Modul 6 - Retrofit

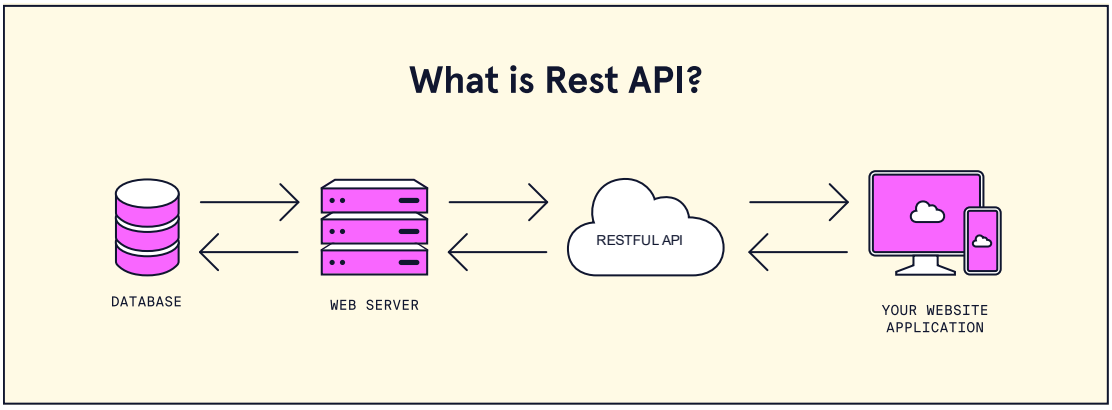
API (Application Programming Interface) adalah antarmuka yang berfungsi sebagai penghubung antara sebuah aplikasi dan aplikasi lainnya, atau antara aplikasi client dan server tempat penyimpanan data, sehingga memungkinan bagi aplikasi client untuk menambahkan data ke server tanpa harus mengakses servernya secara langsung di dalam aplikasi yang dibuat.

API bisa diibaratkan seperti sebuah restoran. Ketika kamu pergi ke restoran, kamu memiliki menu pilihan untuk dipilih - seperti burger, pizza, atau salad. Kamu engga perlu tahu bagaimana hidangan-hidangan ini disiapkan atau apa resepnya, kamu hanya perlu tahu apa yang ingin kamu pesan. Begitu juga dengan API, kamu hanya perlu tahu bagaimana menggunakannya tanpa perlu memahami detail teknis di baliknya. Fungsi dari API adalah agar beberapa aplikasi yang berbeda bisa terhubung dan berinteraksi, dan juga sebagai jembatan antara data yang berasal dari server dan juga aplikasi yang kita gunakan.

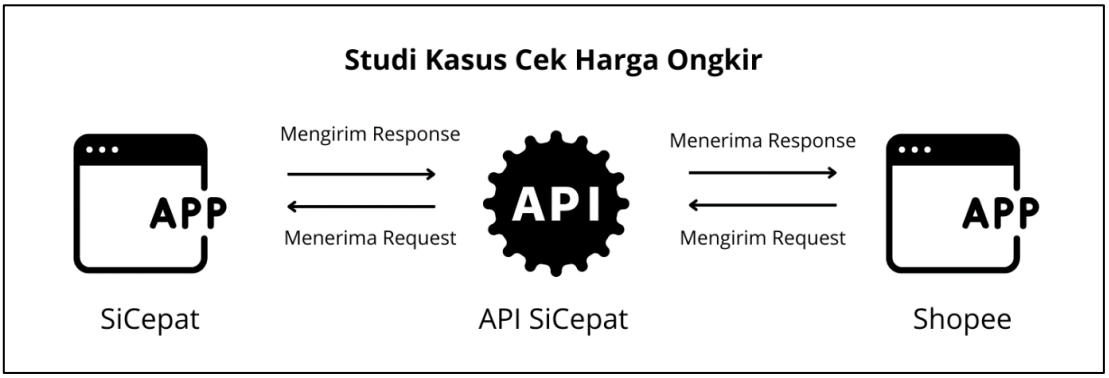
Dalam membuat API, terdapat protokol yang digunakan seperti SOAP, REST, JSON-RPC, gRPC, GraphQL, XML-RPC, Apache Thrift. Kali ini API Protocol yang akan kita bahas lebih mendalam adalah REST API.

REST (Representational State Transfer) adalah serangkaian aturan dan pedoman yang diikuti oleh programmer ketika mereka membuat API berbasis REST. API yang menggunakan protokol REST disebut REST API. REST mememiliki 4 komponen yang digunakan di dalam API, yaitu GET, POST, PUT, DELETE.

1. GET (Read): GET digunakan untuk mendapatkan data.
2. POST (Add): POST digunakan untuk menambahkan data.
3. PUT (Update): PUT digunakan untuk memperbarui data.
4. DELETE (Delete): DELETE digunakan untuk menghapus data.



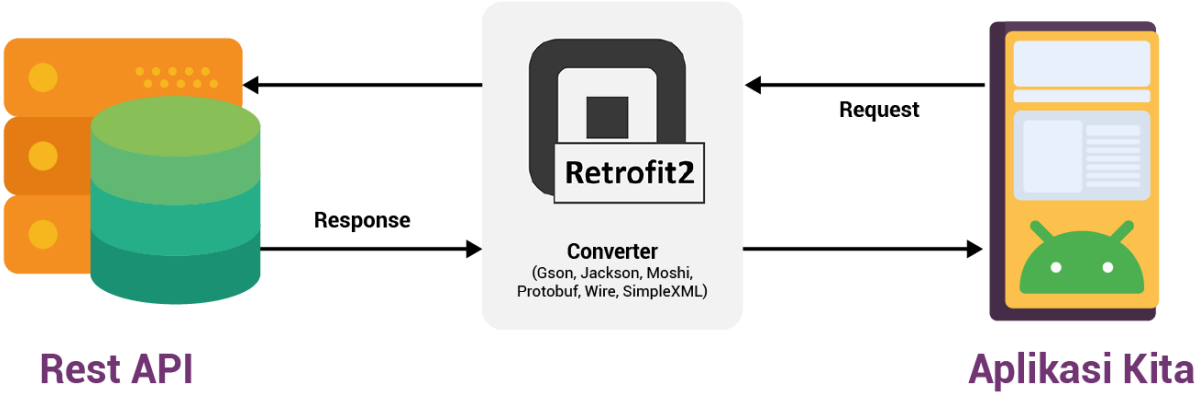
Gambar 1. Cara Kerja Rest API



Gambar 2. Cara Kerja API untuk menghubungkan 2 Aplikasi yang berbeda

Retrofit

Retrofit adalah satu library buatan Square yang populer digunakan untuk melakukan Networking ke Web API. Dengan menggunakan library Retrofit, aplikasi android kita bisa terhubung dan berinteraksi dengan API yang tersedia.

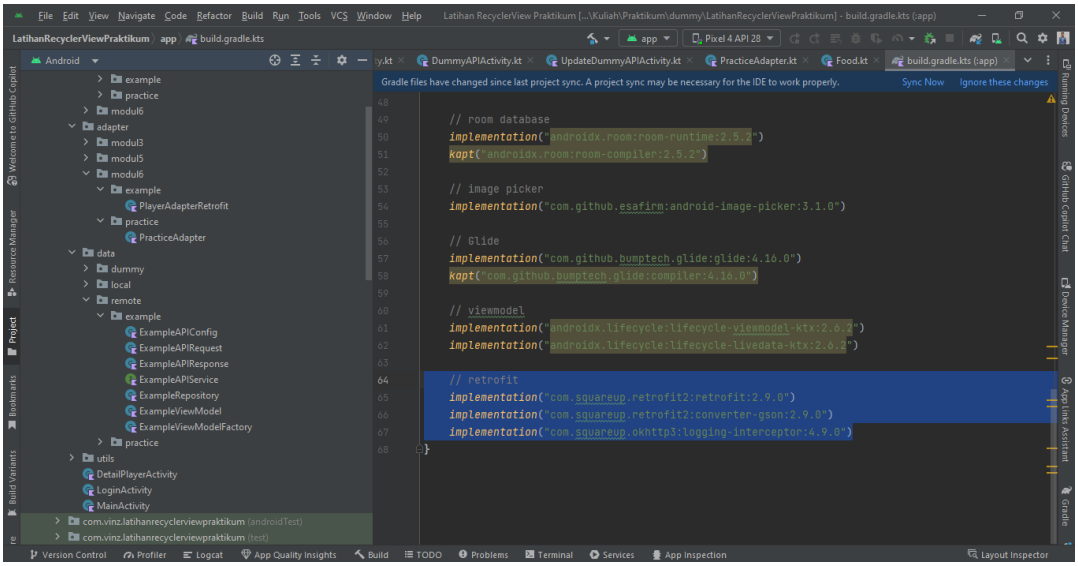


Gambar 3. Cara Kerja Library Retrofit

Ngoding Time

1. Tambahkan library Retrofit di build.gradle.kts (module: app), setelah itu klik sync project dan tunggu sebentar sampai project selesai melakukan sinkronisasi

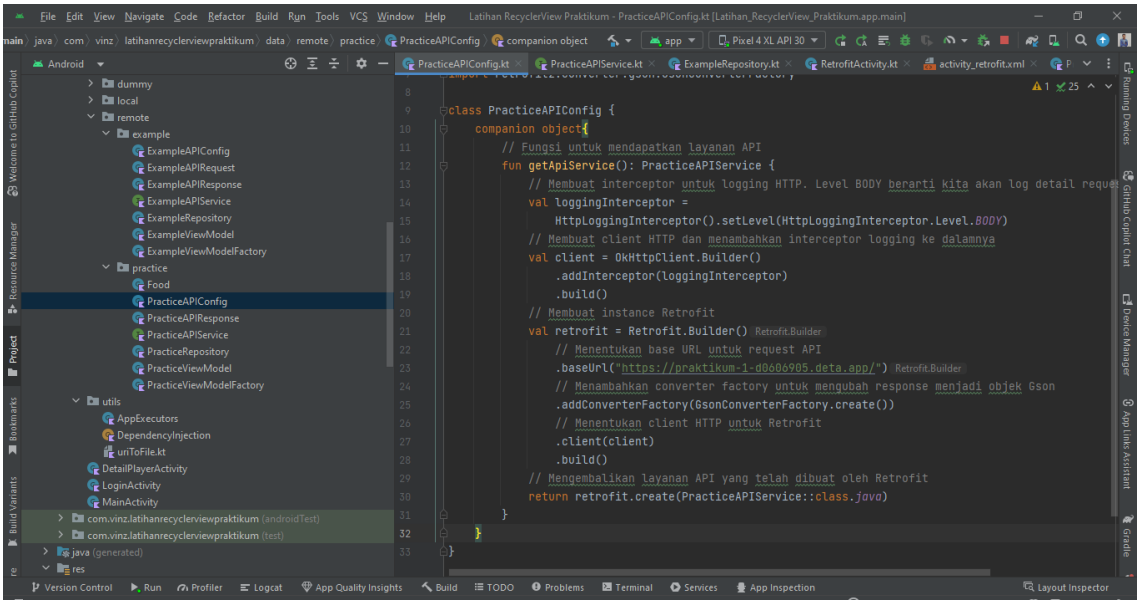
```
// retrofit
implementation("com.squareup.retrofit2:retrofit:2.9.0")
implementation("com.squareup.retrofit2:converter-gson:2.9.0")
implementation("com.squareup.okhttp3:logging-interceptor:4.9.0")
```



2. Buka package data – remote – practice – PracticeAPIConfig, lalu masukkan kode berikut ini

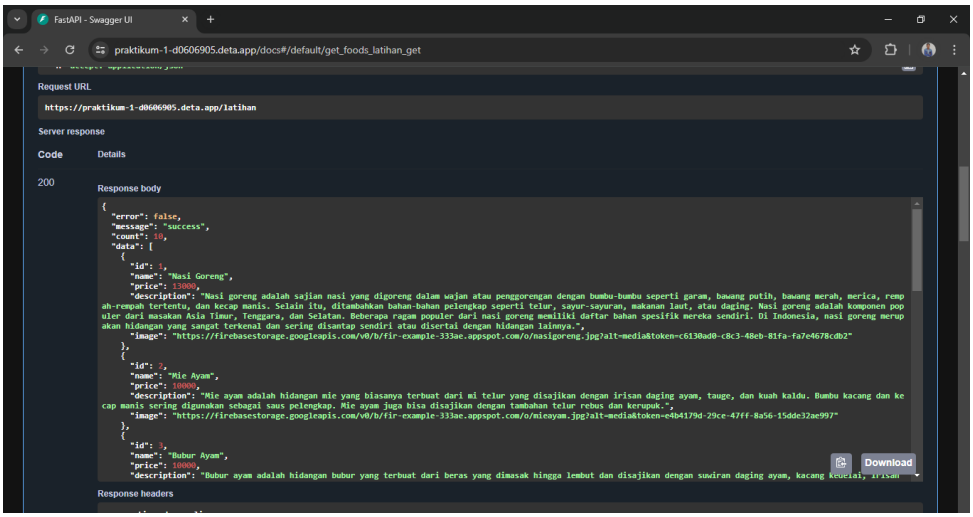
```
class PracticeAPIConfig {
    companion object{
        // Fungsi untuk mendapatkan layanan API
        fun getApiService(): PracticeAPIService {
            // Membuat interceptor untuk logging HTTP. Level BODY berarti kita akan log detail request dan response.
            val loggingInterceptor =
                HttpLoggingInterceptor().setLevel(HttpLoggingInterceptor.Level.BODY)
            // Membuat client HTTP dan menambahkan interceptor logging ke dalamnya
            val client = OkHttpClient.Builder()
                .addInterceptor(loggingInterceptor)
                .build()
            // Membuat instance Retrofit
            val retrofit = Retrofit.Builder()
```

```
            // Menentukan base URL untuk request API
            .baseUrl("https://praktikum-1-d0606905.deta.app/")
            // Menambahkan converter factory untuk mengubah response menjadi objek
Gson
            .addConverterFactory(GsonConverterFactory.create())
            // Menentukan client HTTP untuk Retrofit
            .client(client)
            .build()
        // Mengembalikan layanan API yang telah dibuat oleh Retrofit
        return retrofit.create(PracticeAPIService::class.java)
    }
  }
}
```

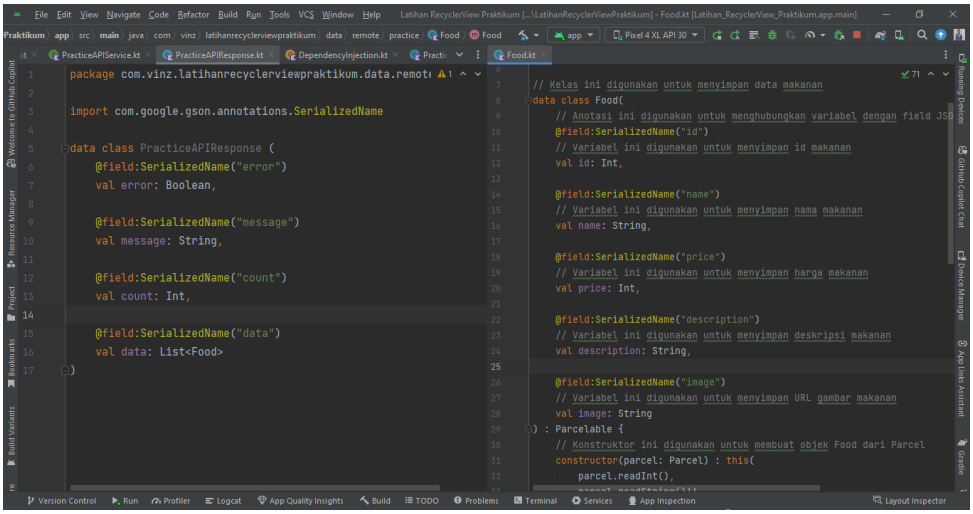


3. Lalu buka link https://praktikum-1-d0606905.deta.app/docs dan kita akan melakukan pemetaan terhadap response API menjadi Data Class yang terdapat di Kotlin

4. Lakukan pemetaan di endpoint /latihan, lihat responsenya, lalu petakan responsenya di dalam PracticeAPIResponse



Gambar 4. API Response



Gambar 5. Setelah dilakukan pemetaan ke Data Class

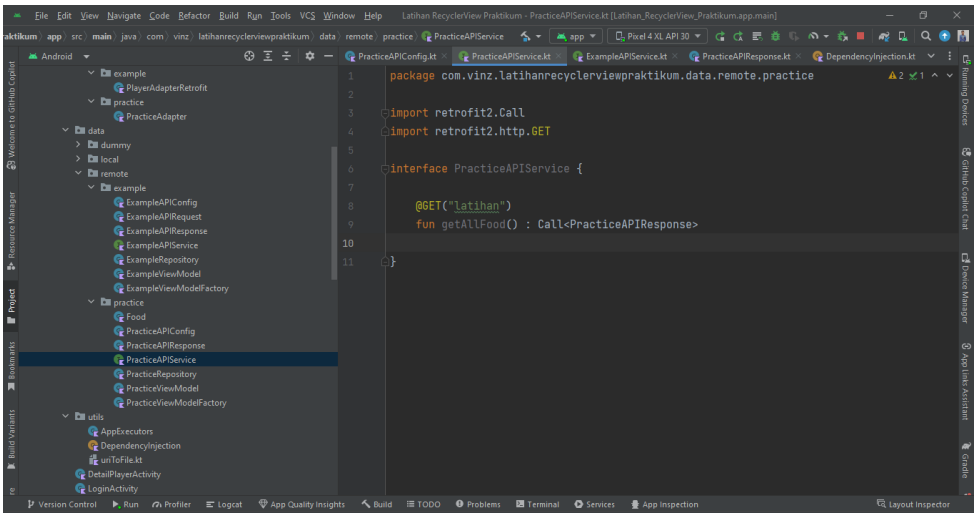
```
data class PracticeAPIResponse (
    @field:SerializedName("error")
    val error: Boolean,

    @field:SerializedName("message")
    val message: String,

    @field:SerializedName("count")
    val count: Int,

    @field:SerializedName("data")
```

```
    val data: List<Food>
)
```

```
// Kelas ini digunakan untuk menyimpan data makanan
data class Food(
    // Anotasi ini digunakan untuk menghubungkan variabel dengan field JSON
    @field:SerializedName("id")
    // Variabel ini digunakan untuk menyimpan id makanan
    val id: Int,

    @field:SerializedName("name")
    // Variabel ini digunakan untuk menyimpan nama makanan
    val name: String,

    @field:SerializedName("price")
    // Variabel ini digunakan untuk menyimpan harga makanan
    val price: Int,

    @field:SerializedName("description")
    // Variabel ini digunakan untuk menyimpan deskripsi makanan
    val description: String,

    @field:SerializedName("image")
    // Variabel ini digunakan untuk menyimpan URL gambar makanan
    val image: String
) : Parcelable {
    // Konstruktor ini digunakan untuk membuat objek Food dari Parcel
    constructor(parcel: Parcel) : this(
        parcel.readInt(),
        parcel.readString()!!,
        parcel.readInt(),
        parcel.readString()!!,
        parcel.readString()!!
```

```
    )

    // Fungsi ini digunakan untuk menulis data Food ke Parcel
    override fun writeToParcel(dest: Parcel, flags: Int) {
        dest.writeInt(id)
        dest.writeString(name)
        dest.writeInt(price)
        dest.writeString(description)
        dest.writeString(image)
    }

    // Fungsi ini digunakan untuk mendeskripsikan jenis konten khusus yang ditangani
oleh Parcelable
    override fun describeContents(): Int {
        return 0
    }

    // Objek companion yang digunakan untuk membuat objek Food dari Parcel dan
array dari Food
    companion object CREATOR : Parcelable.Creator<Food> {
        // Fungsi ini digunakan untuk membuat objek Food dari Parcel
        override fun createFromParcel(source: Parcel): Food {
            return Food(source)
        }

        // Fungsi ini digunakan untuk membuat array dari Food
        override fun newArray(size: Int): Array<Food?> {
            return arrayOfNulls(size)
        }
    }
}
```

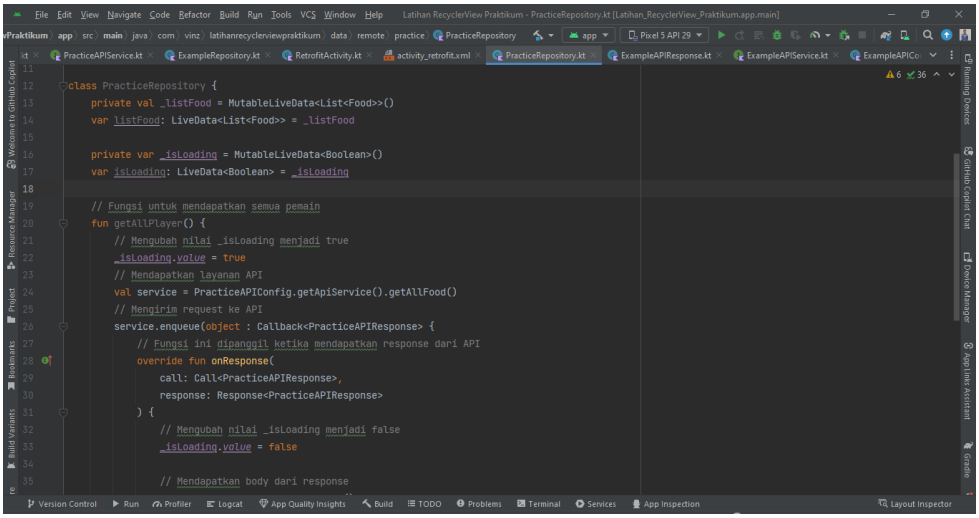
5. Setelah kita lakukan pemetaan di dalam Data Class, langkah selanjutnya adalah kita masuk ke data – remote – practice – PracticeAPIService, lalu masukkan endpoint yang ingin kita gunakan di dalam PracticeAPIService



6. Setelah itu buka data – remote – practice – PracticeRepository, dan masukkan kode berikut ini (ganti kode PracticeRepository menjadi kode ini)

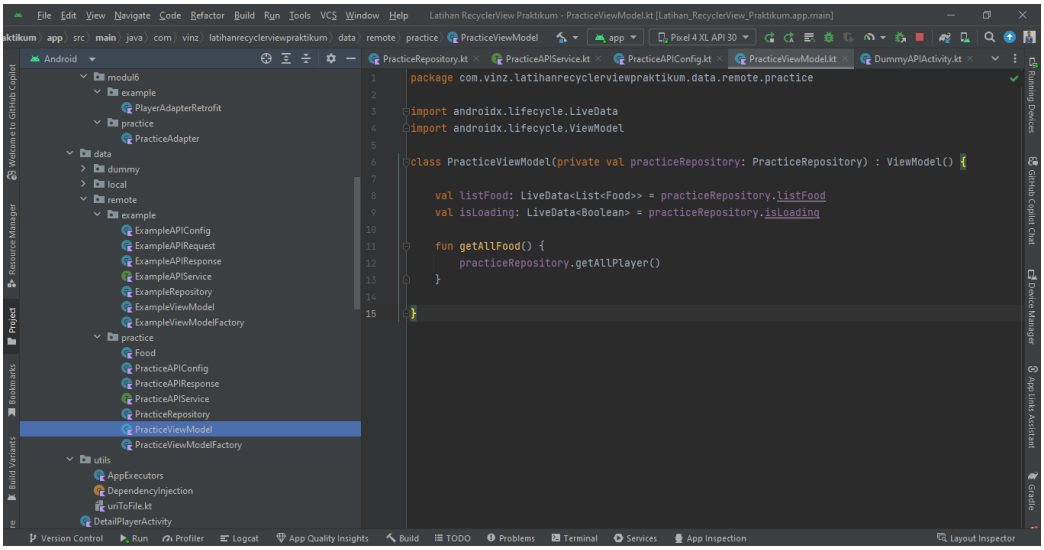
```
class PracticeRepository {
    private val _listFood = MutableLiveData<List<Food>>()
    var listFood: LiveData<List<Food>> = _listFood

    private var _isLoading = MutableLiveData<Boolean>()
    var isLoading: LiveData<Boolean> = _isLoading

    // Fungsi untuk mendapatkan semua pemain
    fun getAllPlayer() {
        // Mengubah nilai _isLoading menjadi true
        _isLoading.value = true
        // Mendapatkan layanan API
        val service = PracticeAPIConfig.getApiService().getAllFood()
        // Mengirim request ke API
        service.enqueue(object : Callback<PracticeAPIResponse> {
            // Fungsi ini dipanggil ketika mendapatkan response dari API
            override fun onResponse(
                call: Call<PracticeAPIResponse>,
```

```
            response: Response<PracticeAPIResponse>
        ) {
            // Mengubah nilai _isLoading menjadi false
            _isLoading.value = false

            // Mendapatkan body dari response
            val responseBody = response.body()
            // Jika response sukses dan responseBody tidak null, ubah nilai _listPlayer
dengan responseBody
            if (response.isSuccessful && responseBody != null) {
                _listFood.value = response.body()!!.data
            } else {
                // Jika response gagal, log pesan error
                Log.e("Error on Response", "onFailure: ${response.message()}")
            }
        }

        // Fungsi ini dipanggil ketika request ke API gagal
        override fun onFailure(call: Call<PracticeAPIResponse>, t: Throwable) {
            // Mengubah nilai _isLoading menjadi false
            _isLoading.value = false
            // Log pesan error
            Log.e("Error on Failure", "onFailure: ${t.message}")
        }
    })
  }
}
```

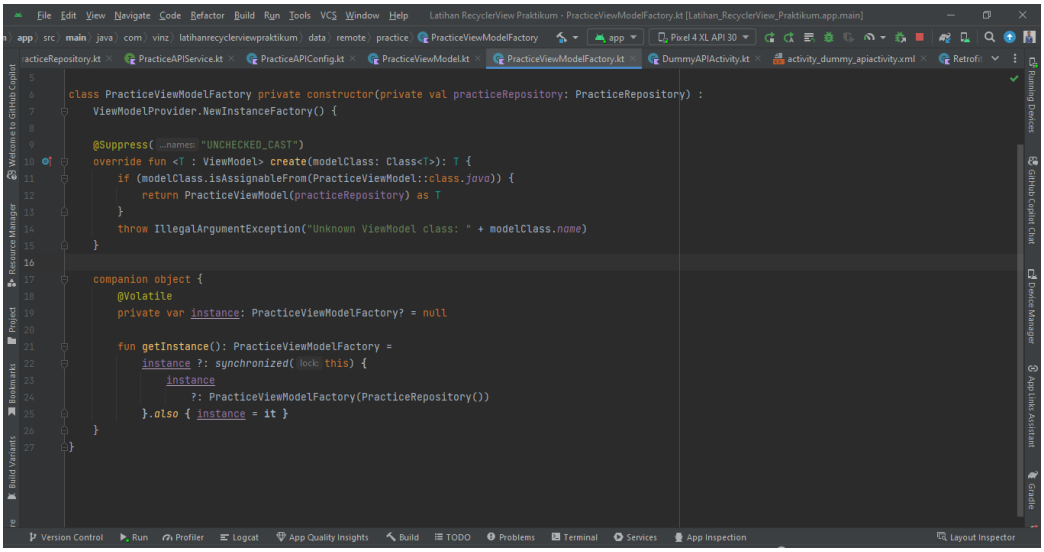


7. Setelah itu, buka data – remote – practice – PracticeViewModel, dan masukkan kode berikut ini (ganti kode PracticeViewModel menjadi kode ini)

```
class PracticeViewModel(private val practiceRepository: PracticeRepository) : ViewModel() {
    val listFood: LiveData<List<Food>> = practiceRepository.listFood
    val isLoading: LiveData<Boolean> = practiceRepository.isLoading

    fun getAllFood() {
        practiceRepository.getAllPlayer()
    }
}
```



8. Lalu buka data – remote – practice – PracticeViewModelFactory, dan masukkan kode berikut ini (ganti kode PracticeViewModelFactory menjadi kode ini)



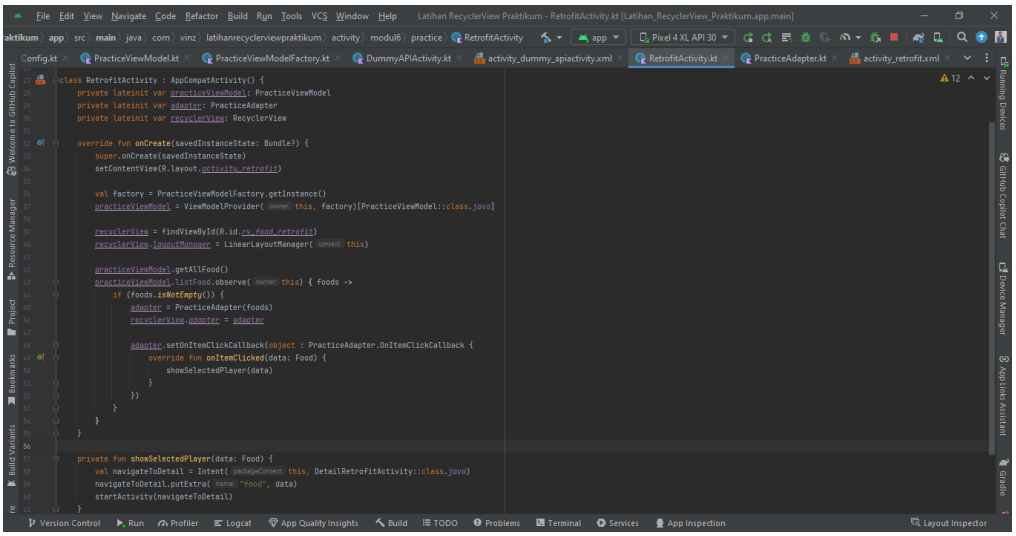
```
class PracticeViewModelFactory private constructor(private val practiceRepository: PracticeRepository) :
    ViewModelProvider.NewInstanceFactory() {

    @Suppress("UNCHECKED_CAST")
    override fun <T : ViewModel> create(modelClass: Class<T>): T {
        if (modelClass.isAssignableFrom(PracticeViewModel::class.java)) {
            return PracticeViewModel(practiceRepository) as T
        }
        throw IllegalArgumentException("Unknown ViewModel class: " + modelClass.name)
    }

    companion object {
        @Volatile
        private var instance: PracticeViewModelFactory? = null

        fun getInstance(): PracticeViewModelFactory =
            instance ?: synchronized(this) {
                instance
                    ?: PracticeViewModelFactory(PracticeRepository())
            }.also { instance = it }
    }
```

```
}
```

9. Terakhir, buka activity – modul6 – practice – RetrofitActivity, dan masukkan kode berikut ini (ganti kode RetrofitActivity menjadi kode ini)

```
class RetrofitActivity : AppCompatActivity() {
    private lateinit var practiceViewModel: PracticeViewModel
    private lateinit var adapter: PracticeAdapter
    private lateinit var recyclerView: RecyclerView

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_retrofit)

        val factory = PracticeViewModelFactory.getInstance()
        practiceViewModel = ViewModelProvider(this,
factory)[PracticeViewModel::class.java]

        recyclerView = findViewById(R.id.rv_food_retrofit)
        recyclerView.layoutManager = LinearLayoutManager(this)

        practiceViewModel.getAllFood()
        practiceViewModel.listFood.observe(this) { foods ->
            if (foods.isNotEmpty()) {
                adapter = PracticeAdapter(foods)
                recyclerView.adapter = adapter

                adapter.setOnItemClickCallback(object :
PracticeAdapter.OnItemClickCallback {
                    override fun onItemClicked(data: Food) {
                        showSelectedPlayer(data)
                    }
                })
            }
        }
```

```
    }

    private fun showSelectedPlayer(data: Food) {
        val navigateToDetail = Intent(this, DetailRetrofitActivity::class.java)
        navigateToDetail.putExtra("food", data)
        startActivity(navigateToDetail)
    }
}
```

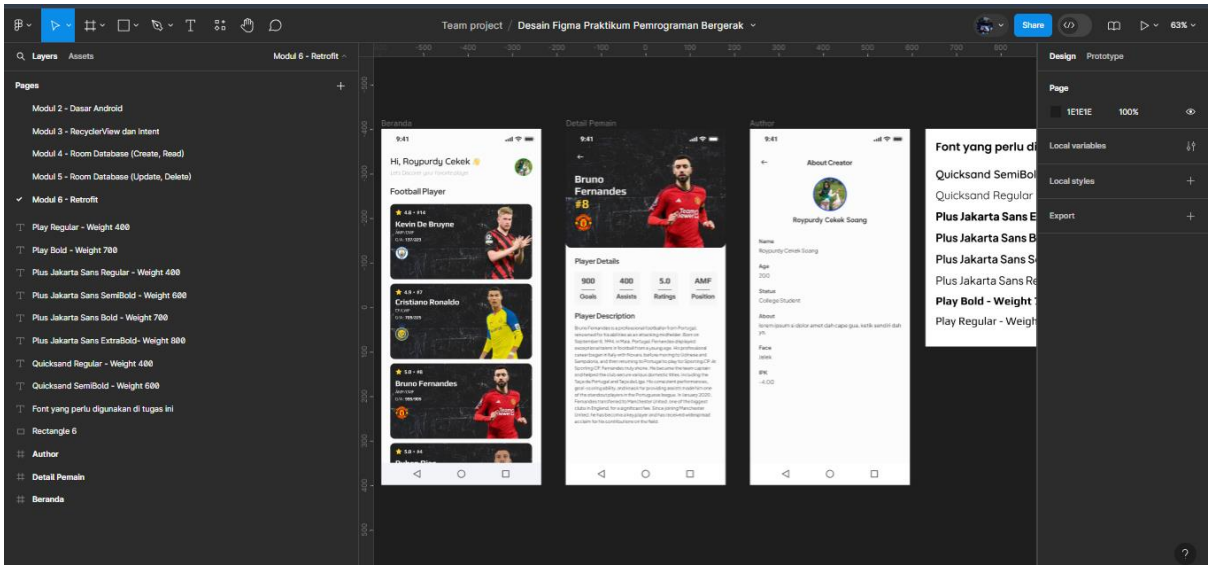


Tugas Pendahuluan

1. Mengapa kita membutuhkan REST API di dalam aplikasi yang kita buat?
2. Berikan 3 studi kasus tentang 2 aplikasi besar yang saling berinteraksi menggunakan REST API (bisa cek di Gambar 2 sebagai contoh)!
3. Apakah bisa kita berinteraksi dengan API tanpa menggunakan library tambahan di Android? Kalau bisa, kenapa hal tersebut bisa terjadi?

Tugas Praktikum

Buatlah aplikasi seperti berikut ini (cek Pages 6 yaitu Modul 6 – Retrofit)!



API Link: https://praktikum-1-d0606905.deta.app

API Docs: https://praktikum-1-d0606905.deta.app/docs

Endpoint untuk tugas: https://praktikum-1-d0606905.deta.app/tugas

Ketentuan dalam pembuatan aplikasi:

1. Menerapkan desain yang tersedia menjadi sebuah aplikasi
2. Menerapkan Retrofit sebagai library untuk pemanggilan API dalam pembuatan aplikasi
3. Menerapkan API sebagai sumber data di dalam aplikasi

4. Menerapkan RecyclerView untuk menampilkan data yang berasal dari API

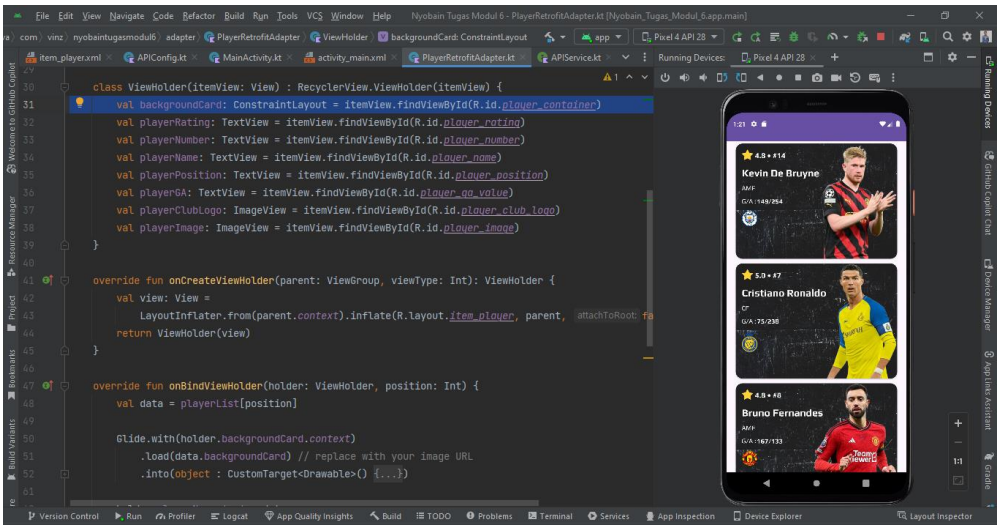
Alur Aplikasi:

1. Ketika user masuk pertama kali, maka akan diarahkan ke Halaman Beranda
2. Halaman berisi konten yang berasal dari API
3. Ketika user melakukan klik terhadap item yang tampil di Halaman Beranda, maka akan diarahkan ke Halaman Detail Pemain
4. Halaman Detail Pemain berisi detail dari pemain tersebut
5. Ketika user klik kembali di Halaman Detail Pemain, maka akan mengembalikan user ke Halaman Beranda
6. Ketika user melakukan klik terhadap profile di pojok kanan atas, maka akan mengarahkan user ke Halaman Author
7. Halaman Author berisi informasi detail dari si pembuat aplikasi

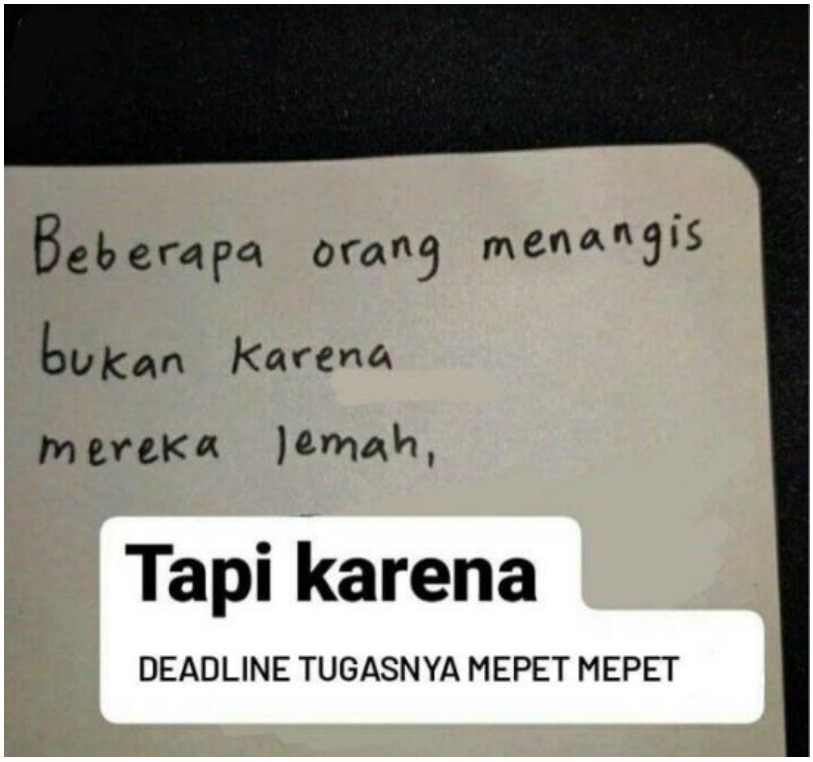
Tips:

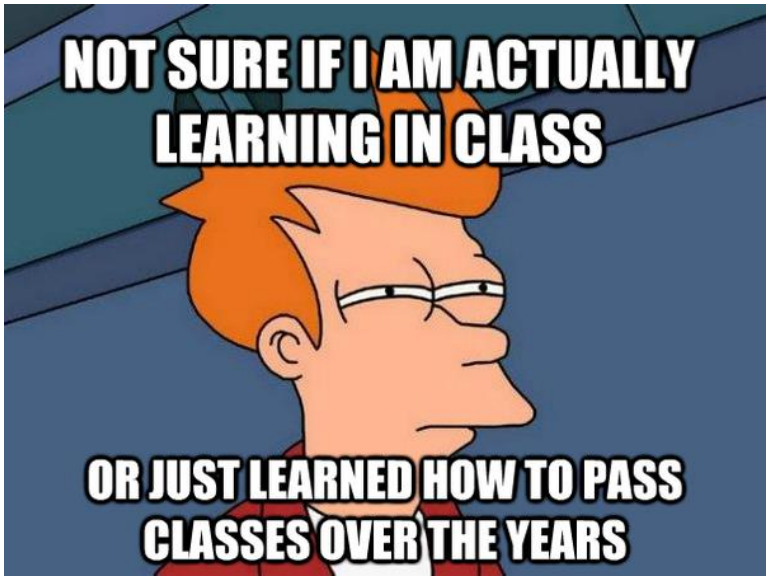
1. Buat background dari Card yang ada di Halaman Beranda, bisa pake ConstraintLayout yang nanti dimasukkan ke dalam Glide (definisikan dulu ConstraintLayout di dalam ViewHolder adapter, lalu panggil di dalam BindViewHolder dan masukkan kode ini)

```
Glide.with(holder.backgroundCard.context)
    .load(data.backgroundCard) // replace with your image URL
    .into(object : CustomTarget<Drawable>() {
        override fun onResourceReady(resource: Drawable, transition: Transition<in
Drawable>?) {
            holder.backgroundCard.background = resource
        }

        override fun onLoadCleared(placeholder: Drawable?) {
            // handle cleanup here
        }
    })
```

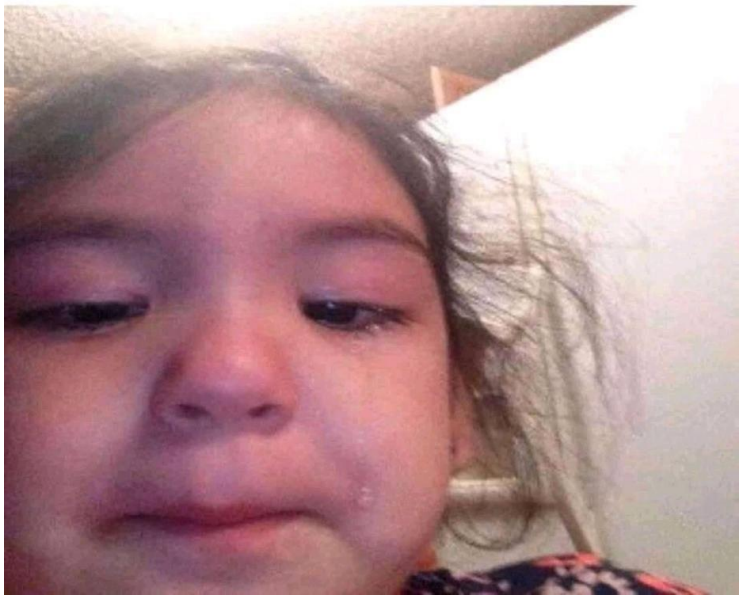


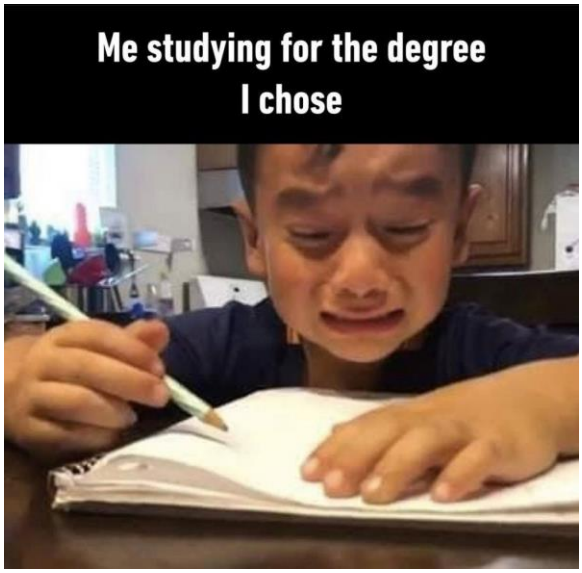
2. Buat modul ini bener-bener diuji tentang pemahaman layouting, semakin sesuai aplikasi dengan desain, tandanya pemahaman layoutnya sudah sangat baik


Beberapa orang menangis
bukan karena
mereka lemah,
Tapi karena
DEADLINE TUGASNYA MEPET MEPET


NOT SURE IF I AM ACTUALLY LEARNING IN CLASS
OR JUST LEARNED HOW TO PASS CLASSES OVER THE YEARS

KU KIRA JURUSAN KU MUDAH
TERNYATA MUDAH-MUDAHAN BISA BERTAHAN




Me studying for the degree
I chose