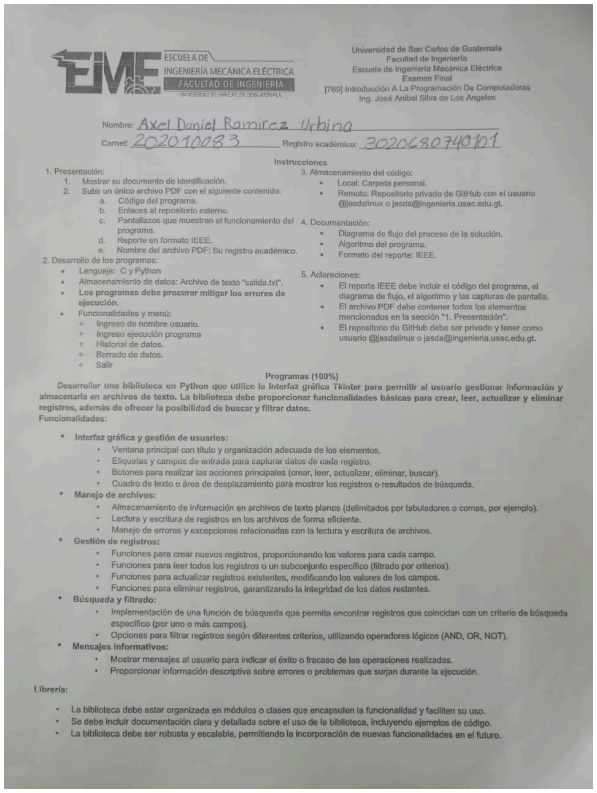
Examen Final

Axel Daniel Ramirez Urbina, 202010083
Joseph Israel Ramirez Urbina, 202108843
Escuela de Mecánica Eléctrica, Facultad de Ingeniería,
Universidad de San Carlos de Guatemala

Documento en GitHub
https://github.com/AxelRamirez12/Final.git

Axel Daniel Ramirez Urbina, 202010083



Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería Mecánica Eléctrica
Examen Final
[769] Introducción A La Programación De Computadoras
Ing. José Aníbal Silva de Los Angeles

Nombre: Axel Daniel Ramirez Urbina
Carnet: 202010083 Registro académico: 3020680740101

Instrucciones

1. Presentación:
   1. Mostrar su documento de identificación.
   2. Subir un único archivo PDF con el siguiente contenido:
      a. Código del programa.
      b. Enlaces al repositorio externo.
      c. Pantallazos que muestren el funcionamiento del programa.
      d. Reporte en formato IEEE.
      e. Nombre del archivo PDF: Su registro académico.
2. Desarrollo de los programas:
   - Lenguaje: C y Python
   - Almacenamiento de datos: Archivo de texto "salida.txt".
   - **Los programas debe procurar mitigar los errores de ejecución.**
   - Funcionalidades y menú:
     - Ingreso de nombre usuario.
     - Ingreso ejecución programa
     - Historial de datos.
     - Borrado de datos.
     - Salir
3. Almacenamiento del código:
   - Local: Carpeta personal.
   - Remoto: Repositorio privado de GitHub con el usuario @jasdalinux o jasda@ingenieria.usac.edu.gt.
4. Documentación:
   - Diagrama de flujo del proceso de la solución.
   - Algoritmo del programa.
   - Formato del reporte: IEEE.
5. Aclaraciones:
   - El reporte IEEE debe incluir el código del programa, el diagrama de flujo, el algoritmo y las capturas de pantalla.
   - El archivo PDF debe contener todos los elementos mencionados en la sección "1. Presentación".
   - El repositorio de GitHub debe ser privado y tener como usuario @jasdalinux o jasda@ingenieria.usac.edu.gt.

Programas (100%)

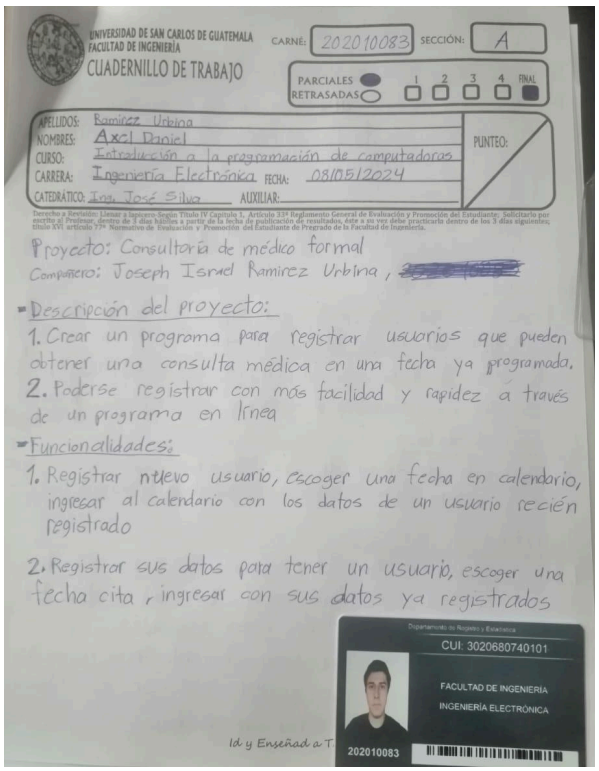
Desarrollar una biblioteca en Python que utilice la interfaz gráfica Tkinter para permitir al usuario gestionar información y almacenarla en archivos de texto. La biblioteca debe proporcionar funcionalidades básicas para crear, leer, actualizar y eliminar registros, además de ofrecer la posibilidad de buscar y filtrar datos.

Funcionalidades:

- **Interfaz gráfica y gestión de usuarios:**
  - Ventana principal con título y organización adecuada de los elementos.
  - Etiquetas y campos de entrada para capturar datos de cada registro.
  - Botones para realizar las acciones principales (crear, leer, actualizar, eliminar, buscar).
  - Cuadro de texto o área de desplazamiento para mostrar los registros o resultados de búsqueda.
- **Manejo de archivos:**
  - Almacenamiento de información en archivos de texto planos (delimitados por tabuladores o comas, por ejemplo).
  - Lectura y escritura de registros en los archivos de forma eficiente.
  - Manejo de errores y excepciones relacionadas con la lectura y escritura de archivos.
- **Gestión de registros:**
  - Funciones para crear nuevos registros, proporcionando los valores para cada campo.
  - Funciones para leer todos los registros o un subconjunto específico (filtrado por criterios).
  - Funciones para actualizar registros existentes, modificando los valores de los campos.
  - Funciones para eliminar registros, garantizando la integridad de los datos restantes.
- **Búsqueda y filtrado:**
  - Implementación de una función de búsqueda que permita encontrar registros que coincidan con un criterio de búsqueda específico (por uno o más campos).
  - Opciones para filtrar registros según diferentes criterios, utilizando operadores lógicos (AND, OR, NOT).
- **Mensajes informativos:**
  - Mostrar mensajes al usuario para indicar el éxito o fracaso de las operaciones realizadas.
  - Proporcionar información descriptiva sobre errores o problemas que surjan durante la ejecución.

Librería:

- **La biblioteca debe estar organizada en módulos o clases que encapsulen la funcionalidad y faciliten su uso.**
- **Se debe incluir documentación clara y detallada sobre el uso de la biblioteca, incluyendo ejemplos de código.**
- **La biblioteca debe ser robusta y escalable, permitiendo la incorporación de nuevas funcionalidades en el futuro.**



UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
FACULTAD DE INGENIERÍA
CUADERNILLO DE TRABAJO

CARNÉ: 202010083 SECCIÓN: A

PARCIALES RETRASADAS 1 2 3 4 FINAL

APELLIDOS: Ramirez Urbina
NOMBRES: Axel Daniel
CURSO: Introducción a la programación de computadoras
CARRERA: Ingeniería Electrónica FECHA: 08/05/2024
CATEDRÁTICO: Ing. José Silva AUXILIAR:
PUNTEO:

Derecho a Revisión: Llenar a lapicero Según Título IV Capítulo I, Artículo 33º Reglamento General de Evaluación y Promoción del Estudiante. Solicitarlo por escrito al Profesor, dentro de 3 días hábiles a partir de la fecha de publicación de resultados. Acta a ser vez debe practicarlo dentro de los 3 días siguientes. Título XVI, artículo 73º Normativo de Evaluación y Promoción del Estudiante de Pregrado de la Facultad de Ingeniería.

Proyecto: Consultaría de médico formal
Compañero: Joseph Israel Ramirez Urbina, [redacted]

- Descripción del proyecto:

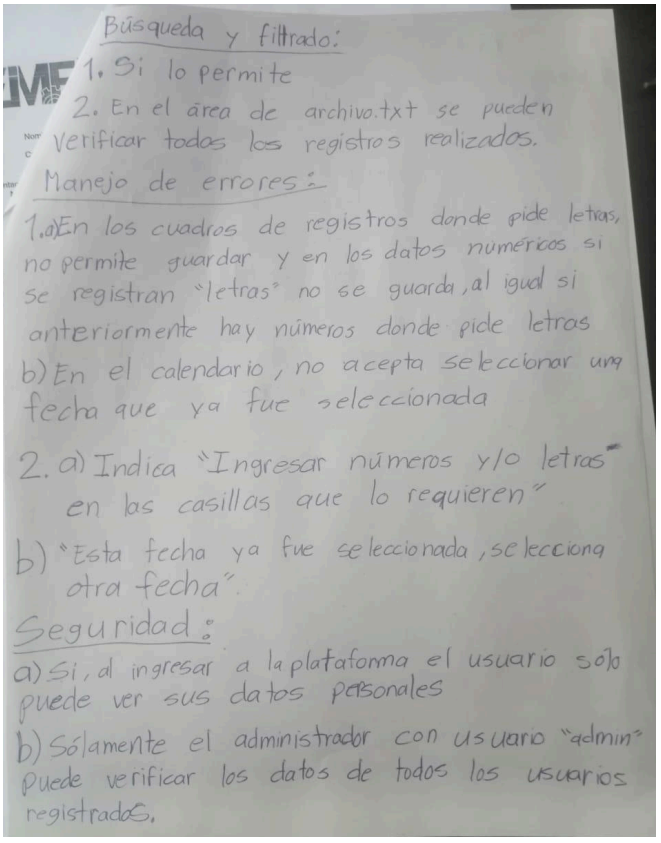
1. Crear un programa para registrar usuarios que pueden obtener una consulta médica en una fecha ya programada.
2. Poderse registrar con más facilidad y rapidez a través de un programa en línea

- Funcionalidades:

1. Registrar nuevo usuario, escoger una fecha en calendario, ingresar al calendario con los datos de un usuario recién registrado
2. Registrar sus datos para tener un usuario, escoger una fecha cita, ingresar con sus datos ya registrados

Id y Enseñad a T

Departamento de Registro y Estadística
CUI: 3020680740101
FACULTAD DE INGENIERÍA
INGENIERÍA ELECTRÓNICA
202010083



Búsqueda y filtrado:

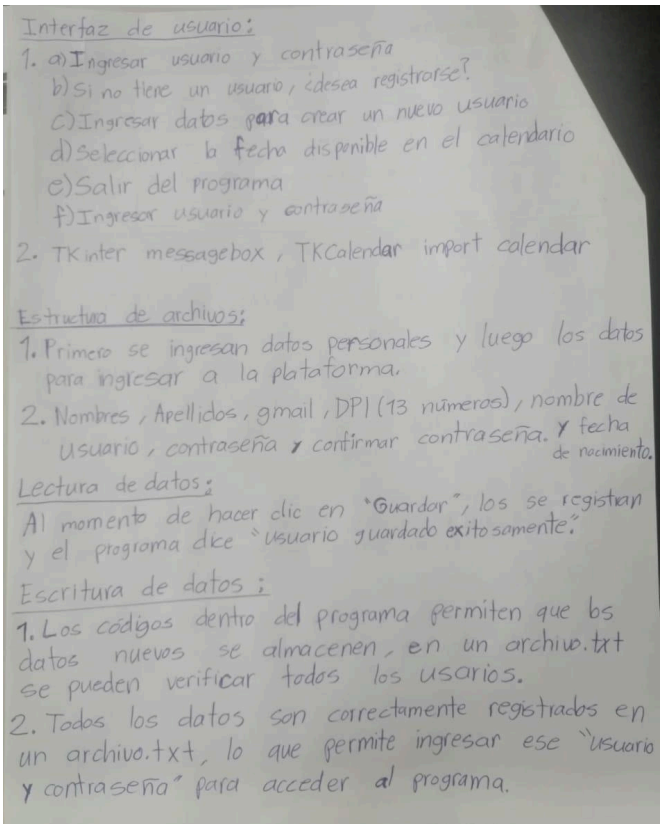
1. Si lo permite
2. En el área de archivo.txt se pueden verificar todos los registros realizados.

Manejo de errores:

1. a) En los cuadros de registros donde pide letras, no permite guardar y en los datos numéricos si se registran "letras" no se guarda, al igual si anteriormente hay números donde pide letras
   b) En el calendario, no acepta seleccionar una fecha que ya fue seleccionada
2. a) Indica "Ingresar números y/o letras" en las casillas que lo requieren"
   b) "Esta fecha ya fue seleccionada, selecciona otra fecha".

Seguridad:

a) Si, al ingresar a la plataforma el usuario sólo puede ver sus datos personales
b) Sólamente el administrador con usuario "admin" puede verificar los datos de todos los usuarios registrados.



Interfaz de usuario:

1. a) Ingresar usuario y contraseña
   b) Si no tiene un usuario, ¿desea registrarse?
   c) Ingresar datos para crear un nuevo usuario
   d) Seleccionar la fecha disponible en el calendario
   e) Salir del programa
   f) Ingresar usuario y contraseña
2. TKinter messagebox, TKCalendar import calendar

Estructura de archivos:

1. Primero se ingresan datos personales y luego los datos para ingresar a la plataforma.
2. Nombres, Apellidos, gmail, DPI (13 números), nombre de usuario, contraseña y confirmar contraseña. y fecha de nacimiento.

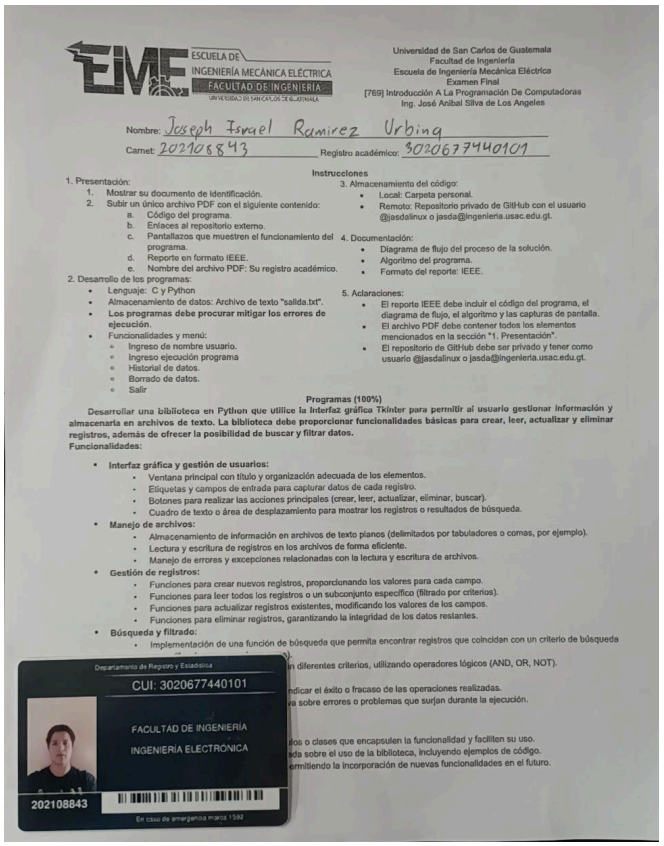
Lectura de datos:

Al momento de hacer clic en "Guardar", los se registran y el progroma dice "usuario guardado exitosamente."

Escritura de datos:

1. Los códigos dentro del programa permiten que los datos nuevos se almacenen, en un archivo.txt se pueden verificar todos los usarios.
2. Todos los datos son correctamente registrados en un archivo.txt, lo que permite ingresar ese "usuario y contraseña" para acceder al programa.

Joseph Israel Ramirez Urbina, 202108843



ESCUELA DE INGENIERÍA MECÁNICA ELÉCTRICA
FACULTAD DE INGENIERÍA
UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería Mecánica Eléctrica
Examen Final
[769] Introducción A La Programación De Computadoras
Ing: José Anibal Silva de Los Angeles

Nombre: Joseph Israel Ramirez Urbina
Carnet: 202108843 Registro académico: 3020677440101

Instrucciones

1. Presentación:
   1. Mostrar su documento de identificación.
   2. Subir un único archivo PDF con el siguiente contenido:
      a. Código del programa.
      b. Enlaces al repositorio externo.
      c. Pantallazos que muestren el funcionamiento del programa.
      d. Reporte en formato IEEE.
      e. Nombre del archivo PDF: Su registro académico.
2. Desarrollo de los programas:
   - Lenguaje: C y Python
   - Almacenamiento de datos: Archivo de texto "salida.txt".
   - **Los programas debe procurar mitigar los errores de ejecución.**
   - Funcionalidades y menú:
     - Ingreso de nombre usuario.
     - Ingreso ejecución programa
     - Historial de datos.
     - Borrado de datos.
     - Salir
3. Almacenamiento del código:
   - Local: Carpeta personal.
   - Remoto: Repositorio privado de GitHub con el usuario @jasdalinux o jasda@ingenieria.usac.edu.gt.
4. Documentación:
   - Diagrama de flujo del proceso de la solución.
   - Algoritmo del programa.
   - Formato del reporte: IEEE.
5. Aclaraciones:
   - El reporte IEEE debe incluir el código del programa, el diagrama de flujo, el algoritmo y las capturas de pantalla.
   - El archivo PDF debe contener todos los elementos mencionados en la sección "1. Presentación".
   - El repositorio de GitHub debe ser privado y tener como usuario @jasdalinux o jasda@ingenieria.usac.edu.gt.

Programas (100%)

Desarrollar una biblioteca en Python que utilice la interfaz gráfica Tkinter para permitir al usuario gestionar información y almacenarla en archivos de texto. La biblioteca debe proporcionar funcionalidades básicas para crear, leer, actualizar y eliminar registros, además de ofrecer la posibilidad de buscar y filtrar datos.

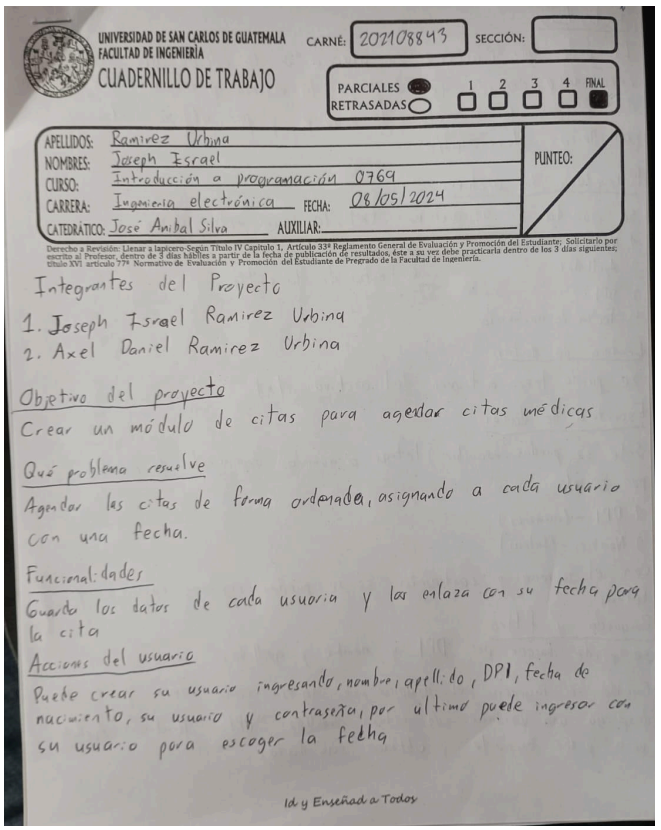
Funcionalidades:

- **Interfaz gráfica y gestión de usuarios:**
  - Ventana principal con título y organización adecuada de los elementos.
  - Etiquetas y campos de entrada para capturar datos de cada registro.
  - Botones para realizar las acciones principales (crear, leer, actualizar, eliminar, buscar).
  - Cuadro de texto o área de desplazamiento para mostrar los registros o resultados de búsqueda.
- **Manejo de archivos:**
  - Almacenamiento de información en archivos de texto planos (delimitados por tabuladores o comas, por ejemplo).
  - Lectura y escritura de registros en los archivos de forma eficiente.
  - Manejo de errores y excepciones relacionadas con la lectura y escritura de archivos.
- **Gestión de registros:**
  - Funciones para crear nuevos registros, proporcionando los valores para cada campo.
  - Funciones para leer todos los registros o un subconjunto específico (filtrado por criterios).
  - Funciones para actualizar registros existentes, modificando los valores de los campos.
  - Funciones para eliminar registros, garantizando la integridad de los datos restantes.
- **Búsqueda y filtrado:**
  - Implementación de una función de búsqueda que permita encontrar registros que coincidan con un criterio de búsqueda
  - en diferentes criterios, utilizando operadores lógicos (AND, OR, NOT).

Departamento de Registro y Estadística
CUI: 3020677440101
FACULTAD DE INGENIERIA
INGENIERIA ELECTRONICA
202108843
En caso de emergencia marcar 1582

ndicar el éxito o fracaso de las operaciones realizadas.
re sobre errores o problemas que surjan durante la ejecución.

los o clases que encapsulen la funcionalidad y faciliten su uso.
da sobre el uso de la biblioteca, incluyendo ejemplos de código.
emitiendo la incorporación de nuevas funcionalidades en el futuro.



UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
FACULTAD DE INGENIERIA
CUADERNILLO DE TRABAJO

CARNÉ: 202108843 SECCIÓN:

PARCIALES 1 2 3 4 FINAL
RETRASADAS

APELLIDOS: Ramirez Urbina
NOMBRES: Joseph Israel
CURSO: Introducción a programación 0769
CARRERA: Ingeniería electrónica FECHA: 08/05/2024
CATEDRÁTICO: José Anibal Silva AUXILIAR:
PUNTEO:

Derecho a Revisión: Llenar a lapicero-Según Título IV Capítulo 1, Artículo 33º Reglamento General de Evaluación y Promoción del Estudiante; Solicitarlo por escrito al Profesor, dentro de 3 días hábiles a partir de la fecha de publicación de resultados, éste a su vez debe practicarla dentro de los 3 días siguientes; título XVI artículo 77º Normativo de Evaluación y Promoción del Estudiante de Pregrado de la Facultad de Ingeniería.

Integrantes del Proyecto

1. Joseph Israel Ramirez Urbina
2. Axel Daniel Ramirez Urbina

Objetivo del proyecto

Crear un módulo de citas para agendar citas médicas

Qué problema resuelve

Agendar las citas de forma ordenada, asignando a cada usuario con una fecha.

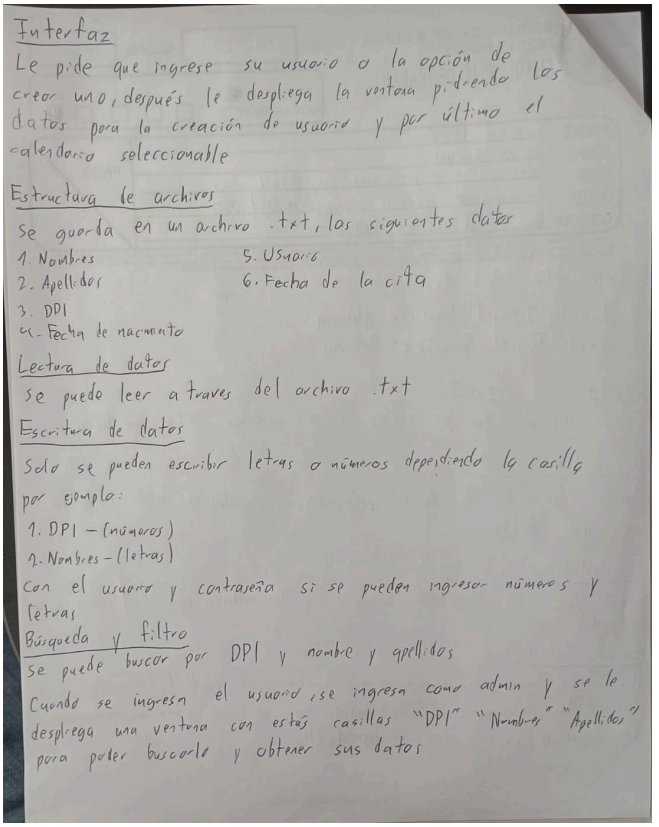
Funcionalidades

Guarda los datos de cada usuario y las enlaza con su fecha para la cita

Acciones del usuario

Puede crear su usuario ingresando, nombre, apellido, DPI, fecha de nacimiento, su usuario y contraseña, por ultimo puede ingresar con su usuario para escoger la fecha

Id y Enseñad a Todos



Interfaz

Le pide que ingrese su usuario o la opción de crear uno, después le despliega la ventana pidiendo los datos para la creación de usuario y por último el calendario seleccionable

Estructura de archivos

Se guarda en un archivo .txt, los siguientes datos

1. Nombres
2. Apellidos
3. DPI
4. Fecha de nacimiento
5. Usuario
6. Fecha de la cita

Lectura de datos

se puede leer a traves del archivo .txt

Escritura de datos

Solo se pueden escribir letras o números dependiendo la casilla por ejemplo:

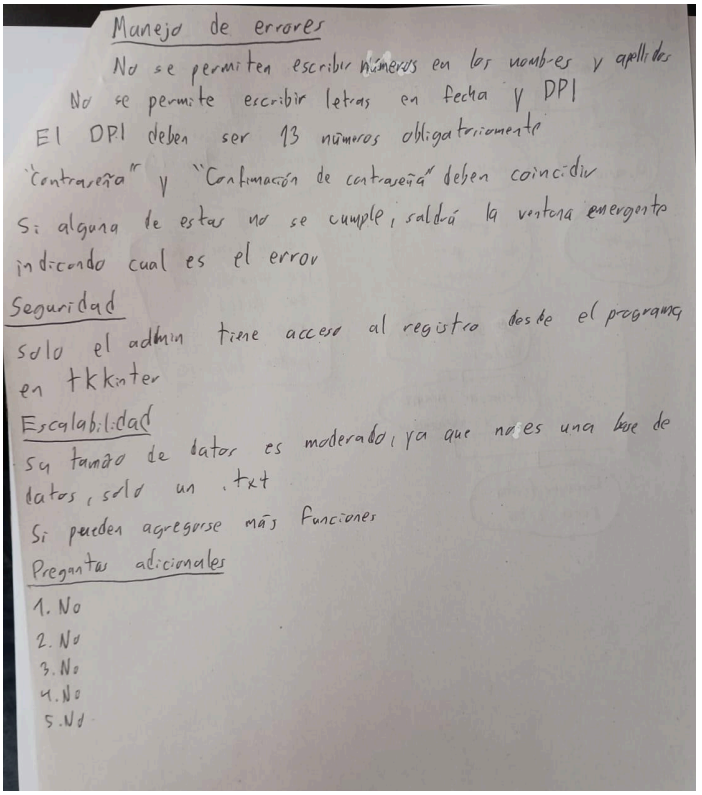
1. DPI – (números)
2. Nombres – (letras)

Con el usuario y contraseña si se pueden ingresar números y letras

Búsqueda y filtro

se puede buscar por DPI y nombre y apellidos

Cuando se ingresa el usuario, se ingresa como admin y se le despliega una ventana con estas casillas "DPI" "Nombres" "Apellidos" para poder buscarlo y obtener sus datos



Manejo de errores

No se permiten escribir números en los nombres y apellidos
No se permite escribir letras en fecha y DPI
El DPI deben ser 13 números obligatoriamente
"Contraseña" y "Confirmación de contraseña" deben coincidir
Si alguna de estas no se cumple, saldrá la ventana emergente indicando cual es el error

Seguridad

solo el admin tiene acceso al registro desde el programa en tkkinter

Escalabilidad

Su tamaño de datos es moderado, ya que no es una base de datos, solo un .txt
Si pueden agregarse más funciones

Preguntas adicionales

1. No
2. No
3. No
4. No
5. No

A. *Resumen*

El programa diseñado en Python para el "Módulo de citas y suministros" de una clínica médica ofrece funcionalidades clave para gestionar las citas de los usuarios de manera eficiente. Permite a los usuarios registrarse proporcionando información personal, incluidos nombres, apellidos, DPI, correo electrónico, fecha de nacimiento, nombre de usuario y contraseña. Además, deben confirmar su contraseña para garantizar la precisión de los datos.

Una vez registrados, los usuarios tienen acceso a un calendario interactivo implementado con la biblioteca Tkcalendar. Este calendario les permite seleccionar una fecha disponible para programar su cita en la clínica del médico. La interfaz gráfica intuitiva facilita la navegación y la selección de fechas adecuadas para la cita.

El sistema también incluye un nivel de administrador, identificado como "Admin", que tiene privilegios especiales. El administrador puede acceder a los datos de todos los usuarios registrados y tiene la capacidad de eliminar usuarios según sea necesario. Esto asegura que el administrador tenga control total sobre la gestión de usuarios y citas en el sistema.

En resumen, el programa proporciona una solución integral para la gestión de citas en una clínica médica, permitiendo a los usuarios registrarse fácilmente y programar citas de manera eficiente, mientras que el administrador tiene control total sobre la gestión de usuarios y datos.

B. *Código*

Citas.py > ClinicaMedicaApp

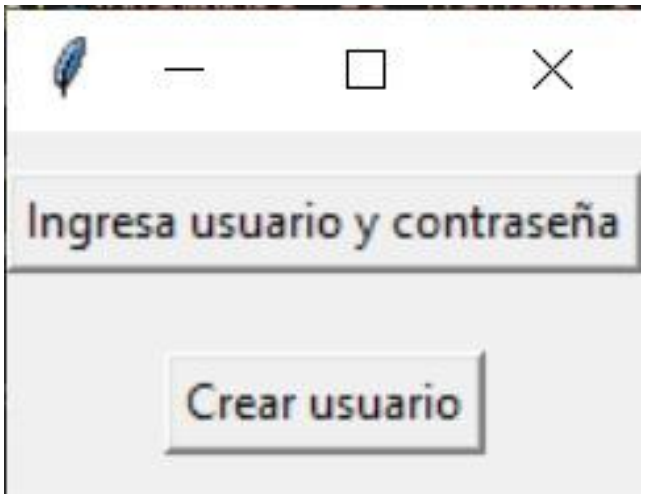
```python
import tkinter as tk
from tkinter import messagebox
from tkinter import simpledialog
from tkcalendar import Calendar
import re

class ClinicaMedicaApp:
    def __init__(self, root):
        self.root = root
        self.root.title("Inicio")

        # Botón para iniciar sesión
        tk.Button(root, text="Ingresa usuario y contraseña", command=self.mostrar_ventana_login).pack(pady=10)

        # Botón para crear un nuevo usuario
        tk.Button(root, text="Crear usuario", command=self.mostrar_modulo_citas).pack(pady=10)

    def mostrar_ventana_login(self):
        usuario = simpledialog.askstring("Inicio de Sesión", "Ingrese su usuario:")
        contrasena = simpledialog.askstring("Inicio de Sesión", "Ingrese su contraseña:", show="*")

        # Aquí podrías agregar la lógica de verificación de usuario y contraseña
        if self.verificar_credenciales(usuario, contrasena):
            if usuario == "admin":
                self.mostrar_buscar_registros()
            else:
                self.mostrar_calendario()
        else:
            respuesta = messagebox.askyesno("Usuario no registrado", "¿Aún no tienes una cuenta? ¿Deseas registrarte?
            if respuesta:
                self.mostrar_modulo_citas()

    def verificar_credenciales(self, usuario, contrasena):
        # Verificar las credenciales en el archivo de usuarios registrados
        if usuario == "admin" and contrasena == "admin123":
            return True
        else:
            with open("datos_pacientes.txt", "r") as file:
                lines = file.readlines()
                for i in range(0, len(lines), 9):
                    saved_user = lines[i + 5].split(":")[1].strip()
                    if saved_user == usuario:
                        saved_pass = lines[i + 7].split(":")[1].strip()
                        if saved_pass == contrasena:
                            return True
        return False

    def mostrar_modulo_citas(self):
        root = tk.Toplevel()
        app = ClinicaMedicaNuevoUsuario(root)

    def mostrar_calendario(self):
        top = tk.Toplevel(self.root)
        top.title("Seleccione una fecha")

        cal = Calendar(top, selectmode="day", year=2024, month=5, day=7)
        cal.pack(padx=20, pady=20)

        tk.Button(top, text="Seleccionar Fecha", command=lambda: self.verificar_disponibilidad(cal.get_date())).pack(

    def verificar_disponibilidad(self, fecha):
        # Aquí agregarías la lógica para verificar si la fecha está disponible
        # Por ahora, simplemente mostraremos un mensaje de éxito
        messagebox.showinfo("Fecha Disponible", f"La fecha {fecha} está disponible para agendar cita.")

    def mostrar_buscar_registros(self):
        root = tk.Toplevel(self.root)
        root.title("Buscar/Eliminar Registros")

        tk.Label(root, text="Buscar/Eliminar por Nombres:").grid(row=0, column=0, sticky="e")
        self.entry_nombres = tk.Entry(root)
        self.entry_nombres.grid(row=0, column=1)

        tk.Label(root, text="Buscar/Eliminar por Apellidos:").grid(row=1, column=0, sticky="e")
        self.entry_apellidos = tk.Entry(root)
        self.entry_apellidos.grid(row=1, column=1)

        tk.Label(root, text="Buscar/Eliminar por DPI:").grid(row=2, column=0, sticky="e")
        self.entry_dpi = tk.Entry(root)
        self.entry_dpi.grid(row=2, column=1)

        tk.Button(root, text="Buscar", command=self.buscar_registros).grid(row=3, columnspan=2, pady=5)

    def buscar_registros(self):
        # Lógica para buscar los registros según los filtros de nombres, apellidos y DPI
        nombres = self.entry_nombres.get()
        apellidos = self.entry_apellidos.get()
        dpi = self.entry_dpi.get()

        # Verificar si los campos están vacíos
        if not nombres and not apellidos and not dpi:
            messagebox.showwarning("Campos Vacíos", "Por favor, ingrese al menos un campo para buscar.")
            return

        with open("datos_pacientes.txt", "r") as file:
            lines = file.readlines()
            found = False
            for i in range(0, len(lines), 9):
                if i + 2 < len(lines):
                    saved_nombres = lines[i].split(":")[1].strip()
                    saved_apellidos = lines[i + 1].split(":")[1].strip()
                    saved_dpi = lines[i + 2].split(":")[1].strip()

                    if nombres.lower() in saved_nombres.lower() and apellidos.lower() in saved_apellidos.lower() and
                        found = True
                        datos_usuario = ""
                        for j in range(i, i + 9):
                            datos_usuario += lines[j]
                        messagebox.showinfo("Datos del Usuario", datos_usuario)
                        # Agregar la opción de eliminar el usuario
                        respuesta = messagebox.askyesno("Eliminar Usuario", "¿Desea eliminar este usuario?")
                        if respuesta:
                            self.eliminar_usuario(i)
            if not found:
                messagebox.showinfo("Usuario no encontrado", "No se encontraron usuarios con los filtros especificado

    def eliminar_usuario(self, index):
        # Eliminar el usuario del archivo
        with open("datos_pacientes.txt", "r") as file:
            lines = file.readlines()
        with open("datos_pacientes.txt", "w") as file:
            for i, line in enumerate(lines):
                if i < index or i >= index + 9:
                    file.write(line)
class ClinicaMedicaNuevoUsuario:
    def __init__(self, root):
        self.root = root
        self.root.title("Módulo de Citas - Nuevo Usuario")

        # Variables para almacenar datos
        self.nombres = tk.StringVar()
        self.apellidos = tk.StringVar()
        self.dpi = tk.StringVar()
        self.fecha_nacimiento = tk.StringVar()
        self.telefono = tk.StringVar()
        self.usuario = tk.StringVar()
        self.correo_electronico = tk.StringVar()
        self.contrasena = tk.StringVar()
        self.confirmar_contrasena = tk.StringVar()

        # Crear etiquetas y campos de entrada
        tk.Label(root, text="Nombres:").grid(row=0, column=0, sticky="e")
        tk.Entry(root, textvariable=self.nombres).grid(row=0, column=1)

        tk.Label(root, text="Apellidos:").grid(row=1, column=0, sticky="e")
        tk.Entry(root, textvariable=self.apellidos).grid(row=1, column=1)

        tk.Label(root, text="Número de DPI (13 números):").grid(row=2, column=0, sticky="e")
        tk.Entry(root, textvariable=self.dpi).grid(row=2, column=1)

        tk.Label(root, text="Fecha de nacimiento:").grid(row=3, column=0, sticky="e")
        tk.Entry(root, textvariable=self.fecha_nacimiento).grid(row=3, column=1)

        tk.Label(root, text="Teléfono:").grid(row=4, column=0, sticky="e")
        tk.Entry(root, textvariable=self.telefono).grid(row=4, column=1)

        tk.Label(root, text="Nombre de usuario:").grid(row=5, column=0, sticky="e")
        tk.Entry(root, textvariable=self.usuario).grid(row=5, column=1)

        tk.Label(root, text="Correo electrónico:").grid(row=6, column=0, sticky="e")
        tk.Entry(root, textvariable=self.correo_electronico).grid(row=6, column=1)

        tk.Label(root, text="Contraseña:").grid(row=7, column=0, sticky="e")
```

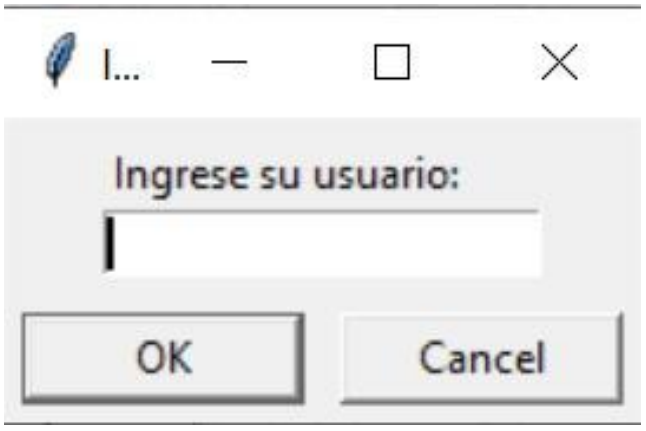
```
        tk.Entry(root, textvariable=self.contrasena, show="*").grid(row=7, column=1)

        tk.Label(root, text="Confirmar contraseña:").grid(row=8, column=0, sticky="e")
        tk.Entry(root, textvariable=self.confirmar_contrasena, show="*").grid(row=8, column=1)

        # Botón para guardar y mostrar datos
        tk.Button(root, text="Guardar", command=self.guardar_datos).grid(row=9, column=1, pady=10)

    def guardar_datos(self):
        (variable) apellidos: str ados por el usuario
        apellidos = self.apellidos.get()
        dpi = self.dpi.get()
        fecha_nacimiento = self.fecha_nacimiento.get()
        telefono = self.telefono.get()
        usuario = self.usuario.get()
        correo_electronico = self.correo_electronico.get()
        contrasena = self.contrasena.get()
        confirmar_contrasena = self.confirmar_contrasena.get()

        # Validar que en las casillas de "Nombres" y "Apellidos" solo se puedan escribir letras
        if not re.match("^[A-Za-z ]+$", nombres) or not re.match("^[A-Za-z ]+$", apellidos):
            messagebox.showerror("Error", "Por favor, ingrese solo letras en los campos de nombres y apellidos.")
            return

        # Validar que en las casillas de "DPI" y  no se puedan escribir letras
```

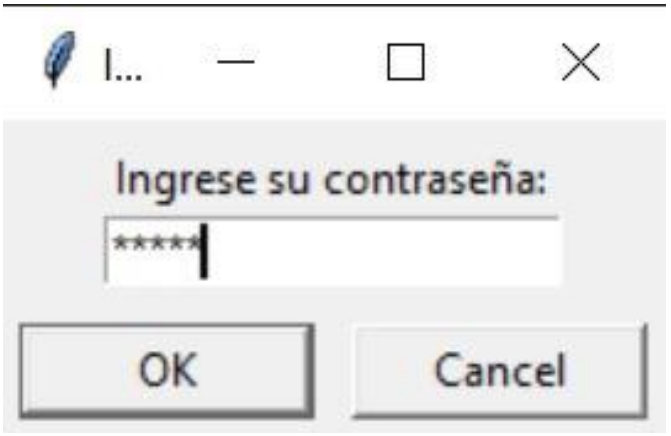
```
        if not dpi.isdigit() :
            messagebox.showerror("Error", "Por favor, ingrese solo números en los campos de DPI ")
            return

        # Verificar que la contraseña coincida con la confirmación
        if contrasena != confirmar_contrasena:
            messagebox.showerror("Error", "Las contraseñas no coinciden.")
            return

        # Verificar si el DPI tiene 13 números
        if len(dpi) != 13:
            messagebox.showerror("Error", "El número de DPI debe contener 13 números.")
            return

        # Verificar que cada usuario sea único
        with open("datos_pacientes.txt", "r") as file:
            lines = file.readlines()
            for i in range(0, len(lines), 9):
                saved_user = lines[i + 5].split(":")[1].strip()
                if saved_user == usuario:
                    messagebox.showerror("Error", "El nombre de usuario ya está en uso.")
                    return

        # Guardar los datos en un archivo de texto
        with open("datos_pacientes.txt", "a") as file:
            file.write(f"Nombres: {nombres}\n")
```

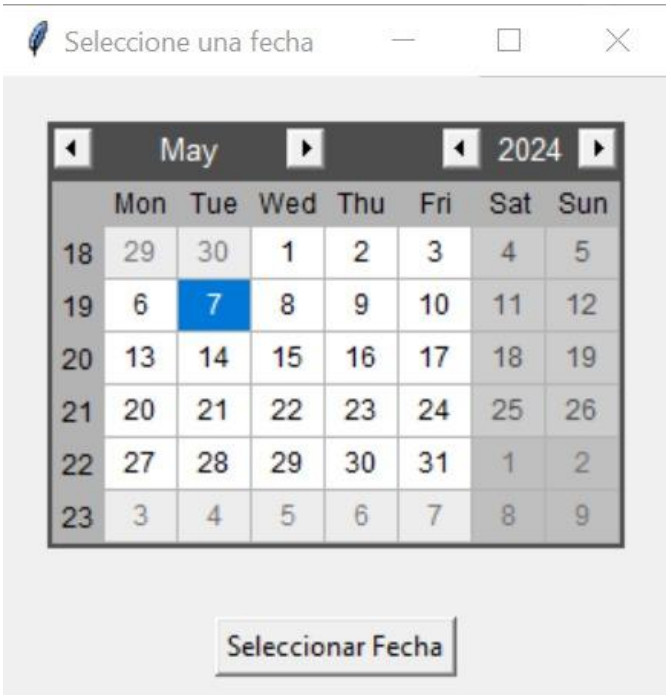
```
            file.write(f"Apellidos: {apellidos}\n")
            file.write(f"DPI: {dpi}\n")
            file.write(f"Fecha de nacimiento: {fecha_nacimiento}\n")
            file.write(f"Teléfono: {telefono}\n")
            file.write(f"Nombre de usuario: {usuario}\n")
            file.write(f"Correo electrónico: {correo_electronico}\n")
            file.write(f"Contraseña: {contrasena}\n")  # Guardar la contraseña en el archivo
            file.write("-------------------------------------------------\n")

        # Mostrar mensaje de éxito
        messagebox.showinfo("Datos Guardados", "Los datos han sido guardados exitosamente.")

        # Cerrar la ventana después de guardar los datos
        self.root.destroy()

# Crear la ventana principal
root = tk.Tk()
app = ClinicaMedicaApp(root)
root.mainloop()
```

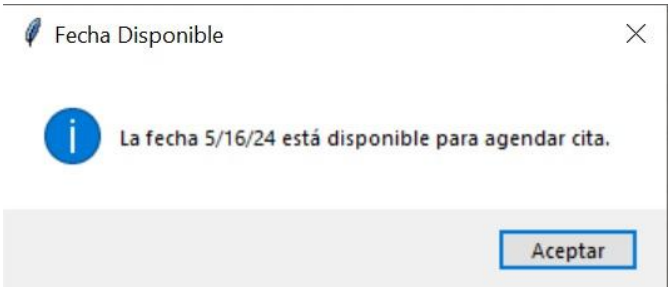
C. Resultados

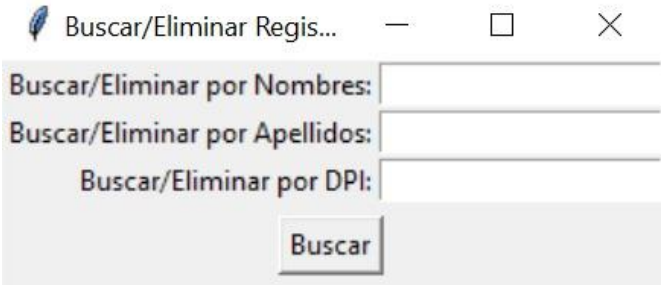


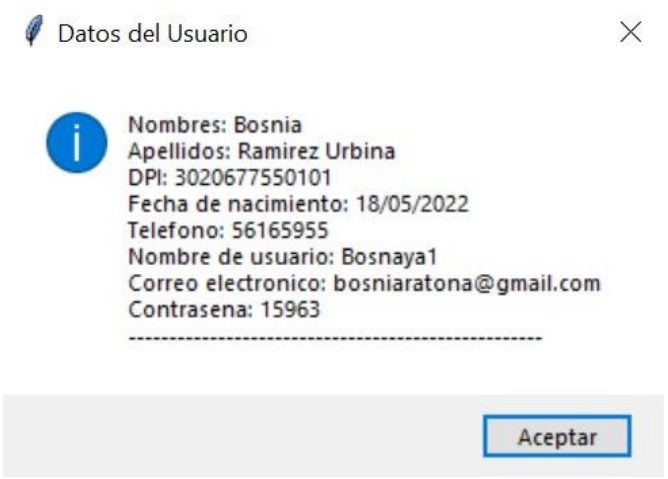













```
datos_pacientes.txt
1   Nombres: Bosnia
2   Apellidos: Ramirez Urbina
3   DPI: 3020677550101
4   Fecha de nacimiento: 18/05/2022
5   Telefono: 56165955
6   Nombre de usuario: Bosnaya1
7   Correo electronico: bosniaratona@gmail.com
8   Contrasena: 15963
9   --------------------------------------------------
10  Nombres: Joseph Israel
11  Apellidos: Ramirez Urbina
12  DPI: 3020677440101
13  Fecha de nacimiento: 09/12/2002
14  Telefono: 56165955
15  Nombre de usuario: Israel777
16  Correo electronico: israelramirezu7@gmail.com
17  Contrasena: 181222
18  --------------------------------------------------
19
```