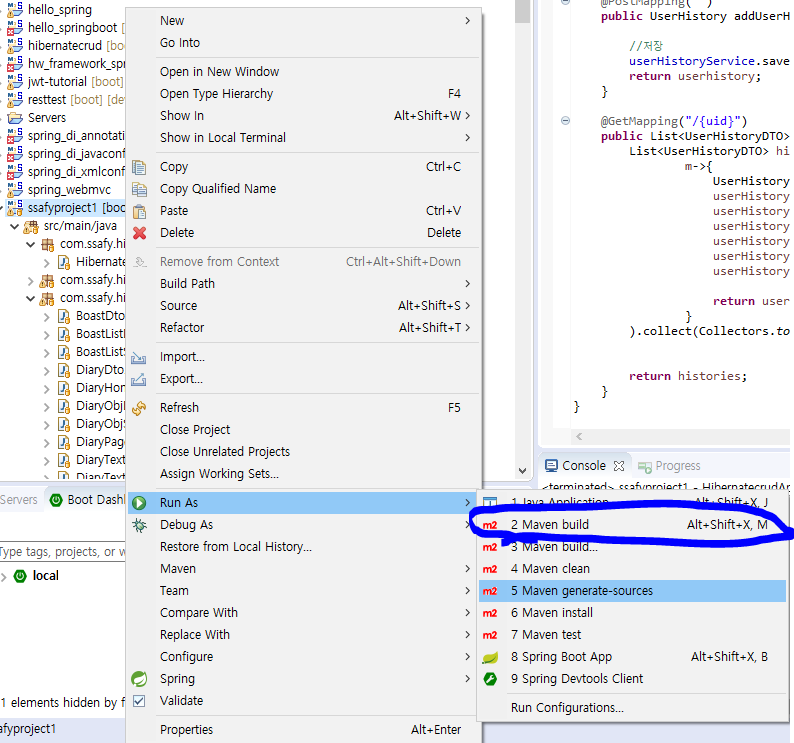
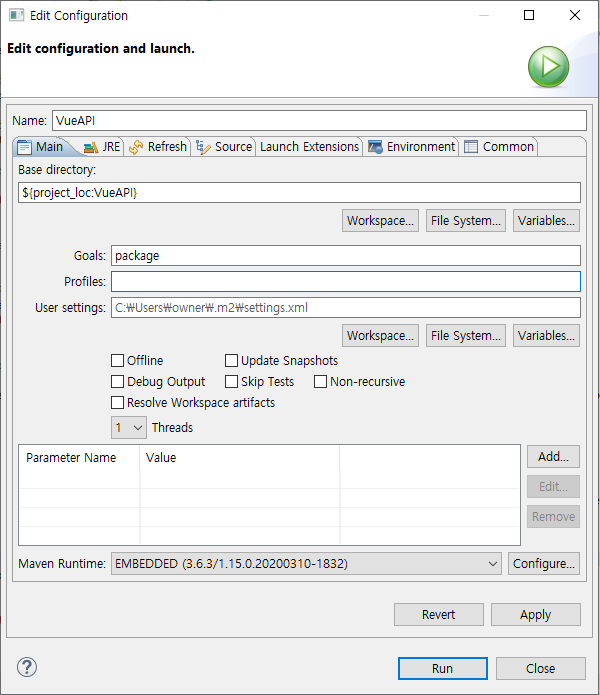
1. 버전 정리
   1. 백엔드
      1. JDK

* Openjdk version 1.8.0 312
* OpenJDK Runtime Environment (build 1.8.0\_312-8u312-b07-0ubuntu1~20.04-b07)
* OpenJDK 64-Bit Server VM (build 25.312-b07, mixed mode)
  + 1. SpringBoot
* SpringBoot 2.7.2
  + 1. DB
* MySQL 8.0.30-0ubuntu0.20.04.2
  + 1. Aws ec2
* Ubuntu 20.04 LTS (GNU/Linux 5.4.0-1018-aws x86\_64)
  + 1. Docker
* Docker version 20.10.17, build 100c701
  + 1. Jenkins
* Jenkins 2.363
  + 1. Nginx
* Nginx/1.18.0 (Ubuntu)
  1. 프론트엔드
     1. 안드로이드 스튜디오 설정
* AndroidStudio 2021.1.1
* JDK: Zulu-8/bin
* CompileSdk : 30 , minSdk : 21, targetSdk 30

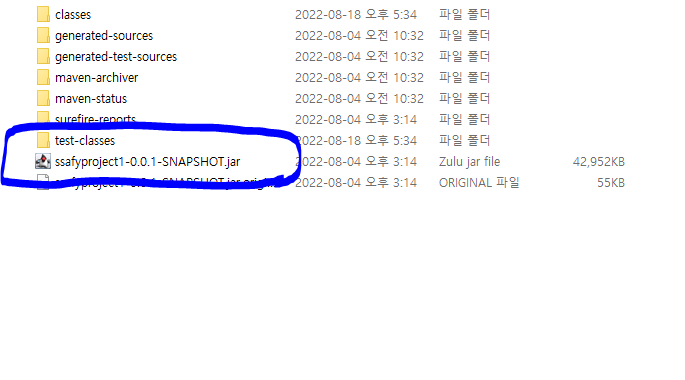
1. 빌드
   1. 백엔드
      1. Gui 빌드



프로젝트 우클릭->Run As->Maven build



Goals에 package 입력, Profiles에 있는 pom.xml 제거후 하단 Run버튼을 클릭해서 build를 시작한다.



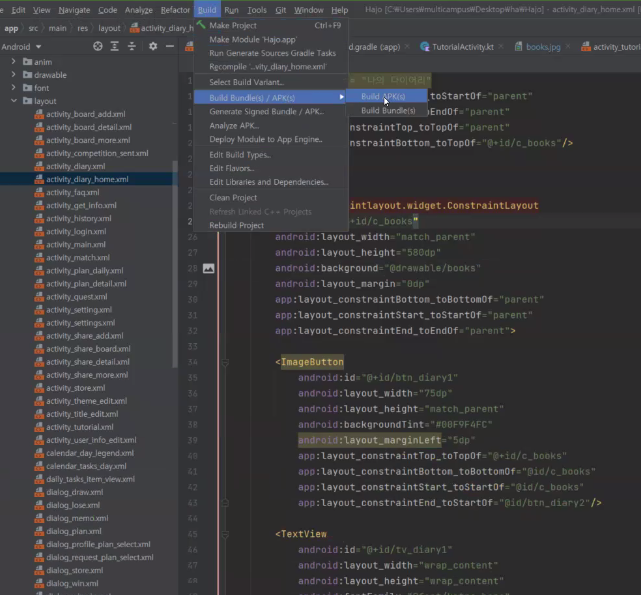
빌드성공시 프로젝트 폴더의 target 폴더에 jar파일이 생성됩니다.

* + 1. 명령어 빌드



Maven 설치 이후 pom.xml이 있는 디렉토리에서 mvn package 명령어를 입력시 target 폴더에 jar파일이 생성됩니다.

* 1. 프론트엔드
     1. GUI 빌드



안드로이드 스튜디오 Build->Build Bundle(s) /APK(s)->Build APK(s) 클릭.



빌드성공시 프로젝트명\app\build\intermediates\apk\debug 폴더에

app-debug.apk 파일이 생성됩니다.

1. 배포 시 특이사항
   1. 백엔드

백엔드는 aws ec2에 Mysql을 설치했고, Docker와 Jenkins를 통한 CI/CD 구축 및 ssl 을 통한 https설정이 되어있습니다. 아래에서 각각에 대한 설치/설정법을 설명하겠습니다.

* + 1. Mysql
* Mysql 설치

sudo apt-get update

sudo apt-get install mysql-server 명령어를 순서대로 입력해 mysql을 설치합니다.

sudo ufw allow mysql을 입력해 mysql 방화벽을 허용해줍니다.

* 실행 설정

sudo systemctl start mysql을 입력해서 mysql을 실행하고,

sudo systemctl enable mysql을 입력해서 서버 재시작시 mysql이 자동실행되게 설정합니다.

* 우분투 시간 변경

sudo dpkg-reconfigure tzdata을 입력해서 서버 시간을 한국시간으로 설정합니다.

* Mysql 계정 설정

sudo mysql -u root -p 를 입력해서 mysql에 root계정으로 진입한다. 계정을 생성하거나, root 비밀번호를 변경합니다.(이 설명에서는 root 계정 사용 기준)

mysql에서 나온 뒤에는 sudo service mysql restart를 입력해 mysql을 재시작합니다.

* + 1. Openjdk 설치
* sudo apt-get update 입력후
* sudo apt-get install openjdk-8-jdk 를 입력해서 openjdk를 설치합니다.
  + 1. Docker 설치
* sudo apt-get install apt-transport-https ca-certificates curl gnupg-agent software-properties-common
* curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add –
* sudo add-apt-repository \ "deb [arch=amd64] https://download.docker.com/linux/ubuntu \ $(lsb\_release -cs) \ stable"
* sudo apt-get update && sudo apt-get install docker-ce docker-ce-cli containerd.io

위의 명령어를 순서대로 실행해서 Docker를 설치합니다.

* gitlab의 springboot 프로젝트의 pom.xml이 있는 위치에 Dockerfile을 생성합니다.
* Dockerfile은 다음과 같이 작성합니다.



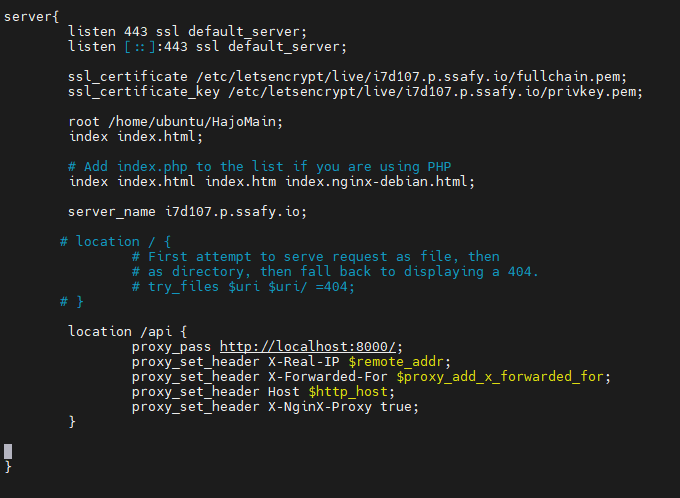
* + 1. Nginx 설치
* sudo apt-get install nginx를 입력해서 nginx를 설치합니다.
  + 1. Docker에 Jenkins 설치 및 실행
* sudo docker run -d --name jenkins -u root --privileged -p '9090:8080' -v '/home/ubuntu/docker-volume/jenkins:/var/jenkins\_home' -v '/var/run/docker.sock:/var/run/docker.sock' -v '/usr/bin/docker:/usr/bin/docker' jenkins/Jenkins

위의 명령어를 실행해서 Docker에 Jenkins를 설치하고 container를 올립니다.

* sudo docker logs Jenkins 를 입력해서 Jenkins 설치에 필요한 패스워드를 가져옵니다.
* 원격서버주소:9090을 입력해서 Jenkins 설치 페이지로 넘어갑니다.
* 계정 설정을 한 뒤에 기본 패키지를 설치합니다
* 이후 설정->플러그인 매니저에서 gitlab에 관련된 플러그인을 설치합니다.
* 새로운 view를 만들고 구성에서 pipeline을 등록합니다.
* Pipeline script인 Jenkinsfile을 프로젝트 최상위에 등록합니다.
* Jenkinsfile의 구성은 아래와 같습니다.



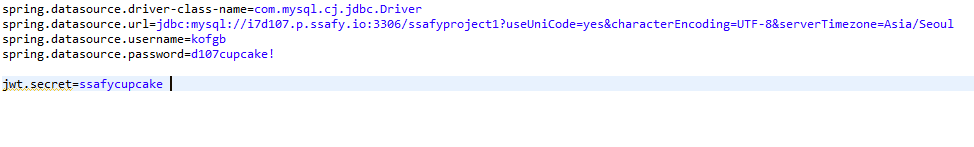
* + 1. ssl 적용
* sudo service nginx stop을 입력해 nginx를 정지합니다.
* sudo apt install python3-certbot-nginx 을 입력해서 certbot을 설치합니다.
* sudo certbot certonly –nginx 를 입력해서 인증서를 발급받습니다. 인증받은 키는 nginx의 default파일에 설정을 해주어야 합니다.
* sudo service nginx start을 입력해 nginx를 실행시킵니다.
* cd /etc/nginx/sites-available를 입력해서 default가 있는 폴더로 이동합니다.
* sudo vi default을 입력해서 default 설정을 수정합니다.



위와 같이 설정을 해줍니다. location /api에 있는 8000포트는 백엔드 서버가 돌아가는 포트번호입니다. 또 root /home/ubuntu/HajoMain은 프론트엔드 배포를 위한 설정입니다.

* 설정을 끝내고 나면 sudo service nginx restart를 입력해 nginx를 재시작합니다.
  1. 프론트엔드
     1. 프론트엔드는 소셜로그인 기능을 사용하는데, 검수가 덜끝나서 부득이하게 테스트버전으로 apk를 배포하고 있습니다.

1. 프로퍼티 정의 파일 목록
   1. 백엔드
      1. application-dev.properties



Application-dev.properties에는 Database에 접속하기 위한 주소,정보와

jwt secret key가 있습니다. git에는 올라가지 않을 예정입니다.