

**Travel Maker (구미의 봄\_D202)**

**포팅 매뉴얼**

목차

**Ⅰ. 빌드 및 배포 …………..…………………………………………………………………………………………………**

1. 개발 환경

2. 환경 변수

3. 배포 특이사항

4. 주요 계정 및 프로퍼티 정의 파일

**Ⅱ. 외부 서비스 사용 …………….…………………………………………………………………………………………**

1. Google Geocoding API
2. Kakao 지역 검색 API
3. AWS S3
4. AWS RDS
5. Firebase

**Ⅲ. DB 덤프 데이터 …….……………………………………………………………………………………………………**

**## .sql 파일 별도 제출**

**Ⅳ. 시연 시나리오 ……………………………………….…………………………………………………………………..**

**Ⅰ. 빌드 및 배포**

**1. 개발 환경**

Android Studio: Flamingo 2022.2.1

IntelliJ: IntelliJ IDEA 2023.2

JVM: (Android, BackEnd 동일) 17

Server: AWS EC2 Ubuntu 20.04.6 LTS

DB: MySQL 8.0.35, Redis 7.2

**2. 환경 변수**

#[Android] build.gradle

buildTypes {  
 release {  
 minifyEnabled false  
 proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pro'  
 buildConfigField "String", "NAVER\_ID\_KEY", properties["NAVER\_ID\_KEY"]  
 }  
}

# [BackEnd] build.gradle

systemProperty 'JASYPT\_KEY', findProperty("JASYPT\_KEY")

**3. 배포 특이사항**

* Android: .apk 파일 (link)
* BackEnd

1. Develop-BackEnd 브랜치에 push or merge
2. Jenkins Webhook을 통해 자동 빌드
3. 빌드된 스프링부트 .jar 파일을 이미지화하여 도커 레포지토리에 push
4. 도커 레포지토리에서 해당 이미지를 pull 받아 도커 컨테이너에 띄워 자동 배포

**4. 주요 계정 및 프로퍼티 파일**

**[Android] local.properties**

**# API Key**

GOOGLE\_API\_KEY = AIzaSyBstc-jEnd0nGtnRnmCskuhRcfwfoJwDkc  
GOOGLE\_CLIENT\_ID = 694623643672aj4bce77jgj28jn4srt54cu3g52rb094.apps.googleusercontent.com  
KAKAO\_API\_KEY = KakaoAK df422ccf7ed496419b447dfe6dbd2ed7

**[BackEnd] application.yml**

# MySQL (배포)

datasource:  
 driver-class-name: com.mysql.cj.jdbc.Driver  
 username: admin  
 password: secretpasswordadmin

url: jdbc:mysql://travelmaker-database.cx6k2mygmj69.ap-northeast-2.rds.amazonaws.com:3306/travelmaker?serverTimezone=Asia/Seoul&characterEncoding=UTF-8

# Redis

redis:  
 host: i10d202.p.ssafy.io  
 port: 6379  
 repositories:  
 enabled: false

# AWS S3

s3:  
 bucket: d202-travelmaker-bucket  
credentials:  
 access-key: AKIA2X326X23ISSLUYNH  
 secret-key: 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

# Cool Sms

coolsms:  
 api:  
 key: NCSR1F5ZVLHSHCJF

secret: OD38EWHXKGIZZXJ3XKKPYTV9FOHNARXW

# Jwt Token

jwt:  
 secret: 908411b7cc7ad34802ef51f16845e7180fc9daa7df1b24f1e2ecc72a91972383

**Ⅱ. 외부 서비스 사용**

**1. Google Geocoding**

- Google Cloud Platform에서 프로젝트를 만든 뒤 API 키 발급

- 입력으로 주어진 위도, 경도로 해당 되는 주소명, 국가를 반환

- 영어권 해외사람들을 위한 동네인증, 또는 외국에 놀러간 한국인이 영어로 주소를 검색할 때 사용

**2. Kakao 지역 검색 API**

- Kakao Developer에서 프로젝트를 만든 뒤 Native API 키 발급

- 주소명, 건물명 등 주소를 나타낼 수 있는 키워드를 통해 해당 키워드에 부합하는 주소 최대 30개를 반환

- 한국권 사람들을 위한 동네인증, 또는 한국에 놀러온 외국인이 한국어로 주소를 검색할 때 사용

**3. CoolSms API**

- 사용자 회원가입 시 sms 인증을 위해 사용

- API 100회 호출 당 2000원 부과

- Coolsms를 통해 받은 인증번호를 redis에 저장하고 값을 꺼내와서 인증에 사용

**4. AWS S3**

- 사진, 영상과 같은 MultipartFile을 저장하고 불러오는 용도로 사용

- 파라미터에 MultipartFile를 넣어 기록 저장 API를 호출하면 해당 MultipartFile이 S3에 저장되고 접근 url을 반환

- 접근 url를 기록 엔티티의 imgUrl로 넣고 클라이언트(안드로이드)에 반환

- imgUrl을 glide 형식으로 띄워주면 해당 파일 보기 가능

**5. AWS RDS**

- 로컬 DB와 운영 DB를 분리하기 위해 사용

- AWS RDS를 운영 DB와 연결하고 운영 서버의 EC2에 설치하여 연동

- 로컬 서버에선 팀원 각자의 local DB를 이용하고, 운영 서버에선 RDS의 공용 DB 이용

**6. Firebase**

- 클라이언트(안드로이드)에 실시간 알림을 보내기 위해 사용

- [1] 클라이언트가 Firebase에서 (기기당) FCM Token을 발급받고 서버에 저장

- [2] 서버에 알림 요청이 들어오면 Firebase에서 Access Token을 발급받아 AccessToken과 FCM Token을 가지고 FCM Message 생성

- [3] FCM Message를 Firebase에 보내면, Firebase에서 클라이언트에 실시간 알림을 전송

**Ⅲ. DB 덤프 데이터 (.sql 파일)**

**Exec 폴더에 .sql 파일 별도 제출합니다.**

**Ⅳ. 시연 시나리오**

# 페르소나

- [인도] 여행 온 해외 여행객 **A**

- [한국] 거주 중인 로컬 한국인 **B**

# 시연 순서

1. 한국에 여행을 온 **A**는 개인 팜플렛을 생성한다.

2. **A**는 여행의 순간을 사진, 영상으로 찍으며 개인 팜플렛에 기록을 남긴다.

3. 우리나라에 여행 온 외국인과 놀고 싶은 **B**는 모임을 개최한다.

4. **A**는 현지인들만이 아는 로컬 명소들을 한국인들과 함께 즐기고 싶어 모임을 찾는다.

5. 모임을 찾던 **A**는 **B**가 개최한 모임에 신청 요청을 한다.

6. 신청 요청을 받은 **B**는 **A**의 프로필을 확인 후 수락을 한다.

7. **B**의 모임 글에 몇 명이 더 모이고 방장인 **B**는 모임을 시작하고 그룹 채팅방이 생성된다.

8. 그룹 채팅방에서 얘기를 나눈 멤버들은 모임을 가지고 그곳에서 순간의 기록들을 사진이나 영상으로 남긴다.

9. 모임이 종료되고 모임에서 남긴 추억들을 팜플렛에 저장해 한 눈에 볼 수 있다.

10. 다른 사람들의 여행 팜플렛을 보며 여행 경험을 공유할 수 있다.