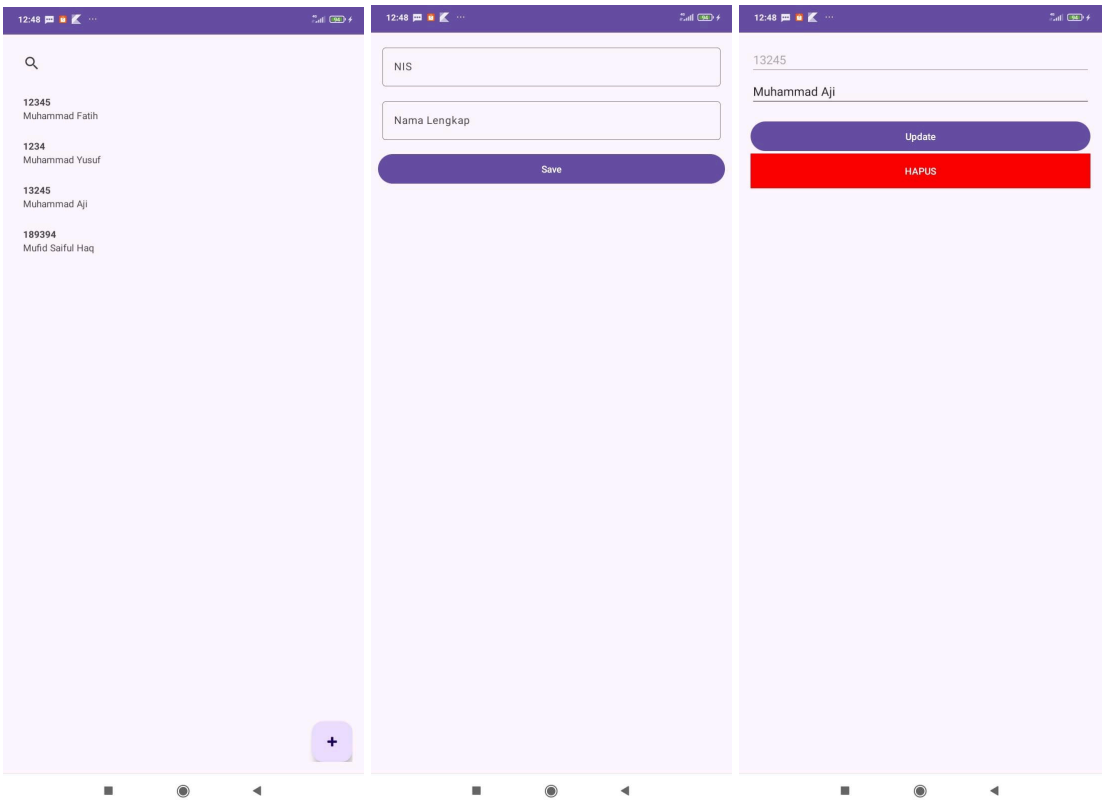
DOKUMENTASI APLIKASI MANAJEMEN DATA SISWA

Project By :
Muhammad Yusuf Sidik Iskandar

PESANTREN TEKNOLOGI INFORMASI DAN KOMUNIKASI
2024/2025

Tampilan Aplikasi Manajemen Siswa



Deskripsi Singkat

Aplikasi Android yang memungkinkan pengguna untuk **menambahkan**, **mengedit**, **menghapus**, dan **mencari** data siswa yang terdiri dari **NIS** dan **Nama Lengkap**. Aplikasi ini menggunakan penyimpanan **Room Database**, serta menampilkan data melalui **RecyclerView**.

Fitur Utama

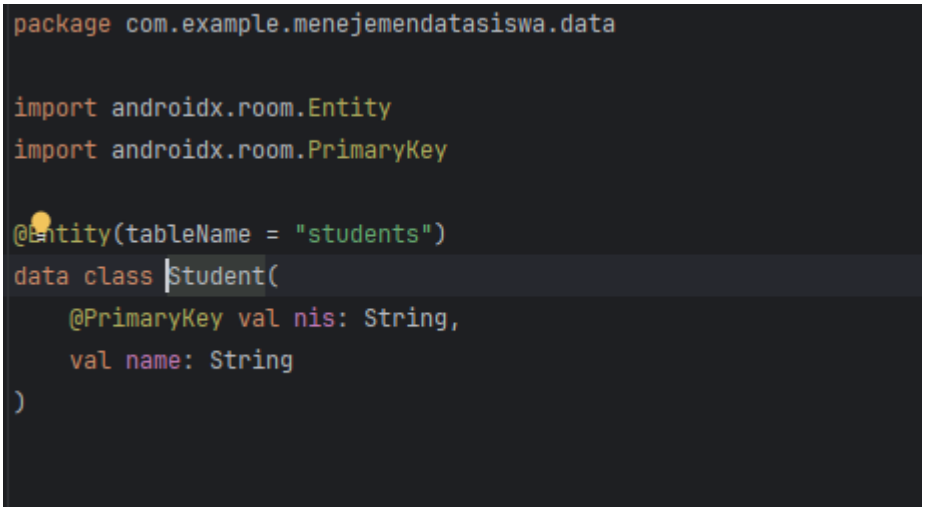
1. Tambah data siswa
2. Edit data siswa
3. Hapus data siswa
4. Tampilkan daftar siswa
5. Pencarian data siswa secara real-time

Teknologi yang Digunakan

1. Kotlin
2. Android Studio
3. Room Database (MVVM Architecture)
4. RecyclerView
5. Material 3 Components
6. LiveData & ViewModel

Penjelasan Komponen Penting

1. Student.kt
   Entity dari Room Database berisi:
   - nis (Primary Key)
   - name (Nama Siswa)



```kotlin
package com.example.menejemendatasiswa.data

import androidx.room.Entity
import androidx.room.PrimaryKey

@Entity(tableName = "students")
data class Student(
    @PrimaryKey val nis: String,
    val name: String
)
```

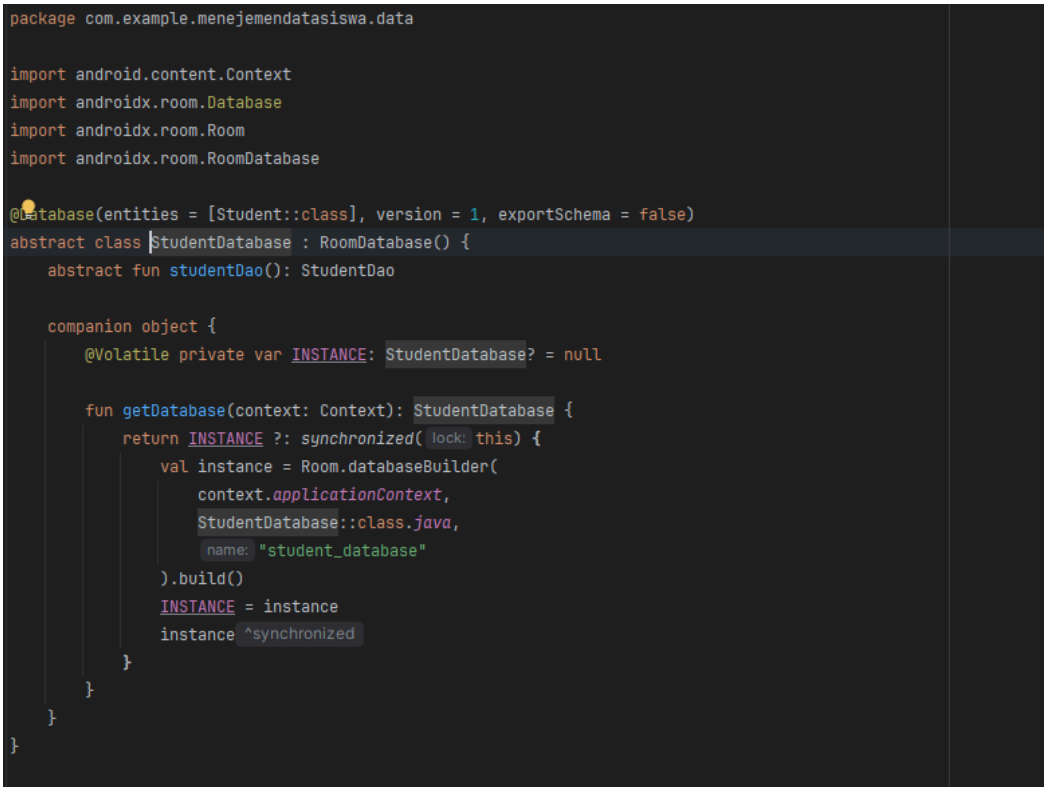
Class **Student** merupakan sebuah *data class* yang berfungsi sebagai representasi dari entitas siswa dalam database. Class ini diberi anotasi **@Entity**, menandakan bahwa **Student** adalah sebuah entitas tabel dalam Room Database dengan nama table **students.** Class ini memiliki dua properti: **nis** yang berperan sebagai primary key dan bertipe **string**, serta **name** yang juga bertipe **string** dan menyimpan nama lengkap siswa. Dengan mendeklarasikan class ini sebagai entitas, Room akan secara otomatis membuat skema tabel berdasarkan properti yang ada.

2. StudentDao.kt
   Interface DAO untuk operasi database:
   - `insertStudent()`
   - `updateStudent()`
   - `deleteStudent()`
   - `getAllStudents()`
   - `searchStudents(query)`

```kotlin
package com.example.menejemendatasiswa.data

import androidx.lifecycle.LiveData
import androidx.room.Dao
import androidx.room.*

@Dao
interface StudentDao {
    @Query("SELECT * FROM students")
    fun getAllStudents(): LiveData<List<Student>>

    @Insert(onConflict = OnConflictStrategy.REPLACE)
    suspend fun insert(student: Student)

    @Query("SELECT * FROM students WHERE name LIKE '%' || :query || '%' OR nis LIKE '%' || :query || '%'")
    fun searchStudents(query: String): LiveData<List<Student>>

    @Update
    suspend fun update(student: Student)

    @Delete
    suspend fun delete(student: Student)
}
```

Interface **StudentDao** merupakan komponen DAO (Data Access Object) yang bertugas menyediakan berbagai method untuk melakukan operasi terhadap tabel siswa. Di dalamnya terdapat beberapa query seperti **getAllStudents()** yang mengembalikan seluruh data siswa sebagai **LiveData**, dan **searchStudents(query)** yang memungkinkan pencarian siswa berdasarkan nama atau NIS dengan menggunakan perintah SQL **LIKE**. Selain itu, terdapat method **insert()** untuk menyimpan data baru, **update()** untuk memperbarui data yang sudah ada, dan **delete()** untuk menghapus data dari database. Semua operasi insert, update, dan delete dijalankan dalam coroutine karena bersifat suspend.

3. StudentDatabase.kt
   Singleton Room Database.



```kotlin
package com.example.menejemendatasiswa.data

import android.content.Context
import androidx.room.Database
import androidx.room.Room
import androidx.room.RoomDatabase

@Database(entities = [Student::class], version = 1, exportSchema = false)
abstract class StudentDatabase : RoomDatabase() {
    abstract fun studentDao(): StudentDao

    companion object {
        @Volatile private var INSTANCE: StudentDatabase? = null

        fun getDatabase(context: Context): StudentDatabase {
            return INSTANCE ?: synchronized( lock: this) {
                val instance = Room.databaseBuilder(
                    context.applicationContext,
                    StudentDatabase::class.java,
                     name: "student_database"
                ).build()
                INSTANCE = instance
                instance ^synchronized
            }
        }
    }
}
```

Class **StudentDatabase** adalah implementasi dari **RoomDatabase** yang mendefinisikan database siswa secara keseluruhan. Class ini menggunakan anotasi **@Database** untuk menentukan entitas yang digunakan (**Student**) serta versi database. Di dalamnya terdapat abstract method **studentDao()** yang akan digunakan untuk mengakses DAO. Singleton pattern digunakan melalui **getDatabase()** agar hanya ada satu instance dari database yang aktif selama aplikasi berjalan, mencegah terjadinya konflik dan penggunaan resource yang tidak perlu.

4. StudentViewModel.kt
   Mengelola data siswa secara reaktif (LiveData) antara DAO dan UI.

```kotlin
package com.example.menejemendatasiswa

import android.app.Application
import androidx.lifecycle.AndroidViewModel
import androidx.lifecycle.LiveData
import androidx.lifecycle.viewModelScope
import com.example.menejemendatasiswa.data.Student
import com.example.menejemendatasiswa.data.StudentDatabase
import kotlinx.coroutines.launch

class StudentViewModel(application: Application) : AndroidViewModel(application) {
    private val repository: StudentRepository
    val allStudents: LiveData<List<Student>>

    init {
        val dao = StudentDatabase.getDatabase(application).studentDao()
        repository = StudentRepository(dao)
        allStudents = repository.allStudents
    }

    fun insert(student: Student) = viewModelScope.launch { this: CoroutineScope
        repository.insert(student)
    }

    fun search(query: String): LiveData<List<Student>> {
        return repository.search(query)
    }

    fun update(student: Student) = viewModelScope.launch { this: CoroutineScope
        repository.update(student)
    }

    fun delete(student: Student) = viewModelScope.launch { this: CoroutineScope
        repository.delete(student)
    }

}
```

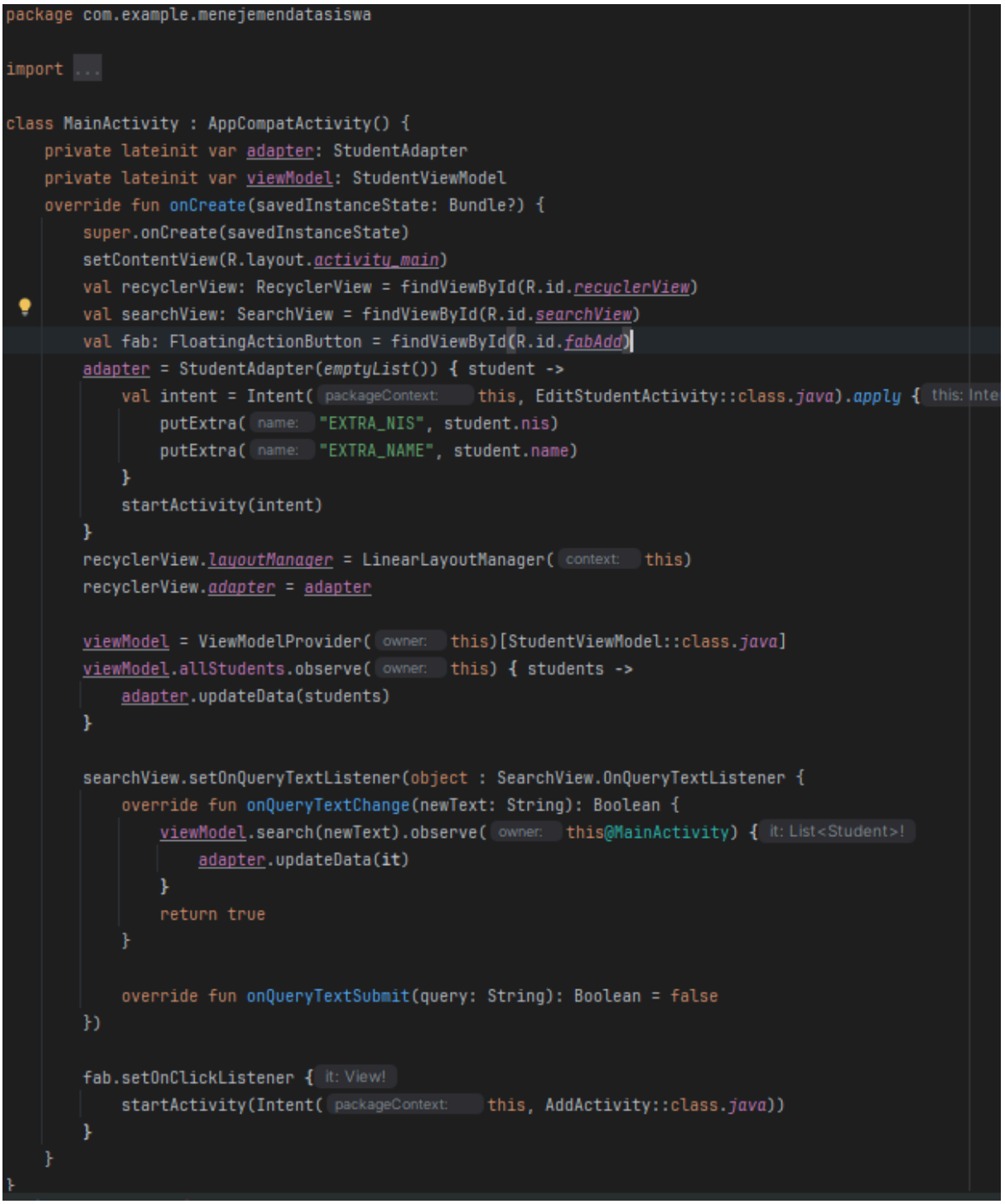
Class **StudentViewModel** adalah turunan dari **AndroidViewModel** yang memiliki peran penting dalam menyediakan data ke UI dan menjaga data tetap bertahan saat terjadi perubahan konfigurasi, seperti rotasi layar. ViewModel ini menginisialisasi repository pada init block dan menyediakan LiveData **allStudents** yang dapat diamati oleh activity. Selain itu, terdapat method seperti **insert(), update(),** dan **delete()** yang dijalankan di dalam coroutine menggunakan **viewModelScope**, sehingga operasi database dilakukan secara asynchronous. Fungsi **search()** mengembalikan hasil pencarian siswa yang juga dibungkus dalam LiveData.

5. StudentAdapter.kt
   Menampilkan semua siswa dalam RecyclerView:
   - Klik item memicu intent untuk edit.
   - Klik tombol delete untuk menghapus.

```kotlin
package com.example.menejemendatasiswa.adapter

import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.TextView
import androidx.recyclerview.widget.RecyclerView
import com.example.menejemendatasiswa.R
import com.example.menejemendatasiswa.data.Student

class StudentAdapter(
    private var studentList: List<Student>,
    private val onItemClicked: (Student) -> Unit
) : RecyclerView.Adapter<StudentAdapter.StudentViewHolder>() {

    inner class StudentViewHolder(itemView: View) : RecyclerView.ViewHolder(itemView) {
        val nisText: TextView = itemView.findViewById(R.id.tvNIS)
        val nameText: TextView = itemView.findViewById(R.id.tvName)

        fun bind(student: Student) {
            nisText.text = student.nis
            nameText.text = student.name
            itemView.setOnClickListener { onItemClicked(student) }
        }
    }

    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): StudentViewHolder {
        val view = LayoutInflater.from(parent.context)
            .inflate(R.layout.item_student, parent, attachToRoot: false)
        return StudentViewHolder(view)
    }

    override fun onBindViewHolder(holder: StudentViewHolder, position: Int) {
        holder.bind(studentList[position])
    }

    override fun getItemCount(): Int = studentList.size

    fun updateData(newList: List<Student>) {
        studentList = newList
        notifyDataSetChanged()
    }
}
```

Class **StudentAdapter** adalah adapter dari RecyclerView yang bertugas untuk menampilkan data siswa dalam daftar. Adapter ini menerima daftar siswa dan callback **onItemClicked** yang akan dijalankan ketika item diklik. Di dalam adapter, terdapat inner class **StudentViewHolder** yang mereferensikan TextView untuk menampilkan NIS dan nama siswa. Fungsi **bind()** akan mengikat data siswa ke tampilan UI. Selain itu, terdapat fungsi **updateData()** untuk memperbarui isi dari adapter ketika data berubah, yang kemudian memanggil **notifyDataSetChanged()** untuk me-refresh tampilan.

6. MainActivity.kt
   Menampilkan semua siswa dalam RecyclerView:
   - Tombol FAB untuk tambah data.
   - SearchView untuk mencari data.
   - Klik item untuk edit.



```
package com.example.menejemendatasiswa

import ...

class MainActivity : AppCompatActivity() {
    private lateinit var adapter: StudentAdapter
    private lateinit var viewModel: StudentViewModel
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        val recyclerView: RecyclerView = findViewById(R.id.recyclerView)
        val searchView: SearchView = findViewById(R.id.searchView)
        val fab: FloatingActionButton = findViewById(R.id.fabAdd)
        adapter = StudentAdapter(emptyList()) { student ->
            val intent = Intent( packageContext: this, EditStudentActivity::class.java).apply { this: Inte
                putExtra( name: "EXTRA_NIS", student.nis)
                putExtra( name: "EXTRA_NAME", student.name)
            }
            startActivity(intent)
        }
        recyclerView.layoutManager = LinearLayoutManager( context: this)
        recyclerView.adapter = adapter

        viewModel = ViewModelProvider( owner: this)[StudentViewModel::class.java]
        viewModel.allStudents.observe( owner: this) { students ->
            adapter.updateData(students)
        }

        searchView.setOnQueryTextListener(object : SearchView.OnQueryTextListener {
            override fun onQueryTextChange(newText: String): Boolean {
                viewModel.search(newText).observe( owner: this@MainActivity) { it: List<Student>!
                    adapter.updateData(it)
                }
                return true
            }

            override fun onQueryTextSubmit(query: String): Boolean = false
        })

        fab.setOnClickListener { it: View!
            startActivity(Intent( packageContext: this, AddActivity::class.java))
        }
    }
}
```

Class **MainActivity** merupakan tampilan utama aplikasi yang menampilkan daftar seluruh siswa dalam bentuk RecyclerView. Di dalamnya, data siswa diobservasi melalui **ViewModel**, dan ditampilkan menggunakan **StudentAdapter**. Activity ini juga dilengkapi dengan **SearchView** yang memungkinkan pengguna mencari siswa

berdasarkan NIS atau nama. Ketika pengguna mengetik di SearchView, ViewModel akan menjalankan pencarian dan hasilnya langsung ditampilkan. Selain itu, terdapat FloatingActionButton yang ketika ditekan akan membuka **AddStudentActivity** untuk menambahkan data siswa baru.

7. AddStudentActivity.kt
   Form input siswa.

```kotlin
package com.example.menejemendatasiswa

import ...

class AddStudentActivity : AppCompatActivity() {
    private lateinit var viewModel: StudentViewModel
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_add_student)

        val etNIS: EditText = findViewById(R.id.edt_nis)
        val etName: EditText = findViewById(R.id.edt_nama_lengkap)
        val btnSave: Button = findViewById(R.id.btn_save_update)

        viewModel = ViewModelProvider( owner: this)[StudentViewModel::class.java]

        btnSave.setOnClickListener { it: View!
            val nis = etNIS.text.toString()
            val name = etName.text.toString()
            if (nis.isNotEmpty() && name.isNotEmpty()) {
                val student = Student(nis, name)
                viewModel.insert(student)
                Toast.makeText( context: this,  text: "Data disimpan", Toast.LENGTH_SHORT).show()
                finish()
            }
        }
    }
}
```

AddStudentActivity adalah activity yang memungkinkan pengguna untuk menambahkan data siswa ke dalam database. Terdapat dua **EditText** untuk mengisi NIS dan nama lengkap siswa. Ketika tombol "Simpan" ditekan, aplikasi akan mengambil input dari kedua field tersebut dan memanggil fungsi **insert()** dari ViewModel untuk menyimpan data tersebut ke database. Jika penyimpanan berhasil, pengguna akan mendapatkan notifikasi berupa **Toast**, dan activity akan tertutup otomatis.

8. EditStudentActivity.kt
   Form update & delete siswa.

```
package com.example.menejemendatasiswa

import ...

class EditStudentActivity : AppCompatActivity() {
    private lateinit var viewModel: StudentViewModel
    private lateinit var nis: String
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_edit_student)

        val etNIS: EditText = findViewById(R.id.etNIS)
        val etName: EditText = findViewById(R.id.etName)
        val btnSave: Button = findViewById(R.id.btnSave)

        viewModel = ViewModelProvider( owner: this)[StudentViewModel::class.java]

        // Ambil data dari Intent
        nis = intent.getStringExtra( name: "EXTRA_NIS") ?: ""
        val name = intent.getStringExtra( name: "EXTRA_NAME") ?: ""

        etNIS.setText(nis)
        etNIS.isEnabled = false
        etName.setText(name)

        // Tombol update
        btnSave.text = "Update"
        btnSave.setOnClickListener { it: View!
            val updatedStudent = Student(nis, etName.text.toString())
            viewModel.update(updatedStudent)
            Toast.makeText( context: this, text: "Data diperbarui", Toast.LENGTH_SHORT).show()
            finish()
        }
```

```
        // Tambahkan tombol hapus ke layout secara dinamis
        val layout = findViewById<LinearLayout>(R.id.layoutAddStudent)
        val btnDelete = Button( context: this).apply { this: Button
            text = "Hapus"
            setBackgroundColor(Color.RED)
            setTextColor(Color.WHITE)
            setOnClickListener { it: View!
                AlertDialog.Builder( context: this@EditStudentActivity)
                    .setTitle("Hapus Data")
                    .setMessage("Yakin ingin menghapus data ini?")
                    .setPositiveButton( text: "Ya") { _, _ ->
                        viewModel.delete(Student(nis, etName.text.toString()))
                        Toast.makeText( context: this@EditStudentActivity, text: "Data dihapus", Toast.LENGTH_SHORT).show()
                        finish()
                    }
                    .setNegativeButton( text: "Batal", listener: null)
                    .show()
            }
        }
        layout.addView(btnDelete)
    }
}
```

EditStudentActivity digunakan untuk mengedit atau menghapus data siswa yang telah ada di database. Data siswa yang akan diedit dikirim melalui intent dari **MainActivity**. Di dalam activity ini, NIS siswa ditampilkan namun tidak dapat diubah, sementara nama masih bisa disunting. Tombol "Update" digunakan untuk memperbarui nama siswa, dan perubahan tersebut dikirim ke ViewModel melalui method **update()**. Activity ini juga secara dinamis menambahkan tombol "Hapus" ke dalam layout. Ketika tombol tersebut ditekan, akan muncul dialog konfirmasi sebelum data benar-benar dihapus dari database.

9. StudentRepository.kt

```kotlin
package com.example.menejemendatasiswa

import androidx.lifecycle.LiveData
import com.example.menejemendatasiswa.data.Student
import com.example.menejemendatasiswa.data.StudentDao

class StudentRepository (private val studentDao: StudentDao) {
    val allStudents: LiveData<List<Student>> = studentDao.getAllStudents()

    suspend fun insert(student: Student) {
        studentDao.insert(student)
    }

    fun search(query: String): LiveData<List<Student>> {
        return studentDao.searchStudents(query)
    }

    suspend fun update(student: Student) {
        studentDao.update(student)
    }

    suspend fun delete(student: Student) {
        studentDao.delete(student)
    }

}
```

Class **StudentRepository** berfungsi sebagai perantara antara ViewModel dan DAO. Repository ini bertanggung jawab untuk mengatur logika pengambilan data dari database, serta menyediakan interface yang bersih untuk digunakan oleh ViewModel. Dalam class ini, **allStudents** menyimpan semua data siswa dalam bentuk LiveData. Selain itu, terdapat fungsi **insert(), update(),** dan **delete()** yang masing-masing akan menjalankan operasi penyimpanan, pembaruan, dan penghapusan data siswa. Fungsi **search(query)** digunakan untuk melakukan pencarian berdasarkan input pengguna.

Alur Penggunaan

1. Aplikasi menampilkan daftar siswa dari database.
2. User dapat:
    - Menambah siswa lewat tombol + (FAB).
    - Klik item untuk mengedit data.
    - Hapus data dari daftar.
    - Cari siswa lewat SearchView.
3. Semua data disimpan secara lokal menggunakan Room.

Catatan Tambahan

- Pencarian berjalan real-time saat pengguna mengetik.
- Setiap perubahan data (tambah/edit/hapus) langsung disimpan ke Room dan ditampilkan secara live melalui LiveData.
- UI bersih dan responsif menggunakan Material Design.