

Nama : Nur Imam Masri

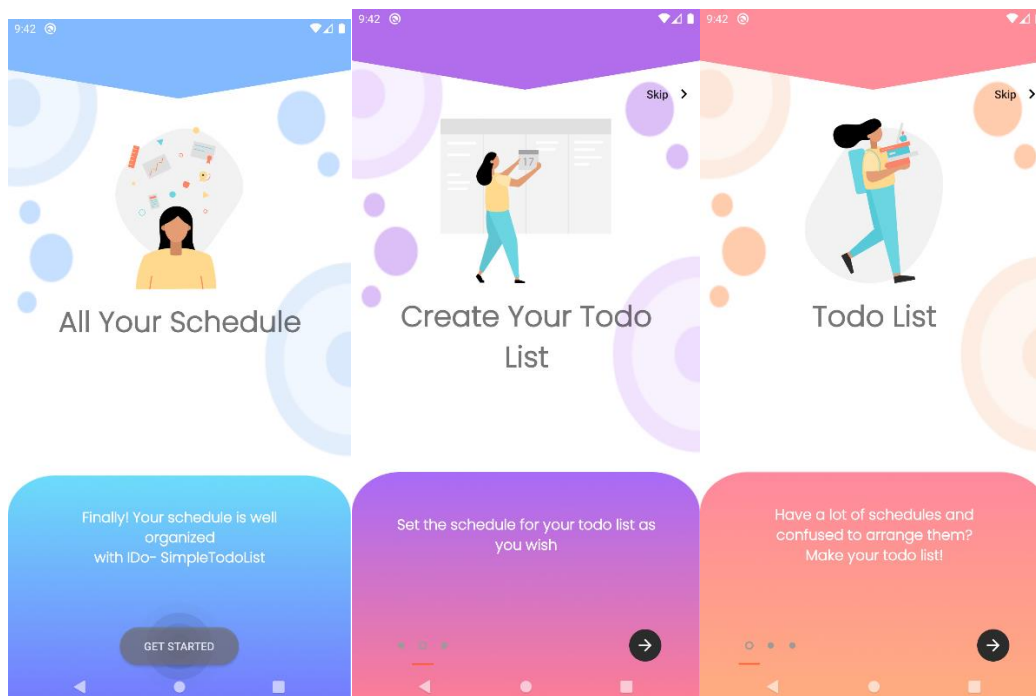
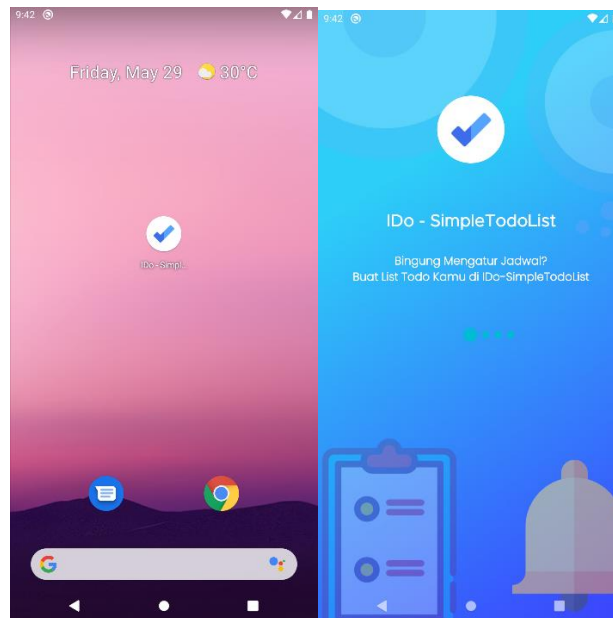
Nim : D121181322

Kelas C Mobile Teknik Informatika

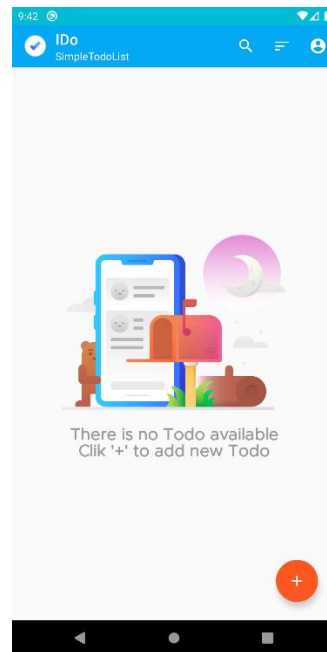
LINK GITHUB : <https://github.com/nurimammasri/IDo-SimpleToDoList.git> atau [klik disini](#)

IDo – Simple Todo List Apps

INTRO APPLICATION



TAMPILAN AWAL APLIKASI



MEMBUAT TODO LIST

Dengan Mengklik tombol button “+” pada bagian bawah dan mengisi form

Two side-by-side screenshots of the 'Create Todo' form. The left screenshot shows the empty form with fields for 'Title', 'Content', 'Date', and 'Time'. The right screenshot shows the form filled with the example 'Makan' (Eat) scheduled for Saturday, 30/05/2020 at 10:08. The form includes a 'Set Due Date and Time' section with 'SET DATE' and 'SET TIME' buttons, and a notification toggle set to '1 hour before'.

FITUR FITUR PADA CREATE TODO

Error ketika field kosong

10:09

Create Todo

Costumize Your Todo

Title

Please enter title

Content

Set your description todo in here

Set Due Date and Time

Date

Sat, 30/05/2020

SET DATE

Time

10:08

SET TIME

set notifications 1 hour before

Reminder set for

Sat, 30/05/2020 10:08

Error Ketika menyatel Alarm Due date yang sudah Expired (Berlalu)

10:11

Create Todo

Costumize Your Todo

Title

Makan

Set your main todo in here

Content

Makan Nasi

Set your description todo in here

Set Due Date and Time

Date

Wed, 27/05/2020

SET DATE

Date Passed, Enter Again

Time

10:11

SET TIME

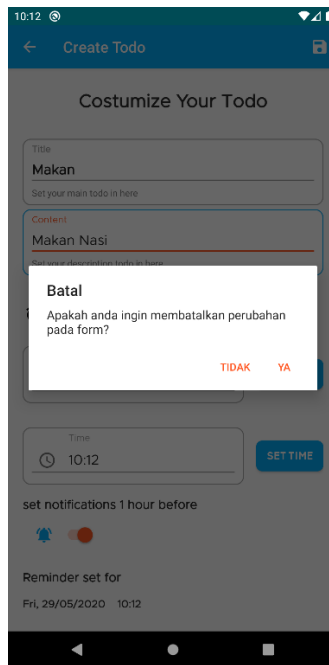
Date Passed, Enter Again

set notifications 1 hour before

Reminder set for

Wed, 27/05/2020 10:11

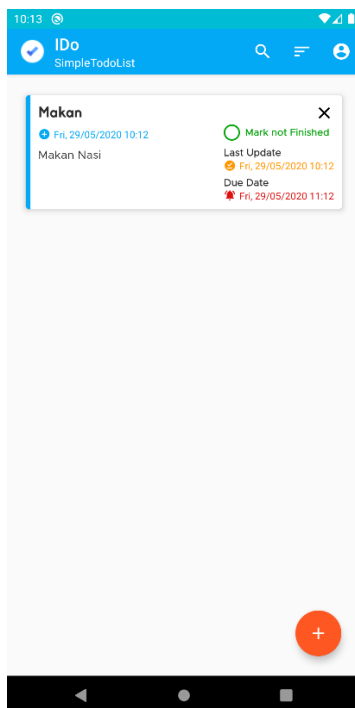
Display Dialog ketika mengklik Tombol Back Tanpa Save



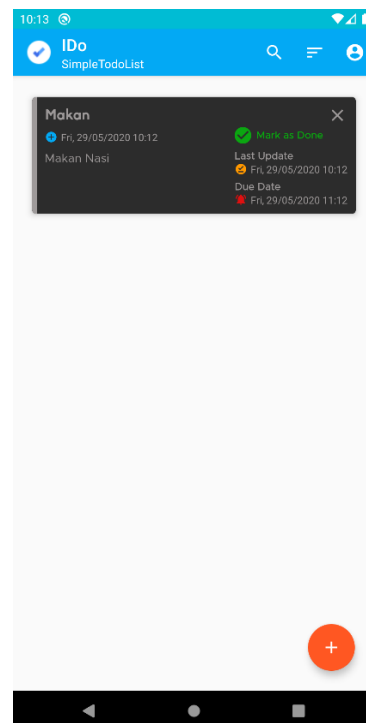
TAMPILAN SETELAH ITEM TODO DI CREATE

Dapat Menandai Ketika Item Telah Complete

Mark not Finished (Belum Selesai)

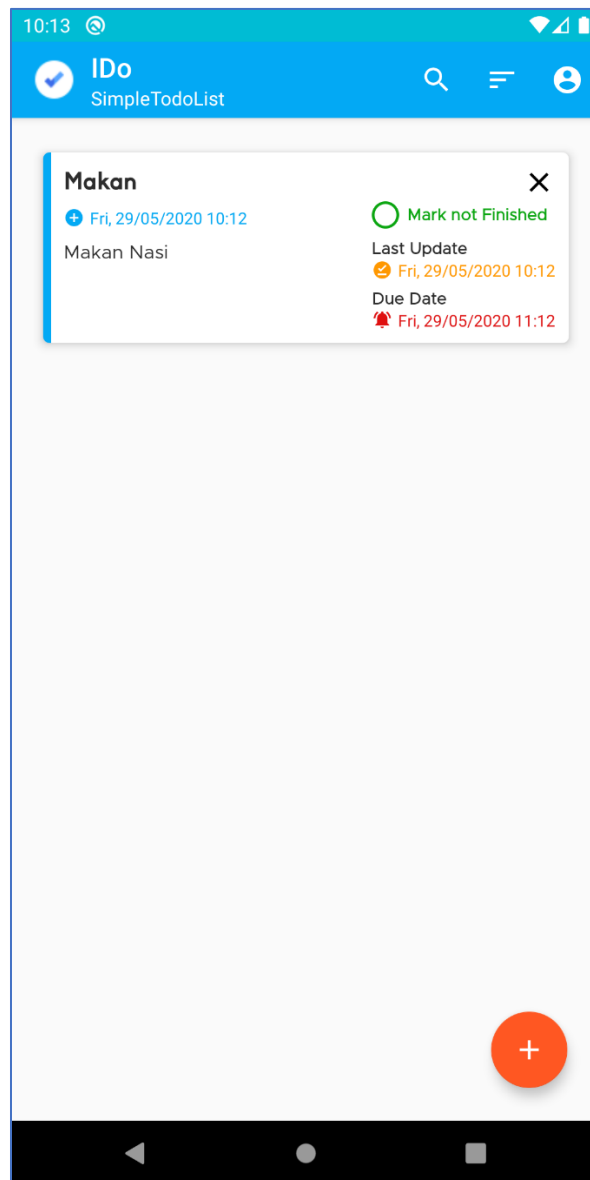


Mark as Done (Selesai)



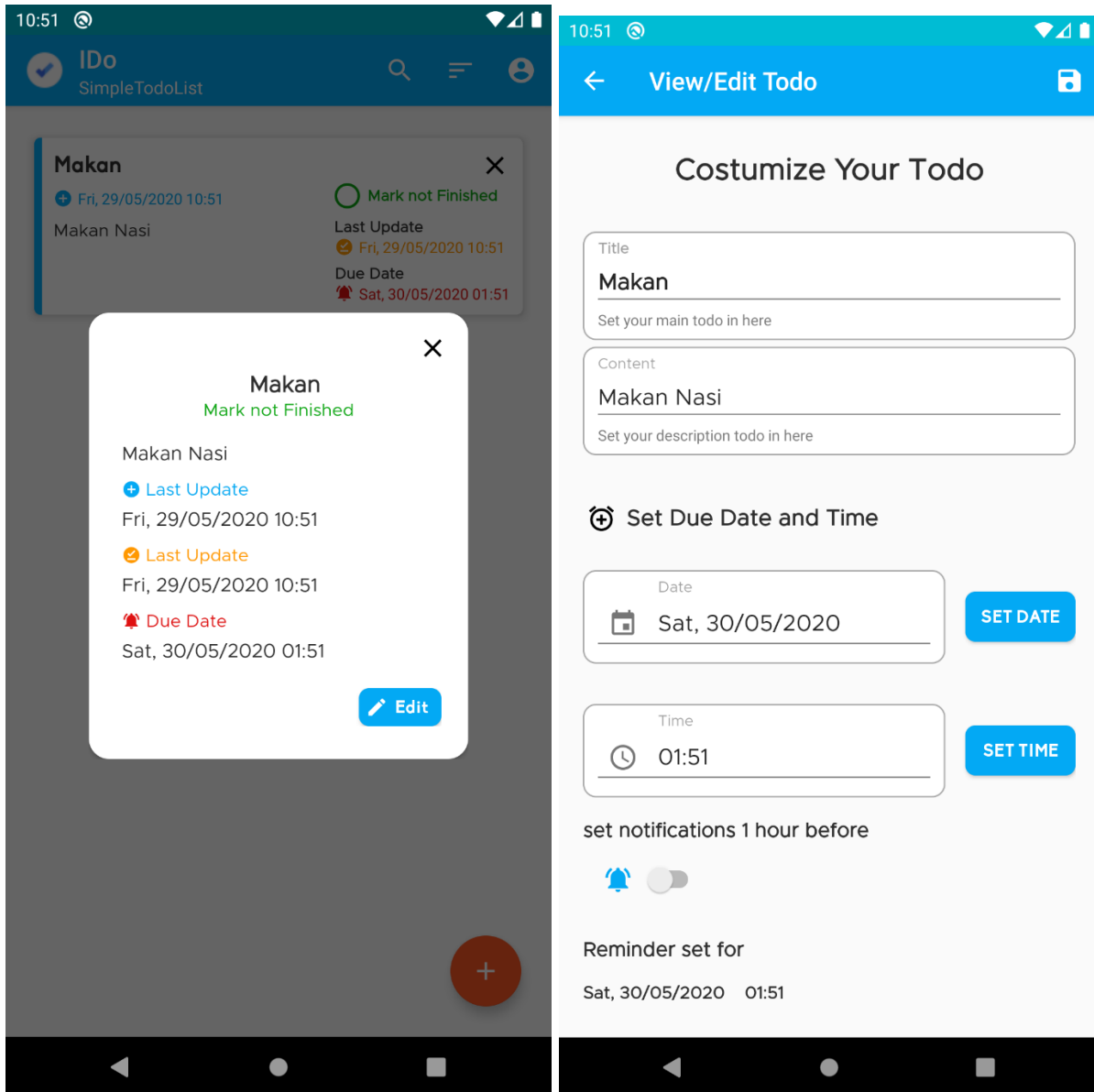
TAMPILAN SATU ITEM

- **Title**
- **Deskripsi Item**
- **Create Date**
- **Last Update**
- **Due Date**
- **Mark (Done / not Finished)**
- **X (Untuk Menghapus)**



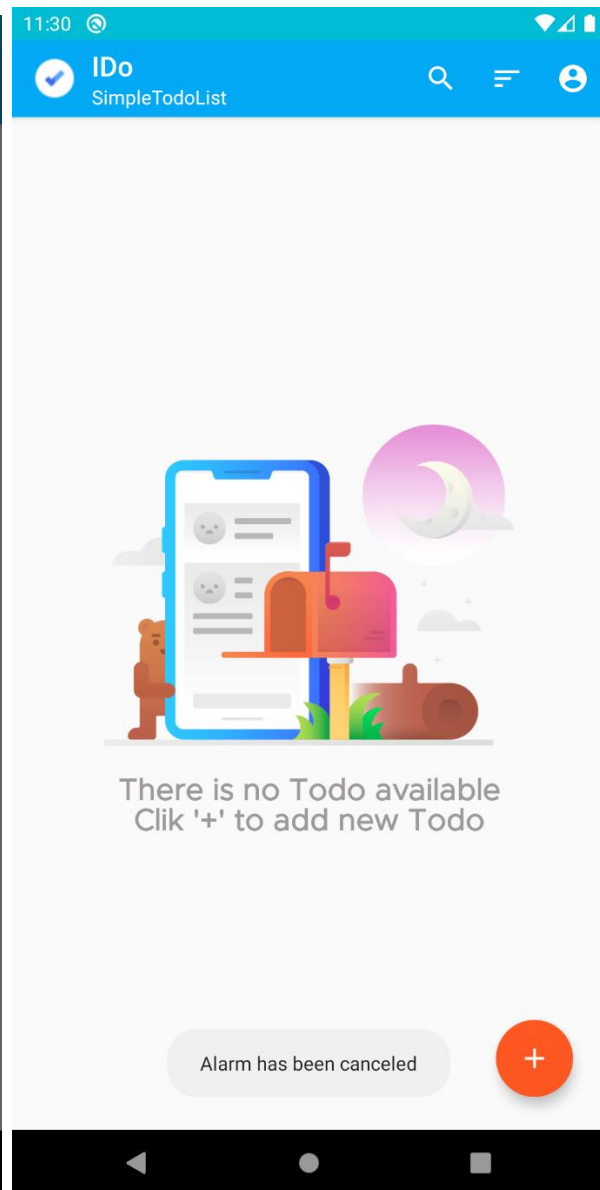
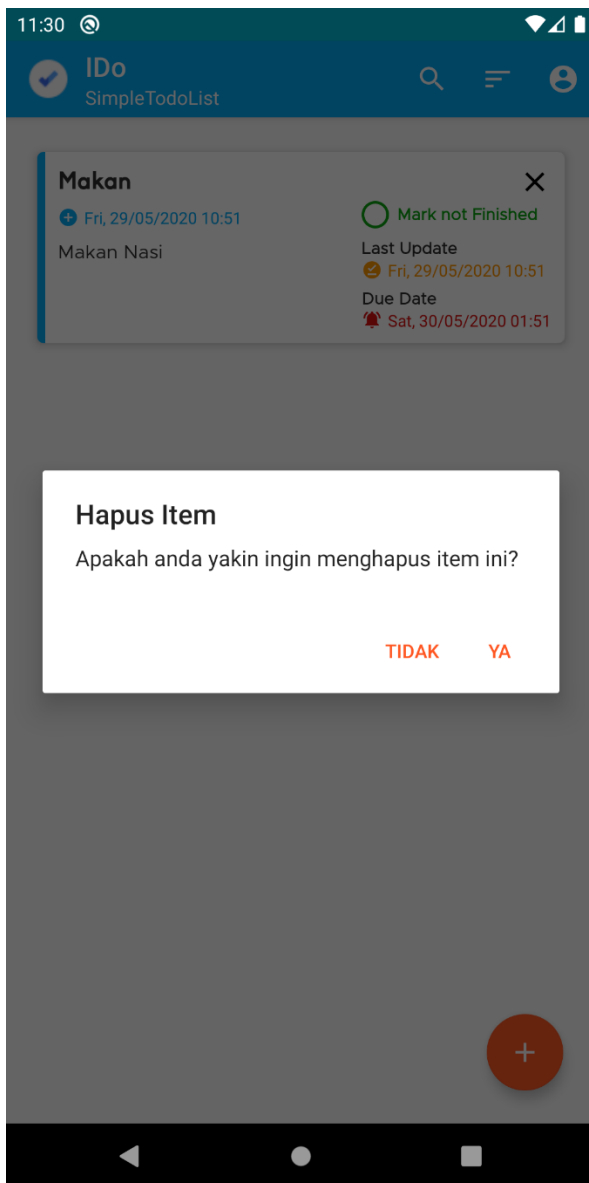
VIEW DAN EDIT TODO LIST

Untuk Menampilkan Detail Todo List dengan mengklik item maka akan muncul dialog, untuk mengedit klik tombol edit, maka akan di alihkan ke bagian form edit



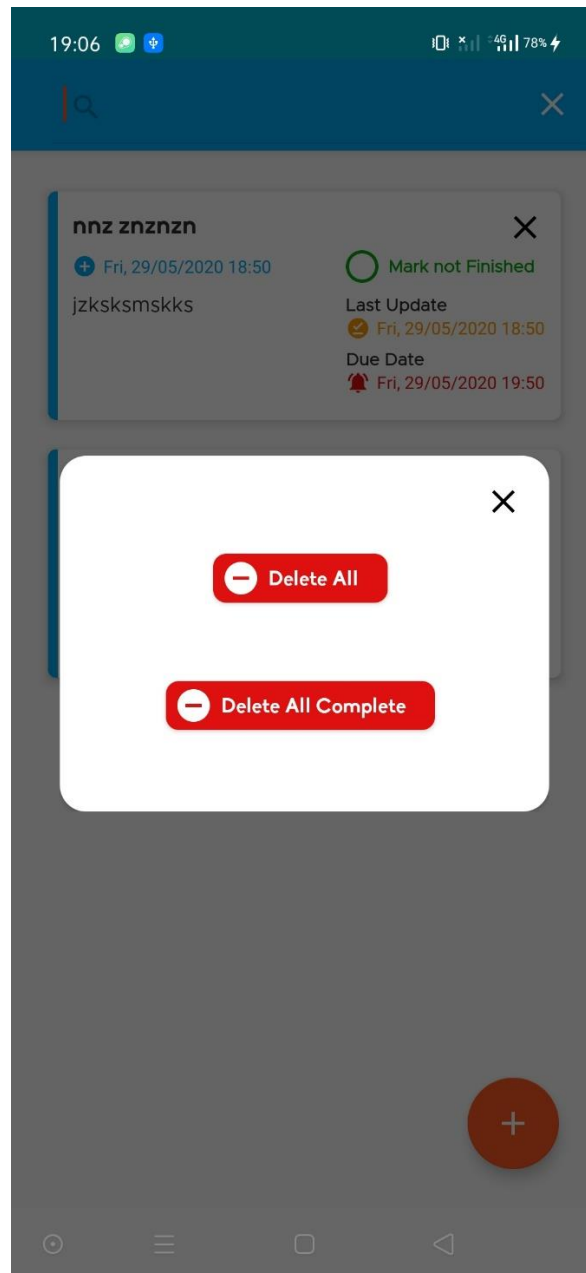
MENGHAPUS ITEM TODO LIST

Untuk menghapus Item List, dengan mengklik tombol silang pada kanan atas item, setelah itu ketika item kosong akan muncul gambar dan text keterangan.



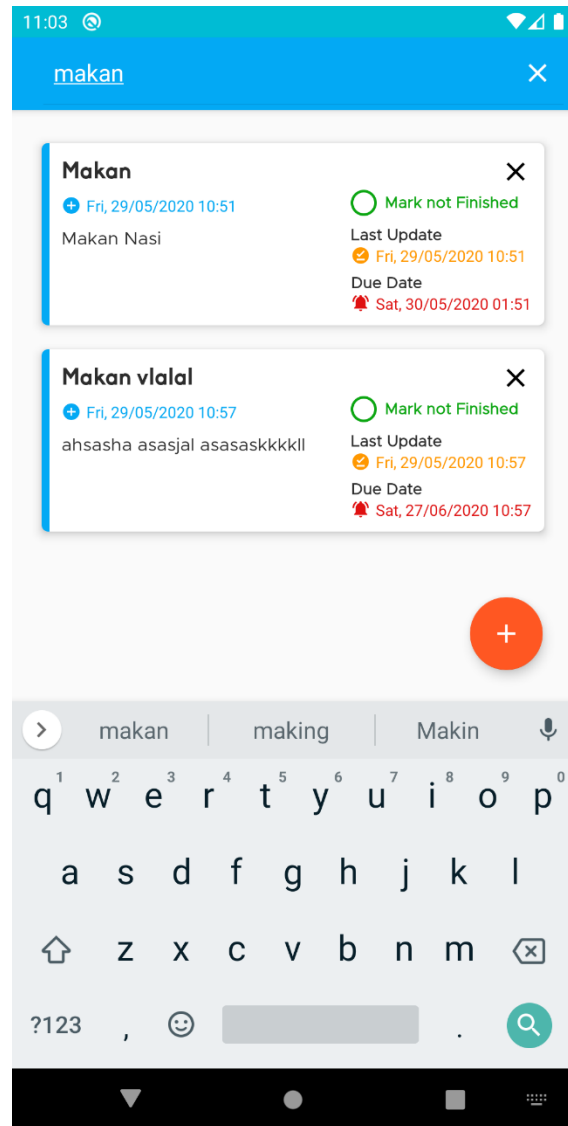
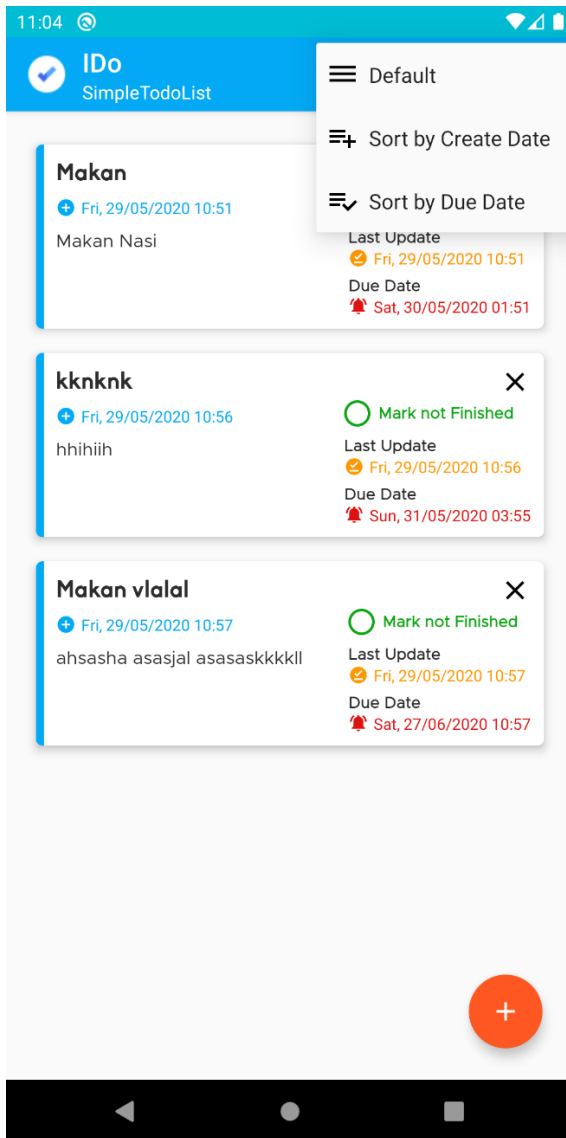
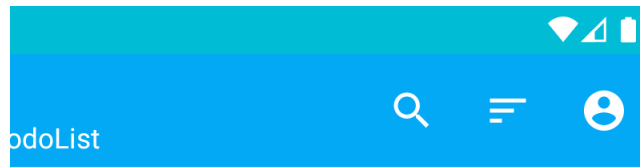
UNTUK MENGHAPUS SELURUH ITEM ATAU ITEM YANG SUDAH COMPLETE

Yakni dengan mengklik lama pada satu item, maka akan muncul dialog untuk menghapus semua “Delete All” dan menghapus yang complete “Delete All Complete”



FUNGSI SEARCH, SORT, DAN MENU ABOUT

Ada pada bagian kanan atas layar utama



11:04



Ido - Simple Todo List



NUR IMAM MASRI

D121181322

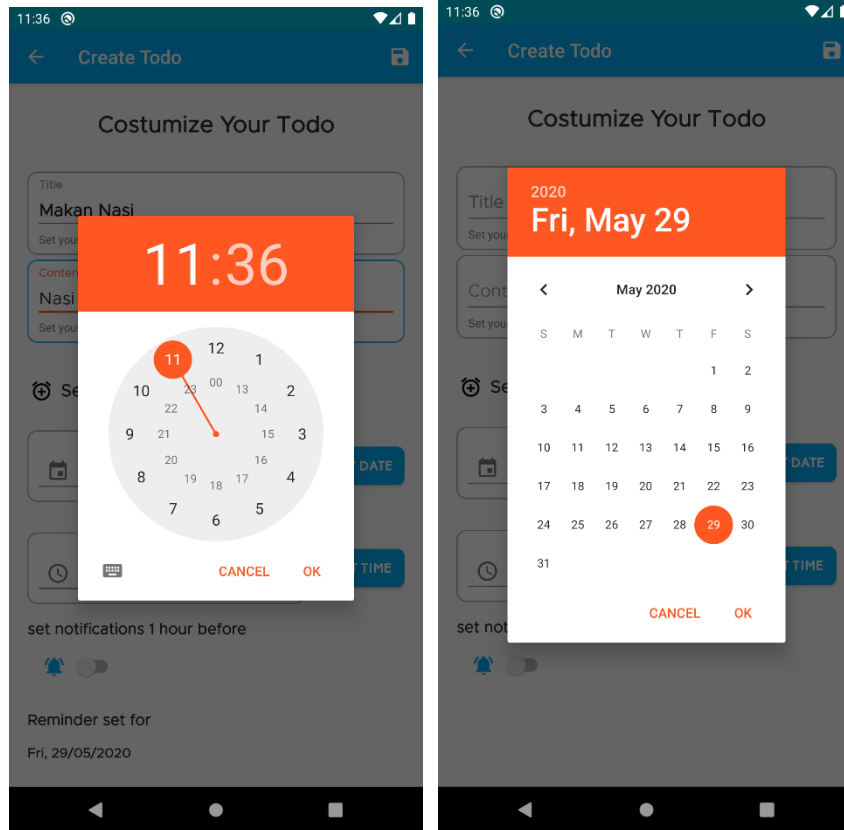
nurimammasri.01@gmail.com



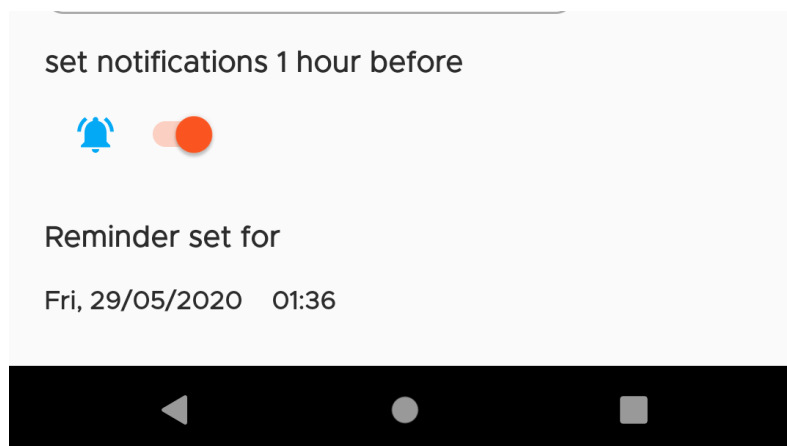
MENYETEL ALARAM

ALARAM PADA DUE DATE DAN ALARAM 1 JAM SEBELUMNYA

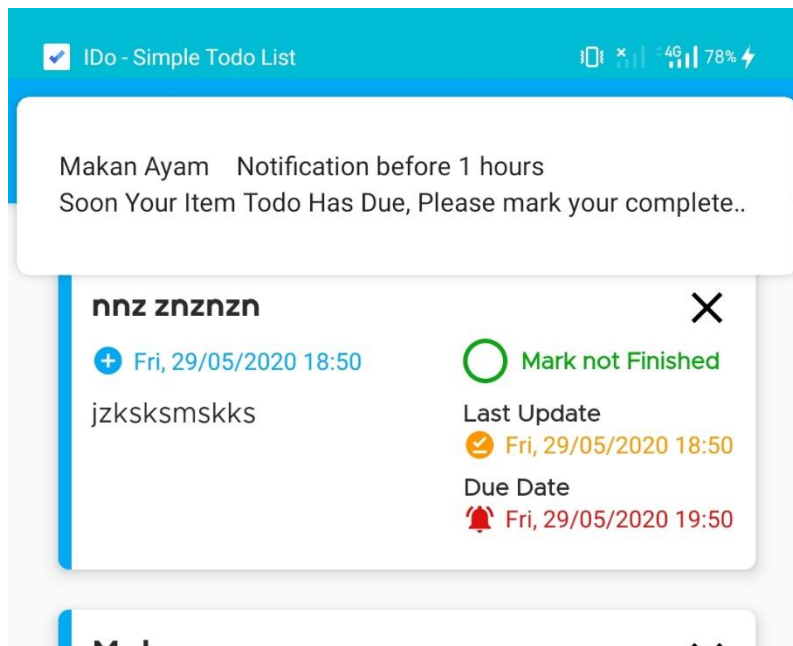
Menyetel Due Date (Waktu Berakhir) dengan Time dan Date Picker Dialog



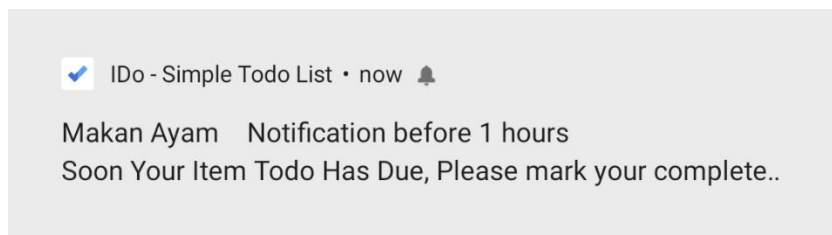
Dan Untuk menyetel Alarm 1 jam sebelum yakni dengan mengaktifkan switch



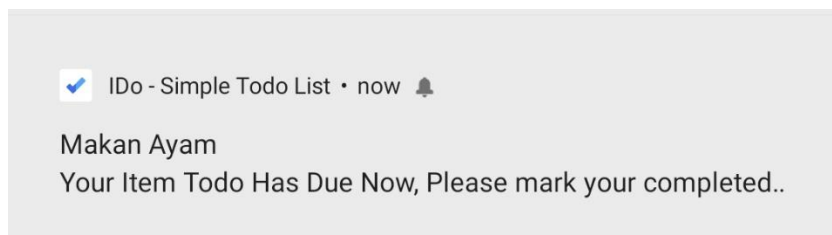
Maka akan muncul notifikasi



- 1 jam sebelum

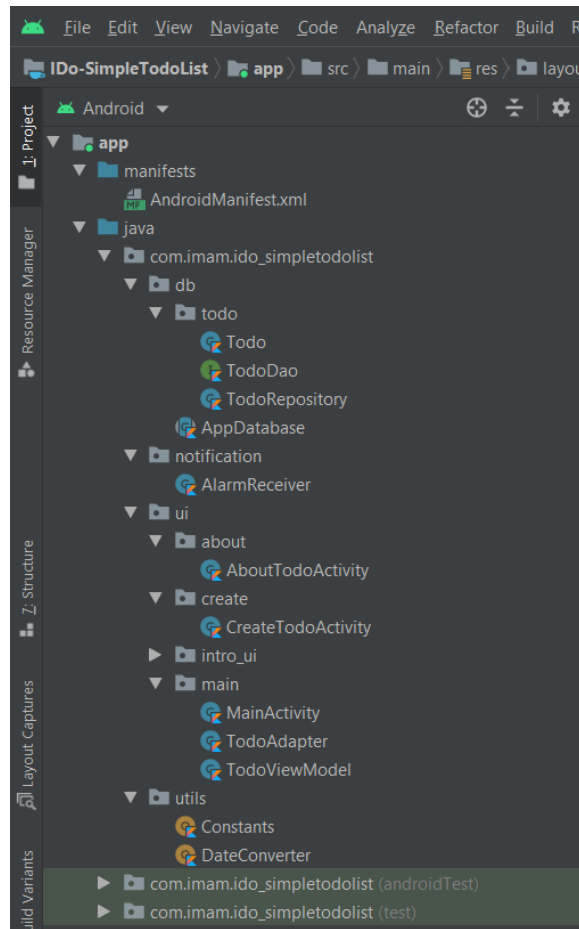


- Pas Pada Due Date



PENJELASAN CODE APLIKASI

SUSUNAN DIRECTORY



- **Db > todo**
Berisi bagian kebutuhan untuk Room Database, **Todo (Entity)**, **TodoDao (DAO)**, **TodoRepository** dan **AppDatabase**.
- **Notification**
Untuk bagian Notification pada **NavigationCompatBuilder** dan **AlarmManager**
- **UI**
Berisi bagian UI untuk Activity, **MainActivity**, **CreateToDoActivity**, **AboutActivity**
Berisi **Adapter (TodoAdapter)** dan **ViewModel (TodoViewModel)** untuk RecyclerView
- **Utils**
Untuk bagian tambahan untuk Konversi Tanggal untuk Calendar dan Cons Extra untuk Intent

ROOM DATABASE

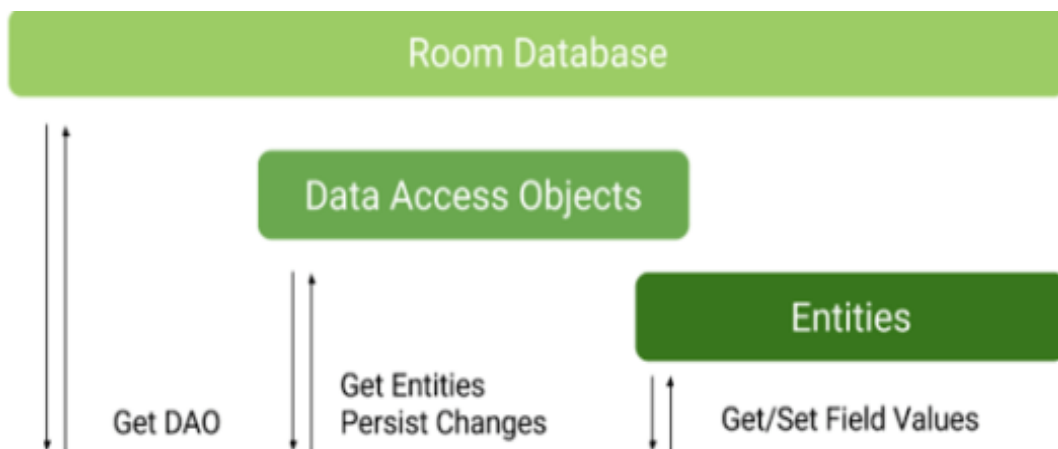
Perbedaan mendasar antara Room dan tutorial SQLite sebelumnya adalah :

1. **sqlite**—Menggunakan SQLiteOpenHelper dan interface SQLite standar (DBHelper, dsb).
2. **room**—Full abstraction menggunakan Room library dan akses yang lebih efisien.

Pada prinsipnya Room dibagi menjadi tiga buah komponen utama, yaitu Database, Entity, dan DAO :

- **Entity** merepresentasikan data pada sebuah tabel seperti di database pada umumnya, dibuat menggunakan annotation pada java data object. Setiap entity mempunyai satu tabel sendiri.
- **DAO** (Data Access Object) menggambarkan method2 yang mengakses database, termasuk methods standar seperti CRUD (Create, Read, Update, Delete). Menggunakan annotation untuk mengikat SQL query ke suatu method.
- **Database** adalah holder class yang menggunakan annotation untuk menampilkan daftar dari entity2 yang ada dan juga database version. Kelas ini juga berisi daftar dari DAO yang ada.

Penggambarannya bisa dilihat pada diagram Room Database di bawah ini :



Room Database

MENAMBAHKAN DEPENDENSI

- **RecyclerView**

```
// RecyclerView  
implementation "androidx.recyclerview:recyclerview:1.1.0"
```

Menggunakan RecyclerView untuk menampilkan data dalam bentuk list. Dengan menghubungkan Data ke View menggunakan ViewModel

- **Coroutines**

```
// coroutines  
implementation "org.jetbrains.kotlinx:kotlinx-coroutines-android:1.3.2"
```

Disebabkan karena Room bekerja dibelakang layar atau Background Thread maka diperlukan penggunaan coroutine

- **Room**

Pada project ini kita perlu menambahkan beberapa library pada build.gradle dan memastikan telah apply kedua plugin di bawah ini:

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

```
// Room  
implementation "androidx.room:room-runtime:2.2.5"  
implementation "androidx.room:room-ktx:2.2.5"  
kapt "androidx.room:room-compiler:2.2.5"
```

- **ViewModel**

```
// ViewModel  
implementation "androidx.lifecycle:lifecycle-extensions:2.2.0"  
kapt "androidx.lifecycle:lifecycle-compiler:2.2.0"
```

Untuk mengakses komponen data untuk mempertahankannya.

- **Material Design**

```
//Material CardView  
implementation 'androidx.cardview:cardview:1.0.0'  
implementation 'com.google.android.material:material:1.1.0'
```

```
//Circle Image  
implementation 'de.hdodenhof:circleimageview:3.0.1'  
  
//Lottie  
implementation 'com.airbnb.android:lottie:3.1.0'
```

CardView

Untuk menampilkan data dari RecyclerView dalam bentuk CardView dengan bayang dan corner radius.

CircleImage

Digunakan untuk menampilkan Image dan bentuk lingkaran

Lottie

Fungsi Library untuk Animasi Json

KOMPONEN ROOM DATABASE

- **Entity**


```

11
12 @Entity(tableName = "todo")
13 @Parcelize
14 data class Todo(
15
16     @PrimaryKey(autoGenerate = true)
17     @ColumnInfo(name = "id")
18     val id: Int? = null,
19
20     @ColumnInfo(name = "title")
21     val title: String,
22
23     @ColumnInfo(name = "content")
24     val content: String,
25
26     @ColumnInfo(name = "created_at", defaultValue = "CURRENT_TIMESTAMP")
27     @TypeConverters(DateConverter::class)
28     val createdAt: Date,
29
30     @ColumnInfo(name = "update_at", defaultValue = "CURRENT_TIMESTAMP")
31     @TypeConverters(DateConverter::class)
32     val updatedAt: Date,
33
34     @ColumnInfo(name = "due_date")
35     @TypeConverters(DateConverter::class)
36     val dueAt: Date,
37
38     @ColumnInfo(name = "finished")
39     var finished: Boolean,
40
41     @ColumnInfo(name = "check_alarm_hour")
42     var checkAlarmHour: Boolean
43 ) : Parcelable

```

Entity adalah kelas yang disimpan di Database. Tabel database eksklusif dibuat untuk setiap kelas yang diberi catatan @Entity.

Entity mewakili objek yang disimpan dalam database. Setiap kelas Entity membuat tabel database baru, dengan masing-masing bidang mewakili kolom. Anotasi digunakan untuk mengonfigurasi entitas, dan proses pembuatannya sangat sederhana.

Terkait dengan kelas diatas, berikut beberapa keterangan yang dapat dipelajari.

1. Kelas menggunakan anotasi @Entity dan menggunakan properti tableName untuk menentukan nama tabel di database.
2. Primary key ditentukan dengan menambahkan anotasi @PrimaryKey pada field yang diinginkan untuk menjadi primary key. Dalam contoh diatas pada field mId.
3. Nama kolom tabel dapat diatur dengan anotasi @ColumnInfo(name="column_name").

Pada Entry App ini Ada beberapa attribute yaitu

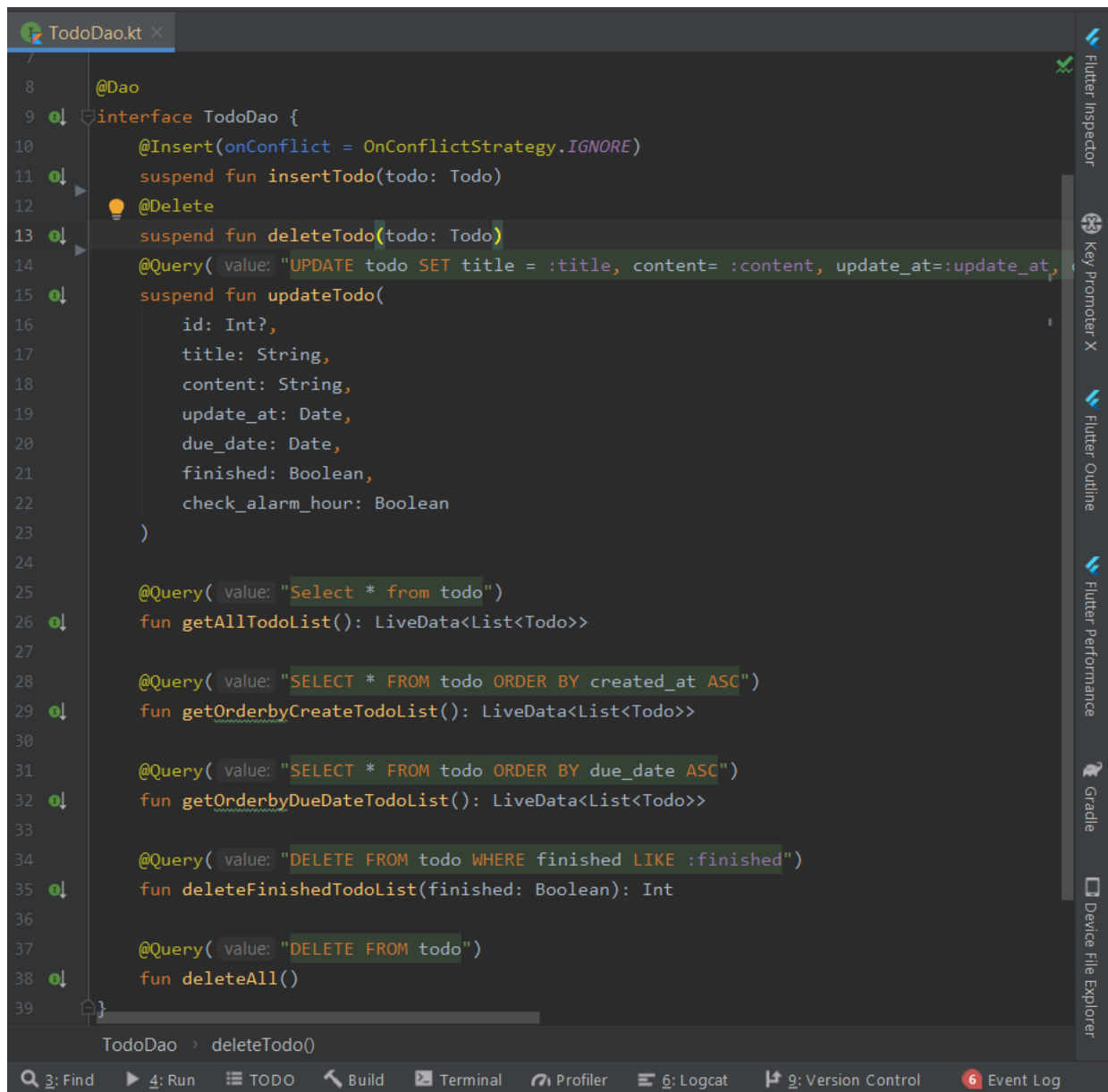
- **ID = sebagai id**

- **Title** = untuk judul Todo List
- **Content** = deskripsi Todo List
- **Create Date** = Tanggal Dibuat
- **Last Update Date** = Update Terakhir
- **Due Date** = Tanggal Berakhir
- **Finished** = Cek Apakah sudah Complete
- **Check Alarm Hour** = untuk mengecek apakah dibutuhkan Alarm 1 jam sebelum

Yang akan dikirimkan ke bagian Activity lain dengan fungsi **Parcelable**

- **DAO**

DAO adalah antarmuka yang dianotasikan dengan @Dao yang memediasi akses ke objek dalam database dan tabelnya. Ada empat anotasi khusus untuk operasi dasar DAO: @Insert, @Update, @Delete, dan @Query.



```
8  @Dao
9  interface TodoDao {
10     @Insert(onConflict = OnConflictStrategy.IGNORE)
11     suspend fun insertTodo(todo: Todo)
12     @Delete
13     suspend fun deleteTodo(todo: Todo)
14     @Query( value: "UPDATE todo SET title = :title, content= :content, update_at=:update_at,")
15     suspend fun updateTodo(
16         id: Int?,
17         title: String,
18         content: String,
19         update_at: Date,
20         due_date: Date,
21         finished: Boolean,
22         check_alarm_hour: Boolean
23     )
24
25     @Query( value: "Select * from todo")
26     fun getAllTodoList(): LiveData<List<Todo>>
27
28     @Query( value: "SELECT * FROM todo ORDER BY created_at ASC")
29     fun getOrderbyCreateTodoList(): LiveData<List<Todo>>
30
31     @Query( value: "SELECT * FROM todo ORDER BY due_date ASC")
32     fun getOrderbyDueDateTodoList(): LiveData<List<Todo>>
33
34     @Query( value: "DELETE FROM todo WHERE finished LIKE :finished")
35     fun deleteFinishedTodoList(finished: Boolean): Int
36
37     @Query( value: "DELETE FROM todo")
38     fun deleteAll()
39 }
```

Yang pada Penerimaan Data dibuat **LIVE DATA**

LiveData Queries, Kamar dirancang untuk bekerja dengan anggun dengan LiveData. Untuk @Query mengembalikan LiveData, cukup selesaikan pengembalian standar dengan LiveData <?> Dan Anda siap untuk pergi.

Pada Entry app ini, Ada beberapa fungsi akses untuk database ada beberapa

- **insertTodo** = untuk insert per item Todo List
- **deleteTodo** = untuk delete per item Todo List
- **updateTodo** = untuk update perubahan pada Todo List

- **getAllTodoList** = untuk mendapatkan (Select) semua item
- **getOrderByCreateTodoList** = untuk memilih item berdasarkan urutan tanggal dibuat
- **getOrderByDueDateTodoList** = untuk memilih item berdasarkan urutan tanggal berakhir
- **deleteFinishedTodoList** = untuk menghapus semua item yang telah complete
- **deleteAll** = menghapus semua item sekaligus

• Repository

Sebagai akses fungsi akses Database ke ViewModel.

```

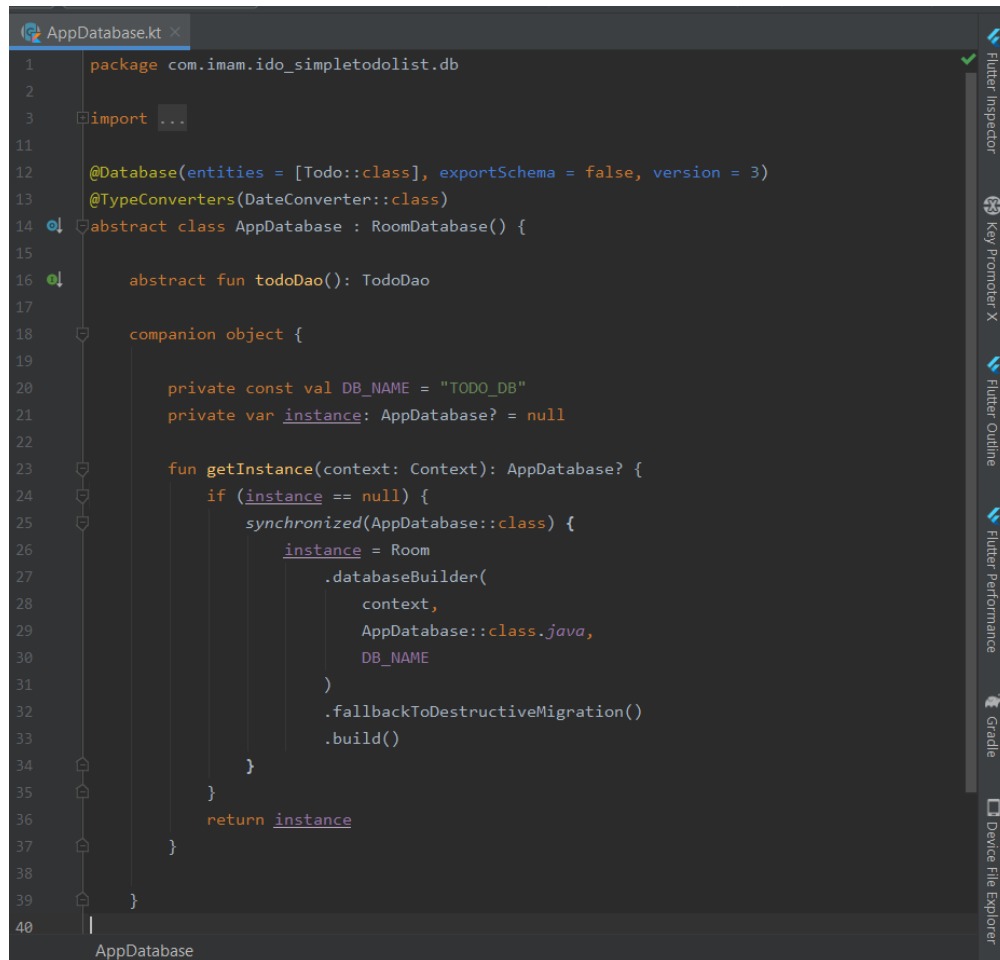
9
10 class TodoRepository(application: Application) {
11     private val todoDao: TodoDao?
12     private var todos: LiveData<List<Todo>>? = null
13     private var todosOrderedbyCreate: LiveData<List<Todo>>? = null
14     private var todosOrderedbyDueDate: LiveData<List<Todo>>? = null
15
16     init {
17         val db = AppDatabase.getInstance(application.applicationContext)
18         todoDao = db?.todoDao()
19         todos = todoDao?.getAllTodoList()
20         todosOrderedbyCreate = todoDao?.getOrderByCreateTodoList()
21         todosOrderedbyDueDate = todoDao?.getOrderByDueDateTodoList()
22     }
23
24     fun getAllTodoList(): LiveData<List<Todo>>? {...}
27
28     fun getOrderByCreateTodoList(): LiveData<List<Todo>>? {...}
31
32     fun getOrderByDueDateTodoList(): LiveData<List<Todo>>? {...}
35
36
37     fun insert(todo: Todo) = runBlocking { this: CoroutineScope
38         this.launch(Dispatchers.IO) {...}
41     }
42
43     fun deleteAll(todo: Todo) {...}
50
51     fun delete(todo: Todo) {...}
58
59     fun deleteFinishedTodoList(todo: Todo) {...}
66
67     fun update(todo: Todo) = {...}

```

Pada bagian ini fungsi yang ada, sama dengan fungsi pada DAO. Pada fungsi ini akan diteruskan menuju ViewModel untuk diakses ke Adapter.

• Database

Komponen Database adalah kelas abstrak yang dianotasikan @Database, yang memperluas RoomDatabase. Kelas mendefinisikan daftar Entitas dan DAO-nya.



```
1 package com.imam.ido_simpлетodolist.db
2
3 import ...
4
11
12 @Database(entities = [Todo::class], exportSchema = false, version = 3)
13 @TypeConverters(DateConverter::class)
14 abstract class AppDatabase : RoomDatabase() {
15
16     abstract fun todoDao(): TodoDao
17
18     companion object {
19
20         private const val DB_NAME = "TODO_DB"
21         private var instance: AppDatabase? = null
22
23         fun getInstance(context: Context): AppDatabase? {
24             if (instance == null) {
25                 synchronized(AppDatabase::class) {
26                     instance = Room
27                         .databaseBuilder(
28                             context,
29                             AppDatabase::class.java,
30                             DB_NAME
31                         )
32                         .fallbackToDestructiveMigration()
33                         .build()
34                 }
35             }
36             return instance
37         }
38     }
39 }
40
AppDatabase
```

- **Data Type dan Konversi Data**

Kolom Datatype secara otomatis ditentukan oleh Room. Sistem akan menyimpulkan dari jenis bidang mana jenis SQLite Datatype lebih memadai. Perlu diingat bahwa sebagian besar POJO Java akan dikonversi dari kotak; Namun, perlu untuk membuat konverter data untuk menangani objek yang lebih kompleks yang tidak dikenal oleh Room secara otomatis, seperti Date dan Enum.

Untuk Ruang untuk memahami konversi data, diperlukan untuk menyediakan **TypeConverters** dan mendaftarkan konverter tersebut di Ruang. Dimungkinkan untuk membuat pendaftaran ini dengan mempertimbangkan konteks khusus—misalnya, jika Anda mendaftarkan TypeConverter dalam Database, semua entitas dari basis data akan menggunakan konverter. Jika Anda mendaftar pada suatu entitas, hanya properti entitas yang dapat menggunakannya, dan seterusnya.

Untuk mengonversi objek Date secara langsung ke selama Room mengoperasi penghematan Long dan kemudian mengonversi Long ke sebuah Date ketika berkonsultasi dengan database, pertama-tama deklarasikan **TypeConverter**.

```
DateConverter.kt x
1 package com.imam.ido_simpletodolist.utils
2
3 import ...
4
5
6 object DateConverter {
7     @TypeConverter
8     @JvmStatic
9     fun toDate(timeStamp: Long?): Date? {
10         return timeStamp?.let { Date(it) }
11     }
12
13     @TypeConverter
14     @JvmStatic
15     fun toTimeStamp(date: Date?): Long? {
16         return date?.time
17     }
18 }
```

MEMBUAT ACTIVITY, ADAPTER DAN VIEWMODEL

- **ViewModel (MVVM Architecture)**

Beberapa layer pada arsitektur MVVM :

Model

Model / entity adalah representasi dari data yang digunakan pada business logic, dapat berupa Plain Old Java Object (POJO), Kotlin Data Classes, dll.

View

Representasi UI dari sebuah aplikasi, pada Android sendiri view ini dapat berupa Activity atau Fragment.

ViewModel

Layer yang berinteraksi langsung dengan Model, serta menyajikan data untuk View layer.

```
1 package com.imam.ido_simpletodolist.ui.main
2
3 import ...
4
5 class TodoViewModel(application: Application) : AndroidViewModel(application) {
6
7     private val repository: TodoRepository = TodoRepository(application)
8     private val allTodoList: LiveData<List<Todo>>? = repository.getAllTodoList()
9     private val allOrderByCreateTodoList: LiveData<List<Todo>>? =
10         repository.getOrderbyCreateTodoList()
11     private val allOrderByDueDateTodoList: LiveData<List<Todo>>? =
12         repository.getOrderbyDueDateTodoList()
13
14     fun insertTodo(todo: Todo) {...}
15
16     fun updateTodo(todo: Todo) {...}
17
18     fun deleteTodo(todo: Todo) {...}
19
20     fun deleteAll(todo: Todo) {...}
21
22     fun deleteFinished(todo: Todo) {...}
23
24     fun getAllTodoList(): LiveData<List<Todo>>? {...}
25
26     fun getOrderbyCreateTodoList(): LiveData<List<Todo>>? {...}
27
28     fun getOrderbyDueDateTodoList(): LiveData<List<Todo>>? {...}
29
30     fun setFinishedItemTodo(todo: Todo) {...}
31
32 }
```

Pada bagian ini akan di buat fungsi yang memiliki fungsi yang sama seperti pada repository.

- Mendeklarasikan RecyclerView Pada View

content_main.xml

```

1      <?xml version="1.0" encoding="utf-8"?>
2
3      <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
4          android:id="@+id/layout_recyclerview"
5          xmlns:tools="http://schemas.android.com/tools"
6          android:padding="16dp"
7          android:layout_below="@+id/toolbarMain"
8          android:layout_width="match_parent"
9          android:layout_height="match_parent"
10         android:orientation="vertical">
11
12         <androidx.recyclerview.widget.RecyclerView
13             android:id="@+id/rv_todo_list"
14             android:layout_width="match_parent"
15             android:layout_height="match_parent"
16             tools:listitem="@layout/item_todo" />
17
18     </LinearLayout>

```

- Membuat Item Row View

Item_todo.xml

```

1      <androidx.cardview.widget.CardView xmlns:android="http://schemas.android.com/apk/res-auto"
2          xmlns:cardview="http://schemas.android.com/apk/res-auto"
3          xmlns:tools="http://schemas.android.com/tools"
4          android:id="@+id/layout_item"
5          android:layout_width="match_parent"
6          android:layout_height="match_parent"
7          card:cardCornerRadius="5dp"
8          card:cardElevation="5dp"
9          card:cardUseCompatPadding="true">
10
11         <ImageView...>
12
13         <RelativeLayout
14             android:id="@+id/content"
15             android:layout_width="match_parent"
16             android:layout_height="match_parent"
17             android:paddingStart="15dp"
18             android:paddingTop="10dp"
19             android:paddingEnd="10dp"
20             android:paddingBottom="10dp">
21
22             <TextView...>
23
24             <TextView...>
25
26             <TextView...>
27
28             <LinearLayout...>
29
30             <ImageView...>
31
32     </RelativeLayout>

```


- Deklarasikan Adapter yang extends ke RecyclerView.Adapter

Untuk memasukkan semua data Anda ke daftar, Anda harus memperluas class RecyclerView.Adapter. Objek ini membuat tampilan untuk item, dan mengganti konten sebagian tampilan dengan item data baru ketika item asli tidak lagi terlihat.

```

1 package com.imam.ido_simpletodolist.ui.main
2
3 import ...
4
5
6
7
8
9
10
11
12
13
14
15
16
17 class TodoAdapter(todoEvents: TodoEvents) : RecyclerView.Adapter<TodoAdapter.ViewHolder>(),
18     Filterable {
19
20     private var todoList: List<Todo> = arrayListOf()
21     private var filteredTodoList: List<Todo> = arrayListOf()
22     private val listener: TodoEvents = todoEvents
23
24     override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): ViewHolder {
25         val view = LayoutInflater.from(parent.context).inflate(R.layout.item_todo, parent,
26             false)
27         return ViewHolder(view)
28     }
29
30     override fun getItemCount(): Int = filteredTodoList.size
31
32     override fun onBindViewHolder(holder: ViewHolder, position: Int) {
33         holder.bind(filteredTodoList[position], listener)
34     }
35
36     class ViewHolder(itemView: View) : RecyclerView.ViewHolder(itemView) {
37         ...
38     }
39
40     /**
41      * Activity uses this method to update todoList with the help of LiveData
42      */
43     fun setAllTodoList(todoItems: List<Todo>) {
44         this.todoList = todoItems
45         this.filteredTodoList = todoItems
46         notifyDataSetChanged()
47     }
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200

```

Yang mana pada RecyclerView ini ada implementasi method yang wajib yaitu

- onCreateViewHolder = untuk mengambil view item dan menginflatnya
- onBindViewHolder = untuk membind atau memasukkan data ke item view
- getItemCount = mendapatkan banyak item yang ada
- ViewHolder = untuk inisialisasi objek View

Dan untuk LIVEDATA menggunakan fungsi tambahan untuk mengupdate yang ada pada AdapterView.

```
/**
 * Activity uses this method to update todoList with the help of LiveData
 * */
fun setAllTodoList(todoItems: List<Todo>) {
    this.todoList = todoItems
    this.filteredTodoList = todoItems
    notifyDataSetChanged()
}
```

Dan membuat interface untuk behaviour item Klik

```
/**
 * RecyclerView touch event callbacks
 * */
interface TodoEvents {
    fun onDeleteClicked(todo: Todo)
    fun onViewClicked(todo: Todo)
    fun onViewLongClicked(todo: Todo)
    fun onCheckFinishedClicked(
        todo: Todo,
        itemView: View
    )
}
```

- Memperbaharui MainActivity

Untuk mengatur segala mengenai behaviour activity_main.xml

```

1 package com.iman.ido_simplertodolist.ui.main
2
3 import xxx
4
5 class MainActivity : AppCompatActivity(), TodoAdapter.TodoEvents {
6
7     private lateinit var alarmReceiver: AlarmReceiver
8
9     private lateinit var todoViewModel: TodoViewModel
10    private lateinit var todoAdapter: TodoAdapter
11
12    override fun onCreate(savedInstanceState: Bundle?) {
13        super.onCreate(savedInstanceState)
14        setContentView(R.layout.activity_main)
15
16        alarmReceiver = AlarmReceiver()
17
18        //Setting Toolbar
19        val toolbar: Toolbar = findViewById(R.id.toolbarMain)
20        setSupportActionBar(toolbar)
21        supportActionBar?.title = "IDo"
22        toolbar.subtitle = "SimpleToDoList"
23        toolbar.setLogo(R.drawable.icon)
24
25        //Setting up RecyclerView
26        rv_todo_list.layoutManager = LinearLayoutManager(context, this)
27        todoAdapter = TodoAdapter(todoEvents: this)
28        rv_todo_list.adapter = todoAdapter
29    }
30
31    //Setting up ViewModel and LiveData
32    todoViewModel = ViewModelProvider(owner: this).get(TodoViewModel::class.java)
33    todoViewModel.getAllTodoList()?.observe(owner: this, Observer { it: List<Todo>!
34        setVisibilityImageEmpty(it)
35        todoAdapter.setAllTodoList(it)
36    })
37
38 }

```

Pada MainActivity, fungsi dibawah ini untuk mengeset LayoutManager ke Adapter melalui ViewModel. Yang mana ada fungsi Observer untuk mengambil data dri ViewModel

```

//Setting up RecyclerView
rv_todo_list.layoutManager = LinearLayoutManager(context, this)
todoAdapter = TodoAdapter(todoEvents: this)
rv_todo_list.adapter = todoAdapter

//Setting up ViewModel and LiveData
todoViewModel = ViewModelProvider(owner: this).get(TodoViewModel::class.java)
todoViewModel.getAllTodoList()?.observe(owner: this, Observer { it: List<Todo>!
    setVisibilityImageEmpty(it)
    todoAdapter.setAllTodoList(it)
})

```

Untuk fungsi Floating Action Button untuk menambahkan data dan berpindah ke CreateToDoActivity

```

//FAB click listener
fab_new_todo.setOnClickListener { it: View!
    val intent = Intent(packageContext: this@MainActivity, CreateToDoActivity::class.java)
    startActivityForResult(intent, Constants.INTENT_CREATE_TODO)
}

```

Untuk mengimplementasikan Interface dari Adapter RecyclerView, untuk mengimplementasikan behaviour dari item seperti delete ketika menekan tombol x pada item dan lain-lain.

```
/**
 * RecyclerView Item callbacks
 */
//Callback when user clicks on Delete note
override fun onDeleteClicked(todo: Todo) {...}

//Callback when user clicks on view note
override fun onViewClicked(todo: Todo) {...}

//Callback when user clicks on view note
override fun onViewLongClicked(todo: Todo) {...}

//Callback when user clicks on CheckFinished
override fun onCheckFinishedClicked(
    todo: Todo,
    itemView: View
) {...}
```

Dan untuk Tiga menu bagian atas untuk **Search**, **Sort** dan **About** di atur pada bagian code ini

```
override fun onOptionsItemSelected(item: MenuItem): Boolean {
    when (item.itemId) {
        R.id.sort_default -> {
            todoViewModel.getAllTodoList()?.observe( owner: this, Observer { it: List<Todo>? }) {
                setVisibilityImageEmpty(it)
                todoAdapter.setAllTodoList(it)
            }
        }
        R.id.sort_bycreate -> {
            todoViewModel.getOrderbyCreateTodoList()?.observe( owner: this, Observer { it: List<Todo>? }) {
                setVisibilityImageEmpty(it)
                todoAdapter.setAllTodoList(it)
            }
        }
        R.id.sort_bydue -> {
            todoViewModel.getOrderbyDueDateTodoList()?.observe( owner: this, Observer { it: List<Todo>? }) {
                setVisibilityImageEmpty(it)
                todoAdapter.setAllTodoList(it)
            }
        }
        R.id.about -> {
            val intent = Intent( packageContext: this@MainActivity, AboutTodoActivity::class.java)
            startActivity(intent)
        }
    }

    return super.onOptionsItemSelected(item)
}
```

Untuk menu menu itu kita buat terlebih dahulu untuk dapat di set pada activity_main.xml, dan pada kode ini juga kita deklarasikan fung **Search dengan Filter dan SearchManager**.

```
* Initialize Menu on Main
**/
override fun onCreateOptionsMenu(menu: Menu?): Boolean {
    menuInflater.inflate(R.menu.main_menu, menu)
    val searchManager = getSystemService(Context.SEARCH_SERVICE) as SearchManager
    val searchItem = menu?.findItem(R.id.search)
    val searchView: SearchView = searchItem?.actionView as SearchView
    val searchIcon = searchView.findViewById<ImageView>(androidx.appcompat.R.id.search_button)
    searchIcon.imageTintList = ColorStateList.valueOf(resources.getColor(R.color.white))
    val closeIcon = searchView.findViewById<ImageView>(androidx.appcompat.R.id.search_close_btn)
    closeIcon.imageTintList = ColorStateList.valueOf(resources.getColor(R.color.white))
    val editText = searchView.findViewById<EditText>(androidx.appcompat.R.id.search_src_text)
    editText.setTextColor(resources.getColor(R.color.white))
    searchView.setSearchableInfo(
        searchManager
            .getSearchableInfo(componentName)
    )
    searchView.maxWidth = Integer.MAX_VALUE
    searchView.setOnQueryTextListener(object : SearchView.OnQueryTextListener {
        override fun onQueryTextSubmit(query: String?): Boolean {
            todoAdapter.filter.filter(query)
            return false
        }

        override fun onQueryTextChange(newText: String?): Boolean {
            todoAdapter.filter.filter(newText)
            return false
        }
    })
    return true
}
```

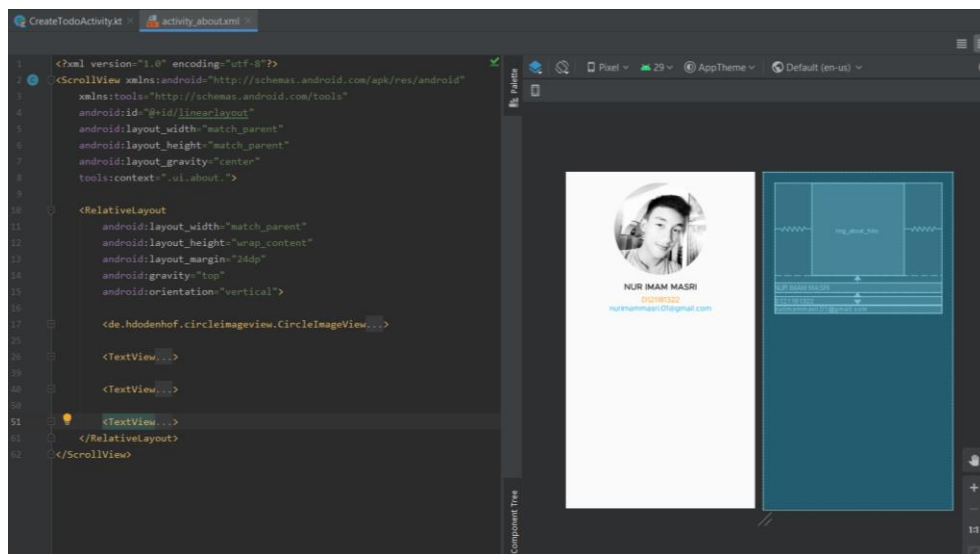
MEMBUAT ACTIVITY TAMBAHAN

- About Activity

```
AboutToDoActivity.kt
1 package com.imam.ido_simpletodolist.ui.about
2
3 import ...
4
5
6
7
8 class AboutToDoActivity : AppCompatActivity() {
9     override fun onCreate(savedInstanceState: Bundle?) {
10         super.onCreate(savedInstanceState)
11         setContentView(R.layout.activity_about)
12
13         supportActionBar?.setDisplayHomeAsUpEnabled(true)
14         supportActionBar?.setDisplayHomeAsUpEnabled(true)
15         supportActionBar?.setHomeAsUpIndicator(R.drawable.ic_arrow_back)
16     }
17
18     override fun onOptionsItemSelected(item: MenuItem?): Boolean {
19         when (item?.itemId) {
20             android.R.id.home -> {
21                 finish()
22             }
23         }
24         return true
25     }
26 }
27
```

Untuk menampilkan bagian Menu About yang berisi profile

- Tampilan View untuk Menu About



- **CreateTodoActivity**

Activity ini dibutuhkan untuk bagian behaviour pada bagian pengisian form kostumisasi Item Todo List, Seperti Empty set dan Due Expired. Dan juga Menu ini juga sekaligus dibuat untuk Bagian Edit dengan prepopulate data (Mengeset data yang ada untuk di edit)

```
1 package com.imam.ido_simpletodolist.ui.create
2
3 import ...
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29 class CreateTodoActivity : AppCompatActivity() {
30
31     private var todo: Todo? = null
32     private lateinit var alarmReceiver: AlarmReceiver
33
34     override fun onCreate(savedInstanceState: Bundle?) {
35         super.onCreate(savedInstanceState)
36         setContentView(R.layout.activity_create_todo)
37
38         alarmReceiver = AlarmReceiver()
39
40         result_duedate.visibility = View.INVISIBLE
41         //Prepopulate existing title and content from intent
42         val intent = intent
43         if (intent != null && intent.hasExtra(Constants.INTENT_OBJECT)) {
44             val todo: Todo? = intent.getParcelableExtra(Constants.INTENT_OBJECT)
45             this.todo = todo
46             prePopulateData(todo)
47         }
48
49         val titleBar =
50             if (todo != null) "View/Edit Todo" else "Create Todo"
51
52         btnDate.setOnClickListener { it.view!
53             handleDateButton()
54         }
55     }
56 }
```

Untuk bagian Eddit, dibutuhkan untuk mengambil data dri item maka prepopulateData digunakan untuk mengeset data awal pada form

```
private fun prePopulateData(todo: Todo?) {
    tv_title.setText(todo?.title)
    tv_content.setText(todo?.content)
    edt_date.setText(todo?.dueAt?.let { toDate(it) })
    edt_time.setText(todo?.dueAt?.let { toTime(it) })
    if (todo != null) {
        switchDueHour.isChecked = todo.check_alarm_hour
    }
    if (todo?.dueAt != null) {
        result_duedate.visibility = View.VISIBLE
        result_date.text = toDate(todo.dueAt)
        result_time.text = toTime(todo.dueAt)
    }
}
```

Untuk mendeklarasikan tombol **Save dan Back** pada menu di **activity_create_todo.xml**

```
    override fun onCreateOptionsMenu(menu: Menu?): Boolean {
        val menuInflate = menuInflater
        menuInflate.inflate(R.menu.menu_save, menu)
        return true
    }

    override fun onOptionsItemSelected(item: MenuItem?): Boolean {
        when (item?.itemId) {
            R.id.save_todo -> {
                saveTodo()
            }
            android.R.id.home -> {
                showAlertDialog()
            }
        }
        return true
    }
}
```

Digunakan untuk memvalidasi *Empty dan Date yang kadaluarsa* dan memberi **Message Error** pada field.

```
private fun validateFields(): Boolean {
    if (tv_title.text.isEmpty()) {
        til_todo_title.error = "Please enter title"
        tv_title.requestFocus()
        return false
    }
    if (tv_content.text.isEmpty()) {
        til_todo_content.error = "Please enter content"
        tv_content.requestFocus()
        return false
    }
    if (edt_date.text.isEmpty()) {
        til_todo_duedate.error = "Please enter due date"
        edt_date.requestFocus()
        return false
    }
    if (edt_time.text.isEmpty()) {
        til_todo_duetime.error = "Please enter due time"
        edt_time.requestFocus()
        return false
    }
}

val format = SimpleDateFormat( pattern: "EEE, dd/MM/yyyy HH:mm", Locale.getDefault())
val dueDate: Date? = format.parse( source: edt_date.text.toString() + " " + edt_time.text.toString())
val calNow = Calendar.getInstance()
val calendar = calNow.clone() as Calendar
calendar.time = dueDate
if (calendar <= calNow) {
    // Today Set time passed, count to tomorrow
    calendar.add(Calendar.DATE, amount: 1);
    edt_date.error = "Date Passed, Enter Again"
    edt_time.error = "Date Passed, Enter Again"
    til_todo_duedate.error = "Date Passed, Enter Again"
}

CreateToDoActivity > onCreate() > btnTime.setOnClickListener(...)
```


Untuk Menampilkan Fungsi **DatePickerDialog** dan **TimePickerDialog**

```
private fun handleDateButton() {
    val calendar = Calendar.getInstance()
    val year: Int = calendar.get(Calendar.YEAR)
    val month: Int = calendar.get(Calendar.MONTH)
    val date: Int = calendar.get(Calendar.DATE)

    val datePickerDialog = DatePickerDialog(
        context: this,
        OnDateSetListener { _, Year, Month, Date ->
            calendar.set(Calendar.YEAR, Year)
            calendar.set(Calendar.MONTH, Month)
            calendar.set(Calendar.DATE, Date)
            val dateString = toDate(calendar.time)
            edt_date.setText(dateString)
        }, year, month, date
    )
    datePickerDialog.show()
}

private fun handleTimeButton() {
    val calendar = Calendar.getInstance()
    val hour: Int = calendar.get(Calendar.HOUR_OF_DAY)
    val minute: Int = calendar.get(Calendar.MINUTE)

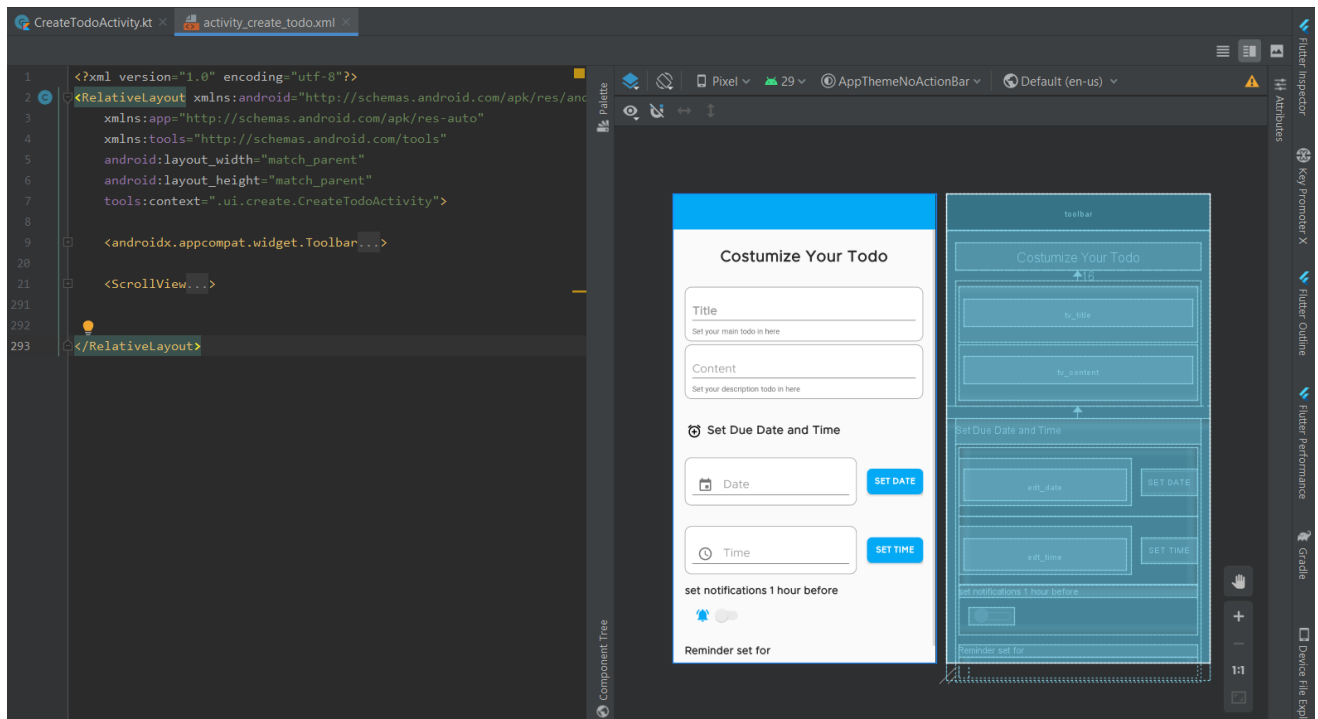
    val timePickerDialog = TimePickerDialog(
        context: this,
        OnTimeSetListener { _, Hour, Minute ->
            calendar.set(Calendar.HOUR_OF_DAY, Hour)
            calendar.set(Calendar.MINUTE, Minute)
            val timeString = toTime(calendar.time)
            edt_time.setText(timeString)
        }, hour, minute, is24HourView: true
    )
}
```

Karena dibutuhkan fungsi untuk Konversi Date ke String maka dibuat fungsi tambahan **toDate()** dan **toTime()**

```
/**
 * Date Format
 */
private fun toDate(date: Date): String {
    val outputFormatter: DateFormat = SimpleDateFormat( pattern: "EEE, dd/MM/yyyy", Locale.getDefault())
    return outputFormatter.format(date)
}

private fun toTime(date: Date): String {
    val outputFormatter: DateFormat = SimpleDateFormat( pattern: "HH:mm", Locale.getDefault())
    return outputFormatter.format(date)
}
}
```

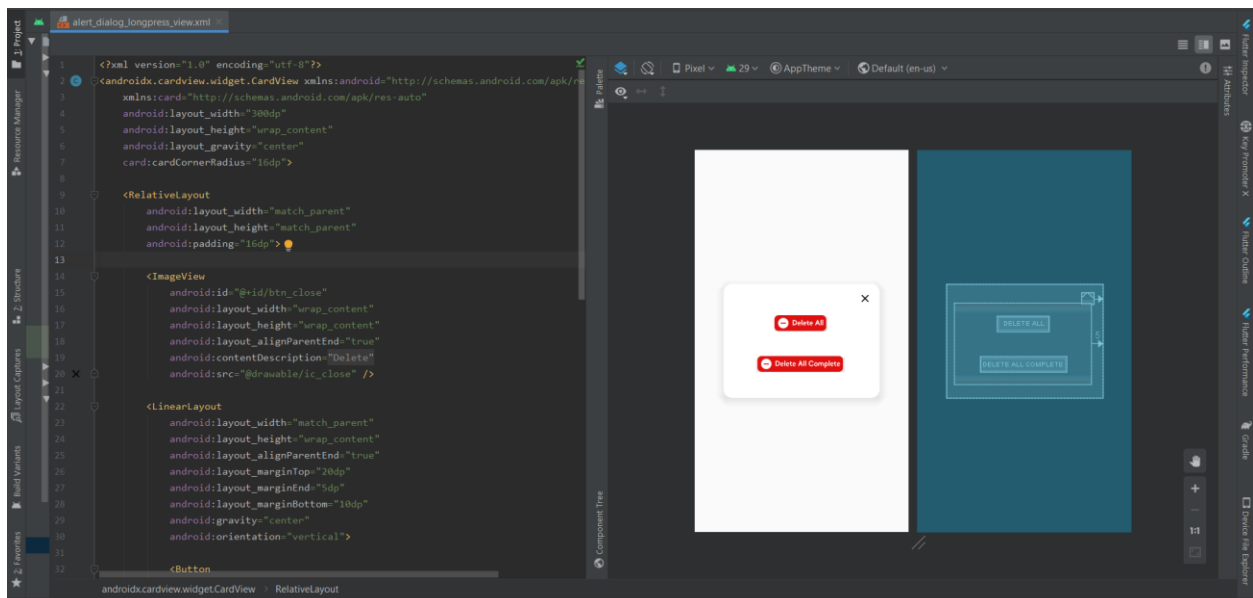
- **Tampilan View untuk Menu AddToDoList**



MEMBUAT VIEW TAMBAHAN

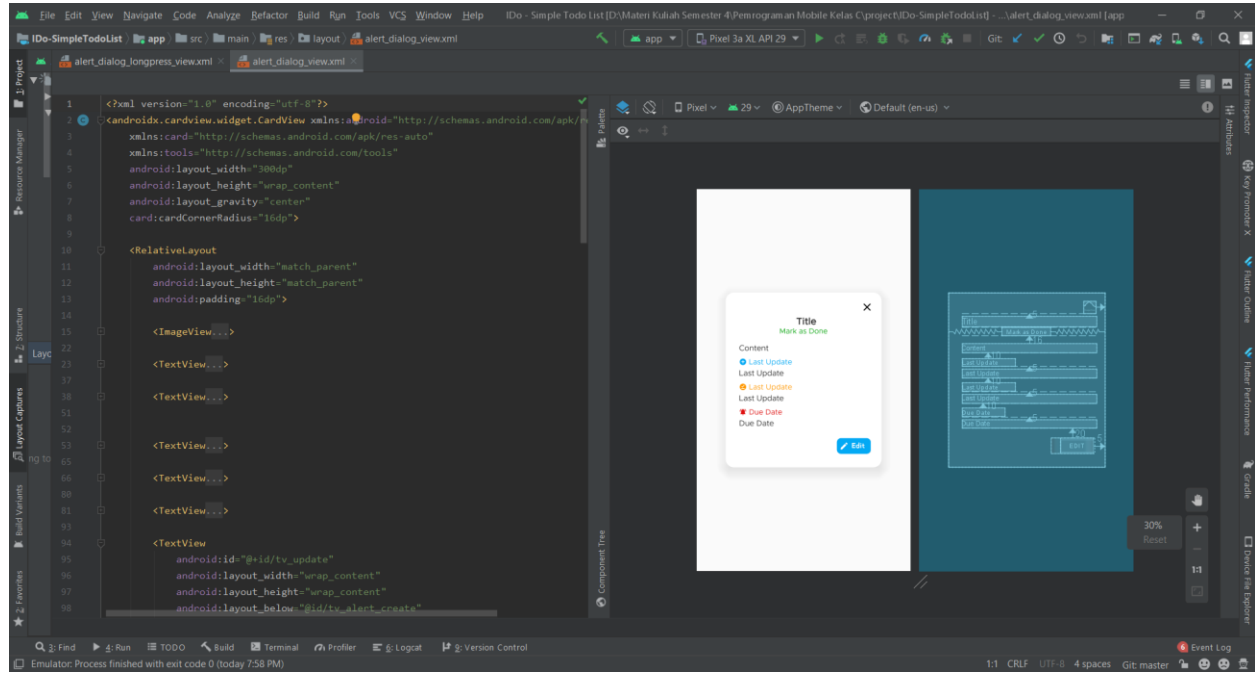
- **alert_dialog_longpress_view.xml**

Digunakan untuk bagian tampilan ketika item di klik lama, yang mana fitur pada view ini digunakan untuk fungsi delete yakni DeleteAll dan Delete All Complete



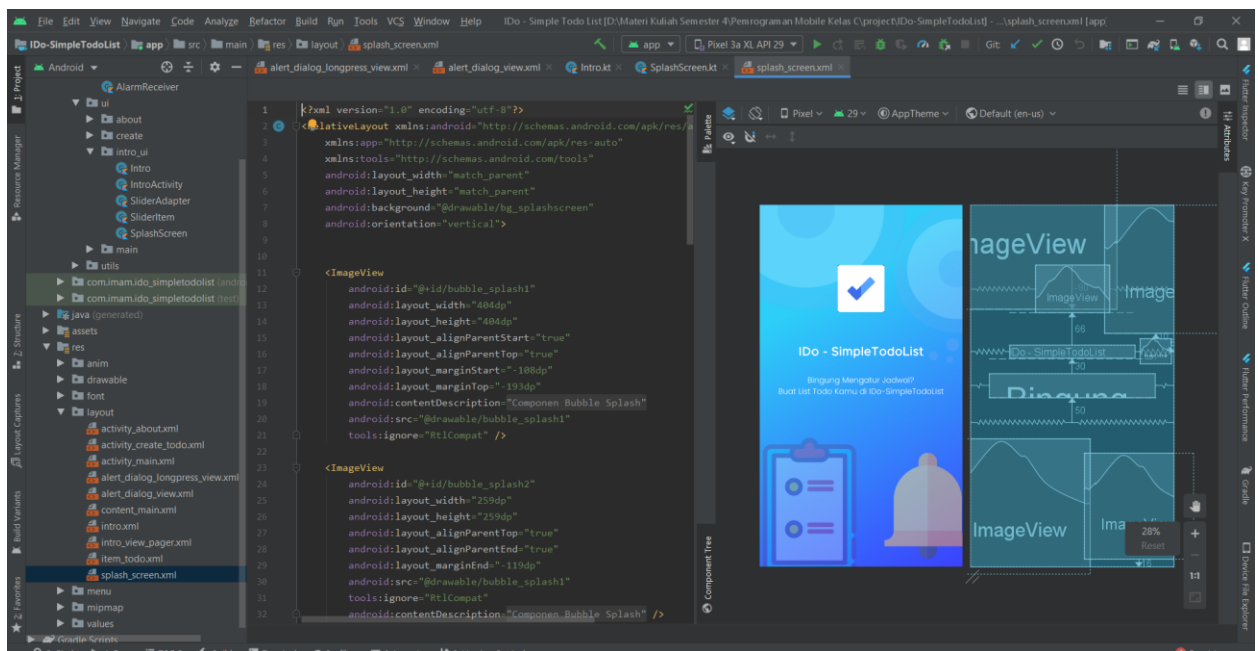
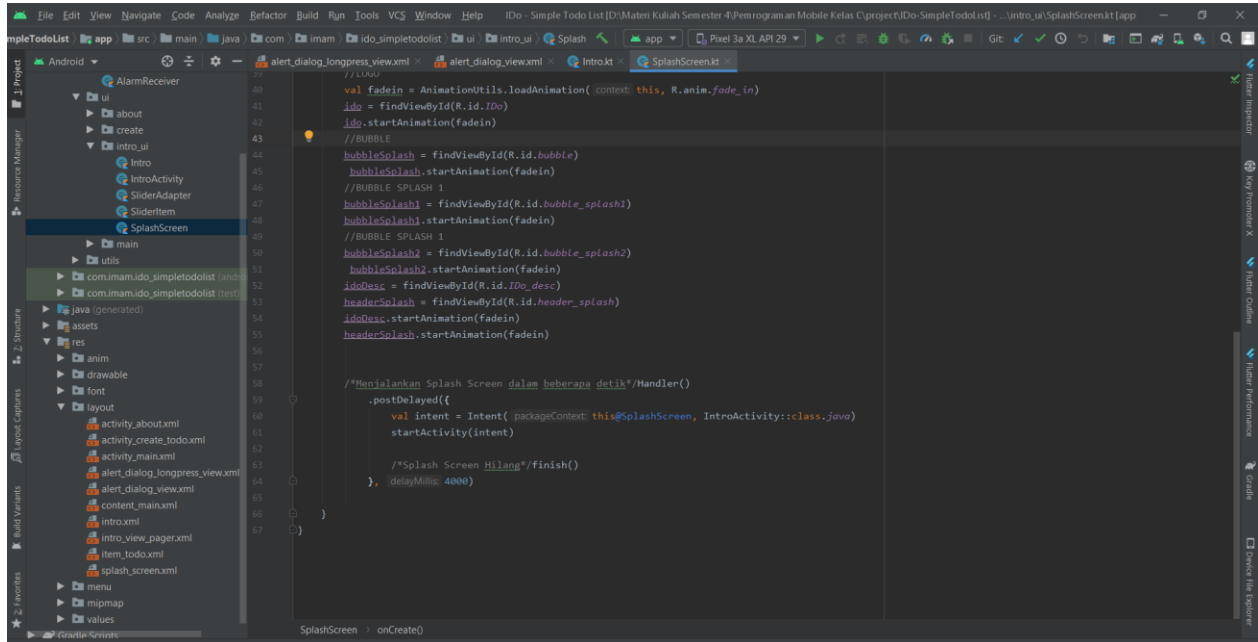
- **alert_dialog_view.xml**

Digunakan untuk bagian View Deskripsi lengkap item dan tombol menuju bagian edit

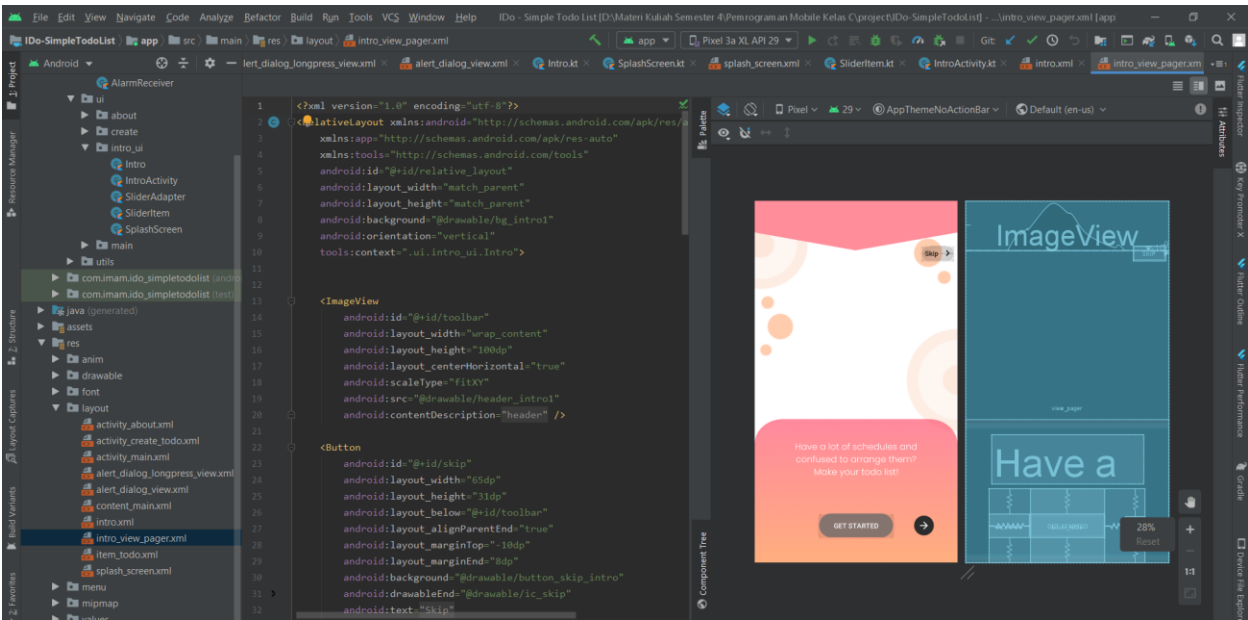


MEMBUAT INTRO

- Splash Screen



- **Bagian Intro Slider**



SETTING ALARAM DAN NOTIFICATION

- Deklarasi Permission dan Receiver di AndroidManifest.xml

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

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

- Deklarasi AlarmReceiver untuk BroadcastReceiver

Ada beberapa fungsi yang dideklarasikan untuk bagian setting notifikasi

1. setOneTimeAlarm()

Diguunakan untuk mendeklarasikan **AlarmManager** untuk set Alarm type pas Due Date (Saat Date Berakhir)

Didalam nya akan di deklarasikan Calendar yang berfungsi untuk mengambil bagian data dueAt yang nantinya diset k bagian Notification Builder

```
fun setOneTimeAlarm(
    context: Context,
    type: String,
    date: Date,
    title: String,
    message: String
) {
    val alarmManager = context.getSystemService(Context.ALARM_SERVICE) as AlarmManager
    val intent = Intent(context, AlarmReceiver::class.java)
    intent.putExtra(EXTRA_MESSAGE, message)
    intent.putExtra(EXTRA_TITLE, title)
    intent.putExtra(EXTRA_TYPE, type)

    val dateString = toDate(date)
    val calendar = Calendar.getInstance()
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
        val millisSinceEpoch =
            LocalDateTime.parse(dateString, DateTimeFormatter.ofPattern("dd/MM/yyyy HH:mm:ss"))
                .atZone(ZoneId.systemDefault())
                .toInstant()
                .toEpochMilli()
        calendar.timeInMillis = millisSinceEpoch
    } else {
        calendar.time = date
    }

    val pendingIntent = PendingIntent.getBroadcast(
        context,
        ID_ONETIME, intent, flags: 0
    )
    alarmManager.set(AlarmManager.RTC_WAKEUP, calendar.timeInMillis, pendingIntent)
```

2. setBeforeHourAlarm()

Digunakan untuk mendeklarasikan **AlarmManager** untuk set Alarm type pas Before Hour (Saat Date 1 jam sebelum berakhir)

Didalam nya akan di deklarasikan Calendar yang berfungsi untuk mengambil bagian data dueAt yang nantinya diset k bagian Notification Builder

```
fun setBeforeHourAlarm(
    context: Context,
    type: String,
    date: Date,
    title: String,
    message: String
) {

    val alarmManager = context.getSystemService(Context.ALARM_SERVICE) as AlarmManager
    val intent = Intent(context, AlarmReceiver::class.java)
    intent.putExtra(EXTRA_MESSAGE, message)
    intent.putExtra(EXTRA_TITLE, title)
    intent.putExtra(EXTRA_TYPE, type)

    val dateString = toDate(date)
    val calendar = Calendar.getInstance()
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
        val millisSinceEpoch =
            LocalDateTime.parse(dateString, DateTimeFormatter.ofPattern("dd/MM/yyyy HH:mm:ss"))
                .atZone(ZoneId.systemDefault())
                .toInstant()
                .toEpochMilli()
        calendar.timeInMillis = millisSinceEpoch
    } else {
        calendar.time = date
    }

    val pendingIntent = PendingIntent.getBroadcast(
        context,
        ID_BEFORE_HOUR, intent, flags: 0
    )
    alarmManager.set(AlarmManager.RTC_WAKEUP, calendar.timeInMillis, pendingIntent)
```

3. cancelAlarm dan finishedAlarm

Ketika tombol edit atau delete “x” di klik maka akan meng – cancel alarm. Maka digunakan fungsi ini untuk menghentikan settingan alarm notifikasi. Pada bagian alarmManager.cancel

Sedangkan untuk ketika item di set “Mark as Done” maka alarm akan di hentikan dengan alasan Todo List sudah Complete

```

fun cancelAlarm(context: Context, type: String) {
    val alarmManager = context.getSystemService(Context.ALARM_SERVICE) as AlarmManager
    val intent = Intent(context, AlarmReceiver::class.java)
    val requestCode =
        if (type.equals(TYPE_ONE_TIME, ignoreCase = true)) ID_ONETIME else ID_BEFORE_HOUR
    val pendingIntent = PendingIntent.getBroadcast(context, requestCode, intent, flags: 0)
    pendingIntent.cancel()

    alarmManager.cancel(pendingIntent)

    Toast.makeText(context, text: "Alarm has been canceled", Toast.LENGTH_SHORT).show()
}

fun finishAlarm(context: Context, type: String) {
    val alarmManager = context.getSystemService(Context.ALARM_SERVICE) as AlarmManager
    val intent = Intent(context, AlarmReceiver::class.java)
    val requestCode =
        if (type.equals(TYPE_ONE_TIME, ignoreCase = true)) ID_ONETIME else ID_BEFORE_HOUR
    val pendingIntent = PendingIntent.getBroadcast(context, requestCode, intent, flags: 0)
    pendingIntent.cancel()

    alarmManager.cancel(pendingIntent)

    Toast.makeText(context, text: "Alarm is turn off (Todo Complete)", Toast.LENGTH_SHORT).show()
}

```

4. showAlarmNotification

Bagian untuk menampilkan Display Notifikasi ke bagian ActionBar

```

// Gunakan metode ini untuk menampilkan notifikasi
private fun showAlarmNotification(
    context: Context,
    title: String,
    message: String,
    notifId: Int
) {

    val CHANNEL_ID = "Channel_1"
    val CHANNEL_NAME = "AlarmManager channel"

    val notificationManagerCompat =
        context.getSystemService(Context.NOTIFICATION_SERVICE) as NotificationManager
    val alarmSound = RingtoneManager.getDefaultUri(RingtoneManager.TYPE_NOTIFICATION)
    val builder = NotificationCompat.Builder(context, CHANNEL_ID)
        .setSmallIcon(R.mipmap.ic_launcher)
        .setContentTitle(title)
        .setContentText(message)
        .setColor(ContextCompat.getColor(context, android.R.color.transparent))
        .setVibrate(LongArrayOf(1000, 1000, 1000, 1000, 1000))
        .setSound(alarmSound)

    /*
    Untuk android Oreo ke atas perlu menambahkan notification channel
    Materi ini akan dibahas lebih lanjut di modul extended
    */
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {

        /* Create or update. */
        val channel = NotificationChannel(
            CHANNEL_ID,

```