Bogdan Stefania-Franciska

258/1

Recipes App

Project description

This project's purpose is mainly to show how microservices and micro-frontends work. It is an application for managing recipes, everyone being able to upload their own, as well as browsing through recipes posted by others.

Backend

As mentioned previously, the backend is based on a microservice architecture. It was implemented in NestJS using Message Pattern and it exposes REST services through an API Gateway, which is the communication point of the backend.

We have 4 microservices, besides the API Gateway there is one for user account handling, one for authentication related topics, and one for the recipes. The API Gateway communicates with all the others through TCP, but also there is communication between the other microservices too.

Some services are secured via JWT Bearer Token, meaning that at login, the user will get a token, which will be the key for him/her to be able to reach those services. However, this token expires after a specific period of time, therefore the user will have to log in again.

Databases

User and recipe details have to be stored, but for having a more secure way of storing the data, there are two PostgreSQL databased used.

Frontend

For the frontend to be highly decoupled, it was implemented using micro-frontend architecture. The chosen language for all of them is React. Regarding the logic, splitting them into micro-frontends was possible by using the so called Webpack Module Federation.

The project is formed by 4 micro-frontends, one acts as a container, it gathers all the other micro-frontend components, and makes possible the navigation, whereas the other three are handling only specific topics. The first one contains user account and authentication related components (Log in, Sign Up), the second illustrates recipes and corresponding operations, and the third one is responsible for showing details about the company, and ways of communicating feedback.

An important part of the frontend is that, for being able to see new recipes without refreshing the page, the backend sends notifications to it when a new recipe is added, using websocket.

Bogdan Stefania-Franciska

258/1

3rd party services

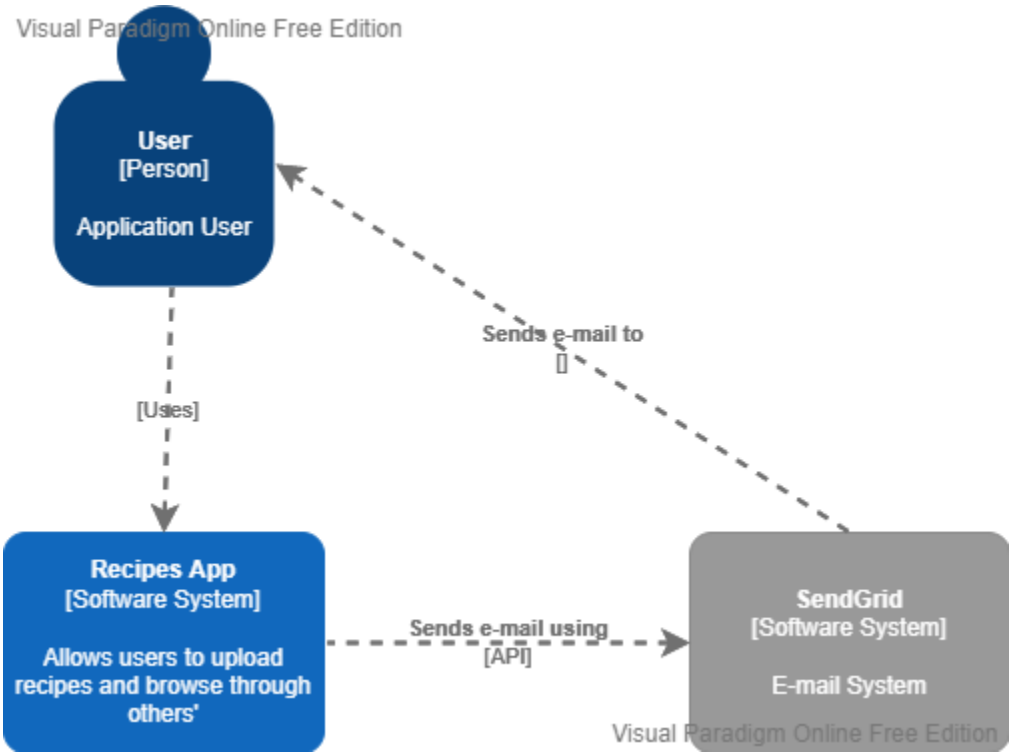
For sending e-mails easily and securely, SendGrid was integrated. For now, it is used only when a user signs up, and he/she gets a welcome e-mail.

Containers

For easier deployment as well as development, each microservice, database and micro-frontend has its own Docker container (and Dockerfile). By using docker-compose, the containers can communicate with each other in their specific networks.

Architecture based on C4 model

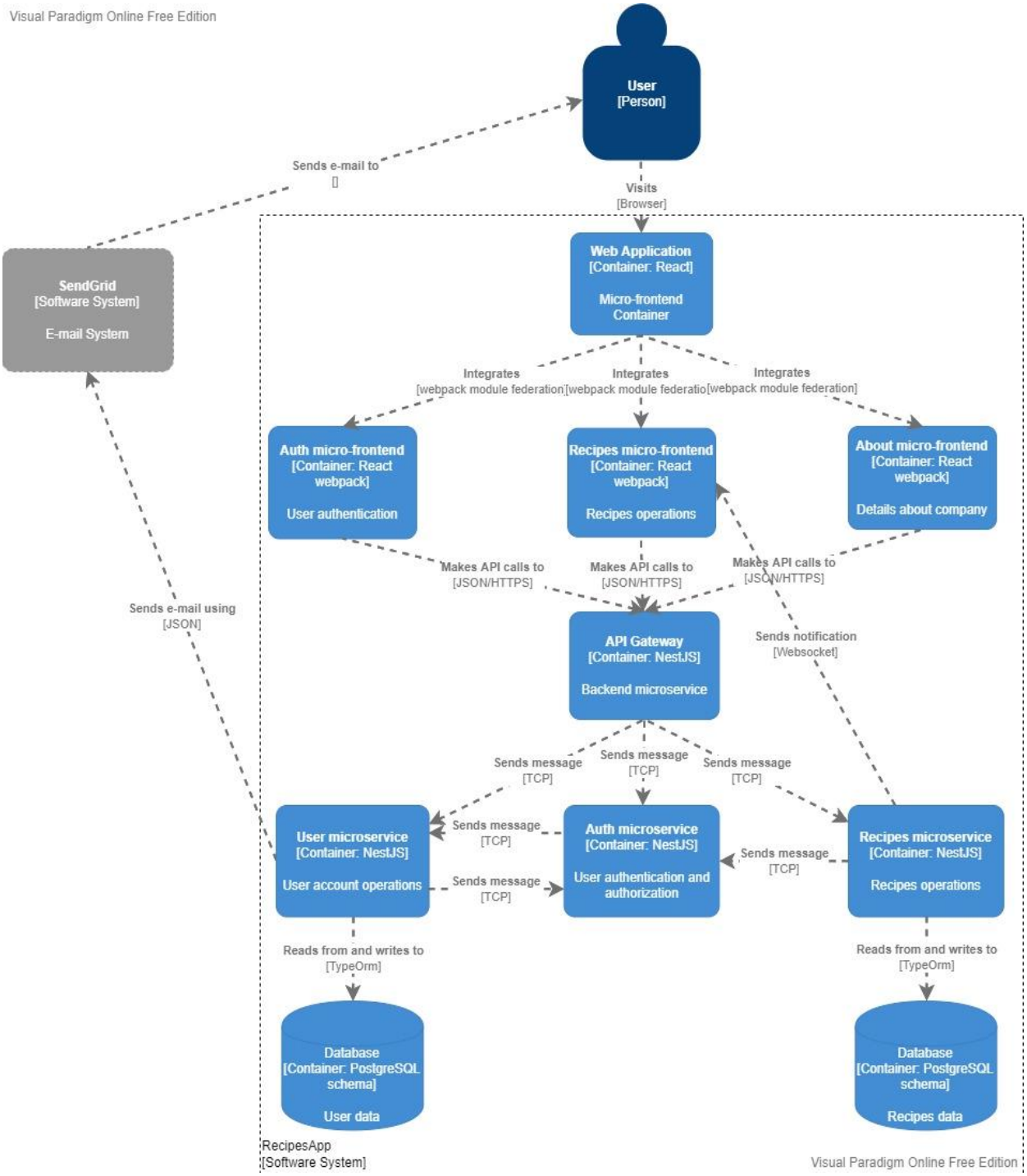
In the system context we can see that the person is connected to the external e-mai system too, besides our recipe app.



Next we can see the second level, the container diagram, where each microservice and micro-frontend is represented as a separate container:

Bogdan Stefania-Franciska

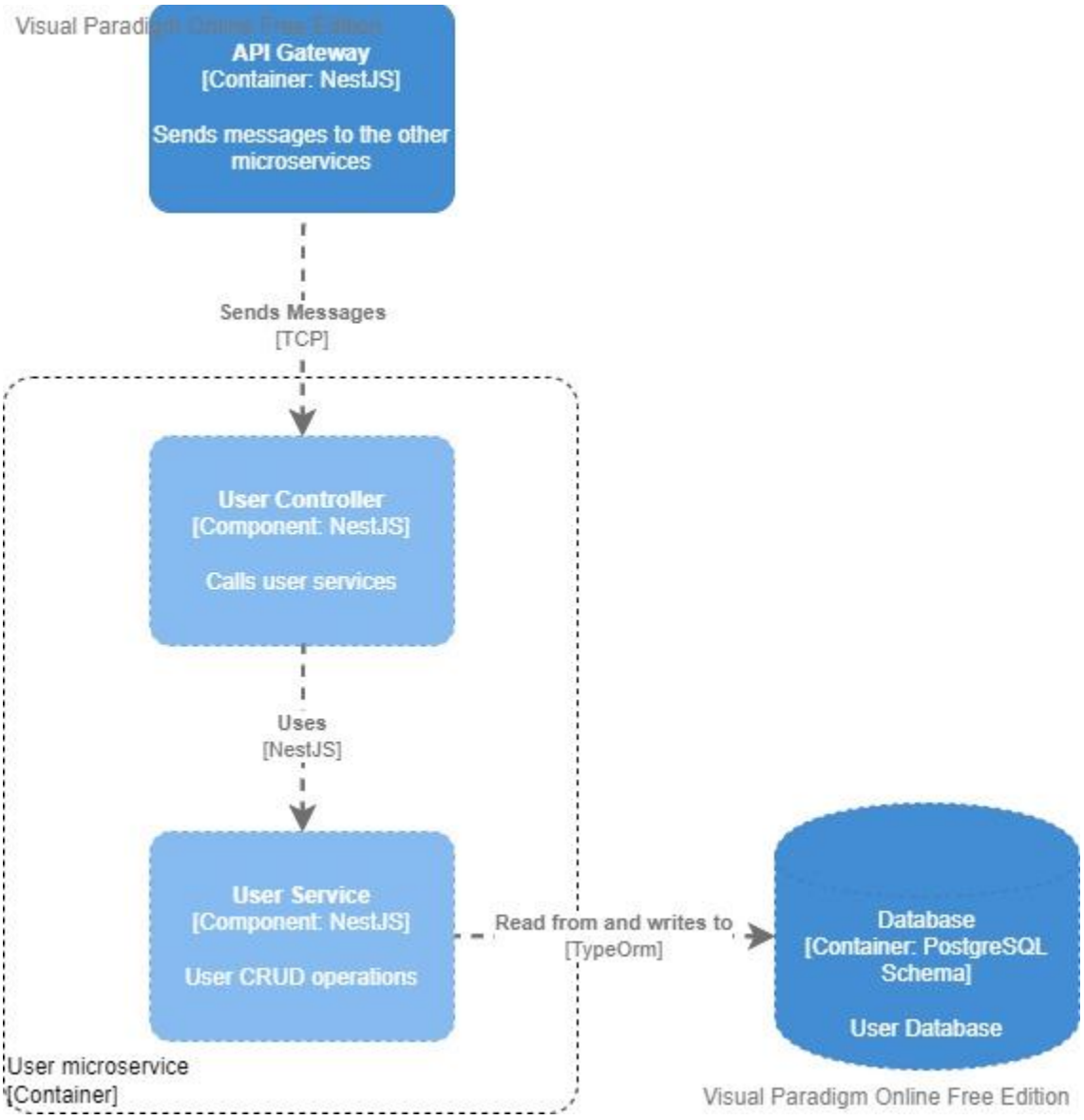
258/1



Bogdan Stefania-Franciska

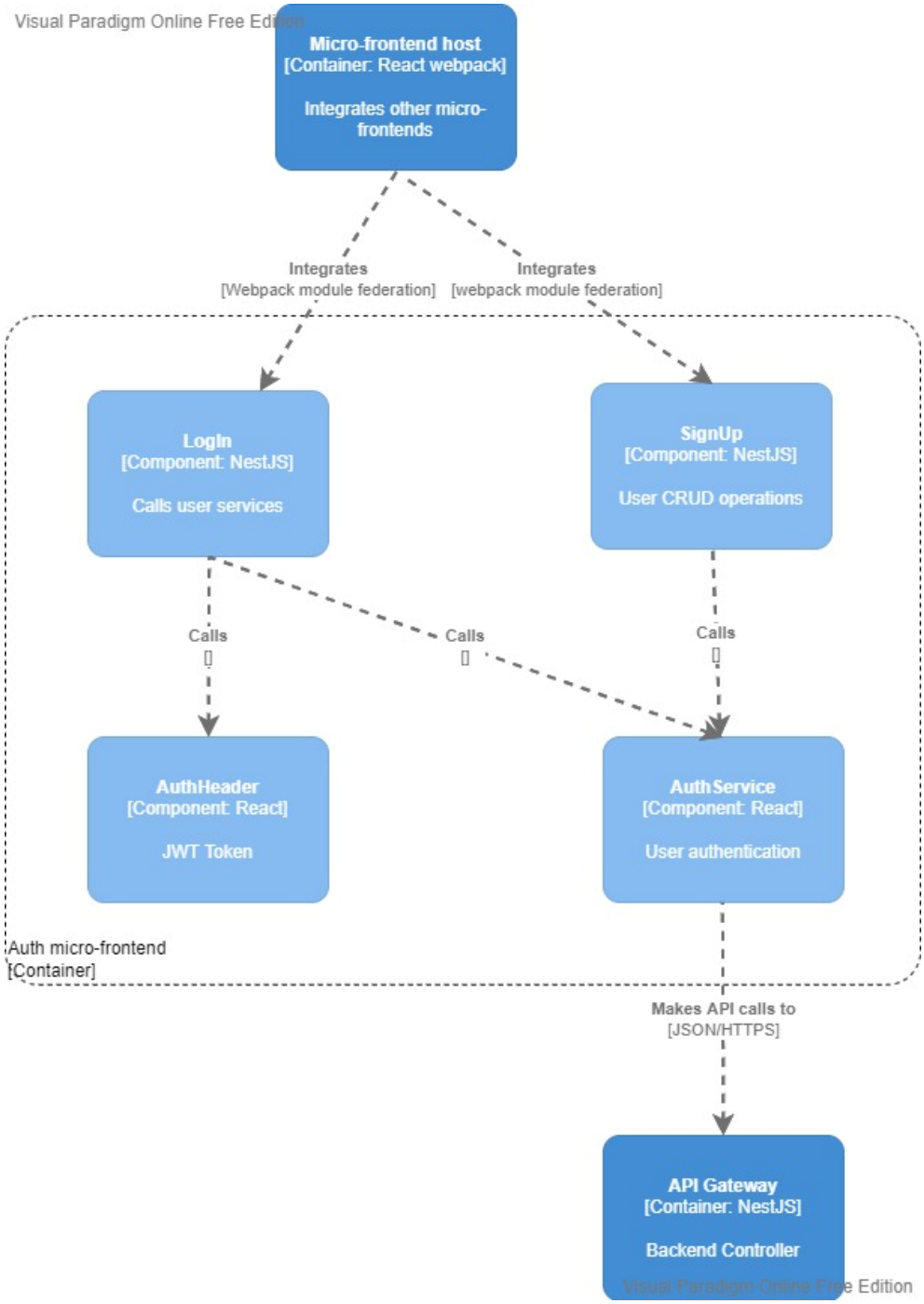
258/1

In the third level, since the internal architecture of the microservices is similar and the micro-frontends look almost the same as well, as representation we can see here the “user” microservice component and the “auth” micro-frontend component:



Bogdan Stefania-Franciska

258/1

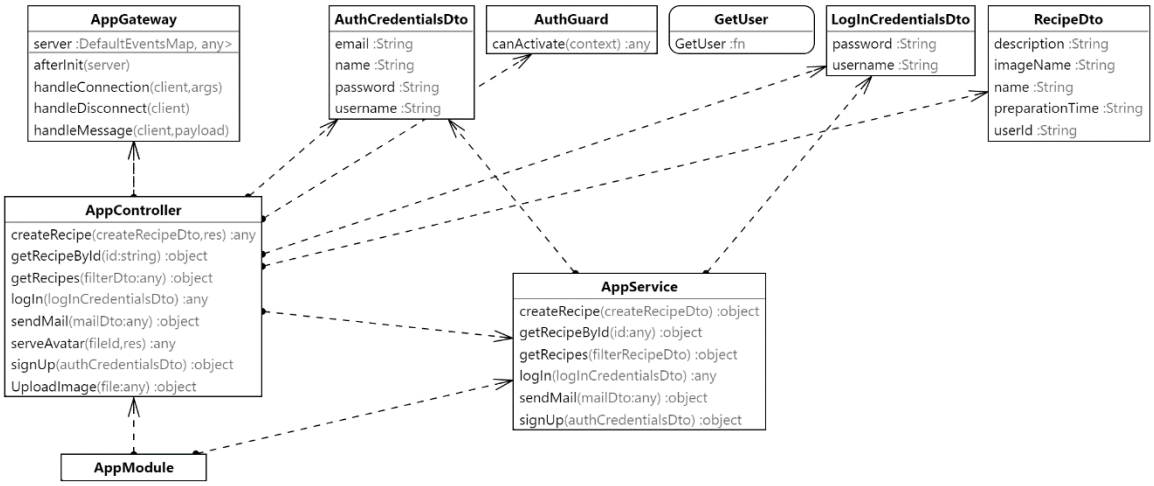


Bogdan Stefania-Franciska

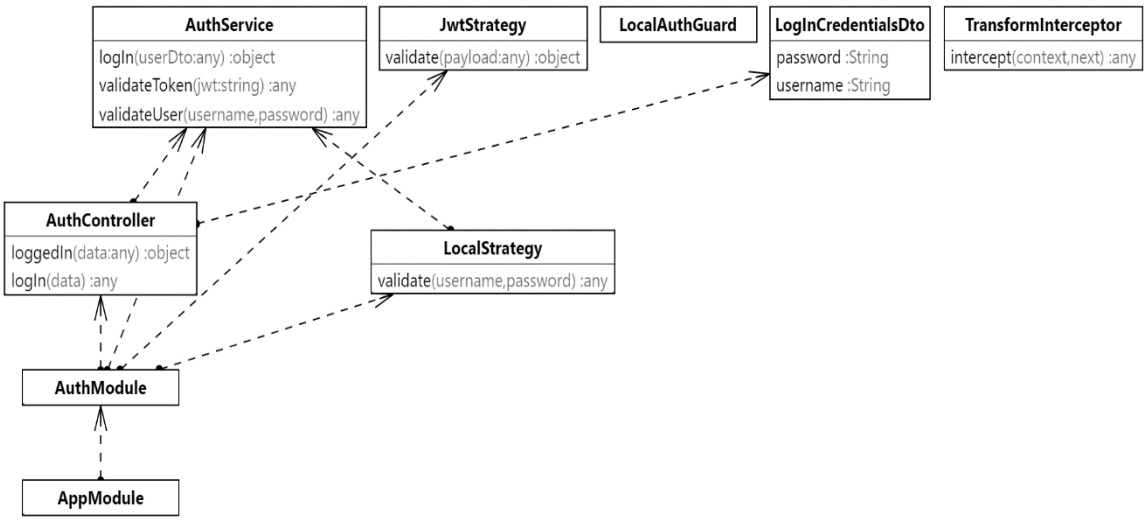
258/1

And finally, the class diagrams look like this:

1. API Gateway



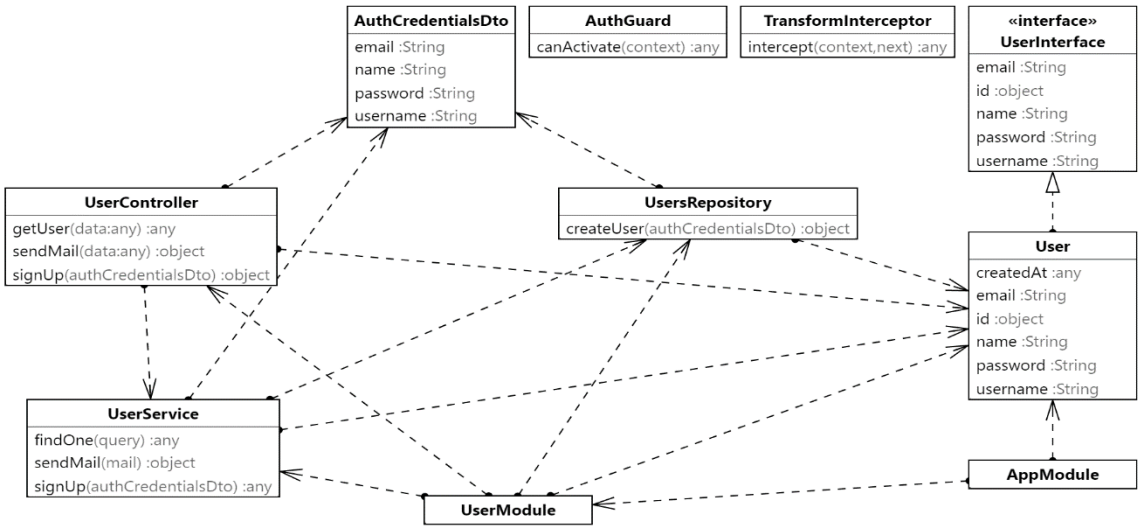
2. Auth Class



Bogdan Stefania-Franciska

258/1

3. User Class



4. Recipes class