https://github.com/GracielaBerio/Proyecto-final--Prog-I.git

Qué contiene Nuestro GitHub:

```
Proyecto-final-Prog-I/

├── .gitignore
├── LICENSE
├── README.md
├── con Tkinter/
│   └── … (archivos para la versión con interfaz gráfica)
├── sin Tkinter/
│   └── … (archivos para la versión sin interfaz gráfica)
└── planificador/
    └── … (el módulo principal “planificador”)
```

Explicación de cada elemento

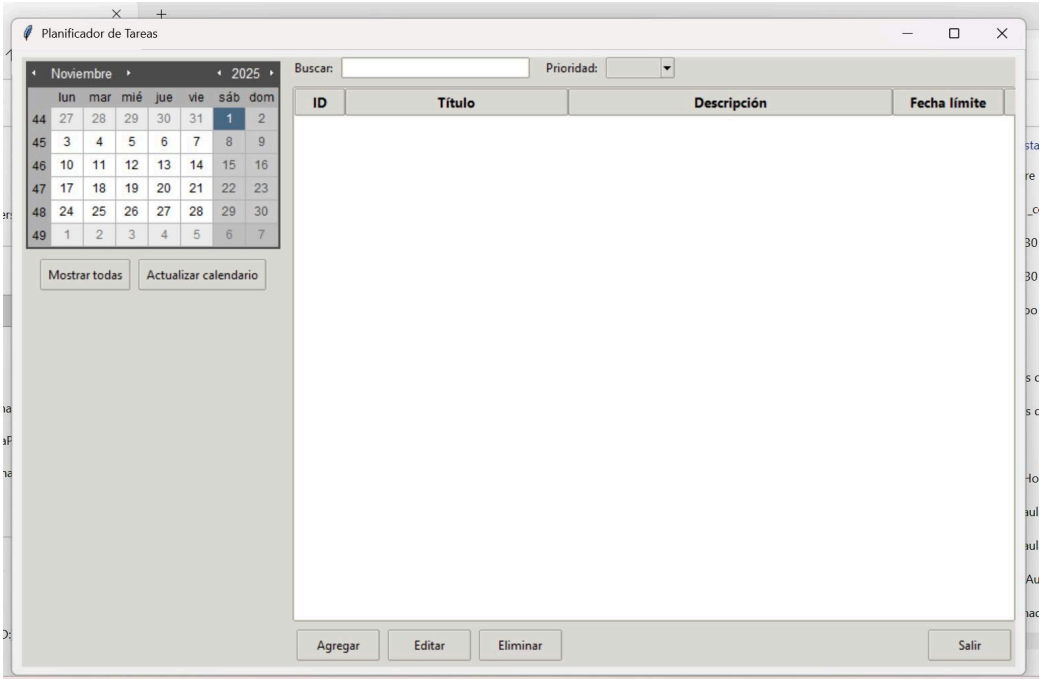
.gitignore
Un archivo estándar de Git que indica qué archivos o carpetas deben ignorarse (por ejemplo archivos temporales, de configuración local, compilados) para que no se suban al repositorio.

LICENSE
Contiene la licencia del proyecto, en este caso parece indicar que el proyecto se distribuye bajo la licencia GPL-3.0. Esto define cómo puede usarse, modificarse o redistribuir el software.

README.md
Es el fichero de introducción/documentación básica del proyecto: qué hace, cómo usarlo, cuáles son sus objetivos. En este caso el README dice que el proyecto es un “Planificador de tareas educativas”, que funciona como aplicación de escritorio, que permite agregar tareas, ver fechas, prioridades, etc. (como se ve en la descripción del repositorio).

con Tkinter/

Diseño de la interfaz gráfica, con tkinter:



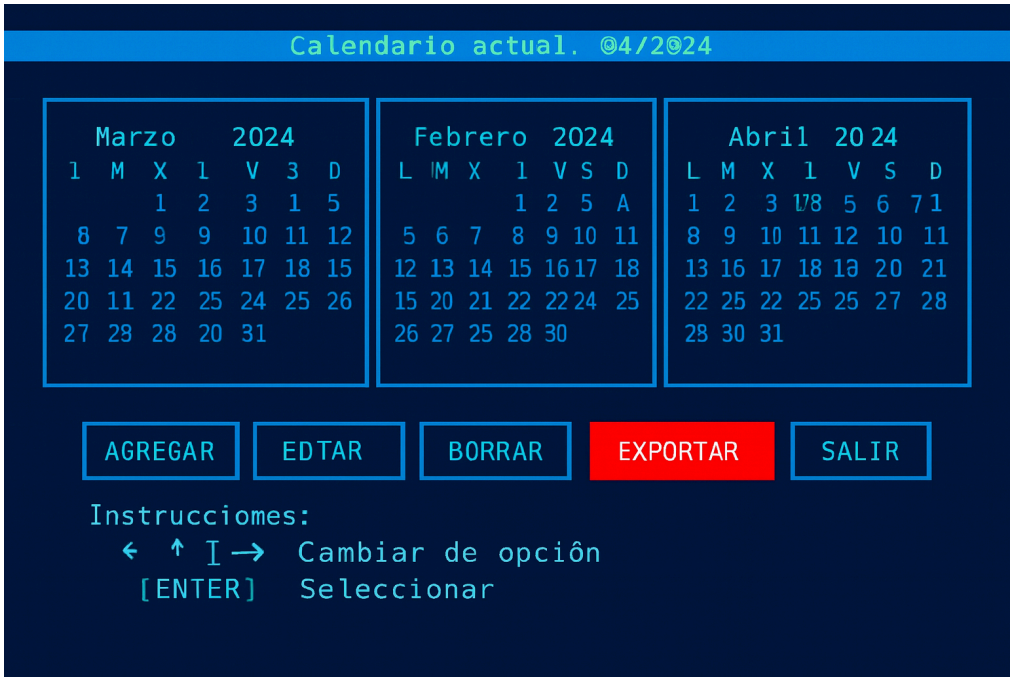
Esta carpeta agrupa probablemente la versión del proyecto con interfaz gráfica basada en Tkinter (biblioteca estándar de Python para GUIs). Es decir: para que el usuario interactúe mediante ventanas, botones, calendario visual, etc.
Dentro de ella estarían los archivos de código que hacen la interfaz, la lógica de eventos, la base de datos, etc.

- **Agregar tareas** con título, descripción, fecha límite y prioridad.
- **Editar o eliminar tareas** guardadas en la base de datos (`tareas.db`).
- **Visualizar un calendario interactivo** (usando `tk calendar`, si está instalado).
- **Resaltar tareas por prioridad** (rojo, amarillo o verde según nivel).
- **Buscar y filtrar tareas** por nombre o prioridad.
- **Mostrar notificaciones** cuando una tarea está próxima a vencer.

La base de datos **SQLite (`tareas.db`)** guarda la información de forma local.

sin Tkinter/

Diseño de la interfaz gráfica, sin tkinter:



Esta carpeta contiene la versión “sin interfaz gráfica” (o con interfaz más simple) del proyecto. Quizás una versión de consola o básica para comprobar la lógica sin la capa visual. Puede ser útil para pruebas, para comparar o para entornos donde no se quiera GUI.

Sus funciones suelen ser las mismas:

- Agregar tareas.
- Listarlas en texto plano.
- Eliminar o actualizar tareas.
- Es ideal para probar la lógica sin necesidad de abrir ventanas ni instalar librerías gráficas.
- incluye un **readme** en Júpiter, con funcionalidades de Diseño, estructura de módulos
- Librerías implementadas también explicadas en júpiter
- Aplicación de py tests
- Versión portable(distr\), en un único archivo EXE generado con pyinstaller solo con ejecutar este archivo desde cualquier dispositivo con windows debería de funcionar.

planificador/ (carpeta de Pruebas)

- Esta carpeta tiene el módulo principal del proyecto final y funcional.
- En esta carpeta se implementan los py_ tests
- Se crea la version portable dentro de la carpeta distr\

Nota Importante:

En cada carpeta de se describen como utilizar las distintas librerías implementadas, y explicaciones extendidas en archivos de formato Júpiter que también tienen la misma profundidad y relevancia que este material en drive ambos son complementarios y no se reemplazan.

Requerimientos del sistema — Planificador de Tareas

Sistema operativo:

- Windows 10 o superior (versión de 64 bits recomendada)

Espacio en disco:

- 100 MB libres para instalación y archivos de datos
- Memoria RAM:
- 4 GB mínimo (8 GB recomendados)

1. Instrucciones de instalación y ejecución

Instalación Versión Portable:

1. Descargá el archivo Proyecto-final--Prog-I/planificador /dist/Planificador_de_tareas.exe
2. Copialo en una carpeta de tu elección (por ejemplo, Escritorio o Documentos).
3. Si el sistema muestra un aviso de seguridad de Windows, elegí Más información, Ejecutar de todos modos.
4. (Opcional) Si se requiere Java o Python, asegúrate de que esté instalado antes de abrir el programa.
5. Hacé doble clic en el archivo para iniciar el Planificador de Tareas.

Instalación versión desarrollador:

Descargar o clonar el repositorio desde GitHub:
git clone https://github.com/GracielaBerio/Proyecto-final--Prog-I.git
Ingresar a la carpeta con Tkinter.
Instalar la librería tkcalendar ejecutando en la terminal:
pip install tkcalendar

Ejecución:

Abrir una terminal en la carpeta del proyecto y ejecutar el archivo principal con el comando:
python Planificador_de_tareas.py

El programa abrirá una ventana donde el usuario puede agregar, editar y eliminar tareas, ver las fechas límite en un calendario, filtrar por prioridad (Alta, Media, Baja), exportar e importar datos en formatos CSV y JSON, y recibir notificaciones de tareas próximas a vencer.

2. Descripción del diseño, hitos y decisiones de desarrollo

El proyecto fue diseñado como una aplicación de escritorio educativa para gestionar tareas.
Existen dos versiones: una sin interfaz gráfica (solo consola) y otra con interfaz visual construida con Tkinter.

Hitos principales del desarrollo:

Diseño de la base de datos SQLite, con una tabla llamada “tareas” que contiene los campos id, título, descripción, fecha límite y prioridad.

Implementación de las funciones básicas de alta, baja, modificación y listado de tareas (CRUD).

Incorporación de la interfaz gráfica con Tkinter y el uso del widget Treeview para visualizar las tareas en forma de tabla.

Integración del calendario mediante tkcalendar, lo que permite seleccionar y marcar fechas de manera visual.

Inclusión de funciones para exportar e importar datos en CSV y JSON.

Agregado de un sistema de notificaciones que alerta sobre tareas próximas a vencer.

Decisiones de desarrollo:

- Se eligió SQLite como base de datos por ser simple, local y no requerir instalación adicional.
- Tkinter se eligió por ser parte del propio Python y permitir construir interfaces limpias y funcionales.
- Se optó por mantener los nombres y textos en español para favorecer la comprensión del código en contextos educativos.

El proyecto está estructurado de forma modular, separando la lógica de la interfaz gráfica y las operaciones con la base de datos.

3. Justificación del uso de las librerías elegidas

tkinter: se utiliza para construir la interfaz gráfica. Es parte de la biblioteca estándar de Python, lo que evita dependencias externas y facilita el aprendizaje.

tkcalendar: se utiliza para mostrar un calendario interactivo y facilitar la selección de fechas. Mejora la experiencia del usuario al visualizar tareas por colores según la prioridad.

sqlite3: se utiliza para la base de datos local. Es confiable, rápida y no requiere instalación.

csv y json: se usan para exportar e importar datos. Permiten guardar copias de las tareas y compartir información fácilmente.

ttk (de tkinter): se usa para mejorar el diseño visual y estilizar los elementos de la interfaz.

Estas librerías fueron elegidas por su facilidad de uso, compatibilidad con distintos sistemas operativos y su valor didáctico, ya que permiten comprender distintos aspectos del desarrollo de software: interfaz, datos y persistencia.

4. Fundamento didáctico: aprendizajes, desafíos y reflexiones

Aprendizajes:

Durante el desarrollo del proyecto se aplicaron conocimientos de programación estructurada, manejo de bases de datos, diseño de interfaces y pensamiento computacional.
Se comprendió la importancia del diseño modular, la validación de datos y la persistencia de información en aplicaciones reales.

Desafíos enfrentados:

Los principales desafíos fueron integrar correctamente el calendario tkcalendar, sincronizar la base de datos con la interfaz gráfica, mantener la aplicación estable, y lograr un diseño claro y agradable para el usuario.
También se trabajó en las validaciones de fechas, formatos y en la implementación del sistema de notificaciones.

Reflexiones del proceso:

El proyecto permitió unir conceptos teóricos con la práctica, desarrollando una aplicación funcional y útil.
A nivel educativo, promovió el trabajo autónomo, la resolución de problemas y la planificación de tareas.

También demostró que la programación puede ser una herramienta creativa para organizar y comunicar ideas visualmente.
En síntesis, el proceso fue una experiencia formativa que integró habilidades técnicas, expresivas y organizativas, consolidando aprendizajes tanto en el área de programación como en la de comunicación visual.

Webgrafía:

Downey, A., Elkner, J., & Meyers, C. (2002). *Aprenda a pensar como un programador con Python*. Recuperado de: https://argentinaenpython.com/quiero-aprender-python/aprenda-a-pensar-como-un-programador-con-python.pdf

El Libro De Python. (s. f.). *Top 50 Ejemplos y Ejercicios*. El Libro De Python. Recuperado de : https://ellibrodepython.com/ejemplos-ejercicios-python

Python Software Foundation. (2024). *Tkinter — Python interface to Tcl/Tk*. Python.org. Recuperado de https://docs.python.org/es/3/library/tkinter.html

Álvarez, A. (s.f.). *Guía Tkinter*. Read the Docs.
Recuperado de
https://buildmedia.readthedocs.org/media/pdf/guia-tkinter/latest/guia-tkinter.pdf

Informatec Digital. (2023). *Tkinter para interfaces gráficas en Python*.
Recuperado de https://informatecdigital.com/tkinter-para-interfaces-graficas-en-python/

tkcalendar. (s. f.). tkcalendar 1.5.0 documentation [Documentación en línea].Recuperado de https://tkcalendar.readthedocs.io/en/stable/

Hipp, D. R. (2024). SQLite Documentation. SQLite.org.

Recuperado de: https://www.sqlite.org/docs.html

DB Browser for SQLite. (2024). DB Browser for SQLite – Official Project Site.

Recuperado de https://sqlitebrowser.org/

Python Software Foundation. (2024). sqlite3 — DB-API 2.0 interface for SQLite databases. Python.org.Recuperado de: https://docs.python.org/3/library/sqlite3.html