



Universidad de Valladolid

**ESCUELA DE INGENIERÍA INFORMÁTICA
DE SEGOVIA**

**Grado en Ingeniería Informática
de Servicios y Aplicaciones**

Fest.io

Alumno: Alejandro Martín Martín & Héctor Torres Bolufer

Tutor: Samuel Encinas Plaza

Índice general

Lista de figuras	III
Lista de tablas	V
I Memoria del Proyecto	3
1. Descripción del proyecto	5
1.1. Objetivos y alcance del proyecto	5
1.2. Motivación	5
1.3. Identificación del entorno tecnológico	6
1.4. Organización del documento	6
1.5. Estado del arte	7
1.5.1. Contexto del mercado	7
1.5.2. Análisis de alternativas de software	7
2. Metodología	9
2.1. Arquitectura	10
2.1.1. Arquitectura lógica detallada	10
2.1.2. Arquitectura lógica de alto nivel	11
2.1.3. Arquitectura física	12
3. Conclusiones	13
3.1. Conclusiones generales	13
3.2. Lineas de futuro	14
II Documentación técnica	15
4. Análisis	17
4.1. Descripción de actores	17
4.2. Diagrama de Casos de Uso	21
4.3. Requisitos	21
4.4. Requisitos Funcionales	23

4.5. Requisitos No Funcionales	26
4.5.1. Restricciones de Tecnología y Desarrollo	26
4.5.2. Seguridad y Privacidad	26
4.5.3. Rendimiento y Eficiencia	26
4.5.4. Disponibilidad y Robustez	26
4.5.5. Unidad y Diseño	27
4.5.6. Mantenibilidad	27
5. Diseño	29
5.1. Modelo Relacional	29
5.2. Diccionario de datos	30
5.2.1. Diccionario de entidades	30
5.2.2. Diccionario de relaciones	33
5.3. Diseño de Interfaz	33
5.4. Diagramas de entidad relación	34
5.5. Diagramas de de secuencia	35
6. Implementación	39
6.1. Paquetes desarrollados	39
6.2. Servidores de aplicación utilizados	39
6.3. Servidor de base de datos utilizado	39
6.4. Tecnologías utilizadas	39
6.5. Herramientas utilizadas	40
6.6. Detalles concretos	40
7. Pruebas	41
7.1. Pruebas de caja negra	41
III Manuales de la Aplicación	47
8. Manual de Instalación	49
8.0.1. Requisitos previos	49
8.0.2. Configurar las variables de entorno	49
8.0.3. Levantar la aplicación	50
8.0.4. Acceder a la aplicación	50
8.0.5. Parar la aplicación	50
9. Manual de Usuario	53
9.1. Manual de Usuario	53
9.1.1. Registro como cliente	53
9.1.2. Inicio de sesión	53
9.1.3. Panel de cliente	53
9.1.4. Explorar festivales	54

9.1.5. Comprar un abono	54
9.1.6. Ver tu ticket	54
9.1.7. Recargar saldo	54
9.1.8. Ajustes de cuenta	54
9.2. Manual de Administración	55
9.2.1. Perfil: Administrador del sistema	55
9.2.2. Perfil: Empresa (gestor de festivales)	55
10. Declaración de IAG	59
Bibliografía	61

Índice de figuras

2.1. Diagrama de la arquitectura logica detallada de la plataforma	10
2.2. Diagrama de la arquitectura logica de alto nivel de la plataforma	11
2.3. Diagrama de la arquitectura física de la plataforma	12
4.1. Diagrama de casos de uso	21
5.1. Diccionario de datos de la entidad ABONO_CLIENTE	30
5.2. Diccionario de datos de la entidad ARTISTA	30
5.3. Diccionario de datos de la entidad CLIENTE	31
5.4. Diccionario de datos de la entidad EMPRESA	31
5.5. Diccionario de datos de la entidad FESTIVAL	32
5.6. Diccionario de datos de la entidad TIPO_ABONO	32
5.7. Diccionario de datos de la entidad TRANSACCION	32
5.8. Diccionario de datos de la relación ACTUAR_FESTIVAL	33
5.9. Diccionario de datos de la relación REALIZAR_TRANSACCION	33
5.10. Diagrama entidad-relación	34
5.11. Diagrama de secuencia para registrar empresa gestora	36
5.12. Diagrama de secuencia para verificar empresa gestora	37

Índice de cuadros

4.1.	CU-01. Registrar empresa gestora	22
4.2.	CU-02. Verificar empresa gestora	23
4.3.	RF-01. Formulario de registro de empresa gestora	23
4.4.	RF-02. Captura de datos de empresa gestora	23
4.5.	RF-03. Validación de campos obligatorios	23
4.6.	RF-04. Validación de formato de datos	24
4.7.	RF-05. Comprobación de unicidad del correo electrónico	24
4.8.	RF-06. Almacenamiento de empresa gestora	24
4.9.	RF-07. Estado inicial pendiente de verificación	25
4.10.	RF-08. Acceso a aplicación empresarial tras registro	25
4.11.	RF-09. Notificación de cuenta pendiente de verificación	25
4.12.	RF-10. Consulta de empresas gestoras registradas	25
4.13.	RF-12. Verificación o restricción de empresa gestora	26
7.1.	TP-01. Inicio de sesión de cliente válido	41
7.2.	TP-02. Inicio de sesión de empresa válido	42
7.3.	TP-03. Inicio de sesión de admin válido	42
7.4.	TP-04. Consulta de datos de cliente	43
7.5.	TP-05. Visibilidad de empresa propia	43
7.6.	TP-06. Control de acceso a empresa ajena	44
7.7.	TP-07. Creación de festival	44
7.8.	TP-08. Añadir artista a festival	45
7.9.	TP-09. Añadir ticket a festival	45
7.10.	TP-10. Compra de ticket válida	46

*Dedicado a
todos los festivales a los que hemos ido*

Parte I

Memoria del Proyecto

Capítulo 1

Descripción del proyecto

El proyecto consiste en el desarrollo de *fest.io*, una plataforma digital para la gestión y venta de abonos de festivales. En ella intervienen tres tipos de usuarios: un administrador, encargado de verificar y bloquear empresas gestoras; las empresas gestoras, que pueden registrar y gestionar festivales; y los clientes, que pueden consultar eventos y comprar abonos. La plataforma permitirá centralizar la publicación de festivales, la compra segura de entradas mediante pasarela de pago y la generación de códigos QR únicos para cada abono. De este modo, *fest.io* ofrece una solución unificada para facilitar tanto la gestión de eventos por parte de las empresas como la experiencia de compra de los clientes.

1.1. Objetivos y alcance del proyecto

El objetivo general de este proyecto es diseñar y desarrollar *fest.io*, una plataforma web orientada a la gestión de festivales y a la venta de abonos.

A nivel personal, este trabajo permite aplicar y consolidar los conocimientos adquiridos durante la asignatura de **Plataformas de Software Empresariales** aplicando también los conceptos de asignaturas como **Metodología de Sistemas de Software Informáticos**. A nivel teórico, el proyecto aborda conceptos como el análisis de requisitos, el modelado de casos de uso y la definición de una arquitectura software adecuada. A nivel práctico, construir una aplicación funcional que permita el registro de usuarios, la verificación de empresas gestoras, la publicación y gestión de festivales, la compra de abonos y la generación de entradas con códigos QR, tanto servidor que procesa información recibida como el cliente que envía esa información.

1.2. Motivación

La motivación principal de este trabajo surge del interés por desarrollar una aplicación web completa que permita aplicar, en un caso práctico, los conocimientos adquiridos a lo largo del grado.

Desde un punto de vista académico, con este trabajo concluye la parte práctica de la asignatura por lo que la motivación se ciñe en aprobarla.

1.3. Identificación del entorno tecnológico

La aplicación se ha desarrollado en un entorno web con arquitectura cliente-servidor. El sistema está formado por una parte frontend, encargada de la interfaz de usuario, y una parte backend, encargada de la lógica de negocio, el acceso a datos y la comunicación con servicios externos.

Frontend

- Node.js
- Express
- TypeScript
- JWT para autenticación
- bcrypt para cifrado de contraseñas
- Prisma para acceso a PostgreSQL

Backend

- Node.js
- Express
- TypeScript
- JWT para autenticación
- bcrypt para cifrado de contraseñas
- Prisma para acceso a PostgreSQL
- TailwindCSS para el diseño de interfaces

Base de Datos

- PostgreSQL

Servicios externos

- API de pagos: <https://webservices.samuelencinas.dev/api#/Payments>
- API musical: <https://webservices.samuelencinas.dev/api#/Music>
- API de imágenes: <https://api.unsplash.com>

1.4. Organización del documento

El documento se estructura en varios capítulos que permiten seguir el proyecto desde su planteamiento inicial hasta su valoración final.

En el capítulo de **Descripción del proyecto** se introduce la plataforma *fest.io*, explicando el contexto general, los objetivos, la motivación y el entorno tecnológico utilizado para su desarrollo.

En el capítulo de **Metodología** se describe la arquitectura de la aplicación y se presentan los principales diagramas que ayudan a comprender la organización lógica y física del sistema.

Finalmente, en el capítulo de **Conclusiones** se analiza el grado de cumplimiento de los objetivos planteados, se valoran los aprendizajes obtenidos durante el desarrollo del proyecto y se proponen posibles mejoras o ampliaciones futuras.

1.5. Estado del arte

1.5.1. Contexto del mercado

El sector de la música en vivo en España está en pleno auge, congregando a más de 20 millones de personas en 2025 (8 millones en festivales) y recaudando 807 millones de euros (un 11.2 % más que en 2024). Las entradas representan el 50 % de los ingresos totales del mercado, dominando la venta digital a través del móvil (70 % de cuota), lo que mitiga el fraude y garantiza un acceso seguro.

1.5.2. Análisis de alternativas de software

Eventbrite

- **Descripción:** Plataforma líder global de gestión de eventos en autoservicio con presencia en 180 países.
- **Funcionamiento:** Permite crear eventos en tres pasos (detalles, entradas y formularios). Genera páginas web dedicadas, mapas de butacas interactivos y automatiza campañas de marketing integrándose con herramientas como Mailchimp.
- **Ventajas:** Reconocimiento de marca global, modelo freemium (gratis para <25 entradas) y control de accesos ágil mediante código de barras móvil.
- **Desventajas:** Tarifas de servicio elevadas en grandes volúmenes, personalización limitada y ausencia de soluciones de marca blanca o autenticación única (SSO).

DICE (Fever)

- **Descripción:** Software y app móvil de venta de entradas enfocado en evitar la reventa abusiva y la especulación de precios.
- **Funcionamiento:** Las entradas viven exclusivamente en la app como un código QR dinámico. Se sincroniza con Spotify y Apple Music para ofrecer recomendaciones personalizadas a los usuarios.
- **Ventajas:** Política nativa anti-reventa con listas de espera, precios finales transparentes sin cargos ocultos y fuerte crecimiento de usuarios activos.
- **Desventajas:** Obliga estrictamente al uso de su app móvil (no permite PDF ni formato físico), cobertura limitada en festivales multidía e incertidumbre corporativa tras su reciente compra por Fever.

Ticketmaster / Live Nation

- **Descripción:** El mayor operador de venta de entradas del mundo, integrado en el conglomerado Live Nation, con contratos de exclusividad en grandes recintos.
- **Funcionamiento:** Modelo B2C a gran escala. Gestiona aforos masivos, asientos numerados, control de acceso físico y posee su propio canal oficial de reventa secundaria.
- **Ventajas:** Infraestructura robusta probada con millones de usuarios simultáneos, escala global única y acceso exclusivo a grandes recintos.
- **Desventajas:** Comisiones abusivas (cargos de servicio superiores al 30 %), nula capacidad de autogestión para organizadores pequeños y carece de tecnologías para festivales como el sistema cashless (pulseras).

Weeetix

- **Descripción:** Solución integral de venta de entradas de origen neerlandés, diseñada específicamente para festivales de cualquier escala.
- **Funcionamiento:** Ofrece pasarelas de pago integrables (iFrame) en la web del organizador, analíticas del recorrido de compra y una app que valida hasta 1.000 entradas por hora mediante smartphones.
- **Ventajas:** Especialización total en festivales, modelo de marca blanca que respeta la identidad visual del organizador y soporte técnico in situ.
- **Desventajas:** Menor reconocimiento de marca orgánico, funciones de marketing muy básicas y ausencia de un marketplace centralizado propio (depende del tráfico del organizador).

Entradium

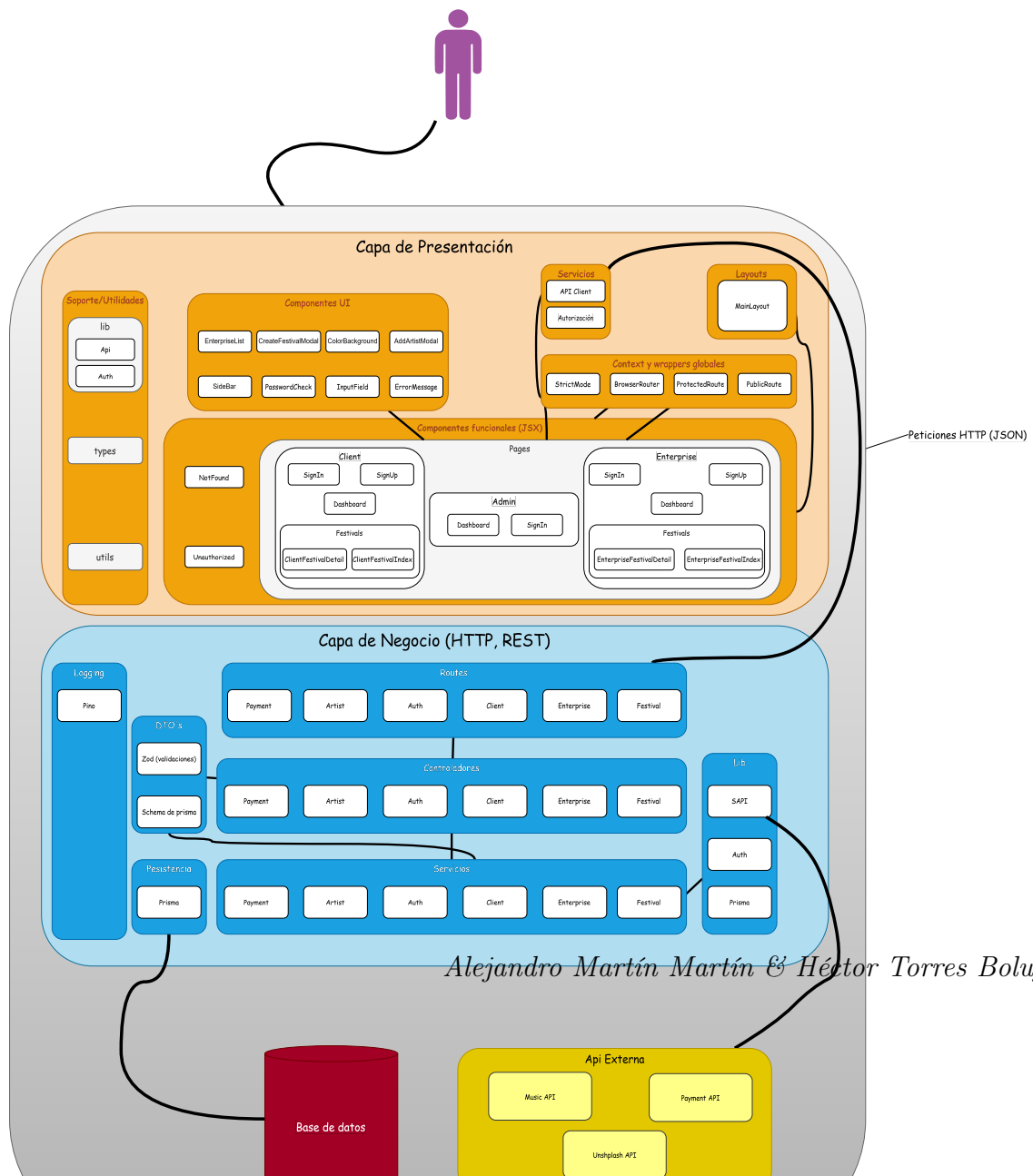
- **Descripción:** Plataforma española de autoservicio orientada al mercado nacional y enfocada en eventos y festivales de formato pequeño o mediano.
- **Funcionamiento:** Herramienta simplificada donde el organizador gestiona de forma autónoma el alta del evento, las tipologías de entradas y el cobro.
- **Ventajas:** Los gastos de gestión más bajos del mercado español, interfaz 100 % local y modelo freemium accesible para promotores emergentes.
- **Desventajas:** Escala reducida sin proyección internacional, carece de analítica profunda, sistemas cashless o apps nativas avanzadas de control de accesos.

Capítulo 2

Metodología

2.1. Arquitectura

2.1.1. Arquitectura lógica detallada



2.1.2. Arquitectura lógica de alto nivel

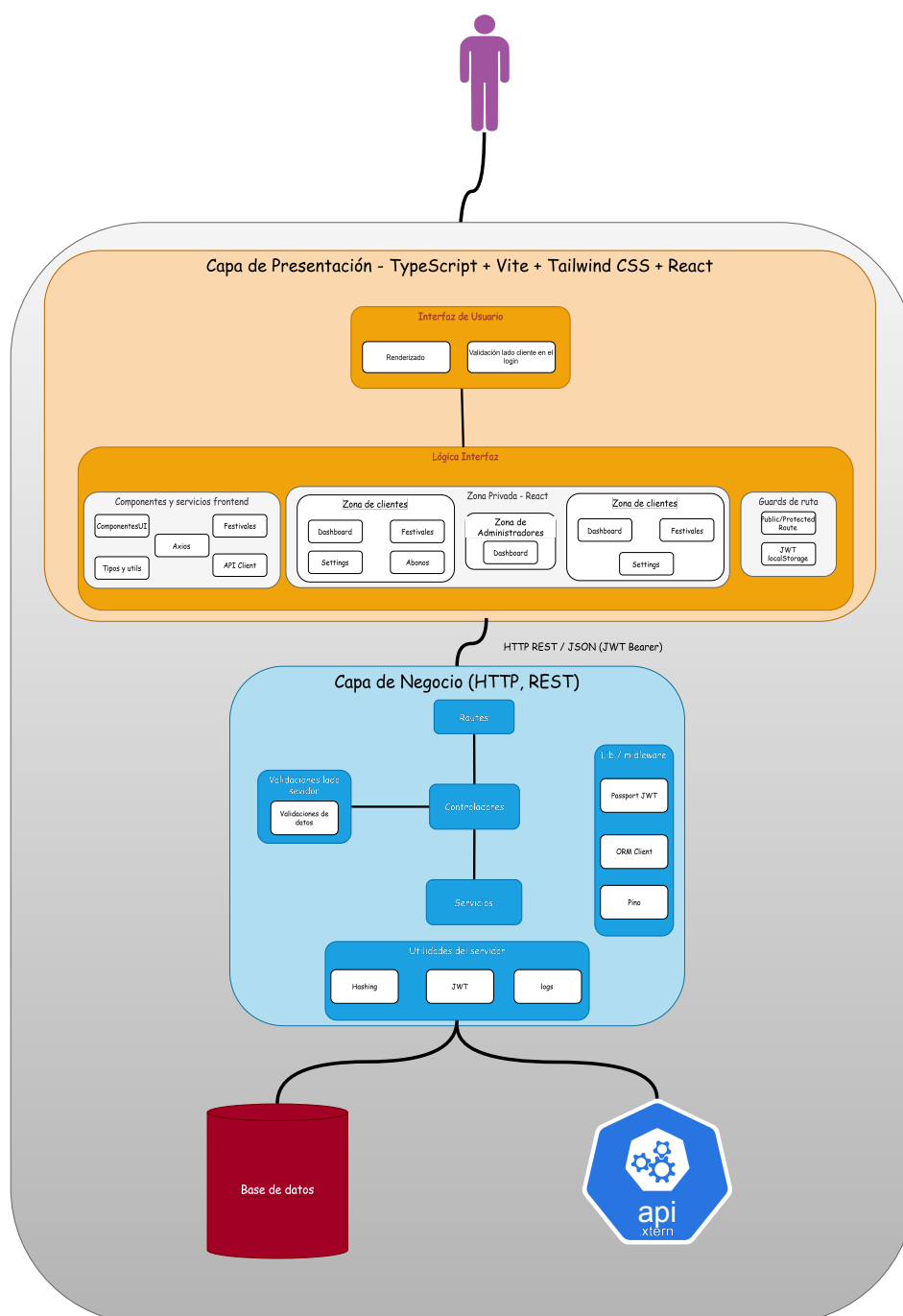


Figura 2.2: Diagrama de la arquitectura logica de alto nivel de la plataforma

2.1.3. Arquitectura física

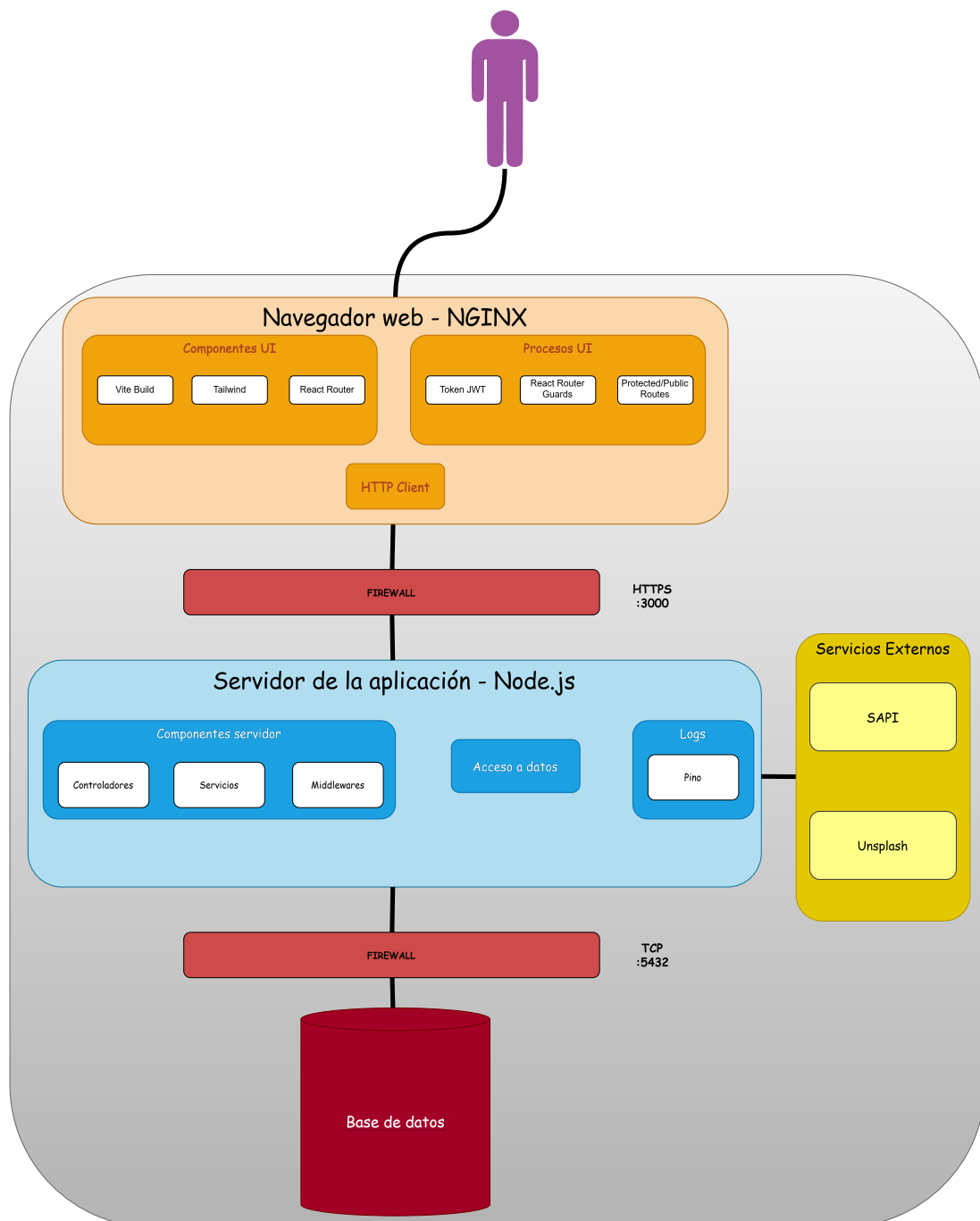


Figura 2.3: Diagrama de la arquitectura física de la plataforma

Capítulo 3

Conclusiones

3.1. Conclusiones generales

El objetivo principal consistía en desarrollar e implementar una aplicación empresarial completa para la gestión y venta de abonos de festivales musicales bajo el nombre comercial **fest.io**, empleando el stack tecnológico basado en Node.js, Express y React.

Tras el proceso de desarrollo se puede afirmar que los objetivos propuestos han sido alcanzados satisfactoriamente:

- Se ha implementado una **API REST** completa con Express y TypeScript, exponiendo endpoints agrupados en seis recursos (`/auth`, `/enterprise`, `/client`, `/festival`, `/artist` y `/logs`). Todos ellos documentados automáticamente mediante la especificación OpenAPI y accesibles a través de SwaggerUI.
- Se ha desarrollado una **capa de presentación** basada en React con Vite y Tailwind CSS que incluye tres zonas privadas, según el rol que ocupen (cliente, empresa gestora y administrador). La navegación está protegida mediante React Router y guards de ruta (`ProtectedRoute` / `PublicRoute`).
- Se ha diseñado e implementado un **modelo de datos relacional** en PostgreSQL, gestionado a través de Prisma ORM, con las entidades necesarias para representar empresas, clientes, festivales, artistas, tipos de abono, transacciones y recargas de pulsera.
- Se ha integrado un **sistema de autenticación y autorización** basado en roles mediante Passport.js con estrategia JWT, bcryptjs para el almacenamiento “seguro” de contraseñas y validación de esquemas con Zod.
- Se han integrado los **servicios web externos** proporcionados por la plataforma API del profesor: la pasarela de pago simulada (`/payments/charge`), el proxy musical hacia la API de Spotify (`/music/search` y `/music/artists/:id`) y la API de imágenes de Unsplash.
- Se han implementado las funcionalidades opcionales de **pulsera cashless**, **histórico de abonos con código QR** generado con `react-qr-code`, **perfil editable** para clientes y empresas, y **observabilidad** del servidor mediante el logger Pino con salida a fichero.

3.2. Líneas de futuro

Como posibles líneas de evolución y mejora del sistema desarrollado, se plantean distintas funcionalidades y ampliaciones que permitirán una mejor experiencia de usuario y aumentar las capacidades de la aplicación:

En primer lugar, se propone una mejora que consiste en la incorporación de un sistema de notificaciones, tanto mediante correo electrónico como dentro de la propia plataforma, que permita informar a clientes y empresas gestoras sobre eventos relevantes (modificaciones de festivales, cancelaciones de festivales, confirmaciones de compra...).

También se plantea la ampliación del módulo de estadísticas y análisis, proporcionando información más detallada sobre ventas, aforo, actividad de usuarios y evolución de festivales.

Por otro lado, resultaría conveniente mejorar el sistema de búsqueda y filtrado de festivales, permitiendo realizar consultas en función de criterios como ubicación, fechas, precios, etc.

En el ámbito de la seguridad, futuras versiones podrían incorporar mecanismos adicionales como autenticación en dos factores, control avanzado de sesiones o limitación de intentos de acceso.

Parte II

Documentación técnica

Capítulo 4

Análisis

4.1. Descripción de actores

ACT-01	Administrador
[Descripción]	▪ Este actor representa al administrador global del sistema. ▪ Es creado directamente mediante el seed de base de datos y no puede registrarse desde la plataforma. ▪ Accede mediante credenciales fijas y su función es supervisar y controlar las empresas gestoras. ▪ Puede iniciar sesión, consultar empresas registradas, verificar empresas, restringir empresas y acceder a los logs del servidor.
[Dependencias]	▪ Empresa gestora ▪ Sistema de autenticación ▪ Sistema de logs
[Comentarios]	El administrador dispone de acceso privilegiado sobre la plataforma y utiliza el endpoint <code>/logs</code> para consultar eventos internos del servidor.

ACT-02	Empresa gestora
[Descripción]	■ Este actor representa a las empresas organizadoras de festivales. ■ Puede registrarse libremente proporcionando razón social, CIF, domicilio, persona de contacto, email, teléfono y contraseña. ■ Tras el registro queda pendiente de validación por parte del administrador. ■ Una vez verificada puede crear, editar y cancelar festivales propios. ■ También puede consultar estadísticas de ventas, editar su perfil y eliminar su cuenta.
[Dependencias]	■ Administrador ■ Festival ■ API Musical — SAPI ■ Sistema de autenticación
[Comentarios]	La edición de festivales únicamente está permitida cuando faltan más de tres días para la fecha de inicio del festival.

ACT-03	Cliente
[Descripción]	■ Este actor representa a los usuarios finales que asisten a festivales. ■ Puede registrarse proporcionando nombre, apellidos, email, DNI, fecha de nacimiento y teléfono. ■ El sistema valida que el usuario sea mayor de edad antes de completar el registro. ■ Una vez autenticado puede consultar festivales, comprar abonos, visualizar su historial de compras, editar su perfil y eliminar su cuenta. ■ Durante la compra se genera un código QR asociado al <code>transactionId</code>.
[Dependencias]	■ Festival ■ Pasarela de pago — SAPI ■ Sistema de autenticación ■ Sistema de generación QR
[Comentarios]	Los datos bancarios se envían únicamente a la pasarela externa y nunca se almacenan completos en la base de datos del sistema.

ACT-04	Pasarela de pago — SAPI
[Descripción]	■ Este actor representa el servicio externo encargado de procesar pagos. ■ El sistema lo utiliza para realizar cobros de abonos mediante el endpoint <code>POST /payments/charge</code>. ■ En caso de aprobación devuelve un <code>transactionId</code>.
[Dependencias]	■ Cliente ■ Sistema de compra de abonos
[Comentarios]	El sistema únicamente almacena el <code>transactionId</code> y los últimos cuatro dígitos de la tarjeta, evitando persistir información sensible completa.

ACT-05	API Musical — SAPI
[Descripción]	▪ Este actor representa un servicio externo utilizado para obtener información musical. ▪ Actúa como proxy hacia la API de Spotify. ▪ El sistema lo consulta para buscar artistas mediante <code>GET /music/search</code>.
[Dependencias]	▪ Empresa gestora ▪ Festival ▪ Spotify API
[Comentarios]	La integración con esta API constituye una funcionalidad adicional implementada en el proyecto para enriquecer la información de los festivales.

4.2. Diagrama de Casos de Uso



Figura 4.1: Diagrama de casos de uso

4.3. Requisitos

En este apartado se describen los requisitos del sistema que se va a desarrollar

CU-01	Registrar empresa gestora
Actor	Empresa gestora
Descripción	La empresa gestora introduce sus datos en el sistema para darse de alta en la plataforma y poder acceder a la aplicación empresarial.
Precondiciones	PRE-1. La empresa gestora no está registrada previamente con el mismo CIF.
Postcondiciones	POST-1. La empresa gestora queda registrada en el sistema. POST-2. La cuenta queda en estado pendiente de verificación.

Flujo normal	FN1. La empresa gestora accede al formulario de registro. FN2. La empresa gestora introduce la razón social, CIF, domicilio social, nombre de contacto, correo electrónico, teléfono y contraseña. FN3. El sistema valida que todos los campos obligatorios estén completos. FN4. El sistema valida que todos los campos estén correctos de acuerdo a las restricciones. FN5. El sistema valida que el correo electrónico no esté ya registrado. FN6. El sistema almacena la empresa gestora. FN7. El sistema permite el acceso a la aplicación empresarial. FN8. El sistema informa de que la cuenta está pendiente de verificación.
Flujo alternativo 1	FA3. Si falta algún dato obligatorio o algún dato es incorrecto, el sistema muestra un mensaje de error y no guarda el registro.
Flujo alternativo 2	FA4. Si el correo electrónico ya existe, el sistema muestra un mensaje de error y no guarda el registro.
Excepciones	E1. La empresa gestora deja campos requeridos sin rellenar. E2. El correo electrónico introducido ya está registrado en el sistema. E3. El formato del CIF, correo electrónico, teléfono o contraseña no es válido.
Prioridad	Alta
Otra info	La cuenta de la empresa gestora podrá acceder a la aplicación empresarial, pero quedará pendiente de verificación hasta que el sistema o un administrador confirme su validez.

Cuadro 4.1: CU-01. Registrar empresa gestora

CU-02	Verificar empresa gestora
Actor	Administrador
Descripción	El administrador valida una empresa gestora registrada para que pueda operar con normalidad en la plataforma.
Precondiciones	PRE-1. El administrador ha iniciado sesión en el sistema. PRE-2. La empresa gestora está registrada en la plataforma.
Postcondiciones	POST-1. La empresa gestora queda verificada o restringida por el usuario.
Flujo normal	FN1. El administrador consulta el listado de empresas gestoras. FN2. El administrador selecciona una empresa pendiente de verificación. FN3. El administrador verifica o restringe la empresa. FN4. El sistema actualiza el estado de la empresa gestora a verificada.
Excepciones	E1. Se produce un error al actualizar el estado de la empresa gestora.
Prioridad	Alta

Otra info	Solo las empresas gestoras verificadas podrán operar con normalidad dentro de la plataforma.
------------------	--

Cuadro 4.2: CU-02. Verificar empresa gestora

4.4. Requisitos Funcionales

ID	RF-01
Descripción	El sistema deberá ofrecer un formulario de registro para empresas gestoras.
Dependencias	CU-01. Registrar empresa gestora
Importancia	Esencial
Prioridad	Alta
Estado	Aprobado
Comentarios	Este requisito permite iniciar el proceso de alta de una empresa gestora en la plataforma.

Cuadro 4.3: RF-01. Formulario de registro de empresa gestora

ID	RF-02
Descripción	El sistema deberá permitir a la empresa gestora introducir: la razón social, CIF, domicilio social, nombre de contacto, correo electrónico, teléfono y contraseña.
Dependencias	CU-01. Registrar empresa gestora
Importancia	Esencial
Prioridad	Alta
Estado	Aprobado
Comentarios	Define la información mínima obligatoria para completar el registro de una empresa gestora.

Cuadro 4.4: RF-02. Captura de datos de empresa gestora

ID	RF-03
Descripción	El sistema deberá validar que todos los campos obligatorios del registro de empresa gestora estén completos.
Dependencias	CU-01. Registrar empresa gestora
Importancia	Esencial
Prioridad	Alta
Estado	Aprobado
Comentarios	Si falta algún campo obligatorio, el sistema no deberá permitir completar el registro.

Cuadro 4.5: RF-03. Validación de campos obligatorios

ID	RF-04
-----------	-------

Descripción	El sistema deberá validar que los datos introducidos en el registro de empresa gestora cumplan las restricciones definidas para cada campo.
Dependencias	CU-01. Registrar empresa gestora
Importancia	Esencial
Prioridad	Alta
Estado	Aprobado
Comentarios	Incluye la validación de formatos como CIF, correo electrónico, teléfono y contraseña.

Cuadro 4.6: RF-04. Validación de formato de datos

ID	RF-05
Descripción	El sistema deberá comprobar que el correo electrónico introducido por la empresa gestora no esté previamente registrado.
Dependencias	CU-01. Registrar empresa gestora
Importancia	Esencial
Prioridad	Alta
Estado	Aprobado
Comentarios	Si el correo ya existe, el sistema deberá impedir el alta y mostrar un mensaje de error.

Cuadro 4.7: RF-05. Comprobación de unicidad del correo electrónico

ID	RF-06
Descripción	El sistema deberá almacenar la empresa gestora cuando todas las validaciones del registro sean correctas.
Dependencias	CU-01. Registrar empresa gestora RF-03, RF-04, RF-05
Importancia	Esencial
Prioridad	Alta
Estado	Aprobado
Comentarios	El alta solo se completará si no existen errores de validación ni conflictos de unicidad.

Cuadro 4.8: RF-06. Almacenamiento de empresa gestora

ID	RF-07
Descripción	El sistema deberá crear la cuenta de la empresa gestora con estado pendiente de verificación.
Dependencias	CU-01. Registrar empresa gestora RF-06
Importancia	Esencial
Prioridad	Alta
Estado	Aprobado

Comentarios	La empresa quedará registrada aunque todavía no haya sido verificada por un administrador.
--------------------	--

Cuadro 4.9: RF-07. Estado inicial pendiente de verificación

ID	RF-08
Descripción	El sistema deberá restringir a la empresa gestora las funcionalidades de la aplicación empresarial mientras esté en proceso de verificación por el administrador o restringida.
Dependencias	CU-01. Registrar empresa gestora RF-06, RF-07, RF-12
Importancia	Importante
Prioridad	Media
Estado	Aprobado
Comentarios	-

Cuadro 4.10: RF-08. Acceso a aplicación empresarial tras registro

ID	RF-09
Descripción	El sistema deberá informar a la empresa gestora de que su cuenta se encuentra pendiente de verificación.
Dependencias	CU-01. Registrar empresa gestora RF-07
Importancia	Importante
Prioridad	Media
Estado	Aprobado
Comentarios	El sistema debe mostrar este estado en la aplicación.

Cuadro 4.11: RF-09. Notificación de cuenta pendiente de verificación

ID	RF-10
Descripción	El sistema deberá permitir al administrador consultar el listado de empresas gestoras registradas.
Dependencias	CU-02. Verificar empresa gestora
Importancia	Esencial
Prioridad	Alta
Estado	Aprobado
Comentarios	Este listado es necesario para que el administrador pueda revisar y modificar el estado de las empresas.

Cuadro 4.12: RF-10. Consulta de empresas gestoras registradas

ID	RF-12
Descripción	El sistema deberá permitir al administrador verificar o restringir una empresa gestora registrada.
Dependencias	CU-02. Verificar empresa gestora RF-10

Importancia	Esencial
Prioridad	Alta
Estado	Aprobado
Comentarios	El administrador puede decidir si la empresa queda habilitada o restringida según corresponda.

Cuadro 4.13: RF-12. Verificación o restricción de empresa gestora

4.5. Requisitos No Funcionales

4.5.1. Restricciones de Tecnología y Desarrollo

- El sistema debe desarrollarse obligatoriamente sobre una stack basada en NodeJS, Express y React.
- Es obligatorio el uso de TypeScript para el tipado de datos.
- La Base de datos debe ser PostgreSQL, utilizando Prisma o algún ORM similar.

4.5.2. Seguridad y Privacidad

- El entorno de todos los componentes deben comunicarse mediante HTTP's.
- El sistema de autenticación y autorización se basa en los roles de passport.
- El sistema debe de guardar las contraseñas de forma segura sin exponerlas.
- Los datos de la tarjeta de crédito del cliente no se almacenarán en la base de datos de la aplicación.
- Los usuarios de la aplicación no deben visualizar todos los datos personales de otros.

4.5.3. Rendimiento y Eficiencia

- El sistema debe tener un rendimiento competitivo, de forma que permita visualizar información solicitada en un tiempo no superior a 5 segundos.
- El proyecto debe ser flexible y escalable, para adaptarse a la demanda a medio plazo o expansiones internacionales.

4.5.4. Disponibilidad y Robustez

- El sistema debe garantizar una disponibilidad del 99,9%.
- La aplicación debe ser resiliente y debe implementar tolerancias a fallos manejándolos de forma adecuada.
- El sistema debe garantizar la integridad de la información de los usuarios y festivales, evitando inconsistencias.

4.5.5. Unidad y Diseño

- El sistema debe contar con una interfaz atractiva y una sólida usabilidad para el usuario final.
- El sistema debe implementar un diseño responsive.
- Se requiere validación de todos los datos recibidos por el usuario (fechas, edad, etc...).

4.5.6. Mantenibilidad

- La API debe ser compatible con la especificación **OpenAPI**, y visualizable mediante **SwaggerUI**.
- El servidor debe contar con un sistema de observabilidad en tiempo de ejecución y registro de *log's*, para consultar operaciones realizadas.

Capítulo 5

Diseño

5.1. Modelo Relacional

EMPRESA (ID, CIF, Persona, Correo, Razon_Social, Domicilio, Telefono, Contraseña, Verificada, Restringida)

FESTIVAL (ID, Nombre, Estado, Ubicación, Aforo, Descripción, F_Inicio, F_Fin, Imagen, Empresa)

FK: Empresa → EMPRESA (ID)

ARTISTA (ID, Imagen, Nombre)

ACTUAR_FESTIVAL (Festival, Artista)

FK: Festival → FESTIVAL (ID)

FK: Artista → ARTISTA (ID)

CLIENTE (ID, Correo, Nombre, Apellidos, DNI, Fecha_Nacimiento, Telefono, Contraseña)

ABONO_CLIENTE (ID, Cliente, Tipo_Abono, Festival, Saldo, Timestamp, Estado)

FK: Cliente → CLIENTE (ID)

FK: Tipo_Abono, Festival → TIPO_ABONO (Nombre, Festival)

TIPO_ABONO (Nombre, Festival, Precio, Stock)

FK: Festival → FESTIVAL (ID)

TRANSACCION (ID, Cantidad, Moneda, Propietario, Timestamp)

REALIZAR_TRANSACCION (Abono, Cliente, Tipo_Abono, Festival, Transaccion, Tipo_Transaccion)

FK: Abono, Cliente, Festival, Tipo_Abono → ABONO_CLIENTE (ID, Cliente, Festival, Tipo_Abono)

5.2. Diccionario de datos

5.2.1. Diccionario de entidades

En esta sección se recogen los diccionarios de datos correspondientes a las entidades principales del sistema.

ID entidad	E6	Nombre	ABONO_CLIENTE					
Definición	ABONO_CLIENTE se define como una entrada digital a un FESTIVAL, adquirida por un CLIENTE, generada tras la TRANSACCION exitosa del pago. Está asociada a un TIPO_ABONO, que determina su precio y categoría. Tras su compra, se le añade un saldo para una pulsera cashless con saldo recargable, accesible mediante un código QR generado tras el pago del abono.							
Reglas								

Atributos	ID	Nombre	Definición	Tipo	Reglas	UNIQUE	DEFAULT	NULL
	E6.1	ID	Identificador único del abono	VARCHAR(255)		Sí	No	No
	E6.2	Saldo	Saldo que contiene el abono	DECIMAL(10, 2)		No	No	No
	E6.3	Timestamp	Marca temporal de adquisición del abono	TIMESTAMP		No	CURRENT_TIMESTAMP	No
	E6.4	Estado	Estado del abono	BOOLEAN		No	TRUE	No

Identificador	ID	Nombre Atributo					
	E6.1	ID					

Figura 5.1: Diccionario de datos de la entidad ABONO_CLIENTE

ID entidad	E3	Nombre	ARTISTA					
Definición	La entidad ARTISTA se define como el intérprete o grupo musical que forma parte del cartel de un FESTIVAL, cuya información puede ser gestionada manualmente por una EMPRESA o enriquecida mediante una API externa (Spotify)							
Reglas								

Atributos	ID	Nombre	Definición	Tipo	Reglas	UNIQUE	DEFAULT	NULL
	E3.1	ID	Identificador unívoco del artista	VARCHAR(255))		Sí	No	No
	E3.2	Nombre	Nombre artístico del artisa	VARCHAR(255))		No	No	No
	E3.3	Imagen	Imagen de perfil del artista	TEXT		No	No	Sí

Identificador	ID	Nombre Atributo						
	E3.1	ID						

Figura 5.2: Diccionario de datos de la entidad ARTISTA

5.2. Diccionario de datos

ID entidad	E5	Nombre	CLIENTE					
Definición	La entidad CLIENTE se define como el usuario final que se registra en la plataforma para adquirir festivales y adquirir abonos							
Reglas	Debe ser mayor de 18 años para poder registrarse el la plataforma							

Atributos	ID	Nombre	Definición	Tipo	Reglas	UNIQUE	DEFAULT	NULL
	E5.1	ID	Identificado único del cliente	VARCHAR(255)		Sí	No	No
	E5.2	Correo	Correo electrónico del usuario	VARCHAR(255)		Sí	No	No
	E5.3	Nombre	Nombre del cliente	VARCHAR(255)		No	No	No
	E5.4	Apellidos	Apellidos del cliente	VARCHAR(255)		No	No	No
	E5.5	DNI	DNI del cliente con formato alfanumérico de 8 número y una letra final	VARCHAR(9)	Formato: 8 números y 1 letra al final (NNNNNNNNL)	No	No	No
	E5.6	Fecha_Nacimiento	Fecha de nacimiento	DATE	Debe ser mayor de edad para poder registrarse y utilizar la plataforma	No	No	No
	E5.7	Telefono	Número de teléfono del cliente	VARCHAR(15)		No	No	No
	E5.8	Contraseña	Contraseña del cliente	VARCHAR(255)		No	No	No

Identificador	ID	Nombre Atributo						
	E5.1	ID						

Figura 5.3: Diccionario de datos de la entidad CLIENTE

ID entidad	E1	Nombre	EMPRESA					
Definición	La entidad EMPRESA se define como la organización legal gestora de festivales que se registra en la aplicación.							
Reglas	Para operar en la plataforma debe estar verificada por el administrador de la empresa. Si la empresa se encuentra restringida, no tendrá acceso a la plataforma.							

Atributos	ID	Nombre	Definición	Tipo	Reglas	UNIQUE	DEFAULT	NULL
	E1.1	ID	Identificador único generado automáticamente con UUID	UUID		Sí	No	No
	E1.2	CIF	Código de identificación fiscal (9 caracteres alfanuméricos, el primero es una letra y los siguientes números)	VARCHAR(9)	El primer carácter del CIF debe ser una letra mayúscula y, seguido de ello, una serie de números	Sí	No	No
	E1.3	Persona	Nombre de la persona de contacto de la empresa	VARCHAR(255)		No	No	No
	E1.4	Correo	Correo electrónico de contacto de la empresa	VARCHAR(255)		Sí	No	No
	E1.5	Razon_Social	Razón social de la empresa	TEXT		No	No	No
	E1.6	Domicilio	Domicilio social de la empresa	TEXT		No	No	No
	E1.7	Telefono	Teléfono de contacto de la empresa	VARCHAR(15)		No	No	No
	E1.8	Contraseña	Contraseña para el acceso a la cuenta	VARCHAR(255)		No	No	No
	E1.9	Verificada	Estado actual de la empresa, está o no verificada por el administrador	BOOLEAN		No	FALSO	No
	E1.10	Restringida	Estado actual de la empresa, está o no restringida por el administrador	BOOLEAN		No	FALSO	No

Identificador	ID	Nombre Atributo
	E1.1	ID

Figura 5.4: Diccionario de datos de la entidad EMPRESA

ID entidad	E2	Nombre	FESTIVAL					
Definición	La entidad FESTIVAL se define como un evento musical organizado y publicado por una EMPRESA gestora							
Reglas								

Atributos	ID	Nombre	Definición	Tipo	Reglas	UNIQUE	DEFAULT	NULL
	E2.1	ID	Identificación única del festival	VARCHAR(255)		Sí	No	No
	E2.2	Nombre	Nombre del festival	VARCHAR(255)		No	No	No
	E2.3	estado_festival	Estado en el que se encuentra el festival (activo o cancelado)	ENUM	ENUM ('Activo', 'Cancelado')	No	Activo	No
	E2.4	Ubicación	Lugar en donde se celebra el evento	TEXT		No	No	No
	E2.5	Aforo	Capacidad máxima de público	INTEGER		No	No	No
	E2.6	Descripcion	Descripción del festival	TEXT		No	No	No
	E2.7	F_inicio	Fecha de inicio del festival	DATE		No	No	No
	E2.8	F_fin	Fecha de inicio del festival	DATE		No	No	No
	E2.9	Imagen	Portada del evento	TEXT		No	IMAGE	Sí

Identificador	ID	Nombre Atributo						
	E2.1	ID						

Figura 5.5: Diccionario de datos de la entidad FESTIVAL

ID entidad	E4	Nombre	TIPO_ABONO					
Definición	La entidad TIPO_ABONO es definida como una categoría de abono que ofrece un FESTIVAL							
Reglas								

Atributos	ID	Nombre	Definición	Tipo	Reglas	UNIQUE	DEFAULT	NULL
	E4.1	Nombre	Nombre del tipo de abono	VARCHAR(255)		No	No	No
	E4.2	Precio	Precio de venta del abono	DECIMAL(10, 2)		No	No	No
	E4.3	Stock	Cantidad de abonos disponibles	INT		No	No	No

Identificador	ID	Nombre Atributo						
	E4.1	Nombre						

Figura 5.6: Diccionario de datos de la entidad TIPO_ABONO

ID entidad	E7	Nombre	TRANSACCION					
Definición	TRANSACCION se define como el registro de una operación económica procesada a través de la pasarela de pago externa, vinculada a la compra de abonos por parte de un CLIENTE							
Reglas								

Atributos	ID	Nombre	Definición	Tipo	Reglas	UNIQUE	DEFAULT	NULL
	E7.1	ID	Identificador unívoco de la transacción	VARCHAR(255)		Sí	No	No
	E7.2	Cantidad	Importe total cobrado	DECIMAL(10, 2)		No	No	No
	E7.3	Moneda	Moneda utilizada	VARCHAR(3)		No	No	No
	E7.4	Propietario	Propietario del abono	VARCHAR(255)		No	No	No
	E7.5	Tiemstamp	Marca de tiempo de la operación	TIMESTAMP		No	CURRENT_TIMESTAMP	No

Identificador	ID	Nombre Atributo
	E7.1	ID

Figura 5.7: Diccionario de datos de la entidad TRANSACCION

5.2.2. Diccionario de relaciones

ID	R1	Nombre	ACTUAR_FESTIVAL					
Definición	La relación ACTUAR_FESTIVAL se define como la relación que vincula un ARTISTA, o varios de ellos, con un FESTIVAL o más, estableciendo su participación en el cartel del festival.							
Reglas								

Entidades	ID	Entidad	Participación	Cardinalidad
	E2	FESTIVAL	1	N
	E3	ARTISTA	0	N

Atributos	ID	Nombre	Definición	Tipo	Reglas	UNIQUE	DEFAULT	NULL
-----------	----	--------	------------	------	--------	--------	---------	------

Figura 5.8: Diccionario de datos de la relación ACTUAR_FESTIVAL

ID	R2	Nombre	REALIZAR_TRANSACCION					
Definición	La relación REALIZAR_TRANSACCION se define como la relación entre ABONO_CLIENTE y TRANSACCION, estableciendo la responsabilidad de pago sobre el CLIENTE a la hora de adquirir el abono en la TRANSACCION							
Reglas								

Entidades	ID	Entidad	Participación	Cardinalidad
	E6	ABONO_CLIENTE	1	N
	E7	TRANSACCION	1	N

Atributos	ID	Nombre	Definición	Tipo	Reglas	UNIQUE	DEFAULT	NULL
	R2.1	Tipo	Tipo de la transacción (compra o recarga)	ENUM	ENUM ('Compra', 'Recarga')	No	No	No

Figura 5.9: Diccionario de datos de la relación REALIZAR_TRANSACCION

5.3. Diseño de Interfaz

El diseño de la aplicación se puede visualizar en este enlace de Figma

5.4. Diagramas de entidad relación

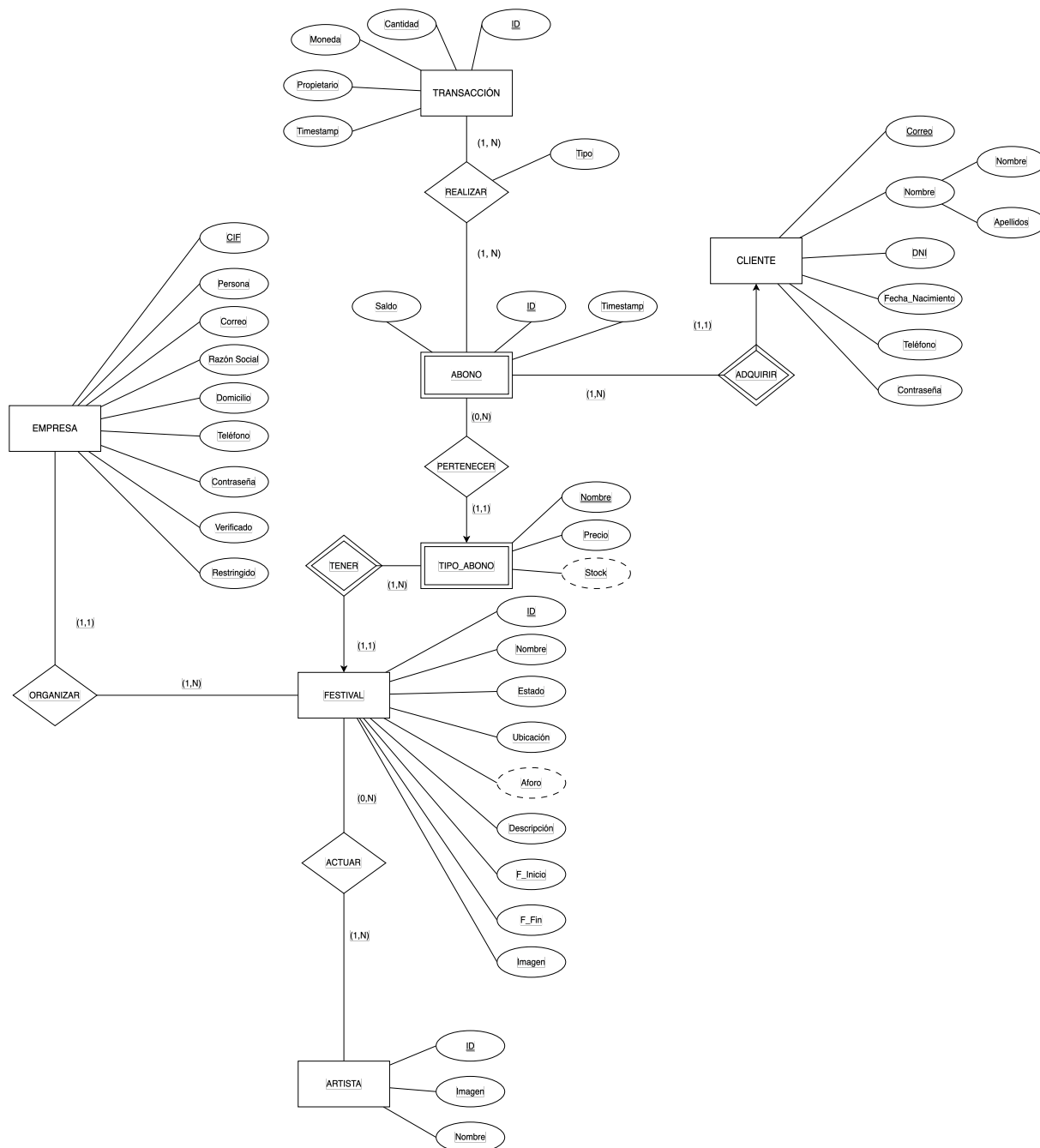


Figura 5.10: Diagrama entidad-relación

5.5. Diagramas de de secuencia

Los diagramas de secuencia describen la secuencia de pasos con que un usuario interacciona con el sistema, y están relacionados con los casos de uso de la etapa de análisis.

A continuación se detallan el diagrama de secuencia del registro de una empresa, y el diagrama de secuencia para verificar una empresa gestora:

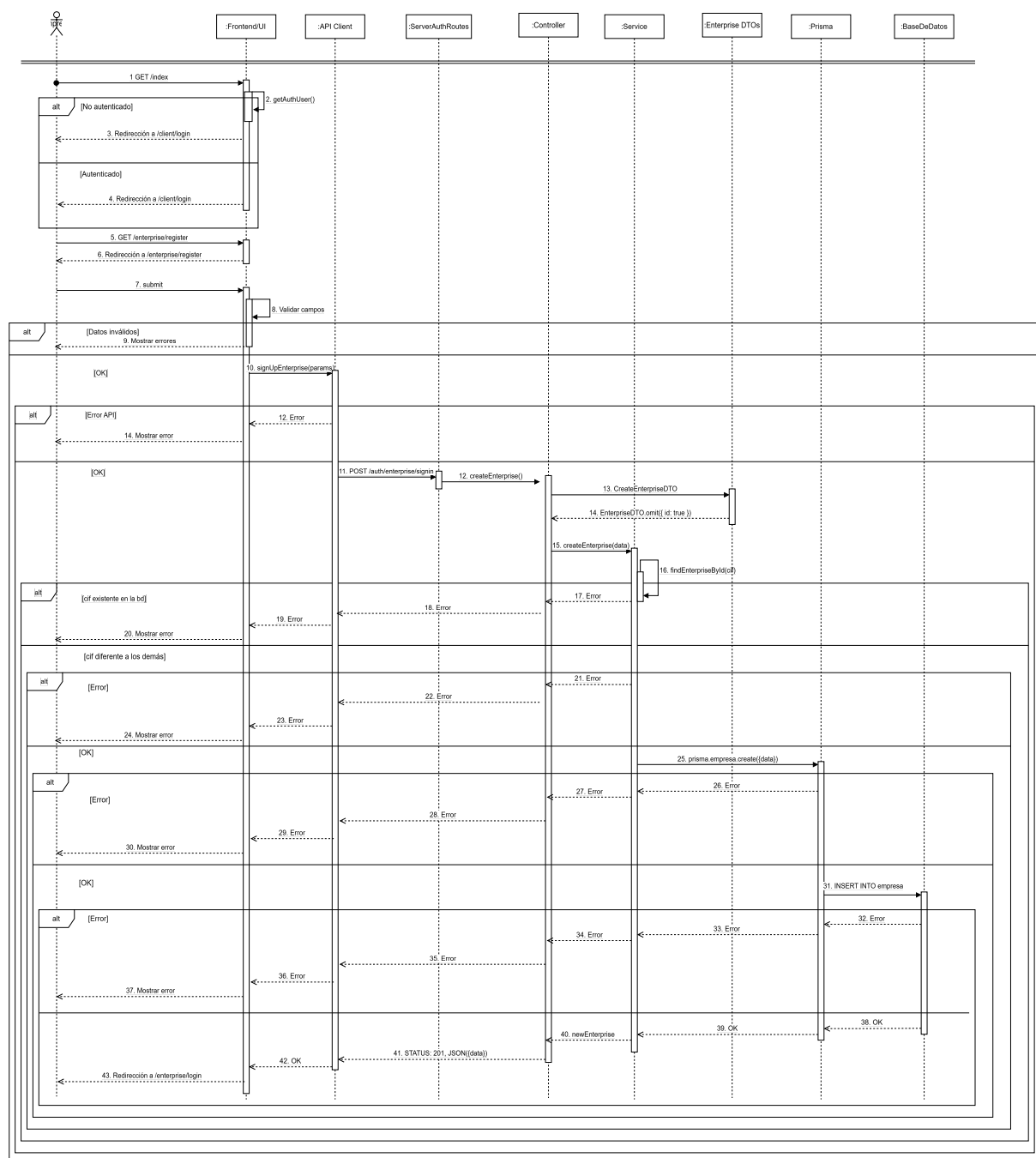


Figura 5.11: Diagrama de secuencia para registrar empresa gestora

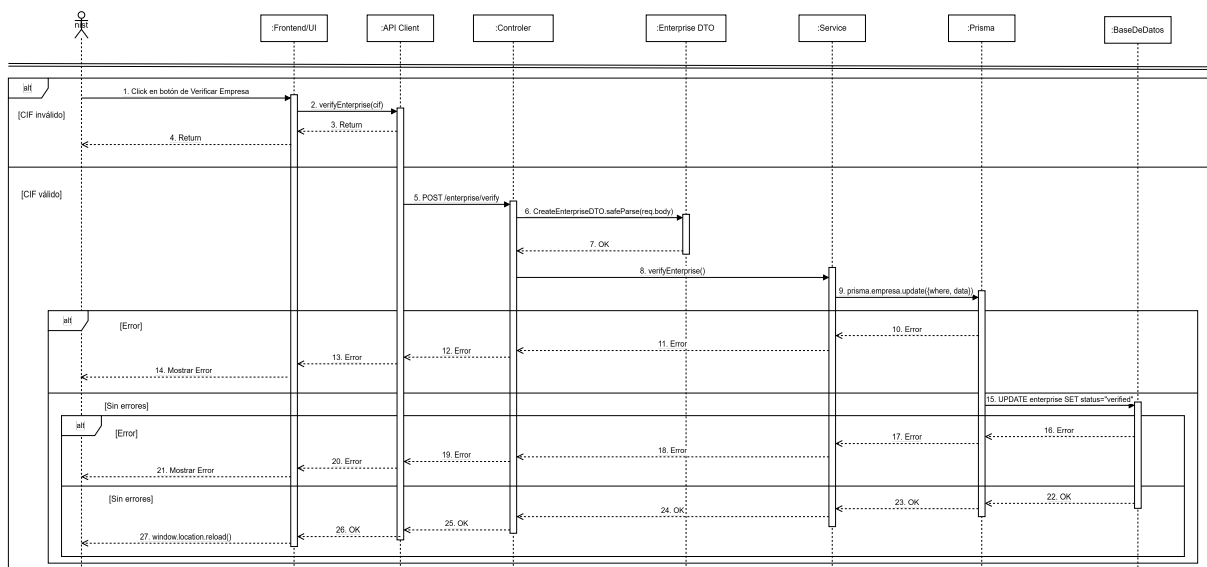


Figura 5.12: Diagrama de secuencia para verificar empresa gestora

Capítulo 6

Implementación

6.1. Paquetes desarrollados

La aplicación se ha dividido en dos paquetes: **Frontend** desarrollado con React y TypeScript, y **Backend** desarrollado con Node.js, Express y TypeScript.

6.2. Servidores de aplicación utilizados

Los servidores utilizados son el **servidor frontend** mediante Vite, utilizado durante el desarrollo de la interfaz React y el **servidor backend** basado en Express.js ejecutado sobre Node.js.

6.3. Servidor de base de datos utilizado

Para la base de datos, se ha optado por usar PostgreSQL para almacenar información de clientes, empresas, festivales, abonos, etc.

6.4. Tecnologías utilizadas

Frontend

- Node.js
- Express
- TypeScript
- TailwindCSS
- JWT para autenticación
- bcrypt para cifrado de contraseñas
- Prisma para acceso a PostgreSQL

Backend

- Node.js
- Express
- TypeScript
- JWT para autenticación
- bcrypt para cifrado de contraseñas
- Prisma para acceso a PostgreSQL

Base de Datos

- PostgreSQL

Servicios externos

- API de pagos: <https://webservices.samuelencinas.dev/api#/Payments>
- API musical: <https://webservices.samuelencinas.dev/api#/Music>
- API de imágenes: <https://api.unsplash.com>

6.5. Herramientas utilizadas

Las herramientas utilizadas durante el desarrollo fueron:

- Visual Studio Code.
- Postman, para pruebas de endpoints.
- Git y GitHub para el control de versiones.
- Postgresql (pgAdmin).
- Docker, para el despliegue y los contenedores.
- DrawIO para realizar los diagramas y arquitecturas del proyecto.
- Microsoft Excel, utilizado con la creación de los diccionarios de datos.
- TailwindCSS, framework para el diseño de la página web.
- Figma, para el diseño de la plataforma.
- Trello, planificación del proyecto.

6.6. Detalles concretos

Para el **sistema de autenticación**, se implementó mediante JWT. Cuando un usuario inicia sesión, el servidor genera un token que se utiliza posteriormente para acceder a rutas protegidas.

Además la **gestión de roles** se puede distinguir por administrador, empresa gestora y cliente.

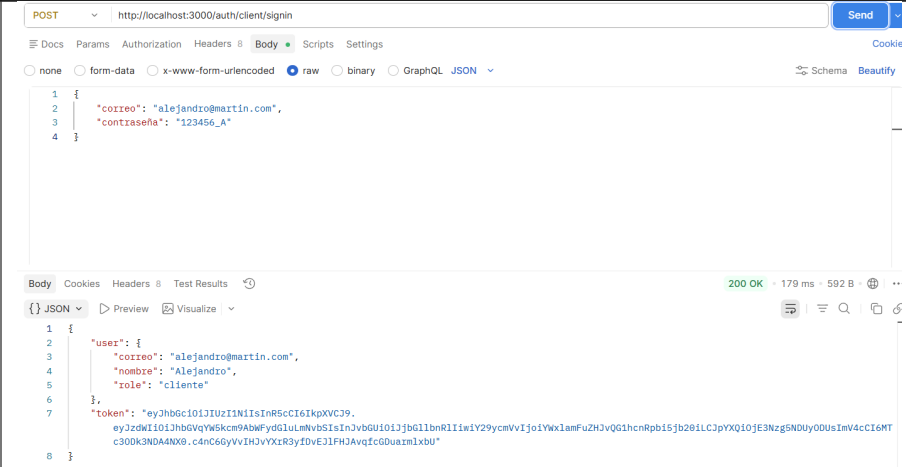
Cada rol dispone de permisos específicos controlados desde middleware en el backend.

El administrador puede consultar **logs** internos del servidor mediante un endpoint protegido (/logs) para supervisión y auditoría básica del sistema.

Capítulo 7

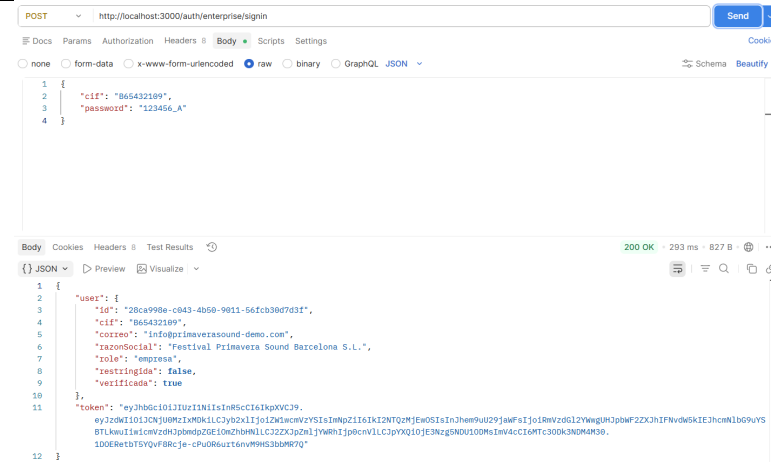
Pruebas

7.1. Pruebas de caja negra

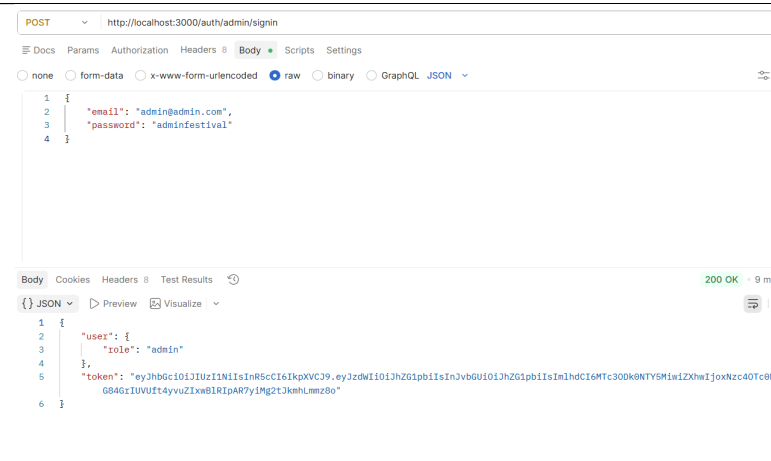
ID	TP-01
Propósito	Verificar inicio de sesión de cliente válido
Prerrequisitos	BD sembrada, backend levantado.
Datos de entrada	POST /auth/client/signin con { "correo": "alejandro@martin.com", "contraseña": "123456_A" }
Resultado esperado	200 OK, devuelve user y token JWT.
Resultado obtenido	

Cuadro 7.1: TP-01. Inicio de sesión de cliente válido

ID	TP-02
Propósito	Verificar inicio de sesión de empresa válido
Prerrequisitos	BD sembrada, backend levantado.
Datos de entrada	POST /auth/enterprise/signin con { "cif": "B65432109", "password": "123456_A" }
Resultado esperado	200 OK, devuelve user y token JWT.

Resultado obtenido	
--------------------	--

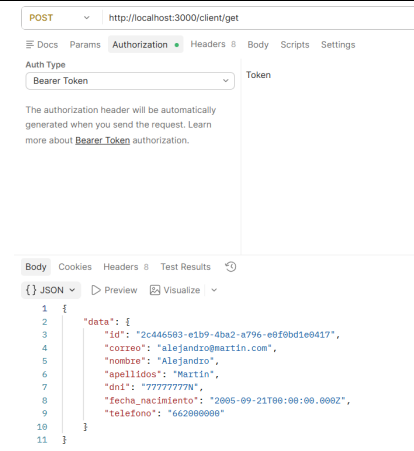
Cuadro 7.2: TP-02. Inicio de sesión de empresa válido

ID	TP-03
Propósito	Verificar inicio de sesión de admin válido
Prerrequisitos	Variables .env con credenciales de admin, backend levantado.
Datos de entrada	POST /auth/admin/signin con { “email”: “admin@admin.com”, “password”: “adminfestival” }
Resultado esperado	200 OK, devuelve token JWT de admin.
Resultado obtenido	

Cuadro 7.3: TP-03. Inicio de sesión de admin válido

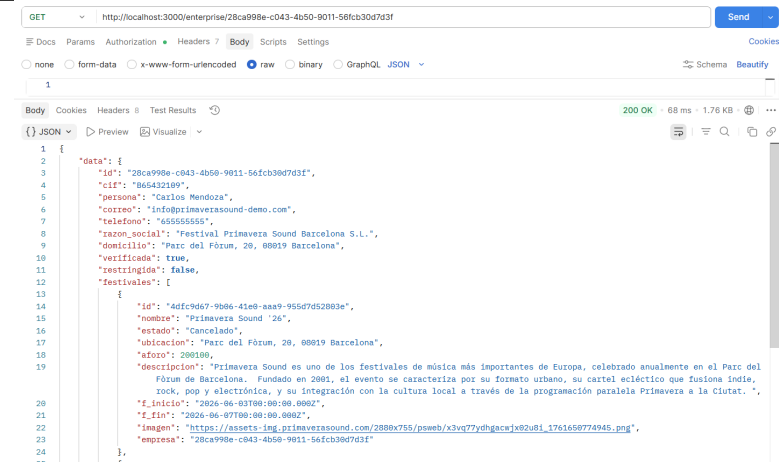
ID	TP-04
Propósito	Comprobar que un cliente puede consultar sus datos
Prerrequisitos	Cliente autenticado con su token.
Datos de entrada	POST /client/get con token de cliente.
Resultado esperado	200 OK, devuelve los datos del cliente autenticado.

7.1. Pruebas de caja negra

Resultado obtenido	
	<pre>1 { 2 "data": { 3 "id": "2c446583-e1b9-4ba2-a796-e8f8bd1e0417", 4 "correo": "alejandromartin.com", 5 "nombre": "Alejandro", 6 "apellidos": "Martin", 7 "dni": "77777777N", 8 "fecha_nacimiento": "2005-09-21T08:08:00.000Z", 9 "telefono": "662989090" 10 } 11 }</pre>

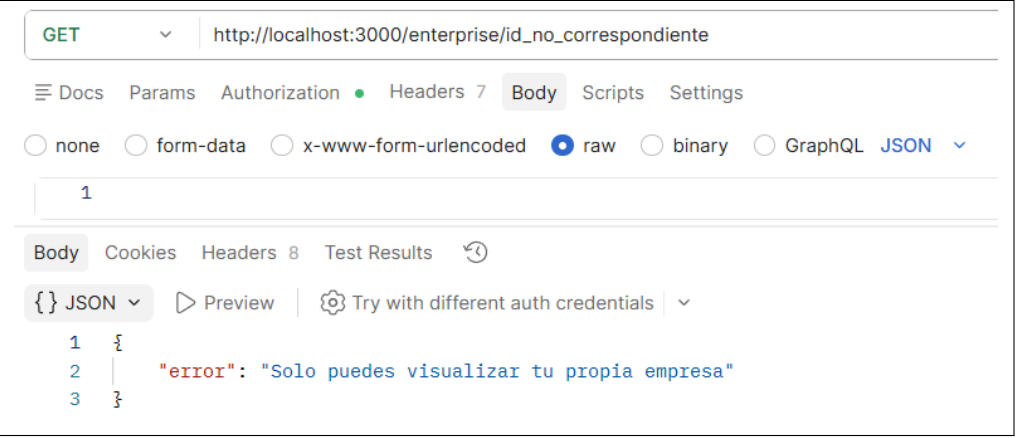
Cuadro 7.4: TP-04. Consulta de datos de cliente

ID	TP-05
Propósito	Comprobar que una empresa solo ve su propia empresa
Prerrequisitos	Token de empresa válido.
Datos de entrada	GET /enterprise/:id usando el id devuelto en su login.
Resultado esperado	200 OK si el id coincide con la empresa del token.

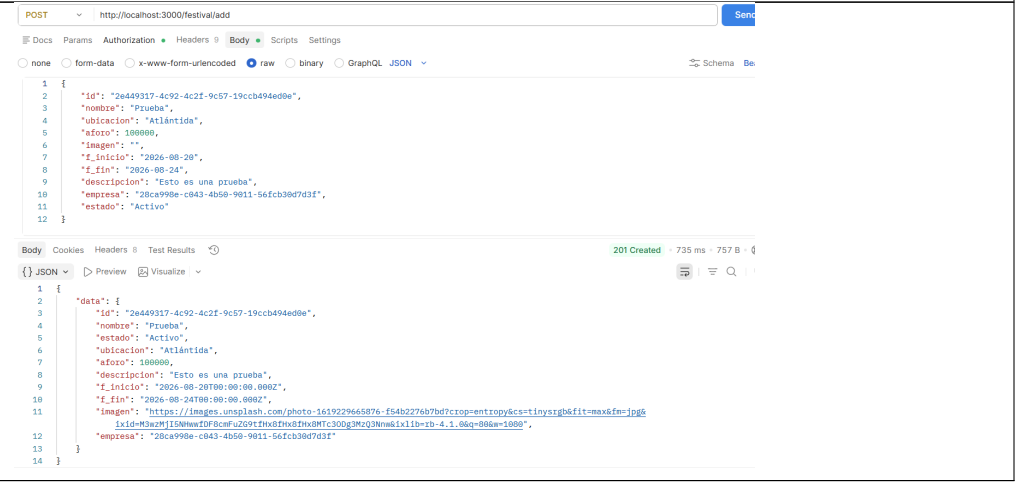
Resultado obtenido	
	<pre>1 { 2 "data": { 3 "id": "28ca998e-c043-4b50-9011-56fcb30d7d3f", 4 "cif": "B66432109", 5 "persona": "Carlos Mendoza", 6 "correo": "info@primaverasound-demo.com", 7 "telefono": "655555555", 8 "razon_social": "Festival Primavera Sound Barcelona S.L.", 9 "domicilio": "Parc del Fòrum, 20, 08019 Barcelona", 10 "verificada": true, 11 "restringida": false, 12 "festivales": [13 { 14 "id": "4dcf9d67-9b66-41e0-aaa9-95d7d52883e", 15 "nombre": "Primavera Sound '26", 16 "estado": "Cancelado", 17 "ubicacion": "Parc del Fòrum, 20, 08019 Barcelona", 18 "aforo": 20000, 19 "descripcion": "Primavera Sound es uno de los festivales de música más importantes de Europa, celebrado anualmente en el Parc del Fòrum de Barcelona. Fundado en 2001, el evento se caracteriza por su formato urbano, su cartel ecléctico que fusiona indie, rock, pop y electrónica, y su integración con la cultura local a través de la programación paralela Primavera a la Ciutat.", 20 "inicio": "2026-06-03T00:00:00.000Z", 21 "fin": "2026-06-07T00:00:00.000Z", 22 "imagen": "https://assets-img.primaverasound.com/288x755/pseeb/x3vq77ydhgackjw92u81_1761650774945.png", 23 "empresa": "28ca998e-c043-4b50-9011-56fcb30d7d3f" 24 } 25] 26 } 27 }</pre>

Cuadro 7.5: TP-05. Visibilidad de empresa propia

ID	TP-06
Propósito	Comprobar acceso no autorizado a empresa ajena
Prerrequisitos	Token de empresa válida de otra empresa.
Datos de entrada	GET /enterprise/:id con un id que no corresponde al token.
Resultado esperado	403 Forbidden.

Resultado obtenido	

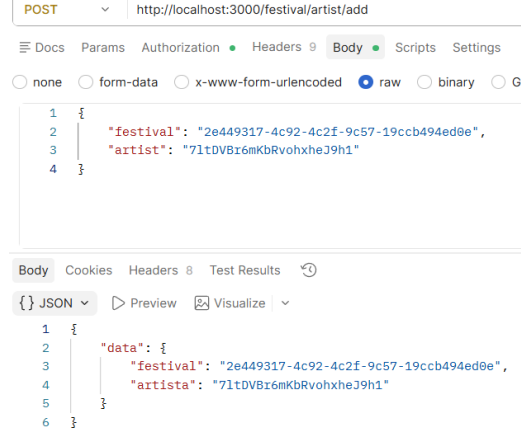
Cuadro 7.6: TP-06. Control de acceso a empresa ajena

ID	TP-07
Propósito	Verificar creación de festival
Prerrequisitos	Token de empresa válida.
Datos de entrada	POST /festival/add con body: id, nombre, estado, ubicacion, f_inicio, f_fin, empresa, descripcion, aforo, imagen.
Resultado esperado	201 Created, devuelve el festival creado.
Resultado obtenido	

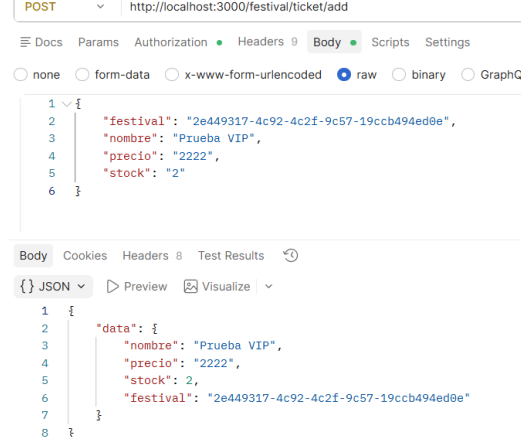
Cuadro 7.7: TP-07. Creación de festival

ID	TP-08
Propósito	Verificar añadir artista a festival
Prerrequisitos	Token de empresa válida, festival existente, artista existente.
Datos de entrada	POST /festival/artist/add con { "festival": "450ec7c9-0c19-47da-9844-7e7451016c72", "artist": "7jy3rLJdDQY210gRLCZ9sD" }.
Resultado esperado	200 OK, devuelve el festival actualizado.

7.1. Pruebas de caja negra

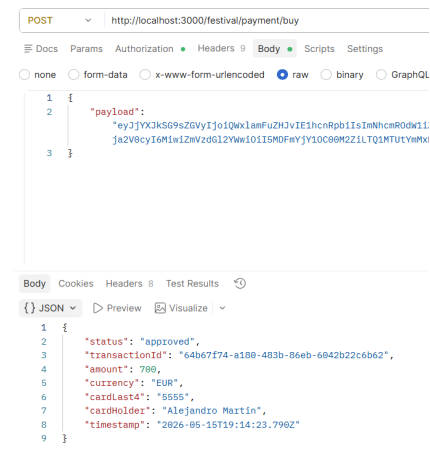
Resultado obtenido	 The screenshot shows a REST client interface. At the top, the method is 'POST' and the URL is 'http://localhost:3000/festival/artist/add'. Below the URL bar, there are tabs for 'Docs', 'Params', 'Authorization', 'Headers', 'Body', 'Scripts', and 'Settings'. The 'Body' tab is selected, and the request body is shown as raw JSON: <pre>{ "festival": "2e449317-4c92-4c2f-9c57-19ccb494ed0e", "artist": "71tDVB6mKbRvohxheJ9h1" }</pre>. Below the request body, there are tabs for 'Body', 'Cookies', 'Headers', 'Test Results', and a refresh icon. The 'Body' tab is selected, and the response body is shown as JSON: <pre>{ "data": { "festival": "2e449317-4c92-4c2f-9c57-19ccb494ed0e", "artista": "71tDVB6mKbRvohxheJ9h1" } }</pre>
--------------------	---

Cuadro 7.8: TP-08. Añadir artista a festival

ID	TP-09
Propósito	Verificar añadir ticket a festival
Prerrequisitos	Token de empresa válida, festival existente.
Datos de entrada	POST /festival/ticket/add con { 'nombre': 'Abono General', 'precio': 300, 'stock': 100, 'festival': '450ec7c9-0c19-47da-9844-7e7451016c72' }.
Resultado esperado	200 OK, devuelve el festival actualizado.
Resultado obtenido	 The screenshot shows a REST client interface. At the top, the method is 'POST' and the URL is 'http://localhost:3000/festival/ticket/add'. Below the URL bar, there are tabs for 'Docs', 'Params', 'Authorization', 'Headers', 'Body', 'Scripts', and 'Settings'. The 'Body' tab is selected, and the request body is shown as raw JSON: <pre>{ "festival": "2e449317-4c92-4c2f-9c57-19ccb494ed0e", "nombre": "Prueba VIP", "precio": "2222", "stock": "2" }</pre>. Below the request body, there are tabs for 'Body', 'Cookies', 'Headers', 'Test Results', and a refresh icon. The 'Body' tab is selected, and the response body is shown as JSON: <pre>{ "data": { "nombre": "Prueba VIP", "precio": "2222", "stock": 2, "festival": "2e449317-4c92-4c2f-9c57-19ccb494ed0e" } }</pre>

Cuadro 7.9: TP-09. Añadir ticket a festival

ID	TP-10
Propósito	Verificar compra de ticket válida
Prerrequisitos	Token de cliente válido, servicio de pago operativo.

Datos de entrada	POST /festival/payment/buy con { "payload": "<base64 de JSON válido con cardHolder, cardNumber, expiryDate, cvv, currency, festival, abono, tickets>" }.
Resultado esperado	200 OK, devuelve el resultado de la operación de pago/compra.
Resultado obtenido	 The screenshot shows a REST client interface with a POST request to http://localhost:3000/festival/payment/buy. The request body is a JSON object with a "payload" field containing a base64-encoded string. The response is a 200 OK status with a JSON body containing transaction details: status (approved), transactionId, amount (799), currency (EUR), cardLast4 (5855), cardHolder (Alejandro Martin), and timestamp. <pre>1 { 2 "payload": "eyJYK3KSG9sZGVyIjo1QWxlamFuZ2h3vIE1hcnRpb1IsImNhcmR0dW1I. 3 js2V8cyI6Miw1ZmVzdG12YmVwI0I1I9MDFayJy1OC88M2Z1LTQ1NTU1YmFwI" 4 } 5 6 7 8 9</pre> <pre>1 { 2 "status": "approved", 3 "transactionId": "64b67f74-a188-483b-86eb-6842b22c6b62", 4 "amount": 799, 5 "currency": "EUR", 6 "cardLast4": "5855", 7 "cardHolder": "Alejandro Martin", 8 "timestamp": "2026-05-15T19:14:23.790Z" 9 }</pre>

Cuadro 7.10: TP-10. Compra de ticket válida

Parte III

Manuales de la Aplicación

Capítulo 8

Manual de Instalación

8.0.1. Requisitos previos

El único requisito para ejecutar la aplicación es tener instalado **Docker Desktop** en el equipo. Docker Desktop incluye tanto el motor de Docker como Docker Compose, la herramienta que orquesta los tres servicios de la aplicación (base de datos, servidor y cliente).

- **Docker Desktop** para macOS, Windows o Linux: <https://www.docker.com/products/docker-desktop>
- Versión mínima recomendada: Docker Desktop 4.x o superior.

8.0.2. Configurar las variables de entorno

En la raíz del proyecto existe el fichero `.env.example` con todas las variables necesarias. Cópialo y renómbralo a `.env`:

Listing 8.1: Copia archivo variables de entorno

```
cp .env.example .env
```

A continuación, abre el fichero `.env` con cualquier editor de texto y rellena los valores reales:

Listing 8.2: Variables de entorno

```
JWT_SECRET=una_cadena_larga_y_aleatoria
ADMIN_EMAIL=admin@tudominio.com
ADMIN_PASSWORD=contrasena_del_administrador
S_ENCINAS_API_USERNAME=usuario_api_encinas
S_ENCINAS_API_PASSWORD=contrasena_api_encinas
UNSPLASH_API_KEY=tu_clave_de_unsplash
```

8.0.3. Levantar la aplicación

Con Docker Desktop en ejecución, abre una terminal en la carpeta raíz del proyecto y ejecuta:

```
docker compose up --build
```

Este comando realiza automáticamente los siguientes pasos:

1. Descarga las imágenes base de Node, nginx y PostgreSQL.
2. Instala las dependencias del servidor y del cliente dentro de los contenedores.
3. Compila el cliente React y lo empaqueta con nginx.
4. Aplica el esquema de la base de datos mediante Prisma.
5. Ejecuta el seed inicial de datos.
6. Arranca los tres servicios coordinados.

El proceso puede tardar varios minutos la primera vez. En ejecuciones posteriores, Docker reutiliza las capas ya construidas y es considerablemente más rápido.

8.0.4. Acceder a la aplicación

Una vez que los tres contenedores estén en marcha, la aplicación es accesible desde el navegador en las siguientes direcciones:

Servicio	URL
Cliente web	http://localhost
API / Servidor	http://localhost:3000
Documentación Swagger	http://localhost:3000/api-docs

8.0.5. Parar la aplicación

Para detener todos los servicios sin borrar los datos:

```
docker compose down
```

Para detener y borrar también los datos de la base de datos (útil para empezar desde cero):

```
docker compose down -v
```

Se incluyen los secretos para el testeo de la práctica en el ordenador del profesor

```
S_ENCINAS_API_USERNAME="grupo15"
S_ENCINAS_API_PASSWORD="martin"
JWT_SECRET="857e6c03fd387f5c197eb29efce3f8c445
30ef0849385668133f0d1de677b54b07dd5d0854cb3ce0315d8bada3bc545f30ec8189a362
ab358c67408f867636b3"
ADMIN_EMAIL="admin@admin.com"
ADMIN_PASSWORD="festivaladmin"
```

UNSPLASH_API_KEY="

ed4ea3b388f4503fa9a5817e2e5250171fd92b3b61ff520ff9f6027cff251a67"

Capítulo 9

Manual de Usuario

Este capítulo final se dedica a desarrollar el manual del usuario final, así como del usuario administrador, si fuese necesario.

9.1. Manual de Usuario

Este manual está dirigido a los usuarios finales de la plataforma: clientes que desean comprar abonos para festivales y gestionar su saldo.

9.1.1. Registro como cliente

1. Accede a `http://localhost`.
2. Haz clic en **Registrarse como cliente** o navega a `/client/signup`.
3. Rellena el formulario con los siguientes datos: nombre y apellidos, DNI, fecha de nacimiento, correo electrónico, teléfono y contraseña.
4. Pulsa **Registrarse**. Si los datos son correctos, serás redirigido al panel de cliente.

9.1.2. Inicio de sesión

1. Navega a `/client/login`.
2. Introduce tu correo electrónico y contraseña.
3. Pulsa **Iniciar sesión**. Si las credenciales son correctas, accederás a tu panel personal.

9.1.3. Panel de cliente

El panel principal muestra todos los festivales en los que tienes abonos activos, agrupados por tipo de abono. Desde aquí puedes:

- Ver el nombre del festival y las fechas.
- Expandir cada festival para ver los abonos comprados.
- Acceder al detalle de cada abono pulsando **Ver detalles**.

9.1.4. Explorar festivales

1. En el menú lateral, accede a la sección **Festivales**.
2. Verás un listado de todos los festivales disponibles con nombre, fechas, ubicación y estado.
3. Puedes filtrar festivales escribiendo en la barra de búsqueda.
4. Los festivales cancelados aparecen marcados visualmente y no permiten compra.

9.1.5. Comprar un abono

1. Desde la lista de festivales, haz clic en el festival que te interese.
2. En la página de detalle verás los tipos de abono disponibles con nombre y precio.
3. Selecciona el tipo de abono deseado y pulsa **Comprar**.
4. Confirma el pago en el modal que aparece.
5. El abono quedará asociado a tu cuenta y aparecerá en tu panel.

9.1.6. Ver tu ticket

1. Desde tu panel o desde un festival, haz clic en **Ver detalles** sobre un abono.
2. Se mostrará tu ticket digital con: tu nombre, el nombre del festival, el tipo de abono (visible como badge), y las fechas del evento.
3. Haz clic en el ticket o en el botón **Voltear** para ver el código QR en el reverso. Este QR es el identificador único del abono y puede usarse para validarlo en el acceso al festival.

9.1.7. Recargar saldo

En la página del ticket, a la derecha del ticket digital encontrarás el panel de saldo:

1. Consulta tu **saldo actual** en euros.
2. Pulsa **Recargar saldo**.
3. En el modal de recarga, introduce el importe que desees añadir.
4. Confirma la operación. El saldo se actualizará inmediatamente.

9.1.8. Ajustes de cuenta

1. Accede a **Ajustes** desde el menú lateral.
2. Puedes actualizar tus datos personales: nombre, apellidos, teléfono y contraseña.
3. Guarda los cambios pulsando el botón correspondiente.
4. Eliminar cuenta: si desees eliminar tu cuenta, haz clic en **Eliminar cuenta** y confirma la acción. Ten en cuenta que esta operación es irreversible y perderás acceso a todos tus abonos.

9.2. Manual de Administración

Este manual está dirigido a dos perfiles: el **administrador del sistema** y los **gestores de empresa** que crean y administran festivales en la plataforma.

9.2.1. Perfil: Administrador del sistema

El administrador tiene acceso a funciones de supervisión y control sobre las empresas registradas en la plataforma.

Inicio de sesión como administrador

1. Navega a `/admin/login`.
2. Introduce las credenciales de administrador definidas en el fichero `.env` (`ADMIN_EMAIL` y `ADMIN_PASSWORD`).
3. Pulsa **Iniciar sesión**.

Panel de administrador

El panel de administrador consta de dos secciones principales:

Lista de empresas. Muestra todas las empresas registradas con su razón social, CIF, correo y estado. Desde aquí el administrador puede:

- **Restringir una empresa:** impide que la empresa pueda operar en la plataforma. Sus festivales dejan de estar disponibles para los clientes.
- **Levantar la restricción:** reactiva una empresa previamente restringida.

Visor de logs del servidor. Muestra en tiempo real los últimos 200 registros de actividad del servidor. Los logs incluyen todas las peticiones HTTP recibidas, errores producidos y eventos relevantes del sistema. Los datos sensibles (tokens de autorización, correos electrónicos completos) aparecen automáticamente enmascarados. Pulsa **Refresh** para actualizar los logs manualmente.

9.2.2. Perfil: Empresa (gestor de festivales)

Las empresas son entidades que crean y gestionan festivales en la plataforma.

Registro de empresa

1. Navega a `/enterprise/signup`.
2. Rellena el formulario con los datos de la empresa: CIF, razón social, nombre de la persona de contacto, correo electrónico, teléfono, domicilio y contraseña.
3. Pulsa **Registrarse**. La cuenta quedará pendiente de verificación por el administrador.

Inicio de sesión como empresa

1. Navega a `/enterprise/login`.
2. Introduce el correo y la contraseña de la empresa.
3. Pulsa **Iniciar sesión**.

Dashboard de empresa

El panel de empresa proporciona una visión completa del rendimiento económico y operativo:

- **Métricas generales:** número total de clientes, abonos vendidos, festivales creados y festivales activos.
- **Datos financieros:** ingresos totales, ingresos por compra de abonos, ingresos por recarga de saldo, saldo pendiente en abonos y ventas desglosadas por tipo de abono con gráficos.
- **Actividad reciente:** lista de las últimas transacciones realizadas en los festivales de la empresa.

Gestión de festivales

Accede a la sección **Festivales** desde el menú lateral.

Crear un festival.

1. Pulsa el botón **Crear festival**.
2. Rellena el formulario: nombre del festival, ubicación, fechas de inicio y fin, aforo máximo, descripción e imagen (opcional, mediante URL de Unsplash).
3. Pulsa **Crear**. El festival aparecerá en tu lista en estado **Activo**.

Gestionar un festival existente. Haz clic en un festival de la lista para acceder a su página de detalle. Desde allí puedes:

- Consultar toda la información del festival.
- **Añadir artistas:** busca artistas por nombre y añádelos al cartel del festival.
- **Eliminar artistas:** elimina artistas del cartel.
- **Gestionar tipos de abono:** define los distintos tipos de entrada disponibles indicando nombre, precio y stock.

Ajustes de empresa

1. Accede a **Ajustes** desde el menú lateral.
2. Puedes actualizar los datos de la empresa: razón social, persona de contacto, teléfono, domicilio y contraseña.
3. Guarda los cambios pulsando el botón correspondiente.

4. Eliminar empresa: si deseas eliminar tu empresa, haz clic en **Eliminar empresa** y confirma la acción. Ten en cuenta que esta operación es irreversible, tus festivales serán cancelados y perderás acceso a la plataforma.

Capítulo 10

Declaración de IAG

Grupo 15, Alejandro Martín & Héctor Torres somos conscientes de que el uso indebido de Inteligencia Artificial Generativa (IAG) puede ser considerado fraude académico y penalizado en consecuencia.

Por ello, declaro de forma responsable que he utilizado herramientas de IAG en la realización de las tareas de formateo de estilo en esta fase de realización del proyecto **FEST.IO**

1. Creación de DTO's
2. JWTStrategy con varias entidades
3. Manejo de edades con zod
4. Uso de Typescript + React Compiler
5. Página 403 (se cambió prácticamente toda)
6. Problema de seguridad en el `middleware`
7. Creación del `seed.ts`
8. Conversión del `seed.ts` a `DML.sql`
9. Restricción temporal en la cancelación del festival.
10. Conversión de SQL a `prisma`.
11. Creación Skeleton modal pasarela de pago.
12. Mejora de código en `prisma`.
13. Skeleton Dashboard empresa.
14. Documentación con **Swagger**.

tal cual puede apreciarse en las siguientes conversaciones:

- <https://chatgpt.com/share/69e891b9-b4c8-83eb-a11f-1bc16d64e7ff>
- <https://claude.ai/share/e86f9518-caee-4ccb-847b-30a6b3f00f8b>
- <https://chatgpt.com/share/69eb589d-3c14-83eb-b10c-b3c7cfd78be4>
- <https://chatgpt.com/share/69ee93dc-addc-83eb-ae94-11462cace8d9>
- <https://chatgpt.com/share/69ee9ee8-28a8-83eb-be5b-2b3712941944>
- <https://chatgpt.com/share/69eea647-509c-83eb-90dd-260dc3eeb090>
- <https://chatgpt.com/share/69f23934-ee38-83eb-a830-b2640ce2d28f>
- <https://chatgpt.com/share/69f34628-3a3c-83eb-80c5-478ea6dbe56f>
- <https://claude.ai/share/afed47eb-6f1b-4c1d-b160-fb156c4d7b92>

- <https://chatgpt.com/share/6a00c2c0-dbd8-83eb-aa60-09e9add3270f>
- <https://claude.ai/share/dc3adc12-70ca-4f09-be77-ecc49cebae55>
- <https://claude.ai/share/fddd4b92-b556-4905-8b54-e263ec5d4af8>
- <https://claude.ai/share/b7fe3ab0-e044-4661-823e-e0cf10d6ab3f>
- La documentación de Swagger se hizo con Claude Code en su totalidad.

Reconozco que los contenidos entregados como resultado de esta actividad han sido creados por mí, asumiendo con ello la plena responsabilidad de su validez y originalidad de cara a su evaluación dentro del proceso de aprendizaje de la asignatura.

Y para que así conste, lo firmo en Segovia y Valencia respectivamente a **16 de Mayo de 2026**

Firma:

Alejandro Martín
Héctor Torres Bolufer

Bibliografía

- [1] Auth0. *jsonwebtoken*. 2026. URL: <https://www.npmjs.com/package/jsonwebtoken> (visitado 16-05-2026).
- [2] Grady Booch, James Rumbaugh e Ivar Jacobson. *The Unified Modeling Language User Guide*. 2.^a ed. Addison-Wesley, 2005.
- [3] Pierre Bourque y Richard E. Fairley. *Guide to the Software Engineering Body of Knowledge: SWEBOOK Version 3.0*. IEEE Computer Society. 2014. URL: <https://www.computer.org/education/bodies-of-knowledge/software-engineering> (visitado 16-05-2026).
- [4] Peter Pin-Shan Chen. «The Entity-Relationship Model—Toward a Unified View of Data». En: *ACM Transactions on Database Systems* 1.1 (1976), págs. 9-36. DOI: 10.1145/320434.320440.
- [5] Docker Inc. *Docker Compose Documentation*. 2026. URL: <https://docs.docker.com/compose/> (visitado 16-05-2026).
- [6] Docker Inc. *Docker Desktop Documentation*. 2026. URL: <https://docs.docker.com/desktop/> (visitado 16-05-2026).
- [7] Express. *Express – Node.js Web Application Framework*. 2026. URL: <https://expressjs.com/> (visitado 16-05-2026).
- [8] Express. *Morgan Middleware*. 2026. URL: <https://expressjs.com/en/resources/middleware/morgan.html> (visitado 16-05-2026).
- [9] ISO/IEC. *ISO/IEC 18004:2024 Information Technology – Automatic Identification and Data Capture Techniques – QR Code Bar Code Symbology Specification*. International Organization for Standardization. 2024. URL: <https://www.iso.org/standard/83389.html> (visitado 16-05-2026).
- [10] ISO/IEC/IEEE. *ISO/IEC/IEEE 29148:2018 Systems and Software Engineering – Life Cycle Processes – Requirements Engineering*. IEEE Standards Association. 2018. URL: <https://standards.ieee.org/standard/29148-2018.html> (visitado 16-05-2026).
- [11] Michael B. Jones, John Bradley y Nat Sakimura. *JSON Web Token (JWT)*. RFC 7519. 2015. URL: <https://www.rfc-editor.org/rfc/rfc7519.html> (visitado 16-05-2026).

- [12] MDN Web Docs. *HTTPS*. 2026. URL: <https://developer.mozilla.org/en-US/docs/Glossary/HTTPS> (visitado 16-05-2026).
- [13] MDN Web Docs. *REST*. 2026. URL: <https://developer.mozilla.org/en-US/docs/Glossary/REST> (visitado 16-05-2026).
- [14] MDN Web Docs. *Working with JSON*. 2026. URL: https://developer.mozilla.org/en-US/docs/Learn_web_development/Core/Scripting/JSON (visitado 16-05-2026).
- [15] Meta Open Source. *React Documentation*. 2026. URL: <https://react.dev/> (visitado 16-05-2026).
- [16] Microsoft. *The TypeScript Handbook*. 2026. URL: <https://www.typescriptlang.org/docs/handbook/intro.html> (visitado 16-05-2026).
- [17] NGINX. *nginx Documentation*. 2026. URL: <https://nginx.org/en/docs/> (visitado 16-05-2026).
- [18] OpenAPI Initiative. *OpenAPI Specification*. 2026. URL: <https://swagger.io/specification/> (visitado 16-05-2026).
- [19] OpenJS Foundation. *Node.js Documentation*. 2026. URL: <https://nodejs.org/api/documentation.html> (visitado 16-05-2026).
- [20] OWASP Foundation. *Authentication Cheat Sheet*. 2026. URL: https://cheatsheetseries.owasp.org/cheatsheets/Authentication_Cheat_Sheet.html (visitado 16-05-2026).
- [21] OWASP Foundation. *Password Storage Cheat Sheet*. 2026. URL: https://cheatsheetseries.owasp.org/cheatsheets/Password_Storage_Cheat_Sheet.html (visitado 16-05-2026).
- [22] OWASP Foundation. *Web Security Testing Guide*. 2026. URL: <https://owasp.org/www-project-web-security-testing-guide/> (visitado 16-05-2026).
- [23] Passport.js. *Passport.js Documentation*. 2026. URL: <https://www.passportjs.org/docs/> (visitado 16-05-2026).
- [24] PCI Security Standards Council. *PCI Data Security Standard*. 2026. URL: <https://www.pcisecuritystandards.org/standards/pci-dss/> (visitado 16-05-2026).
- [25] PostgreSQL Global Development Group. *PostgreSQL Documentation*. 2026. URL: <https://www.postgresql.org/docs/> (visitado 16-05-2026).
- [26] Postman. *Postman Documentation*. 2026. URL: <https://learning.postman.com/docs/introduction/overview> (visitado 16-05-2026).
- [27] Roger S. Pressman. *Ingeniería del software: un enfoque práctico*. McGraw-Hill, 2014.
- [28] Prisma. *Prisma ORM Documentation*. 2026. URL: <https://www.prisma.io/docs> (visitado 16-05-2026).
- [29] React Router. *React Router Installation*. 2026. URL: <https://reactrouter.com/start/declarative/installation> (visitado 16-05-2026).

- [30] Ian Sommerville. *Software Engineering*. 10.^a ed. Pearson, 2016.
- [31] Swagger. *Swagger UI Documentation*. 2026. URL: <https://swagger.io/docs/> (visitado 16-05-2026).
- [32] Tailwind Labs. *Tailwind CSS Documentation*. 2026. URL: <https://tailwindcss.com/docs> (visitado 16-05-2026).
- [33] Unsplash. *Unsplash API Documentation*. 2026. URL: <https://unsplash.com/documentation> (visitado 16-05-2026).
- [34] Vite. *Vite Guide*. 2026. URL: <https://vite.dev/guide/> (visitado 16-05-2026).
- [35] Zod. *Zod Documentation*. 2026. URL: <https://zod.dev/> (visitado 16-05-2026).