13/11/2023
Version 2

MICROSERVICES

JÉRÉMY PERROUAULT

C.Q.R.S.

GESTION DES REQUÊTES COMPOSÉES

CQRS

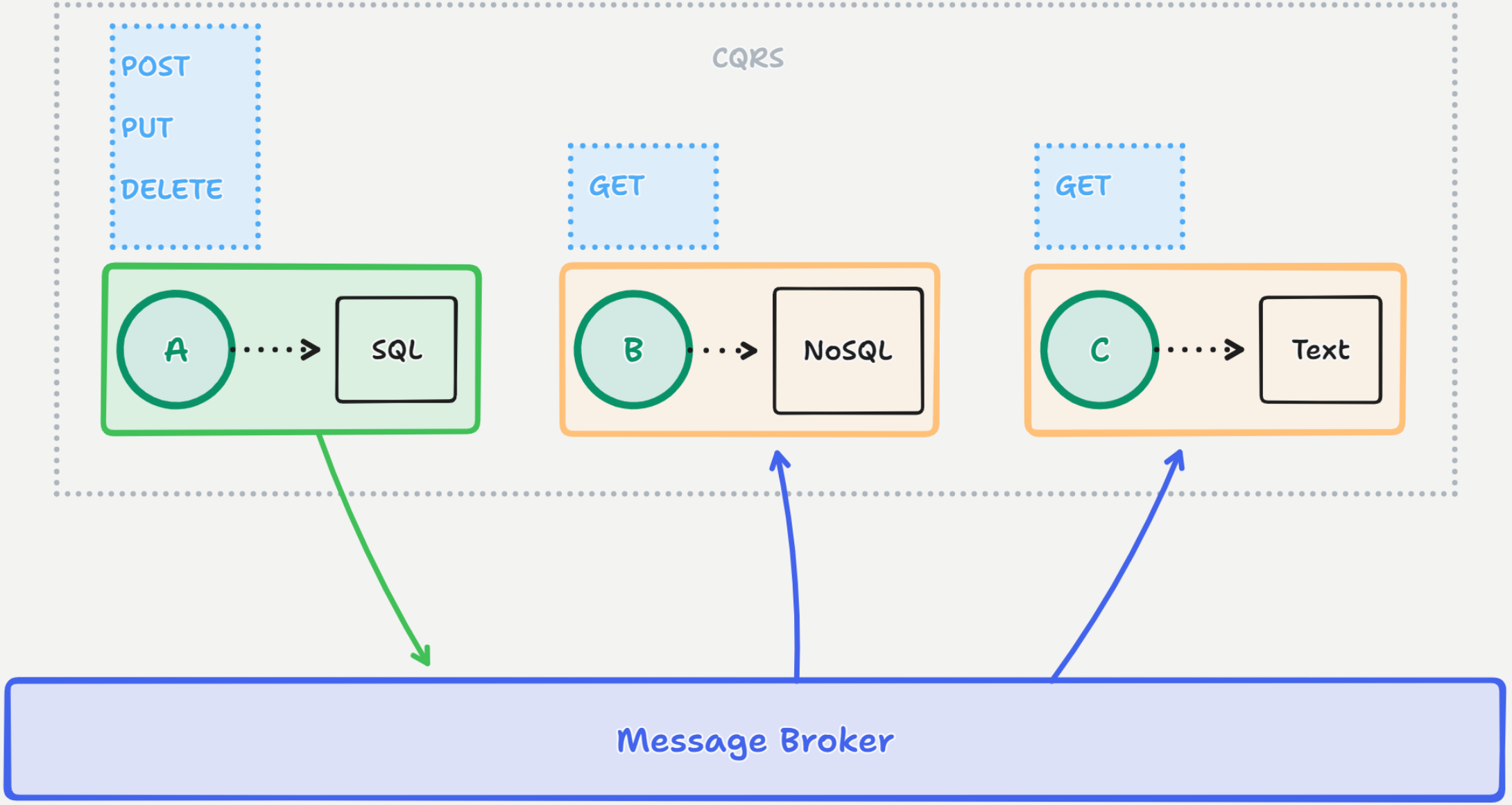
- On peut remplacer notre **Composition API**
  - « Service A » a besoin de « Service B » pour répondre …
  - Il y a donc un appel qui est fait, avec un **Disjoncteur**

CQRS

- Utilisation du Pattern **CQRS** (**C**ommand and **Q**uery **R**esponsibility **S**egregation)
  - L'idée, c'est d'avoir un découpage des services encore plus fins
    - La partie modification des données est séparée de la partie interrogation des données
      - Un service qui se charge des changements d'états (**Command**) `INSERT, UPDATE, DELETE`
      - Un service qui se charge de l'interrogation des données (**Query**) `SELECT`
  - DONC, un service différent … DONC, une base de données différente
    - Et l'implémentation peut varier, avec par exemple
      - **PostgreSQL** pour l'enregistrement des données
      - **MongoDB** pour la lecture
- Et bien sûr, ça sous-entend que les données sont dupliquées !
  - Il devient important d'avoir un stockage des évènements, permettant de les rejouer au cas où

CQRS

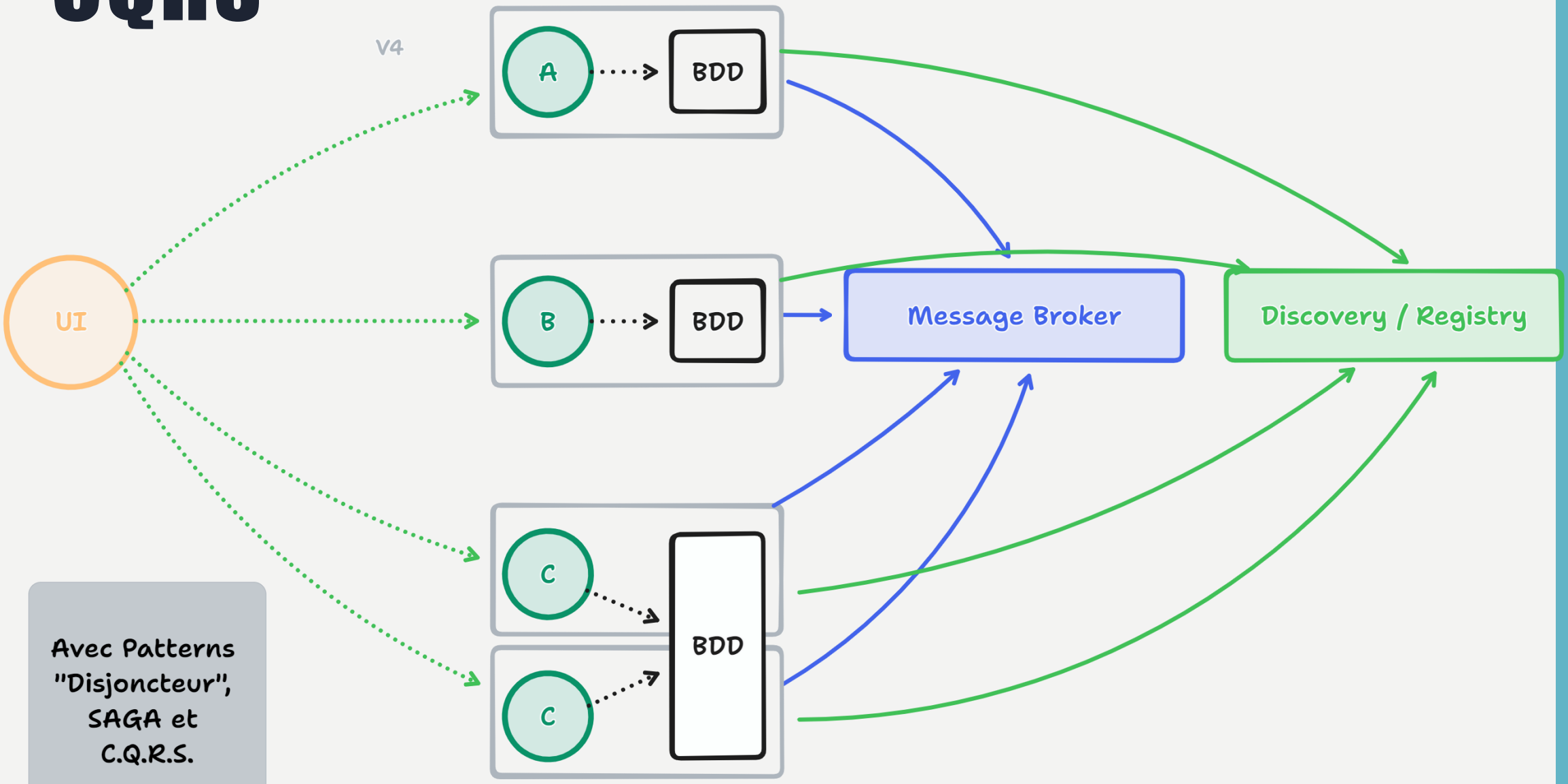
L'utilisation d'un Message Broker n'est absolument pas obligatoire.
C'est néanmoins une solution technique souvent implémentée.
Par ailleurs, le Message Broker est ici généralement un Event Store



CQRS

- Utilisation d'une technologie **Broker** existante …
  - **RabbitMQ**
  - **Kafka**
  - **Amazon SQS**
  - **Axon**
  - **EventStoreDb**
  - **Redis Stream**
  - …

CQRS


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

CQRS

- Utiliser **Kafka** ou **RabbitMQ** déjà mis en place
- Créer un 3eme service qui
  - Se chargera de stocker les informations produits & commentaires dans une nouvelle base de données
  - Permettra de lire les informations produit avec sa note, et commentaire et le nom du produit
- Implémenter les évènements dans les différents services