DTU-Payment-Service

Project Structure

The DTU-Payment-Service consists of 7 projects.

1. The `end-to-end` tests.
2. A Token Management microservice in `token-management-service` responsible for managing, consuming and issuing tokens.
3. An Account Management microservice in `account-management-service` microservice that is responsible for accounts management in DTU Pay.
4. A Payment Management microservice in `payment-management-service` that is responsible for the communication with the SOAP-based bank service.
5. A Report Management microservice in `reporting-service` that is responsible for reporting services for customers, merchants and managers.
6. A facade microservice in `dtupay-facade` that offers an external REST interface. Also, this service communicates through message queues.
7. A message utilities in `messaging-utilities` provided by Hubert Baumeister.

Dockerization

In our docker-compose files, we run the `rabbitMq` container first and we have set up custom healthcheck on which the other services depend to make sure it's ready to accept communication with the other services.

Also, we use the eclipse image `eclipse-temurin:23-alpine` as a base in our service DockerFiles in order to optimize the size of the images.

Prerequisites

Before proceeding to the installation of the project, make sure you have the following technologies installed:

1. Docker
2. Java 23
3. An IDE, such as IntelliJ, or Eclipse

Installation Guide

1. **Clone the repository**: `git clone https://github.com/Zedrichu/DTU-Payment-Service.git`
2. Navigate to the end-to-end-test folder `cd DTU-Payment-Service/end-to-end-test`
3. To build and install the system locally using docker, execute the following script: `sh ./build_deploy_test.sh`
   The script builds and deploys the docker images, run the tests and then stops the images.

   In case you want to keep the docker images turned on after running the tests, instead of running the `./build_deploy_test.sh` from `./end-to-end-test` directory, run the `./build_deploy.sh` script.

Jenkins Access

Jenkins URL

Credentials: `Login: huba | Password: group18`