**애니메이션 적용**

var anim1 = AnimationUtils.loadAnimation(applicationContext,R.anim.alpha1\_anim) //해당 애니메이션을 xml로 작성하여 가져와 적용함

var user=firebaseAuth.currentUser

var userEmail =user!!.email

imgIcon3.setOnClickListener { view ->

imgIcon3.startAnimation(anim1)

}

**drag & drop**

btnSheep.setOnTouchListener(object : View.OnTouchListener {

override fun onTouch(v: View?, event: MotionEvent?): Boolean {

if(event?.action==MotionEvent.ACTION\_DOWN) //해당 버튼을 누르면 좌표값저장 및 상태변경

{

isMoving1=START\_DRAG

if(start\_yn1){

offset\_x1=event.x

offset\_y1=event.y

start\_yn1=false

}

}

else if (event?.action==MotionEvent.ACTION\_UP) //지금 드래그가끝남을 알려줌

{

isMoving1=END\_DRAG

}

else if (event?.action==MotionEvent.ACTION\_MOVE)

{

if(isMoving1==START\_DRAG){ //상대좌표와 절대좌표를 이용하여 드래그 이동 과정

v?.x=event.rawX-offset\_x1

v?.y=event.rawY-offset\_y1

if(checkcount1==0){

firstoffset\_x1=event.rawX-offset\_x1

firstoffset\_y1=event.rawY-offset\_y1

checkcount1=checkcount1+1

}

}

}

return v?.onTouchEvent(event) ?:true

}

})

**실시간 데이터 변경 리스너**

databaseReference.addValueEventListener(object : ValueEventListener{

override fun onCancelled(p0: DatabaseError) { //파이어베이스에서 실시간으로 데이터가 변경되면 호출을함

}

override fun onDataChange(dataSnapshot: DataSnapshot) {

adapter!!.notifyDataSetChanged() //어댑터의 정보업데이트

arrayText.clear()

adapter!!.notifyDataSetChanged()

for(messageData in dataSnapshot.children){ 데이터값을 가져와 뿌려주는역할

val post:textUser = messageData.getValue(textUser::class.java) as textUser

arrayText.add(post)

adapter!!.notifyDataSetChanged()

}

}

})

return view

}

}

**어댑터 연결과정**

var listTodo:List<Todo> = ArrayList()

val myDb:MyDBHeleper= MyDBHeleper(this)

override fun onCreate(savedInstanceState: Bundle?) {

super.onCreate(savedInstanceState)

setContentView(R.layout.activity\_diary)

listTodo=myDb.allTodo()

val adapter=ListTodoAdapter(this,listTodo)

DiaryListView.adapter=adapter

**이메일 전송**

btnMailSend.setOnClickListener { view -> //이메일 전송 리스너 호출

firebaseAuth!!.currentUser!!.sendEmailVerification()

.addOnCompleteListener{

if(it.isSuccessful){

toast("메일이 전송됨")

txtResult.setText("이메일 : "+userEmail + "\n 메일이 전송되었습니다 확인해주세요.")

}else {

toast("메일이 전송되지 않았습니다.")

txtResult.setText("메일이 전송되지 않았습니다 . \n 잠시후 다시시도 혹은 일일 메일 인증횟수를 초과로인한 실패로 다른계정을 이용해주세요.")

}

}

}

**이메일 인증 유무**

btnChecking.setOnClickListener { view ->

firebaseAuth = FirebaseAuth.getInstance() //파이어베이스를 가져옴

user.reload()

if(user.isEmailVerified){ //여기서 인증유무 체크

toast("인증완료 메인으로 이동합니다.")

val intent = Intent(this,RealMainActivity::class.java)

intent.setFlags(Intent.FLAG\_ACTIVITY\_NO\_HISTORY)

startActivity(intent)

finish()

}

**회원 가입**

firebaseAuth!!.createUserWithEmailAndPassword(edtSignEmail.text.toString(),edtSignPw.text.toString()) // 파이어베이스 유저생성메소드 호출후 결과값으로 성공실패를 구분

.addOnCompleteListener(this){

if(it.isSuccessful){

Toast.makeText(this, "가입성공", Toast.LENGTH\_SHORT).show()

finish()

}else {

Toast.makeText(this,"중복된 이메일로 인한 가입실패입니다.", Toast.LENGTH\_SHORT).show()

}

}

**웹뷰**

mwebview.apply { 웹뷰 접근 세팅

webViewClient = WebViewClient()

settings.javaScriptEnabled = true

}

btnMovingUrl.setOnClickListener { view ->

mwebview.loadUrl(EdtUrl.text.toString()) //url로드를 해와서 이동

}

btnPretingUrl.setOnClickListener { view ->

mwebview.goBack() //이전 url로 이동해주는기능

}

mwebview.loadUrl("https://www.naver.com") //해당 url을 로드해줌

// 처리

return view

}

}

**어댑터 클래스**

class ListTextAdapter (val context: Context, val listText:List<textUser>): BaseAdapter() {

// 베이스 어댑터를 상속받음

override fun getView(p0: Int, p1: View?, p2: ViewGroup?): View { //화면에뿌려주는역할

val rowView = View.inflate(context, R.layout.item, null)

rowView.txtText.text=listText[p0].text.toString()

rowView.imgDelete.setOnClickListener {

val firebaseDatabase = FirebaseDatabase.getInstance() //파이어베이스값을가져옴

var databaseReference = firebaseDatabase.getReference("test")// 파이어베이스 db의 test값을 가져옴

databaseReference.child(listText[p0].key.toString()).removeValue().addOnCompleteListener{

Toast.makeText(context,"삭제됨",Toast.LENGTH\_SHORT).show()

}

}

return rowView

}

override fun getItem(p0: Int): Any { //해당 포지션 아이템을 가져오지만 상속을 받으면 언급을 해줘야 하고 딱히 사용하지는않음

return listText[p0]

}

override fun getItemId(p0: Int): Long { //해당 포지션 아이템의 아이디를 가져오지만 상속을 받으면 언급을 해줘야 하고 딱히 사용하지는않음

return 0

}

override fun getCount(): Int { //리스트의 총 크기를 가져오지만 상속을 받으면 언급을 해줘야 하고 딱히 사용하지는않음

return listText.size

}

}

}

**SQLite 이용 과정**

class MyDBHeleper(context:Context): SQLiteOpenHelper(context, DATABASE\_NAME,null,DATABASE\_VER) {

companion object{

private val DATABASE\_NAME="todo.db" //db이름

private val DATABASE\_VER=1 //db버전

private val TABLE\_NAME ="todo" //테이블이름

}

override fun onCreate(p0: SQLiteDatabase?) {

p0!!.execSQL("create table $TABLE\_NAME(id integer primary key autoincrement, title text, message String, writedate long)") //생성

}

override fun onUpgrade(p0: SQLiteDatabase?, p1: Int, p2: Int) {

p0!!.execSQL("drop table if exists $TABLE\_NAME") //업그레이드 경우 기존의버전 삭제후 재생성

onCreate(p0!!)

}

**SQLite 에서 adapter설정**

fun allTodo():List<Todo>{ // Todo 객체를 가져와설정

val listTodo=ArrayList<Todo>()

val sql ="select \* from $TABLE\_NAME"

val db= this.writableDatabase

val cursor = db.rawQuery(sql,null)

if(cursor.moveToFirst()){ // 찾으면서 커서 이동식으로 다음 데이터를 가져옴

do{

val todo= Todo()

todo.title=cursor.getString(1)

todo.message= cursor.getString(2)

todo.writedate= cursor.getLong(3)

listTodo.add(todo)

}

while (cursor.moveToNext())

}

db.close()

return listTodo

}

fun addTodo(todo:Todo){

val db= this.writableDatabase

val values= ContentValues()

values.put("title", todo.title)

values.put("message", todo.message)

values.put("writedate",todo.writedate)

db.insert(TABLE\_NAME,null,values)

db.close()

}

fun updateTodo(todo:Todo){ // alltodo와 비슷하지만 직접 매개변수를 주어 값을 업데이트

val db= this.writableDatabase

val values= ContentValues()

values.put("title", todo.title)

values.put("message", todo.message)

values.put("writedate",todo.writedate)

db.update(TABLE\_NAME,values,"title=?", arrayOf(todo.title))

db.close()

}

fun deleteTodo(todo:Todo){ //검색하여 조건에 맞는 값을 삭제하는 기능

val db=this.writableDatabase

db.delete(TABLE\_NAME,"title=?", arrayOf(todo.title))

db.close()

}

}

**프래그먼트 세팅**

var supportManager = supportFragmentManager

supportManager.beginTransaction() //트랜젝션시작

.replace(R.id.fContainer,ChatFragment()) //해당 프래그먼트 교체

.commit() //반영