

|  |
| --- |
| **DaeRo : 내맘대로, 발 닿는 대로 여행** |
| **포팅 메뉴얼** |
|  |
|  |
|  |
|  |
|  |

목차

[1. Gitlab 소스코드 클론 이후 빌드 및 배포할 수 있도록 정리 3](#_Toc111754981)

[A. 개발환경 3](#_Toc111754982)

[Frontend 3](#_Toc111754983)

[Backend 3](#_Toc111754984)

[AOS 3](#_Toc111754985)

[배포 3](#_Toc111754986)

[IDE 3](#_Toc111754987)

[B. 배포 시 특이사항 4](#_Toc111754988)

[I. SSH 설정 4](#_Toc111754989)

[II. UFW: 우분투 방화벽 설정 4](#_Toc111754990)

[III. tomcat 5](#_Toc111754991)

[IV. nginx 6](#_Toc111754992)

[C. 빌드 & 배포 7](#_Toc111754993)

[I. Frontend 7](#_Toc111754994)

[II. Backend 7](#_Toc111754995)

[III. Android 8](#_Toc111754996)

[2. 프로젝트에서 사용하는 외부 서비스 정보를 정리한 문서 10](#_Toc111754997)

[A. 외부 서비스 정보 10](#_Toc111754998)

[3. 시연 시나리오 11](#_Toc111754999)

[A. 선호도 기반 여행지 추천 11](#_Toc111755000)

[B. 사진과 글로 여행 기록 11](#_Toc111755001)

[C. 내 여행들을 모아보기 12](#_Toc111755002)

[D. 다른 유저들의 여행기 조회 13](#_Toc111755003)

# 1. Gitlab 소스코드 클론 이후 빌드 및 배포할 수 있도록 정리

## A. 개발환경

### Frontend

* HTML / CSS
* JavaScript
* Node.js
* Vue.js : Vue 2

### Backend

* Java : zulu 1.8.0\_342
* MariaDB : 10.6.8

### AOS

* Kotlin : 1.7.10
* SDK : 31
* Gradle : 7.3.3
* JDK : 1.8

### 배포

* Ubuntu : 20.04.4 LTS
* Tomcat : 9.0
* nginx : 1.18.0

### IDE

* Android Studio : Chipmuck 2021.2.1 Patch 1
* IntelliJ 2022.1.3
* VSCode

## B. 배포 시 특이사항

### I. SSH 설정

|  |
| --- |
| /etc/ssh/sshd\_config  (생략)  Port 1997  (생략) |

### II. UFW: 우분투 방화벽 설정

|  |
| --- |
| sudo ufw default deny  sudo ufw enable  sudo ufw allow from (DB 접근할 IP) to any port 13110  sudo ufw allow from 0.0.0.0/0 to any port 443  sudo ufw allow from 0.0.0.0/0 to any port 1997 |

### III. tomcat

/etc/tomcat9/server.xml

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <Server port="-1" shutdown="SHUTDOWN">  <Listener className="org.apache.catalina.startup.VersionLoggerListener" />  <Listener className="org.apache.catalina.core.AprLifecycleListener" SSLEngine="on" />  <Listener className="org.apache.catalina.core.JreMemoryLeakPreventionListener" />  <Listener className="org.apache.catalina.mbeans.GlobalResourcesLifecycleListener" />  <Listener className="org.apache.catalina.core.ThreadLocalLeakPreventionListener" />  <GlobalNamingResources>  <Resource name="UserDatabase" auth="Container"  type="org.apache.catalina.UserDatabase"  description="User database that can be updated and saved"  factory="org.apache.catalina.users.MemoryUserDatabaseFactory"  pathname="conf/tomcat-users.xml" />  </GlobalNamingResources>  <Service name="Catalina">  <Connector port="8080" protocol="HTTP/1.1"  connectionTimeout="20000"  redirectPort="8443" />  <Engine name="Catalina" defaultHost="localhost">  <Realm className="org.apache.catalina.realm.LockOutRealm">  <Realm className="org.apache.catalina.realm.UserDatabaseRealm"  resourceName="UserDatabase"/>  </Realm>  <Host name="localhost" appBase="webapps"  unpackWARs="true" autoDeploy="true">  <Context path="" docBase="back" reloadable="false" />  <Valve className="org.apache.catalina.valves.AccessLogValve" directory="logs"  prefix="localhost\_access\_log" suffix=".txt"  pattern="%h %l %u %t &quot;%r&quot; %s %b" />  </Host>  </Engine>  </Service>  </Server> |

### IV. nginx

/etc/nginx/sites-enabled/default

|  |
| --- |
| server {  listen 80 default\_server;  listen [::]:80 default\_server;  root /var/www/html;  index index.html index.htm index.nginx-debian.html;  server\_name \_;  location / {  try\_files $uri $uri/ =404;  }  }  server {  listen 443 ssl default\_server;  listen [::]:443 ssl default\_server;  ssl\_certificate /etc/letsencrypt/live/i7d110.p.ssafy.io/fullchain.pem;  ssl\_certificate\_key /etc/letsencrypt/live/i7d110.p.ssafy.io/privkey.pem;  root /var/www/vue/dist;  index index.html index.htm index.nginx-debian.html;  server\_name i7d110.p.ssafy.io;  location / {  try\_files $uri $uri/ =404;  }  location /api/ {  # First attempt to serve request as file, then  # as directory, then fall back to displaying a 404.  proxy\_pass http://localhost:8080/;  proxy\_set\_header X-Real-IP $remote\_addr;  proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;  proxy\_set\_header Host $http\_host;  proxy\_set\_header X-NginX-Proxy true;  proxy\_cache\_bypass $http\_secret\_header;  add\_header X-Cache-Status $upstream\_cache\_status;  expires -1;  }  } |

## C. 빌드 & 배포

* git clone https://lab.ssafy.com/s07-webmobile4-sub2/S07P12D110.git

### I. Frontend

|  |
| --- |
| npm i  npm run build |

### II. Backend

* **SpringBoot – application.properties**

1. DB 접속 설정

spring.datasource.driver-class-name=org.mariadb.jdbc.Driver  
spring.datasource.url=jdbc:mariadb://localhost:13110/daero?characterEncoding=UTF-8&serverTimezone=UTC  
spring.datasource.username=d110  
spring.datasource.password=gumi110

1. MyBatis Mapper 경로 설정

mybatis.mapper-locations=mapper/\*\*/\*.xml

1. Gmail SMTP 설정

spring.mail.host=smtp.gmail.com  
spring.mail.port=587  
spring.mail.username=(사용자 Gmail 계정)   
spring.mail.password=(사용자 App 비밀번호)  
spring.mail.properties.mail.smtp.auth=true  
spring.mail.properties.mail.smtp.starttls.enable=true

* **배포**

1. 상단 메뉴 Build 클릭
2. Build Artifact
3. build/libs/ 에 생성된 .war 파일을 SFTP로 /var/lib/tomcat9/webapps/에 전송
4. 전송한 파일 이름을 Tomcat server.xml의 Host > docBase 값에 따라 변경
5. Tomcat이 autoDeploy 설정에 따라 실행파일 배포

* **DB(MariaDB)**

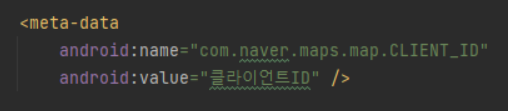
1. apt-get install mariadb-server
2. /etc/mysql/mariadb.conf.d/50-server.cnf
3. Port값 13110으로 변경
4. sudo mysql -u root
5. CREATE USER `d110`@\* IDENTIFIED BY 'gumi110';
6. GRANT ALL PRIVILEGES ON \*.\* TO `d110`@\* IDENTIFIED BY 'gumi110';
7. FLUSH PRIVILEGES;
8. CREATE SCHEMA daero;
9. Spring 프로젝트의 application.properties

spring.datasource.driver-class-name=org.mariadb.jdbc.Driver  
spring.datasource.url=jdbc:mariadb://localhost:13110/daero?characterEncoding=UTF-8&serverTimezone=UTC  
spring.datasource.username=d110  
spring.datasource.password=gumi110

### III. Android

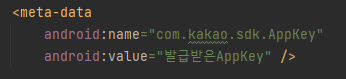
**빌드**

* NaverMapClientId 추가

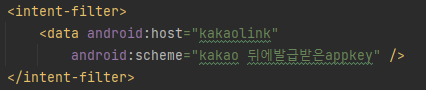


AndroidManifest.xml 파일에서 클라이언트ID 부분에 추가

* 카카오톡 공유하기 AppKey 추가



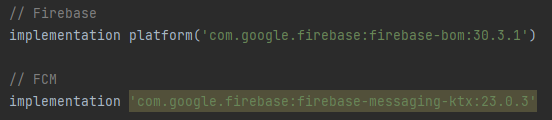
AndroidManifest.xml 파일에서 발급받은AppKey 부분에 추가



AndroidManifest.xml 파일에서 intent-filter에 AppKey 부분에 추가

* FCM 설정

1. 파이어베이스 콘솔에서 프로젝트 생성
2. 좌측 사이드바에서 Cloud Messaging 클릭
3. Android 앱 등록
4. google-service.json 다운로드 후 Project/app 밑에 복사



App 수준 gradle에 dependency 추가

* Build 버튼 클릭

**배포**

1. 상단 메뉴 Build -> Generate Signed Bundle/APK 클릭
2. Android App Bundle 체크 -> Next 클릭
3. Key store path 선택, key store password, alias, password 작성 후 Next 클릭
4. Release 체크 -> Finish 클릭
5. app/release 경로에 abb 파일 생성 완료

# ****2. 프로젝트에서 사용하는 외부 서비스 정보를 정리한 문서****

## A. 외부 서비스 정보

**Naver Map API** – 네이버 지도 API

URL:  [네이버 지도 API 관련 문서 링크](https://navermaps.github.io/android-map-sdk/guide-ko/0.html)

FCM – Firebase 클라우드 메시징

URL: [FCM API 관련 문서 링크](https://firebase.google.com/docs/cloud-messaging/android/client?hl=ko)

**Kakao 공유 API** – 카카오톡 공유하기

URL:  [Kakao Talk 공유하기 API 관련 문서 링크](https://developers.kakao.com/docs/latest/ko/message/android-link)

Tour API – 한국관광공사 관광지 정보

URL:  [한국관광공사 Tour API 4.0](https://api.visitkorea.or.kr/)

Gmail SMTP – 구글 SMTP

URL: [Gmail SMTP 도움말 문서](https://support.google.com/a/answer/176600?hl=ko#zippy=%2Cgmail-smtp-%EC%84%9C%EB%B2%84-%EC%82%AC%EC%9A%A9%ED%95%98%EA%B8%B0)

# 3. 시연 시나리오

## A. 선호도 기반 여행지 추천

- **트립 화면**

앱을 실행시키면 가장 먼저 홈 화면이 보인다.

앱 하단의 키워드 선택하기를 눌러 산과 바다 chip을 선택한다

선택 후 바텀시트를 아래에서 위로 밀면 여행지 화면으로 전환한다.

**- 키워드 추천**

키워드에 따라 여행지를 설명해주는 화면이 나온다.

다시 추천받기, 그만하기 버튼이 있다.

**- 키워드 추천 (대구경북, 먹거리)**

앱 하단의 키워드 선택하기를 눌러 대구경북, 먹거리chip을 다시 선택한다

선택 후 바텀시트를 아래에서 위로 밀면 여행지 화면으로 전환한다.

## B. 사진과 글로 여행 기록

**- 여행지 정보 화면**

키워드에 따라 여행지를 설명해주는 화면이 나온다.

다시 추천받기, 그만하기 버튼이 있다

**- 여행하기 화면**

여행지 정보(이름, 사진, 주소)를 제공하고, 길찾기 버튼과 흔들어서 인증하기 기능이 있고 여행 그만하기와 다른여행지 추천 버튼이 있다

**- 여행 인증 화면**

여행지 정보(이름, 사진, 주소)를 제공하고, 트립스탬프 만들기 버튼과 지금까지 여행지 목록, 여행 그만하기 버튼이 있다

**- 트립스탬프 만들기 화면**

갤러리와 카메라를 선택할 수 있고, 카메라 선택 시 카메라로 사진을 찍어 트립스탬프를 생성한다.

여행만족도를 이미지 버튼을 통해 선택한다.

트림스탬프 저장하기 버튼과 다시 찍기 버튼이 있다

**- 여행 그만하기**

다음 여행지를 추천 받는 버튼이 있다.

주변 여행지 목록과 지금까지 여행지 목록을 제공한다.

여행 그만하기 버튼을 눌러 여행 기록하기와 여행 기록하지 않고 그만하기 버튼을 누를 수 있다.

**- 게시글 작성**

일자별로 트립스탬프 리스트를 제공하고 간단한 코멘트를 남길 수 있다.

다음 버튼을 눌러 다음 일차 화면으로 넘어가거나, 여행 경비 및 여행 만족도 화면으로 전환한다.

여행 경비 및 여행 만족도 화면에서는 여행 경비를 작성할 수 있고 +버튼을 통해 더 많은 경비를 작성할 수 있다.

여행 만족도를 레이팅바로 제공하며 5점이 만점이다.

다음 버튼을 눌러 썸네일 선택 화면으로 이동한다.

썸네일 선택 화면에서는 해당 게시글의 제목과 내용을 입력하고 썸네일을 선택한다.

여행 기록하기 버튼을 통해 게시글을 등록한다.

**- 작성 완료된 게시글 상세 화면**

작성 완료된 여행 기록은 안드로이드 Room DB에서 사라지고 서버에 저장된다.  
저장된 게시글은 마이페이지의 앨범을 통해 확인할 수 있다.

## C. 내 여행들을 모아보기

**- 마이페이지 화면**

마이 페이지 화면에서는 앨범과 내 여정조회를 통해 사용자의 여행 기록을 한 번에 볼 수 있다  
앨범에서는 사용자가 기록한 여행 일지들을 모아 볼 수 있고, 내 여정 조회에서는 지도를 통해 사용자가 다녀온 여행지의 위치를 파악할 수 있다.

## D. 다른 유저들의 여행기 조회

**- 홈화면**

홈 화면으로 이동하면 다른 사람들의 여행기를 목록으로 볼 수 있다.

**- 게시물 상세페이지**

다른 유저의 상세 페이지로 이동 시, 해당 유저가 기록한 여정을 조회할 수 있다.  
페이지 상단에는 유저가 다닌 여행지의 위치를 표시하는 지도 화면을 볼 수 있고, 지도 아래에는 유저의 여행 기록, 좋아요 수, 댓글 수 등을 조회해 볼 수 있다.  
상세 여정에서 사진을 누르면 해당 사진을 찍은 장소와 날짜 등 상세 내용을 볼 수 있다.  
좋아요 버튼을 누를 경우, 좋아요 버튼이 변경되는 것을 확인할 수 있다.

**- 컬렉션으로 자연스럽게 이동**

게시물 상세 페이지에서 좋아요 버튼을 누른 게시글이 컬렉션 목록에 추가되었음을 컬렉션 페이지에서 확인할 수 있다.  
게시물을 누르면 게시물 상세 페이지로 이동이 가능하다.

**- 검색**

검색창에 검색하고자 하는 키워드를 입력한 후, 검색 버튼(혹은 엔터 버튼)을 누른다.  
검색 결과가 유저명, 게시글 상세 내용, 여행지 명 크게 3가지 분류로 나타난다.  
검색 결과의 게시글을 누르면 해당 게시글의 상세 페이지로 이동하고, 유저명을 누르면 해당 유저의 페이지로 이동한다.

**- 다른 유저 페이지로 이동**

다른 유저의 여행 앨범과 여정을 조회할 수 있고, 다른 유저의 여행 상세 페이지로 이동할 수 있다.  
다른 유저 닉네임 옆에 있는 팔로우 버튼을 통해 해당 유저를 팔로우 할 수 있다.