포팅메뉴얼문서 - 빌드및배포

태그

버전 및 설정

BackEnd

- Java 17
- SpringBoot 3.17
- querydsl-jpa 5.0.0
- swagger 3.0.0
- nginx 1.18.0(ubuntu)

build.gradle

```
plugins {
    id 'java'
    id 'org.springframework.boot' version '3.1.7'
    id 'io.spring.dependency-management' version '1.1.4'
}

group = 'com.f17coders'
version = '0.0.1-SNAPSHOT'

java {
    sourceCompatibility = '17'
}

configurations {
    compileOnly {
        extendsFrom annotationProcessor
    }
}

repositories {
    mavenCentral()
}

dependencies {
    implementation 'org.springframework.boot:spring-boot-starter-web'
    implementation 'org.springframework.boot:spring-boot-starter-data-jpa'
    implementation 'org.springframework.boot:spring-boot-starter-validation'
    implementation 'org.springdoc:springdoc-openapi-starter-webmvc-ui:2.0.2'
    compileOnly 'org.projectlombok:lombok'
    annotationProcessor 'org.projectlombok:lombok'
    testImplementation 'org.springframework.boot:spring-boot-starter-test'

    // QueryDSL 설정
    implementation "com.querydsl:querydsl-jpa:5.0.0:jakarta"
    annotationProcessor "com.querydsl:querydsl-apt:5.0.0:jakarta"
    annotationProcessor "jakarta.annotation:jakarta.annotation-api"
    annotationProcessor "jakarta.persistence:jakarta.persistence-api"
    // spring security
    implementation 'org.springframework.boot:spring-boot-starter-oauth2-client'
    implementation 'org.springframework.boot:spring-boot-starter-security'
    testImplementation 'org.springframework.security:spring-security-test'
    // jwt
    implementation 'io.jsonwebtoken:jjwt:0.9.1'
    implementation 'javax.xml.bind:jaxb-api:2.3.1' // jdk 8이상 필요
    // gson
    implementation 'com.google.code.gson:gson:2.9.0'
    implementation 'org.springframework.boot:spring-boot-starter-data-redis'
    // mySql 연결
    runtimeOnly 'com.mysql:mysql-connector-j'
    implementation group: 'org.javassist', name: 'javassist', version: '3.15.0-GA'
    // db
    implementation 'org.mariadb.jdbc:mariadb-java-client'
    // https://mvnrepository.com/artifact/io.springfox/springfox-swagger-ui
    implementation group: 'io.springfox', name: 'springfox-swagger-ui', version: '3.0.0'
```

```
    // mongodb
    implementation 'org.springframework.boot:spring-boot-starter-data-mongodb'
    // websocket
    implementation 'org.springframework.boot:spring-boot-starter-websocket'
}



tasks.named('bootBuildImage') {
    builder = 'paketobuildpacks/builder-jammy-base:latest'
}

tasks.named('test') {
    useJUnitPlatform()
}

// Querydsl 설정
def generatedDir = 'build/generated'

clean {
    delete file (generatedDir)
}
```

application.yml

```
spring:
  datasource:
    username: <사용자 계정>
    driver-class-name: org.mariadb.jdbc.Driver
    password: <사용자의 비밀번호>
    url: jdbc:mariadb://<host>:<port>/<database>?useSSL=false&allowPublicKeyRetrieval=true&serverTimezone
  jpa:
    hibernate:
      ddl-auto: validate
      show_sql: true
      dialect: org.hibernate.dialect.MariaDBDialect
    properties:
      hibernate:
        show_sql: true
        format_sql: true
        use_sql_comments: true
        logging.level:
        org.hibernate.SQL: debug
        #        org.hibernate.type: trace
  data:
    mongodb:
      uri: mongodb://<사용자 계정>:<사용자의 비밀번호>@<host>:<port>/<database>?authSource=admin&authMechanism=
    redis:
      host: <host>
      port: <port>
# OAuth
security:
  oauth2:
    client:
      registration:
        kakao:
          client-id: ffff43456dc6c817e9c170c889116ea1
          client-secret: 9sUohDxFqvuLBt4za5EviV9p1pk12NWn
          redirect-uri: https://i10a810.p.ssafy.io/login/oauth2/code/kakao
          authorization-grant-type: authorization_code
          client-authentication-method: client_secret_post
          client-name: Kakao
          scope:
            - profile_nickname
            - profile_image
      provider:
        kakao:
          authorization-uri: https://kauth.kakao.com/oauth/authorize
          token-uri: https://kauth.kakao.com/oauth/token
          user-info-uri: https://kapi.kakao.com/v2/user/me
          user-name-attribute: id
```

```
logging.level:
  org.hibernate.SQL: info
  org.hibernate.type: info
#  org.springframework.security: info

classhub:
  react:
    domain: "<react host>"

key:
  jwt:
    secret: <jwt-secret>

server:
  port: <port>
```

FrontEnd

- Node.js 20.11.0
- npm: 10.2.4
- vs-code: 1.86.1

Package.json

```
{
  "dependencies": {
    "@emotion/react": "^11.11.3",
    "@emotion/styled": "^11.11.0",
    "@mui/icons-material": "^5.15.5",
    "@mui/material": "^5.15.5",
    "@mui/styled-engine-sc": "^6.0.0-alpha.12",
    "@mui/x-data-grid": "^6.18.7",
    "@redux-devtools/extension": "^3.3.0",
    "@reduxjs/toolkit": "^2.0.1",
    "@stomp/stompjs": "^7.0.0",
    "@tanstack/react-query": "^5.17.15",
    "axios": "^1.6.5",
    "crypto-js": "^4.2.0",
    "dompurify": "^3.0.8",
    "framer-motion": "^11.0.3",
    "jsdom": "^24.0.0",
    "react": "^18.2.0",
    "react-dom": "^18.2.0",
    "react-quill": "^2.0.0",
    "react-redux": "^9.1.0",
    "react-responsive-carousel": "^3.2.23",
    "react-router-dom": "^6.22.0",
    "react-slick": "^0.29.0",
    "redux-persist": "^6.0.0",
    "sockjs-client": "^1.6.1",
    "styled-components": "^6.1.8",
    "sweetalert2": "^11.10.4",
    "sweetalert2-react-content": "^5.0.7",
    "swiper": "^11.0.6"
  },
  "devDependencies": {
    "@types/react": "^18.2.43",
    "@types/react-dom": "^18.2.17",
    "@vitejs/plugin-react": "^4.2.1",
    "eslint": "^8.55.0",
    "eslint-plugin-react": "^7.33.2",
    "eslint-plugin-react-hooks": "^4.6.0",
    "eslint-plugin-react-refresh": "^0.4.5",
    "vite": "^5.0.8"
  }
}
```

- 버전 설치

  ```
  $ npm install
  ```

- 실행

```
$ npm run dev
```

DataBase

- mariadb: 11.2.2
- redis: 7.2.4
- mongoDB: 7.0

외부 서비스 정보

- 소셜 로그인
  - 카카오 로그인 API
    - https://developers.kakao.com/docs/latest/ko/kakaologin/rest-api

---

배포

1. AWS EC2 인스턴스 생성
2. 우분투 서버 세팅

```
$ sudo timedatectl set-timezone Asia/Seoul
$ sudo sed -i 's/ap-northeast-2.ec2.archive.ubuntu.com/mirror.kakao.com/g' /etc/apt/sources.list
$ sudo apt-get -y update && sudo apt-get -y upgrade
```

- Swap 영역 할당

```
$ sudo fallocate -l 4G /swapfile
$ sudo chmod 600 /swapfile
$ sudo mkswap /swapfile
$ sudo swapon /swapfile
$ sudo echo '/swapfile none swap sw 0 0' | sudo tee -a /etc/fstab
$ free -h
```

3. 도커 설치

```
$ sudo apt-get -y install apt-transport-https ca-certificates curl gnupg-agent software-properties-com
$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
$ sudo add-apt-repository "deb [arch=arm64] https://download.docker.com/linux/ubuntu $(lsb_release -cs
$ sudo apt-get -y update
$ sudo apt-get -y install docker-ce docker-ce-cli containerd.io
$ sudo usermod -aG docker ubuntu
```

```
$ sudo service docker restart
```

4. DB 설치
   - Maria DB 설치

```
$ sudo docker pull mariadb:latest
$ docker run -d -p 3306:3306 -e MYSQL_ROOT_PASSWORD={사용자 비밀번호} -v /var/lib/mysql:/var/lib/mysql
```

   - Redis 설치

```
$ docker pull redis
$ docker network create redis-network
$ docker run --name redis-app -p 6379:6379 --network redis-network -it -d redis
```

   - MongoDB 설치

```
$ docker pull mongo
$ docker run --name mongodb -v ~/data:/data/db -d -e MONGO_INITDB_ROOT_USERNAME={사용자이름} -e MONGO_
```

5. DB에 데이터 삽입

5. Nginx 설치

```
$ docker pull nginx
$ docker run -d --restart always -v /home/ubuntu/cert:/etc/letsencrypt/live/i10a810.p.ssafy.io -p 8(
```

6. Nginx.conf 파일 수정

```
$ docker exec -it nginx /bin/bash
```

```
$ vim /etc/nginx/conf.d/default.conf
```

- Nginx.conf

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

server{
    listen 443 ssl;
    server_name {server_url};
    ssl_certificate {ssl};
    ssl_certificate_key {인증키};

    ssl_protocols TLSv1 TLSv1.1 TLSv1.2;
    ssl_prefer_server_ciphers on;

location / {
    proxy_pass {url};
    proxy_set_header Host $host;
    proxy_set_header X-Real_IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
}
location /api/{
    proxy_pass {url};
    proxy_set_header Host $host;
    proxy_set_header X-Real_IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    }
location /login/oauth2/{
    proxy_pass {url};
    proxy_set_header Host $host;
    proxy_set_header X-Real_IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
}

    location /api/chat/ {
            proxy_pass {url};
            proxy_http_version 1.1;
            proxy_set_header Upgrade $http_upgrade;
            proxy_set_header Connection "upgrade";
            proxy_set_header Host $http_host;
    }
}
```

```
$ exit
```

```
$ docker restart nignx
```

7. git clone

```
$ git clone https://lab.ssafy.com/s10-webmobile2-sub2/S10P12A810.git
```

8. BackEnd 실행
   a. docker file 확인

```
FROM openjdk:17
LABEL authors="SSAFYA810"
```

```
COPY build/libs/*.jar app.jar
ENTRYPOINT ["java", "-jar", "app.jar"]
```

b. 프로젝트 빌드

```
$ ./gradlew clean build
```

c. docker build

```
$ docker build -t {docker 이름:태그} .
$ docker run -d -p host포트번호:컨테이너포트번호 -name [컨테이너이름] [이미지]
```

9. FrontEnd 실행

a. docker file 확인

```
## 1. Build React App
FROM node:alpine as builder
WORKDIR /app
COPY package*.json .
RUN npm ci
COPY . .
RUN npm run build

## 2. React를 Nginx로 Serve
FROM nginx:latest

RUN rm /etc/nginx/conf.d/default.conf
COPY ./nginx.conf /etc/nginx/conf.d
RUN ls -al
WORKDIR /usr/share/nginx/html

COPY --from=builder /app/dist /usr/share/nginx/html

EXPOSE 80

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

b. docker build

```
$ docker build -t {docker 이름:태그} .
$ docker run -d -p host포트번호:컨테이너포트번호 -name [컨테이너이름] [이미지]
```