배포가이드

▼ Gitlab 소스 클론 이후 빌드 및 배포할 수 있도록 정리한 문서

1. 사용한 JVM, 웹서버, WAS 제품 등의 종류와 설정값, 버전 :

IDE : intelij ultimate 22.3, vscode

Android studio : Electric Eel | 2022.1.1 Patch 2

백엔드 build.gradle 정보

```
plugins {
    id 'java'
    id 'org.springframework.boot' version '2.7.9'
    id 'io.spring.dependency-management' version '1.0.15.RELEASE'
}

group = 'com.ssafy'
version = '0.0.1-SNAPSHOT'
sourceCompatibility = '11'

configurations {
    compileOnly {
        extendsFrom annotationProcessor
    }
}

repositories {
    mavenCentral()
}

dependencies {
    implementation 'org.springframework.boot:spring-boot-starter-data-jpa'
    implementation 'org.springframework.boot:spring-boot-starter-oauth2-client'
    implementation 'org.springframework.boot:spring-boot-starter-oauth2-resource-server'
    implementation 'org.springframework.boot:spring-boot-starter-security'
    implementation 'org.springframework.cloud:spring-cloud-starter-aws:2.2.6.RELEASE'
    compileOnly 'org.projectlombok:lombok'
    annotationProcessor 'org.projectlombok:lombok'
    testImplementation 'org.springframework.boot:spring-boot-starter-test'
    testImplementation 'org.springframework.security:spring-security-test'
    implementation 'com.mysql:mysql-connector-j:8.0.32'
    implementation 'org.springframework.boot:spring-boot-starter-websocket'
    //실행시 종료안하고 어플리케이션 유지 등 web 관련 기능.
    implementation 'org.springframework.boot:spring-boot-starter-web'

    //swagger 2.9.2 : swagger-ui가 안되는 이슈.
    // classpath문제로 보임.
//  implementation 'io.springfox:springfox-swagger2:2.9.2'
//  implementation 'io.springfox:springfox-swagger-ui:2.9.2'

//  swagger 3.0.0으로 교체.
    implementation group: 'io.springfox', name: 'springfox-boot-starter', version: '3.0.0'
    implementation group: 'io.springfox', name: 'springfox-swagger-ui', version: '3.0.0'

    //코드 변경시 서버 재시작
    implementation 'org.springframework.boot:spring-boot-devtools:2.6.2'

    //configurationProperties
    annotationProcessor "org.springframework.boot:spring-boot-configuration-processor"

    //jjwt
    implementation 'io.jsonwebtoken:jjwt-api:0.11.2'
    runtimeOnly 'io.jsonwebtoken:jjwt-impl:0.11.2'
```

```
    runtimeOnly 'io.jsonwebtoken:jjwt-jackson:0.11.2'

    // redis
    implementation 'org.springframework.boot:spring-boot-starter-data-redis'

    //naver cloud와 소통하는 httpClient
//    implementation group: 'org.apache.httpcomponents', name: 'httpclient', version: '4.5.1'
//    implementation group: 'org.apache.httpcomponents', name: 'httpclient', version: '4.3.6'
//    implementation 'org.springdoc:springdoc-openapi-ui:1.6.15'

}

// gradle build 시, simple.jar 파일 생성 skip 명령어 (파일 경로 : build/libs)
jar {
    enabled = false
}

tasks.named('test') {
    useJUnitPlatform()
}
```

프론트 package.json정보:

```
{
  "name": "finedUI",
  "version": "0.0.1",
  "private": true,
  "scripts": {
    "android": "react-native run-android",
    "ios": "react-native run-ios",
    "start": "react-native start",
    "test": "jest",
    "lint": "eslint ."
  },
  "dependencies": {
    "@react-native-community/checkbox": "^0.5.15",
    "@react-native-community/datetimepicker": "^7.0.0",
    "@react-navigation/bottom-tabs": "^6.5.7",
    "@react-navigation/native": "^6.1.6",
    "@react-navigation/native-stack": "^6.9.12",
    "axios": "^1.3.4",
    "date-fns": "^2.29.3",
    "lottie-react-native": "^6.0.0-rc.3",
    "react": "18.1.0",
    "react-native": "0.70.5",
    "react-native-basic-carousel": "^1.0.9",
    "react-native-geolocation-service": "^5.3.1",
    "react-native-image-base64": "^0.1.4",
    "react-native-image-crop-picker": "^0.39.0",
    "react-native-keychain": "^8.1.1",
    "react-native-linear-gradient": "^2.6.2",
    "react-native-maps": "^1.4.0",
    "react-native-modal-datetime-picker": "^14.0.1",
    "react-native-responsive-dimensions": "^3.1.1",
    "react-native-safe-area-context": "^4.5.0",
    "react-native-screens": "^3.20.0",
    "react-native-vector-icons": "^9.2.0",
    "react-native-wrapped-text": "^1.2.2",
    "react-stomp": "^5.1.0",
    "recoil": "^0.7.7",
    "recoil-nexus": "^0.4.0",
    "rn-material-ui-textfield": "^1.0.9",
    "sockjs-client": "^1.6.1",
    "stompjs": "^2.3.3"
  },
  "devDependencies": {
    "@babel/core": "^7.12.9",
    "@babel/runtime": "^7.12.5",
    "@react-native-community/eslint-config": "^2.0.0",
    "babel-jest": "^26.6.3",
```

```
    "eslint": "^7.32.0",
    "jest": "^26.6.3",
    "metro-react-native-babel-preset": "0.72.3",
    "react-native-dotenv": "^3.4.8",
    "react-test-renderer": "18.1.0"
  },
  "jest": {
    "preset": "react-native"
  }
}
```

빌드시 사용되는 환경변수의 내용 상세 기재.

docker 환경으로 빌드. docker 빌드시 환경변수 옵션은 없고,

백/프론트의 기본 환경설정 파일을 이용함.

백엔드 : 주석으로 상세설명함.
1. application.yml : DB/JPA 정보, 로깅정보, jwt토큰관련정보, oauth2정보, 파일 업로드 multipart 정보, 이미지 업로드시 소스정보, s3서버정보
2. application-aiserver.yml : ai서버에 대한 정보.
3. application-oauth2 : kakao, google 관련 oauth2설정정보.
4. application-redis : redis 관련 정보
5. application-sms : 네이버 클라우드 휴대폰 SMS 메시지 정보.

프론드엔드 : 주석으로 상세히 설명함.

MAIN_URL

TEST_URL

KAKAO_URL_GET_ADDRESS

KAKAO_URL_GET_LNGLAT

KAKAO_MAP_API_KEY

배포시 특이사항 기재:

dockerfile : 이미지생성에 사용. 프로젝트 루트경로에있음.

```
# openjdk:11 버젼 Image pull
FROM openjdk:11

# 해당 build의 jar 파일들을 변수 선언
ARG JAR_FILE="build/libs/*.jar"

# 왼쪽 path에 있는 jar파일들을 app.jar로 복사
COPY ${JAR_FILE} app.jar

# CMD와 비슷함 -> app.jar 파일은 java -jar로 실행
ENTRYPOINT ["java","-jar", "-Duser.timezone=Asia/Seoul","app.jar"]
```

docker-compose.yml : spring boot용. 생성된 이미지 실행에 사용.

```
내용
```

docker-compose.yml : elasticsearch, fastapi 딥러닝서버 배포용.

```
version: "3.1"
services:
  elasticsearch:
    image: docker.elastic.co/elasticsearch/elasticsearch:8.7.0
    container_name: elasticsearch
    # container 실행 순서 보장 (db -> server)
    # depends_on:
    #   - database
    ports:
      - 9200:9200
    environment:
      - discovery.type=single-node
      - bootstrap.memory_lock=true # along with the memlock settings below, disables swapping
      - xpack.security.enabled=false
    # container에 문제가 생겨 종료됐을때 자동 재시작 true
    networks:
      - find_net
networks:
  find_net:
    # 해당 이름의 새 네트워크 생성이 아닌 기존에 생성된 network 활용
    external: true
```

fastapi 딥러닝서버는 직접 로컬로 실행

1. python 3.8.6 환경에서 필요한 패키지 설치.

```
python -m pip install

torch==1.8.0 torchvision==0.9.0
opencv-python
skimage
python-multipart
elasticsearch
fastapi
uvicorn[standard]
matplotlib
torch==1.8.1+cpu torchvision==0.9.1+cpu torchaudio==0.8.1 -f https://download.pytorch.org/whl/torch_stable.html
```

2. 이후 aiserver폴더의 내용을 특정 디렉토리(a)에 옮기고, a에서 AI모델다운로드

AI 모델 다운로드 링크 : https://github.com/Hzzone/MTLFace

특이사항) style2gan 은 사용하지않으나, style2gan이 cuda를 요구함. ec2서버는 gpu가 없기때문에 tar파일과 모델코드에서 style2gan에 관한 부분을 삭제할것.

3. uvicorn main:api —host=0.0.0.0 --reload 으로 fast api 실행.

DB 접속 정보 등 프로젝트(ERD)에 활용되는 주요 계정 및 프로퍼티가 정의된 파일 목록.

주요 계정 :

```
email : ssafy@gmail.com
password :
```

ERD, 아키텍쳐는 readme.md에 기술.

▼ 프로젝트에서 사용하는 외부 서비스 정보를 정리한 문서:

카카오톡 oauth 로그인 설정: https://velog.io/@yunsungyang-omc/카카오-로그인-연동을-통한-OAuth-2.0-이해하기

구글 oauth 로그인 설정: https://yeonyeon.tistory.com/34

네이버클라우드 SMS : https://www.ncloud.com/product/applicationService/sens

카카오 맵 api : https://apis.map.kakao.com/

▼ DB 덤프 파일 최신본:

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/4a253969-5748-4b7b-a370-14d1efbc438c/backup.sql

▼ 시연시나리오:

추후 추가.