

Nombre: Adin Cosme Misael Velasquez Reyes

Carne: 201903094

Elección de la IA

La elección de ChatGPT fue por ya saber como funciona el uso de esta ia y también por que es la única ia que conozco o mas bien he utilizado.

Proceso y prompts utilizados:

El primer prompt utilizado fue uno simple fue el siguiente:

crea un Sistema de cuenta corriente, que ingrese clientes, saldos iniciales, pagos y pueda imprimir tanto en la pantalla como en papel, listado de clientes con sus deudas. crealo con una programa que maneje bases de datos

luego le sugeri a la ia que creara su propio prompt y ue el siguiente

Al inicio, el sistema básico de cuenta corriente se implementó con un menú interactivo, cargando clientes desde XML, permitiendo registrar pagos, y generando listados de clientes. Durante esta etapa, los siguientes prompts fueron empleados:

- "Genera código para cargar clientes desde un archivo XML en una base de datos SQLite."
- "Añade una función para registrar pagos, pero verifica que el saldo no sea negativo."
- "Mejora el menú de interacción, añadiendo iconos y separadores."

Para los siguiente prompt copiaba el código y le sugería que agregara algo nuevo o modificara el código.

Tiempo de trabajo y límites:

La interacción con la IA se realizó en intervalos de aproximadamente 4 horas, antes de que la sesión se desconectara, lo cual ocurría después de 4-6 horas de uso continuo. Esto generaba pausas obligatorias, en ese tiempo justo me preparaba para ir a trabajar. Posteriormente, al regresar del trabajo a me permitía

continuar y dedicaba otras 3 horas, hasta alcanzar el límite de conexión nuevamente, momento en el que aprovechaba para descansar. Al despertar, la IA permitía reanudar la sesión,

Estrategias para continuar el flujo de trabajo:

para continuar con el flujo de trabajo no fue necesario utilizar otra cuenta por lo tanto no se perdía el flujo de trabajo entonces cuando continua trabajando la base de datos de la ia no se veía afectada

Problemas y fallas en el código:

para encontrar errores en el código solo se me facilitaba ejecutándolo o ya sea viendo los errores que marcaba visual studio code

también se me ocurrió copiar el código y preguntarle a la ia que errores podría tener el código y la ia me los proporcionaba.

Al concluir el proyecto, se logró un sistema de cuenta corriente robusto y usable, con funcionalidades esenciales para gestionar clientes, saldos, y deudas de manera eficiente. La colaboración con la IA no solo facilitó la escritura de código y resolución de problemas, sino que permitió optimizar el tiempo de trabajo, alternando entre periodos activos y pasivos según los límites de sesión. El uso de IA fue, sin duda, un catalizador para completar este sistema, proporcionando tanto orientación técnica como soporte para la mejora continua del proyecto.

Documentación del Sistema de Cuenta Corriente en Python

Descripción General

Este programa es un sistema de cuenta corriente para gestionar clientes, sus saldos y deudas. Permite cargar clientes desde un archivo XML, agregar nuevos clientes, registrar pagos y generar listados de clientes con sus deudas. El sistema utiliza una base de datos SQLite para almacenar los datos de los clientes, junto con la biblioteca Pandas para gestionar y presentar la información en formato tabular. Además, incorpora la biblioteca Tkinter para seleccionar archivos XML desde un diálogo de selección de archivos.

Características Principales

1. Cargar clientes desde un archivo XML.
2. Agregar nuevos clientes.
3. Registrar pagos de clientes.
4. Listar todos los clientes y sus deudas.
5. Imprimir el listado de clientes en pantalla y guardarlo en un archivo HTML.
6. Menú interactivo para gestionar las opciones de forma sencilla.

Requisitos

- Python 3.x
- Bibliotecas:
 - sqlite3: Para la gestión de la base de datos.
 - pandas: Para el manejo de tablas y exportación a HTML.
 - tabulate: Para la presentación tabular de datos en consola.
 - tkinter: Para la selección de archivos XML.
 - xml.etree.ElementTree: Para el procesamiento de archivos XML.

Instalación de Bibliotecas

Si no tienes instaladas las bibliotecas necesarias, puedes instalarlas ejecutando:

```
pip install pandas tabulate
```

Estructura del Código

1. Conexión a la Base de Datos

```
def connect_db():  
    conn = sqlite3.connect('cuenta_corriente.db')  
    return conn
```

2. Creación de la Tabla de Clientes

```
def create_table():  
    conn = connect_db()  
    cursor = conn.cursor()  
    cursor.execute('''  
        CREATE TABLE IF NOT EXISTS clientes (  
            id INTEGER PRIMARY KEY,  
            nombre TEXT NOT NULL,  
            saldo REAL NOT NULL,  
            deuda REAL NOT NULL  
        )  
    ''')  
    conn.commit()  
    conn.close()
```

3. Agregar Cliente

```
def agregar_cliente(nombre, saldo_inicial, deuda):  
    conn = connect_db()  
    cursor = conn.cursor()  
    cursor.execute('''  
        INSERT INTO clientes (nombre, saldo, deuda) VALUES (?, ?, ?)  
    ''', (nombre, saldo_inicial, deuda))  
    conn.commit()  
    conn.close()
```

Cargar Clientes desde Archivo XML

```
def cargar_clientes(archivo):  
    tree = ET.parse(archivo)  
    root = tree.getroot()  
  
    for cliente in root.findall('cliente'):  
        nombre = cliente.find('nombre').text  
        saldo = float(cliente.find('saldo').text)  
        deuda = float(cliente.find('deuda').text)  
        agregar_cliente(nombre, saldo, deuda)
```

Seleccionar Archivo XML

```
def seleccionar_archivo():  
    Tk().withdraw() # Ocultar la ventana principal de Tkinter  
    archivo = askopenfilename(title="Seleccionar archivo XML", filetypes=[("Archivos XML",  
    return archivo
```


6. Realizar Pago

7. Verificar si Hay Clientes

```
def hay_clientes():  
    conn = connect_db()  
    cursor = conn.cursor()  
    cursor.execute('SELECT COUNT(*) FROM clientes')  
    count = cursor.fetchone()[0]  
    conn.close()  
    return count > 0
```

8. Listar Clientes y sus Deudas

```
def listar_clientes():  
    conn = connect_db()  
    df = pd.read_sql_query('SELECT * FROM clientes', conn)  
    conn.close()  
    return df
```



9. Imprimir Listado de Clientes

```
def imprimir_listado():  
    if not hay_clientes():  
        print("\n⚠ Error: No hay clientes cargados. Por favor, cargue o agregue clientes")  
        return  
  
    df = listar_clientes()  
    print("\nListado de Clientes y Deudas:")  
    print(tabulate(df, headers='keys', tablefmt='pretty', showindex=False))  
    df.to_html('listado_clientes.html', index=False)  
    print("Listado guardado en listado_clientes.html")
```

10. Mostrar Menú de Opciones

```
def mostrar_menu():  
    print("\n" + "="*40)  
    print("          SISTEMA DE CUENTA CORRIENTE          ")  
    print("="*40)  
    print("♦ Opciones disponibles:")  
    print("  1. 📁 Cargar Clientes desde archivo XML")  
    print("  2. + Agregar Cliente")  
    print("  3. 💰 Realizar Pago")  
    print("  4. 📋 Listar Clientes")  
    print("  5. 🖨 Imprimir Listado")  
    print("  6. ✖ Salir")  
    print("="*40)
```

