Documentación: NTP - Kingdom Barber

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

1. Resumen del Proyecto

El Panel de Gestión de Kingdom Barber (pi_ntp) es una aplicación web de **Inteligencia de Negocios (BI)** desarrollada íntegramente en Python con el framework Streamlit. Su propósito es servir como la herramienta central de administración, consulta y análisis de datos para los gestores de la barbería.

El sistema ofrece potentes módulos de visualización de KPIs, gestión de datos históricos, un innovador **Asistente de Inteligencia Artificial** y un nuevo módulo de **Análisis de Datasets Públicos** para realizar estudios de mercado.

Es un **cliente de datos puro**: toda la información de la barbería es consumida en tiempo real desde la API Central de Java (pi_movil2), asegurando que los análisis reflejen siempre la única fuente de verdad del negocio. La aplicación está desplegada en **Streamlit Cloud**.

2. Objetivos del Proyecto

- **Objetivo Principal:** Proveer una herramienta de Business Intelligence (BI) completa y fácil de usar para la toma de decisiones estratégicas, centralizando la visualización, el análisis de los datos operativos de Kingdom Barber y la exploración de datos de mercado externos.
- **Objetivos Específicos:**
    - **Visualizar KPIs Clave:** Ofrecer un dashboard principal con los indicadores de rendimiento más importantes (ingresos, citas, rendimiento de barberos) de forma clara e interactiva.
    - **Facilitar la Consulta de Datos:** Proveer una interfaz para filtrar y buscar en el historial completo de citas, permitiendo una gestión detallada.
    - **Aprovechar la Inteligencia Artificial:** Integrar un asistente basado en IA (Google Gemini) capaz de generar reportes, responder preguntas en lenguaje natural y crear estrategias de marketing.
    - **Análisis de Mercado:** Implementar una herramienta para analizar y visualizar datasets públicos de datos.gov.co, permitiendo entender la distribución del sector de la belleza en Colombia.
    - **Desacoplamiento Total:** Funcionar como un cliente completamente independiente del back-end, consumiendo datos exclusivamente a través de la API de Java.

3. Stack Tecnológico

- **Lenguaje:** Python
- **Framework Web:** Streamlit

- **Procesamiento de Datos:** Pandas
- **Visualización de Datos:** Plotly Express
- **Generación de Reportes PDF:** FPDF2
- **Inteligencia Artificial:** Google Generative AI (Gemini)
- **Consumo de API:** Librería requests
- **Plataforma de Despliegue:** Streamlit Cloud

4. Arquitectura y Estructura de Carpetas

El proyecto sigue la estructura modular nativa de Streamlit, donde cada página de la aplicación es un archivo Python independiente ubicado en la carpeta pages.

```
pi_ntp/
├── 📁 .streamlit/
│   └── 📜 secrets.toml        # Almacena de forma segura las claves (API_URL, GOOGLE_API_KEY).
│
├── 📁 assets/                 # Recursos estáticos (imágenes, logos).
│
├── 📁 pages/                  # Cada archivo .py es una página de la aplicación.
│   ├── 📜 1_Dashboard.py          # Página principal con KPIs y visualizaciones.
│   ├── 📜 2_Gestion_de_Citas.py   # Página para consultar el historial de citas.
│   ├── 📜 3_Asistente_IA.py       # Suite de herramientas de Inteligencia Artificial.
│   └── 📜 4_Datasets_Reales.py    # Nuevo: Módulo de análisis de datos públicos.
│
├── 📜 inicio.py               # Punto de entrada y página de bienvenida de la aplicación.
├── 📜 data_manager.py         # Módulo centralizado para la comunicación con la API de Java.
├── 📜 report_generator.py     # Lógica para construir los reportes en formato PDF.
└── 📜 requirements.txt        # Lista de todas las dependencias de Python para el despliegue.
```

5. Módulos Principales

1. **Dashboard General (1_Dashboard.py)**
   - **Fuente de datos:** Endpoints de la API Central de Java.
   - **Métricas clave:** Ingresos totales, citas registradas, servicio más popular, barbero con mejor rendimiento.
   - **Gráficos interactivos:** Ingresos por servicio, carga de trabajo por barbero, y evolución de citas en el tiempo, todos construidos con Plotly.
2. **Gestión de Citas (2_Gestion_de_Citas.py)**
   - **Fuente de datos:** Endpoint /historial/citas de la API.
   - **Funcionalidades:** Permite realizar consultas detalladas con filtros por sede, barbero,

cliente y rango de fechas.

3. **Asistente de Inteligencia Artificial (3_Asistente_IA.py)**
    - **Fuente de datos:** Consume los mismos datos que el Dashboard, pero los procesa a través del modelo Gemini.
    - **Funciones principales:**
        - **Generador de Reportes:** Crea un informe ejecutivo en PDF con análisis de negocio.
        - **Analista de Datos Interactivo:** Un chatbot que responde preguntas sobre los datos.
        - **Asistente de Marketing:** Genera ideas y textos para campañas de marketing.
        - **Asesor de Estilo Virtual:** Recomienda cortes de cabello a partir del análisis de una imagen.
        - **Hazme un Nuevo Corte:** Herramienta para visualizar cortes de pelo sobre una imagen.
4. **Análisis de Datasets Reales (4_Datasets_Reales.py)**
    - **Fuente de datos:** Descarga y procesa archivos CSV del portal datos.gov.co.
    - **Funcionalidades:** Analiza y visualiza la distribución de peluquerías y salones de belleza en Colombia. Ofrece dashboards a nivel nacional, departamental y local con filtros interactivos y conclusiones automáticas que resumen los patrones encontrados en los datos.

6. Despliegue y Puesta en Producción

- **Plataforma:** La aplicación está desplegada en **Streamlit Cloud**.
- **Despliegue Continuo (CI/CD):** Streamlit Cloud está conectado al repositorio de GitHub. Cada push a la rama principal (main) dispara un nuevo despliegue automático.
- **Gestión de Secretos:** Las claves sensibles, como la GOOGLE_API_KEY y la API_URL (https://pi-movil2-0.onrender.com), se gestionan a través del sistema de **Secrets** de Streamlit.
- **Dependencias:** El archivo requirements.txt es utilizado por Streamlit Cloud para instalar todas las librerías de Python necesarias en el entorno de producción.
- **URL Pública:** La aplicación es accesible globalmente a través de la URL: **https://kingdombarberdashboard.streamlit.app/**.

7. Flujo de Datos Típico: Carga del Dashboard

1. **Usuario (Streamlit):** El usuario navega a la página del Dashboard.
2. **Front-End (1_Dashboard.py):** El script llama a la función obtener_vista_citas_completa() del módulo data_manager.py.
3. **Gestor de Datos (data_manager.py):**
    - Utiliza la librería requests para hacer peticiones GET a los endpoints de la API en Render.
    - Recibe las respuestas en formato JSON.
    - Usa **Pandas** para convertir cada respuesta JSON en un DataFrame y realizar las uniones (merge) necesarias.

4. **Respuesta al Front-End:** data_manager.py devuelve el DataFrame procesado y unificado al script 1_Dashboard.py.
5. **Actualización de UI (Streamlit):** El script del dashboard utiliza este DataFrame para alimentar los componentes de Plotly y st.metric, renderizando los gráficos interactivos en la interfaz.