Documentación: Web 2 - Kingdom Barber

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

1. Resumen del Proyecto

La Plataforma Web Kingdom Barber (pi_web2) es la interfaz principal orientada al cliente y al personal de la barbería. Desarrollada como una aplicación moderna y de alto rendimiento con **Next.js (React y TypeScript)**, su función es ofrecer una experiencia de usuario fluida, rápida y responsiva para todas las interacciones del día a día.

Este proyecto funciona como un **front-end puro**; no contiene lógica de negocio crítica ni acceso directo a la base de datos. Toda la información que muestra y las acciones que realiza (agendar citas, subir imágenes a la galería, etc.) se comunican exclusivamente a través de peticiones HTTP a la **API Central de Java + Spring Boot (pi_movil2)**, que actúa como la única fuente de verdad.

La aplicación está desplegada globalmente en **Vercel**, asegurando una baja latencia y alta disponibilidad para los usuarios finales.

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 y profesional.
  - **Herramienta para el Barbero:** Brindar al personal un panel simple y protegido donde puedan visualizar su agenda de citas y gestionar su portafolio de trabajos.
  - **Desacoplamiento Total:** Funcionar como un cliente completamente independiente del back-end, asegurando que cualquier cambio en la API no requiera una reescritura completa del front-end.
  - **Diseño Responsivo:** Garantizar que la aplicación sea totalmente funcional y estéticamente agradable 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

basadas en el sistema de archivos y promueve la modularidad a través de componentes reutilizables.

```
pi_web2/
├── 📁 app/            # Directorio principal de enrutamiento y páginas.
│   ├── 📁 dashboard/
│   │   ├── 📁 barber/
│   │   │   └── 📜 page.tsx  # Vista de la agenda para el barbero.
│   │   └── 📁 gallery/
│   │       └── 📜 page.tsx  # Panel para gestionar la galería.
│   ├── 📜 (page.tsx)      # Página de inicio (Home).
│   ├── 📁 services/
│   │   └── 📜 page.tsx      # Página de listado de servicios.
│   ├── 📜 layout.tsx      # Layout principal que envuelve la aplicación.
│   └── 📜 globals.css     # Estilos globales de la aplicación.
│
├── 📁 components/       # Componentes de UI reutilizables.
│   ├── 📁 appointment/  # Componentes del calendario y modal de cita.
│   ├── 📁 auth/         # Componentes para inicio de sesión y perfil.
│   └── ...          # Otros componentes (Footer, Navbar, etc.).
│
├── 📁 public/           # Recursos estáticos (imágenes, logos).
│
└── 📜 middleware.ts     # Middleware para rutas protegidas y autenticación.
```

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 completa de usuarios, sesiones y rutas protegidas.
- **Comunicación con Back-End:** Fetch API nativa para peticiones asíncronas a la API externa.
- **Componentes de UI Notables:**
    - **Calendario:** react-big-calendar
    - **UI Primitivas:** @headlessui/react (para menús y modales accesibles).
    - **Iconos:** lucide-react
- **Plataforma de Despliegue:** Vercel

5. Flujo de Datos: Reserva de una Cita

1. **Cliente (React):** Un usuario autenticado con Clerk selecciona una sede, barbero, servicios y horario en la interfaz del calendario interactivo.
2. **Front-End (Next.js):** El componente AppointmentCalendar.tsx empaqueta los datos del formulario en un objeto JSON.
3. **Petición Fetch:** Se realiza una petición POST al endpoint de producción de la API central: **https://pi-movil2-0.onrender.com/citas-activas**.
4. **Back-End (API de Java):** El AgendamientoController en Spring Boot recibe la petición, la procesa y la persiste en la base de datos PostgreSQL.
5. **Respuesta del Back-End:** La API devuelve el objeto de la cita recién creada 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.

6. Despliegue y Puesta en Producción

- **Plataforma:** La aplicación está desplegada en **Vercel**, una plataforma optimizada para el despliegue de aplicaciones Next.js.
- **Despliegue Continuo (CI/CD):** Vercel está conectado al repositorio de GitHub del proyecto. Cada push a la rama principal (main) dispara automáticamente un nuevo "build" y despliegue de la aplicación.
- **Proceso de Build y Correcciones:** Durante el proceso de "build" en Vercel, se ejecutan verificaciones de calidad de código, incluyendo el linter **ESLint**. Fue necesario realizar ajustes en el código, como la **eliminación de importaciones no utilizadas** y la **corrección de dependencias en hooks de React (useEffect)**, para cumplir con las reglas del linter y permitir que el despliegue se completara con éxito.
- **Variables de Entorno:** La URL de la API de producción y las claves de Clerk se gestionan de forma segura a través de las variables de entorno de Vercel, nunca se exponen en el código fuente.
- **URL Pública:** La aplicación es accesible globalmente a través de la URL: **https://pi-web2-six.vercel.app**