Nielsen project explanation:

This project implements the required operations using Jersey library, the project is deployed as a localhost server using tomcat7. The project uses Azure SQL server for persistent storage.

For simplicity, since it’s only an exercise project, the address, username and password are hardcoded in the source code when accessing the database, you may use it freely.

The project uses JUnit for unit tests, located in src/test/java/carTest.java

For the sake of simplicity, here are the assumptions I used for the Car Service:

1. The car service works 24/7 and appointments can be assigned to any time
2. An appointment has fixed duration time(i.e an hour), so we don’t need to store end time for appointments, only the beginning.
3. The car service has enough workers for any given situation, so it can have any amount of appointments at the same time
4. There are no “persistent clients”, so the data regarding a client should not be stored in a separate table. We only store a name and family name of the client associated with the appointment, thus we have only one table in the database.
5. The status of the appointment is either “Unfinished” or “Finished”, thus we can store the field as boolean.
6. The amount of requests is not huge, so no need to cache sql results

An appointment thus has the following fields:

1. ID (int)
2. Appointment date(Date)
3. Start Time(Time)
4. Price(int)
5. Customer Name(String)
6. Customer Last Name(String)
7. Status(bit)

In order to run the project please use the following maven commands:

mvn clean install

mvn tomcat7:run

For testing, use: mvn test after this

All the API is located under this URI: <http://localhost:9090/carAppointments/rest/appointments/>

Please find below the APIs supported:

1. Get appointment by id, a GET request. URI of the form:  
   [http://localhost:9090/carAppointments/rest/appointments/get/{id](http://localhost:9090/carAppointments/rest/appointments/get/%7Bid)}, where id is the desired id for fetching, will return JSON of the appointment, if not found will return JSON with id=-1
2. Insert appointment, a POST request. URI of the form:  
   <http://localhost:9090/carAppointments/rest/appointments/post>, the body of the request should be of this form(JSON):  
   {

"appointmentDate":"2019-08-29",

"price":300,

"customerName":"Stanislav",

"customerLName":"Radzeiovsky",

"timeStart": "12:30:00"

}

Will return a response, 200 on success 400 on failure

1. Schedule appointment, will create an appointment with random time, a POST request. URI of the form:  
    <http://localhost:9090/carAppointments/rest/appointments/schedule>, the body of the request should be of the same form as 2), but without the “timeStart”. Response is the same as in 2)
2. Update status of the appointment to “Finished”, a PUT request. URI of the form: [http://localhost:9090/carAppointments/rest/appointments/update/{id](http://localhost:9090/carAppointments/rest/appointments/update/%7Bid)} where id is the ID to be updated. Will return 200 on success, 400 on failure.
3. Delete appointment by ID, a DELETE request. URI of the form [http://localhost:9090/carAppointments/rest/appointments/remove/{id](http://localhost:9090/carAppointments/rest/appointments/remove/%7Bid)} where id is the ID to be removed
4. Get by date range, will return all appointments between 2 dates, sorted by price, a GET request. URI of the form [http://localhost:9090/carAppointments/rest/appointments/daterange/{startYear}/{startMonth}/{startDay}/{endYear}/{endMonth}/{endDay](http://localhost:9090/carAppointments/rest/appointments/daterange/%7BstartYear%7D/%7BstartMonth%7D/%7BstartDay%7D/%7BendYear%7D/%7BendMonth%7D/%7BendDay)}. Will return a JSON of all the appointments, if no such appointments, will return an empty JSON list.

You may take a look at examples on how to use the API in file carTest.java under src/test/java.

Looking forward for your feedback!