ESCUELA DE INGENIERÍA MECÁNICA ELÉCTRICA
FACULTAD DE INGENIERÍA
UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

Proyectos de computación aplicados a I.E.

Examen Corto #3

David Antonio Rodas Alvarez

202010039

Fecha: Guatemala, 13 de septiembre del 2024

Tercer Examen Corto*

David Antonio, Rodas Alvarez, 202010039[1, **]

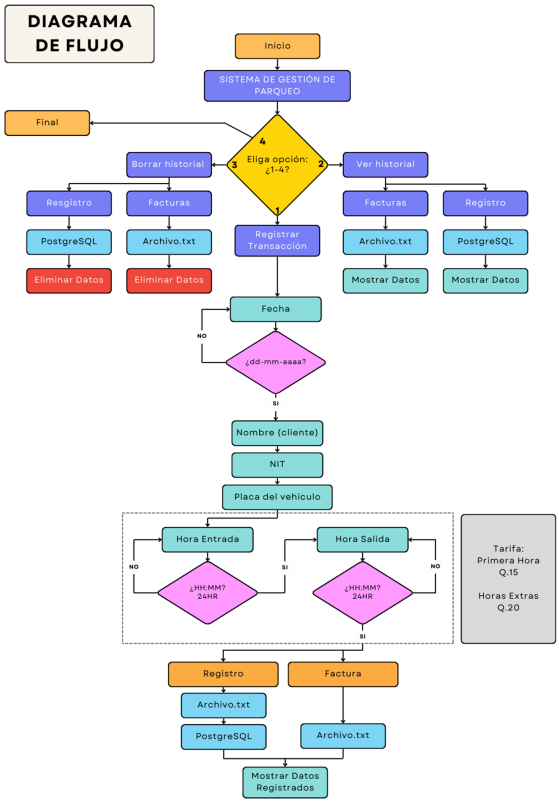
[1] *Escuela de Ingeniería Mecánica Eléctrica , Universidad de San Carlos, Guatemala.*

(Dated: 14 de septiembre de 2024)

I. RESUMEN

La "Academia USAC" es una institución que ofrece una amplia variedad de cursos especializados en el área de ingeniería. Actualmente cuentan con un sistema no muy convencional. Por ello, se requiere de sus servicios como ingeniero con conocimientos en programación para que desarrolle una nueva aplicación en la que se tenga un mejor control de los cursos en los que se inscriben los estudiantes. Se requiere que el sistema pueda ser utilizado por el administradores, profesores y estudiantes, esto implica que la solución deberá tener un diseño agradable e intuitivo de acorde a los requerimientos de la academia.

II. DIAGRAMA DE FLUJO



Fuente: Elaboración Propia, 2024.

III. ALGORITMO

1. **Inicio del Programa**:
   - Se carga el paquete de `database` necesario para conectarse a una base de datos PostgreSQL.
   - Se define la función principal `main()` que controlará el flujo del programa.
2. **Conexión a la Base de Datos**:
   - Se llama a la función `connect_to_db()` para establecer una conexión con la base de datos PostgreSQL para la tabla `salida`.
3. **Bucle Principal**:
   - Se muestra un menú principal con las siguientes opciones:
     a) Registrar transacción
     b) Ver historial
     c) Borrar historial
     d) Salir
4. **Selección de Opción del Menú**:
   - El usuario selecciona una opción del menú.
   - La selección se valida para asegurarse de que es un número válido.
5. **Registrar Transacción (Opción 1)**:
   - Se solicita al usuario que ingrese la **fecha**, **nombre del cliente**, **NIT del cliente**, **la placa del vehículo**, **hora de entrada** y **hora de salida** en el formato HH:MM.
   - Se calcula el **tiempo total en horas** y el **monto total a pagar** basado en las horas de entrada y salida.
   - Se muestra el tiempo total en parqueo y el monto total a pagar.
   - Se guarda la información en la base de datos PostgreSQL en la tabla `salida`.

* Escuela de Ingeniería Mecánica Eléctrica
** e-mail: 3711111370101@ingenieria.usac.edu.gt

- Se guarda la información en un archivo de texto `salida.txt`.
- Se genera y guarda una factura en el archivo de texto `facturas.txt` con los detalles de la transacción.
- Se muestra un mensaje de confirmación al usuario.

6. **Ver Historial (Opción 2)**:
   - Se presenta un submenú que permite elegir entre:
     a) Ver historial de registros (desde la base de datos).
     b) Ver historial de facturas (desde el archivo de texto).
   - El usuario selecciona la opción deseada, y se muestran los registros o las facturas almacenadas.
   - Si no hay registros o facturas, se muestra un mensaje correspondiente.
7. **Borrar Historial (Opción 3)**:
   - Se presenta un submenú que permite elegir entre:
     a) Borrar historial de registros (desde la base de datos).
     b) Borrar historial de facturas (desde el archivo de texto).
   - Se solicita confirmación al usuario antes de proceder.
   - Si se confirma, se eliminan los registros o facturas según la selección del usuario.
   - Se muestra un mensaje de confirmación o cancelación.
8. **Salir del Programa (Opción 4)**:
   - Se muestra un mensaje de despedida.
   - Se cierra la conexión a la base de datos.
   - El programa finaliza.
9. **Repetición del Menú**:
   - Después de cada acción (excepto al salir), se limpia la pantalla y se vuelve a mostrar el menú principal hasta que el usuario seleccione la opción de salir.
10. **Fin del Programa**.

IV. CÓDIGO

A. Octave

```
pkg load database;

% Funcin para conectarse a la base de datos PostgreSQL
function conn = connect_to_db()
  conn = pq_connect(setdbopts("dbname", "postgres", "host",
        "localhost", "port", "5432", "user", "postgres",
        "password", "202010039"));
end

% Funcin para solicitar la fecha
function fecha = solicitar_fecha()
  while true
    fecha = input('Ingrese la fecha (aaaa-mm-dd): ', 's');
    if isempty(regexp(fecha, '^\d{4}-\d{2}-\d{2}$', 'once'))
      printf('Formato de fecha invlido. Debe ser aaaa-mm-dd.\n');
    else
      break;
    end
  end
end

% Funcin para ingresar la hora con validacin de formato
function hora = ingresar_hora(mensaje)
  while true
    hora = input(mensaje, 's');
    if isempty(regexp(hora, '^\d{2}:\d{2}$', 'once'))
      printf('Formato de hora invlido. Debe ser HH:MM.\n');
    else
      break;
    end
  end
end

% Funcin para calcular el monto a pagar
function [tiempo_total, monto_total] =
      calcular_monto(hora_entrada, hora_salida)
    % Convertir horas y minutos
    horas_entrada = str2double(hora_entrada(1:2));
    minutos_entrada = str2double(hora_entrada(4:5));
    horas_salida = str2double(hora_salida(1:2));
    minutos_salida = str2double(hora_salida(4:5));

    % Calcular tiempo total en minutos
    tiempo_total_min = (horas_salida * 60 + minutos_salida) -
          (horas_entrada * 60 + minutos_entrada);
    if tiempo_total_min <= 0
        tiempo_total_min = tiempo_total_min + 24 * 60; % manejar
             si el tiempo cruza la medianoche
    end

    % Convertir tiempo total a horas, redondeando hacia arriba
    tiempo_total = ceil(tiempo_total_min / 60);

    % Calcular monto total
    if tiempo_total > 1
        monto_total = 15 + (tiempo_total - 1) * 20;
    else
        monto_total = 15;
    end
end

% Funcin para guardar la informacin en un archivo de texto
      (salida.txt)
function guardar_en_salida_txt(fecha, nombre_cliente,
      id_vehiculo, hora_entrada, hora_salida)
  try
    ruta_archivo = "salida.txt";
    fid = fopen(ruta_archivo, "a");
    if fid == -1
      error("No se pudo abrir el archivo.");
    end

    fprintf(fid, "Fecha: %s\n", fecha);
    fprintf(fid, "Nombre del cliente: %s\n", nombre_cliente);
    fprintf(fid, "Vehculo: %s\n", id_vehiculo);
    fprintf(fid, "Hora Entrada: %s\n", hora_entrada);
```

```
    fprintf(fid, "Hora Salida: %s\n", hora_salida);
    fprintf(fid, "----------------------------------------\n");

    fclose(fid);
    printf(" Informacin  guardada en 'salida.txt' con xito!\n");
  catch
    printf("Error al escribir en el archivo 'salida.txt'.\n");
  end
end

% Funcin para guardar la factura en el archivo de texto
     (facturas.txt)
function generar_factura_txt(fecha, nombre_cliente, nit_cliente,
     tiempo_total, monto_total)
  try
    ruta_archivo = "facturas.txt";
    fid = fopen(ruta_archivo, "a");
    if fid == -1
      error("No se pudo abrir el archivo.");
    end

    fprintf(fid, "Fecha: %s\n", fecha);
    fprintf(fid, "Nombre del cliente: %s\n", nombre_cliente);
    fprintf(fid, "NIT: %s\n", nit_cliente);
    fprintf(fid, "Tiempo en el parqueo: %d horas\n",
         tiempo_total);
    fprintf(fid, "Monto total a pagar: Q%.2f\n", monto_total);
    fprintf(fid, "----------------------------------------\n");

    fclose(fid);
    printf("Factura guardada en 'facturas.txt' con xito!\n");
  catch
    printf("Error al escribir en el archivo 'facturas.txt'.\n");
  end
end

% Funcin para guardar la informacin en la base de datos
     PostgreSQL
function guardar_info_db(conn, fecha, nombre_cliente,
     id_vehiculo, hora_entrada, hora_salida)
  query = "INSERT INTO salida (fecha, nombre_cliente,
       id_vehiculo, hora_entrada, hora_salida) VALUES ($1, $2,
       $3, $4, $5);";
  valores = {fecha, nombre_cliente, id_vehiculo, hora_entrada,
       hora_salida};

  try
    pq_exec_params(conn, query, valores); % Ejecutar la consulta
         con parmetros
    printf(" Informacin  registrada en la base de datos!\n");
  catch err
    error("Error ejecutando la consulta SQL: %s", err.message);
  end
end

% Funcin para validar que la entrada sea numrica
function valor = validar_entrada_numerica(mensaje)
  while true
    try
      valor = input(mensaje);
      if isnumeric(valor) && isscalar(valor) && valor > 0
        break;
      else
        printf("Entrada no vlida. Debes ingresar un nmero
             positivo.\n");
      end
    catch
      printf("Entrada no vlida. Debes ingresar un nmero.\n");
    end
  end
end

% Funcin para mostrar el historial desde la base de datos,
     ordenado por fecha
function mostrar_historial_postgres(conn)
  query = "SELECT * FROM salida ORDER BY fecha ASC;";
  try
    result = pq_exec_params(conn, query, {}); % Ejecutar la
         consulta con parmetros
    if isempty(result.data)
      printf("No hay registros en la base de datos an.\n");
    else
      printf("\nHistorial de Parqueos (Base de Datos):\n");
      for i = 1:rows(result.data)
        printf("Registro %d:\n", i);
        printf("Fecha: %s\n", result.data{i, 1});
        printf("Nombre del cliente: %s\n", result.data{i, 2});
        printf("Vehculo: %s\n", result.data{i, 3});
        printf("Hora Entrada: %s\n", result.data{i, 4});
        printf("Hora Salida: %s\n\n", result.data{i, 5});
      end
    end
  catch err
    error("Error ejecutando la consulta SQL: %s", err.message);
  end
end

% Funcin para mostrar el historial desde el archivo de texto
     (facturas.txt)
function mostrar_historial_facturas()
  try
    ruta_archivo = "facturas.txt";
    fid = fopen(ruta_archivo, "r");
    if fid == -1
      error("No se pudo abrir el archivo.");
    end

    printf("\nHistorial de Facturas (Archivo de Texto):\n");
    while ~feof(fid)
      linea = fgetl(fid);
      if ischar(linea)
        printf("%s\n", linea);
      end
    end

    fclose(fid);
  catch
    printf("Error al leer el archivo 'facturas.txt'.\n");
  end
end

% Funcin para borrar el historial en la base de datos
function borrar_historial_postgres(conn)
  confirmacion = input("Ests seguro de que deseas borrar todos
       los registros? (s/n): ", "s");
  if lower(confirmacion) == 's'
    query = "DELETE FROM salida;";
    try
      pq_exec_params(conn, query, {}); % Ejecutar la consulta sin
           parmetros
      printf("Historial borrado con xito.\n");
    catch err
      error("Error ejecutando la consulta SQL: %s", err.message);
    end
  else
    printf("Borrado cancelado.\n");
  end
end

% Funcin para borrar el historial de facturas del archivo de
     texto
function borrar_historial_facturas()
  confirmacion = input("Ests seguro de que deseas borrar todas
       las facturas? (s/n): ", "s");
  if lower(confirmacion) == 's'
    ruta_archivo = "facturas.txt";
    fid = fopen(ruta_archivo, "w");
    if fid == -1
      printf("No se pudo abrir el archivo.\n");
    else
      fclose(fid);
      printf("Historial de facturas borrado con xito.\n");
    end
  else
    printf("Borrado cancelado.\n");
  end
end

% Funcin principal con men
function main()
  conn = connect_to_db(); % Conectar a la base de datos
       PostgreSQL
  while true
    clc;  % Limpiar la pantalla
    printf("\nSistema de Gestin de Parqueo\n");
    printf("1. Registrar transaccin\n");
```

```
printf("2. Ver historial\n");
printf("3. Borrar historial\n");
printf("4. Salir\n");

opcion = validar_entrada_numerica("Selecciona una opcin: ");

switch opcion
  case 1
    % Registrar transaccin
    fecha = solicitar_fecha();
    nombre_cliente = input("Por favor, ingresa el nombre del
        cliente: ", "s");
    nit_cliente = input("Ingresa el NIT del cliente: ", "s");
    id_vehiculo = input("Ingresa la identificacin del vehculo
        (nmero de placa): ", "s");
    hora_entrada = ingresar_hora("Ingresa la hora de entrada
        (HH:MM): ");
    hora_salida = ingresar_hora("Ingresa la hora de salida
        (HH:MM): ");

    % Calcular tiempo total y monto a pagar
    [tiempo_total, monto_total] =
        calcular_monto(hora_entrada, hora_salida);

    % Mostrar el monto total en la terminal
    printf("Tiempo total en parqueo: %d horas\n",
        tiempo_total);
    printf("Monto total a pagar: Q%.2f\n", monto_total);

    % Guardar la informacin en la base de datos
    guardar_info_db(conn, fecha, nombre_cliente, id_vehiculo,
        hora_entrada, hora_salida);

    % Guardar la informacin en el archivo salida.txt
    guardar_en_salida_txt(fecha, nombre_cliente, id_vehiculo,
        hora_entrada, hora_salida);

    % Generar factura en facturas.txt
    generar_factura_txt(fecha, nombre_cliente, nit_cliente,
        tiempo_total, monto_total);

    printf("Transaccin registrada con xito.\n");
    pause;

  case 2
    % Ver historial
    while true
      clc;
      printf("\nVer Historial\n");
      printf("1. Ver historial de registros\n");
      printf("2. Ver historial de facturas\n");
      printf("3. Regresar al men principal\n");

      sub_opcion = validar_entrada_numerica("Selecciona una
          opcin: ");

      switch sub_opcion
        case 1
          mostrar_historial_postgres(conn);
          pause;
        case 2
          mostrar_historial_facturas();
          pause;
        case 3
          break;
        otherwise
          printf("Opcin no vlida. Por favor, selecciona una
              opcin correcta.\n");
          pause(2);
      end

      if sub_opcion == 3
        break;
      end
    end

  case 3
    % Borrar historial
    while true
      clc;
      printf("\nBorrar Historial\n");
      printf("1. Borrar historial de registros\n");
      printf("2. Borrar historial de facturas\n");
      printf("3. Regresar al men principal\n");

      sub_opcion = validar_entrada_numerica("Selecciona una
          opcin: ");

      switch sub_opcion
        case 1
          borrar_historial_postgres(conn);
          pause;
        case 2
          borrar_historial_facturas();
          pause;
        case 3
          break;
        otherwise
          printf("Opcin no vlida. Por favor, selecciona una
              opcin correcta.\n");
          pause(2);
      end

      if sub_opcion == 3
        break;
      end
    end

  case 4
    % Salir del programa
    printf("Saliendo del programa. Hasta pronto!\n");
    pq_close(conn); % Cerrar la conexin a la base de datos
    break;

  otherwise
    % Manejar opcin no vlida
    printf("Opcin no vlida. Por favor, selecciona una opcin
        correcta.\n");
    pause(2);
  end

  if opcion == 4
    break; % Salir del bucle si la opcin es 4
  end
 end
end

% Ejecutar la funcin principal
main();
```

Figura 1: Creación de tabla en PostgreSQL

```
postgres=# SELECT * FROM salida;
 nombre_cliente | nit_cliente | id_vehiculo |   fecha    | hora_entrada | hora_salida
----------------+-------------+-------------+------------+--------------+-------------
 David Rodas    | 108510344   | P452WKL     | 16-06-2024 | 2200         | 2300
 David Rodas    | 108510344   | P845QWK     | 16-06-2024 | 1400         | 1600
 David Rodas    | 108510344   | P452QWL     | 16-07-2024 | 2200         | 2300
 D              | D           | P420QWL     | 16-08-2024 | 2200         | 2300
```

Fuente: Elaboración Propia, 2024.

Figura 2: Menu Principal

```
Ver Historial
1. Ver historial de registros
2. Ver historial de facturas
3. Regresar al menú principal
Selecciona una opción:
```

Fuente: Elaboración Propia, 2024.

Figura 3: Registro de facutra Octave

```
Registro 9:
Nombre del cliente: Rodas Alvarez
NIT: 4512151
Vehículo: P234KDL
Fecha: 23-06-2000
Hora Entrada: 2200
Hora Salida: 2300
```

Fuente: Elaboración Propia, 2024.

Figura 4: Historial de entradas Octave)

```
----------------------------------------
Nombre del cliente: Rodas Alvarez
Identificación del vehículo: P234KDL
Tiempo en el parqueo: 1 horas
Monto total a pagar: Q15.00
----------------------------------------
```

Fuente: Elaboración Propia, 2024.

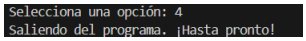
B. Python

```python
import psycopg2
import re

# Funcin para conectarse a la base de datos PostgreSQL
def connect_to_db():
    try:
        conn = psycopg2.connect(
            dbname="postgres",
            host="localhost",
            port="5432",
            user="postgres",
            password="202010039"
        )
        return conn
    except psycopg2.Error as e:
        print(f"Error al conectarse a la base de datos: {e}")
        return None

# Funcin para solicitar la fecha
def solicitar_fecha():
    while True:
        fecha = input('Ingrese la fecha (aaaa-mm-dd): ')
        if not re.match(r'^\d{4}-\d{2}-\d{2}$', fecha):
            print('Formato de fecha invlido. Debe ser aaaa-mm-dd.')
        else:
            return fecha

# Funcin para ingresar la hora con validacin de formato
def ingresar_hora(mensaje):
    while True:
        hora = input(mensaje)
        if not re.match(r'^\d{2}:\d{2}$', hora):
            print('Formato de hora invlido. Debe ser HH:MM.')
        else:
            return hora

# Funcin para calcular el monto a pagar
def calcular_monto(hora_entrada, hora_salida):
    horas_entrada = int(hora_entrada[:2])
    minutos_entrada = int(hora_entrada[3:])
    horas_salida = int(hora_salida[:2])
    minutos_salida = int(hora_salida[3:])

    tiempo_total_min = (horas_salida * 60 + minutos_salida) -
        (horas_entrada * 60 + minutos_entrada)
    if tiempo_total_min <= 0:
        tiempo_total_min += 24 * 60 # manejar si el tiempo cruza
            la medianoche

    tiempo_total = (tiempo_total_min + 59) // 60 # Redondear
        hacia arriba

    if tiempo_total > 1:
        monto_total = 15 + (tiempo_total - 1) * 20
    else:
        monto_total = 15

    return tiempo_total, monto_total

# Funcin para guardar la informacin en un archivo de texto
    (salida.txt)
def guardar_en_salida_txt(fecha, nombre_cliente, id_vehiculo,
    hora_entrada, hora_salida):
    try:
        ruta_archivo = r"D:\Desktop\Proyectos\T6\salida.txt" #
            Actualiza la ruta
        with open(ruta_archivo, "a") as file:
            file.write(f"Fecha: {fecha}\n")
            file.write(f"Nombre del cliente: {nombre_cliente}\n")
            file.write(f"Vehculo: {id_vehiculo}\n")
            file.write(f"Hora Entrada: {hora_entrada}\n")
            file.write(f"Hora Salida: {hora_salida}\n")
            file.write("----------------------------------------\n")
        print(" Informacin  guardada en 'salida.txt' con xito!")
    except IOError:
        print("Error al escribir en el archivo 'salida.txt'.")

# Funcin para guardar la factura en el archivo de texto
    (facturas.txt)
def generar_factura_txt(fecha, nombre_cliente, nit_cliente,
    tiempo_total, monto_total):
    try:
        ruta_archivo = r"D:\Desktop\Proyectos\T6\facturas.txt" #
            Actualiza la ruta
        with open(ruta_archivo, "a") as file:
            file.write(f"Fecha: {fecha}\n")
            file.write(f"Nombre del cliente: {nombre_cliente}\n")
            file.write(f"NIT: {nit_cliente}\n")
            file.write(f"Tiempo en el parqueo: {tiempo_total}
                horas\n")
            file.write(f"Monto total a pagar:
                Q{monto_total:.2f}\n")
            file.write("----------------------------------------\n")
        print("Factura guardada en 'facturas.txt' con xito!")
    except IOError:
        print("Error al escribir en el archivo 'facturas.txt'.")

# Funcin para guardar la informacin en la base de datos
    PostgreSQL
def guardar_info_db(conn, fecha, nombre_cliente, id_vehiculo,
    hora_entrada, hora_salida):
    query = """
    INSERT INTO salida (fecha, nombre_cliente, id_vehiculo,
        hora_entrada, hora_salida)
    VALUES (%s, %s, %s, %s, %s);
    """
    try:
        with conn.cursor() as cur:
            cur.execute(query, (fecha, nombre_cliente,
                id_vehiculo, hora_entrada, hora_salida))
            conn.commit()
        print(" Informacin  registrada en la base de datos!")
    except psycopg2.Error as e:
```

```
        print(f"Error ejecutando la consulta SQL: {e}")

# Funcin para validar que la entrada sea numrica
def validar_entrada_numerica(mensaje):
    while True:
        try:
            valor = int(input(mensaje))
            if valor > 0:
                return valor
            else:
                print("Entrada no vlida. Debes ingresar un nmero
                    positivo.")
        except ValueError:
            print("Entrada no vlida. Debes ingresar un nmero.")

# Funcin para mostrar el historial desde la base de datos,
    ordenado por fecha
def mostrar_historial_postgres(conn):
    query = "SELECT * FROM salida ORDER BY fecha ASC;"
    try:
        with conn.cursor() as cur:
            cur.execute(query)
            records = cur.fetchall()
            if not records:
                print("No hay registros en la base de datos an.")
            else:
                print("\nHistorial de Parqueos (Base de Datos):")
                for i, record in enumerate(records, 1):
                    print(f"Registro {i}:")
                    print(f"Fecha: {record[0]}")
                    print(f"Nombre del cliente: {record[1]}")
                    print(f"Vehculo: {record[2]}")
                    print(f"Hora Entrada: {record[3]}")
                    print(f"Hora Salida: {record[4]}\n")
    except psycopg2.Error as e:
        print(f"Error ejecutando la consulta SQL: {e}")

# Funcin para mostrar el historial desde el archivo de texto
    (facturas.txt)
def mostrar_historial_facturas():
    try:
        with open("facturas.txt", "r") as file:
            print("\nHistorial de Facturas (Archivo de Texto):")
            for linea in file:
                print(linea.strip())
    except IOError:
        print("Error al leer el archivo 'facturas.txt'.")

# Funcin para borrar el historial en la base de datos
def borrar_historial_postgres(conn):
    confirmacion = input("Ests seguro de que deseas borrar todos
        los registros? (s/n): ").lower()
    if confirmacion == 's':
        query = "DELETE FROM salida;"
        try:
            with conn.cursor() as cur:
                cur.execute(query)
                conn.commit()
            print("Historial borrado con xito.")
        except psycopg2.Error as e:
            print(f"Error ejecutando la consulta SQL: {e}")
    else:
        print("Borrado cancelado.")

# Funcin para borrar el historial de facturas del archivo de
    texto
def borrar_historial_facturas():
    confirmacion = input("Ests seguro de que deseas borrar todas
        las facturas? (s/n): ").lower()
    if confirmacion == 's':
        try:
            open("facturas.txt", "w").close()
            print("Historial de facturas borrado con xito.")
        except IOError:
            print("No se pudo abrir el archivo.")
    else:
        print("Borrado cancelado.")

# Funcin principal con men
def main():
    conn = connect_to_db() # Conectar a la base de datos
        PostgreSQL
    if conn is None:
        return

while True:
    print("\nSistema de Gestin de Parqueo")
    print("1. Registrar transaccin")
    print("2. Ver historial")
    print("3. Borrar historial")
    print("4. Salir")

    opcion = validar_entrada_numerica("Selecciona una opcin:
        ")

    if opcion == 1:
        # Registrar transaccin
        fecha = solicitar_fecha()
        nombre_cliente = input("Por favor, ingresa el nombre
            del cliente: ")
        nit_cliente = input("Ingresa el NIT del cliente: ")
        id_vehiculo = input("Ingresa la identificacin del
            vehculo (nmero de placa): ")
        hora_entrada = ingresar_hora("Ingresa la hora de
            entrada (HH:MM): ")
        hora_salida = ingresar_hora("Ingresa la hora de salida
            (HH:MM): ")

        # Calcular tiempo total y monto a pagar
        tiempo_total, monto_total =
            calcular_monto(hora_entrada, hora_salida)

        # Mostrar el tiempo total y el monto total en la
            terminal
        print(f"Tiempo total en parqueo: {tiempo_total} horas")
        print(f"Monto total a pagar: Q{monto_total:.2f}")

        # Guardar la informacin en la base de datos
        guardar_info_db(conn, fecha, nombre_cliente,
            id_vehiculo, hora_entrada, hora_salida)

        # Guardar la informacin en el archivo salida.txt
        guardar_en_salida_txt(fecha, nombre_cliente,
            id_vehiculo, hora_entrada, hora_salida)

        # Generar factura en facturas.txt
        generar_factura_txt(fecha, nombre_cliente,
            nit_cliente, tiempo_total, monto_total)

        print("Transaccin registrada con xito.")

    elif opcion == 2:
        # Ver historial
        while True:
            print("\nVer Historial")
            print("1. Ver historial de registros")
            print("2. Ver historial de facturas")
            print("3. Regresar al men principal")

            sub_opcion = validar_entrada_numerica("Selecciona
                una opcin: ")

            if sub_opcion == 1:
                mostrar_historial_postgres(conn)
            elif sub_opcion == 2:
                mostrar_historial_facturas()
            elif sub_opcion == 3:
                break
            else:
                print("Opcin no vlida. Por favor, selecciona
                    una opcin correcta.")

    elif opcion == 3:
        # Borrar historial
        while True:
            print("\nBorrar Historial")
            print("1. Borrar historial de registros")
            print("2. Borrar historial de facturas")
            print("3. Regresar al men principal")

            sub_opcion = validar_entrada_numerica("Selecciona
                una opcin: ")

            if sub_opcion == 1:
                borrar_historial_postgres(conn)
            elif sub_opcion == 2:
```

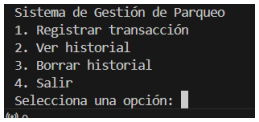
```
                borrar_historial_facturas()
            elif sub_opcion == 3:
                break
            else:
                print("Opcin no vlida. Por favor, selecciona
                    una opcin correcta.")

        elif opcion == 4:
            # Salir del programa
            print("Saliendo del programa. Hasta pronto!")
            conn.close() # Cerrar la conexin a la base de datos
            break

        else:
            print("Opcin no vlida. Por favor, selecciona una
                opcin correcta.")

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

Figura 5: Cierre de Programa



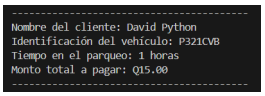
Fuente: Elaboración Propia, 2024.

Figura 6: Menu Principal



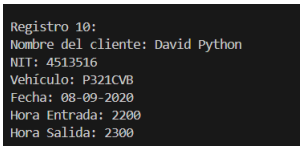
Fuente: Elaboración Propia, 2024.

Figura 7: Registro de facutra Python



Fuente: Elaboración Propia, 2024.

Figura 8: Historial de entradas Python)



Fuente: Elaboración Propia, 2024.

C. Link de GitHub

https://github.com/dvd-r16/proyectos

V. CONCLUSIÓN

1. Durante el horario de clases, se logró desarrollar exitosamente el programa en Octave y Python para calcular el monto a pagar por el servicio de estacionamiento. El programa cumple con todas las especificaciones solicitadas, ofreciendo una tarifa de Q15.00 para la primera hora y Q20.00 para cada hora adicional, contabilizando cada fracción como una hora completa. Además, el programa incluye funcionalidades adicionales, como guardar los resultados en un archivo de texto, leer la información almacenada previamente y borrar el archivo cuando sea necesario. La implementación garantiza una interacción sencilla para el usuario, asegurando que el flujo de ejecución sea claro y eficiente, proporcionando una herramienta eficaz para la gestión automatizada de los datos de estacionamiento.

[1] Ing. Jose Anibal Silva de los Angeles. Proyectos Aplicados para I.E. *Programa del curso*. Ciudad de Guatemala: Universidad de San Carlos, Facultad de Ingeniería, Escuela de Ingeniería Mecánica Eléctrica.