IOFE Mass Manager: Sistema de Automatización de Firmas Digitales

Automatización de flujos de trabajo masivos para la gestión de certificados y firmas digitales mediante integración API.

Resumen Ejecutivo

Este proyecto es una aplicación de escritorio desarrollada en **Python** diseñada para optimizar y automatizar la gestión del ciclo de vida de documentos en la plataforma de firmas digitales IOFEsign.

Nació como una iniciativa proactiva para solucionar un cuello de botella crítico en el departamento de operaciones de **Bureau Veritas (BV)**, transformando un proceso manual y tedioso en un flujo de trabajo automatizado, escalable y accesible para usuarios no técnicos.

El Problema (Contexto y Dolor Inicial)

La empresa gestiona la certificación de conversiones vehiculares para concesionarios que envían lotes masivos de expedientes (entre 50 y 100 vehículos por lote).

El proveedor del servicio de firmas digitales (IOFE) entregó una plataforma cuya funcionalidad estándar limitaba la carga de documentos a uno por uno.

Impacto negativo:

- **Ineficiencia Operativa:** Subir, firmar y descargar 100 expedientes manualmente consumía horas de trabajo repetitivo.
- **Riesgo de Error:** La gestión manual dificultaba el seguimiento de qué documentos habían sido cargados, o anulados.
- **Dependencia Técnica:** El equipo requería una solución ágil sin depender de tickets de soporte al proveedor para cargas masivas.

La Solución

A pesar de no ser parte de mis funciones principales, identifiqué la oportunidad de mejora y desarrollé una solución integral que interactúa directamente con la **API REST** del proveedor.

El sistema permite la gestión masiva (Batch Processing) de tres procesos clave:

1. **Carga Masiva:** Subida de cientos de PDFs a flujos de trabajo específicos.
2. **Anulación Masiva:** Cancelación de firmas basada en listados de Excel.
3. **Descarga Masiva:** Recuperación automática de documentos firmados.

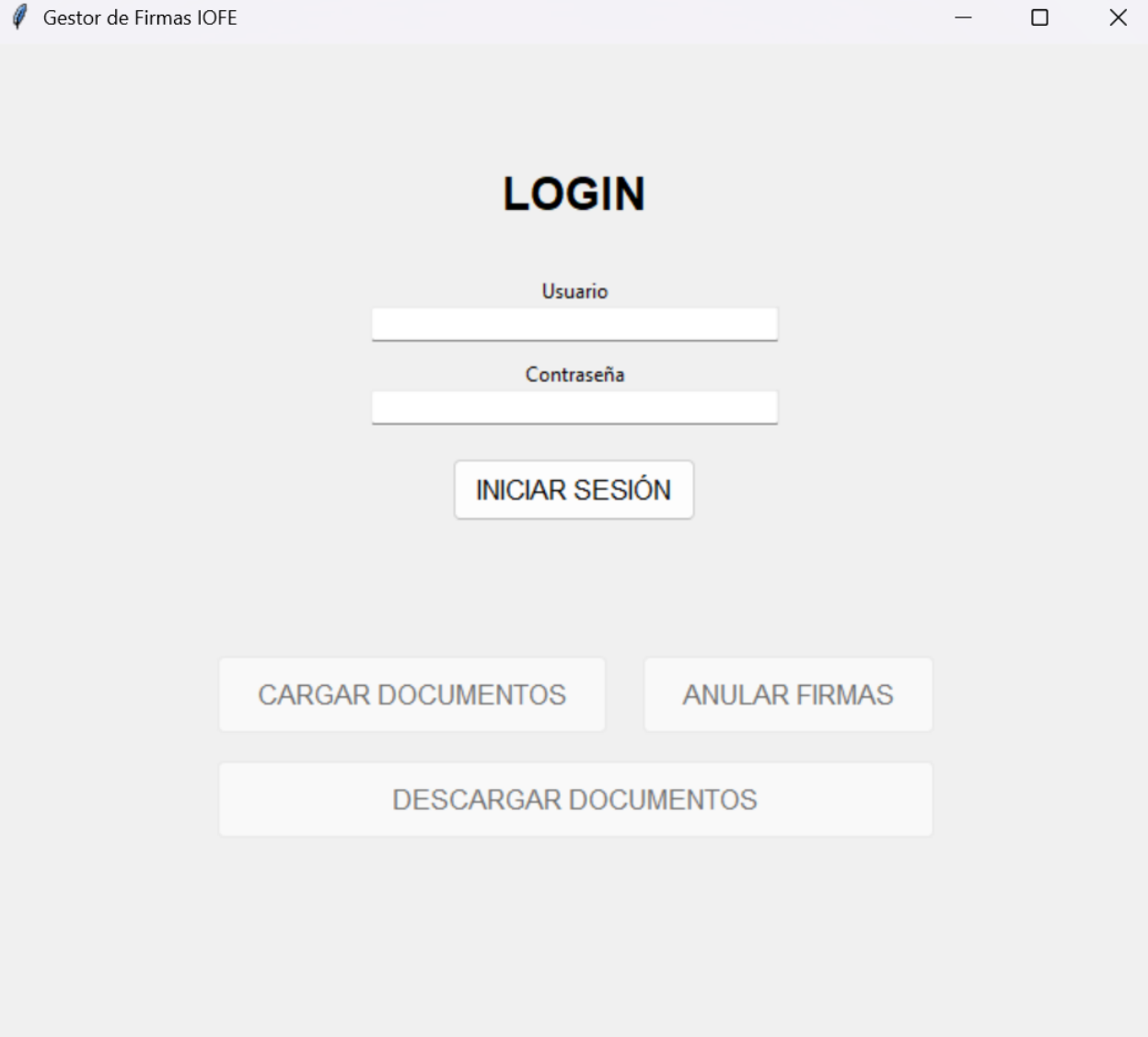
Arquitectura y Tecnologías

- **Lenguaje:** Python 3.x
- **Interfaz Gráfica (GUI):** Tkinter (Diseño intuitivo para usuario final).
- **Backend:** requests para comunicación HTTP/REST con la API de IOFE.
- **Manejo de Datos:** pandas y openpyxl para lectura/escritura de reportes en Excel.
- **Concurrencia:** Uso de threading para evitar congelamientos de la interfaz durante procesos largos.
- **Distribución:** PyInstaller para la generación de ejecutable (.exe).

Funcionalidades Clave

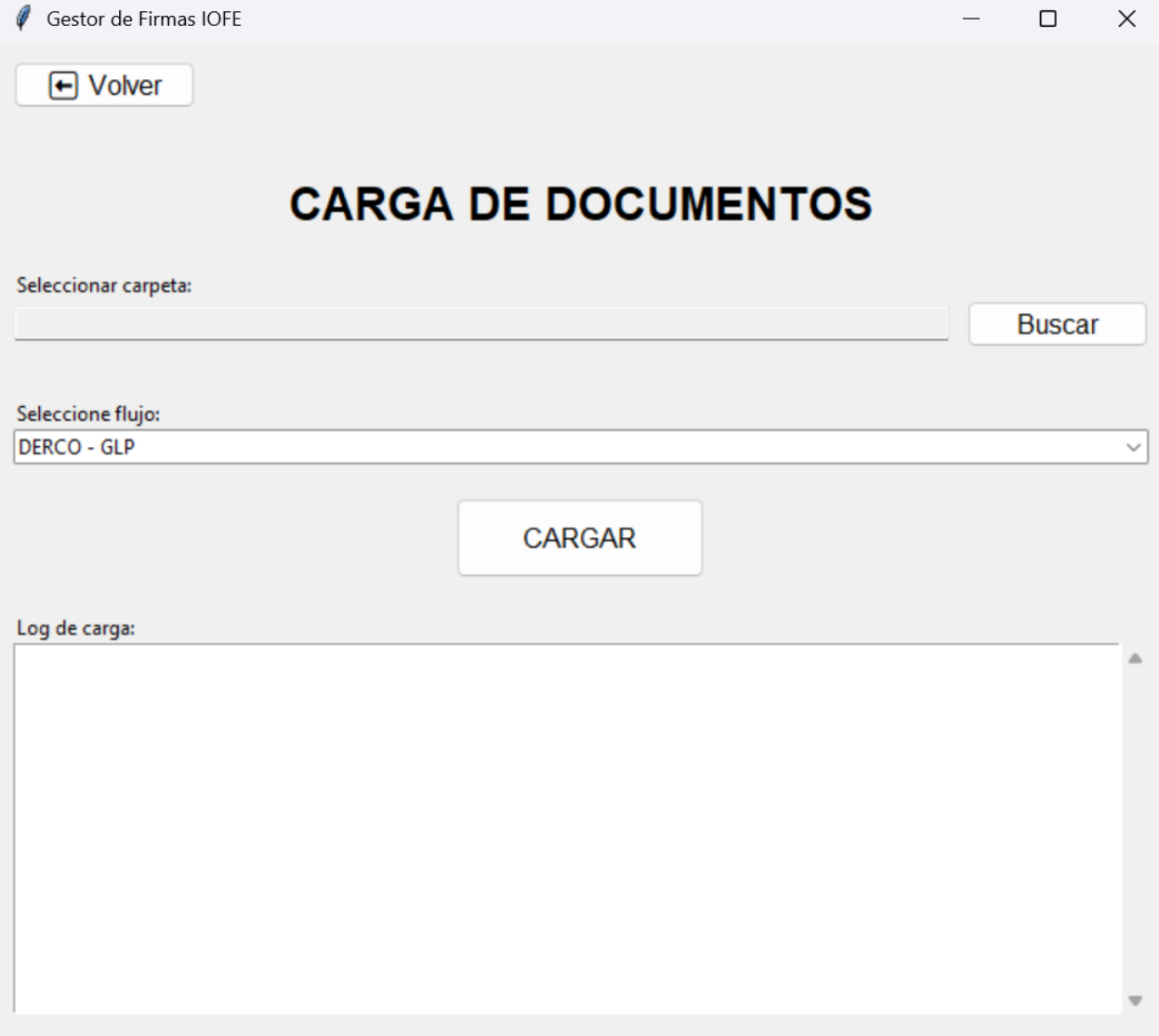
1. Interfaz de Usuario (GUI) y Autenticación

Un panel de control centralizado que gestiona el login contra la API, almacena el token de sesión de forma segura en memoria y habilita las funciones según el estado de la conexión.



2. Carga Inteligente de Documentos

- **Input:** Selección de carpeta local con archivos PDF.
- **Proceso:** El sistema itera sobre los archivos, los convierte a Base64 y los envía al endpoint de carga.
- **Registro en línea:** Añade el registro de cada documento cargado a un Google Sheets que actúa como base de datos, registrando el ID del documento, fecha, hash y estado.



3. Anulación y Descarga basada en Excel

- **Input:** Archivos Excel simples con IDs o Subjects de los documentos.
- **Proceso:** El sistema lee el Excel, valida los datos y ejecuta las peticiones PUT (para anular) o GET (para descargar) en bucle.
- **Feedback:** Barra de progreso y logs en tiempo real para informar al usuario del estado de cada operación.

Destacados Técnicos e Integraciones Avanzadas

Esta solución va más allá de un simple script de automatización; integra servicios en la nube, procesamiento concurrente y bases de datos en tiempo real para crear un ecosistema robusto.

Bot de Sincronización Cloud (Google Sheets Integration)

Implementación de un **bot** autónomo conectado a la API de Google Sheets (gspread + oauth2) que actúa como una base de datos viva y auditoría en tiempo real.

- **Funcionamiento:** Cada vez que el sistema carga un expediente a IOFEsign, el bot captura la metadata crítica (Subject/VIN, IOFE ID, Hash de seguridad, Timestamp) e inyecta una nueva fila en una hoja de cálculo maestra en la nube.
- **Valor Técnico:** Elimina la necesidad de una base de datos SQL local, permitiendo que múltiples usuarios y gerentes visualicen el estado de los lotes en tiempo real desde cualquier dispositivo, garantizando una trazabilidad del 100% sin intervención humana.

Orquestación de API RESTful & Seguridad

Diseño de un cliente HTTP robusto para interactuar con la infraestructura de IOFE.

- **Autenticación:** Manejo automatizado de Tokens JWT (Bearer), gestionando el ciclo de vida de la sesión (login, almacenamiento seguro en memoria, y uso en cabeceras).
- **Payloads Complejos:** Construcción dinámica de peticiones POST multipart/form-data para la transmisión de archivos binarios (PDFs convertidos a Base64) y metadatos JSON en una sola transacción.

Concurrencia y Multihilos (Threading)

Para garantizar una experiencia de usuario (UX) fluida, la aplicación implementa una arquitectura no bloqueante.

- **Hilos en Segundo Plano:** Las operaciones pesadas de red (carga/descarga de cientos de archivos) se ejecutan en *Daemon Threads* separados del hilo principal de la interfaz gráfica (Main Loop).
- **Resultado:** La interfaz nunca se "congela" (No Responde), permitiendo al usuario interactuar con la ventana, ver logs en tiempo real y actualizar la barra de progreso fluidamente mientras el "motor" trabaja intensamente en el backend.

Ingeniería de Datos (Pandas & Excel Automation)

Uso avanzado de la librería pandas para transformar archivos estáticos de Excel en flujos de comandos ejecutables.

- **Data Parsing:** El sistema ingesta hojas de cálculo complejas, limpia los datos (trimming, validación de tipos) y extrae los IDs necesarios para ejecutar anulaciones o descargas masivas, actuando como un puente inteligente entre los reportes administrativos y la API técnica.

Empaquetado y Legado (El valor del .exe)

Uno de los requerimientos autoimpuestos más importantes fue la usabilidad. El objetivo no era solo crear un script que yo pudiera correr, sino dejar una herramienta que el equipo pudiera utilizar de forma autónoma.

Para ello, utilicé **PyInstaller** para compilar todo el código y sus dependencias en un archivo ejecutable (.exe) portable.

Beneficio:

- Permite que cualquier miembro del equipo, sin conocimientos de programación ni Python instalado, pueda ejecutar la automatización.
- Garantiza la continuidad operativa del proceso tras mi salida de la empresa o rotación a otros proyectos.
- Elimina la barrera de entrada técnica para el uso de herramientas avanzadas.

Resultados e Impacto

- **Reducción de Tiempo:** Procesos de carga que tomaban horas se redujeron a minutos, limitados solo por la velocidad de internet. Se redujo el tiempo de procesamiento en aproximadamente un 80%.
- **Trazabilidad:** Creación de logs automáticos en Google Sheets que permiten auditar qué se subió y cuándo.
- **Autonomía del Equipo:** El departamento de operaciones ahora gestiona sus propios lotes masivos sin depender de TI o del proveedor.

Contacto

Anthony Valle Quinde *Ingeniero Mecatrónico | Desarrollador Python & Automatización*.