Challenge

Solution description

1. Backend (Project challenge)

Implemented with Java Spring Boot.

Url: http://localhost:8080

To avoid coding all the “setters” and “getters” I used one additional library – Lombok.

Backend application is divided in 3 layers:

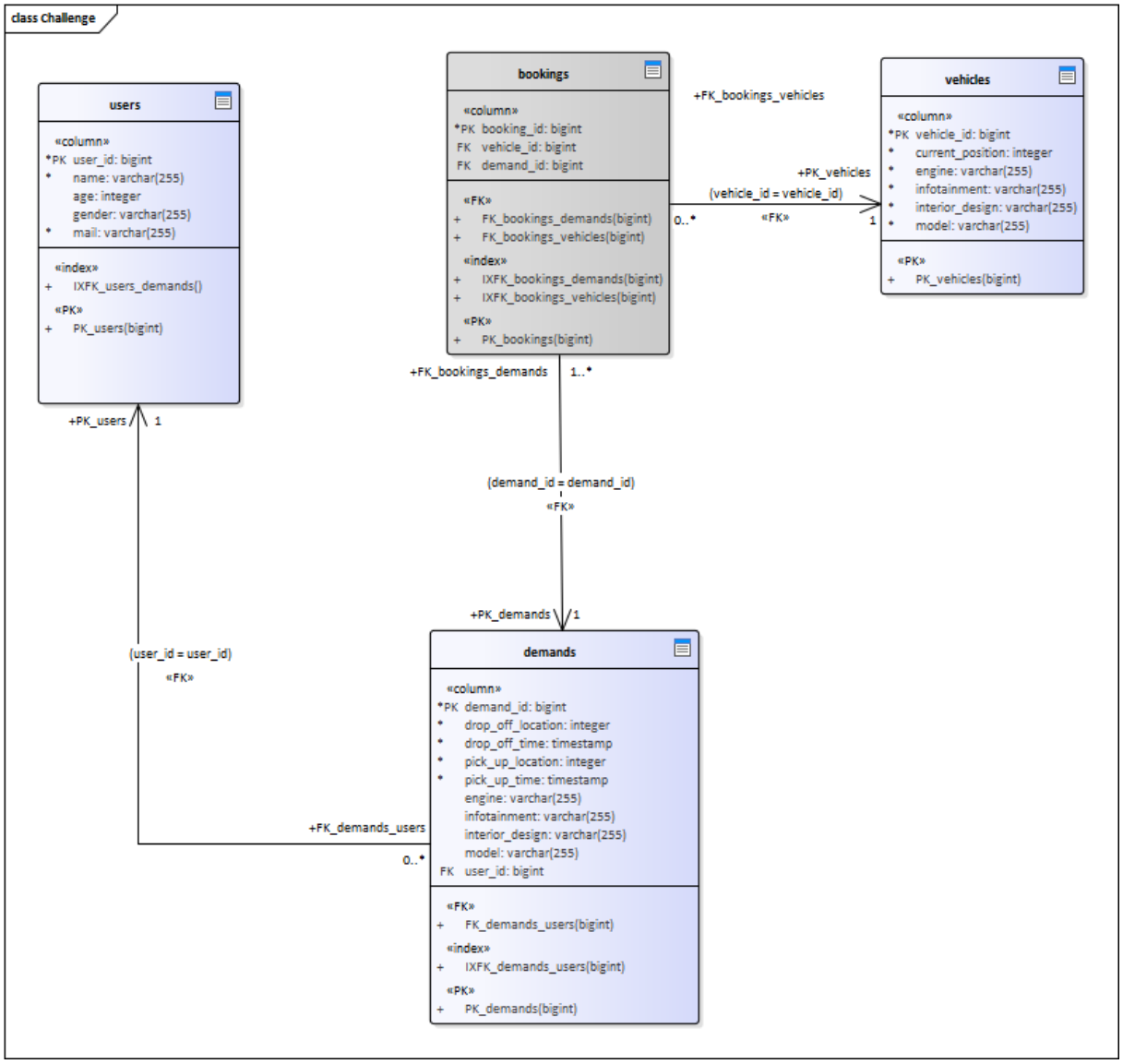
- Service Layer (sl) which contains API and implementation of it. Data validation is implemented in this layer.
- Business layer (bl) which implements business logic
- Data Access Layer (dl) with entities and repositories (DAOs)

Usually for the communication with the clients I use additional SO (Service Objects). An example you will find in UserMgmtImpl. Because of time I didn’t implement it for other controllers.

As database the H2 in memory database is used.

Data model is currently quite simple and contains only three entities: User (table users), Vehicle (table vehicles) and Demand (table demands). To show the traveled distance of the car an additional entity should be introduced: Booking (table bookings). This entity contains information about realized drives for vehicles. Based on that information it will be possible to compute traveled distance based on pick-up and drop-off positions plus the distance driven to next pick-up position of the next booking. An additional constraint will be added to Demand entity. It can’t be deleted any more when related Booking entity exists.

This entity is not implemented yet.



If you want to check the REST interface you can use postman collection in file: postman_collection.json

During the application start some test data is inserted into the database (users and vehicles).

Backend implements also small Scheduled Task which changes position of randomly selected car. The task is triggered each 2 seconds.

Test:

When I'm programming with Spring I usually use 3 levels of tests. I prefer "contract first" approach so the tests are also provided in top-down approach.

1. Starting with controllers I use Spring Boot Test Framework for controllers and I mock other components of the application. Example: DemandMgmtImplTestMvc.java

2. Going down to @services, @components and @repositories I prefer to implement simple jUnit tests which are faster than test with spring context. Here I use mocking with mockito as well.

3. And the last part is full integration test which tests all functionality without mocking any components if possible. Because for this test the whole spring context is needed it starts slowly. It affects the build time of the application. Example: DemandMgmtImplTest.java

2. Frontend (project challengeui)

It is developed with Angular 5. I didn't manage to make it looks smooth and nice with comprehensive CSS. And it is also only a part of functionality which should be developed.

The most "interesting" part is a vehicle page where the current position of vehicles is shown on one dimensional diagram. The component responsible for that implements polling mechanism which updates the data each 2 seconds. The polling is only activated when user visits the vehicle page.

The provided (partial) solution needs a lot of improvements. I hope we will meet and I could explain you what should be done to finish all the tasks.

Best regards,

Przemek