

MÓDULO 2: ARQUITECTURA DE SOFTWARE

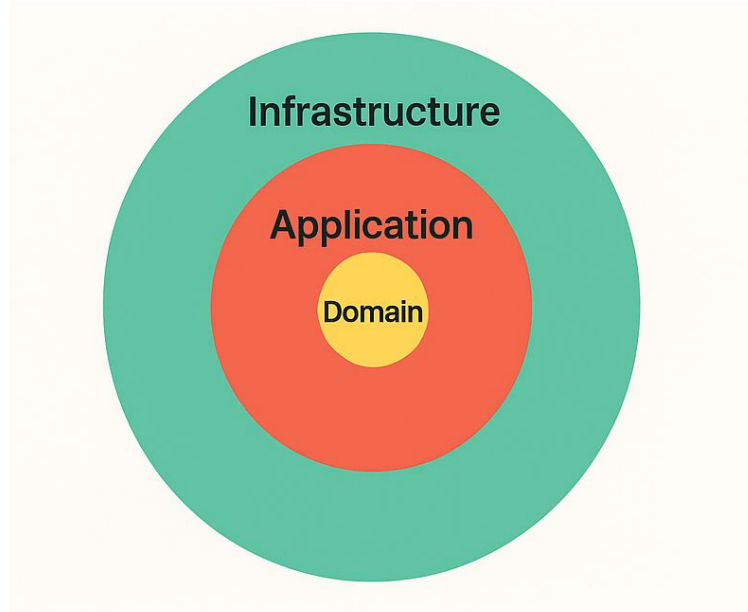
ESTILOS DE ARQUITECTURA CLEAN ARCHITECTURE

INTRODUCCIÓN A LA ARQUITECTURA DE SOFTWARE

BIG school

Estilos de arquitectura: Clean Architecture

Separar el QUÉ del CÓMO

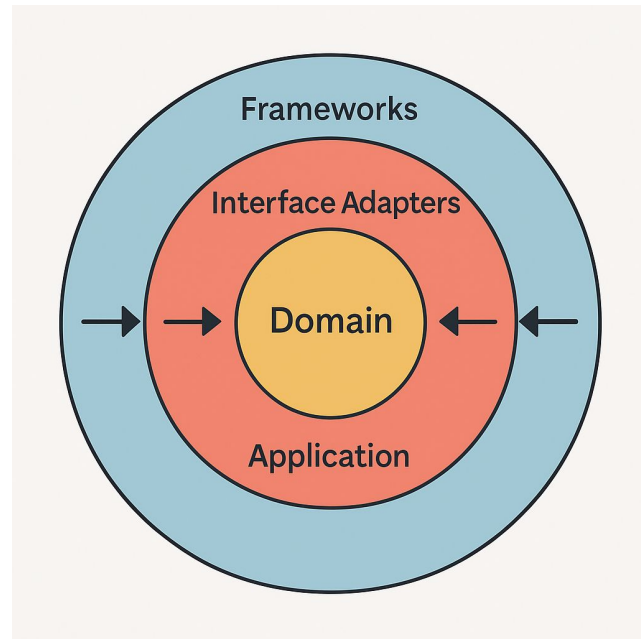


Idea Principal

Capas concéntricas

Dependencias siempre hacia adentro

Dominio no conoce frameworks ni tecnología



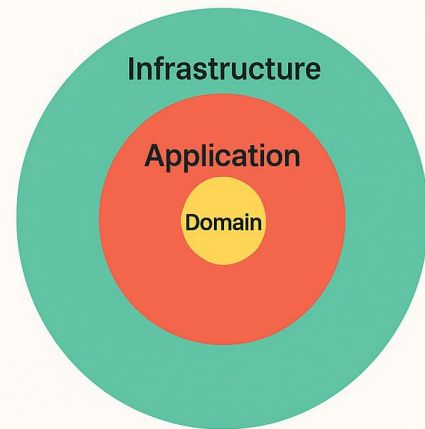
Capas de Clean

Dominio (Entities/Value Objects) → Reglas de negocio

Aplicación (Use Cases/Ports) → Orquestación

Adaptadores → Controladores, repositorios concretos

Infraestructura → DB, HTTP server, frameworks



Beneficios

- ✓ Dominio limpio y estable
- ✓ Alta testabilidad
- ✓ Flexibilidad tecnológica
- ✓ Mantenibilidad a largo plazo

Estructura de proyecto (Node/TS)

```
src/  
  domain/  
  application/  
  adapters/  
    http/  
    persistence/  
  infrastructure/  
  shared/
```

Ejemplo Node/TS

```
// Use case
class UpdateProductPrice {
    constructor(private repo: ProductRepository) {}
    async execute({ id, price }) { ... }
}
```

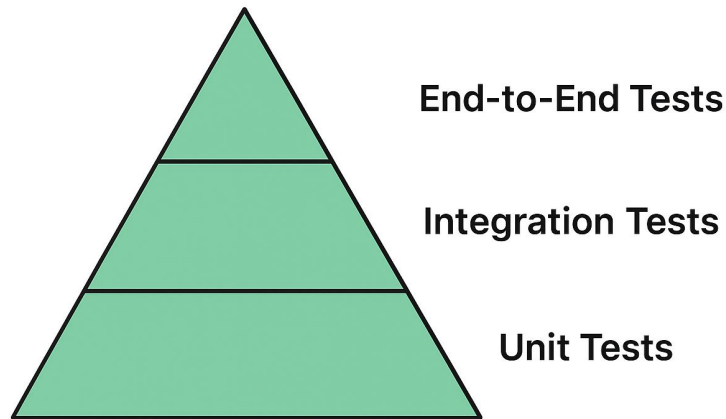
Testing en Clean

Unit tests en dominio y casos de uso

Contract tests para puertos/adaptadores

Integration tests para adaptadores reales

Fake repos para tests rápidos



Errores comunes

- ✗ Anemia de dominio
- ✗ Fugas de framework
- ✗ Overengineering
- ✗ DTOs acoplados a DB

Buenas prácticas

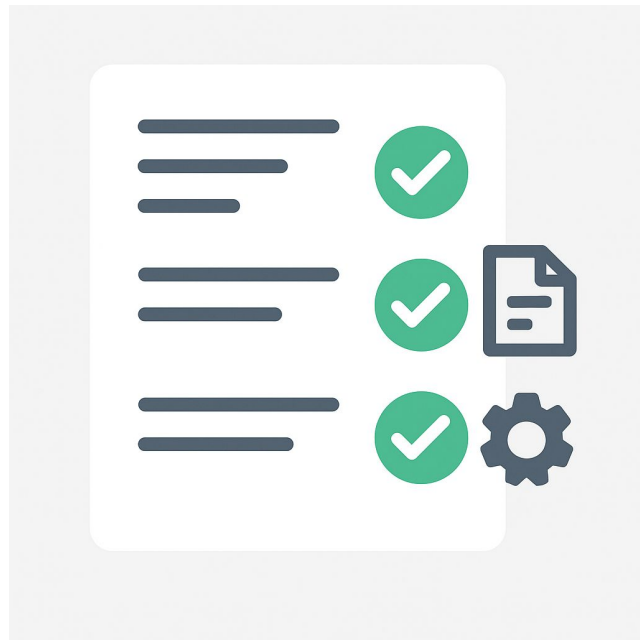
Interfaces claras como puertos

Adaptadores separados en **infra**

DTOs propios por caso de uso


Composition root único

Cross-cutting vía puertos



¿Cuándo conviene?

- ✓ Dominios complejos
- ✓ Cambios tecnológicos previstos
- ✓ Múltiples interfaces (REST, CLI, eventos)
- ✓ Necesidad de testabilidad
- ✗ Apps muy pequeñas / triviales



*"Clean Architecture protege el negocio del cambio
tecnológico"*

→ Próxima lección: **Event-Driven Architecture**

BIG school