Prueba Técnica Excuela: Python, Flask, Servicios RESTful y Autenticación JWT

Objetivo
Desarrollar una API RESTful utilizando Flask que permita la gestión de usuarios y la autenticación mediante JSON Web Tokens (JWT).

Requisitos
1. *Estructura del Proyecto*:
   - El proyecto debe seguir una estructura de carpetas bien organizada.
   - Debe incluir un archivo `README.md` con instrucciones para configurar y ejecutar el proyecto.

2. *Entorno Virtual*:
   - Utilizar un entorno virtual para gestionar las dependencias del proyecto.
   - Incluir un archivo `requirements.txt` con todas las dependencias necesarias.

3. *Base de Datos*:
   - Utilizar cualquier motor como base de datos.
   - Crear un modelo de usuario con los siguientes campos: `id`, `username`, `password_hash`, `email`, `created_at`.
   - Suma puntos (no obligatorio): usar firestore o mongodb

4. *Rutas y Funcionalidad*:
   - Implementar las siguientes rutas:

     - **Registro de Usuario**:
       - Ruta: `/register`
       - Método: `POST`
       - Datos de entrada: `username`, `password`, `email`
       - Validar que el `username` y `email` sean únicos.
       - Guardar la contraseña de forma segura (no en texto plano).

     - **Login de Usuario**:
       - Ruta: `/login`
       - Método: `POST`
       - Datos de entrada: `username`, `password`
       - Validar las credenciales del usuario.
       - Generar un JWT si las credenciales son válidas.

     - **Obtener Información del Usuario**:
       - Ruta: `/user`
       - Método: `GET`

  - Requiere autenticación mediante JWT.
  - Retornar la información del usuario autenticado.

- **Actualizar Información del Usuario**:
  - Ruta: `/user`
  - Método: `PUT`
  - Datos de entrada: `username`, `email`
  - Requiere autenticación mediante JWT.
  - Permitir actualizar el `username` y `email`.

- **Eliminar Usuario**:
  - Ruta: `/user`
  - Método: `DELETE`
  - Requiere autenticación mediante JWT.
  - Eliminar la cuenta del usuario autenticado.

5. *Autenticación y Autorización*:
   - Implementar la generación y verificación de JWT.
   - Proteger las rutas que requieren autenticación utilizando JWT.

6. *Pruebas* (opcional):
   - Incluir pruebas unitarias para las funcionalidades principales de la API.
   - Las pruebas deben verificar el correcto funcionamiento de las rutas y la autenticación.

7. *Documentación* (opcional):
   - Documentar todas las rutas y su uso en el archivo `README.md`.
   - Incluir ejemplos de solicitudes y respuestas para cada ruta.

Entrega
El candidato debe entregar un repositorio (por ejemplo, en GitHub) con el código fuente, incluyendo:
- El código del proyecto.
- Un archivo `README.md` detallado.
- Las pruebas unitarias.
La dirección pública de repositorio debe ser enviada a postulacion-dev@excuela.com.

Criterios de Evaluación
- Funcionalidad: Todas las rutas y funcionalidades deben funcionar correctamente según lo especificado.
- Calidad del Código: El código debe ser limpio, bien organizado y seguir buenas prácticas de desarrollo.
- Seguridad: La implementación de la autenticación y manejo de contraseñas debe ser segura.

Esta prueba pretende evaluar las habilidades del candidato en Python y Flask, así como su capacidad para implementar servicios RESTful y manejar la autenticación con JWT.
BONUS: Los candidatos que desarrollen sus soluciones haciendo uso de estas herramientas y técnicas tendrán más oportunidades de ser elegidos (opcional):

- Entregable dockerizado (docker)
- Autenticación con firebase
- Persistencia con firestore
- Clean architecture.