Lab 9

Part 1

Write a ProductCommandService where you can add, delete and update products. Products have a productNumber, name and price.

Write a StockCommandService where you can add, delete and update the number of products we have in stock. A Stock class has the attributes productNumber, quantity.

Write a ProductQueryService where you can retrieve products. Products have a productNumber, name, price and numberInStock.

Give all services their own Mongo collection.

Make sure that if products or quantity in stock is changed, this will show up in the products we get back from the ProductQueryService service.

Part 2

Change the ProductCommandService so that it uses event sourcing. Everything should work the same as your solution of part1.

Part 3

Suppose we have a flight booking system that contains 3 microservices:

1. Seat reservation system: Reserves a specific seat on the flight and updates its local database.
2. Payment system: Processes the payment and updates its local database.
3. Ticket system: Issues the e-ticket to the customer and updates its local database.

We want that the functionality to reserve a seat, process the payment and issue the ticket is transactional.

Implement all 3 microservices (as simple as possible, but with MongoDB database access) and implement the SAGA transaction using orchestration. Make sure your solution can handle payment errors and ticket issue errors.

Part 4

Implement the SAGA transaction using choreography. Make sure your solution can handle payment errors and ticket issue errors.

What to hand in?

1. A zip file containing all services for part 1
2. A zip file containing all services for part 2
3. A zip file containing all services for part 3
4. A zip file containing all services for part 4