Preguntas

1. **¿Qué es Attribute-Driven Design (ADD) y cuál es su propósito en el diseño de software?**
2. **¿Cómo se relaciona ADD con Clean Architecture en el proceso de diseño de sistemas?**
3. **¿Cuáles son los pasos principales del método ADD para definir una arquitectura de software?**
4. **¿Cómo se identifican los atributos de calidad en ADD y por qué son importantes?**
5. **¿Por qué Clean Architecture complementa ADD en la implementación de una solución?**
6. **¿Qué criterios se deben considerar al definir las capas en Clean Architecture dentro de un proceso ADD?**
7. **¿Cómo ADD ayuda a tomar decisiones arquitectónicas basadas en necesidades del negocio?**
8. **¿Cuáles son los beneficios de combinar ADD con Clean Architecture en un sistema basado en microservicios?**
9. **¿Cómo se asegura que la arquitectura resultante cumpla con los atributos de calidad definidos en ADD?**
10. **¿Qué herramientas o metodologías pueden ayudar a validar una arquitectura diseñada con ADD y Clean Architecture?**

RESPUESTAS

1. Básicamente, ADD se enfoca en tomar decisiones de diseño a partir de los requisitos de calidad, para que el software responda a lo que el negocio necesita en términos de rendimiento, seguridad y otros requisitos críticos.
2. ADD fija el “qué” (los requisitos de calidad) y Clean Architecture el “cómo” (la estructura) del diseño, aumentando así el mantenimiento, la escalabilidad y el acoplamiento débil del software.
3. Los principales paso de ADD incluyen:
    - Definir requisitos de calidad
    - considerar restricciones
    - aplicar tácticas
    - organizar módulos
    - verificar que el diseño satisfaga las exigencias planteadas.
4. Los atributos de calidad provienen de las necesidades del cliente y el negocio, ya que determinan el buen funcionamiento del producto en el entorno real.
5. Clean Architecture proporciona una estructura adecuada para implementar las soluciones de ADD, aumentando así el mantenimiento, el acoplamiento débil y facilitando el crecimiento del proyecto.

6. Hay que tener en cuenta responsabilidades, independencia de los frameworks, inversión de dependencias, reuse del código y que el diseño dé cabida tanto al rendimiento como a la seguridad.
7. ADD proporciona una guía para tomar mejores decisiones de diseño, eligiendo tácticas adecuadas que sean relevantes para los resultados de negocios deseados.
8. Combinar ADD y Clean Architecture proporciona un diseño más robusto, flexible, fácil de dar mantenimiento, escalable y capaz de satisfacer tanto requisitos técnicos como de calidad.
9. La validación se logra revisando el diseño, realizando pruebas de rendimiento, seguridad, análisis de cuellos de botella y corrigiendo lo que sea necesario.
10. Hay varias metodologías y herramientas para llevar esta validación a la práctica, como por ejemplo, las revisiones por pares, pruebas de estrés, los análisis de seguridad, análisis estático de calidad de código y el seguimiento constante de métricas en un entorno de ejecución.