Fecha: 30-enero-2026

No Más Fugas

Autor: Andrés Pérez Ulloa

1. Resumen

El proyecto No Más Fugas es un modelamiento back-end hecho con codificación Python para leer bases de datos en formato '.csv' sobre ingresos y egresos del hogar. Gracias a las funciones de cálculos de egresos, ingresos y saldo se pueden realizar análisis estadísticos sobre cómo se comportan los gastos en el hogar según integrante familiar, tipo de egreso y fecha.

El proyecto No Más Fugas se divide en 5 módulos Python denominados: main.py, menu.py, control_CSV.py, funciones_principales.py, validaciones.py. Tiene una carpeta llamada 'archivo_prueba' que contiene 2 archivos '.csv': prueba_egresos.csv y prueba_ingresos.csv. Finalmente tiene un archivo '.gitignore', que evita que se suban al proyecto 'github' los archivos temporales creados al momento de ejecutar el programa (Imagen 1 e Imagen 2).

Imagen 1 – Directorio Principal

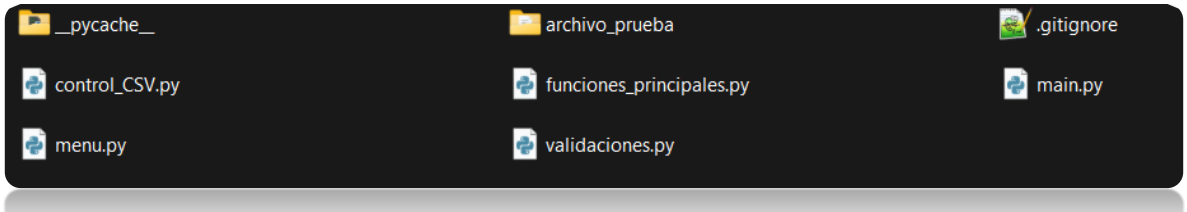
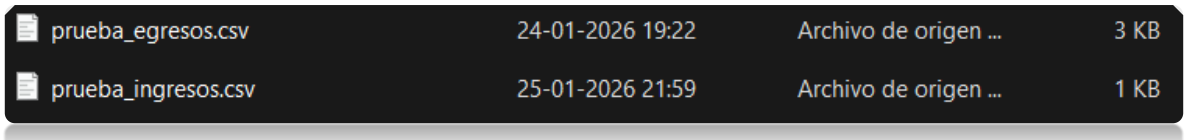


Imagen 2 – Carpeta archivo_prueba



2. Requerimientos

Softwares necesarios: Python

Bases de datos de prueba: Entregados en la carpeta pruebas como prueba_egresos.csv y prueba_egresos.csv

3. Estructuras de datos utilizadas

Diccionarios: En el módulo control_CSV.py, para traer los datos desde las bases de datos se usaron 2 funciones, cargar_egresos_familia y cargar_ingresos_familia. Estas funciones, al leer los datos, luego lo envían a un diccionario llamado egresos_familia e ingresos_familia, cuyas claves:valores son ID:diccionario, finalmente este diccionario tiene todas las otras calves:valores correspondiente a las otras columnas de datos. Ejemplo, egresos_familia = {'1': {'integrante': 'Mamá', 'tipo_egreso': 'luz', 'fecha': '2026-01-10', 'monto': 48500}, '2': {'integrante': 'Papá', 'tipo_egreso': 'agua', 'fecha': '2026-01-15', 'monto': 23200}, ...}

Listas: En el módulo funciones_principales.py, función total_egresos se define esta variable 'egresos = list (egresos_familia.values())', el cual, a partir de los diccionarios que está dentro del diccionario egresos_familia, se elabora una lista de diccionarios. Ejemplo egresos = [{'integrante': 'Papá', 'tipo_egreso': 'agua', 'fecha': '2026-01-15', 'monto': 23200}, {'integrante': 'Mamá', 'tipo_egreso': 'luz', 'fecha': '2026-01-10', 'monto': 48500}, ...]

Tupla: En el módulo funciones_principales.py, función obtener_variables_ingresos está esta variable 'integrantes = tuple(sorted(integrantes))', el cual asigna a la variable integrantes una tupla a partir de un conjunto ordenado alfabéticamente llamado integrantes. Esta tupla contiene lo siguiente integrantes = ('Hija', 'Hijo', 'Papá', 'Mamá')

Conjunto: En este proyecto se definieron varios conjuntos con la función set, pero a modo de ejempo, en el módulo funciones_principales.py, función obtener_variables_ingresos está la variable rango_meses = set(), el cual se llena mediante una iteración del diccionario principal. Solo va a contener valores que no se repitan del diccionario, en este caso sería por ejemplo rango_meses = {1,2,3,4,5,6}

Función Recursiva: En el módulo funciones_principales.py, función recursividad_ingresos_egresos se aplica la recursividad para calcular los valores totales, ya sea de ingreso o de egreso

4. Funcionamiento

El programa No Más Fugas consta de 5 módulos como se había comentado anteriormente, los cuales funcionan de la siguiente manera:

A- Módulo main.py

Es el encargado de cargar el módulo control_CSV y de abrir el módulo del menú. Gracias a la función para cargar CSV programada en el primer módulo, solo se

ingresa la ruta donde está el archivo como argumento de la función, lo que devuelve variables utilizables en el menú y operaciones posteriores.

El programa sólo funcionará si se ejecuta main.py

```python
# Importación de módulos
from menu import abrir_menu
from control_CSV import cargar_ingresos_familia as ingreso
from control_CSV import cargar_egresos_familia as egreso


def main ():

    # Carga de archivos
    egresos_familia = egreso ("archivo_prueba/prueba_egresos.csv")
    ingresos_familia = ingreso ("archivo_prueba/prueba_ingresos.csv")
    #Se llama la función abrir_menú desde módulo menu
    abrir_menu (egresos_familia, ingresos_familia)

#El software se va a ejecutar solo si está en main
if __name__ == "__main__":
    main()
```

B- Módulo menú.py

En este módulo se importa las funciones operacionales principales y la de validación de opciones. Se trae como argumento de la función abrir menú tanto el diccionario de egresos como el de ingresos. Se estilizó un poco la interfaz del menú para mostrar las opciones a escoger por el usuario. Se establecieron 3 opciones operacionales que llaman a funciones dentro del módulo funciones_principales y 1 opción para salir del programa. Las opciones operacionales son de calcular ingresos, egresos y saldo familiar. egresos y saldo familiar.

```python
# Importación de módulos
import funciones_principales
from validaciones import validar_opciones

# Apertura de menú
def abrir_menu (egresos_familia, ingresos_familia):
    print('==Bienvenido a No más Fugas==')
    print('')

    # Se establece variable sesión true para que cuando el menú
    # se cierre, sesión queda en False
    sesion = True

    while sesion:
        print('Escoja una de las siguientes opciones [1-4]: \n')
        print('1- Total ingresos')
        print('2- Total egresos')
        print('3- Saldo total')
        print('4- Salir')
        print('-'*30)
        opcion = input('Ingrese su opción: \n')
        print('-'*30)

        # Menu de opciones y llamada a las respectivas funciones de funciones_principales
        match opcion:
            case '1':
                funciones_principales.total_ingresos(ingresos_familia)
            case '2':
                funciones_principales.total_egresos(egresos_familia)
            case '3':
                funciones_principales.saldo_total(ingresos_familia, egresos_familia)
            case '4':
                print ('Gracias por su preferencia')
                sesion = False
            case _:
                validar_opciones(opcion) #Si la opcion no corresponde, va a llamar a esta función
```

C- Módulo control_CSV.py

Aquí se importó el módulo nativo de csv para poder realizar las lecturas de archivos .csv. Se definieron 2 funciones para leer los archivos y asignarlos en variables: cargar_egresos_familia y cargar_ingresos_familia. En la primera función se lee el archivo prueba_egresos, y almacena los datos en la variable egresos_familia como diccionario de diccionarios. Así mismo se realizó con la segunda función, solo que con la base de datos prueba_ingresos.

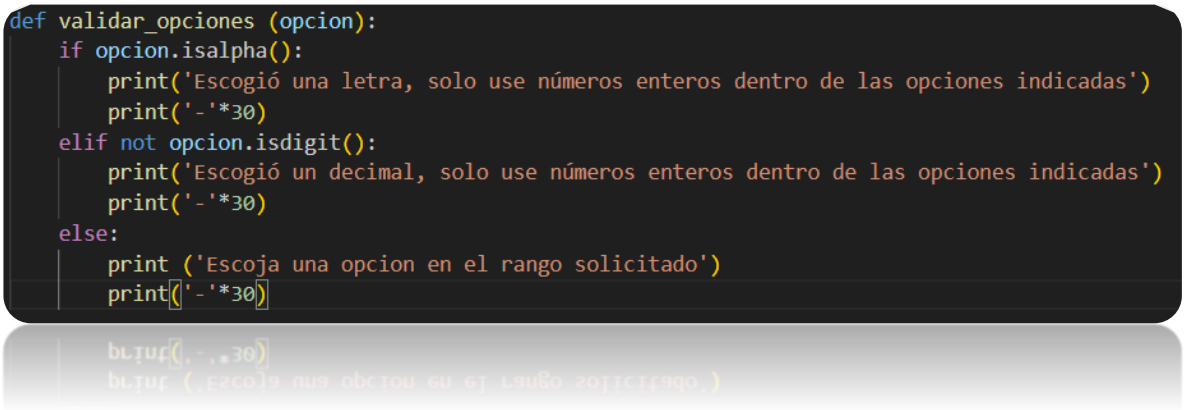
Fecha: 30-enero-2026

```python
# Importación de módulo nativo de Python
import csv

# Se definen 2 funciones tanto para egresos como ingresos
def cargar_egresos_familia (ruta):
    # Se define un diccionario para todos los datos traidos desde el .csv
    egresos = {}

    try:
        # Con la funcion with open se abre el archivo y se cierra al terminar de ejecutar
        with open (ruta, "r", encoding="UTF-8") as archivo:
            lector = csv.DictReader(archivo)
            for fila in lector:
                ID = fila["id_egreso"]
                egresos[ID] = {
                    "integrante": fila['integrante_familiar'],
                    "tipo_egreso": fila['tipo_egreso'],
                    "fecha": fila['fecha'],
                    "monto": int(fila['monto'])
                }
        return egresos
    except FileNotFoundError as e:
        print (f"El archivo {ruta} no existe:", e)
```

```python
def cargar_ingresos_familia (ruta):
    # Se define un diccionario para todos los datos traidos desde el .csv
    ingresos = {}

    # Mediante try/except se verifica que el archivo existe o no
    # Se le da al usuario un mensaje en caso de error
    try:
        # Con la funcion with open se abre el archivo y se cierra al terminar de ejecutar
        with open (ruta, "r", encoding="UTF-8") as archivo:
            lector = csv.DictReader(archivo)
            for fila in lector:
                ID = fila["id"]
                ingresos[ID] = {
                    "integrante": fila['integrante'],
                    "fecha": fila['fecha'],
                    "monto": int(fila['monto'])
                }
        return ingresos
    except FileNotFoundError as e:
        print (f"El archivo {ruta} no existe:", e)
```

D- Módulo validaciones.py

Corresponde a un módulo para validar las opciones ingresadas por el usuario en los diferentes menús mediante la función validar_opciones. Aquí se le indica al usuario cual fue el problema al ingresar su opción y como lo tiene que corregir.



```
def validar_opciones (opcion):
    if opcion.isalpha():
        print('Escogió una letra, solo use números enteros dentro de las opciones indicadas')
        print('-'*30)
    elif not opcion.isdigit():
        print('Escogió un decimal, solo use números enteros dentro de las opciones indicadas')
        print('-'*30)
    else:
        print ('Escoja una opcion en el rango solicitado')
        print('-'*30)
```

E- Módulo funciones_principales.py

Es el módulo principal, en donde están todas las operaciones relacionadas con la base de datos y entrega los resultados al usuario.

Se importó 2 módulos, uno nativo llamado datatime, para trabajar fechas y se importó el módulo personal validaciones.

El módulo funciones_principales consta de 6 funciones: obtener_variables_ingresos, obtener_variables_egresos, recursividad_ingresos_egresos, total_ingresos, total_egresos, saldo_total. Las funciones de obtener variables sirven para obtener variables específicas como fechas e integrantes, y asignarlo en grupos de datos más manejables. La función de recursividad es utilizado en total_ingresos para calcular el total de ingresos. Finalmente están las funciones operacionales de ingresos, egresos y saldos con cálculos específicos con filtros de datos, como separarlos por integrantes, fechas y tipo de egresos.

Fecha: 30-enero-2026

```python
def obtener_variables_egresos(egresos_familia):
    # Se definieron variables para preguntas personalizadas al usuario
    integrantes = set()  # variable se convertira a tupla despues
    rango_meses = set()  # se mantendrá como conjunto
    tipo_egreso = set()  # se mantendrá como conjunto
    for dato in egresos_familia.values():  # itera los valores del diccionario
        # añade a integrantes la itenracion de integrante
        integrantes.add(dato["integrante"])

        # añade a rango meses la iteracion de fechas
        fecha = datetime.strptime(dato["fecha"], "%Y-%m-%d")
        meses = fecha.month
        rango_meses.add(meses)

        # añade a tipo egreso la iteracion correspondiente
        tipo_egreso.add(dato["tipo_egreso"])
    rango_meses = sorted(rango_meses)  # Lo ordena alfabeticamente
    integrantes = tuple(sorted(integrantes))  # Lo ordena y transforma a tupla
    tipo_egreso = sorted(tipo_egreso)  # Lo ordena alfabeticamente
    return integrantes, rango_meses, tipo_egreso
```

```python
def obtener_variables_ingresos(ingresos_familia):
    # Se definieron variables para preguntas personalizadas al usuario
    integrantes = set()  # variable se convertira a tupla despues
    rango_meses = set()  # se mantendrá como conjunto
    for dato in ingresos_familia.values():  # itera los valores del diccionario
        integrantes.add(dato["integrante"])

        fecha = datetime.strptime(dato["fecha"], "%Y-%m-%d")
        meses = fecha.month
        rango_meses.add(meses)
    rango_meses = sorted(rango_meses)  # Lo ordena alfabeticamente
    integrantes = tuple(sorted(integrantes))  # Lo ordena y transforma a tupla
    return integrantes, rango_meses
```

```python
def recursividad_ingresos_egresos(montos, indice=0):
    # Cuando el índice llege al mismo valor que monto, va a terminar la función recursiva
    if indice == len(montos):
        return 0
    else:
        # Va a regresar el monto del indice y va a sumar el siguiente valor
        return montos[indice]['monto'] + recursividad_ingresos_egresos(montos, indice + 1)
```

Fecha: 30-enero-2026

```python
def total_ingresos(ingresos_familia):

    # Menu de opciones
    print("Escoja una de las siguientes opciones [1-3]: \n")
    print("1- Ingresos por integrante")
    print("2- Ingresos totales")
    print("3- Salir")
    print("-" * 30)
    opcion = input('Ingrese su opción: \n')
    print("-" * 30)

    match opcion:
        case "1":
            # Se obtiene solo los integrantes de la funcion con indice 0
            integrantes = obtener_variables_ingresos(ingresos_familia)[0]

            while True:
                print("Integrantes familiares disponibles: ")
                # itera en toda la lista de integrantes familiares
                for integrante in integrantes:
                    print(f"- {integrante}")
                opcion = input("Escoja el nombre del integrante o 'salir' para volver al menu: ")
                print("-" * 30)
                if opcion in integrantes:
                    # Se ejecuta solo si lo escrito por el usuario
                    # esta dentro de la lista de integrantes
                    total = 0
                    for dato in ingresos_familia.values(): #Itera sobre la lista
                        if dato["integrante"] == opcion:
                            total += dato["monto"]
                    print(f"El total de ingresos de {opcion} fue de ${total}")
                    print("-" * 30)
                    break
                elif opcion == "salir":
                    break
                else:
                    print("La opcion no esta dentro de la lista")
                    print("Intente nuevamente")
                    print("-" * 30)
        case "2":
            # Se calcula el total de ingresos familiares del csv con la función recursiva
            ingresos = list(ingresos_familia.values())
            total = recursividad_ingresos_egresos(ingresos)

            print(f"El total de ingresos fue de ${total}")
            print("-" * 30)
        case "3":
            print("Saliendo de la funcion")
        case _:
            # Esto es en caso de que escriba algo que no corresponda
            validaciones.validar_opciones(opcion)
```

Fecha: 30-enero-2026

```
total_egresos(egresos_familia):

# Menu de opciones
print("Escoja una de las siguientes opciones [1-3]: \n")
print("1- Egresos por tipo de egreso")
print("2- Egresos totales")
print("3- Salir")
print("-" * 30)
opcion = input('Ingrese su opción: \n')
print("-" * 30)

match opcion:
    case "1":
        # Se obtiene solo los integrantes de la funcion con indice 0
        tipo_egreso = obtener_variables_egresos(egresos_familia)[2]


        while True:
            # Se obtiene el listado de tipo de egresos disponibles
            print("Tipo de egresos disponibles: ")
            for tipo in tipo_egreso:
                print(f"- {tipo}")
            opcion = input("Escoja el tipo de egreso o 'salir' para volver al menu: ")
            print("-" * 30)
            # Se usa un condicional si la opcion está dentro de la lista de tipo de egreso
            if opcion in tipo_egreso:
                total = 0
                for dato in egresos_familia.values():
                    if dato["tipo_egreso"] == opcion:
                        total += dato["monto"]
                print(f"El total de egresos de {opcion} fue de ${total}")
                print("-" * 30)
                break
            elif opcion == "salir":
                break
            else:
                print("La opcion no esta dentro de la lista")
                print("Intente nuevamente")
                print("-" * 30)
    case "2":
        # Se calcula el total de egresos familiares del csv con la función recursiva
        egresos = list(egresos_familia.values())
        total = recursividad_ingresos_egresos(egresos)

        print(f"El total de egresos fue de ${total}")
        print("-" * 30)
    case "3":
        print("Saliendo de la funcion")
    case _:
        validaciones.validar_opciones(opcion)
```

Fecha: 30-enero-2026

```
def saldo_total(ingresos_familia, egresos_familia):
    # Menu de opciones
    print("Escoja una de las siguientes opciones (1-3): \n")
    print("1- Saldo por mes")
    print("2- Saldo total")
    print("3- Salir")
    print("-" * 30)
    opcion = input("Ingrese su opción: \n")
    print("-" * 30)

    match opcion:
        case "1":
            # Se obtiene un set de meses tanto del diccionario egresos como de ingresos
            meses_egresos = obtener_variables_egresos(egresos_familia)[1]
            meses_ingresos = obtener_variables_ingresos(ingresos_familia)[1]
            # Se combinan ambos sets, para que solo queden los que están en común
            meses_comunes = set(meses_egresos) & set(meses_ingresos)
            # Se realiza un listado de los meses del año para reemplazarlo por los valores numerales
            meses_anio = {
                1: "Enero",
                2: "Febrero",
                3: "Marzo",
                4: "Abril",
                5: "Mayo",
                6: "Junio",
                7: "Julio",
                8: "Agosto",
                9: "Septiembre",
                10: "Octubre",
                11: "Noviembre",
                12: "Diciembre",
            }
            # Finalmente se hace el diccionar clave, valor, donde valor es el mes en palabras
            meses_disponibles = {
                mes: meses_anio[mes] for mes in meses_anio if mes in meses_comunes
            }

            # Ahora entra a la funcion operacional de saldo por meses
            while True:
                # Se obtiene el listado de meses disponibles para el usuario
                print("Meses para consultar:")
                for mes, nombre in meses_disponibles.items():
                    print(f"{mes}. {nombre}")
                # Se le pregunta al usuario el mes a consultar
                opcion = input("Escoja el mes (numero) o salir para volver al menu:").strip().lower()
                print("-" * 30)
                # Si es 'salir', va a regresar al menú principal
                if opcion == 'salir':
                    print("Saliendo de saldos")
                    print("-" * 30)
                    break
                # Se realiza condicional para verificar que el input es entero
                elif opcion.isdigit():
                    # Se transforma a formato entero
                    opcion = int(opcion)
                    # Se verifica que la opción está dentro de la lista de meses
                    if opcion in meses_disponibles:
                        total_ingresos = 0
                        total_egresos = 0
                        for dato in ingresos_familia.values():
                            # Cuando se iteran en las fechas, hay que transformarlo primero en el formato
                            # fecha para realizar la comparación
                            fecha = datetime.strptime(dato["fecha"], "%Y-%m-%d")
                            # Aqui se compara para realizar la suma
                            if fecha.month == int(opcion):
                                total_ingresos += dato["monto"]
                        for dato2 in egresos_familia.values():
                            fecha2 = datetime.strptime(dato2["fecha"], "%Y-%m-%d")
                            if fecha2.month == int(opcion):
                                total_egresos += dato2["monto"]
                        saldo_mes = total_ingresos - total_egresos # Suma saldo total
                        print(f'El total de saldo en el mes de {meses_disponibles[int(opcion)]} fue de ${saldo_mes}')
                        print('-'*30)
                        break
                    else: # En el caso de que sea entero, pero no está en la lista
                        print("La opcion no esta dentro de la lista")
                        print("Intente nuevamente")
                        print("-" * 30)
                else:
                    print ("Ingrese una opción válida, intente nuevamente")
                    print("-" * 30)


        case "2":
            # Operación para obtener saldo entre ingresos y egresos
            egresos = 0
            ingresos = 0
            for dato in egresos_familia.values():
                egresos += dato["monto"]
            for dato2 in ingresos_familia.values():
                ingresos += dato2["monto"]
            saldo_total = ingresos - egresos
            print(f"El saldo total fue de ${saldo_total}")
            print("-" * 30)
        case "3":
            print("Gracias por su preferencia")
        case _:
            validaciones.validar_opciones(opcion)
```

5. Bases de datos

Son bases de datos a modo de ejemplo, solicitados a través de inteligencia artificial sobre ingresos y egresos en rangos de fechas específicas. Los archivos están dentro de la carpeta archivo_prueba y son prueba_ingresos.py y prueba egresos.py. Prueba egresos consta de 6 columnas con 50 entradas de datos y prueba ingresos de 4 columnas con 20 entradas de datos.

```
id_egreso,integrante_familiar,nombre_egreso,tipo_egreso,fecha,monto
1,Mamá,Cuenta luz enero,luz,2026-01-10,48500
2,Papá,Cuenta agua enero,agua,2026-01-15,23200
3,Mamá,Supermercado mes,alimentación,2026-01-20,165000
4,Hijo,Útiles escolares,escolaridad,2026-02-05,42000
5,Papá,Gas domiciliario,gas,2026-02-12,38000
6,Mamá,Cuenta luz febrero,luz,2026-02-10,46200
7,Papá,Cuenta agua febrero,agua,2026-02-15,22500
8,Mamá,Compra supermercado,alimentación,2026-02-22,158000
9,Hija,Mensualidad colegio,escolaridad,2026-03-01,95000
10,Papá,Recarga gas,gas,2026-03-08,41000
```

```
1   id,integrante,fecha,monto
2   1,Papá,2026-01-05,850000
3   2,Mamá,2026-01-10,620000
4   3,Papá,2026-02-05,850000
5   4,Mamá,2026-02-12,620000
6   5,Hijo,2026-02-20,120000
7   6,Papá,2026-03-05,900000
8   7,Mamá,2026-03-11,650000
9   8,Hija,2026-03-25,80000
10  9,Papá,2026-04-05,900000
```