# Proceso con enfoque ADD y Clean Architecture

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

**Attribute-Driven Design (ADD)** es una metodología de diseño arquitectónico que se enfoca en definir la estructura de un sistema a partir de sus **atributos de calidad** (como rendimiento, disponibilidad, seguridad, etc.) y **restricciones tecnológicas**.
Su propósito principal es **guiar el diseño** de la arquitectura considerando desde el inicio los **requisitos no funcionales**, que son clave para el éxito del sistema, más allá de solo cumplir funcionalidades.

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

ADD y Clean Architecture se **complementan**:

- **ADD** se usa para **definir la arquitectura** inicial con base en atributos de calidad.
- **Clean Architecture** se usa para **implementar** esa arquitectura de forma organizada, con **capas separadas** que mejoran el mantenimiento y escalabilidad.

En conjunto, ADD guía el **qué** y el **por qué**, y Clean Architecture guía el **cómo** implementar.

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

1. **Definir atributos de calidad y restricciones.**
2. **Diseñar la arquitectura según esos atributos.**
3. (Complementado con Clean Architecture) **Implementar la arquitectura organizada por capas.**
4. **Validar y refinar la arquitectura** en función de su cumplimiento de los atributos definidos.

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

Se identifican a partir de los **requisitos del sistema** y las **necesidades del negocio**.
Ejemplos incluyen:

- Disponibilidad (ej. 99.9% uptime)
- Rendimiento (ej. respuesta < 2 segundos)
- Seguridad (ej. cifrado de datos)

Son importantes porque afectan directamente la **experiencia del usuario**, el **rendimiento del sistema** y su **fiabilidad**, lo que puede impactar el éxito del producto.

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

Porque Clean Architecture ofrece una **estructura desacoplada** y organizada del código:

- Permite **cambiar tecnologías (DB, frameworks)** sin tocar la lógica de negocio.
- Facilita **pruebas, mantenimiento y escalabilidad**.
- Aplica el **principio de inversión de dependencias**, clave para soluciones robustas.

Esto asegura que las decisiones de arquitectura tomadas con ADD se **implementen correctamente**.

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

- **Separación clara de responsabilidades**:
  - Dominio: lógica de negocio pura.
  - Aplicación: coordinación de casos de uso.
  - Infraestructura: bases de datos, APIs externas.
  - Presentación: interfaz de usuario o API REST.
- **Inversión de dependencias**: las capas internas no deben depender de las externas.
- **Atributos de calidad definidos en ADD**, como rendimiento, seguridad o mantenibilidad.

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

ADD vincula **cada decisión técnica** (como usar cache, replicación, cifrado) con un **atributo de calidad**, que a su vez responde a una **necesidad del negocio** (como alta disponibilidad o cumplimiento legal).
Esto **alinea la arquitectura con los objetivos empresariales** desde el principio.

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

- Diseño **orientado a calidad** (ADD).
- **Modularidad y escalabilidad** (Clean Architecture).
- Posibilidad de **cambiar tecnologías por servicio** sin afectar el núcleo del negocio.
- Mejor **mantenibilidad** y **pruebas más efectivas**.
- Arquitectura **iterativa y adaptable** ante nuevos requisitos o cambios.

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

A través de la **validación** en la última fase del proceso:

- **Pruebas de carga**, seguridad, rendimiento.
- **Revisión de arquitectura** por expertos.
- **Medición de métricas** clave (ej. tiempo de respuesta).
- **Refinamiento de tácticas** (ej. añadir cache, ajustar consultas SQL).

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

- **Pruebas de rendimiento**: JMeter, Gatling.
- **Análisis de seguridad**: OWASP ZAP, SonarQube.
- **Revisión arquitectónica**: ADRs (Architectural Decision Records), ATAM (Architecture Tradeoff Analysis Method).
- **Monitoreo y métricas**: Prometheus, Grafana.
- **Pruebas unitarias y de integración** para validar separación de capas.