Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería Mecánica Eléctrica
[980] Proyectos de Computación AIE P
1er Examen Parcial
Ing. José Anibal Silva de Los Angeles

Nombre: Alan Andrés Mérida Morales

Carnet: 202100023 Registro académico: 3690273450101

Instrucciones

1. Presentación:
   1. Mostrar su documento de identificación.
   2. Subir un único archivo PDF con el siguiente contenido:
      a. Código del programa.
      b. Enlaces al repositorio externo.
      c. Pantallazos que muestren el funcionamiento del programa.
      d. Reporte en formato IEEE.
      e. Nombre del archivo PDF: Su registro académico.
2. Desarrollo de los programas:
   - Lenguaje: octave y python consola.
   - Almacenamiento de datos: **Archivo de texto "salida.txt" y base de datos PostgreSQL.**
   - El programa debe procurar mitigar los errores de ejecución por parte del usuario.
   - Funcionalidades y menú:
     - Ingreso de nombre usuario.
     - Ingreso ejecución programa
     - Historial de datos.
     - Borrado de datos.
     - Salir
3. Almacenamiento del código:
   - Local: Carpeta personal.
   - Remoto: Repositorio privado de GitHub con el usuario @jasdalinux o jasda@ingenieria.usac.edu.gt.
4. Documentación:
   - Diagrama de flujo del proceso de la solución.
   - Algoritmo del programa.
   - Formato del reporte: IEEE.
5. Aclaraciones:
   - El reporte IEEE debe incluir el código del programa, el diagrama de flujo, el algoritmo y las capturas de pantalla.
   - El archivo PDF debe contener todos los elementos mencionados en la sección "1. Presentación".
   - El repositorio de GitHub debe ser privado y tener como usuario @jasdalinux o jasda@ingenieria.usac.edu.gt.

Programa Cobro y Facturación de gasolina.

Descripción General

Se desea desarrollar un programa en el lenguaje de programación **octave y python consola** que permita a una gasolinera llevar un registro de ventas, realizar el cobro y emitir facturas para sus clientes. El sistema debe ser capaz de registrar información sobre el tipo de combustible vendido, la cantidad de litros despachados, el precio por litro, calcular el monto total a pagar, y emitir un recibo de factura. Toda la información registrada debe almacenarse en un archivo de texto y y base de datos PostgreSQL para mantener un historial de transacciones. Requerimientos Funcionales

Ingreso de Datos del Cliente:

- Solicitar al usuario el nombre del cliente.
- Solicitar la identificación del vehículo (por ejemplo, número de placa).

Selección del Tipo de Combustible:

- Ofrecer al usuario la opción de seleccionar entre diferentes tipos de combustible (por ejemplo, Gasolina Regular, Gasolina Premium, Diesel)

e combustible.

litros de combustible a despachar.
úmero positivo.

do la cantidad de litros por el precio del litro del tipo de combustible

Primer Examen Parcial

Alan Andrés Mérida Morales, 202100023[1,*]

[1] *Facultad de Ingeniería, Escuela de Ingeniería Mecánica Eléctrica, Universidad de San Carlos, Ciudad Universitaria, Zona 12, Guatemala.*

I. OBJETIVOS

- Implementar un sistema que permita a los usuarios ingresar sus datos, seleccionar el tipo de combustible y realizar compras de manera eficiente.
- Desarrollar una funcionalidad que gestione el historial de transacciones almacenando datos en una base de datos y en un archivo de texto.
- Proporcionar opciones para visualizar el historial de datos y borrar información para mantener la integridad y actualización del sistema.

II. INTRODUCCIÓN

Este proyecto se centra en el desarrollo de un sistema de gestión para una gasolinera utilizando Python y una base de datos PostgreSQL. El sistema está diseñado para optimizar el proceso de compra de combustible, desde la captura de datos del cliente hasta la emisión de facturas y el mantenimiento de un historial de transacciones. Los usuarios pueden ingresar sus datos personales, seleccionar el tipo de combustible, y realizar compras, mientras que el sistema gestiona la información almacenada en una base de datos y en un archivo de texto. Además, proporciona opciones para visualizar el historial de transacciones y borrar datos, asegurando una administración eficiente y ordenada de las operaciones diarias en la gasolinera. Este enfoque integral no solo mejora la eficiencia operativa, sino que también facilita el seguimiento y la gestión de las transacciones de manera clara y accesible.

* 3690273450101@ingenieria.usac.edu.gt

III. RESULTADOS

A. Códigos realizados

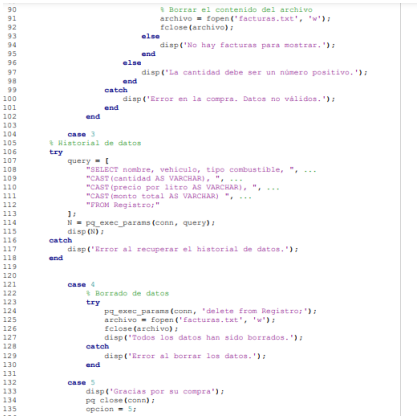
1. *Octave*

```
if (exist('OCTAVE_VERSION', 'builtin') ~= 0)
    % Estamos en Octave
    pkg load database;  % Cargar el paquete para interactuar con bases de datos
end

Cregular = 32.98;
Cpremium = 34.68;
Cdiesel = 30.49;

conn = pq_connect(setdbopts('dbname','usac','host','localhost',
'port','5432','user','postgres','password','A@merida123'));

% Variables para almacenar nombre y placa
nombre = '';
vehiculo = '';

% Menú principal
opcion = 0;
while opcion ~= 5

    disp('1. Ingreso de nombre de usuario');
    disp('2. Compra de combustible');
    disp('3. Registros');
    disp('4. Borrar datos');
    disp('5. Salir');
    opcion = input('Ingrese su elección: ');

    switch opcion
        case 1
            nombre = input('Ingrese su nombre: ', 's');
            vehiculo = input('Ingrese la placa de su vehiculo: ', 's');
            disp('Datos de usuario ingresados correctamente.');

        case 2
            if isempty(nombre) || isempty(vehiculo)
                disp('Debe ingresar primero los datos del usuario en la opción 1.');
            else
                try
                    disp('Tipo de combustible');
                    disp('1. Regular');
                    disp('2. Premium ');
                    disp('3. Diesel ');
                    tipo = input('Indique el combustible que desea: ');

                    switch tipo
                        case 1
                            precio = Cregular;
                            tipoCombustible = 'Regular';
                        case 2
```

Figura 1: Código realizado en octave

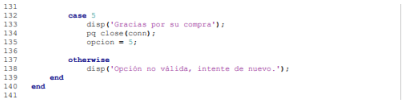
```
                        case 2
                            precio = Cpremium;
                            tipoCombustible = 'Premium';
                        case 3
                            precio = Cdiesel;
                            tipoCombustible = 'Diesel';
                        otherwise
                            disp('Opción no válida, intente de nuevo.');
                            continue;
                    end

                    % Mostrar el precio por litro seleccionado
                    disp(['El precio por litro de ', tipoCombustible, ' es Q', num2str(precio)]);

                    cantidad = input('Ingrese la cantidad deseada (en litros): ');
                    if cantidad > 0
                        MT = cantidad * precio;
                        disp(['El monto total es Q', num2str(MT)]);

                        % Guardar los datos en el archivo de texto
                        archivo = fopen('facturas.txt', 'a');
                        fprintf(archivo, 'Nombre: %s\n', nombre);
                        fprintf(archivo, 'Placa: %s\n', vehiculo);
                        fprintf(archivo, 'Tipo de Combustible: %s\n', tipoCombustible);
                        fprintf(archivo, 'Cantidad (L): %.2f\n', cantidad);
                        fprintf(archivo, 'Precio por Litro: Q%.2f\n', precio);
                        fprintf(archivo, 'Monto Total: Q%.2f\n\n', MT);
                        fclose(archivo);

                        % Guardar los datos en la base de datos
                        query = ['insert into Registro (nombre, vehiculo, tipo_combustible,
                        cantidad, precio_por_litro, monto_total) values (''', nombre, ''', ''',
                        vehiculo, ''', ''', tipoCombustible, ''', ', num2str(cantidad), ', ',
                        num2str(precio), ', ', num2str(MT), ');'];
                        R = pq_exec_params(conn, query);

                        % Imprimir y borrar la factura
                        archivo = fopen('facturas.txt', 'r');
                        if archivo ~= -1
                            disp('----- Factura -----');
                            contenido = fread(archivo, '*char')';
                            fclose(archivo);
                            disp(contenido);

                            % Borrar el contenido del archivo
```

Figura 2: Código realizado en octave



```
                        % Borrar el contenido del archivo
                        archivo = fopen('facturas.txt', 'w');
                        fclose(archivo);
                    else
                        disp('No hay facturas para mostrar.');
                    end
                else
                    disp('La cantidad debe ser un número positivo.');
                end
            catch
                disp('Error en la compra. Datos no válidos.');
            end
        end

    case 3
% Historial de datos
try
    query = [
        "SELECT nombre, vehiculo, tipo combustible, ", ...
        "CAST(cantidad AS VARCHAR), ", ...
        "CAST(precio por litro AS VARCHAR), ", ...
        "CAST(monto total AS VARCHAR) ", ...
        "FROM Registros;"
    ];
    N = pq_exec_params(conn, query);
    disp(N);
catch
    disp('Error al recuperar el historial de datos.');
end


    case 4
        % Borrado de datos
        try
            pq_exec_params(conn, 'delete from Registros;');
            archivo = fopen('facturas.txt', 'w');
            fclose(archivo);
            disp('Todos los datos han sido borrados.');
        catch
            disp('Error al borrar los datos.');
        end

    case 5
        disp('Gracias por su compra');
        pq_close(conn);
        opcion = 5;
```

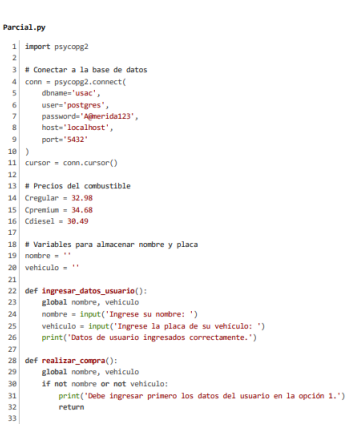
Figura 3: Código realizado en octave



```
    case 5
        disp('Gracias por su compra');
        pq_close(conn);
        opcion = 5;

    otherwise
        disp('Opción no válida, intente de nuevo.');
    end
end
```

Figura 4: Código realizado en octave

2. *Python*



Parcial.py

```
import psycopg2

# Conectar a la base de datos
conn = psycopg2.connect(
    dbname='usac',
    user='postgres',
    password='A@merida123',
    host='localhost',
    port='5432'
)
cursor = conn.cursor()

# Precios del combustible
Cregular = 32.98
Cpremium = 34.68
Cdiesel = 30.49

# Variables para almacenar nombre y placa
nombre = ''
vehiculo = ''

def ingresar_datos_usuario():
    global nombre, vehiculo
    nombre = input('Ingrese su nombre: ')
    vehiculo = input('Ingrese la placa de su vehículo: ')
    print('Datos de usuario ingresados correctamente.')

def realizar_compra():
    global nombre, vehiculo
    if not nombre or not vehiculo:
        print('Debe ingresar primero los datos del usuario en la opción 1.')
        return
```

Figura 5: Código realizado en Python

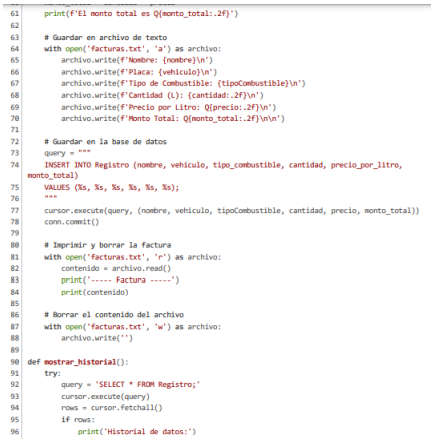
```
    print('Tipo de combustible:')
    print('1. Regular')
    print('2. Premium')
    print('3. Diesel')

    tipo = int(input('Indique el combustible que desea: '))
    if tipo == 1:
        precio = Cregular
        tipoCombustible = 'Regular'
    elif tipo == 2:
        precio = Cpremium
        tipoCombustible = 'Premium'
    elif tipo == 3:
        precio = Cdiesel
        tipoCombustible = 'Diesel'
```

localhost:53038/7cc9dcc9-54e5-4c76-ac78-29f115e685b1/

26/8/24, 11:23 p.m. Parcial.py

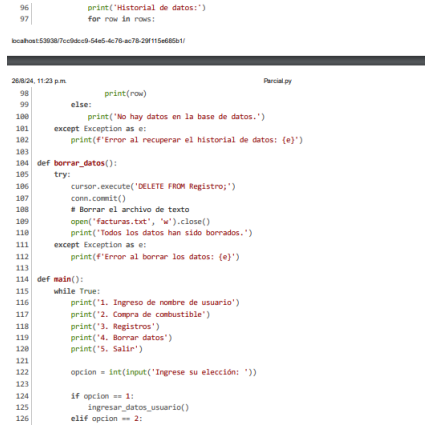
```
    else:
        print('Opción no válida, intente de nuevo.')
        return

    print(f'Precio por litro de {tipoCombustible}: Q{precio:.2f}')
    cantidad = float(input('Ingrese la cantidad deseada (en litros): '))

    if cantidad <= 0:
        print('La cantidad debe ser un número positivo.')
        return

    monto_total = cantidad * precio
    print(f'El monto total es Q{monto_total:.2f}')
```

Figura 6: Código realizado en Python



```
    print(f'El monto total es Q{monto_total:.2f}')

    # Guardar en archivo de texto
    with open('facturas.txt', 'a') as archivo:
        archivo.write(f'Nombre: {nombre}\n')
        archivo.write(f'Placa: {vehiculo}\n')
        archivo.write(f'Tipo de Combustible: {tipoCombustible}\n')
        archivo.write(f'Cantidad (L): {cantidad:.2f}\n')
        archivo.write(f'Precio por Litro: Q{precio:.2f}\n')
        archivo.write(f'Monto Total: Q{monto_total:.2f}\n\n')

    # Guardar en la base de datos
    query = """
    INSERT INTO Registro (nombre, vehiculo, tipo_combustible, cantidad, precio_por_litro,
monto_total)
    VALUES (%s, %s, %s, %s, %s, %s);
    """
    cursor.execute(query, (nombre, vehiculo, tipoCombustible, cantidad, precio, monto_total))
    conn.commit()

    # Imprimir y borrar la factura
    with open('facturas.txt', 'r') as archivo:
        contenido = archivo.read()
        print('----- Factura -----')
        print(contenido)

    # Borrar el contenido del archivo
    with open('facturas.txt', 'w') as archivo:
        archivo.write('')

def mostrar_historial():
    try:
        query = 'SELECT * FROM Registro;'
        cursor.execute(query)
        rows = cursor.fetchall()
        if rows:
            print('Historial de datos:')
```

Figura 7: Código realizado en Python



```
            print('Historial de datos:')
            for row in rows:
```

localhost:53038/7cc9dcc9-54e5-4c76-ac78-29f115e685b1/

26/8/24, 11:23 p.m. Parcial.py

```
                print(row)
        else:
            print('No hay datos en la base de datos.')
    except Exception as e:
        print(f'Error al recuperar el historial de datos: {e}')

def borrar_datos():
    try:
        cursor.execute('DELETE FROM Registro;')
        conn.commit()
        # Borrar el archivo de texto
        open('facturas.txt', 'w').close()
        print('Todos los datos han sido borrados.')
    except Exception as e:
        print(f'Error al borrar los datos: {e}')

def main():
    while True:
        print('1. Ingreso de nombre de usuario')
        print('2. Compra de combustible')
        print('3. Registros')
        print('4. Borrar datos')
        print('5. Salir')

        opcion = int(input('Ingrese su elección: '))

        if opcion == 1:
            ingresar_datos_usuario()
        elif opcion == 2:
```

Figura 8: Código realizado en Python

```python
def main():
    while True:
        print('1. Ingreso de nombre de usuario')
        print('2. Compra de combustible')
        print('3. Registros')
        print('4. Borrar datos')
        print('5. Salir')

        opcion = int(input('Ingrese su elección: '))

        if opcion == 1:
            ingresar_datos_usuario()
        elif opcion == 2:
            realizar_compra()
        elif opcion == 3:
            mostrar_historial()
        elif opcion == 4:
            borrar_datos()
        elif opcion == 5:
            print('Gracias por su compra.')
            cursor.close()
            conn.close()
            break
        else:
            print('Opción no válida, intente de nuevo.')

if __name__ == '__main__':
    main()
```

Figura 9: Código realizado en Python

B. Funcionamiento

1. *Base de datos creada*

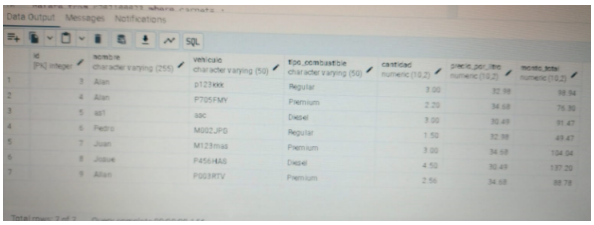


Figura 10: Base de datos creada con PostgreSQL

2. *Octave*

```
1. Ingreso de nombre de usuario
2. Compra de combustible
3. Registros
4. Borrar datos
5. Salir
Ingrese su elección: 1
Ingrese su nombre: Àllan
Ingrese la placa de su vehículo: P003RTV
Datos de usuario ingresados correctamente.
1. Ingreso de nombre de usuario
2. Compra de combustible
3. Registros
4. Borrar datos
5. Salir
Ingrese su elección: 2
Tipo de combustible
1. Regular
2. Premium
3. Diesel
Indique el combustible que desea: 2
El precio por litro de Premium es Q34.68
Ingrese la cantidad deseada (en litros): 2.56
El monto total es Q88.7808
----- Factura -----
Nombre: Àllan
Placa: P003RTV
Tipo de Combustible: Premium
Cantidad (L): 2.56
Precio por Litro: Q34.68
Monto Total: Q88.78
```

Figura 11: Funcionamiento del código en octave

```
1. Ingreso de nombre de usuario
2. Compra de combustible
3. Registros
4. Borrar datos
5. Salir
Ingrese su elección: 3
    data =
    {
      [1,1] = Àlan
      [2,1] = Àlan
      [3,1] = as1
      [4,1] = Pedro
      [5,1] = Juan
      [6,1] = Josue
      [7,1] = Àllan
      [1,2] = p123kkk
      [2,2] = P705FMY
      [3,2] = asc
      [4,2] = M002JPG
      [5,2] = M123mas
      [6,2] = P456HAS
      [7,2] = P003RTV
      [1,3] = Regular
      [2,3] = Premium
      [3,3] = Diesel
      [4,3] = Regular
      [5,3] = Premium
      [6,3] = Diesel
      [7,3] = Premium
      [1,4] = 3.00
      [2,4] = 2.20
      [3,4] = 3.00
      [4,4] = 1.50
      [5,4] = 3.00
```

Figura 12: Funcionamiento del código en octave

3. *Python*

```
PS C:\Users\Usuario\Desktop\usac\semestres\8vo semes
rograms/Python/Python312/python.exe "c:/Users/Usuari
arcial.py"
1. Ingreso de nombre de usuario
2. Compra de combustible
3. Registros
4. Borrar datos
5. Salir
Ingrese su elección: 1
Ingrese su nombre: Andres
Ingrese la placa de su vehículo: P999YTJ
Datos de usuario ingresados correctamente.
1. Ingreso de nombre de usuario
2. Compra de combustible
3. Registros
4. Borrar datos
5. Salir
Ingrese su elección: 2
Tipo de combustible:
1. Regular
2. Premium
3. Diesel
Indique el combustible que desea: 2
Precio por litro de Premium: Q34.68
Ingrese la cantidad deseada (en litros): 2.4
El monto total es Q83.23
----- Factura -----
Nombre: Andres
Placa: P999YTJ
Tipo de Combustible: Premium
Cantidad (L): 2.40
Precio por Litro: Q34.68
Monto Total: Q83.23
```

Figura 13: Funcionamiento del código en python

```
1. Ingreso de nombre de usuario
2. Compra de combustible
3. Registros
4. Borrar datos
5. Salir
Ingrese su elección: 3
Historial de datos:
(3, 'Alan', 'p123kkk', 'Regular', Decimal('3.00'), Decimal('32.98'), Decimal('98.94'))
(4, 'Alan', 'P705FMY', 'Premium', Decimal('2.20'), Decimal('34.68'), Decimal('76.30'))
(5, 'as1', 'asc', 'Diesel', Decimal('3.00'), Decimal('30.49'), Decimal('91.47'))
(6, 'Pedro', 'M002JPG', 'Regular', Decimal('1.50'), Decimal('32.98'), Decimal('49.47'))
(7, 'Juan', 'M123mas', 'Premium', Decimal('3.00'), Decimal('34.68'), Decimal('104.04'))
(8, 'Josue', 'P456HAS', 'Diesel', Decimal('4.50'), Decimal('30.49'), Decimal('137.20'))
(9, 'Allan', 'P003RTV', 'Premium', Decimal('2.56'), Decimal('34.68'), Decimal('88.78'))
(10, 'Andres', 'P999YTJ', 'Premium', Decimal('2.40'), Decimal('34.68'), Decimal('83.23'))
1. Ingreso de nombre de usuario
2. Compra de combustible
3. Registros
4. Borrar datos
5. Salir
Ingrese su elección: 4
Todos los datos han sido borrados.
1. Ingreso de nombre de usuario
2. Compra de combustible
3. Registros
4. Borrar datos
5. Salir
Ingrese su elección:
```

Figura 14: Funcionamiento del código en python

C. Diagrama de flujo del proceso

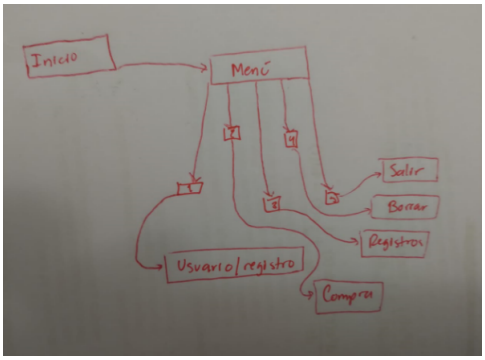


Figura 15: Diagrama de flujo del proceso

D. Algoritmo del Programa

Inicialización

1. Configurar parámetros de conexión a la base de datos.
2. Definir precios del combustible.

Funciones

1. connect_db()
   - Conectar a la base de datos usando psycopg2.
2. ingresar_datos_usuario()
   - Pedir al usuario ingresar su nombre.
   - Pedir al usuario ingresar la placa de su vehículo.
   - Almacenar estos datos en variables globales.
3. realizar_compra(conn)
   - Verificar si el nombre y la placa están ingresados.
     - Si no están ingresados, notificar al usuario.
   - Mostrar opciones de tipo de combustible.
   - Solicitar al usuario seleccionar el tipo de combustible.
     - Establecer el precio según el tipo de combustible seleccionado.
   - Solicitar al usuario ingresar la cantidad de combustible en litros.
     - Si la cantidad es válida (mayor a 0), calcular el monto total.
     - Mostrar el monto total al usuario.
   - Guardar los datos de la compra en un archivo de texto.
   - Insertar los datos de la compra en la base de datos.
   - Imprimir el contenido del archivo de texto.
   - Borrar el contenido del archivo de texto.
4. mostrar_historial(conn)
   - Consultar la base de datos para obtener todos los registros de la tabla Registro.
   - Mostrar los registros al usuario.
5. borrar_datos(conn)
   - Eliminar todos los registros de la tabla Registro en la base de datos.
   - Borrar el contenido del archivo de texto.

Figura 16: Algoritmo del funcionamiento del programa

Menú Principal

1. Iniciar el programa.
2. Mostrar opciones del menú:
   - Opción 1: Ingreso de nombre de usuario
     - Llamar a ingresar_datos_usuario().
   - Opción 2: Compra de combustible
     - Llamar a realizar_compra(conn).
   - Opción 3: Facturación
     - Llamar a mostrar_historial(conn).
   - Opción 4: Borrar datos
     - Llamar a borrar_datos(conn).
   - Opción 5: Salir
     - Cerrar la conexión a la base de datos.
     - Terminar el programa.
3. Repetir el menú hasta que el usuario seleccione la opción 5 para salir.

Figura 17: Algoritmo del funcionamiento del programa

E. Pseudocódigo

```
Iniciar
Configurar conexión a base de datos (DB_PARAMS)
Definir precios del combustible (Cregular, Cpremium, Cdiesel)
Conectar a la base de datos (conn)
nombre = ""
vehiculo = ""
Mientras opción no sea 5
  Mostrar menú principal
  Leer opción del usuario
  Si opción == 1
    Llamar a ingresar_datos_usuario()
  Sino Si opción == 2
    Llamar a realizar_compra(conn)
  Sino Si opción == 3
    Llamar a mostrar_historial(conn)
  Sino Si opción == 4
    Llamar a borrar_datos(conn)
  Sino Si opción == 5
    Mostrar mensaje de salida
    Cerrar conexión a la base de datos (conn)
    Terminar el programa
```

Figura 18: Pseudocódigo del programa

```
  Sino
    Mostrar mensaje de opción no válida
Fin Mientras
Función ingresar_datos_usuario()
  Leer nombre del usuario
  Leer placa del vehículo
  Almacenar nombre y placa en variables globales
```

Figura 19: Pseudocódigo del programa

```
Función realizar_compra(conn)
  Si nombre o placa están vacíos
    Mostrar mensaje para ingresar datos del usuario primero
    Salir de la función

  Mostrar opciones de combustible
  Leer tipo de combustible
  Establecer precio basado en tipo de combustible

  Leer cantidad en litros
  Si cantidad > 0
    Calcular monto total
    Mostrar monto total

    Guardar datos en archivo de texto
    Insertar datos en base de datos

    Leer y mostrar factura
    Borrar contenido del archivo de texto
  Sino
    Mostrar mensaje de cantidad no válida
```

Figura 20: Pseudocódigo del programa

```
Función mostrar_historial(conn)
  Consultar y mostrar todos los registros de la base de datos

Función borrar_datos(conn)
  Eliminar todos los registros de la base de datos
  Borrar contenido del archivo de texto
```

Figura 21: Pseudocódigo del programa

F. Explicación del programa en general

El programa para la gasolinera está diseñado para simplificar y automatizar la gestión de las ventas de combustible. En su funcionamiento general, primero solicita al usuario ingresar su nombre y la placa del vehículo en la opción de Ingreso de nombre de usuario". Esta información se almacena en variables para su uso en transacciones futuras.

En la siguiente fase, el usuario selecciona el tipo de combustible entre las opciones disponibles (Regular, Premium, Diesel) y especifica la cantidad de litros que desea

despachar. El programa muestra el precio por litro correspondiente al tipo de combustible seleccionado y calcula el monto total a pagar. Esta transacción se guarda tanto en una base de datos PostgreSQL como en un archivo de texto, generando así una factura detallada.

El sistema también proporciona la opción de consultar el historial de transacciones almacenadas en la base de datos, permitiendo revisar todos los registros de ventas anteriores. Además, ofrece la posibilidad de borrar todos los datos almacenados, tanto en la base de datos como en el archivo de texto, para mantener la información actualizada y evitar acumulación innecesaria de datos.

El programa está diseñado para ser intuitivo y manejar errores comunes, como la falta de datos del usuario o entradas inválidas, asegurando una experiencia de usuario fluida y eficiente en la gestión de compras de combustible y administración de registros.

G. Repositorio privado

Repositorio creado en Github

IV. CONCLUSIONES

* La implementación del sistema permite un registro automatizado de los datos de clientes y ventas, lo que reduce significativamente la carga administrativa y el riesgo de errores manuales. Al ingresar el nombre del cliente y la placa del vehículo en el sistema, se asegura una recopilación precisa y ordenada de la información necesaria para cada transacción.

* El programa facilita la compra de combustible al mostrar los precios por litro y calcular automáticamente el monto total basado en la cantidad de litros despachados. Además, el sistema genera y almacena facturas detalladas que se pueden imprimir para cada transacción, lo que mejora la transparencia y el control sobre las ventas realizadas.

* El sistema guarda un historial completo de todas las transacciones en una base de datos y un archivo de texto, permitiendo la consulta de registros y la generación de informes. También ofrece funcionalidades para borrar datos tanto en la base de datos como en el archivo de texto, asegurando que la información se mantenga actualizada y relevante para las operaciones de la gasolinera.

Parcial.py

```python
import psycopg2

# Conectar a la base de datos
conn = psycopg2.connect(
    dbname='usac',
    user='postgres',
    password='A@merida123',
    host='localhost',
    port='5432'
)
cursor = conn.cursor()

# Precios del combustible
Cregular = 32.98
Cpremium = 34.68
Cdiesel = 30.49

# Variables para almacenar nombre y placa
nombre = ''
vehiculo = ''

def ingresar_datos_usuario():
    global nombre, vehiculo
    nombre = input('Ingrese su nombre: ')
    vehiculo = input('Ingrese la placa de su vehículo: ')
    print('Datos de usuario ingresados correctamente.')

def realizar_compra():
    global nombre, vehiculo
    if not nombre or not vehiculo:
        print('Debe ingresar primero los datos del usuario en la opción 1.')
        return

    print('Tipo de combustible:')
    print('1. Regular')
    print('2. Premium')
    print('3. Diesel')

    tipo = int(input('Indique el combustible que desea: '))
    if tipo == 1:
        precio = Cregular
        tipoCombustible = 'Regular'
    elif tipo == 2:
        precio = Cpremium
        tipoCombustible = 'Premium'
    elif tipo == 3:
        precio = Cdiesel
        tipoCombustible = 'Diesel'
```

```
    else:
        print('Opción no válida, intente de nuevo.')
        return

    print(f'Precio por litro de {tipoCombustible}: Q{precio:.2f}')
    cantidad = float(input('Ingrese la cantidad deseada (en litros): '))

    if cantidad <= 0:
        print('La cantidad debe ser un número positivo.')
        return

    monto_total = cantidad * precio
    print(f'El monto total es Q{monto_total:.2f}')

    # Guardar en archivo de texto
    with open('facturas.txt', 'a') as archivo:
        archivo.write(f'Nombre: {nombre}\n')
        archivo.write(f'Placa: {vehiculo}\n')
        archivo.write(f'Tipo de Combustible: {tipoCombustible}\n')
        archivo.write(f'Cantidad (L): {cantidad:.2f}\n')
        archivo.write(f'Precio por Litro: Q{precio:.2f}\n')
        archivo.write(f'Monto Total: Q{monto_total:.2f}\n\n')

    # Guardar en la base de datos
    query = """
    INSERT INTO Registro (nombre, vehiculo, tipo_combustible, cantidad, precio_por_litro,
monto_total)
    VALUES (%s, %s, %s, %s, %s, %s);
    """
    cursor.execute(query, (nombre, vehiculo, tipoCombustible, cantidad, precio, monto_total))
    conn.commit()

    # Imprimir y borrar la factura
    with open('facturas.txt', 'r') as archivo:
        contenido = archivo.read()
        print('----- Factura -----')
        print(contenido)

    # Borrar el contenido del archivo
    with open('facturas.txt', 'w') as archivo:
        archivo.write('')

def mostrar_historial():
    try:
        query = 'SELECT * FROM Registro;'
        cursor.execute(query)
        rows = cursor.fetchall()
        if rows:
            print('Historial de datos:')
            for row in rows:
```

```
                print(row)
        else:
            print('No hay datos en la base de datos.')
    except Exception as e:
        print(f'Error al recuperar el historial de datos: {e}')

def borrar_datos():
    try:
        cursor.execute('DELETE FROM Registro;')
        conn.commit()
        # Borrar el archivo de texto
        open('facturas.txt', 'w').close()
        print('Todos los datos han sido borrados.')
    except Exception as e:
        print(f'Error al borrar los datos: {e}')

def main():
    while True:
        print('1. Ingreso de nombre de usuario')
        print('2. Compra de combustible')
        print('3. Registros')
        print('4. Borrar datos')
        print('5. Salir')

        opcion = int(input('Ingrese su elección: '))

        if opcion == 1:
            ingresar_datos_usuario()
        elif opcion == 2:
            realizar_compra()
        elif opcion == 3:
            mostrar_historial()
        elif opcion == 4:
            borrar_datos()
        elif opcion == 5:
            print('Gracias por su compra.')
            cursor.close()
            conn.close()
            break
        else:
            print('Opción no válida, intente de nuevo.')

if __name__ == '__main__':
    main()

```

```
if (exist('OCTAVE_VERSION', 'builtin') ~= 0)
    % Estamos en Octave
    pkg load database;  % Cargar el paquete para interactuar con bases de datos
end

Cregular = 32.98;
Cpremium = 34.68;
Cdiesel = 30.49;

conn = pq_connect(setdbopts('dbname','usac','host','localhost',
'port','5432','user','postgres','password','A@merida123'));

% Variables para almacenar nombre y placa
nombre = '';
vehiculo = '';

% Menú principal
opcion = 0;
while opcion ~= 5

    disp('1. Ingreso de nombre de usuario');
    disp('2. Compra de combustible');
    disp('3. Registros');
    disp('4. Borrar datos');
    disp('5. Salir');
    opcion = input('Ingrese su elección: ');

    switch opcion
        case 1
            nombre = input('Ingrese su nombre: ', 's');
            vehiculo = input('Ingrese la placa de su vehículo: ', 's');
            disp('Datos de usuario ingresados correctamente.');

        case 2
            if isempty(nombre) || isempty(vehiculo)
                disp('Debe ingresar primero los datos del usuario en la opción 1.');
            else
                try
                    disp('Tipo de combustible');
                    disp('1. Regular');
                    disp('2. Premium ');
                    disp('3. Diesel ');
                    tipo = input('Indique el combustible que desea: ');

                    switch tipo
                        case 1
                            precio = Cregular;
                            tipoCombustible = 'Regular';
                        case 2
                            precio = Cpremium;
                            tipoCombustible = 'Premium';
                        case 3
                            precio = Cdiesel;
                            tipoCombustible = 'Diesel';
                        otherwise
                            disp('Opción no válida, intente de nuevo.');
                            continue;
                    end

                    % Mostrar el precio por litro seleccionado
                    disp(['El precio por litro de ', tipoCombustible, ' es Q', num2str(precio)]);

                    cantidad = input('Ingrese la cantidad deseada (en litros): ');
                    if cantidad > 0
                        MT = cantidad * precio;
                        disp(['El monto total es Q', num2str(MT)]);

                        % Guardar los datos en el archivo de texto
                        archivo = fopen('facturas.txt', 'a');
                        fprintf(archivo, 'Nombre: %s\n', nombre);
                        fprintf(archivo, 'Placa: %s\n', vehiculo);
                        fprintf(archivo, 'Tipo de Combustible: %s\n', tipoCombustible);
                        fprintf(archivo, 'Cantidad (L): %.2f\n', cantidad);
                        fprintf(archivo, 'Precio por Litro: Q%.2f\n', precio);
```

```
                        fprintf(archivo, 'Monto Total: Q%.2f\n\n', MT);
                        fclose(archivo);

                        % Guardar los datos en la base de datos
                        query = ['insert into Registro (nombre, vehiculo, tipo_combustible,
                        cantidad, precio_por_litro, monto_total) values (''', nombre, ''', ''',
                        vehiculo, ''', ''', tipoCombustible, ''', ', num2str(cantidad), ', ',
                        num2str(precio), ', ', num2str(MT), ');'];
                        N = pq_exec_params(conn, query);

                        % Imprimir y borrar la factura
                        archivo = fopen('facturas.txt', 'r');
                        if archivo ~= -1
                            disp('----- Factura -----');
                            contenido = fread(archivo, '*char')';
                            fclose(archivo);
                            disp(contenido);

                            % Borrar el contenido del archivo
                            archivo = fopen('facturas.txt', 'w');
                            fclose(archivo);
                        else
                            disp('No hay facturas para mostrar.');
                        end
                    else
                        disp('La cantidad debe ser un número positivo.');
                    end
                catch
                    disp('Error en la compra. Datos no válidos.');
                end
            end

        case 3
    % Historial de datos
    try
        query = [
            "SELECT nombre, vehiculo, tipo_combustible, ", ...
            "CAST(cantidad AS VARCHAR), ", ...
            "CAST(precio_por_litro AS VARCHAR), ", ...
            "CAST(monto_total AS VARCHAR) ", ...
            "FROM Registro;"
        ];
        N = pq_exec_params(conn, query);
        disp(N);
    catch
        disp('Error al recuperar el historial de datos.');
    end


        case 4
            % Borrado de datos
            try
                pq_exec_params(conn, 'delete from Registro;');
                archivo = fopen('facturas.txt', 'w');
                fclose(archivo);
                disp('Todos los datos han sido borrados.');
            catch
                disp('Error al borrar los datos.');
            end

        case 5
            disp('Gracias por su compra');
            pq_close(conn);
            opcion = 5;

        otherwise
            disp('Opción no válida, intente de nuevo.');
    end
end
```