# **Java e Kotlin** Juntos!? Construindo **Apps Android**

**Venilton FalvoJr**

Education Tech Lead na DIO
Doutorando no ICMC-USP

**@falvojr**

# Sobre Mim

**2007**

**Graduação**

Início da Graduação em Engenharia de Computação

**2008**

**1º Emprego**

Estágio como Dev Java/.NET

**2015**

**Enfim Mestre**

Título e carreira como Dev Mobile

**Hoje**

**Doutorado e DIO**

Doutorado e desafios como *Education Tech Lead* na DIO


credential.net/gy7a7rsn

*já expirou, mas foi top* 😅

# Objetivo Geral

Desenvolva os **eventos, integrações e interações de usuário em um App Android**. Para isso, vamos explorar conceitos essenciais, como: **APIs**, **Orientação a Objetos** e **Padrões de Projetos**; tudo na prática usando **Java e Kotlin!**

Nesse sentido, bibliotecas consolidadas como **Glide** e **Retrofit** serão utilizadas com o objetivo de aumentar ainda mais nossa produtividade e qualidade de código.

# Premissas/Revisão

Nosso App já está versionado no GitHub: disponível aqui.

Por isso, os seguintes conteúdos são pré-requisitos:

**C1.** **Desenvolvimento Mobile Nativo Para Android**
Branch: *release/desenvolvimento-mobile-nativo-para-android*

**C2.** **Componentes, Layouts e UI/UX Em Apps Android**
Branch: *release/componentes-layouts-ui-ux-em-apps-android*

# Percurso

**Parte 1** **Criando uma "API" e Modelando seu Domínio**

Parte 2 Conhecendo as Bibliotecas Retrofit e Glide

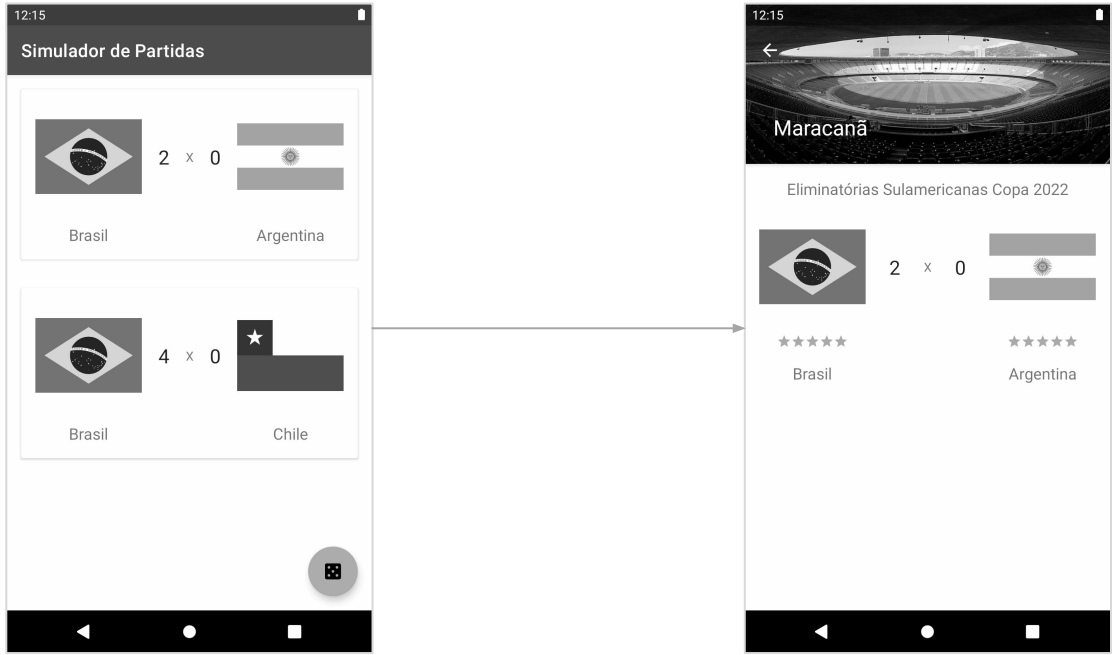
Parte 3 *Parcelable* e Simulação de Partidas 🎲

Parte 1

# Criando uma "API" e Modelando seu Domínio

// Java e Kotlin Juntos!? Construindo Apps Android

# Domínio/Problema




***Listagem/Simulação de Partidas***

***Detalhes da Partida***

# Domínio/Problema

Com base nos protótipos, conseguimos ter uma boa ideia do nosso domínio de aplicação. Nesse contexto, vamos **abstrair as entidades relevantes para o App**, por exemplo:

- Toda **Partida** é realizada em um **Local**;
- Uma **Partida** possui dois **Times** (mandante e visitante);
- Os **Times** têm um nível de força (estrelas)…

# “API” no GitHub Pages 🤨

Uma ***Application Programming Interface*** **(API)**, basicamente, se propõe a expor recursos de um domínio de aplicação. Seu principal objetivo é **definir uma interface para integrações concisas e eficientes**.

Nesse sentido, para que não tenhamos que construir uma API do zero, **vamos prover nossos recursos (partidas) via GET por meio do recurso *GitHub Pages***.

# “API” no GitHub Pages 🤯


```
[
  {
    "descricao": "Eliminatórias Copa 2022",
    "local": {
      "nome": "Maracanã",
      "imagem": "TODO"
    },
    "mandante": {
      "nome": "Brasil",
      "estrelas": 5,
      "imagem": "https://www.bandeirasnacionais.com/data/flags/normal/br.png"
    },
    "visitante": {
      "nome": "Argentina",
      "estrelas": 5,
      "imagem": "https://www.bandeirasnacionais.com/data/flags/normal/ar.png"
    }
  }
  ...
]
```

## Hands On! **Criando a "API" e Classes de Domínio**

*"Falar é fácil.*
*Mostre-me o código!"*

**Linus Torvalds**

# Percurso

~~Parte 1~~ ~~Criando uma "API" e Modelando seu Domínio~~

**Parte 2** **Conhecendo as Bibliotecas Retrofit e Glide**

Parte 3 *Parcelable* e Simulação de Partidas 🎲

Parte 2

# Conhecendo as Bibliotecas Retrofit e Glide

// Java e Kotlin Juntos!? Construindo Apps Android

# Retrofit

**Cliente HTTP** para Android e Java, o qual abstrai incrivelmente a complexidade no **consumo de APIs**. Além disso, possui uma série de conversores JSON, que facilitam a (de)serialização dos dados:

```
// Retrofit (HTTP Client): https://square.github.io/retrofit

implementation 'com.squareup.retrofit2:retrofit:2.9.0'
implementation 'com.squareup.retrofit2:converter-gson:2.9.0'
```

*Nota: Ao obter as partidas da API, vamos listá-las dinamicamente com o RecyclerView.*

# Glide

**Gerenciador de mídia** rápido e eficiente, abstraindo o processo de **carregamento de imagens** em Android, gerenciando desde a decodificação e transformação até o controle de cache das mesmas:

```
// Glide (Image Loading): https://github.com/bumptech/glide

implementation 'com.github.bumptech.glide:glide:4.11.0'
annotationProcessor 'com.github.bumptech.glide:compiler:4.11.0'
```

# *"Falar é fácil. Mostre-me o código!"*

**Linus Torvalds**

# Percurso

~~Parte 1~~ ~~Criando uma "API" e Modelando seu Domínio~~

~~Parte 2~~ ~~Conhecendo as Bibliotecas Retrofit e Glide~~

**Parte 3** ***Parcelable*** **e Simulação de Partidas** 🎲

Parte 3

# *Parcelable* e Simulação de Partidas 🎲

// Java e Kotlin Juntos!? Construindo Apps Android

# *Parcelable*

*Parcelable* é a estratégia de (de)serialização padrão em Apps Android, ou seja, é a implementação utilizada para troca de mensagens entre as telas de um App. Nesse contexto, o **Kotlin Parcelize** é uma alternativa muito simples e efetiva:

```
plugins {
    ...
    id 'kotlin-parcelize'
}
```

## Hands On! **Finalizando Nosso App**

# *“Falar é fácil. Mostre-me o código!”*

**Linus Torvalds**

# Links Úteis

- **Repositório da API no GitHub**
  *DIO* *(main)*
  **Repositório do App no GitHub**
  *DIO (release/java-e-kotlin-juntos-construindo-apps-android)*
- **Retrofit**
  *Square*
- **Glide**
  *Bump Technologies*
- **Parcelable**
  *Google*

# Dúvidas?

> Fórum/Artigos

> Comunidade Online (Discord)