Requisições http – Android kotlin

Prof. Me. Hélio Esperidião

O que é HTTP?

- O HTTP (Hypertext Transfer Protocol) é um protocolo de comunicação usado para transferir dados pela internet.
- Ele define um conjunto de regras que devem ser seguidas tanto pelo cliente quanto pelo servidor para que possam se comunicar de forma eficiente.

Modo de funcionamento

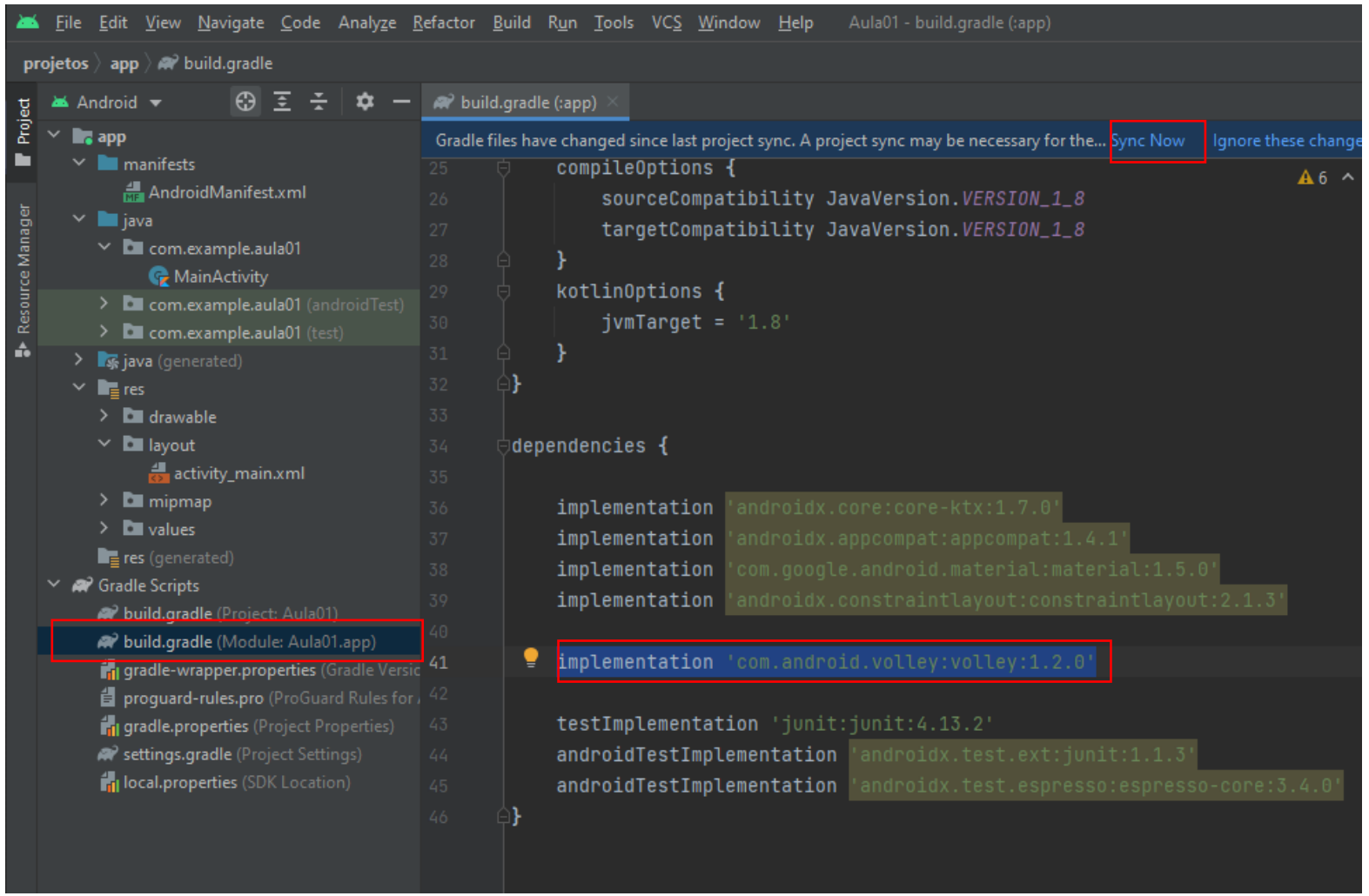
- Quando um cliente faz uma solicitação para um servidor web usando o protocolo HTTP, ele envia uma mensagem de pedido (requisição) contendo informações como o tipo de recurso solicitado.
- O servidor, por sua vez, recebe a solicitação, verifica se a mensagem está formatada corretamente, processa o pedido e envia uma resposta de volta para o cliente.

Configuração: AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
   package="com.example.aula01" >
   <uses-permission android:name="android.permission.INTERNET" />
   <application
      android:allowBackup="true"
      android:icon="@mipmap/ic_launcher"
      android:label="@string/app_name"
      android:roundIcon="@mipmap/ic_launcher_round"
      android:supportsRtl="true"
      android:theme="@style/Theme.Aula01" >
      <activity
         android:name=".MainActivity"
         android:exported="true" >
         <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
         </intent-filter>
      </activity>
   </application>
</manifest>
```

No arquivo AndroidManifest.xml acrescente permissão para acesso a internet

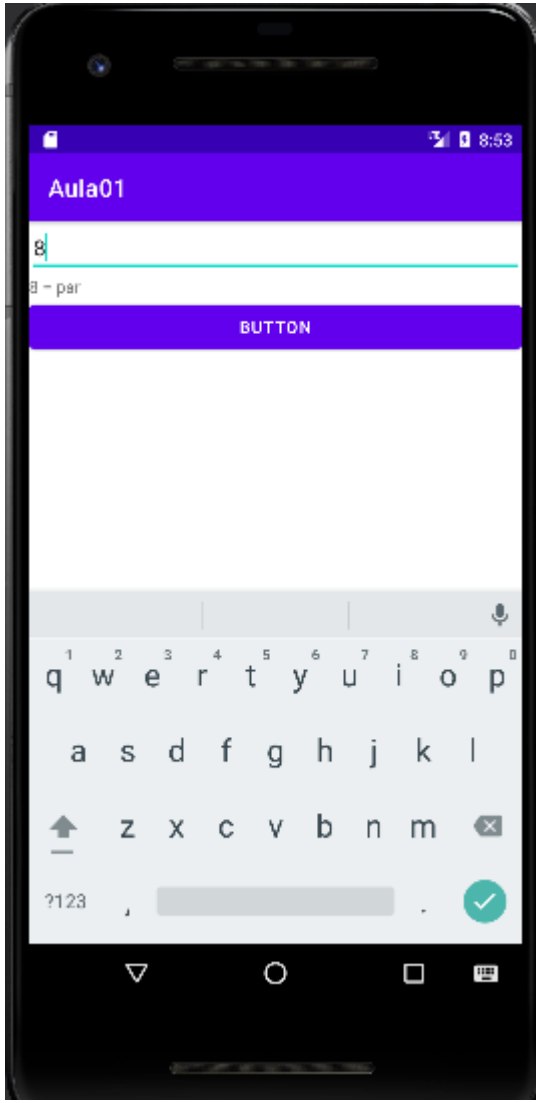
Configuração biblioteca volley:



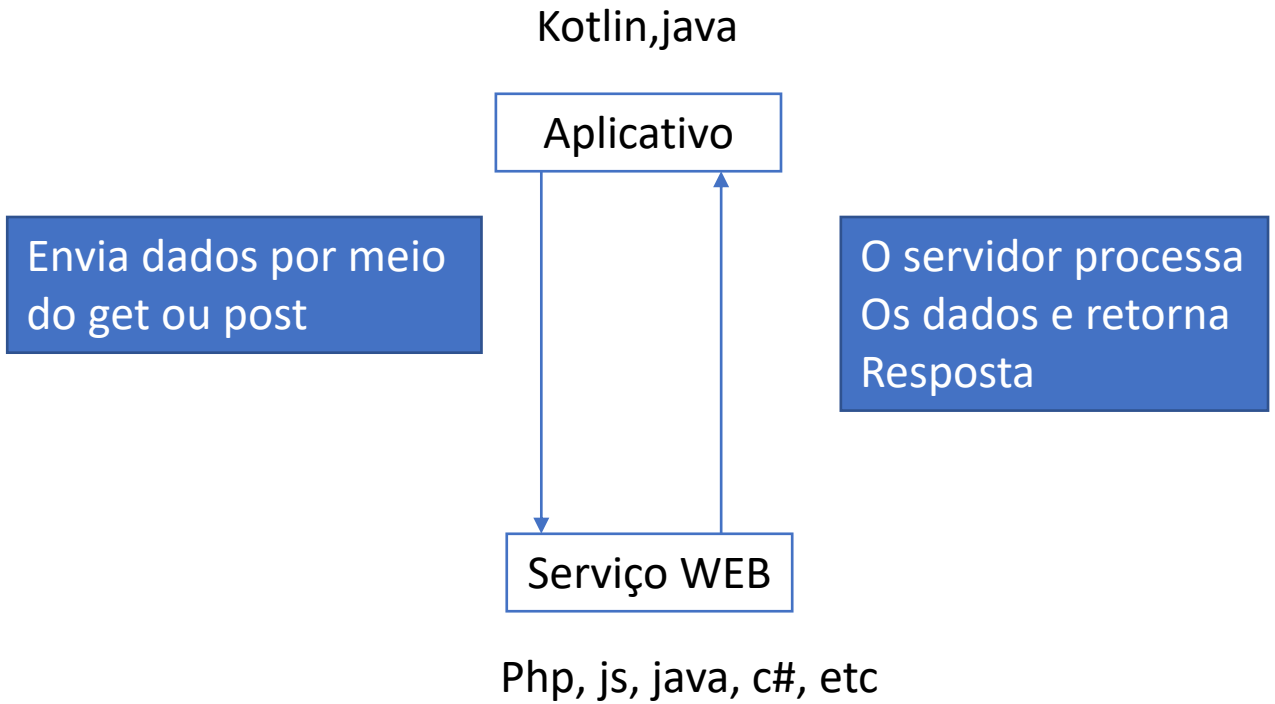
- Acesse o arquivo build.gragle
- Acrecente a biblioteca:
  - implementation 'com.android.volley:volley:1.2.0’
- Click em *sync* para a ferramenta fazer o download e configuração da biblioteca.

xml

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">
    <EditText
       android:id="@+id/txtNumero"
       android:layout_width="match_parent"
       android:layout_height="wrap_content"
        android:inputType="textPersonName"
     />
    <TextView
       android:id="@+id/lblResultado"
       android:layout_width="match_parent"
       android:layout_height="wrap_content"
       android:text="TextView" />
    <Button
       android:id="@+id/btn"
       android:layout_width="match_parent"
       android:layout_height="wrap_content"
       android:text="Button" />
 </LinearLayout>
```



Arquitetura

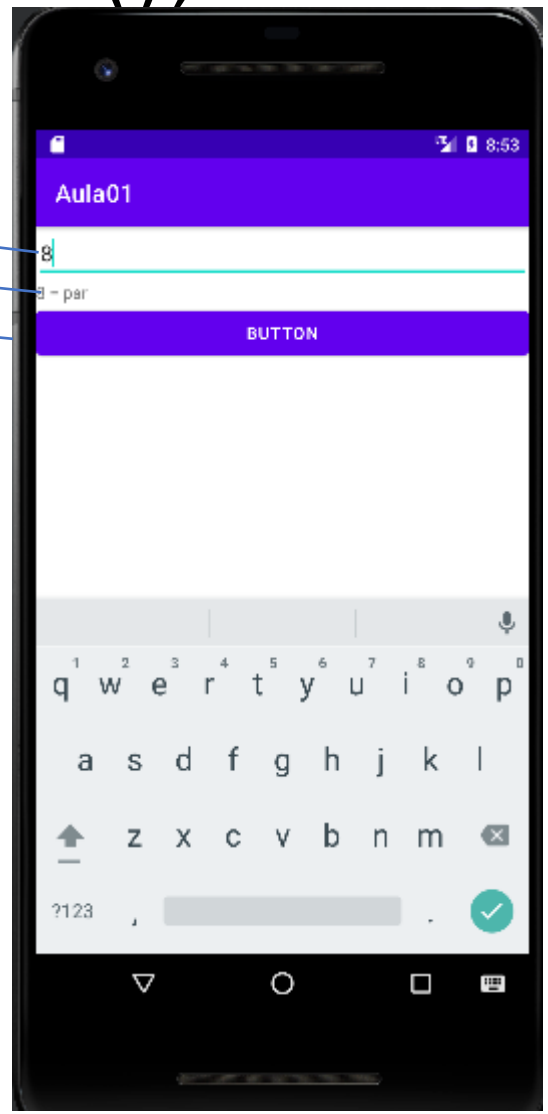


Atributos e método onCreate();

```
lateinit var txtNumero: EditText
lateinit var lblResposta: TextView
lateinit var btn1: Button
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)
    init();
}
fun init(){
    txtNumero = findViewById(R.id.txtNumero)
    lblResposta= findViewById(R.id.lblResultado)
    btn1 = findViewById(R.id.btn)
    btn1.setOnClickListener {
        acessarAPI_PHP_par_impar();
    }
}
```

Inicializando objetos

Executado no click do botão



```
fun acessarAPI_PHP_par_impar() {
    var numero:Int
    numero = txtNumero.text.toString().toInt()
    val queue = Volley.newRequestQueue(this)
    val url = "http://helioesperidiao.com/par.php"
    val requestBody = "txtNumero="+numero + "&msg=test_msg"
    val stringReq : StringRequest = object : StringRequest(Method.POST, url,
        Response.Listener { response ->
            var resposta = response.toString()
            lblResposta.text=resposta
        },
        Response.ErrorListener { error ->
            Log.d("API", "error => $error")
        }
    ){
        override fun getBody(): ByteArray {
            return requestBody.toByteArray(Charset.defaultCharset())
        }
    }
    queue.add(stringReq)
}
```

É executado para receber a resposta do servidor
Tudo que é escrito pelo servidor será armazenado em response.

Só executa em caso de erro

Monta a resposta do servidor

Adiciona a fila de execuções e está pronta para fazer a chamada para o servidor

Lado servidor: http://helioesperidiao.com/par.php

```php
<?php
//verifica se a variável txtNumero existe
if(isset($_POST['txtNumero'])){
   $n = $_POST['txtNumero'];
   if($n%2==0){
    echo "$n = par";
   }else{
    echo "$n = impar";
   }
}else{
   echo "dados não recebidos";
}


?>
```

Verifica o recebimento do post

Armazena o dado recebido em uma variável

Verifica se é verdadeiro ou false e escreve uma resposta que será enviada para o kotlin

Código completo

```
class MainActivity : AppCompatActivity() {
    lateinit var txtNumero: EditText
    lateinit var lblResposta: TextView
    lateinit var btn1: Button
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        init();
    }
    fun init(){
        txtNumero = findViewById(R.id.txtNumero)
        lblResposta= findViewById(R.id.lblResultado)
        btn1 = findViewById(R.id.btn)
        btn1.setOnClickListener {
            acessarAPI_PHP_par_impar();
        }
    }
    fun acessarAPI_PHP_par_impar() {
        var numero:Int
        numero = txtNumero.text.toString().toInt()
        val queue = Volley.newRequestQueue(this)
        val url = "http://helioesperidiao.com/par.php"
        val requestBody = "txtNumero="+numero + "&msg=test_msg"
        val stringReq : StringRequest =
            object : StringRequest(Method.POST, url,
                Response.Listener { response ->
                    // response
                    var resposta = response.toString()
                    lblResposta.text=resposta
                    println(resposta)
                },
                Response.ErrorListener { error ->
                    Log.d("API", "error => $error")
                }
            ){
                override fun getBody(): ByteArray {
                    return requestBody.toByteArray(Charset.defaultCharset())
                }
            }
        queue.add(stringReq)
    }
}
```

Carregando imagens de urls

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">
    <Button
      android:id="@+id/btn"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:text="Button" />
    <ImageView
      android:id="@+id/imageView"
      android:layout_width="match_parent"
      android:layout_height="500dp"
    />
</LinearLayout>
```

A imagem será carregada no evento click

```kotlin
class MainActivity : AppCompatActivity() {
    lateinit var btn1: Button
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        init();
    }
        fun init(){
            btn1 = findViewById(R.id.btn)
            btn1.setOnClickListener {
                DownloadImageFromInternet(findViewById(R.id.imageView)).execute("https://all.accor.com/magazine/imagerie/ferias-na-praia-o-que-fazer-das-f163.jpg")
            }
        }
        @SuppressLint("StaticFieldLeak")
        @Suppress("DEPRECATION")
        private inner class DownloadImageFromInternet(var imageView: ImageView) : AsyncTask<String, Void, Bitmap?>() {
            init {
                Toast.makeText(applicationContext, "Carregando imagem",     Toast.LENGTH_SHORT).show()
            }
            override fun doInBackground(vararg urls: String): Bitmap? {
                val imageURL = urls[0]
                var image: Bitmap? = null
                try {
                    val `in` = java.net.URL(imageURL).openStream()
                    image = BitmapFactory.decodeStream(`in`)
                }
                catch (e: Exception) {
                    Log.e("Error Message", e.message.toString())
                    e.printStackTrace()
                }
                return image
            }
            override fun onPostExecute(result: Bitmap?) {
                imageView.setImageBitmap(result)
            }
        }

}
```

Para usar a classe basta passar a instancia do imageView e o caminho da imagem que será carregada

Classe interna para fazer requisições em background

.kt

Json

- Um acrônimo de JavaScript Object Notation, é um formato compacto, de padrão aberto independente, de troca de dados simples e rápida entre sistemas.

```
{
  “firstName”: “Jonathan”,
  “lastName”: “Freeman”,
  “loginCount”: 4,
  “isWriter”: true,
  “worksWith”: [“Spantree Technology Group”, “InfoWorld”],
  “pets”: [
    {
      “name”: “Lilly”,
      “type”: “Raccoon”
    }
  ]
}
```

Json

```
[
  {
    "id":"01",
    "type":"refrigerante",
    "desc":“Guaraná",
    "qtd":"600ml",
    "img":"http://helioesperidiao.com/img/eu.jpg"
  },
  {
    "id":"02",
    "type":"refrigerante",
    "desc":“laranja",
    "qtd":"600ml",
    "img":"http://helioesperidiao.com/img/eu.jpg"
  },
  {
    "id":"03",
    "type":"refrigerante",
    "desc":"coca cala",
    "qtd":“1L",
    "img":"http://helioesperidiao.com/img/eu.jpg"
  }
]
```

```
fun getJsonProdutosFromWEB() {
    val queue = Volley.newRequestQueue(this)
    val url = "http://helioesperidiao.com/api.php"
    val requestBody = "id=1" + "&msg=test_msg"
    val stringReq : StringRequest =
        object : StringRequest(Method.POST, url,
            Response.Listener { response ->
                val linearLayou: LinearLayout = findViewById(R.id.linerL1)
                var resposta = response.toString()
                val array = JSONArray(resposta)
                val tamanho =array.length()
                for (i in 0 until tamanho ) {
                    val item: JSONObject = array.getJSONObject(i) // recupera o objeto na posição dentro do array
                    var idProduto = item.get("id").toString()
                    var type = item.get("type").toString();
                    var desc = item.get("desc").toString();
                    var qtd = item.get("qtd").toString();
                    var img = item.get("img").toString();
                    var novoTextView = TextView(this)
                    println(idProduto + " - " +type)
                }
            },
            Response.ErrorListener { error ->
                Log.d("API", "error => $error")
            }
        ){
            override fun getBody(): ByteArray {
                return requestBody.toByteArray(Charset.defaultCharset())
            }
        }
    queue.add(stringReq)
}
```

Carregar e processar json da web

```
{
    "id":"01",
    "type":"refrigerante",
    "desc":“Guaraná",
    "qtd":"600ml",
    "img":"http://helioesperidiao.com/img/eu.jpg"
 },
```

```
fun getJsonProdutosFromWEB() {
    val queue = Volley.newRequestQueue(this)
    val url = "http://helioesperidiao.com/api.php"
    val requestBody = "id=1" + "&msg=test_msg"
    val stringReq : StringRequest =
        object : StringRequest(Method.POST, url,
            Response.Listener { response ->
                val linearLayou: LinearLayout = findViewById(R.id.linerL1)
                var resposta = response.toString()
                val array = JSONArray(resposta)
                val tamanho =array.length()
                for (i in 0 until tamanho ) {
                    val item: JSONObject = array.getJSONObject(i) // recupera o objeto na posição dentro do array
                    var idProduto = item.get("id").toString()
                    var type = item.get("type").toString();
                    var desc = item.get("desc").toString();
                    var qtd = item.get("qtd").toString();
                    var img = item.get("img").toString();
                    var novoTextView = TextView(this)
                    novoTextView.text = idProduto + " - " +type
                    novoTextView.layoutParams = LinearLayout.LayoutParams(
                        LinearLayout.LayoutParams.MATCH_PARENT,
                        LinearLayout.LayoutParams.WRAP_CONTENT
                    )
                    linearLayou.addView(novoTextView)
                    println(idProduto + " - " +type)
                }
            },
            Response.ErrorListener { error ->
                Log.d("API", "error => $error")
            }
        ){
            override fun getBody(): ByteArray {
                return requestBody.toByteArray(Charset.defaultCharset())
            }
        }
    queue.add(stringReq)
}
```

Carregar e criar objetos dinamicamente.