

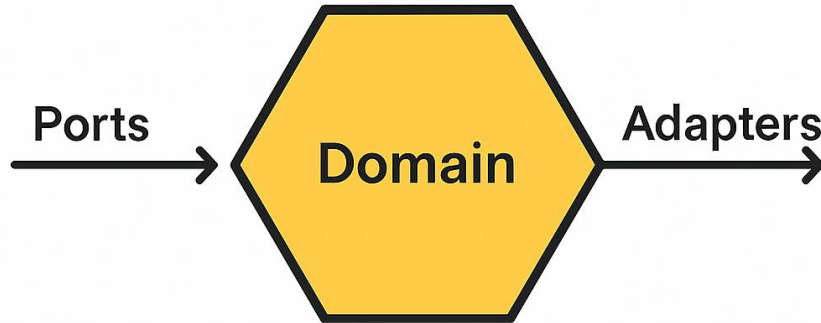
MÓDULO 2: ARQUITECTURA DE SOFTWARE

ESTILOS DE ARQUITECTURA HEXAGONAL (PORTS & ADAPTERS)

INTRODUCCIÓN A LA ARQUITECTURA DE SOFTWARE

Estilos de arquitectura: Hexagonal

Ports & Adapters (Puertos y Adaptadores)



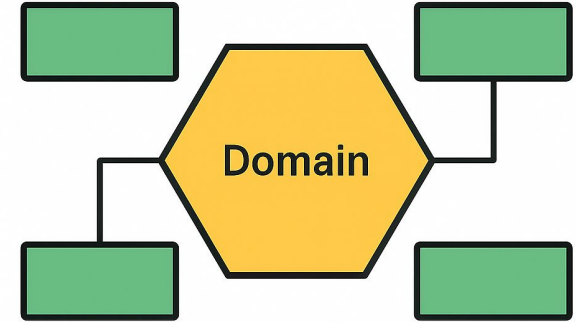
Idea principal

Separar **lógica de negocio** de la infraestructura

Núcleo estable = **dominio**

Bordes flexibles = **adaptadores**

Contratos claros = **puertos**

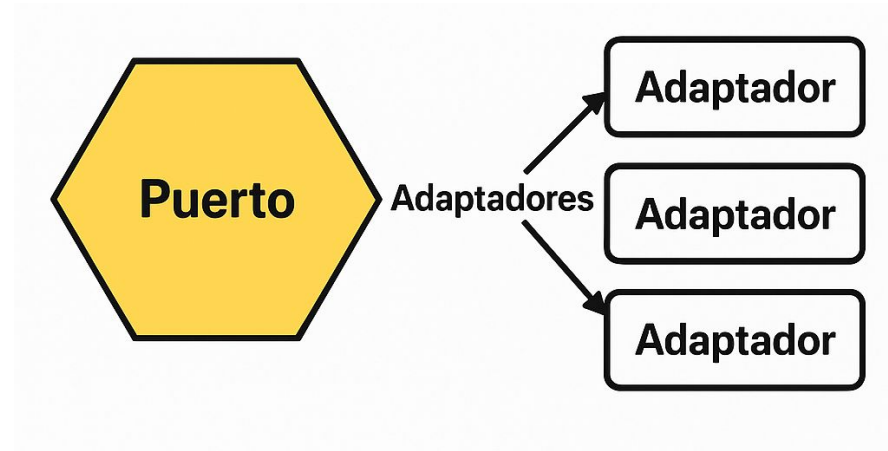


Puerto y Adaptador

- **Puerto** = interfaz/contrato
- **Adaptador** = implementación concreta

Ejemplo:

- `PaymentPort`
- `StripeAdapter` / `PayPalAdapter`



Ejemplo Node/TS

```
// Puerto
interface ProductRepo { findById(); save(); }

// Caso de uso
updateProductPrice(repo, {id, price})

// Adaptador
class PgProductRepo implements ProductRepo { ... }
```

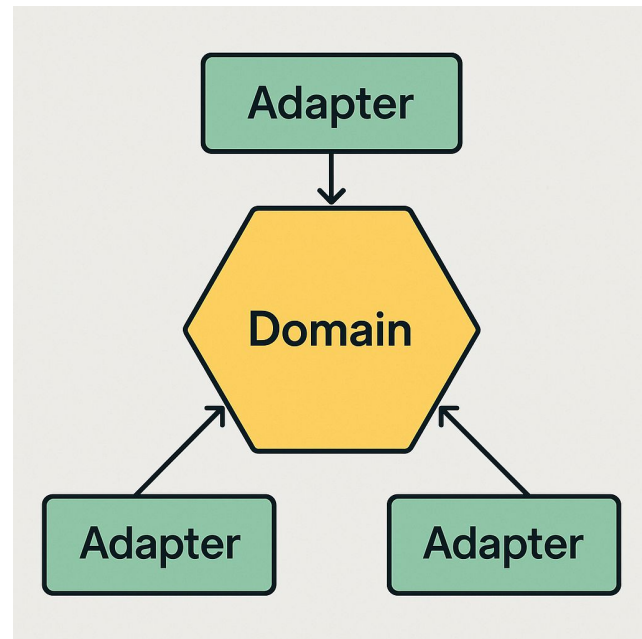
Beneficios

- ✓ Dominio limpio y estable
- ✓ Alta testabilidad
- ✓ Flexibilidad tecnológica
- ✓ Múltiples interfaces de entrada/salida

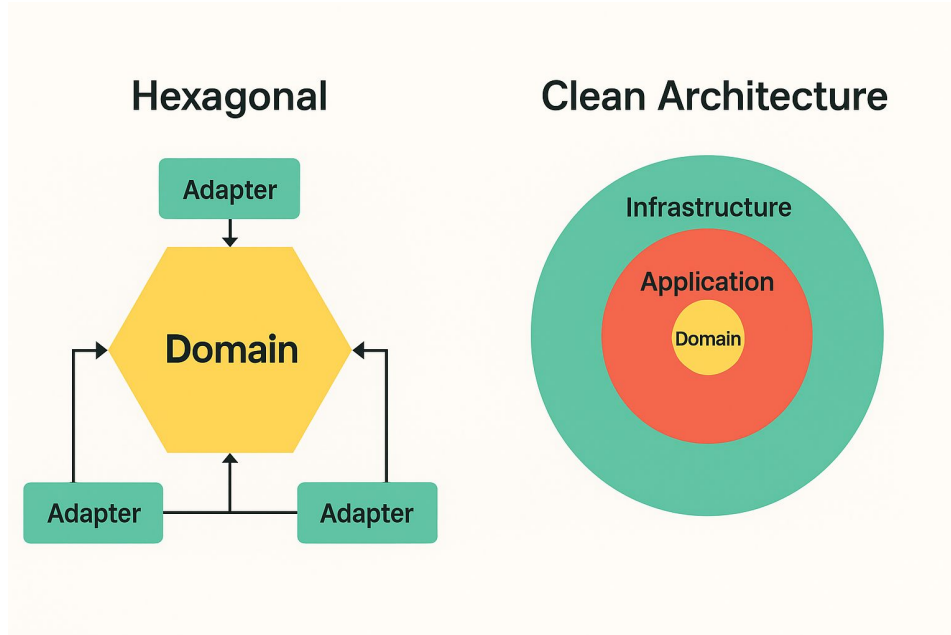
Múltiples entradas/salidas

Un mismo caso de uso puede ser expuesto:

- REST API
- CLI
- Mensajes/Eventos



Hexagonal vs Clean



- **Hexagonal:** foco en puertos y adaptadores
- **Clean:** capas concéntricas
- Ambos → mismo principio: dominio no depende de tecnología

Buenas prácticas

- Interfaces claras como puertos
- Nombres de dominio, no de tecnología
- Adaptadores separados en `infra`
- Wiring en el **composition root**
- Tests de contrato



¿Cuándo conviene?



Dominios complejos



Integraciones múltiples



Alta testabilidad



Apps muy pequeñas o triviales

*"Hexagonal separa el **qué** del **cómo**"*

***Dominio** = qué hace tu sistema.*

***Adaptadores** = cómo se conecta al mundo."*

→ Próxima lección: **Clean Architecture**

BIG school