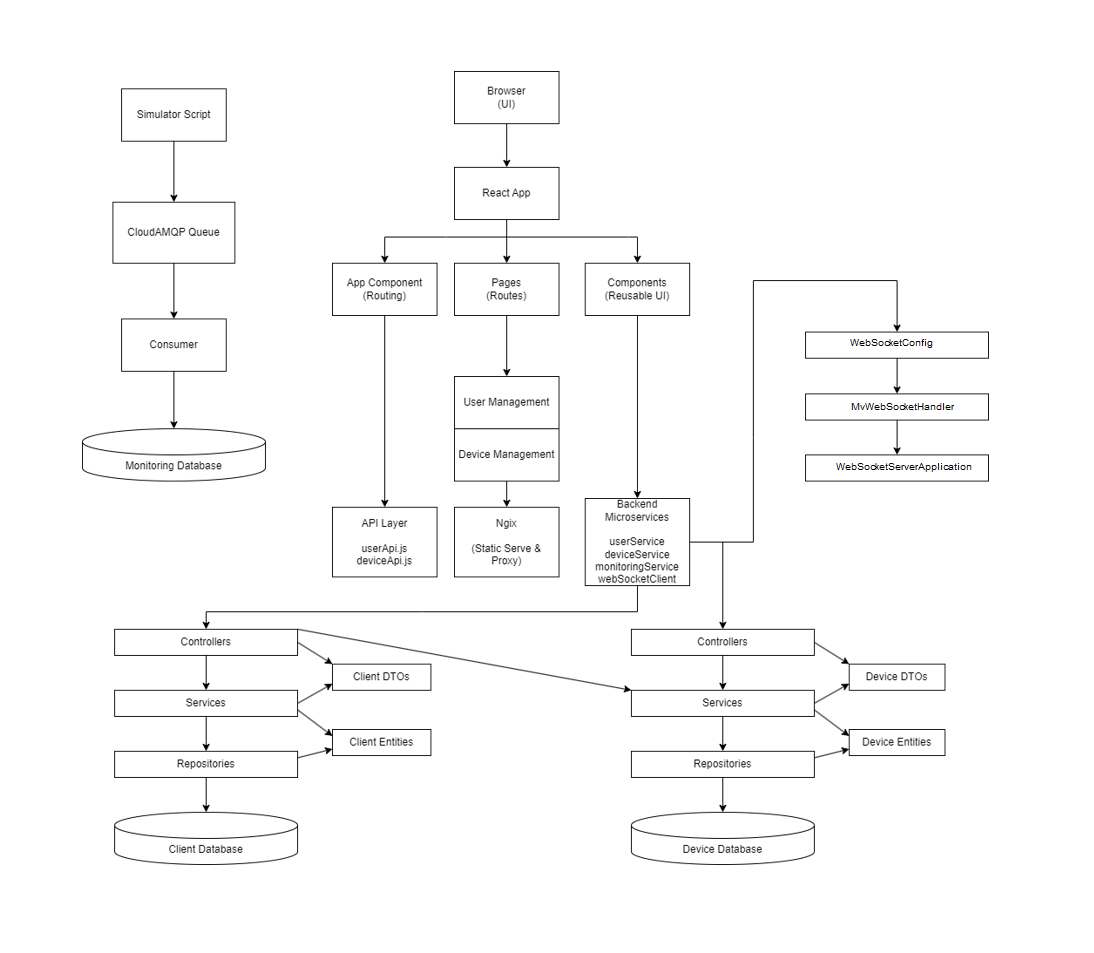
Assignment 3 – distributed systems

Vaida Diana-Laura, group 3

Conceptual architecture of the distributed system

****

UML Deployment diagram

A screenshot of a computer

Description automatically generated

All the functionalities from before work the same for the admin and user role.

A user can sign up, or log in. Logging in requires a token which will be generated by the back end and then decoded with the front end.

Having the role of the admin you can perfom CRUD on users and devices, and map a device to a user.

If you try to enter a page without having the required role or not signed in, you will be redirected to the home page. A JWT is generated for every person using their id and role. Every CRUD mehod in the back end takes the id and role and generates a token, then checks if the user role is admin from the token received from the webpage. The front end takes the generated token from the back end and decodes it (the role is needed so only admin or users can access specific pages) and the name is needed for the chat to display the user name.

Having the role of user means you are redirected to a chat that is using a websocket, and you can also chat with others users who are connected to the chat. Also 2 buttons for logout and devices assigned to that user are present. Logging out means you will be redirected back to home.

All the others micro services from previous assignments are there, running on docker and traefik, person, device, consumer, producer with RabbitMQ and the new chat micro service.

The chat uses a websocket to communicate with the front end, being made of 3 classes. The main class, which starts the microservice, the class that handles the connection between the logic of the chat microservice and the page the user sees with a HTTP session.

A message is formed by the messageID, senderID and the content of the message. A method is used to establish a connection with the websocket, and another to end the connection.

A functionality „user is typing...” is displayed for all the other users that use the chat when one of them writes, and dissappear when the message is finally sent, the message being displayed for everyone to see with the name of the person who sent it.

This works in the back end with methods for handling messages and broadcasting different statuses of the messages (TYPING, STOPTYPING), statuses that are sent back to the front end to handle styling. I use all kinds of types which are only available in the Socket framework, like WebSocketSession, TextMessage which are easier to work with.

The seen functionality is implemented – when all the other users see the message send by a specific user, a seen icon will appear next to the message for that specific user who sent it. For other user to send a ’’seen’’ signal, the cursor has to be in proximity of the chat window.