



FUTURO DO TRABALHO, TRABALHO DO FUTURO

DESENVOLVIMENTO MOBILE COM FLUTTER BÁSICO

Apostila do aluno



MINISTÉRIO DA
CIÊNCIA, TECNOLOGIA
E INOVAÇÕES



Rita Barbosa – Desenvolvedor Mobile
Bruno Rodrigues – Desenvolvedor Mobile
Autor da apostila

Larissa Jessica Alves – Analista de Suporte Pedagógico
Revisão da apostila

Fit Instituto de Tecnologia
Sorocaba, novembro de 2021

Instrutor



Rita de Cassia A. Barbosa, tecnóloga em Processamento de Dados pela Faculdade de Informática Tibiriçá de São Paulo, trabalha na área TI a mais de 25 anos, se especializou em Gestão de Negócios em TI. Atua no FIT como Analista de Capacitação e docente de Ensino Médio Técnico na Etec Martinho Di Ciero – Centro Paula Souza.

Mais informações em: <https://www.linkedin.com/in/rita-barbosa-074641/>

APRESENTAÇÃO

O curso de Desenvolvimento Mobile com Flutter Básico com 18 horas, visa atender a crescente demanda em desenvolvimento de aplicativos para celular no Brasil. Segundo pesquisa realizada pela empresa App Annie no 2º. Semestre de 2021, o Brasil é o país com a maior média de tempo gasto em aplicativos para celular, seguido pela Indonésia com 5.3 horas e Índia com 4.9 horas. Segundo a empresa os resultados sugerem que a pandemia de Covid-19 impulsionou muito o aumento do tempo dos usuários nos aplicativos, ocasionando um crescimento de 45%.

Assim a presente apostila é um instrumento teórico que complementa o curso de Desenvolvimento Mobile com Flutter Básico, executado pelo FIT-Instituto de Tecnologia. Nela apresentaremos a introdução sobre a história do Flutter e da Linguagem DART, o conceito de cross-plataform e bem como a instalação das ferramentas necessárias para desenvolvimento do código, como também, todas as funções, laços e repetições. Finalizaremos o estudo com exercícios práticos, desde a escrita do código, com também emulação do ambiente. Para o Mundo do IOS, apresentaremos apenas os conceitos teóricos básicos.

Desejamos a você prezado aluno que tenha um excelente curso!!

Boa Leitura!!

Indicação de ícones



Saiba mais: *oferece novas informações que enriquecem o assunto ou “curiosidades” e notícias recentes relacionadas ao tema estudado.*



Exemplos: *descreve exemplos sobre o assunto que está sendo abordado.*



Atenção: *indica pontos de maior relevância no texto.*



Avisos: *oferece avisos referente ao assunto..*

Sumário

Aprendendo sobre o Flutter	6
Instalação do Flutter SDK	7
1.1 Instalação do Android Studio.....	10
1.2 Instalação das Extensões do Flutter e Dart.....	14
1.3 SDK Manager.....	16
1.4 Emulador	17
1.5 Instalação Licenças Android.....	25
1.6 Instalação e Configuração do VS Code.....	29
1.7 Configuração do Flutter	30
1.8 Criação do 1º. Projeto – Hello World – Exercício Prático	32
Introdução ao DART	41
1.9 Histórico Dart.....	41
1.10 Variáveis	42
1.11 Operadores	43
1.12 Operadores Relacionais.....	43
1.13 Condicionais.....	44
1.14 Repetições	45
1.15 Funções	46
Widgets Básicos	49
1.16 O que são Widgets?.....	49
1.17 Layout	51
1.18 Linha (Row).....	53
1.19 Coluna (Column)	53
1.20 Texto (Text).....	54
1.21 Botão (Button)	54
1.22 Imagem (Image).....	55

1.23	Exercício Prático	55
	Publicação de Aplicativo nas Lojas: Google Play e Apple Store.....	60
1.24	Criando uma conta de desenvolvedor no Google Play	60
1.25	Criando uma conta de desenvolvedor na App Store.....	61
	Desafio.....	63
	Conclusão.....	64
	Referências.....	65
	CONTROLE DE REVISÃO DO DOCUMENTO / <i>DOCUMENT REVISION</i>	
	<i>CONTROL</i>	66

Aprendendo sobre o Flutter

O **Flutter** é uma estrutura (*framework*) feita para resolver problemas específico. Sua construção foi realizada pela Google para facilitar o desenvolvimento mobile multiplataforma (Android/iOS). De maneira geral, na programação, um *framework* é um conjunto de códigos genéricos capaz de unir trechos de um projeto de desenvolvimento. Para essa estrutura utilizamos o **Dart** como principal linguagem de desenvolvimento que abordaremos mais adiante.

A primeira versão do Flutter teve o codinome "Sky", era executada no sistema operacional Android. Em 2015, foi apresentado na cúpula de desenvolvedores Dart com a intenção declarada de ser capaz de renderizar consistentemente 120 quadros por segundo. Em 4 de dezembro de 2018, o Flutter 1.0 foi lançado no evento Flutter Live, substituindo a primeira versão "estável" do *Framework*. O Flutter Release Preview 2, foi anunciado durante na keynote do Google Developer Days em Xangai.

Em 06 de maio de 2020 foi lançado o Dart SDK na versão 2.8 e o Flutter na versão 1.17.0, onde foi adicionado suporte a API Metal (proporciona melhor desempenho no IOS), melhorando muito o desempenho em dispositivos IOS em 50%, novos widgets do Material e novas ferramentas de rastreamento de rede.

Recentemente, em 16 de dezembro, após a elaboração desta apostila, o Flutter sofreu atualização da sua versão (atual 2.8.1), porém, vamos utilizar a versão 2.5.3 (setembro/2021) na qual foi validada para este curso.

O Flutter permite o desenvolvimento do código em "Cross-Platform". Baseia-se no uso de um *framework* que possibilita ao programador desenvolver um código que seja executado em várias plataformas, em nosso caso específico, ANDROID e IOS.

Instalação do Flutter SDK

O Flutter SDK pode ser utilizado no Windows, MacOS e Linux. Porém, nesse curso abordaremos apenas a instalação em ambiente Windows.

- I. Para iniciarmos a instalação, acesse o link : <https://flutter.dev/> e clique no canto superior direito do site em “Get Started”:

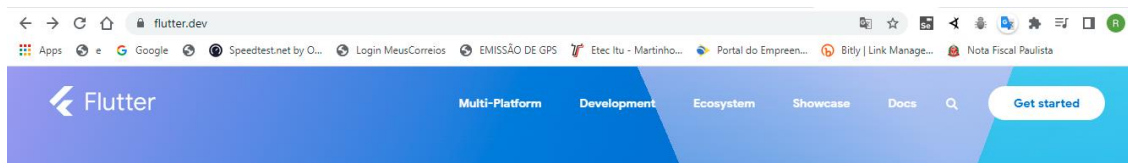


Figura 1: acesso Flutter SDK para download. Fonte: Flutter.

- II. Escolha o Sistema Operacional Windows:

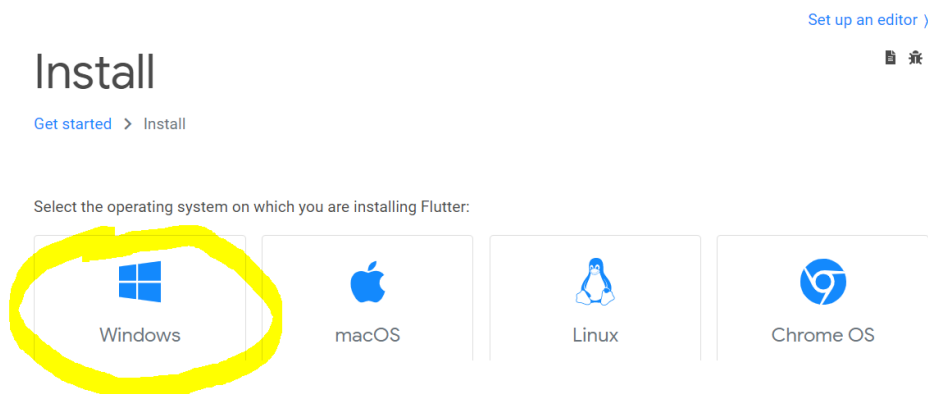


Figura 2: sistema operacional. Fonte: Flutter.

III. Importante atentar-se para os requisitos de sistema:

System requirements

To install and run Flutter, your development environment must meet these minimum requirements:

- **Operating Systems:** Windows 7 SP1 or later (64-bit), x86-64 based.
- **Disk Space:** 1.64 GB (does not include disk space for IDE/tools).
- **Tools:** Flutter depends on these tools being available in your environment.
 - [Windows PowerShell 5.0](#) or newer (this is pre-installed with Windows 10)
 - [Git for Windows 2.x](#), with the **Use Git from the Windows Command Prompt** option.

If Git for Windows is already installed, make sure you can run `git` commands from the command prompt or PowerShell.

Figura 3: requisitos de sistema. Fonte: Flutter.



O site apresentará a versão 2.8.1-stable.zip para instalação, logo abaixo, há a mensagem: “For other release channels, and older builds, see the [SDK releases](#) page”, click em SDK releases, você acessará a página Flutter SDK Releases.

Selecione a Plataforma referente ao seu Sistema Operacional, no nosso caso, será o Windows e clique na versão 2.5.3 para download:

Flutter SDK releases

Development > Tools > SDK > Releases

The Stable channel contains the most stable Flutter builds. See [Flutter's channels](#) for details.

Please note - dev channel has been deprecated. Refer to this [blog](#) for more information.

Windows macOS Linux

Stable channel (Windows)

Select from the following scrollable list:

Version	Ref	Release Date
2.8.1	77d935a	12/16/2021
2.8.0	cf44000	12/8/2021
2.5.3	1811693	10/15/2021

Figura 4: Download Flutter SDK vs 2.5.3. Fonte: site oficial Flutter.

Verifique na sua pasta de Download se o arquivo foi salvo e inicie a instalação do Flutter.



Conforme recomendação do site, procure não instalar o Flutter no diretório C:\Program Files\, pois, este exige um nível maior de privilégios, portanto, sugerimos instalar em c:\Users\<your-user-name>\Documents.

- IV. Para o funcionamento do Flutter, será necessário a instalação do: PowerShell (no caso de versões abaixo do Windows 10) e o Git. Para as versões abaixo do Windows 10, deixaremos o link para instalação do Power Shell:

<https://docs.microsoft.com/pt-br/powershell/scripting/install/installing-powershell-on-windows?view=powershell-7.2#msi>, neste link haverá a explicação passo a passo para instalação e devidas configurações.

1.1 Instalação do Android Studio

Segundo Developers o Android Studio é um IDE (ambiente de desenvolvimento integrado) para desenvolver na plataforma Android. Foi anunciado em 16 de Maio de 2013 na conferência Google I/O. Android Studio é disponibilizado gratuitamente sob a Licença Apache 2.0”.

Para a instalação do Android Studio entre no site: <https://developer.android.com/studio?hl=pt-br>, o site irá identificar a sua versão do Windows, basta apenas clicar no botão verde intitulado”Download Android Studio” e siga os passos de instalação, a versão que iremos instalar é a 2021.1.1.21 Patch 1 no Windows, conforme abaixo:

- I. Esta é a tela de apresentação do Android Studio, clique em “Next”:



Figura 5: instalação do Android Studio. Fonte: autoria própria.

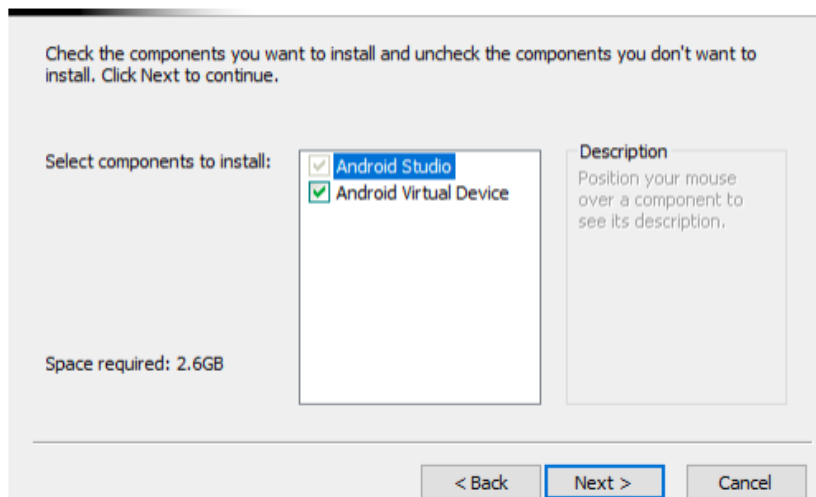


Figura 6: instalação do Android Studio. Fonte: autoria própria.

- II. Na sequência, o Android Studio apresentará o caminho de instalação do produto, que deverá em C:\Program Files\Android\Android Studio. Atenção - Não modifique o caminho de instalação, pois, isso poderá comprometer o funcionamento do IDE.

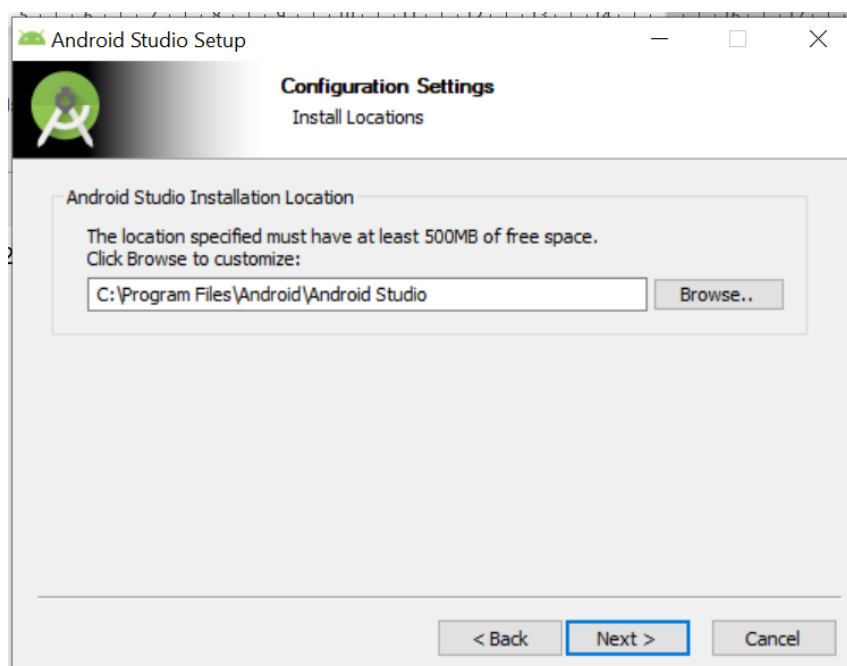


Figura 7: caminho de instalação do Android Studio. Fonte: autoria própria.

III. Selecione “Install”:

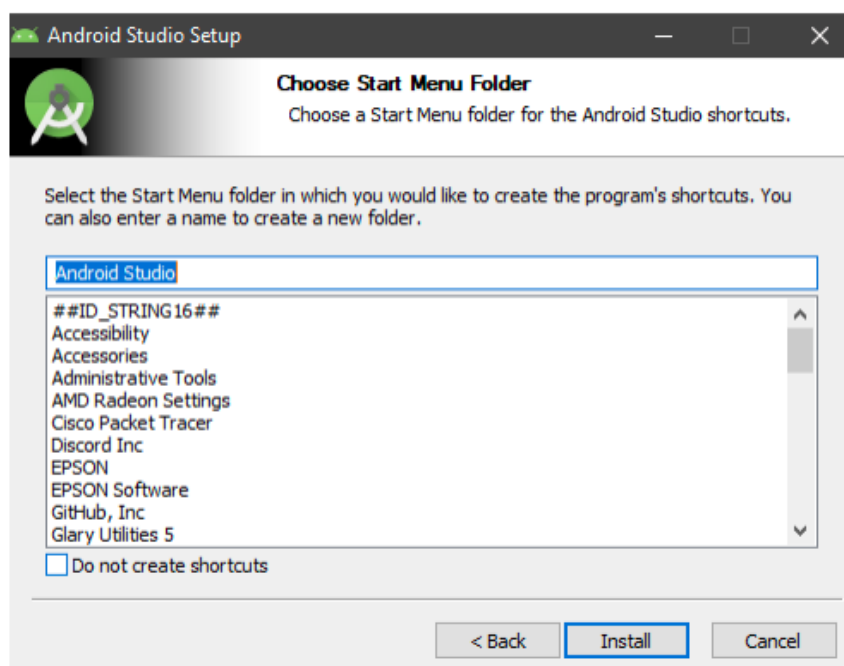


Figura 8: instalação do Android Studio. Fonte: autoria própria.

IV. Clique em “Next”:

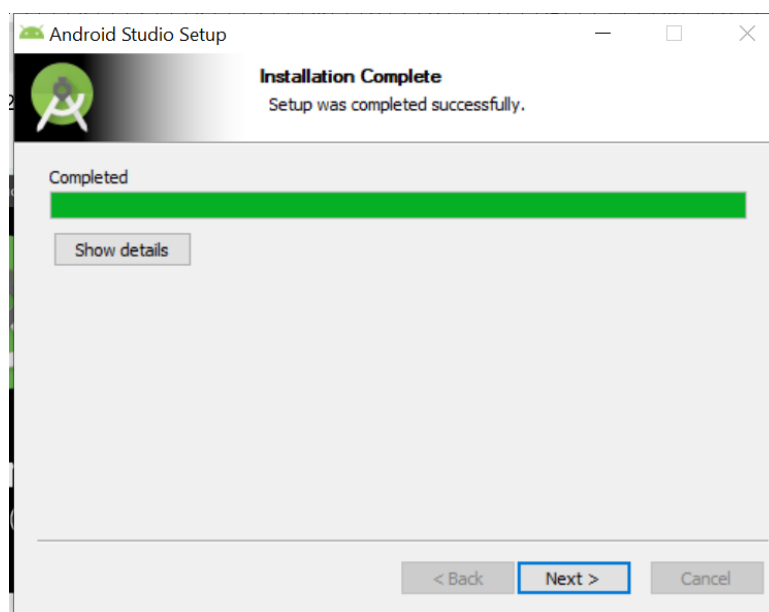


Figura 9: status da instalação do Android Studio. Fonte: autoria própria.

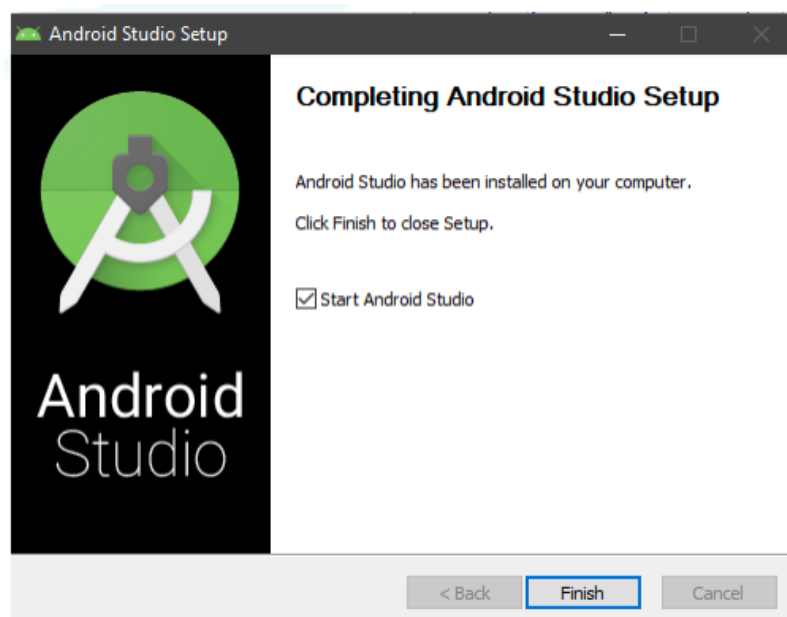


Figura 10: finalização da Instalação. Fonte: Developer.

- V. Ao finalizar a instalação, o Android Studio será inicializado e aparecerá uma tela de apresentação dos novos recursos, bem como alertas de atualização de Plugins, clique em “Upadate” para realizar o download e instalação.

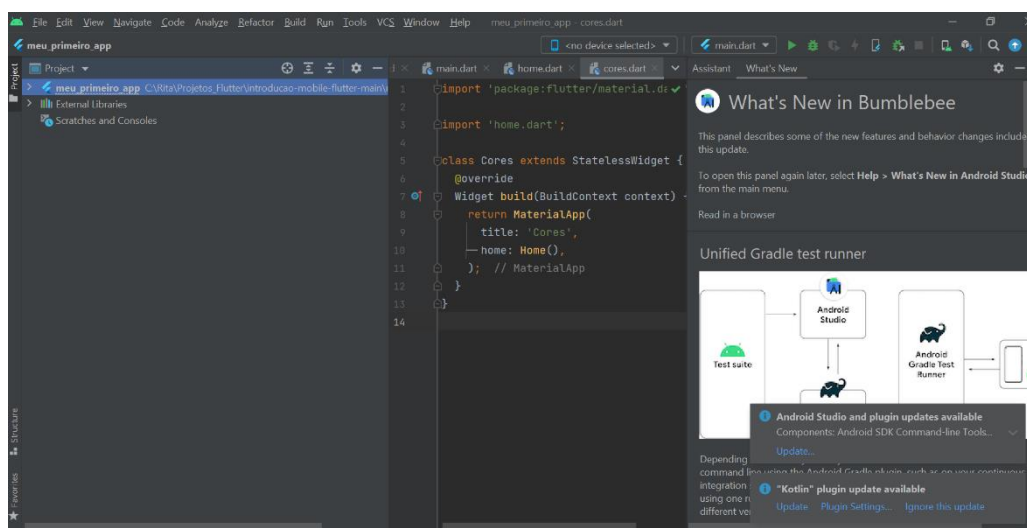


Figura 11: página inicial Android Studio. Fonte: Developer.

- VI. Ao finalizar cada update, clique em “Finish”. Na última atualização de plugins, reinicie se Android Studio para que as atualizações sejam concretizadas.

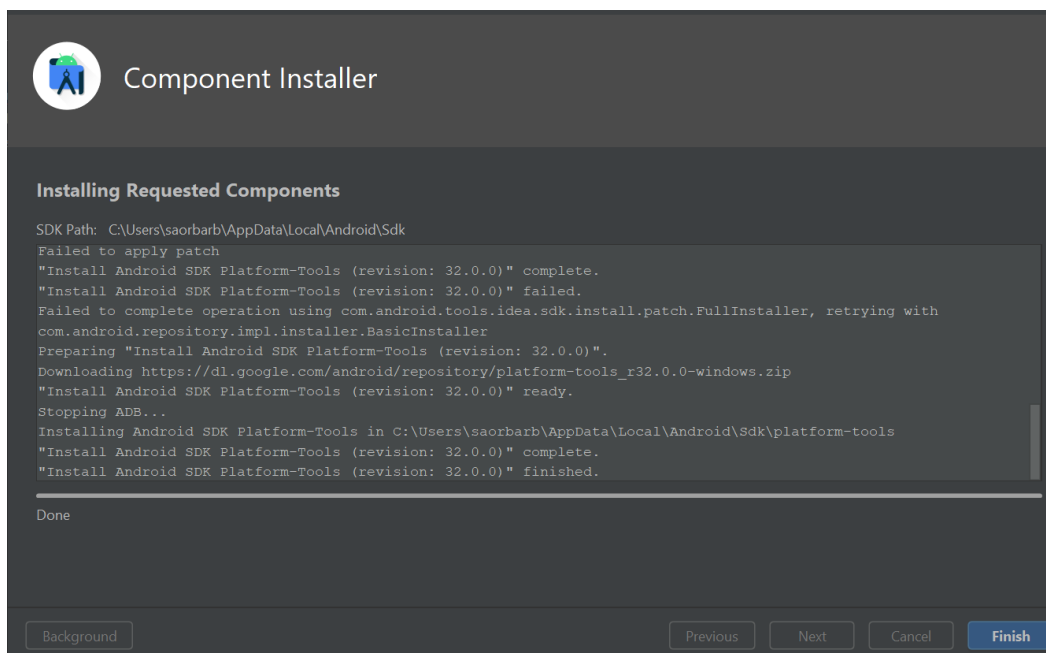


Figura 12: página inicial Android Studio. Fonte: Developer.

1.2 Instalação das Extensões do Flutter e Dart

- I. No menu do lado esquerdo, clique em File, Settings, Plugins, e verifique se os plugins do Flutter e Dart estão instalados, clique na aba “installed” para verificar. Vá até a caixa de pesquisa e digite Flutter e veja se está com o status “instaled”. Faço o mesmo com o Dart.

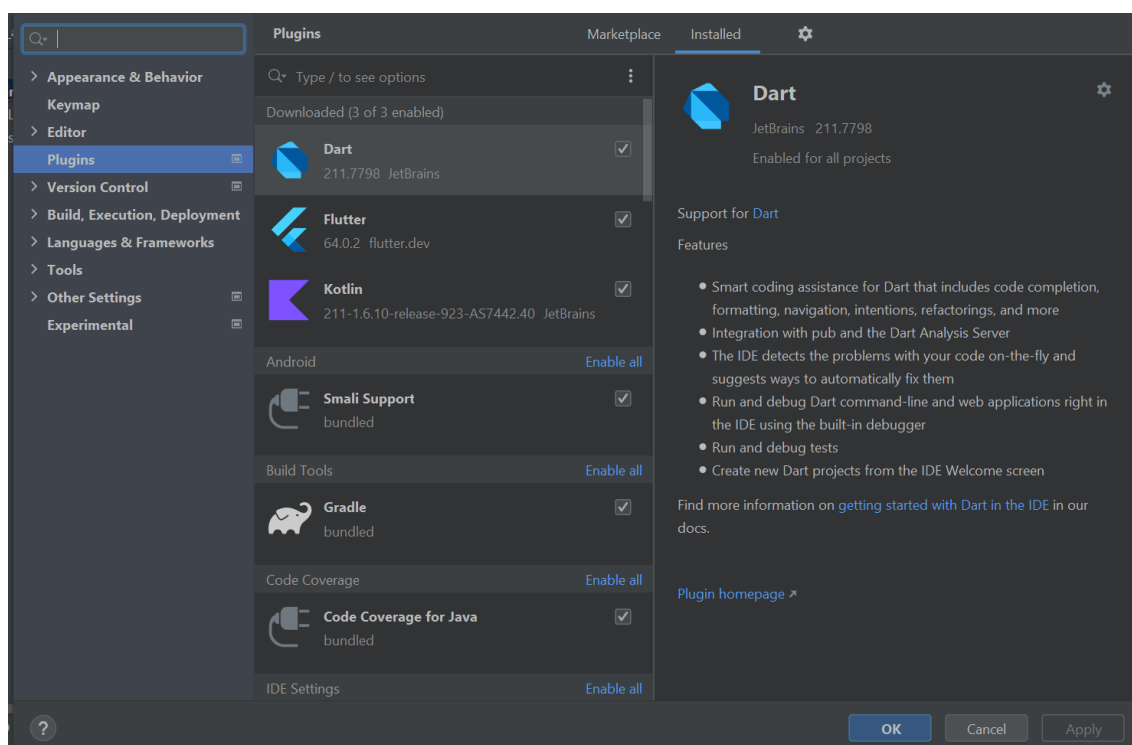


Figura 13: tela de instalação de Plugins. Fonte: autoria própria.

- II. Caso não estejam instalados, pesquise por “Dart”, clique em “Install” e aguarde a instalação até o final:

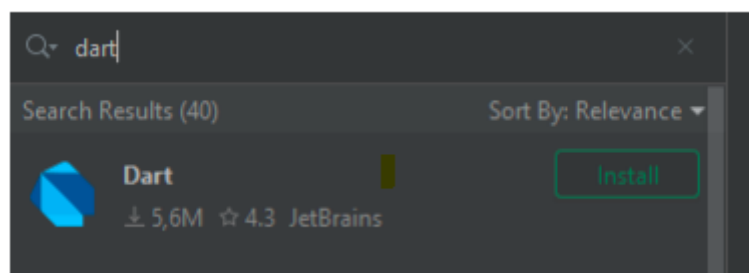


Figura 14: Plugin Dart. Fonte: Developer.

- III. Na sequência pesquise por “Flutter”, clique em “Install” e aguarde a instalação até o final:

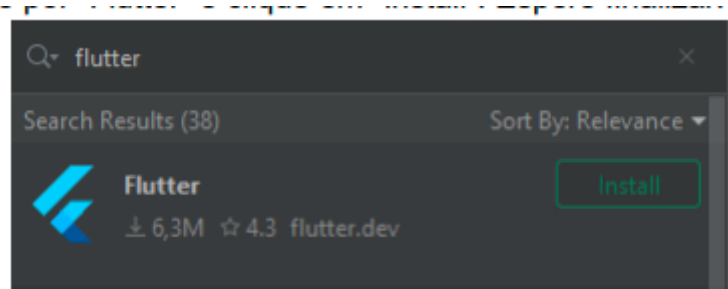


Figura 15: Plugin Flutter. Fonte: Developer.

- IV. Após a instalação, o Android Studio poderá aparecerá um botão ao lado do plugin instalado, 'restart IDE", clique para reinicializar seu Android Studio.

1.3 SDK Manager

No menu principal do Android Studio, clique em Tools, e selecione SDK Manager. Há também a opção via ícone no Menu Principal, conforme figura abaixo do lado direito:

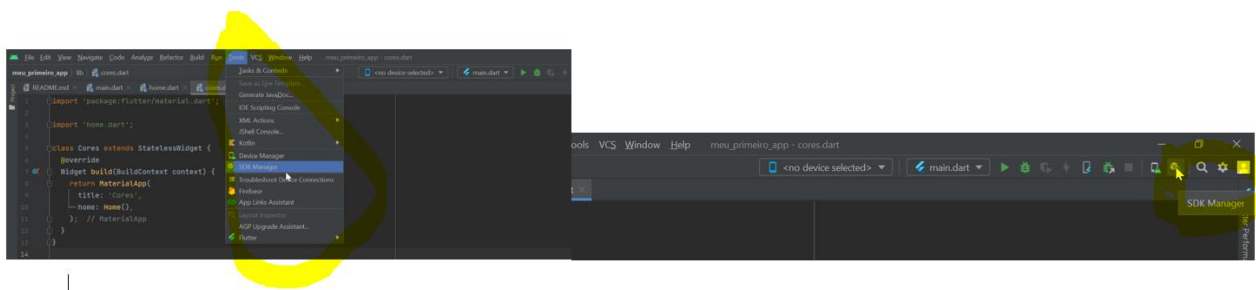


Figura 16: SDK Manager. Fonte: autoria própria.

V. Na opção “SDK Tools”

- Selecione a opção “Android SDK Command line Tools” e click em ok:

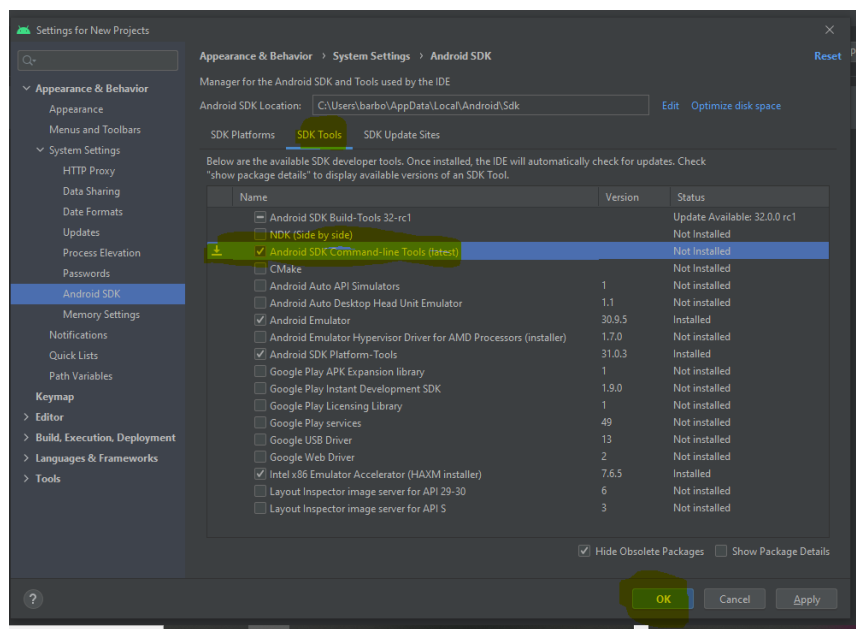


Figura 17: SDK Tools. Fonte: autoria própria.

1.4 Emulador

Para criar o emulador no Android Studio, crie primeiramente um novo projeto. No menu principal, clique em File, New, New Flutter Project, a seguinte tela aparecerá conforme abaixo, veja o caminho apontando para a criação do projeto, deverá ser o mesmo onde o Flutter SDK foi instalado.

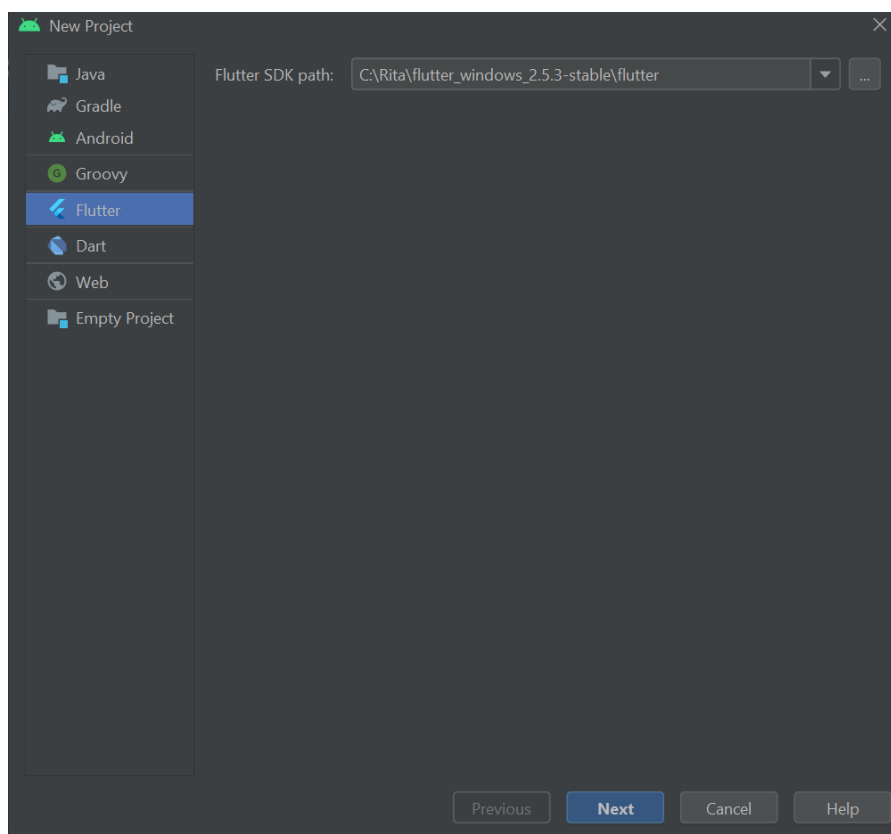


Figura 18: criação de Projeto no Flutter. Fonte: autoria própria.

Na sequência, dê um nome ao seu projeto e selecione o caminho onde está instalado o Flutter, conforme o exemplo abaixo:

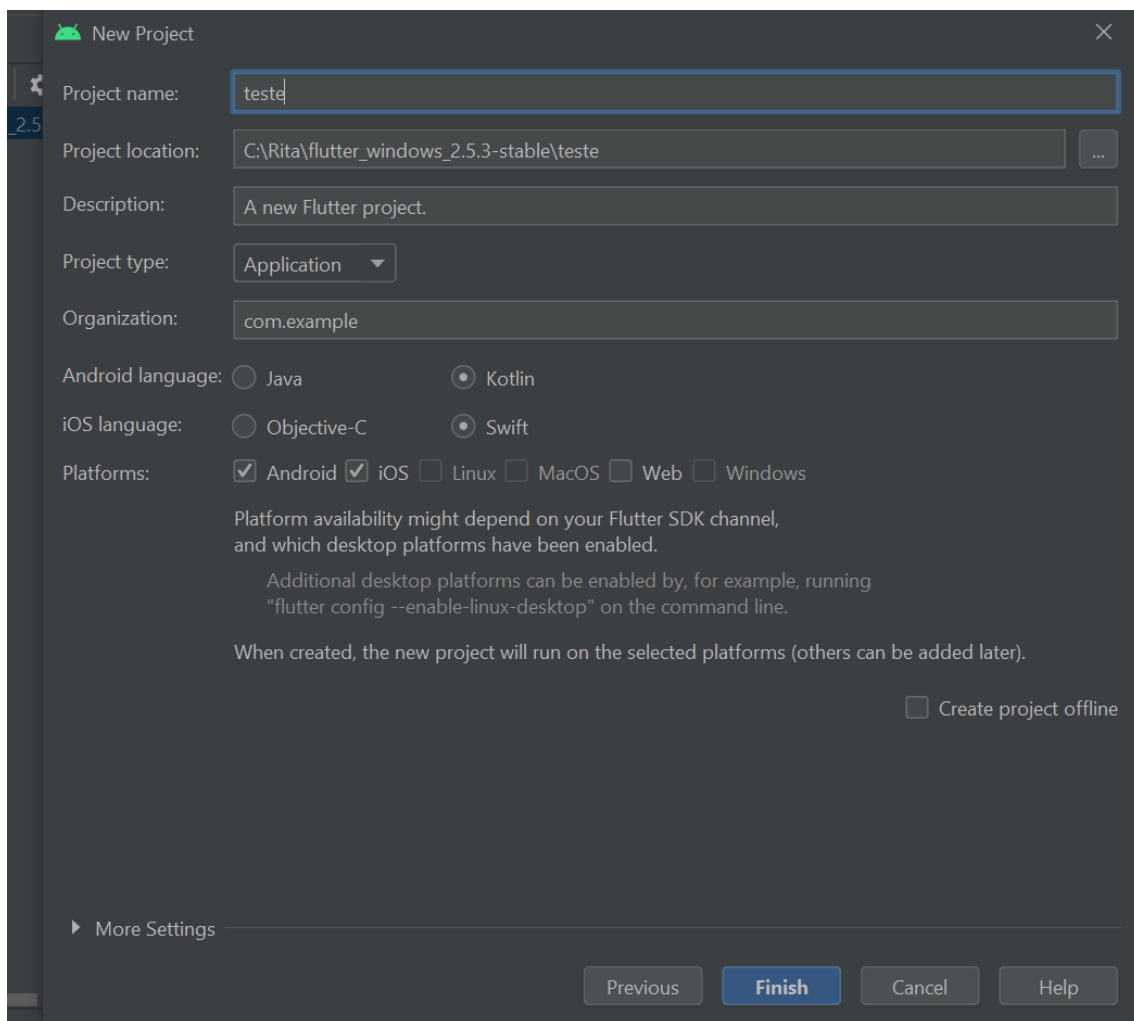


Figura 19: criação de Projeto no Flutter. Fonte: autoria própria.

Em seguida abra o seu projeto, clique em “The Window”, conforme abaixo:

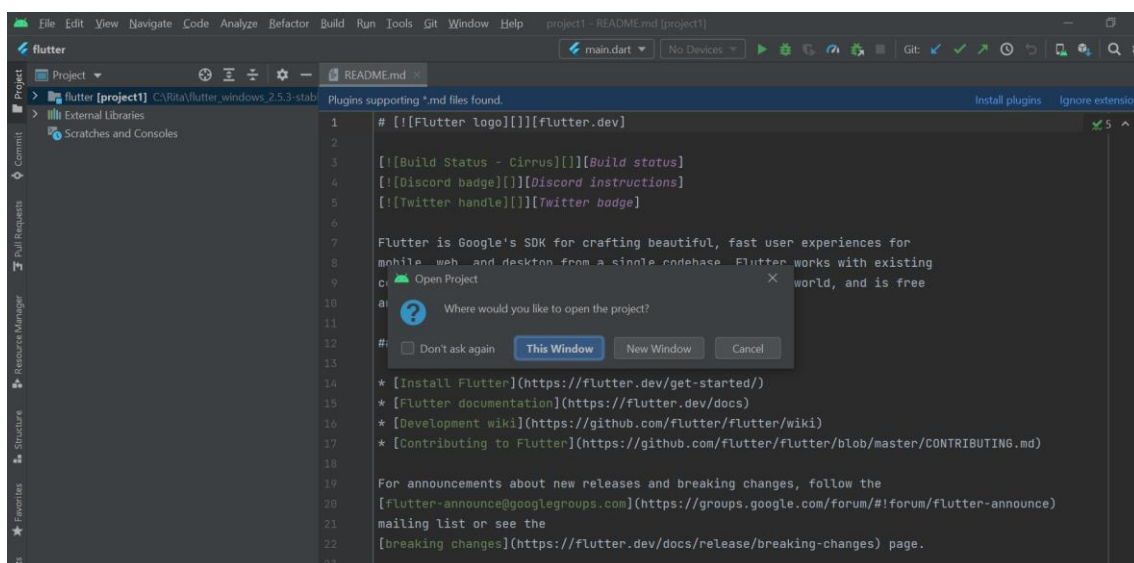


Figura 20: iniciando Projeto no Flutter. Fonte: autoria própria.

- I. Clique na aba, Tools, Device Manager do menu principal do Android Studio:

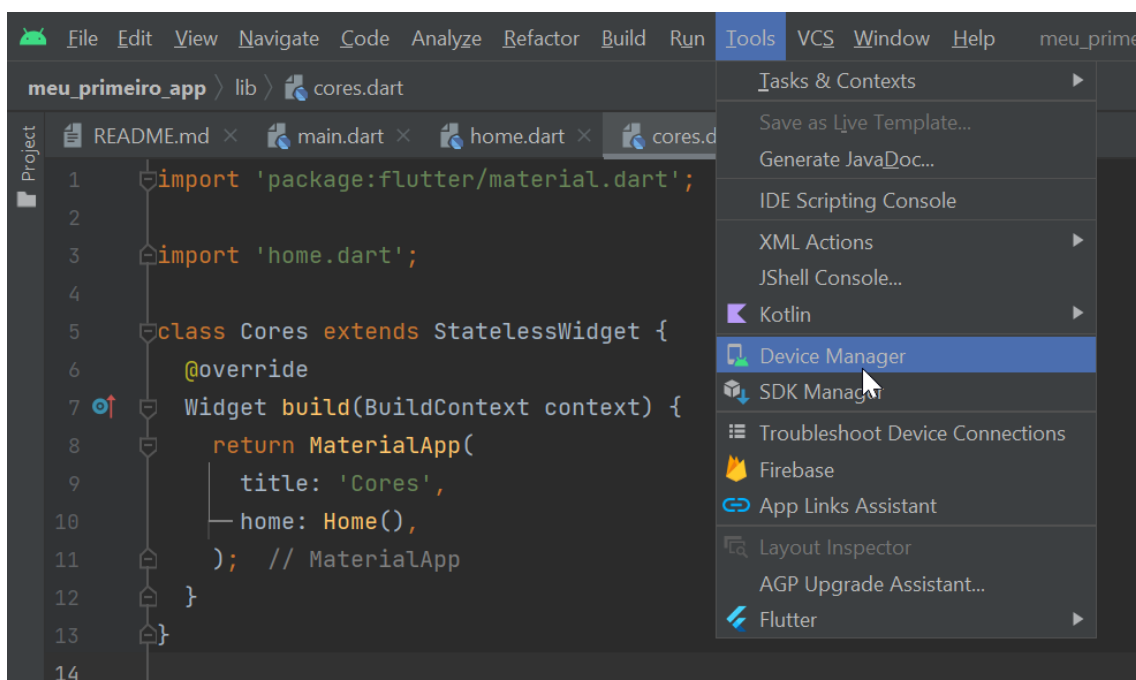


Figura 21: Device Manager. Fonte: autoria própria.

- II. Na sequência, clique em Create Virtual Device, conforme abaixo:

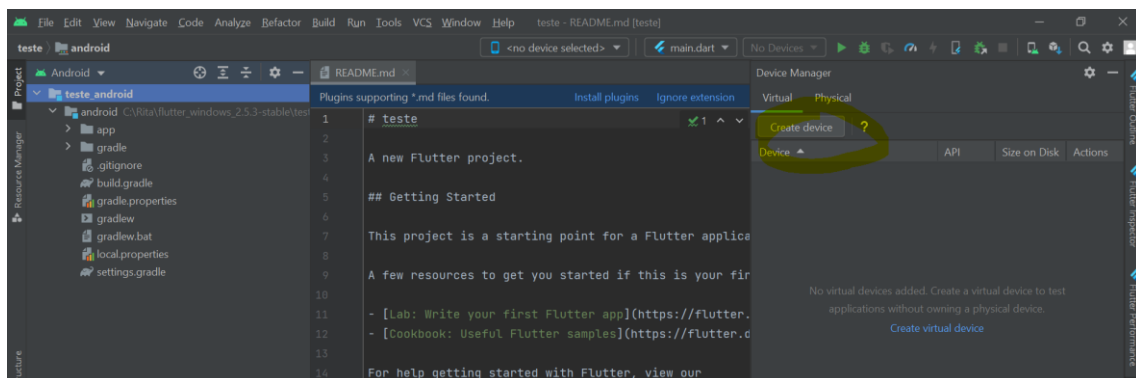


Figura 22: criando Device. Fonte: autoria própria.

- III. Agora, será possível escolher o tipo de dispositivo. A lista de categorias à esquerda apresenta todos os tipos de dispositivos disponíveis para emular. Selecione Phone e na lista de dispositivos, marque o Nexus 6 e clique em “Next”.

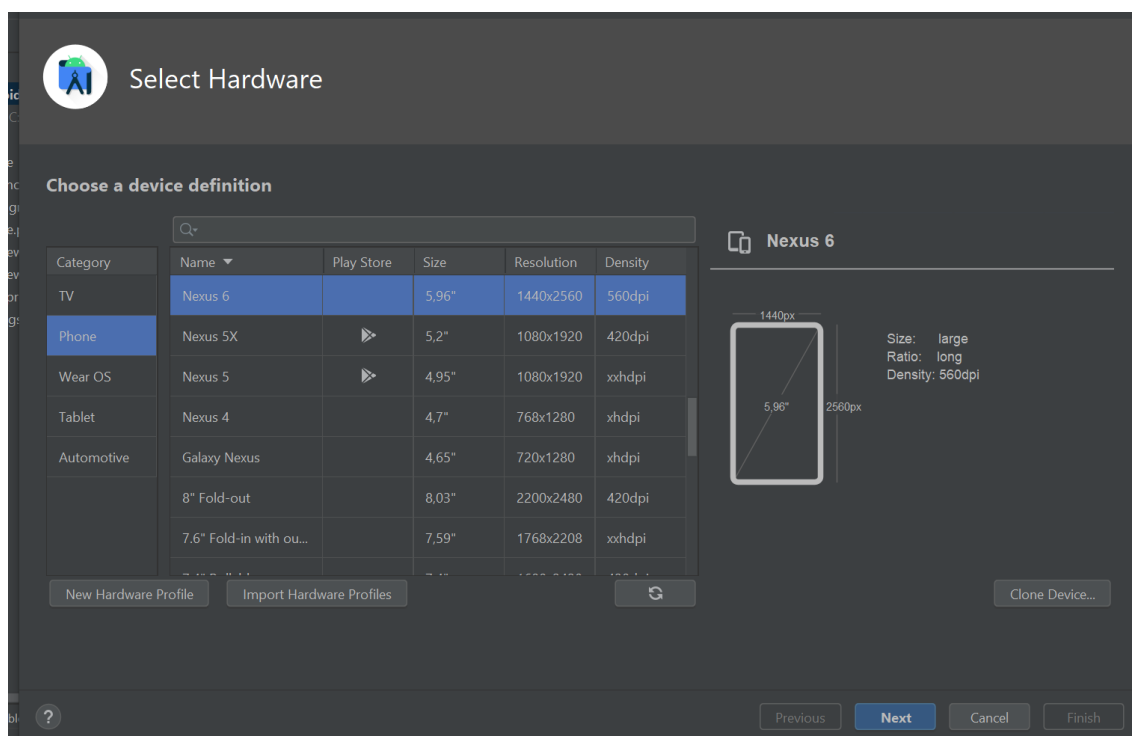


Figura 23: selecionando Dispositivo Nexus. Fonte: autoria própria.

- IV. O próximo passo, será a escolha da versão do Android no dispositivo virtual, que será executado. Selecione, por exemplo o Nougat API 25 e certifique-se o valor x86 na coluna ABI. Faça o download e na sequência clique em “Next”.

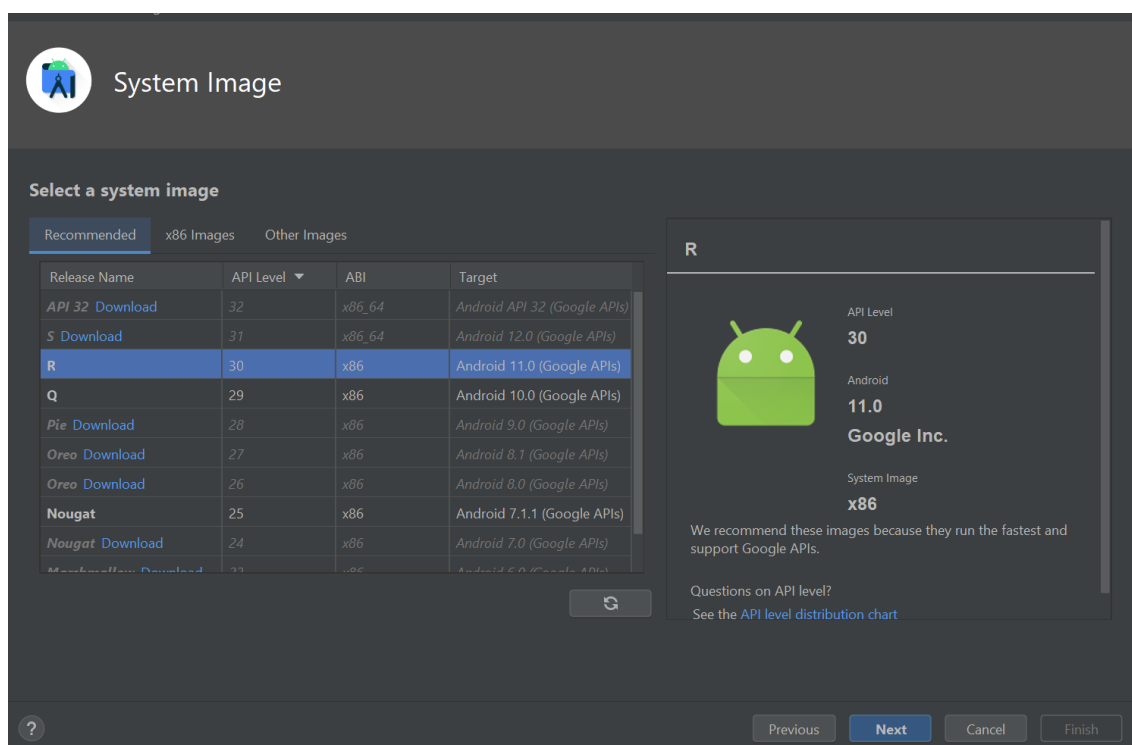


Figura 24: instalação Nougat 6. Fonte: autoria própria.

- V. A última tela permite confirmar suas escolhas e oferece opções para configurar algumas outras propriedades, como nome do dispositivo, orientação de inicialização. Por enquanto, use os padrões e clique em Finish:

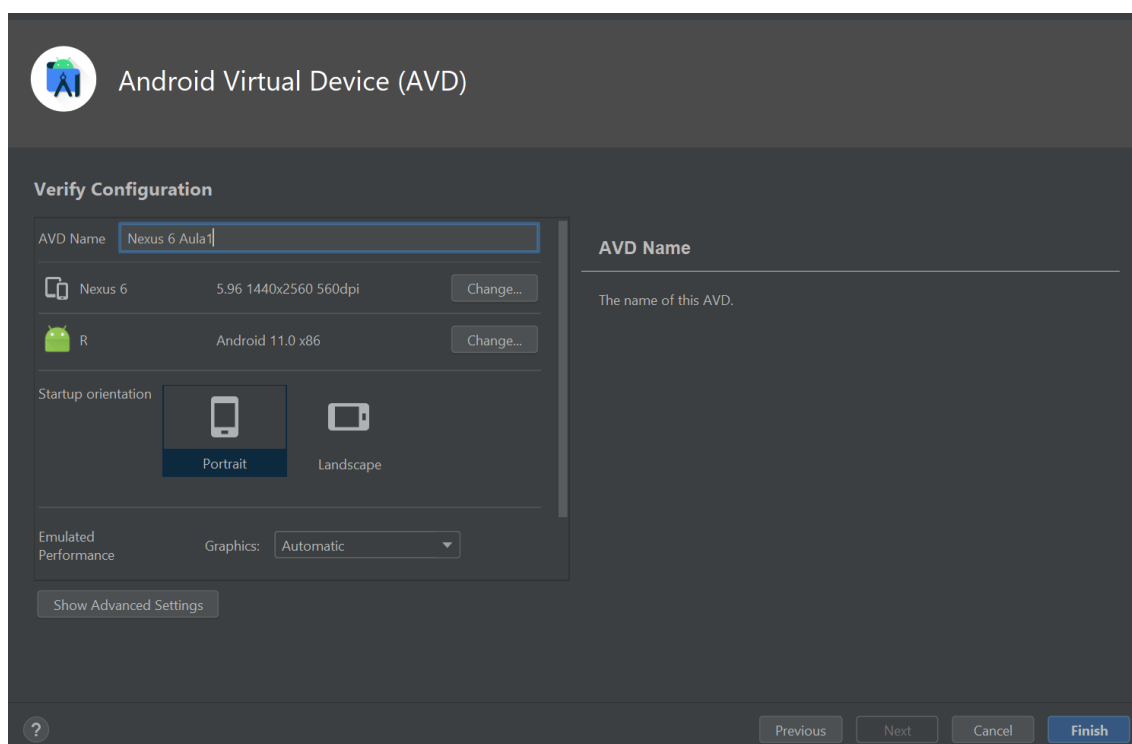


Figura 25: configuração final do AVD Manager. Fonte: autoria própria.

VI. O Emulador aparecerá na tela do AVD Manager, conforme abaixo:

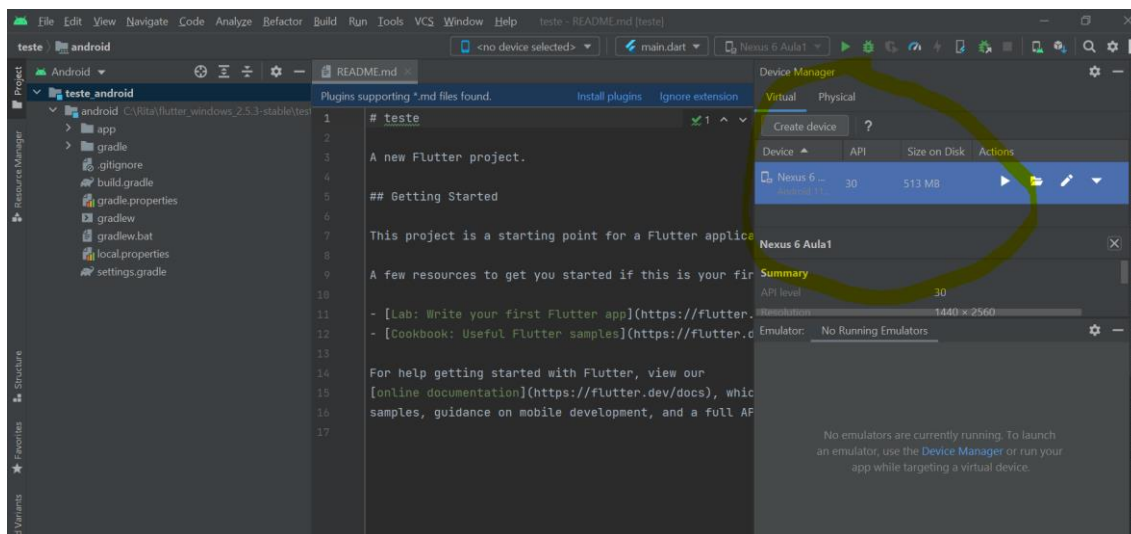


Figura 26: emulador configurado - AVD Manager. Fonte: autoria própria.

VII. No canto direito de cada emulador há uma coluna “Actions”, clique no ícone para executar o emulador, na sequência aparecerá a seguinte tela:

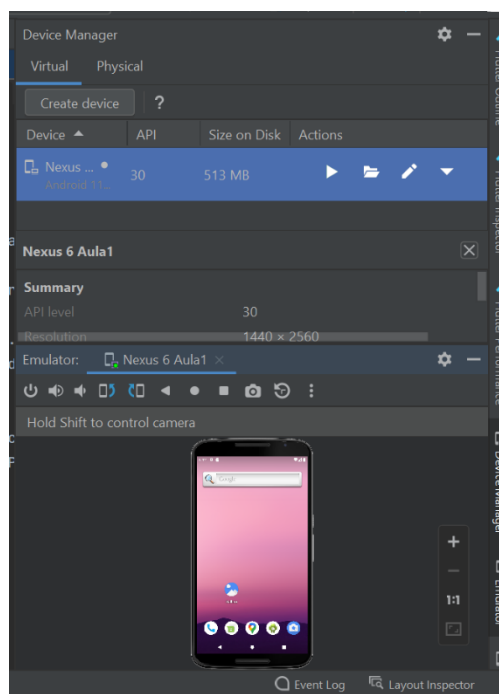


Figura 27: emulador AVD Manager em execução. Fonte: autoria própria.

VIII. Para aumentar a visualização do emulador, clique em +, conforme abaixo

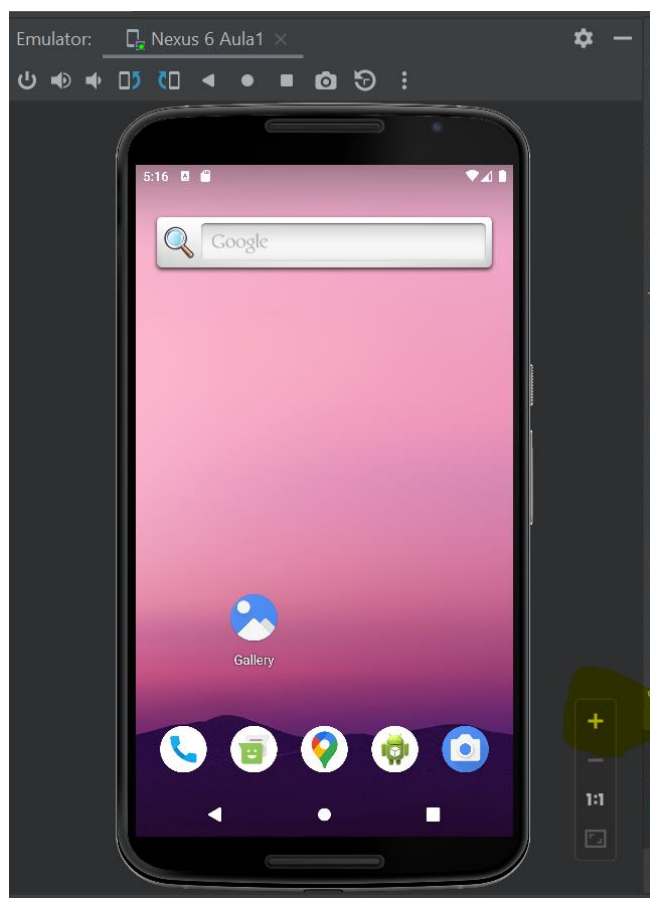


Figura 28: emulador AVD Manager em execução. Fonte: autoria própria.

1.5 Instalação Licenças Android

As licenças do Android se fazem necessárias no processo para sanar alguns erros como:

- *License status is unknown*
- *Android license status unknown*
- *Android sdkmanager tool not found*
- *The system cannot find the path specified*

A seguir, os passos e configurações necessárias:

- I. Clique no menu principal em Tools, SDK Manager:

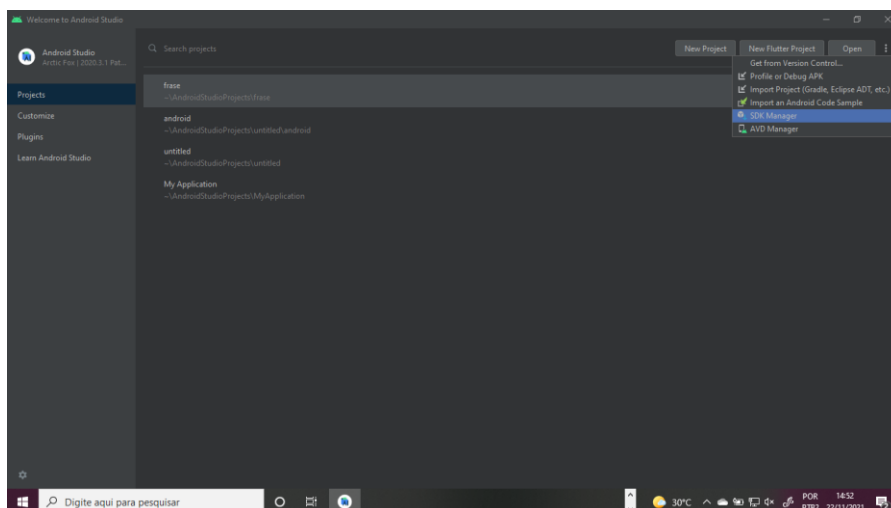


Figura 29: SDK Manager. Fonte: autoria própria.

- II. Em SDK Manager, acesse a guia SDK tools e desmarque a opção hide obsolete package:

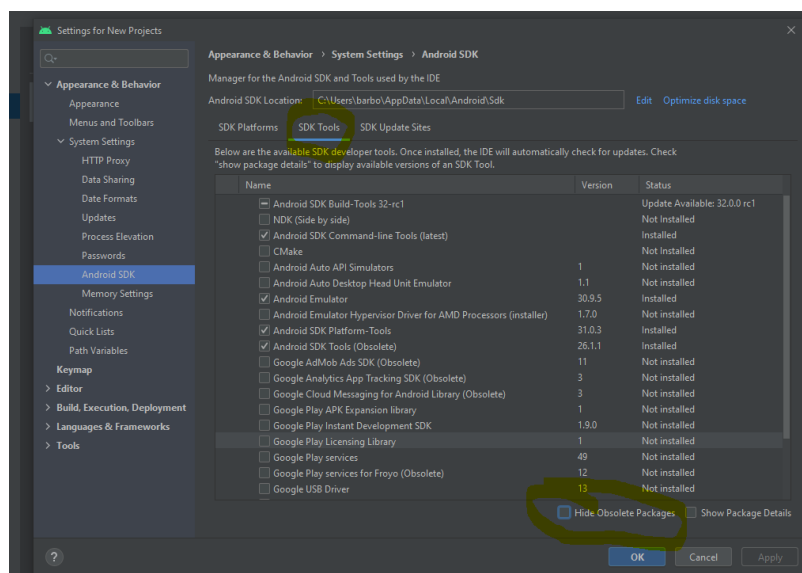


Figura 30: SDK Tools - hide obsolete package. Fonte: autoria própria.

III. Escolha o pacote Android SDK tools obsoleto e clicar em Apply:

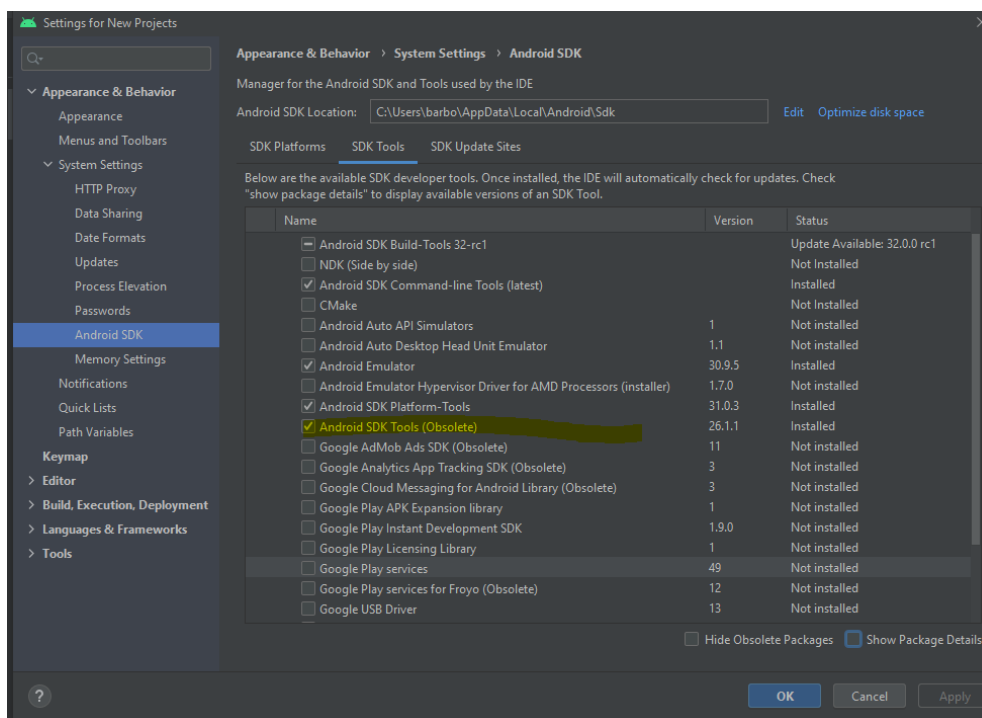


Figura 31: SDK Tools - hide obsolete package. Fonte: autoria própria.

IV. Confirmar o Download do pacote, clicar em Android SDK Tools com Ok:

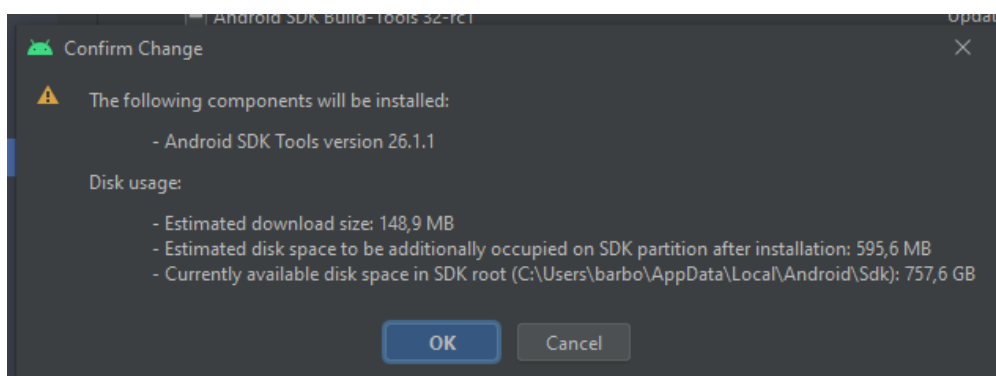


Figura 32: SDK Tools – Download. Fonte: autoria própria.

- V. Para finalizar a instalação do pacote, basta clicar em Finish:

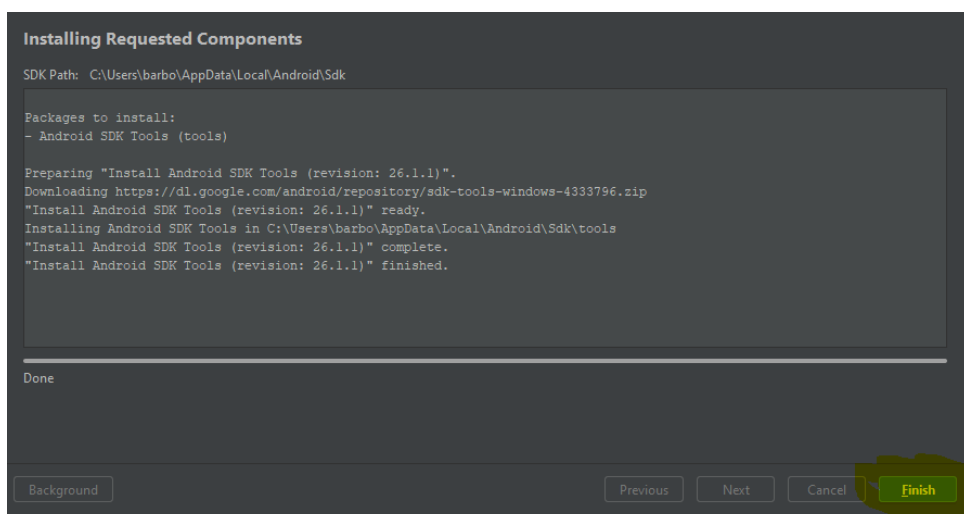


Figura 33: fim da instalação. Fonte: autoria própria.

- VI. Na janela anterior a instalação, click em Ok para fechar o SDK Tools:

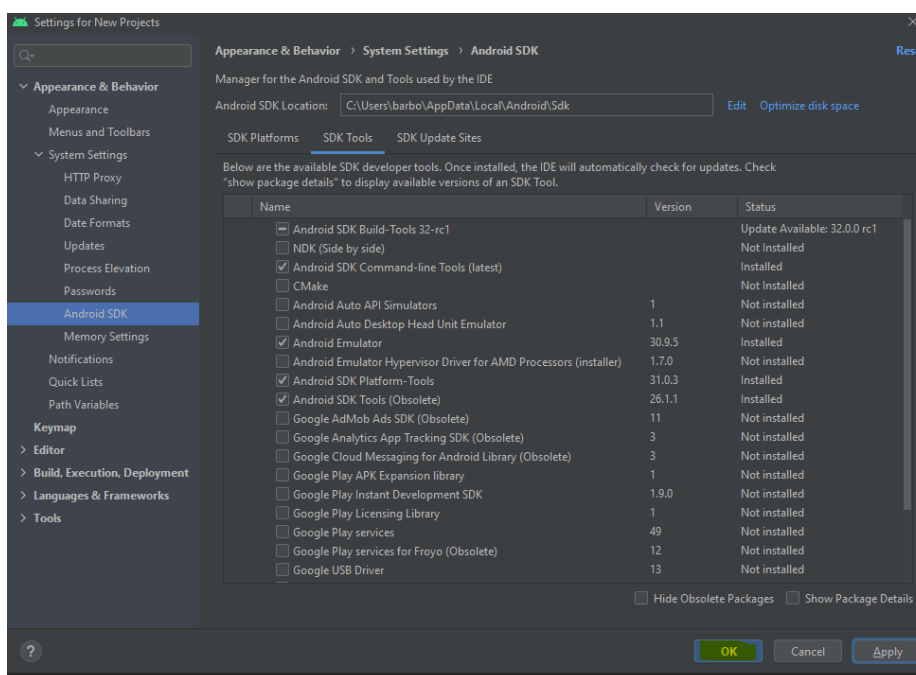


Figura 34: fim da instalação. Fonte: autoria própria.

1.6 Instalação e Configuração do VS Code

Agora que já temos as licenças instaladas, vamos Instalar e Configurar Visual Studio Code.

Esse passo é Opcional, visto que se você quiser utilizar o Android Studio sem problemas. Porém, o Android Studio exige mais recursos de *hardware* do computador e se você possui um computador com pouca capacidade de processamento e memória, o uso do VS Code é uma alternativa interessante. Os passos a seguir valem também para as plataformas Linux e Mac.

- I. Faça o download a partir do endereço:
<https://code.visualstudio.com/download>
- II. Para nosso aprendizado, escolha o Sistema Operacional Windows, conforme abaixo:

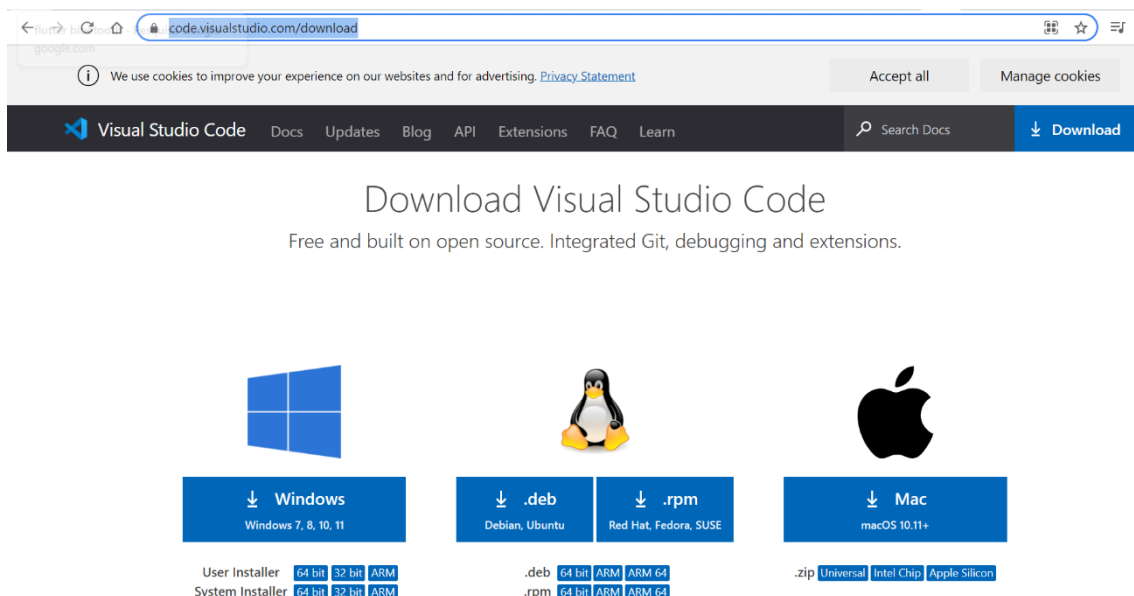


Figura 35: site para download VS Code. Fonte: Visual Studio.

- III. Após a instalação, vamos instalar as extensões do Flutter e Dart, Clique em configurações, e digite as extensões, uma de cada vez:

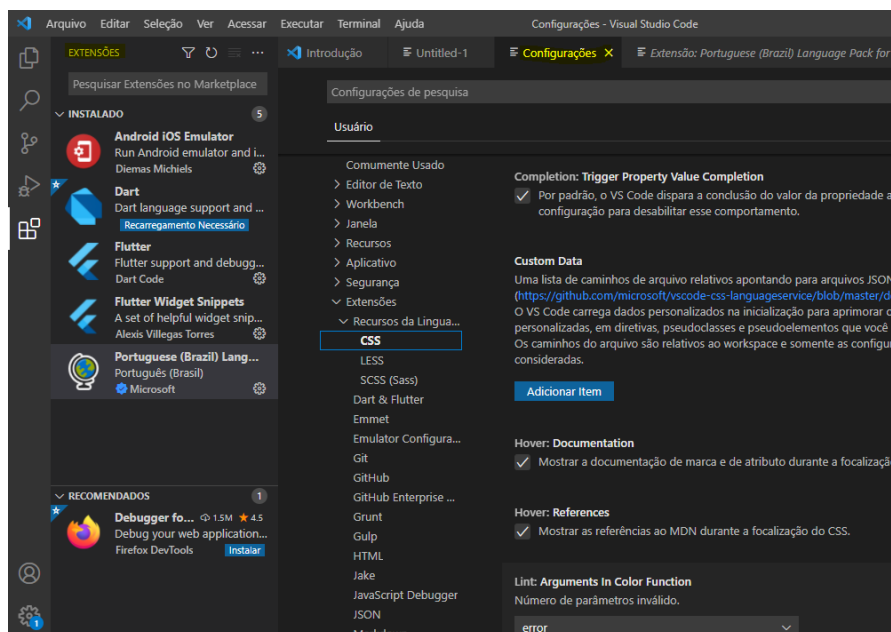


Figura 36: instalação de Extensões VS Code. Fonte: autoria própria.

1.7 Configuração do Flutter

No início realizamos a instalação do Flutter, e agora, com as licenças do Android SDK instaladas, podemos realizar as devidas configurações no Flutter.

- I. No seu computador, vá até o diretório onde foi instalado o Flutter, e execute o arquivo flutter_console.bat.

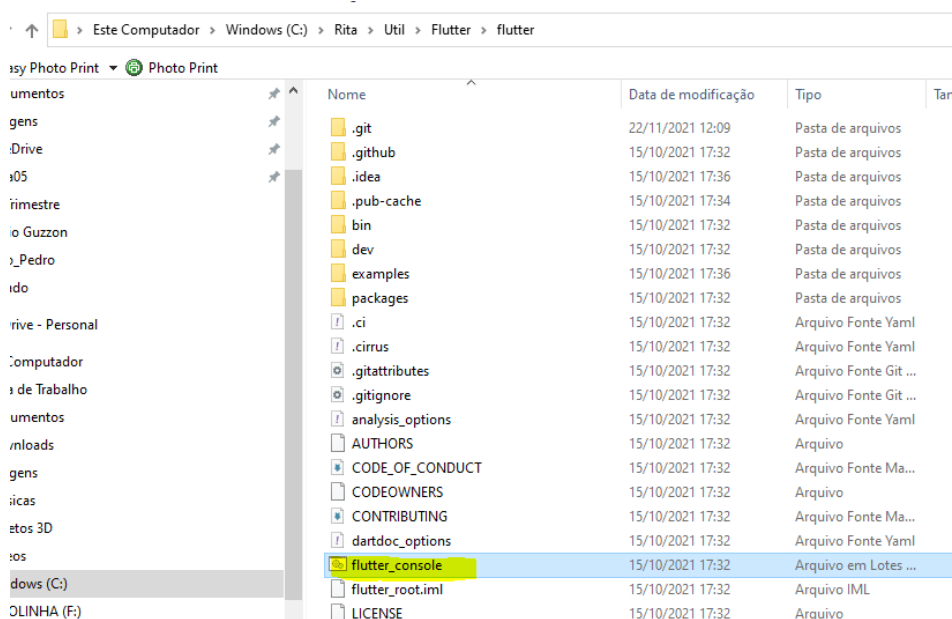
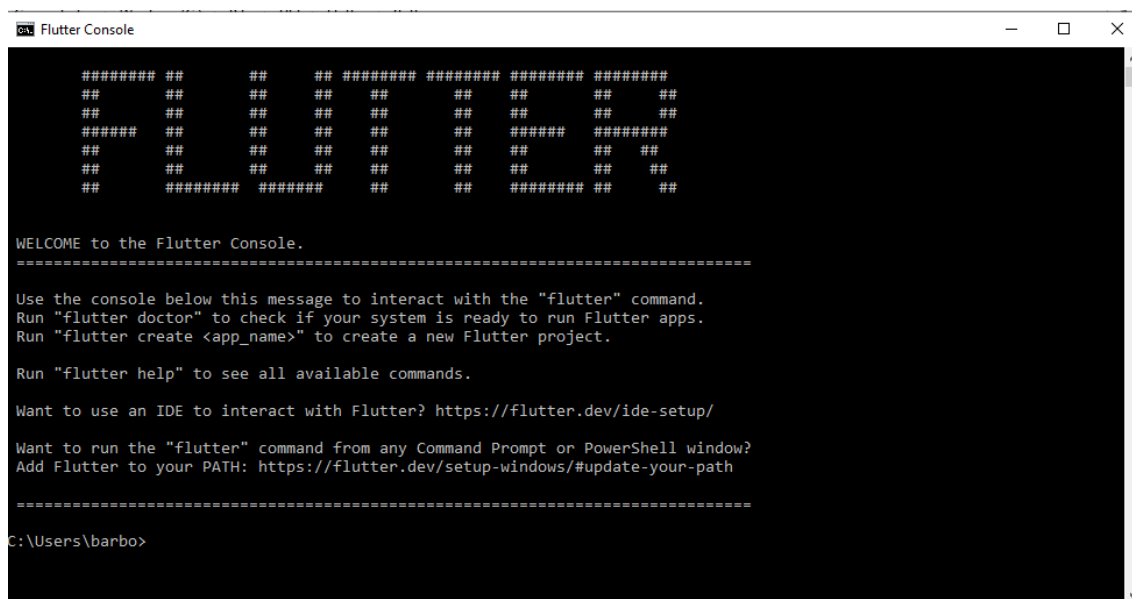


Figura 37: arquivo flutter_console.bat. Fonte: autoria própria.



```
Flutter Console

#####  ##      ##      #####  #####  #####  #####
##      ##      ##      ##      ##      ##      ##
##      ##      ##      ##      ##      ##      ##
#####  ##      ##      ##      #####  #####
##      ##      ##      ##      ##      ##      ##
##      ##      ##      ##      ##      ##      ##
##      #####  #####  ##      ##      #####  ##

WELCOME to the Flutter Console.
=====

Use the console below this message to interact with the "flutter" command.
Run "flutter doctor" to check if your system is ready to run Flutter apps.
Run "flutter create <app_name>" to create a new Flutter project.

Run "flutter help" to see all available commands.

Want to use an IDE to interact with Flutter? https://flutter.dev/ide-setup/

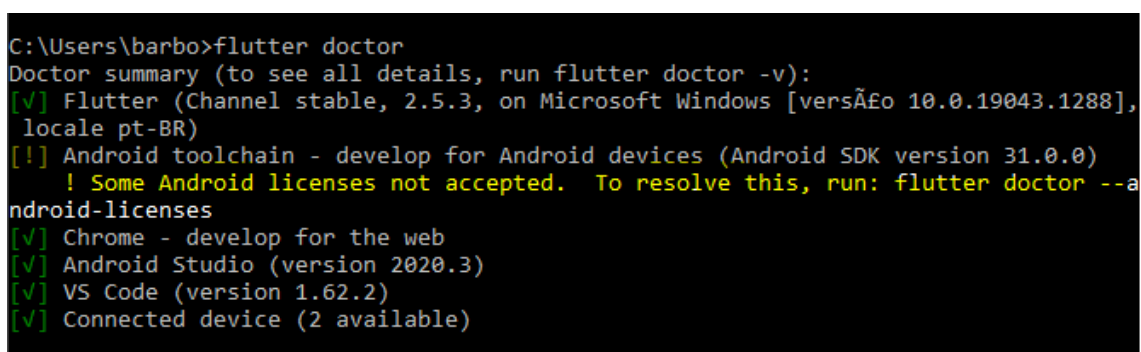
Want to run the "flutter" command from any Command Prompt or PowerShell window?
Add Flutter to your PATH: https://flutter.dev/setup-windows/#update-your-path

=====

C:\Users\barbo>
```

Figura 38: arquivo flutter_console.bat executado. Fonte: autoria própria.

- II. O próximo passo será averiguar se existem erros nas instalações do Android e demais programas, para isso, digite o comando: flutter doctor
- III. Caso a execução do comando apresente algum erro, esse será relacionado a licença do Android, como na figura abaixo:



```
C:\Users\barbo>flutter doctor
Doctor summary (to see all details, run flutter doctor -v):
[✓] Flutter (Channel stable, 2.5.3, on Microsoft Windows [versÃo 10.0.19043.1288],
    locale pt-BR)
[!] Android toolchain - develop for Android devices (Android SDK version 31.0.0)
    ! Some Android licenses not accepted. To resolve this, run: flutter doctor --a
ndroid-licenses
[✓] Chrome - develop for the web
[✓] Android Studio (version 2020.3)
[✓] VS Code (version 1.62.2)
[✓] Connected device (2 available)
```

Figura 39: erro comando Flutter Doctor. Fonte: autoria própria.

- IV. Para resolver esse problema, execute o comando: flutter doctor – android-licenses.
- V. Após execução do comando acima (flutter doctor –android-licenses), será apresentado telas em espera para dar continuidade a

instalação. Pressione Y a cada tela de espera, esta será a confirmação do aceite de configuração das instalações da licença.

- VI. Finalizado o comando, execute novamente o flutter doctor e você terá o seguinte *status*:

```
C:\Users\barbo>flutter doctor
Doctor summary (to see all details, run flutter doctor -v):
[✓] Flutter (Channel stable, 2.5.3, on Microsoft Windows [versão 10.0.19043.1288],
    locale pt-BR)
[✓] Android toolchain - develop for Android devices (Android SDK version 31.0.0)
[✓] Chrome - develop for the web
[✓] Android Studio (version 2020.3)
[✓] VS Code (version 1.62.2)
[✓] Connected device (2 available)

• No issues found!
C:\Users\barbo>
```

Figura 40: erro Flutter Doctor solucionado. Fonte: autoria própria.

1.8 Criação do 1º. Projeto – Hello World – Exercício Prático

Antes de criarmos nosso projeto Hello World, vamos configurar o emulador do Flutter ainda na linha de comando, dentro da console do Flutter, conforme abaixo:

- I. Digite o comando flutter emulators:

```
C:\Users\barbo>flutter emulators
1 available emulator:

Nexus_6_API_22 • Nexus 6 API 22 • Google • android

To run an emulator, run 'flutter emulators --launch <emulator id>'.
To create a new emulator, run 'flutter emulators --create [--name xyz]'.

You can find more information on managing emulators at the links below:
  https://developer.android.com/studio/run/managing-avds
  https://developer.android.com/studio/command-line/avdmanager

C:\Users\barbo>
```

Figura 41: Flutter emuladores. Fonte: autoria própria.

- II. O próximo passo será digitar o comando flutter emulators --launch <emulator id>. Na prática, ficará flutter emulators --launch Nexus_6_API_22:

```
C:\Users\barbo>flutter emulators --launch Nexus_6_API_22
C:\Users\barbo>
```

Figura 42: Flutter emulators launch. Fonte: autoria própria.

- III. Emulador configurado, vamos agora criar no 1º. Projeto Hello World. Ainda na console do Flutter, digite flutter create hello_world, e veja a criação do projeto, conforme abaixo:

```
C:\Users\barbo>flutter create hello_world
Creating project hello_world...
hello_world\lib\main.dart (created)
hello_world\pubspec.yaml (created)
hello_world\README.md (created)
hello_world\test\widget_test.dart (created)
hello_world\.gitignore (created)
hello_world\.idea\libraries\Dart_SDK.xml (created)
hello_world\.idea\libraries\KotlinJavaRuntime.xml (created)
hello_world\.idea\modules.xml (created)
hello_world\.idea\runConfigurations\main_dart.xml (created)
hello_world\.idea\workspace.xml (created)
hello_world\.metadata (created)
hello_world\analysis_options.yaml (created)
hello_world\android\app\build.gradle (created)
hello_world\android\app\src\main\kotlin\com\example\hello_world\MainActivity.kt (created)
hello_world\android\build.gradle (created)
hello_world\android\hello_world_android.iml (created)
hello_world\android\.gitignore (created)
hello_world\android\app\src\debug\AndroidManifest.xml (created)
hello_world\android\app\src\main\AndroidManifest.xml (created)
hello_world\android\app\src\main\res\drawable\launch_background.xml (created)
hello_world\android\app\src\main\res\drawable-v21\launch_background.xml (created)
hello_world\android\app\src\main\res\mipmap-hdpi\ic_launcher.png (created)
hello_world\android\app\src\main\res\mipmap-mdpi\ic_launcher.png (created)
hello_world\android\app\src\main\res\mipmap-xhdpi\ic_launcher.png (created)
```

Figura 43: criação do Projeto Hello World. Fonte: autoria própria.

```

hello_world\android\app\src\main\res\mipmap-mdpi\ic_launcher.png (created)
hello_world\android\app\src\main\res\mipmap-xhdpi\ic_launcher.png (created)
hello_world\android\app\src\main\res\mipmap-xxhdpi\ic_launcher.png (created)
hello_world\android\app\src\main\res\mipmap-xxxhdpi\ic_launcher.png (created)
hello_world\android\app\src\main\res\values\styles.xml (created)
hello_world\android\app\src\main\res\values-night\styles.xml (created)
hello_world\android\app\src\profile\AndroidManifest.xml (created)
hello_world\android\gradle\wrapper\gradle-wrapper.properties (created)
hello_world\android\gradle.properties (created)
hello_world\android\settings.gradle (created)
hello_world\ios\Runner\AppDelegate.swift (created)
hello_world\ios\Runner\Runner-Bridging-Header.h (created)
hello_world\ios\Runner.xcodeproj\project.pbxproj (created)
hello_world\ios\Runner.xcodeproj\xcshareddata\xcschemes\Runner.xcscheme (created)
hello_world\ios\gitignore (created)
hello_world\ios\Flutter\AppFrameworkInfo.plist (created)
hello_world\ios\Flutter\Debug.xcconfig (created)
hello_world\ios\Flutter\Release.xcconfig (created)
hello_world\ios\Runner\Assets.xcassets\AppIcon.appiconset\Contents.json (created)
hello_world\ios\Runner\Assets.xcassets\AppIcon.appiconset\Icon-App-1024x1024@1x.png (created)
hello_world\ios\Runner\Assets.xcassets\AppIcon.appiconset\Icon-App-20x20@1x.png (created)
hello_world\ios\Runner\Assets.xcassets\AppIcon.appiconset\Icon-App-20x20@2x.png (created)
hello_world\ios\Runner\Assets.xcassets\AppIcon.appiconset\Icon-App-20x20@3x.png (created)
hello_world\ios\Runner\Assets.xcassets\AppIcon.appiconset\Icon-App-29x29@1x.png (created)
hello_world\ios\Runner\Assets.xcassets\AppIcon.appiconset\Icon-App-29x29@2x.png (created)
hello_world\ios\Runner\Assets.xcassets\AppIcon.appiconset\Icon-App-29x29@3x.png (created)
hello_world\ios\Runner\Assets.xcassets\AppIcon.appiconset\Icon-App-40x40@1x.png (created)
hello_world\ios\Runner\Assets.xcassets\AppIcon.appiconset\Icon-App-40x40@2x.png (created)
hello_world\ios\Runner\Assets.xcassets\AppIcon.appiconset\Icon-App-40x40@3x.png (created)
hello_world\ios\Runner\Assets.xcassets\AppIcon.appiconset\Icon-App-60x60@2x.png (created)

```

Figura 44: criação do Projeto Hello World – continuação. Fonte: autoria própria.

```

hello_world\ios\Runner\Assets.xcassets\AppIcon.appiconset\Icon-App-60x60@3x.png (created)
hello_world\ios\Runner\Assets.xcassets\AppIcon.appiconset\Icon-App-76x76@1x.png (created)
hello_world\ios\Runner\Assets.xcassets\AppIcon.appiconset\Icon-App-76x76@2x.png (created)
hello_world\ios\Runner\Assets.xcassets\AppIcon.appiconset\Icon-App-83.5x83.5@2x.png (created)
hello_world\ios\Runner\Assets.xcassets\LaunchImage.imageset\Contents.json (created)
hello_world\ios\Runner\Assets.xcassets\LaunchImage.imageset\LaunchImage.png (created)
hello_world\ios\Runner\Assets.xcassets\LaunchImage.imageset\LaunchImage@2x.png (created)
hello_world\ios\Runner\Assets.xcassets\LaunchImage.imageset\LaunchImage@3x.png (created)
hello_world\ios\Runner\Assets.xcassets\LaunchImage.imageset\README.md (created)
hello_world\ios\Runner\Base.lproj\LaunchScreen.storyboard (created)
hello_world\ios\Runner\Base.lproj\Main.storyboard (created)
hello_world\ios\Runner\Info.plist (created)
hello_world\ios\Runner.xcodeproj\project.xcworkspace\contents.xcworkspacedata (created)
hello_world\ios\Runner.xcodeproj\project.xcworkspace\xcshareddata\IDEWorkspaceChecks.plist (created)
hello_world\ios\Runner.xcodeproj\project.xcworkspace\xcshareddata\WorkspaceSettings.xcsettings (created)
hello_world\ios\Runner.xcworkspace\contents.xcworkspacedata (created)
hello_world\ios\Runner.xcworkspace\xcshareddata\IDEWorkspaceChecks.plist (created)
hello_world\ios\Runner.xcworkspace\xcshareddata\WorkspaceSettings.xcsettings (created)
hello_world\hello_world.iml (created)
hello_world\web\favicon.png (created)
hello_world\web\icons\Icon-192.png (created)
hello_world\web\icons\Icon-512.png (created)
hello_world\web\icons\Icon-maskable-192.png (created)
hello_world\web\icons\Icon-maskable-512.png (created)
hello_world\web\index.html (created)
hello_world\web\manifest.json (created)
Running "flutter pub get" in hello_world... 1.813ms
Wrote 81 files.

All done!
In order to run your application, type:

$ cd hello_world
$ flutter run

Your application code is in hello_world\lib\main.dart.

```

Figura 45: criação do Projeto Hello World – finalização. Fonte: autoria própria.

- IV. Projeto criado!! Agora iremos executar o código criado:
- V. Executar Hello World no Android Studio: abra o Android Studio e o projeto criado Hello World, conforme abaixo:

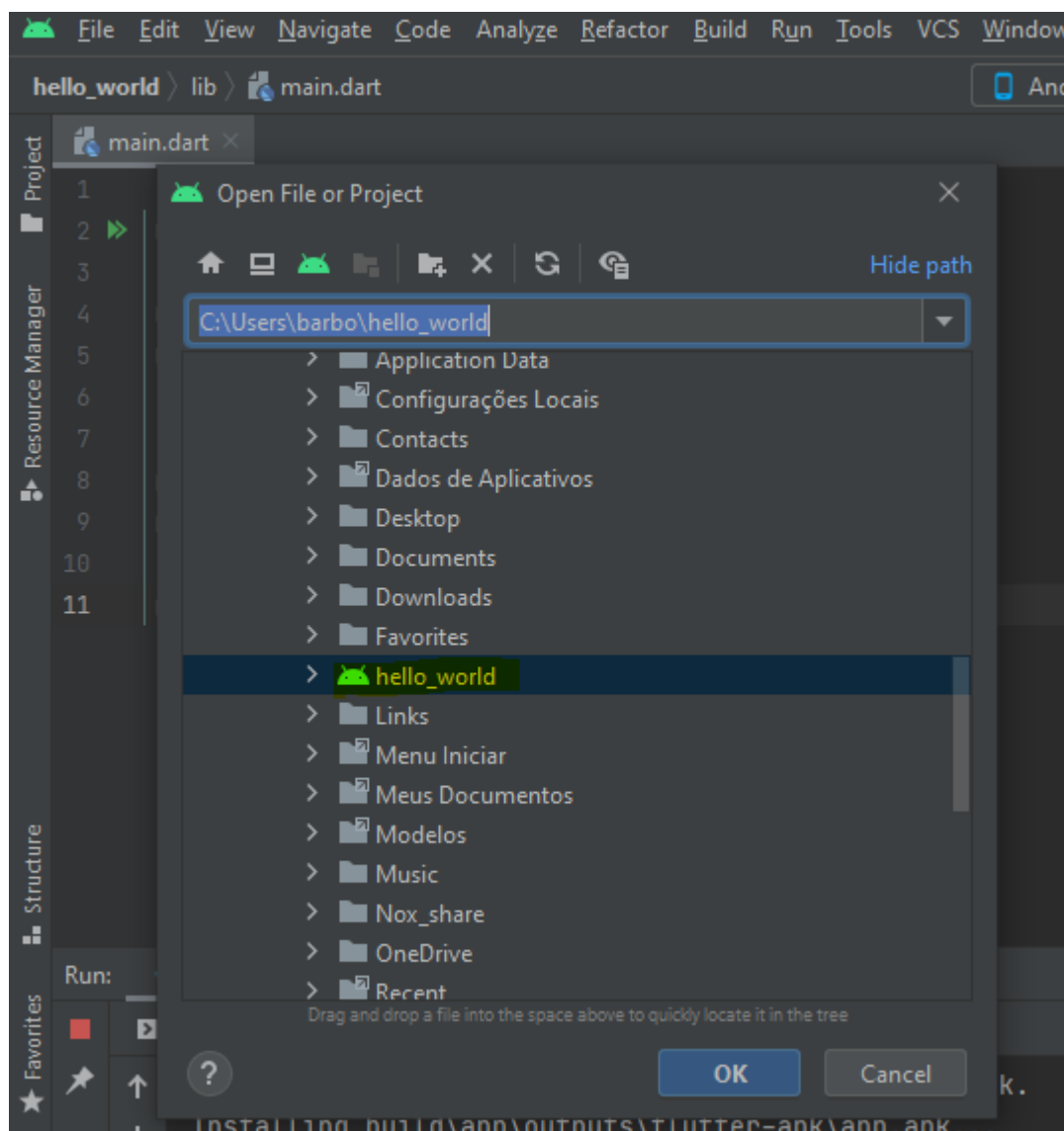


Figura 46: Android Studio, abrindo projeto Hello World. Fonte: autoria própria.

- VI. Para abrir o projeto criado, no menu principal, click em File, Open Project e localizem o arquivo hello_world e click em ok. O Android Studio abrirá o arquivo main.dart onde iremos escrever nosso código.

- VII. Nosso próximo passo será ativar o emulador dentro do Android Studio da seguinte maneira: no canto direito da janela principal, clique no ícone no formato de um celular, conforme abaixo:

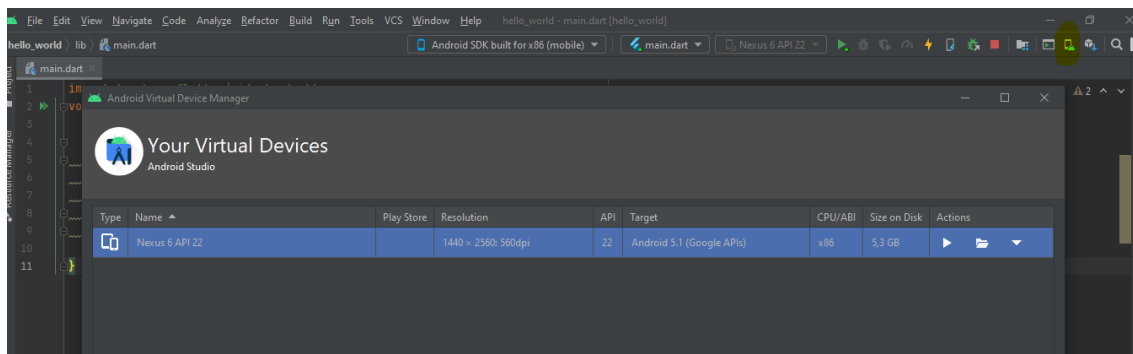


Figura 47: Android Studio, abrindo emulador Android. Fonte: autoria própria.

Note que aparecerá a tela do emulador em execução.

- VIII. Agora iremos voltar para a tela do nosso main.dart para finalmente executarmos nosso programa Hello World, digite o seguinte código no Android Studio:

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

void main() {
  runApp(
    Center(
      child: Text(
        'Hello, world!',
        textDirection: TextDirection.ltr,
      ),
    ),
  );
}
```

- IX. Depois do código digitado, vamos executá-lo, no menu principal, clique em Run e escolha Run Dart, O código será executado e depois de alguns poucos minutos o emulador apresentará a seguinte tela:

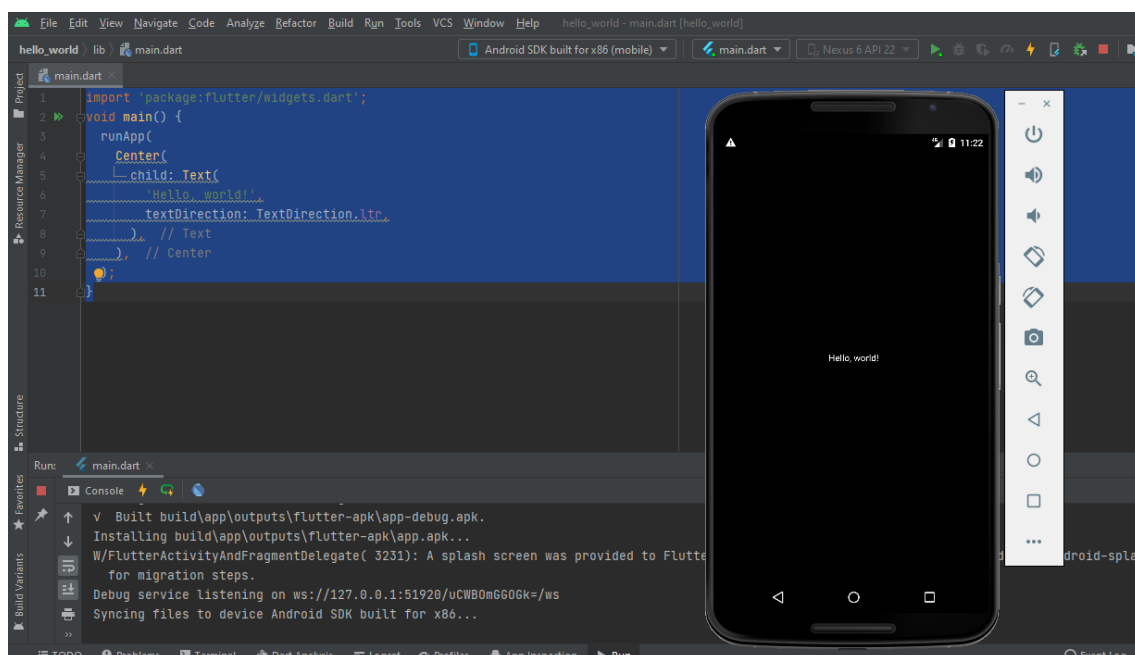


Figura 48: Hello World em execução. Fonte: autoria própria.



Importante frisarmos que para emulação do iOS, se faz necessário uma máquina virtual OSX ou um mac, com IDE XCode.

- X. Executar Hello World no VC Code-** criado nosso projeto Hello World, vamos testá-lo no VS Code, para isto, siga os seguintes passos: na console do Flutter, vá para o diretório onde foi criado o projeto Hello World. No nosso caso, **cd\hello_world**. Digite o comando **flutter clean** (limpa diretório). Na sequência, digite **flutter pub get** (atribui todas as dependências e libraries do flutter). E finalmente, vamos chamar o VS Code. digite o comando **code .**, a tela inicial do VS Code aparecerá da seguinte forma:

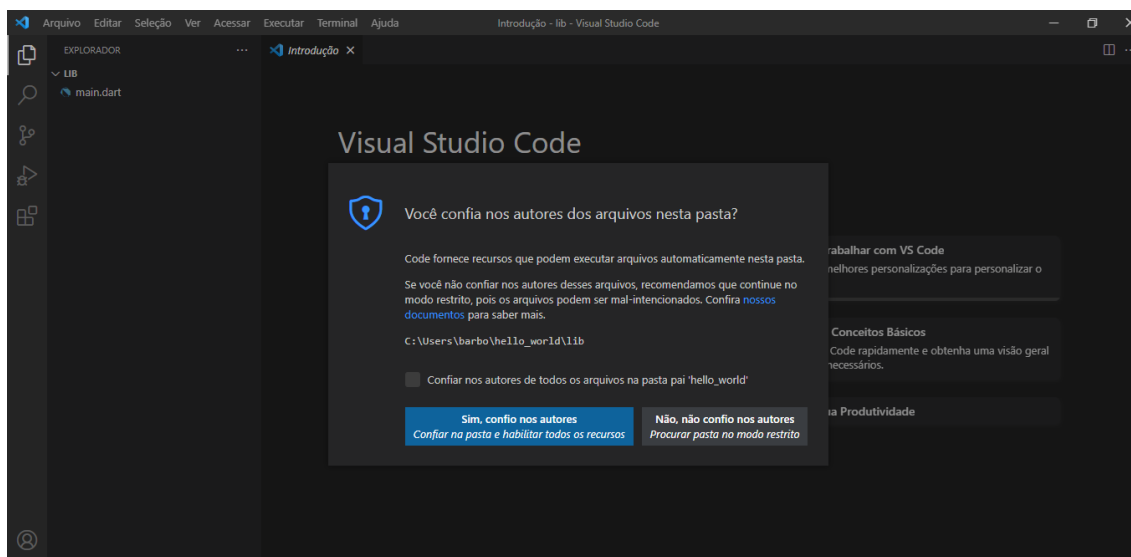


Figura 49: executando VS Code via console do Flutter. Fonte: autoria própria.

- XI. Clique para aceitar e confiar nos autores no VS Code, em seguida, Para editar o aplicativo, clique em: lib > main.dart. Veja que o Código que utilizamos no Android Studio é o mesmo:

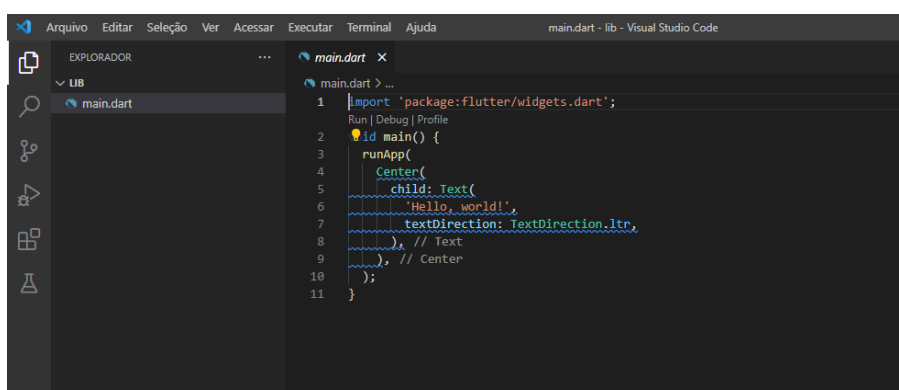


Figura 50: arquivo main.dart. Fonte: autoria própria.

- XII. Para iniciar o código, aperte as teclas CTRL + F5. No topo superior do Visual Studio Code abrirá uma pequena aba. Se você já possui emuladores configurados, eles devem aparecer nessa aba. Se não, basta selecionar a opção “Create New”:

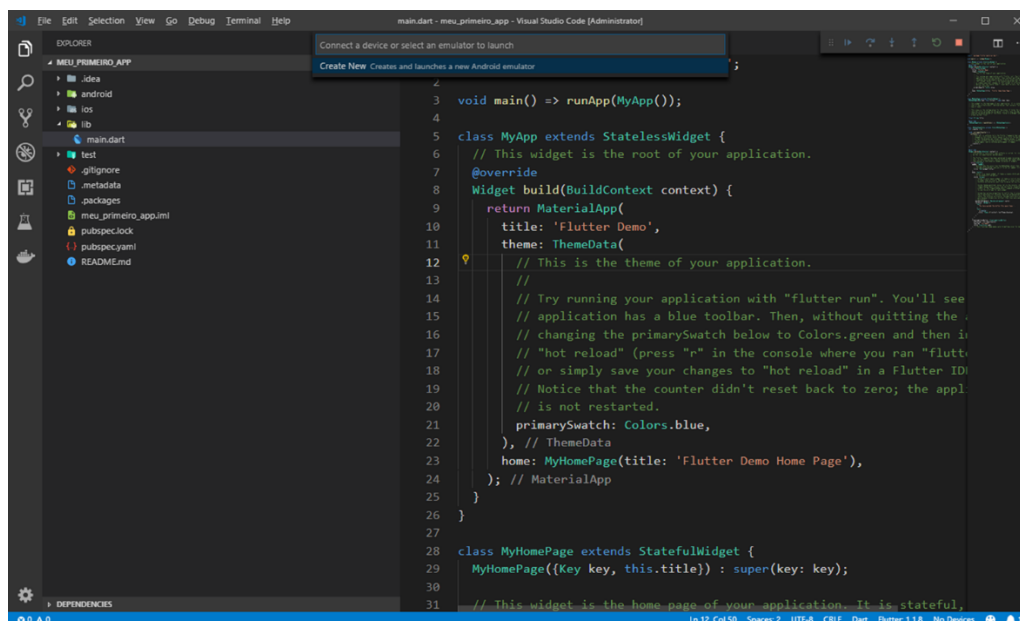


Figura 51: execução Arquivo main.dart. Fonte: autoria própria.

- XIII. O processo deve demorar alguns minutos, e você pode acompanhá-lo através da pequena aba aberta no canto inferior direito:

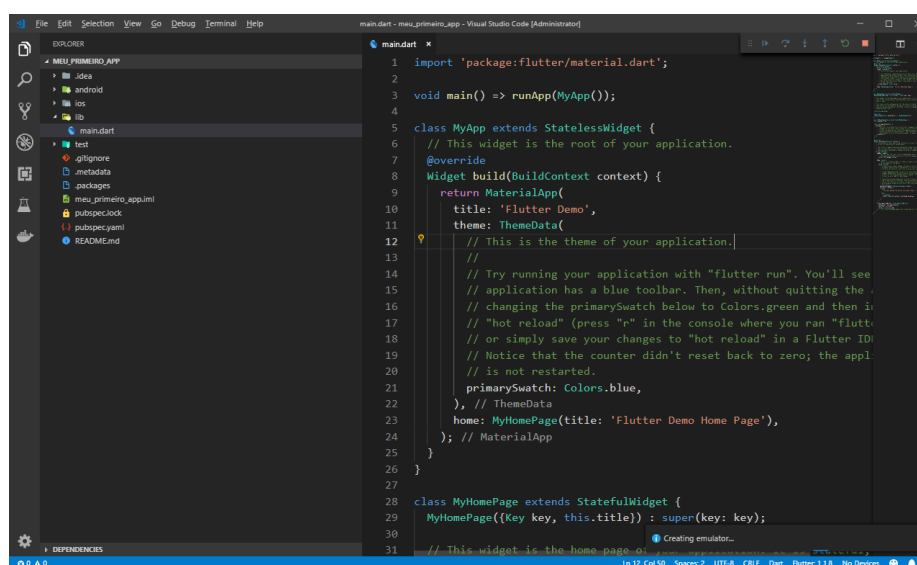


Figura 52: execução emulador. Fonte: autoria própria.

- XIV. Ao final do processo, o emulador abrirá automaticamente. Caso seu aplicativo não seja executado, execute novamente o comando CTRL + F5. Entretanto, como o emulador já estará aberto, não será necessário escolher nenhuma opção. O Flutter ficará responsável por

iniciar a aplicação no emulador e emitir todos os eventos e erros através da aba “DEBUG CONSOLE” no rodapé do Visual Studio Code:

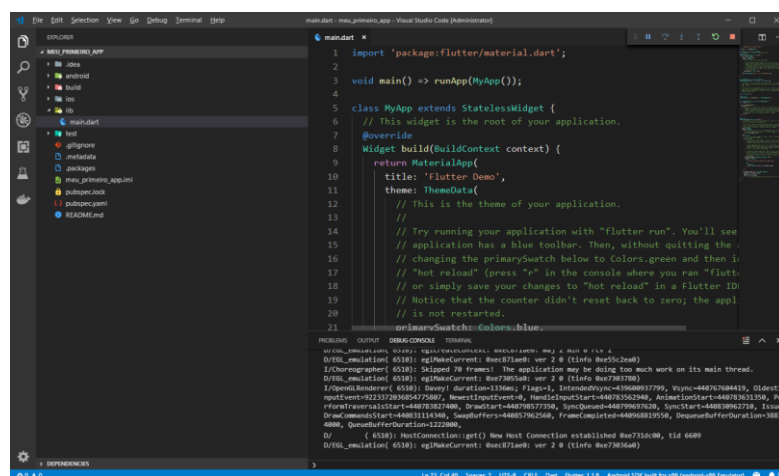


Figura 53: Debug Console do Flutter. Fonte: autoria própria.

Nesse momento, no emulador seu aplicativo já será executado:

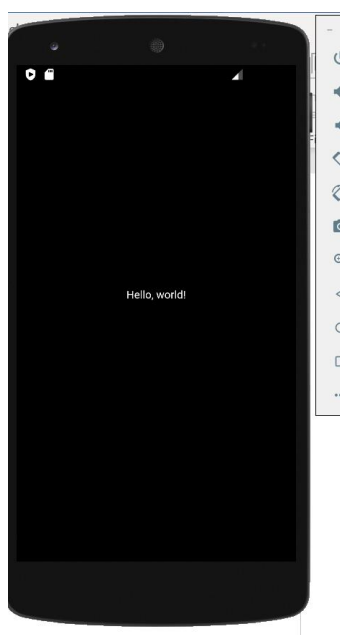


Figura 54: aplicativo criado automaticamente. Fonte: autoria própria.

Introdução ao DART

1.9 Histórico Dart

Dart (originalmente denominada Dash) é uma linguagem de programação voltada à web, desenvolvida pela Google. O objetivo da linguagem Dart inicialmente foi substituir o JavaScript como linguagem principal embutida nos navegadores.

Em novembro de 2013 foi lançada a primeira versão estável- Dart 1.0. e, em agosto de 2018 foi lançado o Dart 2.0, um reboot da linguagem, otimizado para o desenvolvimento client-side (executada no cliente) para Web e dispositivos móveis.

Abaixo algumas características da linguagem:

- A sintaxe é C-like, para quem tem experiência com C#, PHP ou Javascript, não terá dificuldades em aprender a linguagem;
- Segue o paradigma orientado a objetos;
- Todos os objetos herdam da classe Object;
- Fortemente tipada, significa que uma vez que a variável foi declarada com um tipo ela será até o seu fim do mesmo tipo e também normalmente possuem declaração explícita de tipo onde o tipo da variável deve ser especificado logo na sua declaração;
- Palavras reservadas são representadas por um underline () no início do nome de um atributo, método ou classe para torná-lo privado.
- Dart pode ser compilada em ahead-of-time (AOT - processo de compilação que ocorre antes da execução do aplicativo e não durante) e just-in-time (JIT - é a compilação de um programa em tempo de execução).

1.10 Variáveis

Variável é um local para armazenar temporariamente alguma informação. Os principais tipos de variáveis são:

```
void main() {  
    // Variável que armazena números inteiros  
    int idade = 33;  
    print("Idade: $idade");  
  
    // Variável que armazena números decimais  
    double raio = 10.25;  
    print("Raio: $raio");  
  
    // Variável que armazena caracteres e textos  
    String nome = "Kleber";  
    print("Ola $nome, seja bem vindo!");  
  
    // Variável que armazena verdadeiro ou false  
    bool ligado = true;  
    print("Ligado: $ligado");  
  
    // Variável que guarda uma lista genérica  
    List numerosGenericos = [10, "Kleber", true, 20];  
    print(numerosGenericos);  
  
    // Variável que guarda uma lista de números inteiros  
    List<int> numerosInteiros = [10, 20, 30, 40];  
    print(numerosInteiros);  
  
    // Variável que guarda um dicionário com chave e valor em formato texto  
    Map<String, String> nome_sobrenome = {"Kleber": "Andrade",  
    "Claudia": "Trevisan"};  
  
    // Variável sem tipo definido, seu tipo é igual ao tipo do primeiro valor que recebe  
    var sobrenome = nome_sobrenome[nome];  
    print("O sobrenome do $nome é $sobrenome");  
  
    // Constantes (valores imutáveis)  
    const double pi = 3.1416;  
    print("O valor de PI é $pi");  
  
    // Variável dinâmica (neste momento é do tipo inteiro pois recebeu o valor 10)
```

```
dynamic x = 10;
print(x);

// O tipo da variável pode ser alterada em tempo de execução
(agora é um texto)
x = "Curso";
print(x);
}
```

Figura 55: variáveis DART. Fonte. autoria própria.

O resultado será:

```
Idade: 33
Raio: 10.25
Ola Kleber, seja bem vindo!
Ligado: true
[10, Kleber, true, 20]
[10, 20, 30, 40]
O sobrenome do Kleber é Andrade
O valor de PI é 3.1416
10
Curso
```

Figura 56: variáveis DART – resultado. Fonte: Medium.

1.11 Operadores

Os tipos `int`, `double` e `num` (pode ser inteiro ou ponto flutuante) fornecem diversos métodos e propriedades que podem ser utilizados para a transformação e checagem de dados. Dispõem de capacidades para expressões utilizando os operadores `+` (adição), `—` (subtração), `*` (multiplicação), `/` (divisão) e `%` (resto da divisão).

1.12 Operadores Relacionais

Operadores relacionais são usados para comparações no Dart. O resultado de cada expressão é sempre um `bool` (booleano), ou seja terá o valor que é **true** ou **false**.

1.13 Condicionais

As Condicionais são utilizadas para tomada de decisão no código, sendo:

```
void main(){
    double media = 4.9;

    // IF (condição verdadeira) / ELSE
    if (media < 6.0){
        print("Reprovado!");
    } else {
        print("Aprovado!");
    }

    /* Podemos também utilizar IF TERNÁRIO
    * CONDIÇÃO ? RETORNO VERDADEIRO : RETORNO FALSO
    */
    print(media < 6.0 ? "Reprovado!" : "Aprovado");

    /* Toda variável declarada e que não recebe valor,
    automaticamente é nula
    * VARIÁVEL ?? RETORNO CASO SEJA NULO
    */
    String linguagem;
    print(linguagem ?? "Não Informado");

    linguagem = "Dart";
    print(linguagem ?? "Não Informado");

    /* SWITCH / CASE / DEFAULT
    * Utilizado geralmente quando temos constantes
    * Cada cláusula de case não vazia termina com uma instrução
    break, como regra.
    */
    switch(linguagem){
        case "Dart":
            print("É Dart!");
            break;
        case "Java":
            print("É Java!");
            break;
        case "C#":
            print("É C#!");
            break;
        default:
            print("Não sabe no que programa");
    }
}
```

Figura 57: condicionais DART. Fonte: autoria própria.

O resultado de execução deste código será:

```
Reprovado!
Reprovado!
Não Informado
Dart
É Dart!
```

Figura 58: condicionais DART – resultado. Fonte: Medium.

1.14 Repetições

As Repetições são utilizadas para processos de repetição no código, usando os comandos FOR, WHILE, DO/WHILE ou FOREACH sendo:

```
void main(){
    // Repetição de 0 a 5 (conhecemos o número inicial e final)
    // FOR (INICIO; CONDIÇÃO; INCREMENTO)
    for(int i = 0; i < 5; i++){
        print(i);
    }

    // Repetição de 0 a 5
    // INICIO; WHILE (CONDICAO){ INCREMENTO; }
    // Teste condicional no início
    int j = 0;        // Início
    while(j < 5){     // Condição
        print(j);
        j++;         // Incremento
    }

    // Repetição de 0 a 5
    // INICIO; DO { INCREMENTO; } WHILE(CONDICAO);
    // Teste condicional no final
    int k = 0;        // Início
    do {
        print(k);
        k++;         // Incremento
    } while(k < 5);   // Condição

    // Conjunto de números (Lista)
    List numeros = [0, 1, 2, 3, 4];
```

```
// FOREACH
// FOR (VARIÁVEL DENTRO DO CONJUNTO)
for (int numero in numeros){
    print(numero);
}
}
```

Figura 59: repetições DART. Fonte: autoria própria.

Os resultados de execução desse código serão os valores 0, 1, 2, 3, 4 para cada tipo de repetição executada.

1.15 Funções

Funções são objetos e possuem um tipo- Função. Isso significa que as funções podem ser atribuídas a variáveis ou passadas como argumentos para outras funções.

```
// Função sem parâmetros
void escreverBemVindo(){
    print("Seja bem-vindo!");
}

// Quando a função só tem um comando interno, você pode usar desta
forma
void escreverDesculpas() => print("Desculpa, encontramos um
erro.");

// Função com passagem de parâmetros (podem ter quantos parâmetros
quiser)
void calcularSoma(double a, double b){
    double resultado = a + b;
    print(resultado);
}

// Função que retorna uma variável do tipo double
double calcularSubtracao(double a, double b){
    double resultado = a - b;
    return resultado;
}

// Exemplo reduzido de uma função que retorna valor
double calcularAreaCirculo(double raio) => 3.14 * raio * raio;
```



```
// Função com parâmetros opcionais (utiliza-se os parâmetros dentro
de chaves {})
void exibirNomeCursoIdade(String nome, {int idade, String curso}) {
    if(idade != null && curso != null) {
        print("$nome tem $idade anos e faz o curso de $curso.");
    } else if(idade == null && curso != null) {
        print("$nome faz o curso de $curso.");
    } else if(idade != null && curso == null) {
        print("$nome tem $idade anos.");
    } else {
        print("Ola $nome");
    }
}

// Passar funções como parâmetros
void calcular(double a, double b, Function funcao){
    funcao(a, b);
}

// Função principal
void main() {
    // Executando a função escreverBemVindo()
    escreverBemVindo();

    // Executando a função escreverDesculpas()
    escreverDesculpas();

    // Executando a função calcularSoma(a, b)
    calcularSoma(10, 20);

    // Executando a função calcularSubtracao(a, b)
    print(calcularSubtracao(10, 20));

    // Executando a função calcularAreaCirculo(raio)
    print(calcularAreaCirculo(10));

    // Executando a função exibirNomeCursoIdade(nome)
    exibirNomeCursoIdade("Kleber");

    // Executando a função exibirNomeCursoIdade(nome, idade)
    exibirNomeCursoIdade("Kleber", idade: 33);

    // Executando a função exibirNomeCursoIdade(nome, curso)
    exibirNomeCursoIdade("Kleber", curso: "Ciência da Computação");

    // Executando a função exibirNomeCursoIdade(nome, idade, curso)
    exibirNomeCursoIdade("Kleber", idade: 33, curso: "Ciência da
Computação");
}
```

```
// Executando a função calcular(a, b, função), como função foi
passada a calcularSoma(a,b)
calcular(30, 20, calcularSoma);

// Executando a função calcular(a, b, função), como função foi
criado uma função anônima(a,b)
calcular(30, 20, (a, b){
    var resultado = a * b;
    print(resultado);
});
}
```

Figura 60: funções DART. Fonte: autoria própria.

O resultado de execução deste código será:

```
Seja bem-vindo!
Desculpa, encontramos um erro.
30
-10
314
Ola Kleber
Kleber tem 33 anos.
Kleber faz o curso de Ciência da Computação.
Kleber tem 33 anos e faz o curso de Ciência da Computação.
50
600
```

Figura 61: funções DART – resultado. Fonte: Medium.

Widgets Básicos

1.16 O que são Widgets?

Widgets são atalhos que facilitam o acesso a aplicativos e ferramentas no celular. Todo aplicativo Flutter é um Widget formado de outras centenas de Widgets. Os Widgets do Flutter, foram inspirados no React.

Há apenas 02 tipos de Widgets: Stateless e Stateful.

- **Stateless** - Widget sem o controle de estado. Este tipo de widget não possibilita alterações dinâmicas, entenda-o como algo completamente estático. São utilizados para a criação de estruturas não mutáveis nos aplicativos (telas, menus, imagens etc.), ou seja, tudo que não envolva entradas de dados dos usuários, acessos a APIs e coisas que mudem ao longo do processo.
- **Stateful** – Esse Widgets são praticamente o oposto dos Stateless. Possuem estado e por conta disso, se tornam mutáveis. São elementos-chave para o desenvolvimento móvel da forma interativa.

Ilustraremos abaixo esses 02 tipos para melhor entendimento:

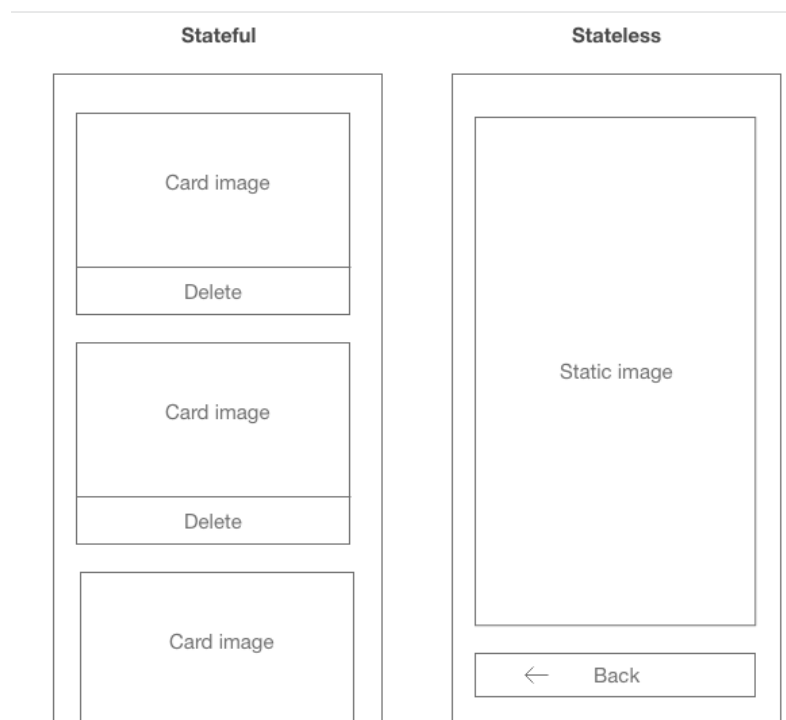


Figura 62: tipos Widgets - Stateless e Stateful. Fonte: Medium.

No lado esquerdo, há uma tela baseada em um widget stateful, ou seja, a tela será atualizada quando um cartão for excluído. Essa ação afetará seu estado, fazendo com que exista uma nova renderização. No lado direito, uma tela apenas exibe uma imagem e que permite uma mudança de rota, para que o usuário retorne à tela anterior, sem alteração de estado alguma.

Podemos classificar os widgets em dois grupos principais: **Layout** e **UI (user interface)**.

Widgets de Layout: são aqueles que se preocupam apenas em posicionar outros widgets, listamos abaixo alguns dos principais:

- Column
- Row
- ScaffoldStack

Widgets de Interface: são aqueles que efetivamente estão visíveis ao usuário, como:

- Text
- Raised

- ButtonSwitch

Destacamos também dois conjuntos de widgets amplamente utilizados, são eles: Material package, que segue as definições de layout do Material Design e o Cupertino package, seguindo as definições de design do iOS. Estes pacotes nos fornecem Widgets ready-to-use, bastando importá-los e utilizá-los no projeto.



Você pode ver mais detalhes sobre a documentação oficial de Widgets em: <https://flutter.dev/docs/development/ui/widgets-intro>

1.17 Layout

O núcleo do mecanismo de layout do Flutter são os widgets. Como já mencionamos, no Flutter, quase tudo é um widget até mesmo modelos de layout são widgets complexos.

Os widgets são organizados em uma árvore de widgets numa hierarquia : pai e filho. Toda a árvore de widgets é o que forma o layout visualiza na tel do celular. Abaixo temos um diagrama da árvore de widgets para esta interface do usuário:

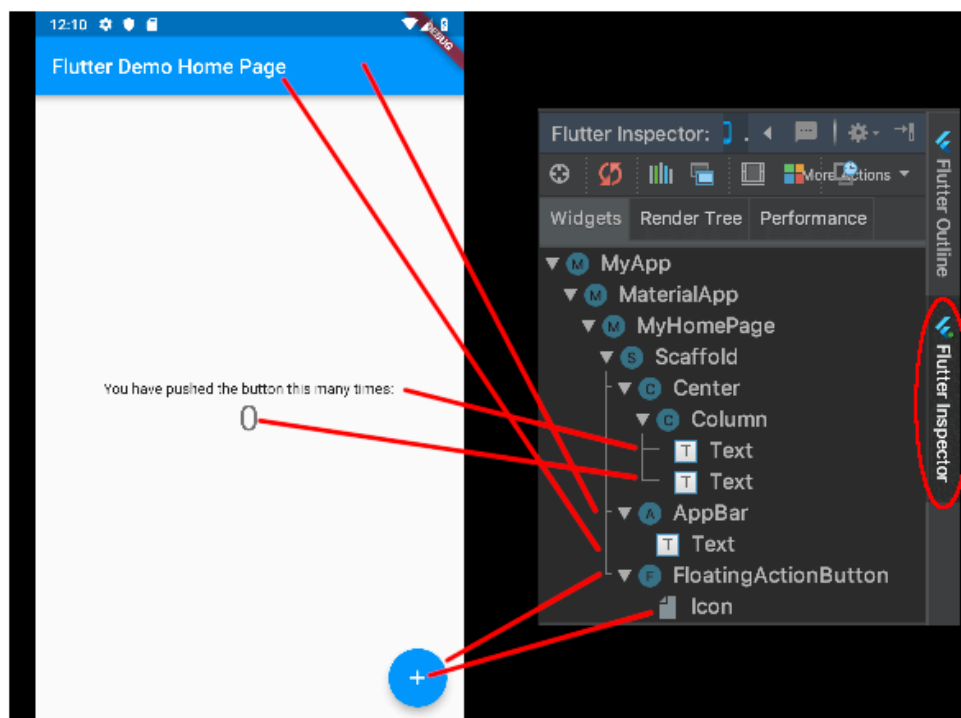


Figura 63: diagrama da Árvore Widgets. Fonte: Macoratti.



O usuário pode, por exemplo, usar um app de rede social. Para fotografar algo, é executado um aplicativo de câmera. Este aplicativo, poderá executar outro aplicativo para gerenciamento de arquivos. Por fim, o usuário retorna ao aplicativo de rede social e publica a foto. Ao finalizar usuário finalizar esse processo, o usuário voltará para o mesmo ponto em que parou no aplicativo de rede social. Além disso, no meio de todo esse processo o usuário pode ainda receber ligações, abrir outros aplicativos e assim por diante

1.18 Linha (Row)

Widget que alinha componentes horizontalmente:

```
Container(
  color: Color(0xFFFF44336),
  child: Row(
    mainAxisAlignment: MainAxisAlignment.spaceBetween,
    children: <Widget>[
      Container(
        color: Color(0xFFFFFC107),
        child: Text('Lorem ipsum')
      ),
      Container(
        color: Color(0xFF3F51B5),
        child: Text('Lorem ipsum'),
      )
    ],
  ),
)
```

Figura 64: Widget Linha (Row). Fonte: Simoes.

1.19 Coluna (Column)

Widget que alinha componentes verticalmente:

```
Container(
  color: Color(0xFFFF44336),
  child: Row(
    mainAxisAlignment: MainAxisAlignment.spaceBetween,
    children: <Widget>[
      Container(
        color: Color(0xFFFFFC107),
        child: Text('Lorem ipsum')
      ),
      Container(
        color: Color(0xFF3F51B5),
        child: Text('Lorem ipsum'),
      )
    ],
  ),
)
```

Figura 65: Widget Coluna (Column). Fonte: Simoes.

1.20 Texto (Text)

Exibe um texto de estilo único:



Figura 66: Widget Text (Text). Fonte: Simoes.

1.21 Botão (Button)

Exibe um botão para a aplicação.

```
/FlatButton(
  color: Colors.blue,
  textColor: Colors.white,
  disabledColor: Colors.grey,
  disabledTextColor: Colors.black,
  padding: EdgeInsets.all(8.0),
  splashColor: Colors.blueAccent,
  onPressed: () {
    /*...*/
  },
  child: Text(
    "FlatButton",
    style: TextStyle(fontSize: 20.0),
  ),
)FlatButton(
  color: Colors.blue,
  textColor: Colors.white,
  disabledColor: Colors.grey,
  disabledTextColor: Colors.black,
  padding: EdgeInsets.all(8.0),
  splashColor: Colors.blueAccent,
  onPressed: () {
    /*...*/
  },
  child: Text(
    "FlatButton",
    style: TextStyle(fontSize: 20.0),
  ),
)
```

Figura 67: Widget Botão (Button). Fonte: autoria própria.

1.22 Imagem (Image)

Exibe imagens, sendo que os formatos suportados são JPEG, PNG, GIF, GIF animado, WebP, WebP Animado, BMP e WBMP, e, podemos exibir imagens locais e imagens remotas, obtidas a partir de uma url.

```
import 'package:flutter/material.dart';
void main() => runApp(MyApp());
class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(title: Text('Exibindo imagem local')),
        body: Column(
          children: <Widget>[
            Image.asset(
              'imagens/logomac.png',
            ),
            Text('Macoratti.net')
          ],
        ),
      ),
    );
  }
}
```

Figura 68: Widget Botão (Button). Fonte: autoria própria.

1.23 Exercício Prático

Vamos praticar, execute o código abaixo no Android Studio, atentando os passos necessários para criar um projeto, ativar o emulador. O resultado final do Aplicativo deverá ser:

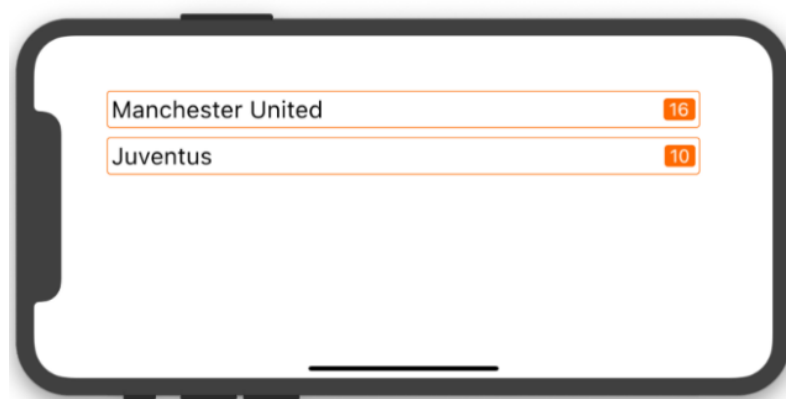


Figura 69: resultado exercício prático Widgets. Fonte: autoria própria.

Abaixo o código a ser inserido no Visual Studio:

```
import 'package:flutter/widgets.dart';
main()=>runApp(App());
class App extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Directionality(
      textDirection: TextDirection.ltr,
      child: Container(
        padding: EdgeInsets.symmetric(vertical: 60.0, horizontal: 80.0),
        color: Color(0xFFFFFFFF),
        child: Conteudo(),
      ),
    );
  }
}
```

```
class Conteudo extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Column(
      children: <Widget>[
        Contador('Manchester United'),
        Contador('Juventus'),
      ],
    );
  }
}
```

```
class Contador extends StatefulWidget {
  final String _nome;
  Contador(this._nome);
```

```

@override
State<Contador> createState()=>_ContadorEstado();
}

class _ContadorEstado extends State<Contador> {
  int conta = 0;

  @override
  Widget build(BuildContext context) {
    return Container(
      margin: EdgeInsets.only(bottom: 10.0),
      padding: EdgeInsets.all(4.0),
      decoration: BoxDecoration(
        border: Border.all(color: Color(0xFFFD6A02)),
        borderRadius: BorderRadius.circular(4.0),
      ),
      child: Row(
        mainAxisAlignment: MainAxisAlignment.spaceBetween,
        children: <Widget>[
          // [widget] é a propriedade que a classe esta armazenando
          // a instancia do [StatefulWidget] ([Contador] no nosso caso)
          _ContadorRotulo(widget._nome),
          _ContadorBotao(
            conta,
            onPressed: () {
              setState(){
                ++conta;
              };
            }
          )
        ],
      ),
    );
  }
}

```

```
}
```

```
class _ContadorRotulo extends StatelessWidget {
  static const textStyle = TextStyle(
    color: Color(0xFF000000),
    fontSize: 26.0,
  );
```

```
  final String _rotulo;
  _ContadorRotulo(this._rotulo);
```

```
  @override
  Widget build(BuildContext context) {
    return Text(
      _rotulo,
      style: _ContadorRotulo.textStyle,
    );
  }
}
```

```
class _ContadorBotao extends StatelessWidget {
  final conta;
  final onPressed;
  _ContadorBotao(this.conta, {@required this.onPressed});
```

```
  @override
  Widget build(BuildContext context) {
    return GestureDetector(
      onTap: onPressed,
      child: Container(
        padding: EdgeInsets.symmetric(horizontal: 6.0),
        decoration: BoxDecoration(
          color: Color(0xFFFFD6A02),
          borderRadius: BorderRadius.circular(4.0)
```

```
),  
child: Center(  
  child: Text(  
    '$conta',  
    style: TextStyle(fontSize: 20.0)  
  ),  
),  
),  
),  
);  
}  
}
```

Publicação de Aplicativo nas Lojas: Google Play e Apple Store

Para publicar um aplicativo nas lojas oficiais, se faz necessário a criação de uma conta de desenvolvedor. Cada uma das Apps Stores (Google e Apple) tem sua própria precificação e ciclos de pagamento.

O Google Play Store (Android) tem o custo de \$25 (Vinte e cinco dólares), pagos uma única vez (PLAY CONSOLE, 2021). Já a Apple App Store (iOS) tem o custo de \$99 (Noventa e nove dólares) anuais, (APPLE DEVELOPER PROGRAM, 2021). Nota-se a diferença entre os preços das 02 lojas. A Apple segue a mesma filosofia que seus produtos, o preço mais alto faz com que a qualidade dos publicadores de apps seja maior (NEONEXUS, 2021).

1.24 Criando uma conta de desenvolvedor no Google Play

Para a criação de uma conta de desenvolvedor no Google Play é necessário acessar o site do Google Play Console e inserir as informações que são solicitadas, conforme figura abaixo:

Google Play Console

prof.eduardo.oliveira@doctum.edu.br

Criar uma nova conta de desenvolvedor

A Conta do Google selecionada será proprietária desta nova conta de desenvolvedor. Caso esteja tentando entrar em uma conta de desenvolvedor existente, solicite um convite do administrador.

Se você for de uma organização, não recomendamos usar uma conta pessoal para configurar contas de desenvolvedor. É possível configurar Contas do Google com qualquer endereço de e-mail existente. [Saiba mais](#)

Para criar uma conta, é necessário pagar a taxa única de registro de US\$ 25. Para concluir o registro da conta, talvez seja preciso verificar sua identidade com um documento de identificação válido. Se não conseguirmos fazer isso, a taxa de registro não será reembolsada.

Nome público do desenvolvedor *

Endereço de e-mail secundário para contato *

Número de telefone para contato *

Contato para desenvolvedores e Termos de Serviço *

☐ Confirmando que li e concordo com o [Contrato de distribuição para desenvolvedores do Google Play](#). Aceito associar minha Conta do Google a esse contrato e confirmo que sou maior de 18 anos de idade.

☐ Confirmando que li e concordo com os [Termos de Serviço do Google Play Console](#). Aceito associar minha Conta do Google a esses Termos de Serviço.

Figura 70: Google Play Console. Fonte: Descomplica.

Será permitido escolher entre publicar o aplicativo desenvolvido com a sua conta pessoal do gmail ou uma mais profissional. Em seguida, preencha as informações, clique em “Criar conta e fazer o pagamento”. Após, concluir as etapas e realizar o pagamento. A publicação dos aplicativos será liberada na loja do Google Play Store.

1.25 Criando uma conta de desenvolvedor na App Store

Para a criação de uma conta de desenvolvedor na App Store é necessário acessar o site ID Apple e em seguida, clicar em “Crie seu ID Apple” e preencher todos os dados, conforme abaixo:

Figura 71: App Store. Fonte: Descomplica.

O Apple ID deve ser um e-mail válido, pois será confirmado posteriormente. Depois do Apple ID aprovado, será necessário acessar a página do Apple Developer e clicar no menu Account. Utilize o e-mail e senha do Apple

ID, aceite o contrato de privacidade da Apple para acessar o ambiente Getting Started e pronto, você já é um desenvolvedor!

Se você deseja publicar na Apple App Store é preciso ser membro do Apple Developer Program, e, para isto, é necessário acessar o site do Apple Developer Program e clicar no botão Enroll para criação da conta como desenvolvedor pessoa física ou como organização (pessoa jurídica). Para a criação de uma conta de organização, será necessário a criação de um D-U-N-S Number da empresa que representa um CNPJ internacional para identificação da organização junto a Apple. Após completar todos os passos acima, o usuário será redirecionado para o pagamento da taxa.

Desafio

Com os conceitos apresentados neste curso, faça um Aplicativo que em Flutter simples que mostra frases de efeito, tipo, frases do dia. Utilize os Widgets básicos. Você poderá desenvolver o layout livremente, e caso queira inovar algo, fique à vontade, esse critério também será avaliado.

Ao final da execução do desafio, insira evidências sobre a atividade (estrutura) em arquivos: PDF, JPG e/ou PNG, podendo inserir na plataforma quantos arquivos quiser ou pode-se zipar os arquivos- ZIP e RAR, inserindo todos os arquivos de uma vez só. As evidências são:

- Código-fonte;
- Imagens da execução do aplicativo com todas as etapas e telas, até o resultado final do programa.

Conclusão

A abordagem desta apostila permite ao leitor uma interação ativa com as principais práticas de Desenvolvimento Mobile com Flutter Básico. Durante o curso aprendemos sobre: o histórico da plataforma Flutter, DART; aprendemos os principais componentes do Flutter - Widgets, IDE oficial para desenvolvimento em Flutter; aprendemos a desenvolver interfaces básicas seguindo os padrões oficiais da plataforma; e por último dicas de como publicar os aplicativos em lojas da Google e Apple.

Obrigado por fazer parte do curso e ter realizado a leitura desta apostila. Espero que o interesse por aplicações móveis apresentado neste curso desperte o interesse nos demais cursos existentes em nossa plataforma.

Referências

Documentação do Flutter - disponível em <https://flutter.dev/> - acessado em novembro/2021.

Download do Flutter – disponível em: <https://docs.flutter.dev/get-started/install/windows> acessado em novembro/2021.

Programação DART em: <https://medium.com/flutter-comunidade-br/introdu%C3%A7%C3%A3o-a-linguagem-de-programa%C3%A7%C3%A3o-dart-b098e4e2a41e> acessado em novembro/2021.

Instalação do Android Studio em: <https://developer.android.com/studio?hl=pt-br> acessado em novembro/2021.

Configuração do Emulador AVD Manager em: <https://www.androidpro.com.br/blog/desenvolvimento-android/emulador-de-android/> acessado em novembro/2021.

Instalação do Visual Code em <https://code.visualstudio.com/download> acessado em novembro/2021.

Variáveis e condicionais, linguagem DART em <https://medium.com/flutter-comunidade-br/introdu%C3%A7%C3%A3o-a-linguagem-de-programa%C3%A7%C3%A3o-dart-b098e4e2a41e> acessado em novembro/2021.

Widgets - Stateless e Stateful em <https://medium.com/flutter-comunidade-br/explorando-widgets-e-layouts-6fd34ecb82f> acessado em novembro/2021.

Diagrama da árvore widgets em http://www.macoratti.net/19/07/flut_layout1.htm acessado em novembro/2021.

Exemplo de widgets: linhas, colunas e texto em <https://www.simo.es.dev/flutter/widgets-do-flutter-basico> acessado em novembro/2021.

Apps nativos para Android e IOS com Flutter, crie Apps como: Youtube, WhatsApp, Uber, OLX e muito mais! em: <https://fit->

tecnologia.udemy.com/course/desenvolvimento-android-e-ios-com-flutter/learn/lecture/13537448#overview – acessado em outubro/2021

The Complete 2021 Flutter Development Bootcamp with Dart em: <https://fit-tecnologia.udemy.com/course/flutter-bootcamp-with-dart/learn/lecture/14481906?start=15#overview> - acessado em outubro/2021

Explorando Widgets e Layouts em Flutter em <https://medium.com/flutter-comunidade-br/explorando-widgets-e-layouts-6fd34ecb82f> acessado em novembro/2021 - acessado em novembro/2021.

Flutter para iniciantes em <https://www.flutterparainiciantes.com.br/basico/widgets> - acessado em novembro/2021.

Desenvolvimento em Plataformas Híbridas em <https://dex.descomplica.com.br/projetos-de-aplicativos-moveis-multiplataforma/desenvolvimento-em-plataformas-hibridas-flutter-pos/desenvolvimento-em-plataformas-hibridas-apresentar-como-e-feita-a-publicacao-de-um-aplicativo-na-app-store-e-google-play/explicacao/1> - acessado em novembro/2021.

CONTROLE DE REVISÃO DO DOCUMENTO / *DOCUMENT REVISION CONTROL*

Revisão	Descrição	Razão	Autor	Data
<i>Review</i>	<i>Description</i>	<i>Reason</i>	<i>Author</i>	<i>Date</i>
A	-	Revisão inicial	Larissa Alves	30/11/21
B	Alteração de logotipia do rodapé do documento	Revisão do modelo do formulário FIT-F.24.1.01-04 Rev. B	Bruno Rafael Rodrigues	01/07/22



FUTURO DO TRABALHO, TRABALHO DO FUTURO

Bom curso



MINISTÉRIO DA
CIÊNCIA, TECNOLOGIA
E INOVAÇÕES

