

Software System Architectures

Final Project – 07 February 2023

**Supervised by :** RIACHI, Ralph

**Teams :**

Julie PONTHOU

Lihao WEI

Vincent TURLIER

Kouba Marie Jeanne Odile N'GUESSAN

**SUMMARY**

[**Thanks**](#_heading=h.cc94i217czc) **3**

[I. Project Diagrams](#_heading=h.2g5h14yzfu1k) 4

[Use Case](#_heading=h.nyg81fegm96z) 4

[Class Diagram](#_heading=h.i3122w15qqha) 5

[Architecture Diagram](#_heading=h.2mxzseuwgpfl) 6

[II. Source code implementing the different use cases](#_heading=h.rk900itqaayh) 7

[III. Installation Operations to run the application on a local PC](#_heading=h.1oovdr1o03f4) 11

[A. Startup](#_heading=h.kfh4uig2a9l3) 11

[B. Operation](#_heading=h.gjagu8ur97yl) 11

[C. Viewing](#_heading=h.51zqri4sgdga) 11

[IV. Swagger project to call the implemented services](#_heading=h.c757qcjr5sir) 12

[**Sources**](#_heading=h.ak0oz2nfcb1s) **14**

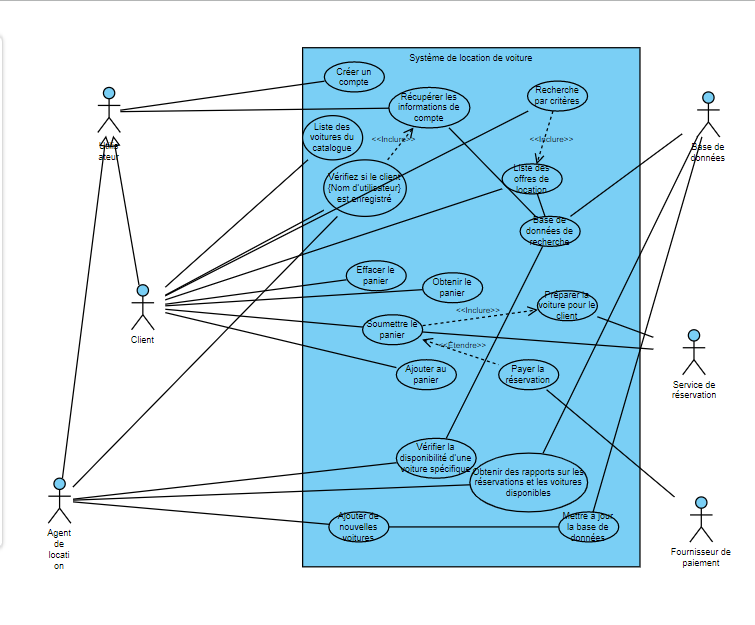
# Thanks

We thank Mr. Raph for his presence throughout the module, for his availability, and his explanations. We come out of this module with the desire to learn more about architecture.

## Project Diagrams

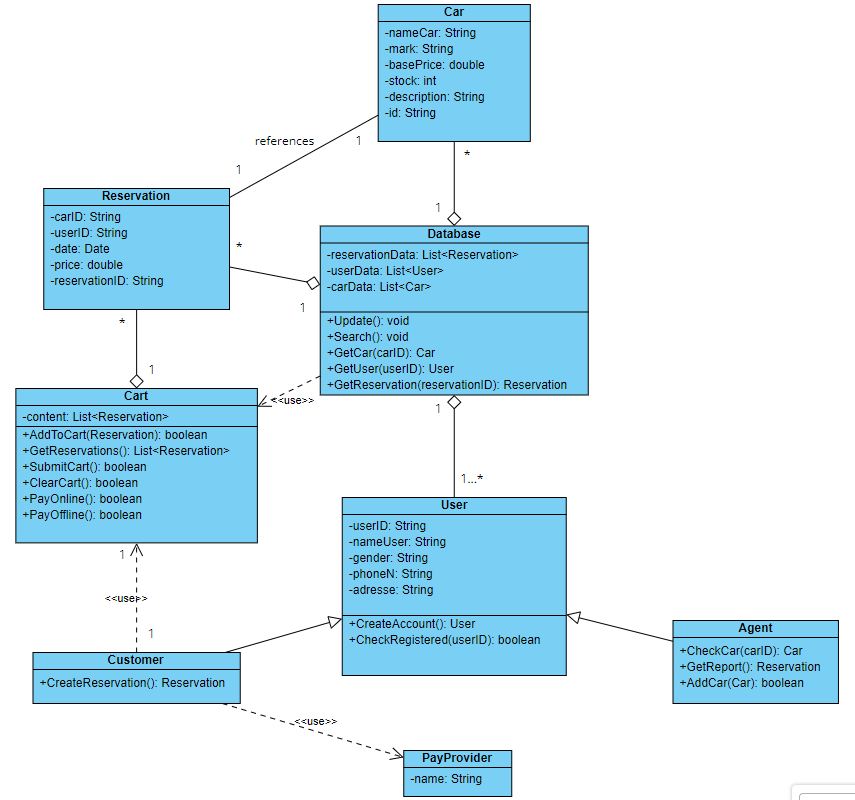
### Use Case

Two types of users: customers and car rental agents. Each user has an account in the system. The agent manages a car catalog. The customer can choose, rent, and pay for his rental. The payment can be done online or at the rental agency. Since the customer has to make his choices before renting a car, a cart management is needed before sending the order. When the payment is successful, a new reservation is created in the system for the car in the cart. A confirmation Email is to be sent to the customer to confirm the reservation. An invoice is included in the confirmation email.

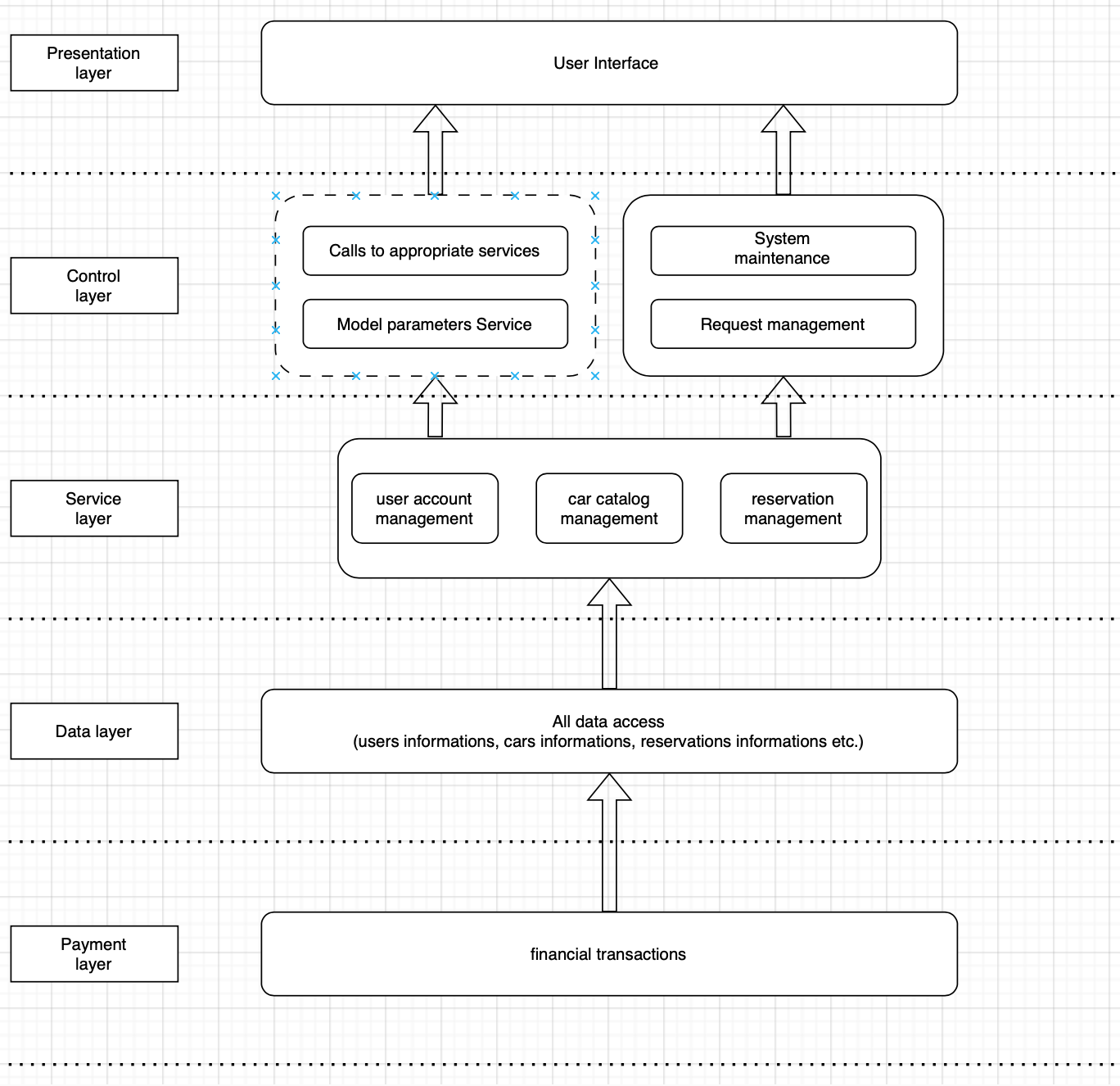


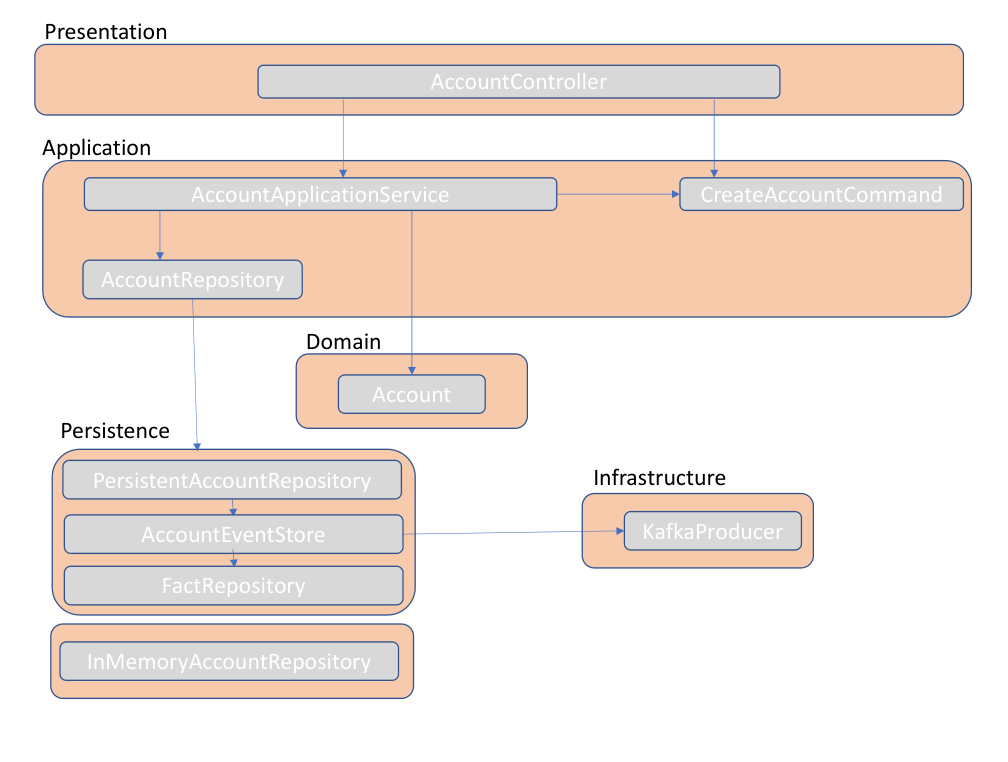
### Class Diagram

The class diagram created based on the use case further clarifies the dependencies between classes and depicts the functions and attributes required in the corresponding entity classes, facilitating the implementation of the project according to the required system architecture.

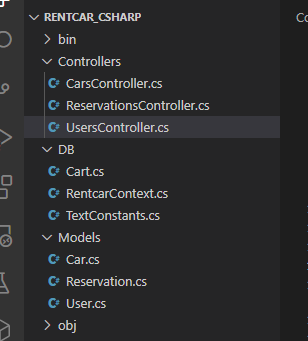


### Architecture Diagram

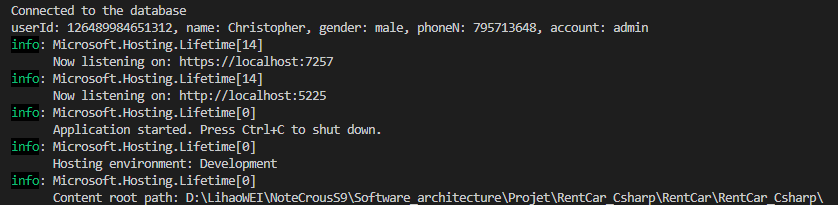




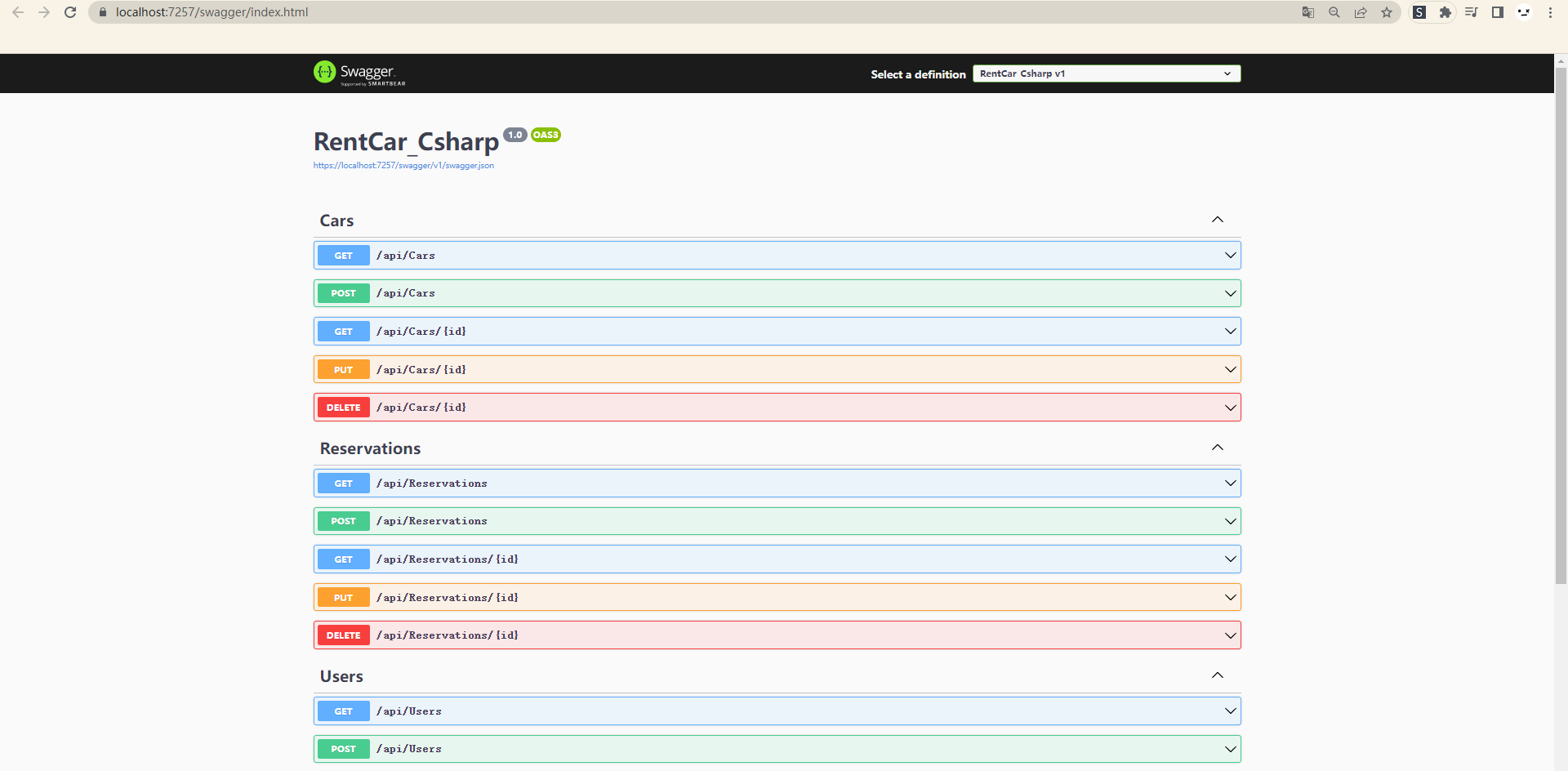
## Source code implementing the different use cases

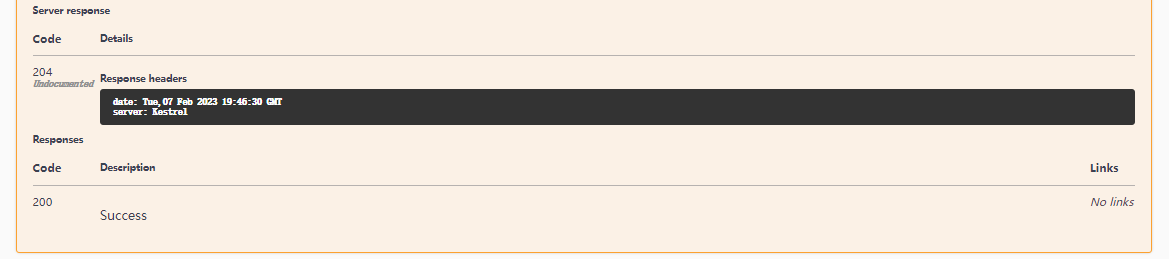


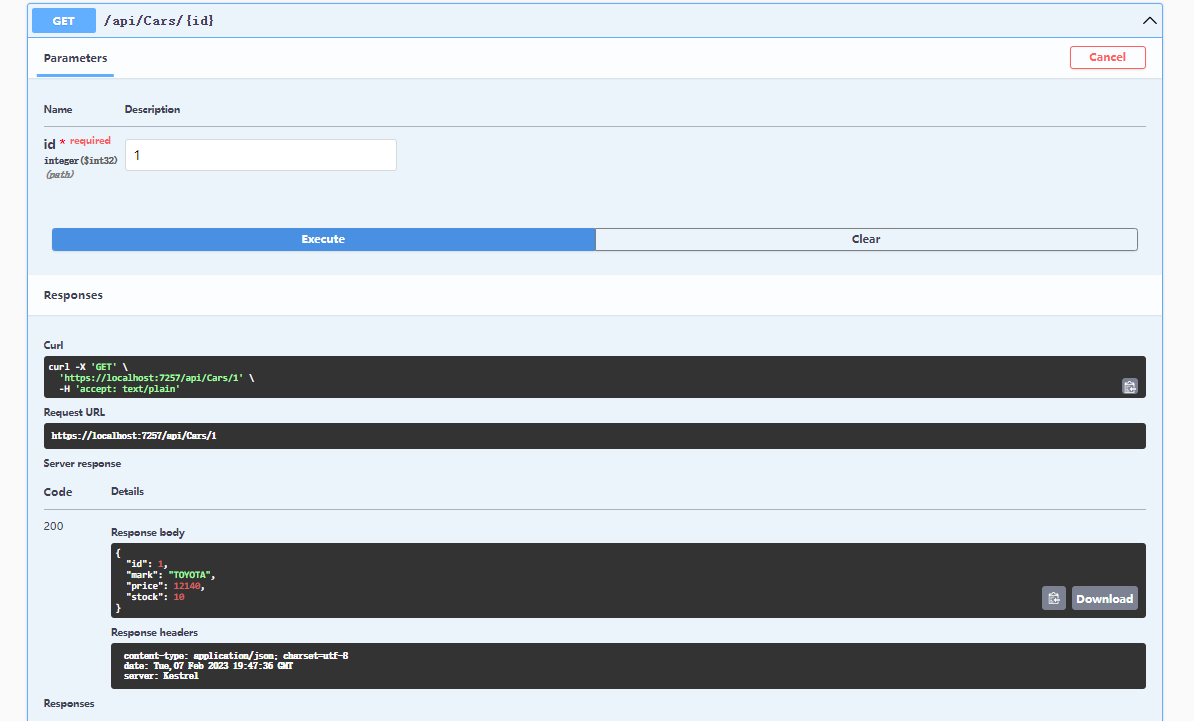
Based on the architecture diagram mentioned above, we divide the entity classes into three folders, DB, Models and Controllers, according to the differences in functionality. This is based on the MVC domain model with some modifications, and the data that used to belong to the Models is separated into a separate folder.



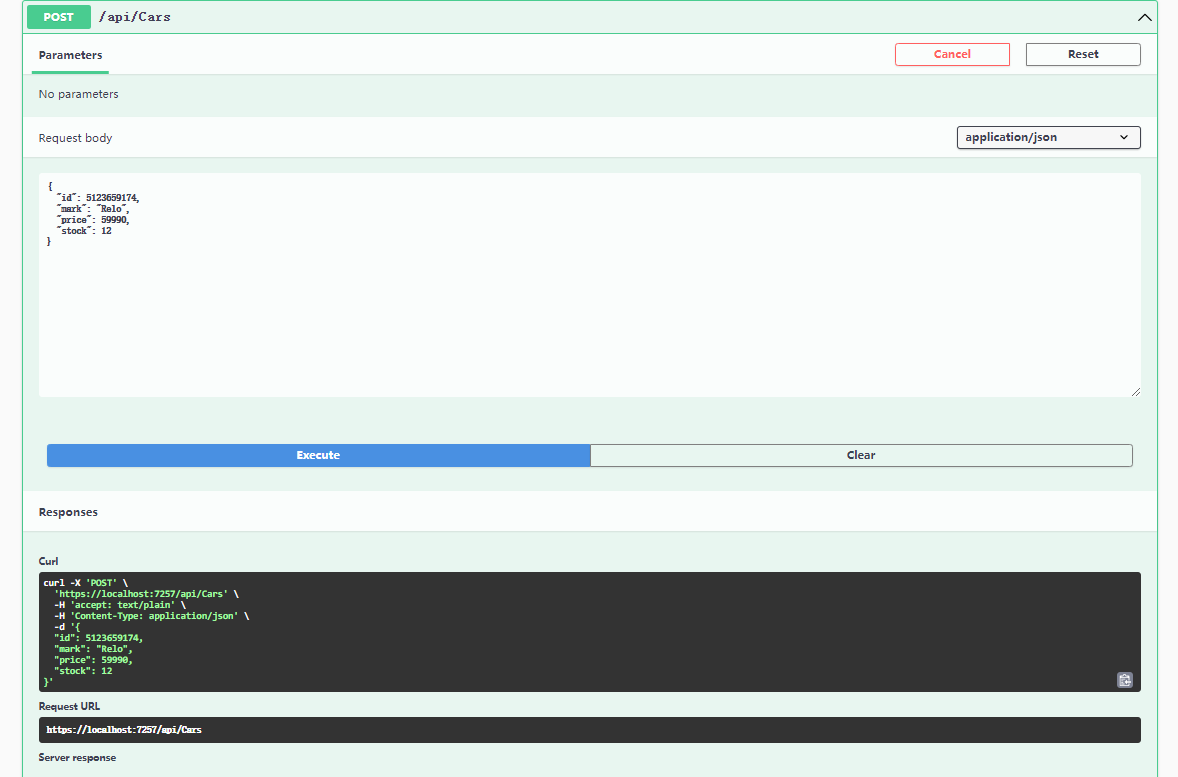
Connection to the Database in the command prompt

Localhost view of the API functions and accesses

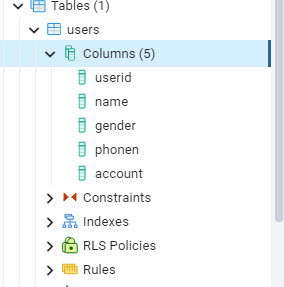
 Ticket of the Server response from an API access viewed with Swagger



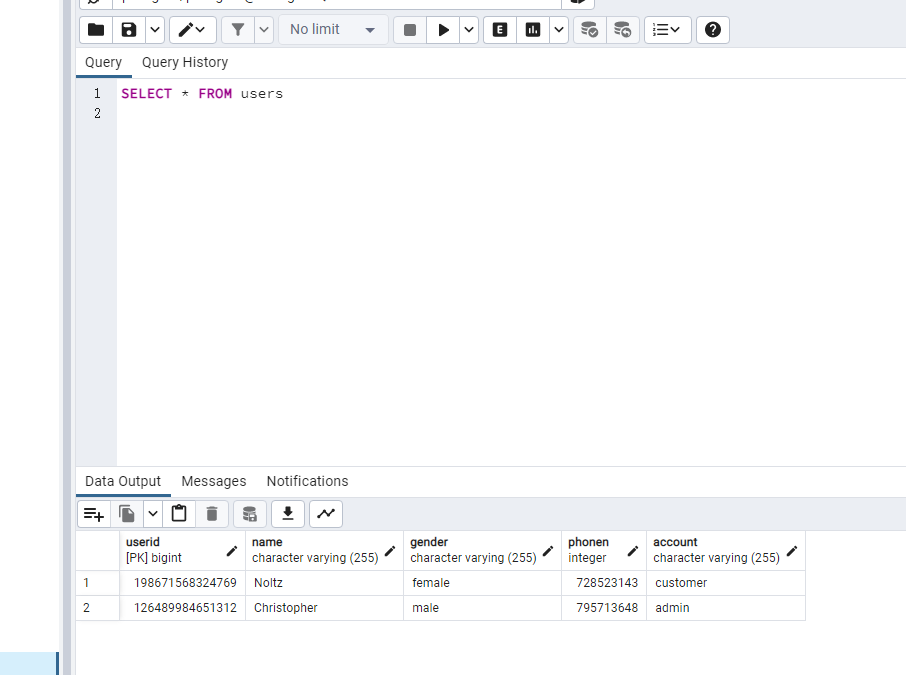
Example of a GET command to access data in the cars table



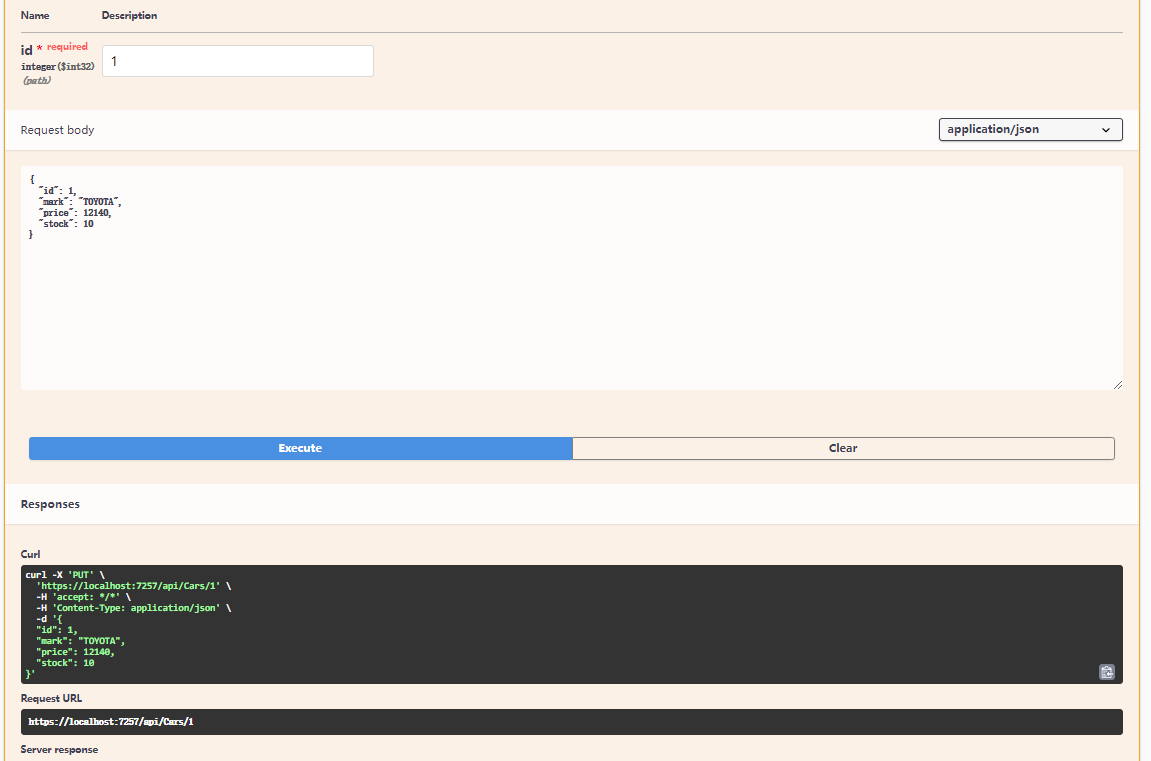
Example of a POST command to input data within the Cars table



Section of the database tables, showing Users columns



Display of the Users DB after a few actions within it



Example of a PUT command to input data within the Cars table

## Installation Operations to run the application on a local PC

### Startup

Install all required dependencies at once (usually only the following libraries need to be installed, links in annex)

* Net (version 6.0 and above)
* Install the Npgsql dependencies

Use command: **dotnet run --project "project root path"**

### Operation

For access the website: write <https://localhost:7257/swagger/index.html> in the address bar

Call and view the API interface: Upon entering the Swagger homepage, you will see featured modules such as "USERS" and "CARS." Upon selecting a module, its contents will be displayed, including options for functions like "GET," "POST," and "DELETE." Select one, click "Try it out," and then "Execute." You can then view the corresponding formatted data from the API interface.

### Viewing

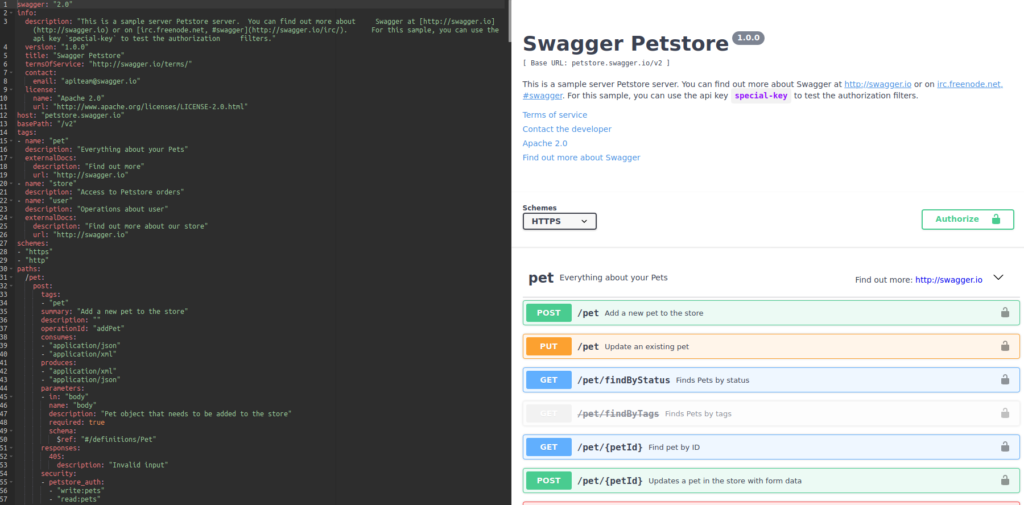
In the display, you will see:

* Request link: API address
* Status code: indicating the result of the returned status of the request (e.g. 200 for a successful request, 404 for not finding the corresponding webpage or data).
* Request body: where you can input data you want to add (this function is effective in POST, and must be written in the given format).
* Details: where you can view the detailed return results.

## IV. Swagger project to call the implemented services

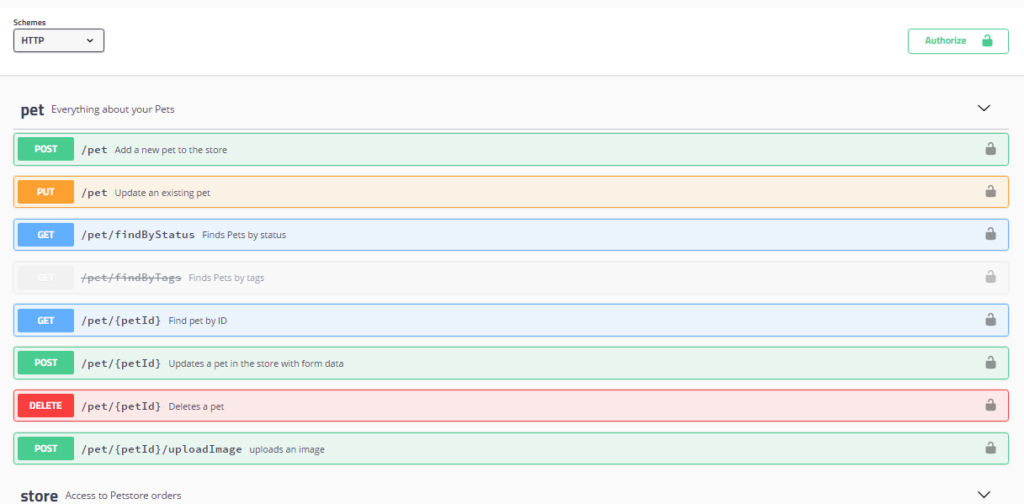
**Swagger** is an interface description language for describing APIs expressed using JSON. Swagger is used with a variety of open source software tools to design, create, document, and use web services.

Swagger Editor is a tool consisting of an editor and a viewer. The editor allows you to define your APIs in yaml format and the viewer offers an interactive view of what has been entered in the editor.

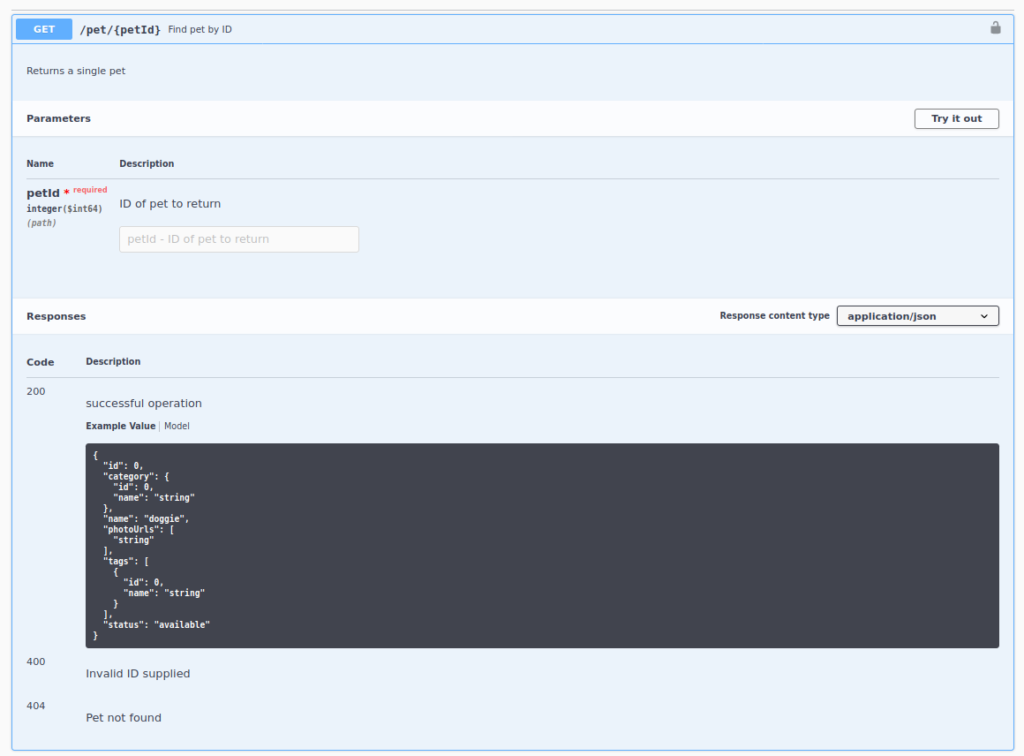


Swagger UI is the graphical interface for viewing and interacting with its APIs.

It thus offers the possibility to easily test them and make them more accessible for the customer.



*List of APIs*



# Sources

Swagger: the complete guide to the essential tool for web development (welovedevs.com) :

[Swagger le framework pour vos API - ITROOM](https://www2.itroom.fr/swagger-le-framework-pour-vos-api/)

Dotnet official download:

<https://dotnet.microsoft.com/en-us/download>

Npgsql package documentation and installation:

<https://www.nuget.org/packages/Npgsql/>