BAB 6 | Networking

Networking dalam konteks Android Studio merujuk pada proses komunikasi antara aplikasi Android yang sedang berjalan dengan sumber data eksternal, seperti server web, database, atau layanan web lainnya. Ini memungkinkan aplikasi untuk mengambil, mengirim, atau bertukar data dengan sumber eksternal.

Dalam pengembangan aplikasi Android, komunikasi jaringan biasanya dilakukan melalui HTTP/HTTPS menggunakan API (Application Programming Interface) yang disediakan oleh server. Android menyediakan beberapa library dan kelas yang memudahkan pengembang untuk melakukan networking, seperti HttpURLConnection, Retrofit, dan LoopJ.

Beberapa tugas yang umum dilakukan dalam networking Android meliputi:

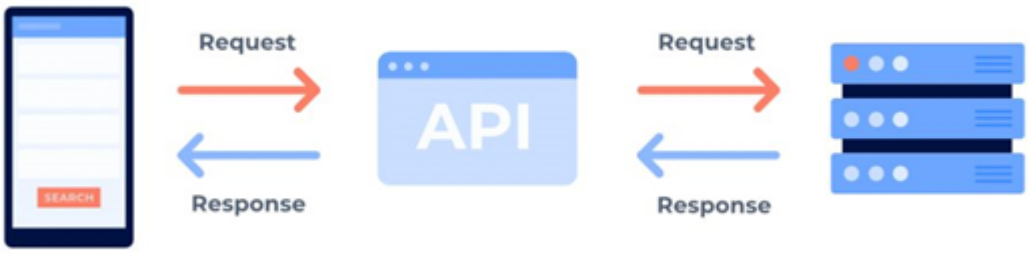
1. Mengambil data dari server (GET request).
2. Mengirim data ke server (POST, PUT, DELETE request).
3. Mengelola respons dari server, seperti JSON atau XML, dan mengubahnya menjadi objek yang dapat digunakan dalam aplikasi.

Dari beberapa tugas umum diatas, pada modul praktikum ini kita akan hanya fokus membahas tentang **mengambil data dari server (Get request)**.

Penting untuk diingat bahwa dalam pengembangan aplikasi, terutama yang berhubungan dengan jaringan, harus memperhatikan aspek keamanan, keandalan, dan performa untuk memastikan bahwa aplikasi dapat berfungsi dengan baik dan aman saat berkomunikasi dengan sumber data eksternal.

API (Application Programming Interface)

API adalah serangkaian definisi dan protokol yang digunakan oleh aplikasi perangkat lunak untuk berkomunikasi satu sama lain ataupun ke server. Dalam konteks pengembangan perangkat lunak, API biasanya digunakan untuk mengintegrasikan antara aplikasi dengan server/database. Untuk lebih memahami penggunaan API silahkan simak gambar dibawah:



Kesimpulan dari gambar diatas adalah API dapat digunakan sebagai jembatan antara aplikasi mobile kita dengan server/database yang tersedia melalui request & response.

Dalam konteks API, request dan response adalah dua komponen penting yang memungkinkan interaksi antara aplikasi atau layanan yang berbeda. Berikut penjelasan singkat mengenai keduanya:

1. **Request (Permintaan)**:
    - Request adalah permintaan yang dikirim oleh klien atau pengguna aplikasi kepada server atau layanan API.
    - Request biasanya berisi informasi mengenai tindakan atau operasi yang ingin dilakukan, seperti mengambil data, menyimpan data, mengupdate data, atau melakukan operasi lainnya.
    - Struktur dari request biasanya tergantung pada jenis API yang digunakan. Misalnya, pada RESTful API, request seringkali berupa HTTP methods seperti GET, POST, PUT, atau DELETE, serta berisi path endpoint dan parameter-parameter yang diperlukan.
2. **Response (Tanggapan)**:
    - Response adalah balasan atau hasil yang diberikan oleh server atau layanan API sebagai respons terhadap request yang diterima.
    - Response berisi data atau informasi yang diminta dalam request atau informasi lain yang relevan dengan operasi yang dilakukan.
    - Struktur dari response juga tergantung pada jenis API yang digunakan. Biasanya, response berisi status code untuk menunjukkan apakah operasi berhasil atau gagal (misalnya, status code 200 untuk sukses, 404 untuk not found, 500 untuk error server), serta body yang berisi data atau informasi yang diminta.

Contoh sederhana:

- Request

```
GET /api/users/123
```

- Response

```
{
  "id": 123,
  "name": "John Doe",
  "email": "johndoe@example.com"
}
```

Penggunaan Retrofit

Retrofit adalah sebuah library yang populer digunakan dalam pengembangan aplikasi Android untuk melakukan HTTP requests ke sebuah web service berbasis RESTful. Library ini memudahkan proses komunikasi antara aplikasi Android dengan server dengan cara mengkonversi data JSON yang diterima dari server menjadi objek Java yang mudah diolah dan sebaliknya.

Berikut adalah beberapa fitur dan keuntungan menggunakan Retrofit:

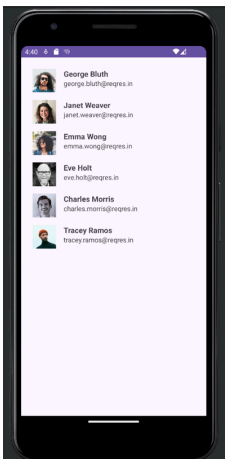
Fitur dan Keuntungan Retrofit:

1. **Mudah Digunakan**: Retrofit menyediakan antarmuka yang mudah untuk mendefinisikan endpoints API dan parameter yang diperlukan untuk request.
2. **Konversi Otomatis**: Retrofit secara otomatis mengkonversi respons dari server yang biasanya berbentuk JSON ke dalam objek Java menggunakan library seperti Gson atau Moshi.
3. **Integrasi dengan OkHttp**: Retrofit dibangun di atas library OkHttp, sehingga memiliki fitur caching, logging, dan interceptors yang kuat.
4. **Support untuk RxJava**: Retrofit mendukung RxJava untuk menghandle asynchronous programming.
5. **Error Handling yang Baik**: Retrofit memudahkan dalam menangani error seperti HTTP status code yang tidak berhasil.

Contoh Penggunaan Retrofit

Jadi untuk contoh kali ini, kita akan menggunakan API dari https://reqres.in/ untuk latihan. Reqres.in adalah sebuah platform mock API yang digunakan untuk pengembangan dan testing aplikasi. Platform ini menyediakan berbagai endpoint API yang dapat digunakan untuk mendapatkan respons yang telah ditentukan tanpa perlu mengimplementasikan backend server yang sebenarnya. Ini sangat berguna untuk pengembang yang ingin melakukan pengujian atau prototyping tanpa harus tergantung pada backend yang nyata.

Output yang kita harapkan:



Penjelasan Kasus:

Kita akan melakukan hit/penembakan API https://reqres.in/api/users?page=1 yang mana akan mengembalikan response berupa JSON. Data yang kita dapatkan dari response tersebut, kemudian kita tampilkan menggunakan RecyclerView.

Source Code:

Karena kita ingin mengakses internet, maka kita harus untuk memberikan permission aplikasi kita untuk mengakses internet:

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" />

      <application
      …
      </application>

</manifest>
```

Setelah itu, kita perlu menginstal beberapa library yang dibutuhkan:

build.gradle.kts

```
 dependencies {

      …
      // Retrofit
      implementation ("com.squareup.retrofit2:retrofit:2.9.0") // Instal
retrofit2
      implementation ("com.squareup.retrofit2:converter-gson:2.9.0") //
Instal retrofit2-converter


      // OkHttp
      implementation ("com.squareup.okhttp3:okhttp:4.9.1") // Instal okhttp
(caching, logging, dll)
      implementation ("com.squareup.picasso:picasso:2.71828") // Instal
picasso (Menampilkan gambar)


 }
```

Jika dua hal diatas sudah aman, maka sekarang kita lanjut untuk membuat tampilan RecycleView nya pada XML:

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
 <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
      xmlns:app="http://schemas.android.com/apk/res-auto"
      android:layout_width="match_parent"
      android:layout_height="match_parent">

     <androidx.recyclerview.widget.RecyclerView
      android:id="@+id/recyclerView"
      android:layout_width="match_parent"
      android:layout_height="match_parent"
      android:padding="16dp"
      app:layoutManager="androidx.recyclerview.widget.LinearLayoutManager"
      />

 </RelativeLayout>
```

User_item.xml

```
<?xml version="1.0" encoding="utf-8"?>
 <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:orientation="horizontal"
      android:padding="8dp">

      <ImageView
      android:id="@+id/avatarImageView"
      android:layout_width="50dp"
         android:layout_height="50dp"
      android:src="@drawable/ic_launcher_background"/>

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

      <TextView
            android:id="@+id/nameTextView"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textStyle="bold"
            android:textSize="16sp"
            android:text="Name" />

      <TextView
```

```
            android:id="@+id/emailTextView"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Email" />

      </LinearLayout>

 </LinearLayout>
```

Sekarang kita masuk ke bagian yang menantangnya di javanya, pertama – tama kita perlu membuat Class baru untuk memanggil atau menetapkan root link API nya dan mendeklarasikan retrofitnya.

RetrofitClient.java

```
import retrofit2.Retrofit;
 import retrofit2.converter.gson.GsonConverterFactory;

 public class RetrofitClient {
      private static final String BASE_URL = "https://reqres.in/";

      private static Retrofit retrofit = null;

      public static Retrofit getClient() {
      if (retrofit == null) {
            retrofit = new Retrofit.Builder()
                  .baseUrl(BASE_URL)
                     .addConverterFactory(GsonConverterFactory.create())
                  .build();
      }
      return retrofit;
      }
 }
```

Selanjutnya kita membuat Class baru lagi yang berfungsi sebagai kode untuk menarik property dari JSON nya nanti.

User.java

```
public class User {

      private int id;
      private String email;
      private String first_name;
      private String last_name;
      private String avatar;

      public int getId() {
      return id;
```

```
    }

    public void setId(int id) {
    this.id = id;
    }

    public String getEmail() {
    return email;
    }

    public void setEmail(String email) {
    this.email = email;
    }

    public String getFirst_name() {
    return first_name;
    }

    public void setFirst_name(String first_name) {
    this.first_name = first_name;
    }

    public String getLast_name() {
    return last_name;
    }

    public void setLast_name(String last_name) {
    this.last_name = last_name;
    }

    public String getAvatar() {
       return avatar;
    }

    public void setAvatar(String avatar) {
    this.avatar = avatar;
    }
}
```

Selanjutnya kita membuat Class baru lagi, untuk menampung hasil dari response nya.

UserResponse.java

```
import java.util.List;

public class UserResponse {

    private List<User> data;
```

```
      public List<User> getData() {
      return data;
      }

     public void setData(List<User> data) {
      this.data = data;
      }
}
```

Kemudian kita akan membuat interface baru untuk memanggil parameter dari link API nya beserta query page nya.

ApiService.java

```
import retrofit2.Call;
import retrofit2.http.GET;
import retrofit2.http.Query;

public interface ApiService {
      @GET("api/users")
      Call<UserResponse> getUsers(@Query("page") int page);
}
```

Next kita membuat Adapter untuk menjadi jembatan antara data yang kita dapat dengan RecycleView nya.

UserAdapter.java

```
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageView;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;

import com.squareup.picasso.Picasso;

import java.util.List;

public class UserAdapter extends
RecyclerView.Adapter<UserAdapter.UserViewHolder> {
      public List<User> userList;

      public UserAdapter(List<User> userList){
      this.userList = userList;
```

```
        }

        public UserViewHolder onCreateViewHolder(@NonNull ViewGroup parent,
int viewType) {
        View view =
LayoutInflater.from(parent.getContext()).inflate(R.layout.user_item, parent,
false);
        return new UserViewHolder(view);
        }

        public void onBindViewHolder(@NonNull UserViewHolder holder, int
position) {
        User user = userList.get(position);
        holder.bind(user);
        }

        public int getItemCount() {
           return userList.size();
        }

        public static class UserViewHolder extends RecyclerView.ViewHolder {

        private ImageView avatarImageView;
        private TextView nameTextView;
        private TextView emailTextView;

        public UserViewHolder(@NonNull View itemView) {
              super(itemView);
              avatarImageView = itemView.findViewById(R.id.avatarImageView);
              nameTextView = itemView.findViewById(R.id.nameTextView);
              emailTextView = itemView.findViewById(R.id.emailTextView);
        }

        public void bind(User user) {
              Picasso.get().load(user.getAvatar()).into(avatarImageView);
              nameTextView.setText(user.getFirst_name() + " " +
user.getLast_name());
              emailTextView.setText(user.getEmail());
        }
        }

 }
```

Dan terakhir kita tinggal memanggil semua function – function yang kita buat di MainActivity kita.

MainActivity.java

```
import androidx.appcompat.app.AppCompatActivity;
 import androidx.recyclerview.widget.LinearLayoutManager;
 import androidx.recyclerview.widget.RecyclerView;

 import android.os.Bundle;
```

```
import java.util.List;

import retrofit2.Call;
import retrofit2.Callback;
import retrofit2.Response;

public class MainActivity extends AppCompatActivity {

      private ApiService apiService;
      private RecyclerView recyclerView;
      private UserAdapter userAdapter;

      @Override
      protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);

      apiService = RetrofitClient.getClient().create(ApiService.class);
      recyclerView = findViewById(R.id.recyclerView);
      recyclerView.setLayoutManager(new LinearLayoutManager(this));

      Call<UserResponse> call = apiService.getUsers(1);

      call.enqueue(new Callback<UserResponse>() {
            @Override
            public void onResponse(Call<UserResponse> call,
Response<UserResponse> response) {
                  if (response.isSuccessful()) {
                  List<User> users = response.body().getData();
                  userAdapter = new UserAdapter(users);
                   recyclerView.setAdapter(userAdapter);
                  } else {
                  // Handle error
                  }
            }

            @Override
            public void onFailure(Call<UserResponse> call, Throwable t) {
                  // Handle failure
            }
      });
      }

}
```

Pastikan perangkat anda terhubung dengan jaringan, kemudian lari / run.