|  |
| --- |
|  |



|  |
| --- |
| **쥐금통** |
| **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)

[4. 서비스 이용 방법 7](#_Toc96072469)

[가) 카카오 페이 7](#_Toc96072470)

[나) Firebase 실시간 알림 8](#_Toc96072471)

[다) Google Cloud Platform 8](#_Toc96072472)

# 개요

## 프로젝트 개요

소비 습관 개선 및 절약 문화 전파를 위한 앱

## 프로젝트 사용 도구

이슈 관리 : JIRA

형상 관리 : Gitlab

커뮤니케이션 : Notion, Mattermost

디자인 : Figma

UCC : 모바비

CI/CD : Jenkins

## 개발환경

**Back**

VS Code : 1.80.0

IntelliJ : 17.0.7+7-b1000.6 amd64

SpringBoot : 2.7.14

Python : 3.7

JVM : 16.0.1 (스프링은 11로 빌드)

SERVER : AWS EC2 Ubuntu 20.04.3 LTS

DB : mysql

**Front**

Android gradle plugin : 7.3.1

Gradle : 7.4

Java : 11

Minsdk : 24

## 외부 서비스

Firebase FCM(Backend): resources폴더 밑에 firebase-key.json저장.

Kakao credentials(Backend): application-keys.yml파일에 작성

S3 credentials(Backend): application-keys.yml 파일에 작성 (과금 발생 가능)

## Gitgnore 처리한 핵심 키들

Spring : giggy-key.json(FCM 키 관련 정보), application-keys.yml(각종 키 보관)

(\src\main\resources에 위치)

# 빌드

## 환경변수 형태

.env.local :

// FIREBASE RTDB 프론트 키

VUE\_APP\_FIREBASE\_APIKEY=

VUE\_APP\_FIREBASE\_AUTHDOMAIN=

VUE\_APP\_FIREBASE\_DATABASEURL=

VUE\_APP\_FIREBASE\_PROJECTID=

VUE\_APP\_FIREBASE\_STORAGEBUCKET=

VUE\_APP\_FIREBASE\_MESSAGING\_SENDERID=

VUE\_APP\_FIREBASE\_APPID=

VUE\_APP\_FIREBASE\_MEASUREMENTID=

// YOUTUBE DATA API KEY

VUE\_APP\_YOUTUBE\_API\_KEY=

// 카카오 공유 서비스 키

VUE\_APP\_KAKAO\_API\_KEY=

.env

DB\_USER=DB호스트 유저 정보

DB\_PASS=DB 비밀번호

DB\_URL= DB 주소

application.yml (앱)

server:  
 port: 포트 번호  
  
spring:  
 datasource:  
 url: 주소   
 username: 유저  
 password: 비밀번호  
 driver-class-name: 드라이버  
 sql:  
 init:  
 encoding: UTF-8  
 jpa:  
 hibernate:  
 ddl-auto: update  
 properties.hibernate.format\_sql : true  
 show-sql : true  
  
 redis :  
 host : 호스트 주소  
 port : 포트 번호  
  
 profiles :  
 include : 사용할 profile  
  
 servlet:  
 multipart:  
 max-file-size: 100MB  
 max-request-size: 100MB

application-keys.yml (앱)

kakao:  
 credentials:  
 clientId: RESTAPI KEY  
  
server:  
 url:  
 bank: 통신할 은행 주소  
  
cloud:  
 aws:  
 s3:  
 bucket: 버킷 이름  
 stack.auto: false  
 region.static: ap-northeast-2  
 credentials:  
 accessKey: 액세스 키  
 secretKey: 비밀 키

application.yml (은행)

server:  
 port: 포트 번호  
  
spring:  
 datasource:  
 driver-class-name: 드라이버  
 url: db 주소

username: 유저  
 password: 비밀번호  
 jpa:  
 hibernate:  
 ddl-auto: update  
 show-sql: true

application.yml (랭킹)

server:  
 port: 포트 번호  
  
spring:  
 autoconfigure:  
 exclude: org.springframework.boot.autoconfigure.jdbc.DataSourceAutoConfiguration  
 redis:  
 host: 호스트 주소  
 port: 포트 번호

## 빌드하기

1) Front

APK 설치

2) Back-spring

Gradle 실행

Bootjar 실행

3) Back-flask

pip install -r requirements.txt

서버 구동 위치에서 python manage.py runserver

## 배포하기

Nginx 설정

upstream jenkins {

server 127.0.0.1:4000;

keepalive 32;

}

map $http\_upgrade $connection\_upgrade {

default upgrade;

'' close;

}

server {

listen 80 default\_server;

listen [::]:80 default\_server;

server\_name tupli.kr;

return 301 https://$server\_name$request\_uri;

}

server {

listen 443 ssl;

listen [::]:443 ssl;

server\_name tupli.kr;

ssl\_certificate /etc/letsencrypt/live/tupli.kr/fullchain.pem;

ssl\_certificate\_key /etc/letsencrypt/live/tupli.kr/privkey.pem;

root /var/www/html/dist;

index index.html;

location / {

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

}

location /api/v1 {

proxy\_pass http://localhost:8080;

proxy\_redirect off;

charset utf-8;

rewrite /api/v1/(.\*) /$1 break;

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 /api/v1/ws-stomp {

proxy\_pass http://localhost:8080;

rewrite /api/v1/(.\*) /$1 break;

proxy\_http\_version 1.1;

proxy\_set\_header Upgrade $http\_upgrade;

proxy\_set\_header Connection "Upgrade";

proxy\_set\_header Host $host;

}

location /flask {

proxy\_pass http://localhost:5000;

proxy\_redirect off;

charset utf-8;

rewrite /flask/(.\*) /$1 break;

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;

}

}

이후 sudo service nginx start

**Django**

Docker-compose.yml

version: '3.7'

services:

  django\_gunicorn:

    volumes:

      - static:/static

    build:

      context: .

    ports:

      - "8084:8084"

  nginx:

    build: ./nginx

    volumes:

      - static:/static

    ports:

      - "81:81"

    depends\_on:

      - django\_gunicorn

volumes:

  static:

Dockerfile

# Ubuntu 20.04 기반 이미지를 사용합니다.

FROM ubuntu:20.04

# 패키지 관리자 업데이트 및 Python 및 Java 11 설치

ARG DEBIAN\_FRONTEND=noninteractive

RUN sed -i 's/archive.ubuntu.com/mirror.kakao.com/g' /etc/apt/sources.list && \

    apt-get update && \

    apt-get install -y python3 python3-pip openjdk-11-jdk

RUN pip install --upgrade pip

COPY ./requirements.txt .

RUN pip install -r requirements.txt

COPY ./giggyanalysis /app

WORKDIR /app

COPY ./entrypoint.sh /

ENTRYPOINT [ "sh", "/entrypoint.sh" ]

Entrypoint.sh

#!/bin/sh

python manage.py migrate --no-input

python manage.py collectstatic --no-input

gunicorn giggyanalysis.wsgi:application --bind 0.0.0.0:8084

nginx.conf

worker\_process auto;

events{

}

http{

    server{

        listen 81;

        include mime.types;

        server\_name j9d208.p.ssafy.io;

        location /static/ {

            alias /data/static

        }

        location /media/{

            alias /data/media/

        }

        location / {

            proxy\_pass http://django\_container\_gunicorn:8084;

            proxy\_set\_header Host $host;

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

        }

    }

}

Nginx/Dockerfile

FROM nginx:1.19.0-alpine

COPY ./default.conf /etc/nginx/conf.d/default/conf

Nginx/default.conf

upstream django {

    server django\_gunicorn:8084

}

server{

    listen 80;

    location / {

        proxy\_pass http://django;

    }

    location /static/ {

        alias /static/;

    }

}

## 서비스 이용 방법

### 카카오 로그인

[백엔드]

* 카카오 key를 application-keys.yml에 분리하여 보관
* Rest-api키와 토큰을 사용하여 카카오 서버와 연동

[프론트]

* 카카오 개발자 사이트에서 어플리케이션 등록 후 네이티브 앱 키 보관
* 네이티브 앱키를 작성할 xml을 만든 후 AndroidMenifest에 <meta-data> 작성
* com.kakao.sdk.auth.AuthCodeHandlerActivity 생성

### Firebase 실시간 알림

1. Firebase 콘솔 프로젝트 만들기

2. 설정 → 프로젝트 설정 → 서비스 계정 → 새 비공개 키 생성

[백엔드]

* 파이어베이스 키를 resources폴더에 저장
* 파이어 베이스 관련 dependency 추가
* 서비스 코드 작성

[프론트]

* 파이어베이스 키를 프로젝트 상위에 저장 (App 폴더와 동급)
* 파이어 베이스 관련 dependency 추가
* 파이어베이스 서비스 작성 및 알림 권한 창 작성

### S3 설정

### 1] 버킷 생성

1. AWS Console > S3 > 버킷 > 버킷 만들기 클릭
2. 버킷 이름을 입력하고 엑세스 차단 설정을 해제한다.

### 2] 사용자 생성

1. AWS console > IAM > 엑세스 관리 > 사용자 > 사용자 추가 클릭  
2. 사용자 이름을 입력하고 다음을 클릭한다.  
3. 직접 정책 연결을 클릭하고, AmozonS3FullAccess 를 선택하고 다음을 클릭한다

4. 사용자 생성을 클릭하면 사용자가 생성된다.

.

### 3] 엑세스 키 생성

1. AWS Console > IAM > 엑세스 관리자 > 사용자 > 생성한 사용자 이름 클릭 > 보안 자격 증명 > 엑세스 키 만들기 클릭
2. 아무거나 클릭하고 다음을 클릭한다.
3. (클릭 하면 엑세스 키 사용사례와 대안을 하단에 띄워주는 기능만 하기때문에 뭘 골라도 상관없다.)
4. 태그를 입력하고 엑세스 키 만들기를 클릭한다.
5. 엑세스 키 생성 완료 화면에서 생성된 공개키와 비밀키를 확인할 수 있다.