

Plataformas de Software Empresariales - Fest.IO

Defensa del Trabajo - Convocatoria Ordinaria

Alejandro Martín Martín Héctor Torres Bolufer

Curso 2025/2026

Índice

1 Análisis

- Comparativo

2 Diseño

- Modelo de datos
- Decisiones de diseño
- Zonas y navegación
- Prototipo

3 Arquitectura

- Componentes
- API REST
- Lógica - Detallada
- Lógica - Alto Nivel
- Docker Compose

- Arquitectura física

4 Implementación

- Backend
- Frontend y funcionalidades opcionales
- Seguridad y autorización

5 Pruebas

- Caja negra

6 Declaración IAG

- Contenido
- Enlaces

7 Conclusión

- Objetivos alcanzados
- Líneas de futuro

Análisis comparativo

Criterio	Eventbrite	DICE / Fever	Ticketmaster	Weeeztix	Entradium	fest.io
Autogestión por organizador	✓ completa	✓ completa	X contrato	✓ completa	✓ completa	✓ completa
Foco festivales	genérico	conciertos	grandes	✓ festivales	genérico	✓ festivales
Pulsera cashless	X	X	partners	✓ nativa	X	✓ opcional
Cartel artistas	X	básica	básica	básica	X	✓ + Spotify
Roles verificación	roles	roles	roles	✓ roles	roles	✓ admin verf.
Pago integrado	✓	✓	✓	✓	✓	✓ SAPI
Entrada QR digital	✓	✓ solo app	✓	✓	✓	✓ react-qr
Comisiones organizador	~6% + fee	fee bajo	X 30% típico	acuerdo	baja/gratis	✓ sin comisión
Observabilidad OpenAPI	✓ analytics	dashboard	✓ avanzado	panel	X básico	✓ Pino+Swagger
Mercado objetivo	global	Europa/UK	global	Europa	España	España

✓ Soportado ~ Parcial X No soportado

Modelo de datos

- **EMPRESA** — CIF, razón social, domicilio, contacto, estado de verificación
- **FESTIVAL** — nombre, ubicación, aforo, fechas, imagen, empresa propietaria
- **ARTISTA** — nombre, imagen (integrado vía Spotify)
- **CLIENTE** — DNI, email, fecha de nacimiento (validación de mayoría de edad)
- **TIPO_ABONO** — nombre, precio, stock (por festival)
- **ABONO_CLIENTE** — saldo cashless, estado, timestamp
- **TRANSACCION** — cantidad, moneda, propietario (solo cardLast4)

Decisiones de diseño

Privacidad financiera

- Solo se persiste el `transactionId` y los últimos 4 dígitos de la tarjeta
- Se codifica el envío de datos del frontend → backend

Reglas de negocio

- La empresa queda **pendiente de verificación** hasta que el administrador la apruebe
- La edición de un festival solo se permite si faltan **más de 3 días** para su inicio

Diseño de interfaz

Tres zonas privadas por rol

- **Administrador** — verificación/restricción de empresas, consulta de logs
- **Empresa gestora** — alta y edición de festivales, estadísticas de ventas
- **Cliente** — compra de abonos, historial con QR, recarga de pulsera

Navegación protegida

- React Router con rutas separadas por rol
- Redirección a página Unauthorized en caso de que el usuario no tenga permisos de lectura.

Diseño de interfaz

Tecnologías de UI

- React + Vite + **Tailwind CSS**
- Diseño **responsive** para móvil y escritorio

Prototipo interactivo en Figma

Ver diseño en Figma

Arquitectura lógica

Patrón cliente-servidor (REST)

- **Frontend:** React + Vite (SPA servida con Nginx)
- **Backend:** Node.js + Express + TypeScript
- **BD:** PostgreSQL gestionada con Prisma ORM
- Comunicación mediante **JSON** sobre HTTP

Servicios externos

- **API — Pasarela de pago:** POST /payments/charge y /refund
- **API — Proxy Spotify:** GET /music/search, /music/artists/:id
- **Unsplash API:** imágenes para festivales

Arquitectura lógica

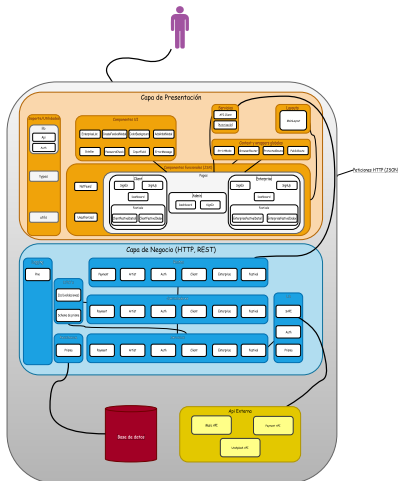
6 recursos expuestos por el backend

- `/auth` Login de los tres roles (cliente, empresa, admin)
- `/enterprise` Registro, verificación y gestión de empresas
- `/client` Registro, perfil y datos del cliente
- `/festival` Festivales, artistas, tipos de abono y pagos
- `/artist` Consulta de artistas (integración Spotify)
- `/logs` Observabilidad del servidor (solo administrador)

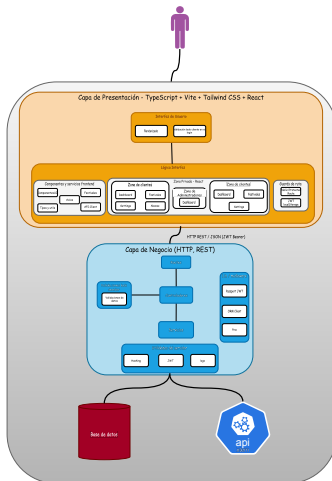
Documentación

API documentada automáticamente con **OpenAPI** y accesible vía **SwaggerUI**

Arquitectura lógica - detallada



Arquitectura lógica - alto nivel

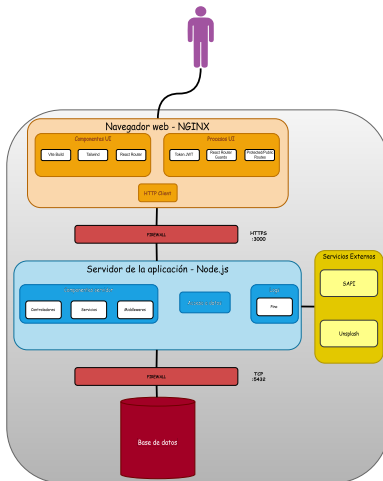


Docker Compose

3 contenedores orquestados

Servicio	Imagen / Build	Puerto
client	React + Nginx (multi-stage build)	:80
server	Express + Prisma (Dockerfile custom)	:3000
postgres	postgres:17-alpine	:5433

Arquitectura física



Implementación — Backend

Node.js + Express + TypeScript

- **Zod** — validación de esquemas en todos los endpoints
- **Passport.js** — autenticación con estrategia JWT
- **bcryptjs** — almacenamiento seguro de contraseñas
- **Prisma ORM** — acceso a PostgreSQL con migraciones declarativas
- **Pino** — logger estructurado con salida a fichero (/logs)
- **SwaggerUI** — documentación OpenAPI autogenerada

Base de datos

- PostgreSQL 17 con seed inicial de datos de prueba
- Migraciones versionadas gestionadas por Prisma

Implementación — Frontend y funcionalidades opcionales

React + Vite + TypeScript

- Tailwind CSS para estilos responsive
- React Router con guards `ProtectedRoute` / `PublicRoute`
- **react-qr-code** — generación de entradas digitales con QR
- Perfil editable para clientes y empresas gestoras

Funcionalidades opcionales implementadas

- **Pulsera cashless** — saldo recargable vinculado al abono
- **Integración Spotify** — cartel de artistas enriquecido con datos musicales
- **Observabilidad** — logs del servidor consultables desde el panel admin
- **Ticket** — componente visual en el cliente

Implementación — seguridad y autorización

Autenticación basada en roles

- Tres roles con permisos independientes: **Admin**, **Empresa gestora**, **Cliente**
- JWT generado en login; enviado en cabecera Authorization
- Middleware valida token y rol antes de cada endpoint protegido

Privacidad y protección de datos

- Contraseñas almacenadas con **bcryptjs**
- Datos bancarios no se persisten: solo transactionId + cardLast4
- Validaciones de entrada con **Zod** en todos los endpoints
- Acceso a datos ajenos devuelve **403 Forbidden**

Pruebas de caja negra

ID	Propósito	Endpoint	Resultado
TP-01	Login de cliente válido	POST /auth/client/signin	✓ 200 OK
TP-02	Login de empresa válido	POST /auth/enterprise/signin	✓ 200 OK
TP-03	Login de administrador válido	POST /auth/admin/signin	✓ 200 OK
TP-04	Cliente consulta sus datos	POST /client/get	✓ 200 OK
TP-05	Empresa ve su propia ficha	GET /enterprise/:id	✓ 200 OK

Pruebas de caja negra

ID	Propósito	Endpoint	Resultado
TP-06	Acceso no autorizado a empresa ajena	GET /enterprise/:id	✓ 403 Forbidden
TP-07	Creación de festival	POST /festival/add	✓ 201 Created
TP-08	Añadir artista a festival (Spotify)	POST /festival/artist/add	✓ 200 OK
TP-09	Añadir tipo de abono a festival	POST /festival/ticket/add	✓ 200 OK
TP-10	Compra de abono con pasarela de pago	POST /festival/payment/buy	✓ 200 OK

Pruebas ejecutadas con **Postman** contra el backend levantado con la BD sembrada.

- 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.

- <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.

Conclusiones — objetivos alcanzados

- API REST completa — 6 recursos documentados con **SwaggerUI / OpenAPI**
- Tres zonas privadas en React con navegación protegida por JWT y rol
- Modelo relacional en PostgreSQL gestionado con **Prisma ORM**
- Autenticación y autorización con **Passport.js + bcrypt + Zod**
- Integración de servicios externos: **pasarela de pago** y **proxy Spotify**
- Funcionalidades opcionales: pulsera cashless, QR, perfil editable, Pino
- Despliegue completo con **Docker Compose** (3 contenedores)

Conclusiones — líneas de futuro

- Sistema de **notificaciones** por email y en plataforma
- **Analítica avanzada**: ventas, aforo y actividad de usuarios
- **2FA** y control avanzado de sesiones

Gracias por la atención

Tiempo de Preguntas