Développement des applications Web basées sur les microservices (Partie Backend)

Responsable du cours:
Dr. Ing. Mariam LAHAMI
Dr. Ing. Sihem LOUKIL
Email: mariem.lahami@enis.tn
sihem.loukil@enis.tn

Présentation du cours

- Objectifs
  - Développement d'applications Web ***robustes et scalables*** basées sur
    - Spring boot côté serveur
    - Angular côté client
- Pré-requis
  - Maîtrise du langage Java (Java Standard Edition)
  - Architecture client/serveur
  - Maîtrise du développement WEB: HTML5, CSS3, JS, AJAX, Servlet et JSP, etc.
  - Connaissance des **patrons MVC** (*Model, View, Controller*) et **DAO** (*Data Access Object*) ;
  - Maîtrise du JPA et JEE

Déroulement du module

- Volume horaire : 32,5 h CI et 13h TP
  - Partie Back end
    - Responsables: Mme Mariam Lahami et Mme Sihem Loukil
  - Partie Front end
    - Responsable Mme Imene Lahyani

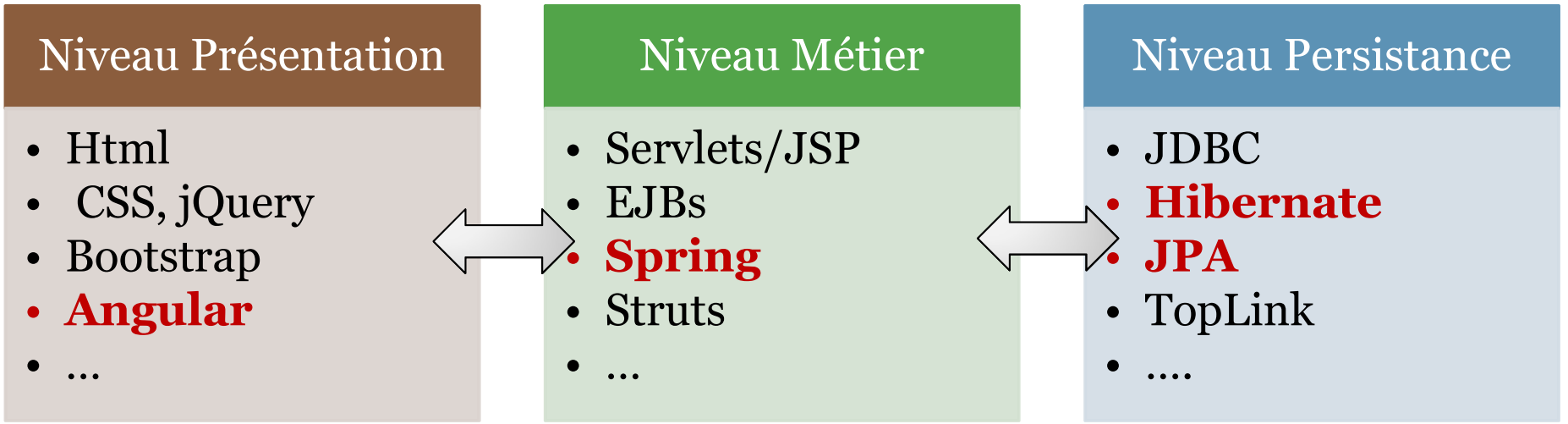
- Notes du module
  - Note de contrôle continu : Présence, travail en classe et Note final d'un projet
  - Note d'examen

Chapitre 1 : Introduction

- Architecture Web
- Exemples d'architectures Web
- Application monolithique vs Application microservices

Architecture Web classique (1/3)

- Une architecture distribuée ***multi-niveaux (ou multi-tiers)***
- Trois couches fondamentales:
  - Couche **présentation ou client**
  - Couche **métier**
  - Couche d'**accès aux données ou de persistance**



Architecture Web classique (2/3)

Couche métier

- Objectif:
  - Traiter les données de l'application
  - En se basant sur des règles qu'on appelle « Règles métiers »
- Composants: tournant sur un serveur :
  - Serveur Web: Tomcat
  - Serveur d'application: Wildfly, Websphere, etc.

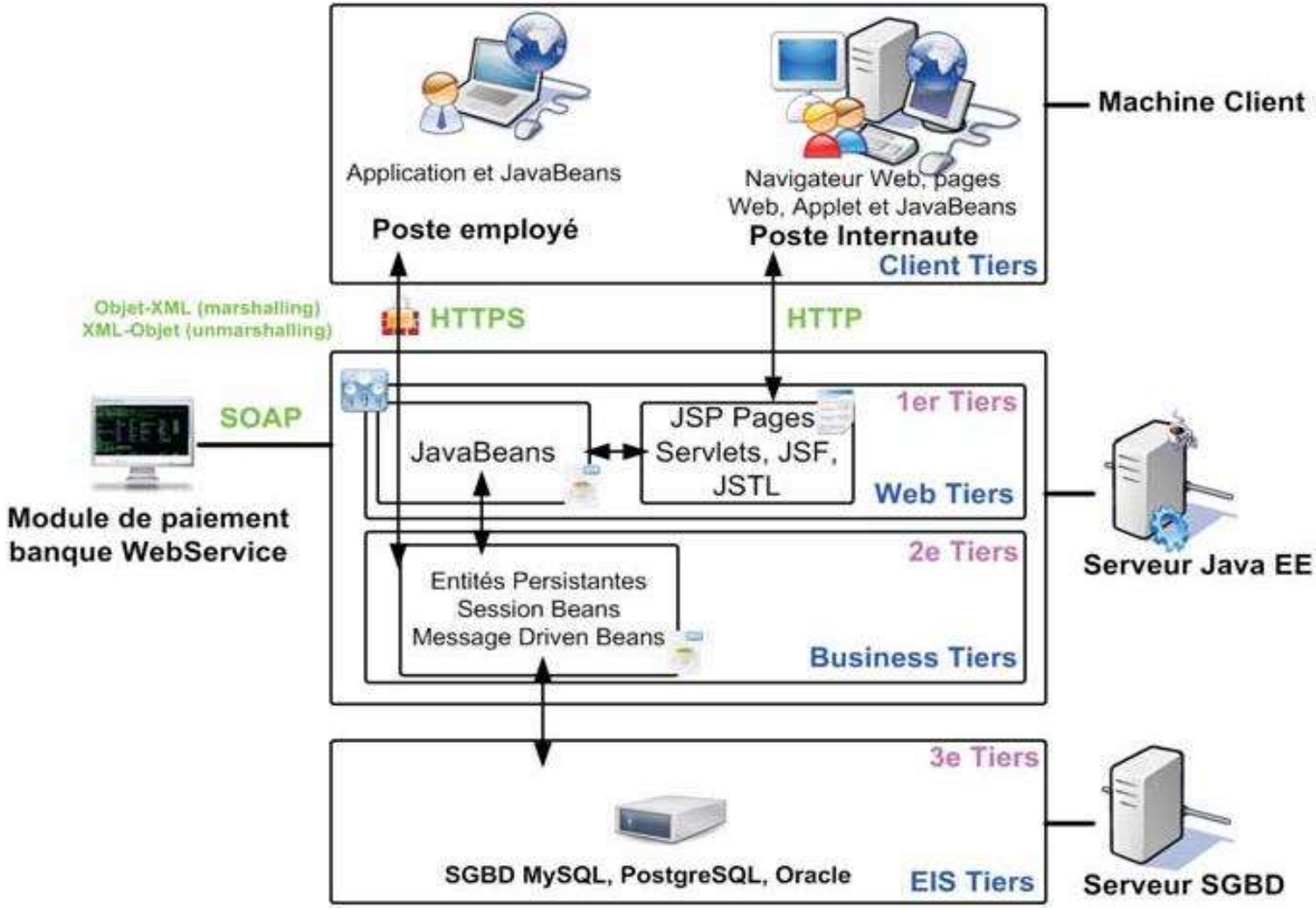
Architecture Web classique (3/3)

Couche persistance

- Objectif: Enregistrement des données de l'application sur un support physique

- Cette couche peut avoir accès à:
  - Des bases de données relationnelles
  - Des bases de données orientées objet telles ObjectStore et les bases de données NoSQL
  - Des ERPs (Enterprise Resource Planning)
    - Gestion des affaires d'une suite d'applications
    - Collecter, stocker, gérer et interpréter les données provenant de nombreuses activités

Exemple d'architectures Web : Architecture JEE



Exemple d'architectures Web : Architecture SOA

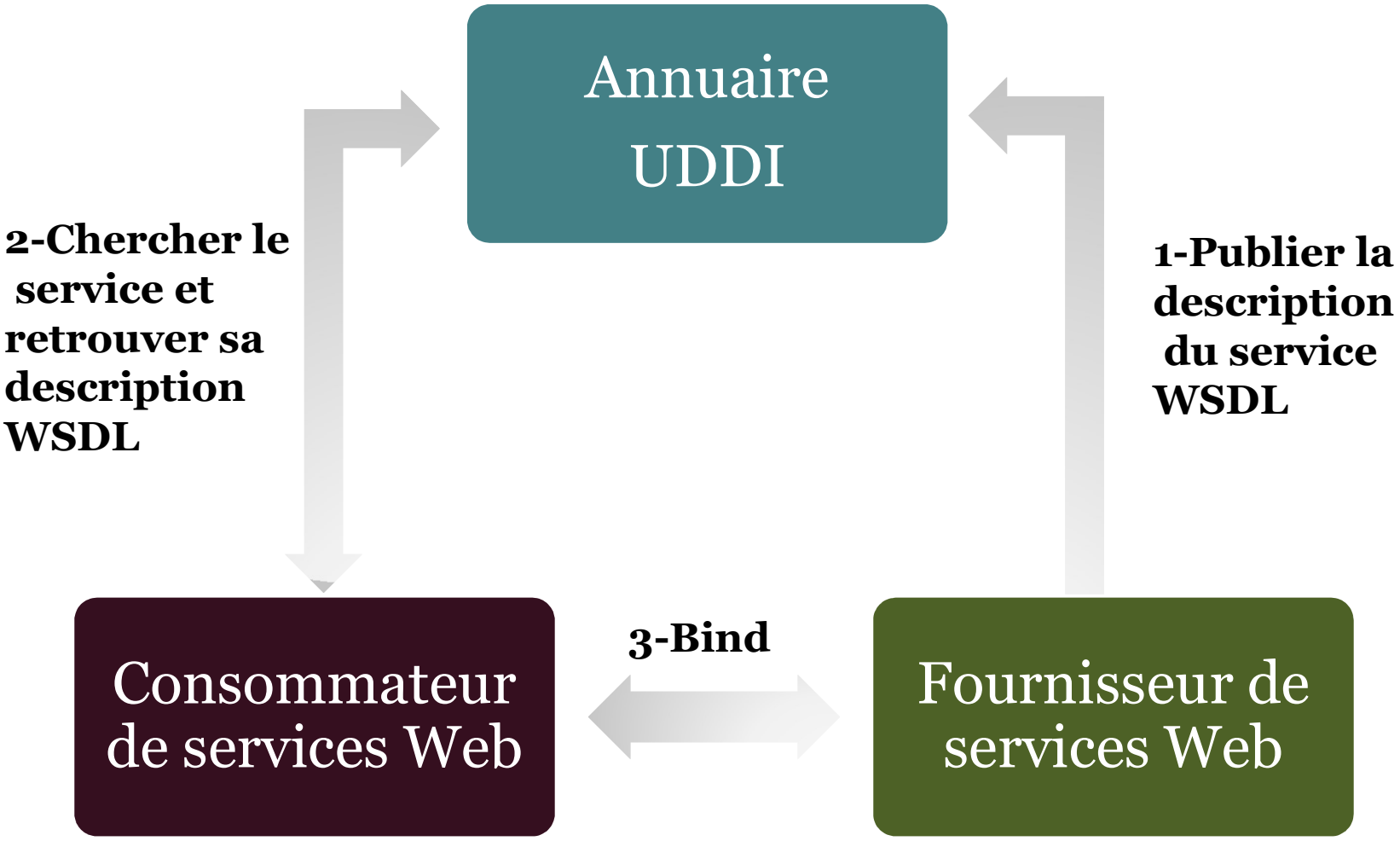
- Architecture orientée services (Service Oriented Architecture (SOA) : un type d'architecture
  - reposant sur les standards de l'Internet
  - permettant à des applications de communiquer sans préoccupation des technologies d'implantations utilisées
  - Favorisant l'interopérabilité
  - Se basant sur les technologies XML
- Les services Web sont une implémentation de l'architecture orientée service

Exemple d'architectures Web : Architecture SOA

- Les Web Services sont des composants Web basés sur les protocoles standard d'Internet: HTTP et XML
  - SW est décrit par XML (WSDL)
  - SW interagit via XML avec d'autres applications (Protocole SOAP)

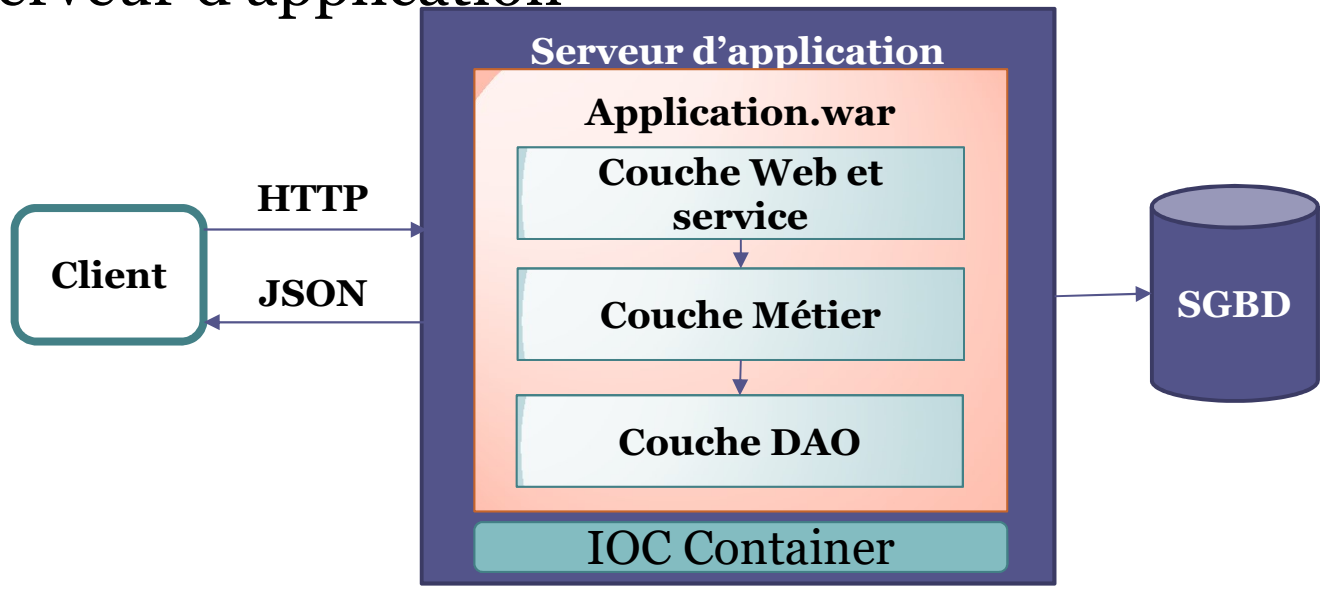
- Un objet métier qui peut être déployé et combiné sur Internet avec une faible dépendance vis-à-vis les technologies et des protocoles.

Exemple d'architectures Web : Architecture SOA



Application monolithique

- Application Web construite comme une seule entité (.jar, .war, .ear) déployée d'une manière unitaire dans un serveur d'application

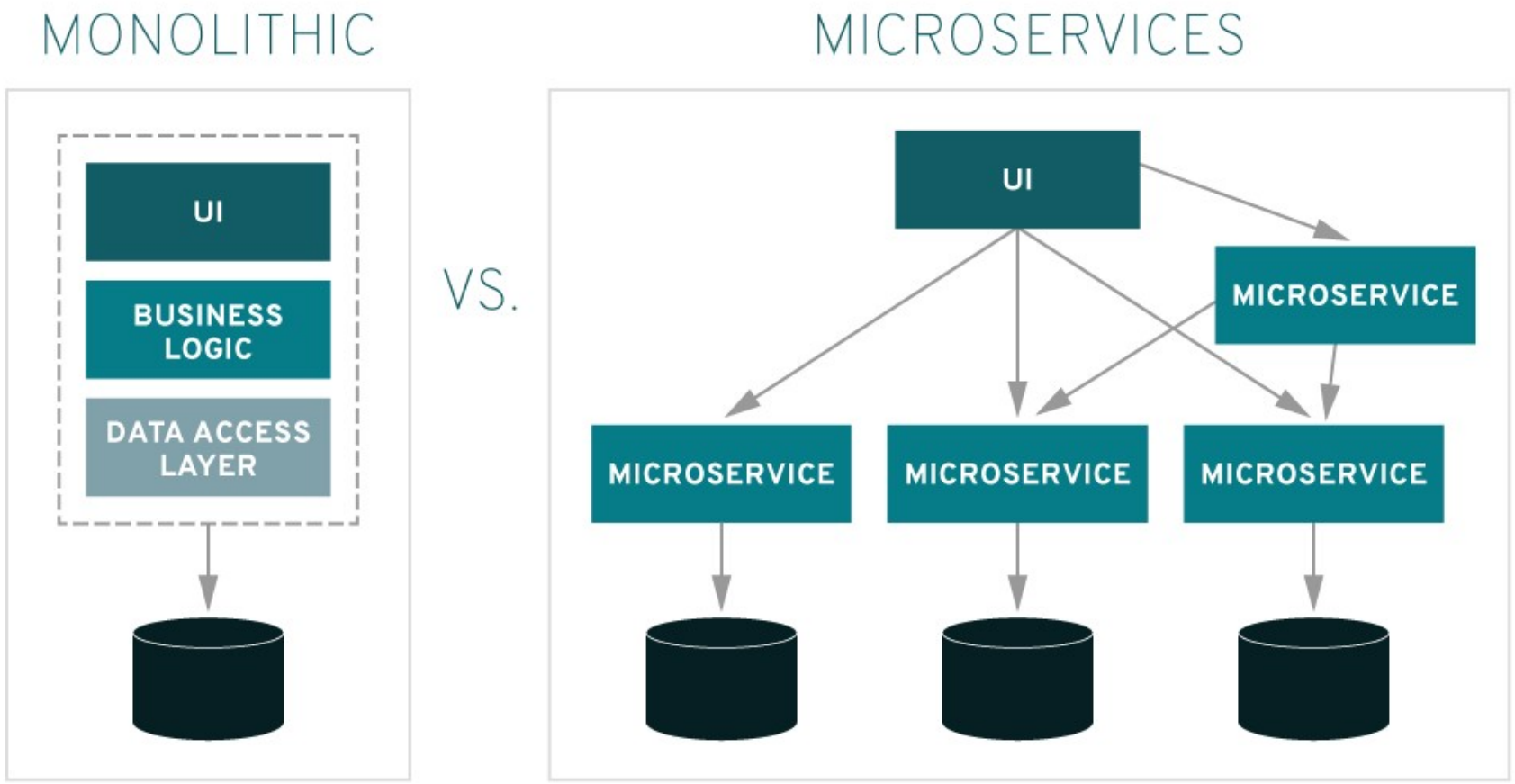


=> Toute modification nécessite la compilation et le déploiement de toute l'application

Limites des applications monolithiques

- Elles centralisent tous les exigences fonctionnelles
- Elles sont réalisées dans une seule technologie.
- Chaque modification nécessite de :
  - Tester les régressions
  - Redéployer toute l'application
- Difficile à faire évoluer au niveau fonctionnel
- Livraison en bloc (Le client attend beaucoup de temps pour commencer à voir les premières versions)
- Couplage fort entre les équipes de développement

Solution : Architecture microservice (MSA)



Application basée sur les microservices

Concepts

- Une application = un assemblage de « petites » unités autonomes et indépendantes
- Chaque microservice réalise une seule fonction métier
  - Ex : vente, CRM, comptabilité, front-end, GUI, etc.
- Chaque microservice est développé, testé et déployé séparément des autres.
- Techniquement, les services sont
  - Programmés dans des langages hétérogènes
  - Exécutés dans des processus séparés
  - Liés à leurs propres supports de persistance
  - Développés et déployés dans des projets distincts

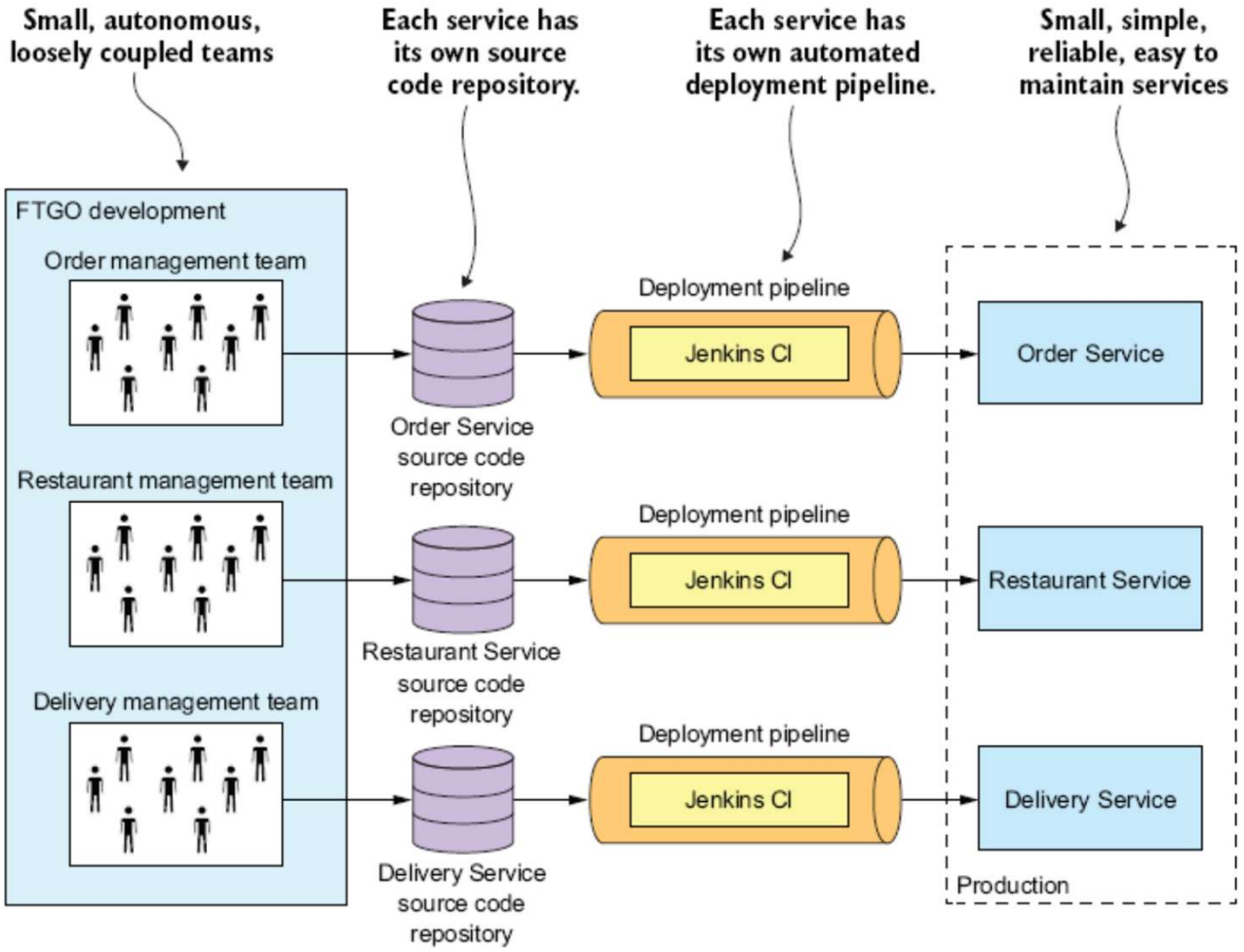
Application basée sur les microservices

Avantages

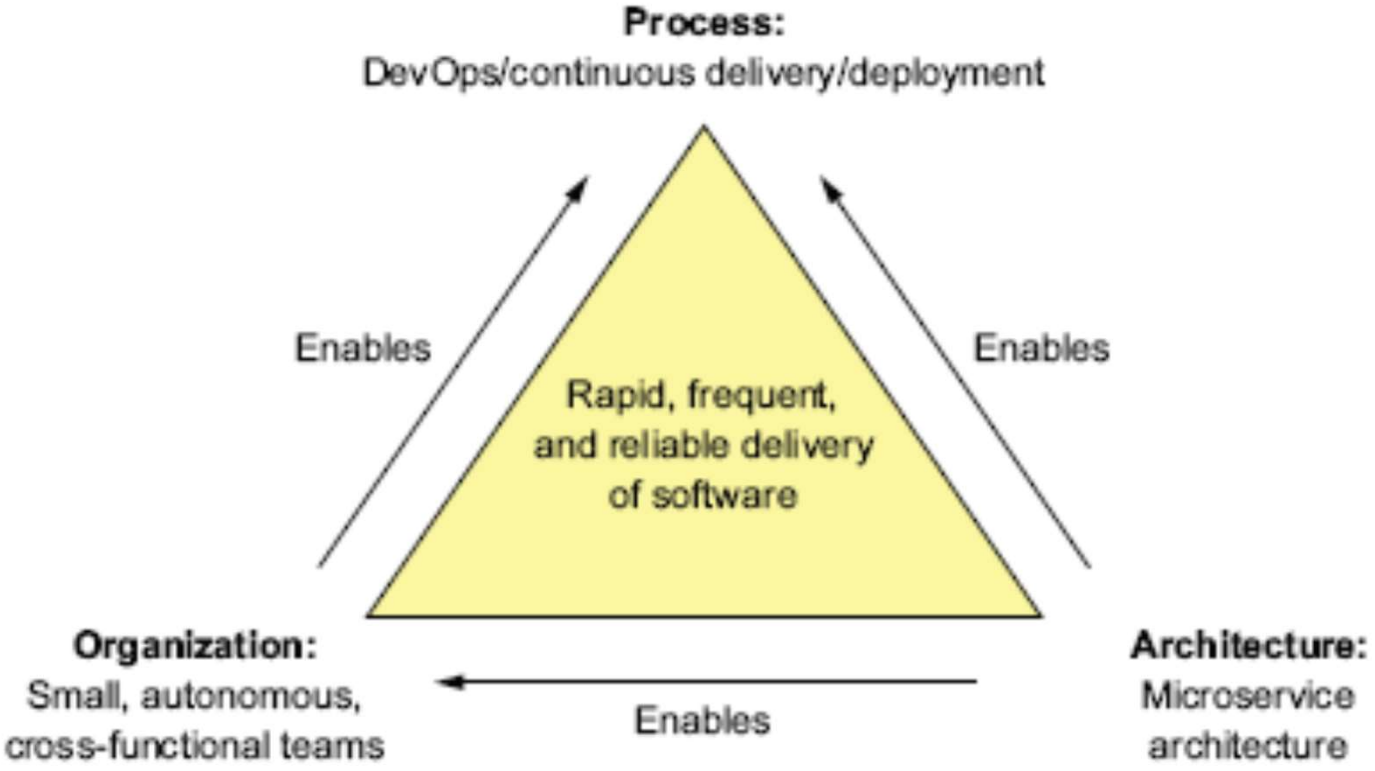
- Couplage faible: chaque microservice est physiquement séparé des autres,
- Indépendance relative entre les différentes équipes
  - Des équipes plus petites : agilité, autonomie, efficacité
  - Chaque équipe est responsable d'un service
    - Choix technologiques, conception, déploiement, maintenance

    => Développement axé sur la fonctionnalité
- Facilité des tests et du déploiement ou de la livraison continue.

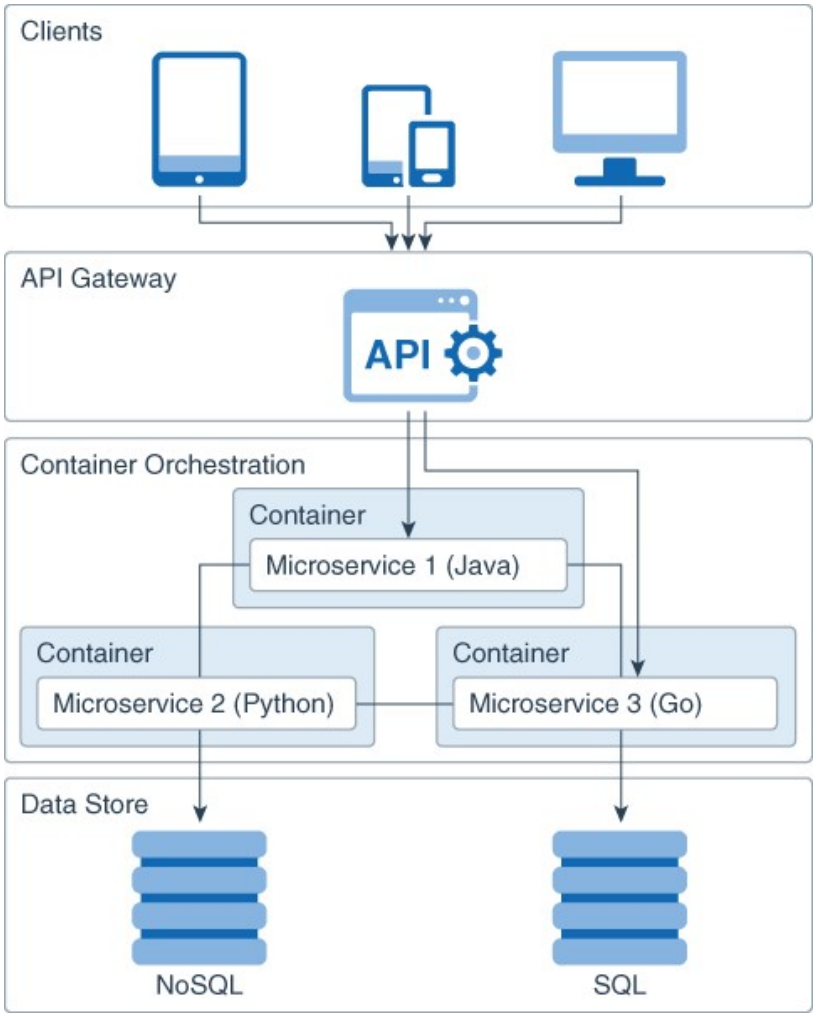
Application basée sur les microservices
Avantages



Vers plus d'agilité

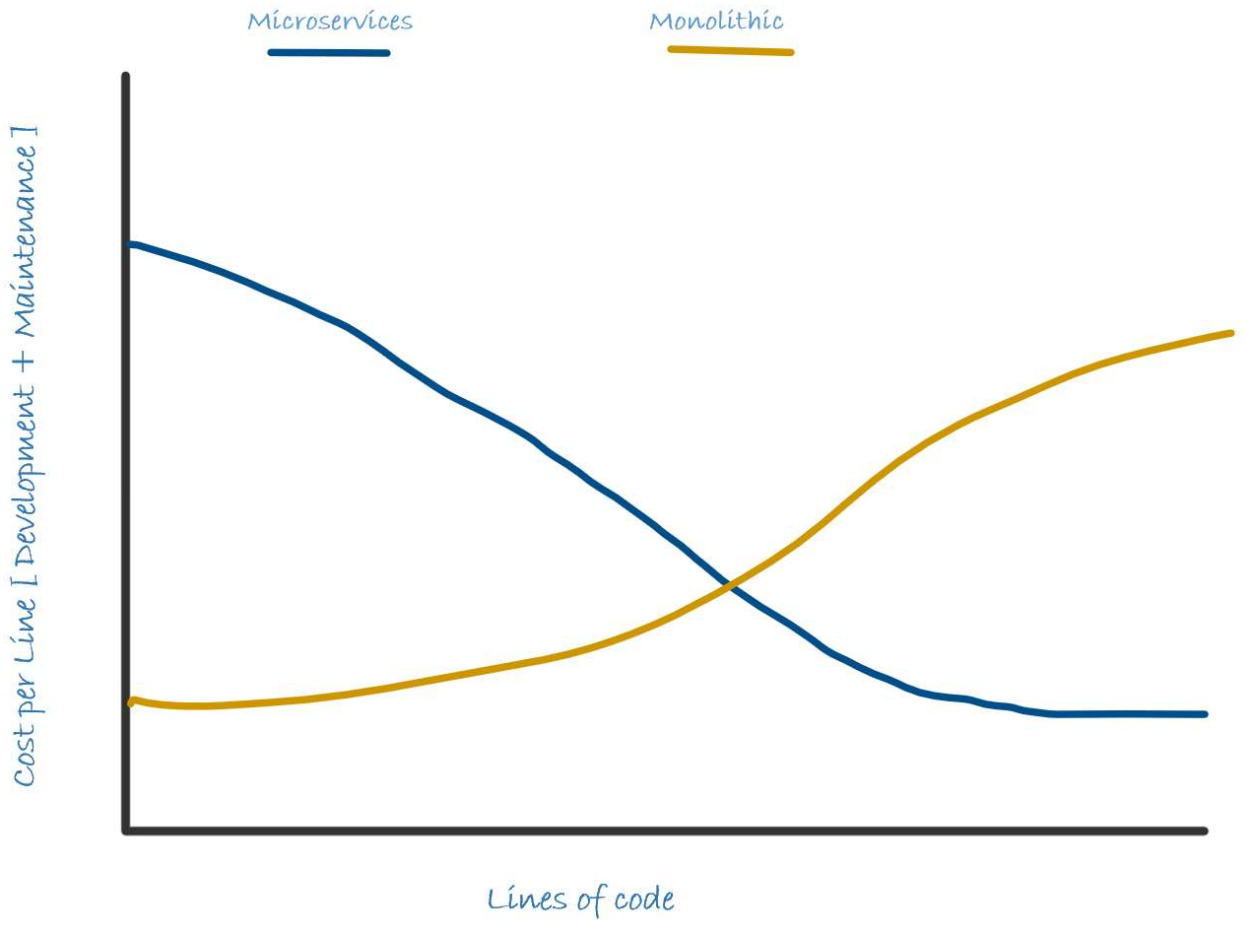


Vers plus d'interopérabilité

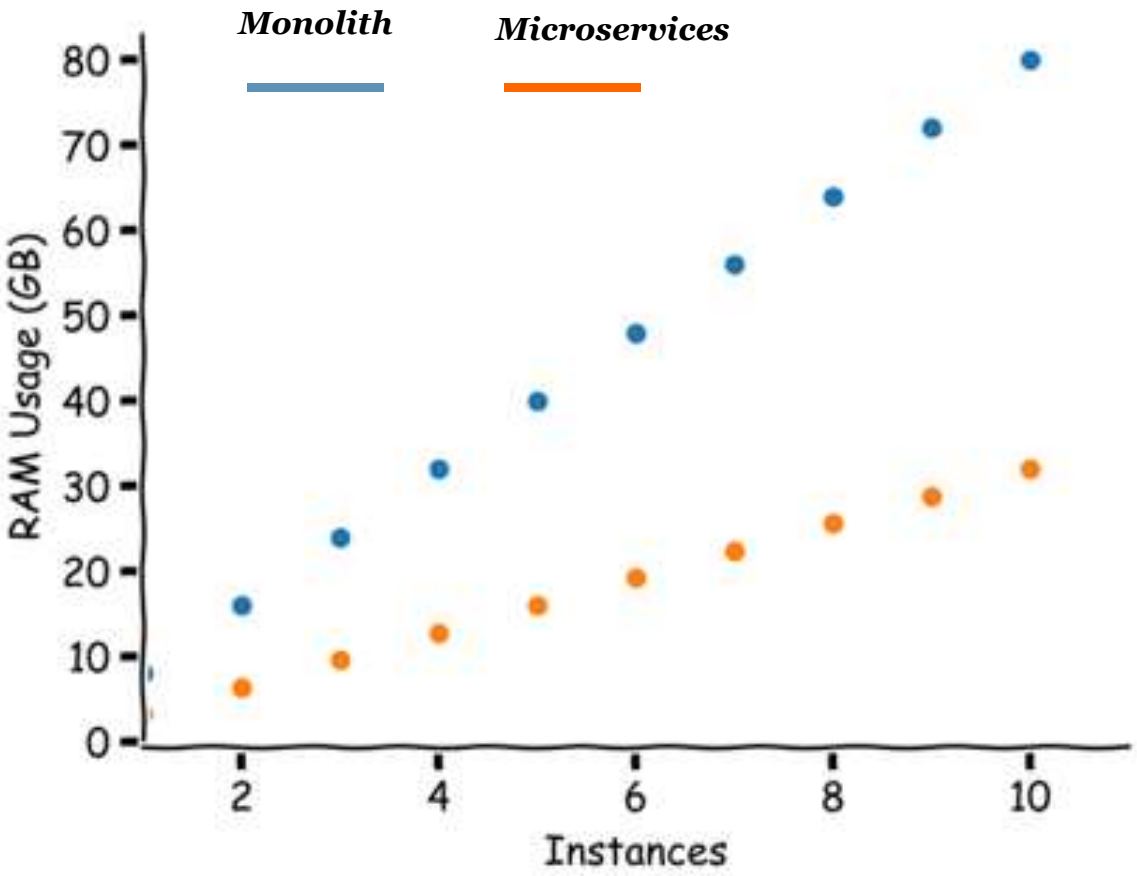


Monolithique vs microservice

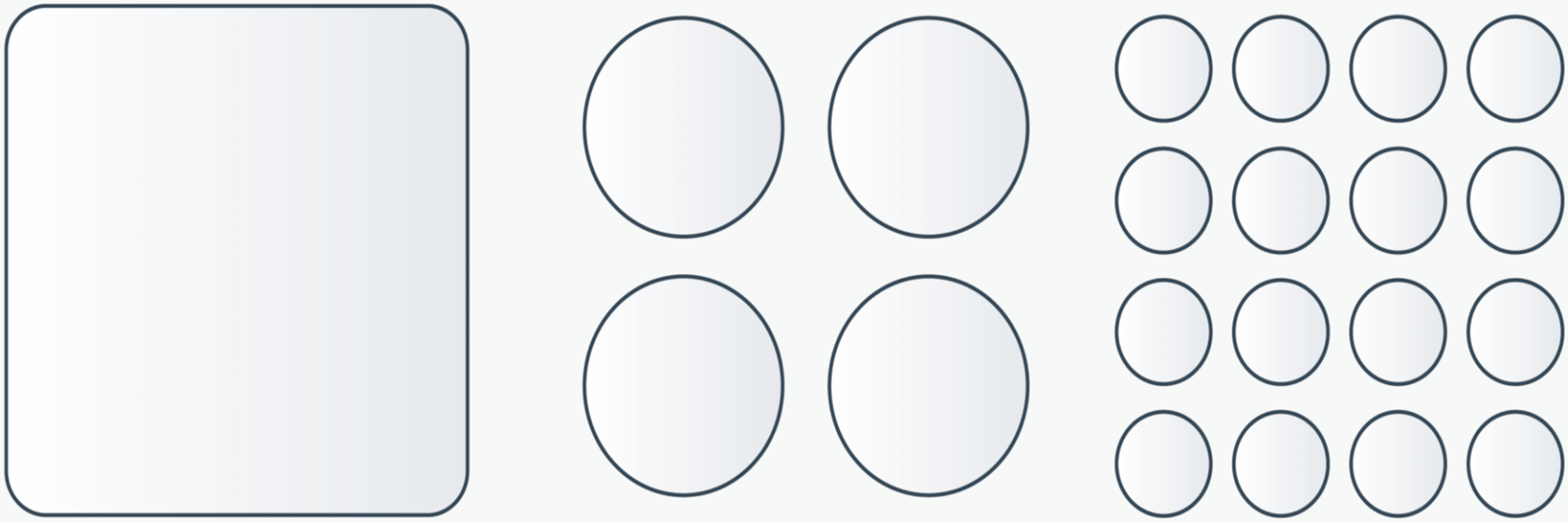
Monolithique vs microservice



Monolithique vs microservice



Monolithique vs SOA vs MSA



Monolithic

Single Unit

SOA

Coarse-grained

Microservices

Fine-grained

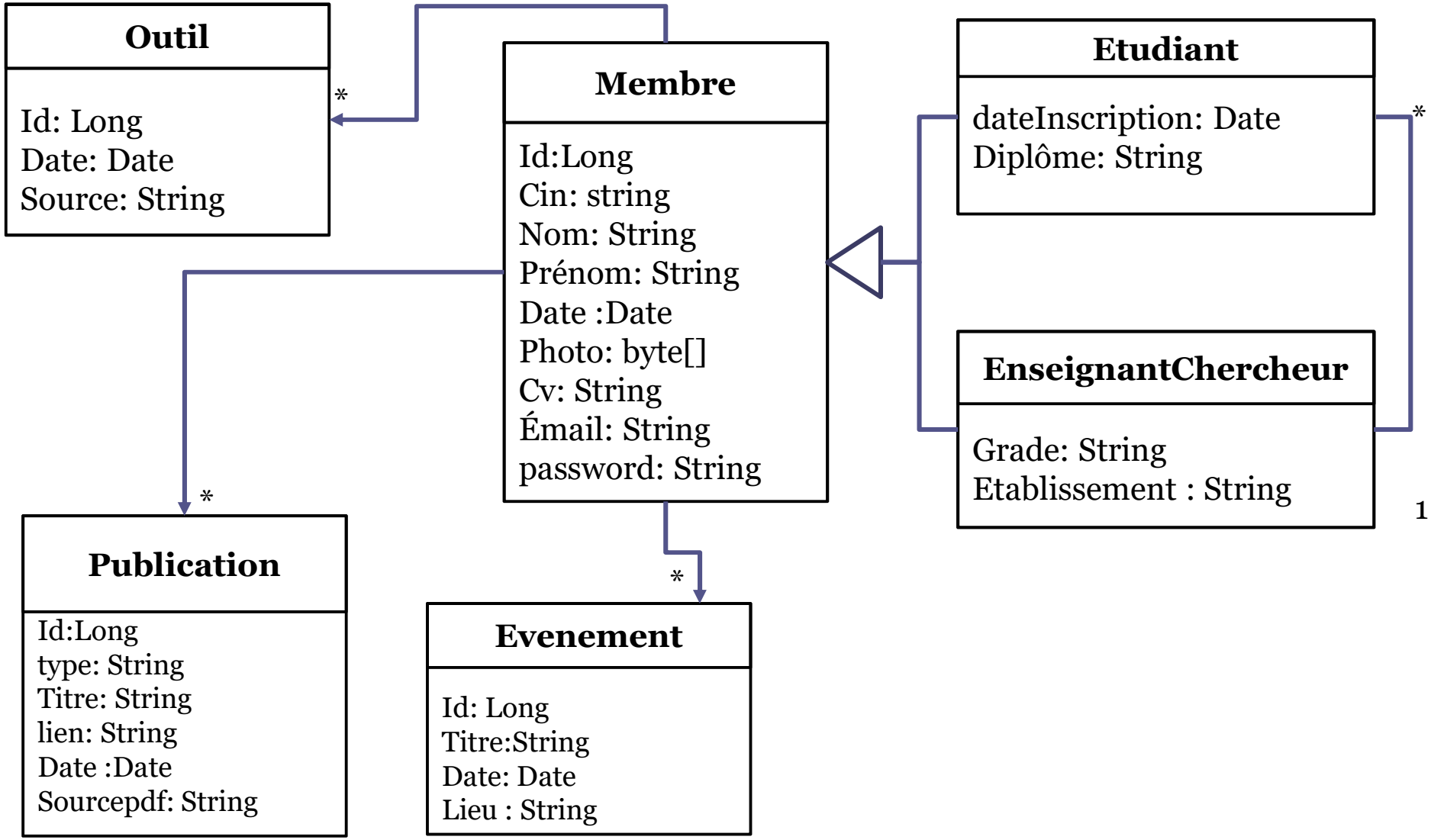
Défis d'une architecture microservices

- **Complexité :** la mise en place d'un tel système distribué se voit assez complexe**.**
- **Tests** : Avoir une multitude de services indépendants peut rendre l'écriture des tests plus complexe, particulièrement quand il y a de nombreuses dépendances entre les services
- **Intégrité des données** : Certaines transactions métiers, qui nécessitent de mettre à jour plusieurs fonctions métiers de l'application, ont besoin de mettre à jour plusieurs bases de données détenues par différents services.

=>mettre en place une cohérence éventuelle des données, ce qui est bien évidemment plus complexe et moins intuitif pour les développeurs.

- **Latence du réseau** : L'utilisation de nombreux services de petite taille peut se traduire par une intensification des communications entre les services.

Étude de cas


Outil
Id: Long
Date: Date
Source: String
Membre
Id:Long
Cin: string
Nom: String
Prénom: String
Date :Date
Photo: byte[]
Cv: String
Émail: String
password: String
Etudiant
dateInscription: Date
Diplôme: String
EnseignantChercheur
Grade: String
Etablissement : String
Publication
Id:Long
type: String
Titre: String
lien: String
Date :Date
Sourcepdf: String
Evenement
Id: Long
Titre:String
Date: Date
Lieu : String
*
*
*
*
1

Étude de cas: 4 microservices