**LAPORAN PRAKTIKUM PENGEMBANGAN APLIKASI BERGERAK**

**ANDROID FUNDAMENTAL 3 - WEEK 9**



Disusun oleh:

|  |  |  |
| --- | --- | --- |
| Nama | : | Alfath Roziq Widhayaka |
| Nim | : | L0122012 |
| Kelas | : | A |

**PROGRAM STUDI INFORMATIKA**

**FAKULTAS TEKNOLOGI INFORMASI DAN SAINS DATA**

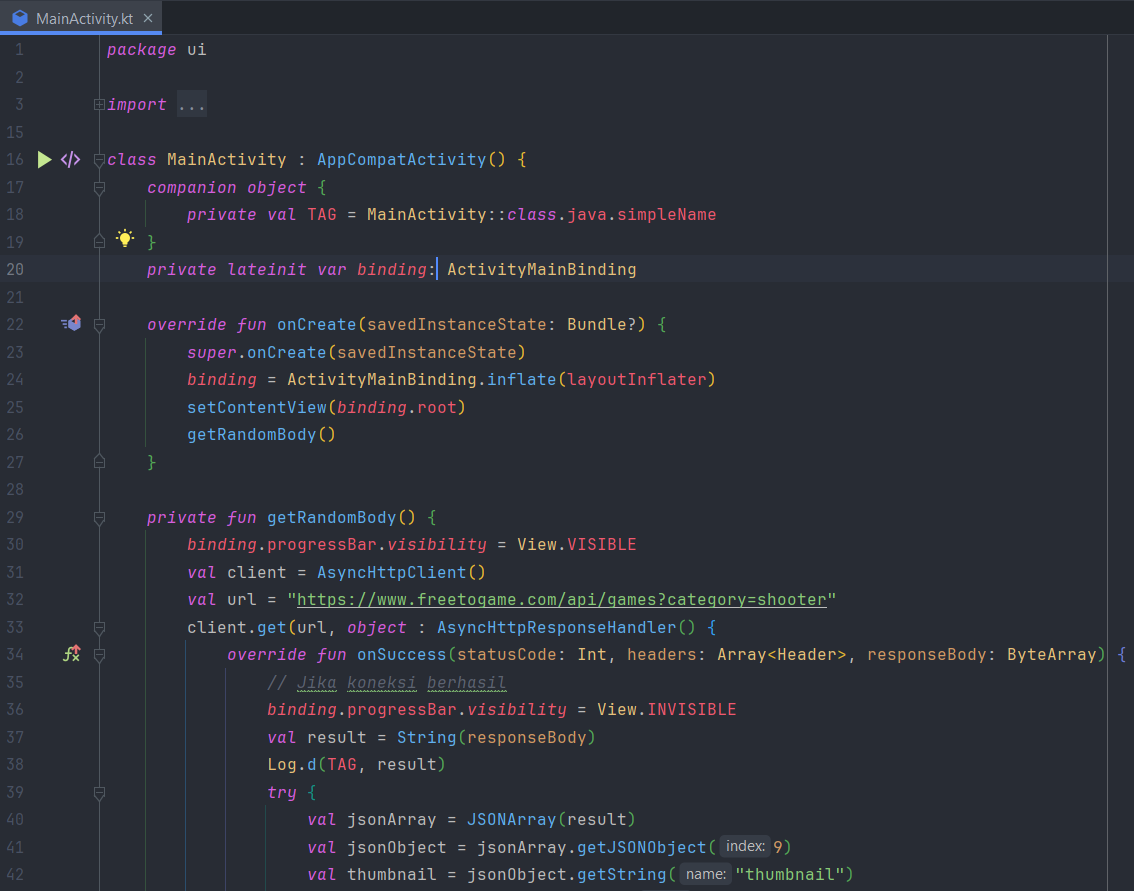
**UNIVERSITAS SEBELAS MARET**

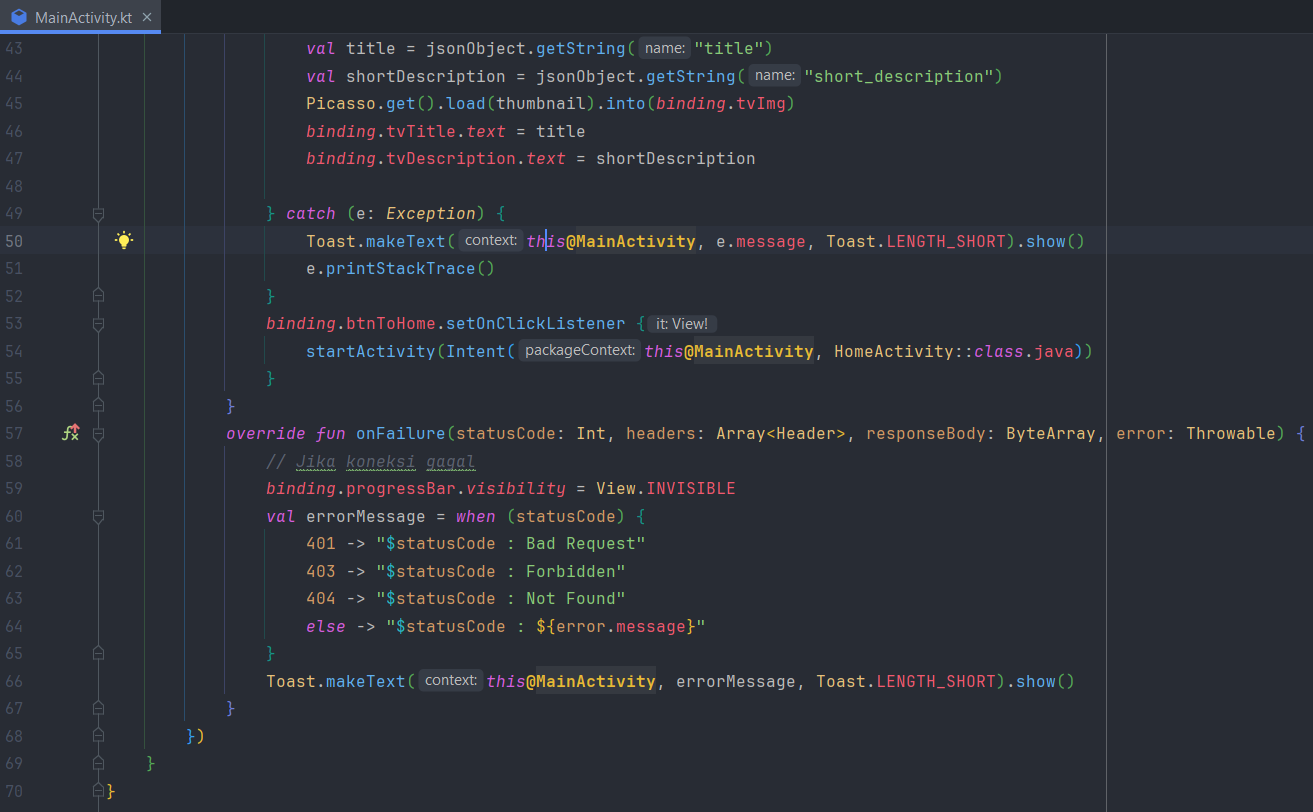
**2024**

1. **Screenshot Source Code**

Source code dibawah ini merupakan source code praktikum PAB ke-9 yang menggunakan **Networking dengan library Retrofit** yang berisi **API** dari public API di Android Studio. **Retrofit** adalah sebuah library Android yang dapat digunakan untuk mempermudah proses pertukaran data antara aplikasi Android dengan server melalui REST API. Retrofit dapat digunakan untuk mengkonsumsi API. Disini saya menggunakan public API FreeToGame dengan link <https://www.freetogame.com/api/games?category=shooter>. Berikut penjelasan masing-masing source code pada aplikasi ini.

1. **Pembuatan Tampilan Utama (Main Activity / Halaman 1).**
2. **MainActivity.kt.**

****

****

*Gambar 1 MainActivity.kt*

Kode di atas mendefinisikan **MainActivity.kt** pada tampilan awal aplikasi yang menggunakan Networking library dengan **LoopJ**. Dengan menggunakan **LoopJ**, aplikasi menampilkan data dari API eksternal menggunakan **AsyncHttpClient**. Pada metode **onCreate**, **ActivityMainBinding** digunakan untuk mengatur tampilan, kemudian **getRandomBody** dipanggil untuk mengambil data pada API nomor 9 dari endpoint API **"https://www.freetogame.com/api/games?category=shooter"**. Pemanggilan data ini terlihat pada kode **val jsonObject = jsonArray.getJSONObject(9)**. Saat data diambil dengan sukses onSuccess, JSON array direspons dikonversi menjadi string, dan elemen kesepuluh diambil untuk menampilkan **thumbnail, title,** dan **short\_description** menggunakan **Picasso** dan **TextView**. Jika ada kesalahan parsing JSON, pesan kesalahan ditampilkan menggunakan Toast. Tombol **btnToHome** diatur untuk memulai **HomeActivity** saat diklik. Jika permintaan API gagal (onFailure), status kode dan pesan kesalahan ditampilkan menggunakan Toast. Proses ini juga melibatkan pengelolaan visibilitas **ProgressBar** untuk memberikan umpan balik visual kepada pengguna selama operasi jaringan berlangsung.

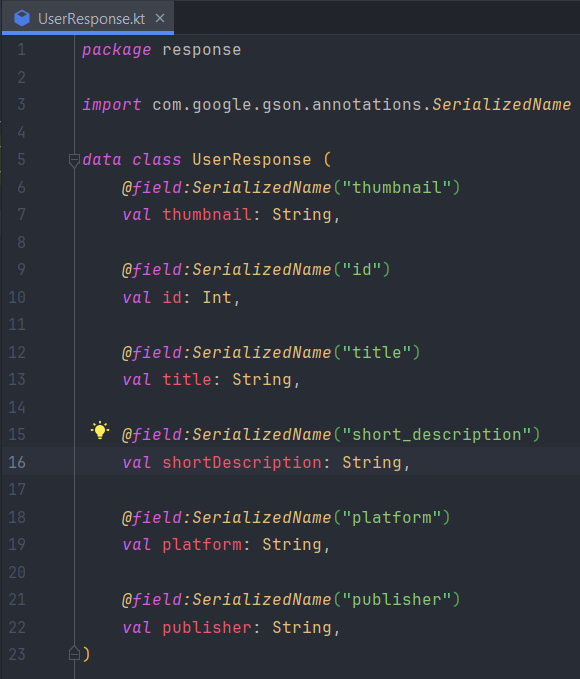
1. **activity\_main.xml.**



*Gambar 2 activity\_main.xml*

Kode XML di atas mendefinisikan tata letak untuk **MainActivity** menggunakan **LinearLayout** dengan orientasi vertikal. Layout ini berisi beberapa elemen UI yang disusun secara vertikal. **ImageView** dengan ID **tvImg** ditampilkan di tengah layar dengan ukuran **300dp x 180dp** untuk menampilkan gambar. Di bawahnya, ada **ProgressBar** dengan ID **progressBar** yang juga ditampilkan di tengah untuk menunjukkan proses memuat API. Selanjutnya, dua TextView masing-masing dengan ID **tvTitle** dan **tvDescription** digunakan untuk menampilkan judul dan deskripsi singkat, dengan teks yang diselaraskan di tengah dan menggunakan gaya italic serta ukuran teks 19sp. Di bagian bawah, ada sebuah tombol **AppCompatButton** dengan ID **btnToHome** digunakan untuk memulai aktivitas halaman selanjutnya saat diklik.

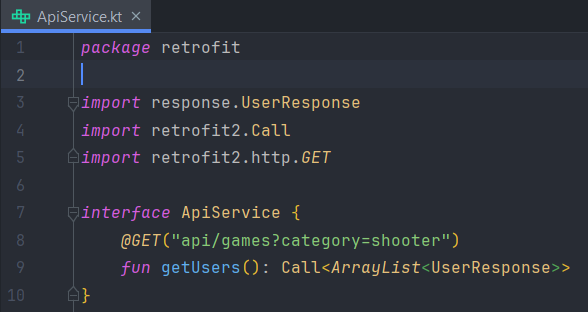
1. **Pembuatan Tampilan List RecyclerView API menggunakan Retrofit (Home Activity / Halaman 2).**
2. **Pembuatan UserResponse.kt pada package response.**

****

*Gambar 3 UserResponse.kt*

Kode di atas digunakan untuk mengonversi data JSON menjadi objek Kotlin. Data class bernama **UserResponse** ini memiliki enam property yaitu **thumbnail**, **id**, **title**, **shortDescription**, **platform**, dan **publisher**. Terdapat 2 nilai yang diambil disini yaitu String dan Int. Masing-masing properti dalam data class ini memiliki anotasi **@SerializedName** yang digunakan untuk menentukan nama key pada JSON yang sesuai dengan properti tersebut. Anotasi ini memastikan bahwa saat data JSON diambil, nilai-nilai dari key yang relevan akan dipetakan ke properti yang sesuai dalam objek Kotlin. Contohnya, **@SerializedName("thumbnail")** mengindikasikan bahwa nilai dari key **"thumbnail"** pada JSON akan diambil dan disimpan dalam properti thumbnail di dalam objek **UserResponse**. Data class ini efektif dalam memetakan data JSON yang diterima dari suatu layanan atau API ke dalam struktur data yang bisa digunakan dalam aplikasi Kotlin.

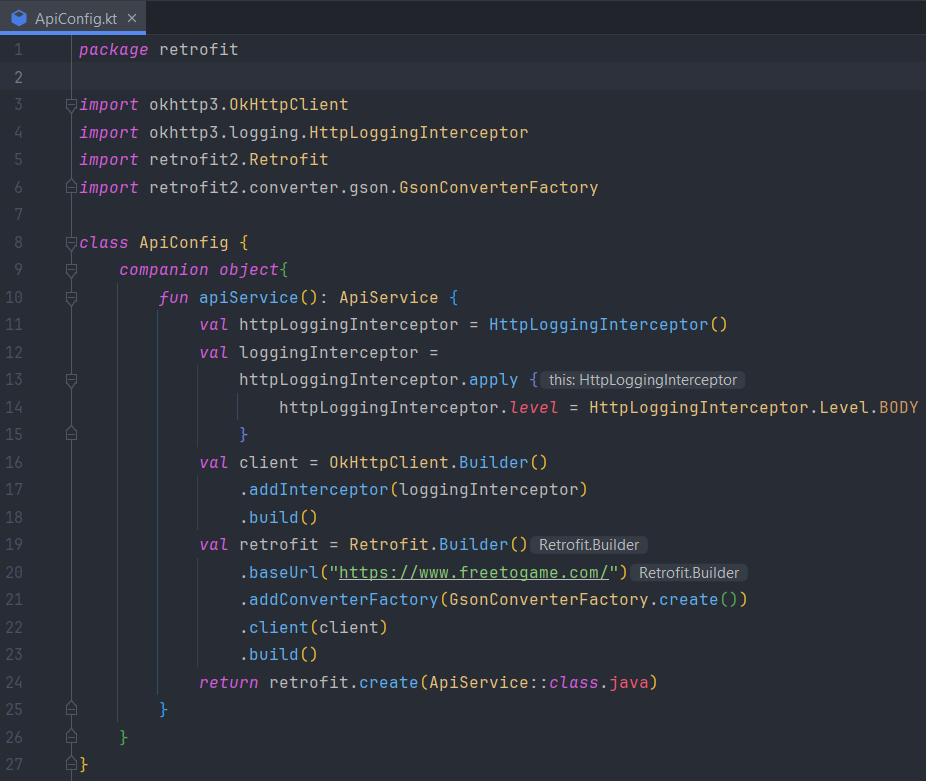
1. **Pembuatan ApiService.kt pada package retrofit.**

****

*Gambar 4 ApiService.kt*

Kode di atas mendefinisikan sebuah interface bernama **ApiService** yang digunakan untuk mendeklarasikan service **HTTP** menggunakan **library Retrofit**. Interface ini berisi sebuah metode **getUsers()** yang ditandai dengan anotasi **@GET** untuk melakukan permintaan **HTTP GET** ke endpoint **"api/games?category=shooter"**. Metode **getUsers()** mengembalikan objek **Call<ArrayList<UserResponse>>** yang merupakan representasi dari permintaan HTTP tersebut. Objek Call ini akan digunakan untuk menjalankan permintaan dan menerima respon dari server dalam bentuk daftar objek UserResponse yang dikemas dalam ArrayList. Dengan kata lain, saat **getUsers()** dipanggil, Retrofit akan mengirimkan permintaan GET ke endpoint yang ditentukan, dan setelah mendapatkan respon dari server, Retrofit akan mengonversi data JSON yang diterima menjadi objek UserResponse dan mengembalikannya dalam bentuk daftar atau array list.

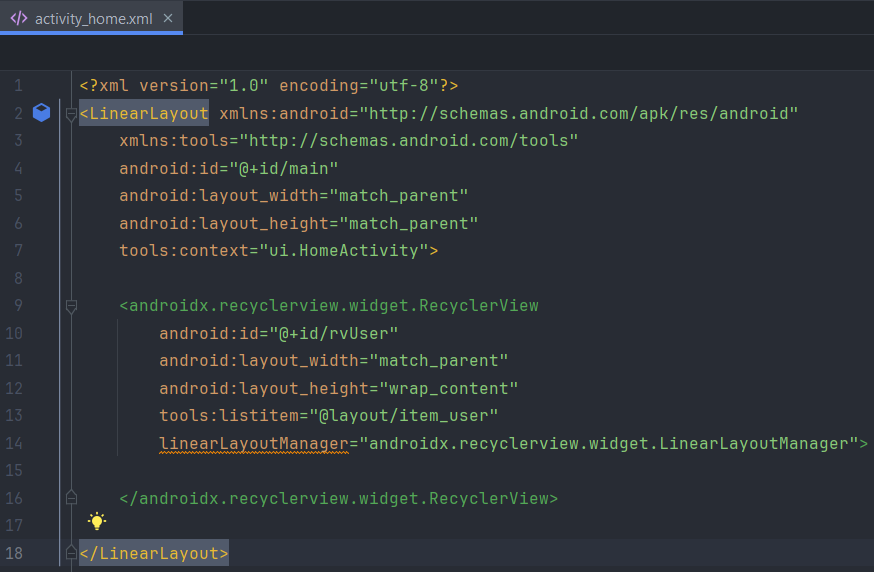
1. **Pembuatan ApiConfig.kt pada package retrofit.**

****

*Gambar 5 ApiConfig.kt*

Kelas **ApiConfig** diatas adalah utilitas untuk mengkonfigurasi **Retrofit** dalam proyek Android. Kelas ini memiliki companion object dengan fungsi **apiService()** yang mengembalikan objek **ApiService**. Fungsi ini membuat instance **HttpLoggingInterceptor** untuk mencatat **HTTP** **request** dan **response** pada level BODY, lalu menambahkannya ke **OkHttpClient**. Selanjutnya, **Retrofit.Builder()** digunakan untuk mengatur base URL **("https://www.freetogame.com/")**, menambahkan **GsonConverterFactory** untuk mengonversi JSON ke objek Java, dan menetapkan **OkHttpClient** yang sudah dikonfigurasi. Fungsi ini akhirnya mengembalikan instance **ApiService** yang siap digunakan untuk melakukan request ke API. Dengan **ApiConfig**, konfigurasi **Retrofit** menjadi lebih efektif dan reusable, cukup memanggil fungsi **apiService()** untuk mendapatkan instance **ApiService**.

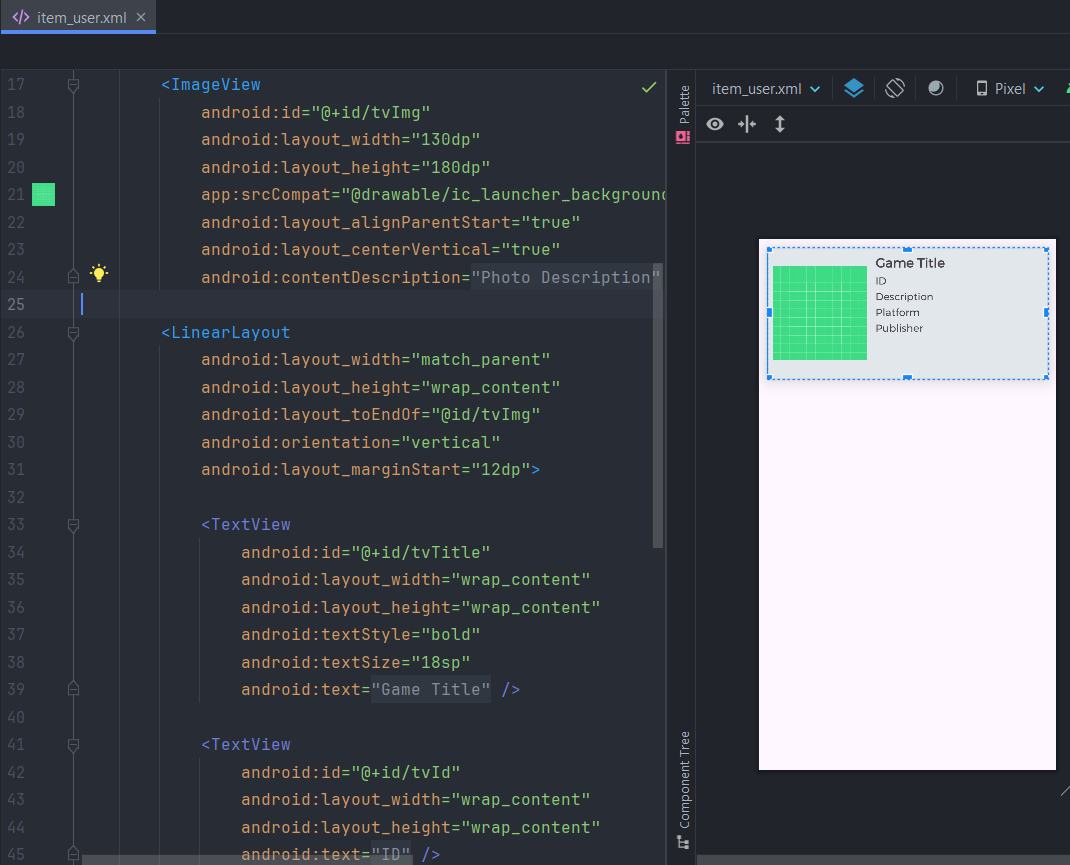
1. **Mengatur layout RecyclerView pada activity\_home.xml.**

****

*Gambar 6 activity\_home.xml*

Kode di atas adalah file layout XML untuk aktivitas pembuatan daftar API pada **HomeActivity.kt** menggunakan **LinearLayout** sebagai elemen root yang mengatur tata letak secara linier. Di dalam LinearLayout ini terdapat sebuah **RecyclerView** dengan ID **@+id/rvUser**, yang berfungsi untuk menampilkan daftar data secara efisien menggunakan **ViewHolder**.

1. **Pembuatan layout RecyclerView dengan file baru bernama item\_user.xml.**

****

*Gambar 7 item\_row.xml*

Kode **item\_row.xml** di atas mendefinisikan tampilan sebuah item dalam bentuk kartu **CardView** yang akan digunakan dalam sebuah **RecyclerView** untuk menampilkan data yang diambil menggunakan **Retrofit** di dalam **HomeActivity.kt**. Di dalam **CardView**, terdapat **RelativeLayout** yang mengatur tata letak anak-anaknya. Sebuah **ImageView** dengan id **tvImg** menampilkan gambar di sisi kiri, dan LinearLayout vertikal di sebelah kanannya menampung beberapa **TextView** dengan id **tvTitle**, **tvId**, **tvDescription**, **tvPlatform**, dan **tvPublisher**, yang masing-masing menampilkan informasi teks yang berbeda. Untuk menghubungkannya dengan **Retrofit** di **HomeActivity.kt**, data yang diambil dari API akan dimasukkan ke dalam model data, kemudian adapter **RecyclerView** akan diisi dengan model ini dan menampilkan setiap item data dalam bentuk yang didefinisikan oleh **item\_row.xml**, sehingga setiap data dari API akan memiliki tampilan kartu yang konsisten dengan desain ini.

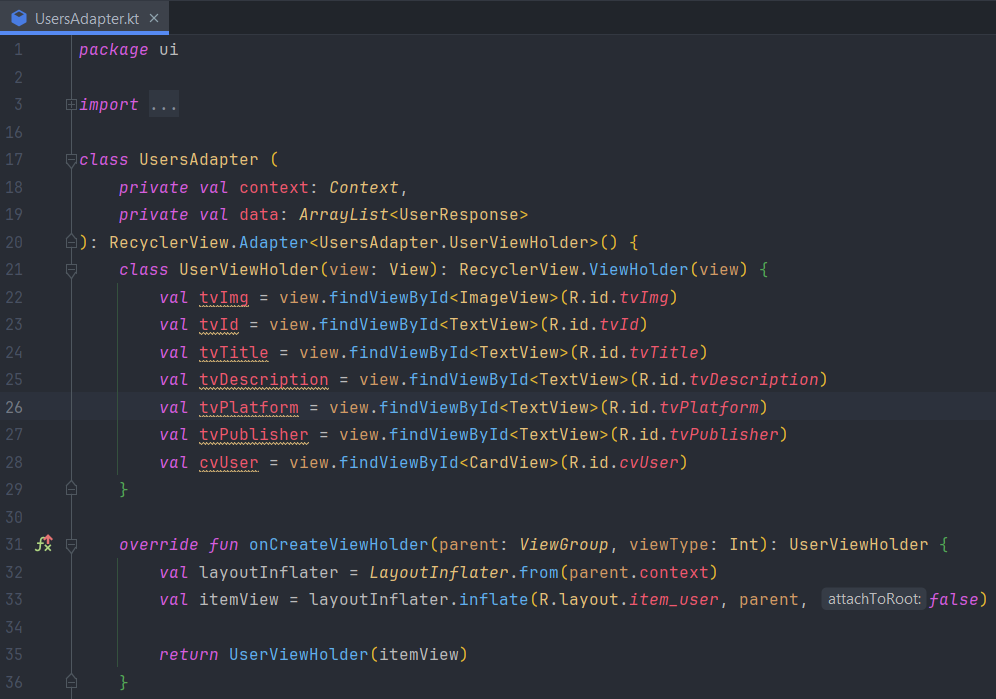
1. **Pengaturan String yang digunakan pada strings.xml**

****

*Gambar 8 strings.xml*

Kode **strings.xml** diatas merupakan beberapa strings yang dipakai pada aplikasi ini yang sering dipanggil pada setiap xml nya.

1. **Membuat kelas UsersAdapter.kt untuk menyambungkan RecyclerView pada package ui.**

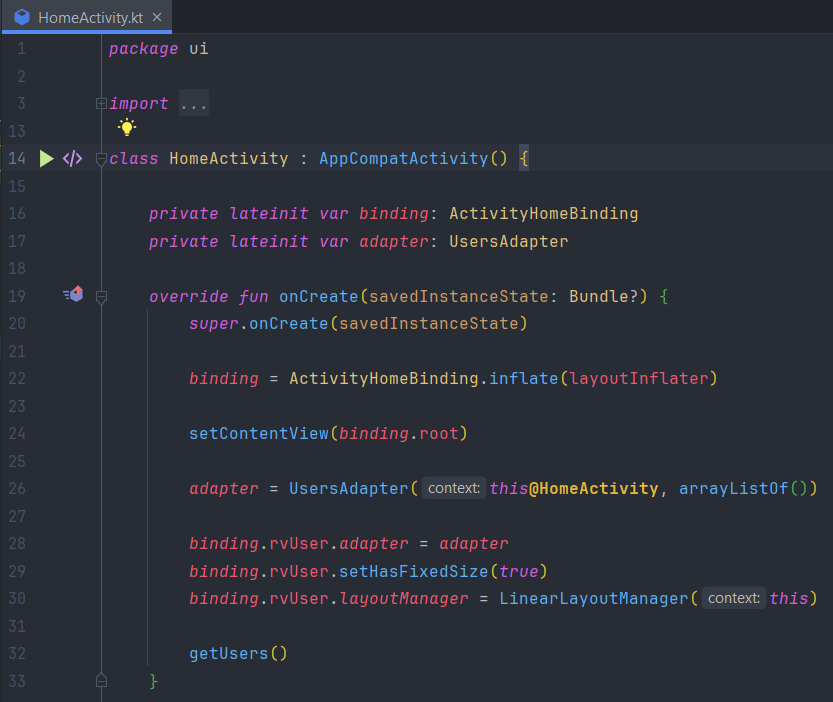
****

****

*Gambar 9 UsersAdapter.kt*

Kelas **UsersAdapter** adalah sebuah adapter yang digunakan untuk menampilkan data **UserResponse** dalam sebuah **RecyclerView**. Kelas ini menerima **Context** dan sebuah **ArrayList** dari UserResponse sebagai parameter konstruktor. UsersAdapter mengimplementasikan **RecyclerView.Adapter<UsersAdapter.UserViewHolder>()**, yang memiliki inner class **UserViewHolder** untuk mendefinisikan tampilan setiap item di dalam RecyclerView. Metode **onCreateViewHolder** digunakan untuk membuat ViewHolder baru berdasarkan layout **item\_user**. Metode **onBindViewHolder** digunakan untuk mengikat data dari **ArrayList** data ke **ViewHolder**, dengan menggunakan **Picasso** untuk memuat gambar dari URL ke dalam **ImageView**, dan mengatur teks dari beberapa **TextView** dengan data yang sesuai. **holder.cvUser.setOnClickListener** ditambahkan untuk menampilkan toast dengan judul item ketika item diklik. Metode **getItemCount** mengembalikan jumlah item dalam ArrayList data. Metode **setData** digunakan untuk mengganti data di dalam adapter dengan data baru, kemudian memberi tahu adapter bahwa dataset telah berubah sehingga RecyclerView dapat diperbarui dengan memanggil notifyDataSetChanged(). Kelas ini mengatur bagaimana data UserResponse ditampilkan dalam RecyclerView.

1. **Mengambil data Retrofit dan menampilkannya di HomeActivity.kt**

****

****

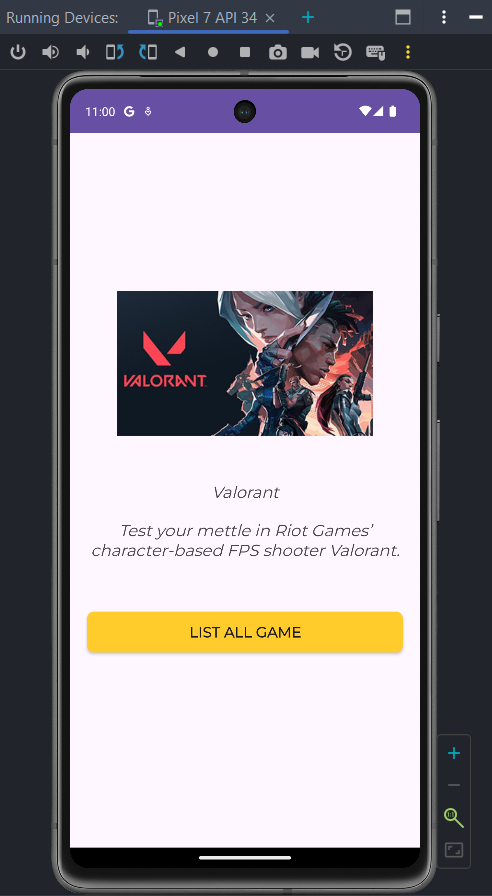
*Gambar 10 HomeActivity.kt*

Pada kelas **HomeActivity**, **ActivityHomeBinding** digunakan untuk menginflate layout dari activity home. **onCreate** menginisialisasi **UsersAdapter** dengan konteks **HomeActivity** dan **array list** kosong, lalu mengatur RecyclerView dengan adapter tersebut, memastikan ukuran tetap dan menggunakan **LinearLayoutManager**. **Method getUsers** melakukan request **API** untuk mengambil data pengguna menggunakan **Retrofit**, dan jika responsenya berhasil, data pengguna diteruskan ke **method setData**. **Method setData** mengupdate data pada adapter dan memberitahu adapter bahwa dataset telah berubah, sehingga RecyclerView dapat diperbarui untuk menampilkan data pengguna yang baru diambil. Jika request gagal, kesalahan dicatat dengan **Log.d**. Kelas ini menggunakan **Retrofit** untuk mengakses API dan menampilkan data pengguna dalam RecyclerView melalui **UsersAdapter**.

1. **Screenshot Terminal**

Berikut merupakan hasil dari praktikum PAB ke-9 yang menerapkan materi Retrofit dan LoopJ yang berisi RecyclerView. Berikut merupakan hasil pada pekerjaan milik saya dengan tema **”Free To Game”** dari link <https://www.freetogame.com/api/games?category=shooter>

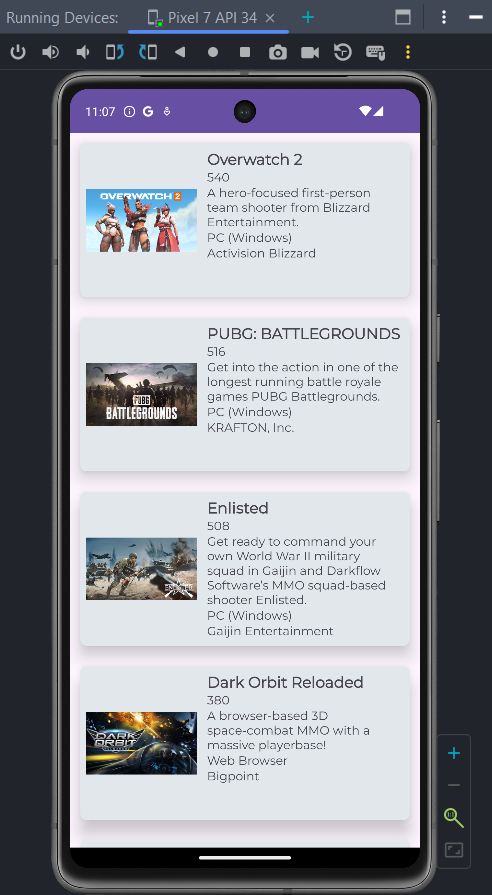
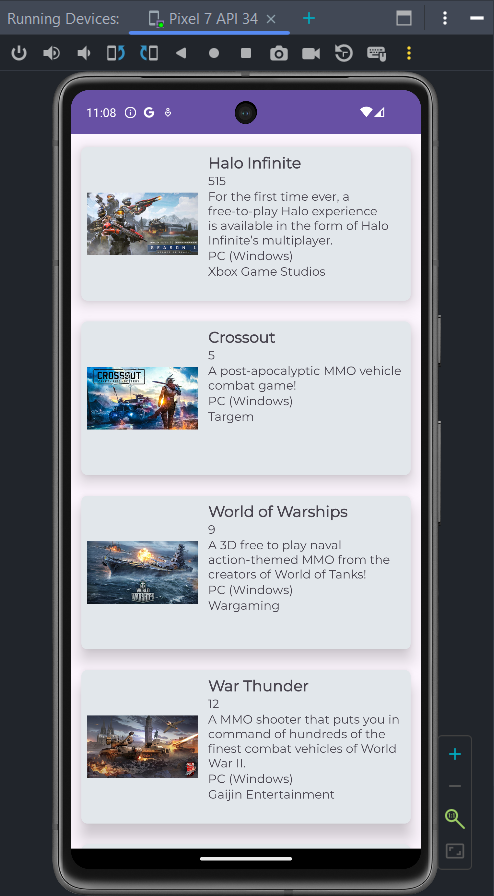
1. **Halaman Awal (Main Activity / Halaman 1)**

****

*Gambar 11 Hasil Tampilan Awal (Main Activity)*

Gambar di atas menampilkan hasil dari emulator setelah aplikasi dijalankan. Halaman awal **MainActivity** ini menampilkan salah satu data dari API yang mengambil elemen objek ke-10, yaitu data tentang game **Valorant**. API yang dipanggil hanya menampilkan **thumbnail**, **judul (title),** dan **deskripsi singkat (short description)**. Di bawah deskripsi singkat terdapat tombol atau **button "LIST ALL GAME"** yang jika diklik akan menuju ke halaman berikutnya, yaitu **HomeActivity**.

1. **Halaman Kedua (Home Activity / Halaman 2)**

*Gambar 12 Hasil Tampilan Kedua (Home Activity)*

Gambar di atas menampilkan hasil dari emulator setelah aplikasi dijalankan. Halaman kedua, yaitu **HomeActivity** yang menampilkan **RecyclerView** yang berisi semua data dari public API. Data yang diambil dari API memiliki enam property yaitu **thumbnail, id, title, shortDescription, platform,** dan **publisher**. Pengguna dapat menggunakan aplikasi ini dengan efisien karena dapat melihat banyak data dari API secara rapi dalam satu aplikasi.

1. **Kesimpulan**

Laporan praktikum ini mendemonstrasikan penggunaan **Retrofit** dan **LoopJ** dalam mengkonsumsi public API pada aplikasi Android. **MainActivity** memanfaatkan **LoopJ** dan **Picasso** untuk menampilkan data game spesifik dari **API FreeToGame** dengan elemen UI yang dinamis, sementara **HomeActivity** menggunakan **Retrofit** dan **RecyclerView** untuk menampilkan daftar lengkap data game secara efisien. Implementasi ini menunjukkan kemampuan aplikasi untuk mengelola dan menampilkan data dari API eksternal dengan baik, memberikan pengalaman pengguna yang interaktif dan responsif.