Flutter: Framework de UI para Android

Alexandre Costa[1] [48039]

Universidade de Évora, Évora, Portugal[1]

Abstract. No mundo mobile existem essencialmente dois sistemas operativos, Android e iOS. Para disponibilizar a mesma aplicação nos dois sistemas operativos, seria necessário escrever o código referente à aplicação em duas linguagens diferentes (Java/Kotlin e Swift). Com o surgimento do framework Flutter, criado pela Google em 2018, esse problema foi ultrapassado. O Flutter foi criado com o objetivo de desenvolver aplicações robustas e elegantes com um único código para diferentes sistemas operativos (Android e iOS).

Keywords: Flutter · Framework · Android · iOS.

1 Introdução

O desenvolvimento de aplicações mobile é uma área em constante crescimento e evolução. Em 2013, existiam 0.7 mil milhões de utilizadores Android. Em 2014 o sistema operativo Android tornou-se o mais popular, atingindo 1 bilião de utilizadores. Estima-se que no ano 2023 o número de utilizadores Android tenha atingido 3.6 biliões, cerca de 45% da população mundial.

Este constante crescimento criou a necessidade de ferramentas, frameworks, e bibliotecas que facilitem e otimizem a construção de aplicações Android. Dentro de uma enorme variedade, surgiu a necessidade dos chamados frameworks de interface de utilizador (UI) com o objetivo de construírem aplicações multiplataforma com aparência nativa. Estes frameworks têm como objetivo simplificar o processo de criação de interfaces de utilizador consistentes e de alta qualidade para aplicações multiplataforma.

Com a crescente necessidade de frameworks UI, o Flutter surge como uma solução inovadora e poderosa. Foi anunciado pela Google em 2015 com a alcunha "Sky" e posteriormente foi lançada a primeira versão estável em dezembro de 2018 já com o nome Flutter. O framework em questão faz uso da linguagem de programação Dart, linguagem também criada pelo Google.
O grande diferencial do Flutter está na agilidade no desenvolvimento de interfaces de aplicações, proporcionando uma ótima experiência de utilizador e um único código para os demais sistemas operativos.

O Flutter faz uso do seu próprio motor de renderização para desenhar widgets. Como afirma o Flutter.dev, "Flutter is different than most other options

for building mobile apps because it doesn't rely on web browser technology nor the set of widgets that ship with each device. Instead, Flutter uses its own high-performance rendering engine to draw widgets.".

2 O Flutter

O Flutter é construído com C, C++, Dart e Skia. A sua arquitetura de camadas facilita o desenvolvimento das aplicações.
A camada Engine é responsável pelas funcionalidades básicas do Flutter, como a renderização das interfaces.
O segundo nível, Embedder, serve como ponte entre o Flutter e o sistema operativo, permitindo o acesso a recursos nativos do aparelho, como a camara, bateria e o GPS.
Por fim, a camada Framework, é responsável pela construção da interface de utilizador e pela sua interação com a aplicação. É o nível mais utilizado pelos programadores, pois fornece as ferramentas necessárias para a construção das interfaces.

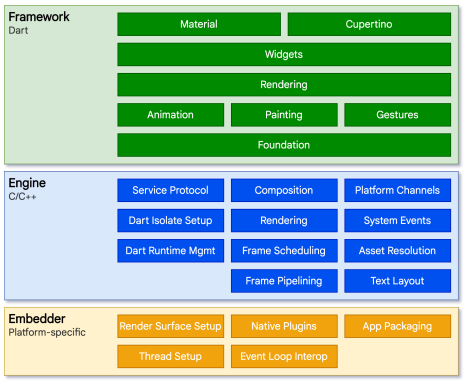

Fig. 1. Camadas da Arquitetura Flutter. Fonte: docs.flutter.dev

2.1 Importância dos Widgets

O Flutter utiliza uma abordagem inovadora para a construção de interfaces, utilizando widgets como blocos de construção básicos.
Os widgets são componentes visuais que definem a interface de uma aplicação.
Existem dois tipos principais de widgets:

– Stateless Widgets: São widgets estáticos que não armazenam dados que mudam. Por exemplo, um widget de texto que apresenta o título de um menu.

– Stateful Widgets: São widgets dinâmicos que podem atualizar a interface do utilizador em resposta a alterações. Por exemplo, um widget para uma caixa de seleção.

A grande vantagem por trás dos widgets é que podem ser facilmente reutilizados em diferentes partes do código e facilmente personalizáveis.

2.2 Linguagem Dart

Dart é uma linguagem de programação criada pela Google em 2011. A linguagem foi criada com o objetivo de substituir o JavaScript, o que não aconteceu. Atualmente, é a linguagem utilizada no framework Flutter.
Dart é uma linguagem de programação fortemente tipada, o que significa que todas as varáveis precisam de ter um tipo declarado. É uma linguagem orientada a objetos e com uma sintaxe bastante idêntica à linguagem C.

2.3 Vantagens no uso do Flutter

Em comparação com o desenvolvimento nativo, o Flutter tem como vantagem um desenvolvimento mais rápido, pois os programadores apenas necessitam de criar um único código e isso reduz significativamente o tempo gasto e o tamanho da equipa necessária para o desenvolvimento da aplicação. O Flutter contém uma enorme variedade de widgets personalizáveis que serão consistentes nos vários sistemas operativos, mantendo a mesma aparência nos vários. O Flutter conta com um recurso denominado Hot Reload, que permite aos programadores visualizar em tempo real as alterações feitas ao código na aplicação em execução sem que seja necessário reiniciar a aplicação, tornando o processo de desenvolvimento mais eficiente.

2.4 Desvantagens no uso do Flutter

Como desvantagens, podemos destacar os poucos anos de vida do Flutter, o que pode levar grandes empresas a hesitarem usar a plataforma. Outro ponto a considerar é o tamanho das aplicações Flutter, que pode ser maior do que o de aplicações nativas. Isso pode ser um problema para dispositivos com recursos limitados.

2.5 Utilização do Flutter

Embora seja uma plataforma relativamente nova, existem muitas grandes empresas a realizar a sua migração para o framework Flutter. Sendo o caso da BMW, Google, eBay, Toyota, entre outras.

3 Caso de estudo

Para compreender melhor as diferenças, tentei criar a mesma aplicação em ambos os ambientes (Kotlin e Dart).
Relativamente ao framework Flutter, a instalação foi simples e rápida. A construção do código foi feita no Visual Studio Code com a extensão do Flutter.
O que mais me saltou à vista foi a grande diferença de código necessária e a sensação da aplicação ser mais "leve", pelo menos no meu ambiente. É possível verificar o uso de widgets na construção da interface, como por exemplo o MaterialApp. Este widget fornece a estrutura geral da aplicação.

```kotlin
package com.example.greetingcard

import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.material3.MaterialTheme
import androidx.compose.material3.Surface
import androidx.compose.material3.Text
import androidx.compose.runtime.Composable
import androidx.compose.ui.Modifier
import androidx.compose.ui.tooling.preview.Preview
import com.example.greetingcard.ui.theme.GreetingCardTheme
import androidx.compose.ui.unit.dp
import androidx.compose.foundation.layout.padding

class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            GreetingCardTheme {
                // A surface container using the 'background' color
from the theme
                Surface(modifier = Modifier.fillMaxSize(), color =
MaterialTheme.colorScheme.background) {
                    Greeting("Android")
                }
            }
        }
    }
}

@Composable
fun Greeting(name: String, modifier: Modifier = Modifier) {
    Text(
            text = "Hello $name! Teste para SMA",
            modifier = modifier.padding(24.dp)
    )
}

@Preview(showBackground = true)
@Composable
fun GreetingPreview() {
    GreetingCardTheme {
        Greeting("Android")
    }
}
```

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

void main() {
  runApp(const MainApp());
}

class MainApp extends StatelessWidget {
  const MainApp({super.key});

  @override
  Widget build(BuildContext context) {
    return const MaterialApp(
      home: Scaffold(
        body: Center(
          child: Text('Hello World! Teste para SMA'),
        ),
      ),
    );
  }
}
```

Fig. 2. À esquerda o código Kotlin e à direita o Dart. Fonte: Alexandre Costa

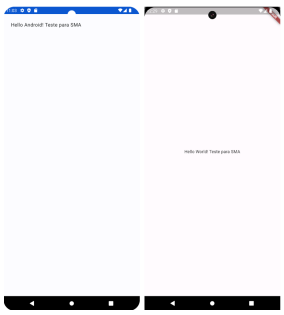

Fig. 3. À esquerda a aplicação em Kotlin e à direita em Dart. Fonte: Alexandre Costa

4 Conclusão

Verifiquei que o Flutter apresenta vantagens significativas para o desenvolvimento de aplicações mobile multiplataforma. Graças ao seu desenvolvimento rápido, permite que a aplicação seja criada com apenas um código para Android e iOS. Conta também com widgets personalizáveis muito próximos dos nativos. Sendo uma boa opção para quem tem disponíveis poucos recursos e pretende uma aplicação multiplataforma rápida e simples. Ainda assim, poderá não ser a melhor alternativa para quem pretende algo muito específico no Android ou iOS, devido a ser um framework relativamente recente e que, por esse motivo, pode não possuir algum tipo de suporte a certas funcionalidades. Por fim, foram usadas ferramentas de IA para tentar melhorar algumas frases e construir uma boa estrutura/índice do trabalho, mesmo não havendo melhorias significativas com o seu uso.

References

1. Coodesh.com, https://coodesh.com/blog/dicionario/o-que-e-flutter/, last accessed 2024/03/27
2. bankmycell.com, https://www.bankmycell.com/blog/how-many-android-users-are-there, last accessed 2024/03/27
3. aws.amazon.com, https://aws.amazon.com/pt/what-is/flutter/, last accessed 2024/03/27
4. Desenvolvimento de um aplicativo utilizando o framework flutter e arquitetura limpa, https://repositorio.pucgoias.edu.br/jspui/handle/123456789/1861, last accessed 2024/03/28
5. flutterparainiciante, https://flutterparainiciantes.com.br/widgets/, last accessed 2024/03/29
6. DART Homepage, https://dart.dev/overview, last accessed 2024/03/29
7. DART Homepage - Hot reload, https://docs.flutter.dev/tools/hot-reload, last accessed 2024/03/29
8. Start building Flutter Android apps on Windows, https://docs.flutter.dev/get-started/install/windows/mobile?tab=later-start, last accessed 2024/03/30
9. Write your first Flutter app, https://docs.flutter.dev/get-started/codelab, last accessed 2024/03/30