종합설계 프로젝트 수행 보고서

프로젝트명	시각장애인을 위한 버스 승하차 보조 시스템 'We Tayo'
팀 번호	S2-3
문서 제목	수행계획서 2차 발표 중간보고서( ) 3차 발표 중간보고서 최종결과보고서

2021.03.04

팀원: 강석원 (팀장)
이지수
홍의성
박형근

지도교수 : 공 기 석 (인)

문서 수정 내역

작성일	대표작성자	버전(Revision)		수정내용
2021.02.22	강석원(팀장)	1.1	수행계획서	최초작성
2021.03.04	강석원(팀장)	1.2	수행계획서	내용추가작성
2021.02.22	강석원(팀장)			
2021.02.22	강석원(팀장)			
2021.02.22	강석원(팀장)			
2021.02.22	강석원(팀장)			
2021.02.22	강석원(팀장)			

문서 구성

진행단계	프로젝트 계획서 발표	중간 발표1 (3월)	중간 발표2	학기 말 발표	최종발표
기본양식	계획서 양식	계획서 양식	계획서 양식	계획서 양식	계획서 양식
포함되는 내용	Ⅰ. 서론 (1~6) Ⅱ. 본론 (1~3) 참고자료	Ⅰ. 서론 (1~6) Ⅱ. 본론 (1~4) 참고자료			

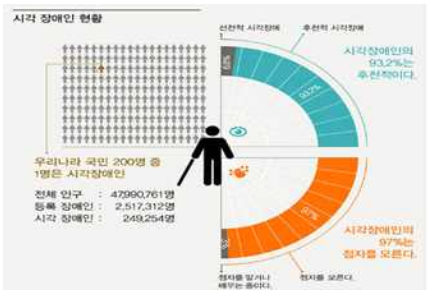
이 문서는 한국산업기술대학교 컴퓨터공학부의
“종합설계” 교과목에서 프로젝트 “시각장애인을 위한 버스 승하차 보조 시스템 We Tayo”을 수행하는
(S2-3)들이 작성한 것으로 사용하기 위해서는 팀원들의 허락이 필요합니다.

목차

Ⅰ.서론

1. 작품 선정 배경 및 필요성

2019년 통계청 등록 장애인 현황에 따르면 우리나라 국민 200명 중 1명은 시각장애가 있는 것으로 파악된다. 그중 6.8%는 선천적 시각장애를, 93.2%는 후천적 시각장애가 있는 것으로 조사되었다. 적지 않은 수의 시각장애인이 우리 주변에서 살아가고 있는 것을 알 수 있다. 2016년 장애인 실태조사에서 시각장애인의 일상생활에서의 불편사항 조사결과 1순위로는 위치 찾기(84.3%), 2순위는 대중교통 탑승(63.7%)이 집계되었다. 기존에 존재하는 점자블록과 대중교통 안내시스템이 시각장애인에게 도움이 되지 못하고 있음을 알 수 있었다.
이에 우리는 시각장애인의 정상적인 사회참여와 그들의 기본권리인 '이동권'을 보장하고 그들도 우리와 같은 사회구성원으로서 인정받고 존중받아야 한다고 생각한다. 시각장애인의 주도적인 버스 이용을 위한 탑승 보조 시스템 'WeTayo'를 개발하기로 하였다. **<그림 1>**은 시각장애인 실태 현황, **<그림 2>**는 일상생활에서 시각장애인이 겪는 불편사항에 대한 조사결과 내용 중 일부이다.



<그림 1> 시각장애인 실태 현황

<그림 2> 일상생활에서 겪는 불편사항

<그림 1>에서 발생원인을 살펴보면 전체 등록 장애인 2,517.312명 중 시각장애인의 수는 249,254명으로 전체 등록 장애인 수 중 10%가 시각장애인임을 알 수 있다.
또한, 전체 시각장애인 중 93.2%가 후천적 요인의 시각장애인으로 많은 수의 시각장애인들이 후천적으로 장애를 지니게 되었음을 알 수 있다. **<그림 2>**를 살펴보면 시각장애인의 불편사항 순위 중 1순위는 위치 찾기(84.3%), 2순위는(63.7%), 3순위는 정보습득(56%)으로 집계되었다. 대중교통 이용이 63.7%로 두 번째 불편사항이라는 것은 기존에 존재하는 점자블록과 버스 안내시스템이 제 기능을 수행하고 있다고 보기 어렵다. 우리는 현재 시각장애인들이 어떻게 버스를 이용하고 있는지 조사해 보았다.
아래의 **<그림 3>**, **<그림 4>**, **<그림 5>**는 시각장애인이 실제 버스 탑승 과정에서 발생하는 문제점들을 MBC 소수의견 "*버스 탄 시각장애인 봤나요?…타고 내리는 게 전쟁 (2019.12.25.)*" 동영상에서 발췌한 것이다.

<그림 3> 정차 위치 혼란

<그림 4> 기존 안내시스템의 부정확성

<그림 5> 하차 벨 위치 찾기

<그림 3>을 보면 시각장애인이 버스 정차 위치를 몰라 엉뚱한 곳에서 버스를 기다리는 것을 알 수 있다. 기존에 존재하는 점자 블럭은 시각장애인들에게 실질적인 도움을 주지 못하며, 버스탑승을 위한 정류장을 찾는 것 조차에 어려움을 겪고 있는 것을 알 수 있다.

<그림 4> 또한 기존에 존재하는 버스 안내시스템의 오작동을 확인할 수 있다. 비장애인들은 버스 안내시스템이 오작동해도 시각적인 정보를 통하여 원하는 버스를 쉽게 선별할 수 있지만, 오로지 청각적인 부분에만 의존해야 하는 시각장애인들에게 버스 안내시스템의 부정확성은 그들에게 있어 큰 애로사항임을 알 수 있다.

<그림 5>를 통해서는 하차 벨을 찾기 어려운 시각장애인의 모습을 볼 수 있다. 막상 어렵게 원하는 버스에 탑승하더라도 하차 벨의 위치를 파악할 수 없는 시각장애인들에게 원하는 곳에 내리기 위한 하차 벨 작동은 꼭 필요한 기능임을 알 수 있다.

우리가 구현하고자 하는 버스 승하차 보조 시스템은 크게 4가지 기능을 구현할 것이다. 첫째, 사용자 위치기반 정류장 선정, 둘째, APP 을 통한 실시간 버스 도착 정보 아림 시스템, 셋째 도착 버스 알림 스피커, 마지막으로 무선 하차 벨의 구현이다.

2. 기존 연구 / 기술 동향 분석

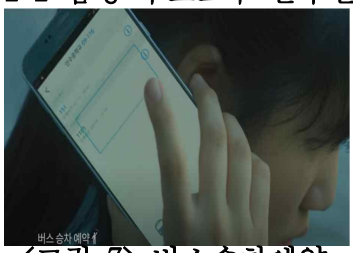
시각장애인의 버스탑승보조시스템을 설계하기 위하여 우리는 기존에 적용된 사례가 있는 탑승보조시스템들을 조사하였다.

2-1 음향유도신호기를 변형한 버스 안내시스템

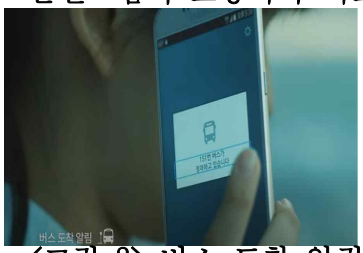
<그림 6> 음향유도신호기를 이용한 안내시스템

<그림 6>은 아주대학교 학생들이 시각장애인의 음향유도신호기를 변형하여 만든 버스 안내시스템이다. 리모컨의 버튼을 누르면 버스 앞문 위의 수신기에서 신호를 받아 버스 번호를 알려주는 기능을 수행한다. 리모컨의 다른 버튼을 누르면 하차 벨을 무선으로 구동할 수 있다.

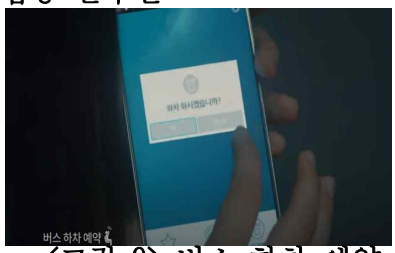
2-2 삼성 투모로우 솔루션 '손길' 팀의 교통약자 버스탑승 솔루션



<그림 7> 버스승차예약



<그림 8> 버스 도착 알림



<그림 9> 버스 하차 예약

<그림 7>, <그림 8>, <그림 9>는 삼성 투모로우 솔루션에서 소개된 '손길' 팀의 교통약자 버스 탑승 솔루션의 기능들이다. 크게 버스승차예약, 정류장 버스 도착 알림, 버스 하차 예약, 하차 벨 기능을 구현한 것을 알 수 있다. 우리 'WeTayo' 팀이 구현하려는 기능들과 거의 유사하다는 것을 알 수 있다.

현재 나와 있는 여러 탑승 보조 시스템들은 각자 나름의 시각장애인을 위한 편의 기능을 제공하고 있지만, 단점이 명확하다. 첫째, 음향유도신호기를 이용한 안내시스템의 경우, 리모컨을 이용하여 단순 버스 번호 알림 기능과 하차 벨 기능만을 제한적으로 제공하고 있다. 둘째, 손길 팀의 교통약자 버스 탑승 솔루션의 경우, 승차예약과 도착 알림, 하차 벨을 구현하였지만, 하나의 노선에 제한적, 시범적으로 적용된 한계가 있다. 또한, 사용자 위치기반의 정류장 제공기능의 부재로 실제 시각장애인이 사용하는 데 제약이 따른다.

우리 'We Tayo' 팀은 이러한 단점을 보완하여 실생활에서 정말 사용되는 실용적인 버스 탑승 시스템을 제작할 것이다. 또한, 이용자들이 시각장애인이기 때문에 그들의 눈높이에 맞춘 원하는 기능 등을 개발하는 것에 초점을 맞추는 것이 목표이다.

[표 1] 현재 출시된 버스탑승보조시스템과 “WeTayo” 과의 비교표

	버스 번호 알림 스피커	무선 하차 벨	실시간 버스 정보	APP을 통한 하차 벨 동작	사용자 위치기반 정류장 선정
We Tayo	O	O	O	O	O
삼성 투모로우 솔루션	O	O	O	O	X
음향유도기 솔루션	O	O	X	X	X

3. 개발목표

‘WeTayo’ 팀은 시각장애인의 주도적인 버스 이용이 가장 큰 목표이다. 이와 더불어 그들도 사회구성원으로서 당당하게 일원으로 인정받아 비장애인과 비장애인이 서로 협력하여 공생하는 생산적인 사회 가치를 창출 할 것이다.

3.1. WeTayo의 개발목표

첫째, 사용자 스마트폰의 GPS를 기반으로 사용자의 현재 위치에서 가장 가까운 정류장을 서버의 공공데이터와 대조하여 사용자에게 안내하는 기능을 제공한다. 둘째, 경기도 공공데이터 실시간 버스 위치를 이용하여 사용자가 탑승하고자 하는 버스가 몇 정거장 뒤에 도착하는지 사용자에게 APP 을 통하여 알릴 수 있도록 구현한다. 또한, 사용자가 탑승을 희망하는 버스의 버스 기사에게 해당 정거장에 시각장애인이 존재함을 알림으로 전하여 시각장애인이 버스를 놓치는 경우를 방지한다. 셋째, 버스가 정류장에 도착했을시, 외부 스피커를 통해 버스 번호를 출력하여 시각장애인이 버스에 탑승할 수 있도록 유도한다. 넷째, 라즈베리 파이의 비콘 센서를 활용하여 사용자가 APP을 통해 하차 벨을 눌렀을 때 하차 벨이 울리도록 구현한다.

3.2. WeTayo의 추가적인 개발목표

현재 WeTayo의 서비스 지역을 경기도 시흥 지역으로 한정하고 있다. 서비스를 제공하며 적용 지역을 시흥을 중심으로 경기도, 수도권으로 확장할 계획이다. 이용대상 또한 시각장애인뿐만 아니라 노인, 어린이, 장애인 등 사회적 약자까지 사용 가능 할 수 있도록 서비스 기능을 확대할 예정이다. 우선적으로 3.1의 목표들을 확실히 구현한 뒤에 추가적으로 확대 적용할 예정이다.

4. 팀 역할 분담

4.1 자료수집에 대한 팀 역할 분담

[표 2] 자료수집 역할 분담표

	역할
강석원 (팀장)	Flutter의 기능에 대한 사전 조사, 네이버 TTS Clova API 적용 조사, GPS 오차 범위 조사, 주제에 대한 유사 사례 조사, 앱에 적용할 UI를 위한 장애인 접근법 조사
박형근	개발 방법 및 개발 환경 조사, 유사 사례 조사, 하드웨어 및 모듈 조사, Flutter의 비콘을 활성화할 수 있는 라이브러리 조사
이지수	Flutter의 기능에 대한 사전 조사, 네이버 TTS Clova API 적용 조사, 하드웨어 및 모듈 조사, 유사 사례 조사
홍의성	AWS 사용을 위한 자료 조사, 프로세스 관리 툴 조사, 버스 공공 API 조사. GPS 오차 범위 조사, Flutter와 DB 및 서버 간의 통신

[표 3] 설계 및 구현 역할 분담표

	역할
강석원 (팀장)	IOS 앱과 서버 통신 설계 및 구현, DB 설계, 앱 UI 설계, 버스 공공 API 적용
박형근	하차 벨 회로설계, Raspberry PI BLE 활성화, 스피커와 센서의 동작 구현
이지수	Android 앱과 서버 통신 설계 및 구현, Clova TTS API 적용, GPS 적용
홍의성	AWS 서버 설계 및 구축, DB 설계 및 물리적 DB 구현

- 설계 및 구현에 대한 표이다. 크게 강석원 학우는 IOS 앱 구현, 박형근 학우는 하드웨어 구현, 이지수 학우는 Android 앱 구현, 홍의성 학우는 DB 및 서버 구현으로 나누어 역할을 분담하였다.

※ 테스트
수시로 Github를 통한 version 및 issue 관리와 공동 테스트 진행

5. 개발 일정

[표 4] 개발 세부일정

	상세 내용	2020		2021					
		11	12	1	2	3	4	5	6~9
주제선정	- 각자의 주제 의견 수렴 후 최종 주제선정								
	- 최종적인 개발목표 수립								
현장조사 및 요구사항 수집	- 관련된 유사 연구 및 사례 조사								
	- 시각장애인의 버스 탑승에 대한 불편사항 조사를 통해 요구사항 수집								
	- 시스템 수행 시나리오 구상								
	- 시나리오의 흐름대로 동작을 위한 H/W, S/W의 개발 구상								
개발 환경 조사 및 학습	- AWS EC2 서버 사용방법 조사								
	- Flutter 개발 환경 조사								
	- 공동 프로세스 관리를 위한 툴 조사								
	- 사용될 공공 API 선정 후 DB와의 크롤링 방법 조사								
개발 환경 구축	- Github의 공동 repository 개설, 프로세스 관리 툴 Zenhub 설치								
	- VS Code 내의 extension을 통해 Flutter 개발 환경 셋팅, Intellij와 MySQL 설치								
	- AWS EC2 서버 구축								
필요 알고리즘 개발	- 장애인 접근법을 고려한 프로토타입에 맞는 UI/UX 확보								
	- 버스 공공 API 크롤링 후 DB와의 연동								
	- 버스 정류장 GPS와 현재 위치 비교 후 가장 가까운 정류장 선정								
	- 비콘과 Raspberry PI BLE 기능을 이용해 무선 하차 벨 동작								
데모 및 유지보수	- 전반적인 시나리오 흐름대로의 동작 여부 테스트								
	- 프로토타입 개발 및 유지보수 진행								
최종검토 및 발표	- 논문, 결과보고서 작성								
	- 최종검토 후 마감								

6. 개발 환경

[표 5] 소프트웨어 개발 환경

S/W Development Environment	구체 사항	비고
AWS EC2	서버 구축을 위한 가상환경 OS	- Ubuntu Linux 20.04 - 최대 30GB - RAM 1GB
MySQL	데이터베이스 SQL IDE	사용 언어 : SQL Version : 8
Intellij	서버 개발을 위한 IDE	사용 언어 : Java Version : 20.2
Visual Studio Code	앱 개발을 위한 Flutter-Dart IDE	사용 언어 : Flutter-Dart Flutter Version : 1.17
Rasbian	Raspberry PI 전용 OS	사용 언어 : C/Pytion Version :1.4
Github Action	자동 빌드 툴	무료
AWS s3	빌드 된 파일을 옮겨 저장하기 위한 곳	5GB까지 무료, get 요청은 2만 건 put 요청을 2천 건
AWS Codedeploy	자동 배포 툴	무료
Nginx	무중단 배포를 위해 포트 포워딩 이용	무료
CloudWatch	EC2 사용량, AWS lambda 스케줄링, 로그 기록 이용	무료
AWS lambda	매일 새벽 2시에 공공 API에서 데이터를 다운로드 받아 DB 업데이트	월 100만 건 또는, 초당 10만 GB 무료

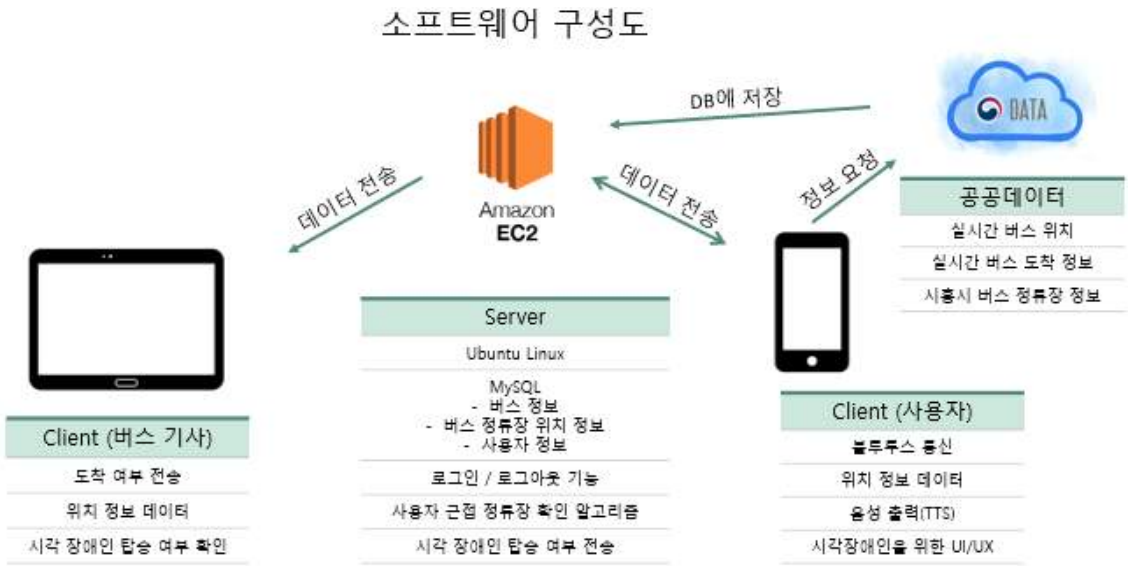
[표 6] 하드웨어 개발 환경

H/W Development Environment	구체 사항	비고
Rasberry PI 3B+	비콘 통신을 위한 BLE 기능, 마그네틱 도어 센서, 스피커, 하차 벨과의 연결	- CPU : 1.4GHz Cortex-A53 - Bluetooth 4.2 - 무선 LAN - 5V/2.5A DC 전원 입력
마그네틱 도어 센서	도착 신호를 위한 도어 센서	입력 전압 : DC ~ 12V 작동 거리 : 약 20mm 크기 : 27*14*10mm
차량 외부 스피커	도착 알림을 위한 스피커	
하차 벨 회로	하차를 위한 하차 벨	

Ⅱ. 본론

1. 개발 내용

소프트웨어적으로 원하는 버스를 원활히 이용 수 있도록 돕는 사용자(시각장애인)와 시각장애인의 탑승을 돕는 버스 기사용 앱을 개발하며 앱 간의 통신을 관리하는 서버를 구축한다. 하드웨어적으로는 소프트웨어에서 제어해줄 라즈베리 파이를 통해 도착을 송출하는 스피커와 사용자가 하차를 원할 시 울릴 하차 벨과의 연결을 수행한다. <그림 10>은 앱과 서버 간의 통신을 나타낸 소프트웨어 구상도이다.



<그림 10> 소프트웨어 구성도

1.1 사용자용 앱

사용자 앱 개발은 Flutter 환경에서 Dart 언어를 이용해 개발하며 앱 간의 통신은 AWS EC2 서버로 관리된다. 또한, 장애인 접근법을 고려한 UI/UX를 반영해 최대한 단순 동작 기능과 앱 화면 Widget의 가시성을 높이고 고 대비의 화면을 적용한다.

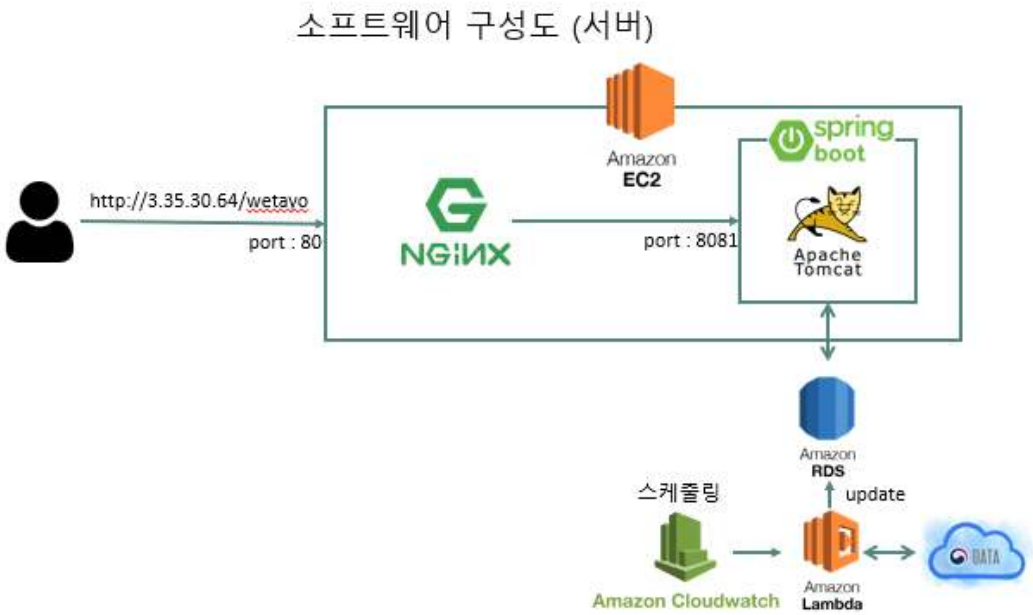
앱 동작 시에는 서버 내에서 GPS 위치를 비교해 사용자와 가장 가까운 정류소를 공공 API의 정류소 좌표를 통해 식별한다. 정류소에 운행되는 버스 명단을 리스트로 안내한다. 타고자 하는 버스 선택 시 해당 버스의 도착 예상 시간을 안내하고 서버를 통해 버스 기사용 앱으로 탑승희망 메시지를 전송한다. 모든 버스 정보는 서버에서 공공 API 내 정보에서 선별된 DB를 통해 가져온다. 앱 내에서 시각장애인에게 음성으로 정보를 전달하는 음성합성(speech synthesis) 기술은 네이버 클로바 TTS를 사용한다. 사용자가 버스에 탑승한 후 하차를 원하면 화면의 하차 벨 버튼 클릭 시 버스에 장착된 라즈베리 파이의 BLE 기능을 이용한 비콘 센서 통신을 통해 무선 하차 벨 기능을 구현한다.

1.2 버스 기사용 앱 기능

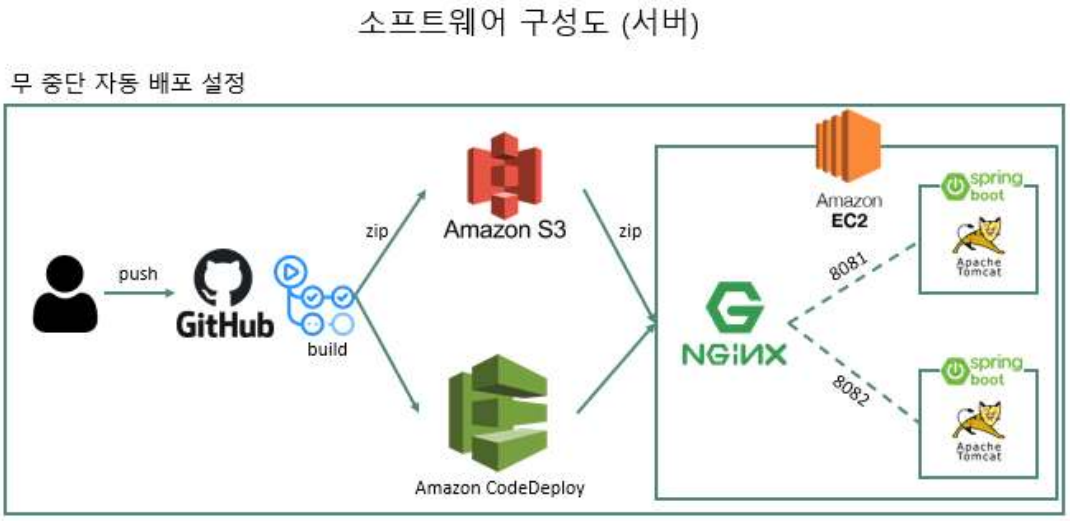
버스 기사용 앱 개발 또한 Flutter 환경에서 Dart 언어를 이용해 개발하며 앱 간의 통신은 AWS EC2 서버로 관리된다.

앱 동작 시에 서버를 통해 사용자 앱으로부터의 탑승희망 메시지를 전달받으면 화면에 해당 메시지를 출력하고 시각장애인이 위치한 정류소를 화면에 표시한다. 해당 정류소에 도착하여 시각장애인이 탑승하면 탑승 확인 메시지를 서버에서 보내며 확인 시에 사라지게 된다.

1.3 서버



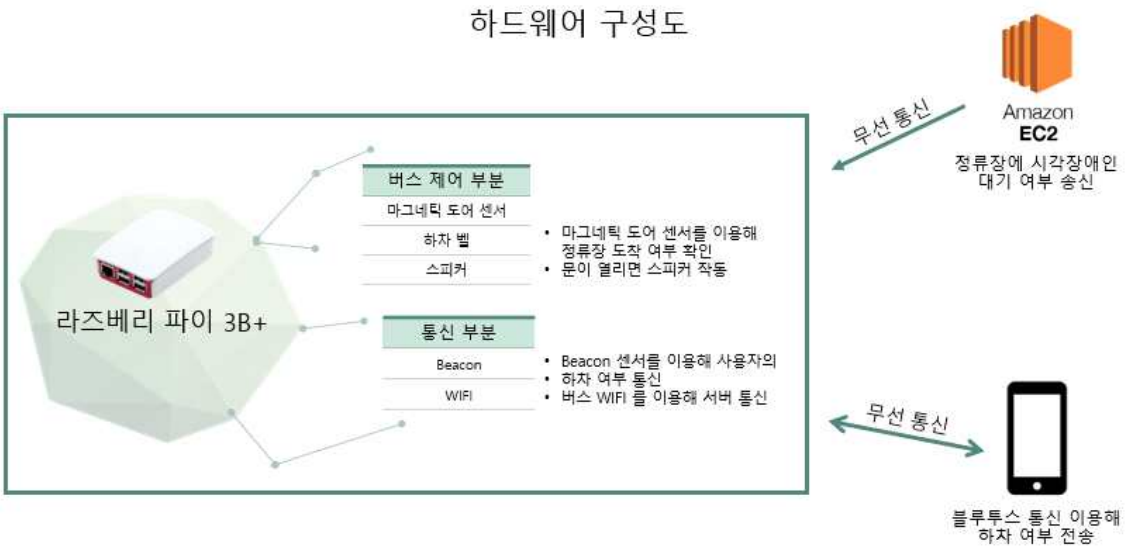
<그림 11> 소프트웨어 구성도 (서버)



<그림 12> 소프트웨어 구성도 (서버)

서버의 개발은 AWS EC2를 사용하여 Ubuntu/Linux OS 환경에서 Apache Tomcat을 구축한다. DB의 경우 공공 API에서 선별된 데이터를 종합해 MySQL을 통해 구축된다. 사용자 앱과 버스 기사 앱 간의 데이터 전송을 담당하며 앱에서 요청하는 정보를 DB에서 가져와 데이터를 처리한다. 또한, 사용자가 개발 후 github action을 통해 자동 build 되며 AWS S3에서 build 된 파일을 옮겨 저장하고 AWS CodeDeploy 에서 자동 배포를 담당한다. 이 과정을 거쳐 AWS EC2 서버 내에서 nginx를 통해 무중단 배포를 위한 포트 포워딩이 이뤄진다.

1.4 하드웨어



<그림13> 하드웨어 구성도

2. 문제 및 해결방안

종합설계를 진행하면서 다양한 문제들에 직면하게 된다. 개발 과정에서 발생이 예상되는 문제와 그에 대한 해결방안을 [표 7]로 정리했다.

[표 7] 버스탑승보조시스템 개발 시 예상되는 문제점과 그에 대한 해결방안 표

순번	문제 및 해결방안
1.	**Q.버스 보조 시스템의 실제 테스트는 어떻게 진행할 것인가?**
	A. 우선 아두이노 보드와 간이 스피커를 구현하여 정상적으로 작동하는지 확인 후, 실제 팀원의 자차에 부착하여 실제 버스 환경에서 적용될 수 있는지 최종 확인할 것이다.
2.	**Q. 실제 버스의 실시간 위치 정보와 노선을 확인하려면 서버의 용량과 기능이 중요한데 어떤 서버를 사용할 것이고, 그 서버를 선택한 이유는 무엇인가?**
	A. 현재 AWS의 프리티어를 사용하고 있다. 최대 1G 용량으로 일정 용량이 초과하면 서버가 다운되는 현상이 발생한다. 향후 데모 테스트를 진행하고 실제 구현단계에서 비용을 지불 하여 서버를 업데이트할 예정이다. 오토 스케일링(Auto Scaling) 기능을 사용하여 유동적으로 서버의 용량을 조절하는 방법도 있다.

3.	**Q.비콘 센서에서 하차 벨이 울리는 기능과 문이 닫히는 기능을 어떻게 구현할 것인가?**
	A.현재 버스회사에서 회로도를 받아 콘솔박스와 하차 벨을 조립하여 간이 하차 벨을 구현하였다. 향후 스피커와 아두이노 보드를 추가하여 하드웨어적 부분을 개발·보수할 예정이다.
4.	**Q.사용자의 실시간 위치를 확인하고 비교하는 알고리즘에서 오류 사항이나 중복 오차값이 발생하지 않는가?**
	A. 이미 스마트폰에 탑재된 GPS 기능 자체가 뛰어나고, '하버사인공식'을 활용한 위치 보정 알고리즘으로 정확한 위칫값을 측정할 수 있다.

3. 시험 시나리오

[표 8]은 전체 프로그램을 통합하여 수행할 때, 문제가 발생하는지 검증하기 위해 작성한 테스트 시나리오이다. 각 기능을 테스트케이스로 정의하여 테스트케이스 수행 시 예상되는 결과를 바탕으로 작성하였다. 테스트케이스의 ID는 각 테스트케이스가 S/W 파트인지 H/W 파트인지 나타낸다.

[표 8] 테스트케이스와 그에 따른 예상 결과표

ID	테스트케이스	정의	예상결과
SW01	사용자 위치 파악 기능	앱 실행 시, GPS 기능을 활성화하여 사용자의 위치를 측정한다.	가장 가까운 정류장을 안내한다.
SW02	버스 선택 기능	서버로부터 해당 정류장에 정차하는 버스의 데이터를 받아 사용자에게 안내한다.	정류장의 버스를 선택할 수 있다.
SW03	버스 선택정보 전송	선택된 버스의 탑승희망 의사가 버스 기사에게 전달된다.	승차 정보가 서버로 전송되고 서버에서 버스 기사에게 전송한다.
HW04	버스 기사 확인	탑승 여부 버스 기사의 앱으로 전송된다.	버스 기사가 확인하고 정류장에서 사용자를 탑승시킬 수 있다.
HW01	버스 도착 알림	정류장에 도착하면 스피커로 버스 번호 송출	스피커로 버스 번호 송출한다.
HW02	비콘 및 하차 벨 동작	앱으로 구현된 하차 벨을 선택하면 하차 벨이 구현된다.	무선 하차 벨 동작

테스트 순서는 첫째, 각 H/W 모듈별, S/W 모듈별 단위테스트, 둘째, 서로 연관성 있는 모듈끼리의 통합테스트, 셋째, 기능별로 서로 관련이 있는 모듈의 통합테스트 순으로 진행한다.

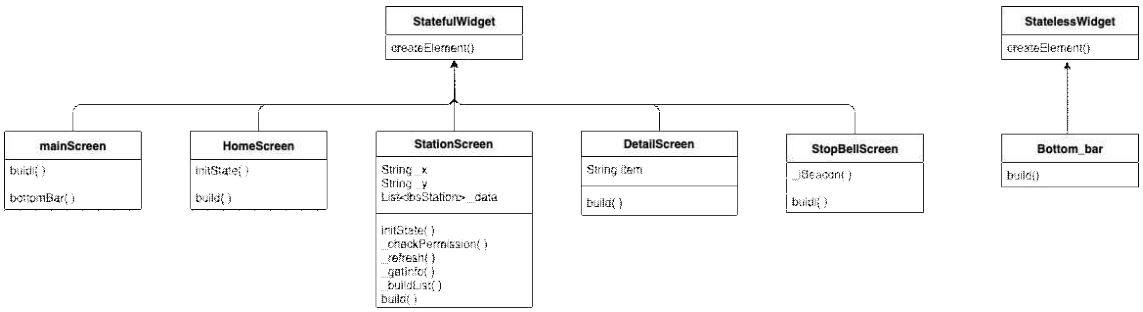
테스팅은 개발 시작과 동시에 진행한다. 이에 따라 개발 후반부에 발생할 결함이나 오류를 최대한 빨리 발견하고 수정하는 것을 목적으로 한다. 또한, 이로 인하여 개발 프로젝트 기간을 전체적으로 단축하는 효과를 기대한다.

4. 상세설계

4.1 소프트웨어(APP) 상세설계

4.1.1 소프트웨어(APP) 시스템 모듈 클래스 설계

아래 **<그림 14>**는 소프트웨어(APP) 시스템 모듈 클래스다이어그램이다. 위젯을 빌드(build) 하는 각 페이지와 페이지의 기능들을 클래스(class)화 하였다.



<그림 14> 소프트웨어(앱) 시스템 모듈 클래스 다이어그램

※ Bottom_bar 클래스는 상태변화가 없는 위젯이기 때문에 StatelessWidget 클래스를 상속하고 그 외 mainScreen, StationScreen, HomeScreen, DetailScreen, stopBellScreen는 상태에 따른 변화를 보여야 하는 이유로 StatefulWidget을 상속받는다.

◆ mainScreen 클래스

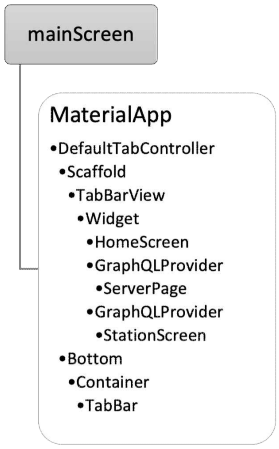
아래 **[표 9]**는 mainScreen에 대한 라이브러리, 변수, 메서드를 정리한 표이다.

[표 9] mainScreen 클래스

이름	mainScreen
기능	사용자 앱의 3가지 탭의 뼈대를 구성한다. 각 탭은 <Widget> 속성 내에서 해당하는 클래스를 호출 하여 위젯을 구성한다. 레이아웃의 아래쪽 Bottom은 Bottom_bar 클래스를 호출 하여 구성한다.
라이브러리	import 'package:flutter/material.dart' import 'package:graphql_flutter/graphql_flutter.dart' import 'package:wetayo_app/api/config.dart' import 'package:wetayo_app/screen/home_screen.dart' import 'package:wetayo_app/screen/station_screen.dart' import 'package:wetayo_app/widget/bottom_bar.dart'
변수	X
메서드	Widget build(BuildContext context)

mainScreen는 사용자 앱의 메인 화면을 구성한다. 공통으로 사용되는 위젯인 Bottom_bar 위젯을 build() 메소드를 통해 위젯을 표현하고 3개의 탭(HomeScreen, stopBellScreen, StationScreen)에 위젯을 그린다.

아래 **<그림 15>**는 mainScreen의 build() 함수의 위젯 속성과 계층을 구체화한 계층도 이다.



<그림 15> build() 함수 위젯 속성의 계층도

mainScreen은 각 탭의 뼈대 역할을 하는 클래스이다. 총 3개의 탭으로 구성되어 있으며 StationScreen 클래스는 GraphQLProvider 메소드를 통해 서버와 통신할 수 있도록 구성하였다.

◆ HomeScreen 클래스

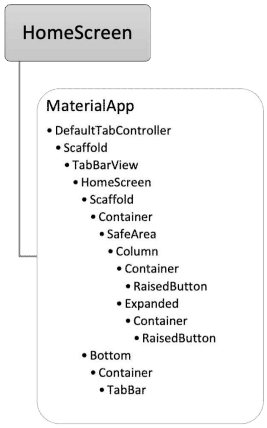
아래 **[표 10]**은 HomeScreen에 대한 라이브러리, 변수, 메서드를 정리한 표이다.

[표 10] HomeScreen 클래스

이름	HomeScreen
기능	사용자 앱의 실행 시 보이는 가장 첫 번째 레아이웃이다.
라이브러리	import 'package:flutter/material.dart'
변수	X
메서드	Widget build(BuildContext context)

HomeScreen은 앱이 실행하면 처음으로 보여주는 첫 번째 레이아웃 클래스이다. 두 개의 RaisedButton 위젯이 존재하며 각각 ‘정류소 선택’과 ‘즐겨찾기’ 버튼이 배치한다.

아래 **<그림 16>**은 HomeScreen의 build() 함수의 위젯 속성과 계층을 구체화한 계층도 이다.



<그림 16> build 함수의 위젯 속성과 계층도

HomeScreen은 MainScreen의 기본 뼈대는 그대로 유지하고 SafeArea 영역 안에 Column(세로)으로 RaisedButton 두 개를 배치하고 있다.
각 RaisedButton은 '정류소 선택', '즐겨찾기' 화면과 연결되며 각 버튼의 세로 크기는 화면을 가득 채울 수 있도록 double.infinity 속성을 주고 높이는 화면 비율에 맞게 6 : 4 비율로 채울 수 있도록 구성하였다.

◆ StationScreen 클래스

아래 **[표 11]**은 StationScreen에 대한 라이브러리, 변수, 메서드를 정리한 표이다.

[표 11] StationScreen 클래스

이름	StationScreen
기능	사용자의 현재 위치(위도, 경도)를 조회하여 100m이내의 가장 가까운 정류소를 화면에 버튼과 리스트로 표시한다.
라이브러리	import 'package:flutter/material.dart' import 'package:graphql_flutter/graphql_flutter.dart' import 'package:wetayo_app/screen/detail_page.dart' import 'package:permission_handler/permission_handler.dart' import 'package:geolocator/geolocator.dart' import 'package:flutter/services.dart'
변수	String _x // 경도 String _y // 위도
메서드	_checkPermission() async _refresh() async Widget build(BuildContext context) Widget _buildList(BuildContext context, QueryResult result)

HomeScreen은 앱이 실행하면 처음으로 보여주는 첫 번째 레이아웃 클래스이다. 두 개의 RaisedButton 위젯이 존재하며 각각 '정류소 선택'과 '즐겨찾기' 버튼이 배치한다.

◆_checkPermisson() 함수

아래 **[표 12]**는 디바이스의 위치조회 권한을 설정하기 위한 메소드 _checkPermission()에 대해 기술한 표이다.

[표 12] _checkPermission( ) 함수

형식	_checkPermission() async
리턴 값	X
설명	앱이 설치된 디바이스에 위치조회 권한을 설정하기 위한 함수이다. (권한이 이미 설정되어 있으면 다시 권한을 요청하지는 않는다.)
예시	_checkPermission()

_checkPermission()는 비동기 함수로 앱 실행 최초에만 실행된다. 위치조회 권한을 부여하기 위해 PermissionHandler()로 권한을 사용자에게 요청한다.

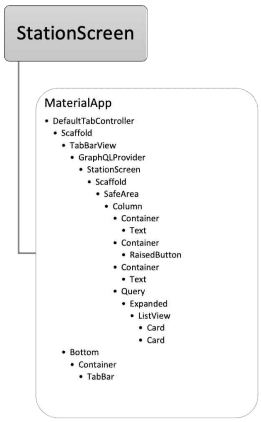
아래 **[표 13]**은 사용자가 위치한 위도, 경도 좌푯값을 조회하기 위한 메소드 _refresh() 에 대해 기술한 표이다.

[표 13] _refresh( ) 함수

형식	_refresh() async
리턴 값	X
설명	사용자의 현재 위치 (위도, 경도)를 조회하기 위한 함수이다.
예시	_refresh()

_refresh()는 비동기 함수로 화면이 새로고침 될 때마다 요청하여 사용자의 현재 위치 (위도, 경도)를 조회한다. 요청이 성공하면 StationScreen의 매개변수인 _x와 _y에 각각 저장되고 쿼리를 요청할 때 해당 변수를 사용한다.

아래 **<그림 17>**은 StationScreen의 build() 함수의 위젯 속성과 계층을 구체화한 계층도 이다.



<그림 17> refresh 함수의 위젯 속성과 계층도

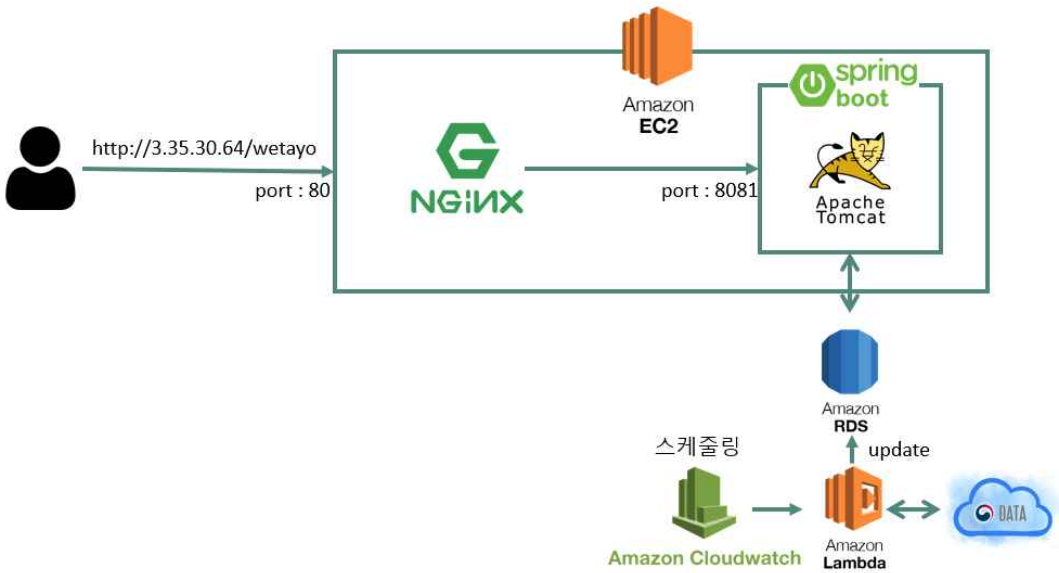
StationScreen 또한 마찬가지로 MainScreen의 기본 뼈대는 그대로 유지하고 있다. 위젯은 Text - RaisedButton - Text - ListView 순서로 빌드 되며 첫 번째 버튼은 요청한 쿼리의 0번째 순서인 가장 가까운 위치의 정류소가 Text로 배치된다.

Query는 Flutter가 제공하는 GraphQL 라이브러리를 사용하여 서버에 요청 하게 된다. 쿼리의 인자는 String _x, String _y, String distance가 차례로 들어가게 되고 응답 결과를 만들어 둔 _buildList 메소드를 호출 하여 return 하게 된다.

4.2.2 서버설계

◆서버 구성도

아래 **<그림18>**은 사용자와 서버 간의 통신과정과 DB 업데이트 과정을 표현한 그림이다.



<그림18> 사용자와 서버 간의 통신과정

AWS EC2는 3.35.30.64의 ip 주소를 갖고 'wetayo'하나의 endpoint를 갖고 이를 통해 query, mutation을 이용할 수 있다.

http://3.35.30.64/wetayo 도메인에 사용자가 요청을 보내면 EC2 내부의 nginx가 http 요청을 받고 해당 도메인을 Apache Tomcat이 작동하고 있는 8081번 포트로 포트포워딩을 해줌으로써 웹 요청을 처리한다.

Spring boot로 작성된 서버는 Amazon RDS의 MySQL 8과 통신한다. DB에 저장된 정류장, 노선정보는 AWS Lambda를 이용하여 공공 API로부터 다운받아 RDS에 update 하도록 구축하였고 매일 새벽 2시에 자동으로 update 되도록 Amazon CloudWatch를 이용하여 스케줄링을 해주었다.

◆DB 테이블 설계

■ RIDE

아래 **[표 14]**는 탑승희망 여부를 표현하는 RIDE 테이블의 컬럼에 대해 정리한 표이다.

[표 14] RIDE 테이블 컬럼

RIDE	
ID	AUTO_INCREASEMENT
STATION_ID	정류장 ID
ROUTE_ID	노선 ID

STATION_ID, ROUTE_ID는 복합키로 해당 복합키를 갖는 컬럼이 존재한다면 탑승희망하는 사용자가 존재한다는 의미이다.

■ ROUTE_STATION

아래 **[표 15]**는 노선별 정류장에 대한 정보를 가지고 있는 테이블의 컬럼에 대해 정리한 표이다.

[표 15] 노선정보 테이블 컬럼

ROUTE_STATION	
STATION_ID	정류장 ID
ROUTE_ID	노선 ID
UP_DOWN	버스 방향 (정, 역)
STATION_ORDER	정류장 순서
STATION_NAME	정류장 이름
ROUTE_NUMBER	노선 번호

■ STATION

아래 **[표 16]**은 정류장에 대한 정보를 가지고 있는 테이블의 컬럼에 대해 정리한 표이다.

[표 16] 정류장 정보 테이블 컬럼

STATION	
STATION_ID	정류장 ID
STATION_NAME	정류장 이름
CENTER_ID	중앙차로 ID
CENTER_YN	중앙역 여부
REGION_NAME	지역 이름
MOBILE_NUMBER	정류장 고유 MOBILE NUMBER
GPS	GPS (위도 , 경도) 정보
DISTRICT_CODE	관할 지역

◆ ROUTE

[표 17] 루트 테이블

ROUTE			
ROUTE_ID	노선 ID	UP_FIRST_TIME	기점 첫차 시간
ROUTE_NUMBER	노선 번호	UP_LAST_TIME	기점 막차 시간
ROUTE_TP	노선 유형	DOWN_FIRST_TIME	종점 첫차 시간
START_STATION_ID	기점 정류소 ID	DOWN_LAST_TIME	종점 막차 시간
START_STATION_NAME	기점 정류소 이름	PEEK_ALLOC	최소 배차 시간
START_STATION_NUMBER	기점 정류소 번호	NPEEK_ALLOC	최대 배차 시간
END_STATION_ID	종점 정류소 ID	COMPANY_ID	운수업체 ID
END_STATION_NAME	종점 정류소 이름	COMPANY_NAME	운수업체 이름
END_STATION_NUMBER	종점 정류소 번호	TEL_NUMBER	운수업체 전화번호
REGION_NAME	지역 이름	DISTRICT_CODE	관할 지역

■ 서버 GraphQL Schema

◆ 요청 Schema

- Query

아래 **[표 18]**은 get 요청인 Query의 종류에 대해 정리한 표이다.

[표 18] get 요청 Query 종류

Query 명	반환 타입	매개변수		설명
getRide	Boolean	stationId	정류장 Id	정류장별 노선의 탑승희망 여부을 조회한다.
		routeId	노선 Id	
getRoute	[Route]	regionName	지역 이름	해당 지역의 노선정보를 조회한다.
getStation	[Station]	gpsY	위도	근접한 정류장 정보를 조회한다.
		gpsX	경도	
		distance	반경 거리	
getStationAndRoute	[StationAndRoute]	gpsY	위도	근접한 정류장 정보와 노선정보를 조회한다.
		gpsX	경도	
		distance	반경 거리	

- Mutation

아래 **[표 19]**는 insert/delete 요청인 Mutation의 종류에 대해 정리한 표이다.

[표 19] Mutation 종류

Query 명	반환 타입	매개변수		설명
createRide	Ride	stationId	정류장 Id	탑승희망을 요청하면 DB에 삽입한다.
		routeId	노선 Id	
deleteRide	Boolean	stationId	정류장 Id	탑승을 완료했거나 버스가 지나갔다면 DB에서 삭제한다.
		routId	노선 Id	

◆ 응답 Schema

- type Station

아래 **[표 20]**은 getStation query의 응답 타입 schema에 대해 정리한 표이다.

[표 20] getStation query schema 정리한 표

변수 명	타입	설명
stationId	Int	정류장 Id
stationName	String	정류장 이름
mobileNumber	String	정류장 mobile 번호
distance	Int	현재 위치까지의 거리

- type Route

아래 **[표 21]**은 getRoute query의 응답 타입 schema에 대해 정리한 표이다.

[표 21] getRoute query schema 정리한 표

변수 명	타입	설명
routeId	Int	노선 Id
routeNumber	String	노선 번호

- type StationAndRoute

아래 **[표 22]**는 getStationAndRoute의 응답 타입 schema에 대해 정리한 표이다.

[표 22] getStationAndRoute의 응답 타입 schema에 대해 정리한 표

변수 명	타입	설명
stationId	Int	정류장 Id
stationName	String	정류장 이름
mobileNumber	String	정류장 mobile 번호
distance	Int	현재 위치까지의 거리
routes	[Route]!	type Route의 배열

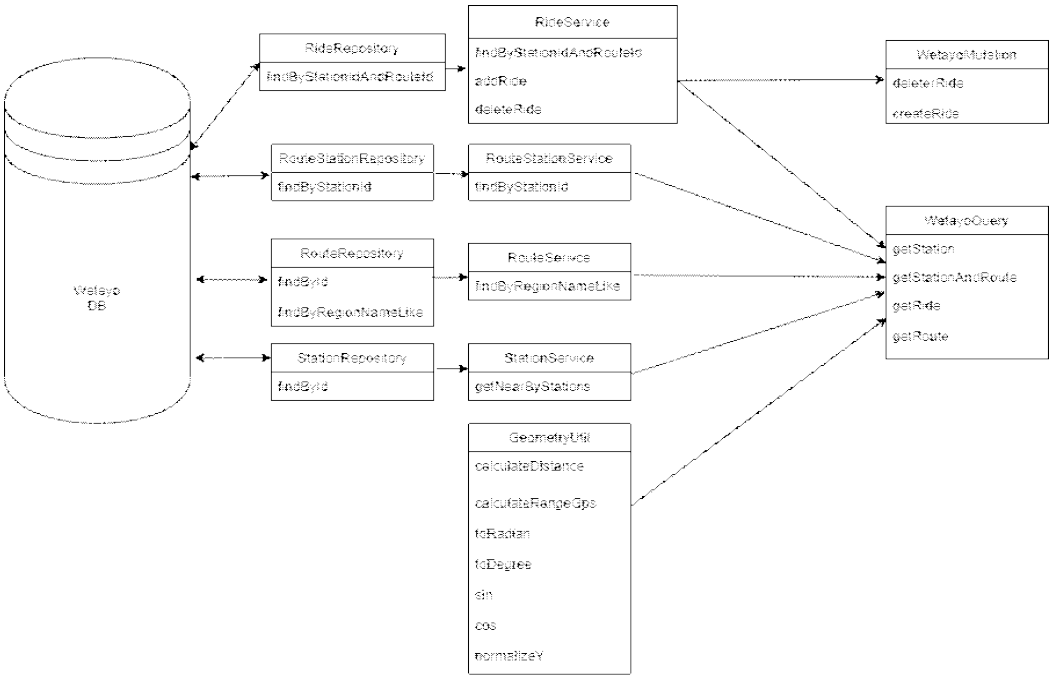
- type Ride

아래 **[표 23]**은 createRide Mutation의 응답 타입인 Ride schema에 대해 정리한 표이다.

[표 23] createRide Mutation의 응답 타입인 Ride schema에 대해 정리한 표

변수 명	타입	설명
routeId	Int!	노선 Id
stationId	Int!	정류장 Id

■ 서버 시스템 모듈 클래스 설계



<그림 19> 서버 시스템 모듈 클래스 다이어그램

spring boot를 이용해 query, mutation 부분으로 나누어 설계했으며, 각 endpoint를 구성하는 service들과 db와 접근하는 repository 인터페이스의 설계도를 나타낸다.

◆ 사용 라이브러리

아래 **[표 24]**는 서버에서 사용한 라이브러리와 사용 목적에 대한 정리한 표이다.

[표 24] 서버에서 사용한 라이브러리와 사용 목적에 대해 정리한 표

사용 라이브러리	spring-boot-starter-data-jpa	DB 접근을 query가 아닌 객체 지향적으로 접근하기 위한 JPA
	spring-boot-starter-validation	Enity(Table)의 컬럼들의 값을 유효성 검사
	graphql-spring-boot-starter:11.0.0	GraphQL 관련 메서드들 제공
	mysql:mysql-connector-java javax.persistence-api jackson-datatype-jts hibernate-spatial	MySQL 접근과 POINT와 같은 Geometry 타입을 사용하기 위한 라이브러리
	org.projectlombok:lombok	Getter, Setter, Constructor 등 자동 생성을 위한 라이브러리
	modelmapper	Entity와 요청 객체 간의 Serialize, Deserialize 를 쉽게 해주는 라이브러리

◆ WetayoQuery 클래스

아래 **[표 25]**는 WetayoQuery에 대한 변수, 메서드를 정리한 표이다.

[표 25] WetayoQuery 클래스

이름	class WetayoQuery
기능	wetayo endpoint에 query가 들어오면 요청에 맞는 응답을 제공해주는 클래스
변수(의존성)	private final StationService stationService
	private final RouteStationService routeStationService
	private final RideSerivce rideSerivce
	private final RouteSerivce routeService
	private final ModelMapper modelMapper
메서드	WetayoQuery(StationService stationService, ModelMapper modelMapper, RouteStationService routeStationService, RideService rideService, RouteSerivce routeService)
	List <StationGraphQLDto> getStation(Double x , Double y ,Double distance)
	List <RouteStationGraphQLDto> getStationAndRoute(Double x , Double y , Double distance)
	Boolean getRide(Integer stationId ,Integer routeId)
	List <RouteDto> getRoutes(String regionName)

WetayoQuery는 모든 Query(get) 을 처리하는 클래스이다. 모든 query를 처리하기에 각 DB와 소통하며 데이터 처리하기 위한 Ride, Route, Station, RouteStation에 대한 service들의 객체와 query에 대한 메서드들을 갖고 있다.

아래 **[표 26]**은 WetayoQuery의 생성자에 대해 기술 한 표이다.

[표 26] WetayoQuery 생성자

형식	WetayoQuery(StationService stationService, ModelMapper modelMapper, RouteStationService routeStationService, RideService rideService, RouteSerivce routeService)
설명	Spring Context가 자동으로 생성하여 사용할 생성자를 선언해준다.

생성자는 각 Entity의 Service들과 객체 간 데이터 Mapping을 위한 객체를 생성자를 통해 의존성 주입을 해준다.

아래 **[표 27]**은 요청 GPS에 근접한 Station 정보를 응답해주는 메서드에 대해 기술한 표이다.

[표 27] WetayoQuery 의 getStation()

형식	*List* <StationGraphQLDto> getStation(Double x , Double y ,Double distance)	
리턴값	정류장 정보가 들어있는 객체 배열	
매개 변수	x	GPS 위도
	y	GPS 경도
	distance	반경 거리
설명	엔드포인트의 getStation query가 들어왔을 때 응답을 해주는 메서드	

GPS 정보와 GeometryUtil의 메서드를 이용하여 기준 GPS로부터 반경 거리 내 위치한 정류장 정보들을 DB에서 조회하여 출력해주는 메서드이다.

아래 **[표 28]**은 요청 GPS에 근접한 Station 정보와 Station에 속한 Route 정보들을 응답해주는 메서드에 대해 기술한 표이다.

[표 28] WetayoQuery 의 getStationAndRoute()

형식	*List* <RouteStationGraphQLDto> getStationAndRoute(Double x , Double y ,Double distance)	
리턴값	정류장 정보와 그에 속한 노선정보가 들어있는 객체 배열	
매개 변수	x	GPS 위도
	y	GPS 경도
	distance	반경 거리
설명	엔드포인트의 getStationAndRoute query가 들어왔을 때 응답을 해주는 메서드	

GPS 정보와 GeometryUtil의 메서드를 이용하여 기준 GPS로부터 반경 거리 내 위치한 정류장 정보와 그에 속한 노선정보들을 출력해주는 메서드이다.

아래 **[표 29]**는 사용자의 탑승희망 여부를 조회하는 메서드에 대해 기술한 표이다.

[표 29] WetayoQuery 의 getStation()

형식	Boolean getRide(Integer stationId,Integer routeId)	
리턴값	Boolean	
매개 변수	stationId	GPS 위도
	routeId	GPS 경도
설명	엔드포인트의 getRide query가 들어왔을 때 응답을 해주는 메서드	

버스 기사가 이용하는 메서드로, 버스 기사의 노선 Id와 알고자 하는 StationId를 가지고 사용자의 탑승희망 여부를 조회하는 메서드이다.

아래 **[표 30]**은 사용자의 탑승희망 여부를 조회하는 메서드에 대해 기술한 표이다.

[표 30] WetayoQuery 의 getStation()

형식	*List*<RouteDto> getRoutes(String regionName)	
리턴값	노선정보가 들어있는 객체 배열	
매개변수	regionName	지역 이름
설명	엔드포인트의 getRoutes query가 들어왔을 때 응답을 해주는 메서드	

버스 기사가 이용하는 메서드로, 버스 기사 앱 로그인 시에 해당하는 지역에 존재하는 모든 버스 노선을 조회하는 메서드이다.

◆ WetayoMutation 클래스

아래 **[표 31]**은 WetayoMutation에 대한 변수, 메서드를 정리한 표이다.

[표 31] WetayoMutation 클래스

이름	class WetayoMutation
기능	wetayo endpoint에 mutation이 들어오면 요청에 맞는 응답을 제공해주는 클래스
변수(의존성)	private final RideService rideService
	private final ModelMapper modelmapper
메서드	WetayoMutation(RideSerivce rideSerivce, ModelMapper modelMapper)
	RideDto createRide(Integer stationId, Integer routeId)
	Boolean deleteRide(Integer stationId, Integer routeId)

WetayoMutation은 모든 Mutation(insert, update,delete)를 처리하는 클래스이다. 사용자의 탑승희망 여부에 대한 mutation을 위해 Ride Service의 객체와 create/delete에 대한 메서드가 정의되어있다.

아래 **[표 32]**는 WetayoMutation의 생성자에 대해 기술 한 표이다.

[표 32] WetayoMutation 생성자

형식	WetayoMutation(RideSerivce rideSerivce, ModelMapper modelMapper)
설명	Spring Context가 자동으로 생성하여 사용할 생성자를 선언해주다.

생성자는 Ride Entity의 Service 객체와 객체 간 데이터 Mapping을 위한 객체를 생성자를 통해 의존성 주입을 해준다.

아래 **[표 33]**은 사용자가 탑승을 희망할 때 그 정보를 DB에 저장하기 위한 메서드에 대해 기술한 표이다.

[표 33] WetayoMutation의 createRide()

형식	RideDto createRide(Integer stationId, Integer routeId)	
리턴값	정류장 정보가 들어있는 객체 배열	
매개 변수	stationId	정류장 id
	routeId	노선 id
설명	엔드포인드의 createRide Mutation에 응답하기 위한 메서드	

사용자가 특정 정류장의 특정 노선을 탑승을 희망하면 DB에 저장하는 메서드이다.

아래 **[표 34]**는 Ride 테이블의 컬럼을 지우기 위한 메서드에 대해 기술한 표이다.

[표 34] WetayoMutation deleteRide()

형식	Boolean deleteRide(Integer stationId, Integer routeId)	
리턴값	Boolean	
매개 변수	stationId	정류장 id
	routeId	노선 id
설명	엔드포인드의 deleteRide Mutation에 응답하기 위한 메서드	

버스 기사 앱에서 사용하는 메서드로 정류장을 지나갔거나, 사용자가 탑승했을 경우 DB에서 컬럼 삭제를 하는 메서드이다.

◆ RideService **클래스**

아래 **[표 35]**는 RideService에 대한 변수, 메서드를 정리한 표이다.

[표 35] RideService 클래스

이름	class RideService
기능	query, Mutation controller에서 Ride 테이블에 접근하여 데이터 이용을 위한 클래스
변수(의존성)	private final RideRepository rideRepository
	private final StationRepository stationRepository
	private final RouteRepository routeRepository
메서드	RideService(RideRepository rideRepository, StationRepository stationRepository, RouteRepository routeRepository)
	Ride findByStationIdAndRouteId(Integer stationId, Integer routeId)
	Ride addRide(Integer stationId, Integer routeId)
	void deleteRide(Integer stationId, Integer routeId)

RideService는 Ride 테이블의 조회, 삽입, 삭제를 위해 메서드가 존재하고 RideRepsitory를 의존성 주입받고 있으며, 요청하는 stationId와 routeId가 존재하는지 유효성 검사를 위해 station과 route Repository를 의존성 주입받고 있다.

아래 **[표 36]**은 WetayoMutation의 생성자에 대해 기술 한 표이다.

[표 36] WetayoMutation 생성자

형식	RideService(RideRepository rideRepository, StationRepository stationRepository, RouteRepository routeRepository)
설명	Spring Context가 자동으로 생성하여 사용할 생성자를 선언해주다.

생성자는 Ride, Route, Station의 조회, 삽입,삭제를 위해 의존성 주입을 생성자를 통해 받는다.

아래 **[표 37]**은 Ride 테이블에 복합키인 routeId와 StationId 조합이 존재하는지 확인하는 메서드에 대해 기술한 표이다.

[표 37] RideService의 findByStationIdAndRouteId()

형식	Ride findByStationIdAndRouteId(Integer stationId, Integer routeId)	
리턴값	조회한 Ride 컬럼에 대한 정보	
매개 변수	stationId	정류장 id
	routeId	노선 id
설명	Ride 테이블의 복합키를 통해 정보를 조회하는 메서드	

삽입 시에 중복 삽입을 막기 위해 테이블에 key가 존재하는지 확인하기 위한 메서드이다.

아래 **[표 38]**은 Ride 테이블에 복합키인 routeId와 StationId 조합으로 새로운 컬럼을 추가하는 메서드에 대해 기술한 표이다.

[표 38] RideService의 addRide()

형식	Ride addRide(Integer stationId, Integer routeId)	
리턴값	삽입 성공한 Ride 테이블 정보	
매개 변수	stationId	정류장 id
	routeId	노선 id
설명	Ride 테이블의 복합키를 통해 정보를 삽입하는 메서드	

사용자가 탑승희망 시에 stationId와 routeId를 가지고 테이블에 탑승희망 여부를 삽입하는 메서드이다.

아래 **[표 39]**는 Ride 테이블에 복합키인 routeId와 StationId 조합으로 컬럼을 삭제하는 메서드에 대해 기술한 표이다.

[표 39] RideService의 deleteRide()

형식	Ride deleteRide(Integer stationId, Integer routeId)	
리턴값	삭제 성공 여부	
매개 변수	stationId	정류장 id
	routeId	노선 id
설명	Ride 테이블의 복합키를 통해 정보를 삭제하는 메서드	

버스 기사 앱에서 사용자가 탑승했거나, 정류장을 지나갔을 시에 stationId와 routeId를 가지고 테이블에 탑승희망 여부를 삭제하는 메서드이다.

◆ RouteService **클래스**

아래 **[표 40]**은 RouteService에 대한 변수, 메서드를 정리한 표이다.

[표 40] RouteService 클래스

이름	class RouteService
기능	query, Mutation controller에서 Route 테이블에 접근하여 데이터 이용을 위한 클래스
변수(의존성)	private final RouteRepository routeRepository
메서드	RouteService(RouteRepository routeRepository)
	List<Route> findByRegionNameLike(String regionName)

RouteService는 Route 테이블의 조회를 위해 메서드가 존재하고 RouteRepository를 생성자를 통해 의존성을 주입받고 있다.

아래 **[표 41]**은 RouteService의 생성자에 대해 기술 한 표이다.

[표 41] RouteService 생성자

형식	RouteService(RouteRepository routeRepository)
설명	Spring Context가 자동으로 생성하여 사용할 생성자를 선언해주다.

생성자는 Route의 조회를 위해 의존성 주입을 생성자를 통해 받는다.

아래 **[표 42]**는 지역 이름으로 노선정보를 조회하는 메서드에 대해 기술한 표이다.

[표 42] RouteService findByRegionNameLike()

형식	List<Route> findByRegionNameLike(String regionName)	
리턴값	노선정보에 대한 객체 배열	
매개변수	regionName	지역 이름
설명	Like 문을 이용해서 해당 지역이 포함하는 노선정보들을 조회한다.	

지역 이름을 가지고 Route 테이블에서 region_name 컬럼을 like 문을 이용하여 조회하는 메서드이다.

◆ RouteStationService **클래스**

아래 **[표 43]**은 RouteStationService에 대한 변수, 메서드를 정리한 표이다.

[표 43] RouteStaionService 클래스

이름	class RouteStationService
기능	query, Mutation controller에서 RouteStation 테이블에 접근하여 데이터 이용을 위한 클래스
변수(의존성)	private final RouteStationRepository routeStationRepository
메서드	RouteStationService(RouteStationRepository routeStationRepository)
	List<RouteStation> findByStationId(Integer id)

RouteStationService는 RouteStation 테이블의 조회를 위해 메서드가 존재하고 RouteStationRepository를 생성자를 통해 의존성을 주입받고 있다.

아래 **[표 44]**는 RouteStationService의 생성자에 대해 기술 한 표이다.

[표 44] RouteStationService 생성자

형식	RouteStationService(RouteStationRepository routeStationRepository)
설명	Spring Context가 자동으로 생성하여 사용할 생성자를 선언해주다.

생성자는 RouteStation의 조회를 위해 의존성 주입을 생성자를 통해 받는다.

아래 **[표 45]**는 key인 StationId로 컬럼을 조회하는 메서드에 대해 기술한 표이다.

[표 45] RouteStationService findByStationId()

형식	List<RouteStation> findByStationId(Integer id)	
리턴값	노선별 정류장 정보에 대한 객체 배열	
매개변수	id	테이블의 key인 stationId
설명	stationId를 가지고 테이블의 컬럼들을 조회하는 메서드	

테이블의 key인 stationId를 가지고 테이블의 컬럼들을 조회하는 메서드이다.

◆ StationService **클래스**

아래 **[표 46]**은 StationService에 대한 변수, 메서드를 정리한 표이다.

[표 46] StaionService 클래스

이름	class StationService
기능	query, Mutation controller에서 Station 테이블에 접근하여 데이터 이용을 위한 클래스
변수(의존성)	private final EntityManager entityManager
메서드	List<Station> getNearByStations(Double gpsX, Double gpsY, Double distance)

RouteStationService는 요청 GPS의 반경 내에 존재하는 정류장들을 조회하기 위한 메서드와 JPA가 아닌 nativeQuery를 위한 EntityMager가 @PersistenceContext를 통해 의존성 주입이 되어있다.

아래 **[표 47]**은 GPS와 반경 거리를 통해 반경 거리 내에 있는 정류장들을 조회하는 메서드이다.

[표 47] StationService getNearByStations()

형식	List<Station> getNearByStations(Double gpsX, Double gpsY, Double distance)	
리턴값	정류장에 대한 정보가 들어있는 객체 배열	
매개변수	gpsX	위도
	gpsY	경도
	distance	반경 거리
설명	GPS 정보와 반경 거리를 가지고 정류장을 조회하는 메서드	

GeometryUtil 클래스를 이용하여 요청 GPS의 반경 거리 내 만큼의 GPS를 구한 후, ‘MBRContains(ST_LINESTRINGRFROMTEXT())‘ 구문을 이용해 해당 GPS 정보가 일치하는 컬럼을 조회하는 메서드이다.

◆ GeometryUtil **클래스**

아래 **[표 48]**은 GeometryUtil에 대한 변수, 메서드를 정리한 표이다.

[표 48] GeometryUtil 클래스

이름	class GeometryUtil
기능	StationService에서 GPS와 반경 거리를 이용해 특정한 값을 계산하는 클래스
메서드	static Location calculateRangeGps(Double x, Double y, Double distance, Double bearing)
	static Integer calculateDistance(double lat1, double lon1, double lat2, double lon2)
	static Double toRaidan(Double coordinate)
	static Double toDegree(Double coordinate)
	static Double sin(Double coordinate)
	static Double cos(Double coordinate)
	static Double normalizeY(Double gpsY)

GeometryUtil은 하버사인 공식을 이용하여 GPS 정보와 반경 거리, 방향(각도)를 가지고 두 점 사이의 거리, 반경 끝점의 GPS 정보를 구하는 메서드가 포함되어있다.

아래 **[표 49]**는 GPS 정보와 반경 거리, 방향을 이용해 특정 방향으로 반경 거리만큼 떨어진 GPS 정보를 구하는 메서드를 기술한 표이다.

[표 49] GeometryUtil의 calculateRangeGps()

형식	static Location calculateRangeGps(Double x, Double y, Double distance, Double bearing)	
리턴값	GPS 정보가 들어있는 객체	
매개변수	baseX	위도
	baseY	경도
	distance	반경 거리
	bearing	방향, 각도
설명	매개변수들을 이용해서 특정 지점부터 특정 방향으로 반경 거리만큼 떨어진 좌표를 구하는 메서드	

bearing을 라디안 표기로 바꿔 하버사인 공식을 이용해서 계산 후 degree 표기로 바꿔 GPS 정보 특정 지점부터 특정 방향으로 반경 거리만큼 떨어진 좌표를 구하는 메서드

아래 [표 50]은 두 GPS 지점 간의 거리를 구하는 메서드에 대해 기술한 표이다.

[표 50] GeometryUtil의 calculateDistance()

형식	static Integer calculateDistance(double lat1, double lon1, double lat2, double lon2)	
리턴값	두 지점 사이의 거리	
매개변수	lat1	첫 번째 지점의 위도
	lon1	첫 번째 지점의 경도
	lat2	두 번째 지점의 위도
	lon2	두 번째 지점의 경도
설명	두 지점 사이의 거리를 구하는 메서드	

구면 코사인 법칙을 이용하여 두 지점 사이의 거리를 구하는 메서드이다.

아래 [표 51]은 각도를 라디안 표기법으로 바꾸는 메서드에 대해 기술한 표이다.

[표 51] GeometryUtil의 toRadian()

형식	Double toRadian(Double coordinate)	
리턴값	특정 변수에 대한 라디안 표기법	
매개변수	coordinate	degree 표기법으로 작성한 각도
설명	degree 표기법을 라디안 표기법으로 바꾸는 메서드	

아래 [표 52]는 각도를 degree 표기법으로 바꾸는 메서드에 대해 기술한 표이다.

[표 52] GeometryUtil의 toDegree()

형식	Double toDegree(Double coordinate)	
리턴값	특정 변수에 대한 degree 표기법	
매개변수	coordinate	라디안 표기법으로 작성한 각도
설명	라디안 표기법을 degree 표기법으로 바꾸는 메서드	

아래 [표 53]은 $\sin\theta$ 의 값을 구하는 메서드에 대해 기술한 표이다.

[표 53] GeometryUtil의 sin()

형식	Double sin(Double coordinate)	
리턴값	특정 변수에 대한 sin 값	
매개변수	coordinate	라디안 표기법으로 작성한 각도
설명	sinθ를 구하는 메서드	

아래 **[표 54]**는 cosθ의 값을 구하는 메서드에 대해 기술한 표이다.

[표 54] GeometryUtil의 cos()

형식	Double cos(Double coordinate)	
리턴값	특정 변수에 대한 cos 값	
매개 변수	coordinate	라디안 표기법으로 작성한 각도
설명	cosθ를 구하는 메서드	

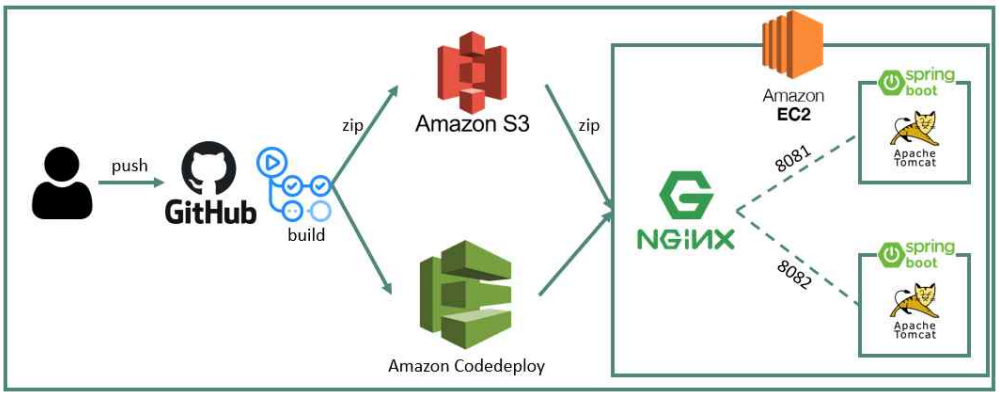
아래 **[표 55]**는 calculateRangeGps메서드의 결괏값을 정규화하는 메서드에 대해 기술한 표이다.

[표 55] GeometryUtil의 nomarlizeY()

형식	Double nomarlizeY(Double gpsY)	
리턴값	정규화한 경도값	
매개 변수	gpsY	경도값
설명	계산된 경도값은 -360도 ~ 360도가 나오나 실제 경도느 -180~ 180이기에 정규화하기 위한 메서드	

◆ 서버 자동 배포 구성도

아래 **<그림 20>**은 서버 자동 배포를 위한 구성도이다.



<그림 20> 서버 자동 배포를 위한 구성도

개발자가 프로그램을 작성 후 github repository에 push를 하게 되면 github action이 이를 감지하여 자동으로 빌드, 압축하고 압축파일을 Amazon S3에 upload 하고 Codedeploy가 S3에서 zip 파일을 받아 압축을 풀고 Appspec.yml 파일의 내용에 맞게 모듈을 실행한다.

총 모듈은 세 개가 실행되며, 현재 구동 중인 서버의 포트가 아닌 다른 포트로 새로운 파일의 서버를 구동시키고, 서버 구동 완료 시에 nginx가 해당 포트로 새로 포트포워딩을 함으로써 무중단 배포를 수행한다.

◆ Appspec

아래 **[표 56]**은 Codedeploy가 실행시킬 구문이 들어있는 모듈을 정리한 파일에 대해 기술한 표이다.

[표 56] Appspec 설정 파일이 실행시키는 모듈들

이름	기능
Run_new_was.sh	현재 실행되고 있는 port 번호 조회 후 다른 port 번호로 새로운 서버(was) 실행
Health_check.sh	새로 실행한 WAS가 실행될 때까지 check 하는 모듈
Switch.sh	Nginx 에 새로 띄운 서버의 포트로 스위칭하는 모듈

Codedeploy가 Appspec.yml에 정의되어있는 특정 모듈들을 실행시키도록 정의하는 파일이다.

Ⅲ.결론

1. 연구 결과
2. 작품 제작 소요재료 목록

참고자료

[1] Flutter 공식 문서-https://flutter.dev/docs

[2] CLOVA Speech Synthesis(CSS)
https://apidocs.ncloud.com/ko/ai-naver/clova_speech_synthesis/

[3] Amazon Elastic Compute Cloud - Linux 인스턴스용 사용 설명서
- AWS

[4] GB302 버스 정보 Open API 서비스 명세서 - 버스 노선 조회 서비스 (SOAP)- 경기도 버스 정보 시스템

[5] GB307 버스정보 OPEN API 명세서 - 버스 위치 정보 조회 서비스 (SOAP) - 경기도 버스 정보 시스템

[6] GB308 버스 정보 Open API 서비스 명세서 - 버스 도착 정보 조회 서비스(SOAP)기도 버스 정보 시스템