Q&A: Codelab

1. ¿Cuál es el propósito principal de Clean Architecture en el desarrollo de software?

Clean Architecture busca dividir el sistema en capas con funciones claras, separando la lógica del negocio de detalles técnicos como la interfaz o el almacenamiento. Esto facilita que el software sea más fácil de mantener, escalar y adaptar a nuevas tecnologías.

2. ¿Qué beneficios aporta Clean Architecture a un microservicio en Spring Boot?

Al usar Clean Architecture en un microservicio con Spring Boot se logra:

- Separar la lógica de negocio del framework.
- Mejorar la facilidad para hacer pruebas.
- Tener un código más limpio y organizado.
- Escalar funcionalidades sin afectar las existentes.
- Cambiar tecnologías (como bases de datos) sin modificar la lógica principal.

3. ¿Cuáles son las principales capas de Clean Architecture y qué responsabilidad tiene cada una?

- **Dominio**: Define las entidades y reglas de negocio sin depender de librerías externas.

- **Aplicación**: Contiene los casos de uso, encargados de coordinar el comportamiento del dominio.

- **Infraestructura**: Implementa las conexiones con tecnologías externas como bases de datos o APIs.

- **Presentación**: Administra la entrada y salida de datos, generalmente con controladores REST.

4. ¿Por qué se recomienda desacoplar la lógica de negocio de la infraestructura en un microservicio?

Porque esto permite modificar o reemplazar tecnologías (como cambiar de base de datos) sin alterar la lógica del sistema, y facilita las pruebas sin necesidad de usar servicios reales.

5. ¿Cuál es el rol de la capa de aplicación y qué tipo de lógica debería contener?

Esta capa gestiona los casos de uso del sistema, aplicando reglas específicas, validaciones o cálculos. Se centra en cómo interactúan las entidades, sin preocuparse por la interfaz o el almacenamiento.

6. ¿Qué diferencia hay entre un UseCase y un Service en Clean Architecture?

- **UseCase**: Representa una acción concreta del sistema y tiene una lógica bien definida.

- **Service**: Es un concepto más amplio que puede mezclar distintas responsabilidades; útil en proyectos simples, pero en sistemas grandes se prefiere dividir en UseCases para mayor claridad.

7. ¿Por qué se recomienda definir Repositories como interfaces en la capa de dominio en lugar de usar directamente JpaRepository?

Porque así se evita depender de una tecnología específica. Al usar interfaces, se puede cambiar la forma de persistencia sin modificar el núcleo del sistema.

8. ¿Cómo se implementa un UseCase en un microservicio con Spring Boot y qué ventajas tiene?

Un UseCase se crea como una clase de servicio en la capa de aplicación que gestiona el uso de repositorios. Las ventajas incluyen:

- Separación clara de la lógica.
- Posibilidad de cambiar reglas sin alterar otras partes.
- Pruebas más sencillas.
- Mejor organización del código.

9. ¿Qué problemas podrían surgir si no aplicamos Clean Architecture en un proyecto de microservicios?

- Las capas pueden mezclarse y crear un código difícil de entender.
- Las pruebas se vuelven más complicadas.
- Cambiar tecnologías puede afectar todo el sistema.
- Se reduce la capacidad de escalar y mantener el proyecto.
- Aumenta el riesgo de errores al agregar nuevas funciones.

10. ¿Cómo Clean Architecture facilita la escalabilidad y mantenibilidad en un entorno basado en microservicios?

Al estructurar cada microservicio de forma modular y separada, se permite que cada componente evolucione, se pruebe y se despliegue de manera independiente, algo esencial para sistemas distribuidos.