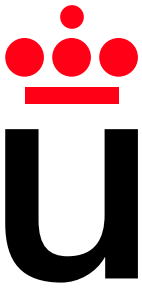


Universidad
Rey Juan Carlos

Escuela Técnica Superior
de Ingeniería Informática

Grado en Ingeniería del Software

Curso 2023-2024

Trabajo Fin de Grado

ENTERPRISE EVENT SOLUTIONS

Autor: Carlos Fernández López
Tutor: Michel Maes Bermejo

Resumen

El presente Trabajo de Fin de Grado (TFG) tiene como objetivo principal el desarrollo de una aplicación web llamada Enterprise Event Solutions, diseñada para la gestión eficiente de eventos corporativos. Esta herramienta aborda las necesidades específicas de las empresas en la planificación, organización y seguimiento de eventos, ofreciendo una solución integral y personalizada.

La memoria de este TFG abarca diversos aspectos esenciales para el desarrollo de la aplicación. Se detallan los objetivos y resultados obtenidos, destacando cómo la plataforma facilita la planificación de eventos, reduce la carga administrativa y permite un seguimiento detallado mediante informes y estadísticas.

Además, se describen las metodologías aplicadas durante el desarrollo del proyecto, incluyendo técnicas de gestión de proyectos ágiles que han permitido una adaptación flexible y eficiente a las necesidades cambiantes del desarrollo.

Se exploran las tecnologías utilizadas, como frameworks de desarrollo web, bases de datos y herramientas de automatización, que han sido seleccionadas por su capacidad para ofrecer una solución robusta y escalable.

También se presentan los requisitos funcionales y no funcionales de la aplicación, abarcando desde la gestión de usuarios y eventos hasta la interfaz amigable y fácil de usar, que permite a los usuarios navegar y utilizar la plataforma sin necesidad de formación extensa.

Por supuesto se incluye una sección para los tests para cubrir el código de la aplicación y verificar el correcto funcionamiento de esta.

En conclusión, Enterprise Event Solutions se presenta como una solución robusta y eficiente para la gestión de eventos corporativos, cumpliendo con los objetivos propuestos y demostrando ser una herramienta valiosa para las empresas en la optimización de sus procesos de organización de eventos.

Palabras clave:

- EVS (Enterprise Event Solutions)
- AWS
- SpringBoot
- Vue
- Seguridad
- Web
- Docker

Índice de contenidos

Índice de tablas

Índice de figuras

Índice de códigos

1 Introducción

1.1. Contexto y alcance

La organización de eventos corporativos es fundamental para fomentar la cultura organizacional, establecer relaciones estratégicas e impulsar el crecimiento de una empresa en el mundo moderno. Sin embargo, la planificación y ejecución de estos eventos presentan numerosos desafíos que pueden comprometer su éxito. Los desafíos comunes incluyen la dificultad de coordinar a varios departamentos, la necesidad de una comunicación fluida y la gestión eficiente de los recursos. Debido a esto, surgió la idea de desarrollar Enterprise Event Solutions (EVS), una aplicación web destinada a satisfacer estas demandas y mejorar significativamente el manejo de eventos corporativos.

La evaluación de una serie de problemas comunes en la gestión de eventos empresariales llevó a la decisión de establecer EVS:

En primer lugar, no hay una plataforma centralizada que permita a las empresas organizar de manera integral todos los aspectos de un evento. Muchas empresas dependen de una variedad de programas que no están conectados, como hojas de cálculo, correos electrónicos y software de gestión de proyectos, lo que hace que las cosas no

funcionen bien y aumenta el riesgo de errores.

La comunicación interna es otro aspecto importante que a menudo se olvida cuando se trata de organizar eventos. La coordinación de equipos y departamentos requiere una herramienta que facilite la comunicación en tiempo real y asegure que todos los involucrados estén informados sobre las actualizaciones y cambios. El objetivo de EVS es facilitar la comunicación, reducir las confusiones y fomentar la colaboración.

La responsabilidad administrativa asociada con la gestión de eventos. La gestión de inscripciones y el seguimiento de la asistencia requieren mucho tiempo y recursos. EVS permite a los organizadores concentrarse en aspectos más estratégicos y creativos del evento, mejorando su calidad y efectividad mediante la automatización de estos procesos.

Contar con herramientas de análisis y seguimiento también es esencial para evaluar el éxito de los eventos y tomar decisiones informadas para futuras planificaciones. Las funcionalidades avanzadas de EVS permiten la creación de informes detallados que brindan a las empresas información útil sobre la participación, el desempeño y las áreas de mejora.

Finalmente, un factor clave en la creación de EVS fue la creciente demanda de soluciones tecnológicas que se adapten a las necesidades específicas de cada empresa. La personalización y la flexibilidad de la plataforma la hacen una herramienta adaptable a una variedad de tipos de eventos y tamaños de negocios, lo que permite que cada usuario maximice su utilidad.

En resumen, la creación de soluciones de eventos empresariales satisface la necesidad de una herramienta completa y efectiva para la gestión de eventos corporativos. El objetivo de EVS es transformar la forma en que las empresas organizan y ejecutan sus eventos, agregando valor y contribuyendo al éxito empresarial al centralizar la planificación, mejorar la comunicación, automatizar procesos administrativos y proporcionar herramientas de análisis.

1.2. Descripción del problema

Según datos oficiales, en España hay 2.922.920 empresas de las cuales solo 5.531 son grandes empresas [1] ¿Qué pasa con las 2.917.389 restantes?.

Problemas de Comunicación y Alcance:

1. **Recursos limitados**: A diferencia de las grandes corporaciones, las PYMEs generalmente no disponen de los mismos recursos financieros y humanos para invertir en estrategias de marketing y comunicación. Esto limita su capacidad para desarrollar campañas efectivas y sostenibles que lleguen a una amplia audiencia.

2. **Falta de canales de comunicación adecuados**: Las grandes empresas suelen tener acceso a una variedad de canales de comunicación, incluidos medios de comunicación masiva, redes sociales gestionadas por equipos especializados y eventos de gran escala. Las PYMEs, por otro lado, a menudo carecen de estos canales, lo que dificulta su capacidad para llegar a nuevos clientes y mantener una comunicación constante con sus públicos objetivo.

3. **Menor visibilidad**: Las grandes empresas tienen la ventaja de una mayor visibilidad de marca, lo que les permite mantenerse en la mente de los consumidores más fácilmente. Las PYMEs luchan por obtener y mantener esta visibilidad, lo que puede resultar en una menor lealtad del cliente y dificultades para captar nuevos mercados.

4. **Tecnología y digitalización**: Muchas PYMEs no tienen acceso a las últimas tecnologías y herramientas digitales que pueden facilitar la comunicación y el marketing. La falta de digitalización no solo afecta su eficiencia operativa sino también su capacidad para implementar estrategias de marketing digital que son cruciales en el mercado actual.

5. **Reducción de costes**: La gestión de eventos corporativos es una herramienta clave para la promoción y la creación de redes, pero muchas PYMEs no pueden permitirse los costes asociados con la organización de eventos a gran escala.

Esto limita su capacidad para interactuar cara a cara con clientes potenciales y fortalecer las relaciones comerciales existentes.

6. **Competencia intensa**: En un mercado saturado, las PYMEs compiten no solo con otras pequeñas empresas sino también con grandes corporaciones que tienen una presencia más consolidada. La competencia intensa puede hacer que sea aún más difícil para las PYMEs destacar y atraer la atención del público.

Impacto en el crecimiento y sostenibilidad:

Estos problemas de comunicación y alcance no solo limitan la capacidad de las PYMEs para crecer, sino que también ponen en riesgo su sostenibilidad a largo plazo. Sin los medios adecuados para llegar a su público objetivo y sin una estrategia de comunicación efectiva, estas empresas enfrentan dificultades para expandirse, innovar y mantenerse competitivas en el mercado. La falta de visibilidad y la incapacidad para interactuar eficazmente con los clientes pueden conducir a una disminución de las ventas y, en última instancia, afectar la viabilidad económica de la empresa.

Necesidad de soluciones eficientes:

Para abordar estos desafíos, surge la necesidad de soluciones eficientes y accesibles que puedan ayudar a las PYMEs a mejorar su comunicación y alcance. Herramientas que centralicen la gestión de eventos, faciliten la automatización de procesos administrativos, mejoren la visibilidad y proporcionen canales de comunicación efectivos son esenciales para permitir que estas empresas compitan en igualdad de condiciones con las grandes corporaciones. **Enterprise Event Solutions** se presenta como una respuesta a estas necesidades, ofreciendo una plataforma integral diseñada específicamente para ayudar a las PYMEs a superar sus limitaciones y alcanzar sus objetivos de crecimiento y sostenibilidad.

1.3. Estudio de alternativas

Contar con las herramientas adecuadas en el campo de la organización de eventos es esencial para el éxito. Debido a su amplia gama de características, las aplicaciones de gestión de eventos se han convertido en una herramienta esencial para simplificar tareas, optimizar procesos y permitir la realización de eventos memorables.

A continuación se muestra una serie de Aplicaciones Web con funcionalidades parecidas a Enterprise Event Solutions:

- **Eventbrite** es una plataforma versátil para la creación y gestión de eventos. Permite a los usuarios vender entradas online con opciones de precios y tarifas personalizables, así como gestionar el registro y la participación de los asistentes. Además, ofrece herramientas de marketing integradas para promocionar los eventos y análisis de datos detallados sobre la asistencia y el rendimiento de los mismos.

- **Wix Events** es una herramienta de creación de páginas de eventos personalizadas dentro del sitio web de Wix. Ofrece funcionalidades para gestionar el registro de los asistentes y proporciona herramientas de marketing para promocionar los eventos. Además, permite a los usuarios realizar un análisis detallado de la asistencia y el rendimiento del evento a través de datos recopilados en la plataforma.

- **Zoom Events** es una plataforma diseñada para la creación y gestión de eventos virtuales e híbridos, como webinars, reuniones y conferencias. Permite interactuar con los participantes a través de funciones como encuestas, preguntas y respuestas, y chat en vivo. Además, ofrece integraciones con otras herramientas de Zoom para una experiencia más completa y proporciona análisis de datos sobre la asistencia y el rendimiento del evento.

El valor añadido distintivo de **Enterprise Event Solutions**, reside en el enfoque centrado en el usuario y la excepcional facilidad de uso. Se ha diseñado la plataforma desde sus cimientos con la premisa de que la accesibilidad y la usabilidad son

fundamentales para garantizar una experiencia óptima para todos los usuarios, independientemente de su nivel de familiaridad con la tecnología.

Lo que distingue a **Enterprise Event Solutions** es la capacidad para llegar a usuarios de todos los niveles de habilidad tecnológica. No todos los usuarios están familiarizados con las últimas tecnologías, y es por eso que he priorizado la accesibilidad en el diseño de la aplicación. Incluso aquellos que pueden sentirse desactualizados en términos de tecnología encontrarán que **EVS** es fácil de utilizar.

2

Objetivos

2.1. Objetivos generales

Para la aplicación de EVS se han marcado los siguientes objetivos principales:

- **Centralizar la planificación de eventos.** La plataforma permitirá a las empresas u organizadores crear y gestionar eventos desde una sola pantalla y a los clientes apuntarse a estos y disponer de las entradas a un solo click.

- **Gestión de la frecuencia de usuarios y eventos en la aplicación.** La aplicación permitirá gestionar a todos los usuarios en el Sistema mediante un usuario Administrador. Esto facilitará la relación Organización-Cliente manteniendo un equilibrio entre la cantidad de unos y de otros.

- **Agrupar las Entradas de los usuarios.** EVS ofrecerá a los usuarios la posibilidad de Agrupar las entradas en un mismo sitio, así como imprimir las entradas.
- **Aplicación de uso general.** Ofrecer una herramienta para, desde el punto de vista de una organización, mejorar la asistencia a sus eventos y, desde el punto de vista del cliente, tener un acceso sencillo e intuitivo a estos.
- **Desarrollo Fullstack.** Aplicar y ampliar los conocimientos adquiridos sobre SpringBoot y Vue.
- **Entrega Continua.** Se automatizarán los procesos de Integración y Entrega Continua para facilitar el desarrollo.

3

Tecnologías, Herramientas y Metodologías

3.1. Tecnologías y herramientas empleadas

En esta sección se detallan las diversas tecnologías y herramientas utilizadas en el desarrollo, pruebas y despliegue de la aplicación.

3.1.1. Backend

Spring Boot

Spring Boot[2] es un framework Java que simplifica el desarrollo de aplicaciones web al eliminar la complejidad de la configuración manual. Ofrece configuración automática, empaquetado de aplicaciones independientes y un inicio rápido integrado, lo

que permite a los desarrolladores crear aplicaciones de manera más rápida y eficiente, centrándose en la lógica de negocio en lugar de la configuración. Genial para el desarrollo de aplicaciones *CRUD*[1].

MySQL y H2

MySQL es un sistema de gestión de bases de datos relacional de código abierto ampliamente utilizado en el desarrollo de aplicaciones web y empresariales. Destaca por su escalabilidad, rendimiento, fiabilidad y facilidad de uso. Ofrece opciones sólidas de seguridad y es compatible con una variedad de plataformas y lenguajes de programación. MySQL es una herramienta poderosa para gestionar eficientemente grandes volúmenes de datos y garantizar la integridad y disponibilidad de la información.

H2 es una base de datos relacional escrita en Java que puede funcionar en modo embebido o en modo servidor. Es una opción popular para desarrollo y pruebas debido a su rápido rendimiento y su capacidad para integrarse fácilmente con aplicaciones Java. A menudo se utiliza en proyectos Spring Boot para pruebas y desarrollo local debido a su facilidad de uso y configuración mínima.

En el caso del desarrollo de **Enterprise Event Solutions** se ha utilizado H2 durante la fase de desarrollo permitiendo realizar pruebas sobre la aplicación a medida que se desarrollaba.Y la base de datos MySQL se ha dockerizado desde un principio, incluso durante el despliegue de la aplicación en un entorno local, para ahorrar recursos.

AWS S3

AWS S3 es un servicio de almacenamiento en la nube ofrecido por Amazon Web Services. Destaca por su escalabilidad, durabilidad, seguridad y facilidad de uso. Permite almacenar grandes cantidades de datos de forma segura y acceder a ellos de manera eficiente a través de una interfaz intuitiva y una API robusta. Con una estructura de precios flexible, AWS S3 es una opción atractiva para empresas que buscan una solución

[1] *CRUD* es un acrónimo en inglés que se refiere a las operaciones básicas de manipulación de datos en aplicaciones informáticas: Create (Crear), Read (Leer), Update (Actualizar) y Delete (Eliminar).

de almacenamiento en la nube rentable y confiable.

Se ha utilizado AWS S3 para almacenar la totalidad de las imágenes de la aplicación permitiendo, de esta forma, que la base de datos principal tenga menos carga de datos mejorando significativamente la velocidad de respuesta de esta. El bucket de S3 se integra en el backend donde, después de generar una archivo apartir de una imagen, ofrece una URL que se almacena en la base de datos.

IntelliJ

IntelliJ IDEA es un entorno de desarrollo integrado (IDE) altamente productivo desarrollado por JetBrains. Diseñado para diversas tecnologías, ofrece una interfaz de usuario intuitiva y funciones avanzadas que mejoran la productividad del desarrollador. Con soporte para múltiples lenguajes y marcos de trabajo, integración con herramientas de desarrollo y características colaborativas, IntelliJ IDEA es una opción popular para el desarrollo de software en Java y otros lenguajes.

Postman

Para realizar pruebas durante el desarrollo de la API se ha hecho uso de Postman. Postman es una herramienta de colaboración para el desarrollo y pruebas de APIs.Permite a los desarrolladores enviar solicitudes HTTP a sus API y recibir respuestas, facilitando la validación y verificación de los endpoints de la API. Con Postman,se pueden crear colecciones de pruebas automatizadas, manejar diferentes entornos de prueba y compartir configuraciones con el equipo de desarrollo, lo que mejora la eficiencia y la calidad del desarrollo de la API.

Swagger

Para generar la documentación de la API se ha utilizado Swagger[3] que es un conjunto de herramientas de código abierto para diseñar, construir y documentar APIs RESTful. Facilita la creación de documentación interactiva y legible de las APIs,describiendo

las operaciones, parámetros y respuestas de manera clara. Además, permite generar automáticamente código cliente y servidor a partir de las especificaciones de la API.Swagger es parte de la OpenAPI Specification, un estándar ampliamente utilizado para describir interfaces de APIs RESTful.

3.1.2. Frontend

Vue.js

Vue.js[4] es un popular marco de trabajo de código abierto para construir interfaces de usuario interactivas en aplicaciones web de una sola página. Se destaca por su enfoque centrado en el componente, su sintaxis declarativa y su eficiente sistema de reactividad. Vue.js facilita la construcción de aplicaciones web complejas al fomentar la reutilización de componentes y ofrecer herramientas integradas para la manipulación del DOM y la gestión del estado de la aplicación.

VSCode

Visual Studio Code (VSCode) es un IDE popular desarrollado por Microsoft, conocido por su versatilidad, rendimiento y comunidad activa. Ofrece características avanzadas y extensiones para diferentes lenguajes de programación. Es altamente personalizable y cuenta con una amplia documentación disponible en su sitio web oficial.

3.1.3. Construcción

Maven

Maven es una herramienta de gestión y comprensión de proyectos utilizada principalmente en proyectos Java. Se destaca por sus capacidades de automatización en la compilación, prueba, empaquetado y despliegue de aplicaciones. Maven gestiona las dependencias del proyecto de manera eficiente, lo que permite a los desarrolladores

centrarse en la escritura de código en lugar de en la configuración y administración de bibliotecas. Además, proporciona un marco de trabajo coherente que facilita la construcción de proyectos complejos y su integración con otras herramientas de desarrollo y sistemas de control de versiones. Maven utiliza un archivo de configuración denominado *pom.xml*[2] , que define la estructura del proyecto y las dependencias necesarias.

3.1.4. Despliegue

Docker

Docker[5] es una plataforma de código abierto que permite crear, implementar y ejecutar aplicaciones en contenedores. Destaca por su portabilidad, eficiencia y facilidad de uso. Con la tecnología de contenedores, Docker ofrece aislamiento de aplicaciones y escalabilidad, lo que simplifica el desarrollo y la administración de aplicaciones en diferentes entornos. En resumen, Docker es una herramienta fundamental para la creación y gestión eficiente de aplicaciones modernas.

Bash

He utilizado un script de bash para poder automatizar dentro de mi EC2 el despliegue de la imagen de la aplicación.

Bash es un intérprete de línea de comandos y lenguaje de scripting ampliamente utilizado en sistemas operativos Unix y Linux. Es esencial para automatizar tareas, administrar sistemas y ejecutar scripts complejos que interactúan con el sistema operativo. Bash permite a los desarrolladores y administradores de sistemas automatizar procesos repetitivos, mejorar la eficiencia y gestionar entornos de desarrollo y producción de manera efectiva.

[2]Project Object Model

AWS EC2

AWS EC2 es un servicio de cómputo en la nube proporcionado por Amazon Web Services. Destaca por su escalabilidad, flexibilidad y seguridad. Permite a los usuarios alquilar capacidad informática según sus necesidades, con opciones de pago por uso. Es fácil de usar y ofrece una variedad de opciones de implementación. En resumen, EC2 es una solución eficiente y confiable para ejecutar aplicaciones y cargas de trabajo en la nube.

3.2. Resumen de las tecnologías

La Tabla 3.1 recoge un resumen de las tecnologías empleadas para el desarrollo de Enterprise Event Solutions:

Tecnología	Uso
SpringBoot	Framework
Maven	Herramienta de construcción
Postman	Herramienta de pruebas durante el desarrollo
Swagger	Software de documentación
AWS S3	Tecnología de de BD en la nube
Vue.js	Framework
MySQL	Tecnología de BD
IntelliJ	Entorno de Desarrollo
VSCode	Entorno de Desarrollo
Docker	Creación de imágenes comprimidas para despliegue
Bash	Lenguaje de programación
AWS EC2	Servidor web en la nube

Tabla 3.1: Tecnologías y sus usos correspondientes

3.3. Metodologías empleadas

En el desarrollo de Enterprise Event Solutions, se ha adoptado una metodología ágil basada en Scrum para asegurar que el proceso de creación fuera eficiente, organizado y adaptable a cambios y mejoras constantes. Aunque he trabajado solo en este

proyecto, se ha implementado de manera rigurosa los principios de Scrum, enfocándose en alcanzar pequeños objetivos semanales y manteniendo una estructura organizada.

Cada semana, se han establecido objetivos específicos y alcanzables, lo que ha permitido avanzar de manera constante y mantener un alto nivel de motivación. Esta división en pequeños objetivos semanales ha ayudado a gestionar mejor el tiempo y a priorizar tareas, asegurando que cada funcionalidad de la aplicación se desarrolle de manera coherente y sin omisiones.

Para organizar y seguir el progreso, se ha utilizado un tablero de Trello. Este tablero ha sido una herramienta invaluable para no olvidar ideas o tareas pendientes. Cada tarjeta en el tablero representaba una tarea o idea específica, y he seguido un flujo de trabajo que incluía las siguientes etapas: por hacer, en progreso, y completado. Esta visualización clara del trabajo pendiente y del progreso realizado me ha permitido mantenerme enfocado y productivo.

Además, al final de cada semana, he realizado una revisión de los objetivos alcanzados y he ajustado el plan para la semana siguiente. Esta práctica de retrospectiva semanal ha sido fundamental para identificar áreas de mejora, solucionar problemas y asegurar que el proyecto siga avanzando de acuerdo con los objetivos establecidos.

En resumen, la adopción de la metodología Scrum, con su enfoque en pequeños objetivos semanales y el uso de un tablero Trello para la gestión de tareas, ha sido clave para el desarrollo exitoso de Enterprise Event Solutions. A pesar de trabajar solo, esta estructura me ha permitido mantener un alto nivel de organización, adaptabilidad y eficiencia en todo el proceso de desarrollo.

En cuanto al desarrollo del código, se ha utilizado un flujo de trabajo basado en Pull Requests (PRs). Cada cambio o nueva funcionalidad se desarrollaba en una rama separada y se integraba al código principal solo después de ser revisada y aprobada mediante un Pull Request. Este enfoque ha permitido mantener un historial claro de los cambios, realizar revisiones detalladas y asegurar que cada nueva adición al código se integrara de manera ordenada y sin conflictos. Este método ha ayudado a mantener la disciplina y la organización en el desarrollo del software.

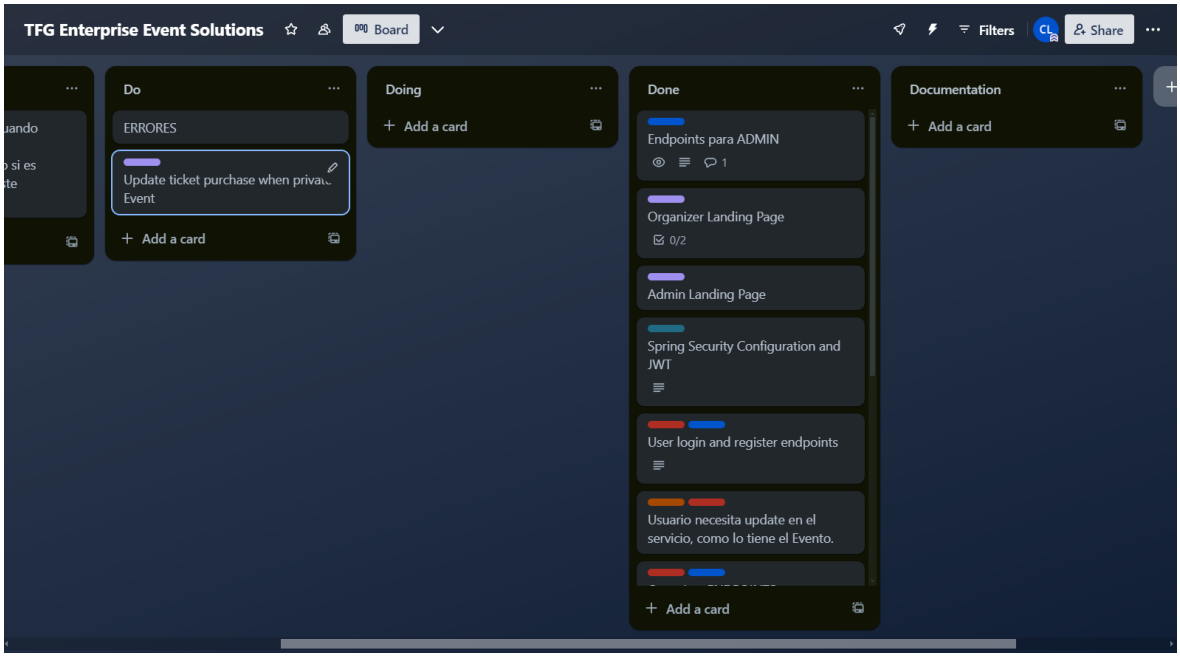


Figura 3.1: Trello en proceso.

El uso de Pull Requests ha ofrecido múltiples ventajas en el proceso de desarrollo. En primer lugar, ha facilitado la identificación y resolución de errores antes de que los cambios se integren en la rama principal. Cada PR actuaba como un punto de control donde podía revisar el código, realizar pruebas y verificar la funcionalidad, asegurando que solo los cambios bien testados y verificados fueran añadidos al proyecto.

Además, este método ha proporcionado una documentación implícita del desarrollo. Cada Pull Request incluía una descripción detallada de los cambios realizados, los problemas que solucionaba o las nuevas características añadidas. Esto no solo facilitó la gestión del proyecto, sino que también creó un registro histórico útil para futuras referencias y para cualquier otra persona que pueda colaborar en el futuro.

Trabajar con ramas separadas para cada nueva funcionalidad o cambio también ha sido crucial para mantener la estabilidad del proyecto. Al aislar el desarrollo de nuevas características en ramas dedicadas, he evitado que los cambios en progreso afecten la estabilidad de la rama principal. Esto ha sido especialmente útil para realizar experimentos o implementar grandes cambios sin riesgo de romper la aplicación.

Finalmente, la disciplina de utilizar Pull Requests, incluso trabajando solo, ha fo-

mentado un enfoque metódico y estructurado al desarrollo. Este flujo de trabajo me ha obligado a pensar críticamente sobre cada cambio, a documentarlo adecuadamente y a asegurarse de que cada PR cumpliera con los estándares de calidad antes de ser fusionado. De hecho, para asegurar que todo cambio que se realizara estuviera controlado por si fuera necesario volver a versiones anteriores hasta los cambios más pequeños se realizaban mediante PRs. Gracias a esto he conseguido revertir cambios en momentos críticos del desarrollo.

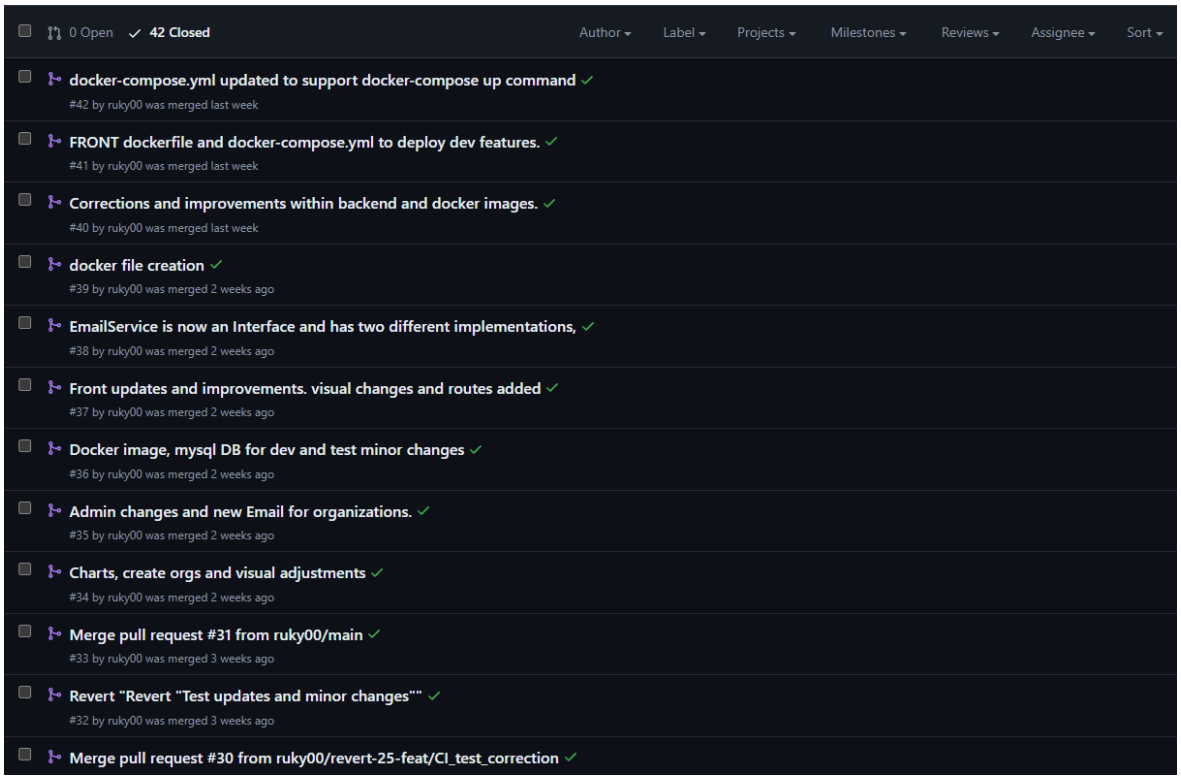


Figura 3.2: PRs del Repositorio.

El total desarrollo se ha realizado en un repositorio de GitHub donde se ha incluido un Readme.md útil para el tanto el despliegue en producción como en desarrollo. De esta forma aún sin la totalidad de los requerimientos, ya sea una cuenta de AWS o de GMail, se podría hacer uso de la aplicación de forma local.

4

Descripción Informática

4.1. Especificación de Requisitos

En esta sección se establecen las necesidades y expectativas a cumplir, así como los elementos clave que guían el diseño y la implementación de la aplicación.

4.1.1. Requisitos Funcionales

La Tabla 4.1 recoge las funcionalidades específicas que el sistema debe ofrecer.

Tabla 4.1: Requisitos Funcionales

RF-001	Login y Registro	El sistema permitirá a los usuarios iniciar sesión con credenciales válidas o registrarse como nuevos usuarios, proporcionando la funcionalidad básica de autenticación y creación de cuentas.
RF-002	Edición completa del perfil por parte de los usuarios	Los usuarios dispondrán de la capacidad total para editar y actualizar la información de su perfil dentro del sistema, lo que incluirá datos personales y otros detalles relevantes, ofreciendo una experiencia de usuario personalizada y adaptable.
RF-003	Autorización de correo electrónico	Cuando un usuario se registre, se enviará un correo electrónico de verificación para asegurar la autenticidad de la dirección de correo electrónico proporcionada.
RF-101	Visualización de cantidad de usuarios registrados por mes	El administrador podrá ver la cantidad de usuarios que se han registrado en el sistema durante cada mes.
RF-102	Visualización de cantidad de usuarios por tipo	El administrador podrá ver la cantidad de usuarios registrados clasificados por tipo (administrador, organizador, cliente) en el sistema.
RF-103	Visualización de cantidad de eventos creados por mes	El administrador podrá ver la cantidad de eventos que se han creado en el sistema durante cada mes.
RF-104	Creación de usuarios de tipo Organización	Se dotará al administrador de la capacidad de crear usuarios con el rol específico de 'Organización', lo que permitirá una gestión más detallada y especializada de usuarios dentro del sistema.

Continúa en la siguiente página

Tabla 4.1: Continuación de la tabla anterior

ID	Resumen	Descripción
RF-105	Filtrado de usuarios por tipo, nombre o email	Se integrará una función que permita al administrador obtener un listado completo de todos los usuarios del sistema, con la posibilidad de aplicar filtros según su tipo de usuario, nombre o dirección de correo electrónico para una búsqueda más eficiente y precisa.
RF-106	Borrado de usuarios por parte del administrador	Se implementará la funcionalidad para que el administrador pueda eliminar usuarios del sistema previa confirmación, permitiendo una gestión eficiente de la base de datos de usuarios y garantizando la integridad y seguridad del sistema.
RF-107	Obtener Usuario en el Sistema	Se implementará la funcionalidad para que el administrador pueda ver información básicas de los usuarios registrados en el sistema
RF-108	Borrado automático	Los usuarios que no hayan confirmado el correo en 15 minutos seran borrados automáticamente del sistema
RF-201	Ver eventos creados por el organizador	El organizador tendrá la capacidad de ver los eventos que ha creado dentro del sistema.
RF-202	Eliminación de eventos por el organizador	Se permitirá al organizador eliminar los eventos que ha creado dentro del sistema.
RF-203	Edición de eventos por el organizador	El organizador podrá editar los eventos que ha creado, lo que incluirá la modificación de detalles como nombre, descripción, fecha, etc.
RF-204	Creación de eventos por el organizador	Se permitirá al organizador crear nuevos eventos proporcionando detalles como nombre, descripción, fecha y, opcionalmente, contraseña para acceder al evento.

Continúa en la siguiente página

Tabla 4.1: Continuación de la tabla anterior

ID	Resumen	Descripción
RF-205	Visualización de cantidad de inscritos en eventos por el organizador	El organizador podrá ver la cantidad de usuarios inscritos en los eventos que ha creado dentro del sistema.
RF-301	Visualización de organizaciones por parte del cliente	Los clientes tendrán la capacidad de ver las organizaciones disponibles en el sistema.
RF-302	Visualización de perfil de organizaciones por parte del cliente	Los clientes podrán ver el perfil de las organizaciones dentro del sistema.
RF-303	Visualización de eventos de organizaciones por parte del cliente	Los clientes podrán ver los eventos que ofrecen las organizaciones dentro del sistema.
RF-304	Inscripción a eventos por parte del cliente	Los clientes podrán inscribirse a los eventos ofrecidos por las organizaciones dentro del sistema.
RF-305	Eventos privados con contraseña	Se permitirá la creación de eventos privados que requieran una contraseña para acceder a ellos.
RF-306	Generación de entradas para eventos	Una vez adquirida la entrada, se generará un código QR para acceder al evento.
RF-307	Manejo de Concurrencia 4.4.4	En caso de que dos usuarios intenten adquirir una entrada simultáneamente, se debe implementar un sistema de colas para evitar incongruencias en la base de datos.
RF-308	Descarga de PDF de la Entrada	Los Clientes tienen que tener la opción de poder descargar la entrada en formato PDF para poder compartirla si quisieran

4.1.2. Requisitos No Funcionales

La Tabla 4.2 recoge las funcionalidades para mejorar la experiencia del usuario, se abordan tanto las limitaciones técnicas como los factores a considerar.

Tabla 4.2: Requisitos No Funcionales

ID	Resumen	Descripción
RNF-001	Desempeño del sistema	El sistema debe ser capaz de manejar hasta 1000 usuarios simultáneos sin experimentar una degradación significativa del rendimiento.
RNF-002	Seguridad de los datos	Todos los datos sensibles almacenados en el sistema deben estar cifrados utilizando un algoritmo de cifrado estándar.
RNF-003	Usabilidad	La interfaz de usuario debe ser intuitiva y fácil de usar, con tiempos de carga de página inferiores a 1 segundo para mejorar la experiencia del usuario.
RNF-004	Conexión a Internet	El sistema debe tener acceso a Internet para permitir la comunicación con servicios externos y el intercambio de datos.
RNF-005	Rendimiento	El sistema debe responder a las solicitudes del usuario dentro de un tiempo aceptable, con tiempos de carga de página y operaciones de procesamiento mínimos.
RNF-006	Accesibilidad	Tiene que ser posible usar la aplicación de forma óptima en dispositivos móviles

4.2. Arquitectura y análisis

Enterprise Event Solutions es una aplicación **Fullstack** creada siguiendo un patrón de diseño MVC. A continuación, se muestra un plano general que será desglosado:

4.2.1. Arquitectura general

Cada uno de los componentes definidos en la Figura 4.1 representa a nivel general los componentes que forman EVS: backend, frontend y base de dato, todas estas dockerizadas y desplegadas en un servicio de AWS EC2 y utilizando Elastic IP para asignar una IP pública fija.

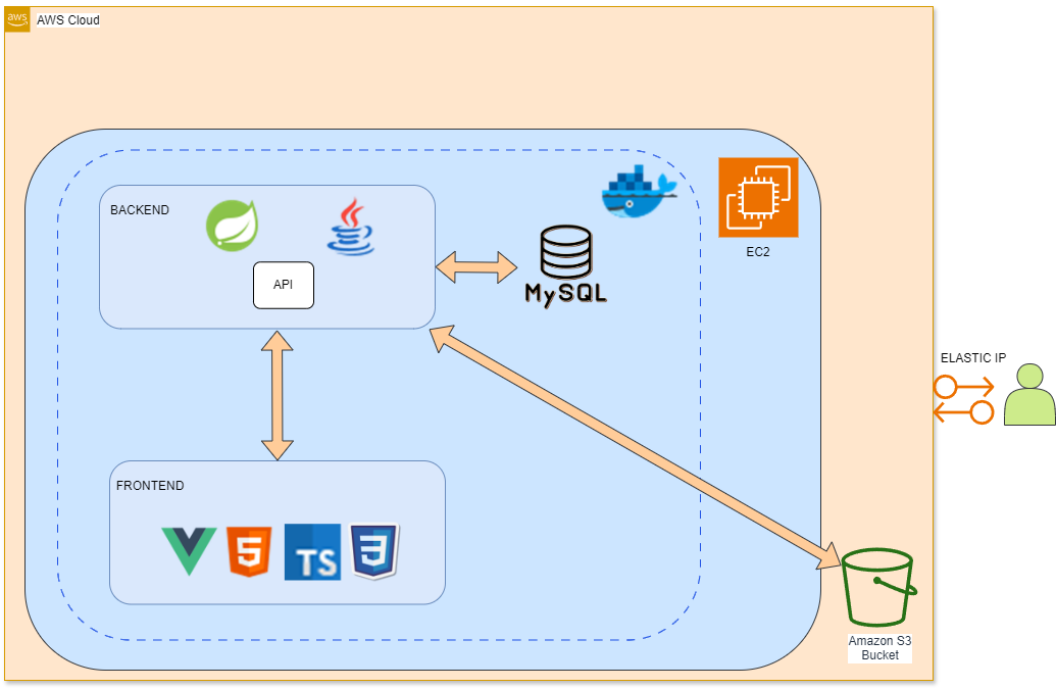


Figura 4.1: Diagrama de Arquitectura de EVS

La figura 4.1 muestra el diagrama de arquitectura de EVS desplegado en la nube de AWS. La arquitectura está dividida en dos componentes principales: el backend y el frontend.

El backend utiliza Spring Boot (Java) para la lógica de negocio y se conecta a una base de datos MySQL. La base de datos está contenedorizada con Docker y se ejecuta en una instancia EC2 de AWS. La API del backend se comunica con el frontend y con la base de datos MySQL.

El frontend está desarrollado con tecnologías web: Vue.js, HTML5, TypeScript y CSS3 son las más relevantes. Los usuarios acceden a la aplicación a través de una dirección IP elástica asignada a la instancia EC2.

Además, el almacenamiento de archivos se maneja utilizando un bucket de Amazon S3, que interactúa con el backend para gestionar los archivos de la aplicación.

En resumen, esta arquitectura de EVS en AWS muestra una solución escalable y eficiente, utilizando tecnologías modernas para el desarrollo web y servicios en la nube para el despliegue y almacenamiento. **Enterprise Event Solutions** combina las fortalezas de **SpringBoot** y **Vue.js** en un patrón MVC para ofrecer una solución robusta y escalable. Con el soporte de **AWS** y **MySQL**, la aplicación está preparada para manejar grandes volúmenes de datos y ofrecer una experiencia de usuario fluida.

4.3. Flujo de la Aplicación

En esta sección se presenta el diagrama de flujo de la aplicación dividida en partes para su mejor comprensión.

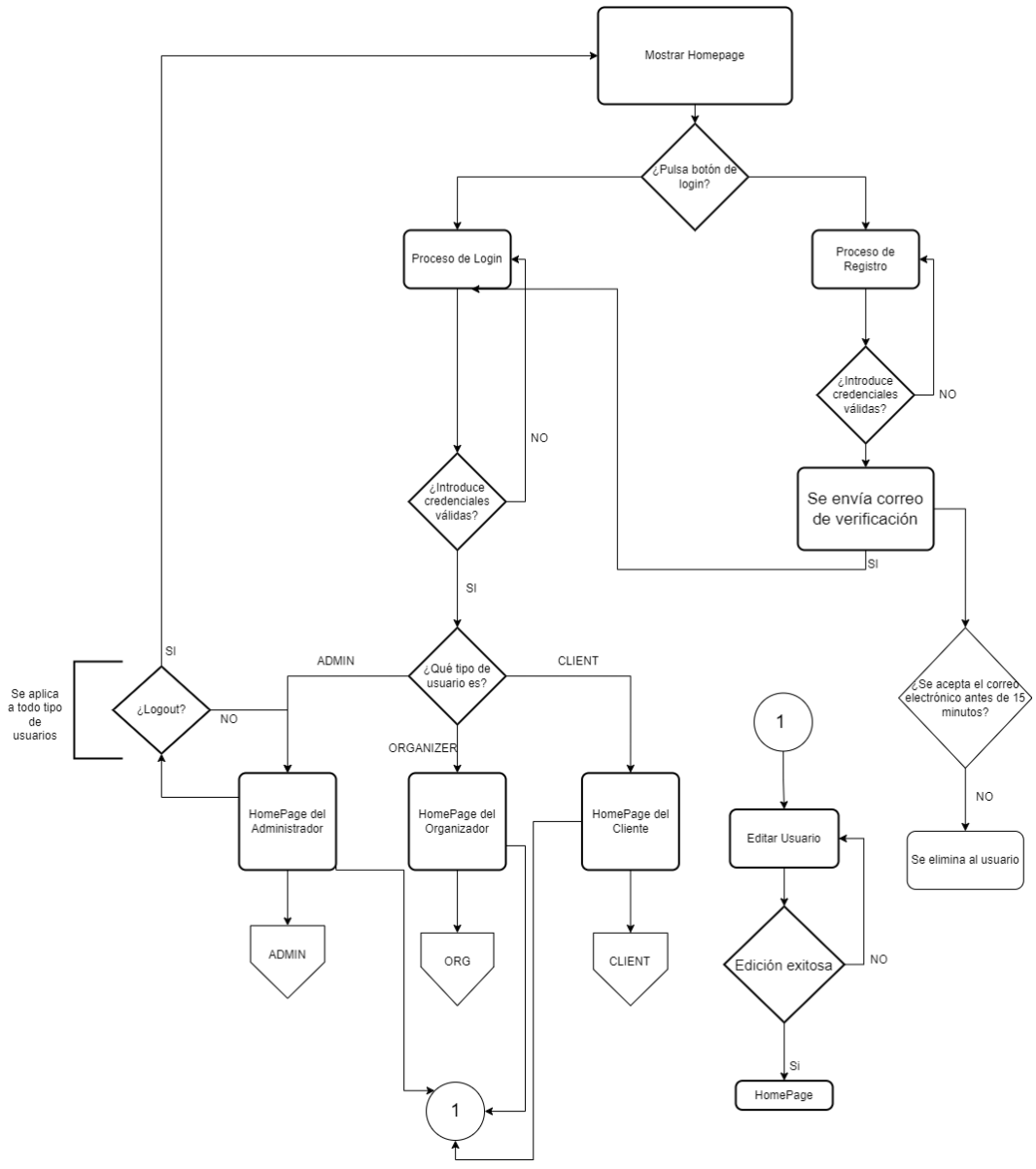


Figura 4.2: Diagrama de Flujo Parte 1

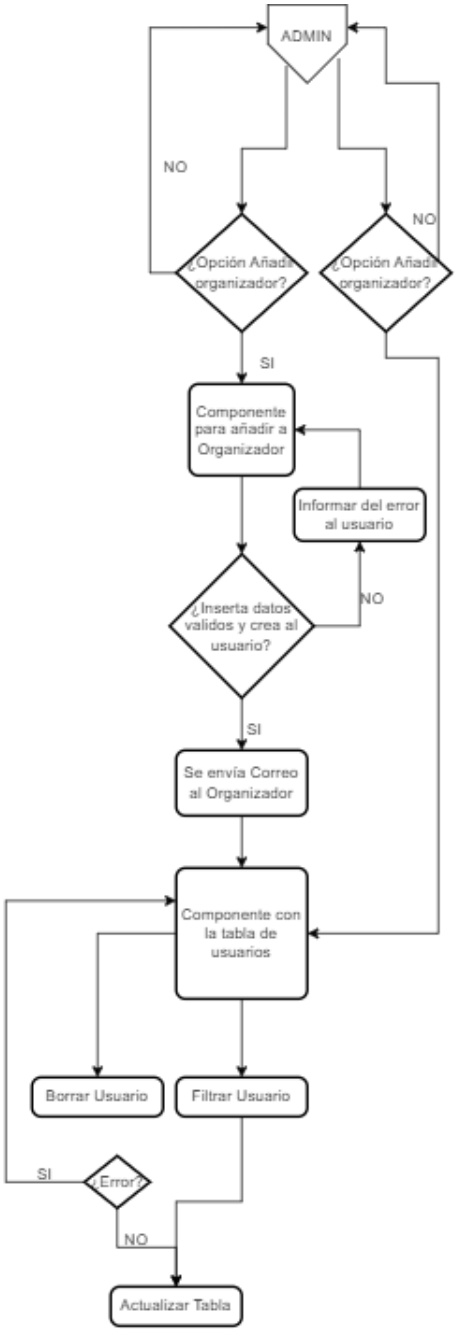


Figura 4.3: Diagrama de Flujo Parte 2

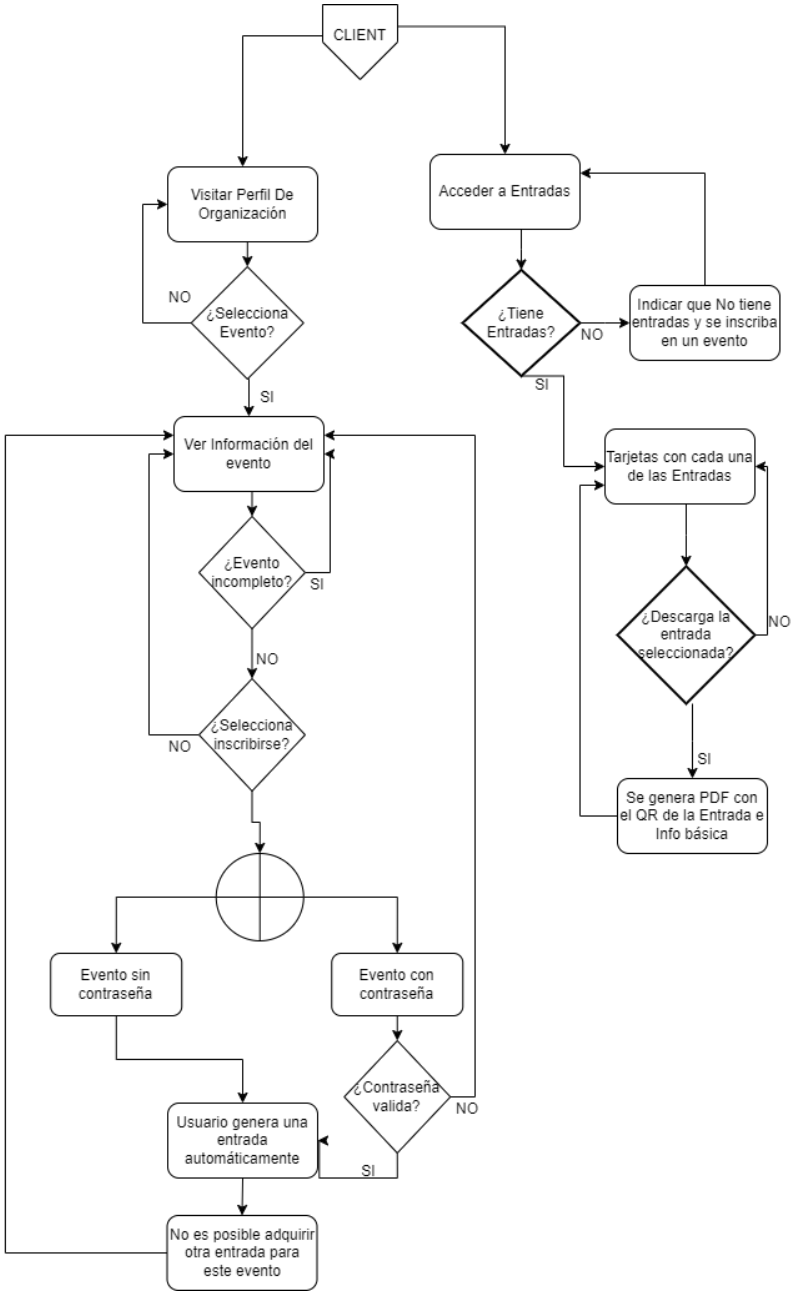


Figura 4.4: Diagrama de Flujo Parte 3

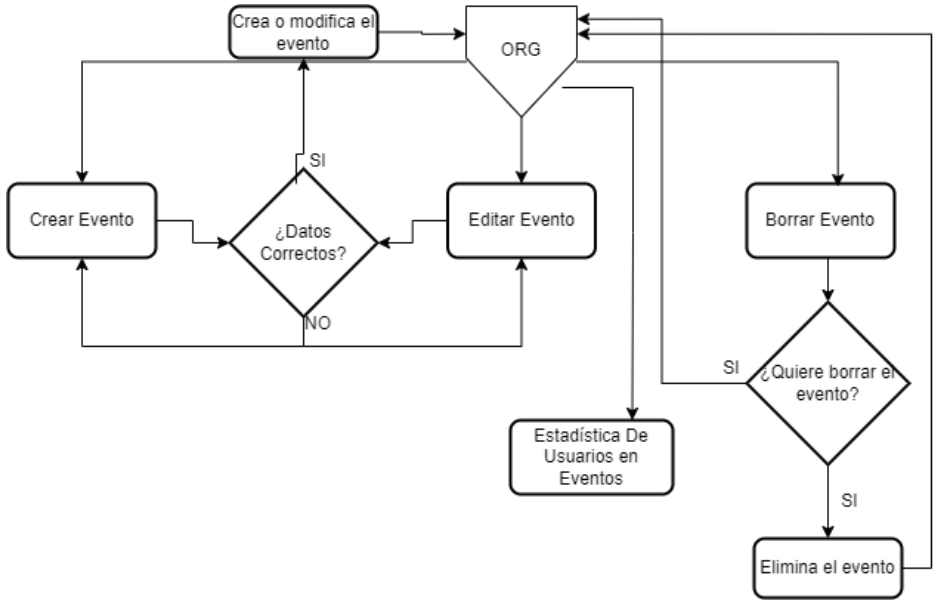


Figura 4.5: Diagrama de Flujo Parte 4

4.4. Diseño e implementación

4.4.1. Modelo

La persistencia en Enterprise Event Solutions se sustenta en una serie de Entidades almacenadas en una base de datos con la que los servicios interactuan mediante las interfaces proporcionadas por JPA 4.4.1. Es el pilar fundamental en el que se sustenta el modelo de negocio de la aplicación.

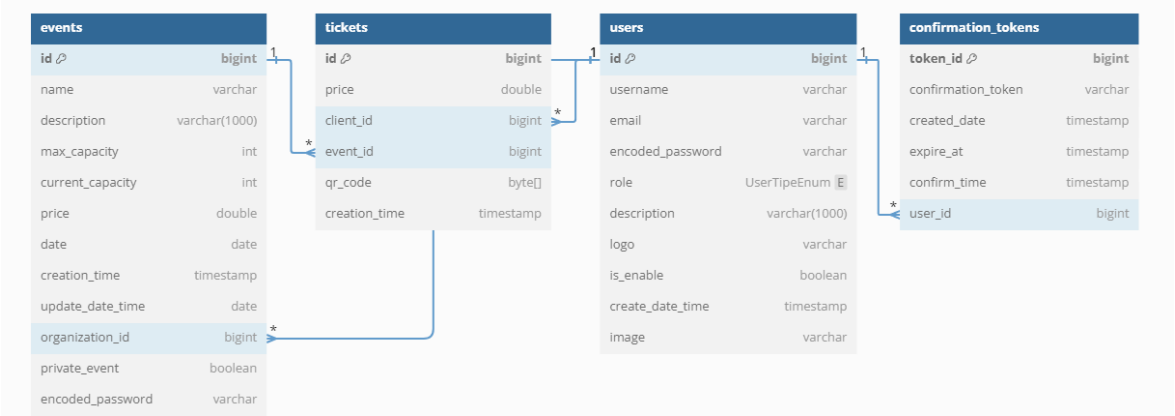


Figura 4.6: Modelo E-R de la BD

JPA

JPA (Java Persistence API) es una especificación de Java que facilita la gestión de la persistencia de datos en aplicaciones Java. En Spring, se usa para simplificar las operaciones CRUD y el manejo de relaciones entre entidades, permitiendo trabajar con objetos en lugar de SQL. Spring Data JPA integra JPA en el ecosistema Spring, proporcionando repositorios predefinidos para operaciones de base de datos comunes.

A continuación se muestra un fragmento de código de Spring Data JPA en Java:

Código 4.1: Ejemplo de Repositorio en Spring Data JPA

```
@Repository
public interface UserRepository extends JpaRepository<User,Long> {
    public Optional<User> findByEmail(String email);
    public List<User> findAllByRole(UserTipeEnum type);
    public Optional<User> findByUsername(String username);
}
```

Como se puede observar, gracias a la gran potencia de Spring y de JPA, simplemente con crear una interfaz con métodos que creemos que vamos a necesitar en nuestros servicios, y sin la necesidad de añadir @Querys, podemos hacer consultas sobre la base de datos.

Pero también podemos hacer nuestras propias Querys sobre la base de datos:

Código 4.2: Ejemplo de Query en Spring Data JPA

```
    @Query("SELECT u FROM _user u WHERE u.isEnable = false AND u.
       createDateTime < :fifteenMinutesAgo")
    List<User> findByEnableFalseAndCreateDateTimeBefore(
       LocalDateTime fifteenMinutesAgo);
```

4.4.2. Controlador

La creación de controladores me ha permitido gestionar un sistema de endpoints que sirve como la interfaz principal para la comunicación entre el cliente y el servidor. Estos controladores se encargan de recibir las solicitudes HTTPS de delegar el procesamiento a los servicios correspondientes. Los servicios encapsulan la lógica de negocio y se comunican con los repositorios JPA para acceder a los datos de la base de datos de manera eficiente. Además, para garantizar la seguridad, se utiliza JWT (JSON Web Tokens), que permite la autenticación y autorización de los usuarios. Los tokens JWT se generan tras un inicio de sesión exitoso y se utilizan para proteger los endpoints, asegurando que solo los usuarios autenticados puedan acceder a recursos específicos. Este enfoque no solo organiza el código de manera modular y mantenible, sino que también proporciona una robusta capa de seguridad para la API.

En la figura 4.7 se puede observar un desglose de las clases del backend y las interrelaciones.

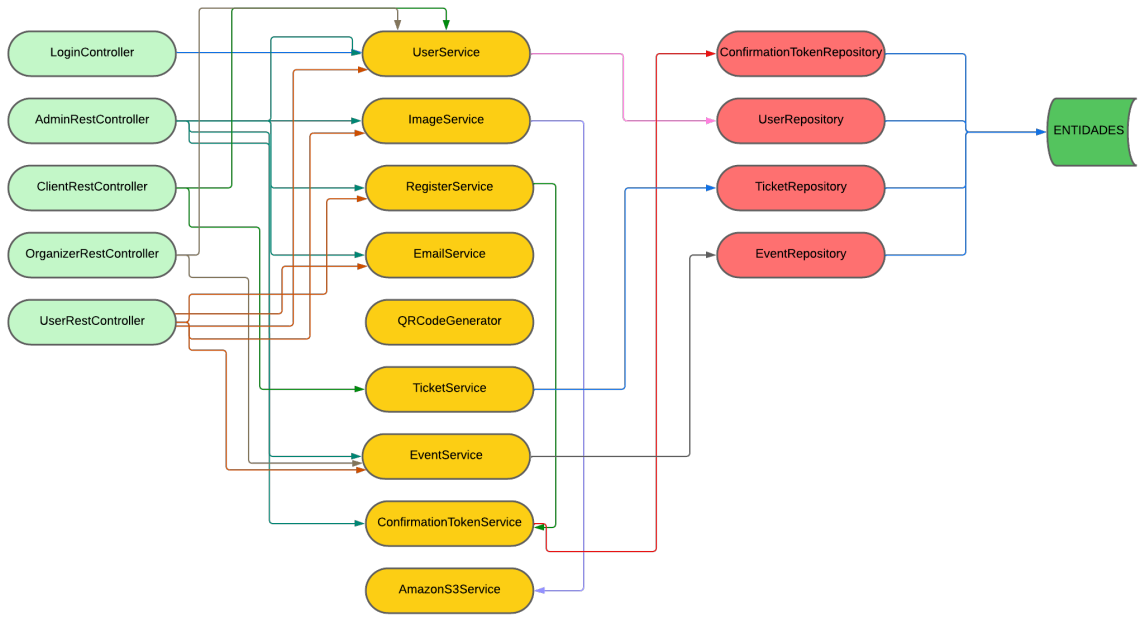


Figura 4.7: Diagrama de Clases de EVS

Además he implementado otras configuraciones para añadir una capa extra de seguridad a la aplicación como configuración CSRF y CORS para limitar las url que pueden hacer uso de los endpoints de la app. Por supuesto se han creado las configuraciones

necesarias para manejar las tecnologías externas como S3, dentro de EVS.

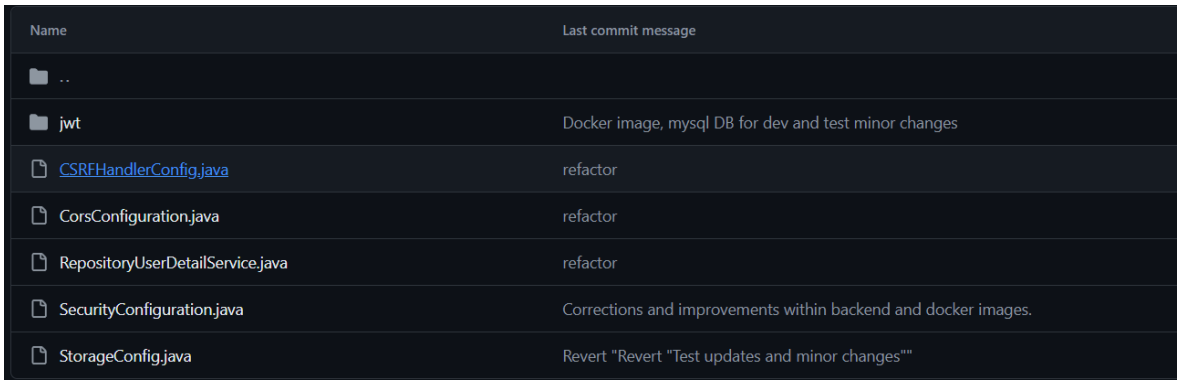


Figura 4.8: Seguridad del backend

Durante la fase de desarrollo se hizo uso de Postman para realizar consultas a la API y verificar el correcto funcionamiento de esta. Además se generó la documentación de la misma con Swagger y en el repositorio de GitHub se encuentra tanto una Colección de Postman para hacer uso de esta API como la documentación de esta en formato JSON.

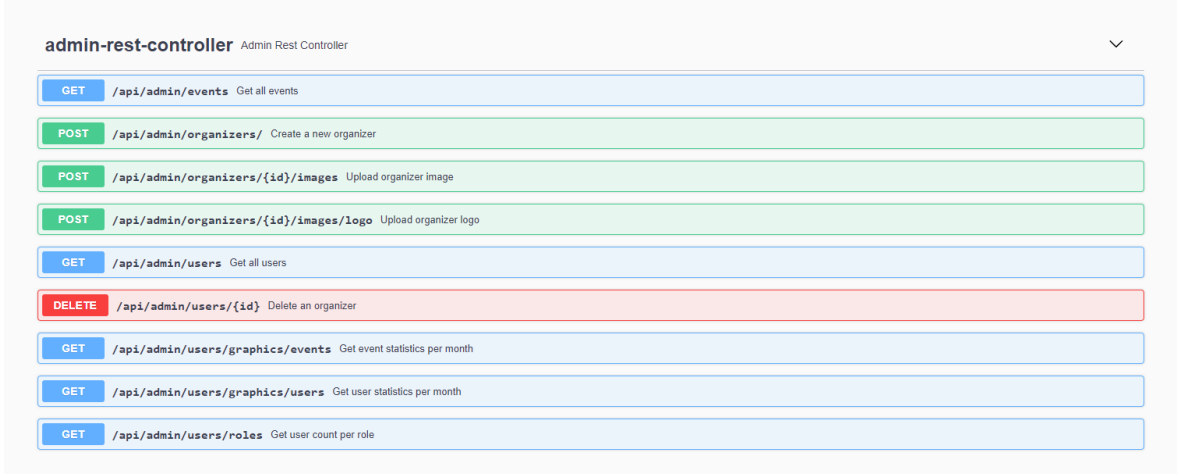


Figura 4.9: Documentación Swagger 1

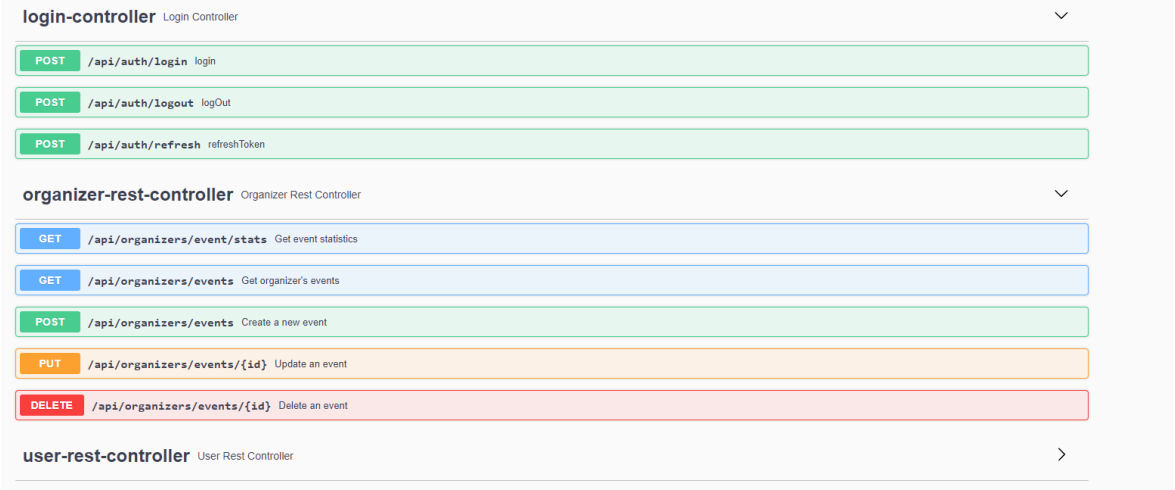


Figura 4.10: Documentación Swagger 2

4.4.3. Vista

El frontend de Enterprise Event Solutions ha sido implementado en Vue.js. Específicamente en Vue3, que trae consigo algunos cambios con respecto a su versión anterior Vue2. La utilización de este framework me ha permitido añadir tanto bibliotecas de componentes y gráficos (Bootstrap5 y Chart.js respectivamente) como relaciones entre los diferentes componentes basado en "props", permitiendo de esta manera diferentes comportamientos del componente dependiendo de que valores se le pasen en el componente padre.

Un ejemplo del uso de estas props sería:

Código 4.3: Ejemplo del Padre

```
1    <event_cards
2    :evento="evento"
3    :is-org="false"
4    >
5    </event_cards>
```

Código 4.4: Ejemplo del Hijo

```
1    <script lang="ts">
```

```
2     import {EventService} from "../services/event.service";
3     import { useRouter } from "vue-router";
4     import { Event } from "../models/Event";
5     import { computed, ref } from "vue";
6     export  default {
7         name: "event_cards",
8         props:{
9           evento: Object as ()=> Event,
10          isOrg: Boolean,
11        },
12    }
```

En el código 4.3, se observa que se inserta en el template un componente con la información del evento. Por otro lado, en el código 4.4, se manejan estos valores que hemos pasado desde el componente padre. En este fragmento, si se le pasa `true` en la variable `is-org`, el componente tendrá un comportamiento diferente a si se le pasa `false`. Ademas en la variable `evento` le pasamos la información del evento para garantizar la reactividad de los componentes de forma individual. De esta forma trabajamos con módulos y no con un elemento.

En las figuras 4.11 y 4.12 se puede ver el mismo componente con Props diferentes.

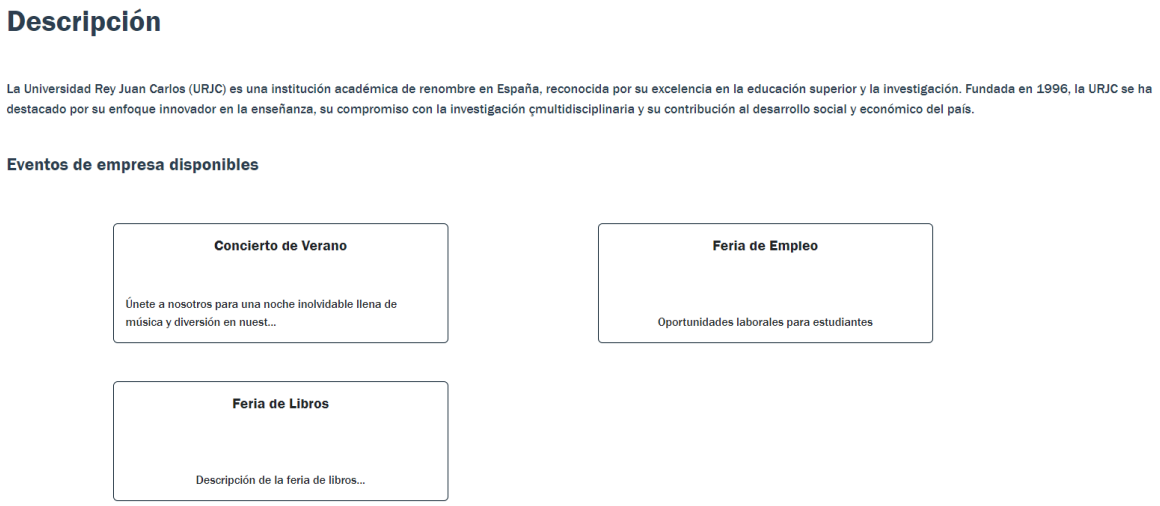


Figura 4.11: Componente Eventos para el Cliente

Figura 4.12: Componente Eventos para el Organizador

4.4.4. Concurrencia en los eventos

La problemática de la adquisición de entradas para eventos con capacidad limitada se presenta cuando una gran cantidad de personas intenta acceder simultáneamente para comprar entradas. Este fenómeno, común en eventos populares, puede causar varios problemas, entre ellos:

- Sobreventa de Entradas: Cuando múltiples usuarios intentan comprar entradas al mismo tiempo, existe el riesgo de que el sistema venda más entradas de las disponibles. Esto ocurre debido a la falta de sincronización adecuada entre las solicitudes concurrentes, lo que puede llevar a la creación de la misma entrada a múltiples usuarios.

- Condiciones de Carrera: Las condiciones de carrera suceden cuando dos o más procesos acceden a recursos compartidos (en este caso, la base de datos de entradas) al mismo tiempo y los resultados dependen del orden en que se ejecutan. Esto puede provocar inconsistencias en los datos y errores en la asignación de entradas.

- Experiencia del Usuario: Un sistema que no maneja adecuadamente la concurrencia puede llevar a tiempos de respuesta lentos, errores y una experiencia de usuario insatisfactoria. Los usuarios pueden experimentar frustración si el sistema falla repetidamente o si no están seguros de si su adquisición ha sido exitosa.

Para abordar estos problemas, se ha utilizado la anotación @Transactional de JPA. Esta solución implica:

- Aislamiento de Transacciones: La anotación @Transactional asegura que cada operación de compra de entradas se ejecute en una transacción aislada. Esto significa que los cambios realizados por una transacción no serán visibles para otras transacciones hasta que se complete. Esto ayuda a prevenir la sobreventa de entradas al garantizar que cada transacción compruebe y reserve las entradas de manera exclusiva.

- Gestión Automática de Transacciones: Con @Transactional, la gestión de transacciones se maneja automáticamente, simplificando el desarrollo y reduciendo la posibilidad de errores humanos. Si una transacción falla, se realiza un rollback automático, asegurando que la base de datos se mantenga en un estado consistente.

- Control de Concurrencia: JPA proporciona varios niveles de aislamiento de transacciones que pueden ser utilizados para controlar la concurrencia. Estos niveles determinan cómo y cuándo los cambios realizados por una transacción se vuelven visibles para otras transacciones, ayudando a evitar las condiciones de carrera.

En resumen, la utilización de @Transactional en JPA permite manejar de manera efectiva la concurrencia durante la adquisición de entradas para eventos. Esta estrategia asegura que cada transacción de adquisición de entradas se procese de forma segura y aislada, manteniendo la integridad de los datos y proporcionando una mejor experiencia de usuario.

Desarrollo de la función

En este apartado se explicará de menor a mayor nivel la implementación de esta funcionalidad y finalmente se explicará el test desarrollado para probarla.

Repositorio

El código 4.5 actualiza la capacidad actual de un evento específico, incrementándola en 1 si no se excede la capacidad máxima permitida. Utiliza una consulta JPQL definida con @Query para actualizar el campo current_capacity, bajo la condición de que la nueva capacidad no supere max_capacity. La consulta se aplica al evento con el ID proporcionado como parámetro. El método devuelve un entero indicando el éxito o el fracaso de la operación. Esto es crucial para prevenir la sobreventa y mantener la consistencia de los datos durante la adquisición de entradas.

Código 4.5: Función incrementCurrentCapacity

```
1    @Transactional
2    @Modifying
3    @Query("UPDATE Event e SET e.current_capacity = e.
        current_capacity + 1 WHERE e.id = :id AND e.current_capacity
         + 1  <= e.max_capacity")
```

```
4     public int incrementCurrentCapacity(@Param("id") Long id);
```

Servicios

En primer lugar, se hace uso de esta función en el servicio EventService. En el código se verifica si un evento específico está lleno al intentar incrementar su capacidad actual en uno. Primero, verifica la existencia del evento por su ID utilizando eventExists(id). Luego, intenta incrementar la capacidad actual del evento mediante eventRepository.incrementCurrentCapacity(id). Si la operación tiene éxito (devolviendo 1), devuelve el evento; de lo contrario, lanza una excepción indicando que el evento está lleno. Este proceso garantiza la integridad de la transacción al manejar concurrencia y prevenir la sobreventa durante la gestión de entradas para eventos.

Código 4.6: Función incrementCurrentCapacity

```
1     public Event checkIfFull(long id){
2         Event event = eventExists(id);
3
4         int i = eventRepository.incrementCurrentCapacity(id);
5         if(i==1){
6             return event;
7         }else{
8             throw new ResponseStatusException(HttpStatus.CONFLICT,"
                  This event is full");
9         }
10    }
11
12    public void deleteEvent(Event event){
13        eventRepository.delete(event);
14    }
15
16    private Event eventExists(Long eventId){
17        Optional<Event> opEvent = eventRepository.findById(eventId)
```

```
            ;
18          if (opEvent.isEmpty()) {
19              throw new ResponseStatusException(HttpStatus.
                    BAD_REQUEST, "Event not longer exist");
20          }
21          return opEvent.get();
22      }
```

En segundo lugar se hace uso de este método en TicketService, donde verdaderamente se adquirirá la entrada para el evento seleccionado. El código 4.7 se encarga de crear un ticket para un usuario en un evento, gestionando dos casos según la privacidad del evento y verificando la capacidad antes de emitir el ticket. Primero, se obtiene el evento y se verifica si está lleno. Si el evento no es privado, se crea el ticket directamente. Si es privado, se verifica la contraseña antes de crear el ticket. Se genera un código QR con los detalles del evento y se guarda el ticket en la base de datos, asegurando la integridad de la transacción y evitando la sobreventa durante la emisión de tickets.

Código 4.7: Función createTicket

```
1       public Ticket createTicket(User user,Long id,String
            passwordToCheck) throws IOException, WriterException {
2           SimpleDateFormat dateFormat = new SimpleDateFormat("dd-MM-
                yyyy HH:mm");
3           Event event = eventService.findById(id).orElseThrow();
4
5           if (!event.isPrivateEvent()){
6               eventService.checkIfFull(id);
7
8               Ticket ticket = new Ticket(user,event);
9               ticket.setPrice(event.getPrice());
10              String formattedDate = dateFormat.format(event.getDate
                    ());
11              saveTicket(ticket);
12              String qrCodeData = "Identificador: " + ticket.getId()
```

```
                + ", Nombre del Evento: " + event.getName() + ",
                Fecha del evento: "+ formattedDate;
13              byte[] qrCodeBytes = qrCodeGenerator.generateQRCode(
                qrCodeData, 200, 200);
14              ticket.setQrCode(qrCodeBytes);
15              saveTicket(ticket);
16
17          return ticket ;
18          }else{
19           if (checkPassword(event.getEncodedPassword(),
                passwordToCheck)){
20            eventService.checkIfFull(id);
21            Ticket ticket = new Ticket(user,event);
22            ticket.setPrice(event.getPrice());
23            String formattedDate = dateFormat.format(event.getDate
                ());
24            saveTicket(ticket);
25            String qrCodeData = "Identificador: " + ticket.getId()
                 + ", Nombre del Evento: " + event.getName() + ",
                Fecha del evento: "+ formattedDate;
26            byte[] qrCodeBytes = qrCodeGenerator.generateQRCode(
                qrCodeData, 200, 200); // Ajusta el tama o del QR
                 seg n tus necesidades
27            ticket.setQrCode(qrCodeBytes);
28            saveTicket(ticket);
29            return ticket ;
30           }
31           else
32               {return null;}
33          }
34      }
```

Controlador

Finalmente la clase ClientRestController, en concreto el método "buyTicket", como se muestra en el código 4.8, se implementa un endpoint POST en un controlador de Spring MVC para comprar un ticket. Utiliza la información del usuario autenticado para buscar su perfil completo. Luego, intenta crear un ticket para un evento especificado, utilizando una contraseña opcional para eventos privados. Si la operación tiene éxito, devuelve el ticket con un estado HTTP 201 Created. Si hay un problema durante la creación del ticket, responde con HTTP 503 Service Unavailable. Si la contraseña para un evento privado no es válida, devuelve HTTP 401 Unauthorized.

Código 4.8: Función buyTicket

```
1    @PostMapping("/tickets/")
2    public ResponseEntity<Ticket> buyTicket(HttpServletRequest
        request, @RequestParam Long id, @RequestBody(required =
        false) String password) {
3        Principal principal = request.getUserPrincipal();
4        User user = userService.findByEmail(principal.getName()).
            orElseThrow();
5        String passwordSent;
6        try {
7            passwordSent = (password == null) ? "" : password;
8            Ticket ticket = ticketService.createTicket(user, id,
                passwordSent);
9            if (ticket != null)
10               return new ResponseEntity<>(ticket, HttpStatus.
                    CREATED);
11       } catch (Exception e) {
12           return new ResponseEntity<>(HttpStatus.
                SERVICE_UNAVAILABLE);
13       }
14       return new ResponseEntity<>(HttpStatus.UNAUTHORIZED);
15   }
```

Test asociado

Como se esperaba, se ha realizado una prueba que verifique el correcto funcionamiento de la gestión de la concurrencia en la adquisición de entradas.

El código 4.9 prueba concurrentemente la compra de tickets para un evento usando múltiples hilos. Primero, se crea un evento de prueba con una capacidad máxima y organizador específicos, guardado mediante eventService.saveEvent(testEvent). Se configura un número elevado de hilos (numberOfThreads) para simular solicitudes concurrentes de compra de tickets. Cada hilo crea un usuario único y realiza una solicitud HTTP simulada usando mockMvc.perform, que intenta comprar un ticket para el evento especificado.

Se utiliza CountDownLatch para coordinar la ejecución simultánea de los hilos: latch para sincronizar el inicio de todos los hilos y doneLatch para esperar a que todos los hilos terminen. Cada hilo verifica el resultado de la solicitud HTTP esperando un código de estado HTTP 201 Created si la compra fue exitosa, o HTTP 409 Conflict si el evento está lleno. Finalmente, se verifica que la capacidad actual del evento se ajuste correctamente al máximo después de todas las operaciones concurrentes.

Código 4.9: Función buyTicket

```
@DisplayName("Concurrent Ticket Purchase Test")
@ParameterizedTest(name = "{index} {0}")
@ValueSource(strings = {"client@urjc.es"})
public void concurrentTicketPurchaseTest(String username) throws
    Exception {
     Event testEvent = new Event();
     testEvent.setName("Test Event");
     testEvent.setMax_capacity(10);
     testEvent.setCurrent_capacity(0);
     Calendar calendar = Calendar.getInstance();
     calendar.set(2025, Calendar.AUGUST, 5, 18, 30);
     testEvent.setDate(calendar.getTime());
     testEvent.setOrganization(organizer);
```

```
13      eventService.saveEvent(testEvent);
14
15      int numberOfThreads = 15; // More than max capacity to test
            concurrency issues
16      CountDownLatch latch = new CountDownLatch(1);
17      CountDownLatch doneLatch = new CountDownLatch(numberOfThreads);
18
19      ExecutorService executorService = Executors.newFixedThreadPool(
            numberOfThreads);
20
21      for (int i = 0; i < numberOfThreads; i++) {
22      int userIndex = i; // Final variable to use inside the lambda
23      executorService.execute(() -> {
24      try {
25          // Creating a unique client user for each thread
26          String uniqueUsername = username + userIndex;
27          User client = new User("Client", uniqueUsername, "pass");
28          userService.saveUser(client);
29
30          System.out.println("User created: " + uniqueUsername);
31
32          latch.await(); // wait for the starting signal
33
34          mockMvc.perform(post("/api/clients/tickets/?id=" +
                testEvent.getId())
35              .with(SecurityMockMvcRequestPostProcessors.user(client.
                    getEmail()).password("pass"))) // Simulating
                    authenticated user
36              .andExpect(result -> {
37                  int status = result.getResponse().getStatus();
38                  if (status == HttpStatus.SC_CONFLICT) {
39                      assertThat(status).isEqualTo(HttpStatus.
                            SC_CONFLICT);
```

```
                } else {
                    assertThat(status).isEqualTo(HttpStatus.
                        SC_CREATED);
                }
            })
            .andDo(print());
    } catch (Exception e) {
        e.printStackTrace();
        System.out.println("Error in thread: " + userIndex + " with
            message: " + e.getMessage());
    } finally {
        doneLatch.countDown();
    }
});
}

latch.countDown(); // Start all threads
doneLatch.await(); // Wait for all threads to finish

// Verify the number of tickets created
Event updatedEvent = eventService.findById(testEvent.getId()).
    orElseThrow();
assertThat(updatedEvent.getCurrent_capacity()).isEqualTo(testEvent.
    getMax_capacity());

executorService.shutdown();
}
```

4.4.5. Problemas enfrentados

Durante la realización del proyecto ha habido que hacer frente a problemas derivados de la falta de conocimiento sobre las herramentas o tecnologías empleadas por lo

que he requerido la ayuda de videos y documentación de las herramientas para poder solventarlo. A continuación hago un resumen de algunos de estos problemas junto con la referencia a sus solución:

- Empaquetado del Frontend erróneo (loading css chunk 323 failed)[6]: Este error no desplegaba correctamente el front de la aplicación empaquetada. Solucioné el error eliminando un import de googleapies.
- Fallo al conectarse a la MySQL(Access denied for user 'root'@'localhost')[7]: El error venía de una mala utilización de las .properties de Spring

4.5. Pruebas

En esta sección, se presentan las pruebas realizadas para asegurar que todos los requisitos se cumplen correctamente y que no hay errores durante la ejecución del sistema. Se incluyen descripciones detalladas de los diferentes test llevados a cabo, así como los resultados obtenidos, con el objetivo de validar el funcionamiento y la estabilidad del software.

Se han realizado tanto pruebas REST como pruebas Unitarias. No se han realizado pruebas de Interfaz.

4.5.1. Pruebas REST

Las pruebas REST son pruebas de software que validan la funcionalidad de las API RESTful, asegurando que los endpoints, métodos HTTP, respuestas y códigos de estado funcionen correctamente.

Para la realización de las pruebas REST se ha incluido también las pruebas de Integración. Cada vez que se realice un conjunto de test, se realiza la conexión con un contenedor Docker con una imagen MySQL de la misma versión que la utilizada en la aplicación para garantizar el correcto funcionamiento sobre esta Base de Datos.

- **Descripción de la prueba:** Verificar que un usuario no registrado puede crear un usuario Cliente.
- **Trazabilidad:** RF-001.
- **Precondiciones:** La aplicación debe estar en ejecución y disponible para recibir solicitudes API.
- **Acciones:**
  1. Crear un nuevo usuario con el rol de 'Client' enviando una solicitud POST a la API con los datos del usuario.
  2. Comprobar que la respuesta HTTP es 201 (Created).
- **Resultados esperados:**
  - El servidor debe responder con un código de estado HTTP 201 (Created) indicando que el usuario ha sido creado correctamente.
  - La información del nuevo usuario debe ser almacenada en la base de datos.
- **Incidencias:**
  - OK. La prueba fue exitosa, el usuario fue creado y la respuesta del servidor fue 201 (Created).

- **Descripción de la prueba:** Verificar que un usuario no registrado puede crear un usuario Organizador/Cliente y luego iniciar sesión correctamente.
- **Trazabilidad:** RF-001.
- **Precondiciones:** La aplicación debe estar en ejecución y disponible para recibir solicitudes API.
- **Acciones:**
  1. Crear un nuevo usuario con el rol de 'Organizer' o 'Client' enviando una solicitud POST a la API con los datos del usuario.

2. Comprobar que la respuesta HTTP es 201 (Created) y que el usuario se ha creado correctamente.
3. Iniciar sesión con el usuario creado enviando una solicitud POST a la API con las credenciales del usuario.
4. Comprobar que la respuesta HTTP es 200 (OK) y que la respuesta indica un inicio de sesión exitoso.

- **Resultados esperados:**
  - El servidor debe responder con un código de estado HTTP 201 (Created) al crear el usuario, indicando que el usuario ha sido creado correctamente.
  - La información del nuevo usuario debe ser almacenada en la base de datos.
  - El servidor debe responder con un código de estado HTTP 200 (OK) al iniciar sesión, indicando un inicio de sesión exitoso.
  - La respuesta de inicio de sesión debe contener un campo 'status' con el valor "SUCCESS".
- **Incidencias:**
  - OK. La prueba fue exitosa, el usuario fue creado y pudo iniciar sesión correctamente.

- **Descripción de la prueba:** Registrarse para un evento.
- **Trazabilidad:** RF-304.
- **Precondiciones:** El sistema debe estar en ejecución y disponible para recibir solicitudes API.
- **Acciones:**
  1. Simular un usuario autenticado con el rol de 'CLIENT'.
  2. Enviar una solicitud POST a la API para registrar al usuario en un evento específico.

3. Verificar que la respuesta HTTP es 201 (Created), lo que indica que la inscripción fue exitosa.

- **Resultados esperados:**
  - El servidor debe responder con un código de estado HTTP 201 (Created), indicando que la inscripción al evento fue exitosa.
- **Incidencias:**
  - OK. La prueba fue exitosa, el usuario se inscribió correctamente.

- **Descripción de la prueba:** Verificar que un usuario puede ver sus entradas.
- **Trazabilidad:** RF-306, RF-304.
- **Precondiciones:** El sistema debe estar en ejecución y disponible para recibir solicitudes API.
- **Acciones:**
  1. Simular un usuario autenticado con el rol de 'CLIENT'.
  2. Enviar una solicitud GET a la API para obtener las entradas del usuario.
  3. Verificar que la respuesta HTTP es 200 (OK) y que la respuesta es un arreglo JSON, lo que indica que las entradas del usuario se recuperaron correctamente.
- **Resultados esperados:**
  - El servidor debe responder con un código de estado HTTP 200 (OK), indicando que la solicitud fue exitosa.
  - La respuesta debe ser un arreglo JSON, que contiene las entradas del usuario.
- **Incidencias:**
  - OK. La prueba fue exitosa, el usuario obtuvo la información de sus entradas.

- **Descripción de la prueba:** Verificar que un usuario puede ver las organizaciones a las que está asociado.
- **Trazabilidad:** RF-301.
- **Precondiciones:** El sistema debe estar en ejecución y disponible para recibir solicitudes API.
- **Acciones:**
  1. Simular un usuario autenticado con el rol de 'CLIENT'.
  2. Enviar una solicitud GET a la API para obtener las organizaciones asociadas al usuario.
  3. Verificar que la respuesta HTTP es 200 (OK) y que la respuesta es un arreglo JSON, lo que indica que las organizaciones se recuperaron correctamente.
- **Resultados esperados:**
  - El servidor debe responder con un código de estado HTTP 200 (OK), indicando que la solicitud fue exitosa.
  - La respuesta debe ser un arreglo JSON, que contiene las organizaciones asociadas al usuario.
- **Incidencias:**
  - OK. La prueba fue exitosa, el usuario obtuvo la información de las organizaciones.

- **Descripción de la prueba:** Verificar que un organizador puede publicar un nuevo evento en el sistema.
- **Trazabilidad:** RF-204.
- **Precondiciones:** El sistema debe estar en ejecución y disponible para recibir solicitudes API.
- **Acciones:**

1. Simular un usuario autenticado con el rol de 'ORGANIZATION'.
2. Crear un nuevo evento con los detalles especificados y convertirlo a formato JSON.
3. Enviar una solicitud POST a la API para publicar el evento.
4. Verificar que la respuesta HTTP es 201 (Created), indicando que el evento se creó correctamente.

- **Resultados esperados:**
  - El servidor debe responder con un código de estado HTTP 201 (Created), indicando que el evento se ha creado correctamente.
- **Incidencias:**
  - OK. La prueba fue exitosa, el usuario pudo crear Eventos.

- **Descripción de la prueba:** Verificar que un organizador puede eliminar un evento existente del sistema.
- **Trazabilidad:** RF-202.
- **Precondiciones:** El sistema debe estar en ejecución y disponible para recibir solicitudes API. Además, debe existir al menos un evento en la base de datos.
- **Acciones:**
  1. Simular un usuario autenticado con el rol de 'ORGANIZER'.
  2. Obtener la lista de eventos disponibles.
  3. Verificar que la respuesta HTTP es 200 (OK) y que contiene al menos un evento.
  4. Identificar el ID del primer evento en la lista para eliminarlo.
  5. Enviar una solicitud DELETE a la API para eliminar el evento utilizando su ID.

6. Verificar que la respuesta HTTP es 200 (OK), indicando que el evento se eliminó correctamente.

- **Resultados esperados:**
  - El servidor debe responder con un código de estado HTTP 200 (OK), indicando que el evento se ha eliminado correctamente.
- **Incidencias:**
  - OK. La prueba fue exitosa, el usuario pudo eliminar un Evento.

4.5.2. Pruebas Unitarias

Las pruebas unitarias se realizan para garantizar que partes específicas del código funcionen como se espera, sin depender de otras partes del sistema. Esto ayuda a detectar errores temprano, asegurar la calidad del código, facilitar la refactorización y proporcionar documentación sobre el comportamiento esperado del código.

A continuación, se muestran las PU correspondientes a los requisitos que no se han probado con pruebas REST para ampliar la cobertura del código.

- **Descripción de la prueba:** Verificar que todos los eventos de un organizador pueden ser obtenidos.
- **Trazabilidad:** RF-201.
- **Precondiciones:** El usuario debe estar autenticado como 'Carlos' con el rol de 'CLIENT'.
- **Acciones:**
  1. Configurar el comportamiento del mock de UserService para que devuelva el organizador cuando se llame a findByUsername con el argumento 'URJC'.
  2. Configurar el comportamiento del mock de EventService para que devuelva una lista falsa de eventos cuando se llame a findByUser con el organizador.

3. Realizar una solicitud GET a la API para obtener los eventos, especificando el parámetro 'org' como 'URJC'.
4. Verificar que la respuesta HTTP es 200 (OK) y que el cuerpo de la respuesta contiene una lista de eventos con un tamaño de 2.
5. Verificar que se llamó al método findByUser en EventService con el organizador como argumento.

- **Resultados esperados:**
  - Se espera que la solicitud obtenga una respuesta con estado HTTP 200 (OK).
  - Se espera una lista de tamaño 2.
- **Incidencias:**
  - OK. La prueba fue exitosa.

- **Descripción de la prueba:** Verificar que un evento puede ser obtenido por su ID.
- **Trazabilidad:** RF-303.
- **Precondiciones:** El usuario debe estar autenticado como 'Carlos' con el rol de 'CLIENT'.
- **Acciones:**
  1. Configurar el comportamiento del mock de EventService para que devuelva el evento1 cuando se llame a findById con el argumento 1L.
  2. Realizar una solicitud GET a la API para obtener el evento con ID 1.
  3. Verificar que la respuesta HTTP es 200 (OK) y que el cuerpo de la respuesta contiene el evento con ID 1.
  4. Verificar que se llamó al método findById en EventService con el argumento 1L.

- **Resultados esperados:**
  - Se espera que la solicitud obtenga una respuesta con estado HTTP 200 (OK).
  - Se espera que contenga el evento con ID 1.
- **Incidencias:**
  - OK. La prueba fue exitosa.

- **Descripción de la prueba:** Verificar que todos los usuarios pueden ser obtenidos por un administrador.
- **Trazabilidad:** RF-107.
- **Precondiciones:** El usuario debe estar autenticado como 'Admin' con el rol de 'ADMIN'.
- **Acciones:**
  1. Configurar el comportamiento del mock de UserService para que devuelva una lista de usuarios falsos cuando se llame a findAll.
  2. Realizar una solicitud GET a la API para obtener todos los usuarios.
  3. Verificar que la respuesta HTTP es 200 (OK) y que el cuerpo de la respuesta contiene una lista de usuarios con al menos 2 elementos.
  4. Verificar que se llamó al método findAll en UserService.
- **Resultados esperados:**
  - Se espera que la solicitud obtenga una respuesta con estado HTTP 200 (OK)
  - Se espera que contenga una lista de usuarios con al menos 2 elementos.
- **Incidencias:**
  - OK. La prueba fue exitosa.

4.6. Distribución y despliegue

En esta sección se describen los pasos y procesos seguidos para desplegar la aplicación web en producción.

4.6.1. Despliegue

Para empaquetar la aplicación se ha utilizado Docker. Se ha creado una imagen de la aplicación y se ha lanzado junto con un contenedor de MySQL para la persistencia de datos. Esto permite ejecutar la aplicación sin enfrentarse a problemas de dependencias.

4.6.2. Cloud

Uno de los objetivos a cumplir era el Despliegue Continuo (CI/CD), es decir, cada vez que se realizara un cambio en la aplicación, se automatizara el proceso de despliegue en la máquina EC2 de AWS.

Este proceso de despliegue automatizado incluye los siguientes pasos:

1. **Ejecución de Pruebas**: Cuando se realiza un commit o merge en la rama principal (main) del repositorio, se ejecutan automáticamente las pruebas relacionadas con la aplicación para asegurar que no hay errores introducidos por los nuevos cambios.

2. **Creación de la Imagen Docker**: Tras la ejecución exitosa de las pruebas, se inicia un GitHub Action que crea una nueva imagen Docker de la aplicación.

3. **Despliegue en EC2**: Una vez que se completa la creación de la imagen Docker, se ejecuta otro GitHub Action que se conecta a la instancia de EC2 de AWS mediante SSH. Esta instancia tiene las siguientes características:

    - Tipo de instancia: t2.micro
    - Sistema operativo: Ubuntu 20.04 LST

- Recursos: 1 vCPU, 1 GB de RAM
- Asignación de IPv4 elástica
- 8 GiB de memoria de almacenamiento

4. **Ejecución de Script de Despliegue**: El GitHub Action que se conecta a la instancia de EC2 ejecuta un script que está preconfigurado en la máquina. Este script se encarga de gestionar los contenedores Docker, detener la versión antigua de la aplicación, eliminar los contenedores obsoletos, y lanzar la nueva imagen Docker.

En el código 4.10 se puede observar el Workflow que se encarga de lanzar un nuevo contenedor la con versión más reciente de la aplicación.

Código 4.10: Action de Automatización CD

```
name: Despliegue en EC2

on:
  workflow_run:
    workflows: ["Construir y Subir Imagen Docker"]
    types:
      - completed

jobs:
  deploy:
    runs-on: ubuntu-latest

    steps:
      - name: Execute deployment script on EC2
        uses: easingthemes/ssh-deploy@v5.0.3
        with:
          SSH_PRIVATE_KEY: ${{ secrets.EC2_SSH_KEY }}
          REMOTE_HOST: ${{ secrets.EC2_HOST }}
          REMOTE_USER: ${{ secrets.EC2_USER }}
```

```
20
21              SCRIPT_AFTER: |
22                cd EnterpriseEventSolutions
23                sudo ./cd.sh
```

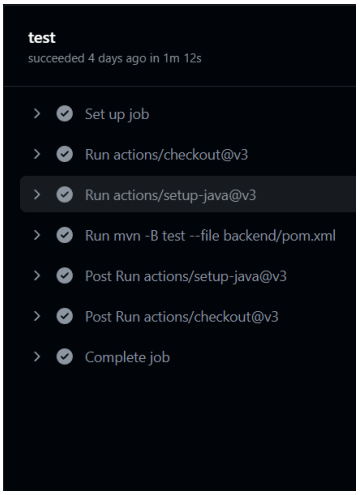
Dentro de la EC2 se ejecuta el código 4.11. Este código se encarga de hacer pull de la nueva imagen de EVS, y relanzar el contenedor de esta con las dependencias pertinentes

Código 4.11: Script de automatización de CD

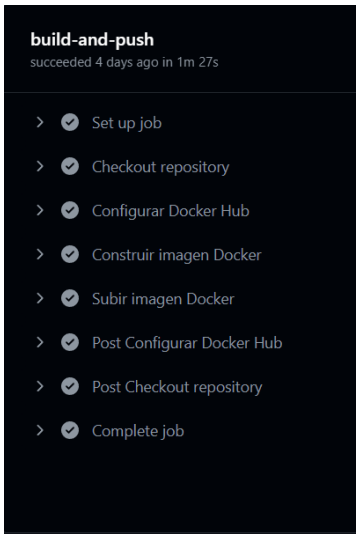
```
1   #!/bin/bash
2
3   # Variables de entorno
4   export EMAIL_SERVICE="**"
5   export EMAIL_TFG="**"
6   export AWS_SECRET_ACCESS_KEY="**"
7   export AWS_REGION="**"
8   export AWS_ACCESS_KEY_ID="**"
9
10  # Nombre de la imagen de Docker
11  IMAGE_NAME="ruky00/evsglobal"
12
13  # Nombre del servicio de la aplicacion en docker-compose.yml
14  SERVICE_NAME="app_prod"
15
16  # Hacer pull de la nueva imagen de Docker
17  echo "Haciendo pull de la nueva imagen de Docker..."
18  docker pull $IMAGE_NAME
19
20  # Detener el contenedor actual de la aplicaci n
21  echo "Deteniendo el contenedor actual de la aplicaci n..."
22  docker-compose stop $SERVICE_NAME
23
24  # Eliminar el contenedor actual de la aplicacion
```

```
25      echo "Eliminando el contenedor actual de la aplicacion..."
26      docker-compose rm -f $SERVICE_NAME
27
28      # Iniciar el contenedor de la aplicaci n nuevamente con la
            nueva versi n de la imagen
29      echo "Iniciando el contenedor de la aplicaci n con la nueva
            version de la imagen..."
30      docker-compose up -d $SERVICE_NAME
31
32      echo "Despliegue completado."
```

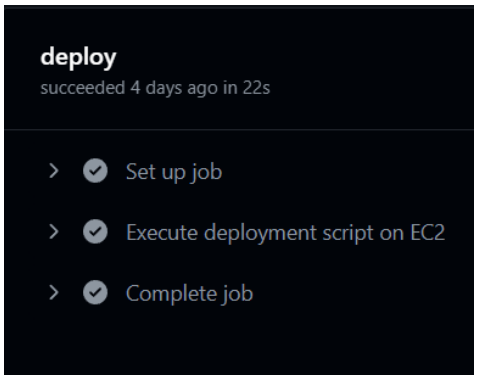
En la figura 4.13 se puede observar, en orden de ejecución, el flujo de las Actions de Github.



(a) Ejecución de los tests



(b) Creación de imagen Docker



(c) Conexión SSH con EC2

Figura 4.13: Flujo de trabajos de GitHub

4.7. Estudio de Caso

Dado que Enterprise Event Solutions es una aplicación diseñada para usuarios comunes, se ha llevado a cabo un estudio para recopilar opiniones y percepciones sobre la aplicación. En este estudio se incluyen preguntas tanto antes como después del uso de la aplicación por parte de los usuarios. Para la recolección de datos, se utilizó Google Forms, y los resultados obtenidos se presentan en la sección de Apéndice.

4.7.1. Participantes

Debido a que la aplicaciones tiene diferentes tipos de usuarios he optado por seleccionar una serie de estos participantes para que interactuaran como organizadores, como clientes o como administradores. En total se ha evaluado a 10 Participantes. Cabe destacar que para la primera interacción con la aplicación se le ha requerido al Participante realizar una serie de pasos en la aplicación para valorar posteriormente la **usabilidad** y **accesibilidad** de esta.

4.7.2. Valoración general

En esta sección se desglosaran las preguntas y se expondrán los resultados.

Preguntas

Para poder completar el estudio se han realizado las siguientes preguntas:

1. Edad
2. Uso habitual de dispositivos electrónicos
3. ¿Sueles acudir a eventos de Empresas u Organizaciones?
4. ¿Sueles leer los correos que te envían empresas promocionando eventos?
5. Como cliente ¿Cuál es tu opinión acerca de una aplicación que concentre la gestión de Eventos con la gestión de tus Entradas?
6. ¿Darías uso a esta aplicación?
7. Como Empresa ¿Cuál es tu opinión acerca de una aplicación que concentré la gestión tus Eventos con la gestión de la asistencia a estos?
8. ¿Darías uso a esta aplicación?
9. Después de haber probado la aplicación. ¿Con que asiduidad la usarías?

10. ¿Cuál es tu opinión general sobre Enterprise Event Solutions?

Resultados

Los resultados del estudio de caso sobre la aplicación Enterprise Event Solutions son muy positivos. El 80 % de los usuarios indicaron que la usarían con frecuencia, y el 100 % la calificaron entre buena y excelente. Esto sugiere que la aplicación cumple con las expectativas y necesidades de los usuarios, demostrando ser una herramienta valiosa y bien recibida en su mercado objetivo. Estos resultados reflejan su eficacia y utilidad, y señalan un alto potencial para su adopción y recomendación futura.

En el apéndice A.1 se pueden ver los resultados desglosados de las preguntas realizadas a los participantes durante el Estudio.

4.7.3. Desempeño

Los usuarios detectaron un problema con la responsividad de la página en dispositivos móviles. Este problema fue corregido tan pronto como se identificó el error.

4.7.4. Diseño, Usabilidad y Accesibilidad

En las primeras versiones de la aplicación, se observó que la interfaz gráfica carecía de ciertas facilidades para la navegación, como elementos identificativos, códigos QR, flechas de navegación, spinners de carga y pantallas de error adecuadas. Además, el diseño dificultaba su uso, lo que llevó a una reestructuración completa de la página.

Es importante destacar que estos comentarios se recopilaron a lo largo del ciclo de vida del proyecto, y no únicamente en su etapa final. Por lo tanto, considero que el contacto continuo con el usuario final es fundamental para lograr un producto refinado. En rasgos generales los participantes consideran Enterprise Event Solutions una aplicación sencilla y fácil de entender.

5

Trabajos futuros y conclusiones

5.1. Futuros proyectos

Como creador de Enterprise Event Solutions, y el significado que tiene para mí, tengo intención de continuar mejorando Enterprise Event Solutions hasta su límite. Algunas funcionalidades extra que podría incluir serían:

- Software de lectura de QR integrada: Validar las entradas desde la propia aplicación para poder acceder al evento.
- Pasarela de pago: Los eventos que sean de pago puedan ser abonados desde la propia aplicación.
- Ampliar el sistema de Correo: Añadir correos de alerta cuando se aproxima la

fecha de un evento.

Las posibilidades con una aplicación de este tipo son infinitas. Estas son algunas de las posibles mejoras a corto-medio plazo.

5.2. Conclusiones

Pese a que ya existen numerosas aplicaciones para gestionar eventos, he querido darle otro enfoque centrado en el la gente de a pie y en las pequeñas empresas, EVS permite concentrar la gestión de clientes y eventos en un mismo sitio ahorrando infinidad de recursos tanto materiales como personales.

La gran aceptación que ha tenido entre los participantes del Estudio de Caso es para mí una recompensa al esfuerzo y las horas dedicadas a este TFG, y por supuesto me animan a continuar con el desarrollo.

Por otro lado me gustaría recalcar que, gracias a realizar este proyecto, he adquirido conocimientos útiles para el desarrollo laboral y personal y por supuesto no se quedará ahí ya que he descubierto que cada vez me gusta más el mundo del Desarrollo de Aplicaciones Web aunque si me tengo que decantar, me quedo con el Backend.

Bibliografía

[1] P. PYME. (2024) La dgipyme publica el informe 'cifras pyme' de enero de 2024. Accedido: 18/05/2024. [Online]. Available: https://plataformapyme.es/es-es/Paginas/noticias-detalles-simple.aspx?idnoticia=832

[2] S. Boot. (2024) Spring boot. Accedido: 20/05/2024. [Online]. Available: https://spring.io/

[3] swagger. (2024) Api development foreveryone. [Online]. Available: https://swagger.io/

[4] Vue.js. (2024) Vue.js. Accedido: 20/05/2024. [Online]. Available: https://vuejs.org/

[5] Docker. (2024) Docker. Accedido: 20/05/2024. [Online]. Available: https://docs.docker.com/

[6] deb. (2024) Chunkloaderror: Loading chunk failed (vuejs + webpack). [Online]. Available: ChunkLoadError:Loadingchunkfailed(VueJS+Webpack)

[7] K. Kariyawasam. (2024) java.sql.sqlexception: Access denied for user 'root'@'localhost' - can't create connection with mysql. [Online]. Available: https://stackoverflow.com/questions/67159538/java-sql-sqlexception-access-denied-for-user-rootlocalhost-cant-create

Apéndices

A

Estudio de Caso

A.1. Formulario Estudio de Caso

A continuación se muestran los datos recogidos del Estudio de Caso:

https://forms.gle/vdjaMq6vTJrPTLeh7

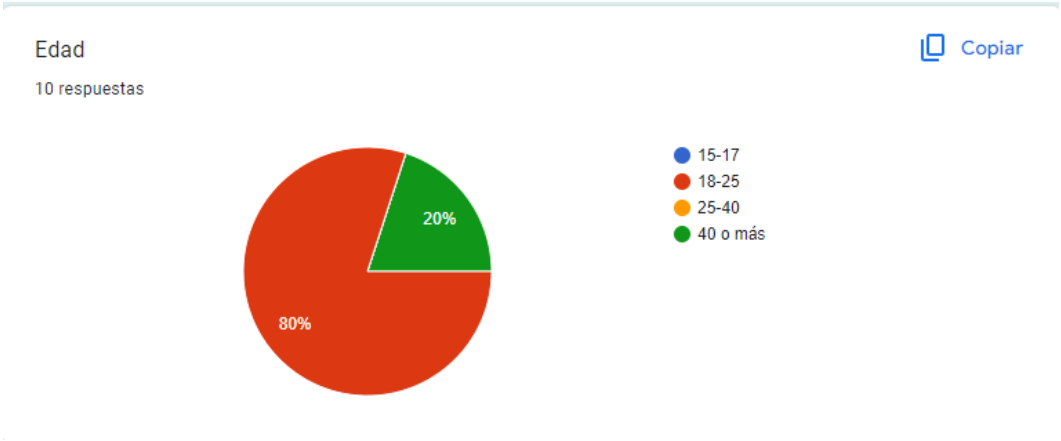


Figura A.1: Formulario Pregunta 1

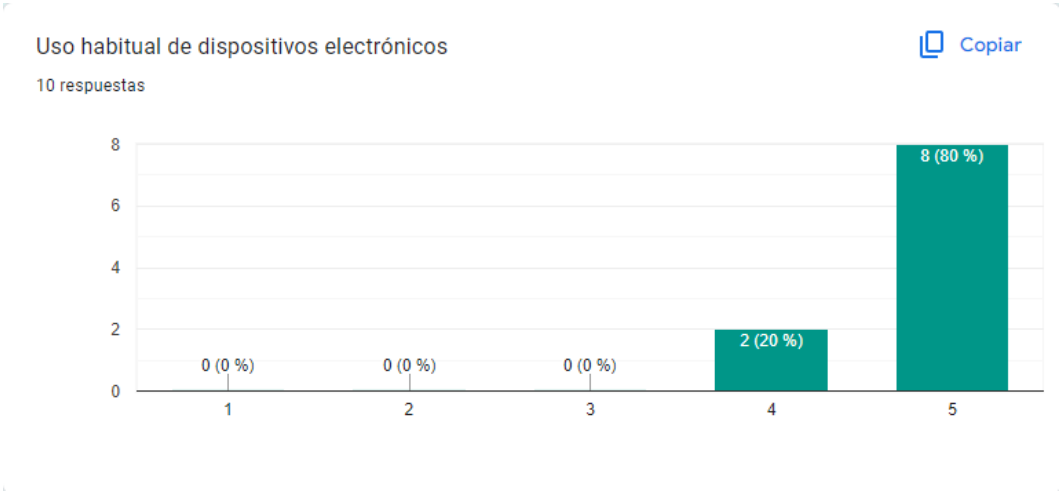


Figura A.2: Formulario Pregunta 2

Figura A.3: Formulario Pregunta 3

Figura A.4: Formulario Pregunta 4

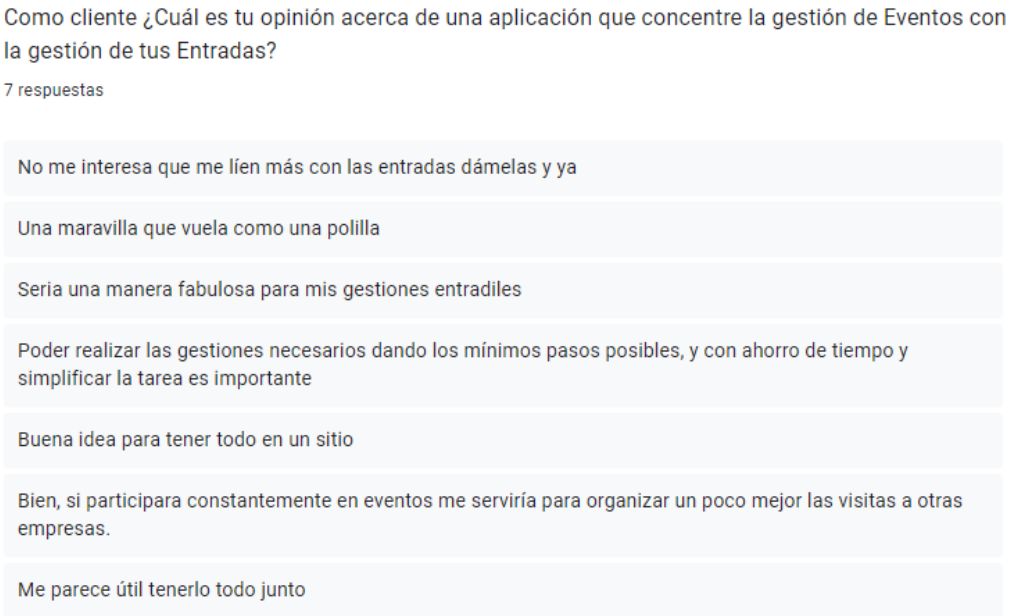


Figura A.5: Formulario Pregunta 5

Como Empresa ¿Cuál es tu opinión acerca de una aplicación que concentré la gestión tus Eventos con la gestión de la asistencia a estos?

7 respuestas

- Como empresa me llamaría la atención porque con eso puedo recopilar más datos sobre mis clientes.
- Una buena manera de hacer conocer a tu empresa
- Es una buena manera de fortalecer la difusión de los eventos gaming que organizo
- Creo que es útil, conocer todos los datos relativos a datos concentrado en estadísticas da una visión rápida de si funciona o no
- Buena iniciativa
- Me seria más útil para poder visualizar de una manera sencilla la participación de personas en los eventos que se organicen en mi empresa.
- Me parece útil tenerlo todo junto

Figura A.6: Formulario Pregunta 6

¿Darías uso a esta aplicación?

Copiar

10 respuestas

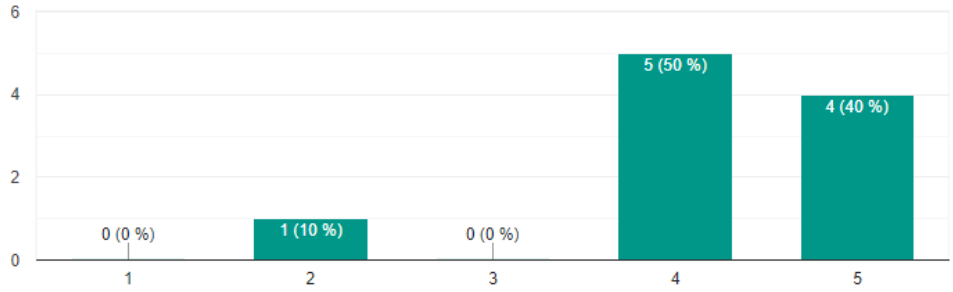


Figura A.7: Formulario Pregunta 7

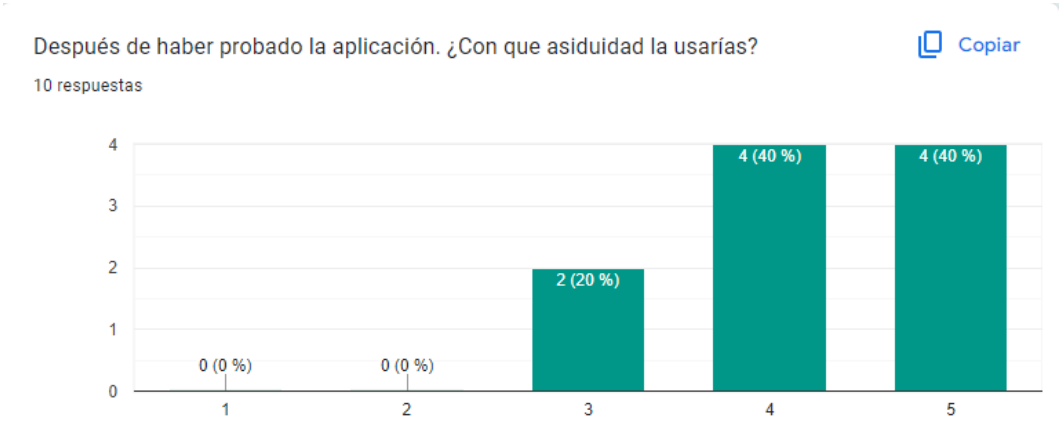


Figura A.8: Formulario Pregunta 8

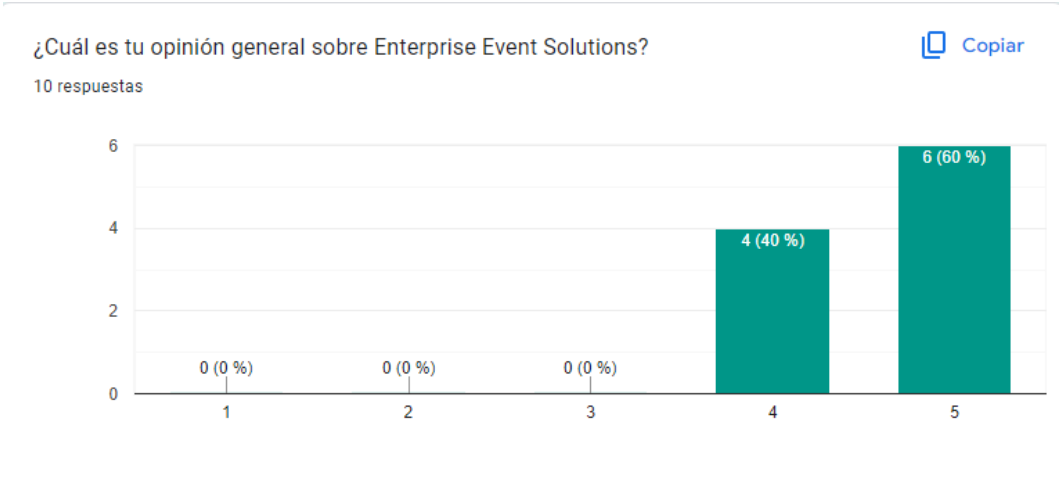


Figura A.9: Formulario Pregunta 9

B

Lanzamiento en Local

B.1. Requisitos

Se recomienda de disponer un sistema operativo Linux o en su defecto un subsistema Linux dentro de nuestro propio ordenador.

- **Docker**:Para instalar Docker sigue las instrucciones de su página principal.[1]
- **npm y Node.js**Para el frontend es necesario estas dos programas. Se pueden instalar desde su sitio oficial.[2]

[1]https://docs.docker.com/get-docker/
[2]https://docs.npmjs.com/downloading-and-installing-node-js-and-npm

B.2. Lanzamiento en producción

Para lanzar la aplicación hay que seguir los siguientes pasos:

B.2.1. Clonar repositorio de Github

Código B.1: Script clonar repositorio

```
git clone https://github.com/ruky00/EnterpriseEventSolutions.git
cd EnterpriseEventSolutions
```

B.2.2. Construir imagen Docker

Ejecuta el siguiente comando:

Código B.2: Crear imagen Docker

```
docker build -t evsglobal -f multistage-dockerfile.Dockerfile .
```

En caso de querer automatizar este proceso, hay un script .sh creado en la carpeta 'backend' que crea y sube al repositorio de DockerHub una nueva versión de la aplicación.

B.2.3. Lanzar aplicación en producción

w

Variables de entorno

Para poder hacer uso de la aplicación en un entorno local en producción hay que definir las variables de entorno de las tecnologías de apoyo para la aplicación, S3 y servicio de correo:

- AWS_ACCESS_KEY_ID: $AWS_ACCESS_KEY_ID
- AWS_SECRET_ACCESS_KEY: $AWS_SECRET_ACCESS_KEY
- AWS_REGION: $AWS_REGION
- EMAIL_TFG: $EMAIL_TFG
- EMAIL_SERVICE: $EMAIL_SERVICE

docker-compose

Finalmente se ejecuta comando 'docker-compose up' que lanzará las imagenes tanto de la base de datos MySQL como la imagen con la última versión de la aplicación.

Código B.3: Lanzar contenedores Docker

```
version:  3.8
services:

  mysql:
    image: mysql:8.0.28
    container_name: evs1
    environment:
      - MYSQL_DATABASE=evs
      - MYSQL_ROOT_PASSWORD=password
    ports:
      - "3306:3306"
    restart: on-failure

  app_prod:
    image: ruky00/evsglobal
    ports:
      - "8443:8443"
    environment:
      - SPRING_PROFILES_ACTIVE=prod
      - SPRING_DATASOURCE_URL=jdbc:mysql://evs1:3306/evs
      - SPRING_DATASOURCE_USERNAME=root
      - SPRING_DATASOURCE_PASSWORD=password
      - AWS_ACCESS_KEY_ID=${AWS_ACCESS_KEY_ID}
      - AWS_SECRET_ACCESS_KEY=${AWS_SECRET_ACCESS_KEY}
      - AWS_REGION=${AWS_REGION}
      - EMAIL_TFG=${EMAIL_TFG}
      - EMAIL_SERVICE=${EMAIL_SERVICE}
    depends_on:
```

```
29        - mysql
30      restart: on-failure
```

Verificar que la aplicación funciona

Abre el navegador y accede a la dirección https://localhost:8443. En caso de que salte alguna advertencia de seguridad, desestímala y continua a la app.

B.3. Lanzar aplicación en desarrollo

En el caso de querer hacer uso de la aplicación sin necesidad de tener una cuenta de AWS ni un correo electrónico, se puede lanzar la aplicación con una versión en desarrollo que carece de las dependencias de S3 y de correo electrónico.

Además de esta forma se pueden realizar cambios en cada una de las secciones de la app sin tener que parar la totalidad de la aplicación.

Ejecutar backend

Ejecuta los comandos B.4 para construir el ejecutable de la aplicación.

Código B.4: Construir ejecutable

```
1 cd backend
2 mvn package
```

Una vez finalizada la ejecución de los test se creará un archivo .jar. Ejecuta el comando B.5 para lanzar el backend.

Código B.5: Lanzar ejecutable

```
1 java -jar "-Dspring.profiles.active=dev" .\target\
     EnterPriseEventSolutions-0.0.1-SNAPSHOT.jar
```

Ejecutar frontend

Por otro lado tenemos que lanzar el fortend. En primer lugar instalamos las dependencias.

Código B.6: Instalar dependencias del frontend

```
cd frotend/EnterPriseEventSolutions
npm i
```

Y ejecutamos el comando B.7 para lanzar el front en desarollo y poder realizar cambios que se verán al momento.

Código B.7: Lanzar frontend

```
npm run serve
```

Verificar correcto funcionamiento

Abre el navegador y accede a la dirección http://localhost:8080 que es donde esta alojado el frontend para verificar que el front funciona correctamente. y hacemos un post de un usuario nuevo para verificar la conexión con el backend.

B.4. Usuarios de Prueba

Para acceder a la aplicación y probar las diferentes características de esta se tiene que hacer con una serie de usuarios con los permisos correspondientes.

- Usuario Administrador: usuario: admin@admin.com contraseña:pass
- Usuario Organización: usuario: inaki@example.com contraseña:pass
- Usuario Cliente: usuario: michel@example.com contraseña:pass

C

Enlace a Enterprise Event Solutions

C.1. Enlace a la aplicación

Se puede hacer uso de la aplicación accediendo al siguiente enlace: **https://18.133.60.104:8443/**

C.2. Enlace al repositorio de GitHub

Se puede acceder al repositorio de EVS en el siguiente enlace:

https://github.com/ruky00/EnterpriseEventSolutions