포팅 매뉴얼

Front

- package.json

```
{
  "name": "noah",
  "version": "0.1.0",
  "private": true,
  "dependencies": {
    "@date-io/date-fns": "^3.0.0",
    "@emotion/react": "^11.11.4",
    "@emotion/styled": "^11.11.5",
    "@mui/lab": "^5.0.0-alpha.169",
    "@mui/material": "^5.15.14",
    "@mui/x-date-pickers": "^7.1.0",
    "@mui/x-date-pickers-pro": "^7.1.0",
    "@react-google-maps/api": "^2.19.3",
    "@testing-library/jest-dom": "^5.17.0",
    "@testing-library/react": "^13.4.0",
    "@testing-library/user-event": "^13.5.0",
    "aos": "^2.3.4",
    "axios": "^1.6.8",
    "date-fns": "^3.6.0",
    "dayjs": "^1.11.10",
    "firebase": "^10.9.0",
    "react": "^18.2.0",
    "react-chartjs-2": "^5.2.0",
    "react-datepicker": "^6.6.0",
    "react-dom": "^18.2.0",
    "react-hot-toast": "^2.4.1",
    "react-icons": "^5.0.1",
    "react-modal": "^3.16.1",
    "react-rating": "^2.0.5",
    "react-router-dom": "^6.22.3",
```

```
    "react-scripts": "5.0.1",
    "react-slick": "^0.30.2",
    "react-spinners": "^0.13.8",
    "slick-carousel": "^1.8.1",
    "styled-components": "^6.1.8",
    "web-vitals": "^2.1.4",
    "workbox-background-sync": "^6.6.0",
    "workbox-broadcast-update": "^6.6.0",
    "workbox-cacheable-response": "^6.6.0",
    "workbox-core": "^6.6.0",
    "workbox-expiration": "^6.6.0",
    "workbox-google-analytics": "^6.6.1",
    "workbox-navigation-preload": "^6.6.0",
    "workbox-precaching": "^6.6.0",
    "workbox-range-requests": "^6.6.0",
    "workbox-routing": "^6.6.0",
    "workbox-strategies": "^6.6.0",
    "workbox-streams": "^6.6.0",
    "zustand": "^4.5.2"
  },
  "scripts": {
    "start": "react-scripts start",
    "build": "react-scripts build",
    "test": "react-scripts test",
    "eject": "react-scripts eject"
  },
  "eslintConfig": {
    "extends": [
      "react-app",
      "react-app/jest"
    ]
  },
  "browserslist": {
    "production": [
      ">0.2%",
      "not dead",
      "not op_mini all"
    ],
```

```
    "development": [
      "last 1 chrome version",
      "last 1 firefox version",
      "last 1 safari version"
    ]
  }
}
```

- env

```
REACT_APP_FIREBASE_API_KEY = AIzaSyCmg0AVb7iGo0eyxGFvUU6AkXx9
REACT_APP_FIREBASE_AUTH_DOMAIN = noah-511a6.firebaseapp.com
REACT_APP_FIREBASE_PROJECT_ID = noah-511a6
REACT_APP_FIREBASE_STORAGE_BUCKET = noah-511a6.appspot.com
REACT_APP_FIREBASE_MESSAGINGSENDER_ID = 622848761229
REACT_APP_FIREBASE_APP_ID = 1:622848761229:web:d576d2283d09a0
REACT_APP_FIREBASE_MEASUREMENT_ID = G-6C7TZ1CV3H
REACT_APP_FIREBASE_VAPID_KEY = BJgOtDxMam8Lgz5rymS0jRg7zLumFZ
# REACT_APP_URL=https://j10b106.p.ssafy.io
REACT_APP_URL="http://localhost:8080"
REACT_APP_GOOGLE_API_KEY = AIzaSyDQuG0EPBRz632DtyOLTtopsQ97Uu
```

yml 파일

- DB

```
spring:
  datasource:
    url: jdbc:mysql://[IP주소]:[포트번호]/[데이터베이스명]?allowP
ublicKeyRetrieval=true&useSSL=true&serverTimezone=Asia/Seou
l
    username: [데이터베이스 사용자명]
    password: [데이터베이스 비밀번호]
    driver-class-name: com.mysql.cj.jdbc.Driver

  jpa:
    hibernate:
      ddl-auto: create/update/validate [데이터베이스 생성 옵션]
```

```
    properties:
      hibernate:
        show_sql: true [SQL 로그 출력]
        format_sql: true [SQL 출력 포맷]

  data:
    redis:
      host: [Redis IP주소]
      port: [Redis 포트번호]
      time-to-live: [Redis 캐시의 TTL(시간-살아있음)]
```

- filter

```
filter-path:
  paths:
    member: "/api/v1/member"
    memberSocial: "/api/v1/member/social"
    memberLogin: "/api/v1/member/login/**"
    memberNickname: "/api/v1/member/nickname/**"
    memberEmail: "/api/v1/member/email/**"
    memberPasswordReset: "/api/v1/member/password-reset"
    apiDocs: "/api-docs/**"
    v2ApiDocs: "/v2/api-docs/**"
    v3ApiDocs: "/v3/api-docs/**"
    webjars: "/webjars/**"
    swagger: "/swagger/**"
    swaggerUi: "/swagger-ui/**"
    swaggerConfig: "/swagger-config/**"
    swaggerResources: "/swagger-resources/**"
    exchangeRate: "/api/v1/exchange/rateinfo"
    market: "api/v1/bank/qr/withdraw"
    nonlogin: "/api/v1/Suggest/nonlogin"

# 비로그인 시에 사용해야하는 경로 인가 작업
```

- jwt

```
jwt:
  SECRET_KEY: [jwt 비밀 키]
  ACCESS_TIME: [엑세스 토큰 유효시간]
  REFRESH_TIME: [리프레시 토큰 유효시간]
```

- mail

```
spring:
  mail:
    host: smtp.gmail.com
    port: 587
    username: [보내는 사람 이메일]
    password: [이메일 비밀번호]
    properties:
      mail:
        smtp:
          auth: true
          starttls:
            enable: true
            required: true
          connection-timeout: 5000
          timeout: 5000
          write timeout: 5000
      auth-code-expiration-millis: 300000
```

- notification

```
schedule:
  notify_pay: 0 0 9 ? * *
  auto_pay: 0 0 10 ? * *
  notify_exchange: 0 10 11 ? * *

# 스케줄러 속성
```

Server

- 버전에 맞게 설치

```
Java - Zulu 17.0.9
node - 20.11.0-alpine
```

- 도커 이미지 버전

```
Nginx - 1.15-alpine
certbot - v0.36.0
redis - latest
mysql - latest
jdk - azul/zulu-openjdk:17
my-node-app
```

- certbot 설정
  - docker-compose.yml

```
version: '3.8'

services:
  nginx:
    container_name: "nginx"
    image: nginx:1.15-alpine
    restart: unless-stopped
    volumes:
      - ./data/nginx/conf.d:/etc/nginx/conf.d
      - ./data/certbot/conf:/etc/letsencrypt
      - ./data/certbot/www:/var/www/certbot
    ports:
      - "80:80"
      - "443:443"
  certbot:
    container_name: "certbot"
    image: certbot/certbot:v0.36.0
    restart: unless-stopped
    volumes:
      - ./data/certbot/conf:/etc/letsencrypt
      - ./data/certbot/www:/var/www/certbot
```

- init-letsencrypt.sh

```
#!/bin/bash

if ! [ -x "$(command -v docker-compose)" ]; then
  echo 'Error: docker-compose is not installed.' >&2
  exit 1
fi

domains=("j10b106.p.ssafy.io") // 각자 등록한 도메인
rsa_key_size=4096
data_path="./data/certbot"
email="wjsguscjf23@naver.com" # Adding a valid address is
staging=0 # Set to 1 if you're testing your setup to avoid

if [ -d "$data_path" ]; then
  read -p "Existing data found for $domains. Continue and
  if [ "$decision" != "Y" ] && [ "$decision" != "y" ]; the
    exit
  fi
fi



if [ ! -e "$data_path/conf/options-ssl-nginx.conf" ] || [
  echo "### Downloading recommended TLS parameters ..."
  mkdir -p "$data_path/conf"
  curl -s https://raw.githubusercontent.com/certbot/certbo
  curl -s https://raw.githubusercontent.com/certbot/certbo
  echo
fi

echo "### Creating dummy certificate for $domains ..."
path="/etc/letsencrypt/live/$domains"
mkdir -p "$data_path/conf/live/$domains"
docker-compose run --rm --entrypoint "\
  openssl req -x509 -nodes -newkey rsa:$rsa_key_size -days
    -keyout '$path/privkey.pem' \
    -out '$path/fullchain.pem' \
```

```
    -subj '/CN=localhost'" certbot
echo



echo "### Starting nginx ..."
docker-compose up --force-recreate -d nginx
echo

echo "### Deleting dummy certificate for $domains ..."
docker-compose run --rm --entrypoint "\
  rm -Rf /etc/letsencrypt/live/$domains && \
  rm -Rf /etc/letsencrypt/archive/$domains && \
  rm -Rf /etc/letsencrypt/renewal/$domains.conf" certbot
echo



echo "### Requesting Let's Encrypt certificate for $domain
#Join $domains to -d args
domain_args=""
for domain in "${domains[@]}"; do
  domain_args="$domain_args -d $domain"
done

# Select appropriate email arg
case "$email" in
  "") email_arg="--register-unsafely-without-email" ;;
  *) email_arg="--email $email" ;;
esac

# Enable staging mode if needed
if [ $staging != "0" ]; then staging_arg="--staging"; fi

docker-compose run --rm --entrypoint "\
  certbot certonly -a webroot -v --debug-challenges -w /va
    $staging_arg \
    $email_arg \
    $domain_args \
    --rsa-key-size $rsa_key_size \
```

```
    --agree-tos \
    --force-renewal" certbot
echo

echo "### Reloading nginx ..."
docker-compose exec nginx nginx -s reload
```

- data/nginx/cof.d/app.conf

```
server {
    listen 80;
    server_name "j10b106.p.ssafy.io";
    server_tokens off;

    location /.well-known/acme-challenge/ {
            root /var/www/certbot;
    }

    location / {
        return 301 https://$host$request_uri;
    }
}
```

- 실행

```
chmod +x init-letsencrypt.sh // 권한 부여 후
./init-letsencrypt.sh // sh 실행 (배포를 시작할 루트 디렉토리에서
```

- app.conf

```
server {
    listen 80;
    listen [::]:80;

    server_name j10b106.p.ssafy.io;
    server_tokens off;

    location /.well-known/acme-challenge/ {
```

```
            allow all;
            root /var/www/certbot;
    }

    location / {
        return 301 https://$host$request_uri;
    }
}

server {
        listen 443 ssl;
        listen [::]:443 ssl;
        server_name j10b106.p.ssafy.io;

        ssl_certificate /etc/letsencrypt/live/j10b106.p.ssafy
        ssl_certificate_key /etc/letsencrypt/live/j10b106.p.s

        location / {
                proxy_pass http://43.203.217.53:3001;
                client_max_body_size 100M;
        }

        location /api {
                proxy_pass http://43.203.217.53:3002;
                client_max_body_size 100M;
        }

        location /swagger-ui {
                proxy_pass http://43.203.217.53:3002;
        }
        location /v3 {
                proxy_pass http://43.203.217.53:3002;
        }
        location /image {
                proxy_pass http://43.203.217.53:3003;
                client_max_body_size 100M;
        }
}
```

- docker-compose.yml

```
version: '3'

services:
  nginx:
    container_name: "nginx"
    image: nginx:1.15-alpine
    restart: unless-stopped
    volumes:
      - ./data/nginx/conf.d:/etc/nginx/conf.d
      - ./data/certbot/conf:/etc/letsencrypt
      - ./data/certbot/www:/var/www/certbot
    ports:
      - "80:80"
      - "443:443"

  front:
    container_name: "front"
    image: nginx:1.15-alpine
    restart: unless-stopped
    ports:
      - "3001:80"
    volumes:
      - ./front/html:/usr/share/nginx/html

  back:
    container_name: "back"
    image: azul/zulu-openjdk:17
    restart: unless-stopped
    ports:
      - "3002:8080"
    volumes:
      - ./back:/home
    command: java -jar /home/backend-0.0.1-SNAPSHOT.jar
```

- node 서버용 Dockerfile

```
FROM node:20.11.0-alpine

workdir /usr/src/app

COPY . .

RUN npm install
RUN npx tsc

expose 5000

workdir /usr/src/app/build
CMD ["node", "server.js"]
```

```
도커파일과 같은 위치에 node src 위치
이미지 빌드
docker build my-node-app .
```

- node, mysql, redis 실행

```
docker run --name mysql -p 3306:3306 -d mysql
docker run --name redis -p 6379:6379 -d redis
docker run --name node -p 3002:5000 -d my-node-app
```

- 빌드

```
프론트
noah 위치에 env파일 복사
npm i로 필요한 라이브러리 다운
npm run build로 빌드

백엔드
backend 내부 /src/main/resources에
yml, firebase/*.json 복사
backend위치에서 ./gradlew build
```

```
혹은 로컬에서 빌드 후 빌드파일을 옮겨도 무방,
빌드하기 전 필요 파일 위치는 동일
```

- 파일 위치

```
.
├── back (백엔드 빌드 파일 위치)
│   ├── backend-0.0.1-SNAPSHOT.jar
├── data
    └── nginx
        └── conf.d
            └── app.conf
├── docker-compose.yml
├── front
│   ├── html (프론트 빌드 파일들, index.html 위치)
│   │   ├── asset-manifest.json
│   │   ├── favicon.ico
│   │   ├── firebase-messaging-sw.js
│   │   ├── index.html
│   │   ├── manifest.json
│   │   ├── noah128.png
│   │   ├── noah192.png
│   │   ├── noah256.png
│   │   ├── noah512.png
│   │   ├── noah64.png
│   │   └── static
├── init-letsencrypt.sh
├── node
│   ├── Dockerfile
│   ├── src (노드 서버 필요 파일들)
└── script.sh
```

- 도커 컴포즈 시작

```
docker-compose up -d
```