🎭 Consola de Python – Sistema de Boletería con IA – Artechito

📘 DOCUMENTACIÓN TÉCNICA – Automatización n8n + Python

Repositorio: *https://github.com/CesarM2105/automatizacion-n8n*

📑 ÍNDICE

⭐ 1. Introducción técnica

Este repositorio implementa un cliente Python que se comunica con un flujo de n8n para ejecutar consultas inteligentes sobre una base de datos.
El objetivo del diseño es:

- enviar una consulta escrita en lenguaje natural,
- recibir la respuesta procesada por IA,
- mostrar los resultados en consola y/o generar archivos,
- manejar múltiples tipos de respuesta (texto, tabla, email, archivo).

El script fue construido para ser limpio, portable, extensible y robusto.

🧩 2. Arquitectura general del sistema

- Python actúa como interfaz de usuario.
- n8n actúa como motor lógico y automático.
- Supa Base almacena la información.

Usuario → Python → n8n (IA + SQL + Lógica) → Supabase → Respuesta a Python

---

🗂️ 3. Estructura del repositorio

automatizacion-n8n/

│── main.py # Programa principal de consola

│── modules/

│ └── api_client.py # Cliente HTTP para comunicarse con n8n

│── requirements.txt # Dependencias del proyecto

🧠 4. Explicación técnica del código

4.1 Diseño técnico del `main.py`

El archivo `main.py` implementa:

- una consola interactiva
- un procesador de tipos de respuestas
- un generador automático de Excel/CSV
- un renderizador profesional con Rich
- un detector de archivos, emails y tablas

- un loop principal robusto

A continuación se detalla cada módulo funcional.

🧱 A) Sistema de rutas de descarga

def obtener_ruta_descargas()

✔ Detecta automáticamente la carpeta disponible en Windows / Linux / Mac.

✔ Prioriza Desktop o Descargas.

✔ Esto garantiza portabilidad sin configuración manual del usuario.

🎨 B) Interfaz limpia en terminal

`banner()`

Limpia pantalla y muestra encabezado ASCII.

`bienvenida()`

Explica al usuario qué puede consultar.

Motivo:
→ User-friendly, profesional, guía inicial sin ver código.

📊 C) Representación de datos y exportación

`imprimir_tabla(datos)`

Convierte listas de diccionarios en una tabla visual con Rich.

`guardar_excel(datos)`

Genera un Excel usando **pandas**.

`guardar_csv(datos)`

Genera CSV con pandas.

Motivo:
✔ pandas es estándar industrial
✔ soporte estable para Excel
✔ serialización nativa de DataFrames
✔ reduce código manual

📩 D) Detección automática del tipo de respuesta

detectar_tipo_respuesta()

Analiza si la respuesta de n8n viene como:

- texto
- tabla
- email
- email con output
- archivo

Esto permite que el sistema actúe correctamente sin que n8n tenga que “avisar”.

Motivo de diseño:
✔ desacoplar Python del workflow
✔ n8n puede evolucionar sin cambios en el cliente

E) Procesador de respuesta

`procesar_respuesta()`

Este es el corazón del cliente Python.

Incluye:

- impresión de mensajes
- conversión JSON → Python
- tablas
- guardar Excel/CSV
- guardar archivos de email
- fallback seguro

Diseño pensado para:

✔ robustez
✔ evitar cierres inesperados
✔ lógica basada en tipo detectado
✔ soporte futuro para nuevos formatos

🔁 F) Loop principal

`main()`

Implementa:

- entrada del usuario
- cierre con palabras clave
- validación de cadena
- envío de request con `enviar_a_n8n()`
- display profesional con spinner
- ejecución continua

4.2 Módulo `api_client.py`

Es un cliente REST minimalista que encapsula:

```
requests.post()
```

```
auth=(USERNAME, PASSWORD)
```

✔ Autenticación básica
✔ Timeout configurable
✔ Manejo de errores unificado

Motivo:
Separar responsabilidades:

- `main.py` → lógica local y UI
- `api_client` → comunicación HTTP

Evita mezclar protocolos en la interfaz de usuario.

🔧 5. Librerías utilizadas y razones técnicas

Librería	Uso	Motivo
os	rutas, limpieza pantalla	Portabilidad multiplataforma
json	decodificar respuestas n8n	Respuestas vienen en JSON
time	time-stamp y spinners	Identificadores únicos de archivo
pandas	Excel/CSV	Estándar de industria, confiable
platform	detectar SO	Soporte automático Windows / Linux / Mac
dotenv	cargar variables .env	Seguridad y buenas prácticas
rich	tablas y UI profesional	Mejor experiencia visual

requests	cliente HTTP	Librería estándar, robusta
requests.auth	auth básica	Seguridad n8n

🛡️ 6. Manejo de errores

Incluye:

- errores HTTP
- errores de JSON inválido
- errores de archivo
- errores de escritura
- errores de entrada de usuario

Cada error se transforma en un mensaje legible:

```
return {"error": str(e)}
```

Evita que el programa “explote” y brinda retroalimentación clara.

🧬 7. Flujo completo Python → n8n → Supabase

1. El usuario escribe algo.
2. Python envía `{ "query": "..." }` al Webhook n8n.
3. n8n evalúa la intención con Gemini.
4. Genera SQL seguro y consulta Supabase.
5. Procesa la respuesta en JSON.
6. Puede generar archivo / mail / texto / tabla.
7. Python recibe el JSON final.
8. Python imprime tabla o genera Excel/CSV.

🧠 8. Decisiones técnicas importantes

✔ UI con Rich
→ Elegido por estética y facilidad en tablas.

✔ pandas para Excel/CSV
→ Solución profesional con soporte nativo.

✔ Separación en módulo `api_client`
→ Facilita cambios futuros en APIs.

✔ Detección inteligente de respuesta
→ Python no depende de estructura rígida.

✔ Sistema de rutas automático
→ Evita configuración manual por el usuario.

🚀 9. Posibles mejoras

- Configuración de rutas desde archivo YAML
- Cache local de respuestas
- Logs en archivo
- CLI con Typer
- Soporte para archivos PDF

👥 10. Créditos

Desarrolladores del código Python:
Mauricio Cuellar & César Mendoza