Distributed Systems

Assignment 1

ONLINE ENERGY UTILITY PLATFORM

Student: Csatlos-Koncz Andrei

Group: 30441

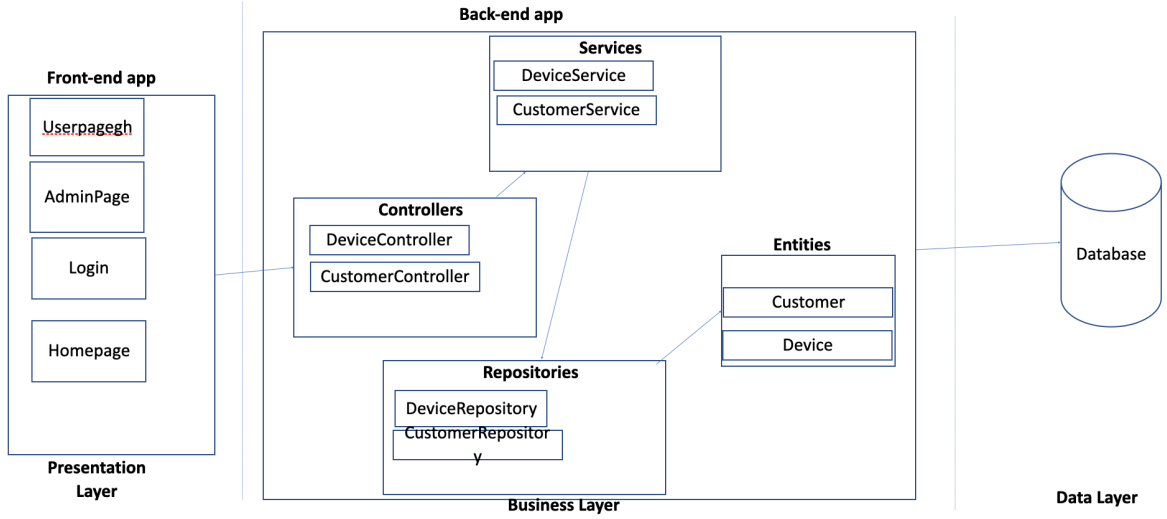
Faculty of Automation and Computer Science, TUCN

2022-2023

1. Conceptual Architecture of the Online Platform

The conceptual architecture used is MVT (Model View Template), employed by dotNet.

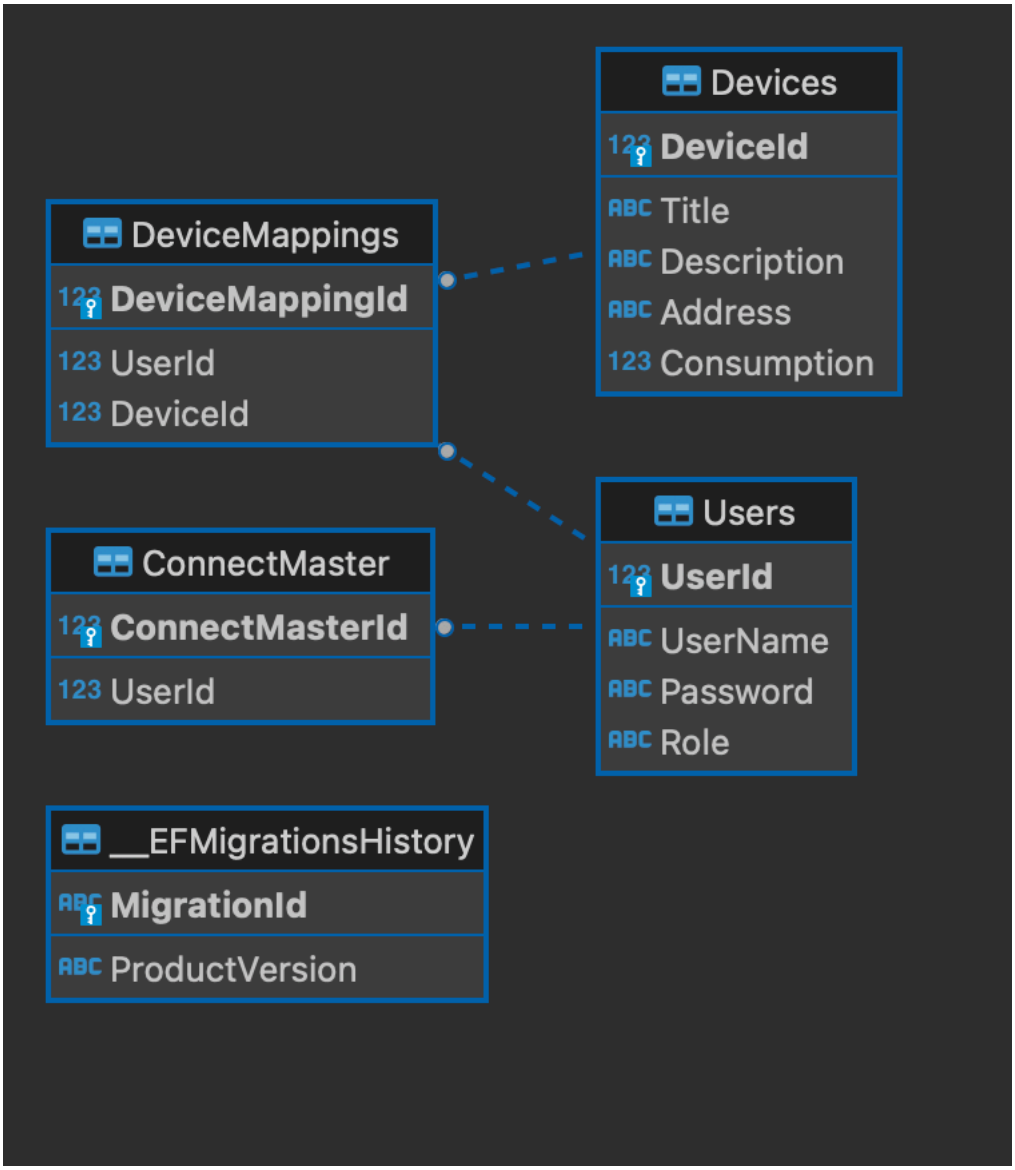
- **Model** – manages the data and is represented by a database (PostgreSQL).
- **View** – receives HTTP requests and sends HTTP responses.
- **Template** – the front-end layer and the dynamic HTML component, represented by Angular.



The Advanced Message Queuing Protocol (AMQP) was first implemented by RabbitMQ, an open-source message-broker program (also known as message-oriented middleware). Since then, RabbitMQ has been expanded with a plug-in architecture to support the Streaming Text Oriented Messaging Protocol (STOMP), MQ Telemetry Transport (MQTT), and other protocols.

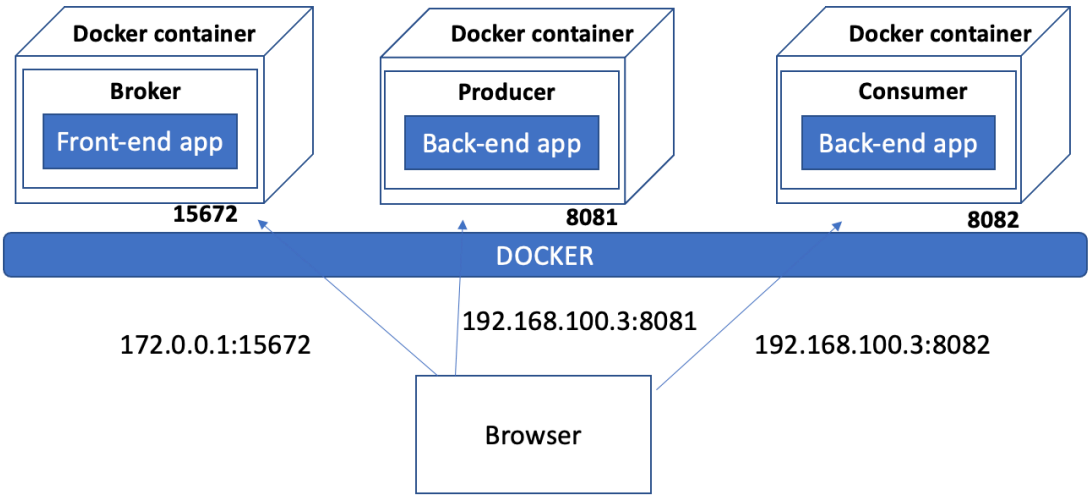
The Open Telecom Platform architecture for clustering and failover serves as the foundation for the RabbitMQ server, which is written in Erlang. For all popular programming languages, client libraries are available to communicate with the broker. The Mozilla Public License is used to release the source code.

2. Database Design



3. UML Deployment Diagram

Back-end: dotNet + C#
Front-end: Angular
Database: PostgreSQL



4. Build and Execution Considerations

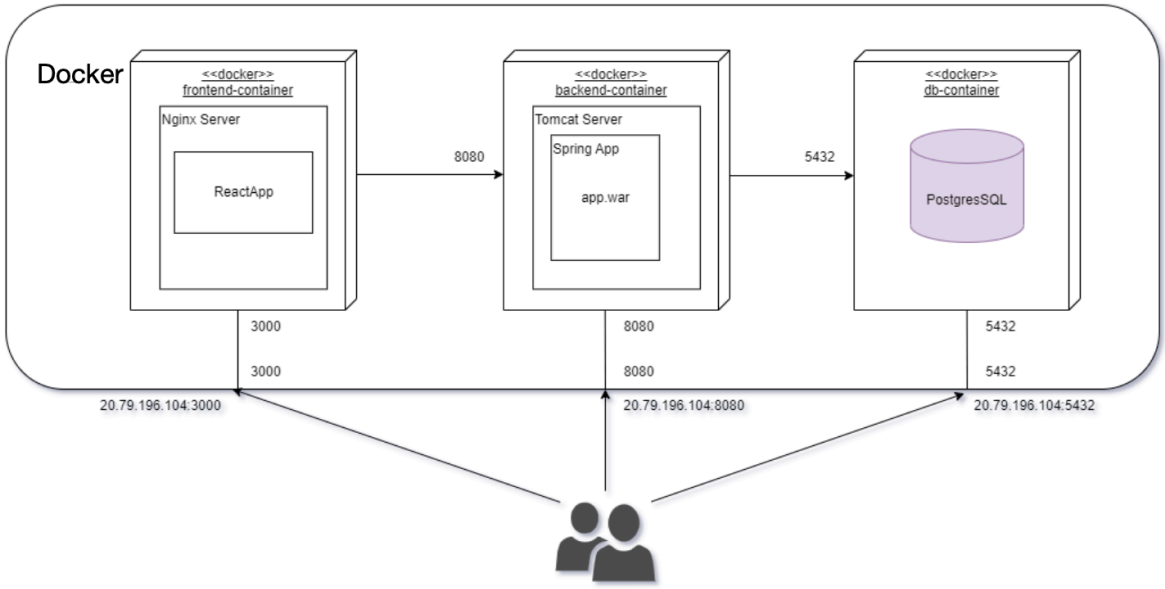
A **docker-compose.yml** file is given for front-end, back-end, and database containers in order to use Docker for deployment. A Docker network is built using the create private network.bat file to connect the containers.
Run the following commands after double-clicking the **create_private_network.bat** file to begin the deploy:

```
docker-compose up —build
```

There will be created three images:
- Postgres
- angular-frontend_angular-integration
- dotnet-integration

And three Docker containers:

- angular-frontend
- dotnet
- postgres



5. gRPC

gRPC is a high-performance, open-source framework for building remote procedure call (RPC) APIs. It uses the Protocol Buffers data serialization format and supports a wide variety of programming languages. gRPC enables client and server applications to communicate transparently, and it allows for the exchange of data using a strongly-typed schema. This makes it well-suited for use in microservices architectures and other distributed systems. Some of the key features of gRPC include bi-directional streaming, flow control, and a built-in health check mechanism. Additionally, gRPC supports features such as authentication and encryption, which can help to secure communications between client and server.

The SendMessage method receives a ChatMessage object from the client and logs the message, sender, and receiver information. If the receiver is currently connected, the message is sent to the receiver's stream. The message is also added to a messages list. The ReceiveMessage method opens a stream for the client, adds the client's username and stream to a subscriptions dictionary, and sends all previous messages in the messages list to the client's stream. The method then enters an infinite loop, waiting for new messages to be sent.