Codelab:

Clean Architecture en microservicios Spring Boot

Desarrollo de Software III

Henao Aricapa, STIVEN – 2259603

Docente:

Salazar, ÁLVARO

Sede Tuluá

Junio de 2025

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

El propósito principal de Clean Architecture es separar las responsabilidades del software en capas bien definidas, desacoplando la lógica de negocio de la infraestructura y la presentación. Esto permite que los sistemas sean más mantenibles, escalables y adaptables al cambio tecnológico.

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

Aplicar Clean Architecture en un microservicio con Spring Boot permite:

- Independencia tecnológica (la lógica no depende de frameworks).
- Mayor facilidad para realizar pruebas unitarias.
- Código más modular y mantenible.
- Escalabilidad: permite agregar nuevas funcionalidades sin romper otras.
- Reemplazo sencillo de tecnologías (por ejemplo, cambiar de JPA a MongoDB sin afectar la lógica central).

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

- **Dominio (Enterprise Business Rules):** Contiene las entidades y reglas de negocio puras, sin dependencia de frameworks.
- **Aplicación (Application Business Rules):** Define los casos de uso (UseCases) y orquesta la lógica del dominio.
- **Infraestructura (Adapters & Frameworks):** Implementa servicios externos (bases de datos, colas, APIs) y configura tecnologías como JPA o Spring.
- **Presentación (Delivery Mechanisms):** Controladores REST y validaciones, gestiona la entrada/salida del sistema.

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

Porque así se facilita el mantenimiento y evolución del sistema. Si la infraestructura cambia (por ejemplo, de MySQL a PostgreSQL), no es necesario modificar la lógica de negocio. Además, se pueden realizar pruebas unitarias más efectivas sin necesidad de instanciar dependencias externas.

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

La capa de aplicación orquesta los flujos del sistema mediante casos de uso. Contiene la lógica de aplicación, como validaciones, cálculos y reglas que coordinan las entidades del dominio, sin preocuparse por cómo se almacenan o muestran los datos.

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

- **UseCase:** Representa una operación específica del sistema (ej. "Listar productos") y se enfoca en una lógica de negocio bien definida.

- **Service:** Es un concepto más amplio que puede contener múltiples responsabilidades. En proyectos pequeños puede ser suficiente, pero en sistemas complejos puede mezclarse lógica de distintos flujos, por lo que es mejor dividir en UseCases.

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

Porque usar interfaces en el dominio permite desacoplar la lógica de negocio de cualquier tecnología de persistencia. De esta forma, se puede cambiar la implementación (JPA, Mongo, REST, etc.) sin modificar la lógica central del sistema.

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

Un UseCase se implementa como una clase de servicio en la capa de aplicación que orquesta el acceso a los repositorios del dominio. Ventajas:

- Aisla la lógica de negocio.
- Facilita cambios en reglas sin afectar controladores o infraestructura.
- Favorece pruebas unitarias.
- Mejora la organización del código por responsabilidades.

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

- Acoplamiento entre capas (por ejemplo, lógica de negocio mezclada con controladores o repositorios).
- Dificultad para realizar pruebas.
- Problemas al cambiar la base de datos o la tecnología.
- Menor mantenibilidad y escalabilidad.
- Riesgo de errores al introducir nuevas funcionalidades.

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

Clean Architecture permite que cada microservicio tenga una estructura modular, desacoplada y bien organizada. Esto hace que cada parte del sistema pueda escalar, testearse, desplegarse y mantenerse de forma independiente, lo cual es clave en arquitecturas distribuidas.