

Curso Técnico Superior Profissional em: Tecnologias e Programação de Sistemas de Informação

Unidade Curricular: Aplicações Centradas em Redes

2.º Ano/1.º Semestre

Docente: Michael Silva / Hugo Perdigão

Flutter - Exemplo 4

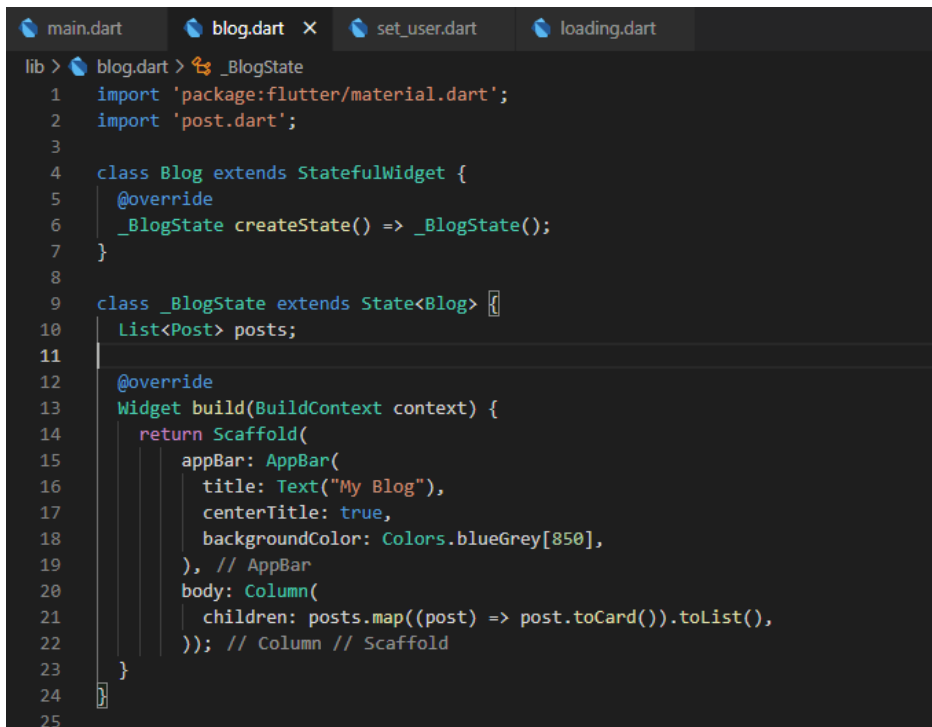
Objetivos:

- Utilizar vários ecrãs (Routes).
- Utilizar vários ecrãs
- Ir buscar informação à internet
- Passar informação entre ecrãs

1) Abrir o projeto “blog”

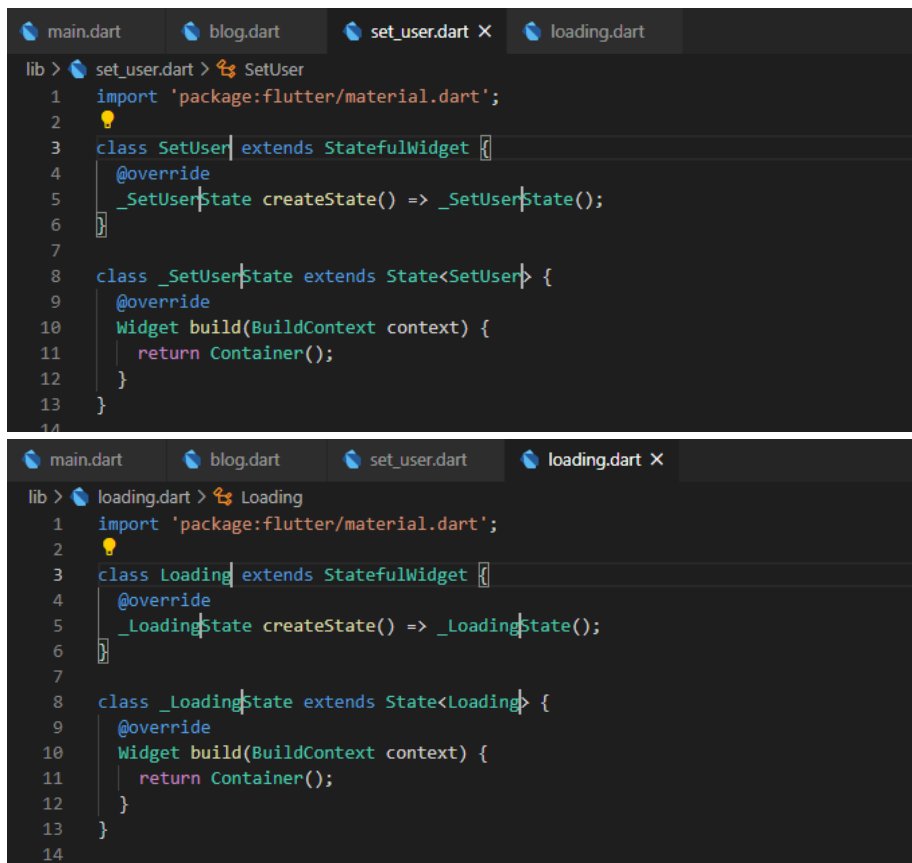
2) Criar o ficheiro blog.dart e colocar as classes Blog e _Blogstate dentro deste (é necessário incluir os imports).

Eliminar o conteúdo da lista de posts.



```
lib > blog.dart > _BlogState
1  import 'package:flutter/material.dart';
2  import 'post.dart';
3
4  class Blog extends StatefulWidget {
5    @override
6    _BlogState createState() => _BlogState();
7  }
8
9  class _BlogState extends State<Blog> {}
10   List<Post> posts;
11
12   @override
13   Widget build(BuildContext context) {
14     return Scaffold(
15       appBar: AppBar(
16         title: Text("My Blog"),
17         centerTitle: true,
18         backgroundColor: Colors.blueGrey[850],
19       ), // AppBar
20       body: Column(
21         children: posts.map((post) => post.toCard()).toList(),
22       )); // Column // Scaffold
23   }
24
25
```

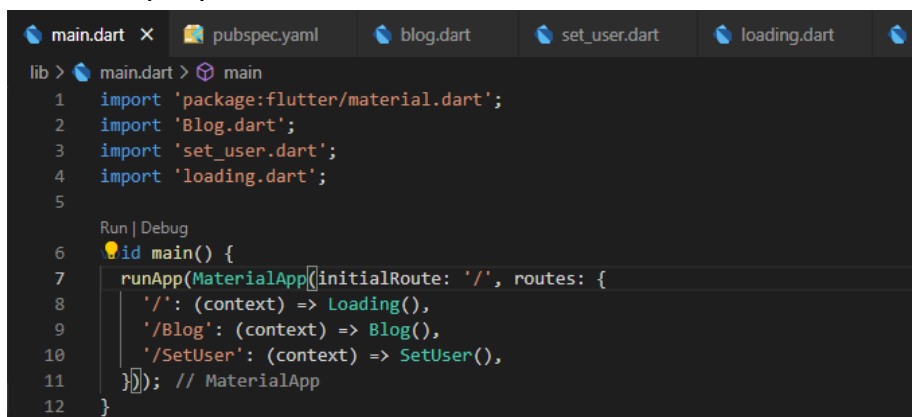
- 3) Da mesma forma, criar os ficheiros `set_user.dart` e `loading.dart` e criar um `statefulWidget` dentro de cada um



```
lib > set_user.dart > SetUser
1 import 'package:flutter/material.dart';
2
3 class SetUser extends StatefulWidget {
4   @override
5   _SetUserState createState() => _SetUserState();
6 }
7
8 class _SetUserState extends State<SetUser> {
9   @override
10  Widget build(BuildContext context) {
11    return Container();
12  }
13 }
14
```

```
lib > loading.dart > Loading
1 import 'package:flutter/material.dart';
2
3 class Loading extends StatefulWidget {
4   @override
5   _LoadingState createState() => _LoadingState();
6 }
7
8 class _LoadingState extends State<Loading> {
9   @override
10  Widget build(BuildContext context) {
11    return Container();
12  }
13 }
14
```

- 4) No ficheiro `main.dart`, importar os restantes ficheiros, criar as routes e em vez de utilizar a propriedade `home`, utilizar `initialRoute`:



```
lib > main.dart > main
1 import 'package:flutter/material.dart';
2 import 'Blog.dart';
3 import 'set_user.dart';
4 import 'loading.dart';
5
6 id main() {
7   runApp(MaterialApp(initialRoute: '/', routes: {
8     '/': (context) => Loading(),
9     '/Blog': (context) => Blog(),
10    '/SetUser': (context) => SetUser(),
11  })); // MaterialApp
12 }
```

- 5) Para poder aceder a funções http é necessário instalar o pacote http, para isso basta adicionar ao ficheiro pubspec.yaml o pacote necessário

```
main.dart pubspec.yaml blog.dart set_user.dart loading.dart
pubspec.yaml
14 # Read more about Android versioning at https://developer.android.com/studio/publish/
15 # In iOS, build-name is used as CFBundleShortVersionString while build-number used as
16 # Read more about iOS versioning at
17 # https://developer.apple.com/library/archive/documentation/General/Reference/InfoPli
18 version: 1.0.0+1
19
20 environment:
21   sdk: ">=2.7.0 <3.0.0"
22
23 dependencies:
24   flutter:
25     sdk: flutter
26   http: ^0.12.2
27
```

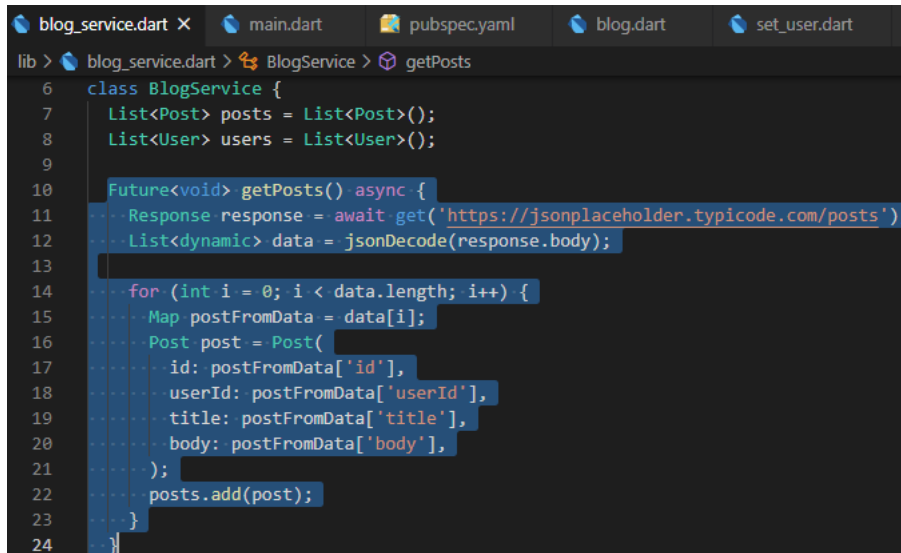
- 6) Antes de começar a utilizar o http e aceder a informação é necessário também criar uma classe User semelhante ao Post mas com os dados do user e o método toCard que cria também um widget card para o User

```
main.dart pubspec.yaml blog.dart set_user.dart loading.dart
lib > user.dart > ...
1 import 'package:flutter/material.dart';
2
3 class User {
4   int id;
5   String name;
6   String username;
7   String email;
8
9   User({this.id, this.name, this.username, this.email});
10
11   Card toCard() {
12     return Card(
13       margin: EdgeInsets.symmetric(horizontal: 20, vertical: 10),
14       child: Padding(
15         padding: const EdgeInsets.all(10.0),
16         child: Column(
17           crossAxisAlignment: CrossAxisAlignment.stretch,
18           children: <Widget>[
19             Text(
20               "${this.name} (${this.email})",
21               style: TextStyle(
22                 fontSize: 18,
23                 color: Colors.red[800],
24                 fontWeight: FontWeight.bold), // TextStyle
25             ), // Text
26           ], // <Widget>[] // Column
27         ), // Padding
28       ); // Card
29   }
30 }
```

- 7) Criar um ficheiro blog_service.dart que vai conter a class BlogService que vai aceder à internet para ir buscar a informação, podemos já criar as duas listas de Utilizadores e Posts

```
blog_service.dart main.dart pubspec.yaml blog.dart set_user.dart
lib > blog_service.dart > BlogService
1 import 'dart:convert';
2 import 'package:blog/post.dart';
3 import 'package:blog/user.dart';
4 import 'package:http/http.dart';
5
6 class BlogService {
7   List<Post> posts = List<Post>();
8   List<User> users = List<User>();
9 }
```

- 8) Esta classe vai ter mais dois métodos `getPost()` e `getUsers()` que vão à internet buscar os posts(<https://jsonplaceholder.typicode.com/posts>) e os users(<https://jsonplaceholder.typicode.com/users>) respetivamente. Estas funções têm que ser declaradas como **async** pois são assíncronas (não se sabe quando terminam e não bloqueiam o restante código).

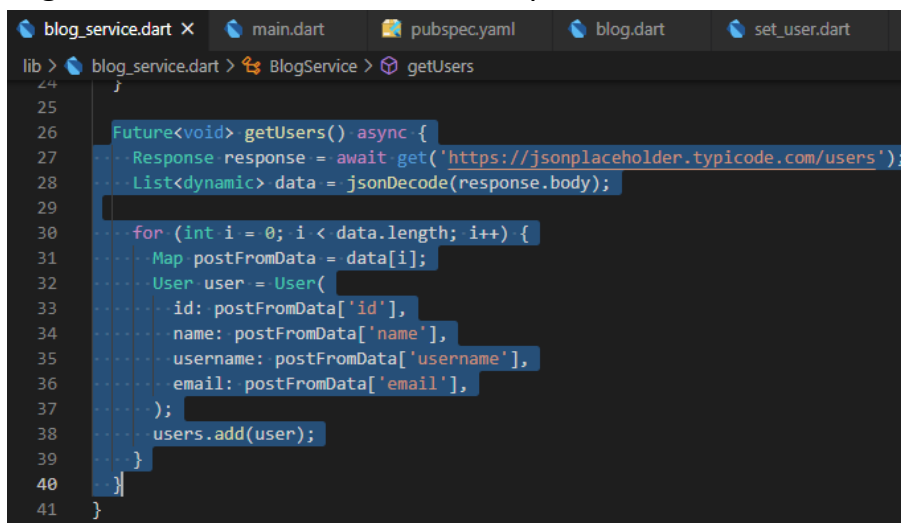


```

lib > blog_service.dart > BlogService > getPosts
6 class BlogService {
7   List<Post> posts = List<Post>();
8   List<User> users = List<User>();
9
10  Future<void> getPosts() async {
11    Response response = await get('https://jsonplaceholder.typicode.com/posts');
12    List<dynamic> data = jsonDecode(response.body);
13
14    for (int i = 0; i < data.length; i++) {
15      Map postFromData = data[i];
16      Post post = Post(
17        id: postFromData['id'],
18        userId: postFromData['userId'],
19        title: postFromData['title'],
20        body: postFromData['body'],
21      );
22      posts.add(post);
23    }
24  }

```

NOTA: a função `get` (no pacote `http`) também é assíncrona, para que o código seguinte não execute é utilizada a palavra `await`.

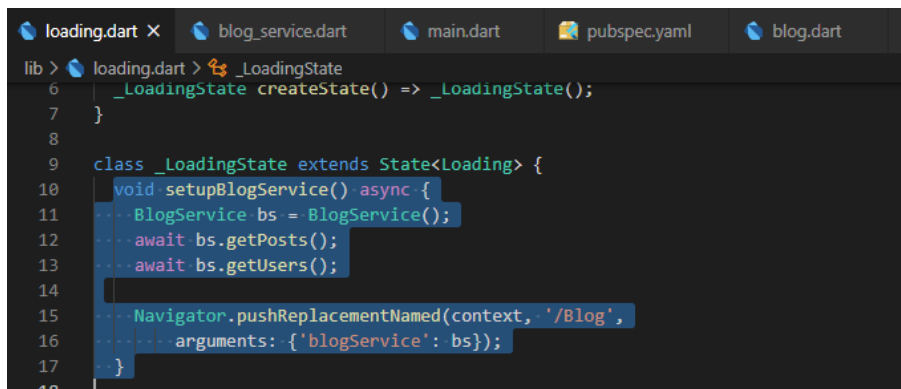


```

lib > blog_service.dart > BlogService > getUsers
24
25
26  Future<void> getUsers() async {
27    Response response = await get('https://jsonplaceholder.typicode.com/users');
28    List<dynamic> data = jsonDecode(response.body);
29
30    for (int i = 0; i < data.length; i++) {
31      Map postFromData = data[i];
32      User user = User(
33        id: postFromData['id'],
34        name: postFromData['name'],
35        username: postFromData['username'],
36        email: postFromData['email'],
37      );
38      users.add(user);
39    }
40  }
41 }

```

- 9) No ficheiro `loading.dart`, criar um método na classe `_LoadingState`, `setupBlogservice` que vai inicializar o serviço



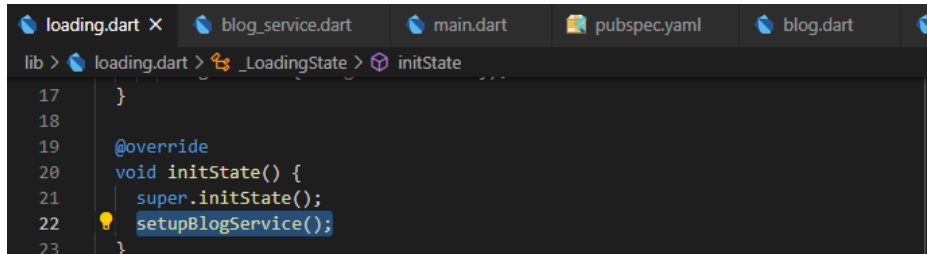
```

lib > loading.dart > _LoadingState
6  _LoadingState createState() => _LoadingState();
7  }
8
9  class _LoadingState extends State<Loading> {
10   void setupBlogService() async {
11     BlogService bs = BlogService();
12     await bs.getPosts();
13     await bs.getUsers();
14
15     Navigator.pushReplacementNamed(context, '/Blog',
16       arguments: {'blogService': bs});
17   }
18 }

```

NOTA: a classe Navigator permite navegar entre as routes fazendo push ou pop como se fosse uma pilha, é sempre necessário passar o contexto e a route, neste caso é passado um 3º argumento que é o serviço.

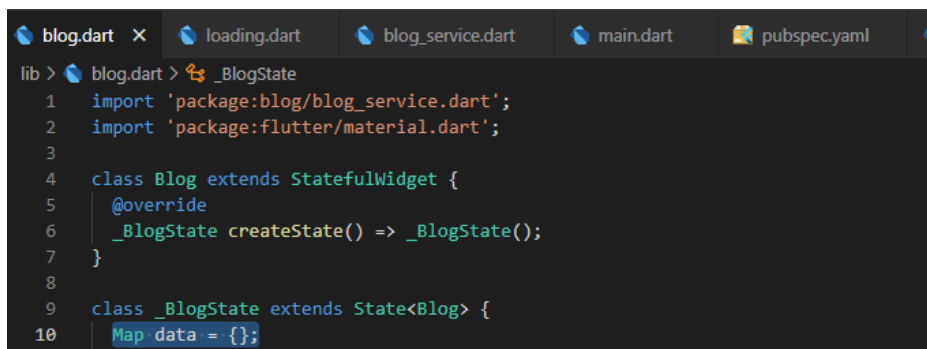
- 10) Este método é depois chamado na função initState (tem que ser overridden) para inicializar o serviço.



```

lib > loading.dart > _LoadingState > initState
17   }
18
19   @override
20   void initState() {
21     super.initState();
22     setupBlogService();
23   }
  
```

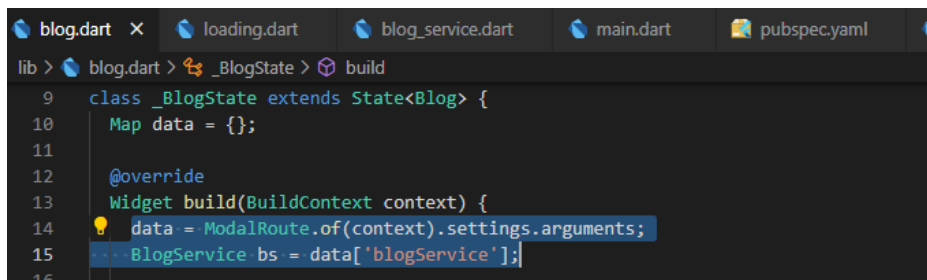
- 11) No ficheiro blog.dart é necessário agora introduzir o argumento que é passado (serviço) para poder popular a lista uma vez que já não existe o array com os dados. Criando um Map.



```

lib > blog.dart > _BlogState
1   import 'package:blog/blog_service.dart';
2   import 'package:flutter/material.dart';
3
4   class Blog extends StatefulWidget {
5     @override
6     _BlogState createState() => _BlogState();
7   }
8
9   class _BlogState extends State<Blog> {
10    Map data = {};
  
```

- 12) Este deve ser atribuído na função build, de onde é depois retirado o serviço

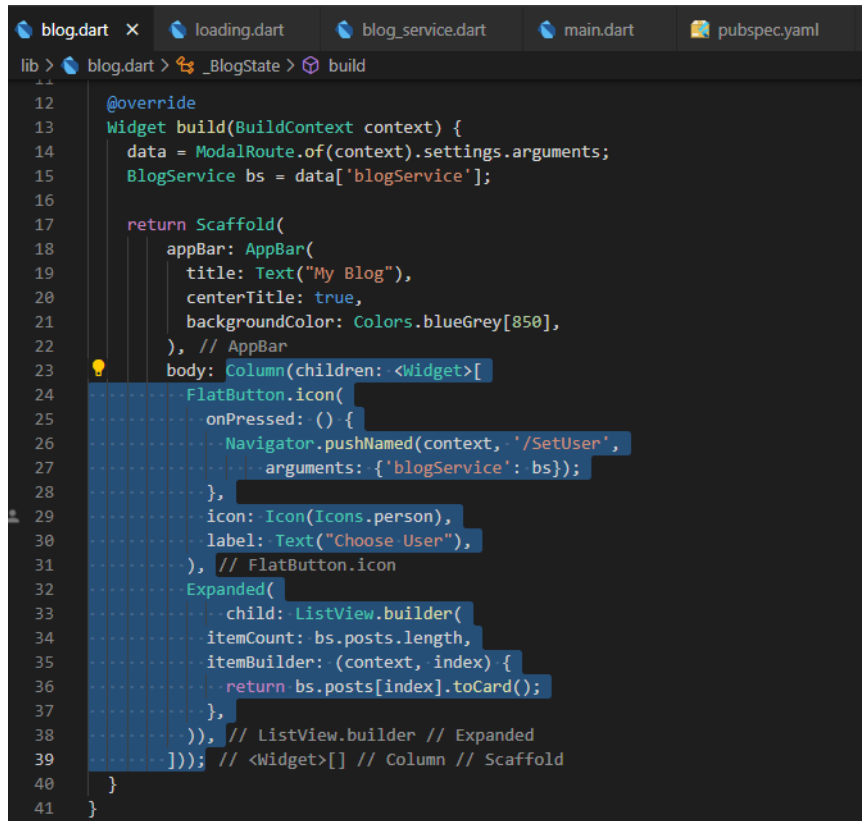


```

lib > blog.dart > _BlogState > build
9   class _BlogState extends State<Blog> {
10    Map data = {};
11
12    @override
13    Widget build(BuildContext context) {
14      data = ModalRoute.of(context).settings.arguments;
15      BlogService bs = data['blogService'];
16
  
```

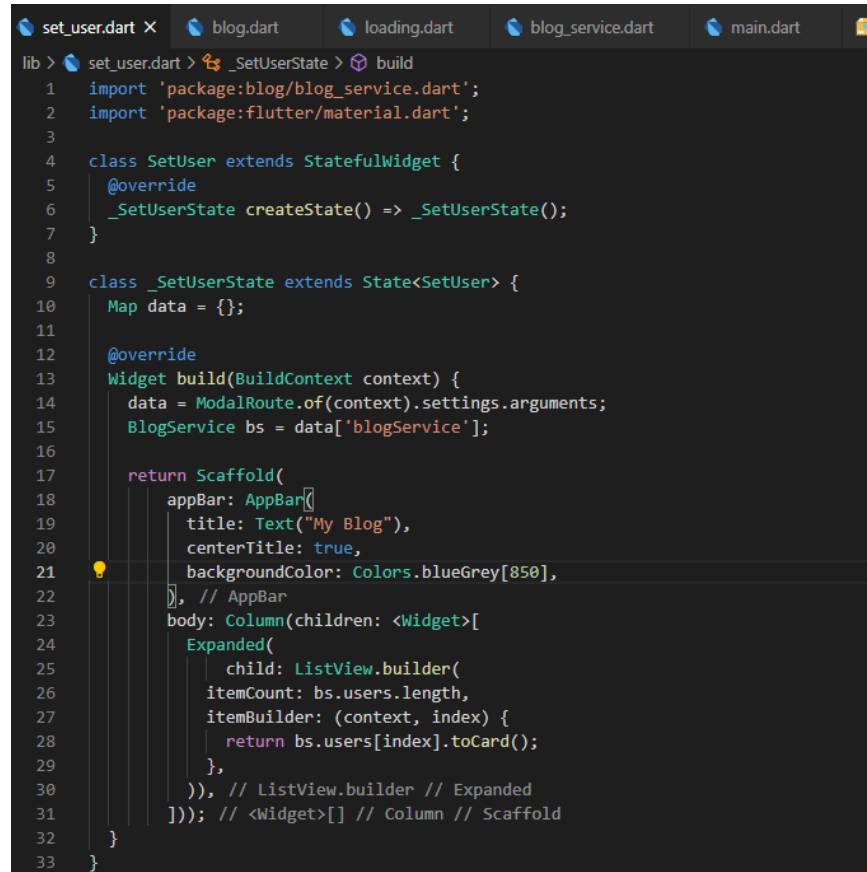
13) Depois deve ser alterado o aspeto da aplicação para:

- Ter um botão para escolher o autor (navega para a route `'/SetUser'`)
- Uma listview em vez de column para permitir o scroll (deve estar dentro de um objeto expanded)
- Utilizar o serviço em vez de da variável anterior.



```
lib > blog.dart > _BlogState > build
12  @override
13  Widget build(BuildContext context) {
14    data = ModalRoute.of(context).settings.arguments;
15    BlogService bs = data['blogService'];
16
17    return Scaffold(
18      appBar: AppBar(
19        title: Text("My Blog"),
20        centerTitle: true,
21        backgroundColor: Colors.blueGrey[850],
22      ), // AppBar
23      body: Column(children: <Widget>[
24        FlatButton.icon(
25          onPressed: () {
26            Navigator.pushNamed(context, '/SetUser',
27              arguments: {'blogService': bs});
28          },
29          icon: Icon(Icons.person),
30          label: Text("Choose User"),
31        ), // FlatButton.icon
32        Expanded(
33          child: ListView.builder(
34            itemCount: bs.posts.length,
35            itemBuilder: (context, index) {
36              return bs.posts[index].toCard();
37            },
38          ), // ListView.builder // Expanded
39      ])); // <Widget>[] // Column // Scaffold
40  }
41 }
```

14) Da mesma forma configurar a class SetUser



```
lib > set_user.dart > _SetUserState > build
1 import 'package:blog/blog_service.dart';
2 import 'package:flutter/material.dart';
3
4 class SetUser extends StatefulWidget {
5   @override
6   _SetUserState createState() => _SetUserState();
7 }
8
9 class _SetUserState extends State<SetUser> {
10   Map data = {};
11
12   @override
13   Widget build(BuildContext context) {
14     data = ModalRoute.of(context).settings.arguments;
15     BlogService bs = data['blogService'];
16
17     return Scaffold(
18       appBar: AppBar(
19         title: Text("My Blog"),
20         centerTitle: true,
21         backgroundColor: Colors.blueGrey[850],
22       ), // AppBar
23       body: Column(children: <Widget>[
24         Expanded(
25           child: ListView.builder(
26             itemCount: bs.users.length,
27             itemBuilder: (context, index) {
28               return bs.users[index].toCard();
29             },
30           ), // ListView.builder // Expanded
31       ])); // <Widget>[] // Column // Scaffold
32   }
33 }
```

Webgrafia

<https://api.flutter.dev/flutter/widgets/StatefulWidget-class.html>

<https://api.flutter.dev/flutter/painting/TextStyle-class.html>

<https://api.flutter.dev/flutter/material/Card-class.html>

<https://api.flutter.dev/flutter/widgets/Column-class.html>

<https://api.flutter.dev/flutter/widgets/SizedBox-class.html>

<https://api.flutter.dev/flutter/painting/EdgeInsets-class.html>