MICROSERVICES

JÉRÉMY PERROUAULT

C.Q.R.S.

GESTION DES REQUÊTES COMPOSÉES

C.Q.R.S.

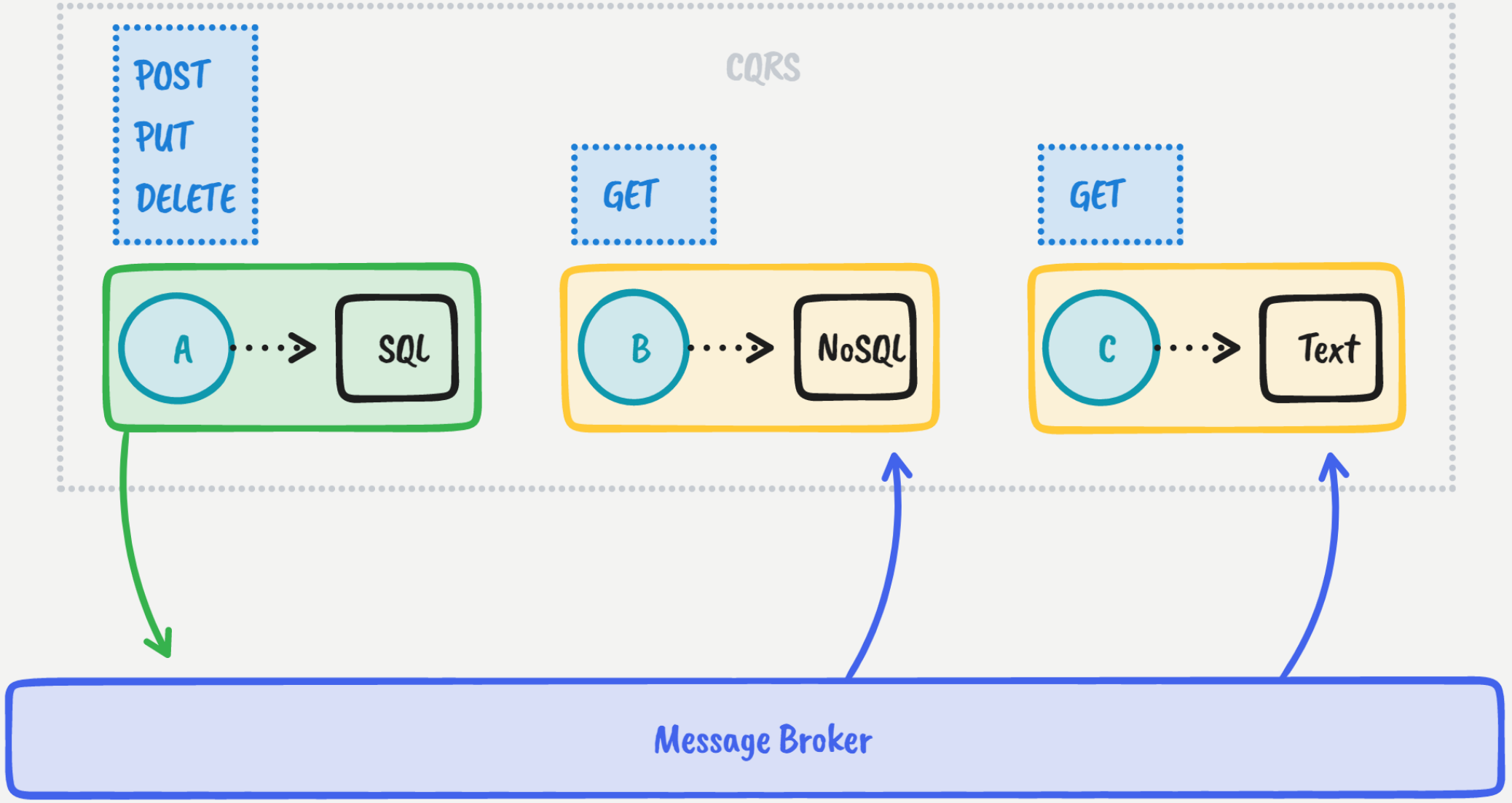
- On peut remplacer notre « Composition API »
  - « Service A » a besoin de « service B » pour répondre …
  - Il y a donc un appel qui est fait, avec un disjoncteur

C.Q.R.S.

- Utilisation du Pattern « C.Q.R.S. » (Command and Query Responsibility Segregation)
  - 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, service différent … DONC, base de données différente
    - Et l'implémentation peut varier
      - PostgreSQL pour l'enregistrement des données
      - MongoDB pour la lecture
- Et bien sûr, ça sous-entend que les données sont dupliquées !

C.Q.R.S.

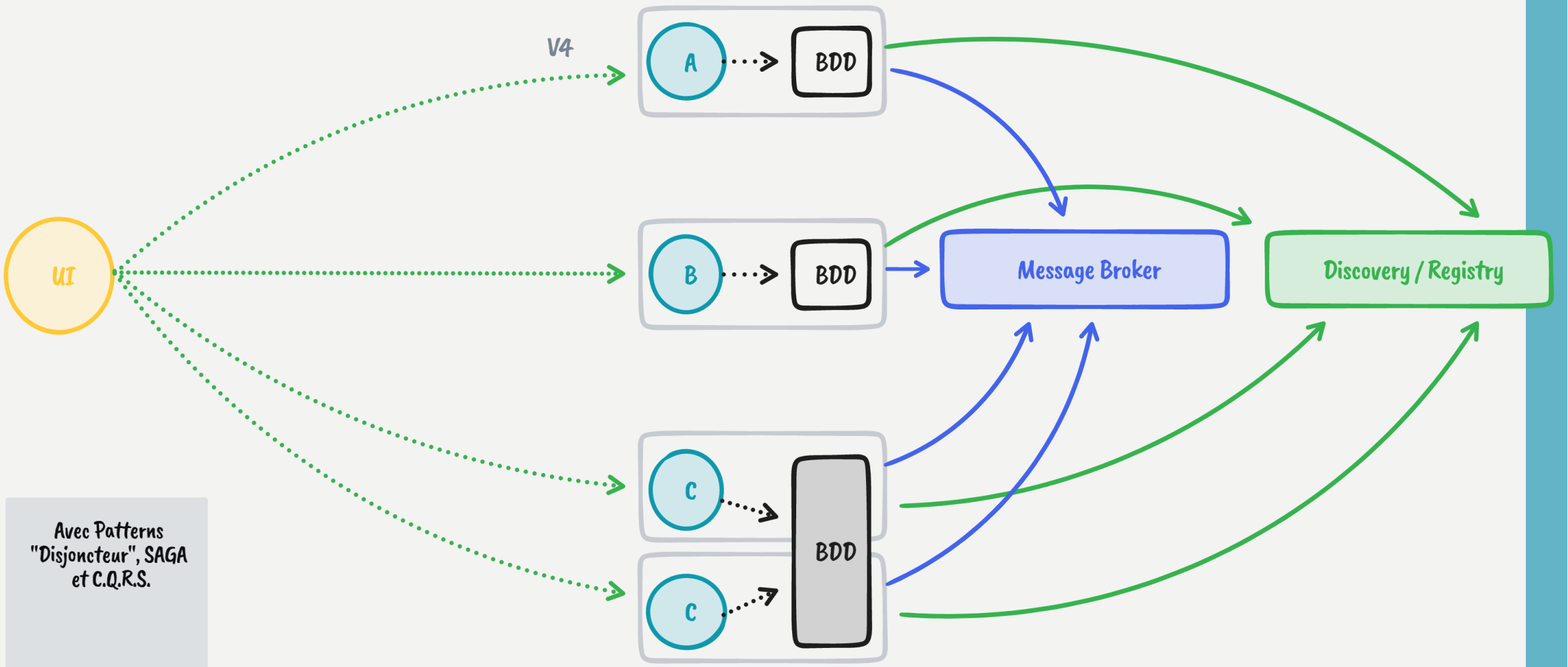
L'utilisation d'un Message Broker n'est absolument obligatoire.
C'est néanmoins une solution technique souvent implémentée.
Par ailleurs, le Message Broker est ici généralement un Event Store



C.Q.R.S.

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

C.Q.R.S.


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

C.Q.R.S.

- Utiliser RabbitMQ déjà mis en place
- Créer un 3eme service qui se chargera de stocker les informations produit & note dans une base MongoDB
- Implémenter les évènements dans les différents services