소프트웨어프로젝트1 Report

Project #2

담당교수: 이우신 교수님

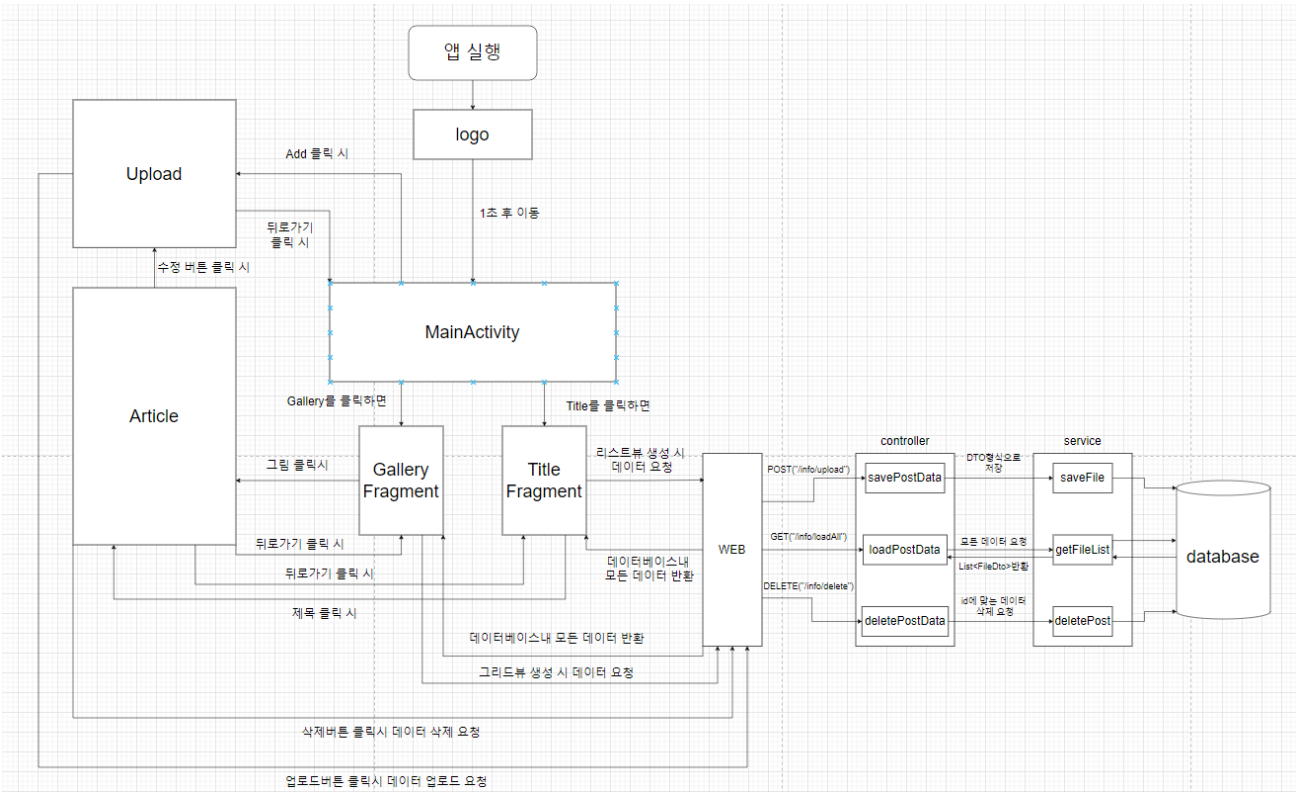
학번: 2019202050

이름: 이강현

[Introduction]

이번 프로젝트에서는 안드로이드에서 제목,글,사진으로 구성된 서버의 데이터베이스에 존재하는 게시물을 그리드뷰 혹은 리스트뷰로 볼 수 있다. 안드로이드는 게시물을 직접 데이터베이스에 업로드할 수 있고 이를 통해 생성된 게시물은 서버의 database에서 관리되며 게시물은 수정과 삭제가 가능하다. 스프링 서버는 안드로이드에서 요청하는 http method에 맞게 데이터를 json형식으로 보내줄 수 있고 이를 안드로이드에서 파싱하여 사용한다. bottomnavigationview를 통한 fragment변환과 액티비티간 intent를 이용한 정보교환, 서버와 데이터를 주고받는 방법등 안드로이드 동작원리를 잘 이해하는 것이 과제에 핵심이다.

[Flow Chart]

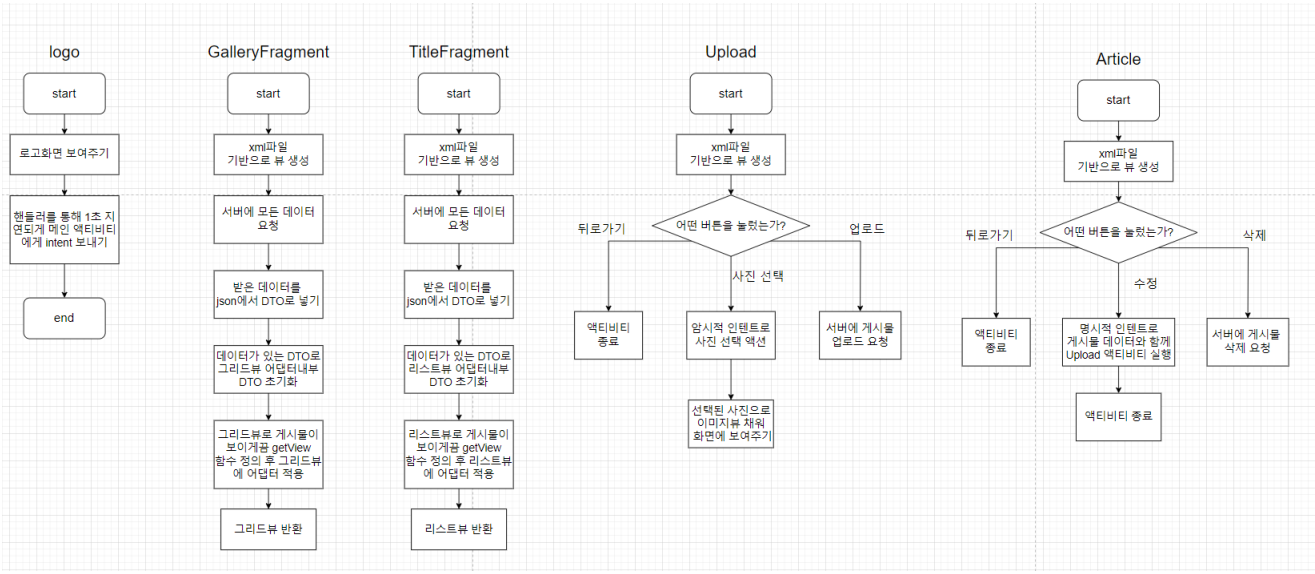


위는 전체적인 흐름도이다. 앱을 실행하면 logo가 뜨고 핸들러를 이용하여 1초지연을 주고 메인 액티비티를 인텐트로 실행하게끔 구현하였다. manifest를 수정하여 앱을 실행할때 제일 먼저 실행되는 액티비티를 로고로 바꿔 구현하였다.

메인 액티비티가 실행되면 아래 네비게이션뷰의 Gallery와 Title부분을 클릭하면 동작하는 리스너가 구현되어 있다. Gallery를 클릭할 경우 GalleryFragment가 화면에 보여지고 Title을 클릭하면 TitleFragment가 보여진다.(기본값은 GalleryFragment이다.) 또 메인 액티비티에 존재하는 Add버튼을 클릭하게 되면 게시물을 업로드할 수 있는

Upload액티비티가 실행된다. 그리고 각각의 Fragment에서 게시물을 클릭하면 Article 액티비티가 실행되며 인텐트를 통해 번들에 게시물 데이터가 함께 보내지기 때문에 Article액티비티에서는 게시물의 수정과 삭제가 가능하다. 수정시에는 폼이 필요하기 때문에 Upload액티비티에 게시물 데이터를 번들로 함께 넘겨주는 방식으로 구현했다.

각 요소들이 서버에 데이터를 필요해하는 경우엔 OkHttp방식으로 데이터를 json형식으로 변환해 스프링 서버와 통신하도록 구현했다.



위는 각 view들의 세부 흐름도이다. logo는 실행시 화면을 보여주고 1초후 메인액티비티를 실행하고 GalleryFragment와 TitleFragment는 그리드뷰로 게시물 데이터를 표현하느냐 리스트뷰로 게시물 데이터를 표현하느냐의 차이지 서버에서 모든 게시물 데이터를 요청해 받아오고 그걸 보여주는 흐름은 동일하다. Upload는 데이터를 입력할 수 있는 폼형태를 xml에 정의해두었고 이를 바탕으로 여러 버튼이 클릭되는 것에 따라 동작흐름이 달라진다. Article또한 Upload와 비슷한 흐름을 가지되 게시물을 보여주고 게시물의 수정 및 삭제가 메인동작이므로 약간 차이점이 존재한다. 표시되지않은 리스너또한 존재하나 이는 start이후 클릭이라는 입력이 언제 들어올지 모르기에 전체적인 흐름만 설명하고 리스너에 의해 변화되는 실행 흐름은 전체 흐름도에 표기해두었다.

[Result]

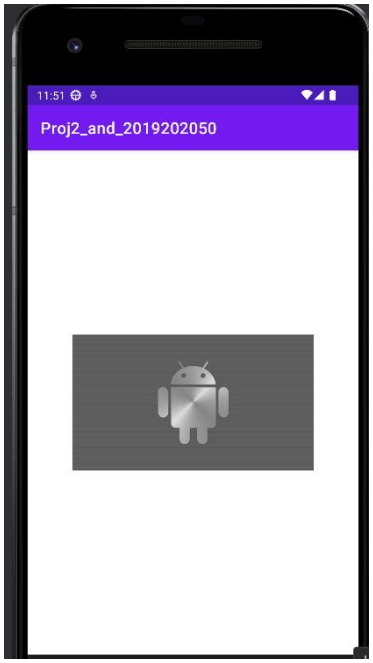
프로젝트를 MVC pattern에 맞게 설명하면 view와 Controller의 역할을 Activity, Fragment에서 수행하고 model의 역할은 서버의 데이터와 DTO와 같은 것이다.

따라서 안드로이드에서 메인 액티비티는 Fragment를 사용자의 클릭에 따라 적절히 보여주어야 하기에 Controller의 역할을 하고 Fragment는 액티비티에 의해 사용자에

게 보여지기 때문에 view의 역할을 하기도 하고 Fragment내의 Adapter는 model에 해당하는 서버내 데이터베이스로부터 데이터를 가져와 뷰를 형성하고 이를 반환해 보여주기도 하기때문에 controller의 역할도 존재한다. 메인 액티비티가 아닌 다른 액티비티도 마찬가지이다. Upload 액티비티는 xml파일로 폼을 보여주기 때문에 View의 역할이 있고 입력받은 데이터를 DTO로 저장하여 서버로 전송하기 위한 로직이 존재하는 model의 역할이 있다.

스프링 서버에서는 안드로이드의 다양한 요청에 따라 Controller가 http method를 처리하고 h2 database를 사용하여 데이터를 저장하고 요청에 따라 데이터를 저장하고 보내주는 model부분이 있고 이번 프로젝트에서는 안드로이드에서 사용자에게 게시물이 보여지기 때문에 서버입장에서는 안드로이드가 view의 역할을 한다고 볼 수 있다.

앱의 실행부터 게시물이 보여지는 화면과 게시물의 생성,수정,삭제가 잘 되는지 예외사항까지 순서대로 진행하며 결과를 검증한다.



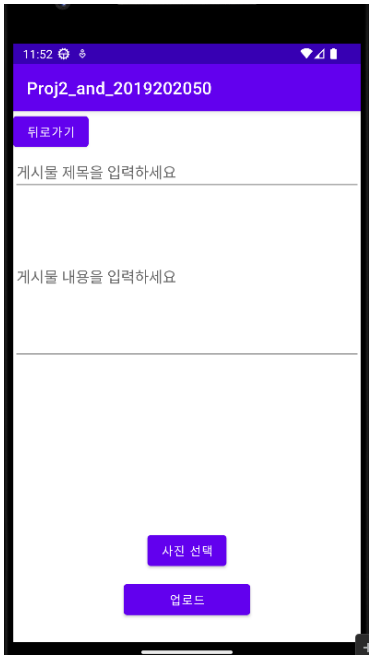
먼저 로고화면이다. 단순한 안드로이드 사진을 첨부했으며 첫 실행때 보여지고 메인 액티비티의 프래그먼트 기본값인 GalleryFragment가 아래와 같이 보여진다.

사진을 첨부해야 하나 사진 불러오기에 문제가 발생해 사진 경로로 대체하여 구현했다.

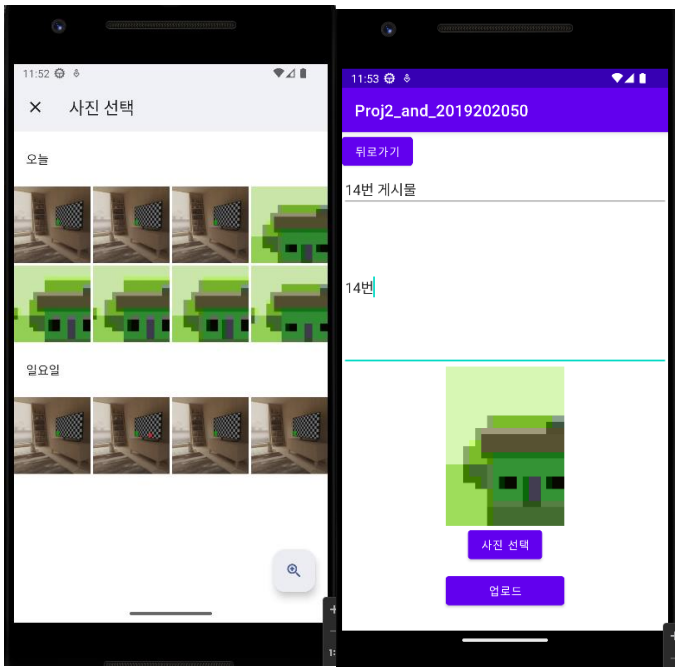
3*4 총 12개의 게시물이 보여진다.

Title을 클릭하면 위와 같이 TitleFragment가 화면에 보여진다.

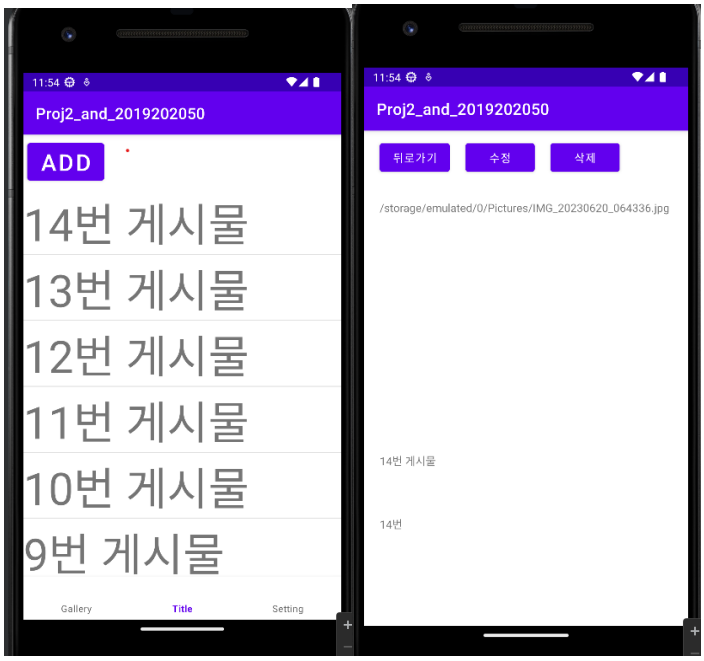
한 화면에 6개의 게시물이 보여진다.



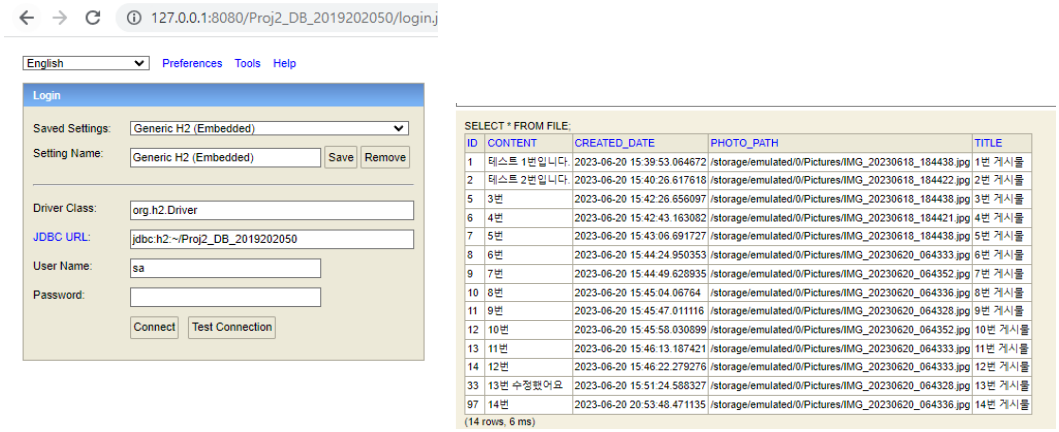
위에서 설명한 두 Fragment에서 ADD버튼을 클릭하면 보여지는 Upload 액티비티이다. 제목과 내용을 입력할 수 있고 사진 선택 버튼을 누르면 아래와 같이 사진을 선택할 수 있다.



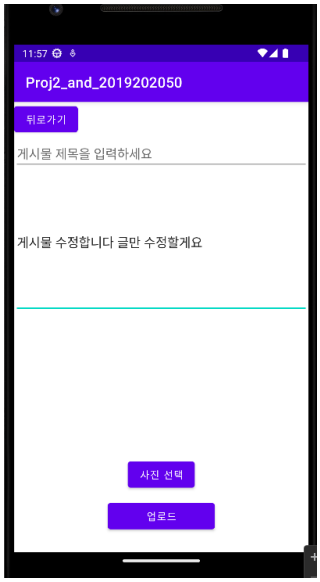
사진을 선택하면 Upload 뷰에 보여지게 되고 위와 같이 14번 게시물이라는 제목을 가진 게시물을 업로드해본다.



업로드를 하게되면 위와같이 게시물이 보여지게 되고 클릭하면 제목과 내용 그리고 사진 경로가 저장된 것을 확인할 수 있다.



위와 같이 서버의 데이터베이스를 확인해보면 14번 게시물이 저장된 것을 알 수 있다.



14번 게시물을 수정하는 모습이다.

수정시에는 게시물 생성과는 다르게 수정하고 싶은 항목만 적으면 그 부분만 데이터베이스에서 수정된다. 위와 같이 업로드하게 되면

SELECT * FROM FILE;

ID	CONTENT	CREATED_DATE	PHOTO_PATH	TITLE
1	테스트 1번입니다.	2023-06-20 15:39:53.064672	/storage/emulated/0/Pictures/IMG_20230618_184438.jpg	1번 게시물
2	테스트 2번입니다.	2023-06-20 15:40:26.617618	/storage/emulated/0/Pictures/IMG_20230618_184422.jpg	2번 게시물
5	3번	2023-06-20 15:42:26.656097	/storage/emulated/0/Pictures/IMG_20230618_184438.jpg	3번 게시물
6	4번	2023-06-20 15:42:43.163082	/storage/emulated/0/Pictures/IMG_20230618_184421.jpg	4번 게시물
7	5번	2023-06-20 15:43:06.691727	/storage/emulated/0/Pictures/IMG_20230618_184438.jpg	5번 게시물
8	6번	2023-06-20 15:44:24.950353	/storage/emulated/0/Pictures/IMG_20230620_064333.jpg	6번 게시물
9	7번	2023-06-20 15:44:49.628935	/storage/emulated/0/Pictures/IMG_20230620_064352.jpg	7번 게시물
10	8번	2023-06-20 15:45:04.06764	/storage/emulated/0/Pictures/IMG_20230620_064336.jpg	8번 게시물
11	9번	2023-06-20 15:45:47.011116	/storage/emulated/0/Pictures/IMG_20230620_064328.jpg	9번 게시물
12	10번	2023-06-20 15:45:58.030899	/storage/emulated/0/Pictures/IMG_20230620_064352.jpg	10번 게시물
13	11번	2023-06-20 15:46:13.187421	/storage/emulated/0/Pictures/IMG_20230620_064333.jpg	11번 게시물
14	12번	2023-06-20 15:46:22.279276	/storage/emulated/0/Pictures/IMG_20230620_064333.jpg	12번 게시물
33	13번 수정했어요	2023-06-20 15:51:24.588327	/storage/emulated/0/Pictures/IMG_20230620_064328.jpg	13번 게시물
97	게시물 수정합니다 글만 수정할게요	2023-06-20 20:53:48.471135	/storage/emulated/0/Pictures/IMG_20230620_064336.jpg	14번 게시물

(14 rows, 0 ms)

위와 같이 글만 수정된 것을 확인할 수 있다.

이 상태에서 14번 게시물을 삭제하게 되면

SELECT * FROM FILE;

ID	CONTENT	CREATED_DATE	PHOTO_PATH	TITLE
1	테스트 1번입니다.	2023-06-20 15:39:53.064672	/storage/emulated/0/Pictures/IMG_20230618_184438.jpg	1번 게시물
2	테스트 2번입니다.	2023-06-20 15:40:26.617618	/storage/emulated/0/Pictures/IMG_20230618_184422.jpg	2번 게시물
5	3번	2023-06-20 15:42:26.656097	/storage/emulated/0/Pictures/IMG_20230618_184438.jpg	3번 게시물
6	4번	2023-06-20 15:42:43.163082	/storage/emulated/0/Pictures/IMG_20230618_184421.jpg	4번 게시물
7	5번	2023-06-20 15:43:06.691727	/storage/emulated/0/Pictures/IMG_20230618_184438.jpg	5번 게시물
8	6번	2023-06-20 15:44:24.950353	/storage/emulated/0/Pictures/IMG_20230620_064333.jpg	6번 게시물
9	7번	2023-06-20 15:44:49.628935	/storage/emulated/0/Pictures/IMG_20230620_064352.jpg	7번 게시물
10	8번	2023-06-20 15:45:04.06764	/storage/emulated/0/Pictures/IMG_20230620_064336.jpg	8번 게시물
11	9번	2023-06-20 15:45:47.011116	/storage/emulated/0/Pictures/IMG_20230620_064328.jpg	9번 게시물
12	10번	2023-06-20 15:45:58.030899	/storage/emulated/0/Pictures/IMG_20230620_064352.jpg	10번 게시물
13	11번	2023-06-20 15:46:13.187421	/storage/emulated/0/Pictures/IMG_20230620_064333.jpg	11번 게시물
14	12번	2023-06-20 15:46:22.279276	/storage/emulated/0/Pictures/IMG_20230620_064333.jpg	12번 게시물
33	13번 수정했어요	2023-06-20 15:51:24.588327	/storage/emulated/0/Pictures/IMG_20230620_064328.jpg	13번 게시물

(13 rows, 1 ms)

위와 같이 서버의 데이터베이스에서도 사라지고 Fragment에도 게시물이 뜨지 않는 것을 확인할 수 있다.

```
E6L_emulation: app_time_stats: avg=500.68ms min=499.03ms max=502.33ms count=2
JSON Payload: {"id":97,"title":"14번 게시물","content":"게시물 수정합니다 글만 수정할게요","photoPath":"\/storage\/emulated\/0\/Pictures\/IMG_20230620_064336.jpg"}
TrafficStats: tagSocket(64) with statsTag=0xffffffff, statsUid=-1
Response: received successful
E6L_emulation: app_time_stats: avg=55.51ms min=12.55ms max=497.03ms count=18
```

추가적으로 위처럼 로그를 확인하여 요청이 성공적으로 처리되었는지를 확인할 수 있다.

기존에는 프래그먼트를 시작할 때 서버에서 모든 게시물 데이터를 로드하기 때문에 새로운 게시물을 생성,수정,삭제시 앱을 재시작하는 것이 유일한 방법이었다. 그 이유는 메인 액티비티에서 프래그먼트를 한번 생성하고 다른 프래그먼트를 보여줄 때는 숨겨두고 다시 요청이 들어오면 보여주었기 때문이다. 따라서 새로고침버튼을 만들거나 어떤 처리를 하고난 후 리스너로 서버에서 데이터를 다시 가져오거나하는 여러 방법이 있으나 프래그먼트를 숨겨두지 않고 매번 재시작하는 것으로 구현하였다.

```
Fragment currentFragment = fragmentManager.getPrimaryNavigationFragment();
if (currentFragment != null) {
    fragmentTransaction.remove(currentFragment);//현재 프래그먼트를 제거함
}
```

위와 현재 프래그먼트를 제거한 후 다른 프래그먼트를 보여주게 되면 게시물 데이터가 바뀌었을 때 다른 프래그먼트를 누른 후 다시 보고싶은 프래그먼트를 누르게 되면 게시물이 업데이트되어 보여진다.

다음은 게시물 생성시 예외처리이다.

```
D/OpenGLRenderer: endAllActiveAnimators on 0x7b03e0965d10 (RippleDrawable) with handle 0x7b02f0962880
D/CompatibilityChangeReporter: Compat change id reported: 78294732; UID 10169; state: ENABLED
D/JSON Payload: {"title":"글자 수 제한 테스트","content":"ㅁㄴㅇㄹㅁㄴㅇ","photoPath":"\/storage\/emulated\/0\/Pictures\/IMG_20230620_064352.jpg"}
D/TrafficStats: tagSocket(98) with statsTag=0xffffffff, statsUid=-1
D/Response: Request failed: 400
D/EGL_emulation: app_time_stats: avg=31.96ms min=12.86ms max=399.38ms count=34
```

```
D/EGL_emulation: app_time_stats: avg=450.21ms min=354.70ms max=499.63ms count=3
D/JSON Payload: {"title":"글자 수 제한","content":"0123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789 0","photoPath":"\/storage\/emulated\/0\/Pictures\/IMG_20230620_064352.jpg"}
D/TrafficStats: tagSocket(96) with statsTag=0xffffffff, statsUid=-1
D/Response: Request failed: 400
```

게시물의 제목이 10자를 넘어갔을 때 또는 내용이 100자를 넘어갔을 때 서버 측에서 요청에 대해 에러 응답을 반환하는 것을 알 수 있다.

<Consideration>

이번 프로젝트는 지난 프로젝트와 비슷하게 게시물을 생성,수정,삭제가 가능한 게시판을 구현하는 것이었고 이를 웹페이지가 아닌 안드로이드 환경에서 진행하는 것이었다. 첫번째로 액티비티와 프래그먼트의 생명주기와 다양한 뷰에 대한 개념이 확실히 잡혀있지 않아 코드를 이해하고 구현하는게 힘들었다. 그래서 단순히 3*4형식으로 그리드뷰를 만드는 것도 힘들었고 리스트를 6개만 나오게 하는 것도 어려웠다. 어떻게 구현하면 될지는 알고 있지만 리스너를 어디에 사용해야 하는 것인지 서버와의 통신은 코드의 어느 부분에 진행해야 하는 것인지등이 헷갈렸다. 두번째로는 json형식으로 서버와 통신하는 것이 너무 어려웠던 것 같다. 이는 OkHttp라는 API를 사용해서 비교적 쉽게 해결했지만 json형식과 DTO형식사이에서 파싱처리를 해야하는 것과 이런 개발툴을 사용하는게 너무 낯설게 느껴졌던 것 같다. 마지막은 결국 해결하지 못한 사진 불러오기이다. 서버와 데이터를 주고받는 것은 해결했으나 게시물의 제목,내용,사진경로를 데이터베이스에 저장하고 이를 불러와 사진 경로를 이용해 사진을 이미지뷰에 넣으려고 했으나 오류가 발생했다.

```
> com.example.proj2_and_2019202050 (test)
java (generated)
res
  > drawable
  v layout
      activity_main.xml
      activity_upload.xml
      fragment_gallery.xml
      fragment_settings.xml
      fragment_title.xml
  > menu
  > mipmap
  > values
  > xml
res (generated)
Gradle Scripts
build.gradle (Project: Proj2_and_2019202050)
```

```java
    }

    public View getView(int position, View convertView, ViewGroup parent) {
        ImageView imageView;
        if (convertView == null) {
            imageView = new ImageView(context);
            imageView.setLayoutParams(new GridView.LayoutParams( w: 400,  h: 340));
            imageView.setScaleType(ImageView.ScaleType.FIT_XY);
        } else {
            imageView = (ImageView) convertView;
        }
        Bitmap bitmap = BitmapFactory.decodeFile(dtoList.get(position).getPhotoPath());
        imageView.setImageBitmap(bitmap);
        return imageView;
    }
}
```

```
app
I/Gralloc4: mapper 4.x is not supported
D/HostConnection: createUnique: call
D/HostConnection: HostConnection::get() New Host Connection established 0x7b0360927c90, tid 14777
W/Gralloc4: allocator 4.x is not supported
D/HostConnection: HostComposition ext ANDROID_EMU_CHECKSUM_HELPER_v1 ANDROID_EMU_native_sync_v2 ANDROID_EMU_native_sync_v3 ANDROID_EMU_native_sync_v4 ANDROID_EMU_dma_v1
W/Parcel: Expecting binder but got null!
E/BitmapFactory: Unable to decode stream: java.io.FileNotFoundException: /storage/emulated/0/Pictures/IMG_20230618_095316.jpg: open failed: EACCES (Permission denied)
E/BitmapFactory: Unable to decode stream: java.io.FileNotFoundException: /storage/emulated/0/Pictures/IMG_20230618_095316.jpg: open failed: EACCES (Permission denied)
E/BitmapFactory: Unable to decode stream: java.io.FileNotFoundException: /storage/emulated/0/Pictures/IMG_20230618_095316.jpg: open failed: EACCES (Permission denied)
E/BitmapFactory: Unable to decode stream: java.io.FileNotFoundException: /storage/emulated/0/Pictures/IMG_20230618_095316.jpg: open failed: EACCES (Permission denied)
D/EGL_emulation: app_time_stats: avg=126.21ms min=0.88ms max=1347.63ms count=11
D/EGL_emulation: app_time_stats: avg=689.42ms min=14.98ms max=10780.99ms count=16
W/System: A resource failed to call close.
```

위와 같은 오류인데 접근 권한이 없다라는 오류메세지와 함께 사진을 불러오는 것에 실패했다. 구글링 후 나온 결론은 안드로이드 버전 10 이후에는 단순히 manifest에

액세스권한을 넣는걸로는 해결이 안된다는 것이었다. 런타임때 액세스 권한을 앱에서 요청하여 권한을 얻게되면 그때 접근이 된다는 것이 결론이었다. 따라서 런타임때 권한 요청 메시지를 사용자에게 보내고 처리하고 싶었으나 결국은 해결하지 못했다. 따라서 사진경로를 사진 대신 넣는 것으로 마무리하였다.

안드로이드와 스프링을 같이 구현해보면서 스마트폰의 앱이 이런식으로 동작하는구나를 알 수 있어서 재밌고 신기했다. 사진처리를 하지 못했지만 그래도 결과적으로는 서버통신도 가능하고 전체적인 동작 흐름도 준수하게 진행되었기에 만족스럽고 다음엔 런타임시에 권한을 요청하고 이를 해결해보는 것도 좋을 것 같다.