|  |  |
| --- | --- |
| **Materi** | **Nilai** |
| Android Studio Minggu Ke-8 SMST5 (Video Mengimplementasikan Fragment pada Android & Playlist Movie Catalog Retrofit + API) | 95 |

**Saya Sudah Belajar dan Mengerti dan Saya BISA**

1. Fragment pada Android adalah bagian dari user interface atau perilaku sebuah aplikasi yang biasanya ditempatkan pada activity yang memungkinkan untuk mendesain activity yang menular. Singkatnya, fragment bisa dikatakan sebagai sub dari activity itu, karena fragment ini tidak bisa bekerja tanpa adanya activity. Jadi fragment ini memerlukan activitynya.

2. Membuat button navigation yang menampilkan icon dan titlenya di dalam aplikasi menggunakan app:menu. Caranya membuat Android Resource Directory di dalam res dengan type menu, kemudian di dalam directory menu ini ditambahkan Android Resource File dengan nama bottom\_menu. Di dalam file bottom\_menu ini diberi 4 item untuk home, dashboard, notification, dan account yang setiap itemnya menyimpan id, icon, dan title. Salah satu itemnya seperti ini

<item  
 android:id="@+id/fr\_home"  
 android:icon="@drawable/ic\_home"  
 android:title="@string/home"/>

Item yang pertama ada item dengan id fr\_home, iconnya diambil dari directory drawable yang sebelumnya sudah ditambahkan beberapa icon dari vector asset, kemudian ada title untuk keterangan icon itemnya. Setelah membuat menu, di dalam activity\_main.xml ditambahkan FrameLayout untuk tampilan layarnya atau halaman isinya (content nya),

<FrameLayout  
 android:id="@+id/fragment\_container"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:layout\_marginBottom="56dp"  
 app:layout\_constraintLeft\_toLeftOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent"/>

Layout diatas diatur untuk lebar dan panjangnya match\_parent atau full layar namun tetap berjarak dengan layar bawah sebesar 56dp. Selanjutnya ditambahkan untuk bottom navigation dengan cara mengetikkan <bottom lalu pilih

<com.google.android.material.bottomnavigation.BottomNavigationView

Dan enter, ketika dienter nanti di dalamnya akan otomatis muncul

android:layout\_width="0dp"  
android:layout\_height="wrap\_content"

yang width dan height nya bisa diatur, kemudian ditambahkan

android:id="@+id/navigation"

untuk diberi id navigation itu, dan layoutnya

app:layout\_constraintBottom\_toBottomOf="parent"  
app:layout\_constraintLeft\_toLeftOf="parent"  
app:layout\_constraintRight\_toRightOf="parent"

untuk mengatur layout navigationnya, dan terakhir adalah mengambil menunya dari directory menu dan file bottom\_menu tadi seperti ini

app:menu="@menu/bottom\_menu"

dengan begitu item yang ada di menu tadi menjadi bottom navigation di activity\_main.xml nya.

3. Menyambungkan fragment dengan item menu bottom navigation, misalkan item home ketika diklik akan membuka fragment home, item dashboard diklik akan membuka fragment dashboard, dan seterusnya. Caranya dari directory com.suhailahnfsella.fragment klik kanan new->fragment->fragment(blank) kemudian ubah name nya dari BlankFragment menjadi HomeFragment untuk fragmentnya home, dan ulangi menambahkan beberapa fragment untuk tiap item menu bottom navigation. Setelah membuat 4 fragment, pasti ada file java dan xmlnya, untuk mengatur tampilan fragment bisa melalui file xml nya. Agar fragment dan item menu tersambung, tambahkan code

implements BottomNavigationView.OnNavigationItemSelectedListener

pada

public class MainActivity extends AppCompatActivity

kemudian pada function onCreate nya tambahkan

BottomNavigationView navigationView = findViewById(R.id.*navigation*);  
navigationView.setOnNavigationItemSelectedListener(this);

Untuk mengambil komponen navigasinya dan diletakkan di dalam objek navigationView lalu dibuatkan functionnya yaitu setOnNavigationItemSelectedListener. Di dalam function tersebut diisi dengan

Fragment fragment = null;

Untuk membuat object fragment yang kosong atau belum terisi, kemudian dibuatkan switch atau pilihan dengan beberapa case

switch (item.getItemId()){  
 case R.id.*fr\_home*:  
 fragment = new HomeFragment();  
 break;  
  
 case R.id.*fr\_dashboard*:  
 fragment = new DashboardFragment();  
 break;  
  
 case R.id.*fr\_notification*:  
 fragment = new NotificationFragment();  
 break;  
  
 case R.id.*fr\_account*:  
 fragment = new AccountFragment();  
 break;

karena sebelumnya mengambil komponen navigation yang memiliki beberapa item dengan masing-masing id, maka dalam switchnya dibuatkan pilihan berdasarkan idnya. Case pertama apabila yang diklik adalah item menu ber-id fr\_home maka fragment akan diisi dengan HomeFragment, case kedua apabila yang diklik adalah item menu ber-id fr\_dashboard maka fragment akan diisi dengan DashboardFragment, begitu seterusnya. Kemudian diluar switch ditambahkan

return loadFragment(fragment);

yang berarti fragment dari switch tadi akan dikirimkan ke dalam function loadFragment. Di dalam loadFragment ini berisi

if (fragment != null){  
 getSupportFragmentManager()  
 .beginTransaction()  
 .replace(R.id.*fragment\_container*, fragment)  
 .commit();  
 return true;  
}

return false;

yang apabila fragment tidak kosong maka dia akan menjalankan atau membuka halaman fragment sesuai dengan isi dari fragment yang dikirim ke function ini tadi. Misalnya isi fragment tadi HomeFragment maka dia akan membuka fragment HomeFragment. Terakhir di dalam function onCreate ditambahkan

loadFragment(new HomeFragment());

agar ketika aplikasi dibuka atau dimuat ulang, loadFragment atau function yang menjalankan fragment terisi denagn HomeFragment, jadi tampilan content aplikasi awalnya adalah fragment HomeFragment.

4. Terdapat Retrofit yang merupakan sebuah library khusus untuk pemrograman android yang fungsinya untuk menghandle sesuatu yang berkaitan dengan koneksi data dari android ke internet. Contohnya merequest data ke server kemudian menghandle apa yang akan terjadi saat koneksi timeout.

5. Menambahkan resource menu yang berisi item, item ini berisi 1 image untuk meletakkan foto nanti dan 2 textView untuk judul serta deskripsi singkatnya. Menu ini nantinya dimasukkan ke dalam recyclerview yang ada di main activity agar item-itemnya dapat tampil di recyclerview.

6. Menambahkan sebuah icon yang berguna untuk search dengan cara menambahkan vector asset dan memilih icon search. Icon tersebut akan otomatis tertambah ke dalam aplikasi dan terletak di kanan atas, begitu juga fungsinya yang langsung bisa digunakan untuk mencari sesuatu.

7. Menambahkan file activity\_detail\_movie di dalam layout dan javanya sebagai halaman baru yang akan muncul ketika salah satu menu item dalam recyclerview diklik (menggunakan onClick nantinya) jadi file activity\_detail\_movie ini berfungsi untuk menampilkan page baru yang berisi detail jelasnya dari item menu yang diklik.

8. Membuat sebuah API yang digunakan di dalam project android studio yaitu movie catalog melalui website tmdb yang nanti bisa diambil API KEY nya atau beberapa pendukung lainnya.

9. Kemudian API\_KEY tadi bisa dicopas dan diletakkan di dalam get-movie-details pada bagian api\_key, lalu memilih movie\_id popular, language nya en-US kemudian send request sehingga muncul beberapa code yang bisa dicopas.

10. Code yang dicopas dari send request tadi diletakkan di <https://www.jsonschema2pojo.org/> namun sebelum mengpaste code yang tadi, ubah source type ke JSON, annotation style ke Gson terlebih dahulu baru dipaste. Jika sudah diubah dan dipaste codenya, bisa dilihat pada preview untuk mengambil code yang bisa digunakan di dalam android studionya. Kemudian code ini diletakkan pada file response dan result dengan mengambil bagian tertentu sesuai tempatnya. Misalnya class example untuk file response dan class result untuk file result.

11. Terdapat tag public static final String URL\_BASE = <https://api.themoviedb.org>; yang berfungsi untuk mengambil url yang digunakan untuk mengambil data secara keseluruhan ke dalam project.

12. Menambahkan beberapa implement pendukung dengan cara mengimportnya di dalam file gradle, contoh implement nya yaitu retrofit, glid, dan lainnya kemudian mengsinkronkannya atau sync.

13. Tag String API\_KEY = “7bc278ada6533356ee65d0ed29880932” berfungsi untuk mengambil data API Key yang telah dibuat sebelumnya untuk mengambil data keseluruhan.

14. Menambahkan object Retrofit dengan nama retrofit yang berisi null atau bernilai null, codenya seperti ini private static Retrofit retrofit = null; kemudian retrofit ini digunakan pada if berikut

public static Retrofit getClient(){  
 if (*retrofit* == null){  
 *retrofit* = new Retrofit.Builder().baseUrl(*BASE\_URL*).addConverterFactory(GsonConverterFactory.*create*()).build();  
 }  
 return *retrofit*;  
}

yang berarti apabila object retrofit bernilai null, maka dia akan menjalankan if yang isinya retrofit diisi dengan new retrofit seperti code diatas yang fungsinya untuk membuat sebuah baseurl baru dengan menggunakan BASE\_URL yang sudah dibuat sebelumnya untuk diconvertkan ke dalam bentuk Gson dan dibuat ulang atau build.

15. Terdapat

imgPoster = itemView.findViewById(R.id.*imgMovie*);  
tvTitle = itemView.findViewById(R.id.*tvTitle*);  
tvDescription = itemView.findViewById(R.id.*tvDescription*);

yang berfungsi untuk mengisi variable tersebut dengan data yang diambil dari komponen ber-id sesuai yang dipilih.

16. private MovieAdapter adapter; digunakan untuk membuat sebuah variable baru dengan type private.

17. title = result.getOriginalTitle(); digunakan untuk mengambil data result dengan mengambil data Title atau judul yang telah disediakan oleh Response sebelumnya dan melalui retrofit ditampilkan kembali ke dalam project.

18. Tag Glide.with(getApplicationContext()) digunakan untuk membuat sebuah icon atau gambar yang telah disediakan oleh website tmdbnya.

19. Tag overview = result.getOverview(); digunakan untuk mengambil data secara keseluruhan untuk ditampilkan kembali.

20. Tag tvTitle.setText(title); digunakan agar tvTitle diisi atau disetting dengan text dari variable title.

21. Tag image = result.getPosterPath(); berfungsi untuk mengambil data gambar yang telah dibuat sebelumnya melalui API yang telah disediakan di dalam file result.

22. Tag

load("https://image.tmdb.org/t/p/w185" + resultList.get(position).getPosterPath()).  
into(holder.imgPoster);

berfungsi untuk mengambil data sebuah gambar dari website tmdb yang telah disediakan dengan menggunakan statement getPosterPath yang akan digunakan di beberapa statement.

23. Tag

holder.tvTitle.setText(resultList.get(position).getTitle());

berfungsi untuk mengsetting atau mengatur sebuah judul yang didapatkan dari data API yang terlah dibuat sebelumnya.

24. Tag

Glide.*with*(context)

Berfungsi untuk membuat sebuah statement Glid dengan menggunakan variable atau komponen context yang telah dibuat sebelumnya.

25. Code

List<Result> mList = response.body().getResults();

adapter = new MovieAdapter(MainActivity.this, mList);  
recyclerView.setAdapter(adapter);

Berarti mList merupakan sebuah object dari List<Result> yang diisi dengan statement function body() dengan mengambil data hasil atau getResults yang terdapat di dalam file response yang di dalamnya terdapat coding API bagian results atau hasil API tersebut. Kemudian terdapat variable adapter yang diisi dengan sebuah objek baru dengan mengambil data dari sebuah MainActivity. Kemudian recyclerView.setAdapter(adapter) berarti recyclerView yang ada di main activity diisi dengan data dari adapternya.

26. Tag

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

Berfungsi agar project tersebut dapat terhubung ke internet secara public dengan melalui beberapa perizinan dalam mengambil data API secara keseluruhan dengan API yang telah dibuat sebelumnya.

27. Tag String LANGUAGE = “en-US” berfungsi untuk mengambil data bahasa dari data keseluruhan yang akan diambil datanya. Tag String CATEGORY = “popular” berfungsi untuk mengambil data dengan category popular sesuai dengan pilihan atau data yang ingin diambil saat pembuatan API tersebut.

28. Code

ApiInterface apiInterface = ApiClient.*getClient*().create(ApiInterface.class);

Berfungsi untuk mengambil data dari sebuah file Interface yang di dalamnya terdapat sebuah statement GET API yang telah dibuat sebelumnya.

29. Code

Call<Response> call = apiInterface.getMovie(CATEGORY, API\_KEY, LANGUAGE, PAGE);

Berfungsi untuk mengambil data dari sebuah file Response yang berisi API dengan menggunakan atau memanggil beberapa data sesuai dengan beberapa variable data yang telah dibaut sebelumnya dan memasukkannya ke dalam sebuah variable.

30. Code

call.enqueue(new Callback<Response>() {

berfungsi untuk menggunakan statement callback atau memanggil kembali sebuah file Response dengan menggunakan statement onResponse

31. Tag view = inflater.inflate(R.layout.item\_movie, parent, false); berfungsi untuk mengisi pada bagian beberapa komponen dengan mengambil id komponen tersebut dengan parent.

32. Tag

searchView = (SearchView) menu.findItem(R.id.*search*).getActionView();

digunakan untuk membuat statement baru dari sebuah objek atau variable yang telah dibuat sebelumnya dengan menggunakan sebuah id komponen tersebut yang digunakan untuk membuat sebuah statement pencarian dari nama atau judul data yang diambil dari API yang telah dibuat sebelumnya di website tmdb.

34. @Path(“category”) String category berfungsi untuk mengambil atau menggunakan statement path dengan dibuatkannya sebuah string variable category yang telah disesuaikan dengan data yang akan diambil.

35. @GET(“/3/search/movie”) digunakan untuk mengambil data secara keseluruhan yang nantinya akan digunakan di dalam beberapa statement dengan mengambil beberapa data menggunakan statement getQuery yang berisi beberapa variable pendukung agar data tersebut dapat diambil dan ditampilkan kembali.

36. Terdapat @GET(“/3/movie/{category}”) untuk mengambil data dari sebuah link yang telah didapatkan sebelumnya dengan melihat beberapa kategori data tersebut.

37. Call<Response> getMovie() digunakan untuk memanggil data/function/file Response (sesuai nama <> nya) yang telah dibuat sebelumnya untuk diisi dengan beberapa element yang dibutuhkan data untuk ditampilkan.

38. Tag private static final String BASE\_URL = ServerConfig.API\_ENDPOINT; digunakan untuk mengatur final sebuah variable BASE\_URL dengan file serverconfig yang telah dibuat sebelumnya.

39. Tag public static final String EXTRA\_MOVIE = “string\_extra”; yang digunakan untuk membuat sebuah String extra yang digunakan dalam beberapa statement yang akan dibuat.

40. getIntent().getParcelableExtra(EXTRA\_MOVIE); digunakan untuk mengirim data antar activity secara keseluruhan dan ditampilkan kembali secara keseluruhan data tersebut.

41. Tag

viewHolder.relativeLayout.setOnClickListener(new View.OnClickListener() {

digunakan untuk membuat sebuah event OnClickListener atau saat user mengklik sebuah opsi tersebut maka akan menjalankan sebuah statement di dalamnya. Di dalam tanda kurung kurawalnya berisi

public void onClick(View v) {  
 Intent intent = new Intent(parent.getContext(), DetailMovieActivity.class);  
 Result result = new Result();  
 result.setOriginalTitle(resultList.get(viewHolder.getAdapterPosition()).getOriginalTitle());  
 result.setOverview(resultList.get(viewHolder.getAdapterPosition()).getOverview());  
 result.setPosterPath(resultList.get(viewHolder.getAdapterPosition()).getPosterPath());  
 intent.putExtra(DetailMovieActivity.*EXTRA\_MOVIE*, result);  
 parent.getContext().startActivity(intent);  
}

yang berarti Intent intent = new Intent dan seterusnya itu berfungsi untuk membuat sebuah intent baru atau proses mengirim data dengan menggunakan Context dari file Detail yang telah dibuat sebelumnya. Kemudian ada Result result yang berfungsi untuk membuat sebuah objek baru. result.setOriginalTitle(resultList.get(viewHolder.getAdapterPosition()).getOriginalTitle()); digunakan untuk mensetting atau mengatur sebuah variable atau komponen dengan mengambil data Tittle API yang telah dibuat sebelumnya di dalam file Result.

42. getSupportActionBar().setDisplayHomeAsUpEnabled(true); digunakan untuk membuat sebuah function atau statement pada panah balik atau back untuk kembali ke halaman utama maupun halaman sebelumnya yang ada di bagian bawah layar mobile.

**Saya Belum Mengerti**

1.