# Manual Técnico - Extractor de CUFE desde PDFs

## 1. Introducción

Este manual técnico proporciona una visión detallada de la arquitectura, funcionalidades y estructura del código del software 'Extractor de CUFE desde PDFs'. El propósito de esta aplicación es procesar archivos PDF, extraer información clave como CUFE, número de páginas y peso del archivo, y almacenarla en una base de datos SQLite.

## 2. Requisitos del Sistema

Para ejecutar esta aplicación, se requiere lo siguiente:  
- Sistema operativo: Windows, macOS o Linux  
- Python 3.13.2 instalado  
- Dependencias necesarias (pueden instalarse con `pip install -r requirements.txt`)

## 3. Instalación

1. Descargue el código fuente desde el repositorio ejecutando el siguiente comando:  
 ```bash  
 git clone https://github.com/nowen21/prueba-python-adres.git  
 ```  
2. Acceda al directorio del proyecto:  
 ```bash  
 cd prueba-python-adres  
 ```  
3. Instale las dependencias necesarias:  
 ```bash  
 pip install -r requirements.txt  
 ```  
4. Ejecute el programa con el siguiente comando:  
 ```bash  
 python extraeCUFE.py  
 ```

## 4. Arquitectura del Software

### 4.1 Descripción General

La aplicación sigue una arquitectura basada en componentes modulares, donde cada módulo tiene una responsabilidad específica. Se compone de los siguientes elementos:

### 4.2 Componentes Principales

1. \*\*Interfaz gráfica (Tkinter)\*\*: Gestiona la UI del programa y la interacción con el usuario.

2. \*\*Módulo de extracción de CUFE\*\*: Utiliza expresiones regulares para extraer el CUFE de los archivos PDF.

3. \*\*Gestión de base de datos (SQLite3)\*\*: Se encarga del almacenamiento y consulta de los datos extraídos.

4. \*\*Manipulación de archivos PDF (PyPDF2)\*\*: Se utiliza para extraer el contenido de los PDFs.

5. \*\*Lógica de validación\*\*: Previene duplicación de archivos y CUFEs en la base de datos.

## 5. Base de Datos

La aplicación almacena los datos en una base de datos SQLite. El esquema de la base de datos contiene una tabla llamada `facturas` con los siguientes campos:

```sql  
CREATE TABLE IF NOT EXISTS facturas (  
 id INTEGER PRIMARY KEY AUTOINCREMENT,  
 nombre\_archivo TEXT UNIQUE,  
 numero\_paginas INTEGER,  
 cufe TEXT UNIQUE,  
 peso\_archivo\_kb REAL  
);  
```

## 6. Funcionalidades Clave

### 6.1 Extracción de CUFE

La aplicación utiliza una expresión regular para extraer el CUFE de los archivos PDF. La expresión utilizada es:  
```python  
CUFE\_PATTERN = re.compile(r'\b([0-9a-fA-F]\n\*){95,100}\b')  
```

### 6.2 Prevención de Duplicados

Antes de insertar un nuevo registro en la base de datos, la aplicación verifica si el archivo o el CUFE ya han sido procesados previamente. Esto se logra mediante consultas SQL como:  
```python  
cursor.execute('SELECT 1 FROM facturas WHERE nombre\_archivo = ?', (nombre\_archivo,))  
```

## 7. Interfaz de Usuario

La interfaz gráfica se basa en Tkinter y utiliza `ttk.Treeview` para mostrar los registros en una tabla.

### 7.1 Botones Principales

1. \*\*📂 Seleccionar PDFs\*\*: Permite al usuario seleccionar archivos PDF para procesar.

2. \*\*📋 Cargar Registros\*\*: Muestra los datos almacenados en la base de datos.

3. \*\*🔄 Limpiar\*\*: Limpia la tabla sin afectar la base de datos.

4. \*\*📍 Ver Ubicación BD\*\*: Muestra la ruta del archivo SQLite.

5. \*\*❌ Salir\*\*: Cierra la aplicación de forma segura.

## 8. Manejo de Errores y Debugging

El programa incluye manejo de errores para evitar fallos inesperados. Algunos ejemplos incluyen:  
- Manejo de archivos inexistentes o corruptos.  
- Verificación de CUFE vacío.  
- Prevención de duplicados en la base de datos.

Si la aplicación falla, se recomienda revisar la consola para ver los mensajes de error y verificar que los archivos PDF sean legibles.

## 9. Conclusión

Este documento proporciona una guía completa sobre la estructura y funcionalidad del 'Extractor de CUFE desde PDFs'. Se recomienda mantener el código modular y realizar mejoras según los requerimientos del usuario.