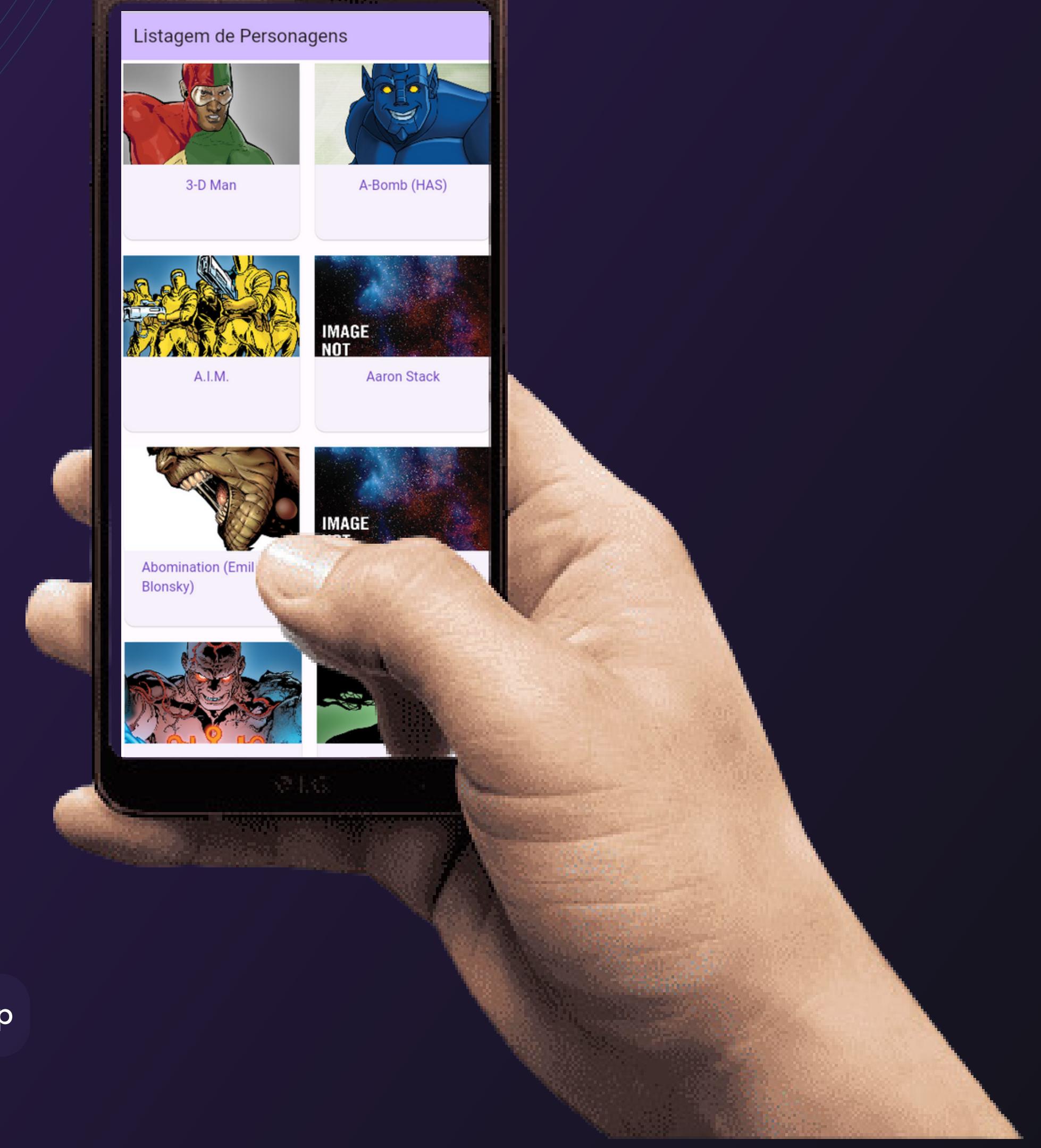
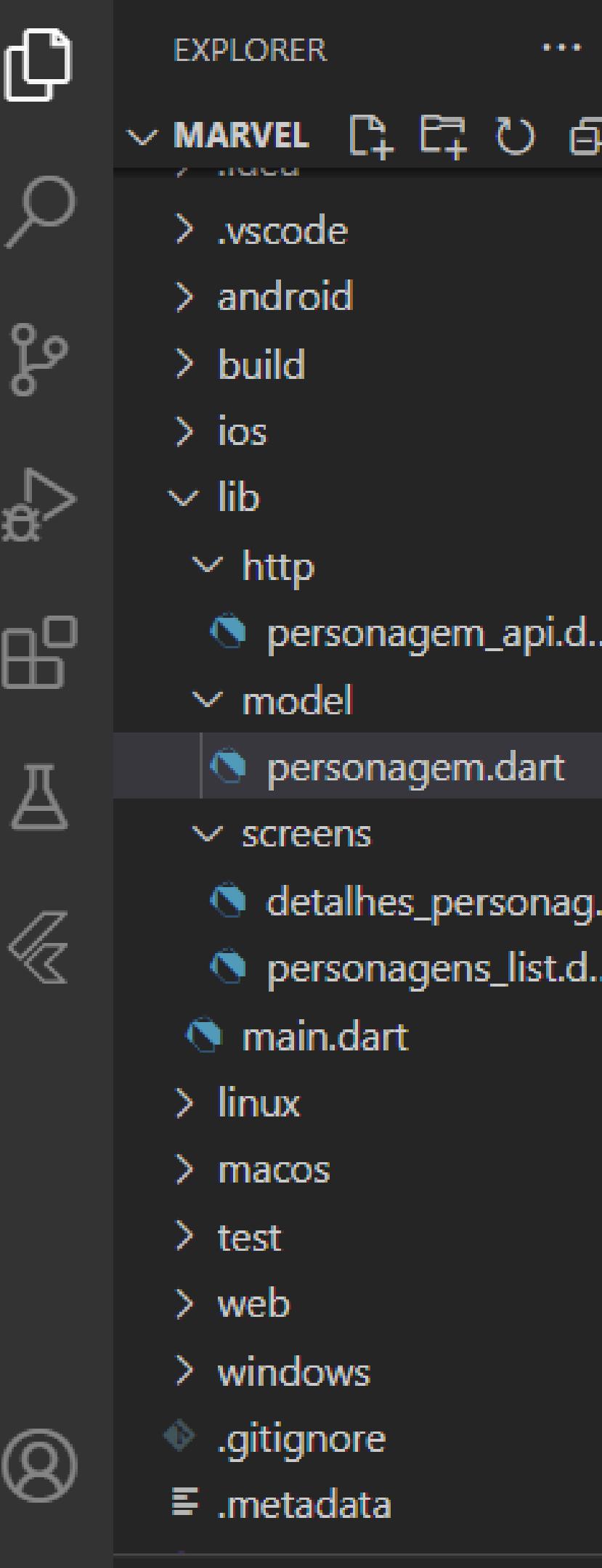


Aplicativo Marvel

Codando nosso App de Integração a API Marvel



👤 <https://github.com/MarceloSonusMobile/marvelApp>



ESTRUTURA DO PROJETO



Pasta http: Esta pasta aloca todos os arquivos responsáveis pela conexão a API.



Pasta model: Esta pasta aloca os arquivos das classes de modelagem da regra de negócio.

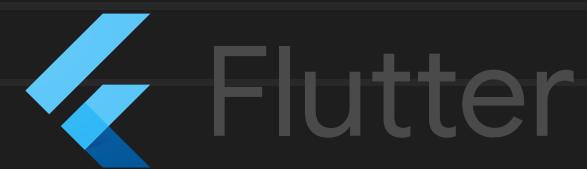


Pasta screens: Esta pasta aloca os arquivos responsáveis pela construção das interfaixes gráficas.



lib > model > personagem.dart > Personagem > Personagem.fromJson

```
1 // Definindo a classe Personagem
2 class Personagem {
3     int id;          // ID do personagem
4     String name;    // Nome do personagem
5     String avatar;  // URL do avatar do personagem
6     // Construtor da classe Personagem com valores padrão
7     Personagem({
8         this.id = 0,
9         this.name = '',
10        this.avatar = '',
11    });
12     // Factory constructor para criar uma instância de Personagem a partir de um mapa JSON
13     factory Personagem.fromJson(Map<String, dynamic> json, String key) {
14         return Personagem(
15             id: json['id'], // Atribuindo o valor do campo 'id' do JSON ao ID do personagem
16             name: json['name'], // Atribuindo o valor do campo 'name' do JSON ao nome do personagem
17             // Criando a URL do avatar combinando 'path' e 'extension' do JSON e adicionando o parâmetro 'key'
18             avatar: '${json['thumbnail']['path']}.${json['thumbnail']['extension']}?$key',
19         );
20     }
21 }
22 |
```



CLASSE
PERSONAGEMAPI

```
4 // Classe que lida com as chamadas à API dos personagens
5 class PersonagemApi {
6     final String publicKey = '0ea28726a5bc45579f222022fccbc6dc';
7     final String hash = '3b038adcc3be49ab173cc9c47713cad1';
8     final String ts = '1';
9     final String url = 'https://gateway.marvel.com/v1/public/characters?';
10
11 // Função para analisar a resposta da API e transformá-la em uma lista de objetos Personagem
12 List<Personagem> parsePersonagem(String responseBody) {
13     final parsed = jsonDecode(responseBody)[ 'data'][ 'results']
14         .cast<Map<String, dynamic>>();
15     return parsed
16         .map<Personagem>((json) =>
17             Personagem.fromJson(json, 'apikey=$publicKey&hash=$hash&ts=$ts'))
18         .toList();
19 }
20 // Função assíncrona para obter a lista de personagens da API
21 Future<List<Personagem>> getPersonagens() async {
22     final response =
23         await http.get(Uri.parse('${url}apikey=$publicKey&hash=$hash&ts=$ts'));
24     if (response.statusCode == 200) {
25         List<Personagem> personagens = parsePersonagem(response.body);
26         return personagens;
27     } else {
28         throw Exception('Falha ao obter informações da API');
29     }
30 }
31 }
```



```
30  Widget build(BuildContext context) {  
31    return Scaffold(  
32      // Definindo a barra superior da tela  
33      appBar: AppBar(  
34        backgroundColor: Theme.of(context).colorScheme.inversePrimary,  
35        title: const Text('Listagem de Personagens'),  
36      ), // AppBar  
37      // Definindo o corpo da tela  
38      body: FutureBuilder<List<Personagem>>(  
39        // Chamando a função da API para obter a lista de personagens no futuro  
40        future: api.getPersonagens(),  
41        builder: (context, snapshot) {  
42          final data = snapshot.data;  
43          // Verificando se os dados da API foram recebidos com sucesso  
44          if (snapshot.hasData) {  
45            // Construindo uma grade (grid) de personagens  
46            return GridView.builder(  
47              gridDelegate: const SliverGridDelegateWithFixedCrossAxisCount(  
48                crossAxisCount: 2,  
49                crossAxisSpacing: 10.0,  
50                mainAxisSpacing: 10.0,  
51              ), // SliverGridDelegateWithFixedCrossAxisCount  
52              itemCount: data!.length,  
53              // Construindo os cards para cada personagem  
54              itemBuilder: (context, index) {  
55                return Card(  
56                  child: Column(  
57                    children: [  
58                      // Exibindo a imagem do personagem  
59                      Image.network(  
60                        snapshot.data![index].avatar,  
61                        height: 120,  
62                      ),  
63                    ],  
64                  ),  
65                );  
66              },  
67            );  
68          }  
69        },  
70      );  
71    );  
72  );  
73};
```

CLASSE
PERSONAGENSLIST

```
62           width: double.infinity,
63             fit: BoxFit.cover,
64           ), // Image.network
65           Padding(
66             padding: const EdgeInsets.all(8.0),
67             // Adicionando um botão de texto com o nome do personagem
68             child: TextButton(
69               onPressed: () {
70                 // Navegando para a tela de detalhes do personagem
71                 navegarDetalhesPersonagem(
72                   snapshot.data![index]);
73               },
74               child: Text(
75                 snapshot.data![index].name,
76                 style: const TextStyle(fontSize: 16),
77               ), // Text // TextButton
78             ), // Padding
79           ],
80         ), // Column
81       ); // Card
82     },
83   ); // GridView.builder
84 } else {
85   // Exibindo um indicador de carregamento enquanto os dados da API são buscados
86   return const Center(
87     child: CircularProgressIndicator(),
88   ); // Center
89 }
90 }, // FutureBuilder
91 ); // Scaffold
92 }
93 }
```



```
3 // Classe que representa a tela de detalhes de um personagem
4 class DetalhesPersonagem extends StatefulWidget {
5   final Personagem personagem; // O personagem a ser exibido
6   const DetalhesPersonagem({super.key, required this.personagem});
7   @override
8   State<DetalhesPersonagem> createState() => _DetalhesPersonagemState();
9 }
10 class _DetalhesPersonagemState extends State<DetalhesPersonagem> {
11   @override
12   Widget build(BuildContext context) {
13     return Scaffold(
14       appBar: AppBar(
15         title: Text('Detalhes do Personagem ${widget.personagem.name}'),
16       ), // AppBar
17       body: Column(
18         children: [
19           Card(
20             child: Column(
21               children: [
22                 // Exibindo a imagem do personagem
23                 Image.network(
24                   widget.personagem.avatar,
25                   width: double.infinity,
26                   fit: BoxFit.cover,
27                 ), // Image.network
28                 Padding(
29                   padding: const EdgeInsets.all(8.0),
30                   // Exibindo o nome do personagem
31                   child: Text(
32                     widget.personagem.name,
33                     style: const TextStyle(fontSize: 16),
34                   ), // Text
35                 ),
36               ],
37             ),
38           ),
39         ],
40       ),
41     );
42   }
43 }
```



FLUTTER RUN ...

flutter run -d chrome
flutter run -d edge

