VOLTAR

Introdução ao Flutter

Nessa aula vamos entender um pouco melhor a arquitetura do Flutter, como uma aplicação é construída e como os projetos são capazes de serem executados em Android e iOS.

NESTE TÓPICO

O que é o Flutter?

Por mais que já tenhamos falando muito sobre este assunto, não podemos deixar de reforçar que o Flutter é uma ferramenta que permite o desenvolvimento de aplicativos para iOS e Android com um único código-fonte.

Isso quer dizer que o resultado de um projeto de desenvolvimento é atingido com um código-fonte unificado e pode ser distribuído para smartphones com iOS (Apple) e Android, através do App Store (Apple) e Google Play Store (Google).

PRECISAMOS DAS LINGUAGENS HÍBRIDAS! :D

Se não existissem linguagens híbridas, teríamos que aprender uma linguagem para cada sistema operacional, pois hoje em dia praticamente todos os aplicativos possuem versões para iOS e Android. Uma aplicação hoje em dia, que se limita à um tipo de sistema operacional apenas, está fadada ao fracasso.

É muito importante lembrarmos que o sistema operacional iOS, da Apple, é proprietário, ou seja, possui o código-fonte licenciado única e exclusivamente para a empresa que o detém, enquanto o Android é um sistema operacional de código aberto desenvolvido pelo Google que pode ser personalizado e distribuído, desde que a empresa que o altere pague ao Google para isso. Sim, é isso mesmo que você leu: O Android Studio é um sistema de código aberto, mas licenciado.

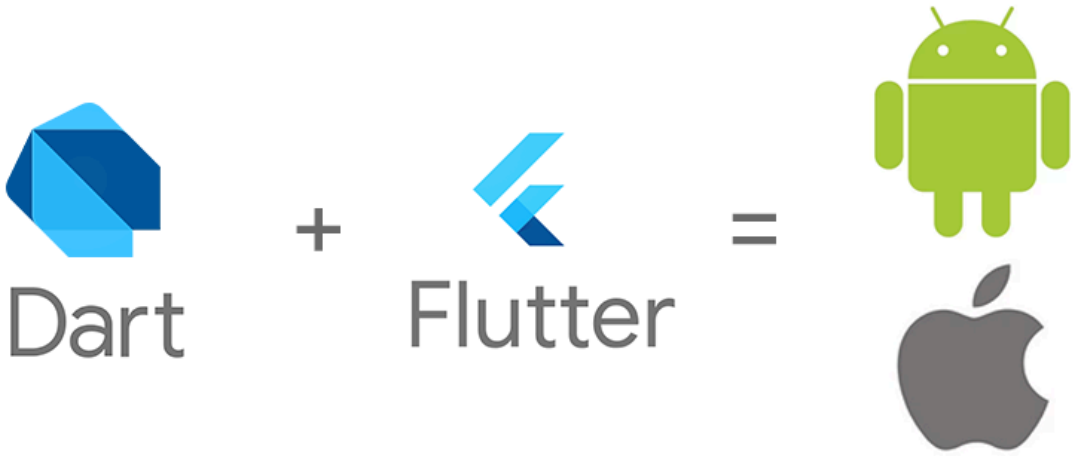
Para desenvolvermos aplicações para estes sistemas operacionais temos que aprender a linguagem que cada fabricante permite. Por exemplo, para o desenvolvimento para iOS, temos que aprender uma linguagem chamada Swift e/ou Objective C e, para desenvolvermos aplicações para Android, temos que aprender Kotlin ou Java. Entretanto, as linguagens e ferramentas de desenvolvimento híbrido vem quebrando este paradigma, pois permitem a geração de um código nativo para cada tipo de sistema operacional.

É claro que não faltam artigos dizendo que a performance de aplicações com desenvolvimento nativo é maior em comparação ao desenvolvimento híbrido, entretanto o Flutter é uma das linguagens que está conseguindo quebrar isso, devido à sua característica e arquitetura.

E é exatamente isso que deixa o Flutter ainda mais legal: Além de ser fácil e de rápida aprendizagem, ele gera aplicações rápidas e realmente bonitas.

Tem-se observado no mercado que muitos desenvolvedores de Flutter não aprendem Flutter como uma nova linguagem, mas sim como sua primeira linguagem, pois sabe-se que muitos destes programadores vêm de áreas não necessariamente correlatas ao desenvolvimento. Muitos designers, por exemplo, trabalham com Flutter e desenvolvem aplicativos maravilhosos, de forma rápida, eficiente e concisa.

Então o Flutter é nada mais do que um kit de ferramentas para desenvolvimento híbrido. Isso, no mundo da programação possui um nome: SKD, que significa *Software Deveoplment Kit*, ou ainda, Kit de Desenvolvimento de Software, que produz código nativo através de um código híbrido.



Código-fonte único que gera Apps incríveis para Android e iOS

MAS POR QUE O GOOGLE FEZ O FLUTTER SE JÁ POSSUI O KOTLIN?

De fato, o Google é detentor do Kotlin, uma linguagem muito simples para desenvolvimento mobile, entretanto, para desenvolvimento nativo, exclusivo para Android. O Dart com Flutter foi desenvolvido por equipes diferentes do Google e rapidamente ganharam notoriedade no mercado, pois é uma aposta (que deu certo) do Google para o mercado de aplicações híbridas.

Então o Flutter fornece uma série de mecanismos que, combinados com a linguagem Dart, faz com que não precisemos reinventar a roda, ou seja, o Flutter possui uma série de componentes (chamados de Widgets) que podem são altamente personalizáveis e permitem criar aplicações incríveis.

PRECISO SABER ALGUMA LINGUAGEM DE PROGRAMAÇÃO PARA DESENVOLVER COM FLUTTER?

Não, não é necessário ter nenhum conhecimento prévio em nenhuma linguagem de programação para começar a desenvolver aplicativos maravilhosos com Flutter. Conhecer outras linguagens sempre ajuda na aprendizagem de uma nova linguagem, mas tudo que você precisa é ter bons conhecimentos de lógica e garra para aprender.

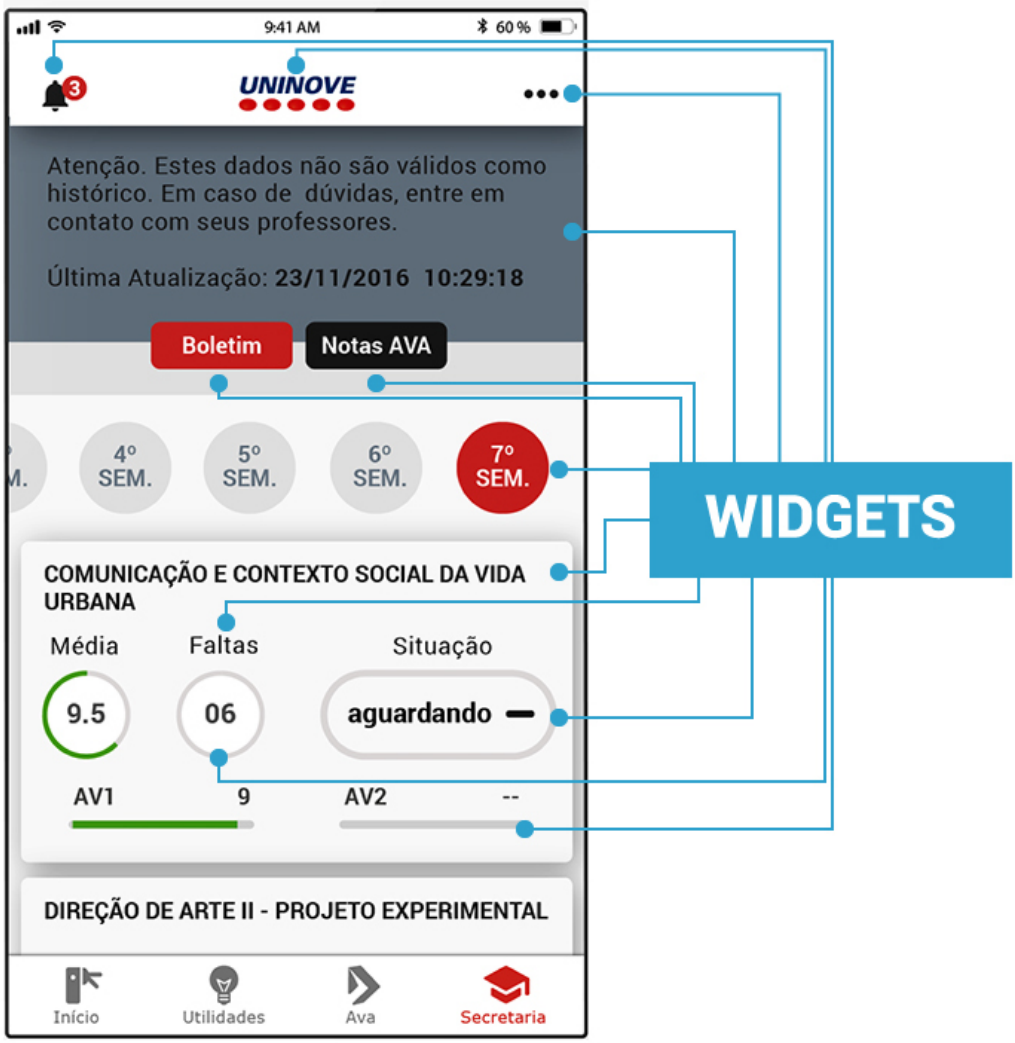
Arquitetura do Flutter

Agora que sabemos o que é o Flutter, precisamos nos aprofundar um pouco mais para saber como ele funciona, sua arquitetura e como os Apps são construídos. Este capítulo é, sem a menor sombra de dúvidas, a introdução mais importante da disciplina, pois é muito importante endentemos como nosso código deve ser construído.

A primeira coisa que temos que entender é que o Flutter cria a interfaces de usuário como código (*User Interface as Code*), em uma estrutura hierárquica de árvore. Isso quer dizer que não existe, até então, nenhuma ferramenta de "drag and drop" (arrastar e soltar) para a montagem das interfaces. Isso quer dizer que tudo é feito em código-fonte. Apesar de isso assustar um pouco no começar, acredite, isso tornará nosso trabalho ainda mais fácil.

Isso quer dizer que no desenvolvimento com o Flutter, todos os componentes visuais são chamados de "widgets" (ferramentas). Veja, por exemplo, a imagem abaixo, que mostra o print do aplicativo da Uninove (pode ser uma versão anterior a que você está usando, pois o aplicativo

evolui constantemente). Se pensarmos nesta tela como exemplo, temos, então, que absolutamente todos os componentes visuais podem ser considerados Widgwets e cada um possui uma característica, um posicionamento, uma imagem (ou não), um texto etc. Se você parar para analisar cuidadosamente, a própria página onde as informações são exibidas é um Widget que possui vários widgets criados dentro dela.



Os widgets são os componentes apontados na tela do Aplicativo Oficial da UNINOVE.

Agora que você sabe que em desenvolvimento com Flutter todos os componentes são widgets hierárquicos e já sabe também que o Flutter utiliza uma arquitetura do tipo *UI as code* (interface de usuário como código), você já pode saber, então, que toda a codificação também é totalmente hierárquica, ou seja, o código-fonte é feito, também, em formato de árvore. Isso quer dizer que logo de cara você já deve saber que seu código precisará ser muito organizado, pois neste tipo de implementação é muito fácil se perder em relação aos parênteses e chaves. E é por isso que devemos utilizar uma IDE (Integrated Development Environment - Ambiente de Desenvolvimento Integrado) que seja poderosa e nos ajude a encontrar e corrigir erros.

Esse tipo de arquitetura é que torna o Flutter tão rápido de ser aprendido e de ser usado para codificar uma interface complexa em tão pouco tempo.

Como um projeto em Flutter é compilado

Compilar significa, resumidamente, pegar um código fonte, interpretá-lo e gerar um executável funcional para o usuário final. Isso deve ser feito de forma autônoma em nossos projetos e é papel do compilador fazer isso.

Em linguagens convencionais e até mesmo clássicas, o seu código fonte passa pelo compilador e é gerado um executável para a plataforma o qual você está executando seu projeto. Por exemplo, na linguagem "C", seus arquivos de código (normalmente com a extensão ".c" ou ".h" geram um ".exe" (no Windows); a linguagem java utiliza extensões ".java" para os arquivos de código fonte, (obs.: as extensões ".jar" e ".war" são nada mais do que um aglomerado de ".class").

Mas e no Flutter? O que acontece quando nosso código é enviado para o compilador? Será que é gerado um "executável" para cada plataforma? Bom, no desenvolvimento em Flutter o que é gerado, na verdade, é um projeto nativo para Android e um projeto nativo para iOS e são estes que são compilados para pela plataforma nativa para suas respectivas plataformas (Android e iOS).

E é exatamente por isso que precisamos da SKD do Android (que vem com o Android Studio) e o xCode (Apple) para gerar nossas aplicações, pois nosso código é convertido para um projeto nativo Android e iOS e estes são recompilados. E também é por isso que as aplicações desenvolvidas com Flutter são tão rápidas, pois são, oriundas um código nativo.

Mas muita calma! Isso não quer dizer que você pode abrir o projeto Android, por exemplo, resultante da compilação em Flutter e sair editando livremente os componentes, pois quem os controla é o Flutter. O Flutter controla cada pixel da aplicação, através de seu Framework. Não é preciso preocupar-se muito com isso no momento, pois o Flutter faz tudo isso para nós, mas é importante saber que você não poderá editar livremente cada um dos projetos gerados, pois você não encontrará os componentes (widgets) convertidos para as plataformas nativas exatamente como seriam desenvolvidos e, por isso, somente conseguimos editar nosso projeto original, em Flutter.

As versões do Flutter

O Flutter com Dart compõem uma dupla relativamente nova e que tem ganhado notoriedade importante no mercado de aplicações móveis nos últimos meses. Isso quer dizer que, quando mais desenvolvedores usam a tecnologia (o que é bom), mas ela evolui e é exatamente isso o que está acontecendo com essa linguagem: a disponibilização de novas versões tem ocorrido com muita frequência.

Portanto é muito importante sabermos que o Flutter ganha novas versões rapidamente, mas isso não quer dizer que a linguagem esteja mudando ou que a linguagem seja instável, muito pelo contrário. A linguagem não apenas já está consolidada como muito estável.

A evolução do Flutter inclui sempre melhorias e correções, assim como a inclusão de novas funcionalidades nativas e melhorias de desempenho (ainda mais). Boa parte das mudanças são internas da própria ferramenta e não nos devemos nos preocupar muito com isso.

Por isso, independentemente de quando você está vendo este material, certamente o Flutter já terá evoluído bastante em relação a versão usada aqui, mas não se preocupe com isso, pois as funcionalidades são mantidas. Isso quer dizer que este curso permanecerá atualizado mesmo que a versão do Flutter mude. Isso apenas quer dizer que algumas funcionalidades que implementaremos aqui neste curso podem ser, com o tempo, implementadas de outras formas com novas atualizações, mas elas permanecerão funcionando.

Sabendo disso, podemos finalmente começar a aprender a como utilizar o Flutter com o Dart.

Primeiros passos: Criação de um projeto

Toda aplicação que formos desenvolver será tratada como um projeto e, por isso, precisamos saber como criar um projeto corretamente.

Para isso, existem, essencialmente, duas formas: Usando o terminal (por linha de comando) ou usando os próprios recursos da IDE que vamos utilizar. Entretanto é importante salientar que:

- Se você for utilizar única e exclusivamente o Visual Studio Code para desenvolver as aplicações, você deverá utilizar a criação do projeto via linha de comando.
- Se você for utilizar o Android Studio ou a IntelliJ, você poderá utilizar a própria IDE para criação do projeto, embora possa, também, utilizar o método por linha de comando no terminal.

Quando criamos um projeto pela primeira vez um aplicativo bastante simples e de exemplo é criado e, nesta aula, vamos explorá-lo para aprender a estrutura de um projeto em Flutter.

O vídeo abaixo mostra como criar um projeto via linha de comando e, embora esteja sendo executado um macOS, não se preocupe pois o resultado é exatamente o mesmo em qualquer plataforma.

Lembre-se, ainda, que o próprio Visual Studio Code, se você for utilizá-lo, possui um mecanismo que mostra o terminal dentro dele mesmo, bastando clicar em "View" (visualizar) e "terminal".

Muitos outros comandos serão aprendidos ao longo deste curso e serão apresentados conforme sua necessidade.

Quiz

Exercício Final

Introdução ao Flutter

INICIAR

Referências

BRACH, Gilard, LARS, Bak. Dart: a new programming language for structured web programming. **GOTO Conference**, 10 out. 2011. Disponível em: https://gotocon.com/dl/goto-aarhus-2011/slides/GiladBracha_and_LarsBak_OpeningKeynoteDartANewProgrammingLanguageForStructuredWebProgramming.pdf. Acesso em: 12 nov. 2020.

DART. **Dart documentation**. *Site*. Disponível em: https://dart.dev/. Acesso em: 12 nov. 2020.

FELIX, Rafael. **Programação orientada a objetos.** São Paulo: Pearson, Ed. Pearson, 2017. *E-book*. Disponível em: https://plataforma.bvirtual.com.br/Leitor/Loader/128217/pdf. Acesso em 19 nov. 2020.

OKEDIRAN, O. O. *et al*. Mobile operating systems and application development platforms: a survey. **Int. J. Advanced Networking and Applications**. v.6, n.1, p. 2195-2201, july-aug. 2014. Disponível em: https://www.ijana.in/download%206-1-9.php?file=V6I1-9.pdf. Acesso em: 12 nov. 2020.

SCHWARZMÜLLER, Maximilian. **Learn Flutter and Dart to Build iOS and Android Apps 2020.** Oreilly, Packt Publishing, 2020. *Vídeo*. Disponível em: https://learning.oreilly.com/videos/learn-flutter-and/9781789951998/. Acesso em 19 nov. 2020.

SINHA, Sanjib . **Quick start guide to Dart programming:** create high performance applications for the web and mobile. Lompoc, CA, EUA: Apress, 2019. *E-book*. Disponível em: https://learning.oreilly.com/library/view/quick-start-guide/9781484255629/. Acesso em: 12 nov. 2020.

WINDMILL, Eric. **Flutter in action.** Nova Iorque: Manning publications, 2020. *E-book*. Disponível em: https://learning.oreilly.com/library/view/flutter-in-action/9781617296147/, Acesso em: 12 nov. 2020.

Avalie este tópico