

# 자율\_PJT\_대전2반\_B201\_포팅\_매뉴얼

B201 : 사과나무 추억걸렸네 - 타임 캡슐 서비스

삼성청년SW아카데미 대전캠퍼스 7기

자율 프로젝트 2022.10.11 ~ 2022.11.18 (6주)

담당 컨설턴트 - 정용기

송제영(팀장), 김낙현, 송선아, 이예은, 조다연, 차송희

## <목차>

B201 : 사과나무 추억걸렸네 - 타임 캡슐 서비스

삼성청년SW아카데미 대전캠퍼스 7기

자율 프로젝트 2022.10.11 ~ 2022.11.18 (6주)

담당 컨설턴트 - 정용기

송제영(팀장), 김낙현, 송선아, 이예은, 조다연, 차송희

## <목차>

Part 1. 포팅 매뉴얼

1. 프로젝트 기술 스택

1.1 이슈관리 : Jira

1.2 형상관리 : Gitlab

1.3 커뮤니케이션 : Mattermost, Notion, WebEx

1.4 개발 환경

1.5 UX/UI

1.6 Database 도구

1.7 Server : AWS EC2 - Ubuntu 20.04 LTS

1.8 기술 스택 상세

2. Property 정의

2.1 Frontend

2.2 Backend

2.3 외부 서비스

3. 빌드 상세내용

3.1 docker-compose.yml

3.2 Backend: Dockerfile( ./backend/backend-dockerfile )

3.3 Mail Backend: Dockerfile( ./backend-mail/mail-dockerfile )

3.4 rabbitmq: Dockerfile( ./backend/rabbit-dockerfile )

3.5 redis Dockerfile( ./backend/redis-dockerfile )

3.6 nginx 설정

3.7 Jenkinsfile

4. 배포 상세 내용 - Backend API

<자동 배포>

4.1. 자동 배포 설정

5. 어플리케이션 실행

5.1 React-Native Cli 환경설정

5.2. .env 파일 - react native project 최상위 폴더에 위치

5.3. 실행

## Part 1. 포팅 매뉴얼

### 1. 프로젝트 기술 스택

1.1 이슈관리 : Jira

1.2 형상관리 : Gitlab

### 1.3 커뮤니케이션 : Mattermost, Notion, WebEx

### 1.4 개발 환경

- OS : Window 10
- IDE
  1. IntelliJ 2022.1.3
  2. Vscode 1.70.0

### 1.5 UX/UI

- figma

### 1.6 Database 도구

- MySQL Workbench

### 1.7 Server : AWS EC2 – Ubuntu 20.04 LTS

- Reverse Proxy : nginx 1.23.1
- WAS : Tomcat Embed Core 9.0.65
- DB : MySQL 8.0.31
- 세션 상태 저장소 : redis 6.2.1
- 메세지큐 : RabbitMQ 3.11.1
- 콘텐츠 스토리지 : firebase storage(google cloud storage)

### 1.8 기술 스택 상세

1. Frontend	2. Backend	3. DB	4. 소켓 통신	5. 스토리지	6. CI/CD
React Native	Java <span>OpenJDK 11</span>	MySQL	RabbitMQ	Firebase Storage	AWS EC2
React	Spring Boot		Redis		Docker
React-native-firebase	Spring Data JPA				Jenkins
sockjs-client	QueryDSL				
webstomp-client	Spring Security				
axios	OAuth 2.0				
	Lombok				
	Spring Data Redis				
	Spring Rabbit				
	Spring Boot Websocket				

#### 1.8.1 Backend 상세

- 빌드 : Gradle 7.5
- Java : OpenJDK 11
- Spring Boot : 2.7.4
  - web
  - webflux
  - mvc
- Spring Data JPA : 2.7.1
- Lombok : 1.18.24
- QueryDsl: 5.0.0

- swagger-ui: 3.0.0
- Spring Data Redis
  - lettuce-core: 6.1.9
- Spring Boot OAUTH 2.0
  - Spring Security Core: 5.7.3
  - Spring Security Oauth Core: 5.7.3
- Spring Rabbit: 2.4.7
- Spring Boot WebSocket
  - Spring messaging: 5.3.23
  - Spring Websocket: 5.3.23
- Firebase SDK: 9.1.0

### 1.8.2 Frontend 상세

- React Native: 0.65.3
- React: 17.0.2
- React-native-firebase : 16.3.0
- sockjs-client : 1.6.1
- webstomp-client : 1.2.6
- axios : 1.1.3

## 2. Property 정의

### 2.1 Frontend

- .env 파일

```
API_APP_KEY = "백엔드 API 요청 URL"
API_BASE_URL = "백엔드 API 요청 URL"
// topic 실제 구독 경로 prefix
WS_APP_DEST_PREFIX = "/app"
// topic을 구독하고 있는 모든 사용자에게 메시지 전달
WS_BROKER_DEST_PREFIX = "/topic"

MQ_LOGIN = "RabbitMQ 유저명"
MQ_PASSCODE = "RabbitMQ 비밀번호"

// 아래 내용은 구글 클라우드 콘솔에서 확인 가능
GOOGLE_API_KEY = "구글 클라우드 API 키"
GOOGLE_WEB_CLIENT_ID = "어플리케이션 클라이언트 ID"
```

### 2.2 Backend

- application.yaml

```
spring:
  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
  jpa:
    open-in-view: false
    hibernate:
      ddl-auto: create
  mvc:
    pathmatch:
      matching-strategy: ant_path_matcher
  rabbitmq:
    port: 5672
  redis:
    port: 6379
```

```

security:
  oauth2:
    resourceserver:
      jwt:
        jwk-set-uri: https://www.googleapis.com/service_accounts/v1/jwk/securetoken%40system.gserviceaccount.com

com:
  cotyledon:
    appletree:
      // firebase 콘솔에서 확인 가능
      firebase-project-id: ${firebase 프로젝트 ID}
      firebase-service-account-secret: classpath:firebase-service-account-production-secret.json
    rabbitmq:
      stomp-port: 61613

logging:
  file:
    name: logs/production.log
  level:
    com.cotyledon.appletree: DEBUG
  org:
    hibernate:
      SQL: DEBUG
      type.descriptor.sql.BasicBinder: TRACE
    springframework.web: DEBUG

// 젠킨스 credential에 저장하여 자동 빌드시 추가하도록 함
spring.redis.host: ${redis 인스턴스}
spring.redis.password: ${redis 비밀번호}

spring.rabbitmq.host: ${rabbitmq 인스턴스}

```

## 2.3 외부 서비스

### 2.3.1 firebase

- **firebase-service-account.json**
  1. Firebase 콘솔에서 프로젝트 생성 (GCP에 기존 프로젝트가 있다면 생략 가능)
  2. 설정 → 프로젝트 설정 → 서비스 계정 → 새 비공개 키 생성
- 위의 설정 파일을 `\src\main\resources`, 또는 `classpath`에 위치 에 위치 시킨다.
  - jenkins credential 파일로 저장하여 빌드 전에 위치 시킴
- **firebase storage 권한 설정**
  - firebase console → storage → rules
  - 스토리지 상의 자원 요청하는 api 호출 시 백엔드에서 custom 클레임 수정

```

rules_version = '2';
service firebase.storage {
  match /b/{bucket}/o {
    function signedIn() {
      return request.auth.uid != null;
    }
    function claimTest(appleId) {
      return request.auth.token.appleId == appleId;
    }
    match /{appleId}/{all=*}* {
      allow read, delete: if signedIn() && request.auth.token.appleId == appleId;
    }
    match /{appleId}/audios/{all=*}* {
      allow read: if claimTest(appleId);
      allow write, delete: if request.resource.contentType.matches('audio/*') && signedIn();
    }
    match /{appleId}/images/{all=*}* {
      allow read: if claimTest(appleId);
      allow write, delete: if request.resource.contentType.matches('image/*') && signedIn();
    }
    match /{appleId}/videos/{all=*}* {
      allow read: if claimTest(appleId);
      allow write, delete: if request.resource.contentType.matches('video/*') && signedIn();
    }
  }
}

```

### 2.3.2 Google map API

- 구글 클라우드 콘솔에서 안드로이드용 키 발급
  - react-native project의 .env파일 해당 내용 추가

```
GOOGLE_API_KEY = "구글 클라우드 API 키"
GOOGLE_WEB_CLIENT_ID = "어플리케이션 클라이언트 ID"
```

## 3. 빌드 상세내용

### 3.1 docker-compose.yml

```
version: '3'

services:
  db:
    image: mysql:latest
    container_name: db
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD: ${MYSQL_ROOT_PASSWORD}
      MYSQL_DATABASE: ${MYSQL_DATABASE}
      MYSQL_USER: ${MYSQL_USER}
      MYSQL_PASSWORD: ${MYSQL_PASSWORD}
    volumes:
      - ${DB_BACKUP}:/var/lib/mysql
    ports:
      - '3306:3306'

  rabbitmq:
    container_name: rabbitmq
    restart: always
    build:
      context: ./backend
      dockerfile: rabbit-dockerfile
    environment:
      RABBITMQ_ERLANG_COOKIE: 'RabbitMQ-My-Cookies'
      RABBITMQ_DEFAULT_USER: ${RABBITMQ_DEFAULT_USER}
      RABBITMQ_DEFAULT_PASS: ${RABBITMQ_DEFAULT_PASS}
    ports:
      - '5672:5672'
      - '61613:61613'

  redis:
    container_name: redis
    build:
      context: ./backend
      dockerfile: redis-dockerfile
    restart: always
    environment:
      REDIS_ARG: ${REDIS_ARG}
    ports:
      - '6379:6379'

  api:
    container_name: api
    build:
      context: ./backend
      dockerfile: backend-dockerfile
    environment:
      SPRING_DATASOURCE_URL: ${SPRING_DATASOURCE_URL}
      SPRING_DATASOURCE_USERNAME: ${MYSQL_USER}
      SPRING_DATASOURCE_PASSWORD: ${MYSQL_PASSWORD}
    ports:
      - '8080:8080'
    depends_on:
      - db
      - rabbitmq
      - redis

  mail:
    container_name: mail
    build:
      context: ./backend-mail
      dockerfile: mail-dockerfile
    environment:
      SPRING_DATASOURCE_URL: ${SPRING_DATASOURCE_URL}
      SPRING_DATASOURCE_USERNAME: ${MYSQL_USER}
```

```

    SPRING_DATASOURCE_PASSWORD: ${MYSQL_PASSWORD}
ports:
  - '65500:65500'
depends_on:
  - db

```

```

# Docker compose에서 사용하는 변수 설정
MYSQL_ROOT_PASSWORD="MySQL root 비밀번호"
MYSQL_DATABASE="MySQL DB"
MYSQL_USER="MySQL 계정"
MYSQL_PASSWORD="MySQL 계정 비밀번호"
DB_BACKUP="DB 데이터 백업할 위치"

REDIS_ARG="--requirepass ${REDIS 비밀번호}"

RABBITMQ_DEFAULT_USER="RabbitMQ 계정"
RABBITMQ_DEFAULT_PASS="RabbitMQ 비밀번호"
RABBIT_ETC=/rabbitmq/etc
RABBIT_DATA=/rabbitmq/lib
RABBIT_LOG=/rabbitmq/log

SPRING_DATASOURCE_URL=jdbc:mysql://${MYSQL docker 컨테이너 명}:3306/${DB명}?useUnicode=true&characterEncoding=utf8&zeroDateTimeBehavior=c

```

### 3.2 Backend: Dockerfile( ./backend/backend-dockerfile )

```

FROM openjdk:11-jdk

WORKDIR /app

COPY ./build/libs/*.jar application.jar

ENV profile default
EXPOSE 8080

CMD ["java", "-jar", "-Dspring.profiles.active=$profile", "application.jar"]

```

### 3.3 Mail Backend: Dockerfile( ./backend-mail/mail-dockerfile )

```

FROM openjdk:11-jdk

WORKDIR /app

COPY ./build/libs/*SNAPSHOT.jar application.jar

ENV profile default
EXPOSE 65500

CMD ["java", "-jar", "-Dspring.profiles.active=$profile", "application.jar"]

```

### 3.4 rabbitmq: Dockerfile( ./backend/rabbit-dockerfile )

```

FROM rabbitmq:3.11.1-management-alpine
// 10-defaults.conf 파일의 경우 credential 정보를 포함하므로 jenkins credential 파일로 저장
COPY 10-defaults.conf /etc/rabbitmq/conf.d/10-defaults.conf
RUN rabbitmq-plugins enable rabbitmq_stomp
EXPOSE 5672 15671 15672 61613

```

### 3.5 redis Dockerfile( ./backend/redis-dockerfile )

```

FROM redis:6.2.1
// redis.conf, user.acl의 경우 credential 정보를 포함하므로 jenkins credential 파일로 저장
COPY redis.conf /usr/local/etc/redis/redis.conf
COPY users.acl /etc/redis/users.acl
EXPOSE 6379
CMD ["redis-server", "/usr/local/etc/redis/redis.conf"]

```

### 3.6 nginx 설정

```
server {
    listen 80;
    listen [::]:80;

    server_name k7b201.p.ssafy.io;

    location ~ /\.well-known/acme-challenge {
        allow all;
        root /usr/share/nginx/html;
        try_files $uri =404;
    }

    location / {
        return 301 https://$server_name$request_uri;
    }
}

server {
    listen 443 ssl;
    listen [::]:443 ssl;

    server_name k7b201.p.ssafy.io;

    ssl_certificate /etc/letsencrypt/live/k7b201.p.ssafy.io/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/k7b201.p.ssafy.io/privkey.pem;
    include /etc/letsencrypt/options-ssl-nginx.conf; # 보안 강화를 위한 옵션 추가
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # 보안 강화를 위한 옵션 추가

    location /api {
        proxy_pass http://127.0.0.1:8080;
        proxy_redirect off;

        rewrite ^/(.*)$ /$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-Host $server_name;
        proxy_set_header X-NginX-Proxy true;

        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
        proxy_http_version 1.1;
    }
}
```

### 3.7 Jenkinsfile

```
pipeline {
    agent any

    // 파이프라인에서 사용할 변수 설정
    environment {
        // 환경변수 파일을 Jenkins 크리덴셜로부터 가져옴
        // 그렇게 하기 위해서 Manage Jenkins > Manage Credentials 에서 크리덴셜 등록 (Kind: Secret file)
        COMPOSE_PRODUCTION = credentials('compose_production')
        BACKEND_PRODUCTION = credentials('backend_production')
        FIREBASE_PRODUCTION = credentials('firebase_production')
        RABBIT_PRODUCTION = credentials('rabbit_production')
        REDIS_CONF_PRODUCTION = credentials('redis_conf_production')
        REDIS_ACL_PRODUCTION = credentials('redis_acl_production')
        EMAIL_CONF_PRODUCTION = credentials('email_config')

        BACKEND_CONTAINER = 'api'
        MAIL_CONTAINER = 'mail'

        // MM 플러그인, Blue Ocean 플러그인 관련
        // MMACCOUNT = '@dss02094' // @아이디 사용 (언급시 알림)
        // MSGSUFFIX = "\nBuild <${RUN_DISPLAY_URL}|#${BUILD_NUMBER}>" // 메시지에 일괄적으로 달릴 링크
    }

    stages {
        // 빌드 전 정리 작업
        stage('pre_deploy') {
            // 병렬 처리 (파일 작업과 도커 작업)
```

```

parallel {
    // 파일 (환경 변수) 세팅
    stage('file_work') {
        stages {
            // 변경 사항을 지움 (백엔드 application-production.yml 때문)
            stage('git_clean') {
                steps {
                    sh 'git clean --force'
                }
            }
        }

        // 파일 세팅
        stage('set_files') {
            steps {
                sh 'cat $BACKEND_PRODUCTION >> backend/src/main/resources/application.yml'
                sh 'cp $FIREBASE_PRODUCTION backend/src/main/resources/firebase-service-account-production-secret.json'
                sh 'chmod 755 backend/src/main/resources/firebase-service-account-production-secret.json'
                sh 'cp $FIREBASE_PRODUCTION backend-mail/src/main/resources/firebase-service-account-production-secret.json'
                sh 'chmod 755 backend-mail/src/main/resources/firebase-service-account-production-secret.json'
                sh 'cp $EMAIL_CONF_PRODUCTION backend-mail/src/main/resources/email.properties'
                sh 'chmod 755 backend-mail/src/main/resources/email.properties'
                sh 'cp $REDIS_CONF_PRODUCTION backend/redis.conf'
                sh 'chmod 755 backend/redis.conf'
                sh 'cp $REDIS_ACL_PRODUCTION backend/users.acl'
                sh 'chmod 755 backend/users.acl'
                sh 'cat $RABBIT_PRODUCTION >> backend/10-defaults.conf'
            }
        }
    }
}

// 도커 관련 작업
stage('docker_work') {
    stages {
        // 안 쓰이는 이미지 제거
        stage('prune_images') {
            steps {
                catchError {
                    sh 'docker image prune --force'
                }
            }
        }

        // 같은 이름을 계속 사용하기 때문에 현재 작동 중인 컨테이너를 지움
        stage('remove_containers') {
            steps {
                catchError {
                    sh "docker rm --force ${BACKEND_CONTAINER} ${MAIL_CONTAINER}"
                }
            }
        }
    }
}

stage('build') {
    steps {
        dir('backend') {
            sh "chmod +x gradlew"
            sh "./gradlew clean bootjar"
        }
        dir('backend-mail') {
            sh "chmod +x gradlew"
            sh "./gradlew clean bootjar"
        }
    }
}

// 배포 본 작업
stage('deploy') {
    parallel {
        stage('docker_build') {
            steps {
                catchError {
                    sh "docker compose --env-file ${COMPOSE_PRODUCTION} up --build -d"
                }
            }
        }
    }
}
}
}

```



## 4. 배포 상세 내용 - Backend API

### <자동 배포>

#### 4.1. 자동 배포 설정

1. 젠킨스 설치
  - 본 프로젝트에서는 젠킨스를 Docker 이미지로 설치하여 사용함
2. 새로운 파이프라인 생성
  - 2.1. 빌드 트리거 설정

##### Build Triggers

☐ Build after other projects are built ?

☐ Build periodically ?

☒ Build when a change is pushed to GitLab. GitLab webhook URL: <http://k7b201.p.ssafy.io:8888/project/appletree> ?

Enabled GitLab triggers

☒ Push Events

☐ Push Events in case of branch delete

☒ Opened Merge Request Events

☐ Build only if new commits were pushed to Merge Request ?

☐ Accepted Merge Request Events

☐ Closed Merge Request Events

#### 2.2. 깃랩 주소 설정 및 Credential(깃랩 인증 정보) 추가

##### Pipeline

Definition

Pipeline script from SCM

SCM ?

Git

Repositories ?

Repository URL ?

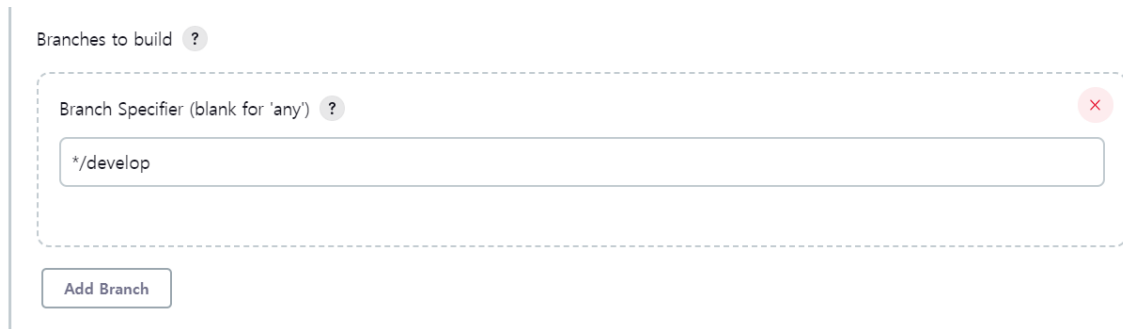
<https://lab.ssafy.com/s07-final/S07P31B201.git>

Credentials ?

appletree

+ Add

### 2.3. 빌드 타겟 브랜치 설정



### 2.4. 브랜치 내 파이프라인 스크립트 설정



### 2.5 GitLab webhook 설정 (Settings → Webhooks)

#### Webhook

[Webhooks](#) enable you to send notifications to web applications in response to events in a group or project. We recommend using an [integration](#) in preference to a webhook.

#### URL

Jenkins 상에서 파이프라인 생성시 gitlab 리포지토리 연동 설정시 확인 가능

URL must be percent-encoded if it contains one or more special characters.

#### Secret token

Jenkins 상에서 파이프라인 생성시 gitlab 리포지토리 연동 설정 시에 생성 가능

Used to validate received payloads. Sent with the request in the `X-Gitlab-Token` HTTP header.

#### Trigger Trigger 설정

☒ Push events

해당 branch에 push 이벤트 발생시 webhook 발생

Push to the repository.

☐ Tag push events

A new tag is pushed to the repository.

☒ Comments

A comment is added to an issue or merge request.

## 5. 어플리케이션 실행

### 5.1 React-Native Cli 환경설정

출처. [공식문서 Quik Start](#) - React Native CLI Quickstart

#### 사전 필수 설치

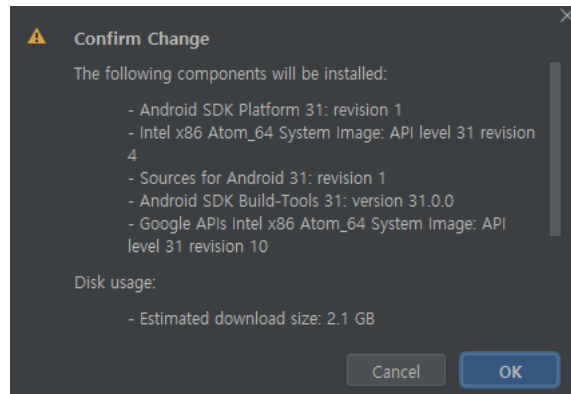
- [Chocolatey](#).
- Node.js 16.17.~
- openJDK 11
- Android Studio(전부 기본, accept로 설치)

- Android SDK
- Android SDK Platform
- Android Virtual Device

- 설치 확인

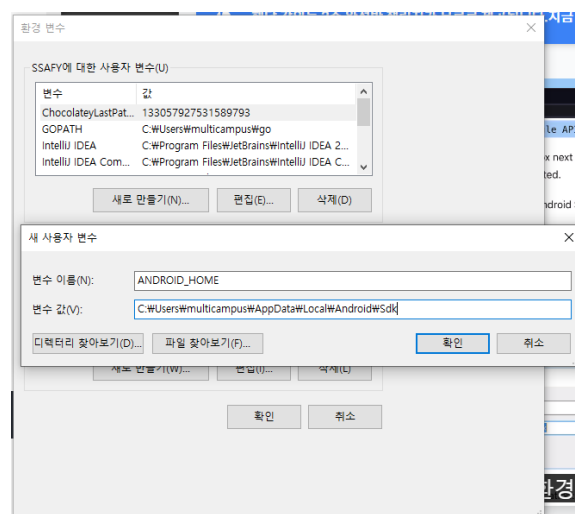
```
NomadWeather node -v
v16.17.0
NomadWeather npm -v
8.15.0
NomadWeather npx react-native --version
8.0.6
NomadWeather choco
Chocolatey v1.1.0
Please run 'choco -?' or 'choco <command> -?' for help menu.
```

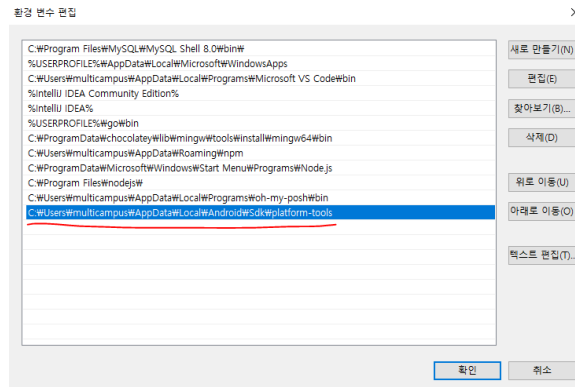
- Android Studio Customize



위에 해당하는 내용 설치 - 공식 문서 참고 -

- 환경변수 설정 - Android SDK, JAVA JDK





같은 방법으로 "JAVA\_HOME" 환경 변수 추가

- 환경변수 확인

```
multicampus Get-ChildItem -Path Env:\
```

Name	Value
ALLUSERSPROFILE	C:\ProgramData
<u>ANDROID_HOME</u>	C:\Users\multicampus\AppData\Local\Android\Sdk
APPDATA	C:\Users\multicampus\AppData\Roaming

## 5.2. .env 파일 - react native project 최상위 폴더에 위치

```
API_APP_KEY = "백엔드 API 요청 URL"
API_BASE_URL = "백엔드 API 요청 URL"
// topic 실제 구독 경로 prefix
WS_APP_DEST_PREFIX = "/app"
// topic을 구독하고 있는 모든 사용자에게 메시지 전달
WS_BROKER_DEST_PREFIX = "/topic"

MQ_LOGIN = "RabbitMQ 유저명"
MQ_PASSCODE = "RabbitMQ 비밀번호"

// 아래 내용은 구글 클라우드 콘솔에서 확인 가능
GOOGLE_API_KEY = "구글 클라우드 API 키"
GOOGLE_WEB_CLIENT_ID = "어플리케이션 클라이언트 ID"
```

## 5.3. 실행

```
# 터미널에서 아래 명령 실행
npx react-native run-android
```