Documentación: Web 2 - Kingdom Barber

Fecha: Octubre, 2025
Autores: Juan Rivera, Andrés Vallejo, Alejandro Urrego

1. Resumen del Proyecto

Kingdom Barber (PI_WEB2.0) es la plataforma web orientada al cliente y al personal de la barbería. Desarrollada como una aplicación moderna con **Next.js (React y TypeScript)**, su principal función es ofrecer una interfaz de usuario fluida, rápida y responsiva para todas las interacciones del día a día.

Este proyecto es un **front-end puro**. No contiene lógica de negocio ni acceso directo a la base de datos. Toda la información que muestra y todas las acciones que realiza (agendar citas, subir imágenes, etc.) se comunican exclusivamente con la **API Central de Java + Spring Boot**, que actúa como la única fuente de verdad.

2. Objetivos del Proyecto

Objetivo Principal

Proveer una experiencia de usuario excepcional y moderna para los clientes y una herramienta de gestión funcional para el personal de la barbería, consumiendo los servicios de una API centralizada.

Objetivos Específicos

- **Experiencia de Cliente Fluida:** Ofrecer un sistema de agendamiento de citas intuitivo y en tiempo real a través de un calendario interactivo.
- **Presentación de la Marca:** Mostrar los servicios, la galería de trabajos y la información de las sedes de una manera visualmente atractiva.
- **Herramienta para el Barbero:** Brindar al personal un panel simple donde puedan visualizar su agenda de citas programadas.
- **Desacoplamiento Total:** Funcionar como un cliente completamente independiente del back-end, comunicándose únicamente a través de peticiones HTTP a la API de Java.
- **Diseño Responsivo:** Asegurar que la aplicación sea totalmente funcional y se vea bien en cualquier dispositivo (móvil, tablet y escritorio).

3. Arquitectura y Estructura de Carpetas

El proyecto utiliza la arquitectura **App Router de Next.js**, que organiza la aplicación por rutas y favorece la modularidad a través de componentes, basada en la estructura de archivos real

del proyecto.

```
pi_web2.0/
├── 📁 app/            # Directorio principal de enrutamiento y páginas.
│   ├── 📁 dashboard/
│   │   └── 📁 barber/
│   │       └── 📜 page.tsx   # Vista de la agenda para el barbero.
│   ├── 📁 gallery/
│   │   └── 📜 page.tsx      # Página de la galería de imágenes.
│   ├── 📜 page.tsx          # Página de inicio (Home).
│   ├── 📜 services/
│   │   └── 📜 page.tsx      # Página de servicios.
│   ├── 📜 layout.tsx        # Layout principal de la aplicación.
│   └── 📜 globals.css       # Estilos globales.
│
├── 📁 components/       # Componentes reutilizables de la interfaz (UI).
│   ├── 📁 appointment/  # Componentes del calendario y modal de cita.
│   ├── 📁 auth/         # Componentes relacionados con la autenticación.
│   ├── 📁 contactform/    # Componente del formulario de contacto.
│   ├── 📁 dashboard/      # Componentes para el panel del barbero.
│   ├── 📁 diagnosis/      # Componente de diagnóstico de API.
│   ├── 📁 layout/         # Componentes del layout (Footer, Navbar).
│   └── 📁 services/       # Componentes para la página de servicios.
│
├── 📁 public/           # Recursos estáticos (imágenes, iconos, logos).
│
├── 📜 middleware.ts     # Middleware para gestionar rutas protegidas.
├── 📜 next.config.ts    # Configuración de Next.js (ej. dominios de imágenes).
└── 📜 tailwind.config.js # Configuración personalizada de Tailwind CSS.
```

4. Stack Tecnológico

- **Framework Principal:** Next.js 13+ (con App Router)
- **Librería de UI:** React 18+
- **Lenguaje:** TypeScript
- **Estilos:** Tailwind CSS
- **Autenticación:** Clerk (@clerk/nextjs) para la gestión de usuarios y sesiones.
- **Comunicación con Back-End:** Fetch API nativa del navegador para peticiones asíncronas.
- **Componentes de UI:**
  - **Calendario:** react-big-calendar
  - **Listas Desplegables:** @headlessui/react

    - **Iconos:** lucide-react
- **Back-End (Externo):** Consume exclusivamente la **API REST de Java + Spring Boot** que se ejecuta en http://localhost:8080.

5. Flujo de Datos: Reserva de una Cita

1. **Cliente (React):** Un usuario autenticado selecciona una sede, barbero, servicios y hora en la interfaz construida con componentes de React.
2. **Front-End (Next.js):** El componente del calendario (AppointmentCalendar.tsx) empaqueta los datos del formulario en un objeto JSON con los nombres de campo que espera la API de Java (ej. fechaInicio, barberId).
3. **Petición Fetch:** Se realiza una petición POST al endpoint de la API central: http://localhost:8080/citas-activas.
4. **Back-End (API de Java):** El AgendamientoController en Spring Boot recibe la petición, la convierte a un objeto NuevaCita, enriquece los datos (añadiendo nombreSede, etc.) y la persiste en la base de datos H2 a través de NuevaCitaRepository.
5. **Respuesta del Back-End:** La API devuelve el objeto NuevaCita completo y guardado en formato JSON, con un código de estado 200 OK.
6. **Actualización de UI (React):** El front-end recibe la respuesta. El estado de React se actualiza con la nueva cita, lo que provoca que la interfaz se re-renderice para mostrar el nuevo evento en el calendario del usuario y una notificación de éxito.