<2016-1 창의적 통합설계2 중간보고서>

스마트 워치용 대중교통 App 개발

4조 허정수, 김정환, 연성민

지도교수 : 신영길 교수님

담당자 : 손기성 차장님, 김태완 선임님(삼성전자)

목차

1. 버스
2. 버스 즐겨찾기
3. 지하철
4. 길 안내

1. Abstract

이 프로젝트에서는 스마트 워치용 대중교통 application 개발을 목표로 한다. Target으로 하는 device의 특성을 잘 파악해 사용자의 편리성과 실용성을 최대한 높일 수 있게 만든다. 초기 버전부터 스토어에 등록해 지속적인 업데이트를 해나가는 것을 목표로 한다.

2. Introduction

2.1 프로젝트 주제 설정의 배경

Wearable device가 타 종류의 device에 비해 갖는 강점은 언제 어디서나 빠르고 편리하게 device를 사용할 수 있다는 것이고, 약점은 작은 화면으로 인한 한 번에 전달할 수 있는 정보의 양이 적다는 것이다.

많은 사람들이 매일 대중교통을 이용하고, 대중교통에 관한 정보검색은 정류장, 노선, 시간과 같은 적은 양의 정보만을 포함하기 때문에 스마트 워치와 같은 wearable device에 대중교통 앱은 효과적일 것이라 예상할 수 있다.

실제로 삼성전자의 갤럭시 기어S2는 출시 전부터 많은 사용자들이 티머니나 캐시비와 같은 교통카드 대용 기능을 원했고, 현재 기어S2는 결제 기능을 지원하고 있다. 하지만 대중교통의 정보(정류장, 노선, 도착 예정 시간 등)를 나타내주는 앱은 현재 2개밖에 존재하지 않고, 그마저도 아주 적은 기능만을 가지고 있거나 버그가 있다.

대중교통정보를 검색할 수 있는 기본적인 기능과 함께, wearable device의 특징을 잘 살릴 수 있는 기능까지 포함한 앱이 구현 된다면 그 효과는 아주 클 것으로 예상된다.

2.2 접근방법

Wearable device용 대중교통 앱은 앞서 언급했듯이 활성화 되어있지 않지만, web이나 mobile용 대중교통 앱은 잘 활성화 되어있다. 이미 상용화 되어있는 앱들을 벤치마킹하여 기본적인 기능의 스펙을 상세화 한다.

또한, 기어S2는 자이로 센서, 방향감지 센서, 심박수 측정 센서 등 많은 센서들을 제공하고 있다. 사용자의 편의를 위해 센서를 이용해 할 수 있는 기능들을 고민한다.

대중교통에 대한 정보는 서울시 교통정보과 및 공공 데이터 포탈에서 제공하는 공공API를 이용하여 받아올 수 있도록 한다.

3. Background Study

3.1 관련 접근방법/기술 장단점 분석

Tizen은 소프트웨어 플랫폼으로 Web과 Native application을 지원하는데, 대중교통 앱은 그 특성상 api 서버와 통신해야 하는 일이 많아 편의성을 위해 Web application을 택했다. 하지만 web application으로 작업시 앱의 background에서 widget을 사용할 수 없어 widget은 native application으로 개발 후 file sytstem으로 두 앱을 연동한다.

3.2 프로젝트 개발환경

먼저 모든 개발은 Windows에서 진행된다. 3명이 진행하는 팀 프로젝트인 만큼 협업을 위해 git을 사용한다. 또한, 기어S2는 Tizen OS를 기반으로 하기 때문에 Tizen Wearable SDK 2.3.1과 Tizen IDE 2.4, Tizen Web Simulator 2.4버전을 이용한다.

4. Goal/Problem & Requirements

4.1 프로젝트의 목적

이번 프로젝트의 목표는 스마트 워치에서 작동하는 대중교통용 application을 제작하는 것이다. 실시간 버스나 지하철 도착 정보 제공과 같은 기본적인 기능뿐만 아니라, wearable device의 특징을 잘 살린 기능을 추가해 사용자의 편리함을 극대화 시킨다.

4.2 프로젝트의 요구사항

- 스마트 워치의 특성상 데이터를 직접 device를 통해 전송하는 것은 비효율적이므로 최소한의 input만을 요구하도록 설계한다.

- 마찬가지로 output data의 양이 많으면 화면에 표현하기가 까다롭다. 간략하고 직관적인 ui를 통해 결과를 나타낼 수 있도록 한다.

- 스마트 워치에 내장된 센서와 제공되는 api를 잘 살린 기능을 추가해 편리성 및 실용성을 향상시킨다.

- Target device는 삼성전자의 갤럭시 기어S2이다.

5. Approach

Web이나 mobile에서는 이미 대중교통 앱이 활성화가 되어있기 때문에, wearable에서의 앱이 사용자에게 특별한 편의성을 주지 못한다면 앱 제작의 큰 의미가 없다. 기존에 상용화된 앱을 벤치마킹하여 분석하되, 똑같은 기능이라도 스마트폰보다 스마트 워치를 이용하는 것이 더 편리함을 주도록 앱을 설계한다.

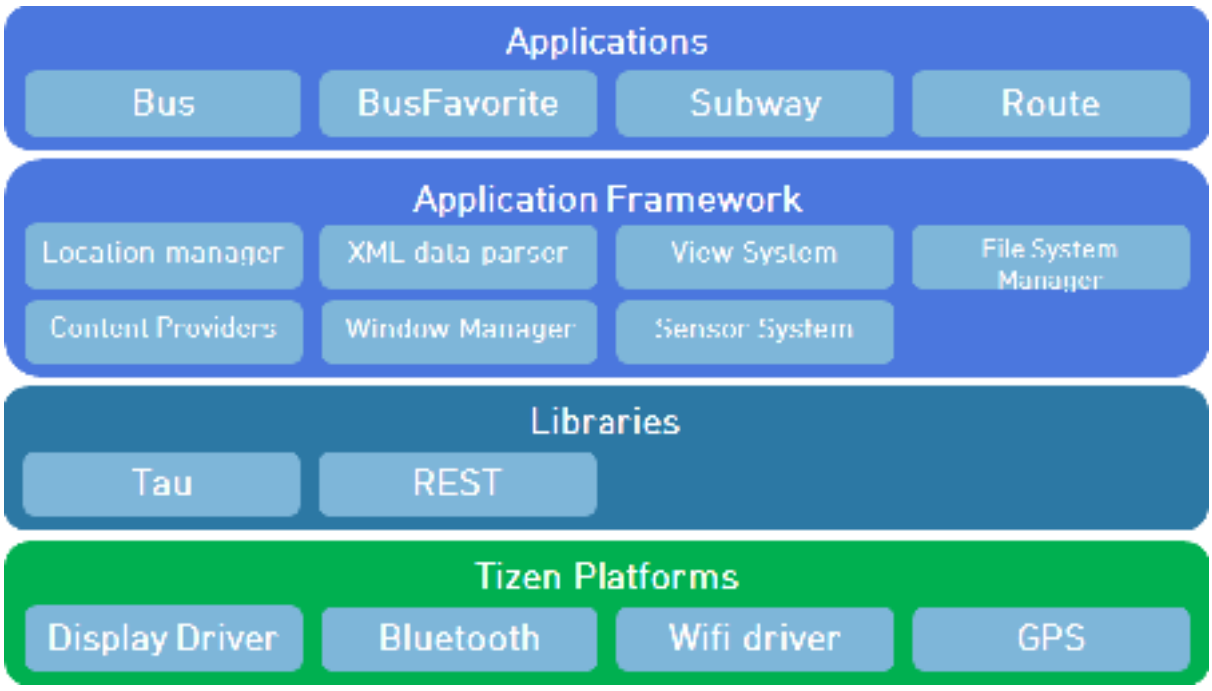
대부분의 대중교통에 관한 정보는 공공 API를 통해 받아올 수 있으므로, html query를 통해 요청하여 xml data를 받아와 필요한 정보만을 추출하여 화면에 띄우도록 한다. UI는 기본적으로 tizen sdk에 내장된 라이브러리를 활용한다.

현재 개발 환경인 tizen의 경우 갤럭시 기어S2뿐만 아닌 여타 wearable기기나 mobile기기 등에서도 쓰이기 때문에 target device가 원형 모양인 경우와 아닌 경우로 나누어 기능을 구현하게 된다. 현 프로젝트에서는 타겟 디바이스를 갤럭시 기어 S2로 한정하여 원형 모양이 아닌 경우는 고려하지 않을 것이다.

또한, tizen에서 제공하는 emulator는 vessel wheel을 사용할 수 없는 등 기능에서 부족한 부분이 많다. 따라서 앱을 test할 때는 가급적이면 remote device manager를 이용해 직접 device에 wgt파일을 올려 시행한다.

6. Project Architecture

6.1 Architecture Diagram



6.2 Architecture Description

어플리케이션에서 필요한 정보를 받기 위해 공공 API에서 데이터를 받아와야 한다. 만약 gear S2가 와이파이와 연결되어 있는 경우 와이파이를 통해서, 만약 연결되어 있지 않다면 휴대폰과 블루투스를 통해서 XML데이터를 받아온다. 이때, 공공 API를 받아오고 쓰기 위해 REST Libraries를 이용한다. 이후, XML data parser를 이용하여 필요한 정보를 추출하여 각각 형식에 맞는 view를 만들어 사용자들에게 보여준다. 조그맣고 동그란 화면의 gear s2에 적절한 화면을 보여주기 위해 Tau library를 이용하며 더 나은 UI를 제공하기 위해 tau library를 수정하여 사용하였다. 각각의 view들에서 자이로 센서나 터치, 베젤을 이용하여 리스트 목록을 볼 수 있으며 어플리케이션의 기능들 중 GPS를 이용하는 기능이 있다. Gear S2의 GPS 기능이 켜져있다면 GPS에서 직접 위치 데이터를 받는다. 그렇지 않다면 모바일의 GPS 위치 데이터를 blue-tooth를 통해 받아와 쓴다. 이 경우, GPS 위치 데이터가 잘 들어왔는지 location manager를 통해 확인하여 GPS 데이터를 받아오지 못하는 경우 적절한 에러 처리를 해준다.

즐겨찾기 기능을 위해서는 웹 어플리케이션과 네이티브 위젯간에 통신이 필요하다. 이 경우 제공해주는 통신 방법이 없기 때문에 File System에 저장하고 거기서 불러오는 방식으로 통신을 진행한다.

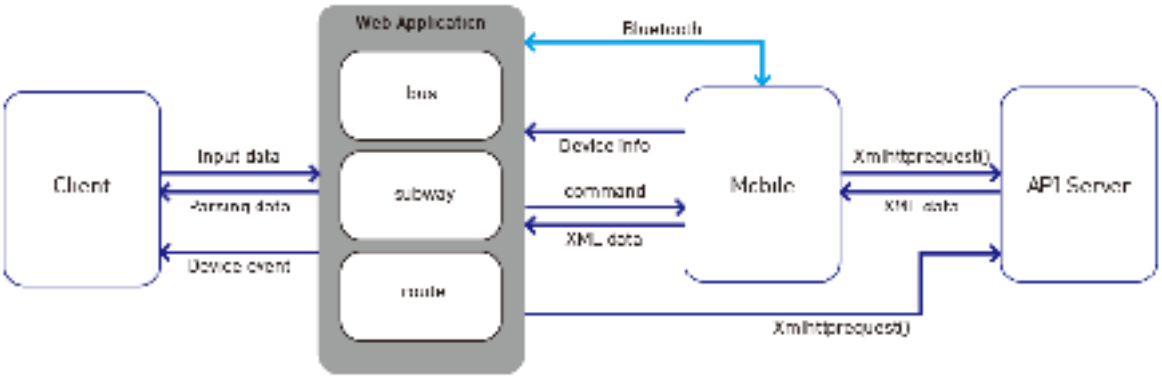
7. Implementation Spec

7.1 Input/Output Interface

이 프로젝트에서 input으로 사용될 수 있는 것은 정류장 id, 노선 번호, 지하철역 등 사용자의 위치와 관련된 data들이다. 버스에서 정류장 id 검색, 노선 번호 검색은 tizen이 기본적으로 제공하는 키패드를 이용해 input을 입력 받는다. 지하철에서 역을 선택하는 것은 주어진 리스트에서 특정 item의 선택을 통해 input을 입력 받는다. 주변 정류장을 찾는 기능은 tizen의 device info api를 통해 GPS 기능을 사용하고, 그 결과값을 input으로 준다.

각각의 input을 목적에 따라 api의 parameter로 넘겨준다. 사용하는 api의 종류는 rest api이고, xml data를 output으로 받는다. 라이브러리로 제공되는 'rest.js' 파일을 이용해 xml data를 파싱하고 사용자가 원하는 데이터만을 추출하여 적절하게 output을 표현해준다.

Data flow diagram은 다음과 같다.



7.2 Modules

이 프로젝트에서는 크게 버스(bus), 지하철(subway), 길 안내(route) 3개의 module이 수평적으로 존재한다고 볼 수 있다. 각각의 module에 대한 설명은 다음과 같다.

A. Bus

1) 정류장 정보 : 특정 정류장에 도착하는 버스의 정보를 확인할 수 있다. 특정 정류장에 도착하는 버스 중 한 버스를 선택하면 2) 노선 정보와 동일한 기능을 수행한다. 정류장은 GPS를 이용해 현재 위치 주변의 정류장을 검색할 수도 있고, 정류장 id를 입력해 직접 조회할 수도 있다.

2) 노선 정보 : 특정 노선의 정보를 확인할 수 있다. 특정 노선이 지나가는 정류장 중 한 정류장을 선택하면 1) 정류장 정보와 동일한 기능을 수행한다.

3) 즐겨찾기 : 자주 이용하는 버스 정류장과 노선을 즐겨 찾기에 등록할 수 있다. 즐겨찾기에 등록된 노선은 앱 background에서 별도의 검색 없이 위젯으로 확인할 수 있다.

B. BusFavorite

A. 3)에서 등록한 즐겨찾는 버스를 어플리케이션의 백그라운드에서도 정보를 확인할 수 있도록 위젯을 생성한다.

C. Subway

1) 역 정보 : 특정 역에 도착하는 열차의 정보를 확인할 수 있다. 지하철역은 GPS를 이용해 현재 위치 주변의 지하철역을 검색할 수도 있고, 리스트에서 지하철역을 직접 찾아 조회할 수도

있다. 조회한 역은 옵션에서 3) 경로 찾기의 출발역이나 도착역으로 지정할 수 있다.

D. Route

현재 위치에서 목적지까지 대중교통을 이용한 경로를 안내한다. 환승에 필요한 정보(지하철역에서 특정 출구로 나갈 것)와 도보 정보(목적지까지의 거리, 방향), 현재 위치와 목적지까지 남은 시간 등을 제공한다.

또한, 현재 타고 있는 버스나 지하철에서 언제 내려야 하는지 알려주는 하차 알림 기능과 승차해야 하는 버스나 지하철이 언제 도착하는지 알려주는 알림 기능도 있다.

8. Solution

8.1 Implementation Details

A. Bus

1) bus.showSurroundingStation(position)

GPS 받아오기에 성공하였을 경우, 1km 반경 내의 주변 정류소를 API를 통하여 읽어오고 createStationList함수를 호출하여 리스트 페이지를 만들어준다.

2) bus.createStationList

주변 버스 정류장 리스트를 만든다. 리스트에는 현재 위치에 대한 거리, 정류장 고유번호를 부가적으로 표시해준다.

3) bus.showBusArrivalTime(arsId)

API를 이용하여 arsId에 해당하는 정류소에 대한 데이터를 받아오고 createArrivalTimeList함수를 호출하여 정류장에 대한 버스 도착 예상시간을 보여준다. 리스트가 다 만들어졌다면 busArrivalTime 페이지를 띄워준다.

4) bus.createArrivalTimeList(data, arsId)

API에서 받아온 data를 파싱하여 도착 예상 시간을 보여주는 리스트를 만든다. 이때, 버스 번호를 클릭하면 bus.busId()함수를 호출하도록 리스트를 만들어준다.

5) bus.busId()

원하는 버스 번호를 입력받으면 그 버스의 routeId를 찾아주는 함수를 호출하여 찾아주고 createBusStationList함수를 호출하여 list를 만들고 busNumberStationList 페이지를 띄워준다.

6) bus.createBusStationList(data)

XML데이터를 파싱하여 정류소 목록을 보여주는 페이지를 만들어준다. 이때, 정류소 목록을 클릭하면 bus.showBusArrivalTime함수를 호출하도록 리스트를 만들어준다.

B. BusFavorite

1) busFavorites.deleteFavorites(stationId)

stationId에 해당하는 목록을 즐겨찾기로부터 제거해준다.

2) busFavorites.showModifyMode(event)

즐겨찾기에서 정류장 클릭시 즐겨찾기 버스 목록을 수정하는 페이지를 띄워준다.

3) busFavorites.createRegisteredStationList(data, init)

즐겨찾기로 등록한 버스 정류장들을 리스트로 만들어준다.

4) busFavorites.showRegisteredStation()

앞서 즐겨찾기로 등록한 버스 정류장 리스트들을 위젯에서 보여준다. 등록된 즐겨찾기 버스 목록이 없다는 "등록하신 즐겨찾기가 없습니다"라고 띄워준다.

5) busFavorites.createBusCheckList(data, mode)

선택한 정류장의 모든 버스 번호를 list에 checkbox와 함께 추가하여 즐겨찾기 등록이 가능한 페이지를 만들어준다.

C. Subway

1) subway.findSurroundingStationsByGps()

Device의 현재 위치를 기준으로 주변의 지하철역을 조회한다. GPS를 이용해 현재 위치를 성공적으로 받아오는 경우에 subway.succeedtoGetGPS(position) 함수를 call하고, 성공적으로 받아오지 못하면 subway.failtoGetGPS(error) 함수를 call한다.

2) subway.succeedtoGetGPS(position)

GPS로 현재 위치를 받아오기에 성공한 경우, 현재 위치를 기준으로 주변의 지하철역을 조회한다. '서울시 좌표기반 근접 지하철역 정보' api를 이용한다. Xml data를 파싱하여 주변 지하철노선과 역명을 리스트로 만들고 #surroundingSubwayStation으로 페이지를 바꿔 띄워준다.

3) subway.failtoGetGPS(error)

GPS로 현재 위치를 받아오기에 실패한 경우, 각 에러에 맞게 에러 메시지를 화면에 toast로 띄워준다.

4) subway.realtimeStationArrival(stationName)

지하철 역을 기준으로 실시간 열차의 도착 정보를 받아온다. '실시간 지하철 도착 정보' api를 이용한다. Xml data를 파싱하여 지하철노선, 방향, 도착 예정 시간을 리스트로 만들고 #arrivalSubwayStation으로 페이지를 바꿔 띄워준다.

D. Route

1) route.showDestination()

설정할 수 있는 목적지를 불러와 보여준다. (현재는 더미 리스트)

2) route.setDestination()

Daum api 를 통하여 목적지의 WGS84 좌표를 읽어온 다음 목적지로 설정한다.

3) route.findway()

현재 위치에서 목적지까지의 대중교통을 이용하여 갈 수 있는 모든 경로를 찾아서 보여준다. 목적지 설정하지 않았을 시, "목적지를 설정해주세요" 라는 메세지를 띄운다.

4) route.getBusRoutes(position)

position 은 현재 위치를 담고 있는 변수, 현재 위치에서 목적지까지 버스만을 이용한 경로를 찾는다.

5) route.getSubwayRoutes(position)

position 은 현재 위치를 담고 있는 변수, 현재 위치에서 목적지까지 지하철만을 이용한 경로를 찾는다.

6) route.getBusSubwayRoutes(position)

position 은 현재 위치를 담고 있는 변수, 현재 위치에서 목적지까지 버스와 지하철 둘 다 이용한 경로를 찾는다.

9. Results

주요 기능에 대한 스크린샷은 위와 같다. 스토어에 등록하는 과정에서 실수가 한 번 있어서 현재는 reject 당한 상태이고, 곧 재등록하여 1주일 내로 스토어에 등록 될 예정이다.

10. Division & Assignment of Work

항목	담당자
Bus 모듈 구현	김정환, 연성민
BusFavortie 모듈 구현	연성민
Subway 모듈 구현	허정수
Route 모듈 구현	김정환, 연성민
UI/UX 개선	김정환, 허정수, 연성민
타이젠 스토어 등록	허정수

11. Conclusion

11.1 구현 과정에서 어려움을 겪은 부분

- [BusFavorite] Tizen 앱은 native와 web application을 지원하는데, 대중교통 앱 특성상 api 서버와 주기적으로 통신해야 하므로 편의성을 위해 web application을 택했다. 그러나 web application은 위젯을 지원하지 않아 결국 native application으로 위젯을 개발했는데, 이 과정에서 tizen이 web app과 native app의 통신을 지원하지 않아 file system을 이용해 데이터를 저장하고 각 app이 접근하는 방식으로 개발하였다. Web app은 html과 javascript로 개발하는데 반해, native app은 C언어로 개발해서 혼란이 있었다.

- [Subway] 서울시에서 제공하는 api 중, 버스와 관련된 데이터는 꽤 신뢰성 있는 데이터를 얻을 수 있는 반면, 지하철과 관련된 데이터는 잘못된 데이터가 너무 많았다. 이에 대한 해결 방법으로 버스는 api를 이용해 실시간 도착정보 및 버스 위치를 받아오게 하고, 지하철은 이미 알려진 시간표 데이터를 이용하여 따로 알고리즘을 구현해 실시간 도착정보 및 현재 위치를 계산하는 방식을 생각했다. 하지만 api가 잘못되었다는 것을 너무 늦게 알아차려 프로젝트 기한 내에 지하철과 관련된 기능들을 수정하기 힘들어 일단 대부분의 기능을 제거한 상태로 스토어에 등록하였다. 후에 새로운 방식으로 다시 구현할 예정이다.

- [Route] 구글 캘린더 api를 통해 목적지를 받아오려고 했으나, web application에서 구글 캘린더 api를 이용하려면 사용자의 서버가 필요해 계획대로 구현하지 못했다.

- [Tizen] 비주류 개발 환경이다 보니 reference 및 example code가 부족해 개발에 많은 어려움이 있었다. 또한, tizen developer 페이지는 로딩이 느려 이용하기 힘들었고 2014년 sdk 1.x 버전의 글이 많아 문제 상황에 대한 해결법을 찾기 힘들었다. 심지어 어플리케이션 등록 실패 후 그 리포트에서 특정 페이지의 reference를 참조하라고 했는데 찾아보니 없는 페이지였다.

- [Sensor] 자이로 센서를 이용해 list에서 위로 넘기는 기능을 하면, 화면이 꺼지는 기본 기능과 겹쳐 제대로 동작하도록 구현하는 것이 힘들다.

11.2 결론

본 프로젝트의 목적은 대중교통 어플리케이션을 만들고 스토어에 올리는 것이라고 할 수 있는데 최종 발표 전까지 스토어에 올리지 못해서 아쉬웠다. 스토어 등록에 처음 해보는 것이라 우리가 생각하는 것보다 거절 당한 이유가 다양하였고 시간이 오래 걸렸다. 회사측 담당자님과 신영길 교수님께서 좋은 의견들을 많이 제시해주셔서 도움이 되었다.

*** [Appendix] User Maunal**

실행환경은 Galaxy Gear S2를 기준으로 한다. 아래와 같은 기능들을 실행하기 위해서는 Galaxy Gear S2가 휴대폰과 블루투스로 연결되어 있고 휴대폰 네트워크가 켜져있어 데이터를 받을 수 있는 상태이거나 Galaxy Gear S2 3G 모델에서 데이터 네트워크를 켜 놓은 상태여야 한다.

1. 버스

A. 노선 번호 검색

노선 번호 검색을 누르면 숫자 키패드가 뜨고 우리가 원하는 버스 번호를 입력하고 엔터키를 누르면 우리가 원하는 버스가 지나는 정류소의 목록을 보여준다. 입력한 숫자와 정확한 노선 번호의 정류소 목록들만 보여준다. (ex. 5516 (O), 7000 (X) 인천 7000번과 같이 정확한 버스 이름이 아니므로) 정류소 목록 중 하나를 선택하면 그 정류소의 버스 목록과 버스 도착 예상 시간을 보여준다.

B. 정류장 검색

정류장 검색은 2가지 기능을 지원한다. 주변 정류장을 찾거나 정류소 ID 숫자를 직접 입력하여 정류장을 검색할 수 있다.

1) 주변정류장 찾기

Galaxy Gear S2의 GPS 기능을 켜놓거나 Gear S2와 휴대폰을 연결하고 휴대폰의 GPS 기능을 켜 놓는다. 그 후, 주변 정류장 찾기 버튼을 누를 시, 현재 위치로부터 1km이내의 주변 정류장들의 목록을 가까운 거리순으로 보여주며 정류장까지의 거리도 표시해준다. 정류장 목록들 중 자신이 선택하고 싶은 정류소를 클릭한다.

2) 정류장 ID 검색

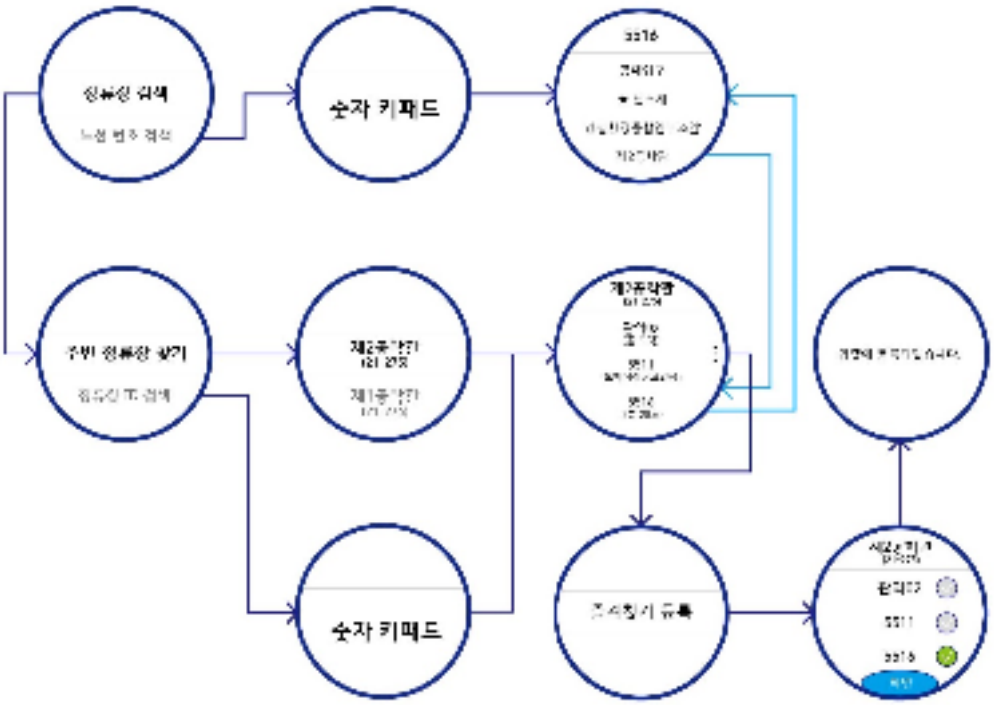
정류장 ID 검색을 누를 시, 숫자 키패드가 뜨고 정류장 번호를 입력할 수 있다. 정류소 번호를 입력하고 엔터키를 누르면 된다.

위와 같은 2가지 방법으로 정류소를 검색하면 정류소를 지나는 버스 목록들을 보여주고 버스들이 정류소까지 각각 남은 시간들을 보여준다. 이 때, 버스 목록들 중 하나를 클릭하면 버스가 지나는 정류소 목록들을 확인할 수 있다.

C. 즐겨찾기 등록

정류소를 지나는 버스 등록 화면에서 오른쪽의 조그만 점3개 버튼을 누르면 즐겨찾기 등록 페이지로 넘어간다. 정류소에서 자신이 원하는 버스 목록들을 체크하고 확인 버튼을 누르면 '위젯에 등록되었습니다' 토스트 팝업이 뜨고 위젯에 버스 목록들이 저장된다.

버스 기능의 전체적인 흐름은 아래의 그림과 같다.



2. 버스 즐겨찾기 Widget

앞서 말한 1.BUS기능에서 즐겨찾기를 등록하지 않으면 위젯 화면에 '즐겨찾기를 등록해주세요'라는 화면이 뜬다. 위젯 하단의 등록 버튼을 누르면 대중교통 알리미 어플리케이션을 실행한다. 즐겨찾기를 등록 후, 위젯을 확인하면 등록한 버스 정류장, 버스들 번호, 노선 방향, 버스 도착 예정 시간이 표시된다. 위젯 화면에서 버스 정류장 이름을 클릭하면 즐겨찾기에 등록한 다른 버스 정류장에 관한 버스 도착 정보가 표시된다. 위젯 화면에서 버스 번호를 클릭하면 현재 정류소에서의 다른 버스의 도착 정보를 표시해준다.

3. 지하철

D. 역 정보*

지하철에서는 역 정보 찾기에서 2가지 기능을 지원한다. 주변 지하철 역을 찾거나 직접 찾을 수 있다. (*현재 공공 API의 기능상의 문제로 이 기능은 지원하지 않습니다.)

1) 주변 지하철 역 찾기

Galaxy Gear S2의 GPS 기능을 켜놓거나 Gear S2와 휴대폰을 연결하고 휴대폰의 GPS 기능을 켜 놓는다. 그 후, 주변 지하철 역 찾기 버튼을 누를 시 자기 주변 1km 이내의 지하철 역 목록을 보여준다. 그 후, 자신이 원하는 지하철 역을 클릭하면 지하철 역의 지하철 도착 예정시간과 종점을 보여준다.

2) 직접 찾기

직접 찾기를 누르면 호선을 선택할 수 있고 호선을 선택하면 호선에 있는 모든 역들을 리스트로 보여준다. 리스트에서 자신이 원하는 지하철역을 누르면 지하철역의 지하철 도착 예정시간과 종점을 보여준다.

지하철 어플리케이션의 전체 흐름은 아래와 같다.