Proyecto Final de Programación Orientada a Objetos

Por: Ing. Fernando Ospina Marín

A continuación, se detallan los requerimientos del proyecto final de la materia:

Requerimientos generales.

1. Modificar, agregar, corregir y hacer funcionar un aplicativo gráfico en Python 3 llamado **plantilla.py**, que usa la librería Tkinter y con la apariencia que se muestra en el **mockup** proporcionado (Anexo 1).
2. El aplicativo debe ser de tipo CRUD (Create, Read, Update, Delete) lo que significa que debe permitir:
   - Crear o insertar información.
   - Leer o consultar la información.
   - Actualizar o modificar la información.
   - Eliminar o borrar la información.
3. Es obligatorio el uso de la base de datos SQLite3 para la persistencia de los datos.
4. Se debe utilizar la base de datos llamada **Participantes.db** (Anexo 2).
5. Se debe crear una tabla en la base de datos una llamada **t_ciudades**. La estructura de la tabla se anexa en este documento y es obligatorio definirla de la misma forma dentro del sistema que se desarrolle (Anexo 3).
6. Se debe usar la tabla llamada **t_participantes** (Anexo 3), a la cual se le debe agregar el atributo "Ciudad" para almacenar en este campo, el nombre de la ciudad de participante al evento.
7. Es obligatorio usar el programa llamado **plantilla.py** (Anexo 4) que se proporciona para alcanzar los objetivos planteados, el cual debe ser modificado para satisfacer los requerimientos solicitados para este proyecto.
8. El trabajo se debe realizar en grupo de acuerdo con los grupos ya definidos.
9. Se escogerá de manera aleatoria el grupo que deba sustentar.
10. Solo un miembro del grupo realiza la sustentación, aunque en ese momento es obligatoria la presencia de todos los integrantes del grupo.
11. En cada clase se destinará tiempo para la elaboración del trabajo y resolverán posibles dudas o inquietudes.
12. La entrega del proyecto se realiza a más tardar el día 23 de febrero a las 11:59 p.m. ó 23:59.
13. Las sustentaciones se realizarán los días 24 y 26 de febrero en el horario de clase y seleccionará de forma aleatoria el grupo para sustentar.
14. Por lo anterior la presencia en ambas sesiones es obligatoria.

Requerimientos específicos.

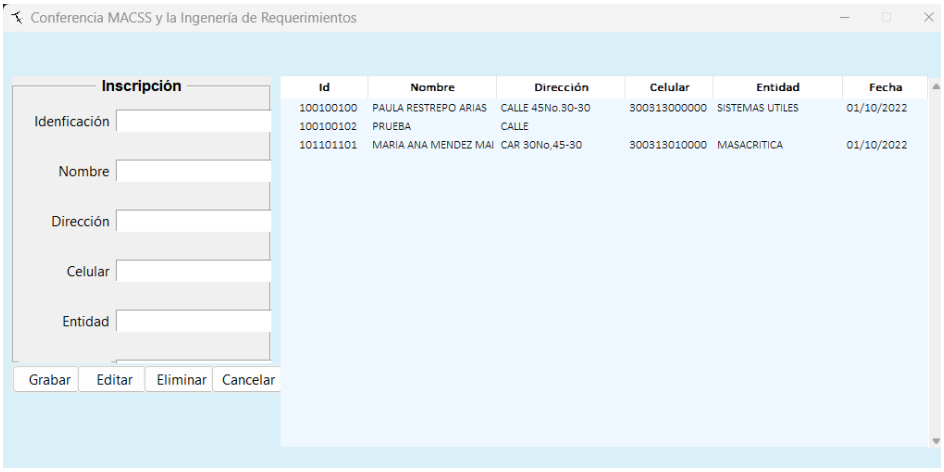
1. La captura del campo que identifica al participante debe ser validados no mayores a 15 caracteres. Si el campo excede la longitud establecida se deberá informar al usuario y eliminar los caracteres sobrantes, sin borrar todo el campo para evitar tener que repetir acciones de formo innecesaria.

2. Se debe incluir un campo de captura en la pantalla para la fecha de la inscripción y la fecha que se solicita deber ser validada como fecha cierta, es decir, no pueden existir fechas como 29 de febrero sin validar que el año sea bisiesto. De igual forma se debe validar la cantidad de días que le corresponde a cada mes.
3. Se debe incluir un campo en la pantalla para capturar la ciudad sin que se tenga que digitar, lo que significa que es obligatorio implementar un mecanismo que despliegue las ciudades por departamento para seleccionar la ciudad usando el mouse. Las ciudades se encuentran en la tabla llamada **t_ciudades**.
4. Los botones que se definen dentro del Mockup deben ser funcionales:
   Botón **Consultar**: debe servir para llamar a través del Id o Nit del participante, todos los datos del mismo y cargar la grilla de datos con los de cada participante.
   Botón **Grabar**: debe servir para grabar la información suministrada, en las tablas de la base de datos. **Nota importante**: cuando un participante ya se encuentra en la base de datos el botón grabar no puede modificar del Id o Nit del mismo por ser llave primaria, por lo que este campo debe estar deshabilitado para este caso en específico. Mediante el uso de una ventana emergente se de confirmar el éxito o fracaso de la acción.
   Botón **Editar**: debe servir para modificar los datos almacenados en la base de datos y debe permitir seleccionar una tupla de la grilla de datos (*treeParticipantes* o componente TreeView) para que se puedan modificar en los campos de captura definidos en el mockup de la aplicación. Mediante el uso de una ventana emergente se debe confirmar el éxito o fracaso de la acción.
   Botón **Eliminar**: debe servir para eliminar participantes de forma individual o debe permitir eliminar un conjunto de participantes que pueden ser **todos**, confirmando su eliminación a través de una ventana emergente que muestre el éxito o fracaso de la acción.
   Botón **Cancelar**: debe servir para cancelar cualquier acción en curso y debe limpiar todos los campos de captura mostrándolos vacíos en la pantalla.
5. Todos los botones deben cambiar de color cuando se pase el mouse por encima de ellos. Lo anterior significa que no se acepta el color por defecto que traen los botones de la librería tkinter o ttk y por lo tanto deberá implementarse el color de manera explicita en el código del programa.

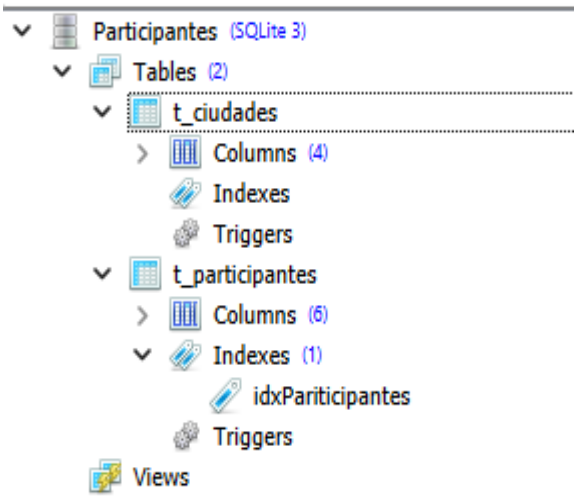
Condiciones importantes.

1. El programa que se entregue debe correr sin fallas o errores para que se pueda sustentar.
2. Si por alguna razón el programa falla y se hace evidente en la terminal se considerará como programa fallido y no se continuará con la sustentación.
3. Solo se permite el uso de la librería gráfica **TkInter** y su subconjunto **Ttk**. Cualquier uso de otras librerías se considerará como programa fallido por no cumplir los requerimientos.
4. Para las demás acciones como la validación de la fecha, es permitido usar cualquier librería que se considere necesaria.
5. El proyecto se debe entregar en formato ZIP y debe contener todo lo necesario para que funcione correctamente.

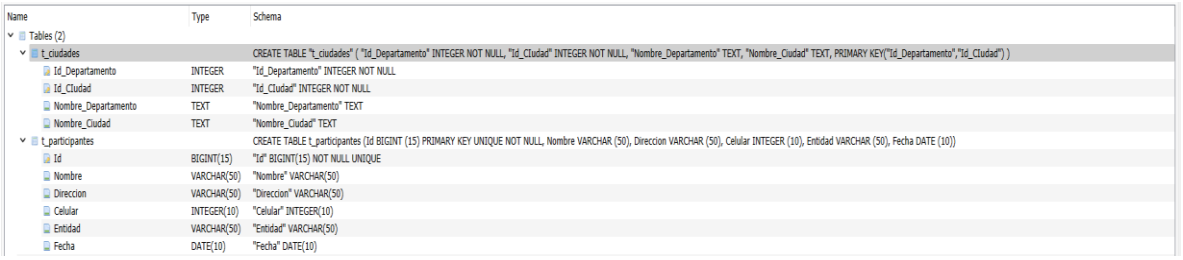
Anexo 1. Mockup de la aplicación



Anexo 2. Diseño de la Base de Datos



Anexo 3. Estructura de las tablas de datos



Name	Type	Schema
Tables (2)		
t_ciudades		CREATE TABLE "t_ciudades" ("Id_Departamento" INTEGER NOT NULL, "Id_CIudad" INTEGER NOT NULL, "Nombre_Departamento" TEXT, "Nombre_Ciudad" TEXT, PRIMARY KEY("Id_Departamento","Id_CIudad"))
Id_Departamento	INTEGER	"Id_Departamento" INTEGER NOT NULL
Id_CIudad	INTEGER	"Id_CIudad" INTEGER NOT NULL
Nombre_Departamento	TEXT	"Nombre_Departamento" TEXT
Nombre_Ciudad	TEXT	"Nombre_Ciudad" TEXT
t_participantes		CREATE TABLE t_participantes (Id BIGINT (15) PRIMARY KEY UNIQUE NOT NULL, Nombre VARCHAR (50), Direccion VARCHAR (50), Celular INTEGER (10), Entidad VARCHAR (50), Fecha DATE (10))
Id	BIGINT(15)	"Id" BIGINT(15) NOT NULL UNIQUE
Nombre	VARCHAR(50)	"Nombre" VARCHAR(50)
Direccion	VARCHAR(50)	"Direccion" VARCHAR(50)
Celular	INTEGER(10)	"Celular" INTEGER(10)
Entidad	VARCHAR(50)	"Entidad" VARCHAR(50)
Fecha	DATE(10)	"Fecha" DATE(10)

Anexo 4. Código en Python.

Se proporciona el código a mejorar, pero se debe tener en cuenta la compatibilidad con Word por si se copia directamente desde este archivo. El código en Python o plantilla.py se publica en el Drive del curso y se recomienda descargarlo desde el sitio.

```
# !/usr/bin/python3
# -*- coding: utf-8 -*-
import tkinter as tk
import tkinter.ttk as ttk
from tkinter import messagebox as mssg
import sqlite3

class Participantes:
    # nombre de la base de datos  y ruta
    path = r'X:/Users/ferna/Documents/UNal/Alumnos/2024_S2/POO/Proy'
    db_name = path + r'/Participantes.db'
    actualiza = None
    def __init__(self, master=None):
        # Top Level - Ventana Principal
        self.win = tk.Tk() if master is None else tk.Toplevel()


        #Top Level - Configuración
        self.win.configure(background="#d9f0f9", height="480", relief="flat", width="1024")
        self.win.geometry("1024x480")
        self.path = self.path +r'/f2.ico'
        self.win.iconbitmap(self.path)
        self.win.resizable(False, False)
        self.win.title("Conferencia MACSS y la Ingenería de Requerimientos")
        self.win.pack_propagate(0)

        # Main widget
        self.mainwindow = self.win

        #Label Frame
        self.lblfrm_Datos = tk.LabelFrame(self.win, width= 600, height= 200, labelanchor= "n",
                                font= ("Helvetica", 13,"bold"))
        #Label Id
        self.lblId = ttk.Label(self.lblfrm_Datos)
        self.lblId.configure(anchor="e", font="TkTextFont", justify="left", text="Idenficación")
        self.lblId.configure(width="12")
        self.lblId.grid(column="0", padx="5", pady="15", row="0", sticky="w")

        #Entry Id
        self.entryId = tk.Entry(self.lblfrm_Datos)
        self.entryId.configure(exportselection="false", justify="left",relief="groove", takefocus=True, width="30")
        self.entryId.grid(column="1", row="0", sticky="w")
        self.entryId.bind("<Key>", self.valida_Identificacion)


        #Label Nombre
        self.lblNombre = ttk.Label(self.lblfrm_Datos)
        self.lblNombre.configure(anchor="e", font="TkTextFont", justify="left", text="Nombre")
        self.lblNombre.configure(width="12")
        self.lblNombre.grid(column="0", padx="5", pady="15", row="1", sticky="w")

        #Entry Nombre
        self.entryNombre = tk.Entry(self.lblfrm_Datos)
        self.entryNombre.configure(exportselection="true", justify="left",relief="groove", width="30")
        self.entryNombre.grid(column="1", row="1", sticky="w")

        #Label Direccion
        self.lblDireccion = ttk.Label(self.lblfrm_Datos)
        self.lblDireccion.configure(anchor="e", font="TkTextFont", justify="left", text="Dirección")
```

```
self.lblDireccion.configure(width="12")
self.lblDireccion.grid(column="0", padx="5", pady="15", row="2", sticky="w")

#Entry Direccion
self.entryDireccion = tk.Entry(self.lblfrm_Datos)
self.entryDireccion.configure(exportselection="true", justify="left",relief="groove", width="30")
self.entryDireccion.grid(column="1", row="2", sticky="w")

#Label Celular
self.lblCelular = ttk.Label(self.lblfrm_Datos)
self.lblCelular.configure(anchor="e", font="TkTextFont", justify="left", text="Celular")
self.lblCelular.configure(width="12")
self.lblCelular.grid(column="0", padx="5", pady="15", row="3", sticky="w")

#Entry Celular
self.entryCelular = tk.Entry(self.lblfrm_Datos)
self.entryCelular.configure(exportselection="false", justify="left",relief="groove", width="30")
self.entryCelular.grid(column="1", row="3", sticky="w")

#Label Entidad
self.lblEntidad = ttk.Label(self.lblfrm_Datos)
self.lblEntidad.configure(anchor="e", font="TkTextFont", justify="left", text="Entidad")
self.lblEntidad.configure(width="12")
self.lblEntidad.grid(column="0", padx="5", pady="15", row="4", sticky="w")

#Entry Entidad
self.entryEntidad = tk.Entry(self.lblfrm_Datos)
self.entryEntidad.configure(exportselection="true", justify="left",relief="groove", width="30")
self.entryEntidad.grid(column="1", row="4", sticky="w")

#Label Fecha
self.lblFecha = ttk.Label(self.lblfrm_Datos)
self.lblFecha.configure(anchor="e", font="TkTextFont", justify="left", text="Fecha")
self.lblFecha.configure(width="12")
self.lblFecha.grid(column="0", padx="5", pady="15", row="5", sticky="w")

#Entry Fecha
self.entryFecha = tk.Entry(self.lblfrm_Datos)
self.entryFecha.configure(exportselection="true", justify="left",relief="groove", width="30")
self.entryFecha.grid(column="1", row="5", sticky="w")
self.entryFecha.bind("<Key>", self.valida_Fecha)


#Configuración del Labe Frame
self.lblfrm_Datos.configure(height="310", relief="groove", text=" Inscripción ", width="330")
self.lblfrm_Datos.place(anchor="nw", relx="0.01", rely="0.1", width="280", x="0", y="0")
self.lblfrm_Datos.grid_propagate(0)

#Botón Grabar
self.btnGrabar = ttk.Button(self.win)
self.btnGrabar.configure(state="normal", text="Grabar", width="9")
self.btnGrabar.place(anchor="nw", relx="0.01", rely="0.75", x="0", y="0")
self.btnGrabar.bind("<1>", self.adiciona_Registro, add="+")

#Botón Editar
self.btnEditar = ttk.Button(self.win)
self.btnEditar.configure(text="Editar", width="9")
self.btnEditar.place(anchor="nw", rely="0.75", x="80", y="0")
self.btnEditar.bind("<1>", self.edita_tablaTreeView, add="+")

#Botón Eliminar
self.btnEliminar = ttk.Button(self.win)
self.btnEliminar.configure(text="Eliminar", width="9")
self.btnEliminar.place(anchor="nw", rely="0.75", x="152", y="0")
self.btnEliminar.bind("<1>", self.elimina_Registro, add="+")

#Botón Cancelar
self.btnCancelar = ttk.Button(self.win)
```

```
    self.btnCancelar.configure(text="Cancelar", width="9",command = self.limpia_Campos)
    self.btnCancelar.place(anchor="nw", rely="0.75", x="225", y="0")

    #tablaTreeView
    self.style=ttk.Style()
    self.style.configure("estilo.Treeview", highlightthickness=0, bd=0, background='AliceBlue', font=('Calibri Light',10))
    self.style.configure("estilo.Treeview.Heading", background='Azure', font=('Calibri Light', 10,'bold'))
    self.style.layout("estilo.Treeview", [('estilo.Treeview.treearea', {'sticky': 'nswe'})])

    self.treeDatos = ttk.Treeview(self.win, height = 10, style="estilo.Treeview")
    self.treeDatos.place(x=380, y=10, height=340, width = 500)

   # Etiquetas de las columnas
    self.treeDatos["columns"]=("Nombre","Dirección","Celular","Entidad","Fecha")
    # Determina el espacio a mostrar que ocupa el código
    self.treeDatos.column('#0',        anchor="w", stretch="true", width=15)
    self.treeDatos.column('Nombre',    stretch="true",           width=60)
    self.treeDatos.column('Dirección', stretch="true",           width=60)
    self.treeDatos.column('Celular',   stretch="true",           width=16)
    self.treeDatos.column('Entidad',   stretch="true",           width=60)
    self.treeDatos.column('Fecha',     stretch="true",           width=12)

   #Encabezados de las columnas de la pantalla
    self.treeDatos.heading('#0',       text = 'Id')
    self.treeDatos.heading('Nombre',   text = 'Nombre')
    self.treeDatos.heading('Dirección',text = 'Dirección')
    self.treeDatos.heading('Celular',  text = 'Celular')
    self.treeDatos.heading('Entidad',  text = 'Entidad')
    self.treeDatos.heading('Fecha',    text = 'Fecha')

    #Scrollbar en el eje Y de treeDatos
    self.scrollbar=ttk.Scrollbar(self.win, orient='vertical', command=self.treeDatos.yview)
    self.treeDatos.configure(yscroll=self.scrollbar.set)
    self.scrollbar.place(x=1000, y=50, height=400)

    #Carga los datos en treeDatos
    self.lee_tablaTreeView()
    self.treeDatos.place(anchor="nw", height="400", rely="0.1", width="700", x="300", y="0")


def valida(self):
    '''Valida que el Id no esté vacio, devuelve True si ok'''
    return (len(self.entryId.get()) != 0 )

def run(self):
    self.mainwindow.mainloop()

def valida_Identificacion(self, event=None):
    ''' Valida que la longitud no sea mayor a 15 caracteres'''
    if event.char:
        if len(self.entryId.get()) >= 15:
            mssg.showerror('Atención!!','.. ¡Máximo 15 caracteres! ..')
            self.entryId.delete(15,"end")
    else:
        self.entryId.delete(15,"end")

def valida_Fecha(self, event=None):
  pass


def carga_Datos(self):
    ''' Carga los datos en los campos desde el treeView'''
    self.entryId.insert(0,self.treeDatos.item(self.treeDatos.selection())['text'])
    self.entryId.configure(state = 'readonly')
    self.entryNombre.insert(0,self.treeDatos.item(self.treeDatos.selection())['values'][0])
    self.entryDireccion.insert(0,self.treeDatos.item(self.treeDatos.selection())['values'][1])
    self.entryCelular.insert(0,self.treeDatos.item(self.treeDatos.selection())['values'][2])
    self.entryEntidad.insert(0,self.treeDatos.item(self.treeDatos.selection())['values'][3])
```

```
        self.entryFecha.insert(0,self.treeDatos.item(self.treeDatos.selection())['values'][4])

    def limpia_Campos(self):
      pass

    def run_Query(self, query, parametros = ()):
        ''' Función para ejecutar los Querys a la base de datos '''
        with sqlite3.connect(self.db_name) as conn:
            cursor = conn.cursor()
            result = cursor.execute(query, parametros)
            conn.commit()
        return result

    def lee_tablaTreeView(self):
        ''' Carga los datos de la BD y Limpia la Tabla tablaTreeView '''
        tabla_TreeView = self.treeDatos.get_children()
        for linea in tabla_TreeView:
            self.treeDatos.delete(linea)
        # Seleccionando los datos de la BD
        query = 'SELECT * FROM t_participantes ORDER BY Id DESC'
        db_rows = self.run_Query(query)
        # Insertando los datos de la BD en la tabla de la pantalla
        for row in db_rows:
            self.treeDatos.insert('',0, text = row[0], values = [row[1],row[2],row[3],row[4],row[5]])

    def adiciona_Registro(self, event=None):
        '''Adiciona un producto a la BD si la validación es True'''
        if self.actualiza:
            self.actualiza = None
            self.entryId.configure(state = 'readonly')
            query = 'UPDATE t_participantes SET Id = ?,Nombre = ?,Dirección = ?,Celular = ?, Entidad = ?, Fecha = ? WHERE Id = ?'
            parametros = (self.entryId.get(), self.entryNombre.get(), self.entryDireccion.get(),
                          self.entryCelular.get(), self.entryEntidad.get(), self.entryFecha.get()
                          )
                         #   self.entryId.get())
            self.run_Query(query, parametros)
            mssg.showinfo('Ok',' Registro actualizado con éxito')
        else:
            query = 'INSERT INTO t_participantes VALUES(?, ?, ?, ?, ?, ?)'
            parametros = (self.entryId.get(),self.entryNombre.get(), self.entryDireccion.get(),
                          self.entryCelular.get(), self.entryEntidad.get(), self.entryFecha.get())
            if self.valida():
                self.run_Query(query, parametros)
                self.limpia_Campos()
                mssg.showinfo('',f'Registro: {self.entryId.get()} .. agregado')
            else:
                mssg.showerror("¡ Atención !","No puede dejar la identificación vacía")
        self.limpia_Campos()
        self.lee_tablaTreeView()

    def edita_tablaTreeView(self, event=None):
        try:
            # Carga los campos desde la tabla TreeView
            self.treeDatos.item(self.treeDatos.selection())['text']
            self.limpia_Campos()
            self.actualiza = True # Esta variable controla la actualización
            self.carga_Datos()
        except IndexError as error:
            self.actualiza = None
            mssg.showerror("¡ Atención !",'Por favor seleccione un ítem de la tabla')
            return

    def elimina_Registro(self, event=None):
     pass

if __name__ == "__main__":
    app = Participantes()
    app.run()
```