TECNO 3F | Programá tu futuro

Municipalidad de Tres de Febrero

www.tresdefebrero.gov.ar/**tecno3f**

#ProgramáTuFuturo

TECNO3F | Programá tu futuro

3F | Municipalidad de Tres de Febrero

Municipalidad de Tres de Febrero

TECNO 3F | Programá tu futuro

PYTHON INTERMEDIO

¡Les damos la bienvenida!

Municipalidad de Tres de Febrero

TECNO 3F | Programá tu futuro

CRUD II

CLASE 12

PARTE LOGICA

TECNO3F | Programá tu futuro

3F | Municipalidad de Tres de Febrero

Tomando Valores

Agregamos 2 variables a los input

```
def input_form(self):
    self.nombre = tk.StringVar()
    self.entry_nombre = tk.Entry(self,textvariable=self.nombre)
    self.entry_nombre.config(width=50, state='disabled',font=('Arial',12))
    self.entry_nombre.grid(row= 0, column=1,padx=10,pady=10, columnspan='2')

    self.duracion = tk.StringVar()
    self.entry_duracion = tk.Entry(self,textvariable=self.duracion)
    self.entry_duracion.config(width=50, state='disabled',font=('Arial',12))
    self.entry_duracion.grid(row= 1, column=1,padx=10,pady=10, columnspan='2')
```

Con estas variables podremos capturar y mostrar lso valores que necesitemos

Las mismas las pondremos en el metodo de bloquear campos con un .set(“) para limpiarlas

Limpiando al cancelar

Agregamos las 2 variables a la funcion bloquear

```
def bloquear_campos(self):
    self.entry_nombre.config(state='disabled')
    self.entry_duracion.config(state='disabled')
    self.entry_genero.config(state='disabled')
    self.btn_modi.config(state='disabled')
    self.btn_cance.config(state='disabled')
    self.nombre.set('')
    self.duracion.set('')
    self.btn_alta.config(state='normal')
```

Les seteamos los datos en blanco

de esta manera se limpiaran los input al presionar sobre cancelar

Listado Peliculas

Vamos a importar el TTK - from tkinter import ttk

```
def mostrar_tabla(self):

    self.tabla = ttk.Treeview(self, columns=('Nombre','Duración','Genero'))
    self.tabla.grid(row=4,column=0,columnspan=4)
    self.tabla.heading('#0',text='ID')
    self.tabla.heading('#1',text='Nombre')
    self.tabla.heading('#2',text='Duración')
    self.tabla.heading('#3',text='Genero')
```

con esto tendriamos la tabla donde enviar los datos de las peliculas

ahora armamos los botones edicar y borrar

Editar y borrar

Estos botones tendran accion sobre la tabla de peliclas

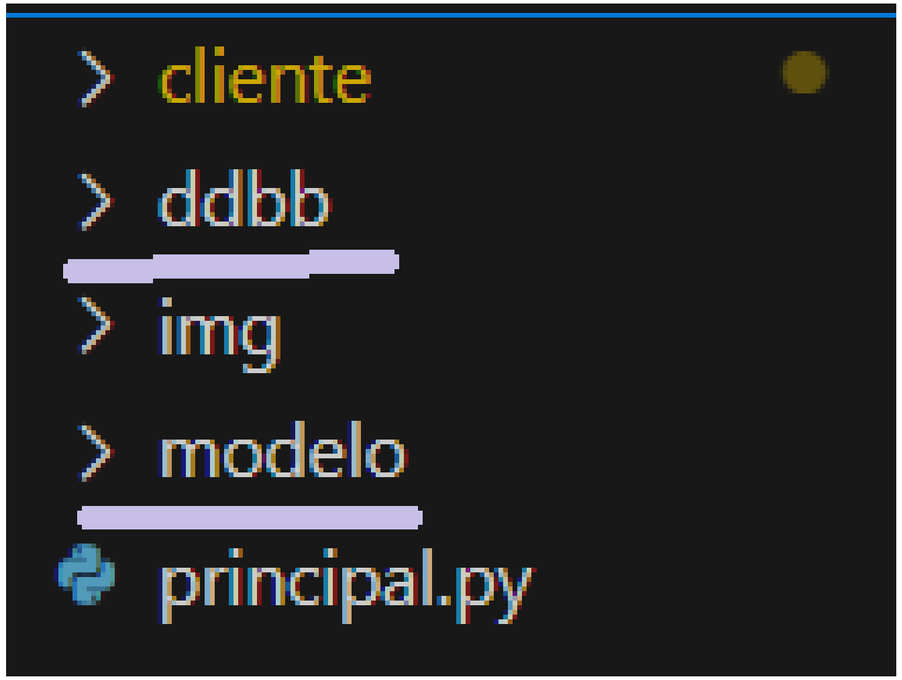
```
self.btn_editar = tk.Button(self, text='Editar')
self.btn_editar.config(width= 20,font=('Arial', 12,'bold'),fg ='#FFFFFF' , bg='#1C500B',
cursor='hand2',activebackground='#3FD83F',activeforeground='#000000')
self.btn_editar.grid(row= 5, column=0,padx=10,pady=10)

self.btn_delete = tk.Button(self, text='Borrar')
self.btn_delete.config(width= 20,font=('Arial', 12,'bold'),fg ='#FFFFFF' , bg='#A90A0A',
cursor='hand2',activebackground='#F35B5B',activeforeground='#000000')
self.btn_delete.grid(row= 5, column=1,padx=10,pady=10)
```

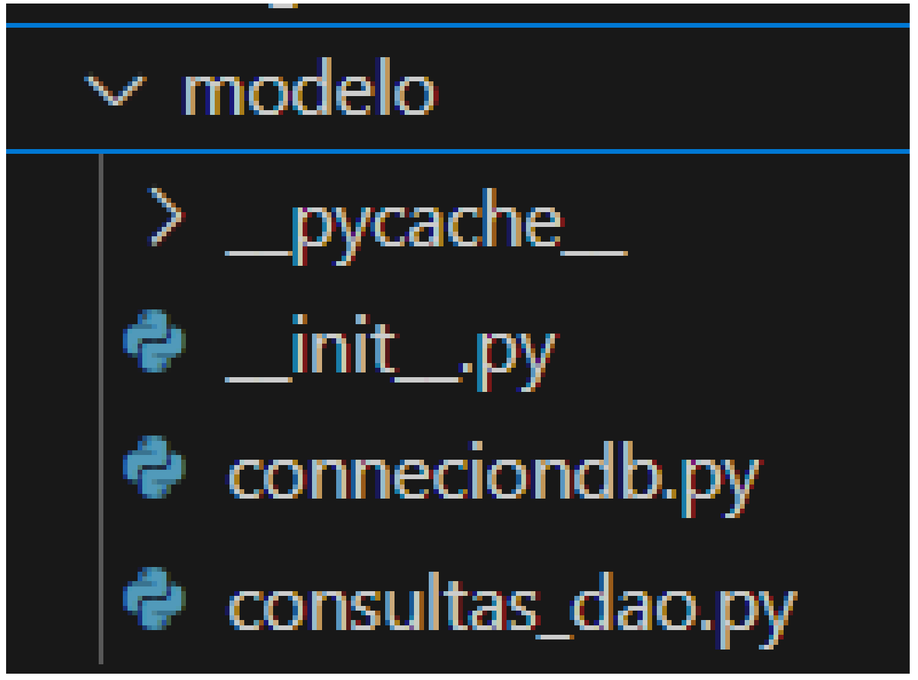
como la lista se carga todo el tiempo, deben estar continuamente habilitados

estos botones van en el metodo de la tabla

Creando la db y la coneccion

Vamos a crear dos subcarpetas





La carpeta de modelo, contendre la coneccion y las consultas

La carpeta ddbb , contendra el archivo de la base de datos de sqlite3

Clase Coneccion

En el archivo conecciondb.py

```python
import sqlite3

class ConeccionDB():
    def __init__(self):
        self.base_datos = 'ddbb/peliculas.db'
        self.conexion = sqlite3.connect(self.base_datos)
        self.cursor = self.conexion.cursor()


    def cerrar_con(self):
        self.conexion.commit()
        self.conexion.close()
```

Creamos la clase coneccion y el metodo para cerrarla

Creando la tabla

En el archivo consultas_dao.py

Podemos crear un script para que nos cree la tabla o tablas que necesitemos, es importante que para que no nos surgen errores inesperados, usemos esto en un try, a si mismo , podriamos crear otra funcionalidad que borre la tabla

```python
from .conneciondb import ConeccionDB

def crear_tabla():
    conn = ConeccionDB()

    sql = '''
            CREATE TABLE IF NOT EXISTS Genero(
            ID INTEGER NOT NULL,
            Nombre VARCHAR(50),
            PRIMARY KEY (ID AUTOINCREMENT)
            );

            CREATE TABLE IF NOT EXISTS Peliculas(
            ID INTEGER NOT NULL,
            Nombre VARCHAR(150),
            Duacion VARCHAR(4),
            Genero INTEGER,
            PRIMARY KEY (ID AUTOINCREMENT),
            FOREIGN KEY (Genero) References Genero(ID)
            );
            '''
    try:
        conn.cursor.execute(sql)
        conn.cerrar_con()

    except:
        pass
```

Clase Pelicula

En el archivo consultas_dao.py

```python
class Peliculas:
    def __init__(self, nombre,duracion, genero):
        self.id_peliculas = None
        self.nombre = nombre
        self.duracion = duracion
        self.genero = genero

    def __str__(self):
        return f'Pelicula[{self.nombre},{self.duracion}, {self.genero}]'
```

Con esto nos ayudaremos a guardar las peliculas

Guardar entrada

En el archivo consultas_dao.py

```
def guardar_peli(pelicula):
    conn = ConeccionDB()

    sql = f"""
            INSERT INTO Peliculas (Nombre,Duacion,Genero)
            VALUES('{pelicula.nombre}','{pelicula.duracion}',{pelicula.genero});
            """

    conn.cursor.execute(sql)
    conn.cerrar_con()
```

Guardar entrada

En el archivo vista.py

```
def guardar_campos(self):
    pelicula = Peliculas(
        self.nombre.get(),
        self.duracion.get(),
        self.entry_genero.current()
    )

    guardar_peli(pelicula)
    self.bloquear_campos()
```

Este metodo forma parte de la clase Frame, recuerden agregar esta funcionalidad al boton de guardar con el command= self.guardar_campos

Tambien recuerden importar tanto la clase como la funcion

Para Actualizar la lista al agregar una nueva entrada agregamos antes de bloquear los cmapos

self.mostrar_tabla()

Listar tabla

En el archivo consultas_dao.py

```python
def listar_peliculas():
    conn = ConeccionDB()
    listar_peliculas = []
    sql = """
            SELECT * FROM Peliculas as p
            inner join Genero as g
            on p.Genero = g.ID;
          """

    try:
        conn.cursor.execute(sql)
        listar_peliculas = conn.cursor.fetchall()
        conn.cerrar_con()

        return listar_peliculas
    except:
        pass
```

Creamos la consulta SQL con el inner para que nos cargue la info de ambas tablas, ya que no queremos que el usuario veo el ID del genero, sino la descripcion o nombre del mismo

Listar tabla

En el archivo vista.py

```
def mostrar_tabla(self):
    self.lista_p = listar_peliculas()
    self.tabla = ttk.Treeview(self, columns=('Nombre','Duración','Genero'))
    self.tabla.grid(row=4,column=0,columnspan=4)
    self.tabla.heading('#0',text='ID')
    self.tabla.heading('#1',text='Nombre')
    self.tabla.heading('#2',text='Duración')
    self.tabla.heading('#3',text='Genero')

    for p in self.lista_p:
        self.tabla.insert('',0,text=p[0],
                          values = (p[1],p[2],p[5]))

    self.btn_editar = tk.Button(self, text='Editar')
    self.btn_editar.config(width= 20,font=('Arial', 12,'bold'),fg ='#FFFFFF' , bg='#1C500B',
    cursor='hand2',activebackground='#3FD83F',activeforeground='#000000')
    self.btn_editar.grid(row= 5, column=0,padx=10,pady=10)
```

Podemos usar el metodo reverse, para cambiar el orden de la lista

self.lista_p.reverse()

Cargar ComboBox

En el archivo vista.py

```python
#aca limpiamos la lista de tuplas que nos retorna la funcion
x = listar_generos()
y = []
for i in x:
    y.append(i[1])

#concatenemos el nuevo array
self.generos = ['Seleccione uno'] + y
self.entry_genero = ttk.Combobox(self, state="readonly")
self.entry_genero['values'] = self.generos
self.entry_genero.current(0)
self.entry_genero.config(width=25, state='disabled',font=('Arial',12))
self.entry_genero.bind("<<ComboboxSelected>>")
self.entry_genero.grid(row= 2, column=1,padx=10,pady=10, columnspan='2')
```

La consulta a la DB es mas sencilla a la mostrada en Listar Tablas

```python
listar_generos = []
sql = """
        SELECT * FROM Genero
      """
```

Se armara una funcion igual solo que con esta query

supernado 10 registros

En el archivo vista.py

```
def mostrar_tabla(self):
    self.lista_p = listar_peliculas()
    self.lista_p.reverse() #para invertir el orden
    self.tabla = ttk.Treeview(self, columns=('Nombre','Duración','Genero'))
    self.tabla.grid(row=4,column=0,columnspan=4,sticky='nse')

    self.scroll = ttk.Scrollbar(self, orient='vertical', command=self.tabla.yview)
    self.scroll.grid(row=4, column=4,sticky='nse')
    self.tabla.configure(yscrollcommand= self.scroll.set)
```

Agregaremos estos comando al metodo del a tabla, lo cual nos creara una Scrollbar para poder navegar en nuestra tabla de manera vertical, si la misma supera los 10 registros

Editar Registro

En el archivo vista.py

```
def editar_registro(self):
    try:
        self.id_peli = self.tabla.item(self.tabla.selection())['text']

        self.nombre_peli_e = self.tabla.item(self.tabla.selection())['values'][0]
        self.dura_peli_e = self.tabla.item(self.tabla.selection())['values'][1]
        self.gene_peli_e = self.tabla.item(self.tabla.selection())['values'][2]

        self.habilitar_campos()
        self.nombre.set(self.nombre_peli_e)
        self.duracion.set(self.dura_peli_e)
        self.entry_genero.current(self.generos.index(self.gene_peli_e))

    except:
        pass
```

agregar esta funcion al boton editar

Editar Registro

En el archivo consultas_dao.py

```
def editar_peli(pelicula, id):
    conn = ConeccionDB()

    sql = f"""
            UPDATE Peliculas
            SET Nombre = '{pelicula.nombre}', Duacion = '{pelicula.duracion}',Genero = {pelicula.
            genero}
            WHERE ID = {id};
            """

    conn.cursor.execute(sql)
    conn.cerrar_con()
```

Editar Registro

En el archivo vista.py

```python
class Frame(tk.Frame):
    def __init__(self, root = None):
        super().__init__(root,width=480,height=320)
        self.root = root
        self.pack()
        self.id_peli = None
```

agregamos un id_peli al constructor para hacer la logica de guardar

```python
class Frame(tk.Frame):

    def guardar_campos(self):
        pelicula = Peliculas(
            self.nombre.get(),
            self.duracion.get(),
            self.entry_genero.current()
        )

        if self.id_peli == None:
            guardar_peli(pelicula)
        else:
            editar_peli(pelicula, int(self.id_peli))

        self.mostrar_tabla()
        self.bloquear_campos()
```

modificamos la funcion de guardar

Exterminar Registro

En el archivo vista.py

```
def eliminar_registro(self):
    try:
        self.id_peli = self.tabla.item(self.tabla.selection())['text']
        borrar_peli(int(self.id_peli))
        self.mostrar_tabla()
    except:
        pass
```

agregar esta funcion al boton borrar

Exterminar Registro

En el archivo consultas_dao.py

```
def borrar_peli(id):
    conn = ConeccionDB()

    sql = f"""
            DELETE FROM Peliculas
            WHERE ID = {id};
            """

    conn.cursor.execute(sql)
    conn.cerrar_con()
```

¡MUCHAS GRACIAS!

TECNO3F | Programá tu futuro

3F | Municipalidad de Tres de Febrero