포팅 메뉴얼

A104: 꼬박꼬박

삼성SW 청년아카데미 서울캠퍼스 7기

자율프로젝트 6주, 2022/10/11~2022/11/21

포팅 매뉴얼

담당 컨설턴트- 이태희

박세현(팀장), 강승리, 박정미, 백경원, 임효정, 홍성영

<<목차>>

1. 기술 스택 ————————————————————————————————-
2. 빌드 상세내용 ————————————————————————————
3. 배포 특이사항 ————————————————————————-————-
4. DB 계정 ————————————————————————————————
5. 프로퍼티 정의 ————————————————————————————
6. 외부 서비스 ————————————————————————————

꼬박꼬박은 웹, 모바일, 워치를 통한 모든 플랫폼을 활용한 습관 형성 챌린지 서비스입니다. 구축된 생태계를 통해 습관형성을 위한 높은 접근성을 제공합니다.

1. 프로젝트 기술 스택

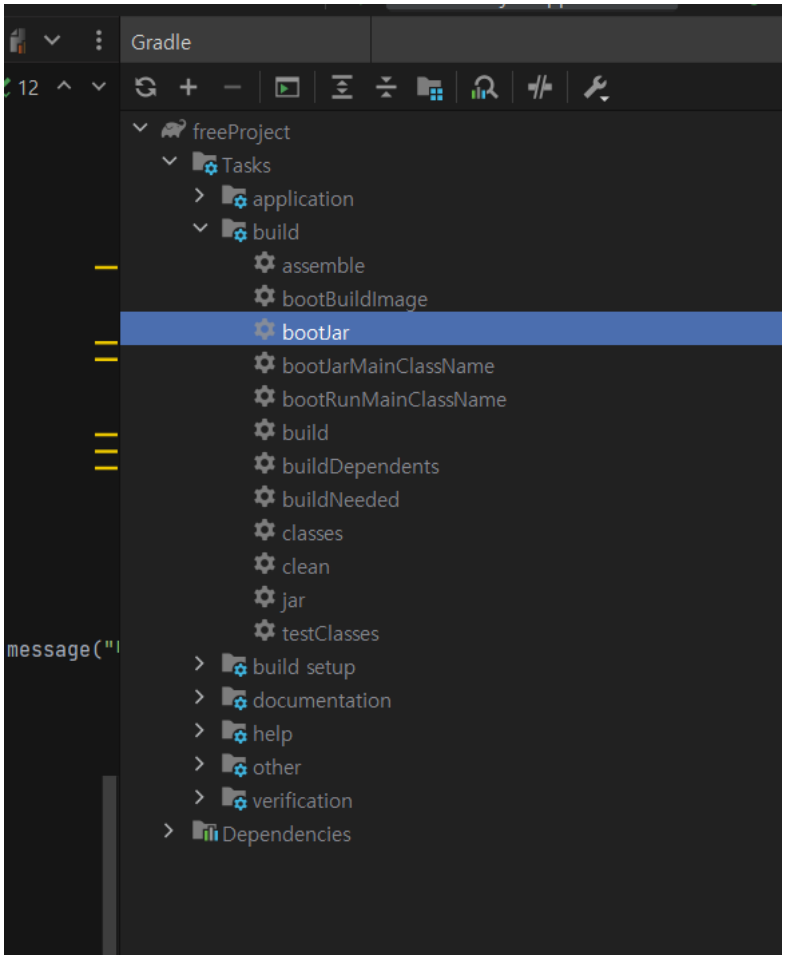
- 이슈관리 :Jira
- 형상관리: GitLab
- CI/CD: Jenkins
- 커뮤니케이션: Mattermost, Notion, Discord, Webex
- 개발환경
  - OS: Windows 10
  - IDE
    - Intellij IDEA Community Edition 2022.1.3
    - Visual Studio Code 1.69.1
  - Database: MySql WorkBench 8.0 CE
  - Server : AWS EC2
- 상세 사용
  - Backend
    - JAVA 8
    - Spring boot, Gradle
    - lombok, Swagger
    - Spring data JPA,

        - Spring Security, JWT
    - Frontend
        - React 18.2.0
        - React-router-dom 6.4.2
        - React-calendar 3.9.0
        - node.js 16.18.0 LTS
    - Android
        - App: `'com.android.application' version '7.3.1'`
        - WearOs: android 30
        - firebase: `'com.google.gms.google-services' version '4.3.10'`
    - AWS EC2
        - Nginx
        - Docker

2. 빌드 상세 내용

Spring build

- 우측의 Gradle -> mafya -> Tasks -> build -> bootJar를 더블 클릭



- build 폴더가 생성되고, 그 아래에 libs 폴더 아래에 `freeProject-0.0.1-SNAPSHOT.jar`가 빌드되었는지 확인

- **실행 방법 (백그라운드)**

```
nohup java –jar freeProject-0.0.1-SNAPSHOT.jar &AI build실행 방법(백그라운드)
```

React build

- node.js 16.18.0 LTS 설치
  - (`npm -v`를 통해 버전 확인 8.19.2)
- git clone 진행
- .env 생성 (생성 위치는 아래 폴더구조 확인 : frontend>kkobak안 위치에 생성하기)
  - .env 안에 들어가야 하는 정보
    - REACT_APP_KAKAOMAP_API={발급받은 키}
    - REACT_APP_KAKAO_REST_API={발급받은 키}
    - REACT_APP_OPENWEATHERMAP_API={발급받은 키}
- 빌드 방법 - clone 받은 /frontend/kkobak로 이동한 후, 다음 커멘드 입력

```
npm install
npm run build
```

- build 파일 생성되었는지 확인

크롬 익스텐션

- `npm run build` 실행을 통해 build 폴더 생성 (생성 위치는 아래 폴더구조 확인 : frontend>kkobak안 위치)
- 크롬 확장 프로그램에서 개발자 모드를 실행
  - 해당 build 파일을 `압축해제된 확장 프로그램을 로드합니다.`통해 로컬에서 실행가능
  - `확장 프로그램 압축`을 통해 해당 build 파일을 압축하여 웹 스토어에 배포 진행.

apk build

- 안드로이드 스튜디오 상단 메뉴 [Build]에서 [Generate Signed Bundle or APK]를 클릭
- [APK]를 선택하고 [Next]를 클릭
- 원하는 경로와 password를 설정후 apk추출

3. 배포 특이사항

docker engine 설치

https://docs.docker.com/engine/install/ubuntu/

- Repository 설정

```
sudo apt-get updatesudo apt-get install \    ca-certificates \    curl \    gnupg \    lsb-release
```

- GPG 키

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

- 도커 설치

```
sudo apt-get updatesudo apt-get install docker-ce docker-ce-cli containerd.io
```

Jenkins 설치

https://www.jenkins.io/download/

- 젠킨스 컨테이너 초기 설정

```
Dockerfile
# Dockerfile

FROM jenkins/jenkins:lts  # 젠킨스 lts 버젼을 다운로드 받음

USER root  # 이후의 명령어의 사용자를 root로 설정함

# docker install
RUN apt-get update && \
 apt-get -y install apt-transport-https \
 ca-certificates \
 curl \
 gnupg2 \
 zip \
 unzip \
 software-properties-common && \
curl -fsSL https://download.docker.com/linux/$(. /etc/os-release; echo "$ID")/gpg > /tmp/dkey; apt-key add /tmp/dkey && \
 add-apt-repository \
 "deb [arch=amd64] https://download.docker.com/linux/$(. /etc/os-release; echo "$ID") \
 $(lsb_release -cs) \
 stable" && \
 apt-get update && \
 apt-get -y install docker-ce

 # docker-compose install
 RUN curl -L "https://github.com/docker/compose/releases/download/1.28.5/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/
bin/docker-compose && \
    chmod +x /usr/local/bin/docker-compose && \
    ln -s /usr/local/bin/docker-compose /usr/bin/docker-compose
```

- 다음의 명령어로 외부 도커 환경과 젠킨스 컨테이너의 볼륨 연결해서 올려질 젠킨스 컨테이너 내부에서 호스트OS의 도커 데몬으로 접근할 수 있도록 설정

```
docker run -d -p 9090:8080 -v /var/run/docker.sock:/var/run/docker.sock -v /home/ubuntu/jenkins_home:/var/jenkins_home jenkins/
jenkins:lts
```

- 서버 url의 9090 포트로 접속하면, Jenkins 초기 화면 등장. `sudo cat /var/lib/jenkins/secrets/initialAdminPassword` 를 통해 초기 비밀번호 확인 후 입력
- install suggested plugins 선택하여 초기 플러그인 설치한 후, admin 계정을 생성한다.
- Docker, git, Node, Java를 설치

젠킨스와 Gitlab 연동(자동 배포)

- backend 프로젝트 생성
  - 필요한 플러그인을 설치 후, project 생성. branches to build는 merge request 또는 push event가 발생했을 경우 자동 빌드되는 브랜치를 의미. 이 프로젝트에서는 backend 브랜치로 설정했음
  - Execute shell을 다음과 같이 설정

```
cd ${WORKSPACE}/backend/kkobakdocker ps -q --filter name=kkobak_spring_container | grep -q . && docker stop kkobak_spring_container
&& docker rm kkobak_spring_containerdocker ps -a -q --filter name=kkobak_spring_container | grep -q . && docker rm kkobak_spring_co
ntainerdocker build -t kkobak_spring_image .docker run -d -p 8080:8080 -v /home/ubuntu/ai/server/ai/facebank:/sehyeon -v /home/ubun
tu/ai/server/ai/identify:/identify --name kkobak_spring_container kkobak_spring_image
```

- frontend 프로젝트 생성
  - 이 프로젝트에서는 branches to build를 frontend로 설정
  - Execute Shell을 다음과 같이 설정

```
cd ${WORKSPACE}/frontend/kkobakdocker ps -q --filter name=kkobak_react_container | grep -q . && docker stop kkobak_react_container
 && docker rm kkobak_react_containerdocker ps -a -q --filter name=kkobak_react_container | grep -q . && docker rm kkobak_react_cont
ainerdocker build -t kkobak_react_image .docker run -d -p 3000:3000 --name kkobak_react_container kkobak_react_image
```

3 - 1 frontend 수동 배포

가) 클론받은 프로젝트 폴더에서 프론트엔드 폴더에 있는 kkobak 폴더로 이동합니다.

나) 도커 이미지 생성에 필요한 Dockerfile은 다음과 같습니다.

```
# DockerFileFROM node:16RUN mkdir -p /usr/src/appWORKDIR /usr/src/appENV PATH /usr/src/app/node_modules/.bin:$PATHCOPY package*.jso
n ./RUN npm installCOPY ./ ./EXPOSE 3000CMD ["npm", "start"]
```

다) 도커 이미지를 생성한 후, 도커 컨테이너를 통해 프론트엔드를 배포합니다.

```
# 저장소 클론git clone https://lab.ssafy.com/s07-final/S07P31A104.git# 프론트엔드 폴더로 이동cd frontend/kkobak# 도커 컨테이너에 있는 기존 도커 이
미지 stopdocker ps -q --filter name=kkobak_react_container | grep -q . && docker stop kkobak_react_container && docker rm kkobak_rea
ct_container# 도커 컨테이너에 있는 기존 도커 이미지 삭제docker ps -a -q --filter name=kkobak_react_container | grep -q . && docker rm kkobak
_react_container# 도커 이미지 생성docker build -t kkobak_react_image .# 도커 이미지 실행docker run -d -p 3000:3000 --name kkobak_react_con
tainer mafya_react_image
```

3 - 2 Backend 수동 배포

Dockerfile 내용 확인

가) 클론받은 프로젝트 폴더에서 백엔드 폴더에 있는 freeProject 폴더로 이동합니다.

나) 도커 이미지를 생성하기 위한 도커파일은 다음과 같습니다.

```
FROM openjdk:8-jdkWORKDIR .COPY build/libs/freeProject-0.0.1-SNAPSHOT.jar application.jarEXPOSE 8080CMD ["java", "-jar", "applicati
on.jar"]
```

다)도커 이미지를 생성한 후, 도커 컨테이너를 통해 백엔드를 배포합니다,

```
git clone https://lab.ssafy.com/s07-final/S07P31A104.git cd backend/kkobakdocker ps -q --filter name=kkobak_spring_container | grep
-q . && docker stop kkobak_spring_container && docker rm kkobak_spring_containerdocker ps -a -q --filter name=kkobak_spring_contain
er | grep -q . && docker rm kkobak_spring_container docker build -t kkobak_spring_image .docker run -d -p 8080:8080 -v /home/ubunt
u/ai/server/ai/facebank:/sehyeon -v /home/ubuntu/ai/server/ai/identify:/identify --name kkobak_spring_container kkobak_spring_image
```

---

3 - 3 SSL 인증서 적용

- SSL 인증서를 적용하기 위해 다음과 같은 명령어를 입력합니다.

```
sudo apt-get install certbot python3-certbot-nginxsudo service stop nginxsudo certbot certonly --standalone -d kkobak.ml
```

- 이 경우 /etc/letsencrypt/live/kkobak.ml에 ssl 인증서가 설치됩니다.
- /etc/nginx/sites-available로 이동, 아래와 같은 파일을 생성합니다.

```
server {         location /{            proxy_pass http://localhost:3000/;          }          location /api{            proxy_pass
http://localhost:8080/;        }             listen 443 ssl;         server_name kkobak.ml;               ssl_certificate /etc/l
etsencrypt/live/kkobak.ml/fullchain.pem; # managed by Certbot               ssl_certificate_key /etc/letsencrypt/live/kkobak.ml/pr
ivkey.pem; # managed by Certbot}server {    if ($host = kkobak.ml) {        return 301 https://$host$request_uri;    } # managed by
Certbot        listen 80;        server_name kkobak.ml;    return 404; # managed by Certbot }
```

- 80 port로 접근할 경우 ssl인증서가 적용된 443 port 로 리다이렉트됩니다.
- 또한 443 port의 /api로 접근 시에는 localhost:8080로 분기 처리됩니다.
- 이후 다음 명령어를 입력하면 ssl 인증서가 적용됩니다

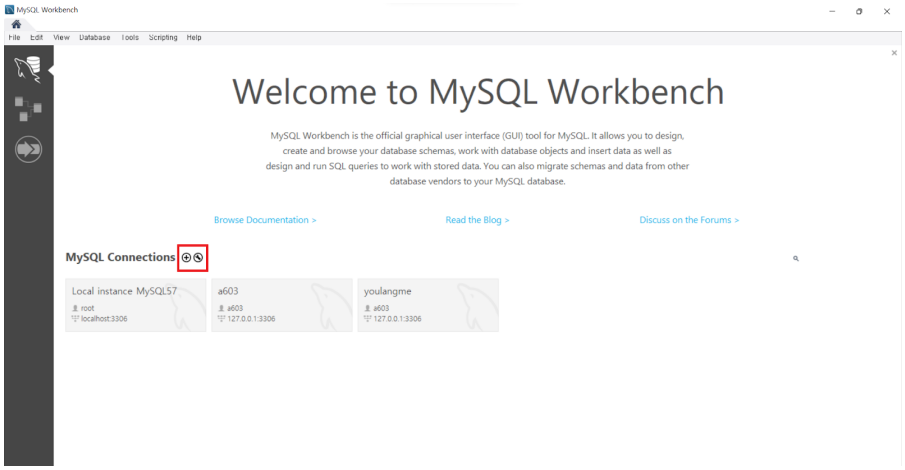
```
sudo ln -s /etc/nginx/sites-available/[파일명] /etc/nginx/sites-enabled/[파일명]
# 필자의 경우 kkobak.conf

sudo nginx -t

sudo service restart nginx
```

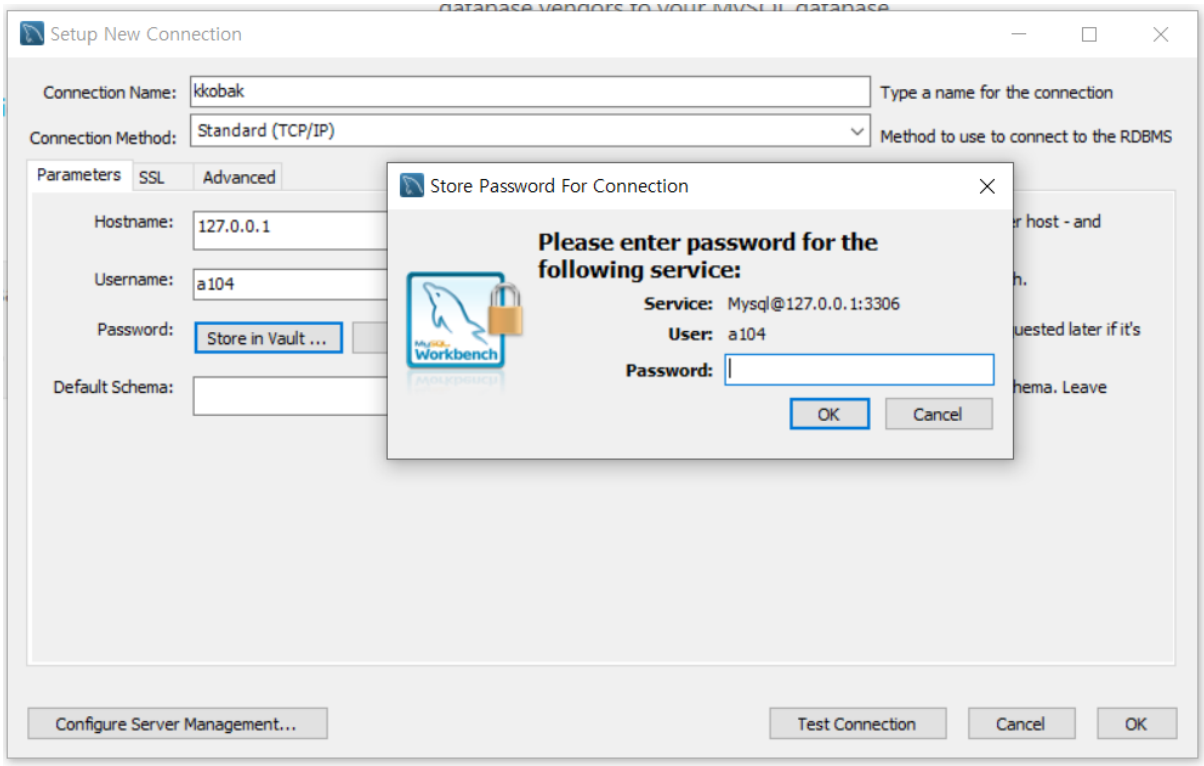
4. DB 계정

가. MySql workbench 추가하기



MySql Workbench를 열어서 새로운 내용을 추가하기 위해 “+” 버튼을 눌러줍니다.

나. EC2 계정 정보 설정



username은 a104, password는 ahdrhchlrh!를 사용하였습니다.

기존 root 계정이 아닌 별도의 계정 a104를 생성하여 프로젝트를 진행하였습니다.

5. 프로퍼티 정의

가) nginx 세팅

- Docker 사용 시에는 /etc/nginx/sites-available로 이동한 후 아래와 같은 파일을 생성합니다.

```
server {         location /{            proxy_pass http://localhost:3000/;          }          location /api{            proxy_pass
http://localhost:8080/;         }               listen 443 ssl;         server_name kkobak.ml;                ssl_certificate /et
c/letsencrypt/live/kkobak.ml/fullchain.pem; # managed by Certbot                ssl_certificate_key /etc/letsencrypt/live/kkobak.m
l/privkey.pem; # managed by Certbot}server {    if ($host = kkobak.ml) {        return 301 https://$host$request_uri;    } # manage
d by Certbot        listen 80;        server_name kkobak.ml;    return 404; # managed by Certbot }
```

- 다음 명령어를 입력합니다.

```
sudo ln -s /etc/nginx/sites-available/[파일명] /etc/nginx/sites-enabled/[파일명]
# 필자의 경우 kkobak.conf

sudo nginx -t

sudo service restart nginx
```

6. 외부 서비스

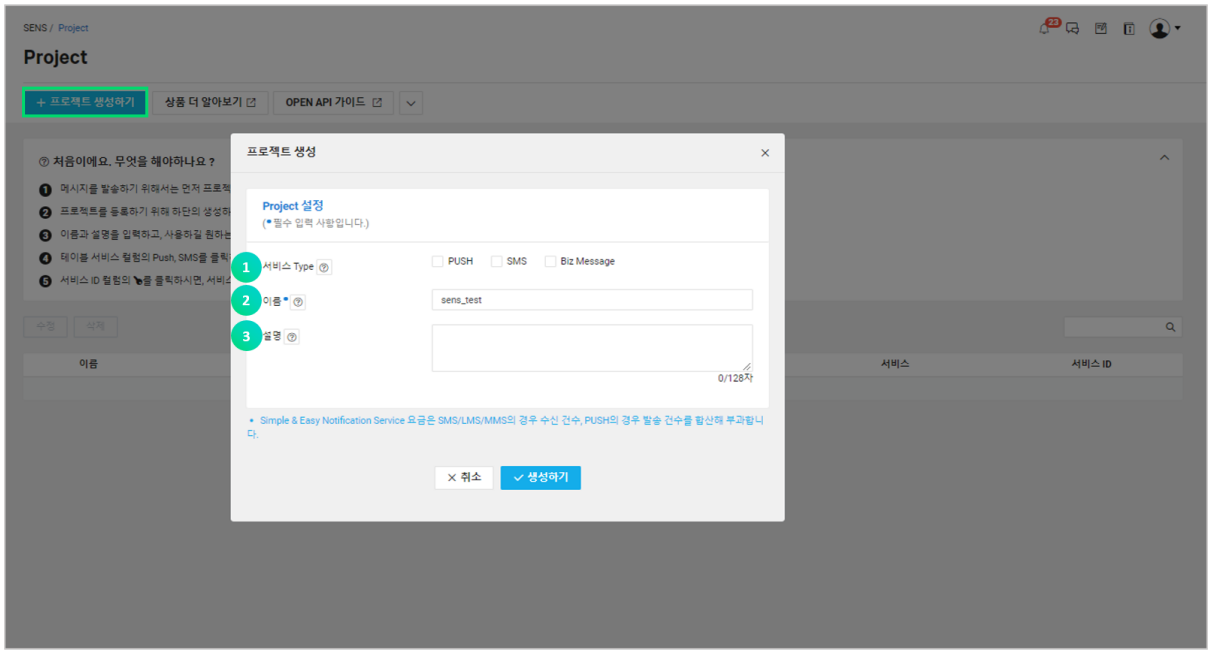
가. 네이버 Simple & Easy Notification Service

인증문자 MMS 기능을 추가하기 위해 네이버 Cloud Platform의 **Simple & Easy Notification Service**를 사용했습니다.
(https://www.ncloud.com/product/applicationService/sens)

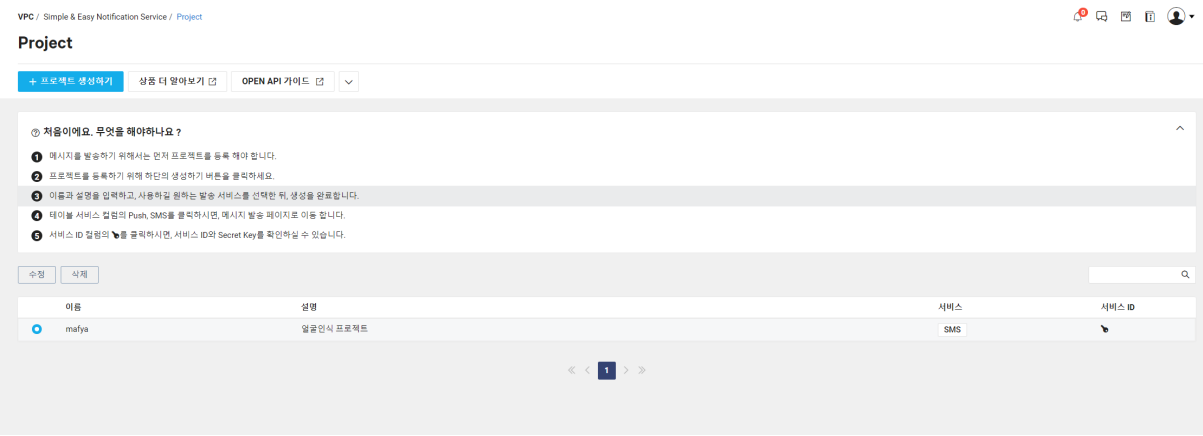
이를 통해 팀원에게 문자 메세지를 전송하는 서비스를 제공 할 수 있게 됩니다.

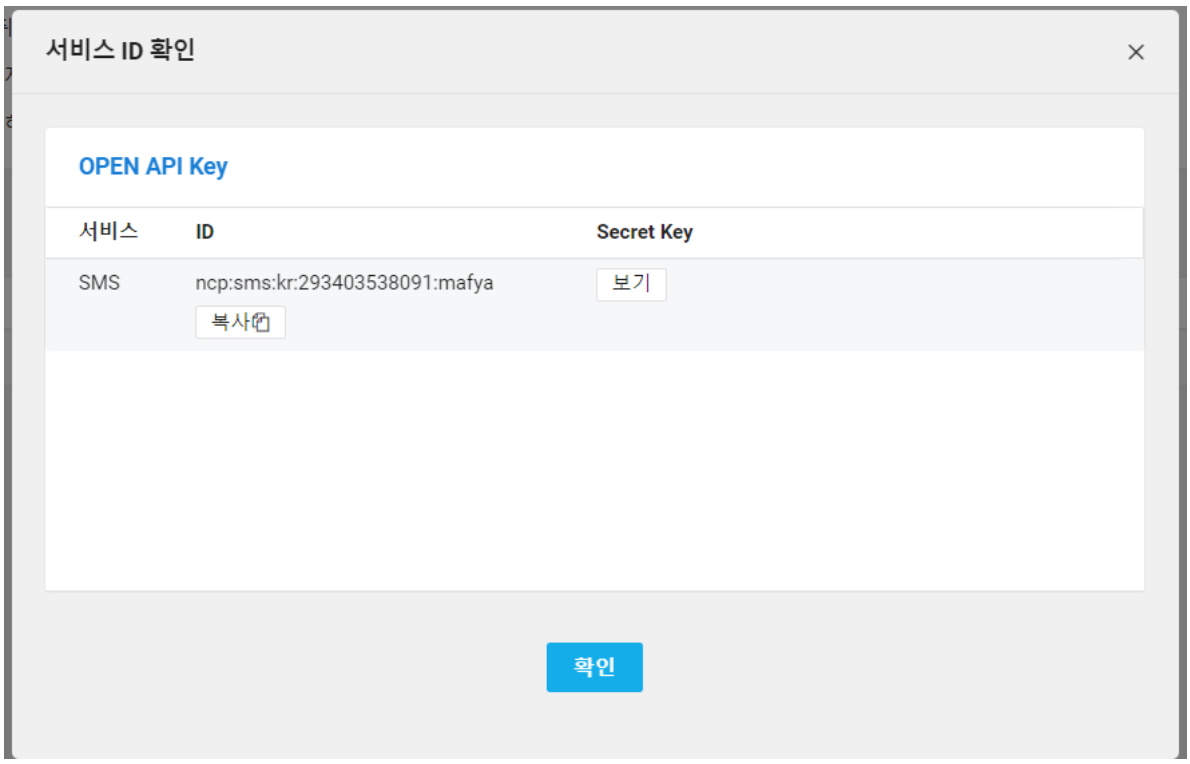
공식 가이드 문서(https://guide.ncloud-docs.com/docs/sens-sens-1-2)

1. **Simple & Easy Notification Service 서비스를 사용하려면 먼저 프로젝트를 생성해야 합니다. 고객의 용도에 따라 여러 개의 프로젝트를 생성하고 사용할 수 있습니다**

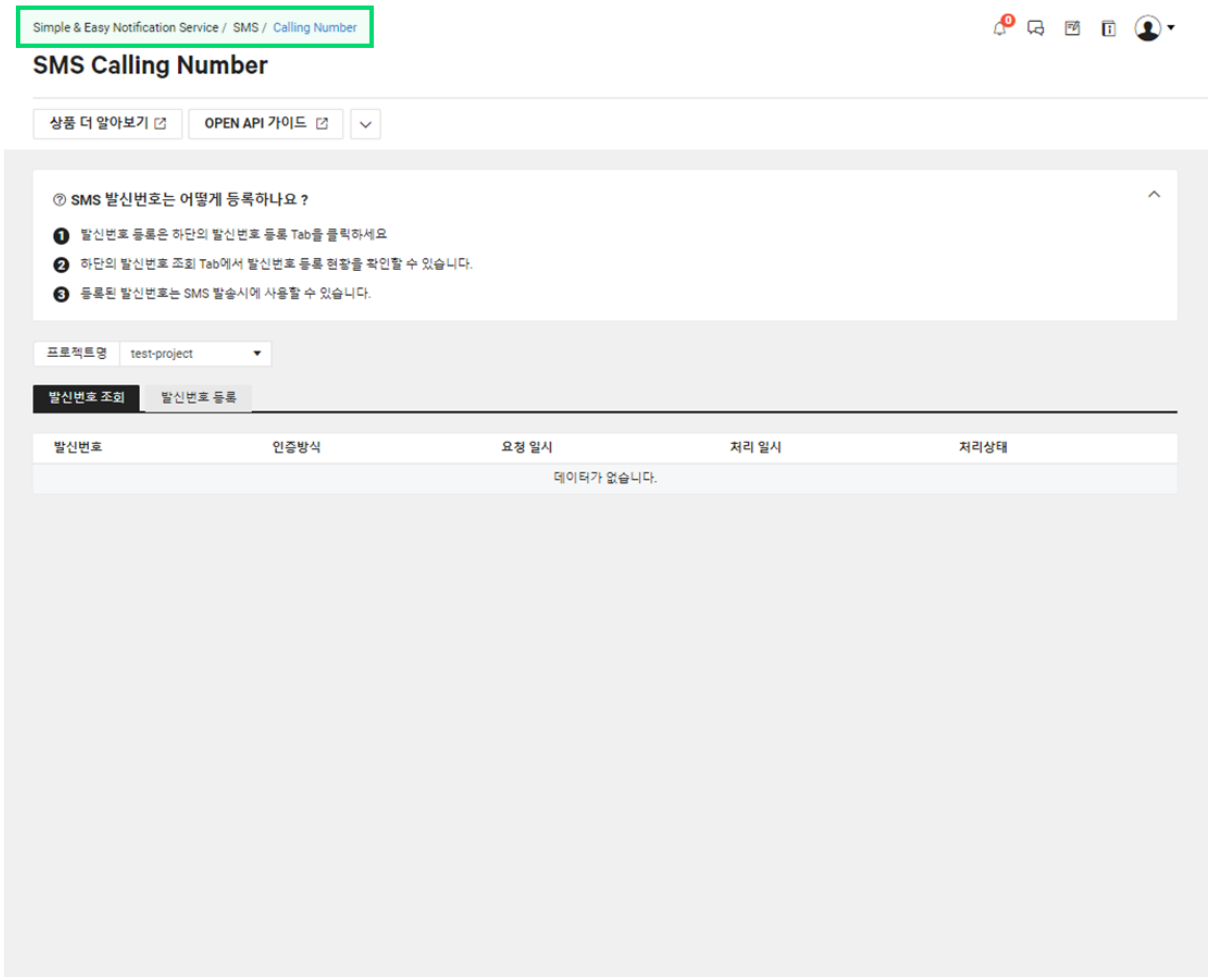


2. **생성한 프로젝트의 ServiceID & SecretKey를 확인합니다.**

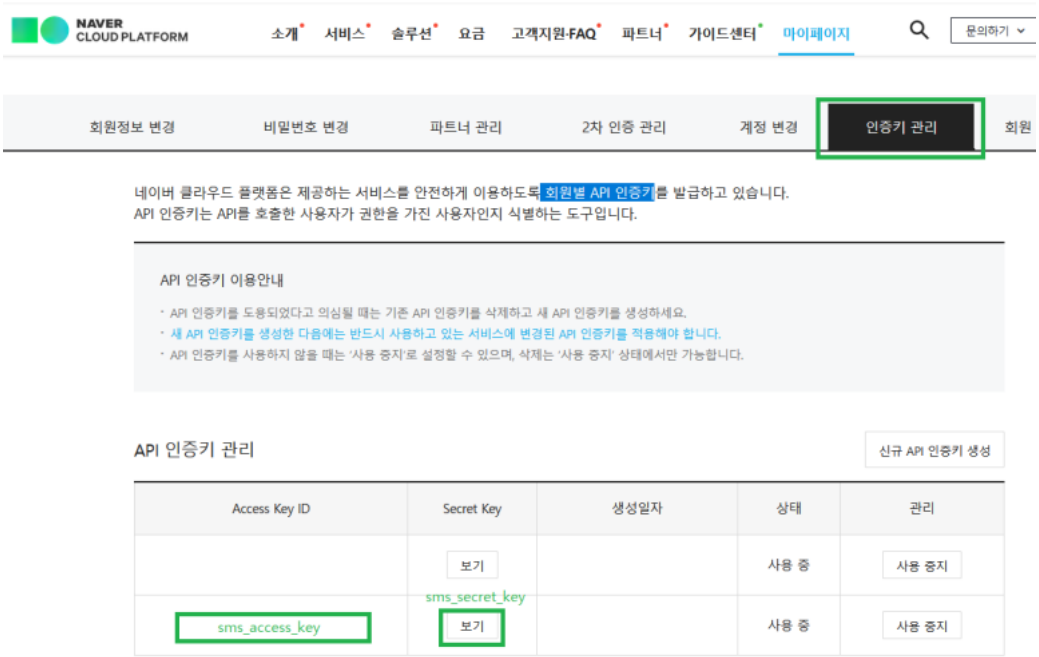




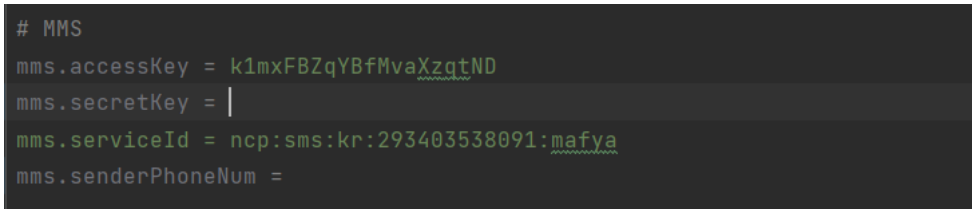
3. **발신번호(senderPhoneNum) 등록하기**



4. **네이버 클라우드 플랫폼의 마이페이지>인증키 관리 에서 AccessKey를 확인합니다.**



5. **Spring project의 application.properties에 MMS 설정에 위의 확인한 키 값들을 넣어줍니다.**



나. 카카오 관련 API (react)

- .env 생성 (생성 위치는 아래 폴더구조 확인 : frontend>kkobak안 위치에 생성하기)
  - .env 안에 들어가야 하는 정보
    - REACT_APP_KAKAOMAP_API={발급받은 키}
    - REACT_APP_KAKAO_REST_API={발급받은 키}
    - REACT_APP_OPENWEATHERMAP_API={발급받은 키}

다. FireBase 관련 http API key

- key=AAAAg7dF6ic:APA91bF9qMg9RmrbUAi8IEnasyAYbkJ_I6dNosRksgN9xdwLq8xz_ZnX2cLgZ84c7Fgsa1LjGWrOJiRGv2VE0YdDWm5ZEmYOTqWJDXu51ezw66O4vu9TSKNRZKuNe-QdlcP