MANUAL TÉCNICO

Backend del Proyecto de Grupos Estudiantiles

Presentado por:

Andreysty Peña Perez

Samuel

David Alejandro Orozco

Raúl Santiago Bermúdez Camacho

Presentado a:

Néstor German Bolívar Pulgarín

Asignatura:

Programación Orientada A Objetos

Universidad Nacional De Colombia

2025

IMPORTACIÓN DE LIBRERÍAS

El proyecto está construido en **Python** utilizando el framework **Flask** para la construcción del servidor backend y **Firebase** como plataforma de servicios en la nube para manejar autenticación, almacenamiento y base de datos.

El archivo **requirements.txt** contiene todas las librerías requeridas para la ejecución correcta del sistema. Su instalación se realiza mediante:

1. Instalación de librerías:

pip install -r requirements.txt

2. Librerías principales utilizadas:

Flask:

- Framework web para crear rutas, manejar peticiones y renderizar plantillas.
- Permite crear un servidor web ligero.
- Gestiona la lógica de navegación del usuario, sesiones y respuestas HTTP.

Firebase-admin:

- SDK de Firebase para gestionar base de datos y autenticación.

  Lo que permite:

  - Conectarse a Firebase_database (base de datos NoSQL).
  - Gestionar usuarios registrados.
  - Crear, leer, eliminar y actualizar documentos en Firebase.
  - Es inicializado globalmente mediante **firebase_global.py**.

Python-dotenv:

- Carga credenciales sensibles desde `.env`.
- Evita exponer claves privadas dentro del código.

gunicorn:

Esta librería funciona como servidor WSGI para ejecutar aplicaciones Flask en entornos de producción.

Su inclusión es fundamental cuando el proyecto se despliega en plataformas como Render, ya que estas no utilizan

el servidor de desarrollo de Flask. Gunicorn permite que la aplicación funcione de manera estable, optimizada y

compatible con el manejo de múltiples solicitudes concurrentes. Gracias a él, Render puede iniciar el backend

utilizando el comando de producción correspondiente.

EJECUCION DEL PROYECTO

1.Clonar el repositorio

Dirígete al repositorio "https://github.com/Lunatico117/proyecto_grupos_estudiantiles" y utiliza el comando `git clone` para descargarlo.
Luego, en la rama principal, crea un archivo llamado `.env` (debe permitir escribir texto normalmente).

2. Creación y configuración del proyecto en Firebase Realtime Database

1. Ingresa a **Firebase**, regístrate o inicia sesión.
2. Crea un **nuevo proyecto**.
3. Dentro del proyecto, ve a **Realtime Database** y copia la **URL** que se genera.
4. En el archivo `.env`, agrega la variable:

```
FIREBASE_DB_URL=https://tu-url-de-firebase.com
```

5. Ahora vuelve a Firebase y haz clic en el **ícono de configuración**.
6. Entra a **Configuración del proyecto** → **Cuentas de servicio**.
7. Selecciona el lenguaje **Python** y haz clic en **Generar nueva clave privada**.
   Esto descargará un archivo `.json` con tus credenciales.
8. Ubica el archivo en un lugar conveniente, copia su ruta completa y en el `.env` agrega:

```
FIREBASE_CREDENTIALS="ruta/al/archivo.json"
```

Guarda los cambios.

3. Ejecución

Ejecuta el archivo `app.py`, ubicado en la raíz del proyecto.
En la consola aparecerá un enlace: ábrelo en tu navegador y podrás usar el proyecto localmente.

ACTIVIDADES DENTRO DE LA CARPETA MODEL

Esta carpeta contiene las clases modelo que representan las entidades principales dentro de la plataforma. Su organización sigue un enfoque similar al del manual que me diste: cada modelo define atributos, métodos de conversión y funciones auxiliares.

1. Usuario.py

Representa a los usuarios registrados en la plataforma.

Atributos principales:

- `id`: Identificador único en Firebase.
- `nombre`: Nombre del usuario.
- `email`: Correo utilizado para iniciar sesión.
- `password`: Contraseña cifrada o en texto (según configuración).
- `grupos_inscritos`: Lista de IDs de los grupos del usuario.

Métodos:

- to_dict()
  Convierte el usuario a diccionario para enviarlo a Firestore.
- from_dict()
  Reconstruye un objeto Usuario a partir de un documento de Firebase.

Este patrón de serialización y deserialización es fundamental para trabajar con Firebase.

Grupo.py

Representa los grupos estudiantiles disponibles dentro de la plataforma.

Atributos:

- `id`
- `nombre`
- `descripcion`
- `categoria`
- `administrador` (ID del usuario creador)
- `miembros` (lista)
- `eventos` (lista)

Métodos:

- `agregar_miembro(id_usuario)`
- `agregar_evento(evento)`
- `to_dict()`

Estos métodos encapsulan la lógica asociada a la gestión interna de los grupos, manteniendo una estructura ordenada y fácilmente expandible.

Evento.py

Define la estructura de los eventos creados por cada grupo.

Atributos:

- `id`
- `titulo`
- `descripcion`
- `fecha`

- `grupo_id` (ID del grupo creador)

Métodos:

- `to_dict()`

- `from_dict()`

ACTIVIDADES DENTRO DE LA CARPETA SERVICES

Esta carpeta contiene la lógica de conexión con Firebase y los métodos CRUD para cada entidad.

1. firebase_global.py

Este archivo contiene:

- Carga de credenciales del archivo JSON de Firebase.
- Inicialización del SDK.
- Creación de la instancia global de Firestore.

2. firebase.py

Este archivo reúne todas las funciones que permiten manipular datos desde y hacia Firebase.

Funciones principales:

Usuarios

- `crear_usuario(datos_usuario)`
- `obtener_usuario_por_id(id)`
- `eliminar_perfil():`
- `Actualizar_usuario`

Grupos

- `crear_grupo(datos_grupo)`
- `obtener_grupos()`
- `actualizar_grupo(id, nuevos_datos)`
- `Eliminar_club(id_grupo)`
- salirme_club(id_grupo)

Eventos

- `crear_evento(datos_evento)`
- `obtener_eventos_por_grupo(grupo_id)`

ACTIVIDADES DEL BACKEND PRINCIPAL

El archivo app.py define rutas, lógica de navegación, interacción con servicios y renderización de plantillas HTML.

Funciones del backend:

- Procesar formularios HTML.
- Manejar sesiones de usuario.
- Renderizar plantillas Jinja2.
- Enviar datos a los servicios de Firebase.
- Navegar entre vistas internas del sistema.

Rutas principales:

Ruta	Ruta
/iniciar_sesion	Validación del usuario mediante correo y contraseña.
/registrarse	Creación de cuenta en la base de datos.
/grupos	Muestra todos los grupos disponibles.
/crear_evento	Registrar eventos asociados a grupos.
/perfil	Mostrar información del usuario y grupos inscritos.

UTILIDADES

Funciones:

- `validar_email(email)`
- `validar_password(password)`
- `validar_campos_vacios(campos)`
- `validar_longitud(campo, minimo, maximo)`

Estas validaciones garantizan:

- Que no existan campos vacíos.
- Que el correo tenga formato correcto.
- Que las contraseñas cumplan requisitos mínimos.
- Que los textos enviados al servidor tengan longitudes permitidas.

INSTALACIÓN DEL SOFTWARE

1. **Descargar el proyecto y extraerlo.**
2. **Crear entorno**

virtual: python -m venv

venv

3. **Activar entorno:**

source venv/bin/activate (Linux/Mac)
venv\Scripts\activate (Windows)

4. **Instalar dependencias:**

pip install -r requirements.txt

5. **Agregar credenciales de Firebase en src/services/**
6. **Ejecutar el**

servidor: python

app.py

ARQUITECTURA DEL SISTEMA

El sistema está diseñado bajo una arquitectura modular que separa claramente la lógica del negocio, la conexión con Firebase y la gestión de rutas. Esto permite que cada componente pueda actualizarse o escalarse sin afectar el resto del proyecto, garantizando mantenibilidad y flexibilidad a largo plazo.

VENTAJAS DEL USO DE FIREBASE

Firebase ofrece múltiples ventajas que fortalecen el funcionamiento del sistema:

- Base de datos NoSQL orientada a documentos, ideal para aplicaciones dinámicas.
- Autenticación centralizada y altamente segura.
- Escalabilidad automática sin necesidad de gestionar servidores.
- Integración nativa con aplicaciones web y backend en Python.

Estas características permiten que el sistema funcione de manera eficiente incluso si aumenta el número de usuarios y grupos.

POSIBLES MEJORAS Y FUTURO DEL PROYECTO

El proyecto puede evolucionar fácilmente agregando nuevas funcionalidades. Algunas recomendaciones:

- Implementar autenticación con Google o redes institucionales.
- Permitir que los administradores publiquen noticias, archivos e imágenes.
- Agregar notificaciones por correo para nuevos eventos o recordatorios.
- Crear un panel de administración con métricas de participación estudiantil.
- Añadir un buscador avanzado de grupos por categorías, palabras clave o popularidad.

Estas mejoras ampliarían la utilidad del sistema y aumentarían su impacto dentro de la comunidad estudiantil.

CONSIDERACIONES DE SEGURIDAD

Para garantizar la integridad de los datos se recomienda:

- Cifrar las contraseñas antes de almacenarlas.
- Limitar permisos en los archivos JSON de Firebase.

- Utilizar variables de entorno para todas las claves privadas.
- Restringir accesos mediante reglas de seguridad de Firestore.

Con estas medidas se evita la filtración de información y se asegura el cumplimiento de buenas prácticas de desarrollo.