Configuration impérative

↳ fichiers de configuration xml

Configuration déclarative

↳ via des annotations dans le code

Java 1.4

→ tout était déclaré techniquement

→ code "pur"

→ fichiers xml de liaison

→ lourd, bcp de fichiers syntaxe à connaître

→ Centralisé

JEE 5+

→ possibilité de passer par annotation

@ <<annotation>>

→ Simplifier la conf par annotations → - de fichiers et de code

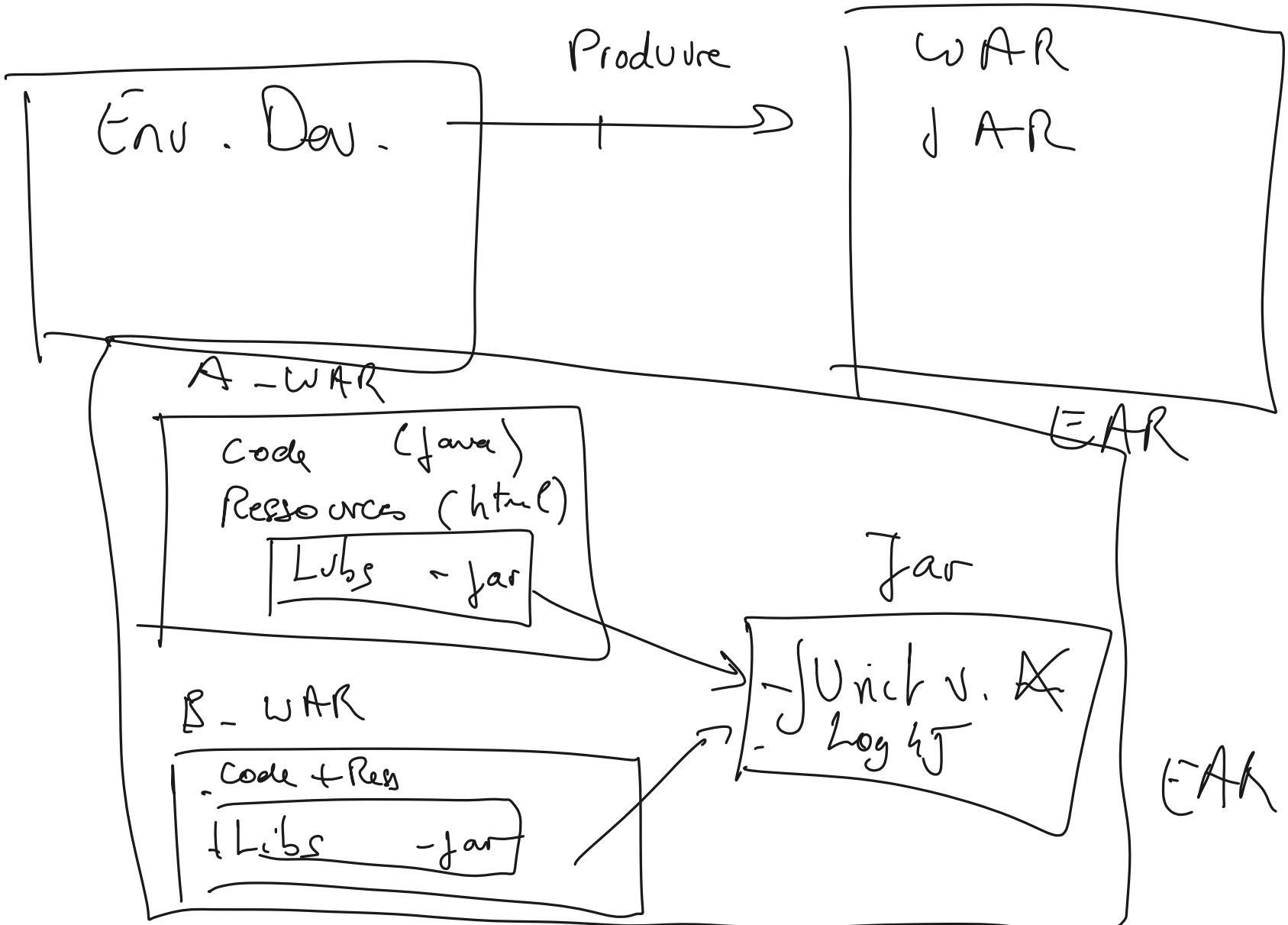
→ DECENTRALISÉ

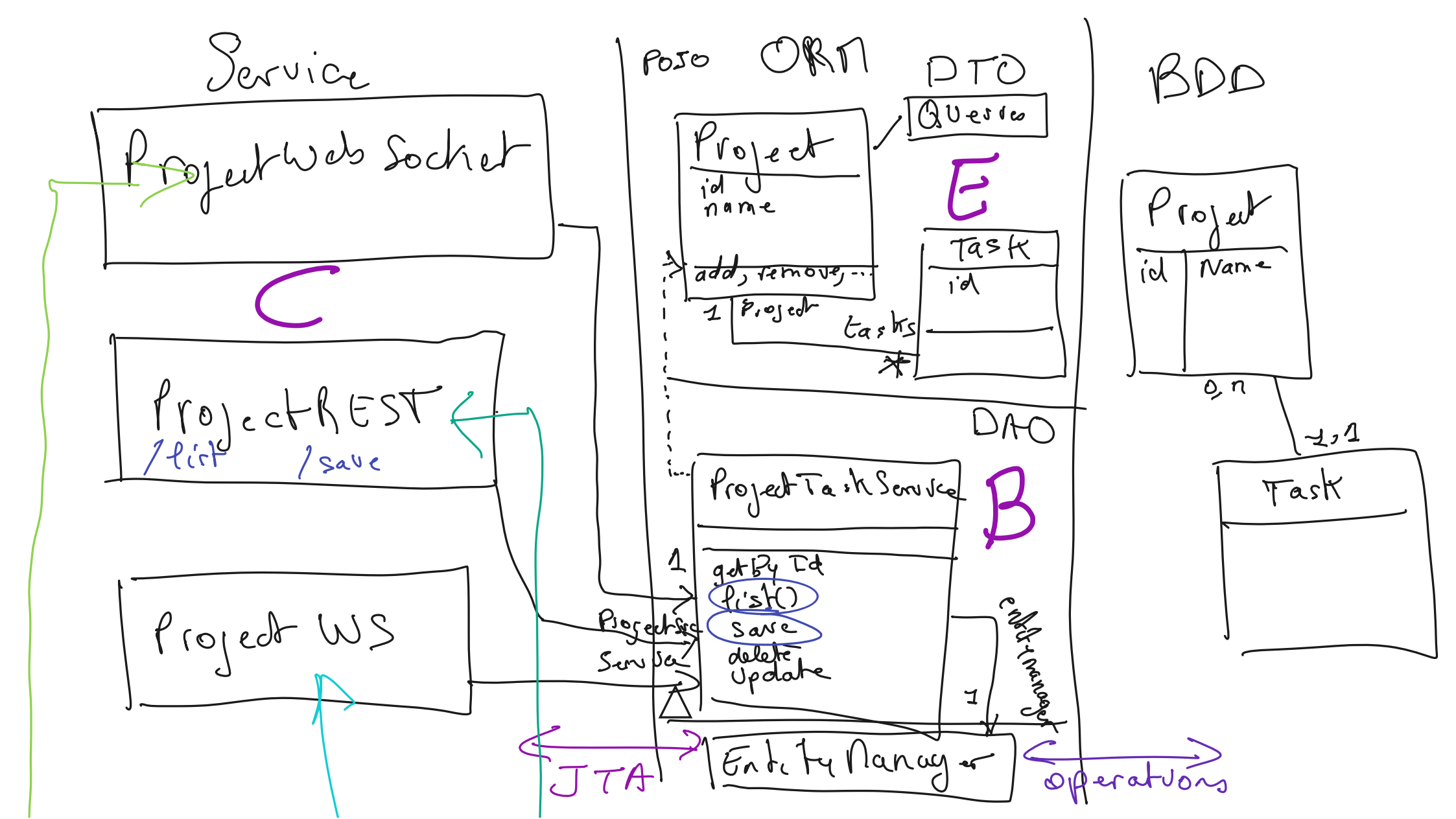
Archive Java = fichier Zip

Java AR ⟶ ~~Java SE~~
EJB, Bibliothèques

Web AR ⟶ Conteneur Web

Enterprise AR ⟶ Livre le JAR et WAR
ensemble


Produire
WAR
JAR
Env. Dev.
A - WAR
Code (Java)
Ressources (html)
Libs - jar
EAR
Jar
- JUnit v. 4
- Log 4J
B - WAR
- Code + Res
+ Libs - jar
EAR


Service
POJO
ORM
DTO
BDD
ProjectWebSocket
C
Project
id
name
add, remove, ...
1
Project
Tasks
*
Task
id
QUeries
E
Project
id
Name
0,n
1,1
Task
ProjectREST
/list
/save
DAO
ProjectTaskService
B
1
getBy Id
list()
save
delete
update
Project Ws
Project Service
entity manager
1
JTA
EntityManager
operations

clients

Page htmls/
Javascript

- clients lourds
- Orchestrateur ESB
- Techno concurrentes (.net)

- html / Ajax
- lourd / JSON
- client App Java → JSF, etc...
- Peripherique Mob.
- interCnx Web App

Design Patterns

GoF → par objet

SingleTon

Façade

factory

Iterator

Boulder

- - - -

ECB

architecture

Hibernate → JPA ⇄ Hibernate → EclipseLink

MVC

ORM

IoC

Struts → Spring mvc → JSF

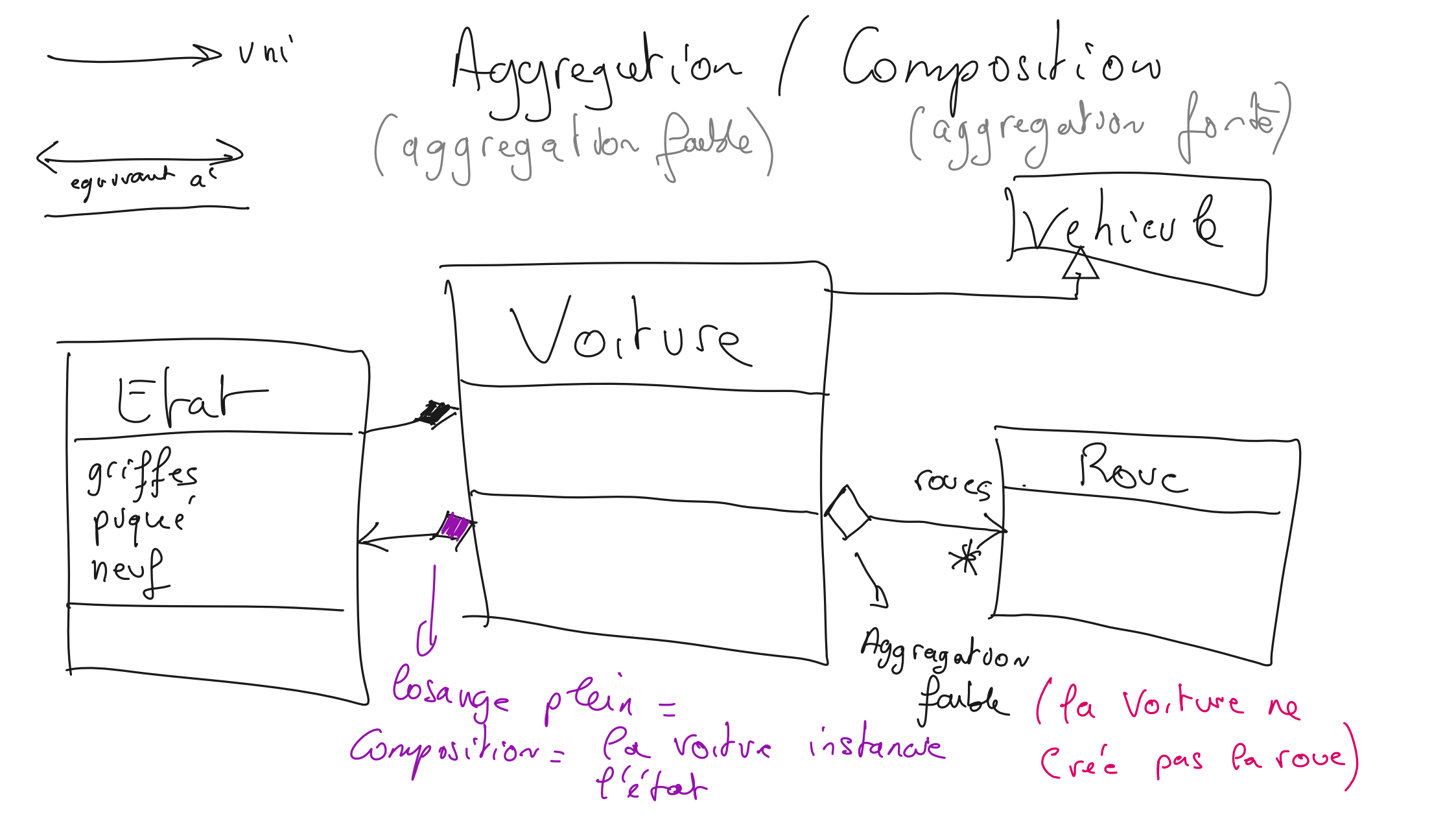
Spring Ioc → CDI

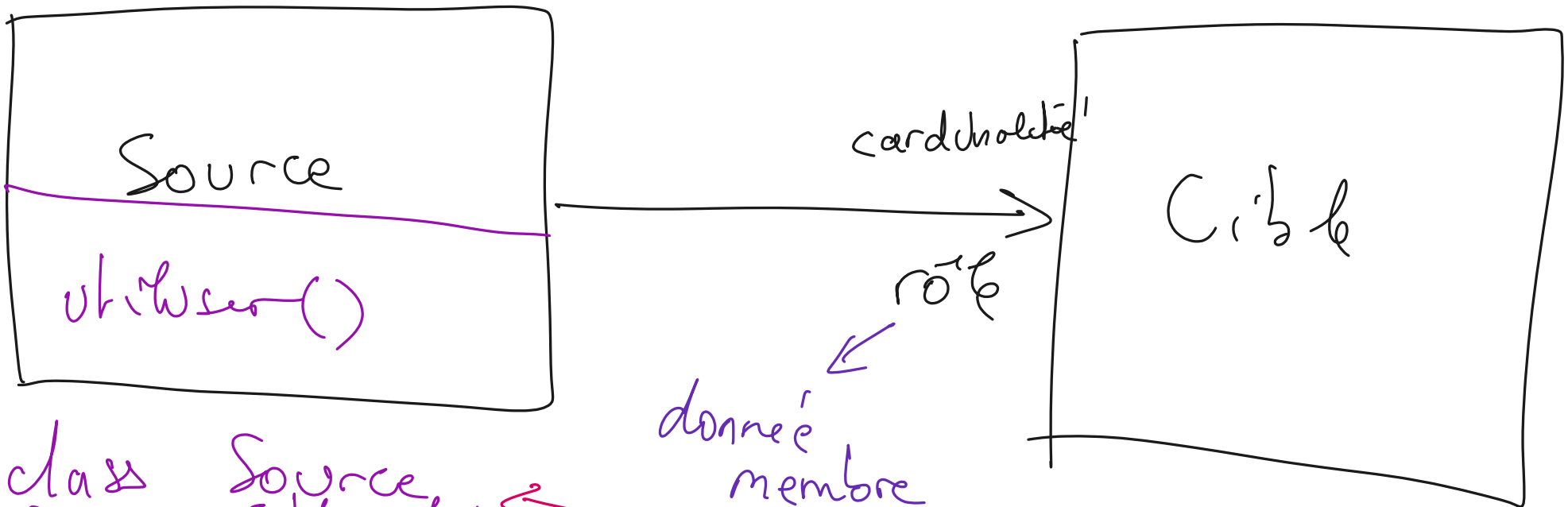
Divers

Grasp (bonne pratiques générale)

ECB

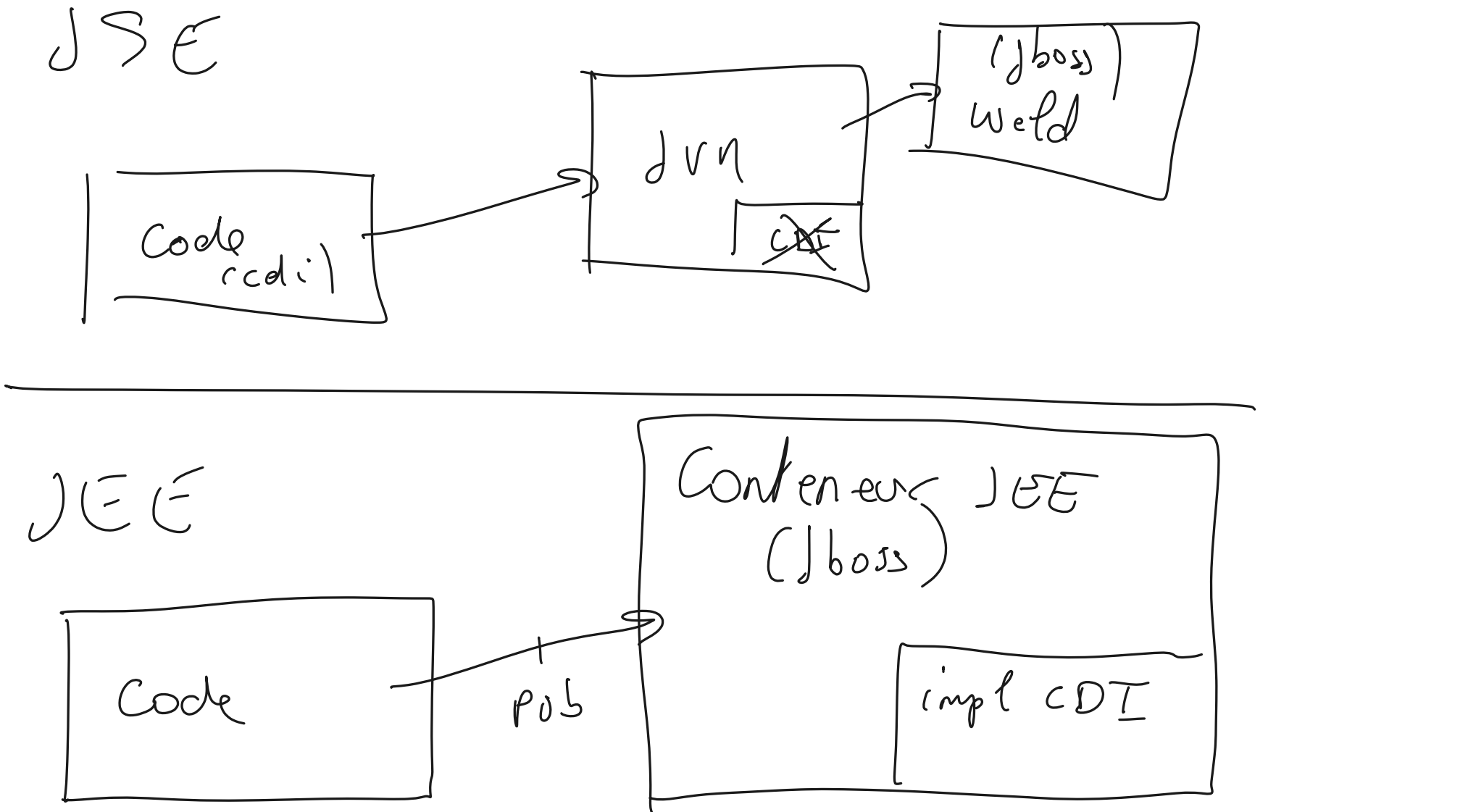
- . . .


vni
equivaut à
Aggregation / Composition
(aggregation faible)
(aggregation forte)
Vehicule
Voiture
Etat
griffes
pique
neuf
roues
Roue
losange plein =
Composition = la voiture instancie l'état
Aggregation faible (la voiture ne crée pas la roue)



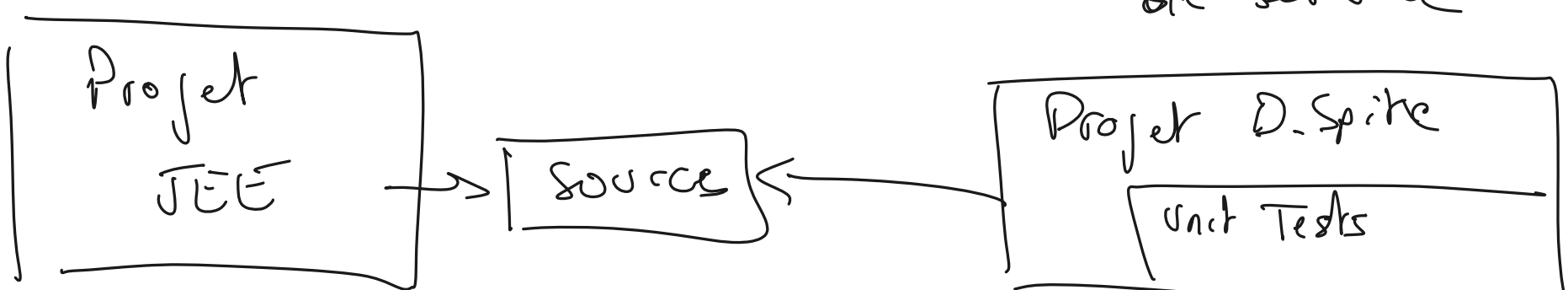

```
class Source
{
    Cible role;
    public Source()
    {
        role = new Cible();
    }
    utiliser() { role.operation(); }
}
```

obtenir une instance par injection.

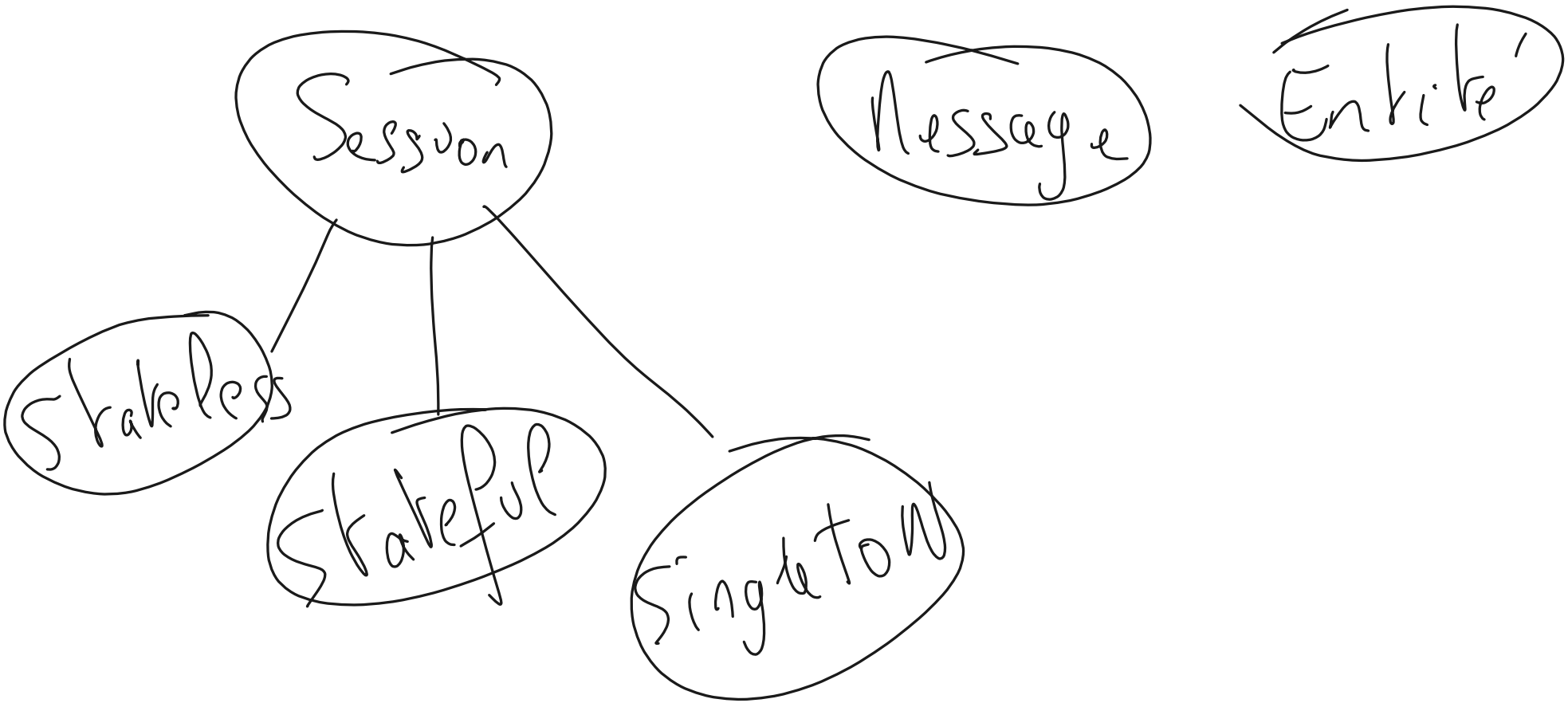

JSE
Code
rcdi
dvn
CDI
Jboss
Weld
JEE
Code
pom
Conteneur JEE
(Jboss)
impl CDI

Tests unitaire de Beans Via DeltaSpike :

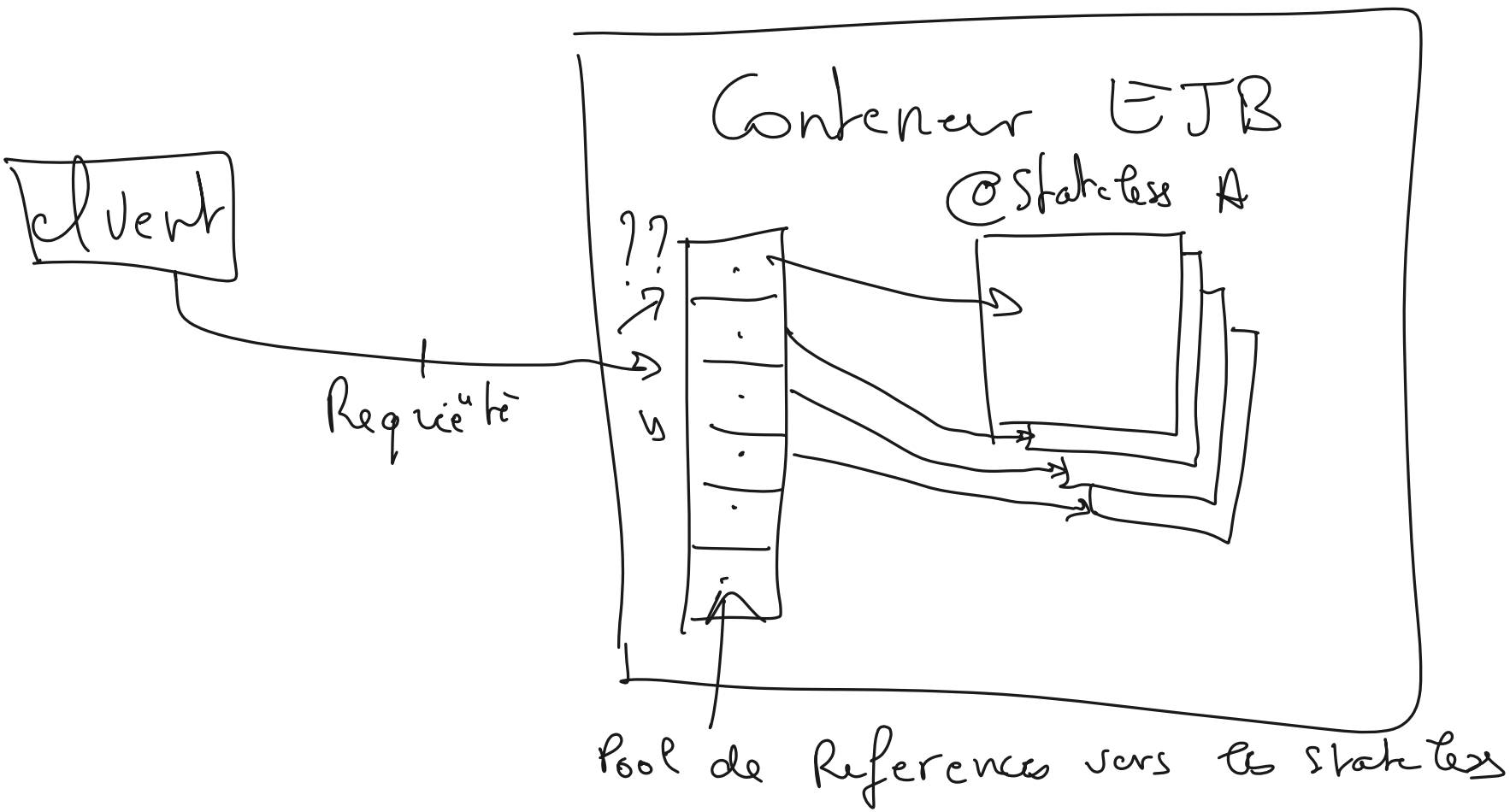
Objectif : Implementer un conteneur CDI permettant l'execution de tests unitaires via JUnit, Beaucoup plus leger qu'un conteneur de service

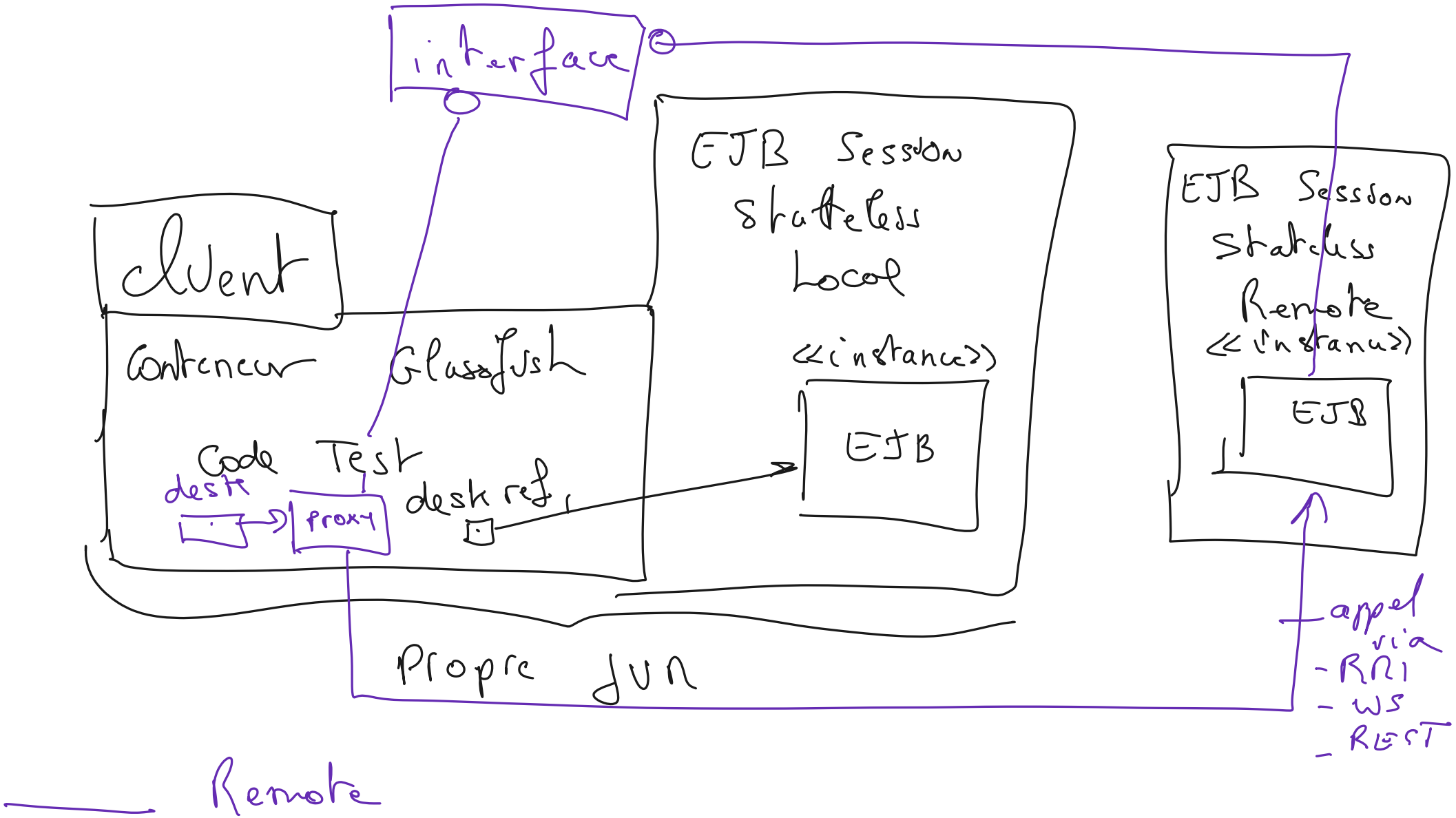


Types d'EJB



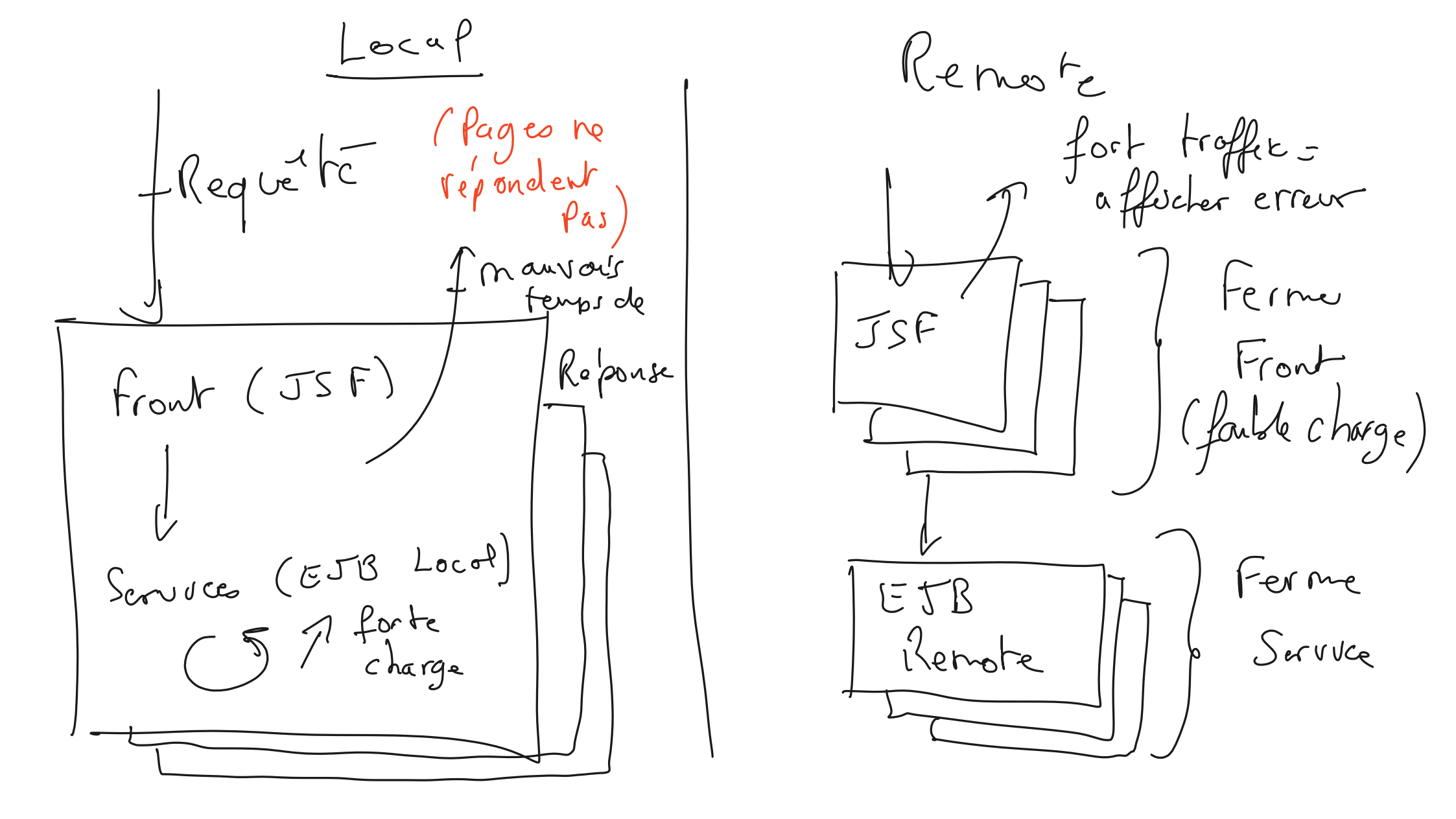
EJB Session Stateless

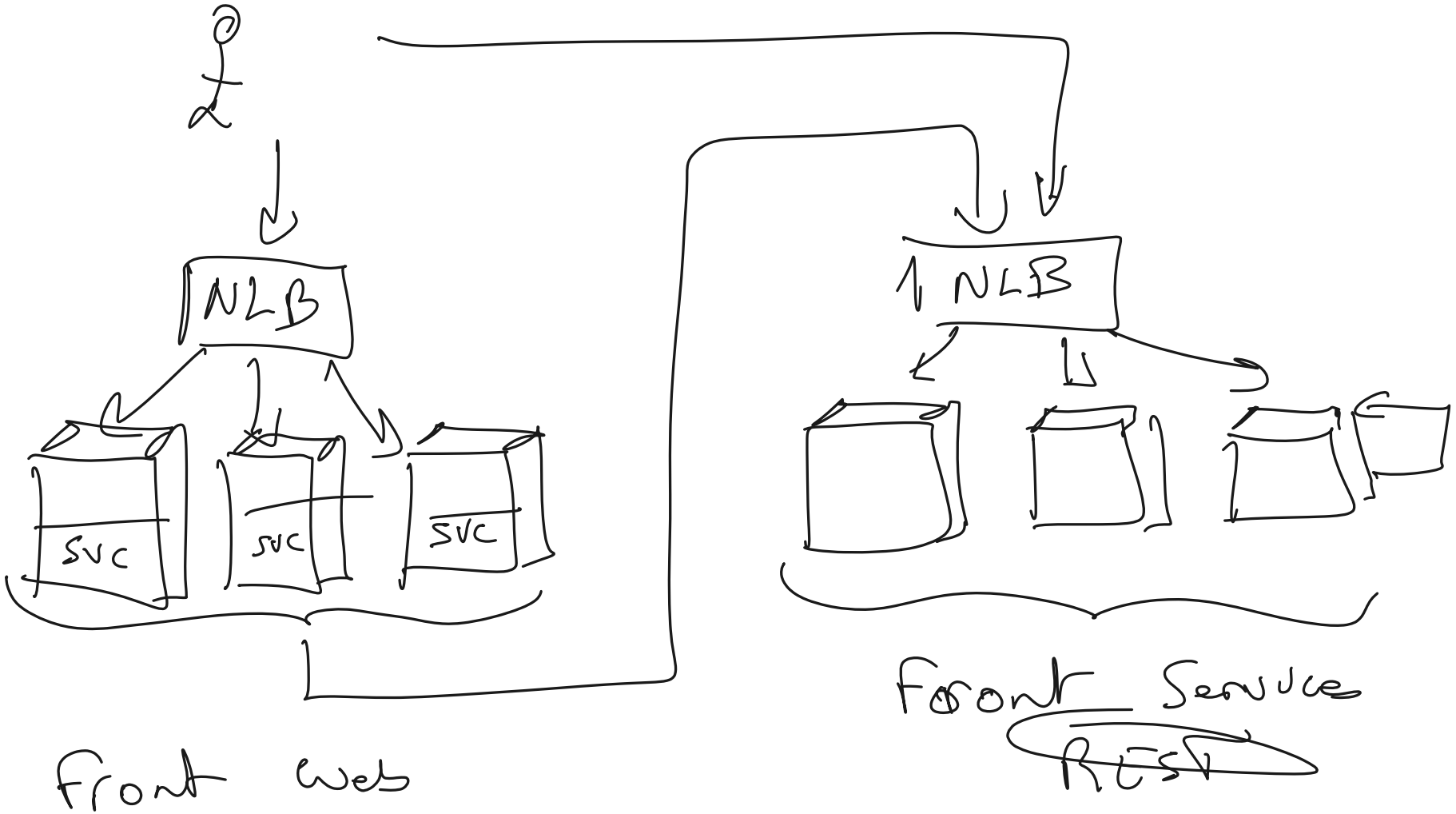



interface
client
Container
Glassfish
Code Test
desk
proxy
desk ref
EJB Session
Stateless
Local
<<instance>>
EJB
EJB Session
Stateless
Remote
<<instance>>
EJB
Propre JVM
appel via
- RMI
- WS
- REST
Remote

Exercice EJB stateless

- Créez un Bean "Personne" avec Nom et Prenom en String
- creez un stateless Session Bean qui prends un Type Personne, et renvoie une personne avec le nom en majuscule
- Testez cet EJB
- obtenez 3 instances de cet EJB
  → est-ce 3 instances différentes ?


Local
Requête
(Pages ne répondent pas)
mauvais temps de
Front (JSF)
Réponse
Service (EJB Local)
forte charge
Remote
fort trafic =
afficher erreur
JSF
Ferme
Front
(faible charge)
EJB
Remote
Ferme
Service


NLB
SVC
SVC
SVC
NLB
Front Web
Front Service
REST

Exercice EJB Session Stateful

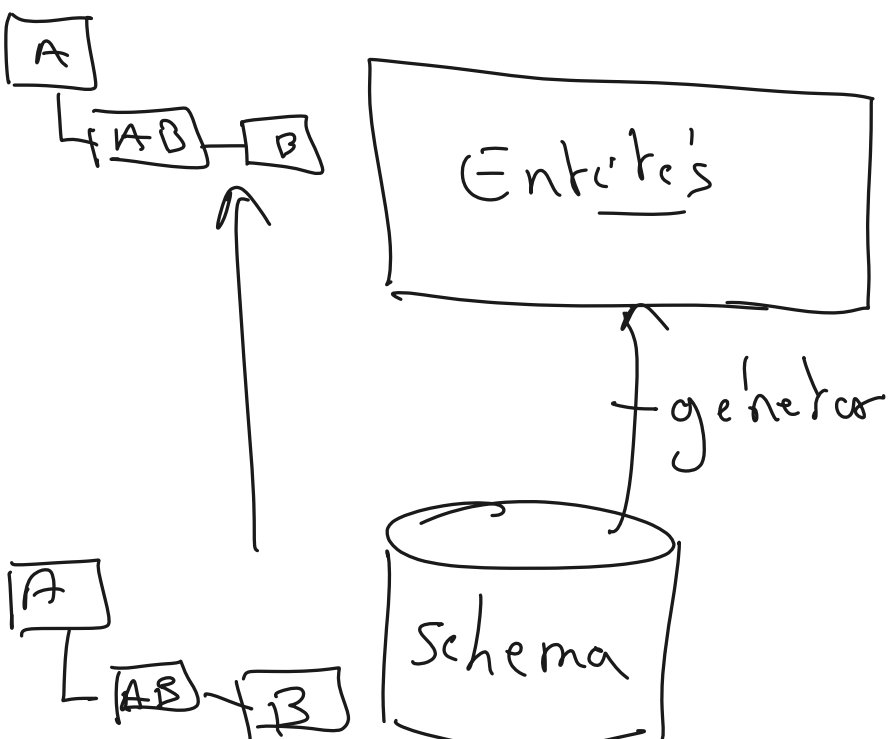
Creer un EJB StackOfStrings

qui peut
- recevoir une chaîne à ajouter à la pile (FIFO ou LIFO ou ...)
- retirer une chaîne de la pile
- obtenir un tableau des chaînes

- instance

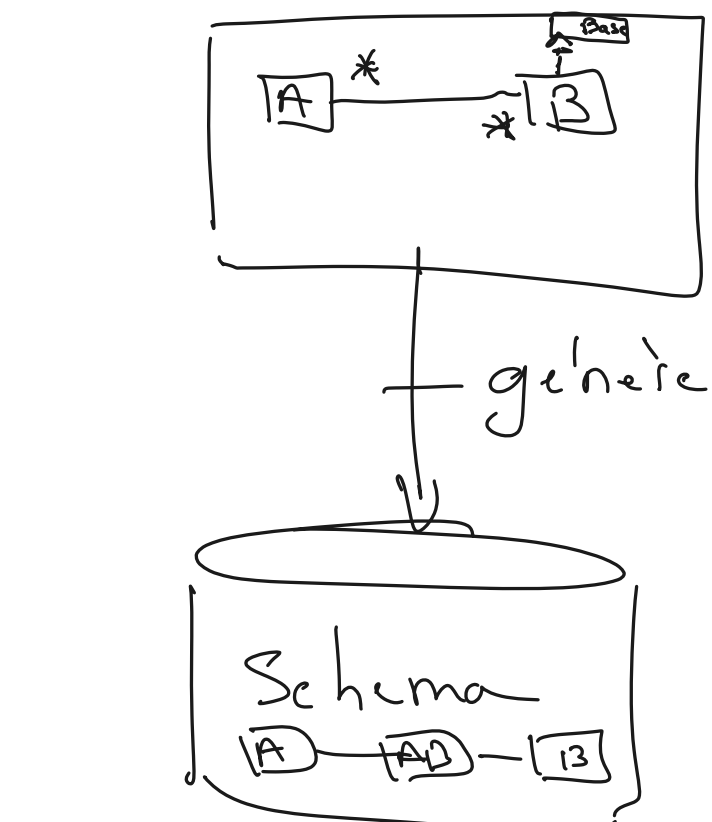
Database first / Bottom Up

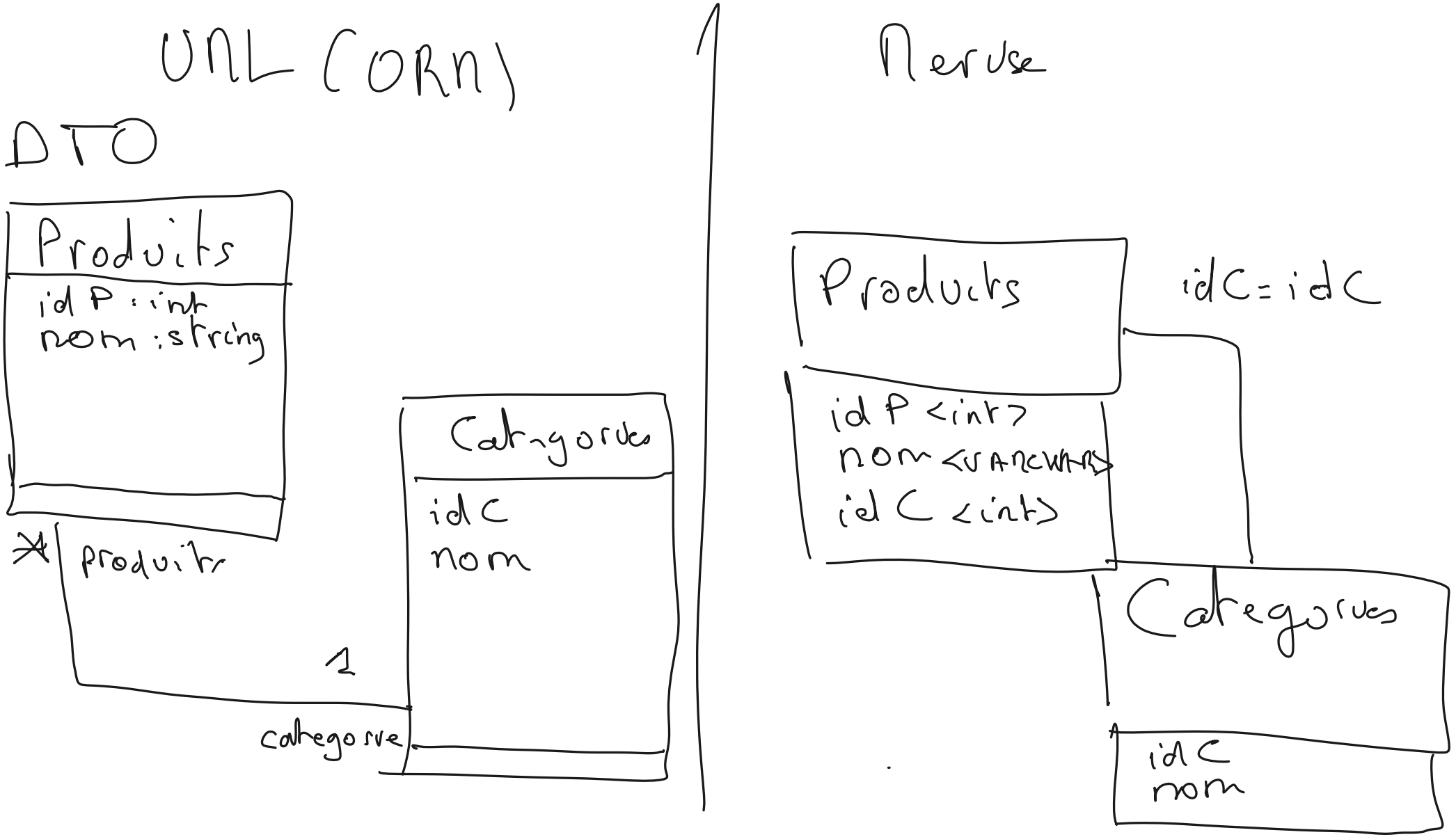
→ Lorsque la BDD existe déjà

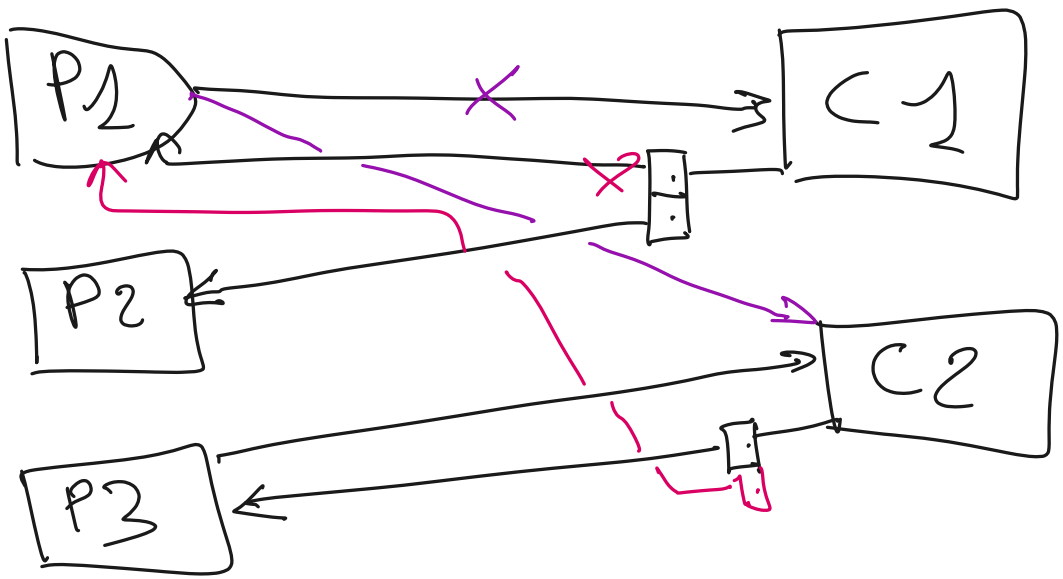


Model First / TopDown

focalise sur le modèle (pur)




UML (ORM)
DTO
Produits
idP : int
nom : string
produits
*
1
categorie
Categories
idC
nom
Nerise
Products
idP <int>
nom <VARCHAR>
idC <int>
idC=idC
Categories
idC
nom



P1 → C1

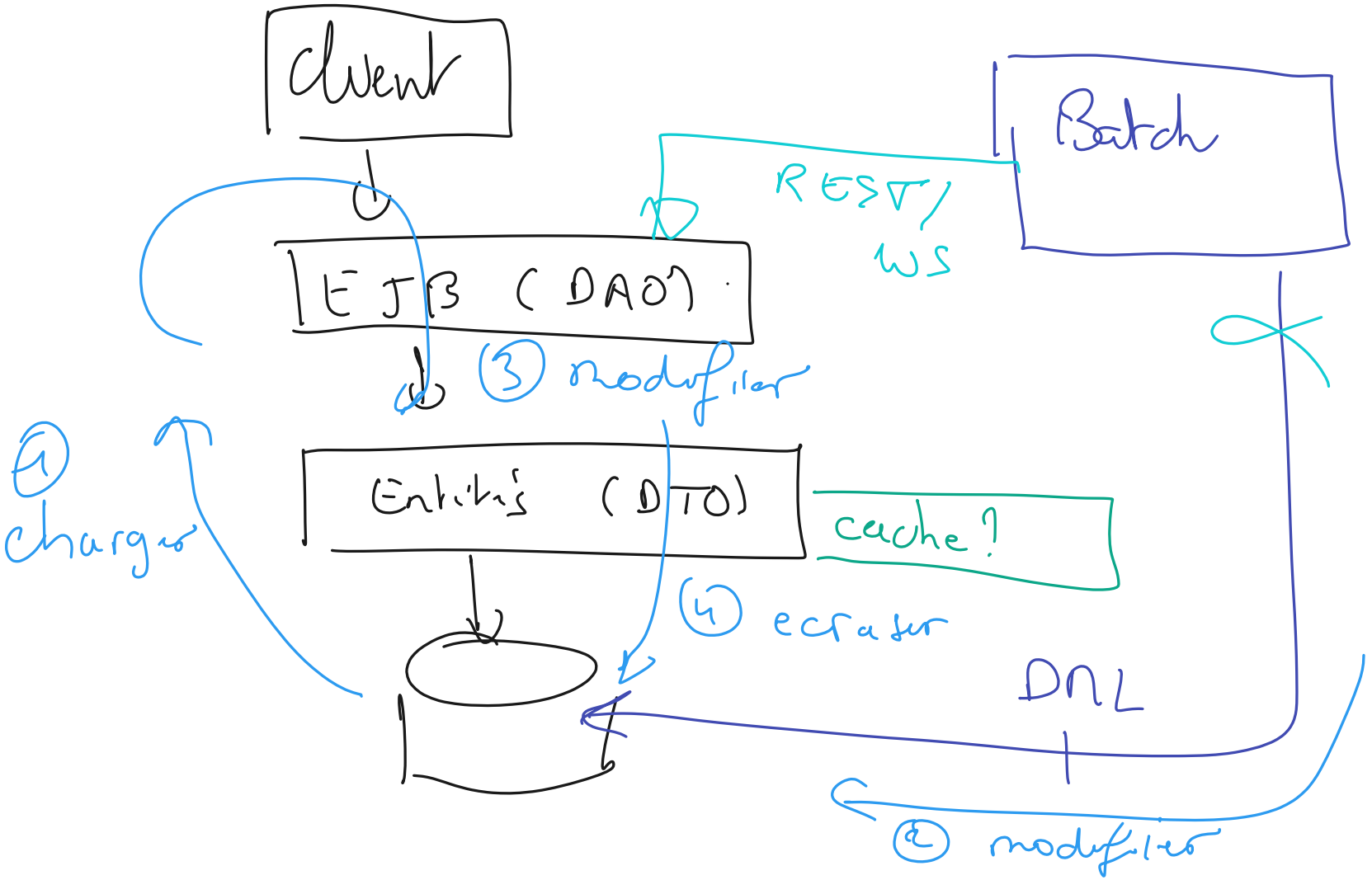
P2 → C1

P3 → C2

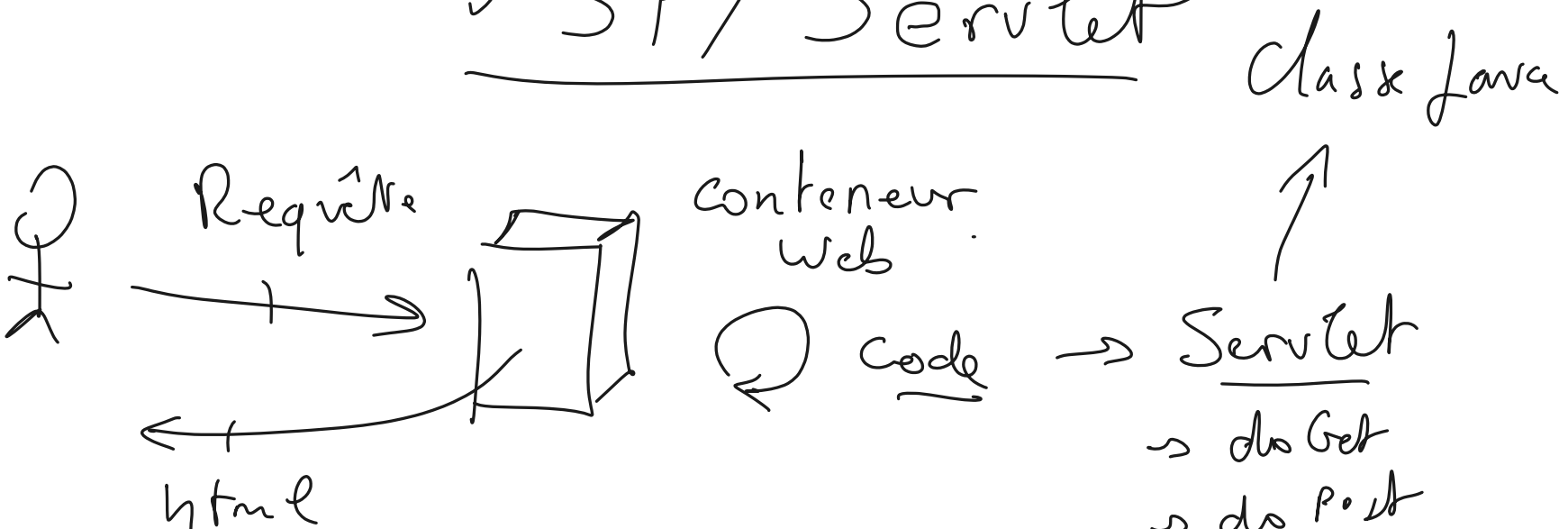
↳ P1 → C2

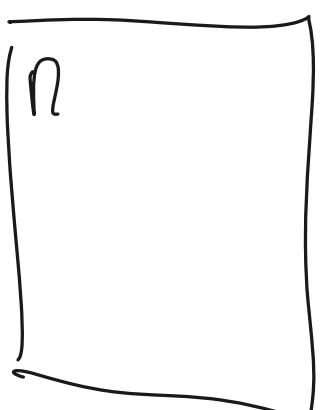
P1.categorie = C2;

C1.produits.remove(p1); C2.produits.add(p1);


client
Batch
REST/
WS
EJB (DAO)
③ modifier
① charger
Entités (DTO)
cache ?
④ écraser
DML
② modifier

JSP / Servlet


Classe Java
Requête
conteneur web
code
Servlet
doGet
doPost
...
Html


n

JSF = implémentation du MVC

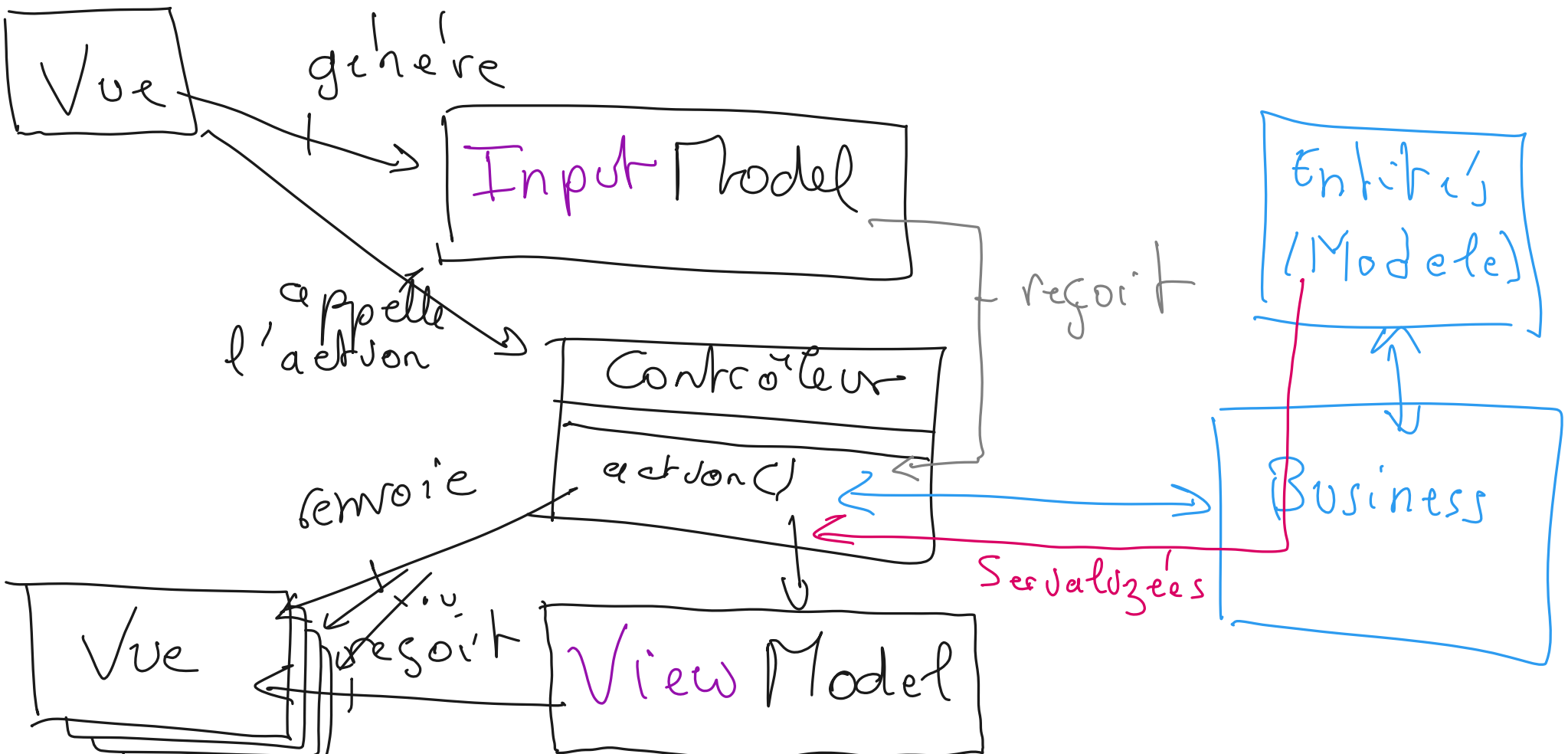
2000 (Hibernate + Struts) → 2010 + (JPA (Hib) Spring (MVC))

Modèle = Données passées entre la Vue et le contrôleur (deux sens)

Vue = Affichage des données (évite de Code)

Contrôleur = interface entre la Vue et le métier (EJB)

Couches

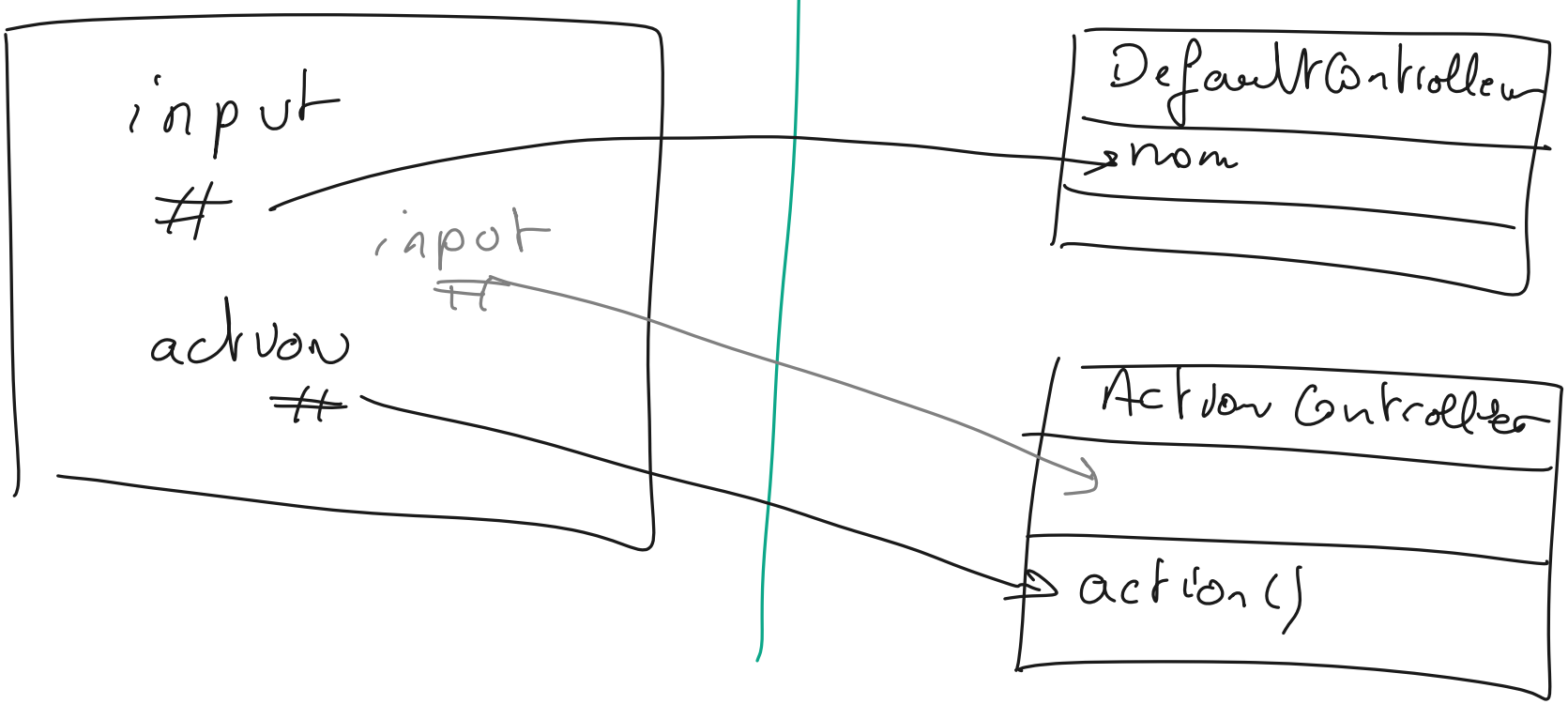

Vue
génère
Input Model
appelle l'action
Contrôleur
action()
reçoit
Entités (Modèle)
Business
renvoie
Vue
reçoit
View Model
Sérialisées

V Vue

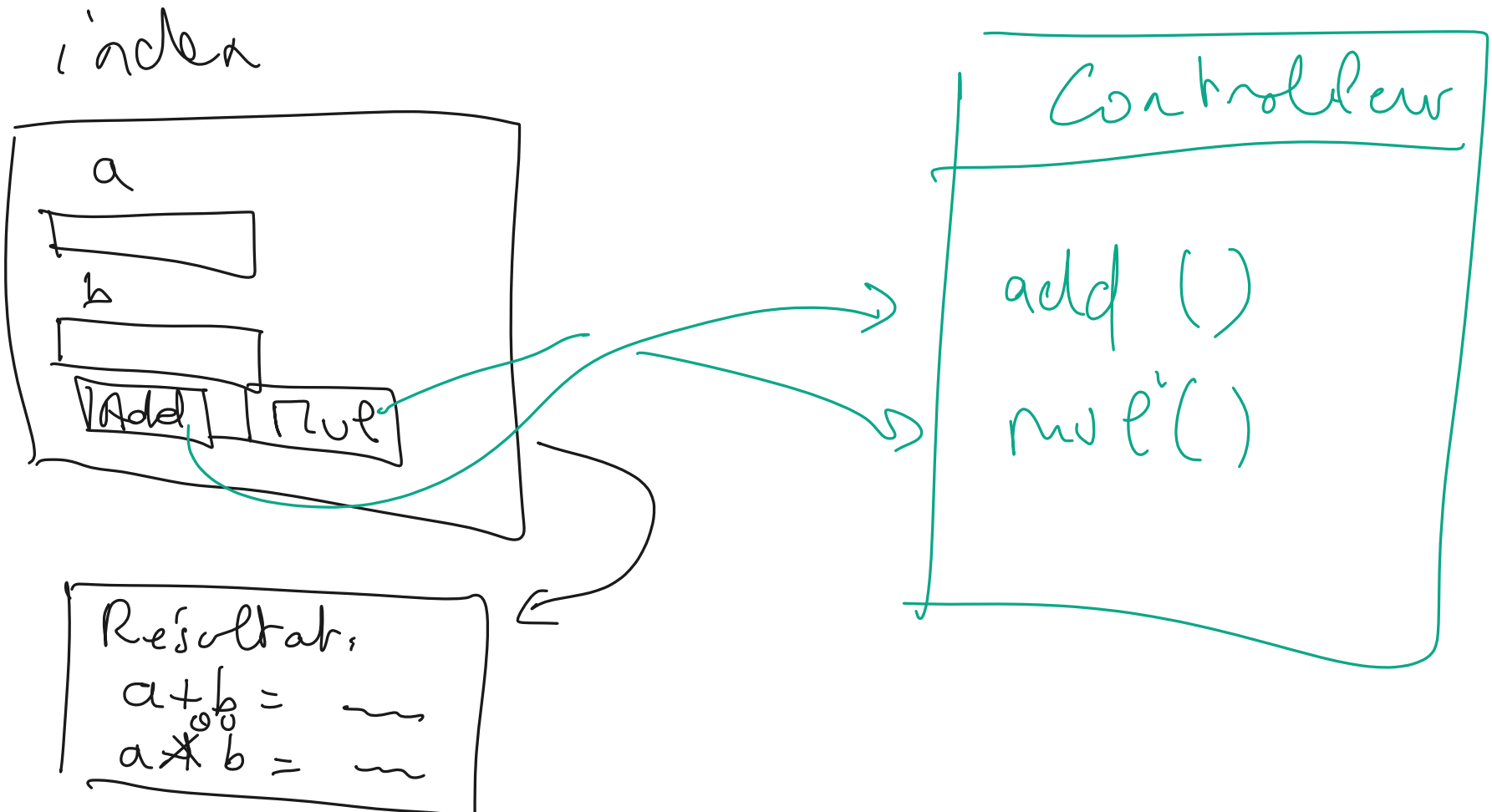
M Beans

→ Modèles

→ Controleurs



Réaliser une calculatrice



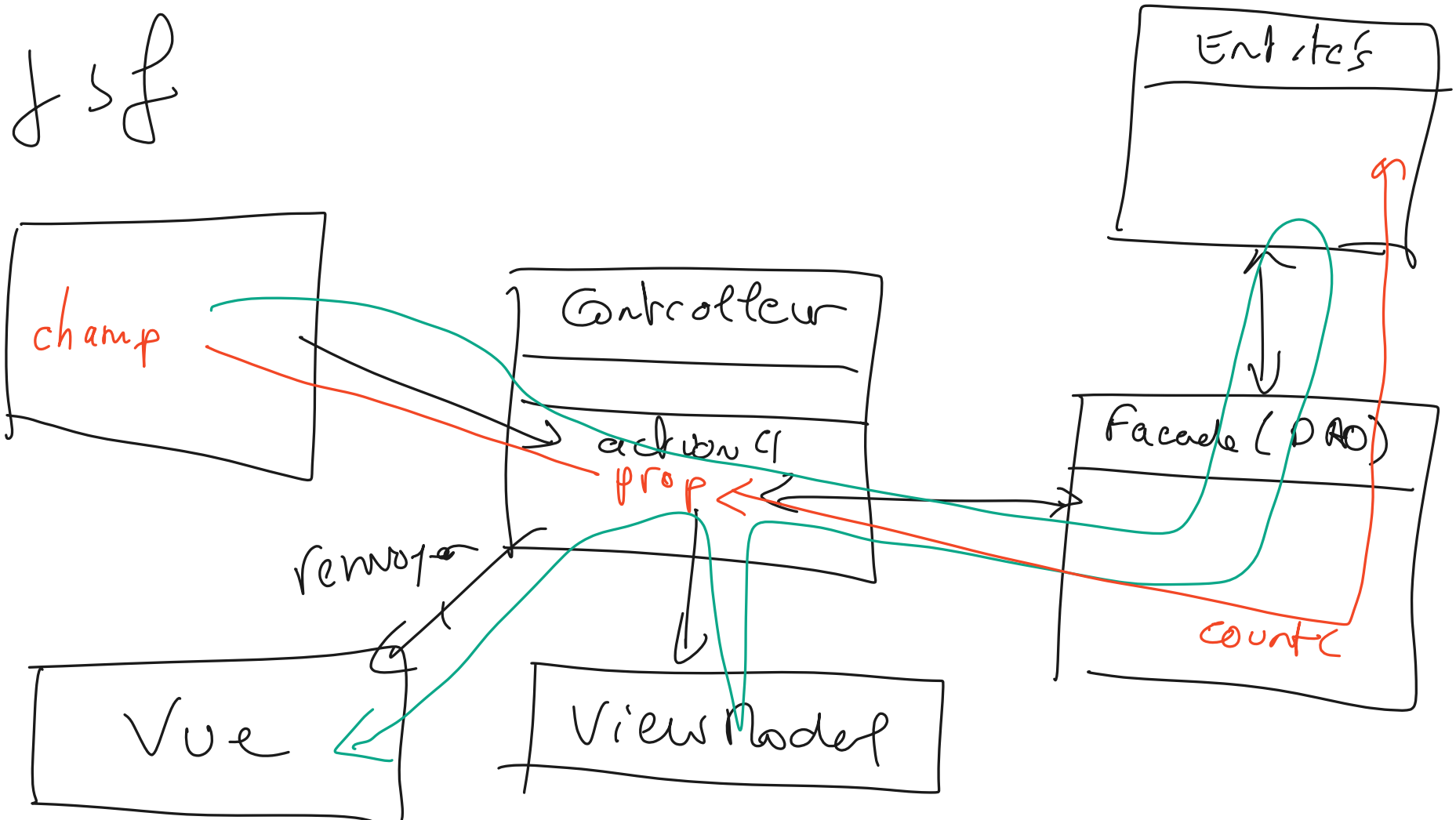
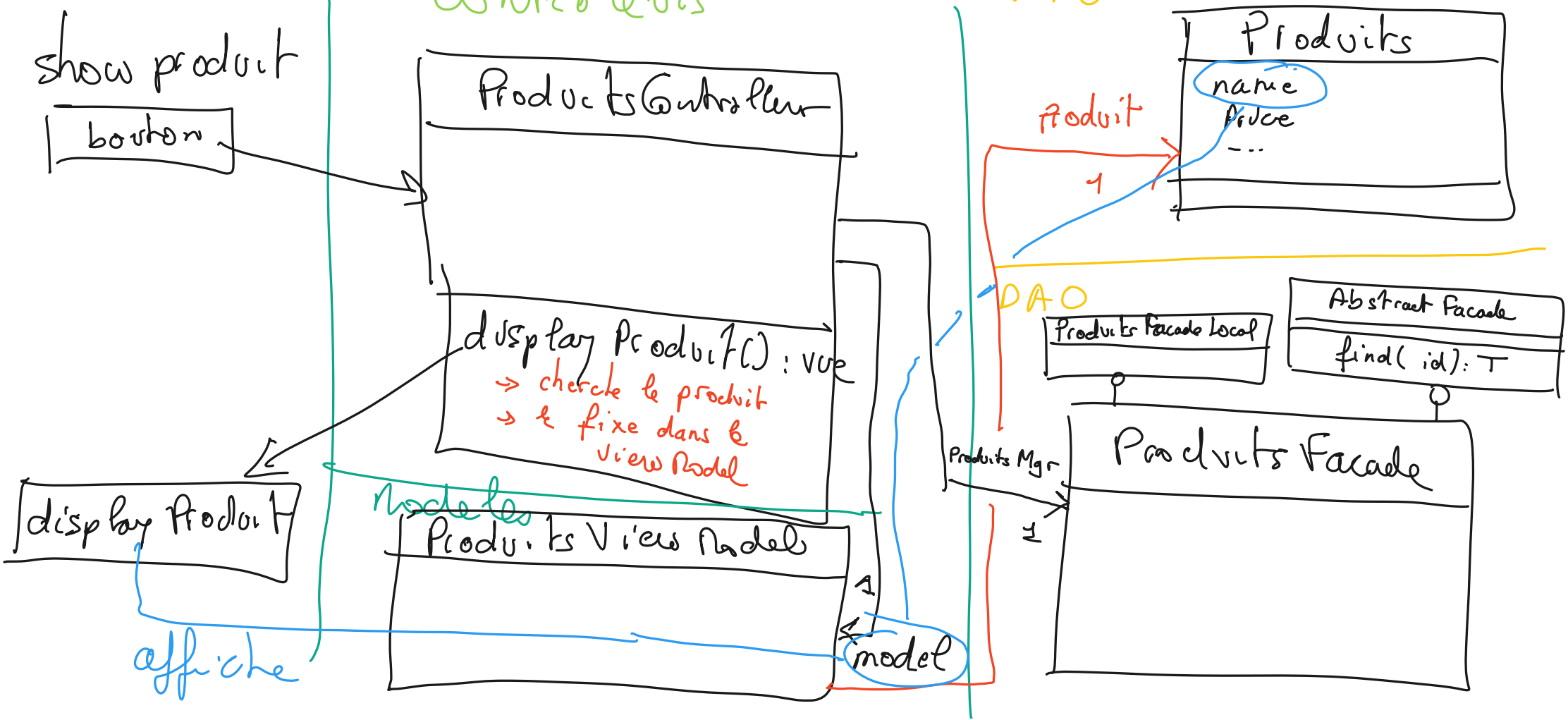

jsf
champ
Controleur
action ()
prop
Entités
Facade (DAO)
count
retour
Vue
ViewModel

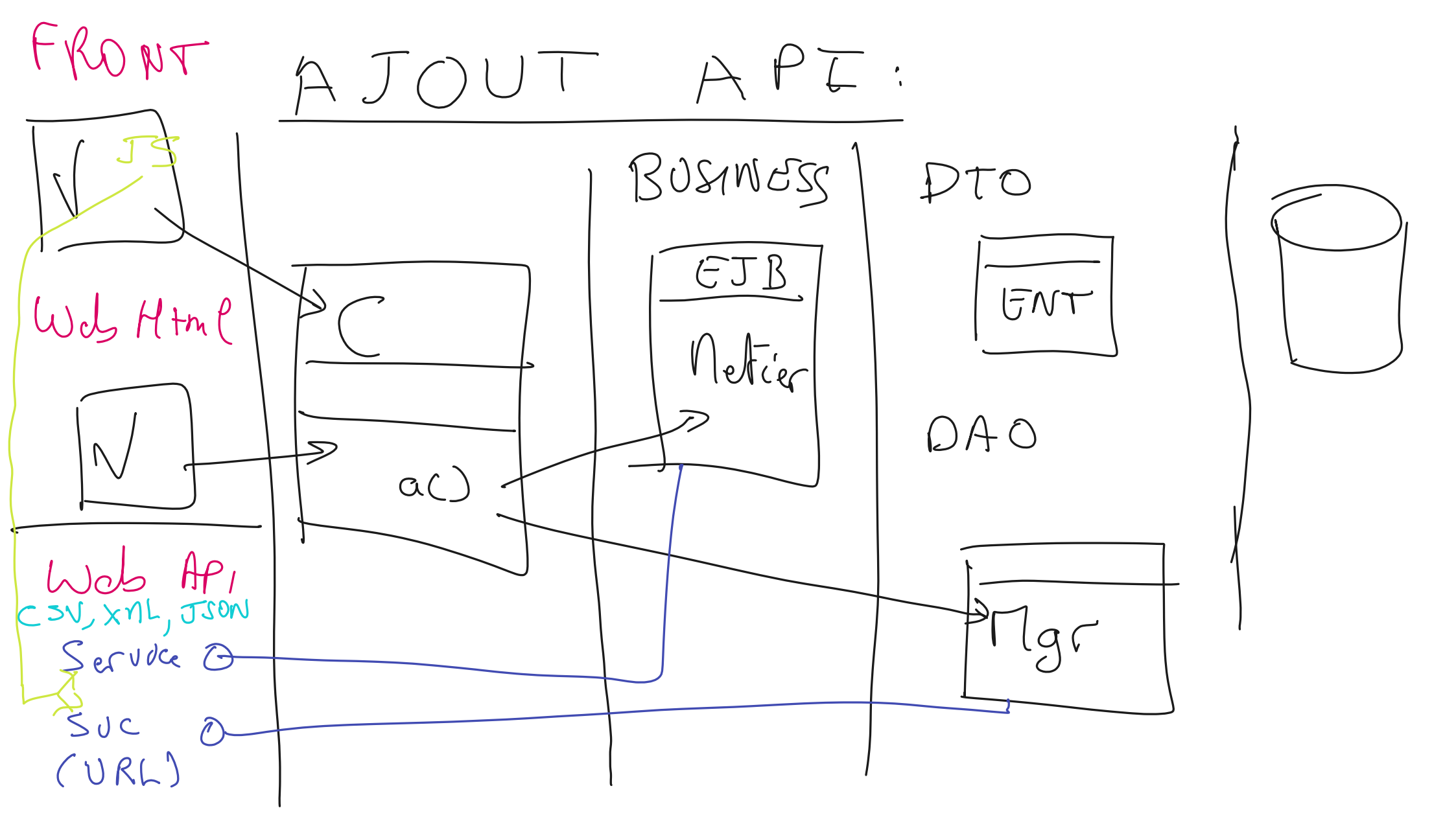
Diagramme de la Solution

Vues

Contrôleurs

DTO

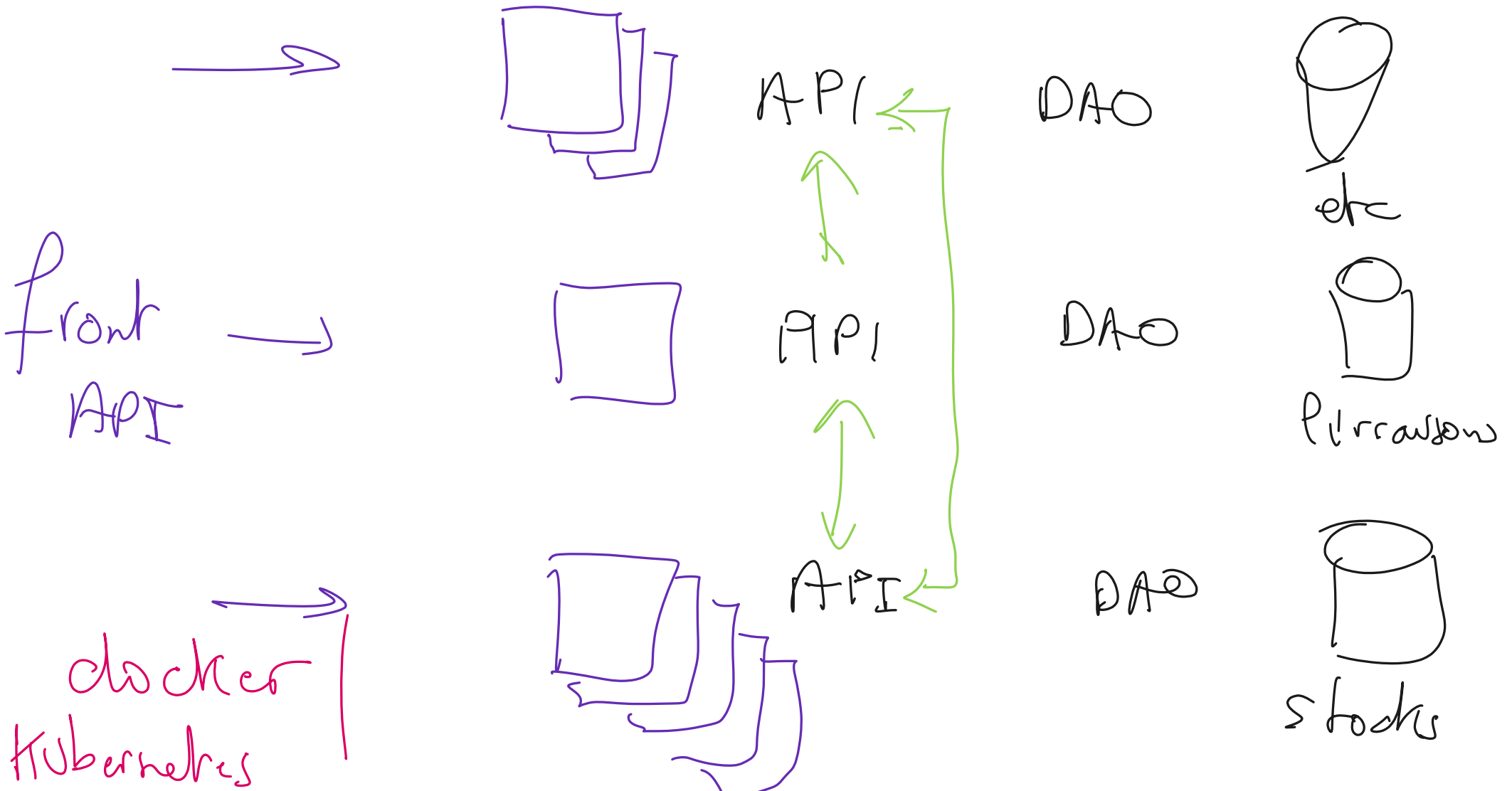



FRONT
AJOUT API :
JS
Web Html
V
C
a()
BUSINESS
EJB
Metier
DTO
ENT
DAO
Mgr
Web API
CSV, XML, JSON
Service
SUC
(URL)

Micro Services (concept)

Apps → API → domaine fonctionnels

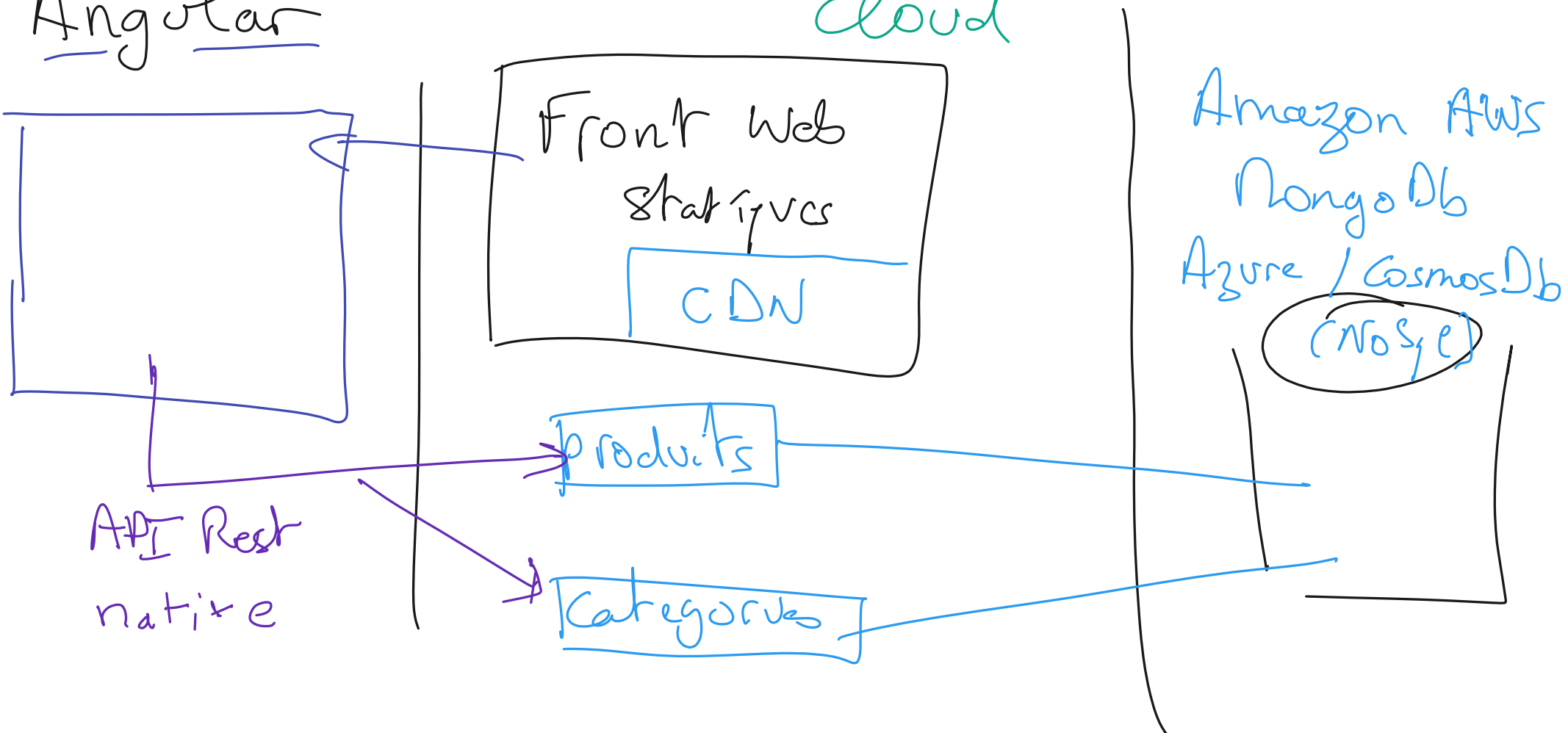
- → Auth
- → gestion clients
- → " stocks
- → " livraisons -


front
API
docker
Kubernetes
API
API
API
DAO
DAO
DAO
etc
Postgres
stocks

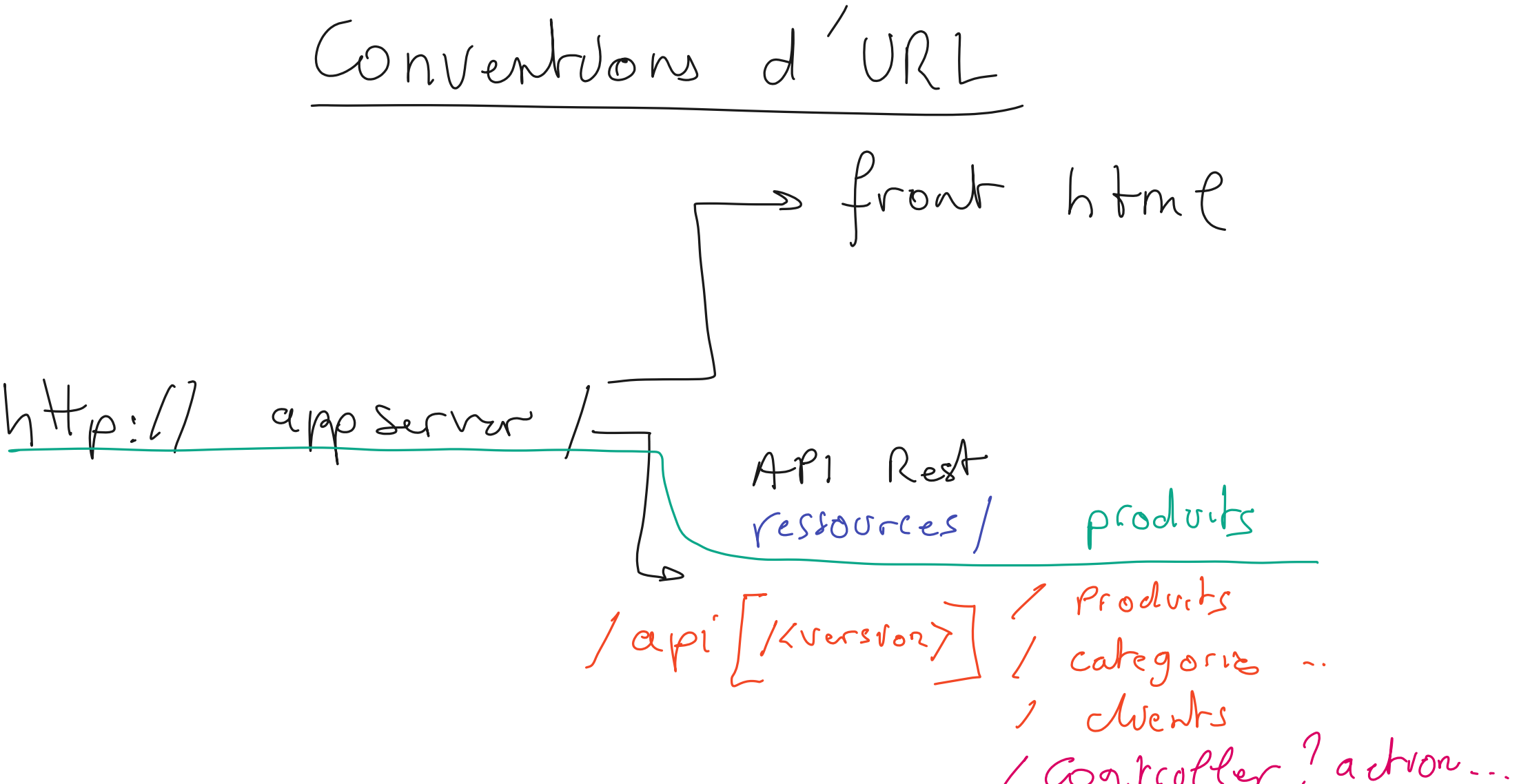
Autres architectures de Services

Angular

cloud



Conventions d'URL

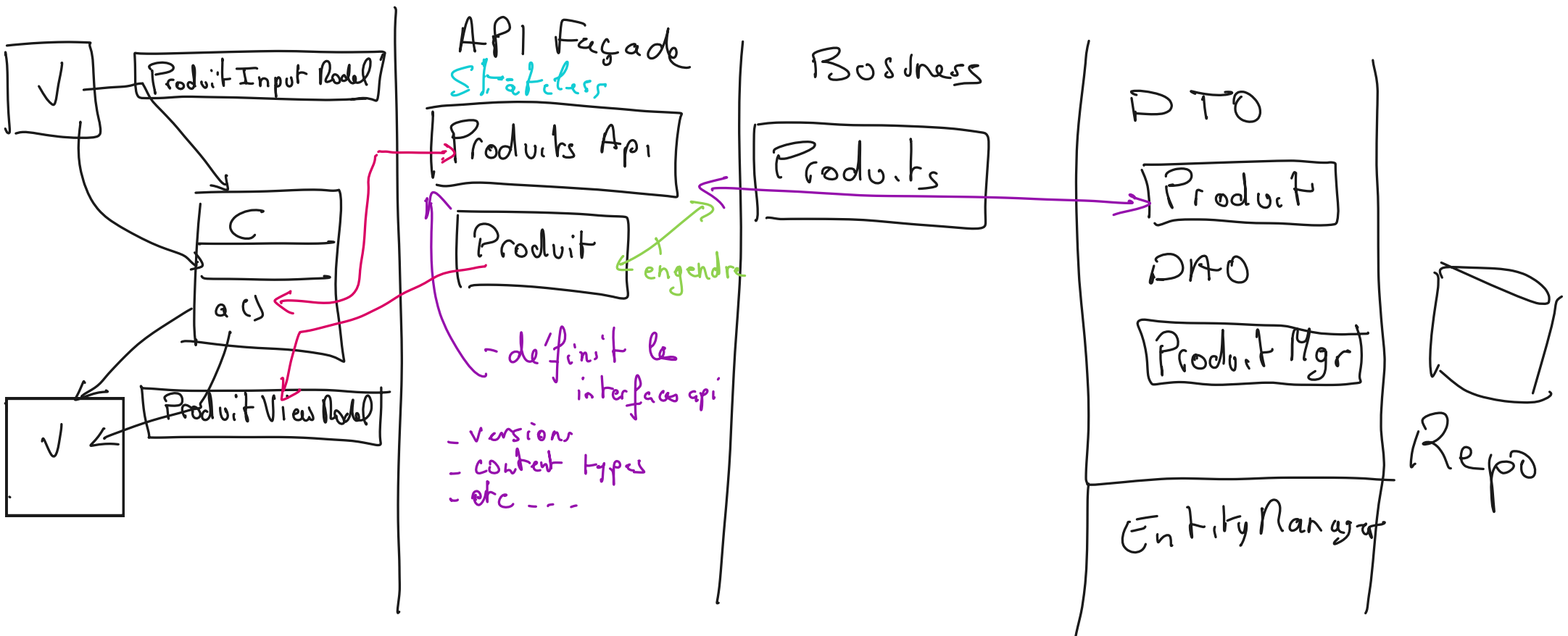


chemins d'URL

/api/clients/1/pays/ville

/api/clients/1?column=pays&column=ville

Diagramme avec Façades



Sécurisation de la couche Service

R → Accès

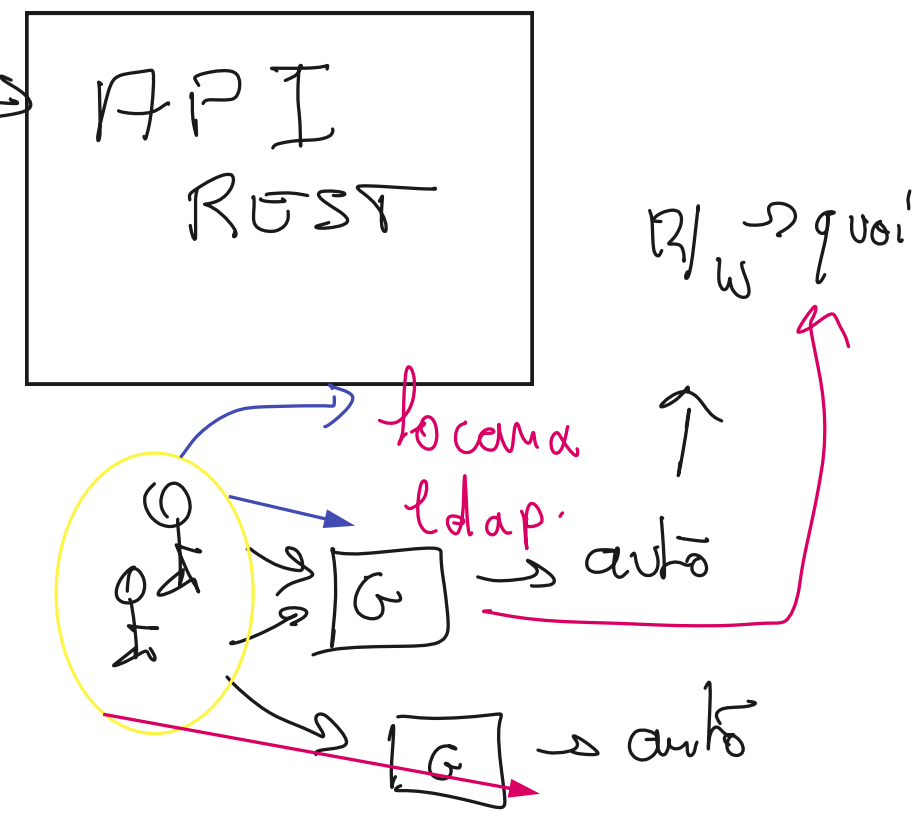
① Protocole

http / TLS (https)

② Authentification (login/pass)

↳ ③ Autorisations RBAC

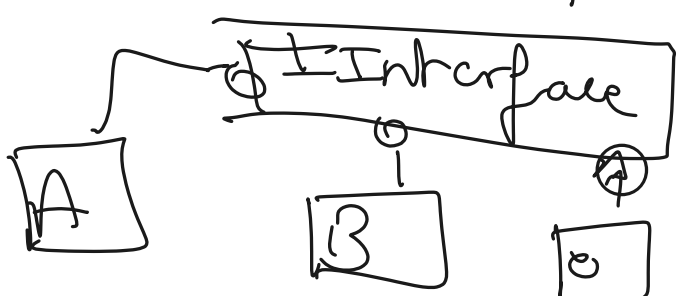
Role Based Aut. Con.



CDI+

@inject IInterface obj;

@inject @Any Instance<IInterface> services;



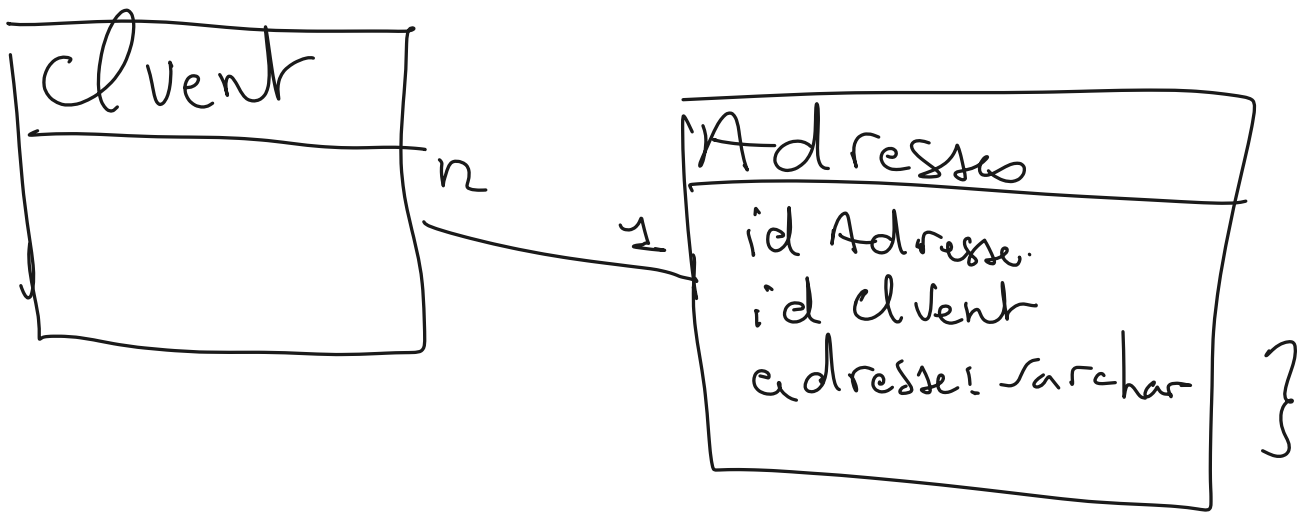
iterable de instances

for(IInterface e: services)

=> 3 iterations = instances de A, B, C

@Id → PK a colonne unique

@EmbeddedId → PK Composé.




```
class Client {

@ElementCollection
List<String> adresses
```

EJB Stateless / Singleton et accès concurrents

Java garantit granularité de base pour accès concurrents

(i++ → pbm en C++)

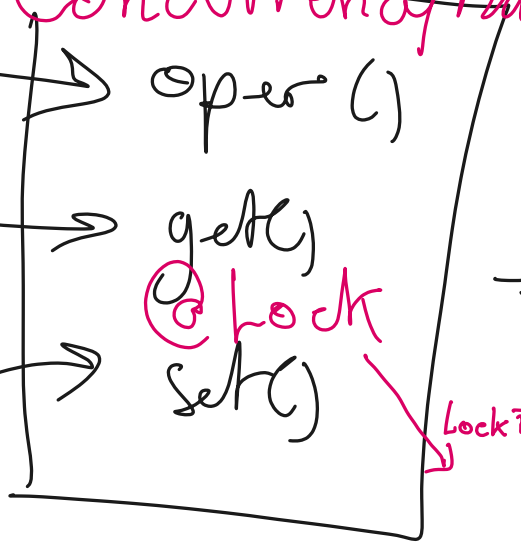
→ pas un pbm en Java

→ Java applique les verrous

→ concurrence gérée par le conteneur par défaut

@ConcurrencyManagement

CMT



→ Bean (à la main)

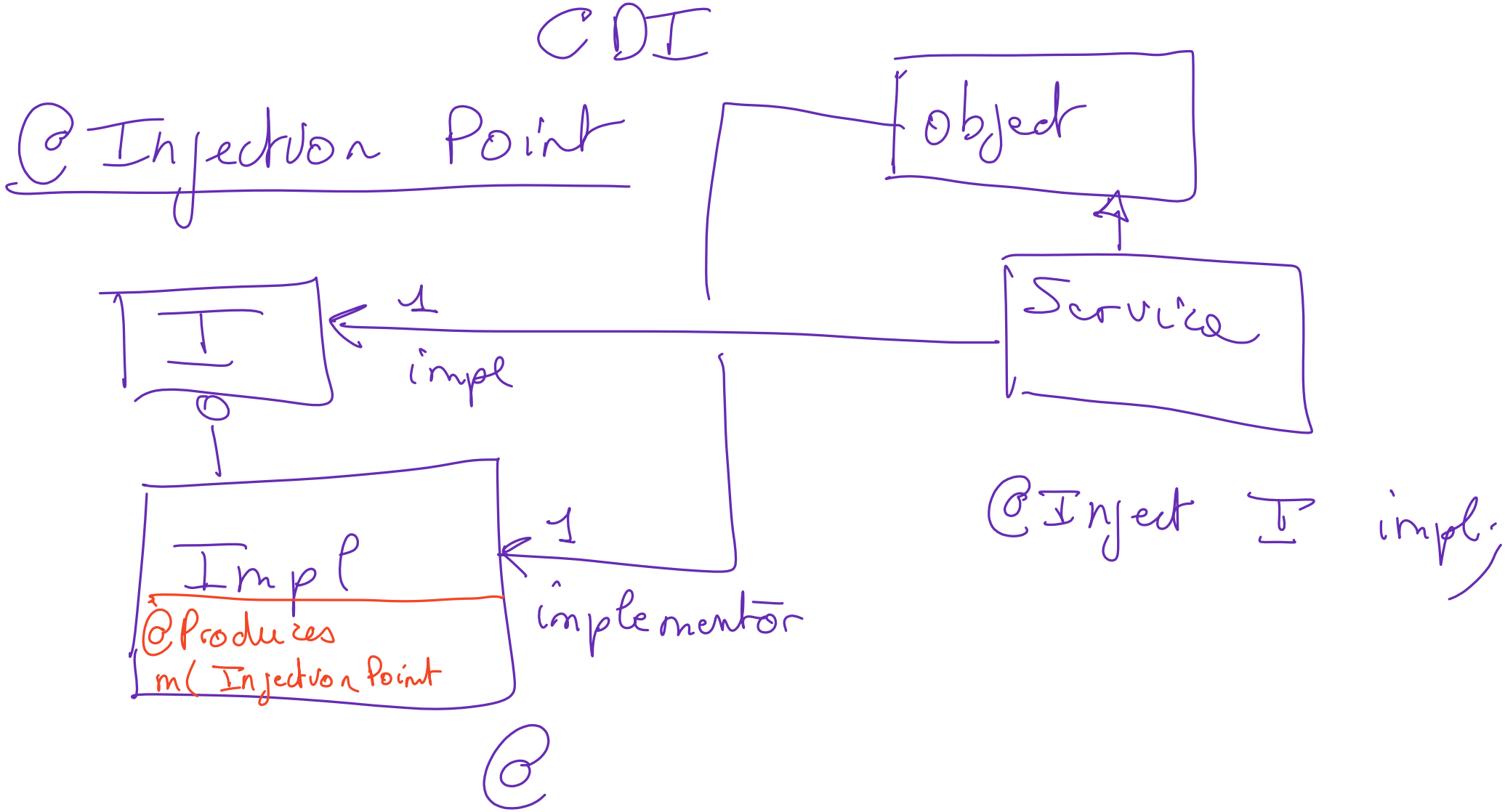
BMT

@AccessTimeout

CMT

get → pas de verrou

set → verrou activé


CDI
@ Injection Point
object
I
1
impl
Service
@ Inject I impl.
Impl
1
implementor
@Produces
m(Injection Point
@

Pattern Java

- @Executor → pattern Asynchrone
- @Decorator → Delegate
- @Observer → Events