Parcial 01. Programa Cobro y Facturación de Gasolina

Alejandra Magalí López Miranda, 201600085[1, *]

[1] *Escuela de Ingenieria Mecanica Electrica, Facultad de ingenieria , Universidad de San Carlos, Guatemala.*

(Dated: 26 de agosto de 2025)

I. INTRODUCCIÓN

El presente programa tiene como objetivo gestionar la facturación de una gasolinera, permitiendo registrar, almacenar y mantener un historial de ventas de combustible de manera eficiente. El sistema se desarrolla en Octave y Python, utiliza PostgreSQL como base de datos para el almacenamiento de la información. Además, mantiene un archivo de texto donde se registran todas las facturas generadas, facilitando la consulta rápida sin necesidad de acceder a la base de datos.

II. OBJETIVOS

A. General

- Desarrollar un sistema de cobro y facturación automatizado para una gasolinera, utilizando Python.

B. Específicos

* Diseñar e implementar un programa que solicite y valide los datos de entrada del usuario.

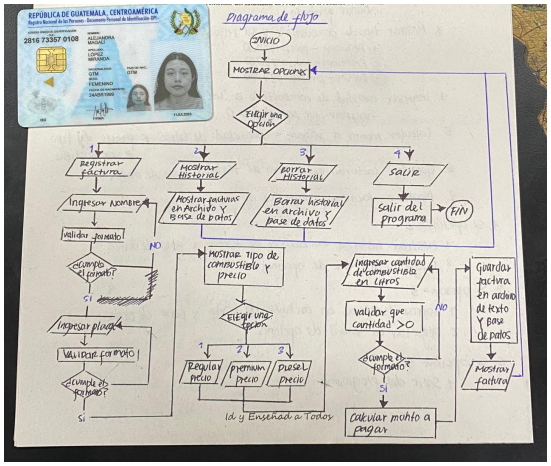
* Programar el cálculo automático del costo total de combustible aplicando las tarifas establecidas y generar facturas en un archivo de texto.

* Incorporar mecanismos de control de errores y almacenamiento de datos en PostgreSQL y archivos de texto, garantizando la integridad y disponibilidad de la información.

III. RESULTADOS

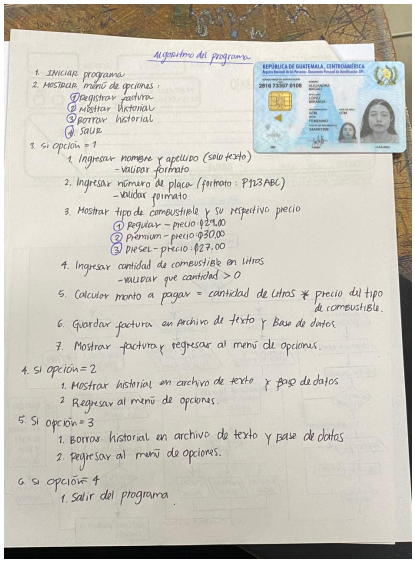
A. Diagrama de flujo

Figura 1



B. Algoritmo

Figura 2



* 2816733570108@ingenieria.usac.edu.gt

C. Pseudocodigo

Figura 3

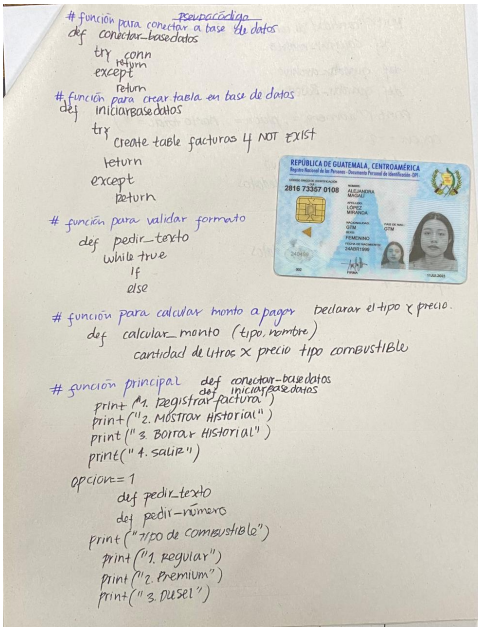
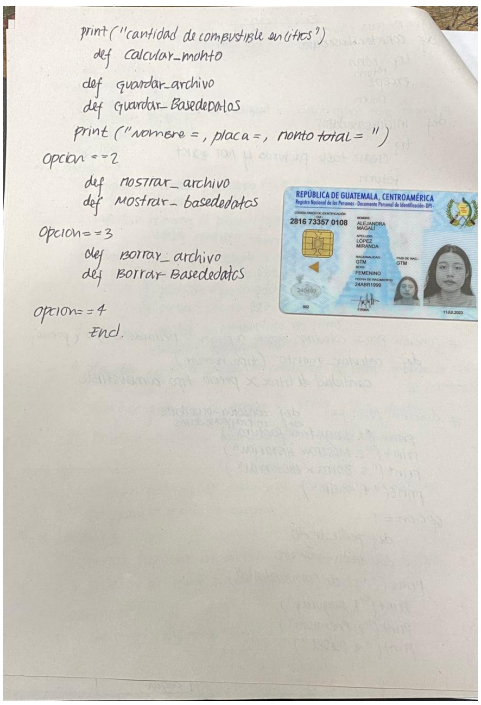


Figura 4



D. Codigo

Figura 5: Python

```python
import psycopg2
import re

#CONEXIÓN A BASE DE DATOS

def conectar_bd():
    try:
        conn = psycopg2.connect(
            "dbname=gasolinera user=postgres password=koala host=localhost port=5432"
        )
        return conn
    except Exception as e:
        print("Error al conectar con la base de datos:", e)
        return None

# CREAR TABLA EN BASE DE DATOS

def inicializar_bd():
    """Crea la tabla en la base de datos si no existe"""
    try:
        conn = conectar_bd()
        if conn:
            with conn.cursor() as cur:
                cur.execute("""
                    CREATE TABLE IF NOT EXISTS facturas_gasolina (
                        cliente TEXT NOT NULL,
                        placa TEXT NOT NULL,
                        combustible TEXT NOT NULL,
                        litros REAL NOT NULL,
                        total REAL NOT NULL
                    )
                """)
                conn.commit()
            conn.close()
    except Exception as e:
        print("Error inicializando base de datos:", e)
```

Figura 6: Octave

```
#Facturación Gasolinera

% Comprobar si es Octave
if (exist('OCTAVE_VERSION','builtin') ~= 0)
  pkg load signal;
  pkg load database;
end

% Ruta del archivo de facturas
archivo_facturas = "C:\\Users\\Alejandra\\Desktop\\980-Proyectos\\Parcial 01\\salida.txt";

% Conexión a la base de datos
conn = pg_connect(setdbopts( ...
    'dbname','gasolinera', ...
    'host','localhost', ...
    'port','5432', ...
    'user','postgres', ...
    'password','koala'));

% Crear tabla si no existe
create_table = [ ...
"CREATE TABLE IF NOT EXISTS facturas_gasolina (", ...
" cliente TEXT NOT NULL,", ...
" placa TEXT NOT NULL,", ...
" combustible TEXT NOT NULL,", ...
" litros REAL NOT NULL,", ...
" total REAL NOT NULL );" ];

pg_exec_params(conn, create_table);

%Funciones auxiliares

function nombre = pedir_nombre()
  valido = false;
  while ~valido
    nombre = strtrim(input("Ingrese nombre y apellido: ", "s"));
    if ~isempty(regexp(nombre, "^[A-Za-z]+( [A-Za-z]+)+$", "once"))
      valido = true;
    else
      disp("Formato inválido. Ejemplo: Juan Pérez");
    endif
  endwhile
```

E. Python

Figura 7: Creando base de datos 'gasolinera'

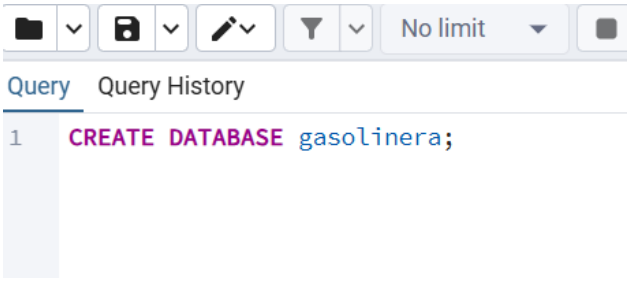


Figura 8: Registrar factura

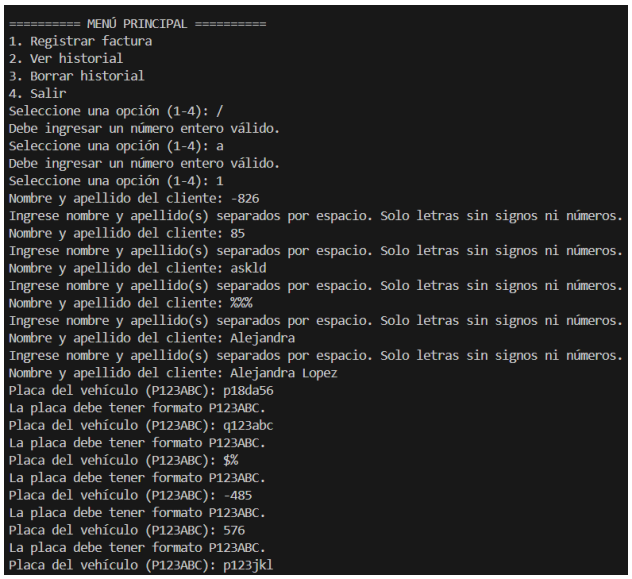


Figura 9: Tipos de combustible con su respectivo precio y factura generada

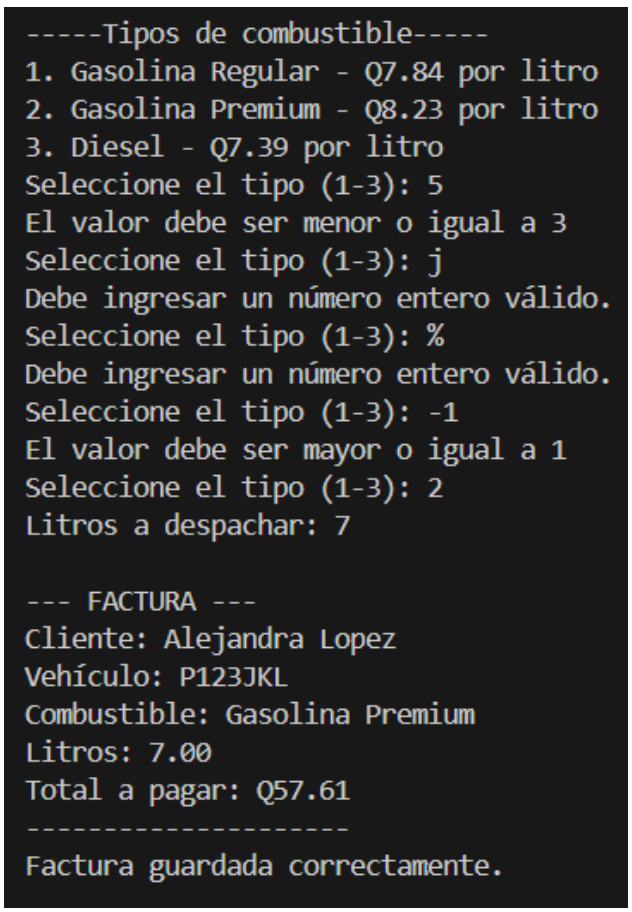


Figura 10: Historial en archivo y base de datos

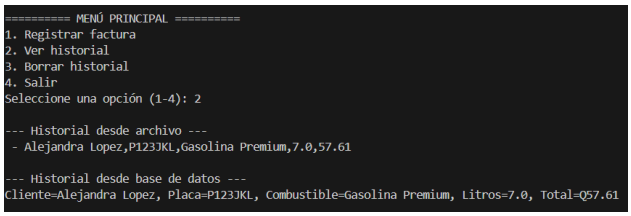
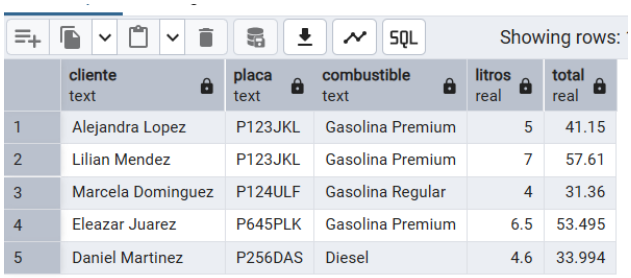


Figura 11: Tabla 'facturasgasolinera' en base de datos



	cliente text	placa text	combustible text	litros real	total real
1	Alejandra Lopez	P123JKL	Gasolina Premium	5	41.15
2	Lilian Mendez	P123JKL	Gasolina Premium	7	57.61
3	Marcela Dominguez	P124ULF	Gasolina Regular	4	31.36
4	Eleazar Juarez	P645PLK	Gasolina Premium	6.5	53.495
5	Daniel Martinez	P256DAS	Diesel	4.6	33.994

Figura 12: Borrar factura especifica

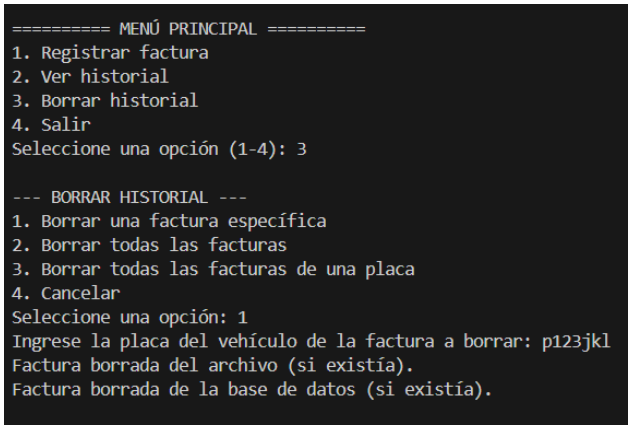
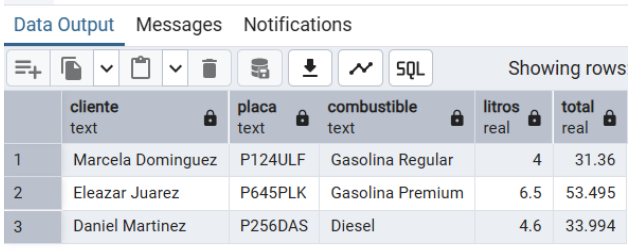


Figura 13: Borrar por placa



	cliente text	placa text	combustible text	litros real	total real
1	Marcela Dominguez	P124ULF	Gasolina Regular	4	31.36
2	Eleazar Juarez	P645PLK	Gasolina Premium	6.5	53.495
3	Daniel Martinez	P256DAS	Diesel	4.6	33.994

Figura 14: Historial borrado

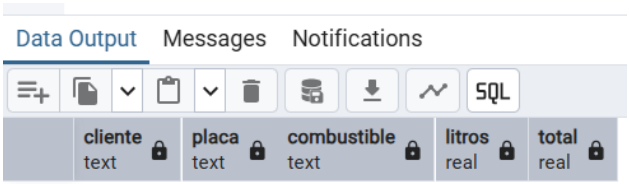


Figura 15: Cancelar borrado

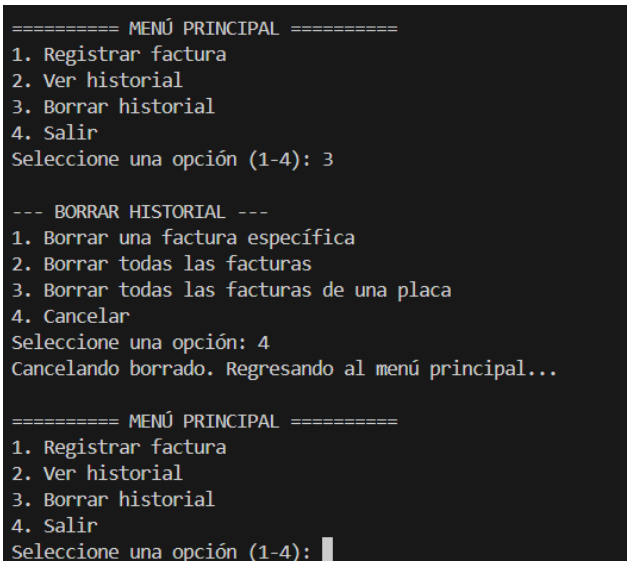
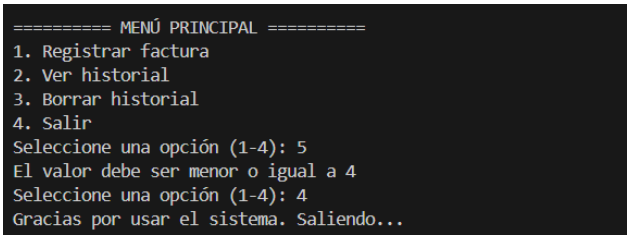
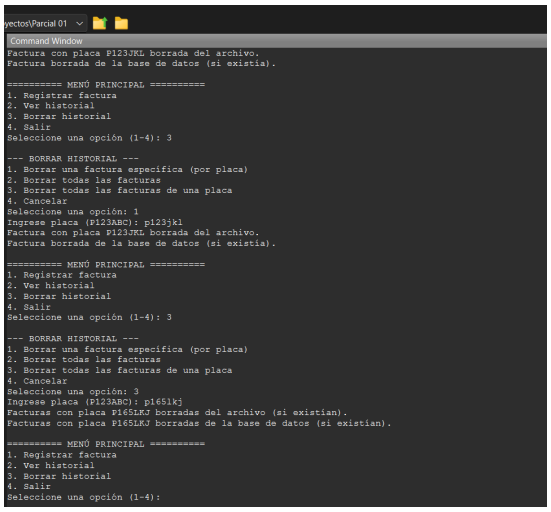


Figura 16: Salir del programa



F. Octave

Figura 17: Programa en Octave



IV. CONCLUSIONES

* La construcción del algoritmo permitió organizar paso a paso las operaciones necesarias, asegurando que la solución fuera comprensible antes de implementarla en código. Gracias a esto, se evita la duplicación de procesos.

* El pseudocódigo facilitó el diseño modular del programa. El uso de funciones con def permitió dividir el sistema en módulos específicos (validaciones, cálculos, guardado de datos y manejo del historial).

* Se incluyó validación en todas las entradas (nombre, placa, litros), lo que garantiza la calidad de los datos antes de almacenarlos en el archivo de texto o la base de datos PostgreSQL.

* Python ofrece una mayor facilidad de desarrollo, control de errores y manejo de bases de datos, mientras que Octave es viable pero más limitado en sintaxis y manejo de operaciones de bases de datos

V. ANEXOS

Link de repositorio: `https://github.com/AlejandraMLM/980-Proyectos.git`