#https://www.youtube.com/watch?v=W2kAF9pKPPE&t=2733s

#Python Tkinter - Aplicación de Escritorio de Productos con Sqlite3, CRUD

from tkinter import ttk

from tkinter import \*

import sqlite3

class Estudiantes:

    db\_name = 'bdestudiantes.db'

    def \_\_init\_\_(self, window):      #Constructor

        self.wind = window          #Propiedad wind que almacena la ventana

        self.wind.title("Aplicación Estudiantes")

        #Crear un contenedor

        frame = LabelFrame(self.wind, text = 'Registro Estudiantes')

        frame.grid(row = 0, column = 0, columnspan = 3, pady = 20)

        #Ingresar identificación estudiante

        Label(frame, text = "Identificación: ").grid(row = 1, column = 0)

        self.identificacion = Entry(frame)

        self.identificacion.focus()     #Posiciona el cursor en esta caja al iniciar la aplicación

        self.identificacion.grid(row = 1, column = 1)

        #Ingresar Nombre

        Label(frame, text = "Nombre").grid(row = 2, column = 0)

        self.nombre = Entry(frame)

        self.nombre.grid(row = 2, column = 1)

        #Botón Agrear estudiante

        #sticky: Va desde W(Este) hasta E(Oeste) - ocupa todo el ancho

        ttk.Button(frame, text = "Guardar Estudiante", command=self.add\_estudiante).grid(row = 3, columnspan = 2, sticky = W + E)

        #Mensaje de salida

        self.message = Label(text = "", fg = "red")

        self.message.grid(row = 3, column = 0, columnspan= 2, sticky=W + E)

        #Tabla

        self.tree = ttk.Treeview(height=10, columns=2)      #height (filas en la tabla) columns(campos en la tabla)

        self.tree.grid(row = 4, column = 0, columnspan=2)

        self.tree.heading('#0', text="Identificación", anchor= CENTER)

        self.tree.heading('#1', text="Nombre", anchor= CENTER)

        #Botones

        ttk.Button(text="ELIMINAR", command = self.borrar\_estudiante).grid(row=5, column=0, sticky= W + E)

        ttk.Button(text="MODIFICAR", command=self.modificar\_estudiante).grid(row=5, column=1, sticky= W + E)

        #Mostrar estudiantes en la tabla

        self.obtener\_estudiantes()

    def ejecute\_consulta(self, consulta, parametros = ()):   #función para hacer consulta a la BD - parametros es una tupla vacia

        with sqlite3.connect(self.db\_name) as conn:          #Método para conexión a la BD, conn guarda la conexión

            posicion = conn.cursor()                         #Me permite saber en qué registro está ubicada

            resultado = posicion.execute(consulta, parametros)

            conn.commit()

        return resultado

    def obtener\_estudiantes(self):  #Función para leer registros

        #Limpiando la tabla estudiantes

        registro = self.tree.get\_children()     #self.tree (tabla) get\_children (obtiene todos los elementos de la tabla)

        for elemento in registro:

            self.tree.delete(elemento)

        #Consultando datos en tabla estudiantes

        consulta = "SELECT \* FROM estudiantes ORDER BY identificacion DESC"

        db\_filas = self.ejecute\_consulta(consulta)

        #Rellenando los datos

        for fila in db\_filas:

            #print(fila)

            self.tree.insert('', 0, text= fila[0], value = fila[1])

            #print(fila)

    def validacion\_estudiante(self):

        return len(self.identificacion.get()) != 0 and len(self.nombre.get()) != 0     #.get captura lo que el usr ingresa

    def add\_estudiante(self):

        if self.validacion\_estudiante():

            consulta = 'INSERT INTO estudiantes VALUES(?, ?)'

            parametros =  (self.identificacion.get(), self.nombre.get())

            self.ejecute\_consulta(consulta, parametros)

            print("Datos guardados")

            self.message["text"] = "El Estudiante {} ha sido agregado exitosamente".format(self.nombre.get())

            self.identificacion.delete(0, END)

            self.nombre.delete(0, END)

            #print(self.identificacion.get())

            #print(self.nombre.get())

        else:

            print('Identificación y/o nombre requerido')

            self.message["text"] = "La identificación y el nombre son requeridos"

        self.obtener\_estudiantes()

    def borrar\_estudiante(self):

        #Limpiar mensaje en pantalla

        self.message["text"] = ""

        try:

            #Capturar el texto=identificación del registro seleccionado

            #ejemplo registro seleccionado -->

            #{'text': '1111', 'image': '', 'values': ['zzzzzz'], 'open': 0, 'tags': ''}

            self.tree.item(self.tree.selection())["text"][0]

        except IndexError as error:

            self.message["text"] = "Selecciona primero el estudiante"

            return

        self.message["text"] = ""

        identificacion = self.tree.item(self.tree.selection())["text"]

        consulta = "DELETE FROM estudiantes WHERE identificacion = ?"

        self.ejecute\_consulta(consulta, (identificacion, ))

        self.message["text"] = "El estudiante {} ha sido eliminado exitosamente".format(identificacion)

        #Actualizar tabla

        self.obtener\_estudiantes()

    def modificar\_estudiante(self):

        self.message["text"] = ""

        try:

            #Capturar el texto=identificación del registro seleccionado

            #ejemplo registro seleccionado -->

            #{'text': '1111', 'image': '', 'values': ['zzzzzz'], 'open': 0, 'tags': ''}

            self.tree.item(self.tree.selection())["text"][0]

        except IndexError as error:

            self.message["text"] = "Selecciona primero el estudiante"

            return

        old\_identificacion = self.tree.item(self.tree.selection())["text"]

        old\_nombre = self.tree.item(self.tree.selection())["values"][0]

        #Crear una ventana encima de la anterior para modificar registro

        self.mod\_wind = Toplevel()

        self.mod\_wind.title = "Modificar Estudiante"

        #old\_identificacion

        Label(self.mod\_wind, text = "Identificación: ").grid(row= 0, column=1)

        Entry(self.mod\_wind, textvariable=StringVar(self.mod\_wind, value =

        old\_identificacion), state="readonly").grid(row = 0, column=2)

        #Nueva identificacion

        Label(self.mod\_wind, text="New ident.: ").grid(row = 1, column = 1)

        self.New\_identificacion = Entry(self.mod\_wind)

        self.New\_identificacion.grid(row = 1, column=2)

        #old\_nombre

        Label(self.mod\_wind, text="Nombre: ").grid(row=2, column=1)

        Entry(self.mod\_wind, textvariable=StringVar(self.mod\_wind, value=

        old\_nombre), state = "readonly").grid(row=2, column=2)

        #Nuevo\_nombre

        Label(self.mod\_wind, text="Nuevo Nombre: ").grid(row=3, column=1)

        self.Nuevo\_nombre = Entry(self.mod\_wind)

        self.Nuevo\_nombre.grid(row=3, column=2)

        Button(self.mod\_wind, text= "ACTUALIZAR", command=lambda:self.mod\_registro(self.New\_identificacion.get(),

        old\_identificacion, self.Nuevo\_nombre.get(), old\_nombre)).grid(row=4, column=2, sticky=W)

    def mod\_registro(self, New\_identificacion, old\_identificacion, Nuevo\_nombre, old\_nombre):

        consulta = "UPDATE estudiantes SET identificacion = ?, nombre = ? WHERE identificacion = ? AND nombre = ?"

        parametros = (New\_identificacion, Nuevo\_nombre, old\_identificacion, old\_nombre)

        self.ejecute\_consulta(consulta, parametros)

        self.mod\_wind.destroy()          #Cuando el usr de clic sobre ACTUALIZAR se cierra la ventana de modificación

        self.message["text"]= "El Estudiante {} ha sido actulizado exitosamente".format(New\_identificacion)

        self.obtener\_estudiantes()

if \_\_name\_\_ == "\_\_main\_\_":   #Comprueba para iniciar la aplicación

    window = Tk()

    aplicacion = Estudiantes(window)

    window.mainloop()