

Documentación Técnica: KINGDOM BARBER (PI_WEB2)

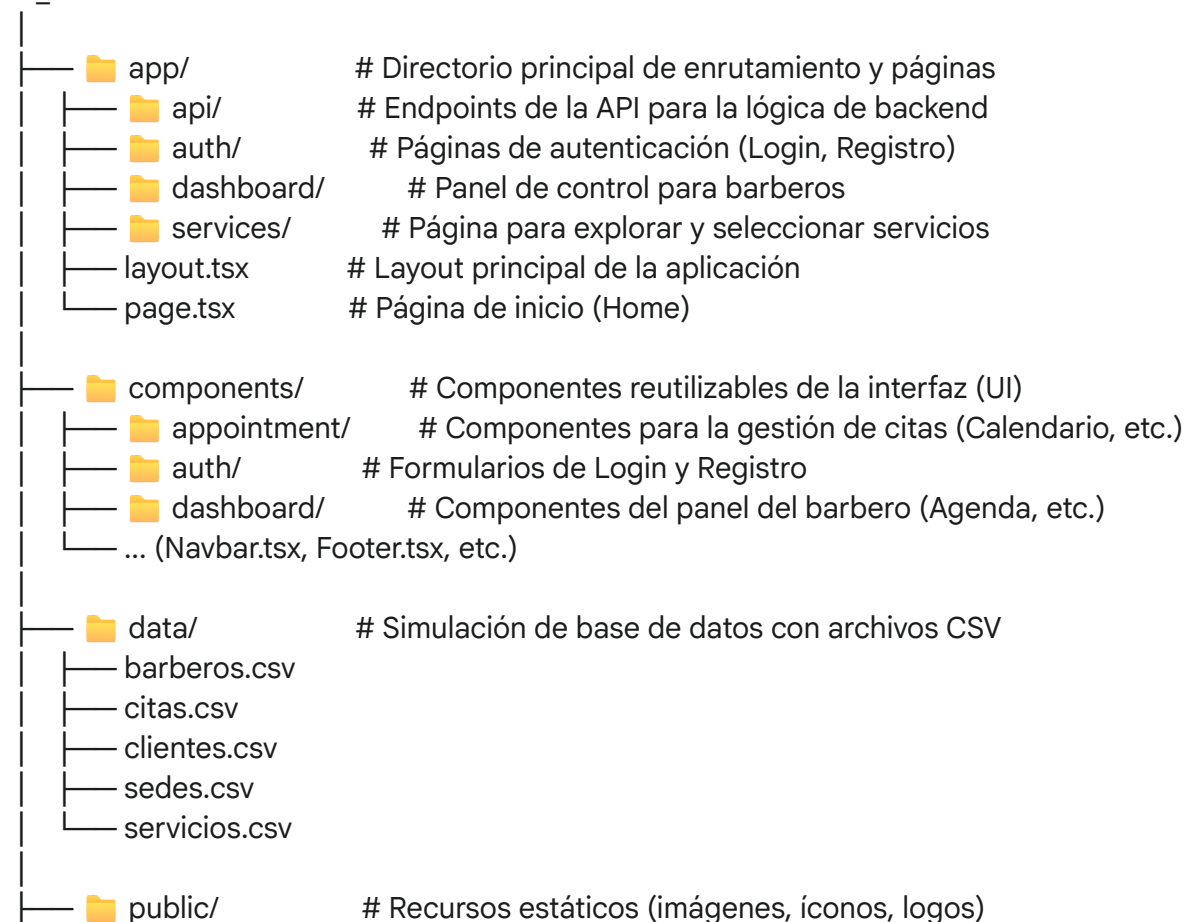
1. Resumen del Proyecto

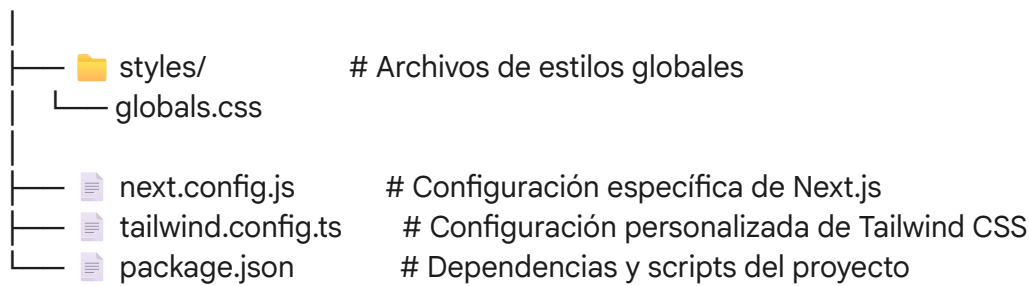
Kingdom Barber es una plataforma web integral desarrollada con **Next.js (React + TypeScript)**, diseñada para modernizar y optimizar la gestión de citas y servicios de una barbería. La aplicación ofrece una experiencia de usuario fluida y diferenciada tanto para los clientes como para los barberos,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

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 para el rendimiento. A continuación se detalla la estructura principal:

PI_WEB2/















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 para gestionar su perfil y citas.
-  **Sistema de Reservas Avanzado:** Calendario interactivo que permite 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.
-  **Geolocalización de Sedes:** Mapa interactivo para visualizar la ubicación de las sucursales y obtener indicaciones.
-  **Formulario de Contacto:** Canal de comunicación directo para consultas, sugerencias o soporte.
-  **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 como barbero.
-  **Gestión de Agenda Personal:** Visualización clara de todas las citas asignadas (diarias, semanales, mensuales) para una organización eficiente del trabajo.
-  **Galería de Trabajos:** Sección para publicar un portafolio de cortes y estilos, utilizando la optimización de imágenes de Next.js (`<Image />`).
-  **Acceso a Analíticas:** Integración con un dashboard externo (desarrollado en Python + Pandas) para visualizar métricas clave de rendimiento, como citas por día, servicios más solicitados, etc.

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.
- **Backend y Almacenamiento de Datos:**
 - **API Routes de Next.js** para la lógica del lado del servidor.
 - **Archivos CSV** como sistema de almacenamiento de datos simulado, gestionados con librerías de parsing en el backend.
- **Análisis de Datos (Integración):** Scripts de **Python** con **Pandas** para el procesamiento y visualización de datos.
- **Despliegue:** Preparado para despliegue continuo en **Vercel**.

5. Conceptos Clave

- **Next.js:** Framework de React que ofrece renderizado del lado del servidor (SSR), generación de sitios estáticos (SSG) y un potente sistema de enrutamiento basado en el sistema de archivos, optimizando el SEO y el rendimiento.
- **React (Componentes):** Librería para construir interfaces de usuario mediante componentes reutilizables, que son piezas de código aisladas (funciones o clases) que renderizan HTML dinámico.
- **Tailwind CSS:** Framework *utility-first* que proporciona clases de bajo nivel para construir diseños directamente en el HTML, agilizando el desarrollo y manteniendo la consistencia visual.
- **API (Application Programming Interface):** Conjunto de reglas que permite a diferentes aplicaciones comunicarse entre sí. En este proyecto, las API Routes de Next.js exponen los datos (ej. lista de barberos) en formato JSON para que el frontend los consuma.
- **CSV como Base de Datos Simulada:** En esta fase del proyecto, se utilizan archivos de texto plano con valores separados por comas (CSV) para almacenar la información. El backend se encarga de leer, escribir y modificar estos archivos para simular las operaciones de una base de datos real (CRUD).

6. Flujo de Datos: Reserva de una Cita

1. **Cliente:** Selecciona los servicios, el barbero y la fecha en la interfaz de React.
2. **Frontend (Next.js):** El componente de reserva empaqueta los datos del formulario y realiza una petición POST a un endpoint de la API (/api/citas).
3. **Backend (API Route):** El endpoint recibe la petición, valida los datos (disponibilidad del barbero, datos del cliente, etc.).
4. **Lógica de Datos:** El servidor lee el archivo citas.csv, agrega una nueva fila con la información de la nueva cita y guarda el archivo actualizado.
5. **Respuesta del Backend:** La API devuelve una respuesta en formato JSON al frontend, confirmando que la cita se ha creado ({ status: 'success', citald: '...' }) o informando de un error.
6. **Actualización de UI:** El frontend recibe la respuesta y actualiza la interfaz para mostrar un mensaje de confirmación y la nueva cita en el calendario del usuario.