# Etude d'un cas concret de Framework : Spring et maquette d'un rapport PDF

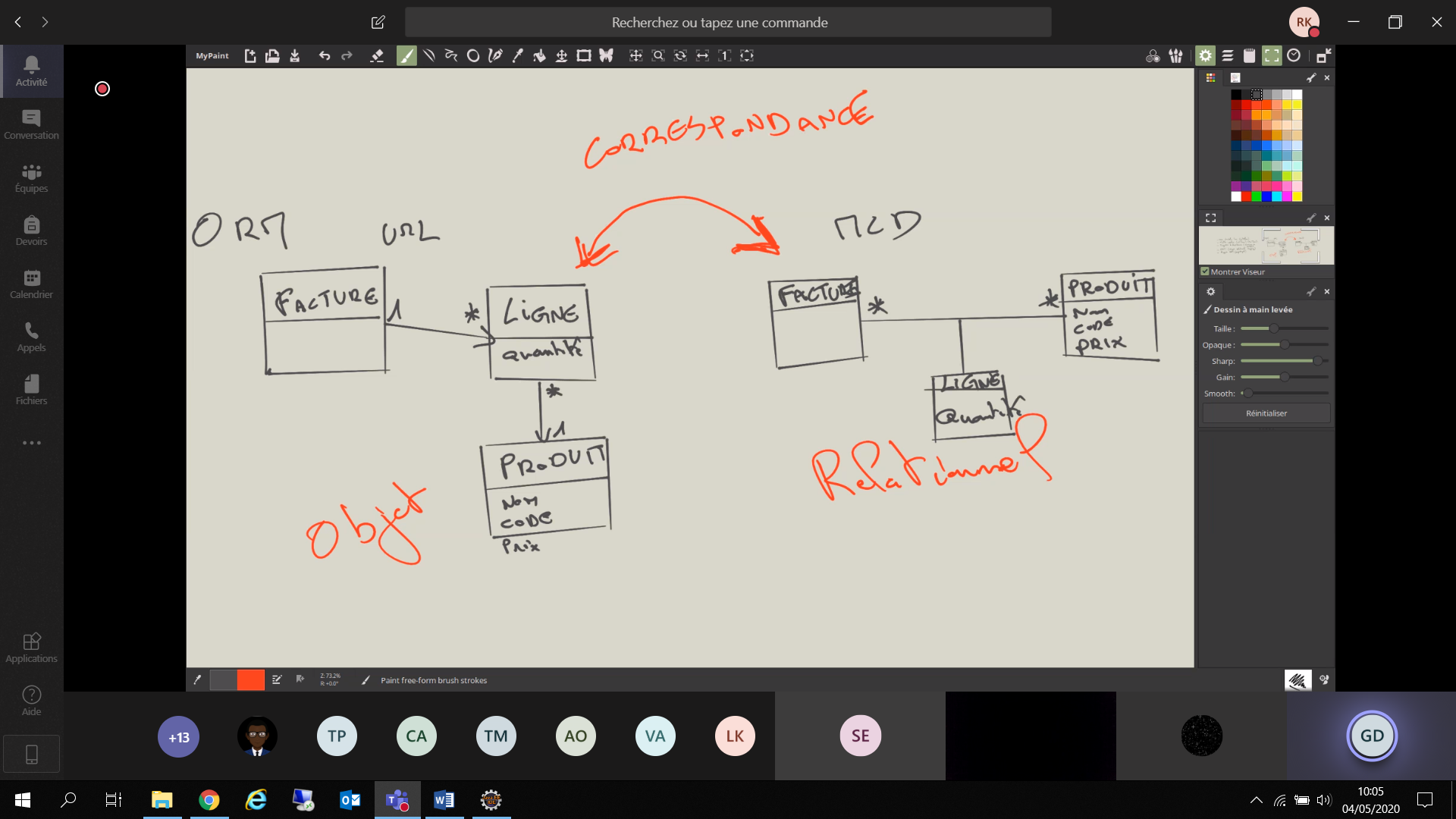
# <https://spoonless.github.io/udev3-spring/>

## ORM (Object Relational Mapping) : <https://spoonless.github.io/epsi-b3-orm/javaee_orm/intro.html>

Exemple d’orm : Hibernate

1er programme de cours : animaux\_orm/

Object Relational Mapping : outil qui permet de mettre en correspondance les diagrammes d’objet et les diagrammes relationnel beaucoup plus rapide et puissant que JDBC ( API de bas niveau ).



Le premier : Hibernate

Un standard qui est utilisé par les ORM comme Hibernate qui encapsule JDBC pour l’emmener vers un plus haut niveau : JPA (Java Persistance API) qui ne s’occupe **que** des relations avec la base de donnée.

Model relationnel a besoin d’un id : @Id

On peux donc rajouter des déclaration sur les attributs de nos entités ex : @Table(name="individu") au dessus de la classe si son nom est différent du nom de la table dans la bdd

Persistence.xml : fichier de configuration de la connexion jdbc a la bdd

Persistence-unit : permet de créer plusieurs connexions de persistence a des bdd ( chaque persistence-unit représente une connexion)

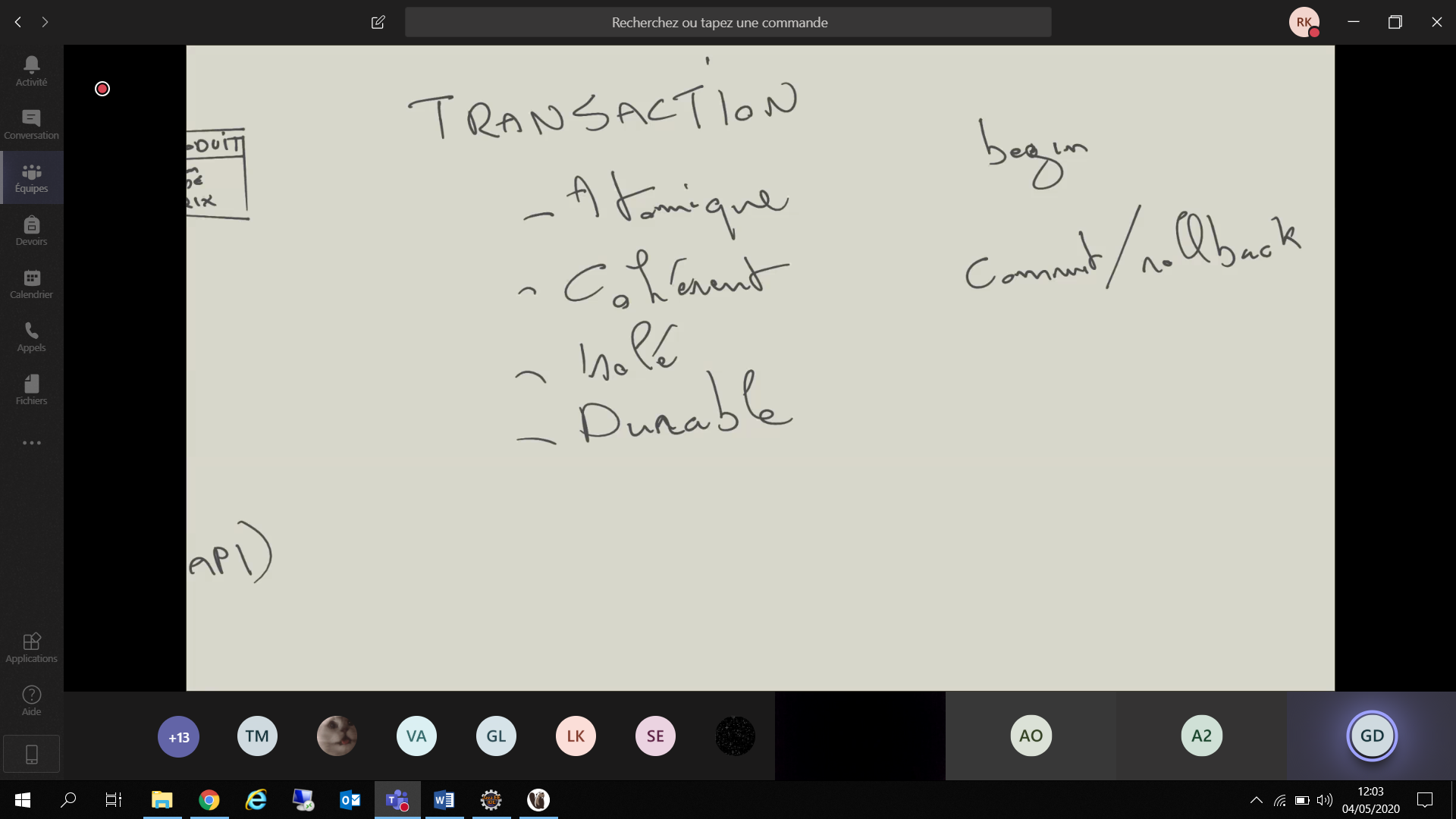
<**property** name=*"hibernate.show\_sql"* value=*"true"* /> : permet d’afficher les requete sql dans la console lorsqu’elles sont executé ( avec em.find(Espece.class,1L) par exemple )

* <https://spoonless.github.io/udev3-spring/javaee_orm/jpa_entites.html>

dans application.java : EntityManagerFactory emf = Persistence.*createEntityManagerFactory*("leNomDeMaPersistence-unit =le nom de la connexion a la bdd");

emf permet ensuite de créer des entity manager qui vont nous permettre de manipuler les objets de la base avec des find, des persist etc…

Attention aux Transactions (em.getTransaction().begin ()-…- em.getTransaction().commit())



Find : retourne l’entité de type donnée et de l’Id donné, null si il ne le trouve pas.

Persist : rendre persistant dans la BDD l’objet donné

espece.setNom("élapidé");

em.getTransaction().begin();

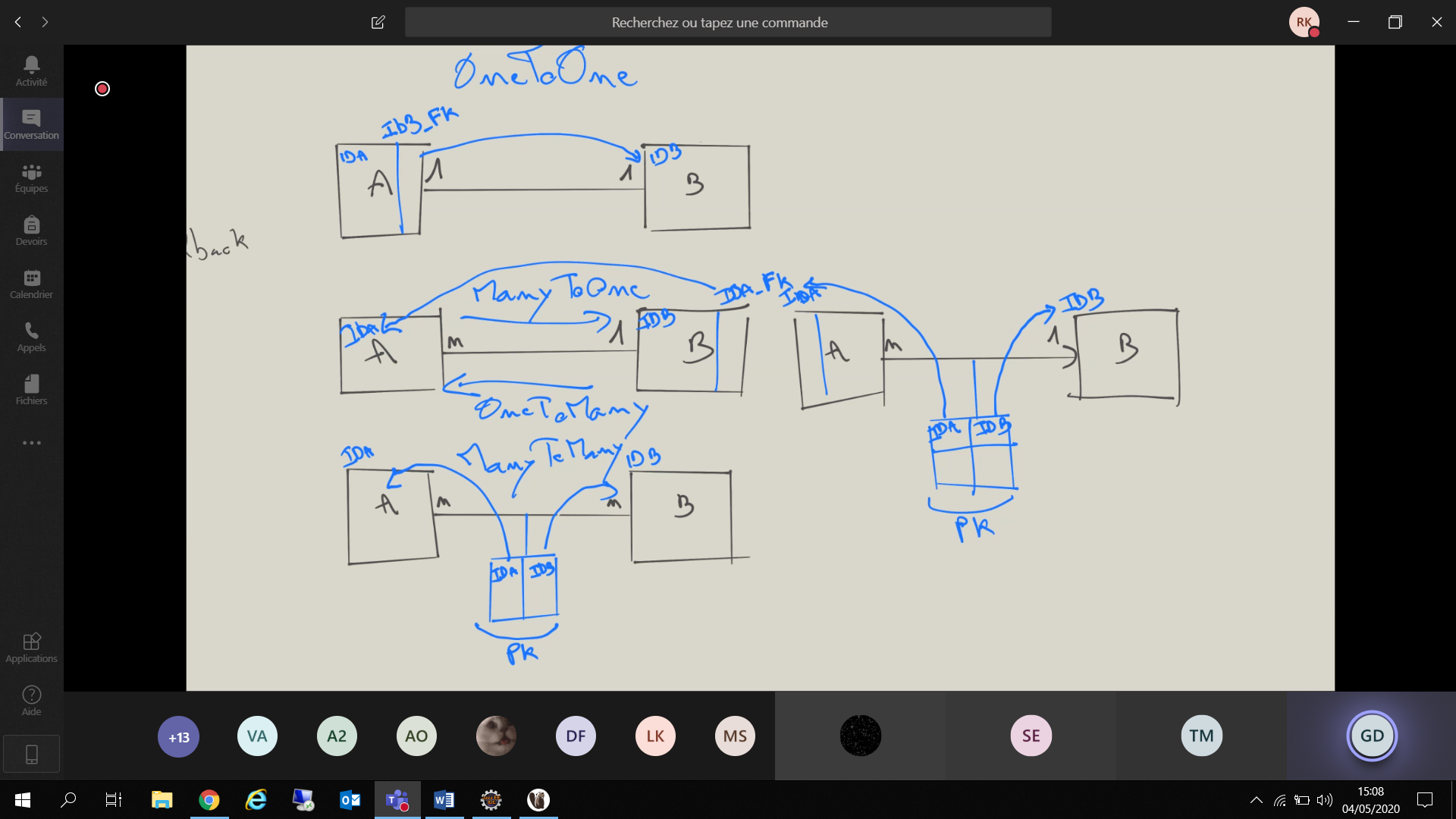
em.persist(espece);

em.getTransaction().commit();

Trois facon de faire du requetage avec JPA :

* Requetes SQL ou requetes natives
* Requetes par programmation ( en utilisant des objet java ) ("select i from Individu i", Individu.class )) : difficile a ecrire et lire
* Requetes JPQL ( Java Persistence Query Language) :
  + Exemple : List<Animal> listAn2 = em.createQuery("select a from Animal a where regime = :regime", Animal.class).setParameter("regime", Regime.carnivore).getResultList();

Les relations entre tables : <https://spoonless.github.io/udev3-spring/javaee_orm/jpa_relations.html>



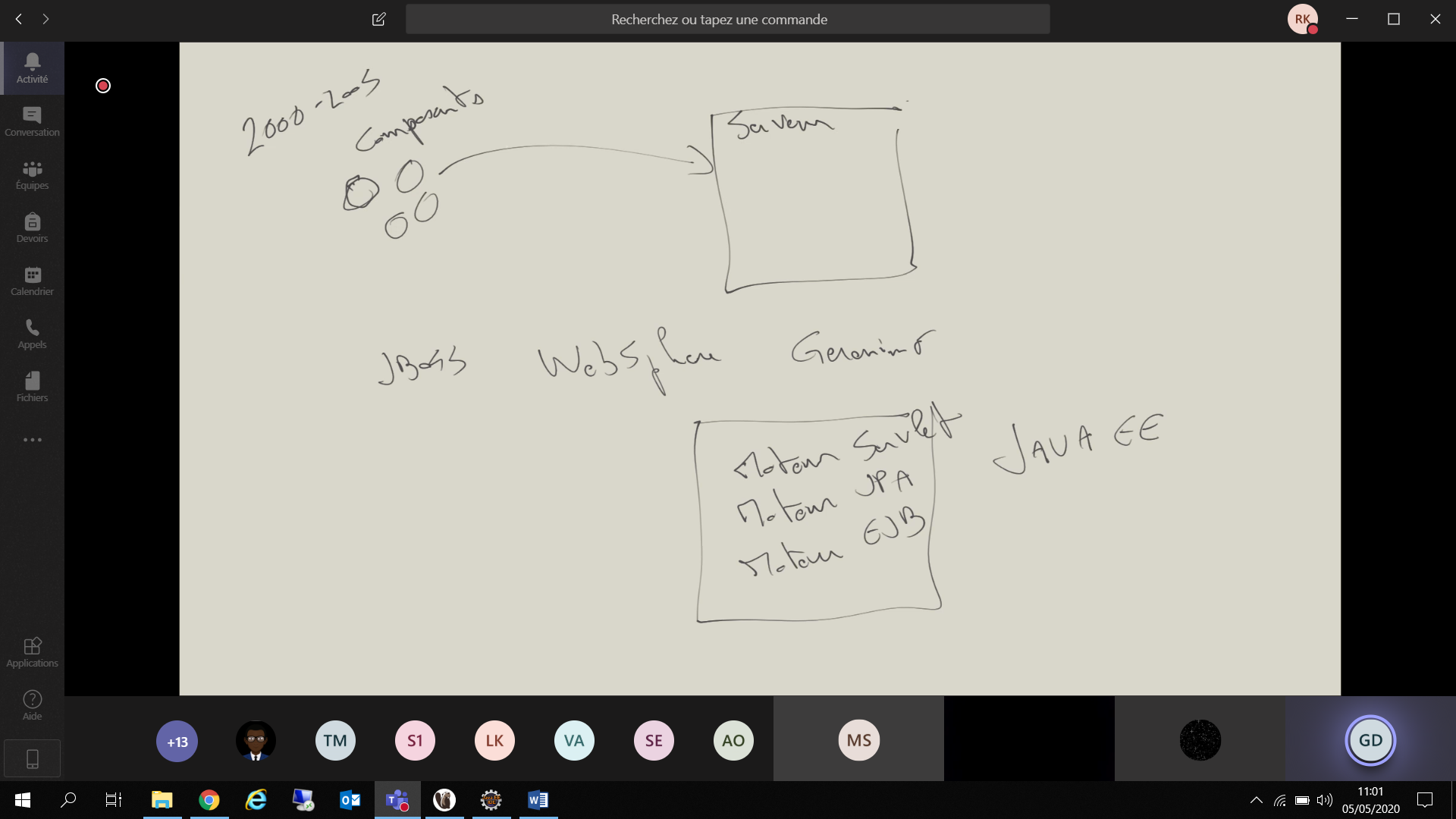
ManyToOne => notreObjet de notre class a gauche du To

## **Spring**

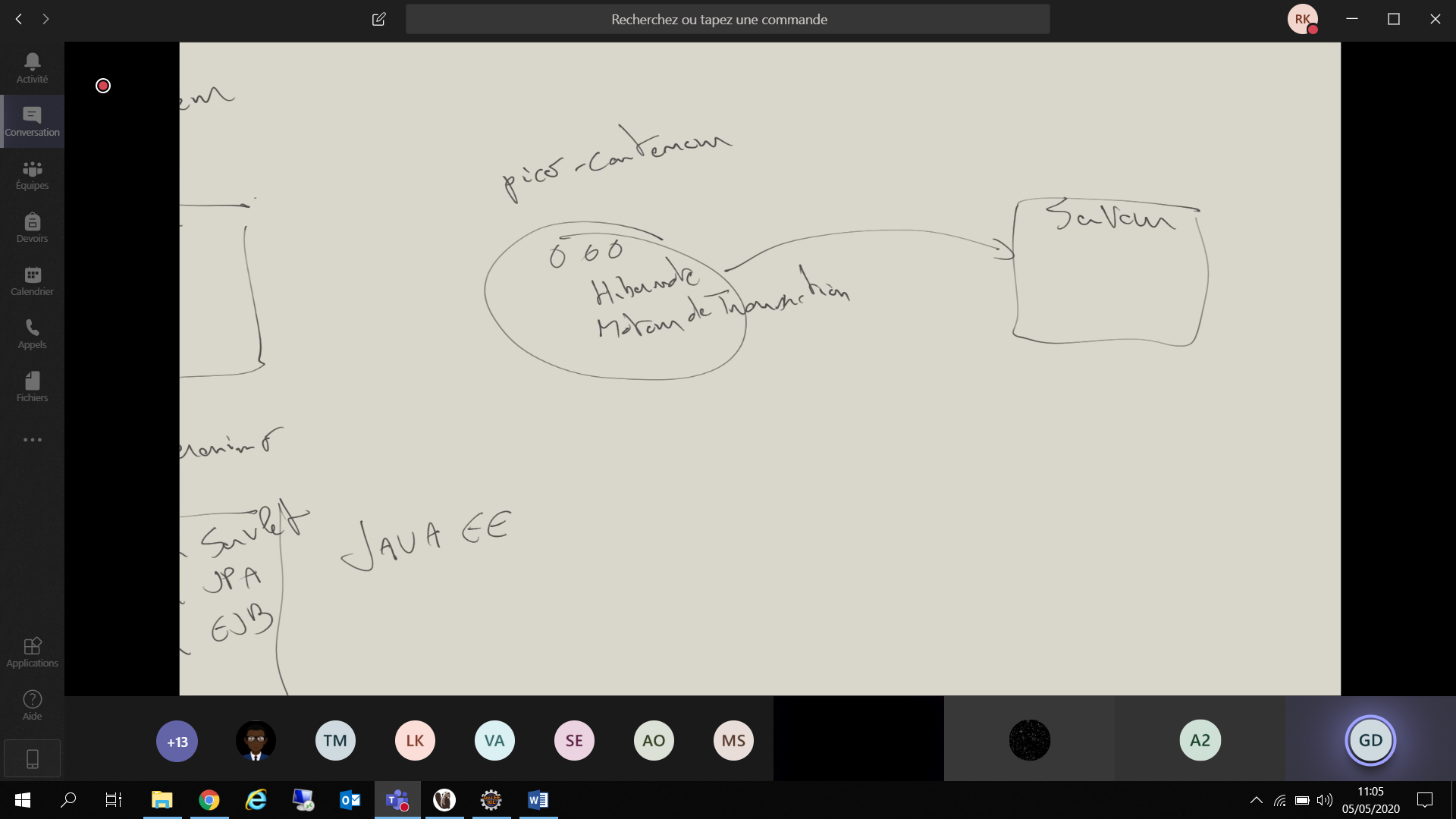
2eme programme de cours : l’appli web /animoz

## MVC

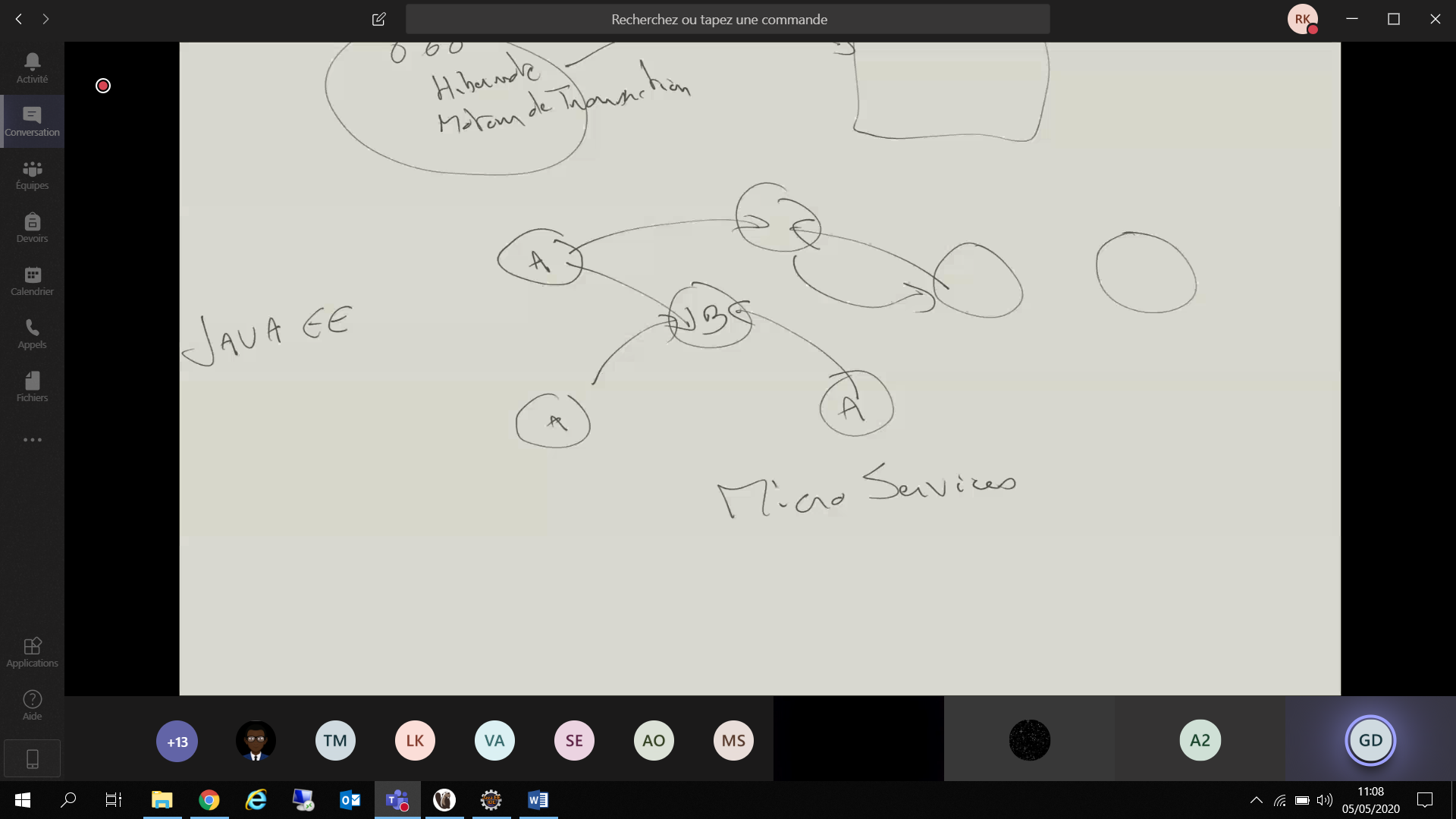
Conteneur

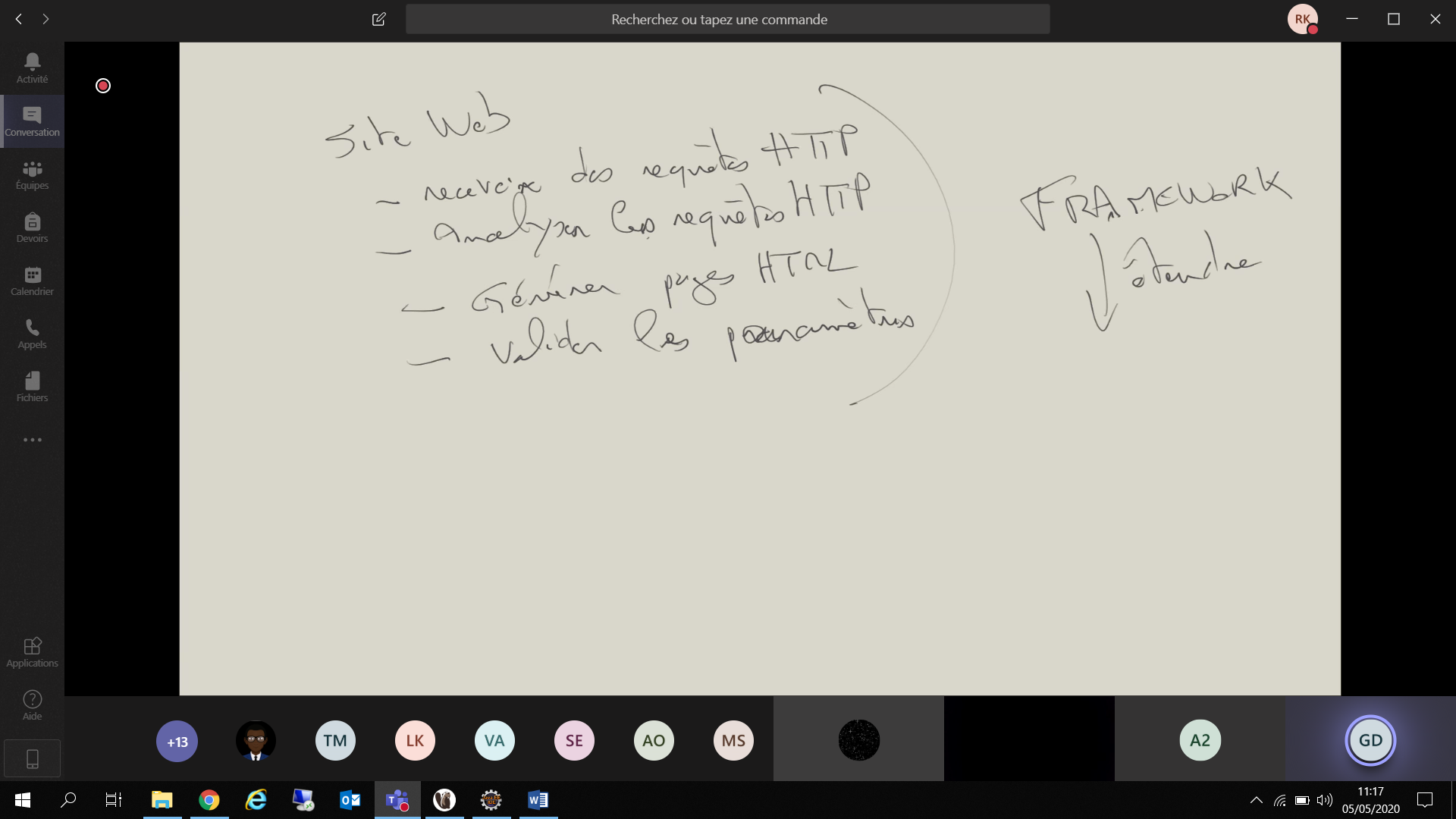


Au début on avait ca : des applications  « monolithiques » puis on est passer en 15 ans a des plus petits serveur conteneur qui ne contenaient que les applications qui elles, contenaient tout les composant dont elles avaient besoin :



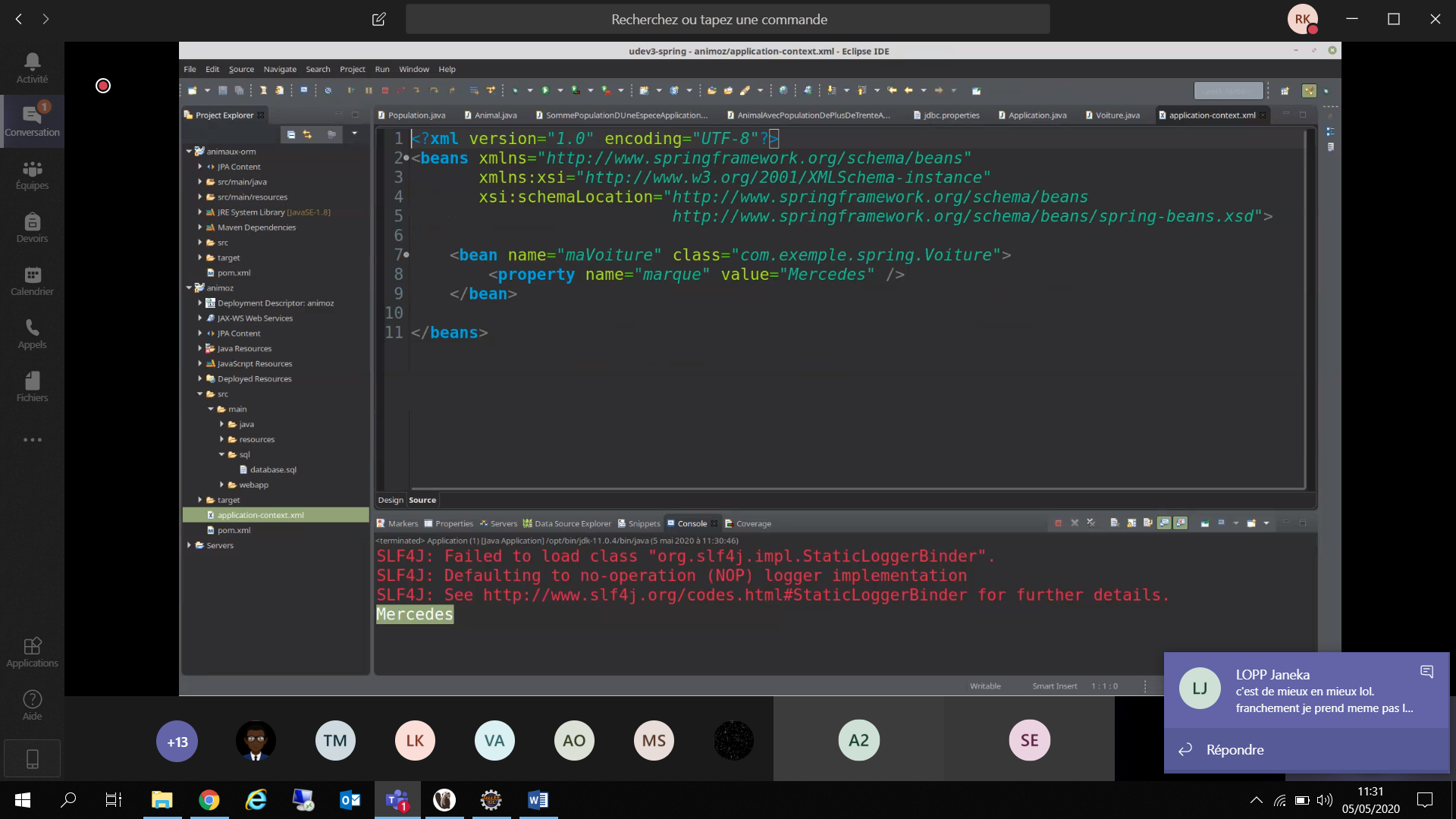
Idée general du createur de Spring framework





## Injection de dépendances (Inversion de contrôle)

Le boulot de base du Spring framework est de créer des objets java a partir des déclarations de beans dans application-context.xml en utilisant l’inversion de contrôle IoC:

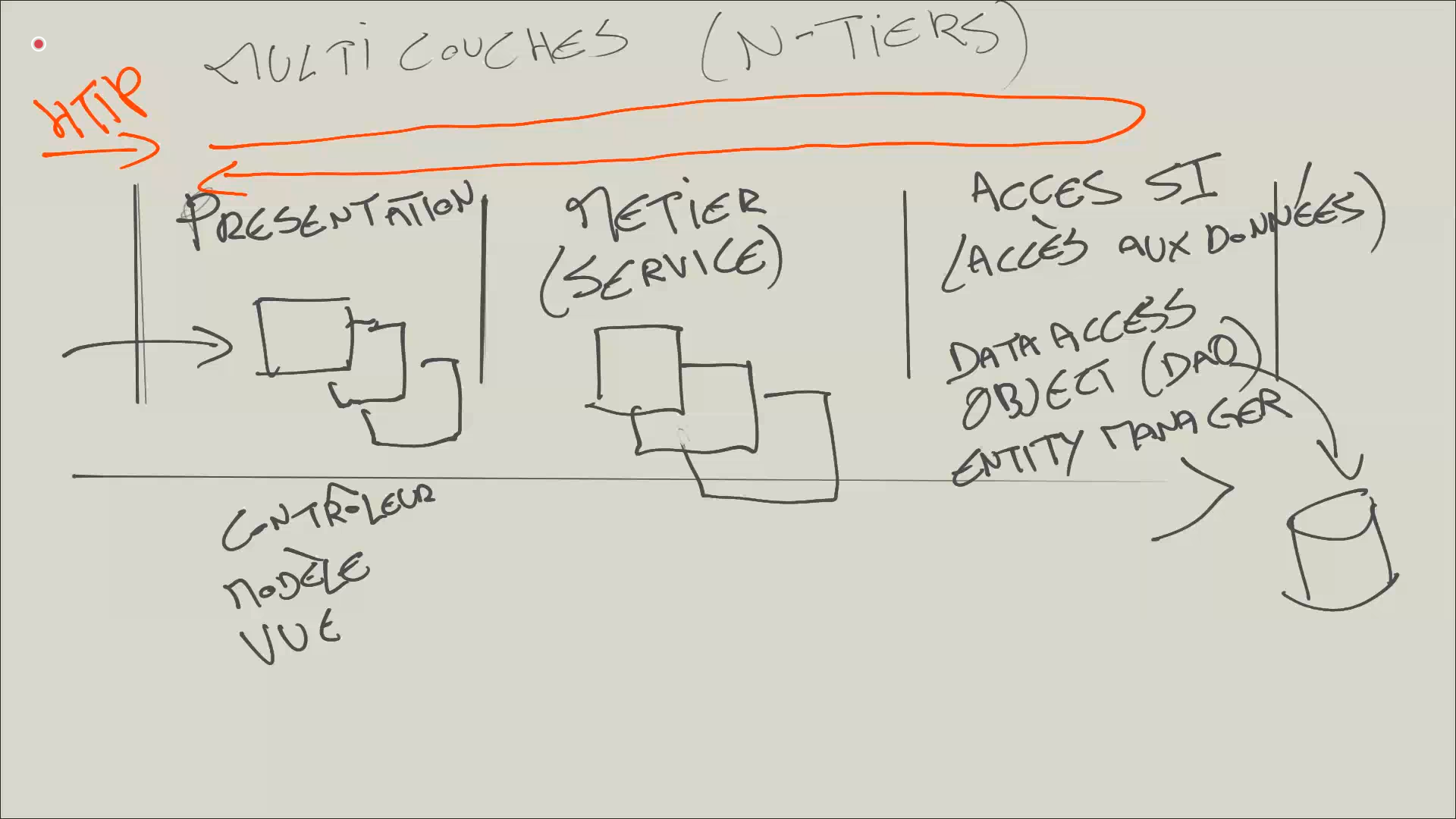


## Développement Application Multi-couches (N-Tiers) (3-Tiers)

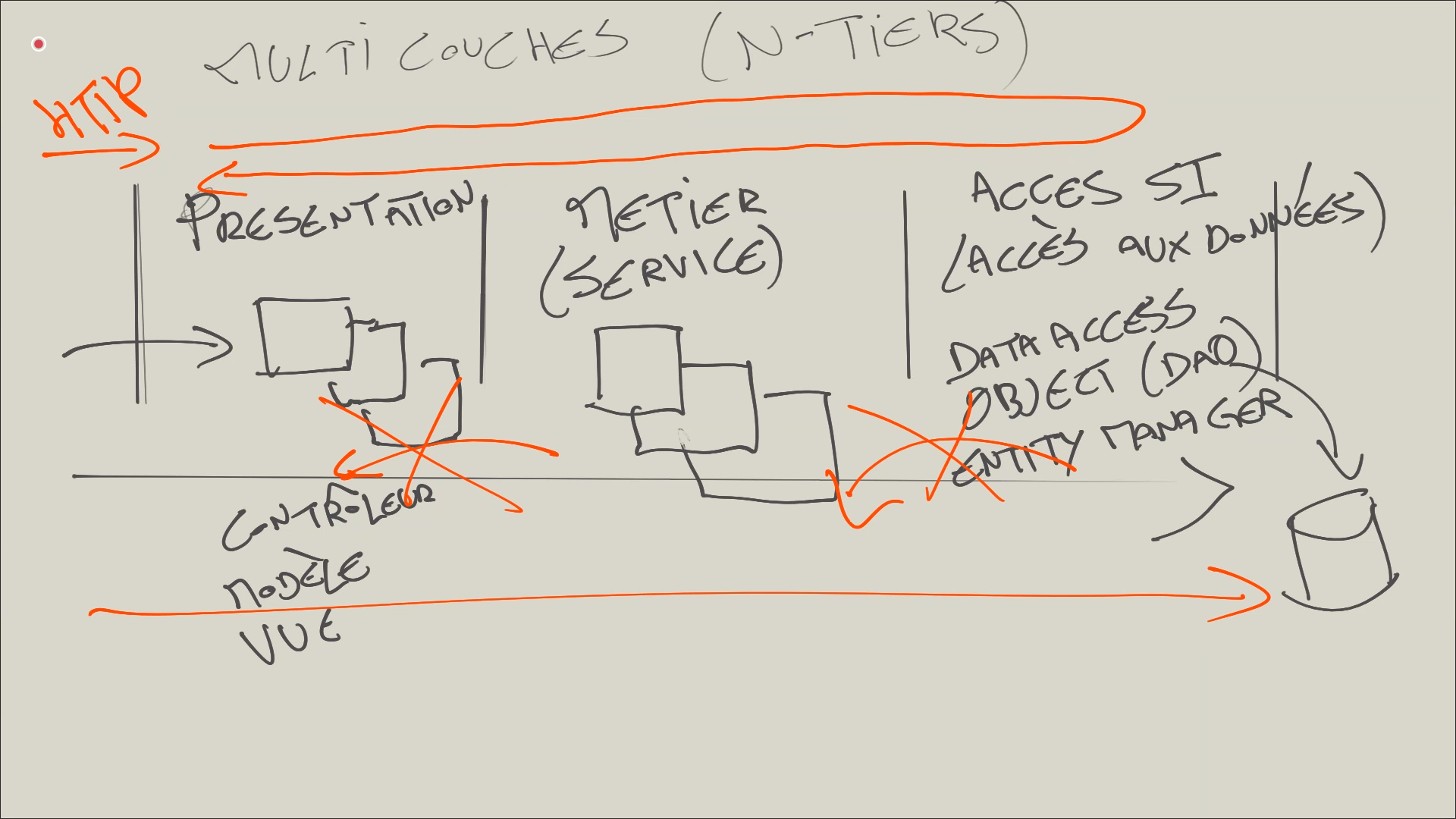
Faire ressortir les responsabilités des différents composants.

Une couche = une responsabilité macro :

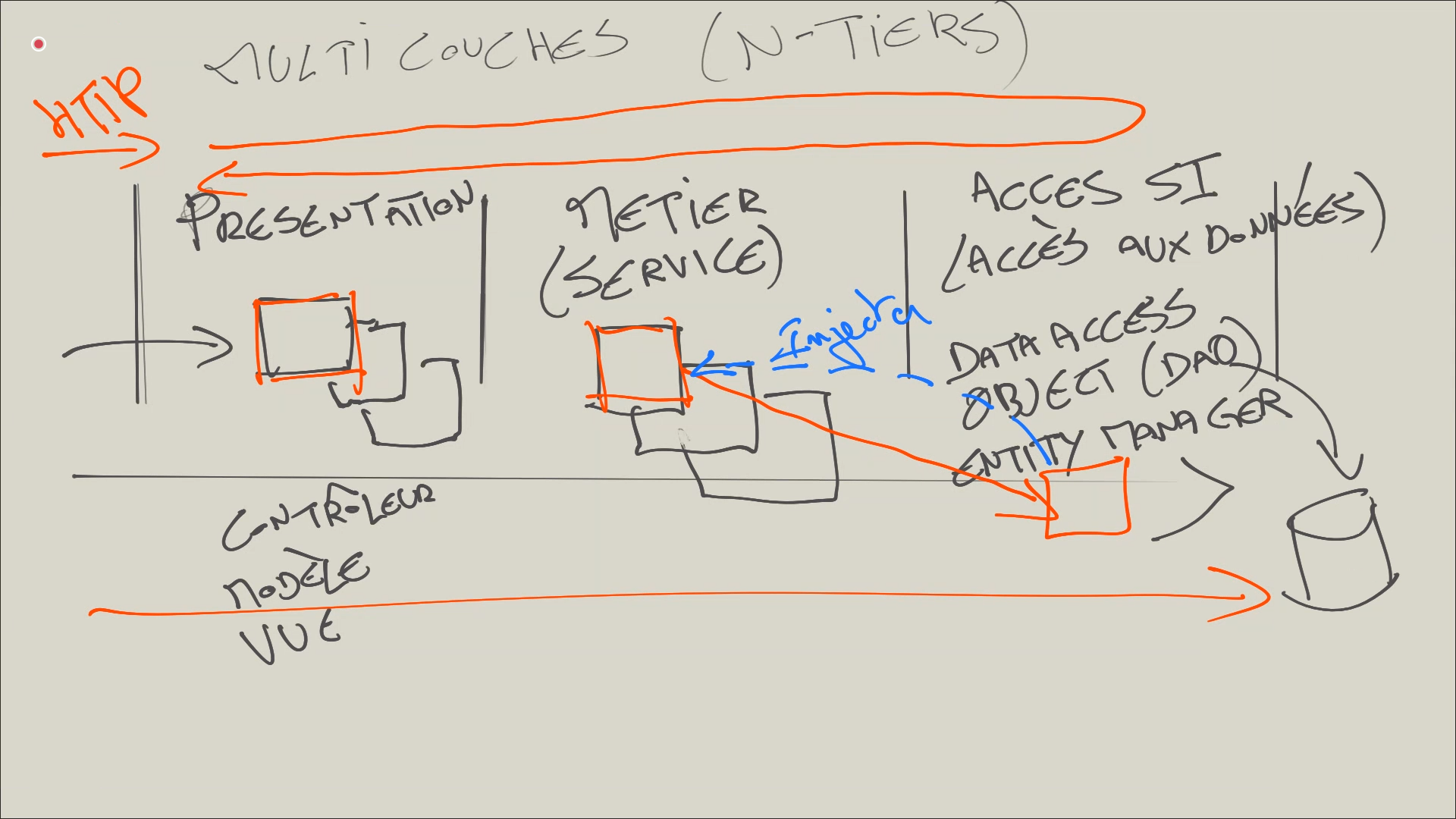
* 1 couche de présentation(Model,Vue,Controller + toutes les potentielle class de formatage des données)
* 1 couche métier ou services ( toutes les fonctionnalités de l’application )
* 1 couche d’accès au SI/ accès aux données (DAO,EntityManager)

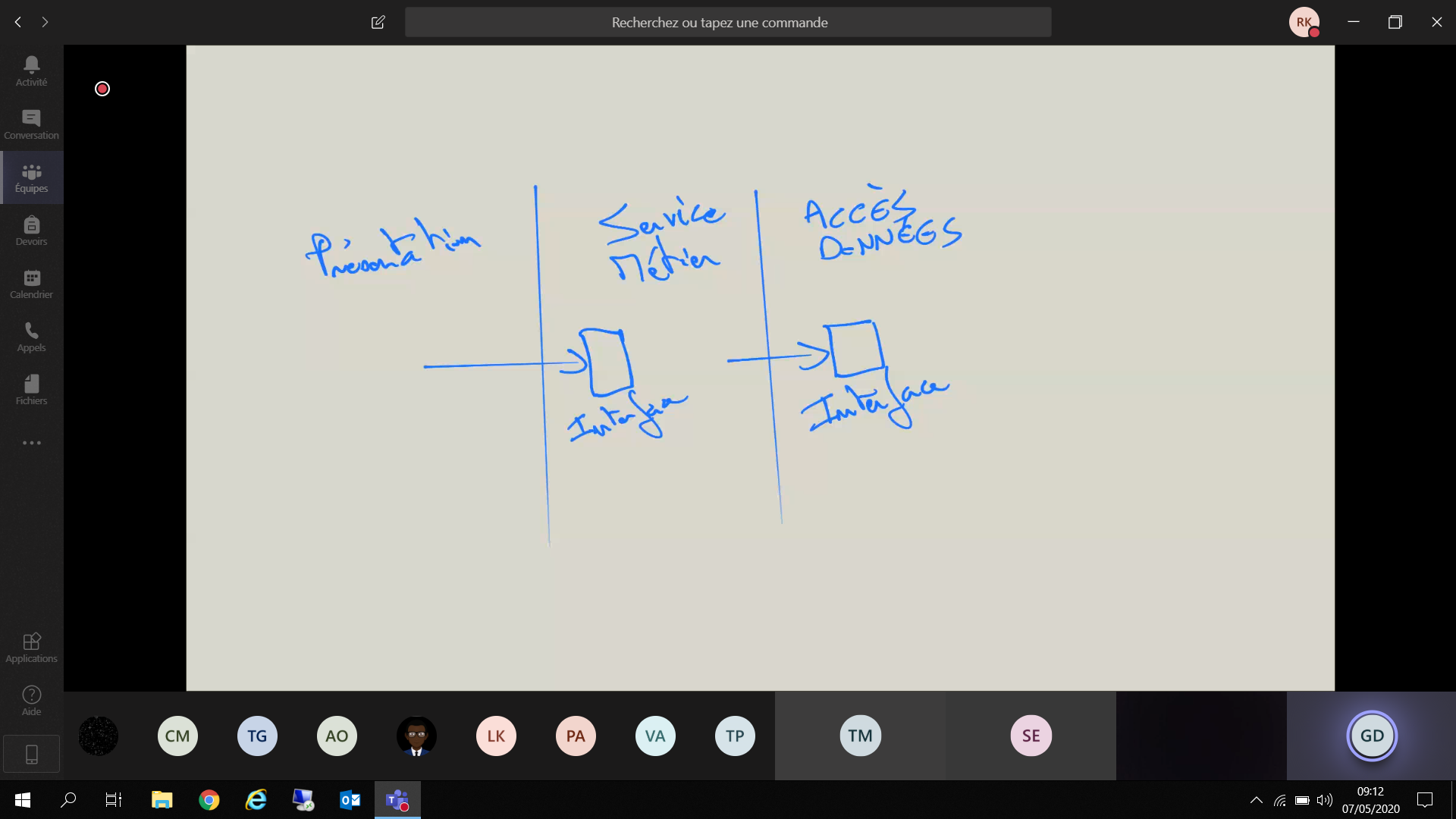


Sens de la requète



Du coup : (les fleches orange sont des dépendances)





DTO : Data Transfert Object = Objet qui contient des données à transmettre (ex : un wagon )

Spring Data : permet de laisser a Spring le soin de créer les DAOs

Pour mettre des images dans un projet Spring :

Src/main/web-app/ressources

## Génération d’état imprimé, génération de rapport de PDF avec JasperReport

Utilisé énormément dans les application d’entreprise : apporte le coté BI (Business Intelligence)

$F{nomvar} = field/champ

$V{nomvar} = variable

* Generation de rapport via programme Java

1ere etape : compilation du rapport

2eme etape : recuperation des données, on remplis le rapport avec 🡪 JasperPrint

3eme etape : On export ces donnée dans un pdf