ADR 005: Use a CQRS Pattern

We have a microservice architecture with several REST services that would need to read and write data. Sometimes it's difficult to design those services to do both tasks efficiently. In the Farmacy Food system, the design of writes (commands) is primarily concerned with the integrity of data and business rules, whereas the design of reads (queries) has performance (response time) as a key requirement.

Decision

We will use the CQRS pattern applied to microservices with this design:
- an independently deployed service takes care queries. It reads data from a query view that is optimized for data reads. ("Optimized" means using DB technologies and techniques that favor performance, such as: NoSQL, denormalized view, memory cache, sharding, replication.)
- an independently deployed service takes care of commands. It may be a REST service that gets POST commands or a reactive service that subscribes to a pub-sub topic and gets messages that updates the primary database for the scope of that service. - a batch component (example: K8S cron job or logic activated by database-triggered events) synchronizes the data between the primary database and the query views. The frequency of data synchronization varies–it can be trigger-based, every 5 minutes, every hour, ..., once a day, etc.

In our design, we have used CQRS for: - Inventory - Orders - Subscription

Rationale

We expect that the app will issue far more query requests than commands. For example, the inventory is checked in many customer operations, such as looking at a catalog meal item. The CQRS pattern is used to address performance (response time) requirements on queries.

Status

Proposed

Consequences

- The CQRS pattern uses data replication for query views and hence *eventual consistency*. Therefore, query services may operate on stale data (during the interval between data synchronization from primary db and query view).
- Need to monitor data synchronization tasks, with notification and proper action in case of failure.