Q&A: Codelab - enfoque ADD y Clean Architecture

1. ¿Qué es Attribute-Driven Design (ADD) y cuál es su propósito en el diseño de software?

ADD es un enfoque de diseño arquitectónico que se centra en los atributos de calidad del sistema, como el rendimiento o la seguridad. Su objetivo es tomar decisiones de diseño basadas en requisitos no funcionales para asegurar que la arquitectura cumpla con los estándares de calidad esperados.

2. ¿Cómo se relaciona ADD con Clean Architecture en el proceso de diseño de sistemas?

ADD se usa al inicio del diseño para establecer las metas de calidad que el sistema debe cumplir. Luego, Clean Architecture entra en juego para implementar esa arquitectura de forma organizada, favoreciendo un diseño desacoplado y fácil de mantener.

3. ¿Cuáles son los pasos principales del método ADD para definir una arquitectura de software?

Los pasos principales son:

1. Determinar los requisitos y los atributos de calidad clave.
2. Identificar limitaciones tecnológicas.
3. Elegir tácticas arquitectónicas específicas para cada atributo.
4. Diseñar los módulos principales y sus relaciones.
5. Validar y ajustar la arquitectura con revisiones y pruebas.

4. ¿Cómo se identifican los atributos de calidad en ADD y por qué son importantes?

Se determinan a partir de los requerimientos del sistema y lo que espera el cliente o negocio. Son esenciales porque guían el diseño hacia objetivos

como la seguridad, disponibilidad o eficiencia, impactando directamente en la calidad del producto final.

5. ¿Por qué Clean Architecture complementa ADD en la implementación de una solución?

Porque Clean Architecture permite traducir lo definido en ADD a un diseño de software concreto, organizado en capas que favorecen el desacoplamiento, la escalabilidad y el mantenimiento, alineándose con las metas de calidad establecidas.

6. ¿Qué criterios se deben considerar al definir las capas en Clean Architecture dentro de un proceso ADD?

Se deben tener en cuenta:

- ★ Separación clara de responsabilidades.
- ★ Independencia de herramientas o tecnologías.
- ★ Que el dominio no dependa de detalles técnicos (inversión de dependencias).
- ★ Posibilidad de reutilizar y probar fácilmente el código.
- ★ Capacidad de soportar atributos clave como seguridad o rendimiento.

7. ¿Cómo ADD ayuda a tomar decisiones arquitectónicas basadas en necesidades del negocio?

ADD traduce los objetivos del negocio en atributos técnicos concretos. Esto permite elegir soluciones específicas (como cacheo o cifrado) que respondan directamente a esas necesidades, asegurando que la arquitectura esté alineada con las metas del sistema.

8. ¿Cuáles son los beneficios de combinar ADD con Clean Architecture en un sistema basado en microservicios?

★ El diseño se enfoca en los requisitos de calidad desde el principio.

★ Los servicios están bien definidos y separados.

★ El sistema es más fácil de mantener y probar.

★ Se pueden escalar componentes sin afectar la lógica principal.

★ Es más sencillo asegurar que cada servicio cumpla con sus atributos clave.

9. ¿Cómo se asegura que la arquitectura resultante cumpla con los atributos de calidad definidos en ADD?

A través de una etapa de validación que incluye:

★ Revisión del diseño arquitectónico.

★ Pruebas específicas (de carga, rendimiento, seguridad).

★ Detección de posibles problemas de rendimiento.

★ Ajustes en la implementación para mejorar la calidad.

10. ¿Qué herramientas o metodologías pueden ayudar a validar una arquitectura diseñada con ADD y Clean Architecture?

Algunas opciones son:

★ Revisiones por expertos o comités de arquitectura (ARB).

★ Herramientas de prueba como JMeter o Gatling.

★ Escáneres de seguridad como OWASP ZAP o SonarQube.

★ Análisis estático del código (ej. PMD, SonarQube).

★ Sistemas de monitoreo como Prometheus con Grafana.

★ Métodos de evaluación como ATAM para analizar compromisos arquitectónicos.