UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Informática

Diseño y desarrollo de una aplicación web de gestión de trabajadores contratados para eventos, desarrollado con Vue.js y Spring

Trabajo Fin de Grado

Grado en Ingeniería Informática

AUTOR/A: Romero Lara, Juan Sebastián

Tutor/a: Busquets Mataix, José Vicente

CURSO ACADÉMICO: 2023/2024

Resumen

En la sociedad actual, usamos diversas tecnologías como códigos QR y aplicaciones móviles para hacer casi todo. Sin embargo, hay un área que todavía no ha aprovechado al máximo estas tecnologías: el trabajo eventual, especialmente en el sector de la hostelería y catering.

Los camareros y camareras de catering suelen obtener trabajos contactando a través de WhatsApp y llamadas telefónicas, lo cual puede ser complicado y estresante, especialmente cuando los eventos se acercan rápidamente. Por eso, en este proyecto queremos desarrollar una aplicación web que ayude a las empresas y a los trabajadores eventuales a conectarse de manera más fácil.

El objetivo principal es crear una aplicación web utilizando el framework Vue.js para la parte visual y el framework de Java, Spring, para la parte técnica. Estas tecnologías se han elegido porque son rápidas, personalizables y seguras, lo que nos permitirá ofrecer una buena experiencia tanto a las empresas que buscan trabajadores como a los trabajadores que buscan empleo temporal.

Con esta aplicación, se pretende hacer más simple y rápido el proceso de contratación para aquellos que no buscan empleo fijo, sino que solo quieren trabajar en eventos o por un tiempo limitado. Además, nos aseguraremos de manejar la información de manera segura y proteger la privacidad de todos los usuarios.

En resumen, este proyecto se enfoca en crear una plataforma en línea que ayude a las empresas y a los trabajadores eventuales en el sector de la hostelería y catering a conectarse de manera más eficiente. Usando Vue.js y Spring, queremos ofrecer una aplicación fácil de usar, que maneje la información de forma segura y proporcione una buena experiencia para todos los usuarios.

Palabras clave: *Spring, Spring Boot, framework, java, javascript, Vue, JPA*

Abstract

In today's society, we use various technologies such as QR codes and mobile apps to do almost everything. However, there is one area that has yet to take full advantage of these technologies: casual labor, especially in the hospitality and catering sector.

Catering waiters and waitresses often get jobs by contacting through WhatsApp and phone calls, which can be complicated and stressful, especially when events are fast approaching. That's why in this project we want to develop a web application that helps companies and casual workers to connect more easily.

The main goal is to create a web application using the Vue.js framework for the visual part and the Java framework, Spring, for the technical part. These technologies have been chosen because they are fast, customisable and secure, which will allow us to offer a good experience to both companies looking for workers and workers looking for temporary employment.

With this application, we aim to make the recruitment process simpler and faster for those who are not looking for permanent employment, but only want to work at events or for a limited time. In addition, we will make sure to handle the information securely and protect the privacy of all users.

In short, this project focuses on creating an online platform that helps companies and casual workers in the hospitality and catering sector to connect more efficiently. Using Vue.js and Spring, we want to deliver an application that is easy to use, handles information securely and provides a good experience for all users.

Keywords: *Spring, Spring Boot, framework, java, javascript, Vue, JPA*

Tabla de contenidos

Índice de figuras

1. Introducción

En la sociedad actual, usamos diversas tecnologías como códigos QR y aplicaciones móviles para hacer casi todo. Sin embargo, hay un área que todavía no ha aprovechado al máximo estas tecnologías: el trabajo eventual, especialmente en el sector de la hostelería y catering.

Los camareros y camareras de catering suelen obtener trabajos contactando a través de WhatsApp y llamadas telefónicas, lo cual puede ser complicado y estresante, especialmente cuando los eventos se acercan rápidamente. Por eso, en este proyecto queremos desarrollar una aplicación web que ayude a las empresas y a los trabajadores eventuales a conectarse de manera más fácil.

El objetivo principal es crear una aplicación web utilizando el framework Vue.js para la parte visual y el framework de Java, Spring, para la parte técnica. Estas tecnologías se han elegido porque son rápidas, personalizables y seguras, lo que nos permitirá ofrecer una buena experiencia tanto a las empresas que buscan trabajadores como a los trabajadores que buscan empleo temporal.

Con esta aplicación, se pretende hacer más simple y rápido el proceso de contratación para aquellos que no buscan empleo fijo, sino que solo quieren trabajar en eventos o por un tiempo limitado. Además, nos aseguraremos de manejar la información de manera segura y proteger la privacidad de todos los usuarios.

En resumen, este proyecto se enfoca en crear una plataforma en línea que ayude a las empresas y a los trabajadores eventuales en el sector de la hostelería y catering a conectarse de manera más eficiente. Usando Vue.js y Spring, queremos ofrecer una aplicación fácil de usar, que maneje la información de forma segura y proporcione una buena experiencia para todos los usuarios.

1.1. Motivación

La motivación de este proyecto es la propia experiencia vivida durante mucho tiempo dentro del sector y la ilusión de llevar a cabo un proyecto propio con todas los retos que esto supone y la necesidad de mejorar en las capacidades obtenidas durante este tiempo y adquirir nuevas.

Este proyecto supone un gran paso en lo que el conocimiento del desarrollo supone, ya que será una experiencia que no solo aportará aptitudes útiles para el futuro, sino la posibilidad de llevar a cabo un proyecto de vida.

Esta aplicación puede implicar si todo funciona de una manera correcta, no solo un beneficio económico para todas las partes implicadas, o sino económico, de calidad de vida y laboral, ya que ese es uno de los principales objetivos, ayudar a que la gente no solo encuentre trabajo, sino que este sea de calidad y poder asegurarse de ello.

La creación de esta aplicación supone para mí la posibilidad de hacer frente a problemas que he podido vivir en el pasado, tanto en mi mismo, como en compañeros, o incluso en mis empleadores. Tanto en casos donde compañeros han tenido que seguir trabajando en un puesto que no les trataba bien, con jefes que les procuraban hacer la vida imposible a jefes de verdad honrados que no eran capaces de encontrar a trabajadores de calidad, no solamente con experiencia, sino con valores de trato hacia el cliente.

Finalmente, la posibilidad de una independencia económica basada en el emprendimiento, también es un gran punto de inspiración para seguir adelante con este proyecto.

1.2. Objetivo del proyecto

El objetivo del proyecto es el diseño exhaustivo de una primera versión de prueba de la aplicación web con su desarrollo inicial, donde poder así probar su funcionamiento y ser capaces de recibir feedback para así poder seguir adelante con su desarrollo futuro más adelante.

Esta primera versión deberá pese a no estar completamente finalizada, estar preparada para un gran número de usuarios o en su defecto, tener un tipo de implementación y diseño que sea altamente escalable y esté bien planteada, también deberá poder estar preparada para aceptar cambios que puedan surgir tanto por aplicar las ideas recibidas en el feedback recibido, como ideas que surjan durante el desarrollo, por lo que será necesario un diseño exhaustivo.

1.3. Otros objetivos

Se pueden obtener diversos objetivos secundarios que concreten más el proyecto:

- Estudiar y comprender el funcionamiento de algunas de las herramientas más utilizadas y populares para el desarrollo de código.
- Mejorar en todos los conceptos bases necesarios de la programación altamente escalable y general.
- Aprender sobre los errores cometidos en el desarrollo de esta aplicación web, para ser capaces de implementar nuevas versiones más eficientes y carentes de fallos.
- Analizar la situación dentro del mercado de las aplicaciones del estilo.
- Obtener unos resultados viables al final del proyecto que resuman todo lo descubierto/aprendido en el mismo.

1.4. Estructura

Tras la revisión de documentación apropiada para la realización de este proyecto, con los conceptos necesarios y básicos para su correcto funcionamiento, se han seleccionado dos de los métodos de programación que mejor se pueden aplicar a las necesidades de este proyecto, como son Vue.js y Spring.

Por lo que tras un estudio de estos conocimientos previamente adquiridos y su funcionamiento, este proyecto se ha dividido en las diferentes secciones que se presentan a continuación.

- *Aspectos generales del proyecto*: esto facilitará las ideas y crea unas bases de funcionamiento adecuado de la aplicación.
- *Metodología*: Una sección dedicada a detallar los métodos empleados durante la investigación y el desarrollo del proyecto.
- *Estado del arte:* En esta sección se añadirán los avances y proyectos sobre la temática de la aplicación
- *Análisis de requisitos:* En este apartado analizaremos de forma extensa el diseño de la aplicación, con sus diferentes tecnologías, riesgos, casos de uso, posibles soluciones y soluciones elegidas.

- *Diseño de la solución:* En esta sección se plantea la solución elegida previamente con mayor extensión

- *Implementación de la solución:* En este apartado se entra más en detalle sobre como se ha realizado la implementación
- *Testing:* Finalmente en este apartado veremos cuáles han sido los tests realizados, y cuales se consideran importantes para un futuro

2. Metodología

Al tratarse de un proyecto donde los recursos de personal son muy limitados, se ha optado por utilizar una metodología que permite cambios rápidos en el proyecto y una revisión constante de que se está haciendo, cómo se está haciendo y si el resultado es el esperado. En resumen, la metodología será una metodología ágil.

Las metodologías ágiles son aquellas que permiten que el proyecto se adapte a las condiciones de una forma rápida y eficaz, de esta forma se consigue flexibilidad y una mayor velocidad para adaptar el proyecto a las circunstancias específicas . Esto se consigue mediante los llamados sprints, que son ciclos de trabajo cortos que permiten la rápida adaptación al cambio.

Concretamente la metodología ágil elegida será el Scrum, que pese a su uso más estándar a equipos, aún así permite la autogestión de una forma muy guiada y la revisión constante del proceso de desarrollo [1]. Esto permite ver concretamente cómo se está avanzando en cada momento, y sobre todo ver concretamente en qué tareas o subtareas se está invirtiendo más tiempo del esperado.

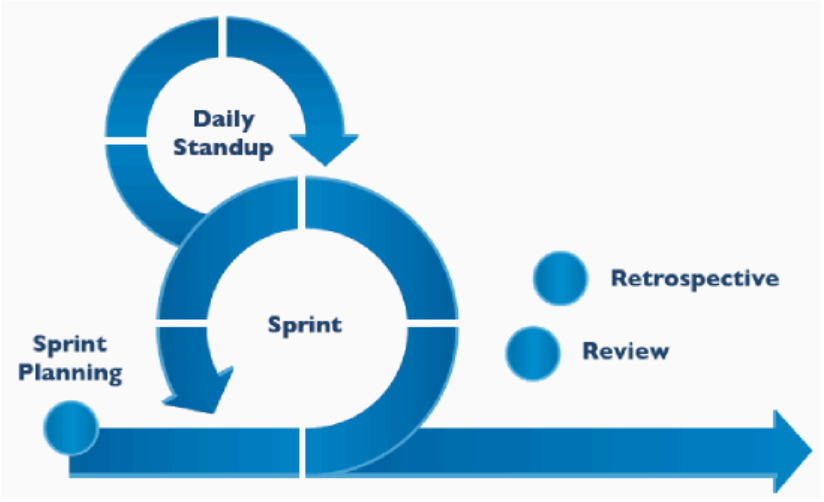

Figura 1. Representación del funcionamiento de la metodología Scrum.[2] (Roberto Touza, 2023)

Aquí, como podemos ver en la Figura 1, el funcionamiento del mismo, y como mencionado anteriormente, esto está más pensado para poner en común con un equipo cual está siendo el avance y las dificultades con figuras como el Scrum Master el Producto Owner, pero en este caso eran revisiones diarias individuales para ver el propio avance.

3. Contexto Social

En la actualidad, la digitalización y las tecnologías móviles han transformado muchos sectores, incluyendo el mercado laboral. Sin embargo, en áreas específicas, como el trabajo en la industria del catering, aún persisten métodos tradicionales de comunicación y organización que no han evolucionado al mismo ritmo, es el caso de los conocidos como “listeros”, que no son más que intermediarios que se llevan una comisión por trabajador aportado [3].

El objetivo de este Estado del Arte es revisar el panorama actual de las soluciones tecnológicas para el trabajo temporal, enfocándose en las aplicaciones disponibles, las tendencias tecnológicas y las innovaciones en seguridad y experiencia del usuario.

Actualmente existen diversas aplicaciones centradas en conectar trabajadores temporales con empleadores, como por ejemplo “Indeed”, “Freelancer” y “TaskRabbit”, estas otorgan soluciones a trabajadores freelance o temporales.

Estas empresas han fomentado la búsqueda de trabajo y la comunicación entre las diferentes empresas y trabajadores, pero en la mayoría de las ocasiones se trata de proyectos de más envergadura, o son puestos de trabajo que no están relacionados con el mundo de la hostelería a corto plazo.

Por otro lado, como aplicaciones de búsqueda de empleo más convencionales podemos encontrar aplicaciones como “Infojobs”, “Monster.com” o “Infoempleo” [4].

Estas aplicaciones pese a su utilidad en la búsqueda de empleo no ofrecen una posibilidad real a las empresas de publicar puestos de forma frecuente debido a su elevado precio, por lo que para un trabajo corto no sería rentable para ellos y el proceso de selección no sería lo suficientemente eficaz.

4. Contexto tecnológico y Tecnologías utilizadas

Ahora pasaremos a explicar algunas de las diferentes tecnologías utilizadas para el desarrollo de la aplicación en cuestión.

4.1. Frameworks y tecnologías transversales:

- **Spring Framework**

Es un framework para aplicaciones Java que facilita el desarrollo de aplicaciones robustas y escalables. Proporciona un conjunto de herramientas y características que incluyen inyección de dependencias, gestión de transacciones, programación orientada a aspectos (AOP), y soporte para el desarrollo de aplicaciones web y APIs RESTful. Spring es especialmente conocido por su capacidad de ser modulable, lo que permite a los desarrolladores usar solo las partes que necesitan, y por su integración con otros frameworks y bibliotecas del ecosistema de Java, como Spring Boot para la creación de aplicaciones con configuración mínima.

La información necesaria para su implementación fue adquirida en su mayor parte mediante su propio sitio web oficial [5].

- **Maven**

Se trata de una herramienta de gestión y automatización de proyectos en Java. Facilita la construcción, documentación y gestión de dependencias del proyecto mediante un archivo de configuración “pom.xml”(Project Object Model). Maven permite compilar, probar, empaquetar y desplegar aplicaciones de manera estandarizada y eficiente, integrando además la posibilidad de gestionar bibliotecas y plugins externos de forma automática.

Otra posibilidad al uso de Maven es Gradle, pero debido a que ya se tenía conocimiento previo sobre esta herramienta, finalmente se ha optado por Maven [6].

- **Spring Boot**

Esto no es más que una extensión del Framework de Spring que facilita el arranque del proyecto, eliminando la necesidad de configurar muchos aspectos del proyecto de forma manual.

Ofrece configuraciones automáticas, un servidor web embebido y herramientas listas para usar que permiten a los desarrolladores crear aplicaciones listas para producción con menos código y configuración, la estructura de la implementación estándar de Spring Boot la podemos ver en la figura 2.

Esta funcionalidad acelera considerablemente el desarrollo de aplicaciones, ya que con una mínima configuración permite iniciar las aplicaciones directamente desde el entorno de desarrollo sin necesidad de ajustes adicionales, y también simplifica las tareas de reinicio y depuración.

De la misma forma que el Spring Framework este conocimiento ha sido adquirido a partir del sitio oficial de *Spring Boot* [7].

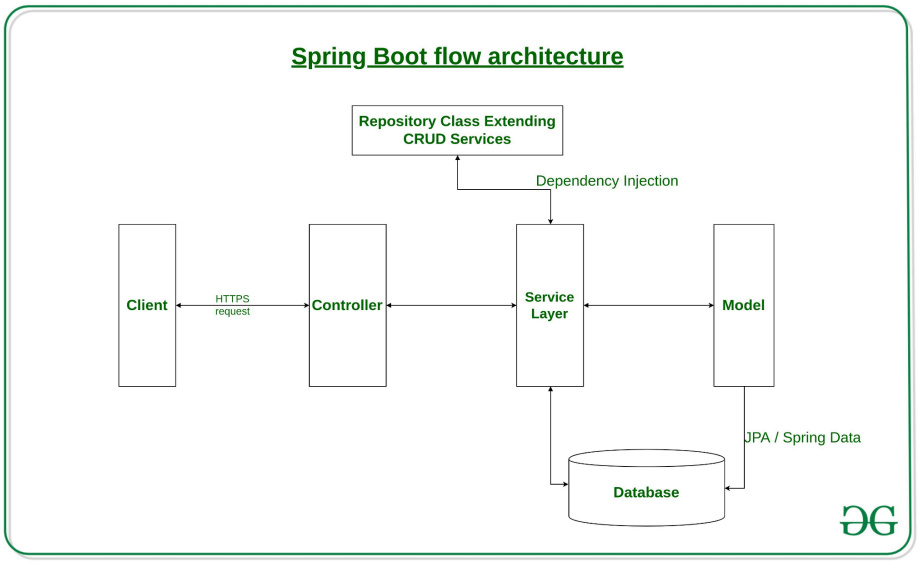

Figura 2. Arquitectura de Spring Boot [8] (Jain, 2024)

- **IntelliJ**

IntelliJ IDEA es un entorno de desarrollo integrado (IDE) para programar en Java y otros lenguajes. Está desarrollado por JetBrains, es conocido por su facilidad de trabajar con él, grandes herramientas de depuración y plugins, y soporte avanzado para frameworks como Spring y herramientas como Maven y Gradle, por lo que es de gran utilidad para este proyecto [9].

- **Git y GitLab**

Por un lado, git es el software de control de versiones más comúnmente utilizado por los desarrolladores de todo el mundo debido a ser de código abierto.

Permite el trabajo concurrente por parte de un equipo en un mismo proyecto de forma paralela, y por supuesto permite gestionar y observar los diferentes cambios es en código de un proyecto, siendo así capaces de tener un historial de todos los cambios y permitiendo la posibilidad de volver a diferentes puntos del pasado de ser necesario [10].

GitLab por otro lado se trata de una plataforma de DevOps que ofrece un entorno para la gestión de repositorios Git, integración continua y entrega continua (CI/CD) y diferentes herramientas de colaboración. Esto será de gran utilidad para poder gestionar el proyecto de una forma cómoda y ver cómo es su avance.

Mediante la integración que tiene Git junto a IntelliJ, permite un uso muy controlado de las versiones para así evitar cometer errores que puedan retrasar el proyecto.

- **SourceTree**

Se trata de una aplicación de interfaz gráfica para gestionar repositorios Git. Está desarrollada por Atlassian, facilita la visualización y gestión de cambios en el código, la resolución de conflictos, y la realización de operaciones como commits, merges y rebases sin necesidad de utilizar la línea de comandos.

Este software suele ser utilizado cuando no se cuenta con IntelliJ, aunque para este proyecto ha sido usado en conjunto dependiendo de la dificultad del “commit” para asegurar el funcionamiento correcto del proyecto [11].

4.2. Tecnologías en Backend:

- **JPA**

Se trata de una especificación de Java que proporciona un marco de trabajo común para la gestión de datos relacionales en aplicaciones Java. Esto nos permite mapear objetos Java a tablas de bases de datos relacionales, facilitando el manejo de las operaciones de persistencia (como insertar, actualizar, eliminar y consultar datos) sin necesidad de escribir consultas SQL directamente [12].

- **Hibernate**

Se trata de un framework de mapeo objeto-relacional (ORM) para Java que implementa JPA. Esta herramienta hace más fácil mapear los atributos entre una base de datos tradicional y el modelo de objetos de una aplicación utilizando anotaciones (Hibernate Annotations) en los beans o archivos XML declarativos. Esto permite adaptarse a una base de datos ya existente o crearla a partir de la información de los objetos. [13]

- **Spring JPA**

Java nos da un conjunto de reglas y directrices para trabajar con bases de datos sin importar el ORM que usemos. Esto establece un estándar para realizar operaciones en la base de datos y ofrece a nuestros repositorios los métodos básicos necesarios para gestionar y desarrollar la capa de persistencia.

A diferencia de Hibernate, que es una herramienta para mapear objetos a entidades de la base de datos, Spring Data JPA es la API que utiliza el ORM para manejar la persistencia de datos. Se ha utilizado mayoritariamente este tipo de conexión debido a su fácil implementación e integración con Spring [14].

- **Lombok**

Se trata de una librería para Java utilizada por su capacidad de simplificar y reducir el código repetitivo como los constructores, getters y setters. Mediante anotaciones se genera automáticamente el código en tiempo de compilación, facilitando así el mantenimiento y legibilidad del proyecto [15].

- **Spring Security**

Spring Security es un framework de Java que nos ofrece soluciones integradas para la autenticación y autorización en las aplicaciones Spring. Esto facilita la protección de aplicaciones mediante configuraciones flexibles y soporte para estándares como OAuth2 (en caso de en un futuro querer poder integrar autenticación mediante terceros), permitiendo gestionar el acceso a recursos y datos de manera segura [16].

- **Oracle DB**

Es un sistema de gestión de bases de datos relacional (RDBMS) desarrollado por Oracle. Es conocido por su robustez, escalabilidad y capacidad para manejar grandes volúmenes de datos. Soporta SQL para consultas y tiene características avanzadas como alta disponibilidad, seguridad, y soporte para transacciones complejas, lo que lo hace muy útil para una aplicación de estas características en la que manejaremos muchos datos [17].

4.3. Tecnologías en Frontend:

- **Vue.js**:

Es un framework progresivo de JavaScript que se usa para construir interfaces de usuario. Se centra en la capa de vista y es fácil de integrar con otros proyectos y bibliotecas. Nos permite crear aplicaciones web dinámicas y reactivas de manera eficiente, utilizando un sistema basado en componentes facilitando así la reutilización y organización del código.

Los conocimientos de este framework han venido mayoritariamente de su página web oficial, que se mantiene muy bien actualizada dada la característica de que es de código abierto [18].

- **Vue Router:**

Es la biblioteca oficial de enrutamiento para aplicaciones de Vue.js. Nos permite definir y gestionar rutas de manera intuitiva y eficiente, facilitando así la creación de aplicaciones de una sola página. Esto nos permite navegar entre diferentes componentes sin necesidad de recargar la página desde el servidor, haciendo que la experiencia sea más fluida [19].

- **Vuetify**:

Se trata de bibliotecas de componentes UI, estos sirven para poder obtener componentes preconstruidos para mantener una sensación de constancia en la aplicación y ahorrando tiempo también acelerando el desarrollo [20].

- **Axios**

Axios es una biblioteca de JavaScript con la que se puede hacer solicitudes HTTP. Es muy popular debido a la simplicidad que tiene, y al uso de promesas, lo que implica más facilidad en las operaciones asíncronas. También permite capturar y modificar solicitudes y respuestas, su manejo de errores también es muy interesante a la hora de tratar con APIs RESTful, como es el caso en esta aplicación web [21].

5. Análisis de requisitos

5.1. Visión

El objetivo de este apartado es definir los diferentes requerimientos de la aplicación que se va a desarrollar.

Para hacer esto se entregará una visión general de todo lo relacionado con la aplicación, como son los actores que participan y sus características.

5.1.1. Actores

Los actores hacen referencia a los roles que los usuarios de los casos de uso tienen al interactuar con la aplicación web.

- **Usuario No Registrado:** Persona o empresa aún sin registrar.

- **Camarero/a (Trabajador/a Eventual):** Persona que busca trabajo en eventos de forma temporal u ocasional.

- **Empresa de Catering:** Empresa que organiza y gestiona eventos que busca personal de trabajo para cubrir necesidades de camareros para un día o evento concreto.

- **Administrador/a del Sistema:** Persona encargada de gestionar la aplicación web, incluyendo moderación de usuarios, gestión de incidencias y en general el mantenimiento.

5.1.2. Características

Estas características representan los diferentes requisitos de alto nivel del nivel de software, los cuales serán descompuestos en requisitos más concretos, los casos de uso.

El sistema se divide en dos componentes o partes independientes que serán los que llevan a cabo las diferentes funciones definidas en el proyecto. Las funciones de cada componente son las siguientes:

Componente Software 1: Aplicación web

- **Sistema de autenticación**

  Implementación de todas las operaciones necesarias para que los usuarios (camareros/as, empresas y administradores) puedan registrarse, iniciar sesión, y recuperar sus credenciales.

- **Gestión de perfiles**

  Funciones que permiten a los usuarios gestionar su perfil personal, incluyendo la actualización de su información y configuración de preferencias.

- **Gestión de ofertas de Trabajo**

  Interfaz para que las empresas puedan publicar ofertas de trabajo y para que los camareros/as puedan buscar y apuntarse a estas ofertas.

- **Gestión de Solicitudes**

  Interfaz que permite a los trabajadores apuntarse a las solicitudes de trabajo presentes

- **Sistema de Comunicación**

  Funcionalidades que permitan la comunicación entre las empresas y los camareros/as a través de mensajes internos.

- **Gestión de notificaciones**

  El sistema de notificaciones para alertar a los usuarios sobre los diferentes sucesos importantes, como nuevas ofertas de trabajo, respuestas a solicitudes, o mensajes recibidos.

- **Gestión de Eventos**

  Las empresas podrán crear, modificar y eliminar eventos, estos eventos serán los que tendrán las ofertas de trabajo.

- **Gestión de Contratos**

  En este apartado las empresas serán capaces de crear, modificar y eliminar contratos asociados a las ofertas de trabajo.

Componente Software 2: Servidor Web

- **Sistema de autenticación**

  Proveer las operaciones necesarias para que un usuario
  pueda tener acceso a las áreas privadas del servidor.

- **Gestión de trabajadores y empresas**

  Operaciones relacionadas con la administración de la información de los usuarios en sí mismo, incluyendo creación, actualización, y eliminación de datos.

- **Gestión de Ofertas de Trabajo**

  Funciones esta vez en el backend para manejar la creación, actualización, eliminación y consulta de ofertas de trabajo

- **Gestión de los registros a Ofertas**

  Manejo de las solicitudes de los camareros/as a las ofertas de trabajo, incluyendo la aceptación o rechazo por parte de las empresas.

- **Sistema de mensajería interna**

  Implementación en el backend de un sistema de mensajería para permitir la comunicación directa entre las empresas y los camareros/as.

- **Gestión de Eventos**

  Funciones esta vez en el backend para manejar la creación, actualización, eliminación y consulta de eventos.

- **Generación de Reportes y Estadísticas**

  Generación de informes y estadísticas sobre el uso del sistema. Que ayuden a entender de una mejor manera el uso y los resultados de la misma.

- **Generación de Contratos**

  Funciones esta vez en el backend para manejar la creación, actualización, eliminación y consulta de contratos.

5.1.3. Requisitos no funcionales

Los requisitos no funcionales son características del sistema que definen el comportamiento del mismo, pero no en términos específicos, ya que esto forma parte de los requisitos funcionales.

Seguridad:

- Protección de datos personales de usuarios utilizando encriptación y asegurarse de cumplir la normativa vigente.
- Autenticación segura con uso de contraseñas cifradas y de una longitud y complejidad adecuada.

Escalabilidad:

- La aplicación debe ser capaz de manejar un número creciente de usuarios y transacciones sin pérdida de rendimiento.

Rendimiento:

- El tiempo de carga de las páginas debe ser rápido para que la experiencia de usuario sea correcta.
- El tiempo de respuesta para las acciones críticas (login, suscribirse a ofertas, mensajería) debe ser mínimo.

Usabilidad:

- La aplicación debe ser intuitiva y fácil de usar tanto en dispositivos móviles, como de escritorio.
- Los usuarios deben poder realizar todas las acciones más comunes en unos pocos clics.

Mantenimiento:

- La aplicación debe estar diseñada para facilitar futuras mejoras y correcciones.
- Uso de buenas prácticas de código, que sea de una forma clara y documentada.
- La aplicación no debe dejar de funcionar pese a que se den fallos

Compatibilidad:

- La aplicación debe ser compatible con los navegadores más utilizados y desde los dispositivos móviles.

5.1.4. Modelo de dominio

Un diagrama de modelo de dominio es un diagrama de clases UML que solo muestra aspectos del dominio, sin incluir detalles de diseño. Sus objetivos incluyen identificar y nombrar los conceptos importantes en el contexto del sistema a desarrollar (como entidades y eventos). Este diagrama ayuda a los desarrolladores a entender mejor el contexto del software. (Escuela de arquitectura tecnológica, 2024)

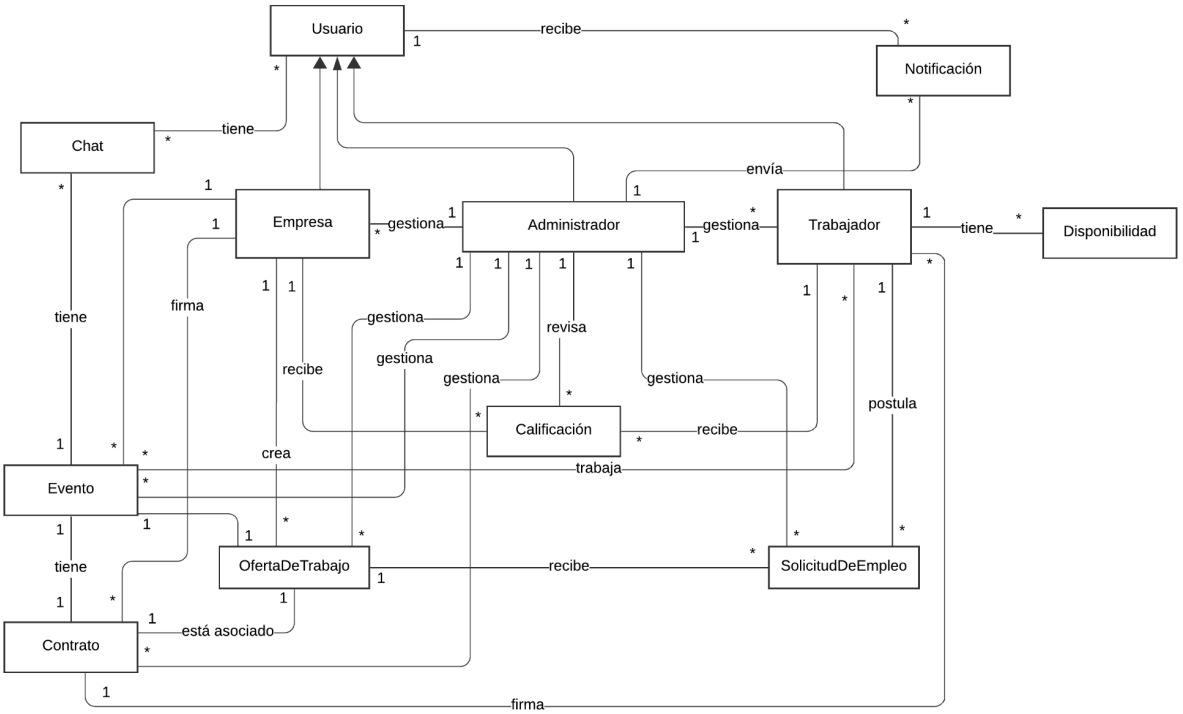

Figura 3. Modelo de dominio. (Creación Propia)

5.1.5. Modelo de contexto

Se trata de un diagrama de clases UML, en este se representan o definen las partes de cada sistema y además los límites entre los diferentes subsistemas, así se puede ver las diferentes entidades que interactúan con él. Es decir, se trata de una vista de alto nivel del sistema.

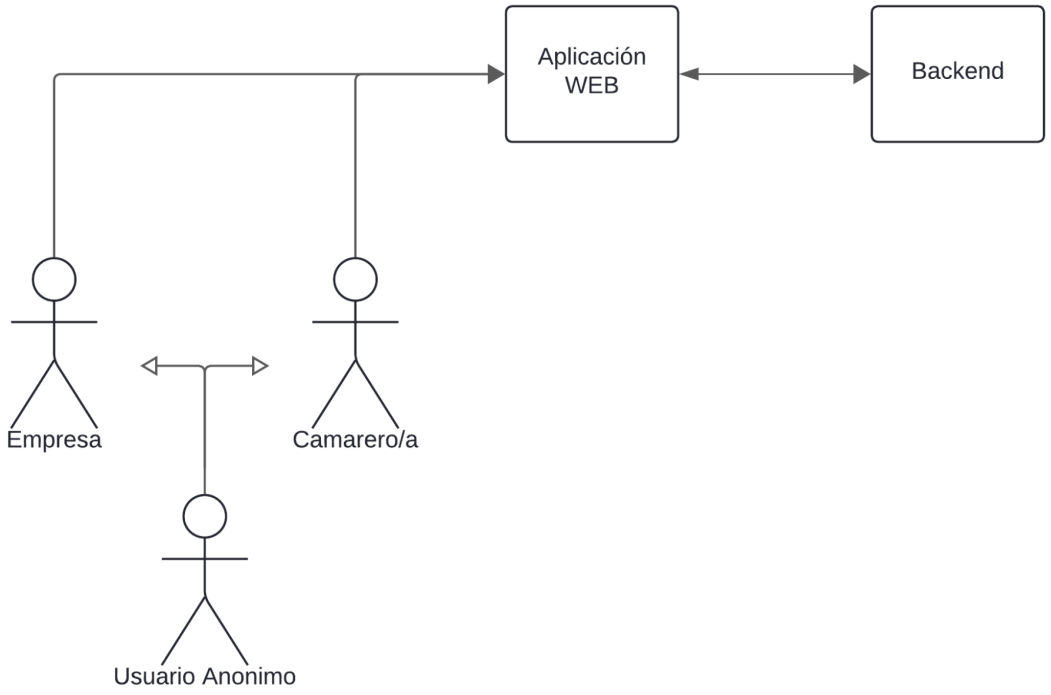

Figura 4. Modelo de contexto. (Creación Propia)

5.2. Casos de uso

En este apartado se definirán los casos de uso con respecto a las características del sistema. Para estos se proporcionan los diagramas de casos de uso para las características más pertinentes e información sobre los mismos, por ejemplo este es el caso de contrato, que se ha ignorado debido a su similitud con oferta de trabajo.

5.2.1. Sistema de autenticación Web App

En este apartado podemos ver el funcionamiento del sistema de autenticación desde el punto de vista de la aplicación web y sus diferentes casos de uso:

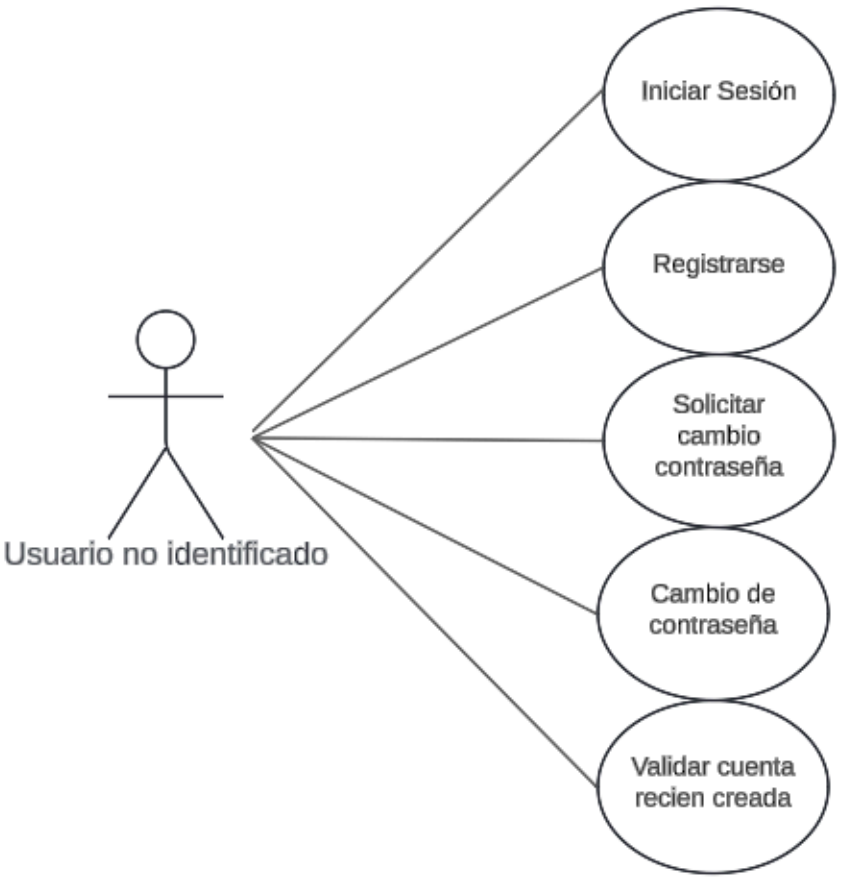

Figura 5. Diagrama de casos de uso en sistema de autenticación Web App. (Creación Propia)

Iniciar Sesión - web

Descripción	Permite a los usuarios registrados acceder a su cuenta mediante sus credenciales.
Actor	Usuario no identificado
Precondición	El usuario debe estar registrado en el sistema.
Postcondición	El usuario inicia sesión en su cuenta y es redirigido a su página de inicio
Flujo de eventos	● El usuario no identificado selecciona la opción "Iniciar Sesión". ● El sistema solicita al usuario ingresar su nombre de usuario y contraseña. ● El usuario ingresa sus credenciales. ● El sistema verifica las credenciales. ● Si las credenciales son correctas, el sistema redirige al usuario a su panel de control.
Flujo alternativo	● Si las credenciales son incorrectas, el sistema muestra un mensaje de error.

Registrarse - web

Descripción	Esto permite a los nuevos usuarios crear una cuenta en el sistema mediante un formulario, ya sea como empresa o como trabajador. Serán necesarios ciertos datos como nombre, dni, contraseña, correo y número de teléfono. La contraseña deberá cumplir unos estándares de seguridad para evitar riesgos debido al trato de datos personales. Tanto dni, número de teléfono, como correo electrónico deberán ser correctos y deben ser únicos para un usuario, en caso de no ser así, saltará una notificación.
Actor	Usuario no identificado
Precondición	Ninguna, cualquier usuario puede registrarse siempre y cuando cumpla los requisitos de dni o número de identificación, correo y teléfono.
Postcondición	El usuario tiene una cuenta creada en el sistema pendiente de validación.
Flujo de eventos	● El usuario no identificado selecciona la opción "Registrarse". ● El sistema solicita información como nombre, correo electrónico, dni o número de identificación, contraseña, y tipo de usuario (trabajador o empresa). ● El usuario ingresa la información requerida. ● El sistema valida la información. ● Si la información es válida, el sistema crea una cuenta pendiente de validación y envía un correo al usuario. ● El usuario debe verificar su correo electrónico para activar la cuenta.
Flujo alternativo	● Si alguno de los datos no es válido el usuario recibirá una notificación de error

Solicitar cambio de contraseña - web

Descripción	Permite a los usuarios iniciar el proceso de recuperación de contraseña. Para esto será necesario aportar el correo electrónico o número de teléfono.
Actor	Usuario no identificado
Precondición	Tener una cuenta creada y debidamente validada y acceso a su número de teléfono o correo electrónico.
Postcondición	Se envía un enlace de recuperación de contraseña al correo electrónico del usuario.
Flujo de eventos	● El usuario selecciona la opción "Olvidé mi contraseña". ● El sistema solicita al usuario su correo electrónico registrado o número de teléfono. ● El usuario ingresa su correo electrónico o número de teléfono. ● El sistema verifica que el correo electrónico o número de teléfono esté registrado. ● Si el correo o número de teléfono está registrado, el sistema envía un enlace de recuperación de contraseña. ● El usuario recibe el enlace y puede cambiar su contraseña.
Flujo alternativo	● Si alguno de los datos no es válido el usuario recibirá una notificación de error

Cambio de contraseña - web

Descripción	Permite a los usuarios cambiar su contraseña después de solicitar un cambio siempre y cuando hayan recibido un correo de cambio de contraseña.
Actor	Usuario No Identificado (inicialmente), Usuario Registrado (después de recibir el enlace)
Precondición	El usuario ha solicitado un cambio de contraseña y ha recibido un enlace de recuperación válido.
Postcondición	La contraseña del usuario se actualiza.
Flujo de eventos	● El usuario hace clic en el enlace de recuperación de contraseña recibido por correo. ● El sistema redirige al usuario a una página de cambio de contraseña. ● El usuario ingresa una nueva contraseña y confirma la misma. ● El sistema valida la nueva contraseña y la actualiza. ● El sistema muestra un mensaje de confirmación de que la contraseña ha sido cambiada.
Flujo alternativo	● Si la contraseña no es válida o es la misma a la previa, el usuario recibirá un mensaje de error.

Validar cuenta recientemente creada - web

Descripción	Los usuarios podrán validar su cuenta mediante un enlace de confirmación enviado a su correo electrónico. Este enlace llevará a la página que confirma la validación
Actor	Usuario No Identificado (Usuario recién registrado)
Precondición	El usuario se ha registrado pero aún no ha validado su cuenta.
Postcondición	La cuenta del usuario queda validada y activada.
Flujo de eventos	● El usuario recibe un correo de confirmación tras registrarse. ● El usuario hace clic en el enlace de validación. ● El sistema valida el enlace y activa la cuenta del usuario. ● El sistema muestra un mensaje de confirmación de cuenta activada.
Flujo alternativo	● Si el enlace ha expirado el usuario no podrá verificar su cuenta. Deberá proceder con el registro otra vez.

5.2.2. Gestión de perfiles Web App

En este apartado podemos ver el funcionamiento del sistema de gestión de perfiles desde el punto de vista de la aplicación web y sus diferentes casos de uso:

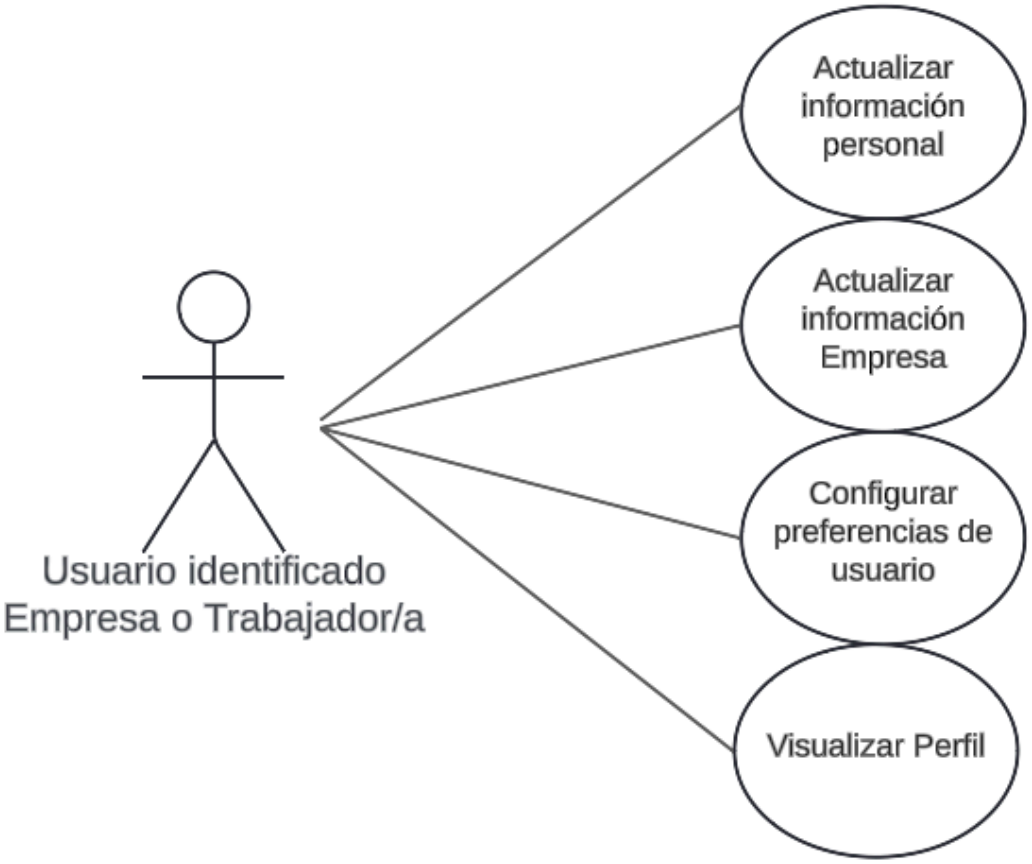

Figura 6. Diagrama de casos de uso gestión de perfiles Web App. (Creación Propia)

Actualizar Información personal trabajador - web

Descripción	Los camareros/as pueden actualizar su información personal. Ya sea para modificaciones en su propio perfil como para añadir nueva información que le pueda beneficiar a la hora de encontrar empleo, experiencia, certificados etc.
Actor	Camarero/a (Trabajador/a de eventos)
Precondición	El usuario debe haber iniciado sesión y tener acceso a su perfil.
Postcondición	La información personal del usuario se actualiza en el sistema.
Flujo de eventos	● El usuario selecciona la opción de editar perfil. ● El sistema muestra la información actual del perfil con posibilidad de cambio. ● El usuario realiza cambios en la información personal. ● El sistema valida y guarda los cambios. ● El sistema muestra un mensaje de confirmación de que la información ha sido actualizada.
Flujo alternativo	● Si alguno de los campos previamente correctos pasa a ser incorrecto, el proceso no podrá ser guardado y el usuario recibirá una notificación para su corrección.

Actualizar Información de Empresa - web

Descripción	Las empresas podrán actualizar su información, como detalles de contacto, personal fijo y descripción de la empresa.
Actor	Empresa
Precondición	La empresa debe haber iniciado sesión y tener acceso a su perfil.
Postcondición	La información de la empresa se actualiza en el sistema.
Flujo de eventos	● La empresa selecciona la opción de editar perfil. ● El sistema muestra la información actual de la empresa con posibilidad de cambio. ● La empresa realiza cambios en la información de perfil. ● El sistema valida y guarda los cambios. ● El sistema muestra un mensaje de confirmación de que la información ha sido actualizada.
Flujo alternativo	● Si alguno de los campos previamente correctos de la empresa pasa a ser incorrecto, el proceso no podrá ser guardado y la empresa recibirá una notificación para su corrección.

Configurar preferencias de usuario - web

Descripción	Los usuarios podrán editar sus preferencias, como su visibilidad, aceptación de mensajes directos, y otras numerosas opciones.
Actor	Usuario identificado
Precondición	El usuario debe haberse identificado
Postcondición	La información de la empresa se actualiza en el sistema.
Flujo de eventos	• El usuario selecciona la opción de configuración. • El sistema muestra la configuración actual del perfil. • El usuario realiza cambios en la configuración del perfil. • El sistema valida y guarda los cambios. • El sistema muestra un mensaje de confirmación de que la información ha sido actualizada.

Visualizar perfil - web

Descripción	Los usuarios, tanto empresas como trabajadores podrán ver su perfil para revisar que todo esté correcto y que estén de acuerdo con lo que las empresas o trabajadores verán de ellos.
Actor	Usuario identificado
Precondición	El usuario debe haber iniciado sesión y tener acceso a su perfil.
Postcondición	La información del perfil será mostrada al usuario

Flujo de eventos	• El usuario selecciona la opción de "ver perfil". • El sistema muestra el propio perfil con la disposición real que verá el resto de usuarios, en caso de todo ser correcto y no obtener un fallo del servidor.

5.2.3. Gestión de Ofertas de empleo Web App

En este apartado podemos ver el funcionamiento del sistema de gestión de ofertas de trabajo desde el punto de vista de la aplicación web y sus diferentes casos de uso:

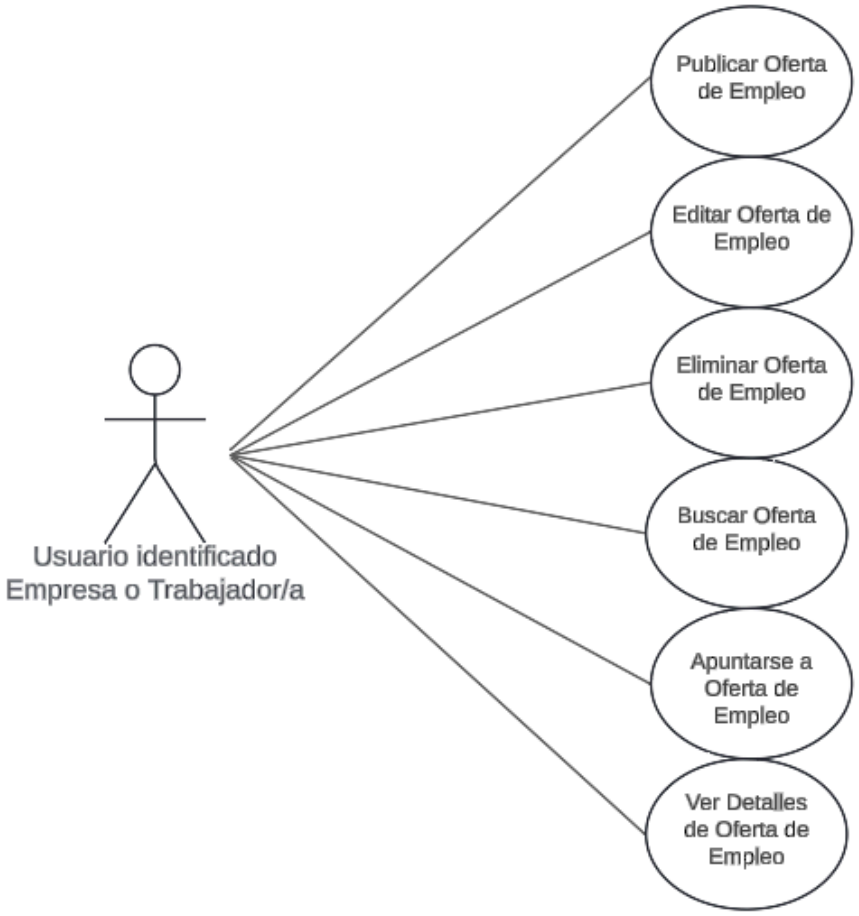

Figura 7. Diagrama de casos de uso gestión de ofertas de empleo Web App. (Creación Propia)

Publicar Oferta de Empleo- web

Descripción	Las empresas publican nuevas ofertas de trabajo para eventos. En estas aparecerá la información relevante para el mismo.
Actor	Empresa
Precondición	La empresa debe haber iniciado sesión en su cuenta y la oferta debe cumplir con la normativa vigente.
Postcondición	La oferta de trabajo se publica y es visible para los camareros/as.
Flujo de eventos	● La empresa selecciona la opción de "publicar una nueva oferta de trabajo". ● El sistema muestra un formulario de oferta de trabajo. ● La empresa completa el formulario con los detalles del trabajo. ● El sistema valida los detalles ingresados. ● Si los detalles son válidos, el sistema publica la oferta de trabajo. ● La oferta de trabajo es visible para los camareros/as en la plataforma. ● El sistema muestra un mensaje de confirmación de que la información ha sido actualizada.
Flujo alternativo	● Si los detalles dados por la empresa no son válidos, la empresa será informada del error para su corrección.

Editar Oferta de Empleo- web

Descripción	Las empresas pueden editar algunos parámetros de las ofertas de empleo previamente creadas.
Actor	Empresa
Precondición	La empresa debe haber iniciado sesión en su cuenta y la oferta debe haber sido creada previamente.
Postcondición	La oferta de trabajo se actualiza y es visible para los camareros/as.
Flujo de eventos	• La empresa selecciona la opción de "editar oferta de trabajo". • El sistema muestra la oferta de empleo. • La empresa actualiza los parámetros con los nuevos detalles. • El sistema valida los detalles ingresados. • Si los detalles son válidos, el sistema actualiza la oferta de trabajo. • La oferta de trabajo es visible para los camareros/as en la plataforma. • El sistema muestra un mensaje de confirmación de que la información ha sido actualizada.
Flujo alternativo	• Si las actualizaciones no son válidas, la empresa será informada del error para su corrección.

Buscar Oferta de Empleo - web

Descripción	Los usuarios con acceso pueden buscar ofertas de empleo, tanto para apuntarse a ellas como para ver las ofertas que aportan las diferentes empresas.
Actor	Camarero/a (Trabajador/a de Eventos)
Precondición	El camarero/a debe haber iniciado sesión en su cuenta.
Postcondición	El camarero/a ve una lista de ofertas de trabajo disponibles.

Eliminar Oferta de Empleo - web

Descripción	Las empresas pueden eliminar ofertas de empleo previamente creadas.
Actor	Empresa
Precondición	La empresa debe haber iniciado sesión en su cuenta y la oferta debe haber sido creada previamente.
Postcondición	La oferta de trabajo se elimina y deja de ser visible para los camareros/as.
Flujo de eventos	• La empresa selecciona la opción de “eliminar oferta de trabajo”. • El sistema muestra la oferta de empleo. • El sistema pregunta el motivo de la eliminación. • La empresa confirma la eliminación de la oferta.

	● La oferta de trabajo deja de ser visible para los camareros/as en la plataforma y en caso de haber estado apuntados reciben una notificación. ● El sistema muestra un mensaje de confirmación de que la oferta ha sido borrada.

Apuntarse a oferta de empleo - web

Descripción	Los camareros tras revisar las diferentes ofertas de empleo pueden realizar cuantas solicitudes quieran para que la empresa tome en cuenta su perfil para el puesto
Actor	Camarero/a (Trabajador/a de Eventos)
Precondición	El camarero/a debe haber iniciado sesión y encontrado una oferta de trabajo que le interese.
Postcondición	El camarero/a envía su solicitud para la oferta de trabajo.
Flujo de eventos	● El camarero/a selecciona una oferta de trabajo. ● El sistema muestra los detalles de la oferta. ● El camarero/a selecciona la opción de apuntarse. ● El sistema solicita confirmar la solicitud. ● El camarero/a confirma la solicitud. ● El sistema envía la solicitud a la empresa.

5.2.4. Gestión de solicitudes de empleo Web App

En este apartado podemos ver el funcionamiento del sistema de gestión de solicitudes desde el punto de vista de la aplicación web y sus diferentes casos de uso:

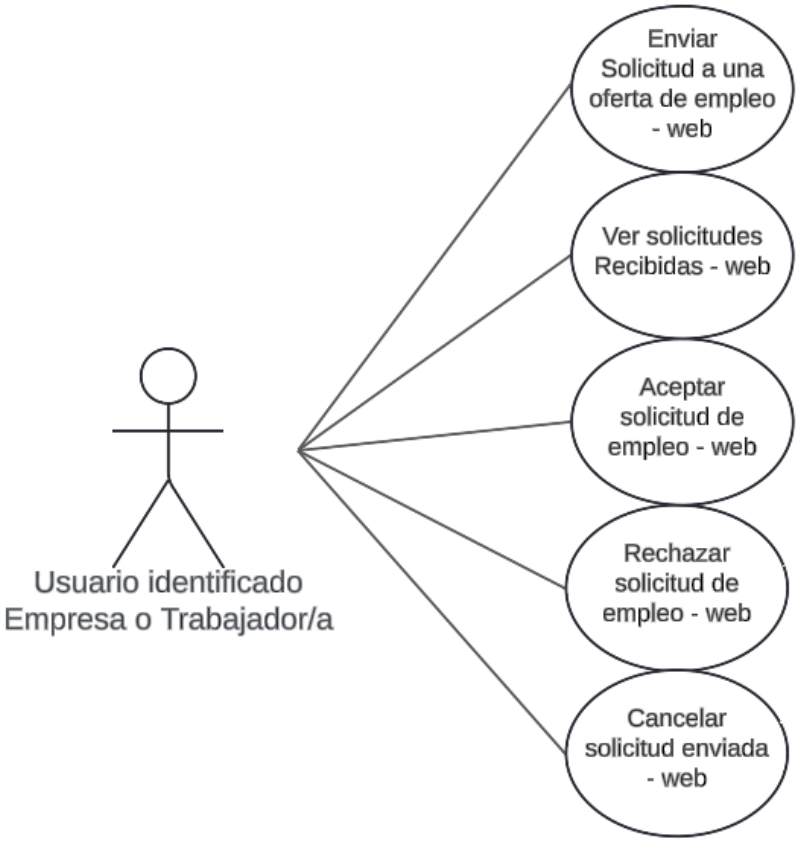

Figura 8. Diagrama de casos de uso gestión de solicitudes de empleo Web App. (Creación Propia)

Revisar solicitudes de empleo recibidas - web

Descripción	Las empresas podrán ver las diferentes solicitudes que han recibido por parte de los trabajadores. Estas podrán ser filtradas por diferentes parámetros , pudiendo así elegir en un primer lugar a aquellos que cumplan sus requerimientos, sea de reseñas de anteriores empresas, años de experiencia o idiomas entre otras.
Actor	Empresas
Precondición	La empresa debe haber iniciado sesión y tener una o varias ofertas de empleo publicadas con solicitudes recibidas.
Postcondición	La empresa podrá ver las solicitudes de una forma clara
Flujo de eventos	• La empresa selecciona la opción “ver solicitudes” dentro de la oferta que tengan publicada • El sistema entrega la lista de candidatos con posibilidad de filtrar.

Aceptar o rechazar solicitud de empleo - web

Descripción	Tras haber visto las diferentes solicitudes de los candidatos la empresa decidirá si aceptar o rechazar a los candidatos, debido a la posibilidad de que para un puesto se presenten muchos candidatos, una vez la empresa haya confirmado a los camareros necesarios de la oferta, todo el resto será rechazado automáticamente.
Actor	Empresas
Precondición	La empresa debe haber recibido solicitudes de trabajo para una oferta publicada.
Postcondición	La solicitud es aprobada o rechazada y el camarero/a es notificado/a de la decisión.
Flujo de eventos	● La empresa inicia sesión y accede a la lista de solicitudes para una oferta específica. ● El sistema muestra las solicitudes recibidas. ● La empresa revisa cada solicitud y selecciona la opción de aprobar o rechazar una solicitud. ● El sistema registra la decisión y actualiza el estado de la solicitud. ● El sistema envía una notificación al camarero/a sobre la decisión tomada.

Enviar solicitud de empleo - web

Descripción	Tras ver las diferentes ofertas de empleo de la aplicación un trabajador podrá mediante la interfaz enviar una solicitud a una de las ofertas de empleo.
Actor	Camarero/a (Trabajador/a Eventual)
Precondición	El trabajador debe estar autenticado y haber encontrado una oferta de trabajo que le interese.
Postcondición	La interfaz web muestra una confirmación de que la solicitud ha sido enviada exitosamente.
Flujo de eventos	● El trabajador inicia sesión en la aplicación web y navega a la sección de ofertas de trabajo. ● El trabajador encuentra una oferta de trabajo de interés y selecciona la opción para postularse. ● La aplicación web muestra un formulario para enviar la solicitud. ● El trabajador completa el formulario y lo envía. ● Si todo va correctamente la interfaz muestra un mensaje de éxito al trabajador.

Cancelar solicitud de empleo enviada - web

Descripción	Permite a un trabajador cancelar una solicitud previamente enviada a una oferta de trabajo a través de la interfaz web.
Actor	Camarero/a (Trabajador/a Eventual)
Precondición	El trabajador debe estar autenticado y haber enviado una solicitud a una oferta de trabajo que aún no ha sido aceptada o rechazada.
Postcondición	La solicitud es cancelada, y la interfaz web muestra una confirmación de la cancelación.
Flujo de eventos	● El trabajador usa la opción de “solicitudes enviadas". ● El trabajador encuentra la solicitud que desea cancelar y selecciona la opción de cancelar. ● La aplicación web envía la acción de cancelación al servidor. ● La aplicación web muestra una confirmación de la cancelación al trabajador.

5.2.5. Sistema de Comunicaciones Web App

En este apartado podemos ver el funcionamiento del sistema de gestión de comunicaciones desde el punto de vista de la aplicación web y sus diferentes casos de uso:

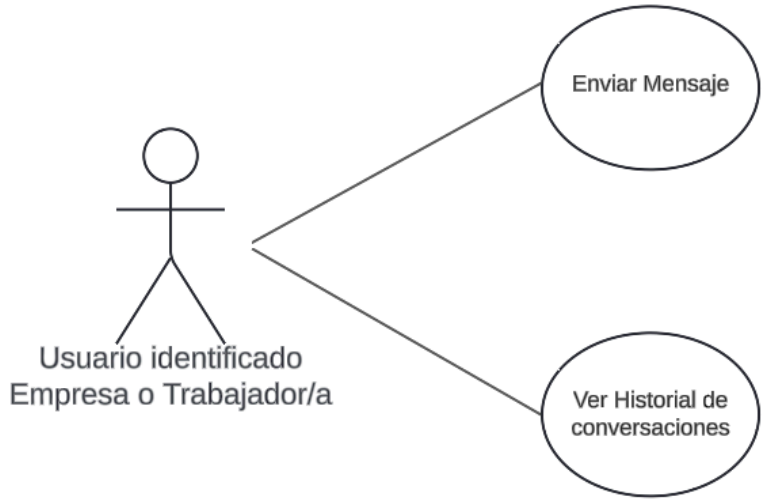

Figura 9. Diagrama de casos de uso Sistema de Comunicaciones Web App. (Creación Propia)

Enviar Mensaje

Descripción	Una vez ha sido aceptada la solicitud por parte de la empresa, tanto la empresa como el trabajador podrán comunicarse entre ellos de forma privada, o en un mensaje difundido a todos los trabajadores aceptados.
Actor	Empresa o Trabajador(Camarero/a)
Precondición	El usuario debe haber iniciado sesión.

Postcondición	El mensaje se envía y queda registrado en el sistema.
Flujo de eventos	• El usuario selecciona la opción de enviar un mensaje sobre la empresa o candidato. • El sistema muestra un formulario de mensaje. • El usuario completa el formulario con el contenido del mensaje. • El destinatario recibe el mensaje.

Ver Historial de conversaciones - web

Descripción	Todos los usuarios identificados pueden acceder a su historial de mensajes donde verán las diferentes conversaciones que tengan o hayan tenido, permitiendo así responder cómodamente.
Actor	Usuario Identificado. Empresa o Trabajador (Camarero/a)
Precondición	El usuario debe haber iniciado sesión.
Postcondición	El usuario ve las diferentes conversaciones pudiendo responder de forma fácil.
Flujo de eventos	• El usuario selecciona la opción de ver “mensajes”. • El sistema muestra las conversaciones.

5.2.6. Gestión de notificaciones Web App

En este apartado podemos ver el funcionamiento del sistema de gestión de notificaciones desde el punto de vista de la aplicación web y sus diferentes casos de uso:

Figura 10. Diagrama de casos de uso gestion de notificaciones Web App. (Creación Propia)

Configurar preferencias de notificación - web

Descripción	Todos los usuarios identificados pueden cambiar los ajustes de sus notificaciones para que se adapten al uso que mejor les convenga.
Actor	Usuario Identificado. Empresa o Trabajador (Camarero/a)
Precondición	El usuario debe haber iniciado sesión.
Postcondición	El usuario cambia las preferencias de configuración.

Flujo de eventos	• El usuario selecciona la opción de “ajustes" y dentro “notificaciones”. • El sistema muestra los ajustes de notificaciones. • El usuario selecciona los cambios y estos se aplican.

5.2.7. Gestión de Eventos WebApp

En este apartado podemos ver el funcionamiento del sistema de gestión de eventos desde el punto de vista de la aplicación web y sus diferentes casos de uso:

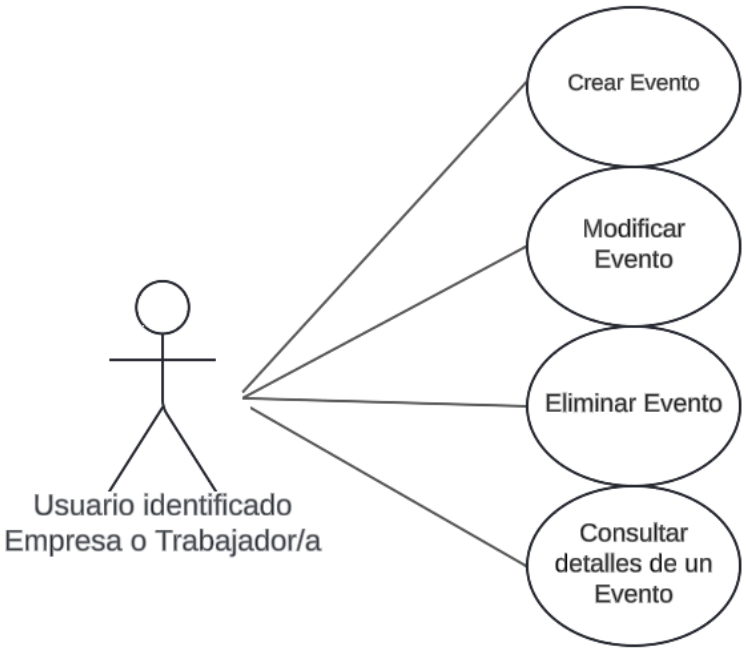

Figura 11. Diagrama de casos de uso gestión de eventos Web App. (Creación Propia)

Creación de un evento - web

Descripción	Una empresa de catering creará un nuevo evento para el cual necesitarán trabajadores. La empresa especifica los detalles del evento, como la fecha, hora, ubicación y requisitos específicos. Dentro de estos eventos estarán los diferentes puestos de trabajo.
Actor	Empresa
Precondición	La empresa debe haber iniciado sesión.
Postcondición	Se crea un nuevo evento, haciéndolo visible para la gestión de ofertas de trabajo y para los trabajadores que buscan empleo.
Flujo de eventos	● La empresa de catering accede a la sección de gestión de eventos. ● Selecciona la opción de "Crear Evento". ● Ingresa los detalles del evento, incluyendo nombre, fecha, hora, ubicación y descripción. ● El sistema valida los datos ingresados (que la fecha dada sea válida). ● El sistema guarda los detalles del evento. ● El sistema confirma la creación del evento y lo muestra en la lista de eventos de la empresa.

Eliminar un evento - web

Descripción	La empresa puede eliminar un evento que ya no sea necesario. Esto podría ocurrir por ejemplo si el evento se cancela.
Actor	Empresa
Precondición	La empresa debe haber iniciado sesión y debe existir al menos un evento creado por la empresa que pueda ser eliminado..
Postcondición	El evento es eliminado, y ya no es visible para los trabajadores ni para la empresa misma en la lista de eventos.
Flujo de eventos	● La empresa accede a la lista de eventos. ● Selecciona un evento específico que desea eliminar. ● El sistema solicita confirmación para eliminar el evento. ● La empresa confirma la eliminación. ● El sistema confirma la eliminación del evento y lo remueve de la lista visible para la empresa.

Modificación de un evento - web

Descripción	Permite a la empresa de catering actualizar los detalles de un evento previamente creado. Esto es útil por si cambian los requisitos, la ubicación, la fecha, o cualquier otro detalle importante del evento.
Actor	Empresa
Precondición	La empresa debe haber iniciado sesión y debe haber al menos un evento creado previamente que pueda ser modificado..
Postcondición	Los detalles del evento son actualizados y los cambios son reflejados en las interfaces de usuario pertinentes.
Flujo de eventos	● La empresa accede a la lista de eventos creados. ● Selecciona un evento específico para modificar. ● El sistema muestra los detalles actuales del evento. ● La empresa edita los campos necesarios. ● El sistema valida los cambios realizados. ● El sistema guarda los cambios en la base de datos. ● El sistema confirma que los detalles del evento han sido actualizados. ● El sistema valida los datos ingresados (que la fecha dada sea válida). ● El sistema guarda los detalles del evento. ● El sistema confirma la creación del evento y lo muestra en la lista de eventos de la empresa.
Flujo alternativo	● Si alguno de los campos previamente correctos pasa a ser incorrecto, el proceso no podrá ser guardado y la empresa recibirá una notificación para su corrección.

Consultar detalles de un evento - web

Descripción	Permite a los usuarios revisar los detalles completos de un evento, incluidos los trabajadores que han aplicado y cualquier solicitud pendiente en caso de ser la propia empresa.
Actor	Usuario Identificado. Empresa o Trabajador (Camarero/a)
Precondición	El usuario debe haber iniciado sesión y debe existir al menos un evento.
Postcondición	Se obtiene la información detallada del evento.
Flujo de eventos	● El usuario accede a algún evento y lo selecciona para revisar. ● El sistema muestra los detalles completos del evento. ○ En caso de ser la propia empresa también puede ver las solicitudes.

5.2.8. Sistema de autenticación (server)

En este apartado podemos ver el funcionamiento del sistema de gestión de autenticaciones desde el punto de vista del server y sus diferentes casos de uso:

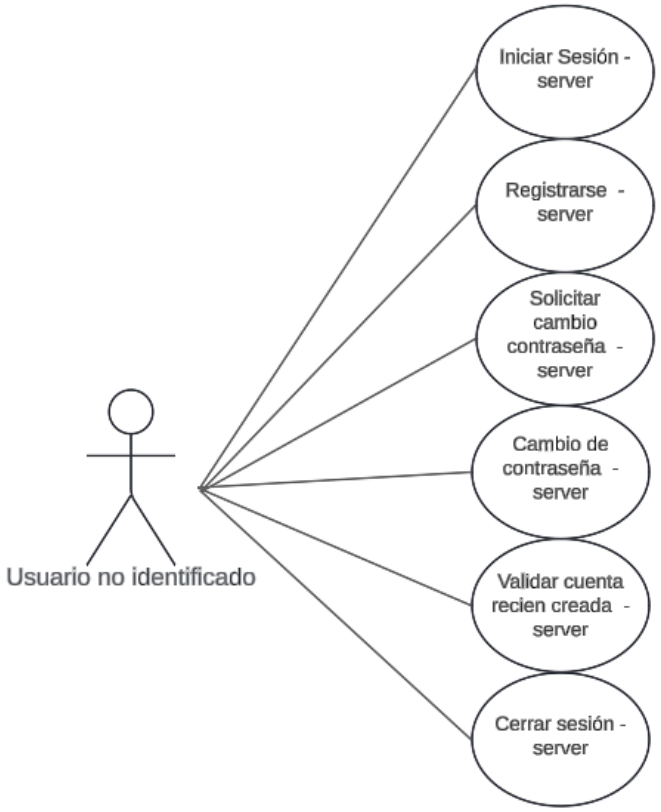

Figura 12. Diagrama de casos de uso del sistema de autenticación en el servidor. (Creación Propia)

Iniciar Sesión - server

Descripción	Permite a los usuarios registrados acceder a su cuenta mediante sus credenciales (usuario y contraseña). Una vez autenticados, los usuarios pueden acceder a las áreas privadas de la aplicación.
Actor	Usuario no identificado
Precondición	El usuario debe estar registrado en el sistema.
Postcondición	El usuario recibe un token de autenticación que le permite acceder a las áreas privadas del servidor.
Flujo de eventos	● El usuario envía una solicitud de inicio de sesión con sus credenciales (usuario y contraseña) al servidor. ● El servidor valida las credenciales comparándolas con las almacenadas en la base de datos. ● Si las credenciales son válidas, el servidor genera un token de autenticación. ● El servidor envía el token de autenticación al usuario. ● El usuario utiliza el token para acceder a las áreas privadas.

Cerrar Sesión - server

Descripción	Permite a un usuario autenticado finalizar su sesión, invalidando su token de autenticación.
Actor	Usuario Autenticado
Precondición	El usuario debe estar registrado en el sistema y tener un token de sesión válido.
Postcondición	El token de autenticación del usuario se invalida y el acceso a áreas privadas es revocado.
Flujo de eventos	● El usuario envía una solicitud de cierre de sesión al servidor. ● El servidor invalida el token de autenticación del usuario. ● El servidor confirma que el cierre de sesión ha sido exitoso.

Registrarse - server

Descripción	Dados ciertos datos como nombre, dni, contraseña, correo y número de teléfono, se creará un usuario no validado y se mandará un correo a la cuenta otorgada. Tanto dni, número de teléfono, como correo electrónico deberán ser correctos y deben ser únicos para un usuario, en caso de no ser así, saltará una notificación.
Actor	Usuario no identificado
Precondición	Enviar los datos del nuevo usuario y que sean correctos.
Postcondición	El usuario tiene una cuenta creada en el sistema pendiente de validación y se le enviará un correo para la validación.
Flujo de eventos	● El usuario envía una solicitud de registro al servidor con sus datos personales (nombre, correo electrónico, contraseña, tipo de usuario, etc.). ● El servidor valida los datos proporcionados (por ejemplo, verifica si el correo electrónico ya está registrado). ● El servidor guarda los datos del nuevo usuario en la base de datos. ● El servidor genera un token de validación y envía un correo electrónico al usuario con un enlace de validación. ● El servidor confirma la creación del usuario y la solicitud de validación pendiente.
Flujo alternativo	● Si alguno de los datos no es válido el usuario recibirá una notificación de error

Solicitar cambio de contraseña - server

Descripción	Permite a un usuario solicitar un cambio de contraseña si la ha olvidado o desea cambiarla por motivos de seguridad. Se envía un enlace de restablecimiento de contraseña al correo electrónico del usuario.
Actor	Usuario no identificado
Precondición	El usuario debe estar registrado en el sistema y proporcionar un correo electrónico válido.
Postcondición	Se envía un enlace de recuperación de contraseña al usuario.
Flujo de eventos	● El usuario accede a la página de cambio de contraseña y proporciona su correo electrónico. ● El servidor verifica que el correo electrónico está asociado a una cuenta existente. ● El servidor genera un token de restablecimiento de contraseña y lo almacena en la base de datos. ● El servidor envía un correo electrónico al usuario con un enlace de restablecimiento de contraseña que incluye el token. ● El servidor confirma la solicitud de cambio de contraseña.
Flujo alternativo	● Si alguno de los datos no es válido el usuario recibirá una notificación de error, por ejemplo si el token está caducado.

Cambio de contraseña - server

Descripción	Se le permite a un usuario cambiar su contraseña utilizando un token de restablecimiento de contraseña válido recibido por correo electrónico.
Actor	Usuario no identificado (inicialmente), usuario registrado (después de recibir el enlace)
Precondición	El usuario debe tener un token de restablecimiento de contraseña válido.
Postcondición	La contraseña del usuario se actualiza en la base de datos.
Flujo de eventos	● El usuario accede al enlace de restablecimiento de contraseña recibido en su correo electrónico. ● El usuario proporciona una nueva contraseña en la página de restablecimiento de contraseña. ● El servidor valida el token de restablecimiento de contraseña. ● El servidor actualiza la contraseña del usuario en la base de datos. ● El servidor desactiva el token de restablecimiento de contraseña. ● El servidor confirma el cambio de contraseña al usuario.
Flujo alternativo	● Si la contraseña no es válida o es la misma a la previa, el usuario recibirá un mensaje de error.

Validar cuenta recientemente creada - web

Descripción	Dado un código ya generado y recibido por el usuario, se identificará a este y se realizará la validación
Actor	Usuario no identificado (Usuario recién registrado)
Precondición	El usuario debe haber recibido un correo electrónico con un enlace de validación después de registrarse.
Postcondición	La cuenta del usuario se valida, permitiéndole acceder a las áreas privadas del servidor.
Flujo de eventos	● El usuario hace clic en el enlace de validación enviado por correo electrónico. ● El servidor recibe la solicitud y extrae el token de validación de la URL. ● El servidor verifica que el token de validación es válido y no ha expirado. ● El servidor marca la cuenta del usuario como validada en la base de datos. ● El servidor confirma la validación de la cuenta y permite el acceso a las áreas privadas.
Flujo alternativo	● Si el enlace ha expirado el usuario el usuario no podrá verificar su cuenta. Deberá proceder con el registro otra vez.

5.2.9. Gestión de trabajadores y empresas - server

En este apartado podemos ver el funcionamiento del sistema de gestión de trabajadores y empresas desde el punto de vista del server y sus diferentes casos de uso:

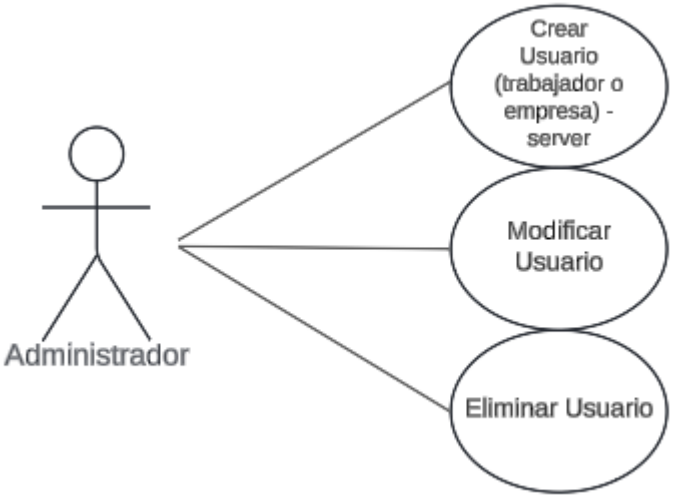

Figura 13. Diagrama de casos de uso del sistema de gestión de trabajadores y empresas en el server. (Creación Propia)

Creación de usuario - server

Descripción	Permite la creación de nuevos usuarios (camareros o empresas) en el sistema. Los detalles del usuario son almacenados en la base de datos.
Actor	Administrador
Precondición	El administrador debe estar autenticado.
Postcondición	Un nuevo usuario es creado y almacenado en la base de datos.

Flujo de eventos	• El administrador envía una solicitud con los detalles del usuario al servidor. • El servidor valida los datos proporcionados. • El servidor almacena los datos del usuario en la base de datos. • El servidor confirma la creación del usuario.

Modificación de usuario - server

Descripción	Permite actualizar la información de un usuario existente (camarero/a o empresa).
Actor	Administrador
Precondición	El administrador debe estar autentificado y el usuario debe existir en la base de datos.
Postcondición	Los datos del usuario se actualizan en la base de datos.
Flujo de eventos	• El administrador selecciona un usuario existente a modificar. • El administrador envía los datos actualizados al servidor. • El servidor valida los datos actualizados. • El servidor actualiza los datos del usuario en la base de datos. • El servidor confirma la actualización del usuario.
Flujo alternativo	• Si alguno de los datos no es válido el administrador recibirá una notificación de error.

Eliminar usuario - server

Descripción	Permite eliminar un usuario existente (camarero/a o empresa) del sistema.
Actor	Administrador
Precondición	El administrador debe estar autentificado y el usuario debe existir en la base de datos.
Postcondición	El usuario es eliminado de la base de datos.
Flujo de eventos	● El administrador selecciona un usuario existente para eliminar. ● El administrador envía una solicitud de eliminación al servidor. ● El servidor elimina el usuario de la base de datos. ● El servidor confirma la eliminación del usuario.

5.2.10. Gestión de ofertas de trabajo - server

En este apartado podemos ver el funcionamiento del sistema de gestión de ofertas de trabajo desde el punto de vista del server y sus diferentes casos de uso:

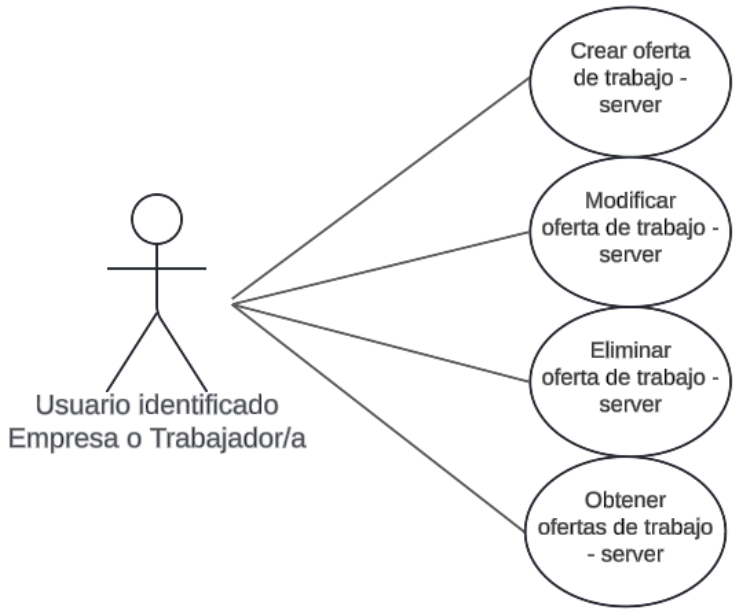

Figura 14. Diagrama de casos de uso del sistema de gestión de ofertas de trabajo en el server. (Creación Propia)

Creación de oferta de trabajo - server

Descripción	Permite a una empresa crear una nueva oferta de trabajo para un evento específico.
Actor	Empresa
Precondición	El administrador debe estar autenticado.
Postcondición	Una nueva oferta de trabajo es creada y asociada a un evento en la base de datos.
Flujo de eventos	● La empresa envía una solicitud con los detalles de la oferta de trabajo al servidor. ● El servidor valida los datos proporcionados. ● El servidor crea una nueva oferta de trabajo en la base de datos. ● El servidor confirma la creación de la oferta de trabajo.
Flujo alternativo	● Si alguno de los datos no es válido el usuario recibirá una notificación de error.

Modificación de oferta de trabajo - server

Descripción	De esta forma la empresa podrá actualizar los detalles de una oferta de trabajo existente dado un formulario.
Actor	Empresa
Precondición	La empresa debe estar autenticada y debe existir una oferta de trabajo para modificar.
Postcondición	Los detalles de la oferta de trabajo son actualizados en la base de datos.
Flujo de eventos	• La empresa selecciona una oferta de trabajo existente para modificarla. • La empresa envía los datos actualizados al servidor. • El servidor valida los datos actualizados. • El servidor actualiza los detalles de la oferta de trabajo en la base de datos. • El servidor confirma la actualización de la oferta de trabajo.
Flujo alternativo	• Si alguno de los datos no es válido la empresa recibirá una notificación de error.

Eliminar oferta de trabajo - server

Descripción	La empresa podrá eliminar una oferta de trabajo que ya no sea relevante o necesaria. También un administrador será capaz de esto en caso de que la oferta no cumpla con la normativa.
Actor	Empresa o Administrador
Precondición	La empresa debe estar autenticada y debe existir una oferta de trabajo para eliminar.
Postcondición	La oferta de trabajo se elimina de la base de datos.
Flujo de eventos	• La empresa selecciona una oferta de trabajo para eliminar. • La empresa envía una solicitud de eliminación al servidor. • El servidor elimina la oferta de trabajo de la base de datos. • El servidor confirma la eliminación de la oferta de trabajo.

Obtener todas las ofertas de trabajo - server

Descripción	Se obtendrá una lista de todas las ofertas de trabajo disponibles en el sistema, con la posibilidad de aplicar filtros según criterios como ubicación, fecha, tipo de evento, etc.
Actor	Camarero/a, Empresa, Administrador

Precondición	El usuario debe estar autenticado.
Postcondición	El servidor devuelve una lista de ofertas de trabajo que cumplen con los criterios o todos en caso de no haber filtros.
Flujo de eventos	● El usuario envía una solicitud para obtener una lista de todas las ofertas de trabajo. ● El servidor procesa la solicitud y aplica los filtros de búsqueda si están presentes. ● El servidor consulta la base de datos para recuperar las ofertas de trabajo que cumplen con los criterios especificados. ● El servidor envía la lista de ofertas de trabajo al frontend para ser mostrada al usuario.
Flujo alternativo	● Si la lista de ofertas de empleo está vacía dará error.

5.2.11. Gestión de eventos - server

En este apartado podemos ver el funcionamiento del sistema de gestión de eventos desde el punto de vista de la aplicación web y sus diferentes casos de uso:

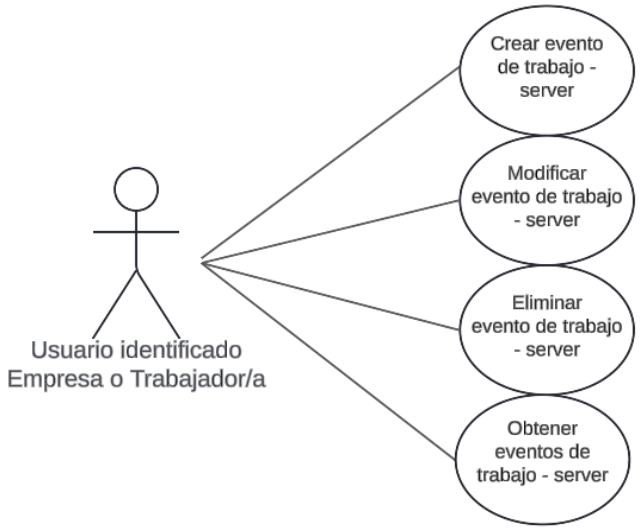

Figura 15. Diagrama de casos de uso del sistema de gestión de eventos de trabajo en el server. (Creación Propia)

Creación de un evento - server

Descripción	Dados unos datos proporcionados por una empresa se puede crear un nuevo evento en el sistema.
Actor	Empresa
Precondición	La empresa debe estar autenticada y tener permisos para crear eventos.
Postcondición	Se crea un nuevo evento y es almacenado en la base de datos, y puede estar disponible para que los trabajadores se apunten a él.

Flujo de eventos	• La empresa envía una solicitud al servidor para crear un evento, incluyendo todos los detalles necesarios. • El servidor valida que la empresa esté autenticada y que la solicitud contenga todos los datos requeridos. • El servidor crea un nuevo evento en la base de datos con la información proporcionada. • El servidor confirma la creación del evento

Modificar un evento - server

Descripción	Dados unos detalles permite a una empresa actualizar los detalles de un evento existente.
Actor	Empresa
Precondición	La empresa debe estar autenticada, tener permisos para modificar eventos, y el evento a modificar debe existir en la base de datos.
Postcondición	Los detalles del evento son actualizados en la base de datos con la nueva información proporcionada.
Flujo de eventos	• La empresa envía una solicitud al servidor para modificar un evento específico. • El servidor verifica que la empresa esté autenticada y tenga permisos para modificar el evento. • El servidor consulta la base de datos para verificar que el evento especificado existe. • El servidor actualiza el registro del evento en la base de datos con los nuevos detalles. • El servidor confirma la modificación del evento y envía una respuesta con los detalles actualizados del evento.

Eliminar un evento - server

Descripción	Esto permite a una empresa eliminar un evento existente del sistema, por ejemplo, si el evento es cancelado, también se permite a un administrador eliminar un evento cuando por ejemplo este no cumple con la normativa vigente.
Actor	Empresa o Administrador
Precondición	La empresa debe estar autenticada, o debe tratarse de un administrador, tener permisos para eliminar eventos, y el evento a eliminar debe existir en la base de datos.
Postcondición	El evento se eliminará de la base de datos y ya no estará disponible para los trabajadores.
Flujo de eventos	● La empresa o administrador envía una solicitud al servidor para eliminar un evento específico. ● El servidor verifica que la empresa esté autenticada y tenga permisos para eliminar el evento. ● El servidor consulta la base de datos para verificar que el evento especificado existe. ● El servidor elimina el registro del evento de la base de datos. ● El servidor confirma la eliminación del evento y envía una respuesta de confirmación.

Obtención de eventos - server

Descripción	Da la posibilidad a los usuarios de obtener una lista de eventos, ya sea todos los eventos disponibles o eventos específicos basados en ciertos criterios de búsqueda.
Actor	Camarero/a, Empresa, Administrador
Precondición	El usuario debe estar autentificado y tener los permisos necesarios para acceder a la información de eventos.
Postcondición	El usuario recibe una lista de eventos que cumplen con los criterios de búsqueda o todos los eventos si no se especifican filtros.
Flujo de eventos	● El usuario envía una solicitud al servidor para obtener una lista de eventos. La solicitud puede incluir parámetros adicionales como filtros de búsqueda. ● El servidor verifica que el usuario esté autenticado y tenga permisos para acceder a la información de los eventos. ● El servidor procesa la solicitud, aplicando cualquier filtro de búsqueda especificado. ● El servidor consulta la base de datos para recuperar los eventos que cumplen con los criterios especificados. ● El servidor envía la lista de eventos al frontend, formateada adecuadamente para ser presentada al usuario.

5.2.12. Gestión de ofertas de empleo - server

En este apartado podemos ver el funcionamiento del sistema de gestión de ofertas de empleo desde el punto de vista del server y sus diferentes casos de uso:

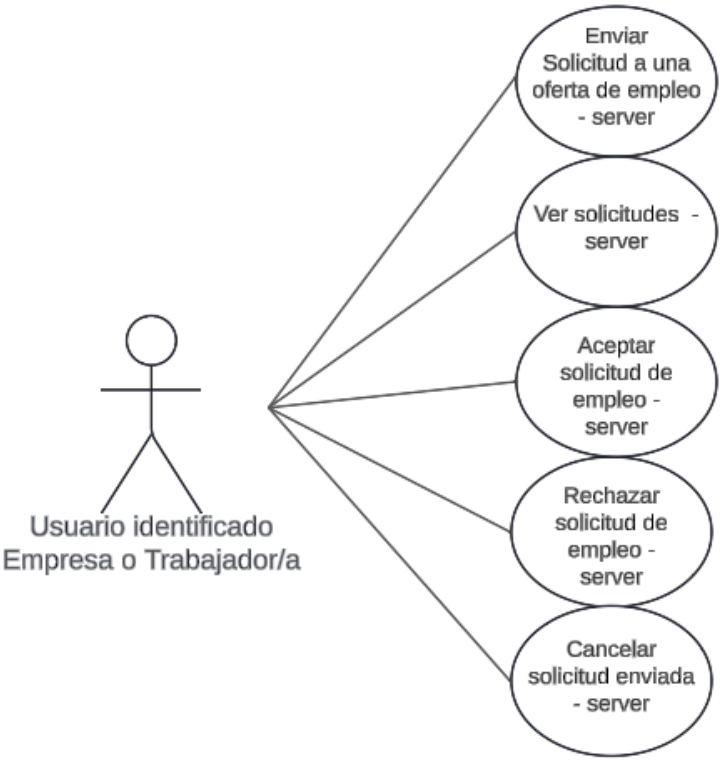

Figura 16. Diagrama de casos de uso del sistema de gestión de ofertas de empleo en el server. (Creación Propia)

Enviar solicitud a una oferta de empleo - server

Descripción	Dada una solicitud por parte de un camarero, le permite a este apuntarse a una oferta de empleo para ser revisado por la empresa.
Actor	Camarero/a (Trabajador/a eventual)
Precondición	El trabajador debe estar autenticado. La oferta de trabajo debe estar abierta a solicitudes.
Postcondición	Se crea una nueva solicitud en la base de datos, vinculando al trabajador con la oferta de trabajo.
Flujo de eventos	● El trabajador selecciona una oferta de trabajo disponible y envía una solicitud. ● El servidor verifica que el trabajador esté autenticado y que la oferta de trabajo esté abierta. ● El servidor crea un nuevo registro de solicitud en la base de datos, vinculando al trabajador con la oferta de trabajo. ● El servidor confirma al trabajador que su solicitud ha sido enviada exitosamente.

Cancelar solicitud de empleo enviada - server

Descripción	Dada una solicitud por parte de un camarero, le permite a este cancelar su solicitud a una oferta de empleo.
Actor	Camarero/a (Trabajador/a eventual)
Precondición	El camarero debe estar autentificado y haber enviado una solicitud a una oferta de trabajo que aún no haya sido aceptada o rechazada.
Postcondición	La solicitud es marcada como cancelada en la base de datos, y la empresa es notificada de la cancelación.
Flujo de eventos	● El trabajador selecciona la solicitud enviada y decide cancelarla. ● El servidor verifica que la solicitud esté en un estado donde pueda ser cancelada . ● El servidor actualiza el estado de la solicitud en la base de datos a "cancelada" o elimina la solicitud. ● El servidor confirma la cancelación al trabajador y notifica a la empresa, si es necesario.

Aceptar o rechazar solicitudes de empleo - server

Descripción	Tras la aceptación o rechazo por parte de la empresa, esta información debe ser guardada y actualizada en la base de datos.
Actor	Empresa
Precondición	La empresa debe estar autenticada, tener solicitudes pendientes, y la oferta de trabajo debe estar todavía abierta.
Postcondición	La solicitud es marcada como aceptada o rechazada en la base de datos, y el trabajador es notificado de la aceptación o rechazo.
Flujo de eventos	● La empresa selecciona una solicitud específica y decide aceptarla o rechazarla. ● El servidor verifica que la solicitud y la oferta de trabajo estén en un estado aceptable para aceptar o rechazar. ● El servidor actualiza el estado de la solicitud en la base de datos a "aceptada" o “rechazada”. ● El servidor envía una notificación al trabajador para informarle que su solicitud ha cambiado de estado.

Ver solicitudes de empleo - server

Descripción	Da la posibilidad a los usuarios de obtener una lista de solicitudes de empleo, ya sea todas las solicitudes o solicitudes concretas según unos filtros.
Actor	Empresa o camarero/a
Precondición	El usuario debe tener una oferta con solicitudes, o haber realizado solicitudes.
Postcondición	Se recibe una lista de solicitudes que pueden ser revisadas y gestionadas.
Flujo de eventos	● El usuarios solicita ver las solicitudes para una oferta de trabajo específica o en general. ● El servidor verifica que el usuario esté autenticado y tenga solicitudes recibidas o enviadas. ● El servidor consulta la base de datos para obtener todas las solicitudes vinculadas. ● El servidor envía la lista de solicitudes al frontend para ser mostrada.

5.3. Análisis de los riesgos

En el desarrollo de esta aplicación web que conecta trabajadores eventuales (camareros/as) con empresas de catering, es muy importante identificar y analizar los riesgos que podrían afectar el éxito del proyecto. Estos riesgos los podemos categorizar en diferentes tipos, incluyendo riesgos de aceptación, satisfacción, tecnológicos y de integración. A continuación se presentan estos riesgos y su análisis:

5.3.1. Riesgos de satisfacción

Estos riesgos se relacionan con la posibilidad de que los usuarios no acaben satisfechos con la experiencia de uso de la aplicación, lo que puede provocar una retención baja y que no tenga un uso continuo.

- **Experiencia de usuario (UX)**

  Una interfaz que acaba siendo demasiado compleja o poco intuitiva puede que los usuarios no acaben satisfechos.

  - *Forma de mitigarlo*: Diseñar desde el principio una interfaz que sea fácil de usar para el usuario. También realizar pruebas con usuarios reales y recibir su feedback para poder mantener la aplicación en un punto correcto de uso.

- **Funcionalidades insuficientes**

  Si la aplicación no cubre las necesidades que pueden tener los usuarios, o las funcionalidades críticas que la diferencien del resto, estos pueden optar por no usarla.

  - *Forma de mitigarlo*: Realizar un análisis de requisitos que esté detallado con los usuarios para poder estar seguros que las funcionalidades necesarias se encuentran presentes. Relacionado con el anterior riesgo, será necesario un feedback y actualización constante.

- **Rendimiento de la aplicación**

  Una aplicación lenta con muchos tiempos de carga, o errores constantes puede provocar que los usuarios acaben frustrados y dejando la aplicación.

  - *Forma de mitigarlo*: Pruebas y tests de rendimiento y usabilidad previo al lanzamiento general. También existe la posibilidad de implementar sistemas para monitorear la aplicación y así detectar los posibles problemas de forma más rápida.

5.3.2. Riesgos de aceptación

Los riesgos de aceptación son aquellos que están relacionados con la posibilidad de que los usuarios deseados (trabajadores temporales y empresas) no acepten o empiecen a usar la aplicación de la forma esperada.

- **Cambio de comportamiento de los usuarios**

  Actualmente los usuarios, tanto trabajadores como empresas usan métodos más tradicionales como llamadas de teléfono, Whatsapps o empresas de trabajo temporal. Pasar a usar una plataforma nueva dedicada puede que al principio cueste por la falta de costumbre o el pensar que la herramienta supone algo más complicado.

  - *Forma de mitigarlo:* Hacer guías sencillas para poder llegar a más gente, en la forma de videos cortos y ofrecer soporte a la gente, con preguntas frecuentes. Y como se ha mencionado anteriormente, mantener la aplicación con una interfaz intuitiva y sencilla.

- **Aceptación de las empresas**

  Las empresas de catering pueden no sentirse confiados como para cambiar sus formas actuales de conseguir trabajadores y pasar a depender de una plataforma digital que acaba de comenzar.

  - *Forma de mitigarlo:* Mostrar experiencias reales de éxito, presentar los beneficios tanto de seguridad como de conseguir el personal que más se adapte a ellos y demostrar cómo la aplicación puede mejorar y facilitar sus procesos.

5.3.3. Riesgos tecnológicos

- **Compatibilidad y mantenimiento de las tecnologías**

  Las tecnologías actuales elegidas (como Vue.js y Spring) pueden volverse obsoletas, o puede que sus actualizaciones dejen de ser compatibles con el sistema actual y las librerías usadas.

  - *Forma de mitigarlo:* Elegir tecnologías con un respaldo de la comunidad fuerte, este ya es el caso ya que tanto Vue.js como Spring son herramientas muy usadas y consolidadas, estando Vue.js no muy por detrás de Angular [37]. También será necesario planear las actualizaciones para que funcionen de forma regular y realizar pruebas de compatibilidad.

- **Escalabilidad del Sistema**

  Si el resultado de la aplicación acaba siendo exitoso, la base de usuarios crecerá, por lo que puede acabar teniendo problemas de rendimiento o capacidad.

  - *Forma de mitigarlo*: Realizar un diseño de la aplicación que tenga la escalabilidad en mente desde un primer momento. Plantear infraestructura en la nube para poder escalar de forma eficaz.

5.3.4. Riesgos de integración

Este riesgo tendrá que ser tratado con especial cuidado, ya que muchas características críticas dependen de esto, este riesgo hace referencia a la capacidad del sistema de integrarse efectivamente con otros sistemas y servicios.

- **Integración con servicios de mensajería:** La necesidad de integrar la aplicación con diversos servicios de mensajería (como notificaciones push, correo electrónico y chat interno) puede presentar desafíos.
    - *Forma de mitigarlo:* Utilizar APIs estándar y que cuenten con mucha documentación y apoyo de la comunidad, también realizar pruebas para poder estar seguros que la comunicación sea fiable.

- **Integración con diferentes librerías:** Pueden producirse fallos al contar con numerosas librerías especializadas para diferentes funciones.
    - *Forma de mitigarlo:* Mantener un control de las versiones utilizadas actualizado y asegurarse que estas versiones sean compatibles entre sí incluso después de actualizar las diferentes librerías.

5.4. Análisis e identificación de posibles soluciones

En este apartado se analizarán las diferentes tecnologías que se han considerado para el desarrollo de la aplicación web, tanto sus ventajas como sus posibles debilidades. Este análisis ha sido realizado teniendo en cuenta los riesgos mencionados en el apartado anterior, con el objetivo de seleccionar las herramientas y enfoques que mejor se cumplan los requisitos y objetivos del proyecto.

El análisis se estructurará en tres partes principales: primero, se abordará la arquitectura general de la aplicación, comparando y evaluando las posibles arquitecturas a implementar, como la modular monolítica y la basada en microservicios. A continuación, se desarrollarán las tecnologías elegidas para el desarrollo del servidor, centrándose en las soluciones más adecuadas. Finalmente, se examinarán las opciones para el desarrollo de la aplicación web, considerando las herramientas que permiten ofrecer una experiencia de usuario óptima y un rendimiento eficiente.

Finalmente, se explicarán las decisiones tecnológicas adoptadas, justificando la elección de cada componente y cómo estos se integran.

5.4.1. Análisis de posibles arquitecturas

La elección de la arquitectura fundamental es un paso muy importante en el desarrollo de cualquier aplicación, ya que define cómo se estructurará el sistema y cómo sus diferentes componentes se relacionan. En este análisis, se han considerado dos enfoques principales: Arquitectura Modular Monolítica y Arquitectura de Microservicios. A continuación, se presentan las ventajas y desventajas de cada una de ellas [22] [23].

- ***Primera Opción: Arquitectura Modular Monolítica***

    - *Ventajas:*

        - *Simplicidad y Rapidez en el desarrollo:*

Este tipo de aplicación permite desarrollar y desplegar la aplicación de manera sencilla y rápida, ya que todo el código está contenido en un solo proyecto. Esto nos facilitará la integración de nuevas funcionalidades y reduce la complejidad asociada con la configuración y el despliegue.

        - *Mantenimiento simple:*

Aunque en la arquitectura monolítica sea un único bloque de código, su modularidad interna permite organizar el proyecto en módulos bien definidos que separan cada una de las funcionalidades. Esto facilita la localización de problemas, la implementación de nuevas funciones, y el mantenimiento en general.

        - *Eficiencia en el desempeño:*

La comunicación entre los diferentes módulos de la aplicación es directa en este tipo de arquitectura, lo que puede resultar en un mejor rendimiento en comparación con la comunicación a través de servicios separados en una arquitectura de microservicios.

        - *Mayor facilidad en las operaciones:*

La administración y operación de la aplicación es más sencilla, ya que no es necesario gestionar el despliegue de múltiples servicios, lo que reduce la necesidad de herramientas y conocimientos adicionales sobre el despliegue simultáneo.

- *Desventajas:*

  - *Escalabilidad Limitada:*

Cuando la aplicación crezca en tamaño y complejidad, puede volverse más difícil de escalar horizontalmente. Sin embargo, este problema puede mitigarse al principio del proyecto con una buena estructuración, y la modularidad va a facilitar una futura posible transición a microservicios de ser necesario.

  - *Riesgo de un acoplamiento fuerte:*

Con este tipo de arquitectura existe el riesgo de que los módulos internos se interrelacionen fuertemente entre sí, lo que puede complicar el mantenimiento y crecimiento de la aplicación web. Sin embargo, con buenas prácticas de diseño modular este riesgo se puede llegar a mitigar.

- ***Segunda Opción: Arquitectura de microservicios***

  - *Ventajas:*

    - *Escalabilidad independiente:*

Con este tipo de arquitectura cada microservicio puede escalar de manera independiente, esto dependerá de la demanda, lo que permite un uso más eficiente de los recursos de personal y tiempo de desarrollo.

    - *Flexibilidad en las tecnologías:*

La arquitectura de microservicios permite utilizar diferentes tecnologías para diferentes servicios, lo que nos ofrece una mayor libertad en la elección de las herramientas y los frameworks que mejor se adapten a cada componente del sistema y casos específicos.

    - *Resiliencia y alta disponibilidad:*

Al tratarse de una arquitectura muy compartimentalizada, un posible fallo en uno de los microservicios no tiene porque afectar al resto de la aplicación, por lo que puede mantener la aplicación en funcionamiento pese a algún error.

- *Desventajas:*

  - *Complejidad operacional:*

La gestión, el despliegue y el monitoreo de múltiples servicios simultáneamente requiere una infraestructura más compleja y herramientas adicionales como puede ser con el uso de tecnologías DevOps, como contenedores (Docker) y orquestadores (Kubernetes).

  - *Desempeño potencialmente menor:*

La comunicación entre los diferentes microservicios puede ser más lenta debido a las llamadas a las diferentes APIs y la necesidad de manejar la latencia de la red, esto puede causar que el tiempo de respuesta durante el funcionamiento de la aplicación sea mayor.

  - *Mayor dificultad en el desarrollo inicial:*

La arquitectura de microservicios requiere configurar y gestionar una infraestructura distribuida puede aumentar el tiempo de desarrollo en las etapas iniciales del proyecto, y ya que en un primer lugar no se dispone de muchos recursos para desarrollar la aplicación, esto supone un gran problema.

5.4.2. Análisis de servicio de autenticación

Para poder hacer frente a los problemas mencionados de la manera más óptima, se optará por una arquitectura sin estado (stateless). Mediante este enfoque se elimina el uso de sesiones tradicionales, en su lugar, se utilizan tokens de autenticación que sirven para identificar a los usuarios y manejar las autorizaciones. Los sistemas stateless mejoran la escalabilidad y facilitan el mantenimiento, ya que cada solicitud es independiente y no depende de estados almacenados en el servidor, en una aplicación como la nuestra de un alto número de usuarios esta es la opción más viable [24].

Entre las tecnologías de autenticación, se consideran:

- **OAuth 2.0:** Se trata de un estándar de autorización que permite a los usuarios dar acceso limitado a sus recursos sin compartir credenciales, pero no es un sistema de autenticación por sí mismo.

- **OpenID Connect:** Este sistema sí que proporciona autenticación basada en OAuth 2.0 y permite obtener información de identidad del usuario.

- **JSON Web Tokens (JWT):** Por último este sistema puede servir para manejar la autenticación y la autorización utilizando tokens que son compactos y seguros. Es útil para aplicaciones sin estado como la nuestra, ya que nos permite almacenar y verificar información de identidad en cada solicitud sin la necesidad de gestionar sesiones.

5.4.3. Análisis de estándares de servidores

Actualmente, existen diversos estándares populares para desarrollar aplicaciones que sigan un enfoque stateless, ya que esta es la ruta que vamos a seguir. Entre las más comunes se encuentran las arquitecturas monolíticas y las de microservicios como hemos mencionado anteriormente. Se utilizará un estándar como es REST, esto nos permitirá diseñar interfaces desacopladas y modulares.

REST (Representational State Transfer) no es una arquitectura en sí misma, sino que es un conjunto de principios que guían el diseño de arquitecturas de software. Para que una aplicación se considere RESTful, debe de cumplir con ciertas restricciones:

Manipulación de recursos a través de un servidor: Los recursos (datos) se asocian a un servidor específico.

Operaciones CRUD: Debe permitir operaciones de “GET”, “POST”, “PUT” y “UPDATE”

Uso de URLs y métodos HTTP: Cada operación se debe realizar mediante una combinación de una URL y un método HTTP adecuado.

Intercambio de datos en formato JSON: Los datos se transfieren habitualmente en formato JSON, facilitando la interoperabilidad.

Códigos de respuesta HTTP: La aplicación debe usar códigos de estado HTTP para indicar el resultado de las operaciones, tanto para casos de éxito como para casos de error.

Como hemos mencionado anteriormente los dos caminos por los que nos vamos a decidir son: un servidor con arquitectura monolítica o uno basado en los microservicios [25].

En primer lugar, el servidor con arquitectura monolítica se trata de un sistema en el que todos los componentes y recursos de la aplicación están integrados en un solo proyecto y se ejecutan como una única unidad. Toda la funcionalidad de la aplicación se encuentra en un único código base, lo que simplifica el desarrollo, despliegue y gestión, pero puede dificultarnos la escalabilidad a medida que el proyecto crece, ya que sería necesario duplicar todo el código para solo expandir una de las partes.

Y en segundo lugar, un servidor con arquitectura de microservicios es otro enfoque donde la aplicación se divide en múltiples servicios independientes, cada uno responsable de una funcionalidad específica.

Estos microservicios se desarrollan, despliegan y escalan de manera autónoma, comunicándose entre sí a través de sus respectivas APIs. Esta arquitectura ofrece mayor flexibilidad, escalabilidad, y resiliencia, aunque conlleva una mayor complejidad operativa y de gestión. Pero por otra parte este tipo de arquitectura suele estar pensado para estar dividido en diferentes máquinas, por lo que el coste de la infraestructura será más elevado. En la figura 17, podemos ver una representación visual de las 2 implementaciones.

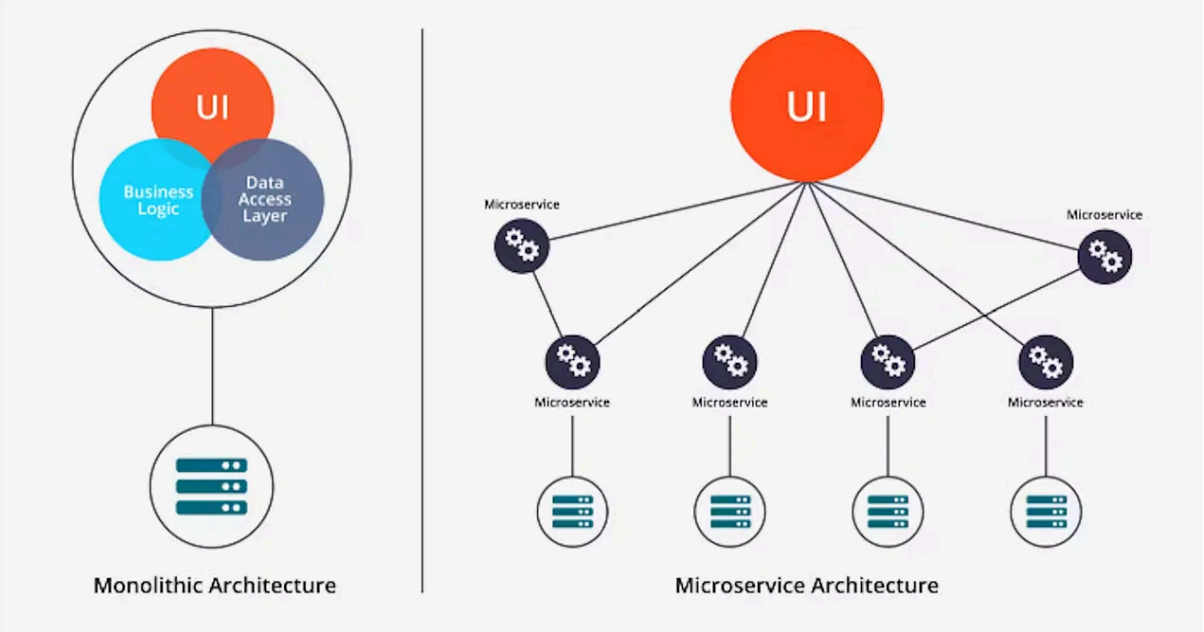

Figura 17. Diferencia entre arquitectura monolítica y de microservicios [25].(Jesús Henríquez, 2022)

5.4.4. Análisis de tecnologías de servidores

Ya que la aplicación web está basada en Spring, es decir debe construirse usando el lenguaje de programación Java, así que será más fácil mantener una constancia en este aspecto para facilitar el desarrollo.

Después de haber realizado un estudio sobre los diferentes frameworks cabe destacar 3 por sus características y casos de uso: "*Spring Boot*", "*Apache Struts*" y "*Grails*" [27].

Spring Boot es el más ampliamente utilizado de los 3, esto implica una comunidad muy grande y activa que puede ayudar a mitigar los riesgos vistos previamente. Ofrece una configuración automática que simplifica el desarrollo, también consta de un ecosistema rico que facilita la integración con múltiples servicios. Sin embargo, su consumo de memoria y tiempos de arranque son mayores en comparación con otras opciones, lo que puede llegar a ser un inconveniente. Pero pese a este inconveniente, sus ventajas superan ampliamente los inconvenientes para las necesidades de la aplicación [7].

Por otro lado, *Apache Struts* es un framework que sigue el patrón de diseño MVC (Modelo-Vista-Controlador). Struts proporciona una estructura bien definida para el desarrollo de aplicaciones web, con un fuerte enfoque en la organización del código y la separación de preocupaciones. Entre sus principales ventajas se encuentra el soporte nativo para formularios, validación y controladores centralizados. Además, también cuenta con una comunidad madura y extensa, lo que asegura un soporte continuo y una amplia disponibilidad de recursos. Por estos motivos Struts podría ser útil en el desarrollo de la aplicación [28].

Finalmente, *Grails* es un framework que también sigue el patrón MVC, pero está basado en Groovy, un lenguaje que se ejecuta sobre la Java Virtual Machine. Grails es conocido por su simplicidad y rapidez en el desarrollo, lo que lo hace muy útil para proyectos que necesitan una implementación rápida. Su enfoque sobre las configuraciones permite centrarse más en la lógica de negocio y menos en la configuración del entorno. Grails también se integra de forma fluida con tecnologías como Spring y Hibernate, lo que permite aprovechar el ecosistema Java con mayor agilidad y flexibilidad en el desarrollo. Es una opción muy útil cuando se busca crear aplicaciones web de manera rápida y eficiente, sin perder la capacidad de escalar y mantener el código a largo plazo, por lo que también puede ser un contendiente [29].

5.4.5. Análisis de tecnología de aplicación web

El enfoque principal para el desarrollo de la aplicación web es construir una interfaz de usuario interactiva y eficiente que funcione de manera independiente del servidor, sin necesidad de que este se encargue de renderizar las páginas. Este enfoque se conoce como aplicación de una sola página o SPA (Single Page Application).

Para el desarrollo frontend, existe una amplia variedad de librerías y frameworks que permiten realizar diversas tareas, lo que puede complicar la selección del más adecuado ya que hay una gran oferta disponible. Para simplificar la elección, se ha limitado la búsqueda a los frameworks más populares y ampliamente adoptados por grandes empresas y desarrolladores. Así se asegura que se utilicen herramientas que no solo sean eficientes y probadas en el mercado, sino también apoyadas por comunidades activas y con soporte continuo.

- **React.js** es una biblioteca de JavaScript desarrollada por Facebook, por lo cual no es un framework como tal. Está enfocada principalmente en la construcción de interfaces de usuario. Está basado en el uso de componentes, lo que facilita la creación de interfaces reutilizables y dinámicas. React ofrece un alto rendimiento, por lo que es una gran opción para aplicaciones interactivas que requieren interfaces de usuario complejas, como es el caso de nuestra aplicación. Tiene mucha flexibilidad y un ecosistema que permite integrarlo fácilmente con otras tecnologías, esto lo hace muy popular entre los desarrolladores de frontend [30].

- **Angular** es un framework desarrollado por Google, que se ha consolidado como una opción muy buena y frecuente para crear aplicaciones web dinámicas y de alto rendimiento. Está basado en TypeScript y sigue el patrón MVC. Angular ofrece una estructura sólida que incluye muchas herramientas integradas que le permiten abarcar prácticamente todos los aspectos del desarrollo SPA. Estas características lo convierten en una opción muy interesante para aplicaciones que requieren modularidad y facilidad de mantenimiento. También cuenta con el respaldo por parte de una gran comunidad y una gran empresa le colocan en un gran puesto como uno de los frameworks con mejor visión a largo plazo. Pero al mismo tiempo, todo este apoyo y comunidad detrás hace que la herramienta avance incluso demasiado rápido, por lo que siempre hay que tratar de estar actualizado para asegurar el funcionamiento correcto [31].

- **Svelte**, a diferencia de otros frameworks como React, que manejan gran parte del trabajo en el navegador del usuario mientras la aplicación está en funcionamiento, Svelte realiza esa carga de trabajo durante el proceso de compilación al construir la aplicación. Esto da como resultado un código JavaScript puro (vanilla) altamente optimizado. Esto elimina gran parte de la sobrecarga de tiempo de ejecución, resultando en aplicaciones más rápidas y ligeras. Svelte es fácil de aprender y permite crear aplicaciones con menos código. Pero como contras, Svelte es un framework con una comunidad mucho más pequeña detrás debido a su novedad, ya que por ahora no ha sido probado en grandes proyectos y su integración con otras herramientas estará menos probada [32].

- **Vue.js** es un framework progresivo de JavaScript que destaca por su ligereza y facilidad de aprendizaje. Está desarrollada por un antiguo empleado de Google y ofrece una solución equilibrada, combinando lo mejor de Angular y React, lo que le permite ser flexible y al mismo tiempo fácil de usar. Vue.js es de código libre por lo que su ecosistema, que incluye herramientas como Vue Router y Vuex, es muy amplio y está siendo mejorado constantemente. También cabe destacar una curva de aprendizaje sencilla y han creado una documentación muy completa. Como contras de Vue.js nos encontramos con un caso similar al de Svelte, es más reciente que React y Angular por lo que no está tan probado como estos dos, pero ya tiene una fuerte base y un gran número de aplicaciones web lo utilizan ya [18].

5.5. Soluciones elegidas para la aplicación

5.5.1. Autentificación

La aplicación utilizará *JSON Web Tokens* (JWT) ya que cumplen el requisito de permitir una autenticación stateless, reduciendo la carga del servidor y facilitando la escalabilidad. Los JWT son seguros al estar firmados digitalmente, lo que garantiza la integridad de la información y evita la necesidad que mantengamos sesiones del lado del servidor. Además, su uso facilita la integración con otros servicios y plataformas, gracias a su adopción como estándar en la industria. Esto es importante ya que posteriormente se podrá implementar OpenID en la aplicación para permitir el acceso con aplicaciones de terceros.

5.5.2. Arquitectura del servidor

Se ha optado por una arquitectura monolítica modular ya que ofrece ventajas importantes para el desarrollo de aplicaciones, como una gestión y despliegue simplificados, útiles para equipos pequeños o proyectos iniciales como es el caso de esta aplicación web. La modularidad dentro de esta arquitectura monolítica permite una separación clara de responsabilidades, facilitando el mantenimiento y la expansión. Además, los servidores con arquitectura monolítica suelen tener mejor rendimiento al evitar la sobrecarga de comunicación entre microservicios sobre todo en aplicaciones donde aún no tiene una gran base de solicitudes. Esta arquitectura también permite una posible transición en el futuro a una arquitectura de microservicios, proporcionando una base sólida para escalar y descomponer el sistema según las necesidades del negocio.

5.5.3. Tecnología en el servidor

En el caso de la tecnología en el servidor se ha optado por *Spring Boot* debido a su facilidad de configuración y rapidez en el desarrollo. Ya que se elimina la necesidad de configuraciones extensas, nos permite concentrarnos en otras partes del código.

Ofrece una amplia gama de características integradas, como es la gestión de dependencias, seguridad y soporte para APIs RESTful , lo que hace que sea una grán opción. Además, su gran comunidad y soporte extenso aseguran una evolución constante y una amplia disponibilidad de recursos, lo que facilita el mantenimiento y la escalabilidad de las aplicaciones.

Para la base de datos se utilizará Oracle con el protocolo de acceso JDBC por los mismos motivos, una comunidad amplia y altamente probada que nos ofrece seguridad a la hora de usarla.

5.5.4. Tecnologías para la aplicación web

Para terminar, se ha elegido Vue.js como la tecnología para el desarrollo de la aplicación web. Tras evaluar las diversas opciones, Vue.js ha demostrado ser la más adecuada debido a su flexibilidad, su rendimiento y su facilidad de aprendizaje. Su enfoque progresivo permite comenzar con un proyecto más pequeño y escalar sin necesidad de reescribir el código. Además, Vue.js cuenta con una amplia comunidad de desarrolladores y una excelente documentación, lo que facilita resolver problemas y la implementación de nuevas funcionalidades. También su capacidad para integrarse con otras tecnologías y herramientas modernas permite que la aplicación web sea mantenible y eficiente. Y por otro lado, la documentación de Vue.js permite tener prácticamente siempre información sobre las funcionalidades que se quieran implementar.

6. Diseño de la solución

En este capítulo se explicará la arquitectura del sistema, basada en el análisis y requisitos vistos en los anteriores apartados .

6.1. Arquitectura del sistema

Como hemos visto se pretende hacer un diseño muy modular y con posibilidad de ser escalado con facilidad, en el cual cada subsistema pueda funcionar independientemente, para así poder tener una mejor disponibilidad y su rendimiento.

Con las tecnologías elegidas mediante el análisis previamente expuesto, podríamos representar el sistema de la forma siguiente (figura 18).

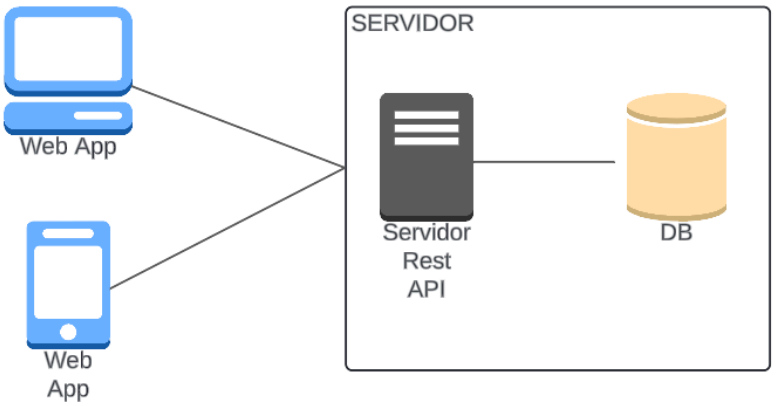

Figura 18. Diagrama de la arquitectura del sistema de la aplicación web. (Creación propia)

El sistema será accesible desde cualquier dispositivo con un acceso a alguno de los buscadores más comunes, ya sea desde móvil o desde un ordenador. Estos accesos se relacionarán con el servidor REST que se encuentra en el servidor, este obtendrá los datos que necesite de la base de datos.

El sistema tendrá el servidor principal, este servidor REST tendrá la arquitectura monolítica modular, por lo que será posible escalar o pasar a una arquitectura de microservicios de una forma más sencilla cuando el número de usuarios así lo requiera.

Este formato de desarrollo monolítico agilizará el proceso en gran medida en comparación a una arquitectura de microservicios.

6.2. Detalles del diseño

En estas secciones del apartado, se va a concretar más el diseño de por una parte la aplicación Web basada en Vue.js y el REST API Server. Ya que las dimensiones finales de cualquiera de los dos proyectos puede resultar en un tamaño muy importante con un nivel de esfuerzo también elevado, es considerado de buena práctica realizar un diseño de la arquitectura concreto y conciso para que pueda ser una ayuda a la hora del desarrollo.

6.3. Diseño de la arquitectura REST API Server

Basándonos en los requisitos y el análisis previo, se ha decidido seguir una estructura monolítica modular y el patrón de diseño de Modelo-Vista-Controlador (MVC). Esta combinación asegura una organización clara y coherente del código que permita su escalabilidad y comodidad a la hora de su mantenimiento.

Al seguir una arquitectura monolítica modular, la funcionalidad de la aplicación está contenida dentro de una única aplicación desplegable, pero organizada en módulos independientes. Estos módulos se agrupan por funcionalidades relacionadas y permiten una clara separación de responsabilidades dentro del servidor, lo que facilita su gestión y posible evolución [33].

La aplicación estará dividida en los siguientes módulos principales.

- **Módulo de Controladores**

  Este módulo gestiona la capa de presentación y entrada de la aplicación. Define los puntos de entrada (endpoints) de la API, recibe las solicitudes HTTP, y se encarga de la validación inicial de datos y autenticación de usuarios. Este módulo es el que implementa los controladores del patrón MVC, actuando como la capa que interactúa directamente con el usuario o cliente de la API.

- **Módulo de Servicios**

  Es donde se encuentra la lógica de la aplicación. Aquí es donde se implementa la mayoría del código, se realizan validaciones complejas, se gestiona la interacción entre diferentes partes de la aplicación. En este módulo se implementan también funcionalidades clave como la mensajería y la gestión de pagos, que pueden interactuar con sistemas externos o internos según sea necesario. Este módulo también se encarga de procesar la entrada de los controladores y preparar las respuestas que serán devueltas obtenidas del módulo de repositorio.

- **Módulo de Repositorios:**

  Se encarga del acceso a datos, la persistencia de la información en la base de datos y también proporciona una capa de abstracción sobre las operaciones CRUD (Crear, Leer, Actualizar, Eliminar). Aquí se definen las entidades del modelo, que representan las tablas de la base de datos, y se gestionan las consultas. Este módulo asegura que la lógica de acceso a datos esté separada de la lógica como tal de la aplicación, siguiendo los principios del patrón MVC.

- **Módulo de Utilerías**

  En este módulo se encuentran clases y métodos reutilizables que son utilizados en toda la aplicación. Incluye funciones auxiliares que no pertenecen estrictamente a ninguna de las capas mencionadas, pero que son esenciales para operaciones comunes, como el manejo de fechas y la encriptación por ejemplo.

- **Módulo de Seguridad**

  Este módulo es imprescindible para poder garantizar que el acceso a la aplicación y la información confidencial con la que trataremos esté protegida. Aquí se implementará toda la lógica relacionada con la seguridad, como la validación de tokens JWT, la protección de endpoints, y la integración con sistemas de autenticación externos si fuera necesario en un futuro, como con “OpenId connect”. En resumen, este módulo se asegura de que solo los usuarios autenticados y con los permisos adecuados puedan acceder a las diferentes partes de la aplicación.

6.4. Diseño de la base de datos de la aplicación

Para el diseño de la base de datos se pretende tener en cuenta la mayor cantidad de posibilidades para que esté lo más completa posible de cara al futuro, previendo así la necesidad de tener que alterar las tablas en un futuro, o por lo menos que estas modificaciones no supongan un gran cambio

Se ha separado en dos partes para poder observarla de forma correcta:

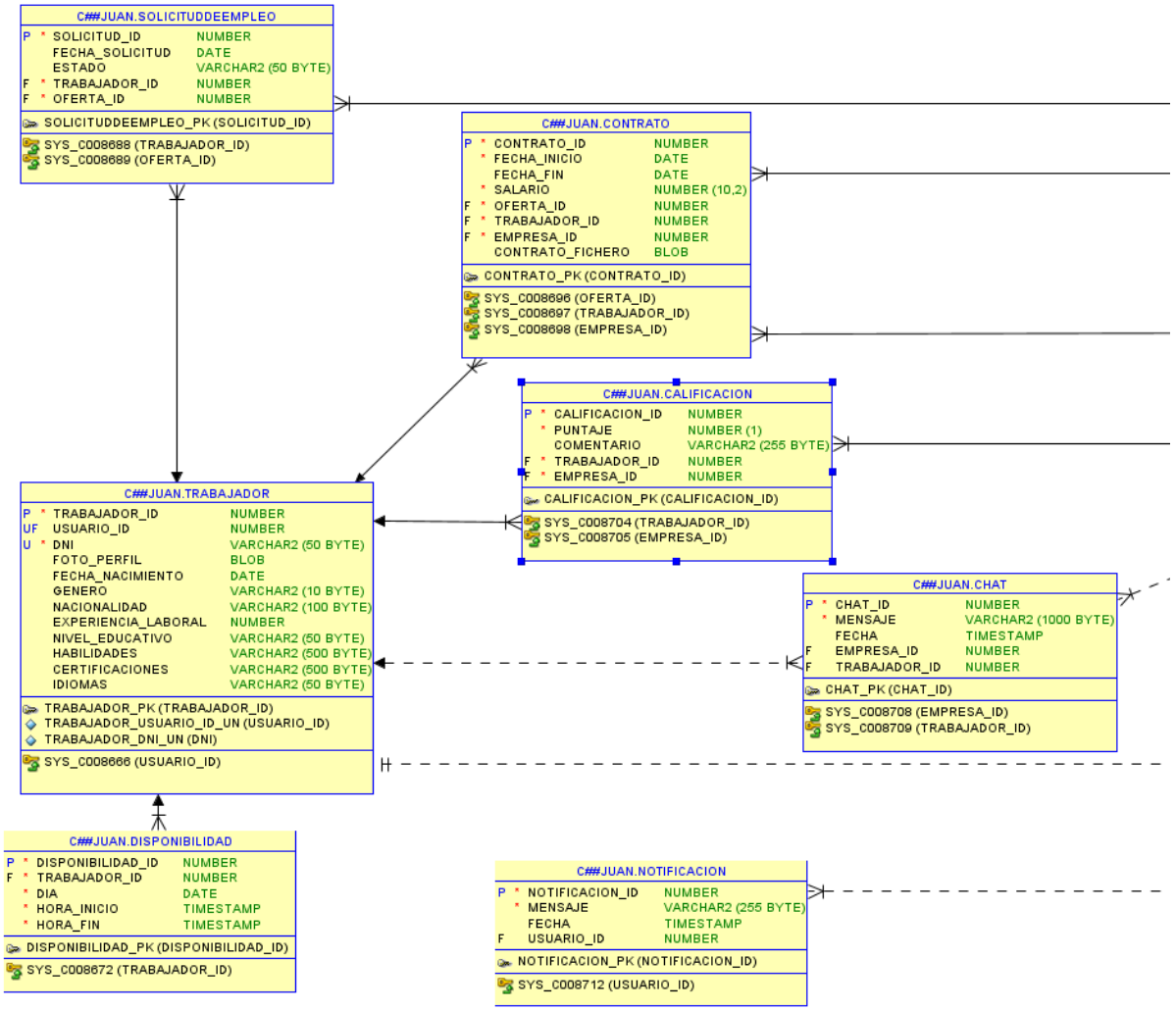

Figura 19. Diseño de la base de datos. Parte izquierda. (Creación Propia)

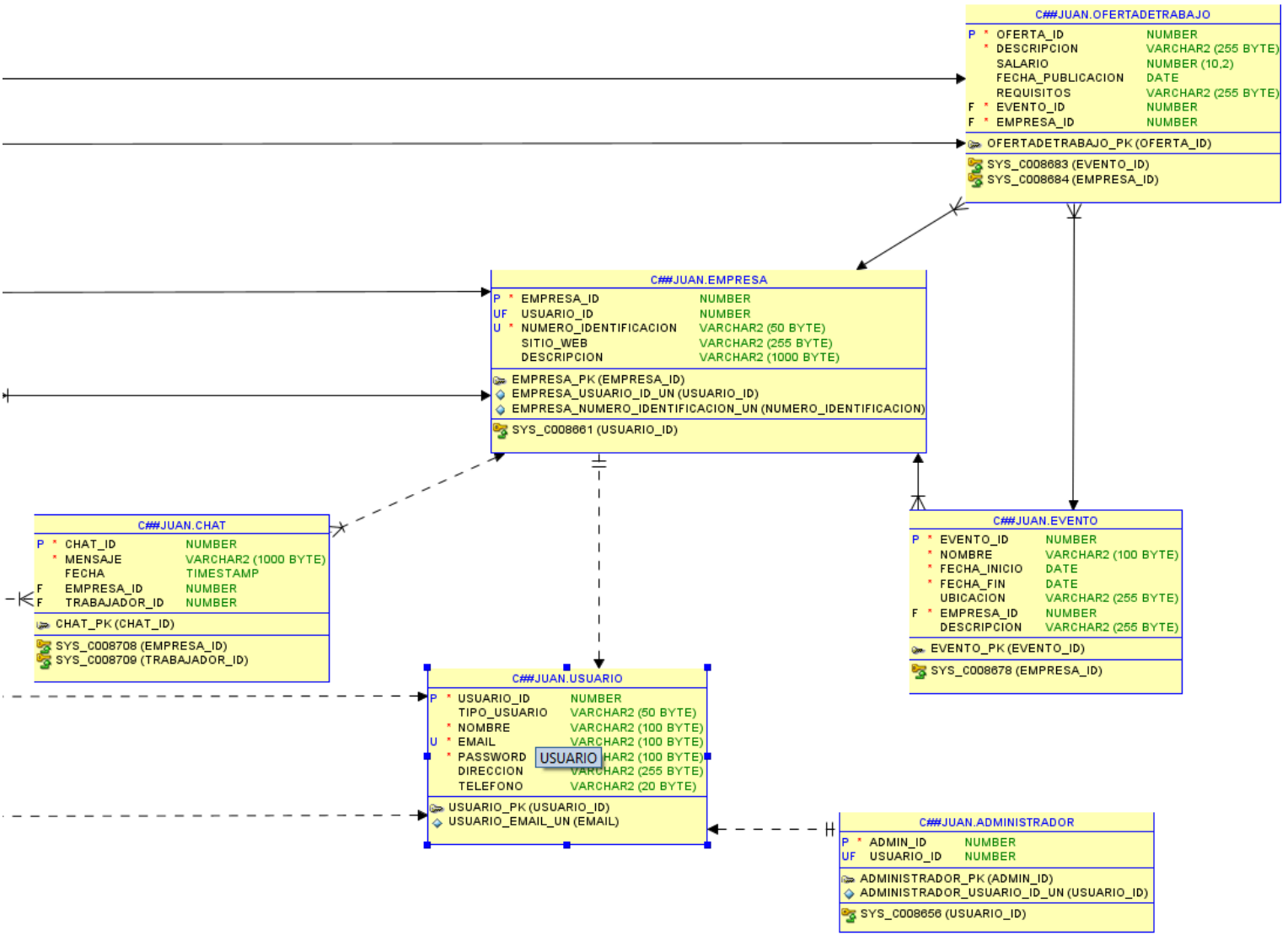

Figura 20. Diseño de la base de datos. Parte derecha. (Creación Propia)

**Comentarios:*

- *C##JUAN es el schema en el que se ha creado localmente, no tendrá el mismo aspecto en la versión final.*
- *Se ha repetido CHAT en las dos imágenes para poder facilitar el visualizado del centro.*

6.5. Diseño de la arquitectura Web App

Para este proyecto la arquitectura del frontend será finalmente con Vue.js, gracias a este framework podremos adaptarnos a las necesidades del proyecto incluso si estas van cambiando. Esta arquitectura se organiza en diversas secciones que separa y organiza los recursos y funcionalidades, pero manteniendo un nivel de cohesión elevado mientras que la dependencia sigue siendo baja entre los módulos [18].

La idea es mantener el diseño de las arquitecturas que utilizan los frameworks de los componentes web, por lo que se dividirá el proyecto en las siguientes secciones.

- **Vistas**

  Las vistas representan las diferentes páginas web de la aplicación, aquí se gestiona la interacción visual con el usuario. Estas vistas contienen todo el material visual y los eventos de usuario, como clics o formularios. Cada vista será responsable de renderizar la interfaz específica para una funcionalidad determinada, como la búsqueda de trabajos, la visualización de ofertas, o la gestión de perfiles.

- **Persistencia**

  La capa de persistencia se encargará de la comunicación con servicios externos, principalmente a través de peticiones HTTP hacia el backend previamente mencionado. Aquí se gestionan las solicitudes para obtener datos de la API, como las ofertas de trabajo disponibles o la información de los usuarios. Esta capa nos asegura que la aplicación pueda obtener y actualizar la información necesaria.

- **Router**

  Es el módulo que asocia las vistas con las URL correspondientes, gestiona la navegación dentro de la aplicación. En Vue.js, se utiliza Vue Router para manejar esta funcionalidad. Dado que la cantidad de rutas puede crecer considerablemente, el router se organizará de manera modular para poder facilitar su escalabilidad y mantenimiento en el futuro.

- **Almacén (Store)**

  Normalmente está implementado con Vuex en aplicaciones Vue.js, es responsable de gestionar el estado global de la aplicación. La función de este almacén es la de una caché centralizada, aquí se guarda la información que debe ser accesible desde diferentes partes de la aplicación, como los datos del usuario autenticado, la lista de ofertas de trabajo, o las preferencias del usuario. La gestión del estado en este almacén facilita la consistencia de los datos a lo largo de toda la aplicación y permite que su manipulación sea eficiente.

- **Componentes**

  Estos componentes son la base de la arquitectura de la aplicación. En Vue.js, los componentes contienen tanto la lógica como la interfaz de usuario para una funcionalidad concreta. Estos componentes son reutilizables y modulares, lo que permite construir las vistas de la aplicación como un conjunto de trozos. Conforme avance el desarrollo , se espera tener un gran número de componentes, y por eso mismo se les asigna un apartado entero. Estos componentes pueden variar en tamaño y dificultad, desde botones hasta formularios completos.

- **Recursos**

  Esta sección de recursos será la encargada de gestionar los archivos que la aplicación necesita, como imágenes, fuentes e iconos. Tener una sección específica para estos recursos nos permite asegurar ser capaces de organizar de una forma sencilla y que puedan ser accesibles desde cualquier parte de la aplicación sin dificultad. Esto tiene 2 ventajas, facilita la gestión de los recursos y ayuda a mantener el código limpio y ordenado.

7. Implementación de la solución

En este apartado se procede a realizar la implementación del proyecto con todas las características necesarias. Ya que se trata de un proyecto de final de carrera no se va a realizar una presentación extensa de todo el proyecto sino de ciertas partes que den a ver una imagen general del mismo.

7.1. Implementación del backend - Servidor REST

Para el desarrollo del backend se han utilizado las librerías de Spring, siguiendo el diseño de la arquitectura mencionada anteriormente. Este diseño se ha ido ampliando según avanzaba el desarrollo de la aplicación. Ya que Spring cuenta con muchas herramientas y librerías actualmente se han ido utilizando a lo largo del desarrollo para aumentar la velocidad de programación. Finalmente los paquetes de la aplicación tienen este aspecto como podemos ver en la figura 21:

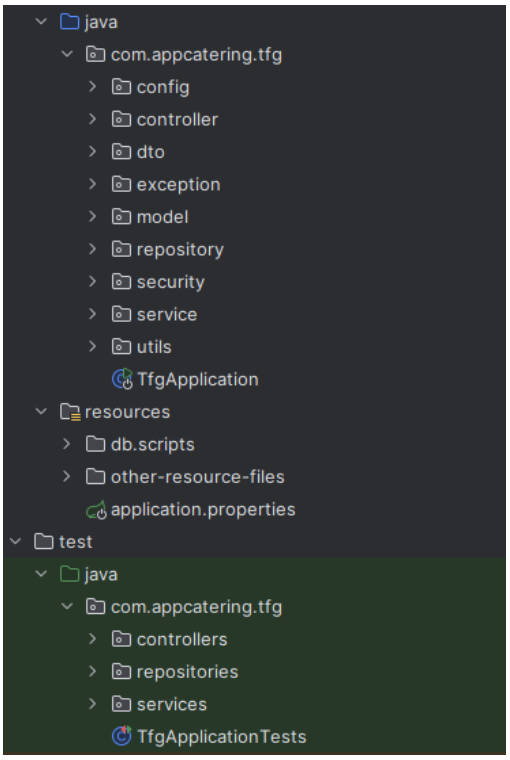

Figura 21. Estructura de los paquetes del backend. (Creación Propia)

7.1.1. Modelos

Una vez hecha la base de datos en oracle, el primer paso será la definición de los modelos de datos, es decir, las clases equivalentes a las tablas del modelo relacional. Estas se crearán en el paquete de com.appcatering.model.

Como comentario cabe destacar que también se hubiera podido hacer al revés, es decir, crear primero los modelos y después las tablas a partir de esto automáticamente, pero en este caso concreto, ya que la base de datos se había hecho previamente para el apartado *5.4*, se ha optado por obtener los modelos a partir de las tablas.

Para asegurar el correcto funcionamiento de los modelos será necesario prestar atención a todas las anotaciones necesarias ya que estas son las encargadas de relacionarse con la tabla de la manera correcta. Ya que existe una gran cantidad de anotaciones, se van a explicar las más comunes e importantes, para esto se va a tomar como ejemplo la clase trabajador, mostrado en la figura 22.

```java
@Getter
@Setter
@Entity
@Table(name = "TRABAJADOR")
public class Trabajador {
    @Id
    @Column(name = "TRABAJADOR_ID", nullable = false)
    private Long id;

    @MapsId
    @OneToOne(fetch = FetchType.LAZY)
    @OnDelete(action = OnDeleteAction.RESTRICT)
    @JoinColumn(name = "USUARIO_ID")
    private Usuario usuario;

    @Size(max = 20)
    @NotNull
    @Column(name = "DNI", nullable = false, length = 50, unique = true)
    private String dni;

    @Column(name = "FOTO_PERFIL")
    private byte[] fotoPerfil;

    @NotNull
    @Column(name = "FECHA_NACIMIENTO")
    private LocalDate fechaNacimiento;

    @Size(max = 10)
    @Column(name = "GENERO", length = 10)
    private String genero;

    @Size(max = 100)
    @Column(name = "NACIONALIDAD", length = 100)
    private String nacionalidad;
```

Figura 22. Modelo de la entidad Trabajador. (Creación Propia)

En primer lugar tenemos anotaciones de la librería de Lombok, @Getter y @Setter. Estas anotaciones nos generan automáticamente los getters y setters para todos los campos de la clase, ahorrando el tiempo de generarlos, y también manteniendo el código más limpio y menos extenso.

Las anotaciones de @Entity y @Table, son las responsables de indicar que se trata de una entidad JPA (tabla en la bbdd) y de especificar el nombre de la misma, sabiendo así lo que hay que mapear.

@Id nos indica cual es la clave primaria y @Column nos indica qué columna exactamente es la que estamos relacionando además de información relevante como sí es única.

Finalmente tenemos las anotaciones @OneToOne, @JoinColumn que son de mucha importancia a la hora de indicar las diferentes relaciones entre las tablas también existen @ManyToOne que deberá ser usada según el tipo de relación [5].

7.1.2. Repositorios

Tras crear los modelos será necesario crear los repositorios, esto no es más que la capa encargada del acceso a la base de datos, nos permitirá realizar operaciones de persistencia de datos (como crear, leer, actualizar y eliminar). Los repositorios pertenecen al patrón de diseño Modelo-Vista-Controlador, por lo que está pensado para facilitar el uso a la capa de servicios.

Cada entidad o tabla de la base de datos que se encuentra en el paquete *Model* cuenta con su propio repositorio, lo que facilita la organización y el mantenimiento del código. Los repositorios están organizados en el Módulo de Repositorios, manteniendo la estructura modular del sistema.

Para poder implementar estos repositorios no es necesario un gran esfuerzo, ya que teniendo las entidades solo hace falta crear una interfaz que extienda JpaRepository, le aportamos dos parámetros que serán el objeto de la entidad y el tipo correspondiente de la id y usaremos la notación @Repository.

Esto directamente nos dejará acceder a los métodos CRUD mencionados y para poder realizar consultas más complejas lo podremos hacer mediante anotaciones como @Query donde directamente podremos escribir nuestra consulta SQL, spring JPA trae integrada la posibilidad de mediante el nombre del método el mismo creará una query interna, pero para este proyecto se ha realizado mediante @Query debido a que las consultas SQL son más fáciles de personalizar y trabajar con ellas, podemos ver un ejemplo en la figura 23.

```java
@Repository
public interface TrabajadorRepository extends JpaRepository<Trabajador, Long> {

    @Query("SELECT t FROM Trabajador t WHERE t.idiomas LIKE %:idioma% AND t.experienciaLaboral >= :experienciaLaboral")
    List<Trabajador> findByIdiomaAndExperienciaLaboral(@Param("idioma") String idioma, @Param("experienciaLaboral") Long experienciaLaboral);
```

Figura 23. Ejemplo de un método en el Repositorio trabajador. (Creación Propia)

7.1.3. Controladores

Pese a que los controladores son una parte fundamental del servidor, es más, se podría decir que la más importante ya que es el punto donde empiezan todo el resto de procesos, esto no implica que su implementación sea muy difícil.

De la misma forma que los modelos y los repositorios, en Spring con un par de anotaciones en una clase puedes tener un controlador en funcionamiento. Estas anotaciones son @RestController para indicar que se trata del controlador, y luego tenemos dos versiones o usar @RequestMapping para indicar la url que que llamará a la clase y métodos, o usar las diferentes anotaciones específicas para operaciones CRUD.

A la hora de usar @RequestMapping será necesario que especifiquemos la ruta que nos llevará hasta aquí, es decir lo que el frontend tendrá que enviar como solicitud, manteniéndonos en el caso de los trabajadores podríamos tener algo tal que así (figura 24):

```java
@RestController
@RequestMapping("/trabajadores")
public class TrabajadorController {
```

Figura 24. Ejemplo de la cabecera del controlador de la entidad Trabajador. (Creación Propia)

Y un método con una anotación crud se vería tal que así (figura 25):

```java
    @GetMapping("/buscar")
    public List<Trabajador> buscarTrabajadores(
            @RequestParam String idioma,
            @RequestParam Long experienciaLaboral) {
        return trabajadorService.buscarTrabajadoresPorIdiomaYExperiencia(idioma, experienciaLaboral);
    }
```

Figura 25. Ejemplo método dentro del controlador de la entidad Trabajador. (Creación Propia)

Como podemos ver la anotación @GetMapping("/buscar") será necesaria para indicar que una llamada GET a la dirección base del servidor + /trabajadores + /buscar, será atendida aquí. También podemos ver las anotaciones @RequestParam, estos indican que para que este método concreto funcione, a parte de la ruta será necesario indicar los parámetros, también existen @PathVariable y @RequestBody para cubrir otro tipo de casos y requerimientos.

Las otra anotaciones necesarias para operaciones CRUD son @PostMapping,

@PutMapping y @DeleteMapping. Con cualquiera de estos métodos se puede trabajar de distintas maneras, ya sea mediante parámetros a parte o indicados en la ruta de la url [34].

7.1.4. Servicios

La capa de servicio será donde se gestione la mayor cantidad de lógica de la aplicación, estará menos relacionada con Spring que las otras capas, y se centrará más puramente en el código. Se le añadirá a la clase la anotación @Service sobre la definición de la clase, es posible añadir el nombre de la instancia ("*nombre*") aunque no es obligatorio suele ser útil para ciertos casos en los que se tienen muchos servicios del mismo tipo.

Con la anotación @Autowired será posible inyectar estos servicios de una forma sencilla, como podemos ver en la figura 26.

```
@Service
public class TrabajadorService {

    @Autowired
    private TrabajadorRepository trabajadorRepository;
```

Figura 26. Ejemplo cabecera del repositorio de la entidad Trabajador.(Creación Propia)

En este caso podemos ver como en el servicio del trabajador estamos inyectando el repositorio con la anotación que he mencionado.

Al ser esta parte la más dependiente del código más estándar, los servicios suelen ser los apartados más extensos de la aplicación, y a su vez serán los que más se relacionarán con otros módulos como bien puede ser con el de utils para validar que los datos sean correctos.

7.1.5. Seguridad

Como se ha mencionado anteriormente se utilizarán los tokens JWT, pero para poder manejar la seguridad de una mejor manera se utilizará la librería de Spring, Spring Security.

Esta librería nos facilitará el poder proteger la aplicación de diversos métodos de ciberataque que podríamos sufrir.

Ya que este apartado es uno de los más complejos, por no decir el más complejo, no se va a explicar en gran medida para no extender el apartado en demasiada medida y se va a explicar de forma más guiada.

Primero será necesario añadir las dependencias al pom.xml, para incluir spring security y JWT. Así podremos trabajar con los tokens necesarios para la autenticación.

Para gestionar los detalles de usuario relacionados con la autenticación, Spring Security nos ofrece una Interfaz llamada "UserDetails" con propiedades y métodos que deben ser sobrescritos por las entidades correspondientes en la implementación, estos métodos serán muy útiles para el control de los tokens, aquí podemos ver los métodos en la figura 27.

```
@Override
public Collection<? extends GrantedAuthority> getAuthorities() {
    return List.of();
}

@Override
public String getUsername() {
    return "";
}

@Override
public boolean isAccountNonExpired() {
    return UserDetails.super.isAccountNonExpired();
}

@Override
public boolean isAccountNonLocked() {
    return UserDetails.super.isAccountNonLocked();
}

@Override
public boolean isCredentialsNonExpired() {
    return UserDetails.super.isCredentialsNonExpired();
}

@Override
public boolean isEnabled() {
    return UserDetails.super.isEnabled();
}
```

Figura 27. Métodos que obtenemos de la interfaz UserDetails. (Creación Propia)

Este es un ejemplo de los métodos que serán implementados en las clases necesarias, en este caso “Usuario” al implementar “UserDetails”.

Posteriormente se creará un servicio que implemente “UserDetailsService” de Spring Security, lo tendremos que configurar para poder definir cómo se recuperan los detalles de los usuarios desde la base de datos.

Tras tener este servicio, crearemos el filtro JWT, este será el encargado de revisar las solicitudes HTTP recibidas. Si tras extraer el token recibido en caso de haberlo, este es válido, será válido para poder acceder a la aplicación.

Después será necesario configurar Spring Security para que utilice el filtro JWT, y definir las reglas de seguridad y acceso, aquí podemos especificar qué rutas podrán ser accedidas sin token, como es el caso de restablecer contraseña y bloqueará todas las demás.

También será necesario crear un controlador REST para poder autenticarse, esto manejará las solicitudes de inicio de sesión y el registro de los usuarios. Así, cuando un usuario se identifica con sus credenciales correctas, este controlador genera el token JWT y se lo devuelve al cliente.

Y finalmente también se ha decidido crear un DTO (“Objeto de transferencia de datos”) que defina la estructura de los datos para las solicitudes y para las respuestas [35].

7.2. Implementación del frontend - web Vue.js

Para el desarrollo de la aplicación web se han utilizado bastantes librerías y herramientas externas que han facilitado significativamente el proceso de implementación, como pueden ser extensiones del IDE de desarrollo Visual Studio Code. En lugar de describir todas estas dependencias, se ha optado por destacar aquellas más relevantes, y la configuración inicial. También se explicará más las herramientas que están más centradas al uso del framework Vue.js, que forma la base del front-end de la aplicación.

La estructura que se ha aplicado para este proyecto es la siguiente (figura 28):

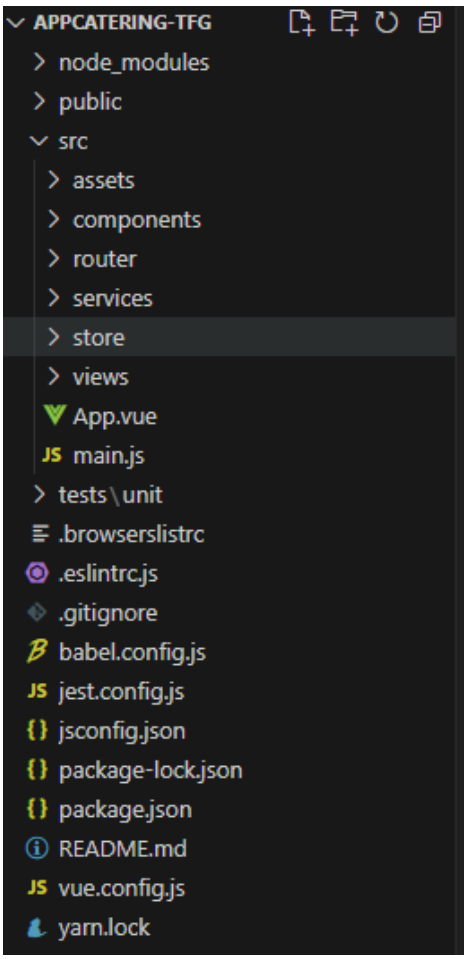

Figura 28. Estructura del frontend de la aplicación web en Vue.js. (Creación Propia)

Para poder comenzar a trabajar con Vue, se ha utilizado Node.js como base. Node.js es un entorno de ejecución de JavaScript que nos permitirá ejecutar código JavaScript en el servidor, fuera del navegador. Este entorno incluye npm (Node Package Manager), una herramienta que nos facilita la gestión de paquetes y dependencias en los proyectos. Una vez tenemos npm instalado, lo utilizaremos para instalar el CLI de Vue (Command Line Interface), que es una herramienta que simplifica mucho la creación, configuración y desarrollo de los proyectos con Vue.js.

Esta herramienta nos permite generar un proyecto de Vue desde cero de manera muy sencilla y con una estructura de carpetas bien organizada para poder mantener la escalabilidad. Además, durante la creación del proyecto, Vue CLI nos ofrece la posibilidad de elegir entre diferentes herramientas y configuraciones predeterminadas (esto lo podemos ver en la figura 29), como el uso de Vuex para la gestión del estado o Vue Router para la navegación entre vistas, lo cual agiliza el proceso de configuración inicial.

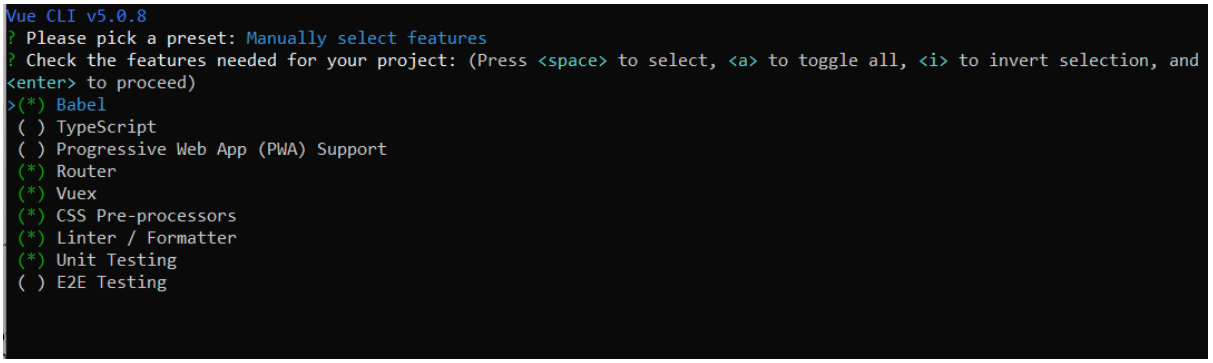

Figura 29. Vue CLI ejemplo de funcionamiento. (Creación Propia)

Para crear un nuevo proyecto, utilizaremos el comando *vue create*. Este comando iniciará el asistente que nos guiará a través de una serie de opciones, lo que nos permite personalizar el proyecto según nuestras necesidades. Podremos seleccionar entre configuraciones básicas o avanzadas, elegir si queremos soporte para TypeScript y configurar el uso de formateadores de código, entre otros. Este proceso nos permite tener un proyecto bien estructurado y listo para empezar a trabajar.

Una vez que el proyecto ha sido creado, es necesario iniciar el servidor de desarrollo para poder visualizar los cambios mientras trabajamos en la aplicación. Normalmente, esto se haría utilizando el comando *npm run serve*, que arranca un servidor local y abre la aplicación en un navegador web. Pero, para este proyecto se ha optado por utilizar "Yarn" en lugar de npm para gestionar las dependencias y scripts del proyecto.

Yarn es un gestor de paquetes similar a npm, pero que, según las experiencias anteriores usando *npm, yarn* nos ofrece ciertas ventajas como mayor velocidad y estabilidad al manejar dependencias, y según la experiencia de muchos, propia incluida, provoca menos fallos. Para iniciar el servidor de desarrollo con Yarn, el comando equivalente sería *yarn serve* [36].

7.2.1. Implementación de componentes

Como mencionado anteriormente, se ha utilizado Vue.js por diversos motivos, siendo uno de ellos el uso de componentes reutilizables lo que nos permite integrar conjuntamente el código HTML, CSS y JavaScript en un mismo archivo, concretamente un componente VUe.js tiene habitualmente 3 partes: **Template**, donde se define el HTML, **Script**, la lógica del componente, como los datos, métodos y los ciclos de vida finalmente tenemos también el apartado de **Style** donde se definen los estilos de CSS, aunque para este proyecto se ha agrupado mucha parte del css externamente.

Este es un ejemplo sencillo de como mostrar un componente en Vue (figura 30):

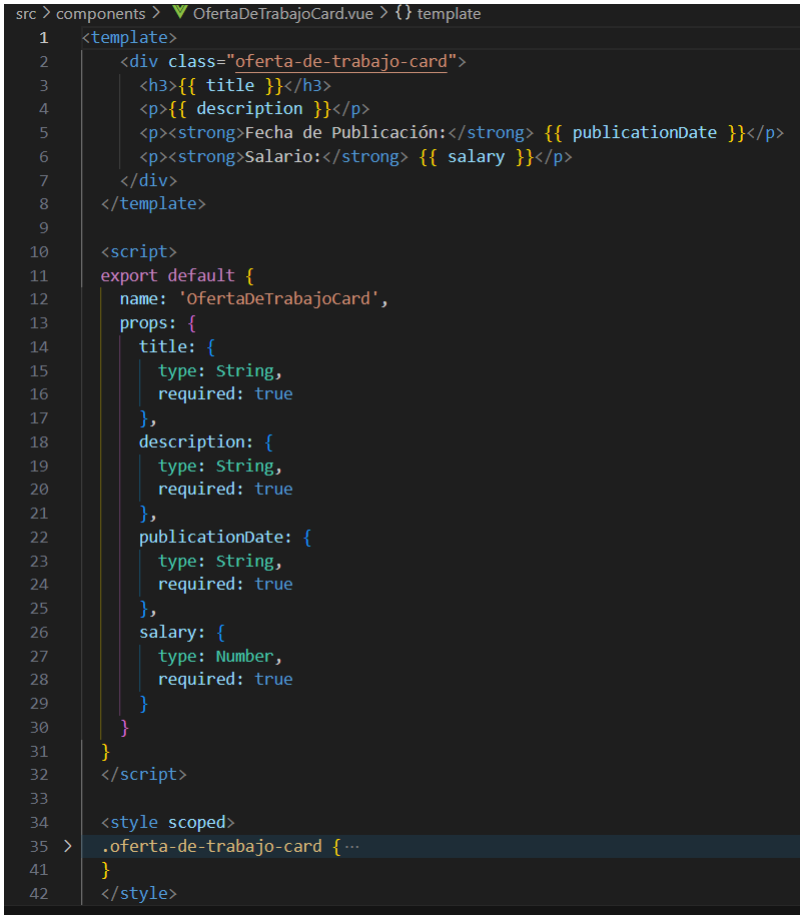

Figura 30. Componente sencillo de muestra de ofertas de trabajo. (Creación Propia)

Para este proyecto se han creado numerosos componentes que pueden ser reutilizados, estos se han guardado en la carpeta *components* para facilitar su organización.

7.2.2. Implementación de vistas

En Vue.js, las vistas son las diferentes pantallas o secciones de la aplicación web. Como la página de inicio o la página de perfil de usuario. Cada una de estas páginas es una vista, que estará compuesta de componentes.

Las vistas permiten organizar la aplicación en diferentes secciones más pequeñas y manejables. Esto facilita tanto el desarrollo como el mantenimiento de la aplicación, cada vista puede tener su propio contenido y diseño, y se puede navegar entre ellas utilizando enlaces o botones que mencionaremos posteriormente. Además, al dividir la aplicación en vistas, se mejora la claridad y la estructura del código, haciendo que sea más fácil de entender y modificar.

Para gestionar estas vistas y la navegación entre ellas, Vue.js utiliza una herramienta llamada *Vue Router*, que mencionaremos en el siguiente apartado.

7.2.3. Implementación del Router

El router es la parte principal que gestiona la navegación entre las diferentes páginas o vistas de la aplicación sin necesidad de recargar completamente la página. Esto nos proporciona una experiencia de usuario fluida y similar a una aplicación de escritorio. Vue Router es la biblioteca oficial para manejar el enrutamiento en aplicaciones Vue.js.

Se encarga de las asociaciones entre URLs y vistas, y es donde se definen los permisos de acceso. Por ejemplo, las vistas de los contratos sólo deben ser accesibles para usuarios autenticados y relacionados con el mismo, si un usuario no autenticado intenta acceder, debe ser bloqueado o redirigido a otra vista.

Para implementar routers, Vue.js ofrece una librería llamada *Vue Router*, que incluye todas las herramientas necesarias. Para esta aplicación, el router se encuentra en la carpeta “router” dentro de la estructura del proyecto, esto suele ser estándar en el desarrollo con Vue.js, esta carpeta se puede ver previamente en la figura.

7.2.4. Implementación de la Store

En las aplicaciones de Vue.js, la store se refiere a un estado centralizado donde se almacenan y gestionan los datos que se compartirán entre los diferentes componentes de la aplicación. Vuex, es la biblioteca oficial de Vue.js para manejar este estado centralizado y mantener la consistencia de los datos a lo largo de la aplicación, facilitando así la gestión y el mantenimiento tanto actual como futuro.

Usar una store será fundamental cuando se desarrolla una aplicación como la que se está creando, ya que necesitamos manejar varios datos importantes que deben ser accesibles y modificables desde múltiples partes de la aplicación. Por ejemplo, las ofertas de trabajo deben ser visibles para los trabajadores y actualizables por las empresas; los eventos deben poder ser creados y gestionados por las empresas, y los trabajadores deben poder ver los eventos disponibles; además, tanto las empresas como los trabajadores tienen perfiles que necesitan ser gestionados, actualizados y accesibles desde varias partes de la aplicación.

La store centraliza el estado en vez de que cada componente maneje su propio estado, lo cual podría llevar a inconsistencias. Así, por ejemplo, una lista de ofertas de trabajo estaría en la store para que cualquier componente pueda acceder a ella.

Con Vuex, las modificaciones del estado se realizan a través de acciones y mutaciones. Las mutaciones son funciones síncronas que cambian el estado, mientras que las acciones pueden ser asíncronas y se utilizan para hacer llamadas a APIs o a otras operaciones antes de cometer mutaciones. La store en Vue.js es reactiva, lo que significa que cualquier componente que dependa de una parte de información almacenada en la store se actualizará automáticamente cuando ese estado cambie, lo cual es crucial para mantener la interfaz de usuario sincronizada con los datos actuales.

En la implementación de la store se incluyen varios módulos para organizar el estado relacionado con diferentes aspectos de la aplicación. Como un módulo de ofertas para mantener una lista de ofertas de trabajo y permitir la creación, edición y eliminación de ofertas; un módulo de eventos para almacenar y gestionar la información de eventos y un módulo de usuarios para gestionar los perfiles de usuario, permitiendo la actualización de información personal y profesional.

Ya que la store de la aplicación ha acabado siendo bastante compleja, no es el mejor ejemplo para ver su funcionamiento de forma clara. Este es un ejemplo sencillo dado por la documentación oficial (figura 31).

```js
import { createApp } from 'vue'
import { createStore } from 'vuex'

// Create a new store instance.
const store = createStore({
  state () {
    return {
      count: 0
    }
  },
  mutations: {
    increment (state) {
      state.count++
    }
  }
})

const app = createApp({ /* your root component */ })

// Install the store instance as a plugin
app.use(store)
```

Figura 31. Store de ejemplo ofrecida por la documentación oficial. (vuejs.org, 2024) [19]

Y desde un componente podremos acceder de la siguiente forma (figura 32):

```
methods: {
  increment() {
    this.$store.commit('increment')
    console.log(this.$store.state.count)
  }
}
```

Figura 32. Ejemplo de acceso a la store desde un componente. (vuejs.org, 2024) [19]

7.2.5. Implementación de la capa de servicio

Esta capa de servicio se crea para organizar y manejar la lógica relacionada con las interacciones con APIs externas o en este caso con los servicios backend de una manera más estructurada y separada del código del componente, es una buena práctica mantener la lógica de la comunicación con el backend separada del resto del código de la aplicación así se evita repetir código constantemente.

En términos simples, se trata de un archivo JavaScript que tiene funciones dedicadas a realizar operaciones específicas, como hacer las peticiones HTTP para obtener los datos de eventos o manejar la autenticación de clientes y empleados en este caso concreto.

Como se ha mencionado la reutilización del código se puede evitar en gran medida con esta capa. Al centralizar las funciones de llamadas a la API en un servicio, estas funciones pueden ser reutilizadas en diferentes partes de la aplicación sin duplicar código.

7.3. Capturas de pantalla de la versión actual de la aplicación

En este apartado se procede a enseñar el estado de la aplicación al momento de la entrega, por lo que algunas funcionalidades no están finalizadas al 100%, o su parte visual no entona con el resto de la aplicación por lo que no se muestran como ejemplo, al tratarse de un proyecto de una gran magnitud las vistas no necesariamente serán las finales y será necesaria su revisión tanto de utilidad como por parte de UI/UX antes de la entrega al público.

Se ha utilizado un estilo de letra de pseudo PascalCase en algunos formularios para tener más claridad y desde un punto de vista estético. También se han utilizado componentes de Vuetify para mantener una constancia en el proyecto.

Figura 33. Ventana principal de acceso. (Creación Propia)

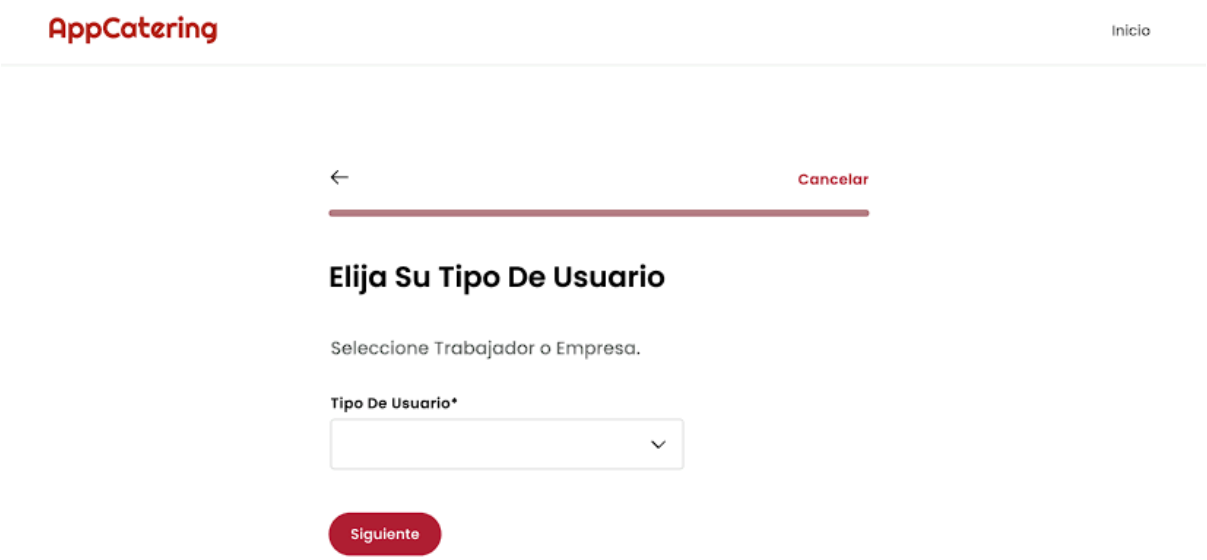

Figura 34. Primera Ventana registro. (Creación Propia)

Figura 35. Segunda Ventana Registro Empresa. (Creación Propia)

Figura 36. Tercera Ventana Registro Empresa. (Creación Propia)

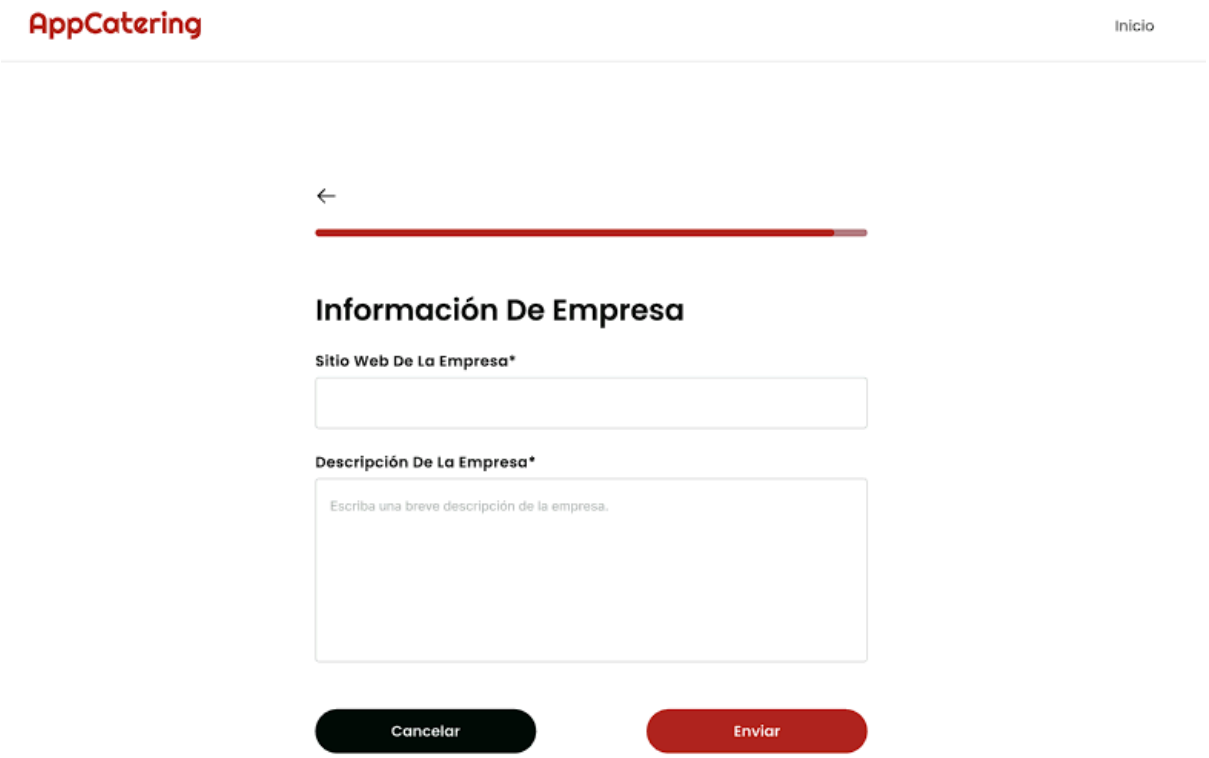

Figura 37. Cuarta Ventana Registro Empresa. (Creación Propia)

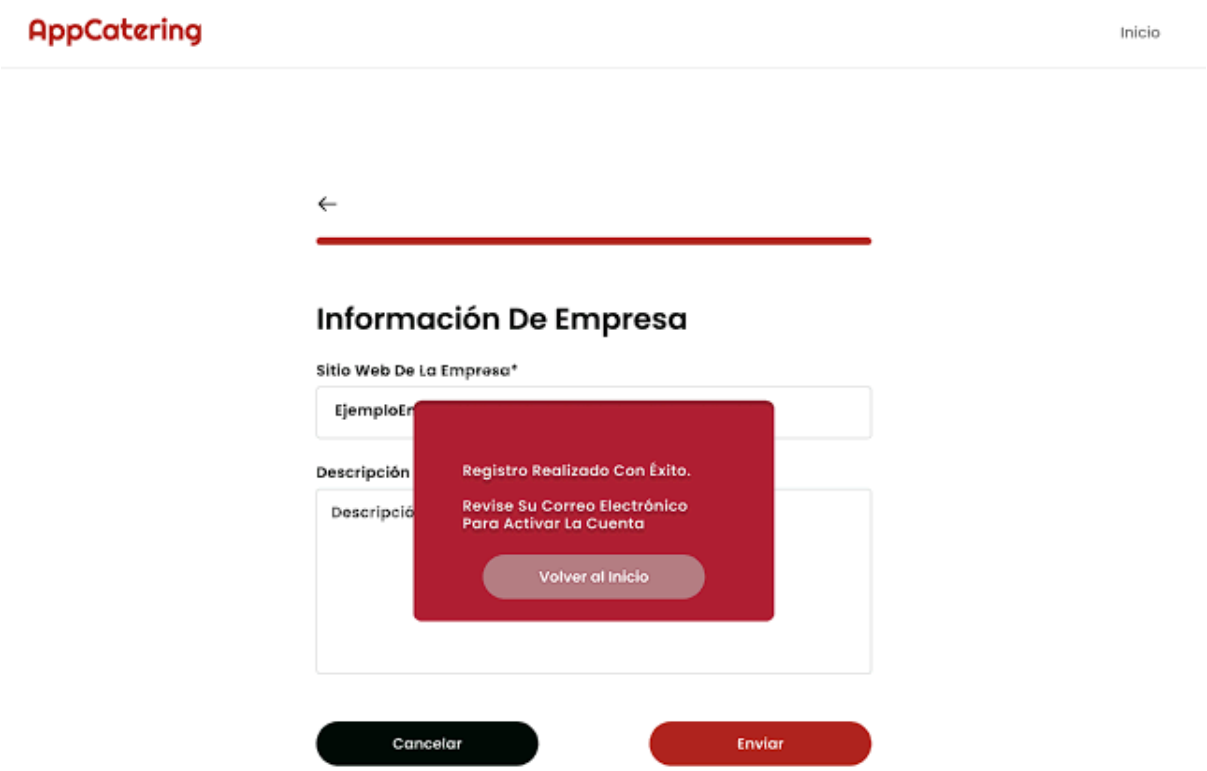

Figura 38. Ejemplo Éxito Ventana Registro Empresa. (Creación Propia)

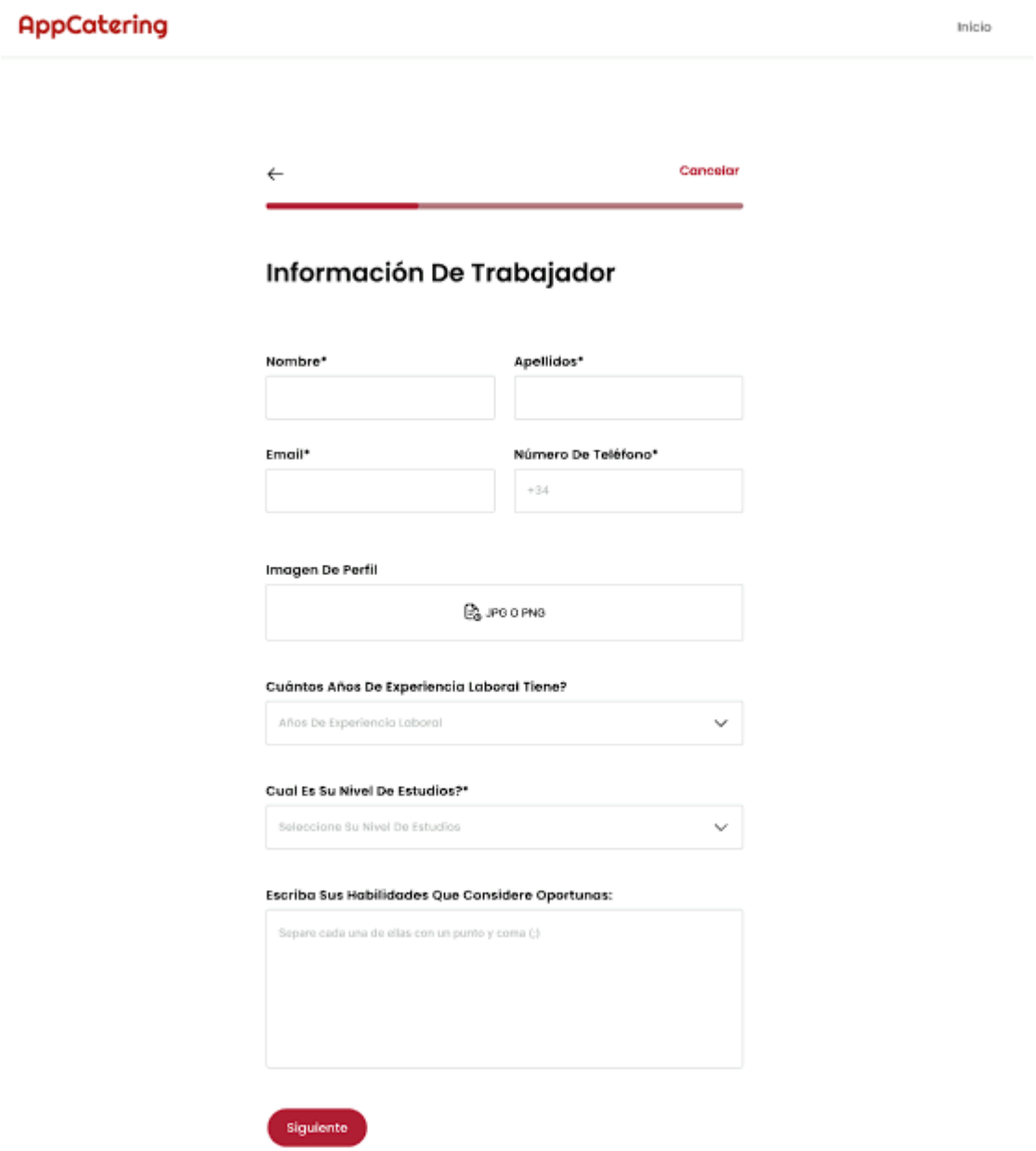

Figura 39. Segunda Ventana Registro Trabajador. (Creación Propia)

Figura 40. Tercera Ventana Registro Trabajador. (Creación Propia)

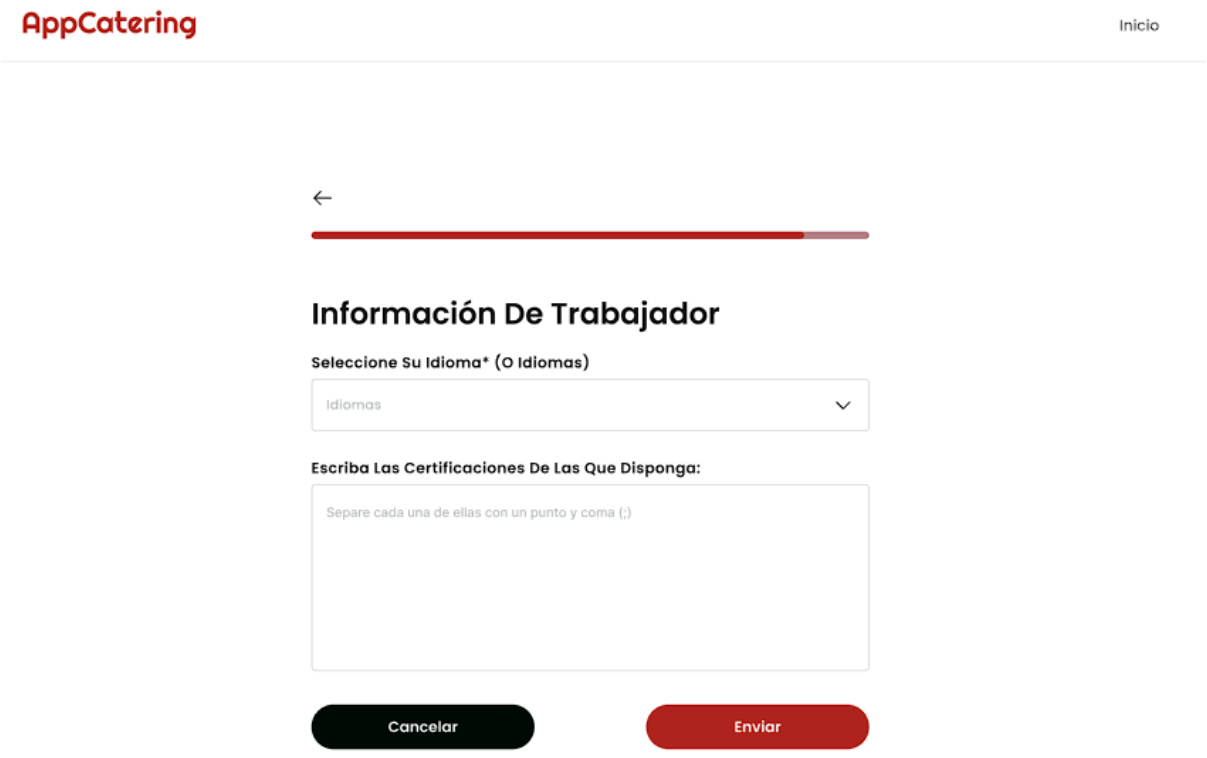

Figura 41. Ventana Final Registro Trabajador. (Creación Propia)

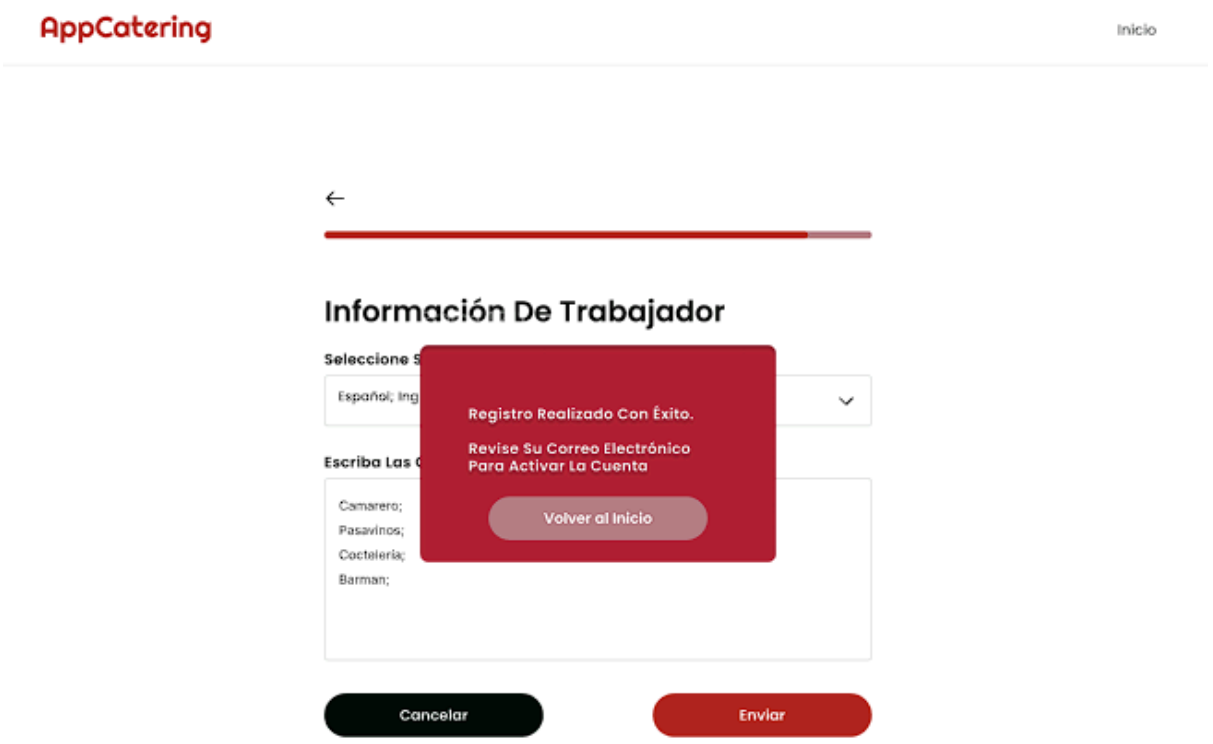

Figura 42. Ejemplo Registro Completo Trabajador. (Creación Propia)

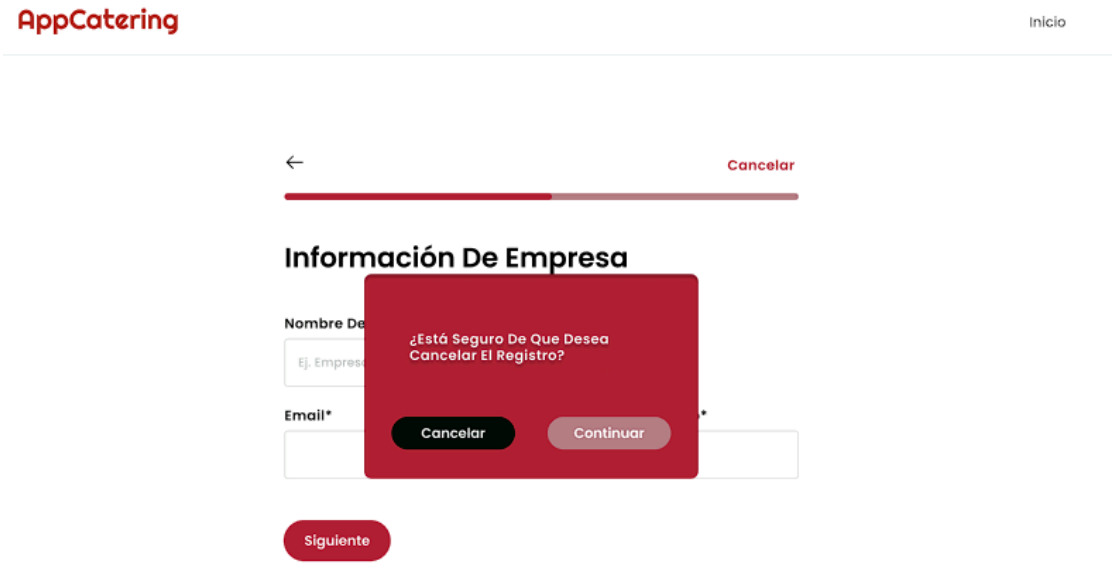

Figura 43. Ejemplo Registro Completo Trabajador. (Creación Propia)

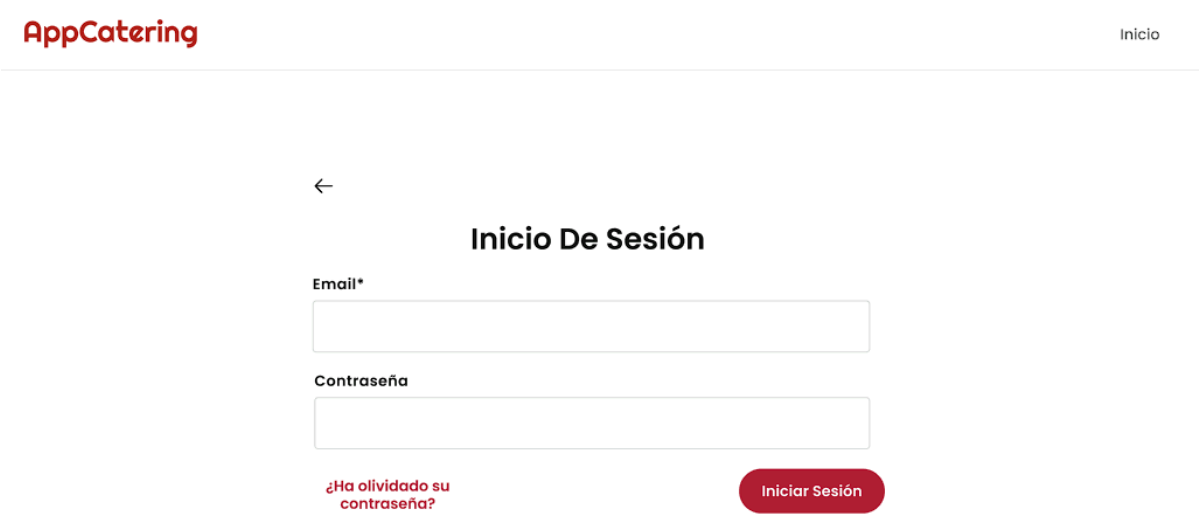

Figura 44. Ventana de Login. (Creación Propia)

Figura 45. Ventana de Recuperación De Contraseña. (Creación Propia)

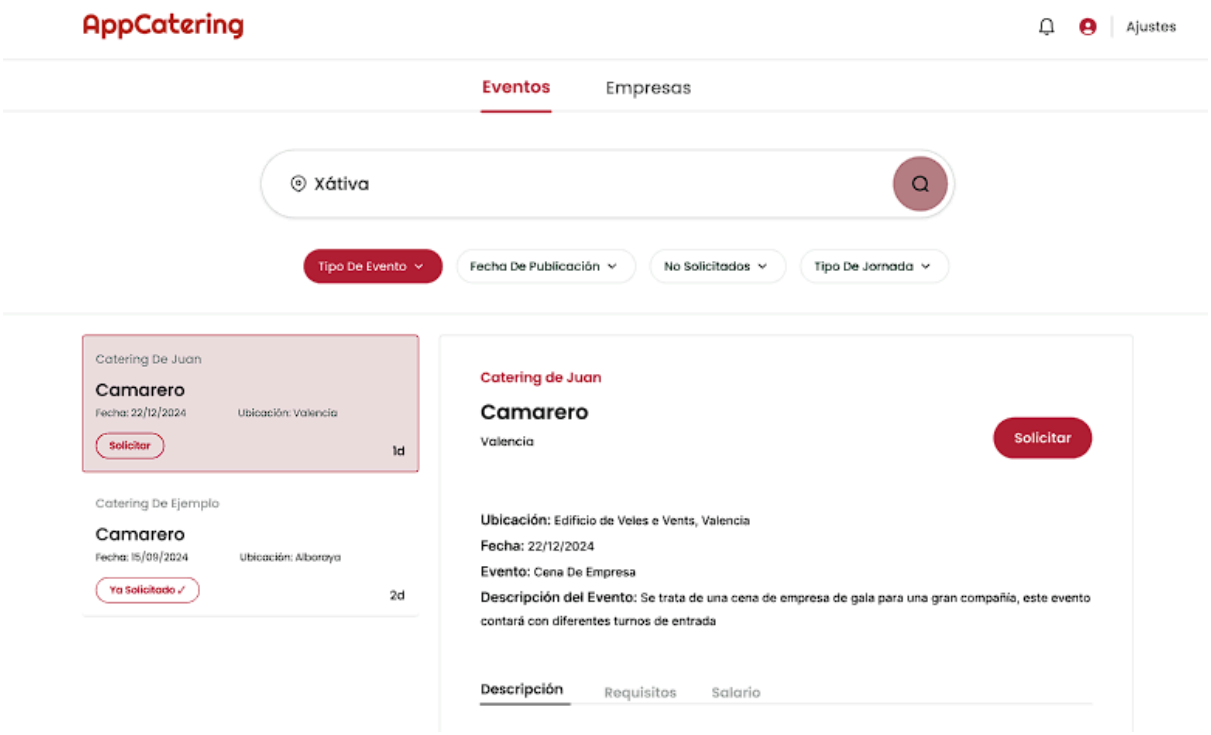

Figura 46. Ejemplo Funcionamiento Búsqueda y solicitud parte 1. (Creación Propia)

Figura 47. Ejemplo Funcionamiento Búsqueda y solicitud parte 2. (Creación Propia)

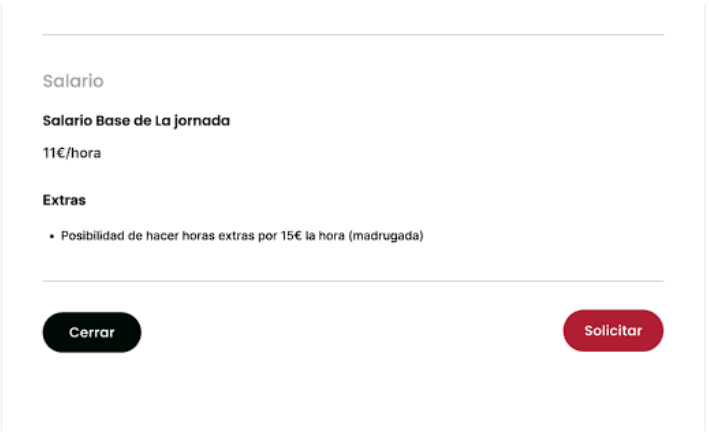

Figura 48. Ejemplo Funcionamiento Búsqueda y solicitud parte 2. (Creación Propia)

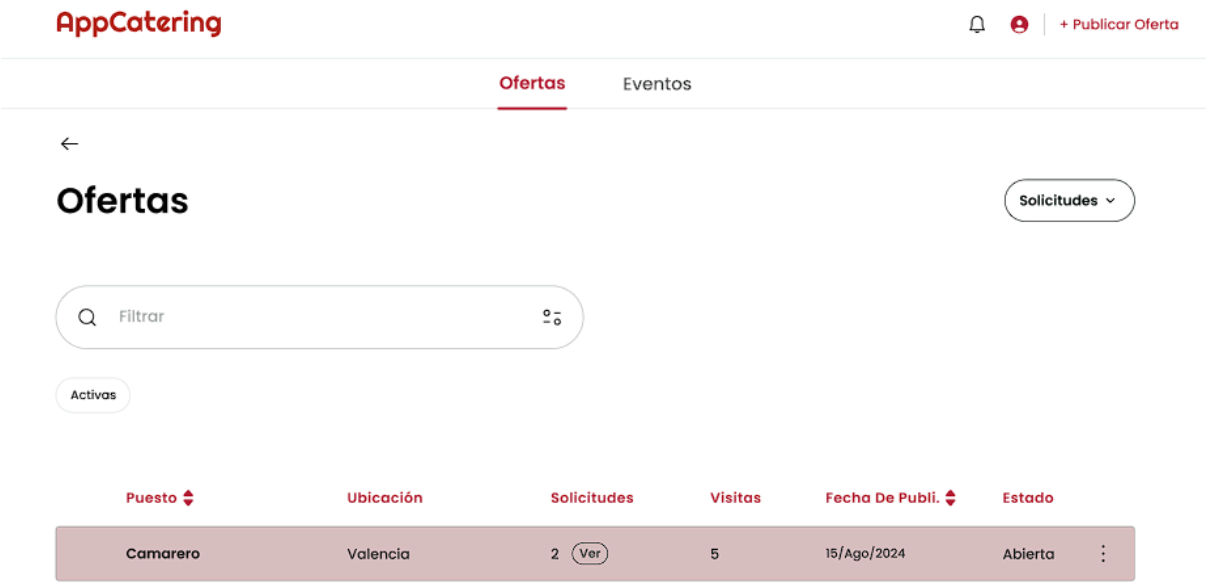

Figura 49. Ejemplo Panel de visualización de ofertas (Creación Propia)

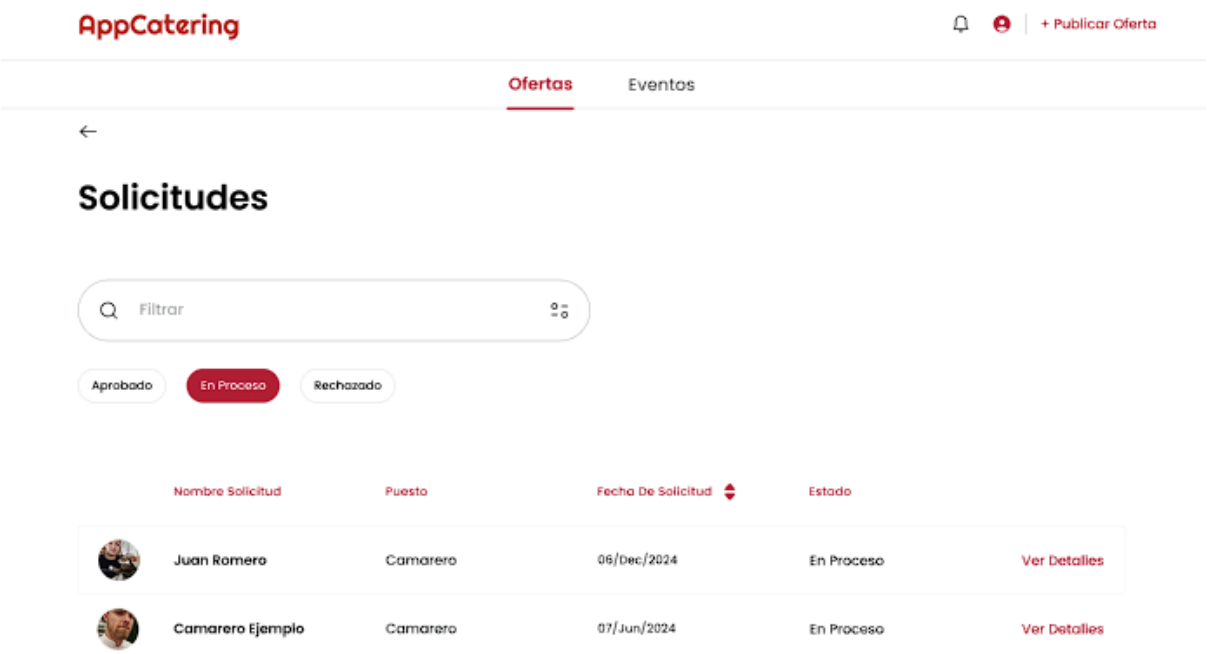

Figura 50. Ejemplo Panel de Control Solicitudes (Creación Propia, imágenes propias con permiso)

8. Testing

En el desarrollo de la aplicación web, el testing ha sido una parte fundamental para garantizar la calidad y estabilidad del software. Se ha preparado el terreno para poder aplicar las mejores prácticas para asegurar que la aplicación funcione correctamente en todos los escenarios posibles y que los errores se detecten y corrijan de manera temprana.

Estos no son solamente los tests realizados sino los que se pretenden aplicar en un futuro.

8.1. Testing Unitario

El testing unitario es una técnica esencial que nos permite verificar que cada componente individual de la aplicación funcione como se espera. Hemos implementado pruebas unitarias para los componentes principales del front-end utilizando Vue Test Utils, JUnits y Mockito, una herramienta que facilita mucho el testing. Estas herramientas nos han permitido simular diferentes estados y comportamientos de los componentes de Vue.js, asegurando que cada uno funcione de manera independiente y según lo previsto. Este es un ejemplo sencillo de un test del controlador que contiene el método presentado anteriormente.

```
@Test
public void testBuscarTrabajadores() {
    Trabajador trabajador1 = new Trabajador();
    trabajador1.setId(1L);
    trabajador1.setUsuario(new Usuario("Juan Romero"));
    trabajador1.setIdiomas("Español");
    trabajador1.setExperienciaLaboral(7L);

    Trabajador trabajador2 = new Trabajador();
    trabajador2.setId(2L);
    trabajador2.setUsuario(new Usuario("Guillermo Gimeno"));
    trabajador2.setIdiomas("Francés");
    trabajador2.setExperienciaLaboral(7L);

    List<Trabajador> trabajadores = Arrays.asList(trabajador1, trabajador2);

    when(trabajadorService.buscarTrabajadoresPorIdiomaYExperiencia(eq("Español"), eq(5L)))
            .thenReturn(trabajadores.subList(0,1));

    List<Trabajador> result = trabajadorController.buscarTrabajadores("Español", 5L);

    boolean foundJuanPerez = false;

    for (int i = 0; i < result.size(); i++) {
        String nombre = result.get(i).getUsuario().getNombre();
        if (nombre.equals("Juan Pérez")) {
            foundJuanPerez = true;
        }
        assertNotEquals("Guillermo Gimeno", nombre);
    }

    assertTrue(foundJuanPerez);
}
```

Figura 51. Ejemplo test unitario. (Creación Propia)

8.2. Testing de integración

Además de las pruebas unitarias, se han llevado a cabo testing de integración para poder asegurar que los diferentes módulos y componentes de la aplicación trabajen juntos sin problemas. Este tipo de pruebas se ha centrado en la interacción entre el front-end y el back-end, así como en la integración de diferentes servicios y APIs. Para este propósito, se han realizado pruebas manuales de uso.

Pero para un futuro se pretende utilizar Selenium, una herramienta que nos permite ejecutar pruebas end-to-end y testing de integración de manera automatizada y eficiente, simulando escenarios de usuario real y verificando el correcto funcionamiento de la aplicación en su conjunto.

8.3. Testing End to End

El testing end-to-end es crucial para poder validar que la aplicación funciona correctamente desde el punto de vista del usuario final. Se ha utilizado Cypress para llevar a cabo las pruebas E2E, que nos permite simular flujos completos de usuario, desde el inicio de sesión hasta la realización de pedidos y la navegación a través de las distintas vistas de la aplicación. Estas pruebas han sido muy útiles para poder identificar problemas que podrían no haber sido detectados en las pruebas unitarias o de integración, asegurando que la experiencia del usuario sea fluida y sin errores.

8.4. Testing de rendimiento

Para garantizar que la aplicación responda bien bajo diversas condiciones de carga, se realizarán pruebas de rendimiento. Utilizando herramientas como Lighthouse y JMeter, se evaluará la velocidad de carga, la capacidad de respuesta y el comportamiento de la aplicación bajo diferentes niveles de tráfico. Estas pruebas nos permitirán identificar y optimizar cuellos de botella en el rendimiento, asegurando que la aplicación sea capaz de manejar un gran número de usuarios simultáneamente sin degradar la experiencia. Pero por el momento no se han realizado aún estos tests.

8.5. Testing de seguridad

La seguridad es un aspecto muy importante de la aplicación, por lo que se realizará un testing de seguridad para identificar posibles vulnerabilidades ya que la experiencia en este aspecto ha sido aprendida en este proyecto. En el futuro, se utilizarán herramientas como OWASP ZAP y se verificará que la aplicación está protegida contra amenazas comunes como inyecciones de SQL y otros ataques. Esto nos permitirá asegurar que los datos de los usuarios y la integridad del sistema estén adecuadamente protegidos.

8.6. Testing de usabilidad

Finalmente, se realizarán pruebas de usabilidad para asegurarnos que la aplicación sea intuitiva y fácil de usar. Estas pruebas se llevarán a cabo con la participación de usuarios reales que utilizaran la aplicación en diferentes dispositivos y entornos, tanto como empresa como trabajadores. Con su feedback, se harán ajustes en la interfaz de usuario y en la experiencia general que sean necesarios, para poder asegurar que la aplicación sea accesible y satisfactoria para todos los usuarios.

9. Conclusiones

El desarrollo de esta aplicación de catering ha sido una experiencia muy enriquecedora y desafiante, que me ha permitido aplicar lo que aprendí en la universidad y al mismo tiempo enfrentar nuevos retos en áreas que no había tratado durante estos años, parte de este conocimiento lo pude aprender en las prácticas de empresa que realicé, pero por lo general he tenido necesidad de aprender sobre muchas de las herramientas utilizadas.

Desde el principio, al imaginar cómo sería la aplicación, supe que podría tener un impacto positivo en el sector del catering, ya que mayoritariamente se trata de gente joven como estudiantes, que muchas veces no tienen otra forma de ingresos, lo cual les obliga a aceptar condiciones que no son beneficiosas para ellos. La idea de conectar a empresas con trabajadores en estos casos donde normalmente no funciona así, me ha motivado a lo largo de todo el proceso de desarrollo por ser algo que he vivido.

El diseño y la implementación del sistema me han llevado a explorar diversas tecnologías y formas de trabajo. Usar frameworks como *Spring Boot* para el backend y Vue.js para el frontend me ha permitido crear una aplicación que considero sólida, segura y que puede crecer. En un principio me parecían tecnologías muy interesantes que tenía ganas de aprender a usar. Estas tecnologías cuentan con no solo muchas ventajas frente a otros frameworks, sino que cuentan con muchas librerías y herramientas adjuntas, por lo que ha supuesto un reto buscar opciones para poder trabajar con estos frameworks para poder facilitar el trabajo. Esto no solo ha mejorado mi conocimiento técnico, sino que también me ha dado una visión más amplia sobre cómo empezar a tratar con proyectos complejos.

Este trabajo realmente ha supuesto un reto para mí, ya que nunca había realizado un proyecto tan grande yo solo, y trabajar con un proyecto tan grande, pese a que se iba realizando poco a poco, ha sido en algunos casos frustrante, ya que todo debía estar interrelacionado y esto ha sido en muchos casos un tanto complicado, pero por otra parte me ha obligado a esforzarme más en plantear el diseño de la aplicación en todos sus aspectos.

Por lo general, todas las etapas del proceso han llevado sus dificultades, desde el principio en el que se planteaba la idea general de la aplicación, pasando por el desarrollo e incluyendo la redacción del proyecto, ya que escribir nunca ha sido de mis puntos fuertes.

Además de la parte técnica, este proyecto me ha ayudado a mejorar en un área muy importante, como es la gestión del tiempo, ya que al tener que compaginarlo con un puesto de trabajo donde no se usan estas tecnologías, no he podido dedicarle todo el tiempo que me hubiera gustado. También hay que destacar la constancia que es necesaria para poder desarrollar cualquier aplicación, así que por todo esto opino que tras este trabajo he ganado confianza para enfrentar futuros desafíos profesionales.

En resumen, crear esta aplicación de catering ha sido una experiencia muy gratificante. Me ha permitido aplicar y expandir mis conocimientos de diferentes asignaturas de la carrera que hasta ahora no había podido aplicar en tanta medida, como ha sido el diseño de bases de datos y, sobre todo, crear una herramienta que podría mejorar significativamente el sector del catering, aunque sea en un futuro. Este proyecto me ha hecho tener ganas de aprender otra vez y posiblemente seguir formándome en el futuro.

9.1. Trabajo futuro

Este proyecto es un proyecto muy grande, que aún requiere mucho desarrollo para poder estar acabado y listo para el público. En un principio no pensaba que supondría tantas horas de desarrollo, pero al tratarse de una aplicación con intención de llegar a muchísimos usuarios hay que pensar en numerosos casos. Por lo que la intención para el futuro es:

- Optimizar la experiencia de usuario, para sería necesario aprender sobre interfaces de usuario y experiencia de usuario (UI/UX) un campo en el que no tengo demasiada experiencia.
- Integración de sistemas de pago dentro de la aplicación, de esta forma nuestra plataforma podría encargarse de la gestión desde el principio del proceso hasta el final.
- Implementar OpenID en el módulo de login, de esta forma se podrá acceder con aplicaciones de terceros, facilitando tanto el login como el registro.
- Ampliación y revisión de la seguridad, el apartado de seguridad, pese al tiempo invertido en ella, no tiene aún la capacidad de estar expuesta al gran público, es necesario implementar muchos tests y revisar arduo y tendido este apartado.
- Desarrollo de una aplicación móvil nativa, pese a que la aplicación web sea compatible con dispositivos móviles, una aplicación de iOS y Android podría ayudar a conseguir usuarios que no disponen de un ordenador o que se sienten más cómodos usando una aplicación.
- Implementación de inteligencia artificial que ayude con las recomendaciones a la hora de seleccionar trabajadores.
- Soporte de idiomas adicionales, esto podría ayudar a empresas que pese a encontrarse en España necesitan gente para servicios en únicamente otro idioma o incluso una expansión internacional.
- Implementar una guía con documentación para los nuevos usuarios.
- Mejoras la gestión de reseñas tanto de empresas como de trabajadores, de esta forma se podrá evitar reseñas injustas.
- Mejoras en el testing e implementación de nuevos modelos de prueba de funcionamiento.

9.2. Relación con estudios cursados

Diseño y gestión de Bases de Datos, Redes de Computadores, Concurrencia y sistemas distribuidos e Ingeniería del software

A lo largo de la carrera se han ido obteniendo diferentes habilidades y conocimientos, que se han podido aprovechar para poder realizar este trabajo de la mejor manera, y han asentado la base para poder aprender el resto de habilidades necesarias, tanto para este trabajo como para la vida laboral.

Obviamente, muchísimas de las asignaturas estudiadas durante estos años tienen implicación en este trabajo, como sobre todo todas las asignaturas en las que se ha podido aprender y practicar cómo trabajar con Java, pero se ha preferido centrarse en solo 4.

Para el diseño de la base de datos, cabe destacar la asignatura de Diseño y gestión de Bases de Datos, donde se han adquirido los conocimientos necesarios para implementar una Base de Datos funcional.

Para la intercomunicación de las diversas partes de la aplicación y estructura del software se destacan Redes de Computadores y Concurrencia y Sistemas Distribuidos.

Y finalmente con la relación de los diferentes casos de uso y la parte más de diseño teórico de la aplicación, se destaca la asignatura de Ingeniería del Software.

Bibliografía

[1] Garrido, S. (2023, November 14). *¿Qué son las metodologías ágiles?* IEBS.

Revisado el: 15/05/2024

https://www.iebschool.com/blog/que-son-metodologias-agiles-agile-scrum/

[2] Roberto Touza. (2023, 11 02). *Metodologia Scrum*. Roberto Touza David.

Revisado el: 15/05/2024

https://robertotouza.com/libros/metodologia-scrum/

[3] El país. (2022, June 3). *Listeros: quién se esconde detrás del mercado negro de la hostelería*. EL PAÍS.

Revisado el: 17/05/2024

https://elpais.com/podcasts/hoy-en-el-pais/2022-06-03/listeros-quien-se-esconde-detras-del-mercado-negro-de-la-hosteleria.html#

[4] SAGE. (2024, January 31). *Mejores webs para buscar empleados: ¡Descúbrelas! I Sage Advice*. Sage.

Revisado el: 17/05/2024

https://www.sage.com/es-es/blog/21-webs-para-buscar-empleados/

[5] Spring.io. (2024, 01 16). *Spring Framework*. Spring.

Revisado el: 17/05/2024

https://spring.io/projects/spring-framework

[6] Apache. (2024). *Maven – Maven Getting Started Guide*. Apache Maven.

Revisado el: 19/05/2024

https://maven.apache.org/guides/getting-started/index.html

[7] Spring.io. (2024, 03 05). *Spring Boot*. Spring.

Revisado por última vez el: 26/05/2024

https://docs.spring.io/spring-boot/index.html

[8] Jain, S. (2024, May 23). *Introduction to Spring Boot*. GeeksforGeeks.

Revisado por última vez el: 26/05/2024

https://www.geeksforgeeks.org/introduction-to-spring-boot/

[9] JetBrains. (2024, July 4). *IntelliJ IDEA overview | IntelliJ IDEA Documentation*.

Revisado por última vez el: 26/05/2024

JetBrains. https://www.jetbrains.com/help/idea/discover-intellij-idea.html

[10] git-scm. (2024, 01 01). *Documentation*. Git.

Revisado por última vez el: 29/05/2024

https://git-scm.com/doc

[11] Atlassian. (2024, 07 05). *Get started with Sourcetree | Sourcetree*. Atlassian Documentation.

Revisado por última vez el: 29/05/2024

https://confluence.atlassian.com/get-started-with-sourcetree

[12] IBM. (2024, 04 16). *Java Persistence API (JPA)*. IBM.

Revisado por última vez el: 29/05/2024

https://www.ibm.com/docs/es/was-liberty/nd?topic=liberty-java-persistence-api-jpa

[13] Hibernate. (2024, 08 06). *Your relational data. Objectively.* Hibernate.

Revisado por última vez el: 03/06/2024

https://hibernate.org/orm/

[14] Spring.io. (2024, 03 07). *Spring Data JPA*. Spring.

Revisado por última vez el: 03/06/2024

https://spring.io/projects/spring-data-jpa

[15] projectlombok.org. (2024, 06 04). *Documentación Lombok*. Project Lombok.

Revisado por última vez el: 03/06/2024

https://projectlombok.org/

[16] Spring.io (2024). Spring Security. Spring.

Revisado por última vez el: 23/08/2024

https://spring.io/projects/spring-security

[17] Oracle.com (2024) Databases Oracle. Oracle

Revisado por última vez el: 25/08/2024

https://www.oracle.com/es/database/

[18] vuejs.org. (2024, 01 01). *Documentation Vue.js*. Vue.js - The Progressive JavaScript Framework | Vue.js.

Revisado por última vez el: 29/08/2024

https://vuejs.org/

[19] router.vuejs.org (2024, 08 16) Documentation Vue Router

Revisado por última vez el: 29/08/2024

https://router.vuejs.org/

[20] vuetify. (2024, 01 10). *Vuetify Documentation*. Vuetify — A Vue Component Framework.

Revisado por última vez el: 15/07/2024

https://vuetifyjs.com/en/

[21] Axios. (2024, 01 10). *Empezando*. Axios.

Revisado por última vez el: 29/08/2024

https://axios-http.com/es/docs/intro

[22] Atlassian. (2024). Arquitectura modular monolítica vs Microservicios.

Revisado por última vez el: 20/06/2024

https://www.atlassian.com/es/microservices/microservices-architecture/microservices-vs-monolith

[23] AWS. (2024, 08 09). La diferencia entre arquitectura monolítica y microservicios.

Revisado por última vez el: 20/06/2024

https://aws.amazon.com/es/compare/the-difference-between-monolithic-and-microservices-architecture/

[24] Medium.com (2024). Autenticación y autorización en medidas de seguridad con Spring.

Revisado por última vez el: 23/08/2024

https://medium.com/somos-pragma/autenticaci%C3%B3n-y-autorizaci%C3%B3n-medidas-de-seguridad-con-spring-security-3f3de4e0b1b2

[25] Álvarez, C. (2024, 06 07). *¿ Qué son los Servicios REST ?* Arquitectura Java.

Revisado por última vez el: 05/07/2024

https://www.arquitecturajava.com/servicios-rest/

[26] Jesús Henríquez. (2022, June 5). *Arquitectura monolítica vs Arquitectura de microservicios*. Jesús Henríquez.

Revisado por última vez el: 05/07/2024

https://jesushenriquez.com/p/arquitectura-monolitica-vs-arquitectura

[27] Medium.com (2024) 10 Best Java Frameworks.

Revisado por última vez el: 05/08/2024

https://medium.com/@rs4528090/10-best-java-frameworks-you-should-know-in-2024-de6561eae72b

[28] Apache Struts. Apache Struts official site.

Revisado por última vez el: 05/08/2024

https://struts.apache.org/

[29] Grails. Grails Documentation.

Revisado por última vez el: 07/08/2024

https://grails.org/

[30] React. (2024, 01 01). *React Reference Overview – React*. React.

Revisado por última vez el: 10/08/2024

https://es.react.dev/reference/react

[31] Angular. (2024, 01 01). *What is Angular? • Angular*. Angular.

Revisado por última vez el: 10/08/2024

https://angular.dev/overview

[32] mdn. (2024, July 28). *Introducción a Svelte | MDN*. MDN Web Docs.

Revisado por última vez el: 11/08/2024

https://developer.mozilla.org/es/docs/Learn/Tools_and_testing/Client-side_JavaScript_frameworks/Svelte_getting_started

[33] Amazon. (2024, 05 10). *Monolítico frente a microservicios: diferencia entre arquitecturas de desarrollo de software - AWS*. AWS.

Revisado por última vez el: 12/08/2024

https://aws.amazon.com/es/compare/the-difference-between-monolithic-and-microservices-architecture/

[34] Spring.io. Building a REST server with spring.

Revisado por última vez el: 25/08/2024

https://spring.io/guides/tutorials/rest

[35] Tiogo, E. C. (2024, March 19). *Implement JWT authentication in a Spring Boot 3 application*. Medium.

Revisado por última vez el: 29/08/2024/

https://medium.com/@tericcabrel/implement-jwt-authentication-in-a-spring-boot-3-application-5839e4fd8fac

[36] yarnpkg. (2024, 01 01). *Getting Started*. Yarn.

Revisado por última vez el: 10/08/2024

https://classic.yarnpkg.com/en/docs/getting-started

[37] w3techs. (2024, 03 05). *Vue.js vs. Angular usage statistics, August 2024*. W3Techs.

Revisado por última vez el: 26/08/2024

https://w3techs.com/technologies/comparison/js-angularjs,js-vuejs

Anexo ODS

ANEXO

OBJETIVOS DE DESARROLLO SOSTENIBLE

Grado de relación del trabajo con los Objetivos de Desarrollo Sostenible (ODS).

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

Reflexión sobre la relación del TFG/TFM con los ODS y con el/los ODS más relacionados.

El objetivo que más se trabaja en este proyecto es el ODS 8 "Trabajo decente y crecimiento económico", este objetivo pretende promover el crecimiento económico inclusivo y sostenible, además de un trabajo decente para todos. Ya que el proyecto se centra en fomentar una plataforma que permite encontrar trabajo de calidad en un sector en el que habitualmente no suele serlo. Ayudando así tanto a gente de las nuevas generaciones que necesiten encontrar un puesto de empleo que les permita compaginar el trabajo con su vida, tanto a jóvenes que aún están estudiando como a personas que puedan ser más mayores con otros factores de vida. La aplicación contará con diferentes apartados de ayuda para que pueda ser usada por cualquiera, y el desconocimiento tecnológico no suponga una barrera de entrada. En todo momento todas las partes se podrán poner en contacto con nosotros, una tercera entidad, que pueda mediar por ellos, tanto por empresas como por trabajadores, pero con especial énfasis la preocupación será que los trabajadores puedan tener un trabajo que cumpla con todos los requerimientos que exige la ley, y sean recompensados por sus diferentes aptitudes.

El proyecto también está muy relacionado con el ODS 9, "Industria, innovación e infraestructuras". Este objetivo promueve la construcción de infraestructuras resistentes, la industrialización inclusiva y sostenible, y el fomento de la innovación. La aplicación web desarrollada no solo emplea tecnología actual, sino que además busca ayudar a la digitalización del sector del catering, un ámbito que tradicionalmente ha dependido de métodos más manuales y menos técnicos. Creando soluciones tecnológicas avanzadas, se ayuda a modernizar las infraestructuras de las empresas dedicadas a este servicio.
A largo plazo, la aplicación tiene el potencial de generar una industria más eficiente y sostenible. Estas mejoras no solo aumentan la rentabilidad de las empresas, sino que también contribuyen a un servicio más de calidad, alineándose con los principios de sostenibilidad promovidos por el ODS 9 y el trabajo digno del ODS 8. Además, al estar diseñada para ser escalable y adaptable, la aplicación fomenta la innovación continua dentro de la industria, asegurando que las empresas puedan incorporar nuevas tecnologías y mejorar sus operaciones a medida que evolucionan las necesidades del mercado.
El proyecto también se orienta a fortalecer las infraestructuras digitales del sector, proporcionando una base tecnológica robusta que no solo permitirá su crecimiento, sino que también garantizará que las empresas que utilicen la plataforma mantengan altos estándares de calidad y eficiencia en un entorno competitivo con respecto a la contratación de personal. Esta capacidad de innovación constante, junto con una infraestructura tecnológica moderna, refuerza el papel del proyecto en la creación de una industria más avanzada, inclusiva y preparada para los desafíos futuros.