JUAN SEBASTIAN BARONA – 2266123

CODELAB 2

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

Es un método estructurado para diseñar arquitecturas de software, enfocado en los *atributos de calidad* como el rendimiento, la seguridad o la escalabilidad. El propósito principal de ADD es ayudar a los arquitectos a tomar decisiones fundamentadas, partiendo de los requerimientos del sistema, especialmente los no funcionales, para construir una arquitectura que los cumpla.

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

ADD se encarga de definir la arquitectura con base en atributos de calidad, mientras que Clean Architecture proporciona una manera estructurada de organizar el código, separando claramente las responsabilidades. En conjunto, ADD nos dice qué atributos necesitamos garantizar y Clean Architecture nos da una guía sobre cómo estructurar el sistema para cumplirlos.

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

Los pasos principales de ADD son:

1. Seleccionar un módulo o elemento a diseñar.
2. Identificar los requerimientos funcionales y los atributos de calidad relevantes.
3. Elegir tácticas arquitectónicas y patrones que satisfagan esos atributos.
4. Dividir el módulo en submódulos.
5. Documentar y validar las decisiones tomadas.

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

Se identifican analizando los requisitos no funcionales del sistema, conversaciones con stakeholders y documentos de especificación. Son importantes porque determinan la experiencia del usuario, el mantenimiento del sistema y su capacidad de evolución. Por ejemplo, si un sistema necesita alta disponibilidad, la arquitectura debe incluir redundancia y balanceo de carga.

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

Porque Clean Architecture refuerza la separación de responsabilidades, la independencia de frameworks y la facilidad de pruebas. Esto ayuda a cumplir atributos de calidad como mantenibilidad, testabilidad y escalabilidad, que ADD prioriza desde el inicio del diseño.

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

Se deben considerar:

- Dependencias unidireccionales (de adentro hacia afuera).
- Independencia de frameworks externos.
- Separación clara entre lógica de negocio, casos de uso y detalles de infraestructura.
- Necesidades específicas del negocio (por ejemplo, si requiere alta seguridad, separar validación y control de acceso).

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

ADD traduce las necesidades del negocio en atributos de calidad y luego en decisiones arquitectónicas. Por ejemplo, si el negocio necesita responder rápido a los usuarios, ADD puede llevarnos a elegir una arquitectura basada en eventos para mejorar la latencia y escalabilidad.

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

- Se construyen servicios alineados con atributos como escalabilidad y resiliencia.
- La separación de responsabilidades facilita la evolución independiente de cada servicio.
- Las decisiones tomadas con ADD se implementan con una estructura clara gracias a Clean Architecture.
- Se mejora la mantenibilidad y la claridad del sistema completo.

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

- A través de revisiones arquitectónicas.
- Aplicando tácticas concretas para cada atributo (por ejemplo, cachés para rendimiento, colas para disponibilidad).
- Haciendo pruebas de carga, pruebas de seguridad o análisis estático de código.
- Validando con prototipos si es necesario.

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

- Herramientas de pruebas como JMeter, Postman o Gatling para validar atributos de rendimiento.

- SonarQube o ArchUnit para análisis estático y validación de dependencias.
- Pruebas automatizadas (unitarias, de integración, de aceptación).