DESAFÍO CRUD PARA LA EXPO-CARRERAS 2024

Diagrama de Flujo General:

1. **Inicio:**
    - ○ Importar las librerías necesarias: Tkinter, MySQL connector, etc.
    - ○ Conectar a la base de datos MySQL.
    - ○ Crear la ventana principal de la aplicación.
    - ○ Definir los elementos de la interfaz: etiquetas, campos de texto, botones, radio buttons, títulos,etc.
    - ○ Cargar la lista de carreras desde la base de datos y poblar los radio buttons.
2. **Acciones del Usuario:**
    - ○ **Crear:**
        - ■ El usuario ingresa los datos en los campos de texto.
        - ■ Al hacer clic en el botón "Crear", se inserta un nuevo registro en la tabla de personas en la base de datos.
    - ○ **Leer:**
        - ■ Se consulta la base de datos y se muestran todos los registros en una tabla o lista.
    - ○ **Actualizar:**
        - ■ El usuario selecciona un registro a modificar.
        - ■ Se muestran los datos del registro seleccionado en los campos de texto.
        - ■ El usuario realiza los cambios y al hacer clic en "Actualizar", se modifican los datos en la base de datos.
    - ○ **Eliminar:**
        - ■ El usuario selecciona un registro a eliminar.
        - ■ Al confirmar la eliminación, se elimina el registro de la base de datos.
3. **Fin:**
    - ○ Cerrar la conexión a la base de datos.
    - ○ Cerrar la ventana principal de la aplicación.

Diseño de las Tablas en MySQL:

Tabla personas:

- id_persona (INT, PK, AI): Identificador único de cada persona.
- apellido (VARCHAR): Apellido de la persona.
- nombre (VARCHAR): Nombre de la persona.
- dni (VARCHAR): Número de documento de identidad.
- telefono (VARCHAR): Número de teléfono.
- correo (VARCHAR): Correo electrónico.
- domicilio (VARCHAR): Dirección.
- ciudad (VARCHAR): Ciudad.

- instagram (VARCHAR): Usuario de Instagram.
- id_carrera (INT): Clave foránea que referencia a la tabla de carreras.

Tabla carreras:

- id_carrera (INT, PK, AI): Identificador único de cada carrera.
- nombre_carrera (VARCHAR): Nombre de la carrera.

Relación entre las tablas:

La tabla `personas` tiene una relación de uno a muchos con la tabla `carreras`.
Esto significa que una persona puede tener una sola carrera, pero una carrera puede tener muchas personas.

Consideraciones Adicionales:

- **Validación de datos:** Es importante validar los datos ingresados por el usuario para evitar errores. Por ejemplo, verificar que el formato del correo electrónico sea correcto, que el DNI tenga la longitud adecuada, etc.
- **Mensajes de error:** Mostrar mensajes claros y concisos al usuario en caso de errores, como por ejemplo cuando se intenta ingresar un registro duplicado. ● **Interfaz gráfica:** Diseñar una interfaz gráfica intuitiva y fácil de usar, utilizando los widgets adecuados de Tkinter.
- **Optimización del código:** Escribir código eficiente y optimizado para evitar problemas de rendimiento, especialmente cuando se trabaja con grandes cantidades de datos.

<u>Ejemplo de cómo tomar valores de los entry e insertarlos en la bdd:</u>

```
import tkinter as tk import
mysql.connector

# Datos de conexión a la base de datos (reemplázalos con tus datos)

  mydb = mysql.connector.connect(
  host="tu_propio_host",
  user="tu_propio_usuario",
  password="tu_contraseña!!!!",
  database="nombre_de_tu_base_de_datos)

mycursor = mydb.cursor()

# Función para insertar un nuevo registro en la base de datos

def insertar_persona():
        apellido = entry_apellido.get()
        nombre = entry_nombre.get() # ... Obtener los valores de los demás campos
        sql = "INSERT INTO personas (apellido, nombre, ...) VALUES (%s, %s, ...)"
        val = (apellido, nombre, ...)
        mycursor.execute(sql, val)
        mydb.commit()
```

```
        print("Registro insertado correctamente.")

 # Crear la ventana principal ventana = tk.Tk()

 ventana.title("CRUD de Personas")

 # Crear los elementos de la interfaz

label_apellido = tk.Label(ventana, text="Apellido:")
 #entry_apellido = tk.Entry(ve….. Crear los demás elementos de la interfaz para los otros campos

 # Botón para insertar
boton_insertar = tk.Button(ventana, text="Insertar", command=insertar_persona)

 # Empaquetar los elementos en la ventana ... (Utilizar grid, pack o place para organizar los elementos)

# Iniciar el bucle principal de la aplicación ventana.mainloop()
```


Conectando las Ventanas:

Para navegar entre las ventanas, podemos utilizar una técnica común en Tkinter: crear múltiples ventanas y mostrar una u otra según sea necesario. Aquí te presento una estructura básica:

Python

```
import tkinter as tk

# Ventana de alta
def ventana_alta():
    # ... (código de la ventana de alta)
    boton_ver_todos = tk.Button(ventana_alta, text="Ver Inscriptos", command=ventana_leer)
    # ...

# Ventana de lectura
def ventana_leer():
    # ... (código de la ventana de lectura)
    # ...

# Crear la ventana principal (opcional, si quieres una ventana principal)
ventana_principal = tk.Tk()
```

```
# Mostrar inicialmente la ventana de alta
ventana_alta()

ventana_principal.mainloop()
```


Conectando las Ventanas:

Para navegar entre las ventanas, podemos utilizar una técnica común en Tkinter: crear múltiples ventanas y mostrar una u otra según sea necesario.Ejeplo:

```
import tkinter as tk

# Ventana de alta
def ventana_alta():
    # ... (código de la ventana de alta)
    boton_ver_todos = tk.Button(ventana_alta, text="Ver Inscriptos", command=ventana_leer)
    # ...

# Ventana de lectura
def ventana_leer():
    # ... (código de la ventana de lectura)
    # ...

# Crear la ventana principal (opcional, si quieres una ventana principal)
ventana_principal = tk.Tk()
```

```
# Mostrar inicialmente la ventana de alta
ventana_alta()

ventana_principal.mainloop()
```


Podemos utilizar un `Checkbutton` para permitir al usuario seleccionar múltiples carreras y realizar un filtro más flexible. Ejemplo:

```
import tkinter as tk

def ventana_leer():
    ventana_leer = tk.Toplevel()
    ventana_leer.title("Inscriptos")

    # Obtener los datos de la base de datos
     datos = obtener_datos_de_la_base_datos() # Función para consultar la base de datos

    # Crear una Listbox
    listbox = tk.Listbox(ventana_leer)

    # Obtener la lista de carreras únicas
    carreras_unicas = set(dato[1] for dato in datos)
```

```
# Crear los Checkbuttons
checkbuttons = {}
for carrera in carreras_unicas:
    var = tk.IntVar()
    checkbutton = tk.Checkbutton(ventana_leer, text=carrera, variable=var)
    checkbutton.pack()
    checkbuttons[carrera] = var

# Botón para filtrar
boton_filtrar = tk.Button(ventana_leer, text="Filtrar",
command=filtrar_por_carrera) boton_filtrar.pack()

# Función para filtrar los datos
def filtrar_por_carrera():
    carreras_seleccionadas = [carrera for carrera, var in checkbuttons.items() if
    var.get()] datos_filtrados = [dato for dato in datos if dato[1] in
    carreras_seleccionadas] listbox.delete(0, tk.END)
    for dato in datos_filtrados:
        listbox.insert(tk.END, dato)

# Mostrar todos los datos inicialmente
filtrar_por_carrera()

listbox.pack()
```

Podemos añadir en cada sector de las interfaces los frames para que quede más ordenada la información que vamos a relevar.
Si no quisieramos que seleccione más de una carrera para filtrar los aspirantes, podemos emplear el desplegable como muestra el siguiente ejemplo:


```
import tkinter as tk

def ventana_leer():
ventana_leer = tk.Toplevel()
ventana_leer.title("Inscriptos") # ... (código para obtener los datos de la base de datos) #
Crear el menú desplegable
carreras = ["Ingeniería", "Medicina", "Derecho", "Contabilidad"]
carrera_seleccionada = tk.StringVar()
carrera_seleccionada.set(carreras[0])
menu_desplegable = tk.OptionMenu(ventana_leer, carrera_seleccionada, *carreras)
menu_desplegable.pack() # Botón para filtrar
boton_filtrar = tk.Button(ventana_leer, text="Filtrar", command=filtrar_por_carrera)
boton_filtrar.pack()

def filtrar_por_carrera():
carrera_elegida = carrera_seleccionada.get()
```

... (código para filtrar los datos según la carrera elegida) # ... (resto del código) 7