<모바일응용 소프트웨어 설계>

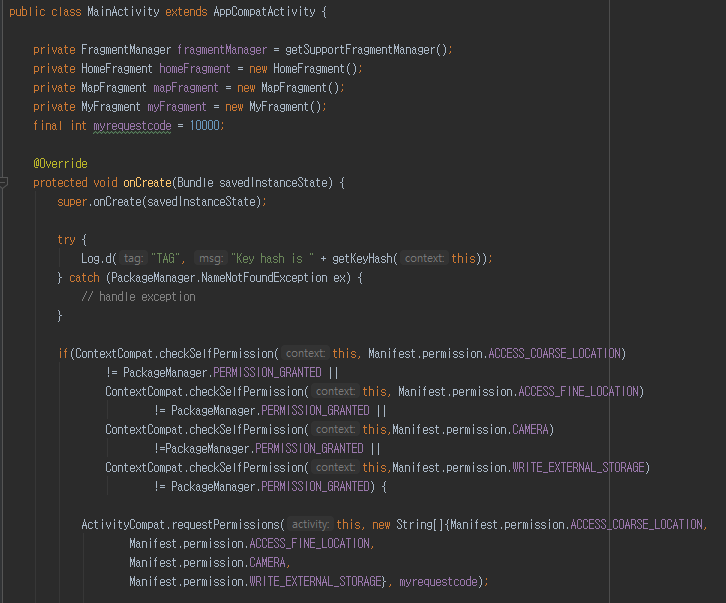
# Mini Project

정보통신공학과

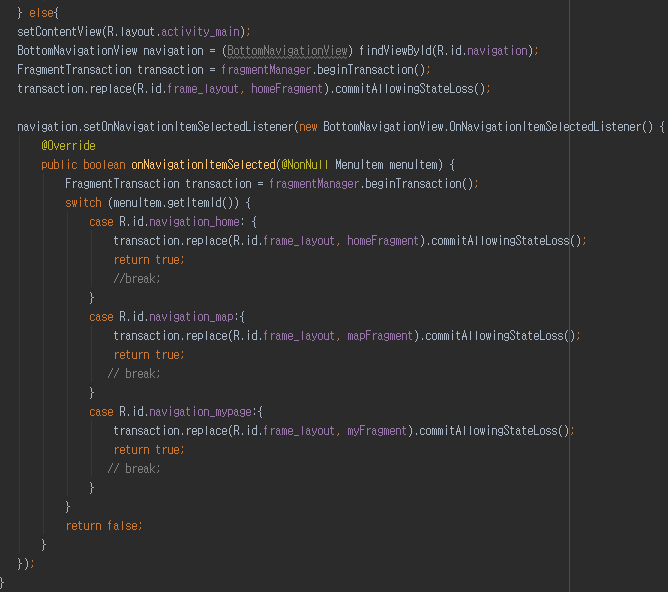
12161774 이 채 은

**[소스 코드]**

1. MainActivity.java



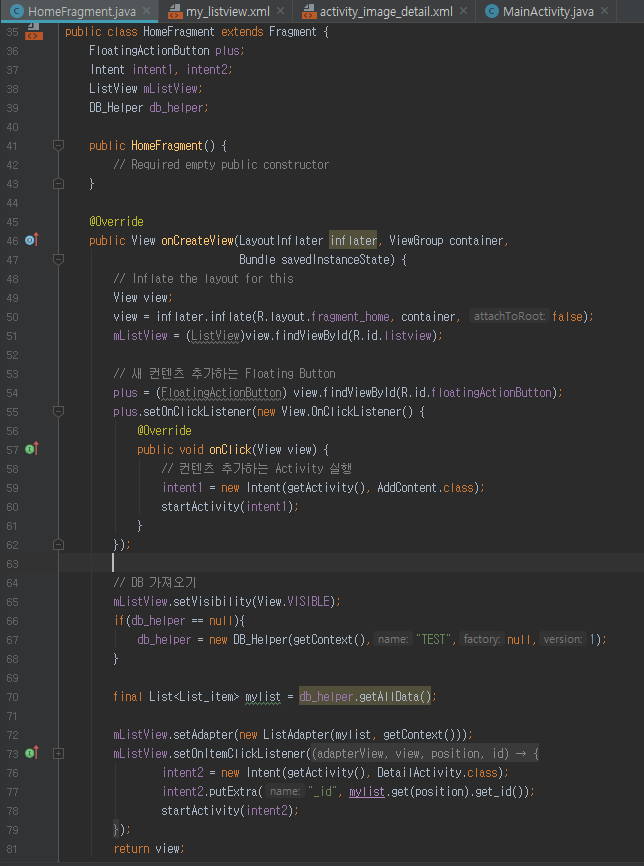
=> 세 개의 탭을 갖는 어플리케이션을 만들기 위해서 Bottom Navigation Activity를 이용했다. 따라서 MainActivity 내에 3개의 Fragment를 갖는다. 먼저 각 Fragment를 첫번째는 Home, 두번째는 Map, 세번째는 My Fragment로 선언해준다. 그리고 이 앱에서 위치, 카메라 등을 사용하기 때문에 run time permission을 하기 위해서 checkSelfPermission을 해서 각 permission을 런타임에 체크한 뒤 앱을 시작한다. 퍼미션이 수락되지 않으면 수락하도록 requestPermission을 한다.



=> 퍼미션이 모두 허락되면 setContentView로 레이아웃을 보여주고, bottomnavigationview를 이용해서 navigation을 정의한다. 그리고 Listener를 등록해서 각 탭이 navigation을 통해 페이지를 이동할 수 있도록 한다.

위의 코드에서 3-4번째 줄은 첫 화면을 HomeFragment 로 설정한 것이다. 따라서 맨 처음에 앱을 들어가면 Home 탭이 첫 화면에 뜨게 된다.

2. HomeFragment.java

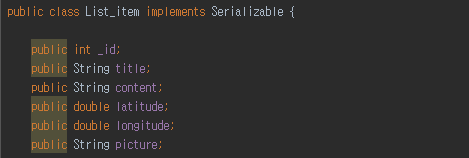


=> 첫번째 탭을 구현하는 HomeFragment이다. Fragment이기 때문에 OnCreateView에서 inflater를 통해서 view를 묶어준다. 그리고 화면에 Listview를 띄우기 위해서 layout에 ListView를 만들어주고, id(listview)로 받아와서 mListView 뷰를 선언해준다.

=> 각 화면에서는 Floating Button을 통해서 새 컨텐츠를 추가하도록 하기 위해서 layout에 floating button 또한 만들고 id(floatingActionButton)으로 받아왔다. 이 버튼을 누르면 새로운 Activity가 실행되고, 원하는 컨텐츠를 추가할 수 있다. 따라서 새로운 액티비티로 화면 전환을 하기 위해서 intent1으로 새로운 Intent를 만들고, startActivity(intent1) 코드로 AddContent라는 컨텐츠 추가 클래스로 넘어가도록 했다.

=> HomeFragment에서는 바로 리스트뷰를 보여주기 위해서 setVisibility를 설정해준다. 또한 리스트뷰의 내용을 채워 주기 위해서 데이터베이스에 저장되어 있는 정보를 이용한다. 따라서 정보를 가지고 오도록 도와주는 DB\_Helper의 객체 db\_helper를 이용한다. 이 객체가 비어 있으면 새로운 객체를 만들어준다. 그리고 Helper에 만들어진 getAllData() 함수를 불러와서 List<List\_item> 에 저장하고, 리스트를 mylist로 저장한다. 그리고 리스트 설정을 도와주는 ListAdapter를 통해서 setAdapter를 해준다. 설정된 리스트뷰의 각 아이템들은 onclick listener를 통해서 클릭이 가능하며, 아이템을 클릭하면 상세 정보를 알 수 있는 DetailActivity로 이동할 수 있다. 따라서 이 이벤트도 intent를 통해서 새로운 액티비티로 넘어갈 수 있도록 했다. 추가로 액티비티로 값까지 같이 전달하고 객체를 넘겨주는 방법으로 putExtra를 이용한다. 리스트의 position을 통해 각 아이템에 접근하고, get\_id()함수를 불러서 각 리스트의 고유 번호인 id를 가지고 새로운 액티비티로 넘어가서 선택된 아이템 고유의 정보를 확인할 수 있도록 했다.

3. List\_item.java

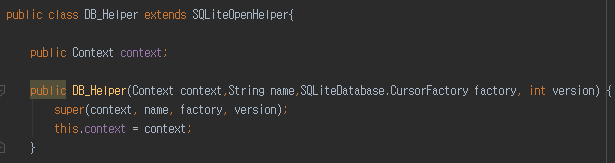


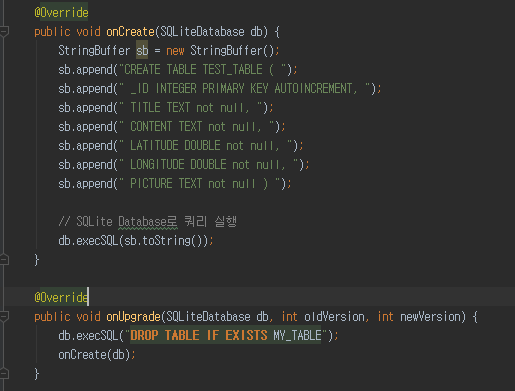


=> 데이터 베이스에 저장되고, 앱 내에서 쓰일 정보에 대한 getter, setter를 정의한 클래스이다. Int 형의 고유 \_id / String 형으로 저장되는 title, content / 위치를 나타내고 double 형으로 저장되는 latitude, longitude / string형으로 저장되는 사진 정보인 picture을 List\_item class 에 설정했다.

4. ListAdapter

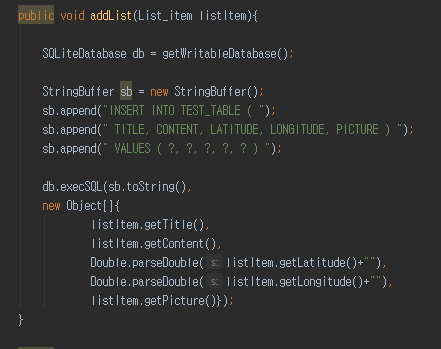
5. DB\_Helper.java



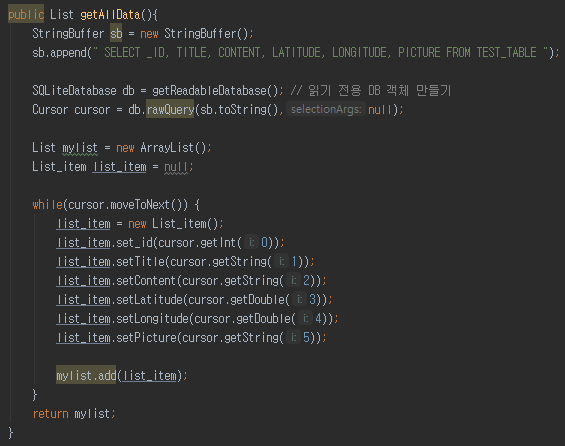


=> 기본적으로 override되는 onCreate와 onUpgrade 함수이다. onCreate 함수에서는 StringBuffer 를 만들고, 데이터베이스에 저장될 TABLE 명과 각 column의 이름, 변수형 등을 append해준다. 그 후 execSQL을 통해서 위의 StringBuffer에 저장된 퀴리문을 실행해준다.

=> onUpgrade 에서는 데이터베이스의 변화가 있을 때 실행되는 내용을 써준다. 만약에 TABLE이 존재하면 DROP 하고, 다시 onCreate로 TABLE을 새로 만든다.



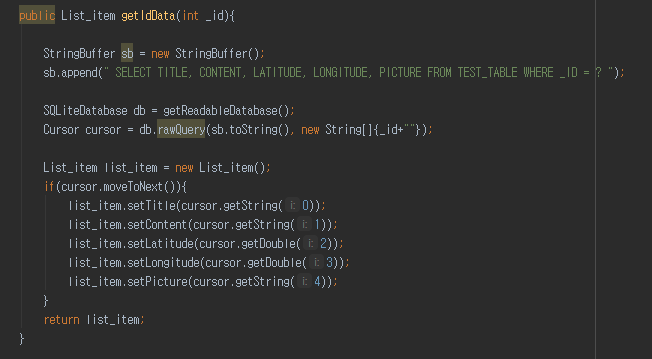
=> addList 함수이다. 마찬가지로 StringBuffer에 쿼리문을 넣어주고, execSQL을 통해서 쿼리를 실행한다. 테이블에 새로운 value를 넣어 주기 위한 함수이다. Object[]를 통해서 getter로 새로운 정보들을 순서대로 배열로 받는다.



=> 테이블의 모든 정보를 받아오는 쿼리를 실해하기 위한 함수이다. 새로운 List mylist 를 ArrayList로 만들어서 받아온 데이터를 담아준다. Cursor는 테이블의 각 열을 가리키는데 while문을 통해서 커서가 처음부터 끝까지 moveToNext하면서 각 열의 데이터 값을 가져온다. List\_item 객체 list\_item으로 cursor가 가리키는 값을 getter를 통해서 받고, mylist에 하나씩 add해준다.



=> ImageData를 가져오기 위한 쿼리를 실행하는 함수이다. 위의 함수들과 같이 list를 만들어서 list\_item과 cursor로 각 열에 접근해서 getter로 값을 받아온다.

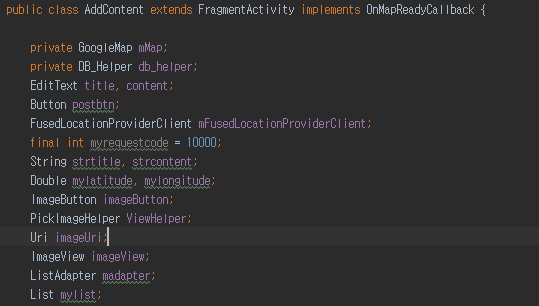


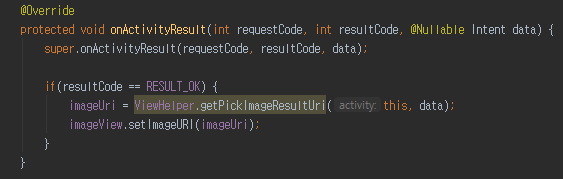
=> Id 에 따른 데이터를 받아 오기 위한 쿼리를 실행하는 함수이다. SELECT에서 WHERE 조건절을 추가해서 \_ID의 값이 전달받은 id인 열의 데이터를 받는다. rawQuery를 이용해서 조건에 맞는 데이터를 select한다.



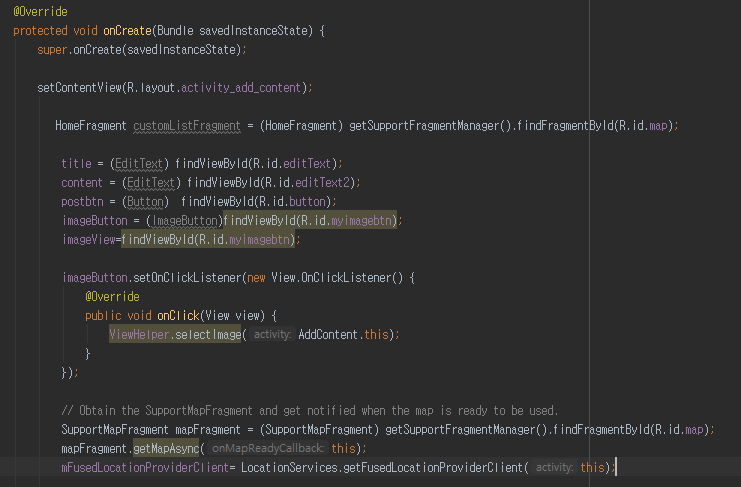
=> Id 에 따른 데이터를 지우기 위한 쿼리를 실행하는 함수이다. execSQL을통해서 해당 열을 DELETE를 한다.

6. AddContent.java

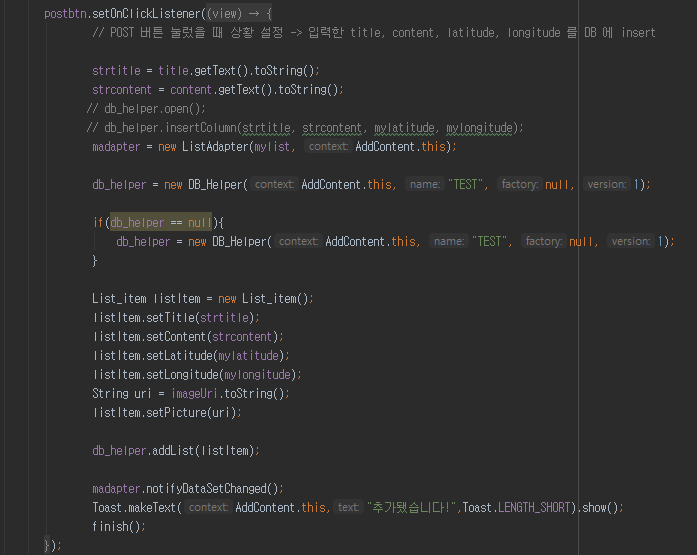




=> 플로팅 버튼을 눌렀을 때 실행되는 addcontent 액티비티에 대한 내용이다. PickImageHelper의 객체를 ViewHelper로 선언 한 뒤 그 클래스의 함수를 이용해서 imageUri를 받아온다. PickImageHelper의 getPickImageResultUri 함수를 통해서 갤러리와 카메라 중 사진을 선택할 수 있도록 한다. 그리고 선택한 이미지의 uri를 받아와서 imageView에 setImageURI로 이미지를 uri를 이용해 보이게 한다.

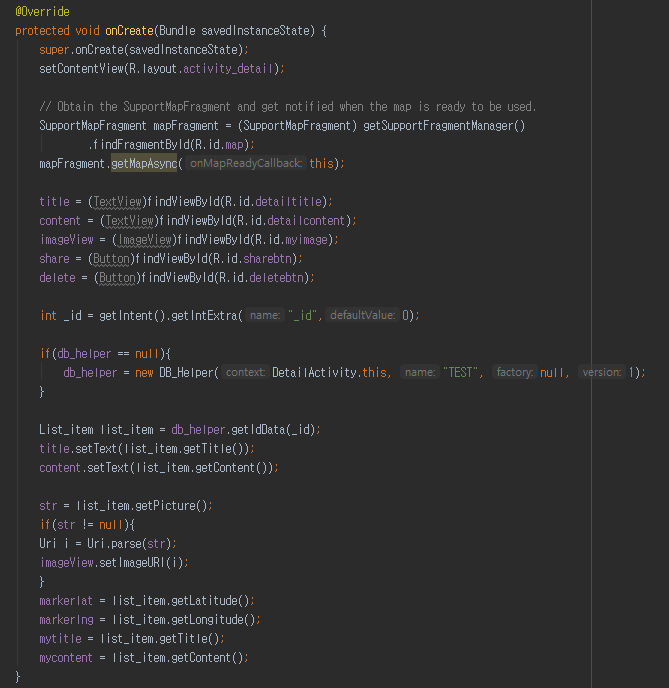


=> imageButton을 클릭하면 ViewHelper 객체를 통해 PickImageHelper의 selectImage로 넘어가서 이미지를 고를 수 있는 intent를 실행할 수 있도록 한다.



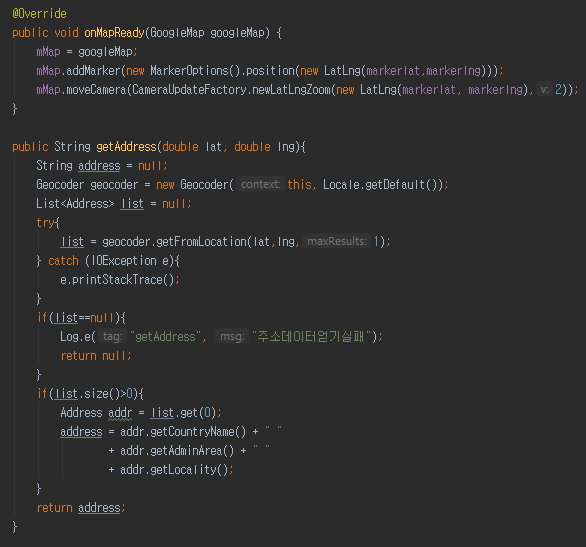
=> POST 버튼에 대한 내용을 구현한다. 입력해준 title, content, 위치, 이미지를 새로 추가해주는 결과를 얻어야한다. 따라서 db\_helper를 통해서 데이터베이스에 접근하고 각 데이터를 setter를 통해서 저장해준다. 그리고 위의 set해준 데이터를 addList를 통해서 새로 추가해준다.

7. DetailActivity.java

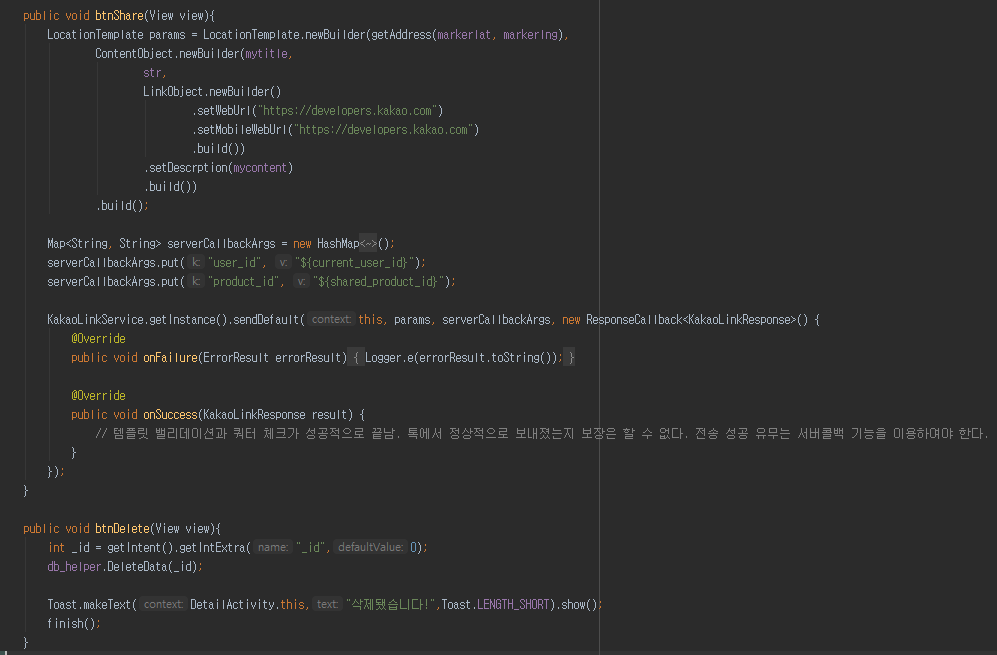


=> 첫번째 탭 리스트 아이템, 두번째 탭에서 마커를 클릭하면 이동하는 새로운 액티비티에 대한 내용이다. 선택한 아이템이나 마커에 대한 정보를 확인할 수 있는 상세 페이지이다. getIntent를 통해서 \_id값으로 IntentExtra를 넘겨받는다. 마찬가지로 db\_helper를 이용해서 데이터베이스에 접근하고 getter들을 이용해서 값을 얻어온다. 얻어온 값들은 layout에 만들어진 각 view들에 나타나게 된다.

이 때 사진은 string 이기 때문에 parse를 이용해서 uri로 바꿔서 이미지를 한번에 나타낼 수 있도록 한다.

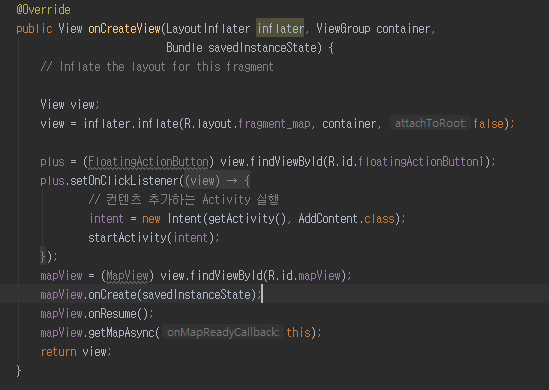


=> 선택한 위치에 대한 내용을 지도에서 바로 보여주기 위해서 GoogleMap을 불러온다. onMapReady를 이용해서 받아온 주소 markerlat, marketlng의 위치에 마커를 추가해서 보여준다. 또한 String getAddress 메소드에서는 지오 코더를 이용해서 얻은 latitude, longitude의 값을 주소로 변경해준다. 이는 나중에 지도를 통해 위치를 공유 할 때 쓰기 위함이다.

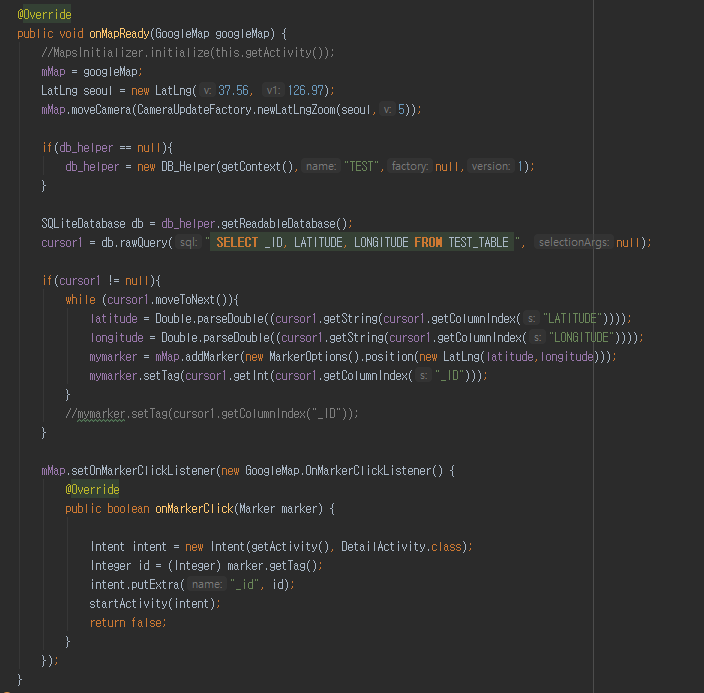


=> SHARE 버튼에 대한 내용이다. 카카오톡 링크를 통해서 타이틀과 내용, 위치를 공유해준다. 이 때 사용되는 값 또한 위에서 getter를 통해서 데이베이스에서 받아온 내용이다.

8. MapFragment.java



=> 플로팅 버튼을 누르면 intent를 통해서 컨텐츠 추가 액티비티로 이동한다.



=> 두번째 탭인 맵 프래그먼트를 누르면 rawQuery를 실행해서 테이블의 latitude, longitude를 얻어오고 cursor.moveToNext를 이용해서 각 열별로 위치를 받아와서 마커를 추가한다. 또한 마커에 setTag를 이용해서 ID를 설정해준다. 그리고 그 아이디를 통해서 마커를 클릭하면 id에 따라서 고유 값을 가지고 상세페이지로 가서 고유 값들을 보여준다.

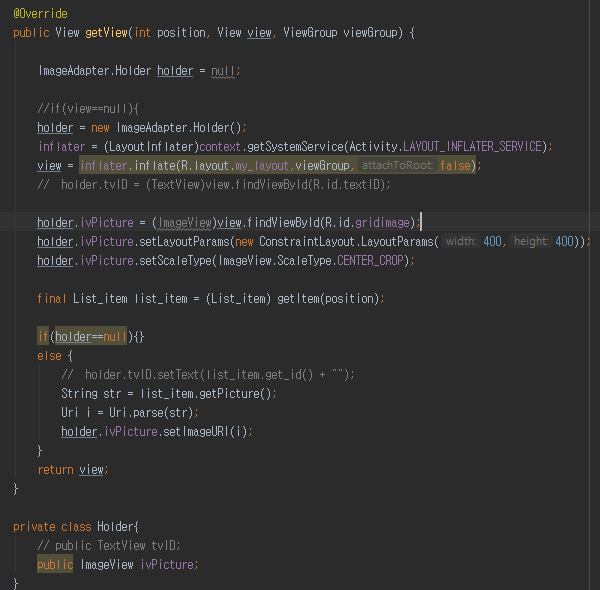
9. MyFragment.java



=> 세번째 탭은 갤러리 형식으로 그리드뷰를 보여주는 것으로 만들었다. gridView를 만들어주고, List를 만들어서 db\_helper를 이용해 데이터베이스에서 Image를 불러온다. 그리고 그리드 뷰에 하나씩 사진을 보여준다.

=> 각 이미지를 클릭하면 이미지에 대한 확대 이미지와 제목, 내용을 보여주는 새로운 ImageDetail 클래스로 intent를 통해서 이동한다.

10. ImageAdapter.java



=> 그리드뷰를 위해서 어댑터를 만들어준다. 리스트뷰와 비슷한 원리로 만들었다.

11. ImageDetail.java



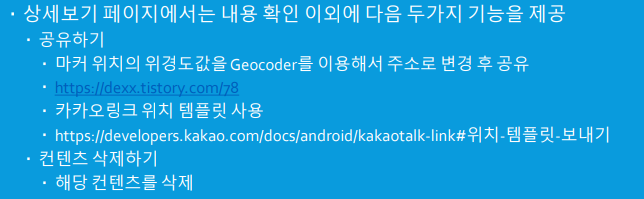
=> 이미지를 선택하면 이미지를 크게 보여주고, 타이틀과 컨텐츠를 확인한다.

**[프로젝트 내용 및 고찰]**





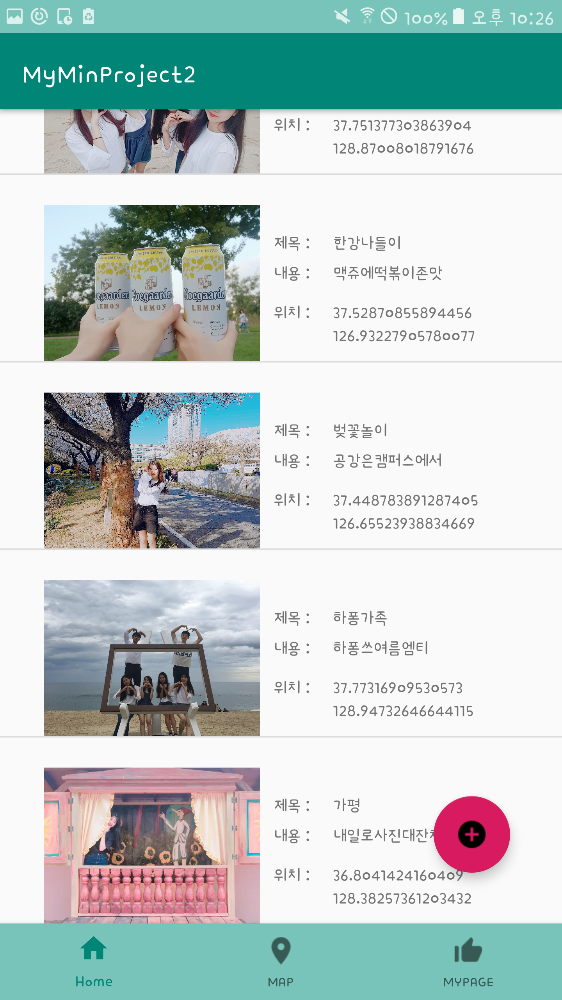




=> 먼저 application을 구동 시킨 환경은 Galaxy Note5 (Android 6.0.1, API23) 입니다. 에뮬레이터가 노트북에서 제대로 작동이 되지 않아서 앱 구동과 테스트를 모두 위의 Device로 진행했습니다.

=> 각 탭을 구현하는 방법과 객체를 넘기는 방법을 알게 됐습니다. 또한 데이터베이스와 앱을 연동하는 방법을 공부할 수 있었습니다. 또한 구글맵을 이용하고, 카카오링크를 통해서 공유하는 방법도 익혔습니다. 리스트뷰와 그리드뷰 등 커스텀 뷰에 대해서도 알게 됐습니다.

=> 아래는 직접 구현한 화면을 캡쳐한 것입니다.



순서대로 첫번째, 두번째, 세번째 탭이다.



순서대로 새로운 컨텐츠를 추가하는 페이지, 그리드뷰에서 이미지를 선택했을 때 확대되는 페이지, 상세보기 페이지입니다.

