

---

# 2021 졸업 작품 포트폴리오

---

3조 20164115 문상현

<https://github.com/no1msh>

# 목차.

- 1 많은 기능과 필요한 지식
- 2 구현 과정
- 3 앞으로의 발전 방향

---

Part 1, **맞은 기능과 필요지식**

---



# 제가 맡은 기능



## 로그인 기능 구현

회원 가입과 로그인을 하여 데이터를 서버에 두고 관리함.



## 커뮤니티 기능 구현

회원들끼리 사진과 글을 직접 작성하여 서버에 올릴 수 있는 게시판을 제작

# 파이어 베이스를 택한 이유

## 파이어 베이스 란?

- 파이어베이스(Firebase)는 2011년 파이어베이스(Firebase, Inc)사가 개발하고 2014년 구글에 인수된 모바일 및 웹 애플리케이션 개발 플랫폼이다. (출처: 위키백과)



## 왜 써야 했을까?

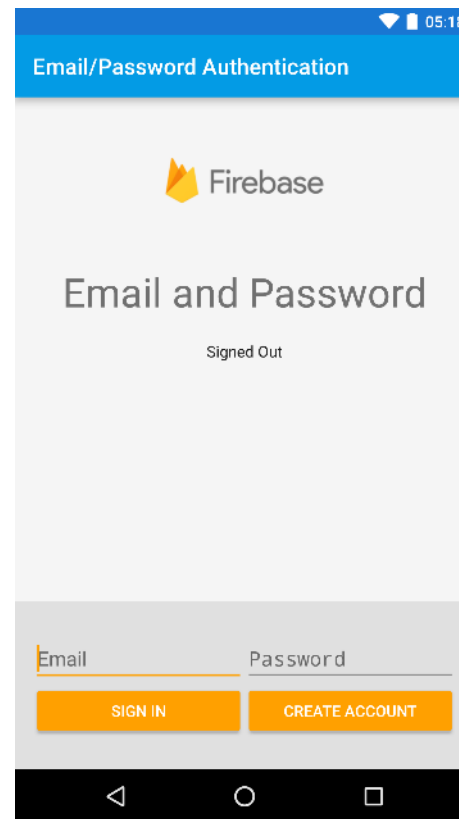
1. 서버 인프라를 따로 구축할 필요가 없다!
  - Firestore, Realtime Database, Authentication, Storage 이미 구축되어 있다.
2. 무료 이용이 가능하다.
  - 특정 사용량 까지 무료이고 사용자의 사용량에 따라 종량제로 유료로 사용 가능.
3. 사용 방법이 쉽다.
  - 백엔드 개발 상식이 없어도 개발이 가능.

# 로그인 기능 구현 계획

## 로그인 기능의 필요 요소

- 이메일로 회원 가입하고 그 바탕으로 로그인이 될 수 있게 해야한다.
- 본인 인증을 요구하는 메일을 발송 한다.
  - 인증을 받아야만 로그인이 가능하게 구현 한다.
- 비밀번호를 재설정 할 수 있어야 한다.
  - 가입 당시 기입했던 이메일로 비밀번호 재설정으로 해결한다.
- 회원 가입 시 비밀번호를 한번 더 입력하는 비밀번호 확인 칸을 만든다.

파이어 베이스의 Authentication(인증) 기능을 활용하면 모두 구현 가능.



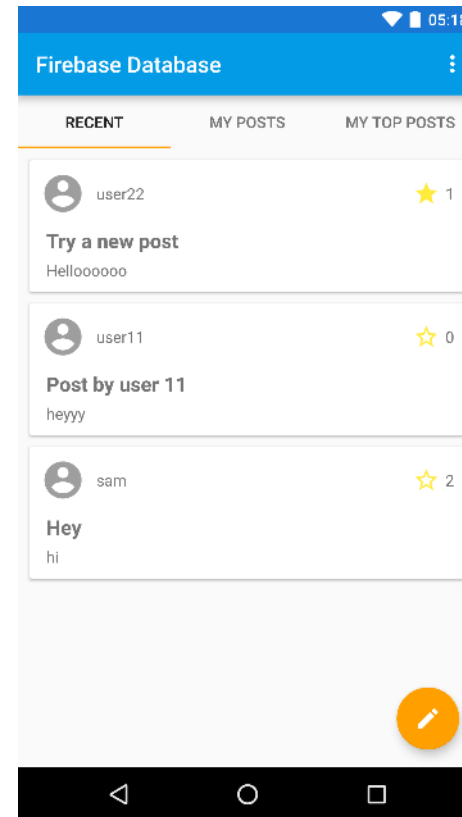
<파이어베이스 로그인 예제>

# 게시판 기능 구현 계획

## 게시판 기능의 필요 요소

- 사용자가 글을 올리면 실시간으로 다른 사용자에게 보여져야 한다.
- 사진을 게시하여 볼 수 있도록 한다.
- 좋아요 및 댓글을 사용 할 수 있어야한다.
- 게시글 목록에서 사진 , 제목, 좋아요, 댓글을 확인 할 수 있어야 한다.

파이어 베이스의 RealTimeDB(실시간 데이터베이스) 기능을 활용하면 모두 구현 가능.

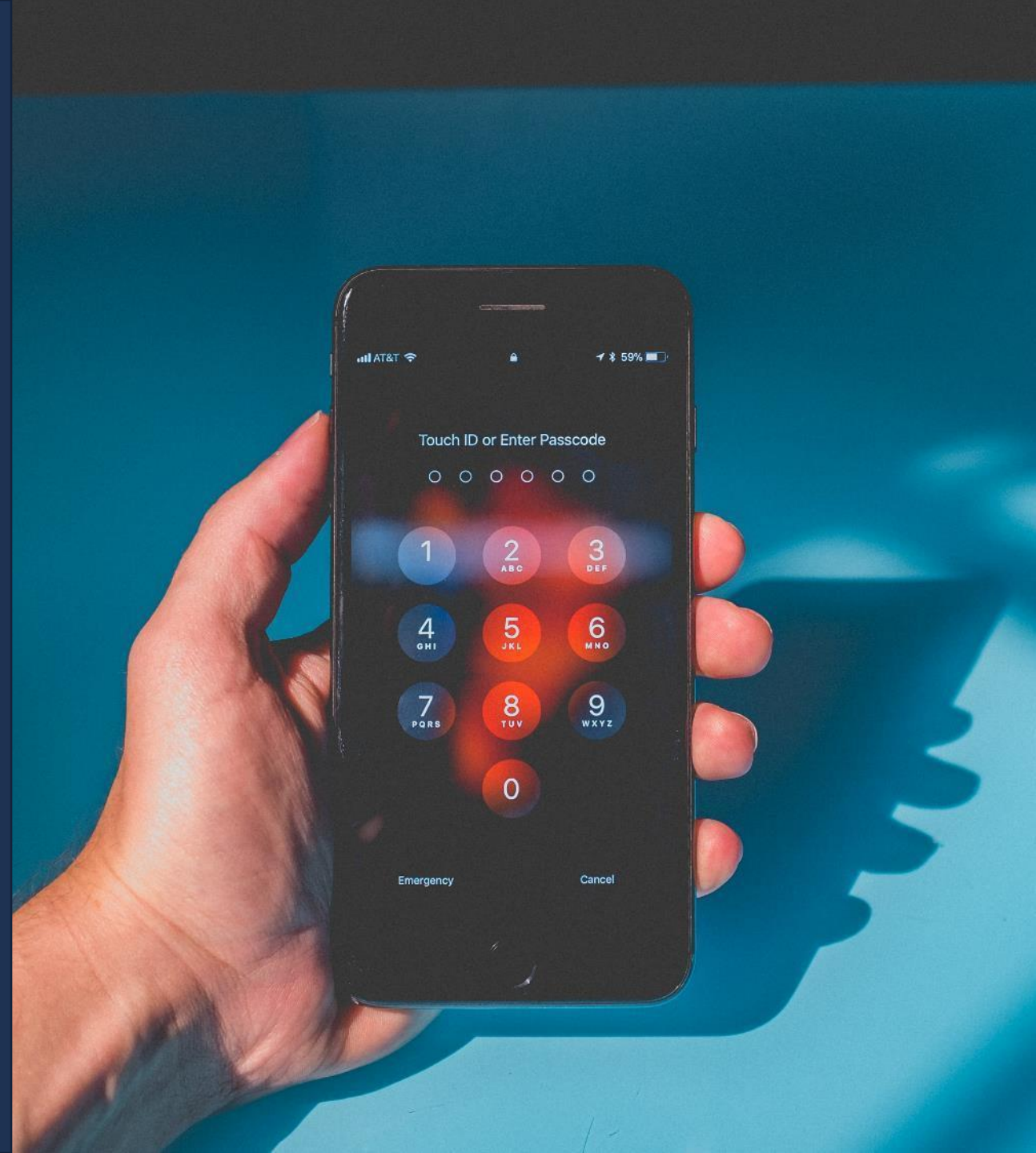


<파이어베이스 게시판 예제>

---

Part 2, 구현 과정

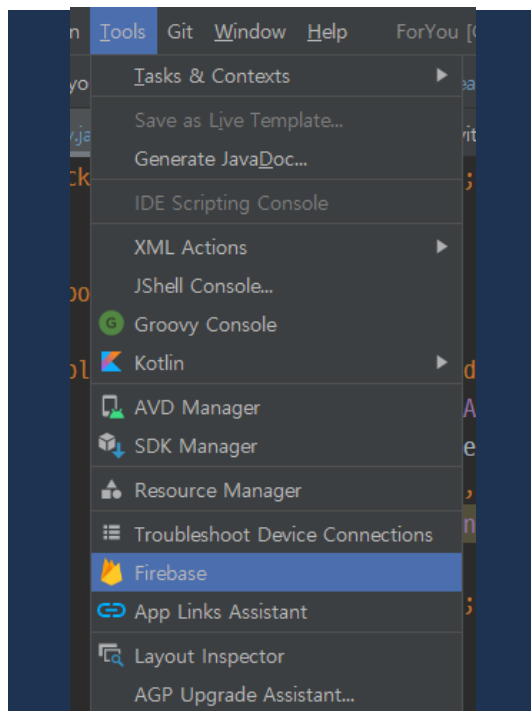
---





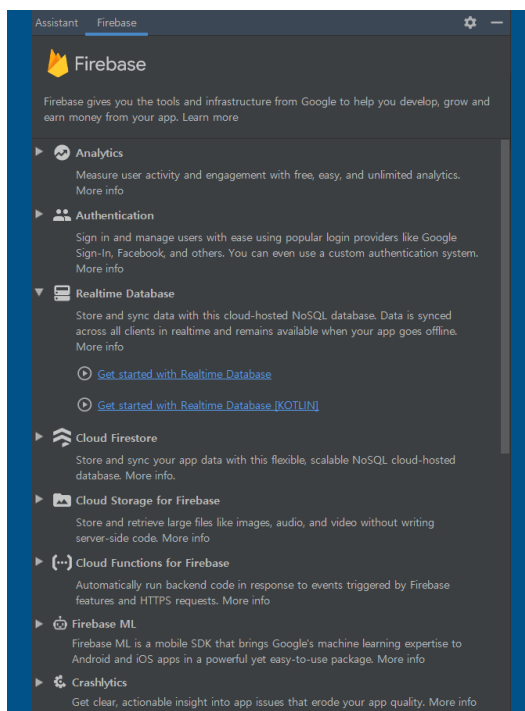
# 파이어 베이스와 안드로이드 스튜디오의 연결 과정

001



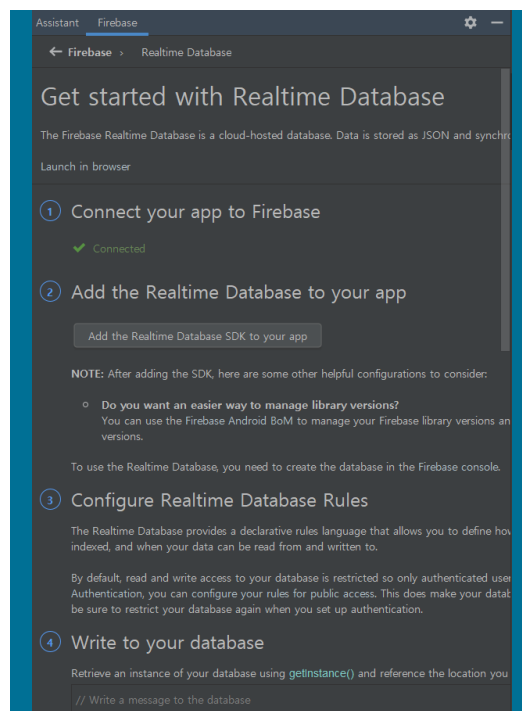
안드로이드 스튜디오 상단 탭  
Tools – Firebase 클릭

002



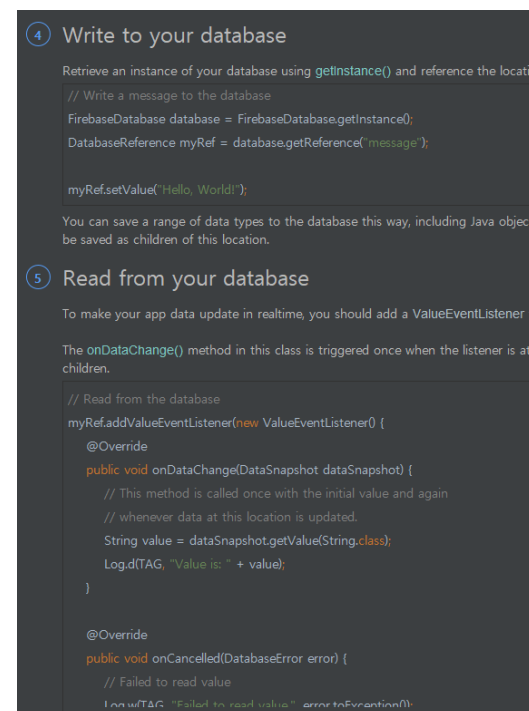
오른쪽 Assistant 탭  
사용할 기능을 클릭

003



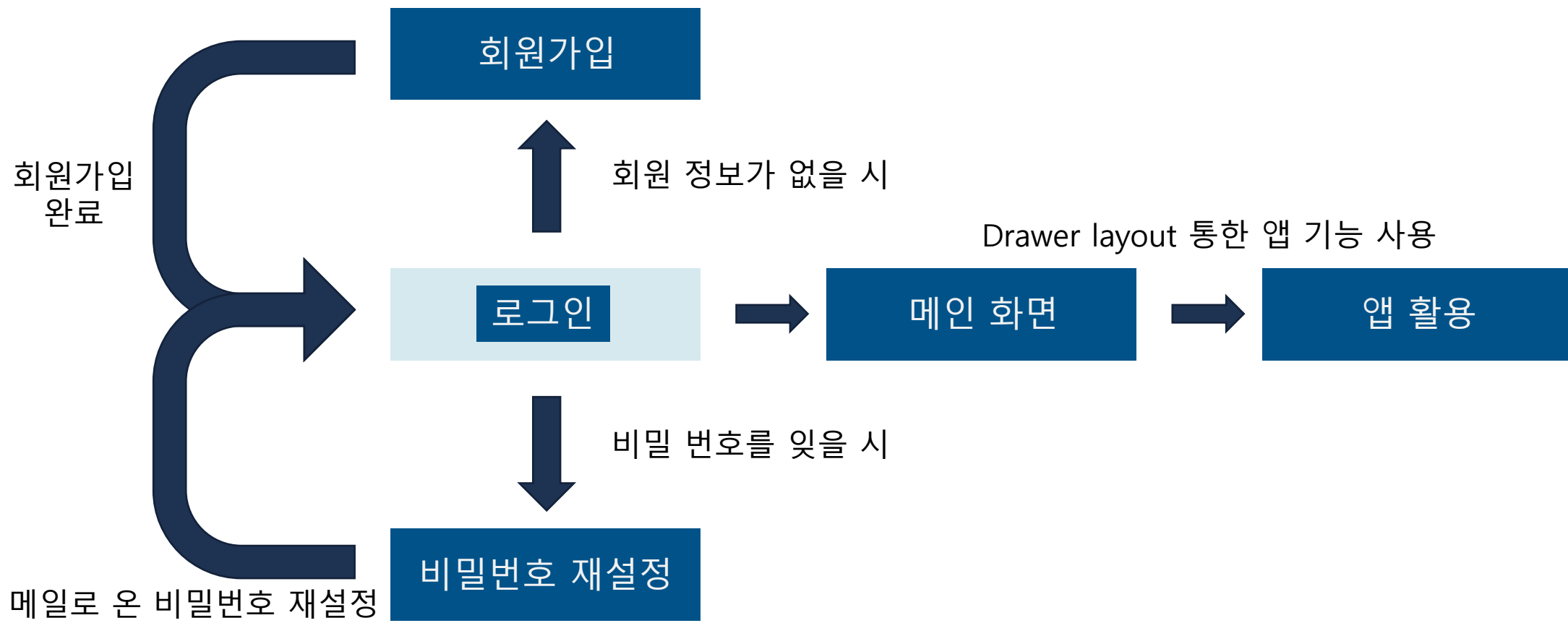
절차대로 수행

004



예시코드 확인 및 실행

# 로그인 화면의 흐름



# 회원가입 화면 구성



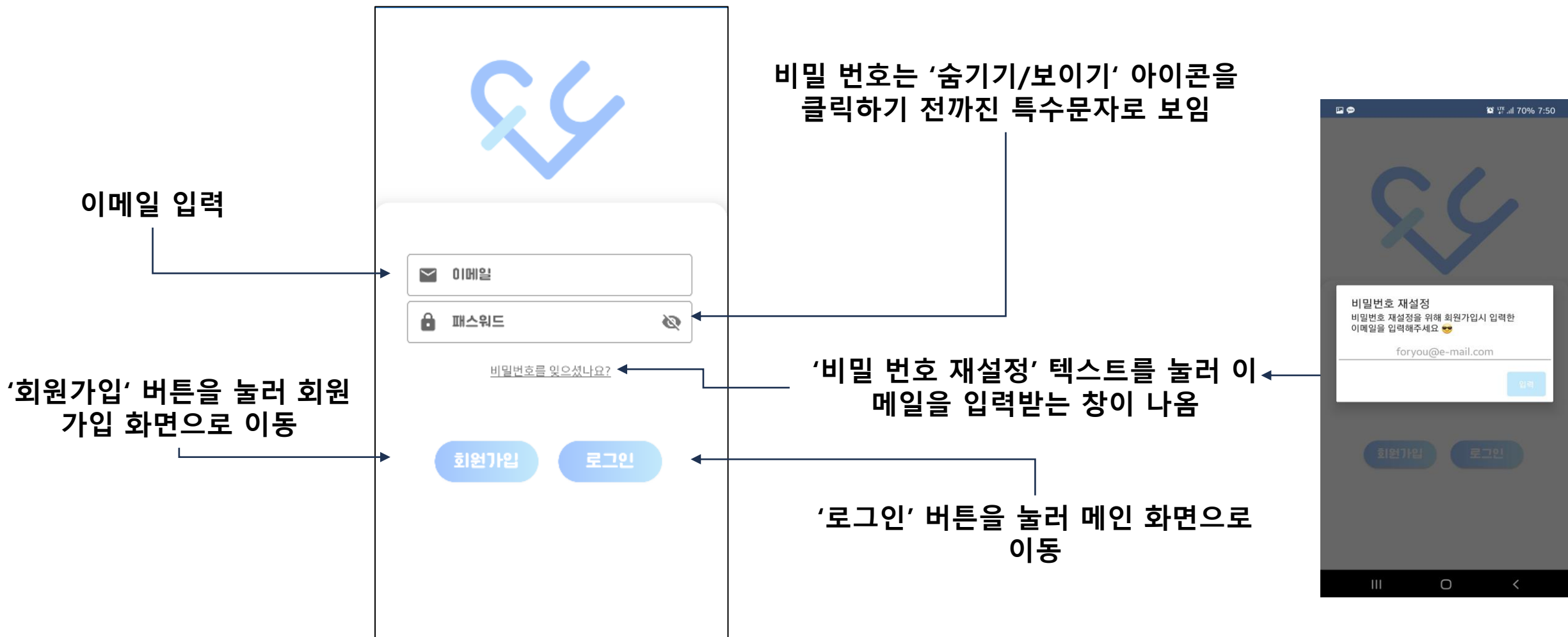
# 회원가입 구현

```
mBtnSubmit.setOnClickListener(new View.OnClickListener() {  
    @Override  
    public void onClick(View v) {  
        mStrEmail = mEtSignUpEmail.getText().toString().trim();  
        mStrPassword = mEtSignUpPassword.getText().toString().trim();  
  
        if (checkCorrectEmail() && checkCorrectPasswordRepeat()) {  
            createUser(mStrEmail, mStrPassword);  
        }  
    }  
});
```

```
private boolean checkCorrectEmail() {  
    String mEmail;  
    mEmail = mEtSignUpEmail.getText().toString().trim();  
    if (!mEmail.contains("@")) {  
        Toast.makeText(getApplicationContext(), text: "올바른 이메일 형식이 아닙니다.", Toast.LENGTH_SHORT).show();  
    }  
    return mEmail.contains("@");  
}  
  
private boolean checkCorrectPasswordRepeat() {  
    String mPassword, mPasswordRepeat;  
    mPassword = mEtSignUpPassword.getText().toString().trim();  
    mPasswordRepeat = mEtSignUpPasswordRepeat.getText().toString().trim();  
    if (!mPassword.equals(mPasswordRepeat)) {  
        Toast.makeText(getApplicationContext(), text: "비밀번호가 서로 일치하지 않습니다.", Toast.LENGTH_SHORT).show();  
    }  
    return mPassword.equals(mPasswordRepeat);  
}
```

```
// 이메일과 패스워드로 회원가입 메소드  
private void createUser(String email, String password) {  
    mFirebaseAuth.createUserWithEmailAndPassword(email, password)  
        .addOnCompleteListener( activity: this, new OnCompleteListener<AuthResult>() {  
            @Override  
            public void onComplete(@NonNull Task<AuthResult> task) {  
                if (task.isSuccessful()) {  
                    // 회원가입 성공시 이메일로 인증을 요구하는 메일 보냄  
                    mFirebaseUser = mFirebaseAuth.getCurrentUser();  
                    mFirebaseUser.sendEmailVerification().addOnCompleteListener(new OnCompleteListener<Void>() {  
                        @Override  
                        public void onComplete(@NonNull Task<Void> task) {  
                            if (task.isSuccessful()) {  
                                Toast.makeText(getApplicationContext(), text: "메일이 성공적으로 전송되었습니다.", Toast.LENGTH_SHORT).show();  
                                Toast.makeText(getApplicationContext(), text: "회원가입 성공", Toast.LENGTH_SHORT).show();  
                                finish(); // 액티비티 종료  
                            }  
                        }  
                    });  
                } else {  
                    // 계정이 중복된 경우  
                    Toast.makeText(getApplicationContext(), text: "이미 존재하는 계정입니다.", Toast.LENGTH_SHORT).show();  
                }  
            }  
        });  
}
```

# 로그인 화면 구성



# 로그인 구현

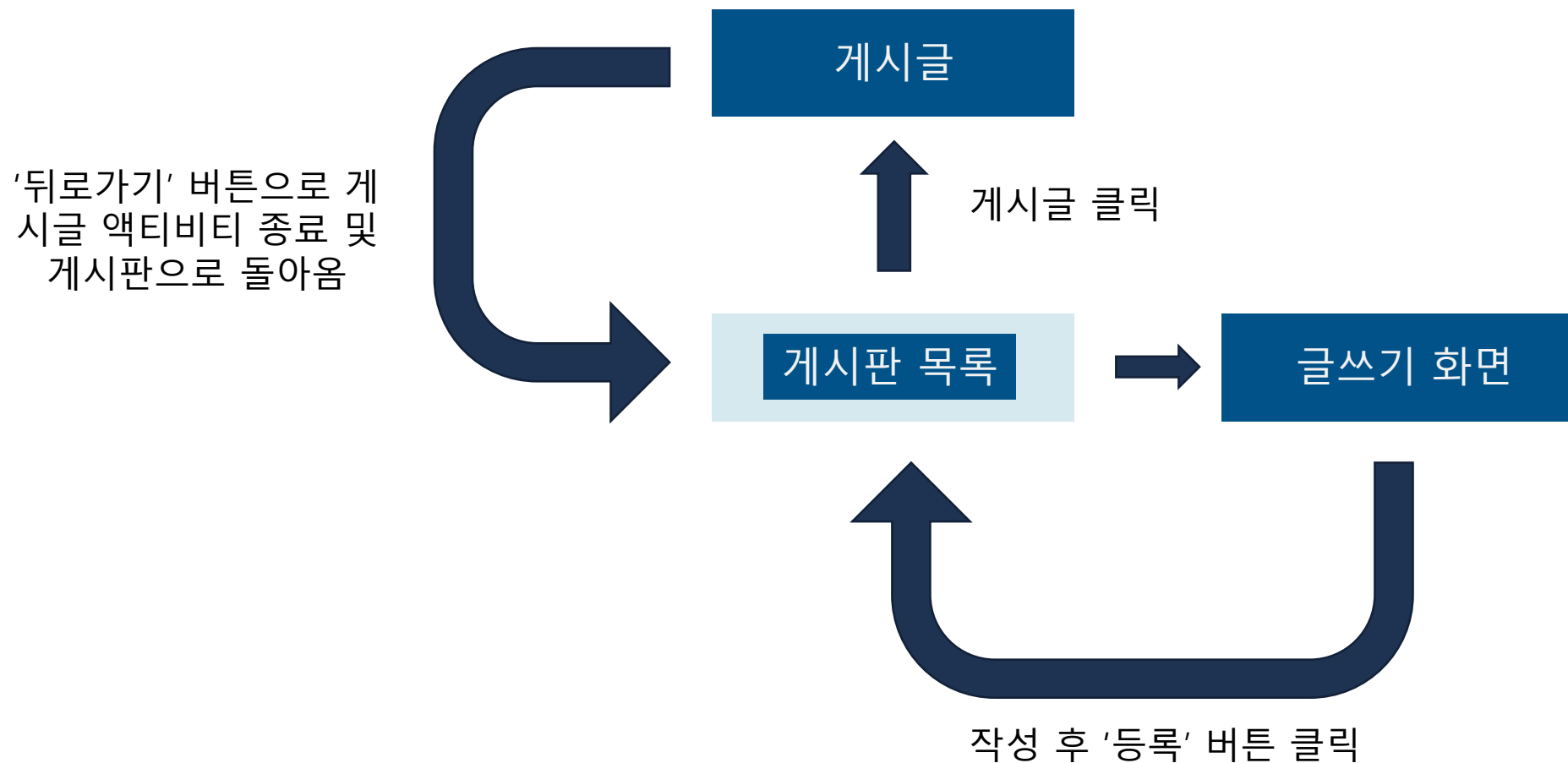
```
mBtnSignIn.setOnClickListener(new View.OnClickListener() {  
    @Override  
    public void onClick(View v) {  
  
        mStrUserEmail = extractEditTextToString(mEtUserEmail);  
        mStrUserPassword = extractEditTextToString(mEtUserPassword);  
  
        if (checkNullEditText(mEtUserEmail) && checkNullEditText(mEtUserPassword)) {  
            if (mStrUserEmail.contains("@")) {  
                signInUser(mStrUserEmail, mStrUserPassword);  
            } else {  
                Toast.makeText(context: SignInActivity.this, text: "올바른 이메일 형식이 아닙니다.", Toast.LENGTH_SHORT).show();  
            }  
        } else {  
            Toast.makeText(getApplicationContext(), text: "이메일 또는 비밀번호를 전부 입력해주세요.", Toast.LENGTH_SHORT).show();  
        }  
    }  
});
```

```
// 로그인 메소드  
public void signInUser(String email, String password) {  
    mFirebaseAuth.signInWithEmailAndPassword(email, password)  
        .addOnCompleteListener( activity: this, new OnCompleteListener<AuthResult>() {  
            @Override  
            public void onComplete(@NonNull Task<AuthResult> task) {  
                if (task.isSuccessful()) {  
                    // 로그인 성공  
                    mFirebaseAuth.addAuthStateListener(mFirebaseAuthStateListener);  
                    Intent intent = new Intent(SignInActivity.this, MainActivity.class);  
                    startActivity(intent);  
                    finish();  
                } else {  
                    // 로그인 실패  
                    Toast.makeText(context: SignInActivity.this, text: "아이디 또는 비밀번호가 일치하지 않습니다.", Toast.LENGTH_SHORT).show();  
                }  
            }  
        });  
}
```

```
mFirebaseAuthStateListener = new FirebaseAuth.AuthStateListener() {  
    @Override  
    public void onAuthStateChanged(@NonNull FirebaseAuth firebaseAuth) {  
        if (mFirebaseUser != null) {  
            if (mFirebaseUser.isEmailVerified()) {  
                Toast.makeText(context: SignInActivity.this, text: "로그인 성공", Toast.LENGTH_SHORT).show();  
                Intent SignUpSuccessIntent = new Intent(packageContext: SignInActivity.this, SignInDoneActivity.class);  
                SignUpSuccessIntent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK | Intent.FLAG_ACTIVITY_CLEAR_TASK | Intent.FLAG_ACTIVITY_CLEAR_TOP);  
                startActivity(SignUpSuccessIntent);  
            } else {  
                Toast.makeText(getApplicationContext(), text: "이메일 인증을 완료해주세요.", Toast.LENGTH_SHORT).show();  
            }  
        }  
    }  
};
```

```
// EditText 의 추출값이 비어있는지 NULL 체크하는 메소드  
private boolean checkNullEditText(EditText editText) {  
    String string = editText.getText().toString().trim();  
    if (string.getBytes().length <= 0) {  
        Toast.makeText(getApplicationContext(), text: "비어있는 값을 채워주세요", Toast.LENGTH_SHORT).show();  
        return false;  
    }  
    return true;  
}
```

# 게시판 화면의 흐름



# 게시판 화면 구성

이미지 표시 (이미지를 등록  
안할 시엔 기본 로고로 표현)

게시글 터치 시 게시글로 이동



게시글 등록시간 표현

좋아요와 댓글 수 표현

글쓰기 플로팅 액션 버튼 클릭  
하여 글쓰기 페이지로 이동



# 게시판 화면 구성

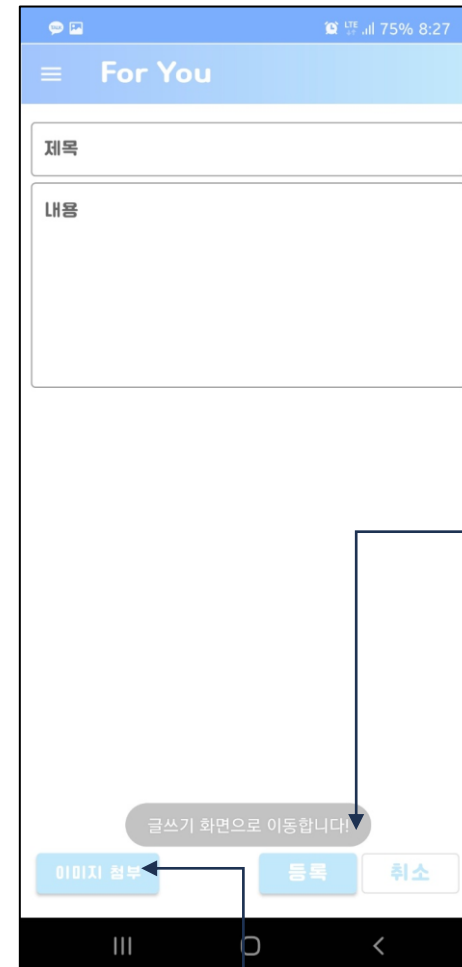


게시글 제목 표현

업로드 한 사진 표현

게시글 내용 표현

좋아요 클릭 및 댓글 입력 가능



게시글 제목 입력

게시글 내용 입력

'등록'버튼 클릭시 게시글이  
업로드,  
'취소' 버튼 클릭시 그대로  
액티비티 종료 및 게시글 목  
록으로 이동

'이미지 첨부' 버튼 클릭시 이미지  
첨부 가능

# 게시판 화면 구현

## Realtime Database

[데이터](#) [규칙](#) [백업](#) [사용량](#)

 결제 사기나 피싱과 같은 악용으로부터 Realtime Database 리소스를 보호하세요.

<https://for-you-android-app-default-rtdb.firebaseio.com/>

for-you-android-app-default-rtdb

Board

Post1

Post2

Post3

Post4

Post5

Post6

Post7

comment

저요저요: "2021-10-11 11:39"

content: "건강관리도 할겸 해서 강아지 산책모임을 구하는데\n관심있으신 분들은 댓글주세요 !"

like


1: "2CH0YW4KMwMXXK0jdijNoTPsEfk2"

postCount: "Post7"

time: "2021-10-11 11:39"

title: "강아지 산책 모임 구합니다!"

uld: "Wt5T1Wct0HdcMTJGoAUgmuaay42"






 데이터베이스 위치: 미국(us-central1)

## Storage

[Files](#) [Rules](#) [Usage](#)

 결제 사기나 피싱과 같은 악용으로부터 Storage 리소스를 보호하세요. [앱 체크 구성](#) 

<gs://for-you-android-app.appspot.com> > Board

<input type="checkbox"/>	이름	크기
<input type="checkbox"/>	 Post2.png	41.75 KB
<input type="checkbox"/>	 Post3.png	2.57 MB
<input type="checkbox"/>	 Post5.png	48.39 KB
<input type="checkbox"/>	 Post6.png	160.86 KB
<input type="checkbox"/>	 Post7.png	998.91 KB

# 게시판 화면 구현

```
database = FirebaseDatabase.getInstance();
databaseReference = database.getReference("Board");
databaseReference.addListenerForSingleValueEvent(new ValueEventListener() {
    @Override
    public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
        // 파이어베이스 데이터베이스의 데이터를 받아오는 곳
        arrayList.clear(); // 기존 배열 리스트 초기화
        for (DataSnapshot snapshot : dataSnapshot.getChildren()) { // 반복문으로 데이터 추출
            Board board = snapshot.getValue(Board.class); // 만들어줬던 Board 객체에 데이터를 담는다.
            arrayList.add(board); // 담은 데이터들을 배열리스트에 넣고 리사이클러뷰로 보낼 준비
        }
        adapter.notifyDataSetChanged(); // 리스트 저장 및 새로고침
    }
}
```

```
FirebaseStorage firebaseStorage = FirebaseStorage.getInstance();
StorageReference boardRef = firebaseStorage.getReference().child("Board/"+arrayList.get(position).getPostCount()+".png");
boardRef.getDownloadUrl().addOnSuccessListener(new OnSuccessListener<Uri>() {
    @Override
    public void onSuccess(Uri uri) {
        Glide.with(holder.itemView).RequestManager
            .load(uri).RequestBuilder<Drawable>
            .error(R.drawable.foryoulogobackground)
            .into(holder.iv_profile);
    }
});
```

```
mBtnWriteSubmit.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        if (mEtWriteTitle.getText() == null || mEtWriteContent.getText() == null) {
            Toast.makeText(context, CommunityWriteActivity.this, "제목과 내용을 입력해주세요", Toast.LENGTH_SHORT).show();
            return;
        } else {
            mStrTitle = mEtWriteTitle.getText().toString();
            mStrContent = mEtWriteContent.getText().toString();
            mTime = getTime();

            FirebaseUser firebaseUser = FirebaseAuth.getInstance().getCurrentUser();
            if (firebaseUser != null) {
                String userId = firebaseUser.getId();
                FirebaseDatabase database = FirebaseDatabase.getInstance();
                DatabaseReference myRef = database.getReference("Board");
                postCount = "Post" + (count);
                Board board = new Board(postCount, mStrTitle, mStrContent, userId, mTime);
                myRef.child(postCount).setValue(board);
                finish();
            }
        }
    }
});
```

```
private void getLikeCount() {
    FirebaseDatabase firebaseDatabase = FirebaseDatabase.getInstance();
    Query likeCount = firebaseDatabase.getReference().child("Board").child(mPostCount).child("like");
    likeCount.addListenerForSingleValueEvent(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot snapshot) {
            mIntLikeCount = (int) snapshot.getChildrenCount();
            Log.e("tag: Firebase", "메소드 안에서의 값 " + mIntLikeCount);
        }

        @Override
        public void onCancelled(@NonNull DatabaseError error) {
            Log.e("tag: Firebase", "");
        }
    });
    return;
}
```

---

Part 3, 앞으로의 발전 방향

---



# 해결 하지 못한 문제점

- SNS 활용 로그인의 부재
  - 카카오톡, 네이버
- 게시판 목록으로 구현한 리사이클러뷰의 로딩 속도
- UI / UX 적 부족 요소
- 스파게티 코드



# 앞으로 공부하고 싶은 방향



## - 아키텍처 패턴 (디자인 패턴)

- 혼자 개발하는게 아닌 다수가 개발할 경우 유지보수와 테스트를 위한 안드로이드 개발자의 필수 요소

## - 코틀린

- 2017년 구글에서 코틀린을 공식언어로 지정
- 2019년 구글에서 만드는 라이브러리들은 코틀린으로 배포할 것을 발표
- 현업에서의 코틀린은 필수 사항

회사명/팀명	언어	아키텍처
PRND Company	Kotlin, Java	Clean, MVVM, MVP
뱅크샐러드	Kotlin	Clean, MVP, MVVM
무신사	Kotlin	MVVM
캐시워크	Kotlin, Java	MVP, MVVM
Coinone Transfer	Kotlin, Java	MVVM
원티드랩	Kotlin	MVVM
플리토	Kotlin, Java	MVP, MVVM
스윙트래커	Kotlin, Java	MVP, MVVM
스폰라디오	Kotlin, Java	Clean, MVP
빙글	Kotlin, Java	MVP, MVVM
아파트너	Kotlin, Java	MVP, MVVM
위드이노베이션	Kotlin, Java	Clean, MVP, MVVM
Station3 다방	Kotlin, Java	MVP, MVVM
Medibloc	Kotlin	Clean, MVVM
잡플래닛	Kotlin, Java	MVP, MVVM
명스플로우	Kotlin, Java	MVVM
에스티유니타스	Kotlin	MVVM



# Q&A

