Prueba Técnica: Desarrollo de un CRUD con FastAPI

Objetivo:

Desarrollar una aplicación RESTful utilizando FastAPI, Pydantic v2, y SQLAlchemy con las siguientes características:

Requerimientos:

1. Configuración Inicial y Modelado:

- Crea una API que maneje una entidad principal (por ejemplo, User) y que tenga relaciones con al menos otras dos entidades secundarias (ej. Post y Comment).
- Las relaciones deben incluir:
    - **Uno a muchos** (Ej. Un User puede tener muchos Post).
    - **Muchos a muchos** (Ej. Un Post puede tener muchos Tags, y un Tag puede tener muchos Post).
- Usa **migraciones** con Alembic:
    - Inicialmente, crea los modelos sin algunas relaciones.
    - Luego, utiliza migraciones para introducir cambios en los modelos, agregando una nueva relación o campo.
- Todas las operaciones con la base de datos deben ser asíncronas.

2. Soft Delete y Mixins:

- Implementa **soft-delete** en los modelos para que las entidades no se eliminen físicamente, pero no aparezcan en los listados de consultas.
- Crea un **mixin** que añada este comportamiento a los modelos.
- Implementa un query personalizado para filtrar elementos que estén eliminados.

3. Timestamps Genéricos:

- Todos los modelos deben incluir los campos created_at y updated_at, manejados de manera genérica a través de un **mixin** reutilizable.

4. Protección de Endpoints:

- Utiliza **OAuth2** con JWT para proteger los endpoints. Los usuarios deben autenticarse para acceder a los endpoints de creación, modificación y eliminación.
- Crea un endpoint de registro y uno de login, generando un token JWT que permita la autenticación para los otros endpoints.

5. Routers y Middleware:

- Separa la lógica en **routers** organizados por entidad (por ejemplo, users.py, posts.py, etc.).

- Crea un **middleware** que registre el tiempo de respuesta de las solicitudes y lo imprima en la consola para cada request.

Extra (opcional):

- Implementa **paginación** en los endpoints de consulta.
- Implementa **validaciones** adicionales utilizando Pydantic v2 (ej. restricciones de longitud de campo, formato de correo electrónico en el usuario).
- Despliega la aplicación utilizando **Docker**.
- Agrega un sistema de permisos para que solo el usuario que creó un post pueda editarlo o eliminarlo.

Criterios de Evaluación:

1. **Estructura y Organización del Código:**
    - Uso correcto de routers, middlewares y mixins.
    - Correcta separación de responsabilidades y modularidad.
2. **Dominio de SQLAlchemy y Pydantic:**
    - Uso apropiado de las relaciones entre tablas.
    - Implementación correcta de migraciones y mixins.
3. **Seguridad:**
    - Correcta implementación de OAuth2 y protección de los endpoints.
4. **Funcionalidad Completa:**
    - El CRUD debe ser totalmente funcional con las relaciones establecidas y el soft-delete correctamente implementado.
5. **Calidad del Código:**
    - Claridad, buenas prácticas de codificación, uso adecuado de PEP8 y manejo de excepciones.