4 Hexagonal application

La aplicación es un intermediario entre el dominio y el usuario.

1. CONNECT

- ¿Qué ocurre con la presentación?
- ¿Puede cambiar o coexistir con otras presentaciones?

> Objetivo: Todo lo que no es dominio son detalles.

2. CONCEPT

Conceptos clave:

- **Driving Adapters**: adaptadores de entrada (presentación).
- **Driven Adapters**: adaptadores de salida (persistencia).

Ideas fundamentales:

- La presentación también es un detalle.
- Pasamos a una visión concéntrica

3. CONCRETE PRACTICE

Partimos de capas de dominio y persistencia con ports y adapters.

Implementar la **Arquitectura Hexagonal** completa para la capa de aplicación.

- Capa de aplicación
  - [ ] Se crea una nueva carpeta `application`.
  - [ ] Se definen *use cases* a partir de los *services* de `domain`.
  - [ ] Se extraen y publican *ports* de los *use cases*.
- Capa de presentación
  - [ ] Desaparece como tal.
  - [ ] Los *handlers* se mueven a `infrastructure`
- Capa de infrastructure
  - [ ] Los handlers dependen de los *ports* de `application`.
  - [ ] Los handlers invocan a los *use cases*
    - (como antes hacían con los *services*).

- Configuración global
  - [ ] Se necesita un clase *configuration* para la inyección de dependencias.
  - [ ] Opcionalmente agrupa los *adapters* de `infrastructure` en `persistence` y `presentation`.

4. CONCLUSIONS

- De capas apiladas a concéntricas
- Dominio en el centro, infraestructura en el exterior
- La aplicación en el medio

> ¿Cómo de independiente es tu lógica de negocio?