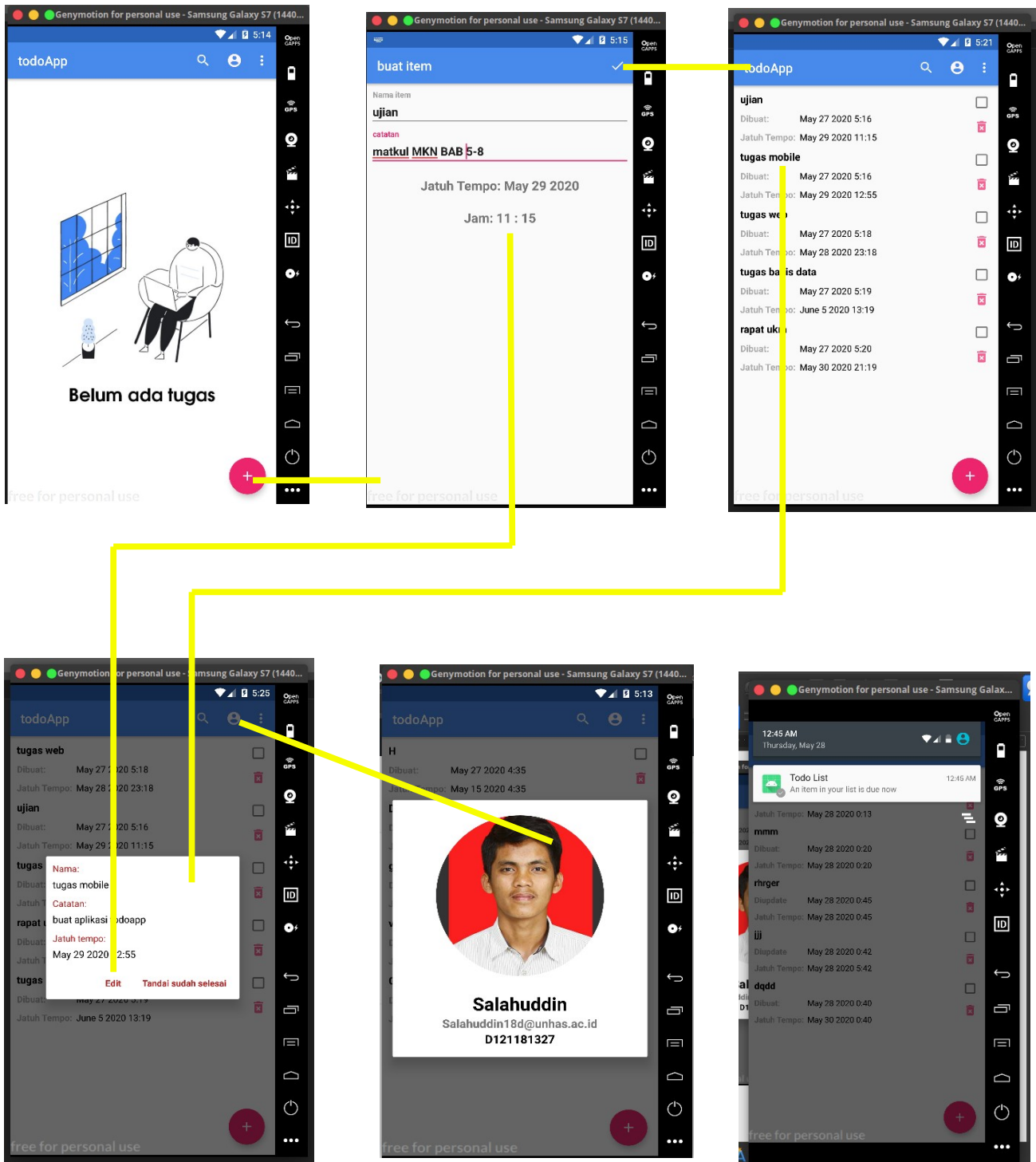


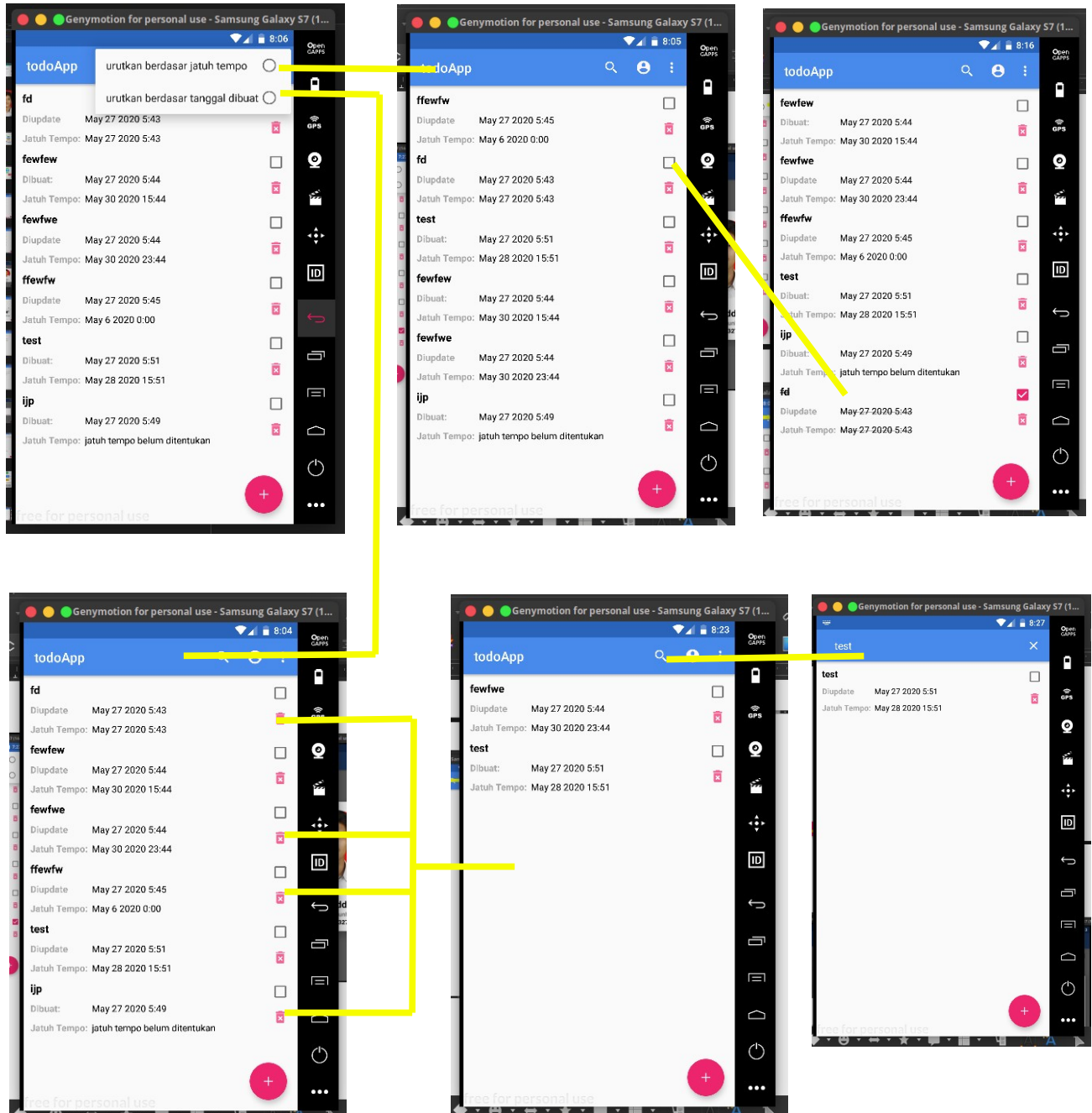
Nama : Salahuddin
NIM : D121181327

Dokumentasi aplikasi TodoApp dengan kotlin, room, mvvm dan livedata

Alur Aplikasi

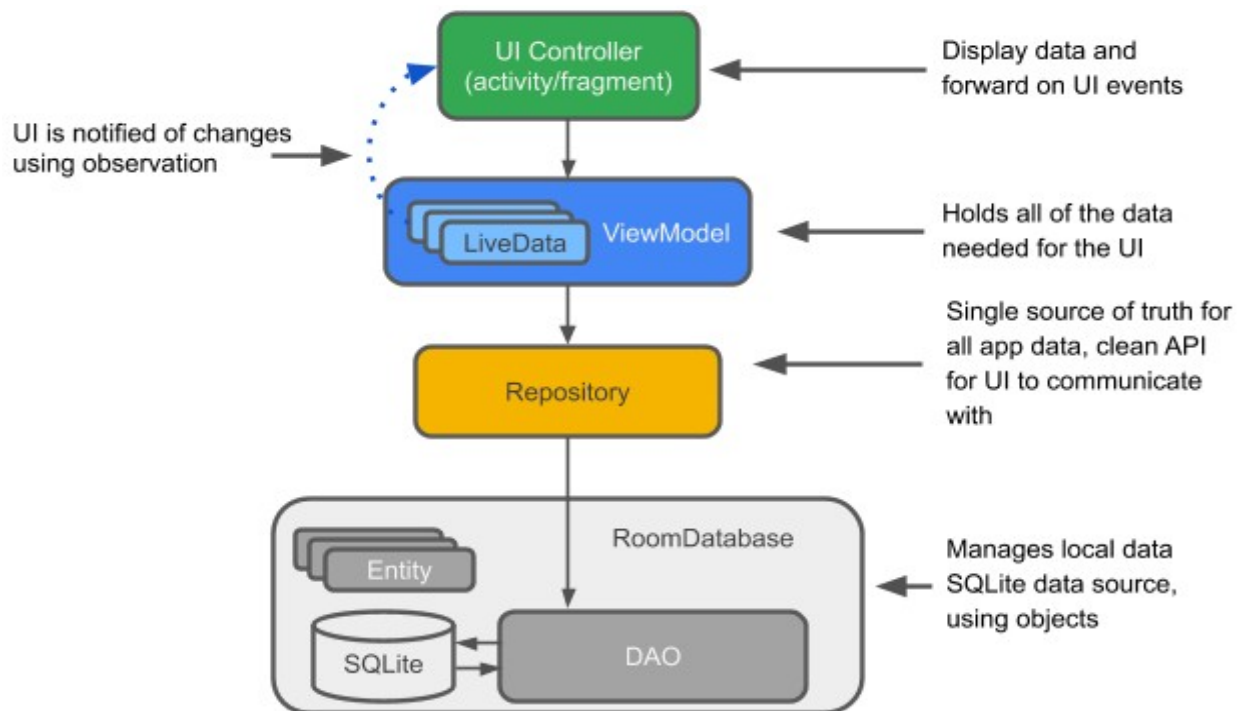


Nama : Salahuddin
NIM : D121181327



Nama : Salahuddin
NIM : D121181327

Arsitektur Aplikasi



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

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

DAO: singkatan "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.

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

ViewModel: Menyediakan data ke UI dan bertindak sebagai pusat komunikasi antara Repository 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).

```
1  apply plugin: 'com.android.application'
2  apply plugin: 'kotlin-android'
3  apply plugin: 'kotlin-android-extensions'
4  apply plugin: 'kotlin-kapt'
5
6  android { ... }
7
8  dependencies {
9      implementation fileTree(dir: 'libs', include: ['*.jar'])
10     implementation "org.jetbrains.kotlin:kotlin-stdlib-jdk7:$kotlin_version"
11     implementation 'androidx.appcompat:appcompat:1.1.0'
12     implementation 'androidx.core:core-ktx:1.2.0'
13     implementation 'androidx.constraintlayout:constraintlayout:1.1.3'
14     testImplementation 'junit:junit:4.12'
15     androidTestImplementation 'androidx.test.ext:junit:1.1.1'
16     androidTestImplementation 'androidx.test.espresso:espresso-core:3.2.0'
17
18     implementation 'androidx.lifecycle:lifecycle-extensions:2.2.0'
19
20     // Room components
21     implementation "androidx.room:room-runtime:2.1.0-alpha04"
22     implementation "androidx.room:room-coroutines:2.1.0-alpha04"
23     kapt "androidx.room:room-compiler:2.1.0-alpha04"
24
25     // Material design
26     implementation "com.google.android.material:material:1.1.0"
27
28     implementation 'de.hdodenhof:circleimageview:2.2.0'
29 }
```

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 proyek ini kita tidak akan melakukan mekanisme *export schema* jika ada pembaharuan versi *database*.

Nama : Salahuddin
NIM : D121181327

```
1 package com.dinel.todoapp.data.database
2
3 import android.content.Context
4 import androidx.room.Database
5 import androidx.room.Room
6 import androidx.room.RoomDatabase
7
8 @Database(entities = [TodoItem::class], version = 1, exportSchema = false)
9 abstract class TodoItemDatabase : RoomDatabase() {
10
11     abstract fun todoDao(): TodoItemDao
12
13     companion object {
14         private var INSTANCE: TodoItemDatabase? = null
15
16         fun getInstance(context: Context): TodoItemDatabase? {
17             if (INSTANCE == null) {
18                 synchronized(TodoItemDatabase::class) {
19                     INSTANCE = Room.databaseBuilder(context,
20                         TodoItemDatabase::class.java,
21                         "todo_db")
22                         .build()
23                 }
24             }
25             return INSTANCE
26         }
27     }
28 }
29
30
```

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

```
1 package com.dinel.todoapp.data.database
2 import ...
3
4 @Dao
5 interface TodoItemDao {
6     @Insert
7     suspend fun saveTodoItem(todoItem: TodoItem)
8     @Insert
9     suspend fun saveTodoItems(todoItems: List<TodoItem>)
10    @Delete
11    suspend fun deleteTodoItem(todoItem: TodoItem)
12    @Update
13    suspend fun updateTodoItem(todoItem: TodoItem)
14    @Query("SELECT * FROM todo ORDER BY id DESC")
15    fun getAllTodoList(): LiveData<MutableList<TodoItem>>
16
17 }
```

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*

```
12 class TodoItemRepository(application: Application) {
13
14     private val todoItemDao: TodoItemDao
15     private val allTodoItems: LiveData<MutableList<TodoItem>>
16
17     init { ... }
18
19     fun saveTodoItems(todoItems: List<TodoItem>) = runBlocking { this: CoroutineScope
20         this.launch(Dispatchers.IO) { ... }
21     }
22
23     fun saveTodoItem(todoItem: TodoItem) = runBlocking { this: CoroutineScope
24         this.launch(Dispatchers.IO) { ... }
25     }
26
27     fun updateTodoItem(todoItem: TodoItem) = runBlocking { this: CoroutineScope
28         this.launch(Dispatchers.IO) { ... }
29     }
30
31     fun deleteTodoItem(todoItem: TodoItem) = runBlocking { this: CoroutineScope
32         this.launch(Dispatchers.IO) { ... }
33     }
34
35     fun getAllTodoList(): LiveData<MutableList<TodoItem>> { ... }
36
37 }
```

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

```
1 package com.dinel.todoapp.viewmodel
2
3 import ...
4
5 class TodoViewModel(application: Application) : AndroidViewModel(application) {
6
7     private val repository: TodoItemRepository = TodoItemRepository(application)
8     private val todoItems: LiveData<MutableList<TodoItem>> = repository.getAllTodoList()
9
10     fun saveTodoItem(todoItem: TodoItem) {...}
11
12     fun saveTodoItems(todoItems: List<TodoItem>) {...}
13
14     fun updateTodoItem(todoItem: TodoItem) {...}
15
16     fun deleteTodoItem(todoItem: TodoItem) {...}
17
18     fun toggleCompleteState(todoItem: TodoItem) {...}
19
20     fun getAllTodoItemList(): LiveData<MutableList<TodoItem>> {...}
21 }
```

ViewModel adalah kelas yang perannya menyediakan data ke UI dan menjaga 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

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <androidx.coordinatorlayout.widget.CoordinatorLayout xmlns:android="http://schemas.android.com/apk/res/android"
3     xmlns:tools="http://schemas.android.com/tools"
4     android:layout_width="match_parent"
5     android:layout_height="match_parent"
6     xmlns:app="http://schemas.android.com/apk/res-auto"
7     android:orientation="vertical"
8     tools:context="MainActivity">
9
10     <include layout="@layout/todo_list"/>
11
12     <ImageView
13         android:id="@+id/iv_empty_task_list"
14         android:layout_width="match_parent"
15         android:layout_height="wrap_content"
16         android:src="@mipmap/empty_task_list_two"
17         android:scaleType="centerCrop"
18         android:layout_gravity="center"
19         android:contentDescription="image"
20         android:visibility="gone"/>
21
22     <com.google.android.material.floatingactionbutton.FloatingActionButton
23         android:id="@+id/fab_add_item"
24         android:layout_width="wrap_content"
25         android:layout_height="wrap_content"
26         android:layout_gravity="end|bottom"
27         android:src="@drawable/ic_add_black_24dp"
28         android:layout_margin="16dp"/>
29 </androidx.coordinatorlayout.widget.CoordinatorLayout>
```

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

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
3     android:layout_width="match_parent"
4     android:layout_height="wrap_content">
5
6     <TextView...>
7
8     <TextView...>
9
10    <TextView...>
11
12    <TextView...>
13
14    <TextView...>
15
16    <CheckBox...>
17
18    <ImageView...>
19 </RelativeLayout>
```

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3     xmlns:tools="http://schemas.android.com/tools"
4     android:orientation="vertical"
5     android:layout_width="match_parent"
6     android:layout_height="match_parent">
7
8     <androidx.recyclerview.widget.RecyclerView
9         android:id="@+id/rv_todo_list"
10        android:layout_width="match_parent"
11        android:layout_height="match_parent"
12        android:layoutAnimation="@anim/layout_animation"
13        tools:listitem="@layout/item_todo_list">
14    </androidx.recyclerview.widget.RecyclerView>
15
16 </LinearLayout>
```

file item_todo_list dan todo_list sebagai ui recycleview

8. Membuat Adapter dan Menambahkan RecyclerView

```
1 class TodoListAdapter(todoItemClickListener: TodoItemClickListener) :
2     RecyclerView.Adapter<TodoListAdapter.ViewHolder>(), Filterable {
3
4     private var todoItemList: List<TodoItem> = arrayListOf()
5     private var filteredTodoItemList: List<TodoItem> = arrayListOf()
6     private val listener: TodoItemClickListener = todoItemClickListener
7
8     override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): ViewHolder {
9         val view =
10             LayoutInflater.from(parent.context).inflate(R.layout.item_todo_list, parent, false)
11         return ViewHolder(view)
12     }
13
14     override fun getItemCount(): Int = filteredTodoItemList.size
15
16     override fun onBindViewHolder(holder: ViewHolder, position: Int) {
17         holder.bindItem(filteredTodoItemList[position], listener)
18     }
19
20     override fun getFilter(): Filter {
21         return object : Filter {
22             override fun performFiltering(constraint: CharSequence?): FilterResults {
23                 val charString = constraint.toString()
24
25                 filteredTodoItemList = if (charString.isEmpty()) {
26                     todoItemList
27                 } else {
28                     val filteredList = arrayListOf<TodoItem>()
29                     for (item in todoItemList) {
30                         if (item.title.contains(charString)) {
31                             filteredList.add(item)
32                         }
33                     }
34                     filteredList
35                 }
36                 FilterResults()
37                 results.values = filteredList
38                 return this
39             }
40         }
41     }
42 }
```

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)
    }
```

menambahkan even listener untuk floating action buttton pada fungsi onCreate

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

memasang adapter recycleview

```
todoViewModel = ViewModelProviders.of( @NonNull this).get(TodoViewModel::class.java)
todoViewModel.getAllTodoItemList().observe( @NonNull 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)
        }
    }
}
```

Menggunakan 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

```
82
83         if (it.size == 0) {
84             displayEmptyTaskListImage()
85         }
86     })
```

Menampilkan gambar empty task jika data kosong

```
165     private fun displayTodoItemCountDialog() {
166         countDialog = Dialog( @NonNull 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 menampilkan dialog about me

```
182
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 menghapus 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

```
188
189     override fun onItemClick(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
221         if (todoItem.dueTime!!.toInt() == 0) {
222             dialog!!.tv_todo_title_content.text = "No Deadline"
223         }
224     }
```

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         }
272 }
```

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 AddEditTodoItemActivity

```
45 override fun onCreate(savedInstanceState: Bundle?) {  
46     super.onCreate(savedInstanceState)  
47     setContentView(R.layout.activity_add_edit_todo_item)  
48  
49     val intent = intent  
50     if (intent != null && intent.hasExtra(Constants.KEY_INTENT)) {  
51         val todoItem: TodoItem = intent.getParcelableExtra(Constants.KEY_INTENT)  
52         this.todoItem = todoItem  
53         diupdated = true  
54  
55         if (todoItem.dueTime!!.toInt() != 0) {  
56             dateSelected = true  
57             timeSelected = true  
58             val list = convertMillis(todoItem.dueTime)  
59  
60             mDueDay = list[0]  
61             mDueMonth = list[1]  
62             mDueYear = list[2]  
63             mDueHour = list[3]  
64             mDueMinute = list[4]  
65         }  
66         fillUIWithItemData(todoItem)  
67     }  
68  
69     tv_todo_due_date.setOnClickListener { (R View)  
70         showDatePickerDialog()  
71     }  
72 }
```

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

```
101 private fun getMilliFromDate(dateFormat: String?): Long {  
102     var date = Date()  
103     val formatter = SimpleDateFormat(pattern = "dd/MM/yyyy HH:mm")  
104     try {  
105         date = formatter.parse(dateFormat)  
106     } catch (e: ParseException) {  
107         e.printStackTrace()  
108     }  
109     return date.time  
110 }  
111 }
```

fungsi untuk mendapatkan waktu dibuat

```
171 private fun showDatePickerDialog() {  
172     mDueDay = Calendar.getInstance().get(Calendar.DAY_OF_MONTH)  
173     mDueMonth = Calendar.getInstance().get(Calendar.MONTH)  
174     mDueYear = Calendar.getInstance().get(Calendar.YEAR)  
175  
176     val datePickerDialog = DatePickerDialog(  
177         context this,  
178         DatePickerDialog.OnDateSetListener { _, year, monthOfYear, dayOfMonth ->  
179             tv_todo_due_date.text =  
180                 ("Jatuh tempo: ${convertNumberToMonthName(monthOfYear)} $dayOfMonth $year")  
181             mDueDay = dayOfMonth  
182             mDueMonth = monthOfYear  
183             mDueYear = year  
184             dateSelected = true  
185         },  
186         mDueYear,  
187         mDueMonth,  
188         mDueDay  
189     )  
190     datePickerDialog.show()  
191 }  
192 }  
193 }
```

Fungsi untuk menampilkan dialog date picker untuk input waktu jatuh tempo

```
111 @RequiresApi(Build.VERSION_CODES.O)  
112 private fun saveTodoItem() {  
113     if (validateFields()) {  
114         val id = if (todoItem != null) todoItem?.id else null  
115         val todo = TodoItem(  
116             id = id,  
117             title = et_todo_title.text.toString(),  
118             note = et_todo_description.text.toString(),  
119             dueTime = dueDate,  
120             dibuat = getMilliFromDate(dateFormat = "yyyy MM dd HH:mm"),  
121             completed = todoItem?.completed ?: false,  
122             diupdate = diupdated  
123         )  
124     }  
125 }
```

fungsi untuk menyimpan data yang telah divalidasi

```
138 private fun validateFields(): Boolean {  
139     if (et_todo_title.text.isEmpty()) {  
140         til_todo_title.error = "Masukkan nama item"  
141         et_todo_title.requestFocus()  
142         return false  
143     }  
144     if (et_todo_description.text.isEmpty()) {  
145         til_todo_description.error = "Masukkan catatan item"  
146         et_todo_description.requestFocus()  
147         return false  
148     }  
149     Toast.makeText(context, this, "Item berhasil disimpan.", Toast.LENGTH_SHORT).show()  
150     return true  
151 }
```

Fungsi untuk memvalidasi inputan user

11. Membuat notifikasi

update file manifest. Tambahkan permission untuk foreground service

```
5 <uses-permission android:name="android.permission.FOREGROUND_SERVICE" />  
6  
7 <application  
8     android:allowBackup="true"  
9     android:icon="@mipmap/ic_launcher"  
10    android:label="@string/app_name"  
11    android:roundIcon="@mipmap/ic_launcher_round"  
12    android:supportRtl="true"  
13    android:theme="@style/AppTheme"  
14    <activity android:name=".AddEditTodoItemActivity" />  
15    <activity android:name=".MainActivity" />  
16        <intent-filter>  
17            <action android:name="android.intent.action.MAIN" />  
18  
19            <category android:name="android.intent.category.LAUNCHER" />  
20        </intent-filter>  
21    </activity>  
22  
23    <receiver  
24        android:name=".notification.AlarmReceiver"  
25        android:enabled="true" />  
26  
27    <service  
28        android:name=".notification.NotificationService"  
29        android:enabled="true" />  
30 </application>
```


Nama : Salahuddin
NIM : D121181327

Buat kelas notifikasiUtils yang mempunyai fungsi set notifikasi dan cencel notifikasi

```
11 class NotificationUtils {  
12  
13     fun setNotification(todoItem: TodoItem, activity: Activity) {  
14  
15         if (todoItem.dueTime!! > 0) {  
16  
17             val alarmManager = activity.getSystemService(Activity.ALARM_SERVICE) as AlarmManager  
18  
19             val alarmIntent = Intent(activity.applicationContext, AlarmReceiver::class.java)  
20  
21             alarmIntent.putExtra(name: "reason", value: "notification")  
22             alarmIntent.putExtra(name: "timestamp", value: todoItem.dueTime)  
23  
24             val calendar = Calendar.getInstance()  
25             calendar.timeInMillis = todoItem.dueTime  
26  
27             val pendingIntent = PendingIntent.getBroadcast(  
28                 activity,  
29                 stringToAscii(todoItem.title),  
30                 alarmIntent,  
31                 PendingIntent.FLAG_CANCEL_CURRENT  
32             )  
33             alarmManager.set(AlarmManager.RTC_WAKEUP, calendar.timeInMillis, pendingIntent)  
34         }  
35     }  
36  
37     fun cancelNotification(todoItem: TodoItem, activity: Activity) {  
38  
39         if (todoItem.dueTime!! > 0) {  
40
```

Buat kelas alarmRecaiver yang mengekstend kelas broadcastReacaiver sebagai recaiver notivikasi

```
8 class AlarmReceiver : BroadcastReceiver() {  
9     override fun onReceive(context: Context, intent: Intent) {  
10         val service = Intent(context, NotificationService::class.java)  
11         service.putExtra(name: "reason", intent.getStringExtra(name: "reason"))  
12         service.putExtra(name: "timestamp", intent.getLongExtra(name: "timestamp",  
13  
14         /**  
15          * A workaround for android versions greater than or equal to 26  
16          * without this check, notification cannot be sent and app crashes.  
17          *  
18          * [WorkManager] can also be used here.  
19          */  
20         if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {  
21             context.startForegroundService(service)  
22         } else {  
23             context.startService(service)  
24         }  
25     }  
26 }
```

Buat kelas NotivikasiService yang mengekstend kelas IntentService sebagai servicer notivikasi

```
17 class NotificationService : IntentService(Constants.INTENT_SERVICE_NAME) {  
18     private lateinit var mNotification: Notification  
19     private val mNotificationId = System.currentTimeMillis()  
20  
21     override fun onHandleIntent(intent: Intent?) {  
22  
23         createChannel()  
24  
25         var timeStamp: Long = 0  
26  
27         if (intent != null && intent.extras != null) {  
28             timeStamp = intent.extras!!.getLong(key: "timestamp")  
29         }  
30  
31         if (timeStamp > 0) {  
32  
33             val context = this.applicationContext  
34             var notificationManager: NotificationManager =  
35                 getSystemService(Context.NOTIFICATION_SERVICE) as NotificationManager  
36             val notifyIntent = Intent(packageContext: this, MainActivity::class.java)  
37  
38             val title = "Todo List"  
39             val message = "An item in your list is due now"  
40  
41             notifyIntent.putExtra(name: "title", value: title)  
42             notifyIntent.putExtra(name: "message", value: message)  
43             notifyIntent.putExtra(name: "notification", value: true)  
44
```