**포팅 매뉴얼**

****

**D207: Just Move It**

삼성SW청년아카데미 구미 캠퍼스 7기

공통 프로젝트

담당 컨설턴트: 유상진

팀원: 최영진(팀장), 강동관, 김효선, 정성우, 정종일, 조경수

**목차**

1. 프로젝트 개요 -------------------------------------------------------------------------1
2. 프로젝트 기술 스택 -------------------------------------------------------------------1
3. 빌드 상세내용 -------------------------------------------------------------------------2
4. 배포 특이사항 -------------------------------------------------------------------------2
5. 서버 환경 세팅 -----------------------------------------------------------------------3
6. 외부 서비스 ---------------------------------------------------------------------------8

**==============================================================**

1. **프로젝트 개요**

영화관에서 일할 당시 키오스크를 사용하기 어려워하는 분들이 많았습니다. 또한 코로나 팬데믹 이후 직접적인 키오스크 조작에 있어 거부감을 느끼는 분들도 많았습니다.

이에 조금 더 편리한 키오스크를 제작해 사용자들에게 보다 나은 경험을 주고자 Just Move It을 만들어보게 되었습니다.

1. **프로젝트 기술 스택**
2. 사용툴
   1. 이슈 관리: Jira
   2. 형상 관리: GitLab
   3. 커뮤니케이션: MatterMost, Notion
   4. 목업 관리: Figma
   5. 기타: miro (플로우 차트 생성)
3. 개발 환경
   1. OS: Window 10
   2. Server: AWS, NginX
   3. FE: HTML5, CSS, ES6, React 18.2.0
   4. BE: Spring Boot, JPA
   5. Android: Java, Retrofit2
   6. Database: MySQL
   7. AI: Teachable Machine, TensorFlow
   8. IoT: Rasbian
4. **빌드 상세내용**
5. BE

C:\S07P12D207\BE\CommonPJT> ./gradlew build

1. FE

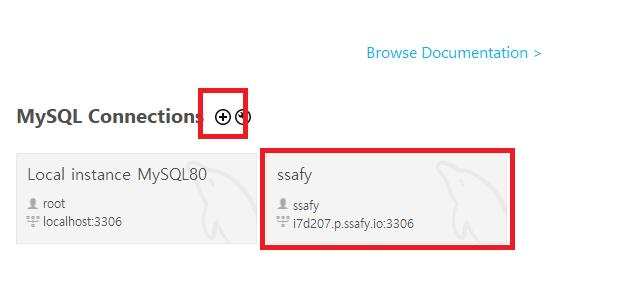
node\_modules를 위한 기본 install. 버전 문제로 강제 설치를 진행해야 합니다.

C:\S07P12D207\FE\newPjt> npm i —force

설치가 완료되면 프로젝트를 시작합니다.

C:\S07P12D207\FE\newPjt> npm run start

1. **배포 특이사항**
2. DB 계정
3. MySQL, Workbench 추가하기



1. EC2 계정 정보 등록

텍스트이(가) 표시된 사진

자동 생성된 설명

username: ssafy, password: 1234

1. **서버 환경 세팅**
2. EC2 세팅
   1. AWS EC2 접속을 위한 MobaXterm 설치
   2. 싸피에서 지급받은 pem 키를 이용하여 세션 등록
   3. EC2 서버에 Java 8,mySql workbench 8.0.29,nodeJs 18.7.0 설치
   4. https 인증을 위한 certbot letsencrypt 에서 ssl 인증
   5. nginx 설치 후 /etc/nginx/sites-available/defualt 파일을 수정

server{

**#HTTP 기본 경로인 80 포트로 접속**

listen 80 default\_server;

server\_name i7d207.p.ssafy.io;

if ($host = i7d207.p.ssafy.io) {

**#Nginx로 들어온 http 주소를 https주소로 변환시켜준다**

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

}

# managed by Certbot

}

server{

**#HTTPS 기본 포트인 443 포트로 접속하여 해당 경로에 맞게 리버스 프록싱 시켜준다**

listen 443 ssl;

listen [::]:443;

server\_name i7d207.p.ssafy.io;

proxy\_hide\_header Access-Control-Allow-Origin;

add\_header 'Access-Control-Allow-Origin' '\*';

**#웹 소켓 경로**

location /ws/socket {

proxy\_pass <http://localhost:8081>;

proxy\_set\_header Host $host;

proxy\_http\_version 1.1;

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 Upgrade $http\_upgrade;

proxy\_set\_header Origin "";

proxy\_set\_header Connection "Upgrade";

}

**#REACT 경로 설정**

location / {

proxy\_pass <http://localhost:3000>;

proxy\_set\_header Host $host;

proxy\_http\_version 1.1;

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 Upgrade $http\_upgrade;

proxy\_set\_header Connection "Upgrade";

}

**#Swagger 경로**

location ~ ^/(swagger|webjars|configuration|swagger-resources|v2|csrf) {

proxy\_pass <http://localhost:8081>;

proxy\_set\_header Host $host;

proxy\_http\_version 1.1;

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 Upgrade $http\_upgrade;

proxy\_set\_header Connection "Upgrade";

}

**#스프링 부트 (컨트롤러) 경로**

location /api {

proxy\_pass <http://localhost:8081>;

proxy\_set\_header Host $host;

proxy\_http\_version 1.1;

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 Upgrade $http\_upgrade;

proxy\_set\_header Connection "Upgrade";

}

ssl\_certificate /etc/letsencrypt/live/i7d207.p.ssafy.io/fullchain.pem;

ssl\_certificate\_key /etc/letsencrypt/live/i7d207.p.ssafy.io/privkey.pem;

}

1. 배포 환경 세팅
   1. Docker-compose 설정

version: '3'

services:

mysql:

image: mysql:latest

container\_name: mysql\_container

ports:

- 3306:3306 # HOST:CONTAINER

environment:

MYSQL\_ROOT\_PASSWORD: Ssafyd207

MYSQL\_PASSWORD: Ssafyd207

MYSQL\_USER: ssafy

MYSQL\_PASSWORD: Ssafyd207

command:

- --character-set-server=utf8mb4

- --collation-server=utf8mb4\_unicode\_ci

volumes:

- /etc/mysql/

networks:

- backend-network

application:

build:

context: ./app

dockerfile: Dockerfile

image: springboot

ports:

- "8081:8081"

depends\_on:

- mysql

container\_name: app\_container

environment:

SPRING\_DATASOURCE\_URL: jdbc:mysql://mysql:3306/ssafy?useSSL=false&serverTimezone=UTC&useLegacyDatetimeCode=false

SPRING\_DATASOURCE\_USERNAME: ssafy

SPRING\_DATASOURCE\_PASSWORD: Ssafyd207

networks:

- backend-network

react:

image: react

ports:

- "3000:3000"

container\_name: node\_container

volumes:

- /S07P12D207/FE/newPjt/:/var/www

networks:

- frontend-network

build:

context: /S07P12D207/FE/newPjt/

dockerfile: Dockerfile

networks:

backend-network:

frontend-network:

* + 1. react docker file

경로: / S07P12D207/FE/newPjt/Dockerfile

FROM node:18.2.0-alpine

WORKDIR /app

COPY package\*.json ./

COPY . ./

RUN npm install -f

EXPOSE 3000

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

* + 1. spring boot docker file

빌드된 자바 파일을 cp 명령어를 이용하여 /etc/nginx/app 폴더 안에 넣어주었다.

FROM openjdk:8-jdk-alpine

ARG JAR\_FILE=CommonPJT-0.0.1-SNAPSHOT.jar

COPY ${JAR\_FILE} app.jar

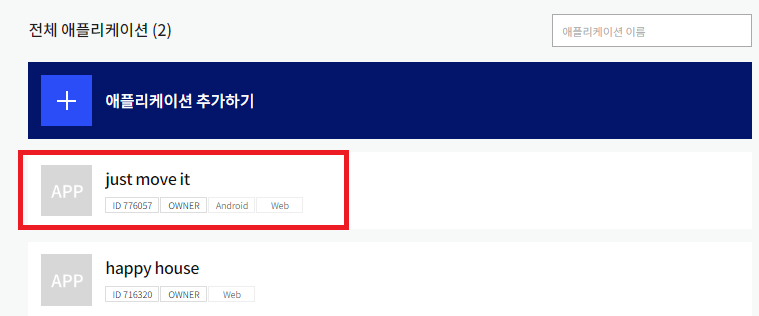
ENTRYPOINT ["java","-jar","/app.jar"]

* + 1. MySQL Docker

1. mySql 도커 이미지 다운로드
2. mySql 도커 컨테이너 생성 후 아이디 생성
3. docker-compose up -d 를 통하여 이미지를 컨테이너에 올림
4. **외부 서비스**
5. 카카오 로그인

안드로이드 앱 서비스의 사용자 로그인 구현, 사용자의 성별과 나이 수집을 목적으로 사용하였습니다.

1. 어플리케이션 추가



1. 플랫폼 등록
2. 개인정보 동의 항목

테이블이(가) 표시된 사진

자동 생성된 설명

1. 카카오 페이

앱 서비스 내 구매 기능을 구현하기 위해 사용하였습니다. 그러나 실제 결제를 위한 제휴 신청은 하지 않았기 때문에 테스트 결제로 남겨두었습니다.

1. 결제 준비
   1. Request Url

POST /v1/payment/ready HTTP/1.1

Host: kapi.kakao.com

Authorization: KakaoAK ${APP\_ADMIN\_KEY}

Content-type: application/x-www-form-urlencoded;charset=utf-8

* 1. Request Parameter

curl -v -X POST "<https://kapi.kakao.com/v1/payment/ready>" \\

-H "Authorization: KakaoAK ${APP\_ADMIN\_KEY}" \\

--data-urlencode "cid=TC0ONETIME" \\

--data-urlencode "partner\_order\_id=partner\_order\_id" \\

--data-urlencode "partner\_user\_id=partner\_user\_id" \\

--data-urlencode "item\_name=MOVIE\_TITLE" \\

--data-urlencode "quantity=1" \\

--data-urlencode "total\_amount=TICKET\_PRICE" \\

--data-urlencode "vat\_amount=200" \\

--data-urlencode "tax\_free\_amount=0" \\

--data-urlencode "approval\_url=https://developers.kakao.com/success" \\

--data-urlencode "fail\_url=https://developers.kakao.com/fail" \\

--data-urlencode "cancel\_url=https://developers.kakao.com/cancel"

* 1. Response

HTTP/1.1 200 OK

Content-type: application/json;charset=UTF-8

{

"tid": "T1234567890123456789",

"next\_redirect\_app\_url": "<https://mockup-pg-web.kakao.com/v1/xxxxxxxxxx/aInfo>",

"next\_redirect\_mobile\_url": "<https://mockup-pg-web.kakao.com/v1/xxxxxxxxxx/mInfo>",

"next\_redirect\_pc\_url": "<https://mockup-pg-web.kakao.com/v1/xxxxxxxxxx/info>",

"android\_app\_scheme": "kakaotalk://kakaopay/pg?url=https://mockup-pg-web.kakao.com/v1/xxxxxxxxxx/order",

"ios\_app\_scheme": "kakaotalk://kakaopay/pg?url=https://mockup-pg-web.kakao.com/v1/xxxxxxxxxx/order",

"created\_at": "2016-11-15T21:18:22"

}

1. 결제 요청

결제 준비 API 응답으로 받은 **next\_redirect\_mobile\_url**을 웹뷰로 띄워 진행.

1. 결제 승인
2. Request URL

POST /v1/payment/approve HTTP/1.1

Host: [kapi.kakao.com](<http://kapi.kakao.com/>)

Authorization: KakaoAK ${APP\_ADMIN\_KEY}

Content-type: application/x-www-form-urlencoded;charset=utf-8

1. Request Param

curl -v -X POST "[<https://kapi.kakao.com/v1/payment/approve>](<https://kapi.kakao.com/v1/payment/approve>)" \

-H "Authorization: KakaoAK ${APP\_ADMIN\_KEY}' \

--data-urlencode "cid=TC0ONETIME" \

--data-urlencode "tid=T1234567890123456789" \

--data-urlencode "partner\_order\_id=partner\_order\_id" \

--data-urlencode "partner\_user\_id=partner\_user\_id" \

--data-urlencode "pg\_token=xxxxxxxxxxxxxxxxxxxx"

1. Response

HTTP/1.1 200 OK

Content-type: application/json;charset=UTF-8

{

"aid": "A5678901234567890123",

"tid": "T1234567890123456789",

"cid": "TC0ONETIME",

"partner\_order\_id": "partner\_order\_id",

"partner\_user\_id": "partner\_user\_id",

"payment\_method\_type": "MONEY",

"item\_name": "MOVIE\_TITLE",

"quantity": 1,

"amount": {

"total": 2200,

"tax\_free": 0,

"vat": 200,

"point": 0,

"discount": 0,

"green\_deposit": 0

},

"created\_at": "2016-11-15T21:18:22",

"approved\_at": "2016-11-15T21:20:47"

}