import tkinter as tk

from tkinter import ttk, messagebox

from DAO import DAO

from Usuario import Usuario

import os

from proyecto import RegistrarTiqueApp

# Importar las librerías necesarias

dao = DAO()

# Crear una instancia del objeto DAO

ventana = tk.Tk()

ventana.title("Sistema de Tiques")

ventana.geometry("800x600")

# Crear la ventana principal de la aplicación

entry\_nombre = None

entry\_password = None

# Definir variables globales para entry\_nombre y entry\_password

def mostrar\_opciones():

# Definir y colocar los widgets necesarios

label\_opciones = tk.Label(ventana, text="¿Qué quieres hacer?")

label\_opciones.pack(pady=10)

# Función para mostrar las opciones disponibles al usuario

btn\_registrar = tk.Button(ventana, text="Registrar", command=mostrar\_formulario\_registrar)

btn\_registrar.pack(pady=5)

# Botón para registrar un nuevo usuario

# Botón para iniciar sesión

btn\_iniciar\_sesion = tk.Button(ventana, text="Iniciar Sesión", command=mostrar\_formulario\_iniciar\_sesion)

btn\_iniciar\_sesion.pack(pady=5)

# Botón para iniciar sesión

def mostrar\_formulario\_registrar():

# ... (código para mostrar el formulario de registro)

# Función para mostrar el formulario de registro de nuevos usuarios

def mostrar\_formulario\_iniciar\_sesion():

# Función para mostrar el formulario de inicio de sesión

def limpiar\_ventana():

# Función para limpiar la ventana y ocultar los widgets

# Función para registrar un nuevo usuario en la base de datos

def registrar\_usuario(nombre, contrasenia, rol\_elegido):

# Función para iniciar sesión en la aplicación

def iniciar\_sesion(nombre, contrasenia):

# Mostrar las opciones al iniciar la aplicación

mostrar\_opciones()

# Ejecutar el bucle principal de la aplicación

ventana.mainloop()

#indica que se debe instalar la biblioteca "mysql-connector-python" utilizando pip

pip install mysql-connector-python

import mysql.connector

import credenciales

from Tique import Tique

from Cliente import Cliente

from Usuario import Usuario

class DAO:

# Define una clase llamada "DAO" con un constructor

def \_\_init\_\_(self):

pass

def conectar(self):

# Establece la conexión con la base de datos

self.\_\_conexion = mysql.connector.connect(\*\*credenciales.get\_credenciales())

self.\_\_cursor = self.\_\_conexion.cursor()

def cerrar(self):

# Cierra la conexión con la base de datos

self.\_\_conexion.commit()

self.\_\_conexion.close()

def registrarTique(self, tique: Tique, cliente: Cliente):

# Función para registrar un nuevo tique en la base de datos

def obtenerTiques(self):

# Función para obtener todos los tiques de la base de datos

def obtenerTique(self, id\_tique: int):

# Función para obtener un tique específico por su ID

def eliminarTique(self, id\_tique: int):

# Función para eliminar un tique de la base de datos por su ID

def actualizarTique(self, tique: Tique):

# Función para actualizar la información de un tique en la base de datos

def registrar\_usuario(self, usuario: Usuario):

# Función para registrar un nuevo usuario en la base de datos

def verificar\_credenciales(self, nombre: str, contraseña: str) -> bool:

# Función para verificar las credenciales de inicio de sesión de un usuario

def obtenerNombreArea(self, id\_area: int) -> str:

# Función para obtener el nombre de un área por su ID

def obtenerNombreTipo(self, id\_tipo: int) -> str:

# Función para obtener el nombre de un tipo por su ID

def obtenerRUTCliente(self, cliente\_id):

# Función para obtener el RUT de un cliente por su ID

def obtenerNombreCriticidad(self, id\_criticidad: int) -> str:

# Función para obtener el nombre de una criticidad por su ID

def obtenerTiquesPorRutCliente(self, rut):

# Función para obtener todos los tiques asociados a un cliente por su RUT

import tkinter as tk

from tkinter import ttk

from DAO import DAO

from Tique import Tique

from Cliente import Cliente

from datetime import date

# Importa módulos y clases necesarios

class RegistrarTiqueApp:

def \_\_init\_\_(self, root):

# Constructor de la clase RegistrarTiqueApp

def actualizar\_id\_area(self, \*args):

# Actualiza el ID del área seleccionada

def actualizar\_id\_criticidad(self, \*args):

# Actualiza el ID de la criticidad seleccionada

def actualizar\_id\_tipo\_tique(self, \*args):

# Actualiza el ID del tipo de tique seleccionado

def registrar\_tique(self):

# Registra un nuevo tique en la base de datos

def obtener\_tiques(self):

# Obtiene todos los tiques de la base de datos y los muestra en el treeview

def buscar\_tiques\_por\_rut(self):

# Busca los tiques asociados a un cliente por su RUT

def mostrar\_tique\_seleccionado(self, event):

# Muestra la información del tique seleccionado en los campos de entrada

# Crear una instancia de Tkinter y la aplicación Registrar Tique

if \_\_name\_\_ == '\_\_main\_\_':

# Inicializa la aplicación y la ventana principal de Tkinter