**Assignment:**

Car Reservation API

The goal of the application is to reserve cars for upcoming rides.

As a user of the system, you can add cars where each car is represented by its make, model and unique identifier following the pattern “C<number>”.

Besides adding a new car, a user can also update one, remove one, or see all the cars (no pagination needed).

The last step in the flow is the ability to reserve a car for a ride at a certain time and duration. The reservation can be taken up to 24 hours ahead and the duration can take up to 2 hours. The system should find an available car, store the reservation if possible and give back a response with the details. The user can also see all the upcoming reservations by calling another dedicated endpoint.

The communication is only via API in the JSON format and the data can be stored just in the memory.

The solution should clearly represent your architectural knowledge and follow the best practices overall including automated testing.

**Technical Requirements:**

\* Java

\* Spring Boot

\* Automated tests

\* REST API + documentation

\* Docker

**Application description and guide:**

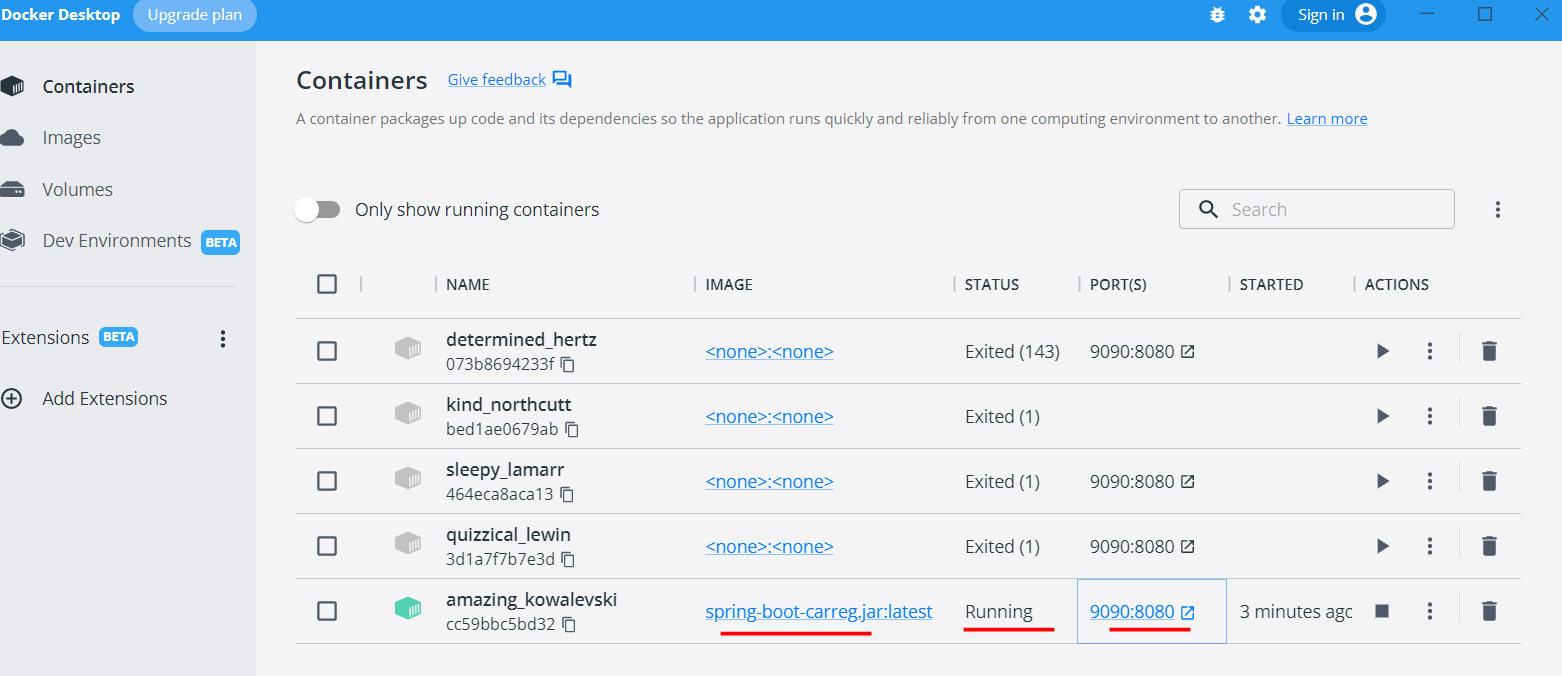
To launch the application you can clone it from <https://github.com/korotkikhandrey/carreserve>

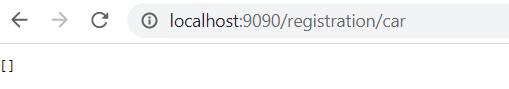
and run it via IDE (class **src/main/java/com/example/carreg/CarregApplication.java**)

or it can be run via docker commands

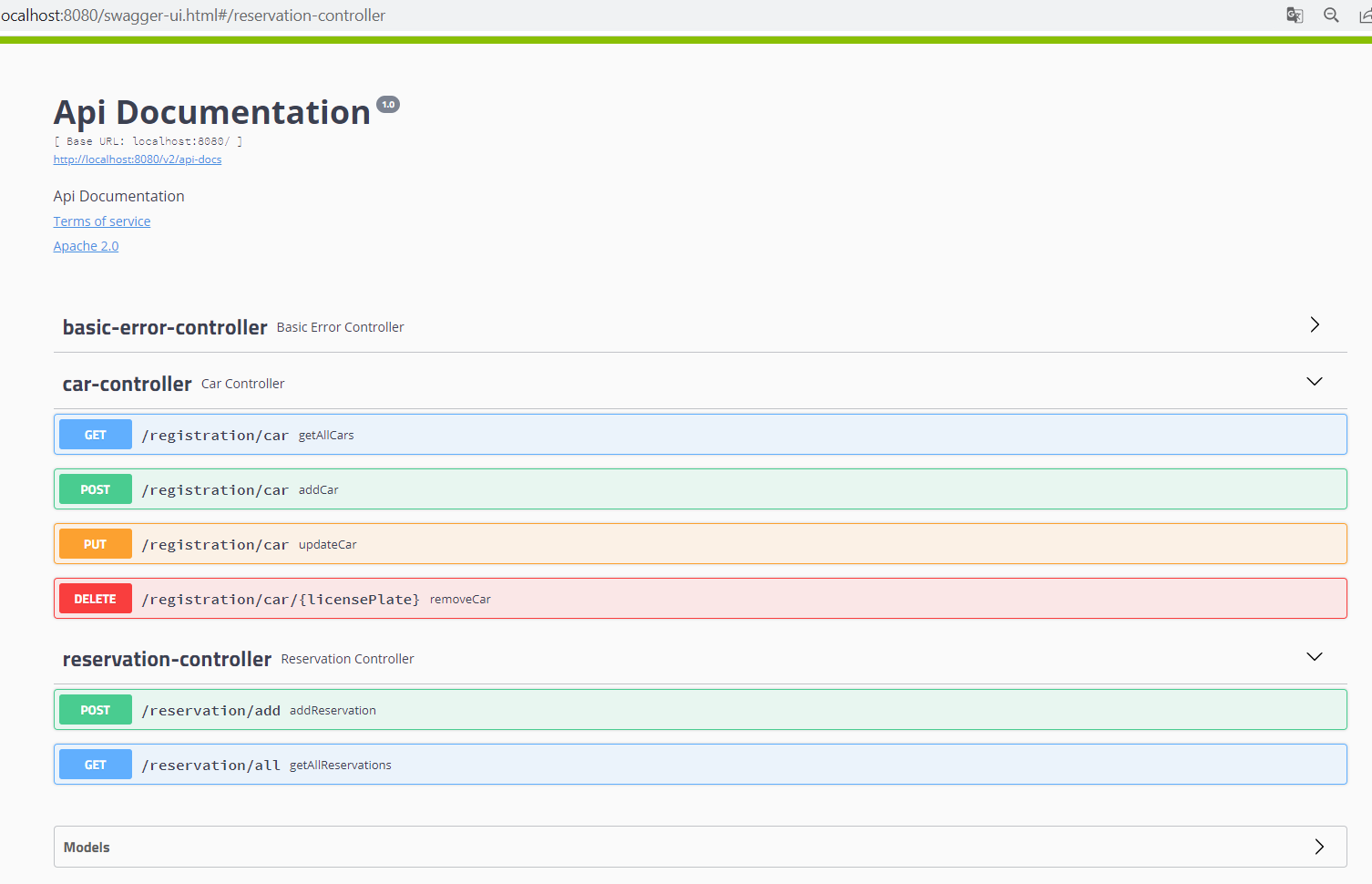
1. **mvn clean install**
2. **Install Docker Desktop and make sure that docker is running (green line at the bottom).**
3. **docker build -t spring-boot-carreg.jar .**
4. **docker run -p 9090:8080 spring-boot-carreg.jar**

Please make sure that after last command docker image is running and the requests can be executed via browser





Swagger is included so it can be visualized with html page. Postman can be also used for POST requests.



Endpoints:

1. POST [**/registration/car**](http://localhost:8080/swagger-ui.html#/operations/car-controller/addCarUsingPOST)

Place the following JSON (or anything you want) into request field:

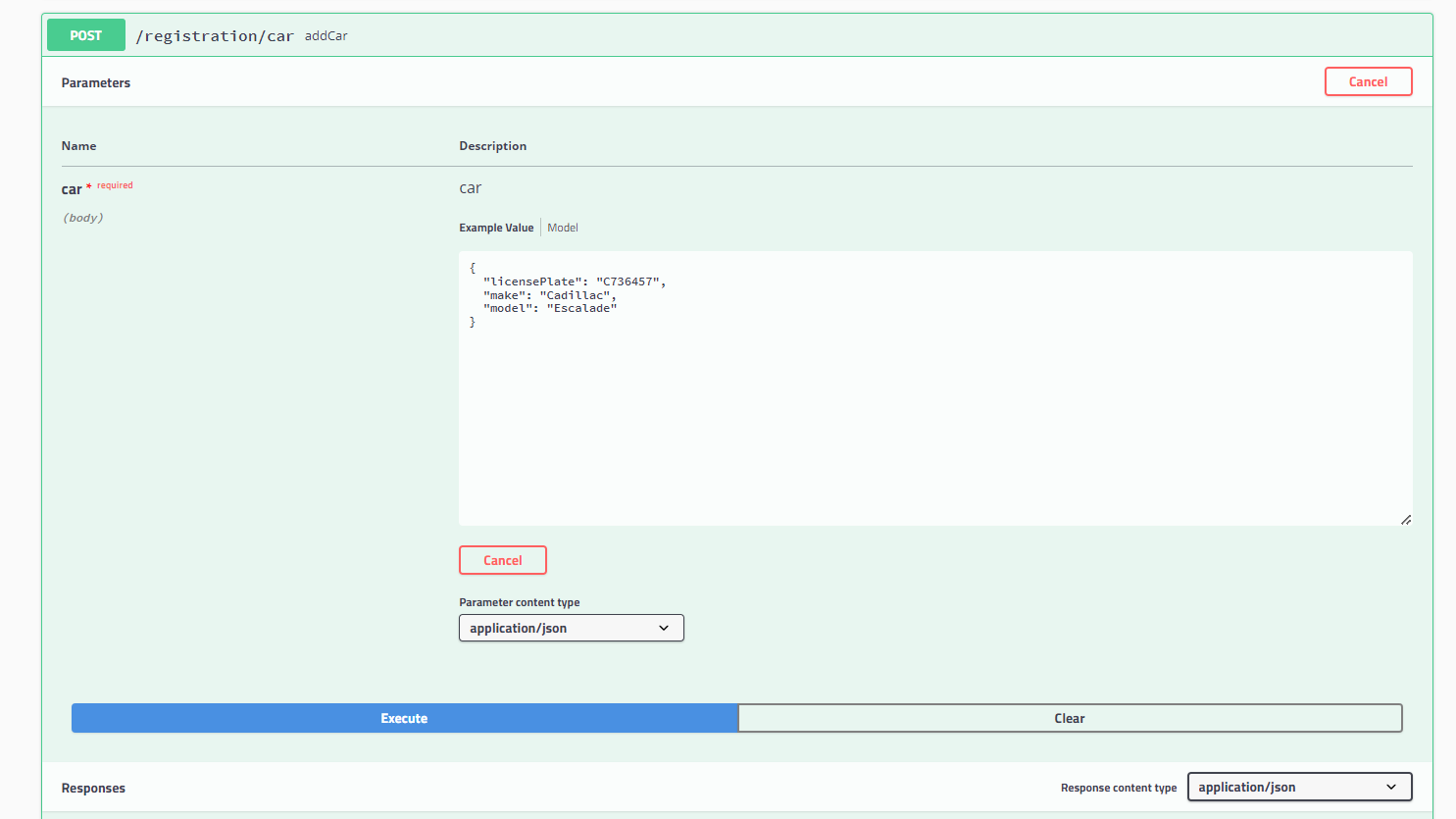
{

"licensePlate": "C736457",

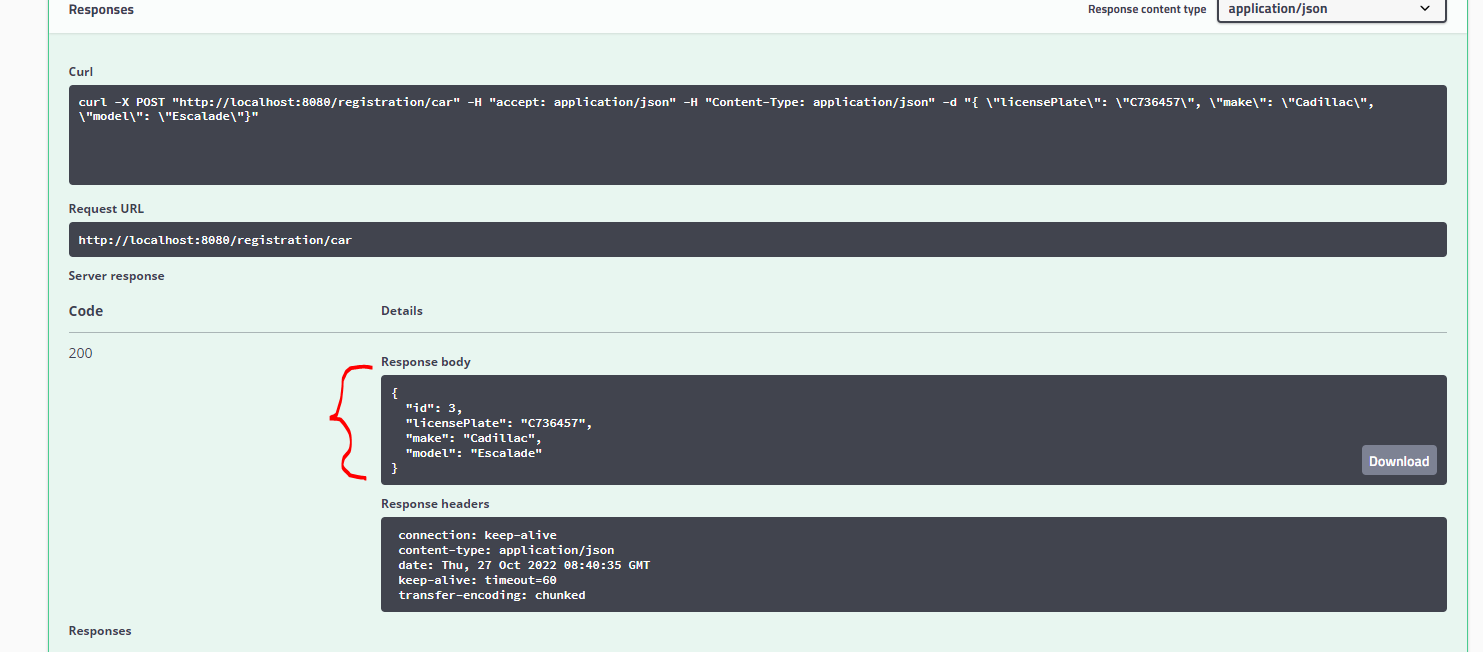
"make": "Cadillac",

"model": "Escalade"

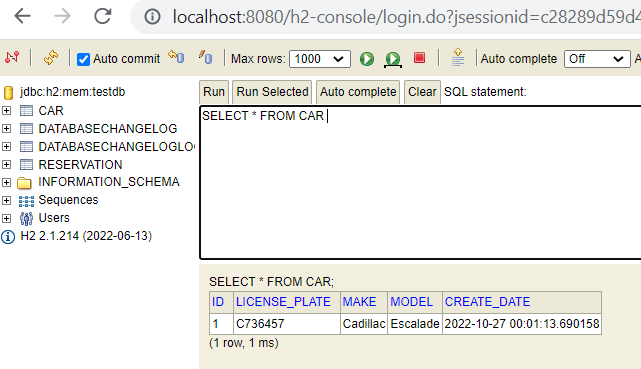
}



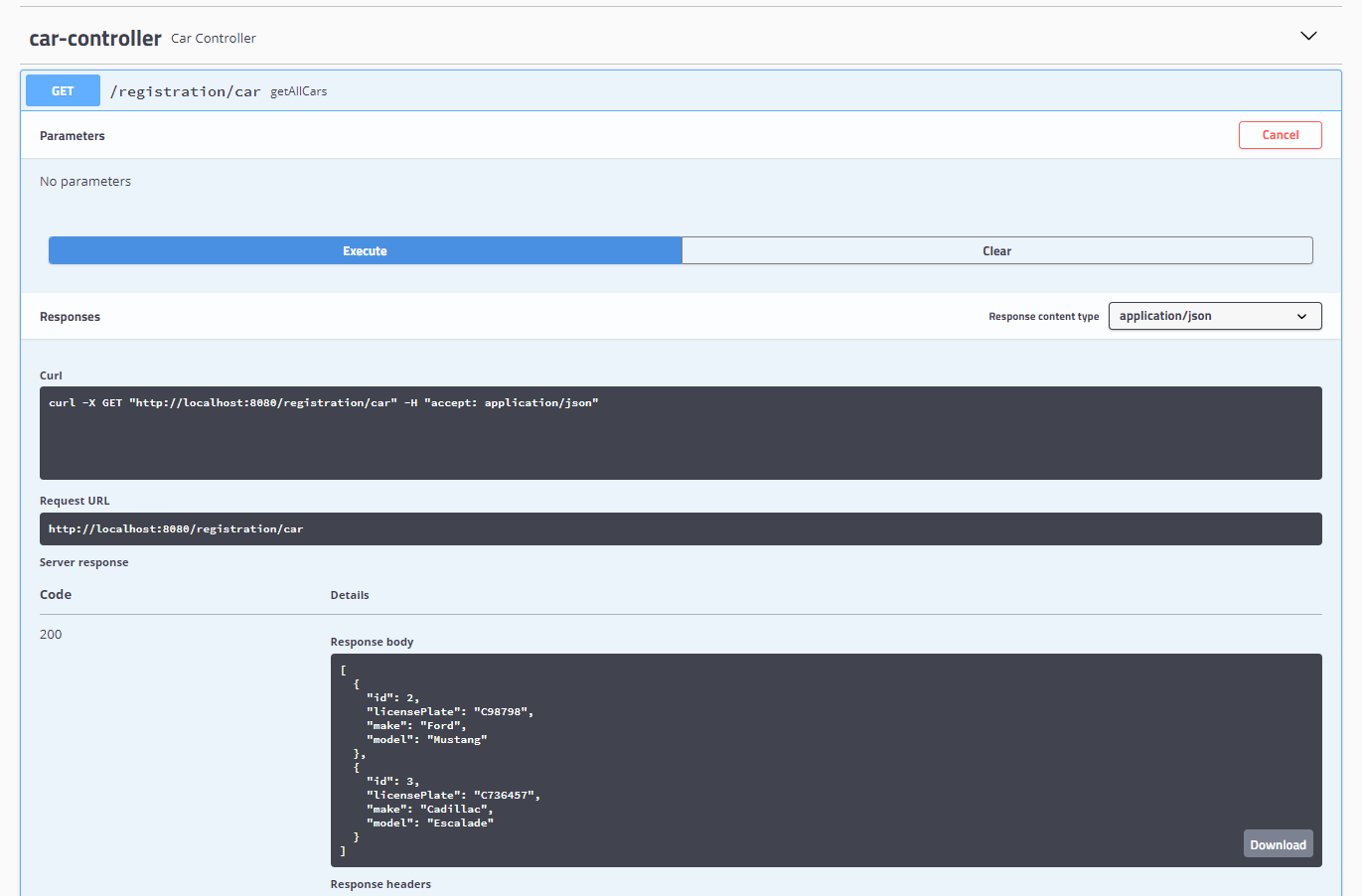
Response :



To check that just go to the database (in memory H2 database has been added)



1. GET [**/registration/car**](http://localhost:8080/swagger-ui.html#/operations/car-controller/addCarUsingPOST) - returns all the cars in the system



1. PUT [**/registration/car**](http://localhost:8080/swagger-ui.html#/operations/car-controller/updateCarUsingPUT)

Place the following json into value field (as an example):

{

"licensePlate": "C736458",

"make": "Cadillac",

"model": "Escalade"

}

So update is performed only for existing car with particular plate license. So to update current Cadillac Escalade we need to provide C736458 (**the license plate of already existing car in the system**) as a request parameter and car object which it should be replaced with:

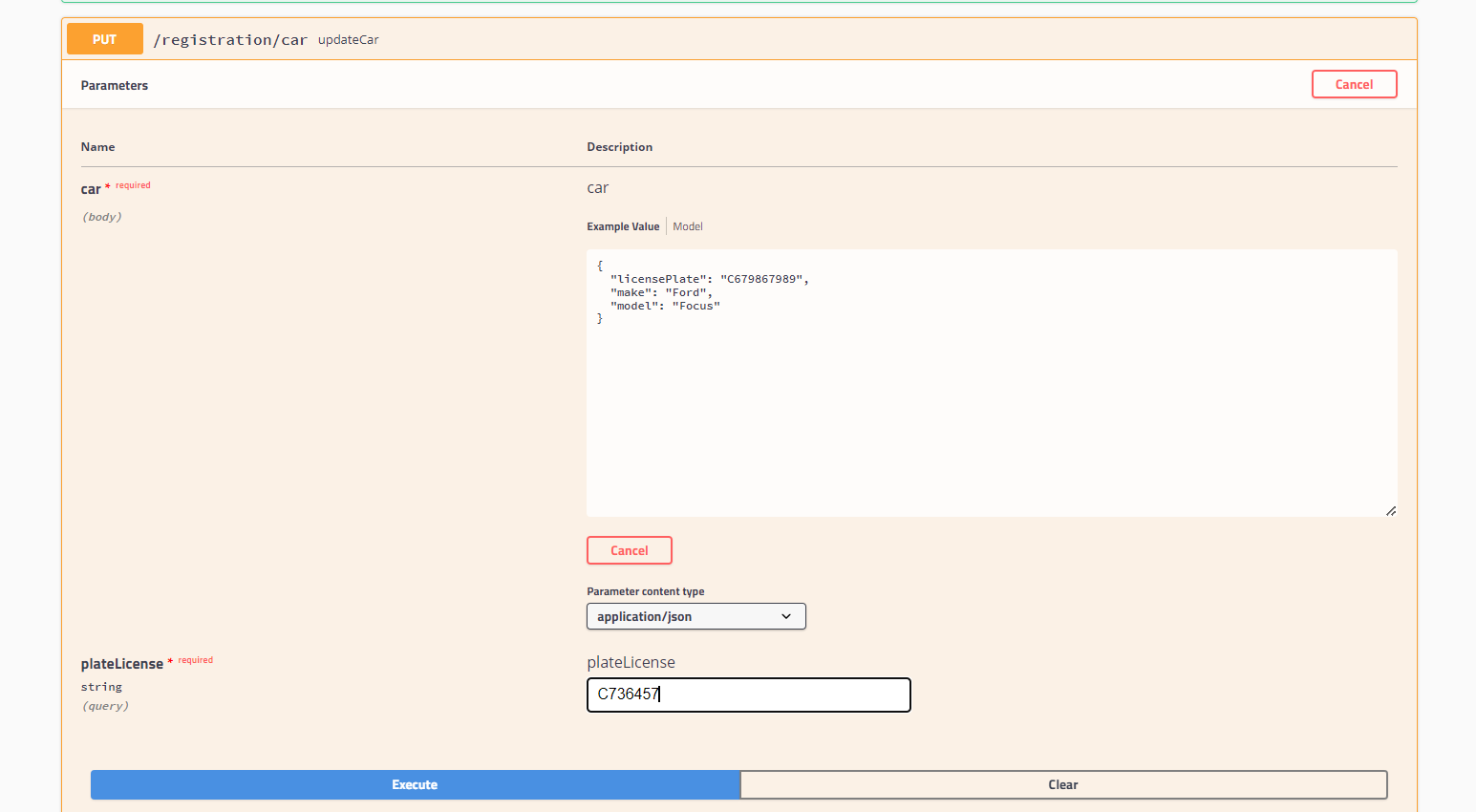
{

"licensePlate": "C679867989",

"make": "Ford",

"model": "Focus"

}



In case of not found car to be updated 500 Error will be returned with appropriate message.

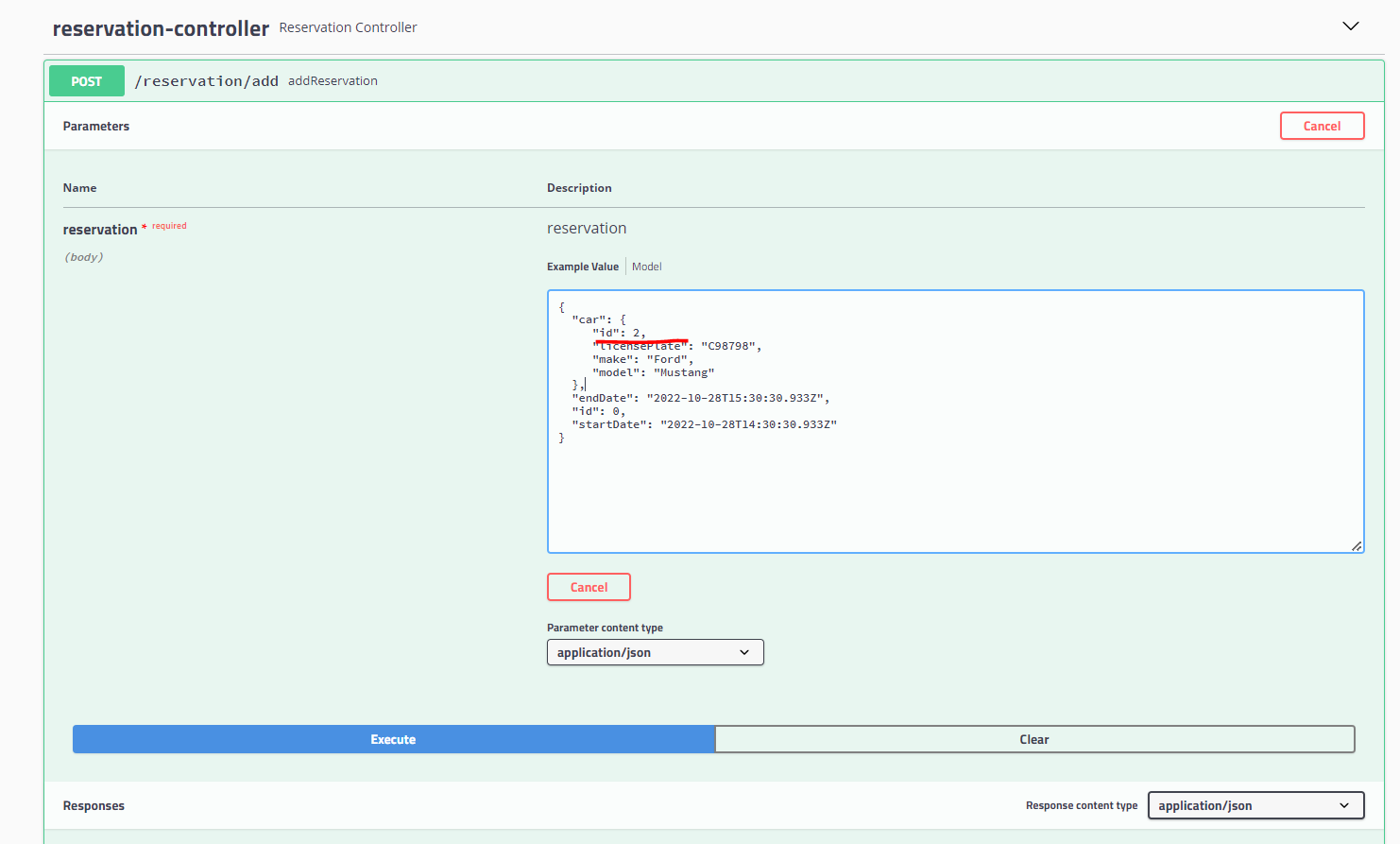
1. DELETE [**/registration/car/{licensePlate}**](http://localhost:8080/swagger-ui.html#/operations/car-controller/removeCarUsingDELETE)

Removes particular car by license plate value.

Also removes all the car reservations assigned to the car.

1. POST [**/reservation/add**](http://localhost:8080/swagger-ui.html#/operations/reservation-controller/addReservationUsingPOST) As it’s been mentioned above we cannot add the reservation before 24 hours from now and it can not be longer than 2 hours. Start date-time should be before end date time. No overlapping with other reservations (car and\or date ranges) – also validated and correspondent message is returned.

Obviously there should be particular car already registered in the system. That’s important, that we need to specify the exact db car id:

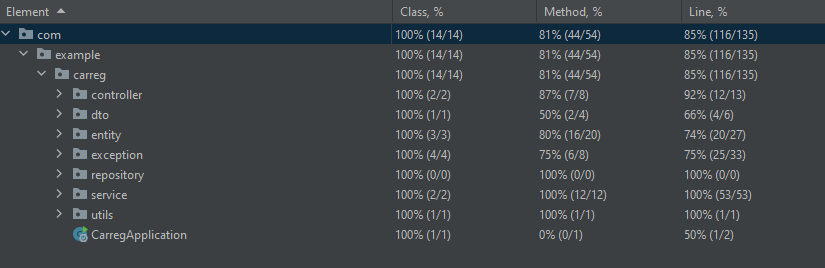




1. GET [**/reservation/all**](http://localhost:8080/swagger-ui.html#/operations/reservation-controller/addReservationUsingPOST). Gets all the reservations in the systems. So that means all the non-occupied timeslots are available for car reservations.



General features: H2 in-memory db and liquibase plugin for db migration have been added into the system. Test coverage is more than 80%:



Thanks a lot for attention😊)