

Proyecto Final de Programación Orientada a Objetos

Por: Ing. Fernando Ospina Marín

A continuación, se detallan los requerimientos del proyecto:

Requerimientos generales.

1. Modificar y hacer funcionar un aplicativo gráfico en Python 3 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 crear un base de datos llamada **Inventario.db** (Anexo 2).
5. Se deben crear dos tablas en la base de datos una llamada **Proveedores** y la otra llamada **Productos**. La estructura de las tablas se anexa en este documento y es obligatorio definirlas de la misma forma dentro del sistema que se desarrolle (Anexo 3).
6. Es obligatorio usar el programa llamado **plantilla.py** (Anexo 4) que se proporciona para alcanzar los objetivos planteados.
7. El trabajo se debe realizar en grupo de acuerdo con los grupos ya definidos.
8. Solo un miembro del grupo realiza la sustentación, aunque en ese momento es obligatoria la presencia de todos los integrantes del grupo.
9. En cada clase se destinará tiempo para la elaboración del trabajo y resolverán posibles dudas o inquietudes.
10. La entrega del proyecto se realiza a más tardar el día 27 de noviembre a las 11:59:59 p.m.
11. Las sustentaciones se realizarán los días 28 y 30 de noviembre en el horario de clase y seleccionará de forma aleatoria el grupo para sustentar.
12. Por lo anterior la presencia en ambas sesiones es obligatoria.

Requerimientos específicos.

1. La captura del campo que identifica al proveedor y al producto deben ser validados no mayores a 15 caracteres.
2. 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. Los botones que se definen dentro del Mockup deben ser funcionales:
Botón Consultar: debe servir para llamar a través del Id o Nit del proveedor todos los datos del mismo y cargar la grilla de datos con los productos que ese proveedor suministra.

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 (*treeProductos* 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 de confirmar 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.

Ingreso de datos

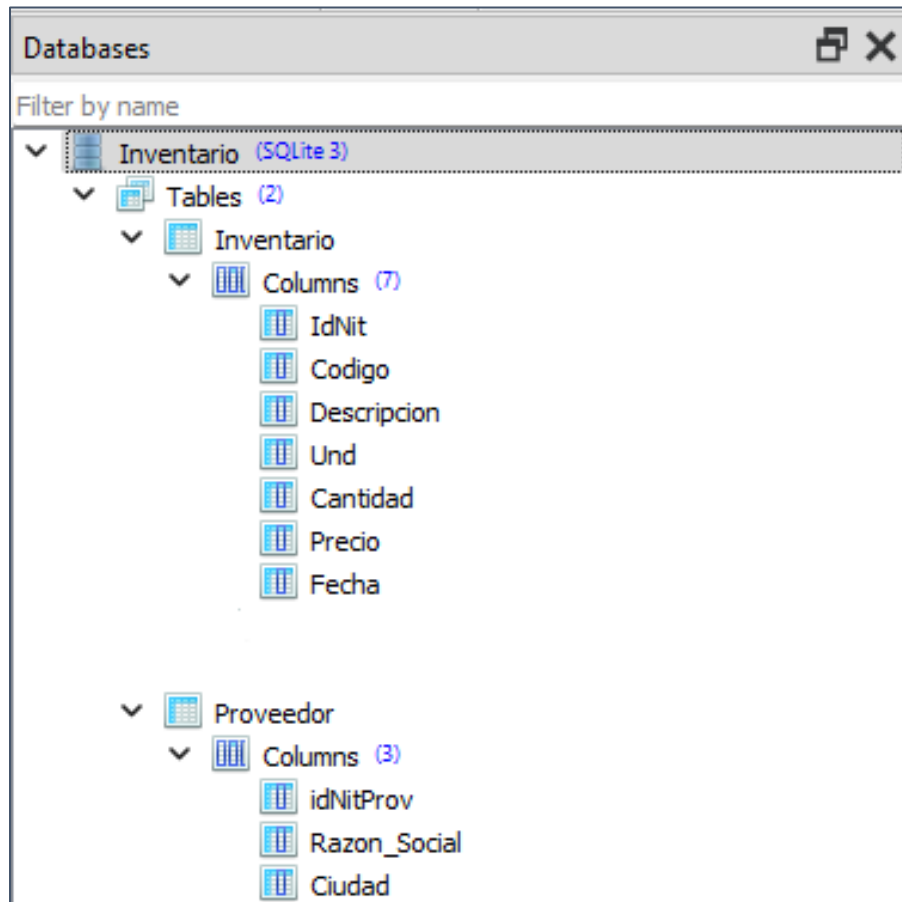
ID/Nit Razon social Ciudad

Código Descripción Unidad




Cantidad Precio \$ Fecha




ID / Nit	Código	Descripción	Unidad	Cantidad	Precio	Fecha
800-1	PT012	Puntilla de 1/5" de acero	M3	22.0	100.0	09/09/2023
800-1	PT011	Puntilla de 1/4" de acero	M3	15.0	100.0	09/09/2023
800-1	PT010	Puntilla de 2/3" de acero	M3	20.0	100.0	09/09/2023
800-1	PT009	Puntilla de 1/3" de acero	M3	19.0	100.0	09/09/2023
800-1	PT008	Puntilla de 1/2" de acero	M3	18.0	100.0	09/09/2023
800-1	PT007	Puntill de 3/4" de acero	M3	17.0	100.0	09/09/2023
800-1	PT006	Puntill de 3" de acero	M3	16.0	100.0	09/09/2023
800-1	PT005	Puntill de 7" de acero	M3	15.0	100.0	09/09/2023
800-1	PT004	Puntill de 6" de acero	M3	14.0	100.0	09/09/2023
800-1	PT003	Puntilla de 5" de acero	M3	13.0	100.0	09/09/2023
800-1	PT002	Puntilla de 4" de acero	M3	11.0	100.0	09/09/2023
800-1	PT001	Puntilla de 3" de acero bindado	M3	10.0	100.0	09/09/2023
800-1	CE011	Cemento	M3	20.0	100.0	09/09/2023
800-1	CE010	Cemento	M3	19.0	100.0	09/09/2023
800-1	CE009	Cemento	M3	18.0	100.0	09/09/2023
800-1	CE008	Cemento	M3	17.0	100.0	09/09/2023
800-1	CE007	Cemento	M3	16.0	100.0	09/09/2023
800-1	CE006	Cemento	M3	15.0	100.0	09/09/2023
800-1	CE004	Cemento	M3	13.0	100.0	09/09/2023
800-1	CE003	Cemento	M3	12.0	100.0	09/09/2023
800-1	CE002	Cemento	M3	11.0	100.0	09/09/2023
800-1	CE001	Cemento	M3	10.0	100.0	09/09/2023
800-1	ARO09	Arena L	M3	18.0	100.0	09/09/2023
800-1	ARO08	Arena L	M3	17.0	100.0	09/09/2023

Anexo 2. Diseño de la Base de Datos



Anexo 3. Estructura de las tablas de datos

Inventario		Table name: Inventario	<input type="checkbox"/> WITHOUT ROWID <input type="checkbox"/> STRICT							
	Name	Data type	Clave Primaria	Foreign Key	Único	Check	No es NULL	Collate	Generated	
1	IdNit	VARCHAR (15)								NULL
2	Codigo	VARCHAR								NULL
3	Descripcion	VARCHAR ...								NULL
4	Und	VARCHAR (10)								NULL
5	Cantidad	DOUBLE								NULL
6	Precio	DOUBLE								NULL
7	Fecha	DATE								NULL

Inventario		Table name: Proveedor	<input type="checkbox"/> WITHOUT ROWID <input type="checkbox"/> STRICT							
	Name	Data type	Clave Primaria	Foreign Key	Único	Check	No es NULL	Collate	Generated	
1	idNitProv	VARCHAR								NULL
2	Razon_Social	VARCHAR ...								NULL
3	Ciudad	VARCHAR ...								NULL

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 aquí. 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 Inventario:
    def __init__(self, master=None):
        self.path = r'X:/Users/ferna/Documents/UNal/Alumnos/2023_S2/ProyInventario'
        self.db_name = self.path + r'/Inventario.db'
        ancho=830;alto=840 # Dimensione de la pantalla
        actualiza = None

        # Crea ventana principal
        self.win = tk.Tk()
        self.win.geometry(f'{ancho}x{alto}')
        self.win.iconbitmap("f2.ico")
        self.win.resizable(False, False)
        self.win.title("Manejo de Proveedores")

        #Centra la pantalla
        self.centra(self.win,ancho,alto)

        # Contenedor de widgets
        self.win = tk.LabelFrame(master)
        self.win.configure(background="#e0e0e0",font="{Arial} 12 {bold}",
                           height=ancho,labelanchor="n",width=alto)
        self.tabs = ttk.Notebook(self.win)
        self.tabs.configure(height=800, width=799)

        #Frame de datos
        self.frm1 = ttk.Frame(self.tabs)
        self.frm1.configure(height=200, width=200)

        #Etiqueta IdNit del Proveedor
        self.lblIdNit = ttk.Label(self.frm1)
        self.lblIdNit.configure(text='Id/Nit', width=6)
        self.lblIdNit.place(anchor="nw", x=10, y=40)

        #Captura IdNit del Proveedor
        self.idNit = ttk.Entry(self.frm1)
        self.idNit.configure(takefocus=True)
        self.idNit.place(anchor="nw", x=50, y=40)
        self.idNit.bind("<Key>", self.validaIdNit)
        self.idNit.bind("<BackSpace>", lambda _:self.idNit.delete(len(self.idNit.get())),'end')

        #Etiqueta razón social del Proveedor
        self.lblRazonSocial = ttk.Label(self.frm1)
        self.lblRazonSocial.configure(text='Razon social', width=12)
        self.lblRazonSocial.place(anchor="nw", x=210, y=40)

        #Captura razón social del Proveedor
        self.razonSocial = ttk.Entry(self.frm1)
        self.razonSocial.configure(width=36)
        self.razonSocial.place(anchor="nw", x=290, y=40)

        #Etiqueta ciudad del Proveedor
        self.lblCiudad = ttk.Label(self.frm1)
        self.lblCiudad.configure(text='Ciudad', width=7)
        self.lblCiudad.place(anchor="nw", x=540, y=40)
```

```

#Captura ciudad del Proveedor
self.ciudad = ttk.Entry(self.frm1)
self.ciudad.configure(width=30)
self.ciudad.place(anchor="nw", x=590, y=40)

#Separador
self.separador1 = ttk.Separator(self.frm1)
self.separador1.configure(orient="horizontal")
self.separador1.place(anchor="nw", width=800, x=0, y=79)

#Etiqueta Código del Producto
self.lblCodigo = ttk.Label(self.frm1)
self.lblCodigo.configure(text='Código', width=7)
self.lblCodigo.place(anchor="nw", x=10, y=120)

#Captura el código del Producto
self.codigo = ttk.Entry(self.frm1)
self.codigo.configure(width=13)
self.codigo.place(anchor="nw", x=60, y=120)

#Etiqueta descripción del Producto
self.lblDescripcion = ttk.Label(self.frm1)
self.lblDescripcion.configure(text='Descripción', width=11)
self.lblDescripcion.place(anchor="nw", x=220, y=120)

#Captura la descripción del Producto
self.descripcion = ttk.Entry(self.frm1)
self.descripcion.configure(width=36)
self.descripcion.place(anchor="nw", x=290, y=120)

#Etiqueta unidad o medida del Producto
self.lblUnd = ttk.Label(self.frm1)
self.lblUnd.configure(text='Unidad', width=7)
self.lblUnd.place(anchor="nw", x=540, y=120)

#Captura la unidad o medida del Producto
self.unidad = ttk.Entry(self.frm1)
self.unidad.configure(width=10)
self.unidad.place(anchor="nw", x=590, y=120)

#Etiqueta cantidad del Producto
self.lblCantidad = ttk.Label(self.frm1)
self.lblCantidad.configure(text='Cantidad', width=8)
self.lblCantidad.place(anchor="nw", x=10, y=170)

#Captura la cantidad del Producto
self.cantidad = ttk.Entry(self.frm1)
self.cantidad.configure(width=12)
self.cantidad.place(anchor="nw", x=70, y=170)

#Etiqueta precio del Producto
self.lblPrecio = ttk.Label(self.frm1)
self.lblPrecio.configure(text='Precio $', width=8)
self.lblPrecio.place(anchor="nw", x=170, y=170)

#Captura el precio del Producto
self.precio = ttk.Entry(self.frm1)
self.precio.configure(width=15)
self.precio.place(anchor="nw", x=220, y=170)

#Etiqueta fecha de compra del Producto
self.lblFecha = ttk.Label(self.frm1)
self.lblFecha.configure(text='Fecha', width=6)
self.lblFecha.place(anchor="nw", x=350, y=170)

#Captura la fecha de compra del Producto
self.fecha = ttk.Entry(self.frm1)
self.fecha.configure(width=10)
self.fecha.place(anchor="nw", x=390, y=170)

```

```

#Separador
self.separador2 = ttk.Separator(self.frm1)
self.separador2.configure(orient="horizontal")
self.separador2.place(anchor="nw", width=800, x=0, y=220)

#tablaTreeView
self.style=ttk.Style()
self.style.configure("estilo.Treeview", highlightthickness=0, bd=0, background="#e0e0e0", 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': 'nsw'})])

#Árbol para mostrar los datos de la B.D.
self.treeProductos = ttk.Treeview(self.frm1, style="estilo.Treeview")

self.treeProductos.configure(selectmode="extended")

# Etiquetas de las columnas para el TreeView
self.treeProductos["columns"]=( "Codigo", "Descripcion", "Und", "Cantidad", "Precio", "Fecha")
# Características de las columnas del árbol
self.treeProductos.column("#0", anchor="w",stretch=True,width=3)
self.treeProductos.column("Codigo", anchor="w",stretch=True,width=3)
self.treeProductos.column("Descripcion", anchor="w",stretch=True,width=150)
self.treeProductos.column("Und", anchor="w",stretch=True,width=3)
self.treeProductos.column("Cantidad", anchor="w",stretch=True,width=3)
self.treeProductos.column("Precio", anchor="w",stretch=True,width=8)
self.treeProductos.column("Fecha", anchor="w",stretch=True,width=3)

# Etiquetas de columnas con los nombres que se mostrarán por cada columna
self.treeProductos.heading("#0", anchor="center", text='ID / Nit')
self.treeProductos.heading("Codigo", anchor="center", text='Código')
self.treeProductos.heading("Descripcion", anchor="center", text='Descripción')
self.treeProductos.heading("Und", anchor="center", text='Unidad')
self.treeProductos.heading("Cantidad", anchor="center", text='Cantidad')
self.treeProductos.heading("Precio", anchor="center", text='Precio')
self.treeProductos.heading("Fecha", anchor="center", text='Fecha')

#Carga los datos en treeProductos
self.lee_treeProductos()
self.treeProductos.place(anchor="nw", height=560, width=790, x=2, y=230)

#Scrollbar en el eje Y de treeProductos
self.scrollbary=ttk.Scrollbar(self.treeProductos, orient='vertical', command=self.treeProductos.yview)
self.treeProductos.configure(yscroll=self.scrollbary.set)
self.scrollbary.place(x=778, y=25, height=478)

# Título de la pestaña Ingreso de Datos
self.frm1.pack(side="top")
self.tabs.add(self.frm1, compound="center", text='Ingreso de datos')
self.tabs.pack(side="top")

#Frame 2 para contener los botones
self.frm2 = ttk.Frame(self.win)
self.frm2.configure(height=100, width=800)

#Botón para Buscar un Proveedor
self.btnBuscar = ttk.Button(self.frm2)
self.btnBuscar.configure(text='Buscar')
self.btnBuscar.place(anchor="nw", width=70, x=200, y=10)

#Botón para Guardar los datos
self.btnGrabar = ttk.Button(self.frm2)
self.btnGrabar.configure(text='Grabar')
self.btnGrabar.place(anchor="nw", width=70, x=275, y=10)

#Botón para Editar los datos
self.btnEditar = ttk.Button(self.frm2)
self.btnEditar.configure(text='Editar')
self.btnEditar.place(anchor="nw", width=70, x=350, y=10)

```

```

#Botón para Elimnar datos
self.btnEliminar = ttk.Button(self.frm2)
self.btnEliminar.configure(text='Eliminar')
self.btnEliminar.place(anchor="nw", width=70, x=425, y=10)

#Botón para cancelar una operación
self.btnCancel = ttk.Button(self.frm2)
self.btnCancel.configure(text='Cancelar', width=80,command = self.limpiaCampos)
self.btnCancel.place(anchor="nw", width=70, x=500, y=10)

#Ubicación del Frame 2
self.frm2.place(anchor="nw", height=60, relwidth=1, y=755)
self.win.pack(anchor="center", side="top")

# widget Principal del sistema
self.mainwindow = self.win

#Fución de manejo de eventos del sistema
def run(self):
    self.mainwindow.mainloop()

""" ..... Métodos utilitarios del sistema ..... """
#Rutina de centrado de pantalla
def centra(self,win,ancho,alto):
    """ centra las ventanas en la pantalla """
    x = win.winfo_screenwidth() // 2 - ancho // 2
    y = win.winfo_screenheight() // 2 - alto // 2
    win.geometry(f'{ancho}x{alto}+{x}+{y}')
    win.deiconify() # Se usa para restaurar la ventana

# Validaciones del sistema
def validaIdNit(self, event):
    """ Valida que la longitud no sea mayor a 15 caracteres"""
    if event.char:
        if len(self.idNit.get()) >= 15:
            mssg.showerror('Atención!!','.. ¡Máximo 15 caracteres! ..')
        else:
            self.idNit.delete(14)

#Rutina de limpieza de datos
def limpiaCampos(self):
    """ Limpia todos los campos de captura"""
    Inventario.actualiza = None
    self.idNit.config(state = 'normal')
    self.idNit.delete(0,'end')
    self.razonSocial.delete(0,'end')
    self.ciudad.delete(0,'end')
    self.idNit.delete(0,'end')
    self.codigo.delete(0,'end')
    self.descripcion.delete(0,'end')
    self.unidad.delete(0,'end')
    self.cantidad.delete(0,'end')
    self.precio.delete(0,'end')
    self.fecha.delete(0,'end')

#Rutina para cargar los datos en el árbol
def carga_Datos(self):
    self.idNit.insert(0,self.treeProductos.item(self.treeProductos.selection())['text'])
    self.idNit.configure(state = 'readonly')
    self.razonSocial.insert(0,self.treeProductos.item(self.treeProductos.selection())['values'][0])
    self.unidad.insert(0,self.treeProductos.item(self.treeProductos.selection())['values'][3])

# Operaciones con la base de datos
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_treeProductos(self):
    """ Carga los datos y Limpia la Tabla tablaTreeView """
    tabla_TreeView = self.treeProductos.get_children()
    for linea in tabla_TreeView:
        self.treeProductos.delete(linea) # Limpia la filas del TreeView

    # Seleccionando los datos de la BD
    query = """SELECT * from Proveedor INNER JOIN Inventario WHERE idNitProv = idNit ORDER BY idNitProv"""
    db_rows = self.run_Query(query) # db_rows contine la vista del query

    # Insertando los datos de la BD en treeProductos de la pantalla
    for row in db_rows:
        self.treeProductos.insert("",0, text = row[0], values = [row[4],row[5],row[6],row[7],row[8],row[9]])

    """ Al final del for queda con la última tupla
        y se usan para cargar las variables de captura
    """

    self.idNit.insert(0,row[0])
    self.razonSocial.insert(0,row[1])
    self.ciudad.insert(0,row[2])
    self.codigo.insert(0,row[4])
    self.descripcion.insert(0,row[5])
    self.unidad.insert(0,row[6])
    self.cantidad.insert(0,row[7])
    self.precio.insert(0,row[8])
    self.fecha.insert(0,row[9])

def adiciona_Registro(self, event=None):
    """Adiciona un producto a la BD si la validación es True"""
    pass

def editaTreeProveedores(self, event=None):
    """ Edita una tupla del TreeView"""
    pass

def eliminaRegistro(self, event=None):
    """Elimina un Registro en la BD"""
    pass

if __name__ == "__main__":
    app = Inventario()
    app.run()

```