Archictecture Microservice Principes et Partiques avec NodeJS

Regis Atemengue
@regis_ate
www.regisatemengue.com

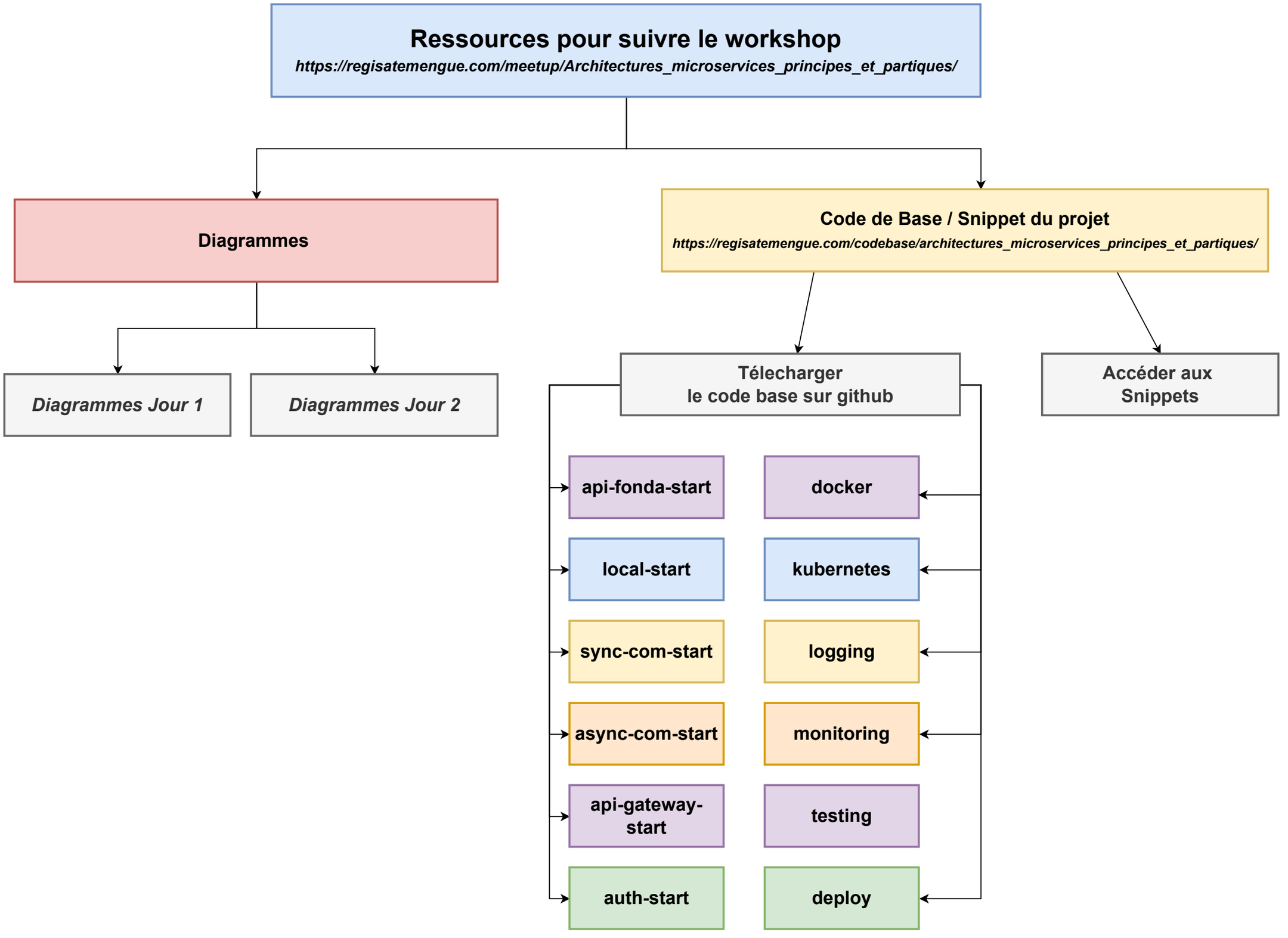
Contributeurs

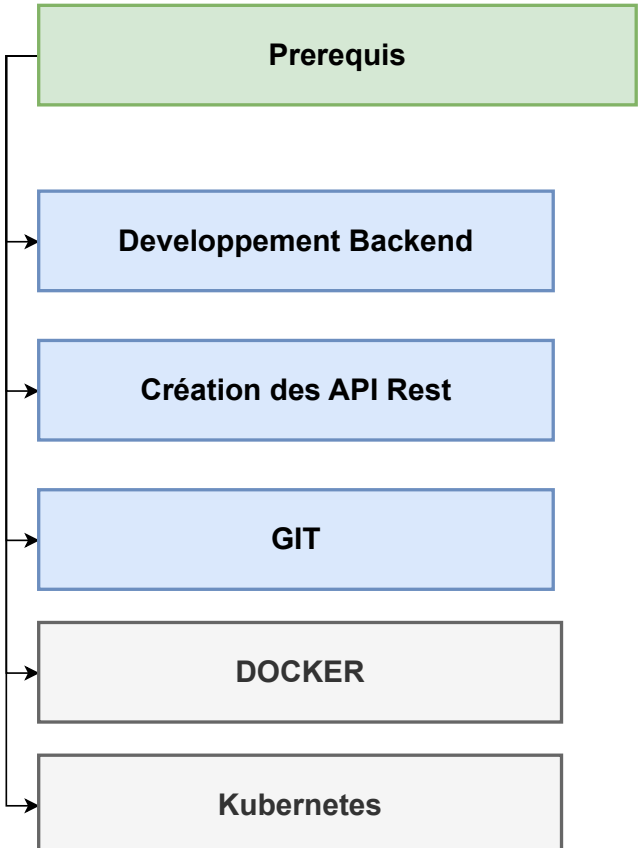
DAVE CHEDJOUN

ABDEL RAZAK

FOLONG EMERSON

CEDRIC MESSI SYLVESTRE


Ressources pour suivre le workshop
https://regisatemengue.com/meetup/Architectures_microservices_principes_et_partiques/
Diagrammes
Diagrammes Jour 1
Diagrammes Jour 2
Code de Base / Snippet du projet
https://regisatemengue.com/codebase/architectures_microservices_principes_et_partiques/
Télecharger
le code base sur github
Accéder aux
Snippets
api-fonda-start
local-start
sync-com-start
async-com-start
api-gateway-
start
auth-start
docker
kubernetes
logging
monitoring
testing
deploy


Prerequis
Developpement Backend
Création des API Rest
GIT
DOCKER
Kubernetes

Environnement de Travail

node v20.11.1

Laravel 10.10

Java 14 ou Superieur

Editeur de Code

Visual Studio Code

Autres

Base de données

mongo DB
(local/ image docker)

Postgresql
(local/ image docker)

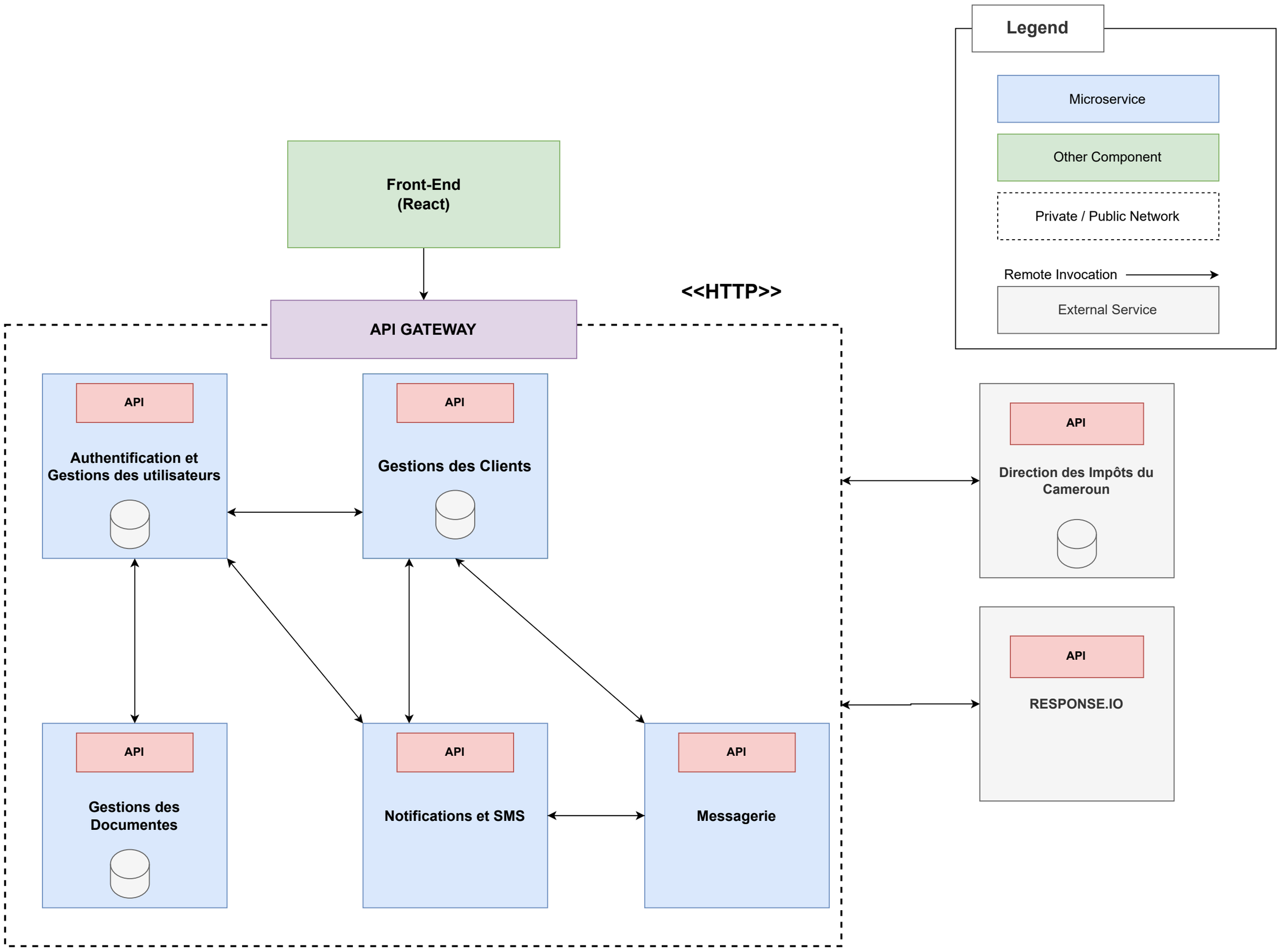
Interface Graphique pour le BD

PgAdmin

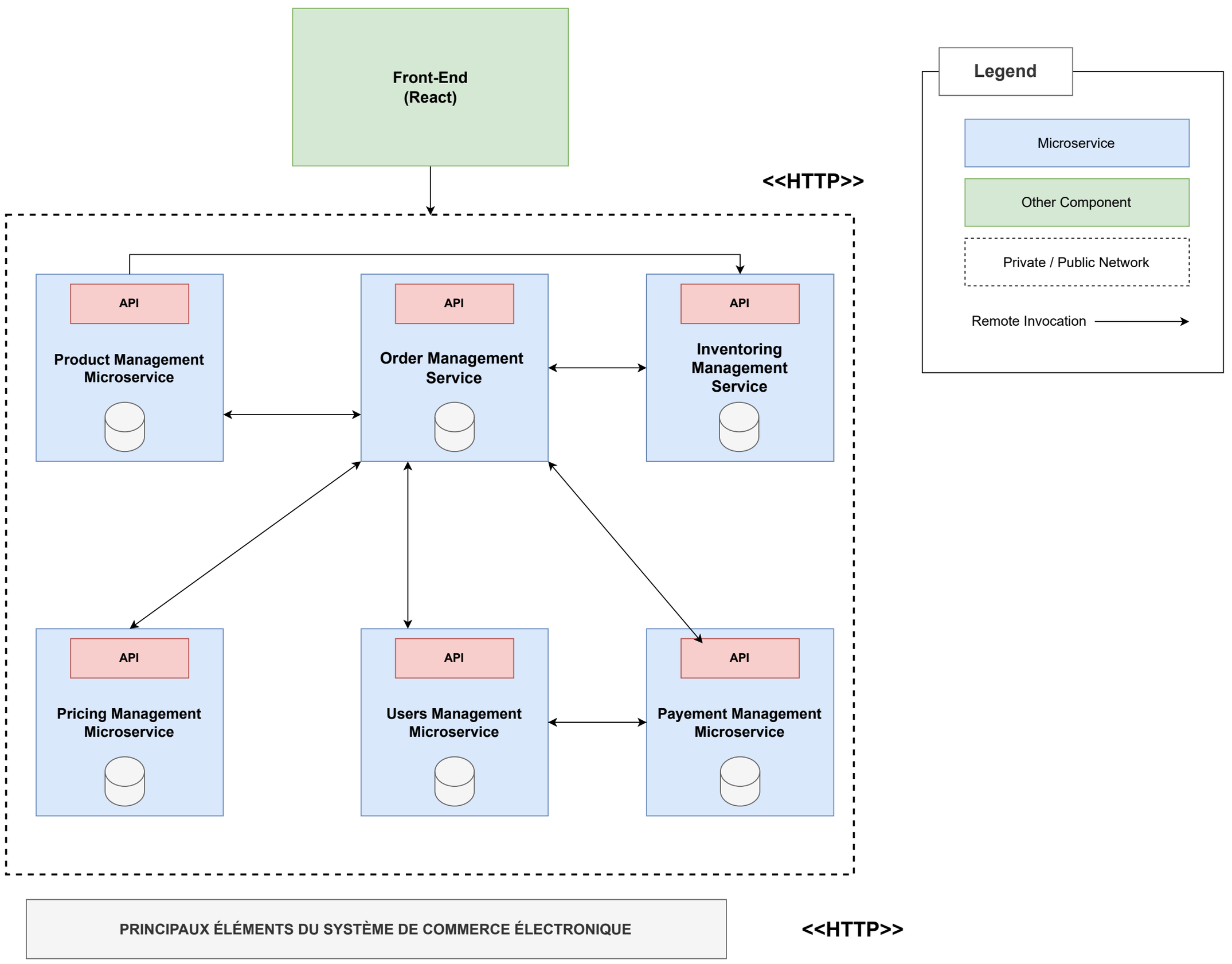
mongodb compass / Studio 3T

Extension VS Code

- Extension VS Code: Rest CLIENT



Architecture de départ



Profils

Développeurs débutants/intermédiaires

Développeurs expérimentés

Etudiants en Informatique L2 L3 M1 et M2

Architecture Finale
Communication Asynchorne

Front-End
(React)

<<HTTPS>>

API GATEWAY

API
Product Management
Microservice

API
Order Management
Service

API
Inventoring
Management
Service

<<AMQP>>

Message Broker

API
Pricing Management
Microservice

API
Users Management
Microservice

API
Payement Management
Microservice

Legend

Microservice

Other Component

Private / Public Network

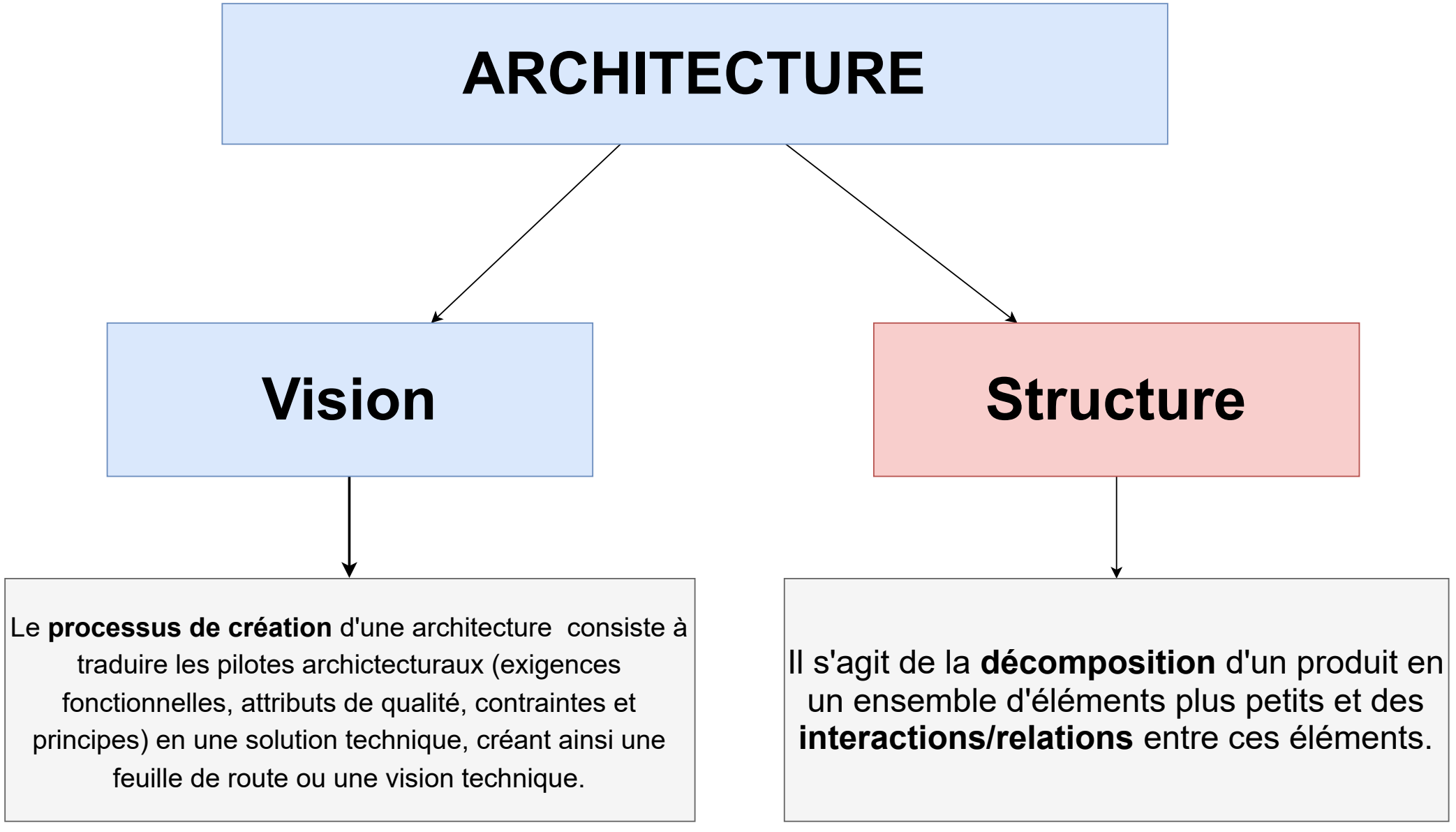
Remote Invocation

Tech Stack

EventBus Message Broker

Kafka
RabbitMQ
Google Pub/Sub
Amazon Services
ActiveMQ
Azure Service Bus

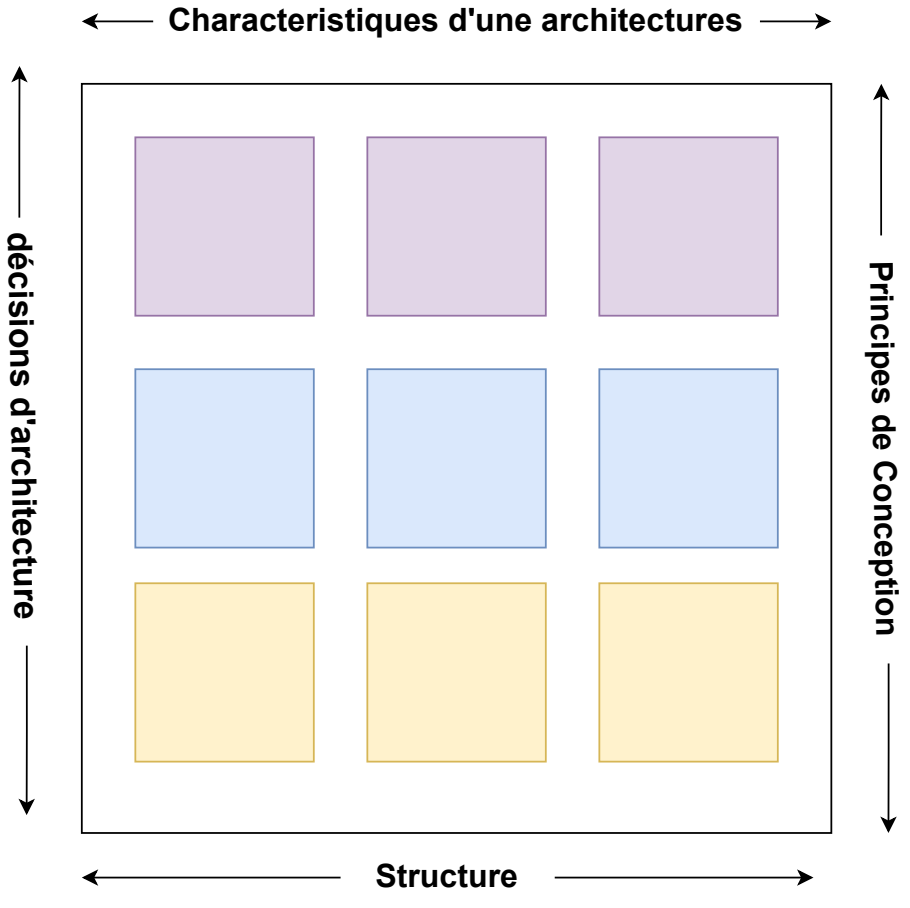
PRINCIPAUX ÉLÉMENTS DU SYSTÈME DE COMMERCE ÉLECTRONIQUE


ARCHITECTURE
Vision
Structure
Le **processus de création** d'une architecture consiste à traduire les pilotes archictecturaux (exigences fonctionnelles, attributs de qualité, contraintes et principes) en une solution technique, créant ainsi une feuille de route ou une vision technique.
Il s'agit de la **décomposition** d'un produit en un ensemble d'éléments plus petits et des **interactions/relations** entre ces éléments.

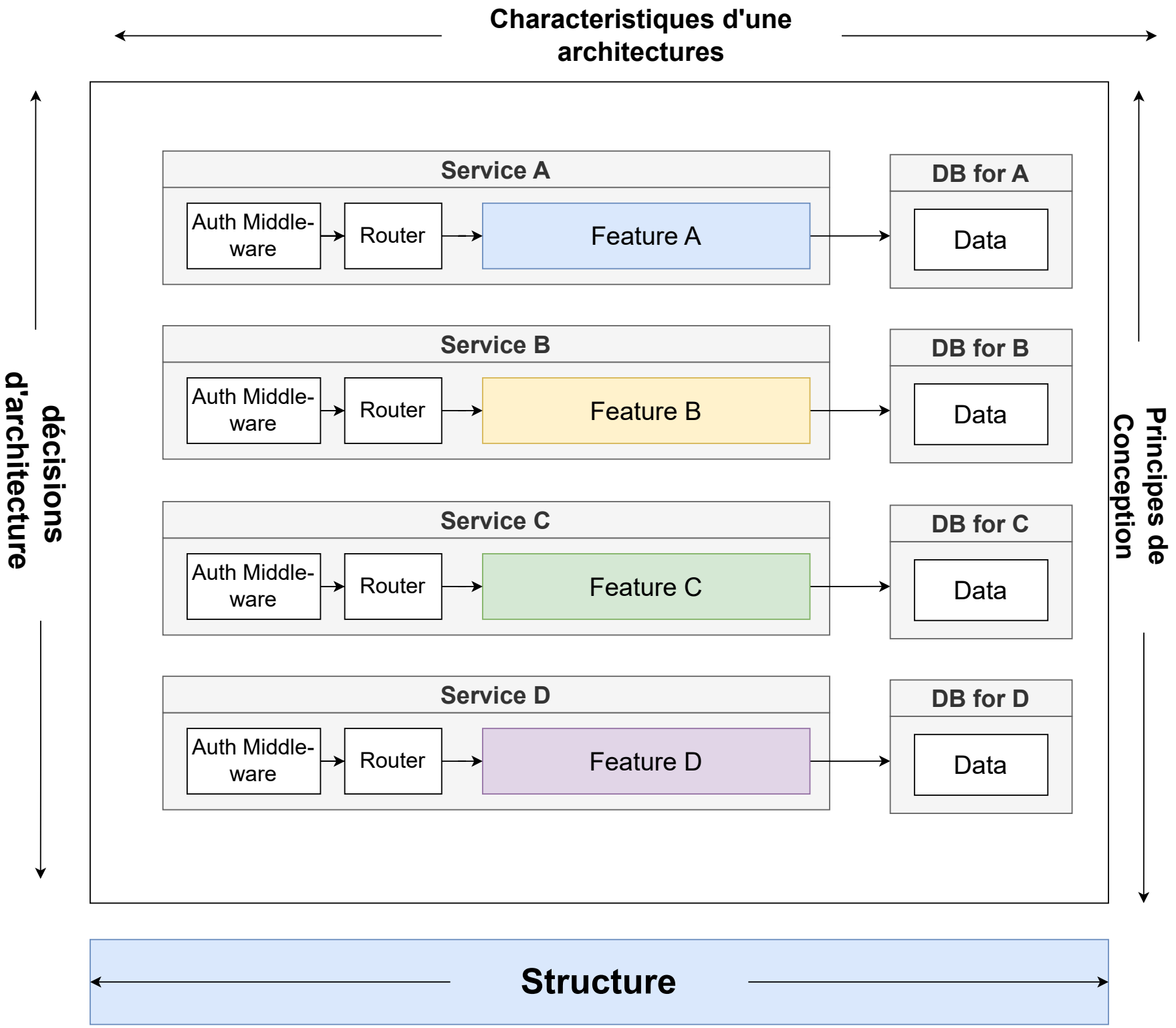
Architecture Logiciel

L'architecture logicielle se compose de la **structure du système** combinée aux **caractéristiques architecturales** ("qualités/capacites") que le système doit prendre en charge, aux **décisions d'architecture** et, enfin, **aux principes de conception.**

Architecture Logiciel



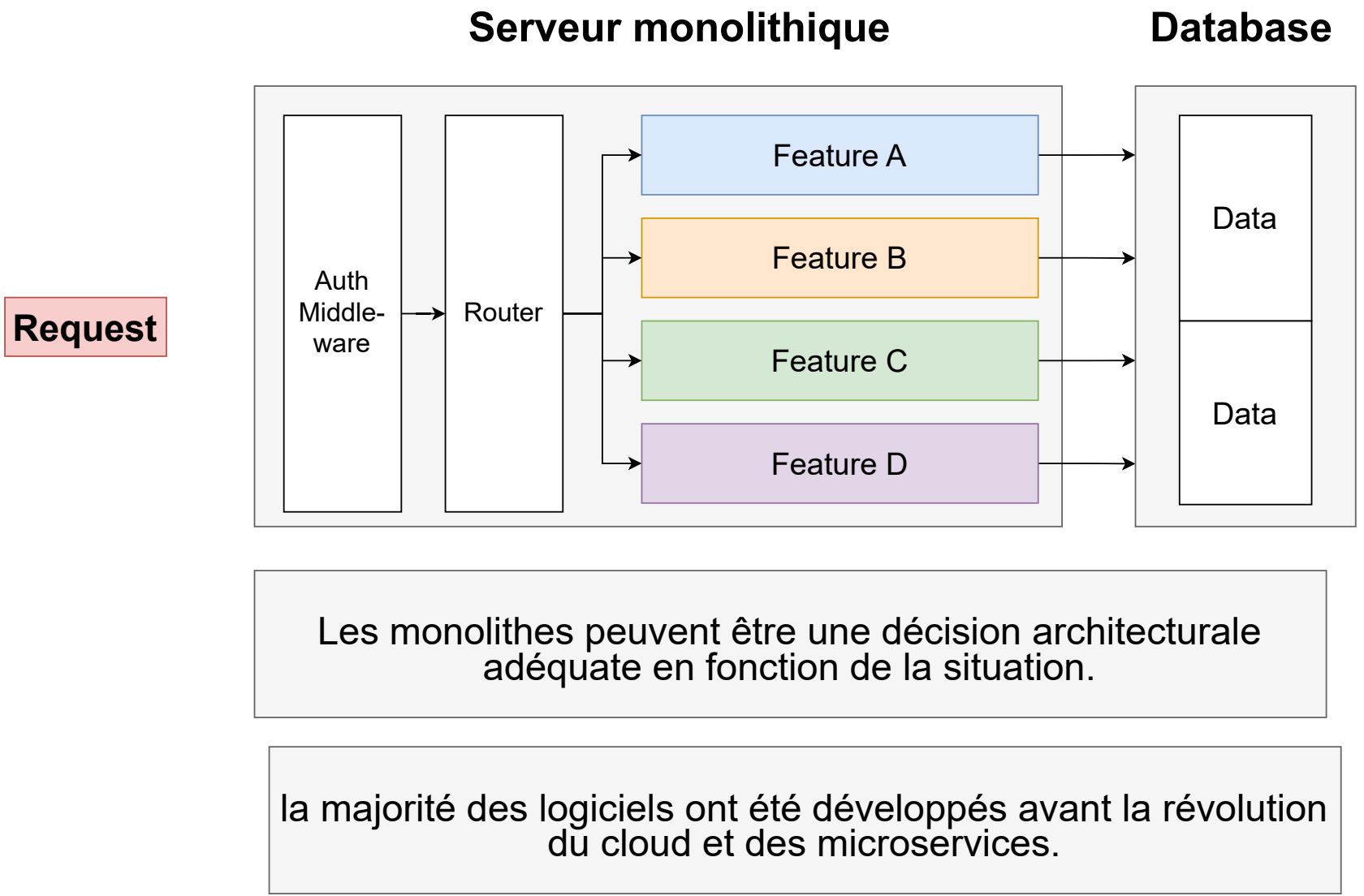
Architecture Logiciel



Styles Architecturaux/ Structure

- Monolithique
- Layered Architecture
- SOA (Service Oriented Architecture)
- Microservices
- Event-Driven Architecture

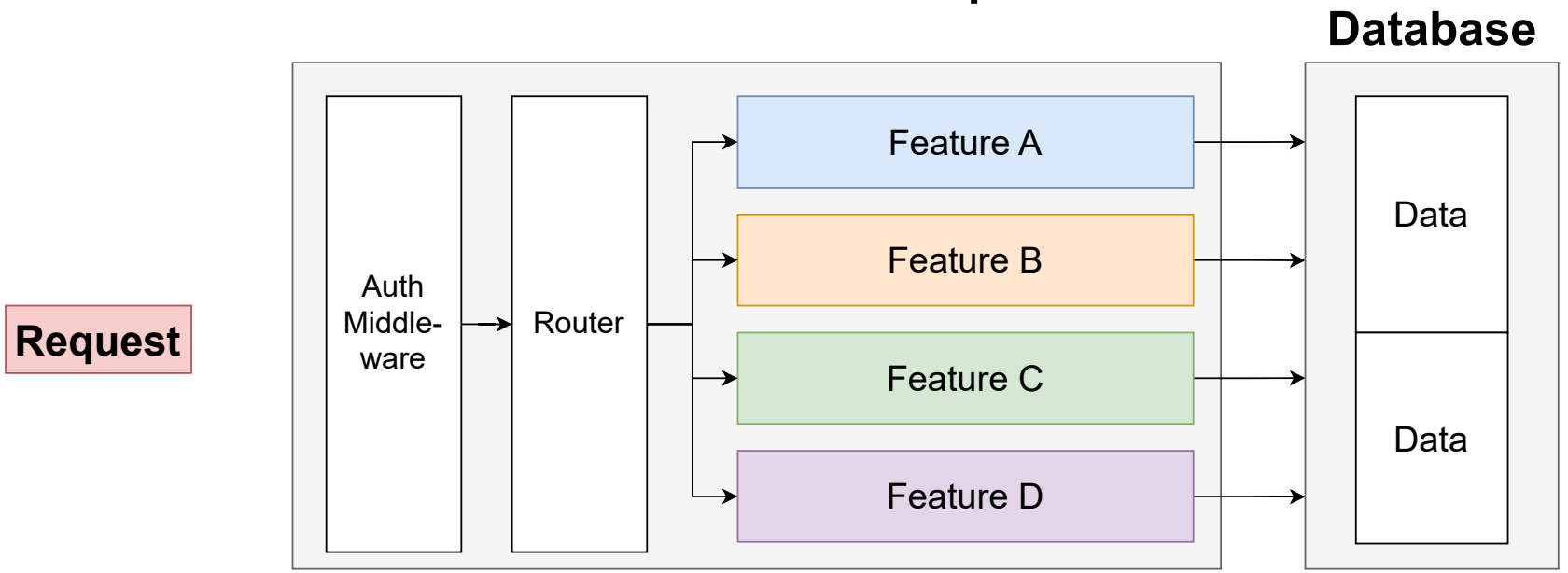
Monolithe/ Monolithique



Les monolithes peuvent être une décision architecturale adéquate en fonction de la situation.

la majorité des logiciels ont été développés avant la révolution du cloud et des microservices.

Serveur monolithique



Un **monolithe** est une application complète qui s'exécute dans un seul processus.

Scalabilité

Team Devs

Désordre

Utilisateurs

Maintenances

Utilisation d'un seul langage de Programmation

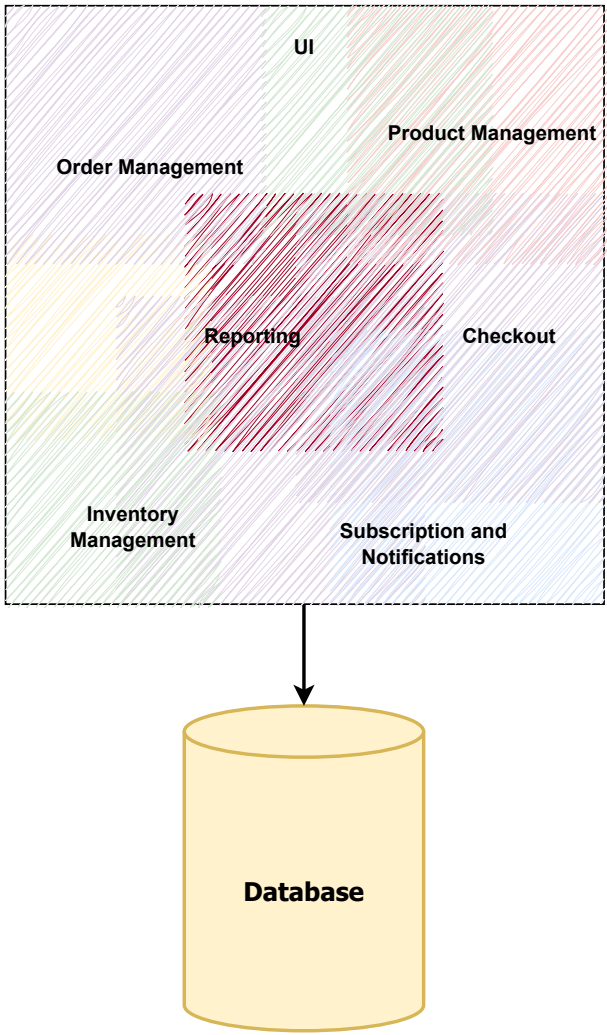
Difficulté de deploiement

Risque d'arrêt

Scalabilité Verticale coûteuse

Réduction de l'agilité sur les technologies meme l'equipe

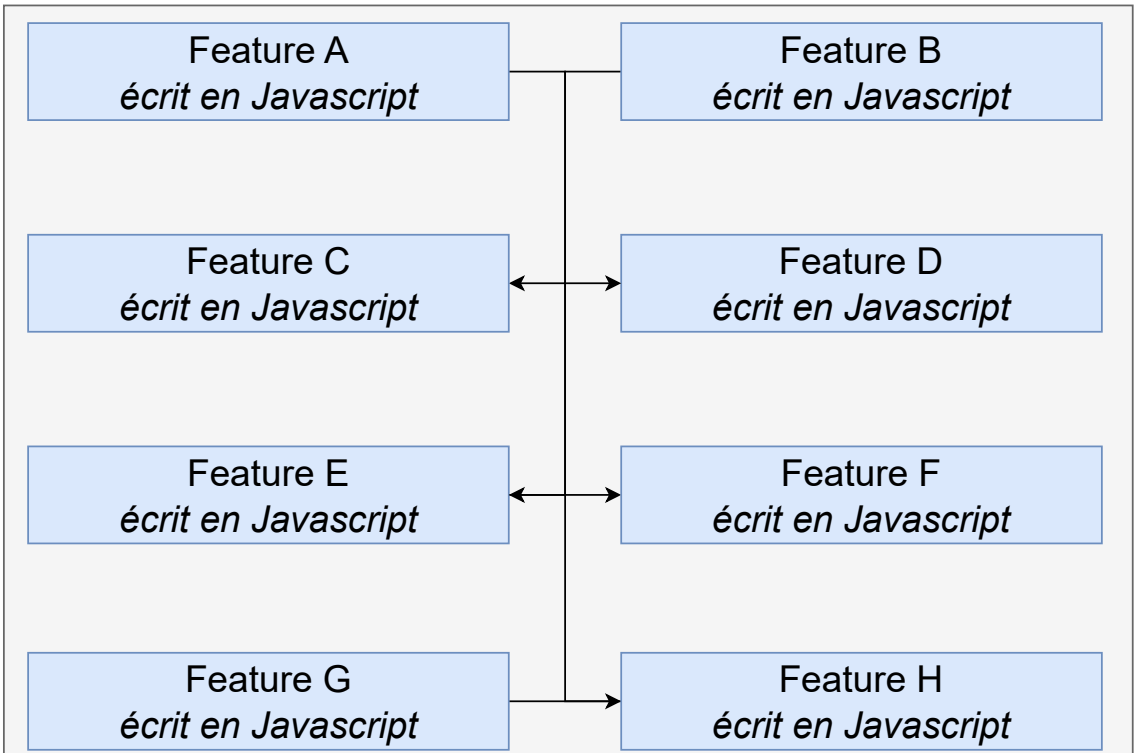
Plusieurs autres PB....

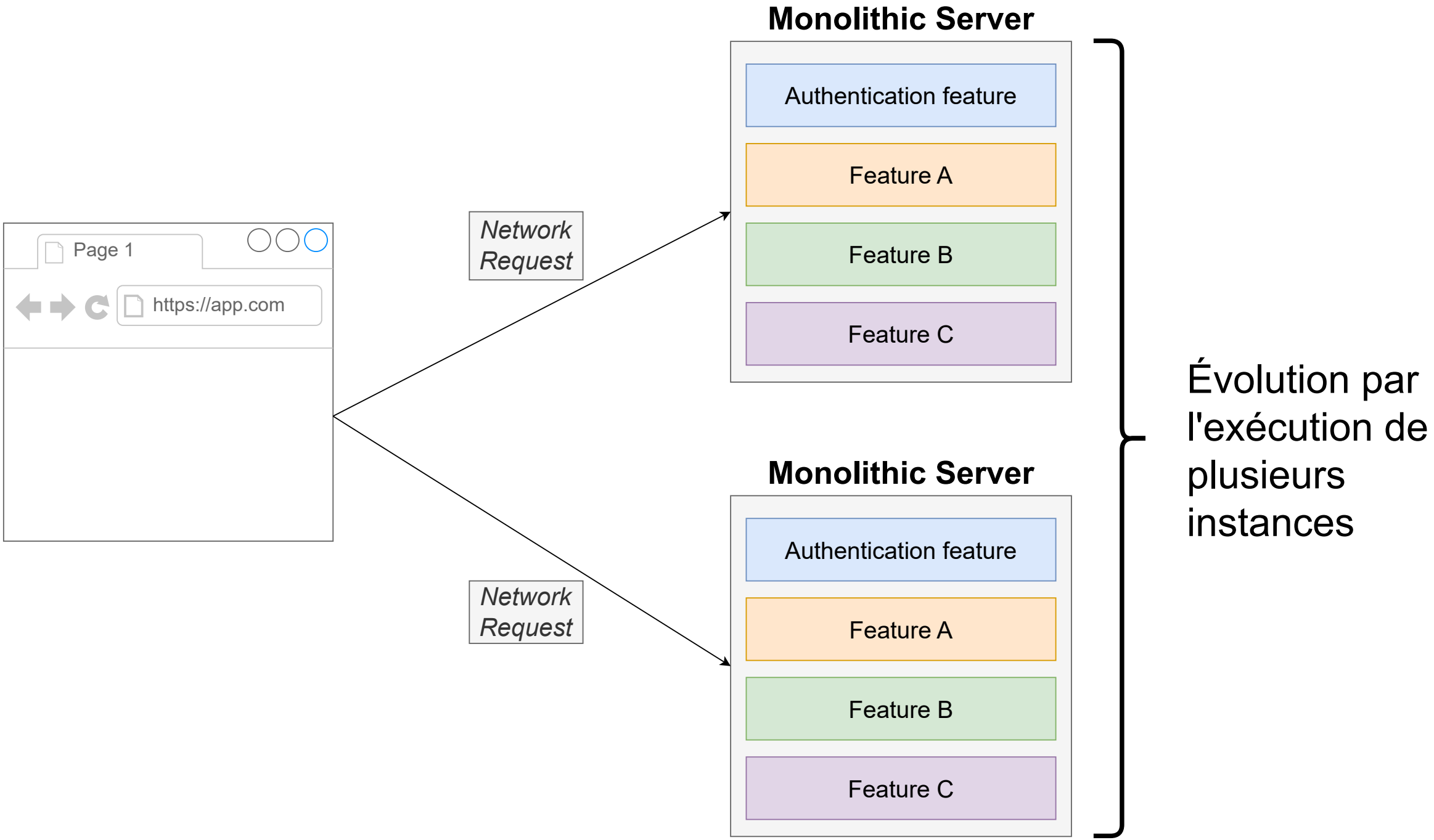


LE COMMERCE ÉLECTRONIQUE EST UN MONOLITHE PATCHWORK DONT LES FRONTIÈRES ENTRE LES DIFFÉRENTS DOMAINES NE SONT PAS CLAIRES

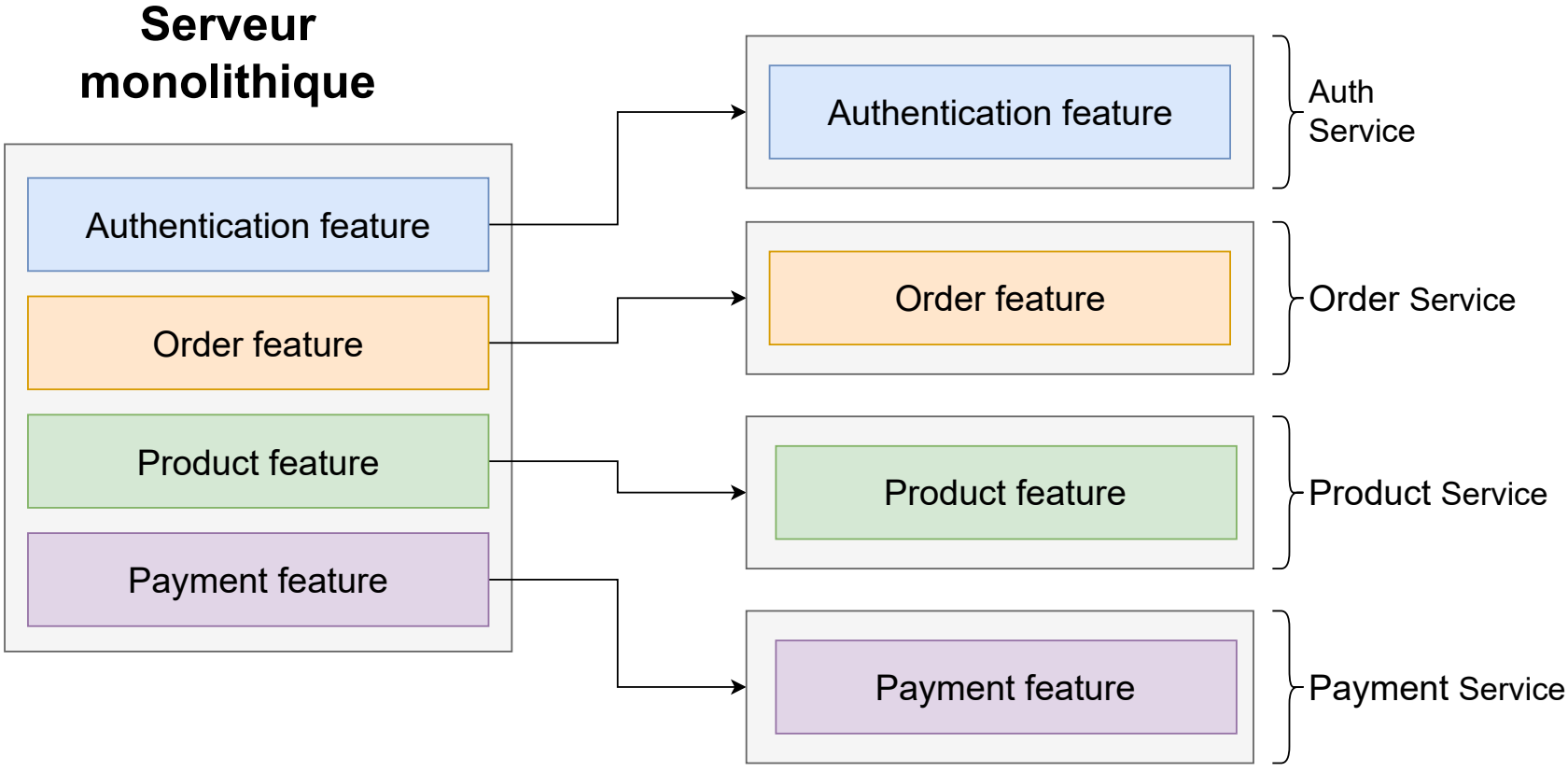
Plusieurs autres PB....

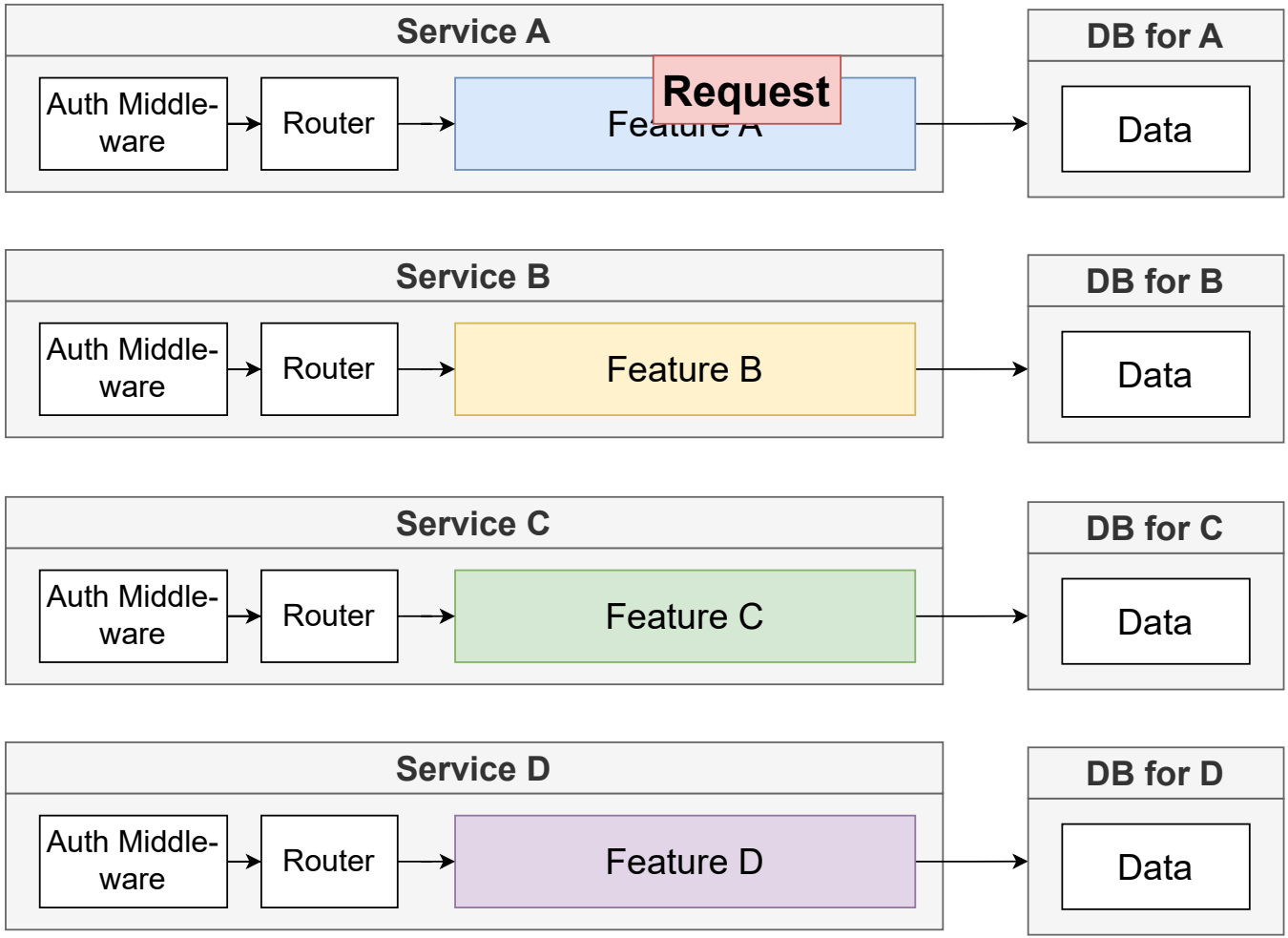
Serveur monolithique

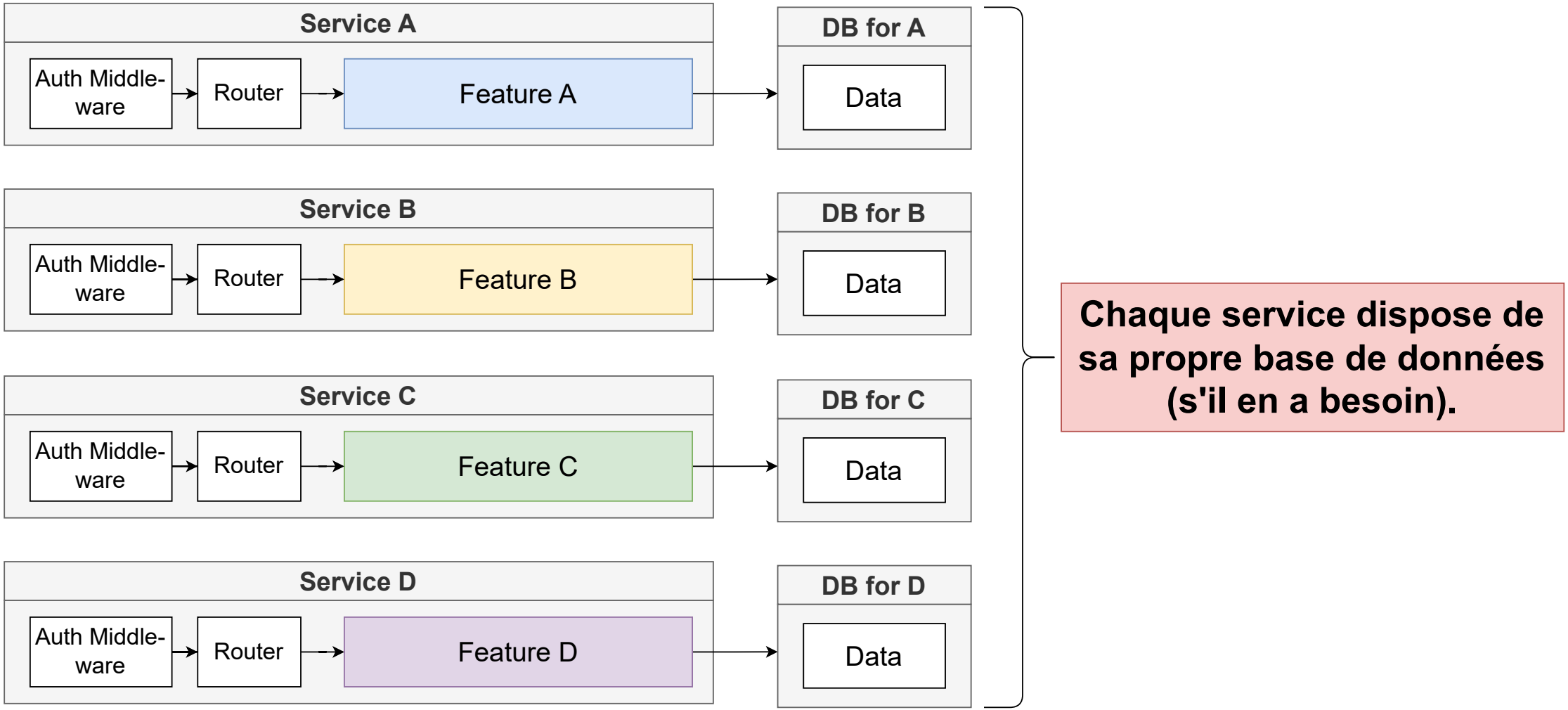


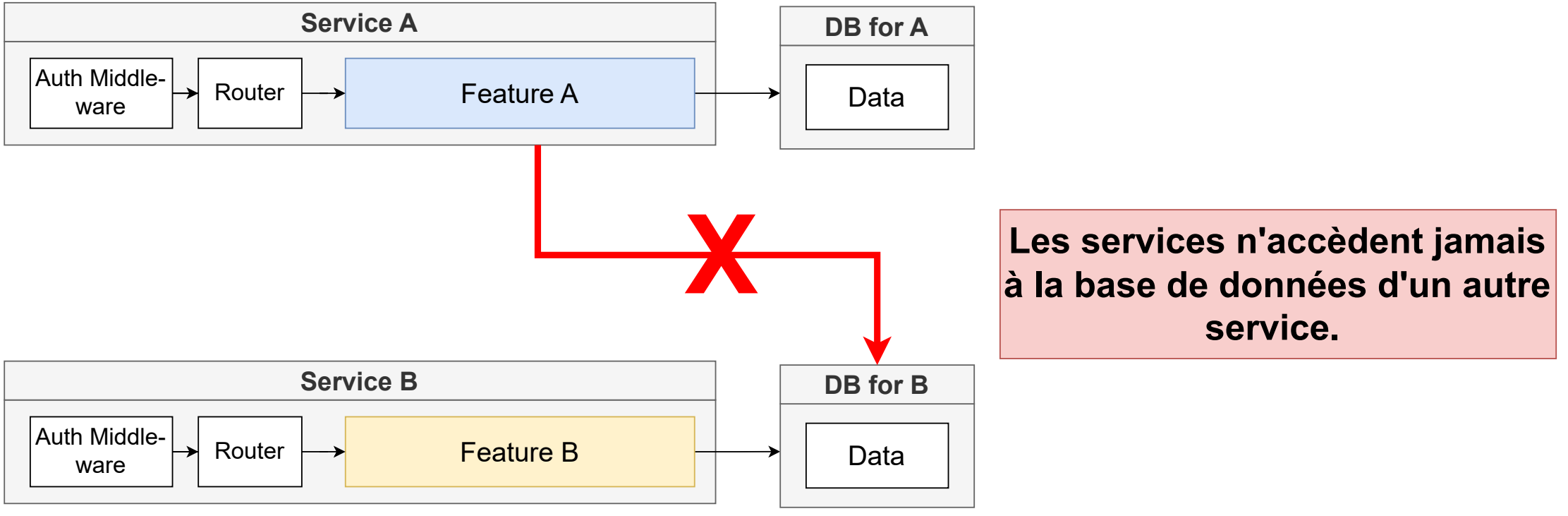

Monolithic Server
Authentication feature
Feature A
Feature B
Feature C
Page 1
https://app.com
Network Request
Network Request
Monolithic Server
Authentication feature
Feature A
Feature B
Feature C
Évolution par l'exécution de plusieurs instances

Microservices


Serveur monolithique
Authentication feature
Order feature
Product feature
Payment feature
Authentication feature
Auth Service
Order feature
Order Service
Product feature
Product Service
Payment feature
Payment Service


Service A
Auth Middle-ware
Router
Request
Feature A
DB for A
Data
Service B
Auth Middle-ware
Router
Feature B
DB for B
Data
Service C
Auth Middle-ware
Router
Feature C
DB for C
Data
Service D
Auth Middle-ware
Router
Feature D
DB for D
Data


Service A
Auth Middle-ware
Router
Feature A
DB for A
Data
Service B
Auth Middle-ware
Router
Feature B
DB for B
Data
Service C
Auth Middle-ware
Router
Feature C
DB for C
Data
Service D
Auth Middle-ware
Router
Feature D
DB for D
Data
Chaque service dispose de sa propre base de données (s'il en a besoin).


Service A
Auth Middle-ware
Router
Feature A
DB for A
Data
Service B
Auth Middle-ware
Router
Feature B
DB for B
Data
Les services n'accèdent jamais à la base de données d'un autre service.

Chaque service dispose de sa propre base de données (s'il en a besoin).

Les services n'accèdent jamais à la base de données d'un autre service.

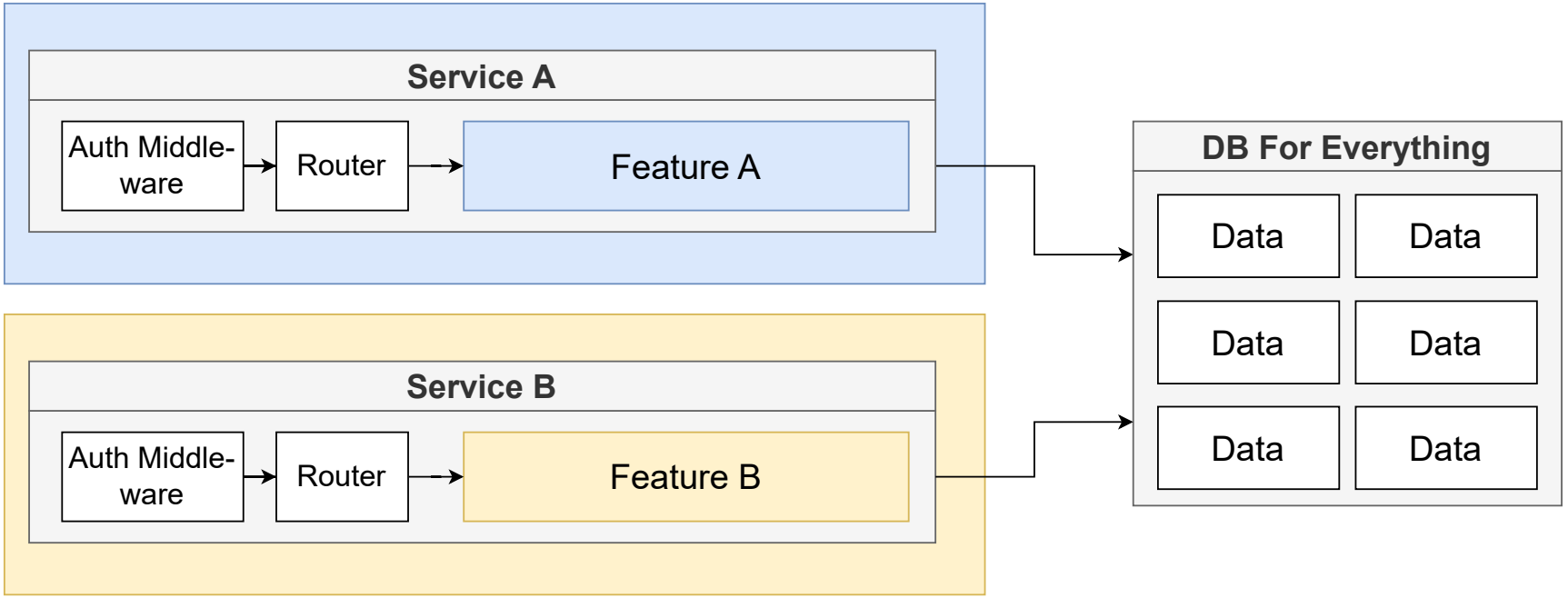
D'accord, mais pourquoi ?

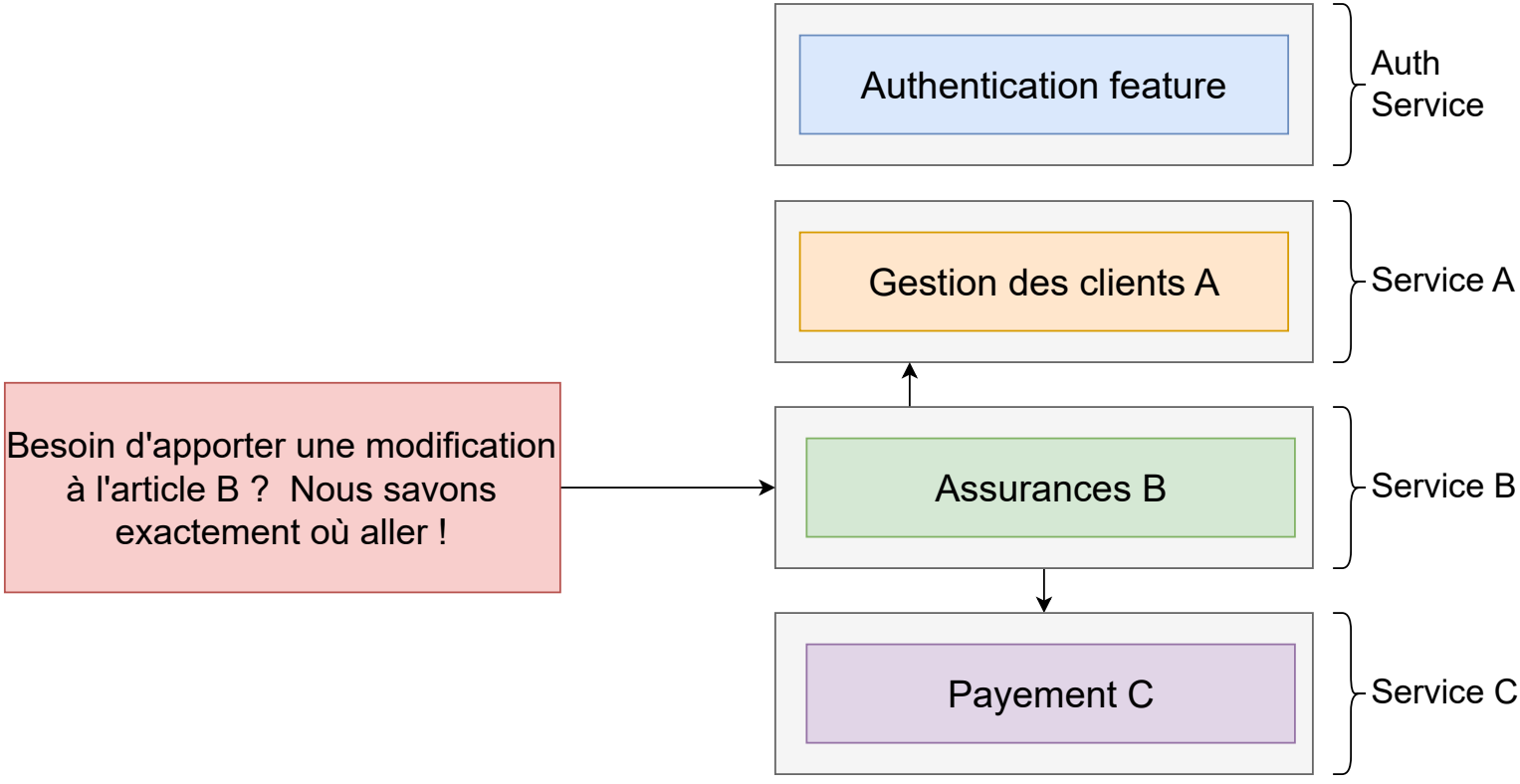
Pourquoi une base de données par service ?

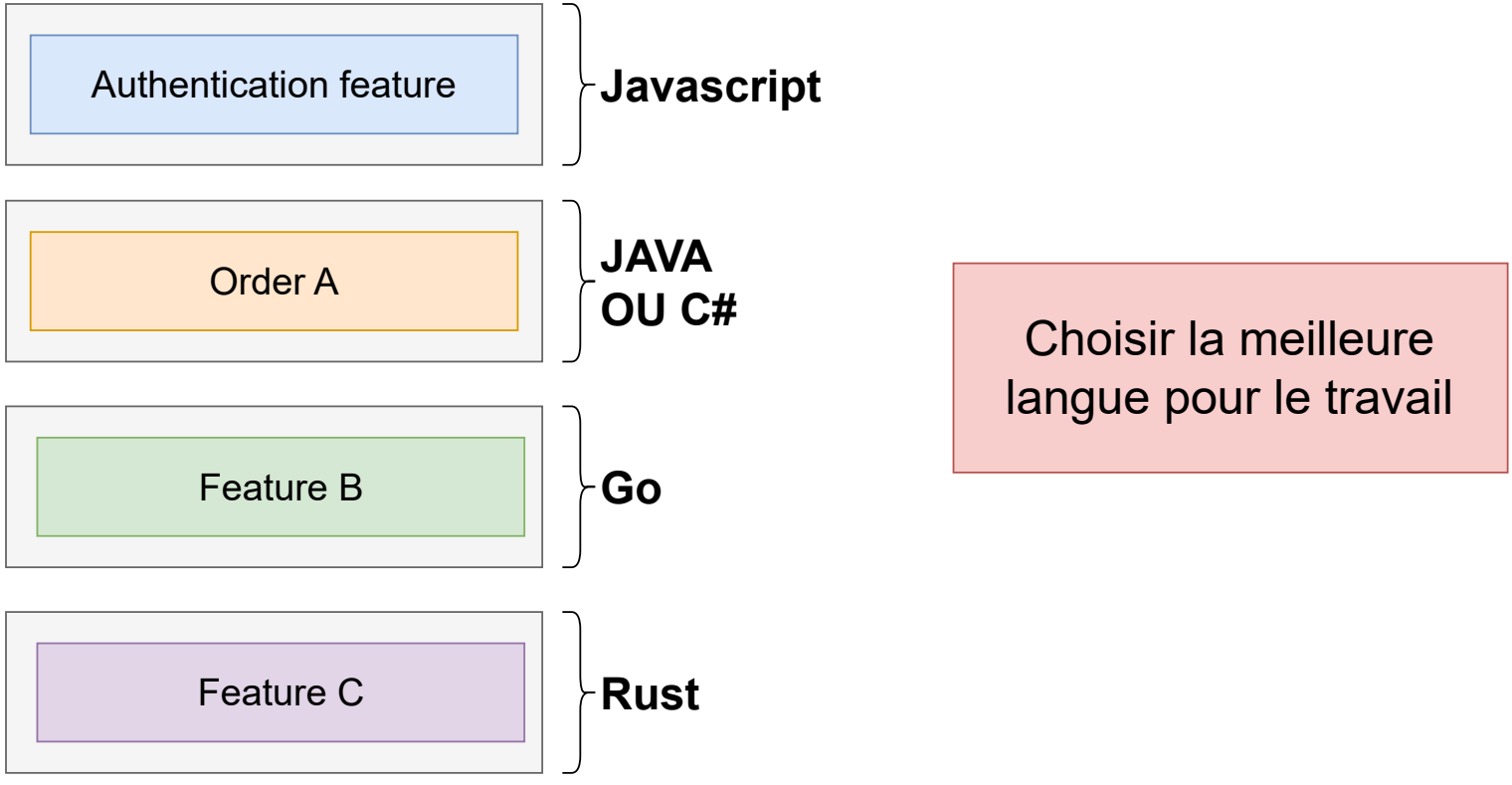
Nous voulons que chaque service fonctionne indépendamment des autres services.

Le schéma ou la structure de la base de données peut changer de manière inattendue.

Certains services peuvent fonctionner plus efficacement avec différents types de bases de données (sql ou nosql).


Service A
Auth Middle-ware
Router
Feature A
Service B
Auth Middle-ware
Router
Feature B
DB For Everything
Data
Data
Data
Data
Data
Data


Authentication feature
Auth Service
Gestion des clients A
Service A
Besoin d'apporter une modification à l'article B ? Nous savons exactement où aller !
Assurances B
Service B
Payement C
Service C


Authentication feature
Javascript
Order A
JAVA
OU C#
Feature B
Go
Feature C
Rust
Choisir la meilleure
langue pour le travail

Un microservice unique contient

Routing	Middlewares	Business Logic	Database access

pour mettre en œuvre une fonctionnalité de notre application

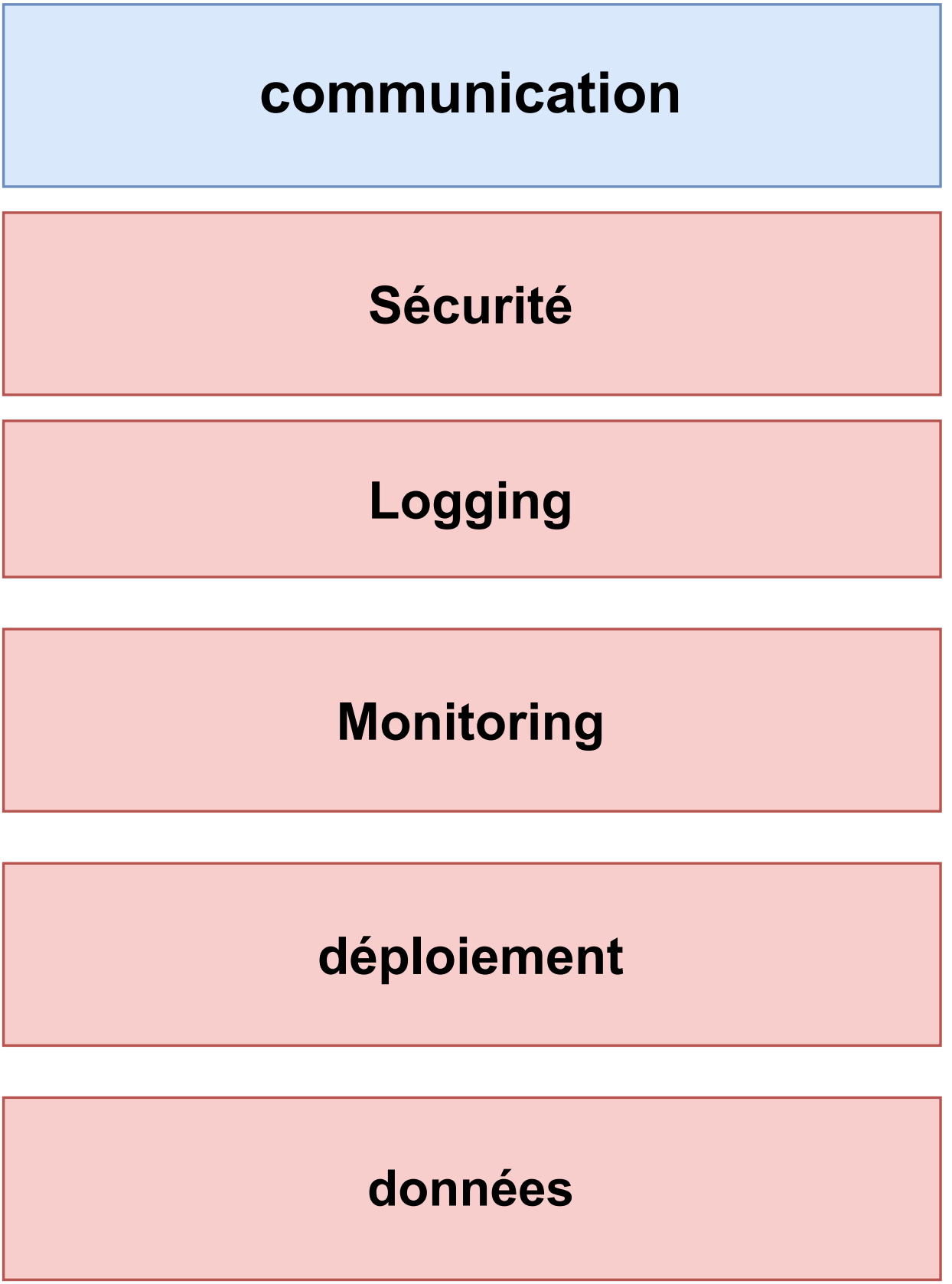
L'architecture microservice est un style architectural qui structure une application comme une collection de services qui sont :

- déployables de manière indépendante
- faiblement couplés
- Organisés autour des capacités de l'entreprise
- Appartenant à une petite équipe

L'architecture microservice permet à une organisation de fournir des **applications** complexes et de grande envergure de manière rapide, fréquente, fiable et durable - une nécessité pour être compétitif et gagner dans le monde d'aujourd'hui.

source : microservices.io

Alors.... quelles sont les grands défis des microservices ?


communication
Sécurité
Logging
Monitoring
déploiement
données