Constantin Andrei-Bogdan
Grupa 30641

Assignment 3

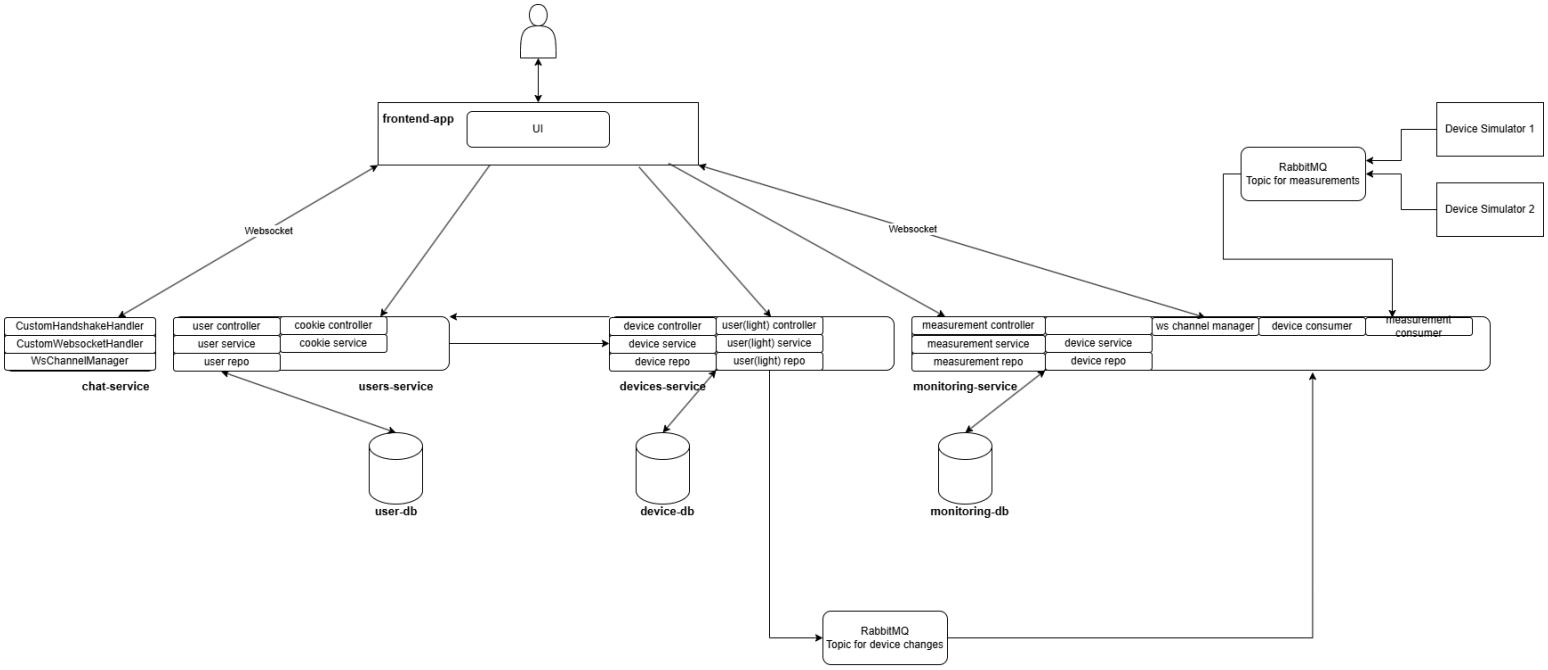

Figure 1 Diagrama arhitectura

Structura proiectului și organizarea pe microservicii

- Frontend- ReactJs
- User microservice + mysql db
- Device microservice + mysql db
- Monitoring microservice +mysql db
- Chat microservice

Logica chat-ului

Pentru a oferi functionalitatea de chat am implementat o logica ce are la baza o mapa de canale. Fiecare channel este destinat unui user pentru a putea comunica cu adminii.

Channelul contine o lista cu sesiunile curente conectate si numele acestuia. Astfel, mesajele intr-un canal vor fi distribuite doar catre sesiunile conectate la aceasta.

Pentru a se conecta la un canal, un user foloseste un link de forma ws/my_name&token=<token>. Tokenul este dat ca si query param pentru a putea autentifica, iar apoi autoriza userul. De exemplu daca my_name este admin, doar userii ce au un token de admin se pot conecta la acel canal. Un user nu poate accesa decat canalul care ii corespunde username-ului lui.

Atunci cand primeste un mesaj userul este notificat daca nu are chatul deschis. Totodata, atunci cand un user scrie un mesaj in chat, celalalt user va vedea acest lucru.

Mesajele trimise au formatul JSON, iar structura lor arata astfel: {content:<content>, sender:<sender>, receiver:<receiver>, action:<action>}. Action-ul poate avea 3 valori: SEEN, TYPING sau MESSAGE.

Constantin Andrei-Bogdan
Grupa 30641

Autentificare si Autorizare

Autentificarea si autorizarea se realizeaza pe baza de token. Tokenul este de tip JWT, fiind encodat si contine userul, data crearii si rolul.

User Service contine partea de middleware de autentificare. Astfel, fiecare serviciu face un request POST catre /token al user service pentru a valida tokenul primit. In acest fel toate serviciile sunt securizate si exista acces pe baza de roluri.

Traefik

Am utilizat Traefik ca load balancer si reverse proxy. Acesta redirecteaza traficul spre o instanta a unui serviciu. Am utilizat 2 replici ale monitoring service, 3 ale device service si tot 3 ale user service, 1 de chat service.

HTTPS si SSL

Pentru partea de requesturi care trec prin nginx, fiind reverse proxy intre frontend si traefik, din browserul userului se folosesc requesturi HTTPS. Am utilizat un ceritifcat self signed, fiind creat cu openssl. Celelalte servicii comunica in continuare prin HTTP in reteaua interna din docker.

Deploy cu docker

Fiecare dintre aplicatii contine Dockerfile si exista un docker-compose.yml global.

Am rulat docker-compose up -d –build pentru a crea imaginea pe baza fisierelor si pentru a o incarca in container.

Ruleaza pe o retea virtuala docker demo-net, creata anterior. Fiecare dintre servicii are propria retea, iar aceastea sunt si in reteaua principala demo_net.

Porturile externe legate sunt 3000 la react-app 80 si la traefik 8003 la 8000 si 8080:8080 pentru a putea monitoriza din dashboard si serviciile legate.

Pentru Traefik am adaugat la fiecare serviciu urmatoarele labels:

```
labels:
    - "traefik.enable=true"
    - "traefik.http.routers.users-service.rule=Host(`users-service.localhost`) || PathPrefix(`/users-service`)"
    - "traefik.http.middlewares.users-service-strip.stripprefix.prefixes=/users-service"
    - "traefik.http.routers.users-service.middlewares=users-service-strip"
    - "traefik.http.services.users-service.loadbalancer.server.port=8080"
     - "traefik.http.services.users-service.loadbalancer.sticky.cookie=true" # in special pentru serviciul de monitoring
```

Astfel, prin traefik, orice request la http://reverse-proxy/users-service este rutat la unul dintre serviciile de users. Optiunea de sticky cookie are rolul de a mentine userul conectat la aceeasi instanta, pentru a mentine sesiunea valida.

Nginx

Am utilizat si nginx ca reverse-proxy pt legatura traefik-aplicatia de front-end. Pentru a nu face publice url-urile cu care lucreaza frontendul in spate, am rutat catre traefik toate requesturile:

```
location ~ ^/api/user/(.*)$ {
    proxy_pass http://reverse-proxy/users-service/$1;
    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
}
```

Am adaugat la containerul de front end, ca volume, certificatul si cheia generate pentru ssl:

```
- ./frontend_sd/ssl/nginx.crt:/etc/ssl/certs/nginx.crt
- ./frontend_sd/ssl/nginx.key:/etc/ssl/private/nginx.key
```

Tot aici am schimbat ca portul expus sa fie 443. Tot traficul de pe 80 va fi rutat catre 443:

```
server {
    listen 80;
    return 301 https://$host$request_uri;
}
```

Concluzie

Astfel, prin adaugarea serviciul de chat, permitem utilizatorilor o mai buna interactiune cu aplicatia, oferind si o securitate mai buna prin utilizarea tokenilor pentru autentificare si autorizare, dar si upgrade-ul la protocolul HTTPS.

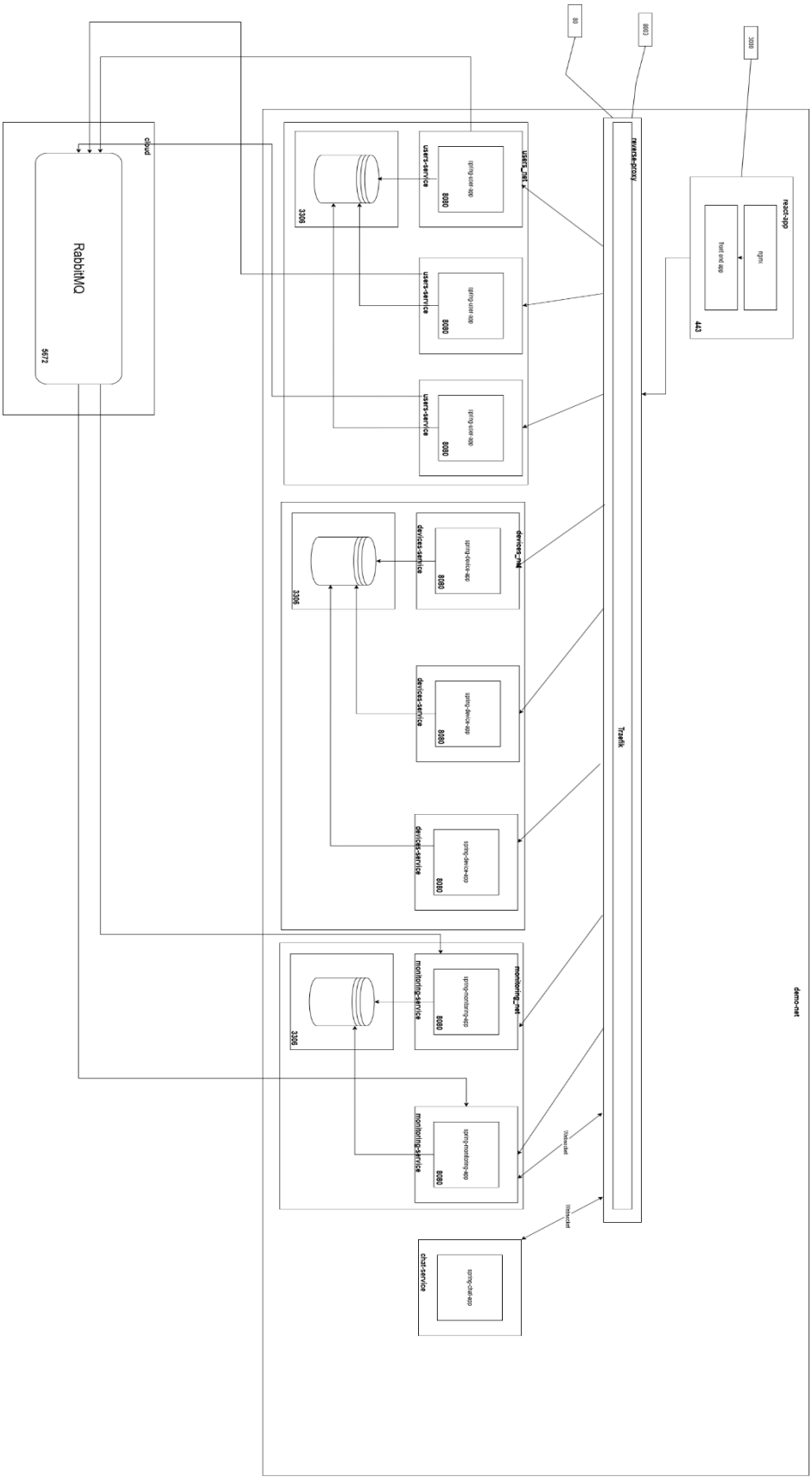

Figure 2 Diagrama de deployment