Event Sourcing/CQRS par la pratique

Speaker : Clément Heliou (Xebia)
Format : Conférence
Date : 7 avril 2017
Slides : https://speakerdeck.com/clementheliou/eventsourcing-par-la-pratique

Clément partage son expérience de 2 ans sur les patterns d'Event/Sourcing et de CQRS dans une grande banque française.

L'Event Sourcing en bref

Pattern d'architecture. Ne dépend pas d'un langage ou d'un outil.
« Capturer tous les changements d'état d'une application comme une séquence d'évènements » Martin Fowler.
Le changement d'état => évènements métiers.
Tout changement a une cause unique.
L'Event Sourcing est en vogue depuis quelques années mais existe depuis des années (système COBOL ou base relationnelle).
Exemple : opérations de crédit stockées par les banques permettant de recalculer le solde d'un compte.

Sauvegarder tous les évènements comportent plusieurs avantages :

- Audit : particulièrement intéressant en Finance
- Analyse/Debug : permet de retrouver comment l'utilisateur s'est retrouvé dans tel état
- Reprise de données : en cas de panne, on peut rejouer les évènements pour remonter un système. Permet également de reproduire un système de la prod vers la recette
- « Performance » : l'Event Sourcing n'amène pas de la performance. Mais l'implémentation de ce pattern utilise des technos scalables

Termes utilisés :

- Commande : requête de l'extérieur sur le système. Une commande est auto-descriptive.
- Fonction de décision : fonction déclenchée à l'arrivée d'une Commande. Elle prend l'état courant, prend la commande puis publie un ou plusieurs évènements. Les évènements sont auto-descriptifs.
- Event Store : stocke les évènements
- Fonction d'évolution : 2^{ième} fonction, pas de code métier, mute l'état de l'application
- Action : commande interne du système qui décide d'effectuer une commande sur lui-même
- Effets de bord : notifications aux partenaires une fois l'application dans un état stable

Contexte métier du retour d'XP

Application « Multiplus » développée à la SG et traitant des matières premières. Ces matières ont un cours dans le temps.
Exemple d'un constructeur de voiture qui achète une quantité de paladium pour ses pots catalitiques.

Exemple :

1. Commande : RecevoirPrix
2. Evènement : PrixReçu
3. PrixReçu stocké dans l'ES
4. Fonction d'évolution : utilisation d'une machine a états finis. Permet de s'affranchir d'un code procédural. Utilisation d'un DSL.

Implémentation :

- Un Pool de Thread par broker de messages (IUM, EMS)
- Queue bloquante pour stocker les commandes
- Un unique thread pour l'event loop. Le temps d'exécution d'une commande doit être extrêmement rapide (quelques ms). Attention : lors de l'arrivée d'une commande, il faudra prendre en compte le temps d'attente dans la queue et le temps d'exécution.

Optimisations non retenues à cause des autres points de contention dans l'application :

- utilisation d'un bus asynchrone. Problème de désynchronisation du temps.
- Démulitplication des Event Loop : demande de l'affinité pour que toutes les commandes d'un même type arrivent dans le même Event Loop
- Démultiplier les Queue de commandes

Ce que Clément a appris :

- On peut (doit ?) commencer l'Event Sourcing sans framework
- Ne pas confondre Command Sourcing et Event Sourcing. Dans le 1er cas, les partenaires doivent être idempotents + photo avec la marge
- Dénommer un événement n'est pas chose aisée. Ne pas publier des choses ambigues. Ne pas utiliser des verbes trop techniques (ex : Créer Utilisateur).

CQRF en bref

Commande and Query Responsability Framework
Pas besoin de librairies ou de framework.
Une commande modifie l'état du système. Une query observe l'état.

Côté écriture, on s'assure que les données soient cohérences (transaction ACID), normalisées.
En lecture, les besoins sont différents : dénormalisation de données.
Accès les performances sur la lecture car en général les applications font plus de lecture que d'écriture.
CQRS permet d'avoir plusieurs vues d'une source de donnée.

Besoin métier sur Multiplus : monitoring métier pour le suivi temps-réel des transactions, contrôles basiques par clients et produits
Attention à l'ordre d'arrivée des évènements : on reprend tous les évènements antérieurs pour recalculer l'état.

Ce que Clément a appris sur CQRS :

- Il existe toujours un contrat même s'il est implicite. Utilisation de protobuf pour sérialiser les évènements
- Poser la 1ière Pierre, à l'édifice CQRS