Modul 5

Pemrograman Berbasis Platform

"*Room with SQLite*"

Mobile Programming
Android Kotlin

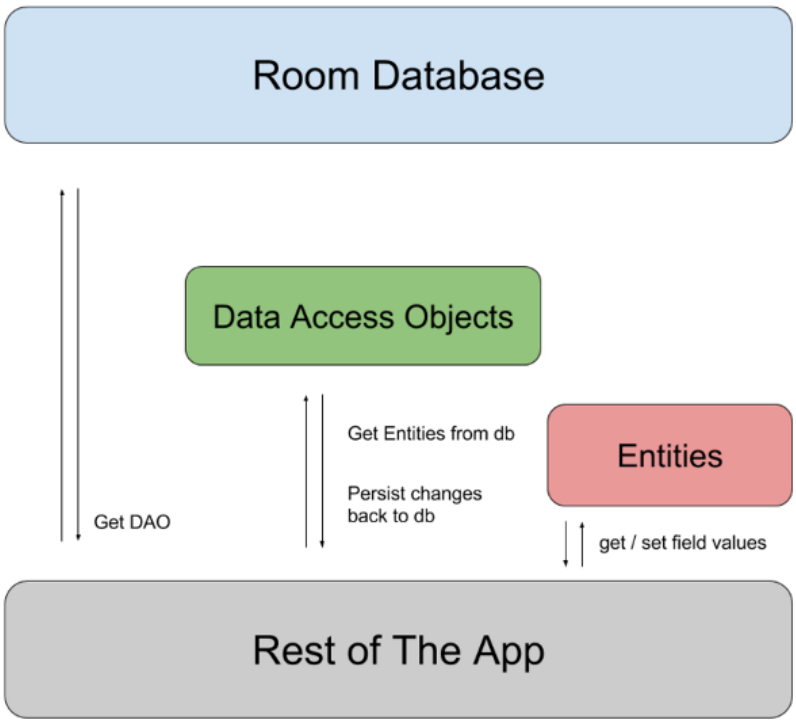
By Chris Yustianto Putra Tangdialla

PROGRAM STUDI INFORMATIKA
FAKULTAS TEKNOLOGI INDUSTRI
UNIVERSITAS ATMA JAYA YOGYAKARTA
TAHUN AJARAN 2022/2023

Data Binding & Persistent Data
Part 2

A. Pengenalan Room

Setelah kita pelajari *Shared Preferenced,* kita bisa memahami bagaimana cara data bisa disimpan dalam database lokal. Dengan *shared preference,* kita bisa menyimpan data-data yang bersifat primitif dengan leluasa. Namun ada beberapa permasalah yang terjadi jika menympan data yang banyak, dan semuanya adalah objek. Hal ini menambah kompleks pembuatan inisialisasi edit untuk menyimpan dan menghapus data. Oleh karena itu, muncullah data persistent yang bernama Room.Room ini dalam kutipan dari website resmi android developer adalah "*Room provides an abstraction layer over SQLite to allow fluent database access while harnessing the full power of SQLite*", kalau diartikan dalam Bahasa Indonesia adalah "Room menyediakan sebuah layer abstract dalam bentuk SQLite untuk mengizinkan untuk database mengakses ketika kita memanfaatkan kelebihan dari SQLite ". simplenya Room adalah salah satu fitur Library dari Android Jetpack, yang mempermudah dalam penggunaan database SQLite pada Android. Sebelum kita bahas cara penggunaan codenya, kita akan sama-sama membedah struktur dari Room.



Dalam Room Android Jetpack, para developer sepakat untuk membuat sebuah struktur konfigurasi dalam pembuatan Room, yaitu Entity, Database, dan Dao (Data Access Object).

a. Entity

   Dibuat dalam bentuk class file. Berfungsi sebagai representasi dari table pada database, dideklarasikan pula kolom-kolom yang diperlukan.

b. Database

   Dibuat dalam bentuk file class. Berfungsi untuk men-extend class RoomDatabase. Biasanya dideklarasikan entity/table yang akan dibuat dan juga versi dari database

c. DAO (Data Access Object)

   Dibuat dalam file interface. Dibuat untuk mengakses data-data yang ada pada database seperti *query* untuk *update, read, delete, dll.*

Class database menyediakan aplikasi Anda dengan instance DAO yang terkait dengan database tersebut. Selanjutnya, aplikasi dapat menggunakan DAO untuk mengambil data dari database sebagai instance dari objek entity data terkait. Aplikasi juga dapat menggunakan entity data yang ditentukan untuk memperbarui baris dari tabel yang sesuai atau membuat baris baru untuk penyisipan. Berikut contoh pemakaian dalam tiap komposisi Room.

1. Entity Data

```
@Entity
data class User(
    @PrimaryKey val uid: Int,
    @ColumnInfo(name = "first_name") val firstName: String?,
    @ColumnInfo(name = "last_name") val lastName: String?
)
```

2. Dao

```
@Dao
interface UserDao {
    @Query("SELECT * FROM user")
    fun getAll(): List<User>

    @Query("SELECT * FROM user WHERE uid IN (:userIds)")
    fun loadAllByIds(userIds: IntArray): List<User>

    @Query("SELECT * FROM user WHERE first_name LIKE :first AND " +
           "last_name LIKE :last LIMIT 1")
    fun findByName(first: String, last: String): User

    @Insert
    fun insertAll(vararg users: User)

    @Delete
```

```
    fun delete(user: User)
}
```

3. Database

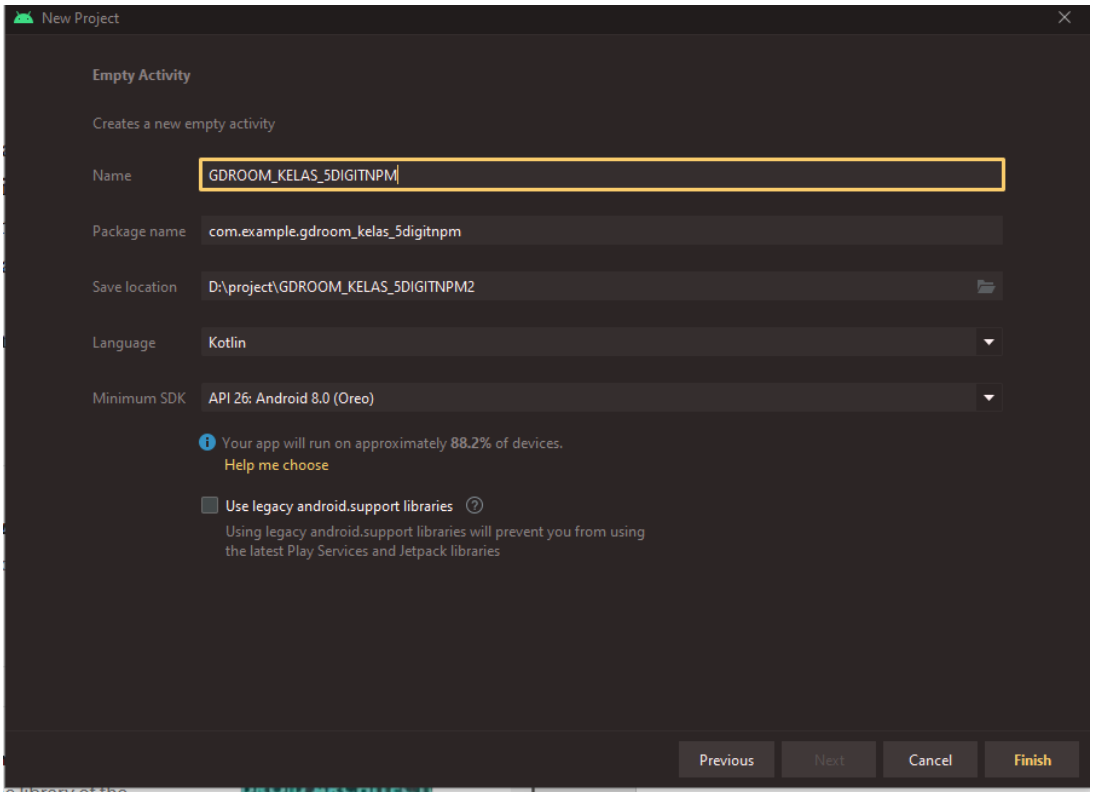
```
abstract fun studentDao(): StudentDao
    companion object {
        private var INSTANCE: StudentDatabase? = null

        fun getInstance(context: Context): StudentDatabase? {
            if (INSTANCE == null) {
                synchronized(StudentDatabase::class) {
                    INSTANCE                                    =
Room.databaseBuilder(context.getApplicationContext(),
                            StudentDatabase::class.java,
"studentdata.db")
                            .build()
                }
            }
            return INSTANCE
        }

        fun destroyInstance() {
            INSTANCE = null
        }
    }
```

perlu untuk diperhatikan pula bahwa untuk melakukan operasi seperti memasukkan data, mengambil data dan lain-lain harus dilakukan di luar *mainthread* dan ini adalah ada aturan *Room* secara *default*. Hal ini disebakan karena melakukan *query* data pada mainthread dapat membuat aplikasi kita menjadi lambat. Oleh karena itu, **sangat disarankan untuk mempelajari Kotlin Coroutines terlebih dahulu.**

B. Guided

Tujuan kali ini adalah membuat sebuah app Note sederhana yang bisa CRUD pada data-data sederhana.

1. Pertama kalian buat dahulu project di android studio kalian dengan format seperti berikut :
   a. Activity : Blank Activity
   b. Name : GDRoom_kelas_5digitnpm
   c. Minimum SDK : API 26 (Oreo 8.0)



2. Setelah kalian membuat project sudah build dengan benar dan selesai, kalian perlu import library room yang sudah disediakan oleh Google Android Developer. Lakukan seperti gambar berikut ini :

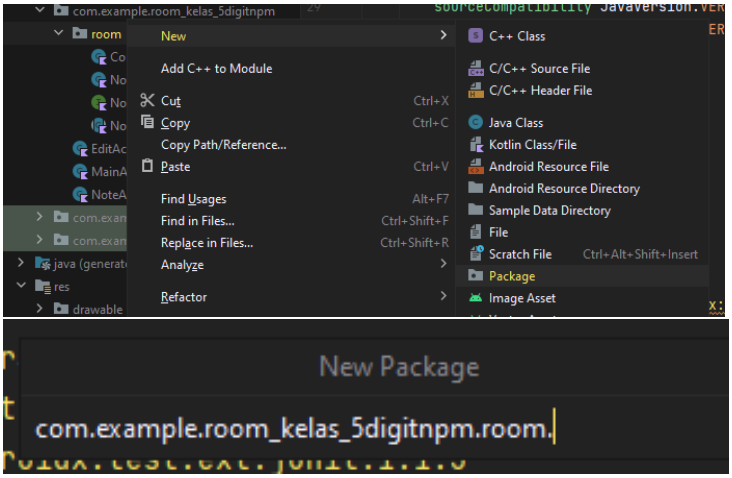
```
plugins {
    id 'com.android.application'
    id 'org.jetbrains.kotlin.android'
    id 'kotlin-android-extensions'
}
apply plugin: 'kotlin-kapt'
```

.....

```
// Room
def room_version = "2.4.3"
implementation "androidx.room:room-runtime:$room_version"
kapt "androidx.room:room-compiler:$room_version"

// Coroutines
def coroutines_version = "1.3.9"
implementation "androidx.room:room-ktx:$room_version"
implementation "org.jetbrains.kotlinx:kotlinx-coroutines
core:$coroutines_version"
```

Lalu kalian *Sync Now*. Tunggu sampai selesai ke build.

3. Kemudian, kalian buka sebuahu package dengan cara pergi ke directory java -> alamat project anda (com.example…..) lalu klik kanan lalu ke new-> package. Lalu isi dengan nama package room.



4. Kemudian kalian buat sebuah file yang berisi entity, Dao, dan Database. Pertama kalian buat dahulu class bernama Note.kt caranya adalah pergi ke package room lalu klik kanan pilih New->Kotlin class/file. Lalu pilih class dengan nama file adalah Note. Kemudian kalian isi dengan code seperti berikut :

```
@Entity
data class Note (
    @PrimaryKey(autoGenerate = true)
    val id: Int,
    val title: String,
    val note: String
    )
```

5. Lalu buat sebuah class interface bernama NoteDao untuk membuat Query yang bisa akses ke Database. Caranya sama seperti no 4 untuk pembuatan file class tapi jenis class adalah interface dengan nama NoteDao. Kemudian kalian ikutin code seperti berikut :

```
@Dao
interface NoteDao {


    @Insert
   suspend fun addNote(note: Note)


    @Update
    suspend fun updateNote(note: Note)


    @Delete
    suspend fun deleteNote(note: Note)


    @Query("SELECT * FROM note")
    suspend fun getNotes() : List<Note>

    @Query("SELECT * FROM note WHERE id =:note_id")
    suspend fun getNote(note_id: Int) : List<Note>
}
```

6. Kemudian, kalian buat lagi sebuah class bernama NoteDB dengan jenis class adalah abstract di package room. Kemudian kalian bisa ikutin code seperti berikut :

```
@Database(
    entities = [Note::class],
    version = 1
)
abstract class NoteDB: RoomDatabase() {

    abstract fun noteDao() : NoteDao

    companion object {

        @Volatile private var instance : NoteDB? = null
        private val LOCK = Any()

        operator fun invoke(context: Context) = instance ?:
synchronized(LOCK){
            instance ?: buildDatabase(context).also {
                instance = it
            }
        }

        private fun buildDatabase(context: Context) =
Room.databaseBuilder(
            context.applicationContext,
            NoteDB::class.java,
            "note12345.db"
        ).build()

    }
}
```

Kemudian kalian buat class lagi dalam package room bernama Constant dengan code seperti berikut :

```
class Constant {
    companion object{
        const val TYPE_READ     = 0
        const val TYPE_CREATE   = 1
        const val TYPE_UPDATE   = 2
    }
}
```

7. Saatnya kalian membuat layout-layout yang dibutuhkan untuk membuat CRUD UI yang bisa dilihat. Kalian buat 2 activity dengan nama sebagai berikut :
   - EditActivity
   - NoteAdapter

   Lalu lakukan code seperti berikut untuk activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/list_note"
        android:layout_width="0dp"
        android:layout_height="0dp"
        app:layout_constraintBottom_toTopOf="@+id/button_create"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        />

    <Button
        android:id="@+id/button_create"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:backgroundTint="@color/yellow"
        android:textColor="@color/black"
        android:text="Tulis Catatan"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintBottom_toBottomOf="parent"
        android:layout_margin="10dp"
        />


</androidx.constraintlayout.widget.ConstraintLayout>
```

Kemudian kalian code xml layout pada bagian activity_edit.xml

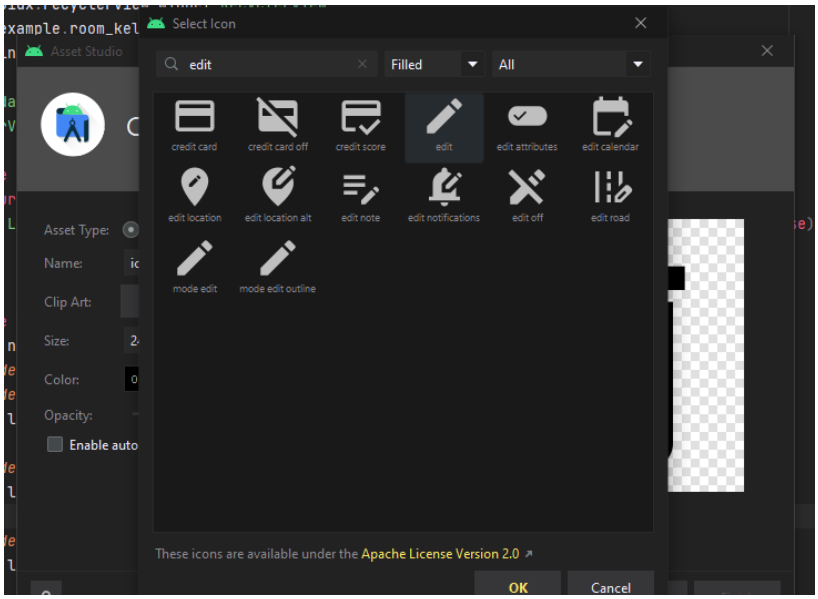
```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="20dp"
    tools:context=".EditActivity">

    <EditText
        android:id="@+id/edit_title"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:hint="Judul"
        android:minHeight="48dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
    <EditText
        android:id="@+id/edit_note"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:hint="Tulis Catatan"
        android:minLines="3"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/edit_title"
        android:layout_marginTop="10dp"
        android:gravity="top"
        />
    <Button
        android:id="@+id/button_save"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:text="SAVE"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/edit_note"
        android:layout_marginTop="20dp"
        />
    <Button
        android:id="@+id/button_update"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:text="UPDATE"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/button_save"
        android:layout_marginTop="20dp"
        />

</androidx.constraintlayout.widget.ConstraintLayout>
```

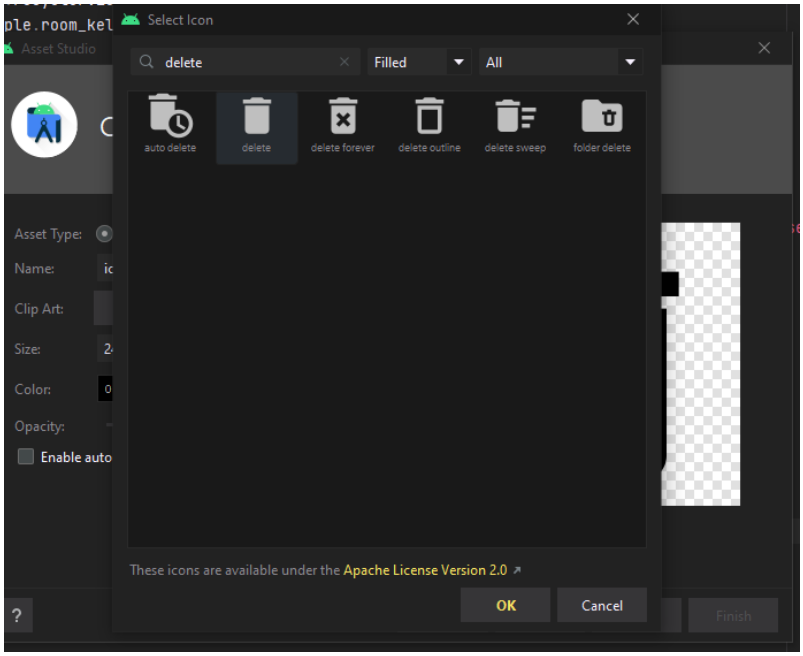
Terakhir untuk adapter_note.xml berikut :

```
```

```
<?xml                 version="1.0"                 encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:padding="10dp">


    <TextView
        android:id="@+id/text_title"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        tools:text="Nanti       kita       cerita       hari       ini"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toStartOf="@+id/icon_edit"
        />
    <ImageView
        android:id="@+id/icon_edit"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@drawable/ic_edit"
        android:padding="10dp"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintEnd_toStartOf="@+id/icon_delete"
        />
    <ImageView
        android:id="@+id/icon_delete"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@drawable/ic_delete"
        android:padding="10dp"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Lalu pada adapter_note.xml pasti error pada src. Itu karena kalian belum import icon yang dibutuhkan oleh project ini, caranya adalah kalian pergi ke res->drawable. Lalu klik kanan pilih New -> new Vector Asset lalu klik gambar pada Clip Art. Kalian bisa cari icon di search bar dengan ketik edit lalu rename dengan nama ic_edit. Lakukan juga untuk icon delete.





8. Setelah semua sudah selesai, kalian bisa pergi ke class NoteAdapter.kt. disini kalian Kembali dengan konsep MVVM namun tidak memakai LiveData karena datanya bersifat statis dan hanya membuat 1 database saja. Lalu kalian lakukan code seperti berikut :

```
class NoteAdapter (private val notes: ArrayList<Note>, private val
listener: OnAdapterListener) :
    RecyclerView.Adapter<NoteAdapter.NoteViewHolder>() {

    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int):
NoteViewHolder {
        return NoteViewHolder(

LayoutInflater.from(parent.context).inflate(R.layout.adapter_note,paren
t, false)
        )
    }

    override fun onBindViewHolder(holder: NoteViewHolder, position:
Int) {
        val note = notes[position]
        holder.view.text_title.text = note.title
        holder.view.text_title.setOnClickListener{
            listener.onClick(note)
        }
        holder.view.icon_edit.setOnClickListener {
            listener.onUpdate(note)
        }
        holder.view.icon_delete.setOnClickListener {
            listener.onDelete(note)
        }

    }

    override fun getItemCount() = notes.size

    inner class NoteViewHolder( val view: View) :
RecyclerView.ViewHolder(view)

    @SuppressLint("NotifyDataSetChanged")
    fun setData(list: List<Note>){
        notes.clear()
        notes.addAll(list)
        notifyDataSetChanged()
    }

    interface OnAdapterListener {
        fun onClick(note: Note)
        fun onUpdate(note: Note)
        fun onDelete(note: Note)
    }
}
```

kemudian kalian pergi ke class EditActivity.kt denga nisi code seperti berikut :

```
class EditActivity : AppCompatActivity() {

    val db by lazy { NoteDB(this) }
    private var noteId: Int = 0

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_edit)
        setupView()
        setupListener()
//
//        Toast.makeText(this,
noteId.toString(),Toast.LENGTH_SHORT).show()
    }

    fun setupView(){
        supportActionBar!!.setDisplayHomeAsUpEnabled(true)
        val intentType = intent.getIntExtra("intent_type", 0)
        when (intentType){
            Constant.TYPE_CREATE -> {
                button_update.visibility = View.GONE
            }
            Constant.TYPE_READ -> {
                button_save.visibility = View.GONE
                button_update.visibility = View.GONE
                getNote()
            }
            Constant.TYPE_UPDATE -> {
                button_save.visibility = View.GONE
                getNote()

            }
        }
    }


    private fun setupListener() {
        button_save.setOnClickListener {
            CoroutineScope(Dispatchers.IO).launch {
                db.noteDao().addNote(
                    Note(0,edit_title.text.toString(),
edit_note.text.toString())
                )
                finish()
            }
        }

        button_update.setOnClickListener {
            CoroutineScope(Dispatchers.IO).launch {
                db.noteDao().updateNote(
                    Note(noteId, edit_title.text.toString(),
edit_note.text.toString())
                )
                finish()
            }
        }
    }
```

```
    fun getNote() {
        noteId = intent.getIntExtra("intent_id", 0)
        CoroutineScope(Dispatchers.IO).launch {
            val notes = db.noteDao().getNote(noteId)[0]
            edit_title.setText(notes.title)
            edit_note.setText(notes.note)
        }
    }

    override fun onSupportNavigateUp(): Boolean {
        onBackPressed()
        return super.onSupportNavigateUp()
    }

}
```

terakhir kalian bisa ke class mainActivity.kt dengan code seperti berikut :

```
class MainActivity : AppCompatActivity() {

    val db by lazy { NoteDB(this) }
    lateinit var noteAdapter: NoteAdapter

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        setupListener()
        setupRecyclerView()
    }

    //berfungsi untuk membuat sebuah note status pada button yang
ditekan untuk CRUD yang dilaksanakan
    //ini berhubungan dengan Constant status pada room
    //cara panggil id dengan memanggil fungsi intetnEdit.
    //jika pada fungsi interface adapterListener berubah, maka object
akan memerah error karena penambahan fungsi.
    private fun setupRecyclerView() {
        noteAdapter = NoteAdapter(arrayListOf(), object :
NoteAdapter.OnAdapterListener{
            override fun onClick(note: Note) {
                //Toast.makeText(applicationContext, note.title,
Toast.LENGTH_SHORT).show()
                intentEdit(note.id,Constant.TYPE_READ)
            }

            override fun onUpdate(note: Note) {
                intentEdit(note.id, Constant.TYPE_UPDATE)
            }

            override fun onDelete(note: Note) {
                deleteDialog(note)
            }
        })
        list_note.apply {
```

```
            layoutManager = LinearLayoutManager(applicationContext)
            adapter = noteAdapter
        }
    }

    private fun deleteDialog(note: Note){
        val alertDialog = AlertDialog.Builder(this)
        alertDialog.apply {
            setTitle("Confirmation")
            setMessage("Are You Sure to delete this data From
${note.title}?")
            setNegativeButton("Cancel", DialogInterface.OnClickListener
{ dialogInterface, i ->
                dialogInterface.dismiss()
            })
            setPositiveButton("Delete", DialogInterface.OnClickListener
{ dialogInterface, i ->
                dialogInterface.dismiss()
                CoroutineScope(Dispatchers.IO).launch {
                    db.noteDao().deleteNote(note)
                    loadData()
                }
            })
        }
        alertDialog.show()
    }

    override fun onStart() {
        super.onStart()
        loadData()
    }

    //untuk load data yang tersimpan pada database yang sudah create
data
    fun loadData() {
        CoroutineScope(Dispatchers.IO).launch {
            val notes = db.noteDao().getNotes()
            Log.d("MainActivity","dbResponse: $notes")
            withContext(Dispatchers.Main){
                noteAdapter.setData( notes )
            }
        }
    }


     fun setupListener() {
        button_create.setOnClickListener{
            intentEdit(0,Constant.TYPE_CREATE)
        }
     }

    //pick data dari Id yang sebagai primary key
     fun intentEdit(noteId : Int, intentType: Int){
         startActivity(
             Intent(applicationContext, EditActivity::class.java)
                 .putExtra("intent_id", noteId)
                 .putExtra("intent_type", intentType)
```

```
            )
        }

}
```

akhirnya kalian bisa coba jalankan projek kalian dengan emulator kalian. Hasilnya sebagai berikut :

Figure 2 gambar Tampilan Awal (Read)

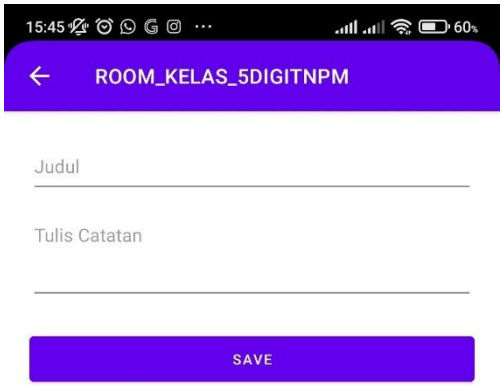
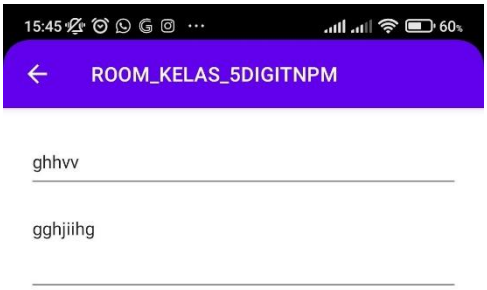


Figure 1 Gambar Create Data (Create)



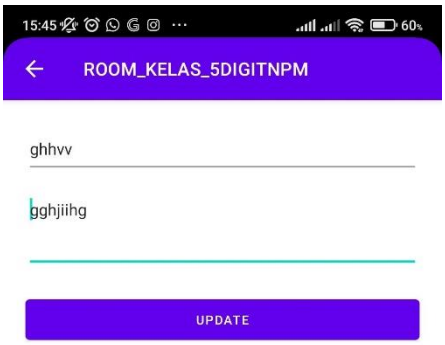


Figure 3 Update Data

Figure 5Ketika ditekan Judul pada salah satu List recyclerview

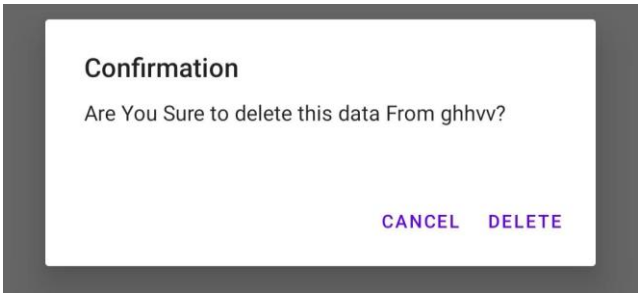


Figure 4 Delete Data