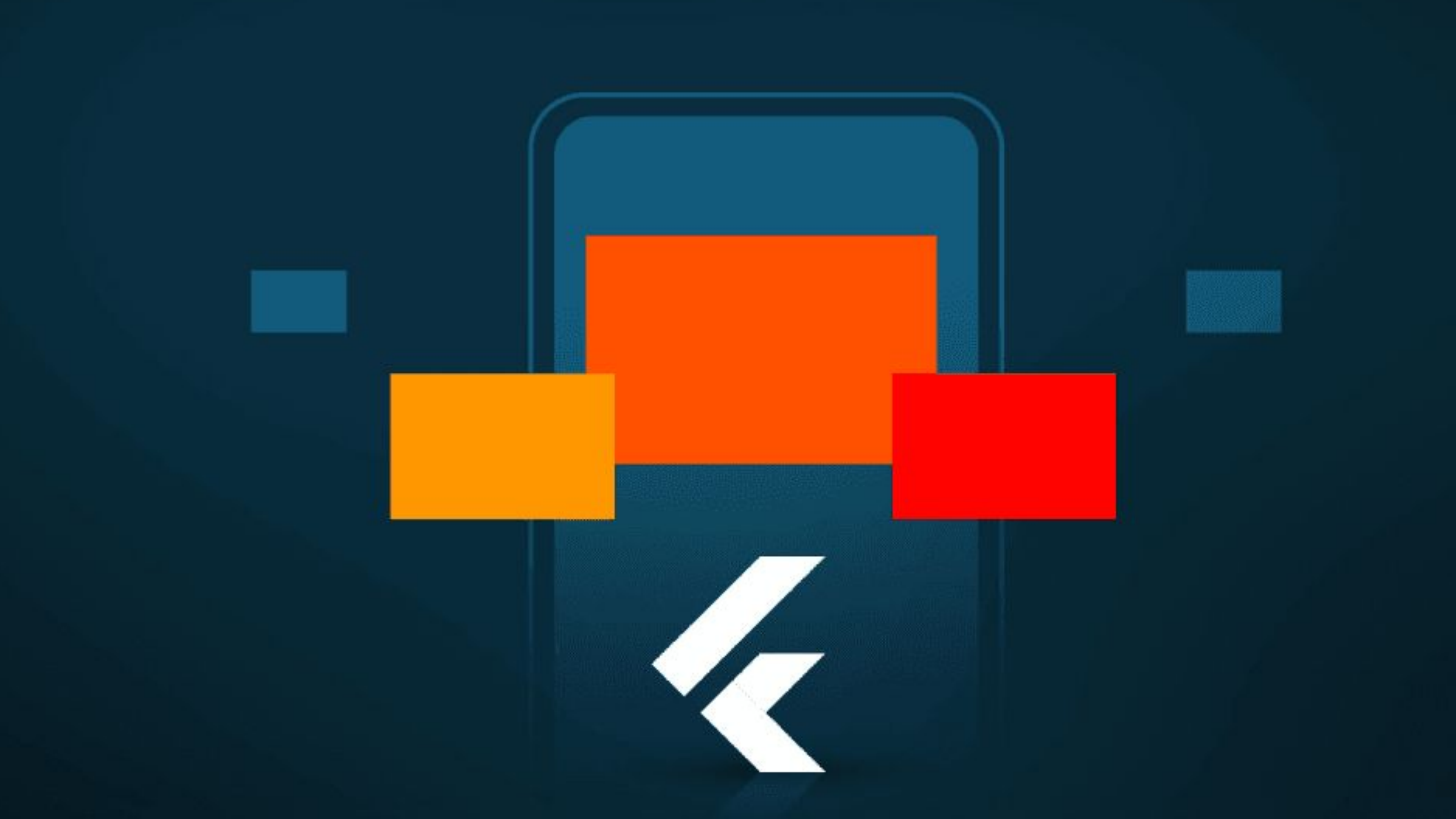


FLUTTER

TEMA: WEBSERVICES COM POKEMON

INTRODUÇÃO

POKE APP

Crie um novo projeto **flutter**:

```
flutter create --org br.com.heiderlopes poke_app
```

FLUTTER

CRIANDO A ESTRUTURA DOS WIDGETS

POKE APP

MAIN

Crie o widget **PokeApp** e execute ele na **main**.

```dart
void main() {
 runApp(const PokeApp());
}

class PokeApp extends StatelessWidget {
 const PokeApp({super.key});

 @override
 Widget build(BuildContext context) {
   return MaterialApp(
     title: 'Pokédex',
     theme: ThemeData(primarySwatch: Colors.red),
     home: const HomeScreen(),
   );
 }
}
```

POKE APP

HOME

Crie o **widget** responsável pela **Home**.

```
class HomeScreen extends StatelessWidget {
 const HomeScreen({super.key});
 @override
 Widget build(BuildContext context) {
   return Scaffold(
     appBar: AppBar(title: const Text("Pokédex"),
centerTitle: true),
     body: Center(
       child: Padding(
         padding: const EdgeInsets.all(24.0),
         child: Column(
           mainAxisAlignment: MainAxisAlignment.center,
           children: [
             // AQUI SERÃO ADICIONADOS OS CAMPONENTES DA
TELA
           ],
         ),
       ),
     ),
   );
 }
}
```

POKE APP

LISTA DE POKEMONS

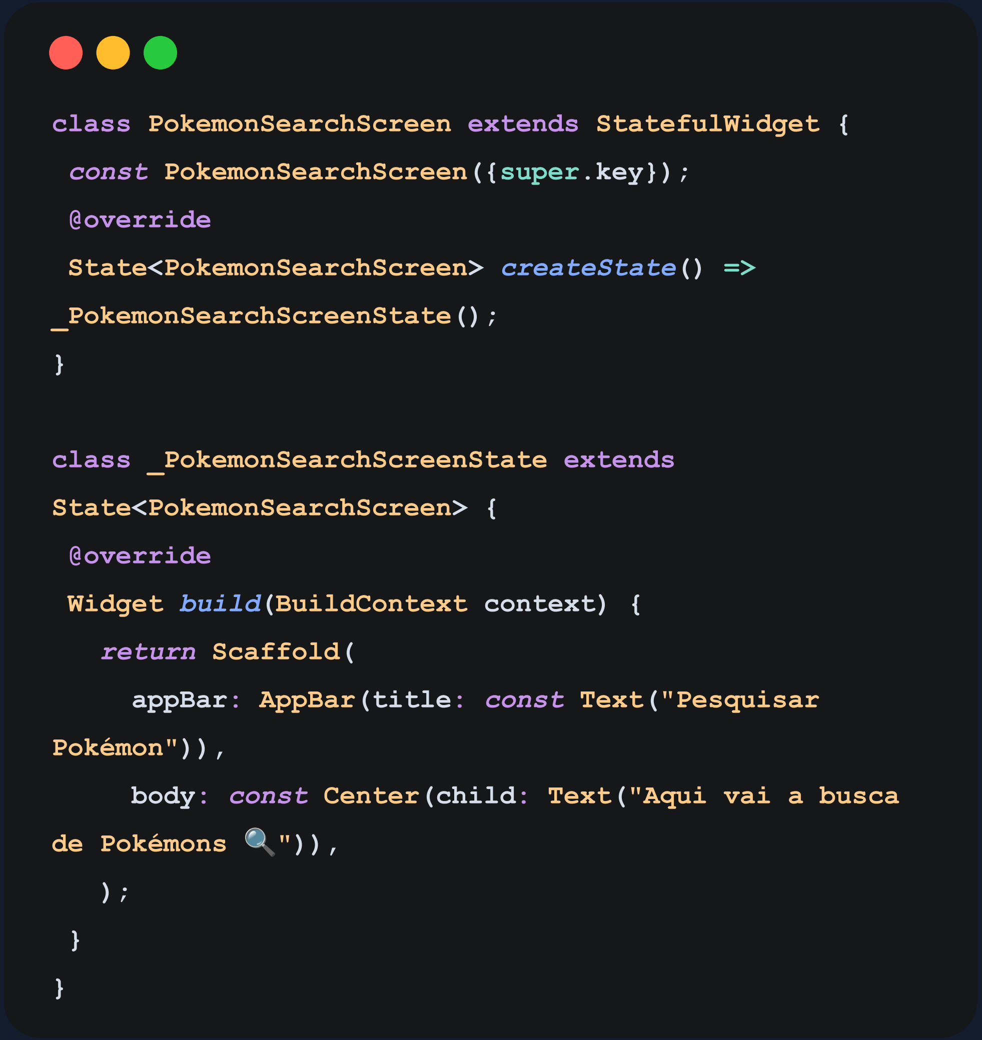
Crie o **widget** responsável por listar os Pokémons chamado **PokemonListScreen.**

```
class PokemonListScreen extends StatefulWidget {
 const PokemonListScreen({super.key});
 @override
 State<PokemonListScreen> createState() =>
_PokemonListScreenState();
}

class _PokemonListScreenState extends
State<PokemonListScreen> {
 @override
 Widget build(BuildContext context) {
   return Scaffold(
     appBar: AppBar(title: const Text("Lista de
Pokémons")),
     body: const Center(child: Text("Aqui vai a lista
de Pokémons")),
   );
 }
}
```

POKE APP

BUSCA DE POKEMONS

Crie o **widget** responsável por pesquisar os Pokémons chamado **PokemonSearchScreen.**



```
class PokemonSearchScreen extends StatefulWidget {
 const PokemonSearchScreen({super.key});
 @override
 State<PokemonSearchScreen> createState() =>
_PokemonSearchScreenState();
}

class _PokemonSearchScreenState extends
State<PokemonSearchScreen> {
 @override
 Widget build(BuildContext context) {
   return Scaffold(
     appBar: AppBar(title: const Text("Pesquisar
Pokémon")),
     body: const Center(child: Text("Aqui vai a busca
de Pokémons 🔍")),
   );
 }
}
```

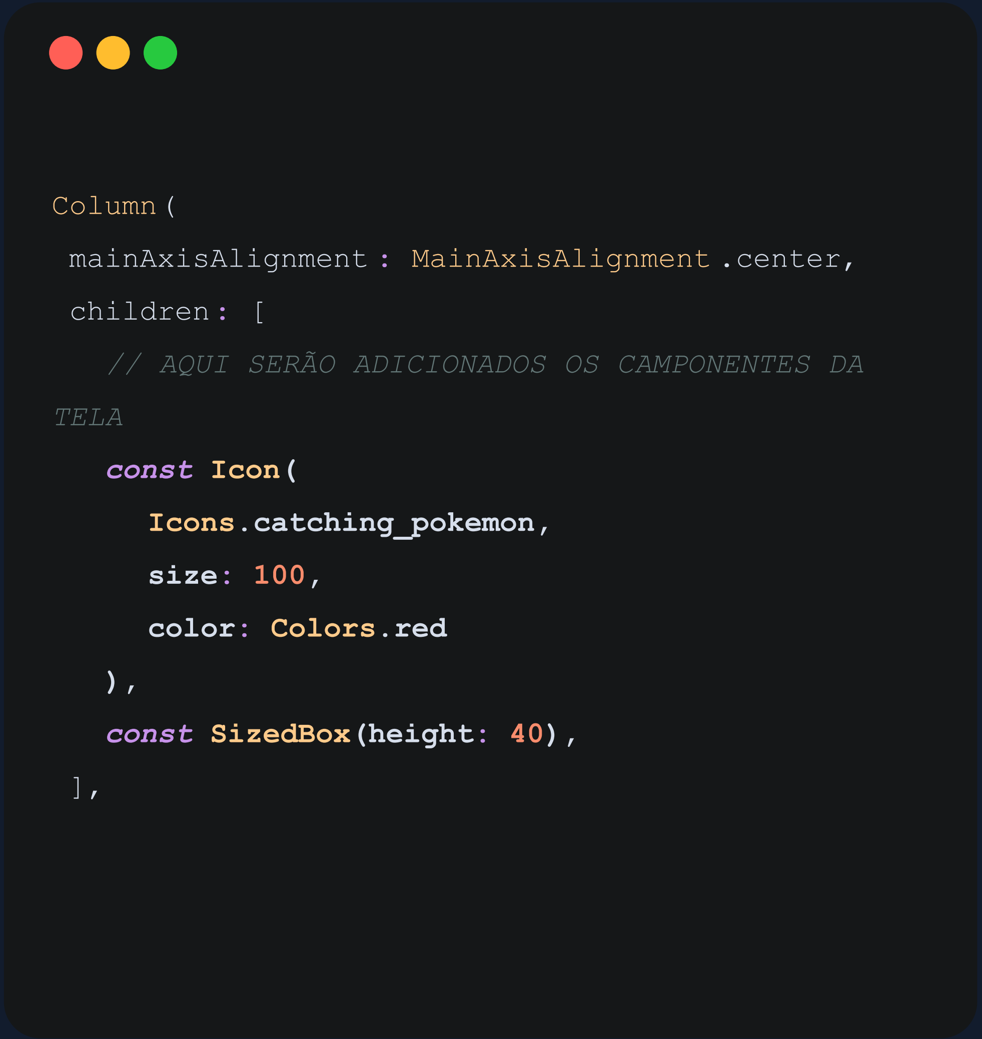
FLUTTER
CRIANDO A HOME

POKE APP

HOME

Dentro do Column criando na estrutura do aplicativo será adicionado o ícone do app.

Abaixo do ícone será adicionado um espaço para o próximo componente utilizando o **SizedBox**.



```
Column(
 mainAxisAlignment: MainAxisAlignment.center,
 children: [
    // AQUI SERÃO ADICIONADOS OS CAMPONENTES DA
TELA
    const Icon(
        Icons.catching_pokemon,
        size: 100,
        color: Colors.red
    ),
    const SizedBox(height: 40),
 ],
```

POKE APP

HOME

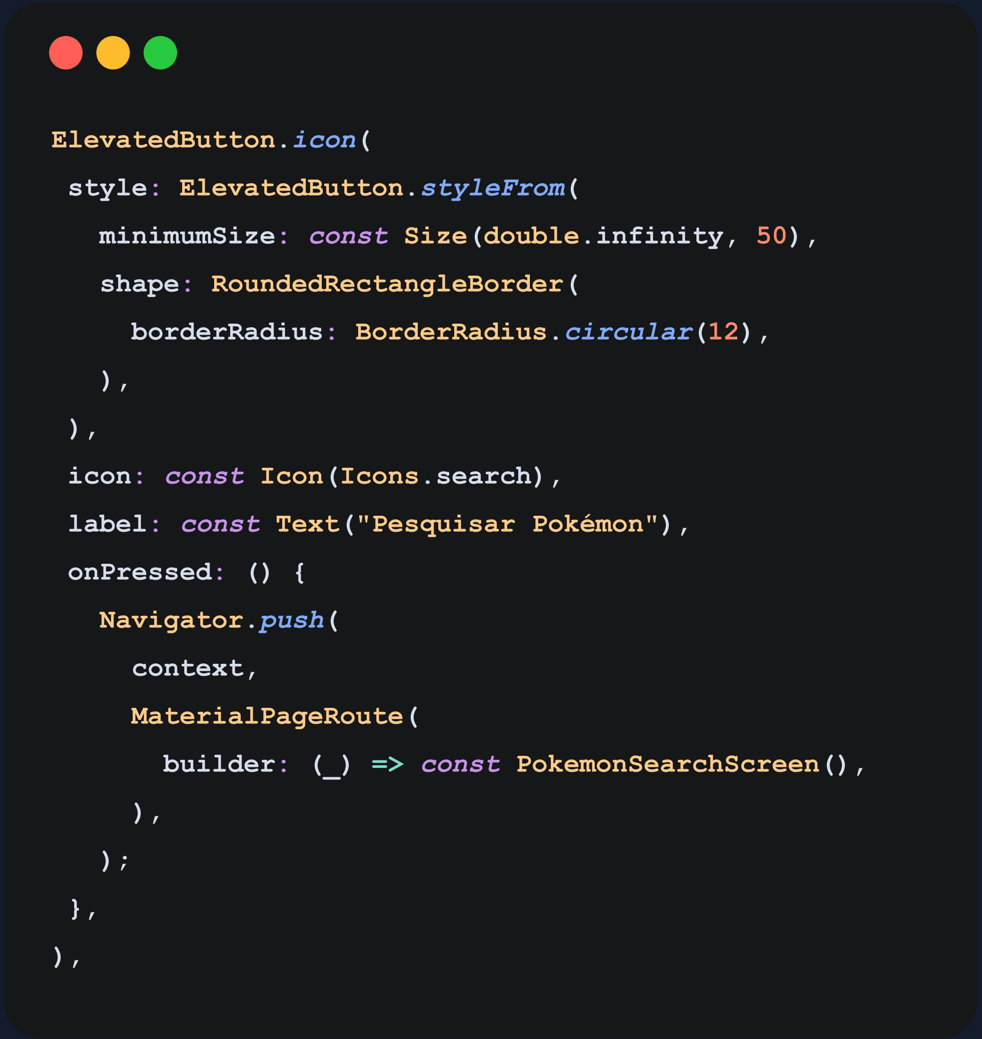
Após o ícone adicione o botão para direcionar para a lista de **Pokémons**.

```
ElevatedButton.icon(
 style: ElevatedButton.styleFrom(
   minimumSize: const Size(double.infinity, 50),
   shape: RoundedRectangleBorder(
     borderRadius: BorderRadius.circular(12),
   ),
 ),
 icon: const Icon(Icons.list),
 label: const Text("Lista de Pokémons"),
 onPressed: () {
   Navigator.push(
     context,
     MaterialPageRoute(
       builder: (_) => const PokemonListScreen(),
     ),
   );
 },
),
const SizedBox(height: 20),
```

POKE APP

HOME

Após o botão de lista de Pokémons adicione o botão para ir para a busca de Pokémons.



```
ElevatedButton.icon(
 style: ElevatedButton.styleFrom(
   minimumSize: const Size(double.infinity, 50),
   shape: RoundedRectangleBorder(
     borderRadius: BorderRadius.circular(12),
   ),
 ),
 icon: const Icon(Icons.search),
 label: const Text("Pesquisar Pokémon"),
 onPressed: () {
   Navigator.push(
     context,
     MaterialPageRoute(
       builder: (_) => const PokemonSearchScreen(),
     ),
   );
 },
),
```

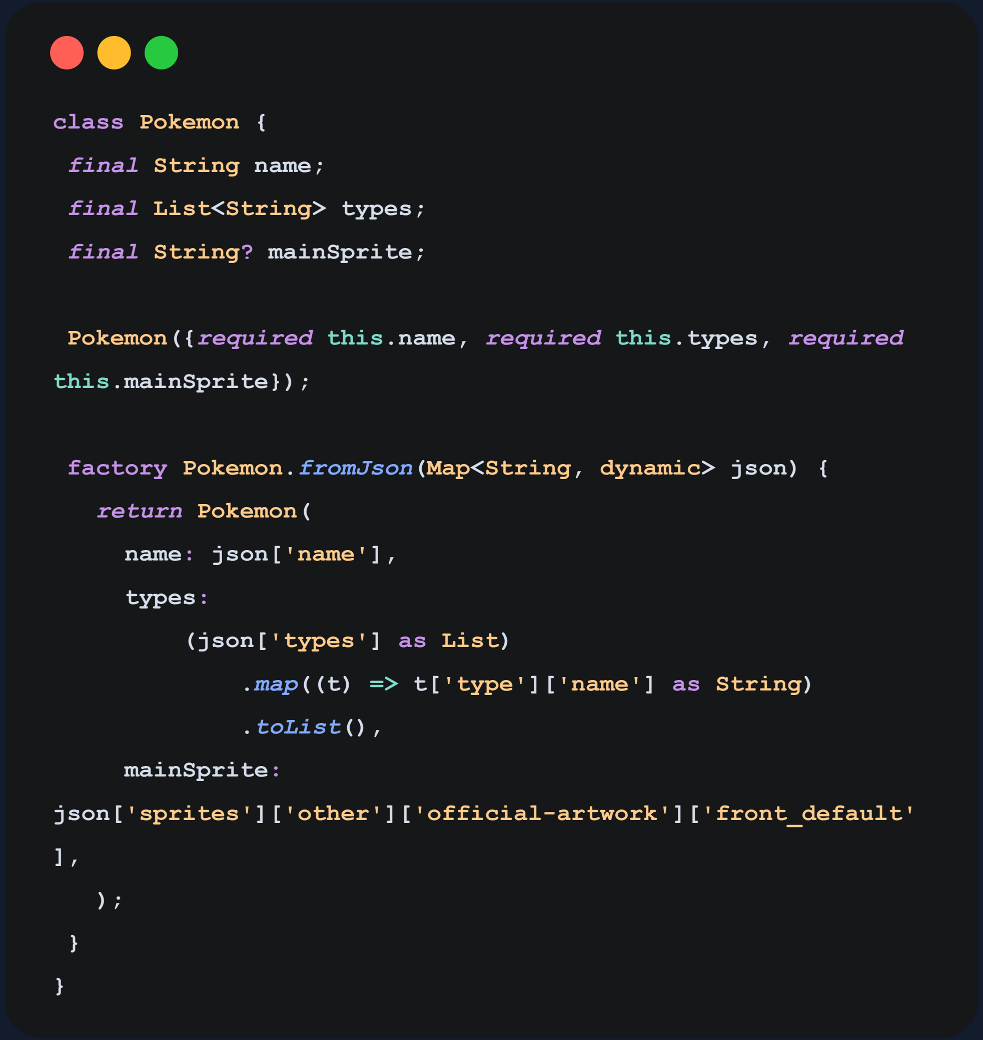
FLUTTER

CRIANDO A PESQUISA

POKE APP

SEARCH POKEMON

Crie o modelo para mapear um **Pokemon** com seus detalhes.

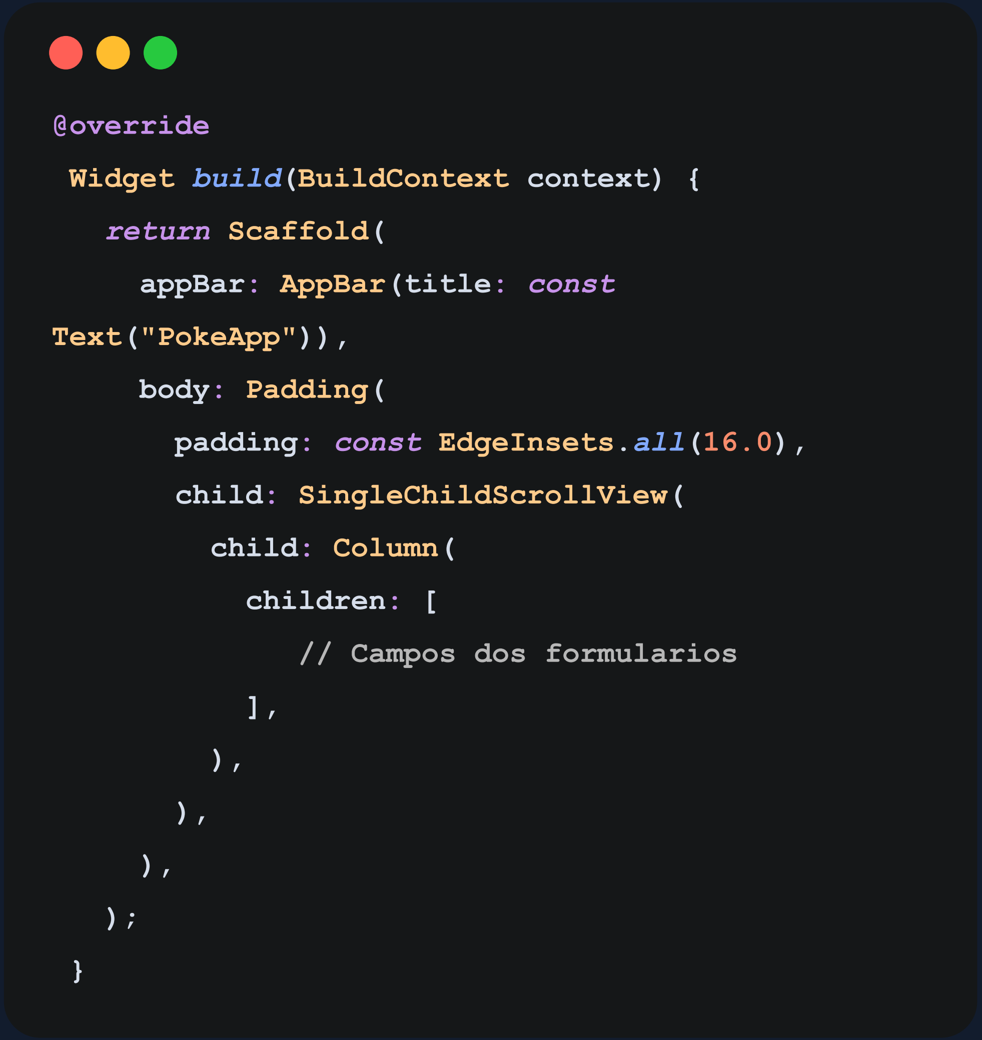


```
class Pokemon {
 final String name;
 final List<String> types;
 final String? mainSprite;

 Pokemon({required this.name, required this.types, required
this.mainSprite});

 factory Pokemon.fromJson(Map<String, dynamic> json) {
   return Pokemon(
     name: json['name'],
     types:
         (json['types'] as List)
             .map((t) => t['type']['name'] as String)
             .toList(),
     mainSprite:
json['sprites']['other']['official-artwork']['front_default'
],
   );
 }
}
```

POKE APP

SEARCH POKEMON

Crie o **build** da tela de **busca do Pokémon.**



```
@override
 Widget build(BuildContext context) {
   return Scaffold(
     appBar: AppBar(title: const
Text("PokeApp")),
     body: Padding(
       padding: const EdgeInsets.all(16.0),
       child: SingleChildScrollView(
         child: Column(
           children: [
              // Campos dos formularios
           ],
         ),
       ),
     ),
   );
 }
```

POKE APP

SEARCH POKEMON

Adicione o Controller do **Input**

```
class _PokemonSearchScreenState extends
State<PokemonSearchScreen> {
 final TextEditingController _controller =
TextEditingController();
```

POKE APP

SEARCH POKEMON

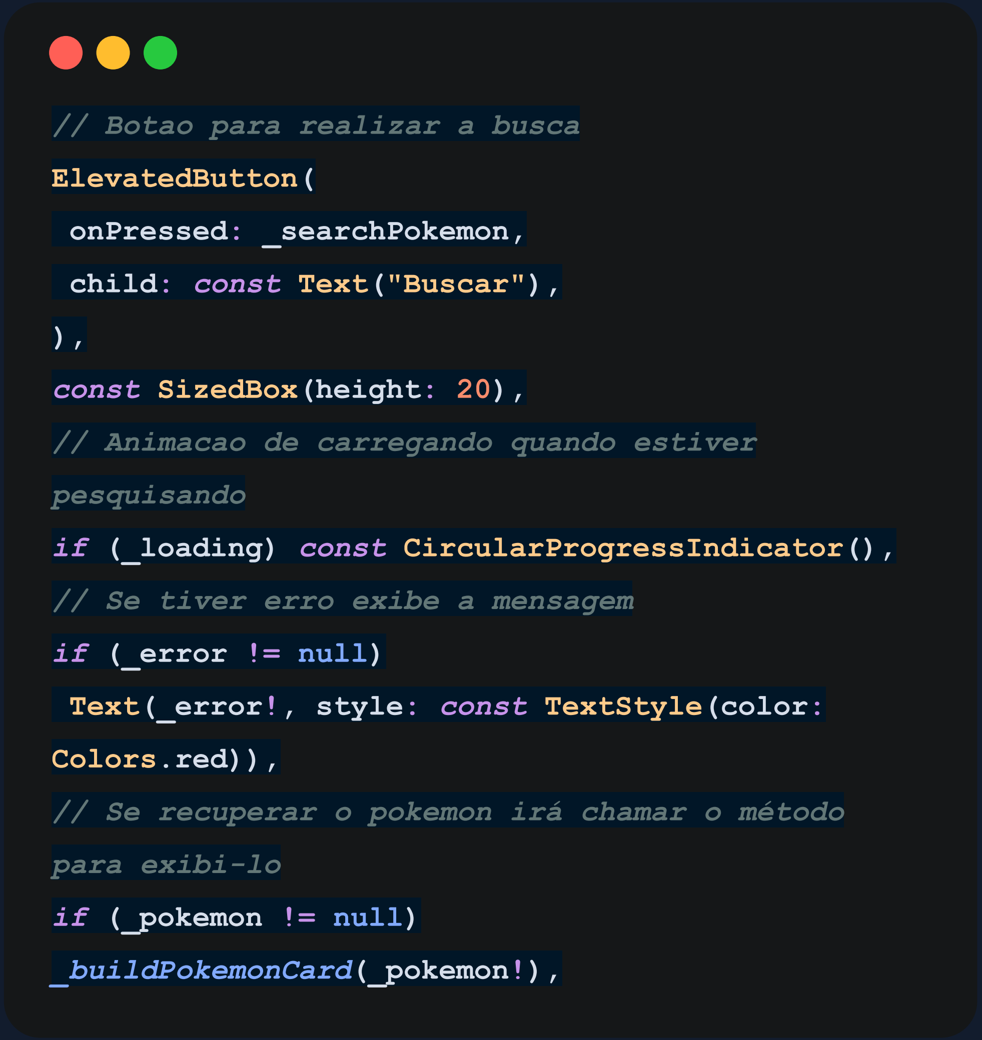
Adicione o **TextInput** dentro do **Column** criado.

```
TextField(
 controller: _controller,
 keyboardType: TextInputType.number,
 decoration: const InputDecoration(
   labelText: "Digite o número do Pokémon",
   border: OutlineInputBorder(),
 ),
),
const SizedBox(height: 12),
```

POKE APP

SEARCH POKEMON

Adicione os **widgets** que irão compor a **tela de busca.**



```
// Botao para realizar a busca
ElevatedButton(
 onPressed: _searchPokemon,
 child: const Text("Buscar"),
),
const SizedBox(height: 20),
// Animacao de carregando quando estiver
pesquisando
if (_loading) const CircularProgressIndicator(),
// Se tiver erro exibe a mensagem
if (_error != null)
 Text(_error!, style: const TextStyle(color:
Colors.red)),
// Se recuperar o pokemon irá chamar o método
para exibi-lo
if (_pokemon != null)
 _buildPokemonCard(_pokemon!),
```

POKE APP

SEARCH POKEMON

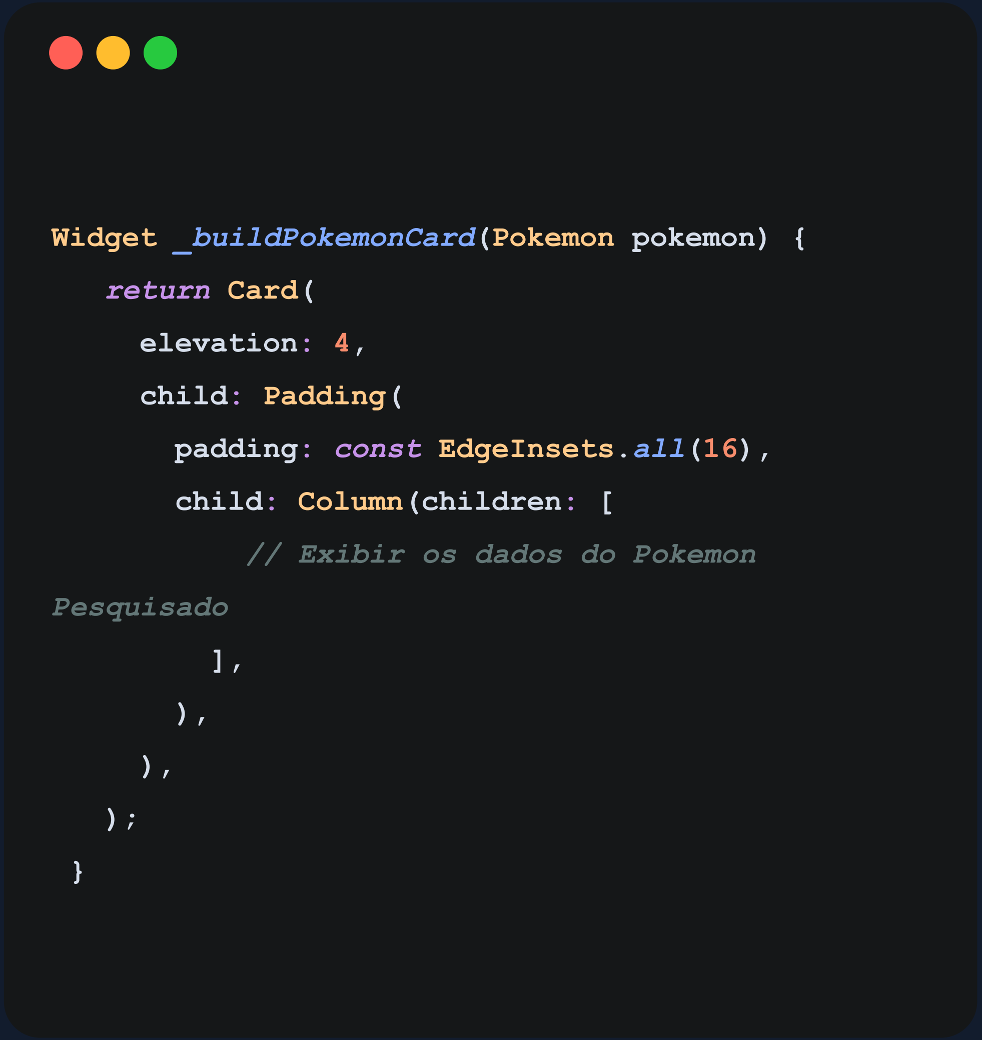
Adicione as variáveis necessárias para os nosso widgets

```dart
class _PokemonSearchScreenState extends
State<PokemonSearchScreen> {
 final TextEditingController _controller =
TextEditingController();
 // Objeto com o Pokemon pesquisado
 Pokemon? _pokemon;
 // Indica que a tela estará carregando os dados
 bool _loading = false;
 // Armazena a mensagem de erro caso aconteça
algum
 String? _error;
```

POKE APP

SEARCH POKEMON

Após o **build** crie a função para montar o widget que irá exibir o **Pokémon**.



```
Widget _buildPokemonCard(Pokemon pokemon) {
    return Card(
      elevation: 4,
      child: Padding(
        padding: const EdgeInsets.all(16),
        child: Column(children: [
            // Exibir os dados do Pokemon
Pesquisado
          ],
        ),
      ),
    );
 }
```

POKE APP

SEARCH POKEMON

Dentro do **Column** do **Card** adicione a imagem do **Pokemon** caso exista.

```
if (pokemon.mainSprite != null)
    Image.network(
        pokemon.mainSprite!,
        height: 300
    ),
```

POKE APP

SEARCH POKEMON

Após a **Imagem** adicione um **Text** para exibir o nome do **Pokémon**.

```
Text(
 pokemon.name.toUpperCase(),
 style: const TextStyle(fontSize: 22,
fontWeight: FontWeight.bold),
),
const SizedBox(height: 10),
```

POKE APP

SEARCH POKEMON

Após o nome do **Pokémon** adicione uma linha para **exibir os tipos de Pokemons**.

```
Row(
   mainAxisAlignment: MainAxisAlignment.center,
   children:
       pokemon.types
           .map(
             (type) => Padding(
               padding: const
EdgeInsets.symmetric(horizontal: 6),
               child: Chip(label: Text(type)),
             ),
           )
           .toList(),
 ),
```

POKE APP

SEARCH POKEMON

Como será realizada a busca dos dados através da API, abra o **pubspec.yaml** e adicione a lib **http**.

```yaml
dependencies:
 flutter:
   sdk: flutter
 http: ^1.2.2
```

POKE APP

SEARCH POKEMON

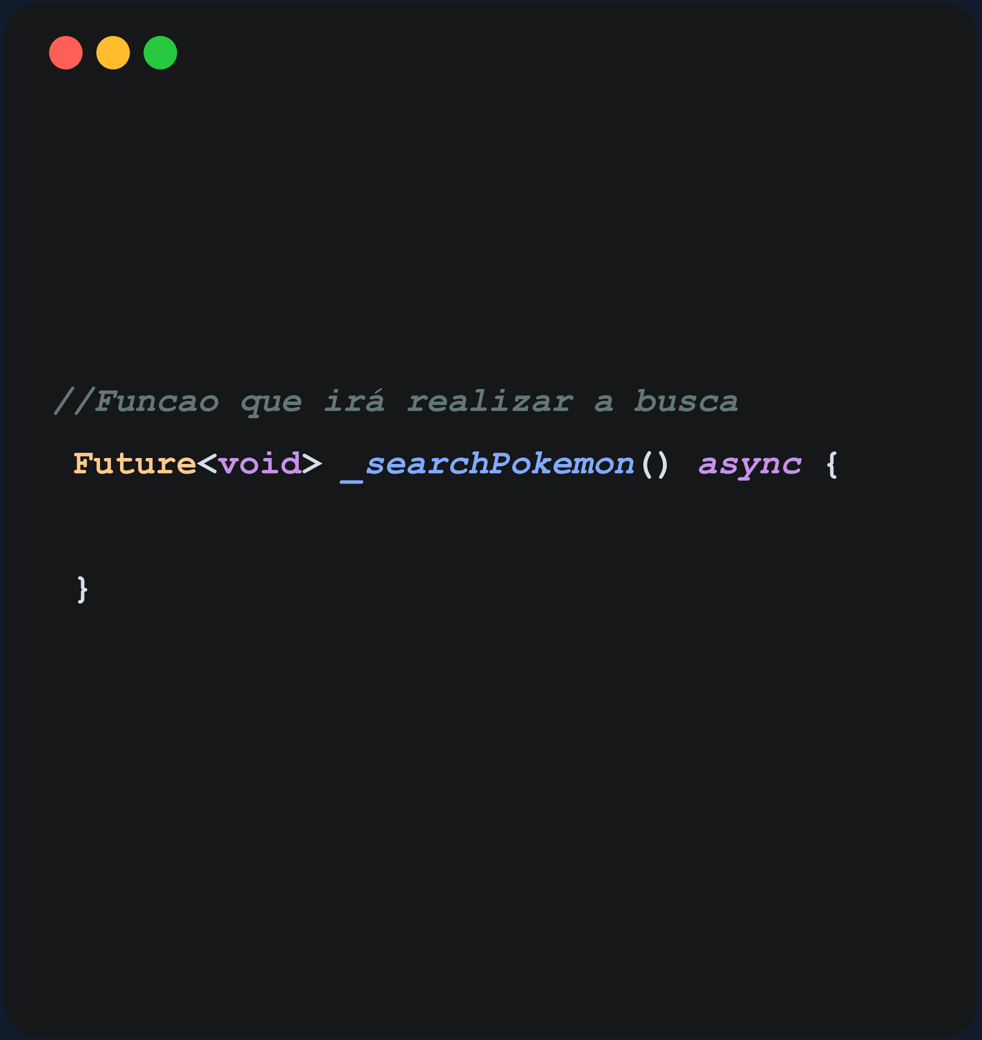
Adicione o **import** no **início** do arquivo.

```dart
import 'package:http/http.dart' as http;
```

POKE APP

SEARCH POKEMON

Crie a função que **irá buscar** o **Pokemon** através da **PokeAPI**.



```
//Funcao que irá realizar a busca
 Future<void> _searchPokemon() async {


 }
```

POKE APP

SEARCH POKEMON

Adicione o seguinte código ao método de **_searchPokemon**.

```
// Valida se contém um ID valido
   final id = int.tryParse(_controller.text);
   if (id == null) {
     setState(() => _error = "Digite um número
válido!");
     return;
   }

   // Altera o estado
   setState(() {
     _loading = true;
     _error = null;
     _pokemon = null;
   });
```

POKE APP

SEARCH POKEMON

Adicione o seguinte código ao método de **_searchPokemon**.

```
try {
 final response = await http.get(
   Uri.parse("https://pokeapi.co/api/v2/pokemon/$id"),
 );
 if (response.statusCode == 200) {
   final data = json.decode(response.body);
   setState(() => _pokemon = Pokemon.fromJson(data));
 } else {
   setState(() => _error = "Pokémon não encontrado!");
 }
} catch (e) {
 setState(() => _error = "Erro de conexão!");
} finally {
 setState(() => _loading = false);
}
```

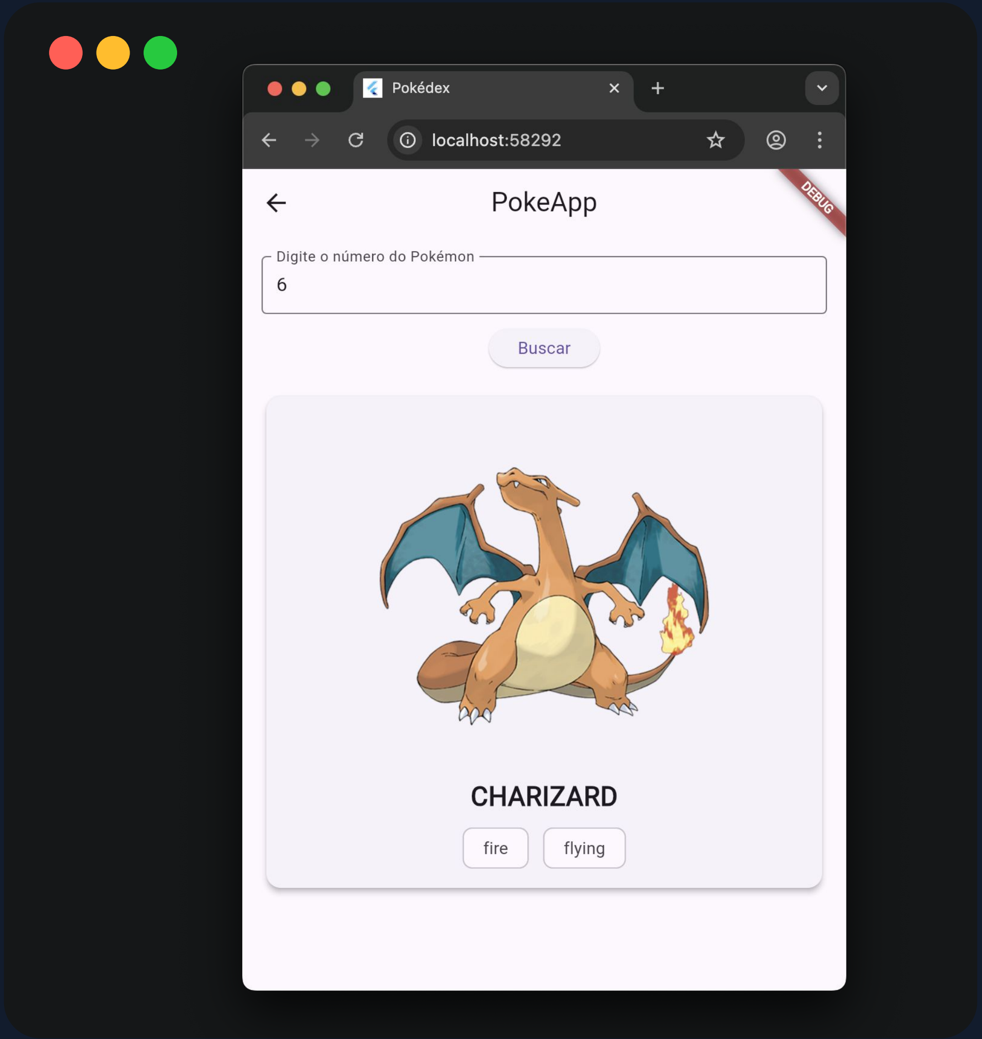
POKE APP

RODANDO O APP

Rode o aplicativo:

```
flutter run
```

Faça uma busca de um Pokémon.



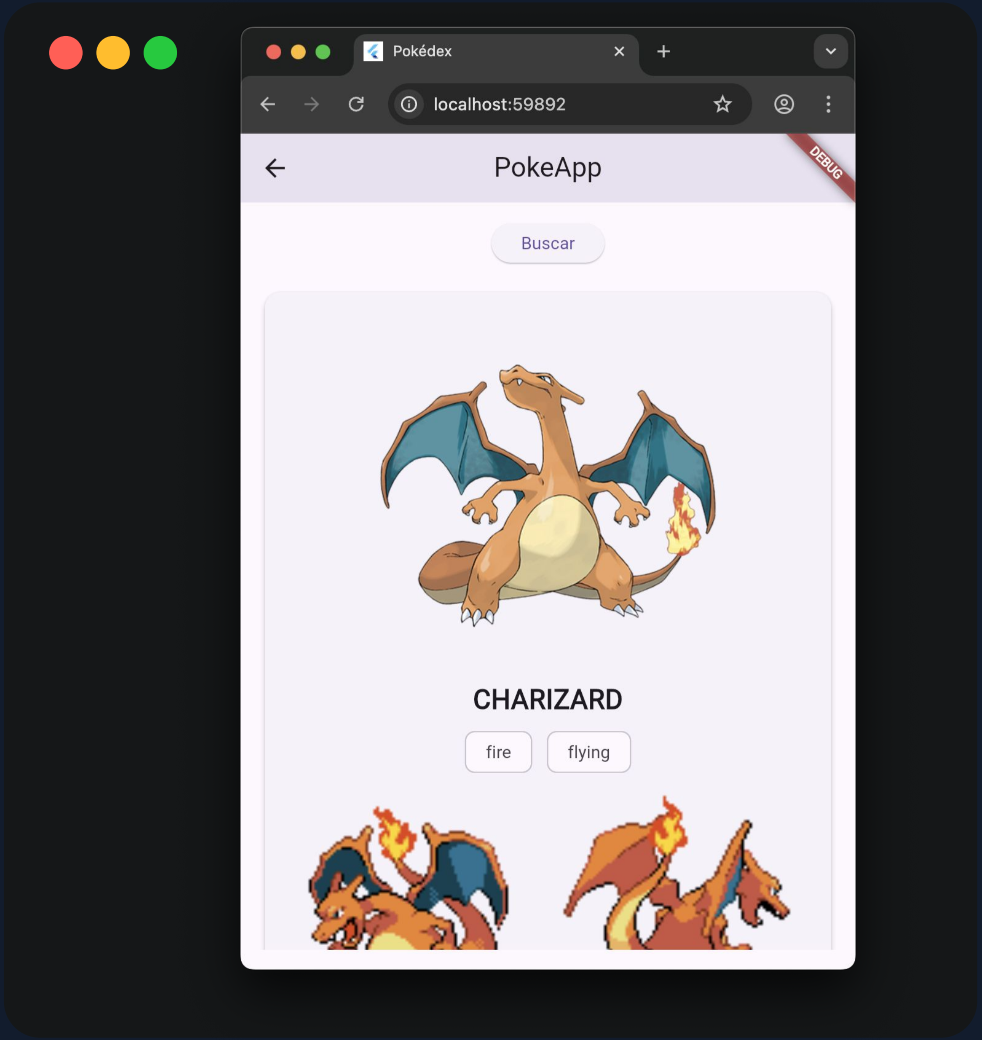
FLUTTER
EXERCÍCIO

POKE APP

EXERCÍCIO

Adicione abaixo dos tipos de **Pokémons** um **Grid** para exibir as **imagens:**

- **back_default**
- **back_shiny**
- **front_default**
- **front_shiny**



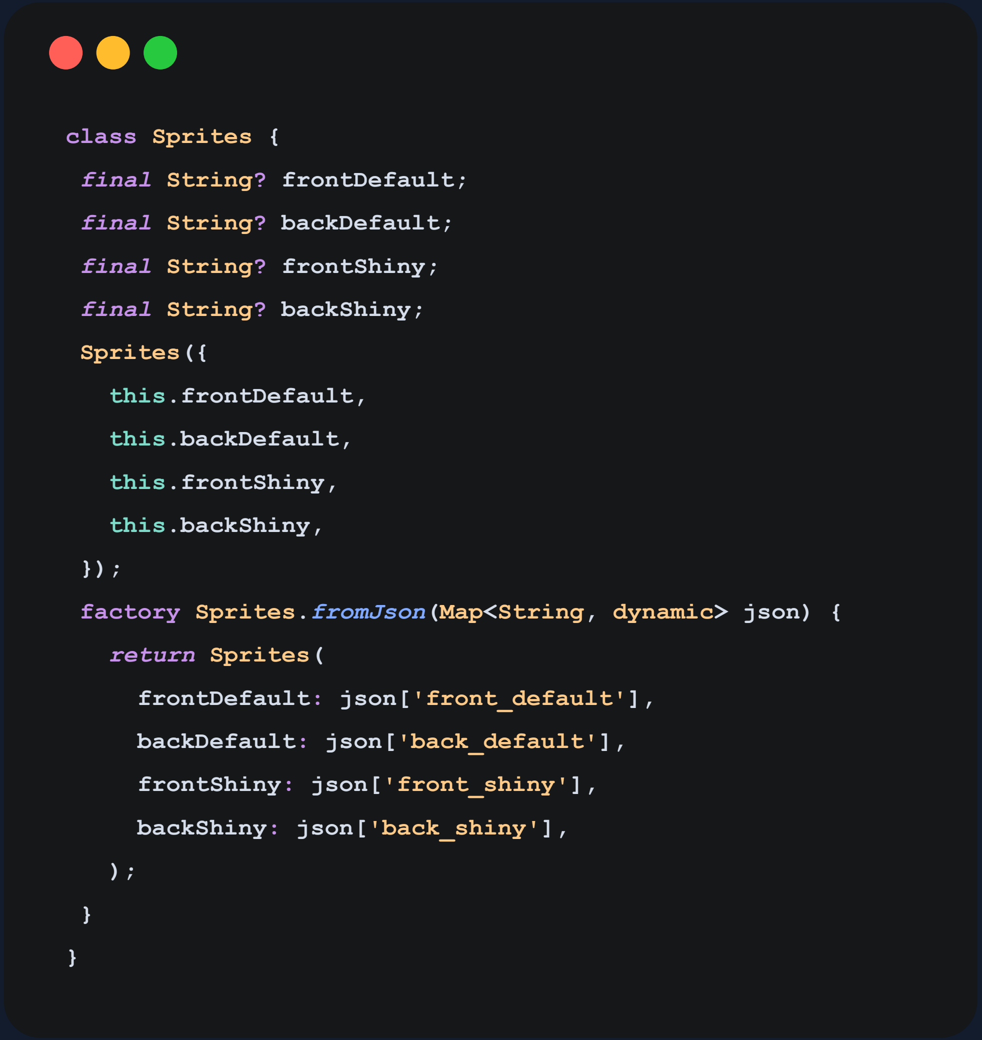
FLUTTER

EXERCÍCIO RESOLVIDO

POKE APP

MODELO

Crie um modelo para representar os **Sprites**.



```
class Sprites {
 final String? frontDefault;
 final String? backDefault;
 final String? frontShiny;
 final String? backShiny;
 Sprites({
   this.frontDefault,
   this.backDefault,
   this.frontShiny,
   this.backShiny,
 });
 factory Sprites.fromJson(Map<String, dynamic> json) {
   return Sprites(
     frontDefault: json['front_default'],
     backDefault: json['back_default'],
     frontShiny: json['front_shiny'],
     backShiny: json['back_shiny'],
   );
 }
}
```

POKE APP

MODELO

Após o **factory Sprites.fromJson** adicione o método que irá retornar a lista com as imagens.

```
// Retorna todas as imagens não nulas em uma
lista
 List<String> get allImages {
   return [
     frontDefault,
     backDefault,
     frontShiny,
     backShiny,
   ].whereType<String>().toList();
 }
```

POKE APP

MODELO

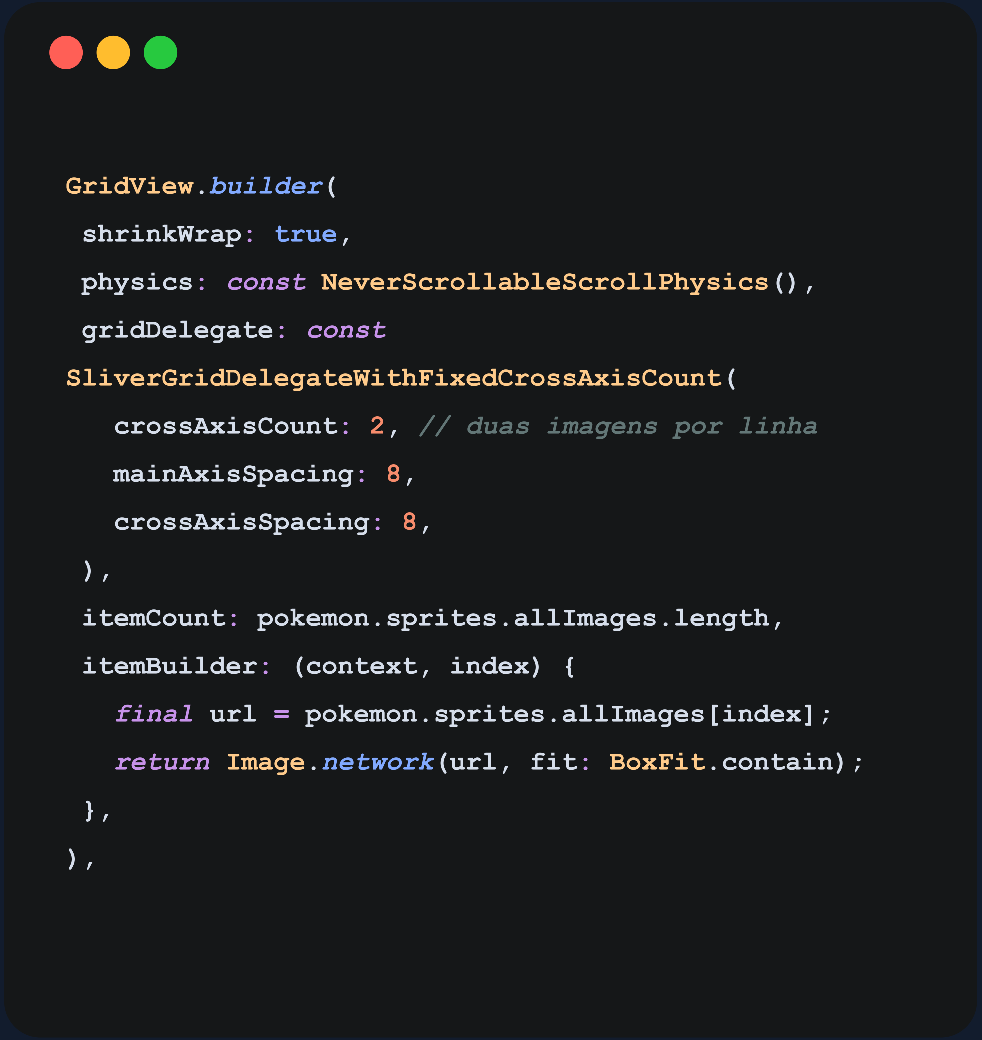
Adicione a propriedade **sprite** ao modelo do **Pokémon**.

```
class Pokemon {
 final String name;
 final List<String> types;
 final String? mainSprite;
 final Sprites sprites;
 Pokemon({required this.name,required this.types, required
this.mainSprite,
   required this.sprites,
 });
 factory Pokemon.fromJson(Map<String, dynamic> json) {
   return Pokemon(
     name: json['name'],
     types: (json['types'] as List)
             .map((t) => t['type']['name'] as String)
             .toList(),
     mainSprite:
json['sprites']['other']['official-artwork']['front_default']
,
      sprites: Sprites.fromJson(json['sprites']),
   );
 }
}
```

POKE APP

EXIBINDO NO GRID

No **_buildPokemonCard** após a **Row** referente aos tipos de Pokémons adicione o **Grid** para exibir as imagens.



```
GridView.builder(
 shrinkWrap: true,
 physics: const NeverScrollableScrollPhysics(),
 gridDelegate: const
SliverGridDelegateWithFixedCrossAxisCount(
   crossAxisCount: 2, // duas imagens por linha
   mainAxisSpacing: 8,
   crossAxisSpacing: 8,
 ),
 itemCount: pokemon.sprites.allImages.length,
 itemBuilder: (context, index) {
   final url = pokemon.sprites.allImages[index];
   return Image.network(url, fit: BoxFit.contain);
 },
),
```

FLUTTER

EXTRA: LISTA DE POKEMONS COM SCROLL INFINITO

POKE APP

MODELO

Crie o modelo do item da lista:

```
class PokemonItemList {
 final String name;
 final String url;

 PokemonItemList({required this.name, required this.url});

 factory PokemonItemList.fromJson(Map<String, dynamic> json)
{
   return PokemonItemList(name: json['name'], url:
json['url']);
 }

 String get imageUrl {
   final id = url.split("/")[url.split("/").length - 2];
   return
"https://raw.githubusercontent.com/PokeAPI/sprites/master/spr
ites/pokemon/$id.png";
 }
}
```

POKE APP

TELA DE LISTAGEM

Adicione as variáveis que serão utilizadas pelos widgets da tela de listagem de **Pokemons**

```dart
class  _PokemonListScreenState  extends
State<PokemonListScreen> {
 final ScrollController _scrollController =
ScrollController();
 final List<PokemonItemList> _pokemons = [];
 int _offset = 0;
 bool _loading = false;
 bool _hasMore = true;
```

POKE APP

TELA DE LISTAGEM

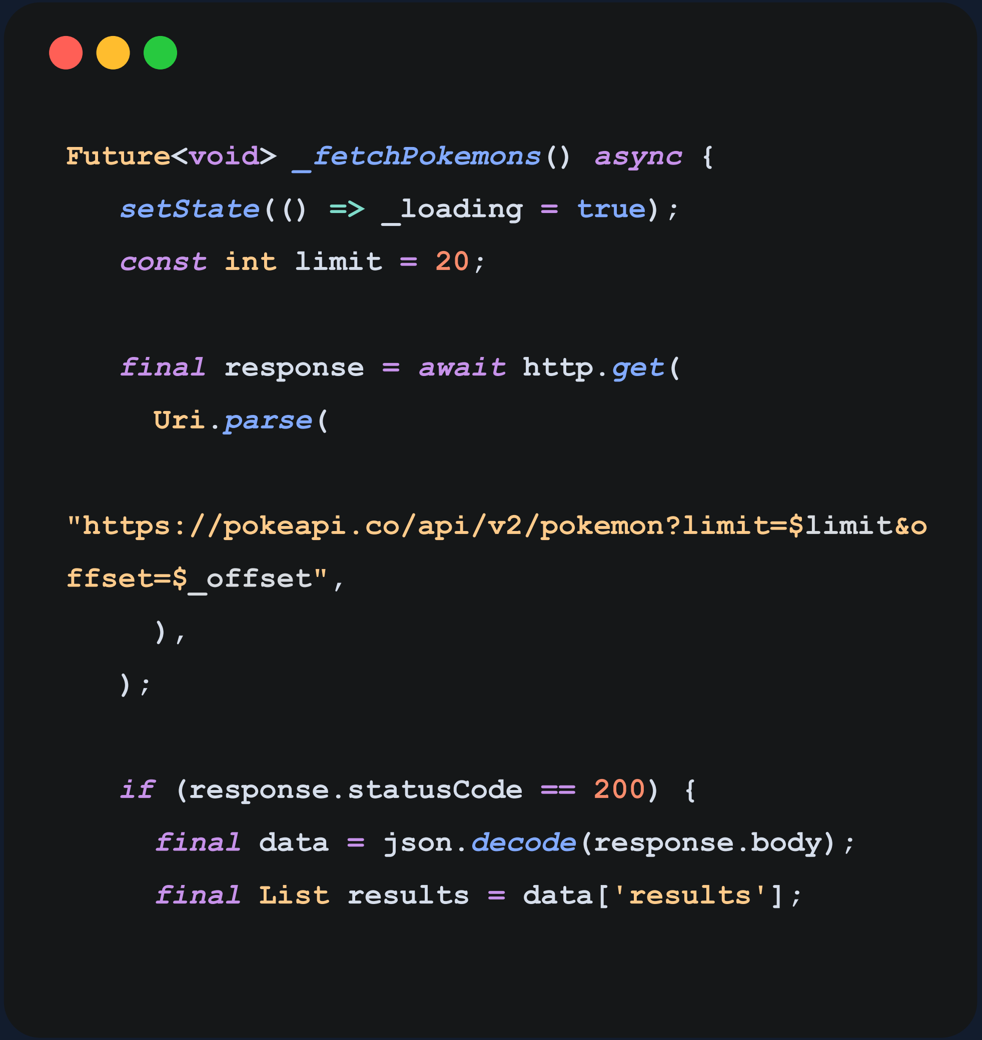
Configure o **initState**

```
@override
void initState() {
  super.initState();
  _fetchPokemons();

  _scrollController.addListener(() {
    if (_scrollController.position.pixels >=

_scrollController.position.maxScrollExtent - 200
&&
        !_loading &&
        _hasMore) {
      _fetchPokemons();
    }
  });
}
```

POKE APP

MÉTODO DE BUSCA

Crie o método para buscar a lista de **Pokémons**

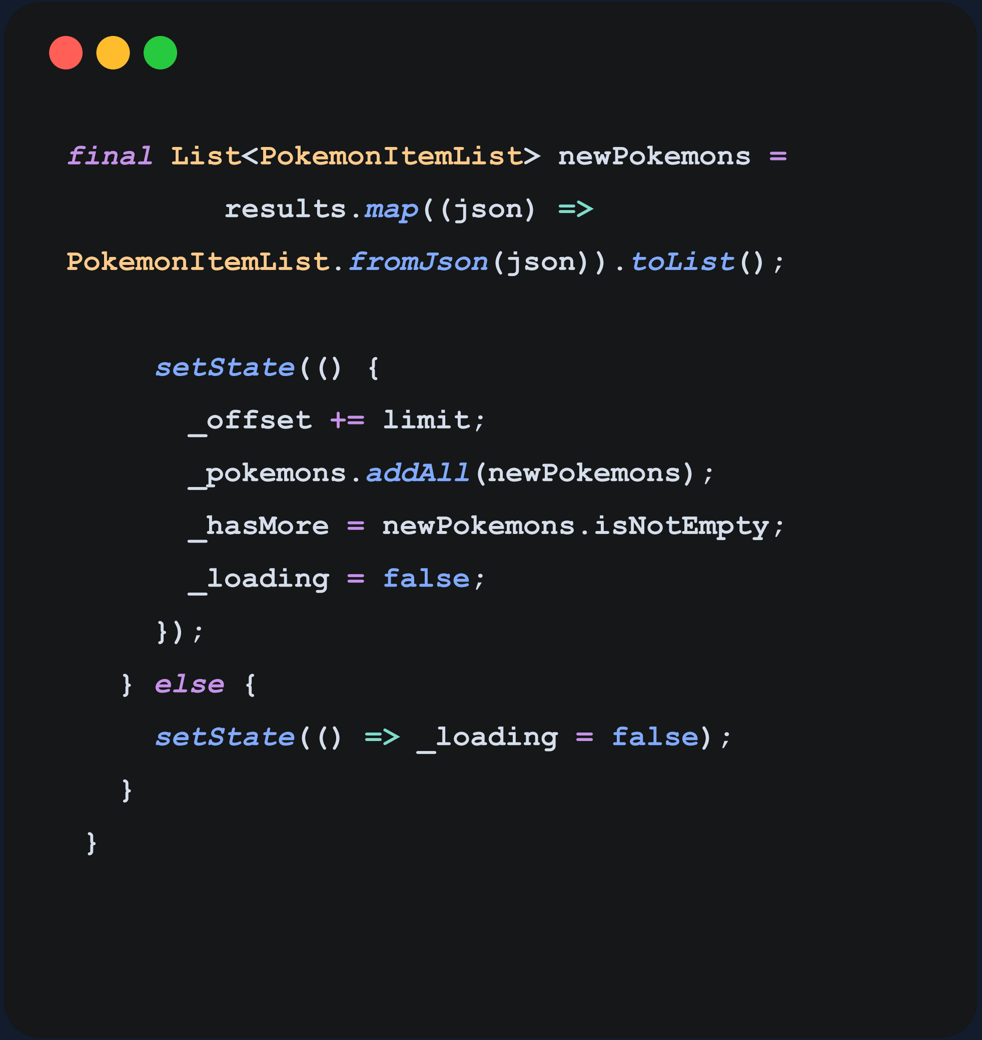


```
Future<void> _fetchPokemons() async {
   setState(() => _loading = true);
   const int limit = 20;

   final response = await http.get(
     Uri.parse(

"https://pokeapi.co/api/v2/pokemon?limit=$limit&o
ffset=$_offset",
     ),
   );

   if (response.statusCode == 200) {
     final data = json.decode(response.body);
     final List results = data['results'];
```

POKE APP

MÉTODO DE BUSCA

Crie o método para buscar a lista de **Pokémons**



```
final List<PokemonItemList> newPokemons =
         results.map((json) =>
PokemonItemList.fromJson(json)).toList();

     setState(() {
       _offset += limit;
       _pokemons.addAll(newPokemons);
       _hasMore = newPokemons.isNotEmpty;
       _loading = false;
     });
   } else {
     setState(() => _loading = false);
   }
 }
```

POKE APP

MOSTRAR POKEMON

Crie o método para buscar a lista de **Pokémons**

```
@override
Widget build(BuildContext context) {
 return Scaffold(
   appBar: AppBar(title: const Text("Pokédex")),
   body: ListView.builder(
     controller: _scrollController,
     itemCount: _pokemons.length + 1,
     itemBuilder: (context, index) {
       if (index < _pokemons.length) {
         final pokemon = _pokemons[index];
         return ListTile(
           leading: Image.network(pokemon.imageUrl),
           title: Text(pokemon.name.toUpperCase()),
         );
       } else {
         return Padding(
           padding: const EdgeInsets.all(16),
           child: Center(
             child:_hasMore ? const CircularProgressIndicator() : const
Text("Todos os Pokémons carregados"),
           ),
         );
       }
     },
   ),
 );
}
```

FLUTTER
EXERCÍCIO 2

RICKANDMORTY

EXERCÍCIO DE FIXAÇÃO

Faça um aplicativo que contenha as seguintes telas.

Home

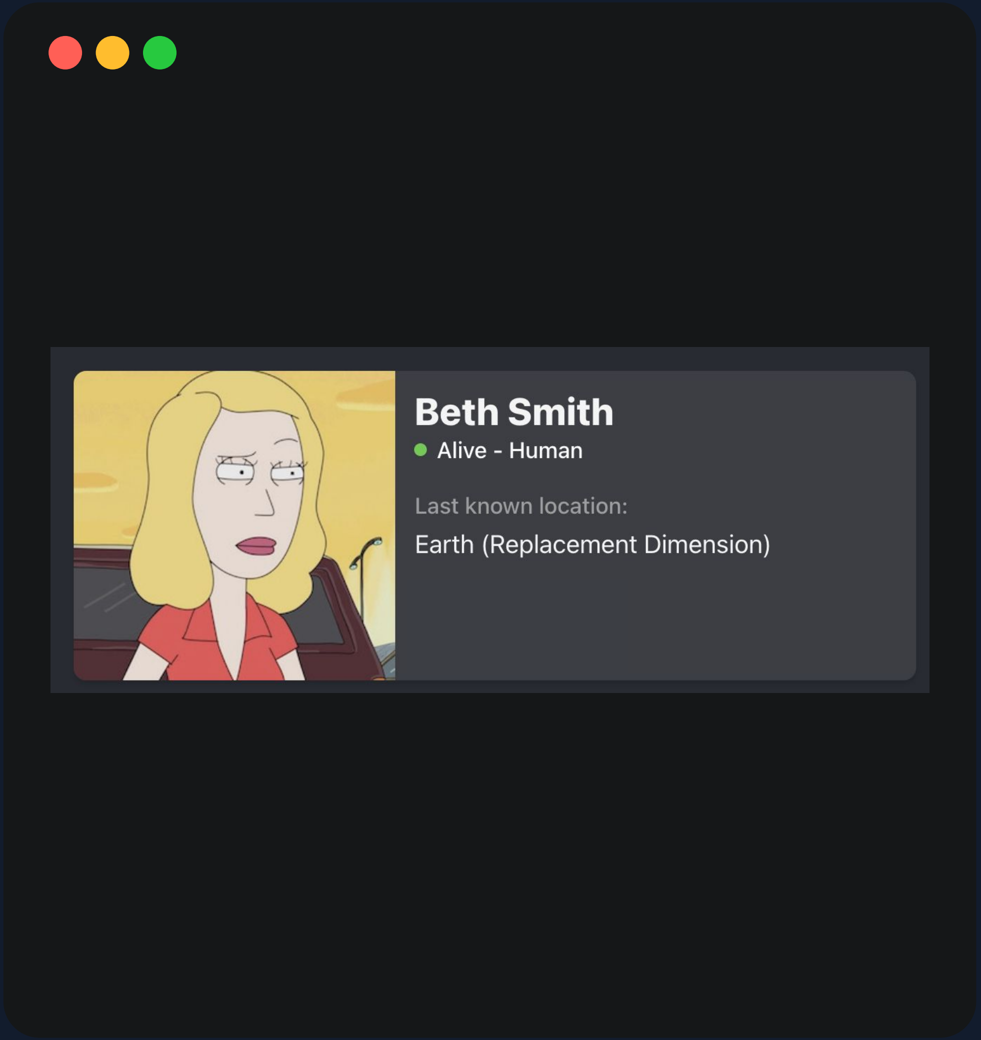
Lista de Personagens

Busca de Personagens

Utilizar a api:

https://rickandmortyapi.com/

*** Segue ao lado uma sugestão dos campos para serem exibidos.**



Beth Smith

● Alive - Human

Last known location:

Earth (Replacement Dimension)

OBRIGADO

heider-lopes-a06b2869