Marc Gutierrez 219122
Edgar Espinos 230024
Distributed Applications

Universitat Pompeu Fabra Barcelona

Project – Development of a distributed application

Design

The system was designed in the first instance to be able to locate all parts in the easiest manner possible. As such we have a class to receive all the calls from the page with the REST API but the functionality of those calls are on other classes that hold all the attributes and methods needed. This was the service end of the system, if we were to talk about the classes that store the data then we separated all the different information in different folders to be able to know the involved classes. And we try to stay as close as we can to a JSON compatible format. For the design of the web page we have decided to stay with a minimalistic page that is not over saturated in order to center our effort in the functionality.

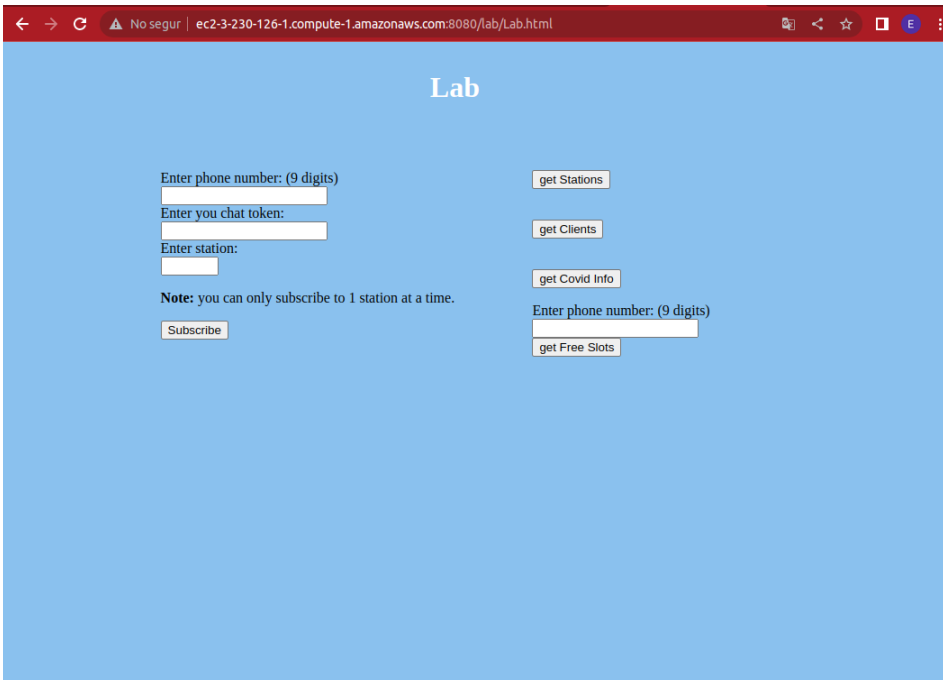


Figure 1: *interface view*

More useful tools that we have used in our design to specify that each phone number is a phone number is applying limiters to 9 digit numbers. If at the moment of the subscription there are more or less than nine digits in the textbox, the system will output this advice. Moreover, we have also taken into account the total number of Bicing stations in the system and you can not subscribe to a station that does not exist. The system will advise you like in here.

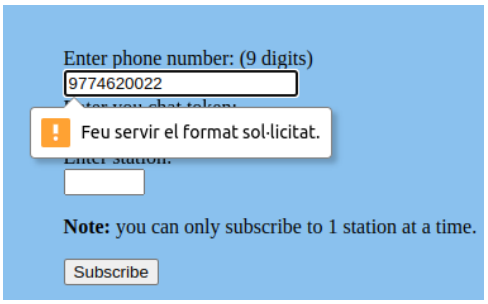


Figure 2: *phone number error format*

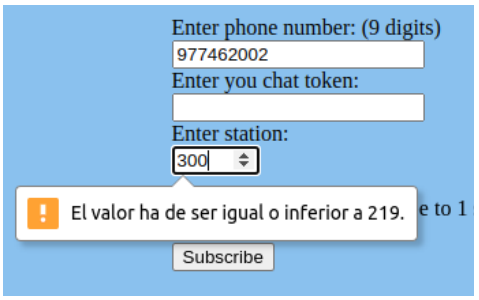
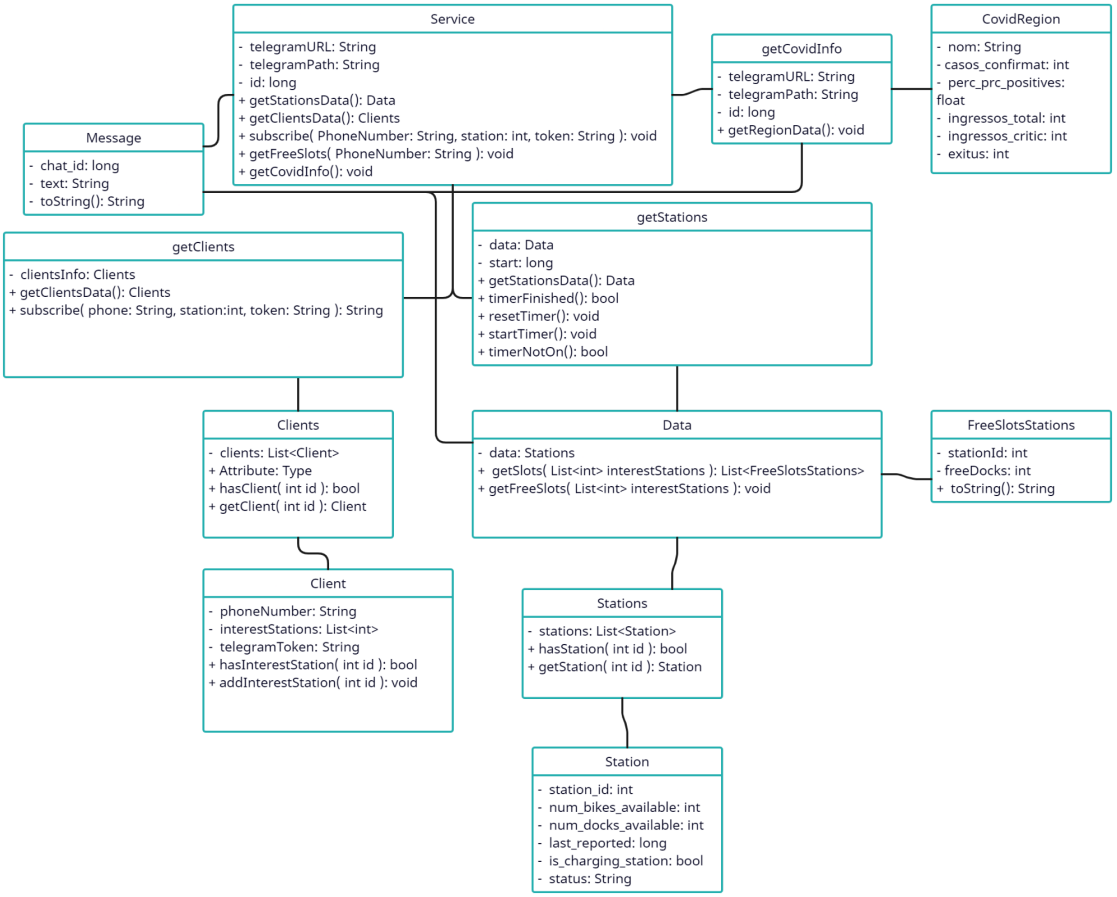


Figure 3: *station out of range*

Furthermore, if you are trying to get the slots available of a station with a phone number that is not subscribed in the system, you will receive a Telegram message that will tell you that such phone number is not available in the database.

Implementation



The constructors, setters and getters have not been included.

Document the REST API of BCV

We have implemented most of the page functionalities with the “GET” method that calls to a certain URL (i.e. /lab/at/Service/Get/Stations or /lab/at/Service/Subscribe to get a JSON with all the Bicing stations available) through an AJAX request. Furthermore, this request calls the required function, via REST API, and returns a JSON file that is shown within the same page without the need of recharging it. We can have this thanks by writing the file in a *div* where there is a scroll option so that the page does not either change in size nor format.

Execution

The application can be run by uploading it to a computer or server, capable of running Tomcat 9, as a *war* file. Once that is done then Tomcat 9, will automatically identify this file and decompress it into the correct form. After that, we will need to start Tomcat and we will be able to access the Tomcat main page from the running server. https://localhost:8080 if accessing to the page it is done from the same machine or simply adding :8080 to the Public DNS of our web service. From Tomcat's main page, we will need to follow this procedure: /warfile_name/webpage.html to be able to access the desired resource. In our case this should be: /lab/Lab.html. Once in the webpage there are forms and buttons that execute the different functionality of the application.

To subscribe a client to the Bicing service, you need to introduce your phone number (in the correct and usual form of 9 digits), your chat token and finally the station you want to be subscribed to. You can only subscribe to one station at a time. After that, you are able to check if the subscription is successful or not by clicking on the get Clients button, this will print on screen the list of users. Furthermore, if you want to know more about the Bicing stations, by clicking on the get Stations button you will see on screen the information of each station updated at the moment. Get Covid info sends the Covid info of Catalunya to your Telegram chat. Finally, if you want to check if there are empty slots available in the stations that you are subscribed to, by entering your phone number and clicking on the get Free Slots button you will receive a Telegram message with the desired info. If the phone number is wrong, you will receive a Telegram message advising you.

Conclusions

We are convinced with our project, the interface is simple and easy to use, with orientative information if you make some mistake introducing the numbers. Moreover, we found it difficult in the initial demo, but we finally found how to print on screen the

info and values that we wanted to, and now we have the useful scroll bar which allows us to have a compact and useful screen.
We also had some difficulties implementing the Telegram messages, and we also wanted to add more features to send clearer messages and also be able to send messages to each chat and not all to the bot chat. There are always things to improve.

To conclude, we have implemented a functional web page that is able to store information, process it and send it, we are satisfied with the work done.