

Parcial 01. Programa Cobro y Facturación de Gasolina

Alejandra Magalí López Miranda, 201600085^{1,*}

¹Escuela de Ingeniería Mecánica Eléctrica, Facultad de ingeniería, 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.

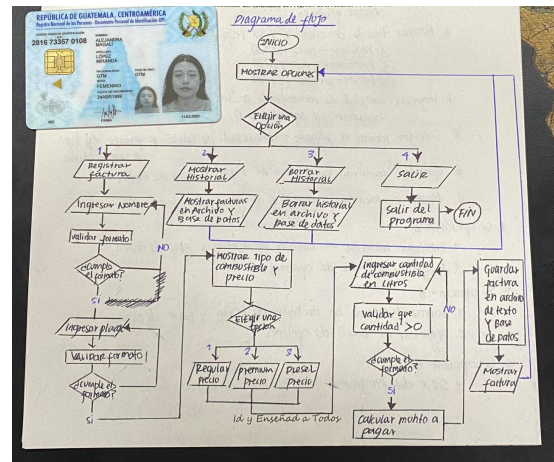
B. Específicos

- * 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

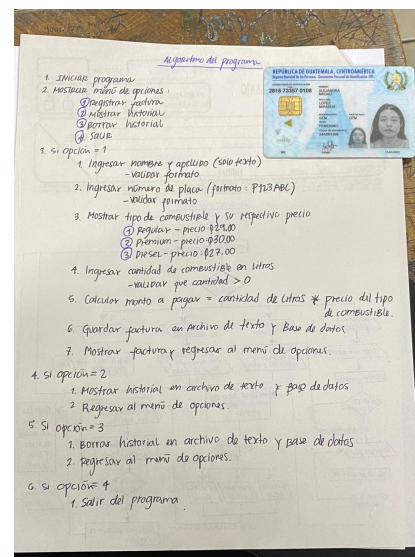
A. Diagrama de flujo

Figura 1



B. Algoritmo

Figura 2



* 2816733570108@ingenieria.usac.edu.gt

C. Pseudocodigo

Figura 3

```

#función para conectar a base de datos
def conectar_base_datos
    try:
        conn = psycopg2.connect(
            dbname='gasolinera', user='postgres', password='koala', host='localhost', port='5432'
        )
    except:
        return None
    return conn

#función para crear tabla en base de datos
def inicializar_base_datos
    try:
        create table facturas if not exist
    except:
        return None
    return conn

#función para validar formato
def pedir_texto
    while True:
        if:
            else

#función para calcular monto a pagar
def calcular_monto
    cantidad de litros x precio tipo combustible

#función principal
def conectar_base_datos
    print("1. Registrar factura")
    print("2. Mostrar historial")
    print("3. Borrar historial")
    print("4. salir")

opcion = 1
def pedir_texto
def pedir_numero
print("tipo de combustible")
print("1. Regular")
print("2. Premium")
print("3. Diesel")

```

Figura 4

```

print("cantidad de combustible en litros")
def calcular_monto
def guardar_archivo
def guardar_base_datos
print("numero = , placa = , monto total = ")

opcion == 2
def mostrar_archivo
def mostrar_base_datos

opcion == 3
def borrar_archivo
def borrar_base_datos

opcion == 4
End

```

D. Codigo

Figura 5: Python

```

1 import psycopg2
2 import re
3
4 #CONEXIÓN A BASE DE DATOS
5
6 def conectar_bd():
7     try:
8         conn = psycopg2.connect(
9             dbname='gasolinera', user='postgres', password='koala', host='localhost', port='5432'
10        )
11        return conn
12    except Exception as e:
13        print("Error al conectar con la base de datos:", e)
14        return None
15
16 # CREAR TABLA EN BASE DE DATOS
17
18 def inicializar_bd():
19     """Crea la tabla en la base de datos si no existe"""
20     try:
21         conn = conectar_bd()
22         if conn:
23             with conn.cursor() as cur:
24                 cur.execute("""
25                     CREATE TABLE IF NOT EXISTS facturas_gasolina (
26                         cliente TEXT NOT NULL,
27                         placa TEXT NOT NULL,
28                         combustible TEXT NOT NULL,
29                         litros REAL NOT NULL,
30                         total REAL NOT NULL
31                     )
32                 """)
33             conn.commit()
34             conn.close()
35     except Exception as e:
36         print("Error inicializando base de datos:", e)
37

```

Figura 6: Octave

```

1 if facturacion Gasolinera
2
3 # Comprobar si es Octave
4 if (exist('OCTAVE_VERSION','builtin') == 0)
5     pkg load signal;
6     pkg load database;
7 end
8
9 # Ruta del archivo de facturas
10 archivo_facturas = "C:\Users\Alejandra\Desktop\980-Proyectos\Parcial 01\valida.txt";
11
12 # Conexión a la base de datos
13 conn = pg_connect(testdboptel ...
14     'dbname','gasolinera', ...
15     'host','localhost', ...
16     'port','5432', ...
17     'user','postgres', ...
18     'password','koala');
19
20 # Crear tabla si no existe
21 create_table = { ...
22     "CREATE TABLE IF NOT EXISTS facturas_gasolina (" ...
23     "cliente TEXT NOT NULL, ...
24     "placa TEXT NOT NULL, ...
25     "combustible TEXT NOT NULL, ...
26     "litros REAL NOT NULL, ...
27     "total REAL NOT NULL);" };
28
29 pg_exec_params(conn, create_table);
30
31 #Funciones auxiliares
32
33 function nombre = pedir_nombre()
34     valido = false;
35     while ~valido
36         nombre = strtrim(input("Ingrese nombre y apellido: ", "s"));
37         if ~isempty(regexprep(nombre, "[A-Za-z]+([A-Za-z]+)+", "none"))
38             valido = true;
39         else
40             disp("Formato inválido. Ejemplo: Juan Pérez");
41         end
42     endwhile
43 end
44

```

E. Python

Figura 7: Creando base de datos 'gasolinera'

```

Query Query History
1 CREATE DATABASE gasolinera;

```

Figura 8: Registrar factura

```
===== MENÚ PRINCIPAL =====
1. Registrar factura
2. Ver historial
3. Borrar historial
4. Salir
Seleccione una opción (1-4): /
Debe ingresar un número entero válido.
Seleccione una opción (1-4): a
Debe ingresar un número entero válido.
Seleccione una opción (1-4): 1
Nombre y apellido del cliente: -826
Ingrese nombre y apellido(s) separados por espacio. Solo letras sin signos ni números.
Nombre y apellido del cliente: 85
Ingrese nombre y apellido(s) separados por espacio. Solo letras sin signos ni números.
Nombre y apellido del cliente: askld
Ingrese nombre y apellido(s) separados por espacio. Solo letras sin signos ni números.
Nombre y apellido del cliente: %%%
Ingrese nombre y apellido(s) separados por espacio. Solo letras sin signos ni números.
Nombre y apellido del cliente: Alejandra
Ingrese nombre y apellido(s) separados por espacio. Solo letras sin signos ni números.
Nombre y apellido del cliente: Alejandra Lopez
Placa del vehículo (P123ABC): p18da56
La placa debe tener formato P123ABC.
Placa del vehículo (P123ABC): q123abc
La placa debe tener formato P123ABC.
Placa del vehículo (P123ABC): %%
La placa debe tener formato P123ABC.
Placa del vehículo (P123ABC): -485
La placa debe tener formato P123ABC.
Placa del vehículo (P123ABC): 576
La placa debe tener formato P123ABC.
Placa del vehículo (P123ABC): p123jkl
```

Figura 10: Historial en archivo y base de datos

```
===== MENÚ PRINCIPAL =====
1. Registrar factura
2. Ver historial
3. Borrar historial
4. Salir
Seleccione una opción (1-4): 2

--- Historial desde archivo ---
- Alejandra Lopez,P123JKL,Gasolina Premium,7.0,57.61

--- Historial desde base de datos ---
Cliente=Alejandra Lopez, Placa=P123JKL, Combustible=Gasolina Premium, Litros=7.0, Total=Q57.61
```

Figura 11: Tabla 'facturasgasolinera' en base de datos

	cliente	placa	combustible	litros	total
	text	text	text	real	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 9: Tipos de combustible con su respectivo precio y factura generada

```
-----Tipos de combustible-----
1. Gasolina Regular - Q7.84 por litro
2. Gasolina Premium - Q8.23 por litro
3. Diesel - Q7.39 por litro
Seleccione el tipo (1-3): 5
El valor debe ser menor o igual a 3
Seleccione el tipo (1-3): j
Debe ingresar un número entero válido.
Seleccione el tipo (1-3): %
Debe ingresar un número entero válido.
Seleccione el tipo (1-3): -1
El valor debe ser mayor o igual a 1
Seleccione el tipo (1-3): 2
Litros a despachar: 7

--- FACTURA ---
Cliente: Alejandra Lopez
Vehículo: P123JKL
Combustible: Gasolina Premium
Litros: 7.00
Total a pagar: Q57.61
-----
Factura guardada correctamente.
```

Figura 12: Borrar factura especifica

```
===== MENÚ PRINCIPAL =====
1. Registrar factura
2. Ver historial
3. Borrar historial
4. Salir
Seleccione una opción (1-4): 3

--- BORRAR HISTORIAL ---
1. Borrar una factura especifica
2. Borrar todas las facturas
3. Borrar todas las facturas de una placa
4. Cancelar
Seleccione una opción: 1
Ingrese la placa del vehículo de la factura a borrar: p123jkl
Factura borrada del archivo (si existía).
Factura borrada de la base de datos (si existía).
```

Figura 13: Borrar por placa

Data Output

Messages

Notifications

<

Figura 14: Historial borrado

Data Output	Messages	Notifications

Figura 15: Cancelar borrado

```

===== MENÚ PRINCIPAL =====
1. Registrar factura
2. Ver historial
3. Borrar historial
4. Salir
Seleccione una opción (1-4): 3

--- BORRAR HISTORIAL ---
1. Borrar una factura específica (por placa)
2. Borrar todas las facturas
3. Borrar todas las facturas de una placa
4. Cancelar
Seleccione una opción: 4
Cancelando borrado. Regresando al menú principal...

===== MENÚ PRINCIPAL =====
1. Registrar factura
2. Ver historial
3. Borrar historial
4. Salir
Seleccione una opción (1-4):

```

Figura 16: Salir del programa

```

===== MENÚ PRINCIPAL =====
1. Registrar factura
2. Ver historial
3. Borrar historial
4. Salir
Seleccione una opción (1-4): 5
El valor debe ser menor o igual a 4
Seleccione una opción (1-4): 4
Gracias por usar el sistema. Saliendo...

```

F. Octave

Figura 17: Programa en Octave

```

factosParcial 01
Command Window
Factura con placa P123JKL borrada del archivo.
Factura borrada de la base de datos (si existia).

===== MENÚ PRINCIPAL =====
1. Registrar factura
2. Ver historial
3. Borrar historial
4. Salir
Seleccione una opción (1-4): 3

--- BORRAR HISTORIAL ---
1. Borrar una factura específica (por placa)
2. Borrar todas las facturas
3. Borrar todas las facturas de una placa
4. Cancelar
Seleccione una opción: 1
Ingrese placa (P123ABC): p123jkl
Factura con placa P123JKL borrada del archivo.
Factura borrada de la base de datos (si existia).

===== MENÚ PRINCIPAL =====
1. Registrar factura
2. Ver historial
3. Borrar historial
4. Salir
Seleccione una opción (1-4): 3

--- BORRAR HISTORIAL ---
1. Borrar una factura específica (por placa)
2. Borrar todas las facturas
3. Borrar todas las facturas de una placa
4. Cancelar
Seleccione una opción: 3
Ingrese placa (P123ABC): p165lkj
Facturas con placa P165LEJ borradas del archivo (si existian).
Facturas con placa P165LEJ borradas de la base de datos (si existian).

===== MENÚ PRINCIPAL =====
1. Registrar factura
2. Ver historial
3. Borrar historial
4. Salir
Seleccione una opción (1-4):

```

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>