Corto*

Emerson Aldair Pérez Rivera, 201902852[1,**]

[1]*Facultad de Ingeniería, Universidad de San Carlos, Edificio T1, Ciudad Universitaria, Zona 12, Guatemala.*

I. CODIGOS UTILIZADOS

A. Codigo Octave

```
% Establecer conexi n con la base de datos PostgreSQL
db_name = "gasolinera";
db_user = "postgres";
db_password = "kuto";
db_host = "localhost";
db_port = "5432";

% Crear la tabla si no existe
create_table_query = {
    "CREATE TABLE IF NOT EXISTS gasolinera (", ...
    "Usuario VARCHAR(50),", ...
    "placa VARCHAR(50),", ...
    "litros int NOT NULL,", ...
    "tipo_gasolina VARCHAR(50)", ...
    ");"
};

% Une las l neas del array de celdas en una sola cadena con saltos de l nea
create_table_query_str = strjoin(create_table_query, "\n");

% Ejecutar el comando SQL para crear la tabla
system(['psql -U ', db_user, ' -d ', db_name, ' -c "', create_table_query_str, '"']);


function info_usuario(user, placa, litros, tipo)
    printf("Bienvenido a la Gasolinera El COLOCHO\n");
    insert_query = sprintf('INSERT INTO gasolinera (Usuario, placa, litros, tipo_gasolina) VALUES (''%s'', ''%s'', %d, ''%s'');', user, placa, litros, tipo);
    system(['psql -U postgres -d gasolinera -c "', insert_query, '"']);
end

function menu()
    printf("QUE DESEA REALIZAR\n");
    printf("1. Ingresar Usuario y Gasolina\n");
    printf("2. Ejecutar Programa\n");
    printf("3. Borrar Informaci n\n");
    printf("4. Historial de Usuarios\n");
    printf("5. Salir\n");
end

function historial()
    select_query = "SELECT * FROM gasolinera;";
    system(['psql -U postgres -d gasolinera -c "', select_query, '"']);
end

function numero = solicitar_entero_positivo(mensaje)
    while true
        entrada = input(mensaje, 's');
        if all(isstrprop(entrada, 'digit'))
            numero = str2num(entrada);
            if numero > 0
                break;
            else
                printf("Error: El n mero debe ser mayor que cero.\n");
            end
        else
            printf("Error: Solo se permiten n meros enteros positivos.\n");
        end
    end
end

opcion = 0;
regular = 15;
premium = 25;
diesel = 5;
directorio = "C:/Users/perez/OneDrive/Documentos/Tareas Proyectos/Primer_Parcial/facturas/";

while opcion != 5
    menu();
    opcion = solicitar_entero_positivo("Ingrese una opci n: ");

    if opcion == 1
        user = input("Ingrese su nombre: ", 's');
        placa = input("Ingrese su placa: ", 's');

        printf("1. Regular Precio: L. 15.00\n");
        printf("2. Premium Precio: L. 25.00\n");
        printf("3. Diesel Precio: L. 5.00\n");

        litros = solicitar_entero_positivo("Ingrese la cantidad de litros: ");
        tipo = solicitar_entero_positivo("Ingrese el tipo de gasolina (1- Regular, 2- Premium, 3- Diesel): ");

        tipo_gasolina = "";
        precio_litro = 0;

        if tipo == 1
            tipo_gasolina = "Regular";
            precio_litro = regular;
            total = litros * precio_litro;
        elseif tipo == 2
            tipo_gasolina = "Premium";
```

---

* Proyectos

** g-mail: 3741958620101@ingenieria.usac.edu.gt

```
91              precio_litro = premium;
92              total = litros * precio_litro;
93          elseif tipo == 3
94              tipo_gasolina = "Diesel";
95              precio_litro = diesel;
96              total = litros * precio_litro;
97          else
98              printf("Tipo de gasolina no v lido
    .\n");
99              continue;
100         end
101
102         info_usuario(user, placa, litros,
    tipo_gasolina);
103         printf("El total a pagar es: %d\n",
    total);
104
105         filename = strcat(directorio, user, ".
    txt");
106
107         % Escribir la informaci n en el archivo
108         fid = fopen(filename, "w");
109         if fid != -1
110             fprintf(fid, "Usuario: %s\n", user);
111             fprintf(fid, "Placa: %s\n", placa);
112             fprintf(fid, "Litros: %d\n", litros)
    ;
113             fprintf(fid, "Tipo de Gasolina: %s\n
    ", tipo_gasolina);
114             fprintf(fid, "Precio por Litro: L.
    %.2f\n", precio_litro);
115             fprintf(fid, "Total a Pagar: L. %.2f
    \n", total);
116             fclose(fid);
117         else
118             printf("Error al abrir el archivo
    para escribir.\n");
119         end
120
121     elseif opcion == 2
122         readUserFile = input('Ingrese el nombre
    del archivo a leer: ', 's');
123         filename = strcat(directorio,
    readUserFile, '.txt');
124
125         fid = fopen(filename, 'r');
126         if fid != -1
127             while ~feof(fid)
128                 line = fgetl(fid);
129                 disp(line);
130             end
131             fclose(fid);
132         else
133             printf('Error al leer el archivo: El
     archivo no existe.\n');
134         end
135
136     elseif opcion == 3
137         deleteUserFile = input('Ingrese el
    nombre del archivo a borrar: ', 's');
138         filename = strcat(directorio,
    deleteUserFile, '.txt');
139
140         if exist(filename, 'file')
141             delete(filename);
142             delete_query = sprintf("DELETE FROM
    gasolinera WHERE Usuario = '%s';",
    deleteUserFile);
143             system(['psql -U postgres -d
    gasolinera -c "', delete_query, '"']);
144             printf('Informaci n borrada del
    archivo y usuario eliminado de la base de
    datos.\n');
145         else
146             printf('Error: El archivo no existe
    .\n');
147         end
148
149     elseif opcion == 4
150         historial();
151
152     elseif opcion == 5
153         printf("Saliendo del programa...\n");
154
155     else
156         printf("Opci n no v lida , por favor
    intente de nuevo.\n");
157     end
158 end
```

B. Codigo Python

```
1   import psycopg2
2 import os
3
4 conn = psycopg2.connect(
5     dbname="parqueo",
6     user="postgres",
7     password="kuto",
8     host="localhost",
9     port="5432"
10 )
11 cursor = conn.cursor()
12
13 create_table_query = '''
14     CREATE TABLE IF NOT EXISTS parqueo (
15     Usuario VARCHAR(50),
16     nit int NOT NULL,
17     placa VARCHAR(50),
18     hora_entrada int NOT NULL,
19     minutos_entrada int NOT NULL,
20     hora_salida int NOT NULL,
21     minutos_salida int NOT NULL
22 );
23 '''
24
25 cursor.execute(create_table_query)
26 conn.commit()
27
28 def info_usuario(user, nit, placa, hora_in,
       minutos_in, hora_out, minutos_out):
29     print("Bienvenido al Parqueo El COLOCHO")
30     cursor.execute("INSERT INTO parqueo (Usuario
       , nit, placa, hora_entrada, minutos_entrada,
        hora_salida, minutos_salida) VALUES (%s, %s
       , %s, %s,%s, %s,%s);", (user, nit, placa,
       hora_in, minutos_in, hora_out, minutos_out))
31     conn.commit()
32
33 def menu():
34     print("QUE DESEA REALIZAR")
35     print("1. Ingresar Informacion de Usuario")
36     print("2. Ejecutar Programa")
37     print("3. Borrar Informaci n")
38     print("4. Historial de Usuarios")
39     print("5. Salir")
40
```

```
def historial():
    cursor.execute("SELECT * FROM parqueo;")
    rows = cursor.fetchall()
    for row in rows:
        print(f"Usuario: {row[0]}, Placa: {row[2]}, Nit: {row[1]}, Hora de entrada: {row[3]}, Minutos de Entrada: {row[4]}, Hora de salida: {row[5]}, Minutos de salida: {row[6]}")

def menu_borrar():
    print("1. Borrar un archivo")
    print("2. Borrar todos los archivos")
    print("3. Salir")

def ver_usuario():
    cursor.execute("SELECT * FROM parqueo;")
    usuario = cursor.fetchall()
    for user in usuario:
        print(f"Usuario: {user[0]}")

def solicitar_entero_positivo(mensaje):
    while True:
        entrada = input(mensaje)
        if entrada.isdigit():
            numero = int(entrada)
            if numero > 0:
                return numero
            else:
                print("Error: El n mero debe ser mayor que cero.")
        else:
            print("Error: Solo se permiten n meros enteros positivos.")

def solicitar_entero_no_negativo(mensaje):
    while True:
        entrada = input(mensaje)
        if entrada.isdigit():
            numero = int(entrada)
            if numero >= 0 and numero < 60:
                return numero
            else:
                print("Error: El n mero debe ser mayor o igual que cero y menor que 60.")
        else:
            print("Error: Solo se permiten n meros enteros positivos.")

def solicitar_entero_positivo_borrar(mensaje):
    while True:
        entrada = input(mensaje)
        if entrada.isdigit():
            numero = int(entrada)
            if numero > 0 and numero < 4:
                return numero
            else:
                print("Error: El n mero debe ser mayor que 0 y menor que 4")
        else:
            print("Error: Solo se permiten n meros enteros positivos.")

def solicitar_hora(mensaje):
    while True:
        entrada = input(mensaje)
        if entrada.isdigit():
            numero = int(entrada)
            if numero >= 0 and numero < 24:
                return numero
            else:
                print("Error: La hora debe ser mayor o igual que cero y menor que 24.")
        else:
            print("Error: Solo se permiten n meros enteros positivos.")


def solicitar_mensaje_string(mensaje):
    while True:
        entrada = input(mensaje)
        if entrada.isalpha():
            return entrada
        else:
            print("Error: Solo se permiten letras.")

opcion = 0
opcionu = 0
primera_hora = 15
hora_restante = 20

directorio = "E:/perez/Documentos/Cursos 2do. Semestre 2024/PROYECTOS DE COMPUTACION APLICADA A I.E. Secci n P/Tareas Proyectos/Corto/facturas"

while opcion != 5:
    menu()
    opcion = solicitar_entero_positivo("Ingrese una opcion: ")

    if opcion == 1:
        user = solicitar_mensaje_string("Ingrese el nombre del usuario: ")
        nit = solicitar_entero_positivo("Ingrese el nit: ")
        placa = input("Ingrese la placa del vehiculo: ")
        hora_in = solicitar_hora("Ingrese la hora de entrada: ")
        minutos_in = solicitar_entero_no_negativo("Ingrese los minutos de entrada: ")
        hora_out = solicitar_hora("Ingrese la hora de salida: ")
        minutos_out = solicitar_entero_no_negativo("Ingrese los minutos de salida: ")

        total_hora = ((hora_out - hora_in -1)* hora_restante) + primera_hora

        if minutos_out != 0 or minutos_in != 0:
            total_minutos = ((minutos_out - minutos_in)*hora_restante)//60
        if minutos_out == 0:
            total_minutos = ((minutos_in)* hora_restante)//60

        total = total_hora + total_minutos
        info_usuario(user, nit, placa, hora_in, minutos_in, hora_out, minutos_out)

        filename = os.path.join(directorio, f"{user}.txt")
        with open(filename, "w") as file:
            file.write("Bienvenido al Parqueo El COLOCHO\n")
            file.write(f"Usuario: {user.
```

```
    capitalize()}\n")
149             file.write(f"Placa: {placa.upper()}\
    n")
150             file.write(f"Nit: {nit}\n")
151             file.write(f"Hora de entrada: {
    hora_in}:{minutos_in}\n")
152             file.write(f"Hora de salida: {
    hora_out}:{minutos_out}\n")
153             file.write(f"Total a pagar: Q. {
    total}\n")
154     elif opcion == 2:
155         ver_usuario()
156         readUserFile = solicitar_mensaje_string(
    "Ingrese el nombre del usuario: ")
157         filename = os.path.join(directorio, f"{
    readUserFile}.txt")
158         try:
159             with open(filename, 'r') as file:
160                 for line in file:
161                     print(line.strip())
162         except FileNotFoundError:
163             print("El archivo no existe.")
164         except Exception as e:
165             print(f"Error al Leer el archivo: {e
    }")
166
167     elif opcion == 3:
168         menu_borrar()
169
170         opcion =
    solicitar_entero_positivo_borrar("Ingrese
    una opci n: ")
171         if opcion == 1:
172             ver_usuario()
173             deleteUserFile = input('Ingrese el
    nombre del archivo a borrar: ')
174             filename = os.path.join(directorio,
    deleteUserFile + '.txt')
175             try:
176                 if os.path.exists(filename):
177                     os.remove(filename)
178                     cursor.execute("DELETE FROM
    parqueo WHERE usuario = %s;", (
    deleteUserFile,))
179                     conn.commit()
180                     print('Informaci n borrada
    del archivo y usuario eliminado de la base
    de datos.')
181                 else:
182                     print('Error: El archivo no
    existe.')
183             except Exception as e:
184                 print('Error al borrar la
    informaci n del archivo:', e)
185         elif opcion == 2:
186             for deletefile in os.listdir(
    directorio):
187                 dataname = deletefile.replace(".
    txt", "")
188                 cursor.execute("DELETE FROM
    parqueo WHERE usuario = %s;", (dataname,))
189             conn.commit()
190             print('Informaci n de la base de
    datos eliminada')
191
192             for deletefile in os.listdir(
    directorio):
193                 os.remove(os.path.join(
    directorio, deletefile))
194             print('Todos los archivos han sido
    eliminados')
195         elif opcion == 3:
196             print("Saliendo del Menu Borrar..")
197     elif opcion == 4:
198         historial()
199     elif opcion == 5:
200         print("Saliendo del programa...")
201     else:
202         print("Opci n no v lida, por favor
    intente de nuevo.")
203
204 cursor.close()
205 conn.close()
```

II. PRESENTACION

Algoritmo

III. CONEXIÓN A LA BASE DE DATOS

- Conectar a la base de datos PostgreSQL con los parámetros proporcionados.
- Crear un cursor para ejecutar consultas SQL.

IV. CREAR TABLA

- Ejecutar una consulta SQL para crear la tabla `parqueo` si no existe.

V. DEFINIR FUNCIONES

- `info_usuario(user, nit, placa, hora_in, minutos_in, hora_out, minutos_out)`: Inserta la información de un usuario en la tabla `parqueo`.
- `menu()`: Muestra el menú principal de opciones.
- `historial()`: Muestra el historial de usuarios almacenados en la tabla `parqueo`.
- `menu_borrar()`: Muestra el menú para borrar archivos.
- `ver_usuario()`: Muestra todos los usuarios registrados en la tabla `parqueo`.
- `solicitar_entero_positivo(mensaje)`: Solicita un número entero positivo al usuario.
- `solicitar_entero_no_negativo(mensaje)`: Solicita un número entero no negativo al usuario.
- `solicitar_entero_positivo_borrar(mensaje)`: Solicita un número entero positivo para opciones de borrado.
- `solicitar_hora(mensaje)`: Solicita una hora (de 0 a 23).

- `solicitar_mensaje_string(mensaje)`: Solicita una cadena de texto al usuario que contenga solo letras.

VI. EJECUCIÓN DEL PROGRAMA

- Inicializar variables `opcion` y `opcionu`.
- Configurar el directorio para guardar archivos.

VII. MENÚ PRINCIPAL

- Mostrar el menú y solicitar la opción deseada.
- Si la opción es 1 (Ingresar Información de Usuario):
  - Solicitar información del usuario: nombre, nit, placa, hora y minutos de entrada y salida.
  - Calcular el total a pagar.
  - Insertar la información en la base de datos.
  - Crear un archivo de texto con la información del usuario.
- Si la opción es 2 (Ejecutar Programa):
  - Mostrar todos los usuarios.
  - Solicitar el nombre del usuario y leer su archivo de texto.
- Si la opción es 3 (Borrar Información):
  - Mostrar el menú de borrado.
  - Si se selecciona opción 1 (Borrar un archivo):
    - Solicitar el nombre del archivo a borrar.
    - Eliminar el archivo y la entrada correspondiente en la base de datos.
  - Si se selecciona opción 2 (Borrar todos los archivos):
    - Eliminar todas las entradas de la base de datos.
    - Eliminar todos los archivos del directorio.
- Si la opción es 4 (Historial de Usuarios):
  - Mostrar el historial de usuarios almacenados en la base de datos.
- Si la opción es 5 (Salir):
  - Salir del programa.

VIII. CERRAR RECURSOS

- Cerrar el cursor y la conexión a la base de datos.

A. Diagrama de Flujo

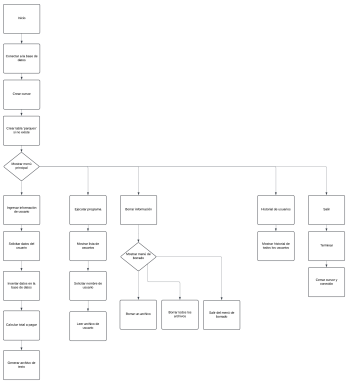

Figura 1
Fuente: Elaboracion Propia, 2024

IX. REPOSITORIO GITHUB