C201: 취미 공유 SNS - 모꼬지

삼성SW청년아카데미 광주캠퍼스 7기

공통프로젝트 **(22.07.05 - 22.08.19)**

포팅 매뉴얼

담당 컨설턴트: 김성재

조성민(팀장), 박세호, 박상현, 임윤혁, 정호진, 허재영

0. 모꼬지란**?**

취미 공유 SNS인 모꼬지는 “여러 사람이 함께 즐기다”라는 순 우리말로 많은 사람들이 다양한 취미를 과정을 밟으며 쉽게 즐길 수 있도록 돕는 웹 서비스 입니다. 사용자는 모꼬지에서 여러 취미를 접할 수 있고 성공 스토리를 쌓을 수 있습니다. 챌린지와 각 챌린지를 구성하는 스테이지를 통해 차근차근 과정을 밟아나갈 수 있습니다. 사용자는 챌린지를 등록하여 타 유저들과 취미를 즐기는 방법도 공유할 수 있습니다. 스테이지에 성공하면 포스팅을 남길 수 있고 타 포스팅에 댓글을 달며 서로의 성공 스토리를 공유합니다.

1. 프로젝트 기술스택

가**.** 이슈관리

- Jira

나**.** 형상관리

- Gitlab

다**.** 커뮤니케이션

- Mattermost
- Notion

라. 개발 환경

- OS: Windows 10
- 사용 **IDE:**
    - IntelliJ IDEA 2022.1.3
    - Visual Studio Code : 1.70.2v
    - UI/UX: Figma

- 백엔드 기술스택

    - Springboot : 2.6.9
    - MariaDB : mariadb Ver 15.1 Distrib 10.3.34-MariaDB, for debian-linux-gnu (x86_64) using readline 5.2
    - AWS : ubuntu 20.04.4 LTS
    - Jenkins : 2.346.2
    - Docker : 20.10.17
    - Openjdk : 11.0.16
    - spring: gradle
    - nginx : nginx/1.18.0 (Ubuntu)

- 프론트엔드 기술스택

    - node.js : 16.15.0v 64bit (LST 버전 사용)
    - npm : 8.5.5v
    - react : 18.2.0v
    - react-router-dom : 6.3.0v

- react-redux : 8.0.2v
- redux Toolkit : 1.8.3v
- TypeScript : 4.7.4v
- Sass : 1.54.4v

2. 프론트엔드 빌드 방법

가**.** 빌드 파일 생성

- 실행 환경을 조성하기 위하여 npm run build를 통해 빌드 파일을 생성합니다.

```
npm run build
```

나**.** 서버 실행

해당 명령어는

- ‘serve라는 웹 서버를 다운받아 실행시킬 때 build라는 디렉토리를 document root’로

하겠다는 뜻으로, 한 번 만 실행시킬 웹 서버를 실행시키는 명령어 입니다.

```
npx serve -s build
```

다. **Dockerfile**

```
FROM node:16.16.0 as builder

# 작업 폴더를 만들고 npm 설치
RUN mkdir /usr/src/app
WORKDIR /usr/src/app
ENV PATH /usr/src/app/node_modules/.bin:$PATH
COPY package.json /usr/src/app/package.json
RUN npm install --silent
RUN npm install react-scripts@5.0.1 -g --silent

# 소스를 작업폴더로 복사하고 앱 실행
COPY . /usr/src/app
CMD ["npm", "start"]
```

배포 명령어 - 박상현

가) AWS EC팅2 DB 세팅

1) MariaDB 패키지 저장소 추가

```
sudo apt-get install apt-transport-https curl
sudo curl -o /etc/apt/trusted.gpg.d/mariadb_release_signing_key.asc
 'https://mariadb.org/mariadb_release_signing_key.asc'
sudo sh -c "echo 'deb https://tw1.mirror.blendbyte.net/mariadb/repo/10.3/ubuntu focal
main'
```

2) MariaDB 설치

```
sudo apt-get update
sudo apt-get install mariadb-server
```

3) root 계정으로 사용자 생성 및 권한 부여

```
sudo mysql -u root -p
CREATE USER '아이디'@'%' IDENTIFIED BY '비밀번호';
GRANT ALL PRIVILEGES ON 데이터베이스.* TO '아이디'@'%';
FLUSH PRIVILEGES;
exit
```

4) 원격 접속을 위한 설정 변경

```
sudo vi /etc/mysql/mariadb.conf.d/50-server.cnf
bind-address = 0.0.0.1 로 변경
```

5) 설정 변경 적용을 위한 재시작

```
sudo systemctl restart mysql
```

6) **default**값 확인

```
mysqld --print-defaults
```

나) **AWS EC2 Docker** 세팅

1) **apt** 업데이트

```
sudo apt-get update
sudo apt-get upgrade
```

2) **Docker**의 **Official GPG Key** 및 저장소 추가

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg
```

3) **Docker Engine** 설치

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

다) **Docker Jenkins** 세팅

1) **Docker Jenkins** 이미지 설치

```
sudo docker pull jenkins:lts
```

2) **Docker Jenkins Volume** 생성

```
sudo docker volume create jenkins-volume
```

3) **Docker Jenkins Image** 실행

```
sudo docker run -d \
-p 8085:8080 \
-v jenkins-volume:/var/jenkins_home \
-v /var/run/docker.sock:/var/run/docker.sock:ro \
-v /var/lib/docker/containers:/var/lib/docker/containers:ro \
--name jenkins \
--network my-network \
jenkins/jenkins:lts
```

4) **Jenkins** 접속 및 기본 설정

sudo docker logs jenkins 에서 **Jenkins** 비밀번호를 복사하고

```
sudo ufw allow 8085                    # aws 방화벽 설정
```

- **sudo ufw allow 8085 # aws** 방화벽 설정

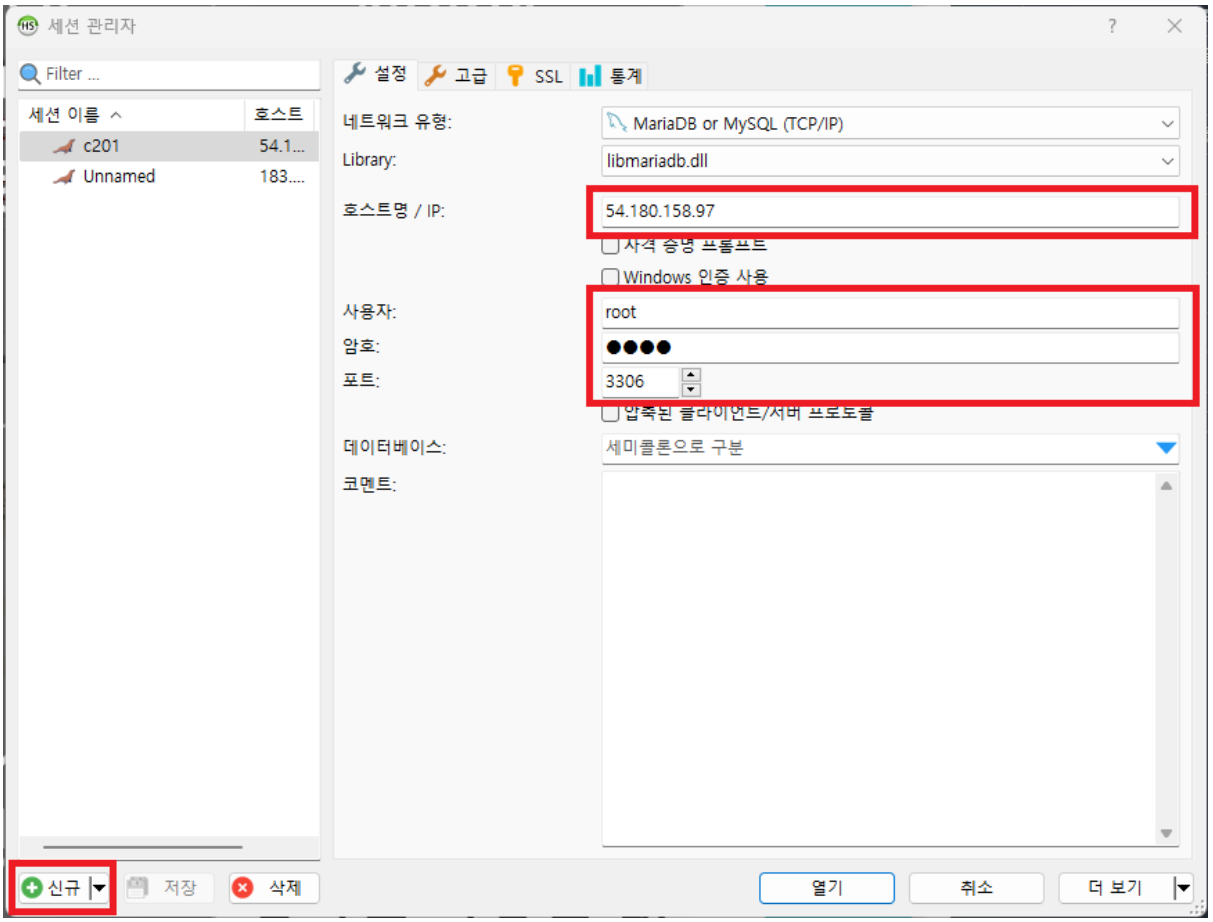
8085 포트로 접속해 복사한 비밀번호 입력 후 설치한 다음 **Admin** 계정 생성

1. **Nginx Default** 값 **-** 박상현

```
server {
        listen 443 ssl default_server;
        listen [::]:443 ssl default_server;
        ssl_certificate /etc/letsencrypt/live/i7c201.p.ssafy.io/fullchain.pem;
        ssl_certificate_key /etc/letsencrypt/live/i7c201.p.ssafy.io/privkey.pem;

        server_name i7c201.p.ssafy.io www.i7c201.p.ssafy.io;

        location / {
                proxy_pass  http://localhost:3000;
        }
        location /api {
                proxy_pass http://localhost:8080;
                proxy_http_version 1.1;
                proxy_set_header Upgrade $http_upgrade;
                proxy_set_header Connection "Upgrade";
                proxy_set_header Host $host;



server {
        listen 443 ssl default_server;
        listen [::]:443 ssl default_server;
        ssl_certificate /etc/letsencrypt/live/i7c201.p.ssafy.io/fullchain.pem;
        ssl_certificate_key /etc/letsencrypt/live/i7c201.p.ssafy.io/privkey.pem;

        server_name i7c201.p.ssafy.io www.i7c201.p.ssafy.io;
```

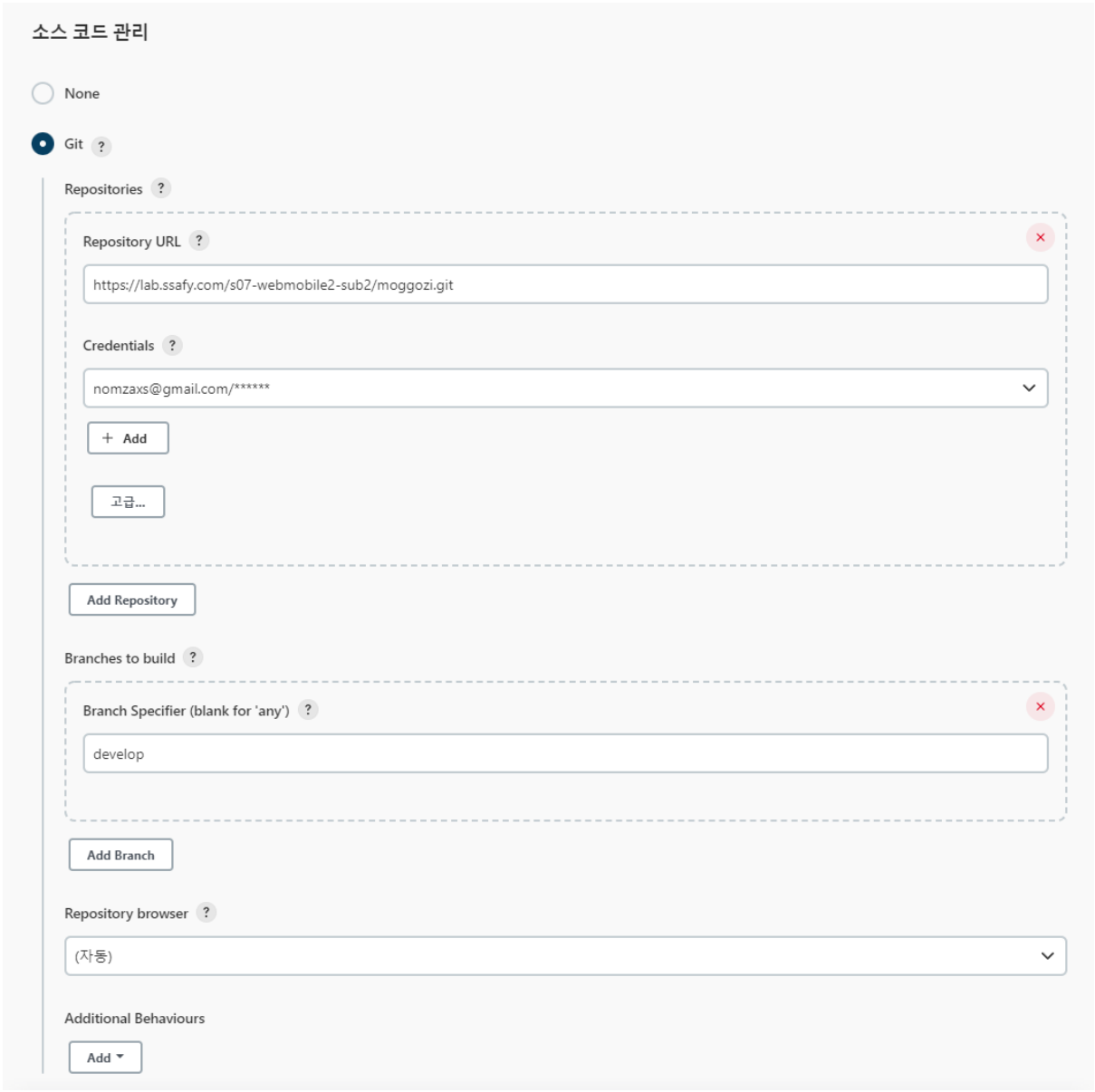
```
        location / {
                proxy_pass  http://localhost:3000;
        }
        location /api {
                proxy_pass http://localhost:8080;
                proxy_http_version 1.1;
                proxy_set_header Upgrade $http_upgrade;
                proxy_set_header Connection "Upgrade";
                proxy_set_header Host $host;
}
}
server {
        listen 80;
        listen [::]:80;

        server_name i7c201.p.ssafy.io;
        # // 서버네임으로 들어오면 https로 연결해줌
        return 301 https://$host$request_uri;
}
```

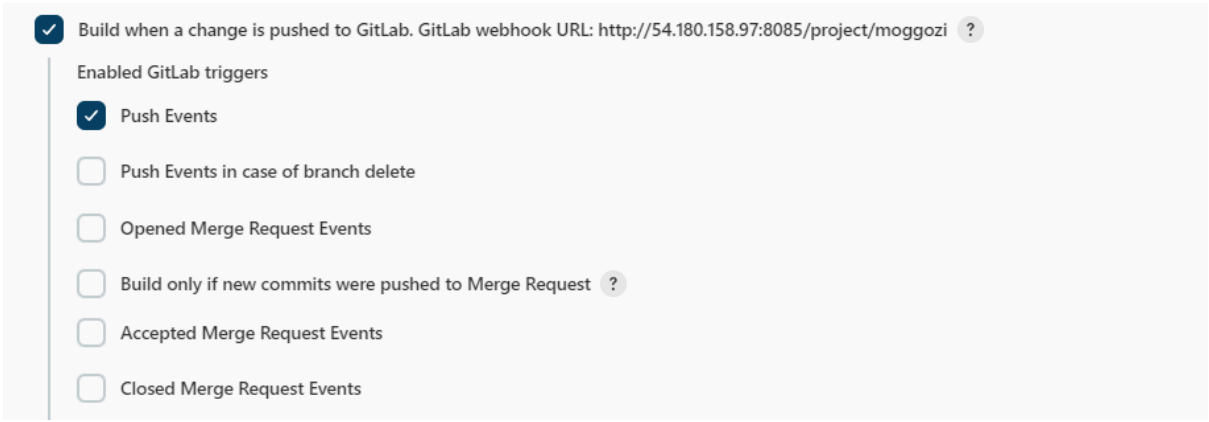
2. **EC2** 세팅 **(Docker, Jenkins, Maria DB** 명령어 포함**) -** 박상현



서버 **ip**와 설정한 **db** 계정과 암호로 접속해 사용한다**.**

Jenkins Git 연동






```
echo '**********Back End Build Start**********'

cd backend/restapiserver
```

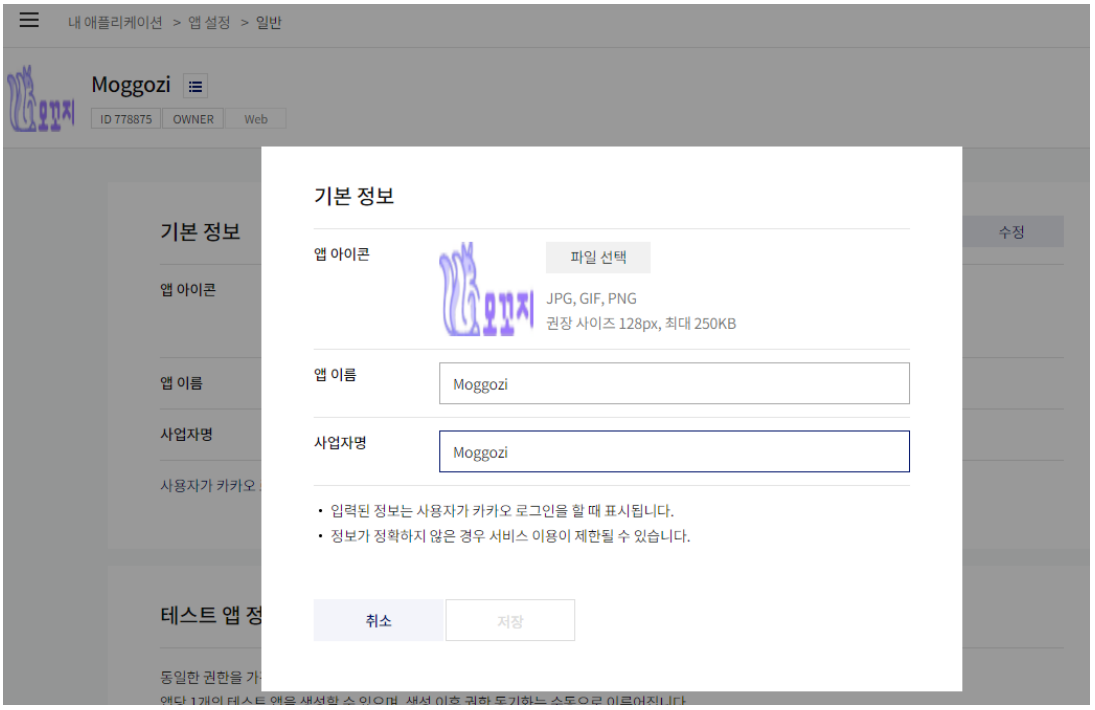
```
chmod +x gradlew

./gradlew clean build

echo '**********Back End Build Done**********'


echo " Execute shell start"


cd ../../frontend


# 기존에 있는 이미지를 삭제합니다.

docker rmi -f react || true


# 도커 빌드

docker build -t react .


echo " Execute shell end"


echo " 빌드 후 조치 start"


# 기존에 있는 컨테이너를 중지합니다.

docker stop react-container || true


# 기존에 있는 컨테이너를 삭제합니다.

docker rm -f react-container || true


# 컨테이너를 설치하고 실행합니다.

docker run -d -p 3000:3000 --name react-container react


echo " 빌드 후 조치 end"
```

빌드 후 프론트는 도커 이미지로

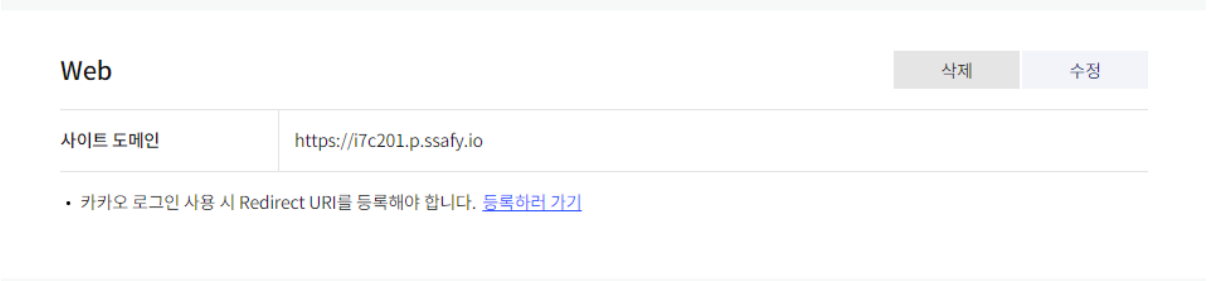
5. 외부 서비스

가. 카카오

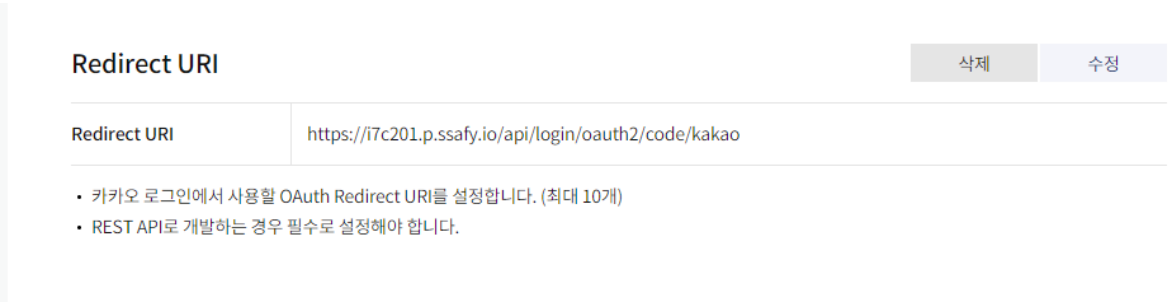
- 서비스의 회원가입/로그인을 위해 카카오 API를 이용하였습니다. 간소한 절차를 통해 서비스를 이용할 수 있어 이용자의 편의성을 높였습니다.
- 카카오 인증 서비스를 이용하기 위해서는 카카오 계정을 통해 Developer로 등록하여야 합니다.
- 사용할 서비스(카카오 로그인)을 선택한 뒤, 기본 정보를 등록합니다.



- 카카오 로그인 시 사용할 도메인과, redirect 도메인을 등록합니다.

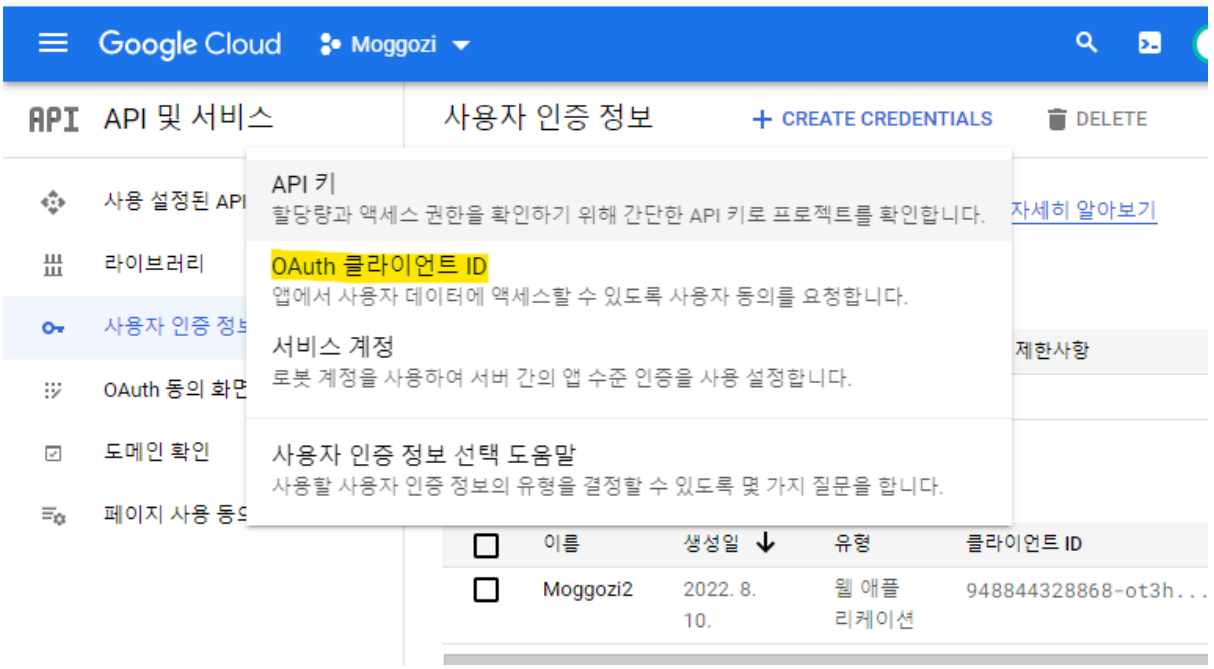


- 리다이렉트 도메인은 카카오로 로그인한 사용자의 인가코드를 받아 구글 서버와 통신하여 액세스토큰, 리프레쉬 토큰을 발급받기 위해 필요한 도메인입니다.

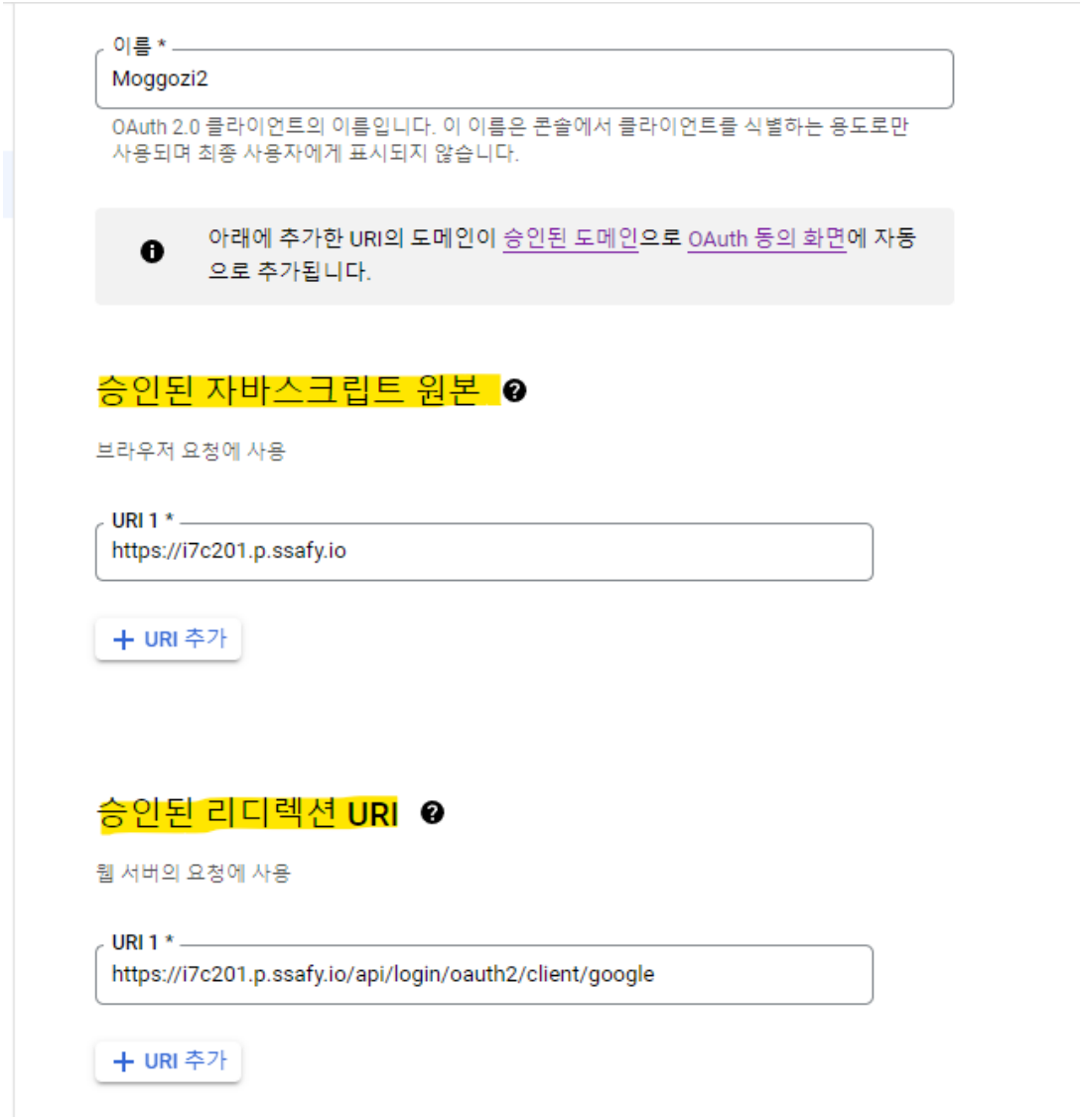


나. 구글

- 서비스의 회원가입/로그인을 위해 구글 API를 이용하였습니다.
- 프로젝트 등록을 한 뒤, OAuth 클라이언트 ID를 등록합니다.
- 승인된 도메인을 사용하지 않을 경우 API는 테스트 버전만 사용가능합니다.
- 이를 위해 도메인을 https로 등록하였습니다.



- 승인된 자바스크립트 원본은 프론트 서버의 주소를 의미합니다.
- 승인된 리다이렉션 URI란, 웹에서 로그인한 사용자의 인가코드를 통해 구글 인증 서버와 액세스 토큰, 리프레시 토큰을 주고받을 수 있는 주소를 의미합니다.



다. 네이버

- 서비스의 회원가입/로그인을 위해 네이버 API를 이용하였습니다.
- 타 API를 이용하는 것과 절차는 같으나 테스트가 아닌 실사용 모드를 위해서는 네이버의 인가가 필요하니다.

3. 파이어베이스 **-** 임윤혁

1) 파이어베이스를 활용한 이유

사진 저장을 외부 스토리지에 저장하여 사용하기 위해 파이어베이스의 **Storage**를 사용합니다**.**

Storage의 폴더 경로를 활용해 사진 이미지 경로를 받아올 수 있어**, DB** 없이 사진을 업로드하고 불러와 사용할 계획이었으나 불러오는 속도가 느려 백엔드 **DB**에 사진 **url**을 저장하여 사용합니다**.**

2) 애플리케이션 추가

2-1) Firebase의 새 프로젝트 생성

2-2) Google 애널리틱스 사용 설정은 끔

2-3) 빌드 **Storage - Storage** 추가

2-4) 프로덕션 모드에서 시작

2-5) 지역을 **asia-northeast3**으로 설정

3) React 프로젝트에서 **Firebase** 사용법

3-1) FE 폴더 **root** 경로에 있는 **.env** 파일에 제품의 **SDK**를 등록

3-2) 타입스크립트를 사용하니 **npm i @types/firebase firebase**로 설치

3-3) .env 파일에 저장된 **key**를 불러올 설정파일을 생성

- **src/fbase/fbase.ts : firebase**라는 이름으로 사용하면 안되니 주의**!**
- 설정 코드

fbase.ts

src > fbase > fbase.ts > ...

```ts
import { initializeApp } from "firebase/app";
import * as firebaseAuth from "firebase/auth";
import { getStorage } from "firebase/storage";

const firebaseConfig = {
  apiKey: process.env.REACT_APP_API_KEY,
  authDomain: process.env.REACT_APP_AUTH_DOMAIN,
  projectId: process.env.REACT_APP_PROJECT_ID,
  storageBucket: process.env.REACT_APP_STORAGEBUCKET,
  messagingSenderId: process.env.REACT_APP_MESSAGING_SENDER_ID,
  appId: process.env.REACT_APP_APP_ID,
};

const app = initializeApp(firebaseConfig);

export const firebaseInstance = firebaseAuth;
export const storageService = getStorage(app, firebaseConfig.storageBucket);
```

4) 사진 저장 및 활용

4-1) storage에 저장할 스냅샷 레퍼런스 **: ref(starageService, <Storage** 경로**/**파일 명**>)**

4-2) 이미지 업로드 **: uploadBytes(**위에서 뽑은 레퍼런스**, image)**

4-3) 파일 다운로드 할 수 있는 **url : getDownLoadURL(**레퍼런스**)**

4-4) 이미지 제거 **: deleteObject(**레퍼런스**)**