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_ntp2.0) es una aplicación web de inteligencia de negocios desarrollada en **Python con Streamlit**. Su propósito es servir como la herramienta central de administración y análisis de datos para la barbería.

El sistema ofrece potentes módulos de visualización de KPIs, gestión de datos históricos y un innovador Asistente de Inteligencia Artificial. Es importante destacar que este proyecto es un **cliente de datos puro**; toda la información que procesa es consumida en tiempo real desde la **API Central de Java + Spring Boot**, asegurando que los análisis reflejen siempre la única fuente de verdad del negocio.

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 y el análisis de todos los datos operativos de Kingdom Barber.

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 (Gemini) capaz de generar reportes, responder preguntas en lenguaje natural y crear estrategias de marketing.
- **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
- **Back-End Consumido:** API REST de Java + Spring Boot (http://localhost:8080)

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.

```
pi_ntp2.0/
├── 📁 .streamlit/      # Configuración de Streamlit (secrets.toml).
├── 📁 assets/          # Recursos estáticos (imágenes, logos).
├── 📁 pages/           # Cada archivo .py es una página de la aplicación.
│   ├── 📜 1_Dashboard.py
│   ├── 📜 2_Gestion_de_Citas.py
│   └── 📜 3_Asistente_IA.py
│
├── 📜 inicio.py        # Punto de entrada y página de bienvenida de la app.
├── 📜 data_manager.py  # Módulo central para la comunicación con la API de Java.
├── 📜 report_generator.py# Lógica para crear los reportes en formato PDF.
└── 📜 requirements.txt  # Lista de todas las dependencias de Python.
```

5. Módulos Principales

1. Dashboard General (1_Dashboard.py)

Ofrece una vista global y gráfica del rendimiento del negocio.

- **Fuente de datos:** API Central de Java.
- **Métricas clave:** Ingresos totales, citas registradas, servicio popular, barbero top.
- **Gráficos interactivos:** Ingresos por servicio, carga de trabajo por barbero, ingresos por barbero y evolución de citas en el tiempo.

2. Gestión de Citas (2_Gestion_de_Citas.py)

Herramienta operativa para consultar y filtrar el historial completo de citas.

- **Fuente de datos:** API Central de Java (/historial/citas).
- **Funcionalidades:** Filtros por sede, barbero, cliente y rango de fechas; tabla de citas detallada y cálculo de ingresos según la selección.

3. Asistente de Inteligencia Artificial (3_Asistente_IA.py)

Suite de herramientas avanzadas que usan los datos filtrados para generar insights.

- **Fuente de datos:** API Central de Java (consume los mismos datos que el Dashboard).

- **Funciones principales:**
  - **Generador de reportes:** Crea un PDF con un análisis de negocio generado por IA.
  - **Analista de Datos Interactivo:** Un chatbot que responde preguntas sobre los datos generando y ejecutando código de Pandas.
  - **Asistente de Marketing:** Genera ideas y textos para campañas de marketing.
  - **Detector de Oportunidades:** Busca patrones específicos en los datos.
  - **Asesor de Estilo Virtual:** Recomienda cortes de cabello a partir de una imagen.

6. Flujo de Datos Típico: Carga del Dashboard

1. **Usuario (Streamlit):** El usuario abre la página del Dashboard.
2. **Front-End (1_Dashboard.py):** El script de la página 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 múltiples peticiones GET a la API de Java (ej. http://localhost:8080/historial/citas, http://localhost:8080/clientes, etc.).
   - Recibe las respuestas en formato JSON.
   - Usa la librería **Pandas** para convertir cada respuesta JSON en un DataFrame.
   - "Traduce" los nombres de las columnas de camelCase (Java) a Pascal_Case (esperado por el resto del código).
   - Realiza las uniones (merge) necesarias para crear una tabla de datos enriquecida.
4. **Respuesta al Front-End:** data_manager.py devuelve el DataFrame procesado al script 1_Dashboard.py.
5. **Actualización de UI (Streamlit):** El script del dashboard usa este DataFrame para alimentar los componentes de **Plotly Express** que renderizan los gráficos interactivos y las métricas en la interfaz de usuario.