DOCUMENTACIÓN TÉCNICA

Sistema de Predicción Cardiovascular

BACKEND

Versión: 1.0

Fecha: 2024

Autor: Equipo de Desarrollo

ÍNDICE

1. INTRODUCCIÓN
2. ARQUITECTURA DEL SISTEMA
3. TECNOLOGÍAS UTILIZADAS
4. ESTRUCTURA DEL PROYECTO
5. CONFIGURACIÓN Y INSTALACIÓN
6. API REST
7. AUTENTICACIÓN Y AUTORIZACIÓN
8. BASE DE DATOS
9. SERVICIOS DE MACHINE LEARNING
10. MICROSERVICIOS
11. FLUJO DE PROCESAMIENTO
12. DESPLIEGUE
13. MANTENIMIENTO Y MONITOREO

1. INTRODUCCIÓN

El backend del Sistema de Predicción Cardiovascular es una aplicación robusta desarrollada en Django que proporciona servicios REST para la gestión de pacientes, predicciones cardiovasculares y análisis de datos médicos. Esta documentación describe la arquitectura, tecnologías, configuración y funcionamiento del sistema.

1.1 Objetivos del Sistema

• Proporcionar una API REST segura y escalable para el frontend • Gestionar la autenticación y autorización de usuarios • Procesar datos médicos y generar predicciones cardiovasculares • Almacenar y gestionar información de pacientes de forma segura • Integrar modelos de machine learning para análisis predictivo • Proporcionar servicios de analytics y reportes

1. ARQUITECTURA GENERAL

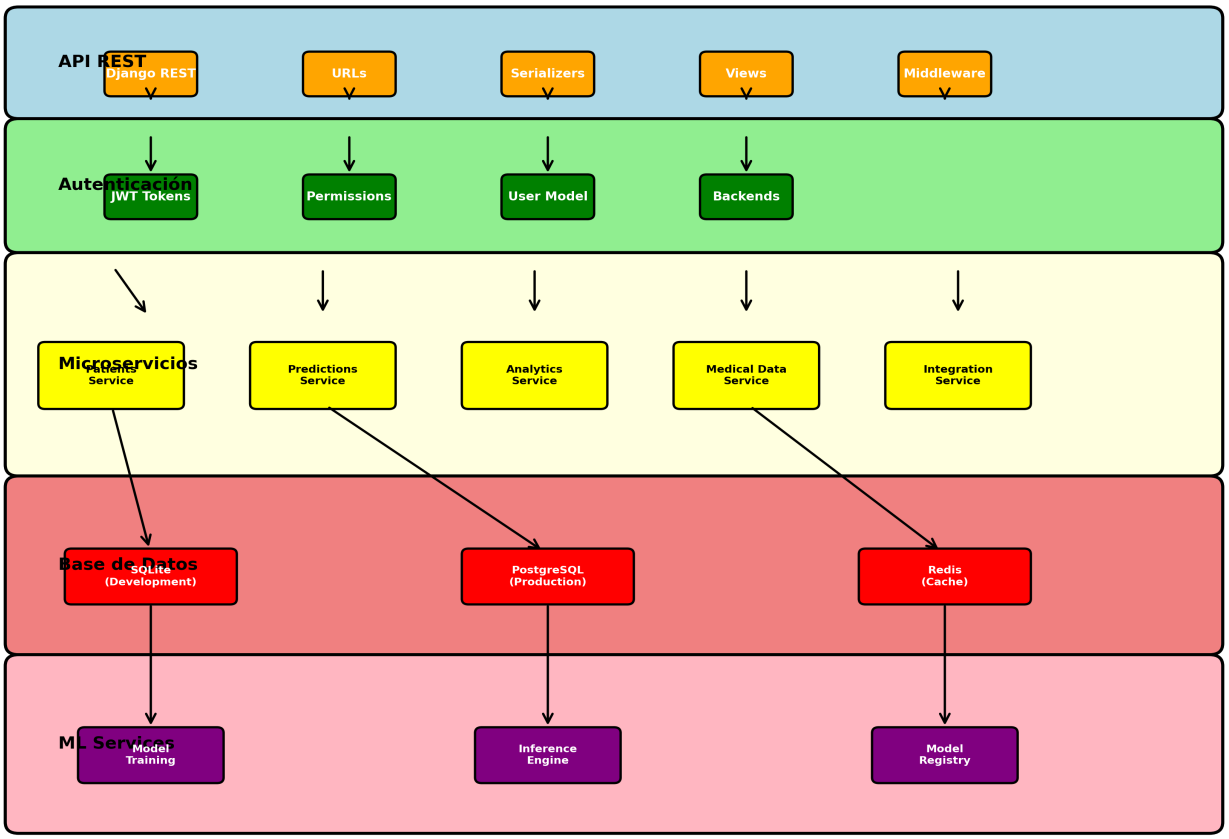
1.1 Visión General del Sistema

El backend del Sistema de Predicción Cardiovascular sigue una arquitectura de microservicios, con capas bien definidas para API REST, autenticación, servicios de ML, base de datos y analytics. Permite escalabilidad, mantenibilidad y alta disponibilidad.

1.2 Componentes Principales

• API REST • Autenticación y Autorización • Microservicios • Base de Datos • Servicios de Machine Learning • Analytics y Reportes

Arquitectura del Backend - Sistema de Predicción Cardiovascular



2. TECNOLOGÍAS Y FRAMEWORKS

3. MODELO DE DATOS

4. API REST

5. MOTOR DE MACHINE LEARNING

6. INTEGRACIÓN Y DEPLOYMENT

7. SEGURIDAD Y PERFORMANCE

8. CÓDIGO FUENTE RELEVANTE

backend/	Raíz del backend
■■■ apps/	Aplicaciones principales
■ ■■■ authentication/	Autenticación y usuarios
■ ■■■ patients/	Gestión de pacientes
■ ■■■ predictions/	Predicciones ML
■ ■■■ analytics/	Análisis y reportes
■ ■■■ medical_data/	Datos médicos
■ ■■■ integration/	Integraciones externas
■■■ ml_models/	Modelos de ML
■■■ config/	Configuraciones
■■■ requirements.txt	Dependencias
■■■ manage.py	Script de gestión

5. CONFIGURACIÓN Y INSTALACIÓN

5.1 Requisitos del Sistema

• Python 3.8 o superior • pip (gestor de paquetes de Python) • Git • PostgreSQL (para producción) • Redis (opcional, para caché)

5.2 Instalación Paso a Paso

Paso	Comando	Descripción
1	git clone <repo>	Clonar el repositorio
2	cd backend	Entrar al directorio del backend
3	python -m venv env	Crear entorno virtual
4	source env/bin/activate	Activar entorno virtual (Linux/Mac)
4	env\Scripts\activate	Activar entorno virtual (Windows)
5	pip install -r requirements.txt	Instalar dependencias
6	python manage.py migrate	Ejecutar migraciones
7	python manage.py createsuperuser	Crear usuario administrador
8	python manage.py runserver	Iniciar servidor de desarrollo

6. API REST

6.1 Endpoints Principales

Método	Endpoint	Descripción
POST	/api/auth/login/	Autenticación de usuario
POST	/api/auth/register/	Registro de nuevo usuario
GET	/api/patients/	Listar pacientes
POST	/api/patients/	Crear paciente
GET	/api/patients/{id}/	Obtener paciente específico
PUT	/api/patients/{id}/	Actualizar paciente
DELETE	/api/patients/{id}/	Eliminar paciente
POST	/api/predictions/	Crear predicción
GET	/api/predictions/	Listar predicciones
GET	/api/analytics/	Obtener analytics

6.2 Ejemplo de Respuesta API


```
{ "id": 1, "patient": { "id": 123, "name": "Juan Pérez", "age": 45,
"gender": "M" }, "prediction": { "risk_score": 0.75, "risk_level": "HIGH",
"confidence": 0.89 }, "created_at": "2024-01-15T10:30:00Z" }
```

7. AUTENTICACIÓN Y AUTORIZACIÓN

7.1 JWT Authentication

El sistema utiliza JSON Web Tokens (JWT) para autenticación stateless. Los tokens contienen información del usuario y permisos, y se validan en cada petición.

7.2 Permisos y Roles

• **Admin:** Acceso completo al sistema • **Doctor:** Gestión de pacientes y predicciones • **Nurse:** Lectura de datos y creación de registros básicos • **Analyst:** Acceso a analytics y reportes

8. BASE DE DATOS

8.1 Modelos Principales

Modelo	Campos Principales	Relaciones
User	username, email, role	OneToMany: Patient
Patient	name, age, gender, dni	ManyToOne: User, OneToMany: MedicalRecord
MedicalRecord	diagnosis, treatment, date	ManyToOne: Patient
Prediction	risk_score, confidence, model_used	ManyToOne: Patient
Analytics	metrics, period, data	ManyToOne: User

9. SERVICIOS DE MACHINE LEARNING

9.1 Pipeline de ML

El sistema incluye un pipeline completo de machine learning que incluye: • Preprocesamiento de datos médicos • Entrenamiento de modelos predictivos • Validación y evaluación de modelos • Inferencia en tiempo real • Monitoreo de rendimiento

9.2 Modelos Implementados

• **Random Forest:** Para predicción de riesgo cardiovascular • **Logistic Regression:** Para clasificación binaria de riesgo • **Gradient Boosting:** Para predicciones más precisas • **Neural Networks:** Para patrones complejos en datos médicos

10. MICROSERVICIOS

10.1 Arquitectura de Microservicios

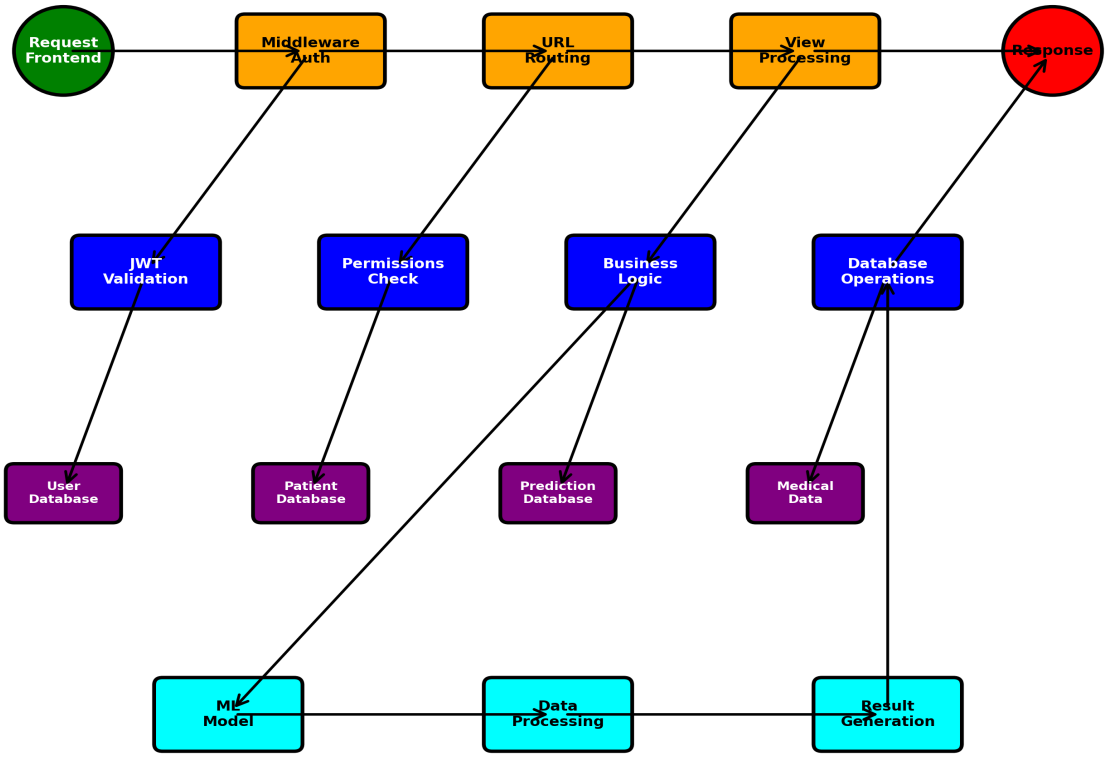
El sistema está diseñado como una arquitectura de microservicios donde cada servicio tiene responsabilidades específicas y puede ser desplegado independientemente.

Servicio	Responsabilidad	Tecnologías
Patient Service	Gestión de pacientes	Django, PostgreSQL
Prediction Service	Predicciones ML	Django, scikit-learn
Analytics Service	Reportes y analytics	Django, pandas
Auth Service	Autenticación	Django, JWT
Integration Service	APIs externas	Django, requests

11. FLUJO DE PROCESAMIENTO

El flujo de procesamiento describe cómo las peticiones son manejadas desde que llegan al servidor hasta que se genera la respuesta, incluyendo autenticación, validación, procesamiento de negocio y generación de respuestas.

Flujo de Procesamiento del Backend



12. DESPLIEGUE

12.1 Configuración de Producción

Para el despliegue en producción, se recomienda: • Usar PostgreSQL como base de datos principal • Configurar Redis para caché • Usar Gunicorn como servidor WSGI • Configurar Nginx como proxy reverso • Implementar HTTPS con certificados SSL • Configurar backups automáticos

12.2 Variables de Entorno

```
DEBUG=False SECRET_KEY=your-secret-key-here
DATABASE_URL=postgresql://user:pass@localhost/dbname
REDIS_URL=redis://localhost:6379
ALLOWED_HOSTS=yourdomain.com,www.yourdomain.com
CORS_ALLOWED_ORIGINS=https://yourdomain.com
```

13. MANTENIMIENTO Y MONITOREO

13.1 Tareas de Mantenimiento

• Actualización regular de dependencias • Monitoreo de logs y errores • Backup de base de datos • Limpieza de datos temporales • Actualización de modelos ML • Monitoreo de rendimiento

13.2 Herramientas de Monitoreo

• Django Debug Toolbar (desarrollo) • Sentry (monitoreo de errores) • Prometheus + Grafana (métricas) • ELK Stack (logs) • Health checks automáticos