IMSE M3

Nikita Antonov 01348746

January 2020

1 How to

1) Make an image for Gradle-based app
First Option:
cd project.imse
./gradlew jibDockerBuild - -image=imseapp

Second Option:
cd project.imse
./gradlew build
go to "libs", move .jar to main repository and name it "imseapp.jar"
docker build -t imseapp .

2) Deploy a stack
docker stack deploy -c stack.yml mysql

3) Start an app
Go to localhost:8080/start

2 NoSQL Database design

ER-Model of my RDBMS:

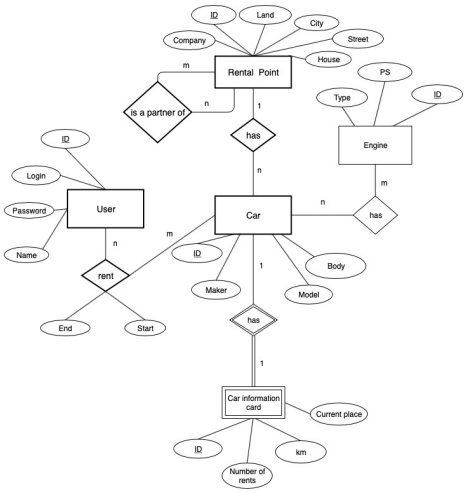

Figure 1: ER-Model

2.1 1:1-Relationship "Car" - "Car information card"

My design decision: It is a 1:1-Relationship, so the only one logical way here is to save "Car information card" by a "Car". Each car has only one Car information card and vice versa, so we don't need much memory here.
Alternative: There is no logical alternative here

2.2 n:m-Relationship "Car" - "Engine"

My design decision: The most logical way here is to save an engine by a car, cause one car has only one engine, so we don't need much memory here and don't have excess references.
Alternative: In theory it is possible to save engines as separate collection and make references to them, but as one car has only one engine, it isn't logical and redundant here.

2.3 1:n-Relationship "Rental Point" - "Car"

My design decision: I will make "Rental Point" as a separate collection with a list of references to the cars, that are located in this rental point.
Alternative: It is also possible to save all cars by the rental points, where they are located. And we don't need much memory for this actually, cause one car is located only in one rental point, so we don't have copies. But for me it is much easier to take information of the car, when it is a separate collection, rather than making a big rental point and searching something inside it.

2.4 Recursive relationship "Rental Point" - "Rental Point"

My design decision: I will save partner rental points in the list of references. **Alternative:** it is also possible to make the separate collection "Partner", but i think, that it is redundant there.

2.5 n:m-Relationship "User" - "Car"

My design decision: I will save "Rent" at the user. Inside i will have start and end date and reference to a car, cause i don't want to make thousand copies of a one car. So i will have at the user a list of all his rents with passing references. **Alternative:** it is also possible to save all information at the user entity, but in this case i will have thousands copies of a one car information, so it isn't logical. It is also possible to make a separate collection "Rent" and reference to it at the user, but in this case it is harder to find the important information so fast, as in my decision.

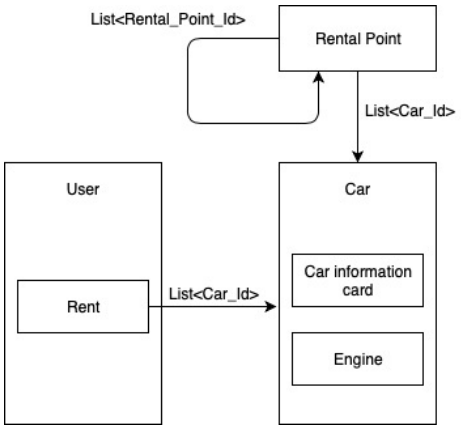

Figure 2: Mongo database diagram (very informal, just to visualize my design decisions)

2.6 Indexing and sharding

I will have indexes for "Car", "Rental point" and "User". As indexes i will use integers (starting from 2601 for "User", from 1 for "Rental point" and from 501 for "Car"). I will have exactly this numbers, because they are so in my RDBMS.
Indexing is very important for sharding, cause Mongo can be a distributed database. I think, that integer values is a goot approach for this goal, cause it is quite easy to split it in intervals. And for me it is easy to migrate from RDBMS and to search between this values. Because of this i have integers.

3 Data migration

I have a special method in my controller (under localhost:8080/migrate) to migrate all my data from MySql to MongoDB.

```
@GetMapping("/migrate")
public String migrate(){
```

Figure 3: Method for migration in controller

It is also possible to make it in one click in cabinet (under "Migration"). First you should log in(it is always occur through MySql, cause it is actually not my main use case, but i should implement it to have a correct work of my main use case). Here user should simply press a button "Migrate" and wait a little bit.

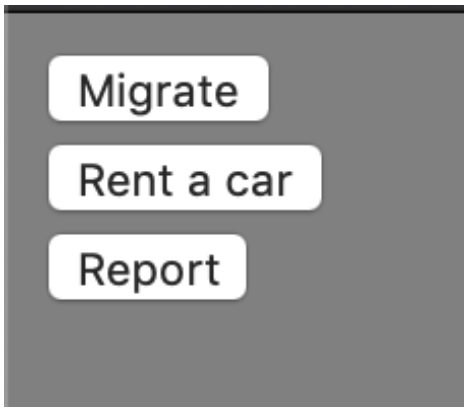

Figure 4: Button for migration in cabinet

4 Main use case and report with MongoDB

I have the same UI for MySql and Mongo, so it is always possible to use one of this databases for my app. One important thing is then user starts for example with mongo, he should do this use case in mongo, not for example enter city in MySql, then car in mongo and so on. But then the user starts the new use case, he can choose the database, which he wants.

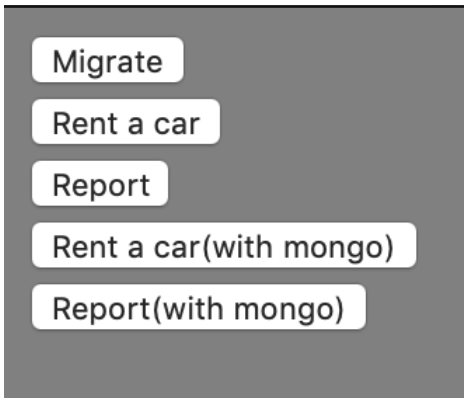

Figure 5: Cabinet with MySql and Mongo

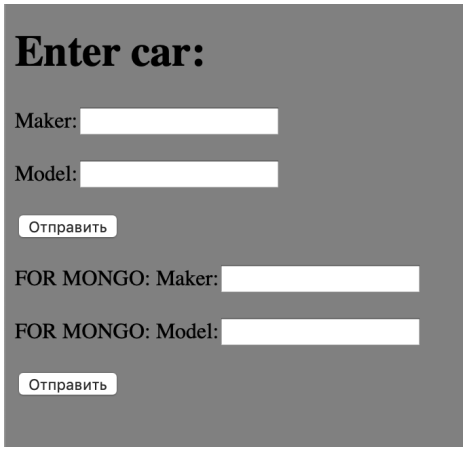

Figure 6: Example of the UI, where it is possible to choose between Mongo and MySql

5 Work protocol

1) ER-Diagram - 1 hour
2) Use case and report diagrams - 2.5 hours
3) Tune spring and databases - 1 hour
4) Tune docker - 3 hours
5) Implement use cases for MS2 - 3 hours
6) Design Mongo - 2 hours
7) Implement MS3 use cases - 2.5 hours