Layout da Agenda (App Lista de contatos)

//////// NO ARQUIVO ***main.dart***

Antes de tudo, importa-se o **material.dart** que é a classe que o flutter usa para os Widgets
Também tem os outros imports das outras classes que a gente vai criar

```
import 'package:flutter/material.dart';
import 'package:listadecontatos/widgets/details.dart';
import 'package:listadecontatos/widgets/list.dart';
```

Em Dart a função main( ) é onde o programa vai rodar: **void main( ) => runApp(MyApp());**
é a função main chamando o construtor do MyApp(). Note que usei o formato de "fat arrow", que é quando se chama uma expressão ou função/metodo que ocupa apenas uma linha com o "=>".

A classe MyApp herda de StatelessWidget, ou seja, ela própria é uma StatelessWidget, que é uma Widget que não muda de estado. O método que ela herda é o **build** que deve ser sobreescrito (@override) da seguinte forma:

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

class MyApp extends StatelessWidget {
  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Agenda de Contatos',
      home: Home(),
    ); // MaterialApp
  }
}
```

Retornando um MaterialApp contendo o título do App e setando a página inicial (home).

Logo após, criamos a classe Home que é chamada no MaterialApp como página inicial. Ela é uma StatefulWidget, ou seja, seu estado é mutável, e é um State (estado) que o muda. Por isso, herda o método **createState** que chama o State de Home (HomeState).

```
class Home extends StatefulWidget {
    @override
    createState() => HomeState();
}
```

A classe HomeState herda de State<Home>, isto é, é um estado da classe Home.
Herda o método build que constrói o app. Ela retorna um **Scaffold**, que é um Layout básico. O Scaffold contém o **AppBar**, que é a barra superior do aplicativo, contendo o título da página atual e, em alguns casos, alguns ícones e botões.

```
class HomeState extends State<Home> {
    @override
    Widget build(BuildContext context) {
        return Scaffold(
            appBar: AppBar(
                title: Text("Lista de contatos"),
            ), // AppBar
            body: ContactList(),
            floatingActionButton: FloatingActionButton(
                child: Icon(Icons.person_add),
                onPressed: () {
                    _goToContact();
                    print("FloatButton");
                },
            ), // FloatingActionButton
```

Assim, o Scaffold também possui o parâmetro body, que é o corpo do layout em si. No nosso tutorial, chamamos **ContactList()**, construtor da classe que criaremos mais adiante para listar os contatos. FloatingActionButton é o botão "flutuante" que tem a função de adicionar contatos. É possível setar ícones para os botões com os materiais Icon e Icons. No parametro onPressed é passada a página do app em que será redirecionado (_goToContact()) faz isso).

Ainda na construção do Scaffold, criamos um BottomNavigationBar com dois Items. (Uma barra de navegação inferior, abaixo do body do Scaffold).

```
            bottomNavigationBar: BottomNavigationBar(
                items: <BottomNavigationBarItem>[
                    BottomNavigationBarItem(
                        icon: Icon(Icons.person),
                        title: Text("Todos")
                    ), // BottomNavigationBarItem
                    BottomNavigationBarItem(
                        icon: Icon(Icons.star),
                        title: Text("Favoritos"),
                    ) // BottomNavigationBarItem
                ] // <BottomNavigationBarItem>[]
            ), // BottomNavigationBar
        ); // Scaffold
    }
```

Criamos o método **_goToContact()** que nos redireciona para a página de detalhes de contato que criaremos logo à frente no tutorial. Ele usa o "stack" de páginas Navigator empurrando a classe Detail.

```
  void _goToContact() {
    Navigator.push(context, MaterialPageRoute(builder: (context) => Detail()));
  }
}
```

//////// NO ARQUIVO ***list.dart***

Criamos então uma nova classe chamada **ContactList**, que listará os contatos no body do Scaffold da página inicial. Ela é uma StatefulWidget, então seu estado irá variar.

```
import 'package:flutter/material.dart';
import 'package:listadecontatos/models/contact.dart';
import 'package:listadecontatos/widgets/details.dart';

class ContactList extends StatefulWidget {
    @override
    createState() => ContactState();
}
```

Criamos então o estado **ContactState** da nossa ContactList. (State<ContactList> diz que ele é um estado de ContactList).
Criamos uma lista de contatos, uma estrutura de dados que irá armazenar temporariamente e em tempo de execução os contatos existentes e adicionados. Então, no construtor de ContactState nós populamos, apenas para testes, algumas contatos (A classe **Contact** também será criada mais à frente, ela é um model, ou seja, uma classe que serve como tipo de objeto para nosso app).
Note que criamos a lista **_contacts** com um _ (underline) na frente do nome da variável. Isso porque em Dart underlines na frente do nome das variáveis ou constantes significa que são privadas e somente esta atual classe tem acesso.

```
class ContactState extends State<ContactList>{
    List<Contact> _contacts = List<Contact>();

    ContactState() {
        _contacts.add(Contact("Teste 1", "12345678", "teste1@teste.com"));
        _contacts.add(Contact("Teste 2", "5242364", "teste2@teste.com"));
        _contacts.add(Contact("Teste 3", "98521452", "teste3@teste.com"));
    }
```

No build de ContactState chamamos o método **listar**. Este método lista (obviamente) os contatos dentro de uma **ListView**. Ele retorna um **Container** contendo como child uma ListView. Quando se programa usando Flutter, alguns materiais (componentes gráficos) possuem nenhuma, uma ou mais *child/children* que são componentes filhos e estão dentro dos componentes que os chamam. Uma forma fácil de criar uma ListView e populá-la é usando o método **builder**. Ele precisa dos dois parâmetros: **itemCount**, que conta a quantidade de itens da ListView; e **itemBuilder** que constrói os itens (chamando _buildTile).

```
@override
Widget build(BuildContext context) {
    return listar();
}

Widget listar() {
    return Container(
        child: ListView.builder(
            itemCount: _contacts.length,
            itemBuilder: (context, i) {
                //if(i.isOdd) Divider();
                return _buildTile(_contacts[i]);
            },
        ) // ListView.builder
    ); // Container
}
```

O método privado (lembre-se do _ underline) **_buildTile** retorna um modelo de **ListTile**, ou seja, como cada "tile" (ou contato) será feito: o título do contato e a função que ele fará quando o usuário tocar nele, **onTap.** OnTap dará um "push" na stack de navegação de páginas Navigator, dando a rota para os detalhes do contato.

```
    Widget _buildTile(Contact element) {
        return ListTile(
            title: Text("${element.name}"),
            onTap: () {
                Navigator.push(context, MaterialPageRoute(
                    builder: (context) => Detail(contact: element)
                )); // MaterialPageRoute
            }
        ); // ListTile
    }
}
```

//////// NO ARQUIVO ***detail.dart***

Criamos então a classe **Detail** (que é StatelessWidget). Ela nos dará os detalhes das informações de cada contato.
TextEditingController controla as ações de input nas caixas de texto para criação e visualização das informações do contato. No construtor da classe inicializamos os TextEditingControllers com os contatos e em cada campo com os atributos respectivos dos contatos “name”, “email” e “tel”, passando para os parametros *text* de TextEditingController as informações.

(Nota: em Dart 2 o operador “**?.**” designa uma condição e atribuição *in-line* que permite acessar os dados apenas se o objeto _contact **não estiver null**. Se estiver, o operador “**??**” nos condiciona que, se o valor é null, então agora será **“”** (string vazia).

```dart
import 'package:flutter/material.dart';
import 'package:listadecontatos/models/contact.dart';

class Detail extends StatelessWidget {
    Contact _contact;
    TextEditingController _controllerName;
    TextEditingController _controllerEmail;
    TextEditingController _controllerTel;

    Detail({contact}) {
        _contact = contact;
        _controllerName = TextEditingController(text: _contact?.name ?? "");
        _controllerEmail = TextEditingController(text: _contact?.email ?? "");
        _controllerTel = TextEditingController(text: _contact?.tel ?? "");
    }
```

| |
\/

Na build de Detail, retornamos um Scaffold, construído com os componentes já vistos e da mesma forma que anteriormente. Importante notar um parametro novo dentro do AppBar: **actions**. Ele nos permite criar ações (widgets) no AppBar. No nosso caso, criamos um botão com ícone que quando pressionado aciona o método privado **_addContact** (veremos logo à frente).
Então criamos o body com um Container contendo como child uma **Column** com vários children. Nos children são criados os **TextField**s onde serão digitados / mostrados os dados do contato e neles são atribuídos os controllers criados anteriormente. (Note que **padding: EdgeInsets.all(15.0)** aparece na construção do Container; é o ajuste de posicionamento).

```
@override
Widget build(BuildContext context) {
    return Scaffold(
        appBar: AppBar(
            title: Text("Detalhes do contato"),
            actions: <Widget>[
                IconButton(icon: Icon(Icons.done, color: Colors.white),
                    onPressed: () => _addContact(),) // IconButton
            ], // <Widget>[]
        ), // AppBar
        body: Container(
            padding: EdgeInsets.all(15.0),
            child: Column(
                children: <Widget>[
                    TextField(
                        decoration: InputDecoration(labelText: "Nome"),
                        maxLines: 1,
                        controller: _controllerName
                    ), // TextField
                    TextField(
                        decoration: InputDecoration(labelText: "Email"),
                        maxLines: 1,
                        controller: _controllerEmail
                    ), // TextField
                    TextField(
                        decoration: InputDecoration(labelText: "Telefone"),
                        maxLines: 1,
                        controller: _controllerTel
                    ) // TextField
                ], // <Widget>[]
            ), // Column
        ), // Container
    ); // Scaffold
}
```

| |

\/

Por fim, implementamos o método **_addContact**. Há uma verificação se o contato é nulo, e se for, cria um novo contato. Se não for, edita os dados do contato já existente. (Nota: **async** denota um método assíncrono).

```
    void _addContact() async {
        if(_contact == null) {
            _contact = Contact(_controllerName.text, _controllerName.text, _controllerEmail.text);
        } else {
            _contact.name = _controllerName.text;
            _contact.email = _controllerEmail.text;
            _contact.tel = _controllerTel.text;
        }
    }
}
```

//////// NO ARQUIVO ***contact.dart***

E então a classe **Contact**. É a classe modelo para o objeto que representa um contato, contendo os atributos name, tel e email e um construtor que já seta os atributos.

```
class Contact {
    String name;
    String tel;
    String email;

    Contact(this.name, this.tel, this.email);
}
```