Desenvolvimento para Dispositivos Móveis

Aula 08

Consumo de APIs REST e Integração com Backend

Prof. Msc. Francisco de Assis de Souza Rodrigues

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

🎯 Objetivos

- Entender como fazer requisições HTTP no Android.
- Consumir uma API RESTful (GET e POST).
- Integrar o app com um backend (ex: JSONPlaceholder, seu próprio backend, etc).
- Mostrar dados em tela com RecyclerView.

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

🔍 1. O que é uma API REST?

📌 **Definição básica:**

Uma **API REST (Representational State Transfer)** é uma forma **padronizada** de criar interfaces de comunicação entre sistemas através da internet, utilizando o **protocolo HTTP**, o mesmo usado para navegar em sites.

> Em outras palavras, uma API REST permite que um aplicativo "converse" com outro sistema — por exemplo, um app Android pedindo dados de um servidor na nuvem.

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

🔍 1. O que é uma API REST?

🔄 Como funciona?

Imagine que você está usando um app de previsão do tempo. Quando você abre o app, ele **faz uma requisição HTTP** para uma **API REST** que fornece os dados climáticos. Essa API retorna uma resposta (geralmente em **JSON**) com as informações do clima atual, que o app então exibe para você.

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

🔍 1. O que é uma API REST?

📬 Métodos HTTP usados nas APIs REST:

Esses métodos representam **ações** que você quer realizar:

Método	Ação	Exemplo na prática
`GET`	Buscar dados	Listar todos os posts de um blog
`POST`	Enviar dados (criar algo novo)	Criar um novo usuário ou comentário
`PUT`	Atualizar dados	Editar o perfil de um usuário
`DELETE`	Remover dados	Excluir um post ou conta

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

🔍 1. O que é uma API REST?

📄 O que é retornado?

A resposta de uma API REST geralmente vem em **formato JSON (JavaScript Object Notation)**, que é leve, fácil de ler e processar.

Exemplo de resposta JSON:

```json
{
  "id": 1,
  "title": "Hello world",
  "body": "This is my first post",
  "userId": 10
}
```

Esse JSON representa um **post de blog** com um título, conteúdo e ID do usuário autor.

UniFAP

🔍 1. O que é uma API REST?

🧠 **Vantagens de usar APIs REST:**

- **Padronização**: fácil de entender e usar com qualquer linguagem (Kotlin, JavaScript, Python…).
- **Independência**: o frontend (ex: app Android) e o backend podem evoluir separadamente.
- **Reutilizável**: uma API pode ser usada por vários tipos de clientes (app mobile, web, etc).

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

🧰 2. Bibliotecas para consumir APIs no Android

No Android, fazer requisições HTTP e trabalhar com JSON "na mão" seria muito trabalhoso e repetitivo. Para facilitar isso, usamos **bibliotecas especializadas** que tornam o processo muito mais simples, seguro e legível.

As duas mais utilizadas para esse fim são: **Retrofit** e **Gson**.

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

🧰 2. Bibliotecas para consumir APIs no Android

🔄 Retrofit — A biblioteca mais usada para consumo de APIs REST

📌 O que é?

Retrofit é uma **biblioteca da Square** (a mesma criadora do OkHttp) que facilita o envio e recebimento de dados de APIs REST usando anotações elegantes no Kotlin.

✅ Por que usar Retrofit?

- Permite definir endpoints como **interfaces**.
- Integra facilmente com Gson, Moshi, RxJava e Coroutines.
- Trata erros e respostas de forma simples.
- Muito mais limpo do que usar HttpUrlConnection.

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

🧰 2. Bibliotecas para consumir APIs no Android

🔄 Retrofit — A biblioteca mais usada para consumo de APIs REST

📦 Exemplo de uso:

```
interface ApiService {
    @GET("posts")
    fun getPosts(): Call<List<Post>>
}
```

Você só precisa criar uma **interface com anotações**, e o Retrofit faz o resto — cria o código necessário para fazer a chamada HTTP real.

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

🧰 2. Bibliotecas para consumir APIs no Android

🔄 Gson — Conversor de JSON para Objetos Kotlin

📌 O que é?

Gson é uma biblioteca do Google usada para **converter dados JSON em objetos Kotlin (ou Java)** e vice-versa.

Ela é especialmente útil quando usamos Retrofit, pois **transforma automaticamente** o JSON da resposta da API em objetos prontos pra usar no código.

✅ Por que usar Gson?

- Poupa o trabalho de fazer o parsing "na mão".
- Funciona de forma transparente com Retrofit.
- Permite converter tanto JSON simples quanto objetos aninhados.

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

🧰 2. Bibliotecas para consumir APIs no Android

🔄 Gson — Conversor de JSON para Objetos Kotlin

📦 **Exemplo:**

Se sua API retorna isso:

```
{
  "id": 1,
  "title": "My post",
  "body": "Content here"
}
```

Você só precisa de uma data class assim:

```
data class Post(
    val id: Int,
    val title: String,
    val body: String
)
```

O Gson, com Retrofit, já converte esse JSON em um objeto Post automaticamente.

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

🧰 2. Bibliotecas para consumir APIs no Android

🔄 Gson — Conversor de JSON para Objetos Kotlin

🧠 Como as duas trabalham juntas?

1. **Retrofit** faz a requisição HTTP (GET, POST, etc.).
2. **Gson** interpreta a resposta (em JSON) e transforma em objetos Kotlin.
3. O seu app usa esses objetos normalmente, como se fossem criados localmente.

🔗 Exemplo completo de integração:

```
val retrofit = Retrofit.Builder()
    .baseUrl("https://jsonplaceholder.typicode.com/")
    .addConverterFactory(GsonConverterFactory.create()) // Aqui entra o Gson!
    .build()

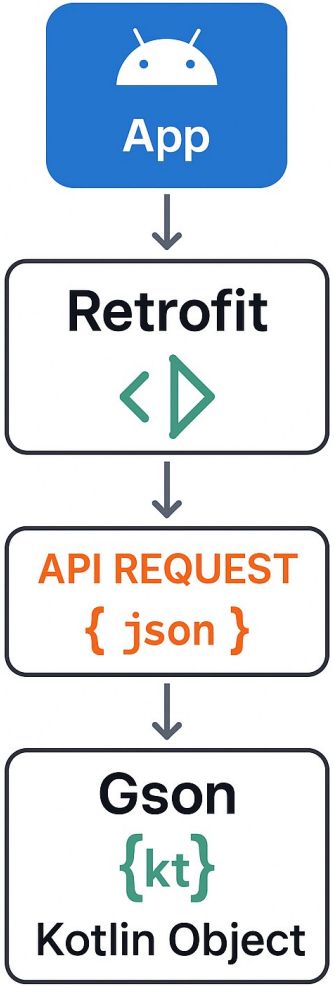
val api = retrofit.create(ApiService::class.java)
```

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

Fluxograma - Consumir uma API REST em um app Android usando as bibliotecas Retrofit e Gson.

🔁 Etapas explicadas

1. 📱 **App (Android)**
   - O usuário interage com o app.
   - O app precisa buscar dados de um servidor (ex: posts, usuários, etc).
2. 🔧 **Retrofit**
   - O Retrofit é responsável por **criar a requisição HTTP** com base na interface da API definida no código (@GET, @POST, etc).
   - Ele constrói a URL, envia a requisição e aguarda a resposta.
3. 🌐 **API Request (JSON)**
   - A requisição é enviada ao servidor (ex: https://api.exemplo.com/posts).
   - O servidor responde com um **JSON**, que é um texto estruturado contendo os dados solicitados.
4. 📦 **Gson → Kotlin Object**
   - O Gson **converte o JSON** recebido em um **objeto Kotlin** (data class) automaticamente.
   - Agora o app pode usar esse objeto como se fosse um dado local.





🛠️ Exemplo Prático
Consumindo uma API com Retrofit

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

🛠️ Exemplo Prático

Consumindo uma API com Retrofit

Configuração do AndroidManifest

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">

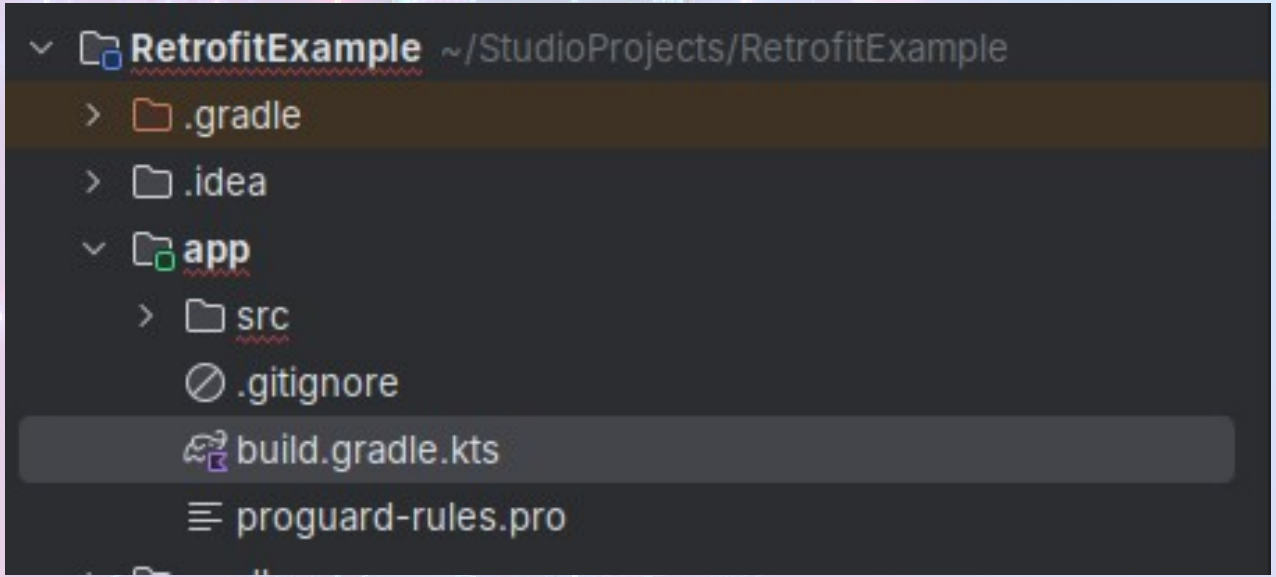
    <uses-permission android:name="android.permission.INTERNET"/>
```

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

🛠️ Exemplo Prático

Consumindo uma API com Retrofit

Configuração do build.gradle



UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

🛠️ Exemplo Prático

Consumindo uma API com Retrofit

Abrir o arquivo build.gradle (Module: app)

Esse é o local onde você adiciona dependências externas.

📌 **Adicione estas linhas na seção dependencies:**

```
implementation("com.squareup.retrofit2:retrofit:2.9.0")
implementation("com.squareup.retrofit2:converter-gson:2.9.0")
```

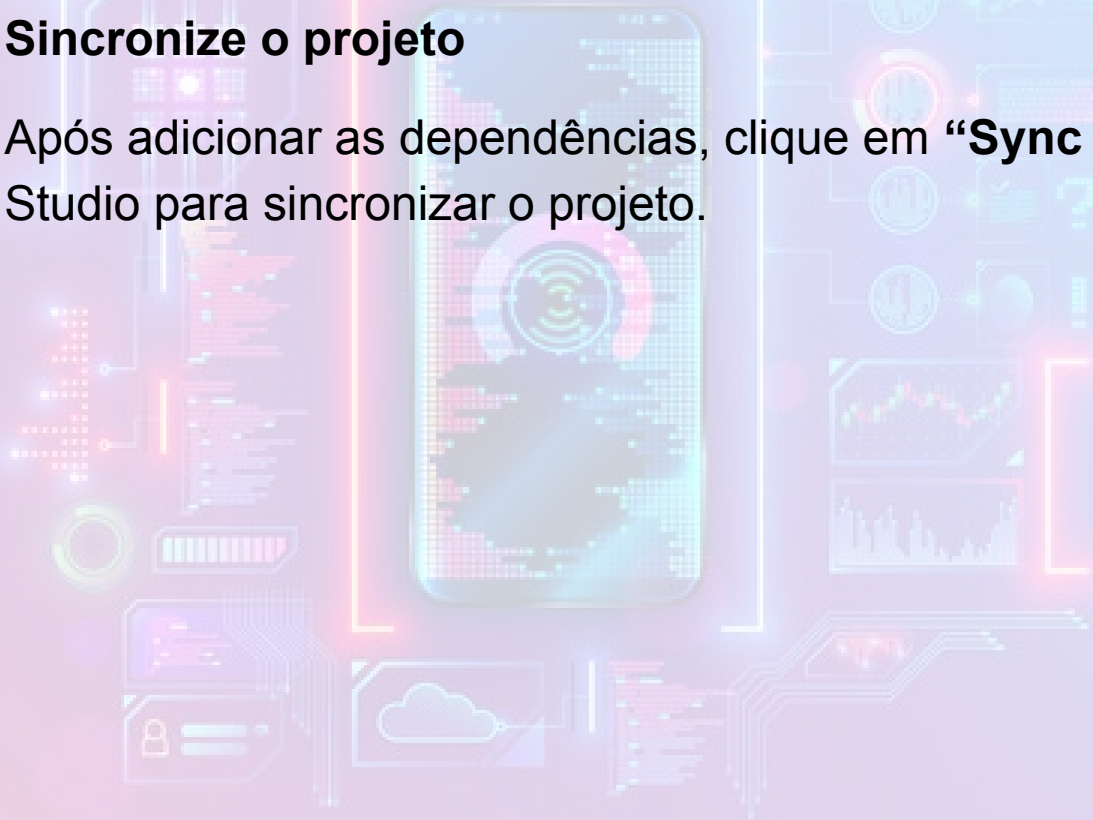
UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

🛠️ Exemplo Prático

Consumindo uma API com Retrofit

Sincronize o projeto

Após adicionar as dependências, clique em **“Sync Now”** no topo do Android Studio para sincronizar o projeto.



UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

🛠️ Exemplo Prático

Consumindo uma API com Retrofit

1. Criar o modelo de dados:

```
data class Post(
    val userId: Int,
    val id: Int,
    val title: String,
    val body: String
)
```

🛠️ Exemplo Prático

Consumindo uma API com Retrofit

2. Definir a interface da API:

```
import retrofit2.Call
import retrofit2.http.GET

interface ApiService {
    @GET("posts")
    fun getPosts(): Call<List<Post>>
}
```

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

🛠️ Exemplo Prático

Consumindo uma API com Retrofit

3. Criar o cliente Retrofit:

```
import retrofit2.Retrofit
import retrofit2.converter.gson.GsonConverterFactory

object RetrofitClient {
    private const val BASE_URL = "https://jsonplaceholder.typicode.com/"

    val instance: ApiService by lazy {
        val retrofit = Retrofit.Builder()
            .baseUrl(BASE_URL)
            .addConverterFactory(GsonConverterFactory.create())
            .build()

        retrofit.create(ApiService::class.java)
    }
}
```

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

🛠️ Exemplo Prático

Consumindo uma API com Retrofit

📲 Mostrar os dados no App (RecyclerView)

5. Activity Main:



UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

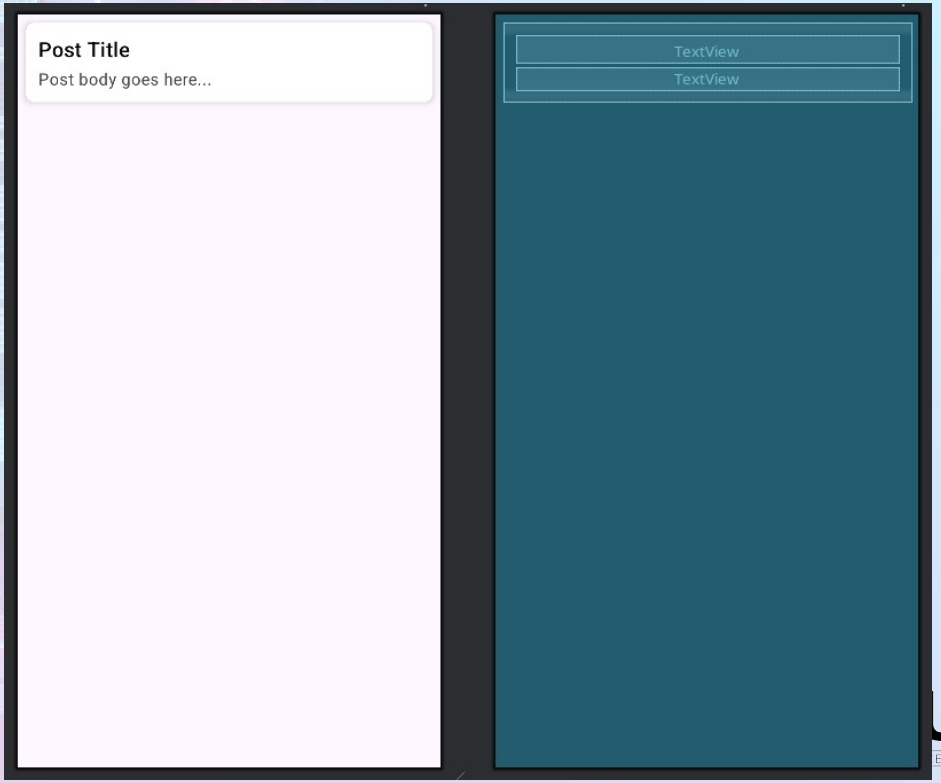
🛠️ Exemplo Prático

Consumindo uma API com Retrofit

📲 Mostrar os dados no App (RecyclerView)

6. Item Post:

Crie um arquivo item_post.xml



UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.cardview.widget.CardView xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:card_view="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    card_view:cardElevation="4dp"
    card_view:cardCornerRadius="8dp"
    android:layout_margin="8dp">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:padding="12dp">
```

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

```
<TextView
    android:id="@+id/titleText"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Post Title"
    android:textAppearance="?attr/textAppearanceHeadline6"
    android:textColor="#000000"
    android:maxLines="1"
    android:ellipsize="end" />
```


UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

```
        <TextView
            android:id="@+id/bodyText"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Post body goes here..."
            android:textAppearance="?attr/textAppearanceBody1"
            android:textColor="#444444"
            android:layout_marginTop="4dp"
            android:maxLines="3"
            android:ellipsize="end" />

    </LinearLayout>

</androidx.cardview.widget.CardView>
```

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

```kotlin
class PostAdapter(private val posts: List<Post>) : RecyclerView.Adapter<PostAdapter.PostViewHolder>() {

    class PostViewHolder(itemView: View) : RecyclerView.ViewHolder(itemView) {
        val title: TextView = itemView.findViewById(R.id.titleText)
        val body: TextView = itemView.findViewById(R.id.bodyText)
    }

    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): PostViewHolder {
        val view = LayoutInflater.from(parent.context).inflate(R.layout.item_post, parent, false)
        return PostViewHolder(view)
    }

    override fun onBindViewHolder(holder: PostViewHolder, position: Int) {
        val post = posts[position]
        holder.title.text = post.title
        holder.body.text = post.body
    }

    override fun getItemCount(): Int = posts.size
}
```

```kotlin
class MainActivity : AppCompatActivity() {
    private lateinit var recyclerView: RecyclerView

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContentView(R.layout.activity_main)
        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
        }

        recyclerView = findViewById(R.id.recyclerView)
        recyclerView.layoutManager = LinearLayoutManager( context: this)

        RetrofitClient.instance.getPosts().enqueue(object : Callback<List<Post>> {
            override fun onResponse(call: Call<List<Post>>, response: Response<List<Post>>) {
                if (response.isSuccessful) {
                    val posts = response.body() ?: emptyList()
                    recyclerView.adapter = PostAdapter(posts)
                }
            }

            override fun onFailure(call: Call<List<Post>>, t: Throwable) {
                Toast.makeText( context: this@MainActivity,  text: "Error: ${t.message}", Toast.LENGTH_LONG).show()
            }
        })

    }
}
```

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

```
class MainActivity : AppCompatActivity() {
    private lateinit var recyclerView: RecyclerView

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContentView(R.layout.activity_main)
        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
        }

        recyclerView = findViewById(R.id.recyclerView)
        recyclerView.layoutManager = LinearLayoutManager( context: this)
```

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

```
    RetrofitClient.instance.getPosts().enqueue(object : Callback<List<Post>> {
        override fun onResponse(call: Call<List<Post>>, response: Response<List<Post>>) {
            if (response.isSuccessful) {
                val posts = response.body() ?: emptyList()
                recyclerView.adapter = PostAdapter(posts)
            }
        }

        override fun onFailure(call: Call<List<Post>>, t: Throwable) {
            Toast.makeText( context: this@MainActivity,  text: "Error: ${t.message}", Toast.LENGTH_LONG).show()
        }
    })


}
}
```

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

```
package br.com.retrofitexample

import android.os.Bundle
import android.widget.Toast
import androidx.activity.enableEdgeToEdge
import androidx.appcompat.app.AppCompatActivity
import androidx.core.view.ViewCompat
import androidx.core.view.WindowInsetsCompat
import androidx.recyclerview.widget.LinearLayoutManager
import androidx.recyclerview.widget.RecyclerView
import retrofit2.Call
import retrofit2.Callback
import retrofit2.Response
```

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

🛠️ Exemplo Prático

Consumindo uma API com Retrofit

Resultado



UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

Github

https://github.com/Mobile-Device-Development/RetrofitExample



🧪 Atividade

Pesquise e:

- Crie uma segunda tela que permite enviar um novo post para a API via método POST.
  - Use EditText para o título e corpo, e um botão que faz a requisição.
- Adicione uma tela para abrir o item quando clicar.
- Qual é a vantagem de separar a camada de dados usando Retrofit?
- O que muda ao consumir uma API real com autenticação?
- Como tratar erros e validar respostas?

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

Bibliografia

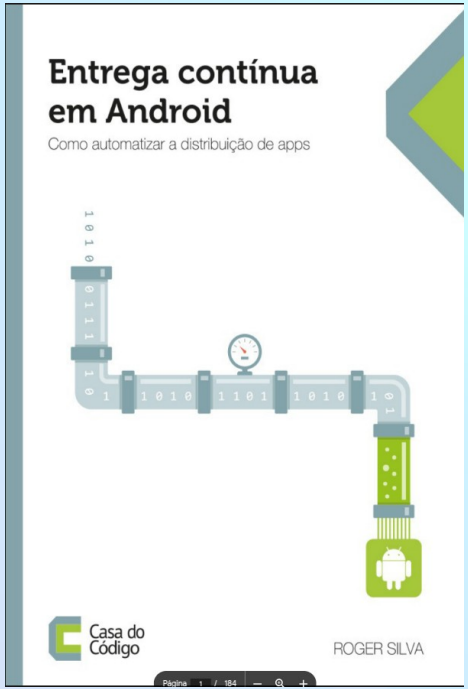
RESENDE, Kassiano. Kotlin com android: crie aplicativos de maneira fácil e divertida. São Paulo, SP: Casa do Código, 2018. E-book. Disponível em: https://plataforma.bvirtual.com.br. Acesso em: 26 fev. 2025.

https://plataforma.bvirtual.com.br/Acervo/Publicacao/213016

SILVA, Roger. Entrega contínua em android: como automatizar a distribuição de apps. São Paulo, SP: Casa do Código, 2016. E-book. Disponível em: https://plataforma.bvirtual.com.br. Acesso em: 26 fev. 2025.

https://plataforma.bvirtual.com.br/Acervo/Publicacao/212670

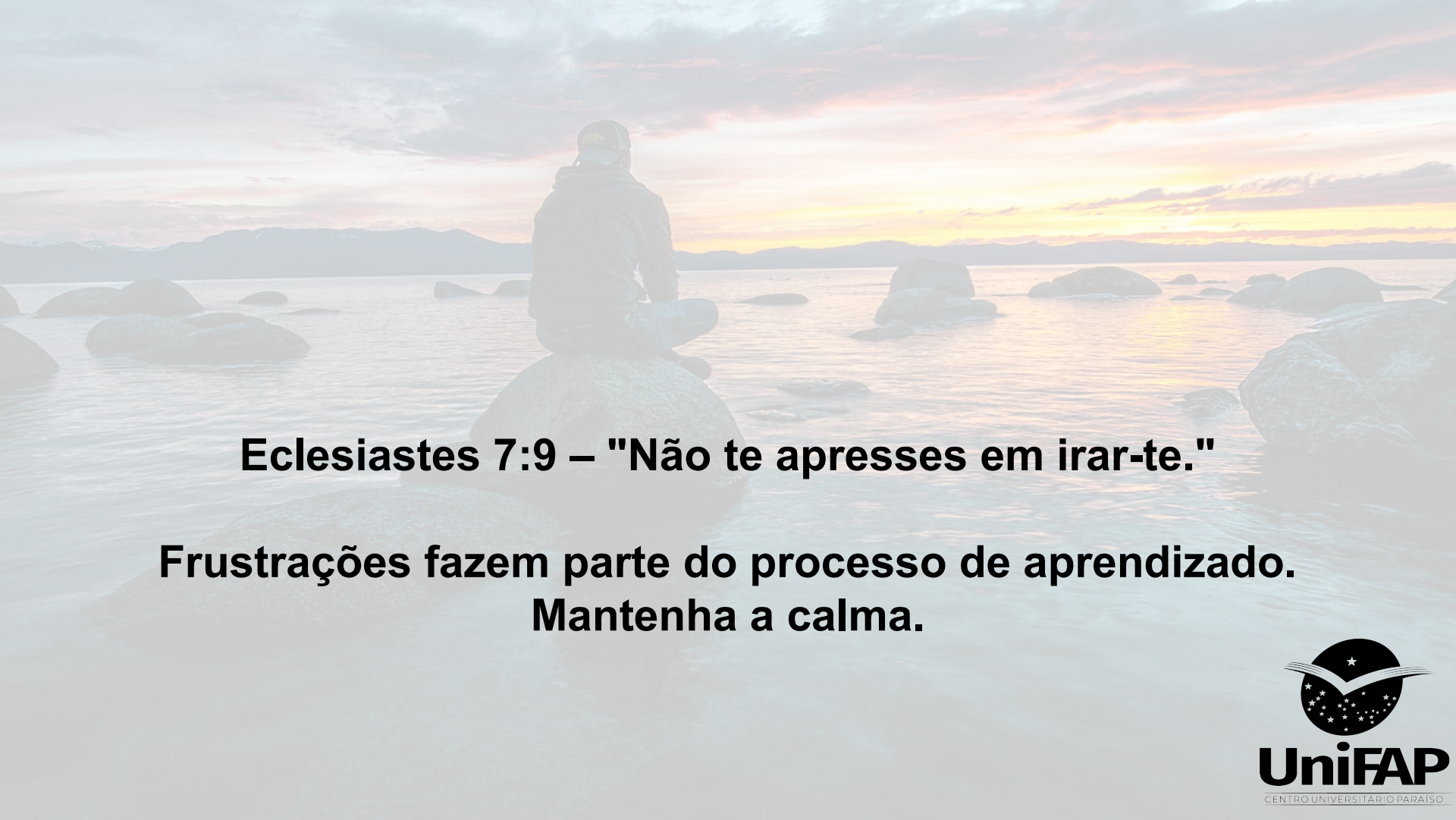




Perguntas?!



UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO



Eclesiastes 7:9 – "Não te apresses em irar-te."

Frustrações fazem parte do processo de aprendizado.
Mantenha a calma.

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO



Obrigado!

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO