Clients Web Service - Multicert Development Challenge

This project was created for the take-home challenge issued by v. It is an implementation of a service meant to manage Clients. It is composed of two folder components:

- multicertDevelopmentChallenge, which is the folder that contains the web service and files for database deployment.
- client, which contains the client which allows interaction with the web service.

Installation And Running

Further down in this README there are better explanations on the nature and structure of this project, but as installation and running might be referenced more, I've decided to have it be the first section. In order to install this project it's only necessary Docker and Maven.

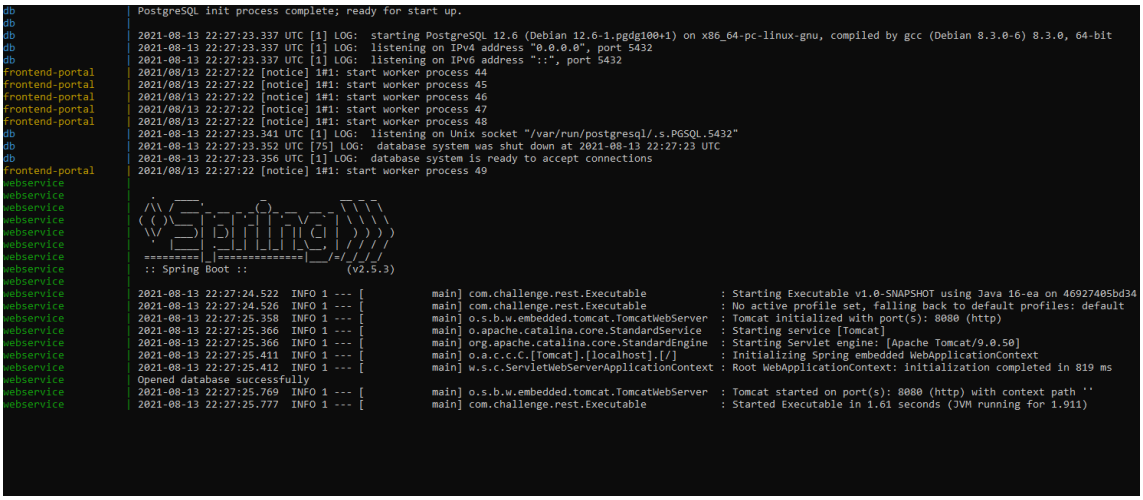
In the zip sent by email, a target folder containing a compile .jar file for the web service is already created. However, to build it simply open a terminal and type:

- mvn clean package

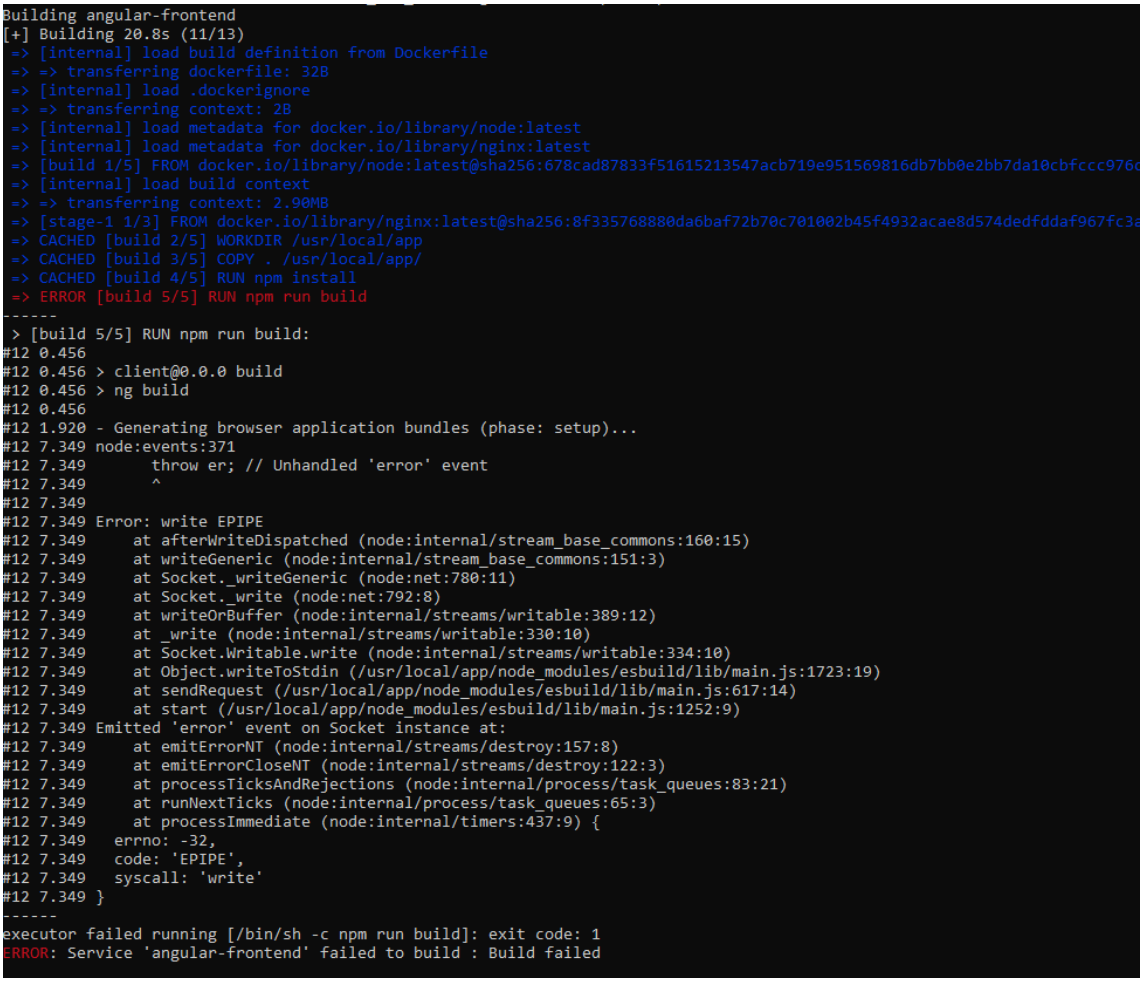
To run the application, open a terminal in the folder containing the client folder and the multicertDevelopmentChallenge folder and type:

- docker-compose up

If the application runs correctly you'll see the following in the terminal:



In testing, one of the pcs used for testing showed the following error:



If this occurs (or a similar error at the same step mentioning CSS Minimizer) please remove the line "RUN npm run build" from the dockerfile in the "client" folder, and type docker-compose up again.

Database

The database was developed using PostgreSQL, and is composed of one tables

- client - A table that contains the clients registered. It is composed of the columns "client_name", which contains the varchar with the name of the client, "client_nif" and "phone_number" which are var chars with a maximum of 9 characters containing the NIF and phone number of the client (the NIF must be at least 9 characters and so is the phone number if it's a portuguese phone number, which is assumed if they have a NIF), and finally "client_address", which contains a varchar containing the address of the client.

The creation of the database is denoted in the file "create.sql", it also populates the database, and is meant to be initiated through Docker.

The database contains the following constraints, meant to maintain the consistency of the data.

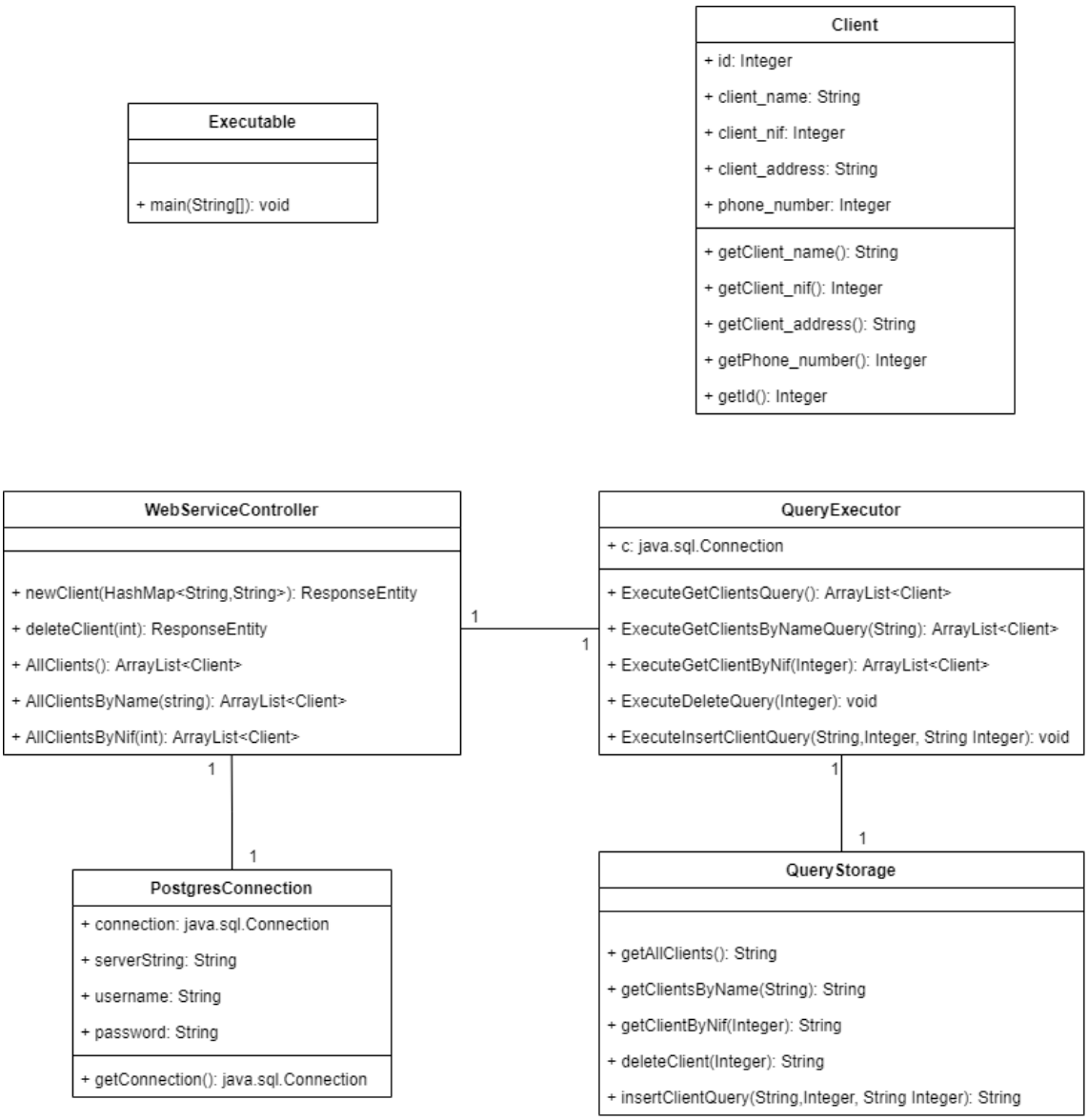
- check_nif_length and check_phone_length, these constraints are meant to make sure both the NIF and the phone number have 9 digits.

Web Service

The Web Service was developed through the use of Java and Spring. It contains a REST API with endpoints so that the Client can perform operations without directly being linked to the database. It contains the following packages in src/main/java/com/challenge:

- models, meant to store models to create json. There is a singular model:
    - Client, meant to model clients
- postgres, which contains classes necessary to connect and query a postgresql database.
- queries, which stores all necessary queries on the class QueryStorage.
- rest, which contains the web service executable class in Executable and the Controller for the Rest API in WebServiceController.

The classes and their interactions can be viewed in the following Class UML Diagram:



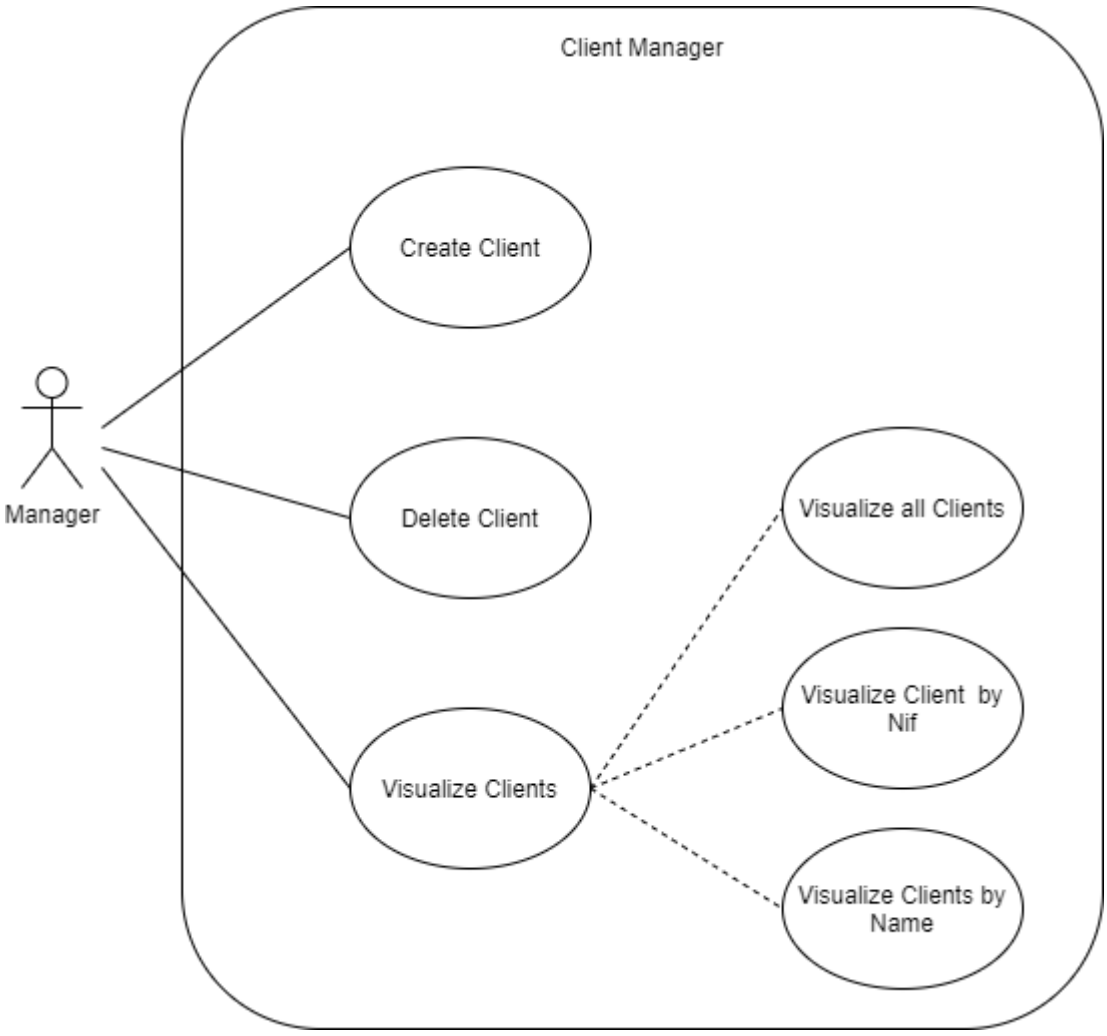
Client

The Client was developed using HTML, CSS, Angular, Bootstrap and Jquery.

It is composed of the following Angular Components, which all load the navbar component, which contains a navigation bar to go between them:

- home-page - The starting page of the client
- create-clienrt - Which allows a user to create a client
- client-viewer - This components allows a user to view all client registers, or delete a specific client
- client-search - This component allows a user to search for a client register by name or NIF, and delete them if they wish

The client allows for all use cases which can be viewed in the following diagram:



Testing

Unit Testing was developed for the database interactions and querying through Junit 4 and Mockito, the tests are stored and executed in the QueryExecutorTest class.

-- Nuno Freitas