DOKUMENTASI APLIKASI ANDROID – MyStudent

1. Nama Aplikasi

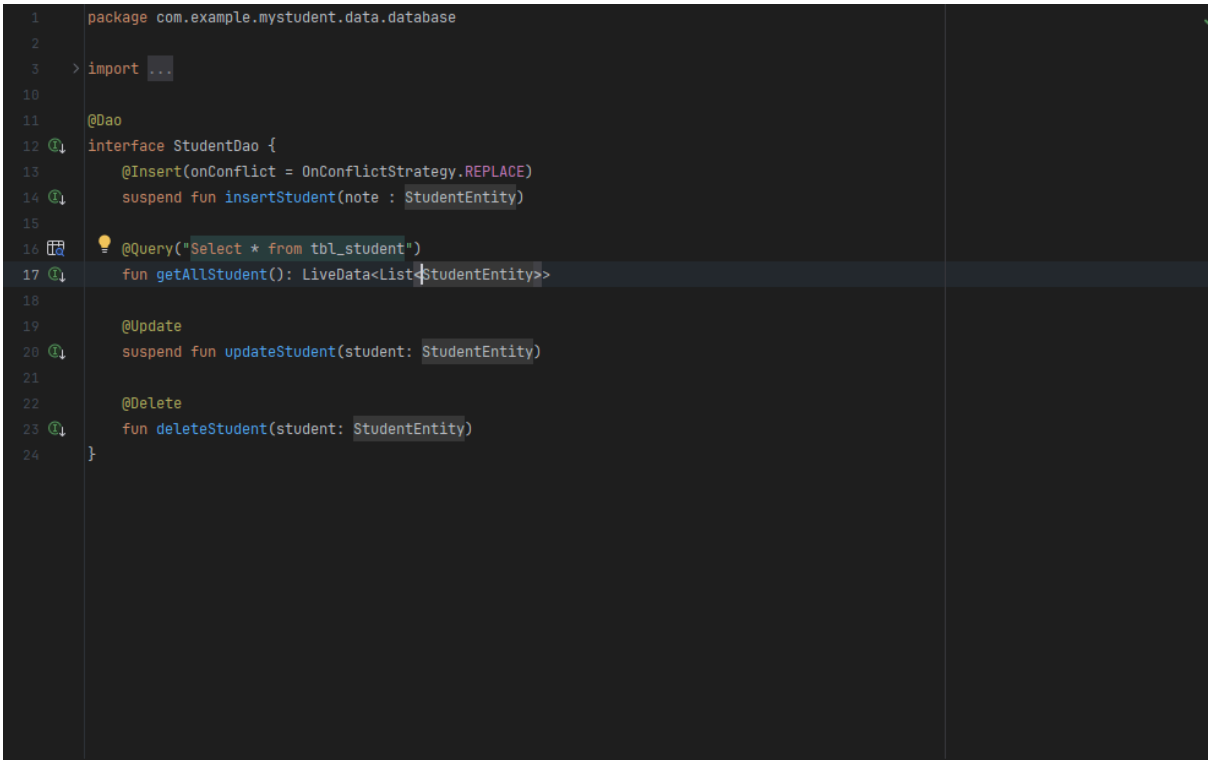
MyStudent
Aplikasi manajemen data siswa berbasis Android.

2. Deskripsi Singkat

MyStudent adalah aplikasi Android yang digunakan untuk menyimpan, mengedit, dan menghapus data siswa berupa nama dan NIS. Aplikasi ini menggunakan arsitektur MVVM dan menyimpan data secara lokal menggunakan Room Database.

3. Struktur Package & Penjelasan Kelas

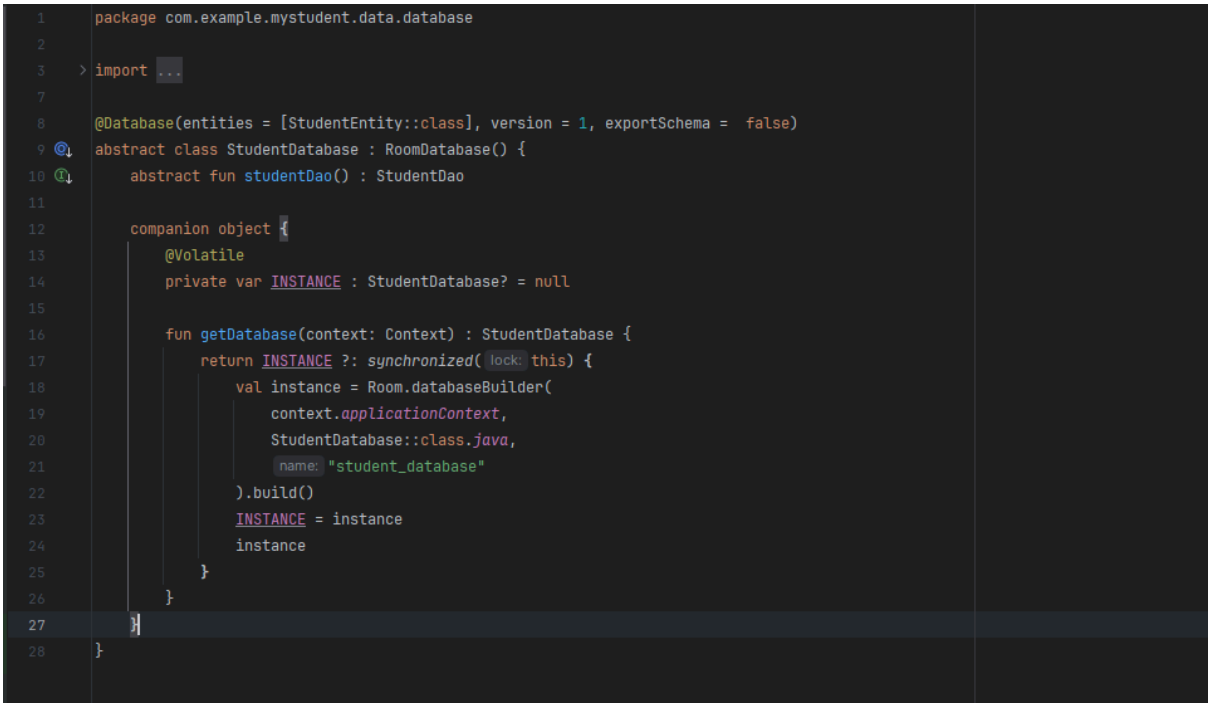
- **StudentDao.kt**



```kotlin
package com.example.mystudent.data.database

import ...

@Dao
interface StudentDao {
    @Insert(onConflict = OnConflictStrategy.REPLACE)
    suspend fun insertStudent(note : StudentEntity)

    @Query("Select * from tbl_student")
    fun getAllStudent(): LiveData<List<StudentEntity>>

    @Update
    suspend fun updateStudent(student: StudentEntity)

    @Delete
    fun deleteStudent(student: StudentEntity)
}
```

Class ini merepresentasikan data siswa yang akan disimpan dalam database lokal menggunakan Room. Setiap objek dari class ini menggambarkan satu baris data dalam tabel bernama "tbl_student" yang terdiri dari tiga atribut utama yaitu id, name, dan nis. Atribut id berfungsi sebagai primary key dan diatur agar ter-generate otomatis, sedangkan name menyimpan nama siswa, dan nis adalah nomor induk siswa. Class ini juga sudah dibuat parcelable agar bisa dikirim melalui intent antar activity.

- **StudentDatabase.kt**



```
package com.example.mystudent.data.database

import ...

@Database(entities = [StudentEntity::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
            }
        }
    }
}
```

Class ini merupakan abstract class yang menjadi tempat utama konfigurasi Room Database. Class ini menyediakan satu-satunya instance database yang bisa diakses oleh seluruh aplikasi. Untuk mencegah terbentuknya lebih dari satu instance, digunakan pola singleton. Di dalamnya juga terdapat fungsi untuk mendapatkan akses ke StudentDao. Database ini diberi nama "student_database" dan dibangun menggunakan Room dengan konfigurasi yang sudah disesuaikan.

- **StudenEntity.kt**

```
package com.example.mystudent.data.database

import ...

@Entity(tableName = "tbl_student")
@Parcelize
data class StudentEntity (
    @field:ColumnInfo(name = "id")
    @field:PrimaryKey(autoGenerate = true)
    val id : Int = 0,

    @field:ColumnInfo(name = "name")
    val name : String = "Unknown",

    @field:ColumnInfo(name = "nis")
    val nis : String = "unknown"

) : Parcelable
```

Class ini merepresentasikan data siswa yang akan disimpan dalam database lokal menggunakan Room. Setiap objek dari class ini menggambarkan satu baris data dalam tabel bernama "tbl_student" yang terdiri dari tiga atribut utama yaitu id, name, dan nis. Atribut id berfungsi sebagai primary key dan diatur agar ter-generate otomatis, sedangkan name menyimpan nama siswa, dan nis adalah nomor induk siswa. Class ini juga sudah dibuat parcelable agar bisa dikirim melalui intent antar activity.

- **StudentRepository.kt**

```kotlin
import ...

class StudentRepository(private val studentDao: StudentDao) {

    fun getAllStudent() : LiveData<List<StudentEntity>>{
        return studentDao.getAllStudent()
    }

    suspend fun insertStudent(studentEntity: StudentEntity){
        val safeStudent = StudentEntity(
            name = studentEntity.name,
            nis =  studentEntity.nis
        )
        studentDao.insertStudent(safeStudent)
    }

    suspend fun updateStudent(studentEntity: StudentEntity){
        val safeStudent = StudentEntity(
            id = studentEntity.id,
            name = studentEntity.name,
            nis = studentEntity.nis
        )
        studentDao.updateStudent(safeStudent)
    }

    fun deleteStudent(studentEntity: StudentEntity){
        studentDao.deleteStudent(studentEntity)
    }
}
```

Class ini bertindak sebagai penghubung antara ViewModel dan data dari database. Repository ini berfungsi untuk mengatur alur pengambilan dan manipulasi data sehingga ViewModel tidak perlu berurusan langsung dengan database. Semua permintaan data atau perintah penyimpanan, penghapusan, dan pembaruan data siswa didelegasikan melalui class ini. Dengan adanya repository, logika bisnis menjadi lebih terstruktur dan modular.

- **StudentViewModel.kt**

```kotlin
package com.example.mystudent.data.repository

import ...

class StudentViewModel(application: Application) : AndroidViewModel(application) {
    private val studentDao = StudentDatabase.getDatabase(application).studentDao()
    private val repository = StudentRepository(studentDao)

    val student : LiveData<List<StudentEntity>> = repository.getAllStudent()

    fun insertStudent(studentEntity: StudentEntity) {
        viewModelScope.launch {
            repository.insertStudent(studentEntity)
        }
    }

    fun updateStudent(studentEntity: StudentEntity) {
        viewModelScope.launch(Dispatchers.IO) {
            repository.updateStudent(studentEntity)
        }
    }

```

```
    fun deleteStudent(studentEntity: StudentEntity) {
        viewModelScope.launch(Dispatchers.IO) {
            repository.deleteStudent(studentEntity)
        }
    }
}
```

Class ini digunakan untuk mengelola dan menyediakan data ke komponen UI dalam aplikasi. ViewModel ini mengamati data dari repository menggunakan LiveData dan memastikan data tersebut tetap bertahan meskipun terjadi perubahan konfigurasi pada activity. Selain itu, proses insert, update, dan delete dijalankan dalam coroutine agar tidak mengganggu thread utama aplikasi. Dengan cara ini, ViewModel membantu menjaga kestabilan UI dan meningkatkan performa aplikasi.

- **AdapterStudent.kt**

```
package com.example.mystudent.ui.Adapter

import ...

class AdapterStudent(private val onClick: (StudentEntity) -> Unit) :
    ListAdapter<StudentEntity, AdapterStudent.NoteViewHolder>(DIFF_CALLBACK){
    class NoteViewHolder(private val binding: ItemStudentBinding) : RecyclerView.ViewHolder(binding.root) {
        fun bind(studentEntity: StudentEntity, onClick: (StudentEntity) -> Unit) {
            binding.tvName.text = studentEntity.name
            binding.tvNis.text = studentEntity.nis
            binding.root.setOnClickListener{ onClick(studentEntity) }
        }
    }

    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): NoteViewHolder {
        val binding = ItemStudentBinding.inflate(LayoutInflater.from(parent.context), parent, attachToParent: false)
        return NoteViewHolder(binding)
    }

    override fun onBindViewHolder(holder: NoteViewHolder, position: Int) {
        val note = getItem(position)
        holder.bind(note, onClick)
    }

    companion object{
        val DIFF_CALLBACK = object  : DiffUtil.ItemCallback<StudentEntity>() {
            override fun areItemsTheSame(oldItem: StudentEntity, newItem: StudentEntity): Boolean {
                return oldItem.id == newItem.id
            }
```

```
    companion object{
        val DIFF_CALLBACK = object  : DiffUtil.ItemCallback<StudentEntity>() {
            override fun areItemsTheSame(oldItem: StudentEntity, newItem: StudentEntity): Boolean {
                return oldItem.id == newItem.id
            }

            override fun areContentsTheSame(oldItem: StudentEntity, newItem: StudentEntity): Boolean {
                return oldItem == newItem
            }

        }
    }

}
```

Class ini berfungsi untuk menampilkan daftar data siswa ke dalam RecyclerView. Adapter ini mengatur bagaimana setiap item siswa ditampilkan dan memperbarui tampilan saat data berubah. Untuk mengoptimalkan proses pembaruan data pada list, digunakan teknik perbandingan otomatis antar item menggunakan bantuan dari DiffUtil, sehingga hanya item yang berubah saja yang diperbarui. Adapter ini juga menyediakan fungsi saat item diklik, yang biasanya akan membuka halaman detail siswa.

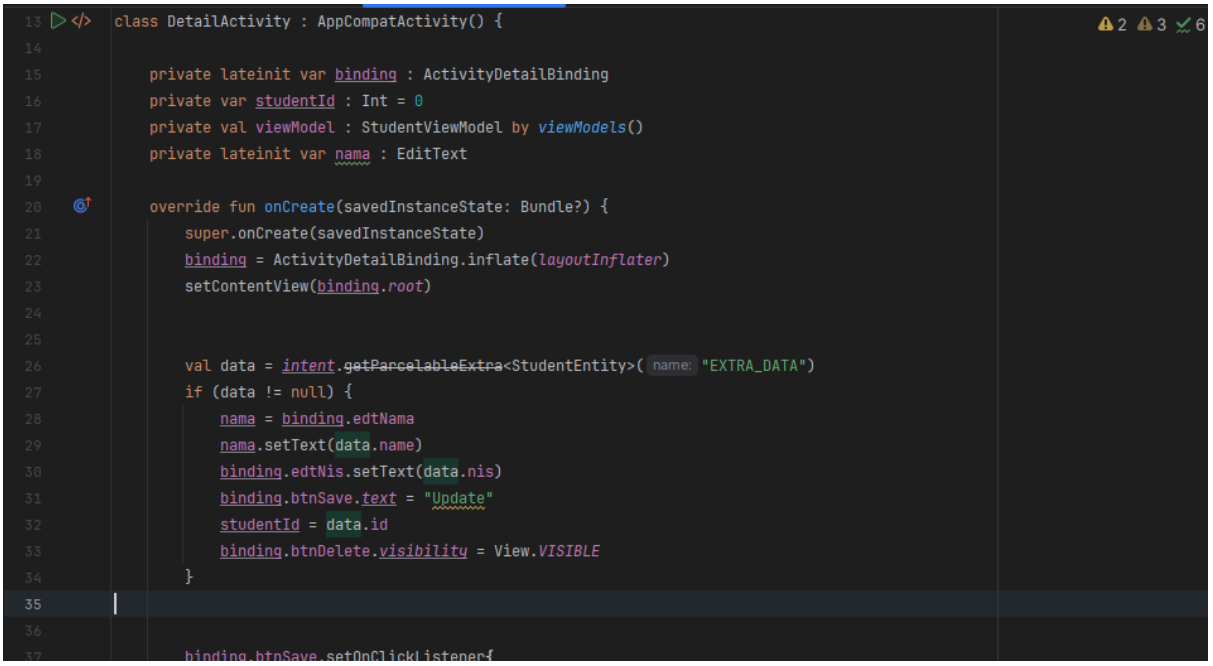
- **MainActivitiy.kt**

```
13
14  class MainActivity : AppCompatActivity() {
15
16      private lateinit var binding: ActivityMainBinding
17      private val viewModel : StudentViewModel by viewModels()
18      private var originalList = listOf<StudentEntity>()
19      private lateinit var adapterStudent: AdapterStudent
20
21      override fun onCreate(savedInstanceState: Bundle?) {
22          super.onCreate(savedInstanceState)
23          binding = ActivityMainBinding.inflate(layoutInflater)
24          setContentView(binding.root)
25
26          adapterStudent = AdapterStudent { studentEntity ->
27              val intent = Intent( packageContext: this, DetailActivity::class.java)
28              intent.putExtra( name: "EXTRA_DATA", studentEntity)
29              startActivity(intent)
30          }
31
32          binding.rvStudent.layoutManager = LinearLayoutManager( context: this)
33          binding.rvStudent.adapter = adapterStudent
34
35          viewModel.student.observe( owner: this) { student ->
36              originalList = student
37              adapterStudent.submitList(student)
38          }
39
40          binding.fabAdd.setOnClickListener {
41              val intent = Intent( packageContext: this, DetailActivity::class.java)
42              startActivity(intent)
43          }
44
45          binding.searchView.setOnQueryTextListener(object : SearchView.OnQueryTextListener {
46              override fun onQueryTextSubmit(query: String?): Boolean {
47                  return false
48              }
49
50              override fun onQueryTextChange(newText: String?): Boolean {
51                  filterStudent(newText)
52                  return true
53              }
54          })
55      }
56
57      private fun filterStudent(query: String?) {
58          val filteredList = if (!query.isNullOrEmpty()) {
59              originalList.filter { it.name.contains(query, ignoreCase: true) }
60          } else {
61              originalList
62          }
63          adapterStudent.submitList(filteredList)
64      }
65  }
```

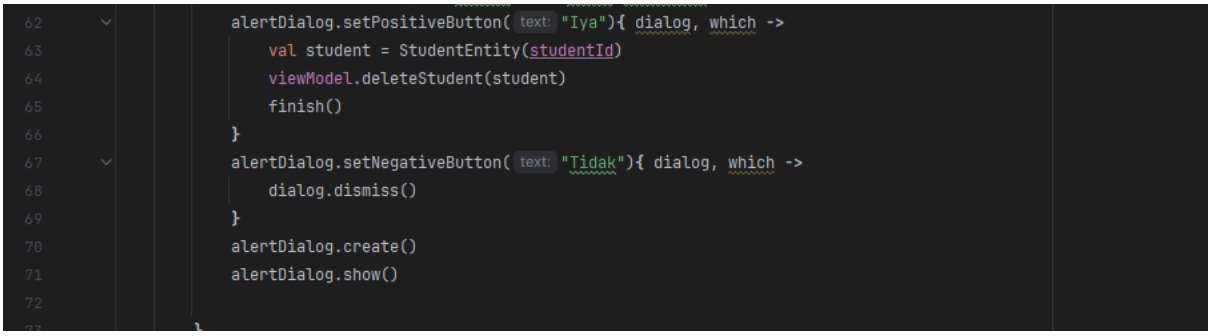
Class ini merupakan tampilan utama aplikasi yang bertanggung jawab untuk menampilkan seluruh data siswa yang tersimpan dalam database. Di dalamnya terdapat fitur pencarian untuk memfilter

daftar siswa berdasarkan nama secara real-time. Selain itu, ada juga tombol untuk menambahkan data baru yang akan membuka halaman detail. Data siswa yang ditampilkan diambil dari ViewModel dan ditampilkan melalui RecyclerView yang terhubung dengan adapter.

- **DetailActivity.kt**



```
class DetailActivity : AppCompatActivity() {

    private lateinit var binding : ActivityDetailBinding
    private var studentId : Int = 0
    private val viewModel : StudentViewModel by viewModels()
    private lateinit var nama : EditText

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding = ActivityDetailBinding.inflate(layoutInflater)
        setContentView(binding.root)


        val data = intent.getParcelableExtra<StudentEntity>( name: "EXTRA_DATA")
        if (data != null) {
            nama = binding.edtNama
            nama.setText(data.name)
            binding.edtNis.setText(data.nis)
            binding.btnSave.text = "Update"
            studentId = data.id
            binding.btnDelete.visibility = View.VISIBLE
        }

        binding.btnSave.setOnClickListener{
            if (binding.btnSave.text == "Update"){
                val student = StudentEntity(
                    studentId,
                    binding.edtNama.text.toString(),
                    binding.edtNis.text.toString()
                )

                viewModel.updateStudent(student)
            } else {
                val student = StudentEntity(
                    studentId,
                    binding.edtNama.text.toString(),
                    binding.edtNis.text.toString()
                )
                viewModel.insertStudent(student)
            }
            finish()

        }

        binding.btnDelete.setOnClickListener {
            val alertDialog = AlertDialog.Builder( context: this)
            alertDialog.setTitle("Peringatan")
            alertDialog.setMessage("Apakah anda ingin menghapus $nama")
            alertDialog.setPositiveButton( text: "Iya"){ dialog, which ->
```



```
        alertDialog.setPositiveButton( text: "Iya"){ dialog, which ->
            val student = StudentEntity(studentId)
            viewModel.deleteStudent(student)
            finish()
        }
        alertDialog.setNegativeButton( text: "Tidak"){ dialog, which ->
            dialog.dismiss()
        }
        alertDialog.create()
        alertDialog.show()

    }
```

Class ini adalah tampilan yang digunakan untuk menambahkan data siswa baru atau mengedit data yang sudah ada. Jika activity ini dibuka dengan membawa data siswa, maka tombol akan berubah menjadi tombol update, dan data siswa tersebut akan dimunculkan ke dalam input. Jika tidak, maka akan dianggap sebagai input baru. Selain itu, class ini juga menyediakan tombol untuk menghapus data siswa yang sedang ditampilkan.