Documentation for Register and Login: Java and Database.

**DATABASE PART:**

Before touching any program, I opened Astah and idealized the new table of the database and how it relates to what we already have.

Diagram

Description automatically generated

Looking at it now, I realize that I forgot to do anything related to the User possibly being a Business. Like, the isBusiness column of the database is null because I forgot about it. I also didn’t do anything regarding the opening hours, it is just an idea. Basically, I just ignored businesses for now. But there is a column in the database for “isBusiness” that should be expanded upon later.

So I did the SQL:

Text

Description automatically generated

The table has to be called “user\_” because user is a reserved keyword. “password” is also a reserved keyword, so the password column had to be called “password\_”.

The first thing I did was define the interface for the UserService. Given that my task was to make Register and Login, those were the methods defined. Login can throw an exception in case the credentials are invalid.

Text

Description automatically generated

The implementation for UserService needs an instance of UserRepository, so that’s what I did after:

Text

Description automatically generated with medium confidence

It just extends CrudRepository and does not need any special methods.

So the UserService Implementation will need a UserRepository and an AddressRepository. It should also be annotated as a Service.

Graphical user interface, website

Description automatically generated

The methods are as follows:

Text

Description automatically generated

Registering a User is very straightforward, just saves him in the database. It is here where some validation could take place, but I did no such thing.

By the way, the User class also had to be created and properly annotated as a JPA Entity. This is how it looks:

Text

Description automatically generated

The Address column is ManyToOne because many users can live in the same address. However, I think there might be a problem where the Repository saves the address as a new one even if two users register the same fields, because the AddressId is a serial integer. Maybe we should do something to prevent this in the future. But it is not a problem for now.

As for the login method, it just checks if the Username and Password provided by the DTO match those present in the Database. Otherwise, it throws an Exception depending on what was the problem.

Text

Description automatically generated

The UserCreationDTO looks like this. I copied it from Kamil’s C#:

Text

Description automatically generatedAddressCreationDTO is what you would expect, so I will not show it.

UserLoginDTO contains the username and password, also as expected:

Text

Description automatically generated

Let’s move on to the protofile.

The UserCreationRequest message has the same fields as the UserCreationDTO.

Text

Description automatically generated

Same goes for UserLoginRequest, which as the same fields as UserLoginDTO.

Text

Description automatically generated

The “services” provided in this protofile are register and login. They take the corresponding requests and both return a UserMessage. This is a UserMessage:

Text

Description automatically generated

AddressMessage just includes the Address fields. You can see that I forgot to include “isBusiness” anywhere until now lol. Problem for later.

And here are the services that I mentioned:

Text

Description automatically generated

After cleaning, installing in maven and fixing any errors with JPA, the GRPC classes were generated. This meant that I was missing implementing the base classes.

Here it is:

Text

Description automatically generated

This is what the register method looks like:Text

Description automatically generated

It’s very straightforward. However, you can see on line 34 that there is a message “getUserMessageFromUser”. This transforms a User java object into a UserMessage grpc object. It was implemented manually and I will show you what it looks like.

Here it is.

All it does is:

1. Get the Address from the User
2. Transform the Address java object into an AddressMessage grpc object
3. And then parse all of this into a UserMessage object and return it.

Text

Description automatically generated

The reason why I extracted this method is because it is also used in the login method:

Text

Description automatically generated

So we get the desired User object from the UserService both in login and register. The functionality of these methods was shown earlier.

That is all for the implementation. So our way to test them is by running the application and using BloomRPC!

This is what happens when we try to register a user:

Graphical user interface, text, application, chat or text message

Description automatically generated

The longitude and latitude are zero because we are note getting them from the API yet. But this is a good start!

And this is what happens when logging in: the sunny scenario, the wrong password and the non-existing user:

SUNNY SCENARIO:

Graphical user interface, text, application

Description automatically generated

When logging in with the right credentials, the User object is returned to us.

WRONG PASSWORD:

Graphical user interface

Description automatically generated with medium confidence

We get an illustrative error saying that the password is wrong. Nice!

NON-EXISTING USER:

A picture containing scatter chart

Description automatically generated

There is also an illustrative error saying that there is no such user. Again, nice!