DOCUMENTACIÓN TÉCNICA

Sistema de Predicción Cardiovascular

FRONTEND

Versión: 1.0

Fecha: 2024

Autor: Equipo de Desarrollo

ÍNDICE

1. INTRODUCCIÓN

El frontend del Sistema de Predicción Cardiovascular es una aplicación web moderna desarrollada en Next.js que proporciona una interfaz de usuario intuitiva y responsiva para la gestión de pacientes, visualización de predicciones cardiovasculares y análisis de datos médicos.

1.1 Objetivos del Frontend

• Proporcionar una interfaz de usuario moderna y responsiva • Facilitar la gestión eficiente de pacientes y datos médicos • Visualizar predicciones cardiovasculares de forma clara • Generar reportes y analytics interactivos • Garantizar una experiencia de usuario excepcional • Implementar autenticación segura y gestión de sesiones

1. ARQUITECTURA GENERAL

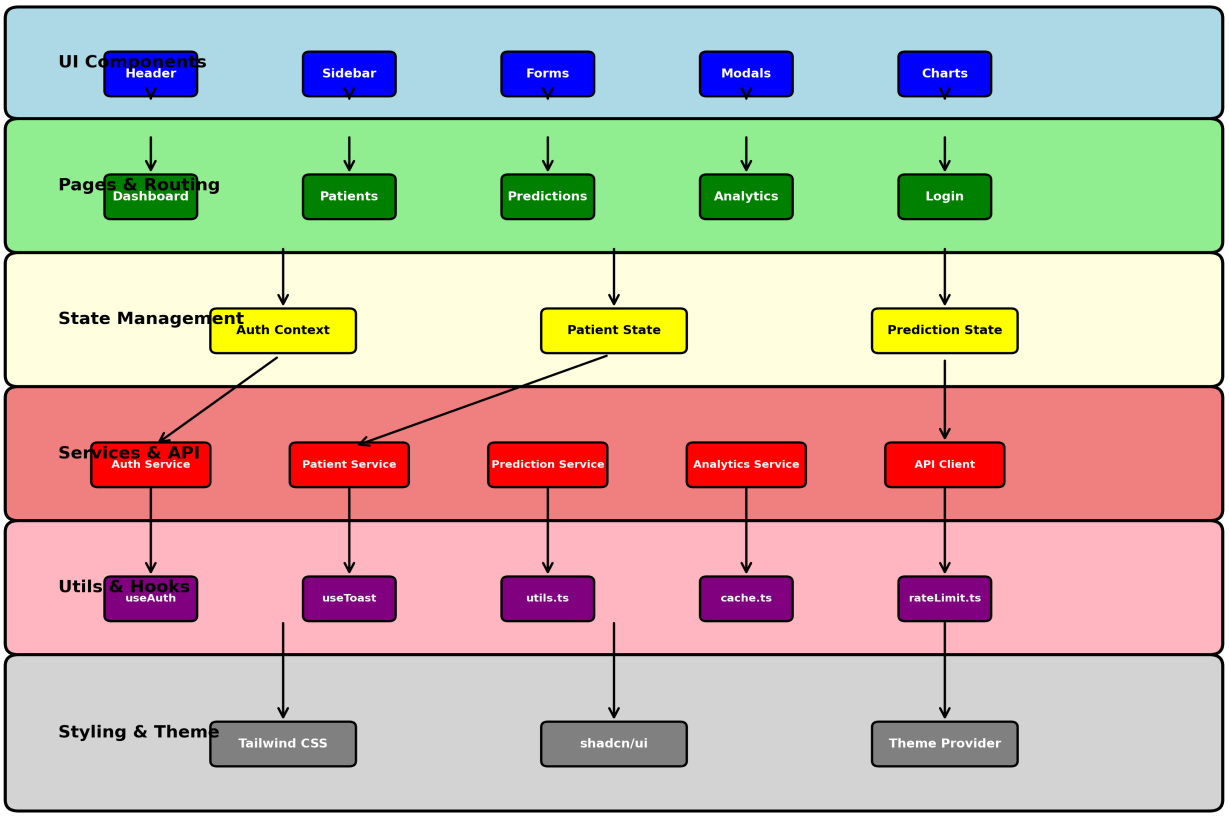
1.1 Visión General del Sistema

El frontend del Sistema de Predicción Cardiovascular es una SPA moderna desarrollada en Next.js 14, con arquitectura de componentes, gestión de estado y servicios API centralizados.

1.2 Componentes Principales

• App Router • Componentes UI • Servicios y API • Hooks personalizados • Gestión de estado • Temas y estilos

Arquitectura del Frontend - 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

frontend/	Raíz del frontend
■■■ app/	App Router (Next.js 14)
■ ■■■ dashboard/	Página del dashboard
■ ■■■ patients/	Página de pacientes
■ ■■■ login/	Página de login
■ ■■■ layout.tsx	Layout principal
■■■ components/	Componentes reutilizables
■ ■■■ ui/	Componentes base (shadcn/ui)
■ ■■■ header/	Componente header
■ ■■■ sidebar/	Componente sidebar
■ ■■■ forms/	Formularios
■■■ lib/	Utilidades y configuraciones
■ ■■■ api.ts	Cliente API
■ ■■■ auth.ts	Autenticación
■ ■■■ utils.ts	Utilidades generales
■■■ hooks/	Custom hooks
■■■ types/	Definiciones TypeScript
■■■ public/	Archivos estáticos

5. CONFIGURACIÓN Y INSTALACIÓN

5.1 Requisitos del Sistema

• Node.js 18+ o superior • npm, yarn o pnpm • Git • Editor de código (VS Code recomendado)

5.2 Instalación Paso a Paso

Paso	Comando	Descripción
1	git clone <repo>	Clonar el repositorio
2	cd frontend	Entrar al directorio del frontend
3	npm install	Instalar dependencias
4	cp .env.example .env.local	Configurar variables de entorno
5	npm run dev	Iniciar servidor de desarrollo
6	npm run build	Construir para producción
7	npm start	Iniciar servidor de producción

6. COMPONENTES UI

6.1 Componentes Principales

Componente	Propósito	Tecnologías
Header	Navegación principal y perfil	Next.js, Tailwind
Sidebar	Menú lateral y navegación	Next.js, Lucide Icons
PatientForm	Formulario de pacientes	React Hook Form, Zod
PredictionChart	Gráficos de predicciones	Recharts, D3.js
DataTable	Tabla de datos	TanStack Table
Modal	Ventanas modales	Radix UI
Toast	Notificaciones	Sonner

7. ROUTING Y NAVEGACIÓN

7.1 App Router (Next.js 14)

El proyecto utiliza el nuevo App Router de Next.js 14 que proporciona: • Routing basado en archivos • Layouts anidados • Server Components por defecto • Streaming y Suspense • Optimizaciones automáticas

7.2 Estructura de Rutas

Ruta	Página	Descripción
/	Home	Página principal
/login	Login	Autenticación
/dashboard	Dashboard	Panel principal
/patients	Patients	Gestión de pacientes
/predictions	Predictions	Predicciones
/analytics	Analytics	Reportes y análisis

8. GESTIÓN DE ESTADO

8.1 Context API

Se utiliza React Context para gestionar el estado global de la aplicación: • AuthContext: Maneja autenticación y sesión del usuario • ThemeContext: Gestiona el tema claro/oscuro • PatientContext: Estado de pacientes y filtros

8.2 Custom Hooks

• useAuth: Hook personalizado para autenticación • usePatients: Hook para gestión de pacientes • usePredictions: Hook para predicciones • useToast: Hook para notificaciones

9. SERVICIOS Y API

9.1 Cliente API

Se utiliza un cliente API centralizado que maneja: • Interceptores para tokens JWT • Manejo de errores global • Rate limiting • Retry automático • Cache de respuestas

9.2 Servicios Especializados

Servicio	Endpoint Base	Funcionalidad
AuthService	/api/auth	Login, registro, logout
PatientService	/api/patients	CRUD de pacientes
PredictionService	/api/predictions	Predicciones ML
AnalyticsService	/api/analytics	Reportes y métricas

10. HOOKS Y UTILIDADES

10.1 Custom Hooks

• useAuth: Gestión de autenticación y sesión • usePatients: CRUD de pacientes con cache • usePredictions: Predicciones con loading states • useToast: Sistema de notificaciones • useMobile: Detección de dispositivos móviles

10.2 Utilidades

• utils.ts: Funciones de utilidad general • cache.ts: Sistema de cache en memoria • rateLimit.ts: Control de rate limiting • retry.ts: Lógica de reintentos • validation.ts: Validaciones de formularios

11. ESTILOS Y TEMAS

11.1 Tailwind CSS

• Configuración personalizada en tailwind.config.ts • Variables CSS para colores y espaciado • Componentes reutilizables con @apply • Responsive design con breakpoints • Dark mode automático

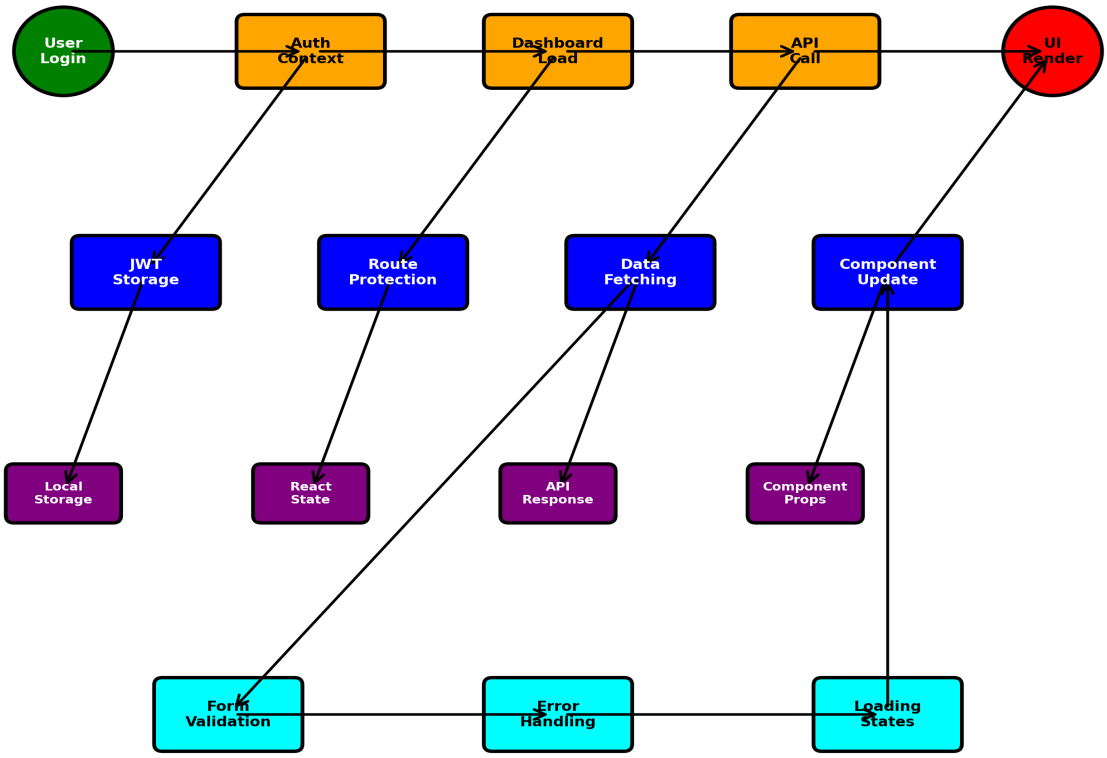
11.2 shadcn/ui

• Componentes base accesibles • Tema consistente y personalizable • Integración con Radix UI • Soporte para dark mode • Componentes: Button, Input, Card, Dialog, etc.

12. FLUJO DE INTERACCIÓN

El flujo de interacción describe cómo los usuarios navegan por la aplicación, cómo se gestiona el estado y cómo se comunican los componentes.

Flujo de Interacción del Frontend



13. DESPLIEGUE

13.1 Configuración de Producción

• Build optimizado con Next.js • Variables de entorno para producción • CDN para assets estáticos • Compresión y minificación • Cache headers apropiados • SSL/HTTPS obligatorio

13.2 Variables de Entorno

```
NEXT_PUBLIC_API_URL=https://api.yourdomain.com
NEXT_PUBLIC_APP_URL=https://yourdomain.com
NEXTAUTH_SECRET=your-secret-key NEXTAUTH_URL=https://yourdomain.com
```

14. OPTIMIZACIÓN Y RENDIMIENTO

14.1 Optimizaciones Implementadas

• Lazy loading de componentes • Code splitting automático • Optimización de imágenes con Next.js Image • Bundle analyzer para monitoreo • Tree shaking para reducir bundle size • Service Worker para cache offline

14.2 Métricas de Rendimiento

• First Contentful Paint (FCP) < 1.5s • Largest Contentful Paint (LCP) < 2.5s • Cumulative Layout Shift (CLS) < 0.1 • First Input Delay (FID) < 100ms • Time to Interactive (TTI) < 3.5s