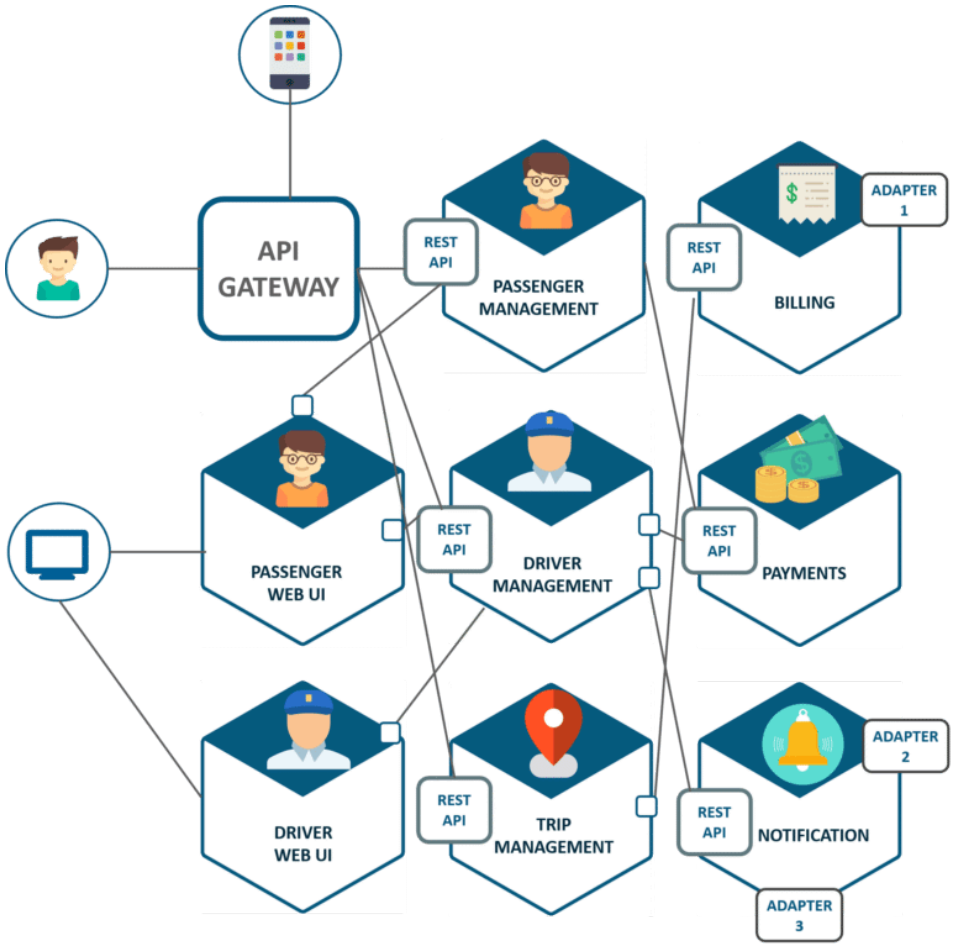
ALOM

🌯 CLEAN ARCHI ET ARCHI HEXAGONALE

Salade, Tomates, Architecture Oignons

UBER


API
GATEWAY
REST
API
PASSENGER
MANAGEMENT
REST
API
BILLING
ADAPTER
1
PASSENGER
WEB UI
REST
API
DRIVER
MANAGEMENT
REST
API
PAYMENTS
DRIVER
WEB UI
REST
API
TRIP
MANAGEMENT
REST
API
NOTIFICATION
ADAPTER
2
ADAPTER
3

PROBLÉMATIQUES :

Comment organiser le code ?

Comment être capable de changer de type de BDD ?

Minimiser le *Blast-Radius* 💣💥 d'un changement

Séparer les responsabilités

L'ARCHITECTURE LOGICIELLE

- Donner forme à la construction d'un système
  - Division en composants
  - Arrangement des composants
  - Communication des composants
- Le but :
  - Faciliter le développement
  - Faciliter le déploiement
  - Faciliter les opérations
  - Faciliter la maintenance

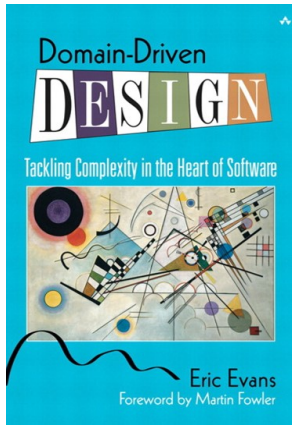
LES CONCEPTS DES ARCHITECTURES LOGICIELLES

- Separation of concerns
  - La BDD ne doit pas dépendre des IHM
  - Les règles métier ne doivent pas dépendre de la BDD
  - Les règles métier ne doivent pas dépendre des IHM

👮 S.O.L.I.D PRINCIPLES

- S : Single Responsability
- O : Open/Closed
- L : Liskov Substitution
- I : Interface Segregation
- D : Dependency Inversion

DOMAIN-DRIVEN DESIGN (DDD)

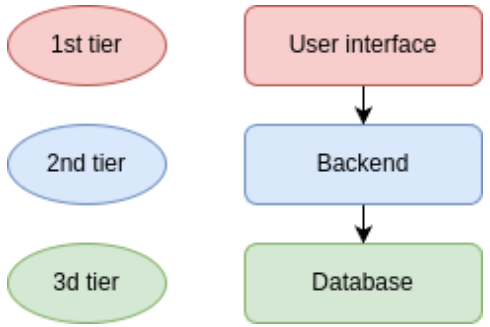


Les archis modernes prennent souvent racine dans les concepts de DDD.

- Domaine Métier : Le cœur de l'application est la logique métier.
- Ubiquitous Language : Un langage commun entre les développeurs et les experts métier.

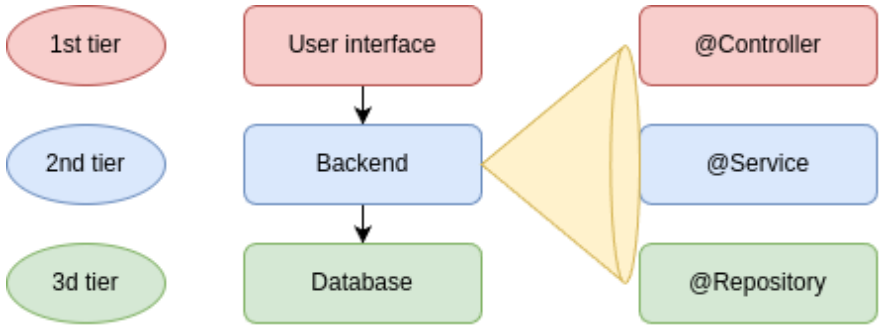
L'ARCHITECTURE N-TIER (ANNÉES 90)

Séparation physique entre les programmes et les BDD, entre les clients et les serveurs



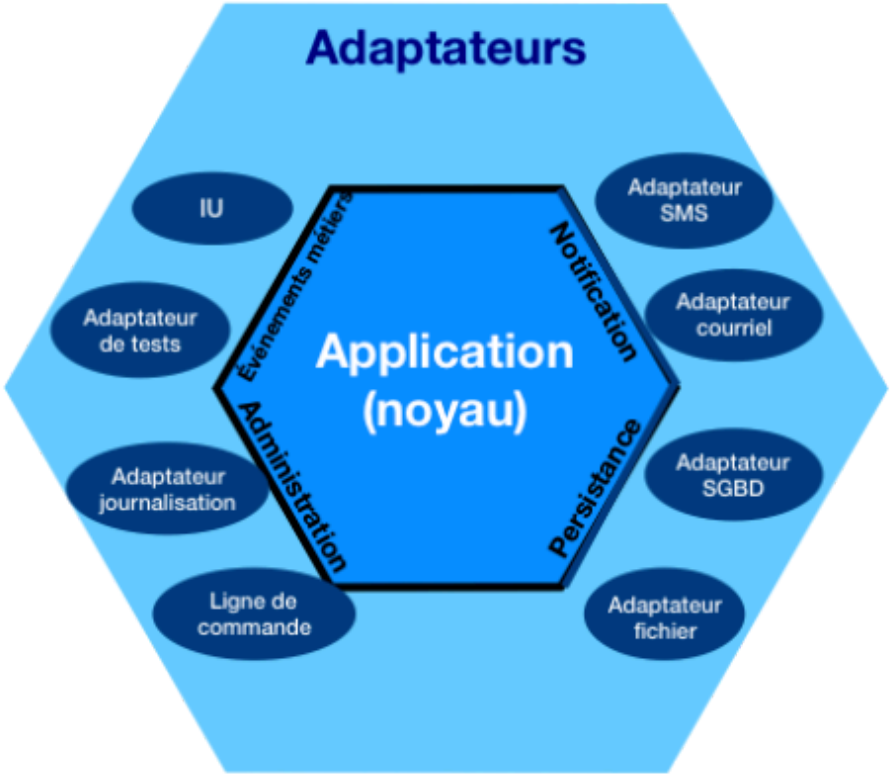
L'ARCHITECTURE N-TIER AVEC SPRING

La vision "classique"



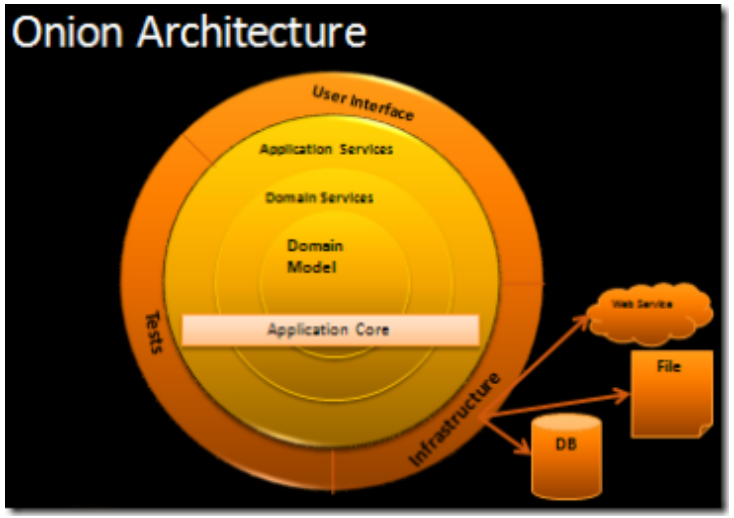
L'ARCHITECTURE HEXAGONALE (PORTS ET ADAPTERS)

Conceptualisé par Alistair Cockburn en 2005



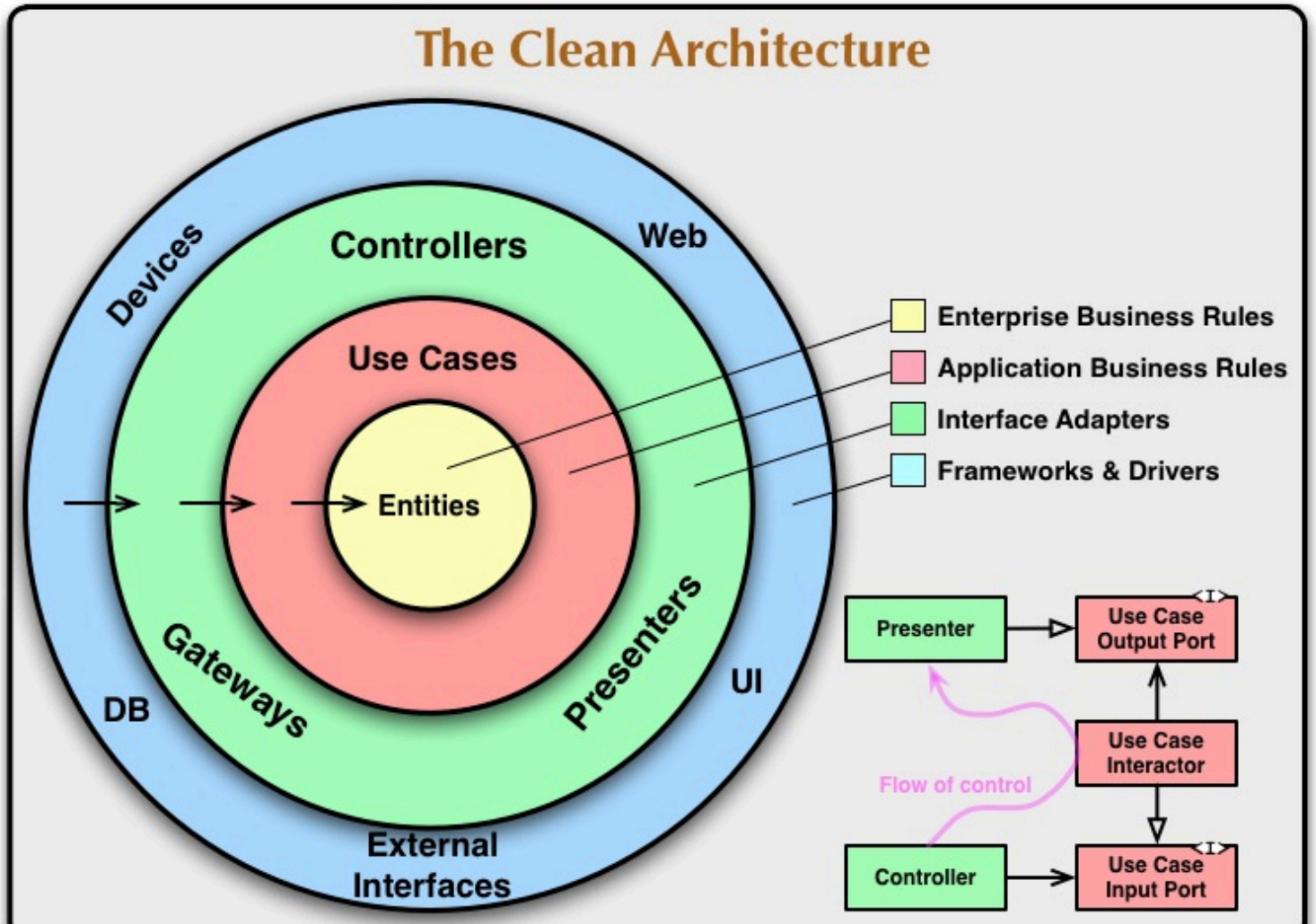
LA ONION ARCHITECTURE

Conceptualisé par Jeffrey Palermo en 2008



LA CLEAN ARCHITECTURE

Conceptualisé par Robert C. Martin en 2012

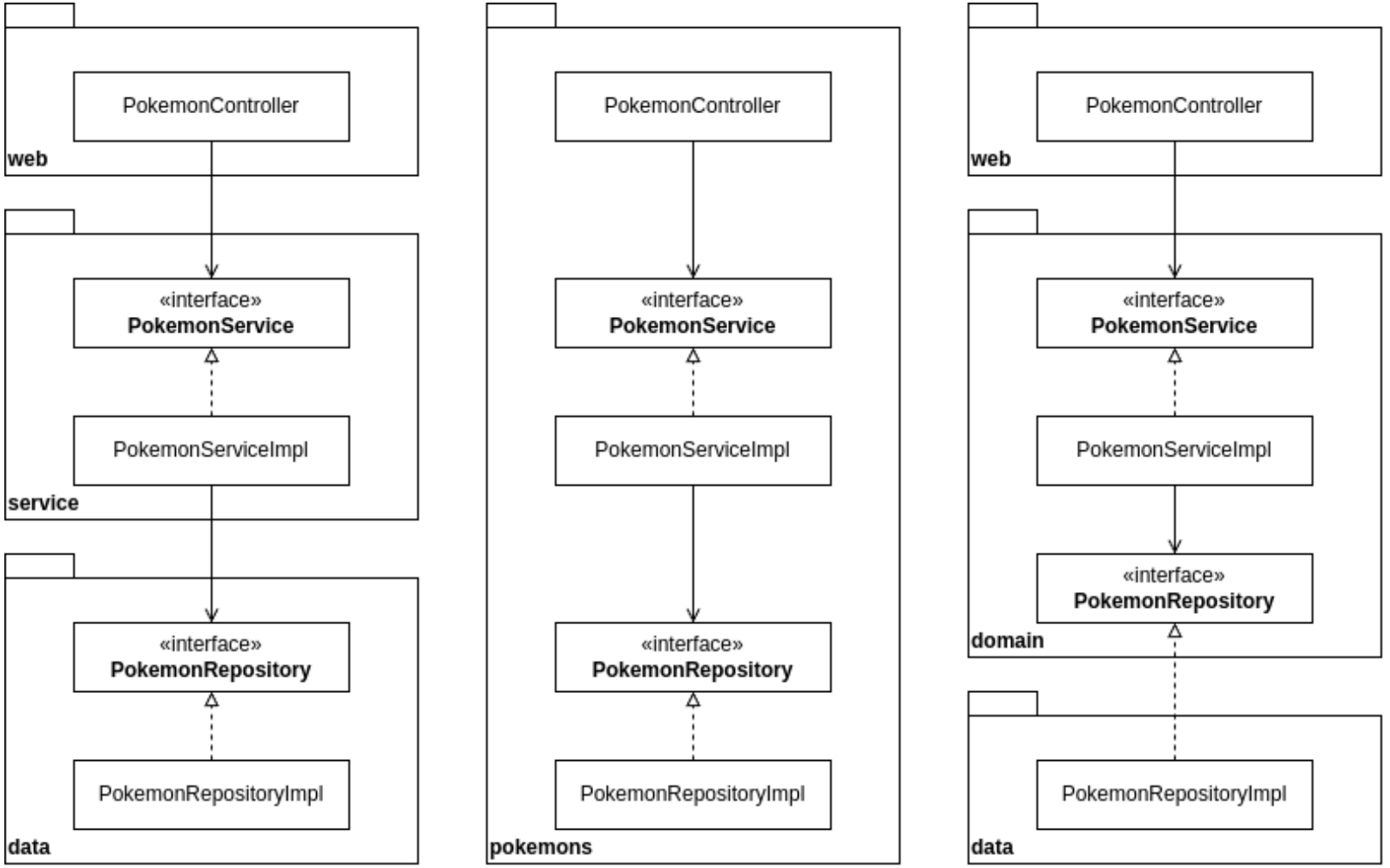


LES ÉLÉMENTS COMMUNS (ET IMPORTANTS)

- Séparer la logique métier des détails techniques
- Indépendance d'un framework
- Testabilité
- Indépendance de la partie UI
- Indépendance de la partie BDD

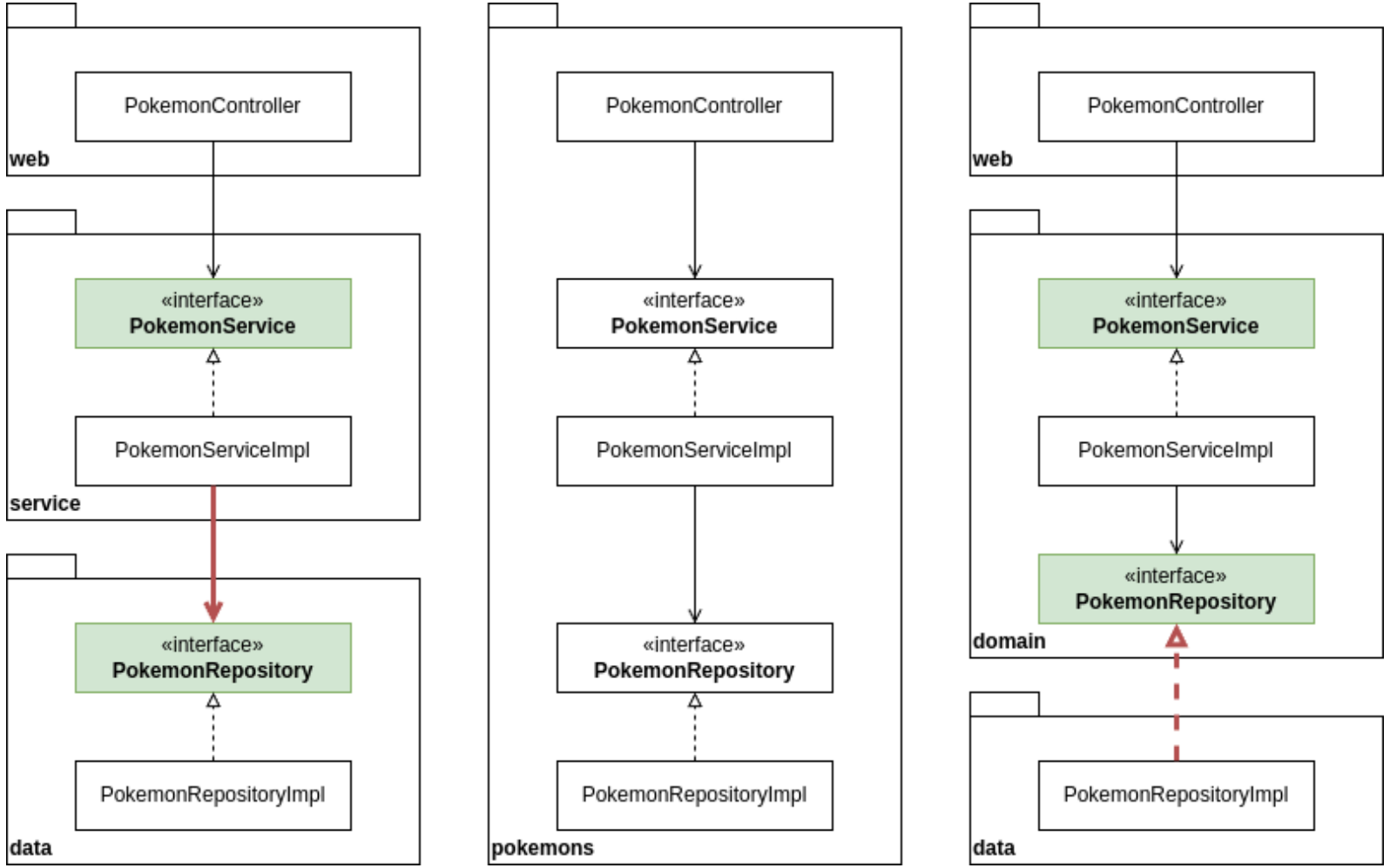
LES MOYENS DE MISE EN PLACE

PACKAGES JAVA

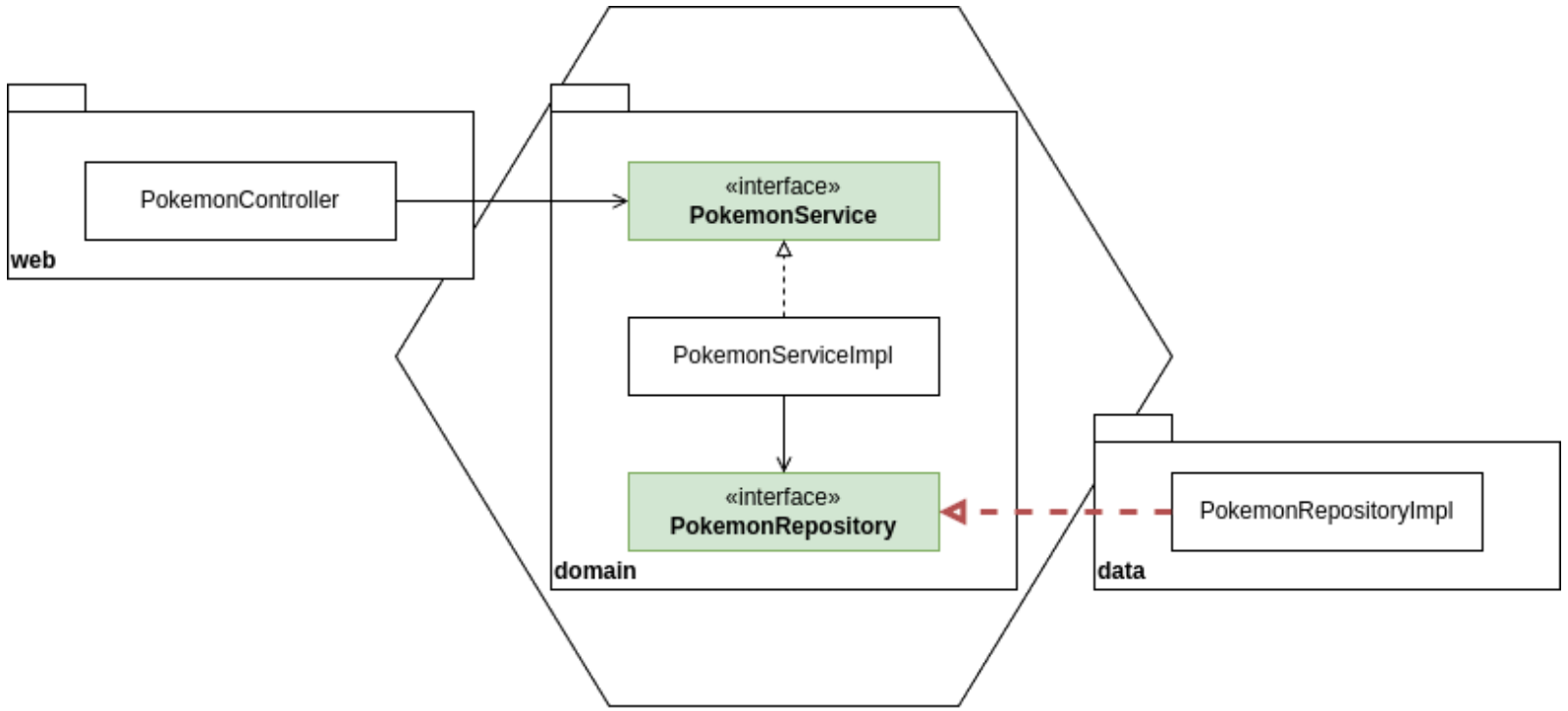


LES MOYENS DE MISE EN PLACE

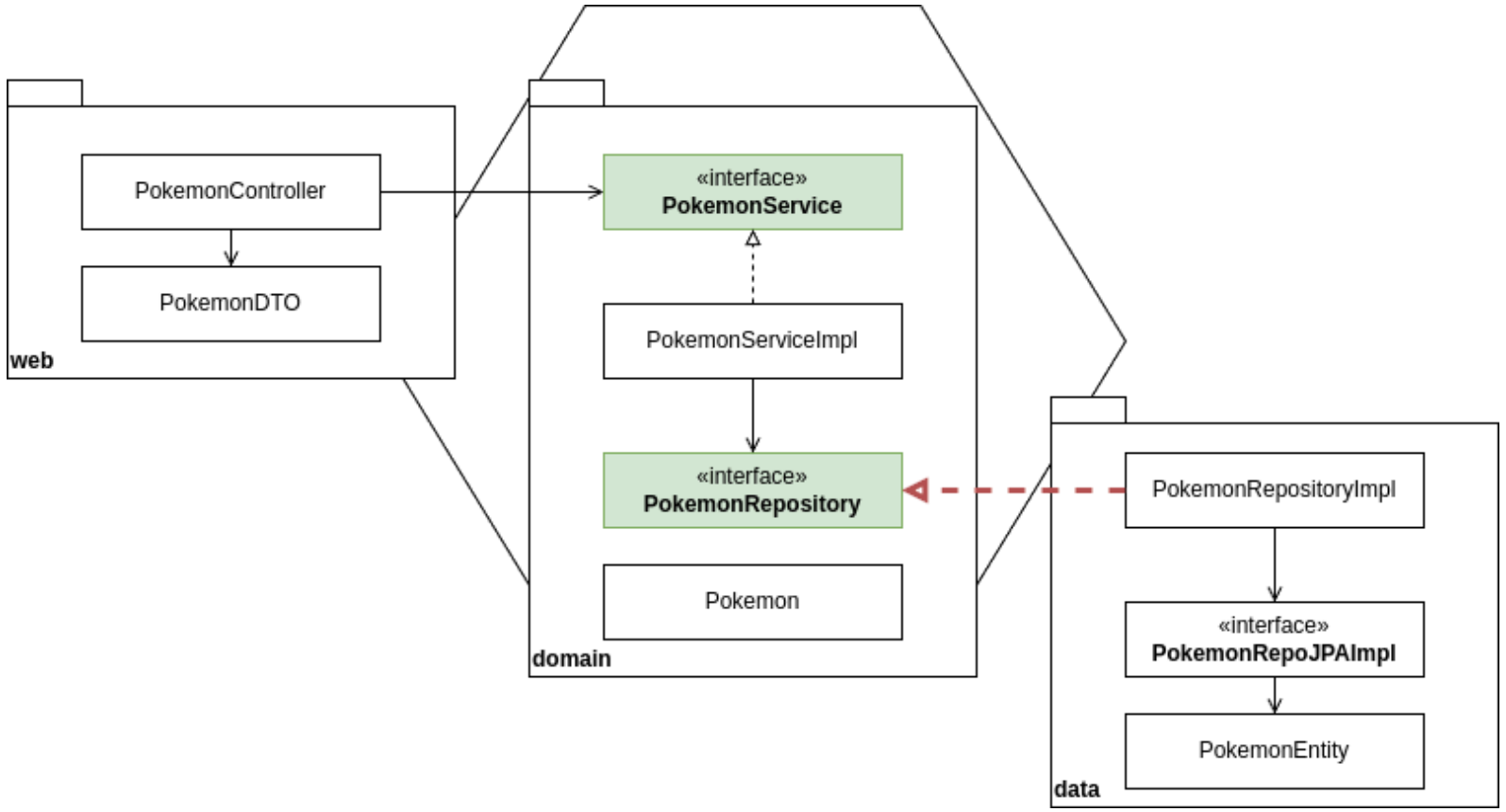
PACKAGES JAVA



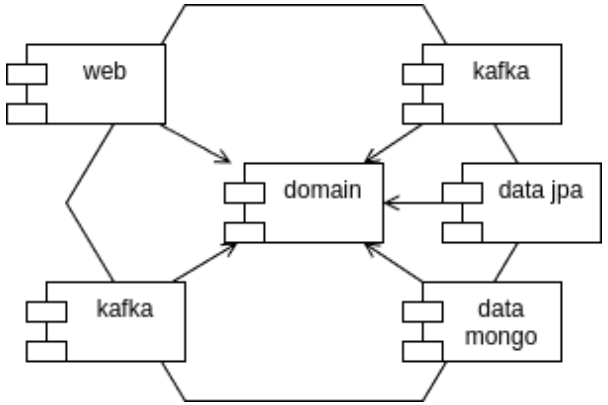
EN VISION HEXAGONALE



SÉPARATION DES CLASSES



SÉPARATION EN MODULES MAVEN



SÉPARATION EN MODULES MAVEN - PARENT

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
    <modelVersion>4.0.0</modelVersion>

    <groupId>com.example</groupId>
    <artifactId>hexagonal-architecture-sample</artifactId>
    <version>1.0-SNAPSHOT</version>
    <packaging>pom</packaging>

    <modules>
        <module>domain</module>
        <module>web</module>
        <module>data-jpa</module>
    </modules>
```

SÉPARATION EN MODULES - DOMAIN

```xml
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
    <modelVersion>4.0.0</modelVersion>

    <parent>
        <groupId>com.example</groupId>
        <artifactId>hexagonal-architecture-sample</artifactId>
        <version>1.0-SNAPSHOT</version>
    </parent>

    <artifactId>domain</artifactId>

    <dependencies>
        <dependency>
```

SÉPARATION EN MODULES - WEB

```xml
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
    <modelVersion>4.0.0</modelVersion>

    <parent>
        <groupId>com.example</groupId>
        <artifactId>hexagonal-architecture-sample</artifactId>
        <version>1.0-SNAPSHOT</version>
    </parent>

    <artifactId>web</artifactId>

    <dependencies>
        <dependency>
```

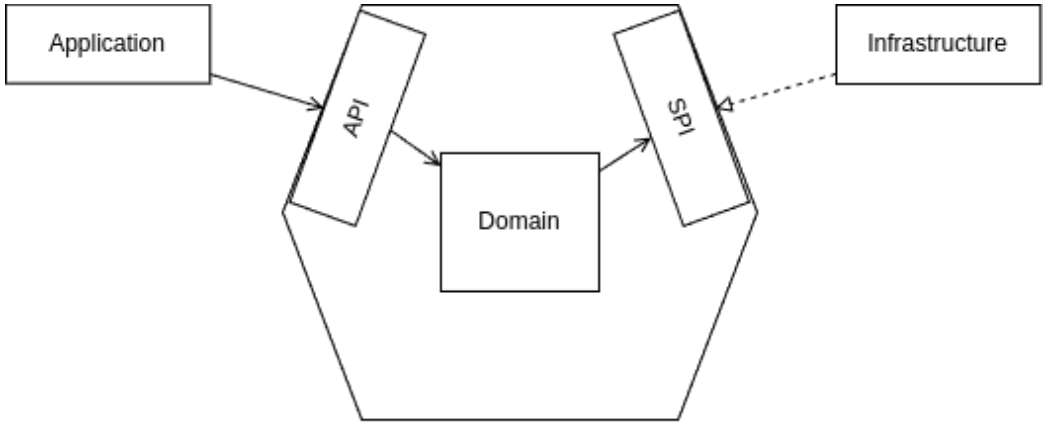
SÉPARATION EN MODULES - DATA-JPA

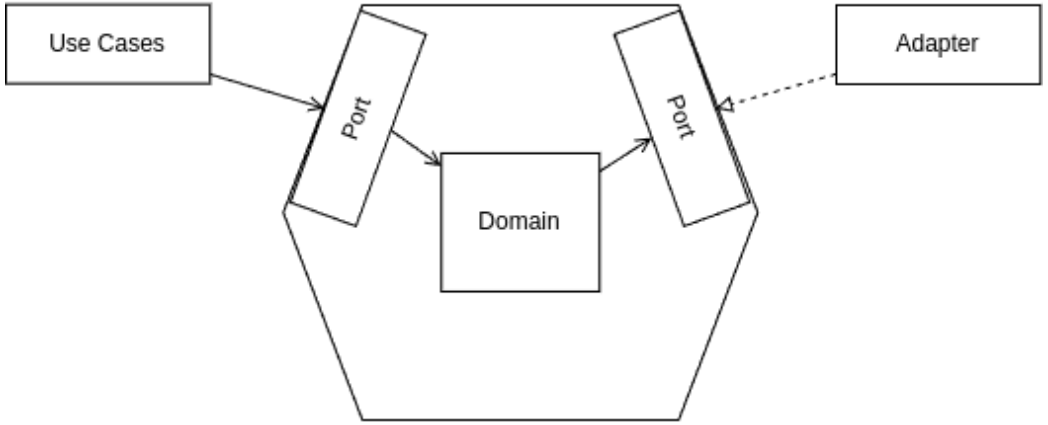
```
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
    <modelVersion>4.0.0</modelVersion>

    <parent>
        <groupId>com.example</groupId>
        <artifactId>hexagonal-architecture-sample</artifactId>
        <version>1.0-SNAPSHOT</version>
    </parent>

    <artifactId>data-jpa</artifactId>

    <dependencies>
        <dependency>
```

CE QU'IL FAUT RETENIR

- Organisation du code et séparation des responsabilités
- SOLID !
- Le domaine métier est *pur*, sans framework
  - annotations customs
  - tolérance pour certaines librairies
    - commons-lang, lombok
    - annotations d'architecture (jMolecules)
- Pas de "meilleure" façon, toujours adapter au contexte

VOCABULAIRES





VOCABULAIRES

- Controlleurs - DTO (Data Transfer Object) - VO (Value Object)
- Use Cases - Services - Domain Objects
- Repositories - Entities
- Ports = interfaces
- Adapters = implémentations

HEXA / CLEAN

- Ports / Adapters
- API / SPI : Application Programming Interface, Service Provider Interface
- In / Out

RESSOURCES SUPPLÉMENTAIRES

- DDD Vite Fait - pdf
- L'Architecture Hexagonale par la pratique, le live coding qui rendra vos applications plus pérennes - YouTube
- Le pattern Hive : une stratégie de modularisation