포팅 메뉴얼

▼ 목차

Ⅰ. 개요

1. 프로젝트 개요

커뮤니티를 동반한 지역 기반 픽업 주문 플랫폼을 제공해

1. 구매자의 합리적인 소비, 양질의 정보 열람
2. 판매자의 원활한 홍보, 커뮤니티를 통한 다른 가게와의 정보 공유

를 통해 지역 브랜드 형성에 기여한다

2. 프로젝트 사용 도구

- 이슈 관리: JIRA
- 형상 관리: Gitlab
- 커뮤니케이션: Notion, Mattermost, 카카오톡
- 디자인: Figma
- UCC: 프리미어프로
- CI/CD: Docker, Jenkins

3. 개발환경

- VSCode: # 버전 적어주세요

- IntelliJ IDEA 2023.1.3
- JVM: 11
- Node.js: 18.17.0
- 서버: AWS EC2 Ubuntu 20.04 LTS (GNU/Linux 5.4.0-1018-aws x86_64)
- DB: MySQL 8.0

4. 외부 서비스

- Kakao OAuth: `.env` 및 `application.yml` 에 해당 내용 존재
- Kakao map
- Naver OAuth: `.env` 및 `application.yml` 에 해당 내용 존재
- Google OAuth: `.env` 및 `application.yml` 에 해당 내용 존재
- 부트페이 API

5. Gitignore 처리한 핵심 키들

- React: .env (최상단 위치) ⇒ 파일명 및 위치 수정해주세요
- Spring: application.yml (/src/main/resources 에 위치)

II. 빌드

1. Gitlab clone

`GITLAB` 을 통해 관리를 한다.

1. gitlab clone

```
git clone https://lab.ssafy.com/s09-webmobile2-sub2/S09P12B208.git
```

2. 본인 브런치로 이동

- 브런치 명은 front-end의 경우 `f_` , back-end의 경우 `b_` 를 앞에 붙인다.

```
git checkout f_soyeon
git checkout b_soyeon
```

3. 본인 브런치에서 작업 후, back-end는 `backend-master` , front-end는 `frontend-master` 에 merge를 하면 서버에 빌드된다.

2. 환경 변수 형태

- application.yml

```
# 스프링
spring:
  # 데이터베이스
  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
    url: [DB URL]
    username: [유저이름]
```

```
    password: [비밀번호]
  # JPA 설정
  jpa:
    show-sql: true
    database: mysql
    properties:
      hibernate:
        format_sql: true
        dialect: org.hibernate.dialect.MySQL8Dialect
        discriminator:
          ignore_explicit_for_joined: true
    .open-in-view: false
  firebase:
    serviceAccountPath: path/to/serviceAccountKey.json
  # 스프링 시큐리티
  security:
    user:
      name: [spring security 이름]
      password: [spring security 비밀번호]
  # multipartfile 설정
  servlet:
    multipart:
      enabled: true
      max-file-size: 50MB
      max-request-size: 50MB

# JPA log
logging:
  level:
    org:
      hibernate:
        SQL: DEBUG
        type:
          descriptor:
            sql:
              BasicBinder: TRACE

# JWT
jwt:
  secret: [JWT 시크릿키]
  access:
    expiration: 3600000 # 1시간(60분) (1000L(ms -> s) * 60L(s -> m) * 60L(m -> h))
    header: Authorization

  refresh:
    expiration: 1209600000 #  (1000L(ms -> s) * 60L(s -> m) * 60L(m -> h) * 24L(h -> 하루) * 14(2주))
    header: Authorization-refresh

# 서버 설정
server:
  port: [서버 포트]
  max-http-header-size: 2MB

# AWS
cloud:
  aws:
    s3:
      bucket: [aws 버킷이름] # .bucket
    region:
      static: [aws 지역]
      auto: false
    stack:
      auto: false
    credentials:
      access-key: [aws access-key]
      secret-key: [aws 시크릿키]


social:
  kakao:
    rest-api: [카카오 로그인 REST API 키]
    redirect-uri: [카카오 로그인 redirect uri]
  naver:
    client-id: [네이버 로그인 client id]
    client-secret: [네이버 로그인 client 시크릿키]
    redirect-uri: [네이버 로그인 redirect uri]
    state: [네이버 로그인 state]
  google:
    client-id: [구글 로그인 client id]
    client-secret: [구글 로그인 시크릿키]
    redirect-uri: [구글 로그인 redirect uri]
```

3. 빌드하기

1) Frontend

```
npm i
npm run build
```

2) Backend

```
Gradle 실행
```

4. 배포하기

1) 우분투 서버 안으로 들어오기

```
ssh -i [PEM KEY] ubuntu@[서버명]
```

본인의 PEM 키가 존재하는 디렉토리, 혹은 경로를 적으면 우분투 서버 안으로 들어올 수 있다.

2) Docker 설치

Docker 공식 사이트를 참고하여 Docker를 설치한다.

3) Docker 내에 MySQL 설치

1. 우선 Docker 내에 MySQL 이미지를 생성해준다.

```
version: '3'  # 파일 규격 버전

services:
  mysql:  # 서비스 명
    image: mysql:latest  # 사용할 이미지
    container_name: [MySQL 컨테이너명] # 컨테이너 이름 설정
    environment:
      MYSQL_DATABASE: [DB 명] # db 이름
      MYSQL_ROOT_PASSWORD: [DB 비밀번호]
      TZ: Asia/Seoul
    command:
      - --lower_cate_table_names=1  # 대소문자 구분
      - --character-set-server=utf8mb4
      - --collation-server=utf8mb4_unicode_ci
    volumes:
      - ./db:/var/lib/mysql  # db 볼륨 처리
    ports:
      - [DB 포트번호]:3306  # 접근 포트 설정 (외부:내부)
    restart: always
```

2. MySQL에 사용자를 추가하고 권한을 부여한다.

```
docker exec -it [MySQL 컨테이너명] bash
```

명령어를 통해 mysql-container를 bash로 접속하고

```
mysql -u root -p
```

명령어를 통해 mysql shell에 접속한다.

여기서 비밀번호를 입력하라고 되어 있는데 아마 본인이 설정한 mysql 비밀번호일 것이다. 나는 `ssafy` 로 설정했다.

```
mysql> CREATE USER 'abc'@'%' IDENTIFIED BY '1234';
Query OK, 0 rows affected (0.02 sec)

mysql> GRANT ALL PRIVILEGES ON *.* TO 'abc'@'%';
Query OK, 0 rows affected (0.02 sec)

mysql> flush privileges;
Query OK, 0 rows affected (0.01 sec)

mysql> quit
Bye
```

이렇게 해서 `abc` 라는 유저에게 `1234` 라는 비밀번호를 주고, 권한을 주는 것이다.

3. MySQL Workbench에서 접속

   Connection 등록에 서버URL, 포트번호, 위에서 설정한 Hostname을 입력한다.

   `Test Connection` 을 누르고, 위에서 설정한 비밀번호를 입력하면 연결이 성공하였음을 확인할 수 있다.

4) Docker 내에 Jenkins 설치

1. 먼저 젠킨스 이미지를 내려받는다.

```
version: '3'  # 파일 규격 버전

services:
  jenkins:
    image: jenkins/jenkins:lts
    container_name: jenkins
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock
      - /jenkins:/var/jenkins_home
    ports:
      - "[포트번호]:8080"
    user: root
```

2. 그런 후, 젠킨스 이미지를 실행한다.

   이 때, 젠킨스 안에서 docker를 실행하기 위해서는 특별한 볼륨이 필요하다. 따라서 아래와 같은 볼륨을 추가하여 젠킨스 내에 docker를 설치할 수 있도록 한다.

```
/var/run/docker.sock:/var/run/docker.sock
```

   만약 여기서 위와같은 볼륨을 추가하지 않았다면

   ⚠️ Cannot connect to the Docker daemon at unix:/ //var/run/docker.sock. Is the docker daemon running?

   과 같은 에러가 발생한다. 따라서 이러한 에러가 발생하였다면 젠킨스를 삭제한 후, 재설치를 한다.

3. 젠킨스 내에서 필요한 plugins 들을 모두 설치해준다.

   본 프로젝트에서는 기본으로 설치하라고 추천해주는 plugins 들 및 다음과 같은 plugins 들을 설치해주었다.

   ▼ 플러그인 목록

플러그인	버전
Ant Plugin	**497.v94e7d9fffa_b_9**

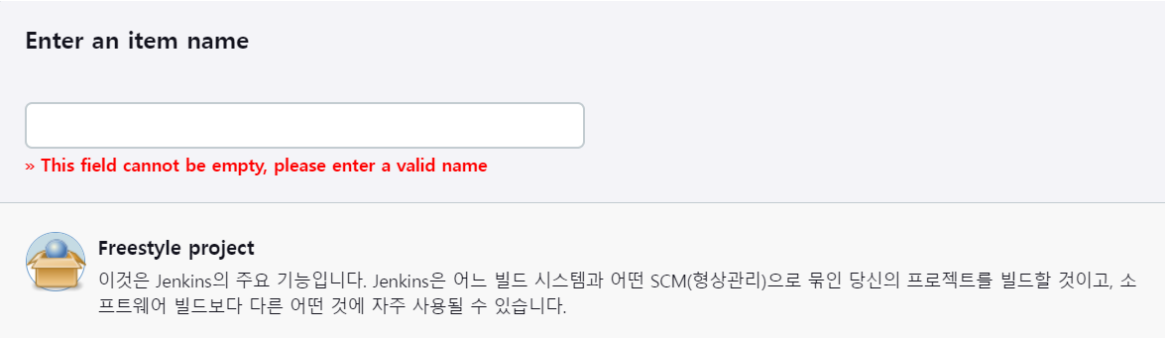
플러그인	버전
Apache HttpComponents Client 4.x API Plugin	4.5.14-150.v7a_b_9d17134a_5
Bootstrap 5 API Plugin	5.3.0-1
bouncycastle API Plugin	2.29
Branch API Plugin	2.1122.v09cb_8ea_8a_724
Build Timeout	1.31
Caffeine API Plugin	3.1.6-115.vb_8b_b_328e59d8
Checks API plugin	2.0.0
Command Agent Launcher Plugin	106.vb_a_b_8f751309c
commons-lang3 v3.x Jenkins API	3.12.0-36.vd97de6465d5b_
commons-text API Plugin	1.10.0-68.v0d0b_c439292b_
Credentials Binding Plugin	631.v861c06d062b_4
Credentials Plugin	1271.v54b_1c2c6388a_
Display URL API	2.3.8
Durable Task Plugin	513.vc48a_a_075a_d93
ECharts API Plugin	5.4.0-5
Email Extension Plugin	2.100
Folders	6.815.v0dd5a_cb_40e0e
Font Awesome API Plugin	6.4.0-2
Generic Webhook Trigger Plugin	1.86.5
Git client plugin	4.4.0
Git plugin	5.2.0
GitHub	1.37.2
GitHub API Plugin	1.314-431.v78d72a_3fe4c3
GitHub Branch Source	1730.vff97c8a_1f804
GitLab API Plugin	5.3.0-91.v1f9a_fda_d654f
GitLab Authentication plugin	1.18
GitLab Branch Source Plugin	664.v877fdc293c89
Gitlab Merge Request Builder	2.0.0
GitLab Plugin	1.7.15
Gradle Plugin	2.8.2
Handy Uri Templates 2.x API Plugin	2.1.8-22.v77d5b_75e6953
Instance Identity	173.va_37c494ec4e
Ionicons API	56.v1b_1c8c49374e
Jackson 2 API Plugin	2.15.2-350.v0c2f3f8fc595
Jakarta Activation API	2.0.1-3
Jakarta Mail API	2.0.1-3
Java JSON Web Token (JJWT) Plugin	0.11.5-77.v646c772fddb_0
JavaBeans Activation Framework (JAF) API	1.2.0-6
JavaMail API	1.6.2-9
JAXB plugin	2.3.8-1
Jersey 2 API	2.40-1
JQuery3 API Plugin	3.7.0-1
JUnit Plugin	1217.v4297208a_a_b_ce
LDAP	682.v7b_544c9d1512
Mailer Plugin	463.vedf8358e006b_
Matrix Authorization Strategy Plugin	3.1.10
Matrix Project Plugin	802.v8013b_40c7edc

플러그인	버전
Mina SSHD API :: Common	2.10.0-69.v28e3e36d18eb_
Mina SSHD API :: Core	2.10.0-69.v28e3e36d18eb_
OkHttp	4.11.0-145.vcb_8de402ef81
Oracle Java SE Development Kit Installer Plugin	73.vddf737284550
OWASP Markup Formatter Plugin	162.v0e6ec0fcfcf6
PAM Authentication plugin	1.10
Pipeline	596.v8c21c963d92d
Pipeline Graph Analysis Plugin	202.va_d268e64deb_3
Pipeline: API	1251.vd4889a_b_0a_065
Pipeline: Basic Steps	1042.ve7b_140c4a_e0c
Pipeline: Build Step	505.v5f0844d8d126
Pipeline: Declarative	2.2144.v077a_d1928a_40
Pipeline: Declarative Extension Points API	2.2144.v077a_d1928a_40
Pipeline: GitHub Groovy Libraries	42.v0739460cda_c4
Pipeline: Groovy	3731.ve4b_5b_857b_a_d3
Pipeline: Groovy Libraries	671.v07c339c842e8
Pipeline: Input Step	477.v339683a_8d55e
Pipeline: Job	1316.vd2290d3341a_f
Pipeline: Milestone Step	111.v449306f708b_7
Pipeline: Model API	2.2144.v077a_d1928a_40
Pipeline: Multibranch	756.v891d88f2cd46
Pipeline: Nodes and Processes	1278.v94b_dc2b_50c6f
Pipeline: REST API Plugin	2.33
Pipeline: SCM Step	415.v434365564324
Pipeline: Stage Step	305.ve96d0205c1c6
Pipeline: Stage Tags Metadata	2.2144.v077a_d1928a_40
Pipeline: Stage View Plugin	2.33
Pipeline: Step API	639.v6eca_cd8c04a_a_
Pipeline: Supporting APIs	848.v5a_383b_d14921
Plain Credentials Plugin	143.v1b_df8b_d3b_e48
Plugin Utilities API Plugin	3.3.0
Resource Disposer Plugin	0.23
SCM API Plugin	676.v886669a_199a_a_
Script Security Plugin	1264.vecf66020eb_7d
SnakeYAML API Plugin	1.33-95.va_b_a_e3e47b_fa_4
SSH Build Agents plugin	2.916.vd17b_43357ce4
SSH Credentials Plugin	308.ve4497b_ccd8f4
SSH server	3.312.v1c601b_c83b_0e
Structs Plugin	324.va_f5d6774f3a_d
Timestamper	1.26
Token Macro	359.vb_cde11682e0c
Trilead API Plugin	2.84.v72119de229b_7
Variant Plugin	59.vf075fe829ccb
Workspace Cleanup Plugin	0.45

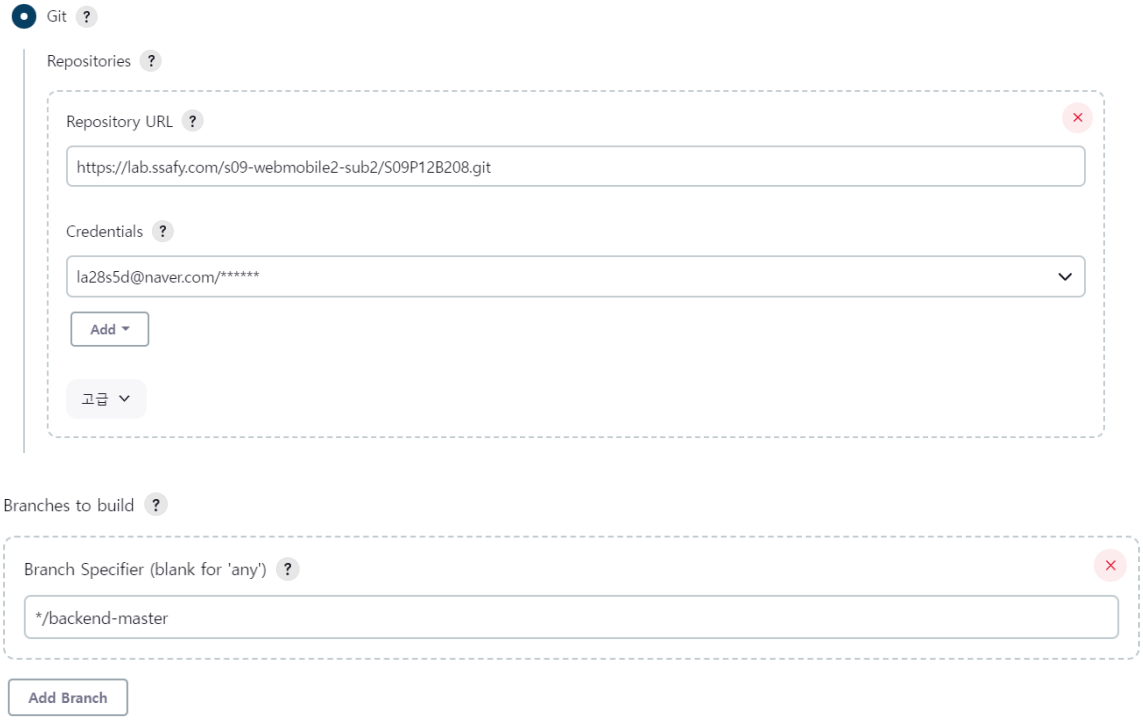
4. 젠킨스 내에 Backend Item, Front Item을 생성한다.

   해당 프로젝트에서는 파이프라인이 아닌 Execute Shell 방식을 사용하였다.

   i. 이름을 지정하고, `Freestyle project` 를 선택해준다.



   ii. 깃 설정을 해주어야 하는데, 이 때, Repository URL에는 해당 Gitlab 링크, 그리고 이에 접근할 수 있는 계정을 설정해준다.



   또한, 어떤 브런치로 이동하여 빌드할 것인지도 설정해준다.

   iii. 자동 재배포를 위해 빌드유발에

   > ✅ Build when a change is pushed to Gitlab webhook URL : XXX

   를 체크해준다. 이 때, `더보기` 를 눌러 `secret key` 를 발급받아야 한다.

   iv. 본 프로젝트에서는 Build Steps 를 Execute shell을 사용해주었다. 따라서 빌드에 필요한 명령어를 작성해준다.

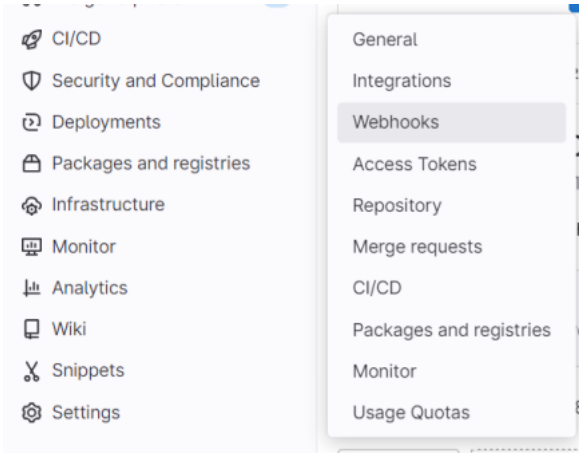
```
cd [스프링 프로젝트 명]
chmod +x gradlew # 권한 설정
./gradlew clean build # 빌드 설정
docker stop [컨테이너명] # 현재 올라가있는 백엔드 서버 멈추기
docker rm [컨테이너명] # 백엔드 서버 삭제
docker rmi [이미지명] # 백엔드 서버 이미지 삭제
docker build -t [이미지명]:latest . # 해당 이름으로 재배포
docker run -d -p [들어올 포트]:[서버 포트] --name [컨테이너명] [이미지명]:latest # 도커를 실행하여 서버를 구동
```

```
cd [React 프로젝트 명]
docker rm -f [이미지명]
docker build -t [이미지명]:latest .
docker run -d -p [들어올 포트]:[서버 포트] --network square-network --name [컨테이너명] [이미지명]:latest
```

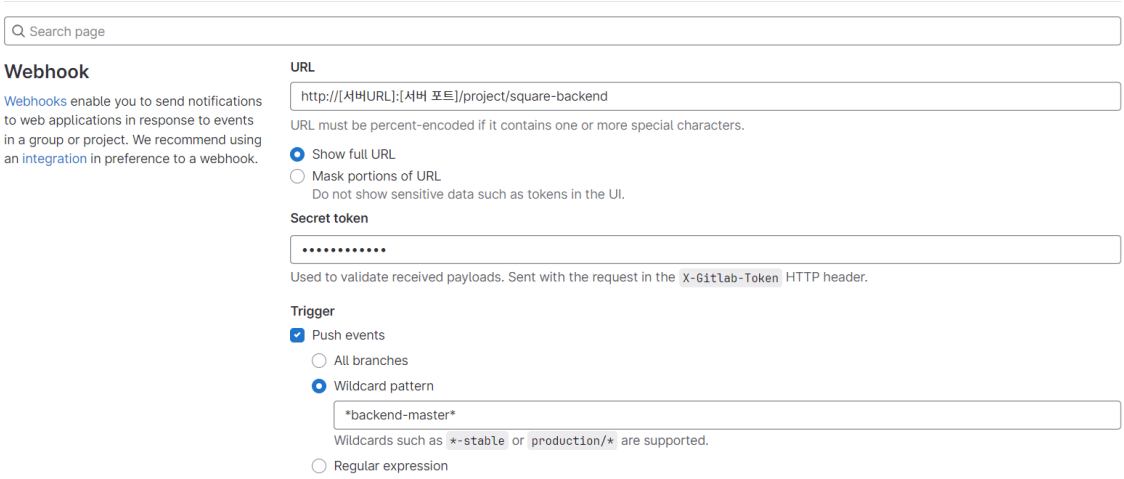
다음과 같은 방법은 이미 있는 이미지를 일단 삭제한 후, 실행해주므로 임의의 동일명의 컨테이너 및 이미지를 생성해준 후 실행한다.

v. Webhook 설정

본인의 Gitlab에 들어가면 Settings → Webhooks가 존재한다.



여기서 Webhooks을 설정할 수 있는데,



다음과 같이 URL 설정 및 secret token을 설정해준다. secret key는 위에서 더보기를 눌러 발급받은 secret key 값을 넣어준다. trigger에는 어떤 브런치에서 push 또는 merge 이벤트가 발생하면 jenkins 에 요청을 보낼지에 대한 설정이다.(이 부분은 확실치 않지만, 다음과 같이 설정하니 정상 작동하였다.)

위와 같은 방식으로 백엔드와 프론트엔드를 도커에 올려주어 실행해주면 된다. (자동으로 실행된다.)

III. 외부 API 이용방법

1. 카카오 로그인 API (Kakao OAuth)

1. 준비: 카카오 개발자센터 어플리케이션 등록
    a. 내 애플리케이션 → 애플리케이션 등록

2. 사이트도메인 등록

   내 어플리케이션 > 앱 설정 > 플랫폼 > Web에 사이트 도메인을 등록한다.

   ```
   https://i9b208.p.ssafy.io
   http://localhost:3000
   ```

3. redirect uri 설정

   내 어플리케이션 > 제품 설정 > 카카오 로그인에 Redirect URI를 등록한다.

   ```
   [프론트에서 설정한 redirect uri]
   ```

   이 redirect uri 를 작성하고, 백과 프론트에서 연결할 때에는 두 주소가 동일하여야 한다.

4. backend — application.yml

   ```
   social:
     kakao:
       rest-api: [내 어플리케이션 > 앱 설정 > 요약정보 > REST API 키]
       redirect-uri: [위에서 설정한 redirect uri]
   ```

5. frontend — .env

   ```
   const kakaoLogin = (): void => {
       // 카카오 OAuth2.0 인증 페이지로 리다이렉트
       window.location.href =
         "https://kauth.kakao.com/oauth/authorize?client_id=409915cf48a47370a92cea926084d5a1&redirect_uri=https://i9b208.p.ssafy.io/l

     };
   ```

2. 네이버 로그인 API (Naver OAuth)

1. 준비: 네이버 개발자센터 어플리케이션 등록
    a. Application → 애플리케이션 등록

2. API 설정

   Application > 내 애플리케이션에 설정을 한다.

   - 사용 API 설정

     우리 서비스는 회원이름, 연락처 이메일 주소, 별명, 휴대전화번호를 필수로 하였다.

   - 로그인 오픈 API 서비스 환경

     서비스환경의 경우에는 PC 웹을 설정하였다.

```
https://i9b208.p.ssafy.io
```

```
[프론트에서 설정한 redirect uri]
```

이 redirect uri 를 작성하고, 백과 프론트에서 연결할 때에는 두 주소가 동일하여야 한다.

3. backend — application.yml

```
social:
  naver:
    client-id: [Application > 내 어플리케이션 > 개요 > 애플리케이션 정보 > Client ID]
    client-secret: [Application > 내 어플리케이션 > 개요 > 애플리케이션 정보 > Client Secret]
    redirect-uri: [위에서 설정한 redirect uri]
    state: [임의의 state 값. 프론트와 백이 동일하여야 한다.]
```

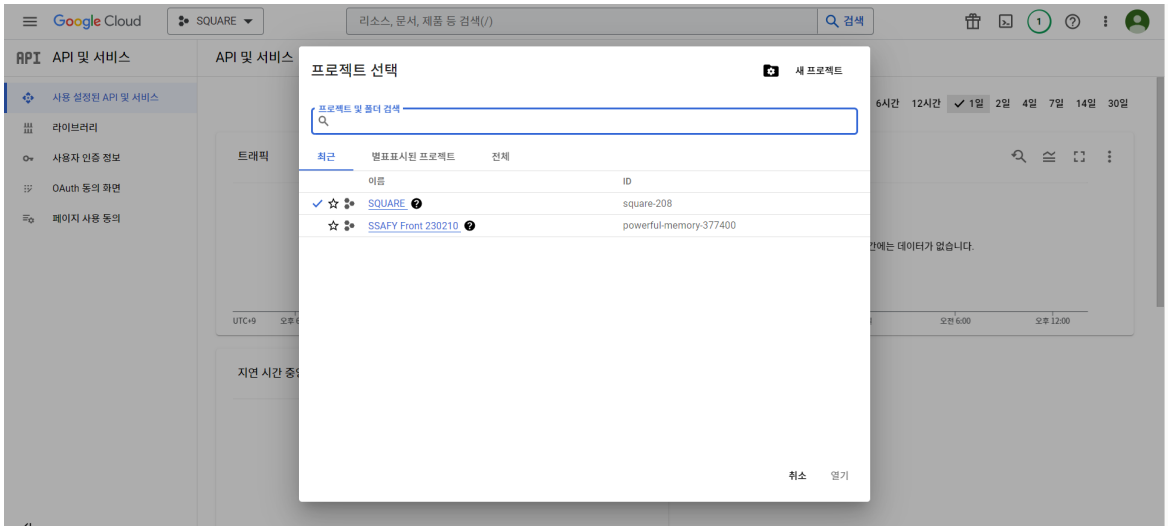
5. frontend — .env

```
const NaverLogin = (): void => {
    window.location.href =
      "https://nid.naver.com/oauth2.0/authorize?client_id=C4jdFBfefASIcQgC9GDg&response_type=code&redirect_uri=https://i9b208.p.ssafy.
  };
```

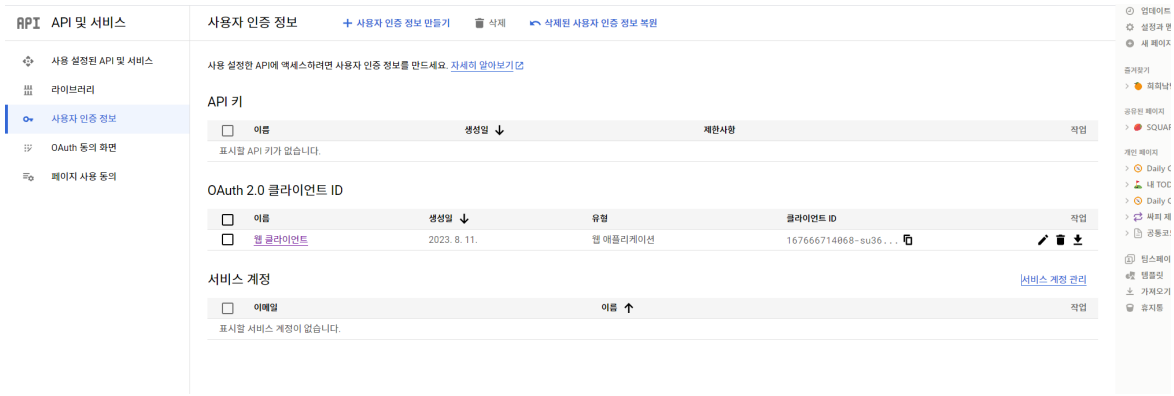
3. 구글 로그인 API (Google OAuth)

1. 어플리케이션 등록

   구글 API에 API 및 서비스에 들어가 프로젝트를 생성한다.



2. OAuth 클라이언트 등록



3. 웹 클라이언트 내 승인된 리디렉션 URI 설정

4. backend — application.yml

```
social:
  google:
    client-id: [API 및 서비스 > 사용자 인증정보 > 웹 어플리케이션의 클라이언트 ID > 클라이언트 ID]
    client-secret: [API 및 서비스 > 사용자 인증정보 > 웹 어플리케이션의 클라이언트 ID > 클라이언트 보안 비밀번호]
    redirect-uri: [위에서 설정한 redirect uri]
```

5. frontend — .env

```
# 추후 추가
```

4. 카카오맵 API

1. 어플리케이션 등록

Kakao 지도 Javscript API 는 키 발급을 받아야 사용할 수 있습니다.
그리고 키를 발급받기 위해서는 카카오 계정이 필요합니다.

키 발급에는 아래 과정이 필요합니다.

1. 카카오 개발자사이트 (https://developers.kakao.com) 접속
2. 개발자 등록 및 앱 생성
3. 웹 플랫폼 추가: 앱 선택 – [플랫폼] – [Web 플랫폼 등록] – 사이트 도메인 등록
4. 사이트 도메인 등록: [웹] 플랫폼을 선택하고, [사이트 도메인] 을 등록합니다. (예: http://localhost:8080)
5. 페이지 상단의 [JavaScript 키]를 지도 API의 appkey로 사용합니다.
6. 앱을 실행합니다.

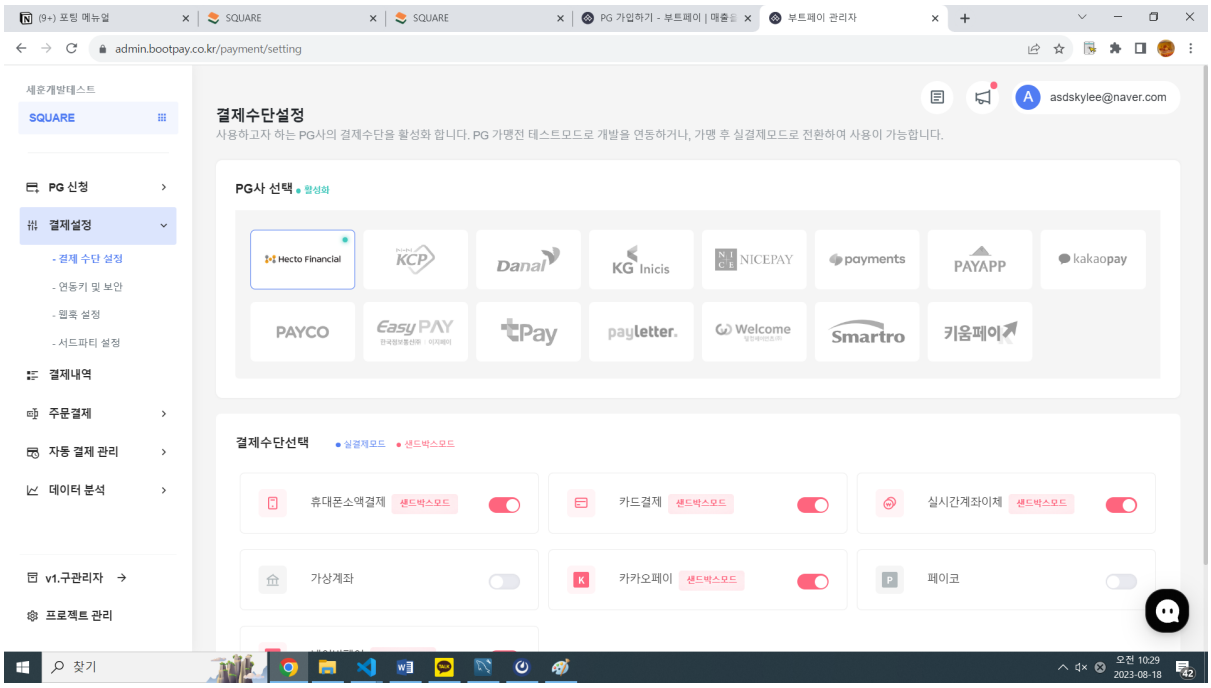
- 등록한 도메인(예: http://localhost:8080)에서 웹 서버를 실행시켜 위 파일을 엽니다.

2. React index.html에 script 입력

```
"//dapi.kakao.com/v2/maps/sdk.js?appkey={본인의JavaScript앱키}&autoload=false&libraries=services,clusterer,drawing
```

5. 부트페이 API

1. 부트페이 사이트 가입 후 APP키 받기
2. PG사 선택 후 '테스트모드(혹은 샌드박스모드)로 등록



이후 아래와 같이 부트페이 통합모듈 호출

```
Bootpay.requestPayment({
          application_id: {본인의APP키},
          price: finalPrice,
          order_name: "SQUARE 주문서",
          order_id: `${location.state.storeName}CODE${Math.floor(
            Math.random() * 10000000
          )}`,
          user: {
            id: "회원아이디",
            username: "회원이름",
            phone,
          },
          items: [finalItem],

          extra: {
            open_type: "iframe",
            display_error_result: true,
            card_quota: "0,2,3",
            escrow: false,
          },
```