# ARCHITECTURE LOGICIELLE
# CQRS ET EVENT SOURCING

Enzo LACROIX Hafez MADI Rémi ANTOINE



iut du Limousin
Université de Limoges

# Qu’est-ce que c’est ?

CQRS (***C**ommand **Q**uery **R**esponsibility **S**egregation*) et Event Sourcing sont des design patterns. CQRS introduit 4 concepts: les commandes, les requêtes, les agrégats et les projections. Les requêtes permettent de lire l’état d’un système sans le modifier et les commandes permettent de modifier cet état. Les agrégats regroupent le modèle d’écriture en une seule entité, ce qui garantit la cohérence des transactions.
Les projections sont une représentation des objets métier selon différentes formes et structures.
Le but de ce pattern, comme son nom l’indique, est de séparer les commandes des requêtes, la partie écriture et lecture.

L'Event Sourcing vise à capturer tous les changements d'état d'une application sous la forme d'une séquence d'événements. On ne stocke plus l’état actuel des données mais l’historique de tous les évènements. Pour récupérer l’état actuel, on a plus qu’à rejouer l’historique de modifications. Un événement est donc un enregistrement d’une action qui a changé l’état du système (ex: "UserRegistered",  "OrderPlaced"), les événements sont immuables et stockés séquentiellement (tableau, liste…).
Ce pattern introduit également les notions de magasin d’événements, structure de données chargée de stocker les événements, agissant comme source de vérité pour l’état du système.

Ces deux patterns se complètent bien, Le CQRS fournit une structure claire pour traiter les commandes et les requêtes, pouvant même aller jusqu’à séparer leurs modèles/bases de données respectives. L’Event Sourcing garantit l’enregistrement de chaque modification dans le système sous forme d’événements, qui peuvent être utilisés pour construire les modèles de lecture requis par les requêtes du CQRS.

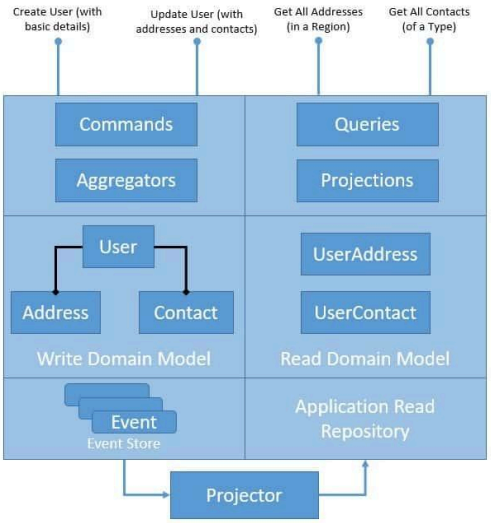


*Schéma CQRS et Event Sourcing*

# Avantages

L'Event Sourcing permet d'avoir une traçabilité, chaque changement d'état étant enregistré en tant qu'événement, on dispose de l'historique complet de ce qui s'est passé, par qui et quand. L'analyse des flux d'événements permet de détecter des schémas, ce qui permet une logique d'entreprise complexe comme la détection de fraudes. On est aussi flexible dans la modélisation des données, il est facile de modifier la manière dont les données sont lues sans en affecter la logique de leur écriture.
Le CQRS quant à lui, permet d'écrire un code propre, facile à étendre et la gestion des accès concurrents.

La combinaison de ces deux patterns permet d'obtenir des architectures de système hautement évolutives, flexibles et robustes qui peuvent évoluer dans le temps tout en maintenant l'intégrité et les performances.

# Limites

CQRS engendre une complexité dans la maintenance et l'administration, de plus, la consistance des données de lecture n'est pas garantie à 100%.
Event Sourcing peut être difficile à appréhender côté développement, l'agrégation de plusieurs magasins d'événements peut être complexe, les modèles sont plus complexes à modifier.
Ces deux patterns introduisent donc une grande complexité de mise en place et apporte une révolution dans la manière de gérer les données ce qui peut déplaire aux développeurs qui doivent les implémenter. Vouloir implémenter ces patterns dans un projet existant peut s'avérer ardu, il est mieux de les utiliser dans un projet dès le départ, ce qui implique une phase d'analyse pour choisir les patterns qui conviennent au projet.

# Application

Nous avons fait une application en Rust démontrant les patterns CQRS et Event Sourcing. Dans cette application, nous avons des clients constitués d'un nom, d'une adresse e-mail et d'un montant d'argent. L'application permet d'afficher les clients, en ajouter et supprimer, les créditer ou débiter d'une certaine somme d'argent. L'affichage des clients est une requête tandis que les actions pour ajouter, supprimer, créditer et débiter un client sont des commandes qui engendrent des événements qui sont enregistrés. L'état actuel d'un client est dérivé en rejouant tous les événements le concernant.
Le code source de l'application se trouve dans:
https://github.com/Mkdirs/cqrs-event-sourcing

L'application peut se lancer dans un conteneur Docker, pour cela il faut:

- Cloner le projet
- Se placer dans la racine du projet
- Lancer les commandes:

```
- docker build -t nom .
- docker run -it --name=<nom> nom:latest
```

Comparer à d'autres principes similaires:

# Sources

https://medium.com/codeshake/cqrs-events-sourcing-avec-gcp-1-3-introduction-e5a62bbdae28
https://www.youtube.com/watch?v=Q0Bz-O67_nI
https://www.baeldung.com/cqrs-event-sourcing-java