😢

포팅 메뉴얼

생성자	
생성 일시	@2023년 2월 16일 오전 10:51
최종 편집자	
최종 편집 일시	@2023년 2월 17일 오전 10:59
태그	

시작
개발 환경
Main Server
    A. Docker 및 Jenkins 설정
        - AWS EC2 내 Docker 설치
        - AWS EC2 내 Jenkins 설치
        - Jenkins 설정 및 GitLab 연동
        - Jenkins 플러그인 설치
        - Jenkins SSH 설정
        - Jenkins와 GitLab 프로젝트 연결
        - 빌드 후 조치 Shell Script
        - 설정 후 프로젝트 구조
    B. MySQL 설정
        - MySQL 패키지 설치 및 서버 실행
        - MySQL Workbench에 원격 호스트의 MySQL 연결하기
    C. Nginx & SSL 설정
    D. Mapbox API Key 발급
    F. Application 시작
Android Client
    A. gradle.properties 설정
        - Mapbox API Key 발급
        - Kakao API 발급
    B. 실행
Admin Client
    A. 패키지 설치
    B. 로컬 호스트 시작
사용자 시나리오
    A. 어플리케이션 설치
    B. 어플리케이션 실행
    C. 회원가입 및 로그인
    D. 경로 그리기 및 경로 목록
    E. 경로 기록하기
    F. 마이페이지 및 기록 상세보기
    G. 기록 공유하기
    H. 설정 및 이용약관

시작

프로젝트 실행을 위해 원격 저장소를 클론합니다.

```
$ git clone https://lab.ssafy.com/s08-webmobile4-sub2/S08P12A401.git
```

개발 환경

Android

- Android Gradle Plugin Version: `7.2.1`
- Kotlin: `1.6.10`
- Gradle Version: `7.4`
- Hilt Version: `2.44`
- navigation Version: `2.5.3`

FrontEnd

- Node.js: `18.13.0`
- npm: `8.19.3`

DevOps

- Docker: `20.10.23`
- Jenkins: `2.375.2`
- Nginx: `nginx/1.18.0`

Server

- AWS EC2: `ubuntu 20.04`
- IntelliJ: `IDEA 2022.3.1`
- SpringBoot: `2.7.7`
- JDK: `OpenJDK 11.0.17`

Database

- MySQL: `8.0.32-0ubuntu0.20.04.2`

관리

- GitLab
- Jira

Main Server

A. Docker 및 Jenkins 설정

- AWS EC2 내 Docker 설치

1. 업데이트 및 HTTP 패키지 설치

```
$ sudo apt update
$ sudo apt-get install -y ca-certificates \
    curl \
    software-properties-common \
    apt-transport-https \
    gnupg \
    lsb-release
```

2. GPG 키 및 저장소 추가

```
$ sudo mkdir -p /etc/apt/keyrings
$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg

$ echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu \
  $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

3. 도커 엔진 설치

```
$ sudo apt update
$ sudo apt install docker-ce docker-ce-cli containerd.io
```

4. 도커 설치 확인하기

```
$ sudo docker ps
```

- AWS EC2 내 Jenkins 설치

1. Jenkins 이미지 다운로드

```
$ docker pull jenkins/jenkins:lts
```

2. Jenkins 컨테이너 설치

```
sudo docker run -d -p 8080:8080 -v /jenkins:/var/jenkins_home --name jenkins -u root jenkins/jenkins:lts
```

3. 젠킨스 도커 컨테이너에 bash 접속

```
docker exec -it jenkins-docker bash
```

- Jenkins 설정 및 GitLab 연동

1. 9090 포트로 접근해 젠킨스 설정

```
http://localhost:9090
```

2. GitLab 연동

   Jenkins 관리 → 시스템 설정 → GitLab

   - Connection name: 연결 이름 지정
   - Gitlab host URL: Gitlab 서버 주소 기입
   - Add Credentials: GitLab에서 API token 생성 후 Credentials 생성

   설정 후 Test Connection을 눌러 정상 연결 확인

- Jenkins 플러그인 설치

프로젝트를 정상적으로 빌드하기 위해서는 다음의 플러그인을 추가로 설치해야 한다.

- NodeJS `1.6.0`
- GitLab `1.7.6`
- JAXB `2.3.8-1`

- Jenkins SSH 설정

1. SSH Key 설정

   Jenkins 관리 → 시스템 설정 → Publish over SSH

   - Key: GitLab에서 SSH Key를 발급 받아 입력
   - SSH Servers
     - Name: Host 주소 입력

- Hostname: Host 이름 입력
- Username: EC2 서버 사용자 이름 입력

- Jenkins와 GitLab 프로젝트 연결

1. Project → 구성 → Configure → 소스코드 관리
   - Git

     Repository URL: GitLab URL

     Credentials: Add → Jenkins → Add Credentials → Kind: Username with password
     - Jenkins와 연동할 GitLab의 Maintainer 이상의 권한을 가지고 있는 사용자 계정으로 로그인

     Branches to build: Build를 실행할 Branch 이름 입력

2. 빌드 유발
   - Build when a change is pushed to Gitlab, GitLab webhook URL:

     Push Events, Opened Merge Request Events 클릭

3. 빌드 환경
   - Provide Node & npm bin/ folder to PATH
     - NodeJS Installation: nodejs-18.13.0

4. Builds Steps
   - Execute shell

     ```
     cd artwalk_backend
     chmod +x gradlew
     ./gradlew clean build
     ```

5. 빌드 후 조치
   - SSH Publishers
     - Name: [호스트 주소]
   - Transfer Set
     - Source files

       ```
       artwalk_backend/build/libs/*.jar
       ```

     - Remove prefix

       ```
       artwalk_backend/build/libs
       ```

     - Remote directory

       ```
       deploy
       ```

     - Exec command

       ```
       sh /home/ubuntu/deploy/deploy_script.sh
       ```

- 빌드 후 조치 Shell Script

1. 빌드 후 자동 배포를 위해 폴더 이동 후 `deploy_script.sh` 작성

```
$ cd /home/ubuntu{혹은 사용자 이름}/deploy/
$ vi deploy_script.sh
```

- deploy_script.sh

```
echo "PID Check..."

// 현재 실행중인 프로세스 확인
CURRENT_PID=$(ps -ef | grep java | grep artwalk_backend* | awk '{print $2}')
NOW=$(date +%y%m%d_%H%M%S)

echo "Running PID: {$CURRENT_PID}"

// 실행 중인 프로세스를 죽이고 새로 빌드 된 프로젝트 실행
if [ -z "$CURRENT_PID" ] ; then
    echo "Project is not running"
else
    echo "kill {$CURRENT_PID}"
    kill -9 $CURRENT_PID
   sleep 10
fi

echo "Deploy Project...."

// jar 파일 실행
nohup java -jar ~/deploy/artwalk_backend-0.0.1-SNAPSHOT.jar > ~/logs/$NOW.log  2> ~/logs/error_$NOW.log &

echo "Done"
```

- 설정 후 프로젝트 구조

B. MySQL 설정

- MySQL 패키지 설치 및 서버 실행

1. MySQL 패키지 설치

```
$ apt-get install mysql-server mysql-client
```

2. 설치 버전 확인

```
$ mysql --version
```

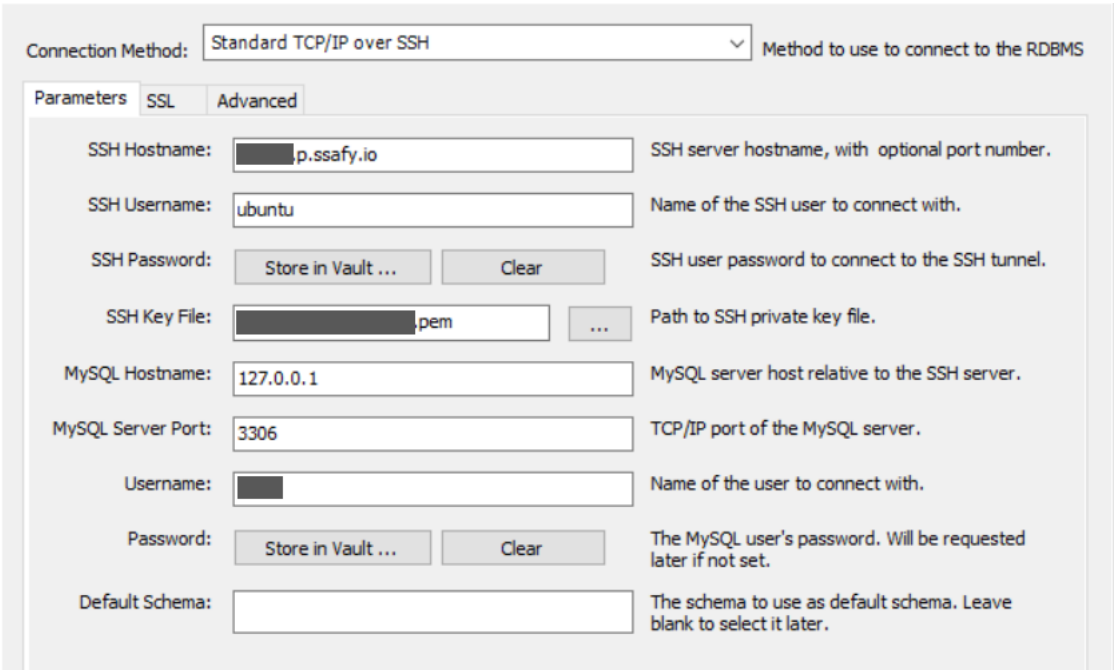
3. MySQL 서버 실행

```
$ service mysql start
```

- MySQL Workbench에 원격 호스트의 MySQL 연결하기

1. MySQL Workbench 8.0 설치
2. MySQL Connections 설정

3. MySQL Connections에 새로운 연결 생성 후 원격 호스트 접속 SSH Key 지정



4. SQL 스크립트 실행
   - File → Open SQL Script로 스크립트 불러오기

5. 스크립트 실행 후 데이터베이스 생성

   database 폴더 내 artwalk.sql 실행 후 `artwalk` 데이터베이스 생성

C. Nginx & SSL 설정

1. Nginx 설치

```
$ sudo apt-get install nginx
```

2. Nginx 설치 확인

```
$ sudo nginx -v
```

3. Let’s Encrypt 설치

```
$ sudo apt-get install letsencrypt
```

4. 인증서 적용 및 .pem 키 발급

```
$ sudo letsencrypt certonly --standalone -d [도메인]
```

5. 발급 경로 확인

```
$ cd /etc/letsencrypt/live/[도메인]
```

6. Nginx 설정 파일 작성

```
$ cd /etc/nginx/sites-available
$ sudo vi [파일명].conf
```

- artwalk_backend.conf

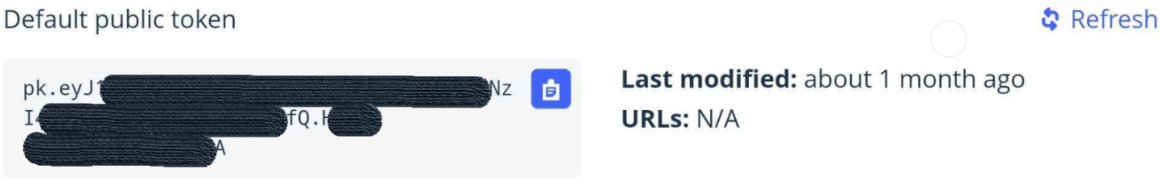
  프로젝트에 사용된 설정 파일

```
server {
  listen 443 ssl;
  ssl_certificate /etc/letsencrypt/live/[도메인]/fullchain.pem;
  ssl_certificate_key /etc/letsencrypt/live/[도메인]/privkey.pem;

  server_name i8a401.p.ssafy.io;

  location / {
    proxy_pass http://localhost:8080;
    proxy_hide_header Access-Control-Allow-Origin;
    add_header 'Access-Control-Allow-Origin' '*' always;
  }

  include /etc/letsencrypt/options-ssl-nginx.conf;
  ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem;
}

server {
  return 301 https://$host$request_uri;
  # managed by Certbot

  listen 80;
  server_name [도메인];
  return 404;
}
```

D. Mapbox API Key 발급

1. mapbox 홈페이지 접속

2. Sign up 버튼을 눌러 회원가입 후 로그인까지 진행
3. Go to account 버튼을 눌러 회원 페이지로 이동
4. Access tokens 섹션에서 Create a token 버튼 클릭
5. Default public token 섹션의 API Key를 복사하여 사용



F. Application 시작

1. **프로젝트 내 지도 사용을 위한 mapbox API KEY 발급**
   - **D. Mapbox API Key 발급 페이지**를 참조해 KEY를 발급한다.

2. **application.properties 수정**

```
$ cd artwalk_backend
$ cd src/main/resources
$ vi application.properties
```

   - [ ] 내부의 설정은 해당하는 값으로 변경

```
server.port=8080

spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.url=jdbc:mysql://[localhost:port]/artwalk?serverTimezone=UTC&characterEncoding=UTF-8&useLegacyDatetimeCode=false
spring.datasource.username=[username]
spring.datasource.password=[password]

# mysql
spring.jpa.database=mysql

# logging level
logging.level.org.hibernate=info

# hibernate
spring.jpa.hibernate.ddl-auto=update
spring.jpa.properties.hibernate.show_sql=true
spring.jpa.properties.hibernate.format_sql=true
spring.jpa.properties.hibernate.use_sql_comments=true

# jwt
jwt.secret=[jwt 암호화에 사용할 임의의 문자열]

# save file path
file.path=artwalk_resource/

# mapbox api
mapbox.api.url=https://api.mapbox.com/styles/v1/mapbox/streets-v11/static/
mapbox.api.key=[mapbox API KEY]

# swagger
spring.mvc.pathmatch.matching-strategy=ant_path_matcher

#spring.mvc.static-path-pattern=/resources/**

# thymeleaf
spring.thymeleaf.prefix=classpath:templates/
spring.thymeleaf.suffix=.html
```

```
spring.thymeleaf.cache=false
```

- **application.properties**
  - spring.datasource.url =MySQL 주소와 포트 지정
  - datasource.username, datasource.password = MySQL 사용자 계정과 비밀번호 지정
  - jwt.secret = JWT 암호화에 사용할 임의의 문자열 입력
  - mapbox.api.key = Mapbox 지도를 호출하기 위한 API 키 입력

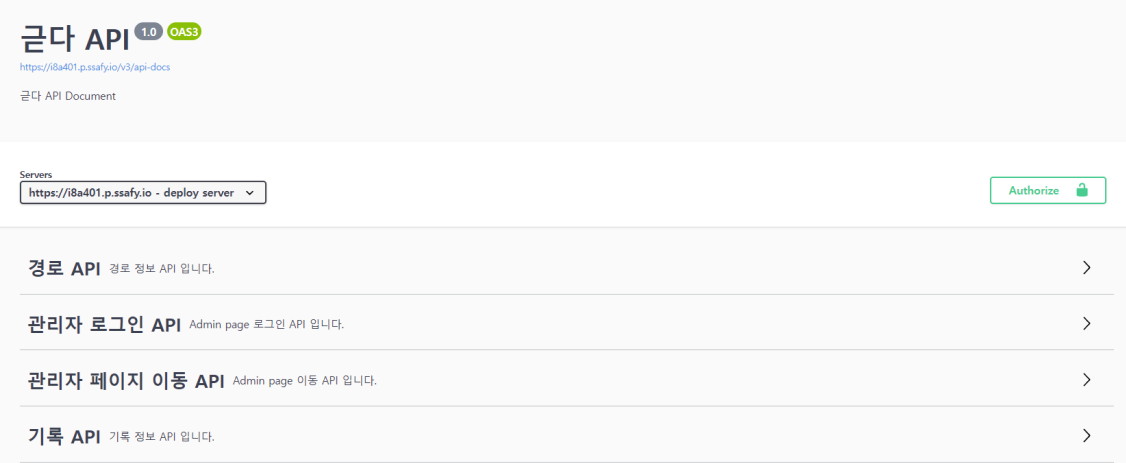
3. **프로젝트 실행**

```
$ cd artwalk_backend
$ chmod +x gradlew
$ ./gradlew clean build
```

4. **Swagger-UI에서 API 테스트**

```
http://localhost:8080/swagger-ui/index.html
```

- **Swagger-UI**



Android Client

A. gradle.properties 설정

gradle.properties 파일에 아래 내용을 복사, 붙여넣기 한 다음 이후 내용을 따라하시면 됩니다.

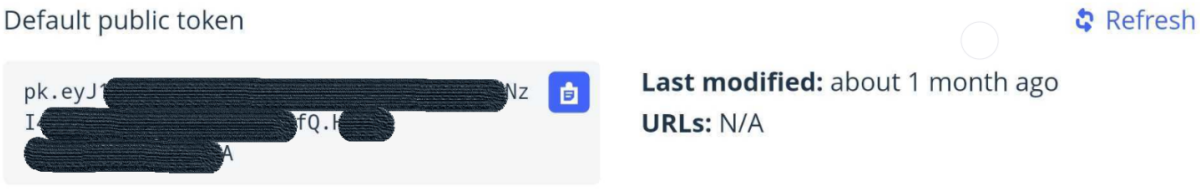
```
org.gradle.jvmargs=-Xmx2048m -Dfile.encoding=UTF-8
android.useAndroidX=true
kotlin.code.style=official
android.nonTransitiveRClass=true

MAPBOX_DOWNLOADS_TOKEN="붙여넣기"
MAPBOX_PUBLIC_ACCESS_TOKEN="붙여넣기"

KAKAO_NATIVE_KEY="붙여넣기"
KAKAO_API_KEY=붙여넣기
```

```
MAPBOX_DIRECTIONS_URL="https://api.mapbox.com/directions/v5/"
ROUTE_BASE_URL="https://[서버 도메인 주소]/route/"
IMAGE_BASE_URL="https://[서버 도메인 주소]/route/thumb/"
IMAGE_BASE_URL_RECORD="https://[서버 도메인 주소]/record/thumb/"
PROFILE_URL="https://[서버 도메인 주소]/user/info/profile"
LOGIN_BASE_URL="https://[서버 도메인 주소]/"
RECORD_BASE_URL="https://[서버 도메인 주소]/record/"
SHARE_BASE_URL="https://[서버 도메인 주소]/share/"
SHARING_BASE_URL="https://[서버 도메인 주소]/sharing/edit/"
```

- Mapbox API Key 발급

1. mapbox 홈페이지 접속
2. Sign up 버튼을 눌러 회원가입 후 로그인까지 진행
3. Go to account 버튼을 눌러 회원 페이지로 이동
4. Access tokens 섹션에서 Create a token 버튼 클릭
5. Default public token 섹션의 API Key를 복사해 아래와 같이 붙여넣는다.

```
MAPBOX_PUBLIC_ACCESS_TOKEN="여기에 붙여넣기"
```



6. token page에서 create token 버튼을 누르고 `Downloads:Read` 를 체크해 새로운 토큰을 발급한다
7. 발급받은 토큰을 복사해 아래와 같이 붙여넣는다.

```
MAPBOX_DOWNLOADS_TOKEN=여기에 붙여넣기
```

- Kakao API 발급

1. 카카오 Developers 접속
2. 로그인 후 Myapplication 탭에 들어가기
3. 어플리케이션 추가하기(앱 이름과 사업자 명 필수)
4. **카카오 로그인 활성화**
   - 내 애플리케이션 -> 제품 설정 -> 카카오 로그인 -> 카카오 로그인 활성화
5. 플랫폼 등록(android)
   - 패키지명 입력(AndroidManifest.xml의 package= “어트리뷰트”) 어트리뷰트 부분 입력
6. 동의 항목 설정(필요한 데이터 체크 선택or필수)
7. 발급받은 키를 아래와 같이 붙여넣기 한다.

```
// 만약 API key가 abc 라고 한다면
KAKAO_NATIVE_KEY="abc"
KAKAO_API_KEY=kakoabc
```

8. 해쉬 키 등록
   - 아래와 같은 로그 코드를 App 파일의 onCreate() 블럭안에 작성하고

- Logcat에서 메세지를 확인해 얻은 해쉬 키를
- 플랫폼 → Android플랫폼 등록 → 키 해시에 입력해준다.

```
Log.d(TAG, "keyhash : ${Utility.getKeyHash(this)}")
```

B. 실행

1. USB로 기기를 연결하거나, Device Manager에서 create device를 눌러 Oreo 버전 이상의 가상 기기를 설치합니다.
2. Run app으로 실행합니다.

Admin Client

A. 패키지 설치

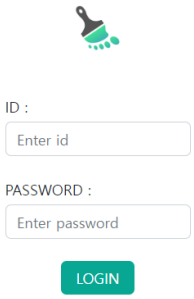
- 스프링부트 프로젝트 실행 시 자동으로 클라이언트가 빌드되도록 설정되어 있음.

B. 로컬 호스트 시작

1. Main Server 실행 후 다음 주소로 접근

```
https://[서버 호스트 주소]/admin
```

2. 관리자 페이지 화면



- 테스트용 관리자 계정

  ID) ssafy@admin.com PW) ssafy1234

사용자 시나리오

A. 어플리케이션 설치

1. 구글 플레이 스토어 근다-ArtWalk 접속
2. 어플리케이션 설치

B. 어플리케이션 실행

C. 회원가입 및 로그인

1. 소셜 회원가입 및 로그인
    a. 하단 카카오 로그인 버튼 클릭
    b. 카카오톡 로그인 진행
2. 일반 회원가입
    a. 회원가입 버튼 클릭
    b. 개인 정보 입력
    c. 회원가입 완료
3. 일반 로그인
    a. 아이디, 비밀번호 입력
    b. 로그인 버튼 클릭

D. 경로 그리기 및 경로 목록

1. 경로 그리기 버튼 클릭
2. 우하단 연필 버튼 클릭하여 그리기 모드 활성
3. 지도에 마커 찍어서 경로 그리기
4. 잘못 그린 경우 연필버튼 왼쪽에 있는 undo 버튼을 눌러 취소하기
5. 경로를 다 그린 후 우상단 저장하기 버튼 눌러 저장하기
6. 경로 목록에서 저장된 경로 확인하기
7. 경로를 눌러 상단 지도에서 확인하기
8. 경로의 시작 버튼을 눌러 경로를 따라 기록하기

E. 경로 기록하기

1. 하단 시작 버튼 클릭하여 기록 시작
2. 화면에 표시된 경로를 따라 이동하기
3. 실시간으로 화면에 나타나는 기록과 경로를 비교하며 이동 경로를 조정
4. 하단 중지 버튼 클릭하여 기록 완료
5. 썸네일을 확인하고 기록 제목을 입력한 후 저장
6. 기록을 저장하지 않을 경우 취소 버튼을 눌러 취소

F. 마이페이지 및 기록 상세보기

1. 기록수, 경로수 확인
2. 기록 썸네일을 클릭하여 해당 기록 상세 보기
3. 제목 오른쪽에 위치한 연필 버튼을 클릭
4. 변경하고 싶은 제목을 입력
5. 체크 버튼을 클릭하여 저장
6. 해당 기록을 작성하는데 걸린 시간과 이동거리 확인
7. 기록 상세보기 화면 우상단에 삭제 버튼을 클릭하여 해당 기록 삭제

G. 기록 공유하기

1. 기록 상세보기 화면 우상단에 공유하기 버튼 클릭
2. 지도를 움직여 원하는 구도로 위치하기
3. 색상과 굵기를 선택하여 기록 선 수정하기

4. 공유 이미지 생성하기 버튼 클릭
5. 완성된 이미지 확인하기
6. 공유하기 버튼을 눌러 공유하기
7. 공유된 링크로 접속하여 공유페이지 접속
8. 이미지 다운로드 버튼을 눌러 이미지 저장
9. 앱에서 확인하기 버튼을 눌러 구글 플레이 스토어로 이동

H. 설정 및 이용약관

1. 마이페이지 우상단 톱니 버튼을 클릭하여 접속
2. 사용방법을 클릭하여 사용설명 페이지 접속
3. 개인정보처리방침/위치기반서비스 이용약관 버튼을 통해 확인
4. 로그아웃 버튼을 눌러 로그아웃 진행
5. 회원 탈퇴 버튼을 눌러 회원 탈퇴 진행