

Proyecto Visvire

Documentación Técnica Integral y Guía de Referencia

Este documento consolida la arquitectura, decisiones técnicas, comandos utilizados y criterios de diseño aplicados en el proyecto Visvire. Está pensado como material de referencia personal, onboarding de desarrolladores y contexto técnico para futuras extensiones o asistencia por IA.

1. Visión General del Proyecto

Visvire es un sistema de control y gestión para una PyME, orientado a inventario, punto de venta, alojamiento y administración. El proyecto fue diseñado con un horizonte de uso mínimo de cinco años, priorizando mantenibilidad, claridad arquitectónica y escalabilidad.

2. Tecnologías y Versiones Utilizadas

- **Node.js:** v24.x (LTS reciente)
- **Angular CLI:** v21.x (Standalone)
- **NestJS:** v11.x
- **PostgreSQL:** v18 (alpine)
- **ORM:** TypeORM
- **Servidor web:** Nginx
- **Contenedores:** Docker + Docker Compose

3. Justificación de Decisiones Técnicas

3.1 Frontend – Angular

Angular fue seleccionado por su arquitectura opinionada, soporte a largo plazo y adecuación para proyectos empresariales. El uso de componentes standalone reduce complejidad estructural y dependencias innecesarias. La organización por features permite escalar el proyecto por dominio funcional.

3.2 Backend – NestJS y TypeORM

NestJS proporciona una arquitectura modular clara y un sistema robusto de inyección de dependencias. TypeORM permite representar el dominio de negocio mediante entidades, facilitando la trazabilidad entre código y base de datos.

3.3 Base de Datos – PostgreSQL en Docker

PostgreSQL fue elegido por su estabilidad y adopción en entornos productivos. Docker permite aislar la base de datos del sistema operativo y garantizar un entorno reproducible entre distintos desarrolladores.

3.4 Infraestructura – Nginx

Nginx actúa como reverse proxy y servidor de archivos estáticos en producción. Permite exponer un único punto de entrada (puertos 80/443), centralizar HTTPS y separar responsabilidades entre frontend y backend.

4. Estructura del Repositorio

Ruta	Descripción
/apps/frontend	Aplicación Angular
/apps/backend	API NestJS
/infra/nginx	Configuración de Nginx
docker-compose.db.yml	PostgreSQL en desarrollo
docker-compose.yml	Stack completo en producción

5. Comandos Utilizados

5.1 Configuración de entorno

```
nvm install 24
nvm use 24
echo 24 > .nvmrc
```

5.2 Creación Frontend Angular

```
npm install -g @angular/cli@21
ng new frontend --routing --style=scss --ssr=false
```

Routing habilitado para SPA, SCSS por escalabilidad y SSR deshabilitado por no ser requerido.

5.3 Creación Backend NestJS

```
npm install -g @nestjs/cli@11
nest new backend --package-manager npm
```

6. Modos de Ejecución

6.1 Desarrollo (DEV)

```
docker compose -f docker-compose.db.yml up -d
npm run start:dev    # apps/backend
npm start            # apps/frontend
```

En desarrollo se prioriza velocidad, hot reload y facilidad de depuración.

6.2 Producción / Simulación local

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

En producción todo corre en contenedores. Nginx sirve Angular y redirige /api al backend.

7. Arquitectura Final

Cliente → Nginx (80/443) → Backend NestJS (3000) → PostgreSQL (5432)

Esta arquitectura reduce la superficie expuesta, mejora la seguridad y facilita el mantenimiento a largo plazo.