

포팅 매뉴얼 (Porting Manual)



삼성 청년 **SW** 아카데미 7기 광주캠퍼스

공동프로젝트

22.07.12 ~ 22.08.19 (6주)

C103 - 여기델까

담당 컨설턴트 : 김성준

유연길(팀장), 박주윤, 오승준, 이성조, 장우주

목차

1. 기술 스택 -----	1
2. AWS EC2 Docker 설정 -----	2
3. AWS EC2 배포를 위한 Nginx 설정 -----	3
4. 외부 서비스 - Tmap API -----	4
5. 시연 시나리오 -----	5

1. 기술 스택

- a. 프로젝트 Story 및 Issue 관리 : Jira
- b. 코드 관리 : GitLab
- c. 팀 커뮤니케이션 : Metamost, Webex
- d. 프로젝트 일정표 관리 : Notion
- e. 개발 환경
 - i. OS 환경 : Window, Ubuntu 20.04
 - ii. IDE
 - 1. VS Code : 1.70.2
 - 2. UI/UX : Figma
 - iii. DataBase : MySQL WorkBench 8.0.29
 - iv. Server : Amazon AWS EC2
 - 1. OS : Ubuntu 20.04.1 LTS
 - 2. Docker : 20.10.12
 - 3. Jenkins : 2.346.3
 - 4. Nginx : 1.18.0
 - v. SSH Client 프로그램 : MobaXterm V22.0
 - vi. 분야별 상세 설명
 - 1. Frontend
 - a. react : 18.2.0
 - b. node : 16.15.1
 - c. react-dom : 18.2.0
 - d. react-router-dom : 6.3.0
 - e. redux : 4.2.0

- f. react-redux : 8.0.2
 - g. styled-components : 5.3.5
 - h. MUI : 5.10.0
 - i. axios : 0.27.2
- 2. Backend
 - a. ?
 - b. ?
- 3. Embedded
 - a. Python : 3.10.4
 - b. opencv-python : 4.6.0.66
 - c. numpy : 1.23.1
 - d. matplotlib : 3.5.2
 - e. mysql-connector : 2.2.9
 - f. mysql-connector-python : 8.0.29

2. AWS EC2 Docker 설정

- 1) AWS EC2 Ubuntu 20.04.2 LTS 환경의 필수 패키지 설치

```
sudo apt-get update
```

```
sudo apt-get install build-essential
```

```
sudo apt-get upgrade
```

- 2) 도커 설치

```
sudo apt-get install docker -y
```

```
docker -v
```

>> 설치된 도커 버전 확인

- 3) Jenkins Image 설치

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

```
docker image
```

>> docker jenkins image 확인

- 4) 설치된 jenkins image를 docker container로 띄우기

```
docker run -d -p 9090:8080 -v /jenkins:/var/jenkins_home --name jenkins -u root  
jenkins/jenkins:lts
```

>> -d : 백그라운드 모드
-p : 로컬 Port : 컨테이너 Port
-v : 디렉터리 연결
-u : 실행할 사용자 지정
-name : 컨테이너 이름 설정

`docker ps -a`

>> docker container 목록 확인

5) Jenkins 접속

웹 브라우저 주소에 [host ip:port] 를 입력하여 접속합니다. ([AWS EC2 IP:9090] 사용)

6) Jenkins에 GitLab 연동

a) 필요한 플러그인 설치

Jenkins 관리 > 플러그인 관리 > GitLab Plugin & Publish over SSH 설치 및 확인 > Dashboard로 돌아가기

b) GitLab 접근을 위한 아이디/비밀번호 및 토큰 저장

1. GitLab 아이디/비밀번호 저장

Jenkins 관리 > Manage Credentials > (global) 클릭 > Add Credentials 클릭 > Kind : Username with password 선택 > Username : gitlab Username 입력 > Password : gitlab 비밀번호 입력 > 저장한 아이디/비밀번호 데이터 이름 입력 > Create 클릭

2. GitLab Access token 저장

Jenkins 관리 > Manage Credentials > (global) 클릭 > Add Credentials 클릭 > Kind : GitLab API token 선택 > Gitlab으로 이동하여 사용자 설정으로 이동 > Access token으로 이동 > token 생성 > 생성된 token key 복사 > API token : 복사한 key 입력 > 저장한 token 데이터 이름 입력 > Create 클릭

c) GitLab 연결

Jenkins 관리 > 시스템 설정 > Gitlab에 연결한 gitlab 링크 입력 (<https://lab.ssafy.com> 사용) > Credentials : 저장한 token 정보 선택 > Test Connection 으로 연결 테스트 (Success 통과)

d) AWS EC2 연결

Jenkins 관리 > 시스템 설정 > Publish over SSH 설정 > AWS EC2 .pem 파일
내용 등록 > 등록할 AWS EC2 이름 자유롭게 입력 > AWS EC2
Hostname(address) 입력 >> AWS EC2 사용자 이름 입력 (ubuntu 사용) >
Test Connection 으로 연결 테스트 (Success 통과)

- e) 배포할 GitLab에 저장된 프로젝트를 컨테이너에 넣기 및 완료된
프로젝트 폴더 AWS EC2에 전달

Dashboard > 새로운 Item > Freestyle project 생성 > 구성 > GitLab
Connection : 저장한 Gitlab 연결 링크 선택 > 소스 코드 관리 Git 선택 >
Repository URL : 프로젝트 Gitlab url 입력 > Credentials : 저장한
아이디/비밀번호 선택 > Branches to build : 빌드할 브랜치 입력 > 빌드 유발
Build when a ~~~ webhook 체크 선택 및 URL 주소 복사 > 프로젝트
Gitlab으로 이동 > 프로젝트 Gitlab 설정에 webhook 선택 > 복사한 URL
붙여넣기 & 토큰 생성 > 생성된 토큰 key 복사 > 빌드 유발로 돌아와 고급
선택 > Secret token에 생성된 토큰 key 붙여넣기 > Build : Execute shell 선택
> Command에 필요한 패키지 설치 명령어 입력 > 빌드 후 조치에 Send build
artifacts over SSH 선택 > SSH Server에 AWS EC2로 전송할 Source file 지정
> Remote directory에 AWS EC2에 저장될 폴더 이름 지정 > 적용 및 저장 >
지금 빌드 (빌드 완료시 정상 작동)

3. AWS EC2 배포를 위한 Nginx 설정

- 1) AWS EC2에 Nginx 및 letsencrypt 설치

```
sudo apt-get update  
sudo apt-get install nginx  
sudo apt-get install letsencrypt -y
```

```
sudo service nginx stop  
>> 실행중인 nginx 서비스 중지
```

- 2) 인증서 발급받기

```
sudo letsencrypt certonly --standalone -d [도메인 입력]  
알림 받을 이메일 입력 > 동의 후
```

```
/etc/letsencrypt/live/도메인/fullchain.pem, /etc/letsencrypt/live/[도메인]/privkey.pem  
해당 경로가 출력창에 확인되면 성공
```

3) nginx 설정하기

```
sudo vi /etc/nginx/sites-available/default
```

default에 추가될 내용

```
server {  
    listen 443 default_server;  
    listen [::]:443 default_server;  
    ssl on;  
    server_name [도메인];  
  
    ssl_certificate /etc/letsencrypt/live/[도메인]/fullchain.pem;  
    ssl_certificate_key /etc/letsencrypt/live/[도메인]/privkey.pem;  
}
```

4) 80 port > 443 port로 redirect 설정하기

```
sudo vi /etc/nginx/sites-available/default
```

default에 추가될 내용

```
server {  
    listen 80;  
    server_name [도메인];  
    root html;  
  
    location / {  
        return 301 https://[도메인]$request_uri;  
    }  
}
```

5) nginx 서비스 재시작

```
$ sudo service nginx restart
```

```
$ sudo service nginx status
```

```
>> nginx 서비스 상태 체크
```

6) 발급받은 인증서 갱신 및 재발급

```
sudo certbot renew
```

3. Getting Started - Local Environment

1) Front-end

```
cd front_main
```

```
>> frontend 디렉토리로 경로 이동
```

```
npm install
```

```
>> 필요한 node module 설치
```

```
npm run build
```

```
>> # 현재 상태 빌드 시작
```

4. 외부 서비스 - Tmap API

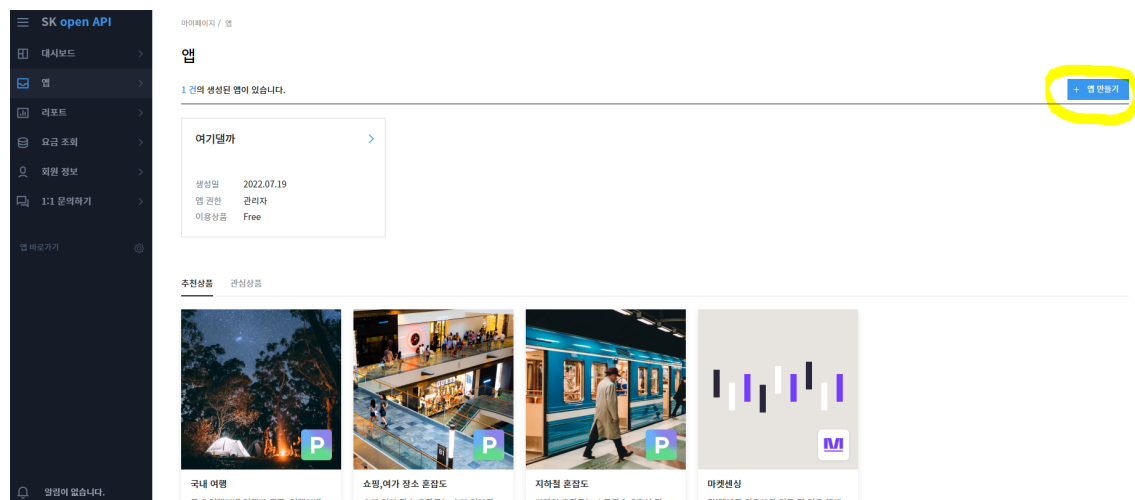
1) Tmap API guide

Tmap API 회원가입 및 앱키 발급받기

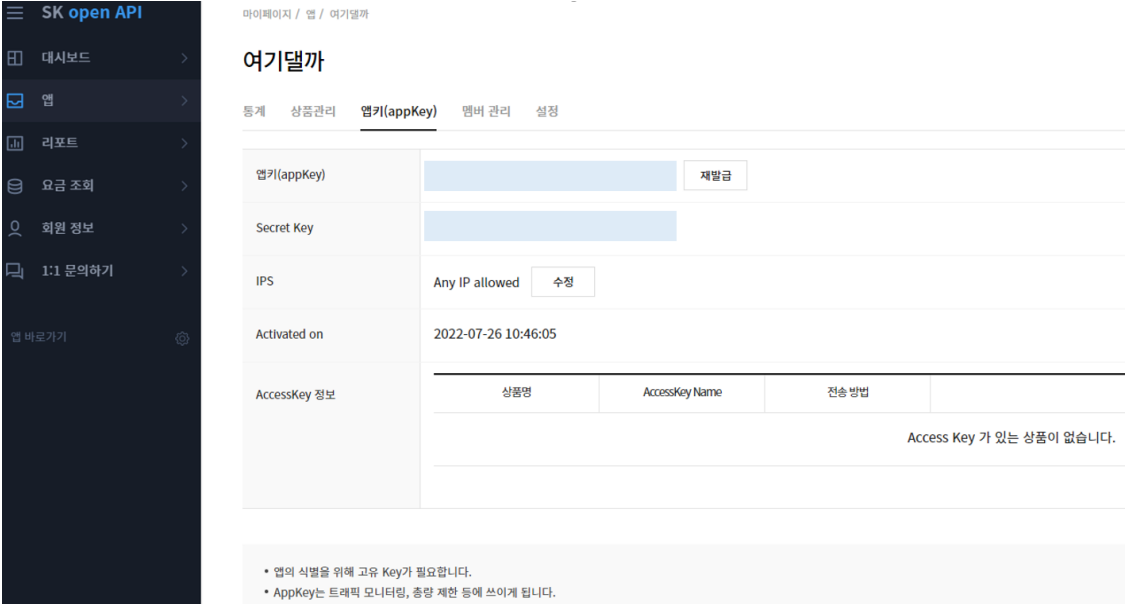
- Tmap API 공식 문서 참조 : <https://tmapapi.sktelecom.com/index.html>

2) Tmap open API

앱 만들기 : <https://openapi.sk.com/>



Tmap API 앱키 발급받기



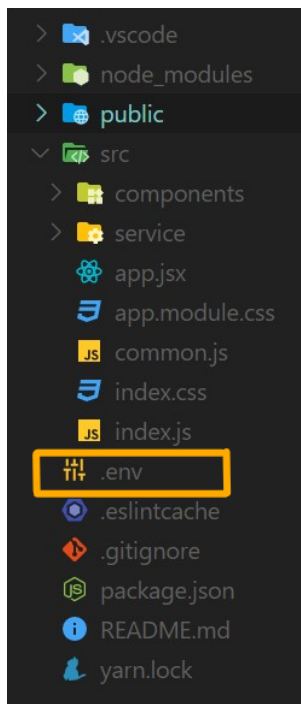
The screenshot shows the SK open API console interface. On the left is a dark sidebar with a menu including '대시보드', '앱', '리포트', '요금 조회', '회원 정보', '1:1 문의하기', and '앱 바로가기'. The main area is titled '여기덜까' and has tabs for '통계', '상품관리', '앱키(appKey)', '멤버 관리', and '설정'. The '앱키(appKey)' tab is active, displaying a form with fields for '앱키(appKey)', 'Secret Key', 'IPS' (set to 'Any IP allowed'), and 'Activated on' (2022-07-26 10:46:05). Below these is a table for 'AccessKey 정보' with columns '상품명', 'AccessKey Name', and '전송 방법'. The table is empty, with a message 'Access Key 가 있는 상품이 없습니다.' at the bottom. A footer note states: '• 앱의 식별을 위해 고유 Key가 필요합니다. • AppKey는 트래픽 모니터링, 총량 제한 등에 쓰이게 됩니다.'

3) index.html

head 부분에 앱키 넣기

```
<script src="https://apis.openapi.sk.com/tmap/jsv2?version=1&appKey=자신의 앱키 ">
</script>
```

4) 환경변수 설정



a. .env 파일 생성

(src 폴더보다 상위, .gitignore와 동등한 위치)

b. .env파일 안에 API_KEY를 입력한다.

(‘REACT_APP_’이라는 접두어를 붙여서 작성하고, 마지막에 ;는 쓰지 않는다.)

REACT_APP_TMAP_API_KEY=애플키

c. .gitignore파일 안에 .env를 추가해준다.

d. index.html 애플키를 변경해준다.

```
<script
```

```
src="https://apis.openapi.sk.com/tmap/jsv2?version=1&appKey=%REACT_APP_TM  
AP_API_KEY%" ></script>
```

5. 시연 시나리오

1) Home Page - Main



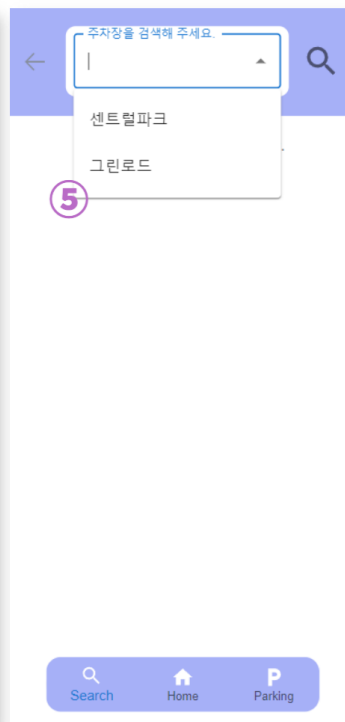
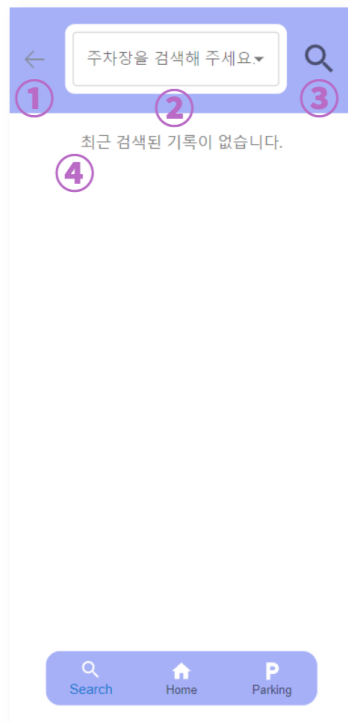
① 확대 / 축소

② 현재 위치

③ 현재 위치 정렬

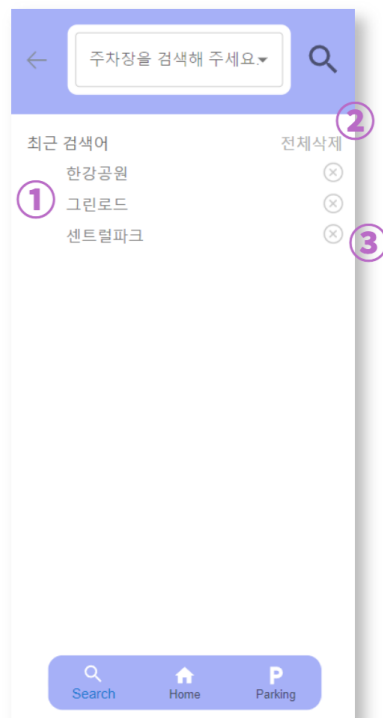
④ 메뉴바

2) Search Page - Search Bar



- ① 뒤로가기
- ② 검색창
- ③ 검색 버튼
- ④ 최근검색기록
- ⑤ 자동완성 드롭다운

3) Search Page - History



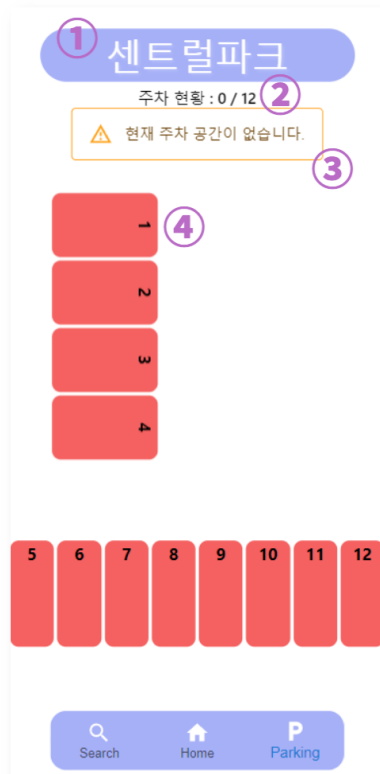
- ① 최근 검색어
- ② 전체 삭제
- ③ 키워드 삭제

4) Home Page - Parking Info



① 주차장 정보창

5) Parking Page - parking Status



① 주차장 이름

② 주차 현황

③ 만차 알람

④ 주차칸