INTEGRANTES:

Atristain De La Vega Jennifher Alejandra

Canseco Yerves Denisse

DESCRIPCION GENERAL DEL PROYECTO:

Herbarium Cherubi es un programa que se basa en Python el cual permite gestionar y organizar un catálogo de plantas, ofreciendo información detallada sobre cada una de ellas. Este programa está diseñado para ayudar a los usuarios a identificar plantas, conocer sus características y aprender sobre los cuidados que cada tipo de planta necesita.

OBJETIVOS DEL PROYECTO:

- OBJETIVO GENERAL:

  El objetivo principal del programa es proporcionar a los usuarios una herramienta práctica para conocer más sobre las plantas, facilitar su cuidado y mejorar la gestión de colecciones personales o profesionales de plantas.

- OBJETIVOS ESPECIFICOS:

- Inventario de Plantas: Base de datos donde se registran diferentes tipos de plantas, con campos como el nombre común, nombre científico, familia botánica, y tipo de planta (interior, exterior, ornamental, medicinal, etc.).

- Visualización de características: Detalles sobre cada planta, como altura promedio, color de las hojas, época de floración, clima adecuado, y velocidad de crecimiento.

- Clasificación por tipo de planta: Filtros que permiten clasificar las plantas por categorías: plantas de sombra, de sol, de interior, de exterior, suculentas, cactus, etc.

- Información sobre cuidados: Guía completa sobre los cuidados necesarios para cada planta, como frecuencia de riego, cantidad de luz solar, tipo de suelo recomendado, temperatura ideal, fertilización, y control de plagas.

- TARGETING: Este sistema está orientado tanto a aficionados a la jardinería como a profesionales del ámbito botánico o agrícola. Además de personas con una variedad de plantas, con invernaderos, con sistemas de hidroponía, etc.

https://github.com/Atristain-J/Herbarium_Cherubi.git

Codigo

```
import tkinter as tk
from tkinter import messagebox, filedialog
from tkinter import ttk
from PIL import Image, ImageTk
import os
import requests  # Asegúrate de tener esta biblioteca instalada para enviar solicitudes HTTP

# Ruta del archivo donde se almacenan los usuarios y plantas
ARCHIVO_USUARIOS = "PROYECTO.txt"
ARCHIVO_PLANTAS = "plantas.txt"
usuarios = {}
plantas = []
```

```
# Configuración del ESP32
ESP32_URL = "http://<ESP32_IP_ADDRESS>/"  # Reemplaza <ESP32_IP_ADDRESS> con la dirección IP de tu ESP32

# Cargar usuarios desde el archivo PROYECTO.txt
def cargar_usuarios():
    try:
        with open(ARCHIVO_USUARIOS, "r") as archivo:
            for linea in archivo:
                usuario, contrasena = linea.strip().split(":")
                usuarios[usuario] = contrasena
    except FileNotFoundError:
        with open(ARCHIVO_USUARIOS, "w") as archivo:
            pass

# Cargar plantas desde el archivo plantas.txt
def cargar_plantas():
    global plantas
    try:
        with open(ARCHIVO_PLANTAS, "r") as archivo:
            for linea in archivo:
                nombre, temperatura, ph, humedad, ruta_imagen = linea.strip().split(",")
                if os.path.isfile(ruta_imagen):  # Verifica si la ruta de la imagen existe
                    plantas.append({
                        "nombre": nombre,
                        "temperatura": temperatura,
                        "ph": ph,
                        "humedad": humedad,
                        "ruta_imagen": ruta_imagen
                    })
                else:
                    print(f"Advertencia: La imagen {ruta_imagen} no existe.")
    except FileNotFoundError:
```

```
        with open(ARCHIVO_PLANTAS, "w") as archivo:
            pass

# Guardar un nuevo usuario en el archivo PROYECTO.txt
def guardar_usuario(usuario, contrasena):
    usuarios[usuario] = contrasena  # Guardar el usuario en el diccionario
    with open(ARCHIVO_USUARIOS, "a") as archivo:
        archivo.write(f"{usuario}:{contrasena}\n")

# Guardar las plantas en el archivo plantas.txt
def guardar_plantas():
    with open(ARCHIVO_PLANTAS, "w") as archivo:
        for planta in plantas:
            linea = f"{planta['nombre']},{planta['temperatura']},{planta['ph']},{planta['humedad']},{planta['ruta_imagen']}\n"
            archivo.write(linea)

# Función para registrar una planta
def registrar_planta(nombre_entry, temp_entry, ph_entry, humedad_entry, label_imagen):
    nombre = nombre_entry.get()
    temperatura = temp_entry.get()
    ph = ph_entry.get()
    humedad = humedad_entry.get()
    ruta_imagen = getattr(label_imagen, 'ruta_imagen', None)  # Obtenemos la ruta de la imagen

    if not ruta_imagen:
        messagebox.showerror("Error", "Por favor, selecciona una imagen.")
        return

    if nombre and temperatura and ph and humedad:
        plantas.append({
            "nombre": nombre,
            "temperatura": temperatura,
```

```
            "ph": ph,
            "humedad": humedad,
            "ruta_imagen": ruta_imagen  # Guardamos la ruta
        })
        guardar_plantas()
        messagebox.showinfo("Éxito", "Planta registrada exitosamente")
        mostrar_bienvenida()
        ventana_formulario.withdraw()
    else:
        messagebox.showerror("Error", "Completa todos los campos e ingresa una imagen.")

# Función para eliminar una planta
def eliminar_planta(planta):
    global plantas
    plantas.remove(planta)
    guardar_plantas()
    mostrar_bienvenida()

# Función para encender el sensor en el ESP32
def encender_sensor():
    try:
        requests.get(f"{ESP32_URL}encender")  # Envía una solicitud GET para encender el sensor
        messagebox.showinfo("Éxito", "Sensor encendido.")
    except Exception as e:
        messagebox.showerror("Error", f"No se pudo encender el sensor: {e}")

# Función para apagar el sensor en el ESP32
def apagar_sensor():
    try:
        requests.get(f"{ESP32_URL}apagar")  # Envía una solicitud GET para apagar el sensor
        messagebox.showinfo("Éxito", "Sensor apagado.")
    except Exception as e:
```

```
        messagebox.showerror("Error", f"No se pudo apagar el sensor: {e}")

# Función para mostrar los detalles de una planta
def mostrar_detalles_planta(planta):
    ventana_detalles = tk.Toplevel(root)
    ventana_detalles.title(f"Detalles de {planta['nombre']}")
    ventana_detalles.geometry("300x550")
    ventana_detalles.config(bg="#F0F5E6")

    ttk.Label(ventana_detalles, text=f"Nombre: {planta['nombre']}", background="#F0F5E6").pack(pady=10)
    ttk.Label(ventana_detalles, text=f"Temperatura: {planta['temperatura']} °C", background="#F0F5E6").pack(pady=10)
    ttk.Label(ventana_detalles, text=f"pH: {planta['ph']}", background="#F0F5E6").pack(pady=10)
    ttk.Label(ventana_detalles, text=f"Humedad: {planta['humedad']} %", background="#F0F5E6").pack(pady=10)

    imagen = Image.open(planta['ruta_imagen'])
    imagen_redimensionada = imagen.resize((200, 200))
    imagen_tk = ImageTk.PhotoImage(imagen_redimensionada)
    label_imagen = ttk.Label(ventana_detalles, image=imagen_tk, background="#F0F5E6")
    label_imagen.image = imagen_tk
    label_imagen.pack(pady=10)

    # Botón para eliminar la planta
    ttk.Button(ventana_detalles, text="Eliminar Planta", command=lambda: eliminar_planta(planta)).pack(pady=10)

    # Botones para controlar el sensor
    ttk.Button(ventana_detalles, text="Encender Sensor", command=encender_sensor).pack(pady=5)
    ttk.Button(ventana_detalles, text="Apagar Sensor", command=apagar_sensor).pack(pady=5)

    ttk.Button(ventana_detalles, text="Regresar", command=ventana_detalles.destroy).pack(pady=10)

# Función para mostrar la pantalla de bienvenida
def mostrar_bienvenida():
```

```
    cerrar_ventanas()

    ventana_bienvenida = tk.Toplevel(root)
    ventana_bienvenida.title("Bienvenida")
    ventana_bienvenida.geometry("400x400")
    ventana_bienvenida.config(bg="#D0F0C0")

    canvas = tk.Canvas(ventana_bienvenida, bg="#D0F0C0")
    scrollbar = ttk.Scrollbar(ventana_bienvenida, orient="vertical", command=canvas.yview)
    scrollable_frame = tk.Frame(canvas, bg="#D0F0C0")

    scrollable_frame.bind(
        "<Configure>",
        lambda e: canvas.configure(scrollregion=canvas.bbox("all"))
    )

    canvas.create_window((0, 0), window=scrollable_frame, anchor="nw")
    canvas.configure(yscrollcommand=scrollbar.set)

    canvas.pack(side="left", fill="both", expand=True)
    scrollbar.pack(side="right", fill="y")

    if plantas:
        for index, planta in enumerate(plantas):
            if os.path.isfile(planta['ruta_imagen']):
                imagen = Image.open(planta['ruta_imagen'])
                imagen_redimensionada = imagen.resize((100, 100))
                imagen_tk = ImageTk.PhotoImage(imagen_redimensionada)

                btn_planta = ttk.Button(scrollable_frame, image=imagen_tk, command=lambda p=planta:
mostrar_detalles_planta(p))
                btn_planta.image = imagen_tk
```

```
            btn_planta.grid(row=index, column=0, padx=20, pady=20)

        ttk.Label(scrollable_frame, text=planta['nombre'], background="#D0F0C0", font=("Arial", 14)).grid(row=index, column=1, padx=10)

        # Botón para eliminar planta
        ttk.Button(scrollable_frame, text="Eliminar", command=lambda p=planta: eliminar_planta(p)).grid(row=index, column=2, padx=10)

    ttk.Button(scrollable_frame, text="Registrar Nueva Planta", command=mostrar_formulario_planta).grid(row=len(plantas) + 1, column=0, columnspan=3, pady=20)
    ttk.Button(scrollable_frame, text="Regresar a Inicio de Sesión", command=lambda: [ventana_bienvenida.destroy(), mostrar_login()]).grid(row=len(plantas) + 2, column=0, columnspan=3)

# Función para mostrar el formulario de registro de planta
def mostrar_formulario_planta():
    cerrar_ventanas()

    ventana_formulario = tk.Toplevel(root)
    ventana_formulario.title("Registrar Nueva Planta")
    ventana_formulario.geometry("400x550")
    ventana_formulario.config(bg="#F0F5E6")

    tk.Label(ventana_formulario, text="Nombre de la planta").pack(pady=5)
    nombre_entry = tk.Entry(ventana_formulario)
    nombre_entry.pack(pady=5)

    tk.Label(ventana_formulario, text="Temperatura (°C)").pack(pady=5)
    temp_entry = tk.Entry(ventana_formulario)
    temp_entry.pack(pady=5)

    tk.Label(ventana_formulario, text="pH").pack(pady=5)
    ph_entry = tk.Entry(ventana_formulario)
```

```
    ph_entry.pack(pady=5)

    tk.Label(ventana_formulario, text="Humedad (%)").pack(pady=5)
    humedad_entry = tk.Entry(ventana_formulario)
    humedad_entry.pack(pady=5)

    label_imagen = ttk.Label(ventana_formulario, background="#F0F5E6")
    label_imagen.pack(pady=5)

    ttk.Button(ventana_formulario, text="Cargar Imagen", command=lambda: cargar_imagen(label_imagen)).pack(pady=5)
    ttk.Button(ventana_formulario, text="Registrar", command=lambda: registrar_planta(nombre_entry, temp_entry,
ph_entry, humedad_entry, label_imagen)).pack(pady=10)
    ttk.Button(ventana_formulario, text="Regresar", command=ventana_formulario.destroy).pack(pady=5)

# Función para cargar una imagen
def cargar_imagen(label):
    ruta_imagen = filedialog.askopenfilename(filetypes=[("Image files", "*.jpg;*.jpeg;*.png")])
    if ruta_imagen:
        label.ruta_imagen = ruta_imagen  # Guardamos la ruta en el label
        imagen = Image.open(ruta_imagen)
        imagen_redimensionada = imagen.resize((100, 100))
        imagen_tk = ImageTk.PhotoImage(imagen_redimensionada)
        label.config(image=imagen_tk)
        label.image = imagen_tk  # Para evitar la recolección de basura

# Función para mostrar la pantalla de inicio de sesión
def mostrar_login():
    cerrar_ventanas()

    ventana_login = tk.Toplevel(root)
    ventana_login.title("Inicio de Sesión")
    ventana_login.geometry("300x200")
```

```
    tk.Label(ventana_login, text="Usuario").pack(pady=5)
    usuario_entry = tk.Entry(ventana_login)
    usuario_entry.pack(pady=5)

    tk.Label(ventana_login, text="Contraseña").pack(pady=5)
    contrasena_entry = tk.Entry(ventana_login, show="*")
    contrasena_entry.pack(pady=5)

    ttk.Button(ventana_login, text="Iniciar Sesión", command=lambda: iniciar_sesion(usuario_entry.get(),
contrasena_entry.get(), ventana_login)).pack(pady=10)
    ttk.Button(ventana_login, text="Registrar Nuevo Usuario", command=lambda:
mostrar_registro(ventana_login)).pack(pady=5)

# Función para mostrar el registro de usuario
def mostrar_registro(ventana_login):
    ventana_login.withdraw()  # Ocultar la ventana de inicio de sesión
    ventana_registro = tk.Toplevel(root)
    ventana_registro.title("Registrar Nuevo Usuario")
    ventana_registro.geometry("300x200")

    tk.Label(ventana_registro, text="Usuario").pack(pady=5)
    usuario_entry = tk.Entry(ventana_registro)
    usuario_entry.pack(pady=5)

    tk.Label(ventana_registro, text="Contraseña").pack(pady=5)
    contrasena_entry = tk.Entry(ventana_registro, show="*")
    contrasena_entry.pack(pady=5)

    ttk.Button(ventana_registro, text="Registrar", command=lambda: registrar_usuario(usuario_entry.get(),
contrasena_entry.get(), ventana_registro)).pack(pady=10)
    ttk.Button(ventana_registro, text="Regresar", command=lambda: [ventana_registro.destroy(),
ventana_login.deiconify()]).pack(pady=5)
```

```
# Función para registrar un nuevo usuario
def registrar_usuario(usuario, contrasena, ventana_registro):
    if usuario in usuarios:
        messagebox.showerror("Error", "El usuario ya existe.")
    else:
        guardar_usuario(usuario, contrasena)
        messagebox.showinfo("Éxito", "Usuario registrado exitosamente")
        ventana_registro.destroy()

# Función para iniciar sesión
def iniciar_sesion(usuario, contrasena, ventana_login):
    if usuario in usuarios and usuarios[usuario] == contrasena:
        ventana_login.destroy()
        mostrar_bienvenida()
    else:
        messagebox.showerror("Error", "Usuario o contraseña incorrectos.")

# Función para cerrar ventanas
def cerrar_ventanas():
    for widget in root.winfo_children():
        widget.destroy()

# Crear la ventana principal
root = tk.Tk()
root.title("Gestión de Plantas")
root.geometry("400x300")
root.config(bg="#F0F5E6")

cargar_usuarios()
cargar_plantas()
```

```
# Mostrar la pantalla de inicio de sesión
mostrar_login()

# Iniciar el bucle principal de la interfaz gráfica
root.mainloop()
```

Material

1. Bomba de agua de 600L/h
2. 2 tubos pvc de 4 x 3mts
3. Focos de calor
4. Sensor de humedad
5. “”” “” temperatura
6. “”” “”ph
7. 2 Acrilico de 150cm x 70cm
8. 2 Acrilico de 120cm x 70cm