Primera Retra 0980[*]

Emerson Aldair Pérez Rivera, 201902852[1, **]
[1] *Facultad de Ingeniería, Universidad de San Carlos, Edificio T1, Ciudad Universitaria, Zona 12, Guatemala.*

I. DIAGRAMA DE FLUJO

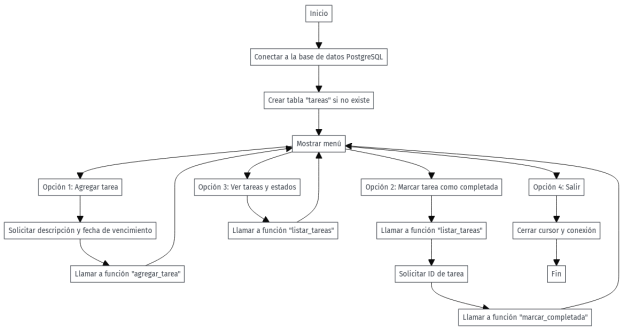

Figura 1
Fuente: Elaboracion Propia, 2024

II. ALGORITMO

III. DESCRIPCIÓN GENERAL

El código Python proporcionado interactúa con una base de datos PostgreSQL para gestionar una lista de tareas. Las principales funcionalidades son:

- **Conexión a la Base de Datos:** Establece una conexión con una base de datos PostgreSQL específica.
- **Creación de Tabla:** Crea una tabla llamada "tareas"si no existe, con campos para el ID, descripción, fecha de vencimiento y estado de completación.
- **Operaciones CRUD:** Implementa las operaciones básicas de Crear, Leer, Actualizar y Eliminar (CRUD) sobre la tabla "tareas".
- **Interfaz de Usuario:** Presenta un menú al usuario para que interactúe con las tareas, permitiendo agregar nuevas tareas, marcarlas como completadas y listar todas las tareas.

IV. ALGORITMO DETALLADO

A. Conexión a la Base de Datos

Se establece una conexión a la base de datos PostgreSQL utilizando la librería `psycopg2`. Se crea un cursor para ejecutar las consultas SQL.

B. Creación de la Tabla

Se ejecuta una consulta SQL para crear la tabla "tareas"si aún no existe. La tabla tiene los siguientes campos:

- `id`: Un número serial que actúa como clave primaria.
- `descripcion`: Un texto que describe la tarea.
- `fecha_vencimiento`: Una fecha que indica la fecha límite de la tarea.
- `completada`: Un valor booleano que indica si la tarea está completada o no.

C. Bucle Principal

Se inicia un bucle `while` que se ejecuta hasta que el usuario selecciona la opción de salir. En cada iteración del bucle:

- Se muestra un menú con las opciones disponibles (agregar tarea, marcar como completada, ver tareas, salir).
- El usuario selecciona una opción.
- Según la opción seleccionada, se ejecuta la función correspondiente:
  - **Agregar Tarea:** Se pide al usuario que ingrese la descripción y la fecha de vencimiento. Se inserta un nuevo registro en la tabla "tareasçon los datos proporcionados.
  - **Marcar como Completada:** Se muestran todas las tareas y se pide al usuario que ingrese el ID de la tarea a marcar como completada. Se actualiza el registro correspondiente en la tabla "tareas"para establecer el campo çompletada.[en] `TRUE`.
  - **Ver Tareas:** Se seleccionan todos los registros de la tabla "tareasz se muestran en pantalla.

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

D. Cierre de la Conexión

Al finalizar el bucle, se cierra el cursor y la conexión a la base de datos.

V. SEUDOCODIGO

```
Inicio

    Conectar a la base de datos PostgreSQL:
        Conexi n = psycopg2.connect(
            dbname="ejer2", user="postgres", password="kuto", host="localhost", port="5432"
        )
        Crear cursor = Conexi n.cursor()

    Crear tabla de tareas si no existe:
        Ejecutar consulta SQL para crear tabla "tareas":
            id (clave primaria, serial)
            descripcion (texto, no nulo)
            fecha_vencimiento (texto, no nulo)
            completada (booleano, por defecto FALSE)
        Confirmar cambios en la base de datos

    Definir funci n agregar_tarea(descripcion, fecha_vencimiento):
        Ejecutar consulta SQL para insertar una nueva tarea con la descripci n y fecha de vencimiento
        Confirmar cambios en la base de datos
        Imprimir "Tarea agregada."

    Definir funci n marcar_completada(tarea_id):
        Ejecutar consulta SQL para actualizar la tarea con el id proporcionado y marcarla como completada
        Confirmar cambios en la base de datos
        Imprimir "Tarea marcada como completada."

    Definir funci n listar_tareas():
        Ejecutar consulta SQL para seleccionar todas las tareas
        Para cada tarea en el resultado:
            Imprimir el ID, descripci n, fecha de vencimiento y estado de completada

    Definir funci n menu():
        Imprimir las opciones disponibles:
            1. Agregar Tareas
            2. Marcar Tarea como Completada
            3. Ver Tareas y Estados
            4. Salir

    Mientras la opci n no sea 4:
        Llamar a la funci n menu()
        Solicitar al usuario que ingrese una opci n
        Si la opci n es 1:
            Solicitar descripci n y fecha de vencimiento para agregar una tarea
            Llamar a la funci n agregar_tarea(descripci n, fecha_vencimiento)
        Si la opci n es 2:
            Llamar a la funci n listar_tareas()
            Solicitar al usuario el ID de la tarea a marcar como completada
            Llamar a la funci n marcar_completada(tarea_id)
        Si la opci n es 3:
            Llamar a la funci n listar_tareas()

    Cerrar cursor y conexi n a la base de datos

Fin
```

VI. CODIGO PYTHON

```python
import psycopg2
import os
import datetime
import time
import re
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

conn = psycopg2.connect(
    dbname="Retra1",
    user="emerson",
    password="kuto",
    host="localhost",
    port="5432"
)

cursor = conn.cursor()

create_table_query = '''
    CREATE TABLE IF NOT EXISTS bitacora (
        accion VARCHAR(255)  NULL
    );
'''
cursor.execute(create_table_query)
conn.commit()

def procesar_nombre_tabla(usuario):
    """Limpia el nombre del usuario para que sea seguro usarlo como nombre de tabla."""
    return re.sub(r'\W+', '', usuario)
def crear_tabla(usuario):
    # Aseg rate de que el nombre de la tabla sea seguro para usar en SQL
    usuario = procesar_nombre_tabla(usuario) # Elimina caracteres no alfanum ricos
    create_table_query = f'''
        CREATE TABLE IF NOT EXISTS {usuario} (
        usuario VARCHAR(50),
        tarea VARCHAR(50),
        fecha VARCHAR(50),
        hora VARCHAR(50),
        accion VARCHAR(50)
    );
    '''
    cursor.execute(create_table_query)
    conn.commit()


def menu():
```

```
    print("QUE DESEA REALIZAR")
    print("1. Ingresar Informacion de Usuario")
    print("2. Ejecutar Programa")
    print("3. Borrar Informaci n")
    print("4. Historial de Usuarios")
    print("5. Salir")

def menu_usuario():
    print("1. Agregar Tarea")
    print("2. Modificar Tarea")
    print("3. Eliminar Tarea")
    print("4. Ver Tareas")
    print("5. Salir")

def ingresar_entero_tarea(mensaje):
    while True:
        try:
            valor = int(input(mensaje))
            if valor > 0 and valor < 5:
                return valor
            else:
                print("El valor debe ser positivo")
        except ValueError:
            print("Entrada inv lida. Por favor, ingrese un n mero entero positivo entre 1 y 4.")

def ingresar_entero_positivo(mensaje):
    while True:
        try:
            valor = int(input(mensaje))
            if valor > 0 and valor < 6:
                return valor
            else:
                print("El valor debe ser positivo entre 1 y 5.")
        except ValueError:
            print("Entrada inv lida. Por favor, ingrese un n mero entero positivo entre 1 y 5.")

def ingresar_string(mensaje):
    while True:
        try:
            valor = str(input(mensaje))
            if valor.isalpha():
                return valor
            else:
                print("El valor debe ser una cadena de caracteres.")
        except ValueError:
            print("Entrada inv lida. Por favor, ingrese un string.")

def ingresar_entero_borrar(mensaje):
    while True:
        try:
            valor = int(input(mensaje))
            if valor > 0 and valor < 4:
                return valor
            else:
                print("El valor debe ser positivo")
        except ValueError:
            print("Entrada inv lida. Por favor, ingrese un n mero entero positivo entre 1 y 3.")

def hora_ejecucion():
    hora = datetime.datetime.now().strftime("%H:%M:%S")
    return hora
def fecha_ejecucion():
    fecha = datetime.datetime.now().strftime("%Y-%m-%d")
    return fecha

def borrar_tabla(usuario):
    # Sanitize table name
    usuario = re.sub(r'\W+', '', usuario)
    drop_table_query = f'DROP TABLE IF EXISTS {usuario};'
    cursor.execute(drop_table_query)
    conn.commit()

def historial_usuario(usuario):
    # Sanitize table name
    usuario = re.sub(r'\W+', '', usuario)
    try:
        cursor.execute(f"SELECT * FROM {usuario}")
        registros = cursor.fetchall()
        if registros:
            for registro in registros:
                print(registro)
        else:
            print("No hay registros para este usuario")
    except psycopg2.Error as e:
        print(f"Error al consultar el historial: {e}")

def agregar_tarea(usuario,user, tarea, fecha, hora, accion):
    insert_query = f"INSERT INTO {usuario} (usuario, tarea, fecha, hora, accion) VALUES (%s, %s, %s, %s, %s)"
    cursor.execute(insert_query, (user, tarea, fecha, hora, accion))
    conn.commit()
    with open(f"{archivos_txt}/{user}.txt", "a") as archivo:
        archivo.write(f"{tarea} - {fecha} - {hora} - {accion}\n")


def modificar_tarea(usuario, tarea_original, nueva_tarea, fecha, hora, accion):
    usuario = procesar_nombre_tabla(usuario)
    update_query = f"UPDATE {usuario} SET tarea = %s, fecha = %s, hora = %s, accion = %s WHERE tarea = %s"
    cursor.execute(update_query, (nueva_tarea, fecha, hora, accion, tarea_original))
    conn.commit()

def eliminar_tarea(usuario, tarea_original):
    usuario = procesar_nombre_tabla(usuario)  # Aseg rate de limpiar el nombre del usuario
    delete_query = f"DELETE FROM {usuario} WHERE tarea = %s"
    cursor.execute(delete_query, (tarea_original,))
    conn.commit()


def ver_tareas(usuario):
    cursor.execute("""SELECT table_name FROM information_schema.tables WHERE table_schema
```

```
    ='public'""")
    tables = cursor.fetchall()
    for table in tables:
        print(table[0])
    usuario = ingresar_string("Ingrese el nombre de la tabla: ")
    select_query = f"SELECT * FROM {usuario}"
    cursor.execute(select_query)
    historial = cursor.fetchall()
    print(historial)
    return cursor.fetchall()

def listar_tareas(usuario):
    """Obtiene y muestra todas las tareas para un usuario espec fico."""
    usuario = procesar_nombre_tabla(usuario)
    select_query = f"SELECT tarea, fecha, hora, accion FROM {usuario}"
    cursor.execute(select_query)
    tareas = cursor.fetchall()

    print("\nTareas disponibles:")
    for i, tarea in enumerate(tareas, start=1):
        print(f"{i}. {tarea[0]} - Fecha: {tarea[1]} Hora: {tarea[2]} Estado: {tarea[3]}")

    return tareas

def acutualizar_bitacora(accionbitacora):
    insert_query = f"INSERT INTO bitacora (accion) VALUES (%s)"
    cursor.execute(insert_query, (accionbitacora,))
    conn.commit()

def listar_usuarios():
    cursor.execute("""SELECT table_name FROM information_schema.tables WHERE table_schema='public'""")
    tables = cursor.fetchall()
    return tables

opcion = 0
archivos_txt = "/home/emerson/Documentos/Tareas_Proyectos_2/Retra_1/Archivos_txt"

if not os.path.exists(archivos_txt):
    os.makedirs(archivos_txt)
    print(f"El directorio {archivos_txt} ha sido creado.")
else:
    print(f"El directorio {archivos_txt} ya existe.")

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

    if opcion == 1:
        print("Ingrese su nombre de usuario: ")
        usuario = ingresar_string("Ingrese su nombre de usuario: ")
        crear_tabla(usuario)
        print("Usuario creado exitosamente.")
        accionbitacora = f"El usuario {usuario} ha sido creado exitosamente a la fecha {fecha_ejecucion()} y a la hora {hora_ejecucion()}"
        acutualizar_bitacora(accionbitacora)
    if opcion == 2:
        menu_usuario()
        opcion_usuario = ingresar_entero_tarea("Ingrese una opci n: ")
        if opcion_usuario == 1:

            usuario = ingresar_string("Ingrese su nombre de usuario: ")
            user = usuario
            tarea = input("Ingrese la tarea: ")
            fecha = fecha_ejecucion()
            hora = hora_ejecucion()
            accion = "Agregar"
            agregar_tarea(usuario,user, tarea, fecha, hora, accion)
            accionbitacora = f"El usuario {user} ha agregado la tarea {tarea} a la fecha {fecha_ejecucion()} y a la hora {hora_ejecucion()}"
            acutualizar_bitacora(accionbitacora)
        if opcion_usuario == 2:
            usuario = ingresar_string("Ingrese su nombre de usuario: ")
            tareas = listar_tareas(usuario)  # Lista las tareas para este usuario

            if not tareas:
                print("No hay tareas para este usuario.")
            else:
                # Mostrar las tareas al usuario
                try:
                    seleccion = int(input("Seleccione una tarea: "))
                    seleccion -= 1

                    if 0 <= seleccion < len(tareas):
                        tarea_original = tareas[seleccion][0]
                        nueva_tarea = input("Ingrese el nuevo nombre de la tarea (o presione Enter para mantenerla igual): ")
                        nueva_tarea = nueva_tarea if nueva_tarea else tarea_original
                        fecha = fecha_ejecucion()
                        hora = hora_ejecucion()
                        accion = "completada"

                        modificar_tarea(usuario, tarea_original, nueva_tarea, fecha, hora, accion)
                        accionbitacora = f"El usuario {usuario} ha modificado la tarea {tarea_original} a la fecha {fecha_ejecucion()} y a la hora {hora_ejecucion()}"
                        acutualizar_bitacora(accionbitacora)
                        print("La tarea ha sido modificada exitosamente.")
                    else:
                        print("Selecci n inv lida. Aseg rate de elegir un n mero de la lista.")
                except ValueError:
                    print("Entrada inv lida. Por favor, ingresa un n mero.")
```

```
253
254         if opcion_usuario == 3:
255             usuario = ingresar_string("Ingrese
    su nombre de usuario: ")
256             tareas = listar_tareas(usuario)
257             user = usuario
258             if not tareas:
259                 print("No hay tareas para este
    usuario.")
260             else:
261
262                 try:
263                     seleccion = int(input("
    Seleccione una tarea: "))
264                     seleccion -= 1
265
266                     if 0 <= seleccion < len(
    tareas):
267                         tarea_original = tareas[
    seleccion][0]
268                         nueva_tarea =
    tarea_original
269                         fecha = fecha_ejecucion
    ()
270                         hora = hora_ejecucion()
271                         accion = "eliminada"
272
273                         eliminar_tarea(usuario,
    tarea_original)
274                         accionbitacora = f"El
    usuario {user} ha eliminado la tarea {
    tarea_original} a la fecha {fecha_ejecucion
    ()} y a la hora {hora_ejecucion()}"
275                         acutualizar_bitacora(
    accionbitacora)
276                         print("La tarea ha sido
    eliminada exitosamente.")
277                     else:
278                         print("Selecci n
    inv lida. Aseg rate de elegir un n mero
    de la lista.")
279                 except ValueError:
280                     print("Entrada inv lida.
    Por favor, ingresa un n mero.")
281
282         if opcion_usuario == 4:
283             usuario = ingresar_string("Ingrese
    su nombre de usuario: ")
284             user = usuario
285             tareas = listar_tareas(usuario)
286             accionbitacora = f"El usuario {user}
     ha visto las tareas a la fecha {
    fecha_ejecucion()} y a la hora {
    hora_ejecucion()}"
287             acutualizar_bitacora(accionbitacora)
288             if not tareas:
289                 print("No hay tareas para este
    usuario.")
290             else:
291                 continue
292
293         if opcion_usuario == 5:
294             accionbitacora = f"Se salio del
    programa a la fecha {fecha_ejecucion()} y a
    la hora {hora_ejecucion()}"
295             acutualizar_bitacora(accionbitacora)
296             break
297
298     if opcion == 3:
299         print("Ingrese su nombre de usuario: ")
300         lista_usuarios = listar_usuarios()
301         usuario = ingresar_string("Ingrese su
    nombre de usuario: ")
302         user = usuario
303         borrar_tabla(usuario)
304         accionbitacora = f"El usuario {user} ha
    borrado su tabla a la fecha {fecha_ejecucion
    ()} y a la hora {hora_ejecucion()}"
305         acutualizar_bitacora(accionbitacora)
306     if opcion == 4:
307         print("Ingrese su nombre de usuario: ")
308         usuario = ingresar_string("Ingrese su
    nombre de usuario: ")
309         historial_usuario(usuario)
310         accionbitacora = f"El usuario {user} ha
    visto su historial a la fecha {
    fecha_ejecucion()} y a la hora {
    hora_ejecucion()}"
311         acutualizar_bitacora(accionbitacora)
312     if opcion == 5:
313         print("Saliendo del programa...")
314         accionbitacora = f"Se salio del programa
     a la fecha {fecha_ejecucion()} y a la hora
    {hora_ejecucion()}"
315         acutualizar_bitacora(accionbitacora)
316         break
317
318 cursor.close()
319 conn.close()
320
321
```

VII. METIGACION DE ERRORES

- **Error de tipo al ingresar una opción que no es un número entero.**
- **Selección de una opción no válida en el menú.**
- **Ingreso de un número muy grande que podría causar un tiempo de espera prolongado en la detección de números primos o perfectos.**
- **Error de división por cero si se intenta verificar el número cero como primo o perfecto.**
- **Fallo al usar caracteres especiales en la entrada de palíndromo (dependiendo de la lógica esperada).**
- **Entrada de texto en lugar de números para las opciones 2 y 3.**
- **Posibles problemas de codificación si se ingresan caracteres no ASCII en algunas configuraciones.**

VIII. DIAGRAMA DE GANTT

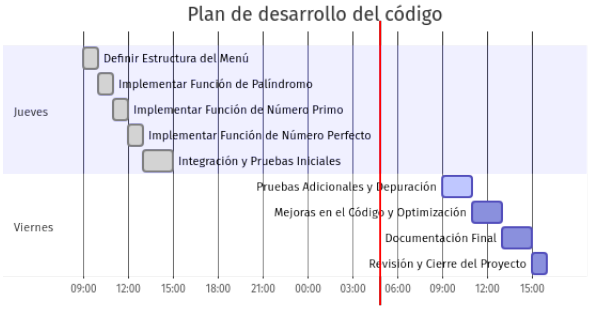

Figura 2

Fuente: Elaboracion Propia, 2024

IX. REPOSITORIO GITHUB

X. VIDEO GOOGLE DRIVE