Community 구현 설명

학번 : 20173324
이름 : 이 상렬

Activity

1. community_main => 커뮤니티 기능의 메인 액티비티
2. community_Detail_PostActivity => 게시글 자세히 보기 관련 액티비티
3. community_Edit_Activity => 작성글 수정 액티비티
4. community_Post_Activity => 게시글 작성 액티비티
5. CmListAdapter => 커뮤니티 어댑터
6. FBAuth => 파이어 베이스에서 유저 정보 가져오는 곳.
7. FBrtbe => realtime firebase에 데이터 업로드 관련.
8. Model => 게시글 모델

XML

1. activity_community_edit.xml => 수정xml
2. activity_community_main.xml => 커뮤니티 메인 화면 xml
3. activity_community_post.xml => 게시글 작성 xml
4. community_details_post.xml => 게시글 클릭시 글 자세히 보기 관련 xml
5. community_item_list.xml => 메인에 표현할 커뮤니티 글 리스트 xml
6. community_setting_btninside.xml => 삭제 수정 선택 버튼 xml

전체적인 동작 구조.

사용자가 커뮤니티 버튼 클릭 -> 게시글 리스트가 화면에 보여짐

-> 글선택시 글을 볼수있음
● 자신의 글 선택시에는 수정 삭제 버튼 이 보임
● 자신의 글이 아닐시 수정 삭제 버튼 보이지 않음
-> 수정 선택시 게시글 수정 and 삭제 선택시 게시글 삭제
->완료후 초기화면으로 화면전환

-> 오른쪽 아래 하단의 연필모양 클릭시 게시글 작성
-> 완료시 초기 리스트화면으로 화면전환

Activity설명

●FBAuth, FBrtbe, Model

파이어베이스 관련 액티비티로 유저의 정보를 firebase에서 읽어오고, 커뮤니티에서 사용되는 데이터들을 파이어 베이스와 연동하기위해 엑티비티 작성. 해당 데이터들을 noticeboard에 저장. 그리고 그곳에 데이터를 저장할 모델 선언. (제목, 내용, uid, time, cid)

```
class FBrtbe {
    companion object {
        private val database = Firebase.database

        val noticeboard = database.getReference( path: "noticeboard")
```

●CmListAdapter

초기 화면에서 작성된 글 리스트들을 보기 위해서 어뎁터 작성. 해당 어뎁터에서 제목, 내용, 내용 일부분, 시간등을 리스트에서 볼수 있도록 선언. 그 후 Model에 연결.

●Community_main

- 커뮤니티의 메인 구현 부분으로 어뎁터를 선언해 메인 액티비티에서 리스트 내용을 볼수있도록 선언. +Model 역시 같이 선언.

-리스트 뷰 클릭시 해당 게시글들을 detail_post로 intent해서 볼수 있도록 연결하였고 넘길때 keylist를 같이 detail_post로 넘겨 해당 게시글을 알수 있도록 구성. 이때 해당글의 댓글도 알수 있도록 cid를 넘겨줌.

-search start에서 아이디 입력을 통해 작성 게시글들을 검색할수 있는 기능 작성.

-writebtn, 오른쪽 아래의 연필 모양 버튼을 클릭시에 게시글 작성 할수 있도록 intent.
이 부분에서 id == 사용자email 값을 넘겨주게됨.

- getFBBoardData()에서 key를 통해 모델에 접근해 사용자의 정보를 가져오게됨. 이때 key를 통해 모델에 근거하여 데이터를 가져옴. 해당 데이터들의 경우 데이터들이 점점 아래로 쌓여 가기 때문에 리스트에서 최신 것이 위로 올라오도록 .reverse()작성. 이때 저장하는 keylist역시도 .reverse()해주어야함.

-메인 화면에서 화면바의 내용을 출력해 주기위해 해당 내용 작성.

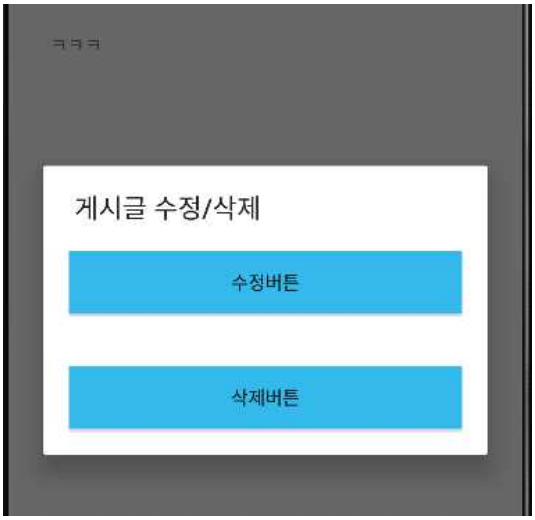
```
override fun onDataChange(dataSnapshot: DataSnapshot) {

    //리스트가 한번더 출력을 하지않도록해줌.
    boardDataList.clear()

    //key를 통해 모델에 접근해 정보 가져옴.
    for (dataModel in dataSnapshot.children) {
        Log.d(TAG, dataModel.toString())

        val item = dataModel.getValue(Model::class.java)
        boardDataList.add(item!!)
        boardKeyList.add(dataModel.key.toString())
    }

    //게시글 순서가 최신게 위로 올라오도록 .reverse()사용
    //순서가 바뀌므로 keylist도 .reverse()
    boardDataList.reverse()
    boardKeyList.reverse()

    Adapter.notifyDataSetChanged()
    Log.d(TAG, boardDataList.toString())

}
```

●Community_Detail_PostActivity

초기화면에서 작성 버튼 클릭시 화면이 intent되어 실행됨.
-title_et, content_et를 통해 작성된 글을 firebase로부터 받아와 볼수 있도록 설정. key를 통해서 해당 선택한 데이터를 찾아 낼수 있도록 설정.

-detail_postActivity에서 키를 통해서 해당 게시글의 정보를 받아오는데 이때 만약 글이 사용자의 글이 아닐 경우에는 수정 삭제 버튼이 보이지 않도록 설정. 만약 작성한 사람과 본인이 동일할 시 해당 버튼들이 보이게됨.

```
    override fun onDataChange(dataSnapshot: DataSnapshot) {
        try {

            val dataModel = dataSnapshot.getValue(Model::class.java)

            Log.d(TAG, dataModel!!.title)

            title_et.text = dataModel!!.title
            content_et.text = dataModel!!.content
            val myUid = intent.getStringExtra( name: "id").toString()
            val writerUid = dataModel!!.uid
            val settingbtn: ImageView = findViewById(R.id.boardSettingIcon)

            if(myUid.equals(writerUid)){

                Log.d(TAG,  msg: "내가 쓴 글")
                settingbtn.isVisible = true

            } else {
                settingbtn.isVisible = false
                Log.d(TAG,  msg: "내가 쓴 글 아님")
            }
```



-수정 삭제버튼을 선택하면 다음과 같은 창이 뜨며 선택 후 해당 기능수행

-삭제의 경우 해당 글을 삭제해주고 동시에 같이있는 댓글들을 삭제해줌.

-수정의 경우 해당 key값을 edit_activity에 넘겨주며 화면을 intent 전환하게됨.

●Community Post_Activity

-작성과 관련된 부분들을 선언해 주고 해당 key값을 통해 firebase에 push()

-해당 코드 실행 후 toast메시지를 통해 사용자에게 작성 완료 통보. 그 후 finish()

```
//해당 key값을 push()
val key = FBrtbe.noticeboard.push().key.toString()

//key를 통해 noticeboard에 해당 게시글 저장.
FBrtbe.noticeboard
      .child(key)
      .setValue(Model(title, content, uid, time, cid))

//완료후 Toast메시지 출력
 Toast.makeText( context: this,  text: "게시글 입력완료!", Toast.LENGTH_LONG).show()

 finish()
```

●Community Edit_Activity

-Detail_PostActivity에서 해당 게시글의 key값을 넘겨 받은 후 key를 통해 해당 게시물 데이터에 접근.

-수정 xml을통해 제목과 내용의 수정한 내용들을 model에 기반해서 작성 후 key값을 통해 해당 게시물의 제목과 수정 내용 저장.

-수정이 완료 된후 toast메시지를 통해 사용자에게 수정 완료됨을 출력. 그 후 finish()