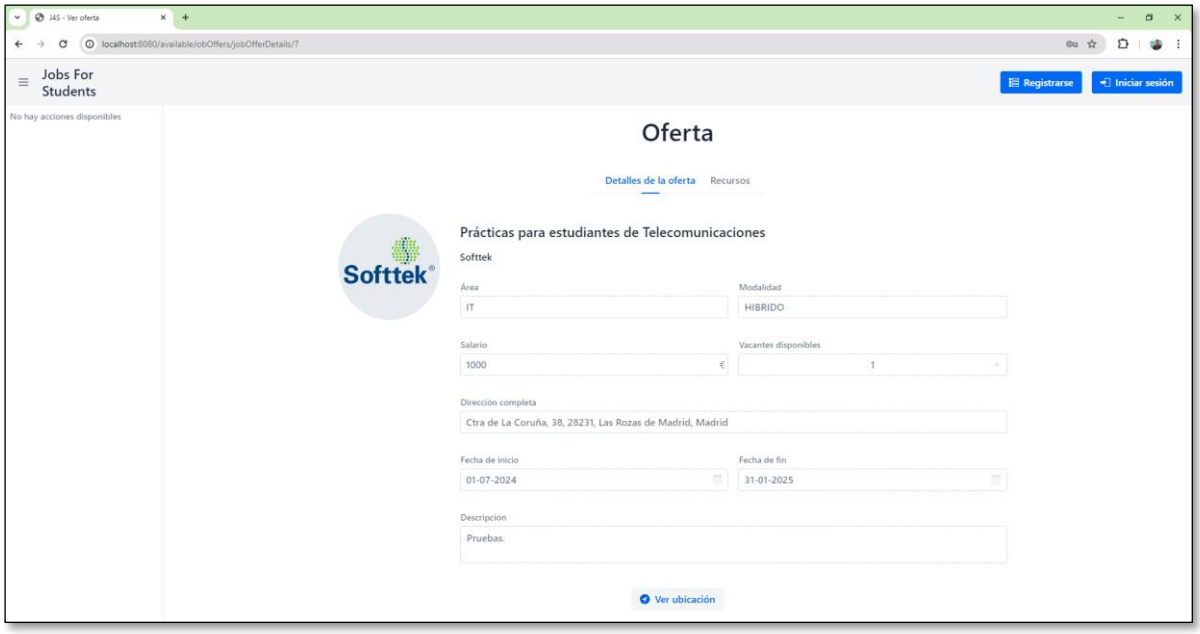

Ilustración 29. Captura de la vista de ver detalles de una oferta (invitado)

- Vistas de usuarios autenticados:

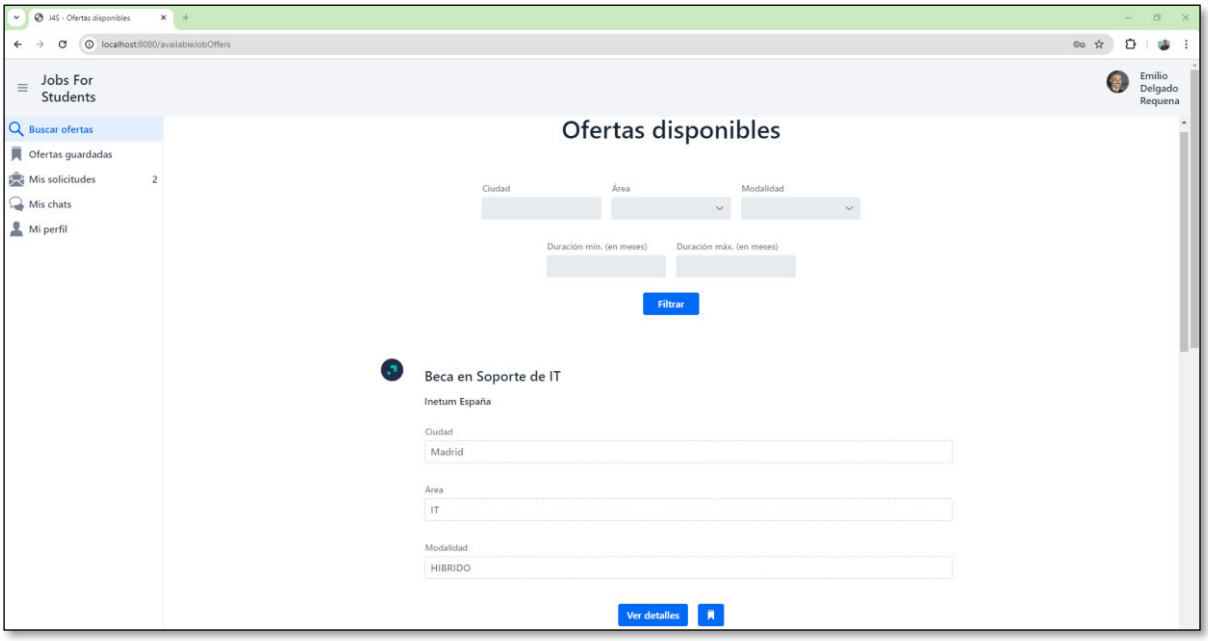

Ilustración 30. Captura de la vista de buscar ofertas (estudiante)

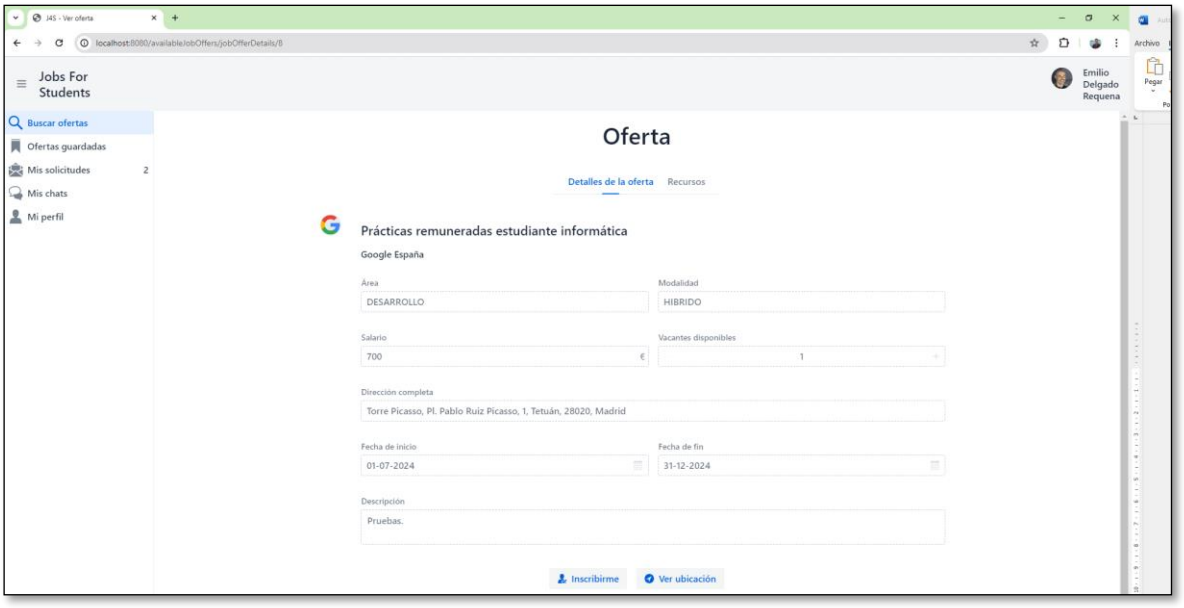

Ilustración 31. Captura de la vista de detalles de una oferta (estudiante)

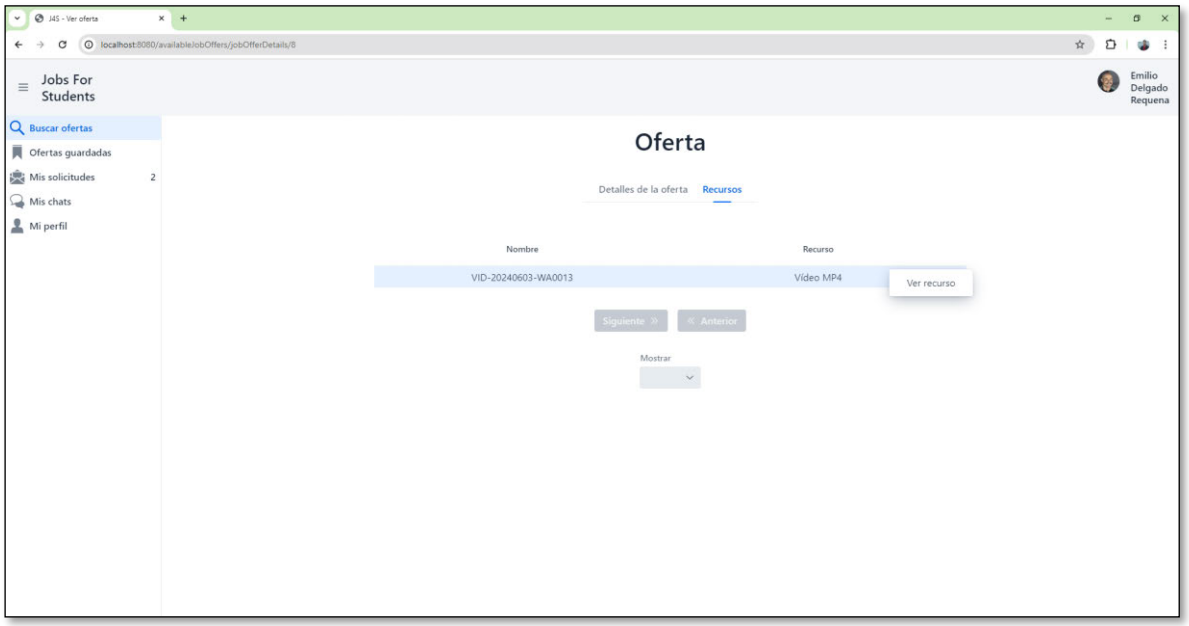

Ilustración 32. Captura de la vista de recursos de una oferta (estudiante)

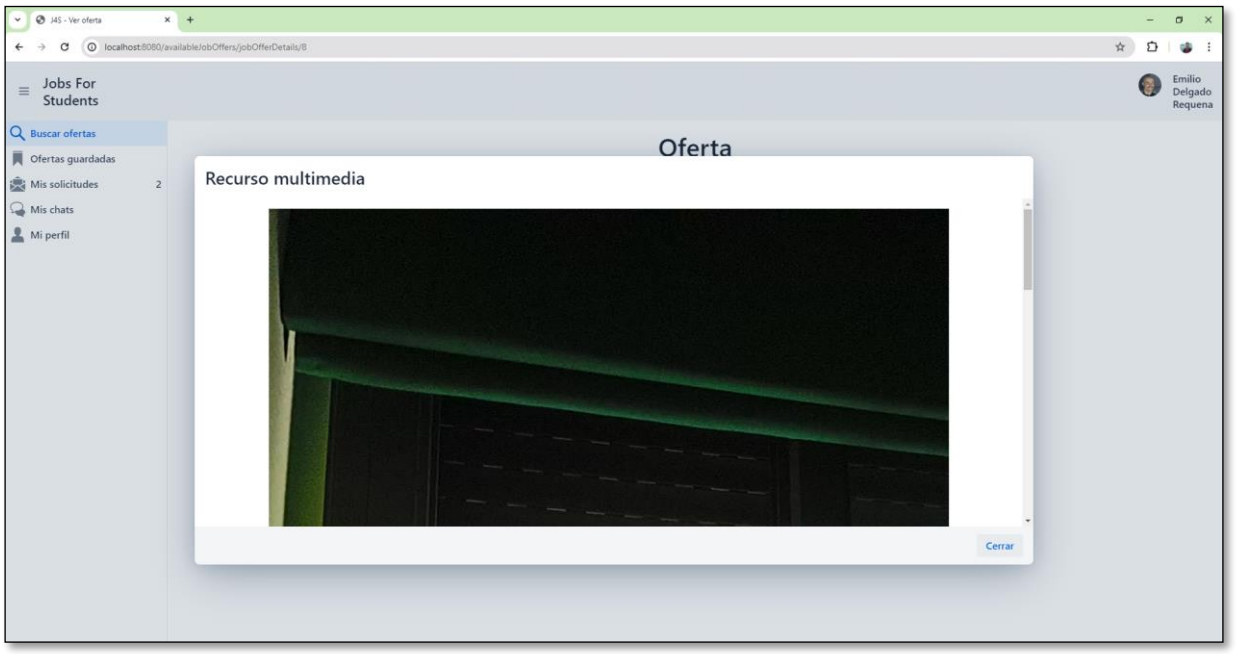

Ilustración 33. Captura de la vista de visualización de recurso multimedia

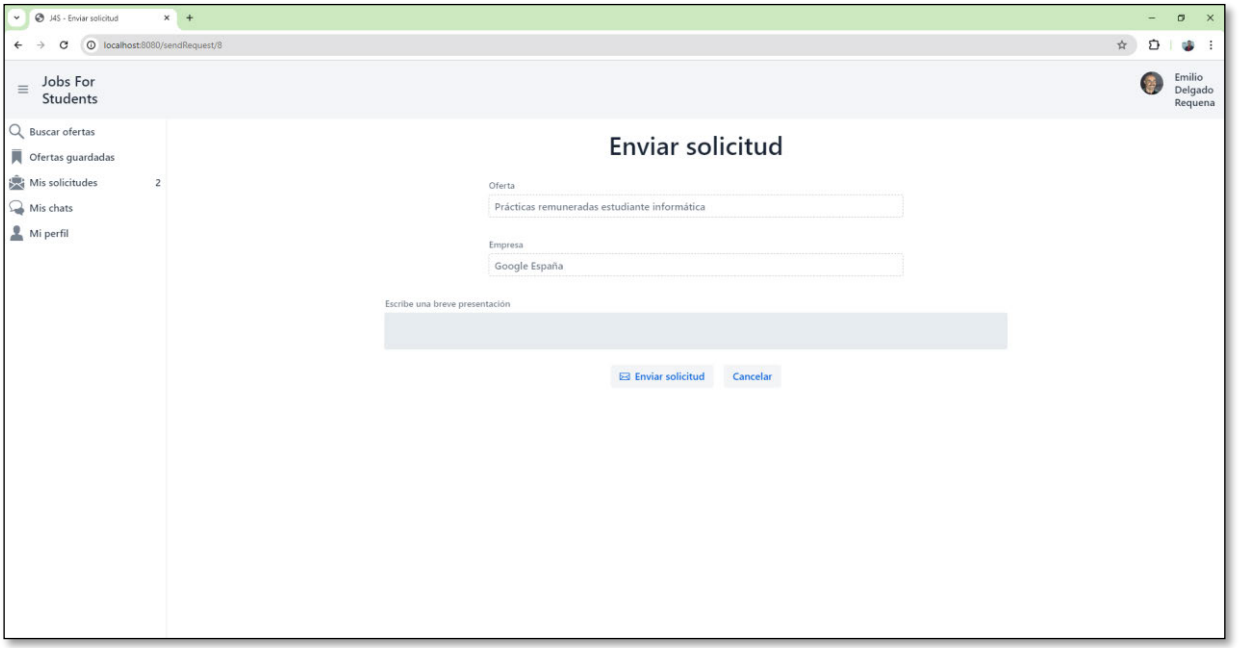

Ilustración 34. Captura de la vista de enviar solicitud (estudiante)

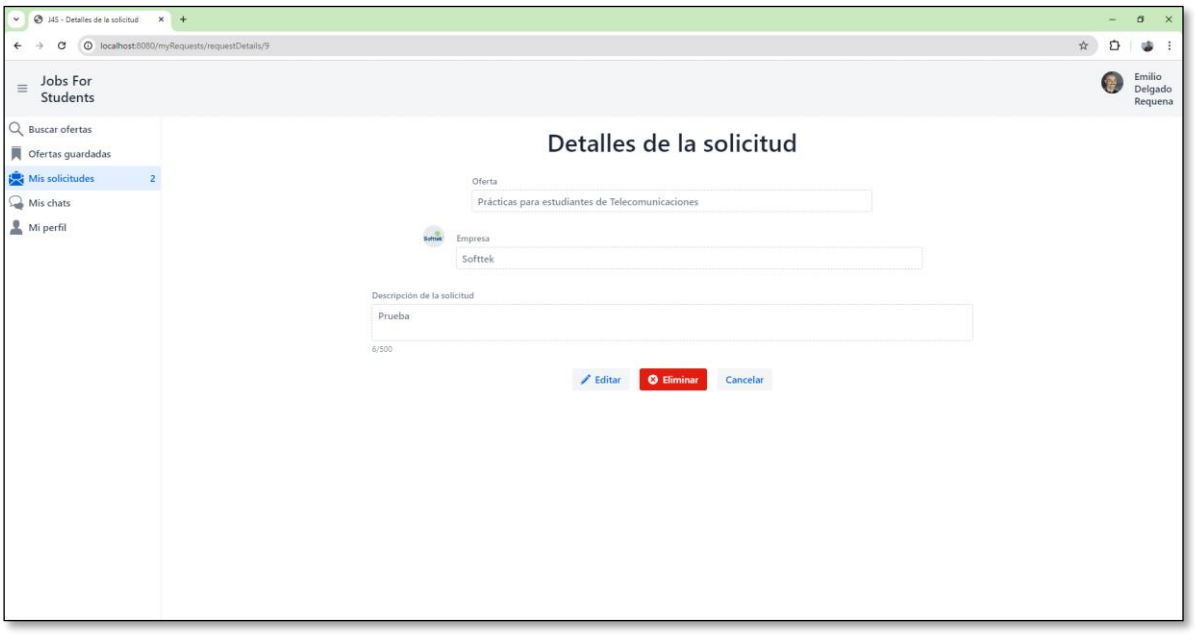

Ilustración 35. Captura de la vista de ver solicitud (estudiante)

Ilustración 36. Captura de la vista de ofertas guardadas (estudiante)

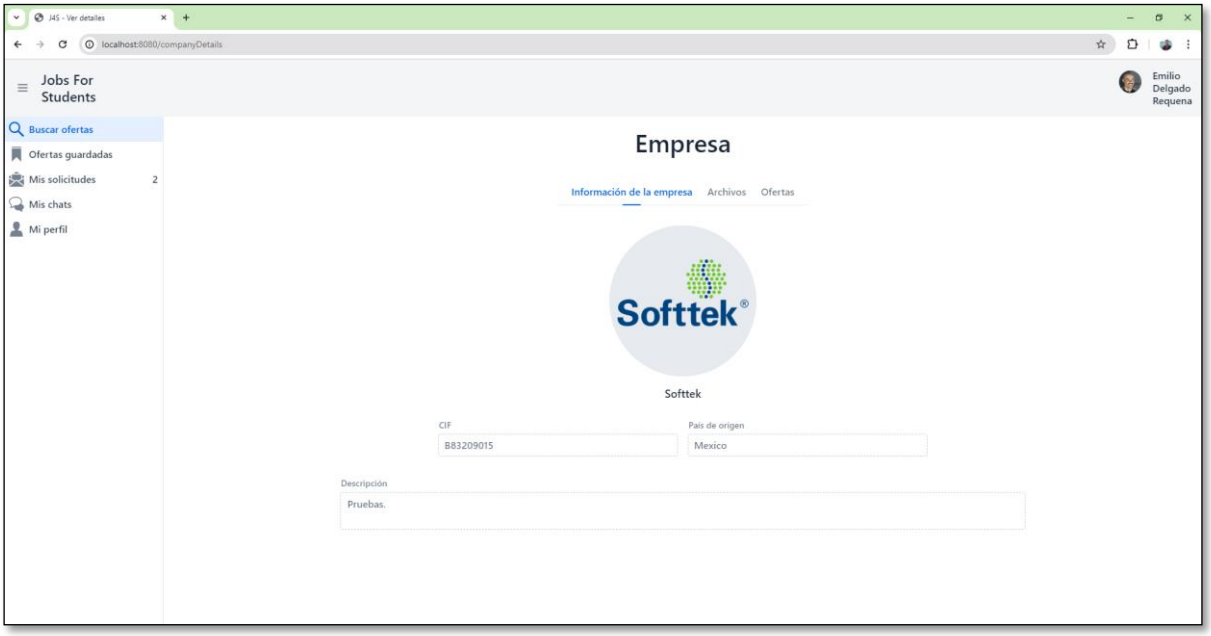

Ilustración 37. Captura de la vista de ver empresa

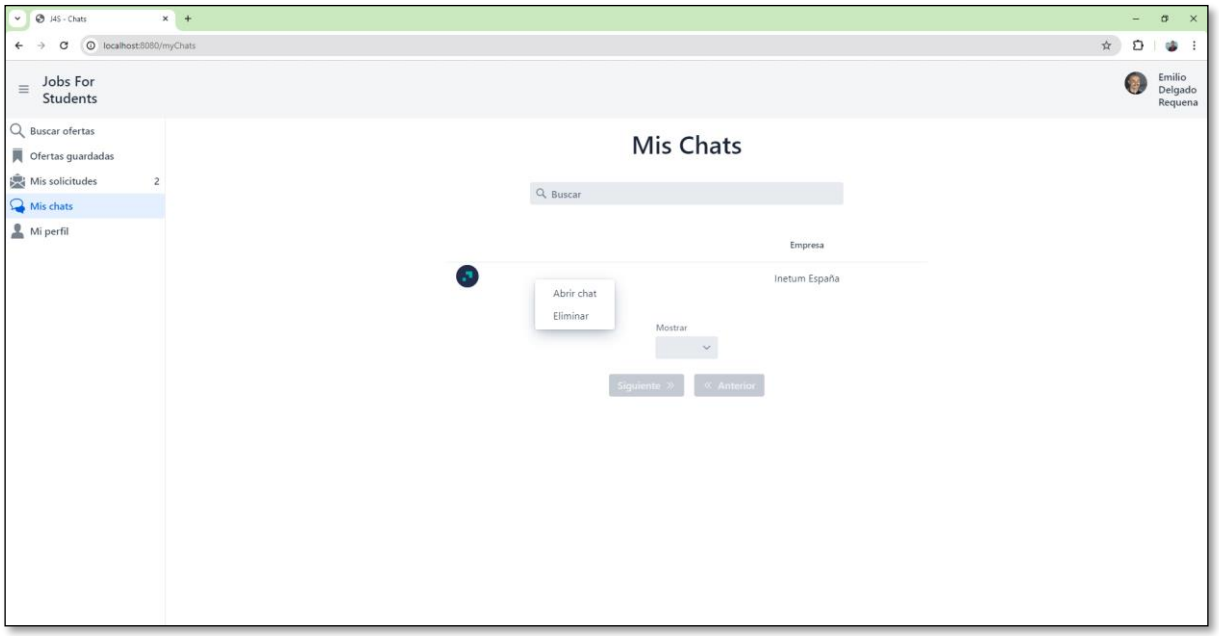

Ilustración 38. Captura de la vista de chats

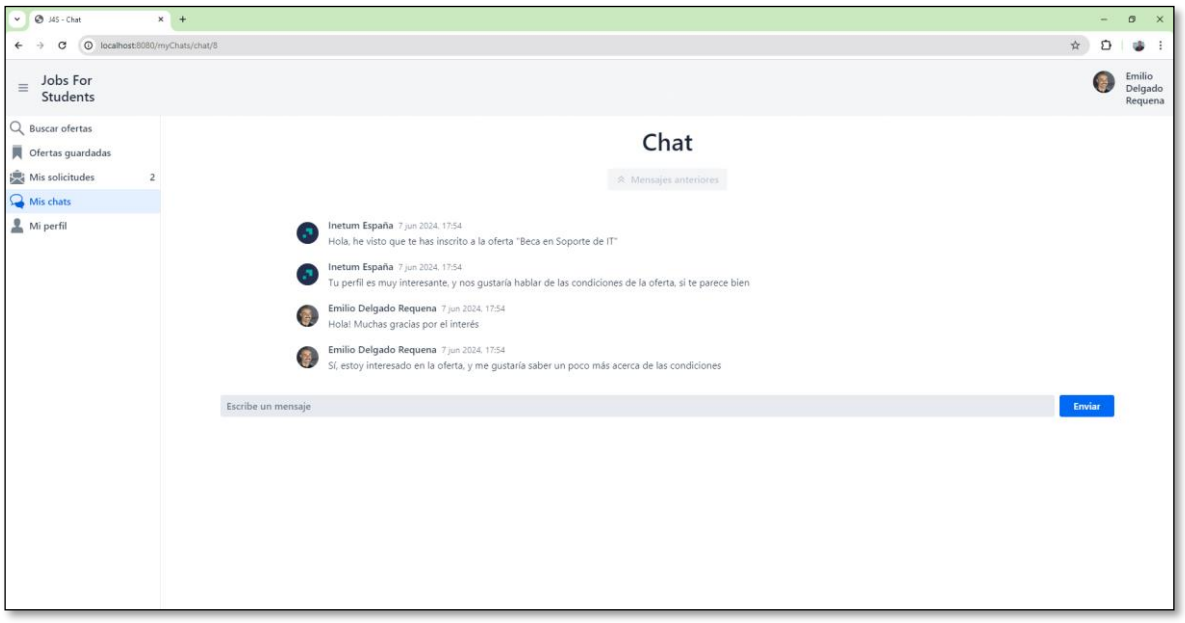

Ilustración 39. Captura de la vista de chat (estudiante y empresa)

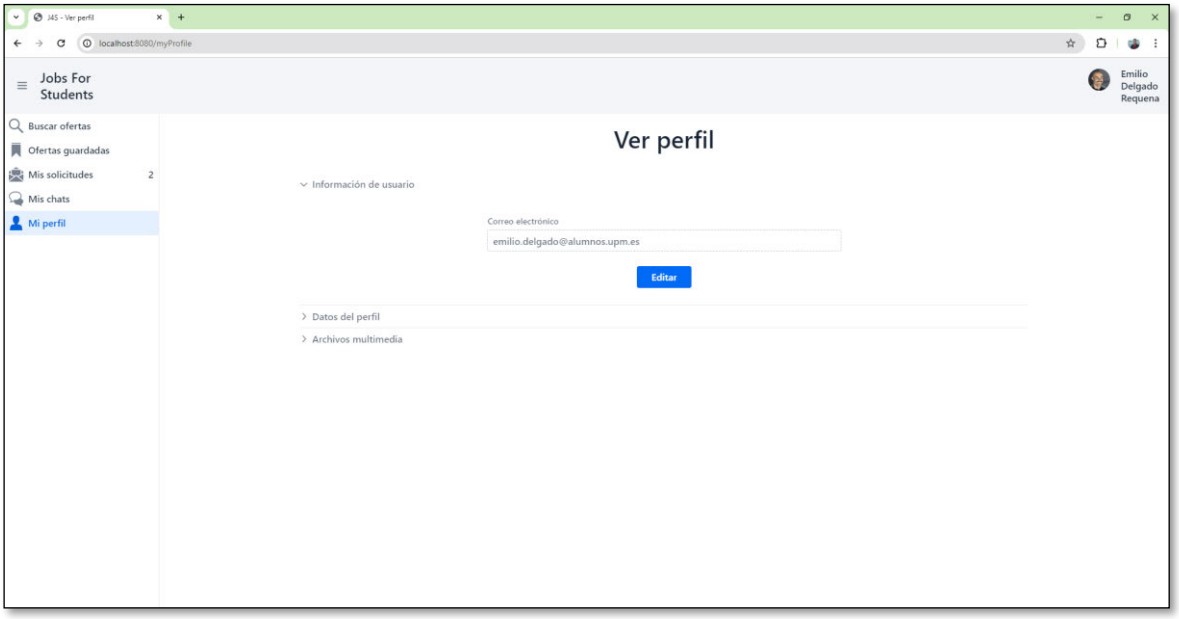

Ilustración 40. Captura de la vista de ver perfil

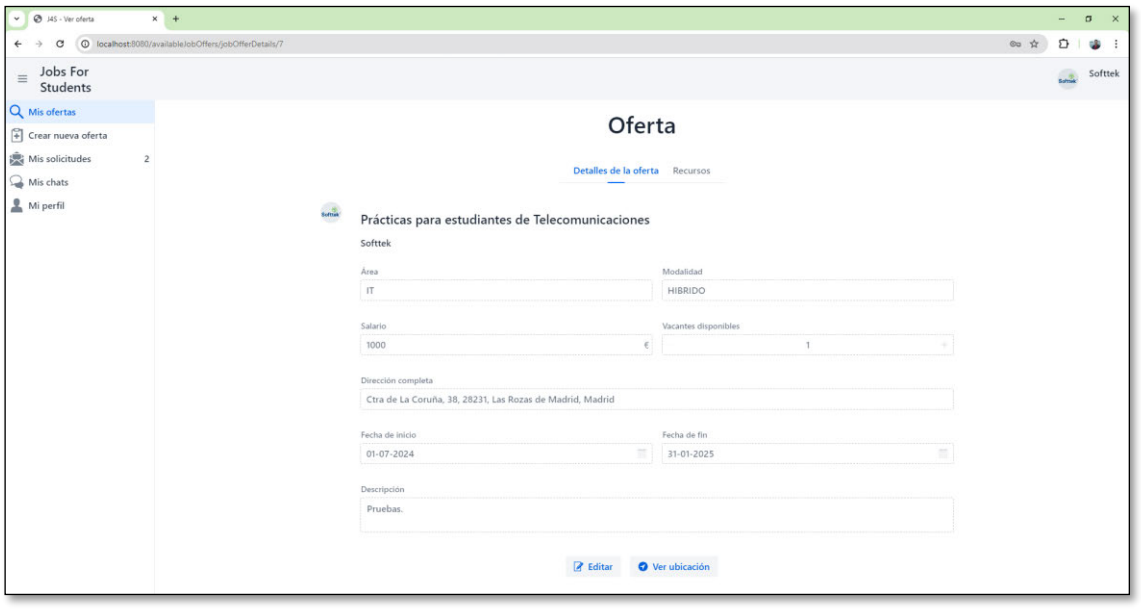

Ilustración 41. Captura de la vista de detalles oferta (empresa) I

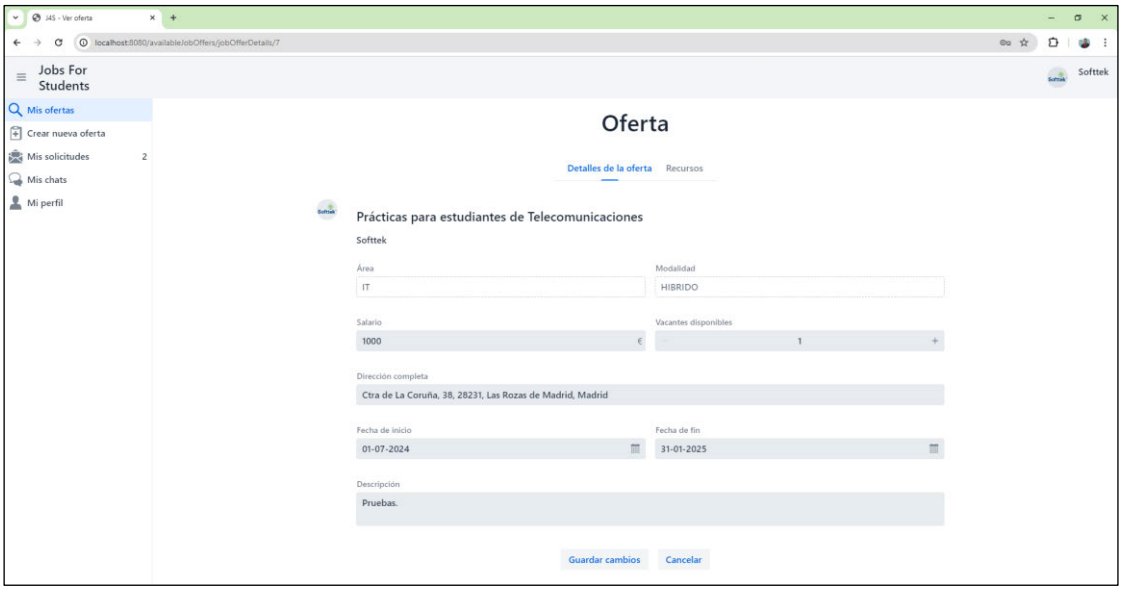

Ilustración 42. Captura de la vista de detalles de oferta (empresa) II

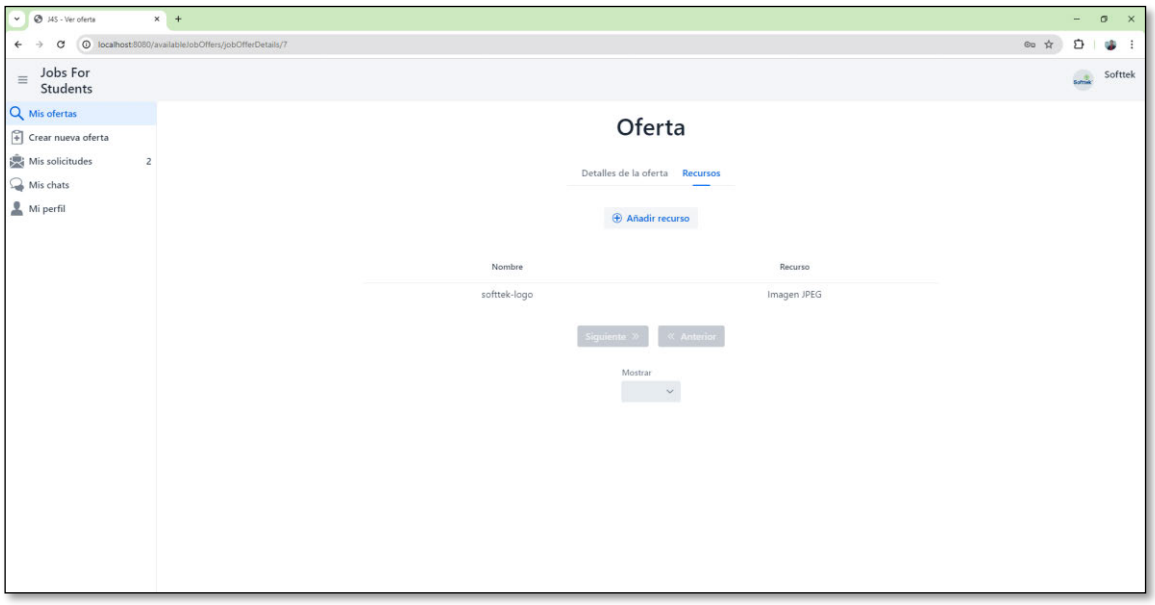

Ilustración 43. Captura de la vista de detalles de oferta (empresa) III

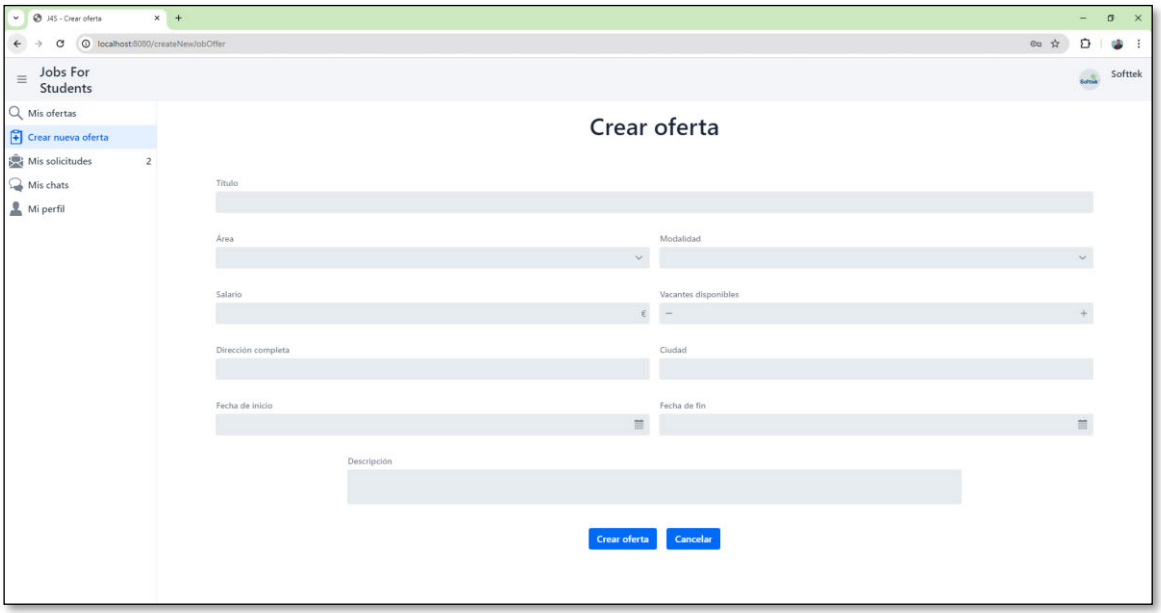

Ilustración 44. Captura de la vista de crear oferta (empresa)

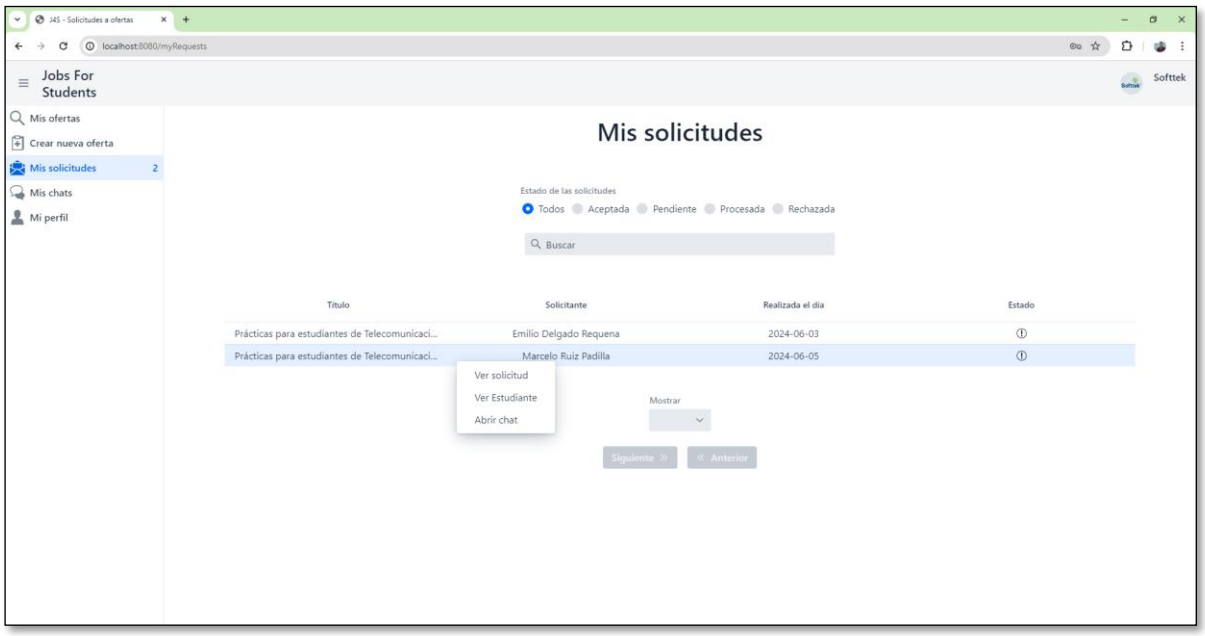

Ilustración 45. Captura de la vista de ver solicitudes (empresa)

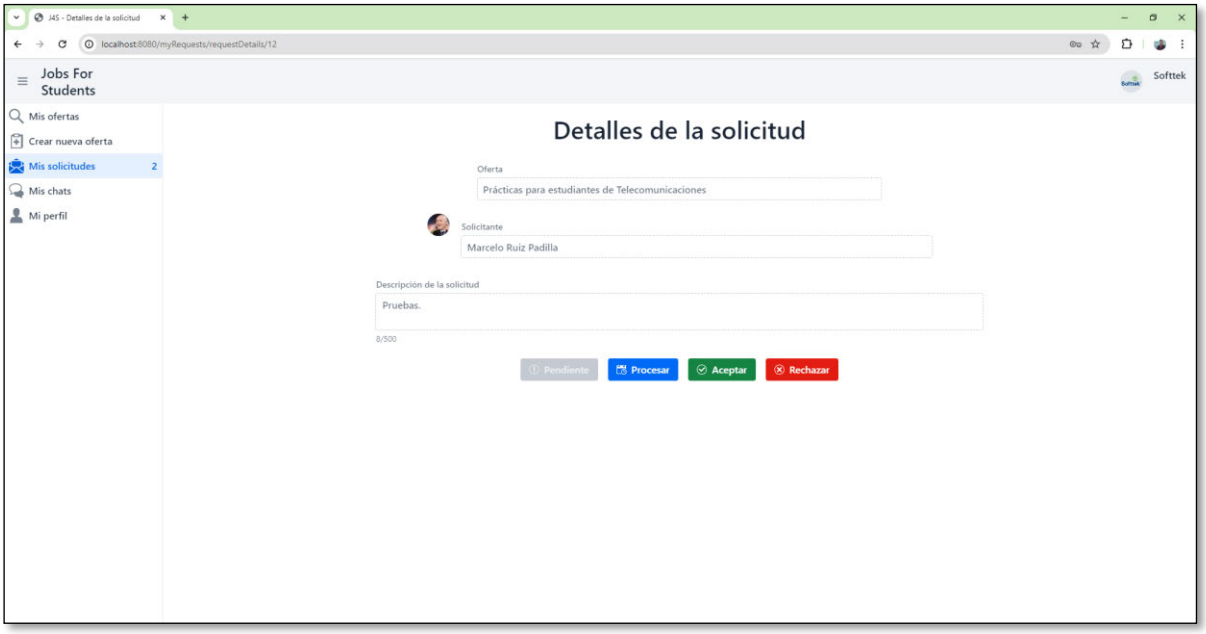

Ilustración 46. Captura de la vista de ver solicitud (empresa)

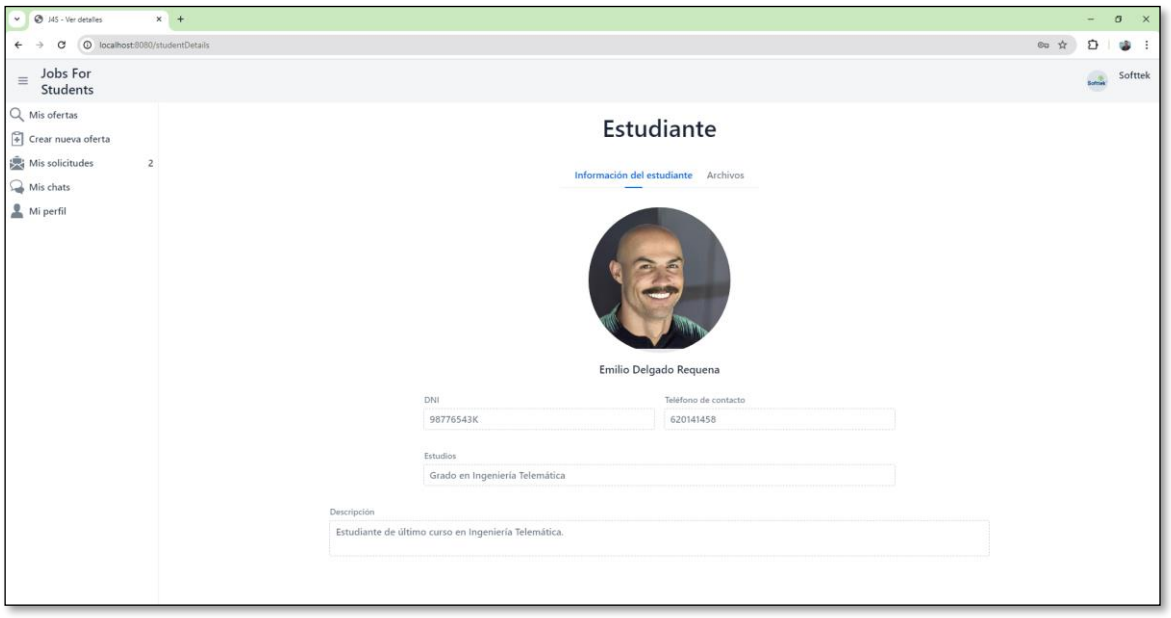

Ilustración 47. Captura de la vista de ver estudiante

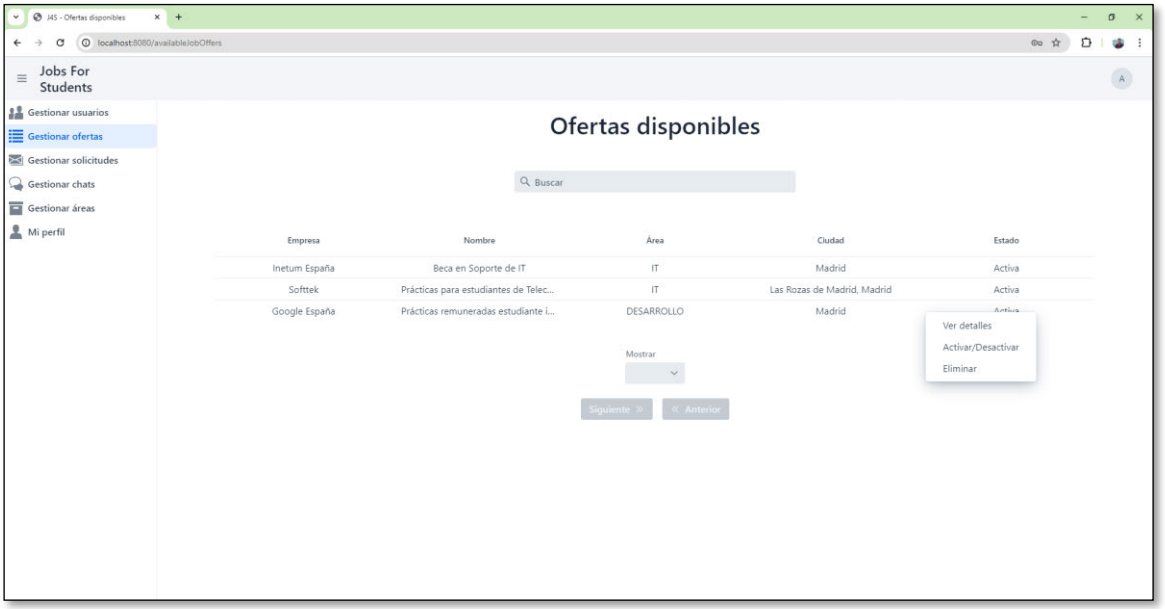

Ilustración 48. Captura de la vista de gestionar ofertas (administrador)

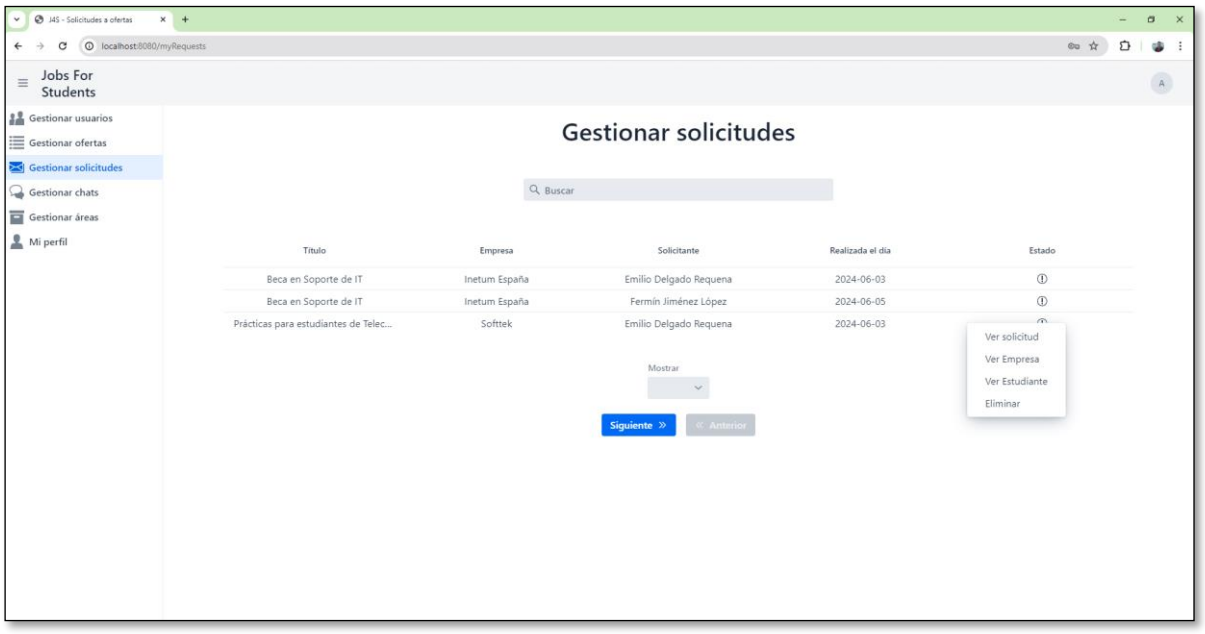

Ilustración 49. Captura de la vista de gestionar solicitudes (administrador)

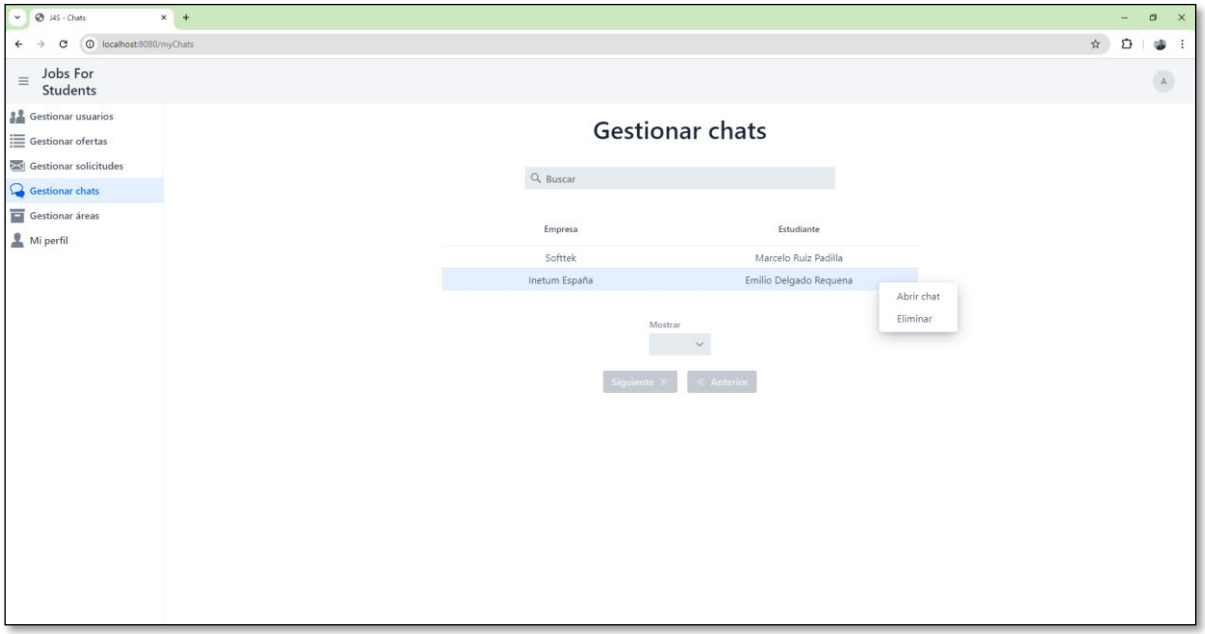

Ilustración 50. Captura de la vista de gestionar chats (administrador)

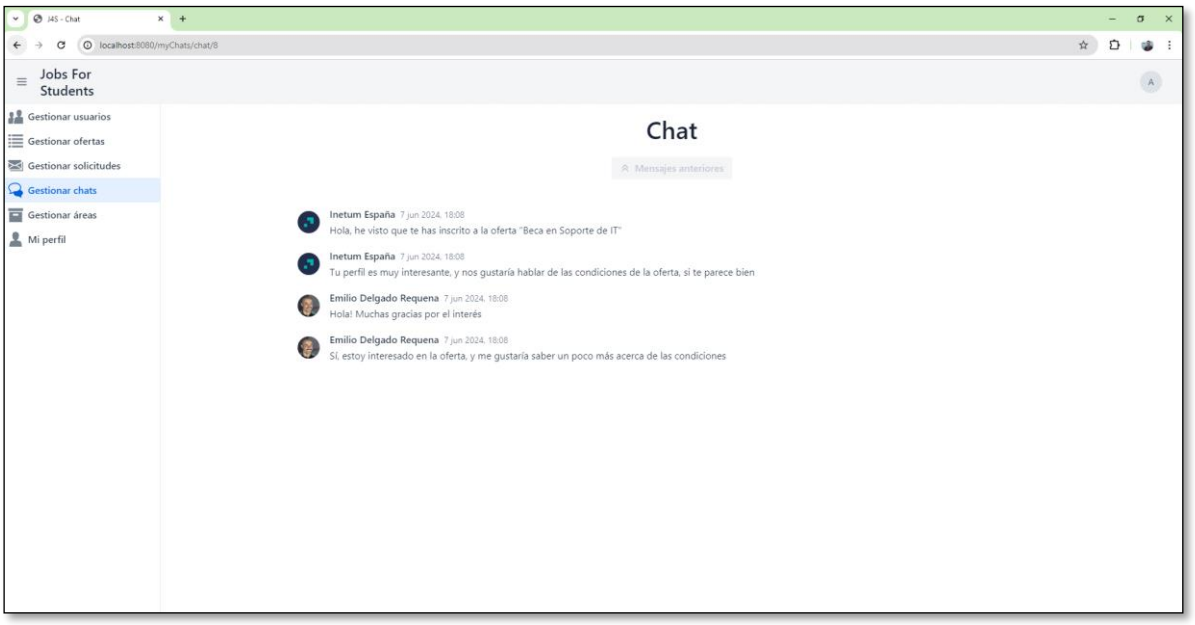

Ilustración 51. Captura de la vista de ver chat (administrador)

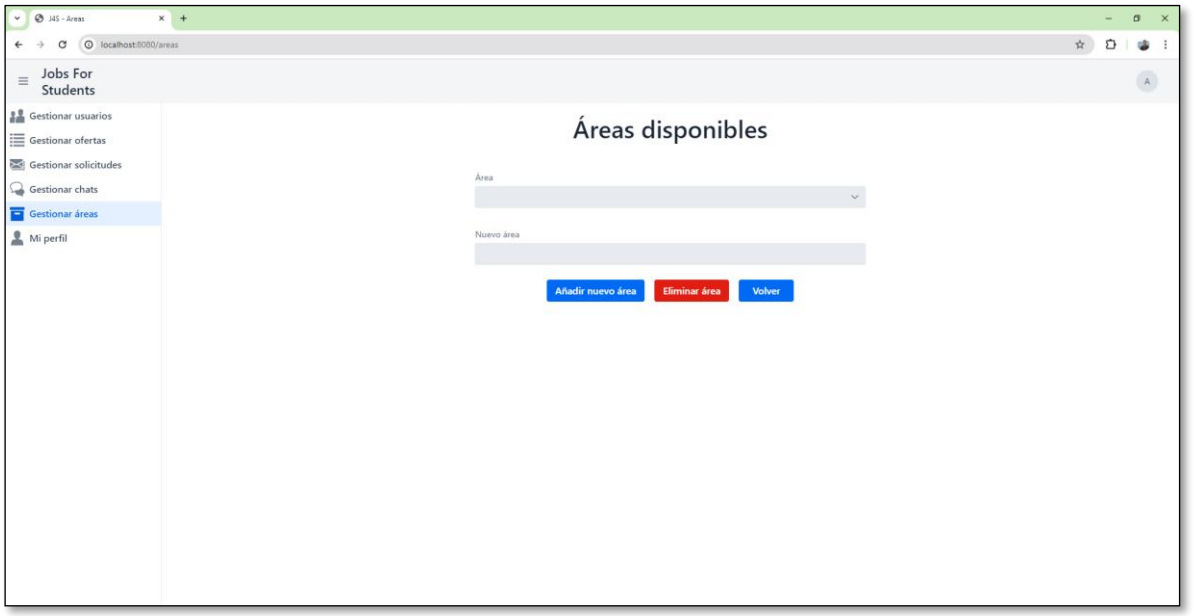

Ilustración 52. Captura de la vista de gestionar áreas

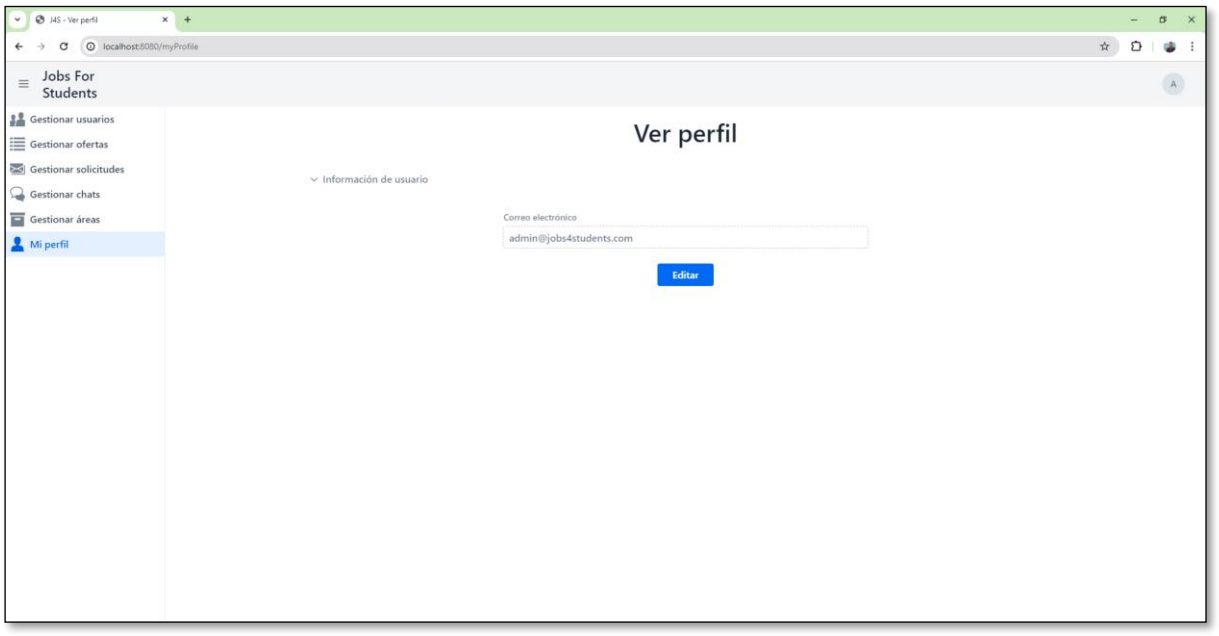

Ilustración 53. Captura de la vista de ver perfil (administrador) I

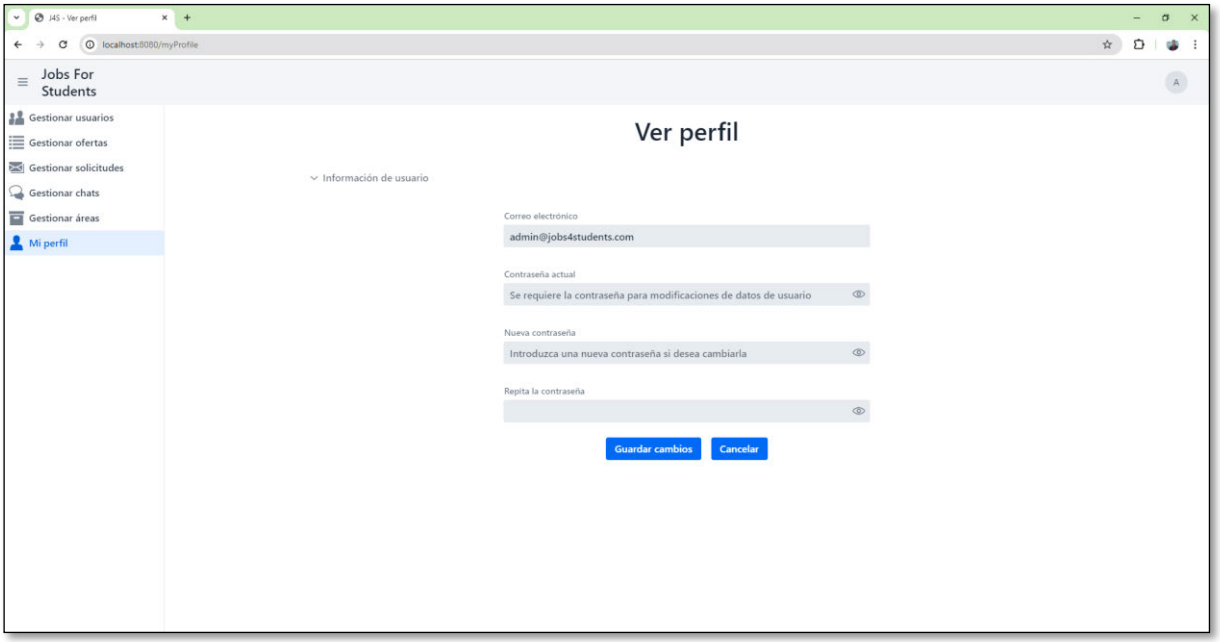

Ilustración 54. Captura de la vista de ver perfil (administrador) II