|  |
| --- |
|  |

|  |
| --- |
| **Pocketing** |
| **포팅 메뉴얼** |
|  |
|  |
|  |
|  |
|  |

목차

[I. 개요 2](#_Toc96072459)

[1. 프로젝트 개요 2](#_Toc96072460)

[2. 프로젝트 사용 도구 2](#_Toc96072461)

[3. 개발환경 2](#_Toc96072462)

[4. 외부 서비스 2](#_Toc96072463)

[5. Gitgnore 처리한 핵심 키들 3](#_Toc96072464)

[II. 빌드 3](#_Toc96072465)

[1. 환경변수 형태 3](#_Toc96072466)

[2. 빌드하기 5](#_Toc96072467)

[3. 배포하기 5](#_Toc96072468)

# 개요

## 프로젝트 개요

요즘 K-POP 팬들은 CD가 아닌 굿즈 때문에 앨범을 산다! 요즘 K-POP 팬들 사이에서는 앨범의 구성품인 포토카드를 얻기 위해서 포토카드를 활발히 거래하지만 이에 특화된 서비스가 없어 불편함을 겪고 있습니다. 저희 포켓팅은 이러한 문제를 해결하고자 K-POP 팬들이 빠르고 쉽고 안전하게 포토카드 거래를 할 수 있도록 돕는 서비스입니다.

## 프로젝트 사용 도구

이슈 관리 : JIRA

형상 관리 : Gitlab

커뮤니케이션 : Notion, Mattermost

디자인 : Figma

UCC : 모바비

CI/CD : Jenkins

## 개발환경

VS Code

IntelliJ

JVM

Node.js

SERVER : AWS EC2 Ubuntu 20.04.3 LTS, AWS EC2 g4dn.xlarge(서울 리전)

DB : PostgreSQL, Redis, ChromaDB

## 외부 서비스

Kakao OAuth : application.yml에 해당 내용 있음

Twitter OAuth : application.yml에 해당 내용 있음

AWS S3

## Gitgnore 처리한 핵심 키들

React : .env

SpringBoot : application.yml, application-prod.yml

FastAPI : .env

# 빌드

## 환경변수 형태

1. SpringBoot application.yml

spring:

profiles:

active: local

server:

port: 8080

cloud:

aws:

s3:

bucket: pocketing-bucket

stack.auto: false

region.static: ap-northeast-2

credentials:

accessKey:

secretKey:

jwt:

secret:

expiration: 60

spring:

application:

name: pocketing-app

datasource:

url: jdbc:postgresql://localhost:5432/pocketing\_db?characterEncoding=UTF-8

username:

password:

driver-class-name: org.postgresql.Driver

jpa:

database: postgresql

hibernate:

ddl-auto: update

show-sql: true

properties:

hibernate:

format\_sql: true # 콘솔에 출력하는 SQL 쿼리 정렬

naming:

physical-strategy: org.springframework.boot.orm.jpa.hibernate.SpringPhysicalNamingStrategy

security:

oauth2:

client:

registration:

kakao:

client-id:

client-secret:

client-name: kakao

redirect-uri:

authorization-grant-type: authorization\_code

scope: account\_email

twitter:

client-id:

client-secret:

client-name: twitter

redirect-uri:

authorization-grant-type: authorization\_code

scope:

- users.read

- tweet.read

provider:

kakao:

authorization-uri:

token-uri:

user-info-uri:

user-name-attribute: id

twitter:

authorization-uri:

token-uri:

user-info-uri:

data:

redis:

host: localhost

port: 6379

1. React .env

# VITE\_API\_BASE\_URL=http://localhost:8080/api VITE\_API\_BASE\_URL=https://k12a406.p.ssafy.io/api VITE\_FASTAPI\_BASE\_URL=https://k12a406.p.ssafy.io/fastapi VITE\_KAKAO\_CLIENT\_ID=

VITE\_TWITTER\_CLIENT\_ID=

NAVER\_MAP\_KEY=

VITE\_STATE\_URL=http://localhost:5173 VITE\_STATE\_URL=https://k12a406.p.ssafy.io

1. ChatbotAI .env
2. YoloGeminiAI .env

AWS\_ACCESS\_KEY\_ID=

AWS\_SECRET\_ACCESS\_KEY=

AWS\_S3\_BUCKET\_NAME=pocketing-bucket

AWS\_REGION=ap-northeast-2

GEMINI\_API\_KEY=

## 빌드하기

1) Front

Jenkinsfile 실행

2) Back-spring

Jenkinsfile 실행

3) FastAPI

Jenkinsfile 실행

## 배포하기

# 전역 설정 (Global Configuration)

worker\_processes auto;

events {

    worker\_connections 1024;

}

# HTTP 설정

http {

    # MIME 타입 설정

    include /etc/nginx/mime.types;

    default\_type application/octet-stream;

    # 기본 서버 블록 (HTTP)

    server {

        listen 80;

        server\_name k12a406.p.ssafy.io;

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

    }

    # HTTPS 서버 블록

    server {

        listen 443 ssl;

        server\_name k12a406.p.ssafy.io;

        ssl\_certificate /etc/letsencrypt/live/k12a406.p.ssafy.io/fullchain.pem;

        ssl\_certificate\_key /etc/letsencrypt/live/k12a406.p.ssafy.io/privkey.pem;

        ssl\_protocols TLSv1.2 TLSv1.3;

        ssl\_ciphers HIGH:!aNULL:!MD5;

        location / {

            root /usr/share/nginx/html;

            index index.html;

            try\_files $uri /index.html;

        }

        # API 프록시 - Spring Boot 서버로 요청 전달

        location /api/ {

            proxy\_pass http://k12a406.p.ssafy.io:8081/api/; # Docker 네트워크의 Spring Boot 서버

            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 https;

        }

        # FastAPI 프록시 - FastAPI 서버로 요청 전달

        location /fastapi/ {

            client\_max\_body\_size 20M;

            proxy\_pass http://3.215.186.171:8000/fastapi/;

            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;

        }

        # 웹소켓 프록시

        location /ws/ {

            proxy\_pass http://k12a406.p.ssafy.io:8081/ws/;

            proxy\_http\_version 1.1;

            proxy\_set\_header Upgrade $http\_upgrade;

            proxy\_set\_header Connection "Upgrade";

        }

        # 챗봇 웹소켓 프록시

        location /chatbot/ws {

            proxy\_pass http://k12a406.p.ssafy.io:8000/chatbot/ws;

            proxy\_http\_version 1.1;

            proxy\_set\_header Upgrade $http\_upgrade;

            proxy\_set\_header Connection "Upgrade";

            # ← 여기부터 추가

            proxy\_set\_header   Host              $host;

            proxy\_buffering     off;

            proxy\_read\_timeout  300s;

            # ← 여기까지

        }

    }

}