건국대학교 2020년도 1학기
컴퓨터공학과 졸업프로젝트
지하철 Seat크릿!
최종 결과 보고서

물리학과 201610379 김나연

https://github.com/NayaKim

생명과학특성학과 201313250 서지혁

https://github.com/limeburst

컴퓨터공학과 201411255 강예나

https://github.com/kyena421

2020. 06. 18.

Contents

1 개요

지하철 Seat크릿은, QR 코드로 좌석 위치를 입력하고, 공석 발생시 알림을 보내고 받을 수 있는 앱과 서버로 구성된 프로젝트입니다.

지하철의 경우 편리하게 좌석 정보를 입력할 수 있는 방법이 없고, 이에 따라 공유 자원인 지하철 좌석이 비효율적으로 활용되고 있습니다. 따라서 지하철 이용객들이 보다 편리하고 효율적으로 지하철 좌석을 이용할 수 있는 어플리케이션을 제작하고자 하였습니다.

편리성, 과정의 복잡성, 정보력 등이 결여되어 꾸준히 사용되지 않는 기존 어플리케이션 또는 프로젝트와 달리 본 프로젝트는 QR코드를 사용한 직접적 접근으로 편리성을 높이고 공석 여부에 더해 대략적인 대기시간을 제공함으로써 사용성을 높였습니다.

지하철 Seat크릿 프로젝트의 결과물은 모두 GitHub 저장소에 MIT 라이선스 하에 오픈 소스로 공개되어 있습니다.

2 2020. 04. 04. 프로젝트 제안

2.1 개발 목표

본 프로젝트의 목표는, 실시간 좌석 정보를 업데이트하여 사람들이 보다 효율적으로 지하철을 이용할 수 있도록 돕는 어플리케이션을 개발하는 것입니다.

지하철 이용객은 각 좌석 앞 발밑에 부착된 QR코드를 인식하거나, 또는 어플리케이션 내 역 선택을 통해 착석여부와 도착역을 입력하고 해당 역에 도착 시 푸쉬 알림을 받을 수 있습니다. 또, 좌석에 앉지 못하고 대기 중인 승객은 동의 하에 공석이 생긴 경우 푸쉬 알림을 받을 수 있습니다.

앱을 통해 각 좌석에 앉은 승객이 하차할 때까지 얼마나 남았는지를 색깔로 알 수 있습니다. 이때 실제 남은 역 수 정보는 서버에게만 제공되어 이용객의 프라이버시를 보호합니다. 애플리케이션은 입력 받은 이용객의 여정 정보를 서버에 등록합니다.

이 프로젝트의 최종 산출물로는, 안드로이드 어플리케이션, 좌석별 정보를 담고 있는 QR코드, 그리고 지하철 현황, 앱 푸쉬, 사용자 여정 관리 등을 위한 데이터베이스 서버가 있습니다.

3 2020. 04. 18. 요구사항 분석

3.1 QR 코드 기능 명세

3.1.1 QR 코드 위치 명세

QR 코드는 지하철 좌석별로 앞 바닥에 부착되어 있다. 좌석에 앉아 있는 승객과 서서 기다리는 승객 모두 이용할 수 있다.

3.1.2 QR 코드 내용 명세

QR 코드에는 앱 ID와 QR 코드 ID가 기록되어 있다. 서버에 미리 등록된 ID로 좌석정보를 확인할 수 있다.

ex) `seatspotter://31c40b62-9c0d-4bba-a3a1-638cacaf4847`

3.1.3 QR 코드 인식 명세

어플 내부에 QR코드를 인식할 수 있는 단추가 있다.

TC1-1: 스마트폰에 해당 어플이 설치 후, 해당 어플 또는 타 어플을 이용해 지하철 좌석 앞 바닥에 부착되어 있는 QR코드를 스캔한다. 해당 어플을 사용한 경우 알맞은 열차정보와 출발역이 등록된다. 타 어플을 사용한 경우 해당 어플로 연결된 후 알맞은 열차정보와 출발역이 등록된다. (관련 명세: 1.1., 1.2., 1.3. 2.2.1., 3.1.1. 3.1.2.)

3.2 앱 기능 명세

지하철 경로검색, 실시간 열차 별 좌석정보를 제공한다.

3.2.1 메인화면 명세

- 2.1.1. 노선도 동작 명세: 노선도는 줌인, 줌아웃, 드래그를 통해서 확인 가능하다.
- 2.1.2. 노선도 역 명세: 노선도의 각 역을 클릭해서 출발역 또는 도착역을 설정할 수 있다.
- 2.1.3. 검색기능 명세: 역명을 검색해서 출발역 또는 도착역을 설정할 수 있다.

TC2.1-1: 어플을 실행 후 메인 화면에서 줌인, 줌아웃, 드래그를 통해 노선도를 확인한다. 수행 결과, 줌인 시 노선도가 확대되고 줌아웃 시 축소, 드래그 시 해당 방향으로 이동하여 노선도를 확인할 수 있다. (관련 명세: 2.1.1.)

TC2.1-2: 어플을 실행 후 메인 화면의 노선도에서 '건국대학교 입구' 역을 클릭한다. 수행 결과, 출발역/도착역 지정 버튼이 표시되어 해당 버튼을 클릭하면 출발역/도착역으로 설정할 수 있다. (관련 명세: 2.1.2., 2.2.2.)

TC2.1-3: 어플을 실행 후 메인 화면 노선도 상단의 '역 검색' 단추를 클릭한다. 검색 입력 박스에 '건국대학교 입구' 역을 입력한다. 검색 입력 박스 우측의 '검색' 버튼을 클릭한다. 화면에 표시되는 '건국대학교 입구' 역을 클릭한다. 수행 결과, 출발역/도착역 지정 버튼이 표시되어 해당 버튼을 클릭하면 출발역/도착역으로 설정할 수 있다. (관련 명세: 2.1.3., 2.2.2.)

3.2.2 여정정보 등록 화면 명세

QR코드 인식 또는 직접 역 정보를 등록하여 도착정보를 조회할 수 있다.

- 2.2.1. QR코드 인식 명세: QR코드를 처음 인식하면 해당 QR코드의 열차정보를 불러와 출발역에 자동 입력된다.
- 2.2.2. 목적지 명세: 이용객은 목적지인 도착역을 검색하거나 노선도에서 클릭하는 방법으로 찾아 등록할 수 있다.
- 2.2.3. 이용객 종류 명세: 여정정보가 등록된 후 좌석에 앉은 승객인지 서서 대기중인 승객인지를 묻는 알림창이 뜬다.
- 2.2.4. 여정정보 명세: 경로 설정을 마친 이용객에게 도착예정시간과 도착역까지 남은 시간을 보여준다.
- 2.2.5. 좌석선택 명세: QR코드를 인식하지 않고 직접 여정정보를 등록한 사용자가 착석승객일 경우, 좌석 선택 창에서 착석한 좌석을 선택할 수 있다. 해당 과정을 스킵하는 경우 2.2.6.의 알림창에서 미동의 사용자로 간주한다.
- 2.2.6. 좌석정보 공유 명세: 경로 설정을 끝낸 이용객이 착석 이용객 일 경우 본인의 좌석정보를 앱 서버에 공유할지 묻는 알림창이 뜬다. 이 때 정보이용에 대한 동의를 위해 "해당 정보는 '분 이내'와 같이 대략적인 정보로 표현되며 어플 내의 좌석 정보 공유 이외의 목적 외에는 사용되지 않음을 알립니다"와 같은 메시지를 표시하여야 한다.
- 2.2.7. 좌석 정보 조회 명세: 2.2.3.의 알림창에서의 선택과 상관없이 승객이 이용중인 열차 칸에 대한 좌석정보가 표시된 창을 띄운다.
- 2.2.8. 좌석 정보 내용 명세: 좌석 정보는 도착까지 남은 시간에 따라 색깔 별로 표시된다. 테이블은 다음과 같다. [5분 이내-연두색/10분 이내-시안색/15분 이내-파란색/20분 이내-보라색/30분 이내-분홍색/30분 이상-빨간색/정보없음-회색/비어있음-노란색/사용중-검정색]. 드래그 하여 열차 칸을 확인할 수 있으며, 색을 구분하기 힘든 이용객을 위해 각 좌석을 클릭하면 남은 시간을 빈 공간에 표시하여 알 수 있게 한다.

- 2.2.9. 좌석 정보 신고 명세: 비어있는 좌석인데 잔여시간이 남아 있는 등 표시된 정보에 오류가 있을 경우 이용객은 '신고하기' 단추를 통해 해당 좌석을 서버에 신고할 수 있다. 신고하기 단추를 누르면 오류목록 리스트가 제공되고 이용객은 그 중 하나를 선택할 수 있다. 오류목록 리스트 TBD.
- 2.2.10. 여정 푸쉬 명세: 착석 중인 승객은 어플 종료 후에도 push알림을 통해 도착지까지 소요시간과 도착예정을 확인 할 수 있으며, push알림 내에서 [지금 하차/알림 종료]를 선택할 수 있다.
- 2.2.11. 공석 푸쉬 명세: 서서 대기 중인 승객은 공석이 생길 시 push알림을 받을 수 있다. 알림은 설정을 통해 커스텀할 수 있다.

TC2.2-1-1: 어플 실행 후 출발역으로 '건대입구'를, 도착역으로 '뚝섬유원지'를 입력한다. 수행 결과, 지하철 시간표에 따라 알맞은 열차정보가 조회되는지 확인한다. (관련 명세: 3.1.1.)

TC2.2-1-2: 착석승객으로 등록 후 열차칸 1-1, 열차방향의 오른편 가장 앞 좌석으로 좌석을 선택한다. 수행 결과, 도착예정시간과 소요시간이 지하철 시간표에 따라 맞게 나오는지 확인한다. (관련 명세: 2.2.3., 2.2.4., 2.2.5., 3.2.1.)

TC2.2-1-3: 좌석정보 공유를 누른다. 어플을 종료한다. 푸쉬알림의 '지금하차'를 누른다. 수행 결과, 입력한 좌석칸이 연두색(5분 이내)로 표시되는지 확인한다. 푸쉬 알림에 도착정보가 정확히 표시되고 예상시간이 현재 시간에 맞춰 줄어드는 것을 확인한다. '지금하차'를 누른후 해당좌석이 공석으로 표시되는지 확인한다. (관련 명세: 2.2.6., 2.2.7., 2.2.8., 2.2.10.)

TC2.2-2-1: 위 테스트케이스의 과정과 동시에 수행합니다. 어플 실행 후 현재 시간 기준으로 '건대입구'역을 지나가며 1번칸에 등록된 열차정보가 담긴 QR코드 인식한다. 도착역으로 '시청'역을 입력한다. 대기승객으로 등록 한다. 수행 결과, 출발역으로 '건대입구'가 나오는지 확인한다. 도착예정시간과 소요시간이 지하철 시간표에 따라 맞게 나오는지 확인한다. (관련 명세: 3.1.1., 3.1.2., 2.2.1., 2.2.2., 2.2.3., 2.2.4., 3.2.1.)

TC2.2-2-2: 1-1번칸의 열차방향의 오른편 가장 앞 좌석이 공석이 될 때까지 기다린다. 해당 좌석 클릭 후 '신고하기' 버튼을 누른다. '공석 표시되어 있는데 사용중이에요'를 선택한다. 수행 결과, 공석 푸쉬알림이 오는지 확인한다. 6번과정 이후 해당 좌석이 검정색으로 표시되는지 확인한다. (관련 명세: 2.2.7., 2.2.8., 2.2.9., 2.2.11.)

3.2.3 설정화면 명세

이용객은 메인화면에서 설정단추를 통해 알림, 표시정보에 대한 내용을 설정할 수 있다.

- **2.3.1. 여정 푸쉬 설정 명세: 착석 중인 이용객이 여정 푸쉬 알림을 받는 것을 [받음/받지않음] 설정할 수 있다.
- 2.3.2. 공석 푸쉬 설정 명세: 좌석 대기 중인 이용객이 공석 알림을 받는 것을 [받음/받지않음] 설정할 수 있다.
- 좌석정보 비활성화 설정 명세: '좌석정보 비활성화'로 좌석 정보를 조회할 수 없고, 공유 알림창을 받지 않을 수 있다. 좌석정보 공유 알림창을 띄우지 않고 '미동의'를 디폴트 값으로 지정한다.

TC2.3-1: 어플 실행 후 설정에서 여정 푸쉬알림을 '받지않음' 설정한다. 테스트케이스 TC2.1-1의 4번 과정까지 진행한다. 수행 결과, 푸쉬알림이 오지 않는 것을 확인한다. (관련 명세: 2.3.1.)

TC2.3-2: 어플 실행 후 설정에서 공석 푸쉬알림을 '받지않음' 설정한다. 테스트케이스 TC2.1-2의 4번 과정까지 진행한다. 수행 결과, 푸쉬알림이 오지 않는 것을 확인한다. (관련 명세: 2.3.2.)

TC2.3-3: 어플 실행 후 설정에서 '좌석정보 비활성화' 설정한다. 테스트케이스 TC2.1-1의 3번 과정까지 진행한다. 수행 결과, 좌석 공유 알림창이 뜨지 않는 것을 확인한다. (관련 명세: 2.3.3.)

3.3 서버 기능 명세

3.3.1 열차정보 로딩 명세

QR코드 인식 후 지하철 도착정보 API와 연계하여 알맞은 정보를 전송한다.

- 3.1.1. 열차정보 명세: QR 코드 인식 후 QR코드에 해당하는 열차정보를 불러올 수 있다. QR 코드 ID는 해당 QR 코드가 부착되어 있는 열차에 대한 매핑을 가진다.
  ex) `31c40b62-9c0d-4bba-a3a1-638cacaf4847`: 2호선 3000번 열차 1-4칸 1번 좌석. 직접 여정정보를 등록한 경우 지하철 시간표 API와 연계하여 정확한 열차정보를 불러올 수 있다.
- 3.1.2. 출발지 불러오기 명세: QR 코드를 인식한 사용자의 열차정보를 얻어낸 후 지하철 도착정보 API와 연계하여 정확한 출발지를 불러올 수 있다.

3.3.2 도착정보 등록 명세

사용자가 입력한 여정정보를 알맞은 형태로 저장한다.

- 3.2.1. 출력 명세: 사용자가 여정정보를 등록하는 과정에서 도착역과 착석 여부를 입력하면, 지하철 도착정보 API와 연계하여 도착 예정 시간과 예상 소요시간을 전송할 수 있다.
- 3.2.2. 저장 명세: 사용자의 스마트폰 정보를 등록 후(미동의시X) 좌석 대기중인 승객일시 도착역과 기기정보를 저장한다. 착석 승객일 시 좌석별로 도착역과 테이블에 따라 분류된 예상소요시간, 기기정보를 저장한다.

3.3.3 좌석정보 조회 명세

테이블에 따라 열차 칸별 좌석정보를 전송한다.

- 3.3.1. 출력 명세: 사용자의 좌석정보 공유여부를 수집한다. 동의여부에 관계없이 해당 열차칸의 좌석정보를 테이블에 따라 분류된 값을 전송한다. 동의하지 않은 사용자의 좌석은 '정보없음'으로 분류된다.
- 3.3.2. 신고 명세: 사용자로부터 좌석정보와 다른 내용 신고를 받고 바른 정보로 저장할 수 있다.

3.3.4 푸쉬 알림 명세

착석 승객과 대기 승객에게 알맞은 푸쉬알림을 전송한다.

- 3.4.1 푸쉬 기기 등록: 푸쉬알림을 보내기 위한 기기정보를 등록한다.
- 3.4.2. 여정 푸쉬 명세: 좌석에 정보가 등록된 사용자에게 도착 예상 소요시간을 푸쉬알림으로 보낸다. 사용자가 '지금하차' 버튼을 누르면 해당 좌석을 공석으로 표시한다.
- 3.4.3. 공석 푸쉬 명세: 대기중인 사용자가 푸쉬알림에 동의했다면 해당 열차칸에 공석 발생 시 푸쉬알림을 보낸다.

3.4 비기능 명세서

3.4.1 사생활 요구사항

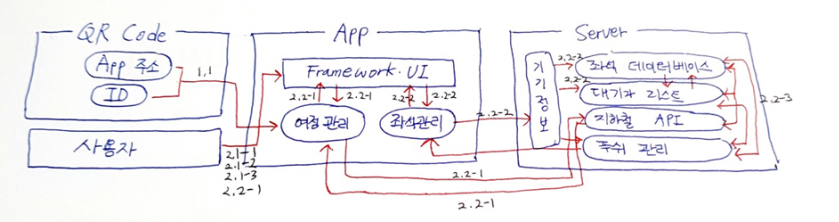
1.1. 착석 사용자의 사생활: 앱의 최종 사용자는 임의의 착석한 사용자의 구체적인 도착정보나 현재 위치정보를 알 수 없다.

3.4.2 배포 요구사항

2.1. 어플리케이션 배포: Google 플레이스토어에 어플리케이션 등록 후, 온라인 배포한다. 개발 사이트에서 온라인 배포한다.

2.2. QR 코드 배포: QR 코드 이미지는 가로 10cm, 세로 10cm 크기로 출력하여 지하철 바닥에 좌석 앞쪽 마다 부착한다.

4 2020. 04. 25. 디자인



지하철 Seat크릿 프로젝트을 구성하는 모듈들은, 크게 사용자, QR 코드, 모바일 앱, 그리고 서버로 나뉘어 집니다. 다이어그램에는 표현되지 않았지만, 서울시의 지하철 실시간 위치정보를 제공하는 서버 컴포넌트도 포함할 수 있을 것 같습니다.

상위 디자인의 인터페이스 설계는 다음과 같이 진행하였습니다. 저희 시스템 아키텍쳐에서는 QR코드도 하나의 컴포넌트로 정의되어 있습니다. 하지만 이번 디자인 단계에서는 QR 코드 등의 하드웨어적인 구성 요소나, QR 코드가 설치되는 지하철 열차칸 등의 환경적인 요소는 고려하지 않았습니다. 따라서 현재로서는 소프트웨어적인 구성 요소에 대한 디자인 고려 사항만이 정의되어 있습니다.

4.1 상위 디자인 – 인터페이스 설계

4.1.1 모바일 애플리케이션

요구사항 번호	설명
2.1.2.	메인 화면 노선도 역 선택
2.1.3.	메인 화면 역명 검색
2.2.1.	여정 화면 QR 코드 인식
2.2.2.	여정 화면 목적지 역 선택
2.2.3.	여정 화면 이용객 종류 선택
2.2.4.	여정 화면 도착 정보 표시

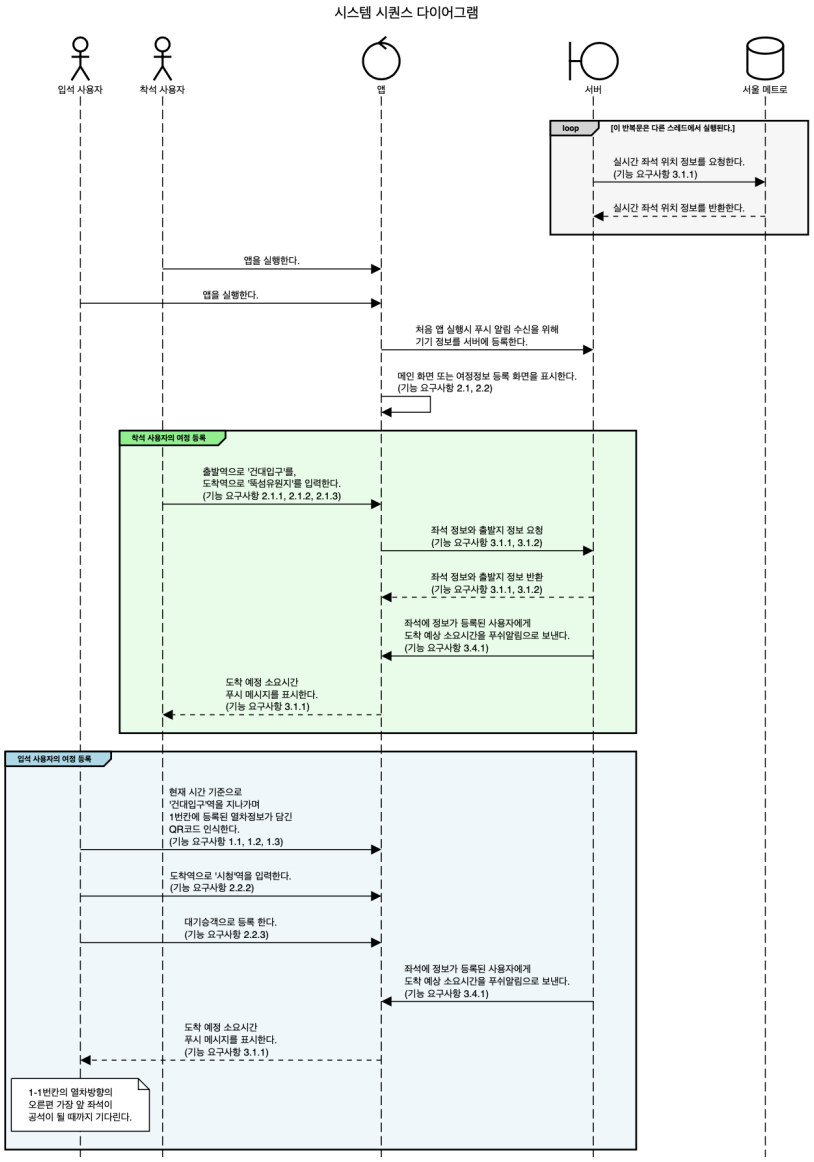
먼저, 첫 번째 주요 모듈인 모바일 애플리케이션의 인터페이스 설계는, 앱에서 제공하는 GUI 구성 요소들을 인터페이스 항목으로 표현하였습니다.

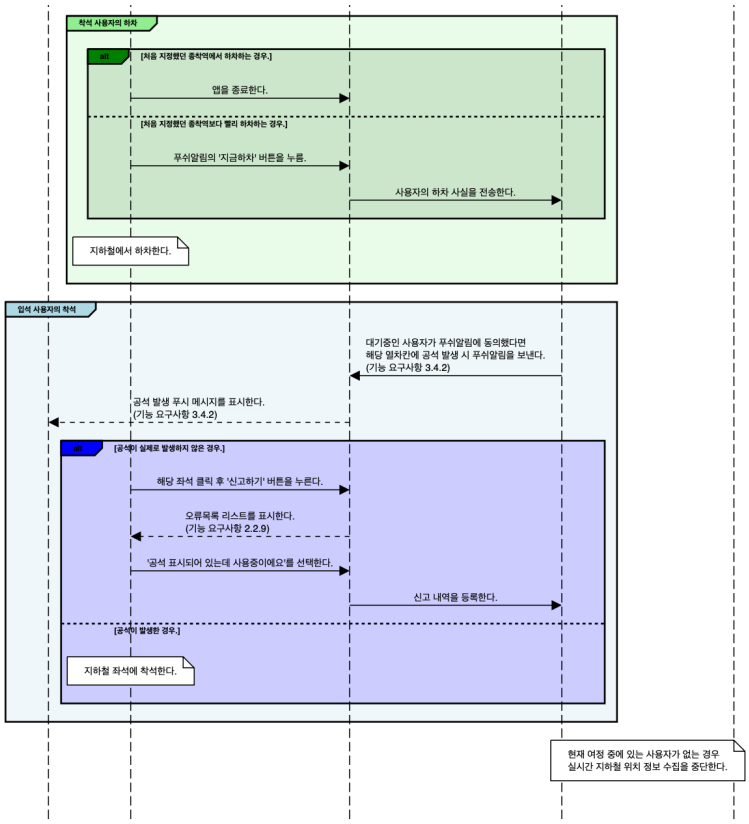
4.1.2 서버

요구사항 번호	HTTP 메서드 및 URL	설명
3.4.1.	POST /devices	푸시 알림 수신 장치 정보 등록
3.2.1., 3.2.2.	POST /itineraries	앱 사용자 여정 정보 저장
3.1., 3.3.1.	GET /seats/{seat_id}	좌석 실시간 위치 정보를 제공
3.3.1.	GET /seats/all	전체 좌석 위치 정보 제공
3.3.2.	POST /feedback	사용자 신고 내역 저장

두 번째 주요 모듈인 서버는 최종 사용자가 조작하는 모바일 애플리케이션으로부터 전송되는, 동기적(synchronous)인 요청을 처리하기 위한 HTTP REST API 형태를 가집니다. 서버의 인터페이스 정의는 HTTP 메서드와 엔드포인트 URL의 목록으로 표현하였습니다.

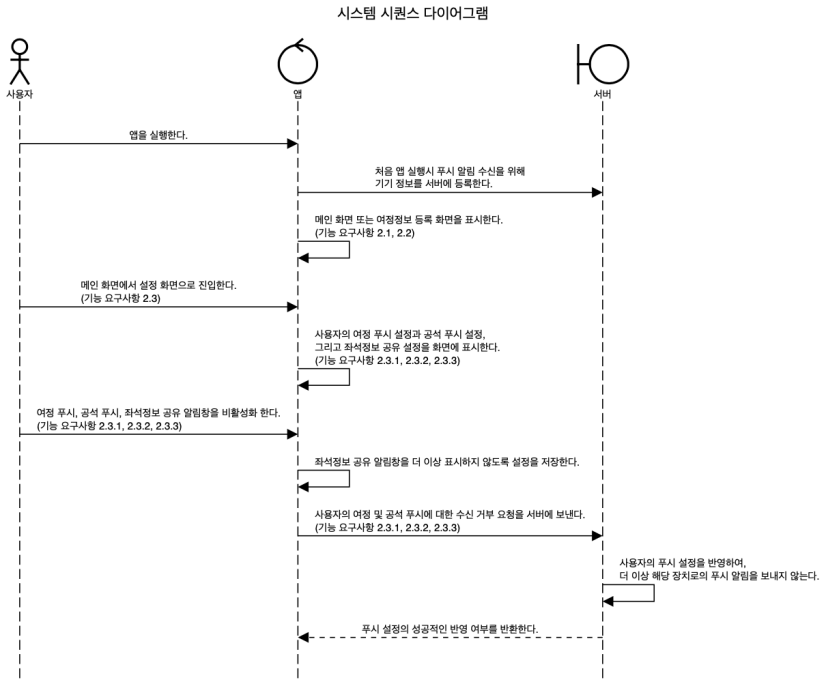
4.2 상위 디자인 – 시스템 시나리오 분석 1





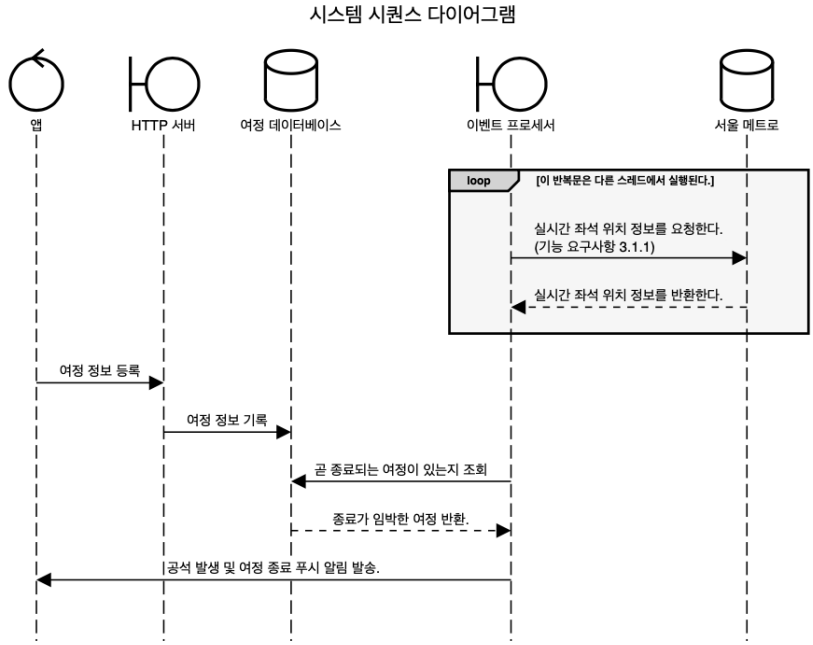
다음은 저희 프로젝트의 핵심적인 시스템 시나리오를 분석하여 시퀀스 다이어그램으로 나타낸 것입니다. 저희 시스템의 프로세스를 시작하는 주요 액터, 즉 행위자는 최종 사용자인 지하철 입석 승객과 착석 승객이 됩니다. 이 두 사용자가 지하철에 탑승한 시점부터 여정을 종료하는 시점까지, 저희 시스템 내에서 어떤 상호작용이 일어나는지 표현하였습니다.

4.3 상위 디자인 – 시스템 시나리오 분석 2



다음은 안드로이드 애플리케이션에서, 사용자가 환경 설정 메뉴를 조작하는 시나리오를, 시퀀스 다이어그램으로 표현한 것입니다. 노선도가 표시되어 있는 메인 화면에서, 환경 설정으로 진입하기 위한 인터페이스와, 푸시 수신 여부 등에 대한 토글 버튼 인터페이스 등을 나타내었습니다.서버 측에서 보내는 푸시 메시지에 대한 수신 여부는 서버 측에 전달을 하고, 좌석 정보 공유 다이얼로그를 표시할지에 대한 여부는 로컬 저장소에 기록한다는 차이가 있습니다.

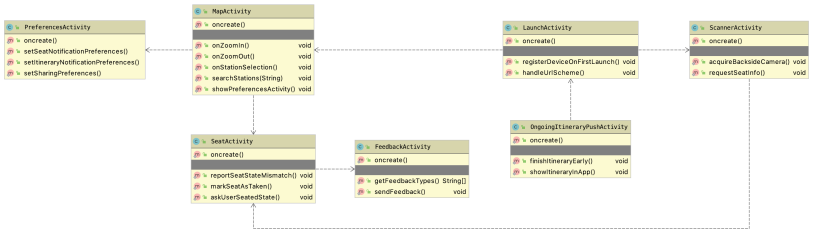
4.4 상위 디자인 – 시스템 시나리오 분석 3



다음은, 아키텍처 설계 다이어그램에서 '푸쉬 관리' 라는 이름으로, 서버의 하위 모듈로 표현된 컴포넌트를 설명합니다. 이벤트 프로세서로 불리는 이 컴포넌트가, 어떻게 여정 정보를 조회하고, 최종적으로 사용자의 앱에 푸시 알림을 발송하는지 표현한 시퀀스 다이어그램입니다.

그림에서 볼 수 있듯이, 푸시 메시지 발송을 관리하는 이벤트 프로세서는 서버에 속해 있습니다. 하지만, 이벤트 프로세서의 실행 라이프사이클은, 앱으로부터의 요청을 동기적, synchronous 하게 처리하는 HTTP 서버와는 독립적으로 존재할 필요가 있습니다.

4.5 상세 디자인 – 안드로이드 애플리케이션

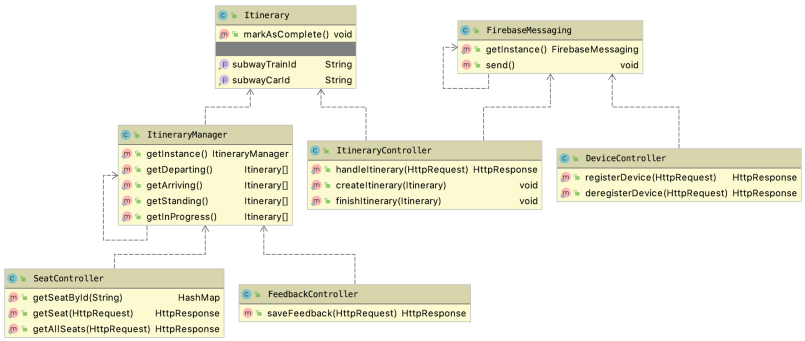


안드로이드 애플리케이션의 클래스 다이어그램을 첨부하였습니다. 이 다이어그램은 대략적인 메서드 프로토타입만 선언되어 있는 스켈레톤 코드로부터 생성되었습니다. 추후 보여드릴 서버 측 클래스 다이어그램도 같은 방식으로 생성하였습니다.

안드로이드 앱은 사용자 장치의 디스플레이에 표시되는 씬, 즉, 안드로이드 도메인 스피시픽 언어로 액티비티라고 불리는 단위로 설계하였습니다. 액티비티는 각각의 뚜렷한 표현 방식을 가지는 화면 별로 분리가 되며, 안드로이드 앱의 생명주기와 밀접하게 연결됩니다.

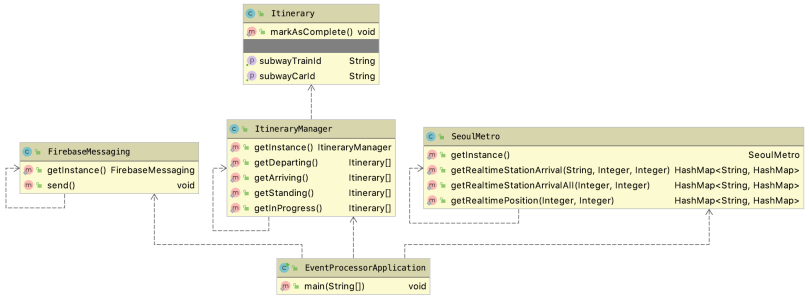
좌석을 선택하거나 여정 정보를 등록하기 위해 사용되는 GUI 엘리먼트와, 장치를 푸시 서버에 등록하는 등 앱 내부적으로만 사용되는 인터페이스의 구현을 함께 표현하였습니다.

4.6 상세 디자인 – HTTP REST API 서버



HTTP REST API 서버의 상세 디자인입니다. MVC 패턴에 맞춰, API 엔드포인트가 제공하는 리소스의 성격에 따라 SeatController, FeedbackController, DeviceController 등으로 분리되어 있습니다. 각 컨트롤러에는 HTTP 엔드포인트를 제공하며, 요청을 처리하고 응답을 반환하기 위한 핸들러 메소드가 정의되어 있습니다.

4.7 상세 디자인 -— 이벤트 프로세서 (가칭)



서버 측 이벤트 프로세서의 클래스 다이어그램입니다. 직전에 설명드린 HTTP REST API 서버가, 모바일 애플리케이션으로부터 전송되는, 동기적(synchronous)인 요청을 처리하기 위한 모듈이라면, 이벤트 프로세서는 최종 사용자 간의 상호작용과 관계 없이 비동기적(asynchronous)으로 작동하는 모듈입니다.

이벤트 프로세서는 서울메트로 API를 통해 주기적으로 통신하며 실시간 지하철 위치정보를 받아오고, 상황에 따라 모바일 애플리케이션 측으로 푸시 메시지를 발송합니다.

4.8 추적성 분석표

마지막으로, 저희 프로젝트의 추적성 분석표를 첨부하였습니다.

5 2020. 06. 18. 구현 결과물

5.1 지하철 Seat크릿 QR 코드

지하철 Seat크릿 QR 코드[2]는 지하철 좌석 밑에 부착되는 형태의 산출물입니다. QR 코드에는 안드로이드 App Link[4]를 지원하는 형식의 URL이 저장되어 있으며, 서비스의 사용자가 안드로이드 기기의 카메라를 통해 스캔하여 앱으로 연결되는 방식으로 활용됩니다.

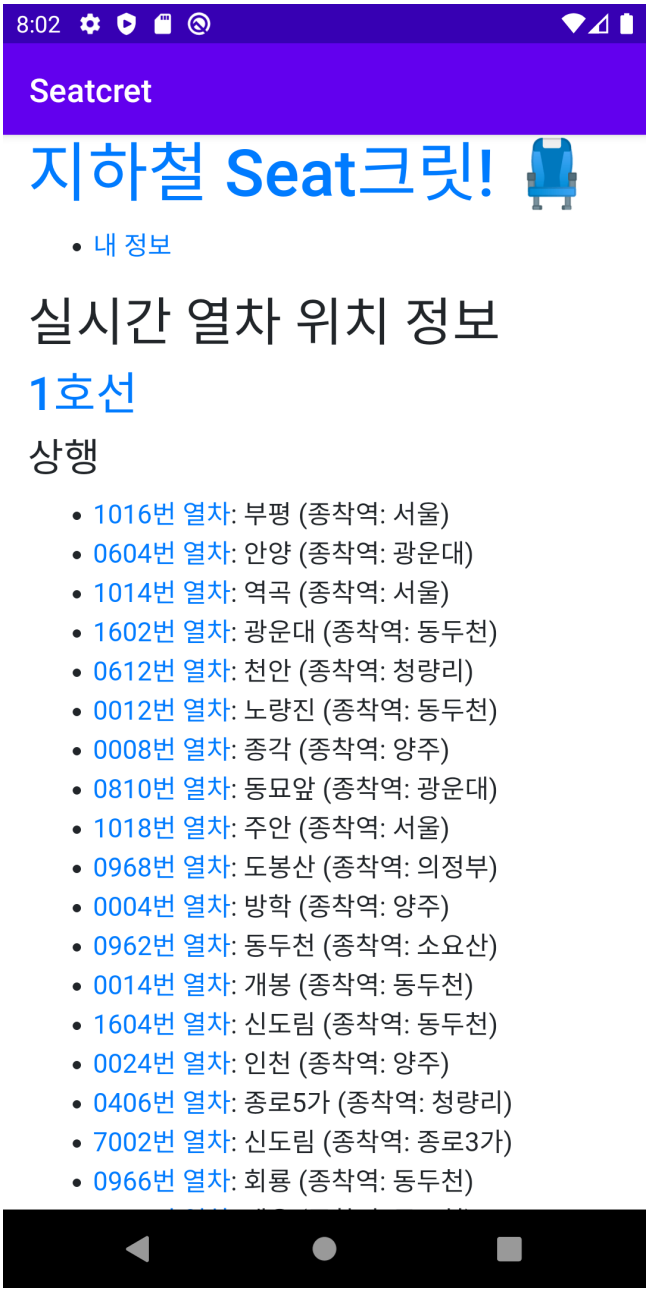
5.2 지하철 Seat크릿 서버

지하철 Seat크릿 서버는 파이썬 프로그래밍 언어[1]로 구현하였으며, GitHub에 MIT 라이선스[6] 하에 오픈 소스로 공개되어 있습니다[8]. 안드로이드 Firebase Cloud Messaging[10] 푸시 토큰, 알림 설정, 여정 정보 등의 회원 정보와, 열차 위치 정보 등의 공용 정보의 저장과 조회를 위한 영속화 계층으로 Redis[7]를 사용했습니다. 지하철 Seat크릿 서버는 보고서 작성 시점을 기준으로, `https://seatcret.ji.hyeok.org` 주소 하에 배포되어 있습니다.

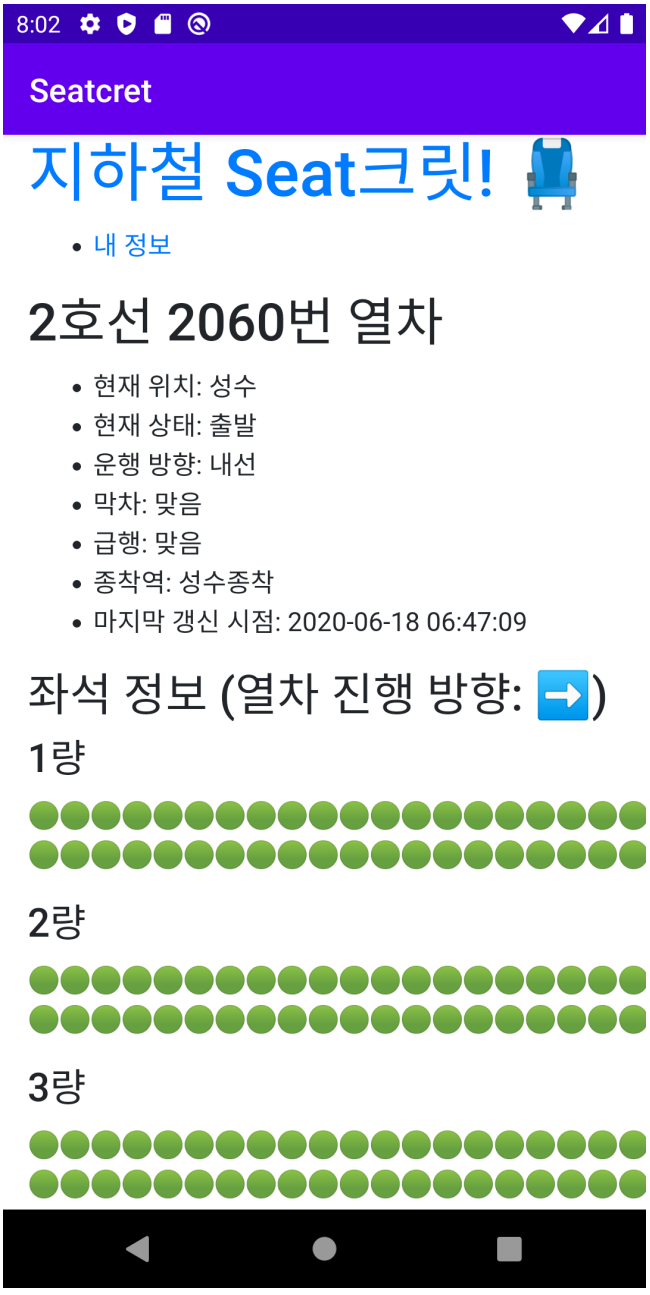
5.3 지하철 Seat크릿 안드로이드 앱

지하철 Seat크릿 안드로이드 앱은 코틀린 프로그래밍 언어[3]로 구현하였으며, GitHub에 MIT 라이선스 하에 오픈 소스로 공개되어 있습니다[5]. 안드로이드 앱은 지하철 Seat크릿 서버에 대한 WebView[9]와 Firebase Cloud Messaging 토큰 발급 및 등록 코드로 이루어져 있습니다.

5.3.1 안드로이드 앱 – 메인 화면

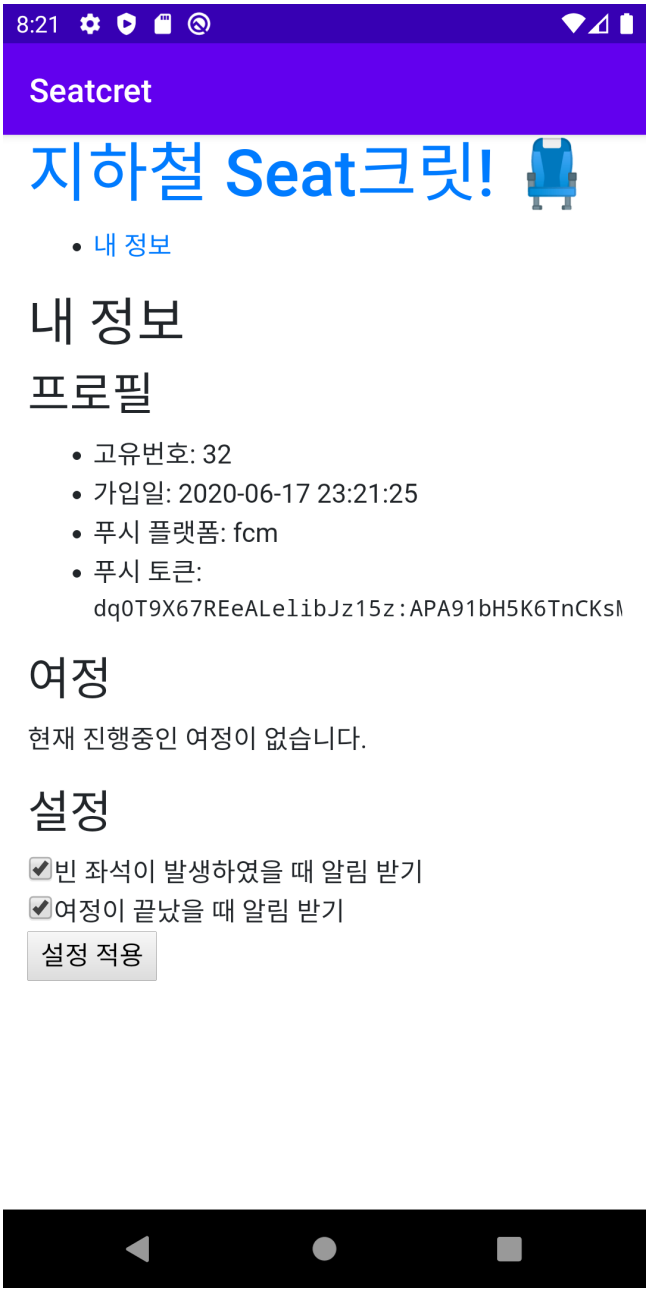


5.3.2 안드로이드 앱 – 열차 화면

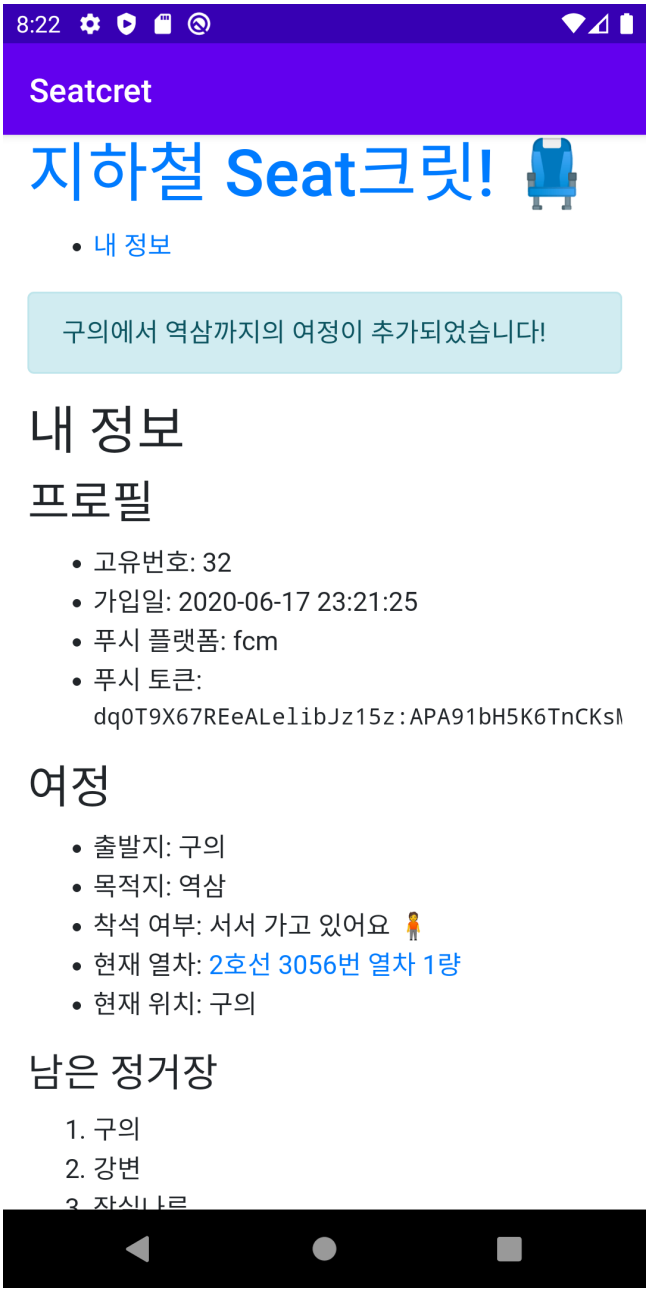


5.3.3 안드로이드 앱 – 좌석 화면

5.3.4 안드로이드 앱 – 프로필 화면



5.3.5 안드로이드 앱 – 여정 화면



5.4 서울 열린데이터 광장 파이썬 SDK

일반적으로, 공개 API 를 제공하는 서비스들은 API를 HTTP 프로토콜을 통해 제공합니다. 또, 그러한 HTTP 엔드포인트를 고객들이 사용하는 다양한 환경에서 손쉽게 호출하고 활용할 수 있도록, 여러 프로그래밍 언어에 대응하는 SDK를 제공하는 경우가 많습니다.

예를 들어 Amazon Web Services 파이썬 SDK[11], 네이버 Open API iOS SDK[12] 등, 여러 회사에서 자사의 서비스에 대한 SDK를 제공합니다. 이러한 SDK들은, HTTP 엔드포인트를 호출하는 로직과 오류 처리 코드 등을, 중복으로 작성하는 노력을 줄여줍니다.

이번 졸업프로젝트를 진행하며 활용하게 된 서울 열린데이터 광장에서는, 서울시에서 공개하는 정보를 확인할 수 있는 API를 제공하고 있습니다. 졸업프로젝트를 위한 지하철 실시간 위치정보 수집을 자동화를 하는 과정에서, 이러한 공개 데이터를 활용하기 쉽도록 서울 열린데이터 광장 파이썬 SDK를 작성하고 공개하였습니다.

파이썬 인터프리터가 설치된 컴퓨터에서, 지금 즉시 파이썬 패키지 설치 도구로 서울 패키지를 설치하고, 서울 패키지를 임포트하여 사용하실 수 있습니다. SDK의 활용 예제와 소스 코드는 깃헙 저장소[13]나 파이썬 패키지 인덱스[14]에서 확인하실 수 있습니다.

서울 열린데이터 광장에서는 약 4,500 개의 오픈 API를 제공하고 있는데, 이번 프로젝트를 위해 지하철 실시간 위치정보 조회에 대한 SDK 메서드를 구현하였습니다. 비록 전체 API 갯수에 비하면 적은 양이지만, 파이썬 데이터클래스와 HTTP 요청 재시도 메커니즘 등, 여러 사용성에 도움이 되는 모범적인 코드를 작성하기 위해 노력했습니다.

또, 예제 코드를 포함한 문서를 게시하고 공식 패키지 인덱스에 등록하는 등 좋은 오픈 소스 프로젝트의 기준에 부합하기 위해 코드 외적인 부분에도 신경을 썼습니다. 마지막으로, 오픈 소스 프로젝트인 만큼, 여러 사람이 관심을 가져 지원하는 API가 더욱 많아지면 좋겠습니다.

References

[1] Python Software Foundation, *Python*, https://www.python.org/

[2] Wikipedia, *QR code*, https://en.wikipedia.org/wiki/QR_code

[3] Jetbrains, *Kotlin Programming Language* https://kotlinlang.org/

[4] Android Developers, *App Link*,
https://developer.android.com/training/app-links

[5] *지하철 Seat크릿! 서버*,
https://github.com/seatcret/server

[6] Open Source Initiative, *The MIT License*
https://opensource.org/licenses/MIT

[7] Redis Labs, *Redis* https://redis.io/

[8] *지하철 Seat크릿! 안드로이드 클라이언트*,
https://github.com/seatcret/android

[9] Android Developers, *WebView*
https://developer.android.com/reference/android/webkit/WebView

[10] Google, *Firebase Cloud Messaging*
https://firebase.google.com/docs/cloud-messaging

[11] Amazon Web Services, *AWS SDK for Python*,
https://github.com/boto/boto3/

[12] Naver, *Naver Maps iOS SDK*,
https://github.com/navermaps/maps.ios

[13] 서지혁, *서울 열린데이터 광장 파이썬 SDK*,
https://github.com/limeburst/seoul-py

[14] 서지혁, *파이썬 패키지 인덱스 – 서울*,
https://pypi.org/project/seoul/