Iteration 2

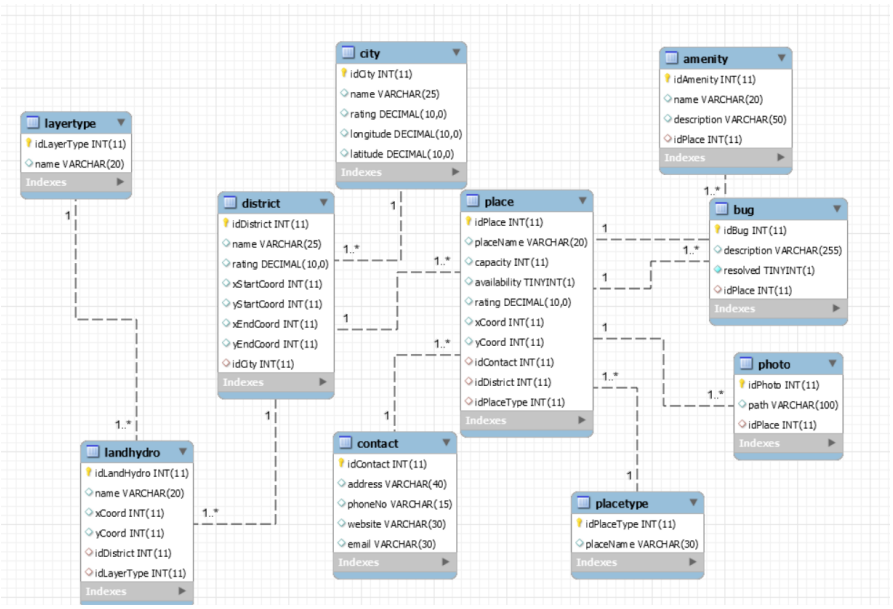
For this iteration I chose to implement the Client-Server architecture using **Spring Remoting**. For this I used two config files, one for the client and the other for the server. For the server config I had to provide the Service with its constructor and the Services interface which the Service implements and also the port for listening when clients connect.

The Services interface has all the methods that the client sends to the server, which the Service class implements. The CityObserver interface has the eventOccured method which is used in the Controller classes to "tell" the other clients that something happened. For example, I used this on the BuilderController, so that when there are at least two builders connected, when one of them builds something new, the other builder sees this new building on the interface, updated, without doing anything.

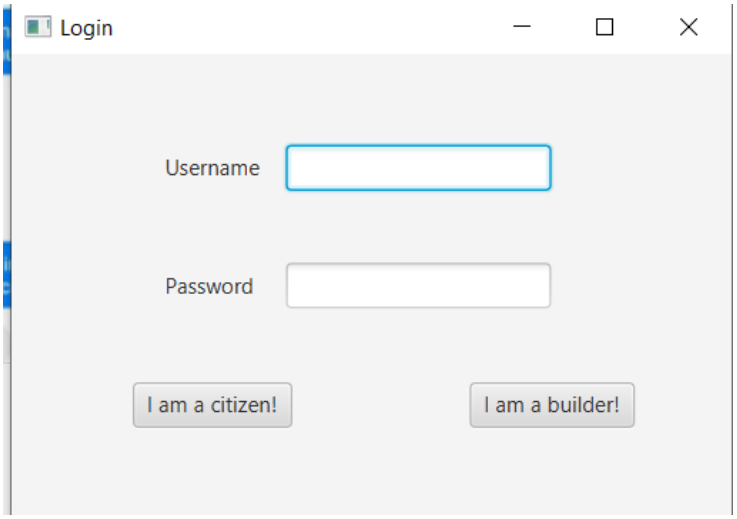
This is done by having a map of logged clients. Basically, when the login is being checked, if it is validated, the new user is put in a map, regardless of which type of user it is (hence the CityObserver parameter, which could be all types of controllers). After building a new place, the notifyBuilders() method is called. This creates a thread pool for notifying when a building has been built. Each thread takes a client (which is represented by the CityObserver) and notifies each of them using the eventOccured method in the controller classes.

First, I need to start the server for listening to the port chosen in the config. After it is started, the clients can connect to the server with success.

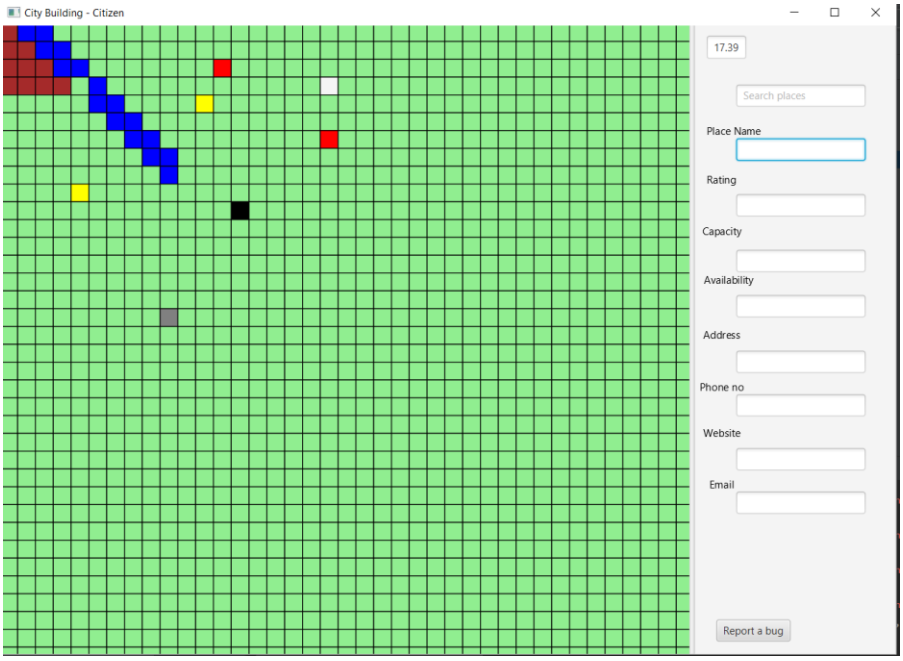
The database has also been modified, adding the Bug entity and changing the availability from VARCHAR to boolean(TINYINT):



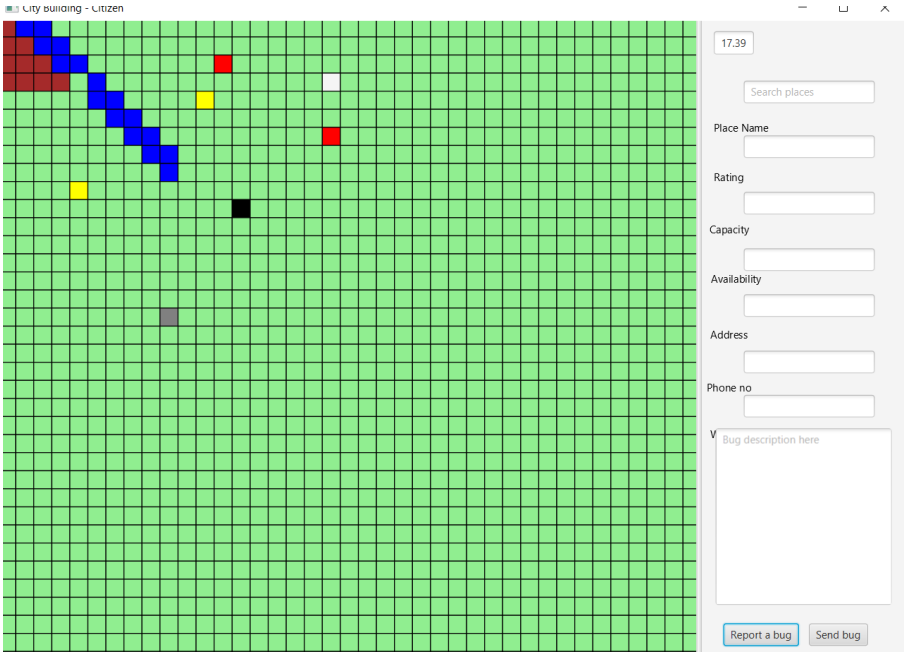
For the interface, I changed the login to allow the citizens to join without credentials.



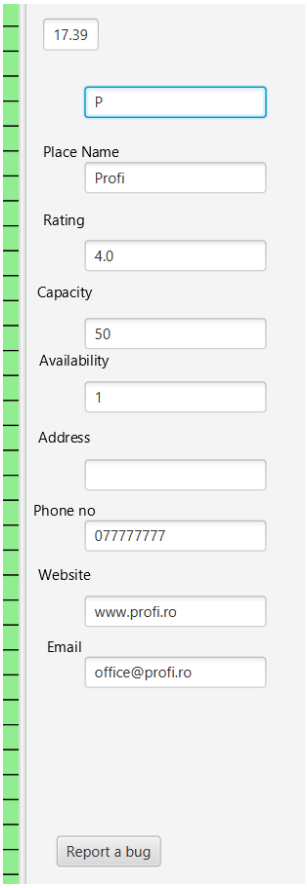
If the client clicks on I am a citizen, this will redirect the citizen to the interface, where reporting a bug is allowed and the temperature can be seen in the city(Ludus) in the top right part of the interface (when the ss was done there were 17.39 degrees Celsius).



If the citizen clicks on report a bug and on a place on the map, this will popup and let the citizen send the bug:



Another feature added is the search feature. It will automatically give you a suggestion of what you were looking for, even if there is only one letter written as it can be seen below by only writing “P”:



The builder interface remains the same as the last iteration. The admin (city council member) can now see all the bugs in the city by clicking on the See Bugs button on the admin interface.

Use cases

There are now three types of actors: **builders, admin** and **citizens**.

For the **builders** and **admin** the first step is to login. If the user is a **builder**, then he can see the map with all the existing places and nature elements. The builder can then add a new place, by abiding to a set of rules (validator), remove a place if it's abandoned or update an existing place. This is done by typing all the necessary information in the right side of the interface in the text boxes then clicking on a button. If the **admin** is logged on, he can add a new rule for building, which all the builders must abide to and also he can see all the bugs in the city.

The **citizens** are not required to log in, they only need to click the **I am a citizen** button. They can see the map and click on a place to report a bug.