휠체어 이용자 저상버스 예약 시스템

작성자

고대영

목차

개발 배경

사회문제

요즘 장애인들의 탈 권리가 대두되면서 휠체어 이용자들의 대중교통이용 문제가 드러남

문제 상황 인식

휠체어 이용자들이 이용할 수 있도록 탑승구가 낮게 설계된 버스를 저상버스라고 함.

하지만 이러한 저상버스 탑승 시 운전기사는 휠체어 이용자가 어느 정류장에서 탑승 할지 모르기 때문에 휠체어 이용자가 탈 수 있는 환경을 마련하기 어려움

예) 휠체어 이용자가 탑승 가능한 뒷문을 휠체어 탑승자의 위치에 맞춰야 함. 휠체어 이용자가 앉을 수 있는 전용 자석이 따로 배치되어있음. 따라서 이 자리에 앉고 있던 사람은 자리를 비켜야 하고 휠체어 이용자가 이 자리를 이용할 수 있도록 세팅 해야함. 하지만 이 세팅 과정이 오래 걸리고 중간에 버스가 출발하게 되면 휠체어 이용자는 사고의 위험이 있음

개발 동기

현실에서 찾아볼 수 있는 문제점

실제로 휠체어 이용자들은 대중교통이용을 꺼려하며 이는 유튜브와 같은 매체에서도 쉽게 찾아볼 수 있다.

문제 해결을 위한 방안

휠체어 이용자가 자신이 탑승할 저상버스를 예약하는 시스템을 만들어서 운전기사와 상호작용 할 수 있도록 하면 된다.

방안에 대한 구체화

누구나 간편하게 사용하는 핸드폰을 이용하여 예약을 간단하게 할 수 있도록 한다. 즉 어플리케이션을 만들어서 사용하도록 한다.

개발 목적

사회적 약자를 위한 시스템

사회적 약자인 장애인을 위한 서비스가 마련되어야 한다고 생각 하였음. 프로그래밍 기술로 충분히 쉽게 사회적 약자를 위한 시스템을 마련할 수 있다면 효율성과 실용성이 매우 높아질 것이라는 기대효과

저상버스 이용 증대

실제로 저상버스를 이용하는 휠체어 이용자를 보기 쉽지 않다. 휠체어 이용자가 적기 때문이 아니라 대중교통이용을 꺼려하기 때문이다. 휠체어 이용자를 위한 저상버스를 마련해 놓고 휠체어 이용자를 위한 서비르를 수립하지 않는다면 저상버스를 만든 기대효과는 반감된다. 따라서 예약 앱을 통해 저상버스 이용률을 증대시킨다.

개발 전 준비 사항

휠체어 이용자 특성 분석

휠체어 이용 특성상 손이 바쁘고 불편

✂✂🖮음성인식 기능 추가

이용자 수를 확인하고 서버관리

✂✂🖮로그인 기능 추가

휠체어 이용자가 탑승 가능한 저상버스 유무 확인

✂✂🖮저상버스의 유무 표시

간단한 인터페이스 구현

✂✂🖮간단한 터치 위주의 인터페이스 구현

버스운전기사 특성 상 단말기를 보지 못함

✂✂🖮음성출력 기능 추가

공공데이터 파싱 연계

공공 데이터 사용

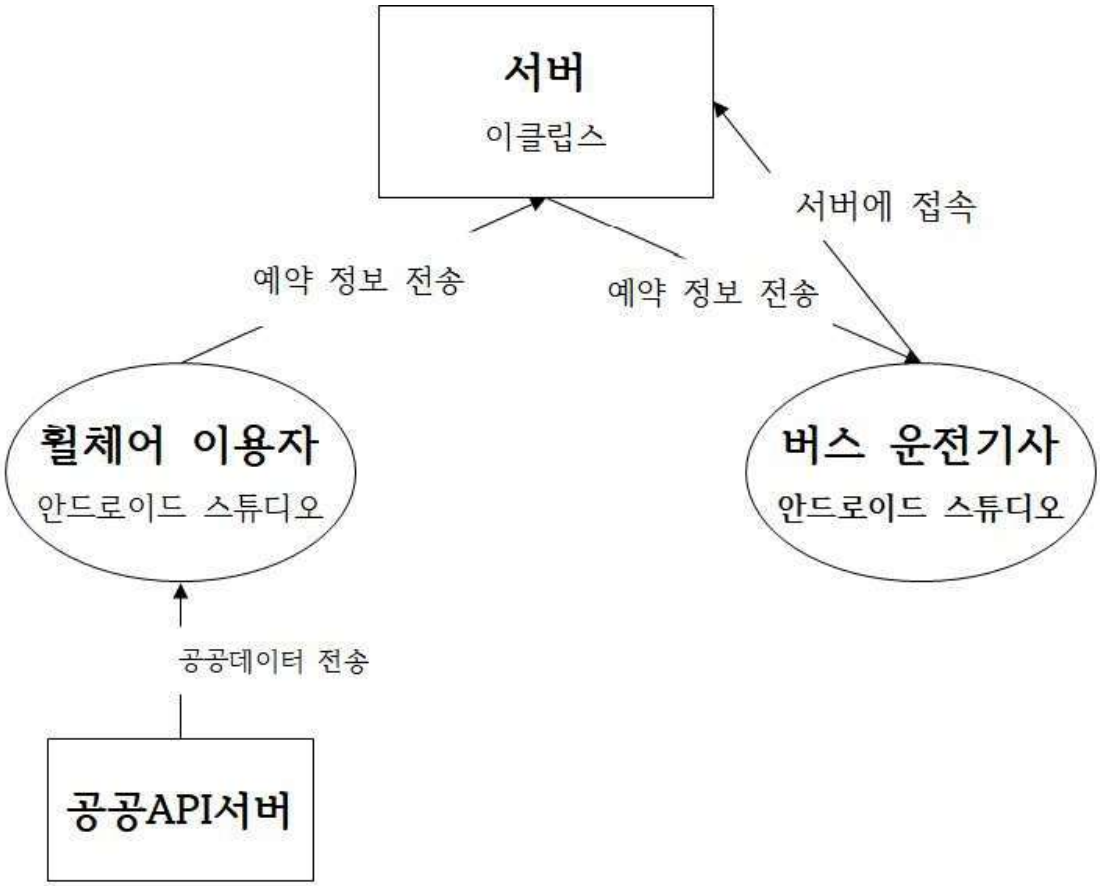
공공 데이터 포털에서 공공 데이터 사용 요청하기

1.경기도_정류소 조회 데이터를 이용하여 정류소데이터를 불러오고, 정류소 id를 추출한다.

2. 추출한 정류소 id를 경기도_버스도착정보 조회 데이터의 입력 값으로 넣어 도착 버스를 불러온다.

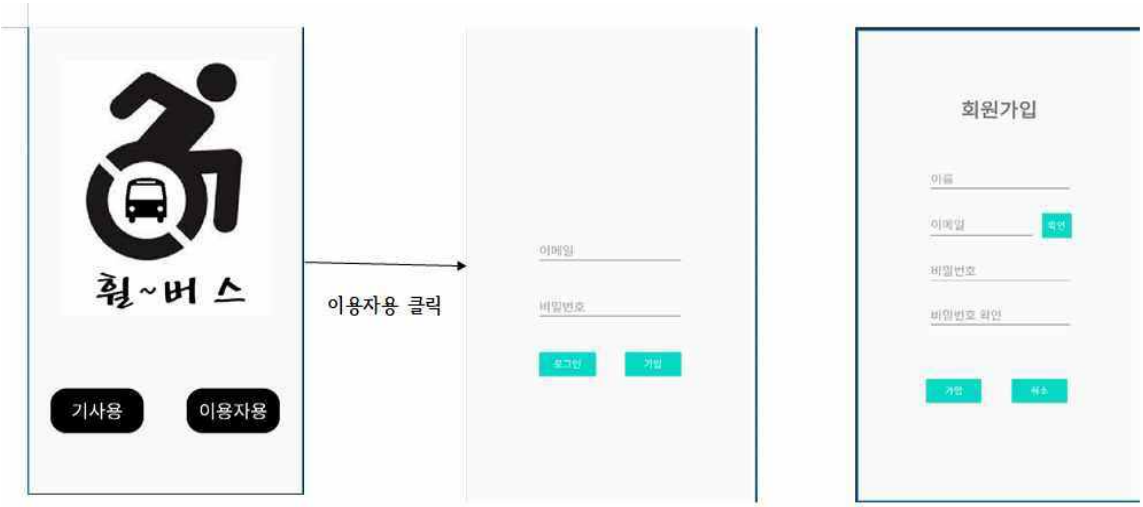
3. 경기도_버스도착정보 데이터에서 출력되는 버스노선id를 경기도_버스노선 조회 데이터를 이용하여 버스 번호로 출력 되도록 한다.

통신 및 데이터 처리 구성



1. 버스 운전기사는 주행 시작 버튼을 눌러 서버에 접속한다.
2. 휠체어 이용자는 정류소 검색을 이용하여 공공API서버에서 데이터를 QKE아온따.
3. 휠체어 이용자는 예약 버튼으로 데이터(자신이 탑승할 정류소와 버스 번호)를 서버에 전송한다.
4. 서버에 데이터가 전달되면 해당 버스 운전기사는 데이터를(정류소와 버스번호)를 실시간으로 알 수 있다.

앱 기능 소개(휠체어 이용자 UI)



앱 실행 시 기사용과 이용자용으로 나뉜다.

아이디와 비밀번호를 입력하여 로그인한다.

회원가입 시 데이터베이스에 저장되어 이용자 수를 확인할 수 있다.

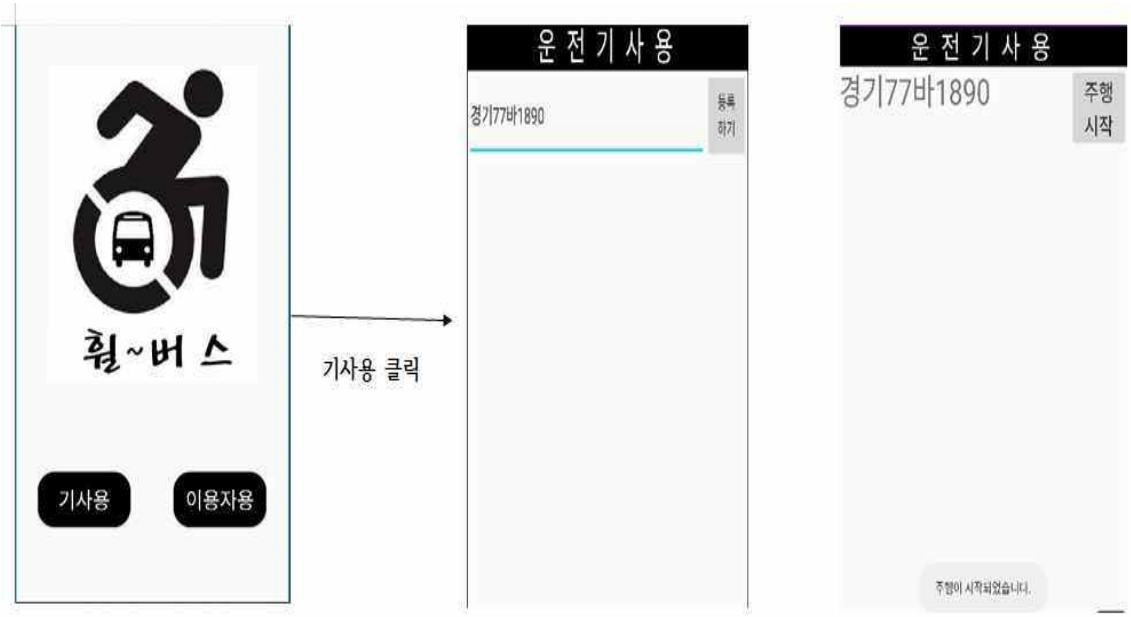
로그인 시 검색창이 나온다.
음성인식 기능을 이용하여 버스 정류소 이름을 말하면 정류장이 검색된다.

정류소명을 입력하면 해당 정류소명이 나온다. 각 정류소는 **정류소 번호** 또는 **정류소 id**로 구분할 수 있도록 하였다.

탑승하고자 하는 **정류소**를 클릭하면 **해당 정류소에 도착하는 버스 정보**가 나온다. **버스 번호, 위치, 도착 예정 시간, 차량 번호, 저상버스 여부**가 출력

탑승하고자 하는 **버스**를 클릭하면 **버스 정보**와 **예약 버튼**이 나온다. **예약 버튼**을 클릭하면 **서버**에 **탑승 데이터**(정류소, 차량 번호)가 **전달**된다.

앱 기능 소개(저상버스 운전 기사용)



앱 실행 시 기사용과 이용자용으로 나뉜다.

저상버스 운전기사는 운전 직전 자신의 차량번호를 등록한다.

주행 시작 바로 직전 주행시작 버튼을 눌러 서버에 접속한다.

개발 주요 기술(회원가입 및 로그인)

1_안드로이드는 MY SQL과 직접적인 연동이 안되기 때문에 '닷홈' 사이트에서 호스팅 계정을 만든다.

2_ 파일질라에 PHP파일을 업로드한다.

3_ 안드로이드 스튜디오에서 자신의 호스팅 DB URL과 연동한다.

RegiisterRequest.java회원가입PHP와연동

```
public class RegisterRequest extends StringRequest{
    //서버 URL 설정(php 파일 연동)
    final static private String URL = "http://kody1117.dothome.co.kr/Register.php";
    private Map<String, String> map;
    //private Map<String, String>parameters;

    public RegisterRequest(String UserEmail, String UserPwd, String UserName,Response.Listener<String> listener) {
        super(Method.POST, URL, listener,  errorListener: null);

        map = new HashMap<>();
        map.put( k: "UserEmail", UserEmail);
        map.put( k: "UserPwd", UserPwd);
        map.put( k: "UserName", UserName);
    }
```

LoginRequest.java로그인PHP와연동

```
public class LoginRequest extends StringRequest {
    //서버 URL 설정(php 파일 연동)
    final static private String URL = "http://kody1117.dothome.co.kr/Login.php";
    private Map<String, String> map;

    public LoginRequest(String UserEmail, String UserPwd, Response.Listener<String> listener) {
        super(Method.POST, URL, listener,  errorListener: null);

        map = new HashMap<>();
        map.put( k: "UserEmail", UserEmail);
        map.put( k: "UserPwd", UserPwd);
    }
```

*변수에 입력 데이터를 저장한다.

RegisterActivity.java

```
join_button = findViewById( R.id.join_button );
join_button.setOnClickListener(view -> {
    final String UserEmail = join_email.getText().toString();
    final String UserPwd = join_password.getText().toString();
    final String UserName = join_name.getText().toString();
    final String PassCk = join_pwck.getText().toString();
```

*객체를 생성하여 데이터를 한번에 서버(ResisterRequest.java)로 전송하고 ResisterRequest에서 해시맵에 데이터를 저장한다.(위에 ResisterRequest 생성자로 전달)

```
    //서버로 Volley를 이용해서 요청
    RegisterRequest registerRequest = new RegisterRequest( UserEmail, UserPwd, UserName, responseListener);
    RequestQueue queue = Volley.newRequestQueue( context: RegisterActivity.this );
    queue.add( registerRequest );
});
```

*회원 가입 후 DB에 저장된 모습

*입력값과 DB를 비교하여 로그인을 할 수 있다.

EditText와 연결

```
login_email = findViewById( R.id.login_email );
login_password = findViewById( R.id.login_password );
```

입력 데이터를 변수에 저장

```
public void onClick(View view) {
    String UserEmail = login_email.getText().toString();
    String UserPwd = login_password.getText().toString();
```

객체를 생성하여 한번에 LoginRequest로 변수의 데이터를 전송 (위에 LoginRequest생성자로 전달)

```
LoginRequest loginRequest = new LoginRequest( UserEmail, UserPwd, responseListener );
RequestQueue queue = Volley.newRequestQueue( context: LoginActivity.this );
queue.add( loginRequest );
```

아이디의 중복을 확인하는 ValoidateRequest Class

```
public class ValidateRequest extends StringRequest {
    //서버 url 설정(php파일 연동)
    final static  private String URL="http://kody1117.dothome.co.kr/UserValidate.php";
    private Map<String, String> map;

    public ValidateRequest(String UserEmail, Response.Listener<String> listener){
        super(Method.POST, URL, listener, errorListener: null);

        map = new HashMap<>();
        map.put( k: "UserEmail", UserEmail);
    }
```

개발 주요 기술(검색기능-음성인식, xml파싱, 리사이클러뷰)

마이크 권한과 인터넷 권한을 위한 AndroidManifest.xml 설정

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.RECORD_AUDIO"/>;
```

****앱에서 데이터를 출력하는 방법 중 하나는 리사이클러뷰이다. 반복적으로 데이터를 출력해야하는 경우 데이터가 들어갈 빈 껍데기를 만들고 껍데기를 계속 출력할 수 있도록 한다. 그 후에 껍데기에 데이터를 집어넣고 반복적으로 출력할 수 있도록 한다. 리사이클러뷰를 사용하기 위해서는 item.class와 adapter등이 필요하다.****

Item.class

자신이 파싱해서 얻어오고 싶은 데이터를 저장하고 string을 리턴하는 클래스를 만든다

```
public class Item {
    String stationName;
    String mobileNo;
    String stationId;

    public String getStationId() { return stationId; }

    public String getStationName() { return stationName; }

    public void setStationId(String stationId) { this.stationId = stationId; }

    public void setStationName(String stationName) { this.stationName = stationName; }

    public String getMobileNo() { return mobileNo; }

    public void setMobileNo(String mobileNo) { this.mobileNo = mobileNo; }
}
```

MyAdapter.calss

앞 서 만든 Item클래스를 이용하여 ArrayList를 만들고 생성자를 만들어 차후에 파싱한 데이터를 받아 저장한다.

```
public class MyAdapter extends RecyclerView.Adapter<MyAdapter.MyViewHolder> {
    private ArrayList<Item> mList;
    private LayoutInflater mInflate;
    private Context mContext;
    private Intent intent;


    public MyAdapter(Context context, ArrayList<Item> itmes) {
        this.mList = itmes;
        this.mInflate = LayoutInflater.from(context);
        this.mContext = context;
    }
```

xml에서 만든 데이터를 출력할 변수와 java의 textview 변수를 연결한다.

```
    public static class MyViewHolder extends RecyclerView.ViewHolder {
        public TextView stationName;
        public TextView mobileNo;
        public TextView stationId;


        public MyViewHolder(View itemView) {
            super(itemView);

            this.mobileNo = itemView.findViewById(R.id.mobileNo);
            this.stationName = itemView.findViewById(R.id.stationName);
            this.stationId = itemView.findViewById(R.id.stationId);

        }
```

Item.class에 저장된 변수의 string을가져와 binding한다.

```
public void onBindViewHolder(@NonNull MyViewHolder holder,int position) {
    //binding
    holder.itemView.setTag(position);
    holder.mobileNo.setText(mList.get(position).getMobileNo());
    holder.stationName.setText(mList.get(position).getStationName());
    holder.stationId.setText(mList.get(position).getStationId());
    //Click event
```

데이터가 표시된 껍데기를 클릭시 intent를 이용하여 다음 액티비티에 변수를 전송한다. 여기서는 선택한 정류소 이름(name)과 다음 액티비티에서 사용될 경기도_버스도착정보조회의 요청변수값으로 전달될 해당 정류소id(id)를 전달한다.

```
public void onClick(View view) {
    int mposition = holder.getAdapterPosition();
    intent = new Intent(mContext,SubActivity.class); //look_memo.class부분에 원하는 화면 연결
    intent.putExtra( name: "id", mList.get(mposition).getStationId()); //변수값 인텐트로 넘기기
    intent.putExtra( name: "name",mList.get(mposition).getStationName());
    intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
    mContext.startActivity(intent); //액티비티 열기
```

Search.class

마이크 사용 권한을 부여하고 음성인식을 실행시키는 코드.

```
private void speak(){
    Intent intent = new Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH);
    intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE_MODEL,
            RecognizerIntent.LANGUAGE_MODEL_FREE_FORM);
    intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE, Locale.KOREA);
    intent.putExtra(RecognizerIntent.EXTRA_PROMPT,  value: "탑승할 정류장을 말하세요");
    if(intent.resolveActivity(getPackageManager())!=null) {
        try {

            startActivityForResult(intent, REQUEST_CODE_SPEECH_INPUT);
        } catch (Exception e) {
            Toast.makeText( context: this,  text: "" + e.getMessage(), Toast.LENGTH_SHORT).show();
        }
    } else{
        Toast.makeText( context: this, text: "마이크 사용불가, 정류장을 입력해주세요.",Toast.LENGTH_SHORT).show();
    }

    }
```

인식한 음성을 검색창(edit)에 출력한다. (TTS기능) 그리고 경기도_버스정류소조회 xml파일을 비동기적으로 파싱하기 위하여 AsyncTask을 호출한다.

비동기적으로 해야하는 이유는 파싱을 백그라운드에서 해야하기 때문

```
protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
    super.onActivityResult(requestCode, resultCode, data);

    switch (requestCode) {
        case REQUEST_CODE_SPEECH_INPUT: {
            if (resultCode == RESULT_OK && data != null) {
                ArrayList<String> result = data.getStringArrayListExtra(RecognizerIntent.EXTRA_RESULTS);
                edit.setText(result.get(0));
            }
            MyAsyncTask myAsyncTask = new MyAsyncTask();
            myAsyncTask.execute();
            break;
        }
    }
}
```

str변수에 edit의 데이터를 저장하고 그것을 인코딩하여 변수 location에 저장한다. 또한 공공데이터포털에서 제공되는 서비스 키를 저장한다.

```
String str= edit.getText().toString();//EditText에 작성된 Text얻어오기
String location = URLEncoder.encode(str);

String idDataKey="zYmgixu%2Fk4l6g7Yr3EbKZyY18Y%2FAm%2FVUpCbn803qZ9bejgTg0N1UBsgSiQkNkgOK1XUcTEcombAGC%2FB1Q%2FhmWg%3D%
```

공공데이터 포털에서 제공하는 형식에 맞게 idrequestUrl에 문자열을 구성한다.

요청변수는 location(정류소 이름)이다.

```
protected String doInBackground(String... strings) {

    idrequestUrl="http://apis.data.go.kr/6410000/busstationservice/getBusStationList?"
            +"serviceKey="+idDataKey
            +"&keyword=" + location;
```

안드로이드에서 제공되는 xml파싱함수를 이용하여 파싱한다.

```
URL url = new URL(idrequestUrl);
InputStream is = url.openStream();
XmlPullParserFactory factory = XmlPullParserFactory.newInstance();
XmlPullParser parser = factory.newPullParser();
parser.setInput(new InputStreamReader(is,  charsetName: "UTF-8"));
```

Xml 문서의 태그가 시작될때마다 자신이 추출하고 싶은 string을 저장하는 boolean 변수를 true로 초기화 시킨다.

```
            case XmlPullParser.START_TAG:
                if (parser.getName().equals("busStationList")) {
                    bus = new Item();
                }
                if (parser.getName().equals("stationId")) b_stationId = true;
                if (parser.getName().equals("mobileNo")) b_mobileNo = true;
                if (parser.getName().equals("stationName")) b_stationName = true;

                break;
```

Xml문서의 내용 태그가 실행되면 자신이 원하는 string을 추출하여 bus(item.class)에 set한다. 자신이 원하는 string이 안닌경우 다음라인으로 넘어가서 순차적으로 검색한다.

```
                case XmlPullParser.TEXT:
                    if (b_mobileNo) {
                        bus.setMobileNo(parser.getText());
                        b_mobileNo = false;
                    } else if (b_stationName) {
                        bus.setStationName(parser.getText());
                        b_stationName = false;
                    } else if(b_stationId){
                        bus.setStationId(parser.getText());
                        b_stationId = false;
                    }
                    break;
            }
            eventType = parser.next();
        }
```

어답터 객체를 사용하여 리스트로 전송하고 출력한다.

```
        protected void onPostExecute(String s) {
            super.onPostExecute(s);

            //어답터 연결
            MyAdapter adapter = new MyAdapter(getApplicationContext(), list);
            recyclerView.setAdapter(adapter);
        }
    }
}
```

개발 주요 기술

(도착 버스 조회-xml 2중파싱,리사이클러뷰)

Busitem.class

경기도_버스도착정보의 파싱 결과를 저장하고 리턴하는 class를 만듦

```
public class Busitem {
    String locationNo1;
    String plateNo1;
    String routeId;
    String predictTime1;
    String lowPlate1;

    public String getLowPlate1(){ return  lowPlate1; }

    public void setLowPlate1(String lowPlate1) { this.lowPlate1 =lowPlate1; }

    public String getLocationNo1() { return locationNo1; }

    public void setLocationNo1(String locationNo1) { this.locationNo1 = locationNo1; }

    public String getPlateNo1() { return plateNo1; }

    public void setPlateNo1(String plateNo1) { this.plateNo1 = plateNo1; }

    public String getRouteId() { return routeId; }

    public void setRouteId(String routeId) { this.routeId = routeId; }

    public String getPredictTime1() { return predictTime1; }

    public void setPredictTime1(String predictTime1) { this.predictTime1 = predictTime1; }
```

SubAdapter.class

앞 서 만든 Busitem클래스를 이용하여 ArrayList를 만들고 생성자를 만들어 차후에 파싱한 데이터를 받아 저장한다

```
public class SubAdapter extends RecyclerView.Adapter<SubAdapter.MyViewHolder> {

    private ArrayList<Busitem> mList;
    private LayoutInflater mInflate;
    private Context mContext;
    private String stopName;
    private Intent intent;

    public SubAdapter(Context context, ArrayList<Busitem> itmes, String name) {
        this.mList = itmes;
        this.mInflate = LayoutInflater.from(context);
        this.mContext = context;
        this.stopName = name;
    }
```

xml에서 만든 데이터를 출력할 변수와 java의 textview 변수를 연결한다.

```
    public static class MyViewHolder extends RecyclerView.ViewHolder {
        public TextView locationNo1;
        public TextView plateNo1;
        public TextView routeId;
        public TextView predictTime1;
        public TextView lowPlate1;

        public MyViewHolder(View itemView) {
            super(itemView);

            locationNo1 = itemView.findViewById(R.id.locationNo1);
            plateNo1 = itemView.findViewById(R.id.plateNo1);
            routeId = itemView.findViewById(R.id.routeId);
            predictTime1 = itemView.findViewById(R.id.predictTime1);
            lowPlate1=itemView.findViewById(R.id.rowPlate1);
        }
    }
```

Item.class에 저장된 변수의 string을가져와 binding한다.

```
public void onBindViewHolder(@NonNull MyViewHolder holder, int position) {
    //binding
    holder.itemView.setTag(position);
    holder.locationNo1.setText(mList.get(position).locationNo1);
    holder.plateNo1.setText(mList.get(position).plateNo1);
    holder.routeId.setText(mList.get(position).routeId);
    holder.predictTime1.setText(mList.get(position).predictTime1);
    holder.lowPlate1.setText(mList.get(position).lowPlate1);
    //Click event
```

Intent를 이용하여 데이터가 표시된 껍데기를 클릭시 putExtra를 이용하여 다음 액티비티에 변수를 전송한다.

```
public void onClick(View view) {
    int mposition = holder.getAdapterPosition();
    intent = new Intent(mContext,ResultActivity.class); //look_memo.class부분에 원하는 화면 연결
    intent.putExtra( name: "route", mList.get(mposition).getRouteId()); //변수값 인텐트로 넘기기
    intent.putExtra( name: "location",mList.get(mposition).getLocationNo1());
    intent.putExtra( name: "plate",mList.get(mposition).getPlateNo1());
    intent.putExtra( name: "predict",mList.get(mposition).getPredictTime1());
    intent.putExtra( name: "name",stopName);
    intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
    mContext.startActivity(intent); //액티비티 열기

}
```

SubActivity.class

인텐트를 이용하여 이전 액티비티에서 전달받은 정류소 이름(name)과 정류소 id(id)를 getExtra로 받아와 각각 name과 routeId변수에 저장한다. 이후 비동기적으로 xml을 파싱하기 위하여 AsnycTask를 실행한다.

```
intent = getIntent();
routeId = intent.getStringExtra( name: "id");
name= intent.getStringExtra( name: "name");

MyAsyncTask myAsyncTask = new MyAsyncTask();
myAsyncTask.execute();
```

routeId를 인코딩 하여 location변수에 저장하고 공공데이터포털에서 제공하는 서비스키를 arrDataKey에 저장한다.

```
String location = URLEncoder.encode(routeId);

String arrDataKey = "zYmqixu%2Fk4l6g7Yr3EbKZyY18Y%2FAm%2FVUpCbn803qZ9bejqTg0N1UBsgSiQkNkgOK1XUcTEcomhAGC%2FB1Q%2FhmWg%3D

String arrRequestUrl;
```

백그라운드로 공공데이터포털에서 제공하는 형식에 맞게 url을 구성한다. 요청변수는 location이다.

```
protected String doInBackground(String... strings) {


    arrRequestUrl="http://apis.data.go.kr/6410000/busarrivalservice/getBusArrivalList?"//요청 URL
            +"serviceKey="+arrDataKey
            +"&stationId=" +location;
```

안드로이드에서 제공하는 함수를 이용하여 xml을 파싱한다.

```
URL url = new URL(arrRequestUrl);
InputStream is = url.openStream();
XmlPullParserFactory factory = XmlPullParserFactory.newInstance();
XmlPullParser parser = factory.newPullParser();
parser.setInput(new InputStreamReader(is, charsetName: "UTF-8"));
```

Xml 문서의 태그가 시작될때마다 자신이 추출하고 싶은 string을 저장하는 boolean 변수를 true로 초기화 시킨다.

```
case XmlPullParser.START_TAG:
    if (parser.getName().equals("busArrivalList")) {
        bus = new Busitem();
    }
    if (parser.getName().equals("locationNo1")) b_locationNo1 = true;
    if (parser.getName().equals("plateNo1")) b_plateNo1 = true;
    if (parser.getName().equals("routeId")) b_routeId = true;
    if (parser.getName().equals("predictTime1")) b_predictTime1 = true;
    if (parser.getName().equals("lowPlate1")) b_lowPlate1 = true;
    break;
```

Xml문서의 내용 태그가 실행되면 자신이 원하는 string을 추출하여 bus(item.class)에 set한다. 자신이 원하는 string이 안닌경우 다음라인으로 넘어가서 순차적으로 검색한다.

```
case XmlPullParser.TEXT:
    if (b_lowPlate1) {
        //bus.setLowPlate1(parser.getText());
        if (parser.getText().equals("1")) {
            bus.setLowPlate1("저상버스");
        }
        else{
            bus.setLowPlate1("휠체어 탑승불가");
        }
        b_lowPlate1 = false;
    }

    else if (b_locationNo1) {
            bus.setLocationNo1(parser.getText());
            b_locationNo1 = false;
        } else if (b_plateNo1) {
            bus.setPlateNo1(parser.getText());
            b_plateNo1 = false;
        } else if (b_routeId) {
            newnum = busNum(parser.getText());
            bus.setRouteId(newnum);
            b_routeId = false;
        } else if (b_predictTime1) {
            bus.setPredictTime1(parser.getText());
            b_predictTime1 = false;
        }
        break;
```

버스노선id를 버스번호로 출력하기 위하여 busNum함수를 실행시켜 2중으로 파싱을 해야한다. 즉 경기도_버스도착정보에서 버스노선id를 추출하여 경기도_버스노선조회의 요청변수에 전달해야한다.

```
} else if (b_routeId) {
    newnum = busNum(parser.getText());
    bus.setRouteId(newnum);
    b_routeId = false;
```

경기도_버스노선조회 xml은 여러개의 데이터를 추출할 필요가 없다. 왜냐하면 버스번호 태그만 추출하면 되기 때문이다. 따라서 xmlPullParser 대신 DocumentBuilderFactory을 사용하여 파싱한다.

UrlBuilder에 공공데이터포털에서 제공하는 형식에 맞게 url을 구성한다.

```
StringBuilder urlBuilder = new StringBuilder("http://apis.data.go.kr/6410000/busrouteservice/getBusRouteInfoItem");
urlBuilder.append("?" + URLEncoder.encode( s: "serviceKey", enc: "UTF-8") + "=zYmgixu%2Fk4l6g7Yr3EbKZyY18Y%2FAm%2FVUp
urlBuilder.append("&" + URLEncoder.encode( s: "routeId", enc: "UTF-8") + "=" + URLEncoder.encode(inner, enc: "UTF-8"))
URL url = new URL(urlBuilder.toString());
HttpURLConnection conn = (HttpURLConnection) url.openConnection();
conn.setRequestMethod("GET");
```

HttpURLConnection을 이용하여 url을 서버에 연결한다.

```
HttpURLConnection conn = (HttpURLConnection) url.openConnection();
conn.setRequestMethod("GET");
conn.setRequestProperty("Content-type", "application/json");
BufferedReader rd;
if (conn.getResponseCode() >= 200 && conn.getResponseCode() <= 300) {
    rd = new BufferedReader(new InputStreamReader(conn.getInputStream()));
} else {
    rd = new BufferedReader(new InputStreamReader(conn.getErrorStream()));
}
StringBuilder sb = new StringBuilder();
String line;
while ((line = rd.readLine()) != null) {
    sb.append(line);
}
rd.close();
conn.disconnect();
```

DocumentBuilderFactory를 이용해서 파싱한다. 시작 태그인 busRouteInfoItem을 만나면 리스트의 길이만큼 for문을 돌리고 추출하고자 하는 routeName태그의 String을 변수 outer에 저장한다.

```
try {
    DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();
    DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
    String parsingUrl = urlBuilder.toString();
    Document doc = dBuilder.parse(parsingUrl);
    doc.getDocumentElement().normalize();
    NodeList nList = doc.getElementsByTagName( s: "busRouteInfoItem");
    for (int temp = 0; temp < nList.getLength(); temp++) {
        Node nNode = nList.item(temp);
        org.w3c.dom.Element eElement = (Element) nNode;
        outer = getTagValue( tag: "routeName", (org.w3c.dom.Element) eElement);
        return outer;
    }
}catch (Exception e){
    e.printStackTrace();
}

return out;
```

RouteName의 문자열을 추출하는 함수

```
protected String getTagValue(String tag, org.w3c.dom.Element eElement) {
    //결과를 저장할 result 변수 선언
    String result = "";
    NodeList nlList = eElement.getElementsByTagName(tag).item( i: 0).getChildNodes();
    if(nlList != null) {
        Node nItem = nlList.item( i: 0);
        if(nItem != null)
            result = nItem.getTextContent();
        //result = nlList.item(0).getTextContent();
    }
    return result;
}
```

이전에 만든 SubAdapter객체를 생성하여 데이터를 연결한다.

```
        //어답터 연결
        SubAdapter adapter = new SubAdapter( getApplicationContext(), list , name);
        recyclerView.setAdapter(adapter);
    }
```

개발 주요 기술

(서버와 소켓 통신)

ResultActivity.class

Intent로 이전 액티비티에서 받아온 데이터를 현재 액티비티 변수에 연결한다.

```
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_result);

intent = getIntent();
routeId= intent.getStringExtra( name: "route");
plateNo1=intent.getStringExtra( name: "plate");
locationNo1=intent.getStringExtra( name: "location");
predictTime1=intent.getStringExtra( name: "predict");
name=intent.getStringExtra( name: "name");

TextView t_name=findViewById(R.id.stopName);
TextView t_routeId=findViewById(R.id.routeId);
TextView t_plateNo1=findViewById(R.id.plateNo1);
TextView t_predictTime1=findViewById(R.id.predictTime1);
TextView t_locationNo1=findViewById(R.id.locationNo1);



t_name.setText(name);
t_routeId.setText(routeId);
t_plateNo1.setText(plateNo1);
t_predictTime1.setText(predictTime1);
t_locationNo1.setText(locationNo1);
```

탑승 정류소 이름과 탑승 차량번호를 구분자 '/'를 이용하여 변수 input에 저장한다. 이후 서버와 소켓통신을 위하여 SocketThread를 이용하여 스레드를 생성

```
input=name.concat("/").concat(plateNo1);

socketButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        SocketThread thread=new SocketThread(input);
        thread.start();
```

생성자로 input을 받아온다

```
String temp;
String input; // 탑승위치/차량번호
PrintWriter out =null;

public SocketThread(String input) { this.input = input; }
```

서버의 ip와 포트를 입력하여 서버소켓에 연결한다.

```
int port = 9999; //포트 번호는 서버측과 똑같이
Socket socket = new Socket( host: "172.20.10.6" , port); // 소켓 열어주기
```

sendAll함수를 이용하여 input(탑승정류장/차량번호)를 넣어서 출력한다.

이 input값이 서버에 전달되고 서버에 연결된 해당 저상버스 운전기사 단말기에 출력된다.

```
        sendAll( s: "user");
        sendAll(input);
        sendAll( s: "quit");
        socket.close(); // 소켓 해제
```

```
private void sendAll(String s) throws UnsupportedEncodingException {
    for(PrintWriter out: list) {
        out.println(s);
        out.flush();
    }
}
```

개발 주요 기술

(소켓 통신,음성출력TTS)

DriverDrive.class

운전기사용 ui에서 차량 번호를 등록하고 주행시작 버튼을 클릭하면
SocketThread함수를 이용해 소켓통신을 위한 스레드가 생성된다.

```
        @Override
        public void onClick(View view) {
            DriverDrive.SocketThread thread=new DriverDrive.SocketThread(busNum);
            thread.start();
            Toast.makeText( context: DriverDrive.this,  text: "주행이 시작되었습니다.", Toast.LENGTH_LONG).show();
        }
    });
}
```

생성자를 이용하여 차량번호를 busNum 변수에 저장한다.

```
    String busNum;
    public SocketThread(String busNum) {
        this.busNum=busNum;        }
```

포트번호와 서버ip를 입력하여 서버에 접속한다.

```
    try{
        int port = 9999; //포트 번호는 서버측과 똑같이
        Socket socket = new Socket( host: "172.20.10.6" , port); // 소켓 열어주기

        in=new BufferedReader(new InputStreamReader(socket.getInputStream()));
```

휠체어 이용자는 자신이 탑승하고자 하는 정보를 '정류소이름/차량번호' 형태로 서버에 전달한다. 버스 기사 단말에서 데이터에 자신의 차량번호(busNum)가 포함된 메세지가 오면 split함수를 이용하여 정류소 이름을 따로 추출하여 변수 busStation에 저장한다.

```
if (inputMsg.contains(busNum)) {
    busStation=inputMsg.split( regex: "/")[0];
    handler.post(new Runnable() {
        @Override
        public void run() {
            Toast.makeText( context: DriverDrive.this,  text: busStation+"정류장에 휠체어 이용자가 있습니다.
        }
    });
```

이후 버스 운전기사 단말에서 음성이 출력되도록 한다.

```
    tts.speak( text: busStation+"정류장에 휠체어 이용자가 있습니다",TextToSpeech.QUEUE_FLUSH, params: null);
}
```

음성 출력을 위한 tts를 구현한다.

```
tts = new TextToSpeech( context: this, new TextToSpeech.OnInitListener() {
    @Override
    public void onInit(int status) {
        if(status != ERROR) {
            // 언어를 선택한다.
            tts.setLanguage(Locale.KOREAN);

        }
    }
});
```

개발 주요 기술

(서버)

소켓통신을 위하여 소켓포트를 생성하고 receiveThread를 호출하여 수신 스레드를 생성한다.

```
1  package wheelchairproject;
2
3  import java.io.*;
6
7  public class MultiServer {
8
9      public static void main(String[] args) {
10         // TODO Auto-generated method stub
11         MultiServer multiServer=new MultiServer();
12         multiServer.start();
13     }
14
15     public void start() {
16         ServerSocket serverSocket = null;
17         Socket socket =null;
18         try {
19             serverSocket = new ServerSocket(9999);
20             while(true) {
21                 System.out.println("[클라이언트 연결대기중]");
22                 socket =serverSocket.accept();
23                 System.out.println("client가 접속함:"+socket.getInetAddress());
24
25                 ReceiveThread receiveThread=new ReceiveThread(socket);
26                 receiveThread.start();
27             }
28         }catch(IOException e) {
29             e.printStackTrace();
30         }finally {
31             if(serverSocket!=null) {
32                 try {
33                     serverSocket.close();
34                     System.out.println("[서버종료]");
35                 }catch(IOException e) {
36                     e.printStackTrace();
```

RecieveThreaed의 생성자로 소켓을 받아오고 out에 송신 데이터를, in에 수신 데이터를 저장한다.

```
45 class ReceiveThread extends Thread{
46     static HashMap<String,String> bus=new HashMap<>();
47@    static List<PrintWriter>list =
48             Collections.synchronizedList(new ArrayList<PrintWriter>());
49     Socket socket =null;
50     BufferedReader in =null;
51     PrintWriter out =null;
52
53@    public ReceiveThread(Socket socket) {
54         this.socket=socket;
55         try {
56             out = new PrintWriter(socket.getOutputStream());
57             in = new BufferedReader(new InputStreamReader(socket.getInputStream()));
58             list.add(out);
59         }catch(IOException e) {
60             e.printStackTrace();
61         }
62     }
63
64@    public void run() {
65
66
67
68         //String busNum="";
69         String order ="";
70         //String busname="";
```

서버는 휠체어 이용자와 버스기사를 구분할 줄 알아야 한다. 휠체어 이용자는 구분자 '/' 가 섞인 데이터만 송신한다. 따라서 구분자 '/' 의 포함 여부로 휠체어 이용자와 버스기사를 구분한다.

만약 데이터에 '/'가 포함되어 있으면 휠체어 이용자의 데이터이므로 기사에게 데이터를 뿌려준다.

```
71          //String stationName="";
72          //int code=0;
73
74
75          try{
76              order=in.readLine();
77              //busname=name;
78
79
80              while(in!=null) {
81
82
83                  String inputMsg=in.readLine();
84                  if(inputMsg.contains("/")) {
85                      sendAll(inputMsg);//"/"이 포함되어 있으면 휠체어 이용자가 보내는 예약 메시지.
86                      System.out.println(inputMsg);
87                  }
88
89                  if("quit".equals(inputMsg))break;
90
91              }
92
93          }catch(IOException e) {
94              System.out.println("["+order+"접속끊김]");
95          }finally {
96              sendAll("["+order+"]님이 나가셨습니다");
97              list.remove(out);
98              try {
99                  socket.close();
100             }catch(IOException e) {
101                 e.printStackTrace();
102             }
103         }
104         System.out.println("["+order+"연결종료]");
```

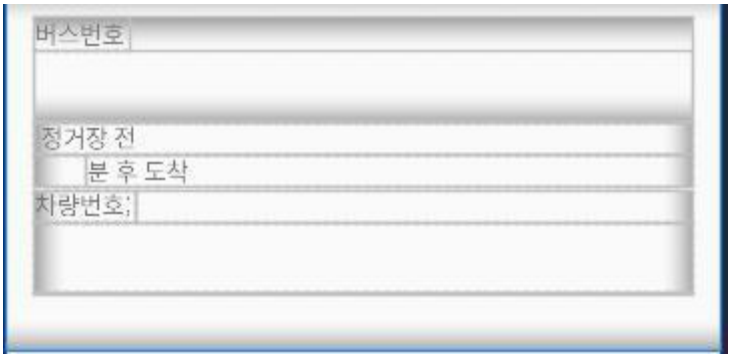
데이터를 뿌려주는 sendAll 함수

```
105     }
106
107     private void sendAll(String s) {
108         for(PrintWriter out: list) {
109             out.println(s);
110             out.flush();
111         }
112     }
113 }
114
```

개발 주요 기술

(UI-XML 리사이클러뷰를 위한 구성)

리사이클러뷰를 구현하기 위해 빈 껍데기를 만든다.

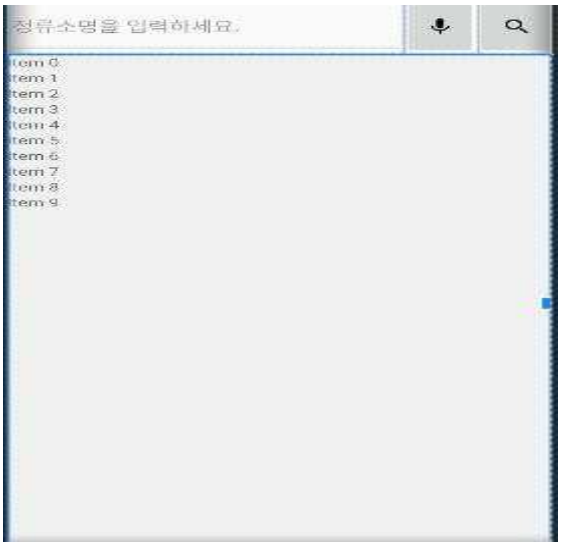


그 후 다른 xml 파일에서 밑에 recyclerview를 선언한다.

```
    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/recyclerView"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:background="#F0F0F0"
        android:scrollbars="vertical" />


</LinearLayout>
```

그러면 이런식으로 밑에 공간이 생기며 item이 나열된다.



이후 자신이 만든 java를 이용하여 xml에서 선언한 리사이클러뷰 아이디와 연결해주면 된다.

```
recyclerView = (RecyclerView) findViewById(R.id.recyclerView2);
recyclerView.setHasFixedSize(true);

LinearLayoutManager layoutManager = new LinearLayoutManager( context: this);
layoutManager.setOrientation(LinearLayoutManager.VERTICAL);
recyclerView.setLayoutManager(layoutManager);
```

Xml을 이용한 UI 구성은 대부분 LinearLayout 구조를 사용하였다. 위에서 부터 순차적으로 구성할 수 있다. 다른 구조에 비해 까다로울 수 있으나 구조가 일정하고 규칙적이므로 사용하였다.

개발 과정 및 문제 해결

개발 과정 1.

데이터 베이스와 XML을 이해하기 위하여 로그인과 회원가입 기능을 가장 먼저 구현 시도.

문제 발생

기능은 구현 하였으나 회원가입버튼 클릭시 데이터베이스와 연동이 안되는 현상이 발생.

이 문제를 해결하기 위해 여러가지 방법을 사용하였으나 안됨.

문제 원인

PHP를 처리하는 프로그램이 존재하지 않았다. Visual studio code나 메모장을 사용하였으나 결국 php 편집기 Brackets을 다운받아서 사용하였더니 해결

개발 과정 2.

파싱 구현을 위해 우선 이클립스에서 구현하였다. 하지만 안드로이드 스튜디오에서는 출력이 되지 않았다.

문제해결

안드로이드 스튜디오는 인터넷에 접속할 때 보안xml파일을 생성하여 그 곳에 사용할 url을 선언해줘야한다. 이 작업이 되지 않아 파싱이 되지 않았던 것.

문제발생

안드로이드 스튜디오에서 지속적으로 데이터의 양이 변하는 api를 파싱한 내용을 어떻게 출력할 것인가도 문제였다. 이클립스 서버에서 파싱을 끝낸 뒤 소켓통신으로 안드로이드 스튜디오로 전송하려 했으나, 그렇게 되면 소켓 스레드가 필요하게 되고 소켓 스레드를 실행하기 위해 추가 작업이 필요하다. 또한 변하는 데이터를 표현하기가 어렵다. 액티비티가 여러개인 안드로이드 스튜디오와 이클립스의 소켓통신으로 파싱데이터를 전송하는것은 부적절하다고 판단.

문제해결

데이터의 양이 변하는 api 파싱 데이터를 출력하기 위하여 리사이클러뷰가 효율적이다. 따라서 리사이클러뷰를 활용하여 구현하였다.

문제발생

경기도_버스도착정보는 정류소 이름이 아닌 고유 정류소id를 요청변수로 한다. 휠체어 이용자가 고유 정류소id를 알 리 없다. 결국 api를 여러개 사용해야 하는 상황이다.

문제해결

휠체어 이용자에게 정류소 이름을 입력받는다. 이후 정류소 이름을 요청변수로 파싱하여 경기도_정류소 조회로 부터 정류소id를 추출한다. 정류소id를 요청변수로 경기도_버스도착정보를 파싱한다.

문제발생

경기도_버스도착정보를 파싱한 결과 버스번호가 아닌 버스노선id가 출력 되었다. 휠체어 이용자는 버스노선id를 알 리 없다. 이렇게 되면 휠체어 이용자는 자신이 타려는 버스를 알 수없다.

문제해결

이 문제를 해결하기 위하여 이중파싱을 사용하였다. 경기도_버스도착정보를 파싱한 후 출력되는 값에서 버스노선id를 요청변수로 하여 경기도_버스노선조회를 파싱하여 데이터를 처리한다. 이 문제를 해결하기 위하여 리사이클러뷰에서 데이터 처리방식에 대해서 또 공부를 해야 했다.

개발 과정 3

소켓통신을 구현하기 위해 이클립스에서 구현하였다. 비교적 안드로이드 스튜디오도 비슷하게 소켓통신이 이루어졌으나, 액티비티가 존재하는 안드로이드 스튜디오에서 스레드를 어떻게 할당해야 할 지 고민했다. 결과적으로 비동기 함수 Asynctask함수를 이용하여 처리하기로 하였다.

문제발생

휠체어 이용자와 버스기사의 통신할 때 멀티스레드를 이용하여 통신하도록 하여야 했다. 휠체어 이용자가 전달한 데이터를 버스기사한테 보내는 통신 방식은 꽤 까다로웠다. 서버는 차량번호로 버스기사를 분하여야 했다. 서버는 데이터를 받고 또 송신하는 과정을 거치게 되면 서버는 너무많은 소켓을 소모한다.]

문제해결

버스기사는 서버에 소켓으로 연결된 상태를 유지하고 휠체어 이용자는 데이터를 전송만 하도록 하였다. 이런식으로 구현을 하면 서버가 굳이 버스기사를 구분할 필요가 없으며 송신을 위한 서버의 스레드를 절약할 수 있다.