빌드 및 배포 정리

서울 1반 A103

프로젝트 기술 버전(툴, 버전)

1. 개발 환경

1) Backend

- Java : OpenJDK 11
- Gradle : 7.6.1
- Intellij : 2022.3.1
- Spring Boot : 2.7.9
- MySQL : 8.0.3

2) Frontend

- npm : 9.6.1
- js : 18.15.0
- VS Code : 1.77.0
- React : 18.2.0
- Redux : 4.2.1
- TypeScript : 4.9.5
- TailWind CSS : 3.2.7
- Styled Component : 5.3.8
- Vite : 4.1.0

3) AI

- fastapi : 0.94.0
- tensorflow : 2.11.0

4) CD/CD

- Server : AWS EC2 Ubuntu 20.04 LTS
- Docker : 20.10.21
- Nginx : 1.18.0
- Jenkins : 2.387.1

5) Cooperation & Communication

- Gitlab
- Jira
- MatterMost

- Notion

2. 빌드 특이사항, 방법

- Backend 빌드 및 실행 방법

```
cd backend

./gradlew clean build -x test

./gradlew bootRun
```

- Frontend 빌드 및 실행 방법

```
cd frontend

npm i

npm run build
```

3. 배포 특이사항, 방법

1) 도커 설치

```
# 패키지 매니저 최신화
$ sudo apt-get update

# Docker 설치
$ sudo apt install docker.io
```

2) Nginx 설치, 설정

Nginx 설치

```
sudo apt install nginx
```

방화벽 설정

- http, https allow

```
sudo ufw allow 'Nginx Full'
```

- 상태가 inactivate 로 나오는 경우

```
sudo ufw enable
```

SSL/TLS 접속을 위한 인증서 발급

- 인증서 발급을 간단하게 발급/갱신하는 패키지 설치

```
sudo snap install --classic certbot
```

- certbot을 활용하여 인증서 발급

```
sudo certbot --nginx

## nginx config file 을 만들지 않고 ssl file 만 필요한 경우
sudo certbot certonly --nginx
```

- Nginx default 파일 설정

```
server {

    listen [::]:443 ssl ipv6only=on; # managed by Certbot
    listen 443 ssl; # managed by Certbot
    ssl_certificate /etc/letsencrypt/live/j8a103.p.ssafy.io/fullchain.pem; # managed by Certbot
    ssl_certificate_key /etc/letsencrypt/live/j8a103.p.ssafy.io/privkey.pem; # managed by Certbot
    include /etc/letsencrypt/options-ssl-nginx.conf; # managed by Certbot
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by Certbot


  location / {
    proxy_pass http://localhost:3000;

  }

  location /api {
    proxy_pass http://localhost:8081;
  }

  location /ai {
    rewrite ^/ai(/.*)$ $1 break;
    proxy_pass https://gomgom-gpu-server.com;
  }

}
server {
    if ($host = j8a103.p.ssafy.io) {
        return 301 https://$host$request_uri;
    } # managed by Certbot


  listen 80 ;
  listen [::]:80 ;
    server_name j8a103.p.ssafy.io;
    return 404; # managed by Certbot


}
```

2) 백엔드 빌드 및 배포

- Dockerfile

```
FROM openjdk:11-jdk-slim
COPY build/libs/backend-0.0.1-SNAPSHOT.jar app.jar
ENTRYPOINT ["java", "-jar", "/app.jar"]
```

- Backend 배포

```
pwd

cd backend

chmod +x gradlew

./gradlew clean build -x test

if (sudo docker ps -a | grep "springboot"); then sudo docker stop springboot;
sudo docker rm springboot; sudo docker rmi springboot; fi

sudo docker build -t springboot .

sudo docker run -d -p 8081:8080 -v my-vol:/images --name springboot springboot
```

2) 프론트엔드 빌드 및 배포

- Dockerfile

```
FROM nginx

RUN mkdir /app

WORKDIR /app

RUN mkdir ./build

ADD ./dist ./build

RUN rm /etc/nginx/conf.d/default.conf

COPY ./nginx.conf /etc/nginx/conf.d

EXPOSE 80

CMD ["nginx", "-g", "daemon off;"]
```

- Frontend 배포

```
cd ../frontend

npm i

npm run build

if (sudo docker ps -a | grep "react"); then sudo docker stop react; sudo docker rm react; sudo docker rmi react; fi

sudo docker build -t react .

sudo docker run -d --name react -p 3000:80 -v my-vol:/app/build/assets/img_backend -v siba-vol:/app/build/ec2 react
```

4. 주요 계정 및 프로퍼티가 정의된 파일 목록

1) Spring Boot

- application.properties

```
server.servlet.context-path=/api

server.servlet.encoding.charset=UTF-8

#database

spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

spring.datasource.url=jdbc:mysql://j8a103.p.ssafy.io:3306/gomgom?serverTimezone=UTC&characterEncoding=UTF-8

#spring.datasource.url=jdbc:mysql://localhost:3306/ssafy_db?serverTimezone=UTC&characterEncoding=UTF-8

spring.datasource.username=ssafy

spring.datasource.password=ssafy

### jpa setting ###

spring.jpa.database-platform=org.hibernate.dialect.MySQL5Dialect

spring.jpa.show-sql=false

spring.jpa.hibernate.ddl-auto=validate

# carmelCase to underbar style

#spring.jpa.hibernate.naming.implicit-strategy=org.springframework.boot.orm.jpa.hibernate.SpringImplicitNamingStrategy

#spring.jpa.hibernate.naming.physical-strategy=org.springframework.boot.orm.jpa.hibernate.SpringPhysicalNamingStrategy

jwt.secret=jwt.secret=dyAeHubOOc8KaOfYB6XEQoEj1QzRlVgtjNL8PYs1A1tymZvvqkcEU7L1imkKHeDa

#60*60*1000

#3*24*60*60*1000

jwt.access_time=3600000

jwt.refresh_time=259200000

#file upload

spring.servlet.multipart.enabled=true

spring.servlet.multipart.max-request-size=30MB

spring.servlet.multipart.max-file-size=10MB

part.upload.path=/images/
```

2) React

- package.json

```
{
  "name": "frontend",
  "private": true,
  "version": "0.0.0",
  "type": "module",
  "babelMacros": {
    "twin": {
      "preset": "styled-components"
    }
  },
  "scripts": {
    "dev": "vite",
    "build": "tsc && vite build",
    "preview": "vite preview"
  },
  "dependencies": {
    "@emotion/react": "^11.10.6",
    "@emotion/styled": "^11.10.6",
    "@mui/icons-material": "^5.11.11",
```

```
    "@mui/material": "^5.11.14",
    "@mui/styled-engine-sc": "^5.11.11",
    "@react-three/drei": "^9.57.3",
    "@react-three/fiber": "^8.12.0",
    "@reduxjs/toolkit": "^1.9.3",
    "@tailwindcss/aspect-ratio": "^0.4.2",
    "axios": "^1.3.4",
    "react": "^18.2.0",
    "react-audio-analyser": "^1.0.0",
    "react-dom": "^18.2.0",
    "react-media-recorder": "^1.6.6",
    "react-player": "^2.12.0",
    "react-redux": "^8.0.5",
    "react-router-dom": "^6.9.0",
    "react-slick": "^0.29.0",
    "react-youtube": "^10.1.0",
    "redux-devtools-extension": "^2.13.9",
    "redux-persist": "^6.0.0",
    "slick-carousel": "^1.8.1",
    "styled-components": "^5.3.8",
    "three": "^0.150.1"
  },
  "devDependencies": {
    "@types/react": "^18.0.27",
    "@types/react-dom": "^18.0.10",
    "@types/react-slick": "^0.23.10",
    "@types/redux-persist": "^4.3.1",
    "@types/styled-components": "^5.1.26",
    "@types/three": "^0.139.0",
    "@vitejs/plugin-react": "^3.1.0",
    "babel-plugin-macros": "^3.1.0",
    "babel-plugin-styled-components": "^2.0.7",
    "postcss": "^8.4.21",
    "tailwindcss": "^3.2.7",
    "twin.macro": "^3.1.0",
    "typescript": "^4.9.3",
    "vite": "^4.1.0"
  }
}
```

3) Tensorflow

- Requirements_tensorflow.txt

```
absl-py==1.4.0
albumentations==1.3.0
anyio==3.6.2
astunparse==1.6.3
cachetools==5.3.0
certifi @ file:///C:/b/abs_85o_6fm0se/croot/certifi_1671487778835/work/certifi
charset-normalizer==3.1.0
click==8.1.3
colorama==0.4.6
fastapi==0.94.0
flatbuffers==23.3.3
gast==0.4.0
google-auth==2.16.3
google-auth-oauthlib==0.4.6
google-pasta==0.2.0
grpcio==1.51.3
h11==0.14.0
h5py==3.8.0
idna==3.4
imageio==2.26.1
importlib-metadata==6.1.0
joblib==1.2.0
keras==2.11.0
lazy_loader==0.2
libclang==16.0.0
Markdown==3.4.3
MarkupSafe==2.1.2
networkx==3.0
numpy==1.24.2
oauthlib==3.2.2
```

```
opencv-python-headless==4.7.0.72
opt-einsum==3.3.0
packaging==23.0
protobuf==3.19.6
pyasn1==0.4.8
pyasn1-modules==0.2.8
pydantic==1.10.6
python-multipart==0.0.6
PyWavelets==1.4.1
PyYAML==6.0
qudida==0.0.4
requests==2.28.2
requests-oauthlib==1.3.1
rsa==4.9
scikit-image==0.20.0
scikit-learn==1.2.2
scipy==1.9.1
six==1.16.0
sniffio==1.3.0
starlette==0.26.0.post1
tensorboard==2.11.2
tensorboard-data-server==0.6.1
tensorboard-plugin-wit==1.8.1
tensorflow==2.11.0
tensorflow-estimator==2.11.0
tensorflow-intel==2.11.0
tensorflow-io-gcs-filesystem==0.31.0
termcolor==2.2.0
threadpoolctl==3.1.0
tifffile==2023.3.21
typing_extensions==4.5.0
urllib3==1.26.15
uvicorn==0.21.0
Werkzeug==2.2.3
wincertstore==0.2
wrapt==1.15.0
zipp==3.15.0
```