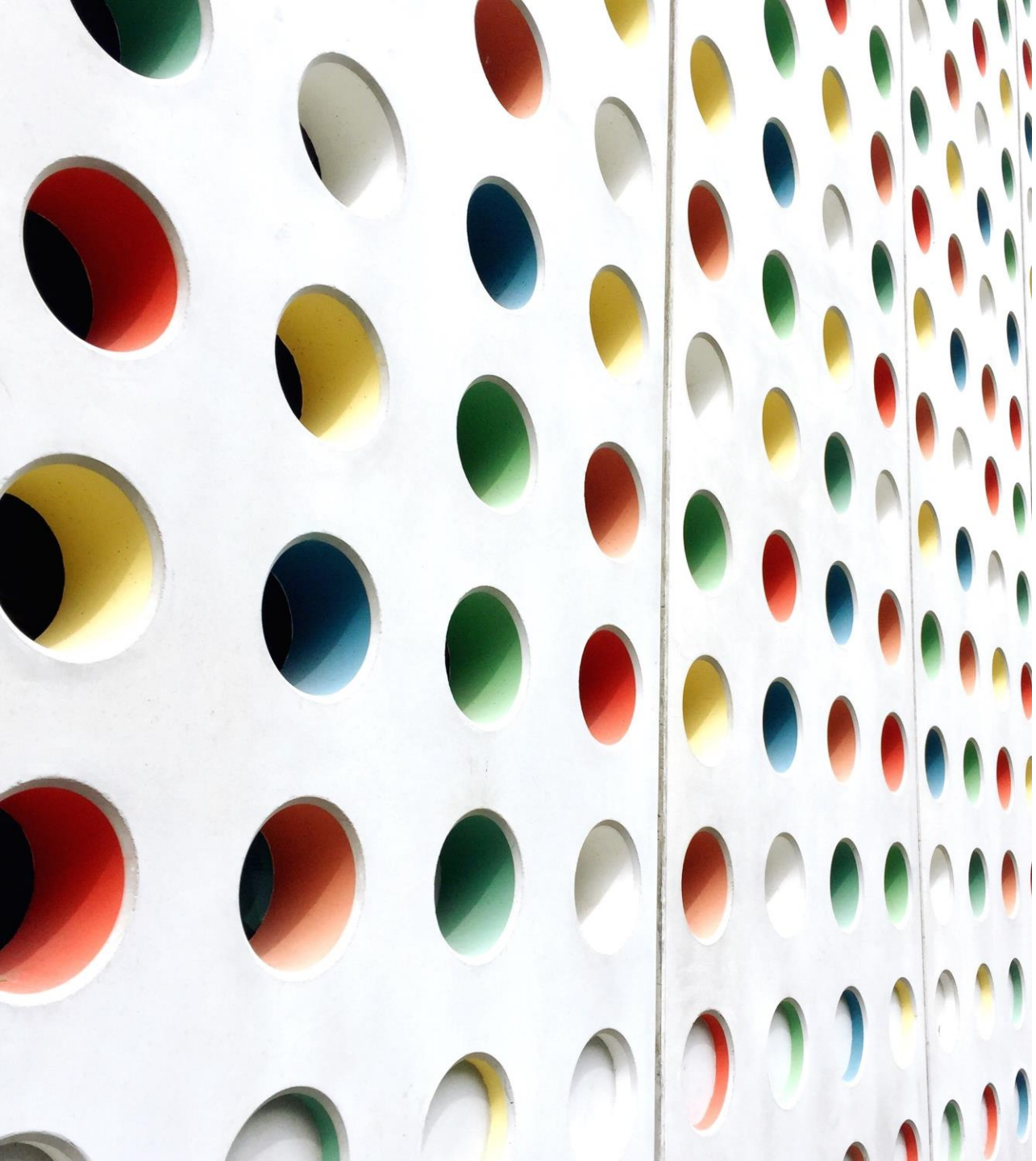


Programação para Dispositivos Móveis

FLUTTER

Professor Jean Carlo Wagner

11/04/2023

Agenda

- Cadastro
- Controle sobre a UI: exemplo básico
- Navegação: exemplo básico

Cadastro

- Um cadastro contempla lista de usuários, assim como métodos para salvar e excluir, ou seja, inserir, alterar e excluir, além da navegação e *widgets Stateless* e *Stateful*.
- O primeiro passo será criar um projeto em um diretório destinado aos projetos e desenvolvimento Flutter/Dart:

```
C:\Users\Jean\Documents\Flutter>flutter create flutter_crud
Creating project flutter_crud...
Running "flutter pub get" in flutter_crud...
Resolving dependencies in flutter_crud... (2.1s)
+ async 2.10.0 (2.11.0 available)
+ boolean_selector 2.1.1
+ characters 1.2.1 (1.3.0 available)
+ clock 1.1.1
+ collection 1.17.0 (1.17.1 available)
+ cupertino_icons 1.0.5
+ fake_async 1.3.1
+ flutter 0.0.0 from sdk flutter
+ flutter_lints 2.0.1
+ flutter_test 0.0.0 from sdk flutter
```

Cadastro

```
+ js 0.6.5 (0.6.7 available)
+ lints 2.0.1
+ matcher 0.12.13 (0.12.15 available)
+ material_color_utilities 0.2.0 (0.3.0 available)
+ meta 1.8.0 (1.9.1 available)
+ path 1.8.2 (1.8.3 available)
+ sky_engine 0.0.99 from sdk flutter
+ source_span 1.9.1
+ stack_trace 1.11.0
+ stream_channel 2.1.1
+ string_scanner 1.2.0
+ term_glyph 1.2.1
+ test_api 0.4.16 (0.5.1 available)
+ vector_math 2.1.4
Changed 24 dependencies in flutter_crud!
Wrote 127 files.

All done!
You can find general documentation for Flutter at: https://docs.flutter.dev/
Detailed API documentation is available at: https://api.flutter.dev/
```

Cadastro

```
If you prefer video documentation, consider: https://www.youtube.com/c/flutterdev

In order to run your application, type:

  $ cd flutter_crud
  $ flutter run

Your application code is in flutter_crud\lib\main.dart.


C:\Users\Jean\Documents\Flutter>cd flutter_crud

C:\Users\Jean\Documents\Flutter\flutter_crud>
```

Cadastro

- Neste cadastro cada usuário pode ter um *avatar*. Para tal, procure por imagens de *avatar*, por exemplo, em sites como https://pixabay.com/images/search/avatar/
- Após selecionar os *avatars*, na pasta lib, crie uma pasta models, a qual representará os modelos.
- Na seqüência, crie um arquivo user.dart, o qual receberá as classes de usuários.

```dart
import 'package:flutter/material.dart';

class User {
  final String id;
  final String name;
  final String email;
  final String avatarUrl;
  const User({
    required this.id,
    required this.name,
    required this.email,
    required this.avatarUrl,
  });
}
```

Cadastro

- Em seguida, crie outra pasta em lib, chamada data e insira um arquivo novo, como por exemplo, dummy_users.dart. Este arquivo receberá o cadastro de cada usuário, com id, name, email e avatarUrl:

```
import 'package:flutter_crud/models/user.dart';

const DUMMY_USERS = {
  '1': const User(
    id: '1',
    name: 'Maria',
    email: 'maria@senac.br',
    avatarUrl:
        'https://cdn.pixabay.com/photo/2021/11/12/03/04/woman-6787784__340.png',
  ),
  '2': const User(
    id: '2',
    name: 'Rafael',
    email: 'rafael@senac.br',
    avatarUrl:
        'https://cdn.pixabay.com/photo/2016/04/01/12/11/avatar-1300582__340.png',
  ),
```

Cadastro

```
  '3': const User(
    id: '3',
    name: 'Michel',
    email: 'michel@senac.br',
    avatarUrl:
        'https://cdn.pixabay.com/photo/2016/03/31/20/11/avatar-1295575__340.png',
  ),
  '4': const User(
    id: '4',
    name: 'Tiago',
    email: 'tiago@senac.br',
    avatarUrl:
        'https://cdn.pixabay.com/photo/2016/03/31/20/31/amazed-1295833__340.png',
  ),
  '5': const User(
    id: '5',
    name: 'Leandro',
    email: 'leandro@senac.br',
    avatarUrl:
        'https://cdn.pixabay.com/photo/2016/03/31/20/27/avatar-1295773__340.png',
  ),
};
```

Cadastro

- Feito o cadastro inicial em dummy_users.dart, vamos criar outra pasta em lib, chamada views. Nesta pasta crie um arquivo user_list.dart o qual substituirá a tela de projeto inicial, referente ao contador com FAB (*Floating Action Button*), cujo tema é azul:

```dart
import 'package:flutter/material.dart';

class UserList extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Lista de usuários'),
      ),
    );
  }
}
```

Cadastro

- No arquivo principal, substituir const MyHomePage(title: 'Flutter Demo Home Page') por UserList(). Obviamente, também excluir a classe MyHomePage.

```
import 'package:flutter/material.dart';
import 'package:fluttercrud/views/user_list.dart';

void main() {
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        primarySwatch: Colors.blue,
        visualDensity:
VisualDensity.adaptivePlatformDensity,
      ),
      home: UserList(),
    );
  }
}
```

Cadastro

- A seguir proceda às modificações conforme abaixo.
- O operador *spread* {... } permite definir um número indefinido de parâmetros para uma função, *array* ou objeto.

```
import 'package:flutter/material.dart';
import 'package:fluttercrud/data/dummy_users.dart';

class UserList extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    const users = {...DUMMY_USERS};

    return Scaffold(
      appBar: AppBar(
        title: Text('Lista de usuários'),
      ),
      body: ListView.builder(
        itemCount: users.length,
        itemBuilder: (ctx, i) => Text(users.values.elementAt(i).name),
      ),);}}
```

Cadastro

- A seguir crie uma pasta para alocar os *widgets*, por exemplo, components e, dentro desta pasta, crie um arquivo chamado user_tile.dart.

```dart
import 'package:flutter/material.dart';

import '../data/dummy_users.dart';

class UserTile extends StatelessWidget {
  final User texto;

  const UserTile(this.user);

  @override
  Widget build(BuildContext context) {
    final avatar = user.avatarUrl == null || user.avatarUrl.isEmpty
        ? CircleAvatar(child: Icon(Icons.person))
        : CircleAvatar(backgroundImage: NetworkImage(user.avatarUrl));
    return ListTile(
      leading: avatar,
    );
  }
}
```

como um "bloco de Lego".

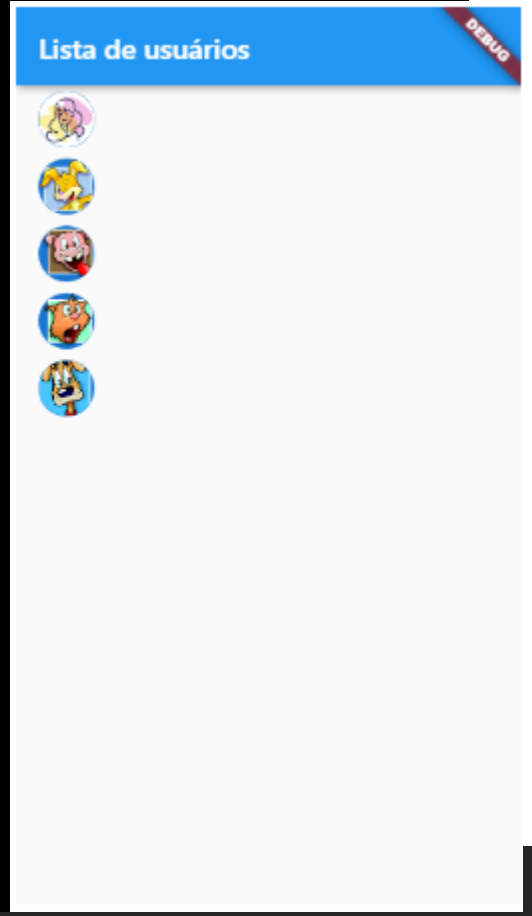
se o *user* não tiver *avatar*, mostrar apenas um ícone padrão.

ctrl + space

Cadastro

- Em user_list troque Text(users.values.elementAt(i).name), por UserTile(users.values.elementAt(i)),

```dart
import 'package:flutter/material.dart';
import 'package:fluttercrud/components/user_tile.dart';
import 'package:fluttercrud/data/dummy_users.dart';

class UserList extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    final users = {...DUMMY_USERS};

    return Scaffold(
      appBar: AppBar(
        title: Text('Lista de usuários'),
      ),
      body: ListView.builder(
        itemCount: users.length,
        itemBuilder: (ctx, i) => UserTile(users.values.elementAt(i)),
      ),
    );
  }
}
```



Cadastro

- Em user_tile inclua as linhas a seguir e salve. Veja o que ocorreu.

```
import 'package:flutter/material.dart';

import '../data/dummy_users.dart';

class UserTile extends StatelessWidget {
  final User user;

  const UserTile(this.user);

  @override
  Widget build(BuildContext context) {
    final avatar = user.avatarUrl == null || user.avatarUrl.isEmpty
        ? CircleAvatar(child: Icon(Icons.person))
        : CircleAvatar(backgroundImage: NetworkImage(user.avatarUrl));
    return ListTile(
      leading: avatar,
      title: Text(user.name),
      subtitle: Text(user.email),
```

Cadastro

```
      trailing: Container(
        width: 100,
        child: Row(children: <Widget>[
          IconButton(
              onPressed: () {}, icon: Icon(Icons.edit), color: Colors.blue),
          IconButton(
              onPressed: () {}, icon: Icon(Icons.delete), color: Colors.red),
        ]),
      ),
    );
  }
}
```

como se fosse new Icon

função vazia

Errata: incluir a linha referente a "trailing" no código.

Cadastro

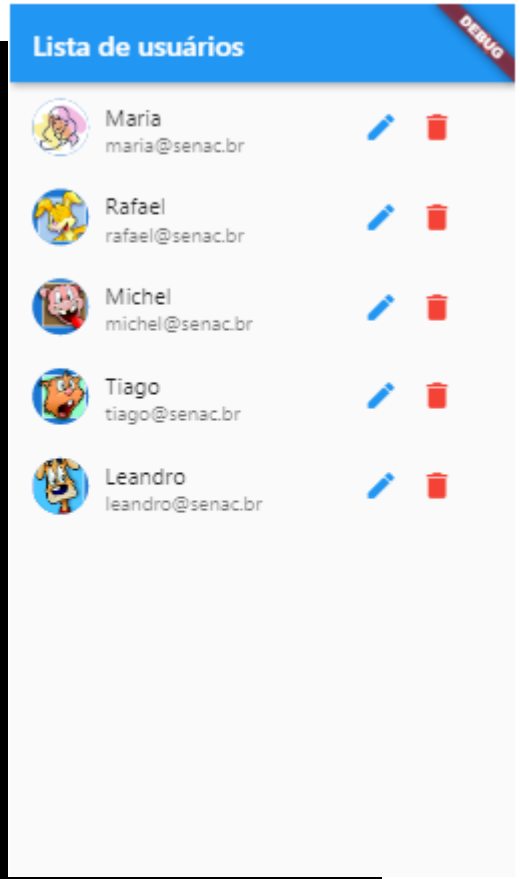
- Em user_list inclua a linha a seguir e salve.

```dart
import 'package:flutter/material.dart';
import 'package:fluttercrud/components/user_tile.dart';
import 'package:fluttercrud/data/dummy_users.dart';
class UserList extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    final users = {...DUMMY_USERS};
    return Scaffold(
      appBar: AppBar(
        title: Text('Lista de usuários'),
        actions: <Widget>[IconButton(onPressed: () {}, icon: Icon(Icons.add))],
      ),
      body: ListView.builder(
        itemCount: users.length,
        itemBuilder: (ctx, i) => UserTile(users.values.elementAt(i)),
      ),
    );
  }
}
```

renderiza apenas o que estiver sendo mostrado na tela, apesar de uma quantidade enorme de cadastrados em lista.

cria um componente

quantidade de itens a serem mostrados na lista



Cadastro

- Em user_list inclua as linhas a seguir e salve.

```
return Scaffold(
  appBar: AppBar(
    title: Text('Lista de usuários'),
    actions: <Widget>[
      Padding(                                  // para evidenciarmos o icon add.
        padding: const EdgeInsets.all(14.0),
        child: IconButton(
          icon: Icon(Icons.add),
          onPressed: () {},
        ),
      ),
    ],
  ),
  body: ListView.builder(
    itemCount: users.length,
    itemBuilder: (ctx, i) => UserTile(users.values.elementAt(i)),
  ),
);
```

Cadastro

- Como utilizaremos, a partir deste ponto, um *provider*, a fim de facilidade, crie uma nova pasta provider em lib. Dentro da pasta provider, crie um arquivo users.dart, o qual controlará a lista de usuários (CRUD por exemplo).
- Em pubspec.yaml, insira a linha a seguir dentro de dependencies:

```
dependencies:
  flutter:
    sdk: flutter
  provider:
```

- Por meio de ***mixins*** podemos adicionar um conjunto de "características" a uma classe sem a necessidade de se utilizar a herança.

❖ Leia o artigo https://www.treinaweb.com.br/blog/o-que-sao-mixins-e-qual-sua-importancia-no-dart para mais informações.

Cadastro

- Carregue o arquivo users.dart com o código a seguir:

```
import 'package:flutter/material.dart';
import 'package:fluttercrud/data/dummy_users.dart';

class Users with ChangeNotifier {
  final Map<String, User> _items = {...DUMMY_USERS};

  //um elemento só será incluído em _items quando o método tradicional for requisitado. Os elementos
serão incluídos por meio do método spread, gerando um novo array, preservando a lista _items
original
  List<User> get all {
    return [..._items.values];
  }

//sempre que "pegarmos" 'all' só para termos 'count', geraremos um clone (lista nova). Assim o
método count retorna quantos elementos temos no clone.
  int get count {
    return _items.length;
  }
}
```

Cadastro

- ChangeNotifier é um *mixin*, além de ter também um padrão Observer*.
- *Na programação reativa, existe uma entidade que envia eventos e existe outra entidade que se inscreve nesses eventos, não importa se são um ou milhares: estaremos ouvindo mudanças nesses dados e simplesmente reagindo a estes eventos.

- Provider é uma classe onde adicionamos o que queremos distribuir na raiz de nossa aplicação e depois simplesmente chamamos Provider.of para receber o valor da memória.
- Para expor um valor usando *provider*, envolva qualquer *widget* em um *provider* e passe sua variável. Então, todos os *widgets* adicionados a esse *provider* conseguirão acessar essa variável. Uma alternativa, para objetos complexos, muitos *providers* vão expor um construtor que receberá uma função para criar o valor.
- Os *providers* permitem que você não apenas exponha um valor, mas também crie, ouça e descarte-o.
- Para expor um objeto recém-criado, use o construtor padrão de um *provider*. Não use o construtor .value se quiser criar um objeto, caso contrário, poderá ter efeitos colaterais indesejados.

Cadastro

- O arquivo main.dart deve ser renderizado para:

```dart
import 'package:flutter/material.dart';
import 'package:fluttercrud/provider/users.dart';
import 'package:fluttercrud/views/user_list.dart';
import 'package:provider/provider.dart';

void main() {
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        primarySwatch: Colors.blue,
        visualDensity: VisualDensity.adaptivePlatformDensity,
      ),
      home: UserList(),
    );
  }
}
```

Cadastro

- Porém, ainda em main.dart, o provider pode envolver, ou o *widget* MaterialApp, ou o *widget* UserList. Como no futuro o *widget* UserList será utilizado para a navegação nas telas, assim vamos envolver o *widget* superior na árvore de *widgets*, MaterialApp, com *provider*:

```
import 'package:flutter/material.dart';
import 'package:fluttercrud/provider/users.dart';
import 'package:fluttercrud/views/user_list.dart';
import 'package:provider/provider.dart';

void main() {
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MultiProvider(
      providers: [
        ChangeNotifierProvider(
          create: (ctx) => Users(),
```

Cadastro

```
        )
      ],
      child: MaterialApp(
        title: 'Flutter Demo',
        theme: ThemeData(
          primarySwatch: Colors.blue,
          visualDensity: VisualDensity.adaptivePlatformDensity,
        ),
        home: UserList(),
      ),
    );
  }
}
```

- A seguir, em user_list.dart, exclua referências a "*dummy*", renderize o código para:

```
import 'package:flutter/material.dart';
import 'package:fluttercrud/components/user_tile.dart';
import 'package:provider/provider.dart';
import '../provider/users.dart';
```

Cadastro

```
class UserList extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    final Users users = Provider.of(context);

    return Scaffold(
      appBar: AppBar(
        title: Text('Lista de usuários'),
        actions: <Widget>[
          Padding(
            padding: const EdgeInsets.all(14.0),
            child: IconButton(
              icon: Icon(Icons.add),
              onPressed: () {},
            ),
          ),
        ],
      ),
      body: ListView.builder(
        itemCount: users.count, //substituir length por count
```

Cadastro

```
        itemBuilder: (ctx, i) =>
            UserTile(users.all.elementAt(i)), //.values. por .all.
      ),
    );
  }
}
```

- Lembrando que sempre que formos criar um *provider*, devemos utilizar o método create.
- Em provider (users.dart) podemos criar alguns métodos para facilitar a aplicação:

```
import 'package:flutter/material.dart';
import 'package:fluttercrud/data/dummy_users.dart';

class Users with ChangeNotifier {
  final Map<String, User> _items = {...DUMMY_USERS};

  //um elemento só será incluído em _items quando o método tradicional for requisitado. Os elementos serão incluídos por meio do método spread, gerando um novo array, preservando a lista _items original
```

Cadastro

```
  List<User> get all {
    return [..._items.values];
  }

//sempre que "pegarmos" 'all' só para termos 'count', geraremos um clone (lista nova). Assim o método count retorna quantos elementos temos no clone.
  int get count {
    return _items.length;
  }


  User byIndex(int i) {
    return _items.values.elementAt(i);
  }
}
```

Cadastro

- Agora, ao invés de acessarmos a lista por all, sempre clonando-a, poderemos acessá-la por byIndex, conforme visto em user_list.dart:

```
...
      body: ListView.builder(
        itemCount: users.count, //substituir length por count
        itemBuilder: (ctx, i) =>
            UserTile(users.byIndex(i)), //.values. por .all.
      ),
    );
  }
}
```

Cadastro

- Em users.dart, proceda às alterações a seguir:

```
...

  int get count {
    return _items.length;
  }

  User byIndex(int i) {
    return _items.values.elementAt(i);
  }

  void put(User user) {
    if (user == null) {
      return;
    }

    //adicionar ou alterar
    notifyListeners();
  }
}
```

refere-se à inserção de usuários na lista

Cadastro

- Considerando ainda o arquivo users.dart, façamos um exemplo:

```
...

  void put(User user) {
    if (user == null) {
      return;
    }
    _items.putIfAbsent(
        '1000',
        () => User(
              id: '1000',
              name: 'teste',
              email: 'teste@teste.com',
              avatarUrl: '',
            ));
    //adicionar ou alterar
    notifyListeners();
  }
}
```

Inserir um ID qualquer, key=1000 por exemplo, e retornar User (de forma fixa).

Cadastro

- A seguir (em users.dart) vamos chamar o método put, sem no entanto acionar o método notifyListeners(), o qual vamos comentar por enquanto.
- Passemos user como parâmetro fixo, sem a verificação se o novo parâmetro do usuário já existe.

```
  void put(User user) {
    if (user == null) {
      return;
    }
    //adicionar
    final id = Random().nextDouble().toString();
    _items.putIfAbsent(
        id,
        () => User(
              id: id,
              name: user.name,
              email: user.email,
              avatarUrl: user.avatarUrl,
            ));
    //alterar
    //notifyListeners();
  }
}
```

- ao inserirmos o widget Random(), automaticamente há a inclusão de: `import 'dart:math';`

Cadastro

- Em user_list.dart vamos acrescentar função *add* ao botão correspondente:

```
...
            child: IconButton(
              icon: Icon(Icons.add),
              onPressed: () {
                users.put(User(
                    id: '',
                    name: 'teste',
                    email: 'aluno@senac.br',
                    avatarUrl: ''));
              },
            ),
...
```

- E em users.dart retire o comentário de notifyListeners():

```
...
    //alterar
    notifyListeners();
  }
```

Cadastro

- Se em user_list.dart fizermos ...

```
...
  @override
  Widget build(BuildContext context) {
    final Users users = Provider.of(context, listen: false);
...
```

... não haverá mudança na tela, mesmo que existam inclusões na lista e, caso alguma inclusão já tenha sido feita, esta pode ser desconsiderada da renderização em tela, mantendo apenas a lista original, isto é, '*mockada*' inicialmente.
Bom, voltemos às inclusões retirando esta configuração e o que foi definido na função vazia do botão *add* em user_list.dart.

Sobre *provider*, existe um bom material aqui: https://pub.dev/packages/provider

!Cadastro

- Em users.dart renderize o código a seguir:

```
...
  void put(User user) {
    if (user == null) {
      return;
    }

    if (user.id != null &&
        user.id.trim().isNotEmpty &&
        _items.containsKey(user.id)) {
      _items.update(user.id, (_) => user);
    }
...
```

Controle sobre a UI: exemplo básico

1

```dart
import 'package:flutter/material.dart';

void main() => runApp(MyApp());

class MyApp extends StatelessWidget {
  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter UI Controls',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
      home: MyHomePage(title: 'Flutter UI Controls Home Page'),
    );
  }
}

class MyHomePage extends StatefulWidget {
  final String title;
  const MyHomePage({Key? key, required this.title}) : super(key: key);

  @override
  _MyHomePageState createState() => _MyHomePageState();
}
```

Controle sobre a UI: exemplo básico

```
class _MyHomePageState extends State<MyHomePage> {
  final GlobalKey<ScaffoldState> _scaffoldKey = new GlobalKey<ScaffoldState>();
  String? _dropdownButtonValue = 'One';
  String? _popupMenuButtonValue = 'One';
  bool? _checkboxValue = true;
  String? _radioBoxValue = 'One';
  double _sliderValue = 10;
  bool _switchValue = false;

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      key: _scaffoldKey,
      appBar: AppBar(
        title: Text(widget.title),
      ),
      body: Container(
        padding: EdgeInsets.only(left: 10, right: 10),
        child: SingleChildScrollView(
          child: Column(
            mainAxisAlignment: MainAxisAlignment.start,
            crossAxisAlignment: CrossAxisAlignment.start,
            children: <Widget>[
              // Buttons
              Column(
                mainAxisAlignment: MainAxisAlignment.start,
                crossAxisAlignment: CrossAxisAlignment.start,
```

Controle sobre a UI: exemplo básico


```
children: <Widget>[
  _GroupText('Buttons'),
  ButtonBar(
    alignment: MainAxisAlignment.start,
    mainAxisSize: MainAxisSize.min,
    buttonMinWidth: 200,
    buttonHeight: 30,
    buttonAlignedDropdown: true,
    layoutBehavior: ButtonBarLayoutBehavior.padded,
    buttonPadding: EdgeInsets.symmetric(vertical: 10),
    children: <Widget>[
      RaisedButton(
        onPressed: () => _showToast('Clicked on RaisedButton'),
        child: Text('Raised Button'),
      ),
      FlatButton(
        onPressed: () => _showToast('Clicked on FlatButton'),
        child: Text('Flat Button'),
      ),
      OutlineButton(
        onPressed: () => _showToast('Clicked on OutlineButton'),
        child: Text('OutlineButton'),
      ),
      Row(
        children: <Widget>[
          Text('IconButton: '),
          IconButton(
```

Controle sobre a UI: exemplo básico


```
        onPressed: () =>
            _showToast('Clicked on IconButton'),
        icon: Icon(Icons.build),
      ),
    ],
  ),
  Row(
    children: <Widget>[
      Text('DropdownButton: '),
      DropdownButton(
          value: _dropdownButtonValue,
          onChanged: (String? value) {
            setState(() {
              _dropdownButtonValue = value;
            });
            _showToast(
                'Changed value of dropdown button to $value');
          },
          items: ['One', 'Two', 'Three', 'Four']
              .map<DropdownMenuItem<String>>(
                  (String value) =>
                      DropdownMenuItem<String>(
                        value: value,
                        child: Text(value),
                      ))
              .toList()),
    ],
  ),
```

Controle sobre a UI: exemplo básico


```
Row(
  children: <Widget>[
    Text('PopupMenuButton: '),
    Text(_popupMenuButtonValue!),
    PopupMenuButton<String>(
        onSelected: (String result) {
          setState(() {
            _popupMenuButtonValue = result;
          });
          _showToast(
              'Selected \'$result\' item on PopupMenuButton');
        },
        itemBuilder: (BuildContext context) =>
            <PopupMenuEntry<String>>[
              const PopupMenuItem<String>(
                value: 'One',
                child: Text('One'),
              ),
              const PopupMenuItem<String>(
                value: 'Two',
                child: Text('Two'),
              ),
              const PopupMenuItem<String>(
                value: 'Three',
                child: Text('Three'),
              ),
```

Controle sobre a UI: exemplo básico


```
                      const PopupMenuItem<String>(
                        value: 'Four',
                        child: Text('Four'),
                      ),
                    ]),
              ],
            ),
          ],
        )
      ],
    ),
    _SpaceLine(),
    // Checkbox
    Column(
        mainAxisAlignment: MainAxisAlignment.start,
        crossAxisAlignment: CrossAxisAlignment.start,
        children: <Widget>[
          _GroupText('Checkbox'),
          Row(
            children: <Widget>[
              Text('Simple checkbox'),
              Checkbox(
                  value: _checkboxValue,
                  onChanged: (bool? newValue) {
                    setState(() {
                      _checkboxValue = newValue;
                    });
```

Controle sobre a UI: exemplo básico

```
                    _showToast(
                        'Changed value of checkbox to $_checkboxValue');
                  }),
            ],
          )
        ]),
    _SpaceLine(),
    // Radio[Box]
    Column(
      mainAxisAlignment: MainAxisAlignment.start,
      crossAxisAlignment: CrossAxisAlignment.start,
      children: <Widget>[
        _GroupText('Radio[Box]'),
        RadioListTile(
            title: const Text('One'),
            value: 'One',
            groupValue: _radioBoxValue,
            onChanged: (String? value) {
              setState(() {
                _radioBoxValue = value;
              });
              _showToast('Changed value of Radio[Box] to $value');
            }),
        RadioListTile(
            title: const Text('Two'),
            value: 'Two',
            groupValue: _radioBoxValue,
            onChanged: (String? value) {
```

Controle sobre a UI: exemplo básico


```
                setState(() {
                  _radioBoxValue = value;
                });
                _showToast('Changed value of Radio[Box] to $value');
              }),
          RadioListTile(
              title: const Text('Three'),
              value: 'Three',
              groupValue: _radioBoxValue,
              onChanged: (String? value) {
                setState(() {
                  _radioBoxValue = value;
                });
                _showToast('Changed value of Radio[Box] to $value');
              }),
          RadioListTile(
              title: const Text('Four'),
              value: 'Four',
              groupValue: _radioBoxValue,
              onChanged: (String? value) {
                setState(() {
                  _radioBoxValue = value;
                });
                _showToast('Changed value of Radio[Box] to $value');
              }),
        ],
      ),
```

Controle sobre a UI: exemplo básico


```
_SpaceLine(),
// Slider
Column(
  mainAxisAlignment: MainAxisAlignment.start,
  crossAxisAlignment: CrossAxisAlignment.start,
  children: <Widget>[
    _GroupText('Slider'),
    Slider(
      value: _sliderValue,
      onChanged: (newValue) {
        setState(() {
          _sliderValue = newValue;
        });
      },
      min: 0,
      max: 100,
      divisions: 50,
      label: _sliderValue.toInt().toString(),
    ),
  ],
),
_SpaceLine(),
// Switch
Column(
  mainAxisAlignment: MainAxisAlignment.start,
  crossAxisAlignment: CrossAxisAlignment.start,
  children: <Widget>[
    _GroupText('Switch'),
```

Controle sobre a UI: exemplo básico


```
        Row(
          children: <Widget>[
            Text('Simple switch'),
            Switch(
              value: _switchValue,
              onChanged: (newValue) {
                setState(() {
                  _switchValue = newValue;
                });
                _showToast('Changed value of Switch to $newValue');
              },
            )
          ],
        ),
      ],
    ),
    _SpaceLine(),
    // TextField
    Column(
      mainAxisAlignment: MainAxisAlignment.start,
      crossAxisAlignment: CrossAxisAlignment.start,
      children: <Widget>[
        _GroupText('TextField'),
        TextField(
          obscureText: false,
          decoration: InputDecoration(
            border: OutlineInputBorder(),
            labelText: 'TextField',
```

Controle sobre a UI: exemplo básico


```
        ),
      ),
    ],
  ),
  _SpaceLine(),
  // DateTimePicker
  Column(
    mainAxisAlignment: MainAxisAlignment.start,
    crossAxisAlignment: CrossAxisAlignment.start,
    children: <Widget>[
      _GroupText('DateTimePicker'),
      OutlinedButton(
        onPressed: () {
          showDatePicker(
            context: context,
            initialDate: DateTime.now(),
            firstDate: DateTime(1970),
            lastDate: DateTime.now(),
          ).then((value) {
            _showToast('Selected date $value');
          });
        },
        child: Text('Open DatePicker'),
      ),
      OutlinedButton(
        onPressed: () {
          showTimePicker(
            context: context,
```

Controle sobre a UI: exemplo básico


```
                    initialTime: TimeOfDay.now(),
                  ).then((value) {
                    _showToast('Selected time $value');
                  });
                },
                child: Text('Open TimePicker'),
              ),
            ],
          ),
          SizedBox(
            height: 50,
          )
        ],
      ),
    ),
    floatingActionButton: FloatingActionButton(
      onPressed: () => _showToast('Clicked on float action button'),
      tooltip: 'Increment',
      child: Icon(Icons.add),
    ),
  );
}

void _showToast(String text) {}

RaisedButton({required void Function() onPressed, required Text child}) {}
```

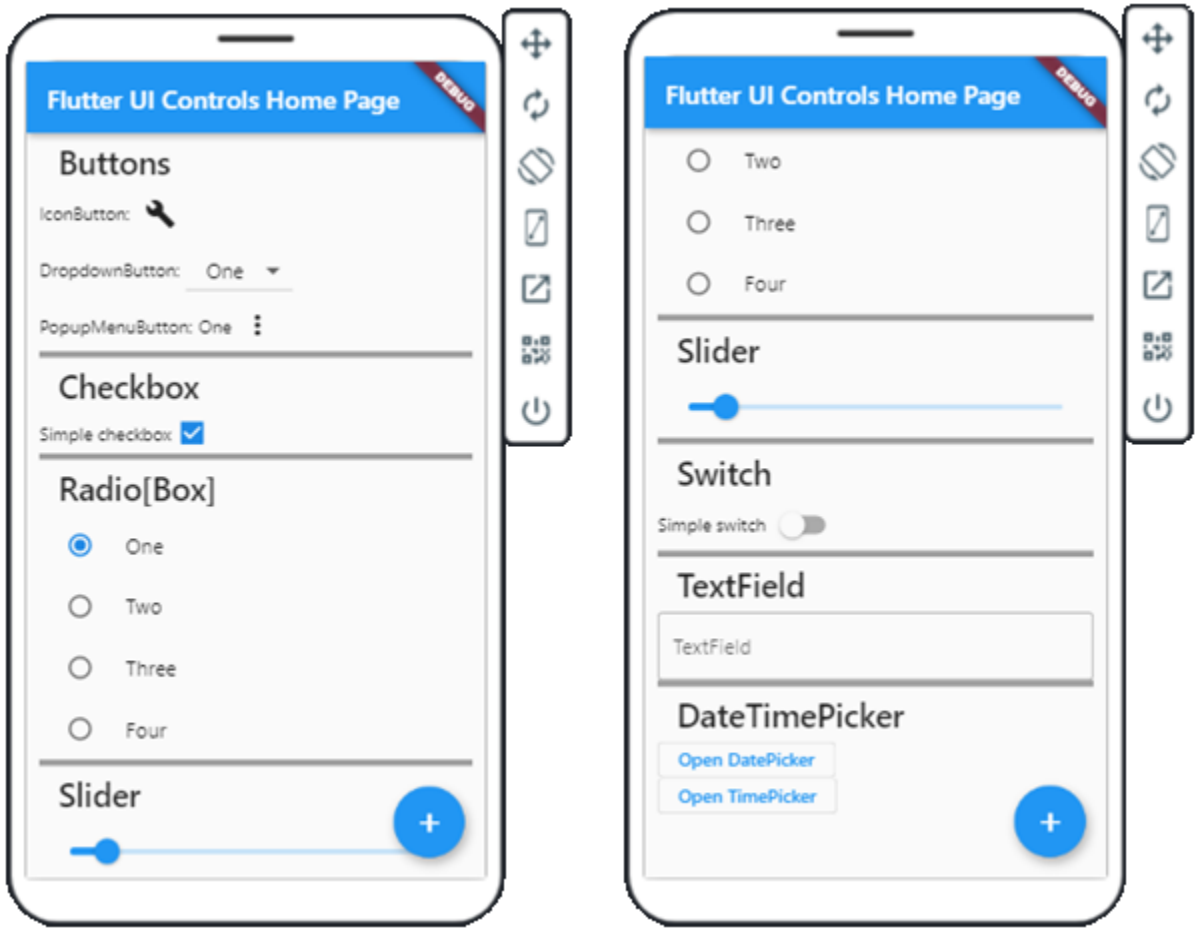
Controle sobre a UI: exemplo básico


```
  FlatButton({required void Function() onPressed, required Text child}) {}

  OutlineButton({required void Function() onPressed, required Text child}) {}
}

class _GroupText extends StatelessWidget {
  final String text;

  const _GroupText(this.text);

  @override
  Widget build(BuildContext context) {
    return Padding(
      padding: EdgeInsets.symmetric(vertical: 5, horizontal: 15),
      child: Text(
        text,
        style: TextStyle(fontSize: 25, fontWeight: FontWeight.w500),
      ),
    );
  }
}
```

Controle sobre a UI: exemplo básico


```dart
class _SpaceLine extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return SizedBox(
      height: 5,
      child: Container(
        color: Colors.grey,
      ),
    );
  }
}
```

Controle sobre a UI: exemplo básico



Navegação: exemplo básico

```dart
import 'package:flutter/material.dart';

void main() => runApp(MyApp());

class MyApp extends StatelessWidget {
  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Navigation Over Screens',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
//      home: MainPage(),

      // Declare routes
      routes: {
        // Main initial route
        '/': (context) => MainPage(),
        // Second route
        '/second': (context) => SecondPage(),
      },
      initialRoute: '/',
    );
  }
}
```

1

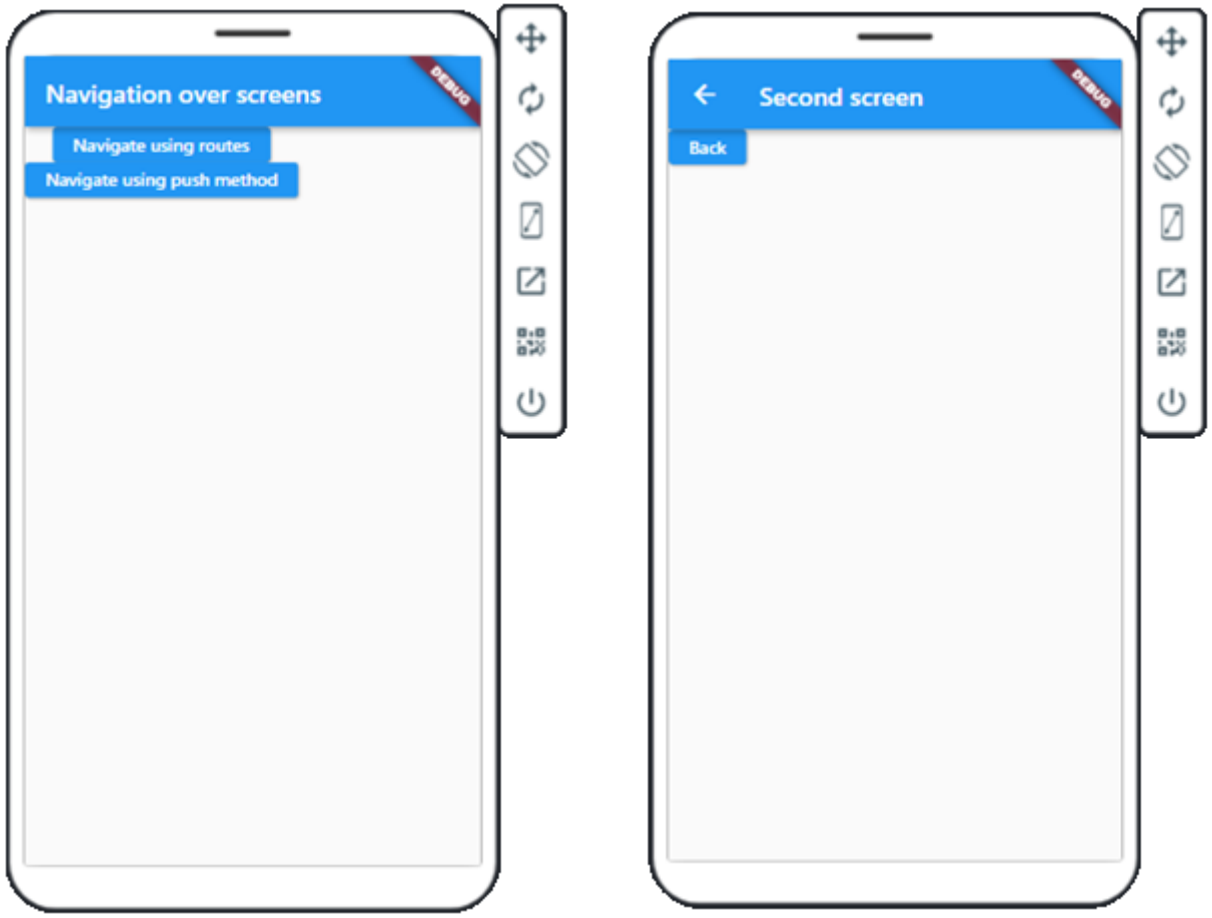
Navegação: exemplo básico

```
class MainPage extends StatelessWidget {
  @override
  Widget build(BuildContext context) =>
      Scaffold(
        appBar: AppBar(
          title: Text('Navigation over screens'),
        ),
        body: Container(
          child: Column(
            children: <Widget>[
              // Navigate using declared route name
              ElevatedButton(
                onPressed: () => Navigator.pushNamed(context, '/second'),
                child: Text('Navigate using routes'),
              ),
              // Navigate using simple push method
              ElevatedButton(
                onPressed: () =>
                    Navigator.push(
                      context,
                      MaterialPageRoute(builder: (context) => SecondPage()),
                    ),
```

Navegação: exemplo básico

```
                child: Text('Navigate using push method'),
              )
            ],
          ),
        ),
      );
}

class SecondPage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Second screen'),
      ),
      body: Container(
        child: ElevatedButton(
          onPressed: () => Navigator.pop(context),
          child: Text('Back'),
        ),
      ),
    );
  }
}
```

Navegação: exemplo básico



Dúvidas!?


KIRIBATI
(Gilbert Group)
Tarawa
KIRIBATI
(Phoenix Group)
KIRIBATI
(Line Group)
KIRIBATI BIOREGIONS