

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.

A. Pembuatan Tampilan Utama (Main Activity / Halaman 1).

a. MainActivity.kt.

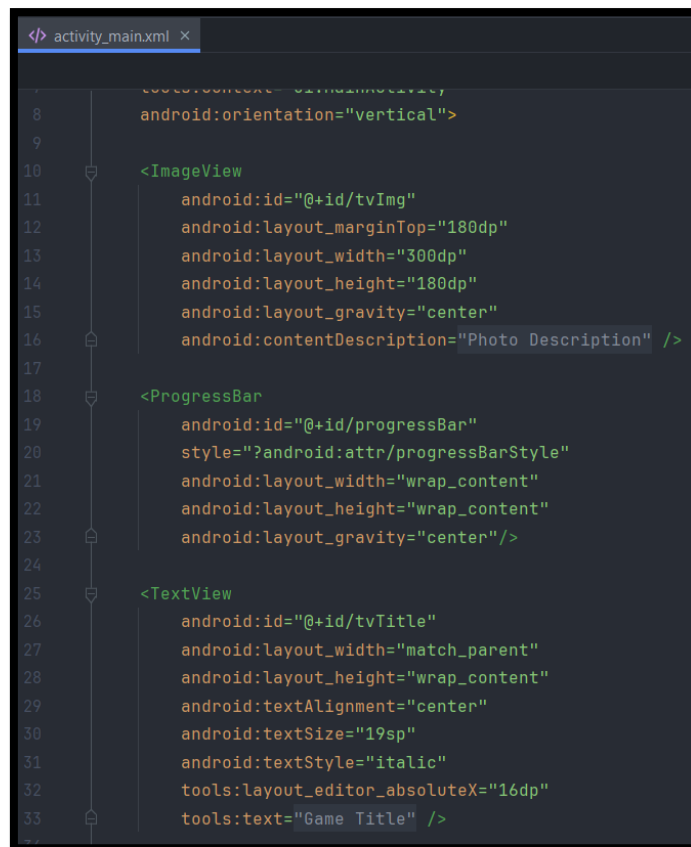
```
1 package ui
2
3 import ...
4
5
6 class MainActivity : AppCompatActivity() {
7     companion object {
8         private val TAG = MainActivity::class.java.simpleName
9     }
10
11     private lateinit var binding: ActivityMainBinding
12
13     override fun onCreate(savedInstanceState: Bundle?) {
14         super.onCreate(savedInstanceState)
15         binding = ActivityMainBinding.inflate(layoutInflater)
16         setContentView(binding.root)
17         getRandomBody()
18     }
19
20     private fun getRandomBody() {
21         binding.progressBar.visibility = View.VISIBLE
22         val client = AsyncHttpClient()
23         val url = "https://www.freetogame.com/api/games?category=shooter"
24         client.get(url, object : AsyncHttpResponseHandler() {
25             override fun onSuccess(statusCode: Int, headers: Array<Header>, responseBody: ByteArray) {
26                 // jika koneksi berhasil
27                 binding.progressBar.visibility = View.INVISIBLE
28                 val result = String(responseBody)
29                 Log.d(TAG, result)
30                 try {
31                     val jsonArray = JSONArray(result)
32                     val jsonObject = jsonArray.getJSONObject(index: 9)
33                     val thumbnail = jsonObject.getString(name: "thumbnail")
34
35                     val title = jsonObject.getString(name: "title")
36                     val shortDescription = jsonObject.getString(name: "short_description")
37                     Picasso.get().load(thumbnail).into(binding.tvImg)
38                     binding.tvTitle.text = title
39                     binding.tvDescription.text = shortDescription
40
41                 } catch (e: Exception) {
42                     Toast.makeText(context: this@MainActivity, e.message, Toast.LENGTH_SHORT).show()
43                     e.printStackTrace()
44                 }
45
46                 binding.btnToHome.setOnClickListener { it: View?
47                     startActivity(Intent( packageContext: this@MainActivity, HomeActivity::class.java))
48                 }
49             }
50             override fun onFailure(statusCode: Int, headers: Array<Header>, responseBody: ByteArray, error: Throwable) {
51                 // jika koneksi gagal
52                 binding.progressBar.visibility = View.INVISIBLE
53                 val errorMessage = when (statusCode) {
54                     401 -> "$statusCode : Bad Request"
55                     403 -> "$statusCode : Forbidden"
56                     404 -> "$statusCode : Not Found"
57                     else -> "$statusCode : ${error.message}"
58                 }
59                 Toast.makeText(context: this@MainActivity, errorMessage, Toast.LENGTH_SHORT).show()
60             }
61         })
62     }
63 }
```

```
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
```

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.

b. activity_main.xml.



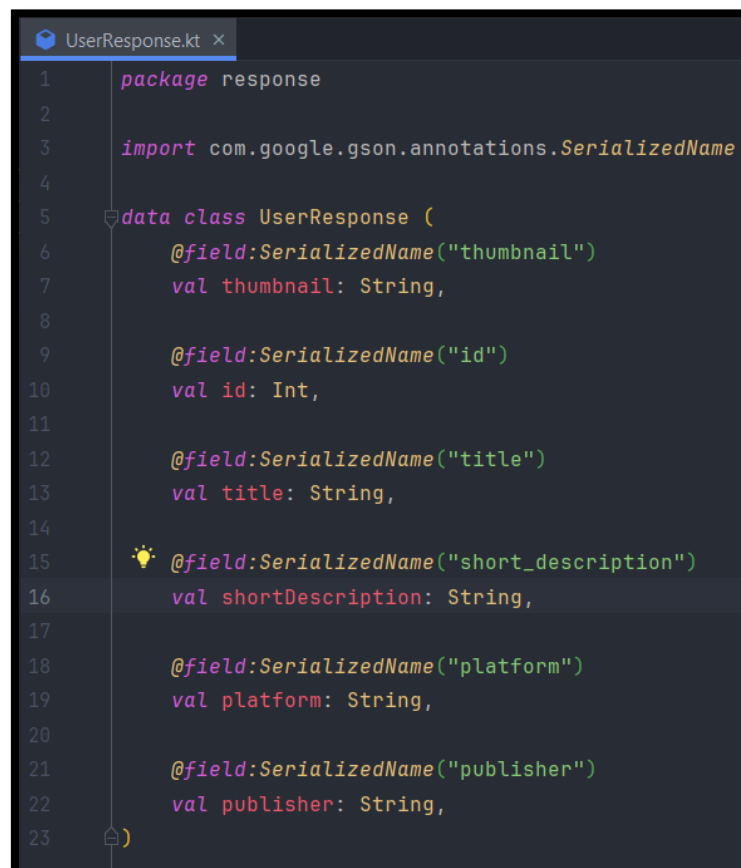
Gambar 2 activity_main.xml

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.

B. Pembuatan Tampilan List RecyclerView API menggunakan Retrofit (Home Activity / Halaman 2).

a. Pembuatan **UserResponse.kt** pada package response.



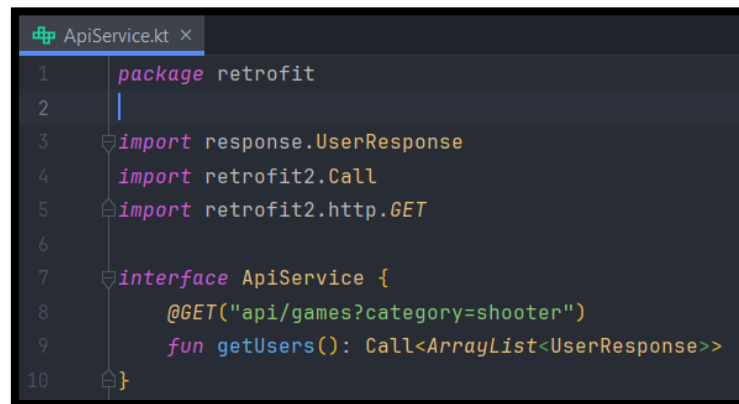
```
1 package response
2
3 import com.google.gson.annotations.SerializedName
4
5 data class UserResponse (
6     @SerializedName("thumbnail")
7     val thumbnail: String,
8
9     @SerializedName("id")
10    val id: Int,
11
12    @SerializedName("title")
13    val title: String,
14
15    @SerializedName("short_description")
16    val shortDescription: String,
17
18    @SerializedName("platform")
19    val platform: String,
20
21    @SerializedName("publisher")
22    val publisher: String,
23 )
```

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.

b. Pembuatan ApiService.kt pada package retrofit.



```
1 package retrofit
2
3 import response.UserResponse
4 import retrofit2.Call
5 import retrofit2.http.GET
6
7 interface ApiService {
8     @GET("api/games?category=shooter")
9     fun getUsers(): Call<ArrayList<UserResponse>>
10 }
```

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.

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

```
1 package retrofit
2
3 import okhttp3.OkHttpClient
4 import okhttp3.logging.HttpLoggingInterceptor
5 import retrofit2.Retrofit
6 import retrofit2.converter.gson.GsonConverterFactory
7
8 class ApiConfig {
9     companion object {
10         fun apiService(): ApiService {
11             val httpLoggingInterceptor = HttpLoggingInterceptor()
12             val loggingInterceptor =
13                 httpLoggingInterceptor.apply { this: HttpLoggingInterceptor
14                     httpLoggingInterceptor.level = HttpLoggingInterceptor.Level.BODY
15                 }
16             val client = OkHttpClient.Builder()
17                 .addInterceptor(loggingInterceptor)
18                 .build()
19             val retrofit = Retrofit.Builder() Retrofit.Builder
20                 .baseUrl("https://www.freetogame.com/") Retrofit.Builder
21                 .addConverterFactory(GsonConverterFactory.create())
22                 .client(client)
23                 .build()
24             return retrofit.create(ApiService::class.java)
25         }
26     }
27 }
```

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**.

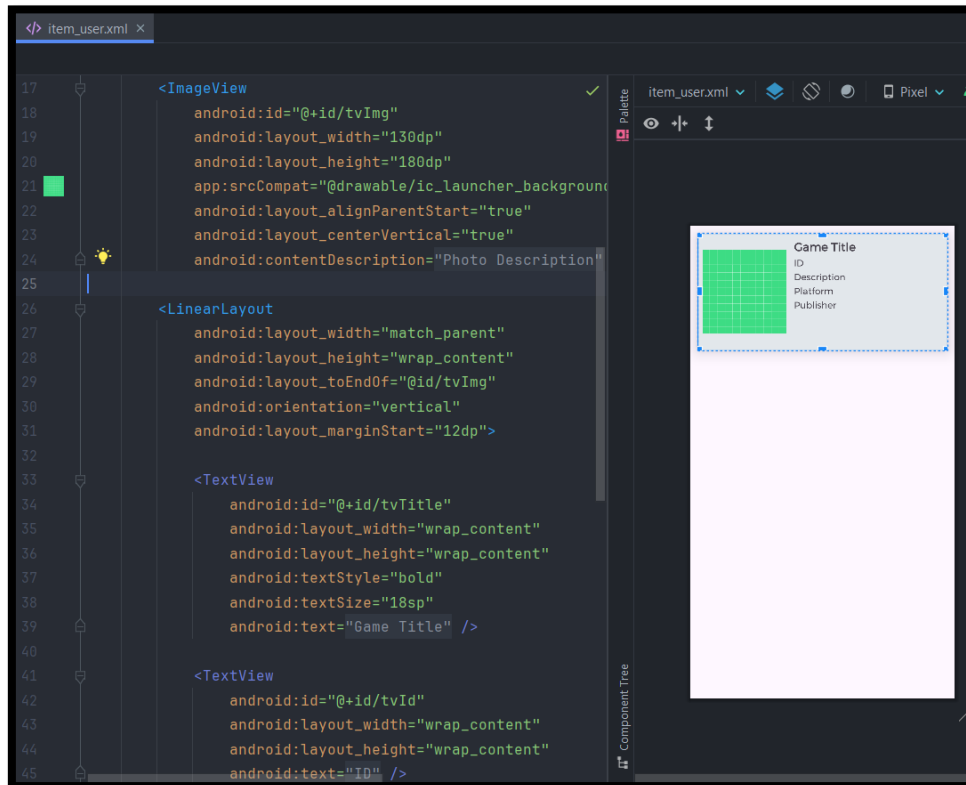
d. Mengatur layout **RecyclerView** pada **activity_home.xml**.

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3     xmlns:tools="http://schemas.android.com/tools"
4     android:id="@+id/main"
5     android:layout_width="match_parent"
6     android:layout_height="match_parent"
7     tools:context="ui.HomeActivity">
8
9     <androidx.recyclerview.widget.RecyclerView
10         android:id="@+id/rvUser"
11         android:layout_width="match_parent"
12         android:layout_height="wrap_content"
13         tools:listitem="@layout/item_user"
14         layoutManager="androidx.recyclerview.widget.LinearLayoutManager">
15
16     </androidx.recyclerview.widget.RecyclerView>
17
18 </LinearLayout>
```

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**.

e. 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.

f. Pengaturan String yang digunakan pada strings.xml

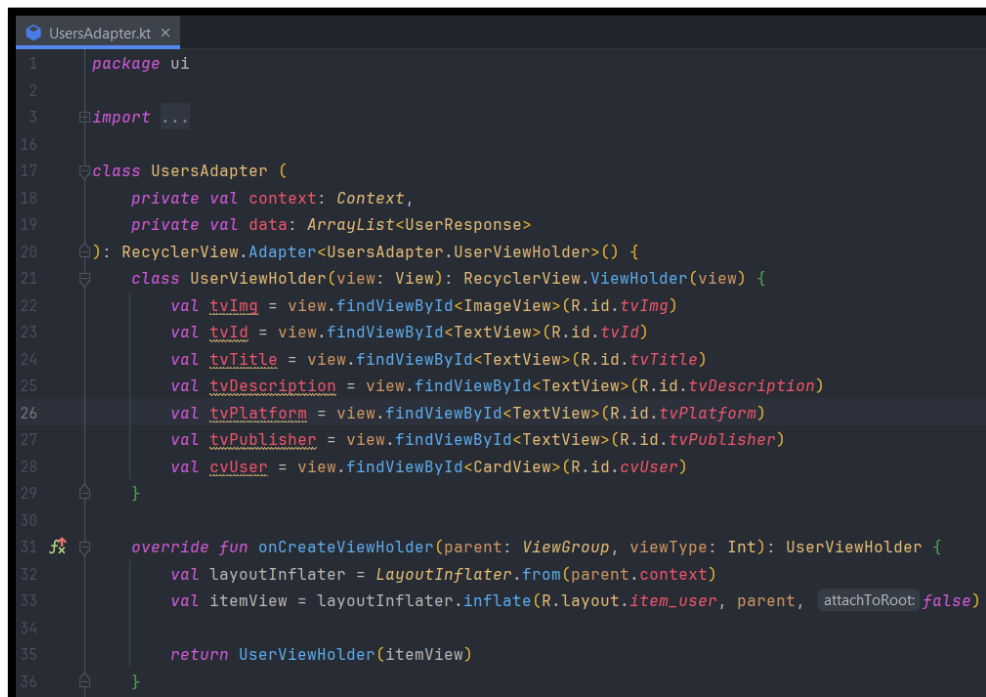


```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">PPAB-09_L0122012_Alfath Roziq Widhayaka</string>
    <string name="titleHome">Free Games</string>
    <string name="id">ID</string>
    <string name="title">Game Title</string>
    <string name="short_description">Description</string>
    <string name="genre">Genre</string>
    <string name="platform">Platform</string>
    <string name="publisher">Publisher</string>
    <string name="desc_photo">Photo Description</string>
    <string name="listGame">List All Game</string>
</resources>
```

Gambar 8 strings.xml

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

g. Membuat kelas UsersAdapter.kt untuk menyambungkan RecyclerView pada package ui.



```
package ui

import ...

class UsersAdapter (
    private val context: Context,
    private val data: ArrayList<UserResponse>
): RecyclerView.Adapter<UsersAdapter.UserViewHolder>() {
    class UserViewHolder(view: View): RecyclerView.ViewHolder(view) {
        val tvImg = view.findViewById<ImageView>(R.id.tvImg)
        val tvId = view.findViewById<TextView>(R.id.tvId)
        val tvTitle = view.findViewById<TextView>(R.id.tvTitle)
        val tvDescription = view.findViewById<TextView>(R.id.tvDescription)
        val tvPlatform = view.findViewById<TextView>(R.id.tvPlatform)
        val tvPublisher = view.findViewById<TextView>(R.id.tvPublisher)
        val cvUser = view.findViewById<CardView>(R.id.cvUser)
    }

    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): UserViewHolder {
        val inflater = LayoutInflater.from(parent.context)
        val itemView = inflater.inflate(R.layout.item_user, parent, attachToRoot: false)

        return UserViewHolder(itemView)
    }
}
```



```

38  override fun onBindViewHolder(holder: ViewHolder, position: Int) {
39      Picasso.get().load(data.get(position).thumbnail).into(holder.tvImg)
40
41      holder.tvId.text = data.get(position).id.toString()
42      holder.tvTitle.text = data.get(position).title
43      holder.tvDescription.text = data.get(position).shortDescription
44      holder.tvPlatform.text = data.get(position).platform
45      holder.tvPublisher.text = data.get(position).publisher
46      holder.cvUser.setOnClickListener { it: View!
47          Toast.makeText(context, text: "" + data.get(position).title, Toast.LENGTH_SHORT).show()
48      }
49  }
50
51  }
52
53  override fun getItemCount(): Int = data.size
54
55  @SuppressWarnings("NotifyDataSetChanged")
56  fun setData(newData: ArrayList<UserResponse>) {
57      data.clear()
58      data.addAll(newData)
59      notifyDataSetChanged()
60  }
61  }

```

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.ViewHolder>()**, yang memiliki inner class **ViewHolder** 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**.

h. Mengambil data Retrofit dan menampilkannya di HomeActivity.kt



```
1 package ui
2
3 import ...
4
13
14 class HomeActivity : AppCompatActivity() {
15
16     private lateinit var binding: ActivityHomeBinding
17     private lateinit var adapter: UsersAdapter
18
19     override fun onCreate(savedInstanceState: Bundle?) {
20         super.onCreate(savedInstanceState)
21
22         binding = ActivityHomeBinding.inflate(layoutInflater)
23
24         setContentView(binding.root)
25
26         adapter = UsersAdapter(context = this@HomeActivity, arrayListOf())
27
28         binding.rvUser.adapter = adapter
29         binding.rvUser.setHasFixedSize(true)
30         binding.rvUser.layoutManager = LinearLayoutManager(context = this)
31
32         getUsers()
33     }
34
35     private fun getUsers() {
36         ApiConfig.apiService().getUsers().enqueue(object : Callback<ArrayList<UserResponse>> {
37             override fun onResponse(
38                 call: Call<ArrayList<UserResponse>>,
39                 response: Response<ArrayList<UserResponse>>
40             ) {
41                 if (response.isSuccessful) {
42                     val data = response.body()
43                     data?.let { it: ArrayList<UserResponse> ->
44                         setData(it)
45                     }
46                 }
47             }
48
49             override fun onFailure(call: Call<ArrayList<UserResponse>>, t: Throwable) {
50                 Log.d(tag = "Error", msg = "" + t.stackTraceToString())
51             }
52         })
53     }
54
55     fun setData(data: ArrayList<UserResponse>) {
56         adapter.setData(data)
57     }
58 }
```

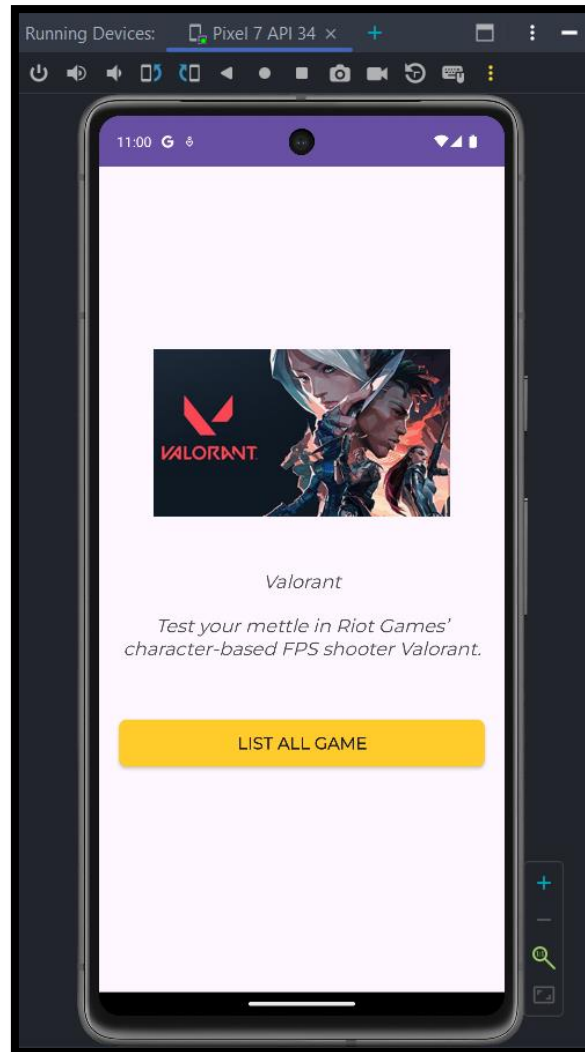
Gambar 10 HomeActivity.kt

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 responsnya 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**.

2. 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>

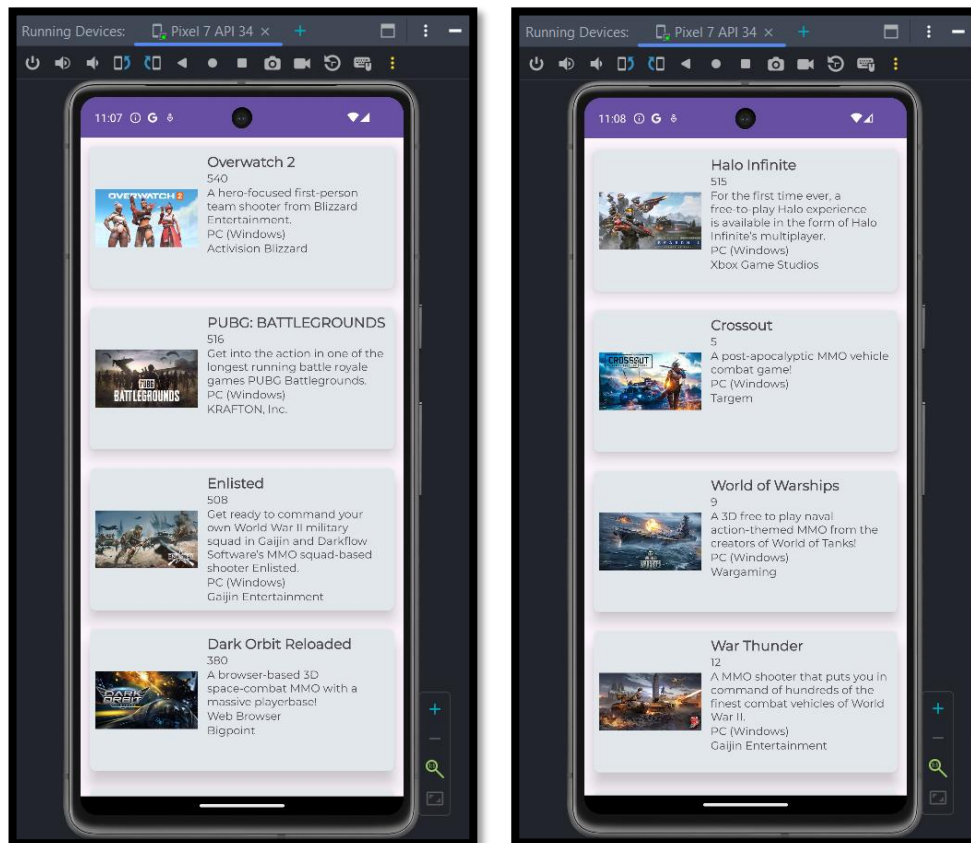
A. Halaman Awal (Main Activity / Halaman 1)



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**.

B. 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.

3. 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.