Arquitectura Neta

Albert Clapés (aclapes@ub.edu)

Facultat de Matemàtiques i Informàtica
Universitat de Barcelona (UB)

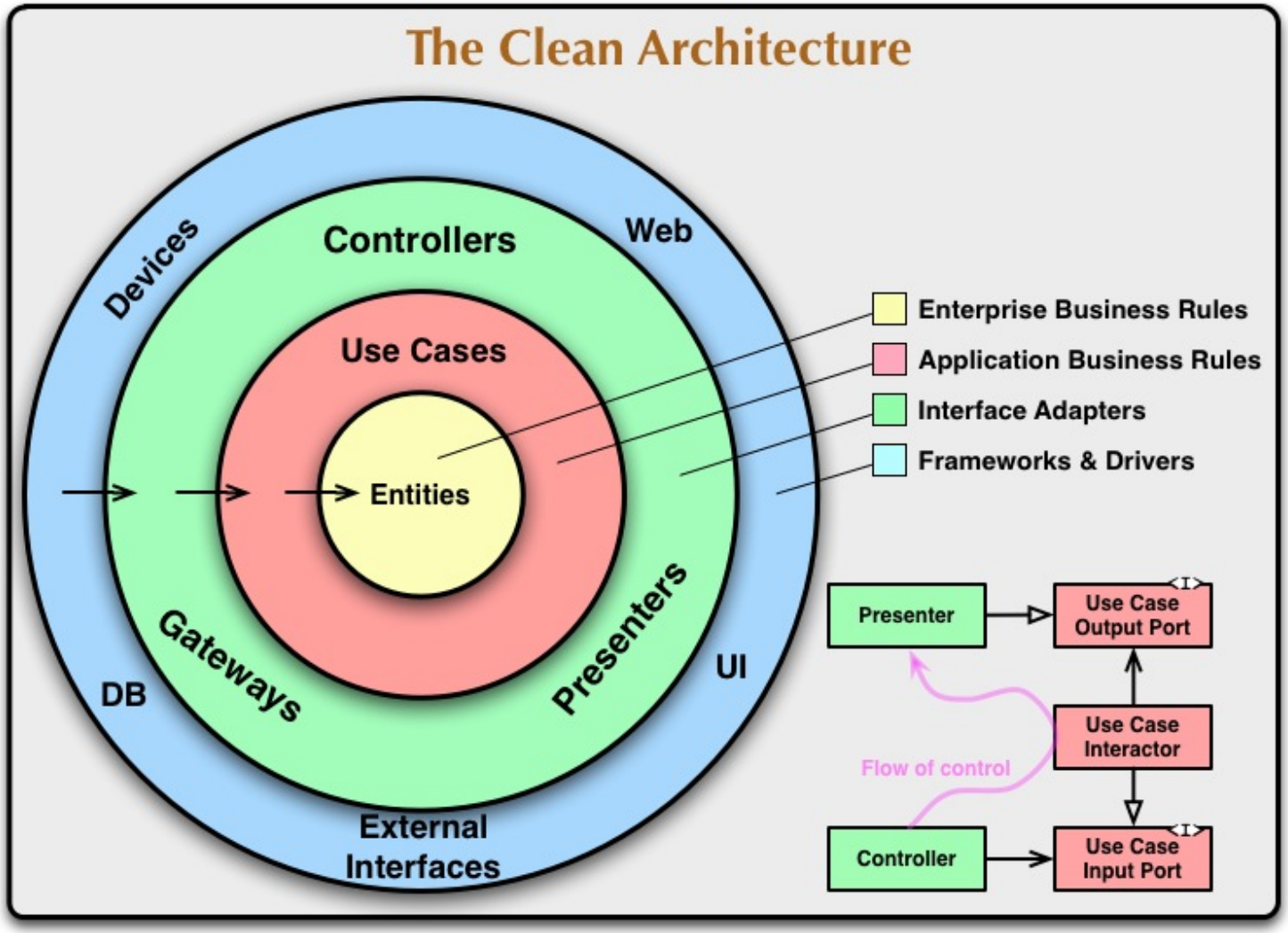
Projecte Integrat de Software (2024/25)

Objectius

Els **objectius** són:

1. Conèixer l'arquitectura neta.
2. Familiaritzar-se amb el principi d'inversió de dependències.
3. Saber aplicar-la en una aplicació d'Android.

Arquitectura Neta



Crèdits

Proposada per Robert C. Martin, prenent idees de:

- **Arquitectura Hexagonal** (o **Ports i Adaptadors**) d'Alistair Cockburn. Font.
- **Arquitectura Ceba** de Jeffrey Palermo. Font.
- **Arquitectura Sorollosa** del mateix Robert C. Martin. Font.
- **Data, Communication, and Interaction (DCI)** de James Coplien i Trygve Reenskaug. Font.
- **Boundary-Control-Entity (BCE)** d'Ivar Jacobson en el seu llibre *Object Oriented Software Engineering: A Use-Case Driven Approach*.

Característiques

- **Estructuració per capes**. Separació de responsabilitats. Al centre el més important i fora els detalls no importants.
- **Identificació de nivells d'abstracció i estabilitat**. Al centre el nivell d'abstracció més alt i el menys propens al canvi.
- **Regla de dependència**. Les capes interiors no depenen de les exteriors (detalls).
- **Testabilitat millorada**. El més important (capes interiors) es poden testejar sense dependre del que no ho és (capes exteriors).
- **Expressivitat del propòsit** mitjançant la definició de classes que representen els casos d'ús[1].

[1] Principal diferència amb l'Arquitectura Ceba

Capa *Enterprise Business Rules*

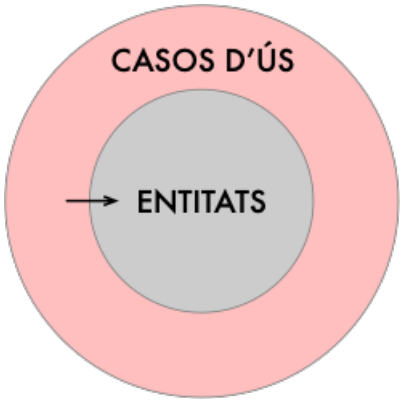
Capa de regles de negoci de l'empresa

- No són específiques de l'aplicació, sinó del negoci.
- El que modelaríeu en un model de domini.
- El que aporta valor al negoci.
- Són els **objectes del domini** de DDD (agregats, entitats, objectes de valor i serveis).

Capa *Application Business Rules*

Capa de regles de negoci de l'aplicació

- Són els **casos d'ús** de l'aplicació.
- Coordinen i deleguen en els objectes del domini.
- Deleguen en altres casos d'ús.
- Interactuen amb els repositoris.
- S'implementen seguint el *patró façana* + una simplificació del *patró comanda*.
- La diferència amb els serveis del domini és que són orientats a l'usuari i, per tant, exposats a la capa d'adaptadors d'interfície. Els serveis no.

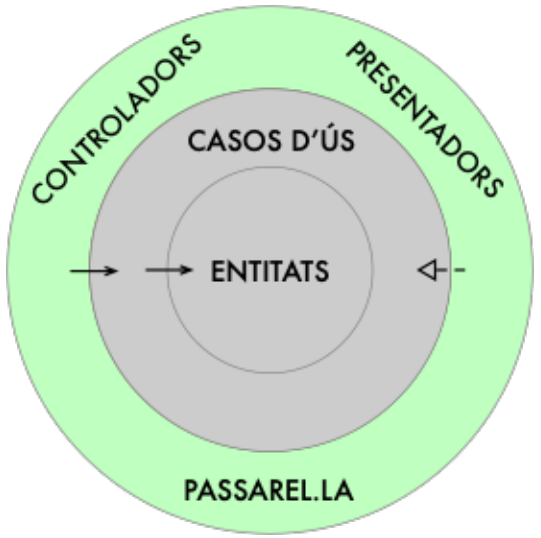


Capa *Interface Adapters*

Capa d'adaptadors d'interfície
Traductors de crides i transformadors de dades entre la capa d'aplicació i la de infraestructura.

Exemples:

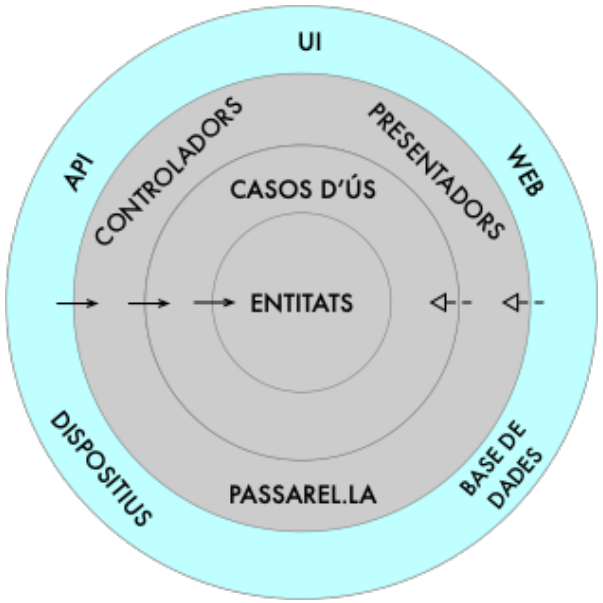
- Controlador que converteix una URL de petició web a paràmetres i crida al cas d'ús que correspongui.
- Presentador que transforma el retorn d'un cas d'ús a una representació presentable per una interfície gràfica.
- Repositori (*Gateway*) que converteix un agregat a *Data Transfer Object* (DTO) abans de demanar-li a la base de dades guardar-lo.



Capa *Frameworks & Drivers*

També anomenada **capa d'infraestructura**

- Implementa els mecanismes d'entrada/de sortida d'informació de/cap a l'exterior.
- Depèn quasi sempre de tecnologies externes (p. ex. *Android* o *Firebase*). Exemple. Invocació de mètodes d'una API externa o codi d'Android per capturar la interacció de l'usuari amb l'aplicació.
- No volem testejar els components d'aquesta capa quan fem TDD perquè alenteix els tests. Tests d'integració, que són menys freqüents, o inspecció visual si es tracta d'UI.



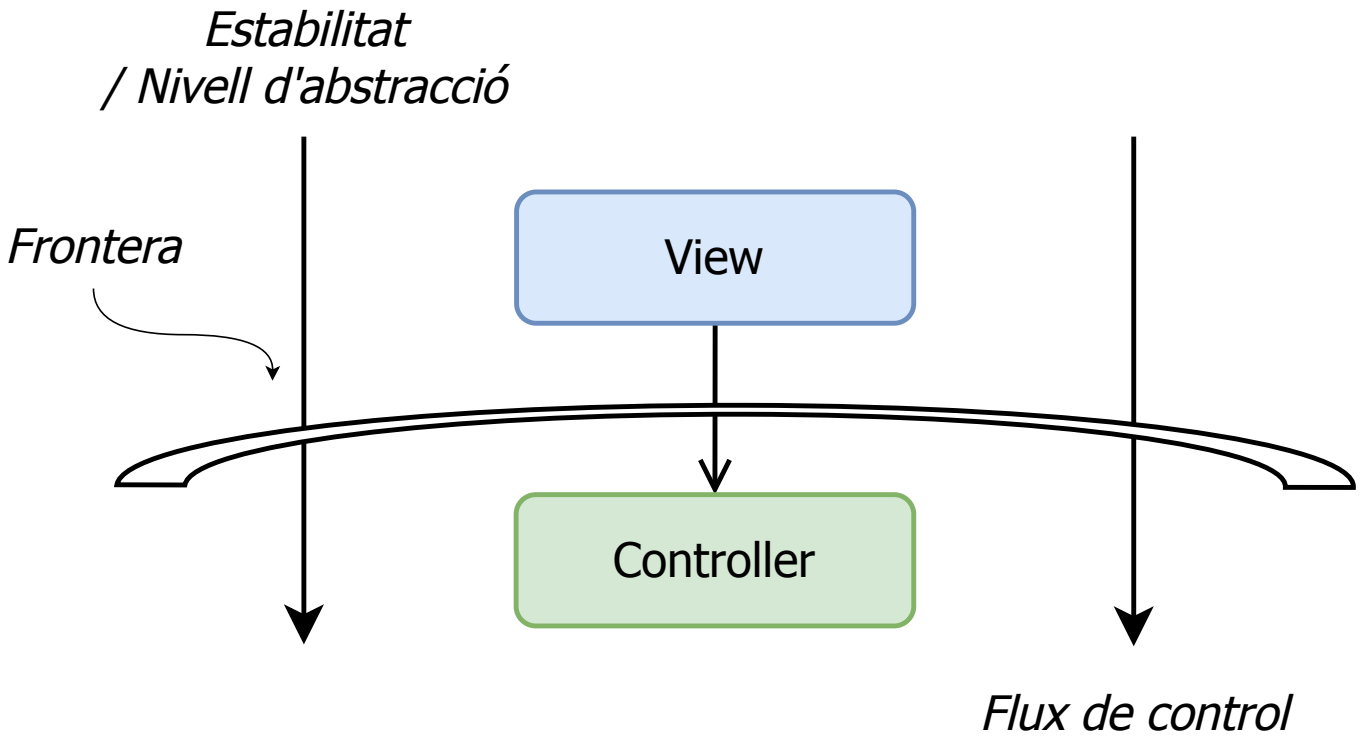
Conceptes clau

Frontera
És la separació entre els components d'una capa i la següent.

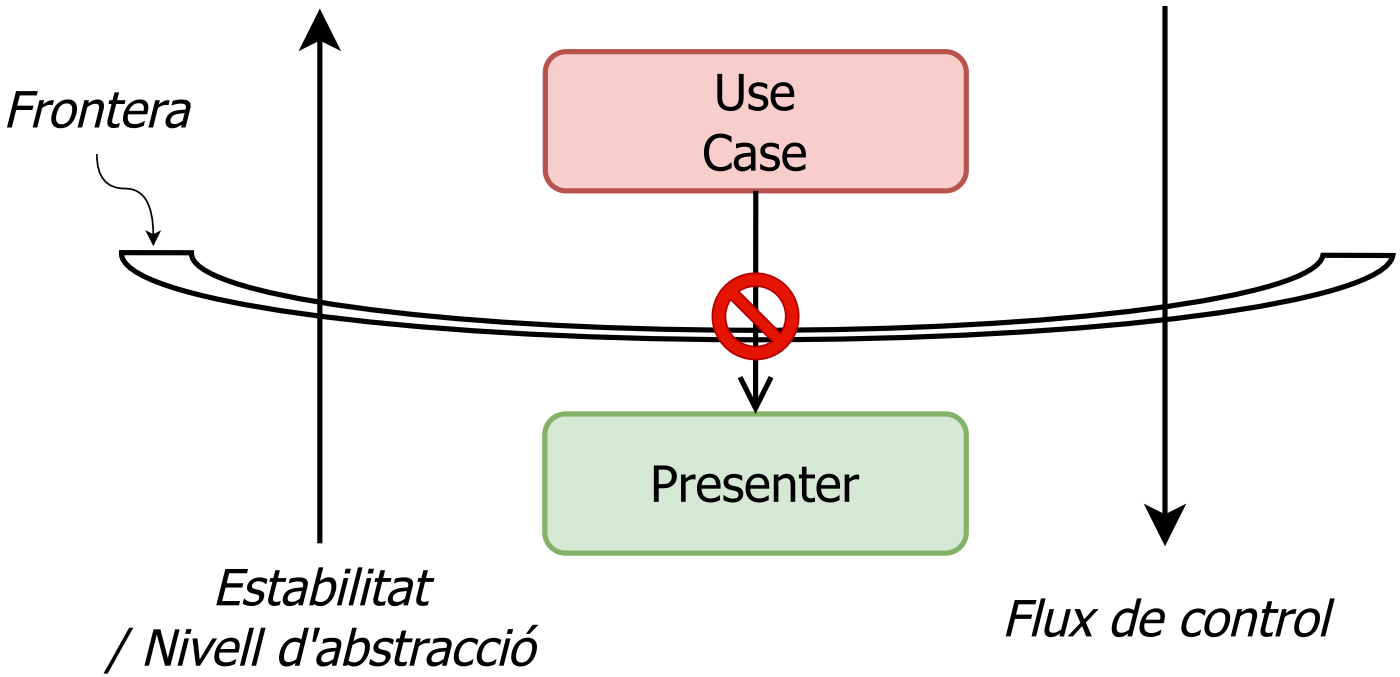
Conceptes clau

Regla de dependència
Quan es travessa una frontera, les dependències han d'apuntar en el sentit del nivell d'abstracció de les polítiques (o, el que és el mateix, de l'estabilitat). O el que és el mateix, de fora cap a dins de les capes circulars concèntriques.

Conceptes clau



Conceptes clau

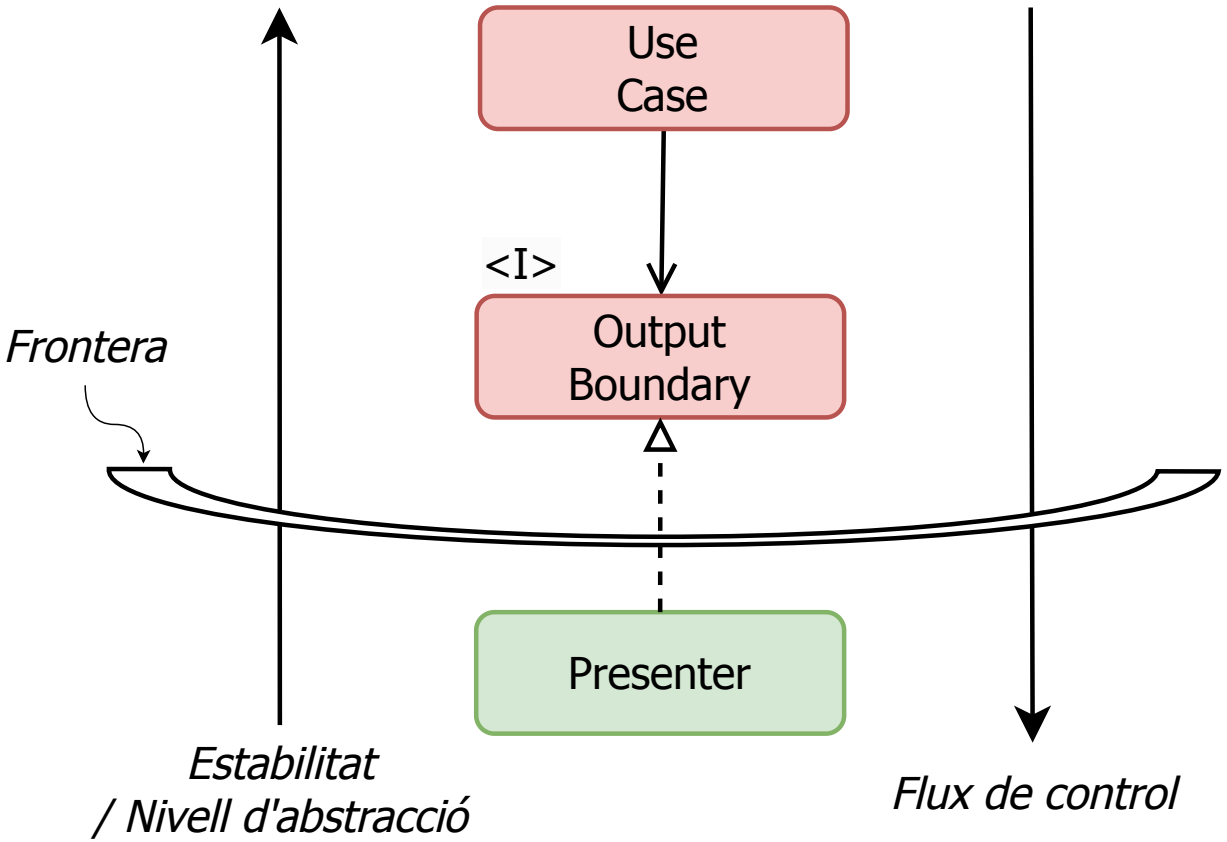


Conceptes clau

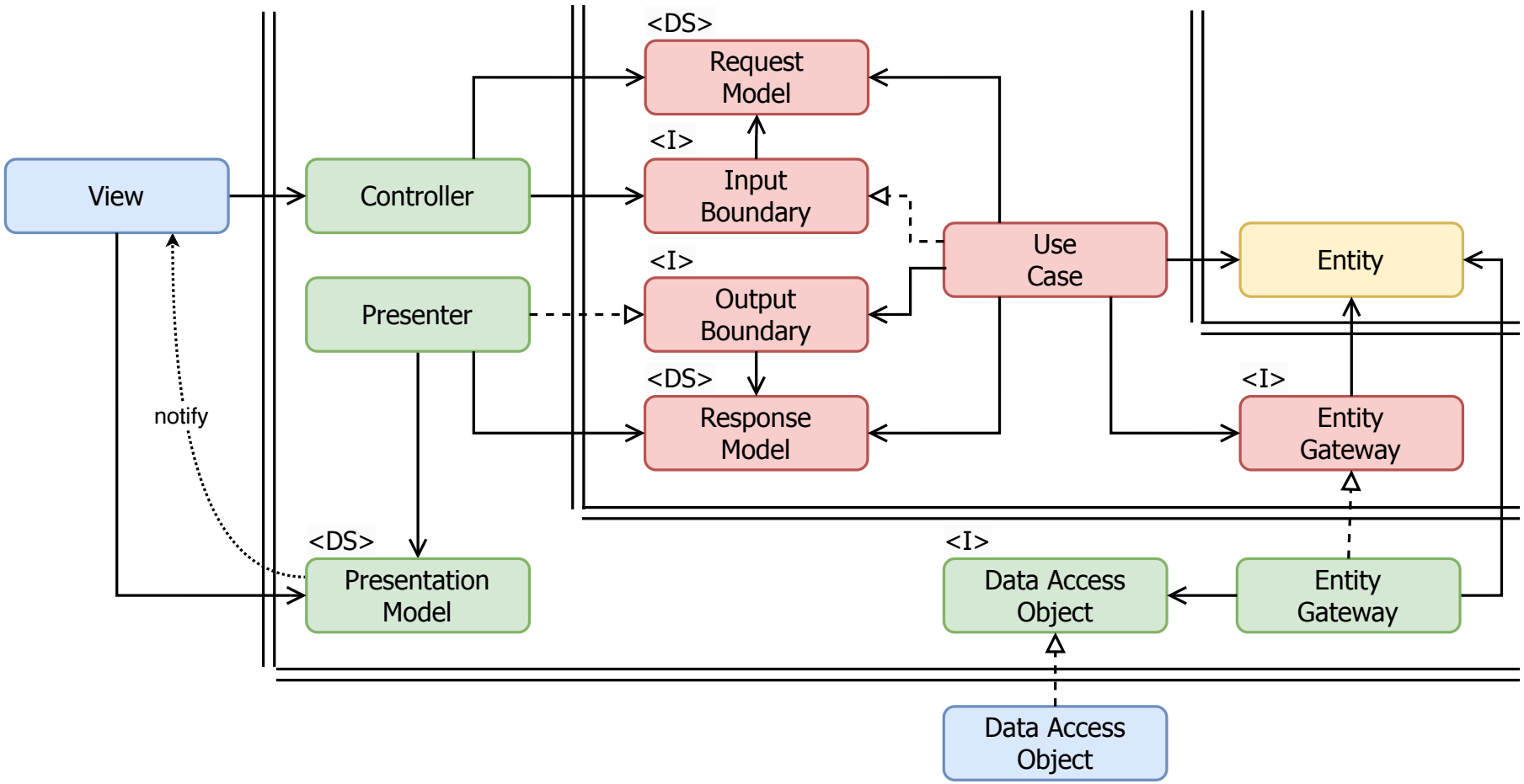
Dependència d'abstraccions
Quan un component de més alt nivell (capa interior) hagi d'interactuar amb un component de més baix nivell (capa exterior), caldrà aplicar *principi d'inversió de dependències* (DIP) per invertir-les i mantenir el sentit del flux de control.

Conceptes clau

Aplicació del principi d'inversió de dependències (DIP)



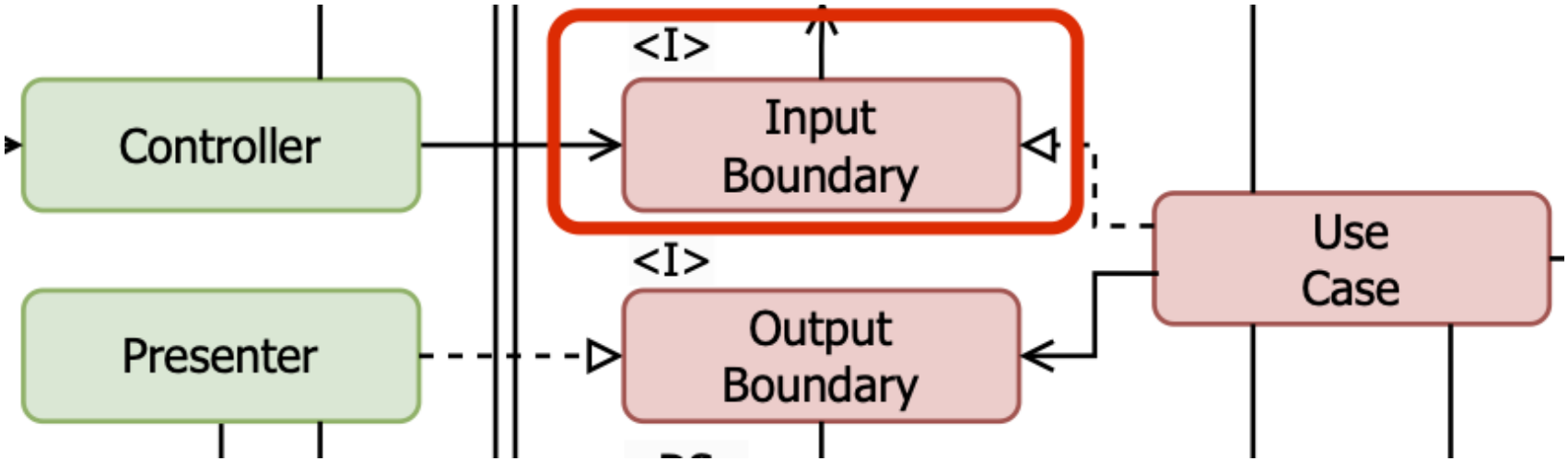
Arquitectura Neta



||: frontera, `<I>`: interfície, `<DS>`: data structure

Arquitectura Neta

Per què aquesta abstracció?



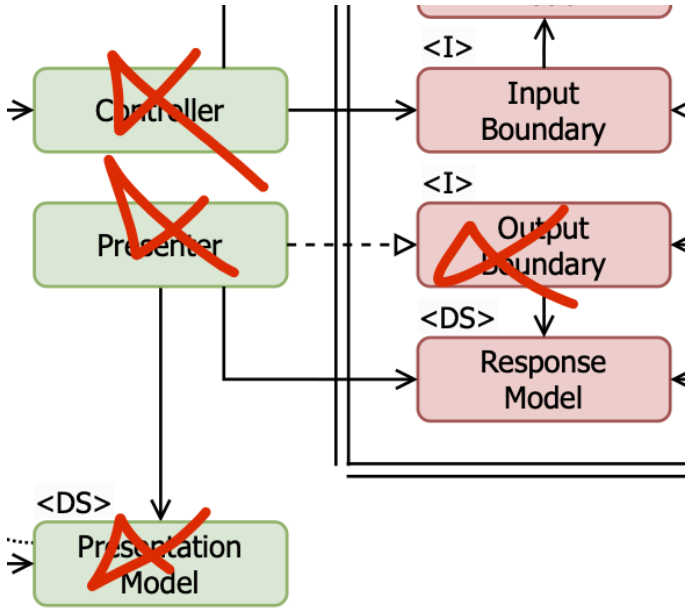
Arquitectura Neta

- Els marcs de treball externs (p. ex. Android) poden causar variacions en les capes més exteriors (infraestructura i adaptadors d'interfície).
- Aplicar-la fil per randa requereix disciplina.
- No aconsellable per projectes petits. Però si no ho feu ara, quan ho fareu?
- Pel projecte, adaptarem l'arquitectura pel context d'Android (a les capes exteriors) i hi farem simplificacions.

Arquitectura Neta pel projecte

Simplificacions i adaptacions:

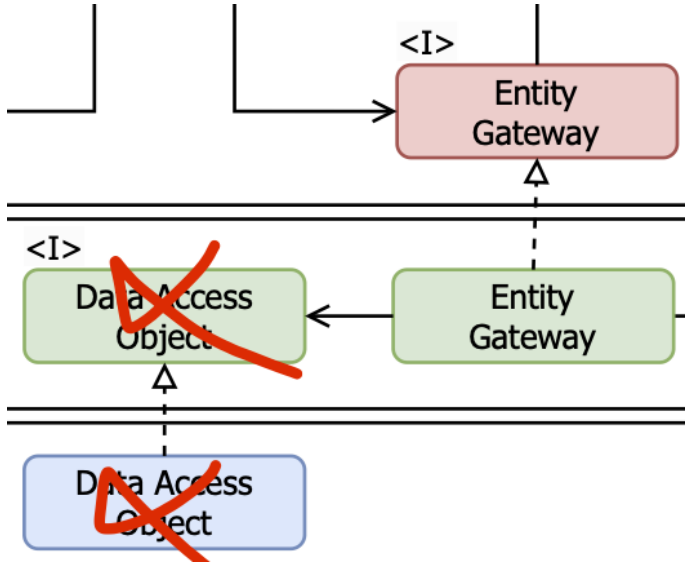
- L'únic component d'adaptador de interfície seran els models de vista, que fan el paper de "Controller" i "Presentation Model" alhora.
- Sense "Presenter" no caldrà tampoc la interfície "Output Boundary".



Arquitectura Neta pel projecte

Simplificacions i adaptacions:

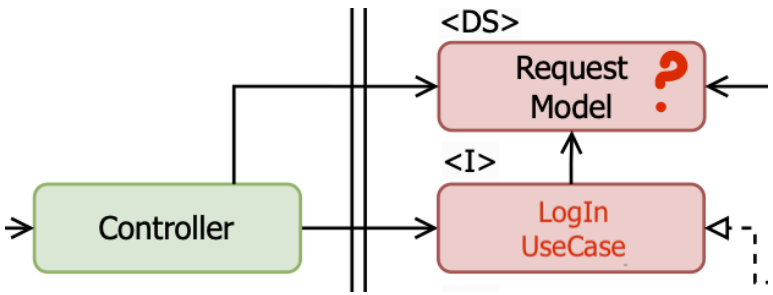
- Els Entity Gateway seran repositoris i la seva implementació serà directament de la capa d'infraestructura perquè dependrà de la API de *Firestore*.
- Sense "Data Access Objectes (DAO)" gràcies a la capacitat de *Firestore* de guardar/carregar agregats directament.



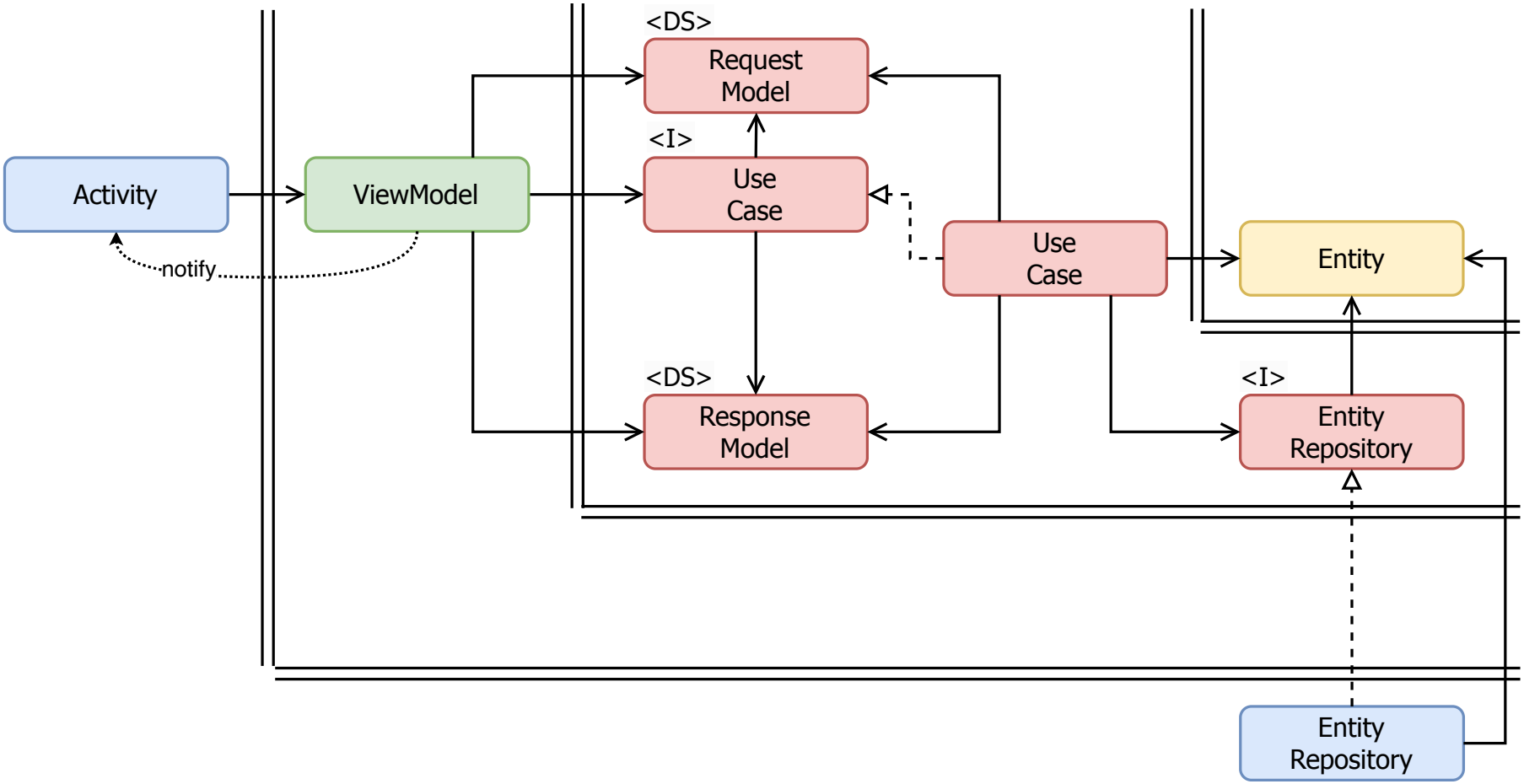
Arquitectura Neta pel projecte

Simplificacions i adaptacions:

- La interfície genèrica "Input Boundary" la convertirem en una interfície específica del cas d'ús, amb el nom del cas d'ús (p. ex. `interface LogInUseCase`).
- Els "Response Model" seran necessaris per no exposar objectes del domini, però els "Request Model" opcionals.



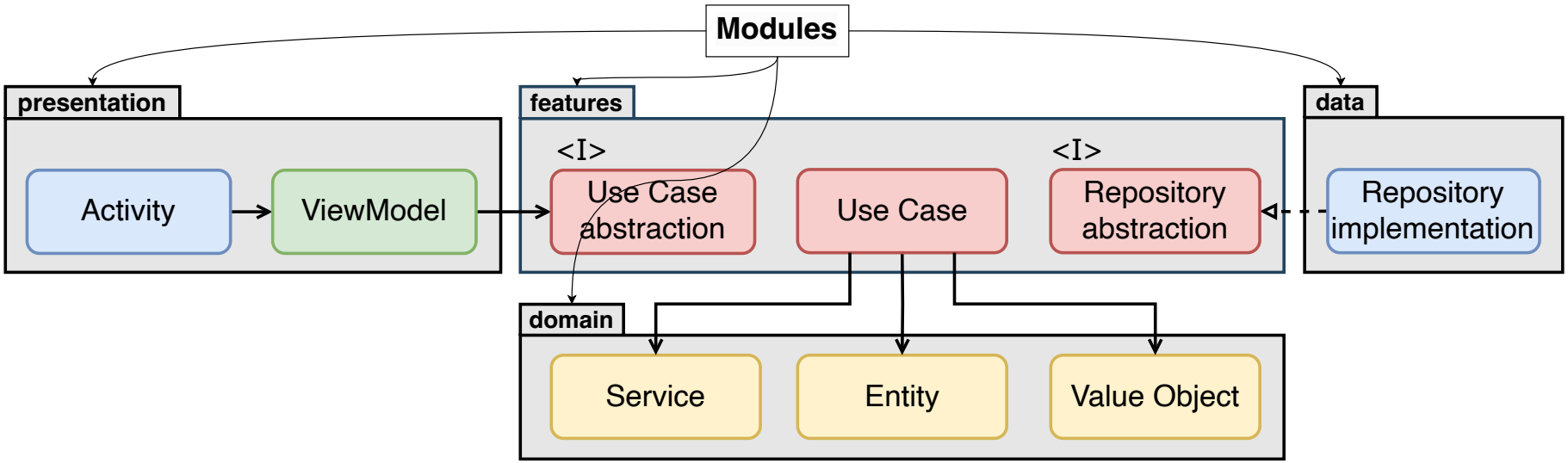
Arquitectura Neta pel projecte



||: frontera, `<I>`: interfície, `<DS>`: data structure

Arquitectura Neta pel projecte: mòduls

Definirem mòduls perquè ens forcen, explícitament, a afegir dependències d'altres mòduls. Amb els paquets podríeu vulnerar la regla de dependència sense adonar-vos-en.



Creació de mòduls i organització de l'Arquitectura Neta a la PI6, que haureu d'aplicar pel projecte.