Distributed Systems Project

Joona Pellinen

The system I chose is a taxi service system.

Task 1

Functional system requirements:

1. Passengers can request a ride and see the status of the ride, such as where the taxi currently is and how long it will take for it to arrive
2. Passengers can register as users which helps with payment and other services
3. Passengers can pay for the ride using different payment methods
4. The system can calculate the taxifare based on the distance and time of the ride
5. Drivers can view the request of the passenger and accept or reject the request
6. The system can provide feedback on the rides based on passenger ratings that they can give to the drivers.

Identified microservices:

1. Passenger ride request microservice, which handles all ride requests and sends them to all available drivers

2. User microservice, which handles the user accounts and their authentication

3. Payment microservice, which handles the payment processing

4. Driver management microservice, which handles drivers accounts, their locations and availability

5. Location microservice, which gets the drivers and users locations

Scope and communication patterns:

1. The passenger ride request microservice provides an API for the passengers, which are requesting rides. The microservice communicates with the driver management microservice to get a list of available drivers. It also communicates with the payment microservice to handle the payment information and transactions.

2. The user microservice provides an authentication method for the users, and the users can register as well as login ot the system. This microservice communicates with the passenger ride request microservice to provide user information and payment microservice to provide payment information.

3. The payment microservice handles all of the payment processing and provides all the payment related information that the other microservices might need. This microservice communicates with the passenger ride request microservice.

4. The driver management microservice manages driver accounts as well as the information that is sent to the passenger ride request microservice: driver availability and location. The microservice provides an API for the drivers to handle their profile and receive ride requests. This microservice communicates with the passenger ride request microservice.

5. Location microservice communicates with driver microservice and user microservice to get their location.

Architecture of the system:

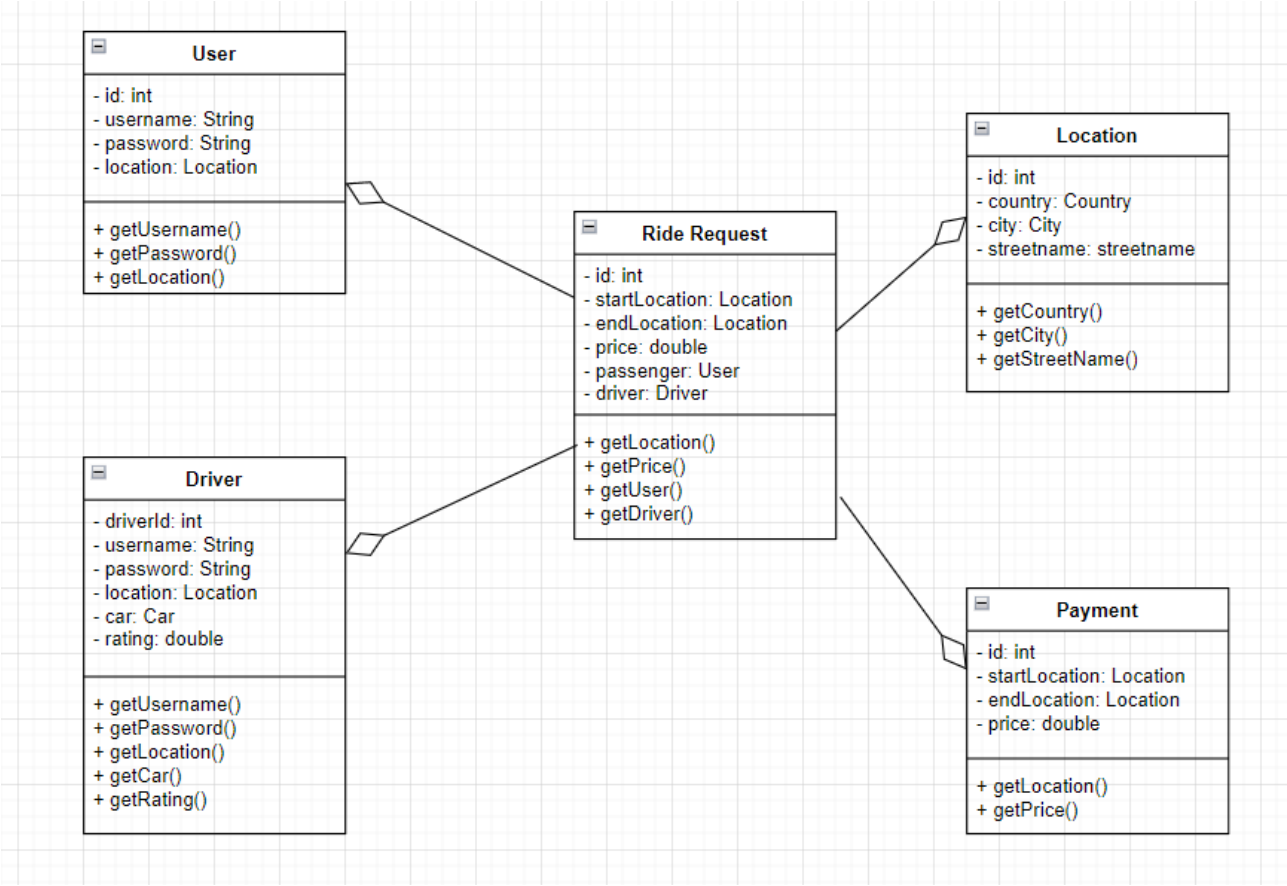

Diagram 1: UML diagram

The previous diagram shows the main communication methods, that the system is using. All of the other microservice are part of the ride request microservice. Communication pattern that is used is REST API. This allows communication between the microservices, because all of the microservices has its own API which communicate between eachother. Other communication patterns that I might use is remote procedure calls.

Limitations that REST API have is that it is more complex system than for example RPC and it has to be well documented to be used in a larger scale system. The remote procedure call has

limitations, such as when the system becomes more compicated and larger, it may not have the capacity to upkeep the system.