# 개요

## 프로젝트 개요

저희는 이용자들이 편안하게 여행만을 즐기게 하기 위해 여행 계획 추천 서비스를 만들었습니다. 제주도에는 방문할 관광지가 정말 많습니다 저희 데이터베이스에 저장된 장소만 약 4천개가량 정도입니다. 그 많은 곳을 찾아보며 여행 경로를 계획하는건 많은 시간과 노력이 필요할 것입니다. 일상이 너무 바빠 여행을 계획할 시간조차 없는 사람들 혹은 여행 경로를 계획하는게 너무 귀찮은 사람들을 위해 놀멍쉬멍이 제주도에 도착할 때부터 제주도를 떠날 때까지 모든 경로를 계획해 드립니다.

## 프로젝트 사용 도구

형상 관리 : Gitlab

이슈 관리 : JIRA

커뮤니케이션 : Notion, Mattermost, Discord

디자인 : Figma

UCC : Movavi

CI/CD : Jenkins

## 개발환경

VS Code : 1.70

IntelliJ : -Build #IU-193.7288.26, built on May 6, 2020

-Runtime version: 11.0.6+8-b520.66 amd64

-VM: OpenJDK 64-Bit Server VM by JetBrains s.r.o

JAVA:

openjdk version "1.8.0\_192"

OpenJDK Runtime Environment (Zulu 8.33.0.1-win64) (build 1.8.0\_192-b01)

OpenJDK 64-Bit Server VM (Zulu 8.33.0.1-win64) (build 25.192-b01, mixed mode)

Node.js : 16.16.0

React: 18.2.0

Redux: 8.0.2

React-router-dom: 6.3.0

SERVER : AWS EC2 Ubuntu 20.04.4 LTS

DB : Mysql 8.0.22

## 외부 서비스

Google Maps API

KakaoNavi API

(1일 5000건 조회 제한)

Kakao Social API

## Gitignore 처리한 핵심 키들

# 빌드

## 환경변수 형태

application.properties

#DB설정

spring.datasource.url=jdbc:mysql://i7a609.p.ssafy.io:3306/A609?allowPublicKeyRetrieval=true&useSSL=false&characterEncoding=UTF-8&serverTimezone=UTC

spring.datasource.username= DB 접근 id

spring.datasource.password= DB 접근 패스워드

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

# JWT 시크릿키

jwt.secret= 시크릿키

#파일 저장경로 설정(서버)

resources.location:/

resources.uri\_path=/api/images

#용량제한

spring.servlet.multipart.maxFileSize=30MB

spring.servlet.multipart.maxRequestSize=30MB

# 카카오 ADMIN 키

kakao.admin = admin key

application-oauth.yml

spring:

security:

oauth2:

client:

registration:

kakao:

client-id: 클라이언트 id

redirect-uri: redirect

client-authentication-method: POST

authorization-grant-type: authorization\_code

scope: profile\_nickname, profile\_image, account\_email

client-name: Kakao

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

## 빌드하기

## 3. 빌드하기

1) FrontEnd:

npm i --force

npm run build

2) BackEnd:

./gradlew clean build

3) Socket Server:

npm i

npm run start

## 3. 배포하기

1) FrontEnd:

Dockerfile:

FROM node:16 as build-stage

WORKDIR /usr/src/app

COPY package\*.json ./

RUN npm install -g npm@8.16.0

RUN npm install --force

COPY . .

RUN npm run build

FROM nginx:stable-alpine as production-stage

RUN rm -rf /etc/nginx/conf.d

COPY conf /etc/nginx

COPY --from=build-stage /usr/src/app/build /usr/share/nginx/html

EXPOSE 80

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

Jenkins 설정:

docker stop frontend || true

docker rm -f frontend || true

docker rmi -f frontend || true

docker build --no-cache -t frontend .

docker run -itd -p 80:80 -p 443:443 -v /etc/letsencrypt/:/etc/letsencrypt/ --name frontend frontend

Nginx 설정:

server {

client\_max\_body\_size 100M;

listen 443 ssl;

listen [::]:443 ssl;

server\_name i7a609.p.ssafy.io; #도메인 주소

ssl\_certificate /etc/letsencrypt/live/i7a609.p.ssafy.io/fullchain.pem;

ssl\_certificate\_key /etc/letsencrypt/live/i7a609.p.ssafy.io/privkey.pem;

ssl\_protocols TLSv1 TLSv1.1 TLSv1.2;

ssl\_ciphers HIGH:!aNULL:!MD5;

location / {

root /usr/share/nginx/html;

index index.html index.htm;

try\_files $uri $uri/ /index.html;

}

location ~\* /(api|oauth2/authorization|/login/oauth2/code) {

proxy\_pass http://i7a609.p.ssafy.io:8080;

proxy\_http\_version 1.1;

proxy\_set\_header Connection "";

proxy\_set\_header Host $host;

proxy\_set\_header X-Real-IP $remote\_addr;

proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;

proxy\_set\_header X-Forwarded-Proto $scheme;

proxy\_set\_header X-Forwarded-Host $host;

proxy\_set\_header X-Forwarded-Port $server\_port;

}

location /socket.io{

proxy\_pass http://i7a609.p.ssafy.io:5000;

}

}

server {

listen 80 default\_server;

listen [::]:80 default\_server;

server\_name i7a609.p.ssafy.io;

location / {

return 301 https://$host$request\_uri;

}

}

2) BackEnd:

Dockerfile:

FROM openjdk:8-jre

EXPOSE 8081

COPY build/libs/backend-0.0.1-SNAPSHOT.jar app.jar

ENTRYPOINT ["java", "-jar", "app.jar"]

Jenkins 설정:

docker stop backend || true

docker rm -f backend || true

docker rmi -f backend || true

docker build --no-cache -t backend .

docker run -itd -p 8080:8080 -v /home/ubuntu/jeju/:/profile\_images/ -d --name=backend backend

3) Socket Server:

Dockerfile:

FROM node:16

WORKDIR /app

COPY package\*.json ./

RUN npm install

COPY . .

EXPOSE 5000

CMD ["npm", "run", "start"]

Jenkins 설정:

docker stop socket || true

docker rm -f socket || true

docker rmi -f socket || true

docker build --no-cache -t socket .

docker run -itd -p 5000:5000 --name socket socket