

GESTOR DE AYUDAS COMUNITARIAS

Uso de Tkinter y Python para ofrecer una manera diferente de reciclaje.

PROYECTO
FINAL DE
ESTRUCTURA
Y
ALGORITMOS
DE DATOS



ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS



ALGORITMOS Y ESTRUCTURAS DE DATOS

ASIGNATURA: Algoritmos y Estructuras de Datos

PROFESOR: Ing. Loarte Byron

PERÍODO ACADÉMICO: 2025-B

PRESENTACIÓN DEL PROYECTO



GRUPO:

ALEJANDRO TACO

MARJORIE VALDIVIEZO

LIZBETH SANCHEZ

OÑA GABRIELA

CARLOSAMA JOSHUA

DOCUMENTACIÓN DEL DESARROLLO DE ACTIVIDADES

POLI AYUDAS COMUNITARIAS

OBJETIVOS ESPECÍFICOS

- Documentar de manera eficiente las funciones, librerías, estructuras de datos y herramientas usadas de nuestro proyecto para guardar constancia de progreso y futura guía para proyectos más grandes.
- Investigar formas de integrar nuevas herramientas a los proyectos para el mejor manejo de datos y su visualización como Tkinter o Github.
- Integrar todos los conocimientos vistos en clases más una investigación personal en grupo para poder presentar una idea que puede ser mejorada a futuro.

DESARROLLO Y RESULTADOS

- Librerías

1. Tkinter

Tkinter es la librería estándar de Python para crear interfaces gráficas. Se utiliza para ventanas, botones, cuadros de texto, etiquetas y tablas.

Métodos usados en el proyecto:

- tk.Tk() Crea la ventana principal de la aplicación.
- tk.Toplevel() Crea ventanas secundarias sin cerrar la principal.
- tk.Label() Permite mostrar texto o imágenes dentro de la interfaz.
- tk.Entry() Se usa para ingresar datos como usuario, contraseña o ID.
- tk.Button() Crea botones interactivos para ejecutar acciones.

2. Tkinter.ttk

Extensión de Tkinter que permite usar componentes más modernos, como tablas y estilos mejorados.

Métodos usados en el proyecto:

- ttk.Treeview() Se utiliza para mostrar la tabla de usuarios.
- treeview.insert() Inserta filas dentro de la tabla.
- treeview.delete() Elimina filas de la tabla antes de actualizarla.
- treeview.heading() Define los encabezados de las columnas.

- treeview.column() Configura el ancho y alineación de cada columna.

3. Tkinter.messagebox

Se usa para mostrar mensajes emergentes al usuario, como alertas, errores o confirmaciones.

Métodos usados en el proyecto:

- messagebox.showinfo() Muestra mensajes informativos (éxito, bienvenida).
- messagebox.showwarning() Advierte sobre campos vacíos o acciones inválidas.
- messagebox.showerror() Indica errores como usuario incorrecto o ID no encontrado.

4. Os

Permite interactuar con el sistema operativo, manejar rutas, archivos y directorios.

Métodos usados en el proyecto:

- os.path.exists() Verifica si un archivo o carpeta existe.
- os.path.join() Une rutas de forma segura entre carpetas.
- os.system() Ejecuta otros archivos .py desde el programa.
- os.path.getmtime() Obtiene la última modificación de un archivo.
- os.path.dirname() Obtiene el directorio donde está el archivo actual.

5. PIL (Pillow)

Se utiliza para cargar, redimensionar y mostrar imágenes dentro de la interfaz gráfica.

Métodos usados en el proyecto:

- Image.open() Abre una imagen desde el disco.
- Image.resize() Cambia el tamaño de una imagen para ajustarla a la ventana.
- ImageTk.PhotoImage() Convierte la imagen para que Tkinter pueda mostrarla.
- Image.LANCZOS Mejora la calidad de la imagen al redimensionar.

6. *Re*

Se usa para validar texto mediante expresiones regulares, evitando caracteres no permitidos.

Métodos usados en el proyecto:

- `re.match()` Verifica que el texto cumpla un patrón definido.
- `^[A-Za-z0-9]*$` Permite solo letras y números.
- Validación en tiempo real Se usa junto con `validatecommand` en `Entry`.

7. *Time*

Permite manejar tiempos y sincronización, usada para actualizaciones automáticas.

Métodos usados en el proyecto:

- `after()` Ejecuta una función cada cierto intervalo de tiempo.
- Estructuras de datos
- *Clases*

Se utilizaron clases para organizar el código en módulos lógicos, permitiendo separar la interfaz, la lógica del programa y el manejo de datos. Esto facilita el mantenimiento, la lectura y la reutilización del código. Cada ventana principal del sistema se representa mediante una clase. Además, las clases permiten agrupar métodos relacionados dentro de un mismo contexto.

```
class VentanaPrincipal:  
  
    def __init__(self):  
        self.app = tk.Tk()  
        self.app.title("Ventana Principal")  
        self.app.geometry("1200x550")  
        self.app.config(bg=color_fondo)  
  
        self.ultima_modificacion = 0  
  
        self.crear_layout()  
        self.actualizar_tabla()  
        self.app.mainloop()  
  
    if __name__ == "__main__":  
        VentanaPrincipal()
```

• *Listas (Arreglos)*

Las listas se usan para almacenar temporalmente los datos de los usuarios leídos desde el archivo de texto. Permiten recorrer, modificar y mostrar la información en

la tabla. También facilitan agregar valores faltantes como el puntaje. Son clave para manejar múltiples registros sin una base de datos.

```
for linea in f:
    partes = [p.strip() for p in linea.split("|")]

    if len(partes) == 7:
        partes.append("0")

    if len(partes) == 8:
        datos.append(partes)
```

- *Archivos*

El sistema utiliza archivos de texto como medio de almacenamiento permanente de los datos. Cada línea representa un usuario y sus campos están separados por un símbolo. Esta estructura permite guardar y recuperar información sin usar bases de datos externas.

```
with open(self.archivo, "a", encoding="utf-8") as f:
    f.write(f"{nuevo_id} | {nom} | {ape} | {self.actividad.get()} | Disponible | {tel} | {ciu}\n")
```

- *Diccionarios*

Aunque no se usan diccionarios explícitos, la información se maneja de forma estructurada por campos (ID, nombre, apellido, etc.), simulando un registro. Cada posición de la lista representa un atributo específico del usuario. Esto permite acceder a los datos por índice de manera ordenada.

```
self.tabla = ttk.Treeview(
    panel_der,
    columns=(
        "ID",
        "Nombre",
        "Apellido",
        "Actividad",
        "Estado",
        "Telefono",
        "Ciudad",
        "Puntaje"
    ),
```

- Funciones

Resumen general de funciones del programa

El sistema desarrollado es una **aplicación de escritorio en Python con Tkinter** que permite gestionar actividades comunitarias de usuarios de manera organizada, usando archivos de texto como almacenamiento.

Inicio de sesión

Permite el acceso al sistema mediante un usuario administrador.
Controla que solo personas autorizadas puedan gestionar los datos del sistema.

Registro de usuarios

Permite ingresar nuevos usuarios al sistema con datos como nombre, apellido, teléfono, ciudad y actividad comunitaria asignada.
Cada usuario recibe un **ID único** generado automáticamente.

Búsqueda de usuarios

Permite buscar un usuario específico mediante su ID.
Muestra su información para revisión sin modificar los datos.

Actualización de datos

Permite modificar la información de un usuario existente, como su actividad o datos personales.
Los cambios se guardan directamente en el archivo del sistema.

Eliminación de usuarios

Permite eliminar un usuario del sistema usando su ID.
El registro se borra del archivo y deja de mostrarse en la tabla.

Completar actividad

Permite marcar una actividad como **completada o incompleta**.

Si se marca como completada, el usuario recibe **100 puntos**, los cuales solo se asignan una vez.

Sistema de recompensas

Permite canjear puntos acumulados por recompensas visuales.

Al adquirir una recompensa, se descuentan los puntos y se muestra una vista previa del objeto.

Visualización en tabla

Muestra todos los usuarios registrados en una tabla dinámica.

La tabla se actualiza automáticamente cuando el archivo de datos cambia.

Gestión de archivos

Guarda y lee los datos de los usuarios desde un archivo .txt.

Permite mantener la información incluso al cerrar el programa.

- Github

Progreso del proyecto en GitHub

El desarrollo del sistema se realizó de forma **incremental**, subiendo avances progresivos al repositorio, lo que permitió construir el programa paso a paso.

1. Estructura inicial del proyecto

Se subieron los primeros archivos base del proyecto, incluyendo la estructura de carpetas (Datos, Recursos) y los archivos principales en Python.

2. Implementación del Login

Se añadió el sistema de inicio de sesión con validaciones de usuario y contraseña, asegurando el acceso controlado al sistema.

3. Registro y visualización de usuarios

Se incorporó la funcionalidad para registrar usuarios y mostrar sus datos en una tabla dinámica, utilizando archivos .txt como almacenamiento.

4. Funciones de gestión (CRUD)

Se agregaron las funciones de:

- Buscar usuarios
- Actualizar información
- Eliminar registros

Cada función se desarrolló en archivos independientes para mantener el código organizado.

5. Sistema de actividades y puntaje

Se implementó la opción de marcar actividades como completadas o incompletas, incluyendo la asignación automática de puntaje.

6. Sistema de recompensas

Se añadió una interfaz para canjear puntos por recompensas visuales, integrando imágenes y actualización del puntaje.

7. Ajustes finales y correcciones

Se realizaron mejoras en:

- Validaciones de entrada
- Evitar duplicación de IDs
- Sincronización automática de la tabla
- Manejo correcto de archivos y rutas

Beneficios del uso de GitHub en el proyecto

- Permite **guardar el avance del proyecto** de forma segura
- Facilita **corregir errores sin perder versiones anteriores**

- Ayuda a mantener una **organización clara del código**
- Permite compartir el proyecto fácilmente para revisión o evaluación

The screenshot shows a GitHub repository page for 'SistemaDeGestionComunitaria'. The repository is public and has 1 branch and 0 tags. The main list of commits shows the following entries:

Author	Commit Message	Time Ago
JoshuaCarlosama	Actualizar recompensas.py Fix	5eb4ffd · 3 hours ago
	Integración de archivos Repositorio	5 hours ago
	Integración de Recursos Repositorio	5 hours ago
	Initial commit	yesterday
	Actualizar actualizar.py Fix	3 hours ago
	Actualizar agregar.py Fix	3 hours ago
	Actualizar buscar.py Fix	3 hours ago
	Actualizar completar.py Fix	3 hours ago
	Actualizar eliminar.py Fix	3 hours ago
	Arreglo de login.py Fix	3 hours ago
	Actualizar main.py Fix	3 hours ago
	Actualizar recompensas.py Fix	3 hours ago
	Integración de requerimientos del sistema	5 hours ago

About: Interfaz Gráfica con Python para un sistema de gestión de ayudas comunitarias.

Releases: No releases published. Create a new release.

Packages: No packages published. Publish your first package.

Languages: Python 100.0%

Suggested workflows: Based on your tech stack.

- SLSA Generic generator**: Configure. Generate SLSA3 provenance for your existing release workflows.
- PyLint**: Configure. Lint a Python application with pylint.

CONCLUSIONES Y RECOMENDACIONES

Conclusiones

- Desarrollamos una interfaz dinámica modular para gestionar ayudas comunitarias con ayuda de Tkinter y librerías para futura escalabilidad
- Integramos técnicas de manejo de datos como clases, arreglos o archivos que nos ayudará entender como es el flujo de un programa.
- Expusimos nuestros avances dentro de la documentación y para futura guía autónoma o compartida para ayudar a recabar la información de forma efectiva a posteriori.

Recomendaciones

- Tener una rama de actividades a realizar antes de empezar a codificar, es decir, planificar antes de empezar a realizar la idea.

- Usar herramientas, api's o guías en internet para facilitar la forma de entender el código y hacerlo más eficiente
- Trabajar en grupo con la ayuda de programas de colaboración o usando medios de comunicación.

BIBLIOGRAFÍA:

codigo Daniel037. (2022, 25 enero). *38 - Ventanas emergentes - Tkinter* [Vídeo]. YouTube. https://www.youtube.com/watch?v=fbVf8nZx_Ps

Dimas. (2021, 24 agosto).  *Curso PYTHON: INTERFACES GRÁFICAS [con TKINTER]* [Vídeo]. YouTube. <https://www.youtube.com/watch?v=MpkTYMzhV0A>

Enrique Barros. (2025, 18 octubre). *Cómo usar Treeview en Tkinter: tablas y listas en tu GUI con Python* [Vídeo]. YouTube. <https://www.youtube.com/watch?v=RBfHqJ8lWDI>

Fitzpatrick, M. (2025, 9 julio). *Tkinter Layouts, designing Python GUI*. Python GUIs. https://www.pythonguis.com.translate.goog/tutorials/use-tkinter-to-design-gui-layout/?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es&_x_tr_pto=tc

Mundo Python. (2019, 28 agosto). *Tutorial de Tkinter (Aprende tkinter en 20 minutos)* [Vídeo]. YouTube. <https://www.youtube.com/watch?v=jqRHhWjKDD8>

os — Interfaces misceláneas del sistema operativo — documentación de Python - 3.10.19. (s. f.). <https://docs.python.org/es/3.10/library/os.html>

pildorasinformaticas. (2018, 31 enero). *Curso Python. Interfaces gráficas I. Video 42* [Vídeo]. YouTube. <https://www.youtube.com/watch?v=hTUJC8HsC2I>

Pillow. (s. f.). *Pillow (PIL Fork).* https://pillow.readthedocs.io.translate.goog/en/stable/?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es&_x_tr_pto=tc

re — Regular expression operations. (s. f.). Python Documentation. <https://docs.python.org/es/3.13/library/re.html>

tkinter.messagebox — Indicadores de mensajes de Tkinter — documentación de Python - 3.10.19. (s. f.). <https://docs.python.org/es/3.10/library/tkinter.messagebox.html>

tkinter.ttk — Tk themed widgets. (s. f.). Python Documentation. <https://docs.python.org/es/dev/library/tkinter.ttk.html>