Desenvolvimento para dispositivos móveis

Introdução ao Flutter

Prof. Chauã Queirolo
https://github.com/chaua/

Sumário

O que é o Flutter?

Flutter é um **framework open-source** da Google

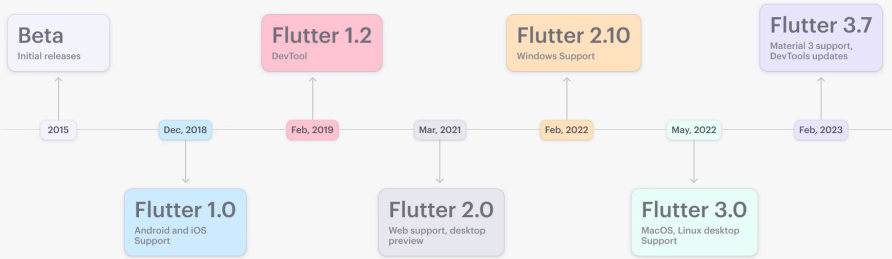
Focado no **desenvolvimento de interfaces gráficas (UI)** modernas e responsivas

Permite criar **aplicações nativas** com uma única base de código para:

- Android
- iOS
- Web
- Desktop (Windows, macOS, Linux)

Histórico

Criado pela **Google,** com foco em **atualizações frequentes** e suporte da **comunidade**

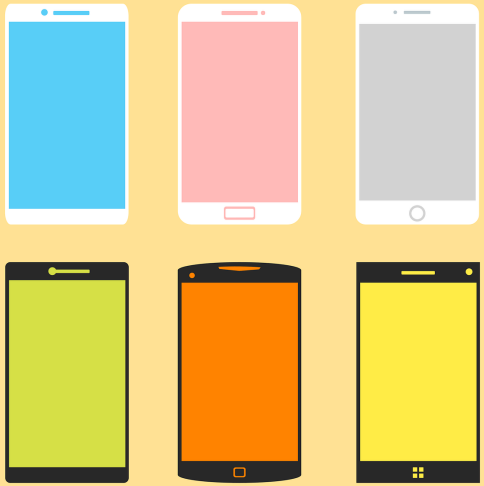


Por que o Flutter foi criado?

Resolver o **problema da fragmentação** entre plataformas móveis

Reduzir o esforço de manter **duas bases de código** (Android e iOS)

Criar uma alternativa aos frameworks existentes com **melhor desempenho, design unificado** e **fácil manutenção**



Multiplataforma

Antes do Flutter, havia duas opções principais:

- Desenvolvimento nativo (dois códigos diferentes)
- Frameworks híbridos (baixa performance e limitações visuais)

Flutter surgiu para oferecer o **melhor dos dois mundos:**

- **Produtividade**
- **Desempenho nativo**

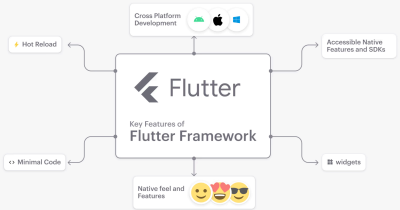
Benefícios para o Desenvolvedor

Desenvolvimento rápido com **Hot Reload**

Maior **consistência visual** entre plataformas

Utilização de uma linguagem moderna (Dart)

Extensa coleção de **widgets personalizáveis**



Flutter na Prática

Google Ads

Alibaba

Multiplataforma

Flutter permite desenvolver **um único código-fonte** para múltiplas plataformas

Menor custo de manutenção e **maior velocidade** de entrega

Experiência visual consistente entre plataformas

Ideal para **MVPs**, **protótipos** e **aplicações comerciais** completas

Alvo de execução

- 📱 Android
- 🍎 iOS
- 💻 Web
- 🖥️ Desktop (Windows, macOS e Linux)

Hot Reload

Recurso que permite **atualizar a interface e lógica do app em tempo real** sem reiniciar

Excelente para:

- Testes de interface
- Ajustes rápidos de layout
- Prototipação dinâmica

Aumenta significativamente a **produtividade do desenvolvedor**

Widgets

Em Flutter, **tudo é um widget:** Textos, botões, imagens, containers, etc

Dois tipos principais:

StatelessWidget – sem estado interno

StatefulWidget – com estado interno

Widgets são altamente personalizáveis e reutilizáveis

Widgets

Flutter oferece widgets prontos para **Material Design** (Android) e **Cupertino** (iOS)

Possibilidade de criar **interfaces personalizadas** além dos padrões nativos

Controle total sobre o comportamento e estilo da interface

Desempenho

Flutter usa o **motor gráfico Skia**, o mesmo usado no Chrome e Android

Skia permite que o Flutter desenhe **todos os pixels diretamente na tela**

Resultado: animações suaves, UI fluida e controle preciso sobre cada elemento gráfico

Vantagens DO Skia

Elimina a dependência dos componentes nativos da plataforma

Permite **mesma aparência e desempenho** em qualquer sistema

Ideal para apps com:

- Animações ricas
- Interfaces customizadas
- Alta responsividade

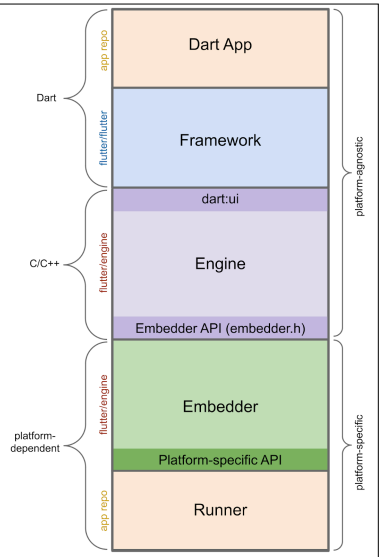
Arquitetura

A arquitetura do Flutter é composta por três camadas principais:

Framework
Interface e lógica

Engine
Renderização e animação

Embedder
Integração com sistema operacional



Framework

Framework escrito em **Dart** – onde os desenvolvedores atuam diretamente

Contém:

- Widgets (UI)
- Material e Cupertino
- Gestão de estado
- Navegação

Modular, orientado a objetos e altamente reativo

Engine

Implementada em **C++** para garantir **alto desempenho gráfico**

Responsável por:

- Renderização via Skia
- Composição de cenas
- Execução de animações
- Integração com o Dart runtime

Atua como **ponte entre o framework e o hardware gráfico**

Embedder

Específico para cada plataforma: Android, iOS, Windows, macOS, Linux

Responsável por:

- Inicializar a engine Flutter
- Exibir a tela (canvas)
- Lidar com entradas do usuário (toques, teclado)
- Acesso a APIs nativas (como câmera, GPS, etc)

Fluxo de Execução Básico

1. App inicia no Embedder, que carrega a Engine
2. Engine inicializa o ambiente Dart e executa o código do Framework
3. Framework cria widgets, compõe a UI e envia instruções para a Engine
4. Engine desenha a tela usando Skia
5. Reações a eventos do usuário retornam pelo Embedder e reiniciam o ciclo

Integração das Camadas

As camadas trabalham em conjunto:

- Framework → lógica do app
- Engine → renderização
- Embedder → ligação com sistema

Tudo isso é **encapsulado de forma transparente** para o desenvolvedor final

Ciclo de Vida de um App

Aplicações Flutter seguem um ciclo de vida controlado pelo WidgetsBinding

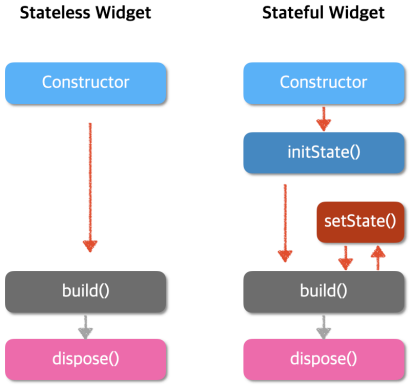
Principais fases:

1. Inicialização (main())
2. Build da árvore de widgets
3. Atualizações de estado
4. Redesenho e animações
5. Encerramento do app

Gerência automática de estado e renderização

Widget Lifecycle

Permite controle refinado do comportamento do app ao longo do tempo



Linguagem Dart

Dart é uma **linguagem de programação moderna e orientada a objetos**

Criada pela **Google** em 2011

Inicialmente pensada para web, mas hoje amplamente usada em apps, desktop e servidores

Suporta compilação **JIT** (tempo de execução) e **AOT** (compilação antecipada)

Características

Sintaxe clara e concisa, parecida com Java e JavaScript

Fortemente tipada, mas com inferência de tipos

Suporte a **programação assíncrona** com async/await

Compatível com orientação a objetos e **paradigma reativo**

Por que o Flutter usa Dart?

Dart foi escolhido por ser:

- Rápido para **prototipação** e **desenvolvimento interativo** (Hot Reload)
- Capaz de compilar para **código nativo (AOT)**
- Flexível e fácil de aprender para desenvolvedores JavaScript, Java, C#, etc
- Projetado com foco em **performance e previsibilidade**

Compilação AOT e JIT

JIT (Just-in-Time)

- Usado durante o desenvolvimento (permite Hot Reload)
- Compila rapidamente em tempo de execução

AOT (Ahead-of-Time)

- Usado para gerar executáveis otimizados
- Melhora o desempenho e o tempo de inicialização do app

Tipagem Forte com Flexibilidade

Dart possui **tipagem estática**, mas com suporte à inferência:

```dart
var nome = "Maria"; // automaticamente reconhecido como String
int idade = 25;
```

Isso reduz erros e melhora a legibilidade e manutenção do código

Assincronismo Simples e Poderoso

Dart lida muito bem com operações assíncronas usando Future

```dart
Future<void> carregarDados() async {
  var dados = await fetchData();
  print(dados);
}
```

Ideal para requisições HTTP, banco de dados local e animações

Vantagens do Dart no Flutter

Código limpo e organizado com **baixo overhead**

Facilita a criação de **interfaces reativas** com a árvore de widgets

Excelente suporte a ferramentas de desenvolvimento:

- VS Code
- Android Studio
- DevTools para análise de desempenho e memória

Widgets e Interface no Flutter

No Flutter, **tudo é um widget**: texto, botão, imagem, layout, animação

Widgets são os **blocos de construção** da interface

A UI é composta de uma **árvore de widgets** ("Widget Tree")

```dart
Text("Olá, Flutter!")
```


Vantagens da Arquitetura por Widgets

Modularidade

widgets são componentes reutilizáveis

Clareza

o código descreve exatamente o que será exibido na tela

Reatividade

alterações de estado atualizam automaticamente os widgets relevantes

Stateless Widget

Representa widgets que **não possuem estado interno**

São imutáveis: o conteúdo só muda se o widget for recriado

Utilizado para:

- Textos fixos
- Ícones
- Layouts estáticos

```dart
class MeuTexto extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Text("Sou um widget sem estado!");
  }
}
```

Stateful Widget

Utilizado quando o widget **possui estado interno**

Ideal para:

- Campos de formulário
- Contadores
- Interações dinâmicas

```
class Contador extends StatefulWidget {
  @override
  _ContadorState createState() ⇒ ContadorState();
}
```


Layouts Comuns: Row e Column

Row: organiza widgets horizontalmente

Column: organiza widgets verticalmente

Ambos aceitam propriedades como mainAxisAlignment e crossAxisAlignment

```
Column(
  children: [Text("Linha 1"), Text("Linha 2")],
)
```


Desenvolvimento

Para iniciar com Flutter, é necessário:

1. Instalar o **Flutter SDK**
2. Escolher uma **IDE compatível** (VS Code ou Android Studio)
3. Ter o **Android SDK** (ou Xcode no macOS para iOS)
4. Simuladores/emuladores ou dispositivos físicos para testes

Instalando o Flutter SDK

1. Acesse o site oficial: https://flutterdev
2. Baixe o SDK para seu sistema operacional
3. Configure a variável de ambiente PATH
4. Execute flutter doctor no terminal

flutter doctor

Comando essencial que verifica a instalação do ambiente

Exibe:

- Versão do Flutter
- Status do Dart
- SDKs (Android/iOS)
- Conectividade com editoresIDEs recomendadas

```
Doctor summary (to see all details, run flutter doctor -
[✓] Flutter (Channel stable, 3.7.0, on macOS 15.3.2 24D8
    en-BR)
[✓] Android toolchain - develop for Android devices (And
[!] Xcode - develop for iOS and macOS
    ✗ Xcode installation is incomplete; a full installat
      development.
      Download at: https://developer.apple.com/xcode/dow
      Or install Xcode via the App Store.
      Once installed, run:
        sudo xcode-select --switch /Applications/Xcode.a
        sudo xcodebuild -runFirstLaunch
[✓] Chrome - develop for the web
[!] Android Studio (version 2024.3)
    ✗ Unable to find bundled Java version.
[✓] IntelliJ IDEA Community Edition (version 2024.2.2)
[✓] Connected device (2 available)
[✓] HTTP Host Availability

! Doctor found issues in 2 categories.
```


IDE

Visual Studio Code

Leve, rápido, com extensões específicas para Flutter/Dart

Android Studio

Mais completo, com ferramentas de design visual e emulador integrado

Executando um App Flutter

Após iniciar um projeto, o app pode ser executado em:

- Emuladores Android/iOS
- Navegador (modo Web)
- Dispositivos físicos conectados

Use flutter run ou o botão "Run" da IDE

Simuladores e Emuladores

O Flutter detecta automaticamente dispositivos conectados

Para Android:

Use o **AVD Manager** no Android Studio

Para iOS:

Use o **Xcode Simulator** (somente no macOS)

É possível rodar diretamente no navegador com:

```
flutter run -d chrome
```

Conclusão

O Flutter se consolida como uma das ferramentas mais modernas para desenvolvimento multiplataforma.

Possui um ecossistema robusto, com alta produtividade, interface rica e ótimo desempenho.

Ideal para projetos que exigem rapidez no desenvolvimento, design responsivo e amplo alcance de plataformas.

Mesmo com algumas limitações, oferece um excelente custo-benefício técnico.

Referências

Flutter – Site Oficial
🔗 https://flutter.dev

Dart Language – Site Oficial
🔗 https://dart.dev

Pub.dev – Gerenciador de Pacotes Flutter
🔗 https://pub.dev

Documentação Oficial do Flutter
🔗 https://docs.flutter.dev