



Vente

Sergio Elías Zúñiga Garrido

Grado de Desarrollo Aplicaciones Web

IES Isidra de Guzmán

Tutor: Borja Bergua

Fecha: 20/11/2025

Índices

Resumen.....	3
Abstract.....	4
Introducción y objetivos del proyecto.....	5
Análisis del problema y justificación de la solución.....	6
Diseño.....	7
Desarrollo: Tecnologías y herramientas utilizadas.....	9
Conclusiones y posibles mejoras futuras.....	11
Manual de instalación.....	12
Manual de usuario.....	15
Bibliografía y recursos consultados.....	19

Resumen

Vente es una aplicación que combina las funciones de una red social y un portal de eventos, concebida para optimizar la organización, promoción y participación en actividades de distinta magnitud, desde grandes espectáculos hasta reuniones privadas. Su objetivo principal es fortalecer la interacción social mediante una plataforma digital que permita a los usuarios crear, descubrir y compartir experiencias tanto presenciales como virtuales.

La aplicación incorpora herramientas de gestión integral de eventos, como el control de aforo, las invitaciones personalizadas, el seguimiento de la participación y el análisis de asistencia. Asimismo, presenta un sistema de cuentas gratuito y de suscripción, complementado con un modelo de progresión basado en la reputación y el mérito, que fomenta la cooperación, la transparencia y la comunidad.

Además, **Vente** integra notificaciones inteligentes y un sistema de recompensas que valora la calidad y constancia de los organizadores, impulsando una cultura de colaboración y reconocimiento dentro de la plataforma. En términos tecnológicos, la aplicación se desarrolla con **Angular** en el frontend, **NestJS** en el backend y **MySQL** como sistema de gestión de bases de datos, garantizando escalabilidad, seguridad y eficiencia operativa.

En conjunto, **Vente** busca consolidarse como una herramienta que promueva la interacción social genuina y el crecimiento colectivo, utilizando la tecnología como un medio para conectar personas y comunidades.

Palabras clave: red social, organización de eventos, colaboración, reputación digital, innovación tecnológica.

Abstract

Vente is an application that integrates the functions of a social network and an event platform, designed to enhance the organization, promotion, and participation in activities of various scales—from large public events to private gatherings. Its primary goal is to strengthen social interaction through a digital environment that enables users to create, discover, and share experiences both in-person and online.

The platform offers comprehensive event management tools, including attendance control, personalized invitations, participation tracking, and analytical features. It implements both free and subscription-based accounts, along with a reputation-based progression system that promotes cooperation, transparency, and community building.

Furthermore, **Vente** includes intelligent notifications and a reward system that acknowledges the quality and consistency of organizers, fostering a collaborative and merit-based ecosystem. From a technological perspective, the application is developed using **Angular** for the frontend, **NestJS** for the backend, and **MySQL** as the database management system, ensuring scalability, security, and operational efficiency.

Overall, **Vente** seeks to establish itself as a platform that promotes genuine social connection and collective growth, leveraging technology as a means to strengthen community engagement.

Keywords: social network, event management, collaboration, digital reputation, technological innovation.

Introducción y objetivos del proyecto

El proyecto **Vente** se plantea como una aplicación híbrida entre una red social y un portal de eventos, cuyo propósito es facilitar la organización, promoción y participación en actividades sociales, culturales o privadas, fomentando la interacción entre personas en entornos tanto físicos como virtuales. A diferencia de las plataformas tradicionales, Vente busca integrar en un solo espacio digital las funcionalidades de descubrimiento, gestión y comunicación, ofreciendo una experiencia social centrada en la comunidad y la reputación de los usuarios.

El objetivo general del proyecto es desarrollar una solución tecnológica escalable, segura y accesible que permita a los usuarios crear, gestionar y compartir eventos, promoviendo la colaboración, la participación responsable y la creación de redes sociales genuinas. Entre los objetivos específicos se destacan:

- Diseñar una plataforma que integre funcionalidades de red social y gestión de eventos en una interfaz moderna e intuitiva.
- Implementar un sistema de niveles, reputación y recompensas que incentive la calidad organizativa y la participación continua.
- Incorporar herramientas de análisis y seguimiento que permitan optimizar la experiencia de organizadores y asistentes.
- Establecer una arquitectura técnica modular que garantice escalabilidad, mantenibilidad y soporte a largo plazo.

En síntesis, Vente aspira a convertirse en un espacio digital que potencie las conexiones humanas reales, utilizando la tecnología como un medio de fortalecimiento social y no como un fin en sí mismo.

Análisis del problema y justificación de la solución

En la actualidad, las plataformas de eventos existentes presentan limitaciones significativas. Muchas se centran únicamente en la venta de entradas o en la difusión de actividades, sin ofrecer herramientas efectivas para la interacción social o la gestión colaborativa. Por otro lado, las redes sociales tradicionales, aunque permiten la difusión de eventos, no están diseñadas para la administración estructurada de aforos, invitaciones o seguimiento de asistencia.

Estas carencias generan dificultades tanto para los organizadores, que carecen de recursos integrales para la planificación, como para los usuarios, que no encuentran espacios seguros ni confiables para descubrir actividades relevantes a sus intereses.

Vente surge como respuesta a esta brecha, proponiendo un modelo híbrido que integra organización, participación y comunidad en una sola aplicación. La solución se fundamenta en tres pilares principales:

1. **Colaboración y mérito:** cualquier usuario puede convertirse en organizador, progresando mediante reputación y valoración verificadas.
2. **Flexibilidad de acceso:** cuenta con modos gratuito y de suscripción, adaptándose tanto a usuarios ocasionales como a profesionales.
3. **Transparencia y confianza:** sistema de validación de asistencia y puntuaciones verificadas que refuerzan la credibilidad de la comunidad.

Esta propuesta busca no solo resolver un problema funcional, sino también fomentar una cultura digital más participativa, ética y cooperativa.

Diseño

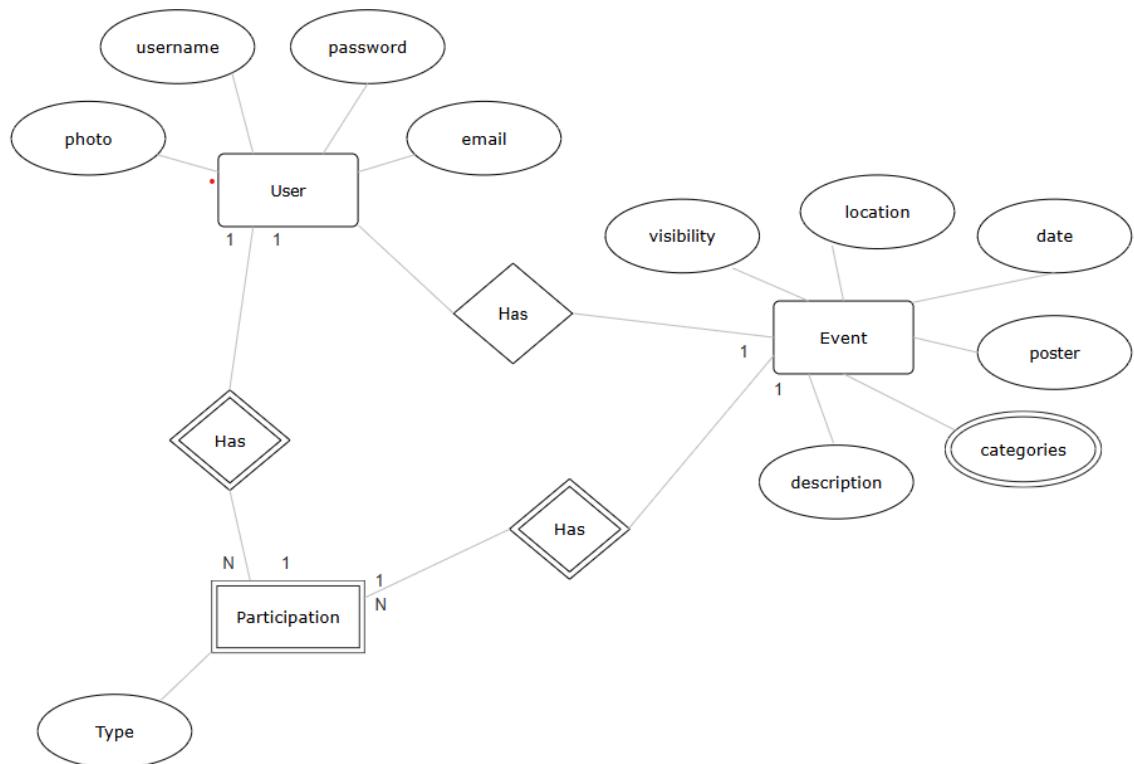


Diagrama entidad-relación (simplificado)

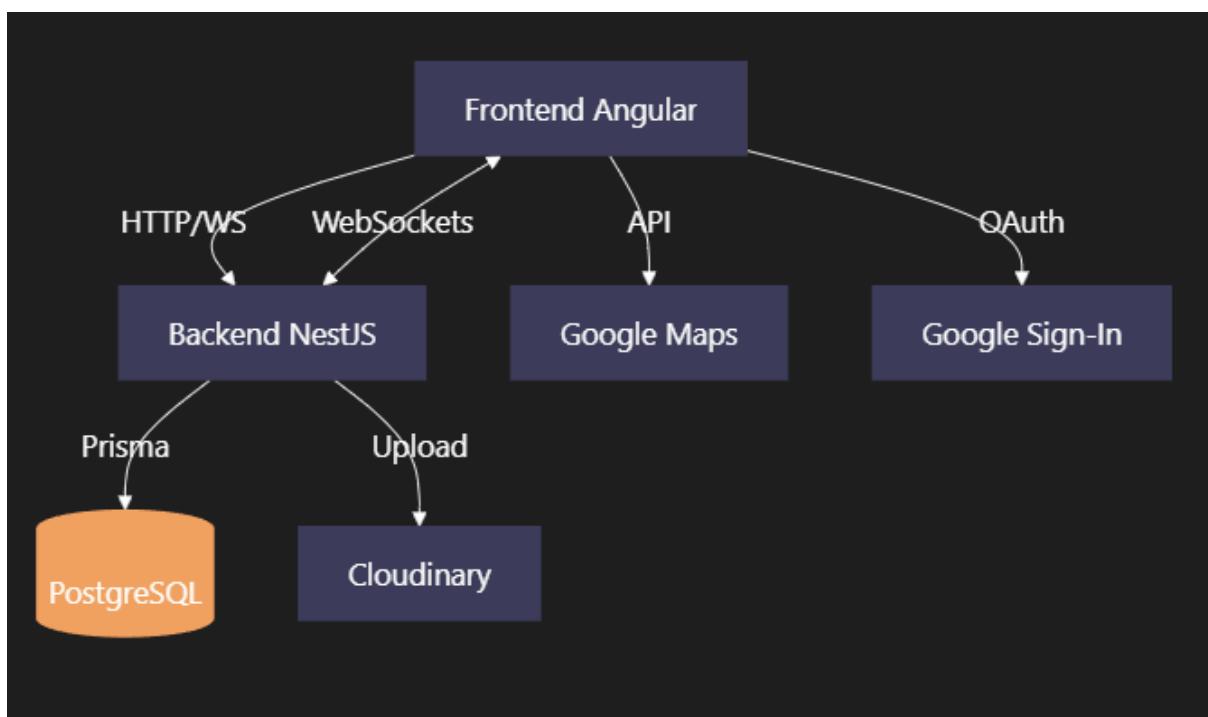


Diagrama arquitectura de la aplicación

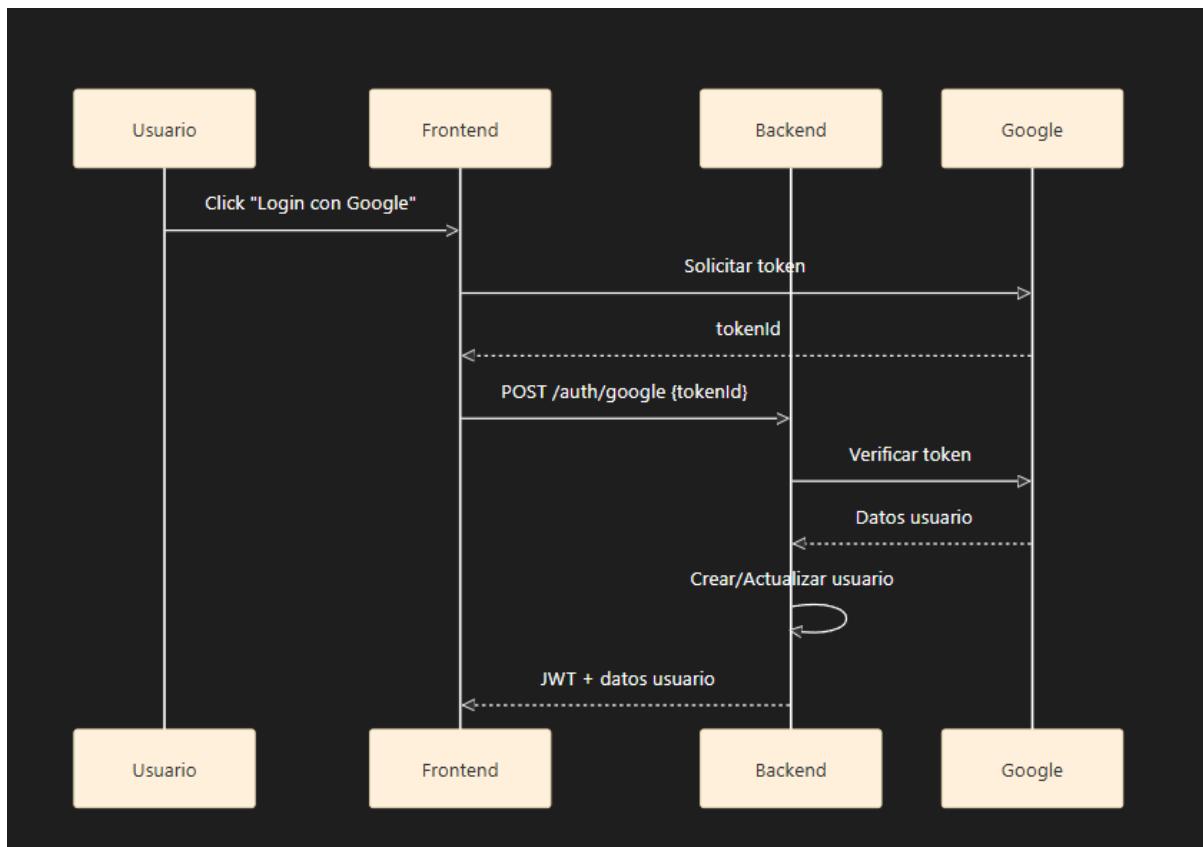
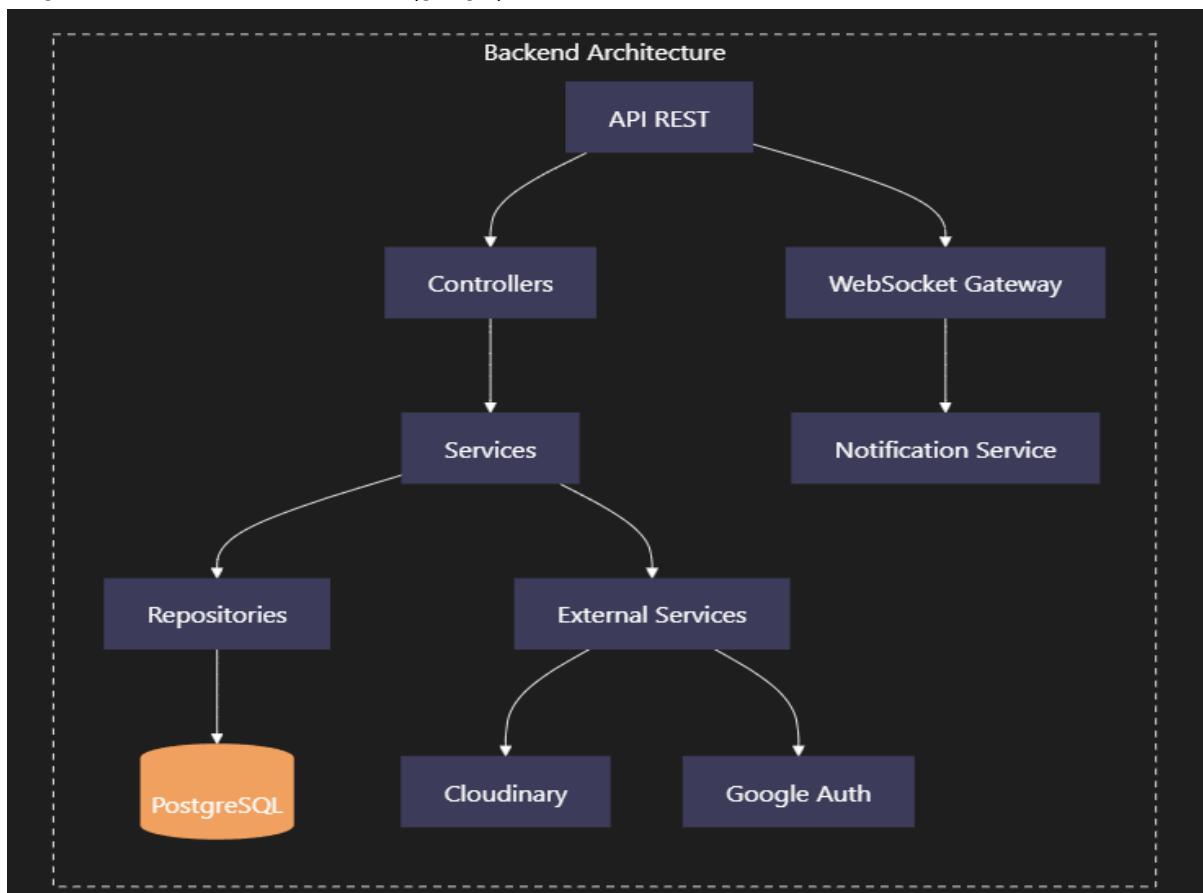


Diagrama secuencia autenticación (google)



Backend arquitectura

Desarrollo: Tecnologías y herramientas utilizadas

Tecnologías principales

El desarrollo de **Vente** se fundamenta en un ecosistema tecnológico moderno que garantiza rendimiento, seguridad y escalabilidad. La aplicación se estructura en dos capas principales —frontend y backend— complementadas por diversas herramientas de soporte que fortalecen la experiencia de desarrollo y la interacción en tiempo real.

Frontend: construido con **Angular 18**, emplea **componentes standalone** y **Angular Router** para una arquitectura modular y de alto rendimiento. Se utiliza **TypeScript** en modo estricto para asegurar la tipificación y la calidad del código. El diseño responsive y la experiencia de usuario se gestionan mediante **TailwindCSS**, mientras que la autenticación y la geolocalización se integran con **Google Identity Services** y **Google Maps API**, respectivamente.

Backend: implementado con **NestJS 10** sobre **Node.js**, utiliza **PostgreSQL** como sistema de gestión de base de datos relacional, administrado mediante **Prisma ORM**. Incluye integración con **Cloudinary** para el almacenamiento externo de imágenes, autenticación mediante **JWT** con soporte de **Google Sign-In**, y tareas programadas mediante **schedulers**. Además, incorpora **WebSockets (Socket.IO)** para el envío y recepción de notificaciones en tiempo real, mejorando la comunicación bidireccional entre cliente y servidor.

El sistema de notificaciones y actualizaciones dinámicas se gestiona a través de **listeners** y **EventEmitters**, los cuales permiten reaccionar a eventos específicos dentro del servidor, como cambios en la participación, nuevas publicaciones o recordatorios de eventos. Esta estructura facilita la propagación inmediata de información relevante a los usuarios conectados, optimizando la interactividad y la inmediatez de la aplicación.

Otras herramientas: se utilizan **RxJS** para la programación reactiva en el frontend, **ngx-toastr** para la gestión de notificaciones visuales, y **Docker** para la contenedorización del entorno de desarrollo y despliegue. Este conjunto tecnológico favorece la modularidad, la reutilización de componentes y la implementación de buenas prácticas de arquitectura limpia.

En su conjunto, estas tecnologías permiten la construcción de una plataforma moderna, segura y escalable, sustentada en principios de desarrollo ágil y orientada a la experiencia del usuario.

Descripción de la arquitectura

La arquitectura de **Vente** se organiza en dos capas principales: **frontend** y **backend**, estructuradas conforme a estándares de desarrollo profesional y documentadas mediante guías técnicas (AGENTS.md). Esta separación de responsabilidades facilita la mantenibilidad, la escalabilidad y la colaboración entre equipos de desarrollo.

Frontend (Angular)

El cliente web se basa en una arquitectura de **componentes independientes y rutas modulares**, siguiendo las recomendaciones oficiales de Angular para el uso de bootstrapApplication y providers globales.

- **Capa de presentación:** compuesta por componentes reutilizables (header, navbar, map, event-card) y páginas funcionales (landing, events, add-event, explore, dashboard).
- **Capa central (core):** gestionada por servicios (ApiService, AuthService, EventsService, GeolocationService, ThemeService), interceptores (auth.interceptor.ts) que manejan autenticación, peticiones HTTP y estado global, guards para las rutas e interfaces usadas.
- **Estilo y experiencia de usuario:** implementados mediante **TailwindCSS**, con diseño adaptable, modo oscuro y notificaciones contextuales.

El uso de **RxJS** y **observables** permite manejar flujos de datos asíncronos, mejorando la respuesta en tiempo real a cambios en la interfaz, como nuevas notificaciones o actualizaciones de eventos emitidas desde el backend.

Backend (NestJS)

El servidor adopta una **arquitectura modular basada en dominios**, que separa las responsabilidades en módulos (auth, user, event, participation, core, cloudinary).

- **Capa de acceso a datos:** gestionada con **Prisma ORM**, que administra las entidades principales (usuarios, eventos, participaciones) bajo un enfoque orientado a repositorios.
- **Capa de lógica de negocio:** desarrollada mediante servicios especializados que procesan las reglas de dominio, incluyendo el manejo de invitaciones cifradas, validaciones con class-validator, generación de tokens JWT y comunicación interna mediante **EventEmitters**.
- **Capa de tiempo real:** a través de **WebSockets (Socket.IO)**, el backend establece canales persistentes para emitir y recibir notificaciones instantáneas, como confirmaciones de asistencia, mensajes entre usuarios o cambios en el estado de un evento. Los **listeners** asociados a estos canales permiten capturar eventos del sistema y transmitirlos de forma inmediata a los clientes conectados.
- **Capa de presentación (API REST):** conformada por controladores enrutados bajo /api, con filtros globales, validaciones y manejo de errores estandarizados.

La comunicación entre frontend y backend se establece mediante **endpoints RESTful seguros** y **canales WebSocket**, garantizando consistencia en los formatos de respuesta, sincronización en tiempo real y una experiencia de usuario fluida y actualizada.

Conclusiones y posibles mejoras futuras

El desarrollo de **Vente** demuestra la viabilidad técnica y conceptual de una plataforma social orientada a la organización de eventos, que integra la funcionalidad de red social con herramientas de gestión profesional. La arquitectura planteada, sustentada en tecnologías modernas como Angular y NestJS, permite un alto grado de escalabilidad, mantenibilidad y adaptabilidad a nuevas necesidades del usuario.

A nivel funcional, la aplicación cumple con los objetivos de ofrecer una experiencia completa: creación de eventos, gestión de asistencia, comunicación entre usuarios y control de reputación. Sin embargo, se identifican líneas claras de mejora y evolución futura:

- **Implementación avanzada de suscripciones:** habilitar planes diferenciados con funcionalidades exclusivas, como estadísticas detalladas, monetización de eventos, campañas promocionales internas y visibilidad prioritaria en el algoritmo de recomendación.
- **Sistema de recompensas dinámico:** incorporar mecanismos gamificados de reconocimiento basados en constancia, valoraciones positivas y participación comunitaria. Estos podrían traducirse en beneficios tangibles dentro de la plataforma (mayor visibilidad, descuentos o ampliación de límites de evento).
- **Expansión de soporte a eventos híbridos:** integración más profunda con servicios de streaming para fortalecer el componente virtual.

En conclusión, **Vente** no solo propone una solución tecnológica a la organización de eventos, sino que establece una nueva forma de interacción social basada en la colaboración, la confianza y la meritocracia digital, proyectándose como una herramienta relevante para la gestión de comunidades en el contexto contemporáneo.

Manual de instalación

→ *git clone https://github.com/sezgox/venteweb*

Backend y base de datos

Requisitos

1. Node.js ≥ v20.0.0
2. npm ≥ v10.0.0
3. Docker ≥ v24.0.0
4. Docker Compose ≥ 2.20.0
5. Prisma Cli ≥ v5.0.0

(Abrir repositorio en terminal)

Moverse a directorio backend →

cd venteweb

Instalación de dependencias →

npm install

Creación fichero .env (cambiar por tus api keys) →

cat <<EOF > .env

=====

DATABASE CONFIGURATION

=====

DATABASE_URL="postgresql://postgres:postgres@localhost:5432/vente?schema=public"

=====

JWT CONFIGURATION

=====

JWT_SECRET="super_secret_jwt_key"

JWT_EXPIRES="1d"

```
JWT_ISSUER="vente-app"
JWT_AUDIENCE="vente-users"

# =====
# GOOGLE AUTHENTICATION
# =====

GOOGLE_CLIENT_ID="your_google_client_id.apps.googleusercontent.com"

# =====
# CLOUDINARY CONFIGURATION
# =====

CLOUDINARY_NAME="your_cloud_name"
CLOUDINARY_API_KEY="your_cloudinary_api_key"
CLOUDINARY_API_SECRET="your_cloudinary_api_secret"

# =====
# EVENT INVITATION CONFIGURATION
# =====

EVENT_ENCRYPTION_KEY="your_32_characters_encryption_key"
EVENT_INVITATION_EXPIRES_IN="2h"

# =====
# SERVER CONFIGURATION
# =====

PORT=3000
```

EOF

Levantar contenedor docker con PostgreSQL →
docker-compose up -d

Inicializar base de datos con Prisma →
npx prisma generate
npx prisma migrate dev --name init

Ejecución del servidor de NestJS →
npm run start:dev

Frontend

Requisitos

1. Node.js ≥ v20.0.0
2. npm ≥ v10.0.0
3. Angular Cli ≥ v18.0.0

(Abrir repositorio en terminal)

Moverse a directorio frontend →

cd ventewebf

Instalación de dependencias →

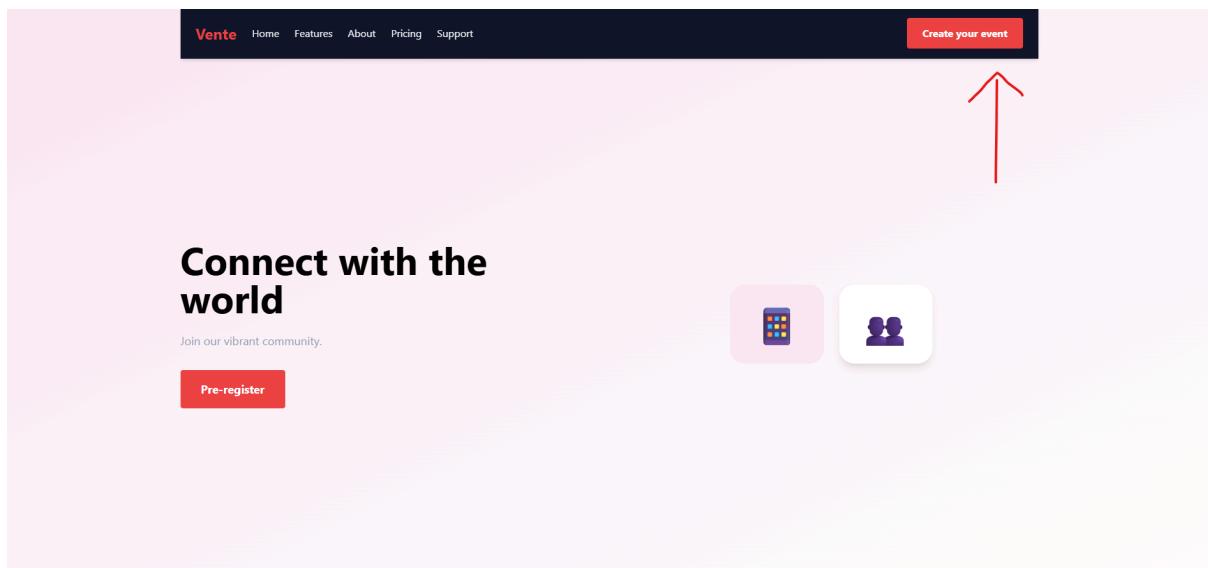
npm install

Ejecución del servidor de desarrollo →

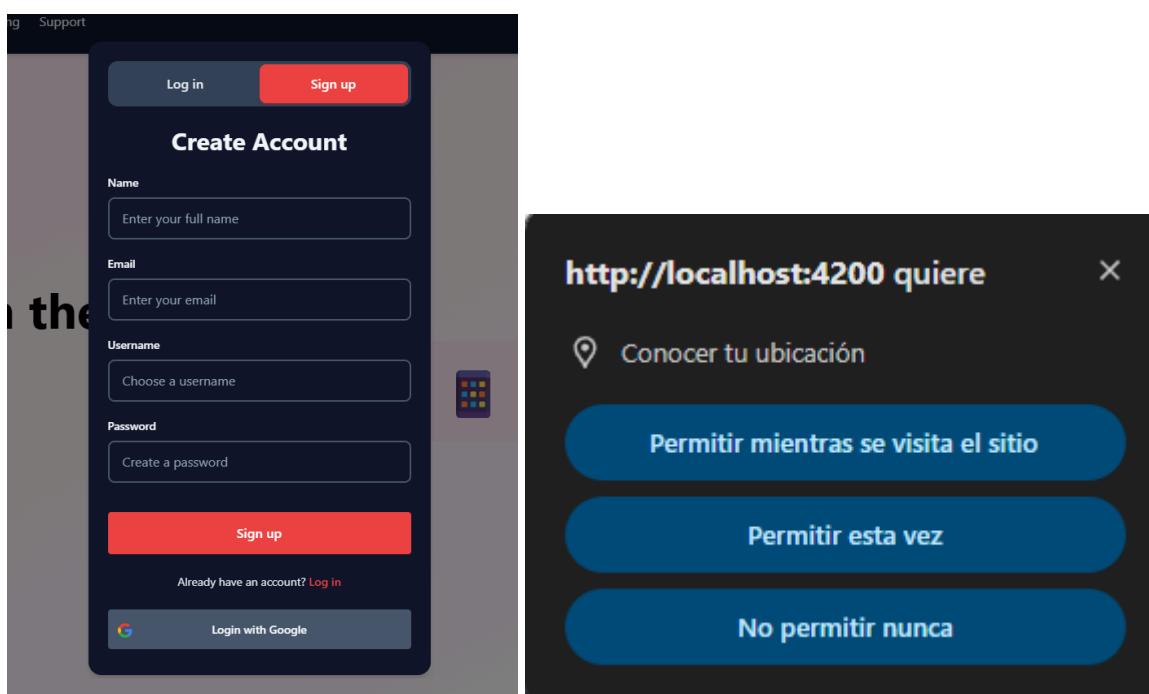
ng s

Manual de usuario

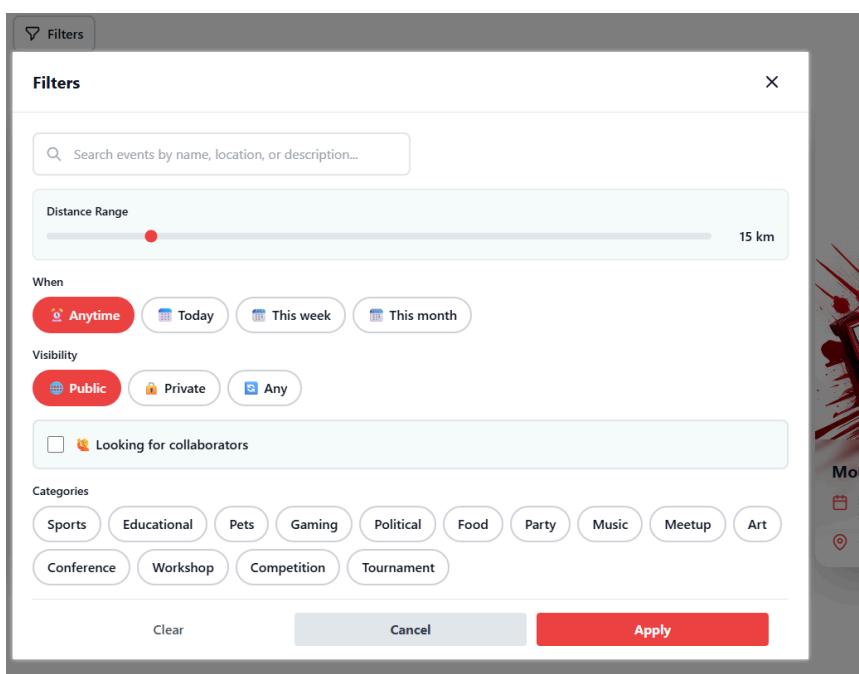
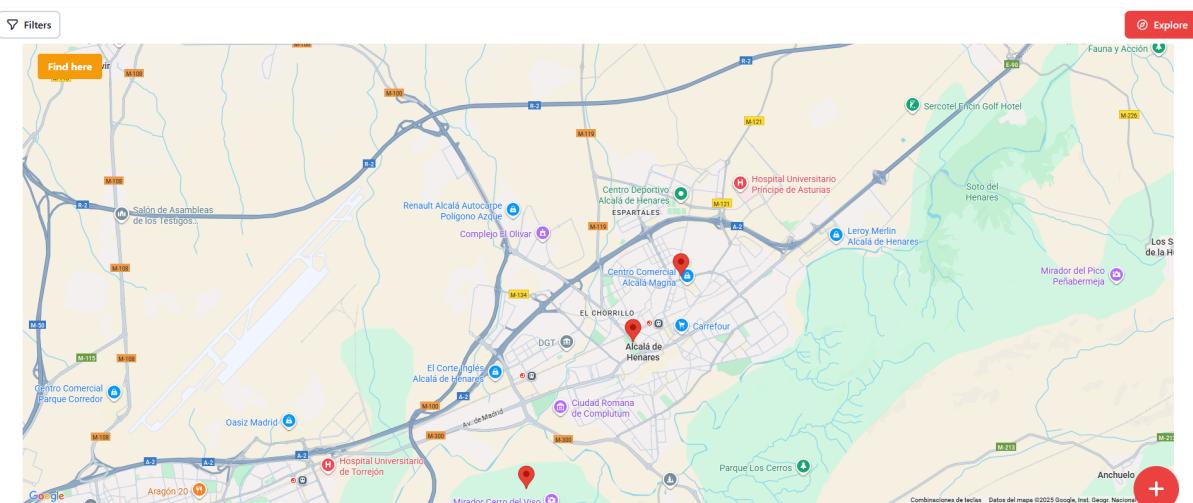
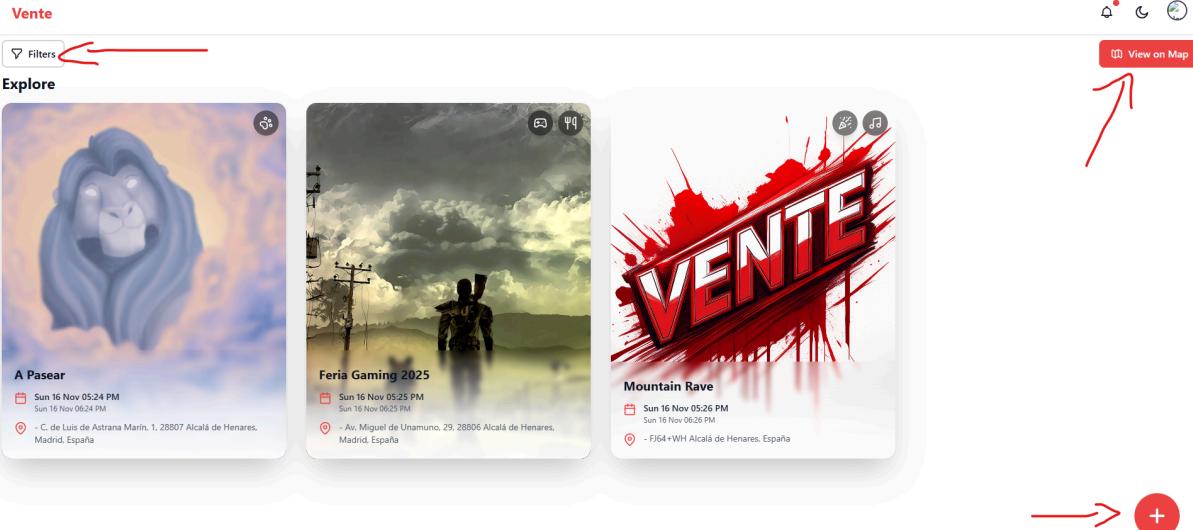
Al entrar a la app, aparecerás en la *Landing Page*.



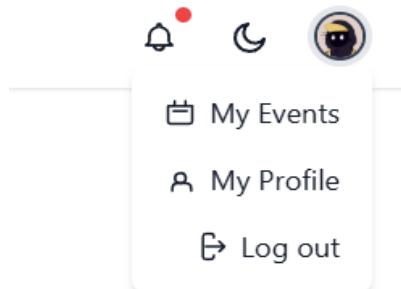
Deberás hacer click en *Create your event*, donde aparecerá un formulario de Login/Register si no has iniciado sesión antes y una vez dentro, deberás dar permiso de ubicación para poder explorar y crear eventos en tu zona:



Entonces, verás la feed con los eventos que hay en tu zona y podrás cambiar la vista al mapa para poder verlos de una manera distinta. Tendrás acceso a filtros, ver en mapa y un botón visible (en todas las vistas) para crear tu propio evento.



A parte de estas secciones principales, existen otras dos vistas con mucha importancia para el usuario:



Dashboard y Profile

Desde el *Dashboard*, podrás acceder a tus eventos, a tus participaciones y también tener el listado de invitaciones que has recibido y de tus peticiones para colaborar en otros eventos

The image displays the 'Dashboard' and 'Profile' sections of the application. The 'Dashboard' part shows 'My Events' and 'My Participations' sections with corresponding backgrounds. Below them are 'My Invitations' and 'My Requests' tabs. The 'Profile' section shows the user's events, including 'Upcoming (3)' events like 'Mountain Rave', 'Feria Gaming 2025', and 'A Pasear'. A red '+' button is located in the bottom right corner of the profile section.

Desde tu *Profile*, tendrás una vista pública de tu perfil, también con tus eventos y participaciones, pero en caso de ver otros perfiles, no tendrás acceso a los eventos y participaciones privados, solamente de aquellos que sean públicos a menos que el usuario sea tu amigo:

The image shows the public profile view for 'Sergio Zuñiga'. It includes basic user info (2 followers, 1 following), a profile picture, and an 'Edit profile' button. Below are tabs for 'Events' (selected), 'Participations', and 'Private'. The 'Events' tab displays a grid of six event cards: 'Mountain Rave', 'Feria Gaming 2025', 'A Pasear', 'Summer Music Festival 2025', 'Alv', and 'Test'. Each card shows a thumbnail, event name, date, time, and location. A red '+' button is located in the bottom right corner of the profile section.

Bibliografía y recursos consultados

Angular. (2025). *Angular Documentation*. Google LLC. Recuperado de <https://angular.dev/>

Anthropic. (2025). *Claude AI*. Recuperado de <https://claude.ai>

Google Developers. (2025). *Google Maps JavaScript API Documentation*. Google LLC. Recuperado de https://developers.google.com/maps/documentation/javascript?hl=es_419

Google Developers. (2025). *Google Maps Geocoding API Documentation*. Google LLC. Recuperado de https://developers.google.com/maps/documentation/geocoding/?hl=es_419

NestJS. (2025). *NestJS Documentation*. Recuperado de <https://docs.nestjs.com/>

NestJS. (2025). *Security: Authentication with JWT*. Recuperado de <https://docs.nestjs.com/security/authentication>

NestJS. (2025). *Recipes: Prisma Integration*. Recuperado de <https://docs.nestjs.com/recipes/prisma>

ngx-toastr. (2025). *ngx-toastr: Toast notifications for Angular*. Recuperado de <https://www.npmjs.com/package/ngx-toastr>

OpenAI. (2025). *ChatGPT (GPT-5) [Modelo de lenguaje]*. Recuperado de <https://chat.openai.com>

Prisma. (2025). *Prisma ORM Documentation*. Recuperado de <https://www.prisma.io/docs>

Socket.IO. (2025). *Socket.IO Documentation*. Recuperado de <https://socket.io/docs/v4>

Tailwind Labs. (2025). *TailwindCSS Documentation*. Recuperado de <https://tailwindcss.com/docs>