# Requisições padrão Rest

## Podemos fazer de duas maneiras

Utilizando a biblioteca Retrofit

Utilizando a classe HttpURLConnection (Bem mais complexa)

## Como configurar o Retrofit

<https://square.github.io/retrofit/>

Configurar ela como dependência no Gradle.

## Como funciona para pegar oque está dentro do endereço

Temos uma variável

Val retrofit = Retrofit.Builder() (Isso é uma classe)

.baseUrl(“aqui vai a url da API”)

.addConverterFactory(CONVERSOR) = aqui é onde vamos converter os dados para objeto

.build() // por ultimo para criar o objeto que vai ficar dentro de retrofit

# Agora vamos fazer a configuração na pratica.

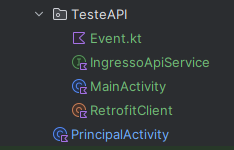
Primeiro estamos adicionando o Gradle

// É recomendado utilizar sempre a mesma versao do RetroFit  
*implementation*("com.squareup.retrofit2:retrofit:2.9.0")  
*implementation*("com.squareup.retrofit2:converter-gson:2.9.0")  
*implementation*("com.google.code.gson:gson:2.8.7")

## Agora precisamos fazer a requisição do manifest porque vamos acessar a internet.

<activity  
 android:name=".splash.SplashAcitivity"  
 android:exported="true"  
 android:screenOrientation="portrait" >  
 <intent-filter>  
 <action android:name="android.intent.action.MAIN" />  
  
 <category android:name="android.intent.category.LAUNCHER" />  
 </intent-filter>  
 </activity>  
  
 <meta-data  
 android:name="preloaded\_fonts"  
 android:resource="@array/preloaded\_fonts" />  
</application>  
<uses-permission android:name="android.permission.INTERNET"/>  
<uses-permission android:name="android.permission.ACCESS\_NETWORK\_STATE"/>

Detalhes dos items que vamos ter que criar



Class/Object RetroFitClient

package com.allephnogueira.projetoingresso.TesteAPI  
  
  
import retrofit2.Retrofit  
import retrofit2.converter.gson.GsonConverterFactory  
  
object RetrofitClient {  
 private const val BASE\_URL = "https://api-content.ingresso.com/"  
  
 val instance: IngressoApiService by *lazy* **{** val retrofit = Retrofit.Builder()  
 .baseUrl(BASE\_URL)  
 .addConverterFactory(GsonConverterFactory.create())  
 .build()  
  
 retrofit.create(IngressoApiService::class.*java*)  
 **}**}

Class Event

São os itens que tem no nosso servidor, os itens que vao voltar

package com.allephnogueira.projetoingresso.TesteAPI  
  
  
data class EventResponse(  
 val items: List<Event>  
)  
  
data class Event(  
 val id: String,  
 val title: String,  
 val date: String,  
 val images: List<Image>  
)  
  
data class Image(  
 val url: String  
)

Nossa activity com os itens principais

class PrincipalActivity : AppCompatActivity() {  
  
  
  
  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 *enableEdgeToEdge*()  
 setContentView(R.layout.*activity\_principal*)  
 Log.d("Debug", "Activity aberta.")  
 ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.*main*)) **{** v, insets **->** val systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars())  
 v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom)  
 insets  
 **}** val imagemTopo: ImageView = findViewById(R.id.*imageAnuncio*) // Imagem do topo  
 val nomeFilmeTopo: TextView = findViewById(R.id.*textNomeTopo*) // Titulo do filme  
 val dataFilmeTopo: TextView = findViewById(R.id.*textData*) // Data do filme  
  
 val api = RetrofitClient.instance  
 // Aqui estamos pegando a interface  
 api.getUpcomingEvents().enqueue(object : retrofit2.Callback<EventResponse> {  
 override fun onResponse(call: Call<EventResponse>, response: Response<EventResponse>) {  
 if (response.*isSuccessful*) { // Verificando se retornou algo  
 val eventResponse = response.body() // Passando para o eventResponse o Body que veio do servidor  
 val events = eventResponse?.items // Passando para o events os items que veio do servidor  
 if (events != null && events.*isNotEmpty*()) { // Testando para ver se tem algo dentro de events  
  
 // Aqui vamos criar um for para exibir um filme de 3 em 3 segundos.  
  
 val firstEvent = events.*random*() // Aqui esta a lista de filmes, vamos exibir um filme aleatorio no topo  
  
  
 nomeFilmeTopo.*text* = firstEvent.title // Passando o titulo  
 dataFilmeTopo.*text* = firstEvent.date // Data do item  
  
 val imageUrl = firstEvent.images.*firstOrNull*()?.url // Pegando a imagem do item  
 Log.d("MainActivity", "Image URL: $imageUrl")  
  
 if (!imageUrl.*isNullOrEmpty*()) { // Tem algo dentro de imagem? Se tiver vamos carregar  
 Glide.with(this@PrincipalActivity)  
 .load(imageUrl) // Carregando a imagem  
 .into(imagemTopo) // Carregando na imagem que criamos.  
 } else {  
 Log.e("MainActivity", "Image URL is null or empty")  
 }  
 } else {  
 Log.e("MainActivity", "No events found")  
 }  
 } else {  
 Log.e("MainActivity", "Falha ao buscar eventos: ${response.errorBody()?.string()}")  
 }  
 }  
  
 override fun onFailure(call: Call<EventResponse>, t: Throwable) {  
 Log.e("MainActivity", "Erro: ${t.message}")  
 }  
 })  
 }  
  
  
}