Documentation:

My app is a lending engine built using Spring Boot for Java as this framework is great for building a backend for this type of service. I'm using Maven as a dependency manager with the following dependencies for the lending microservice: Spring Web, Spring JPA, Spring Actuator and Spring H2 database, ampq and gson . The security microservice has the following dependencies: Spring Web, Spring H2 database, Spring security, ampq and gson. The profile microservice has the same dependencies as the lending microservice. The lending engine microservice runs on port 8080, the security microservice runs on port 8081 and the profile microservice runs on port 8082.

The app is going to have 3 main parts: Security, Profiles, Lending Engine The app will have 3 types of users: simple users, lenders and admins. The User will have the following functionality: Registering,Withdrawing money, Logging in, Repaying Loans and Requesting Loans The Lender will have the following functionality:Registering, Withdrawing money, Logging in, Checking Loans and Lending money.

The lending engine is comprised of 3 main classes :Users, Loan and Loan Application. I'm using 3 repositories to store data for Loan applications, Loans and Users. The loan controller is the rest controller that manages all the endpoints for the lending engine. I'm using the H2 in-memory database for this app, and the repository interfaces serve as the database. The lending engine and the security microservice are connected in the application.properties file using the security.baseurl=http://localhost:8081 in the lending engine. The loan request is a class that is used to model the request that will be sent to the loan controller. The microservice also implements two exceptions: UserNotFoundException and LoanApplcationNotFoundException that are used by the LoanService class in the acceptLoan function to signal a bad accept request. The MessagingConfig class is used to communicate with the RabbitMQ service so that there is a notification when a user is registered by the security microservice. The loanApplicationAdapter class is used to implement functionality for the instantiating a loan request based on the request sent to the endpoint in the loanController.

I'm also using RabbitMQ, a messaging broker - an intermediary for messaging. It gives my microservcies a common platform to send and receive messages so that they safely sit there until they are ready to be

consumed. That way if the security microservice has 100% availability and the other 2 microservices don’t, we won’t have problems like for example: a user is registered by the security microservice, but the other microservices don’t register that user.

The profile microservice is small right now but it will be used to store the profiles (which contain lots of information) of the users of the app, the lenders, the users that are looking for loans and the admins of the app. The security microservice right now registers users with a password and sends the registered user also to the lending microservice. Endpoint requests in the lending engine can also only be made using authorization by using the username of  a user registered in the security microservice.