Aula 03 - Introdução a Programação para Mobile

Computação para Dispositivos Móveis

Prof. Gustavo Custodio
gustavo.custodio@anhembi.br

Conteúdo de Mobile

- Introdução ao Flutter;
- Linguagem Dart;
- Manipulação de widgets;
- Navegação;
- Persistência;
- Consumo de APIs.

Aplicações Móveis

Aula 03 - Introducao a Programacao para Mobile

Aplicações Móveis

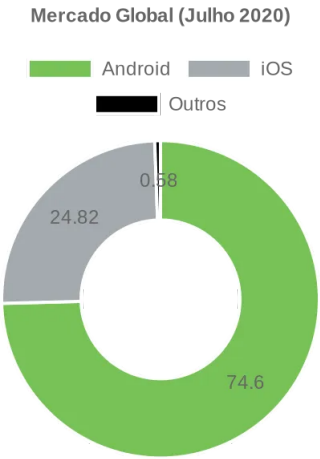
Aplicações Móveis

Aplicações Móveis

- Mercado global praticamente dividido em iOS e Android.
  - Fonte: *StatCounter*

Aplicações Móveis

- Mercado global praticamente dividido em iOS e Android.
  - Fonte: *StatCounter*

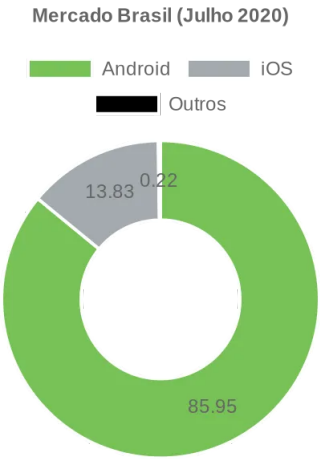


Aplicações Móveis

- E no Brasil?
  - Fonte: *StatCounter*

Aplicações Móveis

- E no Brasil?
  - Fonte: *StatCounter*



Publicando Apps

- **Android**:
  - 25 USD para comprar a conta de desenvolvedor;
  - Sem processo de revisão para a publicação;
  - Pode ser desenvolvido em Windows, Mac ou Linux.

Publicando Apps

- **Android**:
  - 25 USD para comprar a conta de desenvolvedor;
  - Sem processo de revisão para a publicação;
  - Pode ser desenvolvido em Windows, Mac ou Linux.
- **iOS**:
  - 100 USD por ano para manter a conta de desenvolvedor.
  - Processo de revisão rigoroso.
  - Só pode ser desenvolvido em um Mac.

Aula 03 - Introducao a Programacao para Mobile

Aplicações Nativas

Aplicações Nativas

- **Android**
  - Originalmente em **Java**.
  - Hoje permite o uso de **Kotlin**.
  - Desenvolvimento é feito dentro do **Android Studio**.

Aplicações Nativas

- **Android**
  - Originalmente em **Java**.
  - Hoje permite o uso de **Kotlin**.
  - Desenvolvimento é feito dentro do **Android Studio**.
- **iOS**
  - Originalmente programado em **ObjectiveC**.
  - Hoje permite o uso de **Swift**.
  - Desenvolvimento acontece na IDE **XCode**.

Problema

- Necessário manter dois apps.
  - 2 equipes programando em linguagens diferentes.
  - 2 *codebases*.
  - 2 ciclos de *releases* diferentes.
  - 2 vezes o número de *bugs*.

Solução: *Cross-Platform*

- Hoje, soluções que são compiladas em um app nativo:

Solução: *Cross-Platform*

- Hoje, soluções que são compiladas em um app nativo:
  - Xamarin (C#/F#)
  - React Native (JavaScript)
  - Flutter (Dart)

Solução: *Cross-Platform*

- Hoje, soluções que são compiladas em um app nativo:
  - Xamarin (C#/F#)
  - React Native (JavaScript)
  - Flutter (Dart)
- Embora a performance não seja a mesma de apps realmente nativos, a diferença é **imperceptível** na maioria dos casos.

Solução: *Cross-Platform*

- Hoje, soluções que são compiladas em um app nativo:
  - Xamarin (C#/F#)
  - React Native (JavaScript)
  - Flutter (Dart)
- Embora a performance não seja a mesma de apps realmente nativos, a diferença é **imperceptível** na maioria dos casos.
- Utilizaremos o **Flutter**.

Introdução ao Flutter

Aula 03 - Introducao a Programacao para Mobile

O que é Flutter?

O que é Flutter?

- Flutter é um kit de desenvolvimento de interfaces de usuário (UI).

O que é Flutter?

- Flutter é um kit de desenvolvimento de interfaces de usuário (UI).
  - Criado pelo Google em 2015.
  - Utiliza a linguagem de programação **Dart**.
  - Utilizado para desenvolvimento de aplicações para **múltiplas plataformas** (incluindo Android e iOS).

Vantagens

- Redução de tempo de desenvolvimento.
- Utilização da linguagem Dart, que une os melhores elementos de Java e JavaScript.
- Performance próxima de apps nativos.
- Leve quando comparado com outras soluções *Cross-Platform* (*Ionic*).

Desvantagens

- Tamanho dos apps.
- Muitos widgets aninhados.

Desvantagens

- Tamanho dos apps.
- Muitos widgets aninhados.

```
@override
Widget build(BuildContext context) {
  return Padding(
    child: Column(
      children: <Widget>[
        Row(
          children: <Widget>[
            Text('Here we go...'),
            Icon(Icons.format_indent_increase),
            Padding(
              child: Column(
                children: <Widget>[
                  Icon(Icons.photo),
                  Center(
                    child: Text('I can see my house from here!'),
```

Instalação

Aula 03 - Introducao a Programacao para Mobile

Instalação

- O Windows é o sistema operacional mais utilizado, então vamos realizar a instalação nele.

Instalação

- O Windows é o sistema operacional mais utilizado, então vamos realizar a instalação nele.
- O processo de instalação em outros Sistemas Operacionais está disponível em:
  - https://docs.flutter.dev/get-started/install

Instalação

- Baixe o arquivo e extraia o arquivo:
  - `flutter_windows_3.0.5-stable.zip`

Instalação

- Baixe o arquivo e extraia o arquivo:
  - `flutter_windows_3.0.5-stable.zip`
- Adicione o flutter nas variáveis de ambiente.

Instalação

- O Flutter oferece um recurso para validar a instalação.

Instalação

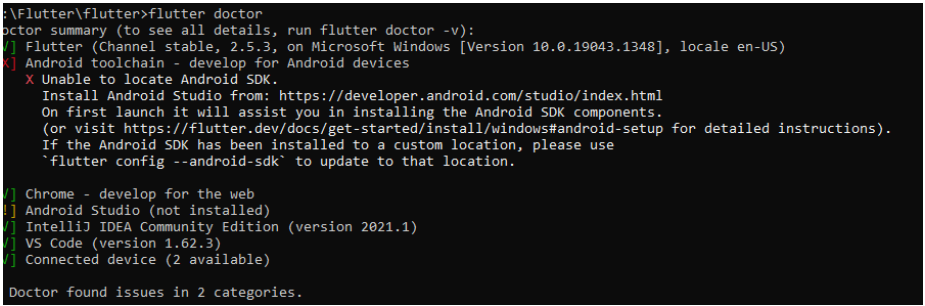
- O Flutter oferece um recurso para validar a instalação.
- Digite na linha de comando:
  - `flutter doctor`.
  - Isso mostrará problemas encontrados com a instalação do Flutter.

Instalação

- Se algum problema for encontrado, você verá uma tela como essa:

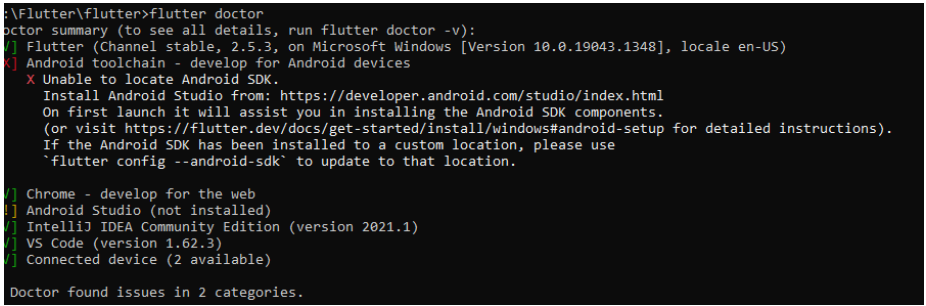
Instalação

- Se algum problema for encontrado, você verá uma tela como essa:



Instalação

- Se algum problema for encontrado, você verá uma tela como essa:



- Neste exemplo, foram encontrados dois problemas.

Instalação

- É necessário a instalação do Android Studio e do Visual Studio para utilizar o Flutter.
  - Não vamos utilizá-los por enquanto, só precisamos instalá-los.

Instalação

- É necessário a instalação do Android Studio e do Visual Studio para utilizar o Flutter.
  - Não vamos utilizá-los por enquanto, só precisamos instalá-los.
- Após completar a instalação, é necessário concordar com algumas licenças:
  - `flutter doctor --android-licenses`

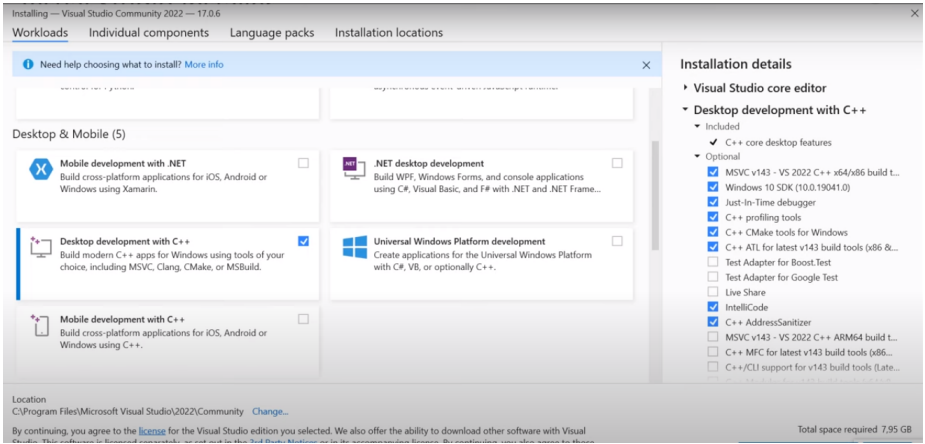
Instalação

- Durante a instalação do Visual Studio, marque a opção:
  - `SDK do Windows 10`;
  - `C++ CMake tools`;
  - `MSVC v142 - VS C++ build tools`.

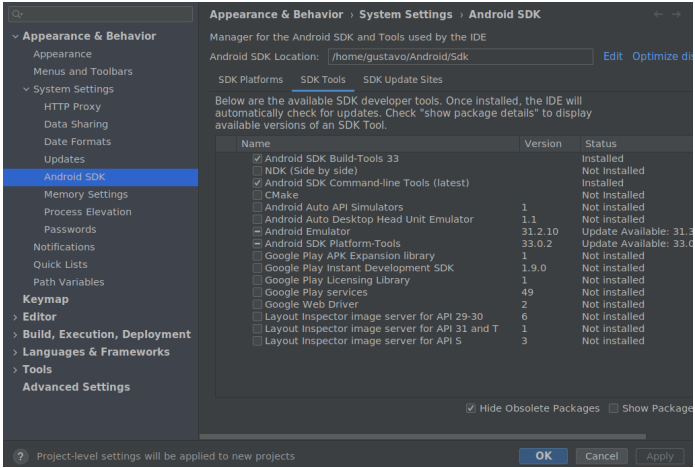
Instalação

- Durante a instalação do Visual Studio, marque a opção:
  - `SDK do Windows 10`;
  - `C++ CMake tools`;
  - `MSVC v142 - VS C++ build tools`.
- É possível que um problema seja encontrado ao concordar com as licenças do Android Studio.
  - Neste caso, abra o Android Studio e clique em Ferramentas → SDK Manager.
  - Procure pela opção SDK *Command-line Tools* e marque ela.

Instalação


Installing — Visual Studio Community 2022 — 17.0.6
Workloads
Individual components
Language packs
Installation locations
Need help choosing what to install? More info
Desktop & Mobile (5)
Mobile development with .NET
Build cross-platform applications for iOS, Android or Windows using Xamarin.
.NET desktop development
Build WPF, Windows Forms, and console applications using C#, Visual Basic, and F# with .NET and .NET Frame...
Desktop development with C++
Build modern C++ apps for Windows using tools of your choice, including MSVC, Clang, CMake, or MSBuild.
Universal Windows Platform development
Create applications for the Universal Windows Platform with C#, VB, or optionally C++.
Mobile development with C++
Build cross-platform applications for iOS, Android or Windows using C++.
Installation details
Visual Studio core editor
Desktop development with C++
Included
C++ core desktop features
Optional
MSVC v143 - VS 2022 C++ x64/x86 build t...
Windows 10 SDK (10.0.19041.0)
Just-In-Time debugger
C++ profiling tools
C++ CMake tools for Windows
C++ ATL for latest v143 build tools (x86 &...
Test Adapter for Boost.Test
Test Adapter for Google Test
Live Share
IntelliCode
C++ AddressSanitizer
MSVC v143 - VS 2022 C++ ARM64 build t...
C++ MFC for latest v143 build tools (x86...
C++/CLI support for v143 build tools (Late...
Location
C:\Program Files\Microsoft Visual Studio\2022\Community
Change...
By continuing, you agree to the license for the Visual Studio edition you selected. We also offer the ability to download other software with Visual
Total space required 7,95 GB

Instalação


Appearance & Behavior › System Settings › Android SDK
Appearance & Behavior
Appearance
Menus and Toolbars
System Settings
HTTP Proxy
Data Sharing
Date Formats
Updates
Android SDK
Memory Settings
Process Elevation
Passwords
Notifications
Quick Lists
Path Variables
Keymap
Editor
Build, Execution, Deployment
Languages & Frameworks
Tools
Advanced Settings
Manager for the Android SDK and Tools used by the IDE
Android SDK Location: /home/gustavo/Android/Sdk
Edit
SDK Platforms
SDK Tools
SDK Update Sites
Below are the available SDK developer tools. Once installed, the IDE will automatically check for updates. Check "show package details" to display available versions of an SDK Tool.
Name
Version
Status
Android SDK Build-Tools 33 Installed
NDK (Side by side) Not Installed
Android SDK Command-line Tools (latest) Installed
CMake Not Installed
Android Auto API Simulators 1 Not installed
Android Auto Desktop Head Unit Emulator 1.1 Not installed
Android Emulator 31.2.10 Update Available: 31.3
Android SDK Platform-Tools 33.0.2 Update Available: 33.0
Google Play APK Expansion library 1 Not installed
Google Play Instant Development SDK 1.9.0 Not installed
Google Play Licensing Library 1 Not installed
Google Play services 49 Not installed
Google Web Driver 2 Not installed
Layout Inspector image server for API 29-30 6 Not installed
Layout Inspector image server for API 31 and T 1 Not installed
Layout Inspector image server for API S 3 Not installed
Hide Obsolete Packages
Show Package
Project-level settings will be applied to new projects
OK
Cancel
Apply

Instalação

- Rode o `flutter doctor` novamente.

Instalação

- Rode o `flutter doctor` novamente.
- Se nenhum problema for encontrado, você verá uma tela como essa:

```
Doctor summary (to see all details, run flutter doctor -v):
[✓] Flutter (Channel stable, 3.0.5, on Manjaro Linux 5.15.59-1-MANJARO, locale
    en_US.utf8)
[✓] Android toolchain - develop for Android devices (Android SDK version 33.0.0)
[✓] Chrome - develop for the web
[✓] Linux toolchain - develop for Linux desktop
[✓] Android Studio (version 2021.2)
[✓] Connected device (2 available)
[✓] HTTP Host Availability

• No issues found!
```

Primeiro App

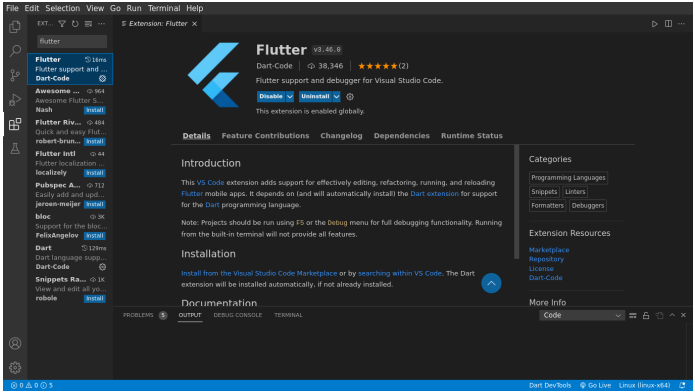
Aula 03 - Introducao a Programacao para Mobile

VSCode

- Utilizaremos o VSCode para criar nosso primeiro app.
- Procure pela extensão Flutter (Dart-code).

VSCode

- Utilizaremos o VSCode para criar nosso primeiro app.
- Procure pela extensão Flutter (Dart-code).



Primeiro App

- Para criar o primeiro app, abra uma aplicação de linha de comando:
  - escolha a pasta que deseja salvar seu projeto.

Primeiro App

- Para criar o primeiro app, abra uma aplicação de linha de comando:
  - escolha a pasta que deseja salvar seu projeto.
  - Em seguida, use o comando:
    - `flutter create meu-primeiro-app`
    - Isso criará um aplicativo base no diretório escolhido.

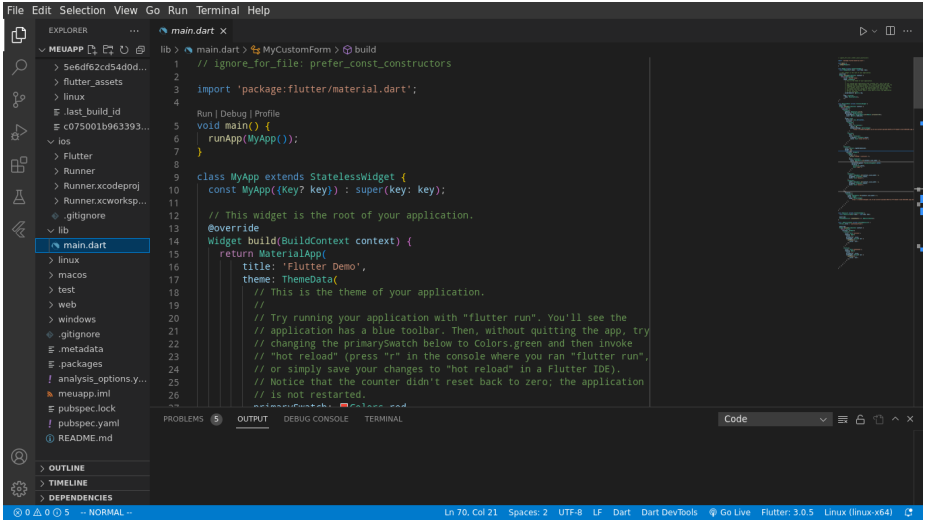
Primeiro App

- Abra a pasta do projeto usando o VSCode.
- Procure o arquivo `main.dart` dentro da pasta `lib`.

Primeiro App

- Abra a pasta do projeto usando o VSCode.
- Procure o arquivo `main.dart` dentro da pasta `lib`.
- Esse é o arquivo principal de nosso projeto e será o que modificaremos por enquanto.

Primeiro App



Primeiro App

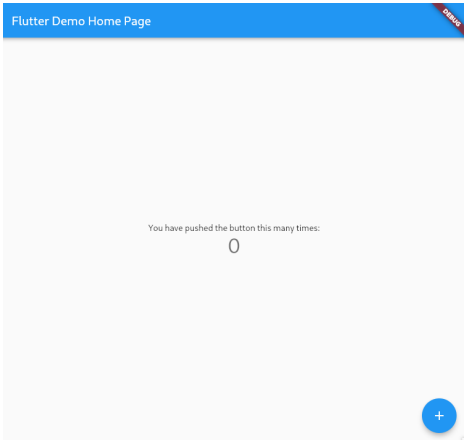
- Aperte F5 no VSCode para rodar o projeto.
  - Você também pode utilizar o comando `flutter run` dentro da pasta de algum projeto.

Primeiro App

- Aperte F5 no VSCode para rodar o projeto.
  - Você também pode utilizar o comando `flutter run` dentro da pasta de algum projeto.
- Por padrão, um app que conta quantas vezes você clicou em um botão é criado.

Primeiro App

- Aperte F5 no VSCode para rodar o projeto.
  - Você também pode utilizar o comando `flutter run` dentro da pasta de algum projeto.
- Por padrão, um app que conta quantas vezes você clicou em um botão é criado.



Primeiro App

- Vamos substituir todo o código gerado automaticamente no `main.dart`.

Primeiro App

- Vamos substituir todo o código gerado automaticamente no `main.dart`.
- Vamos começar escrevendo um *Hello World*.

Primeiro App

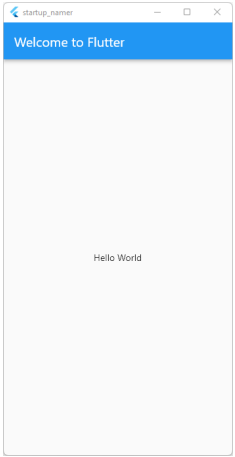
```dart
import 'package:flutter/material.dart';

void main() {
  runApp(const MyApp());
}
class MyApp extends StatelessWidget {
  const MyApp({super.key});
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Welcome to Flutter',
      home: Scaffold(
        appBar: AppBar(
          title: const Text('Welcome to Flutter'),
        ),
        body: const Center(
          child: Text('Hello World'),
        ),
      ),
    );
  }
}
```

Primeiro App

- Resultado:

Primeiro App

- Resultado:



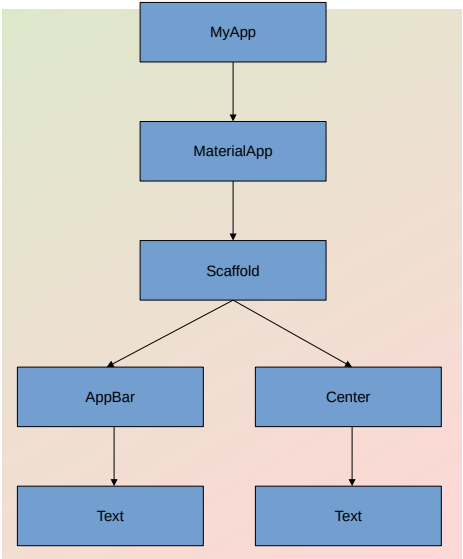
Entendendo o App

- Tudo em Flutter é um Widget!

Entendendo o App

- Tudo em Flutter é um Widget!
- `runApp(const MyApp())`
  - Essa linha inicializa o *framework* Flutter e instancia o `MyApp`.
  - A classe `Myapp` é um widget utilizado para preparar todos os dados globais e iniciar a árvore de widgets.
  - Os widgets que formam nosso app podem ser representados por uma árvore.

Entendendo o App



Entendendo o App

- A classe principal herda de *Stateless Widget*, ou seja, um Widget que não possui estado.
  - Utilizamos esse tipo de Widget quando não temos *inputs* do usuário e, dessa forma, não é necessário alterar algum estado.

Entendendo o App

- A classe principal herda de *Stateless Widget*, ou seja, um Widget que não possui estado.
  - Utilizamos esse tipo de Widget quando não temos *inputs* do usuário e, dessa forma, não é necessário alterar algum estado.
  - Para os outros casos, temos um `StatefulWidget`.

Entendendo o App

- O AppBar representa a barra azul que vemos no aplicativo.
  - É possível alterar a cor dessa barra.

Exercício

- Tente mudar a cor da `AppBar` utilizada.
- Adicione mais widgets `Text` em linhas diferentes.

Introdução à Linguagem Dart

Aula 03 - Introducao a Programacao para Mobile

Introdução ao Dart

Dart: uma Linguagem Familiar

- Linguagem de programação utilizada para desenvolvimento no Flutter.

Dart: uma Linguagem Familiar

- Linguagem de programação utilizada para desenvolvimento no Flutter.
- Criada pelo Google em 2011.

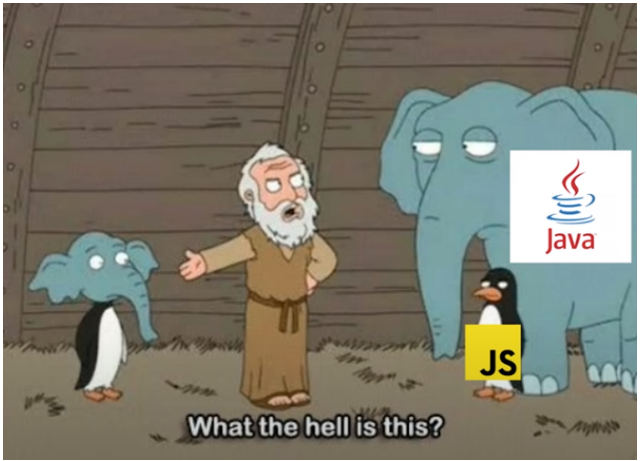
Dart: uma Linguagem Familiar

- Linguagem de programação utilizada para desenvolvimento no Flutter.
- Criada pelo Google em 2011.
- Tinha o objetivo de substituir o JavaScript como linguagem da Web.

Dart: uma Linguagem Familiar

- Linguagem de programação utilizada para desenvolvimento no Flutter.
- Criada pelo Google em 2011.
- Tinha o objetivo de substituir o JavaScript como linguagem da Web.
- Quando foi criada, buscava combinar a natureza dinâmica do JavaScript com o *design* baseado em classes do Java.
  - Lembra bastante o TypeScript.

Dart: uma Linguagem Familiar



Dart: uma Linguagem Familiar

- Características da Linguagem Dart:

Dart: uma Linguagem Familiar

- Características da Linguagem Dart:
  - Variáveis e constantes declaradas usando `var`, `final`, `const`.
  - Possibilidade de definir tipos para variáveis.
  - Classes e construtores.
  - Linguagem *null-safe*.

Linguagem Dart

- Há um *sandbox* da linguagem Dart disponível em:

Linguagem Dart

- Há um *sandbox* da linguagem Dart disponível em:
  - `https://dartpad.dev/`.
- Há também a opção do replit:
  - `https://replit.com/` .

Todos os programas em Dart são iniciados com o método `main` conforme mostrado abaixo:

Linguagem Dart

Todos os programas em Dart são iniciados com o método `main` conforme mostrado abaixo:

```
1  void main() {
2      print("Meu primeiro programa");
3  }
```

Variáveis

- Variáveis em Dart são declaradas definindo tipos, assim como a maioria das linguagens de programação:
  - `int umInteiro = 10;`

Variáveis

- Variáveis em Dart são declaradas definindo tipos, assim como a maioria das linguagens de programação:
  - `int umInteiro = 10;`
- No entanto, também é possível utilizar a palavra reservada `var` para inferir os tipos de variáveis:
  - `var outroInteiro = 20;`

Variáveis

- Assim como o Java, constantes são definidas usando a palavra `final`.
- E assim como o JavaScript, constantes são definidas usando o `const`.

Variáveis

- Assim como o Java, constantes são definidas usando a palavra `final`.
- E assim como o JavaScript, constantes são definidas usando o `const`.
- Quando usamos cada um então?

Variáveis

- A maior diferença entre `final` e `const` é que `const` é determinado durante a compilação e `final` é determinado durante a execução.

Variáveis

- A maior diferença entre `final` e `const` é que `const` é determinado durante a compilação e `final` é determinado durante a execução.
  - Há certas coisas que você não pode atribuir a uma `const`.
  - Exemplo: `Datetime.now()` não funciona com `const` porque é resolvido em tempo de execução.

Strings

- Strings em Dart são muito parecidas com Java.

Strings

- Strings em Dart são muito parecidas com Java.
  - Tipo `String` é escrito com letra maiúscula.
  - O método `contains` é utilizado para verificar se uma substring está contida.

Strings

```
1  void main() {
2    String nome = "Gustavo";
3
4    if (nome.contains("a")) {
5      print("Seu nome contém a letra a.");
6    }
7  }
```

Strings

- A linguagem não permite concatenação de tipos diferentes.

Strings

- A linguagem não permite concatenação de tipos diferentes.

```
1    void main() {
2      String nome = "Michael";
3      int idade = 20;
4
5      print(nome + " tem " + idade + " anos.");
6    }
```

- Encontraremos um erro ao tentar realizar essa operação.

Strings

- Variáveis podem ser impressas junto com Strings utilizando o $:

Strings

- Variáveis podem ser impressas junto com Strings utilizando o $:

```
1
2   void main() {
3     String nome = "Michael";
4     int idade = 20;
5
6     print("${nome} tem ${idade} anos.");
7   }
```

Strings

- Também é possível dispor o conteúdo de uma String em múltiplas linhas delimitando a String por aspas triplas (“”””””””).

Strings

- Também é possível dispor o conteúdo de uma String em múltiplas linhas delimitando a String por aspas triplas (“”””).

```
String textoMultiLinhas = """
  Lorem ipsum dolor sit amet, consetetur sadipscing elitr,
  sed diam nonumy eirmod tempor invidunt ut labore et
  dolore magna aliquyam erat, sed diam voluptua.
  At vero eos et accusam et justo duo dolores et ea rebum.
  Stet clita kasd gubergren, no sea takimata sanctus est
  Lorem ipsum dolor sit amet.
""";
print(textoMultiLinhas);
```

Leitura de Inputs

- No Dart Utilizamos o comando `stdin.readLineSync` para lermos o input de um usuário.

Leitura de Inputs

- No Dart Utilizamos o comando `stdin.readLineSync` para lermos o input de um usuário.
- Devemos importar o pacote `io` do Dart para conseguirmos realizar essa operação:
  - `import "dart:io";`

Leitura de Inputs

```dart
import "dart:io";

void main() {
  print("Digite seu nome de usuário:");
  String? usuario = stdin.readLineSync();
  print("Digite a sua senha:");
  String? senha = stdin.readLineSync();
}
```

Leitura de Inputs

- Para que serve o ponto de interrogação na frente do tipo?
  - Experimente removê-lo e veja o que acontece.

Leitura de Inputs

- Para que serve o ponto de interrogação na frente do tipo?
  - Experimente removê-lo e veja o que acontece.
- Dart é uma linguagem *null-safe*.
  - Ou seja, ela não permite que uma variável assuma valores nulos.
  - Em um input, é possível que a entrada do usuário seja nula.
  - Indicamos isso usando a interrogação.

Leitura de Inputs

- E como lemos inputs de outros tipos (int, double etc)?

Leitura de Inputs

- E como lemos inputs de outros tipos (int, double etc)?
  - `int.parse(stdin.readLineSync()!);`
  - `double.parse(stdin.readLineSync()!);`
  - Adicionamos o ! para obrigar o compilador a assumir que não haverá valores nulos.

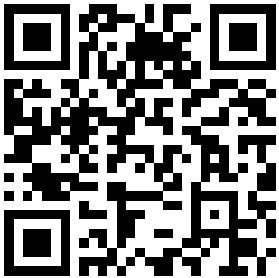
Exercício

- Escreva um programa que leia três inpus:
  - O primeiro nome;
  - O nome do meio;
  - O sobrenome;
- Mostra na tela a concatenação dos nomes.

Referências

- Documentação do flutter.
  `https://docs.flutter.dev/`.
  Acessado 15/08/2022.
- Faq do flutter.
  `https://docs.flutter.dev/resources/faq#what-is-flutter`.
  Acessado 15/08/2022.
- Simone Alessandria and Brian Kayfitz.
  Flutter Cookbook: Over 100 proven techniques and solutions for app development with Flutter 2.2 and Dart.
  Packt Publishing Ltd, 2021.

Conteúdo



https://gustavotcustodio.github.io/usabilidade.html

Obrigado

gustavo.custodio@anhembi.br