Manual de usuario – UNAL Study Assistant

Elaborado por:

Bryan Felipe Jaime Diaz

Miguel Angel Galindo Rubio

Juan David Murillo Jimenez

Presentado a

Néstor German Bolívar Pulgarin

Asignatura:

Programación De Computadores

Universidad Nacional de Colombia

Bogotá, DC

2025

1. Proyecto de Curso

Para este curso se planteó realizar una app para escritorio, que sea útil para el usuario, principalmente estudiantes de ingeniería de primeros semestres que necesiten un poco de ayuda para gestionar su vida Académica, es por eso que se realiza esta app llamada **Gestor Académico Estuplan**. Donde a continuación se explicará cómo puede utilizar esta aplicación paso por paso.

2. Introducción

Este manual de usuario describe el funcionamiento de **Gestor Académico** una aplicación de escritorio desarrollada en Python que actúa como un **gestor académico personal**.

La aplicación permite al usuario:

- Consultar sus **cursos, tareas y anuncios** desde **Google Classroom**.
- Visualizar y gestionar sus **eventos académicos** en mediante la api de **Google Calendar**.
- Registrar y consultar **fechas importantes** relacionadas con su vida académica.
- Centralizar en una sola interfaz sus actividades académicas diarias.

La aplicación está orientada principalmente a:

- Estudiantes que desean organizar su carga académica.
- Personas que utilicen Google Classroom y Google Calendar como herramientas centrales de estudio.

Este documento ofrece una descripción detallada de los requisitos, el proceso de instalación y el uso de cada sección de la herramienta.

---

3. Requisitos del sistema

3.1 Requisitos de hardware

- Almacenamiento: al menos 500 MB libres para el proyecto, librerías y archivos generados (tokens, configuraciones, etc.).
- Conexión a Internet: necesaria para autenticarse y utilizar Google Classroom y Google Calendar.

3.2 Requisitos de software

- Sistema operativo:
    - Windows 10 / Windows 11 (recomendado).
    - (Opcional) Linux o macOS con Python 3 instalado.
- Python:
    - Python 3.10 o superior instalado en el sistema.
- Navegador web:
    - Navegador moderno (Chrome, Edge, Firefox, etc.) para el proceso de autenticación con Google.

---

4. Instalación del entorno

A continuación, se describen los pasos para instalar y ejecutar la aplicación en un entorno local utilizando Python.

4.1 Descarga del proyecto

1. Descargue o copie el proyecto en una carpeta de su preferencia, por ejemplo:
    - `C:\Users\[[USUARIO]]\Documents\[[Gestor Academico]]` en Windows.
2. Asegúrese de que los archivos principales estén presentes, entre ellos:
    - `inicio_app.py`
    - `Login.py`
    - `google_calendar.py`
    - `google_classroom.py`
    - otros archivos auxiliares, imágenes, etc.

4.2 Instalación de Python

1. Descargue Python desde la página oficial:

   https://www.python.org/downloads/

2. Durante la instalación en Windows, marque la opción:
   - **"Add Python to PATH" (Agregar Python al PATH)**.
3. Finalice la instalación.

Para verificar la instalación:

- Abra **Símbolo del sistema** o **PowerShell** y ejecute:

```
python --version
```

Debería mostrar una versión igual o superior a 3.10.

---

5. Creación de entorno virtual e instalación de dependencias

Es recomendable usar un entorno virtual para aislar las dependencias del proyecto.

5.1 Creación del entorno virtual (Windows)

1. Abra **PowerShell** o **CMD** en la carpeta del proyecto `[Gestor Academico]`.
2. Ejecute:

```
python -m venv venv
```

3. Active el entorno virtual:

```
.\venv\Scripts\Activate.ps1
```

Si tiene problemas de ejecución de scripts en PowerShell, puede que necesite modificar la política de ejecución.

5.2 Instalación de librerías necesarias (pip)

Con el entorno virtual activado, instale las librerías utilizadas por el proyecto.

- Para ejecutar la app con los avances del classroom es necesario instalar el pillow con este comando:
  pip install pillow
- Otro comando que vas a necesitar es para instalar el matplotlib
  pip install matplotlib

- Para usar el inicio de sesión de google se necesita instalar lo siguiente:
  pip install google-auth google-auth-oauthlib requests pyrebase4 pillow
  Instalar tambien setuptools python.exe -m pip install setuptools
- Instalar estas dependencias:
  pip install pdf2image
- Toca instalar customtkinter
  py -3.14 -m pip install customtkinter
- tocar instalar firebaseadmin
  py -3.14 -m pip install firebase-admin

---

6. Librerías utilizadas y su función

A continuación se listan las librerías principales utilizadas en la aplicación y su propósito:

6.1 Librerías estándar de Python

- **tkinter** y **ttk**
  Permiten crear la interfaz gráfica de usuario: ventanas, botones, etiquetas, menús, etc.

- **datetime** y **calendar**
  Se utilizan para manejar fechas, horas y cálculos relacionados con eventos y calendario.

- **os**, **os.path**
  Manejo de rutas de archivos, comprobación de existencia de archivos, etc.

- **subprocess**
  Permite ejecutar otros programas (en caso de uso puntual para abrir archivos o procesos

externos).

- **math**
  Funciones matemáticas generales (si se requieren cálculos específicos).

- **webbrowser**
  Abre enlaces en el navegador web predeterminado (por ejemplo, para abrir páginas de Google Classroom o ayuda externa).

6.2 Librerías externas (instaladas con pip)

- **Pillow (PIL)**
  Manejo de imágenes (por ejemplo, iconos, logos de la app, etc.).

- **google-auth**, **google-auth-oauthlib**, **google-api-python-client**
  Conjunto de librerías para autenticarse con Google y consumir APIs de Google.

  - `google_auth_oauthlib.flow.InstalledAppFlow` se utiliza para el flujo OAuth 2.0 (ventana de login de Google).

  - `googleapiclient.discovery.build` para crear servicios de API (Google Calendar, Google Classroom).

---

7. Configuración de credenciales de Google

Para permitir que la aplicación acceda a Google Calendar y Google Classroom, es necesario contar con un archivo de credenciales y autorizar la aplicación en la cuenta de Google del usuario.

7.1 Obtención del archivo `credentials.json`

1. Ingrese a Google Cloud Console:

   https://console.cloud.google.com/

2. Cree un proyecto (si aún no tiene uno) o utilice uno existente.

3. Habilite las APIs necesarias:

   - **Google Calendar API**

- **Google Classroom API**

4. En la sección de **Credenciales**, cree un **ID de cliente OAuth 2.0** de tipo **Aplicación de escritorio**.

5. Descargue el archivo de credenciales (`credentials.json`).

6. Coloque el archivo `credentials.json` en la carpeta principal del proyecto o en la ruta esperada por tu código (ejemplo: mismo directorio donde se encuentra `inicio_app.py`).

7.2 Archivos `token.json`

- La primera vez que se ejecute la aplicación y se intente acceder a Google, se abrirá una ventana en el navegador pidiendo permiso de acceso.

- Una vez concedido el permiso, se generará un archivo `token.json` que contiene el token de acceso.

- Este archivo permite que no sea necesario iniciar sesión cada vez que se usa la aplicación (salvo que el token expire o se revoque).

Nota: No comparta `credentials.json` ni `token.json` con otras personas. Son archivos sensibles.

---

8. Estructura de archivos del proyecto

A continuación se muestra una estructura típica del proyecto:

```
[Gestor Académico]/
├─ inicio_app.py
├─ Login.py
├─ google_calendar.py
├─ google_classroom.py
├─ credentials.json         # (debe ser agregado por el usuario)
├─ token.json               # (se genera automáticamente luego de la
autenticación)
├─ /imágenes       # imágenes, iconos, etc.
└─ /venv        # entorno virtual
```

Archivos clave:

- `Login.py`: define la ventana de inicio de sesión y el manejo básico de usuarios.
- `inicio_app.py`: es el archivo principal que lanza la aplicación y controla la ventana principal y el menú.
- `google_calendar.py`: contiene las funciones para comunicarse con Google Calendar (listar, crear, eliminar eventos).
- `google_classroom.py`: contiene las funciones para autenticarse y obtener cursos, tareas y anuncios de Google Classroom.

---

9. Ejecución de la aplicación

Una vez configurado el entorno virtual e instaladas las dependencias:

1. Asegúrese de que el entorno virtual está activado.
2. En la carpeta del proyecto, ejecute:

```
Login.py
```

3. Debería aparecer la ventana de **Login**.

Si esto ocurre, la aplicación está funcionando correctamente.

---

10. Guía de uso de la aplicación

En esta sección se explica el flujo básico de uso de **Gestor Academico** y qué hace cada pantalla principal.

10.1 Pantalla de Login

Al iniciar el programa, se abre la ventana de **Login** definida en `Login.py`.

Elementos típicos de la ventana:

- **Campo Usuario:** para escribir el nombre de usuario registrado.

- **Campo Contraseña:** para escribir la contraseña del usuario.
- **Botón "Iniciar sesión":** valida los datos en firebase
- **Botón "Registrarse:** permite crear un nuevo usuario en el sistema.

Flujo de uso:

1. Ingrese su nombre de usuario y contraseña.
2. Presione el botón **"Iniciar sesión"**.
3. Si las credenciales son correctas:
    - Se cierra la ventana de Login.
    - Se abre la ventana principal del gestor académico.
4. Si las credenciales son incorrectas:
    - Se mostrará un mensaje de error (por ejemplo, “Usuario o contraseña incorrectos”).
    - Intente nuevamente o use la opción de registro (si está implementada).

10.2 Ventana principal (Menú principal)

Tras iniciar sesión, se muestra la ventana principal definida en `inicio_app.py`.

En ella suele haber:

- Un **panel lateral** con botones como:
    - **Inicio**
    - **Calendario**
    - **Tareas**
    - **Avance**
    - **Extensiones**
    - **Salir**

- Un **área central** donde se muestra el contenido según la sección seleccionada.

Opción: Inicio

- Muestra una vista general y datos rápidos para que el usuario consulte
- Puede mostrar la fecha actual, un resumen de eventos o instrucciones básicas de uso.

10.3 Sección Tareas

Al seleccionar la opción **Tareas**, la aplicación utilizará las funciones de `google_classroom.py` para conectarse a la cuenta del usuario.

Funciones principales:

- **Obtener cursos:** lista de cursos (`obtener_cursos`).
- **Obtener tareas:** tareas de un curso específico (`obtener_tareas`).
- **Obtener anuncios:** anuncios publicados en el curso (`obtener_anuncios`).

Flujo típico:

1. A la primera vez, si no hay token válido, se abrirá una ventana en el navegador pidiendo permisos a Google Classroom.
2. Una vez autorizado, la aplicación descargará la lista de cursos.
3. El usuario podrá seleccionar un curso y ver:
    - Tareas asignadas.
    - Anuncios publicados por el profesor.
4. La aplicación mostrará esta información en el área central (por ejemplo, en tablas, listas o cuadros de texto).

10.4 Sección Calendario

La opción **Google Calendar** hace uso de `google_calendar.py`.

Funciones principales:

- `obtener_eventos(...)`: obtener los eventos del calendario dentro de un rango de fechas.
- `crear_evento(titulo, fecha, hora_inicio, hora_fin)`: crear nuevos eventos.
- `eliminar_evento(event_id)`: eliminar eventos existentes.

Flujo típico de uso:

1. Si no existe un `token.json` válido, se abrirá el navegador para autorizar el acceso a su Google Calendar.
2. La aplicación mostrará los eventos próximos del calendario principal del usuario.
3. El usuario podrá:
    - **Consultar eventos** en un rango de fechas.
    - **Crear evento nuevo**, indicando:
        - Título (por ejemplo, “Parcial de Cálculo II”).
        - Fecha (en formato `YYYY-MM-DD`).
        - Hora de inicio y fin (por ejemplo, `14:00` y `16:00` en formato 24 horas).
4. La aplicación enviará este evento a Google Calendar y mostrará un mensaje de confirmación.

10.5 Sección Avance

Esta sección está pensada para mostrar datos importantes para el estudiante para que este pueda seguir el avance de su carrera, y pueda planificar sus próximas materias a cursar, por lo que en esta sección se le pregunta al usuario a que carrera pertenece, y dependiendo la carrera se le proporciona el pensum de su carrera. Así podrá revisar requisitos para inscribir materias, mirar los créditos e intensidad horaria, entre otras cosas.

10.6 Sección Extensiones

Esta página proporciona al usuario links directos a distintas herramientas que resultan ser muy útiles para gestionar su vida académica

---

11. Archivos de datos: `users.txt`, `token.json`, etc.

-

11.1 Archivos `credentials.json` y `token.json`

- `credentials.json`: archivo de credenciales descargado desde Google Cloud Console.
- `token.json`: archivo que se genera automáticamente al aprobar el acceso a Google Calendar y Google Classroom.

Recomendaciones de seguridad:

- No compartir estos archivos con terceros.
- No subirlos a repositorios públicos.

---

12. Solución de problemas (Troubleshooting)

A continuación, se listan algunos problemas frecuentes y posibles soluciones.

12.1 Error: "ModuleNotFoundError"

Síntoma: Al ejecutar `python inicio_app.py` aparece un error similar a:

> ModuleNotFoundError: No module named 'googleapiclient'

Causa posible: Falta instalar una librería con `pip`.

Solución:

1. Asegúrese de haber activado el entorno virtual.
2. Instale las dependencias necesarias, por ejemplo:

```
pip install google-api-python-client google-auth google-auth-oauthlib
pillow
```

3. Vuelva a ejecutar:

```
python inicio_app.py
```

12.2 Error de credenciales de Google

Síntoma: Mensajes indicando que `credentials.json` no existe o no es válido.

Solución:

1. Verifique que el archivo `credentials.json` está en la carpeta correcta.
2. Asegúrese de que las APIs de **Google Calendar** y **Google Classroom** estén habilitadas en Google Cloud Console.
3. Si el problema persiste, regenere el archivo de credenciales y reemplace el antiguo.

12.3 No se abre el navegador para autenticación

Síntoma: Al intentar acceder a Google no se abre el navegador.

Solución:

1. Verifique que dispone de un navegador web instalado y configurado como predeterminado.
2. Revise si el firewall o antivirus están bloqueando la apertura del navegador.
3. Pruebe ejecutar el programa como administrador en Windows.

12.4 Problemas con (login)

Síntoma: No permite iniciar sesión aunque el usuario exista.

Solución:

1. Revise el formato de los datos (por ejemplo, `usuario;contraseña`).
2. Asegúrese de no tener espacios en blanco extra o saltos de línea incorrectos.
3. Si no es posible intente iniciar sesión con google.

---

13. Seguridad y privacidad

- Las credenciales de Google se manejan mediante el flujo estándar de OAuth 2.0 de Google.
- La aplicación almacena tokens de acceso en `token.json` para evitar pedir credenciales constantemente.
- Los nombres de usuario y contraseñas utilizados en `Login.py` se almacenan en firebase
    - Se recomienda, en una versión futura, utilizar métodos de encriptación o hashes para mayor seguridad.
- No se recomienda compartir los archivos `credentials.json`, `token.json`

---

14. Mantenimiento y futuras mejoras

Este proyecto puede ampliarse con:

- Sistema más robusto de gestión de usuarios (hash de contraseñas, BD local).
- Más integraciones con herramientas académicas.
- Reportes automáticos de tareas y eventos.
- Exportación de la información a formatos como CSV o PDF.

Para actualizar las librerías:

```
pip install --upgrade google-api-python-client google-auth
google-auth-oauthlib pillow
```