

Documentación Técnica: KINGDOM BARBER (PI_WEB2)

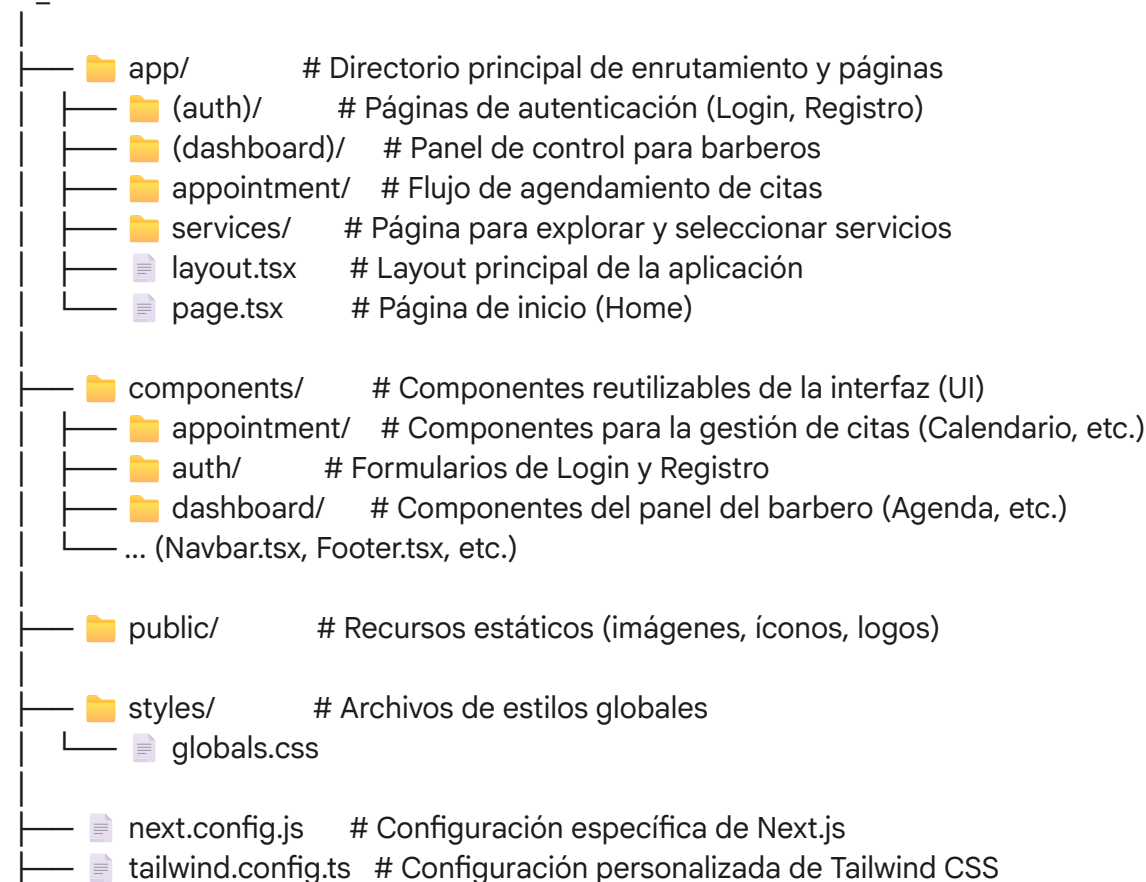
1. Resumen del Proyecto


Kingdom Barber es una plataforma web integral desarrollada con **Next.js (React + TypeScript)**, diseñada para modernizar y optimizar la gestión de citas y servicios de una barbería. La aplicación ofrece una experiencia de usuario fluida y diferenciada tanto para los clientes como para los barberos, facilitando la administración de agendas y la reserva de servicios en tiempo real mediante el consumo de una **API RESTful centralizada**.

2. Arquitectura y Estructura de Carpetas

El proyecto sigue una arquitectura modular basada en **Next.js App Router**, lo que garantiza un código mantenible, escalable y optimizado. Es importante destacar que este proyecto es puramente **frontend** y no contiene lógica de backend ni acceso directo a datos, ya que todo se delega a la API externa.

PI_WEB2/









└─  package.json # Dependencias y scripts del proyecto





3. Funcionalidades Principales

La plataforma se divide en dos grandes módulos según el tipo de usuario.

Para Clientes

-  **Autenticación de Usuarios:** Sistema de registro e inicio de sesión seguro.
-  **Sistema de Reservas Avanzado:** Calendario interactivo que consume la API para seleccionar sede, barbero, servicios y fecha/hora. Ofrece funcionalidades para crear, modificar y cancelar citas.
-  **Exploración de Servicios:** Catálogo detallado de los servicios ofrecidos, con descripciones, precios y duraciones, obtenidos desde la API.
-  **Geolocalización de Sedes:** Mapa interactivo para visualizar la ubicación de las sucursales.
-  **Formulario de Contacto:** Canal de comunicación directo que envía la información a la API.
-  **Página "Sobre Nosotros":** Sección informativa para conocer la historia, el equipo y la misión de Kingdom Barber.

Para Barberos y Administradores

-  **Acceso Exclusivo:** Panel de control personalizado y seguro tras iniciar sesión.
-  **Gestión de Agenda Personal:** Visualización clara de todas las citas asignadas (diarias, semanales, mensuales) obtenidas desde la API.
-  **Galería de Trabajos:** Sección para publicar un portafolio de cortes y estilos, con capacidad para subir y gestionar imágenes a través de la API.
-  **Acceso a Analíticas:** Integración con un dashboard externo para visualizar métricas clave de rendimiento.

4. Stack Tecnológico

- **Frontend:** Next.js 13+ (con App Router), React y TypeScript para un desarrollo robusto y tipado.
- **Estilos:** Tailwind CSS para un diseño rápido, responsivo y basado en utilidades.
- **Comunicación con Backend:** **Fetch API** o librerías como **Axios** para realizar peticiones asíncronas a la API REST central.
- **Backend (Externo):** Consume exclusivamente la **API REST de Node.js + Express** que se ejecuta en `http://localhost:3001`.
- **Despliegue:** Preparado para despliegue continuo en Vercel.

5. Conceptos Clave

- **Next.js:** Framework de React que ofrece renderizado del lado del servidor (SSR) y un potente sistema de enrutamiento, optimizando el SEO y el rendimiento.
- **React (Componentes):** Librería para construir interfaces de usuario mediante componentes reutilizables y un manejo de estado eficiente.
- **Tailwind CSS:** Framework utility-first que agiliza el desarrollo de la interfaz visual.
- **Consumo de API Externa:** La aplicación no tiene lógica de datos propia. Toda la información es solicitada a la API de Node.js, que actúa como única fuente de verdad.

6. Flujo de Datos: Reserva de una Cita

1. **Cliente (React):** El usuario selecciona los servicios, el barbero y la fecha en la interfaz construida con componentes de React.
2. **Frontend (Next.js):** El componente de reserva empaqueta los datos del formulario y realiza una petición POST al endpoint de la API central:
`http://localhost:3001/nuevas_citas`.
3. **Backend (API de Node.js):** El servidor Express recibe la petición, valida los datos y los persiste en su base de datos (en este caso, un archivo `nuevas_citas.json`).
4. **Respuesta del Backend:** La API devuelve una respuesta en formato JSON al frontend, confirmando que la cita se ha creado (ej: { "message": "Cita creada." }).
5. **Actualización de UI (React):** El frontend recibe la respuesta. El estado de React se actualiza, lo que provoca que la interfaz se re-renderice para mostrar un mensaje de confirmación y la nueva cita en el calendario del usuario.