1. **¿Qué es Attribute-Driven Design (ADD) y cuál es su propósito en el diseño de software?**
   Es una metodología para diseñar la arquitectura de software basándose en atributos de calidad del sistema, como rendimiento, seguridad, disponibilidad, escalabilidad, etc. Su propósito es tomar decisiones arquitectónicas guiadas por los requisitos no funcionales, lo cual permite que la arquitectura pueda satisfacer las necesidades del negocio desde el inicio y se adapte a las restricciones técnicas impuestas.
2. **¿Cómo se relaciona ADD con Clean Architecture en el proceso de diseño de sistemas?**
   ADD se utiliza para definir la estructura general del sistema en función de los atributos de calidad, mientras que Clean Architecture organiza la implementación del código en capas bien separadas y desacopladas.
   La relación entre ambos enfoques es complementaria: ADD define qué se necesita y por qué, y Clean Architecture define cómo implementarlo de forma mantenible, escalable y desacoplada
3. **¿Cuáles son los pasos principales del método ADD para definir una arquitectura de software?**
   Los pasos principales del método ADD son:
   - Identificar los requisitos funcionales y no funcionales del sistema.
   - Definir los atributos de calidad clave, como rendimiento, seguridad o disponibilidad.
   - Establecer restricciones tecnológicas como por ejemplo, el uso de microservicios o bases de datos.
   - Seleccionar tácticas arquitectónicas para cumplir los atributos definidos.
   - Definir los módulos principales y cómo interactúan entre ellos.
4. **¿Cómo se identifican los atributos de calidad en ADD y por qué son importantes?**
   Los atributos de calidad se identifican a partir de los requisitos del negocio y del entorno operativo del sistema.

   Son importantes porque orientan las decisiones arquitectónicas clave, como qué tecnologías usar, cómo organizar los componentes y qué patrones aplicar.

5. **¿Por qué Clean Architecture complementa ADD en la implementación de una solución?**
   Lo complementa porque traduce la arquitectura definida en ADD en una estructura de código clara y mantenible.
   A través de capas separadas y la inversión de dependencias, permite que la lógica de negocio se mantenga independiente de frameworks, bases de datos o interfaces externas, facilitando pruebas, escalabilidad y cambios tecnológicos sin reescribir el núcleo del sistema.
6. **¿Qué criterios se deben considerar al definir las capas en Clean Architecture dentro de un proceso ADD?**
   Los criterios clave son:

- Separación de responsabilidades: Cada capa debe tener una función específica.

- Inversión de dependencias: Las capas internas no deben depender de las externas.

- Aislamiento de la lógica de negocio: Debe estar en el centro del sistema, sin depender de detalles técnicos.

- Desacoplamiento de tecnologías: Los detalles de frameworks, bases de datos o servicios externos deben estar aislados en la infraestructura.

7. **¿Cómo ADD ayuda a tomar decisiones arquitectónicas basadas en necesidades del negocio?**
   ADD ayuda a alinear la arquitectura con las prioridades del negocio, al enfocar las decisiones en los atributos de calidad más relevantes.
   Por ejemplo, si el negocio necesita alta disponibilidad, ADD guiará hacia tácticas como replicación y balanceo de carga.
   Esto asegura que la solución no sólo funcione, sino que lo haga de manera efectiva en el contexto empresarial.
8. **¿Cuáles son los beneficios de combinar ADD con Clean Architecture en un sistema basado en microservicios?**
   Algunos de los beneficios son:
   - Arquitectura centrada en calidad: ADD garantiza que los microservicios cumplan con atributos clave como escalabilidad o rendimiento.

   - Desacoplamiento fuerte: Clean Architecture permite que cada microservicio tenga una estructura interna independiente y clara.

   - Facilidad para mantenimiento y evolución: Se pueden cambiar tecnologías o servicios externos sin tocar el núcleo del sistema.

   - Escalabilidad individual: Al tener cada microservicio con lógica de negocio bien aislada, se puede escalar por separado según demanda.

9. **¿Cómo se asegura que la arquitectura resultante cumpla con los atributos de calidad definidos en ADD?**
   Se asegura mediante:
   - Validaciones continuas durante el desarrollo como pruebas de carga, auditorías de seguridad, pruebas de disponibilidad, etc.

   - Revisiones arquitectónicas frecuentes para identificar desviaciones.

   - Métricas e indicadores que evalúan si se cumplen los requisitos definidos, por ejemplo un tiempo de respuesta menor a 2 segundos.

- Iteraciones que permiten refinar tácticas y ajustar la arquitectura con base en los resultados.

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

    Algunas herramientas/metodologías útiles son:

    - Pruebas de carga y rendimiento: JMeter, Gatling, k6.
    - Pruebas de seguridad: OWASP ZAP, SonarQube.
    - Revisión de arquitectura: ADR – Architectural Decision Records.
    - Análisis estático de código: SonarQube, PMD
    - Monitoreo y logging: Prometheus, Grafana, ELK Stack
    - Documentación con C4 Model para visualizar las capas y componentes

.