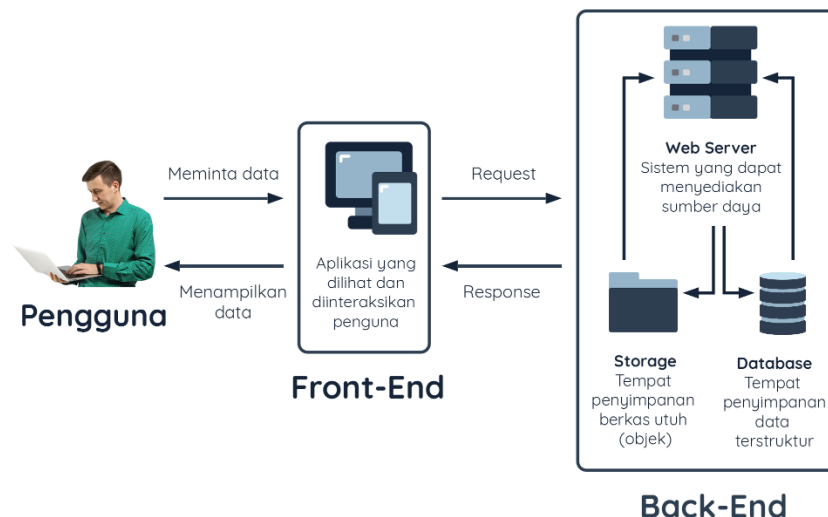


## Room Database Bagian 1 (Create, Read)

Database atau basis data adalah kumpulan data yang dikelola sedemikian rupa berdasarkan ketentuan tertentu yang saling berhubungan sehingga mudah dalam pengelolaannya. Melalui pengelolaan tersebut pengguna dapat memperoleh kemudahan dalam mencari informasi, menyimpan informasi dan membuang informasi. Gampangnya sih, **database adalah sistem yang berfungsi sebagai mengumpulkan file, tabel, atau arsip yang terhubung dan disimpan dalam berbagai media elektronik.**

Dalam sebuah aplikasi, biasanya database akan dijadikan sebagai tempat penyimpanan, di mana data yang tersimpan di dalam database akan ditampilkan di dalam aplikasi. Berikut ini adalah contoh akses database yang tersimpan di dalam sebuah server.



Gambar 1. Ilustrasi Penggunaan Database Server dalam Aplikasi

Berdasarkan cara mengaksesnya, database terbagi menjadi 2 bagian, yang pertama yaitu database online, di mana kita membutuhkan server dan akses internet untuk mengakses database ini. Contoh dari database online adalah data akun sosial media milik kita.

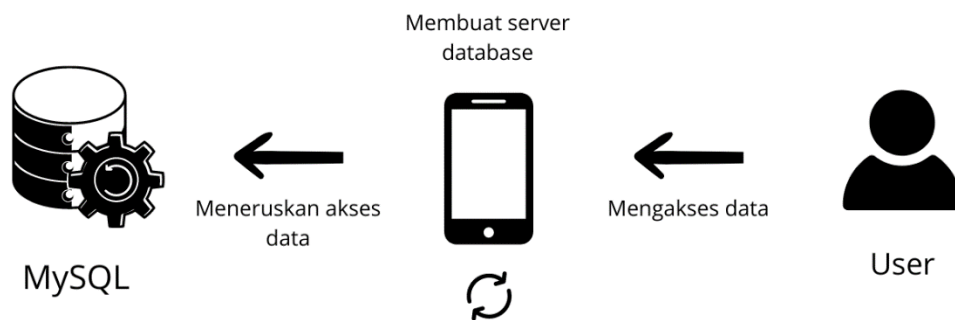
Yang kedua adalah database offline, di mana database tersebut jalan berjalan secara langsung di device kita, sehingga untuk mengaksesnya kita tidak memerlukan internet. Contoh dari database offline adalah history chat dari aplikasi whatsapp.

Berdasarkan jenis dan fungsi, database terbagi menjadi beberapa bagian, yaitu operational database, database warehouse, distributed database, relational database, dan end-user database.

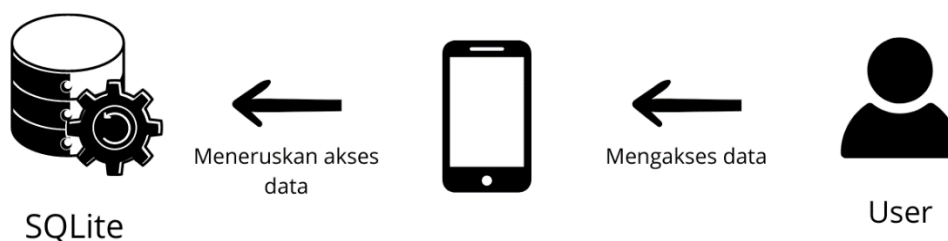
Mungkin kalian cukup familiar dengan relational database, seperti misalnya MySQL, PostgreSQL, dll. Namun kali ini kita akan mempelajari tentang end-user database.

End user database adalah jenis database yang berjalan dan tersimpan di perangkat pribadi milik pengguna (berbanding terbalik dengan database seperti MySQL yang dikenal di simpan di server). Contoh dari penggunaan end user database adalah aplikasi yang diakses secara offline seperti aplikasi catatan, atau salah satu penggunaan yang cukup populer adalah history chat whatsapp.

SQLite adalah salah satu database yang sering digunakan ketika developer ingin menerapkan konsep **End user database**. Alasannya adalah karena SQLite umumnya sudah built-in atau terinstall secara default di dalam sebuah aplikasi, sehingga pengguna tidak perlu untuk melakukan instalasi SQLite secara terpisah karena databasenya sudah tersedia secara langsung di dalam aplikasi yang ingin digunakan. FYI, SQLite dibuat menggunakan Bahasa C, penggunaannya sama seperti SQL database lainnya, namun yang membedakan SQLite dengan SQL database lainnya adalah ukurannya lebih kecil dan lebih cepat.



Gambar 2. Alur mengakses data menggunakan database MySQL



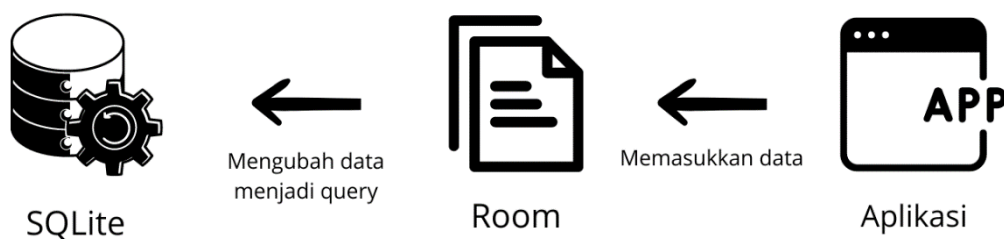
Gambar 3. Alur mengakses data menggunakan database SQLite

Room adalah library buatan Google, Room berjalan di atas SQLite, yang mana Room akan mempermudah kita dalam menggunakan SQLite karena Room memiliki banyak manfaat, salah satunya adalah membuat SQLite yang awalnya Query Oriented menjadi ORM (Object Relational Mapping). Apa itu ORM?

ORM (Object Relational Mapping) adalah konsep mengubah sebuah data menjadi sebuah query. Jadi nantinya proses mengelola data dalam database tidak memerlukan query, sehingga mempermudah bagi user yang menggunakannya.

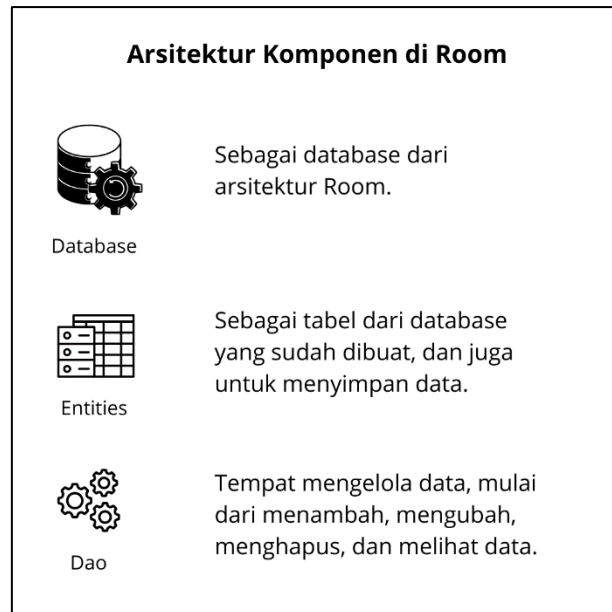


Gambar 4. Proses mengakses database tanpa ORM

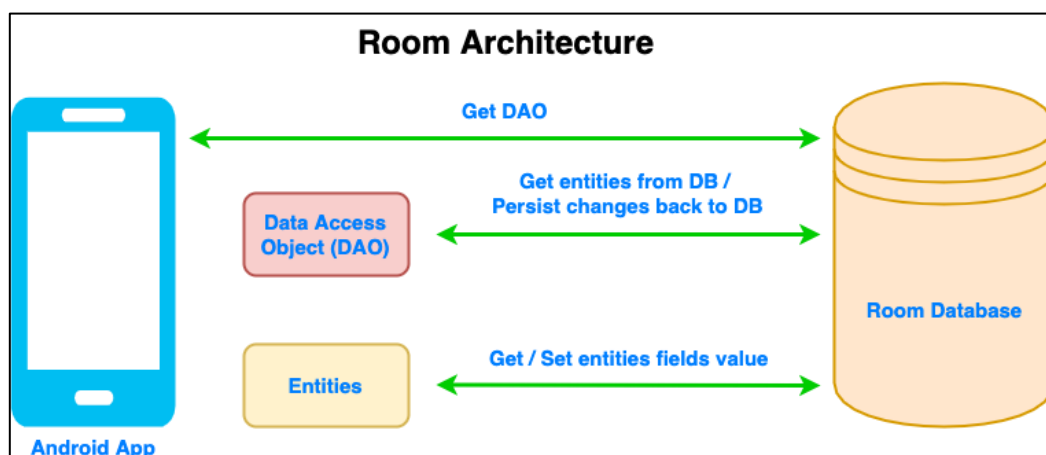


Gambar 5. Proses mengakses database dengan ORM

Manfaat lain Room adalah mengurangi kode boilerplate dan mempermudah migrasi database database.



Gambar 6. Arsitektur Komponen Room



Gambar 7. Alur pengelolaan data di Room

```

@Dao
interface AppDao {

    @Insert(onConflict = OnConflictStrategy.IGNORE)
    fun insertPlayer(player: PlayerDatabase)

    @Update
    fun updatePlayer(player: PlayerDatabase)

    @Delete
    fun deletePlayer(player: PlayerDatabase)

    @Query("SELECT * from playerdatabase ORDER BY player_name ASC")
    fun getAllPlayer(): LiveData<List<PlayerDatabase>>
}
  
```

Gambar 8. Komponen akses data Room

Ada empat komponen akses data di dalam Room, pada pertemuan kali ini, kita akan membahas komponen Insert dan Update.

## **Insert**

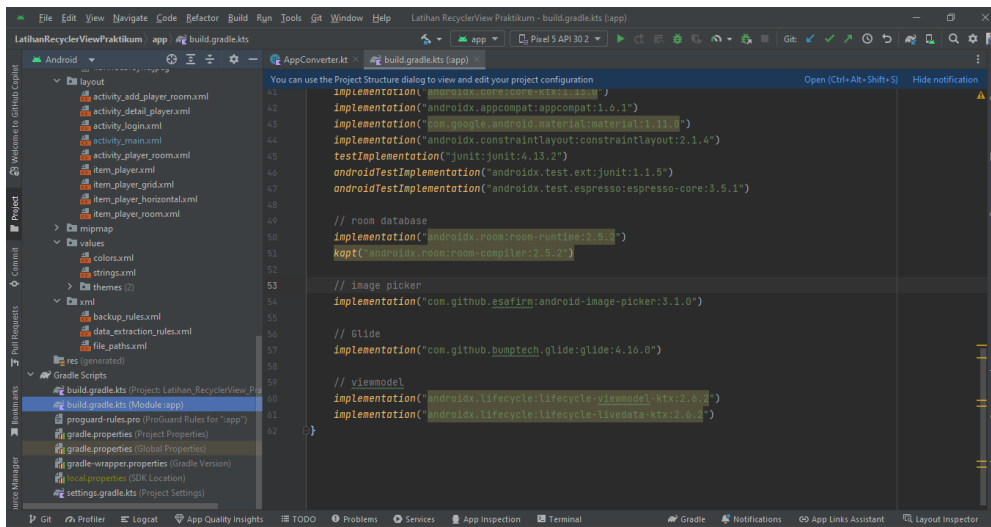
Komponen Insert digunakan untuk memasukkan data ke dalam database. Karena Room mengadopsi konsep ORM, proses memasukkan data dilakukan secara otomatis sehingga kita cukup menuliskan data apa saja yang akan dimasukkan di dalam parameter, nantinya kotlin akan membuat query secara otomatis berdasarkan data yang kita tulis di dalam parameter.

## **Query**

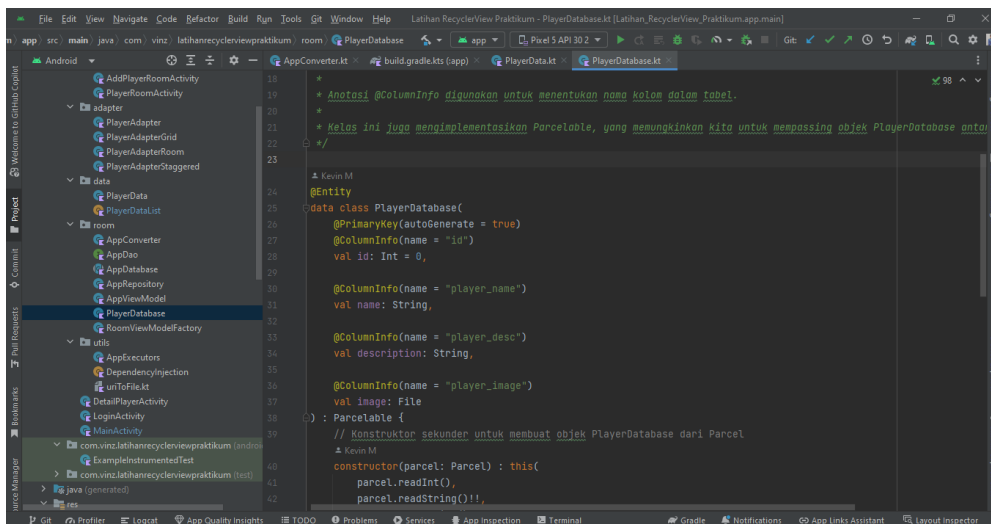
Berbeda dengan konsep ORM yang akan membuat query secara otomatis. Komponen query digunakan untuk membuat query secara manual, sehingga kita perlu menuliskan query yang spesifik berdasarkan kebutuhan kita. Biasanya anotasi query digunakan ketika kita berhubungan dengan melihat data (SELECT), sehingga proses SELECT data bisa dilakukan dengan berbagai macam variasi yang ada (Contoh: melihat data keseluruhan, melihat data berdasarkan nama, melihat data berdasarkan id, dan lain-lain). Namun biarpun begitu, anotasi query bisa digunakan untuk hal lain seperti misalnya Insert Data, Update data, dan masih banyak lagi.

## **Langkah-langkah membuat Room Database**

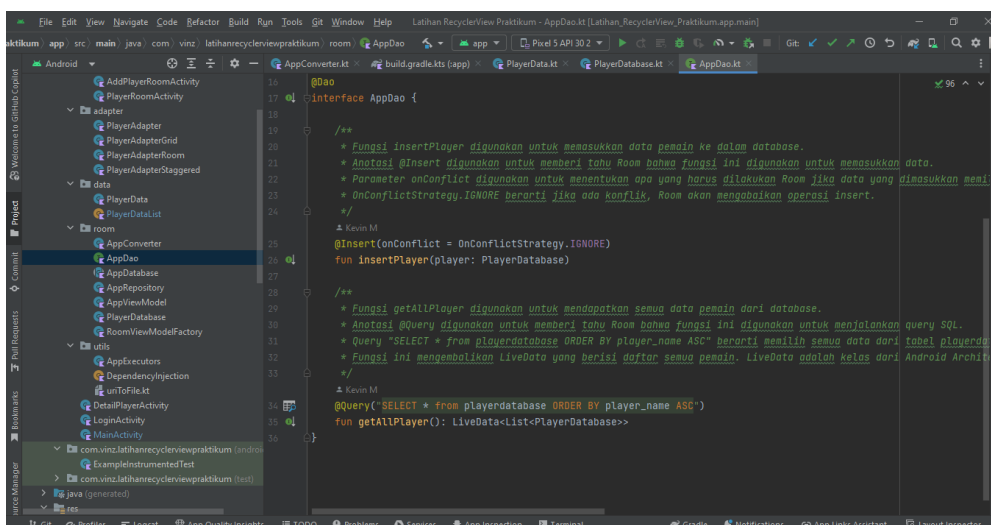
1. Tambahkan library Room Database, Image Picker, Glide dan ViewModel ke dalam build.gradle.kts (module app)



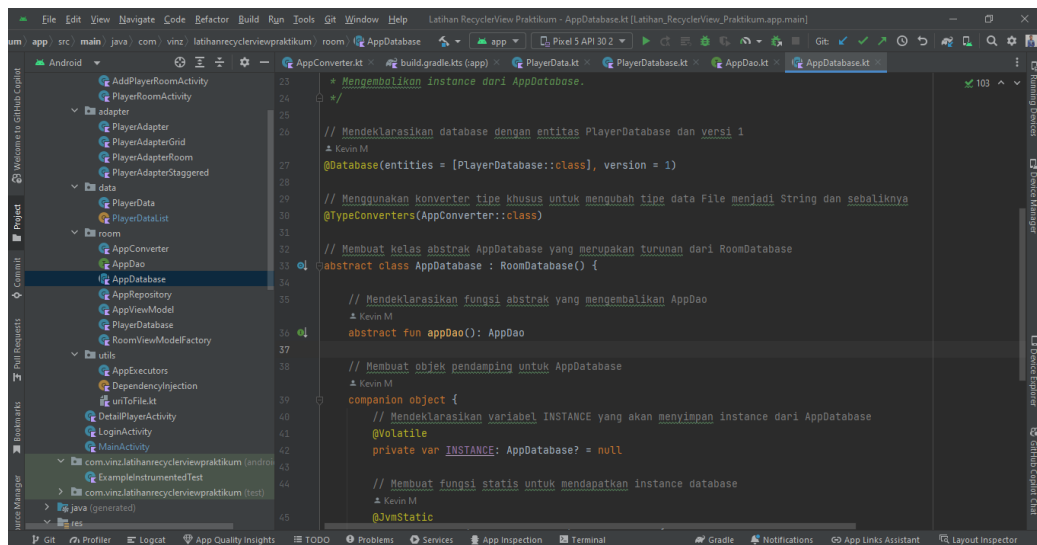
## 2. Buat Entity/Data Class baru sebagai tabel dari database yang kita buat



## 3. Buat Interface baru yang Bernama DAO (Data Access Object) sebagai Alat untuk Mengakses Entity/Data Class yang sudah kita buat tadi (INSERT, UPDATE, DELETE dan GET)

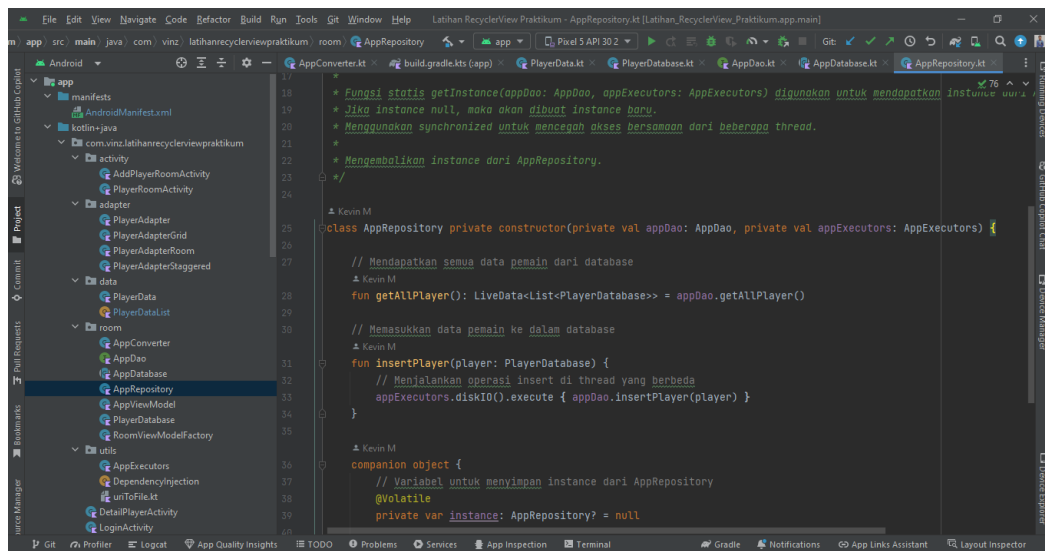


4. Buat abstract class baru sebagai database dari aplikasi yang kita buat



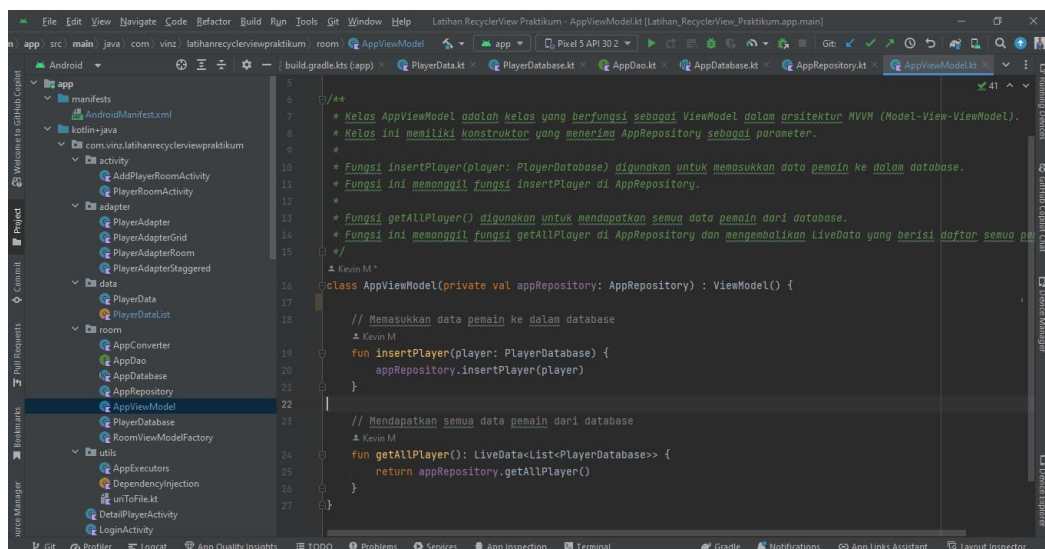
```
1 // Mengembalikan instance dari AppDatabase.
2
3 // Mendefinisikan database dengan entitas PlayerDatabase dan versi 1
4 @Database(entities = [PlayerDatabase::class], version = 1)
5
6 // Menggunakan konverter tipe khusus untuk mengubah tipe data File menjadi String dan sebaliknya
7 @TypeConverters(AppConverter::class)
8
9 // Membuat kelas abstrak AppDatabase yang merupakan turunan dari RoomDatabase
10 abstract class AppDatabase : RoomDatabase() {
11
12     // Mendefinisikan fungsi abstrak yang mengembalikan AppDao
13     abstract fun appDao(): AppDao
14
15     // Membuat objek pendamping untuk AppDatabase
16     companion object {
17         // Mendefinisikan variabel INSTANCE yang akan menyimpan instance dari AppDatabase
18         @Volatile
19         private var INSTANCE: AppDatabase? = null
20
21         // Membuat fungsi statis untuk mendapatkan instance database
22         @JvmStatic
23         fun getInstance(): AppDatabase {
24             // ... (code for getting instance) ...
25         }
26     }
27 }
```

5. Buat class baru sebagai Repository untuk mengakses DAO yang sudah kita buat



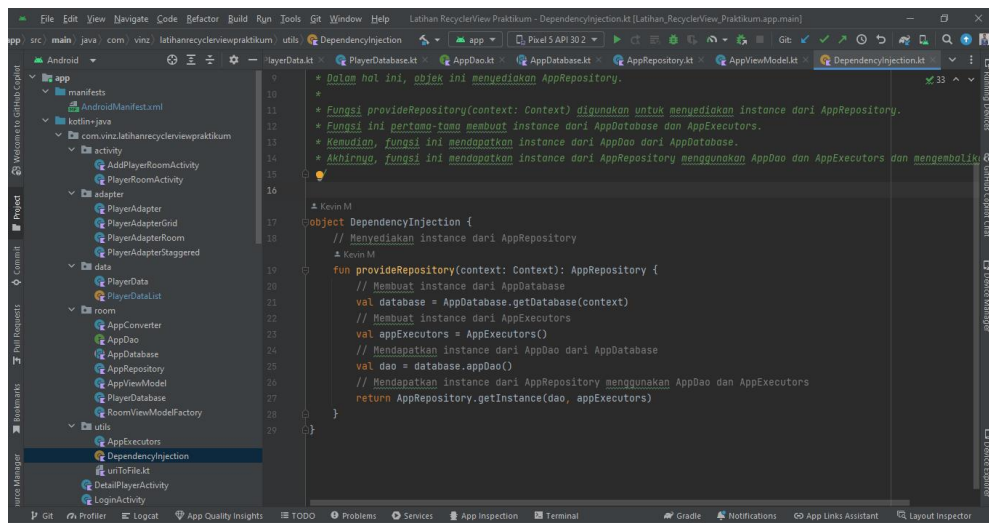
```
1 // Fungsi statis getInstance(appDao: AppDao, appExecutors: AppExecutors) digunakan untuk mendapatkan instance dari AppRepository.
2 // Jika instance null, maka akan dibuat instance baru.
3 // Menggunakan synchronized untuk mencegah akses bersamaan dari beberapa thread.
4 // Mengembalikan instance dari AppRepository.
5
6 class AppRepository private constructor(private val appDao: AppDao, private val appExecutors: AppExecutors) {
7
8     // Mendapatkan semua data pemain dari database
9     fun getAllPlayer(): LiveData<List<PlayerDatabase>> = appDao.getAllPlayer()
10
11     // Memasukkan data pemain ke dalam database
12     fun insertPlayer(player: PlayerDatabase) {
13         // Menjalankan operasi insert di thread yang berbeda
14         appExecutors.diskIO().execute { appDao.insertPlayer(player) }
15     }
16
17     companion object {
18         // Variabel untuk menyimpan instance dari AppRepository
19         @Volatile
20         private var INSTANCE: AppRepository? = null
21
22         fun getInstance(): AppRepository {
23             // ... (code for getting instance) ...
24         }
25     }
26 }
```

6. Buat class baru sebagai ViewModel untuk mengakses repository yang sudah kita buat

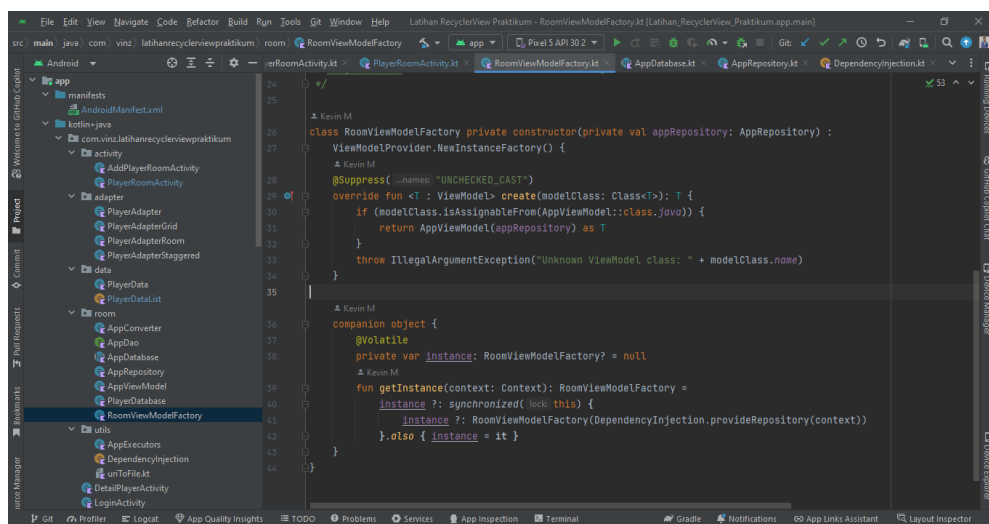


```
1 // Kelas AppViewModel adalah kelas yang berfungsi sebagai ViewModel dalam arsitektur MVVM (Model-View-ViewModel).
2 // Kelas ini memiliki konstruktor yang menerima AppRepository sebagai parameter.
3
4 // Fungsi insertPlayer(pPlayer: PlayerDatabase) digunakan untuk memasukkan data pemain ke dalam database.
5 // Fungsi ini memanggil fungsi insertPlayer di AppRepository.
6
7 // Fungsi getAllPlayer() digunakan untuk mendapatkan semua data pemain dari database.
8 // Fungsi ini memanggil fungsi getAllPlayer di AppRepository dan mengembalikan LiveData yang berisi daftar semua pemain.
9
10 class AppViewModel(private val appRepository: AppRepository) : ViewModel() {
11
12     // Memasukkan data pemain ke dalam database
13     fun insertPlayer(player: PlayerDatabase) {
14         appRepository.insertPlayer(player)
15     }
16
17     // Mendapatkan semua data pemain dari database
18     fun getAllPlayer(): LiveData<List<PlayerDatabase>> {
19         return appRepository.getAllPlayer()
20     }
21 }
```

## 7. Buat class baru sebagai Dependency Injection dari Repository yang dibuat

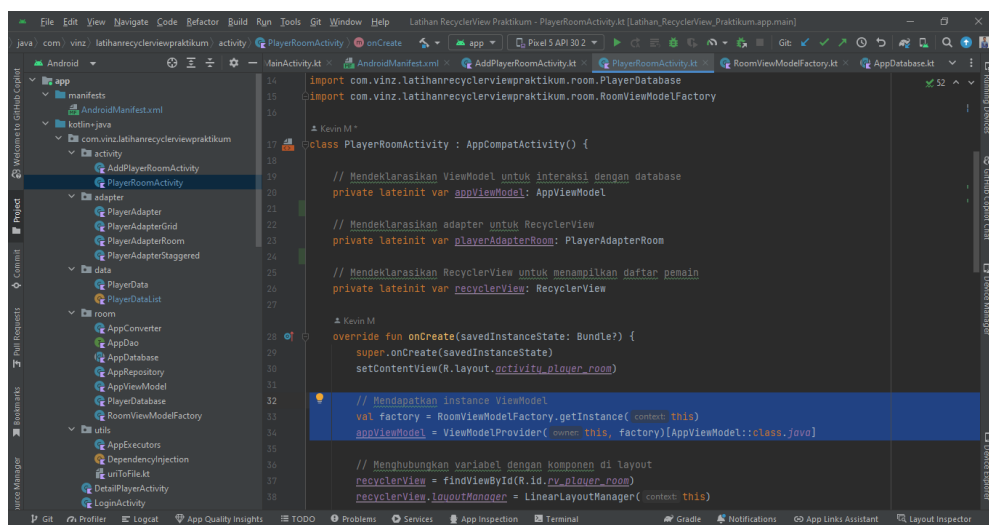


## 8. Buat class baru bernama ViewModelFactory sebagai Injector dari ViewModel



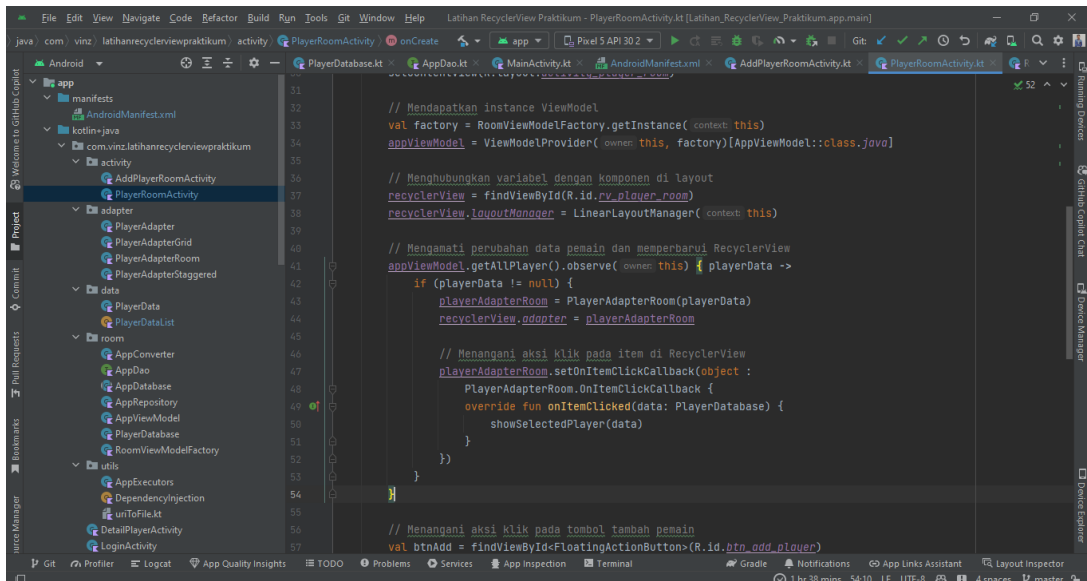
## 9. Setup di bagian RecyclerView (sama seperti di modul 3)

## 10. Panggil ViewModel beserta ViewModelFactory di dalam Activity





## 11. Jangan lupa masukkan juga data yang terdapat di dalam database ke dalam Adapter

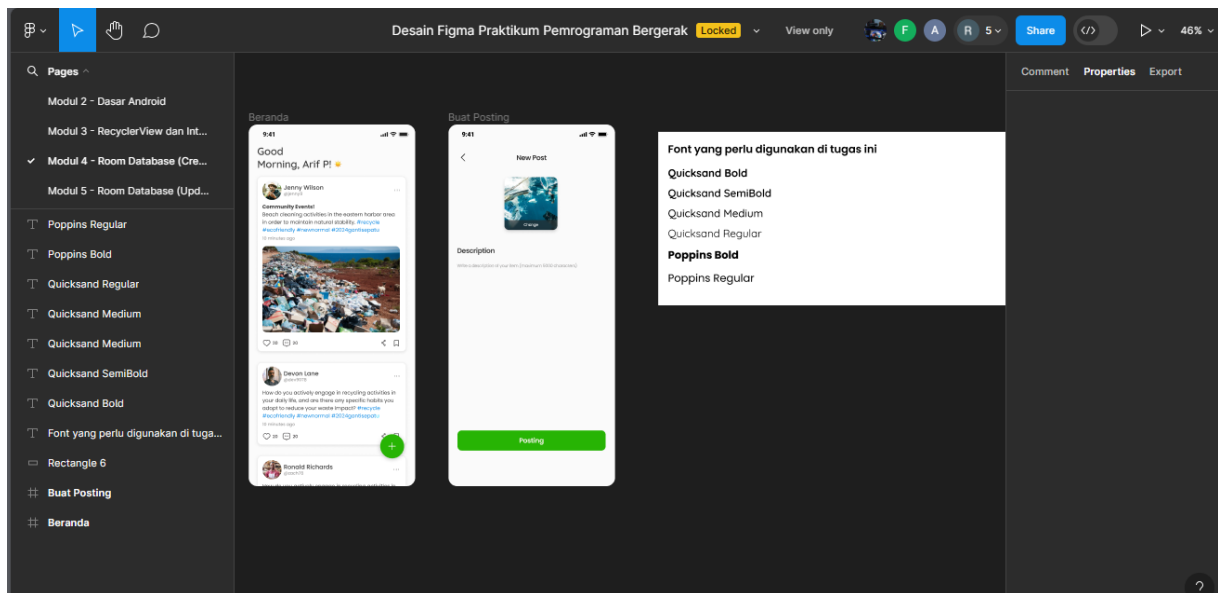


## Tugas Pendahuluan

1. Apa itu Room Database dan kapan kita harus menggunakan Room Database?
2. Bisakah kalian jelaskan korelasi antara Room Database dan RecyclerView dalam hal menampilkan data?
3. Jelaskan alur anotasi INSERT (yang terdapat di dalam DAO) ketika memasukkan data ke dalam Room Database?

## Tugas Praktikum

Buatlah aplikasi praktikum seperti berikut [ini](#) (cek Pages 4 yaitu Modul 4 – Room Database Create, Read)!



Ketentuan dalam pembuatan aplikasi:

1. Menerapkan desain yang tersedia menjadi sebuah aplikasi
2. Menerapkan Room Database dalam pembuatan aplikasi
3. Menerapkan RecyclerView untuk menampilkan data yang tersimpan di dalam Room Database

#### Alur Aplikasi:

1. Ketika user pertama kali membuka aplikasi, maka akan diarahkan ke Halaman Beranda
2. Halaman berada berisi konten yang berasal dari Room Database yang sudah ditambahkan
3. Ketika item love diklik, maka akan menambah jumlah like dari konten yang diklik
4. User bisa menambahkan konten dengan cara klik Floating Button, lalu masukkan konten yang ingin ditambahkan
5. User bisa menambahkan gambar dari HP mereka
6. Ketika tombol posting diklik, maka akan kembali ke Beranda namun dengan konten baru yang berhasil ditambahkan
7. Ketika tombol back diklik di New Post, maka akan mengembalikan user ke Halaman Beranda

#### Tips:

1. Menampilkan data di Halaman Beranda menggunakan RecyclerView yang dikombinasi menggunakan ViewModel dan Repository
2. Untuk membuat Floating Button yang bisa tetap stay di tempat ketika RecyclerView sedang discroll, bisa dengan menggunakan arsitektur seperti berikut (jadi floating button sejajar dengan NestedScrollView, sehingga floating action button tidak ikut ter-scroll karena letaknya berada di luar dari NestedScrollView)

ConstraintLayout

    NestedScrollView

        ConstraintLayout

            RecyclerView

        Floating Action Button