2024년 1학기

소프트웨어프로젝트1

2차 프로젝트

컴퓨터정보공학부

2021202039

이소영

1. Introduction

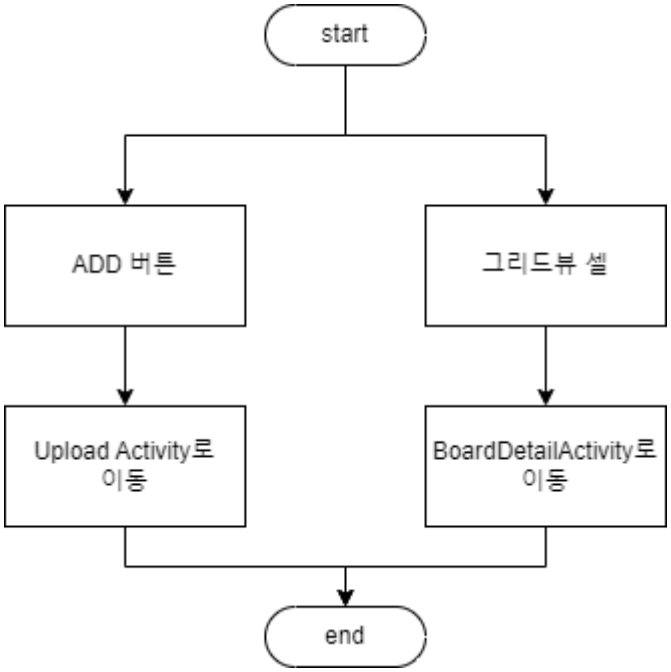
본 프로젝트는 지난 1차 프로젝트에서 나아가 안드로이드 스튜디오와 스프링부트를 이용하여 사진 게시판 애플리케이션을 구현하는 것이다.

사진을 업로드하고 해당 사진에 대한 간단한 소개를 작성하는 것으로, 사진 관련 타이틀 및 글을 작성 후 업로드하고, 갤러리 형식으로 사진을 나열하는 프로젝트이다.

안드로이드 애플리케이션으로, JAVA를 사용하여 구현하고 스프링 및 JSON을 활용하여 데이터 통신을 하여 서버와 안드로이드 간의 통신을 이해한다.

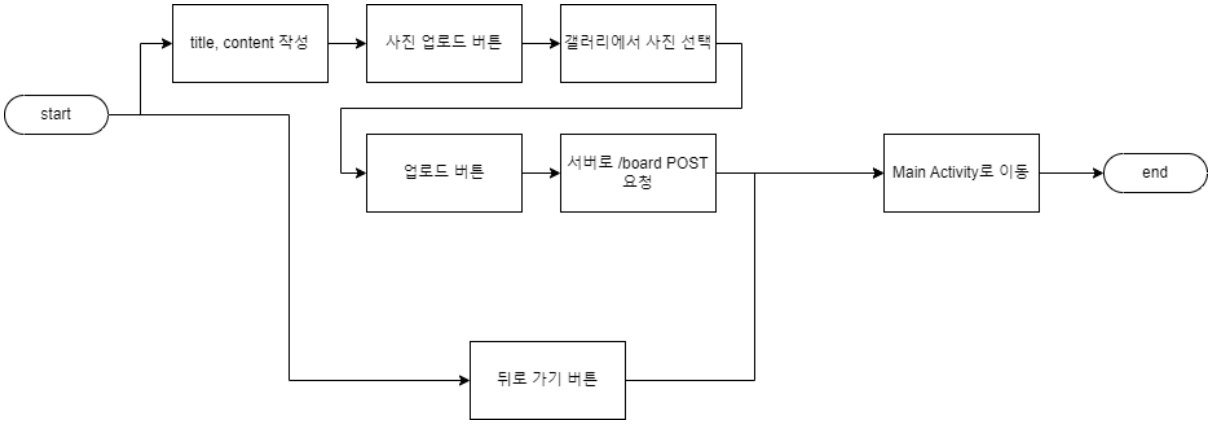
2. Flow chart

MainActiviy에서 ADD 버튼을 클릭하면 업로드 기능으로, 각 그리드뷰의 사진을 클릭하면 상세 조회 기능으로 이동한다.

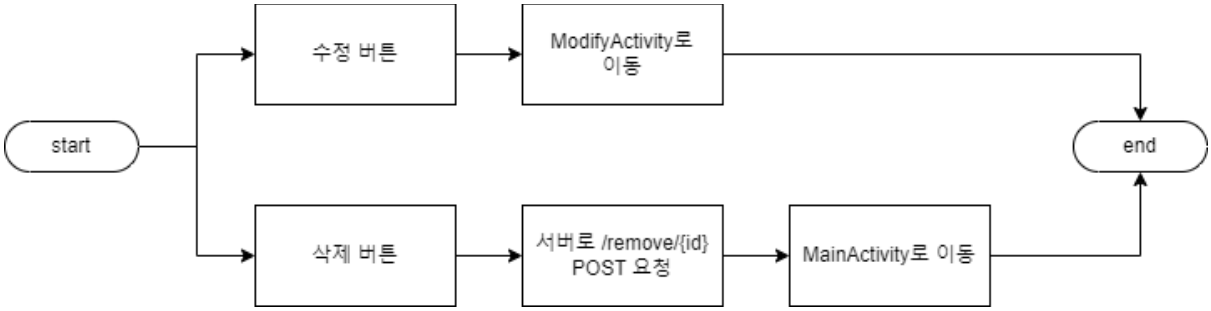


UploadActivity에서는 업로드 기능이 구현되어 있어, 사용자가 입력한 title, content, image 데이터를 스프링 서버로 /board POST로 요청한다.

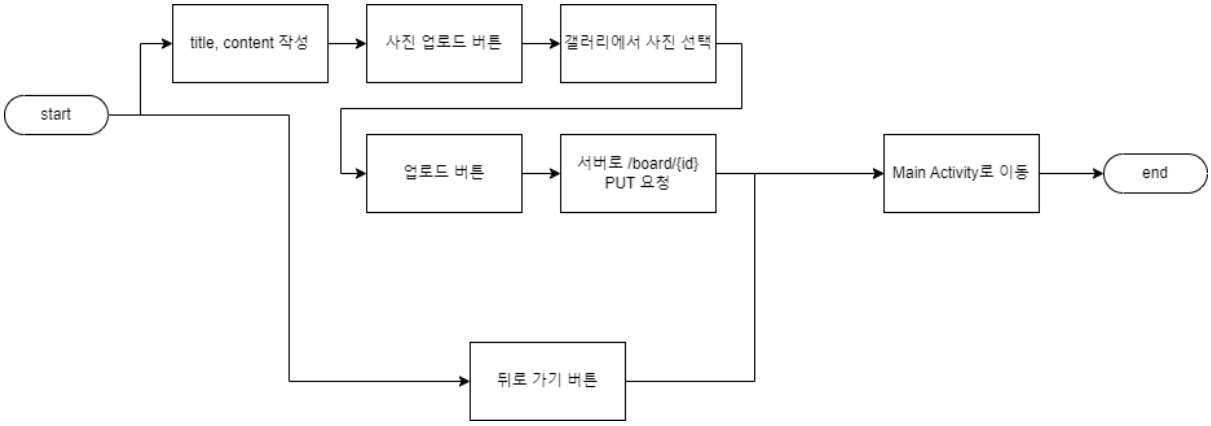
뒤로 가기 버튼, 업로드 버튼 모두 동작 후 MainActivitiy로 이동한다.



BoardDetailActivity에서는 MainActivity에서 전달 받은 Board 객체 정보를 출력할 뿐만 아니라 수정, 삭제 버튼을 제공한다.



ModifyActivity는 UploadActivity와 거의 동일하지만, 서버로 /board/{id} PUT 요청을 보낸다는 점에서 차이가 있다.

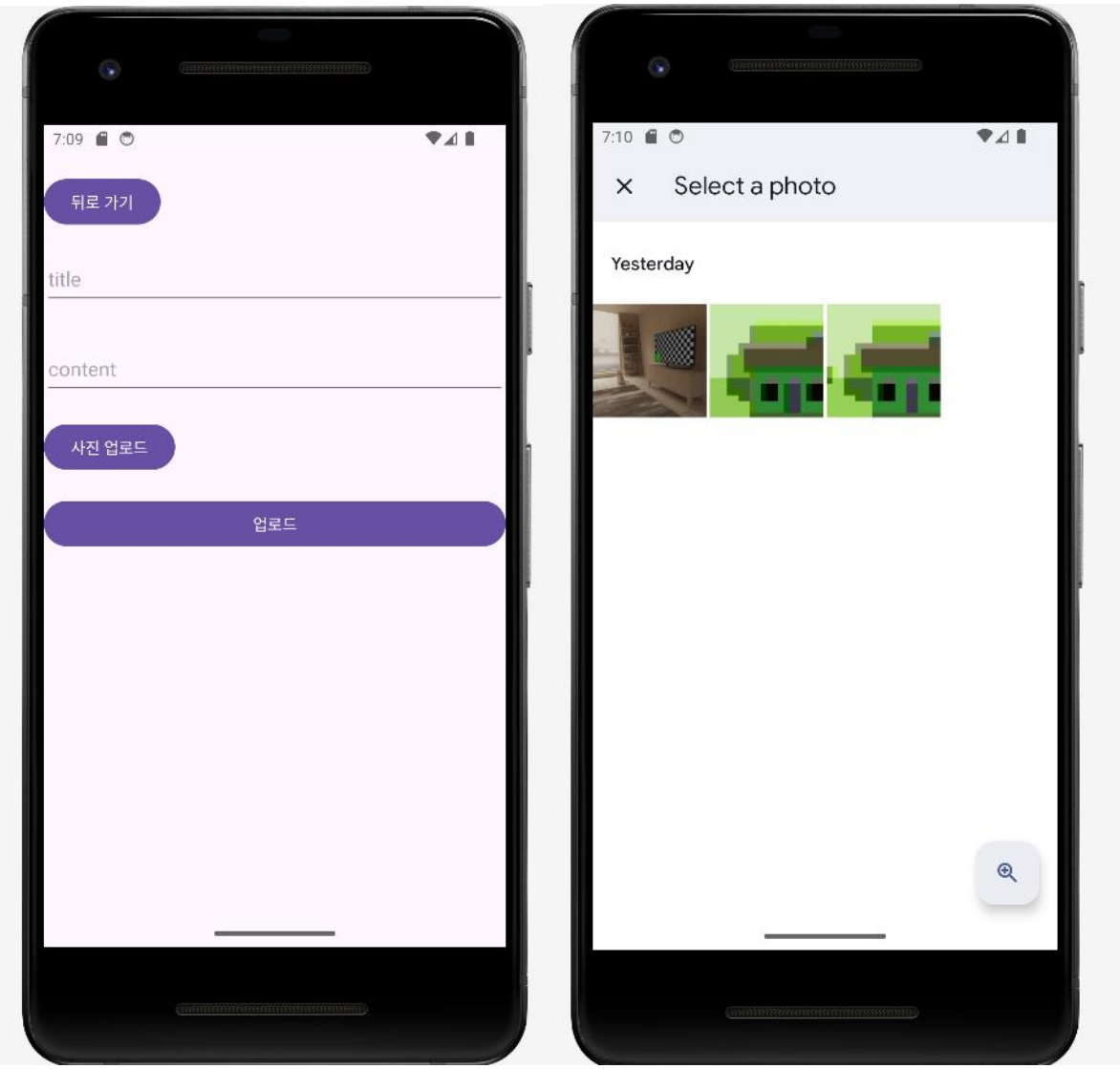


3. Result

시작 화면은 다음과 같이 학번 등의 정보와 업로드를 위한 Add 버튼으로 구성되어 있다. 아직 업로드한 사진이 없으므로 표시된 사진은 없다.

ADD 버튼을 클릭하면 업로드 뷰로 이동한다. 여기서 뒤로 가기 버튼을 클릭하면, 다시 이전 화면으로 돌아간다. title, content를 작성 후 사진 업로드 버튼을 클릭하면 갤러리에서 사진을 선택할 수 있다.

이는 ADD 버튼에 설정된 리스너에 따라, Upload Activity로 이동하는 것이다. title, content를 적은 후 사진 업로드 버튼을 클릭하면 설정된 리스너에 따라, 애플리케이션 외부 즉 갤러리에서 이미지를 가져오도록 한다. 이는 openGallery() 메서드로 구현되어 있다.

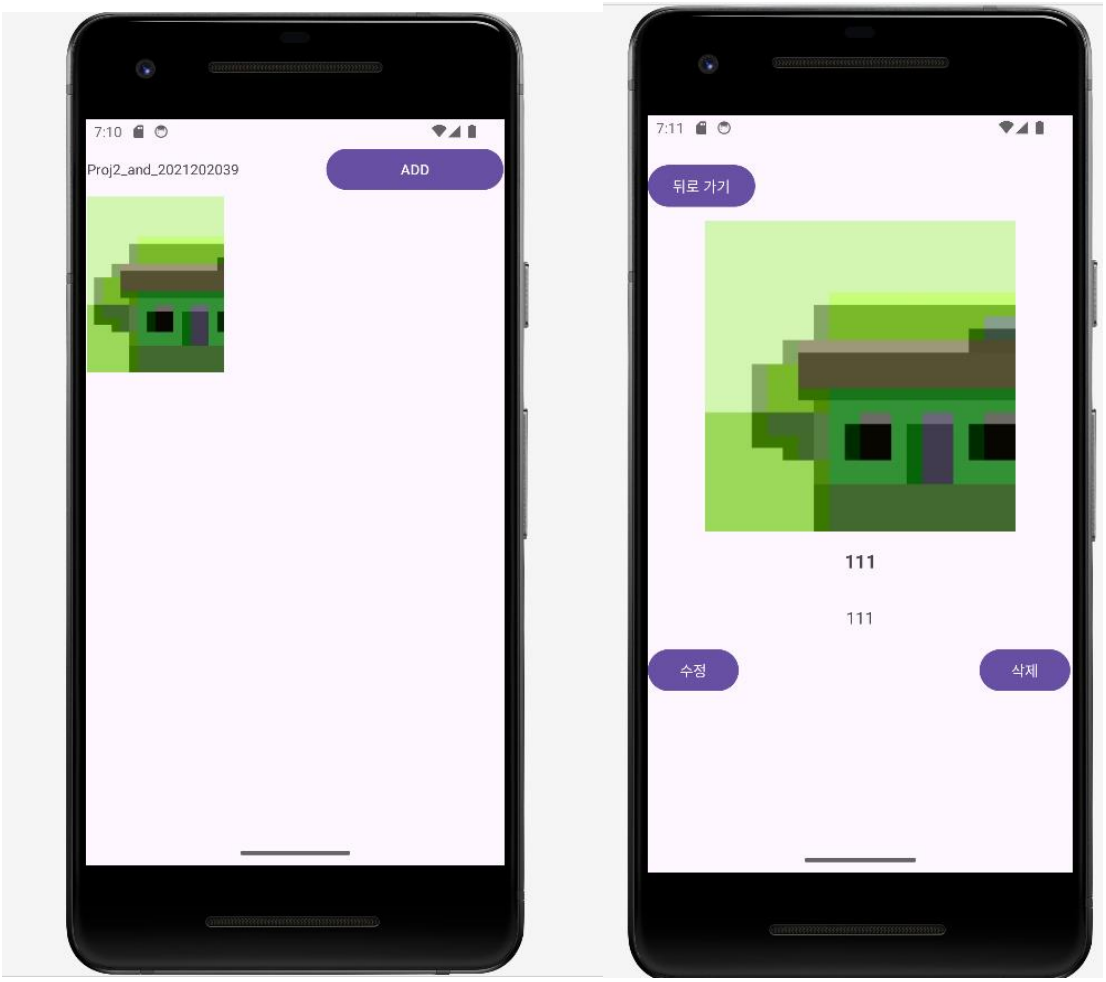


업로드 버튼을 누르면 홈 화면으로 돌아온다. 사진이 추가된 것을 확인할 수 있다. 해당 사진을 클릭하면 사진이 확대되며 이전에 작성한 title, content를 확인할 수 있다.

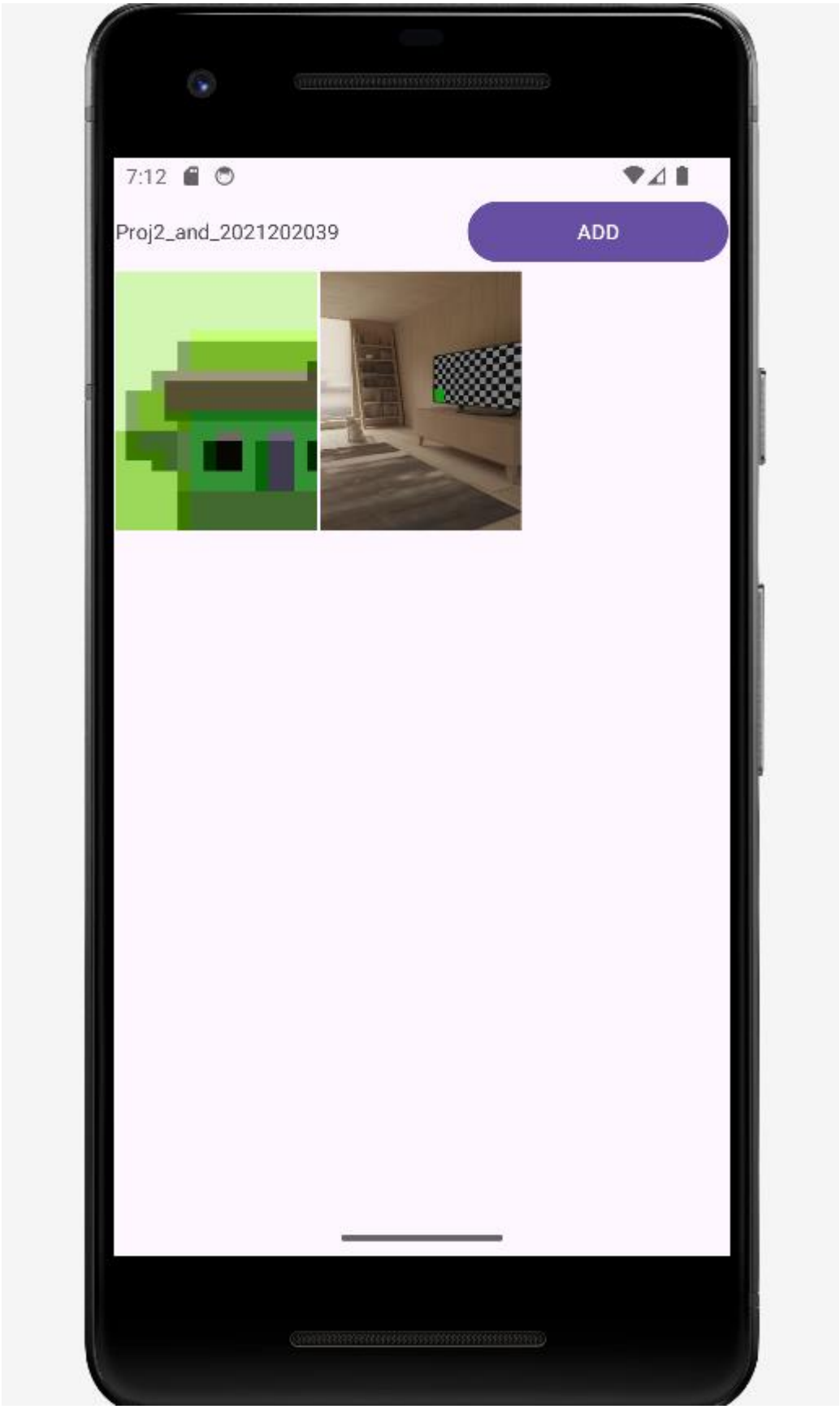
이때 업로드는 서버로 /board POST 요청을 보내며, UploadActivity.java의 UploadTask()에서 처리한다. 업로드 후에는 홈 화면으로 돌아가야 하므로 인텐트를 이용하여 Main Activity로 돌아간다.

Main Activity에서 각 그리드뷰 셀에는 onItemClick()을 이용하여 각 Board 객체가 연결되어 있고, 인텐트를 이용하여 각 데이터를 확인할 수 있는 BoardDetailActivitiy로 이동하게 된다.

이동한 상세 정보에서 뒤로 가기를 누르면 인텐트를 통해, Main Acitivity로 돌아간다.

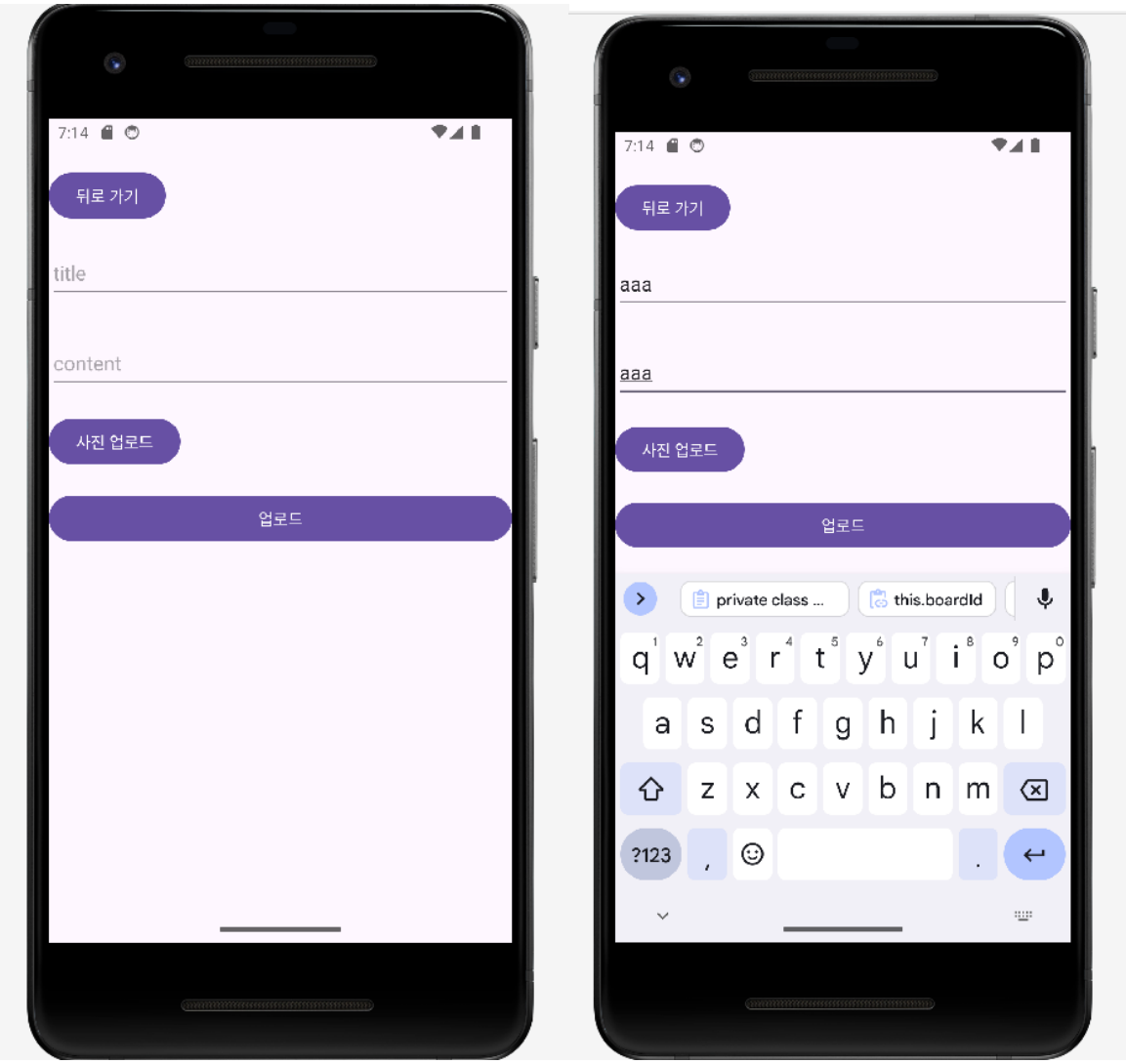


다른 사진을 추가한 모습이다. 이로써 업로드 기능이 정상적으로 동작함을 확인할 수 있다.

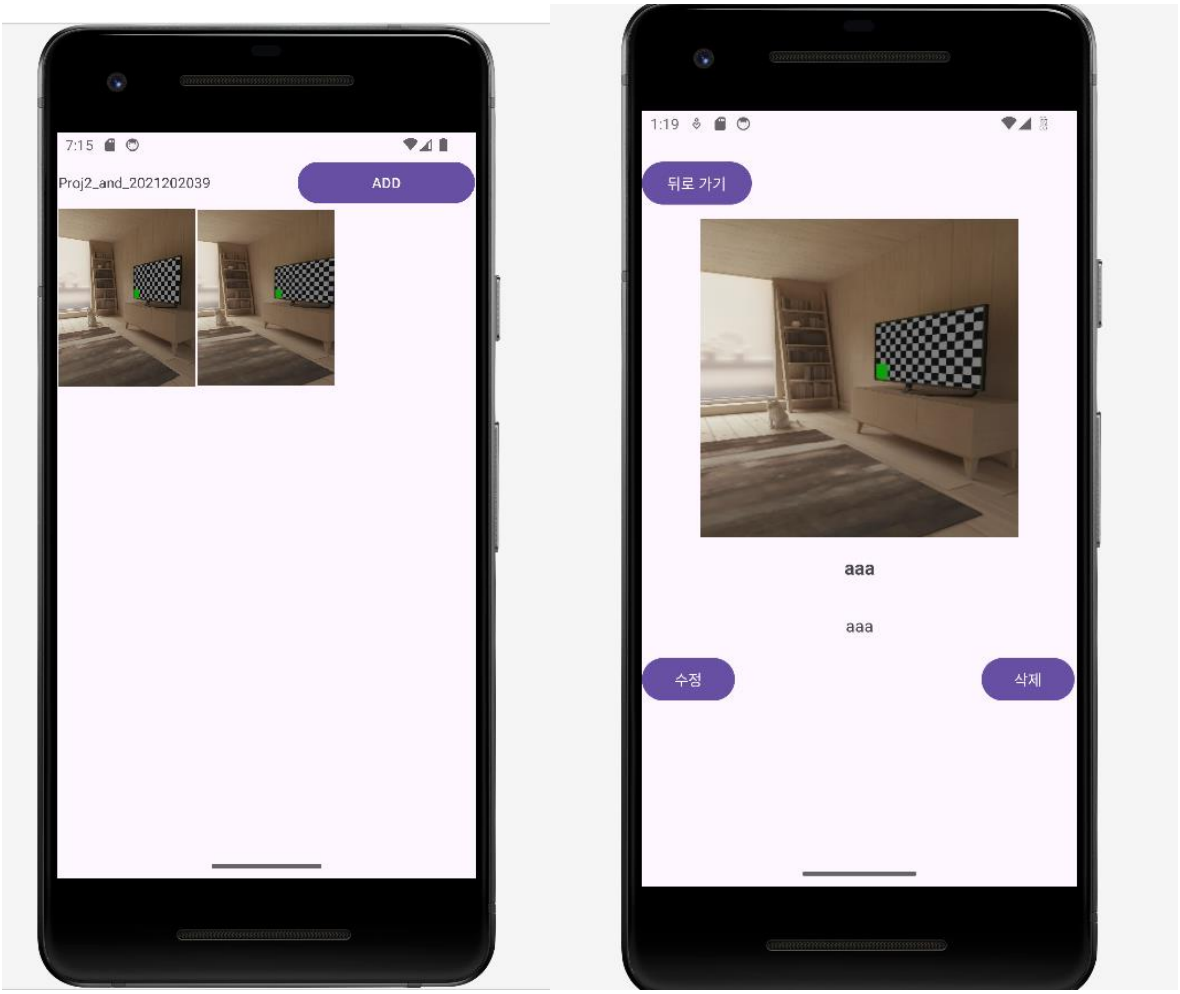

7:12
Proj2_and_2021202039
ADD

처음 업로드한 사진을 클릭한 후 수정을 누르면 업로드 페이지와 같은 페이지를 볼 수 있다. 수정 기능은 BoardDetailAcitivity에서 ModifiyActivity로 해당 Board(이미지, 제목, 내용 데이터) 객체를 가지고 연결된다. UploadActivity와 똑같이 생겼지만 다른 액티비티이다.

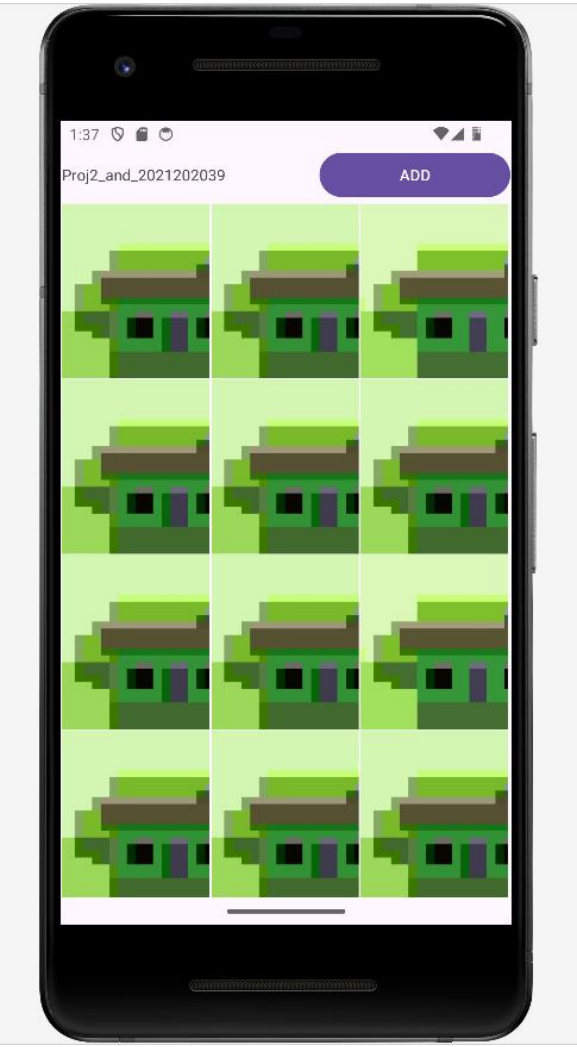
업로드 버튼을 누르면 ModifyActivity에서 ModifyBoardTask()로 서버에게 수정 요청을 한다. 이는 /board/{id} PUT 요청이다. 서버에서는 id를 받아 데이터베이스에 저장된 Board 객체를 조회하고 수정한다.



이미지와 제목, 내용이 모두 수정되었음을 확인할 수 있다.

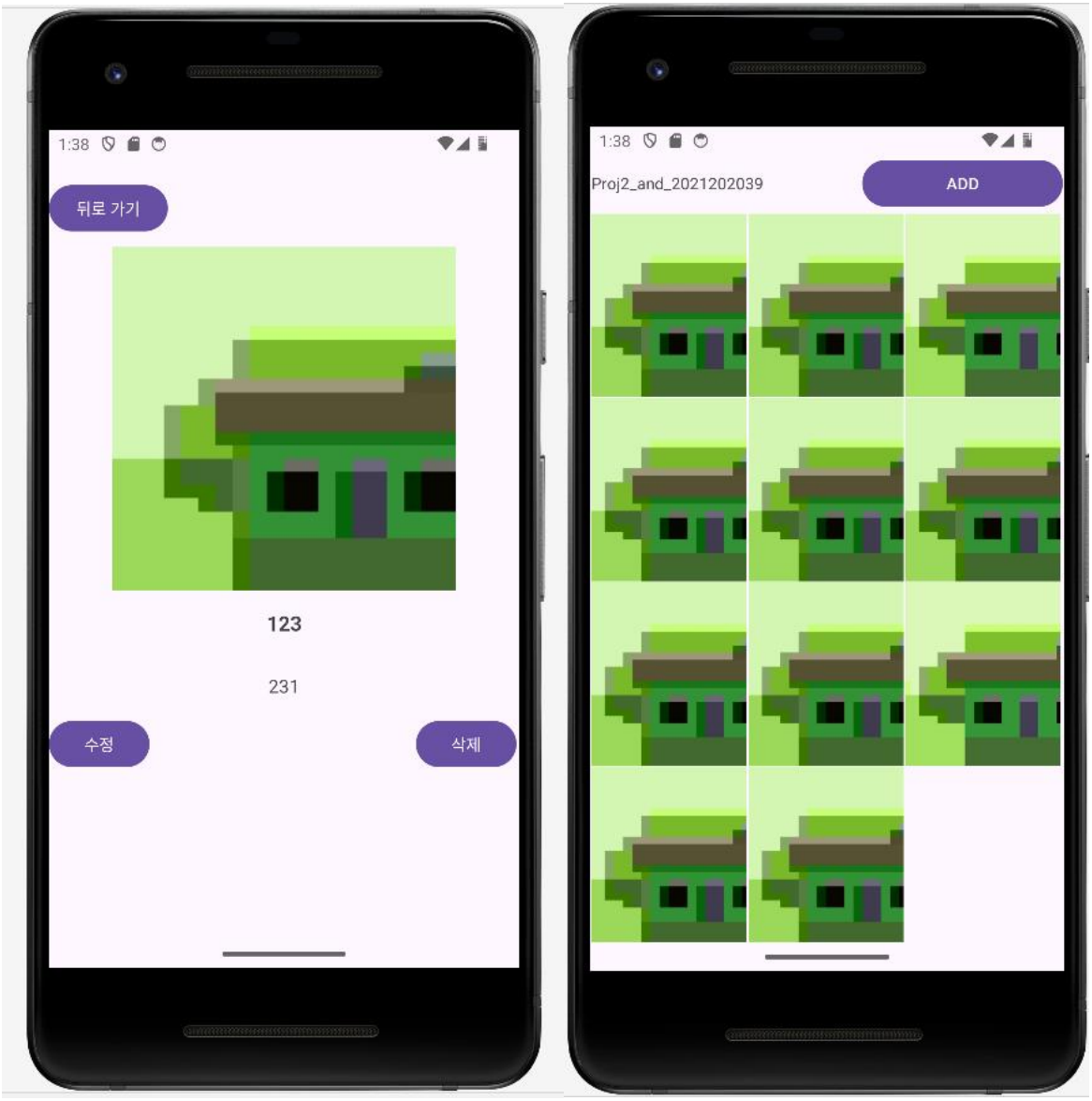


홈 화면에서 갤러리는 3*4 크기의 그리드뷰이다. MyGridAdapert에서 setLayoutParams()를 통해 각 셀 크기를 조정하였다.



한 사진을 선택하고 삭제 버튼을 확인할 수 있다. 이 버튼은 BoradDetailActivity에서 DeleteBoardTask()를 하도록 리스너로 설정되어 있다. DeleteBoardTask()는 서버로 /remove/{id} POST 요청을 보낸다. 서버에서는 id로 Board 객체를 찾아 데이터베이스에서 삭제한다.

갤러리에서 사진이 삭제된 것을 확인할 수 있다.



4. Consideration

안드로이드 스튜디오를 처음 사용해봤기에 몇몇 어려운 점이 있었다. 강의자료에서는 AsyncTask를 이용하여 서버와 통신하였는데, 이를 구현하기 위해 자료조사한 결과 Retrofit2에 대한 자료가 훨씬 많아 AsyncTask를 이용하는 데에 어려움을 겪었다. 그리고 String만 보내는 것이 아니고, 이미지 파일을 전송해야 하기 때문에 일반적인 JSON 데이터가 아니라 form-data 형식으로 전송해야 했다. 이는 MultiPart를 이용하여 해결했다.

그리드뷰에서 각 셀의 비율을 조절하는 속성이 있을 것이라 생각하고 설계했는데 실제로 그런 속성을 찾지 못했다. 3*4 크기로 조절하기 위해서 어댑터에서 setLayoutParams()에 적당한 크기의 너비와 높이 값을 입력하여 3*4에 근접하도록 조절하였다.

실행 중 갑자기 예상치 못한 오류가 발생하는 경우가 종종 있었는데 로그 기록을 살펴 보니, java.lang.RuntimeException: Could not copy bitmap to parcel blob. 오류가 발생했었다. 안드로이드에서 스프링 서버로 또는 액티비티 간에 Board 객체를 주고 받을 때 Board class를 사용하였는데 이때 parcel을 사용하였다. 안드로이드 플랫폼에서 가장 효율적인 데이터 전달 방법이기 때문이다. 이미지를 경로가 아니라 비트맵으로 변환하여 저장하였기 때문에 길이가 길어졌고, 비트맵 객체를 Parcel로 복사하는 과정에서 비트맵 객체가 너무 커서 Parcel이 처리하지 못한 오류로 보았다. 따라서 Board 객체를 생성할 때 이미지의 크기를 조절했다. 아래가 그 경우 중 하나에 해당한다. UploadTask로 서버에 보내기 전, 비트맵 사이즈를 조절한 값으로 넣었다.

```
if (selectedImageBitmap != null && !title.isEmpty() && !content.isEmpty()) {
    Bitmap resizedBitmap = Bitmap.createScaledBitmap(selectedImageBitmap, dstWidth: 300, dstHeight: 300, filter: true);
    new UploadTask(title, content, resizedBitmap).execute();
```

이를 통해 문제를 해결할 수 있었고, 원활히 작동하였다.