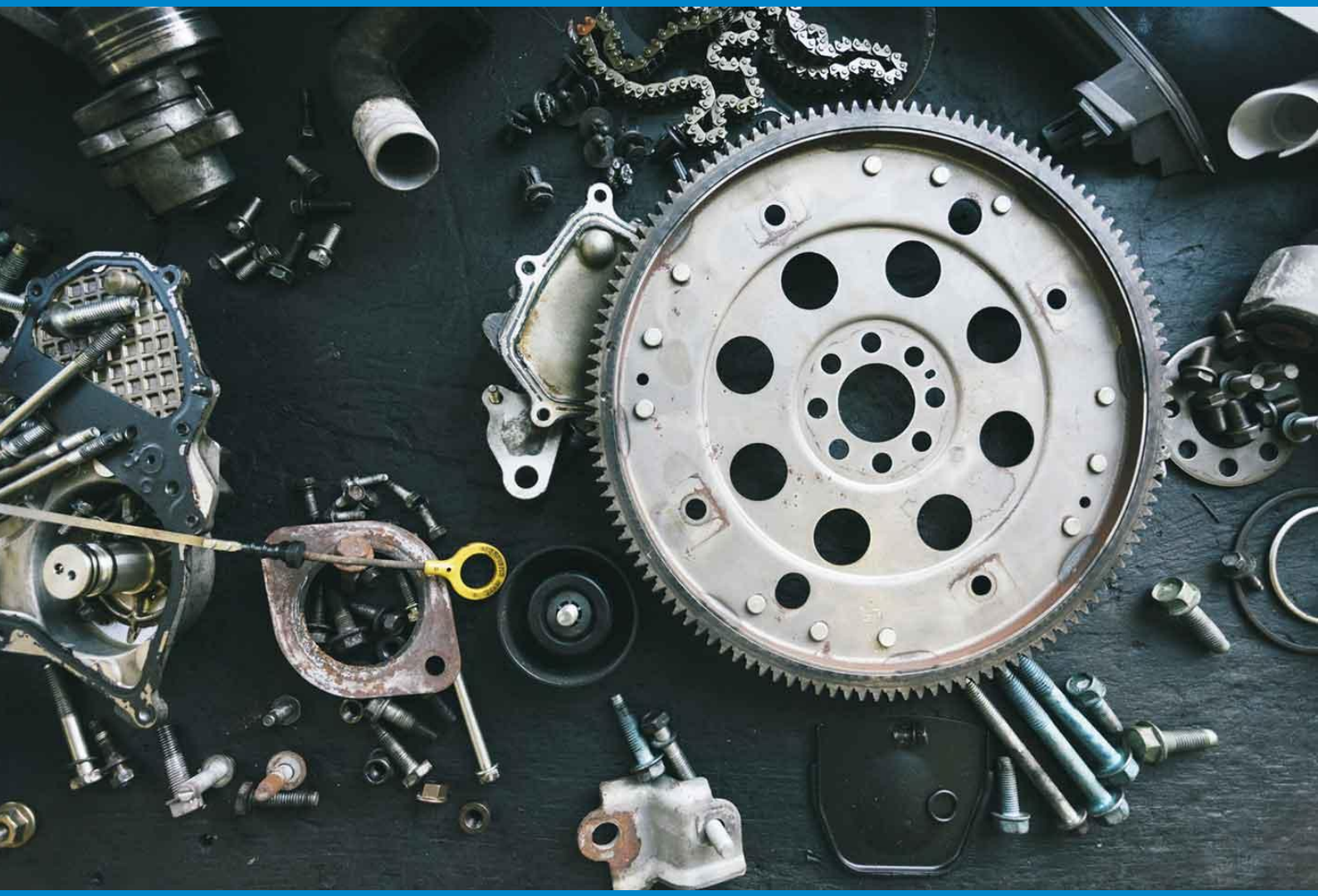

PAMI Automation

Documentacion Tecnica

Bioimágenes
CENTRO DE DIAGNÓSTICO

PAMI Automation

TABLA DE CONTENIDOS

I. INTRODUCCION

1.1 Proposito

Este documento describe la arquitectura técnica y el functionamiento interno de la aplicación PAMI Automation. Esta dirigido a los desarrolladores y/o usuarios técnicos que necesiten entender, mantener o modificar el código fuente de la aplicación.

El objetivo es proporcionar la informacion necesaria para:

- Comprender la estructura del código
- Realizar modificaciones y mantenimiento
- Construir nuevos ejecutables
- Resolver problemas técnicos comunes

1.2 Objetivos

Los objetivos tecnicos de esta documentacion son:

- Facilitar el mantenimiento del codigo fuente
- Reducir el tiempo de comprensión del sistema para nuevos desarrolladores
- Proporcionar guias claras para modificaciones comunes
- Documentar el proceso de build y despliegue

1.3 Requisitos Minimos

Software requerido:

- Python 3.8 o superior
- Acceso a internet para Google Sheets y PAMI
- Sistema operativo Windows (recomendable)

Archivos necesarios:

- credenciales_bio_sheets.json (credenciales internas para API de Google sheets)
- Permisos de escritura en carpetas de trabajo (logs, screenshots, downloads)

II. DESCRIPCION DEL SISTEMA

2.1 Descripcion general

Item	Description/Answer
Nombre del Proceso	PAMI Automation
Area de Aplicacion	Automatizacion Web
Departamento	-
Descripcion Corta	Automatiza la subida y transmisión de documentos médicos en el sistema OME.
Repositorio	https://github.com/FinochioM/PamiAuto (privado)
Lenguaje Principal	Python
Framework GUI	PyQt6
Motor de Automatizacion	Playwright

2.2 Aplicaciones Utilizadas

Nombre de Aplicacion	Version	Lenguaje	Tipo	Metodo de Acceso	Comentarios
PAMI	Web	-	Web	Navegador	Portal principal
Google Sheets	API	-	API	gspread	Input
PyQt6	6.6.0+	Python	Desktop	Nativo	Libreria Grafica
Playwright	1.40.0+	Python	Automatizacion	Chromium	Libreria de Automatizacion

2.3 Flujo de Automatizacion

La aplicacion sigue el siguiente flujo:

1- Conexión a Google Sheets: Lee los casos pendientes desde la hoja de calculo que el usuario elija (Por defecto “prestaciones_PAMI”)
2- Login automático: Se conecta a la pagina de PAMI con usuario y contraseña.
3- Navegacion: Accede al modulo OME y luego al Panel de prestaciones.
4- Procesamiento por lotes: Para cara caso:
    a. Busca por numero de documento (NDO)
    b. Verifica si el código PAMI coincide
    c. Descarga el informe desde la URL
    d. Sube el archivo al sistema
    e. Trasmite la información
5- Actualizacion: Marca los casos como procesados en Google Sheets.
6- Reportes: Genera logs en Excel con los resultados de la automatización.

2.4 Arquitectura del Sistema

```
app.py (GUI Principal)
├── browser_automation.py (Motor de automatización)
├── settings_manager.py (Gestión de configuración)
├── logger.py (Sistema de logging)
├── logs_window.py (Ventana de logs)
├── settings_window.py (Ventana de configuración)
└── config.py (Configuración global)
```

III. DESCRIPCION DE ARCHIVOS

3.1. Archivos principales

app.py:

- Archivo principal que contiene la interfaz gráfica (PyQt6)
- Maneja la ventana principal, campos de usuario/contraseña
- Coordina la ejecución de la automatización en un hilo separado
- Punto de entrada de la aplicación

browser_automation.py

- Contiene toda la lógica de automatización web usando Playwright
- Es el cerebro de la aplicación que:
    - Controla el navegador Chromium
    - Navega por las páginas de PAMI
    - Extrae datos de tablas HTML
    - Maneja modales de subida de archivos
    - Procesa cada caso individual

config.py

- Archivo de configuración que contiene:
    - URLs y credenciales de Google Sheets
    - Configuraciones del navegador (timeouts, directorios)
    - Rangos de fechas para filtrar casos
    - Funciones auxiliares para delays y validaciones
    - Selectores CSS para elementos web (WEB ELEMENT SELECTORS)

3.2. Configuracion y Gestion

settings_manager.py

- Maneja la configuración persistente de la aplicación
- Guarda y carga configuraciones del usuario en un archivo JSON
- Incluye directorios, timeouts y filtros de fecha

logger.py

- Sistema de logging que registra todas las actividades
- Genera reportes en Excel
- Organiza los resultados en hojas separadas: procesados exitosamente, fallidos, y ya procesados anteriormente

3.3. Interfaces de Usuarios

logs_window.py

- Ventana para visualizar archivos de log generados
- Permite abrir logs específicos y navegar a la carpeta de logs

settings_window.py

- Interfaz gráfica para modificar configuraciones
- Incluye timeouts del navegador, directorios de trabajo, filtros de fecha y credenciales de Google Sheets

IV. CONSTRUCCION Y DESPLIEGUE

4.1. Proceso para Buildear

Cuando se hagan cambios en la aplicación si se quiere ejecutar la aplicación con los cambios se debe generar un nuevo ejecutable (.exe). Para generar un nuevo ejecutable se debera ejecutar el siguiente archivo: "build.bat".

Este script (build.bat) hace lo siguiente:

- Verifica Python: Confirma que Python esté instalado
- Crea entorno virtual: Si no existe, crea uno nuevo
- Instala dependencias: Instala todas las librerías desde requirements.txt
- Descarga navegadores: Instala Chromium para Playwright
- Limpia builds anteriores: Elimina carpetas dist/ y build/
- Construye ejecutable: Usa PyInstaller con la configuración en pami_automation.spec
- Verifica resultado: Confirma que el .exe se creó correctamente
- El ejecutable final se genera en dist/PamiAutomation.exe y es completamente portable.

4.2. Configuracion Inicial

Antes del primer uso, asegurate de tener:

- Archivo de credenciales: credenciales_bio_sheets.json para acceso a Google Sheets
- Configuración de red: Acceso a internet para conectar con Google Sheets y PAMI
- Permisos de escritura: En las carpetas de logs, screenshots y downloads

4.3. Modificaciones Comunes

Modificar selectores web (si PAMI cambió su página):

1. Abre config.py
2. Busca la sección "WEB ELEMENT SELECTORS"
3. Actualiza los selectores CSS según los nuevos elementos de la página
4. Reconstruye el ejecutable con build.bat

Cambiar lógica de procesamiento:

- Modifica browser_automation.py en las funciones específicas como process_excel_data() o handle_file_upload_modal()

V. OTROS

5.1. Dependencias

VI. Dependencia (libreria)	Proposito
PyQt6	Interfaz Grafica
Playwright	Automatizacion del navegador
gspread	Conexion con Goolge Sheets
pandas	Manejo de datos y Excel
requests	Descarga de archivos PDF

Todas las dependencias están listadas en requirements.txt y se instalan automáticamente durante el build.