MICROSERVICES

JÉRÉMY PERROUAULT

GATEWAY

UN POINT D'ACCÈS UNIQUE

GATEWAY

- Bien souvent, ces services seront appelés par un client « front »
  - Ce client doit connaitre chaque localisation des différents services
  - Et même en utilisant un « Service Discovery », le client devra gérer plusieurs point d'accès différents

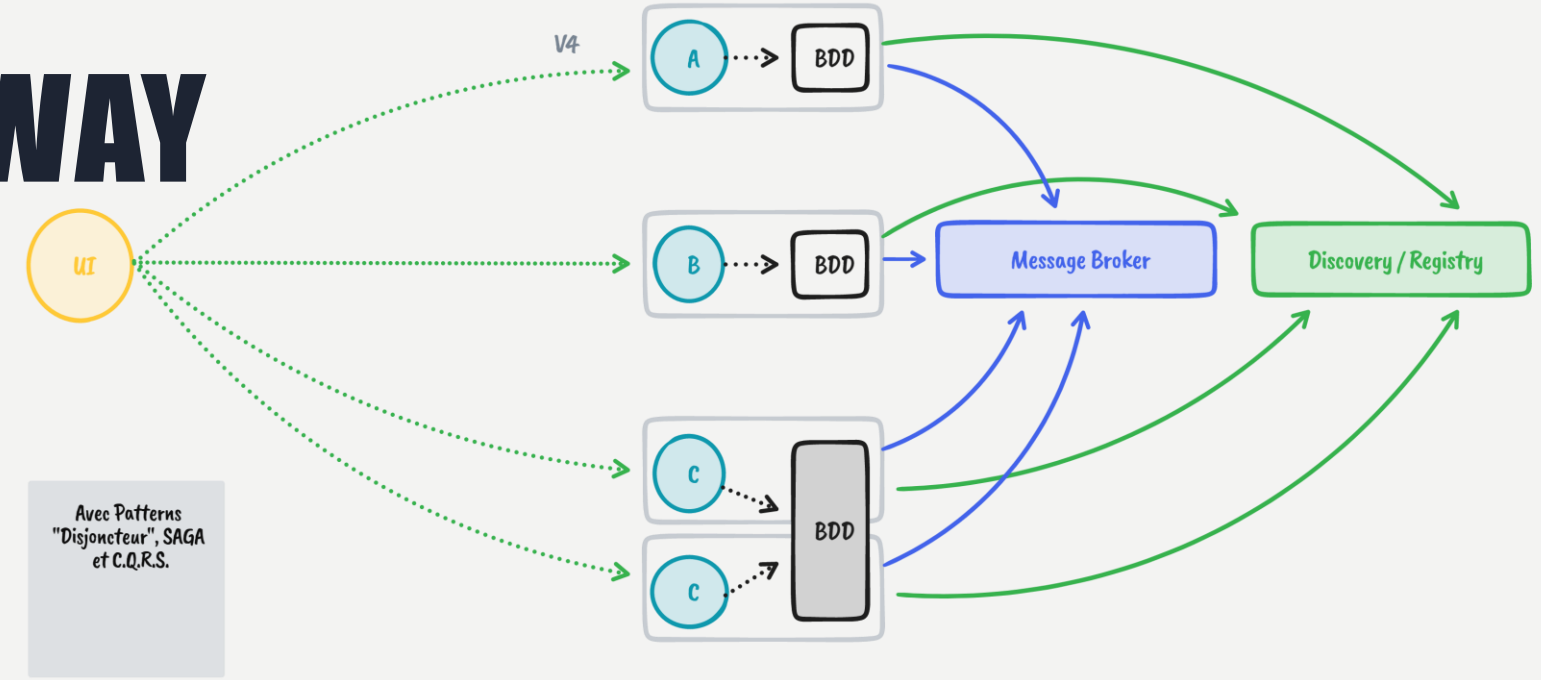
GATEWAY

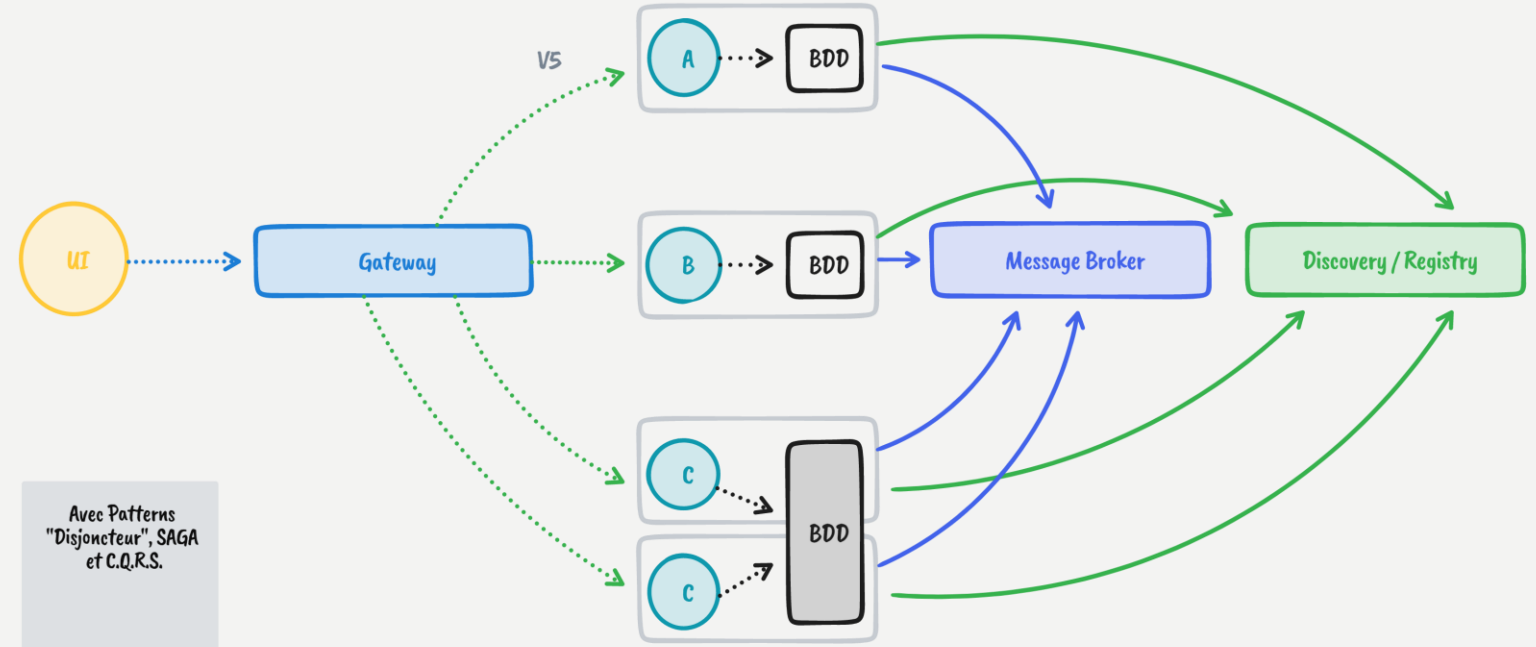
- Utilisation du Pattern « Gateway »
  - L'idée, c'est d'avoir un serveur « front » qui se charge de redistribuer les requêtes aux bons services

GATEWAY

- Utilisation d'une technologie existante …
  - Spring Cloud Gateway
  - …

GATEWAY


V4
A
BDD
UI
B
BDD
Message Broker
Discovery / Registry
C
BDD
C
Avec Patterns "Disjoncteur", SAGA et C.Q.R.S.


V5
A
BDD
UI
Gateway
B
BDD
Message Broker
Discovery / Registry
C
BDD
C
Avec Patterns "Disjoncteur", SAGA et C.Q.R.S.

GATEWAY

- Utiliser Spring Cloud Gateway
- Créer un serveur Gateway

SPRING BOOT

EXEMPLE D'IMPLÉMENTATIONS

SPRING BOOT

- Ajouter la dépendance *spring-cloud-starter-gateway*

```
<dependency>
  <groupId>org.springframework.cloud</groupId>
  <artifactId>spring-cloud-starter-gateway</artifactId>
</dependency>
```

- Le serveur **Gateway** est un client **Eureka**

SPRING BOOT

- Configuration des routes

```
@Bean
public RouteLocator customRouteLocator(RouteLocatorBuilder builder) {
  return builder.routes()
    .route(r ->
      r.path("/api/produit/**")
      .uri("lb://nom-service-a/api/produit")
    )
    .route(r ->
      r.path("/api/service-b/**")
      .uri("lb://nom-service-b/api")
    )
    .build();
}
```

```
http://magateway/api/produit              -> lb://nom-service-a/api/produit
http://magateway/api/service-b/by-id/1    -> lb://nom-service-b/api/by-id/1
```