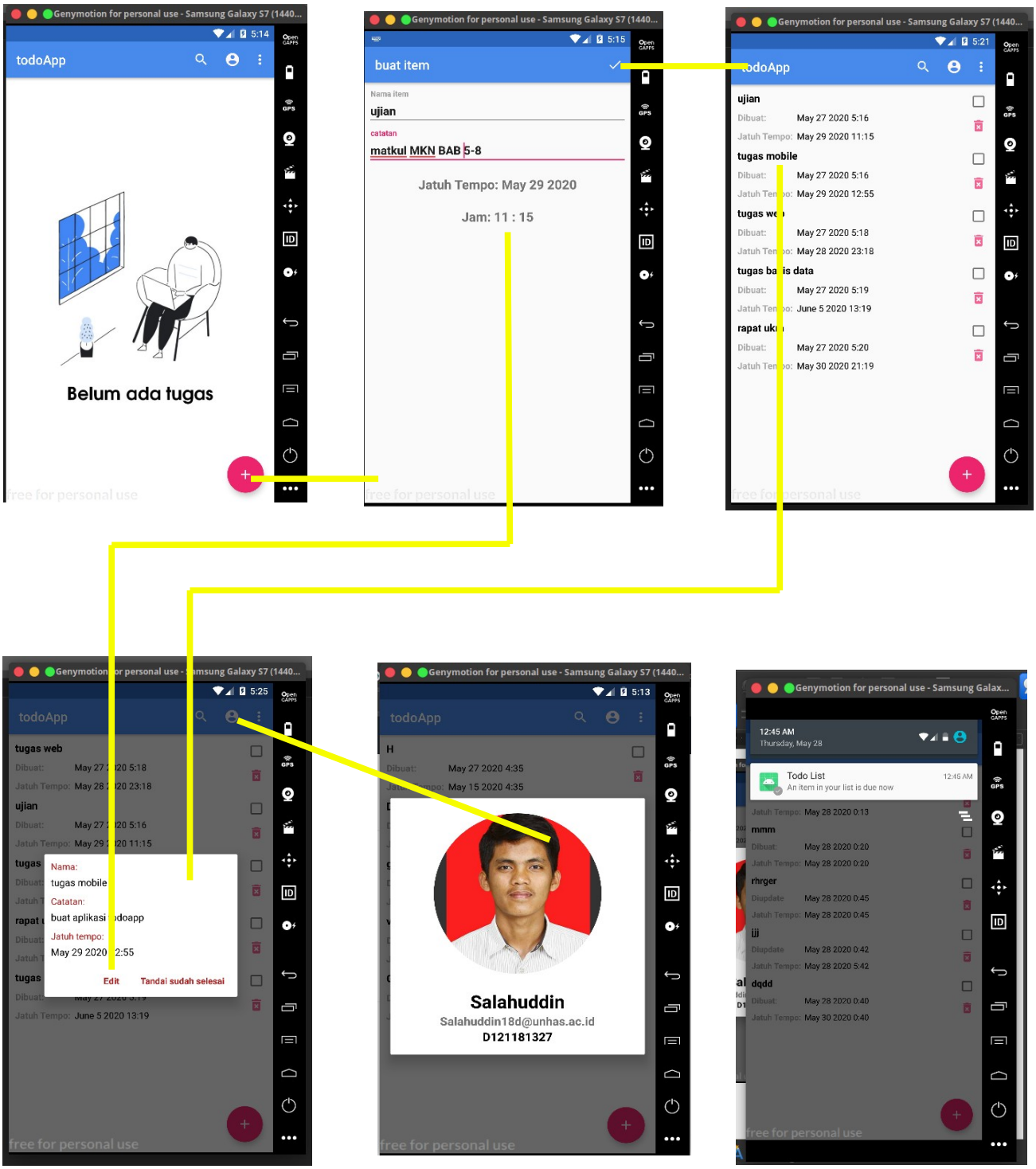
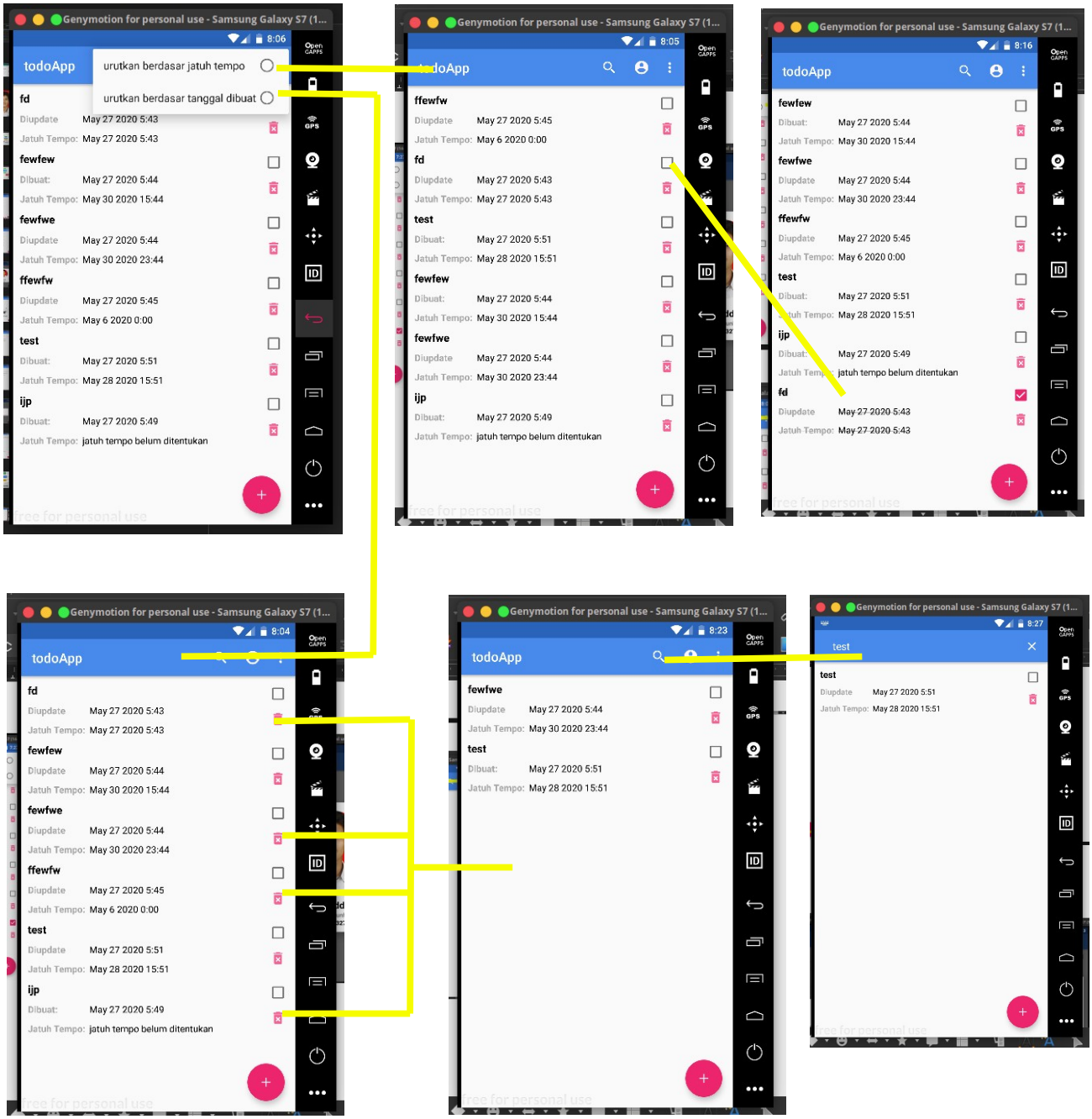
Nama : Salahuddin
NIM : D121181327

Dokumentasi aplikasi TodoApp dengan kotlin, room, mvvm dan livedata

Alur Aplikasi

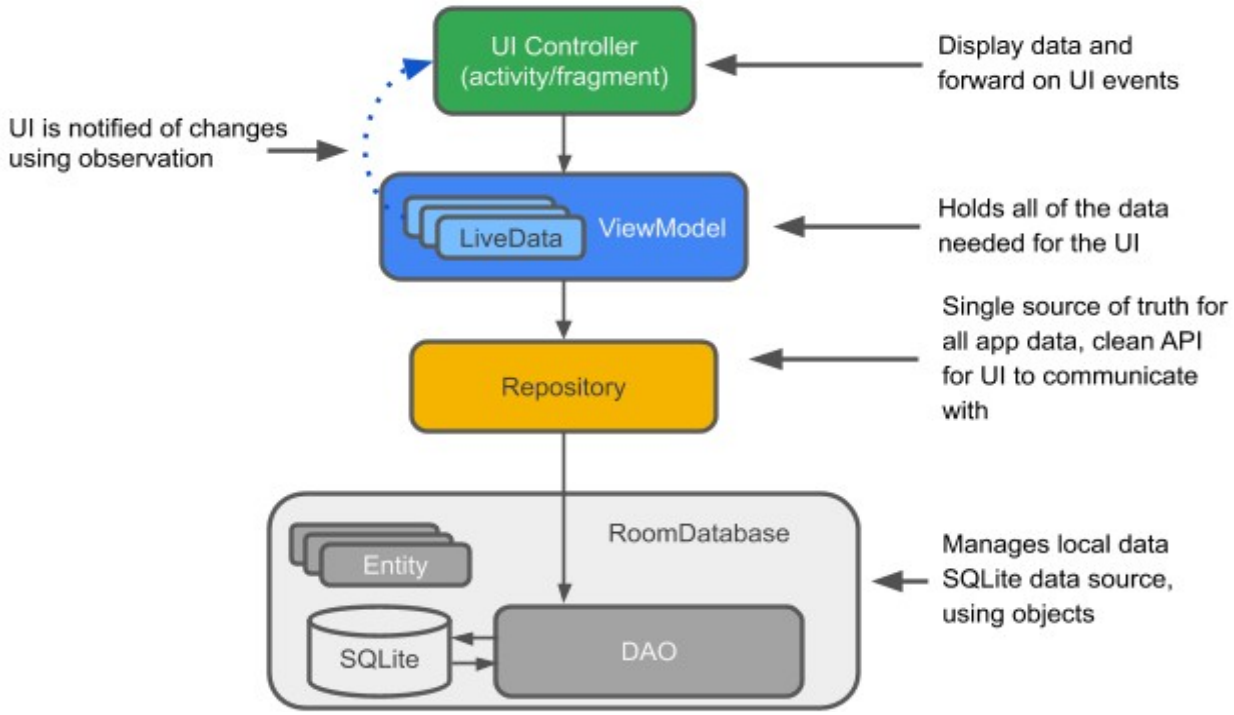


Nama : Salahuddin
NIM : D121181327



Nama : Salahuddin
NIM : D121181327

Arsitektur Aplikasi



Entity: Dalam konteks Komponen Arsitektur, entity adalah kelas beranotasi yang menjelaskan tabel database.

SQLite: Di perangkat, data disimpan dalam basis data SQLite. Room persistence library membuat dan mengelola database tersebut.

DAO: singakatan “data access object” atau objek akses data. Pemetaan query SQL ke fungsi. Anda dulu harus mendefinisikan kueri ini di kelas pembantu. Saat Anda menggunakan DAO, kode Anda memanggil fungsi-fungsi, dan komponen-komponennya mengatur sisanya.

Repositori: Kelas untuk mengelola beberapa sumber data. Selain dari Room persistence library, Repositori juga dapat mengelola sumber data jarak jauh seperti server web.

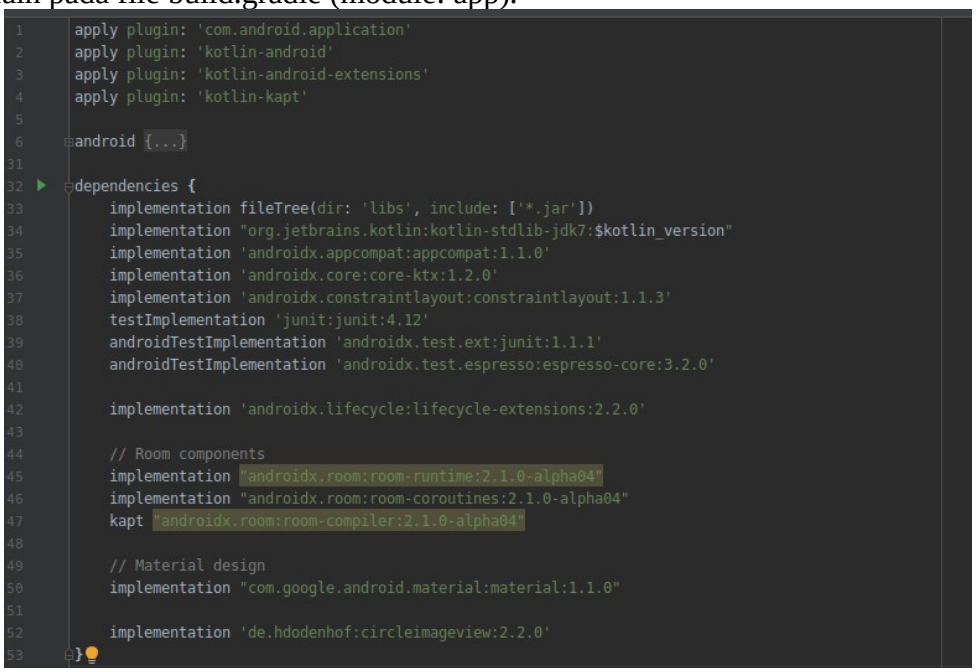
ViewModel: Menyediakan data ke UI dan bertindak sebagai pusat komunikasi antara Repositori dan UI. Menyembunyikan backend dari UI. ViewModel akan bertahan dari perubahan konfigurasi perangkat.

LiveData: Kelas holder data yang mengamati perubahan yang terjadi pada data. Selalu memegang / menyimpan data versi terbaru. Memberitahu pengamatnya ketika data telah berubah. Secara umum, komponen UI mengamati data yang relevan. LiveData sadar akan siklus hidup, sehingga secara otomatis mengelola penghentian dan melanjutkan pengamatan berdasarkan status aktivitas pengamatan atau fragmennya.

Nama : Salahuddin
NIM : D121181327

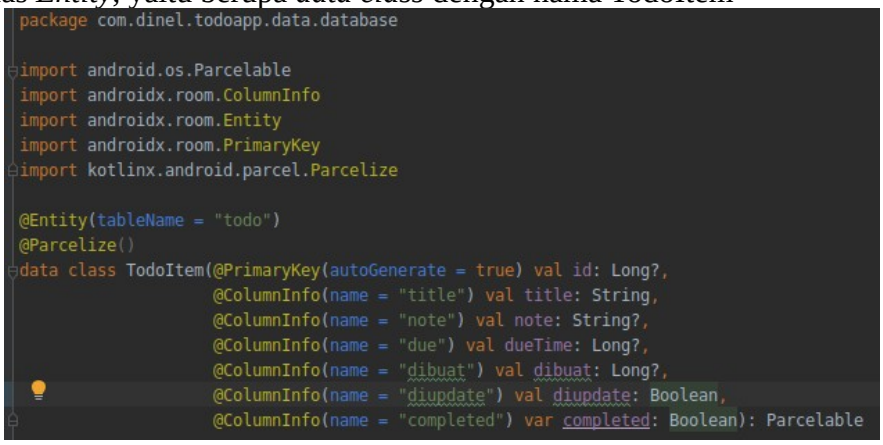
Baris Kode

1. Menambahkan *plugin* kapt Kotlin dan *dependency* yang dibutuhkan seperti room, material design dan library lain pada file build.gradle (module: app).



```
apply plugin: 'com.android.application'
apply plugin: 'kotlin-android'
apply plugin: 'kotlin-android-extensions'
apply plugin: 'kotlin-kapt'

android {...}

dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation "org.jetbrains.kotlin:kotlin-stdlib-jdk7:$kotlin_version"
    implementation 'androidx.appcompat:appcompat:1.1.0'
    implementation 'androidx.core:core-ktx:1.2.0'
    implementation 'androidx.constraintlayout:constraintlayout:1.1.3'
    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'androidx.test.ext:junit:1.1.1'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.2.0'

    implementation 'androidx.lifecycle:lifecycle-extensions:2.2.0'

    // Room components
    implementation "androidx.room:room-runtime:2.1.0-alpha04"
    implementation "androidx.room:room-coroutines:2.1.0-alpha04"
    kapt "androidx.room:room-compiler:2.1.0-alpha04"

    // Material design
    implementation "com.google.android.material:material:1.1.0"

    implementation 'de.hdodenhof:circleimageview:2.2.0'
}
```

2. Membuat kelas *Entity*, yaitu berupa *data class* dengan nama TodoItem



```
package com.dinel.todoapp.data.database

import android.os.Parcelable
import androidx.room.ColumnInfo
import androidx.room.Entity
import androidx.room.PrimaryKey
import kotlinx.android.parcel.Parcelize

@Entity(tableName = "todo")
@Parcelize()
data class TodoItem(@PrimaryKey(autoGenerate = true) val id: Long?,
                    @ColumnInfo(name = "title") val title: String,
                    @ColumnInfo(name = "note") val note: String?,
                    @ColumnInfo(name = "due") val dueTime: Long?,
                    @ColumnInfo(name = "dibuat") val dibuat: Long?,
                    @ColumnInfo(name = "diupdate") val diupdate: Boolean,
                    @ColumnInfo(name = "completed") var completed: Boolean): Parcelable
```

@Entity mengindikasikan bahwa kelas tersebut merupakan entity, didalamnya kita juga dapat memberi nama tabel.

@ColumnInfo untuk memberikan nama kolom.

@PrimaryKey mengindikasikan bahwa kolom tersebut adalah *primary key.*

3. Membuat kelas *Database*-nya dengan nama TodoItemDatabase.
annotation @Database mengindikasikan bahwa kelas tersebut merupakan kelas *Room Database*, serta mendefinisikan *entity,* versi database. Disini exportSchema bernilai *false* karena pada projek ini kita tidak akan melakukan mekanisme *export schema* jika ada pembaharuan versi *database*.

Nama : Salahuddin
NIM : D121181327

```
package com.dinel.todoapp.data.database

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

@Database(entities = [TodoItem::class], version = 1, exportSchema = false)
abstract class TodoItemDatabase : RoomDatabase() {

    abstract fun todoDao(): TodoItemDao

    companion object {
        private var INSTANCE: TodoItemDatabase? = null

        fun getInstance(context: Context): TodoItemDatabase? {
            if (INSTANCE == null) {
                synchronized(TodoItemDatabase::class) {
                    INSTANCE = Room.databaseBuilder(context,
                        TodoItemDatabase::class.java,
                        name: "todo_db")
                        .build()
                }
            }

            return INSTANCE
        }
    }

}
```

4. Membuat kelas *Dao* yaitu berupa *interface* dengan nama TodoItemDao.

```
package com.dinel.todoapp.data.database
import ...
@Dao
interface TodoItemDao {
    @Insert
    suspend fun saveTodoItem(todoItem: TodoItem)
    @Insert
    suspend fun saveTodoItems(todoItems: List<TodoItem>)
    @Delete
    suspend fun deleteTodoItem(todoItem: TodoItem)
    @Update
    suspend fun updateTodoItem(todoItem: TodoItem)
    @Query( value: "SELECT * FROM todo ORDER BY id DESC")
    fun getAllTodoList(): LiveData<MutableList<TodoItem>>

}
```

pada *interface* ini *Room* juga menggunakan *annotation* untuk mengindikasikan jenis kelasnya, yaitu dengan @Dao. Untuk fungsi *insert*, *update*, dan *delete* telah disediakan *annotation* @insert, @update, dan @delete, sedangkan untuk fungsi *read*, kita dapat menggunakan @Query dengan disertakan *query* SQLite-nya pada parameter.

5. Membuat kelas repositori dengan nama TodoItemRepository

```
class TodoItemRepository(application: Application) {

    private val todoItemDao: TodoItemDao
    private val allTodoItems: LiveData<MutableList<TodoItem>>

    init {...}

    fun saveTodoItems(todoItems: List<TodoItem>) = runBlocking { this: CoroutineScope
        this.launch(Dispatchers.IO) {...}
    }

    fun saveTodoItem(todoItem: TodoItem) = runBlocking { this: CoroutineScope
        this.launch(Dispatchers.IO) {...}
    }

    fun updateTodoItem(todoItem: TodoItem) = runBlocking { this: CoroutineScope
        this.launch(Dispatchers.IO) {...}
    }

    fun deleteTodoItem(todoItem: TodoItem) = runBlocking { this: CoroutineScope
        this.launch(Dispatchers.IO) {...}
    }

    fun getAllTodoList(): LiveData<MutableList<TodoItem>> {...}

}
```

Kelas repositori mengabstraksi akses ke berbagai sumber data. Repositori bukan bagian dari pustaka Komponen Arsitektur, tetapi merupakan praktik terbaik yang disarankan untuk pemisahan kode dan arsitektur. Kelas Repositori menyediakan API bersih untuk akses data ke seluruh aplikasi.

Nama : Salahuddin
NIM : D121181327

6. Membuat Kelas ViewModel dengan nama TodoItemViewModel

```
package com.dinel.todoapp.viewmodel

import ...

class TodoViewModel(application: Application) : AndroidViewModel(application) {

    private val repository: TodoItemRepository = TodoItemRepository(application)
    private val todoItems: LiveData<MutableList<TodoItem>> = repository.getAllTodoList()

    fun saveTodoItem(todoItem: TodoItem) {...}

    fun saveTodoItems(todoItems: List<TodoItem>) {...}

    fun updateTodoItem(todoItem:TodoItem) {...}

    fun deleteTodoItem(todoItem: TodoItem) {...}

    fun toggleCompleteState(todoItem: TodoItem) {...}

    fun getAllTodoItemList(): LiveData<MutableList<TodoItem>> {...}
}
```

ViewModel adalah kelas yang perannya menyediakan data ke UI dan manjaga dari perubahan konfigurasi. ViewModel bertindak sebagai pusat komunikasi antara Repositori dan UI. ViewModel adalah bagian dari perpustakaan siklus hidup. Untuk panduan pengantar topik ini, lihat ViewModel.

7. Menambahkan XMLLayout dan Resource

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

    <include layout="@layout/todo_list"/>

    <ImageView
        android:id="@+id/iv_empty_task_list"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:src="@mipmap/empty_task_list_two"
        android:scaleType="centerCrop"
        android:layout_gravity="center"
        android:contentDescription="image"
        android:visibility="gone"/>

    <com.google.android.material.floatingactionbutton.FloatingActionButton
        android:id="@+id/fab_add_item"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="end|bottom"
        android:src="@drawable/ic_add_black_24dp"
        android:layout_margin="16dp" />
```

file activity_main.xml menampilkan recycleview gambar jika data masih kosong dan floating bottm

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

    <TextView...>

    <TextView...>

    <TextView...>


    <TextView...>

    <TextView...>

    <CheckBox...>


    <ImageView...>

</RelativeLayout>
```

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/rv_todo_list"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layoutAnimation="@anim/layout_animation"
        tools:listitem="@layout/item_todo_list">
    </androidx.recyclerview.widget.RecyclerView>

</LinearLayout>
```

file item_todo_list dan todo_list sebagai ui recycleview

8. Membuat Adapter dan Menambahkan RecyclerView

```
class TodoListAdapter(todoItemClickListener: TodoItemClickListener) :
    RecyclerView.Adapter<TodoListAdapter.ViewHolder>(), Filterable {

    private var todoItemList: List<TodoItem> = arrayListOf()
    private var filteredTodoItemList: List<TodoItem> = arrayListOf()
    private val listener: TodoItemClickListener = todoItemClickListener

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

    override fun getItemCount(): Int = filteredTodoItemList.size

    override fun onBindViewHolder(holder: ViewHolder, position: Int) {
        holder.bindItem(filteredTodoItemList[position], listener)
    }

    override fun getFilter(): Filter {
        return object : Filter() {
            override fun performFiltering(constraint: CharSequence?): FilterResults {
                val charString = constraint.toString()

                filteredTodoItemList = if (charString.isEmpty()) {
                    todoItemList
                } else {
                    val filteredList = arrayListOf<TodoItem>()
                    for (item in todoItemList) {
```

IDE and Plugin Updates

Nama : Salahuddin
NIM : D121181327

9. Mengupdate kelas MainActivity

```
})
fab_add_item.setOnClickListener { it: View!
    clearSearchView()
    val intent = Intent( packageContext: this@MainActivity, AddEditTodoItemActivity::class.java)
    startActivityForResult(intent, Constants.INTENT_CREATE_TODO_ITEM)
}
```

menambhakan even listener untuk floating action butttom pada fungsi oncreate

```
rv_todo_list.layoutManager = LinearLayoutManager( context: this)
todoAdapter = TodoListAdapter( todoItemClickListener: this)
rv_todo_list.adapter = todoAdapter
```

memasang adapter recycleviw

```
todoViewModel = ViewModelProviders.of( activity: this).get(TodoViewModel::class.java)
todoViewModel.getAllTodoItemList().observe( owner: this, Observer { it ->

    val itemsWithNoDeadline = mutableListOf<TodoItem>()
    val completedItems = mutableListOf<TodoItem>()

    for (item in it) {
        if (item.dueTime!!.toInt() == 0 && !item.completed) {
            itemsWithNoDeadline.add(item)
        } else if (item.completed) {
            completedItems.add(item)
        }
    }
```

Mengunakan viewModel untuk mengamati perubahan data

```
63          if (!Constants.urut) {
64              it.sortBy { it.dueTime }
65          } else {
66              it.sortBy { it.dibuat }
67          }
```

fungsi mengurutkan berdasar jatuh tempo atau waktu dibuat

```
83          if (it.size == 0) {
84              displayEmptyTaskListImage()
85          }
86      })
```

Menapilkan gambar empty task jika data kosong

```
165     private fun displayTodoItemCountDialog() {
166         countDialog = Dialog( context: this)
167         countDialog!!.requestWindowFeature(Window.FEATURE_NO_TITLE)
168         countDialog!!.setCancelable(true)
169         countDialog!!.setContentView(R.layout.activity_about)
170
171
172         countDialog!!.show()
173     }
174
```

Fungsi untuk menampilakan dialog about me

```
183     override fun onDeleteClicked(todoItem: TodoItem) {
184         todoViewModel.deleteTodoItem(todoItem)
185
186         NotificationUtils().cancelNotification(todoItem, activity: this)
187     }
188
```

Fungsi untuk menghandle ketika ikon delete diklik yang akan menhapus data dan membatalkan notifikasi

```
override fun onCheckClicked(todoItem: TodoItem) {
    if (!todoItem.completed) {
        NotificationUtils().cancelNotification(todoItem, activity: this)
    } else if (todoItem.completed && todoItem.dueTime!! > 0 && System.currentTimeMillis() < todoItem.dueTime) {
        NotificationUtils().setNotification(todoItem, activity: this)
    }

    todoViewModel.toggleCompleteState(todoItem)
}
```

Fungsi untuk menghandle ketika checkbox diklik

```
189     override fun onItemClicked(todoItem: TodoItem) {
190         clearSearchView()
191
192         // display the details of the item in a dialog.
193         displayEventDetails(todoItem)
194
195     }
```

Fungsi untuk menghandle ketika item di klik maka akan memanggil fungsi displayeventDetails dengan parameter todoitem

```
212     private fun displayEventDetails(todoItem: TodoItem) {
213         dialog = Dialog( context: this)
214         dialog!!.requestWindowFeature(Window.FEATURE_NO_TITLE)
215         dialog!!.setCancelable(true)
216         dialog!!.setContentView(R.layout.todo_item_display_details_dialog)
217
218         dialog!!.tv_todo_title_content.text = todoItem.title
219         dialog!!.tv_todo_description_content.text = todoItem.note
220
```

fungsi displayeventDetails yang menampilkan dialog berisi data dari item yang diklik

```
265         dialog!!.button_edit_todo_item.setOnClickListener { it: View!
266             NotificationUtils().cancelNotification(todoItem, activity: this)
267             val intent = Intent( packageContext: this@MainActivity, AddEditTodoItemActivity::class.java)
268             intent.putExtra(Constants.KEY_INTENT, todoItem)
269             startActivityForResult(intent, Constants.INTENT_EDIT_TODO_ITEM)
270             dialog!!.dismiss()
271         }
```

fungsi untuk menghandle ketika tombol edit diklik pada dialog displayeventDetails yang akan akan membuat intent untuk pindah activity dan mengirim data item untuk diedit

Nama : Salahuddin
NIM : D121181327

10.Buat class activity untuk menambahkan item dengan nama AddEditTodoItemAcivity

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_add_edit_todo_item)

    val intent = intent
    if (intent != null && intent.hasExtra(Constants.KEY_INTENT)) {
        val todoItem: TodoItem = intent.getParcelableExtra(Constants.KEY_INTENT)
        this.todoItem = todoItem
        diupdated = true

        if (todoItem.dueTime!!.toInt() != 0) {
            dateSelected = true
            timeSelected = true
            val list = convertMillis(todoItem.dueTime)

            mDueDay = list[0]
            mDueMonth = list[1]
            mDueYear = list[2]
            mDueHour = list[3]
            mDueMinute = list[4]
        }

        fillUIWithItemData(todoItem)
    }

    tv_todo_due_date.setOnClickListener { it: View!
        showDatePickerDialog()
    }
```

mengecek jika intent mengirim data key intent maka todoitem berisi data todo item yang akan di edit. Jika tidak maka todoitem masih bernilai null

```
private  fun getMilliFromDate(dateFormat: String?): Long {
    var date = Date()
    val formatter = SimpleDateFormat( pattern: "dd/MM/yyyy HH/mm")
    try {
        date = formatter.parse(dateFormat)
    } catch (e: ParseException) {
        e.printStackTrace()
    }
    return date.time
}
```

fungsi untuk mendapatkan waktu dibuat

```
private fun showDatePickerDialog() {
    mDueDay = Calendar.getInstance().get(Calendar.DAY_OF_MONTH)
    mDueMonth = Calendar.getInstance().get(Calendar.MONTH)
    mDueYear = Calendar.getInstance().get(Calendar.YEAR)

    val datePickerDialog = DatePickerDialog(
        context: this,
        DatePickerDialog.OnDateSetListener { _, year, monthOfYear, dayOfMonth ->

            tv_todo_due_date.text =
                ("""Jatuh Tempo: ${convertNumberToMonthName(monthOfYear)} $dayOfMonth $year""")

            mDueDay = dayOfMonth
            mDueMonth = monthOfYear
            mDueYear = year
            dateSelected = true
        },
        mDueYear,
        mDueMonth,
        mDueDay
    )
    datePickerDialog.show()
}
```

Fungsi untuk menmpilakn dilalog date picker untuk input waktu jatuh tempo

```
@RequiresApi(Build.VERSION_CODES.O)
private fun saveTodoItem() {
    if (validateFields()) {
        val id = if (todoItem != null) todoItem?.id else null
        val todo = TodoItem(
            id = id,
            title = et_todo_title.text.toString(),
            note = et_todo_description.text.toString(),
            dueTime = dueDate,
            dibuat = getMilliFromDate( dateFormat: "yyyy MM dd HH.mm"),
            completed = todoItem?.completed ?: false,
            diupdate = diupdated
        )
```

fungsi untuk menyimpan data yang telah divalidasi

```
private fun validateFields(): Boolean {
    if (et_todo_title.text.isEmpty()) {
        til_todo_title.error = "Masukkan nama item"
        et_todo_title.requestFocus()
        return false
    }
    if (et_todo_description.text.isEmpty()) {
        til_todo_description.error = "Masukkan catatan item"
        et_todo_description.requestFocus()
        return false
    }
    Toast.makeText( context: this,  text: "Item berhasil disimpan.", Toast.LENGTH_SHORT).show()
    return true
}
```

Fungsi untuk memvalidasi inputan user

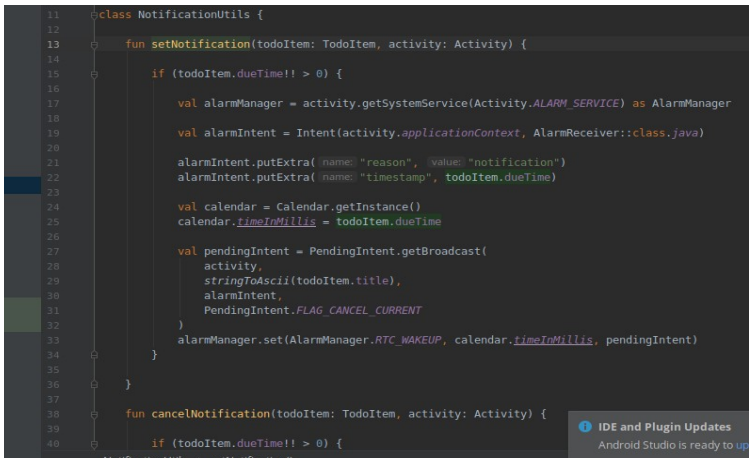
11. Membuat notifikasi

update file manifest. Tambahkan permision untuk foregroud service

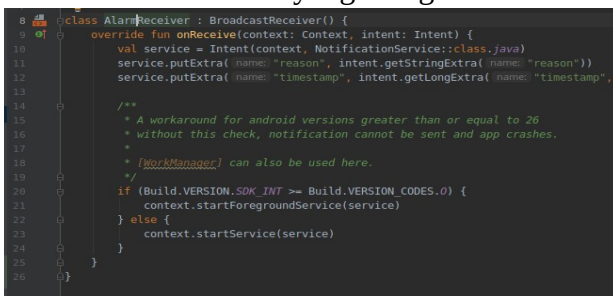
```
<uses-permission android:name="android.permission.FOREGROUND_SERVICE" />

<application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="todoApp"
    android:roundIcon="@mipmap/ic_launcher_round"
    android:supportsRtl="true"
    android:theme="@style/AppTheme">
    <activity android:name=".AddEditTodoItemActivity" />
    <activity android:name=".MainActivity">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />

            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>

    <receiver
        android:name=".notification.AlarmReceiver"
        android:enabled="true" />

    <service
        android:name=".notification.NotificationService"
        android:enabled="true" />
</application>
```

Nama : Salahuddin
NIM : D121181327

Buat kelas notifikasiUtils yang mempyunyai fungsi set notifikasi dan cencel notifikasi



Buat kelas alarmRecaiver yang mengekstend kelas brodcastReacaiver sebagai recaiver notivikasi



Buat kelas NotivikasiService yang mengekstend kelas IntentService sebagai servicer notivikasi