Technical University of Denmark

02267: Software-development of Web-services

DTUPay

Jeff Gyldenbrand - Student no.: s202790
Christian Bruun Nielsen - Student no.: s153867
Ergys Kajo - Student no.: s181412
Daniil Provornii - Student no.: s192678
Antoine Sébert - Student no.: s193508
Hussain Tariq Awana - Student no.: s181434

January 19, 2021

Contents

1 DTUPay - Description and Scope

The goal of the exam project for 02267:*Software-development of Web-services* is to design and implement a web application as a collection of microservices. The suggested name for the application is DTUPay and, according to the exam description, it has a similar functionality to the backend of a widely spread MobilePay app from Danske Bank. In order to make DTUPay work in the same way, several micro-services were implemented, while each of them focuses on a specific functionality of DTUPay.

Much like MobilePay, DTUPay has to implement easy transfers of money between two users. The main scenario is defined to be a transfer from a customer to the merchant, while the bank assistance have been completed outside of the system and is out of the implementation scope. Therefore DTUPay needs to focus only on the transfer of money between two accounts of registered users. Three micro-services have been outlined for implementation: AccountService, PaymentService and TokenService. Moreover there is an end-to-end testing system designed to simulate the client side scenarios (both for the customer and the merchant), where the main usage steps are performed.

In order to describe each microservice independently, there are specifications presented as follows:

- **AccountService** is delegated to handle the account management for users. This entails registration and storage of the application users.

- **TokenService** manages the allocation of tokens for customers. The tokens are used to validate the customer and keep its identity anonymous in the payment service while a transaction occurs. There are some rules in regards to how many tokens an account can have allocated at a time, with six being the maximum amount of unused tokens.

- **PaymentService** is tasked with handling the transfer of money between accounts. It is also the only entity to diferentiate between customer and merchant in its business logic.

- **End-to-end Test**, referenced also as *ClientApps* runs the main user scenarios and verifies the output. Conceptually it covers the interface for customer (called CustomerAPI) and merchant (MerchantAPI) to interact with the business logic or i.e. connection to REST adapters on the DTUPay side. Therefore this system verifies whether the users are correctly performing the requests they can access.

To start with the project, the team discussed the resource triangle. There are two parameters which are fixed (time and quality) but functionality was variable. One can control the functionality by sorting out the order of the user stories in the project. Team progressed with the use of Agile Software Development, and prioritizing the user stories, which satisfy the requirements with the highest value for users.

2 Architecture

2.1 General diagram

The general diagram below is an illustration of final implemented system. There is shown the transformation of one monolithic to different micro-services like:

- Payment Service
- Account Service
- Token Service

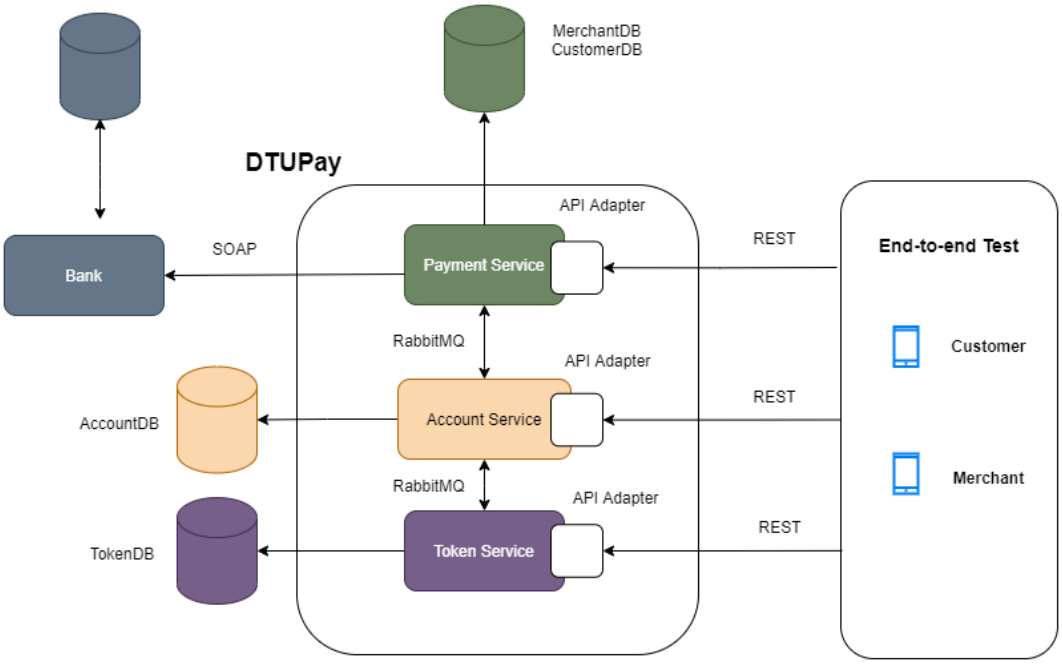

Figure 1: DTU Pay Diagram

2.2 End to End Test System

The merchant and customer applications are simulated via end to end test system. The system is able to create valid HTTP requests to a specific service adapter in DTU Pay infrastructure.

Customer API
+String registerCustomer(String name, String account) +String[] getTokensForCustomer(int tokensAmount, String customerID)

Figure 2: End to end class diagram

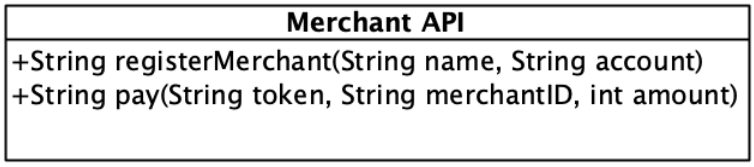

Figure 3: End to end class diagram

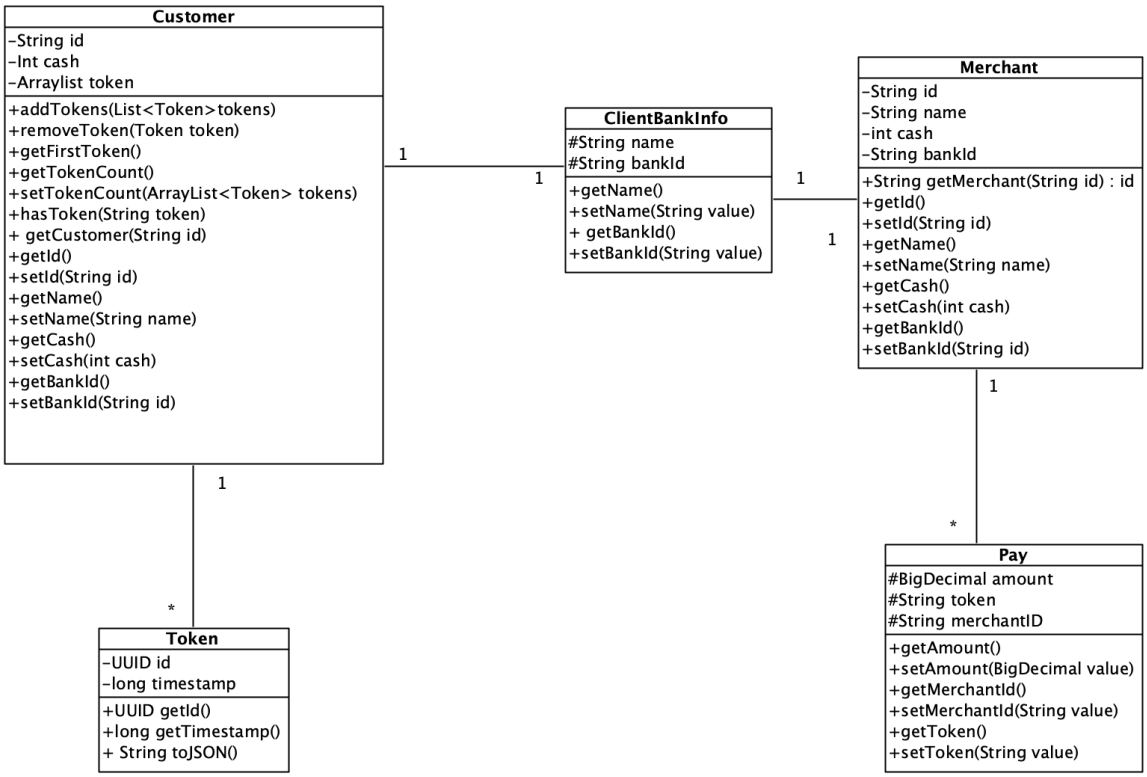

Figure 4: Class diagram

2.3 Payment Microservice

In this microservice, the merchant initiates a payment from the customer. Then the payment service check for the validation of the token provided by the customer and if it is valid the payment will be successful.

Resource	Endpoint	Meaning
/pay	POST	Make a translation between merchant and customer

Figure 5: Payments

2.4 Token Microservice

Token is an object used by the customer as the proof of transactions in the app. A customer will request for a token or tokens up to maximum limit(5) and use the token for payment transaction. The token are given to merchant to make payment and who will processed the payment to DTUPay then it will be validate and perform the transaction. The token is composed of a UUID (universal unique identifier) and a timestamp.

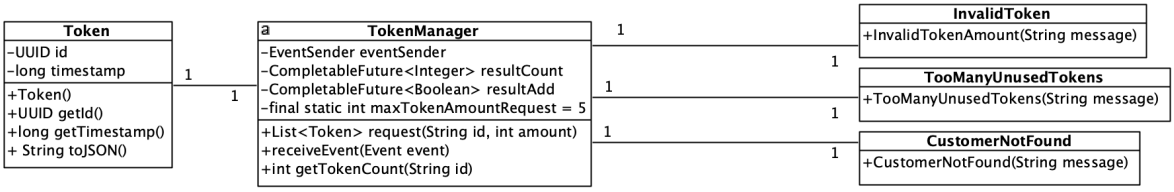

Figure 6: Class diagram of token service

Resource	Endpoint	Meaning
/token	GET	Acquire a list of tokens
/token/{id}	GET	Get a specific token
/token/request/{id}	GET	Request amount of tokens

Figure 7: Token

2.5 Account Microservice

In this microservice there are stored generic information about the user profile in the DTUPay database.

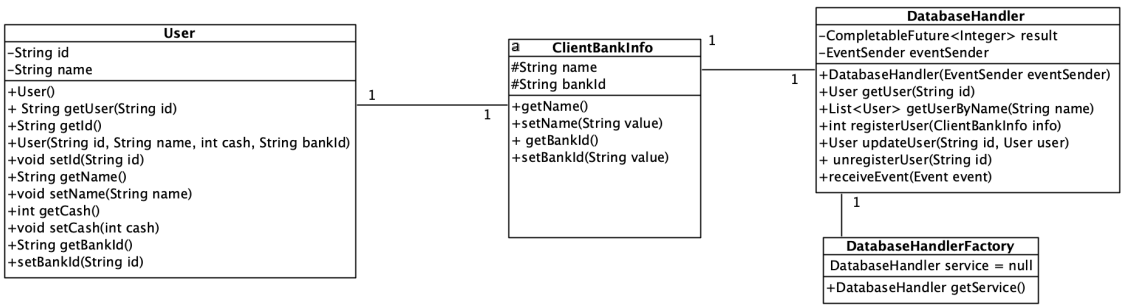

Figure 8: Class diagram for account databasehandler

Resource	Endpoint	Meaning
/account/{id}	GET	Get the bank account form provided bank id
/account/{id}	DELETE	Retire the bank account

Figure 9: Account

Resource	Endpoint	Meaning
/accounts	GET	Get a list of all bank account
/accounts	POST	Create a bank account given provided user and balance

Figure 10: Accounts

3 Short description of the team work

Teamwork in the group was carried out in different places such as Microsoft Teams, Zoom, and Discord. Every day, started with a team meeting at 9 am and followed by a team meeting at 4 pm. During this time, everyone started to do individual tasks first, so that they can solve problem scenario and helped each other. To resolve team's issues and personal problems were discussed the with TA's and Hubert via Zoom meetings.
For the final project, the tasks were divided into different sub teams. Like the first week team started with a team meeting at 9 am where sub teams specified and were given separated daily tasks and a follow up meeting was held at 5 pm where each sub teams provided an overview of the work that the did during the day and worked together on unsolved tasks.

4 Description of the YAML-files

The REST-interfaces are formulated as swagger YAML-files[1] to provide a good overview, and to do some quick and dirty testing of the REST-endpoints. The files are uploaded as separate files.

4.1 Users.yml

The default port for this service is 8083.

Endpoint	Method	Parameters	Produces / Consumes
/users/	POST	JSON	plain text
/users?name={name}	GET	{name} users name	JSON

4.2 Pay.yml

The default port for this service is 8081.

[1] https://swagger.io

Endpoint	Method	Parameters	Consumes
/pay/	POST	JSON	plain text

4.3 Token.yml

The default port for this service is 8082.

Endpoint	Method	Parameters	Produces
/token	GET	*null*	plain text
/token/{id}	GET	{id} customer ID	plain text
/token/request/{id}?amount={n}	GET	{id} customer ID, {n} number	JSON

5 Repository

The source code is present in GitLab repository at URL:

`https://gitlab.com/02267-group-6/dtupay.git`

6 Contribution Table

Part	Team member
Payment Service	Jeff, Christian
Account Service	Jeff, Daniil, Christian
Token Service	Antoine
Database Overlay	Jeff
End-to-end tests	Daniil, Hussain, Ergys
Docker and scripts	Antoine, Daniil
Jenkins setup	Antoine
Description and Scope	Christian, Daniil
Architecture	Ergys, Daniil, Hussain
Short description of the team work	Ergys, Hussain
Users guide	Daniil
Description of the YAML-files	Jeff, token: Antoine
Installation guide	Jeff, tests: Antoine