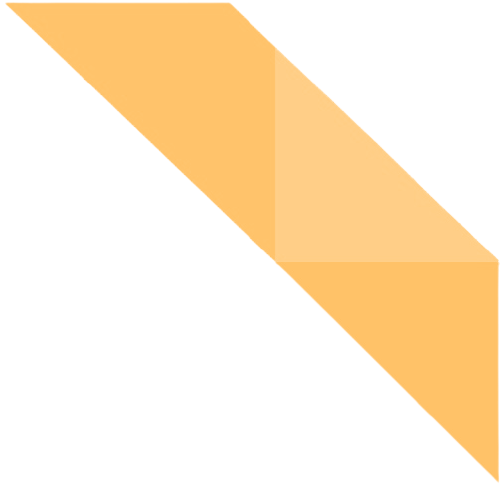
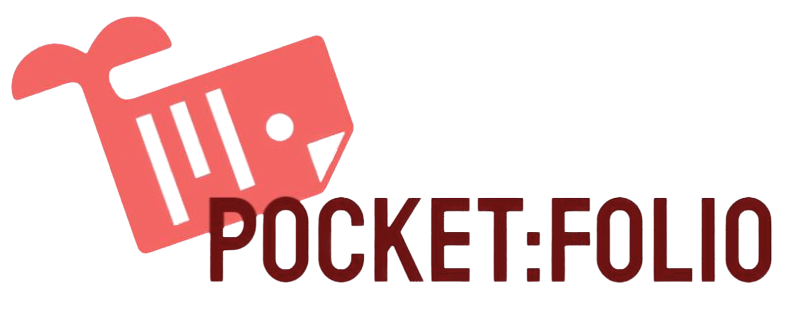
|  |
| --- |
| **Pocket:folio** |
| **포팅 매뉴얼** |





목차

[**I. 개요 2**](#_1fob9te)

[**1. 프로젝트 개요 2**](#_3znysh7)

[**2. 프로젝트 사용 도구 2**](#_2et92p0)

[**3. 개발환경 2**](#_tyjcwt)

[**4. 외부 서비스 3**](#_3dy6vkm)

**[II. 빌드 3](#_4d34og8)**

[**1. 환경변수 형태 3**](#_2s8eyo1)

[**2. 빌드하기**](#_17dp8vu) 5

[**3. 배포하기**](#_3rdcrjn)5

# 개요

## 프로젝트 개요

포켓폴리오는 Pocket과 Portfolio를 합쳐 만든 이름으로 자신의 포트폴리오를 3D 공간인 포켓에 전시하고 공유할 수 있는 서비스입니다.

여러 개의 포켓을 포트폴리오의 컨셉에 맞게 꾸밀 수 있어 사용자의 개성을 드러낼 수 있습니다.

각 카테고리별 좋아요 순, 유사도 기반 등의 방식으로 메인에 추천 포켓이 노출되기에 사용자들의 취향이나 좋은 아이디어를 볼 수 있으며 포켓 내의 방명록을 통해 유저 간 소통이 가능합니다.

## 프로젝트 사용 도구

* 이슈 관리: JIRA
* 형상 관리: Gitlab
* Open API 관리: Swagger, Postman
* 커뮤니케이션: Notion, Mattermost, Discord
* 디자인: Figma
* UCC: 모바비
* CI/CD: Docker, Jenkins

## 개발환경

* Backend: Spring-Boot, Django
* Frontend: React
* SERVER: AWS EC2 Ubuntu 20.04 LTS, Jenkins, NGINX
* DB: MySQL

## 외부 서비스

* 구글 OAuth (소셜 로그인)
* 카카오 OAuth (소셜 로그인)
* AWS S3 (파일 저장 서버)
* 번역 API (포켓 아이템 DB 저장 시 사용)

## Gitgnore 처리한 핵심 키들

* Spring Boot: application-env.properties
* Django: .env

# 빌드

## 환경변수 형태

* application-env.properties

|  |
| --- |
| # MySQL  spring.datasource.username=[유저이름]  spring.datasource.password=[비밀번호]  # SSL  server.ssl.enable=true  server.ssl.key-store=classpath:keystore.p12  server.ssl.key-store-type=PKCS12  server.ssl.key-store-password=[SSL 비밀번호]  security.require-ssl=true  # Google Login  spring.security.oauth2.client.registration.google.client-id=[Google Client ID]  spring.security.oauth2.client.registration.google.client-secret=[Google Clent PW]  spring.security.oauth2.client.registration.google.scope=email  # AWS  properties.file.aws-access-key=[AWS access key]  properties.file.aws-secret-key=[AWS secret key]  # Naver  naver.api.url=https://openapi.naver.com/v1/papago/n2mt  naver.client.id=[Naver Client ID]  naver.client.secret=[Naver Client Secret]  #Kakao  spring.security.oauth2.client.registration.kakao.client-id=[Kakao Client ID]  spring.security.oauth2.client.registration.kakao.client-secret=[Kakao Client PW]  spring.security.oauth2.client.registration.kakao.scope=account\_email  spring.security.oauth2.client.registration.kakao.client-name=Kakao  spring.security.oauth2.client.registration.kakao.authorization-grant-type=authorization\_code  spring.security.oauth2.client.registration.kakao.redirect-uri=https://k7e101.p.ssafy.io/api/login/oauth2/code/kakao  spring.security.oauth2.client.registration.kakao.client-authentication-method=POST  spring.security.oauth2.client.provider.kakao.authorization-uri=https://kauth.kakao.com/oauth/authorize  spring.security.oauth2.client.provider.kakao.token-uri=https://kauth.kakao.com/oauth/token  spring.security.oauth2.client.provider.kakao.user-info-uri=https://kapi.kakao.com/v2/user/me  spring.security.oauth2.client.provider.kakao.user-name-attribute=id |

* .env

|  |
| --- |
| # Django  DJANGO\_SECRET=[Django Secret Key]  # MySQL  MYSQL\_NAME = [DB이름]  MYSQL\_HOST = [HOST]  MYSQL\_PORT = [PORT]  MYSQL\_USER = [유저이름]  MYSQL\_PASS = [비밀번호] |

## 빌드하기

* Frontend

|  |
| --- |
| npm install  npm run build  npm start |

* Back-spring

|  |
| --- |
| Gradle 실행  Bootjar 실행 |

* Back-django

|  |
| --- |
| pip install -r requirements.txt  python manage.py runserver 8000 |

## 배포하기

* Nginx config

|  |
| --- |
| server {  listen 80;  server\_name k7e101.p.ssafy.io;  location / {  return 301 https://$host$request\_uri;  }  }  server {  listen 443 ssl;  server\_name k7e101.p.ssafy.io;  client\_max\_body\_size 50m;  access\_log /var/log/nginx/access.log;  error\_log /var/log/nginx/error.log;  ssl\_certificate /etc/letsencrypt/live/k7e101.p.ssafy.io/fullchain.pem;  ssl\_certificate\_key /etc/letsencrypt/live/k7e101.p.ssafy.io/privkey.pem;  ssl\_protocols TLSv1 TLSv1.1 TLSv1.2 SSLv3;  ssl\_ciphers ALL;  proxy\_redirect off;  charset utf-8;  proxy\_http\_version 1.1;  proxy\_set\_header Upgrade $http\_upgrade;  proxy\_set\_header Connection "upgrade";  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-Nginx-Proxy true;  location / {  proxy\_pass http://k7e101.p.ssafy.io:3000;  }  location /api {  proxy\_pass https://k7e101.p.ssafy.io:8081;  }  location /api/v2 {  proxy\_pass http://k7e101.p.ssafy.io:8000;  }  } |

Jenkins Pipeline

|  |
| --- |
| pipeline {  agent any  stages {  stage ('clone') {  steps {  git branch: 'dev', credentialsId: 'gitlab-soaeng', url: 'https://lab.ssafy.com/s07-final/S07P31E101.git'  }  }  stage ('build') {  parallel {  stage('backend-build') {  steps{  sh 'docker build -t back-spring ./backend/Pocketfolio'  sh 'docker build -t back-django ./backend/Recommend'  }  }  stage('frontend-build') {  steps{  sh 'docker build -t front ./frontend/pocketfolio'  }  }  }  }  stage('Deploy') {  steps{  // Backend - SpringBoot  sh 'docker stop back-spring | true'  sh 'docker rm back-spring | true'  sh 'docker run -d -p 8081:8081 -e TZ=Asia/Seoul --name back-spring -u root back-spring'  // Backend - Django  sh 'docker stop back-django | true'  sh 'docker rm back-django | true'  sh 'docker run -d -p 8000:8000 -e TZ=Asia/Seoul --name back-django -u root back-django'    // Frontend  sh 'docker stop front | true'  sh 'docker rm front | true'  sh 'docker run -d -p 3000:3000 --name front front'    // Nginx  sh 'docker stop nginx | true'  sh 'docker rm nginx | true'  sh 'docker run -d -p 80:80 -p 443:443 -v /etc/letsencrypt/:/etc/letsencrypt/ -v /home/ubuntu/nginx/conf/default.conf:/etc/nginx/conf.d/default.conf --name nginx nginx'  }  }  stage('Finish') {  steps{  sh 'docker images -qf dangling=true | xargs -I{} docker rmi {}'  }  }  }  } |