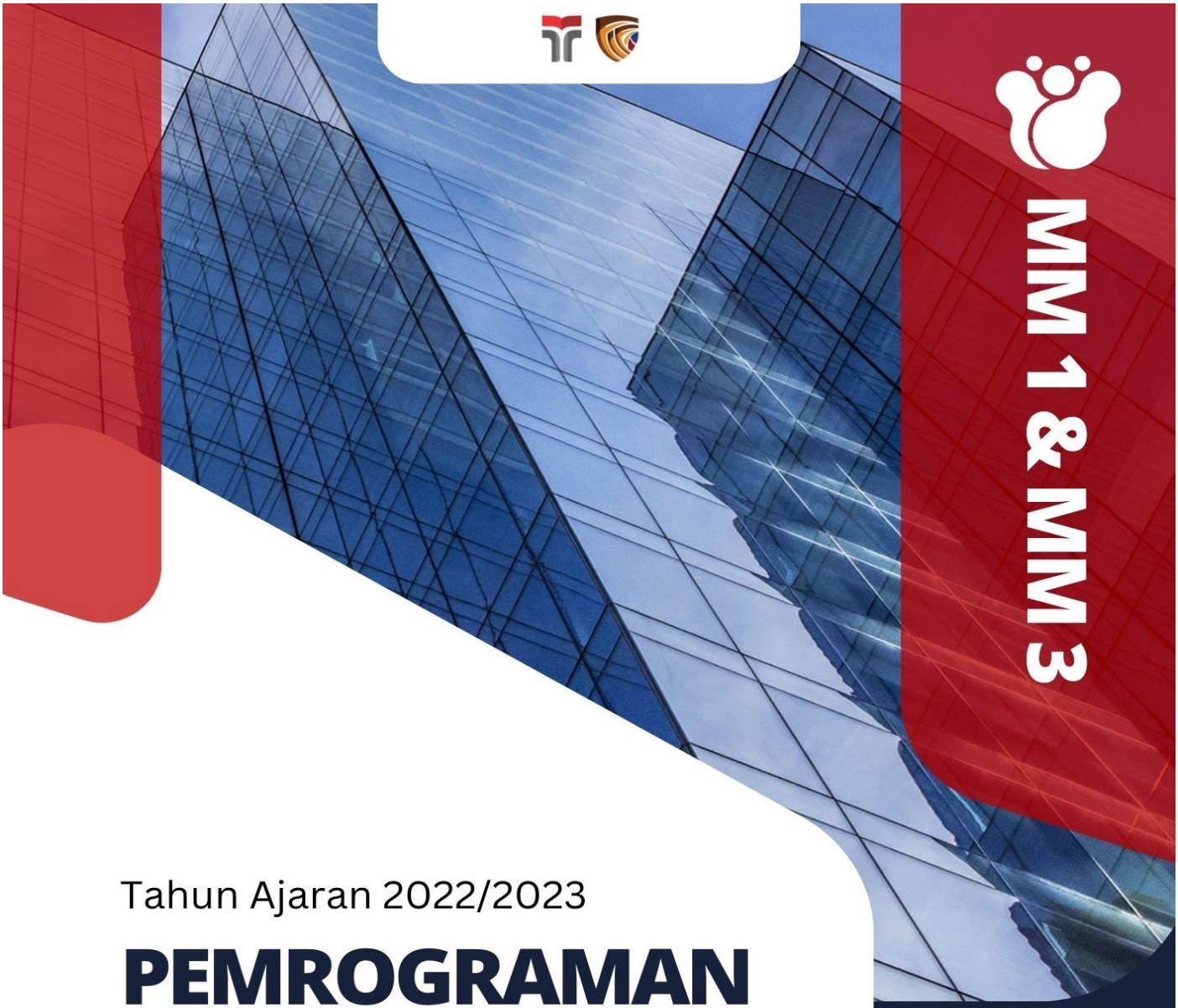


MM 1 & MM 3

Tahun Ajaran 2022/2023

PEMROGRAMAN
PERANGKAT BERGERAK
(KOTLIN)

Dosen : Muhammad Lulu Latif Usman, S.Pd., M.Han

Asprak : Bunga Laelatul Muna

MODUL 8

RETROFIT – GET API

(Link Github: bungagana/CobaGetAPI (github.com))

(Link Youtube : https://youtu.be/FYZz7NrJTUg)

Retrofit digunakan untuk mempermudah aplikasi android kita mengambil data dari api server. Dengan menggunakan Retrofit kita lebih mudah untuk melakukan request melalui HTTP. Request yang disediakan Retrofit ada lima yaitu GET, POST, PUT, DELETE, dan HEAD.

Untuk praktikum kali ini kita akan mencoba implementasi Get rest API

Rest API

https://apitani.burunghantu.id/sub/restapi-slim/public/datamahasiswa/

PRAKTIKUM

Tambahkan Dependecy Dibawah Ini :

```
implementation 'com.squareup.retrofit2:retrofit:2.9.0'
implementation 'com.squareup.retrofit2:converter-gson:2.9.0'
implementation 'androidx.recyclerview:recyclerview:1.1.0'
implementation 'androidx.cardview:cardview:1.0.0'
implementation 'com.squareup.okhttp3:logging-interceptor:4.9.3'
```

Buat BuildFeature viewBinding:

Pada gradle Android tambahkan ViewBinding untuk memudahkan kita dalam mendapatkan ID pada View dengan menambahkan syntax dibawah ini pada gradle

```
android {
    namespace 'com.example.cobaget'
    compileSdk 33

    buildFeatures{
        viewBinding true
    }
```

Setting Perizinan Untuk Akses Link Eksternal Pada AndroidManifest.xml

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

Kemudian dalam Update Android versi 8+ dengan versi api 28+ akses data API harus menggunakan format HTTPS. Sedangkan untuk tutorial Rest API masih menggunakan HTTP maka langkah berikutnya tambahkan Uses Clear Text Traffic menjadi true dengan menambahkan setting pada bagian application dengan menambahkan Source Code dibawah

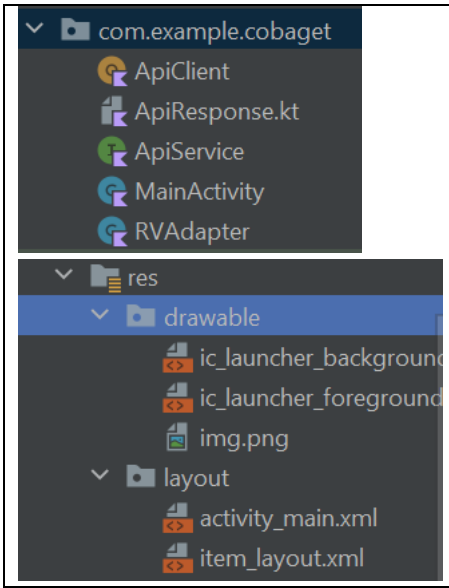
```
android:usesCleartextTraffic="true"
```

Sehingga Code Lengkap di AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">
    <uses-permission android:name="android.permission.INTERNET">
    </uses-permission>

    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/Theme.CobagetAPI"
        tools:targetApi="31"
        android:usesCleartextTraffic="true">
        <activity
            android:name=".MainActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

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

</manifest>
```

File Yang Akan Kita Buat

Akan lebih baik jika file APIClient, ApiService dipisah menjadi package baru(Packgae API). Karena nanti setiap link API yang berbeda akan memiliki file apiclient dan api service yang berbeda pula. Tapi karena dalam praktikum ini masih sederhana kita gabungkan menjadi satu package



Kita Buat Resource File yang akan menampung data list yang ada pada API (item_layout.xml)
Resource file ini akan berbentuk cardview yang nantinya akan kita panggil id nya di dalam RecyclerView yang ada pada activity_main.

```
<!-- item_mahasiswa.xml -->
<androidx.cardview.widget.CardView
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:card_view="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/cv_main"
    android:layout_margin="8dp"
    card_view:cardCornerRadius="4dp">
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal">

    <ImageView
        android:id="@+id/imageProfile"
        android:layout_width="60dp"
        android:layout_height="match_parent"
        android:src="@drawable/img"/>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:padding="16dp">

        <TextView
            android:id="@+id/nimTextView"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textSize="16sp"
```

```
            android:text="21102010"
            android:textStyle="bold"/>

        <TextView
            android:id="@+id/namaTextView"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginTop="8dp"
            android:text="Bunga"
            android:textSize="14sp"/>

        <TextView
            android:id="@+id/telpTextView"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginTop="8dp"
            android:text="08386132"
            android:textSize="14sp"/>

    </LinearLayout>
    </LinearLayout>
</androidx.cardview.widget.CardView>
```

Activity_main.xml

Dalam activity_main ini kita akan buatkan RecyclerView yang terdapat property

tools:listitem="@layout/item_layout"/>

yang diambil dari id milik item_layout

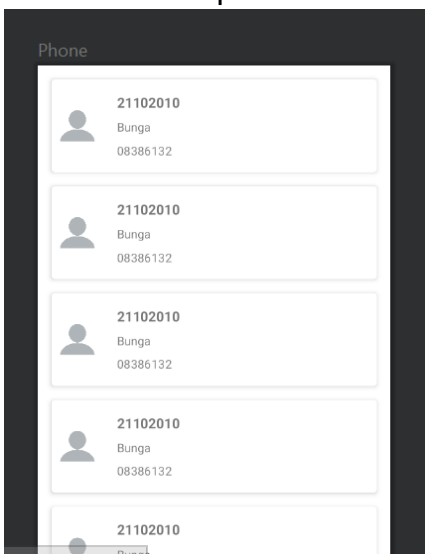
Sedangkan properti

app:layoutManager="androidx.recyclerview.widget.LinearLayoutManager"

Berfungsi untuk mengatur tampilan RecyclerView dalam keadaan Vertikal

```
<!-- activity_main.xml -->
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:orientation="vertical">

    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/rvMain"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:padding="8dp"

app:layoutManager="androidx.recyclerview.widget.LinearLayoutManager"
        tools:listitem="@layout/item_layout"/>

</LinearLayout>
```

Output



Buat file DataClass Kotlin (“ApiResponse.Kt”)
File ini sama saja berfungsi untuk menampung variable yang ada dalam link API JSON nya.

- Model ini memiliki dua properti yaitu status dan data. Properti status menyimpan status respon dari API, seperti "success" atau "error". Properti data adalah daftar Mahasiswa yang merupakan hasil dari permintaan.

- Jika kita perhatikan, data di API memiliki 2 menu atau kolom yaitu Status dan Data. Maka di file ‘ApiResponse.Kt’ nya pun harus sama. Karena jika format data model di API dan di Program berbeda maka data di API tidak akan muncul Ketika di run.

{"status":"success","data":[{"NIM":"12520211","Nama":"Anda","Telepon":"0888"

```kotlin
package com.example.cobaget

import com.google.gson.annotations.SerializedName

data class ApiResponse(
    @SerializedName("status") val status: String,
    @SerializedName("data") val data: List<Mahasiswa>
)

data class Mahasiswa(
    @SerializedName("NIM") val nim: String,
    @SerializedName("Nama") val nama: String,
```

```
    @SerializedName("Telepon") val telepon: String
)
```

Buat File Interface Kotlin ('ApiService')

- Interface ini digunakan untuk mengatur endpoint-endpoint HTTP yang akan digunakan untuk mengakses data dari API.
- Interface ApiService ini digunakan sebagai kontrak untuk menghubungkan aplikasi Android dengan API yang menyediakan endpoint "datamahasiswa/".
- Metode `@GET("datamahasiswa/")` akan mengembalikan objek
- `Call<ApiResponse>`, yang nantinya akan digunakan untuk melakukan pemanggilan HTTP ke endpoint tersebut.

```
package com.example.cobaget

import retrofit2.Call
import retrofit2.http.GET

interface ApiService {
    @GET("datamahasiswa/")
    fun getdatamahasiswa(): Call<ApiResponse>
}
```

Buat Objek Class ('ApiClient')

Dengan menggunakan ApiClient ini, kita dapat dengan mudah mengakses endpoint-endpoint API menggunakan objek apiService yang diberikan.

- `BASE_URL`: Konstanta yang menyimpan URL dasar dari API yang akan diakses.
- `val apiService`: Properti yang mengembalikan instance dari ApiService. Properti ini menggunakan getter khusus yang melakukan konfigurasi klien HTTP, Retrofit, dan mengembalikan objek yang dihasilkan dari Retrofit.create() dengan tipe ApiService
- `HttpLoggingInterceptor()`: Digunakan untuk logging HTTP request dan response. Level logging diatur sebagai BODY, yang berarti semua detail request dan response akan dicatat.
- `OkHttpClient`: Klien HTTP yang digunakan oleh Retrofit. Di sini, kita menambahkan HttpLoggingInterceptor ke klien untuk melakukan logging.
- `val retrofit`: Objek Retrofit yang digunakan untuk membuat klien HTTP dan mengkonversi respons HTTP menjadi objek Kotlin. Di sini, kita mengkonfigurasi klien HTTP, menambahkan GsonConverterFactory untuk mengkonversi JSON menjadi objek Kotlin, dan mengatur base URL API.

- retrofit.create(ApiService::class.*java*): Membuat instance dari ApiService berdasarkan konfigurasi Retrofit.

```
package com.example.cobaget

import okhttp3.OkHttpClient
import okhttp3.logging.HttpLoggingInterceptor
import retrofit2.Retrofit
import retrofit2.converter.gson.GsonConverterFactory

object ApiClient {

    const val BASE_URL =
"https://apitani.burunghantu.id/sub/restapi-slim/public/"
    val apiService : ApiService
    get(){
        val interceptor = HttpLoggingInterceptor()
        interceptor.level = HttpLoggingInterceptor.Level.BODY
        val client = OkHttpClient.Builder()
            .addInterceptor(interceptor)
            .build()
        val retrofit = Retrofit.Builder()
            .client(client)
            .addConverterFactory(GsonConverterFactory.create())
            .baseUrl(BASE_URL)
            .build()
        return retrofit.create(ApiService::class.java)
    }
}
```

Buat File Kelas Kotlin ('RVAdapter')

File ini digunakan untuk menghubungkan data 'Mahasiswa' dengan tampilan list item yang ada di RecyclerView

- class RVAdapter : Kelas utama dari Adapter, mewarisi RecyclerView.Adapter dan menggunakan generic type MyViewHolder sebagai ViewHolder yang digunakan.
- class MyViewHolder: Kelas yang merepresentasikan ViewHolder untuk setiap item dalam RecyclerView. Di sini, kita menginisialisasi dan merujuk ke elemen-elemen tampilan yang ada dalam item_layout melalui findViewById().
- onCreateViewHolder(): Metode yang dipanggil ketika RecyclerView membutuhkan ViewHolder baru untuk menampilkan item. Di sini, kita mengembalikan instance dari MyViewHolder dengan menginflasi tampilan item_layout.
- onBindViewHolder(): Metode yang dipanggil ketika RecyclerView ingin mengaitkan data dengan ViewHolder tertentu. Di sini, kita mengatur nilai-nilai TextView dalam ViewHolder berdasarkan data yang diberikan pada

posisi yang sesuai dalam dataList. Juga, kita menambahkan onClickListener pada cvMain (CardView) untuk menampilkan pesan Toast ketika item diklik.

- getItemCount(): Metode yang mengembalikan jumlah item dalam RecyclerView, berdasarkan ukuran dataList
- setData(): Metode untuk mengatur data baru pada Adapter. Di sini, dataList diperbarui dengan data baru yang diterima dan metode notifyDataSetChanged() dipanggil untuk memberi tahu RecyclerView bahwa data telah berubah.

```
package com.example.cobaget
import android.content.Context
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.TextView
import android.widget.Toast
import androidx.cardview.widget.CardView
import androidx.recyclerview.widget.RecyclerView

class RVAdapter(
    private val context: Context,
    private val dataList: ArrayList<Mahasiswa>
) : RecyclerView.Adapter<RVAdapter.MyViewHolder>() {

    class MyViewHolder(val view: View) :
RecyclerView.ViewHolder(view) {
        val tvNim = view.findViewById<TextView>(R.id.nimTextView)
        val tvNama = view.findViewById<TextView>(R.id.namaTextView)
        val tvTelp = view.findViewById<TextView>(R.id.telpTextView)
        val cvMain = view.findViewById<CardView>(R.id.cv_main)
    }

    override fun onCreateViewHolder(parent: ViewGroup, viewType:
Int): MyViewHolder {
        val layoutInflater = LayoutInflater.from(parent.context)
        val itemView = layoutInflater.inflate(R.layout.item_layout,
parent, false)
        return MyViewHolder(itemView)
    }

    override fun onBindViewHolder(holder: MyViewHolder, position:
Int) {
        holder.tvNim.text = dataList[position].nim
        holder.tvNama.text = dataList[position].nama
        holder.tvTelp.text = dataList[position].telepon
        holder.cvMain.setOnClickListener {
            Toast.makeText(context, dataList[position].nama,
Toast.LENGTH_SHORT).show()
        }
    }

    override fun getItemCount(): Int = dataList.size

    fun setData(data: List<Mahasiswa>) {
        dataList.clear()
        dataList.addAll(data)
```

```
            notifyDataSetChanged()
        }
    }
```

MainActivity.Kt

Di MainActivity kita akan mengatur tampilan Activity, menginisialisasi RecyclerView, dan melakukan pengambilan data dari API menggunakan Retrofit.

- onCreate () : Metode ini dipanggil ketika Activity dibuat. Di sini, kita mengatur tampilan Activity menggunakan ActivityMainBinding yang dihasilkan dari ActivityMainBinding.inflate(*layoutInflater*). Selanjutnya, kita menginisialisasi adapter sebagai instance dari RVAdapter dengan menggunakan this@MainActivity sebagai konteks dan arrayListOf() sebagai data awalnya. Kemudian, kita mengatur adapter tersebut pada RecyclerView rvMain yang ada dalam layout menggunakan binding.rvMain.*adapter* = adapter. Terakhir, kita memanggil fungsi remoteGetdatamahasiswa() untuk melakukan pengambilan data dari API.
- remoteGetdatamahasiswa() : Metode ini berfungsi untuk melakukan pengambilan data mahasiswa dari API menggunakan Retrofit. Kita menggunakan ApiClient.apiService untuk mendapatkan instance dari ApiService yang telah dihubungkan dengan Retrofit. Kemudian, kita melakukan pemanggilan API getdatamahasiswa() menggunakan enqueue() untuk melakukan operasi secara asynchronous.
- Dalam callback onResponse(), jika responsenya berhasil (response.isSuccessful), kita mengambil data dari body responsenya dan kemudian memanggil fungsi setDataToAdapter() untuk mengatur data ke adapter. Jika terjadi onFailure, kita mencatat errornya menggunakan Log.d().
- setDataToAdapter(): Metode ini digunakan untuk mengatur data ke adapter. Kita memanggil fungsi setData() pada adapter dan memberikan data mahasiswa yang diterima dari API. Dalam hal ini, kita menggunakan List<Mahasiswa> sebagai tipe data parameter.

```
package com.example.cobaget
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.util.Log
import com.example.cobaget.databinding.ActivityMainBinding
import retrofit2.Call
import retrofit2.Callback
import retrofit2.Response

class MainActivity : AppCompatActivity() {

    private lateinit var binding: ActivityMainBinding
    private lateinit var adapter: RVAdapter

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
```

```
        binding = ActivityMainBinding.inflate(layoutInflater)
        setContentView(binding.root)

        adapter = RVAdapter(this@MainActivity, arrayListOf())
        binding.rvMain.adapter = adapter
        binding.rvMain.setHasFixedSize(true)

        remoteGetdatamahasiswa()
    }

    private fun remoteGetdatamahasiswa() {
        ApiClient.apiService.getdatamahasiswa().enqueue(object :
Callback<ApiResponse> {
            override fun onResponse(call: Call<ApiResponse>,
response: Response<ApiResponse>) {
                if (response.isSuccessful) {
                    val apiResponse = response.body()
                    val data = apiResponse?.data
                    if (data != null) {
                        setDataToAdapter(data)
                    }
                }
            }

            override fun onFailure(call: Call<ApiResponse>, t:
Throwable) {
                Log.d("Error", t.stackTraceToString())
            }
        })
    }

    private fun setDataToAdapter(data: List<Mahasiswa>) {
        adapter.setData(data)
    }
}
```

Output