Apps Multiplataforma com Flutter

DevMob

George Rappel & Patrick Sasso

UFRJ

Quem somos?

George Rappel

Patrick Sasso

- Membros do DevMob
- Alunos de Ciência da Computação
- Flutter desde Agosto/2018 (Beta)
- Experiência com Android (Java)

DevMob

UFRJ

DevMob :)

Grupo de interesse e extensão, criado por alunos do curso de Ciência da Computação em 2017, com o foco em aprender a desenvolver aplicativos móveis e compartilhar esse conhecimento.

- Projetos com
  - Flutter
  - Android Nativo (Java e Kotlin)
  - React Native, Ionic
  - iOS Nativo (Swift)

DevMob

UFRJ

1. O que é Flutter?

Flutter

- Framework de Desenvolvimento de Aplicativos Multiplataforma
- Multiplataforma? Híbrido?
- Mantido pelo Google
- Baseado em Dart
- Versão 1.0 lançada em Dezembro/2018

Flutter - Pontos positivos

- Boa performance
- Desenvolvimento prático e rápido
  - Widgets próprios, prontos pra uso
- Interface consistente entre plataformas (e versões do sistema operacional)
- Boa documentação com exemplos
- Equipe presente e ativa (vídeos, podcasts)
- Open Source

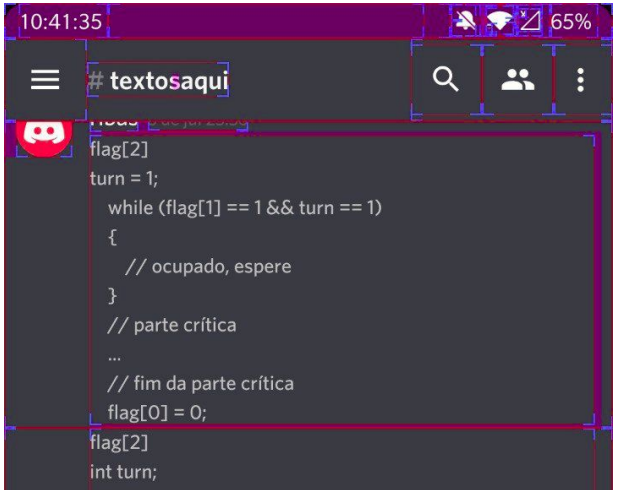
Flutter - Pontos negativos

- Framework Novo
  - Ainda tem poucas bibliotecas e conteúdo online
- Pouca separação entre código de interface x funcionalidade
- Interface diferente do app nativo
  - Pode fazer diferença em apps que usem códigos diferentes para cada funcionalidade/parte do app.

Flutter - Funcionamento

- Funciona como um framework para jogos, roda em uma engine própria, não usa os componentes nativos do Sistema.
- Usa GPU pra maior parte do processamento.

React Native

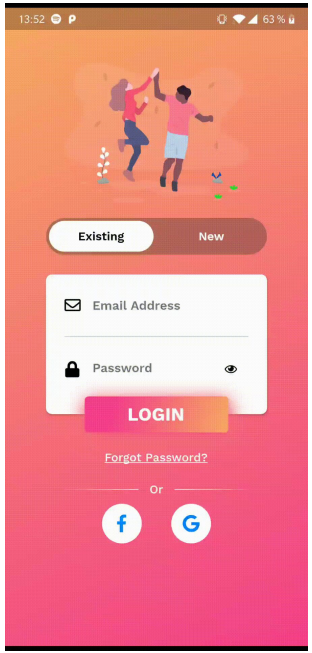


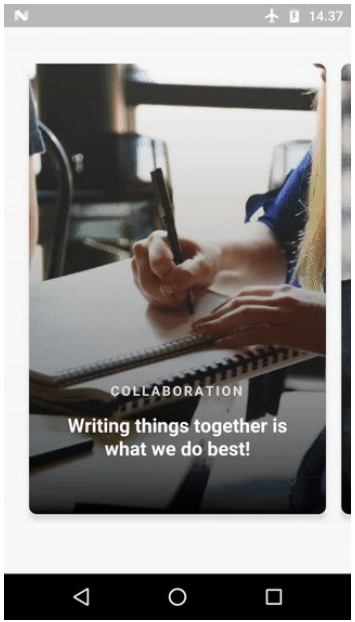
Flutter

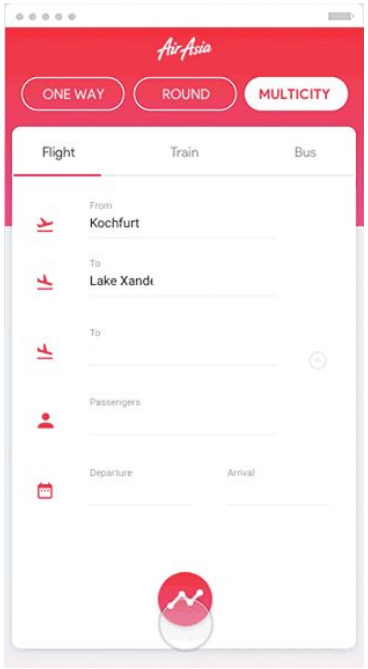
DevMob

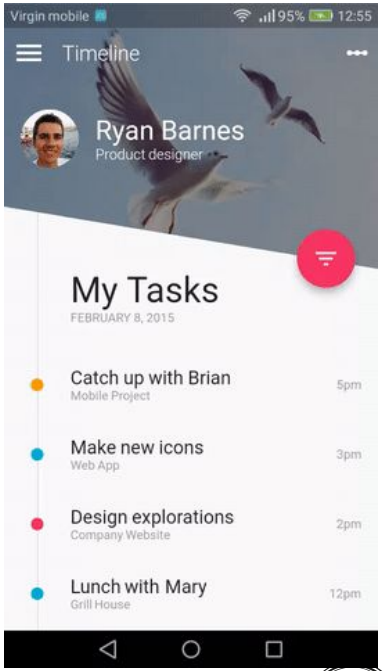
UFRJ

Flutter - Exemplos design









DevMob

UFRJ

2. Flutter vs. Demais

Flutter vs. Nativo

- Flutter
  - Maior praticidade e simplicidade
  - Desenvolvimento mais barato e rápido
- Nativo
  - Melhor performance
  - Facilidade de acesso ao recursos do dispositivo (Hardware e Funcionalidades externas)

Flutter vs. Híbridos

- Flutter
  - Dart
- React Native (Facebook)
  - Javascript/JSX
- Xamarin (Microsoft)
  - C#/XAML
- Ionic
  - Typescript/HTML/CSS

Flutter vs. React Native

- Flutter (Dart)
  - Melhor performance (sem pontes entre código e plataforma)
  - Interface consistente entre plataformas
  - Tamanho maior do app
- React Native (Javascript/JSX)
  - Grande comunidade Open Source
  - Performance média
  - Ainda em Beta (desde Março/2015)

Flutter vs. React Native

- Empresas abandonando React Native por problemas
  - Erros inexplicáveis e inconsistências
  - Imaturidade da plataforma
  - Tudo passa, menos o beta do React Native
- Organizações adotando Flutter
  - Nubank (também usa nativo e react native)
  - Alibaba
  - DevMob

3. Dart

Dart

- Linguagem de Programação Orientada a Objeto
- Pertence ao Google
- Lançada em Novembro/2013 (v1.0)
- Mobile, Web, Back-end, linha de comando, Desktop (preview)

Dart

Compilador online de dart:

https://dartpad.dartlang.org/

Dart - Sintaxe básica

```
void main(){
  int inteiro = 11;
  final double numero = 10.15;

  print(numero);

  var algo = "esta certo " + numero.toString();   // var? e String?
  print(algo);
}
```

Dart - Variáveis

- Tudo é Object (até números), tudo não inicializado é null
- Tipagem - exemplos:
  - var, num, int, double, String, bool
  - List<T>, DateTime, Set<T>, Object
- Modificadores
  - final - não pode ser alterada depois da inicialização
  - const - valor constante definido durante a compilação
  - Variável começando com _ (underline) - variável privada para uma classe/arquivo

Dart - Funções

- Tipagem opcional para variáveis, funções e parâmetros

```
int multiplica_2(int num){
    return num * 2;
}

// é equivalente a: (use com atenção)

multiplica_2(num){
    return num * 2;
}
```

Dart - Funções

- Parâmetros Opcionais e Nomeados (usamos muito em flutter)

```
int multiplica_2({int num}){
    return num * 2;          // Exceção se num não for passada/inicializada
}

int multiplica_2({int num=0}){ // Inicializa com 0 se não for passado
    return num * 2;
}

int multiplica_2({@required int num}){ // anotação p/ parâmetro obrigatório

    return num * 2;
}

multiplica_2(num: 3);  // Chamada da função
```

Dart - Funções

- Parâmetros Opcionais e Nomeados

```
int funcao(double p1, double p2, { double pf, @required int creditos }){
    // Contas
}

 // Parametros obrigatórios vem em ordem, opcionais em qualquer ordem

funcao(5, 7, creditos: 3, pf: 3);
```

Dart - Funções

- Funções curtas de uma linha (“Sintaxe de Seta”)

```
int multiplica_2(int num){
    return num * 2;
}

// é igual a:
int multiplica_2(int num) => num * 2; // Não precisa de return ou { }

// ou ainda:
multiplica_2(num) => num * 2;
```

Dart - Funções

- Funções Anônimas

```
List<int> minhaLista = [1, 2, 3];
// Função como parâmetro
minhaLista.forEach(
        (int n){
                print(n * 2);
        }
);
```

```
void funcao(int n){
        print(n * 2);
}
List<int> minhaLista = [1, 2, 3];
minhaLista.forEach( funcao );
minhaLista.forEach((n)=>funcao(n));
// Errado:
minhaLista.forEach(funcao(n));
minhaLista.forEach(funcao());
```

DevMob

UFRJ

Dart - Classes

- Criando uma classe

```
class Minicurso {
        String nome;
        int vagas;
        int inscritos;

        Minicurso(this.nome, this.vagas, {this.inscritos=0});

        @override
        String toString() => nome + " - Ocupacao: " + inscritos.toString()
                                + "/" + vagas.toString();
}
```

DevMob

UFRJ

Dart - Classes

- Usando uma classe

```
void main(){
    Minicurso curso = new Minicurso(“Flutter”, 12);
    Minicurso curso2 = Minicurso(“Kotlin vs Java”, 15, inscritos: 10);

    curso.inscritos = 9; // Acessa e altera um atributo do curso

    print( curso.toString() ); // Exibe “Flutter - Ocupacao: 9/12”
    print( curso2.toString() );
}
```

DevMob

UFRJ

Dart - Operadores Práticos

- Operadores para lidar com valores nulos
  - minicurso**?.**ocupacao //Acessa *ocupacao* apenas se *minicurso* != null. Evita possíveis excessões.

  - *minicurso.nome* ***??*** *“Sem nome”* //Retorna *minicurso.nome* se != null, se não, retorna a string “Sem nome”.

- Cascade - Permite várias operações em um mesmo objeto
  - minicurso..ocupacao = 10
    ..nome = “Um Minicurso”
    ..vagas = 15;

DevMob

UFRJ

Dart - Tour da Linguagem

https://dart.dev/guides/language/language-tour

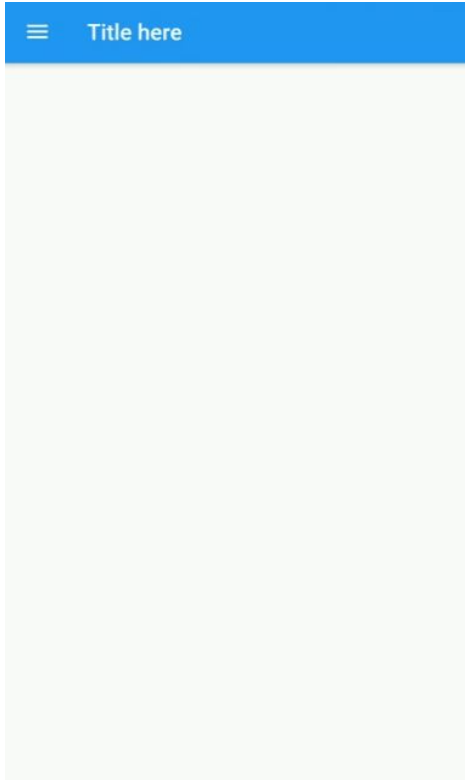
4. Flutter

Flutter

- Tudo são widgets
- Um Widget é um componente visual
  - Como uma tag do HTML ou uma view do XML do Android.
- Exemplos de widgets
  - Text (permite a utilização de textos no app)
  - Row e Column (estrutura o app em linhas e colunas)
  - Container (cria um retângulo que pode ser estilizado)
  - E muitos outros como appBar, Drawer, Icon, Button, Slider, Picker ...
- O aplicativo é criado fazendo uma combinação de widgets fazendo uma árvore de widgets.

Flutter - Exemplo

```
Scaffold(
  appBar: AppBar(
    title: Text("Title here"),
  ),
  drawer: Drawer(),
)
```

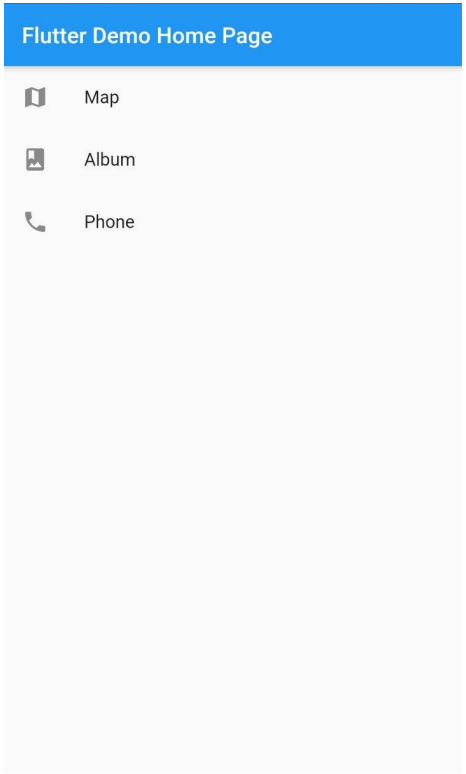


DevMob

UFRJ

Flutter - Exemplo 2

```
ListView(
  children: <Widget>[
    ListTile(
      leading: Icon(Icons.map)
      title: Text("Map"),
    ),
    ListTile(
      leading: Icon(Icons.photo_album)
      title: Text("Album"),
    ),
    ListTile(
      leading: Icon(Icons.phone)
      title: Text("Phone"),
    ),
  ],
)
```

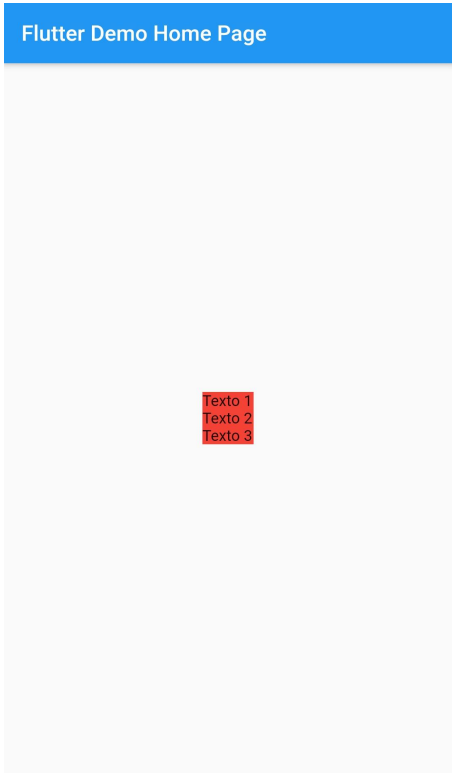


DevMob

UFRJ

Flutter - Exemplo 3

```
Container(
  color: Colors.red,
  child: Column(
    mainAxisSize: MainAxisSize.min,
    children: <Widget>[
      Text("Texto 1"),
      Text("Texto 2"),
      Text("Texto 3"),
    ],
  ),
)
```



DevMob

UFRJ

Flutter - Estados

- Os widgets podem ser classificados como Stateless ou Stateful widgets.
- O “Estado” é a situação/configuração dos elementos da tela em cada instante.

Flutter - Stateless Widget

Stateless - Como o nome diz, ele não tem estado interno, ou seja, uma vez desenhado na tela ele é imutável, a menos que eles sejam reconstruídos. Ex: botão, texto fixo.

Flutter - Stateful Widget

Stateful - São dinâmicos e tem estado, sendo assim podem ser facilmente modificados

Flutter - Estados exemplos

Um app de contador, se for Stateless, mesmo que a variável cresça quando se toca um botão , o número novo nao muda na tela.

Usando Stateful, a cada mudança você pode chamar a função *setState* para atualizar o número na tela.

Flutter - Criando um novo projeto

- Com o VS Code aberto pressione ctrl+alt+P.
- Digite Flutter e selecione “Flutter : New Project”.
- Digite o nome do projeto quando pedido.
- Escolha onde será salvo.

Flutter - Estrutura do projeto

- lib/ - Onde estarão todos os códigos do projeto.
- pubspec.yaml - Arquivo de configuração, onde são adicionadas bibliotecas externas entre outros, funciona como um packages.json da web.
- test/ - Onde são armazenados os códigos de testes automatizados.
- ios/ e android/ - Código específico para cada plataforma, configurações específicas para cada SO são feitas aqui.

Flutter

Live Coding

Flutter

Código do projeto:

https://github.com/DevMobUFRJ/workshop-flutter/

Referências

- Tour da Linguagem Dart: https://dart.dev/guides/language/language-tour
- Documentação do Flutter: https://flutter.dev/
- Tutorias da própria equipe: https://flutter.dev/docs/cookbook
- Widget da Semana: https://www.youtube.com/watch?v=b_sQ9bMltGU&list=PLjxrf2q8roU23XGwz3Km7sQZFTdB996iG
- Lista selecionada de widgets e libs boas: https://github.com/Solido/awesome-flutter
- Flutter/React Native/Xamarin/Ionic: https://apptunix.com/blog/frameworks-cross-platform-mobile-app-development
- Airbnb abandona React Native https://medium.com/airbnb-engineering/sunsetting-react-native-1868ba28e30a

Obrigado!

Patrick Sasso

GitHub: @pksasso

George Rappel

GitHub: @georgerappel

Comentários e Dúvidas:

devmob@dcc.ufrj.br

DevMob

UFRJ