

學 士 學 位 論 文

전기차 충전소 경로 및 대기시간 기반
추천 및 예약시스템

충남대학교

공과대학 컴퓨터공학과

정 민 우

홍 석 우

정 준 희

지도교수 최 훈

2021 年 2 月

전기차 충전소 경로 및 대기시간 기반 추천 및 예약시스템

지도교수 최 훈

이 논문을 공학사학위
청구논문으로 제출함

2020 年 12 月

충 남 대 학 교

공과대학 컴퓨터공학과

| | | | |
|-----------|---|---|---|
| 201402421 | 정 | 민 | 우 |
| 201502129 | 홍 | 석 | 우 |
| 201602069 | 정 | 준 | 희 |

목 차

| | |
|---|----|
| I. 서론 | 1 |
| 1.1 연구 배경 및 목표 | 1 |
| 1.2 연구 내용 | 1 |
| 1.3 논문의 구성 | 2 |
| II. 배경 기술 | 3 |
| 2.1 프로그래밍 언어 | 3 |
| 2.1.1 Swift | 3 |
| 2.1.2 Node.js | 3 |
| 2.2 플랫폼 | 4 |
| 2.2.1 Google Cloud Platform | 4 |
| 2.2.2 Naver Cloud Platform | 5 |
| 2.2.3 공공데이터 포털 | 6 |
| III. 전기차 충전소 경로 및 대기시간 기반 추천 및 예약시스템 설계 | 7 |
| 3.1 서비스 요구사항 | 7 |
| 3.2 Class Diagram | 8 |
| 3.3 Sequence Diagram | 21 |
| 3.4 Service Diagram | 25 |
| 3.5 활용한 데이터 | 26 |
| 3.5.1 데이터 설명 | 26 |
| 3.5.2 데이터 가공 방법 | 27 |
| IV. 전기차 충전소 경로 및 대기시간 기반 추천 및 예약시스템 구현 | 28 |
| 4.1 Application | 28 |
| 4.2 Server | 38 |
| V. 전기차 충전소 경로 및 대기시간 기반 추천 및 예약시스템 시험 | 41 |
| 5.1 테스트 시나리오 | 41 |

| | |
|-------------------|----|
| 5.2 테스트 환경 | 41 |
| 5.3 프로젝트 데모 | 42 |
| 5.4 결과 분석 | 50 |
| VI. 결론 | 51 |
| VII. 참고문헌 | 52 |

그림 목차

| | |
|---|----|
| <그림 1> Google Cloud Platform | 4 |
| <그림 2> Naver Cloud Platform | 5 |
| <그림 3> 전기차 충전 포트 | 6 |
| <그림 4> Application & Server Class Diagram | 8 |
| <그림 5> 충전소 표시 | 21 |
| <그림 6> 단일 충전소 검색 | 22 |
| <그림 7> 충전소 예약 | 23 |
| <그림 8> 경로 기반 충전소 추천 | 24 |
| <그림 9> 전체 서비스 구성도 | 25 |
| <그림 10> 공공데이터 포털 데이터 | 26 |
| <그림 11> 데이터 전처리 코드 | 27 |
| <그림 12> Chargers 구조체 선언 코드 | 28 |
| <그림 13> Charger 구조체 상태변수 | 29 |
| <그림 14> 공공데이터 포털 데이터(JSON 형태) | 30 |
| <그림 15> Chargers 구조체 초기화 코드 | 30 |
| <그림 16> chargerArray 배열 선언 코드 | 31 |
| <그림 17> 지도 실행 시, 초기화 변수 및 카메라 줌 설정 코드 | 31 |
| <그림 18> marker 클릭 시 이벤트 처리 함수 | 32 |
| <그림 19> 충전소 세부정보 알림창 | 32 |
| <그림 20> Url Request 함수 | 33 |
| <그림 21> 충전소의 세부정보를 반환하는 함수 | 34 |
| <그림 22> 예약 정보를 반환하는 함수 | 35 |
| <그림 23> 예약을 요청하는 sendPost 함수 | 36 |
| <그림 24> 경로를 받아 저장하는 DirectionPaths 구조체 코드 | 37 |
| <그림 25> 경로를 요청하는 directionRequest 함수 코드 중 일부 | 37 |
| <그림 26> DataBase에 구현된 evCharger Table | 38 |
| <그림 27> 충전소 세부정보를 반환하는 함수 | 39 |
| <그림 28> 충전기 세부정보를 반환하는 함수 | 39 |

| | |
|------------------------------------|----|
| <그림 29> 경로상 사용 가능한 충전소 정보를 전달하는 함수 | 40 |
| <그림 30> 애플리케이션 초기화면 | 42 |
| <그림 31> 현재 위치 표시 화면 | 43 |
| <그림 32> 충전소 세부정보 화면 | 44 |
| <그림 33> 사용 여부를 확인할 수 있는 화면 | 45 |
| <그림 34> 예약 요청 화면 | 46 |
| <그림 35> 예약 완료 화면 | 46 |
| <그림 36> 출발지, 목적지 설정 화면 | 47 |
| <그림 37> 경로상의 충전소 표시 화면 | 47 |
| <그림 38> 경로상의 충전소 클릭 시 화면 | 48 |
| <그림 39> 경로상의 충전소 예약 요청 화면 | 49 |
| <그림 40> 경로상의 충전소 예약 완료 화면 | 49 |

표 목차

| | |
|--|----|
| <표 1> UserManager Class Description | 9 |
| <표 2> UserInfo Class Description | 10 |
| <표 3> ChargingStation Class Description | 11 |
| <표 4> CafInfo Class Description | 13 |
| <표 5> ReservationInfo Class Description | 14 |
| <표 6> Location Class Description | 15 |
| <표 7> EVCharger Class Description | 16 |
| <표 8> DisplayEVChargingStationMap Class Description | 17 |
| <표 9> Server: RespondClientRequest Class Description | 18 |
| <표 10> Server: ExternalAPIManager Class Description | 18 |
| <표 11> Server: MakeEVChargingStationMap Class Description | 19 |
| <표 12> SearchChargingStation Class Description | 20 |

I. 서 론

1. 연구 배경 및 목표

현재 환경오염, 특히 대기오염 문제가 중요한 이슈로 떠오르고 있다[1]. 이로 인해, 정부에서는 경유차를 제한하고 석탄, 화력발전도 규제하는 노력을 시도하고 있다. 그와 동시에 전기자동차, 수소자동차 등 친환경 에너지를 이용하는 자동차 구매를 장려하고 지원하는 정책들이 실행되고 있다. 2020년 2월 기준, 우리나라의 친환경 자동차는 60만대를 돌파했으며 그 상승 폭은 점점 커지고 있다[2]. 이와 동시에 전기자동차 충전소 설립도 증가하고 있으며, 충전소의 증가와 함께 충전소를 쉽게 이용할 수 있게 하는 서비스가 요구되고 있다. 전기자동차를 이용해서 환경오염 및 연료비 절감으로 인한 금전적인 비용은 줄어들지만, 충전 시간이 오래 걸리기 때문에 시간적인 비용은 증가하게 된 것이 현실이다. 기존 휘발유차는 연료를 채우는데 1분 이내의 시간이 소요되지만, 전기자동차는 급속충전이라고 해도 15~30분의 시간이 소요되고, 완속 충전의 경우 4~5시간이 소요된다[3]. 전기차 충전소의 개수도 주유소 숫자보다 적기 때문에 충전소로의 이동 시간도 더 오래 걸리는 현실이다. 따라서 가까운 충전소를 빠르게 찾고, 장거리 주행을 할 시에는 가는 경로에 있으며 대기시간을 최소화할 수 있는 충전소를 알고 있다면 소모되는 시간을 줄일 수 있을 것으로 예상된다.

본 연구의 목표는 전기자동차 사용자가 자신의 위치, 충전 포트, 대기시간이 가장 짧은 충전소를 이용하도록 도와주고 전기자동차 운전자의 시간을 절감시켜주는 것이다. 본 연구는 전기차 사용자의 시간 절약을 위하여 이동 경로상에 있는 전기차 충전소의 위치와 상태를 알려주고 적절한 충전소를 추천하며, 사용자가 선택한 충전소를 예약하는 시스템을 설계 및 구현하였다.

2. 연구 내용

본 연구에서 구현하는 전기차 충전소 경로 및 대기시간 기반 추천 및 예약시스템은 크게 사용자인 End User, iOS 환경에서 동작하는 애플리케이션, Linux를 기반으로 한 Cloud Server로 구성되어있다. 사용자는 아이폰 애플리케이션의 지도를 통해 충전소의 활성화 여부를 알 수 있으며, 활성화된 충전소는 시간대별로 예약이 가능하다. 현재 위치에서 목적

지를 입력하면 경로상에서 활성화된, 예약 가능한 충전소를 보여주고 사용자는 원하는 충전소를 클릭해 예약 기능을 진행할 수 있도록 한다.

3. 논문의 구성

본 논문의 서론에서는 연구 배경 및 목표, 연구 내용에 대해서 다룬다. 2장에서는 본 연구의 배경 기술이 되는 프로그래밍 언어, Google, Naver 플랫폼에 대해 설명한다. 제3장에서는 전기차 충전소 경로 및 대기시간 기반 추천 및 예약시스템의 설계, 구현, 프로토타입을 통한 테스트에 대해 다루며 본 연구에서 활용한 데이터에 대해 설명한다. 제6장에서는 본 연구의 결론과 향후 기대효과, 추후 연구 제언에 대해 다룬다.

II. 배경 기술

1. 프로그래밍 언어

1) Swift

Swift[4]는 애플의 iOS와 macOS를 위해서 Apple사에서 개발한 프로그래밍 언어이다. 2014년 6월 2일 처음 소개되었다. 기존의 애플 운영체제용 언어였던 Object-C와 공존하기 위한 목적으로 만들어졌기 때문에 LLVM으로 빌드되고, 같은 런타임을 공유하게 된다. 클로저, 다중 리턴타입, 네임스페이스, 제네릭스, 타입유추 등 현대 프로그래밍 언어의 기능이 많이 포함되어있다. Swift의 특징은 크게 3가지로 이야기할 수 있다.

안전성(safe), 신속성(fast), 더 나은 표현성(expressive)이다. guard구문, 변수에 개체가 없는 텅 빈 상태를 금지하는 등 사용자의 문법적 실수를 줄이는 구문들이 존재하여 안전한 프로그래밍을 구현한다. 또한 Swift는 C언어를 기반으로 한 프로그래밍 언어를 대체하려는 목적으로 만들어졌기 때문에 성능 예측이 가능하고, 실행 속도의 최적화와 컴파일러의 지속적인 개량을 통해 더 빠른 컴파일 성능을 구현할 수 있다. 그리고 Swift는 더욱 사용하기 편하고 보기 좋은 문법을 구사한다. C, C++, Java와 다르게 함수형 프로그래밍과 패러다임과 프로토클 지향 프로그래밍 패러다임을 보여준다.

2) Node.js

Node.js는 크롬 자바스크립트 엔진으로 빌드된 자바스크립트 런타임으로 싱글 쓰레드 기반의 비동기 프로그래밍 언어이다. 일반적으로 하나의 쓰레드가 요청을 받아 처리하고, 다른 요청에 대한 작업을 수행한다. I/O 요청이 끝나면 이벤트를 받아서 처리하는 이벤트 방식을 사용한다. 이로 인해서, CPU가 I/O 응답을 기다리는 시간이 필요 없고, 대부분의 연산 작업에 사용되기 때문에 높은 효율성을 가지며, 특히 하나의 Thread로 여러 개의 요청을 처리하는 구조로 되어 있으므로 사용자의 요청에 관한 문제를 처리할 수 있는데 최적화되어 있으므로 프로젝트에 Node.js를 활용하였다.

2. 플랫폼

1) Google Cloud Platform



<그림 1> Google Cloud Platform

구글 클라우드 플랫폼(Google Cloud Platform)[5]은 구글 검색과 유튜브 같은 최종 End-User 제품을 위해 내부적으로 구글이 사용하고 있는 동일한 지원 인프라 구조 위에서 호스팅을 제공하는 구글의 클라우드 컴퓨팅 서비스이다. 클라우드 플랫폼 제공자들은 단순 웹사이트에서 복잡한 애플리케이션에 이르는 일련의 프로그램들을 빌드하기 위한 개발자 제품들을 제공한다. 제공하는 서비스는 Google App Engine, Google Compute Engine, Google Cloud Datastore, Google Cloud Storage, Google BigQuery, Google Cloud Functions, Google Cloud SQL 등이 있다. 다양한 OS(Ubuntu, CentOS, Red Hat, Windows 등)을 제공하고 우분투의 여러 가지 버전이나 다른 배포판을 제공한다. 본 연구에서는 Google Cloud Server에서 Ubuntu 18.04버전으로 호스팅 받아서 백엔드 서버로 사용했다. 구글 클라우드의 장점은 컴퓨팅 파워 규모 조정이 가능하며 보안이 강력하다는 것과 강력한 빅데이터와 머신러닝 관련 클라우드를 제공하고, 고급 소프트웨어 정의 네트워크를 활용해서 일관되고 확장 가능한 결과를 제공한다는 점이 있다.

2) Naver Cloud Platform



〈그림 2〉 Naver Cloud Platform

출발지에서 도착지로 가는 경로를 찾는 작업은 그 자체로 하나의 연구 주제이다. 본 연구에서는 이미 완성되어있는 외부 경로 검색 API를 활용하여 전기차 충전소를 추천해주는 시스템을 다룬다. 외부에서 제공하는 경로 검색 API는 여러 가지 후보가 있었지만, 사용의 편의성을 고려해서 네이버 클라우드 플랫폼에서 제공하고 있는 Directions 5 API[6]를 선택했다. Directions 5 API는 HTTPS 프로토콜[7]을 이용하는 API이다. 해당 API를 이용하기 위해서는 출발지와 도착지를 필수 파라미터로 입력해주어야 하며, 그 밖에 경유지, 탐색 옵션, 차종, 유종, 연비 등을 선택 파라미터로 입력할 수 있다. 본 연구에서는 필요에 따라 필수 파라미터인 출발지와 도착지만을 입력 파라미터로 사용했다.

입력 파라미터인 출발지와 도착지의 형태는 해당 API에서 정의하고 있는 'Request Position Format'을 따라야 하는데 그중에서 출발지와 도착지는 각각 길이가 2인 1차원 배열로, 경도, 위도 순서로 위치 좌표 정보를 나타내야 한다.

해당 API를 사용하기 위해서는 네이버 클라우드 플랫폼에 가입하여 애플리케이션 등록 후 키를 취득해야만 사용할 수 있다.

3) 공공데이터 포털

| 전기차 급속충전 포트 | | | |
|-------------|---|---|--|
| | 차데모 | DC콤보 | AC3상 |
| |  |  |  |
| 개발시기 | 2010년 | 2011년 | 2012년 |
| 개발주체 | 도요전력 | GM 등 미국, 독일 7개 기업 | 르노자동차 |
| 특징 | 완속과 급속충전 간 전파 간섭 우려가 적음 | 완속, 급속 충전포트 하나로 효율적 | 인프라 구축비용 적음 |

<그림 3> 전기차 충전 포트

국내 전기차 충전소의 정보는 '전기자동차 보급 활성화 및 전기자동차 사용자를 위한 전국 공공 및 민간 충전기 운영현황 실시간 제공'이라는 명목으로 한국환경공단에서 OpenAPI를 제공하고 있다. 해당 API에 대한 정보는 공공데이터 포털[8]을 통해 접근할 수 있다. API에서는 두 종류의 상세기능을 제공하고 있다. 첫 번째로 충전기의 전체적인 정보를 전달해주는 기능이다. 해당 정보는 핵심적으로 각 충전소를 구분하는 ID와 주소, 위도, 경도 등의 위치 정보를 포함하며, 각 충전소에는 여러 충전기가 존재할 수 있으므로 소속 충전기를 구분하기 위한 별도의 충전기 ID와 충전기의 상태 및 충전기 타입 등과 그 밖에 여러 부가 정보를 포함한다.

해당 API에서 제공하는 두 번째 기능은 충전기의 상태 정보만을 전달해주는 기능이다. 이 상태 정보 조회 기능을 이용하면 충전기 상태를 나타내기 위한 최소 필수 정보인 충전소 ID, 충전기 ID, 충전기 상태, 상태갱신일시를 획득할 수 있다.

상술한 정보들은 모두 필요하지만, 그중에서도 충전기 타입은 특히 중요한데, 특정 전기자동차는 호환되는 충전 포트만 사용할 수 있기 때문이다. 급속충전에 사용되는 상용 포트는 크게 세 종류가 있다. 본 연구에서는 이처럼 필수적인 정보를 사용자에게 보여줌으로써 사용자가 적절한 충전소를 이용할 수 있도록 편의성을 높였다.

Ⅲ. 전기차 충전소 경로 및 대기시간 기반 추천 및 예약시스템 설계

본 장에서는 전기차 충전소 경로 및 대기시간 기반 추천 및 예약시스템의 설계에 대해 서술한다. 해당 시스템은 크게 Application, Server로 구성되어있다. 각 구성 요소에 대해 서비스 요구사항, 클래스 다이어그램, 각 클래스에 대한 명세, 시퀀스 다이어그램, 서비스 다이어그램, 그리고 활용한 데이터에 대해 기술한다.

1. 서비스 요구사항

본 절에서는 이 연구에서 목표로 하는 전기차 충전소 경로 및 대기시간 기반 추천 및 예약시스템의 설계를 위한 기본적인 서비스 요구사항 분석을 기술한다.

요구사항 1. 충전소 표시

사용자는 전기차 충전소를 표시한 지도를 애플리케이션 실행 화면에서 볼 수 있으며, 지도상에서 전기차 충전소를 선택하여 해당 충전소의 정보 및 소속 충전기들의 상태 정보를 확인할 수 있다.

요구사항 2. 단일 충전소 검색

사용자는 주소만 입력하거나 주소와 충전 포트를 함께 입력하여 해당하는 충전소를 검색할 수 있다.

요구사항 3. 충전소 예약

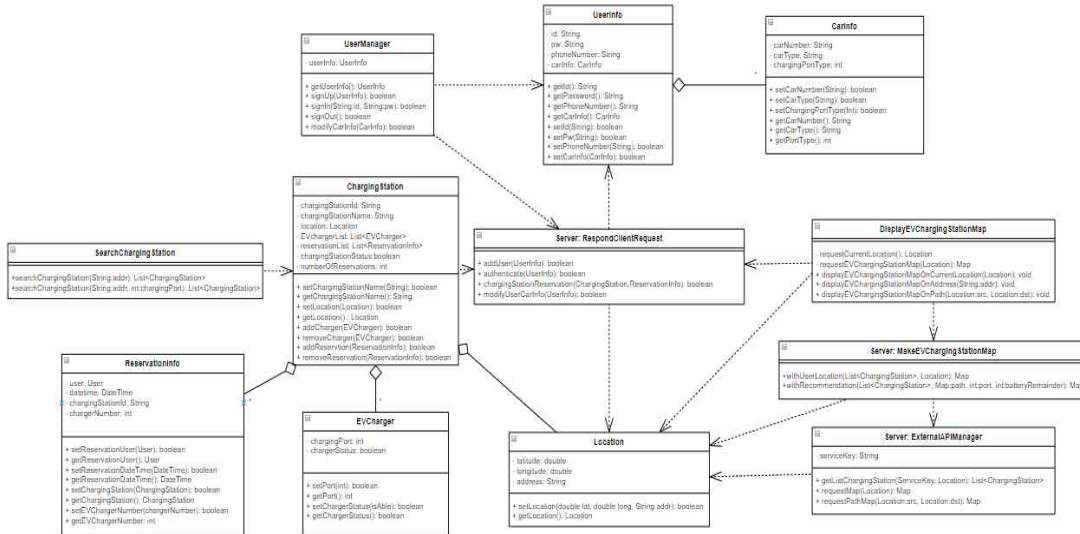
사용자는 지도상에서 전기차 충전소를 선택한 상태에서 특정 충전기를 선택하여 원하는 시간대에 충전기 사용을 예약할 수 있다.

요구사항 4. 경로 기반 충전소 추천

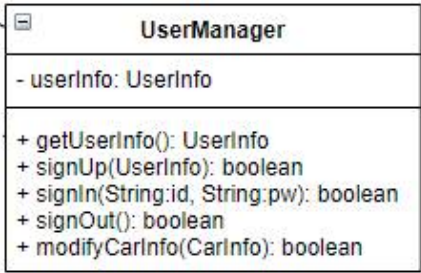
사용자가 출발지와 도착지를 입력하면 입력에 대한 경로를 기반으로 하여 경로와 함께 경로 인근의 충전소가 표시된 지도를 애플리케이션 실행 화면에서 볼 수 있다.

2. Class Diagram

앞 절에서 제시한 서비스 요구사항을 제공하도록 기능을 설계하였다. 설계 모델은 설계 단계의 클래스 다이어그램과 각 클래스에 대한 명세를 포함한다.



<그림 4> Application & Server Class Diagram

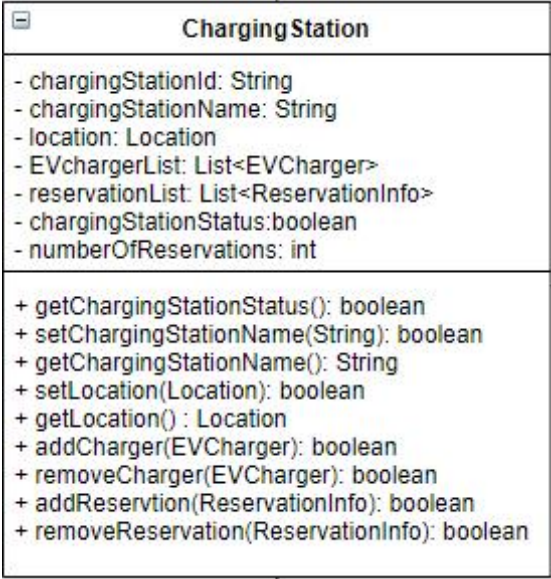
| | | | | |
|----------------|--|---|----------------|----------------|
| Class Name | UserManager | | | |
| Class Diagram |  <pre> classDiagram class UserManager { -userInfo: UserInfo +getUserInfo(): UserInfo +signUp(UserInfo): boolean +signIn(String: id, String: pw): boolean +signOut(): boolean +modifyCarInfo(CarInfo): boolean } </pre> | | | |
| Responsibility | 회원 동작을 다루는 클래스이다. 사용자-회원의 정보 UserInfo를 통해 동작을 수행한다. | | | |
| Attribute | Type | Name | | Description |
| | UserInfo | userInfo | | 회원에 대한 정보 |
| Operation | Return Type | Method Name | Parameter Type | Parameter Name |
| | UserInfo | getUserInfo | | |
| | Description | Information 값을 반환해주는 함수이다. 반환 내용에는 나이, 이름, 전화번호, 차량 정보가 있다. | | |
| | | | | |
| | Return Type | Method Name | Parameter Type | Parameter Name |
| | boolean | signIn | | |
| | Description | 로그인을 하기 위한 함수이다. | | |
| | | | | |
| | Return Type | Method Name | Parameter Type | Parameter Name |
| | boolean | signUp | UserInfo | userInfo |
| | Description | 회원가입을 하기 위한 함수이다. UserInfo를 통해 회원가입을 수행한다. | | |
| | | | | |
| | Return Type | Method Name | Parameter Type | Parameter Name |
| | boolean | signOut | | |
| | Description | 로그아웃을 하기 위한 함수이다. | | |

〈표 1〉 UserManager Class Description

| | | | |
|----------------|---|----------------|----------------|
| Class Name | UserInfo | | |
| Class Diagram | <pre> classDiagram class UserInfo { -id: String -pw: String -phoneNumber: String -carInfo: CarInfo +getId(): String +getPassword(): String +getPhoneNumber(): String +getCarInfo(): CarInfo +setId(String): boolean +setPw(String): boolean +setPhoneNumber(String): boolean +setCarInfo(CarInfo): boolean } </pre> | | |
| Responsibility | 사용자 정보를 나타내는 클래스이다. | | |
| Attribute | Type | Name | Description |
| | String | phoneNumber | 전화번호 |
| | Carinfo | carInfo | 차량정보 |
| | String | id | 아이디 |
| Operation | String | passwd | 비밀번호 |
| | Return Type | Method Name | Parameter Type |
| | | getPhoneNumber | String |
| | Description | 전화번호를 반환한다. | |
| | Return Type | Method Name | Parameter Type |
| | CarInfo | getId | |
| | Description | 회원 아이디를 반환한다. | |
| | Return Type | Method Name | Parameter Type |
| | CarInfo | getId | |
| | Description | 회원 아이디를 반환한다. | |
| | Return Type | Method Name | Parameter Type |
| | CarInfo | getPw | |
| | Description | 회원 비밀번호를 반환한다. | |
| | Return Type | Method Name | Parameter Type |
| | boolean | setPhoneNumber | String |
| | Description | 전화번호를 설정한다. | |
| | Return Type | Method Name | Parameter Type |
| | boolean | setCarInfo | CarInfo |
| | Description | 차량 정보를 설정한다. | |

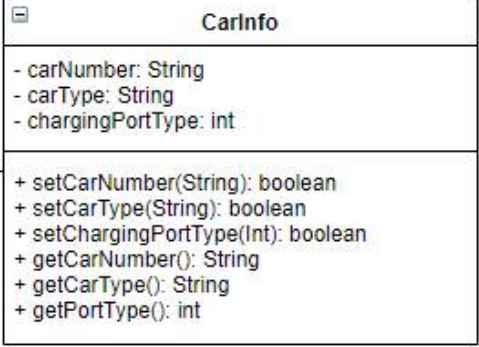
| | Return Type | Method Name | Parameter Type | Parameter Name |
|--|-------------|----------------|----------------|----------------|
| | boolean | setId | String | |
| | Description | 회원 아이디를 설정한다. | | |
| | | | | |
| | Return Type | Method Name | Parameter Type | Parameter Name |
| | boolean | setPw | | |
| | Description | 회원 비밀번호를 설정한다. | | |

<표 2> UserInfo Class Description

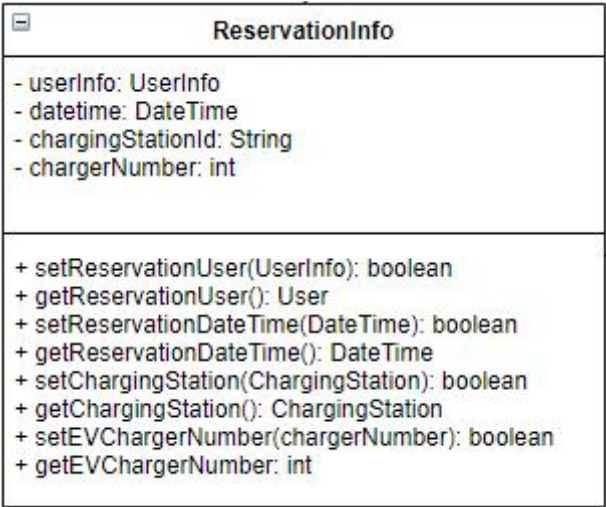
| | | | |
|----------------|---|-----------------------|----------------------------|
| Class Name | ChargingStation | | |
| Class Diagram |  | | |
| Responsibility | 하나의 충전소 클래스이다. 충전기 리스트, 충전소 위치 등의 정보를 담고 있고 충전기 추가, 삭제, 예약 등의 기능을 담당한다. | | |
| Attribute | Type | Name | Description |
| | String | chargingStationID | 충전소 고유번호 |
| | String | chargingStationName | 충전소 이름 |
| | Location | Location | 충전소 위치 객체 |
| | List<EVcharger> | EVchargerList | 충전기 리스트 |
| | List<Reservation> | reservationList | 예약 리스트 |
| | boolean | chargingStationStatus | 현재 사용 가능한 충전기 존재 여부를 나타낸다. |
| | int | numberOfReservations | 충전소에 존재하는 충전기 개수를 나타낸다. |

| | | | |
|-----------|-------------|------------------------|-----------------|
| Operation | Return Type | Method Name | Parameter Type |
| | boolean | setChargingStationName | String |
| | Description | 충전소 이름 정보를 설정하는 함수이다. | |
| | Return Type | Method Name | Parameter Type |
| | String | getChargingStationName | |
| | Description | 충전소 이름을 반환하는 함수이다. | |
| | Return Type | Method Name | Parameter Type |
| | boolean | setLocation | Location |
| | Description | 충전소 위치 정보를 설정하는 함수이다. | |
| | Return Type | Method Name | Parameter Type |
| | Location | getLocation | |
| | Description | 충전소 위치 정보를 반환하는 함수이다. | |
| | Return Type | Method Name | Parameter Type |
| | boolean | addCharger | EVCharger |
| | Description | 충전소에 충전기 객체를 추가한다. | |
| | Return Type | Method Name | Parameter Type |
| | boolean | removeCharger | EVCharger |
| | Description | 충전소에서 충전기 객체를 삭제한다. | |
| | Return Type | Method Name | Parameter Type |
| | boolean | addReservation | ReservationInfo |
| | Description | 충전소에 예약 객체를 추가한다. | |
| | Return Type | Method Name | Parameter Type |
| | boolean | removeReservation | ReservationInfo |
| | Description | 충전소에서 예약 객체를 삭제한다. | |

<표 3> ChargingStation Class Description

| | | | | |
|----------------|--|---------------------|----------------|----------------|
| Class Name | CarInfo | | | |
| Class Diagram |  <pre> classDiagram class CarInfo { -carNumber: String -carType: String -chargingPortType: int +setCarNumber(String): boolean +setCarType(String): boolean +setChargingPortType(int): boolean +getCarNumber(): String +getCarType(): String +getPortType(): int } </pre> | | | |
| Responsibility | 차량에 관한 정보를 담고있는 클래스이다. | | | |
| Attribute | Type | Name | | Description |
| | String | carNumber | | 차량번호 |
| | String | carType | | 차량종류 |
| | int | chargingPortType | | 차량 충전 타입 |
| Operation | Return Type | Method Name | Parameter Type | Parameter Name |
| | boolean | setCarNumber | String | |
| | Description | 차량 번호를 설정한다. | | |
| | Return Type | Method Name | Parameter Type | Parameter Name |
| | String | getCarNumber | | |
| | Description | 차량 번호를 반환한다. | | |
| | Return Type | Method Name | Parameter Type | Parameter Name |
| | boolean | setCarType | String | |
| | Description | 차량 종류를 설정한다. | | |
| | Return Type | Method Name | Parameter Type | Parameter Name |
| | String | getCarType | | |
| | Description | 차량 종류를 반환한다. | | |
| | Return Type | Method Name | Parameter Type | Parameter Name |
| | boolean | setChargingPortType | int | |
| | Description | 차량 충전 포트 타입을 설정한다. | | |
| | Return Type | Method Name | Parameter Type | Parameter Name |
| | int | getChargingPortType | | |
| | Description | 차량 충전 포트 타입을 반환한다. | | |

<표 4> CarInfo Class Description

| | | | |
|----------------|---|----------------------------|------------------|
| Class Name | ReservationInfo | | |
| Class Diagram |  <pre> classDiagram class ReservationInfo { -userInfo: UserInfo -datetime: DateTime -chargingStationId: String -chargerNumber: int +setReservationUser(UserInfo): boolean +getReservationUser(): User +setReservationDateTime(DateTime): boolean +getReservationDateTime(): DateTime +setChargingStation(ChargingStation): boolean +getChargingStation(): ChargingStation +setEVChargerNumber(chargerNumber): boolean +getEVChargerNumber: int } </pre> | | |
| Responsibility | 예약에 관한 정보를 나타내는 클래스이다. | | |
| Attribute | Type | Name | Description |
| | User | user | 예약자 정보 |
| | DateTime | datetime | 예약할 시간 정보 |
| | String | chargingStationID | 예약할 충전소 고유번호 |
| Operation | int | chargerNumber | 어떤 충전기를 예약할지의 정보 |
| | Return Type | Method Name | Parameter Type |
| | boolean | setReservationUser | User |
| | Description | 예약하기 원하는 사용자를 지정하는 함수이다. | |
| | Return Type | Method Name | Parameter Type |
| | User | getReservationUser | |
| | Description | 예약 사용자에게 관한 정보를 반환하는 함수이다. | |
| | Return Type | Method Name | Parameter Type |
| | boolean | setReservationDateTime | DateTime |
| | Description | 예약 날짜 및 시간을 지정하는 함수이다. | |
| | Return Type | Method Name | Parameter Type |
| | DateTime | getReservationDateTime | |
| | Description | 예약 날짜 및 시간을 반환하는 함수이다. | |
| | Return Type | Method Name | Parameter Type |
| | boolean | setChargingStation | ChargingStation |
| | Description | 충전소를 예약정보에 저장하는 함수이다. | |

| | | | |
|--|-----------------|----------------------|----------------|
| | | | |
| | Return Type | Method Name | Parameter Type |
| | ChargingStation | getChargingStation | |
| | Description | 예약하는 충전소를 반환하는 함수이다. | |
| | | | |
| | Return Type | Method Name | Parameter Type |
| | boolean | setEVChargerNumber | chargerNumber |
| | Description | 예약하는 충전기 정보를 저장한다. | |
| | | | |
| | Return Type | Method Name | Parameter Type |
| | EVCharger | getEVChargerNumber | |
| | Description | 예약하는 충전기 정보를 반환한다. | |

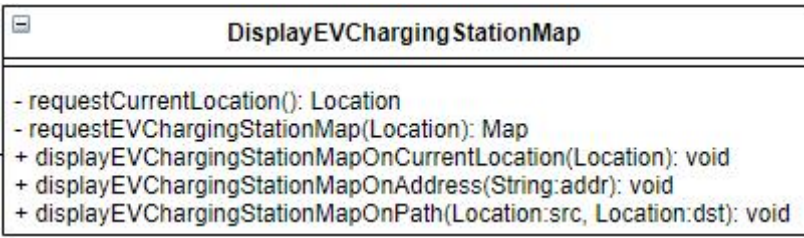
<표 5> ReservationInfo Class Description

| | | | | |
|----------------|---|-----------------------|------------------------|-----------------|
| Class Name | Location | | | |
| Class Diagram | <div><div><div><div></div><div>Location</div></div><div><div>- latitude: double</div><div>- longitude: double</div><div>- address: String</div></div><div><div>+ setLocation(double lat, double long, String addr): boolean</div><div>+ getLocation(): Location</div></div></div></div> | | | |
| Responsibility | 위치 정보에 관한 내용을 나타내는 클래스이다. | | | |
| Attribute | Type | Name | Description | |
| | double | latitude | 위도 정보 | |
| | double | longitude | 경도 정보 | |
| | String | Address | 주소 정보 | |
| Operation | Return Type | Method Name | Parameter Type | Parameter Name |
| | boolean | setLocation | double, double, String | lat, long, adds |
| | Description | 충전소 위치 정보를 저장하는 함수이다. | | |
| | Return Type | Method Name | Parameter Type | Parameter Name |
| | Loaction | getLocation | | |
| | Description | 충전소 위치 정보를 반환하는 함수이다. | | |

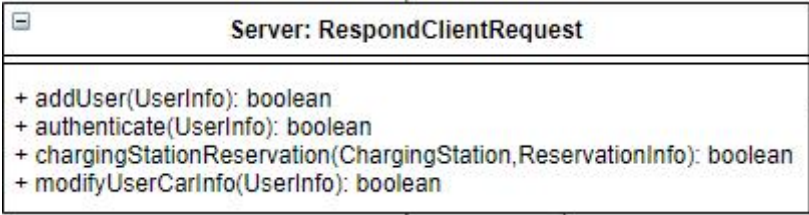
<표 6> Location Class Description

| | | | | |
|----------------|---|---------------------------|------------------|----------------|
| Class Name | EVCharger | | | |
| Class Diagram | <div><div><div><div><div></div><div>EVCharger</div></div><div><div>- chargingPort: int</div><div>- chargerStatus: boolean</div></div><div><div>+ setPort(int): boolean</div><div>+ getPort(): int</div><div>+ setChargerStatus(isAble): boolean</div><div>+ getChargerStatus(): boolean</div></div></div></div></div> | | | |
| Responsibility | 충전소의 정보에 관한 내용을 나타내는 클래스이다. | | | |
| Attribute | Type | Name | Description | |
| | int | chargingPort | 충전 포트 타입 정보를 저장 | |
| | boolean | chargerStatus | 충전기 이용 가능 여부를 저장 | |
| Operation | Return Type | Method Name | Parameter Type | Parameter Name |
| | boolean | setPort | int | |
| | Description | 충전기의 포트타입 정보를 저장하는 함수이다. | | |
| | Return Type | Method Name | Parameter Type | Parameter Name |
| | int | getPort | | |
| | Description | 충전기의 포트타입 정보를 반환하는 함수이다. | | |
| | Return Type | Method Name | Parameter Type | Parameter Name |
| | boolean | setChargerStatus | boolean | isAble |
| | Description | 충전기의 사용 가능 여부를 저장하는 함수이다. | | |
| | Return Type | Method Name | Parameter Type | Parameter Name |
| | boolean | getChargerStatus | | |
| | Description | 충전기의 사용 가능 여부를 반환하는 함수이다. | | |

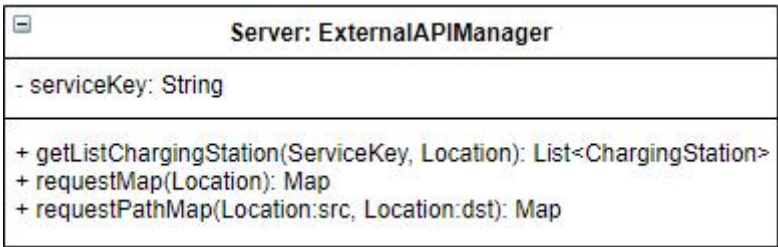
<표 7> EVCharger Class Description

| | | | | |
|----------------|--|--|--------------------|-----------------|
| Class Name | DisplayEVChargingStationMap | | | |
| Class Diagram |  <pre> classDiagram class DisplayEVChargingStationMap { -requestCurrentLocation(): Location -requestEVChargingStationMap(Location): Map +displayEVChargingStationMapOnCurrentLocation(Location): void +displayEVChargingStationMapOnAddress(String:addr): void +displayEVChargingStationMapOnPath(Location:src, Location:dst): void } </pre> | | | |
| Responsibility | 충전소가 표시된 지도를 서버에 요청하여 얻은 결과를 화면에 출력하는 클래스이다. | | | |
| Operation | Return Type | Method Name | Parameter Type | Parameter Name |
| | Location | requestCurrentLocation | | |
| | Description | 사용자에게 현재 위치의 정보 제공을 요구한다. 동의하지 않으면 null, 동의하면 Location을 반환한다. | | |
| | Return Type | Method Name | Parameter Type | Parameter Name |
| | Map | requestEVChargingStationMap | Location | currentLocation |
| | Description | 현재 위치 기반의 충전소가 표시된 지도를 요청하여 얻은 결과를 반환한다. | | |
| | Return Type | Method Name | Parameter Type | Parameter Name |
| | | displayEVChargingStationMapOnCurrentLocation | Location | currentLocation |
| | Description | 현재 위치 기반의 충전소가 표시된 지도를 화면상에 출력한다. | | |
| | Return Type | Method Name | Parameter Type | Parameter Name |
| | | displayEVChargingStationMapOnAddress | String | address |
| | Description | 주소 검색 결과를 기반으로 충전소가 표시된 지도를 화면상에 출력한다. | | |
| | Return Type | Method Name | Parameter Type | Parameter Name |
| | | displayEVChargingStationMapOnPath | Location, Location | src, dst |
| | Description | 출발지와 도착지를 입력받아 경로를 탐색하고, 탐색 경로를 기반으로 하여 추천 충전소가 표시된 지도를 화면상에 출력한다. | | |

<표 8> DisplayEVChargingStationMap Class Description

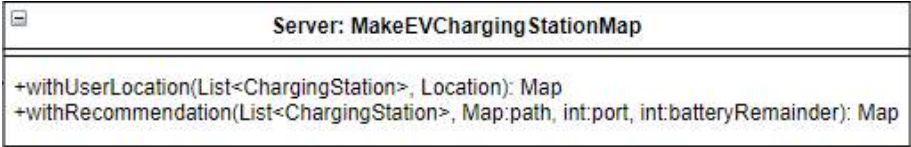
| | | | |
|----------------|---|----------------------------|-------------------------------------|
| Class Name | Server: RespondClientRequest | | |
| Class Diagram |  <pre> classDiagram class "Server: RespondClientRequest" { +addUser(UserInfo): boolean +authenticate(UserInfo): boolean +chargingStationReservation(ChargingStation, ReservationInfo): boolean +modifyUserCarInfo(UserInfo): boolean } </pre> | | |
| Responsibility | 충전소가 표시된 지도를 서버에 요청하여 얻은 결과를 화면에 출력하는 클래스이다. | | |
| Operation | Return Type | Method Name | Parameter Type |
| | boolean | addUser | UserInfo |
| | Description | 회원 정보를 DB에 저장한다. | |
| | Return Type | Method Name | Parameter Type |
| | boolean | authenticate | UserInfo |
| | Description | DB 쿼리를 통해 회원 인증을 수행한다. | |
| | Return Type | Method Name | Parameter Type |
| | boolean | chargingStationReservation | ChargingStation, ReservationInfo |
| | Description | 특정 충전소에 충전 예약을 신청한다. | |
| | Return Type | Method Name | Parameter Type |
| | boolean | modifyUserCarInfo | UserInfo |
| | Description | DB에서 회원의 차량 정보를 수정한다. | |

<표 9> Server: RespondClientRequest Class Description

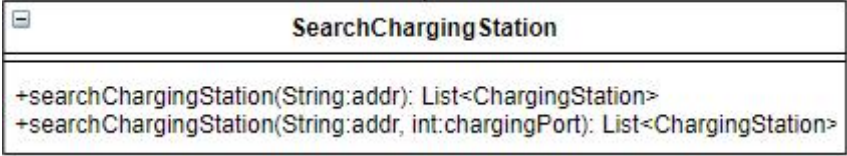
| | | | |
|----------------|--|--|--|
| Class Name | Server: ExternalAPIManager | | |
| Class Diagram |  <pre> classDiagram class "Server: ExternalAPIManager" { -serviceKey: String +getListChargingStation(ServiceKey, Location): List<ChargingStation> +requestMap(Location): Map +requestPathMap(Location:src, Location:dst): Map } </pre> | | |
| Responsibility | 충전소가 표시된 지도를 생성하기 위해 외부 API (공공데이터포털, 구글 지도)에 접근하여 필요한 정보를 가져오는 클래스다. | | |

| Attribute | Type | Name | Description | |
|-----------|-----------------------|---|----------------------|----------------------|
| | String | serviceKey | 외부 API 접근 권한을 위한 Key | |
| Operation | Return Type | Method Name | Parameter Type | Parameter Name |
| | List<ChargingStation> | getListChargingStation | String, Location | serviceKey, location |
| | Description | 위치 정보를 기반으로 하여 공공데이터포털 API에 요청하여 충전소 데이터를 List로 가져온다. | | |
| | Return Type | Method Name | Parameter Type | Parameter Name |
| | Map | requestMap | Location | location |
| | Description | 위치 정보를 기반으로 하여 네이버지도 API에 요청하여 지도를 가져온다. | | |
| | Return Type | Method Name | Parameter Type | Parameter Name |
| | Map | requestPathMap | Location, Location | src, dst |
| | Description | 출발지, 도착지의 정보를 기반으로 하여 네이버지도 API에 요청하여 경로를 표시한 지도를 가져온다. | | |

<표 10> Server: ExternalAPIManager Class Description

| Class Name | Server: MakeEVChargingStationMap | | | |
|----------------|---|---|--------------------------------------|---------------------------------------|
| Class Diagram |  <pre> classDiagram class "Server: MakeEVChargingStationMap" { +withUserLocation(List<ChargingStation>, Location): Map +withRecommendation(List<ChargingStation>, Map:path, int:port, int:batteryRemainder): Map } </pre> | | | |
| Responsibility | 서버에서 클라이언트의 요청에 응답하기 위해 전기차 충전소를 표시한 지도를 생성한다. | | | |
| Operation | Return Type | Method Name | Parameter Type | Parameter Name |
| | Map | withUserLocation | List<ChargingStation>, Location | |
| | Description | 사용자 정보를 기반으로 지도를 생성한다. | | |
| | Return Type | Method Name | Parameter Type | Parameter Name |
| | boolean | withRecommendation | List<ChargingStation>, Map, int, int | list, pathMap, port, batteryRemainder |
| | Description | 경로 및 충전 포트, 배터리 잔량을 기반으로한 추천 충전소를 표시한 지도를 생성한다. | | |

<표 11> Server: MakeEVChargingStationMap Class Description

| | | | | |
|----------------|--|------------------------------|----------------|--------------------------|
| Class Name | SearchChargingStation | | | |
| Class Diagram |  <pre> classDiagram class SearchChargingStation { +searchChargingStation(String:addr): List<ChargingStation> +searchChargingStation(String:addr, int:chargingPort): List<ChargingStation> } </pre> | | | |
| Responsibility | 주소 또는 주소+충전 포트를 기반으로 충전소 목록을 검색한다. | | | |
| Operation | Return Type | Method Name | Parameter Type | Parameter Name |
| | List<ChargingStation> | searchCharging Station | String | address |
| | Description | 주소를 기반으로 충전소 목록을 검색한다. | | |
| | Return Type | Method Name | Parameter Type | Parameter Name |
| | List<ChargingStation> | searchCharging Station | String, int | address, chargingPort |
| | Description | 주소+충전 포트를 기반으로 충전소 목록을 검색한다. | | |

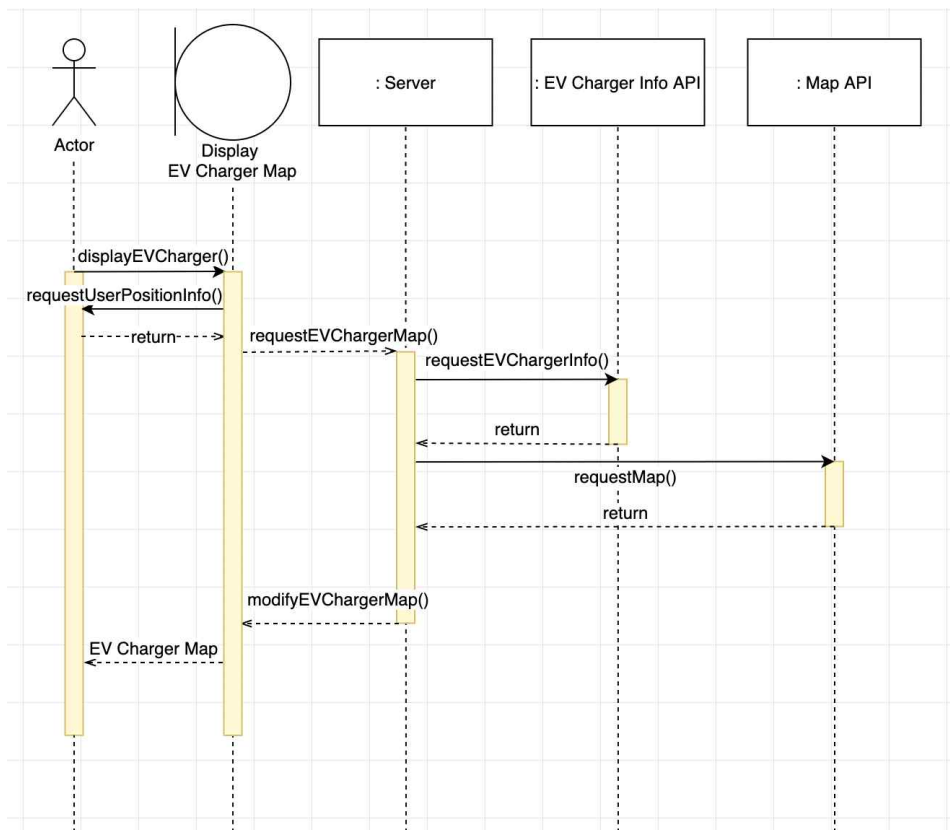
<표 12> SearchChargingStation Class Description

3. Sequence Diagram

본 절에서는 전기차 충전소 경로 및 대기시간 기반 추천 및 예약시스템의 설계 모델(시퀀스 다이어그램)을 기술한다. 앞 절에서 제시한 클래스 다이어그램 및 각 클래스의 명세를 바탕으로 클래스들의 인스턴스인 객체 간의 상호작용을 표현한다.

1) 충전소 표시

사용자의 위치 정보를 기반으로 하여 전기차 충전소 지도를 화면에 출력한다.

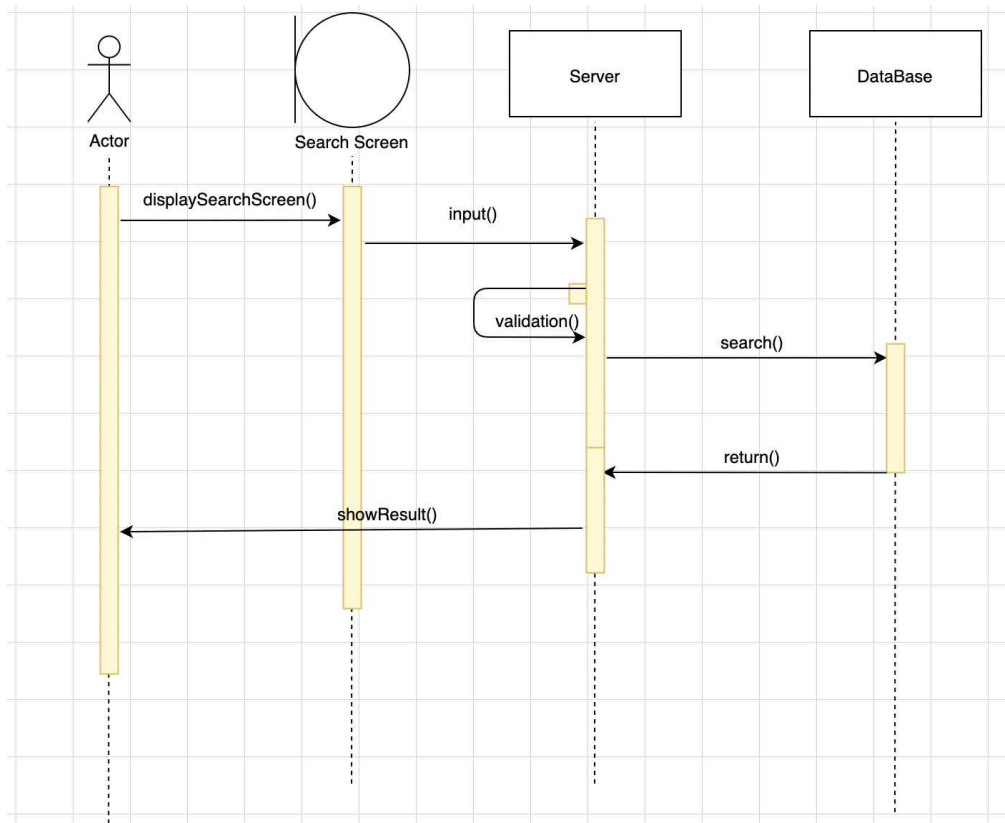


<그림 5> 충전소 표시 시퀀스 다이어그램

1. 사용자는 시스템으로부터 위치 정보 제공 동의를 요구받는다.
2. 사용자가 위치 정보 제공을 동의하면, 사용자의 위치 정보를 기반으로 서버가 전기차 충전소 공공데이터 API와 지도 API에 요청하여 데이터를 받는다.
3. 외부 API의 응답을 바탕으로 전기차 충전소 지도를 사용자의 위치 정보에 맞도록 수정하여 사용자에게 출력한다.

2) 단일 충전소 검색

사용자가 충전소 이름으로 검색하면 서버에서 해당하는 충전소에 대한 정보를 DB에 검색해서 해당하는 정보를 사용자에게 보여준다.

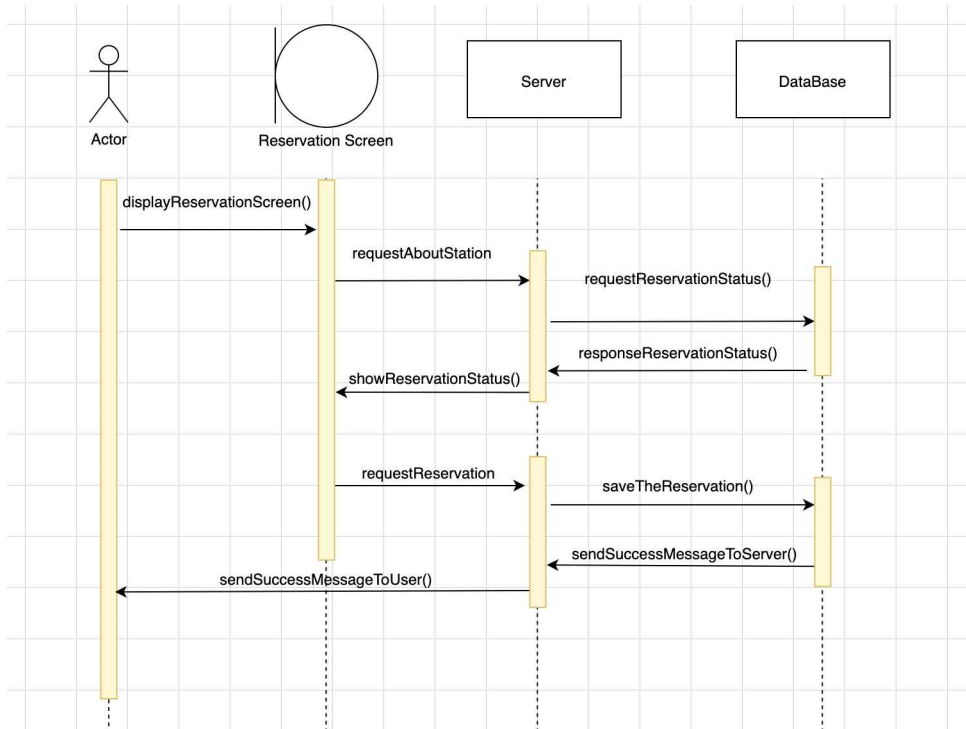


<그림 6> 단일 충전소 검색 시퀀스 다이어그램

1. 사용자가 충전소 검색을 클릭하면 displaySearchScreen()를 통해 충전소 검색 페이지로 이동한다.
2. 사용자는 검색 필터 중 원하는 것을 선택해서 검색어를 입력하면 input 함수가 실행된다.
3. input()이 실행되면 서버에서는 validation()을 통해 검색어의 형식을 검사한다.
4. 검색어 형식이 올바르다면 데이터베이스에 데이터를 요청한다
5. 데이터베이스에서 결과값을 서버로 돌려준다.
6. 서버에서 결과값을 사용자에게 보여준다.

3) 충전소 예약

사용자가 충전소를 예약해서 충전소에 도착했을 경우 대기하지 않고 곧바로 이용할 수 있도록 예약 서비스를 제공한다.

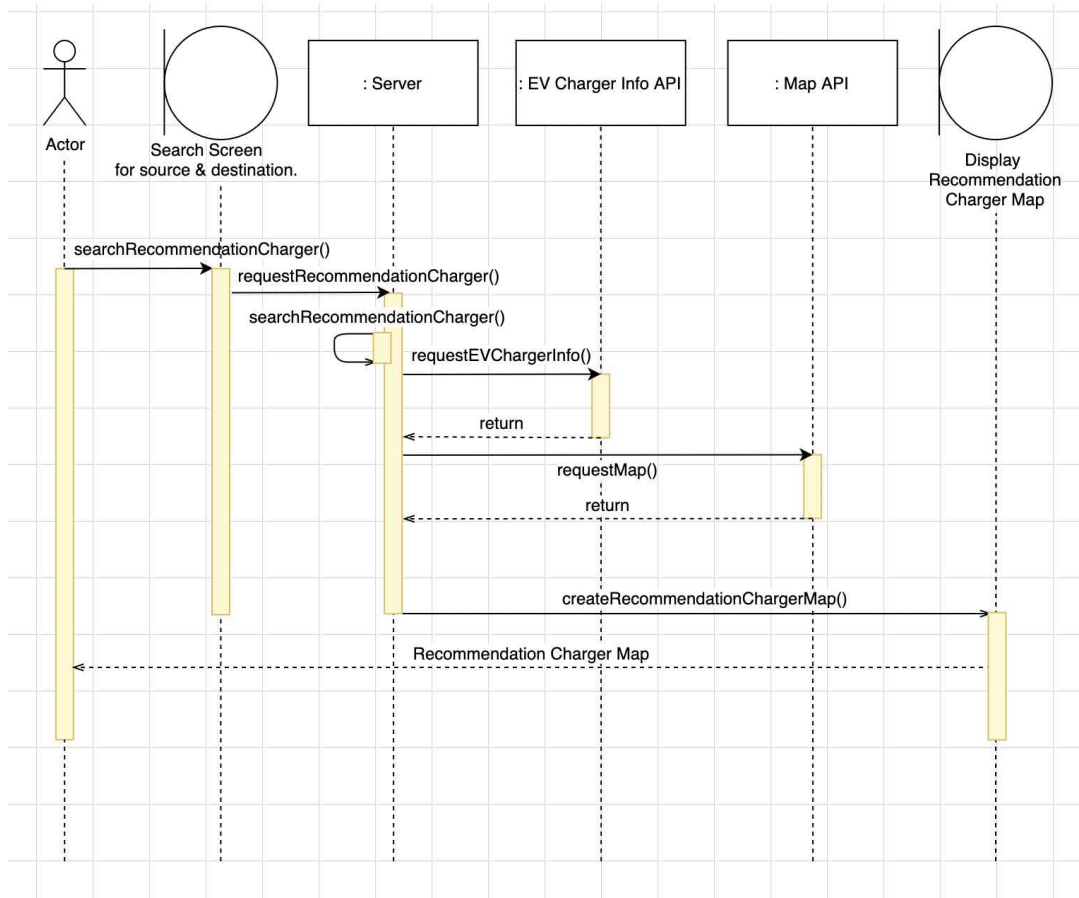


<그림 7> 충전소 예약 시퀀스 다이어그램

1. 사용자는 예약 페이지로 접근한다.
2. 사용자가 어떤 충전소를 선택하면 그 충전소의 예약 현황을 먼저 보여주기 위해 서버에게 요청한다.
3. 서버는 다시 DB에 요청한다.
4. DB는 예약 현황을 다시 서버에게 알려준다.
5. 서버는 다시 사용자에게 예약 현황을 알려준다.
6. 사용자가 예약 실행 버튼을 누르면 서버로 전송된다.
7. 서버는 다시 DB에 요청한다.
8. DB는 저장하고 서버에게 성공했음을 알린다.
9. 서버는 사용자에게 예약이 성공했음을 알린다.

4) 경로 기반 충전소 추천

차량 정보가 서버에 등록되어있는 사용자가 출발지와 도착지를 입력하면 해당 경로상에서 적절한 전기차 충전소를 추천해준다.

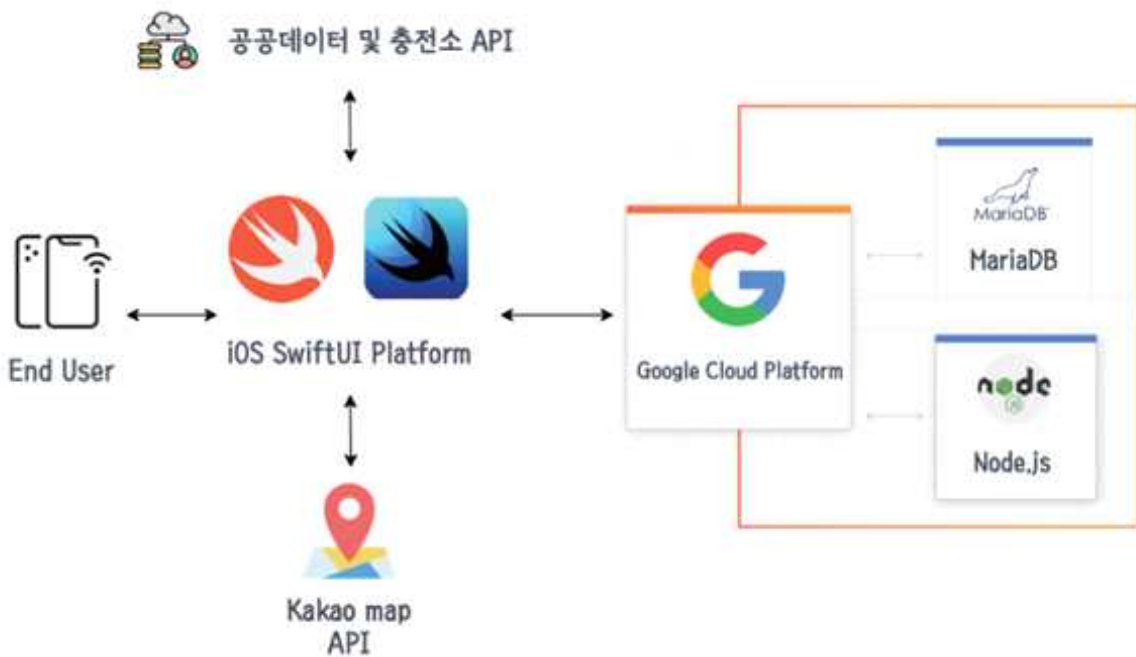


<그림 8> 경로 기반 충전소 추천 시퀀스 다이어그램

1. 사용자가 출발지와 도착지를 화면에 입력한다.
2. 입력된 출발지 도착지가 서버로 전송되어 사용자의 차량 정보와 함께 파라미터로 사용되어 추천 충전소를 찾는다.
3. 추천 충전소 결과물을 바탕으로 전기차 충전소 공공데이터 포털 API와 지도 API에 요청한다.
4. 외부 API의 응답을 받아 추천 전기차 충전소 지도를 생성하여 사용자의 화면에 출력한다.

4. Service Diagram

본 절에서는 전기차 충전소 경로 및 대기시간 기반 추천 및 예약시스템의 전체적인 구조를 보여준다. 앞 절에서 제시한 설계에 따라 iOS 애플리케이션을 중심으로 상호작용이 이루어진다. iOS 애플리케이션은 Swift를 바탕으로 이루어진 SwiftUI Platform에서 작동한다. iOS 애플리케이션으로부터의 요청은 서버 인프라인 Google Cloud Platform 내부 시스템에서 처리한다. Node.js에서 외부 요청 처리 로직이 수행되고, 데이터는 내부적으로 데이터베이스 시스템인 MariaDB를 이용한다. 또한, iOS 애플리케이션은 전기차 충전소의 데이터를 획득하기 위해서 공공데이터 포털 API와 상호작용한다. 그리고 iOS 애플리케이션은 경로 정보를 획득하기 위해서 Naver Directions 5 API와 상호작용한다.



<그림 9> 전체 서비스 구성도

5. 활용한 데이터

본 절에서는 전기차 충전소 경로 및 대기시간 기반 추천 및 예약시스템에서 활용하고 있는 데이터를 설명한다. 앞 절에서는 iOS 애플리케이션을 중심으로 서비스 제공을 위한 다양한 상호작용을 설명하였으므로 이러한 상호작용을 실현하는 데이터가 무엇이고, 어떻게 데이터를 가공하여 처리하는지 기술한다.

1) 데이터 설명

데이터는 공공데이터 포털의 ‘전기자동차 충전소 현황[9]’을 사용하였다. 한국환경공단에서 제공하는 국내 전기자동차 충전소 위치 및 상태 정보 등을 통해 형식에 맞춰 DB에 넣어 데이터를 저장하였다. 충전소 이름, 위치와 같은 정적인 정보는 App 실행 후, 로컬에서 불러와 빠르게 정보를 가져올 수 있다.

충전기 상태 정보는 매분 단위로 API를 호출해 DB에 업데이트하여 실시간 상태 정보(1: 통신 이상, 2: 충전 대기, 3: 충전 중, 4: 운영중지, 5: 점검 중, 9: 상태 미확인)를 참조해서 정의하였다. API 호출 시, XML 형태로 데이터가 전송되며 약 10000개의 충전소 정보를 가공해서 사용하였다.

```
{
  "statNm": "범서읍 제2민원실 공용주차장",
  "statId": "ME195020",
  "chgerId": "01",
  "chgerType": "06",
  "addr": "울산광역시 울주군 범서읍 구영리216-3",
  "lat": "35.571183",
  "lng": "129.244193",
  "useTime": "24시간 이용가능",
  "busiId": "ME",
  "busiNm": "환경부",
  "busiCall": "1661-9408",
  "stat": "2",
  "statUpdDt": "20200912183432",
  "powerType": "급속(50kW)"
}
```

<그림 10> 공공데이터 포털 데이터

2) 데이터 가공 방법

App과의 통신에서 필요한 데이터 형식을 맞추기 위해서 API에서 받아온 정보를 처리하는 간단한 함수를 작성해서 가공하였다.

XML 형식으로 불러온 데이터는 DB 저장과 App과의 정보 전달 효율성을 높이기 위해 JSON[10] 형태로 변환하였다. XML 데이터를 변환하기 위해서 Node.js의 xml-js 라이브러리를 사용하였고, 처음 App을 실행할 때, 매분 충전기 상태를 업데이트할 때 해당 함수를 이용한다.

```
function (cb) {
  request({
    url: url + queryParams,
    method: 'GET'
  }, function (error, response, body) {
    if (error) {
      console.log(`err=> ${err}`);
    } else {
      if (response.statusCode == 200) {
        var result = body
        var xmlToJson = convert.xml2json(result, {
          compact: true,
          spaces: 4
        });
        const parsedData = JSON.parse(xmlToJson);
        df = JSON.parse(xmlToJson).response.body.items.item;
        df_length = df.length;
        var item_length = parsedData.response.body.items.item.length;
        var item = parsedData.response.body.items.item;

        console.log('Complete: Convert XML to JSON');
        cb(null);
      }
    }
  });
},
```

<그림 11> 데이터 전처리 코드

Ⅳ 전기차 충전소 경로 및 대기시간 기반 추천 및 예약시스템 구현

본 장에서 구현하는 전기차 충전소 경로 및 대기시간 기반 추천 및 예약시스템 구현에 대해 서술한다. 앞 장에서 기술한 설계를 바탕으로, Application과 Server 구현에서의 핵심 사항을 작성한 코드를 보여주며 설명한다.

1. Application

본 절에서는 전기차 충전소 경로 및 대기시간 기반 추천 및 예약시스템에서 사용자와 상호 작용하는 부분인 iOS 애플리케이션의 구현에 대해 기술한다.

1) Chargers.swift

```
//여기서부터는 구조체정의
struct Chargers:Codable{
    var chargers: [Charger]?

    enum CKeys:String,CodingKey{
        case chargers = "chargers"
    }

    init(from decoder: Decoder) throws {
        let values = try decoder.container(keyedBy: CKeys.self)
        chargers = try values.decodeIfPresent([Charger].self, forKey:
            .chargers)
    }
}
```

<그림 12> Chargers 구조체 선언 코드

Chargers 구조체: 변수로 Charger 배열을 갖는다. 이 배열에는 모든 충전소 정보가 저장된다.

```

struct Charger:Codable
{
    var statNm:String?
    var statId:String?
    var chgerId:String?
    var chgerType:String?
    var addr:String?
    var lat:String?
    var lng:String?
    var useTime:String?
    var busiId:String?
    var busiNm:String?
    var busiCall:String?
    var stat:String?
    var statUpdDt:String?
    var powerType:String?

    enum Keys: String,CodingKey{
        case statNm
        case statId
    }
}

```

<그림 13> Charger 구조체 상태변수

Charger 구조체: 공공데이터 포털에서 제공해주는 충전소 이름, 충전소 ID, 주소, 위도, 경도, 사용 가능한 시간, 충전 타입 등의 정보들을 변수로 정의해서 JSON 타입의 충전소 정보를 읽어 구조체를 생성하도록 구현했다.

```

{ "chargers" : [
  {
    "statNm": "롯데리아 산북점",
    "statId": "CV000000",
    "chgerId": "01",
    "chgerType": "02",
    "addr": "전라북도 군산시 공단대로 566",
    "lat": "35.97039200000",
    "lng": "126.68426200000",
    "useTime": "24시간 이용가능",
    "busiId": "CV",
    "busiNm": "대형제비터",
    "busiCall": "1522-2573",
    "stat": "2",
    "statUpdDt": "20200928102337",
    "powerType": null
  },

```

<그림 14> 공공데이터 포털 데이터(JSON 형태)

공공데이터 포털에서 제공하는 충전소 데이터(JSON 형태)

```

init(from decoder: Decoder) throws {
    let values = try decoder.container(keyedBy: Keys.self)
    statNm = try values.decodeIfPresent(String.self, forKey: .statNm)
    statId = try values.decodeIfPresent(String.self, forKey: .statId)
    chgerId = try values.decodeIfPresent(String.self, forKey: .chgerId)
    chgerType = try values.decodeIfPresent(String.self, forKey: .chgerType)
    addr = try values.decodeIfPresent(String.self, forKey: .addr)
    lat = try values.decodeIfPresent(String.self, forKey: .lat)
    lng = try values.decodeIfPresent(String.self, forKey: .lng)
    useTime = try values.decodeIfPresent(String.self, forKey: .useTime)
    busiId = try values.decodeIfPresent(String.self, forKey: .busiId)
    busiNm = try values.decodeIfPresent(String.self, forKey: .busiNm)
    busiCall = try values.decodeIfPresent(String.self, forKey: .busiCall)
    stat = try values.decodeIfPresent(String.self, forKey: .stat)
    statUpdDt = try values.decodeIfPresent(String.self, forKey: .statUpdDt)
    powerType = try values.decodeIfPresent(String.self, forKey: .powerType)
}

```

<그림 15> Chargers 구조체 초기화 코드

다음과 같이 구조체 초기화 코드를 JSON 형식에 맞게 작성하고, JSON 데이터를 읽으면서 데이터를 Decoding해 구조체에서 정의한 변수에 맞게 저장한다.

```
let chargerArray = try decoder.decode(Chargers.self, from: chargerData) //json을 디코드해서
데이터화
```

<그림 16> chargerArray 배열 선언 코드

위 그림과 같이 코드 한 줄로 chargerArray 배열에 모든 충전소의 세부정보가 저장된다.

2) MapViewController.swift

```
let camera = GMSCameraPosition.camera(withLatitude: 35.554539999999, longitude: 129.38815999999, zoom: 15.0)
let mapView = GMSMapView.map(withFrame: self.view.frame, camera: camera)
mapView.delegate = self
mapView.isMyLocationEnabled = true
mapView.settings.myLocationButton = true;
self.view.addSubview(mapView)
self.view = mapView

for mark in markerList{
    mark.map = mapView
}
}
```

<그림 17> 지도 실행 시, 초기화 변수 및 카메라 줌 설정 코드

위 그림은 Google Map UI를 화면에 표시하기 위한 부분이다. Camera에는 처음 지도를 실행했을 때의 위치, 확대/축소 범위를 저장한다. 또한 mapView.delegate = self 코드는 맵을 클릭했을 때의 이벤트를 설정하도록 해주는 부분이다. 그리고 현재 위치 표시를 Enable하게 해준다. mapView를 실행하게 되면 사용자가 정해진 값을 바탕으로 지도를 UI에 보여준다.

for문은 충전소 마커를 찍어주는 부분으로, markerList에는 모든 충전소의 위도, 경도 정보가 들어있어 모든 markerList값을 지도에 마커로 추가해주게 된다.


```

func mapView(_ mapView: GMSMapView, didTap marker: GMSMarker) -> Bool { //Marker에 가는 이벤트
    let statId = marker.title
    let desc = try? getChargerStatus(statId!)

    let alert = UIAlertController(title: "충전소 세부정보", message: desc, preferredStyle: .alert)

    alert.addAction(UIAlertAction(title: "예약", style: .default, handler: { (action) in //예약버튼눌렀을때

        selectedId = statId!
        //음성낼 바인딩
        if let controller = self.storyboard?.instantiateViewController(identifier: "ChargersTableViewController"){

            // 2. 찾은 컨트롤러로 이동한다. (push Controller)
            self.navigationController?.pushViewController(controller, animated: true)
        }

    }

    alert.addAction(UIAlertAction(title: "확인", style: .cancel, handler: nil))

    self.present(alert, animated: true)

    return true
}
}

```

<그림 18> marker 클릭 시 이벤트 처리 함수

이 함수는 Marker를 클릭했을 때 발생하는 이벤트에 대해 정의한 함수이다. 클릭한 마커를 파라미터로 받아와서 해당 마커(충전소)의 충전소 id와 세부정보를 getCharStatus함수를 호출해서 가져온다. getCharStatus함수는 충전소의 id(고유값)으로 검색해서 세부정보를 텍스트 형식으로 반환한다. 따라서 이 정보를 desc변수에 저장하고, UIAlertController를 호출해서 alert창에 충전소 세부정보를 표시해주고 예약 버튼까지 보여주는 기능을 수행한다.

위 이벤트가 실행되면 다음과 같은 Alert 창이 표시된다.



<그림 19> 충전소 세부정보 알림창

3) URLRequest.swift

```
func httpRequestHandler(_ ChargerID:String) throws -> String {
    let baseUrl = "http://34.64.73.242:3000/api/getChargerInfo/"
    let completedUrl = baseUrl + ChargerID
    let url = URL(string: completedUrl)
        let response = try String(contentsOf: url!)
    let firstIndex = response.index(response.startIndex, offsetBy: 1)
    let lastIndex = response.index(response.endIndex, offsetBy: -1)
    let parsedResponse = "\(response[firstIndex..
```

<그림 20> Url Request 함수

위 그림은 Url Request를 하는 함수이다. 충전소의 이름, 주소, 위도, 경도 등의 정적 정보들은 저장해서 사용할 수 있지만, 현재 이용 가능 상태와 같은 동적 정보들은 사용자가 요청할 때마다 서버에서 확인을 해야 한다. 그 과정을 http 리퀘스트를 통해서 자동화해주는 함수이다. 충전소 ID(고유값)을 파라미터로 받아서 http 요청을 보내고 결과 값을 파싱해서 JSON형식의 데이터를 String타입으로 반환하는 기능을 수행한다.


```

func getChargerStatus(_ charger: String) throws -> String{

    let urlResponse = try httpRequestHandler(charger)
    let parsedSegment = splitTheResponse(urlResponse)
    var i = 0
    var numberOfChargers = parsedSegment.count
    var numberOfAvailable = 0
    var returnValue:String = ""
    var isAvailable:String = "이용 불가능"
    while i < numberOfChargers{
        if let chargerData = parsedSegment[i].data(using: .utf8)
        {
            let decoder = JSONDecoder()

            do {

                let chargerData = try decoder.decode(Charger.self, from: charg
                if chargerData != nil{
                    let stateName:String? = chargerData.statNm
                    let chargerAddress:String? = chargerData.addr
                    let chargerUseTime:String? = chargerData.useTime
                    let chargerBusinessName:String? = chargerData.busiNm
                    let chargerState:String? = chargerData.stat
                    if chargerState == "2" {
                        numberOfAvailable += 1
                    }
                    if(i==0) {
                        if stateName != nil{
                            returnValue += ("충전소 명 : " + stateName! + "\n")
                        }
                    }
                }
            } catch {
                print("JSONDecoderError: \(error)")
            }
        }
        i += 1
    }
    return returnValue
}

```

<그림 21> 충전소의 세부정보를 반환하는 함수

위 그림은 충전소의 ID를 받아 세부정보를 반환해주는 함수이다. 위에서 정의한 httpRequestHandler함수를 사용해서 JSON 형식의 데이터를 파싱한 뒤, 사용자가 보기 편한 방식으로 변환해주는 함수이다. 즉, id를 파라미터로 서버에 httpRequestHandler함수를 이용해서 호출하고, 리턴값을 바탕으로 파싱해서 각각의 변수에 넣어준 뒤, 그 변수들을 보기 좋게 바꿔주는 역할을 한다.

```

func httpRequestHandlerGetReservationInfo(_ StatID:String, _ ChargerID:String) throws -> [String] {
    let baseUrl = "http://34.64.73.242:3000/api/getReserveInfo/"
    let completedUrl = baseUrl + StatID + "/" + ChargerID
    let url = URL(string: completedUrl)
    let response = try String(contentsOf: url!)
    let firstIndex = response.index(response.startIndex, offsetBy: 2)
    let lastIndex = response.index(response.endIndex, offsetBy: -2)
    let parsedResponse = "\(response[firstIndex..

```

<그림 22> 예약 정보를 반환하는 함수

충전소 ID를 받아와 예약 정보를 반환하는 함수이다. baseUrl에 저장된 주소로 httpRequest를 전송하고, 받은 정보를 바탕으로 예약 현황을 확인한다.

```

func sendPost(paramText: String, urlString: String) {
    // paramText를 데이터 형태로 변환
    let paramData = paramText.data(using: .utf8)

    // URL 객체 정의
    let url = URL(string: urlString)

    // URL Request 객체 정의
    var request = URLRequest(url: url!)
    request.httpMethod = "PUT"
    request.httpBody = paramData

    // HTTP 메시지 헤더
    request.addValue("application/x-www-form-urlencoded", forHTTPHeaderField: "Content-Type")
    request.setValue(String(paramData!.count), forHTTPHeaderField: "Content-Length")

    // URLSession 객체를 통해 전송, 응답값 처리
    let task = URLSession.shared.dataTask(with: request) { (data, response, error) in

        // 서버가 응답이 없거나 통신이 실패
        if let e = error {
            NSLog("An error has occurred: \(e.localizedDescription)")
            return
        }

        // 응답 처리 로직
        DispatchQueue.main.async() {
            // 서버로부터 응답된 스트링 표시
            let outputStr = String(data: data!, encoding: String.Encoding.utf8)
            print("result: \(outputStr!)")
        }
    }

    // POST 전송
    task.resume()
}

func makeReservation(stdid: String, chgid: String, time:String) {
    let paramText = "statId=\(stdid)&chgerId=\(chgid)&times=\(time)"
    sendPost(paramText: paramText, urlString: "http://34.64.73.242:3000/api/reserves")
}

```

<그림 23> 예약을 요청하는 sendPost 함수

위 그림은 충전소 예약 기능을 구현한 부분이다. sendPost 함수는 Url을 만들어서 서버에 POST request를 전송한다. 전송된 정보로 인해 서버DB의 충전소 예약 테이블을 변경한다. makeReservation 함수에서는 충전소 id와 충전기 id, 예약 시간을 파라미터로 받아 makePost를 호출해주는 방식으로 예약 기능을 마친다.

4) Direction Request

```
8  import Foundation
9
10 typealias LngLatPosition = [Double]
11
12 struct DirectionPaths : Codable {
13     var list: [LngLatPosition]?
14
15     init(from decoder: Decoder) throws {
16         list = try decoder.singleValueContainer().decode([LngLatPosition].self)
17     }
18 }
```

<그림 24> 경로를 받아 저장하는 DirectionPaths 구조체 코드

위 그림은 네이버 Direction 5 API를 통해 경로를 응답받아서 저장하기 위한 구조체이다. 경로는 길이 2인 [경도, 위도] 형식의 위치 좌표 정보를 여러 개 갖는 list로 이루어진다.

```
119 func directionRequest() {
120     let origin_lng = String(format: "%lf", Double(origin.coordinate.longitude))
121     let origin_lat = String(format: "%lf", Double(origin.coordinate.latitude))
122     let destination_lng = String(format: "%lf", Double(destination.coordinate.longitude))
123     let destination_lat = String(format: "%lf", Double(destination.coordinate.latitude))
124     // let option: String
125     let parameters = "start=" + origin_lng + "," + origin_lat
126         + "&" + "goal=" + destination_lng + "," + destination_lat
127     // + "&" + "option=" + option
128
129     let baseUrl: String = "https://naveropenapi.apigw.ntruss.com/map-direction/v1/driving"
130
131     let completeUrl = URL(string: baseUrl + "?" + parameters)
132     var request = URLRequest(url: completeUrl!)
133     request.addValue("application/json", forHTTPHeaderField: "Accept")
134     request.addValue(naverClientId, forHTTPHeaderField: "X-NCP-APIGW-API-KEY-ID")
135     request.addValue(naverClientSecret, forHTTPHeaderField: "X-NCP-APIGW-API-KEY")
136 }
```

<그림 25> 경로를 요청하는 directionRequest 함수 코드 중 일부

위 그림은 네이버 Direction 5 API를 이용해 경로를 요청하기 위한 함수이다. 출발지, 도착지의 위치 좌표 정보를 이용해서 요청 Url을 생성한다.

2. Server

본 절에서는 앞 절에서 기술한 Application에서 이루어지는 요청을 처리하는 Server의 구현에 대해 기술한다.

1) DB (DataBase)

공공데이터 포털 API에서 불러온 정보는 형식에 맞춰 구글 클라우드 서버의 DB에 저장한다. DB Table의 형태는 한국환경공단에서 제공하는 데이터의 형식을 참고해 작성하였다. 충전소의 고유 ID인 statId와 충전소 내부에 존재하는 충전기의 고유 ID인 chgerID를 묶어 Primary Key로 관리하여 중복되지 않도록 구현하였다.

| Field | Type | Null | Key | Default | Extra |
|-----------|--------------|------|-----|---------|-------|
| statNm | varchar(100) | NO | | NULL | |
| statId | varchar(8) | NO | PRI | NULL | |
| chgerId | varchar(2) | NO | PRI | NULL | |
| chgerType | varchar(2) | NO | | NULL | |
| addr | varchar(150) | NO | | NULL | |
| lat | varchar(20) | NO | | NULL | |
| lng | varchar(20) | NO | | NULL | |
| useTime | varchar(50) | YES | | NULL | |
| busiId | varchar(2) | YES | | NULL | |
| busiNm | varchar(50) | YES | | NULL | |
| busiCall | varchar(20) | YES | | NULL | |
| stat | varchar(1) | YES | | NULL | |
| statUpdDt | varchar(14) | YES | | NULL | |
| powerType | varchar(20) | YES | | NULL | |

<그림 26> DataBase에 구현된 evCharger Table

App에서 클릭 이벤트를 통해 충전소 정보를 확인할 시, 그림 26의 DB 테이블 정보를 기준으로 통신한다. 매분 statId와 chgerId를 기준으로 stat 값을 업데이트해 충전소들의 실시간 상태를 반영한다.

2) REST API

앱과 통신하기 위해서 REST API[11] 형태를 기준으로 Node.js의 express[12] 라이브러리를 적용하였다. REST란, 데이터를 이름(자원의 표현)으로 구분하여 데이터의 상태(정보)를 주고 받는 모든 것을 의미한다. HTTP 프로토콜을 사용하므로 REST API 사용을 위한 별도의 인프라 구축이 필요 없고, 표준 프로토콜을 따르는 모든 플랫폼에서 사용할 수 있다.

```
app.get("/api/getChargerInfo/:statId", (req, res) => {
  const { statId } = req.params;
  connection.query(`SELECT * FROM test where statId = "${statId}"`, (error, rows, fields) => {
    if(error){
      throw error;
    }
    else{
      res.send(rows);
      console.log('Info is : ', rows);
    }
  });
});
```

<그림 27> 충전소 세부정보를 반환하는 함수

충전소 클릭 시, 충전소에 관한 모든 정보를 전달하는 get 함수이다. 해당 함수를 통해 사용자는 충전소의 추가 정보(사용 가능 시간, 현재 예약 현황 등)를 요청하고, 서버는 해당 정보를 사용자에게 전송한다.

```
app.get("/api/getChargerStatus/:statId", (req, res) => {
  const { statId } = req.params;
  connection.query(`SELECT statNm, statId, chgerId, stat, statUpdDt FROM test where statId = "${statId}"`, (error, rows, fields) => {
    if(error){
      throw error;
    }
    else{
      res.send(rows);
      console.log("Info is : ", rows);
    }
  });
});
```

<그림 28> 충전기 세부정보를 반환하는 함수

충전소에 존재하는 충전기 클릭 시, 충전기에 관한 세부정보를 전달하는 get 함수이다. 해당 함수를 통해 사용자는 충전기의 추가 정보를 요청하고, 서버는 해당 정보를 사용자에게 전송한다.

```

app.get("/api/getStationMarker/:bigLat/:smallLat/:bigLong/:smallLong/:stat", (req,
res) => {
    const { bigLat } = req.params;
    const { smallLat } = req.params;
    const { bigLong } = req.params;
    const { smallLong } = req.params;
    const { stat } = req.params;
    connection.query(`SELECT * FROM test where ((lat BETWEEN "${smallLat}" AND
"${bigLat}") AND (lng BETWEEN "${smallLong}" AND "${bigLong}")) AND (stat = "${stat
}"))`, (error, rows, fields) => {
        if(error){
            throw error;
        }
        else{
            res.send(rows);
            console.log('Info is : ', rows);
        }
    });
});

```

<그림 29> 경로상 사용 가능한 충전소 정보를 전달하는 함수

사용자가 출발지와 목적지를 설정하면, 해당 경로 내에서 사용 가능한 충전소를 나타내주는 함수이다. 사용자의 화면에 현재 예약이 없고, 사용이 가능한 경로상에 있는 충전소를 확인할 수 있다.

V. 전기차 충전소 경로 및 대기시간 기반 추천 및 예약시스템 시험

본 장에서는 전기차 충전소 경로 및 대기시간 기반 추천 및 예약시스템의 정상적인 기능 확인을 위한 테스트를 진행하였다. 테스트 시나리오 과정에 따라 충전소 표시, 세부정보 확인, 예약, 경로 찾기, 충전소 추천 기능을 테스트하였다. 테스트 시나리오는 다음과 같다.

1. 테스트 시나리오

테스트는 다음과 같은 시나리오를 따라 이루어진다.

1. iOS 애플리케이션 실행 및 지도 보기 클릭
2. 현재 위치를 클릭해서 공대 5호관 주변으로 이동
3. 근처 충전소 마커를 클릭해 세부정보 확인
4. 예약 버튼을 클릭해 현재 이용 가능한 충전소 확인
5. 13:00 - 14:00 시간대 예약
6. 경로 찾기로 충남대 - 대전역 경로 검색
7. 경로상의 충전소 마커 하나를 클릭해 세부정보 확인
8. 예약 버튼을 클릭해 19:00 - 20:00 시간대 예약

2. 테스트 환경

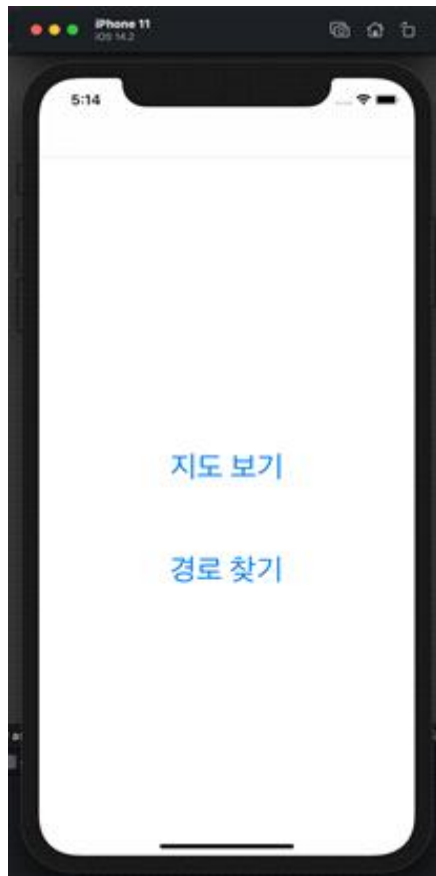
앞 절의 테스트 시나리오는 다음과 같은 환경에서 구동하였다.

테스트 환경: Xcode 12.2, Swift 5.3, iPhone11 Simulator

3. 프로젝트 데모

본 절에서는 앞 두 절에서 기술한 테스트 환경에서 테스트 시나리오를 수행한 결과를 보여준다. 각 시나리오 수행단계는 iOS 애플리케이션의 단계별 동작 화면과 단계별 설명으로 구성된다.

1. iOS 애플리케이션 실행 및 지도 보기 클릭



<그림 30> 애플리케이션 초기화면

사용자는 전기차 충전소 경로 및 대기시간 기반 추천 및 예약시스템 실행 시 본 화면을 처음 마주한다. 초기화면에서 주변 위치의 충전소를 확인하는 ‘지도 보기’ 버튼과 경로상의 충전소를 추천받을 수 있는 ‘경로 찾기’ 버튼을 클릭할 수 있다.

2. 현재 위치를 클릭해서 공대 5호관 주변으로 이동



<그림 31> 현재 위치 표시 화면

‘지도 보기’ 버튼을 클릭한 후, 사용자는 현재 위치를 지도에 반영하는 기능을 사용할 수 있다. 오른쪽 아래의 동그란 버튼을 클릭하면, 사용자의 현재 위치를 기반으로 지도가 이동한다. 지도에서 표시되는 빨간색 마커는 주위에 존재하는 전기차 충전소를 의미한다. 마커 클릭 시, 충전소 세부정보를 확인할 수 있다.

3. 근처 충전소 마커를 클릭해 세부정보 확인



<그림 32> 충전소 세부정보 화면

사용자는 충전소 세부정보를 확인할 수 있다. 세부정보는 공공데이터 포털에서 가져온 정보를 실시간으로 반영하고 있으며, ‘확인’ 버튼을 클릭 시, 창을 닫을 수 있고, ‘예약’ 버튼을 클릭 시 예약을 진행할 수 있는 화면으로 이동한다.

4. 예약 버튼을 클릭해 현재 이용 가능한 충전소 확인



<그림 33> 사용 여부를 확인할 수 있는 화면

예약을 진행할 수 있는 화면으로 초록색 아이콘은 현재 사용가능한 충전소, 빨간색 아이콘은 현재 사용 불가능한 충전소를 의미한다. 사용 유무와 상관없이 특정 시간대에 예약한 사람이 없으면 사용자는 예약을 진행할 수 있다.

5. 13:00 - 14:00 시간대 예약



<그림 34> 예약 요청 화면



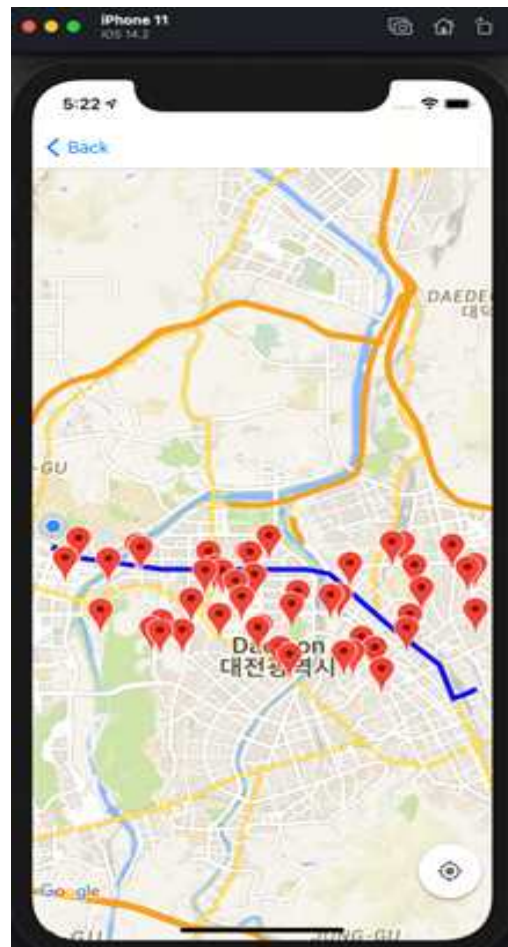
<그림 35> 예약 완료 화면

사용자는 특정 시간대를 기준으로 예약을 진행할 수 있다. <예약 불가>는 예약이 불가능하다. 당일 시간대를 기준으로 예약을 진행할 수 있다. 예약이 완료되면 해당 시간대는 <예약 불가> 표시가 나타난다.

6. 경로 찾기로 충남대 - 대전역 경로 검색



<그림 36> 출발지, 목적지 설정 화면



<그림 37> 경로상의 충전소 표시 화면

‘경로 찾기’ 버튼을 클릭 시, 사용자는 목적지와 출발지를 기준으로 적절한 충전소를 추천받을 수 있다. 경로상에 표시되는 충전소는 현재 사용이 가능한 충전소들을 필터링한 결과이며 해당 충전소를 클릭 시 예약을 진행할 수 있다.

7. 경로상의 충전소 마커 하나를 클릭해 세부정보 확인



<그림 38> 경로상의 충전소 클릭 시 화면

사용자는 추천된 충전소를 기반으로 예약을 진행할 수 있다. 목적지까지 가는 경로에 있는 충전소이므로 다른 길을 알아보지 않고 곧바로 충전소로 이동할 수 있다. 해당 기능을 사용함으로써 사용자는 추천 서비스에 대한 편의성을 느낄 수 있고, 예약서비스를 통해 시간을 절약할 수 있다.

8. 예약 버튼을 클릭해 19:00 - 20:00 시간대 예약



<그림 39> 경로상의 충전소
예약 요청 화면



<그림 40> 경로상의 충전소
예약 완료 화면

사용자는 필터링된 충전소 예약을 진행할 수 있다. ‘지도 보기’에서 진행한 예약보다 목적지까지 가는 경로상의 충전소를 예약하는 것으로 시간 절약과 편리함을 느낄 수 있다. 이전에 설명한 예약 기능과 마찬가지로 당일 특정 시간대를 예약할 수 있고, 예약이 완료되면 <예약 불가>로 표시되는 것을 확인할 수 있다.

4. 결과 분석

데모 시나리오를 따라 테스트를 진행한 결과, 모든 과정이 성공적으로 수행되었다. 애플리케이션을 실행하고 지도 보기를 클릭하면 초깃값이 지정된 지도가 Google Map API를 통해서 실행되고, 사용자가 우측 하단의 현재 위치 버튼을 클릭하면 현재 위치로 지도가 이동한다.

또한, 현재 위치 주변의 전기자동차 충전소가 마커로 표시된다. 해당 마커는 정적 정보로서 이미 충전소 리스트에 모든 충전소의 위도, 경도, 이름, 주소 등의 정보가 저장되어있어 지도에 보여줄 수 있다. 사용자가 그 마커 중 하나를 클릭하면 클릭한 충전소 ID를 이용해서 서버에 요청 보내고, 돌아온 응답을 파싱해서 사용자에게 현재 그 충전소의 상태 정보를 보여준다. 그리고 사용자가 그 정보를 보고 예약 버튼을 클릭하면, 이전에 전송한 ID 값을 바탕으로 상태 정보를 더 상세하게 보여주고, 이용 가능한 충전소는 초록색 UI로, 불가능한 충전소는 빨간색 UI로 표시한다.

사용자가 충전소를 클릭하면 예약 테이블 페이지로 넘어가고, 가능한 시간대에 예약을 요청하면, 서버로 POST 요청이 전송되어 해당 충전소의 예약 테이블의 값을 수정하면서 예약이 완료된다.

다음으로 경로 찾기 버튼을 클릭하고, 출발지, 도착지를 각각 충남대학교 정문, 대전역으로 입력하면 (Google Place API를 통한 auto complete) 지정된 출발지, 도착지 정보를 바탕으로 Naver Map API를 통해서 경로 정보를 JSON 형식으로 받아온다. 요청된 데이터들을 Google Map API의 Poly Line으로 이어주면서 경로를 보여준다. 또한, 그 주변의 충전소를 보여주는 데, 이때는 이용 가능한 충전소만 필터링 과정을 거쳐서 보여준다. 충전소 마커를 클릭하면 동일한 방식으로 예약이 진행된다. 이 모든 과정이 이용에 불편이 없을 정도의 속도로 API Request 및 Response가 이루어지는 것을 확인할 수 있었다.

Ⅵ. 결론

본 연구를 통해서 전기자동차 충전소 탐색 애플리케이션 제작을 완료했다. 애플리케이션을 통해서 전기자동차 사용자는 현재 위치한 지역 주변의 충전소를 검색할 수 있고, 해당 충전소의 세부정보(충전소 이름, 주소, 포트, 이용 가능 여부 등)를 확인할 수도 있다. 그래서 전기자동차를 이용하면서 충전소를 찾는 시간을 줄여줄 수 있다. 또한, 충전소 예약시스템을 제시해서 적용했다. 이를 통해서 사용자에게 도착 예상 시간에 맞추어 충전소를 예약하고, 지연시간 없이 빠르게 충전소를 이용할 수 있는 서비스를 제공한다. 추가로 경로상의 충전소 탐색 기능은 사용자가 출발지, 도착지 정보를 입력하면 이동할 경로를 파악해서, 경로상에 존재하는 충전소 중에서 이용이 가능한 충전소만 필터링하는 과정을 거친 뒤 사용자에게 보여준다. 따라서 전기차 사용자가 장거리를 운전해야 할 경우, 이동할 경로에 있는 충전소를 파악할 수 있고 예약시스템을 통해 예약해서 충전 계획을 세울 수 있다. 그로 인해서 충전으로 인해 소요되는 총 시간을 줄여줄 수 있다. 이 시스템을 통해 늘어나고 있는 전기차 사용자에게 편의를 제공할 것이다. 그리고 특정 충전소가 추가될 시 DB에 해당 충전소의 정보만 넣어준다면, 사용자의 화면에서 그대로 표시될 수 있는 유연성을 지니고 있다.

본 연구에서 제작한 애플리케이션을 조금 더 사용자 맞춤형으로 적용한다면, 사용자의 배터리 잔량, 포트 등의 정보를 구체적으로 입력받아서 가장 최적의 충전소를 개인 맞춤형으로 추천해줄 수 있을 것이다. 또한, 충전소에 차량 인식 카메라를 설치한다면 이 데이터를 공유해서 현재 해당 충전소에 대기 중인 차량의 대수까지 확인이 가능할 것으로 예상된다. 이를 활용하면 사용자들의 편의성을 한층 더 증가시킬 것이다.

VII. 참고 문헌

- [1] A new wave of air pollution crises,
<https://www.unenvironment.org/news-and-stories/story/new-wave-air-pollution-crises-what-can-be-done>
- [2] 친환경차 60만대 돌파,
<https://www.korea.kr/news/policyNewsView.do?newsId=148868349>
- [3] 충전속도에 따른 전기차 충전기 종류,
https://www.ev.or.kr/portal/chargerkind?pMENUST_ID=21629
- [4] About Swift - The Swift Programming Language, <https://docs.swift.org/swift-book/>
- [5] Google Cloud Platform, <https://cloud.google.com/compute?hl=ko>
- [6] Naver API - Directions5,
https://docs.ncloud.com/ko/naveropenapi_v3/maps/direction/driving.html
- [7] HTTPS Protocol, <https://ko.wikipedia.org/wiki/HTTPS>
- [8] 공공데이터 포털, <http://www.cvinfo.com/news/articleView.html?idxno=11449>
- [9] 전기자동차 충전소 현황,
<https://data.go.kr/tcs/dss/selectApiDataDetailView.do?publicDataPk=15059139>
- [10] what is JSON, <https://www.json.org/json-en.html>
- [11] REST API란?, <https://www.redhat.com/ko/topics/api/what-is-a-rest-api>
- [12] Node.js - express library, <https://expressjs.com/ko/4x/api.html#app.all>