



Ionic

---

**PROF. GUILHERME A. MADALOZZO**

# Ionic

---

Framework criado para ser executado em cima do Angular, inicialmente

- Hoje não é mais assim
- Já se tornou independente

Possui possibilidades de ramificações para outras tecnologias

- Angular
- VueJS
- ReactJS
- JS puro

# Ionic

---

Surgiu para unir tecnologias vindas da web para o desenvolvimento de aplicativos

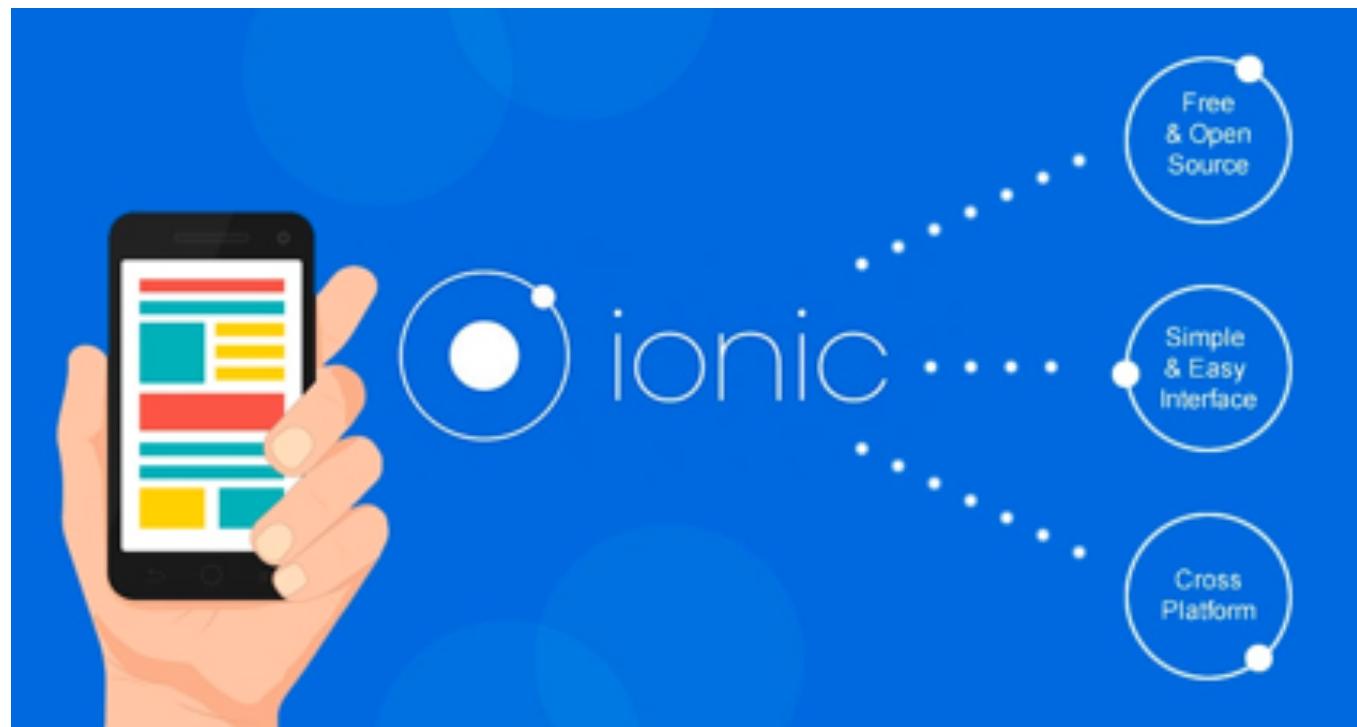
- A ideia era fugir do conceito nativo
- Aumenta o número de mão de obra
- Dissemina a ideia da tecnologia
- Reduz o custo de produção de aplicações mobile



# Ionic

---

Surgiu para unir tecnologias vindas da web para o desenvolvimento de aplicativos



# Ionic

---

Possibilita acesso a hardware nativo

- Logicamente, em alguns casos, o desempenho pode não ser o esperado
- Porém, vale a pena pelo custo e agilidade para entrada no mercado

# Ionic

<https://ionicframework.com>

The screenshot shows the official Ionic Framework website. At the top, there's a navigation bar with links for Platform, Developers, Enterprise, Pricing, Blog, Support, Log in, and a prominent blue 'Get Started' button. Below the navigation, the text 'Ionic Framework 5' is displayed, followed by a large, bold headline: 'Design. Animation. Performance.' A bulleted list of features follows: ✓ New UI components and gestures, ✓ Brand new custom animations API, ✓ New icons, colors, and theming, and ✓ Official React & Angular integrations. To the right of the text, there are several examples of Ionic UI components: a search bar with a large title, an alert message, a social media feed with icons for Music, Movies, and Games, a friends list with Justin Vernon, a toggle switch, a message card from Keanu Reeves, and a slider.

# Ionic

---

A documentação do framework é bem elaborada

Podemos evoluir o conhecimento da tecnologia acompanhando seus documentos

# Ionic

---

A documentação do framework é bem elaborada

Podemos evoluir o conhecimento da tecnologia acompanhando seus documentos

Vamos fazer a instalação

<https://ionicframework.com/docs>

## Getting Started

Overview

Environment Setup

CLI Installation

Packages & CDN

Next Steps

# Ionic Framework

Ionic Framework is an open source UI toolkit for building performant, high-quality mobile and desktop apps using web technologies — HTML, CSS, and JavaScript — with integrations for popular frameworks like Angular and React.

Get started building by [installing Ionic](#) or following our [First App Tutorial](#) to learn the main concepts.



## Installation Guide

Step-by-step guides to setting up your system and installing the framework.

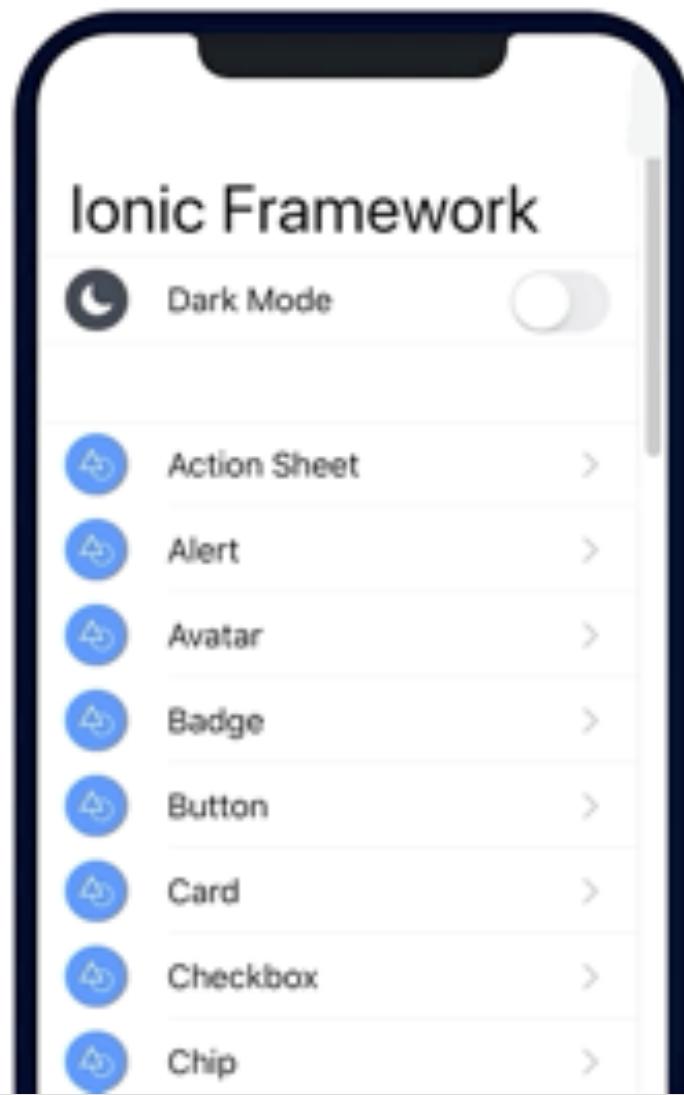


## UI Components

Dive into Ionic Framework's beautifully designed UI component library.

iOS

Android



# Ionic

# Environment Setup

To get started with Ionic Framework, the only requirement is a [Node & npm](#) environment.

Of course, a code editor is also required. [Visual Studio Code](#) is recommended. Visual Studio Code is a free, batteries-included text editor made by Microsoft.

## Terminal

Much of Ionic development requires familiarity with the command line. If you're new to the command line, see [this Blog Post](#) for a quick introduction.

In general, we recommend using the built-in terminals. Many third-party terminals work well with Ionic, but [may not be supported](#).

# Ionic

# Installing Ionic

Ionic apps are created and developed primarily through the Ionic command-line utility. The Ionic CLI is the preferred method of installation, as it offers a wide range of dev tools and help options along the way. It is also the main tool through which to run the app and connect it to other services, such as Ionic Appflow.

## Install the Ionic CLI

Before proceeding, make sure your computer has [Node.js](#) installed. See [these instructions](#) to set up an environment for Ionic.

Install the Ionic CLI with npm:

```
$ npm install -g @ionic/cli
```

# Ionic

---

## Ionic Packages

Ionic provides different packages that can be used to quickly get started using Ionic Framework or Ionicons in a test environment, Angular, any other framework, or none at all.

### Ionic Framework CDN

Ionic Framework can be included from a CDN for quick testing in a Plunker, Codepen, or any other online code editor!

It's recommended to use jsdelivr to access the Framework from a CDN. To get the latest version, add the following inside the `<head>` element in an HTML file, or where external assets are included in the online code editor:

[Copy](#)

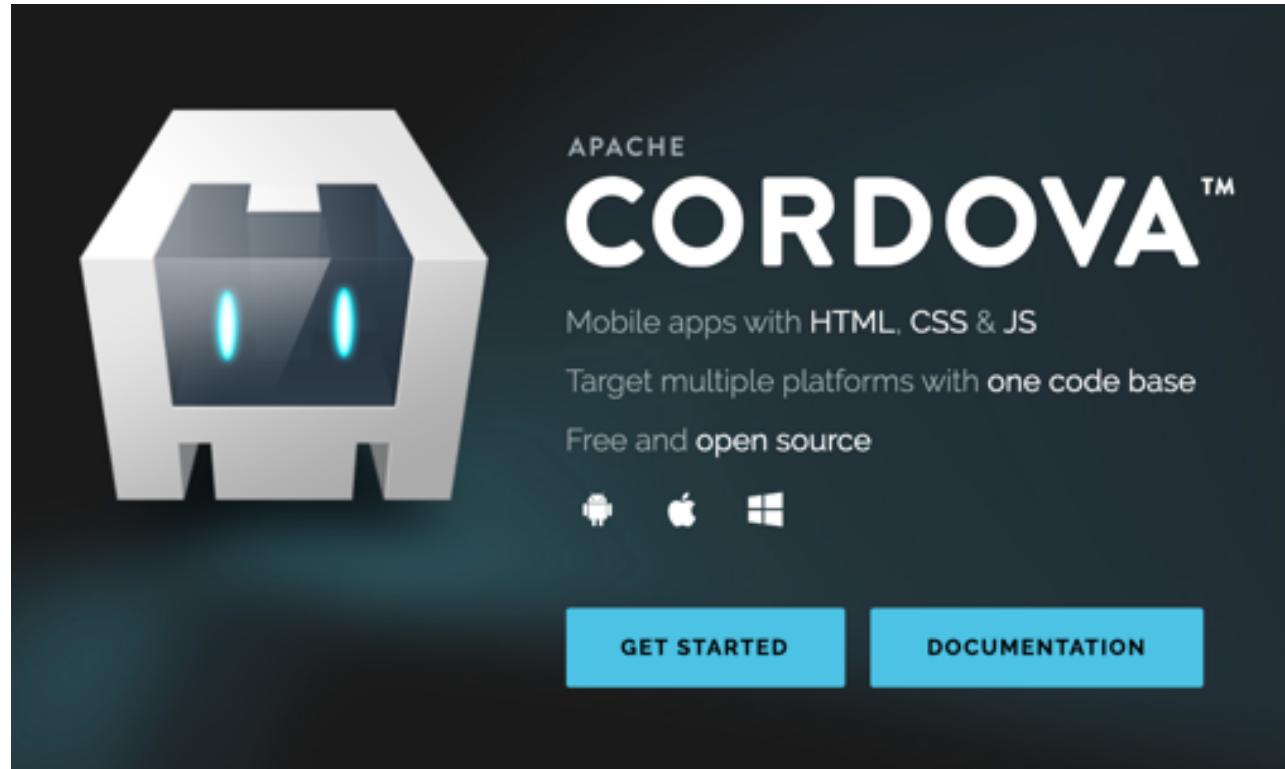
[HTML](#)

```
<script type="module" src="https://cdn.jsdelivr.net/npm/@ionic/core/dist/ionicons.umd.module.js"></script>
```

# Multiplataforma

Ionic é um framework multi-plataforma

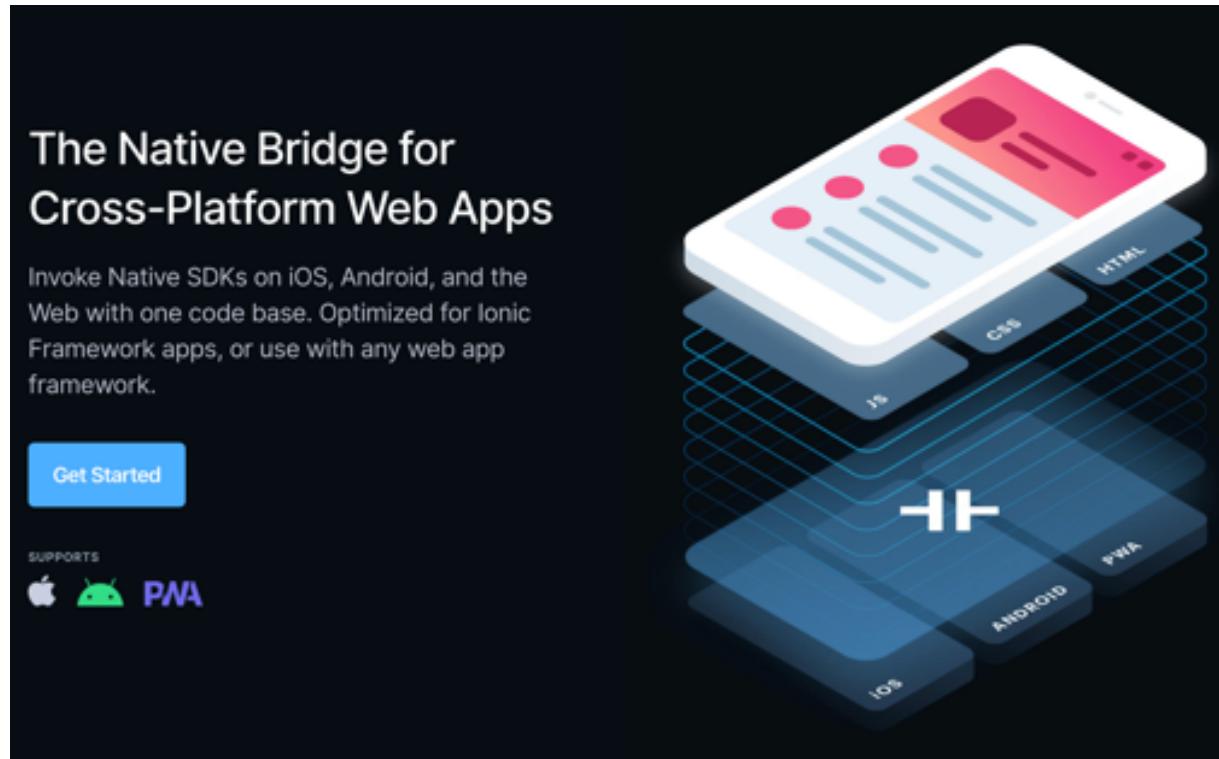
- Para que isso ocorra, ele trabalha com o Cordova (antigo PhoneGAP) desde a versão 1.x



# Multiplataforma

Ionic é um framework multi-plataforma

- Também, na versão 4.x adicionaram o uso do Capacitor



# Installing

# Installing

## Mac OSx

- Instalação do Java ([https://www.java.com/pt\\_BR/download/mac\\_download.jsp](https://www.java.com/pt_BR/download/mac_download.jsp))
- Instalação do Android Studio (<https://developer.android.com/studio/?hl=pt-br>)
- Instalação do NodeJS (<https://nodejs.org/en/>) dê preferencia para versão LTS
- Ionic CLI - (<https://ionicframework.com/getting-started#cli>), dê preferência à versão ionic;
- Instalação do VSCode (<https://code.visualstudio.com/docs/?dv=osx>)
- Baixe e coloque na pasta Documents o Gradle (<https://gradle.org/next-steps/?version=4.10.2&format=bin>)

Após instalar todos os softwares necessários precisamos configurar as variáveis de ambiente, para isso você pode instalar o vim ou nano, que são editores de terminal.

Com o editor já disponível abra o terminal e digite sudo vim `~/.bash_profile`, e adicione os seguintes caminhos

```
export JAVA_HOME =/Library/Java/JavaVirtualMachines/jdkVERSAO/Contents/Home  
export ANDROID_HOME=/Users/SEU-USUARIO/Library/Android/sdk  
export PATH=$ANDROID_HOME/platform-tools:$PATH  
export PATH=$ANDROID_HOME/tools:$PATH  
export PATH=$ANDROID_HOME/tools/bin:$PATH  
export PATH=$ANDROID_HOME/build-tools:$PATH  
export PATH=$ANDROID_HOME/emulator:$PATH  
export PATH=$PATH:/Users/SEU-USUARIO/Documents/gradle-VERSAO/bin/  
export PATH
```

Após realizar configuração e salvar feche o terminal.

Para testar se as variáveis estão corretas, após reabrir o terminal digite acho \$JAVA\_HOME ou acho \$ANDROID\_HOME e deve ser exibido os caminhos que você colocou acima, se isso não acontecer tente verificar se seguiu o procedimento corretamente.

## Windows

- Instalação do Java ([https://www.java.com/pt\\_BR/download/mac\\_download.jsp](https://www.java.com/pt_BR/download/mac_download.jsp))
- Instalação do Android Studio (<https://developer.android.com/studio/?hl=pt-br>)
- Instalação do NodeJS (<https://nodejs.org/en/>) dê preferencia para versão LTS
- Ionic CLI - (<https://ionicframework.com/getting-started#cli>), dê preferência à versão ionic;
- Instalação do VSCode (<https://code.visualstudio.com/Download>)
- Baixe e coloque na pasta Documents o Gradle (<https://gradle.org/next-steps/?version=4.10.2&format=bin>)

pós a instalação de todos os pré-requisitos acima você precisa configurar no PATH do seu sistema.

1. Encontre o ícone "Meu Computador";
2. Clique com o botão direito em propriedades;
3. Configurações avançadas do Sistema;
4. Aba Avançado;
5. Variáveis de Ambiente;
6. Na sessão variáveis do sistema, clique em Novo..., digite o nome da variável **JAVA\_HOME** e no valor o caminho exemplo "`c:/program files/java/jdkVERSAO`";
7. Abra o seu Android Studio, na aba de boas vindas clique em **Configure**, depois em **Settings**, veja o caminho de onde foi instalado o sdk do android no campo **Android SDK Location**.
8. Voltamos ao Variáveis do sistema e clique novamente em Novo..., no nome da variável digite **ANDROID\_HOME** e no valor o caminho onde está o SDK do Android;
9. Encontre a variável já criada o **Path**, selecione, e clique em editar, agora adicione o caminho do seu SDK do Android usando a variável de ambiente criada acima utilizando o comando `%ANDROID_HOME%`;
10. Depois adicione mais algumas linhas de configuração à estava variável Path, conforme o texto abaixo:

```
%ANDROID_HOME%\platform-tools  
%ANDROID_HOME%\tools  
%ANDROID_HOME%\tools\bin  
%ANDROID_HOME%\build-tools  
%ANDROID_HOME%\emulator
```

Após realizar os procedimentos apresentados, abra o seu terminal e digite `java --version` se estiver correto o sistema deve retornar a versão do mesmo.

# Installing

---

Depois de tudo instalado, vamos se o IonicCLI está funcional

- Vou executar o comando de informações pelo terminal do VS Code
- Vamos utilizar bastante o terminal, pode ser um PowerShell (windows), terminal MacOS ou Linux ou o próprio terminal do VS Code
- Também, a IDE pode ser de sua preferência

Para analisarmos as informações do IonicCLI, vamos executar no terminal

→ `ionic info`

# Installing

Depois de tudo instalado, vamos se o IonicCLI está funcional

- Vou executar o comando de informações pelo terminal do VS Code

Para analisarmos as informações do IonicCLI, vamos executar no terminal

→ ionic info

```
+ Dev ionic info
[WARN] You are not in an Ionic project directory. Project context may be missing.

Ionic:
  Ionic CLI : 5.4.16

Utility:
  cordova-res : not installed
  native-run  : not installed

System:
  NodeJS : v12.16.2
  npm   : 6.14.4
  OS    : macOS Catalina

Ionic CLI update available: 5.4.16 → 6.9.2
The package name has changed from ionic to @ionic/cli!
To update, run: npm uninstall -g ionic
Then run: npm i -g @ionic/cli
```

# Installing

Depois de tudo instalado, vamos se o IonicCLI está funcional

- Vou executar o comando de informações pelo terminal do VS Code

Para analisarmos as informações do IonicCLI, vamos executar no terminal

→ ionic info

```
→ Dev ionic info
[MARN] You are not in an Ionic project directory. Project context may be missing.

Ionic:
  Ionic CLI : 6.9.3

Utility:
  cordova-res : not installed
  native-run  : not installed

System:
  NodeJS    : v12.16.2
  npm       : 6.14.4
  OS        : macOS Catalina
```

# O Projeto

# O Projeto

---

Vamos desenvolver um marcador de pontuação

- O usuário poderá cadastrar e configurar jogos padrões
  - Truco paulista
  - Truco castelhano
  - Canastra
  - Fodinha
  - Pontinho
  - Etc...

# O Projeto

---

Vamos desenvolver um marcador de pontuação

- O usuário poderá criar uma nova partida com base em uma configuração de jogo
  - Quantidade de jogadores (os jogadores poderão acompanhar a pontuação da partida usando um QR Code)
  - Local do jogo (vamos usar a API do Google Maps para isso)
  - Jogadores poderão entrar na partida para registrar histórico de jogos
  - Poderá ter jogador sem registro no app, também

# O Projeto

---

Vamos desenvolver um marcador de pontuação

- Esta é uma visão geral do projeto
- Vamos tentar ver tudo em aula, caso não de tempo, terão material disponível com todo conteúdo

Ao término do projeto, vamos gerar a PWA para publicação nas lojas de aplicativos

# O Projeto

---

Para chegar no objetivo, vamos passar por algumas camadas

- Desenvolvimento da API para gerenciamento dos dados
  - Laravel/NodeJS/PythonFlask/Etc
- Desenvolvimento do aplicativo
  - Usaremos o novo conceito do Ionic, o Ionic Native
- Configuração API na Heroku
- Geração PWA
- Publicação do projeto

[GitHub](#)

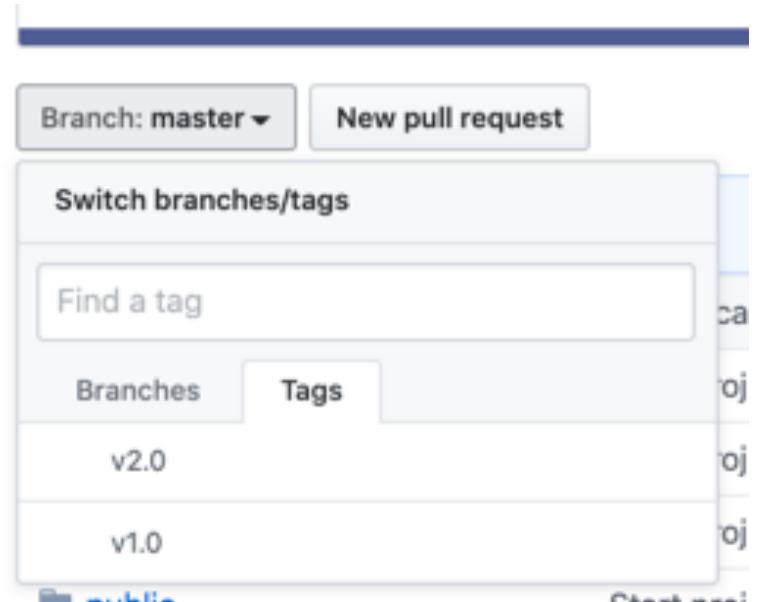
---

**HTTPS://GITHUB.COM/GUIMADALOZZO/MARCADONIC.GIT**

# GitHub

Todo o código apresentado nestes slides estarão disponíveis, em forma de TAGs, no GitHub

**<https://github.com/guimadalozzo/Marcadonic.git>**



Criando...

# Criando...

---

Vamos criar e configurar nosso primeiro projeto, Marcodonic

Para criarmos um novo projeto Ionic, vamos executar

→ ionic start marcadonic

# Criando...

Vamos criar e configurar nosso primeiro projeto, Marcodonic

Para criarmos um novo projeto Ionic, vamos executar

→ ionic start marcadonic

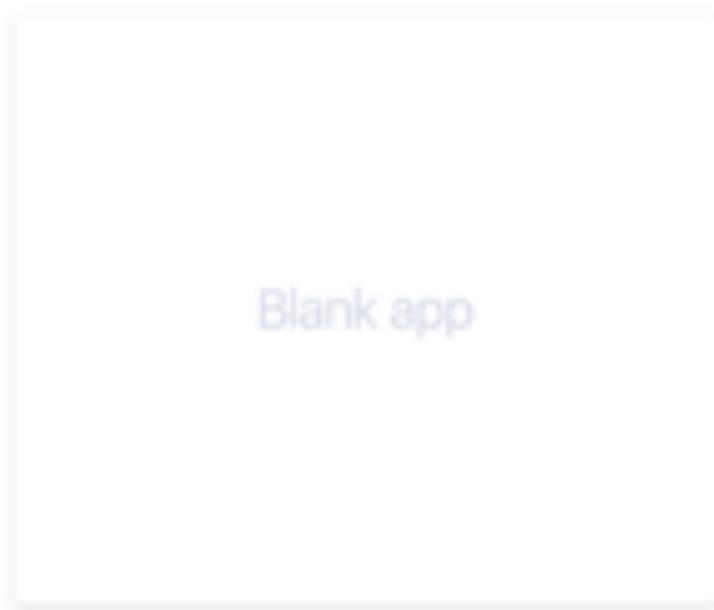
Temos que escolher o Framework que utilizaremos, no nosso caso será o **Angular**

```
→ Dev ionic start marcadonic
```

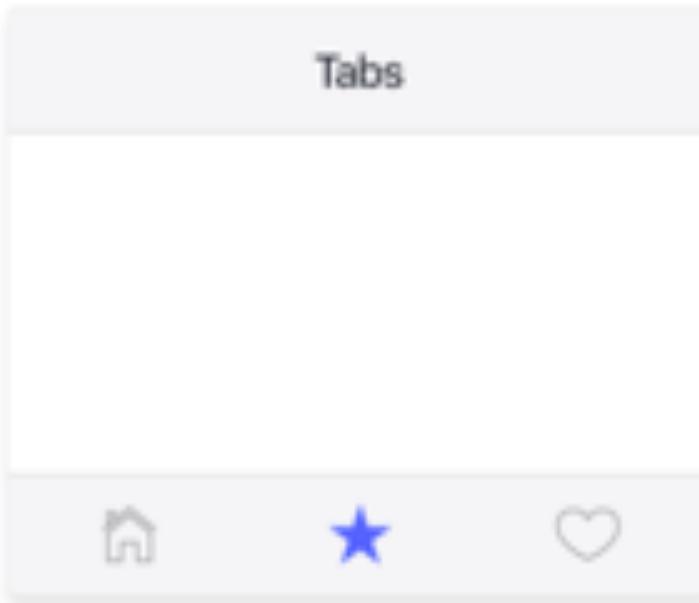
Pick a framework! 😊

```
Please select the JavaScript framework to use for your new app. To bypass this prompt next time, supply a value for the  
--type option.
```

```
? Framework: (Use arrow keys)  
➤ Angular | https://angular.io  
React | https://reactjs.org
```



\$ ionic start myApp blank



\$ ionic start myApp tabs



\$ ionic start myApp sidemenu

### ? Starter template:

tabs	A starting project with a simple tabbed interface
sidemenu	A starting project with a side menu with navigation in the content area
blank	<b>A blank starter project</b>
list	A starting project with a list
my-first-app	An example application that builds a camera with gallery
conference	A kitchen-sink application that shows off all Ionic has to offer

# Criando...

Vamos criar e configurar nosso primeiro projeto, Marcodonic

Para criarmos um novo projeto Ionic, vamos executar

→ ionic start marcadonic

Agora, o IonicCLI pede para escolhermos um template, escolheremos **blank**

? Framework: Angular

Let's pick the perfect starter template! ↗

Starter templates are ready-to-go Ionic apps that come packed with everything you need to build your app. To bypass this prompt next time, supply **template**, the second argument to **ionic start**.

? Starter template:

tabs	A starting project with a simple tabbed interface
sidemenu	A starting project with a side menu with navigation in the content area
> blank	<b>A blank starter project</b>
list	A starting project with a list
my-first-app	An example application that builds a camera with gallery
conference	A kitchen-sink application that shows off all Ionic has to offer

# Criando...

---

Vamos criar e configurar nosso primeiro projeto, Marcodonic

Para criarmos um novo projeto Ionic, vamos executar

→ ionic start marcadonic

Agora, o IonicCLI se queremos utilizar o Capacitor para melhor performance, podemos dizer que **sim (y)**

```
? Starter template: blank
✓ Preparing directory ./marcadonic - done!
✓ Downloading and extracting blank starter - done!
? Integrate your new app with Capacitor to target native iOS and Android? (y/N) y
```

# Criando...

---

Vamos criar e configurar nosso primeiro projeto, Marcodonic

Para criarmos um novo projeto Ionic, vamos executar

→ ionic start marcadonic

Por fim, vamos permitir ou não o compartilhamento de dados com o Angular Team → Google Analytics

? Would you like to share anonymous usage data with the Angular Team at Google under Google's Privacy Policy at <https://policies.google.com/privacy>? For more details and how to change this setting, see <http://angular.io/analytics>. (y/N) y█

# Criando...

---

Com o projeto criado, podemos executá-lo

Vamos executar

→ `ionic serve`

# Criando...

Com o projeto criado, podemos executá-lo

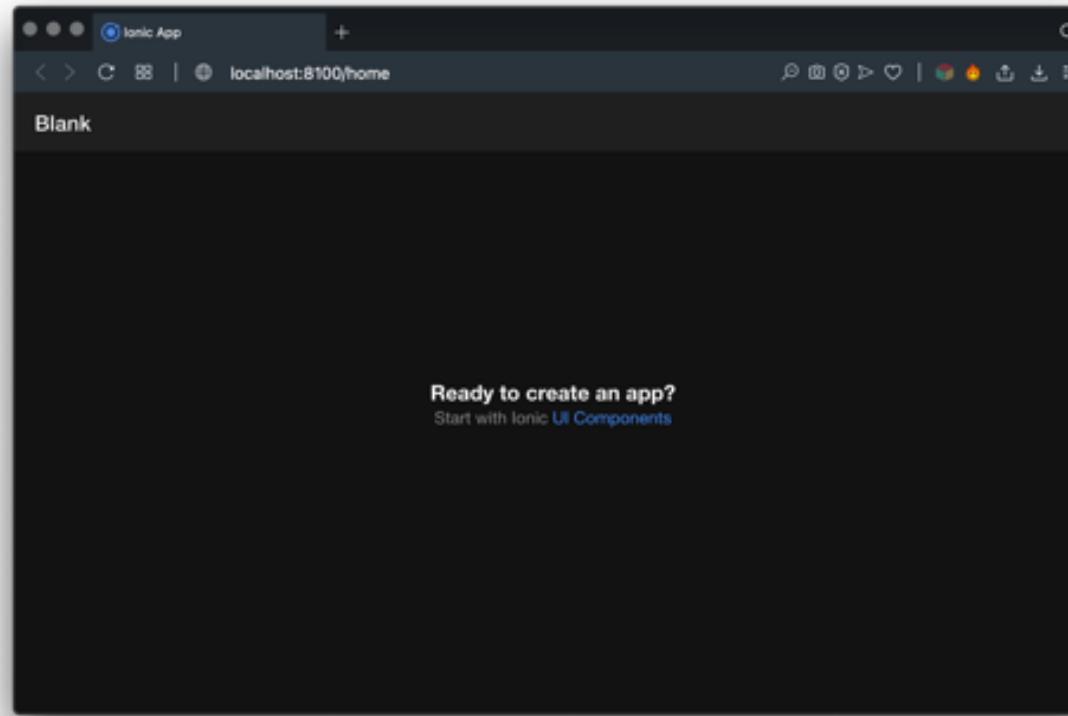
Vamos executar

→ ionic serve

```
→ marcadonic git:(master) ionic serve
> ng run app:serve --host=localhost --port=8100
[ng] Compiling @angular/core : es2015 as esm2015
[ng] Compiling @ionic-native/core : module as esm5
[ng] Compiling @angular/compiler/testing : es2015 as esm2015
[ng] Compiling @angular/core/testing : es2015 as esm2015
[ng] Compiling @ionic-native/splash-screen : module as esm5
[ng] Compiling @angular/common : es2015 as esm2015
[ng] Compiling @ionic-native/status-bar : module as esm5
[ng] Compiling @angular/platform-browser : es2015 as esm2015
[ng] Compiling @angular/router : es2015 as esm2015
[ng] Compiling @angular/common/http : es2015 as esm2015
[ng] Compiling @angular/platform-browser-dynamic : es2015 as esm2015
[ng] Compiling @angular/platform-browser/testing : es2015 as esm2015
```

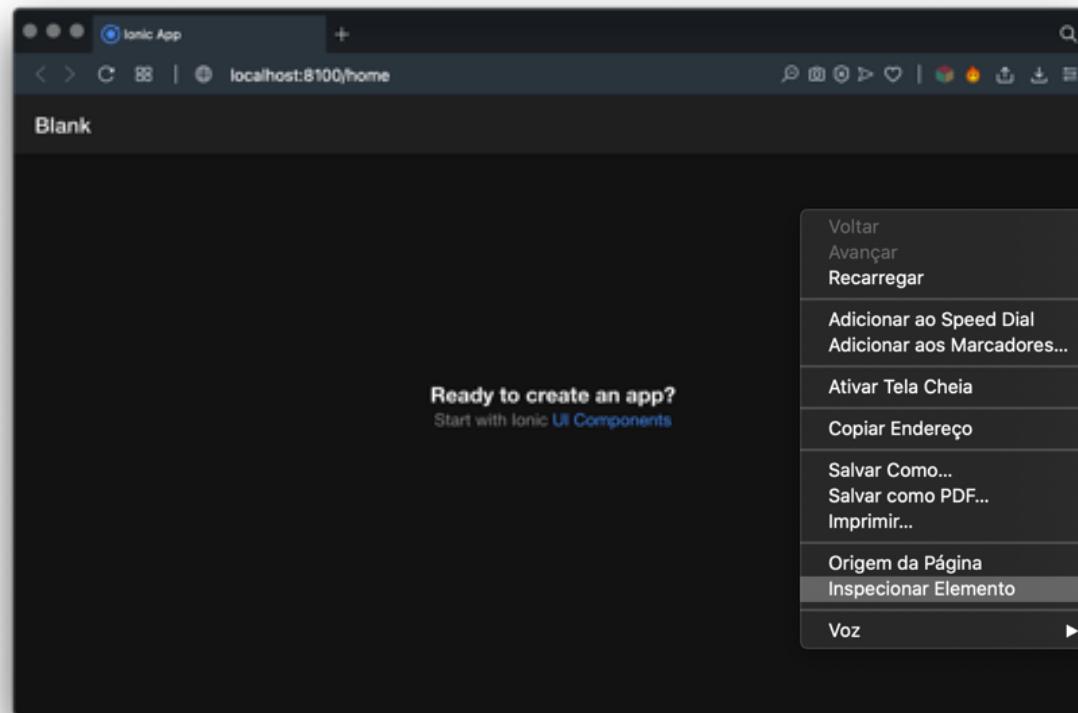
# Criando...

Quando executamos o projeto ele é iniciado no navegador, podemos testar por lá



# Criando...

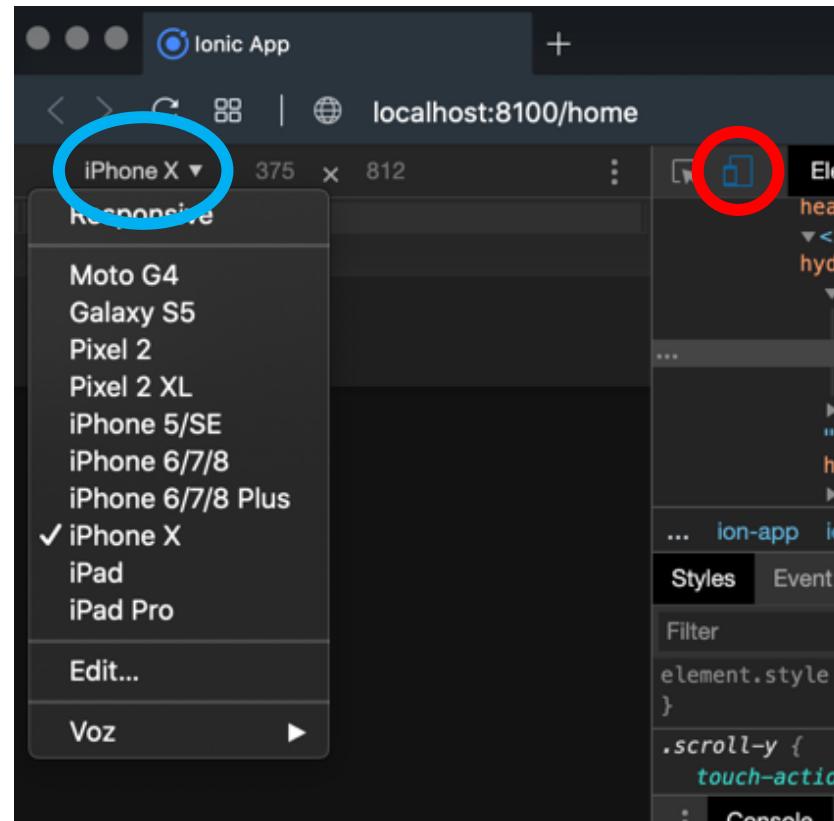
Quando executamos o projeto ele é iniciado no navegador, podemos testar por lá



# Criando...

Quando executamos o projeto ele é iniciado no navegador, podemos testar por lá

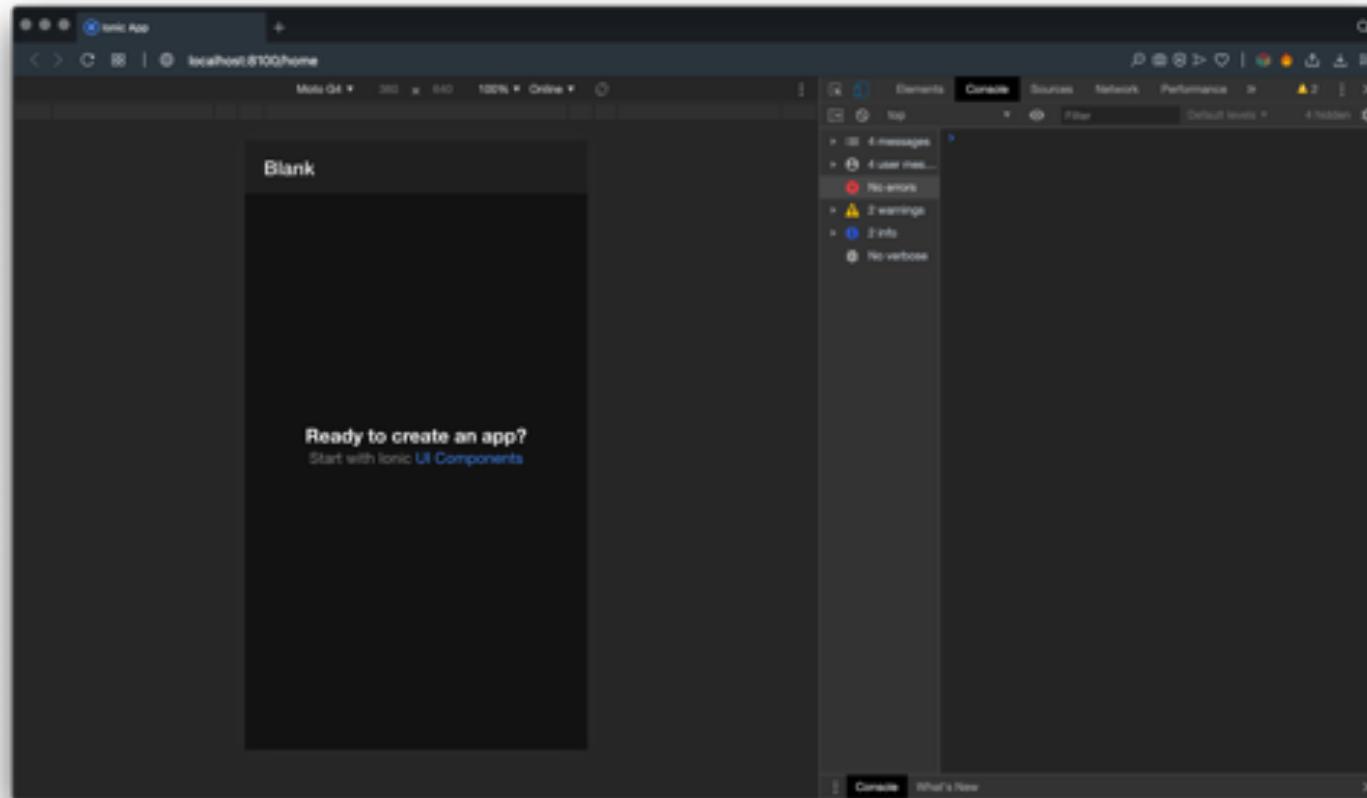
Vamos escolher um padrão de tela que desejamos



Vamos trocar a visualização para modo mobile

# Criando...

Quando executamos o projeto ele é iniciado no navegador, podemos testar por lá



# Criando...

---

Quando executamos o projeto ele é iniciado no navegador, podemos testar por lá



# Estrutura

# Estrutura

---

Vamos entender a estrutura gerada

- Abrindo o projeto no VS Code, vamos entender a estrutura de diretórios e arquivos

# Estrutura

```
✓ marcadonic
  > e2e
  > node_modules
  > src
    ◆ .gitignore
    { } angular.json
    └ browserlist
    { } capacitor.config.json
    📲 config.xml
    ● ionic.config.json
    ↻ karma.conf.js
    { } package-lock.json
    { } package.json
    { } tsconfig.app.json
    ✎ tsconfig.json
    { } tsconfig.spec.json
    { } tslint.json
```

# Estrutura

✓ marcadonic

> e2e

> node\_modules

> src

◆ .gitignore

{ } angular.json

☒ browserslist

{ } capacitor.config.json

RSS config.xml

● ionic.config.json

K karma.conf.js

{ } package-lock.json

{ } package.json

{ } tsconfig.app.json

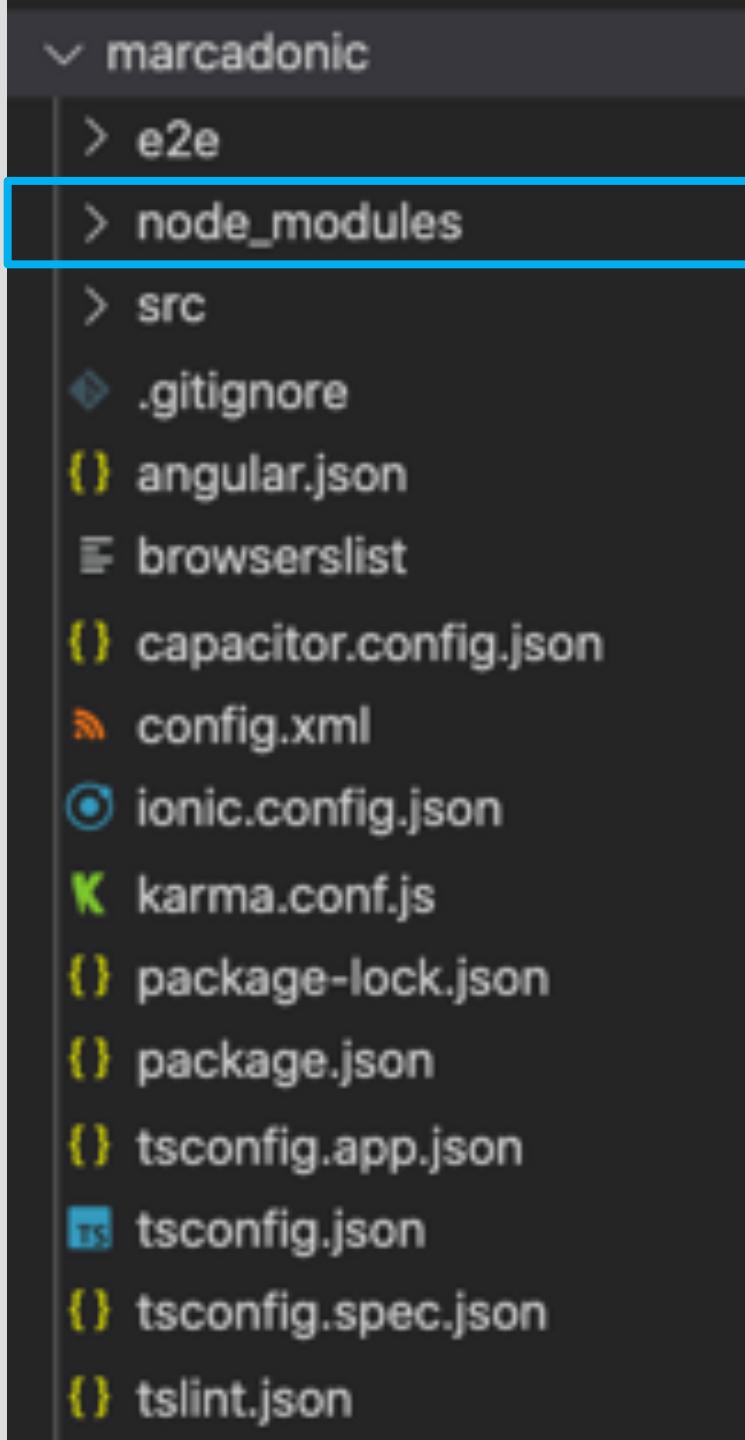
ts tsconfig.json

{ } tsconfig.spec.json

{ } tslint.json

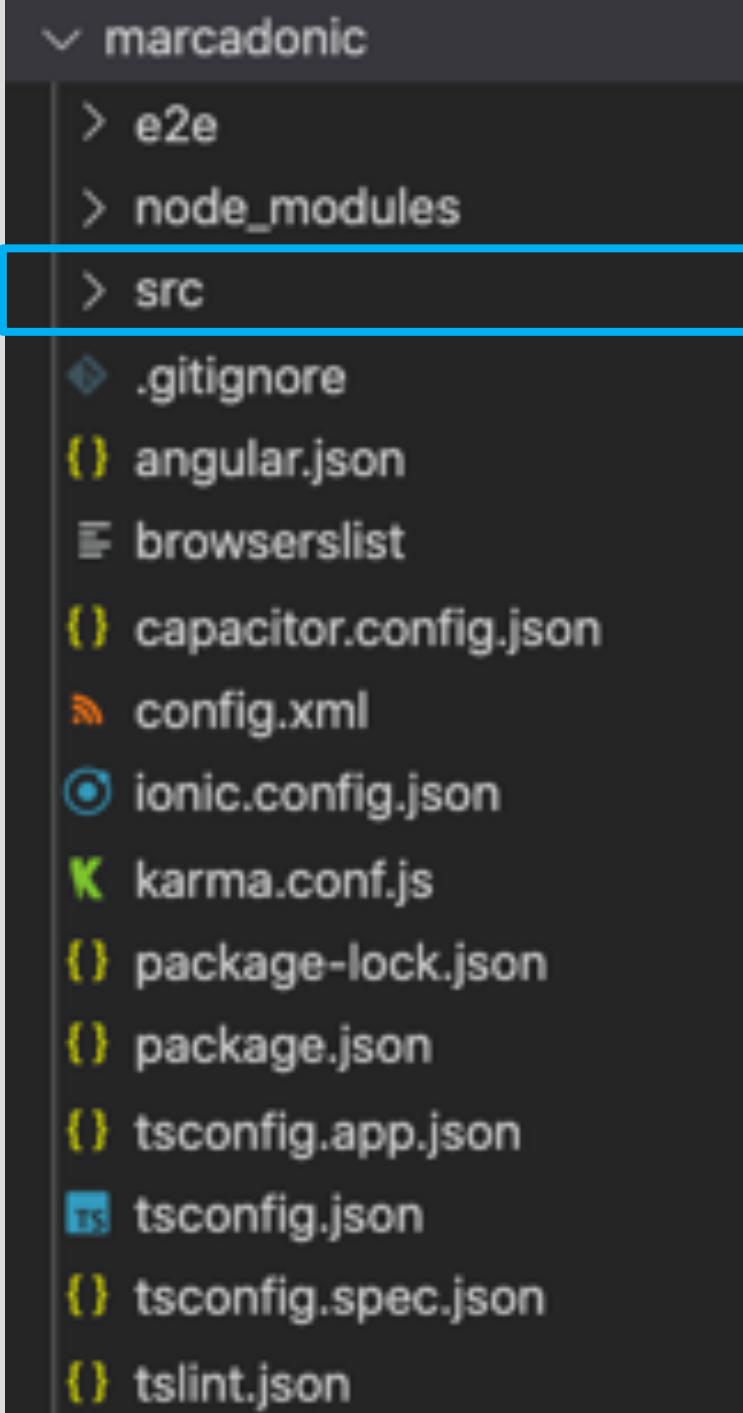
Pasta onde ficam os testes unitários do projeto

# Estrutura



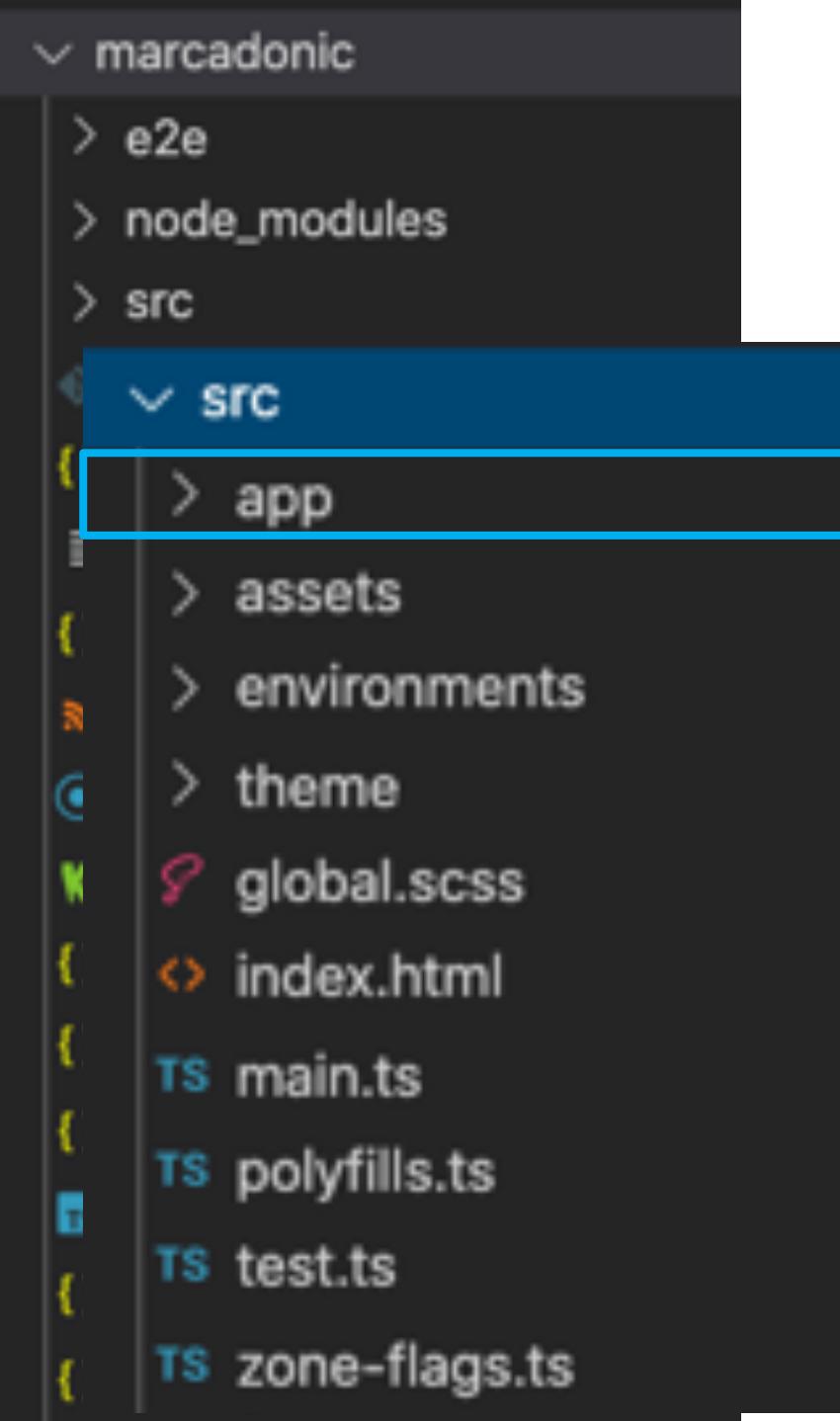
Pasta onde ficam os módulos do node, os padrões e os que podemos instalar ao longo do desenvolvimento

# Estrutura



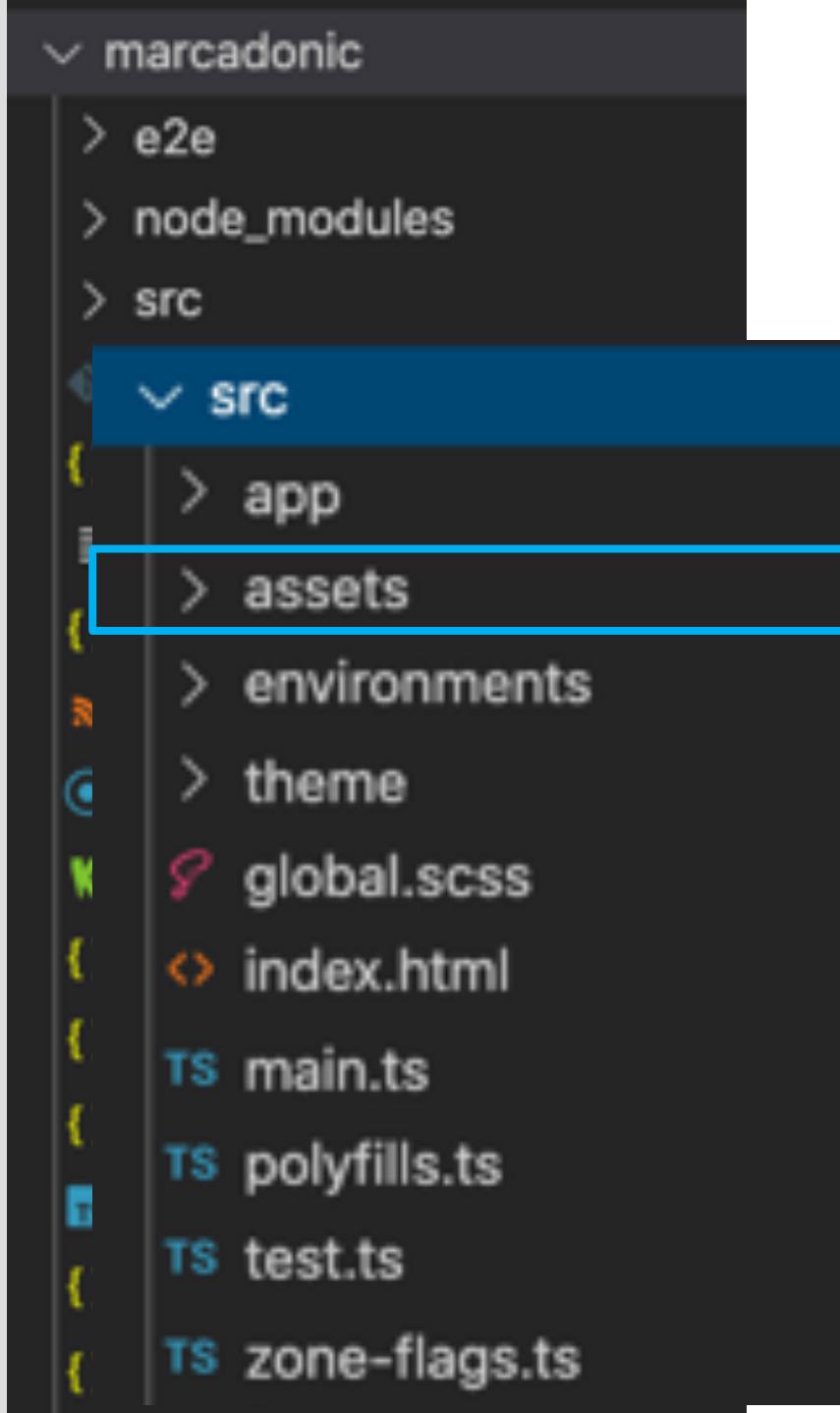
Pasta principal para o desenvolvimento

# Estrutura



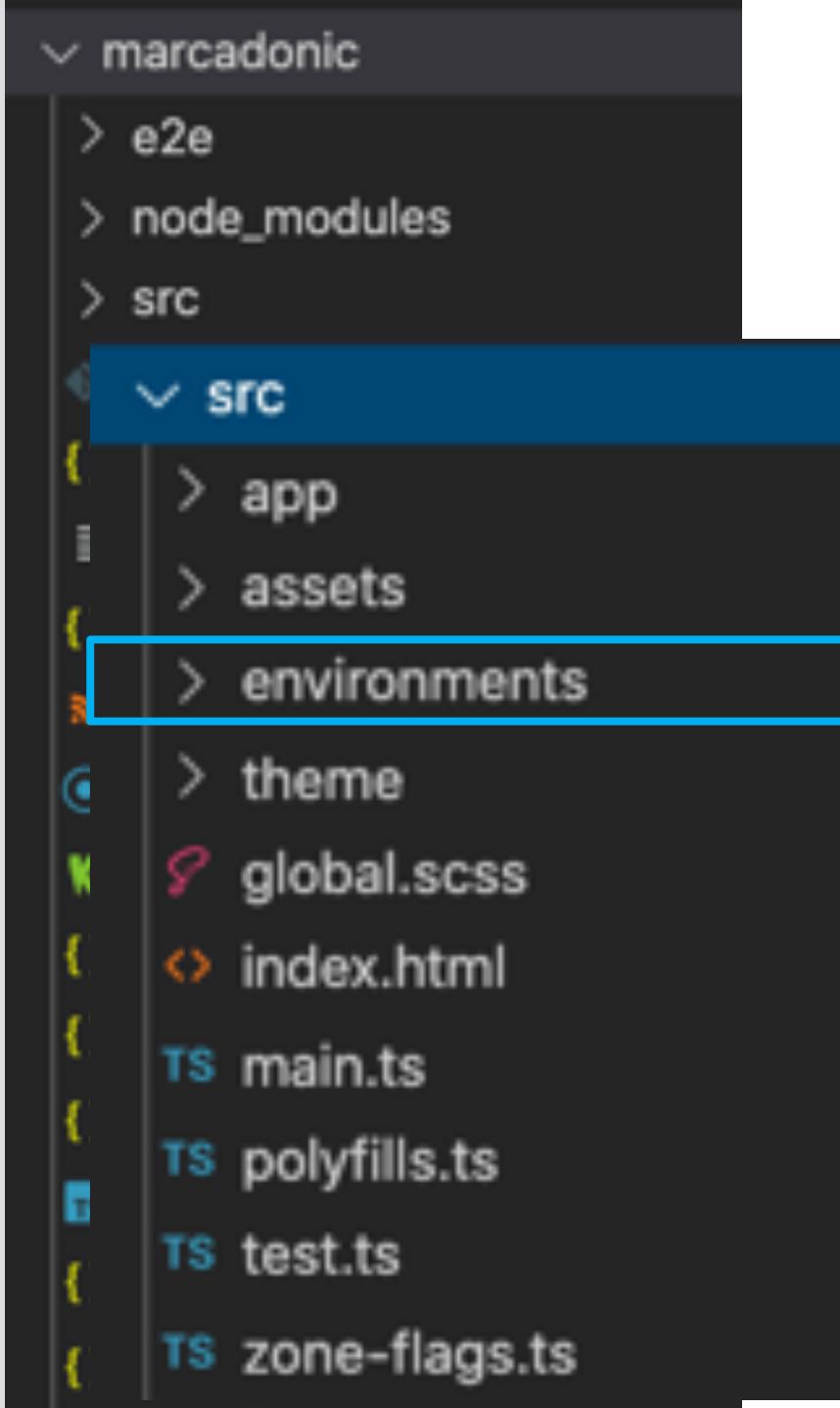
Pasta onde ficarão as páginas e componentes do aplicativo  
O componente atual (home) está configurado nela

# Estrutura



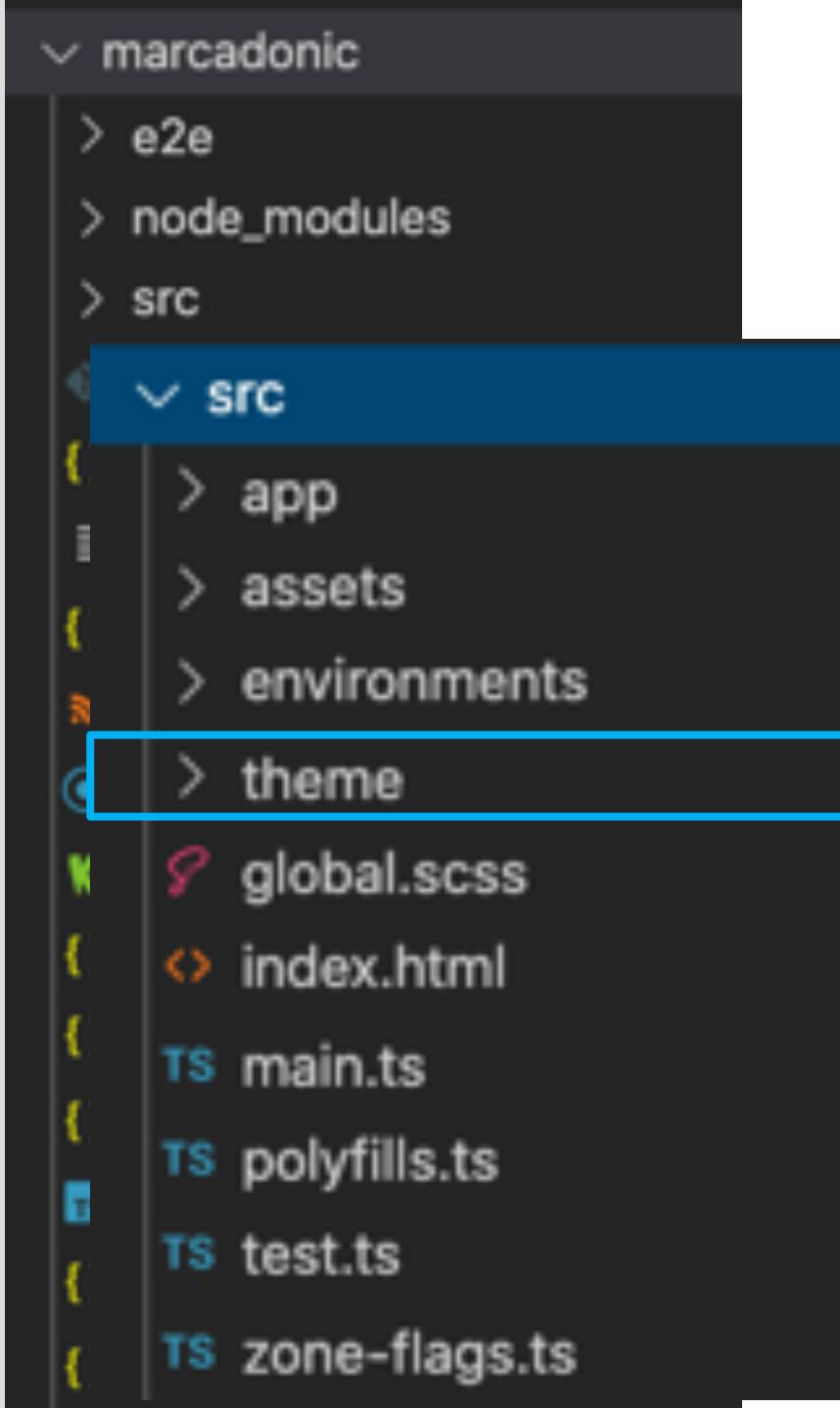
Pasta contendo imagens e outros artefatos necessários para a aplicação

# Estrutura



Pasta para gerenciamento dos ambientes de execução da aplicação

# Estrutura



Armazena os estilos globais da aplicação

# Estrutura

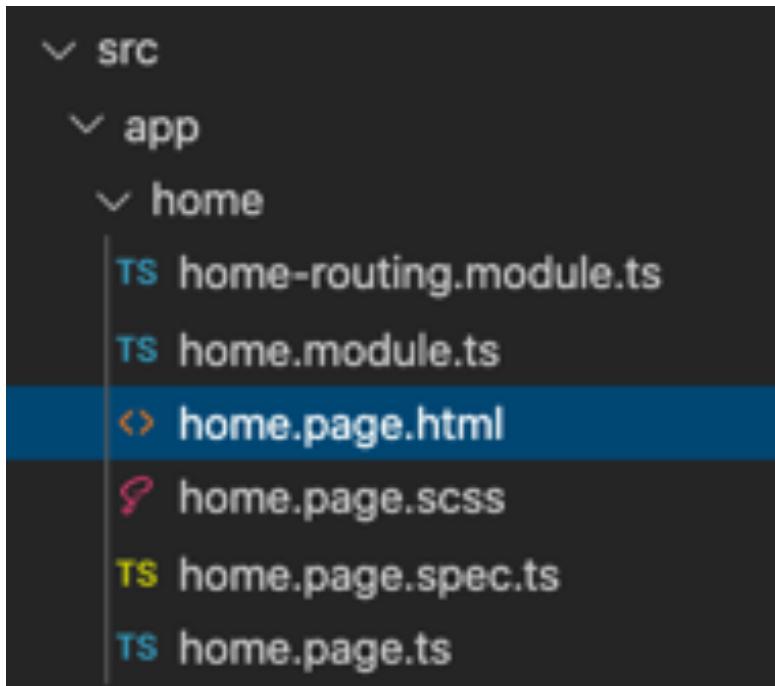
---

Vamos fazer uma simples alteração para verificar as mudanças no aplicativo

# Estrutura

Vamos fazer uma simples alteração para verificar as mudanças no aplicativo

- Entramos em src/app/ e depois vamos na page "home"
- Abriremos o html desta page → home.page.html



Vamos fazer uma simples alteração para verificar as mudanças no aplicativo

- Entramos em src/app/ e depois vamos na page "home"
- Abriremos o html desta page → home.page.html

## Estrutura

↳ home.page.html ×

```
marcadonic > src > app > home > ↳ home.page.html > ...
1   <ion-header [translucent]="true">
2     <ion-toolbar color="success">
3       <ion-title>
4         | Jogos
5       </ion-title>
6     </ion-toolbar>
7   </ion-header>
8
9   <ion-content [fullscreen]="true">
10    <ion-header collapse="condense">
11      <ion-toolbar>
12        | <ion-title size="large">Jogos</ion-title>
13      </ion-toolbar>
14    </ion-header>
15
16    <div id="container">
17      <strong>Sem jogos configurados!</strong>
18      <p>Vamos configurar um novo jogo?</p>
19    </div>
20  </ion-content>
```

# Estrutura

Vamos fazer uma simples alteração para verificar as mudanças no aplicativo

- Entramos em src/app/ e depois vamos na page "home"
- Abriremos o html desta page → home.page.html

```
↳ home.page.html ×  
marcadonic > src > app > home > ↳ home.page.html > ...  
1   <ion-header [translucent]="true">  
2     <ion-toolbar color="success">  
3       <ion-title>  
4         | Jogos  
5       </ion-title>  
6     </ion-toolbar>  
7   </ion-header>  
8  
9   <ion-content [fullscreen]="true">  
10    <ion-header collapse="condense">  
11      <ion-toolbar>  
12        | <ion-title size="large">Jogos</ion-title>  
13      </ion-toolbar>  
14    </ion-header>  
15  
16    <div id="container">  
17      | <strong>Sem jogos configurados!</strong>  
18      | <p>Vamos configurar um novo jogo?</p>  
19    </div>  
20  </ion-content>
```

Alteramos o estilo de cores para Success (verde)

# Estrutura

Vamos fazer uma simples alteração para verificar as mudanças no aplicativo

- Entramos em src/app/ e depois vamos na page "home"
- Abriremos o html desta page → home.page.html

```
↳ home.page.html ×  
marcadonic > src > app > home > ↳ home.page.html > ...  
1   <ion-header [translucent]="true">  
2     <ion-toolbar color="success">  
3       <ion-title>  
4         | Jogos  
5       </ion-title>  
6     </ion-toolbar>  
7   </ion-header>  
8  
9   <ion-content [fullscreen]="true">  
10    <ion-header collapse="condense">  
11      <ion-toolbar>  
12        | <ion-title size="large">Jogos</ion-title>  
13      </ion-toolbar>  
14    </ion-header>  
15  
16    <div id="container">  
17      | <strong>Sem jogos configurados!</strong>  
18      | <p>Vamos configurar um novo jogo?</p>  
19    </div>  
20  </ion-content>
```

Alteramos o estilo de cores para Success (verde)

```
✓ src  
  > app  
  > assets  
  > environments  
  ✓ theme  
    ↳ variables.scss
```

Podemos encontrar os estilos de cores globais em variables.scss

# Estrutura

Vamos fazer uma simples alteração para verificar as mudanças no aplicativo

- Entramos em src/app/ e depois vamos na page "home"
- Abriremos o html desta page → home.page.html

◦ home.page.html ×

marcadonic > src > app > home > ◦ home.page.html > ...

1   <ion-header [translucent] = "true">

29  
30    /\* success \*/

31    --ion-color-success: #2dd36f;  
32    --ion-color-success-rgb: 45, 211, 111;  
33    --ion-color-success-contrast: #ffffff;

34    --ion-color-success-contrast-rgb: 255, 255, 255;

35    --ion-color-success-shade: #28ba62;

36    --ion-color-success-tint: #42d77d;

37

15  
16    <div id = "container">  
17      <strong>Sem jogos configurados!</strong>  
18      <p>Vamos configurar um novo jogo?</p>  
19    </div>  
20  </ion-content>

Alteramos o estilo de cores para Success (verde)

Podemos encontrar os estilos de cores globais em variables.scss

# Estrutura

Vamos fazer uma simples alteração para verificar as mudanças no aplicativo

- Entramos em src/app/ e depois vamos na page "home"
- Abriremos o html desta page → home.page.html

↳ home.page.html ×

```
marcadonic > src > app > home > ↳ home.page.html > ...
1   <ion-header [translucent]="true">
2     <ion-toolbar color="success">
3       <ion-title>
4         | Jogos
5       </ion-title>
6     </ion-toolbar>
7   </ion-header>
8
9   <ion-content [fullscreen]="true">
10    <ion-header collapse="condense">
11      <ion-toolbar>
12        | <ion-title size="large">Jogos</ion-title>
13      </ion-toolbar>
14    </ion-header>
15
16    <div id="container">
17      | <strong>Sem jogos configurados!</strong>
18      | <p>Vamos configurar um novo jogo?</p>
19    </div>
20  </ion-content>
```

Alteramos o título da página  
para "Jogos"

# Estrutura

Vamos fazer uma simples alteração para verificar as mudanças no aplicativo

- Entramos em src/app/ e depois vamos na page "home"
- Abriremos o html desta page → home.page.html

◦ home.page.html ×

marcadonic > src > app > home > ◦ home.page.html > ...

```
1  <ion-header [translucent]="true">
2    <ion-toolbar color="success">
3      <ion-title>
4        | Jogos
5        </ion-title>
6      </ion-toolbar>
7    </ion-header>
8
9    <ion-content [fullscreen]="true">
10      <ion-header collapse="condense">
11        <ion-toolbar>
12          <ion-title size="large">Jogos</ion-title>
13        </ion-toolbar>
14      </ion-header>
15
16      <div id="container">
17        <strong>Sem jogos configurados!</strong>
18        <p>Vamos configurar um novo jogo?</p>
19      </div>
20    </ion-content>
```

Também, alteramos o título do conteúdo para "Jogos"

# Estrutura

Vamos fazer uma simples alteração para verificar as mudanças no aplicativo

- Entramos em src/app/ e depois vamos na page "home"
- Abriremos o html desta page → home.page.html

↳ home.page.html ×

```
marcadonic > src > app > home > ↳ home.page.html > ...
1   <ion-header [translucent]="true">
2     <ion-toolbar color="success">
3       <ion-title>
4         | Jogos
5       </ion-title>
6     </ion-toolbar>
7   </ion-header>
8
9   <ion-content [fullscreen]="true">
10    <ion-header collapse="condense">
11      <ion-toolbar>
12        | <ion-title size="large">Jogos</ion-title>
13      </ion-toolbar>
14    </ion-header>
15
16    <div id="container">
17      <strong>Sem jogos configurados!</strong>
18      <p>Vamos configurar um novo jogo?</p>
19    </div>
20  </ion-content>
```

Por fim, alteramos o conteúdo apresentado em tela

# Estrutura

Vamos fazer uma simples alteração para verificar as mudanças no aplicativo

- Entramos em src/app/ e depois vamos na page "home"
- Abriremos o html desta page → home.page.html
  
- Este é o resultado obtido nas alterações



# Estrutura

Vamos fazer uma simples alteração para verificar as mudanças no aplicativo

- Entramos em `src/app/` e depois vamos na page "home"
- Abriremos o html desta page → [home.page.html](#)
- Este é o resultado obtido nas alterações
- É importante salientar que o Ionic já vem configurado para respeitar as configurações do ambiente em execução
- Se mudarmos o padrão do navegador para modo escuro (dark), o Ionic irá se adaptar



# Estrutura

Vamos fazer uma simples alteração para verificar as mudanças no aplicativo

- Entramos em `src/app/` e depois vamos na page "home"
- Abriremos o html desta page → [home.page.html](#)
- Este é o resultado obtido nas alterações
- É importante salientar que o Ionic já vem configurado para respeitar as configurações do ambiente em execução
- Se mudarmos o padrão do navegador para modo escuro (dark), o Ionic irá se adaptar

Jogos

**Sem jogos configurados!**  
Vamos configurar um novo jogo?

# Iniciando o Projeto

[GitHub](#)

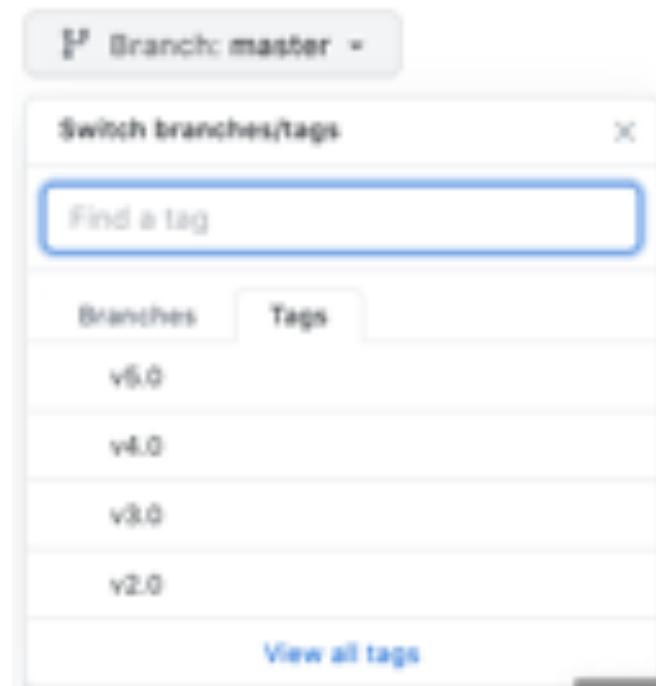
---

**HTTPS://GITHUB.COM/GUIMADALOZZO/PLAYINGSCORE**

# GitHub

Todo o código apresentado nestes slides estarão disponíveis, em forma de TAGs, no GitHub

**<https://github.com/guimadalozzo/PlayingScore>**



# Iniciando o Projeto

---

Vamos começar o projeto de nosso aplicativo

Como explicado na aula anterior, iOS e Android estão focando seus design em um padrão de tabs

- Então, não vamos fugir deste conceito
- Dessa forma, iremos criar um app com base em tabs

Sendo mais criativo, vamos criar o app chamado PlayingScore

```
$ ionic start PlayingScore tabs
```

# Iniciando o Projeto

Vamos começar o projeto de nosso aplicativo

Como explicado na aula anterior, iOS e Android estão focando seus design em um padrão de tabs

- Então, não vamos fugir deste conceito
- Dessa forma, iremos criar um app com base em tabs

Sendo mais criativo, vamos criar o app chamado PlayingScore

```
[+ Dev ionic start PlayingScore tabs
Pick a framework! 😊
Please select the JavaScript framework for your new app. To bypass this prompt next time, supply a value for the --type option.
? Framework: (Use arrow keys)
❯ Angular | https://angular.io
React   | https://reactjs.org
```



# Iniciando o Projeto

Vamos começar o projeto de nosso aplicativo

Como explicado na aula anterior, iOS e Android estão focando seus design em um padrão de tabs

- Então, não vamos fugir deste conceito
- Dessa forma, iremos criar um app com base em tabs

Sendo mais criativo, vamos criar o app chamado PlayingScore

```
? Framework: Angular
✓ Preparing directory ./PlayingScore - done!
✓ Downloading and extracting tabs starter - done!
? Integrate your new app with Capacitor to target native iOS and Android? (y/N) y
```



Diga SIM  
ao Capacitor

# Iniciando o Projeto

---

Vamos começar o projeto de nosso aplicativo

Como explicado na aula anterior, iOS e Android estão focando seus design em um padrão de tabs

- Então, não vamos fugir deste conceito
- Dessa forma, iremos criar um app com base em tabs



# GitHub

<https://github.com/guimadalozzo/PlayingScore>

# Iniciando o Projeto

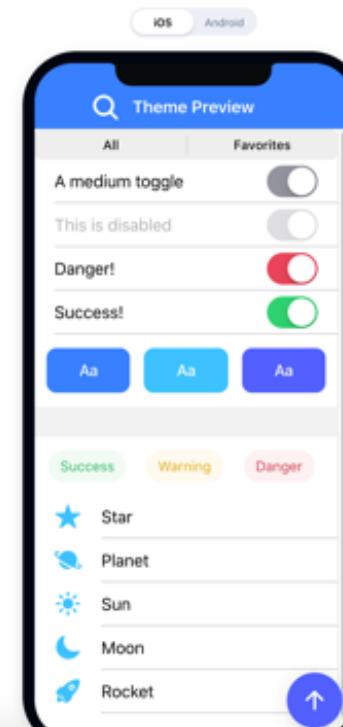
Como comentado anteriormente, o Ionic trabalha com um conceito de *themes*

- O site de documentação possui um gerador de tema, com base em uma cor desejada  
<https://ionicframework.com/docs/theming/color-generator>

## Color Generator

Create custom color palettes for your app's UI. Update a color's hex values, check the demo app on the right to confirm, then copy and paste the generated code directly into your Ionic project.

<input checked="" type="radio"/> Primary	#3880ff
<input checked="" type="radio"/> Secondary	#3dc2ff
<input checked="" type="radio"/> Tertiary	#5260ff
<input checked="" type="radio"/> Success	#2dd361
<input checked="" type="radio"/> Warning	#ffc409
<input checked="" type="radio"/> Danger	#eb445a
<input checked="" type="radio"/> Dark	#222428
<input checked="" type="radio"/> Medium	#92949c
<input checked="" type="radio"/> Light	#f4f5f8



# Iniciando o Projeto

Como comentado anteriormente, o Ionic trabalha com um conceito de *themes*

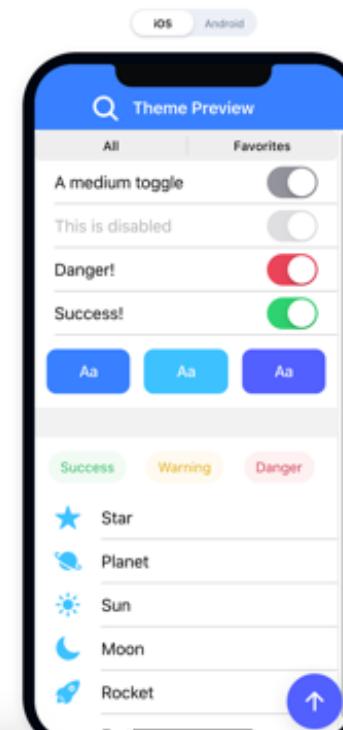
- O site de documentação possui um gerador de tema, com base em uma cor desejada  
<https://ionicframework.com/docs/theming/color-generator>

## Color Generator

Create custom color palettes for your app's UI. Update a color's hex values, check the demo app on the right to confirm, then copy and paste the generated code directly into your Ionic project.

<input checked="" type="radio"/> Primary	#3880ff
<input checked="" type="radio"/> Secondary	#3dc2ff
<input checked="" type="radio"/> Tertiary	#5260ff
<input checked="" type="radio"/> Success	#2dd361
<input checked="" type="radio"/> Warning	#ffc409
<input checked="" type="radio"/> Danger	#eb445a
<input checked="" type="radio"/> Dark	#222428
<input checked="" type="radio"/> Medium	#92949c
<input checked="" type="radio"/> Light	#f4f5f8

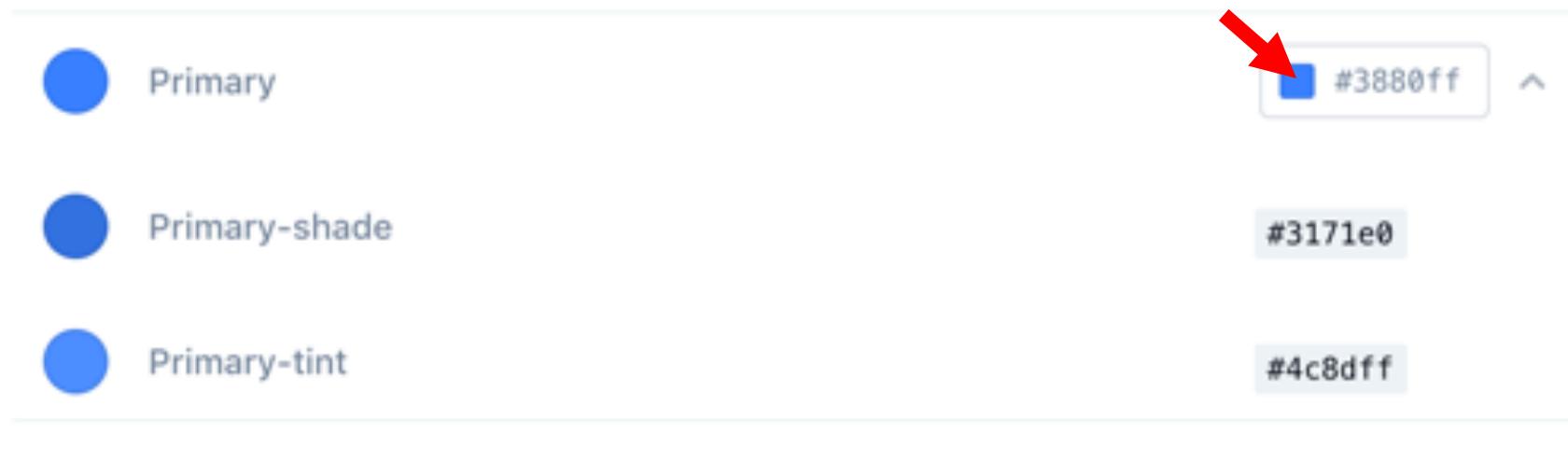
Dessa forma o Ionic já cria o tema conforme a cor padrão desejada



# Iniciando o Projeto

Como comentado anteriormente, o Ionic trabalha com um conceito de *themes*

- O site de documentação possui um gerador de tema, com base em uma cor desejada  
<https://ionicframework.com/docs/theming/color-generator>



The screenshot shows a color palette interface from the Ionic documentation. It displays three color swatches with their corresponding hex codes: Primary (#3880ff), Primary-shade (#3171e0), and Primary-tint (#4c8dff). A red arrow points to the Primary swatch, which is highlighted with a blue square. The background of the slide features a large blue rectangle on the left side with the text "Clique na cor primária" (Click on the primary color).

Color Name	Hex Code
Primary	#3880ff
Primary-shade	#3171e0
Primary-tint	#4c8dff

Clique na cor primária

# Iniciando o Projeto

Como comentado anteriormente, o Ionic trabalha

- O site de documentação possui um gerador de tema:  
<https://ionicframework.com/docs/theming/color-palettes>

Escolha a cor desejada



# Iniciando o Projeto

Como comentado anteriormente, o Ionic trabalha com um conceito de *themes*

- O site de documentação possui um gerador de tema, com base em uma cor desejada  
<https://ionicframework.com/docs/theming/color-generator>

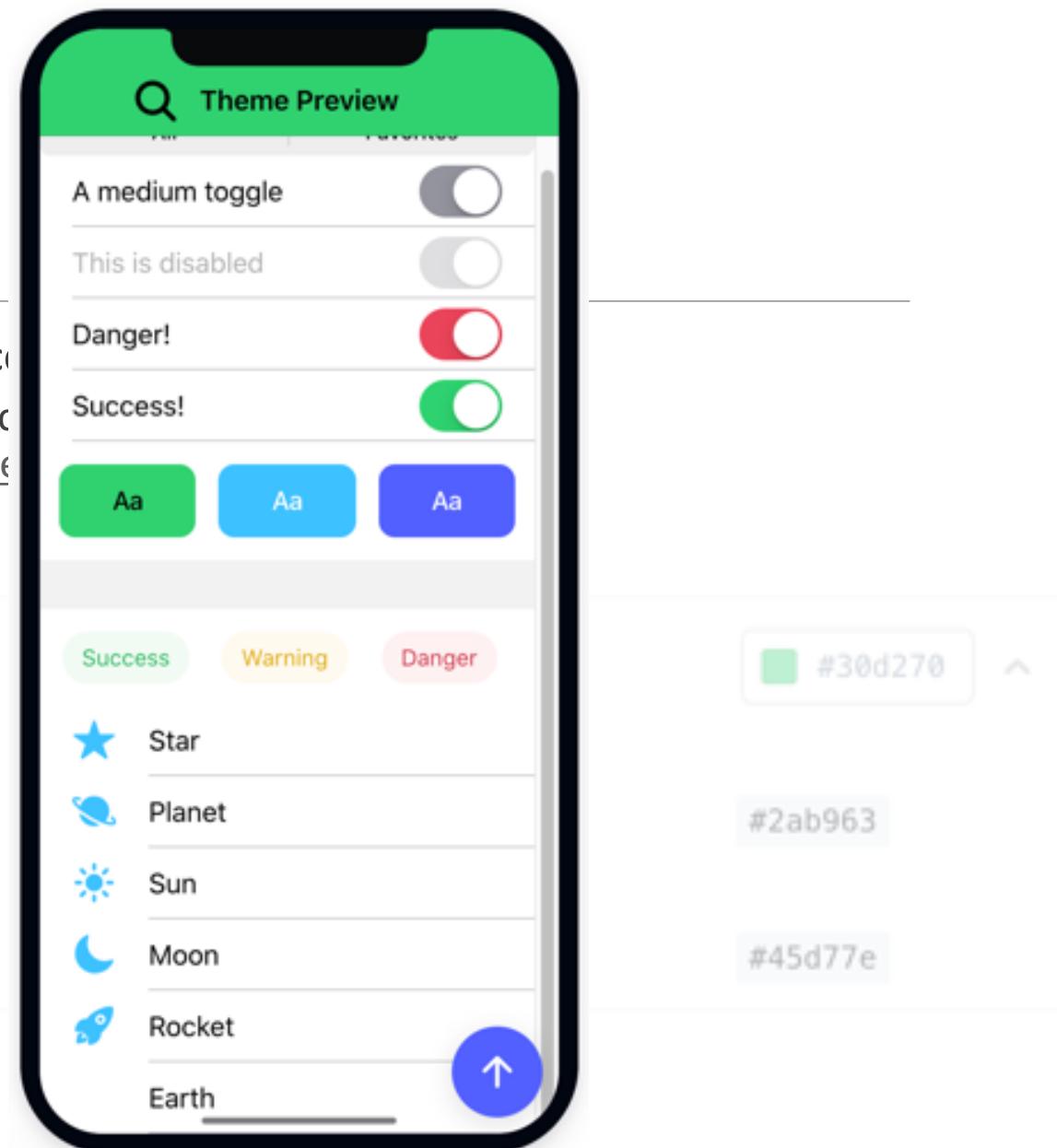
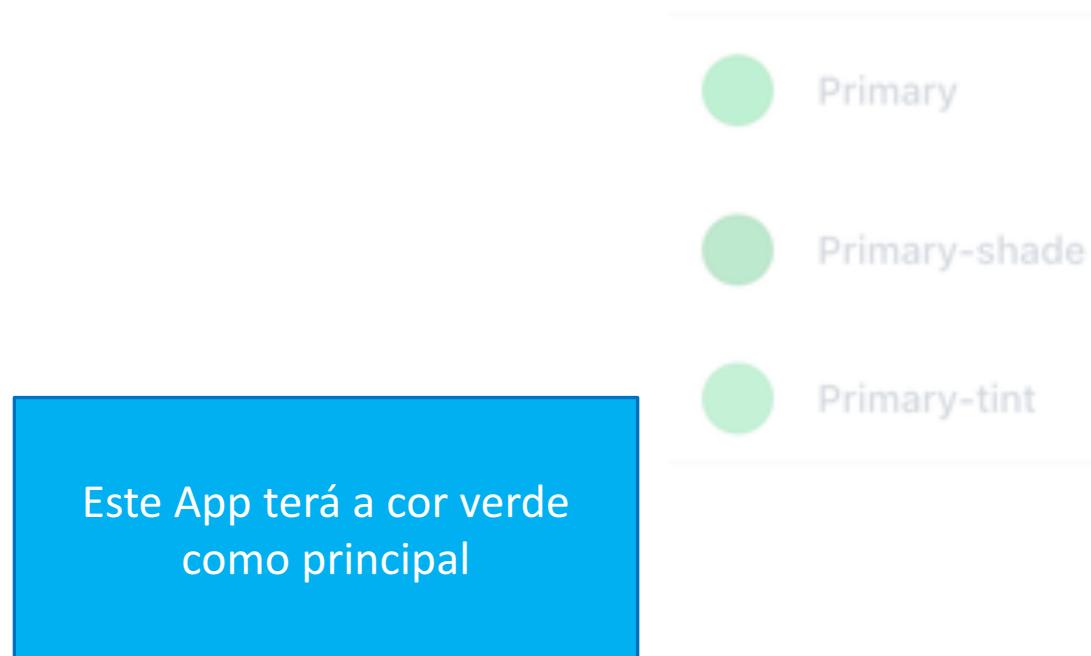


Este App terá a cor verde  
como principal

# Iniciando o Projeto

Como comentado anteriormente, o Ionic trabalha com:

- O site de documentação possui um gerador de tema, com link:  
<https://ionicframework.com/docs/theming/color-generator>



# Iniciando o F

Como comentado anteriormente

- O site de documentação possui:  
<https://ionicframework.com/>

Por fim, defino todos os níveis de cores desejados para o aplicativo

	Primary	 #30d270	▼
	Secondary	 #00b5ff	▼
	Tertiary	 #ffd700	▼
	Success	 #2dd36f	▼
	Warning	 #ffc409	▼
	Danger	 #eb445a	▼
	Dark	 #222428	▼
	Medium	 #92949c	▼
	Light	 #f4f5f8	▼

# Iniciando o Projeto

Como comentado anteriormente

- O site de documentação possui um link:  
<https://ionicframework.com/docs>

Logo abaixo, o site cria o CSS para nós

## CSS Variables

Copy

```
:root {  
  --ion-color-primary: #30d270;  
  --ion-color-primary-rgb: 48,210,112;  
  --ion-color-primary-contrast: #000000;  
  --ion-color-primary-contrast-rgb: 0,0,0;  
  --ion-color-primary-shade: #2ab963;  
  --ion-color-primary-tint: #45d77e;  
  
  --ion-color-secondary: #00b5ff;  
  --ion-color-secondary-rgb: 0,181,255;  
  --ion-color-secondary-contrast: #000000;  
  --ion-color-secondary-contrast-rgb: 0,0,0;  
  --ion-color-secondary-shade: #009fe0;  
  --ion-color-secondary-tint: #labcff;  
  
  --ion-color-tertiary: #ffd700;  
  --ion-color-tertiary-rgb: 255,215,0;  
  --ion-color-tertiary-contrast: #000000;  
  --ion-color-tertiary-contrast-rgb: 0,0,0;  
  --ion-color-tertiary-shade: #aa94aa;  
}
```

# Iniciando o Projeto

Como comentado anteriormente

- O site de documentação possui um link:  
<https://ionicframework.com/docs>

## CSS Variables

Copy

```
:root {  
  --ion-color-primary: #30d270;  
  --ion-color-primary-rgb: 48,210,112;  
  --ion-color-primary-contrast: #000000;  
  --ion-color-primary-contrast-rgb: 0,0,0;  
  --ion-color-primary-shade: #2ab963;  
  --ion-color-primary-tint: #45d77e;  
  
  --ion-color-secondary: #00b5ff;  
  --ion-color-secondary-rgb: 0,181,255;  
  --ion-color-secondary-contrast: #000000;  
  --ion-color-secondary-contrast-rgb: 0,0,0;  
  --ion-color-secondary-shade: #009fe0;  
  --ion-color-secondary-tint: #labcff;  
  
  --ion-color-tertiary: #ffd700;  
  --ion-color-tertiary-rgb: 255,215,0;  
  --ion-color-tertiary-contrast: #000000;  
  --ion-color-tertiary-contrast-rgb: 0,0,0;  
  --ion-color-tertiary-shade: #aa9400;}
```

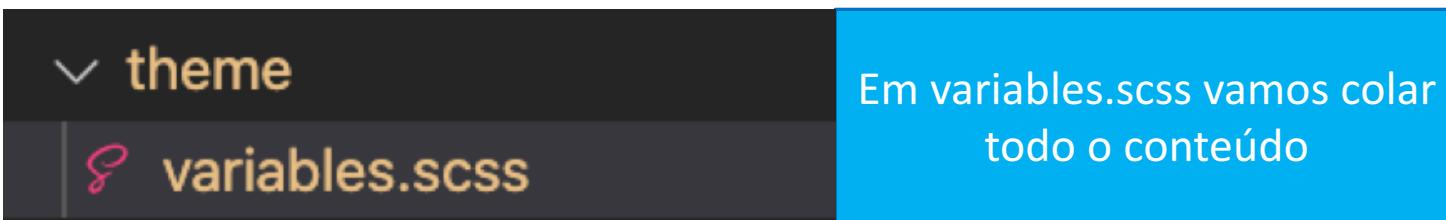


Vamos clicar em COPY

# Iniciando o Projeto

Como comentado anteriormente, o Ionic trabalha com um conceito de *themes*

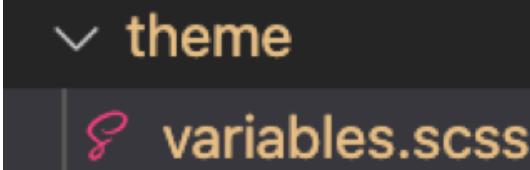
- O site de documentação possui um gerador de tema, com base em uma cor desejada  
<https://ionicframework.com/docs/theming/color-generator>



# Iniciando o Projeto

Como comentado anteriormente, o Ionic trabalha com um conceito de *themes*

- O site de documentação possui um gerador de tema, com base em uma cor desejada  
<https://ionicframework.com/docs/theming/color-generator>



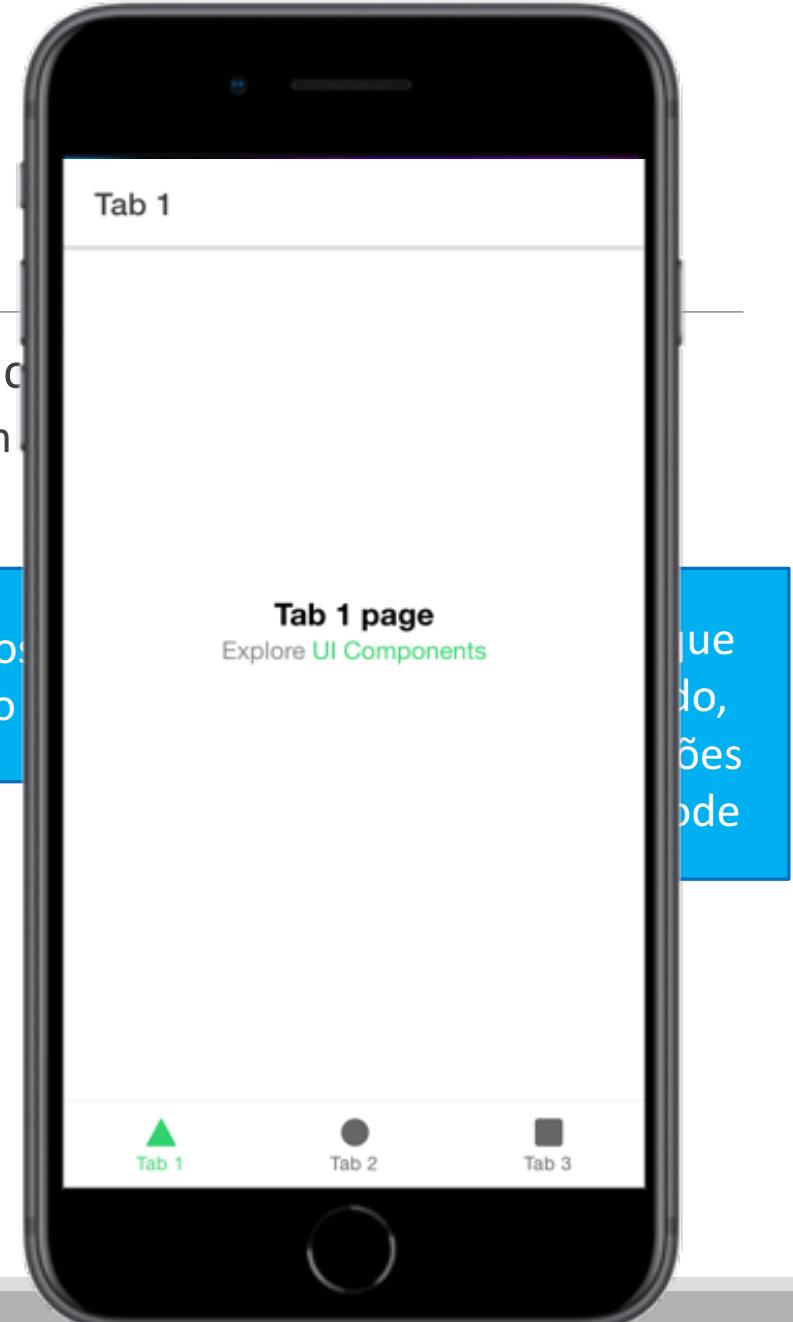
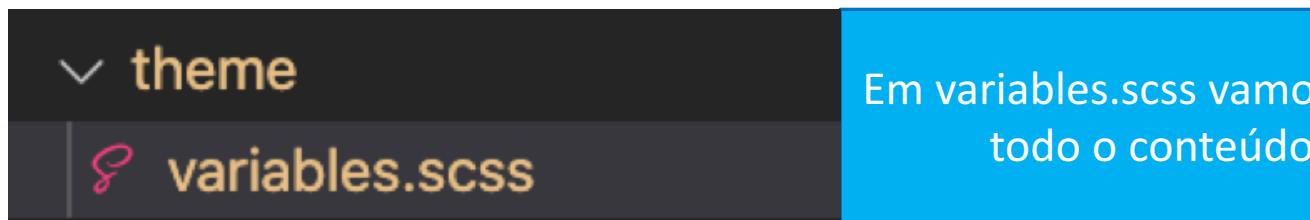
Em variables.scss vamos colar  
todo o conteúdo

É importante ressaltar que  
colando todo o conteúdo,  
perdemos as configurações  
de cores para o Dark Mode

# Iniciando o Projeto

Como comentado anteriormente, o Ionic trabalha com um conceito de tema.

- O site de documentação possui um gerador de tema, com base em <https://ionicframework.com/docs/theming/color-generator>

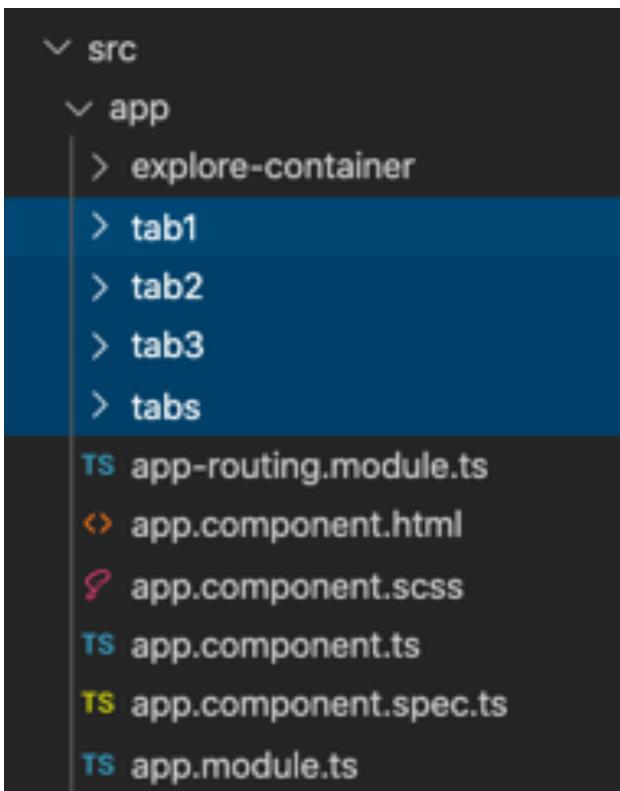


Entendendo as Tabs

# Entendendo as Tabs

Neste novo app criamos com conceito de tabs

- Com isso, o source (src) acaba sendo projetado com alguns itens de menu tab



# Entendendo as Tabs

---

Neste novo app criamos com conceito de tabs

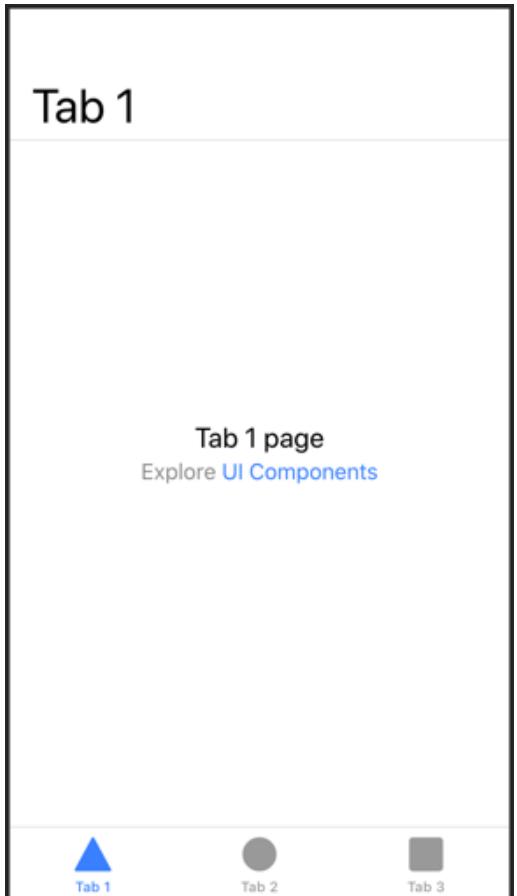
- Com isso, o source (src) acaba sendo projetado com alguns itens de menu tab
- Se executarmos o projeto podemos ver a criação destas tabs  
\$ ionic serve

# Entendendo as Tabs

Neste novo app criamos com conceito de tabs

- Com isso, o source (src) acaba sendo projetado com alguns itens de menu tab
- Se executarmos o projeto podemos ver a criação destas tabs  
\$ ionic serve

The screenshot shows a file explorer interface with a dark theme. The 'src' folder contains an 'app' folder, which in turn contains 'explore-container', 'tab1', 'tab2', 'tab3', and 'tabs'. Below the 'app' folder are several files: 'app-routing.module.ts' (highlighted in teal), 'app.component.html' (highlighted in orange), 'app.component.scss' (highlighted in pink), 'app.component.ts' (highlighted in teal), 'app.component.spec.ts' (highlighted in yellow), and 'app.module.ts' (highlighted in teal). The 'explore-container' folder is currently selected.

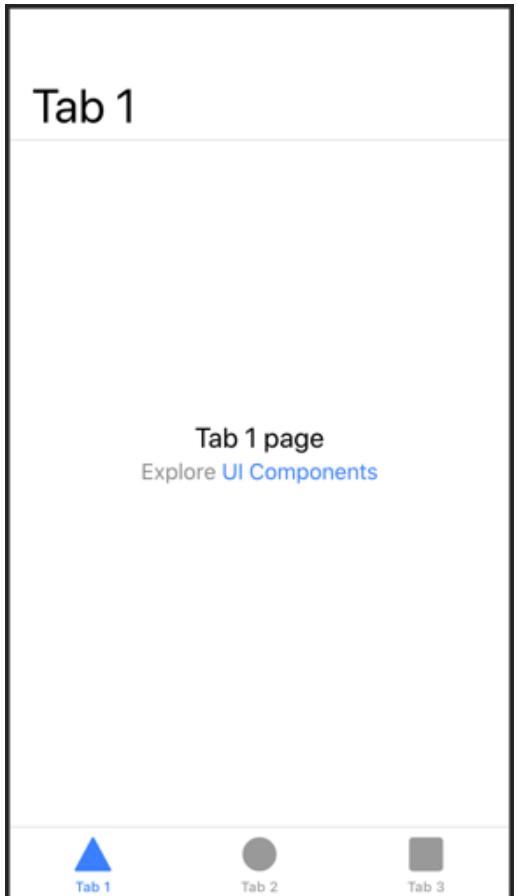
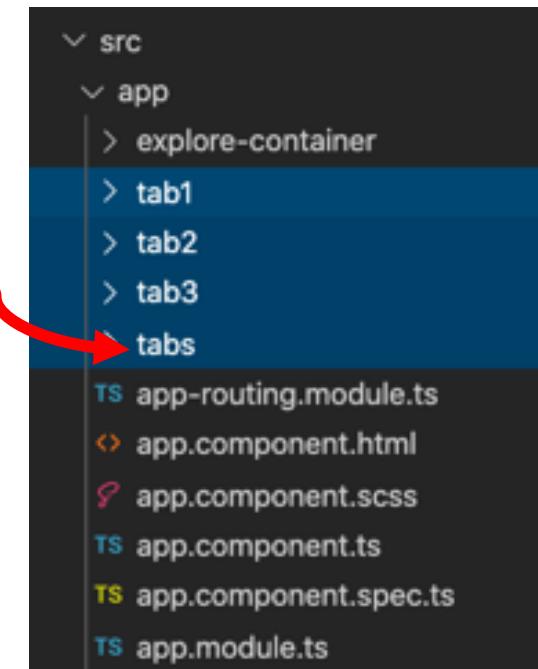


# Entendendo as Tabs

Neste novo app criamos com conceito de tabs

- Com isso, o source (src) acaba sendo projetado com alguns itens de menu tab
- Se executarmos o projeto podemos ver a criação destas tabs  
\$ ionic serve

A principal é a "tabs"  
É o módulo que gerencia  
as outras tabs



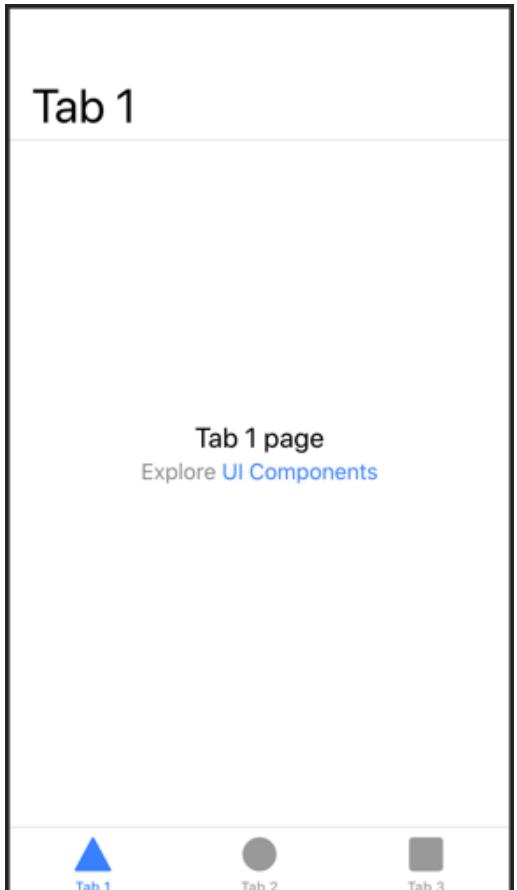
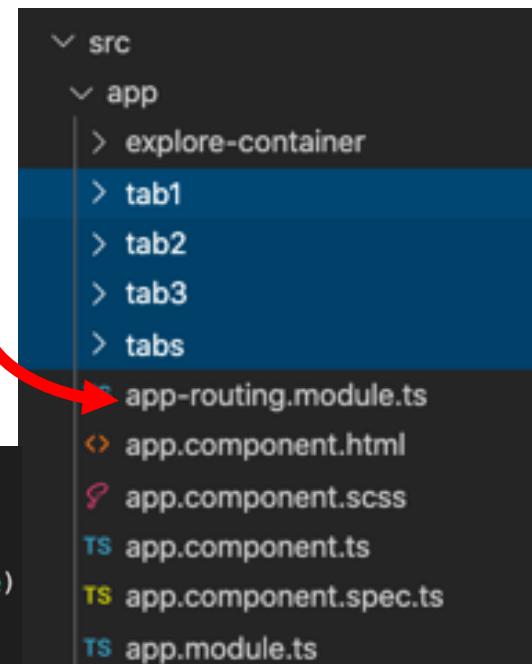
# Entendendo as Tabs

Neste novo app criamos com conceito de tabs

- Com isso, o source (src) acaba sendo projetado com alguns itens de menu tab
- Se executarmos o projeto podemos ver a criação destas tabs  
\$ ionic serve

Podemos ver isso analisando o  
app-routing.module.ts

```
const routes: Routes = [
  {
    path: '',
    loadChildren: () => import('./tabs/tabs.module').then(m => m.TabsPageModule)
  }
];
```

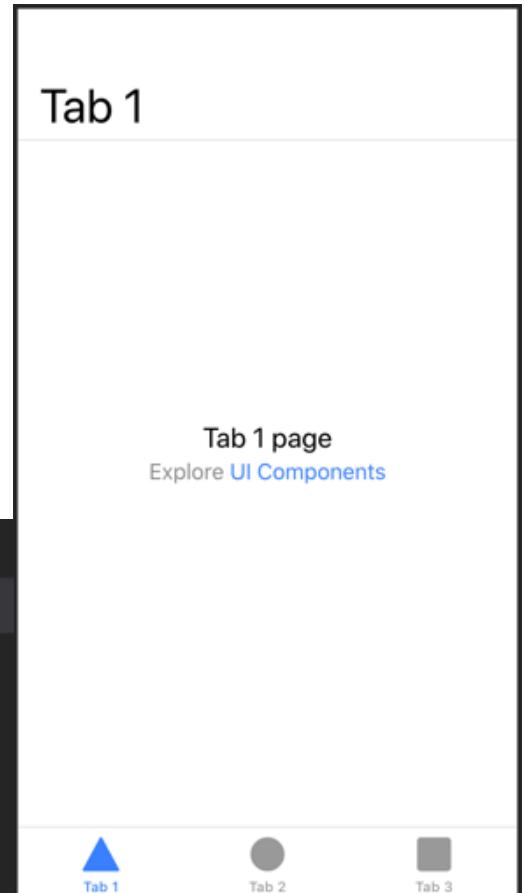
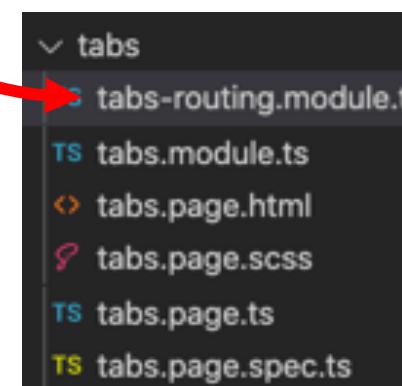


# Entendendo as Tabs

Neste novo app criamos com conceito de tabs

- Com isso, o source (src) acaba sendo projetado com alguns itens de menu tab
- Se executarmos o projeto podemos ver a criação destas tabs  
\$ ionic serve

Olhando o módulo de roteamento do tabs



# Entendendo as Tabs

Neste novo tema:

- Com isso
- Se executa

\$ ionic

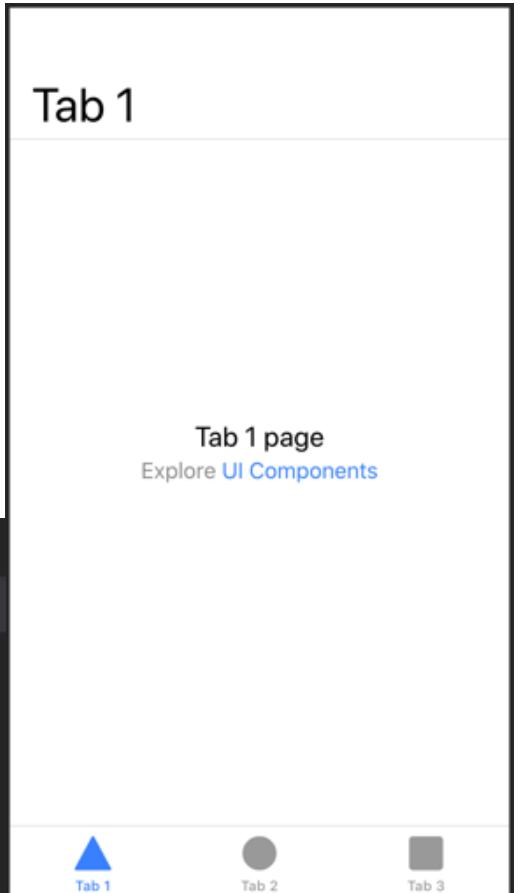
```
const routes: Routes = [
  {
    path: 'tabs',
    component: TabsPage,
    children: [
      {
        path: 'tab1',
        loadChildren: () => import('../tab1/tab1.module').then(m => m.Tab1PageModule)
      },
      {
        path: 'tab2',
        loadChildren: () => import('../tab2/tab2.module').then(m => m.Tab2PageModule)
      },
      {
        path: 'tab3',
        loadChildren: () => import('../tab3/tab3.module').then(m => m.Tab3PageModule)
      },
      {
        path: '',
        redirectTo: '/tabs/tab1',
        pathMatch: 'full'
      }
    ]
  },
  {
    path: '',
    redirectTo: '/tabs/tab1',
    pathMatch: 'full'
  }
];
```

Podemos ver as configurações de rotas com redirecionamento para tab1

mens de menu tab

tabs

- TS tabs-routing.module.ts
- TS tabs.module.ts
- tabs.page.html
- tabs.page.scss
- TS tabs.page.ts
- TS tabs.page.spec.ts

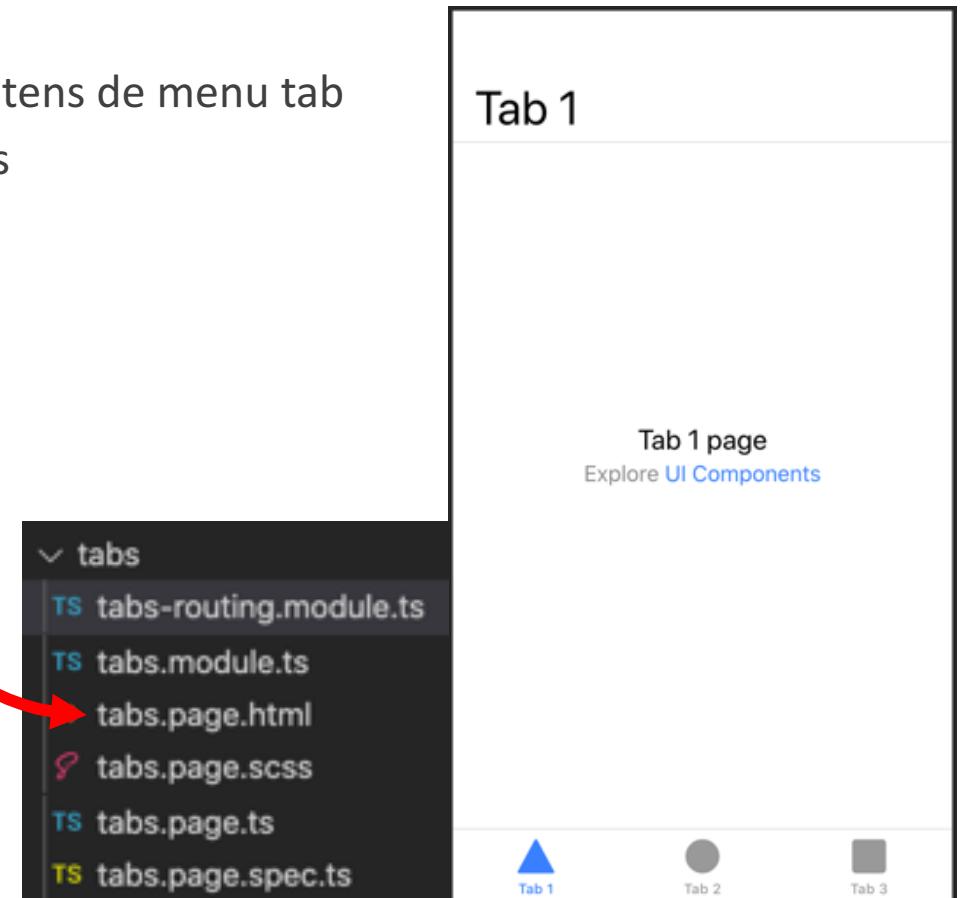


# Entendendo as Tabs

Neste novo app criamos com conceito de tabs

- Com isso, o source (src) acaba sendo projetado com alguns itens de menu tab
- Se executarmos o projeto podemos ver a criação destas tabs  
\$ ionic serve

No HTML podemos  
ver as ações

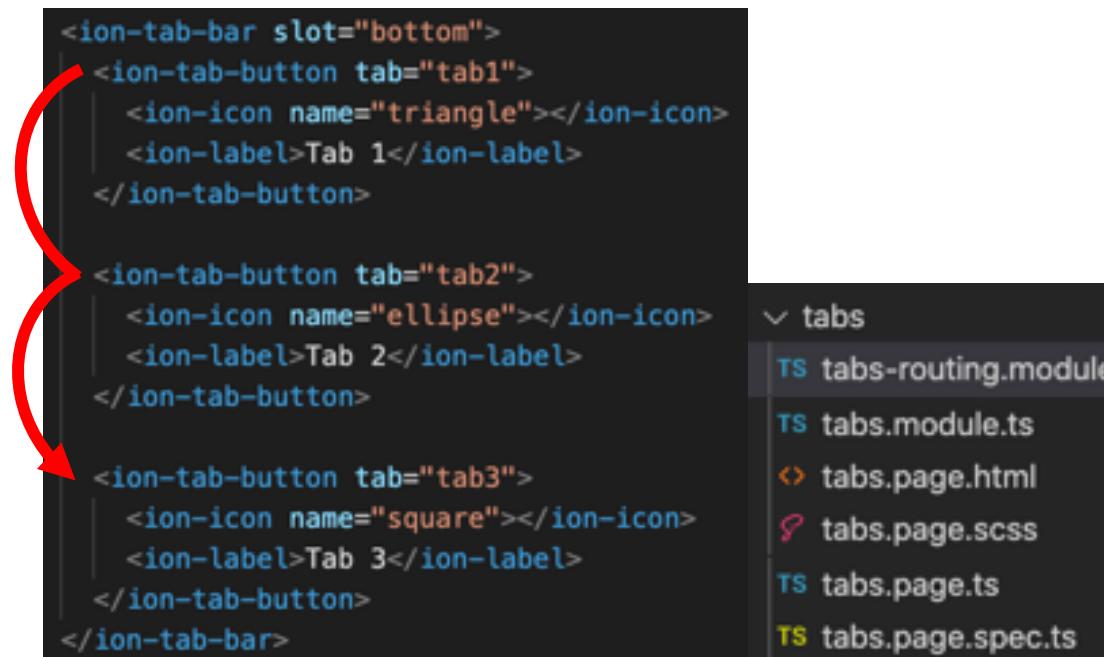


# Entendendo as Tabs

Neste novo app criamos com conceito de tabs

- Com isso, o source (src) acaba sendo projetado com alguns itens de menu tab
- Se executarmos o projeto podemos ver a criação destas tabs  
\$ ionic serve

Ações dos botões  
do tab menu

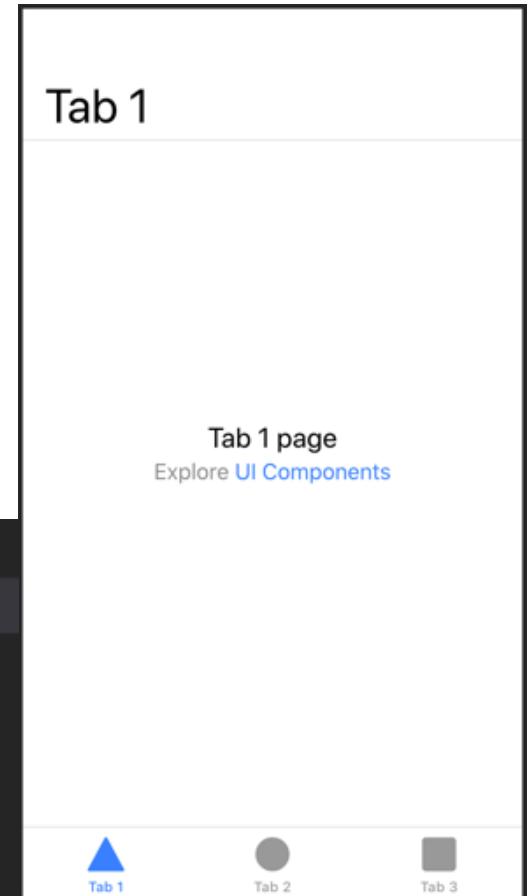


The screenshot shows the file structure of an Ionic project under the 'src' directory. A red arrow points from the text 'Ações dos botões do tab menu' to the code snippet. The code is as follows:

```
<ion-tab-bar slot="bottom">
  <ion-tab-button tab="tab1">
    <ion-icon name="triangle"></ion-icon>
    <ion-label>Tab 1</ion-label>
  </ion-tab-button>

  <ion-tab-button tab="tab2">
    <ion-icon name="ellipse"></ion-icon>
    <ion-label>Tab 2</ion-label>
  </ion-tab-button>

  <ion-tab-button tab="tab3">
    <ion-icon name="square"></ion-icon>
    <ion-label>Tab 3</ion-label>
  </ion-tab-button>
</ion-tab-bar>
```

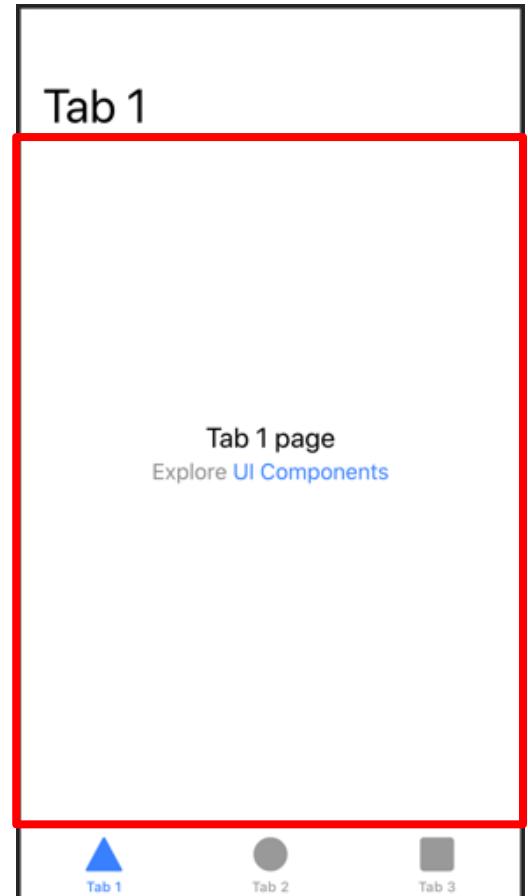
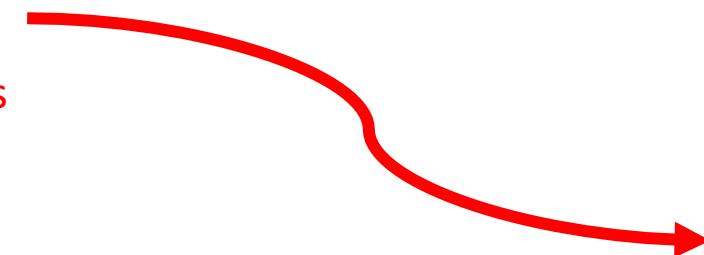


# Entendendo as Tabs

Neste novo app criamos com conceito de tabs

- Com isso, o source (src) acaba sendo projetado com alguns itens de menu tab
- Se executarmos o projeto podemos ver a criação destas tabs  
\$ ionic serve

Analisando a Tab1,  
podemos notar que ela  
não possui roteamento  
para outras páginas



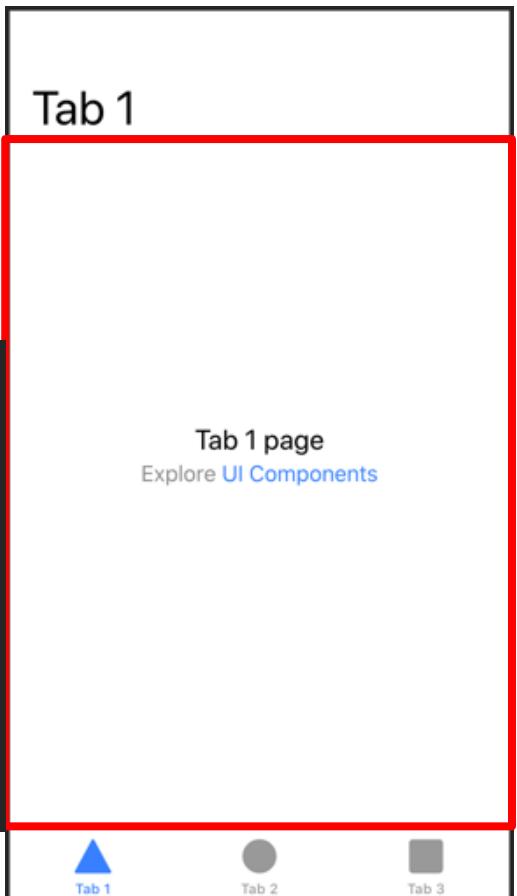
# Entendendo as Tabs

Neste novo app criamos com conceito de tabs

- Com isso, o source (src) acaba sendo projetado com alguns itens de menu tab
- Se executarmos o projeto podemos ver a criação destas tabs  
\$ ionic serve

Vamos analisar  
este componente

```
tab1
  TS tab1-routing.module.ts
  TS tab1.module.ts
  ⚡ tab1.page.html
  ❌ tab1.page.scss
  TS tab1.page.ts
  TS tab1.page.spec.ts
```

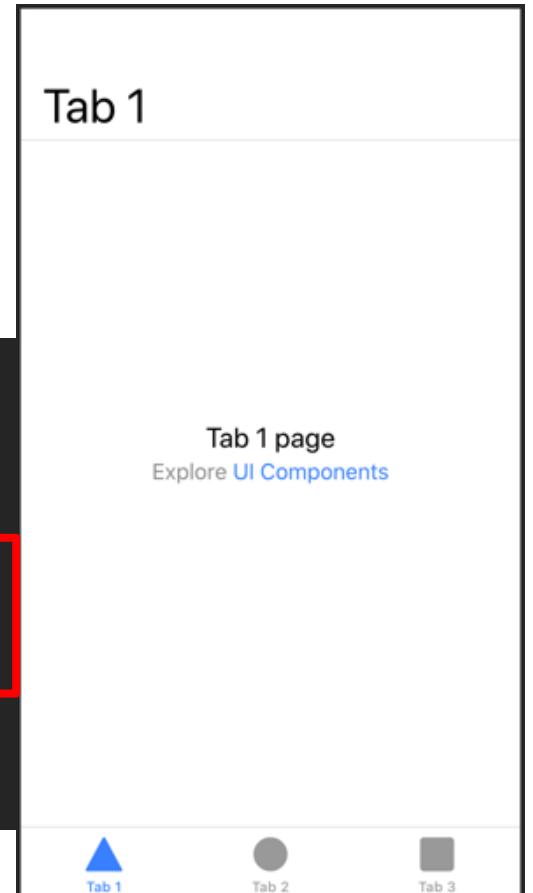
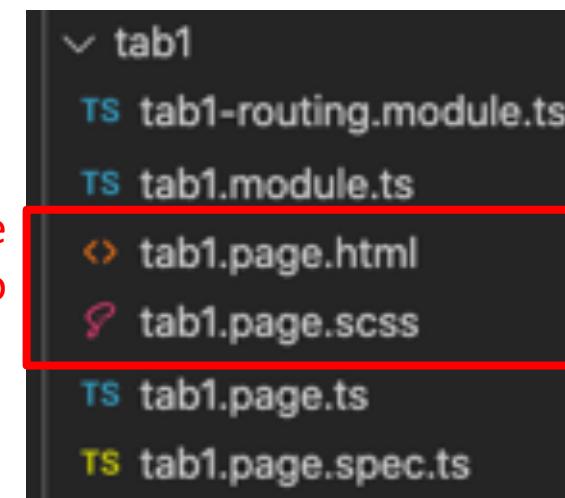


# Entendendo as Tabs

Neste novo app criamos com conceito de tabs

- Com isso, o source (src) acaba sendo projetado com alguns itens de menu tab
- Se executarmos o projeto podemos ver a criação destas tabs  
\$ ionic serve

Arquivos de controle  
destinados a visão

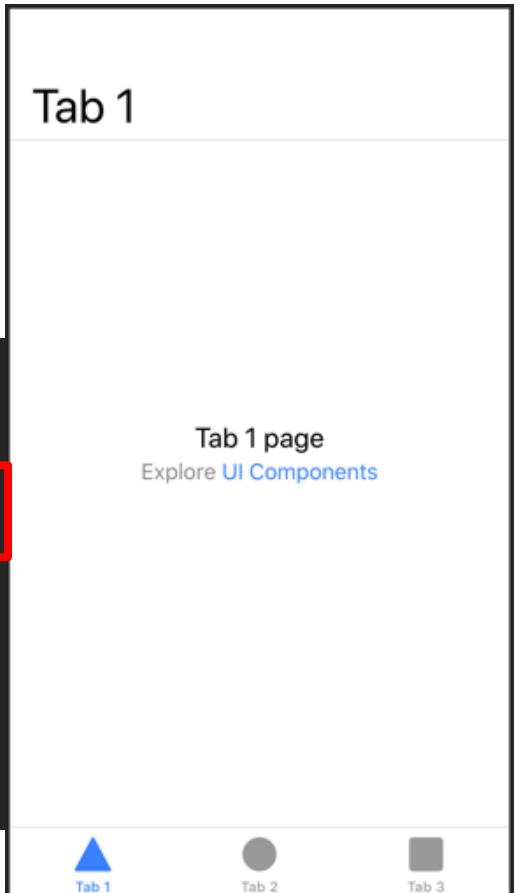
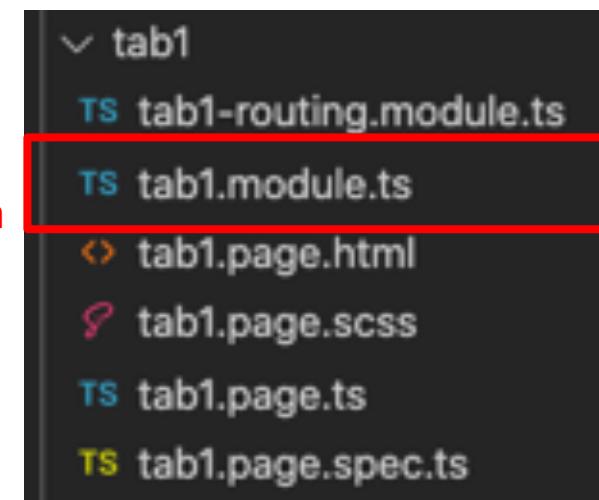


# Entendendo as Tabs

Neste novo app criamos com conceito de tabs

- Com isso, o source (src) acaba sendo projetado com alguns itens de menu tab
- Se executarmos o projeto podemos ver a criação destas tabs  
\$ ionic serve

Arquivo para gerenciamento dos módulos utilizados pela página

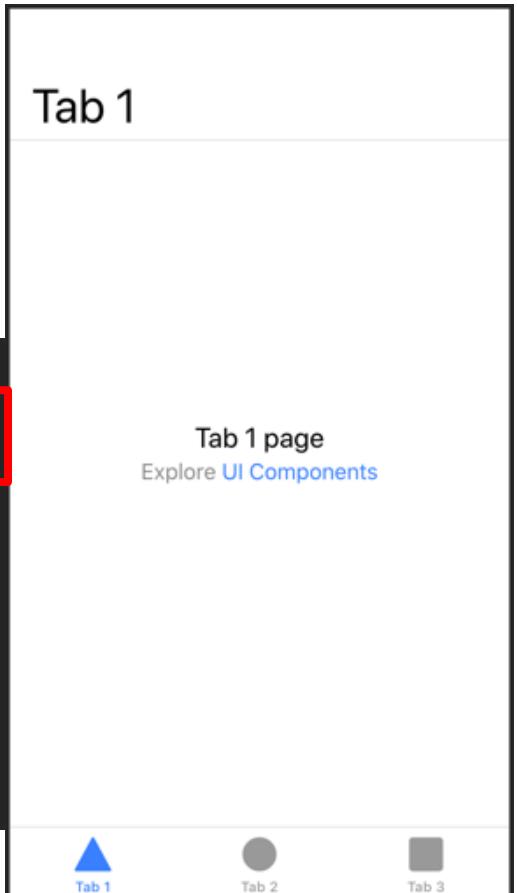
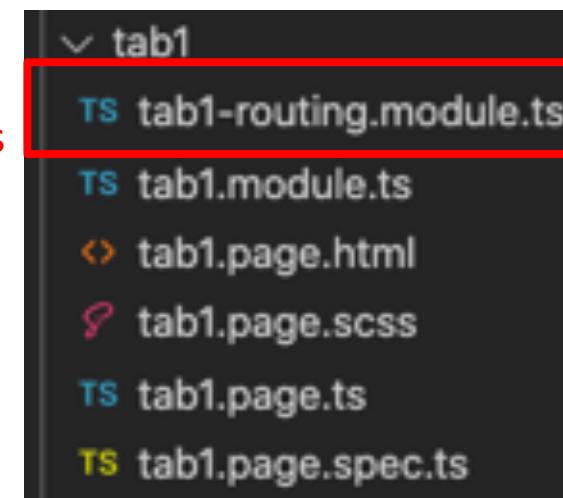


# Entendendo as Tabs

Neste novo app criamos com conceito de tabs

- Com isso, o source (src) acaba sendo projetado com alguns itens de menu tab
- Se executarmos o projeto podemos ver a criação destas tabs  
\$ ionic serve

Arquivo para  
controle de rotas

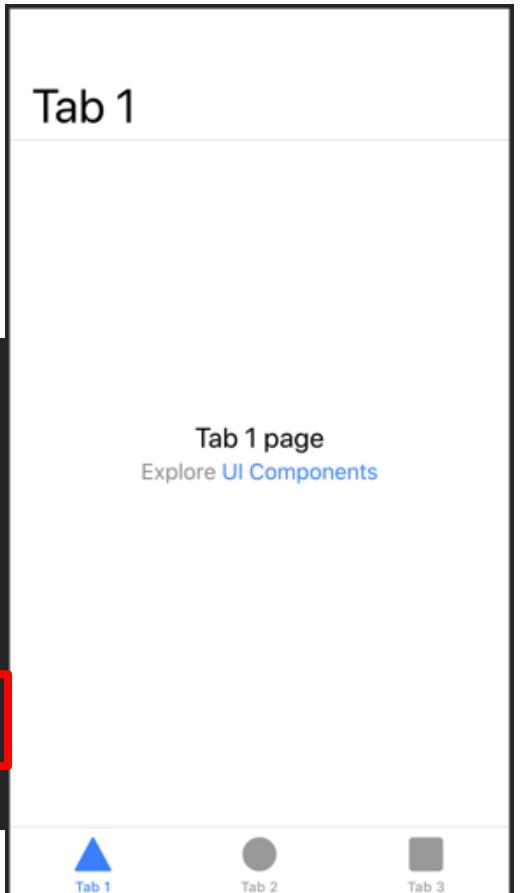
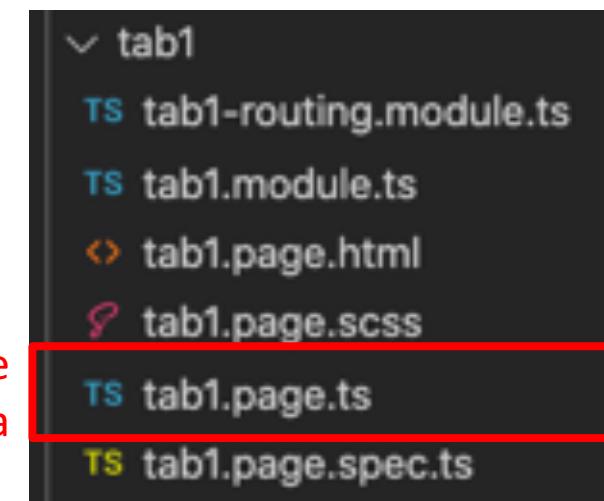


# Entendendo as Tabs

Neste novo app criamos com conceito de tabs

- Com isso, o source (src) acaba sendo projetado com alguns itens de menu tab
- Se executarmos o projeto podemos ver a criação destas tabs  
\$ ionic serve

Arquivo para organização e comportamento da página



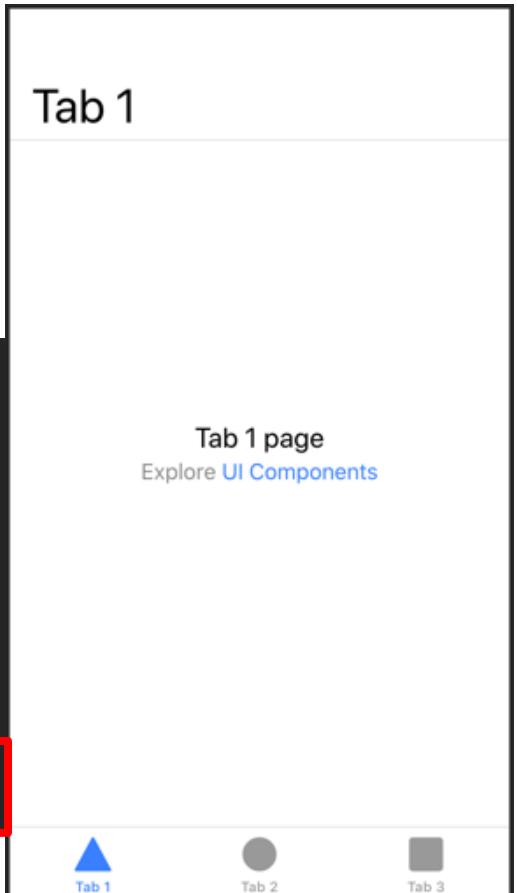
# Entendendo as Tabs

Neste novo app criamos com conceito de tabs

- Com isso, o source (src) acaba sendo projetado com alguns itens de menu tab
- Se executarmos o projeto podemos ver a criação destas tabs  
\$ ionic serve

Arquivo de teste

```
tab1
  TS tab1-routing.module.ts
  TS tab1.module.ts
  ⚡ tab1.page.html
  ❌ tab1.page.scss
  TS tab1.page.ts
  TS tab1.page.spec.ts
```

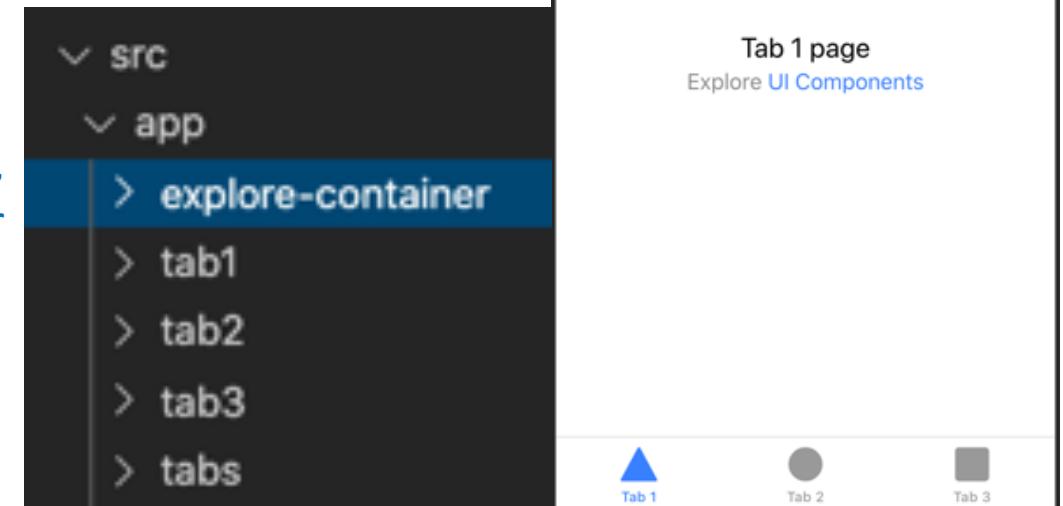


# Entendendo as Tabs

Neste novo app criamos com conceito de tabs

- Com isso, o source (src) acaba sendo projetado com alguns itens de menu tab
- Se executarmos o projeto podemos ver a criação destas tabs  
\$ ionic serve

Podemos notar, também, que junto com as tabs, temos um elemento chamado explore-container



# Entendendo as Tabs

```
TS tab1.module.ts ×  
src > app > tab1 > TS tab1.module.ts > ...  
1 import { IonicModule } from '@ionic/angular';  
2 import { NgModule } from '@angular/core';  
3 import { CommonModule } from '@angular/common';  
4 import { FormsModule } from '@angular/forms';  
5 import { Tab1Page } from './tab1.page';  
6 import { ExploreContainerComponentModule } from '../explore-container/explore-container.module';  
7  
8 import { Tab1PageRoutingModule } from './tab1-routing.module';  
9  
10 @NgModule({  
11   imports: [  
12     IonicModule,  
13     CommonModule,  
14     FormsModule,  
15     ExploreContainerComponentModule,  
16     Tab1PageRoutingModule  
17   ],  
18   declarations: [Tab1Page]  
19 })  
20 export class Tab1PageModule {}
```

Tab 1

Tab 1 page  
Explore UI Components

Todas as tabs utilizam este componente



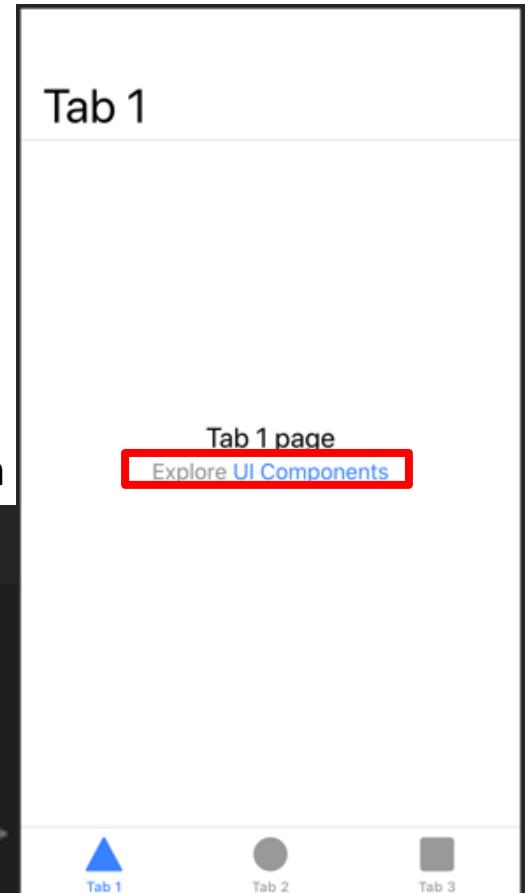
# Entendendo as Tabs

Neste novo app criamos com conceito de tabs

- Com isso, o source (src) acaba sendo projetado com alguns itens de menu tab
- Se executarmos o projeto podemos ver a criação destas tabs  
\$ ionic serve

Responsável por renderizar esta mensagem

```
⌄ explore-container.component.html ⌄
src > app > explore-container > ⌄ explore-container.component.html > ...
1   <div id="container">
2     <strong>{{ name }}</strong>
3     <p>Explore <a target="_blank"
4           rel="noopener noreferrer"
5           href="https://ionicframework.com/docs/components">UI Components</a></p>
6   </div>
```



# Entendendo as Tabs

Neste novo app criamos com conceito de tabs

- Com isso, o source (src) acaba sendo projetado com alguns itens de menu tab
- Se executarmos o projeto podemos ver a criação destas tabs  
\\$ ionic serve

Tab 1

No HTML da Tab usa-se assim

```
16 |     <app-explore-container name="Tab 1 page"></app-explore-container>
17 |     </ion-content>
```

```
↳ explore-container.component.html ×
src > app > explore-container > ↳ explore-container.component.html > ...
1   <div id="container">
2     <strong>{{ name }}</strong>
3     <p>Explore <a target="_blank"
4           rel="noopener noreferrer"
5           href="https://ionicframework.com/docs/components">UI Components</a></p>
6   </div>
```

Tab 1 page
Explore UI Components



# Organizando Tabs

# Organizando Tabs

Vamos organizar as tabs padrões do sistema

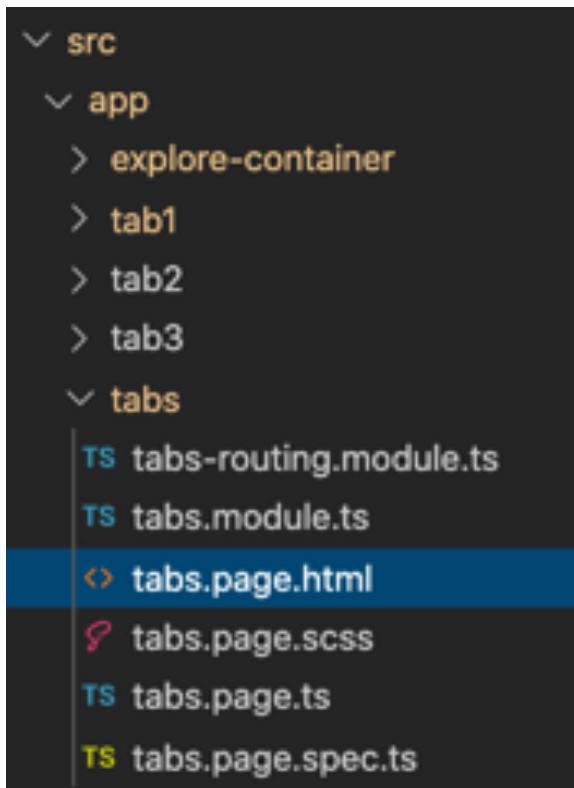
- Para isso, vamos alterar o HTML de tabs

```
└─ src
    └─ app
        ├─ explore-container
        ├─ tab1
        ├─ tab2
        ├─ tab3
        └─ tabs
            ├─ TS tabs-routing.module.ts
            ├─ TS tabs.module.ts
            ├─ <> tabs.page.html
            ├─ TS tabs.page.scss
            ├─ TS tabs.page.ts
            └─ TS tabs.page.spec.ts
```

# Organizando Tabs

Vamos organizar as tabs padrões do sistema

- Para isso, vamos alterar o HTML de tabs

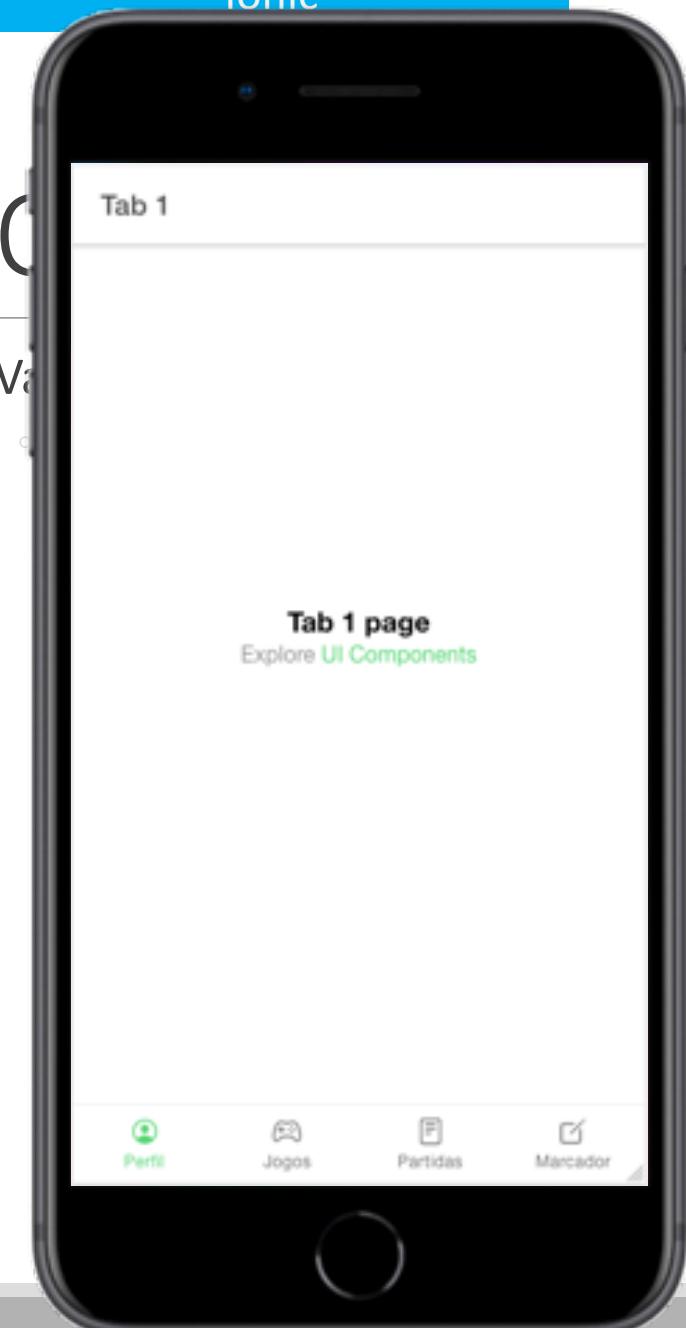


```
<ion-tabs>
  <ion-tab-bar slot="bottom">
    <ion-tab-button tab="tab1">
      <ion-icon name="person-circle-outline"></ion-icon>
      <ion-label>Perfil</ion-label>
    </ion-tab-button>

    <ion-tab-button tab="tab2">
      <ion-icon name="game-controller-outline"></ion-icon>
      <ion-label>Jogos</ion-label>
    </ion-tab-button>

    <ion-tab-button tab="tab3">
      <ion-icon name="reader-outline"></ion-icon>
      <ion-label>Partidas</ion-label>
    </ion-tab-button>

    <ion-tab-button tab="tab4">
      <ion-icon name="create-outline"></ion-icon>
      <ion-label>Marcador</ion-label>
    </ion-tab-button>
  </ion-tab-bar>
</ion-tabs>
```



# Tabs

Vamos ao sistema  
de tabs

## Tab 1 page

Explore UI Components

Perfil    Jogos    Partidas    Marcador

```
<ion-tabs>
  <ion-tab-bar slot="bottom">
    <ion-tab-button tab="tab1">
      <ion-icon name="person-circle-outline"></ion-icon>
      <ion-label>Perfil</ion-label>
    </ion-tab-button>

    <ion-tab-button tab="tab2">
      <ion-icon name="game-controller-outline"></ion-icon>
      <ion-label>Jogos</ion-label>
    </ion-tab-button>

    <ion-tab-button tab="tab3">
      <ion-icon name="reader-outline"></ion-icon>
      <ion-label>Partidas</ion-label>
    </ion-tab-button>

    <ion-tab-button tab="tab4">
      <ion-icon name="create-outline"></ion-icon>
      <ion-label>Marcador</ion-label>
    </ion-tab-button>
  </ion-tab-bar>
</ion-tabs>
```



# Tabs

Vantagens  
do sistema  
de tabs

Tab 1 page  
Explore UI Components

```
<ion-tabs>
  <ion-tab-bar slot="bottom">
    <ion-tab-button tab="tab1">
      <ion-icon name="person-circle-outline"></ion-icon>
      <ion-label>Perfil</ion-label>
    </ion-tab-button>
    <ion-tab-button tab="tab2">
      <ion-icon name="game-controller-outline"></ion-icon>
      <ion-label>Jogos</ion-label>
    </ion-tab-button>
    <ion-tab-button tab="tab3">
      <ion-icon name="reader-outline"></ion-icon>
      <ion-label>Partidas</ion-label>
    </ion-tab-button>
    <ion-tab-button tab="tab4">
      <ion-icon name="create-outline"></ion-icon>
      <ion-label>Marcador</ion-label>
    </ion-tab-button>
  </ion-tab-bar>
</ion-tabs>
```



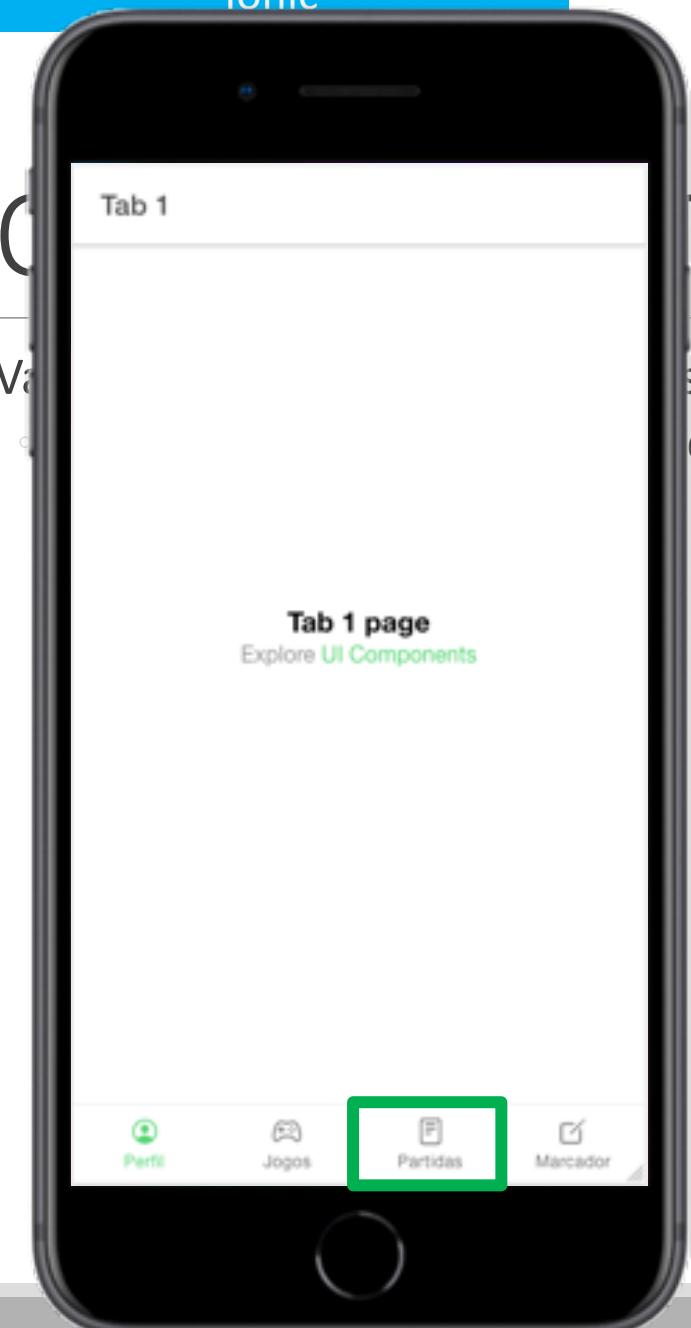
# Tabs

s do sistema  
de tabs

Tab 1 page  
Explore UI Components

Perfil Jogos Partidas Marcador

```
<ion-tabs>
  <ion-tab-bar slot="bottom">
    <ion-tab-button tab="tab1">
      <ion-icon name="person-circle-outline"></ion-icon>
      <ion-label>Perfil</ion-label>
    </ion-tab-button>
    <ion-tab-button tab="tab2">
      <ion-icon name="game-controller-outline"></ion-icon>
      <ion-label>Jogos</ion-label>
    </ion-tab-button>
    <ion-tab-button tab="tab3">
      <ion-icon name="reader-outline"></ion-icon>
      <ion-label>Partidas</ion-label>
    </ion-tab-button>
    <ion-tab-button tab="tab4">
      <ion-icon name="create-outline"></ion-icon>
      <ion-label>Marcador</ion-label>
    </ion-tab-button>
  </ion-tab-bar>
</ion-tabs>
```



# Tabs

s do sistema  
de tabs

## Tab 1 page

Explore UI Components



Perfil



Jogos



Partidas



Marcador

## tabs.page.html

```
src > app > tabs > tabs.page.html > ...  
1 <ion-tabs>  
2   <ion-tab-bar slot="bottom">  
3     <ion-tab-button tab="tab1">  
4       <ion-icon name="person-circle-outline"></ion-icon>  
5       <ion-label>Perfil</ion-label>  
6     </ion-tab-button>  
7     <ion-tab-button tab="tab2">  
8       <ion-icon name="game-controller-outline"></ion-icon>  
9       <ion-label>Jogos</ion-label>  
10     </ion-tab-button>  
11     <ion-tab-button tab="tab3">  
12       <ion-icon name="reader-outline"></ion-icon>  
13       <ion-label>Partidas</ion-label>  
14     </ion-tab-button>  
15     <ion-tab-button tab="tab4">  
16       <ion-icon name="create-outline"></ion-icon>  
17       <ion-label>Marcador</ion-label>  
18     </ion-tab-button>  
19   </ion-tab-bar>  
20 </ion-tabs>
```



# Tabs

s do sistema  
de tabs

Tab 1 page  
Explore UI Components



Perfil



Jogos



Partidas



Marcador

```
<ion-tabs>
  <ion-tab-bar slot="bottom">
    <ion-tab-button tab="tab1">
      <ion-icon name="person-circle-outline"></ion-icon>
      <ion-label>Perfil</ion-label>
    </ion-tab-button>

    <ion-tab-button tab="tab2">
      <ion-icon name="game-controller-outline"></ion-icon>
      <ion-label>Jogos</ion-label>
    </ion-tab-button>

    <ion-tab-button tab="tab3">
      <ion-icon name="reader-outline"></ion-icon>
      <ion-label>Partidas</ion-label>
    </ion-tab-button>

    <ion-tab-button tab="tab4">
      <ion-icon name="create-outline"></ion-icon>
      <ion-label>Marcador</ion-label>
    </ion-tab-button>
  </ion-tab-bar>
</ion-tabs>
```

# Organizando Tabs

---

Vamos organizar as tabs padrões do sistema

- As tabs que vieram criadas na aplicação não serão utilizadas, são apenas para exemplo
- Então, vamos criar as 4 tabs necessárias para nossa aplicação

# Organizando Tabs

---

Vamos organizar as tabs padrões do sistema

- As tabs que vieram criadas na aplicação não serão utilizadas, são apenas para exemplo
- Então, vamos criar as 4 tabs necessárias para nossa aplicação
- Para criar a maioria de nossos elementos vamos fazer uso do Ionic-CLI
- Sempre que criar/excluir elementos, é indicado parar o servidor

Para criarmos nova página vamos executar, via terminal, dentro da pasta do projeto

```
$ ionic g
```

# Organizando Tabs

Vamos organizar as tabs padrões do sistema

- As tabs que vieram criadas na aplicação não serão utilizadas, são apenas para exemplo
- Então, vamos criar as 4 tabs necessárias para nossa aplicação
- Para criar a maioria de nossos elementos vamos fazer uso do Ionic-CLI
- Sempre que criar/excluir elementos, é indicado parar o servidor

Para criarmos nova página vamos executar, via terminal, dentro da pasta do projeto

```
$ ionic g
```

```
[→ PlayingScore git:(master) ✘ ionic g
? What would you like to generate? (Use arrow keys)
❯ page
component
service
module
class
directive
guard
(Move up and down to reveal more choices)
```

O Ionic-CLI indica opções,  
e vamos escolher "page"

# Organizando Tabs

Vamos organizar as tabs padrões do sistema

- As tabs que vieram criadas na aplicação não serão utilizadas, são apenas para exemplo
- Então, vamos criar as 4 tabs necessárias para nossa aplicação
- Para criar a maioria de nossos elementos vamos fazer uso do Ionic-CLI
- Sempre que criar/excluir elementos, é indicado parar o servidor

Para criarmos nova página vamos executar, via terminal, dentro da pasta do projeto

```
$ ionic g
```

```
[→ PlayingScore git:(master) ✘ ionic g
? What would you like to generate? page
? Name/path of page: perfil
```

Na sequência, damos um nome para nossa nova página

```
[→ PlayingScore git:(master) ✘ ionic g
? What would you like to generate? page
[?] Name/path of page: perfil
> ng generate page perfil --project=app
? Would you like to share anonymous usage data about this project with the Angular Team at
Google under Google's Privacy Policy at https://policies.google.com/privacy? For more
details and how to change this setting, see http://angular.io/analytics. Yes
```

Thank you for sharing anonymous usage data. Would you change your mind, the following command will disable this feature entirely:

```
ng analytics project off
```

```
CREATE src/app/perfil/perfil-routing.module.ts (347 bytes)
CREATE src/app/perfil/perfil.module.ts (472 bytes)
CREATE src/app/perfil/perfil.page.scss (0 bytes)
CREATE src/app/perfil/perfil.page.html (125 bytes)
CREATE src/app/perfil/perfil.page.spec.ts (647 bytes)
CREATE src/app/perfil/perfil.page.ts (256 bytes)
UPDATE src/app/app-routing.module.ts (534 bytes)
[OK] Generated page!
```

```
[→ PlayingScore git:(master) ✘ ionic g
? What would you like to generate? page
[?] Name/path of page: perfil
> ng generate page perfil --p
? Would you like to share and
Google under Google's Privacy
[details and how to change thi
```

Thank you for sharing anonymously! This command will disable this feature for this project.

ng analytics project off

```
CREATE src/app/perfil/perfil-
CREATE src/app/perfil/perfil.
CREATE src/app/perfil/perfil.
CREATE src/app/perfil/perfil.
CREATE src/app/perfil/perfil.
CREATE src/app/perfil/perfil.
CREATE src/app/perfil/perfil.
UPDATE src/app/app-routing.module.ts (534 bytes)
[OK] Generated page!
```

- ✓ app
- > explore-container
- ✓ perfil
  - TS perfil-routing.module.ts
  - TS perfil.module.ts
  - ↔ perfil.page.html
  - ⌚ perfil.page.scss
  - TS perfil.page.ts
  - TS perfil.page.spec.ts

ect with the Angular Team at [https://angular.com/privacy](#)? For more information about our privacy and analytics. Yes

our mind, the following

# Organizando Tabs

---

Vamos organizar as tabs padrões do sistema

- Automaticamente, dentro das possíveis rotas da aplicação, o Ionic-CLI cria a rota da nova page

# Organizando Tabs

Vamos organizar as tabs padrões do sistema

- Automaticamente, dentro das possíveis rotas da aplicação, o Ionic-CLI cria a rota da nova page

```
✓ app
  > explore-container
  > perfil
  > tab1
  > tab2
  > tab3
  > tabs
TS app-routing.module.ts
    4   const routes: Routes = [
    5     {
    6       path: '',
    7       loadChildren: () => import('./tabs/tabs.module').then(m => m.TabsPageModule)
    8     },
    9     {
   10       path: 'perfil',
   11       loadChildren: () => import('./perfil/perfil.module').then( m => m.PerfilPageModule)
   12     }
   13   ];

```

# Organizando Tabs

---

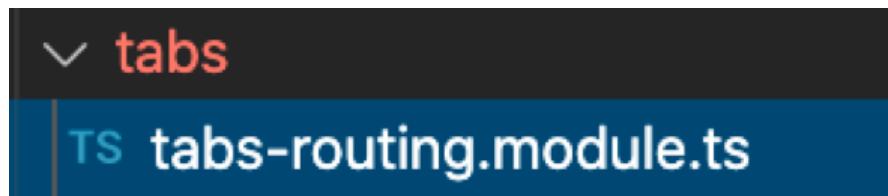
Vamos organizar as tabs padrões do sistema

- Mas, o que nos interessa mesmo, é termos esta rota no controle de tabs

# Organizando Tabs

Vamos organizar as tabs padrões do sistema

- Mas, o que nos interessa mesmo, é termos esta rota no controle de tabs



```
ts tabs-routing.module.ts ×

src > app > tabs > ts tabs-routing.module.ts > [e] routes > ↗ children
  2   import { RouterModule, Routes } from '@angular/router';
  3   import { TabsPage } from './tabs.page';
  4
  5   const routes: Routes = [
  6     {
  7       path: 'tabs',
  8       component: TabsPage,
  9       children: [
 10         {
 11           path: 'tabPerfil',
 12           loadChildren: () => import('../perfil/perfil.module').then(m => m.PerfilPageModule)
 13         },
 14         {
 15           path: 'tab1',
 16           loadChildren: () => import('../tab1/tab1.module').then(m => m.Tab1PageModule)
 17         }
 18       ]
 19     }
 20   ];
 21
 22   export { routes };
```

# Organizando Tabs

---

Vamos organizar as tabs padrões do sistema

- No HTML dos tabs, vamos adicionar como ação o tabPerfil

# Organizando Tabs

Vamos organizar as tabs padrões do sistema

- No HTML dos tabs, vamos adicionar como ação o tabPerfil

```
<tabs.page.html >

src > app > tabs > <tabs.page.html> <ion-tabs> <ion-tab-bar slot="bottom">
1   <ion-tabs>
2     <ion-tab-bar slot="bottom">
3
4       <ion-tab-button tab="tabPerfil">
5         <ion-icon name="person-circle-outline"></ion-icon>
6         <ion-label>Perfil</ion-label>
7       </ion-tab-button>
```

# Organizando Tabs

---

Vamos organizar as tabs padrões do sistema

- Vamos criar as outras 3 pages e já fazer os ajustes de perfil

# Organizando Tabs

Vamos organizar as tabs padrões do sistema

- Vamos criar as outras 3 pages e já fazer os ajustes de perfil

```
[→ PlayingScore git:(master) ✘ ionic g
? What would you like to generate? page
[?] Name/path of page: jogos
> ng generate page jogos --project=app
CREATE src/app/jogos/jogos-routing.module.ts (343 bytes)
CREATE src/app/jogos/jogos.module.ts (465 bytes)
CREATE src/app/jogos/jogos.page.scss (0 bytes)
CREATE src/app/jogos/jogos.page.html (124 bytes)
CREATE src/app/jogos/jogos.page.spec.ts (640 bytes)
CREATE src/app/jogos/jogos.page.ts (252 bytes)
UPDATE src/app/app-routing.module.ts (647 bytes)
[OK] Generated page!
```

```
[→ PlayingScore git:(master) ✘ ionic g
? What would you like to generate? page
[?] Name/path of page: partidas
> ng generate page partidas --project=app
CREATE src/app/partidas/partidas-routing.module.ts (355 bytes)
CREATE src/app/partidas/partidas.module.ts (486 bytes)
CREATE src/app/partidas/partidas.page.scss (0 bytes)
CREATE src/app/partidas/partidas.page.html (127 bytes)
CREATE src/app/partidas/partidas.page.spec.ts (661 bytes)
CREATE src/app/partidas/partidas.page.ts (264 bytes)
UPDATE src/app/app-routing.module.ts (772 bytes)
[OK] Generated page!
```

```
[→ PlayingScore git:(master) ✘ ionic g
? What would you like to generate? page
[?] Name/path of page: marcador
> ng generate page marcador --project=app
CREATE src/app/marcador/marcador-routing.module.ts (355 bytes)
CREATE src/app/marcador/marcador.module.ts (486 bytes)
CREATE src/app/marcador/marcador.page.scss (0 bytes)
CREATE src/app/marcador/marcador.page.html (127 bytes)
CREATE src/app/marcador/marcador.page.spec.ts (661 bytes)
CREATE src/app/marcador/marcador.page.ts (264 bytes)
UPDATE src/app/app-routing.module.ts (897 bytes)
[OK] Generated page!
```

# Organizando Tabs

---

Vamos organizar as tabs padrões do sistema

- Agora ajustaremos as ações dos tabs

# Organizando Tabs

Vamos organizar as tabs padrões do projeto.

- Agora ajustaremos as ações dos tabs

## tabs

TS tabs-routing.module.ts

TS tabs.module.ts

⌚ tabs.page.html

⌚ tabs.page.scss

TS tabs.page.ts

TS tabs.page.spec.ts

## tabs.page.html

src > app > tabs > ⌚ tabs.page.html > ...

```
1  <ion-tabs>
2    <ion-tab-bar slot="bottom">
3
4      <ion-tab-button tab="tabPerfil">
5        <ion-icon name="person-circle-outline"></ion-icon>
6        <ion-label>Perfil</ion-label>
7      </ion-tab-button>
8
9      <ion-tab-button tab="tabJogos">
10        <ion-icon name="game-controller-outline"></ion-icon>
11        <ion-label>Jogos</ion-label>
12      </ion-tab-button>
13
14      <ion-tab-button tab="tabPartidas">
15        <ion-icon name="reader-outline"></ion-icon>
16        <ion-label>Partidas</ion-label>
17      </ion-tab-button>
18
19      <ion-tab-button tab="tabMarcador">
20        <ion-icon name="create-outline"></ion-icon>
21        <ion-label>Marcador</ion-label>
22      </ion-tab-button>
23
24    </ion-tab-bar>
25  </ion-tabs>
```

# Organizando Tabs

---

Vamos organizar as tabs padrões do sistema

- Agora ajustaremos as rotas das tabs

# Organizando Tabs

```
ts tabs-routing.module.ts ×
src > app > tabs > ts tabs-routing.module.ts > ...
2   import { RouterModule, Routes } from '@angular/router';
3   import { TabsPage } from './tabs.page';
4
5   const routes: Routes = [
6     {
7       path: 'tabs',
8       component: TabsPage,
9       children: [
10         {
11           path: 'tabPerfil',
12           loadChildren: () => import('../perfil/perfil.module').then( m => m.PerfilPageModule)
13         },
14         {
15           path: 'tabJogos',
16           loadChildren: () => import('../jogos/jogos.module').then(m => m.JogosPageModule)
17         },
18         {
19           path: 'tabPartidas',
20           loadChildren: () => import('../partidas/partidas.module').then(m => m.PartidasPageModule)
21         },
22         {
23           path: 'tabMarcador',
24           loadChildren: () => import('../marcador/marcador.module').then(m => m.MarcadorPageModule)
25         },
26         {
27           path: '',
28           redirectTo: '/tabs/tabJogos',
29           pathMatch: 'full'
30         }
31       ]
32     },
33     {
34       path: '',
35       redirectTo: '/tabs/tabJogos',
36       pathMatch: 'full'
37     }
38 ];
39
40 @NgModule({
41   imports: [RouterModule.forChild(routes)],
42   exports: [RouterModule]
43 })
44 export class TabsPageRoutingModule {}
```

# Organizando Tabs

```
ts tabs-routing.module.ts ×  
src > app > tabs > ts tabs-routing.module.ts > ...  
2 import { RouterModule, Routes } from '@angular/router';  
3 import { TabsPage } from './tabs.page';  
4  
5 const routes: Routes = [  
6   {  
7     path: 'tabs',  
8     component: TabsPage,  
9     children: [  
10       {  
11         path: 'tabPerfil',  
12         loadChildren: () => import('../perfil/perfil.module').then( m => m.PerfilPageModule)  
13       },  
14       {  
15         path: 'tabJogos',  
16         loadChildren: () => import('../jogos/jogos.module').then(m => m.JogosPageModule)  
17       },  
18       {  
19         path: 'tabPartidas',  
20         loadChildren: () => import('../partidas/partidas.module').then(m => m.PartidasPageModule)  
21       },  
22       {  
23         path: 'tabMarcador',  
24         loadChildren: () => import('../marcador/marcador.module').then(m => m.MarcadorPageModule)  
25       },  
26       {  
27         path: '',  
28         redirectTo: '/tabs/tabJogos',  
29         pathMatch: 'full'  
30       }  
31     ]  
32   },  
33   {  
34     path: '',  
35     redirectTo: '/tabs/tabJogos',  
36     pathMatch: 'full'  
37   }  
38 ];  
39  
40 @NgModule({  
41   imports: [RouterModule.forChild(routes)],  
42   exports: [RouterModule]  
43 })  
44 export class TabsPageRoutingModule {}
```

Organização das tabs

# Organizando Tabs

```
ts tabs-routing.module.ts x
src > app > tabs > ts tabs-routing.module.ts > ...
2   import { RouterModule, Routes } from '@angular/router';
3   import { TabsPage } from './tabs.page';
4
5   const routes: Routes = [
6     {
7       path: 'tabs',
8       component: TabsPage,
9       children: [
10         {
11           path: 'tabPerfil',
12           loadChildren: () => import('../perfil/perfil.module').then( m => m.PerfilPageModule)
13         },
14         {
15           path: 'tabJogos',
16           loadChildren: () => import('../jogos/jogos.module').then(m => m.JogosPageModule)
17         },
18         {
19           path: 'tabPartidas',
20           loadChildren: () => import('../partidas/partidas.module').then(m => m.PartidasPageModule)
21         },
22         {
23           path: 'tabMarcador',
24           loadChildren: () => import('../marcador/marcador.module').then(m => m.MarcadorPageModule)
25         },
26         {
27           path: '',
28           redirectTo: '/tabs/tabJogos',
29           pathMatch: 'full'
30         }
31       ]
32     },
33     {
34       path: '',
35       redirectTo: '/tabs/tabJogos',
36       pathMatch: 'full'
37     }
38   ];
39
40   @NgModule({
41     imports: [RouterModule.forChild(routes)],
42     exports: [RouterModule]
43   })
44   export class TabsPageRoutingModule {}
```

Caso nenhuma dessas rotas for selecionada, então carrega a de jogos

# Organizando Tabs

---

Vamos organizar as tabs padrões do sistema

- Agora ajustaremos os HTMLs, inserindo os devidos nomes nos cabeçalhos

# Organizando Tabs

Vamos organizar as tabs padrões do sistema

- Agora ajustaremos os HTMLs, inserindo os devidos nomes nos cabe

```
<> jogos.page.html <  
  
src > app > jogos > <> jogos.page.html > ...  
1   <ion-header [translucent] = "true">  
2     <ion-toolbar color = "primary">  
3       <ion-title>Jogos</ion-title>  
4     </ion-toolbar>  
5   </ion-header>
```

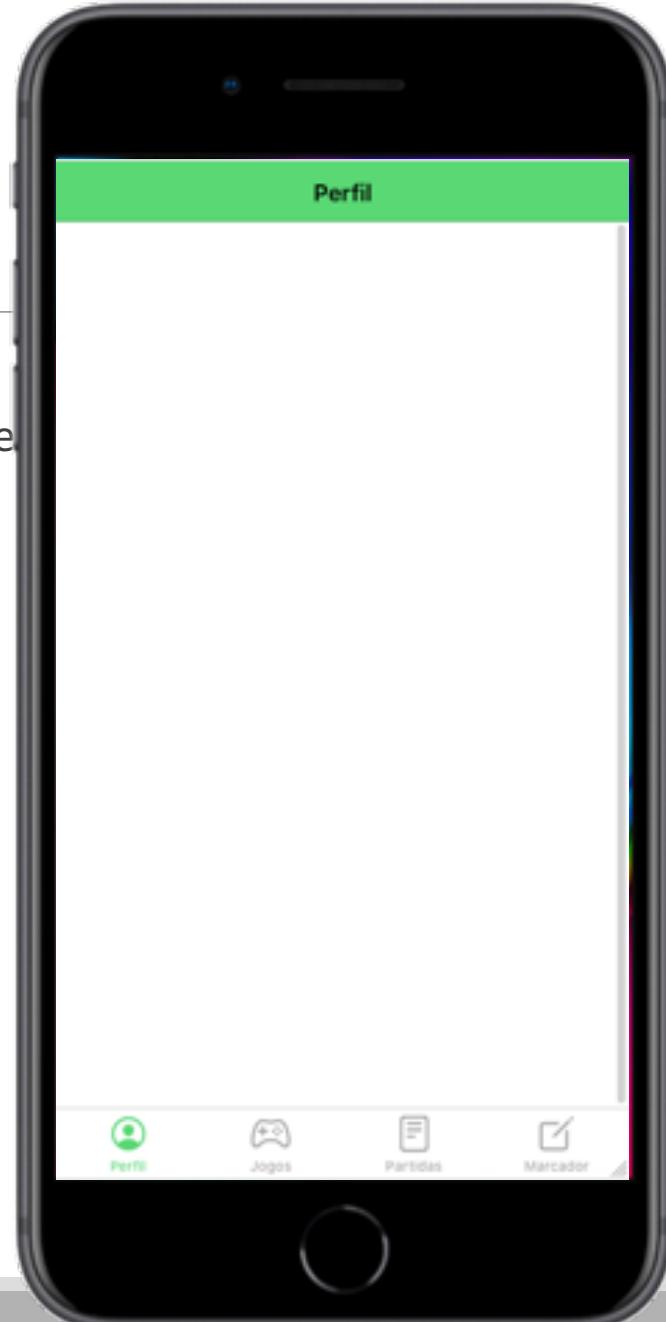


# Organizando Tabs

Vamos organizar as tabs padrões do sistema

- Agora ajustaremos os HTMLs, inserindo os devidos nomes nos cabeçalhos

```
perfil.page.html ×  
  
src > app > perfil > perfil.page.html > ...  
1   <ion-header [translucent] = "true">  
2     <ion-toolbar color = "primary">  
3       <ion-title> Perfil </ion-title>  
4     </ion-toolbar>  
5   </ion-header>
```



# Organizando Tabs

Vamos organizar as tabs padrões do sistema

- Agora ajustaremos os HTMLs, inserindo os devidos nomes nos cabeçalhos

```
<> partidas.page.html ×  
  
src > app > partidas > <> partidas.page.html > ...  
1   <ion-header [translucent] = "true">  
2     <ion-toolbar color = "primary">  
3       <ion-title>Partidas</ion-title>  
4     </ion-toolbar>  
5   </ion-header>
```



# Organizando Tabs

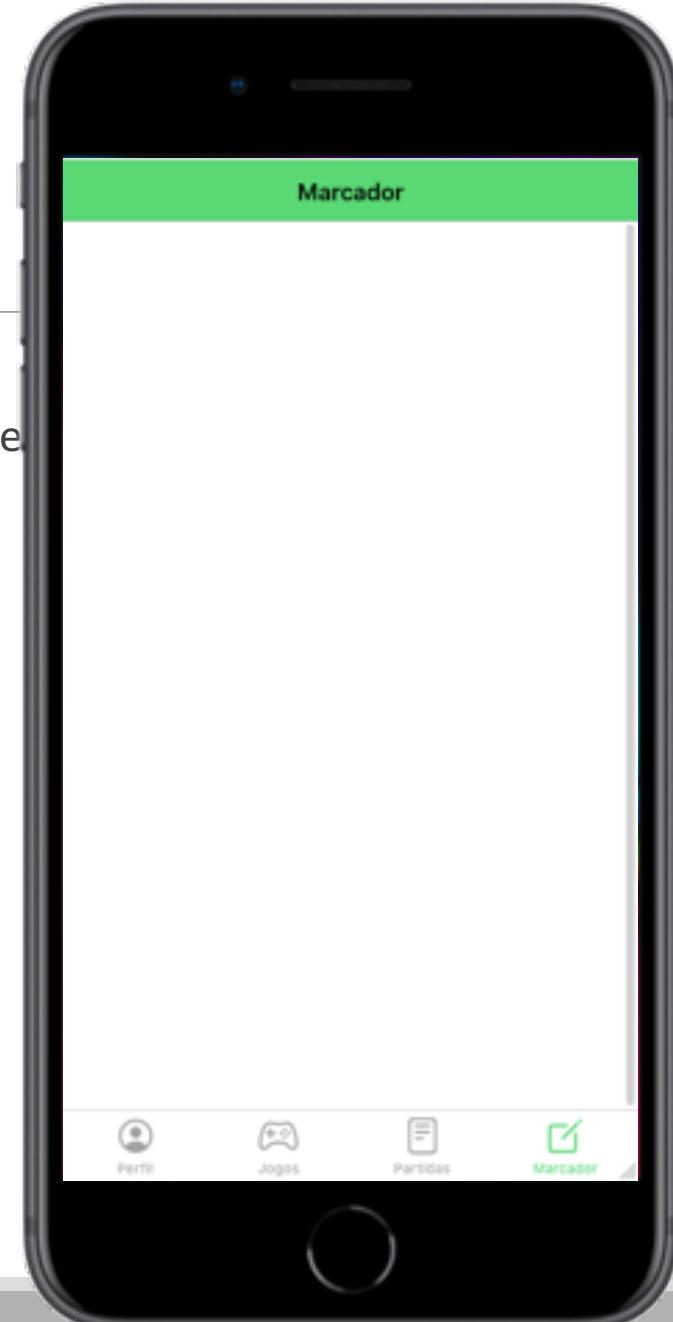
Vamos organizar as tabs padrões do sistema

- Agora ajustaremos os HTMLs, inserindo os devidos nomes nos cabeçalhos

```
↳ marcador.page.html ×
```

```
src > app > marcador > ↳ marcador.page.html > ...
```

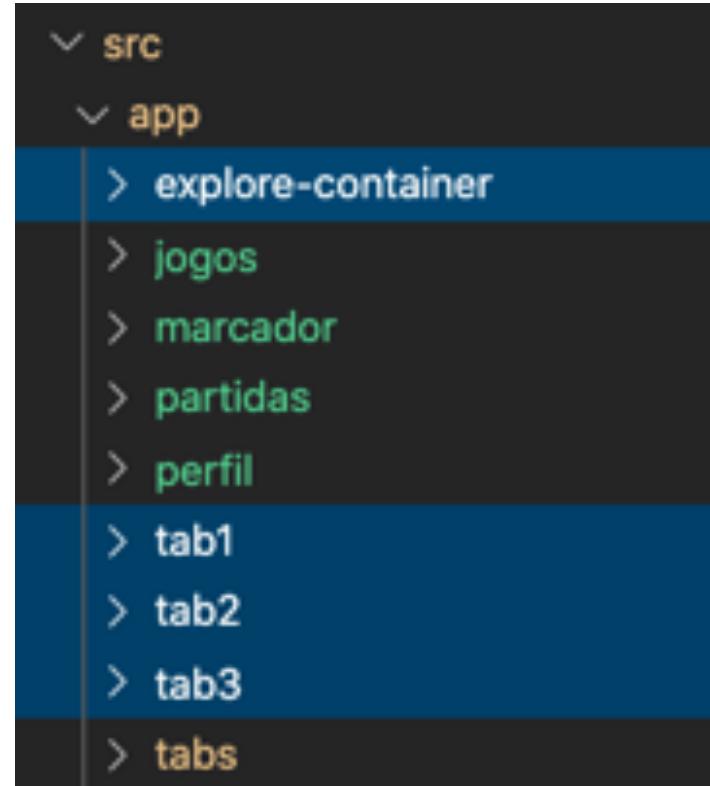
```
1   <ion-header [translucent] = "true">
2     <ion-toolbar color = "primary">
3       <ion-title> Marcador </ion-title>
4     </ion-toolbar>
5   </ion-header>
```



# Organizando Tabs

Vamos organizar as tabs padrões do sistema

- Para finalizar esta organização, podemos apagar as 4 pastas selecionadas na imagem a seguir
  - tab1
  - tab2
  - tab3
  - explore-container



Não usaremos mais estas páginas

Jogos Page

# Jogos Page

---

Vamos iniciar o projeto de nossa tela de listagem de jogos configurados

- O usuário terá seus jogos cadastrados e favoritados
- Também, poderá acompanhar os jogos (públicos) configurados pelos seus amigos

# Jogos Page

```
src
  app
    jogos
      jogos-routing.module.ts
      jogos.module.ts
      jogos.page.html
      jogos.page.scss
      jogos.page.ts
```

```
<!-- jogos.page.html -->

src > app > jogos > <!-- jogos.page.html --> ...
1   <ion-header [translucent]="true">
2     <ion-toolbar class="ion-text-center" color="primary">
3       <ion-title>Jogos</ion-title>
4     </ion-toolbar>
5   </ion-header>
6
7   <ion-content>
8     <ion-tabs>
9       <ion-tab-bar class="tab-bar" color="light" slot="top">
10      <ion-tab-button layout="icon-start" tab="tabPerfil">
11        <ion-icon name="star"></ion-icon>
12        <ion-label>Meu Favoritos</ion-label>
13      </ion-tab-button>
14
15      <ion-tab-button layout="icon-start" tab="tabJogos">
16        <ion-icon name="people"></ion-icon>
17        <ion-label>Jogos de Amigos</ion-label>
18      </ion-tab-button>
19    </ion-tab-bar>
20  </ion-tabs>
21 </ion-content>
```

# Jogos Page

```
src
  app
    jogos
      jogos-routing.module.ts
      jogos.module.ts
      jogos.page.html
      jogos.page.scss
      jogos.page.ts
```

## ↳ jogos.page.html X

```
src > app > jogos > ↳ jogos.page.html > ...
```

```
1   <ion-header [translucent]="true">
2     <ion-toolbar class="ion-text-center" color="primary">
3       <ion-title>Jogos</ion-title>
4     </ion-toolbar>
5   </ion-header>
```

O Header será configurado para conter seu texto centralizado, com tema primary

```
6
7   <ion-content>
8     <ion-tabs>
9       <ion-tab-bar class="tab-bar" color="light" slot="top">
10      <ion-tab-button layout="icon-start" tab="tabPerfil">
11        <ion-icon name="star"></ion-icon>
12        <ion-label>Meu Favoritos</ion-label>
13      </ion-tab-button>
14
15      <ion-tab-button layout="icon-start" tab="tabJogos">
16        <ion-icon name="people"></ion-icon>
17        <ion-label>Jogos de Amigos</ion-label>
18      </ion-tab-button>
19    </ion-tab-bar>
20  </ion-tabs>
21 </ion-content>
```

# Jogos Page

```
src
  app
    jogos
      jogos-routing.module.ts
      jogos.module.ts
      jogos.page.html
      jogos.page.scss
      jogos.page.ts
```

↳ jogos.page.html X

src > app > jogos > ↳ jogos.page.html > ...

```
1   <ion-header [translucent]="true">
2     <ion-toolbar class="ion-text-center" color="primary">
3       <ion-title>Jogos</ion-title>
4     </ion-toolbar>
5   </ion-header>
6
```

```
7   <ion-content>
8     <ion-tabs>
9       <ion-tab-bar class="tab-bar" color="light" slot="top">
10      <ion-tab-button layout="icon-start" tab="tabPerfil">
11        <ion-icon name="star"></ion-icon>
12        <ion-label>Meu Favoritos</ion-label>
13      </ion-tab-button>
14
15      <ion-tab-button layout="icon-start" tab="tabJogos">
16        <ion-icon name="people"></ion-icon>
17        <ion-label>Jogos de Amigos</ion-label>
18      </ion-tab-button>
19    </ion-tab-bar>
20  </ion-tabs>
21 </ion-content>
```

O conteúdo, até aqui, será  
uma novo tab-bar, porém  
superior

# Jogos Page

```
src
  app
    jogos
      jogos-routing.module.ts
      jogos.module.ts
      jogos.page.html
      jogos.page.scss
      jogos.page.ts
```

## ↳ jogos.page.html X

```
src > app > jogos > ↳ jogos.page.html > ...
1   <ion-header [translucent]="true">
2     <ion-toolbar class="ion-text-center" color="primary">
3       <ion-title>Jogos</ion-title>
4     </ion-toolbar>
5   </ion-header>
6
7   <ion-content>
8     <ion-tabs>
9       <ion-tab-bar class="tab-bar" color="light" slot="top">
10      <ion-tab-button layout="icon-start" tab="tabPerfil">
11        <ion-icon name="star"></ion-icon>
12        <ion-label>Meu Favoritos</ion-label>
13      </ion-tab-button>
14
15      <ion-tab-button layout="icon-start" tab="tabJogos">
16        <ion-icon name="people"></ion-icon>
17        <ion-label>Jogos de Amigos</ion-label>
18      </ion-tab-button>
19    </ion-tab-bar>
20  </ion-tabs>
21 </ion-content>
```

O primeiro elemento do tab-bar será para apresentar os favoritos

# Jogos Page

```
✓ src
  ✓ app
    ✓ jogos
      ts jogos-routing.module.ts
      ts jogos.module.ts
      <✓ jogos.page.html>
      ? jogos.page.scss
      ts jogos.page.ts
```

<✓ jogos.page.html >

src > app > jogos > <✓ jogos.page.html > ...

```
1   <ion-header [translucent]="true">
2     <ion-toolbar class="ion-text-center" color="primary">
3       <ion-title>Jogos</ion-title>
4     </ion-toolbar>
5   </ion-header>
6
7   <ion-content>
8     <ion-tabs>
9       <ion-tab-bar class="tab-bar" color="light" slot="top">
10      <ion-tab-button layout="icon-start" tab="tabPerfil">
11        <ion-icon name="star"></ion-icon>
12        <ion-label>Meu Favoritos</ion-label>
13      </ion-tab-button>
14
15      <ion-tab-button layout="icon-start" tab="tabJogos">
16        <ion-icon name="people"></ion-icon>
17        <ion-label>Jogos de Amigos</ion-label>
18      </ion-tab-button>
19
20    </ion-tab-bar>
21  </ion-tabs>
22
23  </ion-content>
```

O segundo elemento do tab-bar será para apresentar os jogos dos amigos

# Jogos Page

```
src
  app
    jogos
      jogos-routing.module.ts
      jogos.module.ts
      jogos.page.html
      jogos.page.scss
      jogos.page.ts
```

## ↳ jogos.page.html X

```
src > app > jogos > ↳ jogos.page.html > ...
1   <ion-header [translucent]="true">
2     <ion-toolbar class="ion-text-center" color="primary">
3       <ion-title>Jogos</ion-title>
4     </ion-toolbar>
5   </ion-header>
6
7   <ion-content>
8     <ion-tabs>
9       <ion-tab-bar class="tab-bar" color="light" slot="top">
10      <ion-tab-button layout="icon-start" tab="tabPerfil">
11        <ion-icon name="star"></ion-icon>
12        <ion-label>Meu Favoritos</ion-label>
13      </ion-tab-button>
14
15      <ion-tab-button layout="icon-start" tab="tabJogos">
16        <ion-icon name="people"></ion-icon>
17        <ion-label>Jogos de Amigos</ion-label>
18      </ion-tab-button>
19    </ion-tab-bar>
20  </ion-tabs>
21 </ion-content>
```

O tab-bar será na cor cinza (light), parte superior (slot top) e pertencente a classe "tab-bar" (configuramos no SCSS)

# Jogos Page

```
src
  app
    jogos
      jogos-routing.module.ts
      jogos.module.ts
      jogos.page.html
      jogos.page.scss
      jogos.page.ts
```

## ↳ jogos.page.html X

```
src > app > jogos > ↳ jogos.page.html > ...
```

```
1   <ion-header [translucent]="true">
2     <ion-toolbar class="ion-text-center" color="primary">
3       <ion-title>Jogos</ion-title>
4     </ion-toolbar>
5   </ion-header>
6
7   <ion-content>
8     <ion-tabs>
9       <ion-tab-bar class="tab-bar" color="light" slot="top">
10      <ion-tab-button layout="icon-start" tab="tabPerfil">
11        <ion-icon name="star"></ion-icon>
12        <ion-label>Meu Favoritos</ion-label>
13      </ion-tab-button>
14
15      <ion-tab-button layout="icon-start" tab="tabJogos">
16        <ion-icon name="people"></ion-icon>
17        <ion-label>Jogos de Amigos</ion-label>
18      </ion-tab-button>
19    </ion-tab-bar>
20  </ion-tabs>
21 </ion-content>
```

Ambos os tab-button terão  
o ícone a esquerda  
(icon-start) e a escrita  
depois

# Jogos Page

```
✓ src
  ✓ app
    ✓ jogos
      ts jogos-routing.module.ts
      ts jogos.module.ts
      <✓ jogos.page.html>
      ? jogos.page.scss
      ts jogos.page.ts
```

```
<✓ jogos.page.html >

src > app > jogos > <✓ jogos.page.html > ...
1   <ion-header [translucent] = "true">
2     <ion-toolbar class = "ion-text-center" color = "primary">
3       <ion-title>Jogos</ion-title>
4     </ion-toolbar>
5   </ion-header>
6
7   <ion-content>
8     <ion-tabs>
9       <ion-tab-bar class = "tab-bar" color = "light" slot = "top">
10      <ion-tab-button layout = "icon-start" tab = "tabPerfil">
11        <ion-icon name = "star"></ion-icon>
12        <ion-label>Meu Favoritos</ion-label>
13      </ion-tab-button>
14
15      <ion-tab-button layout = "icon-start" tab = "tabJogos">
16        <ion-icon name = "people"></ion-icon>
17        <ion-label>Jogos de Amigos</ion-label>
18      </ion-tab-button>
19    </ion-tab-bar>
20  </ion-tabs>
21 </ion-content>
```

Agora, vamos alterar o jogos.page.scss

## Jogos Page

```
└─ src
  └─ app
    └─ jogos
      ├─ ts jogos-routing.module.ts
      ├─ ts jogos.module.ts
      ├─ <> jogos.page.html
      └─ ? jogos.page.scss
      └─ ts jogos.page.ts
      └─ ts jogos.page.spec.ts
```

Agora, vamos alterar o jogos.page.scss

## Jogos Page

```
src
  app
    jogos
      jogos-routing.module.ts
      jogos.module.ts
      jogos.page.html
      jogos.page.scss
      jogos.page.ts
      jogos.page.spec.ts
```

### jogos.page.scss ×

```
src > app > jogos > jogos.page.scss
```

```
1 .tab-bar {
2   height: 30px;
3 }
4
5 ion-icon {
6   font-size: 12px;
7 }
```

Agora, vamos alterar o jogos.page.scss

## Jogos Page

```
src
  app
    jogos
      jogos-routing.module.ts
      jogos.module.ts
      jogos.page.html
      jogos.page.scss <-- selected
      jogos.page.ts
      jogos.page.spec.ts
```

### jogos.page.scss ×

```
src > app > jogos > jogos.page.scss
```

```
1 .tab-bar {
2   height: 30px;
3 }
```

```
4
5 ion-icon {
6   font-size: 12px;
7 }
```

Criamos a classe para  
ajustar a altura do tab-bar

Agora, vamos alterar o jogos.page.scss

## Jogos Page

```
src
  app
    jogos
      jogos-routing.module.ts
      jogos.module.ts
      jogos.page.html
      jogos.page.scss
      jogos.page.ts
      jogos.page.spec.ts
```

jogos.page.scss ×

src > app > jogos > jogos.page.scss

```
1 .tab-bar {
2   height: 30px;
3 }
```

```
4
5 ion-icon {
6   font-size: 12px;
7 }
```

Alteramos a classe de ícones para reduzir seu tamanho

# Jogos Page



# Jogos Page



**GitHub**  
**V2.0**

# Criando as Pages de Jogos

# Criando as Pages de Jogos

---

Temos que criar duas novas páginas

- JogosFavoritos
- JogosAmigos

Para criarmos estas novas páginas vamos executar, via terminal, dentro da pasta do projeto

```
$ ionic g
```

# Criando as Pages de Jogos

```
[→ PlayingScore git:(master) ionic g
? What would you like to generate? page
[?] Name/path of page: jogos_favoritos
\> ng generate page jogos_favoritos --project=app
CREATE src/app/jogos-favoritos/jogos-favoritos-routing.module.ts (380 bytes)
CREATE src/app/jogos-favoritos/jogos-favoritos.module.ts (530 bytes)
CREATE src/app/jogos-favoritos/jogos-favoritos.page.scss (0 bytes)
CREATE src/app/jogos-favoritos/jogos-favoritos.page.html (134 bytes)
CREATE src/app/jogos-favoritos/jogos-favoritos.page.spec.ts (704 bytes)
CREATE src/app/jogos-favoritos/jogos-favoritos.page.ts (291 bytes)
UPDATE src/app/app-routing.module.ts (1049 bytes)
[OK] Generated page!
```

```
[→ PlayingScore git:(master) ✘ ionic g
? What would you like to generate? page
[?] Name/path of page: jogos_amigos
> ng generate page jogos_amigos --project=app
CREATE src/app/jogos-amigos/jogos-amigos-routing.module.ts (368 bytes)
CREATE src/app/jogos-amigos/jogos-amigos.module.ts (589 bytes)
CREATE src/app/jogos-amigos/jogos-amigos.page.scss (0 bytes)
CREATE src/app/jogos-amigos/jogos-amigos.page.html (131 bytes)
CREATE src/app/jogos-amigos/jogos-amigos.page.spec.ts (683 bytes)
CREATE src/app/jogos-amigos/jogos-amigos.page.ts (279 bytes)
UPDATE src/app/app-routing.module.ts (1189 bytes)
[OK] Generated page!
```

# Criando as Pages de Jogos

```
[+ PlayingScore git:(master) ionic g
? What would you like to generate?
[?] Name/path of page to generate
\> ng generate page
CREATE src/app/jogos
CREATE src/app/jogos
CREATE src/app/jogos
CREATE src/app/jogos
CREATE src/app/jogos
CREATE src/app/jogos
UPDATE src/app/app-routing.module.ts
[OK] Generated page!
```

```
[+ PlayingScore git:(master) ionic g
? What would you like to generate?
[?] Name/path of page to generate
> ng generate page
CREATE src/app/jogos
CREATE src/app/jogos
CREATE src/app/jogos
CREATE src/app/jogos
CREATE src/app/jogos
CREATE src/app/jogos
UPDATE src/app/app-routing.module.ts
[OK] Generated page!
```

```
src
  app
    jogos
    jogos-amigos
    jogos-favoritos
    marcador
    partidas
    perfil
    tabs
      module.ts (368 bytes)
      09 bytes)
      bytes)
      31 bytes)
      (683 bytes)
      bytes)
```

```
  dule.ts (380 bytes)
  (530 bytes)
  (0 bytes)
  (134 bytes)
  ts (704 bytes)
  91 bytes)
```

```
  UPDATE src/app/app-routing.module.ts (1189 bytes)
  [OK] Generated page!
```

# Criando as Pages de Jogos

---

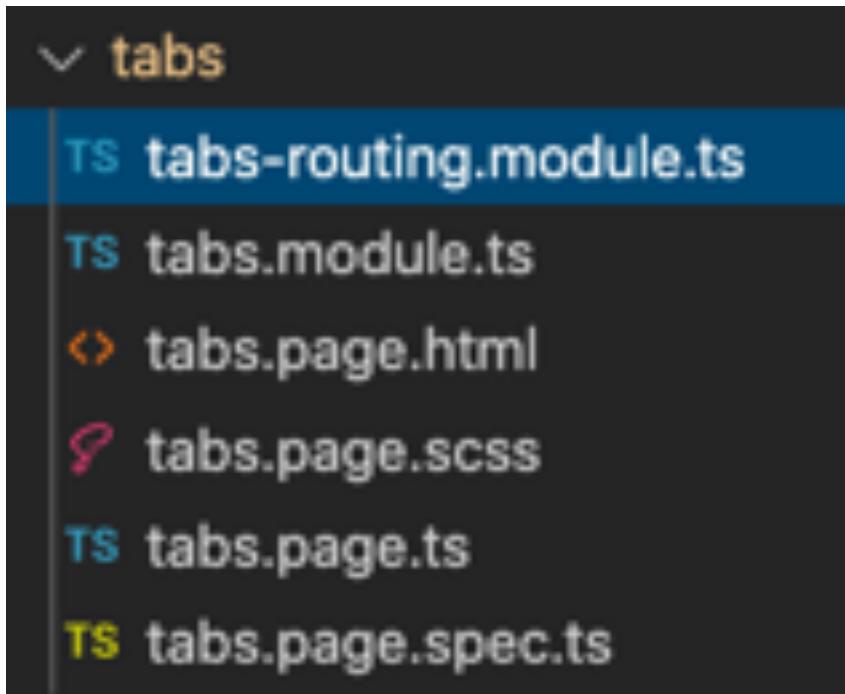
Com estas novas páginas criadas, vamos configurar a navegação de rotas

- Para isso, vamos acessar o arquivo tabs-routing.module.ts

# Criando as Pages de Jogos

Com estas novas páginas criadas, vamos configurar a navegação de rotas

- Para isso, vamos acessar o arquivo tabs-routing.module.ts



PlayingScore &gt; src &gt; app &gt; tabs &gt; TS tabs-routing.module.ts &gt; [e] routes &gt; ⚡ children &gt; ⚡ loadChildren

```
1 import { NgModule } from '@angular/core';
2 import { RouterModule, Routes } from '@angular/router';
3 import { TabsPage } from './tabs.page';

4
5 const routes: Routes = [
6   {
7     path: 'tabs',
8     component: TabsPage,
9     children: [
10       {
11         path: 'tabPerfil',
12         loadChildren: () => import('../perfil/perfil.module').then(m => m.PerfilPageModule)
13       },
14       {
15         path: 'tabJogos',
16         loadChildren: () => import('../jogos/jogos.module').then(m => m.JogosPageModule),
17       },
18       {
19         path: 'tabJogos/tabJogosFavoritos',
20         loadChildren: () => import('../jogos-favoritos/jogos-favoritos.module').then(m => m.JogosFavoritosPageModule)
21       },
22       {
23         path: 'tabJogos/tabJogosAmigos',
24         loadChildren: () => import('../jogos-amigos/jogos-amigos.module').then(m => m.JogosAmigosPageModule)
25       },
26     ],
27   }
];
```

PlayingScore &gt; src &gt; app &gt; tabs &gt; TS tabs-routing.module.ts &gt; [e] routes &gt; ⚡ children &gt; ⏪ loadChildren

```
1 import { NgModule } from '@angular/core';
2 import { RouterModule, Routes } from '@angular/router';
3 import { TabsPage } from './tabs.page';

4
5 const routes: Routes = [
6   {
7     path: 'tabs',
8     component: TabsPage,
9     children: [
10       {
11         path: 'tabPerfil',
12         loadChildren: () => import('../perfil/perfil.module').then(m => m.PerfilPageModule)
13       },
14       {
15         path: 'tabJogos',
16         loadChildren: () => import('../jogos/jogos.module').then(m => m.JogosPageModule),
17       },
18       {
19         path: 'tabJogos/tabJogosFavoritos',
20         loadChildren: () => import('../jogos-favoritos/jogos-favoritos.module').then(m => m.JogosFavoritosPageModule)
21       },
22       {
23         path: 'tabJogos/tabJogosAmigos',
24         loadChildren: () => import('../jogos-amigos/jogos-amigos.module').then(m => m.JogosAmigosPageModule)
25       },
26     ],
27   }
];
```

PlayingScore > src > app > tabs > **TS tabs-routing.module.ts** > [e] routes > ⚡ children > ⏪ loadChildren

```
1 import { NgModule } from '@angular/core';
2 import { RouterModule, Routes } from '@angular/router';
3 import { TabsPage } from './tabs.page';

4
5 const routes: Routes = [
6   {
7     path: 'tabs',
8     component: TabsPage,
9     children: [
10       {
11         path: 'tabPerfil',
12         loadChildren: () => import('../perfil/perfil.module').then(m => m.PerfilPageModule)
13       },
14       {
15         path: 'tabJogos',
16         loadChildren: () => import('../jogos/jogos.module').then(m => m.JogosPageModule),
17       },
18       {
19         path: 'tabJogos/tabJogosFavoritos',
20         loadChildren: () => import('../jogos-favoritos/jogos-favoritos.module').then(m => m.JogosFavoritosPageModule)
21       },
22       {
23         path: 'tabJogos/tabJogosAmigos',
24         loadChildren: () => import('../jogos-amigos/jogos-amigos.module').then(m => m.JogosAmigosPageModule)
25       },
26     ],
27   }
];
```

# Criando as Pages de Jogos

---

Com estas novas páginas criadas, vamos configurar a navegação de rotas

- Agora que as rotas estão criadas, vamos ajustar as ações dos botões da tela de Jogos

# Criando as Pages de Jogos

↳ jogos.page.html ✘

```
PlayingScore > src > app > jogos > ↳ jogos.page.html > ...
1   <ion-header [translucent]="true">
2     <ion-toolbar class="ion-text-center" color="primary">
3       <ion-title>Jogos</ion-title>
4     </ion-toolbar>
5   </ion-header>
6
7   <ion-content>
8     <ion-tabs>
9       <ion-tab-bar class="tab-bar" color="light" slot="top">
10      <ion-tab-button layout="icon-start" tab="tabJogosFavoritos">
11        <ion-icon name="star"></ion-icon>
12        <ion-label>Meus Favoritos</ion-label>
13      </ion-tab-button>
14
15      <ion-tab-button layout="icon-start" tab="tabJogosAmigos">
16        <ion-icon name="people"></ion-icon>
17        <ion-label>Jogos de Amigos</ion-label>
18      </ion-tab-button>
19    </ion-tab-bar>
20  </ion-tabs>
21
22  </ion-content>
```

# Criando as Pages de Jogos

↳ jogos.page.html ✘

PlayingScore > src > app > jogos > ↳ jogos.page.html > ...

```
1   <ion-header [translucent]="true">
2     <ion-toolbar class="ion-text-center" color="primary">
3       <ion-title>Jogos</ion-title>
4     </ion-toolbar>
5   </ion-header>
6
7   <ion-content>
8     <ion-tabs>
9       <ion-tab-bar class="tab-bar" color="light" slot="top">
10      <ion-tab-button layout="icon-start" tab="tabJogosFavoritos">
11        <ion-icon name="star"></ion-icon>
12        <ion-label>Meus Favoritos</ion-label>
13      </ion-tab-button>
14
15      <ion-tab-button layout="icon-start" tab="tabJogosAmigos">
16        <ion-icon name="people"></ion-icon>
17        <ion-label>Jogos de Amigos</ion-label>
18      </ion-tab-button>
19    </ion-tab-bar>
20  </ion-tabs>
21
22 </ion-content>
```

# Criando as Pages de Jogos

↳ jogos.page.html ✘

PlayingScore > src > app > jogos > ↳ jogos.page.html > ...

```
1   <ion-header [translucent]="true">
2     <ion-toolbar class="ion-text-center" color="primary">
3       <ion-title>Jogos</ion-title>
4     </ion-toolbar>
5   </ion-header>
6
7   <ion-content>
8     <ion-tabs>
9       <ion-tab-bar class="tab-bar" color="light" slot="top">
10      <ion-tab-button layout="icon-start" tab="tabJogosFavoritos">
11        <ion-icon name="star"></ion-icon>
12        <ion-label>Meus Favoritos</ion-label>
13      </ion-tab-button>
14
15      <ion-tab-button layout="icon-start" tab="tabJogosAmigos">
16        <ion-icon name="people"></ion-icon>
17        <ion-label>Jogos de Amigos</ion-label>
18      </ion-tab-button>
19    </ion-tab-bar>
20  </ion-tabs>
21
22  </ion-content>
```

# Criando as Pages de Jogos

---

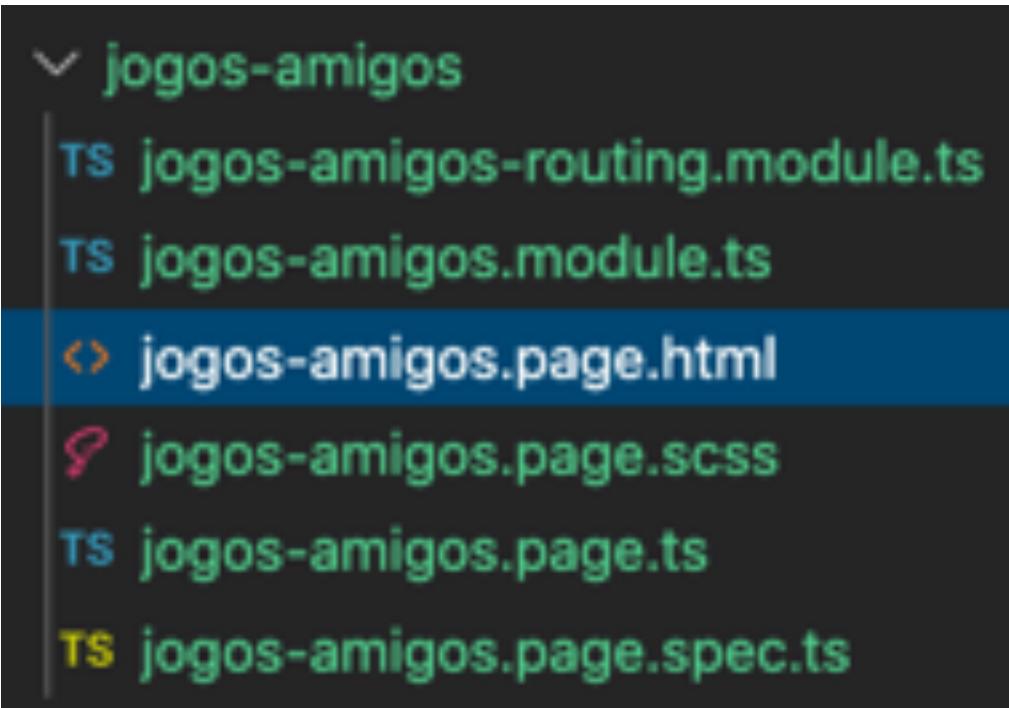
Com estas novas páginas criadas, vamos configurar a navegação de rotas

- Até aqui, ok, conseguimos fazer a navegação para estar telas
- Mas, vamos projetar elas

# Criando as Pages de Jogos

Com estas novas páginas criadas, vamos configurar a navegação de rotas

- Até aqui, ok, conseguimos fazer a navegação para estar telas
- Mas, vamos projetar elas



```
└─ jogos-amigos
    TS jogos-amigos-routing.module.ts
    TS jogos-amigos.module.ts
    < TS jogos-amigos.page.html
    SCSS jogos-amigos.page.scss
    TS jogos-amigos.page.ts
    TS jogos-amigos.page.spec.ts
```

Começando pela jogos-amigos

# Criando as Pages de Jogos

```
<> jogos-amigos.page.html <  
PlayingScore > src > app > jogos-amigos > <> jogos-amigos.page.html > ...  
1   <ion-header>  
2     <ion-toolbar class="ion-text-center" color="primary">  
3       <ion-buttons slot="start">  
4         <ion-back-button defaultHref="tabs"></ion-back-button>  
5         <ion-title>Jogos</ion-title>  
6       </ion-buttons>  
7     </ion-toolbar>  
8   </ion-header>  
9  
10  <ion-content>  
11    <ion-tabs>  
12      <ion-tab-bar class="tab-bar" color="light" slot="top">  
13        <ion-tab-button layout="icon-start">  
14          <ion-icon name="people"></ion-icon>  
15          <ion-label>Jogos de amigos</ion-label>  
16        </ion-tab-button>  
17      </ion-tab-bar>  
18    </ion-tabs>  
19  </ion-content>
```

# Criando as Pages de Jogos

↳ jogos-amigos.page.html ×

PlayingScore > src > app > jogos-amigos > ↳ jogos-amigos.page.html > ...

```
1   <ion-header>
2     <ion-toolbar class="ion-text-center" color="primary">
3       <ion-buttons slot="start">
4         <ion-back-button defaultHref="tabs"></ion-back-button>
5         <ion-title>Jogos</ion-title>
6       </ion-buttons>
7     </ion-toolbar>
8   </ion-header>
9
10  <ion-content>
11    <ion-tabs>
12      <ion-tab-bar class="tab-bar" color="light" slot="top">
13        <ion-tab-button layout="icon-start">
14          <ion-icon name="people"></ion-icon>
15          <ion-label>Jogos de amigos</ion-label>
16        </ion-tab-button>
17      </ion-tab-bar>
18    </ion-tabs>
19  </ion-content>
```

Vamos projetar o cabeçalho

# Criando as Pages de Jogos

↳ jogos-amigos.page.html ×

PlayingScore > src > app > jogos-amigos > ↳ jogos-amigos.page.html > ...

```
1   <ion-header>
2   <ion-toolbar class="ion-text-center" color="primary">
3       <ion-buttons slot="start">
4           <ion-back-button defaultHref="tabs"></ion-back-button>
5           <ion-title>Jogos</ion-title>
6       </ion-buttons>
7   </ion-toolbar>
8 </ion-header>
9
10  <ion-content>
11      <ion-tabs>
12          <ion-tab-bar class="tab-bar" color="light" slot="top">
13              <ion-tab-button layout="icon-start">
14                  <ion-icon name="people"></ion-icon>
15                  <ion-label>Jogos de amigos</ion-label>
16              </ion-tab-button>
17          </ion-tab-bar>
18      </ion-tabs>
19  </ion-content>
```

Apresentamos uma toolbar com cor primaria (verde) e com texto centralizado (padrão até aqui)

# Criando as Pages de Jogos

↳ jogos-amigos.page.html ×

PlayingScore > src > app > jogos-amigos > ↳ jogos-amigos.page.html > ...

```
1   <ion-header>
2     <ion-toolbar class="ion-text-center" color="primary">
3       <ion-buttons slot="start">
4         <ion-back-button defaultHref="tabs"></ion-back-button>
5         <ion-title>Jogos</ion-title>
6       </ion-buttons>
7     </ion-toolbar>
8   </ion-header>
9
10  <ion-content>
11    <ion-tabs>
12      <ion-tab-bar class="tab-bar" color="light" slot="top">
13        <ion-tab-button layout="icon-start">
14          <ion-icon name="people"></ion-icon>
15          <ion-label>Jogos de amigos</ion-label>
16        </ion-tab-button>
17      </ion-tab-bar>
18    </ion-tabs>
19  </ion-content>
```

Adicionamos um botão com posição "start"

# Criando as Pages de Jogos

↳ jogos-amigos.page.html ×

PlayingScore > src > app > jogos-amigos > ↳ jogos-amigos.page.html > ...

```
1   <ion-header>
2     <ion-toolbar class="ion-text-center" color="primary">
3       <ion-buttons slot="start">
4         <ion-back-button defaultHref="tabs"></ion-back-button>
5       <ion-title>Jogos</ion-title>
6     </ion-buttons>
7   </ion-toolbar>
8 </ion-header>
9
10  <ion-content>
11    <ion-tabs>
12      <ion-tab-bar class="tab-bar" color="light" slot="top">
13        <ion-tab-button layout="icon-start">
14          <ion-icon name="people"></ion-icon>
15          <ion-label>Jogos de amigos</ion-label>
16        </ion-tab-button>
17      </ion-tab-bar>
18    </ion-tabs>
19  </ion-content>
```

Adicionamos o botão de controle para voltar

# Criando as Pages de Jogos

↳ jogos-amigos.page.html ×

PlayingScore > src > app > jogos-amigos > ↳ jogos-amigos.page.html > ...

```
1   <ion-header>
2     <ion-toolbar class="ion-text-center" color="primary">
3       <ion-buttons slot="start">
4         <ion-back-button defaultHref="tabs"></ion-back-button>
5         <ion-title>Jogos</ion-title>
6       </ion-buttons>
7     </ion-toolbar>
8   </ion-header>
9
10  <ion-content>
11    <ion-tabs>
12      <ion-tab-bar class="tab-bar" color="light" slot="top">
13        <ion-tab-button layout="icon-start">
14          <ion-icon name="people"></ion-icon>
15          <ion-label>Jogos de amigos</ion-label>
16        </ion-tab-button>
17      </ion-tab-bar>
18    </ion-tabs>
19  </ion-content>
```

A função deste botão é retornar para a página "tabs"

# Criando as Pages de Jogos

↳ jogos-amigos.page.html ×

PlayingScore > src > app > jogos-amigos > ↳ jogos-amigos.page.html > ...

```
1   <ion-header>
2     <ion-toolbar class="ion-text-center" color="primary">
3       <ion-buttons slot="start">
4         <ion-back-button defaultHref="tabs"></ion-back-button>
5         <ion-title>Jogos</ion-title>
6       </ion-buttons>
7     </ion-toolbar>
```

TS tabs-routing.module.ts ×

PlayingScore > src > app > tabs > TS tabs-routing.mo

```
49   {
50     path: '',
51     redirectTo: '/tabs/tabJogos',
52     pathMatch: 'full'
53 }
```

" color="light" slot="top">  
icon-start">  
></ion-icon>  
gos</ion-label>

```
18   </ion-tabs>
19 </ion-content>
```

Lembrando que a TabsPage redireciona para tabJogos

# Criando as Pages de Jogos

↳ jogos-amigos.page.html ×

PlayingScore > src > app > jogos-amigos > ↳ jogos-amigos.page.html > ...

```
1   <ion-header>
2     <ion-toolbar class="ion-text-center" color="primary">
3       <ion-buttons slot="start">
4         <ion-back-button defaultHref="tabs"></ion-back-button>
5         <ion-title>Jogos</ion-title>
6       </ion-buttons>
7     </ion-toolbar>
8   </ion-header>
9
10  <ion-content>
11    <ion-tabs>
12      <ion-tab-bar class="tab-bar" color="light" slot="top">
13        <ion-tab-button layout="icon-start">
14          <ion-icon name="people"></ion-icon>
15          <ion-label>Jogos de amigos</ion-label>
16        </ion-tab-button>
17      </ion-tab-bar>
18    </ion-tabs>
19  </ion-content>
```

Mantemos o título jogos

# Criando as Pages de Jogos

```
<> jogos-amigos.page.html <  
PlayingScore > src > app > jogos-amigos > <> jogos-amigos.page.html > ...  
1   <ion-header>  
2     <ion-toolbar class="ion-text-center" color="primary">  
3       <ion-buttons slot="start">  
4         <ion-back-button defaultHref="tabs"></ion-back-button>  
5         <ion-title>Jogos</ion-title>  
6       </ion-buttons>  
7     </ion-toolbar>  
8   </ion-header>  
9  
10  <ion-content>  
11    <ion-tabs>  
12      <ion-tab-bar class="tab-bar" color="light" slot="top">  
13        <ion-tab-button layout="icon-start">  
14          <ion-icon name="people"></ion-icon>  
15          <ion-label>Jogos de amigos</ion-label>  
16        </ion-tab-button>  
17      </ion-tab-bar>  
18    </ion-tabs>  
19  </ion-content>
```

# Criando as Pages de Jogos

↳ jogos-amigos.page.html ×

PlayingScore > src > app > jogos-amigos > ↳ jogos-amigos.page.html > ...

```
1   <ion-header>
2     <ion-toolbar class="ion-text-center" color="primary">
3       <ion-buttons slot="start">
4         <ion-back-button defaultHref="tabs"></ion-back-button>
5         <ion-title>Jogos</ion-title>
6       </ion-buttons>
7     </ion-toolbar>
8   </ion-header>
9
10  <ion-content>
11    <ion-tabs>
12      <ion-tab-bar class="tab-bar" color="light" slot="top">
13        <ion-tab-button layout="icon-start">
14          <ion-icon name="people"></ion-icon>
15          <ion-label>Jogos de amigos</ion-label>
16        </ion-tab-button>
17      </ion-tab-bar>
18    </ion-tabs>
19  </ion-content>
```

Vamos projetar o conteúdo

# Criando as Pages de Jogos

↳ jogos-amigos.page.html ×

PlayingScore > src > app > jogos-amigos > ↳ jogos-amigos.page.html > ...

```
1   <ion-header>
2     <ion-toolbar class="ion-text-center" color="primary">
3       <ion-buttons slot="start">
4         <ion-back-button defaultHref="tabs"></ion-back-button>
5         <ion-title>Jogos</ion-title>
6       </ion-buttons>
7     </ion-toolbar>
8   </ion-header>
9
10  <ion-content>
11    <ion-tabs>
12      <ion-tab-bar class="tab-bar" color="light" slot="top">
13        <ion-tab-button layout="icon-start">
14          <ion-icon name="people"></ion-icon>
15          <ion-label>Jogos de amigos</ion-label>
16        </ion-tab-button>
17      </ion-tab-bar>
18    </ion-tabs>
19  </ion-content>
```

Criamos um tab com cor light (cinza fraco) e classe tab-bar. Mesma configuração da tela de jogos

# Criando as Pages de Jogos

↳ jogos-amigos.page.html ×

PlayingScore > src > app > jogos-amigos > ↳ jogos-amigos.page.html > ...

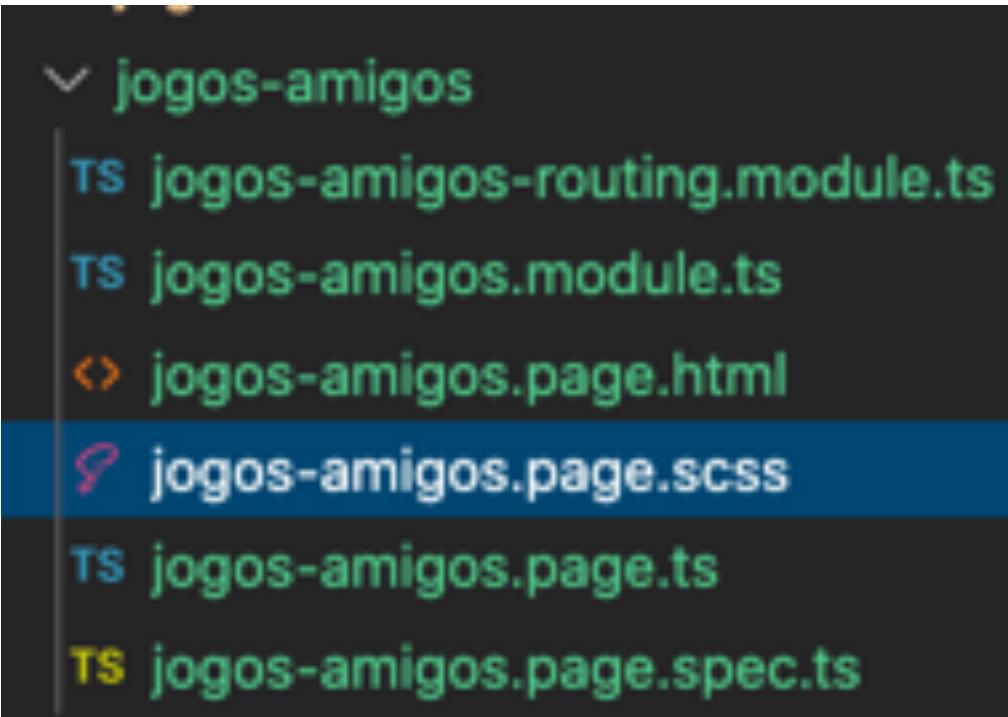
```
1   <ion-header>
2     <ion-toolbar class="ion-text-center" color="primary">
3       <ion-buttons slot="start">
4         <ion-back-button defaultHref="tabs"></ion-back-button>
5         <ion-title>Jogos</ion-title>
6       </ion-buttons>
7     </ion-toolbar>
8   </ion-header>
9
10  <ion-content>
11    <ion-tabs>
12      <ion-tab-bar class="tab-bar" color="light" slot="top">
13        <ion-tab-button layout="icon-start">
14          <ion-icon name="people"></ion-icon>
15          <ion-label>Jogos de amigos</ion-label>
16        </ion-tab-button>
17      </ion-tab-bar>
18    </ion-tabs>
19  </ion-content>
```

Criamos um conteúdo com ícone e texto

# Criando as Pages de Jogos

Com estas novas páginas criadas, vamos configurar a navegação de rotas

- Até aqui, ok, conseguimos fazer a navegação para estar telas
- Mas, vamos projetar elas

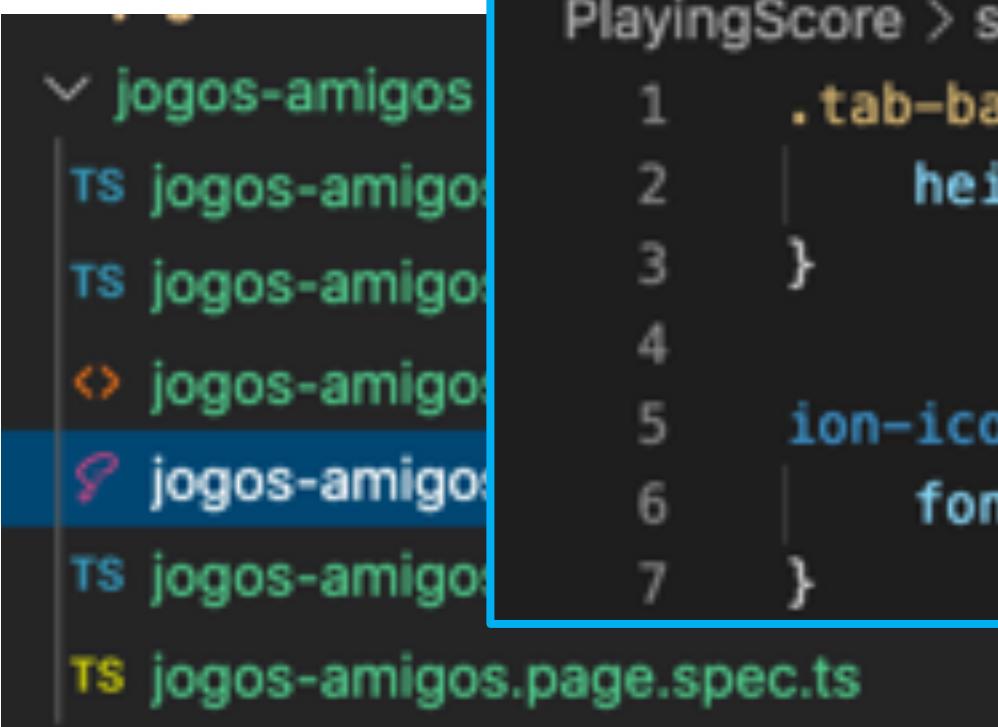


Agora, vamos ajustar o SCSS

# Criando as Pages de Jogos

Com estas novas páginas

- Até aqui, ok, consegui
- Mas, vamos projetar e



♀ jogos-amigos.page.scss ×

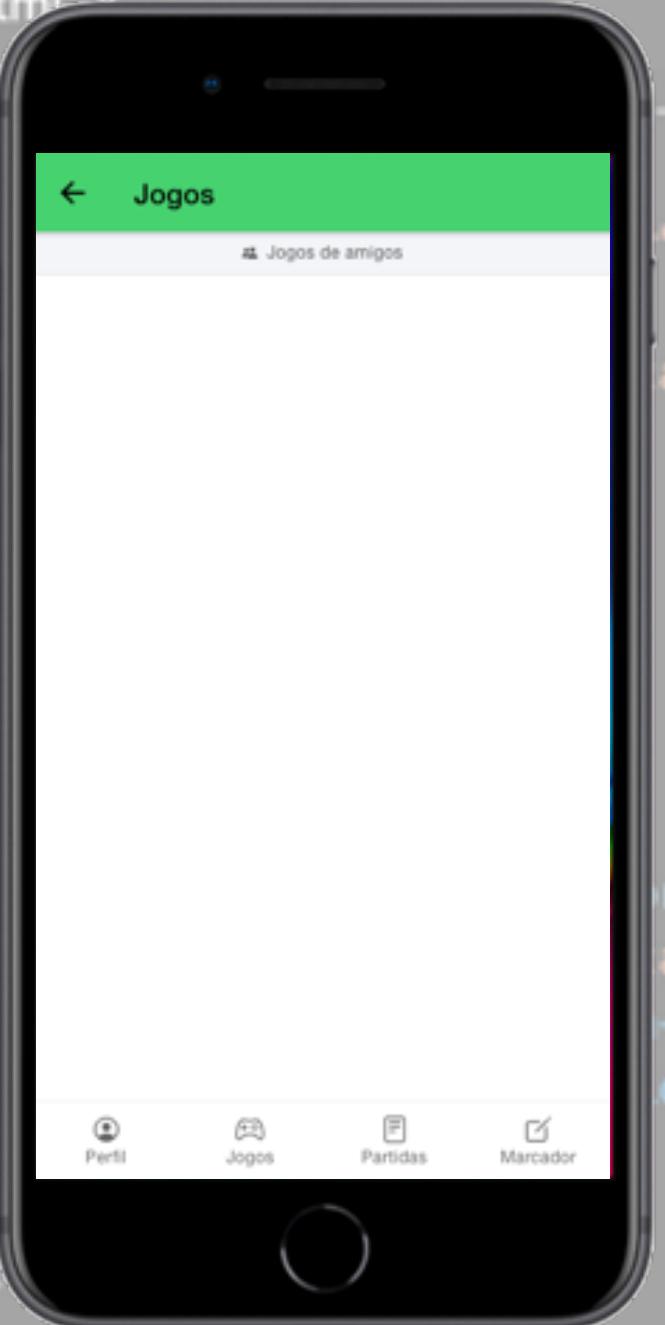
PlayingScore > src > app > jogos-amigos >

```
1  .tab-bar {  
2    height: 30px;  
3  }  
4  
5  ion-icon {  
6    font-size: 12px;  
7  }
```

SCSS

# Criando as Pages de Jogos

```
<-- jogos-amigos.page.html -->  
PlayingScore > src > app > jogos-amigos.page.html > ...  
" color="primary">  
abs"></ion-back-button>  
  
r="light" slot="top">  
art">  
-icon>  
on-label>  
  
1   <ion-header>  
2     <ion-toolbar color="primary">  
3       <ion-buttons side="start" style="background-color: #f0f0f0; border-radius: 10px; padding: 5px; margin-bottom: 10px;">  
4         <ion-back-button style="color: inherit; font-size: 1.5em; margin-right: 10px;">  
5           <ion-icon name="arrow-back" style="font-size: 1.5em; color: inherit; margin-right: 5px;"></ion-icon>  
6           <ion-label style="margin-right: 10px;">Jogos de amigos7         </ion-back-button>  
8       </ion-buttons>  
9     </ion-toolbar>  
10    <ion-content>  
11      <ion-tabs style="border-bottom: 1px solid #ccc; border-radius: 10px; padding: 5px; margin-bottom: 10px;">  
12        <ion-tab>  
13          <ion-tab-bar style="border-bottom: 1px solid #ccc; border-radius: 10px; padding: 5px; margin-bottom: 10px;">  
14            <ion-tab-item style="width: 25%; text-align: center; border-bottom: 1px solid #ccc; border-radius: 10px; padding: 5px; margin-bottom: 10px;">  
15              <ion-label style="font-size: 1.2em; margin-bottom: 5px;">Perfil16              <ion-icon name="person" style="font-size: 1.5em; color: inherit; margin-bottom: 5px;"></ion-icon>  
17            </ion-tab-item>  
18            <ion-tab-item style="width: 25%; text-align: center; border-bottom: 1px solid #ccc; border-radius: 10px; padding: 5px; margin-bottom: 10px;">  
19              <ion-label style="font-size: 1.2em; margin-bottom: 5px;">Jogos20              <ion-icon name="game-controller" style="font-size: 1.5em; color: inherit; margin-bottom: 5px;"></ion-icon>  
21            </ion-tab-item>  
22            <ion-tab-item style="width: 25%; text-align: center; border-bottom: 1px solid #ccc; border-radius: 10px; padding: 5px; margin-bottom: 10px;">  
23              <ion-label style="font-size: 1.2em; margin-bottom: 5px;">Partidas24              <ion-icon name="list" style="font-size: 1.5em; color: inherit; margin-bottom: 5px;"></ion-icon>  
25            </ion-tab-item>  
26            <ion-tab-item style="width: 25%; text-align: center; border-bottom: 1px solid #ccc; border-radius: 10px; padding: 5px; margin-bottom: 10px;">  
27              <ion-label style="font-size: 1.2em; margin-bottom: 5px;">Marcador28              <ion-icon name="checkmark" style="font-size: 1.5em; color: inherit; margin-bottom: 5px;"></ion-icon>  
29            </ion-tab-item>  
30          </ion-tab-bar>  
31        </ion-tab>  
32      </ion-tabs>  
33    </ion-content>
```

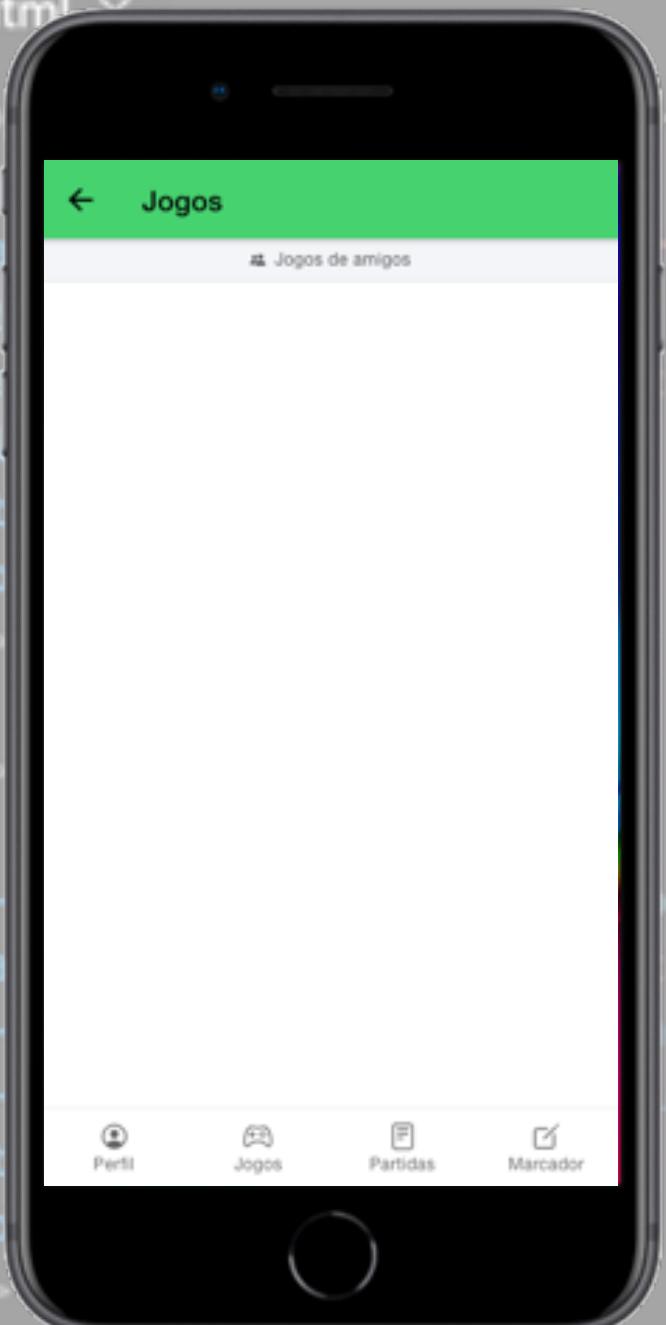


# Criando as Pages de Jogos

↳ jogos-amigos.page.html ↳

PlayingScore > src > ap

```
1   <ion-header>
2     <ion-toolbar color="primary">
3       <ion-buttons start>
4         <ion-button href="#" icon="arrow-back" shape="round" size="large" type="button">
5           <ion-icon name="arrow-back">
6         </ion-button>
7       </ion-buttons>
8     </ion-toolbar>
9   </ion-header>
10  <ion-content>
11    <ion-tabs>
12      <ion-tab>
13        <ion-tab-bar color="light" slot="top">
14          <ion-tab>
15            <ion-icon name="person">
16            <ion-label> Perfil </ion-label>
17          </ion-tab>
18        </ion-tab-bar>
19      </ion-tab>
20    </ion-tabs>
21  </ion-content>
```



-amigos.page.html > ...

" color="primary">

abs"></ion-back-button>

O mesmo faremos para a tela de  
Meus Favoritos

> r="light" slot="top">

art">

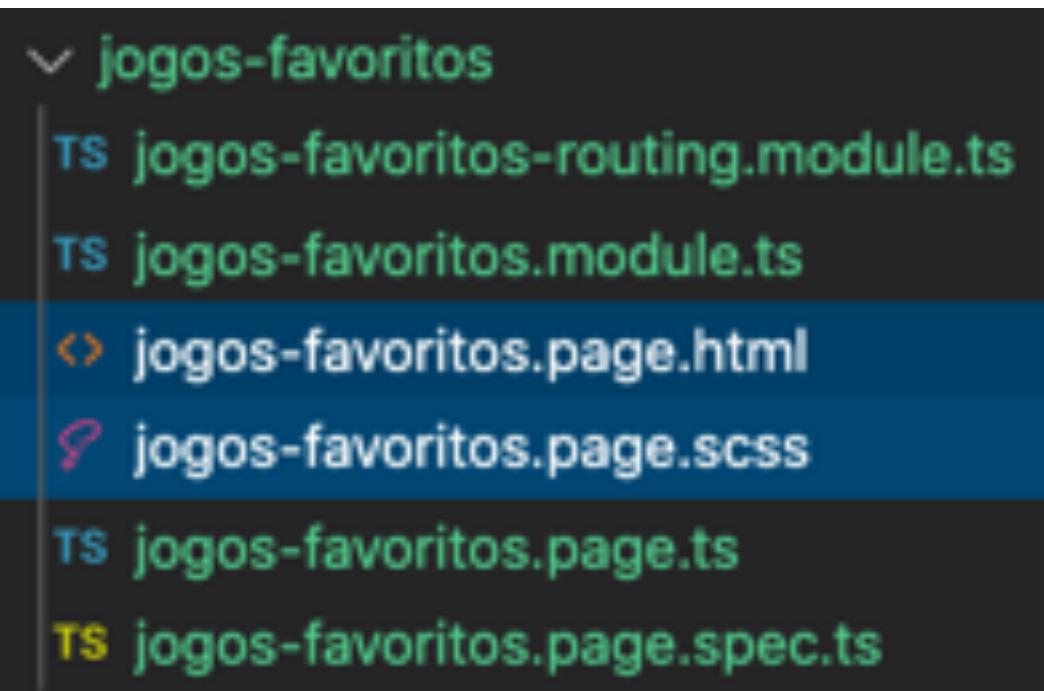
-icon>

on-label>

# Criando as Pages de Jogos

Com estas novas páginas criadas, vamos configurar a navegação de rotas

- Até aqui, ok, conseguimos fazer a navegação para estar telas
- Mas, vamos projetar elas

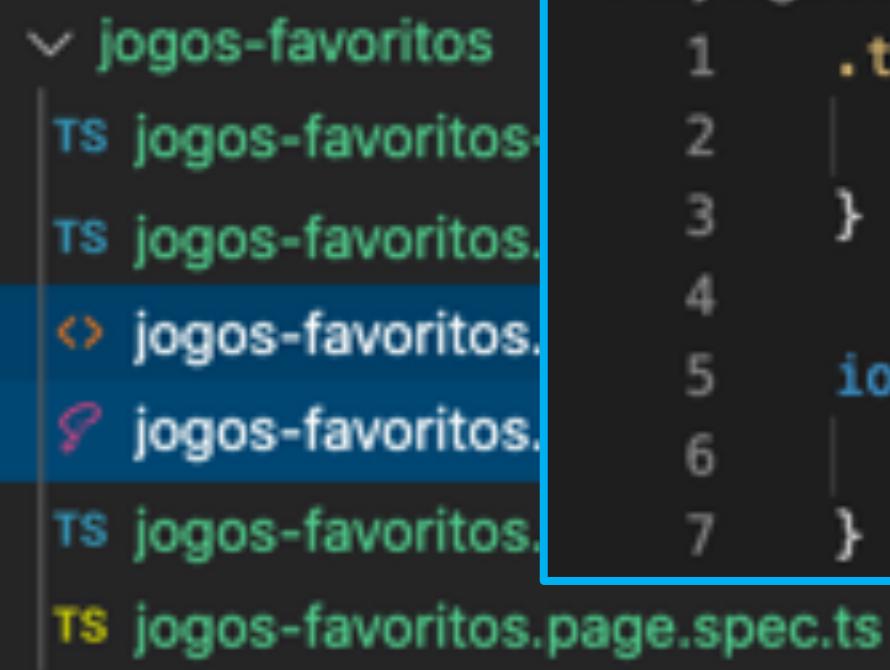


Agora vamos para o jogos-favoritos

# Criando as Pages de Jogos

Com estas novas páginas

- Até aqui, ok, conseguimos
- Mas, vamos projetar elas



A screenshot of a file explorer interface. On the left, there is a sidebar with several items under a category named "jogos-favoritos". The items listed are: "jogos-favoritos", "jogos-favoritos", "jogos-favoritos", "jogos-favoritos", "jogos-favoritos", "jogos-favoritos", and "jogos-favoritos.page.spec.ts". The "jogos-favoritos.page.spec.ts" file is highlighted with a blue background.

ξ jogos-favoritos.page.scss X

PlayingScore > src > app > jogos-

```
1   .tab-bar {  
2     height: 30px;  
3   }  
4  
5   ion-icon {  
6     font-size: 12px;  
7   }
```

as

jogos-favoritos

# Criando as Pages de Jogos

```
<> jogos-favoritos.page.html <x>

PlayingScore > src > app > jogos-favoritos > <> jogos-favoritos.page.html > ...

1   <ion-header>
2     <ion-toolbar class="ion-text-center" color="primary">
3       <ion-buttons slot="start">
4         <ion-back-button defaultHref="tabs"></ion-back-button>
5         <ion-title>Jogos</ion-title>
6       </ion-buttons>
7     </ion-toolbar>
8   </ion-header>
9
10  <ion-content>
11    <ion-tabs>
12      <ion-tab-bar class="tab-bar" color="light" slot="top">
13        <ion-tab-button layout="icon-start">
14          <ion-icon name="star"></ion-icon>
15          <ion-label>Meus Favoritos</ion-label>
16        </ion-tab-button>
17      </ion-tab-bar>
18    </ion-tabs>
19  </ion-content>
```

# Criando as Pages de Jogos

↳ jogos-favoritos.page.html ↳

PlayingScore > src > ap

```
1  <ion-header>
2    <ion-toolbar color="primary">
3      <ion-buttons start>
4        <ion-button icon="star" slot="primary" text="Meus Favoritos" type="button" value="Meus Favoritos">
5      </ion-buttons>
6    </ion-toolbar>
7  </ion-header>
8
9
10 <ion-content>
11   <ion-tabs>
12     <ion-tab>
13       <ion-tab-bar color="light" slot="top">
14         <ion-tab-item icon="person" label="Perfil" selected="true" value="Perfil">
15           <ion-label>
16             <ion-icon name="person" type="ionicon" value="ionicon">
17           </ion-label>
18         </ion-tab-item>
19       </ion-tab-bar>
20     </ion-tab>
21   </ion-tabs>
22 </ion-content>
```



jogos-favoritos.page.html > ...

" color="primary">

tabs"></ion-back-button>

or="light" slot="top">

start">

icon>

on-label>

# Criando as Pages de Jogos

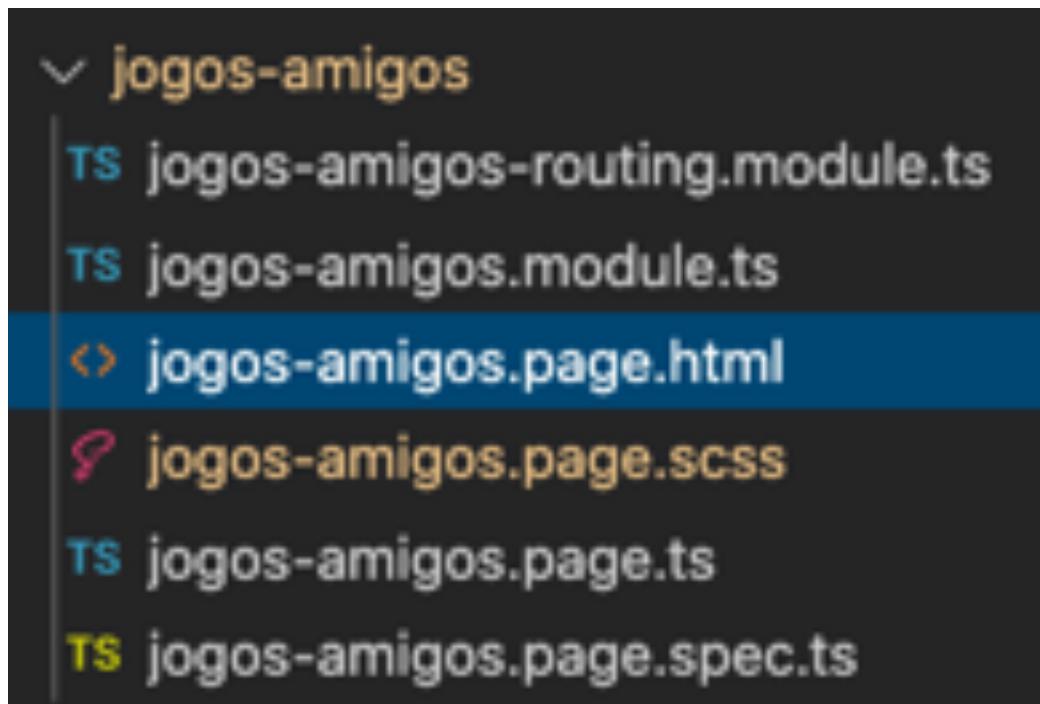


# Projetando as Telas de Jogos

# Projetando as Telas de Jogos

Vamos projetar as telas dos jogos

- Começaremos pela tela de Jogos de Amigos



# Projetando as Telas de Jogos

---

Vamos projetar as telas dos jogos

- Começaremos pela tela de Jogos de Amigos

# Vamos projetar as telas dos jogos

## Começaremos pela tela de Jogos de Amigos



## Vamos projetar as telas dos jogos

- Começaremos pela tela de Jogos de Amigos



### jogos-amigos.page.html

```
PlayingScore > src > app > jogos-amigos > jogos-amigos.page.html > ion-content
1   <ion-header>
2     <ion-toolbar class="ion-text-center" color="primary">
3       <ion-buttons slot="start">
4         <ion-back-button defaultHref="tabs"></ion-back-button>
5         <ion-title>Jogos</ion-title>
6       </ion-buttons>
7     </ion-toolbar>
8
9     <ion-toolbar class="tab-bar-sub" color="light" >
10       <ion-tab-button class="subtitle" layout="icon-start">
11         <ion-icon name="people"></ion-icon>
12         <ion-label>Jogos de amigos</ion-label>
13       </ion-tab-button>
14     </ion-toolbar>
15   </ion-header>
```

## Vamos projetar as telas dos jogos

- Começaremos pela tela de Jogos de Amigos



Mantemos a toolbar apresentando o botão de voltar e o título da página

### jogos-amigos.page.html

```
PlayingScore > src > app > jogos-amigos > jogos-amigos.page.html > ion-content
1  <ion-header>
2    <ion-toolbar class="ion-text-center" color="primary">
3      <ion-buttons slot="start">
4        <ion-back-button defaultHref="tabs"></ion-back-button>
5        <ion-title>Jogos</ion-title>
6      </ion-buttons>
7    </ion-toolbar>
8
9    <ion-toolbar class="tab-bar-sub" color="light" >
10      <ion-tab-button class="subtitle" layout="icon-start">
11        <ion-icon name="people"></ion-icon>
12        <ion-label>Jogos de amigos</ion-label>
13      </ion-tab-button>
14    </ion-toolbar>
15  </ion-header>
```

## Vamos projetar as telas dos jogos

- Começaremos pela tela de Jogos de Amigos



Passamos o ion-tab-button para dentro de um ion-toolbar do cabeçalho

### jogos-amigos.page.html

```
PlayingScore > src > app > jogos-amigos > jogos-amigos.page.html > ion-content
1   <ion-header>
2     <ion-toolbar class="ion-text-center" color="primary">
3       <ion-buttons slot="start">
4         <ion-back-button defaultHref="tabs"></ion-back-button>
5         <ion-title>Jogos</ion-title>
6       </ion-buttons>
7     </ion-toolbar>
8
9     <ion-toolbar class="tab-bar-sub" color="light" >
10       <ion-tab-button class="subtitle" layout="icon-start">
11         <ion-icon name="people"></ion-icon>
12         <ion-label>Jogos de amigos</ion-label>
13       </ion-tab-button>
14     </ion-toolbar>
15   </ion-header>
```

## Vamos projetar as telas dos jogos

- Começaremos pela tela de Jogos de Amigos

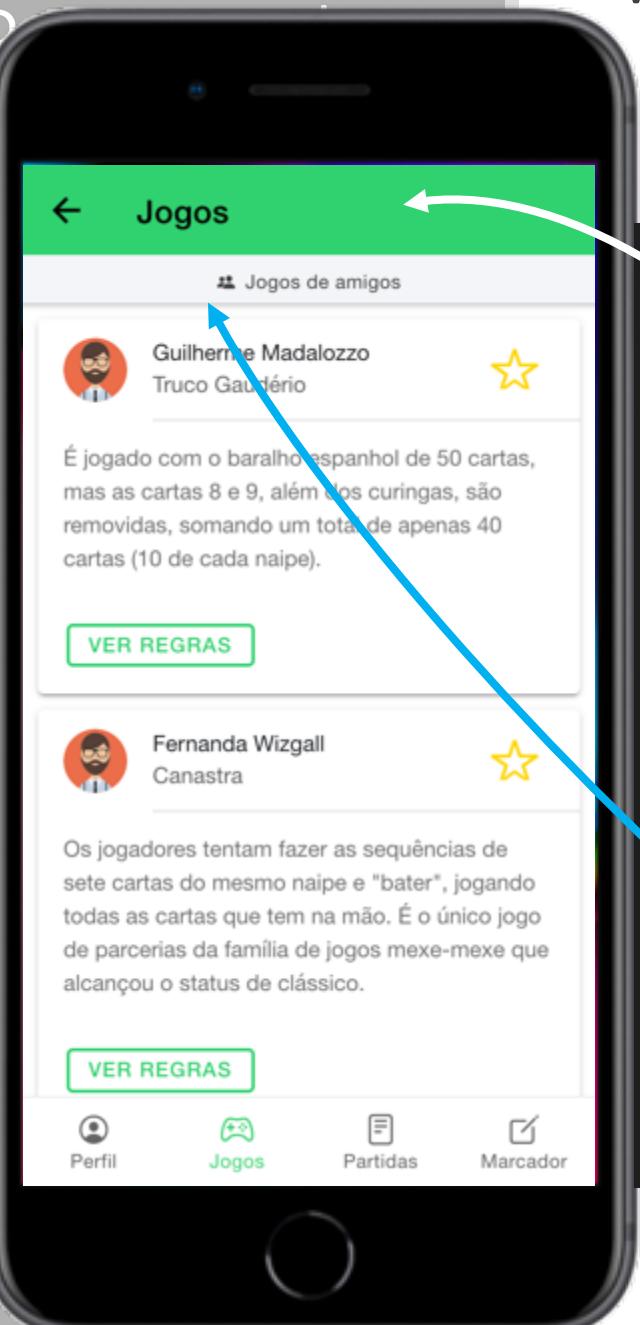


### jogos-amigos.page.html

```
PlayingScore > src > app > jogos-amigos > jogos-amigos.page.html > ion-content
1  <ion-header>
2    <ion-toolbar class="ion-text-center" color="primary">
3      <ion-buttons slot="start">
4        <ion-back-button defaultHref="tabs"></ion-back-button>
5        <ion-title>Jogos</ion-title>
6      </ion-buttons>
7    </ion-toolbar>
8
9    <ion-toolbar class="tab-bar-sub" color="light" >
10      <ion-tab-button class="subtitle" layout="icon-start">
11        <ion-icon name="people"></ion-icon>
12        <ion-label>Jogos de amigos</ion-label>
13      </ion-tab-button>
14    </ion-toolbar>
15  </ion-header>
```

## Vamos projetar as telas dos jogos

- Começaremos pela tela de Jogos de Amigos



```
jogos-amigos.page.html X

PlayingScore > src > app > jogos-amigos > jogos-amigos.page.html > ion-content

1 <ion-header>
2   <ion-toolbar class="ion-text-center" color="primary">
3     <ion-buttons slot="start">
4       <ion-back-button defaultHref="tabs"></ion-back-button>
5       <ion-title>Jogos</ion-title>
6     </ion-buttons>
7   </ion-toolbar>
8
9   <ion-toolbar class="tab-bar-sub" color="light" >
10    <ion-tab-button class="subtitle" layout="icon-start">
11      <ion-icon name="people"></ion-icon>
12      <ion-label>Jogos de amigos</ion-label>
13    </ion-tab-button>
14  </ion-toolbar>
15 </ion-header>
```

# Vamos projetar as telas dos jogos

## Começaremos pela tela de Jogos de Amigos





Vamos projetar as telas dos jogos  
Começaremos pela tela de Jogos de Amigos

Vamos  
ao conteúdo  
da página

Para projetarmos o conteúdo da  
página, vamos utilizar o ion-card

<https://ionicframework.com/docs/api/card>

## ion-card

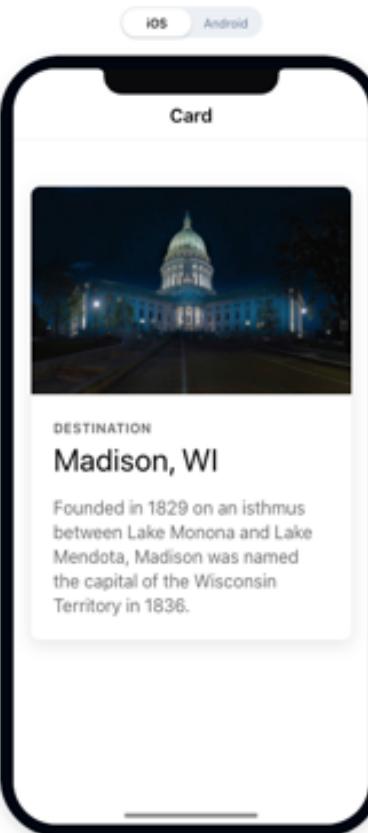
CONTENTS

Usage  
Properties  
CSS Shadow Parts  
CSS Custom Properties

Cards are a standard piece of UI that serves as an entry point to more detailed information. A card can be a single component, but is often made up of some header, title, subtitle, and content. `ion-card` is broken up into several sub-components to reflect this. Please see `ion-card-content`, `ion-card-header`, `ion-card-title`, `ion-card-subtitle`.

### Usage

ANGULAR JAVASCRIPT REACT STENCIL VUE [Copy](#)  
`<ion-card>`  
  `<ion-card-header>`  
    `<ion-card-subtitle>Card Subtitle</ion-card-subtitle>`  
    `<ion-card-title>Card Title</ion-card-title>`  
  `</ion-card-header>`  
  
  `<ion-card-content>`



# Vamos projetar as telas dos jogos

- Começaremos pela tela de Jogos de Amigos



jogos-amigos.page.html X

PlayingScore > src > app > jogos-amigos > jogos-amigos.page.html > ion-content > ion-card > ion-item > ion-

```
16<ion-content>
17  <ion-card>
18    <ion-item>
19      <ion-avatar slot="start">
20        
21      </ion-avatar>
22
23      <ion-label class="ion-text-wrap">
24        <ion-text color="dark"><h3>Guilherme Madalozzo</h3></ion-text>
25        <p>Truco Gaudério</p>
26      </ion-label>
27
28      <ion-button color="tertiary" fill="clear">
29        <ion-icon size="large" name="star-outline" color="tertiary"></ion-icon>
30      </ion-button>
31    </ion-item>
32
33    <ion-card-content>
34      É jogado com o baralho espanhol de 50 cartas, mas as cartas 8 e 9, além dos curingas,
35      são removidas, somando um total de apenas 40 cartas (10 de cada naipe).
36    </ion-card-content>
37
38    <ion-card-content>
39      <ion-button size="small" fill="outline" slot="end">Ver Regras</ion-button>
40    </ion-card-content>
41
42  </ion-card>
```

## Vamos projetar as telas dos jogos

- Começaremos pela tela de Jogos de Amigos



```
16
17 <ion-content>
18   <ion-card>
19     <ion-item>
20       <ion-avatar slot="start">
21         
22       </ion-avatar>
23
24       <ion-label class="ion-text-wrap">
25         <ion-text color="dark"><h3>Guilherme Madalozzo</h3></ion-text>
26         <p>Truco Gaudério</p>
27       </ion-label>
28
29       <ion-button color="tertiary" fill="clear">
30         <ion-icon size="large" name="star-outline" color="tertiary"></ion-icon>
31       </ion-button>
32     </ion-item>
33
34   <ion-card-content>
35     É jogado com o baralho espanhol de 50 cartas, mas as cartas 8 e 9, além dos curingas, são removidas, somando um total de apenas 40 cartas (10 de cada naipe).
36   </ion-card-content>
37
38   <ion-card-content>
39     <ion-button size="small" fill="outline" slot="end">Ver Regras</ion-button>
40   </ion-card-content>
41
42 </ion-card>
```

Para projetarmos o conteúdo da página, vamos utilizar o ion-card

## Vamos projetar as telas dos jogos

- Começaremos pela tela de Jogos de Amigos



jogos-amigos.page.html X

PlayingScore > src > app > jogos-amigos > jogos-amigos.page.html > ion-content

```
16<ion-content>
17  <ion-card>
18    <ion-item>
19      <ion-avatar slot="start">
20        
21      </ion-avatar>
22
23      <ion-label class="ion-text-wrap">
24        <ion-text color="dark"><h3>Guilherme Madalozzo</h3></ion-text>
25        <p>Truco Gaudério</p>
26      </ion-label>
27
28      <ion-button color="tertiary" fill="clear">
29        <ion-icon size="large" name="star-outline" color="tertiary"></ion-icon>
30      </ion-button>
31    </ion-item>
32
33    <ion-card-content>
34      É jogado com o baralho espanhol de 50 cartas, mas as cartas 8 e 9, além dos curingas, são removidas, somando um total de apenas 40 cartas (10 de cada naipe).
35
36    <ion-card-content>
37      <ion-button size="small" fill="outline" slot="end">Ver Regras</ion-button>
38    </ion-card-content>
39
40  </ion-card>
```

Montamos o ion-item que apresenta a foto e o nome do amigo, nome do jogo e opção de tornar favorito

## Vamos projetar as telas dos jogos

- Começaremos pela tela de Jogos de Amigos



```
 16
 17 <ion-content>
 18   <ion-card>
 19     <ion-item>
 20       <ion-avatar slot="start">
 21         
 22       </ion-avatar>
 23
 24       <ion-label class="ion-text-wrap">
 25         <ion-text color="dark"><h3>Guilherme Madalozzo</h3></ion-text>
 26         <p>Truco Gaudério</p>
 27       </ion-label>
 28
 29       <ion-button color="tertiary" fill="clear">
 30         <ion-icon size="large" name="star-outline" color="tertiary"></ion-icon>
 31       </ion-button>
 32     </ion-item>
 33
 34     <ion-card-content>
 35       É jogado com o baralho espanhol de 50 cartas, mas as cartas 8 e 9, além dos curingas, são removidas, somando um total de apenas 40 cartas (10 de cada naipe).
 36     </ion-card-content>
 37
 38     <ion-card-content>
 39       <ion-button size="small" fill="outline" slot="end">Ver Regras</ion-button>
 40     </ion-card-content>
 41
 42   </ion-card>
```

Foto do amigo

## Vamos projetar as telas dos jogos

- Começaremos pela tela de Jogos de Amigos



↳

jogos-amigos.page.html X

PlayingScore > src > app > jogos-amigos > jogos-amigos.page.html > ion-content

```
16<ion-content>
17  <ion-card>
18    <ion-item>
19      <ion-avatar slot="start">
20        
21      </ion-avatar>
22
23
24      <ion-label class="ion-text-wrap">
25        <ion-text color="dark"><h3>Guilherme Madalozzo</h3></ion-text>
26        <p>Truco Gaudério</p>
27      </ion-label>
28
29      <ion-button color="tertiary" fill="clear">
30        <ion-icon size="large" name="star-outline" color="tertiary"></ion-icon>
31      </ion-button>
32    </ion-item>
33
34    <ion-card-content>
35      É jogado com o baralho espanhol de 50 cartas, mas as cartas 8 e 9, além dos curingas, são removidas, somando um total de apenas 40 cartas (10 de cada naipe).
36
37  </ion-card-content>
38
39  <ion-card-content>
40    <ion-button size="small" fill="outline" slot="end">Ver Regras</ion-button>
41  </ion-card-content>
42</ion-card>
```

Nome do amigo e do jogo

## Vamos projetar as telas dos jogos

- Começaremos pela tela de Jogos de Amigos



jogos-amigos.page.html X  
PlayingScore > src > app > jogos-amigos > jogos-amigos.page.html > ion-content

```
16<ion-content>
17  <ion-card>
18    <ion-item>
19      <ion-avatar slot="start">
20        
21      </ion-avatar>
22
23      <ion-label class="ion-text-wrap">
24        <ion-text color="dark"><h3>Guilherme Madalozzo</h3></ion-text>
25        <p>Truco Gaudério</p>
26      </ion-label>
27
28
29      <ion-button color="tertiary" fill="clear">
30        <ion-icon size="large" name="star-outline" color="tertiary"></ion-icon>
31      </ion-button>
32    </ion-item>
33
34    <ion-card-content>
35      É jogado com o baralho espanhol de 50 cartas, mas as cartas 8 e 9, além dos curingas,
36      são removidas, somando um total de apenas 40 cartas (10 de cada naipe).
37    </ion-card-content>
38
39    <ion-card-content>
40      <ion-button size="small" fill="outline" slot="end">Ver Regras</ion-button>
41    </ion-card-content>
42  </ion-card>
```

Botão para tornar o jogo para favorito

## Vamos projetar as telas dos jogos

- Começaremos pela tela de Jogos de Amigos



```
PlayingScore > src > app > jogos-amigos > jogos-amigos.page.html > ion-content

16
17  <ion-content>
18    <ion-card>
19      <ion-item>
20        <ion-avatar slot="start">
21          
22        </ion-avatar>
23
24        <ion-label class="ion-text-wrap">
25          <ion-text color="dark"><h3>Guilherme Madalozzo</h3></ion-text>
26          <p>Truco Gaudério</p>
27        </ion-label>
28
29        <ion-button color="tertiary" fill="clear">
30          <ion-icon size="large" name="star-outline" color="tertiary"></ion-icon>
31        </ion-button>
32      </ion-item>
33
34      <ion-card-content>
35        É jogado com o baralho espanhol de 50 cartas, mas as cartas 8 e 9, além dos curingas,
36        são removidas, somando um total de apenas 40 cartas (10 de cada naipe).
37      </ion-card-content>
38
39      <ion-card-content>
40        <ion-button size="small" fill="outline" slot="end">Ver Regras</ion-button>
41      </ion-card-content>
42    </ion-card>
```

Conteúdo do card com a descrição do jogo

## Vamos projetar as telas dos jogos

- Começaremos pela tela de Jogos de Amigos



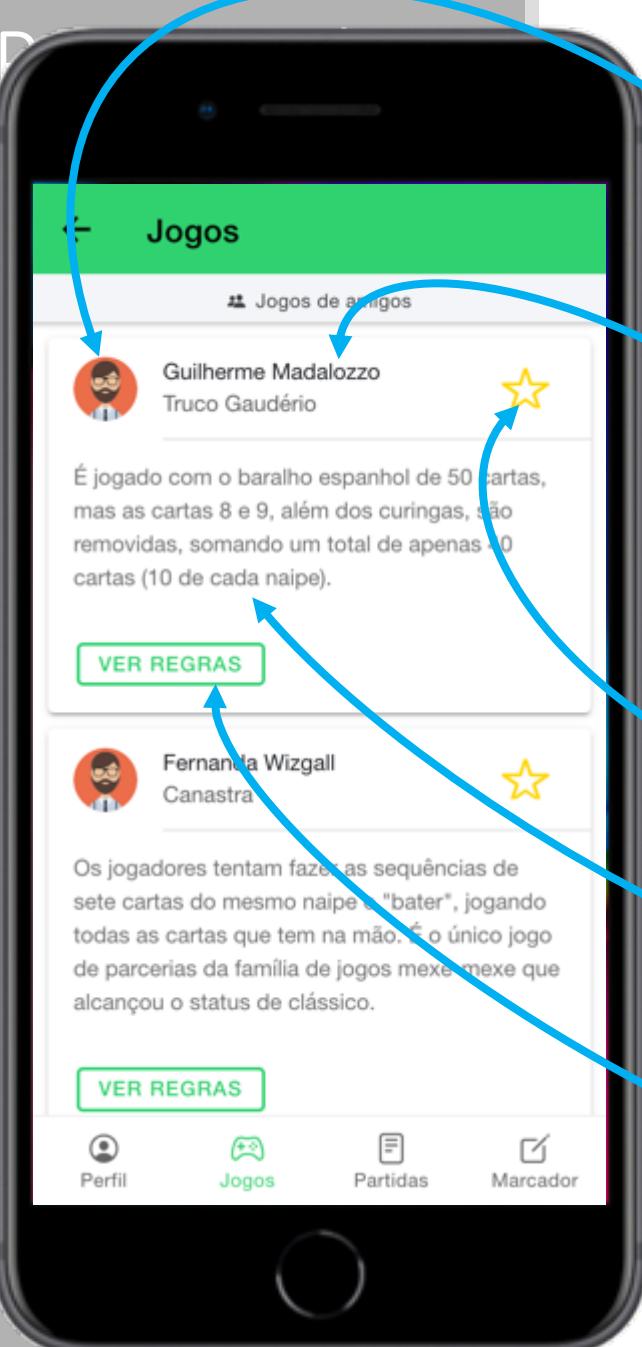
jogos-amigos.page.html X  
PlayingScore > src > app > jogos-amigos > jogos-amigos.page.html > ion-content

```
16<ion-content>
17  <ion-card>
18    <ion-item>
19      <ion-avatar slot="start">
20        
21      </ion-avatar>
22
23      <ion-label class="ion-text-wrap">
24        <ion-text color="dark"><h3>Guilherme Madalozzo</h3></ion-text>
25        <p>Truco Gaudério</p>
26      </ion-label>
27
28      <ion-button color="tertiary" fill="clear">
29        <ion-icon size="large" name="star-outline" color="tertiary"></ion-icon>
30      </ion-button>
31    </ion-item>
32
33    <ion-card-content>
34      É jogado com o baralho espanhol de 50 cartas, mas as cartas 8 e 9, além dos curingas, são removidas, somando um total de apenas 40 cartas (10 de cada naipe).
35
36    </ion-card-content>
37
38    <ion-card-content>
39      <ion-button size="small" fill="outline" slot="end">Ver Regras</ion-button>
40    </ion-card-content>
41
42  </ion-card>
```

Botão para ter acesso a tela de regras

# Vamos projetar as telas dos jogos

- Começaremos pela tela de Jogos de Amigos



```
jogos-amigos.page.html X  
PlayingScore > src > app > jogos-amigos > jogos-amigos.page.html > ion-content > ion-card > ion-item > ion-content  
16  
17 <ion-content>  
18 <ion-card>  
19   <ion-item>  
20     <ion-avatar slot="start">  
21         
22     </ion-avatar>  
23  
24     <ion-label class="ion-text-wrap">  
25       <ion-text color="dark"><h3>Guilherme Madalozzo</h3></ion-text>  
26       <p>Truco Gaudério</p>  
27     </ion-label>  
28  
29     <ion-button color="tertiary" fill="clear">  
30       <ion-icon size="large" name="star-outline" color="tertiary"></ion-icon>  
31     </ion-button>  
32   </ion-item>  
33  
34   <ion-card-content>  
35     É jogado com o baralho espanhol de 50 cartas, mas as cartas 8 e 9, além dos curingas,  
36     são removidas, somando um total de apenas 40 cartas (10 de cada naipe).  
37   </ion-card-content>  
38  
39   <ion-card-content>  
40     <ion-button size="small" fill="outline" slot="end">Ver Regras</ion-button>  
41   </ion-card-content>  
42 </ion-card>
```

## Vamos projetar as telas dos jogos

- Para termos mais jogos, foram criados outros dois cards



## Vamos projetar as telas dos jogos

- Para termos mais jogos, foram criados outros dois cards



```
44 <ion-card>
45   <ion-item>
46     <ion-avatar slot="start">
47       
48     </ion-avatar>
49
50     <ion-label class="ion-text-wrap">
51       <ion-text color="dark"><b>Fernanda Wizgall</b></ion-text>
52       <p>Canastra</p>
53     </ion-label>
54
55     <ion-button color="tertiary" fill="clear">
56       <ion-icon size="large" name="star-outline" color="tertiary"></ion-icon>
57     </ion-button>
58   </ion-item>
59
60   <ion-card-content>
61     Os jogadores tentam fazer as sequências de sete cartas do mesmo naipe e "bater",
62     jogando todas as cartas que tem na mão. É o único jogo de parcerias da
63     família de jogos mexe-mexe que alcançou o status de clássico.
64   </ion-card-content>
65
66   <ion-card-content>
67     <ion-button size="small" fill="outline" slot="end">Ver Regras</ion-button>
68   </ion-card-content>
69 </ion-card>
```

## Vamos projetar as telas dos jogos

- Para termos mais jogos, foram criados outros dois cards



```
71 <ion-card>
72   <ion-item>
73     <ion-avatar state="start">
74       
75     </ion-avatar>
76
77     <ion-label classe="ion-text-wrap">
78       <ion-text color="dark"><h3>Guilherme Madalozzo</h3></ion-text>
79       <p>Basquete 21</p>
80     </ion-label>
81
82     <ion-button color="tertiary" fill="clear">
83       <ion-icon size="large" name="star-outline" color="tertiary"></ion-icon>
84     </ion-button>
85   </ion-item>
86
87   <ion-card-content>
88     O jogo é jogado com qualquer número de jogadores em uma meia quadra,
89     mas normalmente quando jogadores insuficientes estão disponíveis para,
90     pelo menos, jogar três-em-três. Vinte e um é um jogo individual que não
91     utilizam jogo da equipe.
92   </ion-card-content>
93
94   <ion-card-content>
95     <ion-button size="small" fill="outline" state="end">Ver Regras</ion-button>
96   </ion-card-content>
97 </ion-card>
98 </ion-content>
```

## Vamos projetar as telas dos jogos

- Para concluir, precisamos fazer alterações no SCSS



## Vamos projetar as telas dos jogos

- Para concluir, precisamos fazer alterações no SCSS



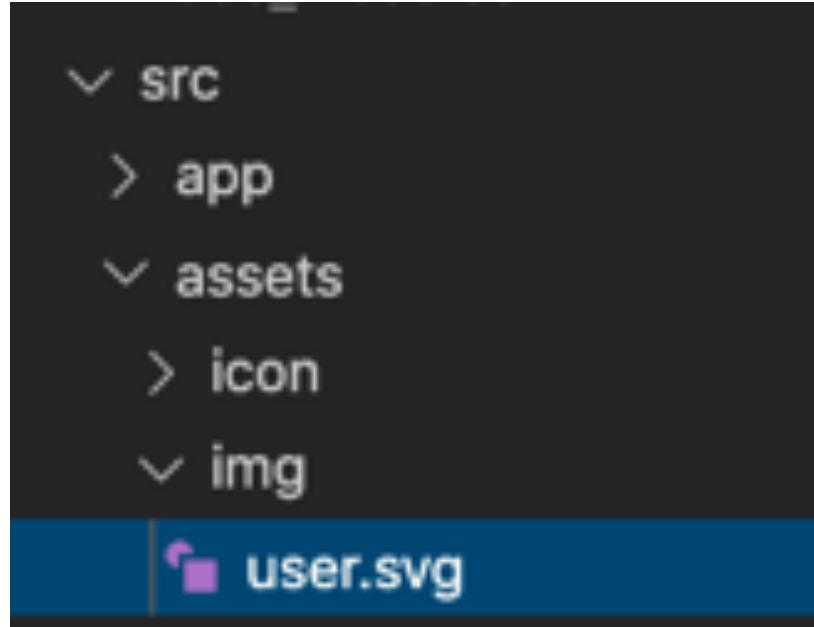
♀ jogos-amigos.page.scss ×

PlayingScore > src > app > jogos-amigos > ♀ jogos-amigos

```
1 .tab-bar-sub {  
2   height: 30px;  
3   position: relative;  
4   background-color: ■#f4f5f8 !important;  
5   text-align: center !important;  
6 }  
7  
8 .subtitle {  
9   position: relative;  
10  top: -11px !important;  
11  font-size: 12px;  
12 }  
13  
14 ion-icon {  
15   font-size: 12px;  
16 }
```

## Vamos projetar as telas dos jogos

- Para concluir, precisamos fazer alterações no SCSS



Vamos criar uma pasta chamada img dentro de src/assets e adicionar uma imagem (png/svg) de usuário

<https://github.com/guimadalozzo/PlayingScore/tree/master/src/assets/img>

# Projetando as Telas de Jogos

---

Vamos projetar as telas dos jogos

- Agora, vamos projetar a tela de Jogos Favoritos

# Projetando as Telas de Jogos

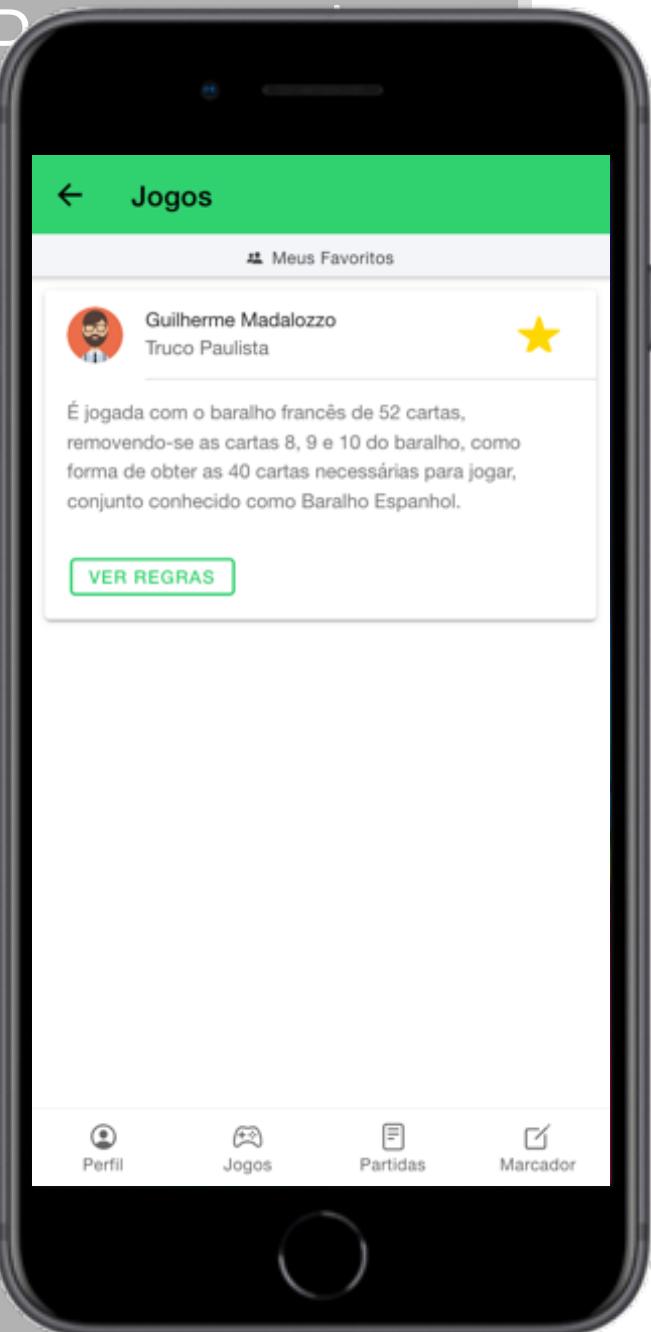
Vamos projetar as telas dos jogos

- Agora, vamos projetar a tela de Jogos Favoritos

```
⌄ jogos-favoritos
  TS jogos-favoritos-routing.module.ts
  TS jogos-favoritos.module.ts
  ⌄ jogos-favoritos.page.html
  SCSS jogos-favoritos.page.scss
  TS jogos-favoritos.page.ts
  TS jogos-favoritos.page.spec.ts
```

## Vamos projetar as telas dos jogos

- Agora, vamos projetar a tela de Jogos Favoritos



## Vamos projetar as telas dos jogos

- Agora, vamos projetar a tela de Jogos Favoritos



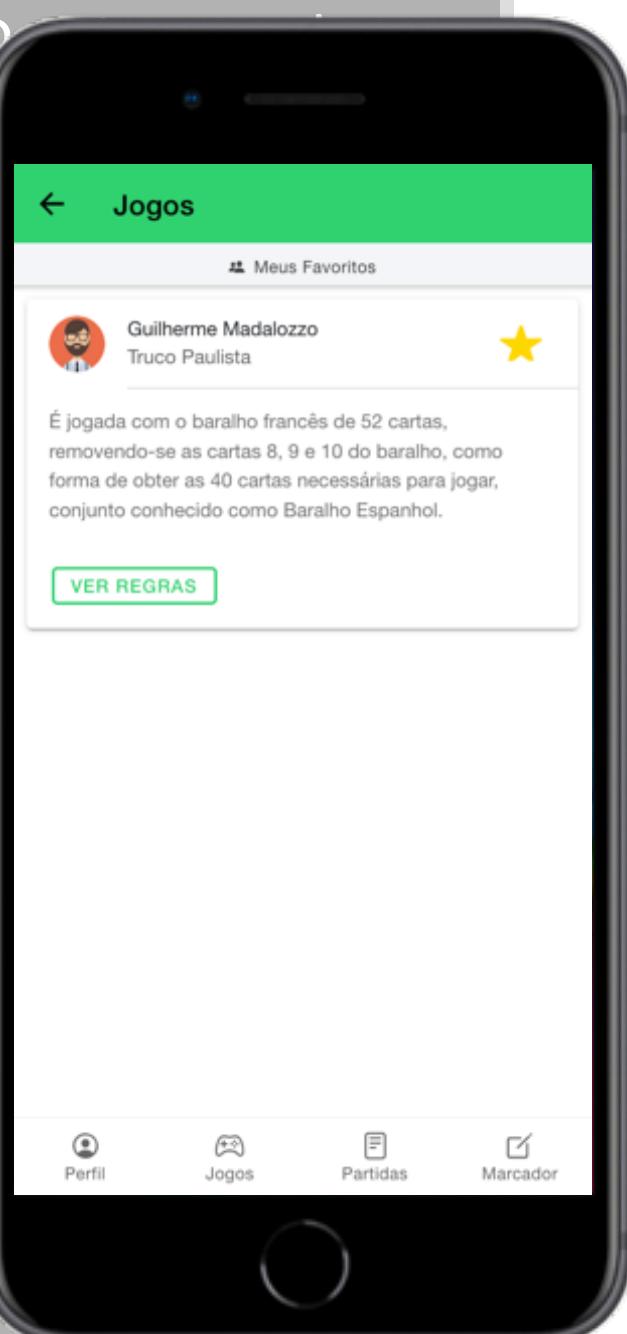
### jogos-favoritos.page.html

```
PlayingScore > src > app > jogos-favoritos > jogos-favoritos.page.html > ion-content
1  <ion-header>
2      <ion-toolbar class="ion-text-center" color="primary">
3          <ion-buttons slot="start">
4              <ion-back-button defaultHref="tabs"></ion-back-button>
5          <ion-title>Jogos</ion-title>
6      </ion-buttons>
7  </ion-toolbar>
8
9  <ion-toolbar class="tab-bar-sub" color="light" >
10     <ion-tab-button class="subtitle" layout="icon-start">
11         <ion-icon name="people"></ion-icon>
12         <ion-label>Meus Favoritos</ion-label>
13     </ion-tab-button>
14 </ion-toolbar>
15 </ion-header>
```

Esta tela será bastante semelhante com a de Jogos de Amigos

## Vamos projetar as telas dos jogos

- Agora, vamos projetar a tela de Jogos Favoritos



O cabeçalho será mantido com o título e o botão de voltar

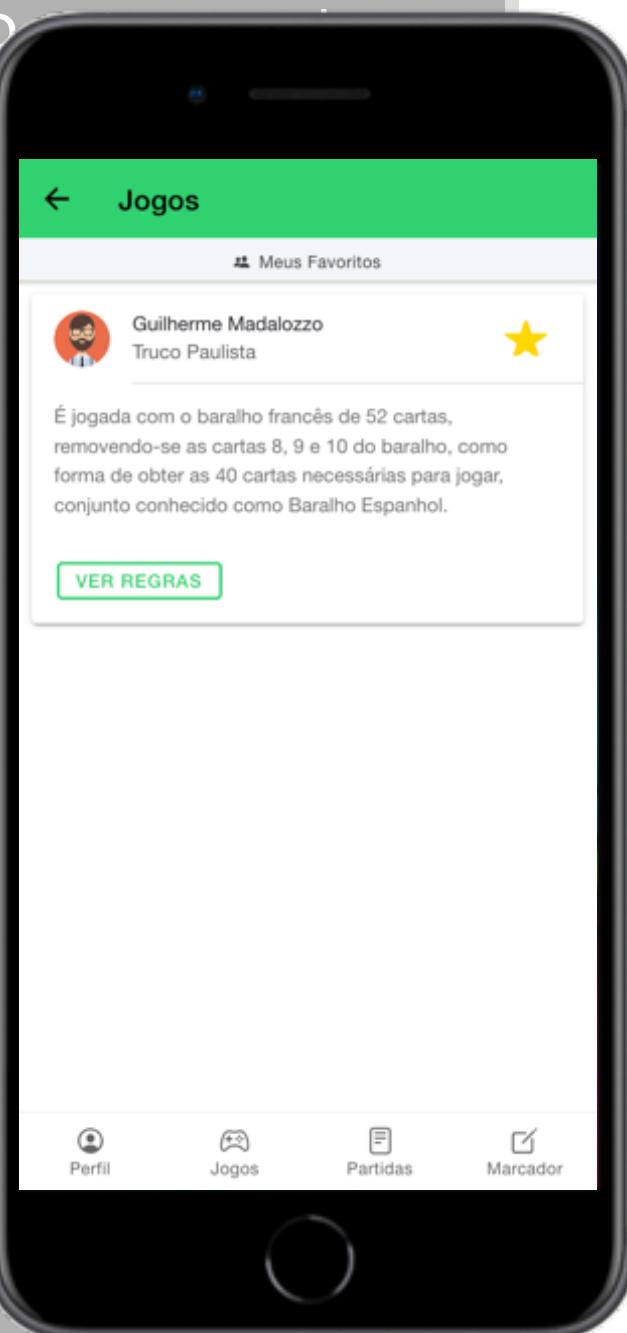
### ◦ jogos-favoritos.page.html ×

PlayingScore > src > app > jogos-favoritos > ◦ jogos-favoritos.page.html > ⚡ ion-content

```
1  <ion-header>
2      <ion-toolbar class="ion-text-center" color="primary">
3          <ion-buttons slot="start">
4              <ion-back-button defaultHref="tabs"></ion-back-button>
5          <ion-title>Jogos</ion-title>
6      </ion-buttons>
7  </ion-toolbar>
8
9      <ion-toolbar class="tab-bar-sub" color="light" >
10         <ion-tab-button class="subtitle" layout="icon-start">
11             <ion-icon name="people"></ion-icon>
12             <ion-label>Meus Favoritos</ion-label>
13         </ion-tab-button>
14     </ion-toolbar>
15 </ion-header>
```

## Vamos projetar as telas dos jogos

- Agora, vamos projetar a tela de Jogos Favoritos



### ◦ jogos-favoritos.page.html ×

PlayingScore > src > app > jogos-favoritos > ◦ jogos-favoritos.page.html > ⚡ ion-content

```
1  <ion-header>
2      <ion-toolbar class="ion-text-center" color="primary">
3          <ion-buttons slot="start">
4              <ion-back-button defaultHref="tabs"></ion-back-button>
5          <ion-title>Jogos</ion-title>
6      </ion-buttons>
7  </ion-toolbar>
8
9  <ion-toolbar class="tab-bar-sub" color="light" >
10     <ion-tab-button class="subtitle" layout="icon-start">
11         <ion-icon name="people"></ion-icon>
12         <ion-label>Meus Favoritos</ion-label>
13     </ion-tab-button>
14 </ion-toolbar>
15 </ion-header>
```

E agora teremos a adição do  
ion-tab-button

## Vamos projetar as telas dos jogos

- Agora, vamos projetar a tela de Jogos Favoritos

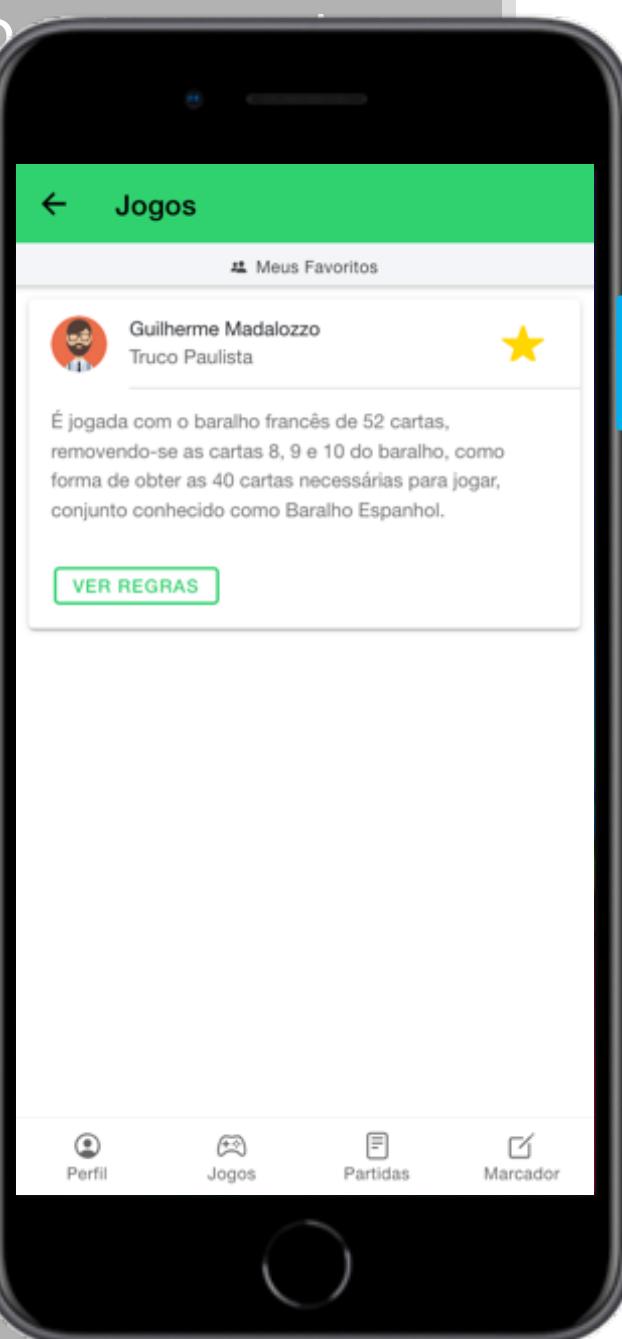


O conteúdo desta tela, para ilustrar,  
terá apenas 1 card de favoritos



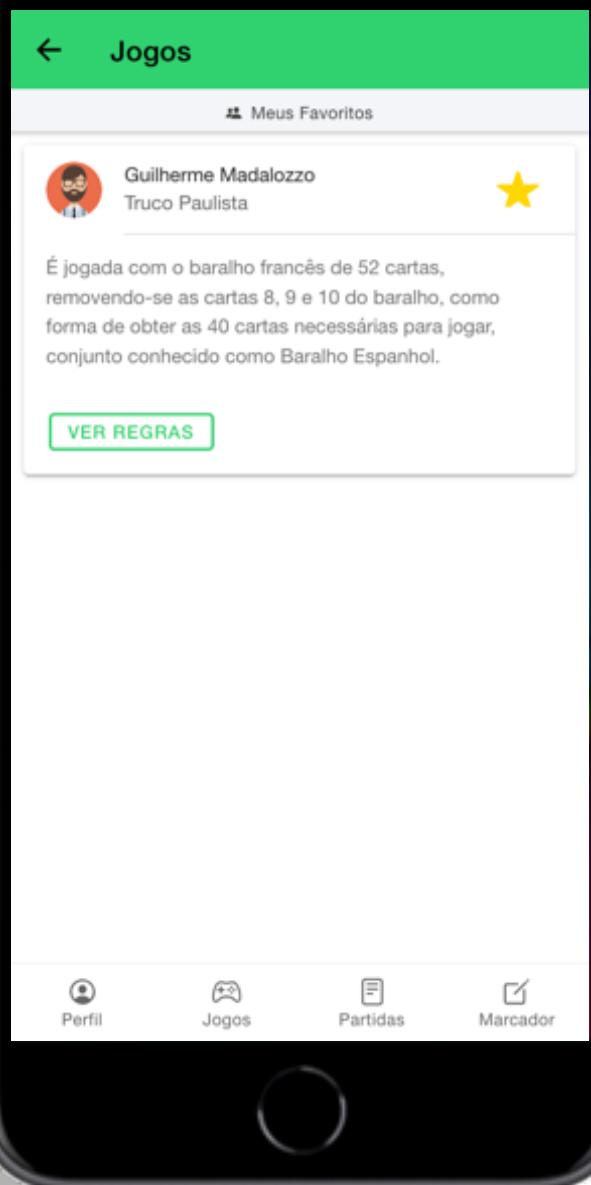
```
① jogos-favoritos.page.html ×
PlayingScore > src > app > jogos-favoritos > ② jogos-favoritos.page.html > ③ ion-header > ④ ion-toolbar.tab-bar-s
16
17  <ion-content>
18
19    <ion-card>
20      <ion-item>
21        <ion-avatar slot="start">
22          
23        </ion-avatar>
24
25        <ion-label class="ion-text-wrap">
26          <ion-text color="dark"><h3>Guilherme Madalozzo</h3></ion-text>
27          <p>Truco Paulista</p>
28        </ion-label>
29
30        <ion-button color="tertiary" fill="clear">
31          <ion-icon size="large" name="star" color="tertiary"></ion-icon>
32        </ion-button>
33      </ion-item>
34
35      <ion-card-content>
36        É jogada com o baralho francês de 52 cartas, removendo-se as cartas
37        8, 9 e 10 do baralho, como forma de obter as 40 cartas necessárias para jogar,
38        conjunto conhecido como Baralho Espanhol.
39      </ion-card-content>
40
41      <ion-card-content>
42        <ion-button size="small" fill="outline" slot="end">Ver Regras</ion-button>
43      </ion-card-content>
44    </ion-card>
45
46  </ion-content>
```

O card é composto pelo avatar da foto "dono" do registro de jogo



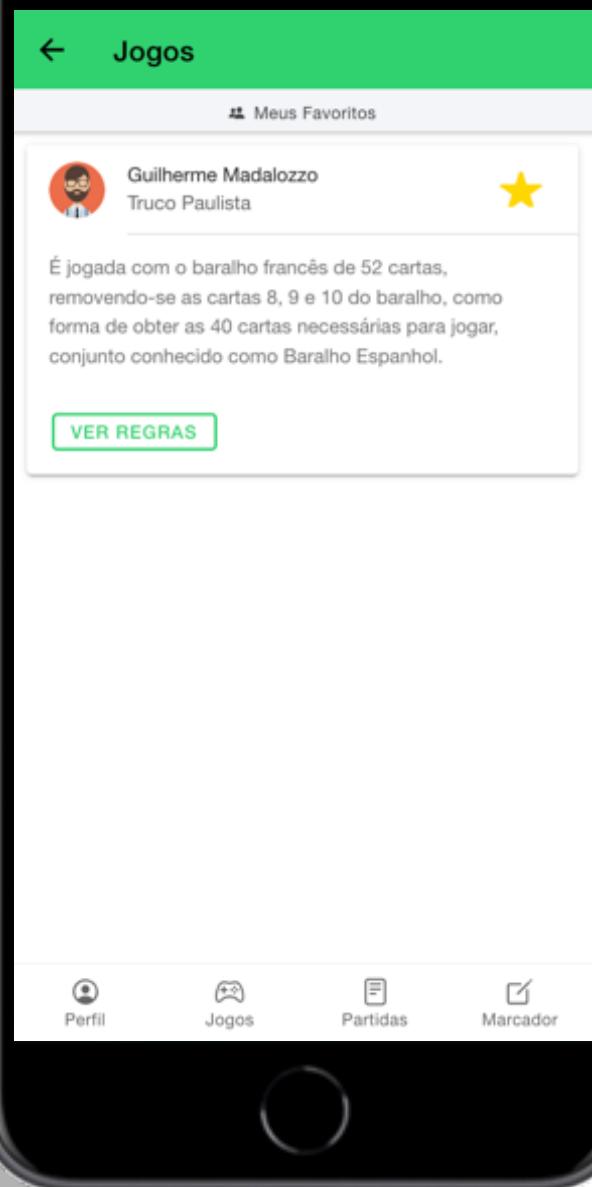
```
① <ion-content>
②   <ion-card>
③     <ion-item>
④       <ion-avatar slot="start">
⑤         
⑥       </ion-avatar>
⑦
⑧       <ion-label class="ion-text-wrap">
⑨         <ion-text color="dark"><h3>Guilherme Madalozzo</h3></ion-text>
⑩         <p>Truco Paulista</p>
⑪       </ion-label>
⑫
⑬       <ion-button color="tertiary" fill="clear">
⑭         <ion-icon size="large" name="star" color="tertiary"></ion-icon>
⑮       </ion-button>
⑯     </ion-item>
⑰
⑱     <ion-card-content>
⑲       É jogada com o baralho francês de 52 cartas, removendo-se as cartas
⑳       8, 9 e 10 do baralho, como forma de obter as 40 cartas necessárias para jogar,
⑳       conjunto conhecido como Baralho Espanhol.
⑳     </ion-card-content>
⑳
⑳     <ion-card-content>
⑳       <ion-button size="small" fill="outline" slot="end">Ver Regras</ion-button>
⑳     </ion-card-content>
⑳   </ion-card>
⑳
⑳ </ion-content>
```

Teremos o nome do amigo e do jogo



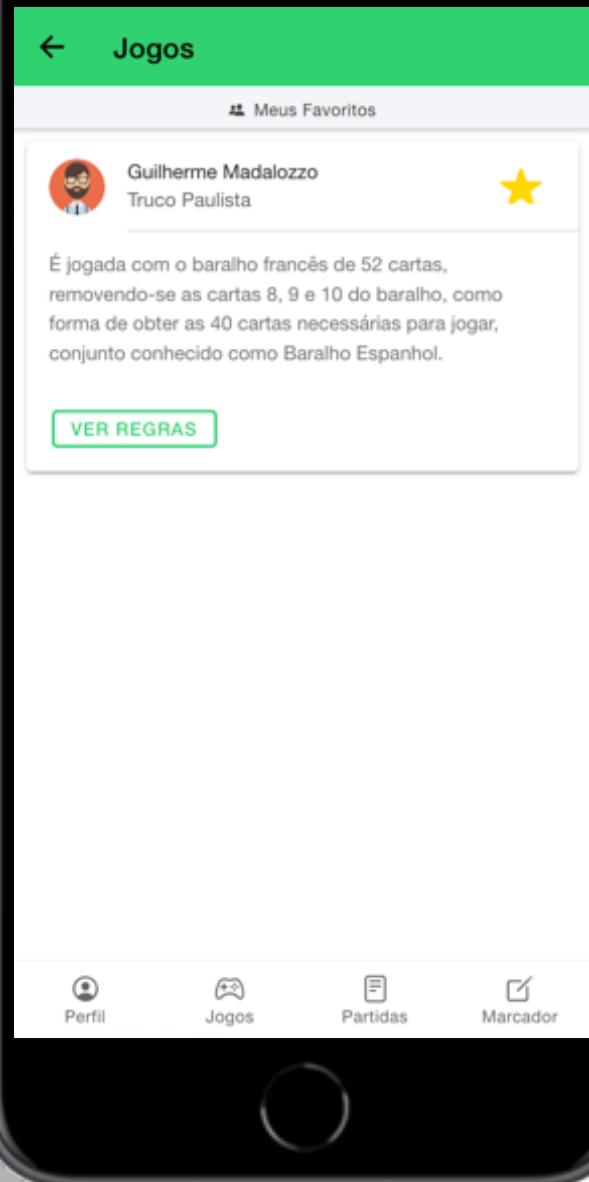
```
> jogos-favoritos.page.html <  
PlayingScore > src > app > jogos-favoritos > > jogos-favoritos.page.html >   
16  
17  <ion-content>  
18  
19    <ion-card>  
20      <ion-item>  
21        <ion-avatar slot="start">  
22            
23        </ion-avatar>  
24  
25        <ion-label class="ion-text-wrap">  
26          <ion-text color="dark"><h3>Guilherme Madalozzo</h3></ion-text>  
27          <p>Truco Paulista</p>  
28        </ion-label>  
29  
30        <ion-button color="tertiary" fill="clear">  
31          <ion-icon size="large" name="star" color="tertiary"></ion-icon>  
32        </ion-button>  
33      </ion-item>  
34  
35      <ion-card-content>  
36        É jogada com o baralho francês de 52 cartas, removendo-se as cartas  
37        8, 9 e 10 do baralho, como forma de obter as 40 cartas necessárias para jogar,  
38        conjunto conhecido como Baralho Espanhol.  
39      </ion-card-content>  
40  
41      <ion-card-content>  
42        <ion-button size="small" fill="outline" slot="end">Ver Regras</ion-button>  
43      </ion-card-content>  
44    </ion-card>  
45  
46  </ion-content>
```

O botão de estrela que pode desfavoritar o jogo

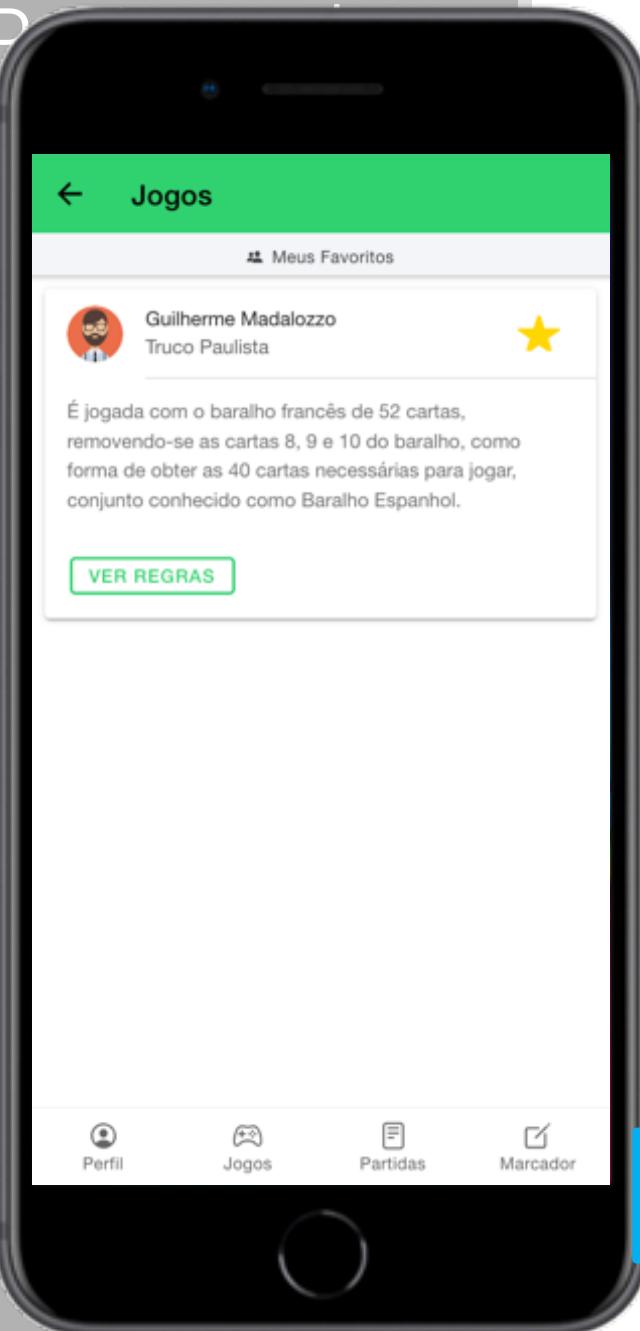


```
① <ion-content>
②   <ion-card>
③     <ion-item>
④       <ion-avatar slot="start">
⑤         
⑥       </ion-avatar>
⑦
⑧       <ion-label class="ion-text-wrap">
⑨         <ion-text color="dark"><h3>Guilherme Madalozzo</h3></ion-text>
⑩         <p>Truco Paulista</p>
⑪       </ion-label>
⑫
⑬       <ion-button color="tertiary" fill="clear">
⑭         <ion-icon size="large" name="star" color="tertiary"></ion-icon>
⑮       </ion-button>
⑯     </ion-item>
⑰
⑱     <ion-card-content>
⑲       É jogada com o baralho francês de 52 cartas, removendo-se as cartas
⑳       8, 9 e 10 do baralho, como forma de obter as 40 cartas necessárias para jogar,
⑳       conjunto conhecido como Baralho Espanhol.
⑳     </ion-card-content>
⑳
⑳     <ion-card-content>
⑳       <ion-button size="small" fill="outline" slot="end">Ver Regras</ion-button>
⑳     </ion-card-content>
⑳   </ion-card>
⑳
⑳ </ion-content>
```

A ação do jogo



```
16<ion-content>
17  <ion-card>
18    <ion-item>
19      <ion-avatar slot="start">
20        
21      </ion-avatar>
22
23      <ion-label class="ion-text-wrap">
24        <ion-text color="dark"><h3>Guilherme Madalozzo</h3></ion-text>
25        <p>Truco Paulista</p>
26      </ion-label>
27
28      <ion-button color="tertiary" fill="clear">
29        <ion-icon size="large" name="star" color="tertiary"></ion-icon>
30      </ion-button>
31    </ion-item>
32
33  <ion-card-content>
34    É jogada com o baralho francês de 52 cartas, removendo-se as cartas
35    8, 9 e 10 do baralho, como forma de obter as 40 cartas necessárias para jogar,
36    conjunto conhecido como Baralho Espanhol.
37  </ion-card-content>
38
39  <ion-card-content>
40    <ion-button size="small" fill="outline" slot="end">Ver Regras</ion-button>
41  </ion-card-content>
42
43  </ion-card>
44
45</ion-content>
```

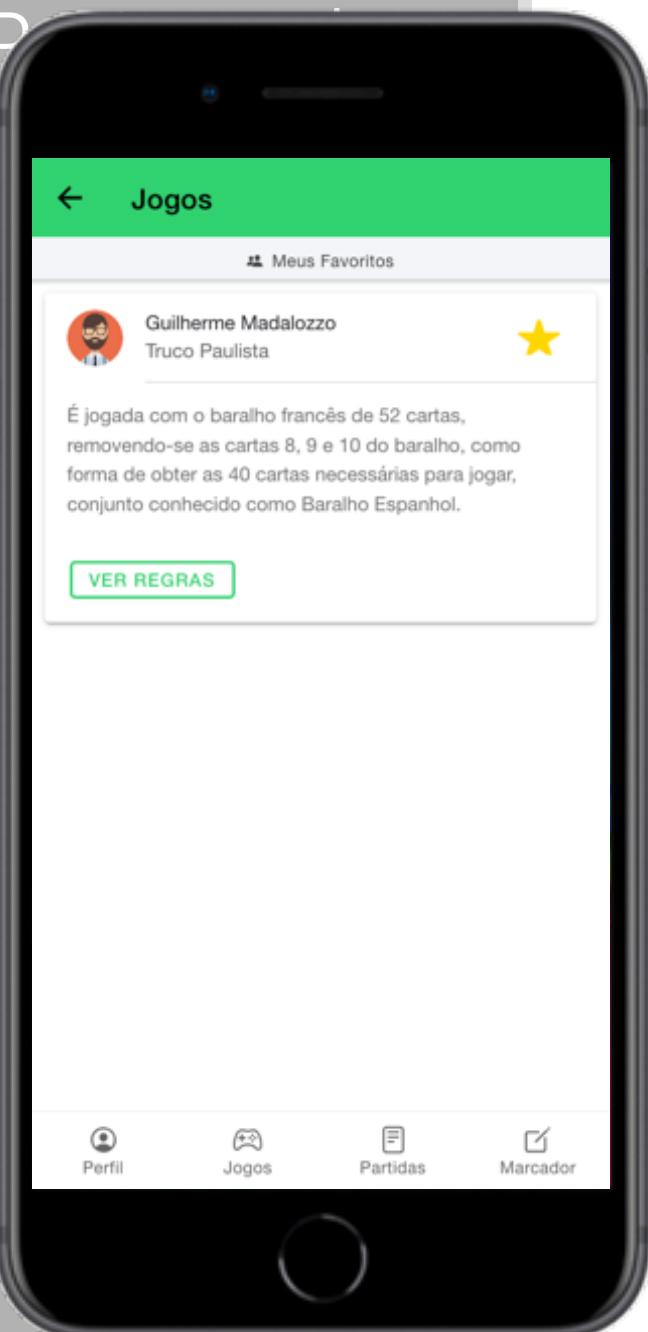


```
①  <ion-content>
②      <ion-card>
③          <ion-item>
④              <ion-avatar slot="start">
⑤                  
⑥              </ion-avatar>
⑦
⑧              <ion-label class="ion-text-wrap">
⑨                  <ion-text color="dark"><h3>Guilherme Madalozzo</h3></ion-text>
⑩                  <p>Truco Paulista</p>
⑪              </ion-label>
⑫
⑬              <ion-button color="tertiary" fill="clear">
⑭                  <ion-icon size="large" name="star" color="tertiary"></ion-icon>
⑮              </ion-button>
⑯          </ion-item>
⑰
⑱          <ion-card-content>
⑲              É jogada com o baralho francês de 52 cartas, removendo-se as cartas
⑳                  8, 9 e 10 do baralho, como forma de obter as 40 cartas necessárias para jogar,
⑳                  conjunto conhecido como Baralho Espanhol.
⑳          </ion-card-content>
⑳
⑳          <ion-card-content>
⑳              <ion-button size="small" fill="outline" slot="end">Ver Regras</ion-button>
⑳          </ion-card-content>
⑳
⑳      </ion-card>
⑳
⑳  </ion-content>
```

Botão para analisar as regras

Vamos projetar as telas dos jogos

- Agora, vamos projetar a tela de Jogos Favoritos

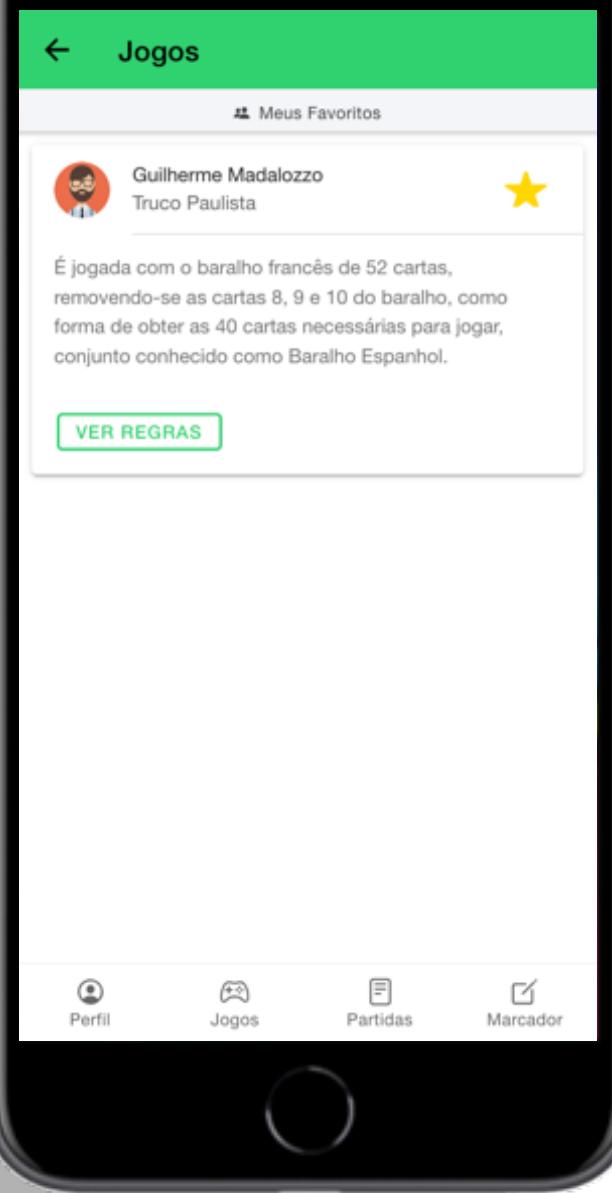


Agora, temos que ajustar o SCSS

## ? jogos-favoritos.page.scss ×

PlayingScore > src > app > jogos-favoritos > ? jogos-favori

```
1 | .tab-bar-sub {  
2 |     height: 30px;  
3 |     position: relative;  
4 |     background-color: ■#f4f5f8 !important;  
5 |     text-align: center !important;  
6 | }  
7 |  
8 | .subtitle {  
9 |     position: relative;  
10 |     top: -11px !important;  
11 |     font-size: 12px;  
12 | }  
13 |  
14 | ion-icon {  
15 |     font-size: 12px;  
16 | }
```



# Projetando as Telas de Jogos

---

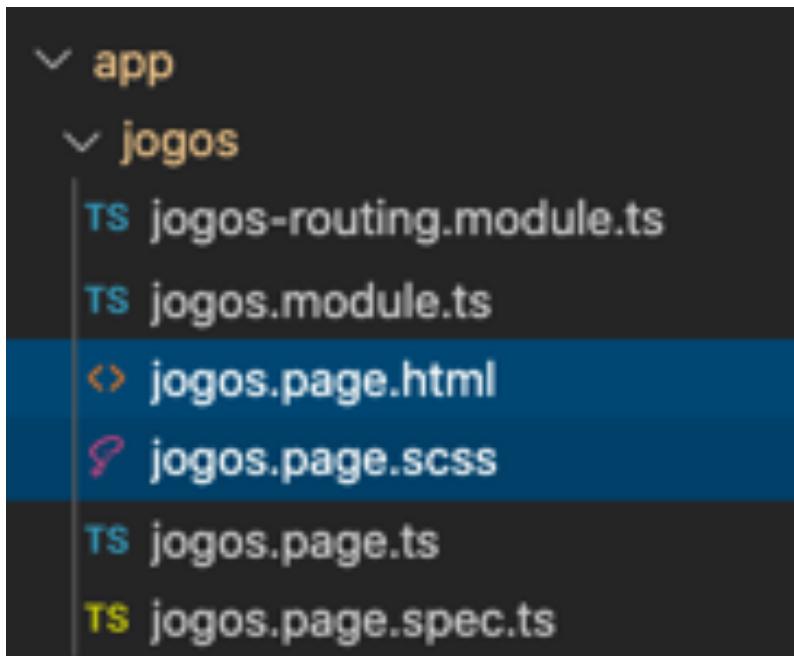
Vamos projetar as telas dos jogos

- Agora, vamos projetar a tela principal de Jogos
- Esta tela irá listar os jogos do usuário logado, e dar opção de inserir novo jogo

# Projetando as Telas de Jogos

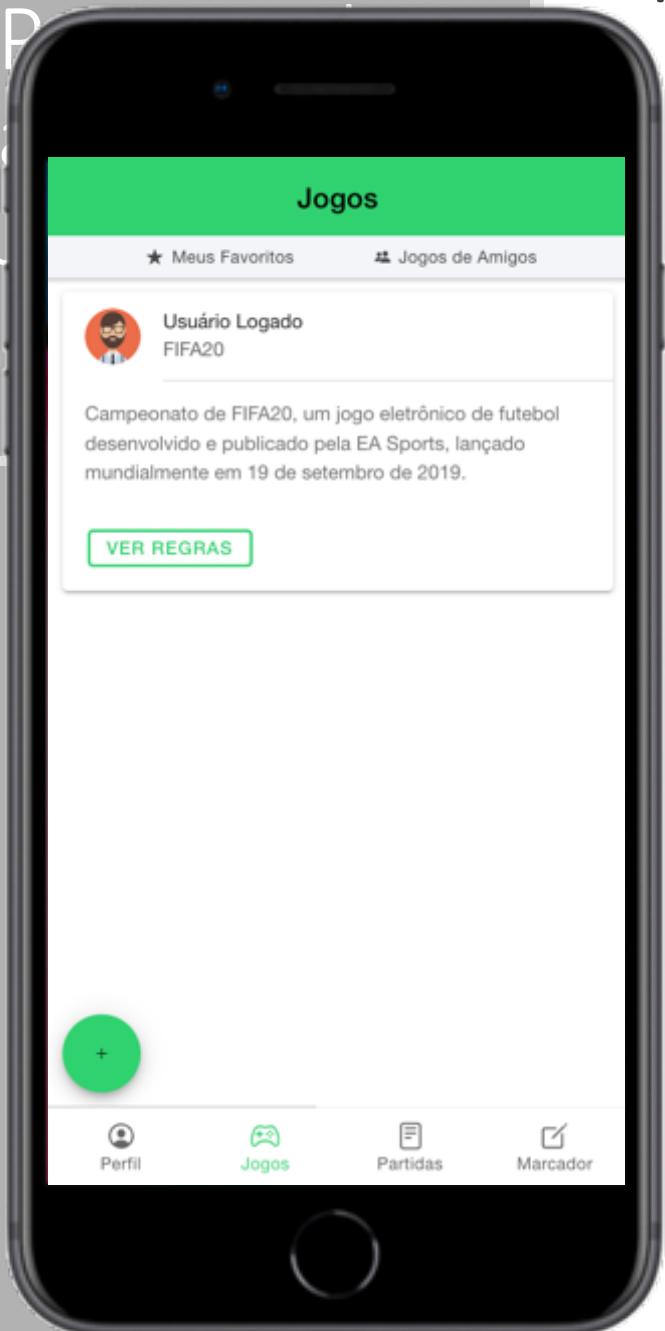
Vamos projetar as telas dos jogos

- Agora, vamos projetar a tela principal de Jogos
- Esta tela irá listar os jogos do usuário logado, e dar opção de inserir novo jogo



## Vamos projetar as telas dos jogos

- Agora, vamos projetar a tela principal de Jogos
- Esta tela irá listar os jogos do usuário logado, e dar opção de inserir novo jogo



## Vamos projetar as telas dos jogos

- Agora, vamos projetar a tela principal de Jogos



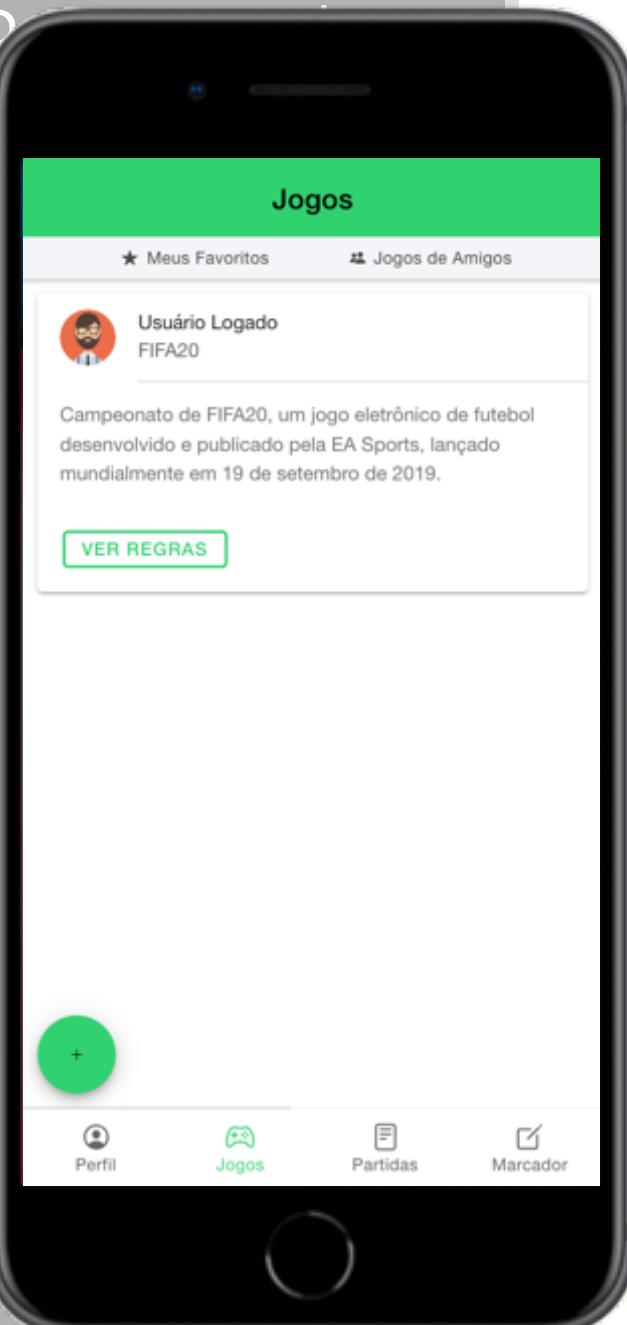
◦ jogos.page.html ◊

PlayingScore > src > app > jogos > ◦ jogos.page.html > ion-content > ion-card > ion-item

```
1   <ion-header [translucent]="true">
2     <ion-toolbar class="ion-text-center" color="primary">
3       <ion-title>Jogos</ion-title>
4     </ion-toolbar>
5
6     <ion-toolbar class="tab-bar-sub" color="light" >
7       <ion-tabs>
8         <ion-tab-bar class="tab-bar-sub" color="light" slot="top">
9           <ion-tab-button layout="icon-start" tab="tabJogosFavoritos">
10             <ion-icon name="star"></ion-icon>
11             <ion-label>Meus Favoritos</ion-label>
12           </ion-tab-button>
13
14           <ion-tab-button layout="icon-start" tab="tabJogosAmigos">
15             <ion-icon name="people"></ion-icon>
16             <ion-label>Jogos de Amigos</ion-label>
17           </ion-tab-button>
18         </ion-tab-bar>
19       </ion-tabs>
20     </ion-toolbar>
21   </ion-header>
```

## Vamos projetar as telas dos jogos

- Agora, vamos projetar a tela principal de Jogos



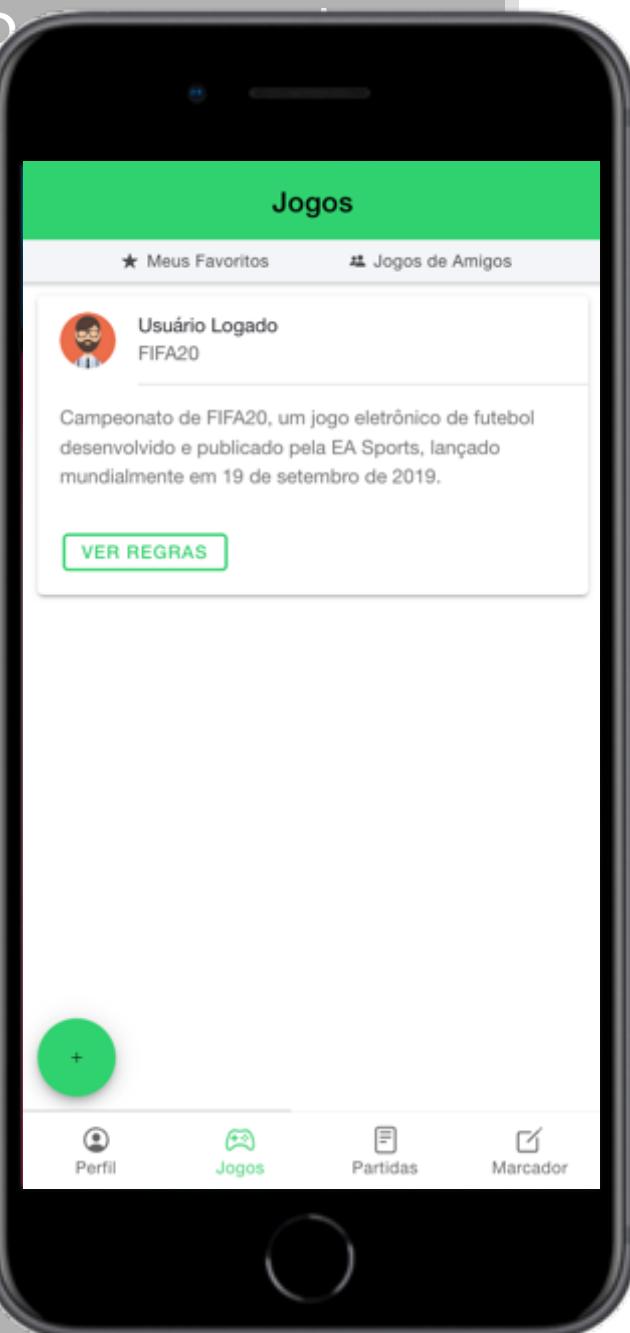
↳ jogos.page.html ✎

```
PlayingScore > src > app > jogos > ↳ jogos.page.html > ion-content
1   <ion-header [translucent]="true">
2     <ion-toolbar class="ion-text-center" color="primary">
3       <ion-title>Jogos</ion-title>
4     </ion-toolbar>
5
6     <ion-toolbar class="tab-bar-sub" color="light" >
7       <ion-tabs>
8         <ion-tab-bar class="tab-bar-sub" color="light" slot="top">
9           <ion-tab-button layout="icon-start" tab="tabJogosFavoritos">
10             <ion-icon name="star"></ion-icon>
11             <ion-label>Meus Favoritos</ion-label>
12           </ion-tab-button>
13
14           <ion-tab-button layout="icon-start" tab="tabJogosAmigos">
15             <ion-icon name="people"></ion-icon>
16             <ion-label>Jogos de Amigos</ion-label>
17           </ion-tab-button>
18         </ion-tab-bar>
19       </ion-tabs>
20     </ion-toolbar>
21   </ion-header>
```

Faremos a alteração do cabeçalho adicionando as ações como subtítulo

## Vamos projetar as telas dos jogos

- Agora, vamos projetar a tela principal de Jogos



◦ jogos.page.html ×

PlayingScore > src > app > jogos > ◦ jogos.page.html > ion-content

```
1   <ion-header [translucent] = "true">
2     <ion-toolbar class="ion-text-center" color="primary">
3       <ion-title>Jogos</ion-title>
4     </ion-toolbar>
5
6     <ion-toolbar class="tab-bar-sub" color="light" >
7       <ion-tabs>
8         <ion-tab-bar class="tab-bar-sub" color="light" slot="top">
9           <ion-tab-button layout="icon-start" tab="tabJogosFavoritos">
10             <ion-icon name="star"></ion-icon>
11             <ion-label>Meus Favoritos</ion-label>
12           </ion-tab-button>
13
14           <ion-tab-button layout="icon-start" tab="tabJogosAmigos">
15             <ion-icon name="people"></ion-icon>
16             <ion-label>Jogos de Amigos</ion-label>
17           </ion-tab-button>
18         </ion-tab-bar>
19       </ion-tabs>
20     </ion-toolbar>
21   </ion-header>
```

Mantemos o título da página

## Vamos projetar as telas dos jogos

- Agora, vamos projetar a tela principal de Jogos



↳ jogos.page.html ✎

```
PlayingScore > src > app > jogos > ↳ jogos.page.html > ion-content
1   <ion-header [translucent]="true">
2     <ion-toolbar class="ion-text-center" color="primary">
3       <ion-title>Jogos</ion-title>
4     </ion-toolbar>
5
6     <ion-toolbar class="tab-bar-sub" color="light" >
7       <ion-tabs>
8         <ion-tab-bar class="tab-bar-sub" color="light" slot="top">
9           <ion-tab-button layout="icon-start" tab="tabJogosFavoritos">
10          <ion-icon name="star"></ion-icon>
11          <ion-label>Meus Favoritos</ion-label>
12        </ion-tab-button>
13
14        <ion-tab-button layout="icon-start" tab="tabJogosAmigos">
15          <ion-icon name="people"></ion-icon>
16          <ion-label>Jogos de Amigos</ion-label>
17        </ion-tab-button>
18      </ion-tab-bar>
19    </ion-tabs>
20  </ion-toolbar>
21 </ion-header>
```

Criamos o toolbar com as duas opções de navegação

# Vamos projetar as telas dos jogos

- Agora, vamos projetar a tela principal de jogos

jogos.page.html x

PlayingScore > src > app > jogos > jogos.page.html > ...

```
43
44
45
46
47
48
49
50
51
```

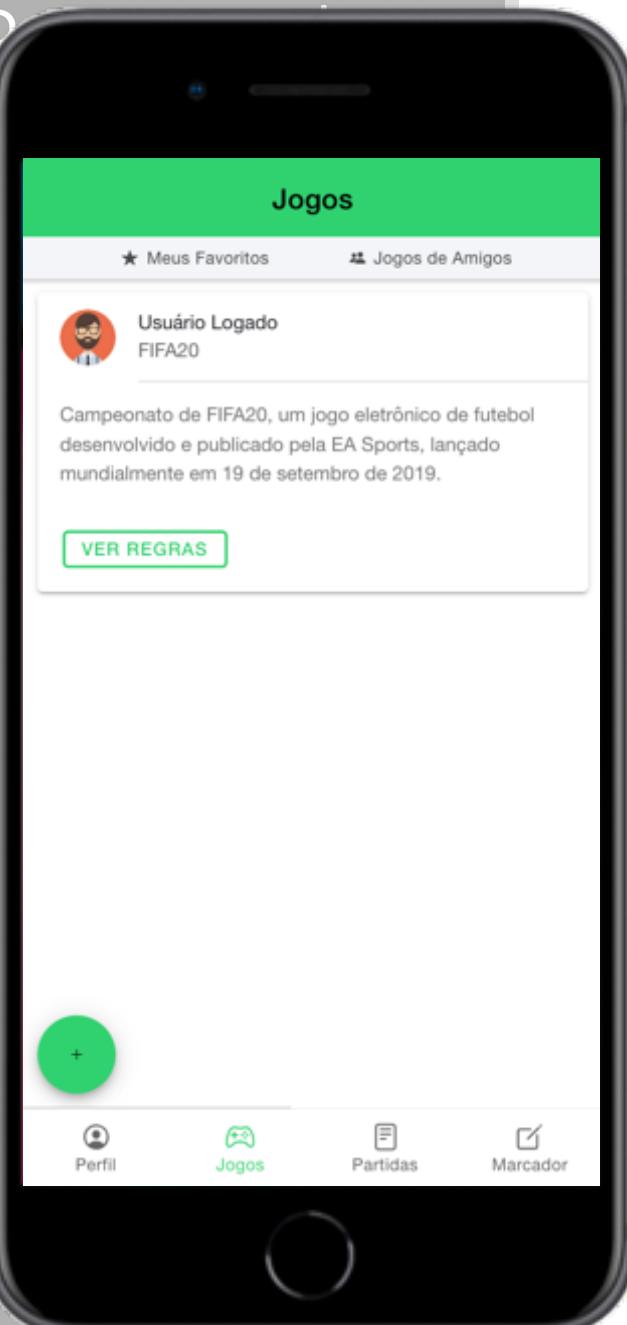
```
<ion-content>
  <ion-card>
    <ion-item>
      <ion-avatar slot="start">
        
      </ion-avatar>

      <ion-label class="ion-text-wrap">
        <ion-text color="dark"><h3>Usuário Logado</h3></ion-text>
        <p>FIFA20</p>
      </ion-label>
    </ion-item>

    <ion-card-content>
      Campeonato de FIFA20, um jogo eletrônico de futebol desenvolvido e publicado pela EA Sports, lançado mundialmente em 19 de setembro de 2019.
    </ion-card-content>

    <ion-card-content>
      <ion-button size="small" fill="outline" slot="end">Ver Regras</ion-button>
    </ion-card-content>
  </ion-card>

  <ion-fab vertical="bottom" horizontal="start" slot="fixed">
    <ion-fab-button size="large"><ion-icon name="add"></ion-icon></ion-fab-button>
  </ion-fab>
</ion-content>
```



## Vamos projetar as telas dos jogos

- Agora, vamos projetar a tela principal de jogos

jogos.page.html x

PlayingScore > src > app > jogos > jogos.page.html > ...

```
43
44
45
46
47
48
49
50
51
```

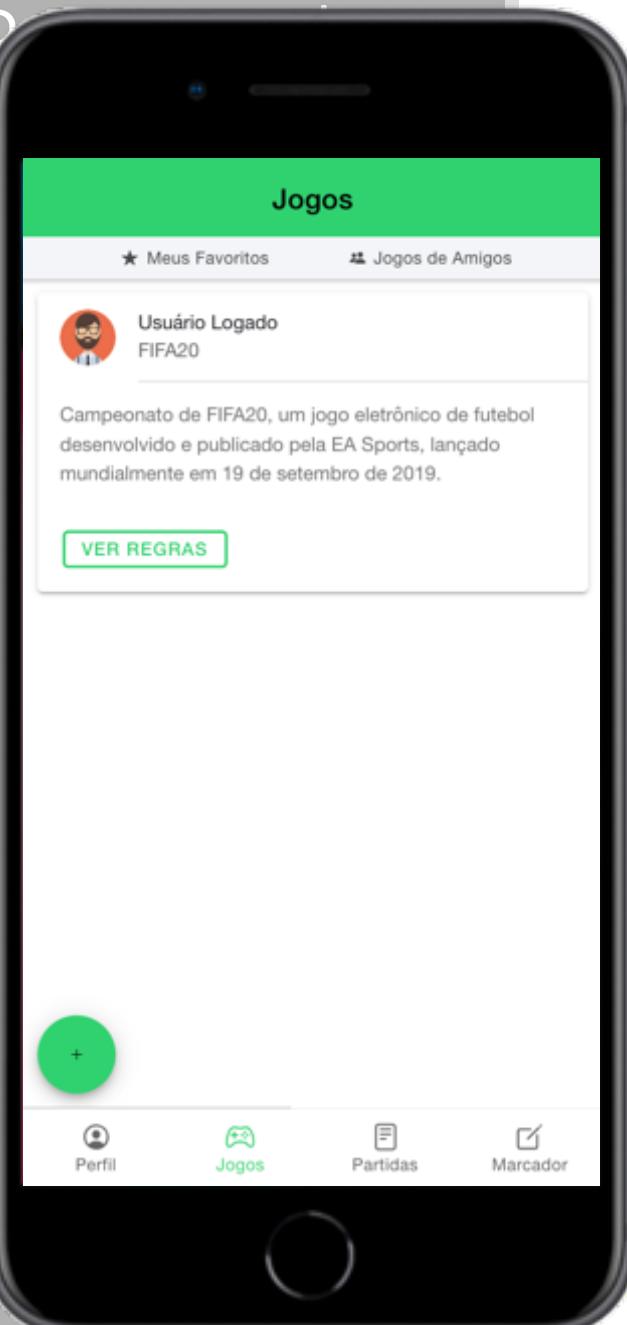
```
<ion-content>
  <ion-card>
    <ion-item>
      <ion-avatar slot="start">
        
      </ion-avatar>

      <ion-label class="ion-text-wrap">
        <ion-text color="dark"><h3>Usuário Logado</h3></ion-text>
        <p>FIFA20</p>
      </ion-label>
    </ion-item>

    <ion-card-content>
      Campeonato de FIFA20, um jogo eletrônico de futebol
      desenvolvido e publicado pela EA Sports,
      lançado mundialmente em 19 de setembro de 2019.
    </ion-card-content>

    <ion-card-content>
      <ion-button size="small" fill="outline" slot="end">Ver Regras</ion-button>
    </ion-card-content>
  </ion-card>

  <ion-fab vertical="bottom" horizontal="start" slot="fixed">
    <ion-fab-button size="large"><ion-icon name="add"></ion-icon></ion-fab-button>
  </ion-fab>
</ion-content>
```



Na camada de conteúdo desta tela, vamos criar um card exemplo e o botão de adicionar

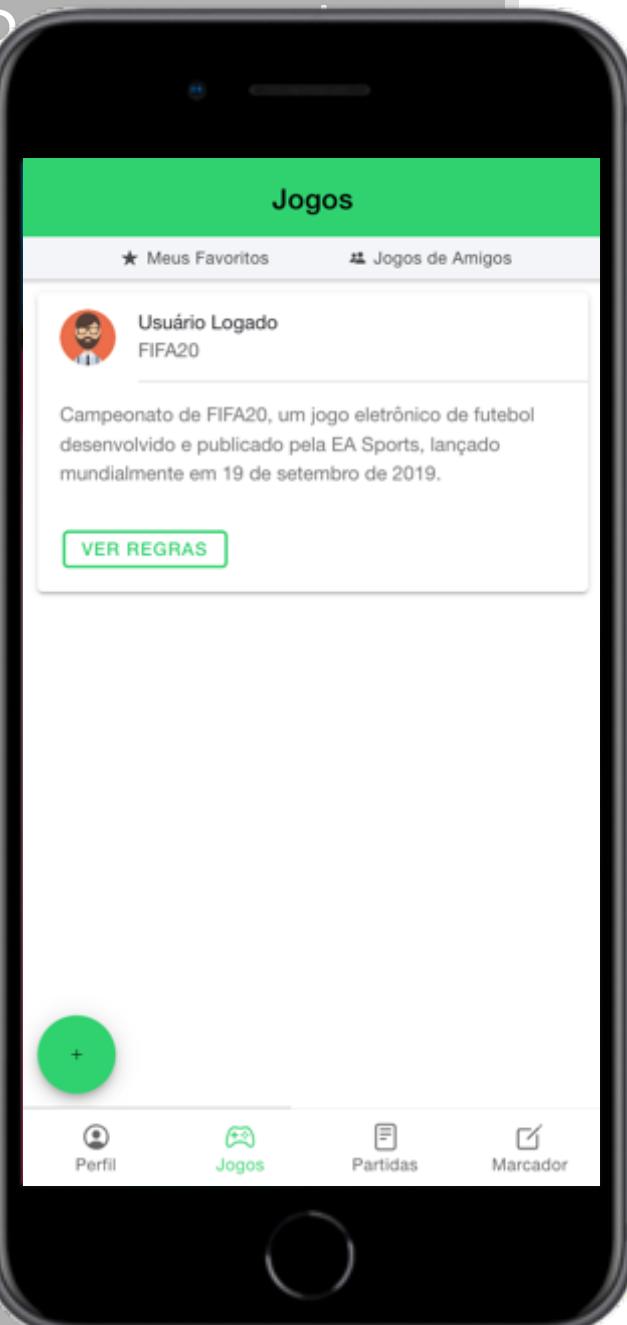
## Vamos projetar as telas dos jogos

- Agora, vamos projetar a tela principal de jogos

jogos.page.html x

PlayingScore > src > app > jogos > jogos.page.html > ...

```
23 | 
24 | <ion-content>
25 |   <ion-card>
26 |     <ion-item>
27 |       <ion-avatar slot="start">
28 |         
29 |       </ion-avatar>
30 | 
31 |       <ion-label class="ion-text-wrap">
32 |         <ion-text color="dark"><h3>Usuário Logado</h3></ion-text>
33 |         <p>FIFA20</p>
34 |       </ion-label>
35 |     </ion-item>
36 | 
37 |     <ion-card-content>
38 |       Campeonato de FIFA20, um jogo eletrônico de futebol
39 |       desenvolvido e publicado pela EA Sports,
40 |       lançado mundialmente em 19 de setembro de 2019.
41 |     </ion-card-content>
42 | 
43 |     <ion-card-content>
44 |       <ion-button size="small" fill="outline" slot="end">Ver Regras</ion-button>
45 |     </ion-card-content>
46 |   </ion-card>
47 | 
48 |   <ion-fab vertical="bottom" horizontal="start" slot="fixed">
49 |     <ion-fab-button size="large"><ion-icon name="add"></ion-icon></ion-fab-button>
50 |   </ion-fab>
51 | </ion-content>
```



Projetamos o card exemplo, assim como nas demais telas, mas sem o botão de favoritar

## Vamos projetar as telas dos jogos

- Agora, vamos projetar a tela principal de jogos

jogos.page.html x

PlayingScore > src > app > jogos > jogos.page.html > ...

```
43
44
45
46
47
48
49
50
51
```

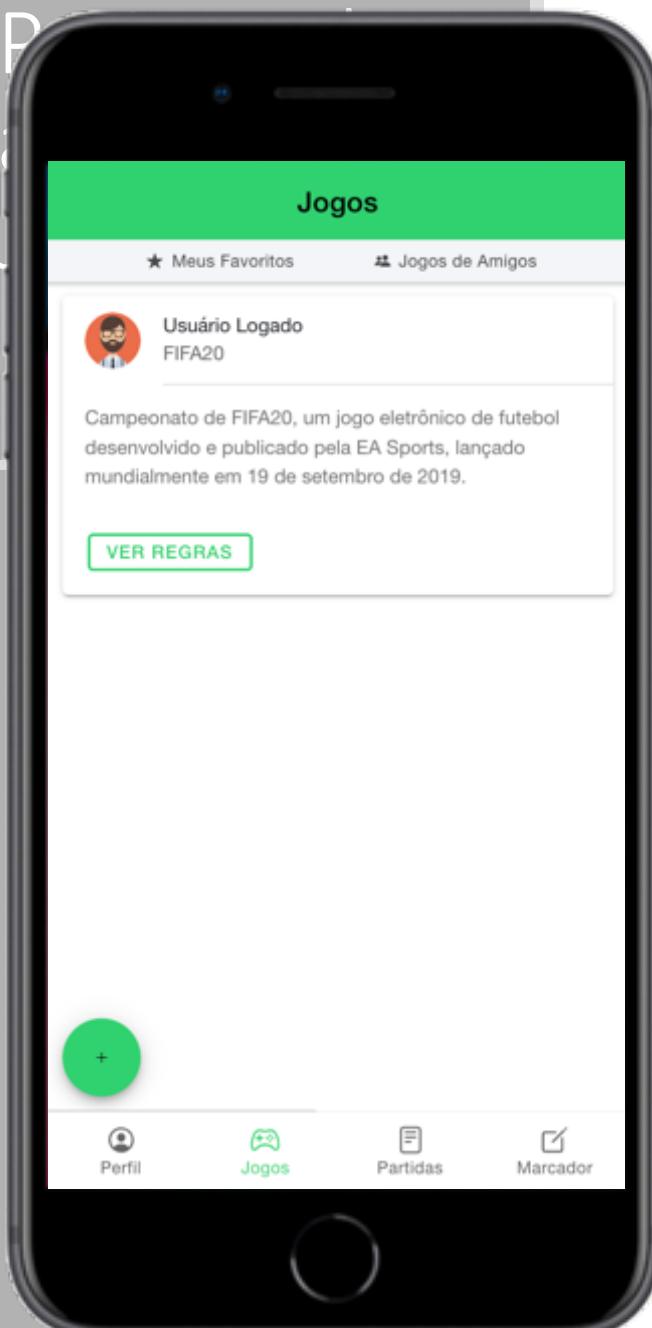
```
<ion-content>
  <ion-card>
    <ion-item>
      <ion-avatar slot="start">
        
      </ion-avatar>

      <ion-label class="ion-text-wrap">
        <ion-text color="dark"><h3>Usuário Logado</h3></ion-text>
        <p>FIFA20</p>
      </ion-label>
    </ion-item>

    <ion-card-content>
      Campeonato de FIFA20, um jogo eletrônico de futebol
      desenvolvido e publicado pela EA Sports,
      lançado mundialmente em 19 de setembro de 2019.
    </ion-card-content>

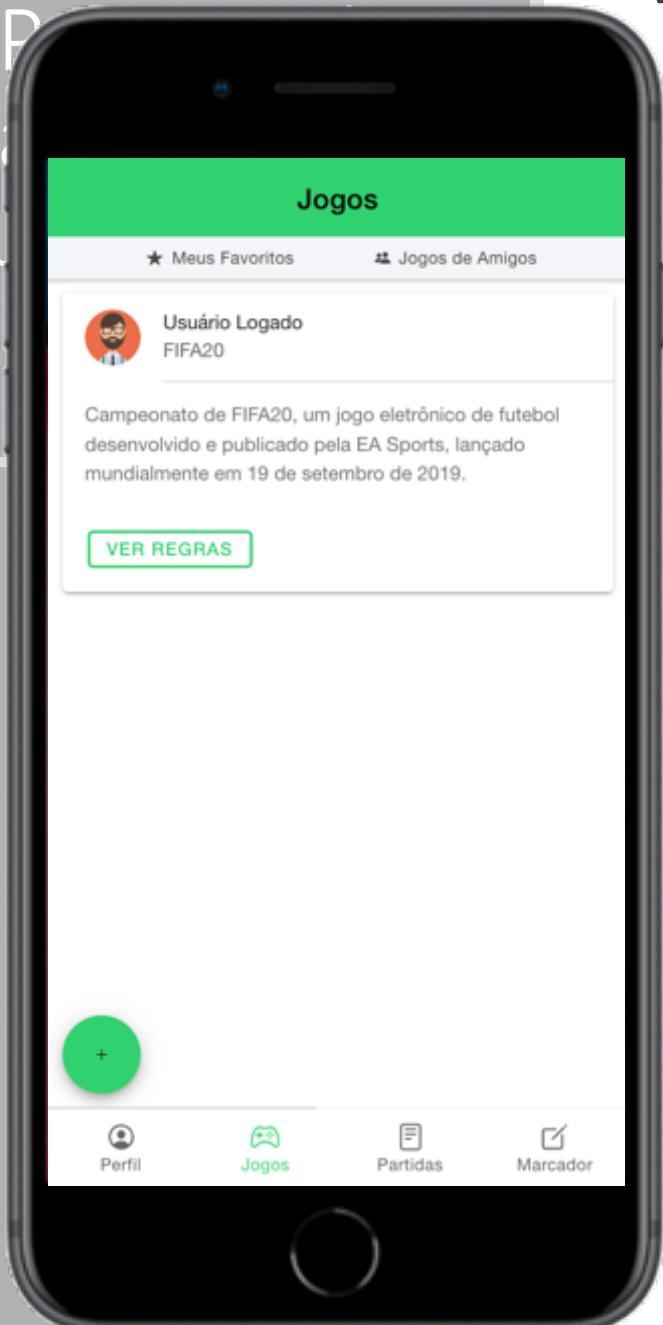
    <ion-card-content>
      <ion-button size="small" fill="outline" slot="end">Ver Regras</ion-button>
    </ion-card-content>
  </ion-card>
</ion-content>
```

Adicionamos um botão flutuante, posicionado na parte inferior esquerda da tela

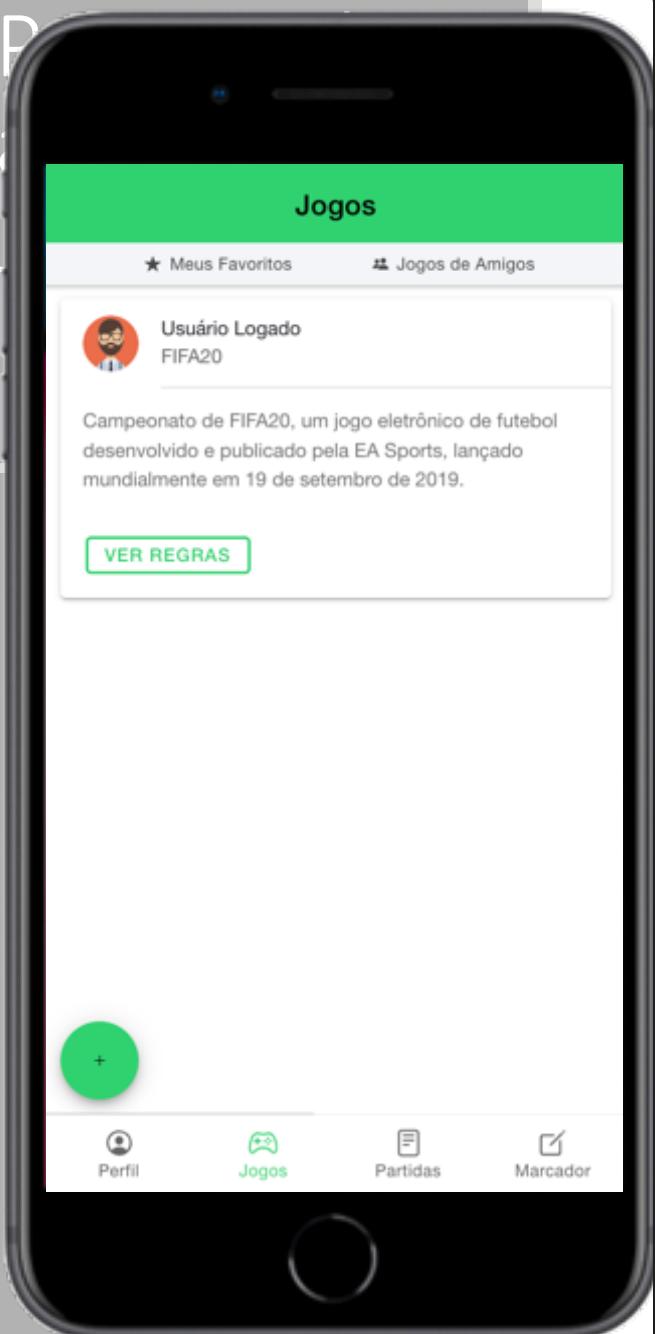


## Vamos projetar as telas dos jogos

- Agora, vamos projetar a tela principal de Jogos
- Esta tela irá listar os jogos do usuário logado, e dar opção de inserir novo jogo



O SCSS será o mesmo das outras telas projetadas



## ♀ jogos.page.scss ×

PlayingScore > src > app > jogos > ♀ jogos.page.scss > ...

```
1  .tab-bar-sub {  
2      height: 30px;  
3      position: relative;  
4      background-color: ■#f4f5f8 !important;  
5      text-align: center !important;  
6  }  
7  
8  .subtitle {  
9      position: relative;  
10     top: -11px !important;  
11     font-size: 12px;  
12  }  
13  
14  ion-icon {  
15      font-size: 12px;  
16  }
```



# Página de Perfil de Usuário

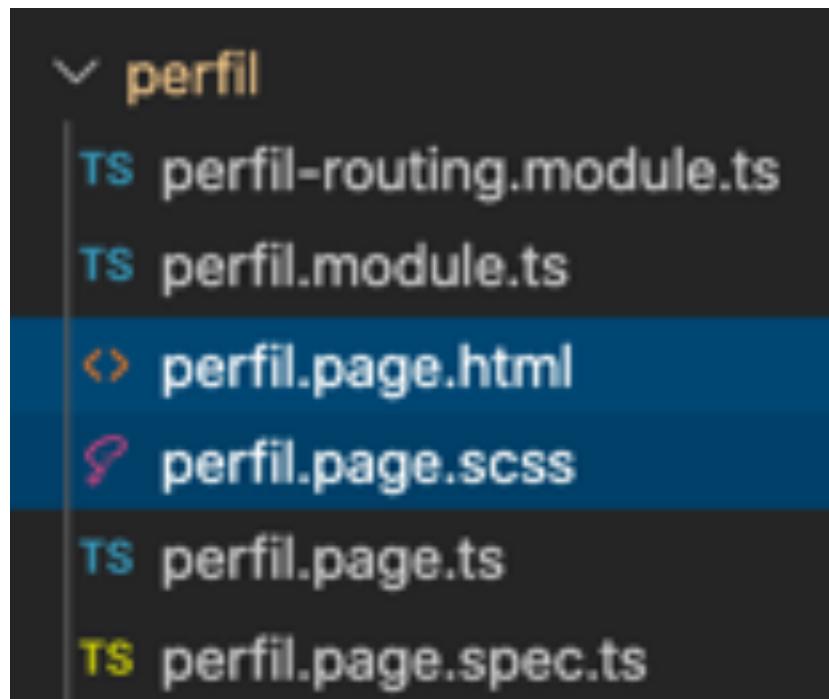
# Página de Perfil de Usuário

---

Agora, vamos projetar a página de perfil de usuário

# Página de Perfil de Usuário

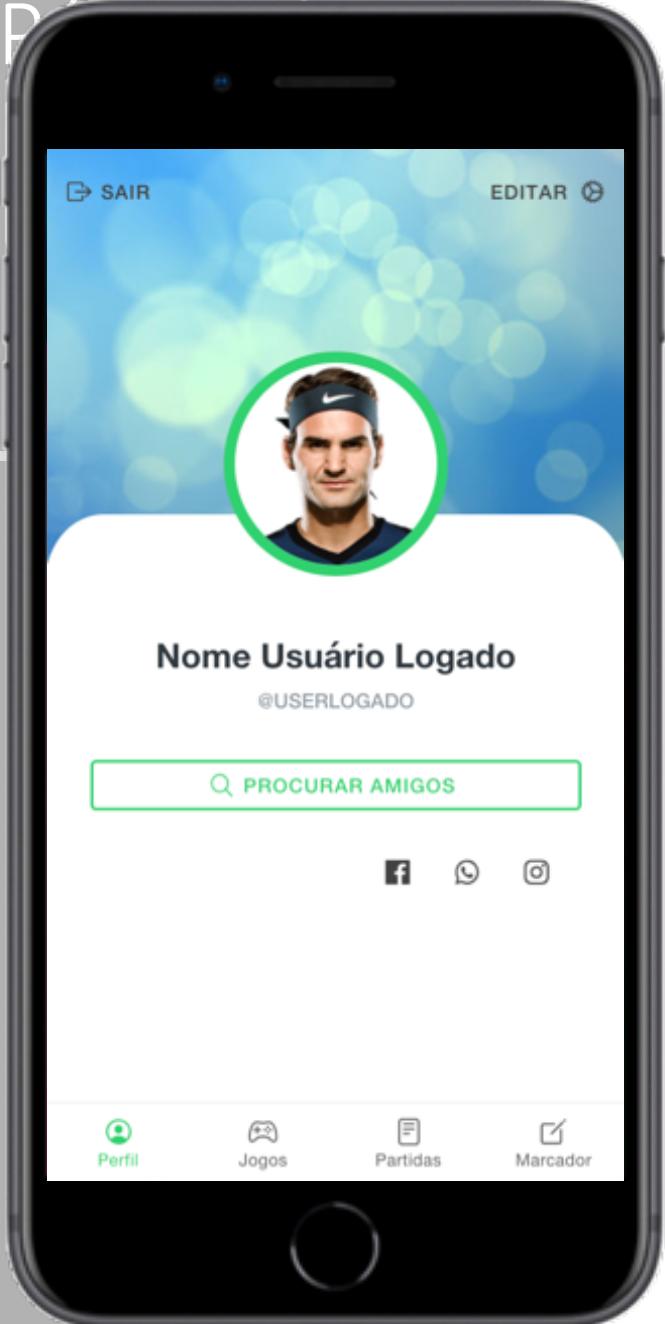
Agora, vamos projetar a página de perfil de usuário



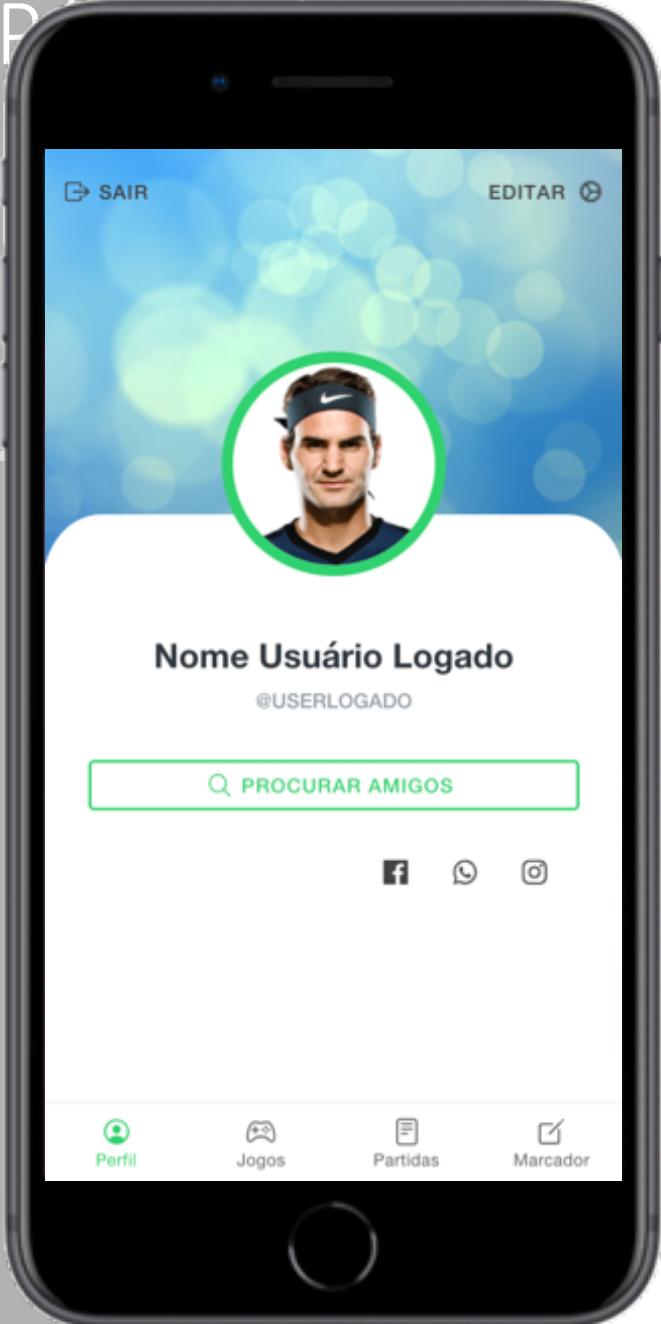
The screenshot shows a file explorer interface with a dark theme. A folder named 'perfil' is expanded, revealing its contents. Inside 'perfil', there are several files and modules:

- `perfil-routing.module.ts` (TS)
- `perfil.module.ts` (TS)
- `perfil.page.html` (HTML)
- `perfil.page.scss` (SCSS)
- `perfil.page.ts` (TS)
- `perfil.page.spec.ts` (TS)

Agora, vamos projetar a página de perfil de usuário



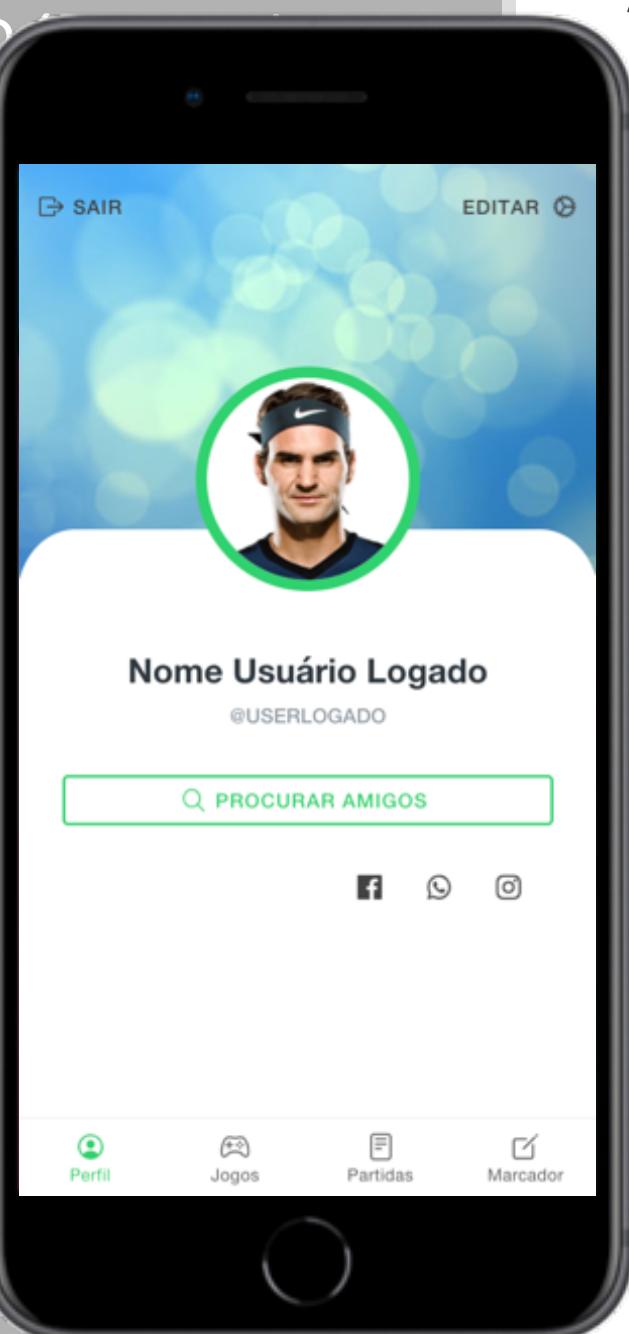
Agora, vamos projetar a página de perfil de usuário



Esta página não terá cabeçalho  
Teremos direto conteúdo

Também, iremos trabalhar com "div"  
para que possamos ver, de fato,  
a semelhança com modelagem web

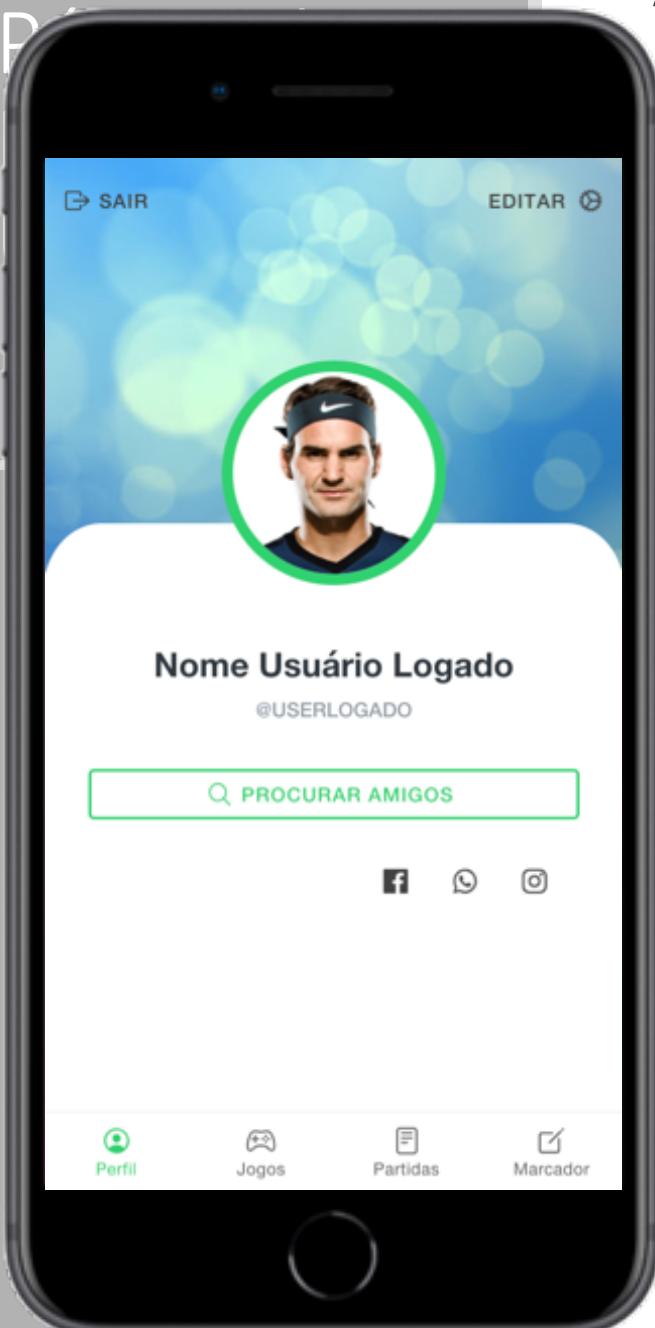
Agora, vamos projetar a página de perfil de usuário



## perfil.page.html ×

```
PlayingScore > src > app > perfil > ◊ perfil.page.html > ion-content > div.card > div
1   <ion-content fullscreen="true" slot="fixed">
2     <ion-toolbar>
3       <ion-buttons slot="start">
4         <ion-button>
5           Sair
6           <ion-icon slot="start" name="exit-outline"></ion-icon>
7         </ion-button>
8       </ion-buttons>
9
10      <ion-buttons slot="secondary">
11        <ion-button>
12          Editar
13          <ion-icon slot="end" name="cog-outline"></ion-icon>
14        </ion-button>
15      </ion-buttons>
16    </ion-toolbar>
17
```

Agora, vamos projetar a página de perfil de usuário

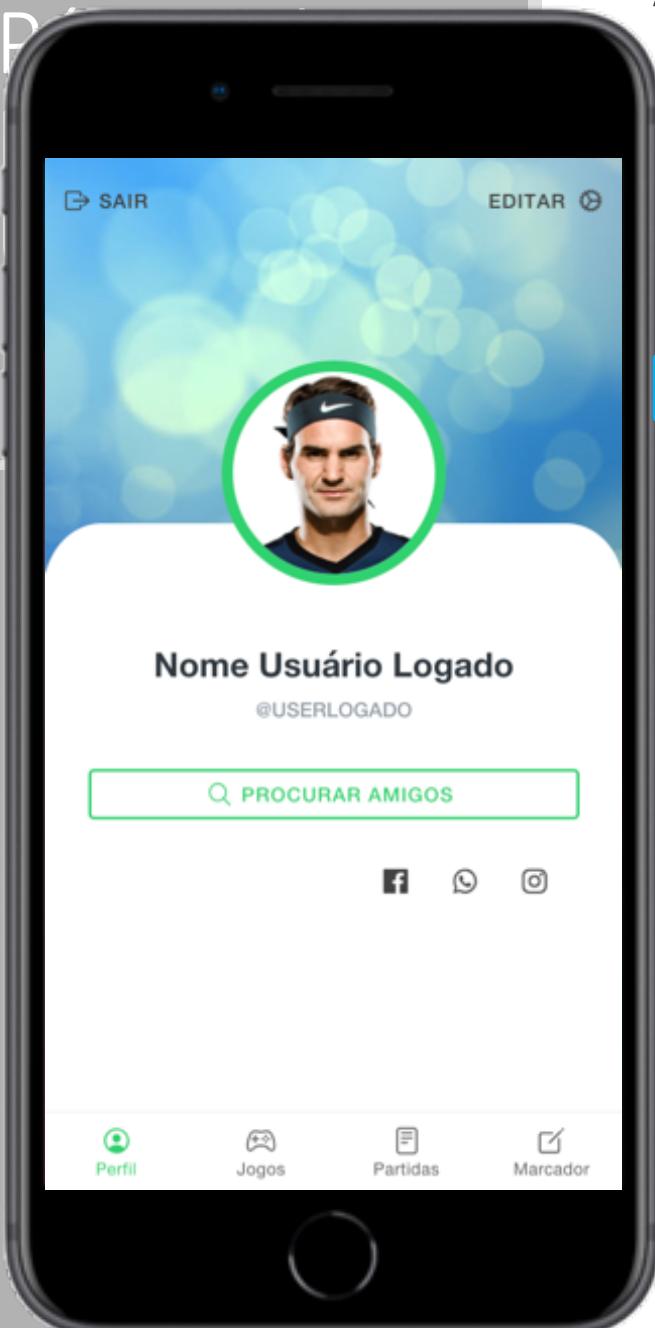


### perfil.page.html

```
PlayingScore > src > app > perfil > perfil.page.html > ion-content > div.card > div
1   <ion-content fullscreen="true" slot="fixed">
2     <ion-toolbar>
3       <ion-buttons slot="start">
4         <ion-button>
5           Sair
6           <ion-icon slot="start" name="exit-outline"></ion-icon>
7         </ion-button>
8       </ion-buttons>
9
10      <ion-buttons slot="secondary">
11        <ion-button>
12          Editar
13          <ion-icon slot="end" name="cog-outline"></ion-icon>
14        </ion-button>
15      </ion-buttons>
16    </ion-toolbar>
17
```

Primeiro projetamos uma toolbar para ações de "sair" e "editar" perfil

Agora, vamos projetar a página de perfil de usuário

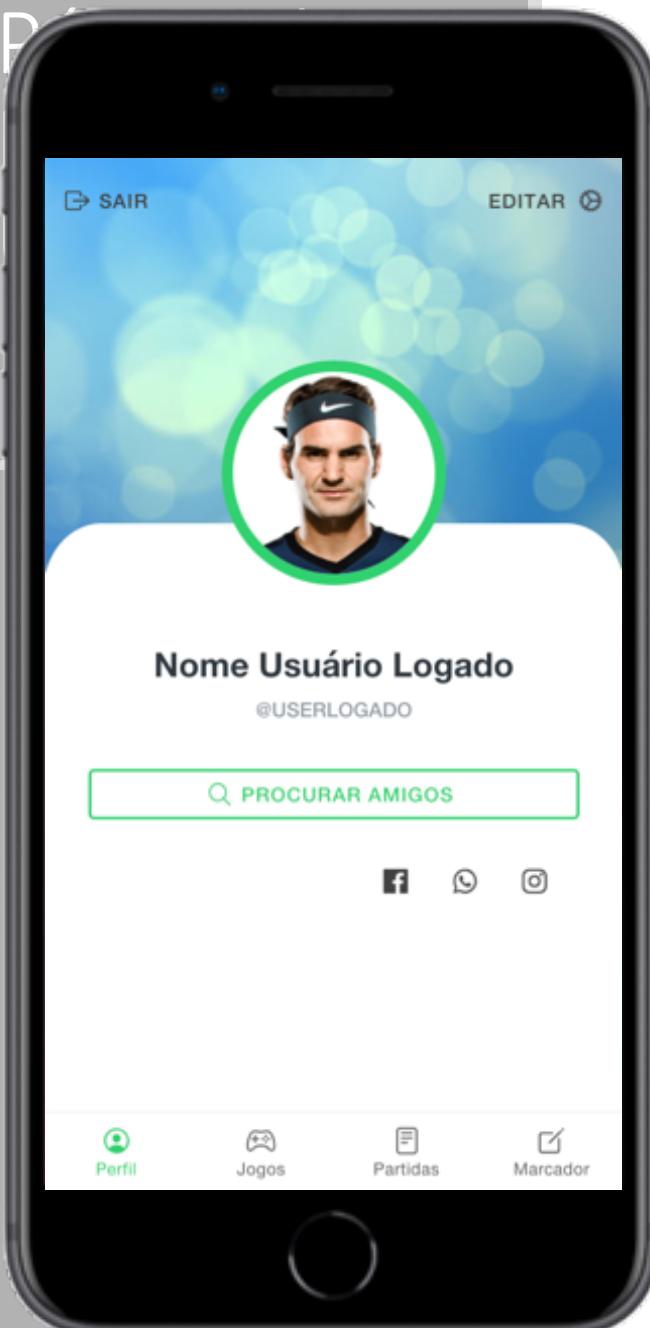


### perfil.page.html

```
PlayingScore > src > app > perfil > perfil.page.html > ion-content > div.card > div
1   <ion-content fullscreen="true" slot="fixed">
2     <ion-toolbar>
3       <ion-buttons slot="start">
4         <ion-button>
5           Sair
6           <ion-icon slot="start" name="exit-outline"></ion-icon>
7         </ion-button>
8       </ion-buttons>
9
10      <ion-buttons slot="secondary">
11        <ion-button>
12          Editar
13          <ion-icon slot="end" name="cog-outline"></ion-icon>
14        </ion-button>
15      </ion-buttons>
16    </ion-toolbar>
17
```

Primeiro projetamos uma toolbar para ações de "sair" e "editar" perfil

Agora, vamos projetar a página de perfil de usuário

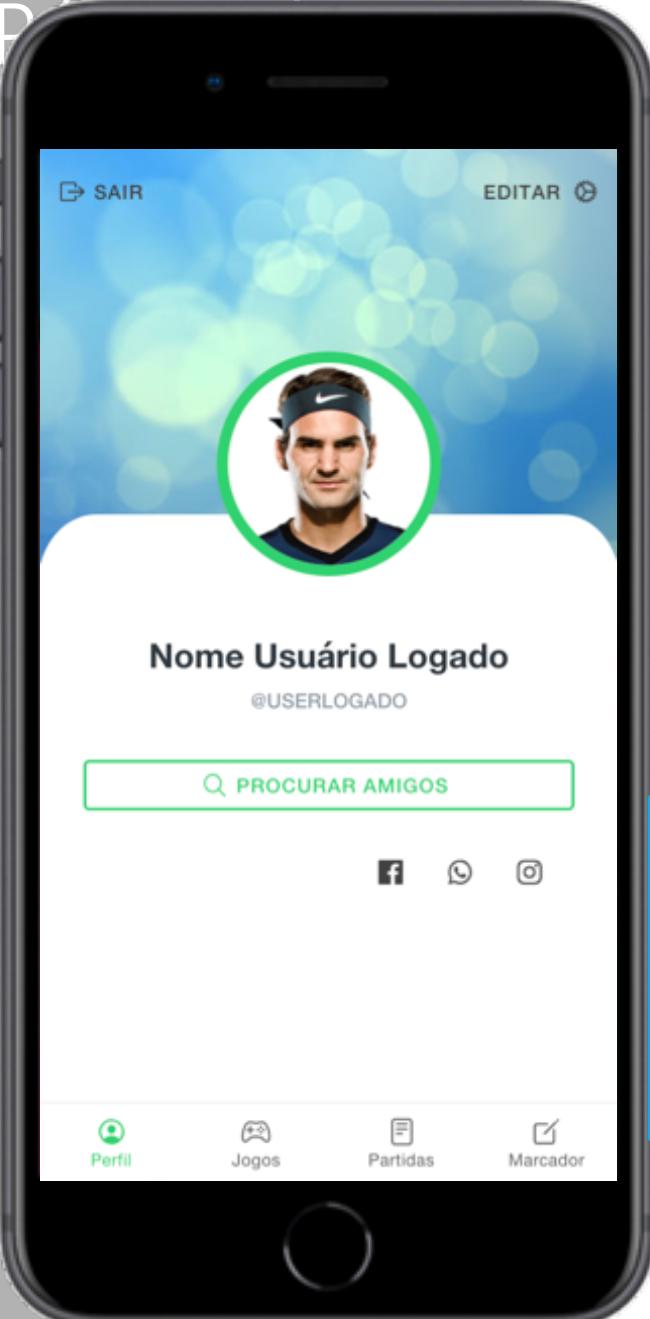


## perfil.page.html

```
PlayingScore > src > app > perfil > perfil.page.html > ion-content > ion-toolbar > ion-buttons slot="start" > ion-button > Sair <ion-icon slot="start" name="exit-outline"></ion-icon> </ion-button> </ion-buttons> <ion-buttons slot="secondary" > ion-button > Editar <ion-icon slot="end" name="cog-outline"></ion-icon> </ion-button> </ion-buttons> </ion-toolbar>
```

Adicionamos o botão "sair" com ícone a esquerda do texto e a esquerda da área da tela

Agora, vamos projetar a página de perfil de usuário

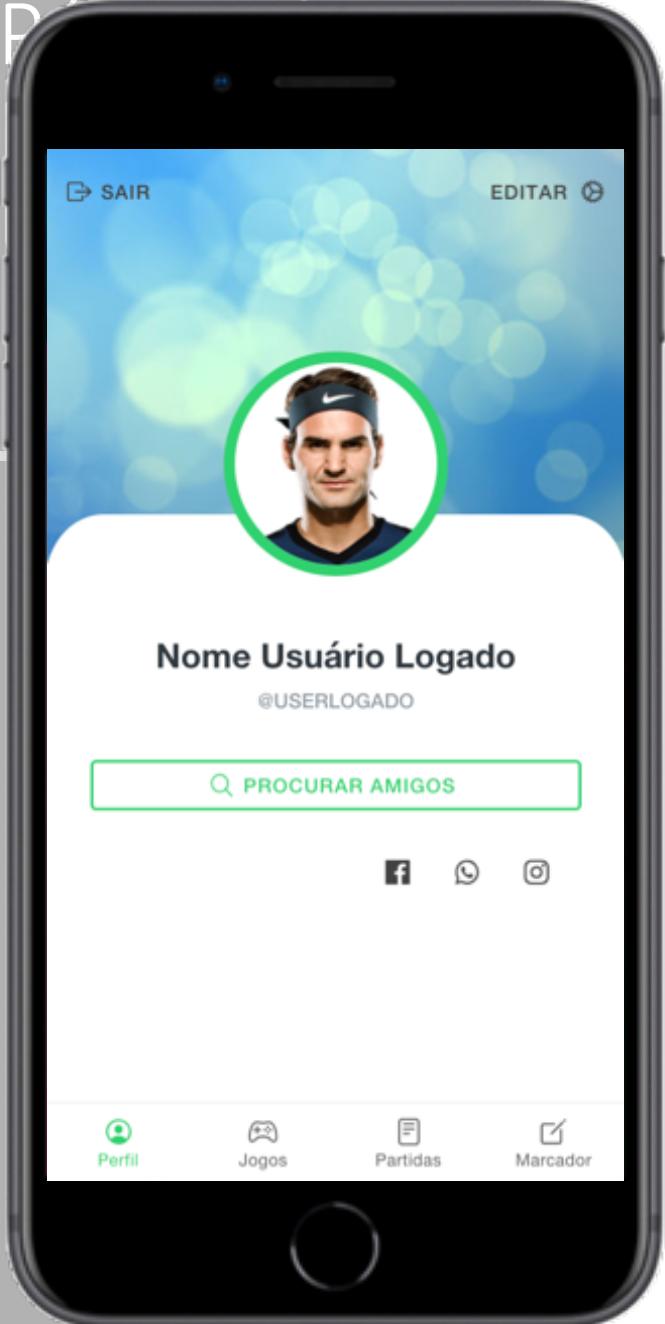


## perfil.page.html

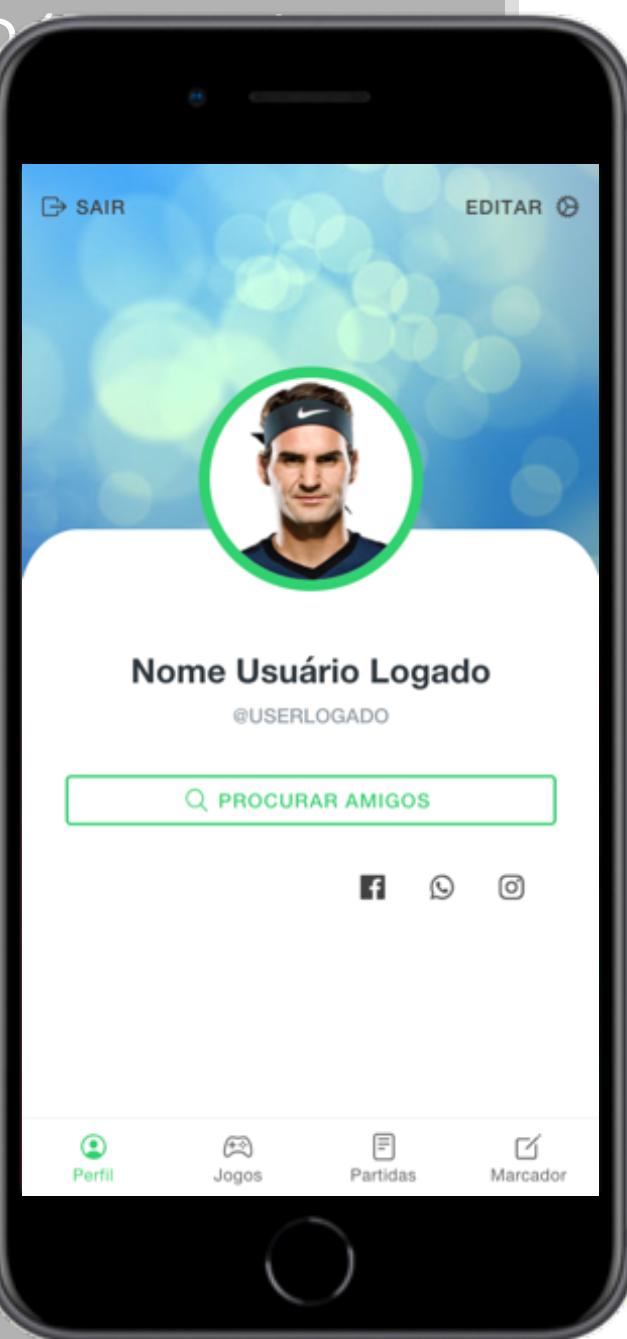
```
PlayingScore > src > app > perfil > perfil.page.html > ion-content > ion-toolbar > ion-buttons > ion-button > Sair <ion-icon slot="start" name="exit-outline"></ion-icon> </ion-button> </ion-buttons> <ion-buttons slot="secondary"> <ion-button> Editar <ion-icon slot="end" name="cog-outline"></ion-icon> </ion-button> </ion-buttons> </ion-toolbar>
```

Adicionamos o botão "editar" com ícone a direta do texto e a direita da área da tela

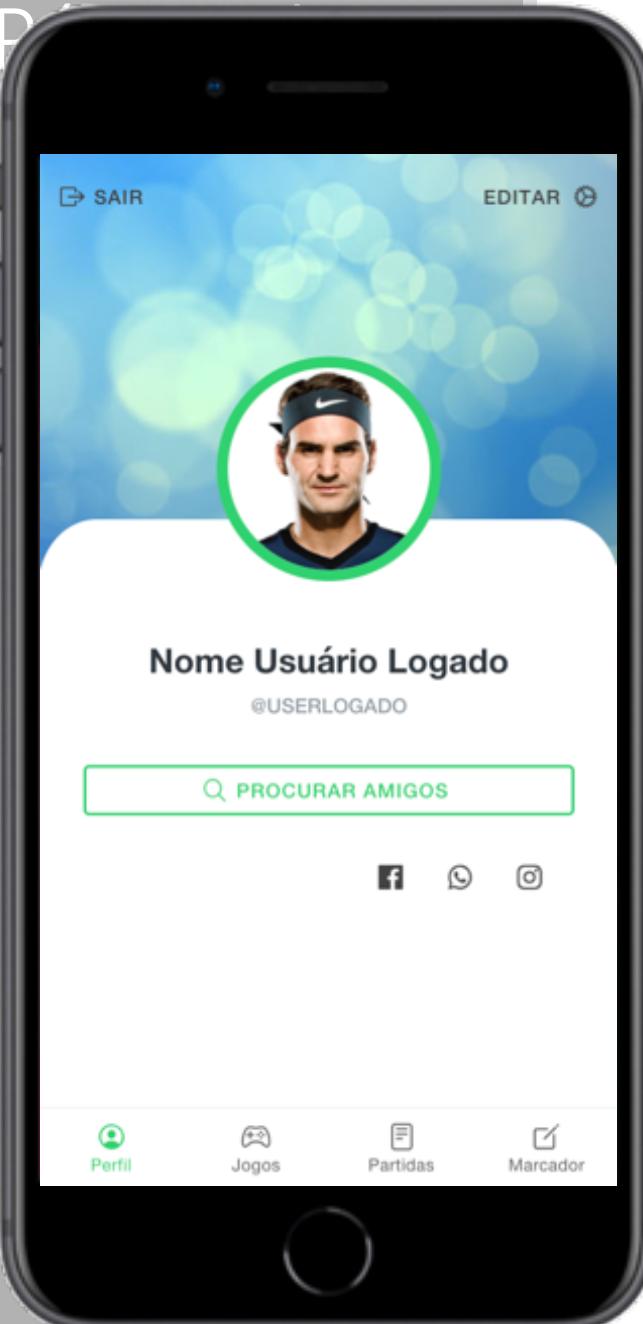
Agora, vamos projetar a página de perfil de usuário



Vamos projetar o conteúdo da tela



```
perfil.page.html X
PlayingScore > src > app > perfil > perfil.page.html > ion-content > ion-toolbar > ion-
1/   <div class="card">
18     <div class="header">
19       <div class="avatar">
20         
21       </div>
22     </div>
23
24
25     <div class="card-body">
26
27       <div class="user-meta ion-text-center">
28         <h3 class="username">Nome Usuário Logado</h3>
29         <h5 class="nickname">@userlogado</h5>
30       </div>
31
32       <ion-button fill="outline" expand="block">
33         Procurar Amigos
34         <ion-icon slot="start" name="search-outline"></ion-icon>
35       </ion-button>
36
37       <ion-toolbar>
38         <ion-buttons slot="secondary">
39           <ion-button class="bt-sociais">
40             <ion-icon name="logo-facebook"></ion-icon>
41           </ion-button>
42
43           <ion-button class="bt-sociais">
44             <ion-icon name="logo-whatsapp"></ion-icon>
45           </ion-button>
46
47           <ion-button class="bt-sociais">
48             <ion-icon name="logo-instagram"></ion-icon>
49           </ion-button>
50         </ion-buttons>
51       </ion-toolbar>
52
53     </div>
54   </div>
55 </ion-content>
```

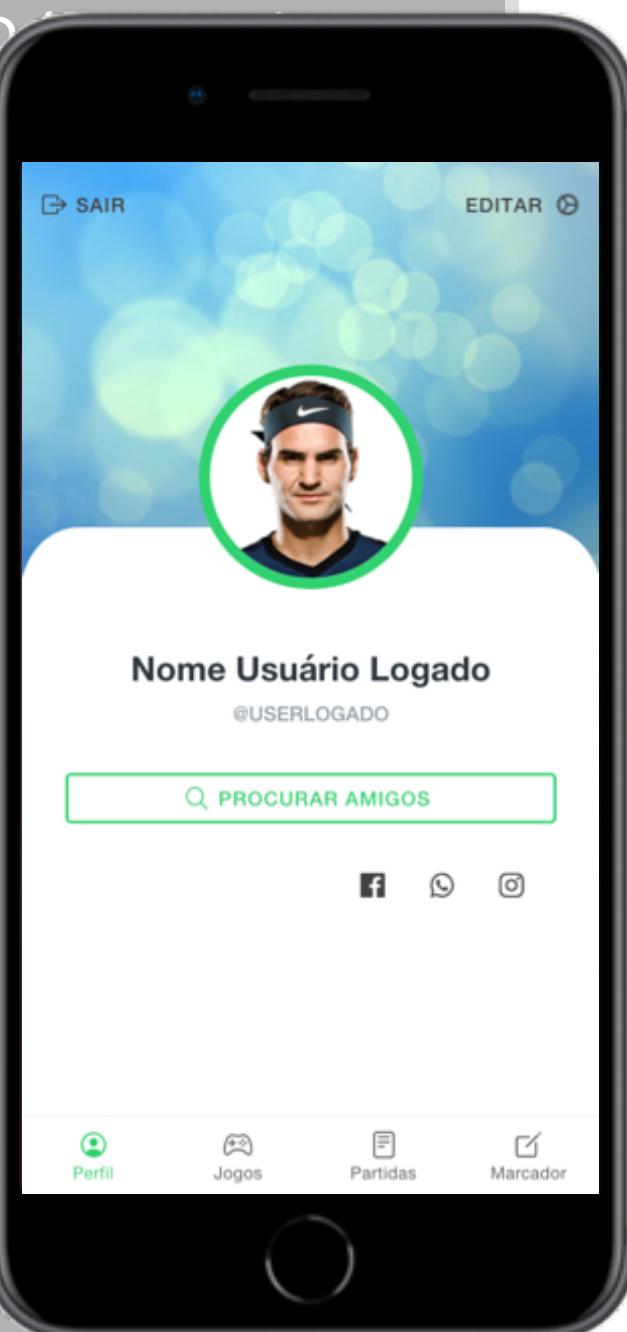


```
perfil.page.html
PlayingScore > src > app > perfil > perfil.page.html > ion-content > ion-toolbar > ion-content
17
18     <div class="card">
19         <div class="header">
20             <div class="avatar">
21                 
22             </div>
23         </div>
24
25         <div class="card-body">
26
27             <div class="user-meta ion-text-center">
28                 <h3 class="username">Nome Usuário Logado</h3>
29             </div>
30
31             <ion-list>
32                 <ion-item>
33                     <ion-icon name="logo-facebook"></ion-icon>
34                     <ion-icon name="logo-whatsapp"></ion-icon>
35                     <ion-icon name="logo-instagram"></ion-icon>
36                 </ion-item>
37             </ion-list>
38
39             <ion-buttons>
40                 <ion-button class="bt-sociais">
41                     <ion-icon name="logo-facebook"></ion-icon>
42                 </ion-button>
43
44                 <ion-button class="bt-sociais">
45                     <ion-icon name="logo-whatsapp"></ion-icon>
46                 </ion-button>
47
48                 <ion-button class="bt-sociais">
49                     <ion-icon name="logo-instagram"></ion-icon>
50                 </ion-button>
51             </ion-buttons>
52
53         </div>
54     </div>
55 </ion-content>
```

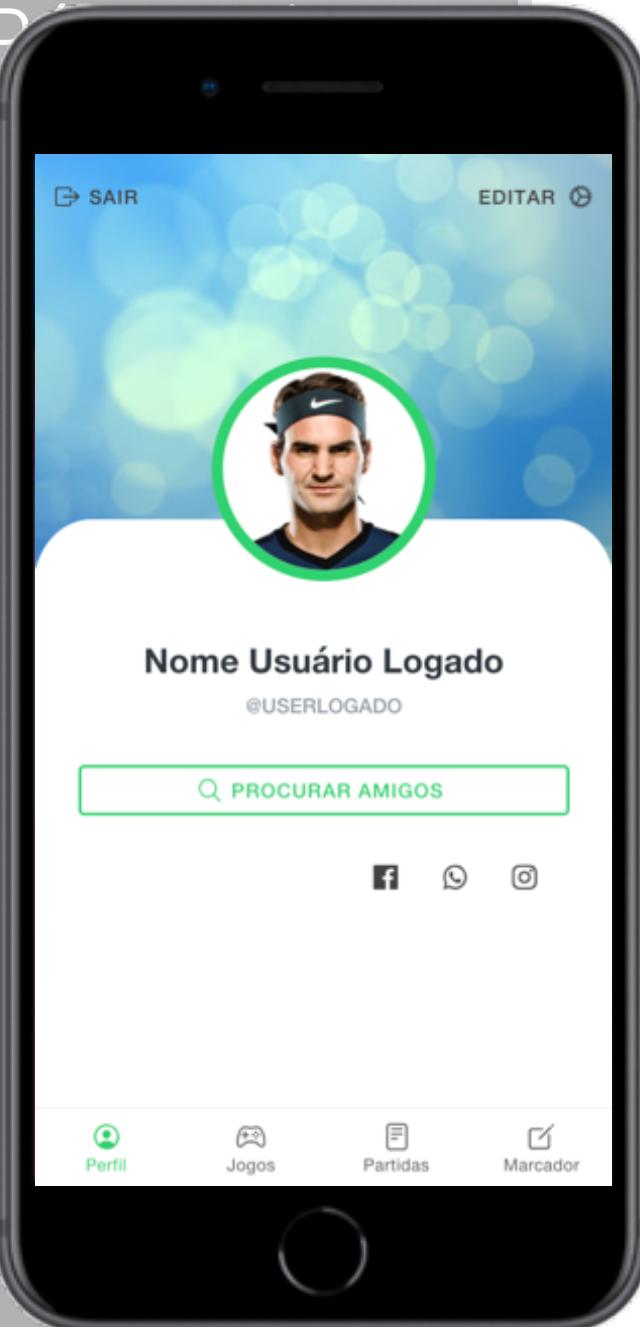
Adicionamos uma div para o "card", outra para o "header" e outra para o "avatar"  
Adicionamos a foto que deve ser inserida na pasta img do assets

Adicionamos uma div para o "card-body", outra para o "user-meta"

Apresentamos o nome do usuário e o apelido

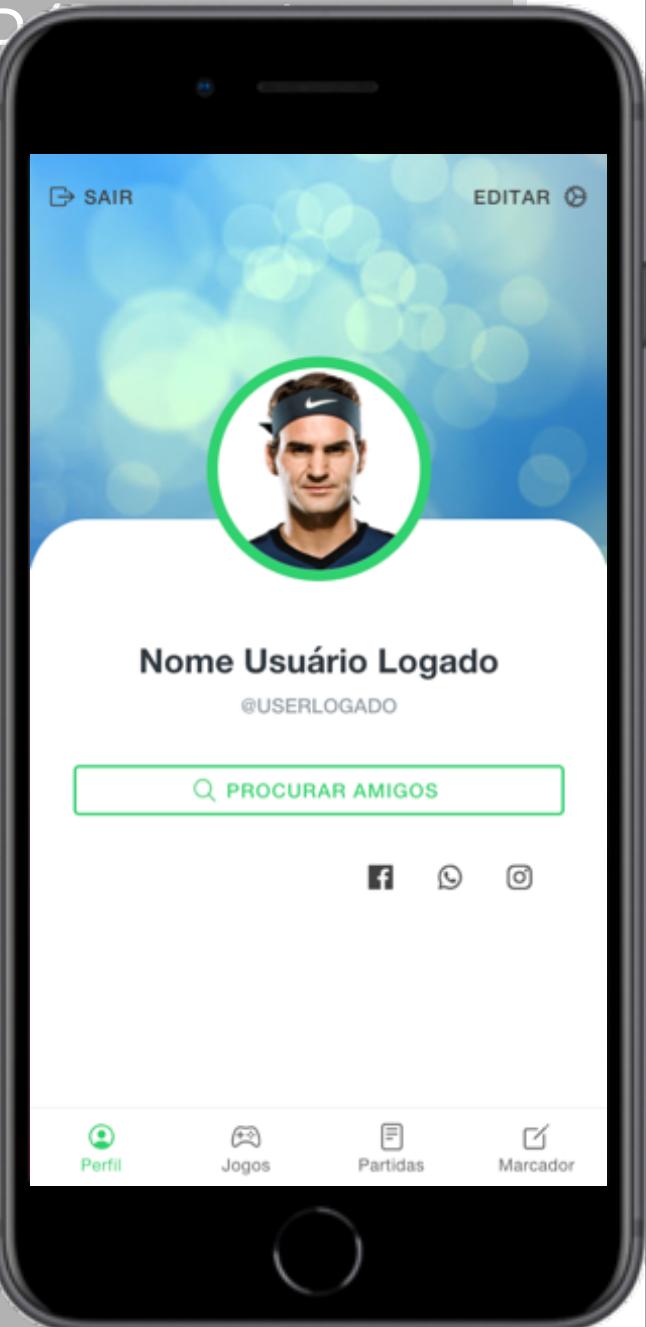


```
perfil.page.html
PlayingScore > src > app > perfil > perfil.page.html > ion-content > ion-toolbar > ion-content
1 / 55
18   <div class="card">
19     <div class="header">
20       <div class="avatar">
21         
22       </div>
23     </div>
24
25     <div class="card-body">
26
27       <div class="user-meta ion-text-center">
28         <h3 class="username">Nome Usuário Logado</h3>
29
30         <div class="card-body">
31
32           <div class="user-meta ion-text-center">
33             <h3 class="username">Nome Usuário Logado</h3>
34             <h5 class="nickname">@userlogado</h5>
35           </div>
36
37           <ion-button class="bt-sociais">
38             <ion-icon name="logo-whatsapp"></ion-icon>
39           </ion-button>
40
41           <ion-button class="bt-sociais">
42             <ion-icon name="logo-instagram"></ion-icon>
43           </ion-button>
44         </ion-buttons>
45       </div>
46     </div>
47   </ion-content>
```



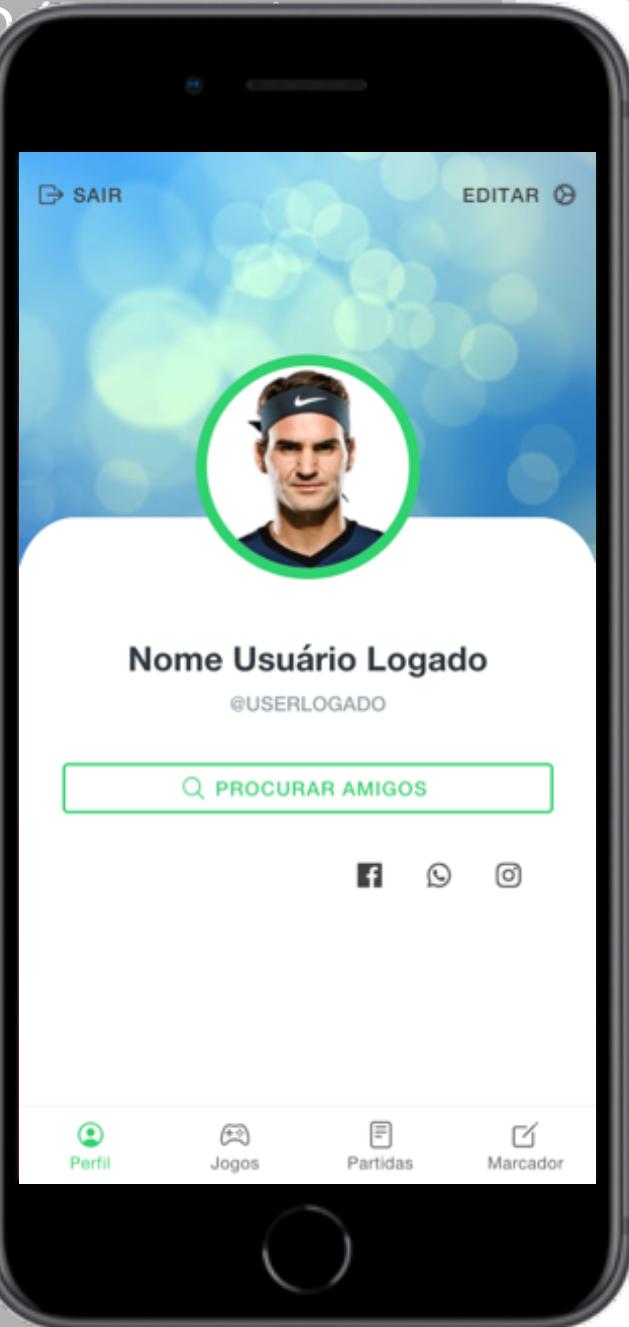
```
perfil.page.html
PlayingScore > src > app > perfil > perfil.page.html > ion-content > ion-toolbar > ion-
1/   18  <div class="card">
19    <div class="header">
20      <div class="avatar">
21        
22      </div>
23    </div>
24
25    <div class="card-body">
26
27      <div class="user-meta ion-text-center">
28        <h3 class="username">Nome Usuário Logado</h3>
29        <h5 class="nickname">@userlogado</h5>
30      </div>
31
32      <ion-button fill="outline" expand="block">
33        Procurar Amigos
34        <ion-icon slot="start" name="search-outline"></ion-icon>
35      </ion-button>
36
37      <ion-button class="bt-sociais">
38        <ion-icon name="logo-facebook"></ion-icon>
39      </ion-button>
40
41      <ion-button class="bt-sociais">
42        <ion-icon name="logo-whatsapp"></ion-icon>
43      </ion-button>
44
45      <ion-button class="bt-sociais">
46        <ion-icon name="logo-instagram"></ion-icon>
47      </ion-button>
48
49    </ion-buttons>
50
51  </ion-toolbar>
52
53  </div>
54
55 </ion-content>
```

Adicionamos o botão para pesquisa de amigos no aplicativo



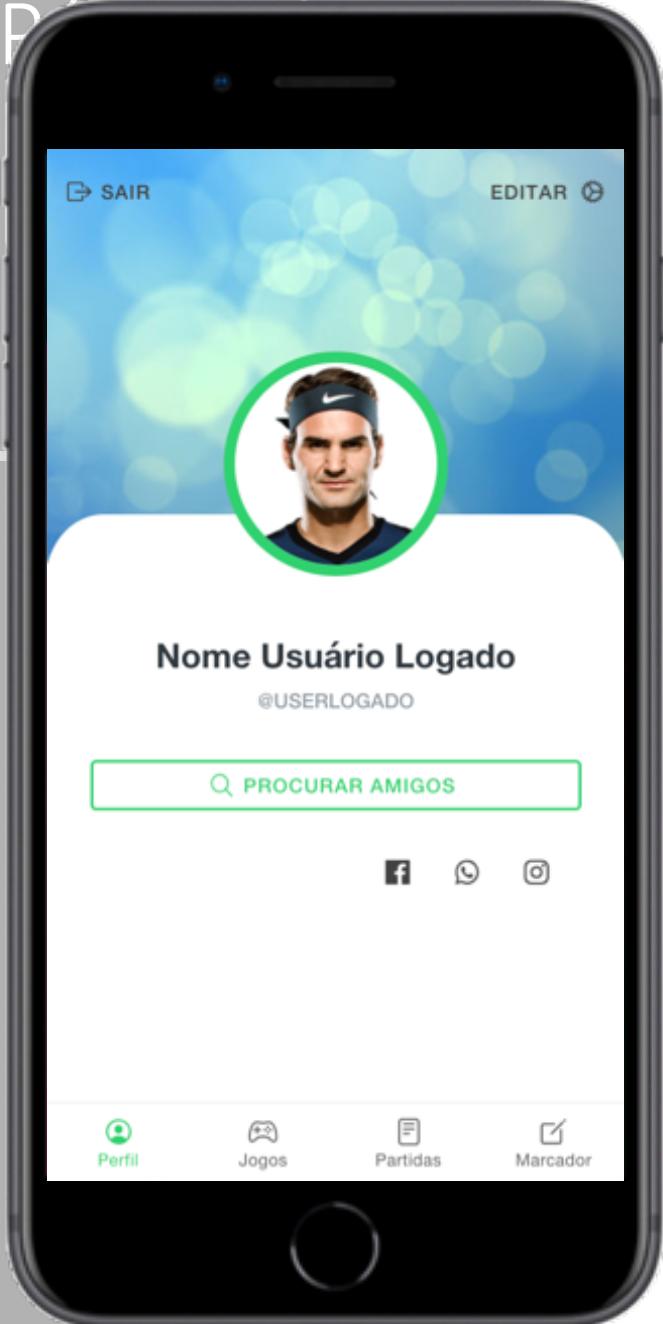
```
perfil.page.html
PlayingScore > src > app > perfil > perfil.page.html > ion-content > ion-toolbar > ion-content
1/ 18   <div class="card">
19     <div class="header">
20       <div class="avatar">
21         
22       </div>
23     </div>
24
25   <ion-toolbar>
26     <ion-buttons slot="secondary">
27       <ion-button class="bt-sociais">
28         <ion-icon name="logo-facebook"></ion-icon>
29       </ion-button>
30
31       <ion-button class="bt-sociais">
32         <ion-icon name="logo-whatsapp"></ion-icon>
33       </ion-button>
34
35       <ion-button class="bt-sociais">
36         <ion-icon name="logo-instagram"></ion-icon>
37       </ion-button>
38
39     </ion-buttons>
40   </ion-toolbar>
41
42 </div>
43
44 </div>
45
46
47 </div>
48
49 </div>
50
51 </div>
52
53 </div>
54
55 </ion-content>
```

Por fim, adicionamos a toolbar para acesso as redes sociais



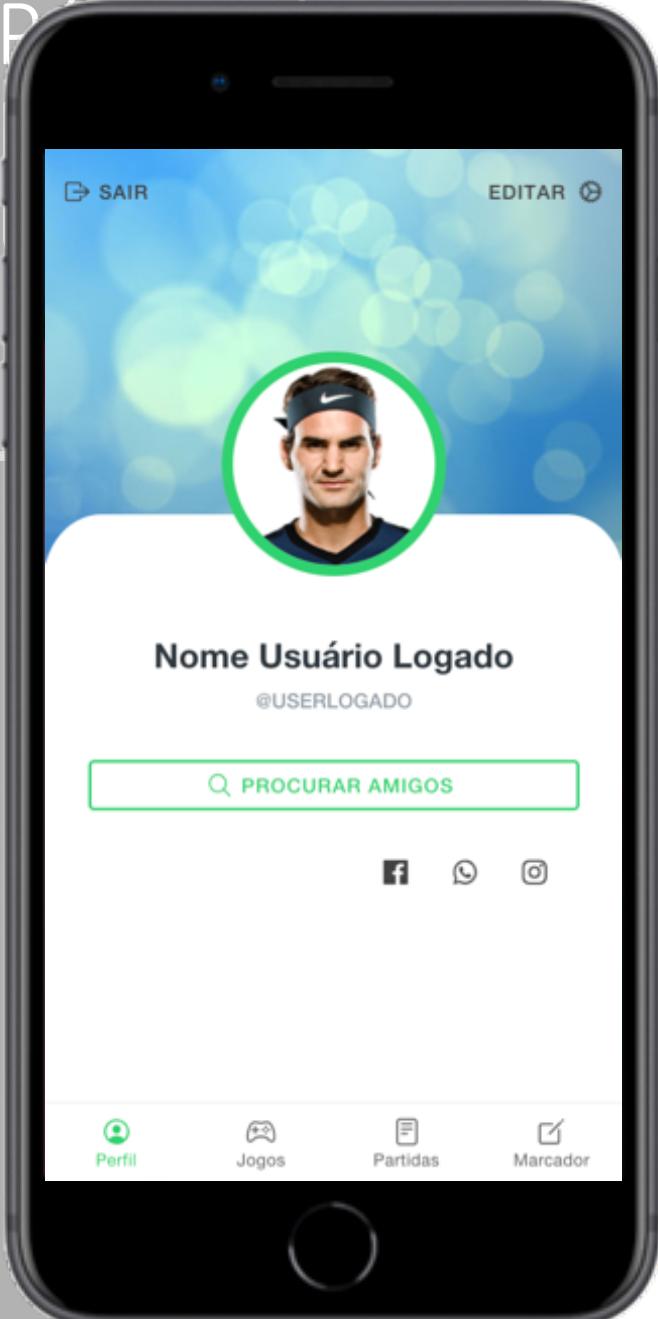
```
perfil.page.html X
PlayingScore > src > app > perfil > perfil.page.html > ion-content > ion-toolbar > ion-content
1 / 55
18 <div class="card">
19   <div class="header">
20     <div class="avatar">
21       
22     </div>
23   </div>
24
25   <div class="card-body">
26
27     <div class="user-meta ion-text-center">
28       <h3 class="username">Nome Usuário Logado</h3>
29       <h5 class="nickname">@userlogado</h5>
30     </div>
31
32     <ion-button fill="outline" expand="block">
33       Procurar Amigos
34       <ion-icon slot="start" name="search-outline"></ion-icon>
35     </ion-button>
36
37     <ion-toolbar>
38       <ion-buttons slot="secondary">
39         <ion-button class="bt-sociais">
40           <ion-icon name="logo-facebook"></ion-icon>
41         </ion-button>
42
43         <ion-button class="bt-sociais">
44           <ion-icon name="logo-whatsapp"></ion-icon>
45         </ion-button>
46
47         <ion-button class="bt-sociais">
48           <ion-icon name="logo-instagram"></ion-icon>
49         </ion-button>
50       </ion-buttons>
51     </ion-toolbar>
52
53   </div>
54 </div>
55 </ion-content>
```

Agora, vamos projetar a página de perfil de usuário



Agora, vamos organizar o arquivo SCSS

Agora, vamos projetar a página de perfil de usuário



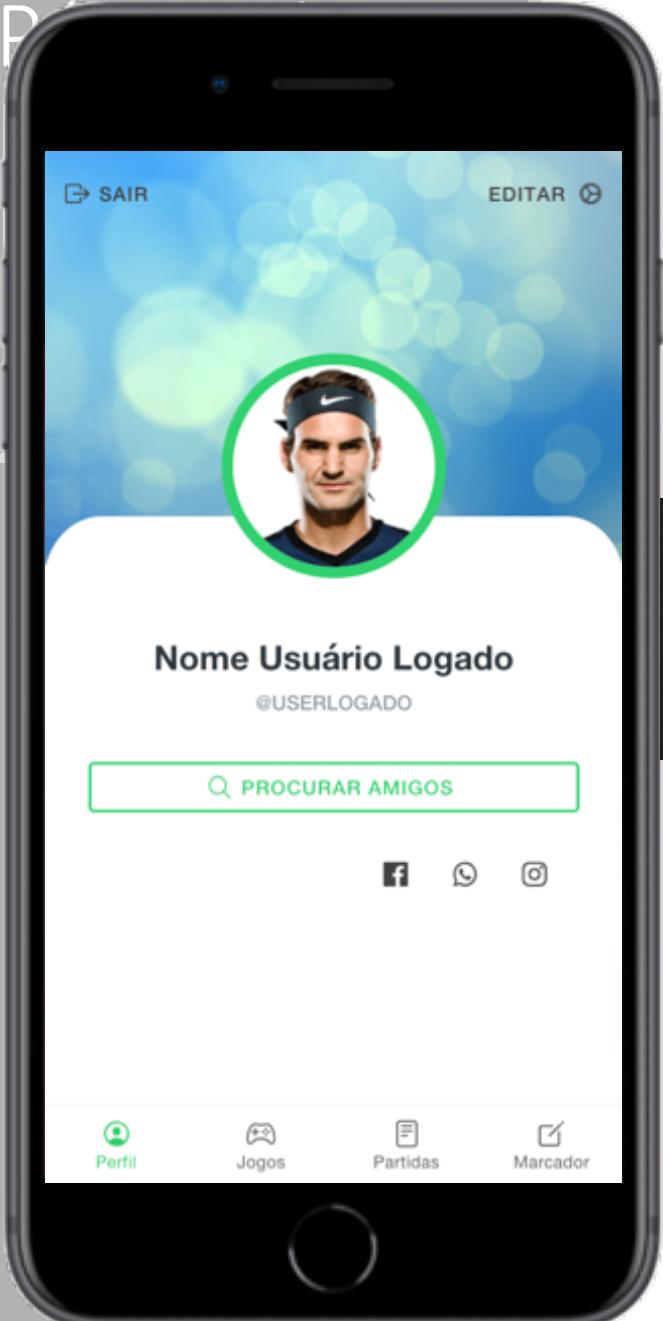
Agora, vamos organizar o arquivo SCSS

Criamos o estilo principal do conteúdo, com imagem de fundo

perfil.page.scss ×

```
PlayingScore > src > app > perfil > perfil.page.scss > .card
1 ion-content {
2   --background: url(../../assets/img/background_full.jpg) no-repeat center center fixed, #fff;
3   position: relative;
4   height: 100%;
5   width: 100%;
6 }
```

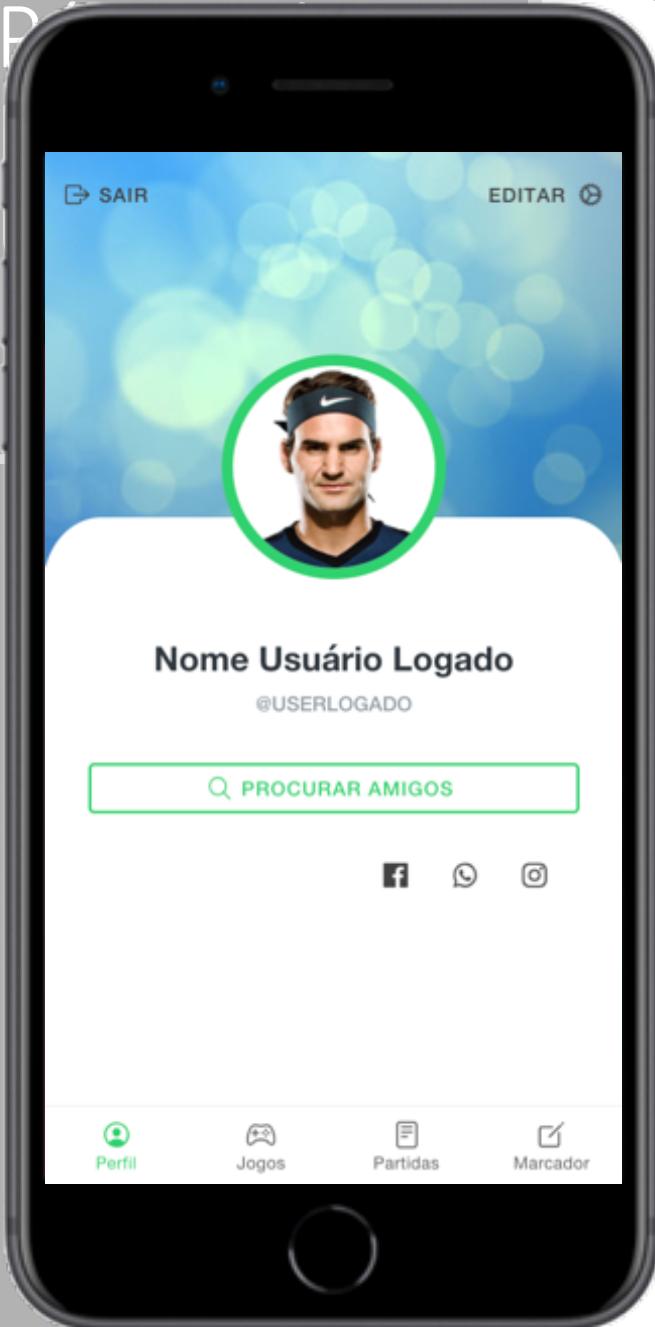
Agora, vamos projetar a página de perfil de usuário



Agora, vamos organizar o arquivo SCSS

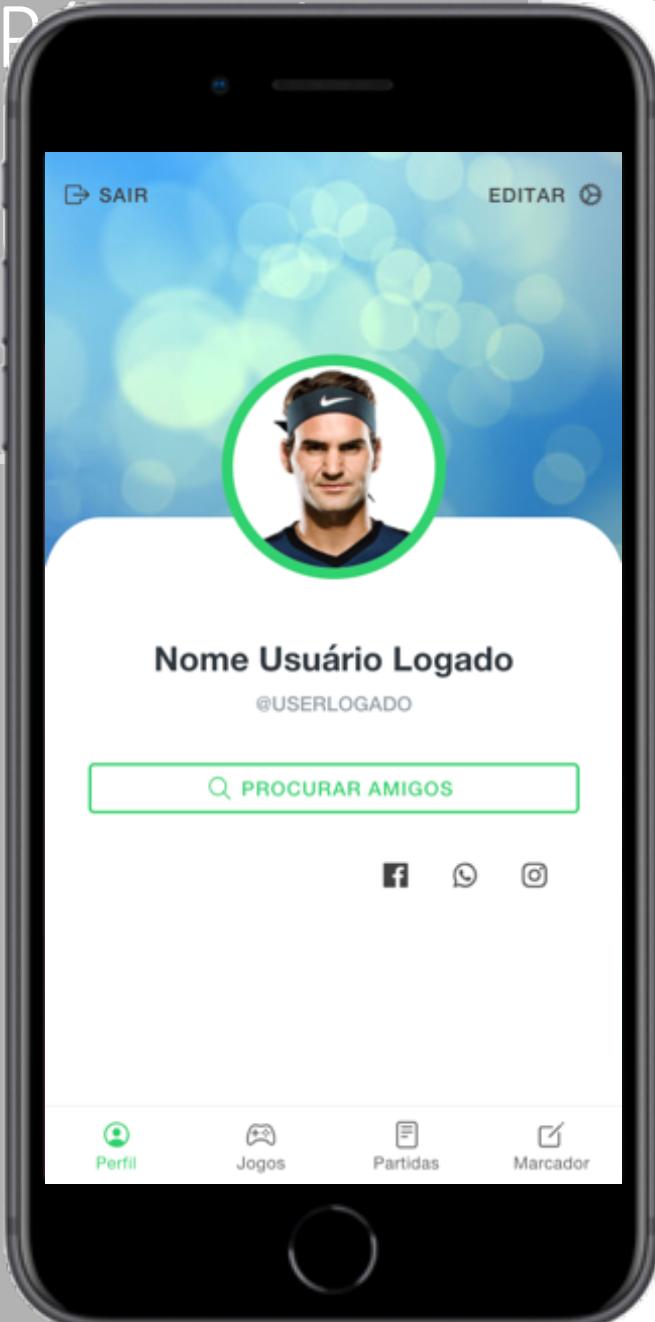
Configuramos transparência no fundo das toolbars

```
8   ion-toolbar {  
9     --background: transparent;  
10 }
```



```
12 .card {  
13     margin: 0 auto;  
14     border-radius: 10px;  
15 }  
16 .header {  
17     height: 200px;  
18 }  
19 .avatar {  
20     width: 160px;  
21     height: 160px;  
22     position: relative;  
23     margin: 0 auto;  
24 }  
25 .img {  
26     display: block;  
27     border-radius: 50%;  
28     position: absolute;  
29     bottom: calc(-1*(80px + 4px));  
30     border: 8px solid #30d270;  
31     background-color: #fff;  
32 }  
33 }  
34 }  
35 }
```

Configuramos o card e todos  
seus "filhos"

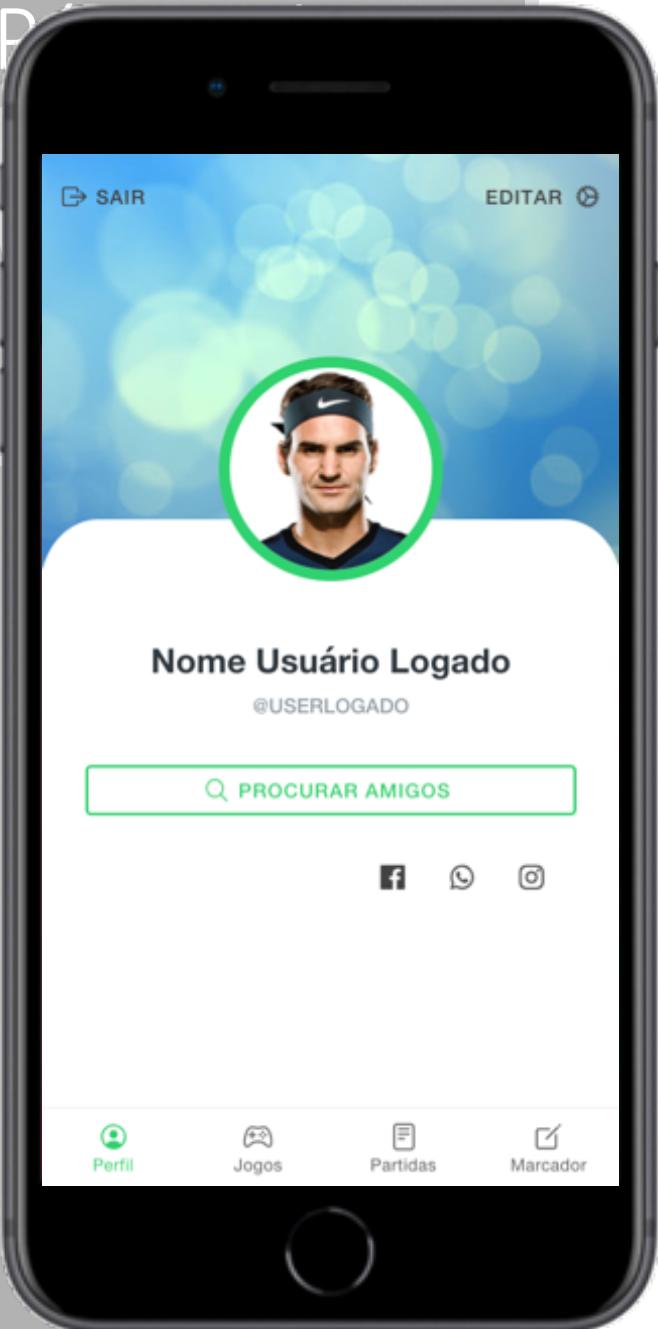


```
usuário
37 .card-body {
38     background-color: #ffffff;
39     padding: 30px;
40     height: calc(100vh - (200px + 56px));
41     overflow: hidden;
42
43     border-radius: 10%;
44
45     .user-meta {
46         padding-top: 40px;
47
48         .username {
49             font-size: 24px;
50             font-weight: 600;
51             color: #303940;
52         }
53
54         .nickname {
55             font-size: 90%;
56             color: #949ea6;
57             text-transform: uppercase;
58             margin: 0 auto;
59             padding-bottom: 20px;
60         }
61     }
62 }
63
64 .bt-sociais {
65     padding-right: 10px;
66 }
```

usuário

Configuramos o card-body e todos seus "filhos"

Também, os botões de redes sociais



Server Side

# Server Side

---

Vamos montar o servidor de acesso a dados com NodeJS e Express

# Server Side

---

O que é o Node?

É um framework javascript feito para executar no lado do servidor da aplicação,  
famoso backend

# Server Side

---

O que é o Express?

É um framework javascript, do NodeJS, feito para trabalharmos com APIs

# Server Side

---

Também, trabalharemos com TypeScript e TypeORM

- ORM → Object Relational Mapping
- Precisamos instalar o ts-node para fazer uso do TypeORM

Para instalarmos o TypeORM CLI, vamos executar no terminal

→ sudo npm install --global ts-node

```
[→ OnVestAI git:(master) ✘ sudo npm install --global ts-node
/usr/local/bin/ts-script -> /usr/local/lib/node_modules/ts-node/dist/bin-script-deprecated.js
/usr/local/bin/ts-node -> /usr/local/lib/node_modules/ts-node/dist/bin.js
/usr/local/bin/ts-node-transpile-only -> /usr/local/lib/node_modules/ts-node/dist/bin-transpile.js
/usr/local/bin/ts-node-script -> /usr/local/lib/node_modules/ts-node/dist/bin-script.js
npm WARN ts-node@8.10.2 requires a peer of typescript@>=2.7 but none is installed. You must install peer dependencies yourself.

+ ts-node@8.10.2
added 8 packages from 40 contributors in 2.635s
```

# Server Side

---

Também, trabalharemos com TypeScript e TypeORM

- ORM → Object Relational Mapping
- Precisamos instalar o ts-node para fazer uso do TypeORM

Para instalarmos o TypeORM, vamos executar no terminal

→ sudo npm install --global typeorm

```
[→ OnVestAI git:(master) ✘ sudo npm install --global typeorm
/usr/local/bin/typeorm -> /usr/local/lib/node_modules/typeorm/cli.js
+ typeorm@0.2.25
added 103 packages from 361 contributors in 9.74s
```

# Server Side

---

Também, trabalharemos com TypeScript e TypeORM

- ORM → Object Relational Mapping
- Precisamos instalar o ts-node para fazer uso do TypeORM

Agora, vamos criar o projeto da API

- Então, vá até a pasta do nosso projeto (PlayingScore) e execute o comando de criação  
→ `typeorm init --name ps.api --express --database mysql`

```
[→ PlayingScore git:(master) ✘ typeorm init --name ps.api --express --database mysql
Project created inside /Users/madalozzo/Dropbox/1-UPF/2020/Semestre_1/POS-Ionic/Dev/PlayingScore/ps.api directory.
```

# Server Side

---

O projeto da API está criado com esta estrutura

```
└─ ps.api
    ├─ src
    └─ .gitignore
    { } ormconfig.json
    { } package.json
    ⓘ README.md
    { } tsconfig.json
```

# Server Side

O projeto da API está criado com esta estrutura

The screenshot shows a file explorer interface with a dark theme. On the left, there is a tree view of files and folders:

- ps.api (selected)
- src
- .gitignore
- ormconfig.json
- package.json (highlighted with a blue border)
- README.md
- tsconfig.json

To the right of the file list, a large blue callout box points to the "package.json" file with the following text:

Arquivo para configurar os pacotes do projeto

# Server Side

O projeto da API está criado com esta estrutura

```
└ ps.api
  └ src
  └ .gitignore
  └ ormconfig.json
  └ package.json
  └ README.md
  └ tsconfig.json
```

Arquivo para configurar os as libs  
e módulos do projeto

# Server Side

O projeto da API está criado com esta estrutura

```
▽ ps.api
  > src
  ◆ .gitignore
  {} ormconfig.json
  {} package.json
  ⓘ README.md
  {} tsconfig.json
```

Apresenta os próximos passos  
depois da criação do projeto

# Server Side

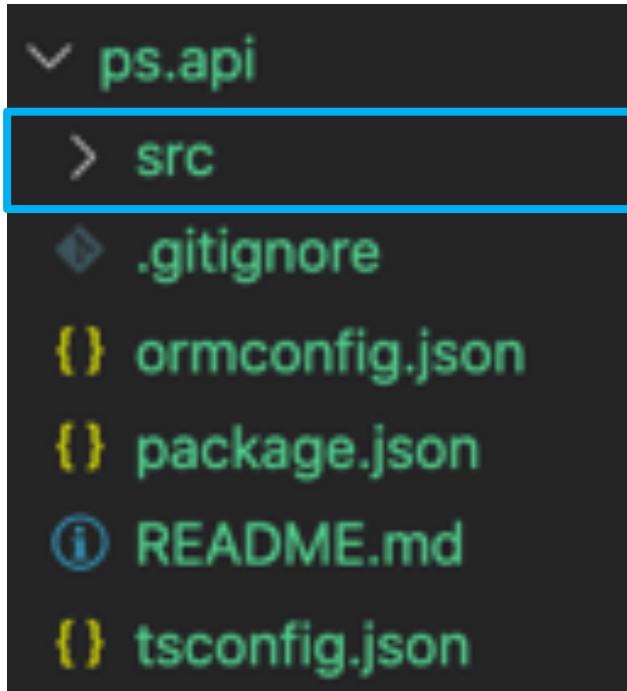
O projeto da API está criado com esta estrutura

```
▽ ps.api
  > src
  ◆ .gitignore
  { ormconfig.json
  { package.json
  ⓘ README.md
  { tsconfig.json
```

Possui todas as configurações de acesso ao banco de dados

# Server Side

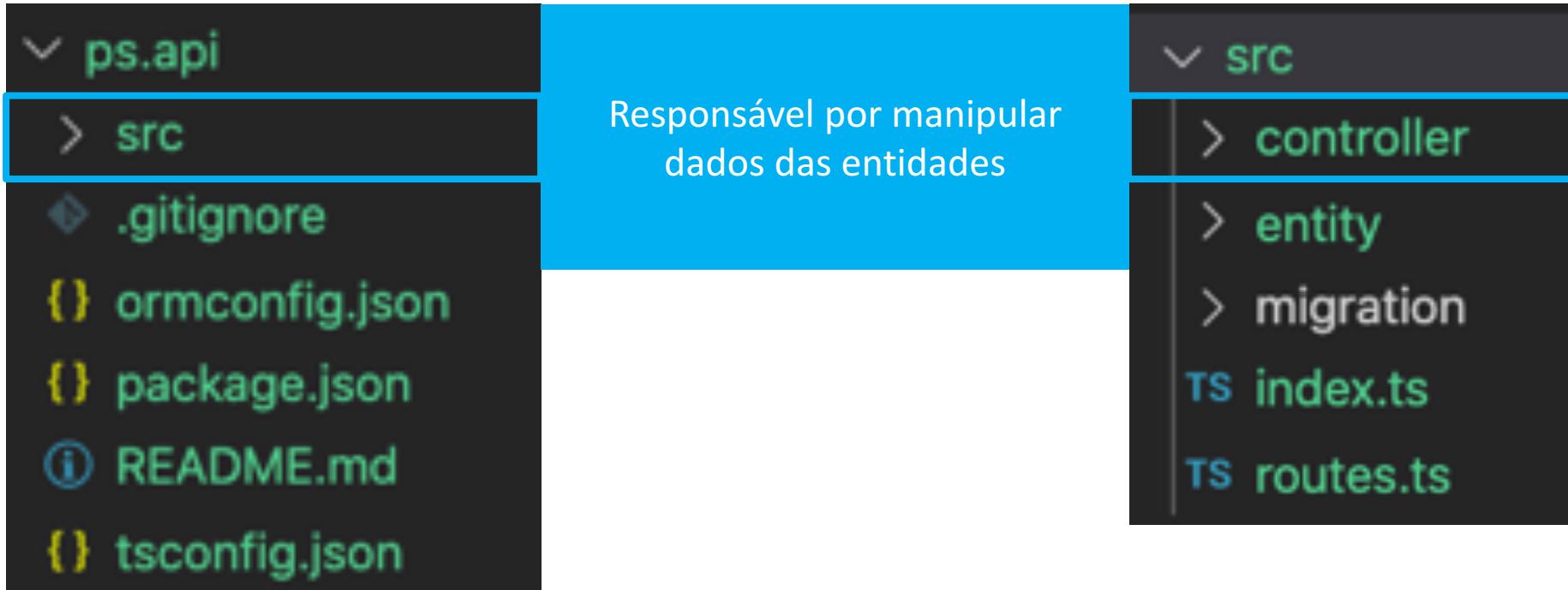
O projeto da API está criado com esta estrutura



Área de desenvolvimento de  
nossa projeto

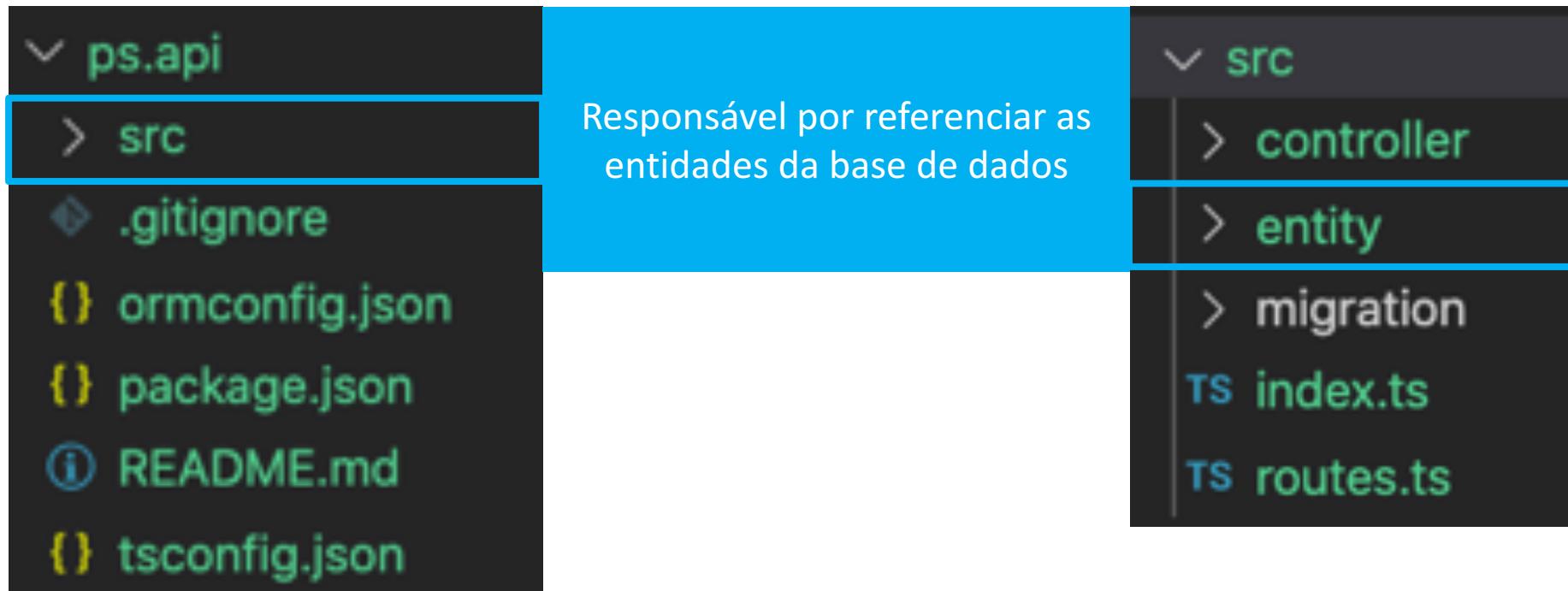
# Server Side

O projeto da API está criado com esta estrutura



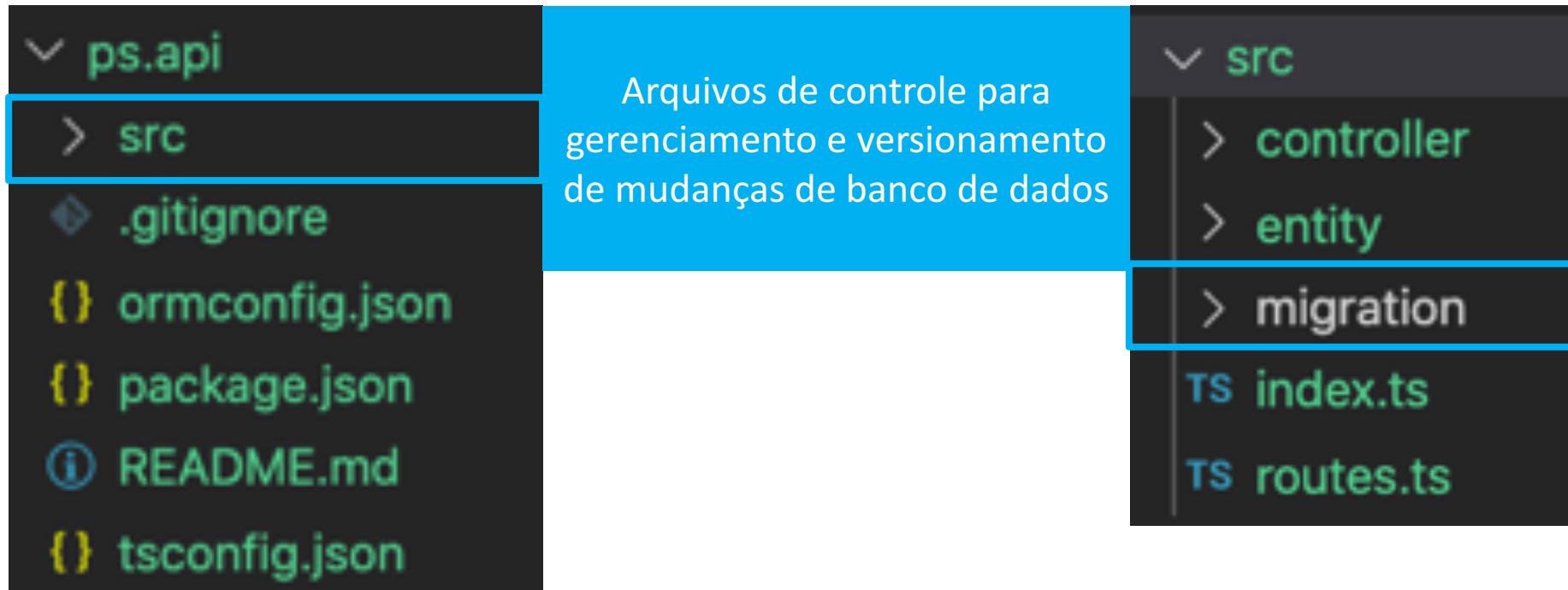
# Server Side

O projeto da API está criado com esta estrutura



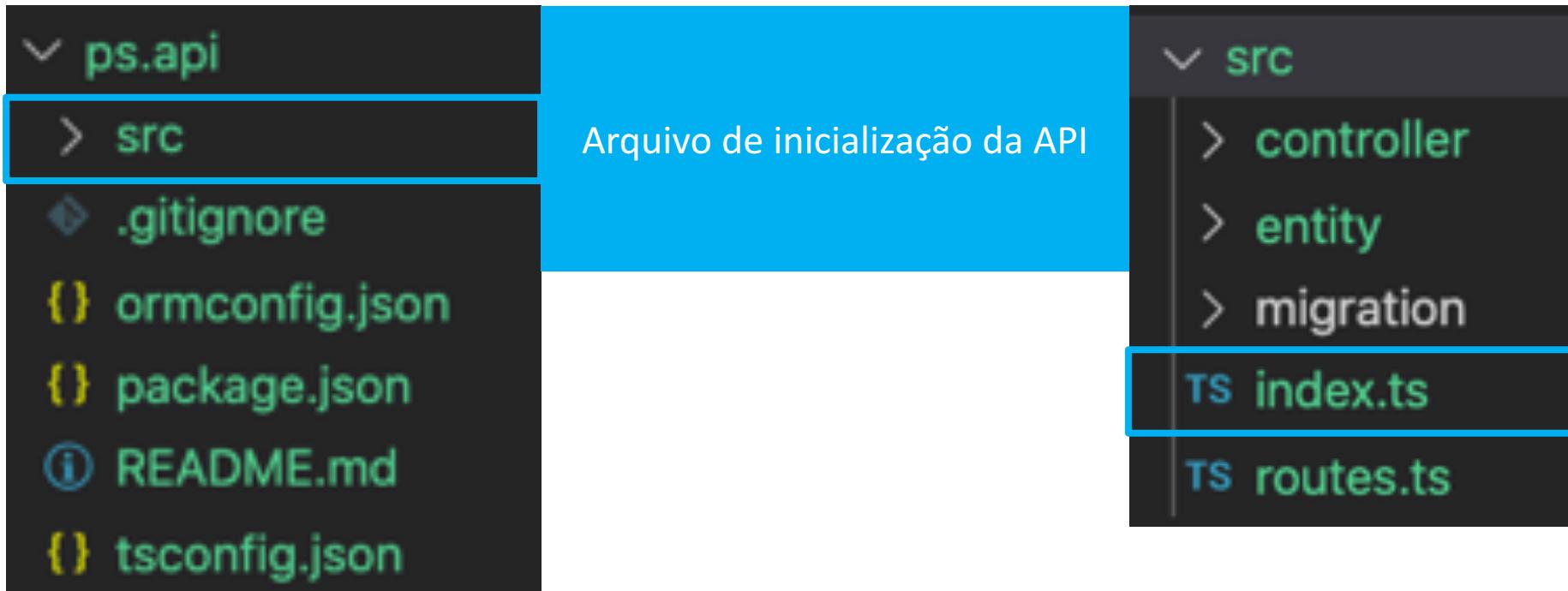
# Server Side

O projeto da API está criado com esta estrutura



# Server Side

O projeto da API está criado com esta estrutura



# Server Side

O projeto da API está criado com esta estrutura

The screenshot shows a file explorer interface with two main sections. On the left, a dark sidebar lists the root directory contents: ps.api, src, .gitignore, ormconfig.json, package.json, README.md, and tsconfig.json. The 'src' folder is highlighted with a blue border. On the right, a light sidebar shows the contents of the src directory: controller, entity, migration, index.ts, and routes.ts. The 'routes.ts' file is also highlighted with a blue border. A central white area contains the text 'Os caminhos disponíveis pela API'.

```
ps.api
  src
  .gitignore
  ormconfig.json
  package.json
  README.md
  tsconfig.json

  src
    controller
    entity
    migration
    index.ts
    routes.ts

  Os caminhos disponíveis pela API
```

# Server Side

Para executar a API, precisamos seguir os passos apresentados no readme

## ① README.md ×

```
PlayingScore > ps.api > ① README.md > □ # Awesome Project Build with
1   # Awesome Project Build with TypeORM
2
3   Steps to run this project:
4
5   1. Run `npm i` command
6   2. Setup database settings inside `ormconfig.json` file
7   3. Run `npm start` command
```

# Server Side

Para executar a API, precisamos seguir os passos apresentados no

O comando "npm i" instalará todas as dependências do projeto

## ① README.md ×

```
PlayingScore > ps.api > ① README.md > ┌ # Awesome Project Build with
  1   # Awesome Project Build with TypeORM
  2
  3   Steps to run this project:
  4
  5   1. Run `npm i` command
  6   2. Setup database settings inside `ormconfig.json` file
  7   3. Run `npm start` command
```

# Server Side

Para executar a API, precisamos seguir os passos apresentados no

O comando "npm i" instalará todas as dependências do projeto

## ① README.md X

```
[→ ps.api git:(master) ✘ npm i
npm notice created a lockfile as package-lock.json. You should commit this file.
npm WARN ps.api@0.0.1 No repository field.
npm WARN ps.api@0.0.1 No license field.

added 177 packages from 471 contributors and audited 179 packages in 15.899s

3 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

7 3. Run `npm start` command

# Server Side

Para executar a API, precisamos seguir os passos apresentados no

Vamos configurar os dados de  
acesso ao banco

## ① README.md ×

```
PlayingScore > ps.api > ① README.md > ② # Awesome Project Build with
  1   # Awesome Project Build with TypeORM
  2
  3   Steps to run this project:
  4
  5   1. Run `npm i` command
  6   2. Setup database settings inside `ormconfig.json` file
  7   3. Run `npm start` command
```

# Server Side

Para executar a A

① README

PlayingScore

1 # A

2 St

3

4

5 1.

6 2.

7 3.

{} ormconfig.json ×

PlayingScore > ps.api > {} ormconfig.json > ...

```
1 [
2   "type": "mysql",
3   "host": "localhost",
4   "port": 3306,
5   "username": "root",
6   "password": "MINHASENHA",
7   "database": "playing_score",
8   "synchronize": true,
9   "logging": false
```

Vamos configurar os dados de  
ao banco

ject Build with

.json` file

# Server Side

Para executar a API, precisamos seguir os passos apresentados no

Vamos configurar os dados de acesso ao banco

① README.md X

PlayingScore > ps.api > ① README.md > # Awesome Project Build with



Schema Name:

playing\_score

Rename References

Default Collation:

utf8 - utf8\_general\_ci

# Server Side

Para executar a API, precisamos seguir os passos apresentados no

Vamos configurar os dados de  
acesso ao banco

## ① README.md ×

```
PlayingScore > ps.api > ① README.md > ② # Awesome Project Build with
  1   # Awesome Project Build with TypeORM
  2
  3   Steps to run this project:
  4
  5   1. Run `npm i` command
  6   2. Setup database settings inside `ormconfig.json` file
  7   3. Run `npm start` command
```

# Server Side

Para executar a API, precisamos seguir os passos apresentados no

Vamos configurar os dados de acesso ao banco

① README.md ×

PlayingScore > ps.api > ① README.md > ② # Awesome Project Build with  
1    **# Awesome Project Build with TypeORM**  
2

[→ ps.api git:(master) ✘ npm start

> ps.api@0.0.1 start /Users/madalozzo/Dropbox/1-UPF/2020/Semestre\_1/POS-Ionic/Dev/PlayingScore(ps.api  
> ts-node src/index.ts

Express server has started on port 3000. Open <http://localhost:3000/users> to see results

# Server

Para executar a

A screenshot of a web browser window displaying JSON data. The address bar shows "localhost:3000/users". The browser interface includes standard navigation buttons (back, forward, refresh) and a search bar with the same URL. Below the address bar, there are tabs for "JSON", "Raw Data", and "Headers", with "JSON" being the active tab. A toolbar below the tabs includes "Save", "Copy", "Collapse All", "Expand All", and a "Filter JSON" input field. The main content area displays two objects, indexed 0 and 1, each with properties: id, firstName, lastName, and age.

```
0:
  id: 1
  firstName: "Timber"
  lastName: "Saw"
  age: 27
1:
  id: 2
  firstName: "Phantom"
  lastName: "Assassin"
  age: 24
```

# Server

Para executar a

A screenshot of a web browser window titled "localhost:3000/users". The address bar also shows "localhost:3000/users". The page content displays a JSON array of two user objects. The first object (index 0) has id 1, firstName "Timber", lastName "Saw", and age 27. The second object (index 1) has id 2, firstName "Phantom", lastName "Assassin", and age 24.

	id	firstName	lastName	age
0:	1	"Timber"	"Saw"	27
1:	2	"Phantom"	"Assassin"	24

Nosso projeto está funcional

Insomnia - PlayingScore

Send 200 OK 123 ms 119 B 37 Minutes Ago

No Environment Cookies

Body Auth Query Header Docs

Filter

GET PlayingScore

Preview Header Cookie Timeline

```
1< 1
2< 2
3<   {
4<     "id": 1,
5<     "FirstName": "Eduardo",
6<     "LastName": "Silva",
7<     "Age": 23
8<   },
9<   {
10<     "id": 2,
11<     "FirstName": "Phantom",
12<     "LastName": "Nassarini",
13<     "Age": 24
14<   }
15<
```

Select a body type from above

6.1 Home /books [x] search

Apenas como dica, utilize o Insomnia para testar as rotas em desenvolvimento

# Server Side

Para executar a API, precisamos seguir os passos apresentados no readme

```
TS index.ts  X

PlayingScore > ps.api > src > TS index.ts > ...

30
31     // start express server
32     app.listen(3000);
33
34     // insert new users for test
35     await connection.manager.save(connection.manager.create(User, {
36         firstName: "Timber",
37         lastName: "Saw",
38         age: 27
39     }));
40     await connection.manager.save(connection.manager.create(User, {
41         firstName: "Phantom",
42         lastName: "Assassin",
43         age: 24
44     }));
45
46     console.log("Express server has started on port 3000. Open http://localhost:3000/users to see results");
47
48 }).catch(error => console.log(error));
```

Isso ocorre pois, entre tantas outras coisas, o index.ts cria dois usuários na inicialização

# Server Side

---

Agora vamos configurar o index.ts conforme nossas necessidades

- Primeiro, vamos fazer uma limpa e deixar apenas o que nos é útil

# Server Side

Agora vamos configurar o index.ts conforme nossas necessidades

- Primeiro, vamos fazer uma limpa e deixar apenas o que nos é útil

```
ts index.ts  x

PlayingScore > ps.api > src > ts index.ts > ...
1 import "reflect-metadata";
2 import {createConnection} from "typeorm";
3 import * as express from "express";
4 import * as bodyParser from "body-parser";
5 import {Request, Response} from "express";
6 import {Routes} from "./routes";
7
8 // create express app
9 const app = express();
10 app.use(bodyParser.json());
11
12 // register express routes from defined application routes
13 Routes.forEach(route => {
14     (app as any)[route.method](route.route, (req: Request, res: Response, next: Function) => {
15         const result = (new (route.controller as any))[route.action](req, res, next);
16         if (result instanceof Promise) {
17             result.then(result => result !== null && result !== undefined ? res.send(result) : undefined);
18
19         } else if (result !== null && result !== undefined) {
20             res.json(result);
21         }
22     });
23 });

});
```

# Server Side

Agora vamos configurar o index.ts conforme nossas necessidades

- Primeiro, vamos fazer uma limpa e deixar apenas o que nos é útil

ts index.ts

```
PlayingScore > ps.api > src > ts index.ts > ...
1 import "reflect-metadata";
2 import {createConnection} from "typeorm";
3 import * as express from "express";
4 import * as bodyParser from "body-parser";
5 import {Request, Response} from "express";
6 import {Routes} from "./routes";
7
8 // create express app
9 const app = express();
10 app.use(bodyParser.json());
11
12 // register express routes from defined application routes
13 Routes.forEach(route => {
14     (app as any)[route.method](route.route, (req: Request, res: Response, next: Function) => {
15         const result = (new (route.controller as any))[route.action](req, res, next);
16         if (result instanceof Promise) {
17             result.then(result => result !== null && result !== undefined ? res.send(result) : undefined);
18
19         } else if (result !== null && result !== undefined) {
20             res.json(result);
21         }
22     });
23 });

Criamos um app com base no express
```

# Server Side

Agora vamos configurar o index.ts conforme nossas necessidades

- Primeiro, vamos fazer uma limpa e deixar apenas o que nos é útil

ts index.ts ×

```
PlayingScore > ps.api > src > ts index.ts > ...
1 import "reflect-metadata";
2 import {createConnection} from "typeorm";
3 import * as express from "express";
4 import * as bodyParser from "body-parser";
5 import {Request, Response} from "express";
6 import {Routes} from "./routes";
7
8 // create express app
9 const app = express();
10 app.use(bodyParser.json());
11
12 // register express routes from defined application routes
13 Routes.forEach(route => {
14     (app as any)[route.method](route.route, (req: Request, res: Response, next: Function) => {
15         const result = (new (route.controller as any))[route.action](req, res, next);
16         if (result instanceof Promise) {
17             result.then(result => result !== null && result !== undefined ? res.send(result) : undefined);
18
19         } else if (result !== null && result !== undefined) {
20             res.json(result);
21         }
22     });
23 });


```

Mantemos o controlador de rotas do express

# Server Side

Agora vamos configurar o index.ts conforme nossas necessidades

- Primeiro, vamos fazer uma limpa e deixar apenas o que nos é útil

ts index.ts

```
PlayingScore > ps.api > src > ts index.ts > ...
1 import "reflect-metadata";
2 import {createConnection} from "typeorm";
3 import * as express from "express";
4 import * as bodyParser from "body-parser";
5 import {Request, Response} from "express";
6 import {Routes} from "./routes";
7
8 // create express app
9 const app = express();
10 app.use(bodyParser.json());
11
12 // register express routes from defined application routes
13 Routes.forEach(route => {
14     (app as any)[route.method](route.route, (req: Request, res: Response, next: Function) => {
15         const result = (new (route.controller as any))[route.action](req, res, next);
16         if (result instanceof Promise) {
17             result.then(result => result !== null && result !== undefined ? res.send(result) : undefined);
18
19         } else if (result !== null && result !== undefined) {
20             res.json(result);
21         }
22     });
23 });

});
```

Este controlador de rotas é genérico, podemos passar a rota que quisermos que ele irá entender

# Server Side

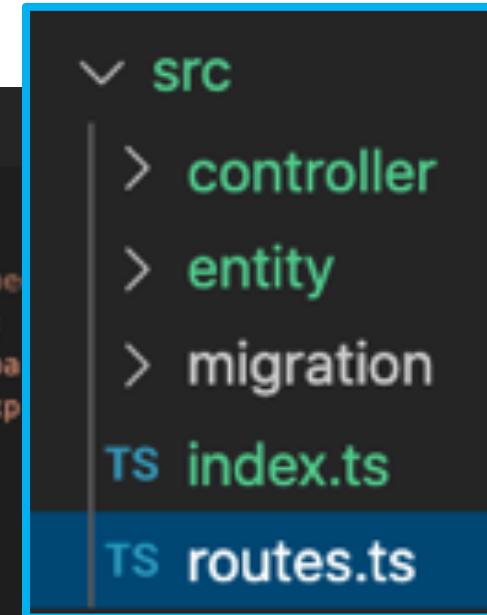
Agora vamos configurar o index.ts conforme nossas necessidades

- Primeiro, vamos fazer uma limpa e deixar apenas o que nos é útil

```
TS index.ts  x

PlayingScore > ps.api > src > TS index.ts > ...
1  import "reflect-metadata";
2  import {createConnection} from "typeorm";
3  import * as express from "express";
4  import * as bodyParser from "body-parser";
5  import {Request, Response} from "express";
6  import {Routes} from "./routes";
7
8  // create express app
9  const app = express();
10 app.use(bodyParser.json());
11
12 // register express routes from defined application routes
13 Routes.forEach(route => {
14   (app as any)[route.method](route.route, (req: Request, res: Response, next: Function) => {
15     const result = (new (route.controller as any))[route.action](req, res, next);
16     if (result instanceof Promise) {
17       result.then(result => result !== null && result !== undefined ? res.send(result) : undefined);
18
19     } else if (result !== null && result !== undefined) {
20       res.json(result);
21     }
22   });
23 });


```



Isso facilita o entendimento das rotas, uma vez que poderemos, então, utilizar um arquivo específico para isso

Agora vamos configurar o index.ts conforme nossas necessidades

- Primeiro, vamos fazer uma limpa e deixar apenas o que nos é útil

TS routes.ts ×

```
PlayingScore > ps.api > src > TS routes.ts > ...
1   import {UserController} from "./controller/UserController";
2
3   export const Routes = [
4     {
5       method: "get",
6       route: "/users",
7       controller: UserController,
8       action: "all"
9     },
10    {
11      method: "get",
12      route: "/users/:id",
13      controller: UserController,
14      action: "one"
15    },
16    {
17      method: "post",
18      route: "/users",
19      controller: UserController,
20      action: "save"
21    },
22    {
23      method: "delete",
24      route: "/users/:id",
25      controller: UserController,
26      action: "remove"
27    }
28  ];
```

✓ SRC

```
> controller
> entity
> migration
TS index.ts
TS routes.ts
```

ined application routes

```
e.route, (req: Request, res: Response, next: Function) => {
  controller as any))[route.action](req, res, next);
  se) {
    result !== null && result !== undefined ? res.send(result) : undefined);
  && result !== undefined) {
```

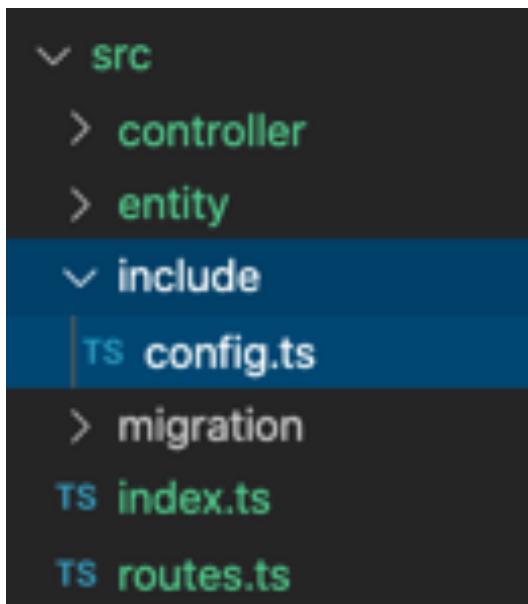
Isso facilita o entendimento das rotas, uma vez que poderemos, então, utilizar um arquivo específico para isso

# Server Side

---

Agora vamos configurar o index.ts conforme nossas necessidades

- Primeiro, vamos fazer uma limpa e deixar apenas o que nos é útil
- Agora, vamos criar um arquivo de configuração do ambiente → include/config.ts

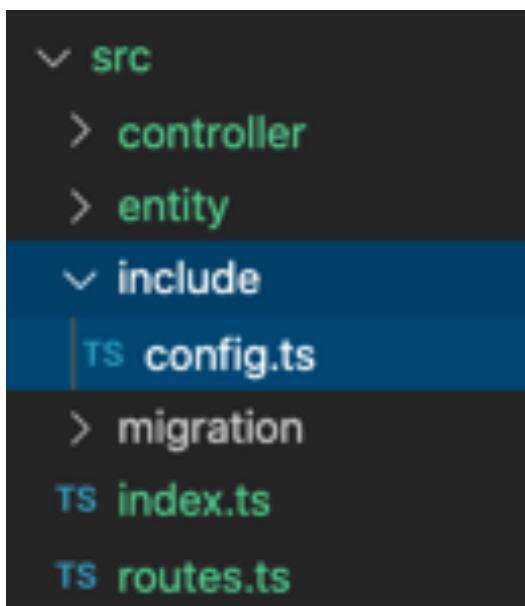


```
src
  controller
  entity
  include
    config.ts
  migration
  index.ts
  routes.ts
```

# Server Side

Agora vamos configurar o index.ts conforme nossas necessidades

- Primeiro, vamos fazer uma limpa e deixar apenas o que nos é útil
- Agora, vamos criar um arquivo de configuração do ambiente → include/config.ts



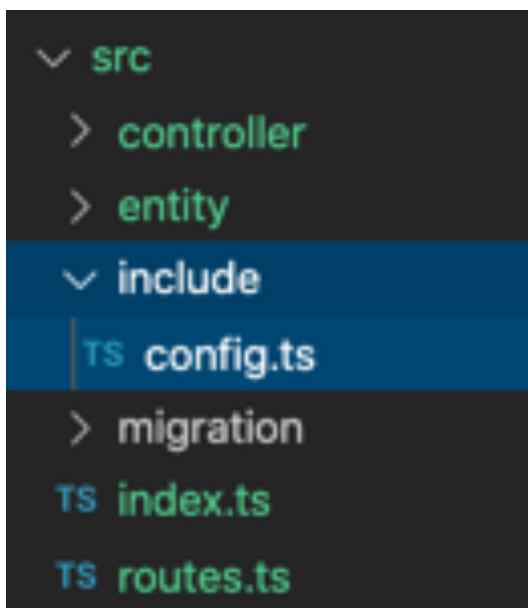
```
config.ts  X
PlayingScore > ps.api > src > include > config.ts
1  export default {
2    |   port: process.env.PORT || 3000
3  }
```

# Server Side

Agora vamos configurar o index.ts conforme nossas necessidades

- Primeiro, vamos fazer uma limpa e deixar apenas o que nos é útil
- Agora, vamos criar um arquivo de configuração do ambiente → include/config.ts

Vamos exportar todo o conteúdo de nosso config, deixando visível a quem utilizar



The image shows a file tree on the left and a code editor on the right. The file tree under 'src' includes 'controller', 'entity', 'include', 'migration', 'index.ts', and 'routes.ts'. The 'include' folder contains a 'config.ts' file, which is highlighted with a blue selection bar. The code editor shows the contents of 'config.ts':

```
TS config.ts ×  
PlayingScore > ps.api > src > include > TS config.ts  
1  export default {  
2      port: process.env.PORT || 3000  
3  }
```

# Server Side

Agora vamos configurar o index.ts conforme nossas necessidades

- Primeiro, vamos fazer uma limpa e deixar apenas o que nos é útil
- Agora, vamos criar um arquivo de configuração do ambiente → include/config.ts

Configuramos a variável "port" que será responsável por configurar a porta do servidor

The image shows a file explorer on the left and a code editor on the right. The file explorer displays a directory structure under 'src': 'controller', 'entity', 'include' (which contains 'config.ts'), 'migration', 'index.ts', and 'routes.ts'. The 'config.ts' file is selected. The code editor shows the contents of 'config.ts':

```
TS config.ts ×  
PlayingScore > ps.api > src > include > TS config.ts  
1 export default {  
2   port: process.env.PORT || 3000  
3 }  
4  
5 }
```

The line 'port: process.env.PORT || 3000' is highlighted with a blue rectangle.

# Server Side

Agora vamos configurar o index.ts conforme nossas necessidades

- Primeiro, vamos fazer uma limpa e deixar apenas o que nos é útil
- Agora, vamos criar um arquivo de configuração do ambiente → include/config.ts

Caso não exista variável de ambiente (env.PORT) configurada, utilizar 3000

The image shows a file explorer on the left and a code editor on the right. The file explorer displays a directory structure under 'src': 'controller', 'entity', 'include' (which contains 'config.ts'), 'migration', 'index.ts', and 'routes.ts'. The 'config.ts' file is selected. The code editor shows the contents of 'config.ts':

```
TS config.ts ×  
PlayingScore > ps.api > src > include > TS config.ts  
1 export default {  
2   port: process.env.PORT || 3000  
3 }  
4  
5 }
```

The line 'port: process.env.PORT || 3000' is highlighted with a blue rectangle.

# Server Side

---

Agora vamos configurar o index.ts conforme nossas necessidades

- Primeiro, vamos fazer uma limpa e deixar apenas o que nos é útil
- Agora, vamos criar um arquivo de configuração do ambiente → include/config.ts
- Estamos prontos para utilizar a configuração no index.ts

# Server Side

Agora vamos configurar o index.ts conforme nossas necessidades

- Primeiro, vamos fazer uma limpa e deixar apenas o que nos é útil
- Agora, vamos criar um arquivo de configuração do ambiente → include/config.ts
- Estamos prontos para utilizar a configuração no index.ts

```
TS index.ts  ×

PlayingScore > ps.api > src > TS index.ts > ...
1  import "reflect-metadata";
2  import {createConnection} from "typeorm";
3  import * as express from "express";
4  import * as bodyParser from "body-parser";
5  import {Request, Response} from "express";
6  import {Routes} from "./routes";
7  import config from "./include/config";
8
9  // create express app
10 const app = express();
11 app.use(bodyParser.json());
12
13 // register express routes from defined application routes
14 Routes.forEach(route => {
15     (app as any)[route.method](route.route, (req: Request, res: Response, next: Function) => {
16         const result = (new (route.controller as any))[route.action](req, res, next);
17         if (result instanceof Promise) {
18             result.then(result => result !== null && result !== undefined ? res.send(result) : undefined);
19
20         } else if (result !== null && result !== undefined) {
21             res.json(result);
22         }
23     });
24 });
25
26 app.listen(config.port, '0.0.0.0', () => {
27     console.log("Playing Score API started");
28 });


```

# Server Side

Agora vamos configurar o index.ts conforme nossas necessidades

- Primeiro, vamos fazer uma limpa e deixar apenas o que nos é útil
- Agora, vamos criar um arquivo de configuração do ambiente → include/config.ts
- Estamos prontos para utilizar a configuração no index.ts

TS index.ts ×

```
PlayingScore > ps.api > src > TS index.ts > ...
1  import "reflect-metadata";
2  import {createConnection} from "typeorm";
3  import * as express from "express";
4  import * as bodyParser from "body-parser";
5  import {Request, Response} from "express";
6  import {Routes} from "./routes";
7  import config from "./include/config";
8
9  // create express app
10 const app = express();
11 app.use(bodyParser.json());
12
13 // register express routes from defined application routes
14 Routes.forEach(route => {
15     (app as any)[route.method](route.route, (req: Request, res: Response, next: Function) => {
16         const result = (new (route.controller as any))[route.action](req, res, next);
17         if (result instanceof Promise) {
18             result.then(result => result !== null && result !== undefined ? res.send(result) : undefined);
19
20         } else if (result !== null && result !== undefined) {
21             res.json(result);
22         }
23     });
24 });
25
26 app.listen(config.port, '0.0.0.0', () => {
27     console.log("Playing Score API started");
28 });

Adicionamos o import do arquivo de configuração criado
```

# Server Side

Agora vamos configurar o index.ts conforme nossas necessidades

- Primeiro, vamos fazer uma limpa e deixar apenas o que nos é útil
- Agora, vamos criar um arquivo de configuração do ambiente → include/config.ts
- Estamos prontos para utilizar a configuração no index.ts

TS index.ts X

```
PlayingScore > ps.api > src > TS index.ts > ...
1  import "reflect-metadata";
2  import {createConnection} from "typeorm";
3  import * as express from "express";
4  import * as bodyParser from "body-parser";
5  import {Request, Response} from "express";
6  import {Routes} from "./routes";
7  import config from "./include/config";
8
9  // create express app
10 const app = express();
11 app.use(bodyParser.json());
12
13 // register express routes from defined application routes
14 Routes.forEach(route => {
15     (app as any)[route.method](route.route, (req: Request, res: Response, next: Function) => {
16         const result = (new (route.controller as any))[route.action](req, res, next);
17         if (result instanceof Promise) {
18             result.then(result => result !== null && result !== undefined ? res.send(result) : undefined);
19
20         } else if (result !== null && result !== undefined) {
21             res.json(result);
22         }
23     });
24 });
25
26 app.listen(config.port, '0.0.0.0', () => {
27     console.log("Playing Score API started");
28 });


```

Utilizamos o arquivo de configuração para controlar a porta e iniciar a API

# Server Side

Agora vamos configurar o index.ts conforme nossas necessidades

- Primeiro, vamos fazer uma limpa e deixar apenas o que nos é útil
- Agora, vamos criar um arquivo de configuração do ambiente → include/config.ts
- Estamos prontos para utilizar a configuração no index.ts

ts index.ts ×

```
PlayingScore > ps.api > src > ts index.ts > ...
1 import "reflect-metadata";
2 import {createConnection} from "typeorm";
3 import * as express from "express";
4 import * as bodyParser from "body-parser";
5 import {Request, Response} from "express";
6 import {Routes} from "./routes";
7 import config from "./include/config";
8
```

```
[→ ps.api git:(master) ✘ npm start
```

```
> ps.api@0.0.1 start /Users/madalozzo/Dropbox/1-UPF/2020/Semestre_1/POS-Ionic/Dev/PlayingScore(ps.api
> ts-node src/index.ts
```

```
Playing Score API started
```

```
18         result.then(result => result !== null && result !== undefined ? res.send(result) : undefined);
19
20     } else if (result !== null && result !== undefined) {
21         res.json(result);
22     }
23 });
24 });
25
26 app.listen(config.port, '0.0.0.0', () => {
27     console.log("Playing Score API started");
28 });


```

# Server Side

---

Agora vamos configurar o index.ts conforme nossas necessidades

- Com essa mudança perdemos a conexão com o banco de dados, então vamos ajustar utilizando async await

# Server Side

Agora vamos configurar o index.ts conforme nossas necessidades

- Com essa mudança perdemos a conexão com o banco de dados, então vamos ajustar utilizando async await

```
TS index.ts  ×

PlayingScore > ps.api > src > TS index.ts > ...

25
26 app.listen(config.port, '0.0.0.0', async () => {
27   console.log(`Playing Score API started in port ${config.port}`);
28   try {
29     await createConnection();
30     console.log('Connected to Database');
31   } catch (error) {
32     console.error(`Error to connect to database: ${error}`);
33   }
34 });


```

# Server Side

Agora vamos configurar o index.ts conforme nossas necessidades

- Com essa mudança perdemos a conexão com o banco de dados, então vamos ajustar utilizando async await

```
TS index.ts  ×  
PlayingScore > ps.api > src > TS index.ts > ...  
25  
[→ ps.api git:(master) ✘ npm start  
> ps.api@0.0.1 start /Users/madalozzo/Dropbox/1-UPF/2020/Semestre_1/POS-Ionic/Dev/PlayingScore(ps.api  
> ts-node src/index.ts  
  
Playing Score API started in port 3000  
Connected to Database  
31      } catch (error) {  
32          console.error(`Error to connect to database: ${error}`);  
33      }  
34  });
```

# Criando o Model User

# Criando o Model User

---

O model user que vem criado com o TypeORM será modificado conforme nossa necessidade

- Primeiro, vamos entender como funciona o ORM

# Criando o Model User

O model user que vem criado com o TypeORM será modificado conforme nossa necessidade

- Primeiro, vamos entender como funciona o ORM

```
ts User.ts •  
PlayingScore > ps.api > src > entity > ts User.ts > ...  
1 import {Entity, PrimaryGeneratedColumn, Column} from "typeorm";  
2  
3 @Entity()  
4 export class User {  
5  
6     @PrimaryGeneratedColumn()  
7     id: number;  
8  
9     @Column()  
10    firstName: string;  
11  
12    @Column()  
13    lastName: string;  
14  
15    @Column()  
16    age: number;  
17  
18}
```

Quando a aplicação é executada  
o ORM irá aplicar as mudanças  
dos models na base de dados

# Criando o Model User

O model user que vem criado com o TypeORM será modificado conforme nossa necessidade

- Primeiro, vamos entender como funciona o ORM

The screenshot shows two code files in a code editor:

- User.ts**: A TypeScript file defining a User entity. It includes annotations for primary generated columns, first name, last name, and age.
- ormconfig.json**: A JSON configuration file for TypeORM. It specifies the database as "playing\_score" and sets "synchronize" to true.

A callout box highlights the "synchronize": true line in the ormconfig.json file with the text: "No ormconfig.ts está ativado o synchronize".

```
ts User.ts
PlayingScore > ps.api > src > entity > ts User.ts > ...
1 import {Entity, PrimaryGeneratedColumn, Column} from "typeorm";
2
3 @Entity()
4 export class User {
5
6   @PrimaryGeneratedColumn()
7   id: number;
8
9   @Column()
10  firstName: string;
11
12  @Column()
13  lastName: string;
14
15  @Column()
16  age: number;
17}
18

{} ormconfig.json ×
PlayingScore > ps.api > {} ormconfig.json > ...
7
8
9
"database": "playing_score",
... "synchronize": true,
"logging": false
```

# Criando o Model User

O model user que vem criado com o TypeORM será modificado conforme nossa necessidade

- Primeiro, vamos entender como funciona o ORM

```
ts User.ts •  
PlayingScore > ps.api > src > entity > ts User.ts > ...  
1 import {Entity, PrimaryGeneratedColumn, Column} from "typeorm";  
2  
3 @Entity()  
4 export class User {  
5  
6     @PrimaryGeneratedColumn()  
7     id: number;  
8  
9     @Column()  
10    firstName: string;  
11  
12    @Column()  
13    lastName: string;  
14  
15    @Column()  
16    age: number;  
17  
18}
```

Caso a entidade (tabela) user  
não estiver criada no banco, o  
TypeORM irá criar

# Criando o Model User

O model user que vem criado com o TypeORM será modificado conforme nossa necessidade

- Primeiro, vamos entender como funciona o ORM

```
ts User.ts •  
PlayingScore > ps.api > src > entity > ts User.ts > ...  
1 import {Entity, PrimaryGeneratedColumn, Column} from "typeorm";  
2  
3 @Entity()  
4 export class User {  
5  
6     @PrimaryGeneratedColumn()  
7     id: number;  
8  
9     @Column()  
10    firstName: string;  
11  
12    @Column()  
13    lastName: string;  
14  
15    @Column()  
16    age: number;  
17  
18}
```

Os atributos da tabela seguirão a configuração da classe

# Criando o Model User

---

O model user que vem criado com o TypeORM será modificado conforme nossa necessidade

- Agora, vamos configurar conforme nossa necessidade

# Criando o Model User

```
ts User.ts  X

PlayingScore > ps.api > src > entity > ts User.ts > ...
1  import {Entity, PrimaryGeneratedColumn, Column} from "typeorm";
2
3  @Entity()
4  export class User {
5
6    @PrimaryGeneratedColumn("uuid")
7    uid: string;
8
9    @Column({ type: 'varchar', length: 100 })
10   name: string;
11
12  @Column({ type: 'varchar', length: 40 })
13   nick: string;
14
15  @Column({ type: 'varchar', length: 100 })
16   photo: string;
17
18  @Column({ type: 'varchar', length: 100 })
19   email: string;
20
21  @Column({ type: 'varchar', length: 100 })
22   nick_instagram: string;
23
24  @Column({ type: 'varchar', length: 100 })
25   nick_facebook: string;
26
27  @Column({ type: 'varchar', length: 100 })
28   whatsapp: string;
29
30  @Column({ type: 'varchar', length: 100 })
31   password: string;
32
33  @Column({ default: true })
34   isPublic: boolean;
35
36  @Column({ default: true })
37   isActive: boolean;
38
39  @Column()
40   createdAt: Date;
41
42 }
```

# Criando o Model User

```
ts User.ts  X

PlayingScore > ps.api > src > entity > ts User.ts > ...
1  import {Entity, PrimaryGeneratedColumn, Column} from "typeorm";
2
3  @Entity()
4  export class User {
5
6    @PrimaryGeneratedColumn("uuid")
7    uid: string;
8
9    @Column({ type: 'varchar', length: 100 })
10   name: string;
11
12   @Column({ type: 'varchar', length: 40 })
13   nick: string;
14
15   @Column({ type: 'varchar', length: 100 })
16   photo: string;
17
18   @Column({ type: 'varchar', length: 100 })
19   email: string;
20
21   @Column({ type: 'varchar', length: 100 })
22   nick_instagram: string;
23
24   @Column({ type: 'varchar', length: 100 })
25   nick_facebook: string;
26
27   @Column({ type: 'varchar', length: 100 })
28   whatsapp: string;
29
30   @Column({ type: 'varchar', length: 100 })
31   password: string;
32
33   @Column({ default: true })
34   isPublic: boolean;
35
36   @Column({ default: true })
37   isActive: boolean;
38
39   @Column()
40   createdAt: Date;
41
42 }
```

Todos os atributos da classe possuem um conceito chamado decoration

# Criando o Model User

```
ts User.ts  X

PlayingScore > ps.api > src > entity > ts User.ts > ...
1  import {Entity, PrimaryGeneratedColumn, Column} from "typeorm";
2
3  @Entity()
4  export class User {
5
6      @PrimaryGeneratedColumn("uuid")
7      uid: string;
8
9      @Column({ type: 'varchar', length: 100 })
10     name: string;
11
12     @Column({ type: 'varchar', length: 40 })
13     nick: string;
14
15     @Column({ type: 'varchar', length: 100 })
16     photo: string;
17
18     @Column({ type: 'varchar', length: 100 })
19     email: string;
20
21     @Column({ type: 'varchar', length: 100 })
22     nick_instagram: string;
23
24     @Column({ type: 'varchar', length: 100 })
25     nick_facebook: string;
26
27     @Column({ type: 'varchar', length: 100 })
28     whatsapp: string;
29
30     @Column({ type: 'varchar', length: 100 })
31     password: string;
32
33     @Column({ default: true })
34     isPublic: boolean;
35
36     @Column({ default: true })
37     isActive: boolean;
38
39     @Column()
40     createdAt: Date;
41
42 }
```

Criamos um uid como UUID

DICA!

<https://www.youtube.com/watch?v=9vtcTJwGt-w>



UUID (Universally  
Unique IDentifier ou  
GUID) // Dicionário do ...

Código Fonte TV

YouTube - 20 de abr. de 2020

# Criando o Model User

```
ts User.ts  X

PlayingScore > ps.api > src > entity > ts User.ts > ...
1 import {Entity, PrimaryGeneratedColumn, Column} from "typeorm";
2
3 @Entity()
4 export class User {
5
6   @PrimaryGeneratedColumn("uuid")
7   uid: string;
8
9   @Column({ type: 'varchar', length: 100 })
10  name: string;
11
12  @Column({ type: 'varchar', length: 40 })
13  nick: string;
14
15  @Column({ type: 'varchar', length: 100 })
16  photo: string;
17
18  @Column({ type: 'varchar', length: 100 })
19  email: string;
20
21  @Column({ type: 'varchar', length: 100 })
22  nick_instagram: string;
23
24  @Column({ type: 'varchar', length: 100 })
25  nick_facebook: string;
26
27  @Column({ type: 'varchar', length: 100 })
28  whatsapp: string;
29
30  @Column({ type: 'varchar', length: 100 })
31  password: string;
32
33  @Column({ default: true })
34  isPublic: boolean;
35
36  @Column({ default: true })
37  isActive: boolean;
38
39  @Column()
40  createdAt: Date;
41
42 }
```

Atributos VARCHAR

# Criando o Model User

```
ts User.ts  X

PlayingScore > ps.api > src > entity > ts User.ts > ...
1  import {Entity, PrimaryGeneratedColumn, Column} from "typeorm";
2
3  @Entity()
4  export class User {
5
6      @PrimaryGeneratedColumn("uuid")
7      uid: string;
8
9      @Column({ type: 'varchar', length: 100 })
10     name: string;
11
12     @Column({ type: 'varchar', length: 40 })
13     nick: string;
14
15     @Column({ type: 'varchar', length: 100 })
16     photo: string;
17
18     @Column({ type: 'varchar', length: 100 })
19     email: string;
20
21     @Column({ type: 'varchar', length: 100 })
22     nick_instagram: string;
23
24     @Column({ type: 'varchar', length: 100 })
25     nick_facebook: string;
26
27     @Column({ type: 'varchar', length: 100 })
28     whatsapp: string;
29
30     @Column({ type: 'varchar', length: 100 })
31     password: string;
32
33     @Column({ default: true })
34     isPublic: boolean;
35
36     @Column({ default: true })
37     isActive: boolean;
38
39     @Column()
40     createdAt: Date;
41
42 }
```

Atributos BOOLEAN com default

# Criando o Model User

```
ts User.ts  X

PlayingScore > ps.api > src > entity > ts User.ts > ...
1  import {Entity, PrimaryGeneratedColumn, Column} from "typeorm";
2
3  @Entity()
4  export class User {
5
6    @PrimaryGeneratedColumn("uuid")
7    uid: string;
8
9    @Column({ type: 'varchar', length: 100 })
10   name: string;
11
12  @Column({ type: 'varchar', length: 40 })
13   nick: string;
14
15  @Column({ type: 'varchar', length: 100 })
16   photo: string;
17
18  @Column({ type: 'varchar', length: 100 })
19   email: string;
20
21  @Column({ type: 'varchar', length: 100 })
22   nick_instagram: string;
23
24  @Column({ type: 'varchar', length: 100 })
25   nick_facebook: string;
26
27  @Column({ type: 'varchar', length: 100 })
28   whatsapp: string;
29
30  @Column({ type: 'varchar', length: 100 })
31   password: string;
32
33  @Column({ default: true })
34   isPublic: boolean;
35
36  @Column({ default: true })
37   isActive: boolean;
38
39  @Column()
40   createdAt: Date;
41
42 }
```

Atributo DATE

# Criando o Model User

```
ts User.ts  X

PlayingScore > ps.api > src > entity > ts User.ts > ...
1 import {Entity, PrimaryGeneratedColumn, Column} from "typeorm";
2
3 @Entity()
4 export class User {
5
6   @PrimaryGeneratedColumn("uuid")
7   uid: string;
8
9   @Column({ type: 'varchar', length: 100 })
10  name: string;
11
12  @Column({ type: 'varchar', length: 40 })
13  nick: string;
14
15  @Column({ type: 'varchar', length: 100 })
16  photo: string;
17
18  @Column({ type: 'varchar', length: 100 })
19  email: string;
20
21  @Column({ type: 'varchar', length: 100 })
22  nick_instagram: string;
23
24  @Column({ type: 'varchar', length: 100 })
25  nick_facebook: string;
26
27  @Column({ type: 'varchar', length: 100 })
28  whatsapp: string;
29
30  @Column({ type: 'varchar', length: 100 })
31  password: string;
32
33  @Column({ default: true })
34  isPublic: boolean;
35
36  @Column({ default: true })
37  isActive: boolean;
38
39  @Column()
40  createdAt: Date;
41
42 }
```

Se paramos para analisar, temos campos que irão se repetir em diferentes tabelas

# Criando o Model User

```
ts User.ts  X

PlayingScore > ps.api > src > entity > ts User.ts > ...
1  import {Entity, PrimaryGeneratedColumn, Column} from "typeorm";
2
3  @Entity()
4  export class User {
5
6      @PrimaryGeneratedColumn("uuid")
7      uid: string;
8
9      @Column({ type: 'varchar', length: 100 })
10     name: string;
11
12     @Column({ type: 'varchar', length: 40 })
13     nick: string;
14
15     @Column({ type: 'varchar', length: 100 })
16     photo: string;
17
18     @Column({ type: 'varchar', length: 100 })
19     email: string;
20
21     @Column({ type: 'varchar', length: 100 })
22     nick_instagram: string;
23
24     @Column({ type: 'varchar', length: 100 })
25     nick_facebook: string;
26
27     @Column({ type: 'varchar', length: 100 })
28     whatsapp: string;
29
30     @Column({ type: 'varchar', length: 100 })
31     password: string;
32
33     @Column({ default: true })
34     isPublic: boolean;
35
36     @Column({ default: true })
37     isActive: boolean;
38
39     @Column()
40     createdAt: Date;
41
42 }
```

Se paramos para analisar, temos campos que irão se repetir em diferentes tabelas

# Criando o Model User

---

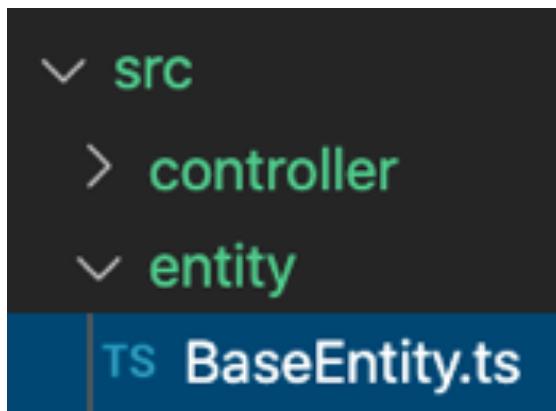
O model user que vem criado com o TypeORM será modificado conforme nossa necessidade

- Como temos atributos que serão padrões em diferentes entidades, vamos criar uma classe abstrata para que não precisemos ficar replicando os campos

# Criando o Model User

O model user que vem criado com o TypeORM será modificado conforme nossa necessidade

- Como temos atributos que serão padrões em diferentes entidades, vamos criar uma classe abstrata para que não precisemos ficar replicando os campos



Criar dentro de entity o arquivo BaseEntity.ts

# Criando o Model User

O model user que vem criado com o TypeORM será modificado conforme nossa necessidade

- Como temos atributos que serão padrões em diferentes entidades, vamos criar uma classe abstrata para que não precisemos ficar replicando os campos repetidos

```
ts BaseEntity.ts ×  
PlayingScore > ps.api > src > entity > ts BaseEntity.ts > ...  
1 import { PrimaryGeneratedColumn, Column, CreateDateColumn, UpdateDateColumn } from "typeorm";  
2  
3 export abstract class BaseEntity {  
4  
5   @PrimaryGeneratedColumn("uuid")  
6   uid: string;  
7  
8   @Column({ default: true })  
9   isPublic: boolean;  
10  
11  @Column({ default: true })  
12  isActive: boolean;  
13  
14  @CreateDateColumn({ type: 'timestamp' })  
15  createdAt: Date;  
16  
17  @UpdateDateColumn({ type: 'timestamp' })  
18  updatedAt: Date;  
19  
20 }
```

Adicionamos nesta classe todos os atributos em comum que as tabelas apresentam

# Criando o Model User

O model user que vem criado com o TypeORM será modificado conforme nossa necessidade

- Como temos atributos que serão padrões em diferentes entidades, vamos criar uma classe abstrata para que não precisemos ficar replicando os campos repetidosBaseEntity

```
ts BaseEntity.ts ×  
PlayingScore > ps.api > src > entity > ts BaseEntity.ts > ...  
1 import { PrimaryGeneratedColumn, Column, CreateDateColumn, UpdateDateColumn } from "typeorm";  
2  
3 export abstract class BaseEntity {  
4  
5     @PrimaryGeneratedColumn("uuid")  
6     uid: string;  
7  
8     @Column({ default: true })  
9     isPublic: boolean;  
10  
11    @Column({ default: true })  
12    isActive: boolean;  
13  
14    @CreateDateColumn({ type: 'timestamp' })  
15    createdAt: Date;  
16  
17    @UpdateDateColumn({ type: 'timestamp' })  
18    updatedAt: Date;  
19  
20}
```

Note que mudamos a Column para um CreateDateColumn

# Criando o Model User

O model user que vem criado com o TypeORM será modificado conforme nossa necessidade

- Como temos atributos que serão padrões em diferentes entidades, vamos criar uma classe abstrata para que não precisemos ficar replicando os campos repetidosBaseEntity

```
ts BaseEntity.ts ×  
PlayingScore > ps.api > src > entity > ts BaseEntity.ts > ...  
1 import { PrimaryGeneratedColumn, Column, CreateDateColumn, UpdateDateColumn } from "typeorm";  
2  
3 export abstract class BaseEntity {  
4  
5     @PrimaryGeneratedColumn("uuid")  
6     uid: string;  
7  
8     @Column({ default: true })  
9     isPublic: boolean;  
10  
11    @Column({ default: true })  
12    isActive: boolean;  
13  
14    @CreateDateColumn({ type: 'timestamp' })  
15    createdAt: Date;  
16  
17    @UpdateDateColumn({ type: 'timestamp' })  
18    updatedAt: Date;  
19  
20 }
```

Note que adicionamos um campo updatedAt com UpdateDateColumn

# Criando o Model User

---

O model user que vem criado com o TypeORM será modificado conforme nossa necessidade

- Agora, vamos atualizar o User.ts

# Criando o Model User

O model user que vem criado com o TypeORM serve:

- Agora, vamos atualizar o User.ts

TS User.ts X

```
PlayingScore > ps.api > src > entity > TS User.ts > ...
1   import {Entity, PrimaryGeneratedColumn, Column} from "typeorm";
2   import { BaseEntity } from "./BaseEntity";
3
4   @Entity({ name: 'User' })
5   export class User extends BaseEntity {
6
7     @Column({ type: 'varchar', length: 100 })
8     name: string;
9
10    @Column({ type: 'varchar', length: 40 })
11    nick: string;
12
13    @Column({ type: 'varchar', length: 100 })
14    photo: string;
15
16    @Column({ type: 'varchar', length: 100 })
17    email: string;
18
19    @Column({ type: 'varchar', length: 100 })
20    nick_instagram: string;
21
22    @Column({ type: 'varchar', length: 100 })
23    nick_facebook: string;
24
25    @Column({ type: 'varchar', length: 100 })
26    whatsapp: string;
27
28    @Column({ type: 'varchar', length: 100 })
29    password: string;
30
31 }
```

# Criando o Model User

O model user que vem criado com o TypeORM serve:

- Agora, vamos atualizar o User.ts

TS User.ts ×

```
PlayingScore > ps.api > src > entity > TS User.ts > ...
1 import {Entity, PrimaryGeneratedColumn, Column} from "typeorm";
2 import { BaseEntity } from "./BaseEntity";
3
4 @Entity({ name: 'User' })
5 export class User extends BaseEntity {
6
7     @Column({ type: 'varchar', length: 100 })
8     name: string;
9
10    @Column({ type: 'varchar', length: 40 })
11    nick: string;
12
13    @Column({ type: 'varchar', length: 100 })
14    photo: string;
15
16    @Column({ type: 'varchar', length: 100 })
17    email: string;
18
19    @Column({ type: 'varchar', length: 100 })
20    nick_instagram: string;
21
22    @Column({ type: 'varchar', length: 100 })
23    nick_facebook: string;
24
25    @Column({ type: 'varchar', length: 100 })
26    whatsapp: string;
27
28    @Column({ type: 'varchar', length: 100 })
29    password: string;
30
31 }
```

Fizemos o import do BaseEntity

# Criando o Model User

O model user que vem criado com o TypeORM ser

- Agora, vamos atualizar o User.ts

Utilizamos com extends esta nova classe

TS User.ts X

```
PlayingScore > ps.api > src > entity > TS User.ts > ...
1   import {Entity, PrimaryGeneratedColumn, Column} from "typeorm";
2   import { BaseEntity } from "./BaseEntity";
3
4   @Entity({ name: 'User' })
5   export class User extends BaseEntity {
6
7     @Column({ type: 'varchar', length: 100 })
8     name: string;
9
10    @Column({ type: 'varchar', length: 40 })
11    nick: string;
12
13    @Column({ type: 'varchar', length: 100 })
14    photo: string;
15
16    @Column({ type: 'varchar', length: 100 })
17    email: string;
18
19    @Column({ type: 'varchar', length: 100 })
20    nick_instagram: string;
21
22    @Column({ type: 'varchar', length: 100 })
23    nick_facebook: string;
24
25    @Column({ type: 'varchar', length: 100 })
26    whatsapp: string;
27
28    @Column({ type: 'varchar', length: 100 })
29    password: string;
30
31 }
```

# Criando o Model User

O model user que vem criado com o TypeORM serve:

- Agora, vamos atualizar o User.ts

Note que “forçamos” o nome da tabela para ser “User”

TS User.ts ×

```
PlayingScore > ps.api > src > entity > TS User.ts > ...
1   import {Entity, PrimaryGeneratedColumn, Column} from "typeorm";
2   import { BaseEntity } from "./BaseEntity";
3
4   @Entity({ name: 'User' })
5   export class User extends BaseEntity {
6
7     @Column({ type: 'varchar', length: 100 })
8     name: string;
9
10    @Column({ type: 'varchar', length: 40 })
11    nick: string;
12
13    @Column({ type: 'varchar', length: 100 })
14    photo: string;
15
16    @Column({ type: 'varchar', length: 100 })
17    email: string;
18
19    @Column({ type: 'varchar', length: 100 })
20    nick_instagram: string;
21
22    @Column({ type: 'varchar', length: 100 })
23    nick_facebook: string;
24
25    @Column({ type: 'varchar', length: 100 })
26    whatsapp: string;
27
28    @Column({ type: 'varchar', length: 100 })
29    password: string;
30
31 }
```

# Criando o Model User

---

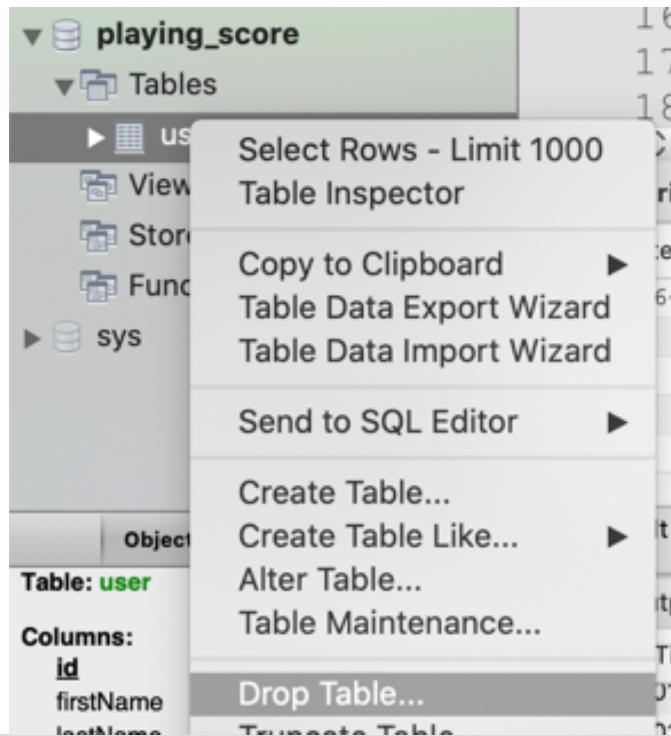
O model user que vem criado com o TypeORM será modificado conforme nossa necessidade

- Podemos, agora, executar o projeto para que esta tabela seja criada
- Como já executamos antes, vamos apagar a tabela na base de dados primeiro

# Criando o Model User

O model user que vem criado com o TypeORM será modificado conforme nossa necessidade

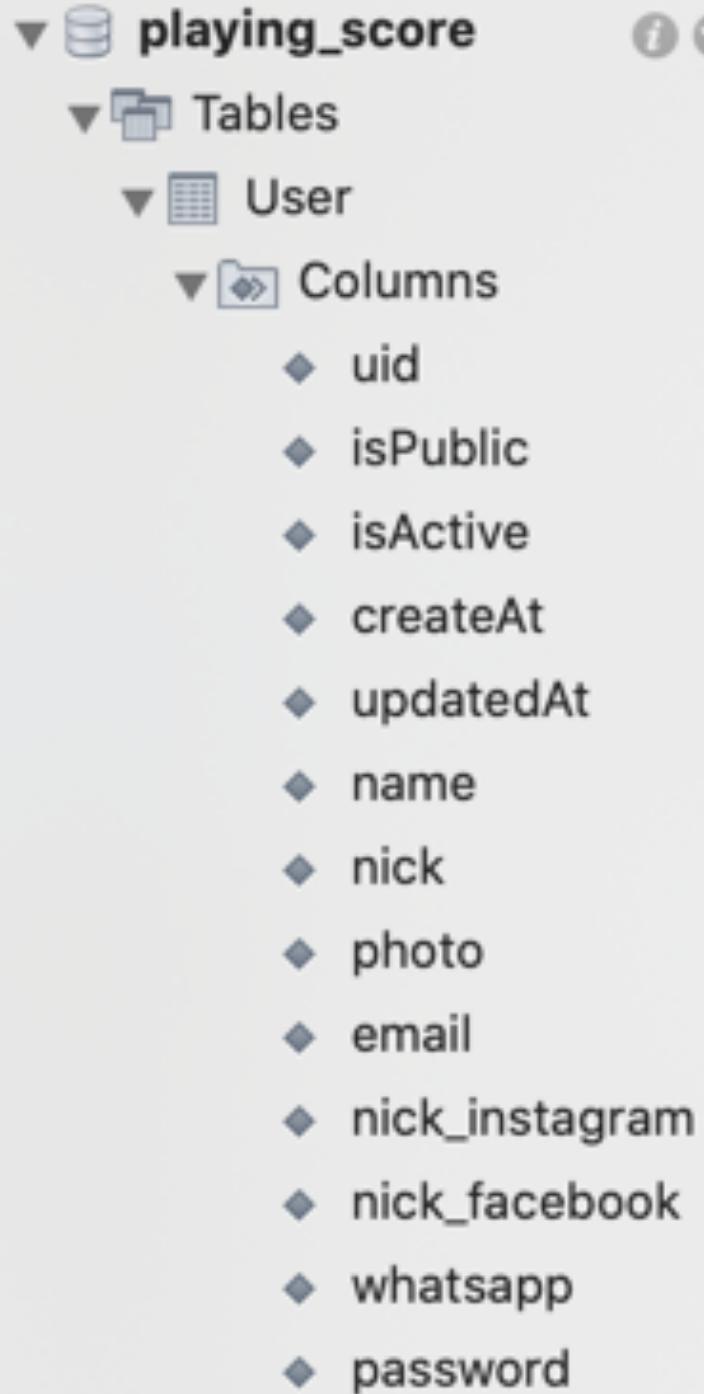
- Podemos, agora, executar o projeto para que esta tabela seja criada
- Como já executamos antes, vamos apagar a tabela na base de dados primeiro



# Criando o Model User

O model user que vem criado com o TypeORM será modificado conforme:

- Podemos, agora, executar o projeto para que esta tabela seja criada
- Como já executamos antes, vamos apagar a tabela na base de dados primeiro
- Ao executarmos `npm start` a tabela será criada



# Criando o Model User

---

Com o model criado, agora vamos programar uma classe para controle de validação dos dados

# Criando o Model User

Com o model criado, agora vamos programar uma classe para controle de validação dos dados

```
✓ src
  > controller
  ✓ entity
    TS BaseEntity.ts
    TS BaseValidation.ts
    TS User.ts
```

Criar dentro de entity o arquivo BaseValidation.ts

# Criando o Model User

```
ts BaseValidation.ts x
PlayingScore > ps.api > src > entity > ts BaseValidation.ts > ...
1  export abstract class BaseValidation {
2
3      notifications: Array<{ message: string }>;
4
5      constructor() {
6          this.notifications = new Array<{ message: string }>();
7      }
8
9      valid(): boolean {
10         return this.notifications.length == 0;
11     }
12
13     get allNotifications(): Array<{ message: string }> {
14         return this.notifications;
15     }
16
17     AddNotification(message: string): void {
18         this.notifications.push({ message: message });
19     }
20
21     isTrue(value, message) {
22         if (value)
23             this.notifications.push({ message: message });
24     }
25
26    isRequired(value, message) {
27         if (!value || value.length <= 0)
28             this.notifications.push({ message: message });
29     }
30
31     hasMinLen(value, min, message) {
32         if (!value || value.length < min)
33             this.notifications.push({ message: message });
34     }
35
36     hasMaxLen(value, max, message) {
37         if (!value || value.length > max)
38             this.notifications.push({ message: message });
39     }
40
41     isFixedLen(value, len, message) {
42         if (value.length != len)
43             this.notifications.push({ message: message });
44     }
45
46     isEmail(value, message) {
47         var reg = new RegExp(/^\w+([-+.']\w+)*@\w+([\.-]\w+)*\.\w+([\.-]\w+)*$/);
48         if (!reg.test(value))
49             this.notifications.push({ message: message });
50     }
51
52 }
```

A ideia desta classe é colocarmos todas as validações necessárias, veremos uma a uma

# Criando o Model User

```
ts BaseValidation.ts ×  
PlayingScore > ps.api > src > entity > ts BaseValidation.ts > ...  
1  export abstract class BaseValidation {  
2  
3      notifications: Array<{ message: string }>;  
4  
5      constructor() {  
6          this.notifications = new Array<{ message: string }>();  
7      }  
8  
9  
10  
11  
12  
13  
14  
15  
16      export abstract class BaseValidation {  
17  
18  
19          notifications: Array<{ message: string }>;  
20  
21  
22          constructor() {  
23              this.notifications = new Array<{ message: string }>();  
24          }  
25  
26  
27          valid(): boolean {  
28              return this.notifications.length == 0;  
29          }  
30  
31  
32  
33  
34          get allNotifications(): Array<{ message: string }> {  
35              return this.notifications;  
36          }  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52      }  
53  }
```

# Criando o Model User

```
ts BaseValidation.ts ×  
PlayingScore > ps.api > src > entity > ts BaseValidation.ts > ...  
1  export abstract class BaseValidation {  
2  
3    notifications: Array<{ message: string }>;  
4  
5    constructor() {  
6      this.notifications = new Array<{ message: string }>();  
7    }  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52 }
```

TS BaseValidation.ts ×

PlayingScore > ps.api > src > entity > ts BaseValidation.ts > BaseValidation >

```
1  export abstract class BaseValidation {  
2  
3    notifications: Array<{ message: string }>;  
4  
5    constructor() {  
6      this.notifications = new Array<{ message: string }>();  
7    }  
8  
9    valid(): boolean {  
10      return this.notifications.length == 0;  
11    }  
12  
13    get allNotifications(): Array<{ message: string }> {  
14      return this.notifications;  
15    }  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52 }
```

Criamos a classe, abstrata, BaseValidation

# Criando o Model User

```
ts BaseValidation.ts ×  
PlayingScore > ps.api > src > entity > ts BaseValidation.ts > ...  
1   export abstract class BaseValidation {  
2  
3     notifications: Array<{ message: string }>;  
4  
5     constructor() {  
6       this.notifications = new Array<{ message: string }>();  
7     }  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52 }
```

ts BaseValidation.ts ×

PlayingScore > ps.api > src > entity > ts BaseValidation.ts > BaseValidation >

```
1   export abstract class BaseValidation {  
2  
3     notifications: Array<{ message: string }>;  
4  
5     constructor() {  
6       this.notifications = new Array<{ message: string }>();  
7     }  
8  
9     valid(): boolean {  
10       return this.notifications.length == 0;  
11     }  
12  
13     get allNotifications(): Array<{ message: string }> {  
14       return this.notifications;  
15     }  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52 }
```

Teremos uma lista de notificações, caso usuário não informe os dados de maneira correta

# Criando o Model User

O construtor inicializa a lista

```
ts BaseValidation.ts >
PlayingScore > ps.api > src > entity > ts BaseValidation.ts > ...
1   export abstract class BaseValidation {
2
3     notifications: Array<{ message: string }>;
4
5     constructor() {
6       this.notifications = new Array<{ message: string }>();
7     }
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52 }
```

TS BaseValidation.ts X

PlayingScore > ps.api > src > entity > ts BaseValidation.ts > BaseValidation >

```
1   export abstract class BaseValidation {
2
3     notifications: Array<{ message: string }>;
4
5     constructor() {
6       this.notifications = new Array<{ message: string }>();
7     }
8
9     valid(): boolean {
10       return this.notifications.length == 0;
11     }
12
13     get allNotifications(): Array<{ message: string }> {
14       return this.notifications;
15     }
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52 }
```

# Criando o Model User

```
ts BaseValidation.ts
PlayingScore > ps.api > src > entity > ts BaseValidation.ts > ...
1   export abstract class BaseValidation {
2
3     notifications: Array<{ message: string }>;
4
5     constructor() {
6       this.notifications = new Array<{ message: string }>();
7     }
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52 }
```

## ts BaseValidation.ts X

PlayingScore > ps.api > src > entity > ts BaseValidation.ts > BaseValidation >

```
1   export abstract class BaseValidation {
2
3     notifications: Array<{ message: string }>;
4
5     constructor() {
6       this.notifications = new Array<{ message: string }>();
7     }
8
9     valid(): boolean {
10       return this.notifications.length == 0;
11     }
12
13     get allNotifications(): Array<{ message: string }> {
14       return this.notifications;
15     }
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52 }
```

Método que retorna se os dados informados são válidos

# Criando o Model User

```
ts BaseValidation.ts ×  
PlayingScore > ps.api > src > entity > ts BaseValidation.ts > ...  
1  export abstract class BaseValidation {  
2  
3    notifications: Array<{ message: string }>;  
4  
5    constructor() {  
6      this.notifications = new Array<{ message: string }>();  
7    }  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52 }
```

ts BaseValidation.ts ×

PlayingScore > ps.api > src > entity > ts BaseValidation.ts > BaseValidation >

```
1  export abstract class BaseValidation {  
2  
3    notifications: Array<{ message: string }>;  
4  
5    constructor() {  
6      this.notifications = new Array<{ message: string }>();  
7    }  
8  
9    valid(): boolean {  
10      return this.notifications.length == 0;  
11    }  
12  
13    get allNotifications(): Array<{ message: string }> {  
14      return this.notifications;  
15    }  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52 }
```

Método que retorna todas as notificações de dados inválidos

# Criando o Model User

```
ts BaseValidation.ts
PlayingScore > ps.api > src > entity > ts BaseValidation.ts > ...
1  export abstract class BaseValidation {
2
3    notifications: Array<{ message: string }>;
4
5    constructor() {
6      this.notifications = new Array<{ message: string }>();
7    }
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52 }
```

Qual a diferença destes 2 métodos com retornos???

TS BaseValidation.ts X

PlayingScore > ps.api > src > entity > ts BaseValidation.ts > BaseValidation >

```
1  export abstract class BaseValidation {
2
3    notifications: Array<{ message: string }>;
4
5    constructor() {
6      this.notifications = new Array<{ message: string }>();
7    }
8
9    valid(): boolean {
10      return this.notifications.length == 0;
11    }
12
13    get allNotifications(): Array<{ message: string }> {
14      return this.notifications;
15    }
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52 }
```

# Criando o Model User

Qual a diferença destes 2 métodos com retornos???

- 1) if (**BaseValidation.valid()**) { ... }
- 2) lista = **BaseValidation.allNotifications**;

```
1  export abstract class BaseValidation {
2
3     notifications: Array<{ message: string }>;
4
5     constructor() {
6         this.notifications = new Array<{ message: string }>();
7     }
8
9
10    valid(): boolean {
11        return this.notifications.length == 0;
12    }
13
14    get allNotifications(): Array<{ message: string }> {
15        return this.notifications;
16    }
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52 }
```

# Criando o Model User

```
ts BaseValidation.ts <pre></pre>
PlayingScore > ps.api > src > entity > ts BaseValidation.ts > ...
1   export abstract class BaseValidation {
2
3     notifications: Array<{ message: string }>;
4
5     constructor() {
6       this.notifications = new Array<{ message: string }>();
7     }
8
9
10
11
12
13
14
15
16
17   AddNotification(message: string): void {
18     this.notifications.push({ message: message });
19   }
20
21   isTrue(value, message) {
22     if (value)
23       this.notifications.push({ message: message });
24   }
25
26   isRequired(value, message) {
27     if (!value || value.length <= 0)
28       this.notifications.push({ message: message });
29   }
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52 </pre>
```

# Criando o Model User

```
ts BaseValidation.ts
PlayingScore > ps.api > src > entity > ts BaseValidation.ts > ...
1   export abstract class BaseValidation {
2
3     notifications: Array<{ message: string }>;
4
5     constructor() {
6       this.notifications = new Array<{ message: string }>();
7     }
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52 }
```

## TS BaseValidation.ts ×

PlayingScore > ps.api > src > entity > **TS BaseValidation.ts** > ...

```
16
17   AddNotification(message: string): void {
18     this.notifications.push({ message: message });
19   }
20
21   isTrue(value, message) {
22     if (value)
23       this.notifications.push({ message: message });
24   }
25
26   isRequired(value, message) {
27     if (!value || value.length <= 0)
28       this.notifications.push({ message: message });
29 }
```

Método para adicionar nova notificação de validação caso necessite

# Criando o Model User

```
ts BaseValidation.ts >
PlayingScore > ps.api > src > entity > ts BaseValidation.ts > ...
1   export abstract class BaseValidation {
2
3     notifications: Array<{ message: string }>;
4
5     constructor() {
6       this.notifications = new Array<{ message: string }>();
7     }
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52 }
```

## TS BaseValidation.ts ×

PlayingScore > ps.api > src > entity > **TS BaseValidation.ts** > ...

```
16
17   AddNotification(message: string): void {
18     this.notifications.push({ message: message });
19   }
20
21   isTrue(value, message) {
22     if (value)
23       this.notifications.push({ message: message });
24   }
25
26   isRequired(value, message) {
27     if (!value || value.length <= 0)
28       this.notifications.push({ message: message });
29 }
```

Método para analisar se o valor informado é verdadeiro

# Criando o Model User

Método para analisar se o valor informado contém conteúdo

```
ts BaseValidation.ts >
PlayingScore > ps.api > src > entity > ts BaseValidation.ts > ...
1   export abstract class BaseValidation {
2
3     notifications: Array<{ message: string }>;
4
5     constructor() {
6       this.notifications = new Array<{ message: string }>();
7     }
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52 }
```

## TS BaseValidation.ts ×

```
Playingscore > ps.api > src > entity > ts BaseValidation.ts > ...
16
17   AddNotification(message: string): void {
18     this.notifications.push({ message: message });
19   }
20
21   isTrue(value, message) {
22     if (value)
23       this.notifications.push({ message: message });
24   }
25
26  isRequired(value, message) {
27     if (!value || value.length <= 0)
28       this.notifications.push({ message: message });
29 }
```

# Criando o Model User

```
ts BaseValidation.ts ×  
PlayingScore > ps.api > src > entity > ts BaseValidation.ts > ...  
1  export abstract class BaseValidation {  
2  
3      notifications: Array<{ message: string }>;  
4  
5      constructor() {  
6          this.notifications = new Array<{ message: string }>();  
7      }  
8  
9      validate(...args: any[]): void {  
10         const [value, ...rest] = args;  
11         const validationRules = value.validationRules;  
12         const validationResults: ValidationResult[] = [];  
13  
14         validationRules.forEach((rule) => {  
15             switch (rule.type) {  
16                 case 'hasMinLen':  
17                     validationResults.push({  
18                         rule,  
19                         hasMinLen(value, rule.min, rule.message)  
20                     });  
21                 case 'hasMaxLen':  
22                     validationResults.push({  
23                         rule,  
24                         hasMaxLen(value, rule.max, rule.message)  
25                     });  
26                 case 'isRegex':  
27                     validationResults.push({  
28                         rule,  
29                         isRegex(value, rule.regex, rule.message)  
30                     });  
31                 case 'isEmail':  
32                     validationResults.push({  
33                         rule,  
34                         isEmail(value, rule.message)  
35                     });  
36                 case 'isFixedLen':  
37                     validationResults.push({  
38                         rule,  
39                         isFixedLen(value, rule.length, rule.message)  
40                     });  
41                 case 'isNotEmail':  
42                     validationResults.push({  
43                         rule,  
44                         isNotEmail(value, rule.message)  
45                     });  
46                 case 'isNotEmailOrEmpty':  
47                     validationResults.push({  
48                         rule,  
49                         isNotEmailOrEmpty(value, rule.message)  
50                     });  
51             }  
52         }  
53     }  
54 }  
  
TS BaseValidation.ts ×  
PlayingScore > ps.api > src > entity > ts BaseValidation.ts > ...  
1  export abstract class BaseValidation {  
2  
3      notifications: Array<{ message: string }>;  
4  
5      constructor() {  
6          this.notifications = new Array<{ message: string }>();  
7      }  
8  
9      validate(...args: any[]): void {  
10         const [value, ...rest] = args;  
11         const validationRules = value.validationRules;  
12         const validationResults: ValidationResult[] = [];  
13  
14         validationRules.forEach((rule) => {  
15             switch (rule.type) {  
16                 case 'hasMinLen':  
17                     validationResults.push({  
18                         rule,  
19                         hasMinLen(value, rule.min, rule.message)  
20                     });  
21                 case 'hasMaxLen':  
22                     validationResults.push({  
23                         rule,  
24                         hasMaxLen(value, rule.max, rule.message)  
25                     });  
26                 case 'isRegex':  
27                     validationResults.push({  
28                         rule,  
29                         isRegex(value, rule.regex, rule.message)  
30                     });  
31                 case 'isEmail':  
32                     validationResults.push({  
33                         rule,  
34                         isEmail(value, rule.message)  
35                     });  
36                 case 'isFixedLen':  
37                     validationResults.push({  
38                         rule,  
39                         isFixedLen(value, rule.length, rule.message)  
40                     });  
41                 case 'isNotEmail':  
42                     validationResults.push({  
43                         rule,  
44                         isNotEmail(value, rule.message)  
45                     });  
46                 case 'isNotEmailOrEmpty':  
47                     validationResults.push({  
48                         rule,  
49                         isNotEmailOrEmpty(value, rule.message)  
50                     });  
51             }  
52         }  
53     }  
54 }  
  
BaseValidation.ts > 1 file
```

# Criando o Model User

```
ts BaseValidation.ts ×
PlayingScore > ps.api > src > entity > ts BaseValidation.ts > ...
1  export abstract class BaseValidation {
2
3      notifications: Array<{ message: string }>;
4
5      constructor() {
6          this.notifications = new Array<{ message: string }>();
7      }
8
9      validate(...args) {
10         const [value, ...rest] = args;
11         const validationRules = rest.map(rule);
12
13         validationRules.forEach((rule) => {
14             rule(value);
15         });
16
17         if (this.notifications.length === 0) {
18             return true;
19         }
20
21         const validationError = this.notifications[0];
22
23         if (validationError.message === undefined) {
24             throw validationError;
25         }
26
27         const validationErrorList = this.notifications.map(error => error.message);
28
29         throw validationErrorList;
30     }
31
32     hasMinLen(value, min, message) {
33         if (!value || value.length < min)
34             this.notifications.push({ message: message });
35
36         hasMaxLen(value, max, message) {
37             if (!value || value.length > max)
38                 this.notifications.push({ message: message });
39         }
40
41         isFixedLen(value, len, message) {
42             if (value.length != len)
43                 this.notifications.push({ message: message });
44         }
45
46         isEmail(value, message) {
47             var reg = new RegExp(/^\w+([-.\'])\w+@\w+([-.\'])\w+\.(\w+([-.\'])\w+)*$/);
48             if (!reg.test(value))
49                 this.notifications.push({ message: message });
50         }
51
52     }
}
```

Método para verificar se determinado valor tem tamanho mínimo

# Criando o Model User

```
ts BaseValidation.ts ×
PlayingScore > ps.api > src > entity > ts BaseValidation.ts > ...
1  export abstract class BaseValidation {
2
3      notifications: Array<{ message: string }>;
4
5      constructor() {
6          this.notifications = new Array<{ message: string }>();
7      }
8
9      validate(...args: any[]): void {
10         const [value, ...options] = args;
11         const { min, max, message } = options;
12         if (!value) {
13             this.notifications.push({ message });
14         } else if (min > value.length) {
15             this.notifications.push({ message });
16         } else if (max < value.length) {
17             this.notifications.push({ message });
18         }
19     }
20
21     isString(value: string): boolean {
22         return typeof value === 'string';
23     }
24
25     hasMinLen(value: string, min: number, message: string): void {
26         if (!value || value.length < min) {
27             this.notifications.push({ message });
28         }
29     }
30
31     hasMaxLen(value: string, max: number, message: string): void {
32         if (!value || value.length > max) {
33             this.notifications.push({ message });
34         }
35     }
36     isFixedLen(value: string, len: number, message: string): void {
37         if (value.length !== len) {
38             this.notifications.push({ message });
39         }
40     }
41     isEmail(value: string, message: string): void {
42         const reg = new RegExp(/^\\w+([-.\\.]\\w+)*@\\w+([-.]\\w+)*\\.\\w+([-.]\\w+)*$/);
43         if (!reg.test(value)) {
44             this.notifications.push({ message });
45         }
46     }
47     isCPF(value: string, message: string): void {
48         const reg = new RegExp(
49             `^\\d{3}\\.\\d{3}\\.\\d{3}[-]\\d{2}$` );
50         if (!reg.test(value)) {
51             this.notifications.push({ message });
52         }
53     }
54 }
```

Método para verificar se determinado valor tem tamanho máximo

# Criando o Model User

```
ts BaseValidation.ts ×
PlayingScore > ps.api > src > entity > ts BaseValidation.ts > ...
1  export abstract class BaseValidation {
2
3      notifications: Array<{ message: string }>;
4
5      constructor() {
6          this.notifications = new Array<{ message: string }>();
7      }
8
9      validate(...args) {
10         const [value, ...validators] = args;
11         let errors: string[] = [];
12
13         validators.forEach((validator) => {
14             const result = validator(value);
15             if (!result) {
16                 errors.push(result);
17             }
18         });
19
20         if (errors.length) {
21             throw new Error(errors.join(' - '));
22         }
23     }
24
25     hasMinLen(value, min, message) {
26         if (!value || value.length < min) {
27             this.notifications.push({ message: message });
28         }
29     }
30
31     hasMaxLen(value, max, message) {
32         if (!value || value.length > max) {
33             this.notifications.push({ message: message });
34         }
35     }
36
37     isEmail(value, message) {
38         var reg = new RegExp(/^\w+([.-. ]\w+)*@\w+([.-. ]\w+)*\.\w+([.-. ]\w+)*$/);
39         if (!reg.test(value)) {
40             this.notifications.push({ message: message });
41         }
42     }
43
44     isFixedLen(value, len, message) {
45         if (value.length != len) {
46             this.notifications.push({ message: message });
47         }
48     }
49
50     isNotEmail(value, message) {
51         var reg = new RegExp(/^\w+([.-. ]\w+)*@\w+([.-. ]\w+)*\.\w+([.-. ]\w+)*$/);
52         if (reg.test(value)) {
53             this.notifications.push({ message: message });
54         }
55     }
56 }
```

Método para verificar se determinado valor tem tamanho específico

# Criando o Model User

ts BaseValidation.ts ×  
PlayingScore > ps.api > src > entity > ts BaseValidation.ts > ...  
1 export abstract class BaseValidation {  
2  
3 notifications: Array<{ message: string }>;  
4  
5 constructor() {  
6 this.notifications = new Array<{ message: string }>();  
7 }  
8  
9 validate(...)  
10 )  
11 get value(): string | undefined {  
12 return undefined;  
13 }  
14 set value(value: string) {  
15 this.validate(value);  
16 }  
17 AddNotification(message: string): void {  
18 this.notifications.push({ message });  
19 }  
20 isRequired(): boolean {  
21 return false;  
22 }  
23 hasMinLen(min: number, message: string): void {  
24 if (!value || value.length < min) {  
25 this.notifications.push({ message });  
26 }  
27 }  
28 hasMaxLen(max: number, message: string): void {  
29 if (!value || value.length > max) {  
30 this.notifications.push({ message });  
31 }  
32 }  
33 isFixedLen(len: number, message: string): void {  
34 if (value.length != len) {  
35 this.notifications.push({ message });  
36 }  
37 }  
38 hasEmailFormat(message: string): void {  
39 if (!value || !isEmail(value)) {  
40 this.notifications.push({ message });  
41 }  
42 }  
43 isEmail(value: string, message: string): void {  
44 const reg = new RegExp(/^[\w+([-.\'])\w+]@[\w+([-.\'])\w+]\*\.\w+([-.\'])\w+)\$/);  
45 if (!reg.test(value)) {  
46 this.notifications.push({ message });  
47 }  
48 }  
49 validateEmail(): void {  
50 this.hasEmailFormat("Email");  
51 }  
52}

Método para verificar se determinado valor, através de uma expressão regular, é um e-mail válido

# Criando o Model User

---

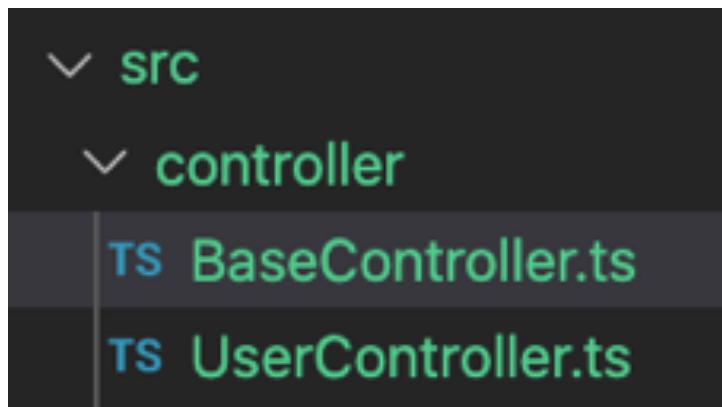
Até aqui temos um entidade base e um validador base

- Agora, vamos projetar um controlador base
- Notem que estamos preparando toda a base da API, para depois iniciarmos a programação, de fato, das regras de negócio de nosso aplicativo

# Criando o Model User

Até aqui temos um entidade base e um validador base

- Agora, vamos projetar um controlador base
- Notem que estamos preparando toda a base da API, para depois iniciarmos a programação, de fato, das regras de negócio de nosso aplicativo



Criar BaseController.ts

# Criando o Model User

Até aqui temos um entidade base e um validador base

- Agora, vamos projetar um controlador base
- Notem que estamos preparando toda a base da API, para depois iniciarmos a programação, de fato, das regras de negócio de nosso aplicativo

TS BaseController.ts ×

PlayingScore > ps.api > src > controller > TS BaseController

```
1  export abstract class BaseController {  
2  
3  }
```

Esta classe também será abstrata e deverá conter tudo que a classe UserController tem, porém de forma genérica

# Criando o Model User

TS BaseController.ts X

```
PlayingScore > ps.api > src > controller > TS BaseController.ts > ...
1 import { Repository, getRepository } from 'typeorm';
2 import { NextFunction, Request, Response } from 'express';
3
4 export abstract class BaseController<GENERIC_CLASS> {
5
6     private _repository: Repository<GENERIC_CLASS>;
7
8     constructor(entity: any) {
9         this._repository = getRepository<GENERIC_CLASS>(entity);
10    }
11
12    async all(request: Request, response: Response, next: NextFunction) {
13        return this._repository.find();
14    }
15
16    async one(request: Request, response: Response, next: NextFunction) {
17        return this._repository.findOne(request.params.id);
18    }
19
20    async save(request: Request, response: Response, next: NextFunction) {
21        return this._repository.save(request.body);
22    }
23
24    async remove(request: Request, response: Response, next: NextFunction) {
25        let userToRemove = await this._repository.findOne(request.params.id);
26        await this._repository.remove(userToRemove);
27    }
28
29 }
```

# Criando o Model User

TS BaseController.ts X

```
PlayingScore > ps.api > src > controller > TS BaseController.ts > ...
1 import { Repository, getRepository } from 'typeorm';
2 import { NextFunction, Request, Response } from 'express';
3
4 export abstract class BaseController<GENERIC_CLASS> {
5
6     private _repository: Repository<GENERIC_CLASS>;
7
8     constructor(entity: any) {
9         this._repository = getRepository<GENERIC_CLASS>(entity);
10    }
11
12    async all(request: Request, response: Response, next: NextFunction) {
13        return this._repository.find();
14    }
15
16    async one(request: Request, response: Response, next: NextFunction) {
17        return this._repository.findOne(request.params.id);
18    }
19
20    async save(request: Request, response: Response, next: NextFunction) {
21        return this._repository.save(request.body);
22    }
23
24    async remove(request: Request, response: Response, next: NextFunction) {
25        let userToRemove = await this._repository.findOne(request.params.id);
26        await this._repository.remove(userToRemove);
27    }
28
29 }
```

Basicamente, colei todos os métodos do UserController e configurei para um repositório genérico

# Criando o Model User

---

Até aqui temos um entidade base e um validador base

- Agora vamos utilizar esta classe genérico no UserController

# Criando o Model User

Até aqui temos um entidade base e um validador base

- Agora vamos utilizar esta classe genérico no UserController

```
TS UserController.ts ×

PlayingScore > ps.api > src > controller > TS UserController.ts > ...

2 import { User } from "../entity/User";
3 import { BaseController } from "./BaseController";
4
5 ✓ export class UserController extends BaseController<User> {
6
7   ✓ constructor() {
8     super(User);
9   }
10
11 }
```

# Criando o Model User

Até aqui temos um entidade base e um validador base

- Agora vamos utilizar esta classe genérico no UserController

```
TS UserController.ts ×

PlayingScore > ps.api > src > controller > TS UserController.ts > ...

2 import { User } from "../entity/User";
3 import { BaseController } from "./BaseController";
4
5 ✓ export class UserController extends BaseController<User> {
6
7   ✓ constructor() {
8     super(User);
9   }
10
11 }
```

Usamos o extends para tratar o BaseController como classe "pai"

# Criando o Model User

Até aqui temos um entidade base e um validador base

- Agora vamos utilizar esta classe genérico no UserController

```
TS UserController.ts ×

PlayingScore > ps.api > src > controller > TS UserController.ts > ...

2 import { User } from "../entity/User";
3 import { BaseController } from "./BaseController";
4
5 ✓ export class UserController extends BaseController<User> {
6
7   ✓ constructor() {
8     super(User);
9   }
10
11 }
```

Informamos que a classe  
(genérica) é a entidade User

# Criando o Model User

Até aqui temos um entidade base e um validador base

- Agora vamos utilizar esta classe genérico no UserController

```
TS UserController.ts ×

PlayingScore > ps.api > src > controller > TS UserController.ts > ...

2 import { User } from "../entity/User";
3 import { BaseController } from "./BaseController";
4
5 ✓ export class UserController extends BaseController<User> {
6
7   ✓ constructor() {
8     super(User);
9   }
10
11 }
```

No construtor passamos o tipo  
da classe genérica para nossa  
classe "pai"

# Criando o Model User

Até aqui temos um entidade base e um validador base

- Agora vamos utilizar esta classe genérico no UserController

```
TS UserController.ts ×

PlayingScore > ps.api > src > controller > TS UserController.ts > ...

2 import { User } from "../entity/User";
3 import { BaseController } from "./BaseController";
4
5 ✓ export class UserController extends BaseController<User> {
6
7   ✓ constructor() {
8     super(User);
9   }
10 }
11 }
```

Sempre que usarmos métodos de classes extends, temos que informar o super no construtor

# Criando o Model User

---

Até aqui temos um entidade base e um validador base

- No BaseController.ts vamos fazer o uso do BaseValidation.ts

# Criando o Model User

```
TS BaseController.ts ×

PlayingScore > ps.api > src > controller > TS BaseController.ts > ...
1  import { Repository, getRepository } from 'typeorm';
2  import { NextFunction, Request, Response } from 'express';
3  import { BaseValidation } from '../entity/BaseValidation';
4
5  export abstract class BaseController<GENERIC_CLASS> extends BaseValidation {
6
7      private _repository: Repository<GENERIC_CLASS>;
8
9      constructor(entity: any) {
10          super();
11          this._repository = getRepository<GENERIC_CLASS>(entity);
12      }
13
14      async all() {
15          return this._repository.find();
16      }
17
18      async one(request: Request) {
19          return this._repository.findOne(request.params.id);
20      }
21
22      async save(model: any) {
23          // Se for atualização
24          if (model.uid) {
25              let _modelRegister = await this._repository.findOne(model.uid);
26              // se encontrar o registro, usa o assign para mesclar os dados alterados
27              if (_modelRegister) Object.assign(_modelRegister, model);
28          }
29          if (this.valid())
30              return this._repository.save(model);
31          else
32              return {
33                  status: 400,
34                  errors: this.allNotifications
35              }
36      }
37
38      async remove(request: Request) {
39          // não excluimos registros, apenas deixamos inativos
40          let _modelRegister: any;
41          _modelRegister = await this._repository.find(request.params.id);
42          if (_modelRegister) _modelRegister.isActive = false;
43          return await this._repository.save(_modelRegister);
44      }
45
46  }
```

# Criando o Model User

```
TS BaseController.ts ×  
PlayingScore > ps.api > src > controller > TS BaseController.ts > ...  
1 import { Repository, getRepository } from 'typeorm';  
2 import { NextFunction, Request, Response } from 'express';  
3 import { BaseValidation } from '../entity/BaseValidation';  
4  
5 export abstract class BaseController<GENERIC_CLASS> extends BaseValidation {  
6  
7     private _repository: Repository<GENERIC_CLASS>;  
8  
9     constructor(entity: any) {  
10         super();  
11         this._repository = getRepository<GENERIC_CLASS>(entity);  
12     }  
13  
14     async all() {  
15         return this._repository.find();  
16     }  
17  
18     async one(request: Request) {  
19         return this._repository.findOne(request.params.id);  
20     }  
21  
22     async save(model: any) {  
23         // Se for atualização  
24         if (model.uid) {  
25             let _modelRegister = await this._repository.findOne(model.uid);  
26             // se encontrar o registro, usa o assign para mesclar os dados alterados  
27             if (_modelRegister) Object.assign(_modelRegister, model);  
28         }  
29         if (this.valid())  
30             return this._repository.save(model);  
31         else  
32             return {  
33                 status: 400,  
34                 errors: this.allNotifications  
35             }  
36     }  
37  
38     async remove(request: Request) {  
39         // não excluímos registros, apenas deixamos inativos  
40         let _modelRegister: any;  
41         _modelRegister = await this._repository.find(request.params.id);  
42         if (_modelRegister) _modelRegister.isActive = false;  
43         return await this._repository.save(_modelRegister);  
44     }  
45  
46 }
```

Adicionamos extends do BaseValidation

# Criando o Model User

```
TS BaseController.ts ×  
PlayingScore > ps.api > src > controller > TS BaseController.ts > ...  
1 import { Repository, getRepository } from 'typeorm';  
2 import { NextFunction, Request, Response } from 'express';  
3 import { BaseValidation } from '../entity/BaseValidation';  
4  
5 export abstract class BaseController<GENERIC_CLASS> extends BaseValidation {  
6  
7     private _repository: Repository<GENERIC_CLASS>;  
8  
9     constructor(entity: any) {  
10         super();  
11         this._repository = getRepository<GENERIC_CLASS>(entity);  
12     }  
13  
14     async all() {  
15         return this._repository.find();  
16     }  
17  
18     async one(request: Request) {  
19         return this._repository.findOne(request.params.id);  
20     }  
21  
22     async save(model: any) {  
23         // Se for atualização  
24         if (model.uid) {  
25             let _modelRegister = await this._repository.findOne(model.uid);  
26             // se encontrar o registro, usa o assign para mesclar os dados alterados  
27             if (_modelRegister) Object.assign(_modelRegister, model);  
28         }  
29         if (this.valid())  
30             return this._repository.save(model);  
31         else  
32             return {  
33                 status: 400,  
34                 errors: this.allNotifications  
35             }  
36     }  
37  
38     async remove(request: Request) {  
39         // não excluímos registros, apenas deixamos inativos  
40         let _modelRegister: any;  
41         _modelRegister = await this._repository.find(request.params.id);  
42         if (_modelRegister) _modelRegister.isActive = false;  
43         return await this._repository.save(_modelRegister);  
44     }  
45  
46 }
```

No construtor executamos o super para que o construtor "pai" execute antes

# Criando o Model User

```
TS BaseController.ts ×

PlayingScore > ps.api > src > controller > TS BaseController.ts > ...
1  import { Repository, getRepository } from 'typeorm';
2  import { NextFunction, Request, Response } from 'express';
3  import { BaseValidation } from '../entity/BaseValidation';
4
5  export abstract class BaseController<GENERIC_CLASS> extends BaseValidation {
6
7      private _repository: Repository<GENERIC_CLASS>;
8
9      constructor(entity: any) {
10          super();
11          this._repository = getRepository<GENERIC_CLASS>(entity);
12      }
13
14      async all() {
15          return this._repository.find();
16      }
17
18      async one(request: Request) {
19          return this._repository.findOne(request.params.id);
20      }
21
22      async save(model: any) {
23          // Se for atualização
24          if (model.uid) {
25              let _modelRegister = await this._repository.findOne(model.uid);
26              // se encontrar o registro, usa o assign para mesclar os dados alterados
27              if (_modelRegister) Object.assign(_modelRegister, model);
28          }
29          if (this.valid())
30              return this._repository.save(model);
31          else
32              return {
33                  status: 400,
34                  errors: this.allNotifications
35              }
36      }
37
38      async remove(request: Request) {
39          // não excluímos registros, apenas deixamos inativos
40          let _modelRegister: any;
41          _modelRegister = await this._repository.find(request.params.id);
42          if (_modelRegister) _modelRegister.isActive = false;
43          return await this._repository.save(_modelRegister);
44      }
45
46  }
```

Retiramos os elementos não utilizados dos parâmetros dos métodos

# Criando o Model User

```
ts BaseController.ts ×

PlayingScore > ps.api > src > controller > ts BaseController.ts > ...
1 import { Repository, getRepository } from 'typeorm';
2 import { NextFunction, Request, Response } from 'express';
3 import { BaseValidation } from '../entity/BaseValidation';
4
5 export abstract class BaseController<GENERIC_CLASS> extends BaseValidation {
6
7     private _repository: Repository<GENERIC_CLASS>;
8
9     constructor(entity: any) {
10         super();
11         this._repository = getRepository<GENERIC_CLASS>(entity);
12     }
13
14     async all() {
15         return this._repository.find();
16     }
17
18     async one(request: Request) {
19         return this._repository.findOne(request.params.id);
20     }
21
22     async save(model: any) {
23         // Se for atualização
24         if (model.uid) {
25             let _modelRegister = await this._repository.findOne(model.uid);
26             // se encontrar o registro, usa o assign para mesclar os dados alterados
27             if (_modelRegister) Object.assign(_modelRegister, model);
28         }
29         if (this.valid())
30             return this._repository.save(model);
31         else
32             return {
33                 status: 400,
34                 errors: this.allNotifications
35             }
36     }
37
38     async remove(request: Request) {
39         // não excluímos registros, apenas deixamos inativos
40         let _modelRegister: any;
41         _modelRegister = await this._repository.find(request.params.id);
42         if (_modelRegister) _modelRegister.isActive = false;
43         return await this._repository.save(_modelRegister);
44     }
45
46 }
```

Refatoraremos o método save conforme necessidade: editar ou inserir

# Criando o Model User

```
TS BaseController.ts ×

PlayingScore > ps.api > src > controller > TS BaseController.ts > ...
1  import { Repository, getRepository } from 'typeorm';
2  import { NextFunction, Request, Response } from 'express';
3  import { BaseValidation } from '../entity/BaseValidation';
4
5  export abstract class BaseController<GENERIC_CLASS> extends BaseValidation {
6
7      private _repository: Repository<GENERIC_CLASS>;
8
9      constructor(entity: any) {
10          super();
11          this._repository = getRepository<GENERIC_CLASS>(entity);
12      }
13
14      async all() {
15          return this._repository.find();
16      }
17
18      async one(request: Request) {
19          return this._repository.findOne(request.params.id);
20      }
21
22      async save(model: any) {
23          // Se for atualização
24          if (model.uid) {
25              let _modelRegister = await this._repository.findOne(model.uid);
26              // se encontrar o registro, usa o assign para mesclar os dados alterados
27              if (_modelRegister) Object.assign(_modelRegister, model);
28
29          if (this.valid())
30              return this._repository.save(model);
31          else
32              return {
33                  status: 400,
34                  errors: this.allNotifications
35              }
36      }
37
38      async remove(request: Request) {
39          // não excluímos registros, apenas deixamos inativos
40          let _modelRegister: any;
41          _modelRegister = await this._repository.find(request.params.id);
42          if (_modelRegister) _modelRegister.isActive = false;
43          return await this._repository.save(_modelRegister);
44      }
45
46  }
```

Analisamos se os dados são válidos

# Criando o Model User

```
TS BaseController.ts ×

PlayingScore > ps.api > src > controller > TS BaseController.ts > ...
1  import { Repository, getRepository } from 'typeorm';
2  import { NextFunction, Request, Response } from 'express';
3  import { BaseValidation } from '../entity/BaseValidation';
4
5  export abstract class BaseController<GENERIC_CLASS> extends BaseValidation {
6
7      private _repository: Repository<GENERIC_CLASS>;
8
9      constructor(entity: any) {
10          super();
11          this._repository = getRepository<GENERIC_CLASS>(entity);
12      }
13
14      async all() {
15          return this._repository.find();
16      }
17
18      async one(request: Request) {
19          return this._repository.findOne(request.params.id);
20      }
21
22      async save(model: any) {
23          // Se for atualização
24          if (model.uid) {
25              let _modelRegister = await this._repository.findOne(model.uid);
26              // se encontrar o registro, usa o assign para mesclar os dados alterados
27              if (_modelRegister) Object.assign(_modelRegister, model);
28          }
29          if (this.valid())
30              return this._repository.save(model);
31          else
32              return {
33                  status: 400,
34                  errors: this.allNotifications
35              }
36      }
37
38      async remove(request: Request) {
39          // não excluímos registros, apenas deixamos inativos
40          let _modelRegister: any;
41          _modelRegister = await this._repository.find(request.params.id);
42          if (_modelRegister) _modelRegister.isActive = false;
43          return await this._repository.save(_modelRegister);
44      }
45
46  }
```

Refatoramos o método remove,  
pois não removemos registros,  
apenas inativamos

Na linha 41 utilizar findOne( )

# Criando o Mod

```
TS BaseController.ts ×  
PlayingScore > ps.api > src > controller > TS BaseController.ts > ...  
1 import { Repository, getRepository } from 'typeorm';  
2 import { NextFunction, Request, Response } from 'express';  
3 import { BaseValidation } from '../entity/BaseValidation';  
4  
5 export abstract class BaseController<GENERIC_CLASS> extends BaseValidation {
```

TS BaseController.ts ×

```
PlayingScore > ps.api > src > controller > TS BaseController.ts > ...  
1 import { Repository, getRepository } from 'typeorm';  
2 import { NextFunction, Request, Response } from 'express';  
3 import { BaseValidation } from '../entity/BaseValidation';  
4  
5 export abstract class BaseController<GENERIC_CLASS> extends BaseValidation {  
6  
7     private _repository: Repository<GENERIC_CLASS>;  
8  
9     constructor(entity: any) {  
10         super();  
11         this._repository = getRepository<GENERIC_CLASS>(entity);  
12     }  
13 }
```

```
38     async remove(request: Request) {  
39         // não excluímos registros, apenas deixamos inativos  
40         let _modelRegister: any;  
41         _modelRegister = await this._repository.find(request.params.id);  
42         if (_modelRegister) _modelRegister.isActive = false;  
43         return await this._repository.save(_modelRegister);  
44     }  
45 }  
46 }
```

Refatoramos o método remove,  
pois não removemos registros,  
apenas inativamos

# Criando o Model User

```
TS BaseController.ts ×

PlayingScore > ps.api > src > controller > TS BaseController.ts > ...
1  import { Repository, getRepository } from 'typeorm';
2  import { NextFunction, Request, Response } from 'express';
3  import { BaseValidation } from '../entity/BaseValidation';
4
5  export abstract class BaseController<GENERIC_CLASS> extends BaseValidation {
6
7      private _repository: Repository<GENERIC_CLASS>;
8
9      constructor(entity: any) {
10         super();
11         this._repository = getRepository<GENERIC_CLASS>(entity);
12     }
13
14     async all() {
15         return this._repository.find();
16     }
17
18     async one(request: Request) {
19         return this._repository.findOne(request.params.id);
20     }
21
22     else
23         return {
24             status: 400,
25             errors: this.allNotifications
26         }
27
28     async remove(request: Request) {
29         // não excluímos registros, apenas deixamos inativos
30         let _modelRegister: any;
31         _modelRegister = await this._repository.find(request.params.id);
32         if (_modelRegister) _modelRegister.isActive = false;
33         return await this._repository.save(_modelRegister);
34     }
35
36 }
37
38 }
```

Refatoramos o método remove,  
pois não removemos registros,  
apenas inativamos

Na linha 41 utilizar findOne( )

# Criando o

M

```
22     async save(model: any) {
23         // Se for atualização
24         if (model.uid) {
25             let _modelRegister = await this._repository.findOne(model.uid);
26             // se encontrar o registro, usa o assign para mesclar os dados alterados
27             if (_modelRegister) Object.assign(_modelRegister, model);
28         }
29         if (this.valid())
30             return this._repository.save(model);
31         else
32             return {
33                 status: 400,
34                 errors: this.allNotifications
35             }
36     }
```

```
38     async remove(request: Request) {
39         // não excluímos registros, apenas deixamos inativos
40         let _modelRegister: any;
41         _modelRegister = await this._repository.find(request.params.id);
42         if (_modelRegister) _modelRegister.isActive = false;
43         return await this._repository.save(_modelRegister);
44     }
45 }
46 }
```

Refatoramos o método remove,  
pois não removemos registros,  
apenas inativamos

# Criando o Model User

```
ts BaseController.ts ×

PlayingScore > ps.api > src > controller > ts BaseController.ts > ...
1 import { Repository, getRepository } from 'typeorm';
2 import { NextFunction, Request, Response } from 'express';
3 import { BaseValidation } from '../entity/BaseValidation';
4
5 export abstract class BaseController<GENERIC_CLASS> extends BaseValidation {
6
7     private _repository: Repository<GENERIC_CLASS>;
8
9     constructor(entity: any) {
10         super();
11         this._repository = getRepository<GENERIC_CLASS>(entity);
12     }
13
14     async all() {
15
16         return await this._repository.find();
17     }
18
19     async remove(request: Request) {
20         // não excluímos registros, apenas deixamos inativos
21         let _modelRegister: any;
22         _modelRegister = await this._repository.find(request.params.id);
23         if (_modelRegister) _modelRegister.isActive = false;
24         return await this._repository.save(_modelRegister);
25     }
26
27     async removeById(id: string) {
28         let model: any;
29         model = await this._repository.findOne(id);
30         if (!model) {
31             return {
32                 status: 404,
33                 errors: this.allNotifications
34             }
35         }
36         return await this._repository.remove(model);
37     }
38
39     async remove(request: Request) {
40         // não excluímos registros, apenas deixamos inativos
41         let _modelRegister: any;
42         _modelRegister = await this._repository.find(request.params.id);
43         if (_modelRegister) _modelRegister.isActive = false;
44         return await this._repository.save(_modelRegister);
45     }
46 }
```

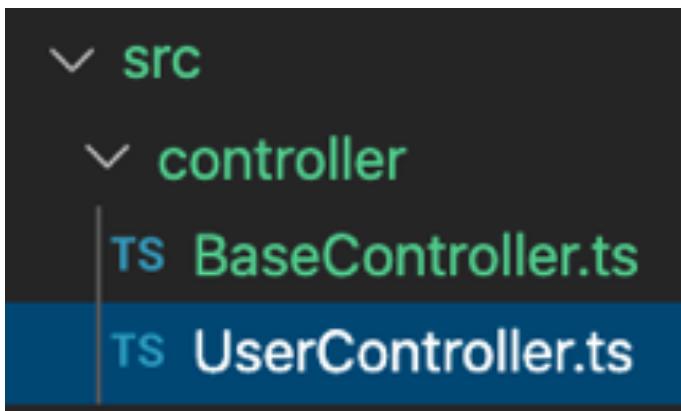
Refatoramos o método remove,  
pois não removemos registros,  
apenas inativamos

Na linha 41 utilizar findOne( )

# Criando o Model User

Para salvar os dados do usuário, precisamos passar a camada de validação antes

- Então, vamos fazer um overwrite no save



# Criando o Model User

---

Para salvar os dados do usuário, precisamos passar a camada de validação antes

- Então, vamos fazer um overwrite no save

# Criando o Model User

TS UserController.ts ×

```
PlayingScore > ps.api > src > controller > TS UserController.ts > ...
  2 import { getRepository } from "typeorm";
  3 import { User } from "../entity/User";
  4 import { BaseController } from "./BaseController";
  5
  6 export class UserController extends BaseController<User> {
  7
  8     constructor() {
  9         super(User);
 10    }
 11
 12    async save(request: Request) {
 13        let _user = <User>request.body;
 14        super.isRequired(_user.name, 'Informe seu nome!');
 15        super.isRequired(_user.nick, 'Informe seu apelido!');
 16        super.isRequired(_user.email, 'Informe seu e-mail!');
 17        super.isRequired(_user.password, 'Informe sua senha!');
 18        return super.save(_user);
 19    }
 20
 21 }
```

# Criando o Model User

TS UserController.ts ×

PlayingScore > ps.api > src > controller > **TS UserController.ts** > ...

```
2 import { getRepository } from "typeorm";
3 import { User } from "../entity/User";
4 import { BaseController } from "./BaseController";
5
6 export class UserController extends BaseController<User> {
7
8     constructor() {
9         super(User);
10    }
11
12    async save(request: Request) {
13        let _user = <User>request.body;
14        super.isRequired(_user.name, 'Informe seu nome!');
15        super.isRequired(_user.nick, 'Informe seu apelido!');
16        super.isRequired(_user.email, 'Informe seu e-mail!');
17        super.isRequired(_user.password, 'Informe sua senha!');
18        return super.save(_user);
19    }
20
21 }
```

Criamos método overwrite  
"save"

# Criando o Model User

TS UserController.ts ×

PlayingScore > ps.api > src > controller > **TS UserController.ts** > ...

```
2 import { getRepository } from "typeorm";
3 import { User } from "../entity/User";
4 import { BaseController } from "./BaseController";
5
6 export class UserController extends BaseController<User> {
7
8     constructor() {
9         super(User);
10    }
11
12    async save(request: Request) {
13        let _user = <User>request.body;
14        super.isRequired(_user.name, 'Informe seu nome!');
15        super.isRequired(_user.nick, 'Informe seu apelido!');
16        super.isRequired(_user.email, 'Informe seu e-mail!');
17        super.isRequired(_user.password, 'Informe sua senha!');
18        return super.save(_user);
19    }
20}
21}
```

Cast dos dados da requisição  
para classe User

# Criando o Model User

TS UserController.ts ×

PlayingScore > ps.api > src > controller > **TS UserController.ts** > ...

```
2 import { getRepository } from "typeorm";
3 import { User } from "../entity/User";
4 import { BaseController } from "./BaseController";
5
6 export class UserController extends BaseController<User> {
7
8     constructor() {
9         super(User);
10    }
11
12    async save(request: Request) {
13        let _user = <User>request.body;
14        super.isRequired(_user.name, 'Informe seu nome!');
15        super.isRequired(_user.nick, 'Informe seu apelido!');
16        super.isRequired(_user.email, 'Informe seu e-mail!');
17        super.isRequired(_user.password, 'Informe sua senha!');
18        return super.save(_user);
19    }
20}
21}
```

Defino os campos obrigatórios

# Criando o Model User

TS UserController.ts ×

PlayingScore > ps.api > src > controller > **TS UserController.ts** > ...

```
2 import { getRepository } from "typeorm";
3 import { User } from "../entity/User";
4 import { BaseController } from "./BaseController";
5
6 export class UserController extends BaseController<User> {
7
8     constructor() {
9         super(User);
10    }
11
12    async save(request: Request) {
13        let _user = <User>request.body;
14        super.isRequired(_user.name, 'Informe seu nome!');
15        super.isRequired(_user.nick, 'Informe seu apelido!');
16        super.isRequired(_user.email, 'Informe seu e-mail!');
17        super.isRequired(_user.password, 'Informe sua senha!');
18        return super.save(_user);
19    }
20}
21}
```

Executo o método save da classe super (pai) para, de fato, salvar os dados

# Criando o Model User

TS UserController.ts ×

```
PlayingScore > ps.api > src > controller > TS UserController.ts > ...
2   import { getRepository } from "typeorm";
3   import { User } from "../entity/User";
4   import { BaseController } from "./BaseController";
5
6   export class UserController extends BaseController<User> {
7
8     constructor() {
9       super(User);
10    }
11
12    async save(request: Request) {
13      let _user = <User>request.body;
14      super.isRequired(_user.name, 'Informe seu nome!');
15      super.isRequired(_user.nick, 'Informe seu apelido!');
16      super.isRequired(_user.email, 'Informe seu e-mail!');
17      super.isRequired(_user.password, 'Informe sua senha!');
18      return super.save(_user);
19    }
20
21 }
```

Note que não atribuímos os campos nullable na entity, precisamos fazer isso

# Criando o Model User

Vamos ajustar nossa entidade para adicionar os campos que podem ser nulos

```
TS BaseEntity.ts ×  
PlayingScore > ps.api > src > entity > TS BaseEntity.ts > ...  
1 import { PrimaryGeneratedColumn, Column, CreateDateColumn, UpdateDateColumn } from "typeorm";  
2  
3 export abstract class BaseEntity {  
4  
5     @PrimaryGeneratedColumn("uuid")  
6     uid: string;  
7  
8     @Column({ default: true })  
9     isPublic: boolean;  
10  
11    @Column({ default: true })  
12    isActive: boolean;  
13  
14    @CreateDateColumn({ type: 'timestamp', nullable: true })  
15    createdAt: Date;  
16  
17    @UpdateDateColumn({ type: 'timestamp', nullable: true })  
18    updatedAt: Date;  
19  
20}
```

# Criando o Model User

Vamos ajustar nossa entidade para adicionar os campos que podem ser nulos

```
TS BaseEntity.ts ×  
PlayingScore > ps.api > src > entity > TS BaseEntity.ts > ...  
1 import { PrimaryGeneratedColumn, Column, CreateDateColumn, UpdateDateColumn } from "typeorm";  
2  
3 export abstract class BaseEntity {  
4  
5     @PrimaryGeneratedColumn("uuid")  
6     uid: string;  
7  
8     @Column({ default: true })  
9     isPublic: boolean;  
10  
11    @Column({ default: true })  
12    isActive: boolean;  
13  
14    @CreateDateColumn({ type: 'timestamp', nullable: true })  
15    createdAt: Date;  
16  
17    @UpdateDateColumn({ type: 'timestamp', nullable: true })  
18    updatedAt: Date;  
19  
20 }
```

Primeiro deixamos nullable os atributos timestamp da classe base

# Criando o Model User

```
ts User.ts  X  
PlayingScore > ps.api > src > entity > ts User.ts > ...  
1  import {Entity, PrimaryGeneratedColumn, Column} from "typeorm";  
2  import { BaseEntity } from "./BaseEntity";  
3  
4  @Entity({ name: 'User' })  
5  export class User extends BaseEntity {  
6  
7      @Column({ type: 'varchar', length: 100 })  
8      name: string;  
9  
10     @Column({ type: 'varchar', length: 40 })  
11     nick: string;  
12  
13     @Column({ type: 'varchar', length: 100, nullable: true })  
14     photo: string;  
15  
16     @Column({ type: 'varchar', length: 100 })  
17     email: string;  
18  
19     @Column({ type: 'varchar', length: 100, nullable: true })  
20     nick_instagram: string;  
21  
22     @Column({ type: 'varchar', length: 100, nullable: true })  
23     nick_facebook: string;  
24  
25     @Column({ type: 'varchar', length: 100, nullable: true })  
26     whatsapp: string;  
27  
28     @Column({ type: 'varchar', length: 100 })  
29     password: string;  
30  
31 }
```

Depois, ajustamos a classe user

# Criando o Model User

---

Vamos ajustar nossa entidade para adicionar os campos que podem ser nulos

- Agora deve-se fazer o drop da tabela e executar o `npm start` novamente

# Criando o Model User

---

Vamos ajustar nossa entidade para adicionar os campos que podem ser nulos

- Agora deve-se fazer o drop da tabela e executar o `npm start` novamente



# Criando Model Game

# Criando Model Game

Vamos projetar nosso CRUD de jogos

- Para isso, vamos criar uma entidade chama Game.ts



The screenshot shows a file explorer interface with a dark theme. The 'src' directory is expanded, revealing its contents. Inside 'src', there are two main folders: 'controller' and 'entity'. The 'entity' folder is also expanded, showing four files: 'BaseEntity.ts', 'BaseValidation.ts', 'Game.ts', and 'User.ts'. The 'Game.ts' file is highlighted with a blue selection bar at the bottom of the list.

```
src
  controller
  entity
    BaseEntity.ts
    BaseValidation.ts
    Game.ts
    User.ts
```

# Criando Model Game

```
TS game.ts  X

PlayingScore > ps.api > src > entity > TS game.ts > ...
1  import { BaseEntity } from "./BaseEntity";
2  import { Entity, Column } from "typeorm";
3
4  @Entity({ name: 'Game' })
5  export default class Game extends BaseEntity {
6
7      @Column({ type: 'varchar', length: 100 })
8      name: string;
9
10     @Column({ type: 'varchar', length: 500 })
11     description: string;
12
13     @Column({ type: 'varchar', length: 500, nullable: true })
14     link_rules: string;
15
16     @Column()
17     teams_limit: number;
18
19     @Column()
20     players_per_team_limit: number;
21
22 }
```

# Criando Model Game

Vamos projetar nosso CRUD de jogos

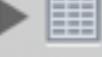
- Para isso, vamos criar uma entidade chama Game.ts

Executando  
npm start

▼  playing\_score

▼  Tables

►  Game

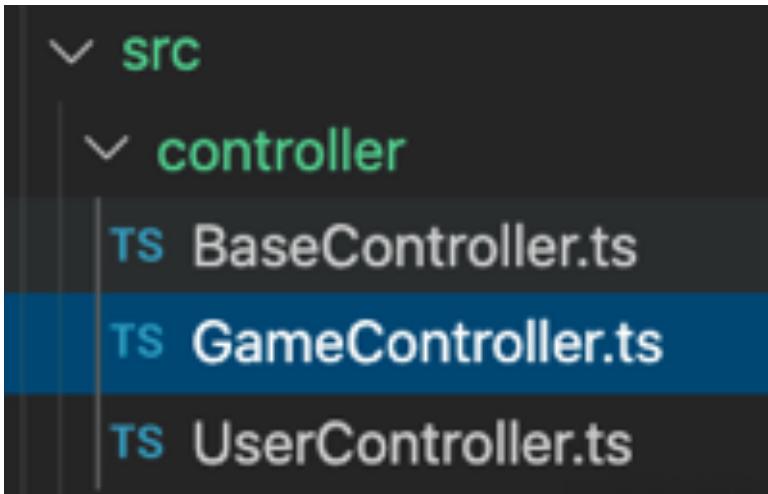
►  User

Column	Datatype
uid	VARCHAR(36)
isPublic	TINYINT(4)
isActive	TINYINT(4)
createdAt	TIMESTAMP(6)
updatedAt	TIMESTAMP(6)
name	VARCHAR(100)
description	VARCHAR(500)
link_rules	VARCHAR(500)
teams_limit	INT(11)
players_per_t...	INT(11)

# Criando Model Game

Vamos projetar nosso CRUD de jogos

- Com a entidade criada, vamos criar o GameController.ts



# Criando Model Game

Vamos projetar nosso CRUD de jogos

- Com a entidade criada, vamos criar o GameController.ts

```
TS GameController.ts X

PlayingScore > ps.api > src > controller > TS GameController.ts > ...

1 import { Request } from 'express';
2 import Game from "../entity/Game";
3 import { BaseController } from "./BaseController";
4
5 export class GameController extends BaseController<Game> {
6
7     constructor() {
8         super(Game);
9     }
10
11    async save(request: Request) {
12        let _game = <Game>request.body;
13        super.isRequired(_game.name, 'Informe o nome do jogo!');
14        super.isRequired(_game.description, 'Informe a descrição do jogo!');
15        super.isRequired(_game.teams_limit, 'Informe o limite de equipes!');
16        super.isRequired(_game.players_per_team_limit, 'Informe o limite de jogadores por equipe!');
17        return super.save(_game);
18    }
19
20 }
```

# Criando Model Game

Vamos projetar nosso CRUD de jogos

- Agora, demos ajustar nossas rotas de games
- Para isso, vamos alterar o arquivo routes.ts

The screenshot shows a file explorer interface with a dark theme. The 'src' directory is expanded, revealing its contents. Inside 'src', there are four sub-directories: 'controller', 'entity', 'include', and 'migration'. Below these, there are two files: 'index.ts' and 'routes.ts'. The 'routes.ts' file is highlighted with a blue bar at the bottom of the list, indicating it is the current file being edited.

```
src
  controller
  entity
  include
  migration
  index.ts
  routes.ts
```

# Criando Model Game

Vamos projetar nosso CRUD de jogos

- Agora, demos ajustar nossas rotas de games
- Para isso, vamos alterar o arquivo routes.ts

```
TS routes.ts X

PlayingScore > ps.api > src > TS routes.ts > ...

1 import { UserController } from "./controller/UserController";
2 import { GameController } from "./controller/GameController";
3
4 export const Routes = [
5     { method: "get", route: "/users", controller: UserController, action: "all" },
6     { method: "get", route: "/users/:id", controller: UserController, action: "one" },
7     { method: "post", route: "/users", controller: UserController, action: "save" },
8     { method: "delete", route: "/users/:id", controller: UserController, action: "remove" },
9
10    { method: "get", route: "/games", controller: GameController, action: "all" },
11    { method: "get", route: "/games/:id", controller: GameController, action: "one" },
12    { method: "post", route: "/games", controller: GameController, action: "save" },
13    { method: "delete", route: "/games/:id", controller: GameController, action: "remove" }
14];
```

# Criando Model Game

Vamos projetar nosso CRUD de jogos

- Agora, demos ajustar nossas rotas de games
- Para isso, vamos alterar o arquivo routes.ts

TS routes.ts X

```
PlayingScore > ps.api > src > TS routes.ts > ...
1 import { UserController } from "./controller/UserController";
2 import { GameController } from "./controller/GameController";
3
4 export const Routes = [
5   { method: "get", route: "/users", controller: UserController, action: "all" },
6   { method: "get", route: "/users/:id", controller: UserController, action: "one" },
7   { method: "post", route: "/users", controller: UserController, action: "save" },
8   { method: "delete", route: "/users/:id", controller: UserController, action: "remove" },
9
10  { method: "get", route: "/games", controller: GameController, action: "all" },
11  { method: "get", route: "/games/:id", controller: GameController, action: "one" },
12  { method: "post", route: "/games", controller: GameController, action: "save" },
13  { method: "delete", route: "/games/:id", controller: GameController, action: "remove" }
14];
```

Teremos 2 rotas do tipo GET,  
1 POST e 1 DELETE

# Criando Model Game

Vamos projetar nosso CRUD de jogos

- Agora, demos ajustar nossas rotas de games
- Para isso, vamos alterar o arquivo routes.ts

TS routes.ts X

```
PlayingScore > ps.api > src > TS routes.ts > ...
1 import { UserController } from "./controller/UserController";
2 import { GameController } from "./controller/GameController";
3
4 export const Routes = [
5   { method: "get", route: "/users", controller: UserController, action: "all" },
6   { method: "get", route: "/users/:id", controller: UserController, action: "one" },
7   { method: "post", route: "/users", controller: UserController, action: "save" },
8   { method: "delete", route: "/users/:id", controller: UserController, action: "remove" },
9
10  { method: "get", route: "/games", controller: GameController, action: "all" },
11  { method: "get", route: "/games/:id", controller: GameController, action: "one" },
12  { method: "post", route: "/games", controller: GameController, action: "save" },
13  { method: "delete", route: "/games/:id", controller: GameController, action: "remove" }
14];
```

Toda rota tem uma tipagem  
de método HTTP

# Criando Model Game

Vamos projetar nosso CRUD de jogos

- Agora, demos ajustar nossas rotas de games
- Para isso, vamos alterar o arquivo routes.ts

TS routes.ts X

```
PlayingScore > ps.api > src > TS routes.ts > ...
1 import { UserController } from "./controller/UserController";
2 import { GameController } from "./controller/GameController";
3
4 export const Routes = [
5   { method: "get", route: "/users", controller: UserController, action: "all" },
6   { method: "get", route: "/users/:id", controller: UserController, action: "one" },
7   { method: "post", route: "/users", controller: UserController, action: "save" },
8   { method: "delete", route: "/users/:id", controller: UserController, action: "remove" },
9
10  { method: "get", route: "/games", controller: GameController, action: "all" },
11  { method: "get", route: "/games/:id", controller: GameController, action: "one" },
12  { method: "post", route: "/games", controller: GameController, action: "save" },
13  { method: "delete", route: "/games/:id", controller: GameController, action: "remove" }
14];
```

Toda rota tem um PATH a ser executado no servidor

# Criando Model Game

Vamos projetar nosso CRUD de jogos

- Agora, demos ajustar nossas rotas de games
- Para isso, vamos alterar o arquivo routes.ts

TS routes.ts X

```
PlayingScore > ps.api > src > TS routes.ts > ...
1 import { UserController } from "./controller/UserController";
2 import { GameController } from "./controller/GameController";
3
4 export const Routes = [
5   { method: "get", route: "/users", controller: UserController, action: "all" },
6   { method: "get", route: "/users/:id", controller: UserController, action: "one" },
7   { method: "post", route: "/users", controller: UserController, action: "save" },
8   { method: "delete", route: "/users/:id", controller: UserController, action: "remove" },
9
10  { method: "get", route: "/games", controller: GameController, action: "all" },
11  { method: "get", route: "/games/:id", controller: GameController, action: "one" },
12  { method: "post", route: "/games", controller: GameController, action: "save" },
13  { method: "delete", route: "/games/:id", controller: GameController, action: "remove" }
14];
```

Toda rota tem um controlador que será responsável pela manipulação dos dados

# Criando Model Game

Vamos projetar nosso CRUD de jogos

- Agora, demos ajustar nossas rotas de games
- Para isso, vamos alterar o arquivo routes.ts

TS routes.ts X

```
PlayingScore > ps.api > src > TS routes.ts > ...
1 import { UserController } from "./controller/UserController";
2 import { GameController } from "./controller/GameController";
3
4 export const Routes = [
5   { method: "get", route: "/users", controller: UserController, action: "all" },
6   { method: "get", route: "/users/:id", controller: UserController, action: "one" },
7   { method: "post", route: "/users", controller: UserController, action: "save" },
8   { method: "delete", route: "/users/:id", controller: UserController, action: "remove" },
9
10  { method: "get", route: "/games", controller: GameController, action: "all" },
11  { method: "get", route: "/games/:id", controller: GameController, action: "one" },
12  { method: "post", route: "/games", controller: GameController, action: "save" },
13  { method: "delete", route: "/games/:id", controller: GameController, action: "remove" }
14];
```

Toda rota tem uma ação que, geralmente, é um método do controlador

# Criando Model Game

---

Vamos projetar nosso CRUD de jogos

- Agora, demos ajustar nossas rotas de games
- Para isso, vamos alterar o arquivo routes.ts

Vamos testar nossa API de Games no Insomnia ou PostMan

# Criando Model Game

Vamos projetar nosso CRUD de jogos

- Agora, demos ajustar nossas rotas de games
- Para isso, vamos alterar o arquivo routes.ts

Vamos fazer um GET para localhost:3000/games

The screenshot shows a Postman request for a GET operation on the URL `localhost:3000/games`. The response is a 200 OK status with a response time of 16 ms and a body size of 2 B. The response body is an empty array: `[ ]`.

Method	URL	Status	Time	Size
GET	localhost:3000/games	200 OK	16 ms	2 B

Request Headers:

- Body
- Auth
- Query
- Header
- Docs

Response Headers:

- Preview
- Header

Response Body:

```
[ ]
```

# Criando Model Game

Vamos projetar nosso CRUD de jogos

- Agora, demos ajustar nossas rotas de games
- Para isso, vamos alterar o arquivo routes.ts

Vamos fazer um POST com dados para localhost:3000/games

The screenshot shows a Postman request for a POST operation to the endpoint `localhost:3000/games`. The response status is `200 OK`, the time taken is `352 ms`, and the response body is displayed in JSON format.

**Request Headers:**

- POST `localhost:3000/games`
- Send

**Response Headers:**

- `200 OK`
- `352 ms`
- `48`

**Request Body (JSON):**

```
1+ {  
2   "name": "Truco",  
3   "description": "DescriçãoTruco é um popular jogo de cartas praticado em  
diversos locais da América do Sul e algumas regiões da Espanha e Itália. É  
jogado com o baralho espanhol, por dois, quatro ou seis jogadores, divididos em  
dois lados opostos",  
4   "teams_limit": "2",  
5   "players_per_team_limit": "2"  
6 }
```

**Response Body (JSON):**

```
1+ {  
2   "name": "Truco",  
3   "description": "DescriçãoTruco é um popular jogo de cartas praticado em  
diversos locais da América do Sul e algumas regiões da Espanha e Itália. É  
jogado com o baralho espanhol, por dois, quatro ou seis jogadores, divididos em  
dois lados opostos",  
4   "teams_limit": "2",  
5   "players_per_team_limit": "2"  
6   "createdAt": "2020-07-10T14:00:00.000Z",  
7   "updatedAt": "2020-07-10T14:00:00.000Z"
```

# Criando Model Game

Vamos projetar nosso CRUD de jogos

- Agora, demos ajustar nossas rotas de games
- Para isso, vamos alterar o arquivo routes.ts

Vamos fazer um GET, novamente, para localhost:3000/games

The screenshot shows a REST API testing interface. At the top, there's a header bar with the text "Vamos fazer um GET, novamente, para localhost:3000/games". Below this, the main interface has several tabs: "Body", "Auth", "Query", "Header", "Preview", "Cookie", and "Timeline". The "Preview" tab is currently selected and displays the response body of a recent request. The request details are as follows:

- Method: GET
- URL: localhost:3000/games
- Status: 200 OK
- Time: 5.81 ms
- Size: 486 B

The "Preview" tab shows the JSON response body:

```
1: {
2:   {
3:     "uid": "361170c5-a13d-4eef-85bd-6d1e91981e41",
4:     "isPublic": true,
5:     "isActive": true,
6:     "createdAt": "2020-07-02T20:00:48.478Z",
7:     "updatedAt": "2020-07-02T20:00:48.478Z",
8:     "name": "Truco",
9:     "description": "O Truco é um jogo de cartas muito popular no Brasil. É jogado com um baralho de 52 cartas e pode ser jogado por duas ou mais pessoas.",
10:    "rules": [
11:      {
12:        "rule": "O objetivo do Truco é ganhar mais cartas que o adversário em cada mão."
13:      }
14:    ],
15:    "deck": [
16:      {
17:        "card": "A de Copas"
18:      }
19:    ]
20:  }
21:}
```

# Criando Model Game

Vamos projetar nosso CRUD de jogos

- Agora, demos ajustar nossas rotas de games
- Para isso, vamos alterar o arquivo routes.ts

Vamos fazer um DELETE para localhost:3000/games/"<uid>"

DELETE `localhost:3000/games/361170c5-a13d-4eef-85bd-6d1e91981e41` Send **200 OK** 52.6 ms 485 B

Body **Auth** **Query** Header Docs Preview **Header** **Cookie** Timeline

```
1 - {
2   "uid": "361170c5-a13d-4eef-85bd-6d1e91981e41",
3   "isPublic": true,
4   "isActive": false,
5   "createdAt": "2020-07-02T20:00:48.478Z",
6   "updatedAt": "2020-07-02T20:11:02.000Z",
7   "name": "Truco",
8   "description": "DescriçãoTruco é um popular jogo de..."
```

# Criando Model Game

Vamos projetar nosso CRUD de jogos

- Agora, demos ajustar nossas rotas de games
- Para isso, vamos alterar o arquivo routes.ts

Vamos fazer um DELETE para localhost:3000/games/"<uid>"

The screenshot shows a Postman request for a DELETE operation on the URL `localhost:3000/games/361170c5-a13d-4eef-85bd-6d1e91981e41`. The response is a 200 OK status with a duration of 52.6 ms and a body size of 485 B. The response body is a JSON object with the following content:

```
1: {
2:   "uid": "361170c5-a13d-4eef-85bd-6d1e91981e41",
3:   "isActive": true,
4:   "isActive": false, // Line 4 is highlighted with a blue box
5:   "createdAt": "2020-07-02T20:00:48.478Z",
6:   "updatedAt": "2020-07-02T20:11:02.000Z",
7:   "name": "Truco",
8:   "description": "DescriçãoTruco é um popular jogo de"}
```

A callout box with a blue background and white text points to the fourth line of the JSON response, highlighting the value `false` under the `isActive` key.

# Criando Model Game

Vamos projetar nosso CRUD de jogos

- Agora, demos ajustar nossas rotas de games
- Para isso, vamos alterar o arquivo routes.ts

```
TS BaseController.ts ×  
PlayingScore > ps.api > src > controller > TS BaseController.ts  
--  
13  
14     async all() {  
15         return this._repository.find({  
16             where: {  
17                 isActive: true  
18             }  
19         });  
20     }  
}
```

Vamos alterar o método ALL do BaseController.ts adicionando o where de isActive = true

# Criando Model Game

Vamos projetar nosso CRUD de jogos

- Agora, demos ajustar nossas rotas de games
- Para isso, vamos alterar o arquivo routes.ts

Vamos fazer um GET, novamente, para localhost:3000/games

The screenshot shows a Postman API request for a GET operation on the URL `localhost:3000/games`. The request was sent and received a **200 OK** response with a total time of **12.8 ms** and a body size of **2 B**. The response body is displayed as a JSON array with one element: `[{}]`.

Method	URL	Status	Time	Size
GET	localhost:3000/games	200 OK	12.8 ms	2 B

Request Headers:

- Body
- Auth
- Query
- Header
- Docs

Response Headers:

- Preview
- Header 6
- Cookie

Response Body:

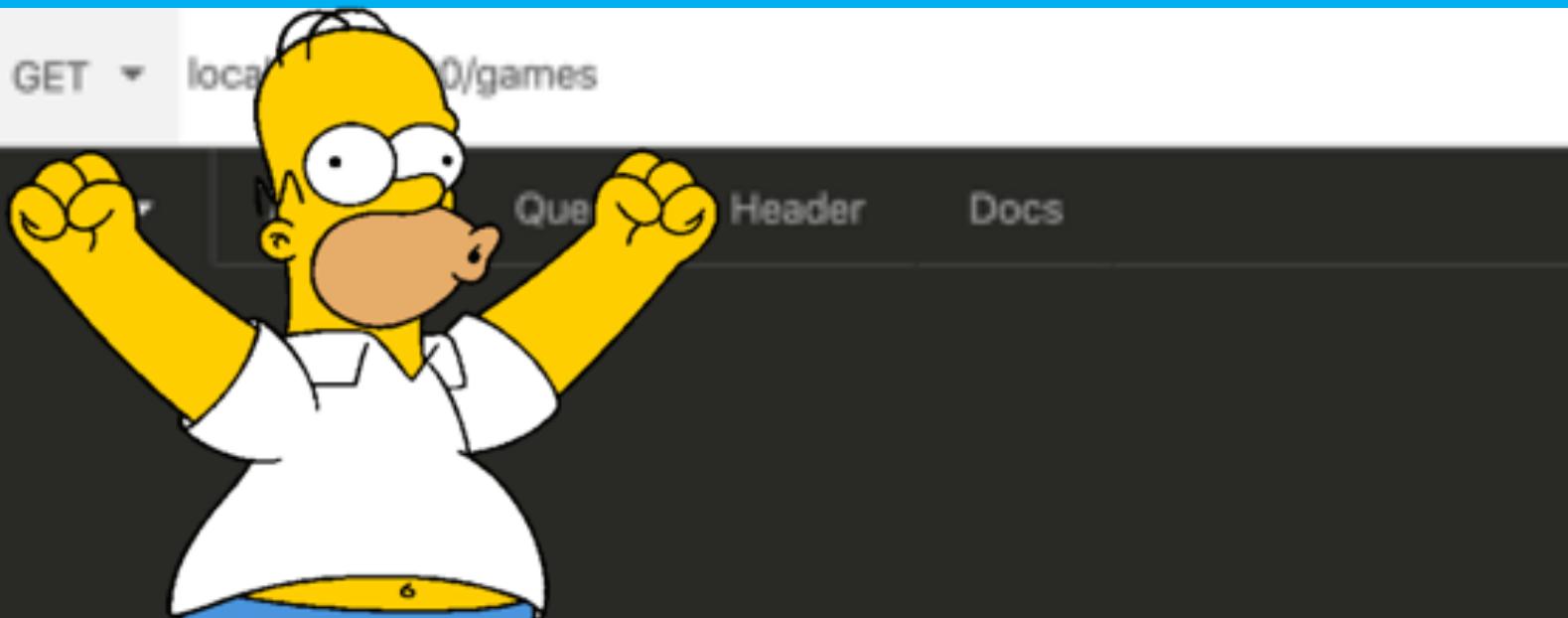
```
1 [{}]
```

# Criando Model Game

Vamos projetar nosso CRUD de jogos

- Agora, demos ajustar nossas rotas de games
- Para isso, vamos alterar o arquivo routes.ts

Vamos fazer um GET, novamente, para localhost:3000/games



# Criando Model Game

Antes de seguirmos para o App, precisamos instalar o pacote de CORS (controle de acesso)

<https://www.youtube.com/watch?v=GZV-FUdeVwE>



**CORS (Cross-Origin Resource Sharing em 6 minutos) // Dicionário do Programador**

Código Fonte TV 6:39 11 mil visualizações • há 4 meses

👉 **QUER AJUDAR O CANAL?** → <https://codft.me/clubecdfs> Quem trabalha com Front-End e Back-End precisam conhecer melhor como os ...

# Criando Model Game

Antes de seguirmos para o App, precisamos instalar o pacote de CORS (controle de acesso)

- Para isso vamos instalar, dentro da pasta de nossa API este pacote
- → npm install cors

```
[→ ps.api git:(master) ✘ npm install cors
npm WARN ps.api@0.0.1 No repository field.
npm WARN ps.api@0.0.1 No license field.

+ cors@2.8.5
added 1 package from 1 contributor and audited 180 packages in 2.234s

3 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

# Criando Model Game

Antes de seguirmos para o App, precisamos instalar

- Agora, no index.ts vamos adicionar o uso dele

```
TS index.ts ×  
PlayingScore > ps.api > src > TS  
13  
14 // Cors  
15 app.use(cors());  
16
```

```
TS index.ts ×  
PlayingScore > ps.api > src > TS index.ts > ...  
1 import "reflect-metadata";  
2 import {createConnection} from "typeorm";  
3 import * as express from "express";  
4 import * as cors from 'cors';  
5 import * as bodyParser from "body-parser";  
6 import {Request, Response} from "express";  
7 import {Routes} from "./routes";  
8 import config from "./include/config";  
9  
10 // create express app  
11 const app = express();  
12 app.use(bodyParser.json());  
13  
14 // Cors  
15 app.use(cors());  
16
```

Listando Games da API no APP

# Listando Games da API no APP

---

Com a API funcional, recebendo e retornando dados, vamos consumir e apresentar estes dados na nossa aplicação

# Listando Games da API no APP

Com a API funcional, recebendo e retornando dados, vamos consumir e apresentar estes dados na nossa aplicação

- Primeiro, vamos criar um serviço para consumir a API

Então, dentro do projeto Ionic, vamos executar

- → ionic g

```
[→ PlayingScore git:(master) ✘ ionic g
? What would you like to generate?
  enum
  page
  component
> service
  module
  class
  directive
```

# Listando Games da API no APP

Com a API funcional, recebendo e retornando dados, vamos consumir e apresentar estes dados na nossa aplicação

- Primeiro, vamos criar um serviço para consumir a API

Então, dentro do projeto Ionic, vamos executar

- → ionic g

```
[→ PlayingScore git:(master) ✘ ionic g
? What would you like to generate?
  enum
  page
  component
> service
  module
  class
  directive
```

Escolher SERVICE e chamar de API

# Listando Games da API no APP

---

Com a API funcional, recebendo e retornando dados, vamos consumir e apresentar estes dados na nossa aplicação

- Agora, vamos criar o model de games

# Listando Games da API no APP

Com a API funcional, recebendo e retornando dados, vamos consumir e apresentar estes dados na nossa aplicação

- Agora, vamos criar o model de games

Então, dentro do projeto Ionic, vamos executar

- → ionic g

```
[→ PlayingScore git:(master) ✘ ionic g
? What would you like to generate?
  component
  service
  module
> class
  directive
  guard
  pipe
```

Escolher CLASS e chamar de models/game

# Listando Games da API no APP

```
✓ src
  ✓ app
    > jogos
    > jogos-amigos
    > jogos-favoritos
    > marcador
    ✓ models
      TS game.ts
      TS game.spec.ts
    > partidas
    > perfil
    > tabs
    TS api.service.ts
    TS api.service.spec.ts
    TS app-routing.module.ts
    <> app.component.html
    ❀ app.component.scss
    TS app.component.ts
    TS app.component.spec.ts
    TS app.module.ts
```

Criamos o model para games

Criamos arquivo para controle da API

# Listando Games da API no APP

Vamos começar programando o serviço

```
✓ app
  > jogos
  > jogos-amigos
  > jogos-favoritos
  > marcador
  > models
  > partidas
  > perfil
  > tabs
TS api.service.ts
TS api.service.spec.ts
```

# Listando Games da API no APP

```
ts api.service.ts X

PlayingScore > src > app > ts api.service.ts > ...
1  import { Injectable } from '@angular/core';
2  import { HttpClient, HttpHeaders, HttpErrorResponse } from '@angular/common/http';
3  import { Game } from './models/game';
4  import { Observable, throwError } from 'rxjs';
5  import { retry, catchError } from 'rxjs/operators';
6
7  @Injectable({
8    providedIn: 'root'
9  })
10 export class ApiService {
11
12   // API path
13   base_path = 'http://localhost:3000/games';
14
15   constructor(private http: HttpClient) { }
16
17   // Http Options
18   httpOptions = {
19     headers: new HttpHeaders({
20       'Content-Type': 'application/json'
21     })
22   }
23
24   // Handle API errors
25   handleError(error: HttpErrorResponse) {
26     if (error.error instanceof ErrorEvent) {
27       // A client-side or network error occurred. Handle it accordingly.
28       console.error('An error occurred:', error.error.message);
29     } else {
30       // The backend returned an unsuccessful response code.
31       // The response body may contain clues as to what went wrong,
32       console.error(
33         `Backend returned code ${error.status}, ` +
34         `body was: ${error.error}`);
35     }
36     // return an observable with a user-facing error message
37     return throwError('Something bad happened; please try again later.');
38   };
39
40   // Get Game data
41   getList(): Observable<Game> {
42     return this.http
43       .get<Game>(this.base_path)
44       .pipe(
45         retry(2),
46         catchError(this.handleError)
47       )
48     }
49 }
```

# Listando Games da API no APP

```
ts api.service.ts x
PlayingScore > src > app > ts api.service.ts > ...
1 import { Injectable } from '@angular/core';
2 import { HttpClient, HttpHeaders, HttpErrorResponse } from '@angular/common/http';
3 import { Game } from './models/game';
4 import { Observable, throwError } from 'rxjs';
5 import { retry, catchError } from 'rxjs/operators';
6
7 @Injectable({
8   providedIn: 'root'
9 })
10 export class ApiService {
11
12   // API path
13   base_path = 'http://localhost:3000/games';
14
15   constructor(private http: HttpClient) { }
16
17   // ...
18
19
20
21
22
23
24
25
26
27
28
29
30
31   // The response body may contain clues as to what went wrong,
32   console.error(
33     `Backend returned code ${error.status}, ` +
34     `body was: ${error.error}`);
35   }
36   // return an observable with a user-facing error message
37   return throwError('Something bad happened; please try again later.');
38 };
39
40   // Get Game data
41   getList(): Observable<Game> {
42     return this.http
43       .get<Game>(this.base_path)
44       .pipe(
45         retry(2),
46         catchError(this.handleError)
47       )
48   }
49 }
```

Configuramos o path para acessar games da API e o construtor com acesso HTTP

# Listando Games da API no APP

ts api.service.ts x

```
PlayingScore > src > app > ts api.service.ts > ...
1 import { Injectable } from '@angular/core';
2 import { HttpClient, HttpHeaders, HttpErrorResponse } from '@angular/common/http';
3 import { Game } from './models/game';
4 import { Observable, throwError } from 'rxjs';
5 import { retry, catchError } from 'rxjs/operators';
6
7 @Injectable({
8   providedIn: 'root'
9 })
10 export class ApiService {
11
12   // API path
13   base_path = 'http://localhost:3000/games';
14
15   constructor(private http: HttpClient) { }
16
17   // Http Options
18   httpOptions = {
19     headers: new HttpHeaders({
20       'Content-Type': 'application/json'
21     })
22   }
23
24   /**
25    * Get Game data
26    */
27   getList(): Observable<Game> {
28     return this.http
29       .get<Game>(this.base_path)
30       .pipe(
31         retry(2),
32         catchError(this.handleError)
33       )
34   }
35
36   // return an observable with a user-facing error message
37   return throwError('Something bad happened; please try again later.');
38 };
39
40
41   // Get Game data
42   getList(): Observable<Game> {
43     return this.http
44       .get<Game>(this.base_path)
45       .pipe(
46         retry(2),
47         catchError(this.handleError)
48       )
49 }
```

Criamos uma var para opções HTTP, retornando JSON

# Listando Games da API no APP

Criamos um gerenciador de erros, para caso venham ocorrer

```
ts api.service.ts x
PlayingScore > src > app > ts api.service.ts > ...
1 import { Injectable } from '@angular/core';
2 import { HttpClient, HttpHeaders, HttpErrorResponse } from '@angular/common/http';
3 import { Game } from './models/game';
4 import { Observable, throwError } from 'rxjs';
5 import { retry, catchError } from 'rxjs/operators';
6
7 @Injectable({
8   providedIn: 'root'
9 })
10
11 // Handle API errors
12
13 handleError(error: HttpErrorResponse) {
14   if (error.error instanceof ErrorEvent) {
15     // A client-side or network error occurred. Handle it accordingly.
16     console.error('An error occurred:', error.error.message);
17   } else {
18     // The backend returned an unsuccessful response code.
19     // The response body may contain clues as to what went wrong,
20     console.error(
21       `Backend returned code ${error.status}, ` +
22       `body was: ${error.error}`);
23   }
24   // return an observable with a user-facing error message
25   return throwError('Something bad happened; please try again later.');
26 }

27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46   catchError(this.handleError)
47 }
48 }
49 }
```

# Listando Games da API no APP

Criamos método que retorna uma lista de games, com base no model criado anteriormente

```
ts api.service.ts x
PlayingScore > src > app > ts api.service.ts > ...
1 import { Injectable } from '@angular/core';
2 import { HttpClient, HttpHeaders, HttpErrorResponse } from '@angular/common/http';
3 import { Game } from './models/game';
4 import { Observable, throwError } from 'rxjs';
5 import { retry, catchError } from 'rxjs/operators';
6
7 @Injectable({
8   providedIn: 'root'
9 })
10 export class ApiService {
11
12   // API path
13   base_path = 'http://localhost:4000/games';
14
15   constructor(private http: HttpClient) { }
16
17   // Get Game data
18   getList(): Observable<Game> {
19     return this.http
20       .get<Game>(this.base_path)
21       .pipe(
22         retry(2),
23         catchError(this.handleError)
24       )
25     }
26
27   handleError(error: HttpErrorResponse) {
28     let errorMessage = '';
29     if (error.error instanceof ErrorEvent) {
30       errorMessage = `Error: ${error.error.message}`;
31     } else {
32       errorMessage = `Error status code: ${error.status}, error message: ${error.message}`;
33     }
34     return throwError(errorMessage);
35   }
36
37 }
38
39 }
40
41 }
42
43 }
44
45 }
46
47 }
48
49 }
```

# Listando Games da API no APP

Agora, vamos modificar o model Game

The screenshot shows a file tree structure:

- src/
  - app/
    - > jogos
    - > jogos-amigos
    - > jogos-favoritos
    - > marcador
  - models/
    - | TS game.ts
    - | TS game.spec.ts

# Listando Games da API no APP

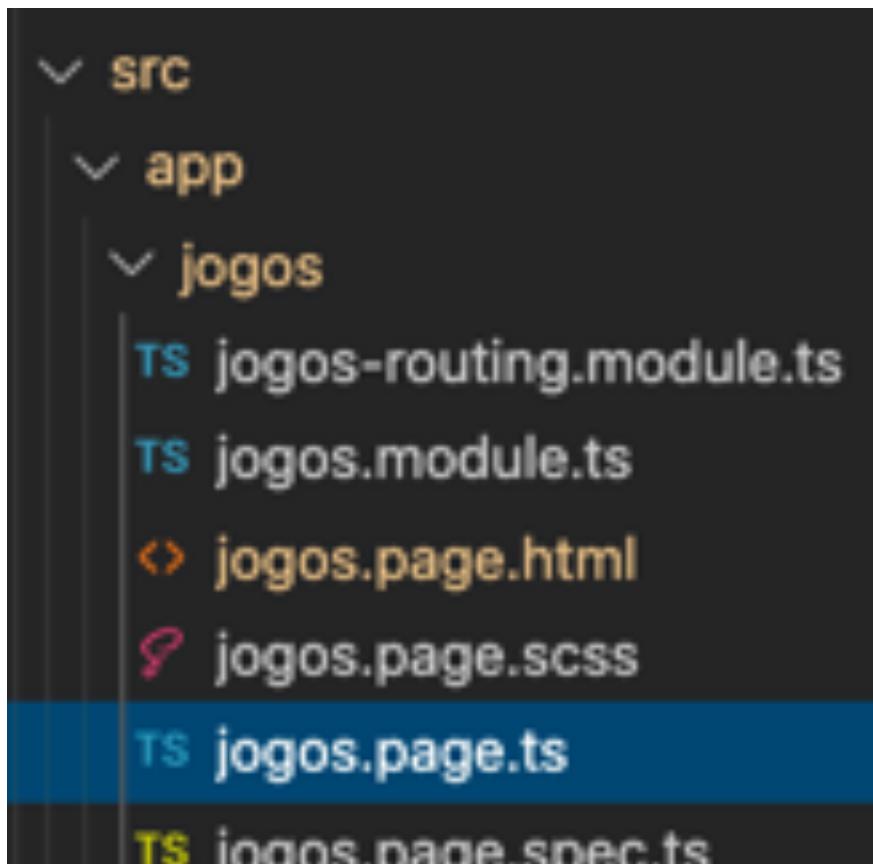
TS game.ts ×

PlayingScore > src > app > models > TS game.ts > ...

```
1  export class Game {  
2  
3      name: string;  
4      description: string;  
5      link_rules: string;  
6      teams_limit: number;  
7      players_per_team_limit: number;  
8  
9  }
```

# Listando Games da API no APP

Agora, vamos alterar o jogos.page.ts, para fazer uso deste serviço



```
src
  app
    jogos
      jogos-routing.module.ts
      jogos.module.ts
      jogos.page.html
      jogos.page.scss
      jogos.page.ts
      jogos.page.spec.ts
```

# Listando Games da API no APP

ts jogos.page.ts ×

```
PlayingScore > src > app > jogos > ts jogos.page.ts > ...
1 import { Component, OnInit } from '@angular/core';
2 import { ApiService } from '../api.service';
3
4 @Component({
5   selector: 'app-jogos',
6   templateUrl: './jogos.page.html',
7   styleUrls: ['./jogos.page.scss'],
8 })
9 export class JogosPage implements OnInit {
10
11   gamesData: any;
12
13   constructor(public apiService: ApiService) {
14     this.gamesData = [];
15   }
16
17   ngOnInit() { }
18
19   ionViewWillEnter() {
20     // Used ionViewWillEnter as ngOnInit is not
21     // called due to view persistence in Ionic
22     this.getAllGames();
23   }
24
25   getAllGames() {
26     //Get saved list of students
27     this.apiService.getList().subscribe(response => {
28       console.log(response);
29       this.gamesData = response;
30     })
31   }
32 }
```

# Listando Games da API no APP

ts jogos.page.ts ×

```
PlayingScore > src > app > jogos > ts jogos.page.ts > ...
1 import { Component, OnInit } from '@angular/core';
2 import { ApiService } from '../api.service';
3
4 @Component({
5   selector: 'app-jogos',
6   templateUrl: './jogos.page.html',
7   styleUrls: ['./jogos.page.scss'],
8 })
9 export class JogosPage implements OnInit {
10
11   gamesData: any;
12
13   constructor(public apiService: ApiService) {
14     this.gamesData = [];
15   }
16
17   ionViewWillEnter() {
18     // called due to view persistence in Ionic
19     this.getAllGames();
20   }
21
22   getAllGames() {
23     this.apiService.getList().subscribe(response => {
24       console.log(response);
25       this.gamesData = response;
26     });
27   }
28
29 }
30
31 }
32 }
```

Criamos um atributo gamesData para ser acessado no HTML, via Angular

Inicializamos o atributo sem conteúdo

# Listando Games da API no APP

ts jogos.page.ts ×

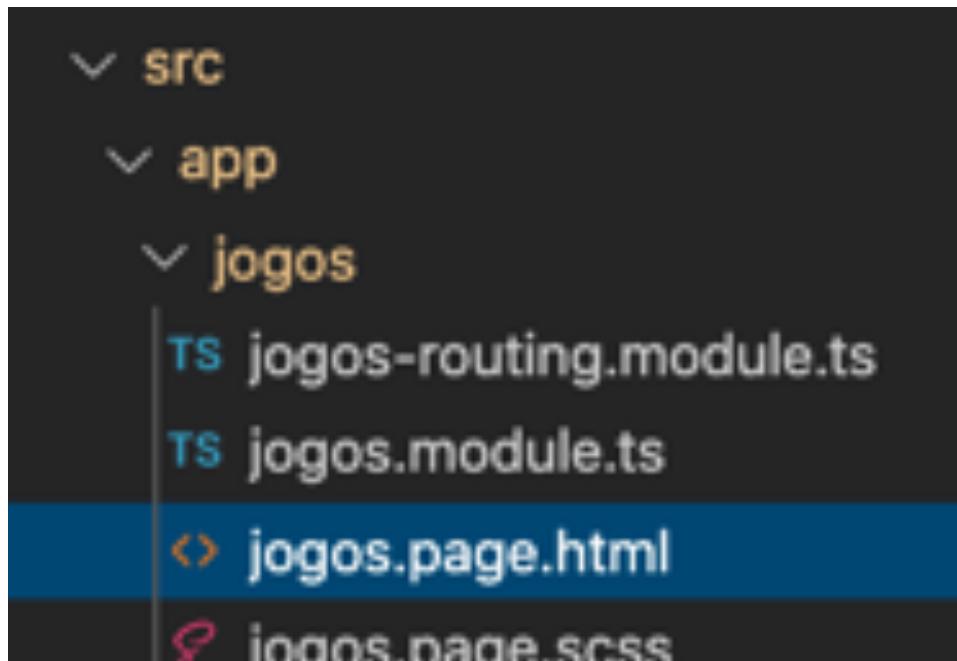
```
PlayingScore > src > app > jogos > ts jogos.page.ts > ...
1 import { Component, OnInit } from '@angular/core';
2 import { ApiService } from '../api.service';
3
4 @Component({
5   selector: 'app-jogos',
6   templateUrl: './jogos.page.html',
7   styleUrls: ['./jogos.page.scss'],
8
9   17   ngOnInit() { }
10
11  18
12  19   ionViewWillEnter() {
13
14    20     // Used ionViewWillEnter as ngOnInit is not
15
16    21     // called due to view persistence in Ionic
17
18    22       this.getAllGames();
19
20
21  23   }
22
23
24
25  25   getAllGames() {
26
27     //Get saved list of students
28     this.apiService.getList().subscribe(response => {
29
30       29         console.log(response);
31
32       30         this.gamesData = response;
31
32   })
```

Criamos um método para buscar todos games ativos, getAllGames()

Executamos este método no ionViewWillEnter

# Listando Games da API no APP

Por fim, temos que alterar o html de jogos-page



# Listando Games da API no APP

```
① jogos.page.html ×
PlayingScore > src > app > jogos > ① jogos.page.html > ⚡ ion-header > ⚡ ion-toolbar.tab-bar-sub > ⚡ ion-t
23
24  <ion-content>
25  |   <ion-card *ngFor="let game of gamesData">
26  |     <ion-item>
27  |       <ion-avatar slot="start">
28  |         
29  |       </ion-avatar>
30
31       <ion-label class="ion-text-wrap">
32         <ion-text color="dark"><h3>Usuário Logado</h3></ion-text>
33         <p>{{ game.name }}</p>
34       </ion-label>
35     </ion-item>
36
37     <ion-card-content>
38       {{ game.description }}
39     </ion-card-content>
40
41     <ion-card-content>
42       <ion-button size="small" fill="outline" slot="end">Ver Regras</ion-button>
43     </ion-card-content>
44   </ion-card>
45
46
47   <ion-fab vertical="bottom" horizontal="start" slot="fixed">
48     <ion-fab-button size="large"><ion-icon name="add"></ion-icon></ion-fab-button>
49   </ion-fab>
50 </ion-content>
```

# Listando Games da API no APP

```
① jogos.page.html ×
PlayingScore > src > app > jogos > ② jogos.page.html > ③ ion-header > ④ ion-toolbar.tab-bar-sub > ⑤ ion-ta
23
24  <ion-content>
25 |   <ion-card *ngFor="let game of gamesData">
26 |     <ion-item>
27 |       <ion-avatar slot="start">
28 |         
29 |       </ion-avatar>
30 |
31       <ion-label class="ion-text-wrap">
32         <ion-text color="dark"><h3>Usuário Logado</h3></ion-text>
33         <p>{{ game.name }}</p>
34       </ion-label>
35     </ion-item>
36
37     <ion-card-content>
38       {{ game.description }}
39     </ion-card-content>
40
41     <ion-card-content>
42       <ion-button size="small" fill="outline" slot="end">Ver Regras</ion-button>
43     </ion-card-content>
44   </ion-card>
45
46
47   <ion-fab vertical="bottom" horizontal="start" slot="fixed">
48     <ion-fab-button size="large"><ion-icon name="add"></ion-i>
49   </ion-fab>
50 </ion-content>
```

Vamos replicar o ion-card conforme quantidade de registros retornados

# Listando Games da API no APP

```
① jogos.page.html ×  
PlayingScore > src > app > jogos > ② jogos.page.html > ③ ion-header > ④ ion-toolbar.tab-bar-sub > ⑤ ion-t  
23  
24  <ion-content>  
25  |   <ion-card *ngFor="let game of gamesData">  
26  |     <ion-item>  
27  |       <ion-avatar slot="start">  
28  |           
29  |       </ion-avatar>  
30  
31       <ion-label class="ion-text-wrap">  
32         <ion-text color="dark"><h3>Usuário Logado</h3></ion-text>  
33  |         <p>{{ game.name }}</p>  
34       </ion-label>  
35     </ion-item>  
36  
37     <ion-card-content>  
38  |       {{ game.description }}  
39     </ion-card-content>  
40  
41     <ion-card-content>  
42       <ion-button size="small" fill="outline" slot="end">Ver Regras</ion-button>  
43     </ion-card-content>  
44   </ion-card>  
45  
46  
47   <ion-fab vertical="bottom" horizontal="start" slot="fixed">  
48     <ion-fab-button size="large"><ion-icon name="add"></ion-  
49   </ion-fab>  
50 </ion-content>
```

Vamos apresentar os dados no HTML {{ ... }} com base nos dados retornados pelo Angular

# Listando Games da API no APP

---

Por fim, vamos registrar o uso do módulo do HttpClient nos módulos do App

# Listando Games da API no APP

Por fim, vamos registrar o uso do módulo do HttpClient nos módulos do

```
<div>
  <ul>
    <li>> jogos</li>
    <li>> jogos-amigos</li>
    <li>> jogos-favoritos</li>
    <li>> marcador</li>
    <li>> models</li>
    <li>> partidas</li>
    <li>> perfil</li>
    <li>> tabs</li>
    <li>TS api.service.ts</li>
    <li>TS api.service.spec.ts</li>
    <li>TS app-routing.module.ts</li>
    <li><span></span> app.component.html</li>
    <li><span>S</span> app.component.scss</li>
    <li>TS app.component.ts</li>
    <li>TS app.component.spec.ts</li>
    <li>TS app.module.ts</li>
  </ul>
</div>
```

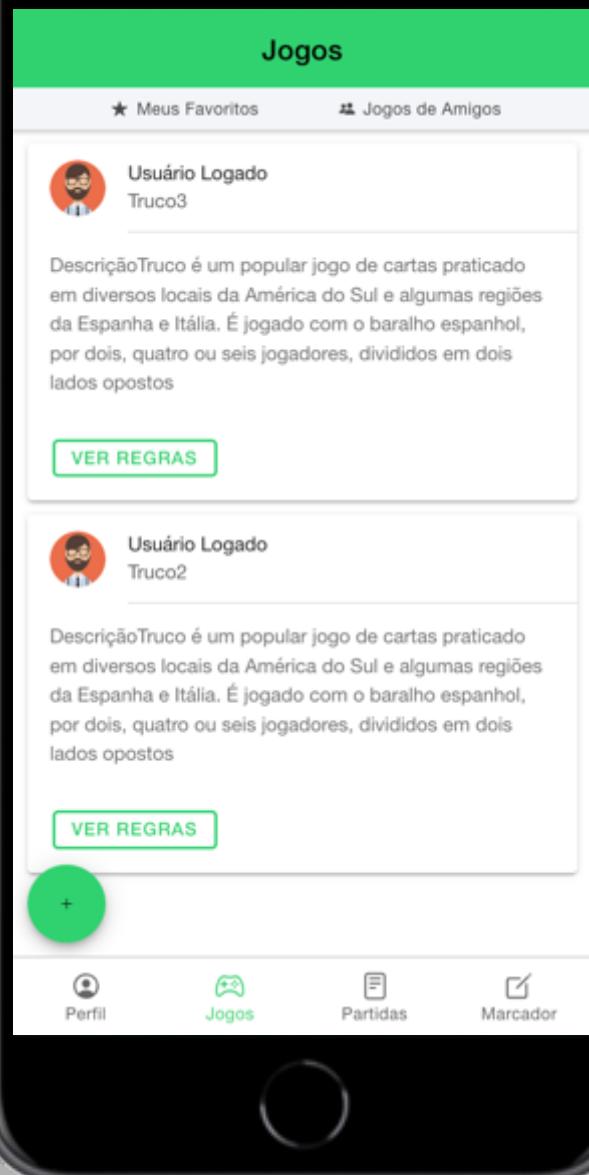
# Listando Games da API no APP

Por fim, vamos registrar o uso do módulo do HttpClient nos módulos do App

TS app.module.ts ×

PlayingScore > src > app > TS app.module.ts > AppModule

```
12 import { HttpClientModule } from '@angular/common/http';
13
14 @NgModule({
15   declarations: [AppComponent],
16   entryComponents: [],
17   imports: [BrowserModule, IonicModule.forRoot(), AppRoutingModule, HttpClientModule],
18   providers: [
19     StatusBar,
```



GET → localhost:3000/games Send 200 OK 9.02 ms 973 B

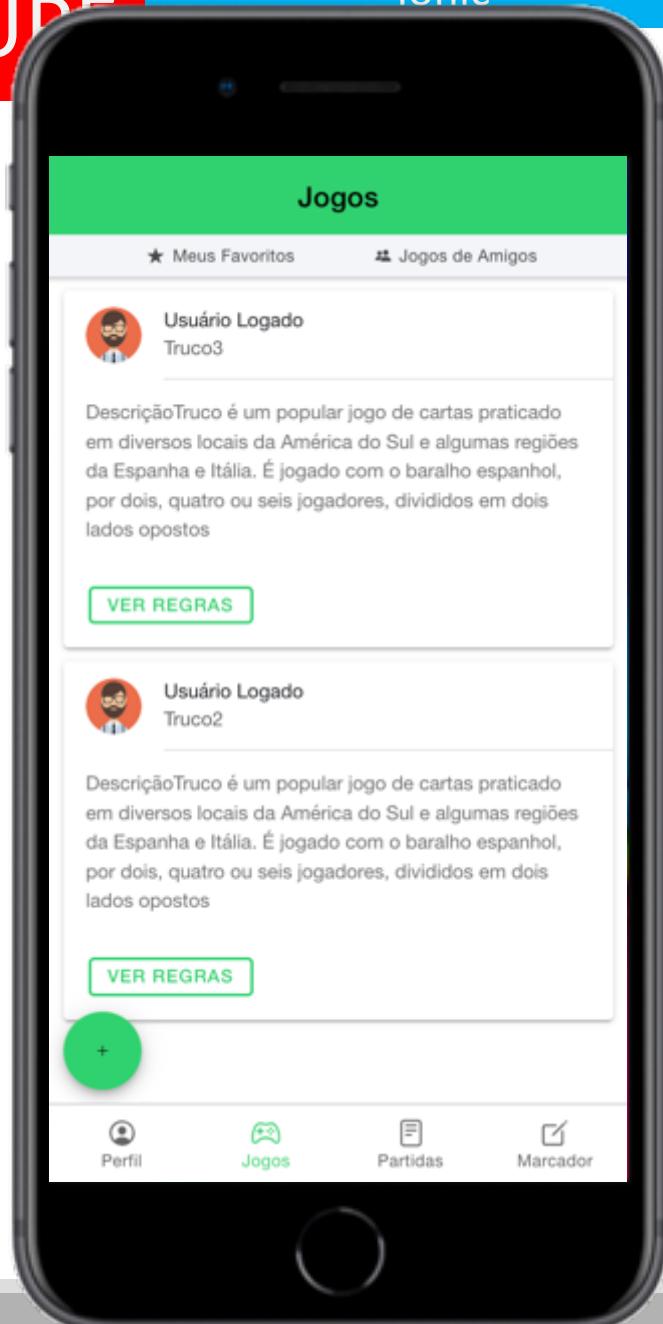
Body Auth Query Head Preview Header 7 Cookie Timeline

```

1 [ 
2   {
3     "uid": "82d8510d-2848-4957-a323-626230295e32",
4     "isPublic": true,
5     "isActive": true,
6     "createdAt": "2020-07-03T03:34:39.368Z",
7     "updatedAt": "2020-07-03T03:34:39.368Z",
8     "name": "Truco3",
9     "description": "DescriçãoTruco é um popular jogo de cartas praticado em diversos locais da América do Sul e algumas regiões da Espanha e Itália. É jogado com o baralho espanhol, por dois, quatro ou seis jogadores, divididos em dois lados opostos"
10    "link_rules": null,
11    "teams_limit": 2,
12    "players_per_team_limit": 2
13  },
14  {
15    "uid": "99f12e40-9c92-4c0d-9a81-a65f97cbb98a",
16    "isPublic": true,
17    "isActive": true,
18    "createdAt": "2020-07-03T03:22:03.760Z",
19    "updatedAt": "2020-07-03T03:22:03.760Z",
20    "name": "Truco2",
21    "description": "DescriçãoTruco é um popular jogo de cartas praticado em diversos locais da América do Sul e algumas regiões da Espanha e Itália. É jogado com o baralho espanhol, por dois, quatro ou seis jogadores, divididos em dois lados opostos"
22    "link_rules": null,
23    "teams_limit": 2,
24    "players_per_team_limit": 2
25  }
26 ]

```

Select a body type from above.



# Games da API no APP



# Criando Componente

# Criando Componente

---

Para melhor trabalhamos com padronização de UI, é interessante criarmos componentes para usar nas páginas

- Desta forma, evitamos a replicação de SCSS e HTML

# Criando Componente

Para melhor trabalhamos com padronização de UI, é interessante criarmos componentes para usar nas páginas

- Desta forma, evitamos a replicação de SCSS e HTML

Então, agora vamos criar o componente utilizado nas listagens de jogos

- Dentro do projeto Ionic, vamos executar
- → ionic g

```
→ PlayingScore git:(master) ionic g
? What would you like to generate?
  enum
  page
> component
  service
  module
  class
  directive
```

Criaremos um “component”

# Criando Componente

Para melhor trabalhamos com padronização de UI, é interessante criarmos componentes para usar nas páginas

- Desta forma, evitamos a replicação de SCSS e HTML

Então, agora vamos criar o componente utilizado nas listagens de jogos

- Dentro do projeto Ionic, vamos executar
- → ionic g

Daremos o nome de "components/jogo-card"

```
→ PlayingScore git:(master) ionic g
? What would you like to generate? component
? Name/path of component: components/jogo-card
```

Desta forma o ionic irá gerar uma pasta components

# Criando Componente

Para melhor trabalhar com componentes para usar nas páginas

- Desta forma, evitamos...

Então, agora vamos...

- Dentro do projeto
- → ionic g...

```
src
  app
    components / jogo-card
      <> jogo-card.component.html
      ⚡ jogo-card.component.scss
      TS jogo-card.component.ts
      TS jogo-card.component.spec.ts
```

# Criando Componente

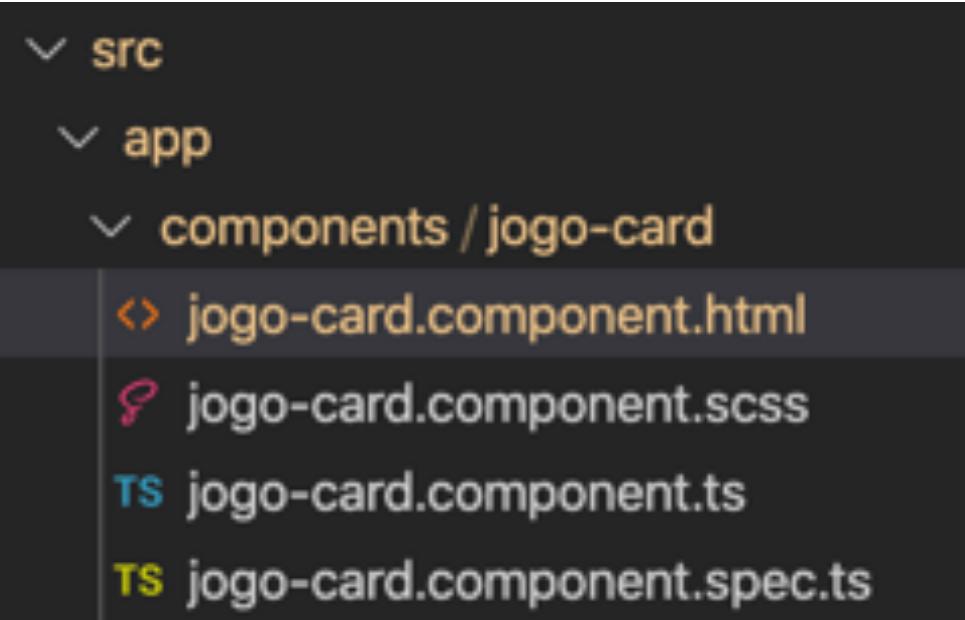
---

Agora, vamos iniciar a montagem da tela e programação do conteúdo

# Criando Componente

Agora, vamos iniciar a montagem da tela e programação do conteúdo

- Iniciaremos com o HTML do componente



```
src
  app
    components / jogo-card
      <> jogo-card.component.html
      ⚡ jogo-card.component.scss
      TS jogo-card.component.ts
      TS jogo-card.component.spec.ts
```

# Criando Componente

Agora, vamos iniciar a montagem da tela e programação do conteúdo

- Iniciaremos com o HTML do componente

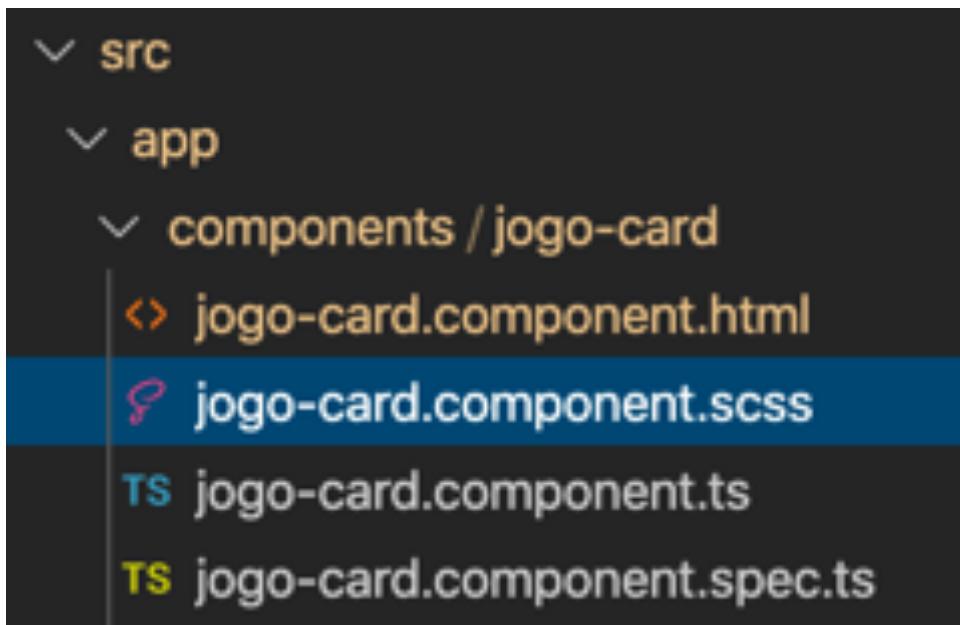
```
<> jogo-card.component.html <  
  
PlayingScore > src > app > components > jogo-card > <> jogo-card.component.html > ...  
1   <ion-card>  
2     <ion-item>  
3       <ion-avatar slot="start">  
4           
5       </ion-avatar>  
6  
7       <ion-label class="ion-text-wrap">  
8         <ion-text color="dark"><h3>Usuário Logado</h3></ion-text>  
9         <p>{{ name }}</p>  
10        </ion-label>  
11    </ion-item>  
12  
13    <ion-card-content>  
14      {{ description }}  
15    </ion-card-content>  
16  
17    <ion-card-content>  
18      <ion-button size="small" fill="outline" slot="end">Ver Regras</ion-button>  
19    </ion-card-content>  
20  </ion-card>
```

Basicamente, copiaremos o ION-CARD configurado na página de jogos.page.html

# Criando Componente

Agora, vamos iniciar a montagem da tela e programação do conteúdo

- O mesmo faremos para o SCSS



# Criando Componente e

Agora, vamos iniciar a montagem da tela e programação do conteúdo

- O mesmo faremos para o SCSS

jogo-card.component.scss ×

PlayingScore > src > app > components > jogo-card > jogo-card.component.scss

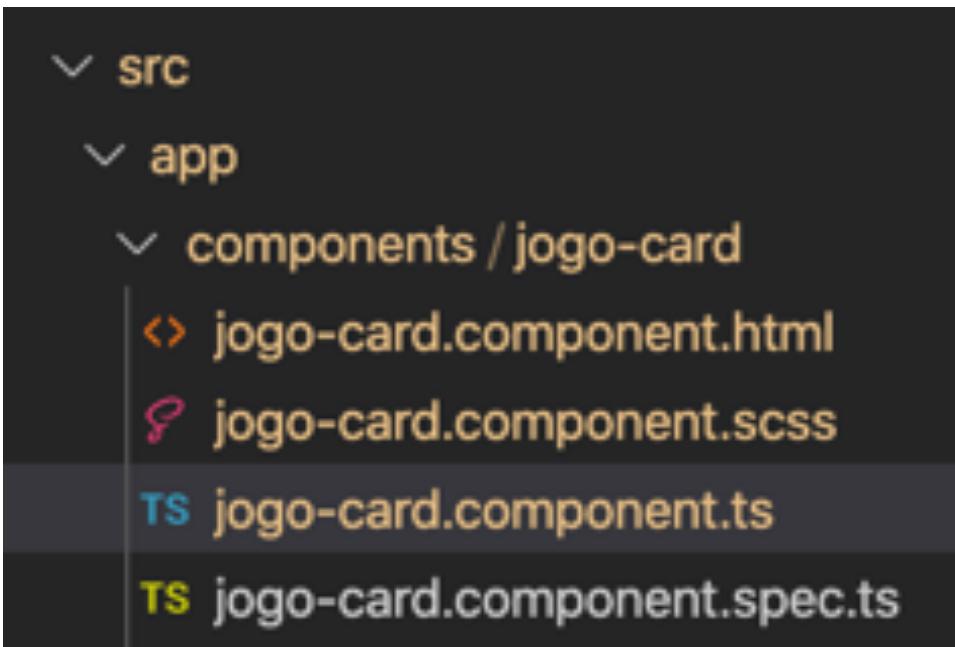
```
1  .tab-bar-sub {  
2      height: 30px;  
3      position: relative;  
4      background-color: #f4f5f8 !important;  
5      text-align: center !important;  
6  }  
7  
8  .subtitle {  
9      position: relative;  
10     top: -11px !important;  
11     font-size: 12px;  
12  }  
13  
14  ion-icon {  
15      font-size: 12px;  
16  }
```

Basicamente, copiaremos o conteúdo configurado no SCSS de jogos.page.scss

# Criando Componente

Agora, vamos iniciar a montagem da tela e programação do conteúdo

- Vamos configurar o componente, informando quais atributos ele conterá



```
src
  app
    components / jogo-card
      <> jogo-card.component.html
      📄 jogo-card.component.scss
      TS jogo-card.component.ts
      TS jogo-card.component.spec.ts
```

# Criando Componente

Agora, vamos iniciar a montagem da tela e programação do conteúdo

- Vamos configurar o componente, informando quais atributos ele conterá

TS jogo-card.component.ts ×

PlayingScore > src > app > components > jogo-card > TS jogo-card.component

```
1 import { Component, OnInit, Input } from '@angular/core';
2
3 @Component({
4   selector: 'app-jogo-card',
5   templateUrl: './jogo-card.component.html',
6   styleUrls: ['./jogo-card.component.scss'],
7 })
8 export class JogoCardComponent implements OnInit {
9   @Input() name: string;
10  @Input() description: string;
11
12  constructor() { }
13
14  ngOnInit() {}
15
16 }
```

# Criando Componente e

Agora, vamos iniciar a montagem da tela e programação do componente.

- Vamos configurar o componente, informando quais atributos ele receberá.

Adicionaremos dois atributos que serão do tipo Input (Angular)

jogo-card.component.ts ×

```
PlayingScore > src > app > components > jogo-card > TS jogo-card.component.ts
1   import { Component, OnInit, Input } from '@angular/core';
2
3   @Component({
4     selector: 'app-jogo-card',
5     templateUrl: './jogo-card.component.html',
6     styleUrls: ['./jogo-card.component.scss'],
7   })
8   export class JogoCardComponent implements OnInit {
9     @Input() name: string;
10    @Input() description: string;
11
12    constructor() { }
13
14    ngOnInit() {}
15
16 }
```

# Criando Componente e

Agora, vamos iniciar a montagem da tela e programação do componente.

- Vamos configurar o componente, informando quais atributos ele receberá.

Neste TS temos o atributo **SELECTOR**, que é a maneira configurada para chamarmos este componente em outras páginas.

TS jogo-card.component.ts ×

```
PlayingScore > src > app > components > jogo-card > TS jogo-card.component.ts
1 import { Component, OnInit, Input } from '@angular/core';
2
3 @Component({
4   selector: 'app-jogo-card',
5   templateUrl: './jogo-card.component.html',
6   styleUrls: ['./jogo-card.component.scss'],
7 })
8 export class JogoCardComponent implements OnInit {
9   @Input() name: string;
10  @Input() description: string;
11
12  constructor() { }
13
14  ngOnInit() {}
15
16 }
```

# Criando Componente

---

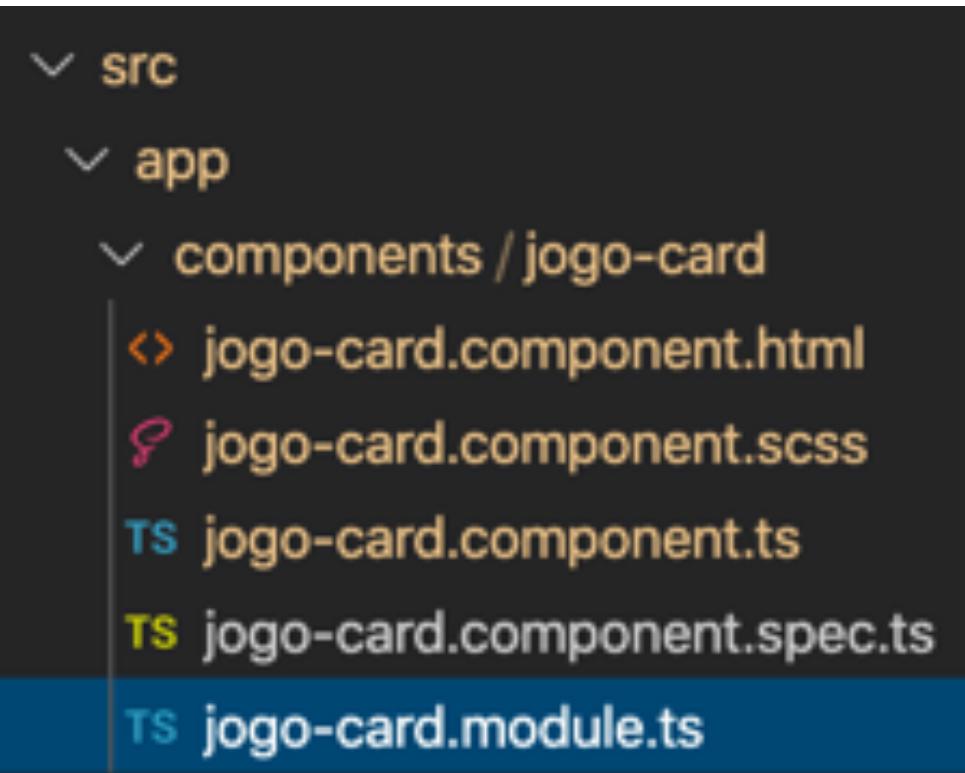
Agora, vamos iniciar a montagem da tela e programação do conteúdo

- Para finalizarmos a configuração do componente, vamos criar um arquivo de módulo

# Criando Componente

Agora, vamos iniciar a montagem da tela e programação do conteúdo

- Para finalizarmos a configuração do componente, vamos criar um arquivo de módulo



```
src
  app
    components / jogo-card
      jogo-card.component.html
      jogo-card.component.scss
      jogo-card.component.ts
      jogo-card.component.spec.ts
      jogo-card.module.ts
```

# Criando Componente e

Agora, vamos iniciar a montagem da tela e programação do componente.

- Para finalizarmos a configuração do componente, vamos criar um módulo para ele.

Criamos este módulo como se fosse o módulo de uma página qualquer

TS jogo-card.module.ts ×

```
PlayingScore > src > app > components > jogo-card > TS jogo-card.module.ts > .
```

```
1 import { NgModule } from '@angular/core';
2 import { CommonModule } from '@angular/common';
3 import { FormsModule } from '@angular/forms';
4
5 import { IonicModule } from '@ionic/angular';
6
7 import { JogoCardComponent } from './jogo-card.component';
8
9 @NgModule({
10   imports: [ CommonModule, FormsModule, IonicModule ],
11   declarations: [JogoCardComponent],
12   exports: [JogoCardComponent]
13 })
14 export class JogoCardComponentModule {}
```

# Criando Componente e

Agora, vamos iniciar a montagem da tela e programação do componente.

- Para finalizarmos a configuração do componente, vamos criar um módulo.

É a melhor forma de estrutura. Um módulo Angular que cria e carrega o componente.

TS jogo-card.module.ts X

```
PlayingScore > src > app > components > jogo-card > TS jogo-card.module.ts > .
```

```
1 import { NgModule } from '@angular/core';
2 import { CommonModule } from '@angular/common';
3 import { FormsModule } from '@angular/forms';
4
5 import { IonicModule } from '@ionic/angular';
6
7 import { JogoCardComponent } from './jogo-card.component';
8
9 @NgModule({
10   imports: [ CommonModule, FormsModule, IonicModule ],
11   declarations: [JogoCardComponent],
12   exports: [JogoCardComponent]
13 })
14 export class JogoCardComponentModule {}
```

# Criando Componente

---

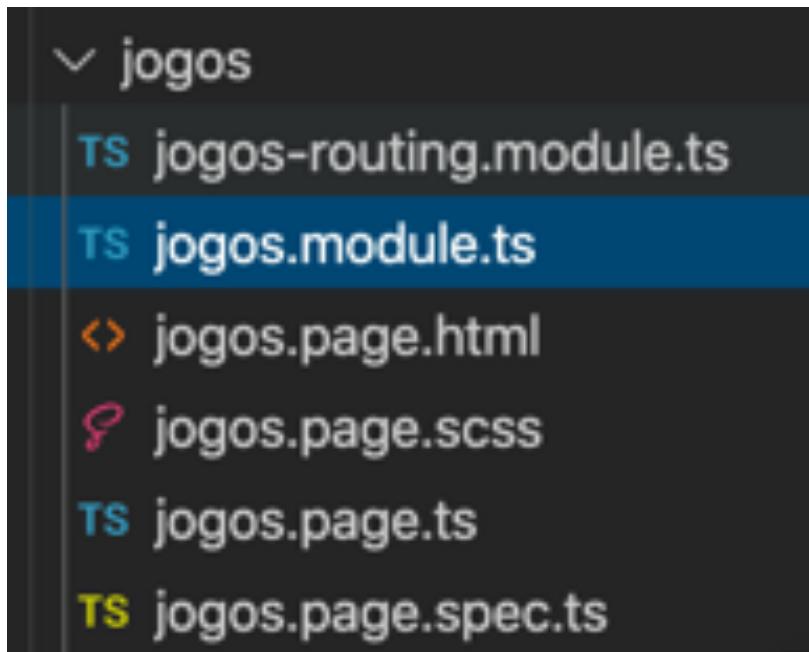
Agora, vamos iniciar a montagem da tela e programação do conteúdo

- PRONTO!
- Nosso componente está criado e configurado para uso

# Criando Componente

Agora, vamos iniciar a montagem da tela e programação do conteúdo

- Vamos fazer uso deste novo componente na página de jogos



# Criando Componente e

Agora, vamos iniciar a montagem da tela e programação do componente.

- Vamos fazer uso deste novo componente na página de jogos

ts jogos.module.ts ×

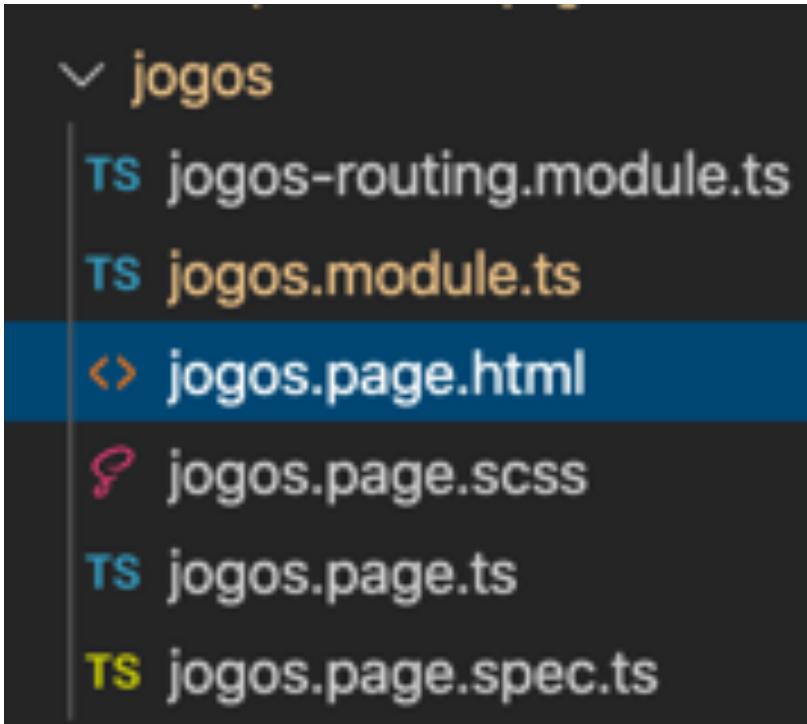
```
PlayingScore > src > app > jogos > ts jogos.module.ts > ...
1 import { NgModule } from '@angular/core';
2 import { CommonModule } from '@angular/common';
3 import { FormsModule } from '@angular/forms';
4
5 import { IonicModule } from '@ionic/angular';
6
7 import { JogosPageRoutingModule } from './jogos-routing.module';
8
9 import { JogosPage } from './jogos.page';
10 import { JogoCardComponentModule } from '../../components/jogo-card/jogo-card.module';
11
12 @NgModule({
13   imports: [
14     CommonModule,
15     FormsModule,
16     IonicModule,
17     JogosPageRoutingModule,
18     JogoCardComponentModule,
19   ],
20   declarations: [
21     JogosPage
22   ]
23 })
24 export class JogosPageModule {}
```

Adicionamos o uso do módulo do componente JogoCard

# Criando Componente

Agora, vamos iniciar a montagem da tela e programação do conteúdo

- Por fim, vamos adicionar o uso do componente no HTML



# Criando Componente

Agora, vamos iniciar a montagem da tela e programação do conteúdo

- Por fim, vamos adicionar o uso do componente no HTML

```
↳ jogos.page.html ×

PlayingScore > src > app > jogos > ↳ jogos.page.html > ...
19      </ion-tabs>
20      </ion-toolbar>
21  </ion-header>
22
23  <ion-content>
24      <app-jogo-card *ngFor="let game of gamesData"
25          name="{{ game.name }}"
26          description="{{ game.description }}>
27      </app-jogo-card>
28
29      <ion-fab vertical="bottom" horizontal="start" slot="fixed">
30          <ion-fab-button size="large"><ion-icon name="add"></ion-icon></ion-fab-button>
31      </ion-fab>
32  </ion-content>
```

# Criando Componente

Agora, vamos iniciar a montagem da tela e programação do conteúdo

- Por fim, vamos adicionar o uso do componente no HTML

```
↳ jogos.page.html ×

PlayingScore > src > app > jogos > ↳ jogos.page.html > ...
19           </ion-tabs>
20           </ion-toolbar>
21       </ion-header>
22
23   <ion-content>
24       <app-jogo-card *ngFor="let game of gamesData"
25           name="{{ game.name }}"
26           description="{{ game.description }}>
27   </app-jogo-card>
28
29   <ion-fab vertical="bottom" horizontal="start" slot="fixed">
30       <ion-fab-button size="large"><ion-icon name="add"></ion-icon></ion-fab-button>
31   </ion-fab>
32 </ion-content>
```

Utilizamos o novo componente como tag, com o nome configurado no atributo SELECTOR do componente

# Criando Componente

Agora, vamos iniciar a montagem da tela e programação do conteúdo

- Por fim, vamos adicionar o uso do componente no HTML

```
↳ jogos.page.html ×

PlayingScore > src > app > jogos > ↳ jogos.page.html > ...
19      </ion-tabs>
20      </ion-toolbar>
21  </ion-header>
22
23  <ion-content>
24    <app-jogo-card *ngFor="let game of gamesData"
25      name="{{ game.name }}"
26      description="{{ game.description }}>
27    </app-jogo-card>
28
29    <ion-fab vertical="bottom" horizontal="start" slot="fixed">
30      <ion-fab-button size="large"><ion-icon name="add"></ion-icon></ion-fab-button>
31    </ion-fab>
32  </ion-content>
```

O loop for do Angular,  
usamos neste componente

# Criando Componente

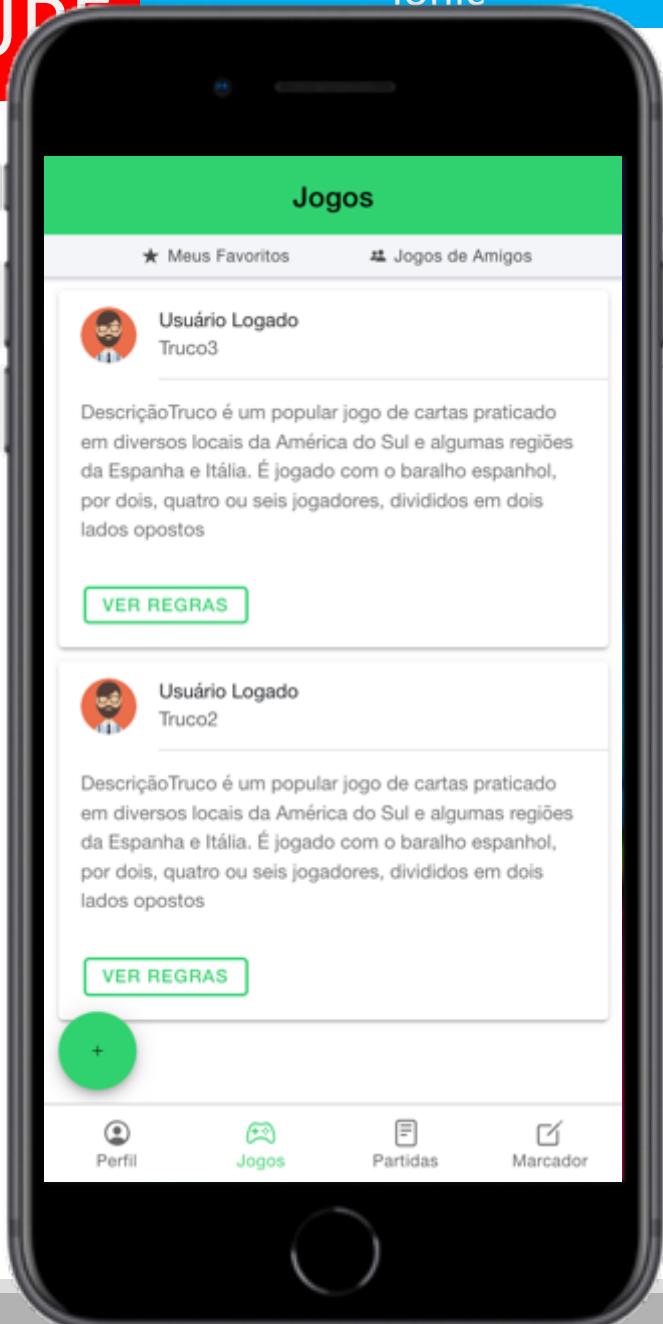
Agora, vamos iniciar a montagem da tela e programação do conteúdo

- Por fim, vamos adicionar o uso do componente no HTML

```
↳ jogos.page.html ×

PlayingScore > src > app > jogos > ↳ jogos.page.html > ...
19      </ion-tabs>
20      </ion-toolbar>
21  </ion-header>
22
23  <ion-content>
24      <app-jogo-card *ngFor="let game of gamesData"
25          name="{{ game.name }}"
26          description="{{ game.description }}>
27  </app-jogo-card>
28
29  <ion-fab vertical="bottom" horizontal="start" slot="fixed">
30      <ion-fab-button size="large"><ion-icon name="add"></ion-icon></ion-fab-button>
31  </ion-fab>
32  </ion-content>
```

Atribuímos valores aos 2  
Inputs configurados no  
componente



# Componente

Montagem da tela e programação do conteúdo  
para o uso do componente no HTML



# Organizando os Serviços

# Organizando os Serviços

---

Até aqui temos o serviço da API sendo consumido no arquivo `ApiService.ts`, na raiz do projeto

- Vamos organizar de uma forma que os serviços fiquem estruturados no local adequado

# Organizando os Serviços

Até aqui temos o serviço da API sendo consumido no arquivo ApiService.ts, na raiz do projeto

- Vamos organizar de uma forma que os serviços fiquem estruturados no local adequado

Então, dentro do projeto Ionic, vamos executar

- → ionic g

```
→ PlayingScore git:(master) ✘ ionic g
? What would you like to generate?
page
component
> service
module
class
directive
guard
```

Criaremos um “service”

# Organizando os Serviços

Até aqui temos o serviço da API sendo consumido no arquivo ApiService.ts, na raiz do projeto

- Vamos organizar de uma forma que os serviços fiquem estruturados no local adequado

Então, dentro do projeto Ionic, vamos executar

- → ionic g

Daremos o nome de "services/jogos"

```
→ PlayingScore git:(master) ✘ ionic g
? What would you like to generate? service
? Name/path of service: services/jogos
```

Desta forma o ionic irá gerar uma pasta services

# Organizando os Serviços

---

Até aqui temos o serviço da API sendo consumido no arquivo `ApiService.ts`, na raiz do projeto

- Podemos apagar o arquivo `jogos.service.spec.ts` pois não trabalharemos com teste

# Organizando os Serviços

---

Até aqui temos o serviço da API sendo consumido no arquivo `ApiService.ts`, na raiz do projeto

- Vamos passar todo o código de `ApiService.ts` para o arquivo `jogos.service.ts`
- Só tomaremos cuidado para trocar o nome da classe e para ajustar os imports com problema

# Organizando os Serviços

Até aqui temos o serviço da API sendo consumido no arquivo ApiService.ts, na raiz do projeto

- Vamos passar todo o código de ApiService.ts para o arquivo jogos.service.ts
- Só tomaremos cuidado para trocar o nome da classe e para ajustar os imports com problema

```
TS jogos.service.ts ×

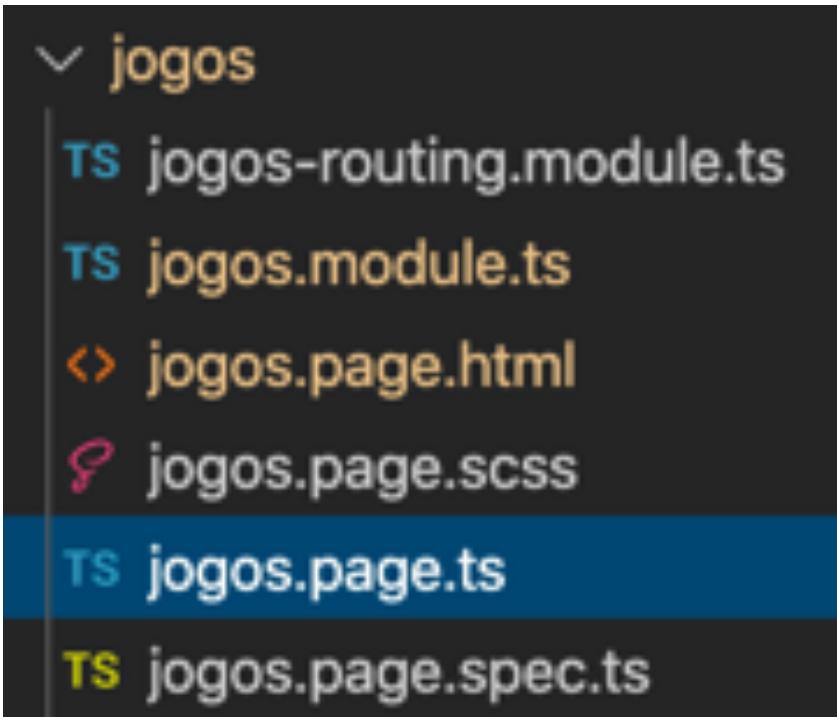
PlayingScore > src > app > services > TS jogos.service.ts > 🛡 JogosService

1 import { Injectable } from '@angular/core';
2 import { HttpClient, HttpHeaders, HttpErrorResponse } from '@angular/common/http';
3 import { Game } from '../models/game'; // Line 3
4 import { Observable, throwError } from 'rxjs';
5 import { retry, catchError } from 'rxjs/operators';
6
7 @Injectable({
8   providedIn: 'root'
9 })
10 export class JogosService { // Line 10
11
12   // API path
13   base_path = 'http://localhost:3000/games';
14
15   constructor(private http: HttpClient) { }
```

# Organizando os Serviços

Até aqui temos o serviço da API sendo consumido no arquivo ApiService.ts, na raiz do projeto

- Para fazermos uso deste novo serviço, precisamos alterar o TS da página de jogos



# Organizando os Serviços

Até aqui temos o serviço da API sendo consumido no arquivo `ApiService.ts`, na raiz do projeto

- Para fazermos uso deste novo serviço, precisamos alterar o TS da página de jogos

`TS jogos.page.ts ×`

```
PlayingScore > src > app > jogos > TS jogos.page.ts > ...
1 import { JogosService } from './services/jogos.service';
2 import { Component, OnInit } from '@angular/core';
3
4 @Component({
5   selector: 'app-jogos',
6   templateUrl: './jogos.page.html',
7   styleUrls: ['./jogos.page.scss'],
8 })
9 export class JogosPage implements OnInit {
10
11   gamesData: any;
12
13   constructor(public apiService: JogosService) {
14     this.gamesData = [];
15   }
16
17   ngOnInit() { }
```

# Organizando os Serviços

Até aqui temos o serviço da API sendo consumido no arquivo ApiService.ts, na raiz do projeto

- Para fazermos uso deste novo serviço, precisamos alterar o TS da página de jogos

TS jogos.page.ts ×

PlayingScore > src > app > jogos > **TS jogos.page.ts** > ...

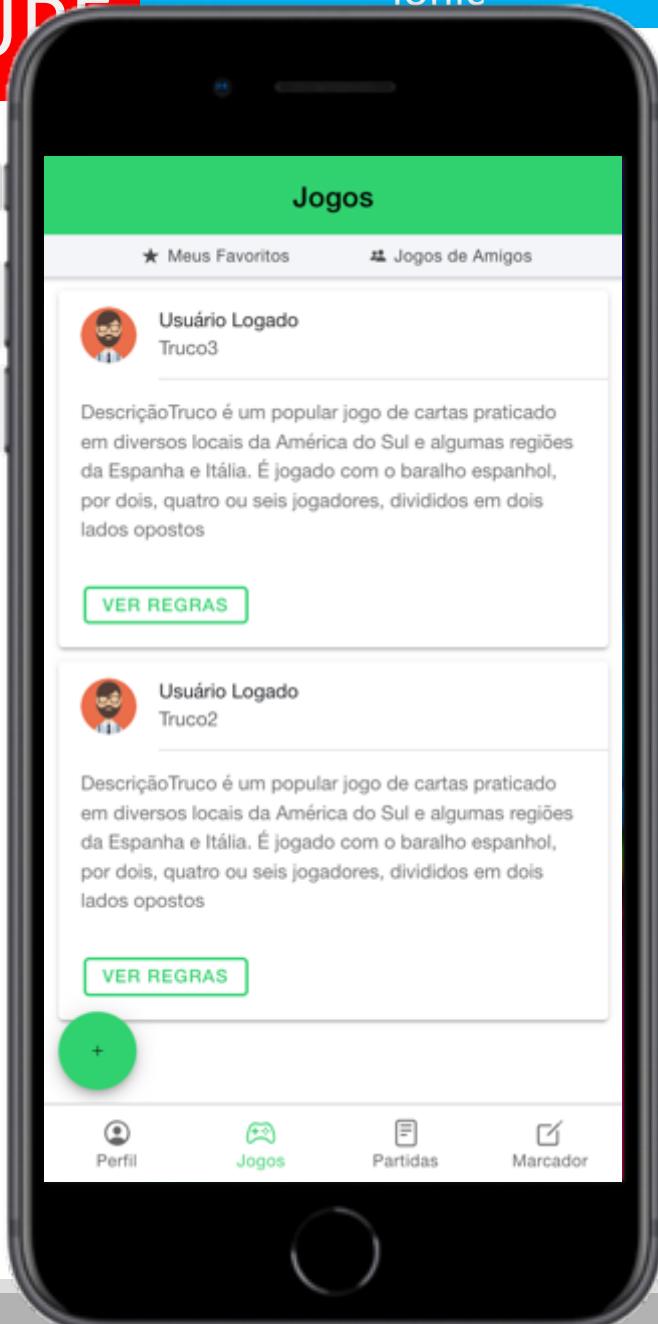
```
1 import { JogosService } from './services/jogos.service';
2 import { Component, OnInit } from '@angular/core';
3
4 @Component({
5   selector: 'app-jogos',
6   templateUrl: './jogos.page.html',
7   styleUrls: ['./jogos.page.scss'],
8 })
9 export class JogosPage implements OnInit {
10
11   gamesData: any;
12
13   constructor(public apiService: JogosService) {
14     this.gamesData = [];
15   }
16
17   ngOnInit() { }
```

# Organizando os Serviços

---

Até aqui temos o serviço da API sendo consumido no arquivo `ApiService.ts`, na raiz do projeto

- PRONTO!
- Agora temos uma pasta própria para configurar os serviços



# do os Serviços

o da API sendo consumido no arquivo ApiService.ts, na raiz do projeto

pasta própria para configurar os serviços

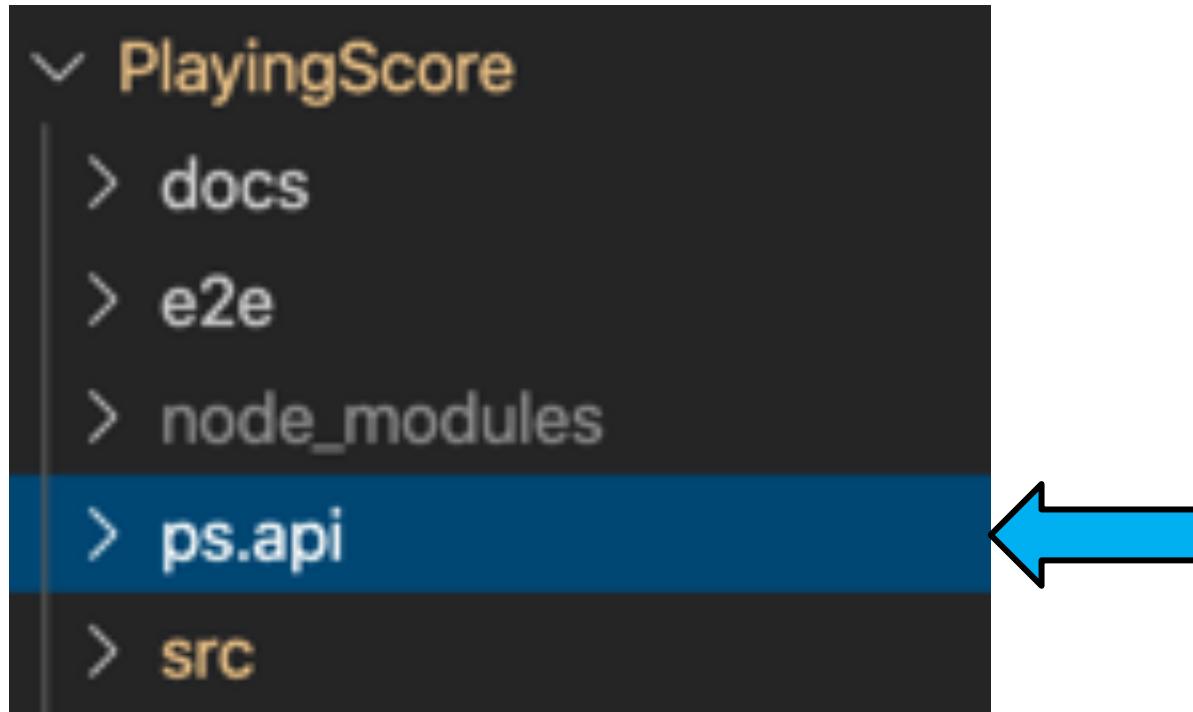


# Trabalhando com Autenticação

# Trabalhando com Autenticação

Agora, vamos fazer a autenticação do nosso app

- Para isso, inicialmente iremos projetar a API



# Trabalhando com Autenticação

---

Vamos fazer um pequeno ajusto no controlador do usuário, para que a senha seja criptografada com md5

- Primeiro, temos que instalar o pacote md5

Então, dentro do projeto da API, vamos executar

- → npm install md5

# Trabalhando com Autenticação

---

Vamos fazer um pequeno ajusto no controlador do usuário, para que a senha seja criptografada com md5

# Trabalhando com Autenticação

TS UserController.ts ×

```
PlayingScore > ps.api > src > controller > UserController.ts > ...
1 import { Request } from 'express';
2 import { getRepository } from "typeorm";
3 import { User } from "../entity/User";
4 import { BaseController } from "./BaseController";
5
6 import * as md5 from 'md5';
7
8 export class UserController extends BaseController<User> {
9
10    constructor() {
11        super(User);
12    }
13
14    async save(request: Request) {
15        // Conceito de desestruturação
16        let { name, nick, email, password } = <User>request.body;
17        // Validação dos campos obrigatórios
18        super.isRequired(name, 'Informe seu nome!');
19        super.isRequired(nick, 'Informe seu apelido!');
20        super.isRequired(email, 'Informe seu e-mail!');
21        super.isRequired(password, 'Informe sua senha!');
22
23        let _user = new User();
24        _user.name = name;
25        _user.nick = nick;
26        _user.email = email;
27        if (password)
28            _user.password = md5(password);
29
30        return super.save(_user);
31    }
32
33 }
```

# Trabalhando

TS UserController.ts ×

PlayingScore > ps.api > src > controller > **ts UserController.ts > ...**

```
1  import { Request } from 'express';
2
3  export class UserController {
4      constructor(private _repository: UserRepository) {}
5
6      async index(request: Request, response: Response) {
7          const users = await this._repository.findAll();
8
9          return response.json(users);
10     }
11
12     async show(request: Request, response: Response) {
13         const user = await this._repository.findById(request.params.id);
14
15         if (!user) {
16             return response.status(404).json({ error: 'User not found' });
17         }
18
19         return response.json(user);
20     }
21
22     async save(request: Request, response: Response) {
23         // Conceito de desestruturação
24         let { name, nick, email, password } = <User>request.body;
25
26         // Validação dos campos obrigatórios
27         super.isRequired(name, 'Informe seu nome!');
28         super.isRequired(nick, 'Informe seu apelido!');
29         super.isRequired(email, 'Informe seu e-mail!');
30         super.isRequired(password, 'Informe sua senha!');
31
32         let _user = new User();
33         _user.name = name;
34         _user.nick = nick;
35         _user.email = email;
36         if (password)
37             _user.password = md5(password);
38
39         return super.save(_user);
40     }
41
42     async update(request: Request, response: Response) {
43         const user = await this._repository.findById(request.params.id);
44
45         if (!user) {
46             return response.status(404).json({ error: 'User not found' });
47         }
48
49         const { name, nick, email, password } = request.body;
50
51         if (name)
52             user.name = name;
53         if (nick)
54             user.nick = nick;
55         if (email)
56             user.email = email;
57         if (password)
58             user.password = md5(password);
59
60         return super.update(user);
61     }
62
63     async delete(request: Request, response: Response) {
64         const user = await this._repository.findById(request.params.id);
65
66         if (!user) {
67             return response.status(404).json({ error: 'User not found' });
68         }
69
70         return super.delete(user);
71     }
72 }
```

Trabalhamos com conceito de desestruturação do ECM6

# Trabalhando

```
TS UserController.ts ×  
PlayingScore > ps.api > src > controller > UserController.ts > ...  
1 import { Request } from 'express';  
  
14     async save(request: Request) {  
15         // Conceito de desestruturação  
16         let { name, nick, email, password } = <User>request.body;  
17         // Validação dos campos obrigatórios  
18         super.isRequired(name, 'Informe seu nome!');  
19         super.isRequired(nick, 'Informe seu apelido!');  
20         super.isRequired(email, 'Informe seu e-mail!');  
21         super.isRequired(password, 'Informe sua senha!');  
22  
23         let _user = new User();  
24         _user.name = name;  
25         _user.nick = nick;  
26         _user.email = email;  
27         if (password)  
28             _user.password = md5(password);  
29
```

```
30         return super.save(_user);  
31     }  
32 }  
33 }
```

Mantemos a validação, mas nos campos em novo formato

# Trabalhando

TS UserController.ts ×

PlayingScore > ps.api > src > controller > **ts UserController.ts > ...**

```
1   import { Request } from 'express';
2
3
4   async save(request: Request) {
5       // Conceito de desestruturação
6       let { name, nick, email, password } = <User>request.body;
7       // Validação dos campos obrigatórios
8       super.isRequired(name, 'Informe seu nome!');
9       super.isRequired(nick, 'Informe seu apelido!');
10      super.isRequired(email, 'Informe seu e-mail!');
11      super.isRequired(password, 'Informe sua senha!');
12
13      let _user = new User();
14      _user.name = name;
15      _user.nick = nick;
16      _user.email = email;
17      if (password)
18          _user.password = md5(password);
19
20      return super.save(_user);
21
22
23
24
25
26
27
28
29
30
31
32
33 }
```

Instanciamos um objeto de  
User()

# Trabalhando

```
TS UserController.ts ×  
PlayingScore > ps.api > src > controller > UserController.ts > ...  
1 import { Request } from 'express';  
  
14     async save(request: Request) {  
15         // Conceito de desestruturação  
16         let { name, nick, email, password } = <User>request.body;  
17         // Validação dos campos obrigatórios  
18         super.isRequired(name, 'Informe seu nome!');  
19         super.isRequired(nick, 'Informe seu apelido!');  
20         super.isRequired(email, 'Informe seu e-mail!');  
21         super.isRequired(password, 'Informe sua senha!');  
22  
23         let _user = new User();  
24             _user.name = name;  
25             _user.nick = nick;  
26             _user.email = email;  
27             if (password)  
28                 _user.password = md5(password);  
29  
30             return super.save(_user);  
31         }  
32     }  
33 }
```

O atributo password trabalhamos com a criptografia md5

# Trabalhando com Autenticação

---

Por segurança, antes de executarmos as regras de negócio no controlador, temos sempre que verificar a autenticidade do usuário

- Geralmente controlamos isso via TOKEN na requisição
- E é dessa forma que faremos neste projeto

# Trabalhando com Autenticação

---

Por segurança, antes de executarmos as regras de negócio no controlador, temos sempre que verificar a autenticidade do usuário

- Também, vamos trabalhar com o JWT
- *JWT é interessante para que possamos criptografar nosso token, não deixando exposto*

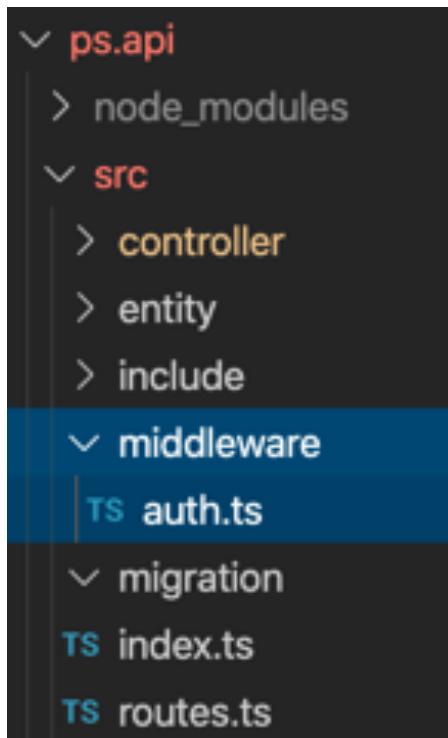
Então, dentro do projeto da API, vamos executar

- → `npm install jsonwebtoken`

# Trabalhando com Autenticação

Por segurança, antes de executarmos as regras de negócio no controlador, temos sempre que verificar a autenticidade do usuário

- Vamos criar o arquivo *auth.ts* dentro da pasta *middleware* (deve ser criada)



# Trabalhando com Autenticação

```
TS auth.ts  X

PlayingScore > ps.api > src > middleware > TS auth.ts > ...

1 import { Response, Request } from 'express';
2 import { verify } from 'jsonwebtoken';
3 import config from "../include/config";
4
5 export default async (req: Request, res: Response, next: Function) => {
6     // Buscamos por um TOKEN em algum nível de dados da requisição
7     let token = req.query.token || req.headers['x-token-access'] || req.body.token;
8
9     if (token) {
10         try {
11             // Guarda as infos de usuário
12             let _userAuth = verify(token, config.secretKey);
13             req.userAuth = _userAuth;
14             // Libera a rota para que o express continue sua execução
15             next();
16         } catch (error) {
17             res.status(401).send({
18                 message: 'Token inválido!'
19             });
20             return;
21         }
22     } else {
23         res.status(401).send({
24             message: 'Serviço disponível apenas para usuário autenticados!'
25         });
26         return;
27     }
28 }
29 }
```

Vamos desenvolver uma classe assíncrona que irá analisar as requisições de usuário

# Trabalhando com Autenticação

TS auth.ts

PlayingScore > ps.api > src > middleware > **TS auth.ts** > ...

```
1 import { Response, Request } from 'express';
2 import { verify } from 'jsonwebtoken';
3 import config from '../include/config';

18     message: 'Token inválido!'
19 });
20 return;
21 }
22 } else {
23     res.status(401).send({
24         message: 'Serviço disponível apenas para usuário autenticados!'
25 });
26 return;
27 }
28 }
29 }
```

Vamos desenvolver uma classe assíncrona que irá analisar as requisições de usuário

# Trabalhando com Autenticação

TS auth.ts X

PlayingScore > ps.api > src > middleware > **TS auth.ts** > ...

```
1 import { Response, Request } from 'express';
```

```
5 export default async (req: Request, res: Response, next: Function) => {
6     // Buscamos por um TOKEN em algum nível de dados da requisição
7     let token = req.query.token || req.headers['x-token-access'] || req.body.token;
8
9     if (token) {
10         try {
11             // Guarda as infos de usuário
12             let _userAuth = verify(token, config.secretKey);
13             req.userAuth = _userAuth;
14             // Libera a rota para que o express continue sua execução
15             next();
16         } catch (error) {
17             res.status(401).send({
18                 message: 'Token inválido!'
19             });
20             return;
21         }
22     } else {
23         res.status(401).send({
24             message: 'Serviço disponível apenas para usuário autenticados!'
25         });
26         return;
27     }
28 }
29 }
```

Pegamos o TOKEN nos níveis de requisição possíveis: query (URL), cabeçalho ou dados

# Trabalhando com Autenticação

TS auth.ts X

PlayingScore > ps.api > src > middleware > **ts auth.ts** > ...

```
1 import { Response, Request } from 'express';
```

```
5 export default async (req: Request, res: Response, next: Function) => {
6     // Buscamos por um TOKEN em algum nível de dados da requisição
7     let token = req.query.token || req.headers['x-token-access'] || req.body.token;
8
9     if (token) {
10         try {
11             // Guarda as infos de usuário
12             let _userAuth = verify(token, config.secretKey);
13             req.userAuth = _userAuth;
14             // Libera a rota para que o express continue sua execução
15             next();
16         } catch (error) {
17             res.status(401).send({
18                 message: 'Token inválido!'
19             });
20             return;
21         }
22     } else {
23         res.status(401).send({
24             message: 'Serviço disponível apenas para usuário autenticados!'
25         });
26         return;
27     }
28 }
29 }
```

Caso encontre um TOKEN, analisa se é válido

# Trabalhando com Autenticação

TS auth.ts X

PlayingScore > ps.api > src > middleware > **TS auth.ts** > ...

```
1 import { Response, Request } from 'express';
```

```
5 export default async (req: Request, res: Response, next: Function) => {
6     // Buscamos por um TOKEN em algum nível de dados da requisição
7     let token = req.query.token || req.headers['x-token-access'] || req.body.token;
8
9     if (token) {
10         try {
11             // Guarda as infos de usuário
12             let _userAuth = verify(token, config.secretKey);
13             req.userAuth = _userAuth;
14
15             // Libera a rota para que o express continue sua execução
16             next();
17         } catch (error) {
18             res.status(401).send({
19                 message: 'Token inválido!'
20             });
21             return;
22         }
23     } else {
24         res.status(401).send({
25             message: 'Serviço disponível apenas para usuário autenticados!'
26         });
27     }
28 }
29 }
```

Utilizamos o verify do JWT  
Obs: depois configuramos o secretKey

# Trabalhando com Autenticação

TS auth.ts X

PlayingScore > ps.api > src > middleware > **TS auth.ts** > ...

```
1 import { Response, Request } from 'express';
```

```
5 export default async (req: Request, res: Response, next: Function) => {
6     // Buscamos por um TOKEN em algum nível de dados da requisição
7     let token = req.query.token || req.headers['x-token-access'] || req.body.token;
8
9     if (token) {
10         try {
11             // Guarda as infos de usuário
12             let _userAuth = verify(token, config.secretKey);
13             req.userAuth = _userAuth;
14             // Libera a rota para que o express continue sua execução
15             next();
16         } catch (error) {
17             res.status(401).send({
18                 message: 'Token inválido!'
19             });
20             return;
21         }
22     } else {
23         res.status(401).send({
24             message: 'Serviço disponível apenas para usuário autenticados!'
25         });
26         return;
27     }
28 }
29 }
```

Estando OK, libera a rota

```
=> {
| req.body.token;
```

ação

autenticados!

# Trabalhando com Autenticação

TS auth.ts X

PlayingScore > ps.api > src > middleware > **ts auth.ts** > ...

```
1 import { Response, Request } from 'express';
```

```
5 export default async (req: Request, res: Response, next: Function) => {
6     // Buscamos por um TOKEN em algum nível de dados da requisição
7     let token = req.query.token || req.headers['x-token-access'] || req.body.token;
8
9     if (token) {
10         try {
11             // Guarda as infos de usuário
12             let _userAuth = verify(token, config.secretKey);
13             req.userAuth = _userAuth;
14             // Libera a rota para que o express continue sua execução
15             next();
16         } catch (error) {
17             res.status(401).send({
18                 message: 'Token inválido!'
19             });
20             return;
21         }
22     } else {
23         res.status(401).send({
24             message: 'Serviço disponível apenas para usuário autenticados!'
25         });
26         return;
27     }
28 }
29 }
```

O TOKEN não sendo válido, retorna o erro

# Trabalhando com Autenticação

TS auth.ts X

PlayingScore > ps.api > src > middleware > **TS auth.ts** > ...

```
1 import { Response, Request } from 'express';

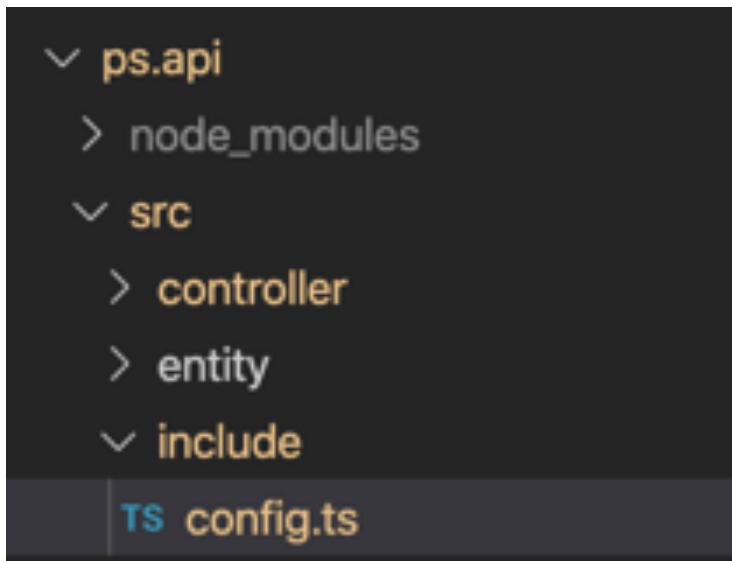
5 export default async (req: Request, res: Response, next: Function) => {
6     // Buscamos por um TOKEN em algum nível de dados da requisição
7     let token = req.query.token || req.headers['x-token-access'] || req.body.token;
8
9     if (token) {
10         try {
11             // Guarda as infos de usuário
12             let _userAuth = verify(token, config.secretKey);
13             req.userAuth = _userAuth;
14             // Libera a rota para que o express continue sua execução
15             next();
16         } catch (error) {
17             res.status(401).send({
18                 message: 'Token inválido!'
19             });
20             return;
21         }
22     } else {
23         res.status(401).send({
24             message: 'Serviço disponível apenas para usuário autenticados!'
25         });
26         return;
27     }
28
29 }
```

Caso não tenha um TOKEN, retorna erro informando que as rotas são para usuários autenticados

# Trabalhando com Autenticação

Por segurança, antes de executarmos as regras de negócio no controlador, temos sempre que verificar a autenticidade do usuário

- No código anterior, utilizamos a config secretKey
- Temos que configurar esta chave no include/config.ts



# Trabalhando com Autenticação

Por segurança, antes de executarmos as regras de negócio no controlador, temos sempre que verificar a autenticidade do usuário

- No código anterior, utilizamos a config secretKey
- Temos que configurar esta chave no include/config.ts

TS config.ts ×

PlayingScore > ps.api > src > include > TS config.ts > ...

```
1  export default {  
2    port: process.env.PORT || 3000,  
3    secretKey: process.env.SECRETKEY || 'd0dbe97c-a7c4-4e2e-8053-aa78ca8a1cba'  
4  }
```

# Trabalhando com Autenticação

Por segurança, antes de executarmos as regras de negócio no controlador, temos sempre que verificar a autenticidade do usuário

- No código anterior, utilizamos a config secretKey
- Temos que configurar esta chave no include/config.ts

TS config.ts ×

PlayingScore > ps.api > src > include > TS config.ts > ...

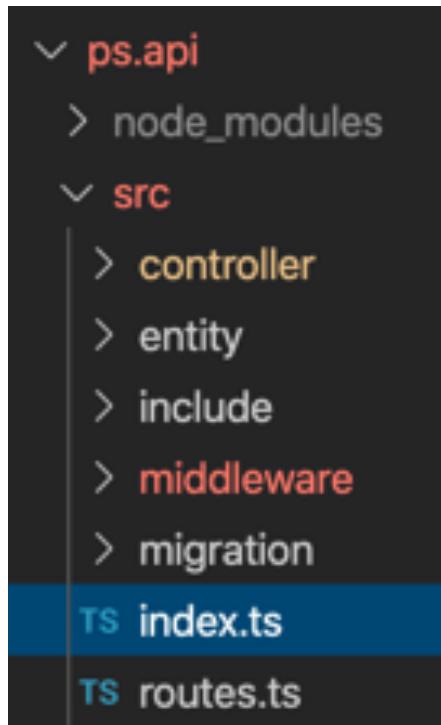
```
1  export default {  
2    port: process.env.PORT || 3000,  
3    secretKey: process.env.SECRETKEY || 'd0dbe97c-a7c4-4e2e-8053-aa78ca8a1cba'  
4  }
```

Para gerar esta chave utilizei um gerador de GUID  
<https://www.guidgenerator.com/online-guid-generator.aspx>

# Trabalhando com Autenticação

Por segurança, antes de executarmos as regras de negócio no controlador, temos sempre que verificar a autenticidade do usuário

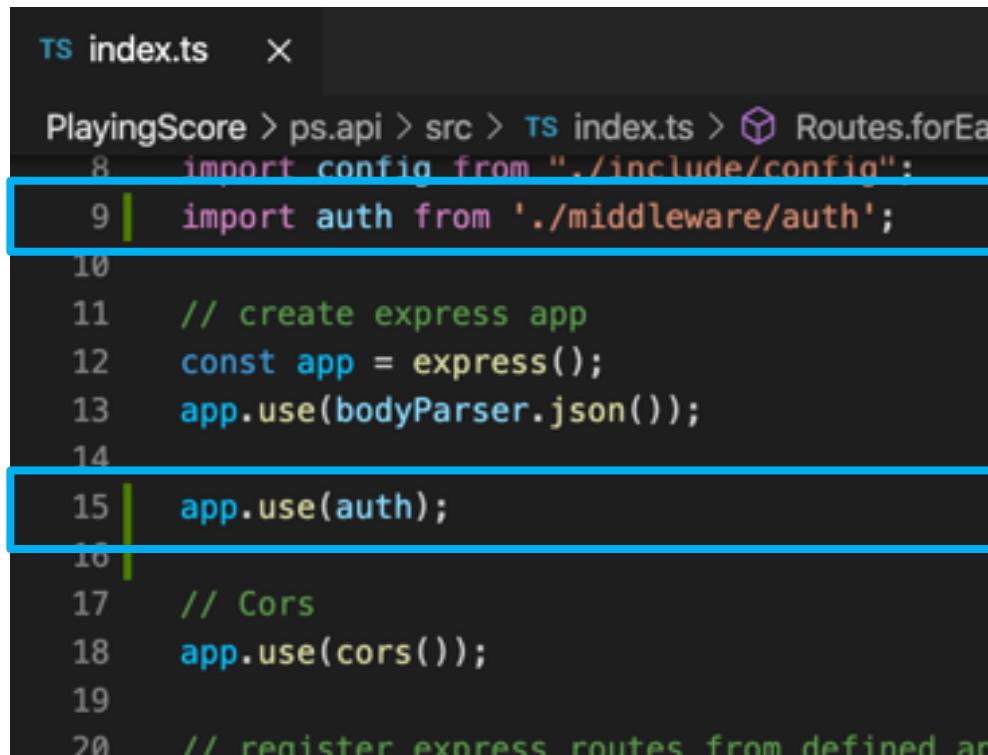
- Com o middleware criado, vamos adicionar seu uso no arquivo principal da API, index.ts



# Trabalhando com Autenticação

Por segurança, antes de executarmos as regras de negócio no controlador, temos sempre que verificar a autenticidade do usuário

- Com o middleware criado, vamos adicionar seu uso no arquivo principal da API, index.ts



```
ts index.ts  X

PlayingScore > ps.api > src > ts index.ts > Routes.forEach
8 import config from './include/config';
9 import auth from './middleware/auth';
10
11 // create express app
12 const app = express();
13 app.use(bodyParser.json());
14
15 app.use(auth);
16
17 // Cors
18 app.use(cors());
19
20 // register express routes from defined an
```

# Trabalhando com Autenticação

Por segurança, antes de executarmos as regras de negócio no controlador, temos sempre que verificar a autenticidade do usuário

- Alguns rotas precisam ser públicas
  - Cadastro de usuário e validação de usuário, por exemplo
- Então, vamos configurar e organizar nossas rotas públicas no config.ts

```
src
  controller
  entity
  include
    config.ts
```

# Trabalhando com Autenticação

Por segurança, antes de executarmos as regras de negócio no controlador, temos sempre que verificar a autenticidade do usuário

- Alguns rotas precisam ser públicas
  - Cadastro de usuário e validação de usuário, por exemplo
- Então, vamos configurar e organizar nossas rotas públicas

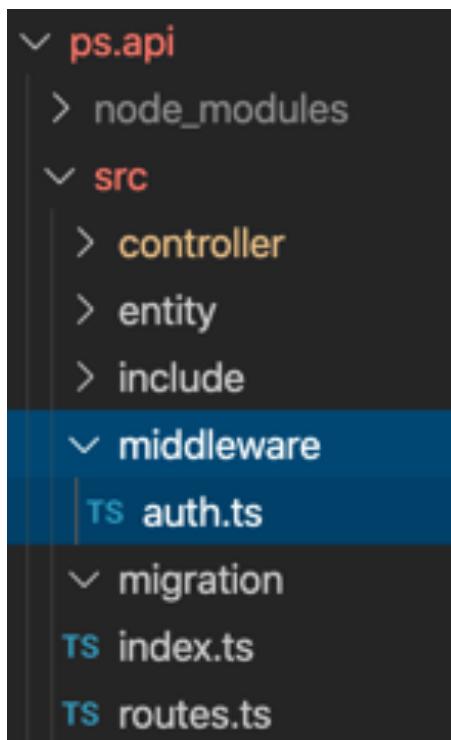
```
TS config.ts  ×

PlayingScore > ps.api > src > include > TS config.ts > ...
1  export default {
2    port: process.env.PORT || 3000,
3    secretKey: process.env.SECRETKEY || 'd0dbe97c-a7c4-4e2e-8053-aa78ca8a1cba',
4    publicRoutes: process.env.PUBLICROUTES || [
5      'users',
6      'auth'
7    ]
8 }
```

# Trabalhando com Autenticação

Por segurança, antes de executarmos as regras de negócio no controlador, temos sempre que verificar a autenticidade do usuário

- Agora precisamos tratar estas rotas no middleware



# Trabalhando com Autenticação

Por segurança, antes de executarmos as regras de negócio no controlador, temos sempre que verificar a autenticidade do usuário

- Agora precisamos tratar estas rotas no middleware

Definimos um controle que irá buscar a rota que está sendo solicitada, caso esta rota estiver dentro das rotas públicas, não valida o TOKEN

TS auth.ts X

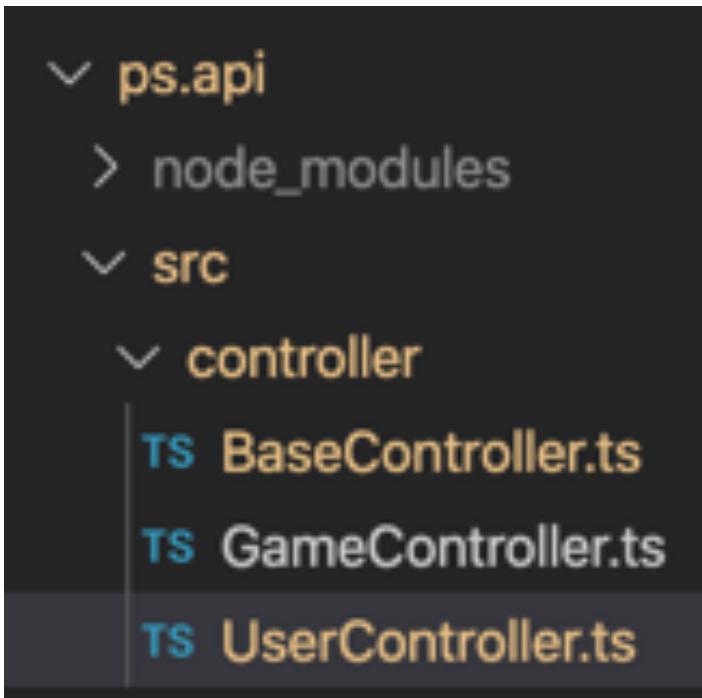
PlayingScore > ps.api > src > middleware > TS auth.ts > default

```
5  export default async (req: Request, res: Response, next: Function) => {
6      // Buscamos por um TOKEN em algum nível de dados da requisição
7      let token = req.query.token || req.headers['x-token-access'] || req.body.token;
8      let publicRoutes = <Array<String>>config.publicRoutes;
9      let isPublicRoute: boolean = false;
10
11     publicRoutes.forEach(url => {
12         let isPublic = req.url.includes(url);
13         if (isPublic)
14             isPublicRoute = true;
15     });
16
17     if (isPublicRoute)
18         next();
19     else if (token) {
20         try {
21             // Guarda as infos de usuário
22         } catch (err) {
23             res.status(401).send('Unauthorized');
24         }
25     }
26 }
```

# Trabalhando com Autenticação

Por segurança, antes de executarmos as regras de negócio no controlador, temos sempre que verificar a autenticidade do usuário

- Agora, vamos criar o método de autenticação no controlador do usuário



The image shows a dark-themed file explorer interface. On the left, there's a sidebar with navigation icons. The main area displays a hierarchical file structure:

- ps.api** (highlighted in orange)
- > node\_modules
- src** (highlighted in orange)
- < controller (highlighted in orange)
- TS BaseController.ts** (highlighted in blue)
- TS GameController.ts** (highlighted in blue)
- TS UserController.ts** (highlighted in blue)

The files are color-coded: orange for the main folder and blue for the TypeScript files.

# Trabalhando com Autenticação

TS UserController.ts ×

```
PlayingScore > ps.api > src > controller > ts UserController.ts > UserController > save
```

```
14
15     async auth(request: Request) {
16         let { email, password } = request.body;
17         if (!email || !password) {
18             return {
19                 status: 400,
20                 message: 'Para entrar no sistema informe e-mail e senha!'
21             };
22         }
23
24         let user = await this.repository.findOne({ email: email, password: md5(password) });
25         if (user) {
26             let _payload = {
27                 uid: user.uid,
28                 name: user.name,
29                 nick: user.nick,
30                 email: user.email
31             }
32             return {
33                 status: 200,
34                 message: {
35                     user: _payload,
36                     token: sign({
37                         _payload,
38                         tm: new Date().getTime()
39                     }, config.secretKey)
40                 }
41             }
42         }
43         else {
44             return {
45                 status: 404,
46                 message: 'Ocorreram erros para efetuar o login!'
47             }
48         }
49     }
```

# Trabalhando com Autenticação

ts UserController.ts ×

```
PlayingScore > ps.api > src > controller > ts UserController.ts > UserController > save
14
15     async auth(request: Request) {
16         let { email, password } = request.body;
17         if (!email || !password) {
18             return {
19                 status: 400,
20                 message: 'Para entrar no sistema informe e-mail e senha!'
21             };
22         }
23     }
24
25     async login(user: User, password: string): Promise<User> {
26         const userFromDB = await this.userService.getUserByEmail(user.email);
27         if (!userFromDB) {
28             return null;
29         }
30         const isUserValid = await bcrypt.compare(password, userFromDB.password);
31         if (!isUserValid) {
32             return null;
33         }
34         const token = jwt.sign({ user }, config.secretKey)
35         return { ...user, token };
36     }
37
38     async logout(user: User): Promise<void> {
39         const userFromDB = await this.userService.getUserByEmail(user.email);
40         if (!userFromDB) {
41             return null;
42         }
43         const isUserValid = await bcrypt.compare(password, userFromDB.password);
44         if (!isUserValid) {
45             return null;
46         }
47         const token = jwt.sign({ user }, config.secretKey)
48         return { ...user, token };
49     }
50 }
```

O método auth irá receber um json contendo o email e senha, sendo obrigatórios

```
15     async auth(request: Request) {
16         let { email, password } = request.body;
17         if (!email || !password) {
18             return {
19                 status: 400,
20                 message: 'Para entrar no sistema informe e-mail e senha!'
21             };
22         }
23     }
24
25     async login(user: User, password: string): Promise<User> {
26         const userFromDB = await this.userService.getUserByEmail(user.email);
27         if (!userFromDB) {
28             return null;
29         }
30         const isUserValid = await bcrypt.compare(password, userFromDB.password);
31         if (!isUserValid) {
32             return null;
33         }
34         const token = jwt.sign({ user }, config.secretKey)
35         return { ...user, token };
36     }
37
38     async logout(user: User): Promise<void> {
39         const userFromDB = await this.userService.getUserByEmail(user.email);
40         if (!userFromDB) {
41             return null;
42         }
43         const isUserValid = await bcrypt.compare(password, userFromDB.password);
44         if (!isUserValid) {
45             return null;
46         }
47         const token = jwt.sign({ user }, config.secretKey)
48         return { ...user, token };
49     }
50 }
```

```
39         const userFromDB = await this.userService.getUserByEmail(user.email);
40         if (!userFromDB) {
41             return null;
42         }
43         const isUserValid = await bcrypt.compare(password, userFromDB.password);
44         if (!isUserValid) {
45             return null;
46         }
47         const token = jwt.sign({ user }, config.secretKey)
48         return { ...user, token };
49     }
50 }
```

# Trabalhando com Autenticação

```
TS UserController.ts ×  
PlayingScore > ps.api > src > controller > UserController.ts > UserController > save  
14  
15     async auth(request: Request) {  
16         let { email, password } = request.body;
```

```
24         let user = await this.repository.findOne({ email: email, password: md5(password) });  
25         if (user) {  
26             let _payload = {  
27                 uid: user.uid,  
28                 name: user.name,  
29                 nick: user.nick,  
30                 email: user.email  
31             };  
32             return {  
33                 status: 200,  
34                 message: {  
35                     user: _payload,  
36                     token: sign({  
37                         _payload,  
38                         tm: new Date().getTime()  
39                     }, config.secretKey)  
40                 }  
41             }  
42         }  
43         else {  
44             return {  
45                 status: 404,  
46                 message: 'Ocorreram erros para efetuar o login!'  
47             }  
48         }  
49     }
```

Utilizando o repository do classe base, vamos executar o findOne para procurar o usuário

# Trabalhando com A+

```
TS UserController.ts ×  
PlayingScore > ps.api > src > controller > TS UserController.ts > UserController > save  
14  
15 |     async auth(request: Request) {  
16 |         let { email, password } = request.body;  
17 |         let user = await this.repository.findOne({ email, password: md5(password) });  
18 |         if (user) {  
19 |             return user;  
20 |         }  
21 |         throw new Error("User not found or password is incorrect");  
22 |     }  
23 | }  
24 |  
25 |
```

## TS BaseController.ts ×

```
PlayingScore > ps.api > src > controller > TS BaseController.ts > BaseController  
1  
2  
3  
4  
5 | export abstract class BaseController<GENERIC_CLASS> extends EntityController<GENERIC_CLASS> {  
6 |     protected _repository: Repository<GENERIC_CLASS>;  
7 |  
8 |     get repository(): Repository<GENERIC_CLASS> {  
9 |         return this._repository;  
10 |     }  
11 |  
12 |     constructor(entity: any) {  
13 |         super(entity);  
14 |         this._repository = entity === User ? UserRepository : null;  
15 |     }  
16 | }
```

Como o repository é um atributo privado da classe base, precisamos de um método GET para podermos utilizar na classe filha

# Trabalhando com Autenticação

```
TS UserController.ts ×  
PlayingScore > ps.api > src > controller > UserController.ts > UserController > save  
14  
15     async auth(request: Request) {  
16         let { email, password } = request.body;
```

```
24         let user = await this.repository.findOne({ email: email, password: md5(password) });  
25         if (user) {  
26             let _payload = {  
27                 uid: user.uid,  
28                 name: user.name,  
29                 nick: user.nick,  
30                 email: user.email  
31             };  
32             return {  
33                 status: 200,  
34                 message: {  
35                     user: _payload,  
36                     token: sign({  
37                         _payload,  
38                         tm: new Date().getTime()  
39                     }, config.secretKey)  
40                 }  
41             };  
42         }  
43         else {  
44             return {  
45                 status: 404,  
46                 message: 'Ocorreram erros para efetuar o login!'  
47             };  
48         }  
49     }
```

Caso o usuário exista, vamos retornar um json de código 200 contendo os dados encontrados

# Trabalhando com Autenticação

ts UserController.ts ×

PlayingScore > ps.api > src > controller > ts UserController.ts > UserController > save

```
14  
15     async auth(request: Request) {  
16         let { email, password } = request.body;
```

```
24         let user = await this.repository.findOne({ email: email, password: md5(password) });  
25         if (user) {  
26             let _payload = {  
27                 uid: user.uid,  
28                 name: user.name,  
29                 nick: user.nick,  
30                 email: user.email  
31             }  
32             return {  
33                 status: 200,  
34                 message: {  
35                     user: _payload,  
36                     token: sign({  
37                         _payload,  
38                         tm: new Date().getTime()  
39                     }, config.secretKey)  
40                 }  
41             }  
42         }  
43         else {  
44             return {  
45                 status: 404,  
46                 message: 'Ocorreram erros para efetuar o login!'  
47             }  
48         }  
49     }
```

A mensagem retornada conterá o token criptografado pelo JWT (método sign)

# Trabalhando com Autenticação

ts UserController.ts ×

PlayingScore > ps.api > src > controller > ts UserController.ts > UserController > save

```
14
15     async auth(request: Request) {
16         let { email, password } = request.body;
```

```
24         let user = await this.repository.findOne({ email: email, password: md5(password) });
25         if (user) {
26             let _payload = {
27                 uid: user.uid,
28                 name: user.name,
29                 nick: user.nick,
30                 email: user.email
31             }
32             return {
33                 status: 200,
34                 message: {
35                     user: _payload,
36                     token: sign({
37                         _payload,
38                         tm: new Date().getTime()
39                     }, config.secretKey)
40                 }
41             }
42         }
43         else {
44             return {
45                 status: 404,
46                 message: 'Ocorreram erros para efetuar o login!'
47             }
48         }
49     }
```

```
49 }
```

Caso o usuário não seja encontrado, retornaremos o erro

404

# Trabalhando com Autenticação

ts UserController.ts ×

PlayingScore > ps.api > src > controller > ts UserController.ts > UserController > save

```
14
15     async auth(request: Request) {
16         let { email, password } = request.body;
17         if (!email || !password) {
18             return {
19                 status: 400,
20                 message: 'Para entrar no sistema informe e-mail e senha!'
21             };
22         }
23     }
```

```
1 import { Request } from 'express';
2 import { User } from "../entity/User";
3 import { BaseController } from "./BaseController";
4 import { sign } from 'jsonwebtoken';
5 import config from "../include/config";
6 import * as md5 from 'md5';

7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
```

# Trabalhando com Autenticação

Por segurança, antes de executarmos as regras de negócio no controlador, temos sempre que verificar a autenticidade do usuário

- Agora, precisamos criar uma rota para a autenticação

```
src
  controller
  entity
  include
  middleware
  migration
  index.ts
  routes.ts
```

TS routes.ts X

PlayingScore &gt; ps.api &gt; src &gt; TS routes.ts &gt; ...

```
1 import { UserController } from "./controller/UserController";
2 import { GameController } from "./controller/GameController";
3
4 export const Routes = [
5     { method: "get", route: "/users", controller: UserController, action: "all" },
6     { method: "get", route: "/users/:id", controller: UserController, action: "one" },
7     { method: "post", route: "/users", controller: UserController, action: "save" },
8     { method: "post", route: "/auth", controller: UserController, action: "auth" }, // Line 8 highlighted with a blue border
9     { method: "delete", route: "/users/:id", controller: UserController, action: "remove" },
10
11    { method: "get", route: "/games", controller: GameController, action: "all" },
12    { method: "get", route: "/games/:id", controller: GameController, action: "one" },
13    { method: "post", route: "/games", controller: GameController, action: "save" },
14    { method: "delete", route: "/games/:id", controller: GameController, action: "remove" }
15];
```

# Trabalhando com Autenticação

---

Por segurança, antes de executarmos as regras de negócio no controlador, temos sempre que verificar a autenticidade do usuário

- Agora, precisamos criar uma rota para a autenticação

Vamos testar nossa autenticação no Insomnia ou PostMan

# Trabalhando com Autenticação

Por segurança, antes de executarmos as regras de negócio no controlador, temos sempre que verificar a autenticidade do usuário

- Agora, precisamos criar uma rota para a autenticação

Vamos fazer um POST para localhost:3000/users com dados de cadastro

The screenshot shows a Postman request for a POST operation to the endpoint `localhost:3000/users`. The request was sent via JSON, and the response status was `200 OK` with a response time of `174 ms` and a size of `365 B`. The request body contained user registration data, and the response body showed the newly created user record with an auto-generated ID.

JSON	Auth	Query	Header	Docs	Preview	Header	Cookie	Timeline
<pre>1+ { 2   "name": "Guilherme Afonso Madalozzo", 3   "nick": "@guimadalozzo", 4   "email": "guimadalozzo@gmail.com", 5   "password": "123456" 6 }</pre>					<pre>1+ { 2   "name": "Guilherme Afonso Madalozzo", 3   "nick": "@guimadalozzo", 4   "email": "guimadalozzo@gmail.com", 5   "password": "e10adc3949ba59abbe56e057f20f883e", 6   "createdAt": "2020-07-15T17:51:12.509Z", 7   "updatedAt": "2020-07-15T17:51:12.509Z", 8   "photo": null, 9   "nick_instagram": null,</pre>			

# Trabalhando com Autenticação

Por segurança, antes de executarmos as regras de negócio no controlador, temos sempre que verificar a autenticidade do usuário

- Agora, precisamos criar uma rota para a autenticação

Vamos fazer um POST para localhost:3000/auth com dados CORRETOS para acesso

The screenshot shows a Postman interface with the following details:

- Method:** POST
- URL:** localhost:3000/auth
- Status:** 200 OK
- Time:** 32.6 ms
- Size:** 514 B
- Created:** Just Now
- Preview (JSON):**

```
1+ {
2+   "status": 200,
3+   "message": {
4+     "user": {
5+       "uid": "3b20a42d-ab60-410c-8fb2-a85042b79a99",
6+       "name": "Guilherme Afonso Madalozzo",
7+       "nick": "@guimadalozzo",
8+       "email": "guimadalozzo@gmail.com"
9+     },
10+    "token": "ewJibHGT104LTU5T1N4TcToREctST5tWV10_cw1fcGEE5hC9hZCT5ew11nM0101"
```

# Trabalhando com Autenticação

Por segurança, antes de executarmos as regras de negócio no controlador, temos sempre que verificar a autenticidade do usuário

- Agora, precisamos criar uma rota para a autenticação

Vamos fazer um POST para localhost:3000/auth com dados ERRADOS para acesso

The screenshot shows a Postman request to `localhost:3000/auth` via a POST method. The request body is JSON containing an email and password. The response is a 200 OK status with a message indicating errors occurred during the login attempt.

Method	URL	Status	Time	Size	Created
POST	localhost:3000/auth	200 OK	11.9 ms	64 B	Just Now

**Request Body:**

```
1 - {
2   "email": "guimadalozzo@gmail.com",
3   "password": "asdads"
4 }
```

**Response Preview:**

```
1 - {
2   "status": 404,
3   "message": "Ocorreram erros para efetuar o login!"
4 }
```

# Trabalhando com Autenticação

Por segurança, antes de executarmos as regras de negócio no controlador, temos sempre que verificar a autenticidade do usuário

- Agora, precisamos criar uma rota para a autenticação

Vamos fazer um GET para localhost:3000/games para ver se bloqueia uma rota NÃO pública

The screenshot shows a Postman request for a GET operation to the endpoint `localhost:3000/games`. The request was sent at "A Minute Ago". The response status is **401 Unauthorized**, with a duration of **1.78 ms** and a size of **69 B**. The request body is JSON, containing the following data:

```
1: {  
2:   "email": "guimodalorzzo@gmail.com",  
3:   "password": "123456"  
4: }
```

The response body is also JSON, indicating that the service is only available for authenticated users:

```
1: {  
2:   "message": "Serviço disponível apenas para usuários autenticados!"  
3: }
```

# Trabalhando com Autenticação

Por segurança, antes de executarmos as regras de negócio no controlador, temos sempre que verificar a autenticidade do usuário

- Agora, precisamos criar uma rota para a autenticação

Vamos fazer um GET para localhost:3000/games para ver se bloqueia uma rota NÃO pública

The screenshot shows a browser's developer tools Network tab. A request for `localhost:3000/games` is listed. The status is **401 Unauthorized**, the time is **1.78 ms**, and the size is **69 B**. The timestamp is **A Minute Ago**. The response body preview shows the JSON object: 

```
1: {  
2:   "message": "Serviço disponível apenas para usuários  
3:   autenticados!"}
```

 To the left of the Network tab, there is a cartoon illustration of Homer Simpson with his arms raised in a triumphant or excited pose.

# Autenticação no App

# Autenticação no App

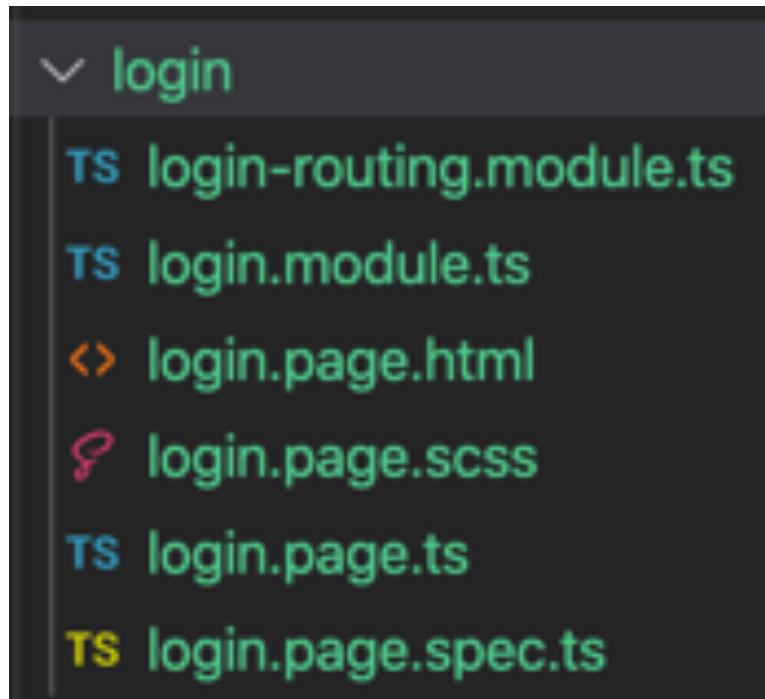
Para autenticação no App vamos precisar de uma nova página (LoginPage) e um novo serviço para consumir a API (AuthService)

```
[ionic g
? What would you like to generate? page
? Name/path of page: login
```

```
[→ PlayingScore git:(master) ✘ ionic g
? What would you like to generate? service
[?] Name/path of service: services/auth
```

# Autenticação no App

Para autenticação no App vamos precisar de uma nova página (LoginPage) e um novo serviço para consumir a API (AuthService)



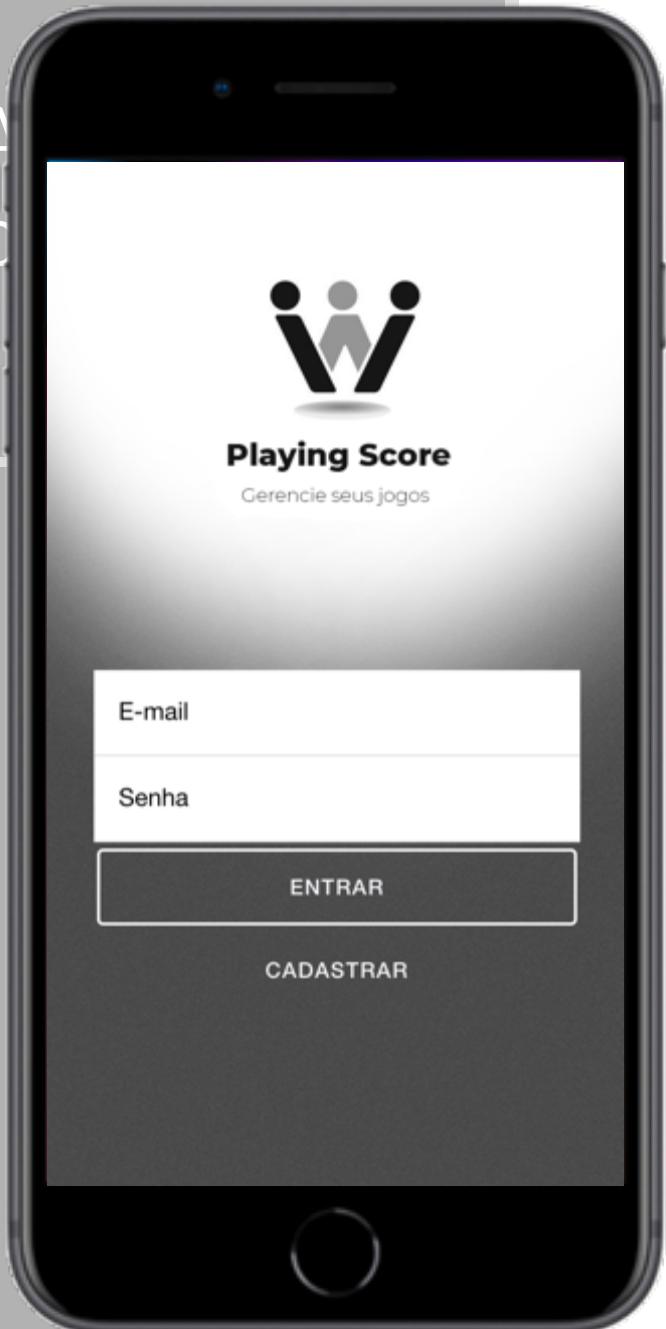
# Autenticação no App

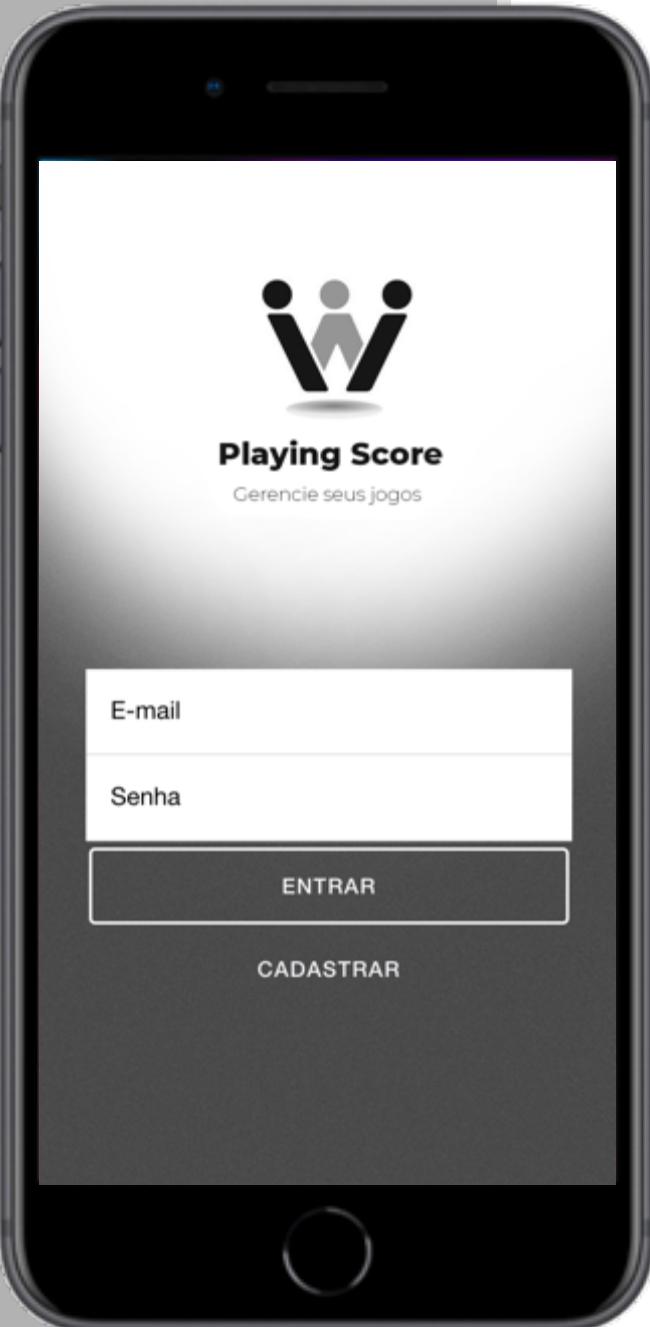
---

Para autenticação no App vamos precisar de uma nova página (LoginPage) e um novo serviço para consumir a API (AuthService)

- Começaremos projetando a tela de login

Vamos começar pelo  
login.page.ts

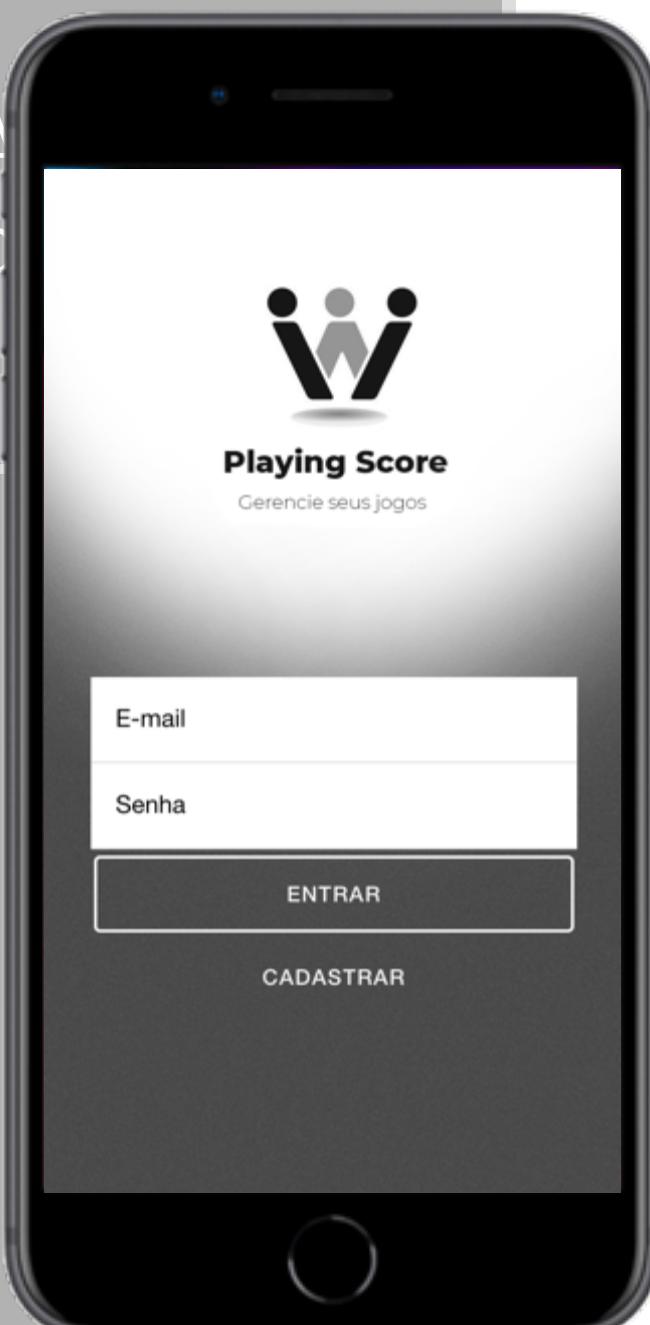




### ts login.page.ts ×

PlayingScore > src > app > login > **ts** login.page.ts > ...

```
1 import { Component, OnInit } from '@angular/core';
2 import { Router } from '@angular/router';
3 import { User } from '../models/user';
4 import { AuthService } from '../services/auth.service';
5
6 @Component({
7   selector: 'app-login',
8   templateUrl: './login.page.html',
9   styleUrls: ['./login.page.scss'],
10 })
11 export class LoginPage implements OnInit {
12
13   userForm: User = new User();
14
15   constructor(
16     private authService: AuthService,
17   ) { }
18
19   ngOnInit() { }
20
21   login() {
22     this.authService.login(this.userForm);
23   }
24 }
25 }
```



### ts login.page.ts ×

PlayingScore > src > app > login > **ts** login.page.ts > ...

```
1 import { Component, OnInit } from '@angular/core';
2 import { Router } from '@angular/router';
3 import { User } from '../models/user';
4 import { AuthService } from '../services/auth.service';
5
6 @Component({
7   selector: 'app-login',
8   templateUrl: './login.page.html',
9   styleUrls: ['./login.page.scss'],
10 })
11 export class LoginPage implements OnInit {
12
13   userForm: User = new User();
14
15   constructor(
16     private authService: AuthService,
17   ) { }
18
19   ngOnInit() { }
20
21   login() {
22     this.authService.login(this.userForm);
23   }
24 }
```

Faremos uso do AuthService para controlar possibilidade de Login

### ts login.page.ts ×

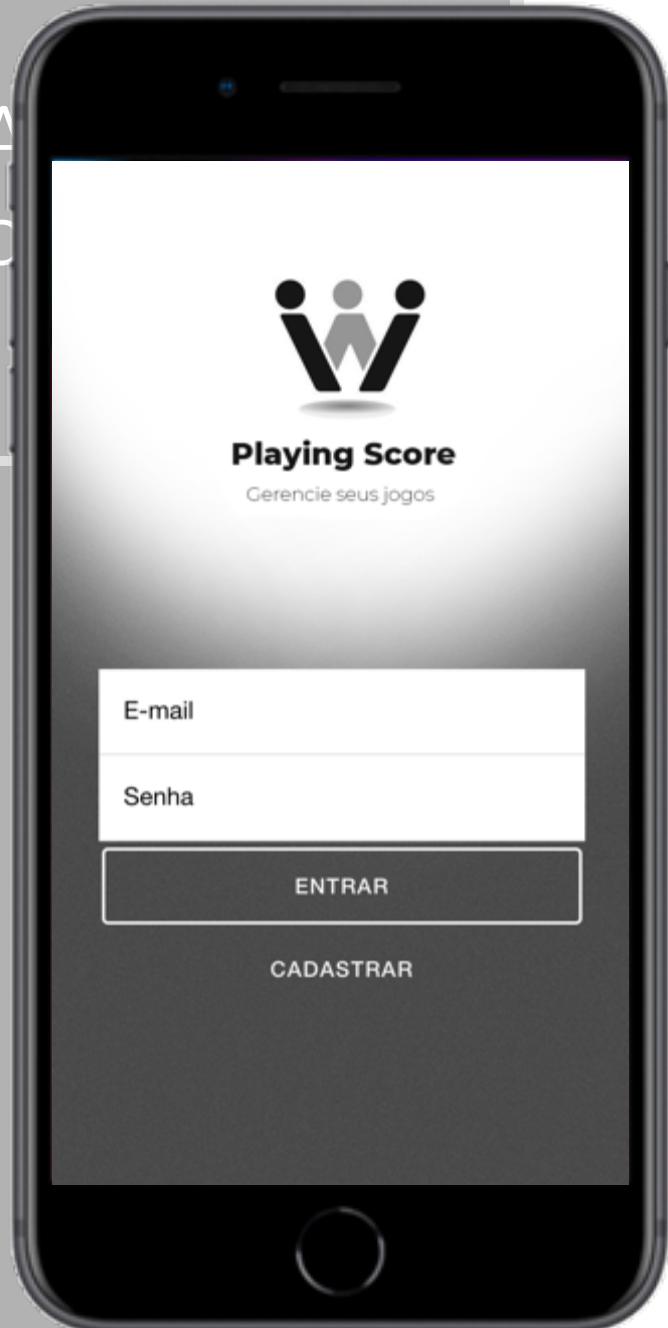
PlayingScore > src > app > login > **ts** login.page.ts > ...

```
1 import { Component, OnInit } from '@angular/core';
2 import { Router } from '@angular/router';
3 import { User } from '../models/user';
4 import { AuthService } from '../services/auth.service';
5
6 @Component({
7   selector: 'app-login',
8   templateUrl: './login.page.html',
9   styleUrls: ['./login.page.scss'],
10 })
11 export class LoginPage implements OnInit {
12
13   userForm: User = new User();
14
15   constructor(
16     private authService: AuthService,
17   ) { }
18
19   ngOnInit() { }
20
21   login() {
22     this.authService.login(this.userForm);
23   }
24
25 }
```



Desenvolvemos o método de login para passar os dados do formulário para o serviço tentar o acesso

Agora, vamos ao SCSS



Agora, vamos ao SCSS

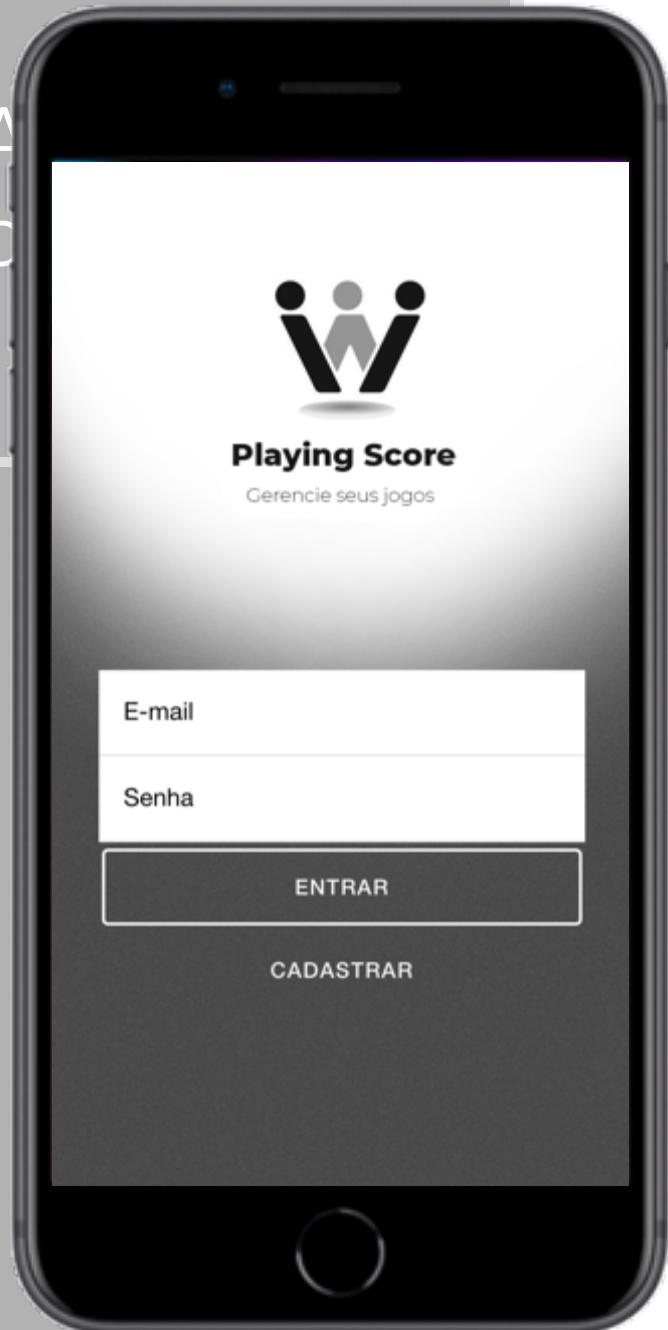


login.page.scss ×

PlayingScore > src > app > login > login.page.scss > ...

```
1  ion-content {  
2    --background: url(../../assets/img/FundoLogin.png) no-repeat center/cover fixed;  
3  }  
4  
5  .card {  
6  
7    .header {  
8      height: 300px;  
9    }  
10   .body {  
11     background-color: transparentize($color: #faf6f6, $amount: 1);  
12     padding: 30px;  
13     height: 450px !important;  
14   }  
15   .btn {  
16     height: 50px;  
17   }  
18 }  
19 }  
20 }
```

Agora, vamos ao  
HTML



Agora, vamos ao  
HTML

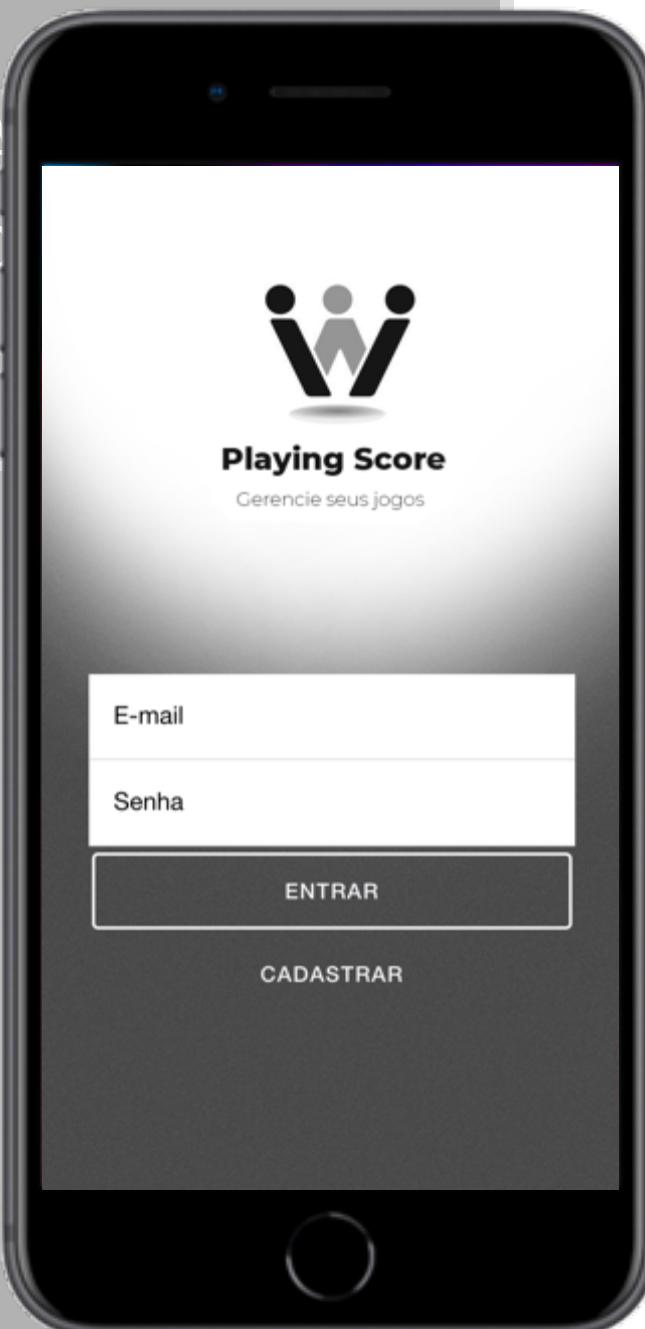


login.page.html ×

PlayingScore > src > app > login > login.page.html > ...

```
1  <ion-content>
2    <div class="card">
3      <div class="header"></div>
4
5      <div class="body">
6        <ion-toolbar>
7          <ion-item>
8            <ion-label position="floating">E-mail</ion-label>
9            <ion-input type="email" [(ngModel)]="userForm.email"></ion-input>
10           </ion-item>
11
12          <ion-item>
13            <ion-label position="floating">Senha</ion-label>
14            <ion-input type="password" [(ngModel)]="userForm.password"></ion-input>
15           </ion-item>
16        </ion-toolbar>
17
18        <ion-button class="btn" color="light" fill="outline" (click)="login()" expand="block">Entrar</ion-button>
19        <ion-button class="btn" color="light" fill="clear" expand="block">Cadastrar</ion-button>
20      </div>
21    </div>
22  </ion-content>
```

Atribuímos os valores dos campos do userForm para posterior login

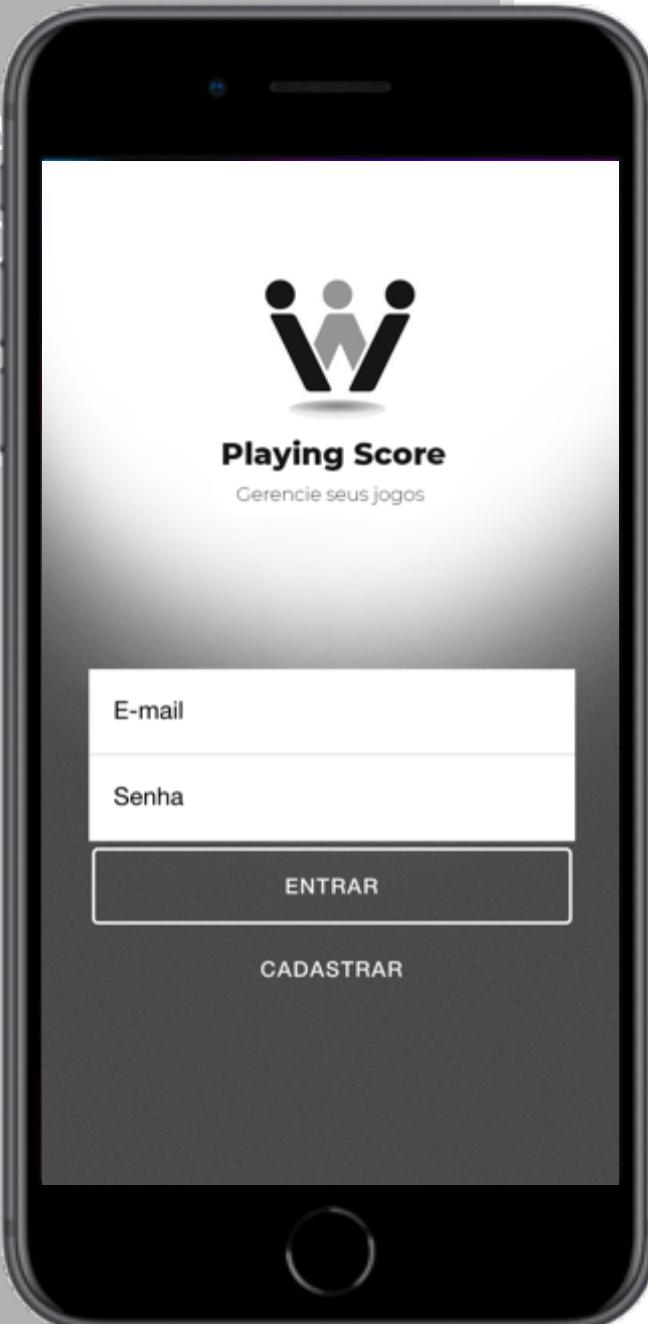


login.page.html ×

PlayingScore > src > app > login > login.page.html > ...

```
1  <ion-content>
2    <div class="card">
3      <div class="header"></div>
4
5      <div class="body">
6        <ion-toolbar>
7          <ion-item>
8            <ion-label position="floating">E-mail</ion-label>
9            <ion-input type="email" [(ngModel)]="userForm.email"></ion-input>
10         </ion-item>
11
12        <ion-item>
13          <ion-label position="floating">Senha</ion-label>
14          <ion-input type="password" [(ngModel)]="userForm.password"></ion-input>
15        </ion-item>
16      </ion-toolbar>
17
18      <ion-button class="btn" color="light" fill="outline" (click)="login()" expand="block">Entrar</ion-button>
19      <ion-button class="btn" color="light" fill="clear" expand="block">Cadastrar</ion-button>
20    </div>
21
22  </div>
23 </ion-content>
```

Definimos a ação  
CLICK do botão entrar



login.page.html ×

PlayingScore > src > app > login > login.page.html > ...

```
1  <ion-content>
2    <div class="card">
3      <div class="header"></div>
4
5      <div class="body">
6        <ion-toolbar>
7          <ion-item>
8            <ion-label position="floating">E-mail</ion-label>
9            <ion-input type="email" [(ngModel)]="userForm.email"></ion-input>
10           </ion-item>
11
12          <ion-item>
13            <ion-label position="floating">Senha</ion-label>
14            <ion-input type="password" [(ngModel)]="userForm.password"></ion-input>
15           </ion-item>
16        </ion-toolbar>
17
18        <ion-button class="btn" color="light" fill="outline" (click)="login()" expand="block">Entrar</ion-button>
19        <ion-button class="btn" color="light" fill="clear" expand="block">Cadastrar</ion-button>
20      </div>
21    </div>
22  </ion-content>
```

# Autenticação no App

Para autenticação no App vamos precisar de uma nova página (LoginPage) e um novo serviço para consumir a API (AuthService)

- Com a tela de Login modelada, vamos para a camada de negócio
- Vamos começar criando o model (class) para User

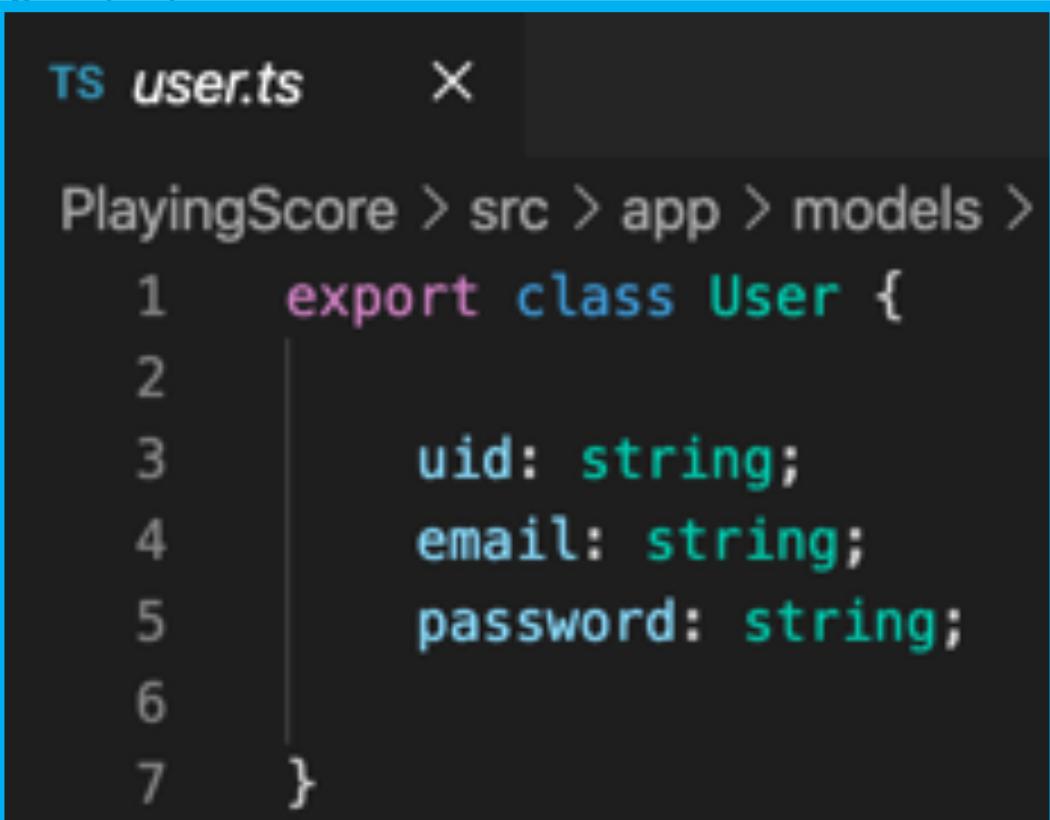
```
[→ PlayingScore git:(master) ✘ ionic g  
? What would you like to generate? class  
? Name/path of class: models/user
```

```
∨ models  
  TS game.ts  
  TS user.ts
```

# Autenticação no App

Para autenticação no App vamos precisar de uma nova página (LoginPage) e um novo serviço para consumir a API (AuthService).

- Com a tela de Login modelada
- Vamos começar criando o modelo



```
TS user.ts      X
PlayingScore > src > app > models >
1   export class User {
2
3     uid: string;
4     email: string;
5     password: string;
6
7 }
```

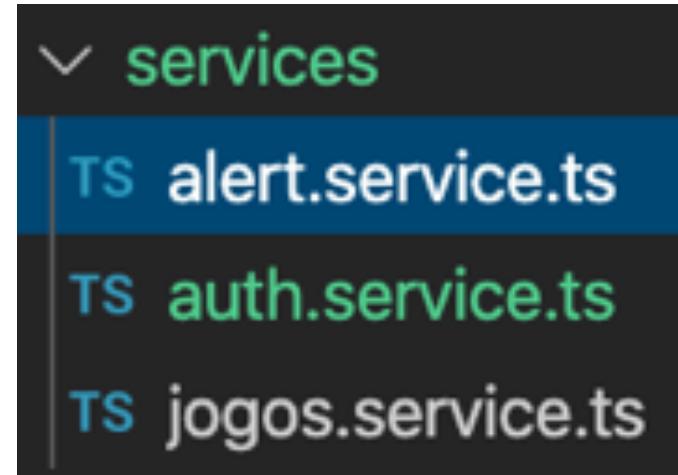
Por ora, vamos  
adicionar apenas  
esses 3 campos

# Autenticação no App

Para autenticação no App vamos precisar de uma nova página (LoginPage) e um novo serviço para consumir a API (AuthService)

- Antes de passarmos para o consumo da API (serviço), vamos criar um serviço para controle de alertas

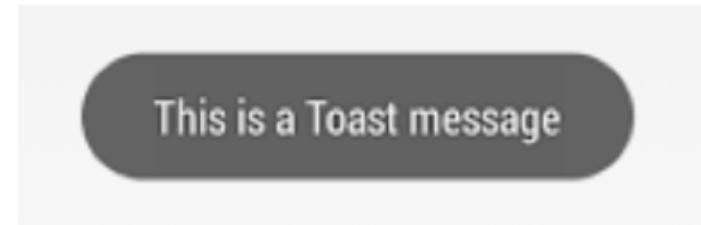
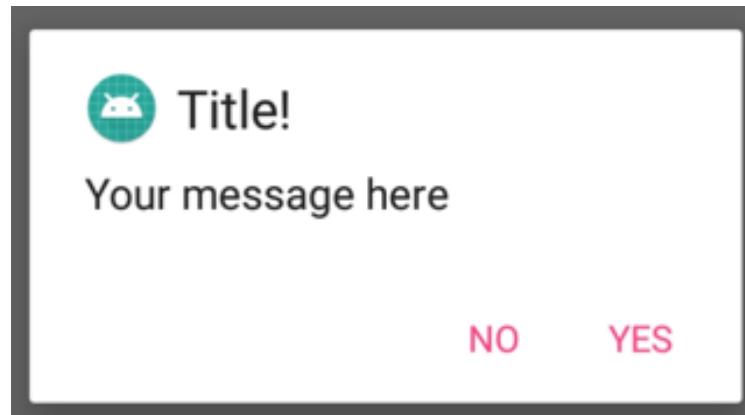
```
[→ PlayingScore git:(master) ✘ ionic g
? What would you like to generate? service
? Name/path of service: services/alert
```



# Autenticação no App

Para autenticação no App vamos precisar de uma nova página (LoginPage) e um novo serviço para consumir a API (AuthService)

- Antes de passarmos para o consumo da API (serviço), vamos criar um serviço para controle de alertas
- Este AlertService será responsável por padronizar as formas de alerta da aplicação
  - MessageDialog
  - Toast



# Autenticação no App

```
ts alert.service.ts X

PlayingScore > src > app > services > TS alert.service.ts > ...

1 import { Injectable } from '@angular/core';
2 import { AlertController, ToastController } from '@ionic/angular';
3
4 @Injectable({
5   providedIn: 'root'
6 })
7 export class AlertService {
8
9   constructor(
10     private alertController: AlertController,
11     private toastCtrl: ToastController
12   ) {}
13
14   async toast(title: string, position: any = 'top'): Promise<void> {
15     const toast = await this.toastCtrl.create({ message: title, position, duration: 3000 });
16     await toast.present();
17   }
18
19   async alert(title: string, message: string): Promise<void> {
20     const alert = await this.alertCtrl.create({
21       header: title,
22       message,
23       buttons: ['ok'],
24       backdropDismiss: false
25     });
26     await alert.present();
27   }
28
29   async confirm(title: string, message: string, callback: any): Promise<void> {
30     const alert = await this.alertCtrl.create({
31       header: title,
32       message,
33       buttons: [
34         {
35           text: 'Não', role: 'Cancel', handler: () => {
36             console.log('Confirm:Say:No');
37           }
38         },
39         { text: 'Sim', handler: () => { callback(); } }
40       ]
41     });
42     await alert.present();
43   }
44 }
```

# Autenticação

TS alert.service.ts X

PlayingScore > src > app > services > TS alert.service.ts > AlertService

```
1 import { Injectable } from '@angular/core';
2 import { AlertController, ToastController } from '@ionic/angular';
3
4 @Injectable({
5   providedIn: 'root'
6 })
7 export class AlertService {
8
9   constructor(
10     private alertCtrl: AlertController,
11     private toastCtrl: ToastController
12   ) { }
```

```
39   ]
40   });
41   });
42   await alert.present();
43 }
44 }
```

No construtor da classe criamos dois novos atributos privados, um para controle do Alert e outro para o Toas

# Autenticação no App

```
ts alert.service.ts X

PlayingScore > src > app > services > ts alert.service.ts > ...
1 import { Injectable } from '@angular/core';
2 import { AlertController, ToastController } from '@ionic/angular';
3
4 @Injectable({
5   providedIn: 'root'
6 })
7 export class AlertService {
8
9   constructor(
10     private alertController: AlertController,
11     private toastCtrl: ToastController
12   ) {}
13
14   async toast(title: string, position: any = 'top'): Promise<void> {
15     const toast = await this.toastCtrl.create({ message: title, position, duration: 3000 });
16     await toast.present();
17 }
```

```
14   async toast(title: string, position: any = 'top'): Promise<void> {
15     const toast = await this.toastCtrl.create({ message: title, position, duration: 3000 });
16     await toast.present();
17 }
```

```
26   await alert.present();
27 }
28
29   async confirm(title: string, message: string, callback: any): Promise<void> {
30     const alert = await this.alertCtrl.create({
31       header: title,
32       message,
33       buttons: [
34         {
35           text: 'Não', role: 'Cancel', handler: () => {
36             console.log('Confirm:Say:No');
37           }
38         },
39         { text: 'Sim', handler: () => { callback(); } }
40       ]
41     });
42     await alert.present();
43   }
44 }
```

Desenvolvemos o método para o padrão do Toast

# Autenticação no App

```
ts alert.service.ts X  
PlayingScore > src > app > services > ts alert.service.ts > ...  
1 import { Injectable } from '@angular/core';  
2 import { AlertController, ToastController } from '@ionic/angular';  
3  
4 @Injectable({  
5   providedIn: 'root'  
6 })  
7 export class AlertService {  
8  
9   constructor(  
10    private alertController: AlertController,  
11    private toastCtrl: ToastController  
12  ) {}  
13  
14   async toast(title: string, position: any = 'top'): Promise<void> {
```

```
19     async alert(title: string, message: string): Promise<void> {  
20       const alert = await this.alertCtrl.create({  
21         header: title,  
22         message,  
23         buttons: ['ok'],  
24         backdropDismiss: false  
25       });  
26       await alert.present();  
27     }
```

```
41     });
42     await alert.present();
43   }
44 }
```

Desenvolvemos o método para o padrão do Alert de mensagem

# Autenticação no Ann

```
ts alert.service.ts X  
PlayingScore > src > app > services > ts alert.service.ts > ...  
1 import { Injectable } from '@angular/core';  
2 import { AlertController, ToastController } from '@ionic/angular';  
3  
4 @Injectable({  
5   providedIn: 'root'  
6 })  
7 export class AlertService {
```

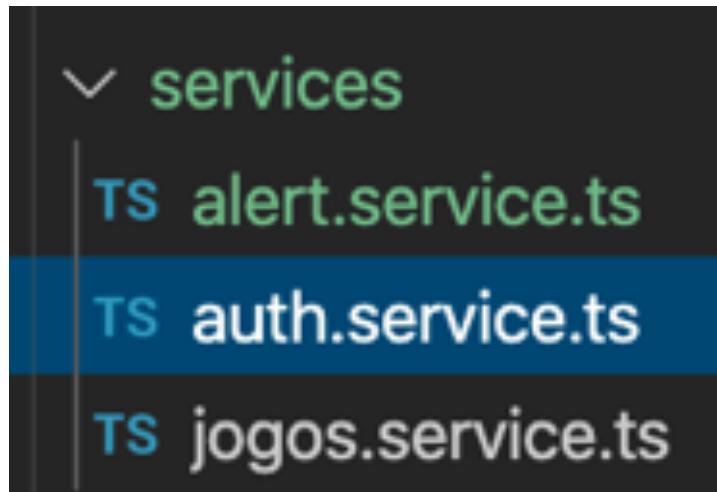
```
29   async confirm(title: string, message: string, callback: any): Promise<void> {  
30     const alert = await this.alertCtrl.create({  
31       header: title,  
32       message,  
33       buttons: [  
34         {  
35           text: 'Não', role: 'Cancel', handler: () => {  
36             console.log('Confirm:Say:No');  
37           }  
38         },  
39         { text: 'Sim', handler: () => { callback(); } }  
40       ]  
41     });  
42     await alert.present();  
43   }  
44 }
```

Desenvolvemos o método para o padrão do Alert de mensagens de confirmação

# Autenticação no App

Para autenticação no App vamos precisar de uma nova página (LoginPage) e um novo serviço para consumir a API (AuthService)

- Agora, podemos programar o AuthService para controle de login



# Autenticação no App

```
TS auth.service.ts X

PlayingScore > src > app > services > TS auth.service.ts > ...
1  import { AlertService } from './alert.service';
2  import { environment } from '../../../../../environments/environment';
3  import { Injectable } from '@angular/core';
4  import { HttpClient } from '@angular/common/http';
5  import { Router } from '@angular/router';
6  import { User } from '../models/user';
7
8  @Injectable({ providedIn: 'root' })
9  export class AuthService {
10
11    constructor(
12      private router: Router,
13      private httpClient: HttpClient,
14      public alertService: AlertService,
15    ) {}
16
17    login(user: User) {
18      this.httpClient.post(`${environment.url_api}/auth`, user)
19        .subscribe(data => {
20          let status = data['status'];
21          let message = data['message'];
22
23          if (status == 200) {
24            this.alertService.toast("Login efetuado com sucesso!");
25            localStorage.setItem("PS:USER_INFO", JSON.stringify(message));
26
27            this.router.navigate(['/tabs']);
28          }
29          else
30            this.alertService.alert("Erro", "Informações incorretas!");
31        }, error => {
32          console.log(error);
33        });
34    }
35
36    logout() {
37      localStorage.removeItem("PS:USER_INFO");
38      this.router.navigate(['/login']);
39    }
40 }
```

# Autenticação no Angular

TS auth.service.ts X

```
PlayingScore > src > app > services > TS auth.service.ts > ...
1 import { AlertService } from './alert.service';
2 import { environment } from '../../../../../environments/environment';
3 import { Injectable } from '@angular/core';
4 import { HttpClient } from '@angular/common/http';
5 import { Router } from '@angular/router';
```

TS auth.service.ts X

PlayingScore > src > app > services > TS auth.service.ts > AuthService > login

```
1 import { AlertService } from './alert.service';
2 import { environment } from '../../../../../environments/environment';
3 import { Injectable } from '@angular/core';
4 import { HttpClient } from '@angular/common/http';
5 import { Router } from '@angular/router';
6 import { User } from '../models/user';
7
8 @Injectable({ providedIn: 'root' })
9 export class AuthService {
10
11     constructor(
12         private router: Router,
13         private httpClient: HttpClient,
14         public alertService: AlertService,
15     ) { }
16
17     login() {
18         const user = {
19             email: 'user@example.com',
20             password: 'password'
21         };
22
23         this.httpClient.post('https://api.example.com/login', user)
24             .subscribe(response => {
25                 const token = response['token'];
26
27                 if (token) {
28                     localStorage.setItem('token', token);
29
30                     this.router.navigate(['/login']);
31                 }
32             });
33     }
34 }
```

Criamos a classe com o construtor e seus atributos:

**router**

Controlar a navegação ao logar

**httpClient**

Efetuar a requisição na API

**alertService**

Apresentação de retorno ao usuário

# Autenticação no App

ts auth.service.ts X

```
PlayingScore > src > app > services > ts auth.service.ts > ...
1 import { AlertService } from './alert.service';
2 import { environment } from '../../../../../environments/environment';
3 import { Injectable } from '@angular/core';
4 import { HttpClient } from '@angular/common/http';
5 import { Router } from '@angular/router';
6 import { User } from '../models/user';
```

```
17 login(user: User) {
18     this.httpClient.post(`.${environment.url_api}/auth`, user)
19         .subscribe(data => {
20             let status = data['status'];
21             let message = data['message'];
22
23             if (status == 200) {
24                 this.alertService.toast("Login efetuado com sucesso!");
25                 localStorage.setItem("PS:USER_INFO", JSON.stringify(message));
26
27                 this.router.navigate(['/tabs']);
28             }
29             else
30                 this.alertService.alert("Erro", "Informações incorretas!");
31         }, error => {
32             console.log(error);
33         });
34 }
```

38 this.router.navigate(['/login']);
39 }
40 }

Método de login() para controlar o acesso

# Autenticação no App

ts auth.service.ts X

```
PlayingScore > src > app > services > ts auth.service.ts > ...
1 import { AlertService } from './alert.service';
2 import { environment } from '../../../../../environments/environment';
3 import { Injectable } from '@angular/core';
4 import { HttpClient } from '@angular/common/http';
5 import { Router } from '@angular/router';
6 import { User } from '../models/user';
```

```
17     login(user: User) {
18         this.httpClient.post(`/${environment.url_api}/auth`, user)
19             .subscribe(data => {
20                 let status = data['status'];
21                 let message = data['message'];
22
23                 if (status == 200) {
24                     this.alertService.toast("Login efetuado com sucesso");
25                     localStorage.setItem("PS:USER_INFO", JSON.stringify(message));
26
27                     this.router.navigate(['/tabs']);
28                 }
29                 else
30                     this.alertService.alert("Erro", "Informações incorretas!");
31             }, error => {
32                 console.log(error);
33             });
34 }
```

```
38         this.router.navigate(['/login']);
39     }
40 }
```

Requisitamos um POST para a URL de acesso, passando dados do USER como parâmetro

Para efetuar a requisição, pegamos a URL da API das variáveis de ambiente do app

# Autenticação no App

ts auth.service.ts X

```
PlayingScore > src > app > services > ts auth.service.ts > ...
1 import { AlertService } from './alert.service';
2 import { environment } from '../../../../../environments/environment';
3 import { Injectable } from '@angular/core';
4 import { HttpClient } from '@angular/common/http';
5 import { Router } from '@angular/router';
6 import { User } from '../models/user';
```

```
17     login(user: User) {
18         this.httpClient.post(`.${environment.url_api}/auth`, user)
19             .subscribe(data => {
20                 let status = data['status'];
21                 if (status === 'OK') {
22                     this.alertService.success('Login efetuado com sucesso');
23                     this.router.navigate(['/tabs']);
24                 } else {
25                     this.alertService.error('Informações incorretas!');
26                 }
27             }, error => {
28                 console.log(error);
29             });
30     }
31 }
32 }
33 }
34 }
```

38 this.router.navigate(['/login']);
39 }
40 }

Requisitamos um POST para a URL de acesso, passando dados do USER como parâmetro

Para efetuar a requisição, pegamos a URL da API das variáveis de ambiente do app

# Autenticação no App

ts auth.service.ts X

```
PlayingScore > src > app > services > ts auth.service.ts > ...
1 import { AlertService } from './alert.service';
2 import { environment } from '../../../../../environments/environment';
3 import { Injectable } from '@angular/core';
4 import { HttpClient } from '@angular/common/http';
5 import { Router } from '@angular/router';
6 import { User } from '../models/user';
```

```
17     login(user: User) {
18         this.httpClient.post(`/${environment.url_api}/auth`, user)
19             .subscribe(data => {
20
21
22
23
24
25     export const environment = [
26         production: false,
27         url_api: "http://localhost:3000"
28     ];
29
30         this.alertService.alert("Erro", "Informações incorretas!");
31     }, error => {
32         console.log(error);
33     });
34 }
```

38 this.router.navigate(['/login']);
39 }
40 }

Requisitamos um POST para a URL de acesso, passando dados do USER como parâmetro

Para efetuar a requisição, pegamos a URL da API das variáveis de ambiente do app

# Autenticação no App

ts auth.service.ts X

```
PlayingScore > src > app > services > ts auth.service.ts > ...
1 import { AlertService } from './alert.service';
2 import { environment } from '../../../../../environments/environment';
3 import { Injectable } from '@angular/core';
4 import { HttpClient } from '@angular/common/http';
5 import { Router } from '@angular/router';
6 import { User } from '../models/user';
```

```
17     login(user: User) {
18         this.httpClient.post(`/${environment.url_api}/auth`, user)
19             .subscribe(data => {
20                 let status = data['status'];
21                 let message = data['message'];
22
23                 if (status == 200) {
24                     this.alertService.toast("Login efetuado com sucesso!");
25                     localStorage.setItem("PS:USER_INFO", JSON.stringify(message));
26
27                     this.router.navigate(['/tabs']);
28                 }
29                 else
30                     this.alertService.alert("Erro", "Informações incorretas!");
31             }, error => {
32                 console.log(error);
33             });
34 }
```

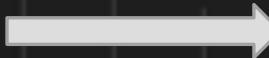
38 this.router.navigate(['/login']);
39 }
40 }

Com o dado retornado, coletamos a mensagem e o código do status recebido

# Autenticação no App

ts auth.service.ts X

```
PlayingScore > src > app > services > ts auth.service.ts > ...
1 import { AlertService } from './alert.service';
2 import { environment } from '../../../../../environments/environment';
3 import { Injectable } from '@angular/core';
4 import { HttpClient } from '@angular/common/http';
5 import { Router } from '@angular/router';
6 import { User } from '../models/user';
```

```
17 login(user: User) {
18     this.httpClient.post(`.${environment.url_api}/auth`, user)
19         .subscribe(data => {
20             let status = data['status'];
21             let message = data['message'];
22
23             if (status == 200) {
24                  this.alertService.toast("Login efetuado com sucesso!");
25                 localStorage.setItem("PS:USER_INFO", JSON.stringify(message));
26
27                 this.router.navigate(['/tabs']);
28             }
29             else
30                 this.alertService.alert("Erro", "Informações incorretas!");
31         }, error => {
32             console.log(error);
33         });
34 }
```

38 this.router.navigate(['/login']);
39 }
40 }

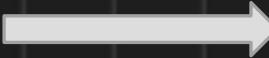
Caso o Status for 200, aciona o alerta Toast informando sucesso

# Autenticação no App

Salvamos um atributo  
(OS:USER\_INFO) na memória do dispositivo com os dados recebidos

```
ts auth.service.ts X

PlayingScore > src > app > services > ts auth.service.ts > ...
1 import { AlertService } from './alert.service';
2 import { environment } from '../../../../../environments/environment';
3 import { Injectable } from '@angular/core';
4 import { HttpClient } from '@angular/common/http';
5 import { Router } from '@angular/router';
6 import { User } from '../models/user';

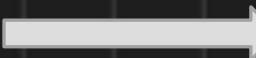
17 login(user: User) {
18     this.httpClient.post(`.${environment.url_api}/auth`, user)
19         .subscribe(data => {
20             let status = data['status'];
21             let message = data['message'];
22
23             if (status == 200) {
24                 this.alertService.toast("Login efetuado com sucesso!");
25                 localStorage.setItem("PS:USER_INFO", JSON.stringify(message));
26
27                 this.router.navigate(['/tabs']);
28             }
29             else
30                 this.alertService.alert("Erro", "Informações incorretas!");
31         }, error => {
32             console.log(error);
33         });
34     }

38         this.router.navigate(['/login']);
39     }
40 }
```

# Autenticação no App

ts auth.service.ts X

```
PlayingScore > src > app > services > ts auth.service.ts > ...
1 import { AlertService } from './alert.service';
2 import { environment } from '../../../../../environments/environment';
3 import { Injectable } from '@angular/core';
4 import { HttpClient } from '@angular/common/http';
5 import { Router } from '@angular/router';
6 import { User } from '../models/user';
```

```
17 login(user: User) {
18     this.httpClient.post(`.${environment.url_api}/auth`, user)
19         .subscribe(data => {
20             let status = data['status'];
21             let message = data['message'];
22
23             if (status == 200) {
24                 this.alertService.toast("Login efetuado com sucesso!");
25                 localStorage.setItem("PS:USER_INFO", JSON.stringify(message));
26
27                 
28             }
29             else
30                 this.alertService.alert("Erro", "Informações incorretas!");
31         }, error => {
32             console.log(error);
33         });
34 }
```

38 this.router.navigate(['/login']);
39 }
40 }

Enviamos o usuário para a navegação de tabs

# Autenticação no App

ts auth.service.ts X

```
PlayingScore > src > app > services > ts auth.service.ts > ...
1 import { AlertService } from './alert.service';
2 import { environment } from '../../../../../environments/environment';
3 import { Injectable } from '@angular/core';
4 import { HttpClient } from '@angular/common/http';
5 import { Router } from '@angular/router';
6 import { User } from '../models/user';
```

```
17 login(user: User) {
18     this.httpClient.post(`.${environment.url_api}/auth`, user)
19         .subscribe(data => {
20             let status = data['status'];
21             let message = data['message'];
22
23             if (status == 200) {
24                 this.alertService.toast("Login efetuado com sucesso!");
25                 localStorage.setItem("PS:USER_INFO", JSON.stringify(message));
26
27                 this.router.navigate(['/tabs']);
28             }
29             else
30                 this.alertService.alert("Erro", "Informações incorretas!");
31         }, error => {
32             console.log(error);
33         });
34 }
```

38 this.router.navigate(['/login']);
39 }
40 }

Caso o status não seja 200, disparamos alerta de erro

# Autenticação no App

TS auth.service.ts X

```
PlayingScore > src > app > services > TS auth.service.ts > ...
1 import { AlertService } from './alert.service';
2 import { environment } from '../../../../../environments/environment';
3 import { Injectable } from '@angular/core';
4 import { HttpClient } from '@angular/common/http';
5 import { Router } from '@angular/router';
6 import { User } from '../models/user';
7
8 @Injectable({ providedIn: 'root' })
9 export class AuthService {
10
11     constructor{
```

```
36         logout() {
37             localStorage.removeItem("PS:USER_INFO");
38             this.router.navigate(['/login']);
39         }
40     }
```

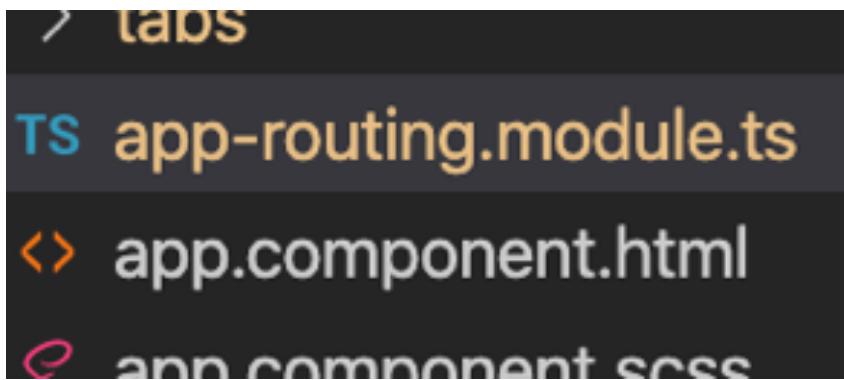
```
27             this.router.navigate(['/tabs']);
28         }
29     }
30     else
31     this.alertService.alert("Erro", "Informações incorretas!");
32 }, error => {
33     console.log(error);
34 });
35
36     logout() {
37         localStorage.removeItem("PS:USER_INFO");
38         this.router.navigate(['/login']);
39     }
40 }
```

O método de logout apaga os dados do usuário do dispositivo e redireciona-o para a tela de login

# Autenticação no App

Para autenticação no App vamos precisar de uma nova página (LoginPage) e um novo serviço para consumir a API (AuthService)

- Agora, vamos organizar as rotas da aplicação



Adicionamos a rota da página de Login e mantemos a rota tabs como a principal

TS app-routing.module.ts X

PlayingScore > src > app > TS app-routing.module.ts > [e] routes

```
1 import { NgModule } from '@angular/core';
2 import { PreloadAllModules, RouterModule, Routes } from '@angular/router';
3
4 const routes: Routes = [
5   {
6     path: 'login',
7     loadChildren: () => import('./login/login.module').then(m => m.LoginPageModule)
8   },
9   {
10     path: '',
11     children: [
12       { path: '', loadChildren: () => import('./tabs/tabs.module').then(m => m.TabsPageModule) },
13       { path: 'tabJogos', loadChildren: () => import('./jogos/jogos.module').then(m => m.JogosPageModule) },
14     ]
15   },
16   {
17     path: 'perfil',
18     loadChildren: () => import('./perfil/perfil.module').then(m => m.PerfilPageModule)
19   }
]
```

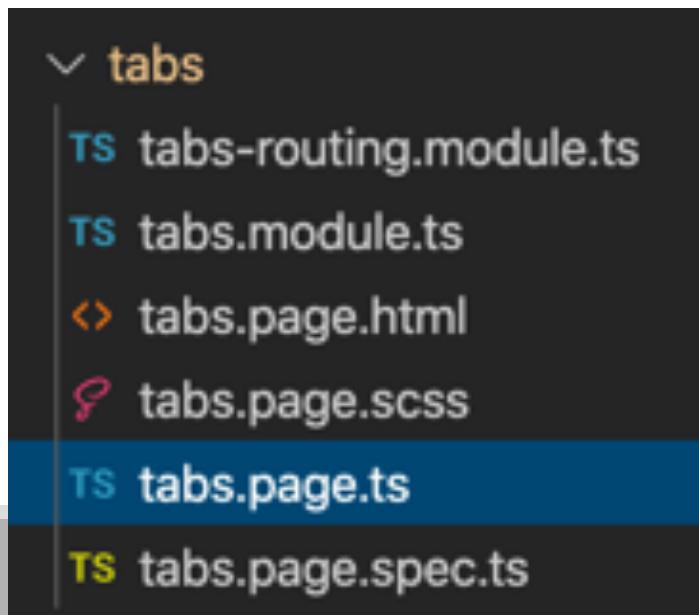
Adicionamos a rota de tabJogos como children de tabs ("")

# Autenticação no App

Para autenticação no App vamos precisar de uma nova página (LoginPage) e um novo serviço para consumir a API (AuthService)

- Sempre, a primeira página a executar é a página de Tabs
- Ela que contém a pilha de navegações permitidas

Então, nela, precisamos configurar o controle se o usuário está ou não autenticado



# Autenticação no App

TS tabs.page.ts ×

```
PlayingScore > src > app > tabs > TS tabs.page.ts > ...
1 import { Component } from '@angular/core';
2 import { NavController } from '@ionic/angular';
3
4 @Component({
5   selector: 'app-tabs',
6   templateUrl: 'tabs.page.html',
7   styleUrls: ['tabs.page.scss']
8 })
9 export class TabsPage {
10   constructor(
11     private navCtrl: NavController
12   ) { }
13
14   ngOnInit(): void {
15     if (!localStorage.getItem("PS:USER_INFO"))
16       this.navCtrl.navigateRoot('/login');
17   }
18 }
```

# Autenticação no App

TS tabs.page.ts ×

PlayingScore > src > app > tabs > TS tabs.page.ts >

```
1 import { Component } from '@angular/core';
2 import { NavController } from '@ionic/angular';
3
4 @Component({
5   selector: 'app-tabs',
6   templateUrl: 'tabs.page.html',
7   styleUrls: ['tabs.page.scss']
8 })
9 export class TabsPage {
10   constructor(
11     private navCtrl: NavController
12   ) { }
13
14   ngOnInit(): void {
15     if (!localStorage.getItem("PS:USER_INFO"))
16       this.navCtrl.navigateRoot('/login');
17   }
18 }
```

O construtor cria o controle de navegação

# Autenticação no App

TS tabs.page.ts X

PlayingScore > src > app > tabs > TS tabs.page.ts >

```
1 import { Component } from '@angular/core';
2 import { NavController } from '@ionic/angular';
3
4 @Component({
5   selector: 'app-tabs',
6   templateUrl: 'tabs.page.html',
7   styleUrls: ['tabs.page.scss']
8 })
9 export class TabsPage {
10   constructor(
11     private navCtrl: NavController
12   ) { }
13
14   ngOnInit(): void {
15     if (!localStorage.getItem("PS:USER_INFO"))
16       this.navCtrl.navigateRoot('/login');
17   }
18 }
```

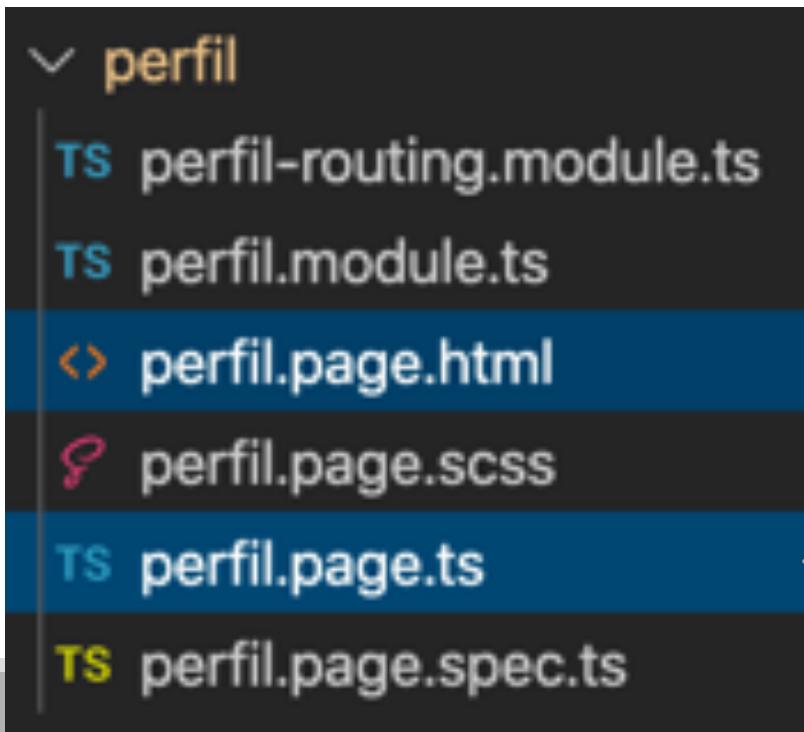
No OnInit verificamos se o usuário está autenticado, não estando, redirecionamos para a tela de login

Sabemos que o usuário não está autenticado caso não contenha registro na memória do dispositivo

# Autenticação no App

Para autenticação no App vamos precisar de uma nova página (LoginPage) e um novo serviço para consumir a API (AuthService)

- Por fim, vamos possibilitar o Logout
- Para isso, vamos alterar o TS e o HTML do PerfilPage



A screenshot of a file explorer interface showing the 'perfil' directory structure. The files listed are:

- perfil-routing.module.ts
- perfil.module.ts
- perfil.page.html
- perfil.page.scss
- perfil.page.ts
- perfil.page.spec.ts

The files 'perfil.page.html' and 'perfil.page.ts' are highlighted with a blue background, indicating they are the files being modified.

# Autenticação no App

TS perfil.page.ts ×

PlayingScore > src > app > perfil > **TS** perfil.page.ts > ...

```
1 import { Component, OnInit } from '@angular/core';
2 import { AuthService } from '../services/auth.service';
3
4 @Component({
5   selector: 'app-perfil',
6   templateUrl: './perfil.page.html',
7   styleUrls: ['./perfil.page.scss'],
8 })
9 export class PerfilPage implements OnInit {
10   constructor(private authService: AuthService) { }
11
12   ngOnInit() { }
13
14   logout(){
15     this.authService.logout();
16   }
17 }
```

# Autenticação no App

TS perfil.page.ts ×

PlayingScore > src > app > perfil > **TS** perfil.page.ts > ...

```
1 import { Component, OnInit } from '@angular/core';
2 import { AuthService } from '../services/auth.service';
3
4 @Component({
5   selector: 'app-perfil',
6   templateUrl: './perfil.page.html',
7   styleUrls: ['./perfil.page.scss'],
8 })
9 export class PerfilPage implements OnInit {
10   constructor(private authService: AuthService) { }
11
12   ngOnInit() { }
13
14   logout(){
15     this.authService.logout();
16   }
17 }
```

No construtor iniciamos o AuthService

# Autenticação no App

TS perfil.page.ts ×

PlayingScore > src > app > perfil > **TS** perfil.page.ts > ...

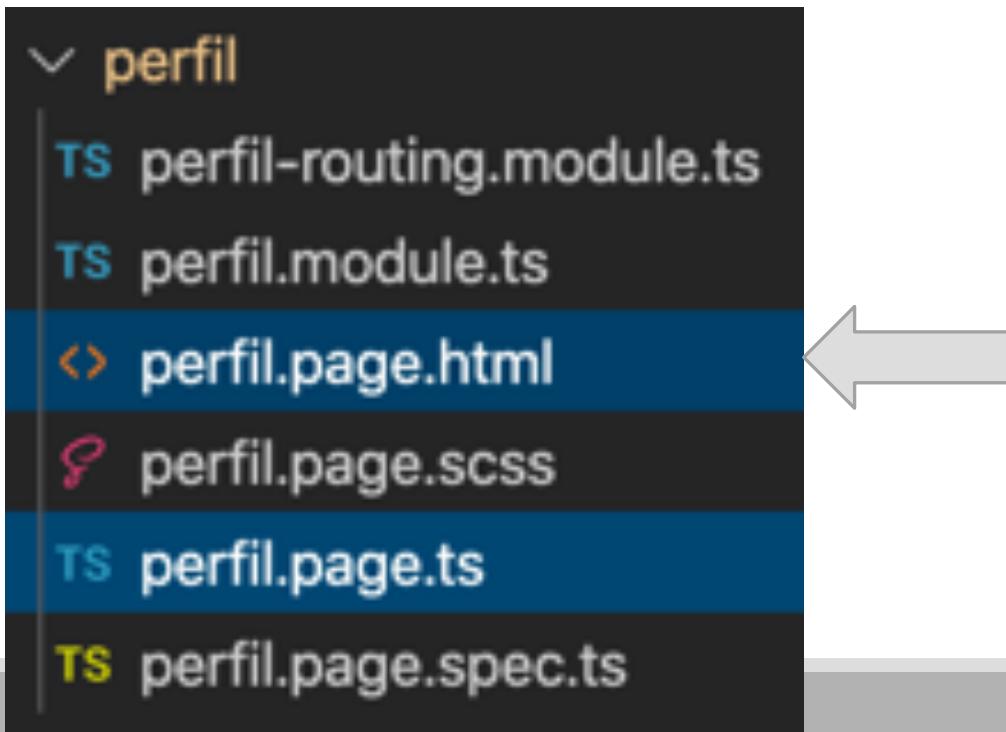
```
1 import { Component, OnInit } from '@angular/core';
2 import { AuthService } from '../services/auth.service';
3
4 @Component({
5   selector: 'app-perfil',
6   templateUrl: './perfil.page.html',
7   styleUrls: ['./perfil.page.scss'],
8 })
9 export class PerfilPage implements OnInit {
10   constructor(private authService: AuthService) { }
11
12   ngOnInit() { }
13
14   logout(){
15     this.authService.logout();
16   }
17 }
```

Criamos um método logout que executa a classe de AuthService

# Autenticação no App

Para autenticação no App vamos precisar de uma nova página (LoginPage) e um novo serviço para consumir a API (AuthService)

- Por fim, vamos possibilitar o Logout
- Para isso, vamos alterar o TS e o HTML do PerfilPage



```
perfil
  perfil-routing.module.ts
  perfil.module.ts
  perfil.page.html
  perfil.page.scss
  perfil.page.ts
  perfil.page.spec.ts
```

# Autenticação no App

No HTML vamos adicionar o CLICK para logout() no button SAIR

perfil.page.html

```
PlayingScore > src > app > perfil > perfil.page.html > ion-content > div.card
1   <ion-content fullscreen="true" slot="fixed">
2     <ion-toolbar>
3       <ion-buttons slot="start">
4         <ion-button (click)="logout()">
5           Sair
6           <ion-icon slot="start" name="exit-outline"></ion-icon>
7         </ion-button>
8       </ion-buttons>
9
10      <ion-buttons slot="end" style="margin-top: 10px;">
```

