Approche Objet - Modèle de Conception

Marie Beurton-Aimar
et
Xavier Blanc

November 16, 2021

Repository et couche Infra

- La couche Infra ? quelle utilité ? quelle solution aux problèmes de la séparation des responsabilités (separation of concerns) ?

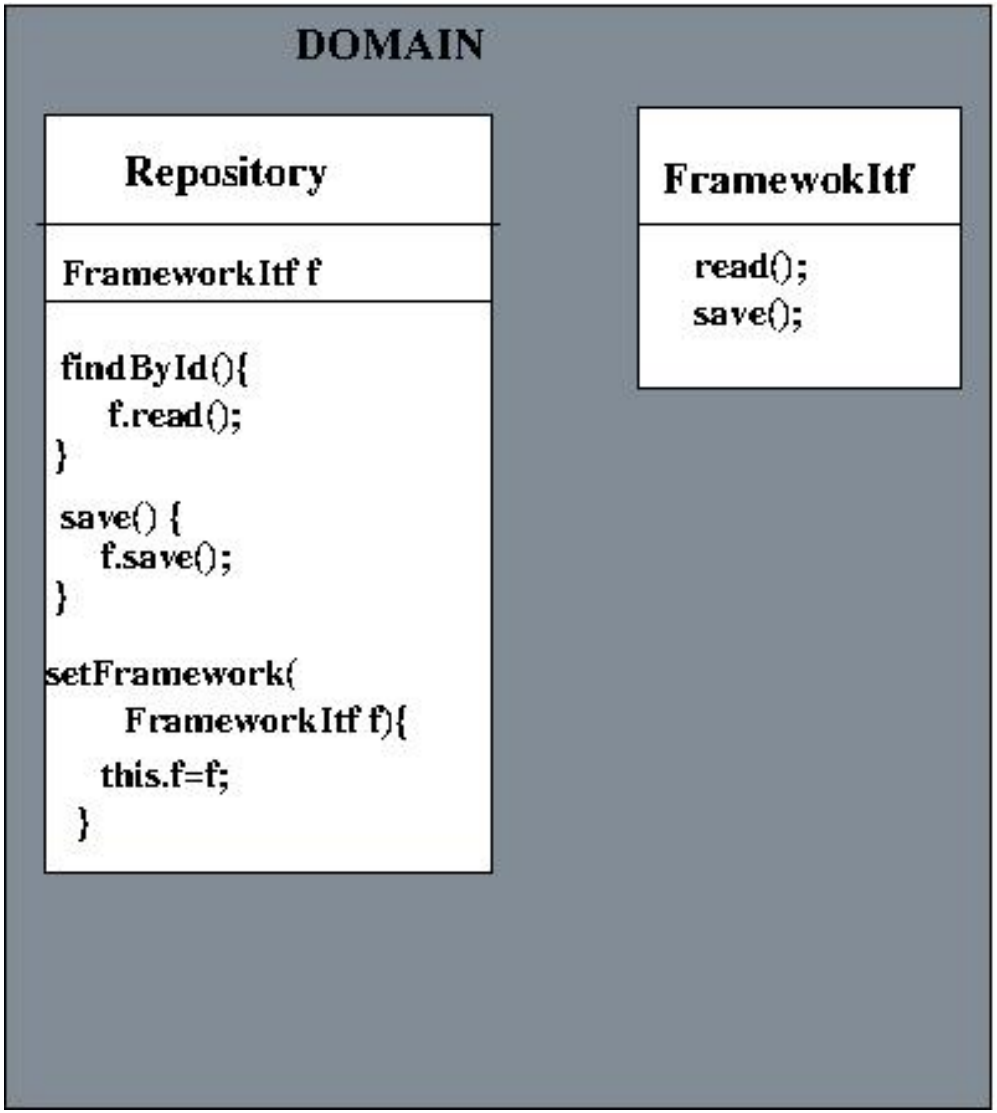
Repository et Code Technique

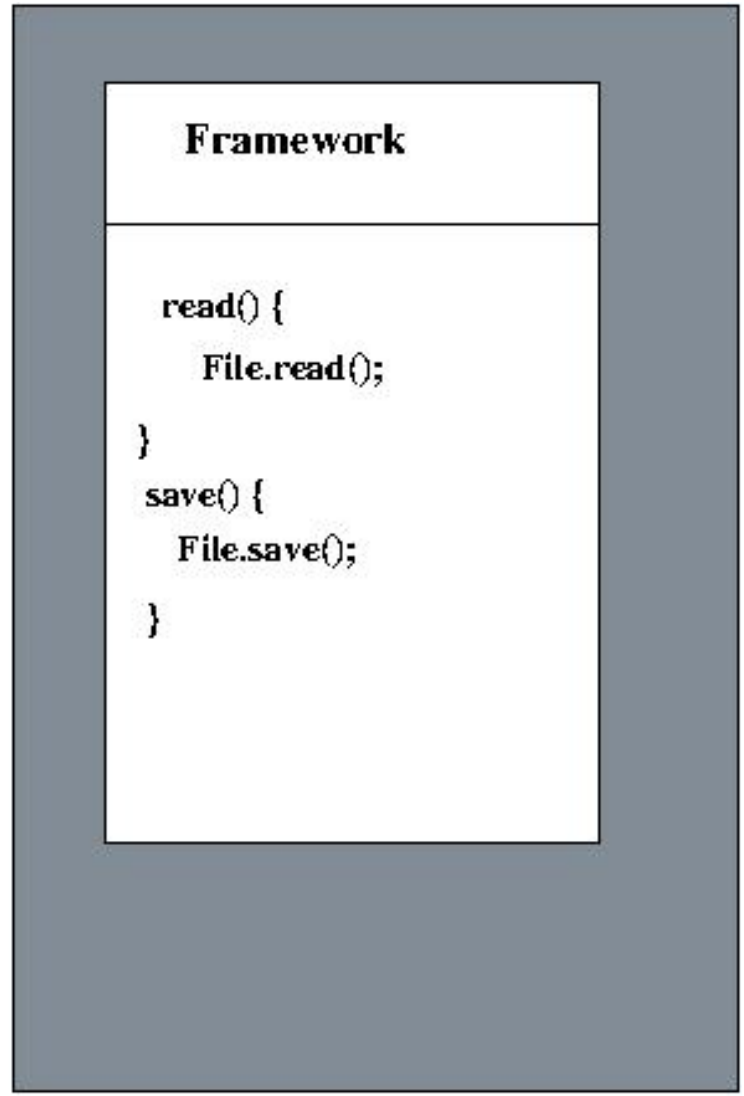
- Le Domaine doit rester abstrait.
- Exemple précédent : le jeu d' échecs - le Repository contenait du code technique.

```
public class GameRepositoryFile{
  public Game findGameById(GameId gameId){
    File f = new File(gameId);
    Game g = new Game(f.read() ...) ;
    return g;
  }
}
public void save(Game g){
  File f = new File(g.gameId);
  f.write( g ...);
}
```

- La couche Domain ne devrait pas dépendre du framework technique.
- Corollaire : Le framework technique doit être au service du Domain.

Interface et inversion de dépendance

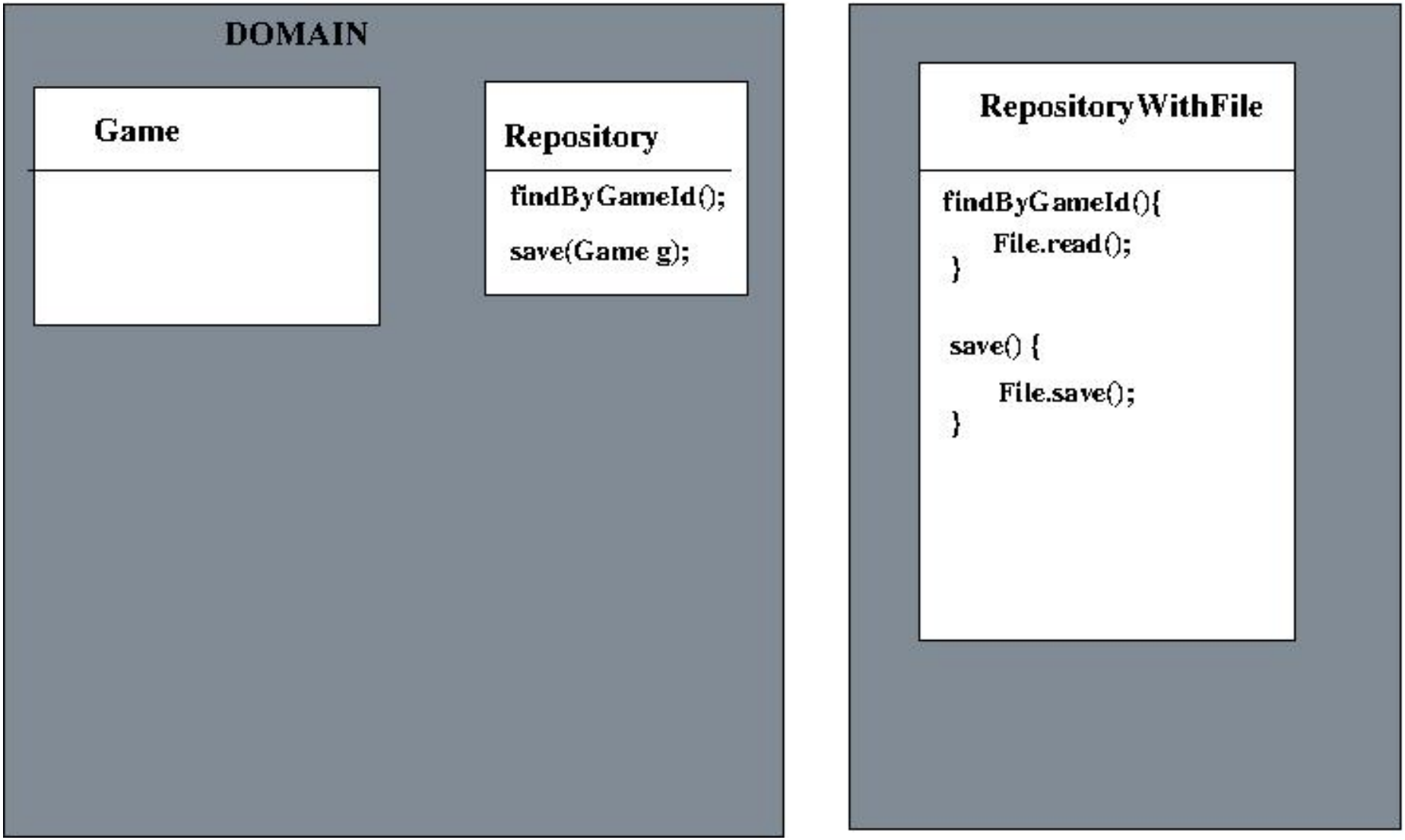




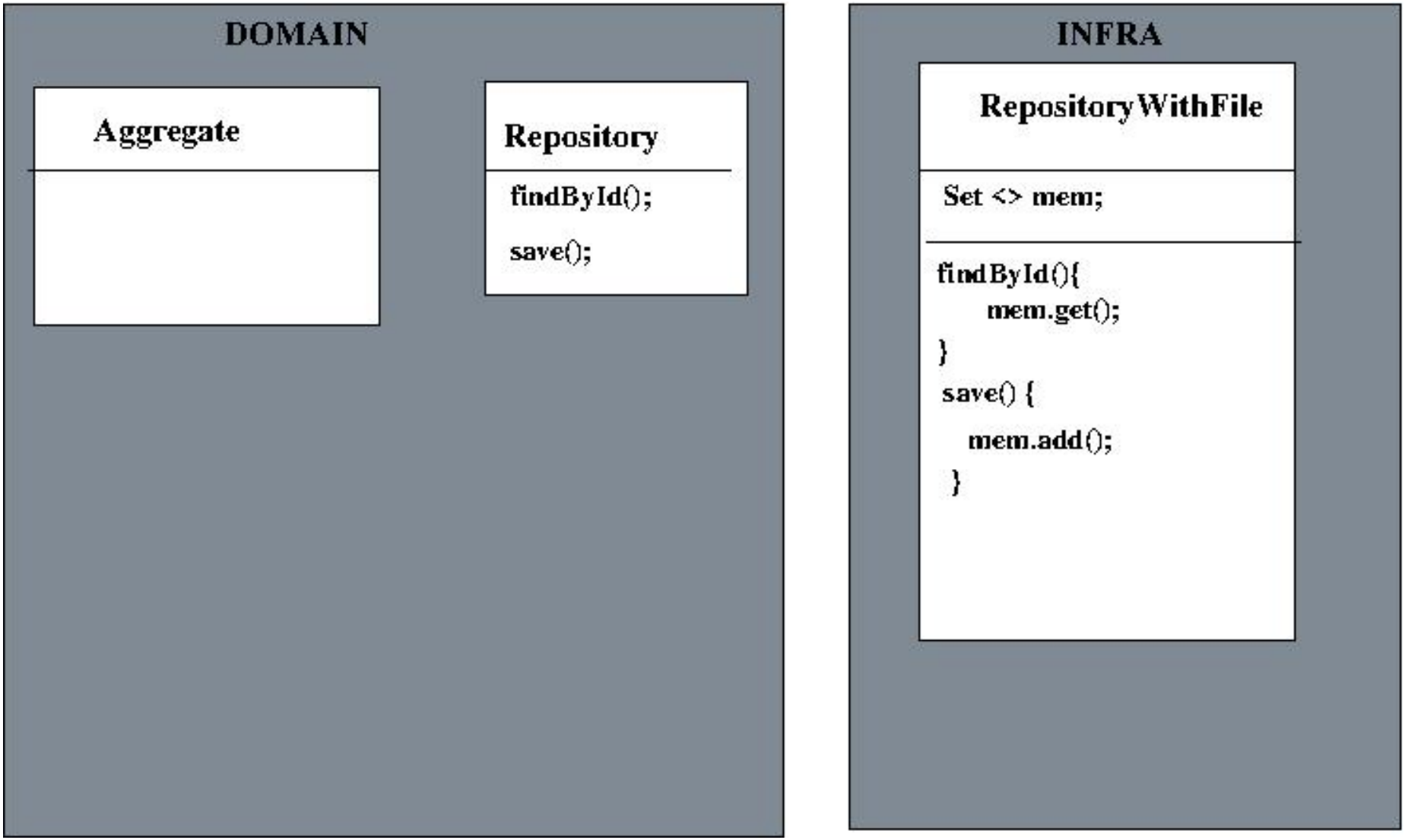
Interface et inversion de dépendance

- L'exploitation d'une interface permet d'inverser les dépendances.
  - Définir le framework par une interface :`save()` `load()`
- L'appelant ne dépend que de l'interface :
  - L'interface appartient à la couche de l'appelant.
- L'appelé réalise l'interface :
  - L'appelé est dans une autre couche.
- A l'exécution on devra lier l'appelant à celui qui réalisera l'interface.

Repository concret



Repository concret en mémoire



La Couche Infrastructure

- Cache tout ce qui est volatile, ou qui peut changer en fonction de la technologie, du déploiement, du système d' exploitation, de la méthode de transport des données ...
- Prend en charge la conversion des données en provenance d'autres domaines.
- Important : il est important de disposer d'infrastructure qui permette la réutilisation de fonctionnalités communes pour accélèrer le développemnent de nouvelles applications :
  - Persistance
  - Logging, messaging
  - Domain events

La Couche Application

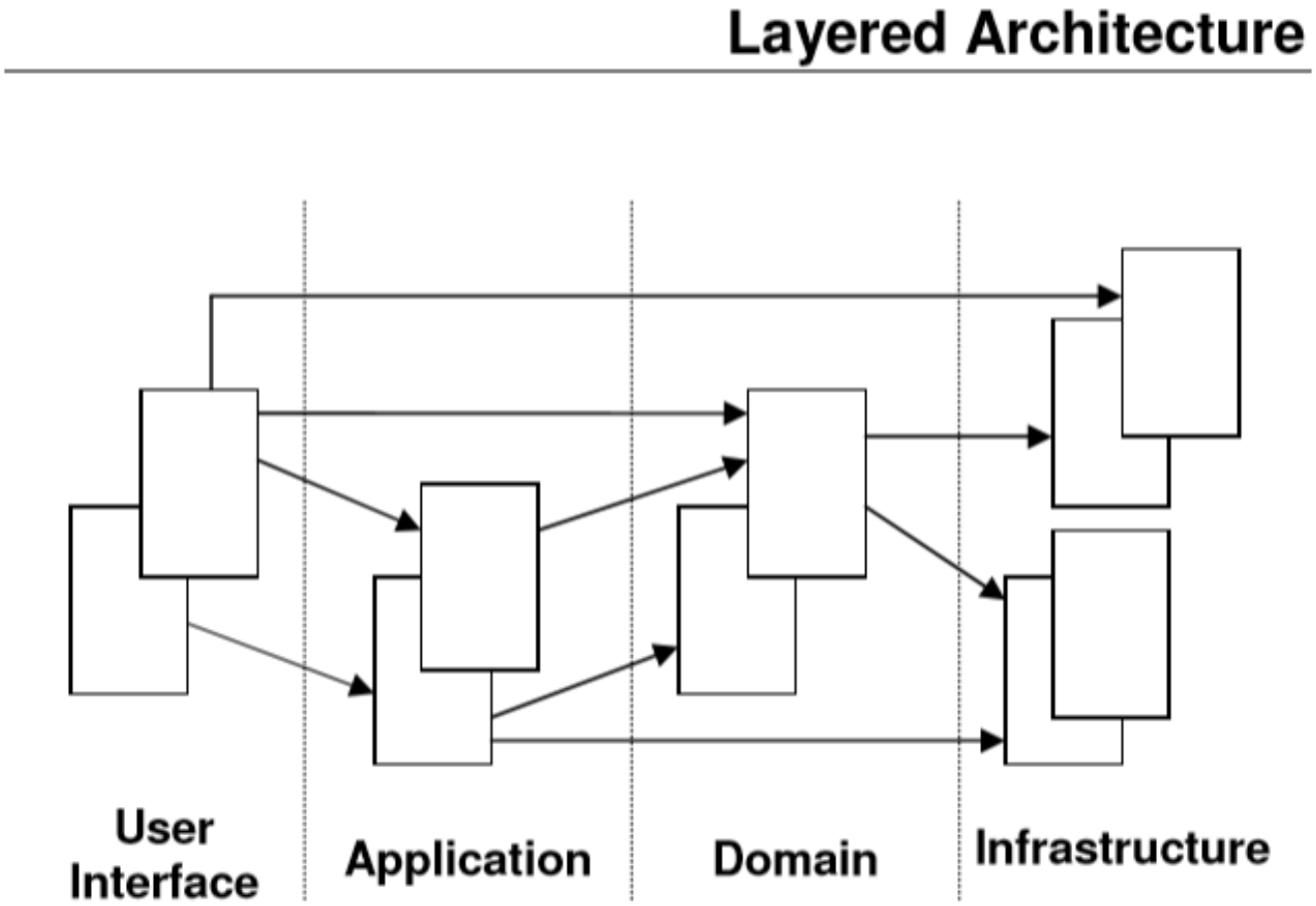
- Une vision orientée **service** - sans état du Domain.
- Objectif : éviter que le client / User Interface manipule directement les `Aggregate` qui sont les points d'accès au `Domain`.
- La couche `Application` va s'occuper de définir les tâches à effectuer pour réaliser une tâche d'application donnée.
- Elle est responsable du mandat du travail du `Domain` nécessaire et interagit avec d' autres services - externes ou non.

La Couche Application

- Example : Application financière.
  - Implémentation de la capacité d'un utilisateur particulier à approuver une opération.
  - Quand cette opération est validée une entité du modèle change d' état (approuvée0).
  - D'autres acteurs doivent être informés par un message.
  - La couche `Domain` n' a pas à prendre en charge le service de messagerie.
- La couche `Application` est une couche d'isolation : les comportements de cette couche ne doivent pas être affectés par un changement de code dans les autres couches : `Domain`, `Infra`.

Couche UI

- Aussi appelée couche *Présentation* dans les traductions en francais.



Couche UI

- Accède à la couche `Application`
- Manipule les types rendus visibles par la couche `Application`.
- Autant de `UI` que de moyens d'accès à l'`Application` : interface web, interface graphique ...
- Exemple du jeu d'échecs graphiques :
  - les objets graphiques correspondant pièces ?
  - sont une vue des pièces,
  - pas métier, pas infra, pas application

Les Données de l'UI

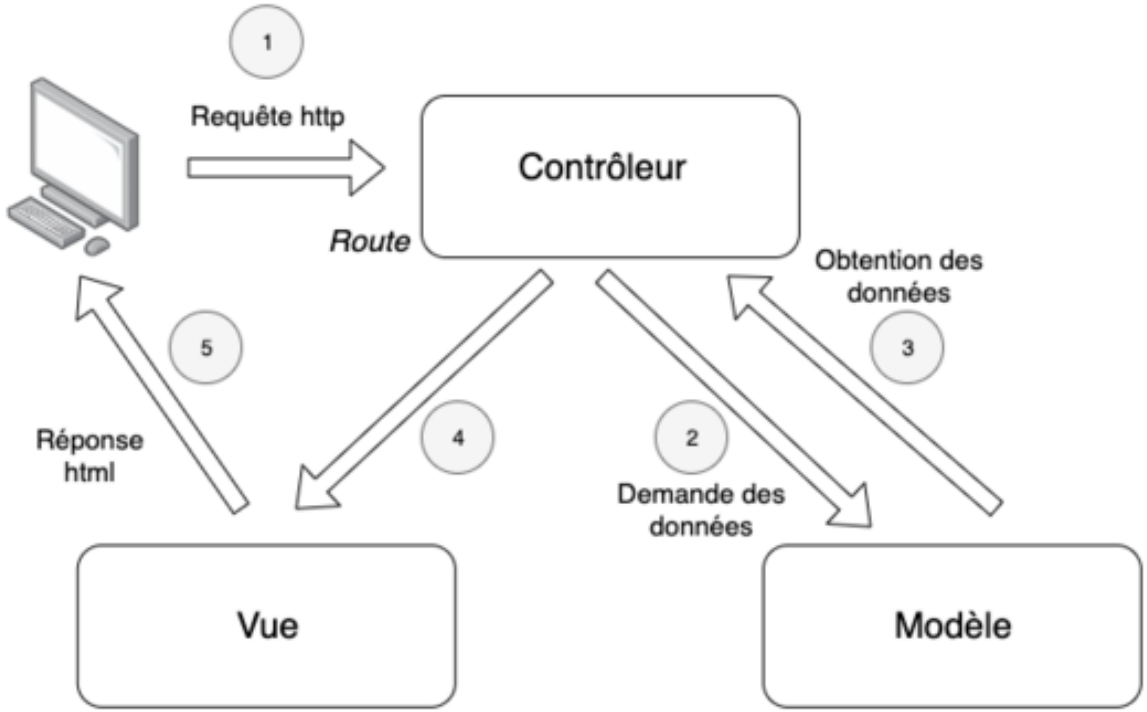
- L'UI dispose de ses propres données.
- Certaines données sont liées avec les données métier : localisation, pieces ....
- D'autres sont complètement indépendantes : style des pièces, de l'échiquier ...

Les interactions de l' UI

- L' UI propose des interactions utilisateurs,
  - Action que l'utilisateur peut faire.
  - Animation que l'utilisateur peut voir, entendre, toucher ....

Le Modèle MVC

- Séparation du modèle, de la vue et du controleur.
- La vue est visible.
- Le controleur gère les interactions.



- Le modèle contient les données abstraites de l'UI.

Les Challenges de l' UI

- Objectifs :
  - Etre réactifs (¿30ms)
  - Offrir une expérience utilisateur.
- Points d' attention :
  - Synchronisation avec le back
  - Optimisation des performances.