Documentación Técnica Completa - INMOVA

Índice

Descripción General

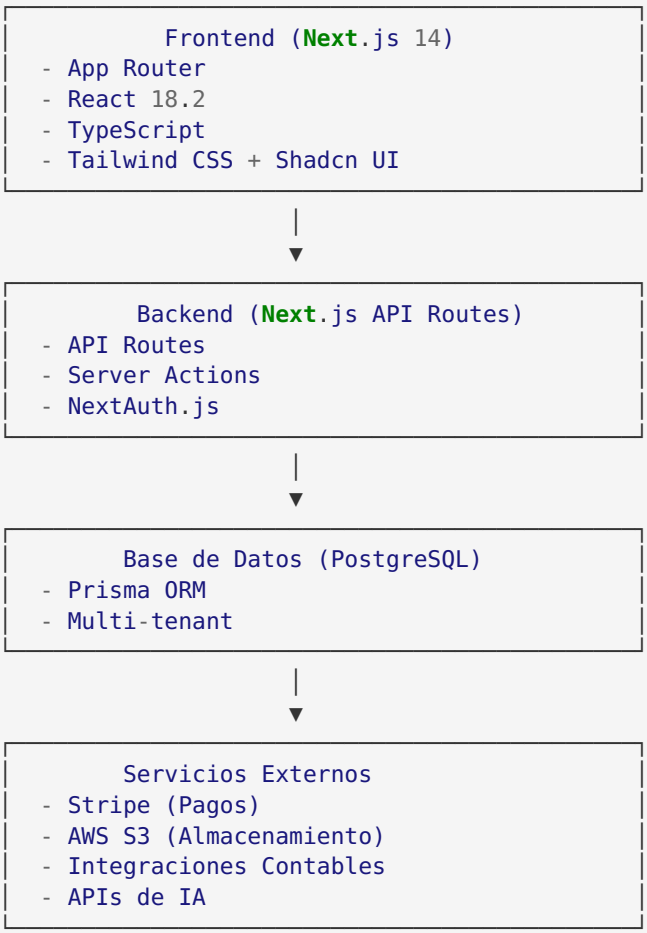
INMOVA es una plataforma SaaS multi-vertical para la gestión integral de inmuebles. El sistema está diseñado para consolidar todas las operaciones inmobiliarias en una única plataforma, eliminando la fragmentación de software y reduciendo costos operativos.

Características Principales

- **88 módulos profesionales** organizados en 7 verticales de negocio
- **Multi-empresa (Multi-tenant)** con soporte completo
- **7 verticales**: Residencial, STR, Coliving, House Flipping, Construcción, Servicios Profesionales, Comercial
- **Integraciones contables**: Zucchetti, ContaSimple, Sage, Holded, A3 Software, Alegra
- **PWA** con funcionalidad offline
- **White Label** personalización completa por empresa

Arquitectura del Sistema

Stack Tecnológico



Patrones de Diseño

1. **Repository Pattern**: Todos los accesos a datos se realizan a través de Prisma
2. **Service Layer**: Lógica de negocio encapsulada en servicios (`lib/`)
3. **API Routes**: RESTful endpoints en `app/api/`
4. **Component-Based**: UI modular con componentes reutilizables

---

Tecnologías Utilizadas

Frontend

- **Next.js 14.2.28**: Framework React con App Router
- **React 18.2**: Librería UI
- **TypeScript 5.2**: Tipado estático
- **Tailwind CSS 3.3**: Framework CSS utility-first
- **Shadcn UI**: Componentes UI accesibles
- **Framer Motion**: Animaciones
- **Recharts**: Visualización de datos

- **React Query**: State management y caché
- **Zustand**: State management global
- **Date-fns**: Manipulación de fechas
- **React Hook Form**: Gestión de formularios
- **Zod**: Validación de schemas

Backend

- **Next.js API Routes**: Endpoints RESTful
- **NextAuth.js 4.24**: Autenticación
- **Prisma 6.7**: ORM para PostgreSQL
- **bcryptjs**: Hash de contraseñas
- **jsonwebtoken**: JWT tokens
- **Tesseract.js**: OCR

Base de Datos

- **PostgreSQL**: Base de datos relacional
- **Prisma**: ORM y migrations

Servicios Externos

- **Stripe**: Procesamiento de pagos
- **AWS S3**: Almacenamiento de archivos
- **Sendgrid/SMTP**: Email

DevOps

- **Yarn**: Gestor de paquetes
- **ESLint**: Linting
- **Prettier**: Code formatting
- **Docker**: Containerización (opcional)

Estructura del Proyecto

```
nextjs_space/
│
├── app/                          # App Router de Next.js
│   ├── (auth)/                   # Rutas de autenticación
│   ├── admin/                    # Panel super-admin
│   ├── api/                      # API Routes
│   ├── dashboard/                # Dashboard principal
│   ├── landing/                  # Landing pages
│   ├── portal-inquilino/         # Portal inquilinos
│   ├── portal-propietario/       # Portal propietarios
│   ├── portal-proveedor/         # Portal proveedores
│   ├── [modulos]/                # Módulos específicos
│   ├── layout.tsx                # Layout principal
│   ├── page.tsx                  # Home page
│   └── globals.css               # Estilos globales
│
├── components/                   # Componentes React
│   ├── ui/                       # Componentes UI (Shadcn)
│   ├── layout/                   # Layouts (Sidebar, Header)
│   ├── automation/               # Componentes de IA
│   ├── dashboard/                # Componentes dashboard
│   └── pwa/                      # Componentes PWA
│
├── lib/                          # Lógica de negocio
│   ├── services/                 # Servicios de negocio
│   ├── hooks/                    # Custom React Hooks
│   ├── utils.ts                  # Utilidades
│   ├── db.ts                     # Prisma client
│   ├── auth-options.ts           # Configuración NextAuth
│   ├── permissions.ts            # RBAC
│   └── [*-service.ts]            # Servicios específicos
│
├── prisma/                       # Prisma ORM
│   └── schema.prisma             # Schema de BD
│
├── public/                       # Archivos estáticos
│   ├── favicon.svg
│   ├── manifest.json             # PWA manifest
│   └── sw.js                     # Service Worker
│
├── scripts/                      # Scripts de utilidad
│   ├── seed.ts                   # Seed de datos
│   ├── create-super-admin.ts
│   └── [otros scripts]
│
├── types/                        # Definiciones TypeScript
│   └── next-auth.d.ts
│
├── .env                          # Variables de entorno
├── next.config.js                # Configuración Next.js
├── tailwind.config.ts            # Configuración Tailwind
├── tsconfig.json                 # Configuración TypeScript
└── package.json                  # Dependencias
```

Módulos Principales

1. Gestión de Propiedades

- **Edificios** (`/edificios`): CRUD completo de propiedades
- **Unidades** (`/unidades`): Gestión de unidades/apartamentos
- **Inquilinos** (`/inquilinos`): Base de datos de inquilinos
- **Contratos** (`/contratos`): Contratos de alquiler

2. Finanzas

- **Pagos** (`/pagos`): Tracking de pagos
- **Gastos** (`/gastos`): Control de gastos
- **Facturación B2B** (`/admin/facturacion-b2b`): Facturación empresarial
- **Contabilidad** (`/contabilidad`): Panel del Director Financiero

3. Mantenimiento

- **Solicitudes** (`/mantenimiento`): Corrective maintenance
- **Preventivo** (`/mantenimiento-preventivo`): Planned maintenance
- **Mantenimiento Pro** (`/mantenimiento-pro`): Advanced features
- **Órdenes de Trabajo** (`/ordenes-trabajo`): Work orders

4. Multi-Vertical

- **STR** (`/str`): Short-Term Rental management
- **House Flipping** (`/flipping`): Investment projects
- **Construcción** (`/construction`): Construction projects
- **Professional** (`/professional`): Professional services
- **Coliving** (`/room-rental`): Room rental management

5. Automatización e IA

- **Asistente IA** (`/asistente-ia`): Chatbot inteligente
- **OCR** (`/ocr`): Document scanning
- **Analytics** (`/analytics`): Advanced analytics
- **BI** (`/bi`): Business Intelligence

6. Administración

- **Super Admin** (`/admin`): Panel de super-administrador
- **Usuarios** (`/admin/usuarios`): User management
- **Clientes** (`/admin/clientes`): Client companies
- **Módulos** (`/admin/modulos`): Module management
- **Personalización** (`/admin/personalizacion`): White Label

7. Comunicación

- **Chat** (`/chat`): Chat interno
- **SMS** (`/sms`): SMS notifications
- **Notificaciones** (`/notificaciones`): Notification center
- **Email**: Email integration

Base de Datos

Schema Principal (Prisma)

Entidades Core

```
model Company {
  id                  String    @id @default(cuid())
  nombre              String
  cif                 String?
  activo              Boolean   @default(true)
  subscriptionPlanId  String?
  subscriptionPlan    SubscriptionPlan? @relation(fields: [subscriptionPlanId])

  // Relaciones
  users               User[]
  buildings           Building[]
  tenants             Tenant[]
  // ... más relaciones
}

model User {
  id                  String    @id @default(cuid())
  email               String    @unique
  name                String
  password            String
  role                UserRole  @default(gestor)
  companyId           String
  company             Company   @relation(fields: [companyId])
}

model Building {
  id                  String    @id @default(cuid())
  nombre              String
  direccion           String
  companyId           String
  company             Company   @relation(fields: [companyId])
  units               Unit[]
}

model Unit {
  id                  String    @id @default(cuid())
  nombre              String
  buildingId          String
  building            Building  @relation(fields: [buildingId])
  rentaMensual        Float?
  contracts           Contract[]
}

model Tenant {
  id                  String    @id @default(cuid())
  nombre              String
  email               String
  telefono            String?
  companyId           String
  company             Company   @relation(fields: [companyId])
  contracts           Contract[]
}

model Contract {
  id                  String    @id @default(cuid())
  unitId              String
  unit                Unit      @relation(fields: [unitId])
  tenantId            String
  tenant              Tenant    @relation(fields: [tenantId])
  fechaInicio         DateTime
  fechaFin            DateTime
  rentaMensual        Float
```

```
  payments              Payment[]
}
```

Roles de Usuario

```
enum UserRole {
  super_admin      // Acceso total a todas las empresas
  administrador    // Admin de una empresa
  gestor           // Manager con permisos limitados
  operador         // Operador básico
  soporte          // Equipo de soporte
}
```

Índices y Optimización

El schema incluye índices optimizados para:
- Búsquedas por `companyId`
- Queries de contratos activos
- Pagos pendientes
- Solicitudes de mantenimiento

Autenticación y Autorización

NextAuth.js Configuration

Archivo: `lib/auth-options.ts`

```
export const authOptions: NextAuthOptions = {
  adapter: PrismaAdapter(prisma),
  providers: [
    CredentialsProvider({
      // Credentials authentication
    })
  ],
  session: {
    strategy: 'jwt'
  },
  callbacks: {
    async jwt({ token, user }) {
      // Add custom fields to JWT
    },
    async session({ session, token }) {
      // Add custom fields to session
    }
  }
}
```

RBAC (Role-Based Access Control)

Archivo: `lib/permissions.ts`

```
const PERMISSIONS = {
  super_admin: {
    read: true,
    create: true,
    update: true,
    delete: true,
    manageUsers: true,
    manageCompany: true,
    impersonateClients: true
  },
  administrador: {
    read: true,
    create: true,
    update: true,
    delete: true,
    manageUsers: true
  },
  gestor: {
    read: true,
    create: true,
    update: true,
    delete: false
  },
  operador: {
    read: true,
    create: false,
    update: false,
    delete: false
  }
}
```

Middleware Protection

Archivo: `middleware.ts`

- Rate limiting
- Security headers (CSP)
- Role-based route protection
- Session validation

APIs y Servicios

Estructura de API Routes

```
app/api/
├── auth/                    # Autenticación
├── buildings/               # CRUD edificios
├── units/                   # CRUD unidades
├── tenants/                 # CRUD inquilinos
├── contracts/               # CRUD contratos
├── payments/                # CRUD pagos
├── maintenance/             # Mantenimiento
├── admin/                   # Super-admin APIs
├── accounting/              # Integraciones contables
├── stripe/                  # Stripe integration
├── room-rental/             # Coliving APIs
├── str/                     # Short-Term Rental APIs
└── [más módulos]/
```

Servicios de Negocio

Todos los servicios están en `lib/` con naming convention `*-service.ts`:

- `accounting-service.ts`: Integración contable
- `stripe-config.ts` / `stripe-customer.ts`: Stripe
- `room-rental-service.ts`: Coliving
- `screening-service.ts`: Screening inquilinos
- `ocr-service.ts`: OCR de documentos
- `valoracion-service.ts`: Valoración inmuebles
- `sms-service.ts`: SMS notifications
- `calendar-service.ts`: Calendario unificado
- `ai-assistant-service.ts`: Asistente IA
- Y muchos más...

Integración con Stripe

Payment Intents: Para pagos únicos
Subscriptions: Para rentas mensuales recurrentes
Webhooks: `/api/stripe/webhook` para eventos

Deployment

Variables de Entorno Requeridas

```
# Base de Datos
DATABASE_URL="postgresql://..."

# NextAuth
NEXTAUTH_URL="https://tu-dominio.com"
NEXTAUTH_SECRET="tu-secret-largo-y-aleatorio"

# Stripe
STRIPE_SECRET_KEY="sk_..."
STRIPE_PUBLISHABLE_KEY="pk_..."
STRIPE_WEBHOOK_SECRET="whsec_..."

# AWS S3 (Opcional)
AWS_BUCKET_NAME="tu-bucket"
AWS_REGION="eu-west-1"
AWS_ACCESS_KEY_ID="..."
AWS_SECRET_ACCESS_KEY="..."

# Integraciones Contables (Opcional)
ZUCCHETTI_CLIENT_ID="..."
CONTASIMPLE_AUTH_KEY="..."
SAGE_API_KEY="..."
# etc.
```

Build Process

```
# 1. Instalar dependencias
yarn install

# 2. Generar Prisma Client
yarn prisma generate

# 3. Ejecutar migraciones
yarn prisma migrate deploy

# 4. Build de producción
yarn build

# 5. Iniciar servidor
yarn start
```

Deployment en Producción

El proyecto está configurado para deployment con:
- Standalone output mode
- Optimización de imágenes
- Static file serving
- Build caching

Mantenimiento

Logs

Sistema de logging estructurado en `lib/logger.ts`:
- `logger.info()`: Información general
- `logger.error()`: Errores
- `logger.warn()`: Advertencias
- `logger.debug()`: Debug (solo en dev)

Monitoreo

- **Health Check**: `/api/system/health`
- **Audit Logs**: Tabla `AuditLog` para auditoría
- **Error Tracking**: Sentry (opcional)

Backups

- Base de datos: Backups automáticos de PostgreSQL
- Archivos: S3 con versionado
- Servicio de backup: `lib/backup-service.ts`

Updates

```
# Actualizar dependencias
yarn upgrade-interactive

# Verificar
yarn tsc --noEmit
yarn build
```

Testing

Ejecuta el testing completo:

```
yarn test
```

El testing incluye:
- ✅ TypeScript compilation
- ✅ Next.js build
- ✅ Dev server startup
- ✅ API endpoints health
- ✅ Runtime validation

Contacto y Soporte

- **Email**: soporte@inmova.com
- **Web**: https://inmova.app
- **Documentación**: Ver archivos `.md` en el proyecto

Última actualización: Diciembre 2025
Versión: 1.0
Autor: Enxames Investments SL