Project title: Web banking project

Team: Denys Herasymuk

Project aim

In this project I am going to develop a web banking application based on a microservice architecture. The system should have the next functionality:

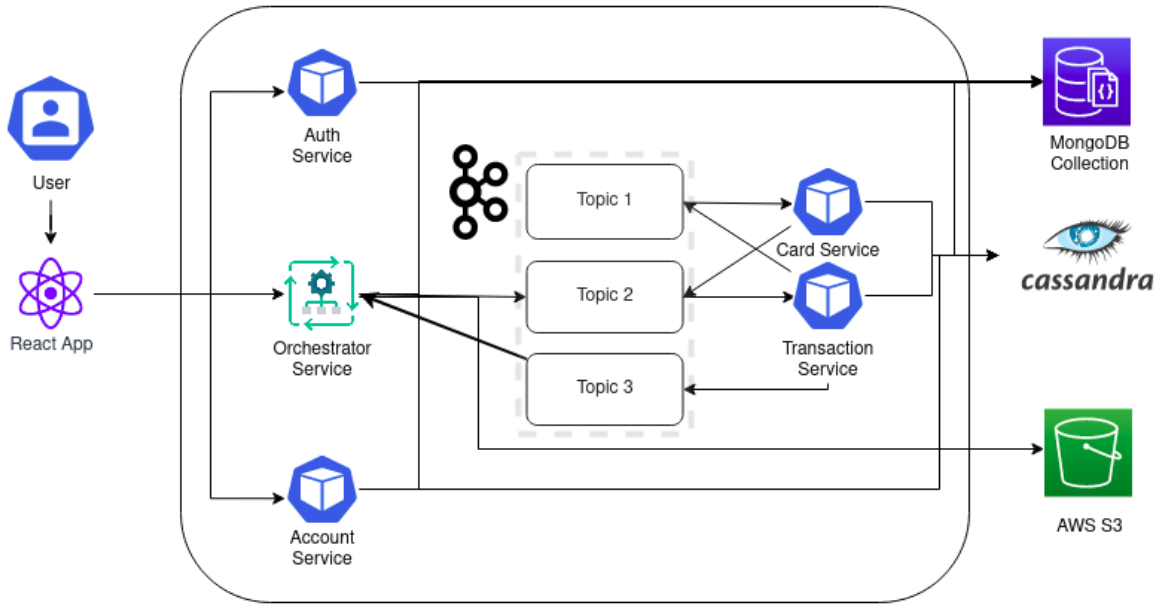
- User registration
- User login
- Send money from one card to another
- Top up your card balance
- List your transactions on the main page

The main motivation is the next. Last spring semester I had a team course project with the same topic, which we developed using Python. But now I want to have practice with all technologies below. Since I already had a similar project, I reuse React UI from there. Therefore, in this Java course project I want to create a REST **backend** for this UI. Here is a course project mentioned above (please take a look and put a star :+)) – https://github.com/denysgerasymuk799/UCU_Software_Architecture_Project

Technology stack

- Technologies: Kafka, Java
- Frameworks: Spring Boot, Kafka Streams for Spring
- Databases: AWS Keyspaces (serverless Cassandra), Cloud MongoDB
- Java features: development of microservices, REST API development, interaction with Kafka (event-driven design), interaction with two NoSQL databases, logic with JWT tokens for authorization

Architecture overview



On the above diagram, you can see a high level architecture of the web banking system. Let's analyze it from the beginning. User opens a website with React App and interacts with its functionality. Front-end communicates with 3 REST API microservices: Auth Service, Orchestrator Service and Account Service.

REST API Microservices

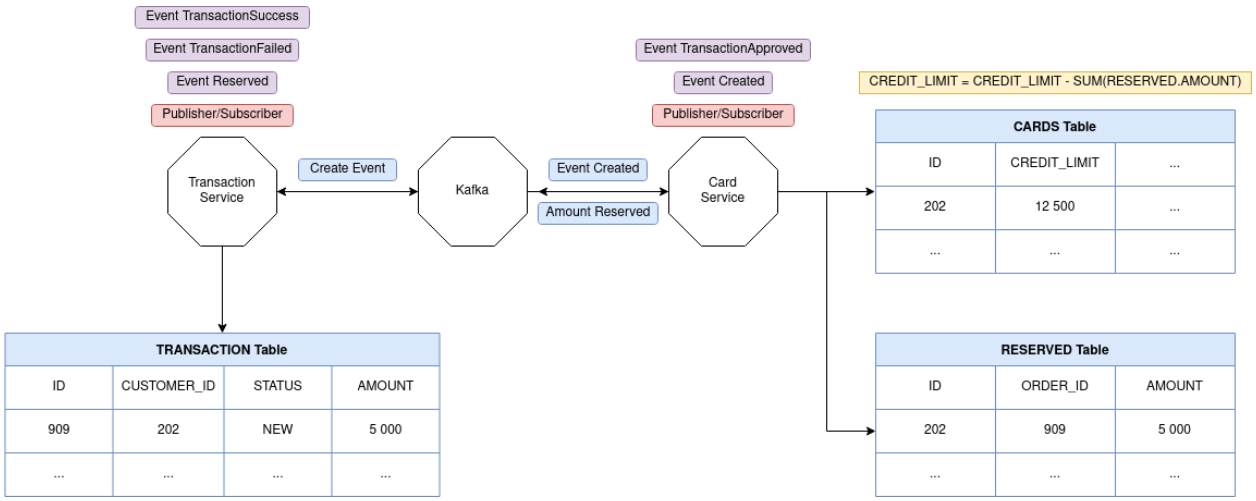
Auth Service enables registration and login functionality in the system. When a user is registering, this service assigns a card number to him from the prepared card pool and saves his data in MongoDB collection. Additionally, it saves the assigned card number in the 'Cards' table in AWS Keyspaces, since this NoSQL database is used for transaction logic including sending money from one card to another and card money top-up. When a user is signed in, a JWT token is generated and returned to the web, which saves it in the user browser cookie.

Orchestrator Service handles transactions and returns their statuses to the web. When the front-end sends him a new transaction, it puts it in a separate topic in Kafka for future processing with Card and Transaction services. Since interaction with Kafka is asynchronous, we need to know status when a specific transaction is processed. For that, Orchestrator service is polling AWS S3 with special folder structures for faster searching of each user transaction statuses.

Account Service returns a specific card balance and past transactions to the front-end. It is needed to display them on the main page of the web.

Transaction Logic

Card and Transaction Services process each transaction based on the following logic with Cards and Reserved tables.



This logic is based on events and it is very similar to that one, which is used by real fintech applications, since, firstly, money is reserved, and after specific checks they are withdrawn.

Note that I am going to develop this system locally without deployment on AWS cloud. I have a great desire to do it, but currently I see that I have no enough time for that. Also since I will develop this system alone, I will have no enough time to create a well-designed system of microservices, which will satisfy the majority of microservice patterns, but I know them :-)