

Decisiones Técnicas

Plataforma de Inteligencia y Monitoreo para PYMEs

1. Stack Tecnológico Seleccionado

Frontend

- **Framework:** React Js
- **Lenguaje:** TypeScript
- **Build Tool:** Vite
- **Styling:** Tailwind CSS 3
- **Routing:** React Router
- **HTTP Client:** Axios
- **Visualizaciones:** Recharts
- **State Management:** React Context API + Hooks

Backend

- **Framework:** Laravel 11 Lenguaje: PHP 8.2+
- **Base de Datos:** MySQL 8.0
- **Autenticación:** Laravel Sanctum
- **API:** RESTful

DevOps

- Control de Versiones: Git + GitHub
- Testing Backend: PHPUnit
- Frontend: Vitest + React Testing Library
- Deploy Frontend: Vercel
- Deploy Backend: Railway

2. Justificación de Decisiones

¿Por qué React?

- Ecosistema maduro
- Componentes reutilizables
- Virtual DOM optimiza rendimiento Excelente para SPAs
- Gran cantidad de librerías de visualización

Alternativas evaluadas: Vue.js, Angular

Decisión: React por balance entre productividad y ecosistema

¿Por qué TypeScript?

- Detección de errores en desarrollo
- Autocompletado y mejor DX
- Código más mantenible
- Facilita refactoring
- Interfaces claras para datos de API

¿Por qué Vite?

- Extremadamente rápido (HMR instantáneo)
- Build optimizado out-of-the-box
- Configuración mínima
- Soporte nativo de TypeScript

Alternativas: Create React App, Next.js

Decisión: Vite por velocidad y simplicidad

¿Por qué Laravel?

- Framework PHP más popular
- Eloquent ORM facilita manejo de DB
- Laravel Sanctum para auth API
- Migraciones y seeders
- Validación robusta
- Excelente documentación

Alternativas: Node.js/Express, Django

Decisión: Laravel por productividad

¿Por qué MySQL?

- ✓ Base de datos relacional madura
- ✓ Excelente integración con Laravel
- ✓ Soporte de transacciones ACID
- ✓ Amplia disponibilidad en hosting

Alternativas: PostgreSQL, MongoDB **Decisión:**

MySQL suficiente para MVP

¿Por qué Tailwind CSS?

- ✓ Desarrollo rápido con utility classes
- ✓ Diseño responsive fácil
- ✓ Bundle size optimizado
- ✓ Consistencia de diseño

Alternativas: CSS Modules, Styled Components **Decisión:**

Tailwind por velocidad

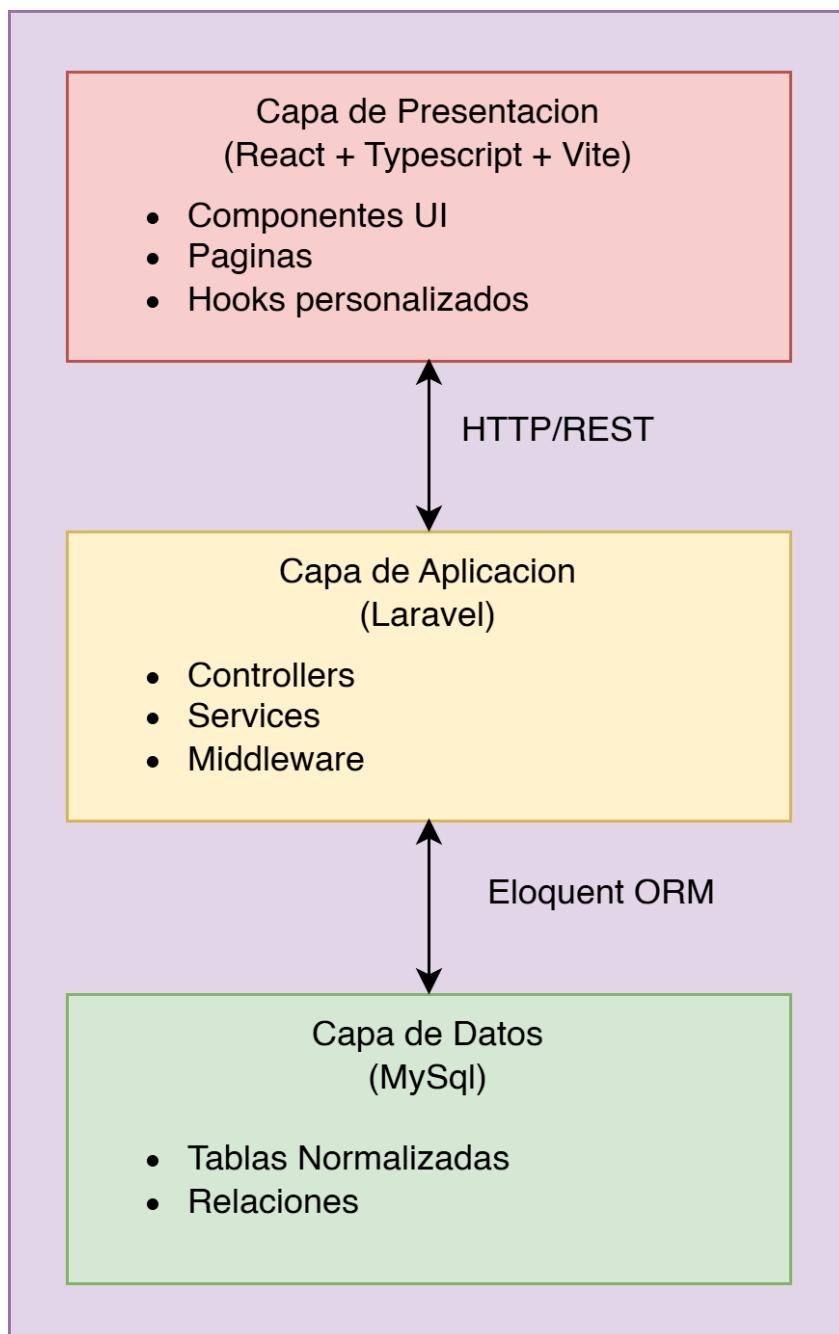
¿Por qué Recharts?

- ✓ Componentes React nativos
- ✓ Sintaxis declarativa
- ✓ Responsive por defecto
- ✓ Suficiente para MVP

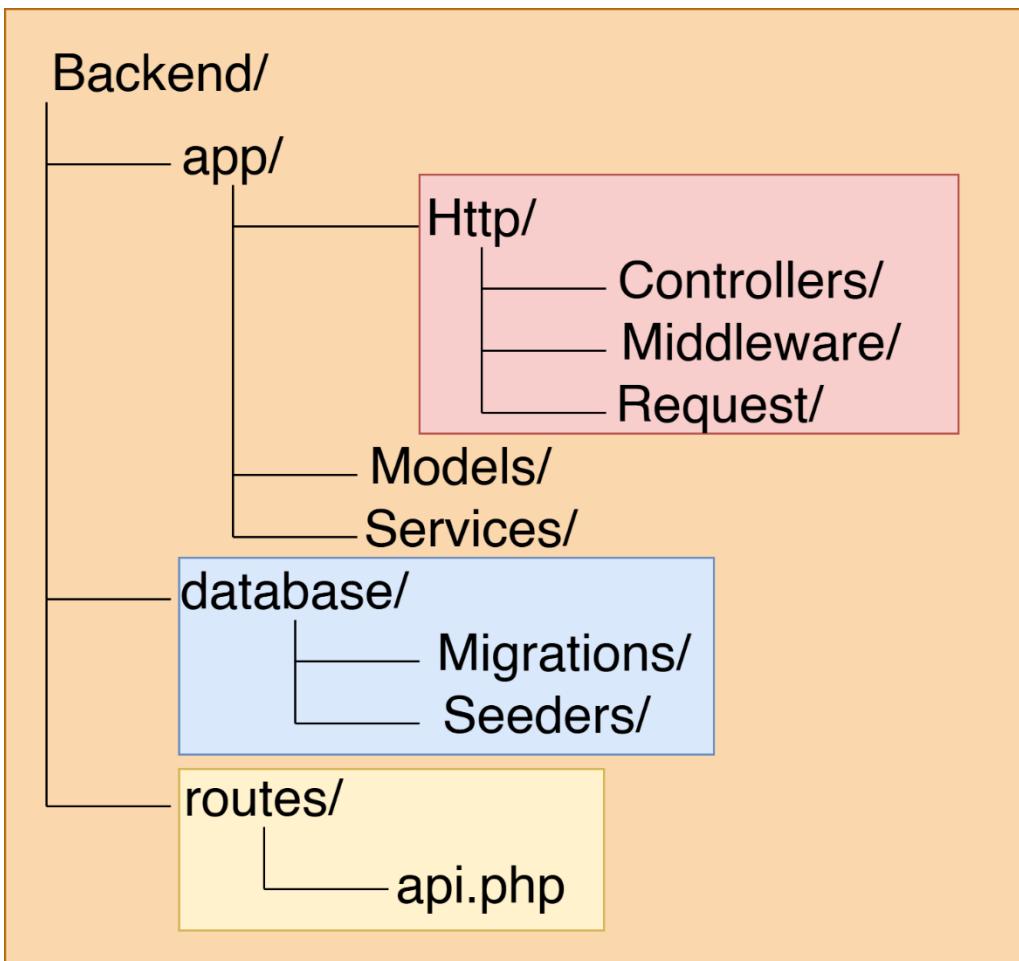
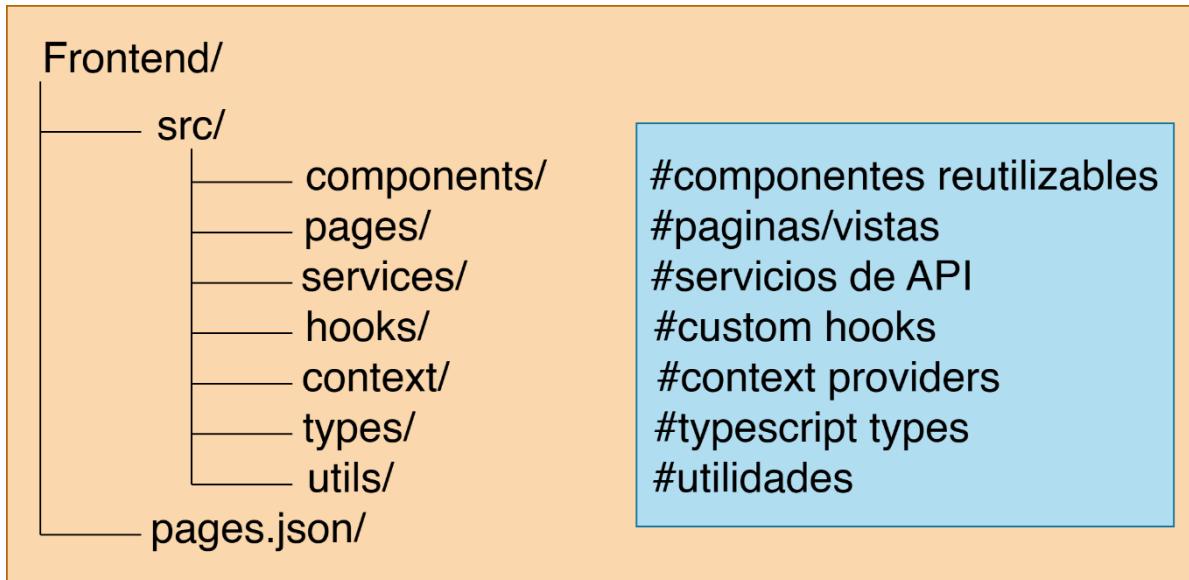
Alternativas: Chart.js, D3.js **Decisión:**

Recharts por simplicidad

3. Arquitectura de 3 Capas



4. Estructura de Carpetas



5. Estrategia de Seguridad

Autenticación

- Contraseñas hasheadas con bcrypt
- Tokens de Sanctum con expiración
- Middleware de autenticación
- Middleware de roles **Validación**
- Form Requests en Laravel
- Validación en frontend con TypeScript Sanitización de inputs

Protección de APIs

- CORS configurado
- Rate limiting
- HTTPS en producción
- Validación de tokens

6. Testing Strategy

Backend (PHPUnit)

- Unit Tests: Modelos, Services
- Feature Tests: Endpoints de API
- Coverage objetivo: > 70%

Frontend (Vitest)

- Unit Tests: Componentes, hooks
- Integration Tests: Flujos completos
- Coverage objetivo: > 60%

7. Deployment Strategy

Frontend (Vercel)

- Optimizado para React/Vite
- Deploy automático desde GitHub
- CDN global
- HTTPS automático

Backend (Railway)

- Soporte nativo de Laravel
- MySQL incluido
- Deploy desde GitHub
- Variables de entorno fáciles

8. Comparativa de Alternativas

Aspecto	Elegida	Alt 1	Alt 2	Razón
Frontend	React	Vue	Angular	Ecosistema
Build	Vite	CRA	Next.js	Velocidad
Backend	Laravel	Node.js	Django	Productividad
DB	MySQL	PostgreSQL	MongoDB	Suficiente
Styling	Tailwind	CSS Modules	Styled-C	Velocidad
Charts	Recharts	Chart.js	D3.js	Balance
Deploy FE	Vercel	Netlify	GH Pages	Optimizado
Deploy BE	Railway	Heroku	DO	Moderno