Rest API
Untuk setting Rest API bisa gunakan api ini : https://apitani.burunghantu.id/sub/restapi-slim/public/datamahasiswa/ .

Setting Gradle
Pada langkah ini kita harus menambahkan tools tambahan pada **build.gradle(Module)** di **project Android Studio**. Penambahan dilakukan dengan menambahkan syntax dibawah ini untuk menambahkan fungsi Rest API

```
    implementation 'com.squareup.retrofit2:retrofit:2.9.0'
    implementation 'com.squareup.retrofit2:converter-gson:2.9.0'
    implementation 'com.squareup.okhttp3:logging-interceptor:4.9.3'
```

Langkah berikutnya adalah menambahkan componen untuk **ViewModel** dan **LiveData** dengan menambahkan syntax dibawah ini pada **gradle**

```
    implementation "androidx.activity:activity-ktx:1.4.0"
    implementation "androidx.lifecycle:lifecycle-viewmodel-ktx:2.4.1"
    implementation "androidx.lifecycle:lifecycle-livedata-ktx:2.4.1"
```

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

```
buildFeatures{
    viewBinding true
}
```

Setting Gradle
Pada **AndroidManifest.xml** setting untuk aplikasi Android dapat **mengakses jaringan internet** pada **perangkat** dengan menambahkan source code dibawah ini **didalam manifest** dan **di atas application**

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

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 ini

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

Sehingga Source Code Lengkap **AndroidManifest.xml** akan menjadi seperti dibawah ini

```
<?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"/>

    <application
        android:usesCleartextTraffic="true"
        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.LatihanRestAPI"
        tools:targetApi="31">
        <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>
```

Theme

Pada theme digunakan untuk menuliskan **style** yang akan digunakan pada aplikasi. Untuk **style** utama dibuat **parent** diganti menggunakan theme material dengan **syntax parent**

```
Theme.MaterialComponents.DayNight.NoActionBar
```

Untuk Source Code lengkap hasil penambahan diatas pada **themes.xml (Light)** adalah sebagai berikut :

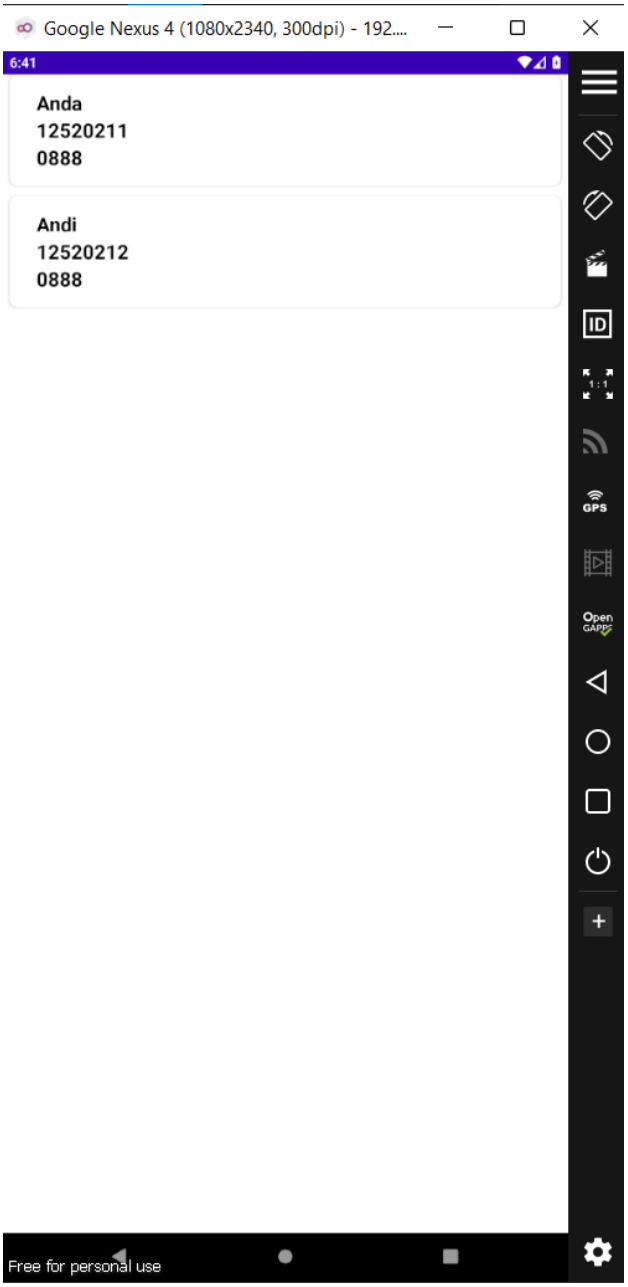
```
<resources>
    <!-- Base application theme. -->
    <style name="Theme.LatihanRestAPI"
parent="Theme.MaterialComponents.DayNight.NoActionBar">
        <!-- Primary brand color. -->
        <item name="colorPrimary">@color/purple_500</item>
        <item name="colorPrimaryVariant">@color/purple_700</item>
        <item name="colorOnPrimary">@color/white</item>
        <!-- Secondary brand color. -->
        <item name="colorSecondary">@color/teal_200</item>
        <item name="colorSecondaryVariant">@color/teal_700</item>
        <item name="colorOnSecondary">@color/black</item>
```

```
        <!-- Status bar color. -->
        <item
name="android:statusBarColor">?attr/colorPrimaryVariant</item>
        <!-- Customize your theme here. -->
    </style>
</resources>
```

Untuk Source Code lengkap hasil penambahan diatas pada **themes.xml (Dark/Night)** adalah sebagai berikut :

```
<resources>
    <!-- Base application theme. -->
    <style name="Theme.LatihanRestAPI"
parent="Theme.MaterialComponents.DayNight.NoActionBar">
        <!-- Primary brand color. -->
        <item name="colorPrimary">@color/purple_200</item>
        <item name="colorPrimaryVariant">@color/purple_700</item>
        <item name="colorOnPrimary">@color/black</item>
        <!-- Secondary brand color. -->
        <item name="colorSecondary">@color/teal_200</item>
        <item name="colorSecondaryVariant">@color/teal_200</item>
        <item name="colorOnSecondary">@color/black</item>
        <!-- Status bar color. -->
        <item
name="android:statusBarColor">?attr/colorPrimaryVariant</item>
        <!-- Customize your theme here. -->
    </style>
</resources>
```

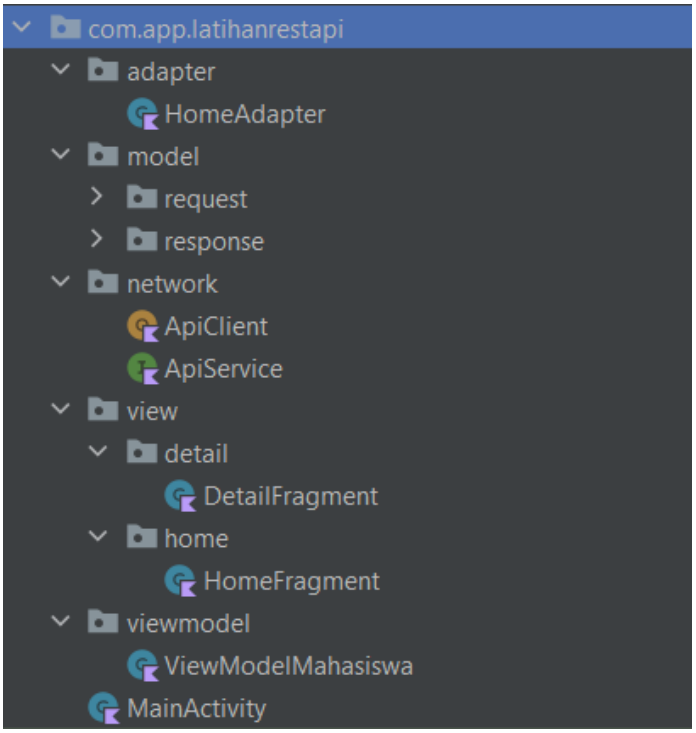
1. **Menampilkan Data dengan GET Method Menggunakan REST API**
   Untuk menampilkan data dengan **GET** Method dimana pada **User Interface** Aplikasi adalah menggunakan **Recyclerview** dengan **Cardview**. Aplikasi yang akan dibuat pada tutorial ini akan seperti pada gambar 1 dimana pada **User Interface** Aplikasi adalah menggunakan Recyclerview dengan **Cardview**.



Gambar 1

1.1. MVVM Design Pattern

MVVM ini digunakan untuk mengatur penempatan file pada project kita, sehingga file yang ada tidak tercampur dengan file lain.



adapter berfungsi untuk menyimpan file adapter, **model** untuk menyimpan file **data class** dan juga **api response** dan **api request**, **network** berfungsi untuk menyimpan file **ApiClient** dan **ApiService**, **view** berfungsi untuk menyimpan file **view** (**Fragment** / **Activity**), **viewmodel** berfungsi untuk menyimpan file **viewModel**.

1.2. activity_main.xml

activity_main.xml merupakan layout yang digunakan untuk **Activity Main** atau **Activity utama** yang akan digunakan. **FragmentContainerView** adalah elemen dalam XML layout file yang berfungsi sebagai wadah untuk menampilkan fragment-fragment dalam aplikas. Untuk syntax dari **activity_main.xml** dapat dilihat dibawah ini.

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


   <androidx.fragment.app.FragmentContainerView
       android:id="@+id/fragmentContainerView"

android:name="androidx.navigation.fragment.NavHostFragment"
       app:defaultNavHost="true"
       android:layout_width="match_parent"
       android:layout_height="match_parent"
```

```
        app:navGraph="@navigation/nav"

        />

</androidx.constraintlayout.widget.ConstraintLayout>
```

1.3. user_list.xml

Sedangkan **user_list.xml** digunakan untuk layout Item yang akan dimunculkan pada **Activity Main**. Untuk syntax dari **user_list.xml** dapat dilihat pada syntax dibawah ini.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.cardview.widget.CardView
    android:id="@+id/cardView"
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:elevation="6dp"
    app:cardCornerRadius="8dp"
    android:layout_height="wrap_content"
    android:layout_marginStart="8dp"
    android:layout_marginEnd="8dp"
    android:layout_marginBottom="10dp"
    xmlns:app="http://schemas.android.com/apk/res-auto">

    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_margin="15dp"
        android:layout_height="wrap_content">

        <TextView
            android:id="@+id/txtNama"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textSize="20sp"
            android:textStyle="bold"
            android:textColor="@color/black"
            android:layout_marginStart="12dp"
            android:text="Nama : Raynaldi Zulfikar"
            />
        <TextView
            android:id="@+id/txtNim"
            android:layout_below="@id/txtNama"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textSize="20sp"
            android:textStyle="bold"
            android:textColor="@color/black"
            android:layout_marginStart="12dp"
            android:text="Nim : 20104042"
            />
        <TextView
            android:id="@+id/txtTelepon"
            android:layout_below="@id/txtNim"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
```

```
            android:textSize="20sp"
            android:textStyle="bold"
            android:textColor="@color/black"
            android:layout_marginStart="12dp"
            android:text="Telepon : 082223018700"
            />

    </RelativeLayout>

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

1.4. fragment_home.xml

fragment_home.xml merupakan file layout yang berfungsi untuk menampilkan data api ke layout dengan menggunakan recyclerview. Untuk syntax dari fragment_home.xml dapat dilihat di bawah ini.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".view.home.HomeFragment">


    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/rvUser"
        android:layout_width="match_parent"
        tools:listitem="@layout/user_list"
        android:layout_height="match_parent">


    </androidx.recyclerview.widget.RecyclerView>

</androidx.constraintlayout.widget.ConstraintLayout>
```

1.5. fragment_detail.xml

fragment_detail.xml merupakan file layout yang berfungsi untuk menampilkan data detail mahasiswa ke layout. Untuk syntax dari fragment_detail.xml dapat dilihat di bawah ini.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".view.detail.DetailFragment">

    <TextView
        android:id="@+id/txtnim"
        android:layout_width="wrap_content"
```

```
        android:layout_height="wrap_content"
        android:layout_marginTop="36dp"
        android:text="NIM"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.498"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />


    <TextView
        android:id="@+id/txtnama"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="44dp"
        android:text="Nama"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/txtnim" />

    <TextView
        android:id="@+id/txtTelepon"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="77dp"
        android:text="Telepon"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.498"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/txtnama" />


</androidx.constraintlayout.widget.ConstraintLayout>
```

1.6. ApiClient.kt

Pada ApiClient.kt digunakan untuk menghubungkan aplikasi android dengan Server Rest API yang diguankan. Dalam tutorial Server Rest API dipanggil pada jaringan lokal dimana URL menggunkana IP Address pada contoh kasus lain Server Rest API dapat menggunakan URL berupa domain dengan ekstensi www. Untuk Source Code dapat dilihat dibawah ini.

```
package com.app.latihanrestapi.network

import retrofit2.Retrofit
import retrofit2.converter.gson.GsonConverterFactory

object ApiClient {
    const val BASE_URL= "https://apitani.burunghantu.id/"

    val instance: ApiService by lazy {
        val retrofit = Retrofit.Builder()
```

```
            .baseUrl(BASE_URL)
            .addConverterFactory(GsonConverterFactory.create())
            .build()
        retrofit.create(ApiService :: class.java)
    }
}
```

1.7. ApiInterface.kt

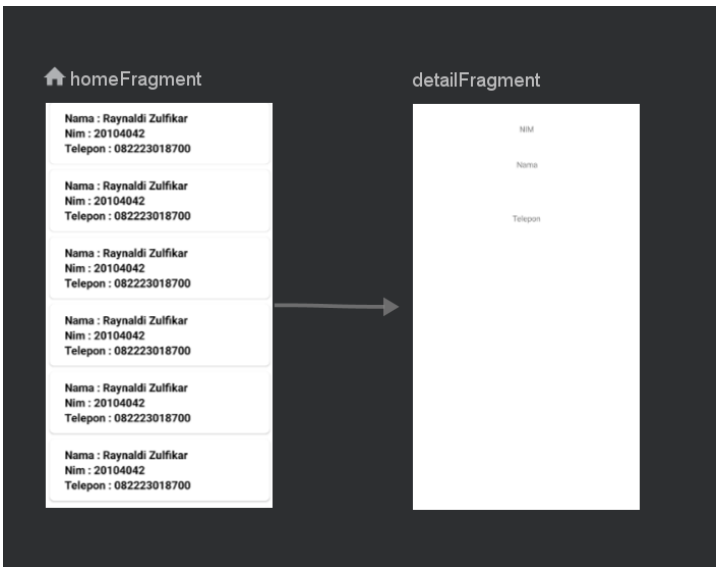
Interface ApiService mendefinisikan dua metode untuk berinteraksi dengan Rest API. Metode pertama, getDataMahasiswa(), mengambil data mahasiswa melalui HTTP GET request ke endpoint "sub/restapi-slim/public/datamahasiswa/".

Sedangkan metode kedua, getDetailMahasiswa(), mengambil detail data mahasiswa berdasarkan NIM melalui HTTP GET request ke endpoint "sub/restapi-slim/public/datamahasiswa/{nim}".

```
package com.app.latihanrestapi.network

import
com.app.latihanrestapi.model.response.ResponseDataMahasiswa
import
com.app.latihanrestapi.model.response.ResponseDetailDataMahasisw
a
import retrofit2.Call
import retrofit2.http.GET
import retrofit2.http.Path

interface ApiService {
    @GET("sub/restapi-slim/public/datamahasiswa/")
    fun getDataMahasiswa() : Call<ResponseDataMahasiswa>

    @GET("sub/restapi-slim/public/datamahasiswa/{nim}")
    fun getDetailMahasiswa(@Path("nim") nim : String) :
Call<ResponseDetailDataMahasiswa>
}
```

1.8. Navigation Component

Navigation Component ini berfungsi untuk mengatur alur perpindahan antar fragment.



1.9. HomeAdapter.kt

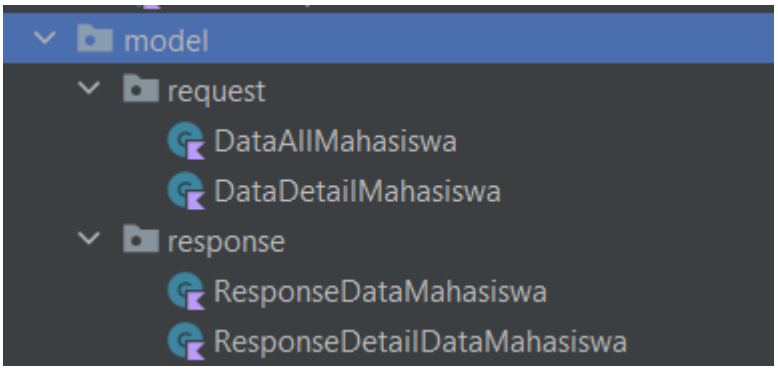
HomeAdapter berfungsi untuk menjadi jembatan antara data dan tampilan dalam daftar item yang ditampilkan di halaman utama (home) aplikasi. Dengan menggunakan HomeAdapter, kita dapat menghubungkan data yang diterima dari sumber data (misalnya daftar objek) dengan tampilan item yang ditampilkan dalam RecyclerView atau ListView.

```
package com.app.latihanrestapi.adapter

import android.os.Bundle
import android.view.LayoutInflater
import android.view.ViewGroup
import androidx.navigation.Navigation
import androidx.recyclerview.widget.RecyclerView
import com.app.latihanrestapi.databinding.UserListBinding
import com.app.latihanrestapi.model.request.DataAllMahasiswa

class HomeAdapter(private var dataMhs : List<DataAllMahasiswa>)
: RecyclerView.Adapter<HomeAdapter.ViewHolder>(){

    class ViewHolder(val binding : UserListBinding) :
RecyclerView.ViewHolder(binding.root)

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

    override fun getItemCount(): Int {
        return dataMhs.size
    }
```

```
    override fun onBindViewHolder(holder: ViewHolder, position:
Int) {
        holder.binding.txtNama.text = dataMhs[position].nama
        holder.binding.txtNim.text = dataMhs[position].nIM
        holder.binding.txtTelepon.text =
dataMhs[position].telepon
        holder.binding.cardView.setOnClickListener{
            val bundle = Bundle()
            bundle.putString("nim",dataMhs[position].nIM)
            bundle.putString("nama",dataMhs[position].nama)
            bundle.putString("telepon",dataMhs[position].telepon)

Navigation.findNavController(it).navigate(com.app.latihanrestapi
.R.id.action_homeFragment_to_detailFragment, bundle)
        }
    }
}
```

1.10. Model Request dan Response

Model Request digunakan untuk mengatur data yang dikirim dari aplikasi ke server atau API, sementara Model Responsedigunakan untuk mengatur data yang diterima dari server sebagai response.



1.11. ViewModelMahasiswa.kt

ViewModelMahasiswa berfungsi untuk memisahkan logika bisnis dan tampilan terkait entitas Mahasiswa dalam aplikasi. Melalui **ViewModelMahasiswa**, pengelolaan dan pemrosesan data Mahasiswa dapat dilakukan secara terpisah dari komponen UI seperti activity atau fragment.

```
package com.app.latihanrestapi.viewmodel

import androidx.lifecycle.MutableLiveData
import androidx.lifecycle.ViewModel
import com.app.latihanrestapi.model.request.DataAllMahasiswa
import
com.app.latihanrestapi.model.response.ResponseDataMahasiswa
import
com.app.latihanrestapi.model.response.ResponseDetailDataMahasisw
a
import com.app.latihanrestapi.network.ApiClient
```

```
import retrofit2.Call
import retrofit2.Callback
import retrofit2.Response

class ViewModelMahasiswa : ViewModel(){
   private val getDataMahasiswa =
MutableLiveData<List<DataAllMahasiswa>?>()
   private val detailMahasiswa =
MutableLiveData<ResponseDetailDataMahasiswa?>()

   fun getDataMahasiswa() :
MutableLiveData<List<DataAllMahasiswa>?>{
       return getDataMahasiswa
   }
   fun getDetailMahasiswa():
MutableLiveData<ResponseDetailDataMahasiswa?> {
       return detailMahasiswa
   }


   fun showDataMahasiswa(){
       ApiClient.instance.getDataMahasiswa().enqueue(object :
Callback<ResponseDataMahasiswa>{
           override fun onResponse(
               call: Call<ResponseDataMahasiswa>,
               response: Response<ResponseDataMahasiswa>)
           {
               if (response.isSuccessful){

getDataMahasiswa.postValue(response.body()?.data)
               }else{
                   getDataMahasiswa.postValue(null)
               }
           }
           override fun onFailure(call:
Call<ResponseDataMahasiswa>,t: Throwable){
               getDataMahasiswa.postValue(null)
           }
       })
   }

   fun getDetailData(nim: String) {
       ApiClient.instance.getDetailMahasiswa(nim).enqueue(object
: Callback<ResponseDetailDataMahasiswa> {
           override fun onResponse(
               call: Call<ResponseDetailDataMahasiswa>,
               response: Response<ResponseDetailDataMahasiswa>
           ) {
               if (response.isSuccessful) {
                   detailMahasiswa.postValue(response.body())
               } else {
                   detailMahasiswa.postValue(null)
               }
           }

           override fun onFailure(call:
Call<ResponseDetailDataMahasiswa>, t: Throwable) {
```

```
                detailMahasiswa.postValue(null)
            }
        })
    }
}
```

1.12. MainActivity.kt

MainActivity.kt merupakan **Activity** utama pada aplikasi Android yang dibuat. Karena kita menggunakan **Navigation Component** dan menggunakan fragment, MainActivity ini hanya digunakan sebagai **fragment host** yang di setting pada **activity_main.xml**

```
package com.app.latihanrestapi

import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle

class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
    }
}
```

1.13. DetailFragment.kt

Detail fragment ini

```
package com.app.latihanrestapi.view.detail

import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.Toast
import androidx.fragment.app.Fragment
import androidx.lifecycle.ViewModelProvider
import
com.app.latihanrestapi.databinding.FragmentDetailBinding
import com.app.latihanrestapi.viewmodel.ViewModelMahasiswa

class DetailFragment : Fragment() {
    lateinit var viewModel : ViewModelMahasiswa
    lateinit var binding : FragmentDetailBinding
    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        // Inflate the layout for this fragment
        binding =
FragmentDetailBinding.inflate(layoutInflater,container,false)
        return binding.root

    }
```

```
    override fun onViewCreated(view: View, savedInstanceState:
Bundle?) {
        super.onViewCreated(view, savedInstanceState)
        val nim = arguments?.getString("nim")

        viewModel =
ViewModelProvider(requireActivity()).get(ViewModelMahasiswa::c
lass.java)

viewModel.getDetailMahasiswa().observe(viewLifecycleOwner){
            if (it != null){
                binding.txtnim.text = it.data?.nIM
                binding.txtnama.text = it.data?.nama
                binding.txtTelepon.text = it.data?.telepon
            }else{
                Toast.makeText(context,"Data tidak
ditemukan",Toast.LENGTH_SHORT).show()
            }
        }
        viewModel.getDetailData(nim!!)
    }
}
```

1.14. HomeFragment.kt

```
  package com.app.latihanrestapi.view.home

import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.fragment.app.Fragment
import androidx.lifecycle.ViewModelProvider
import androidx.recyclerview.widget.LinearLayoutManager
import com.app.latihanrestapi.adapter.HomeAdapter
import com.app.latihanrestapi.databinding.FragmentHomeBinding
import com.app.latihanrestapi.viewmodel.ViewModelMahasiswa

class HomeFragment : Fragment() {
    lateinit var binding : FragmentHomeBinding
    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        // Inflate the layout for this fragment
        binding =
FragmentHomeBinding.inflate(layoutInflater,container,false)
        return binding.root
    }

    override fun onViewCreated(view: View, savedInstanceState:
Bundle?) {
        super.onViewCreated(view, savedInstanceState)

        val viewModel =
ViewModelProvider(this).get(ViewModelMahasiswa::class.java)
```

```
        viewModel.getDataMahasiswa().observe(viewLifecycleOwner)
{
            if (it != null) {
                binding.rvUser.layoutManager =
                    LinearLayoutManager(context,
LinearLayoutManager.VERTICAL, false)
                val adapter = HomeAdapter(it)
                binding.rvUser.adapter = adapter
            }else{
                binding.rvUser.visibility = View.GONE
            }
        }
        viewModel.showDataMahasiswa()
    }
}
```