

Entrega tarea M6-03

Buenas noches, primeramente saludarlo por ser un excelente profesor he aprendido mucho desde que inicie el curso y también he tenido varios dolores de cabeza pero es parte del proceso de aprendizaje, bueno intente en reemplazar el código sqlite3 al orm, pero varios días intentando, leyendo, investigando no pude llegar al cambio, así que bueno en las próximas capturas de pantallas, veras varias líneas de código comentadas por el intento fallido, así que le pido que me explique como seria reemplazar el código al orm.

En las próximas capturas hice varios cambios:

1. intente reemplazar el código sqlite2 al orm, por eso hay varios import en los inicios.

```
app.py x db.py
1 import sqlite3
2 from idlelib.colorizer import color_config
3 from tkinter import ttk
4 from tkinter import *
5 from sqlalchemy import Column, Integer, String
6 from sqlalchemy.dialects import sqlite
7 from sqlalchemy import create_engine
8 from sqlalchemy.orm import sessionmaker
9
10 import db
11 from db import session
12
13 """class Producto(db.Base):
14     __tablename__ = 'producto'
15     __table_args__ = {'sqlite_autoincrement': True}
16     id = Column(Integer, primary_key=True)
17     nombre = Column(String(100), nullable=False)
18     precio = Column(Integer, nullable=False)
19     categoria = Column(String(50), nullable=False)
20     cantidad = Column(Integer, nullable=False)"""
21
22 class VentanaPrincipal(): 1 usage
23     db = "database/productos.db"
24
25
26 def __init__(self, root):
27
28     #self.engine = create_engine('sqlite:///database/productos.db')
29     #self.Session = sessionmaker(bind=self.engine)
30
31     self.ventana = root #se cambia la variable root a ventana
32     self.ventana.title("App Gestor de Productos") #titulo de la ventana
33     self.ventana.resizable(1,1) #activa la redimension de la ventana
34     self.ventana.wm_iconbitmap("recursos/principal.ico") #icono de la ventana
35
36
37
38
39     #Creamos el contenedor principal(frame) se puede crear varios contenedores donde se puede
40     #almacenar varios widget dentro de un frame
41     frame = LabelFrame(self.ventana, text="Registrar un nuevo Producto") #le pasamos la variable self.ventana y le colocamos el titulo
42     frame.grid(row=0, column=0, pady=0, columnspan=10)#ubicacion de donde estara el frame fila 0, columna 0 con un margen de 20 y ocupa la columna 1,2 y 3
43
44     #Label Nombre
45     self.etiqueta_nombre = Label(frame, text="Nombre: ") #se crea la etiqueta nombre
46     self.etiqueta_nombre.grid(row=1, column=0) #ubicacion de la etiqueta
47
48
```

2. Creo el Label/Entry de categoría y cantidad así se muestra en la ventana principal, también cree dos funciones que era cambiar el tema de la app, pero no logre hacerlo en la ventana completa, así que vera esa función a media.

```
app.py x db.py
class VentanaPrincipal():
    def __init__(self, root):

        #Entry caja donde se coloca la informacion
        self.nombre = Entry(frame)
        self.nombre.grid(row=1, column=1)
        self.nombre.focus()

        #Label Precio
        self.etiquetaPrecio = Label(frame, text="Precio: ")#se crea la etiqueta precio
        self.etiquetaPrecio.grid(row=2, column=0)#ubicacion de la etiqueta

        #Entry precio
        self.precio = Entry(frame)
        self.precio.grid(row=2, column=1)

        #Label Categoria
        self.etiqueta_categoria = Label(frame, text="Categoria: ")
        self.etiqueta_categoria.grid(row=3, column=0)

        #Entry Categoria
        self.categoria = Entry(frame)
        self.categoria.grid(row=3, column=1)

        #Label Cantidad
        self.etiqueta_cantidad = Label(frame, text="Cantidad: ")
        self.etiqueta_cantidad.grid(row=4, column=0)

        #Entry Cantidad
        self.cantidad = Entry(frame)
        self.cantidad.grid(row=4, column=1)

        # Boton añadir producto
        self.boton_anadir = ttk.Button(frame, text="Guardar Producto", command=self.add_producto)
        self.boton_anadir.grid(row=5, columnspan=4, sticky=W + E) #se fuerza con sticky la ocupacion que usara el grid mediante coordenada n,s,e,o

        # Boton Oscuro
        self.modoo Oscuro = ttk.Button(frame, text="Oscuro", command=self.modoo Oscuro)
        self.modoo Oscuro.grid(row=3, column=4)

        # Boton Claro
        self.modoo Claro = ttk.Button(frame, text="Claro", command=self.modoo Claro)
        self.modoo Claro.grid(row=2, column=4)
```

3. En la tabla que tiene su estilo, le modifique la cantidad de columna que tendrá ya que se creo varios label como la categoría/cantidad. También esta el método modo_oscuro/claro, como le mencione que no pude cambiarle el color de fondo en todo el programa hice varios intentos, modificaciones sin lograrlo.

```
22 class VentanaPrincipal(): 1 usage
23
24 def __init__(self, root):
25     self.modo_claro.grid(row=2, column=4)
26
27     # Mensaje informativo para el usuario
28     self.mensaje = Label(text='', fg='red', anchor=CENTER)
29     self.mensaje.grid(row=4, column=0, columnspan=5, pady=0, sticky=W + E)
30
31     # Tabla de Productos
32     # Estilo personalizado para la tabla
33     style = ttk.Style() #se define al objeto quien tendra el estilo
34     style.configure(style='mystyle.Treeview', highlightthickness=0, bd=0, background="#fcfcfc",
35                     font=('Calibri', 11, 'bold')) # se modifica la fuente de la tabla
36     style.configure(style='mystyle.Treeview Heading', background="#fcfcfc",
37                     font=('Calibri', 12, 'bold')) # Se modifica la fuente de las cabeceras
38     style.layout(style='mystyle.Treeview', layoutspec: [('mystyle.Treeview.treearea', {'sticky': 'nsw'})])
39
40     # Estructura de la tabla
41     self.tabla = ttk.Treeview(height=20, columns=("#0", "1", "2"), style='mystyle.Treeview')
42     self.tabla.grid(row=5, column=0, columnspan=2)
43     self.tabla.heading('#0', text='NOMBRE', anchor=CENTER) # Encabezado 0
44     self.tabla.heading('#1', text='PRECIO', anchor=CENTER) # Encabezado 1
45     self.tabla.heading('#2', text='CATEGORIA', anchor=CENTER) #Encabezado 2
46     self.tabla.heading('#3', text='STOCK', anchor=CENTER) #Encabezado 3
47
48     self.boton_eliminar = ttk.Button(text='ELIMINAR', command=self.del_producto)
49     self.boton_eliminar.grid(row=6, column=0, sticky=W + E)
50     self.boton_editar = ttk.Button(text='EDITAR', command=self.edit_producto)
51     self.boton_editar.grid(row=6, column=1, sticky=W + E)
52
53     self.get_productos()
54
55 def modo_oscuro(self): 1 usage
56     self.ventana.config(background="#4a4a4a")
57     self.tabla.config()
58
59 def modo_claro(self): 1 usage
60     self.ventana.config(background="#fcfcfc")
61
62 def db_consulta(self, consulta, parametros=()): 4 usages (1 dynamic)
63
64     with sqlite3.connect(self.db) as con:
65         cursor = con.cursor()
66         resultado = cursor.execute(consulta, parametros)
67         con.commit()
68     return resultado
```

4. En el método `get_productos` intente conectarme a sqlalchemy, pero me dio varios errores, modifique el código muchísimas veces sin lograrlo.

```
class VentanaPrincipal():
    def db_consulta(self, consulta, parametros=()):
        return resultado

    #session = self.Session()
    #try:
    #    resultado=session.query(consulta, parametros)
    #    session.commit()
    #    return resultado
    #finally:
    #    session.close()

    def get_productos(self):
        registros_tabla = self.tabla.get_children()
        for fila in registros_tabla:
            self.tabla.delete(fila)

        #query= db.session.query(Producto).order_by("nombre").all()
        query = "SELECT * FROM producto ORDER BY nombre DESC"
        registros = self.db_consulta(query)
        for fila in registros:
            print(fila)
            self.tabla.insert( parent="", index=0, text=fila[1], values=fila[2:])

    def validacion_nombre(self):
        return self.nombre.get().strip() != ""

    def validacion_precio(self):
        try:
            precio = float(self.precio.get())
            return precio > 0
        except ValueError:
            return False

    def validacion_categoria(self):
        return self.categoria.get().strip() != ""

    def validacion_cantidad(self):
        try:
            cantidad = int(self.cantidad.get())
            return cantidad > 0
        except ValueError:
            return False

    def add_producto(self):
        if not self.validacion_nombre():
```

5. En este punto hice varios cambios, me costo pero lo logre:
- En el query, agregue los signos de interrogación faltante.
 - Luego en los parámetros se agrego las variables nuevas, categoría, cantidad

```
22 class VentanaPrincipal(): 1 usage
181 def add_producto(self): 1 usage
183     if not self.validacion_nombre():
184         print("El nombre es obligatorio")
185         self.mensaje['text'] = 'El nombre es obligatorio y no puede estar vacio'
186         return
187     if not self.validacion_precio():
188         print("El Precio es obligatorio")
189         self.mensaje['text'] = 'El precio es obligatorio y debe ser un numero valido mayor que 0'
190         return
191     if not self.validacion_categoria():
192         print("La categoria es obligatorio")
193         self.mensaje['text'] = 'La Categoria es obligatorio y no puede estar vacio'
194         return
195     if not self.validacion_cantidad():
196         print("La cantidad es obligatorio")
197         self.mensaje['text'] = 'La cantidad es obligatorio y debe ser un numero valido mayor que 0'
198         return
199
200
201
202     query = 'INSERT INTO producto VALUES(NULL, ?, ?,?,?)'
203     parametros = (self.nombre.get(), self.precio.get(), self.categoria.get(), self.cantidad.get())
204     self.db_consulta(query, parametros)
205     print("Datos Guardados")
206     self.mensaje['text'] = 'producto {} añadido con éxito'.format(self.nombre.get())
207     self.nombre.delete(first=0, END) #Borrar el campo nombre del formulario
208     self.precio.delete(first=0, END) #Borrar el campo precio del formulario
209     self.categoria.delete(first=0, END)
210     self.cantidad.delete(first=0, END)
211     #print(self.nombre.get())
212     #print(self.precio.get())
213
214     self.get_productos()
215
216 def del_producto(self): 1 usage
217     # print(self.tabla.item(self.tabla.selection()))
218     #print(self.tabla.item(self.tabla.selection())['text'])
219     # print(self.tabla.item(self.tabla.selection())['values'])
220     #print(self.tabla.item(self.tabla.selection())['values'][0])
221
222     self.mensaje['text'] = '' # Mensaje inicialmente vacio
223     # Comprobacion de que se seleccione un procuto para poder eliminarlo
224     try:
225         self.tabla.item(self.tabla.selection())['text'][0]
226     except IndexError as e:
227         self.mensaje['text'] = 'Por favor, seleccione un producto'
228     return
```

- c. En el método edit_producto tiene las variables categoría, cantidad.
- d. En el método VentanaEditarProducto también se agregó las nuevas variables

```
app.py db.py
22 class VentanaPrincipal(): 1 usage
216 def del_producto(self): 1 usage
229
230 self.mensaje['text'] = ''
231 nombre = self.tabla.item(self.tabla.selection())['text']
232 query = 'DELETE FROM producto WHERE nombre = ?' # Consulta SQL
233 self.db_consulta(query, parametros=(nombre,)) #Ejecutar la consulta
234 self.mensaje['text'] = 'Producto {} eliminado con éxito'.format(nombre)
235 self.get_productos() #actualizar la tabla de productos
236
237 def edit_producto(self): 1 usage
238 try:
239     nombre = self.tabla.item(self.tabla.selection())['text']
240     precio = self.tabla.item(self.tabla.selection())['values'][0]
241     categoria = self.tabla.item(self.tabla.selection())['values'][1]
242     cantidad = self.tabla.item(self.tabla.selection())['values'][2]
243     VentanaEditarProducto(self, nombre, precio, categoria, cantidad, self.mensaje)
244 except IndexError:
245     self.mensaje['Text'] = 'Por favor, seleccione un producto'
246
247 class VentanaEditarProducto(): 1 usage
248
249 def __init__(self, ventana_principal, nombre, precio, categoria, cantidad, mensaje):
250     self.ventana_principal = ventana_principal
251     self.nombre = nombre
252     self.precio = precio
253     self.categoria = categoria
254     self.cantidad = cantidad
255     self.mensaje = mensaje
256
257
258 self.ventana_editar = Toplevel()
259 self.ventana_editar.title("Editar Producto")
260
261 # Creacion del contenedor Frame para la edicion del producto
262 frame_ep = LabelFrame(self.ventana_editar, text="Editar el siguiente producto", font=('Calibri',16, 'bold'))
263 frame_ep.grid(row=0, columnspan=10, pady=20, padx=20)
264
265 #Label y entry para el Nombre antiguo (solo lectura)
266 Label(frame_ep, text="Nombre antiguo: ", font=('Calibri', 13)).grid(row=1, column=0)
267 Entry(frame_ep, textvariable=StringVar(self.ventana_editar, value=nombre), state='readonly', font=('Calibri', 13)).grid(row=1, column=1)
268
269 #Label y Entry para el Nombre nuevo
270 Label(frame_ep, text="Nombre nuevo: ", font=('Calibri', 13)).grid(row=2, column=0)
271 self.input_nombre_nuevo = Entry(frame_ep, font=('Calibri', 13))
272 self.input_nombre_nuevo.grid(row=2, column=1)
273 self.input_nombre_nuevo.focus()
274
```

- e. la ventana editar también se agregaron los nuevos cambios de las variables nuevas me costo un monto manejar las filas/columnas de la ventana.
- f. En el query UPDATE también se agregaron las variables nuevas

```

247 class VentanaEditarProducto(): 1 usage
249 def __init__(self, ventana_principal, nombre, precio, categoria, cantidad, mensaje):
274
275     # Precio antiguo (solo lectura)
276     Label(frame_ep, text="Precio antiguo: ", font=('Calibri', 13)).grid(row=3, column=0)
277     Entry(frame_ep, textvariable=StringVar(self.ventana_editar, value=precio), state='readonly', font=('Calibri', 13)).grid(row=3, column=1)
278
279     # Precio nuevo
280     Label(frame_ep, text="Precio nuevo: ", font=('Calibri', 13)).grid(row=4, column=0)
281     self.input_precio_nuevo = Entry(frame_ep, font=('Calibri', 13))
282     self.input_precio_nuevo.grid(row=4, column=1)
283
284     #Categoria Antigua
285     Label(frame_ep, text="Categoria antigua: ", font=('Calibri', 13)).grid(row=5, column=0)
286     Entry(frame_ep, textvariable=StringVar(self.ventana_editar, value=categoria), state='readonly', font=('Calibri', 13)).grid(row=5, column=1)
287     #Categoria nueva
288     Label(frame_ep, text="Nueva categoria: ", font=('Calibri', 13)).grid(row=6, column=0)
289     self.input_categoria_nuevo = Entry(frame_ep, font=('Calibri', 13))
290     self.input_categoria_nuevo.grid(row=6, column=1)
291
292     #Cantidad antigua
293     Label(frame_ep, text="Cantidad antigua: ", font=('Calibri', 13)).grid(row=7, column=0)
294     Entry(frame_ep, textvariable=StringVar(self.ventana_editar, value=cantidad), state='readonly', font=('Calibri', 13)).grid(row=7, column=1)
295     #Cantidad Nueva
296     Label(frame_ep, text="Cantidad nueva: ", font=('Calibri', 13)).grid(row=8, column=0)
297     self.input_cantidad_nuevo = Entry(frame_ep, font=('Calibri', 13))
298     self.input_cantidad_nuevo.grid(row=8, column=1)
299
300
301
302     # Boton Actualizar
303     self.boton_actualizar = ttk.Button(frame_ep, text="Actualizar Producto", command=self.actualizar)
304     self.boton_actualizar.grid(row=9, columnspan=2, sticky=W+E)
305
306
307
308 def actualizar(self): 1 usage
309     nuevo_nombre = self.input_nombre_nuevo.get() or self.nombre
310     nuevo_precio = self.input_precio_nuevo.get() or self.precio
311     nuevo_categoria = self.input_categoria_nuevo.get() or self.categoria
312     nuevo_cantidad = self.input_cantidad_nuevo.get() or self.cantidad
313
314     if nuevo_nombre and nuevo_precio and nuevo_categoria and nuevo_cantidad:
315         query = 'UPDATE producto SET nombre = ?, precio = ?, categoria = ?, cantidad = ? WHERE nombre = ?'
316         parametros = (nuevo_nombre, nuevo_precio, nuevo_categoria, nuevo_cantidad, self.nombre)
317         self.ventana_principal.db_consulta(query, parametros)
318         self.mensaje['text'] = f'El producto {self.nombre} ha sido actualizado con exito'
319     else:

```

Bueno este es el fin de la app, se que los cambios no parecen mucho pero si me costo un montón mas en el intento de reemplazo de la orm, me gustaría que me indicara cual es la solución. Asi que bueno. Muchas gracias.