Tarea: Crear el “INIT” de un Proyecto Backend con Arquitectura Hexagonal y CQRS

Objetivo

El objetivo es establecer una base sólida para un proyecto backend que implemente:

- **Arquitectura Hexagonal** para garantizar independencia del dominio.
- **CQRS (Command Query Responsibility Segregation)** para operaciones diferenciadas de lectura y escritura.
- **Bundle-contexts** como estrategia para agrupar lógicamente los componentes, facilitando la escalabilidad y el manejo modular del proyecto.

Queremos evaluar tu capacidad para estructurar, configurar y pensar en un diseño robusto que permita crecer el proyecto sin comprometer la mantenibilidad.

Requisitos

Lenguaje y Frameworks

- Python **3.x**
- **FastAPI** como framework web.
- **SQLAlchemy** para manejo de datos.

Arquitectura Hexagonal

- El dominio debe ser independiente de las herramientas externas (frameworks, bases de datos, etc.).
- Crear adaptadores para la interacción con:
    - Infraestructura (bases de datos, colas de mensajes).
    - Interfaces (API REST y consumidores de eventos).

CQRS

- **Separación estricta** entre comandos y consultas.
- Los comandos deben procesarse mediante **RabbitMQ** (con al menos un consumidor implementado).
- Las consultas deben ejecutarse directamente contra el modelo de lectura.

Bundle-contexts

- Define al menos dos **contexts** lógicos: users y auth.
- Cada context debe incluir:

    - Dominio
    - Aplicación
    - Infrastructura
- Los contexts deben estar agrupados bajo un sistema modular para facilitar la reutilización.

Implementar una gestión básica de usuarios:

1. **Comando**: Crear un usuario con nombre, correo y contraseña (almacenada de forma segura).
2. **Consulta**: Obtener información de un usuario por su ID.

Requisitos Adicionales

1. **Pruebas**
    - Escribe pruebas unitarias para los casos de uso.
    - Asegúrate de cubrir al menos el 80% del código de la capa de dominio.
2. **Docker**
    - Proporciona un **Dockerfile** y un **docker-compose.yml** con los servicios necesarios (RabbitMQ, base de datos).
3. **Documentación**
    - Incluye un README detallado que explique:
        - La estructura del proyecto.
        - Cómo ejecutar el proyecto y las pruebas.
        - Decisiones arquitectónicas.

Puntos de Evaluación

1. **Diseño Modular**
    - Uso correcto de la arquitectura hexagonal.
    - Implementación efectiva de bundle-contexts.
2. **Separación de Responsabilidades**
    - Las capas deben estar claramente definidas y respetar su propósito.
3. **CQRS**
    - Adecuada separación entre lectura y escritura.
    - Uso efectivo de RabbitMQ.
4. **Escalabilidad**
    - La estructura del código debe permitir añadir nuevos contexts sin dificultades.
5. **Pruebas**
    - Cobertura y calidad de las pruebas unitarias.
6. **Bundle-contexts**

    - Implementación clara y funcional de los contexts.
7. **Inyección de Dependencias**
    - Agregar soporte para inyección de dependencias en todos los componentes, permitiendo una inicialización y configuración dinámica con bajo acoplamiento.