TIC

Lección 3: Ejercitación Aplicativo Tareas

Tiempo de ejecución: 4 horas

Planteamiento de la sesión

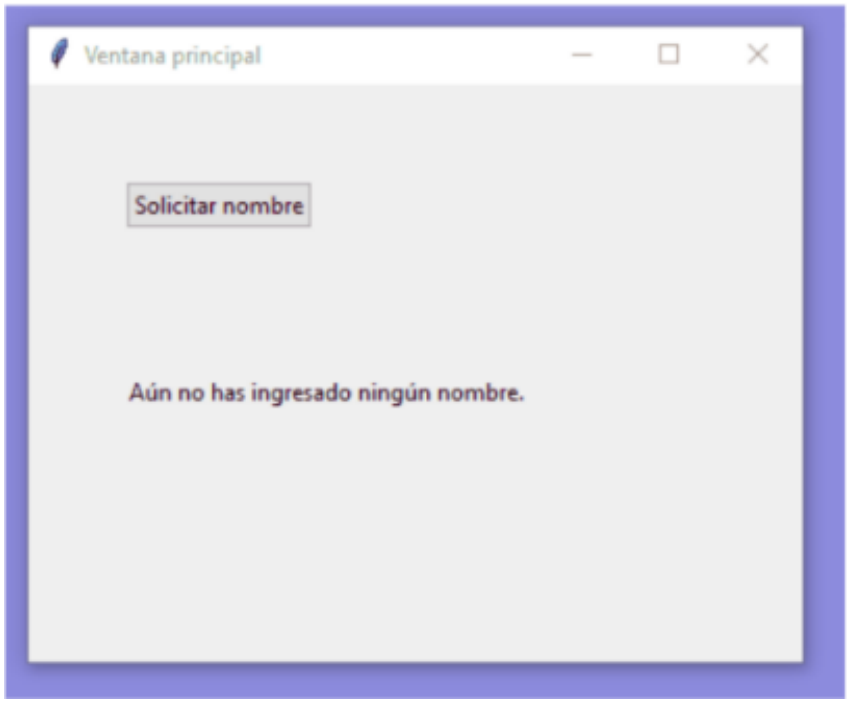
Con los conocimientos adquiridos es hora de trabajar en un proyecto un poco más retador y es un aplicativo para la gestión de tareas, para este aplicativo vamos a contar con los siguientes elementos:

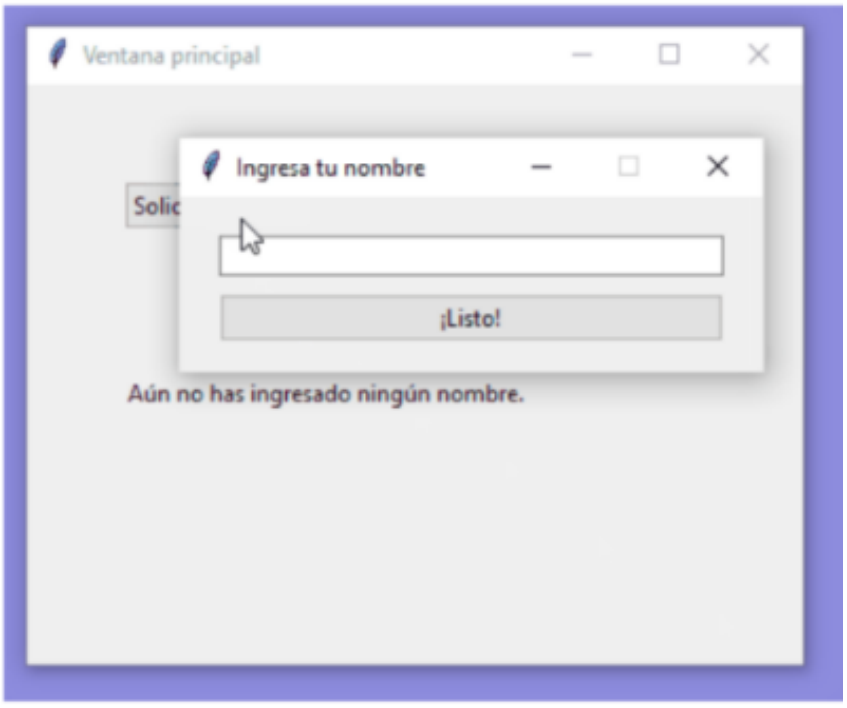
Cada tarea debe tener un nombre y un estado, los estados son **pendiente, en progreso y finalizado.** El nombre de la tarea es un texto que indica lo que se debe, por ejemplo “Comprar leche en el super”.

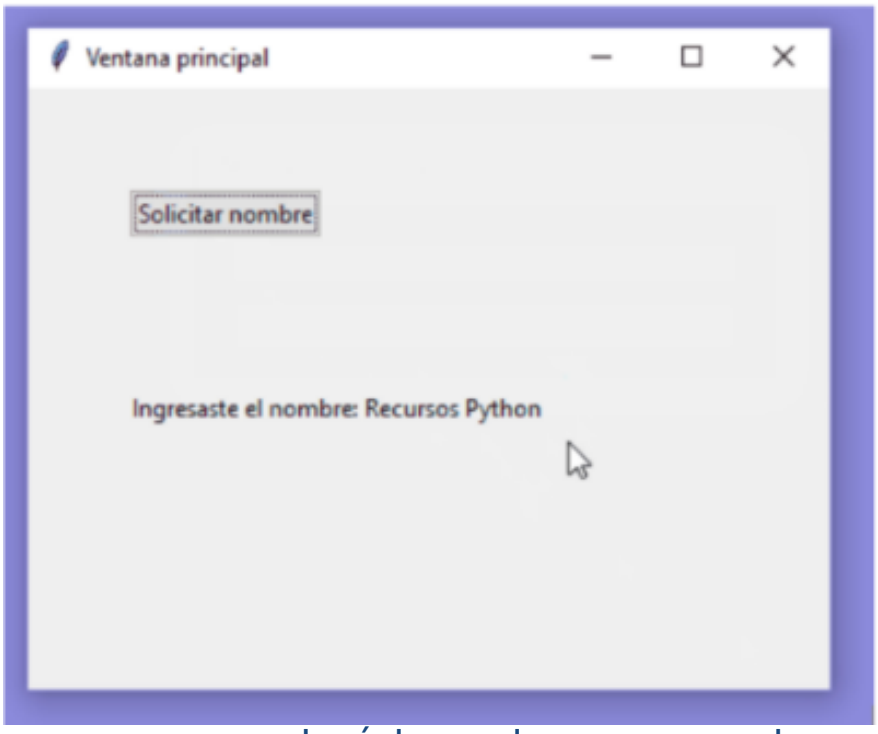
El sistema debe contar con un formulario que permita la creación de tareas, que por defecto siempre se pondrán en estado **pendiente,** el formulario debe constar de un campo de texto y un botón para confirmar la tarea creada.

Las tareas deben poder ser actualizadas en sus estados, es decir que una tarea debe utilizar un mecanismo para pasar de un estado a otro. Las tareas deben poder eliminarse.

Como reto adicional para hacer más completo nuestro proyecto, podemos utilizar ventanas adicionales para el formulario, así no nos ocupa espacio en el listado de tareas, algo como lo siguiente:


Ventana principal
Solicitar nombre
Aún no has ingresado ningún nombre.


Ventana principal
Ingresa tu nombre
¡Listo!
Aún no has ingresado ningún nombre.



Una guía que nos ayudará la podemos encontrar en Ventanas secundarias en Tcl/Tk (tkinter) – Recursos Python
Otro elemento que nos suma, es que podamos editar el texto de las tareas y modificar el orden en el que aparecen, así sabremos a qué darle más prioridad.

Un reto final para este proyecto, es que podamos guardar la lista de tareas en un archivo txt o csv y que podamos recuperar el listado de tareas desde un archivo txt o csv.

Un par de guías que nos puede ayudar aquí son:

- Examinar archivo o carpeta en Tk (tkinter) – Recursos Python
- Bloc de notas simple con Tk (tkinter) – Recursos Python

A continuación se presenta un código que se puede tomar como base para completar el reto, el código no cumple todo lo que se solicita, la interfaz tiene fuente pequeña y los tamaños de los elementos pueden ser pequeños, lo que genera una experiencia de usuario regular.

A continuación se presenta un código que se puede tomar como base para completar el reto, el código no cumple todo lo que se solicita, la interfaz tiene fuente pequeña y los tamaños de los elementos pueden ser pequeños, lo que genera una experiencia de usuario regular.

Considere modificar la apariencia de los componentes y agregar las
funcionalidades faltantes.

```
import tkinter as tk
from tkinter import messagebox
import tkinter.simpledialog
import csv

class TaskManagerApp:
def __init__(self, master):
self.master = master
self.master.title("Task Manager App")

self.tasks = []
self.load_tasks_from_file()

self.task_frame = tk.Frame(self.master)
self.task_frame.pack(padx=10, pady=10)

self.task_listbox = tk.Listbox(self.task_frame, width=50, height=15)
self.task_listbox.pack(side=tk.LEFT, fill=tk.BOTH, expand=True)

self.scrollbar = tk.Scrollbar(self.task_frame)
self.scrollbar.pack(side=tk.RIGHT, fill=tk.Y)
```

```
self.task_listbox.config(yscrollcommand=self.scrollbar.set)

self.scrollbar.config(command=self.task_listbox.yview)

self.add_task_button = tk.Button(self.master, text="Add Task"
,command=self.add_task)
self.add_task_button.pack(pady=5)

self.delete_task_button = tk.Button(self.master, text= "Delete
Task",command=self.delete_task)
self.delete_task_button.pack(pady=5)

self.refresh_task_list()

def refresh_task_list(self):
self.task_listbox.delete(0, tk.END)
for task in self.tasks:
self.task_listbox.insert(tk.END,f"{task['description']}- {task['status']}")

def add_task(self):
description =tkinter.simpledialog.askstring("Add Task","Enter task
description:")
if description:
self.tasks.append({"description": description,"status": "Pending"})

self.refresh_task_list()
self.save_tasks_to_file()

def delete_task(self):
selected_index = self.task_listbox.curselection()
if selected_index:
task_index = selected_index[0]
del self.tasks[task_index]
self.refresh_task_list()
self.save_tasks_to_file()
```

```
def complete_task(self):
selected_index = self.task_listbox.curselection()
if selected_index:
task_index = selected_index[0]
self.tasks[task_index]["status"] ="Completed"
self.refresh_task_list()
self.save_tasks_to_file()

def save_tasks_to_file(self):
with open("tasks.csv","w",newline=,encoding="utf-8") as file:
writer = csv.DictWriter(file, fieldnames=["description","status"])

writer.writeheader()
for task in self.tasks:
writer.writerow(task)

def load_tasks_from_file(self):
try:with open("tasks.csv","r",encoding="utf-8") as file:
reader = csv.DictReader(file)
self.tasks = [row for row in reader]
except FileNotFoundError:
# If file doesn't exist, initialize an empty list of tasks
self.tasks = []

def main():
root = tk.Tk()
app = TaskManagerApp(root)
root.mainloop()

if __name__ =="__main__":
main()
```


APPLY