

Documentação do Aplicativo– Ionic 7 / Angular

Etapa 1 – Criação do projeto

O desenvolvimento começou com um novo projeto Ionic 7 criado pelo terminal. Foi selecionado o template “tabs”, que gera três páginas iniciais com uma barra de navegação inferior. Esse template se mostrou ideal, pois facilita a criação de múltiplas telas e a troca entre elas.

Depois da instalação, a pasta do projeto foi acessada e verificada no navegador com o comando de execução (ionic serve), garantindo que o aplicativo base funcionava corretamente.

```
C:\Users\Joãozinho>ionic start meuApp tabs --type=angular --capacitor
? Would you like to build your app with Standalone Components on NgModules?
Standalone components are the default way to build with Angular that simplifies the way you build your app.
To learn more, visit the Angular docs:
https://angular.dev/guide/components

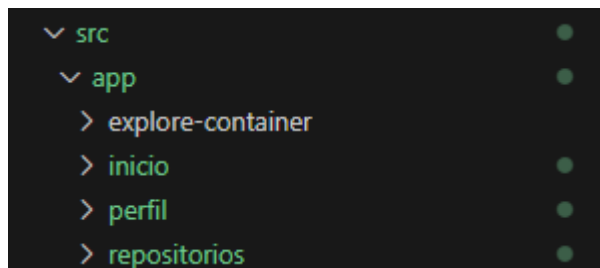
Standalone
? Preparing directory \meuApp in 815.40ms
? Downloading and extracting tabs starter in 412.26ms
? Ionic integrations enable capacitor --quiet -- meuApp io.ionic.starter
? npm.cmd i --save -E @capacitor/cli@latest
npm warn deprecated inflight@1.0.6: This module is not supported, and leaks memory. Do not use it. Check out lru-cache if you want a good and tested way to coalesce async requests by a key value, which is much more comprehensive and powerful.
npm warn deprecated rimraf@3.0.2: Rimraf versions prior to v4 are no longer supported
npm warn deprecated glob@7.2.3: Glob versions prior to v9 are no longer supported
added 1253 packages, and audited 1254 packages in 52s
250 packages are looking for funding
  run `npm fund` for details
found 0 vulnerabilities
? npm.cmd i -D -E @capacitor/cli@latest
added 62 packages, and audited 1316 packages in 10s
287 packages are looking for funding
  run `npm fund` for details
found 0 vulnerabilities
? npm.cmd i --save -E @capacitor/haptics @capacitor/app @capacitor/keyboard @capacitor/status-bar
added 4 packages, and audited 1320 packages in 7s
287 packages are looking for funding
  run `npm fund` for details
found 0 vulnerabilities
? capacitor.cmd init meuApp io.ionic.starter --web-dir www
? Creating capacitor.config.ts in C:\Users\Joãozinho\meuApp in 968.60ms
[success] capacitor.config.ts created!

Next steps:
https://capacitorjs.com/docs/getting-started#where-to-go-next
[OK] Integration capacitor added!

Installing dependencies may take several minutes.
```

Etapa 2 – Organização das abas e rotas

O template original possuía as páginas padrão **tab1**, **tab2** e **tab3**. Essas pastas foram renomeadas para **inicio**, **perfil** e **repositorios** para refletir o propósito de cada uma. As rotas internas foram atualizadas no arquivo de configuração da página de abas para apontar para os novos nomes. Dessa forma, a barra de navegação inferior passou a mostrar as opções “Apresentação”, “Meu Perfil” e “Repositórios”.



Etapa 3 – Habilitação do acesso à API

Como o aplicativo precisaria buscar dados do GitHub, o serviço de rede foi ativado. Na versão 7, o Ionic utiliza componentes standalone; portanto, a configuração do **HttpClient** foi feita diretamente no arquivo principal (main.ts). Com isso, todo o sistema passou a poder enviar requisições HTTP.

```
src > TS main.ts > ...
1 | import { enableProdMode } from '@angular/core';
2 | import { bootstrapApplication } from '@angular/platform-browser';
3 | import { RouteReuseStrategy, provideRouter } from '@angular/router';
4 | import { IonicRouteStrategy, provideIonicAngular } from '@ionic/angular/standalone';
5 |
6 | import { provideHttpClient } from '@angular/common/http';
7 |
8 | import { routes } from './app/app.routes';
9 | import { AppComponent } from './app/app.component';
10 | import { environment } from './environments/environment';
11 |
12 | if (environment.production) {
13 |   enableProdMode();
14 | }
```

Etapa 4 – Criação do serviço do GitHub

Foi criado um serviço novo chamado **GithubService**, responsável por cuidar de todas as chamadas à API pública do GitHub.

Dentro dele, definiu-se um endereço base e um método que recebe o nome do usuário e retorna os repositórios públicos correspondentes.

Esse serviço foi registrado como “root provider”, tornando-o acessível em qualquer página.

```
17 | export class GithubService {
18 |   // URL base da API do GitHub
19 |   private readonly apiUrl = 'https://api.github.com';
20 |
21 |   // Injeta o HttpClient no construtor
22 |   constructor(private http: HttpClient) { }
23 |
24 |   // Método que busca os repositórios de um usuário
25 |   getRepos(username: string): Observable<GithubRepo[]> {
26 |     return this.http.get<GithubRepo[]>(`${this.apiUrl}/users/${username}/repos`);
27 |   }
28 | }
29 |
```

Etapa 5 – Página de Apresentação

A página **Início** passou a funcionar como tela introdutória.

O título e o texto de boas-vindas foram editados e incluíram-se dois botões grandes: um direcionando para a página de perfil e outro para a página de repositórios.

Para que os botões funcionassem, foi adicionada a diretiva de navegação do Angular. O estilo centraliza o conteúdo verticalmente e dá destaque ao nome do app.



Etapas 6 – Página do Perfil

A segunda aba transformou-se em uma página pessoal com um **card de identificação**.

Dentro do card foram inseridos:

a foto do aluno, o nome completo, o código da turma, a unidade e o horário.

A imagem foi configurada para ficar circular e levemente flutuante sobre o card.

Durante os testes iniciais, a parte superior da foto era cortada pelo cabeçalho;

para corrigir, o card foi afastado para baixo e a foto ganhou um z-index mais alto, garantindo que ficasse totalmente visível.

Na parte inferior da página, adicionaram-se dois botões: um leva aos repositórios e outro retorna à página Inicial.

Esses botões receberam cores diferentes para evidenciar qual é a ação principal.

22:28

100%



localhost

1



Meu Perfil



**João Gabriel Franco
de Castro**

Código da Turma

24103084

Unidade

Unisuam CG1

Horário

08:30 - 12:00

VER MEUS REPOSITÓRIOS

VOLTAR AO INÍCIO

Apresentação

Meu Perfil

Repositórios

Etapa 7 – Página de Repositórios e integração com API

A terceira página, **Repositórios**, foi a que recebeu a maior atenção.

Nela foi implementada uma **busca dinâmica**:

um campo de pesquisa permite digitar qualquer nome de usuário do GitHub e carregar automaticamente seus repositórios públicos.

Foi adicionado um campo de entrada do tipo *searchbar* e um botão de pesquisa.

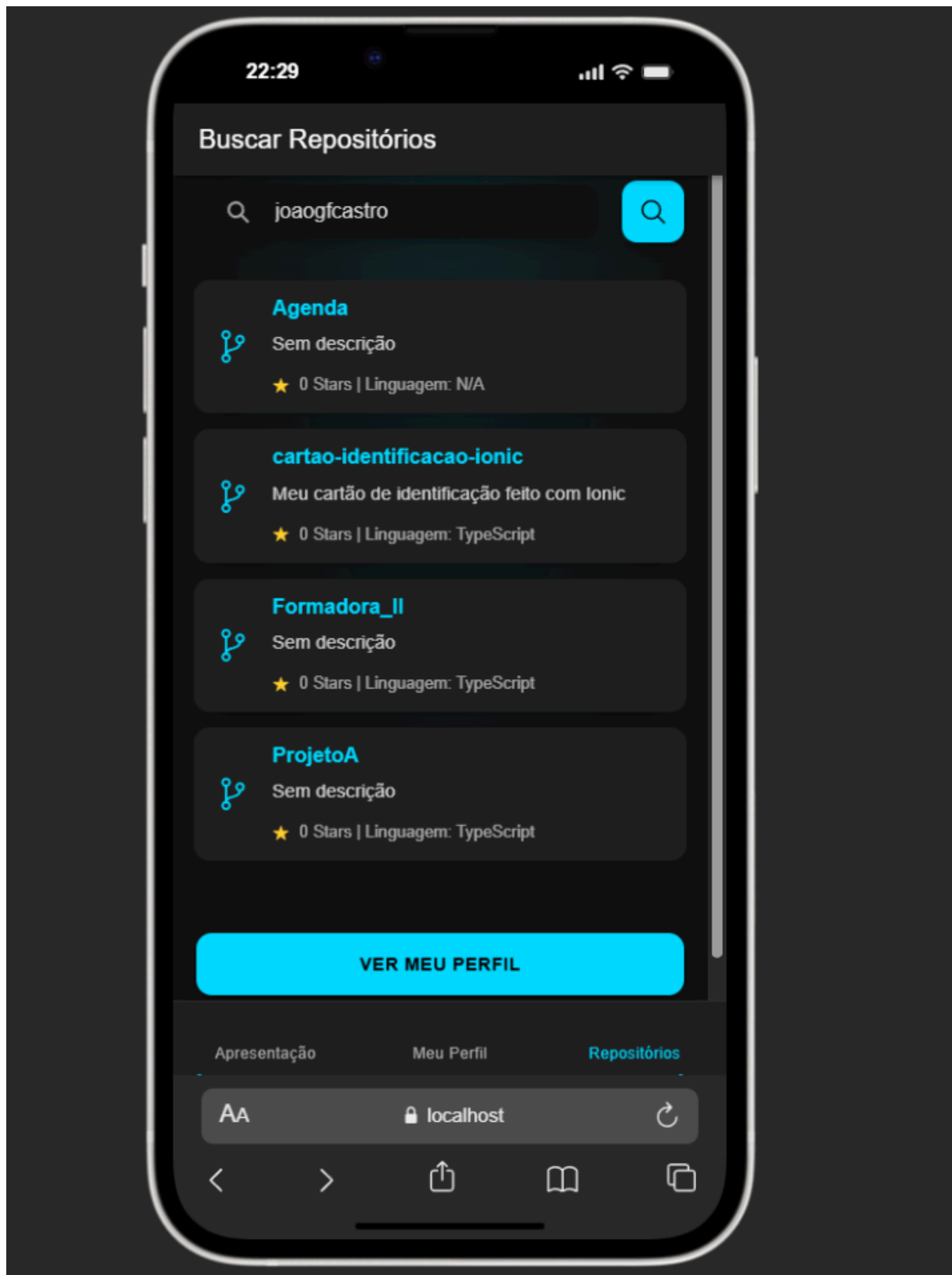
O valor digitado é capturado, e o método do serviço é chamado para obter os dados.

O aplicativo exibe um indicador de “carregando”, os resultados ou uma mensagem de erro caso o usuário não exista.

Os repositórios aparecem em uma lista estilizada como cartões, com ícone, nome, descrição, número de estrelas e linguagem de programação.

Links clicáveis abrem o repositório direto no navegador.

Na parte inferior, a navegação continua disponível com botões para o perfil e para o início.



Etapa 8 – Ajustes visuais e tema

Para unificar o visual, o aplicativo recebeu um **tema escuro global** com detalhes em ciano. O plano de fundo ganhou um leve gradiente radial que dá profundidade,

enquanto os cards e botões mantêm bordas arredondadas e sombras sutis.

Botões principais têm fundo colorido, e os de retorno usam o estilo “outline”, criando uma hierarquia visual clara.

Os espaçamentos e margens foram ajustados com *gap* para alinhar e distribuir todos os elementos de forma moderna.

Etapa 9 – Testes e validação

O projeto foi executado em ambiente de desenvolvimento usando *ionic serve*.

Todos os botões, links e requisições foram testados:

- A página inicial leva corretamente às demais.
- O card do perfil exibe a foto completa e informações formatadas.
- A aba de repositórios realiza buscas tanto para o usuário padrão quanto para outros nomes digitados.

As respostas da API foram verificadas diretamente no navegador, confirmando a conexão e a renderização dinâmica dos dados.

Conclusão

O aplicativo foi desenvolvido integralmente em Ionic 7 com Angular standalone.

Cumpre todos os requisitos do projeto:

- usa um template diferente personalizando as abas,
- possui três páginas conectadas por navegação mútua,
- integra-se a uma API externa,
- exibe informações pessoais em um card com foto,
- e apresenta uma interface moderna, escura e responsiva.

O resultado é um app funcional, visualmente equilibrado e totalmente navegável, onde o usuário pode pesquisar dados reais do GitHub e visualizar seu próprio perfil em um ambiente unificado.

Figura 9 – Aplicativo final rodando no navegador com as três abas funcionando e tema escuro aplicado.