Ensino Médio integrado ao curso Técnico em informática

Ana Gabriela Enes Rocha

Bruno Farias de Almeida

Caroline Batista Pereira

Fabricia Maria Oliveira de Almeida

Marcus Vinicius da Silva Souza

Rosangela dos Santos Silva

Yuri Santana Costa

Java/Android

CUBATÃO
2021

Ana Gabriela Enes Rocha (448), Bruno Farias de Almeida (417), Caroline Batista Pereira (418), Fabricia Maria Oliveira de Almeida (417), Marcus Vinicius da Silva Souza (418), Rosangela dos Santos Silva (417), Yuri Santana Costa (418)

Java/Android

Trabalho de conclusão de curso solicitado pelo docente à disciplina de Projeto de Sistemas lecionada no Instituto Federal de Educação, Ciências e Tecnologia de São Paulo – Campus Cubatão.

Orientador(a): Prof. Mauricio Neves Asenjo

CUBATÃO
2021

SUMÁRIO

INTRODUÇÃO

Os aplicativos fazem cada vez mais parte da rotina das pessoas. Eles podem ser usados para diversas finalidades, como chamar um táxi, pedir uma refeição, baixar músicas e assistir a vídeos. Sem dúvida, são uma ótima opção para tornar a vida mais prática.

Devido a esta importância dos aplicativos na atualidade, o professor orientador sugeriu o desenvolvimento de 4 aplicativos que serão apresentados neste documento através de um passo a passo de desenvolvimento, utilizando a linguagem Java. Neste trabalho iremos falar um pouco sobre o histórico do Java para aplicação Android e a suas características, além de trazermos a evolução do Android.

Histórico e principais características do Java

O Sistema Android foi desenvolvido por Andy Rubin, Rich Miner, Nick Sears e Chris White, empresários já iniciados no ramo da tecnologia. E veio a surgir em 2003, na cidade de Palo Alto na Califórnia.

Os criadores resolveram focar no mercado mobile, pois o mercado não era tão amplo quanto gostariam para lançar um sistema inovador para câmeras digitais, o que era a ideia original.

O Open Source, baseado no Kernel Linux, um novo meio de sistema operacional móvel foi oferecido pela equipe de desenvolvedores. A ideia era ser simples aos criadores e ser acessível para todas as pessoas que quisessem ter acesso ao sistema. Este constava com uma interface simples, funcional e também integrada a vários instrumentos.

Em 2007, a criação do Android deu um passo grande quando fabricantes como Samsung, Sony, HTC, Google e fabricantes de Hardware como Qualcomm reuniram-se em um consórcio de tecnologia e fundaram a Open Handset Alliance. O objetivo era a criação de uma plataforma de código aberto para smartphones, e o resultado foi o primeiro Android comercial do mercado, rodando em um HTC Dream. Com a chegada do Android, o próprio conceito de smartphone foi remodelado e hoje é o sistema mais utilizado no mundo.

É importante mencionar que o Android sempre foi criticado por causa das brechas e dos malwares. O problema é que os alvos são quase sempre aparelhos com versões antigas do Android, que têm menos atualizações de segurança. E a facilidade de submeter apps na Google Play é muito positiva para desenvolvedores, mas também facilita muito a entrada de vírus.

O Android não começou bem como companhia e a empresa não conseguia levantar dinheiro de investidores, além das operadoras que não queriam saber de um serviço que poderia tirar o controle da indústria das mãos delas. E com isso, uma marca interessada na aquisição foi a Google. Ele com uma incrível estratégia, ofereceu 10 milhões de dólares aos desenvolvedores que realizassem os melhores apps para Android, partindo da primeira versão pública do Android SDK.

A 1.5 foi a primeira versão do Android com nome de doce: a Cupcake, que trouxe suporte a outros teclados virtuais, gravação de vídeo e autorrotação de tela. O robozinho verde símbolo da marca é um design original. Sendo a grande vantagem do Android para manter essa marca, a democratização do smartphone. Cada nova versão do Android é um doce ou sobremesa diferente e em ordem alfabética, aí sempre fica aquela expectativa para saber qual é o próximo.

Interessante mencionar que o grande porta-voz do Android no mundo foi um brasileiro, Hugo Barra, de 2008 a 2013. Sundar Pichai entrou em seu lugar, e mais tarde virou o CEO de toda a Google.

Por que a utilização da linguagem Java?

O Java Android surgiu, provavelmente, para tornar o desenvolvimento mobile mais confortável. Ela é uma linguagem que está em toda a parte: celular, veículos, em pequenos robôs, enfim, está rodando em quase todo lugar.

É mais fácil aprender durante o desenvolvimento orientado a objetos e também ajuda a manter um sistema com flexibilidade e ainda extensível. Além de que, quase tudo nele está num celular, hardware e equipamentos eletrônicos.

O Android Studio é baseado no IntelliJ IDEA, umas das ferramentas mais poderosas de desenvolvimento Java, oferecendo muitos recursos que melhoram a produtividade na criação de aplicativos para Android. Aliás, facilita muito a aprendizagem, encaminhando xml com configurações e separado por pastas, visando o desempenho do usuário que irá mexer no software.

Algumas vantagens relevantes:

- Além de ser uma linguagem, é uma plataforma de desenvolvimento;
- É possível obter material para estudo de maneira fácil, pois os grupos de usuário Java são muito fortes em todo o mundo;
- Grande número de frameworks;
- A Máquina virtual Java atualmente roda cerca de 350 linguagens;
- É possível desenvolver em qualquer sistema operacional para qualquer sistema operacional.

Evolução do Sistema Android

Versão	Data de Lançamento	Características Gerais:
1.0	23/09/2007	Primeira versão a ser comercializada. Foi baseada na versão 2.6.25 do Kernel Linux. Possuía as seguintes atribuições: • Atualiza aplicativos através do aplicativo Market; • Sincronização do Gmail com o app Gmail; • Pastas que permitem o agrupamento de vários ícones de Apps em um único ícone de pasta na tela inicial.
1.5- Cupcake	30/04/2009	Foi lançada, com base no Karnel Linux 2.6.27 Primeira versão a usar um nome baseado em um doce, algo que seria feito para as versões seguintes. Possuía as seguintes atribuições: • Gravação e reprodução de vídeo em MPEG-4 e 3GP; • Recursos de colar e copiar adicionado ao navegador Web; • Suporte para teclados virtuais de terceiro com previsão de texto e dicionário para palavras personalizadas do usuário.
1.6 - Donut	15/09/2009	Foi baseada no Karnel Linux 2.6.29 e possuía alterações, como: • Suporte para resoluções de tela (320×240 e 800×480)

		• Caixa de pesquisa rápida na parte superior da tela inicial, que irá arrastar seu histórico do navegador e contatos.
2.0- Eclair	26/10/2009	Foi baseado no Karnel Linux 2.6.29 e possuía alterações, como: • Teclado recebe dicionário mais inteligente; • Pesquisa por palavras-chaves entre os SMS guardados; • Calendário com opções para marcar os convidados de um evento.
2.0.1- Eclair	3/12/2009	Foi baseado no Karnel Linux 2.6.29 e possuía alterações, como: • Câmera com suporte flash e zoom digital; • Teclado virtual melhora velocidade de resposta; • Acesso às informações de um contato específico, ou seja, basta pressionar a foto do contato desejado e escolher entre as opções disponíveis.
2.1- Eclair	11/01/2010	Foi baseado no Karnel Linux 2.6.29 e possuía as seguintes atribuições: • Cinco telas iniciais; • Live Wallpaper; • Nova galeria de imagens e vídeos; • Introdução da digitação de texto por voz.
2.2- Froyo	20/05/2010	É até cinco vezes mais rápida do que a versão anterior (Eclair) e possuía alterações, como: • Navegador web nativo três vezes mais rápido;

		• Câmera ganha menu que possibilita acesso mais fácil às principais funções; • Possibilidade de armazenar aplicativos no cartão SD.
2.3-Gingerbread	6/12/2010	Foi baseado no kernel Linux 2.6.35 e possuía alterações, como: • Suporte a aparelhos com câmeras frontais; • Suporte a sensores de movimentos; • Teclado virtual melhorado; • Chamadas pela internet.
3.0- Honey Comb	22/02/2011	Foi baseado no Kernel Linux 2.6.36. O tablet foi lançado em 24 de fevereiro em 2011, por esta razão foi a primeira atualização de Android que suportava tablets.
4.0 Ice cream Sandwich	19/10/11	Baseado no Kernel Linux 3.0.1, foi a versão que adicionou o ScreenShot.
4.1 Jelly Bean	9/7/2012	Foi baseado no Kernel Linux 3.0.31. O principal objetivo do Jelly Bean, em relação as versões anteriores, era melhorar a funcionalidade e desempenho da interface do usuário. A melhora envolveu o projeto “Project Butter”, que possuía os seguintes recursos: • Antecipação do toque; • Buffer triplo; • Taxa de quadros fixos de 60 fps, com intenção de criar uma interface fluída e “suave como uma manteiga".
4.4 Kit Kat	15/01/2013	O sistema Android kit kat foi feito em parceria com a marca Nestle, onde levou o nome de um dos doces mais famosos da marca. Possuía a seguintes alterações: • Mudanças visuais;

		• Melhor aproveitamento de memória; • Google Now e serviços do Google; • Aplicativos em tela inteira.
5.0 Lollipop	3/11/2014	Possuía novas novidades, em relação ao Design, como: • Sistema muito limpo e fluído; • Tela de Bloqueio; • Exibição das notificações; • Modo de economia de bateria.
5.1 Lollipop	7/07/2015	Um ano após o primeiro lançamento o Android traz outras diversidades, como: • Vários chips de operadora suportados; • Maior proteção contra roubo e perda; • Wifi e Bluetooth ajustável pela barra de notificações.
6.MarshMallow	5/10/2015	Foi lançado primeiro com o codinome “Android M " e possuía as seguintes atualizações: • USB type-c; • Adaptação de memória externa (SD cards) como parte da memória interna; • Suporte para leitores de impressão digital; • Backup e restauração automática no drive para dados e aplicativos.
8.0 Oreo	21/08/2017	O Android Oreo possuí as seguintes atribuições: • O Google assistente estava como assistente virtual; • Acompanhamento no consumo de baterias por parte dos aplicativos e pelos serviços do Android; • O valor da porcentagem da bateria em negrito;

		• Espaço existente entre a tela de atalhos e as notificações foi retirado deixando tudo em apenas uma página.
9.0 Pie	6/08/2018	Esta versão possuía os seguintes atributos: • **Brilho Adaptativo:** entende a forma como cada pessoa gosta de definir o controle de brilho, considerando a iluminação do ambiente ao redor; • **Configurações redesenhadas como:** capturas de telas melhores; controles com volumes simplificados; maneira mais fácil de gerenciar notificações e um botão de confirmação de rotação; • **Seleção inteligente de texto:** Reconhece o significado do texto, que está sendo digitado e sugere ações relevantes; • **Novo Sistema de navegação:** Com apenas um botão de início consegue-se fazer inúmeras ações, ou seja, o novo sistema traz menos botões e mais gestos.
10. Android	3/09/2019	Esta versão foi uma das únicas que não possui o nome de algum doce. O nome é “Android 10”. Possui os seguintes atributos: • Suporte a conexão de internet móvel 5G; • **Screen-Capture:** função de gravar os movimentos na tela; • Melhor suporte a dispositivos dobráveis; • **Sensor Privacy:** função em desabilitar todos os sensores.

11. Android	2/07/2020	Esta versão, assim como a anterior, foi uma das únicas que não possuía o nome de algum doce. O nome é “Android 11”. Possui os seguintes atributos: • **Mensagens Prioritárias:** Agora é possível marcar como: “mensagem prioritária” em contatos considerados importantes para o usuário; • **Histórico de Notificações:** Toda as notificações podem ser descartadas, inclusive as que estavam em andamento; • **Gravação de Tela nativa:** Um recurso do iOS que agora chegou para todos celulares com o Android 11.

Os 3 principais IDE’s para desenvolvimento Java/Android

Eclipse

Eclipse, IDE criado pela empresa IBM, para desenvolvimento em Java, suporta diversas linguagens apenas com a instalação de plugins (C/C++, PHP, Python, Kotlin, entre outras), por meio dessas instalações o desenvolvedor incrementa as funcionalidades do Eclipse.

A plataforma do Eclipse também oferece vários pacotes de desenvolvimento como, Eclipse JDT, a base para qualquer plug-in na linguagem Java, o Eclipse SDK, um pacote de distribuição da IDE Java, o Eclipse WTP (Web Tools Platform), usado para desenvolvimento de linguagem para web, e o compilador do JDT, que é seu próprio compilador Java, sendo mais rápido e de código aberto.

A plataforma SWT usada pelo Eclipse, permite a criação de aplicações gráficas, como aparência de aplicações nativas, sem sacrificar a compatibilidade, o SWT promove acesso direto a estes componentes ao programador, podendo configurá-los e posicioná-los da forma que desejar.

IntelliJ IDEA

Lançada em 2008 pela empresa JetBrains, a IntelliJ IDEA fez sucesso entre programadores desde então. Logo na versão inicial, a aplicação já oferecia recursos importantes, tais como: Suporte a SQL, Java, framework Seam e WebServices RESTful.

Atualmente, o foco permanece o mesmo, no entanto, foram desenvolvidas muitas outras ferramentas que o acompanharam as atualizações do programa. Entre as principais características IDE estão os plug-ins de qualidade, ótimo suporte técnico e sistema considerado intuitivo, sendo ideal para projetos iniciais.

Atualmente a ferramenta está disponível em dois principais segmentos, o pago, chamado de Ultimate, antes da compra poderá experimentar um Trial gratuito durante 30 dias, contando com todos os recursos para uso deliberadamente e, a versão inteiramente gratuita e denominada Community, restrita a limitação das funcionalidades disponíveis,

Netbeans

O NetBeans suporta inúmeras outras linguagens de programação importantes, no contexto tecnológico atual, a principal entre as funções desse IDE seria amenizar as falhas existentes no trabalho dos programadores, oferecendo ferramentas como: compilar, escrever, editar e remover bugs de código.

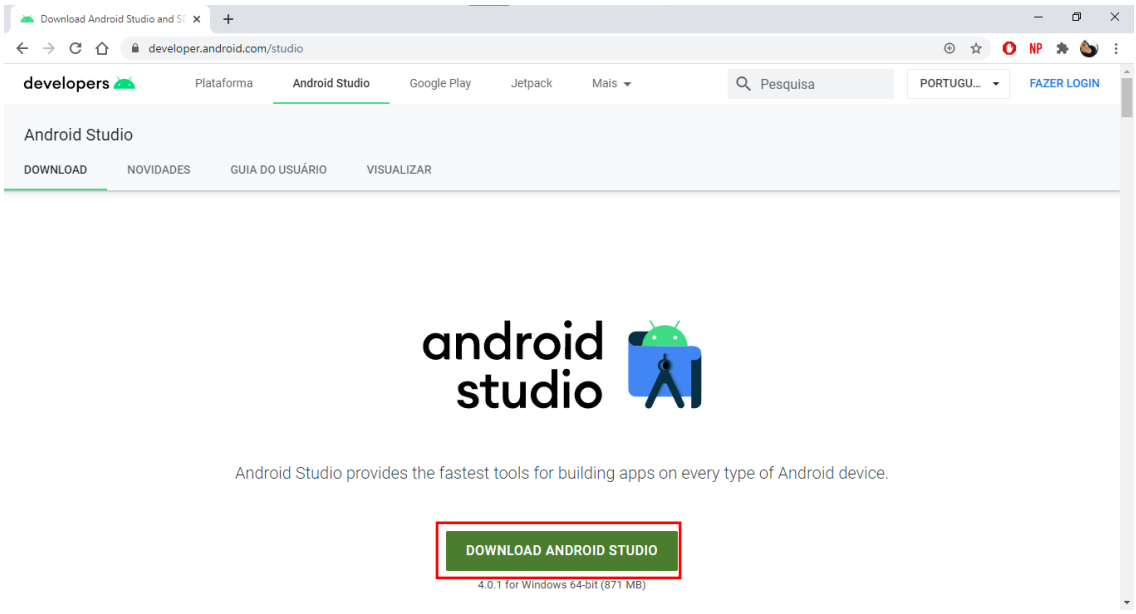
Desenvolvido pela Sun Microsystems, durante muito tempo, a empresa manteve a ferramenta sob seu domínio, então ao ser adquirida pela Oracle, foi liberado o código fonte da IDE, para torná-la uma plataforma OpenSource.

Os pontos positivos do Netbeans são recursos inovadores, a proposta gratuita e OpenSource da IDE, as críticas estão voltadas aos travamentos apresentados e ao sistema ultrapassado.

Instalação do Android Studio

O Android Studio pode ser baixado no link disponível na seguinte URL

https://developer.android.com/studio

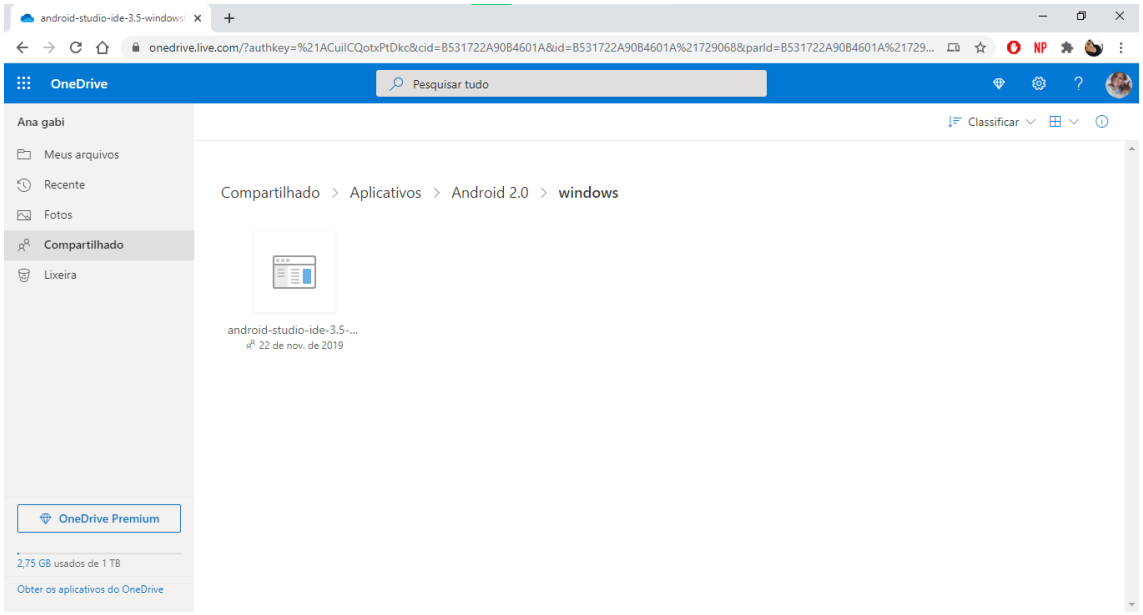


Baixando a versão mais recente disponível.

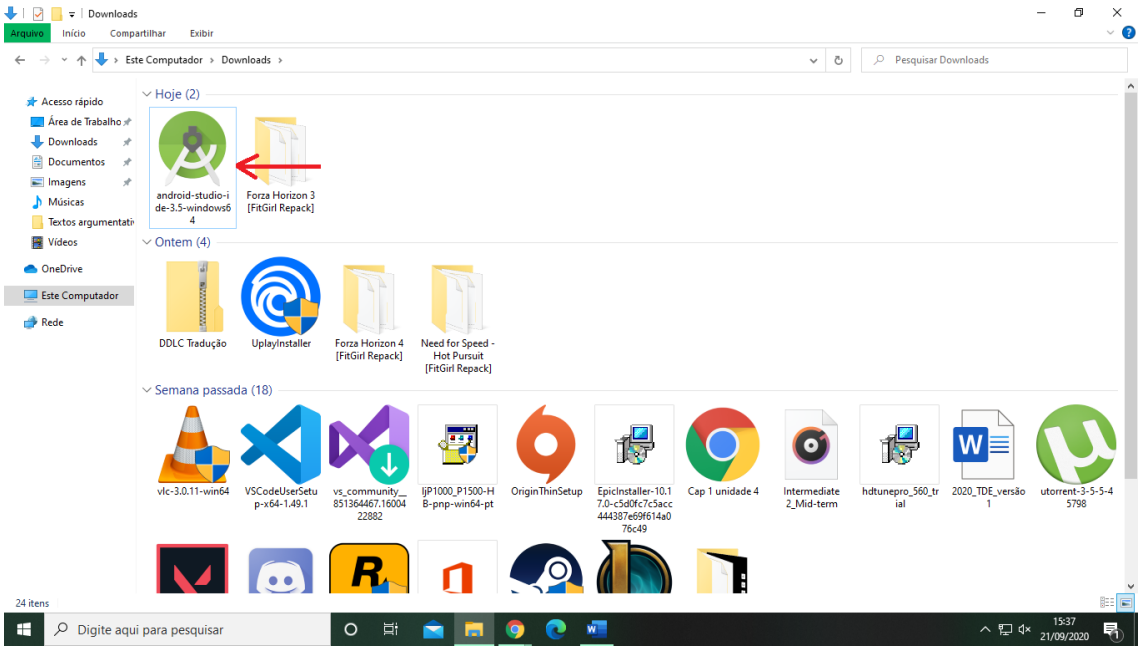
No meu caso, utilizarei a versão do Android Studio 3.5.

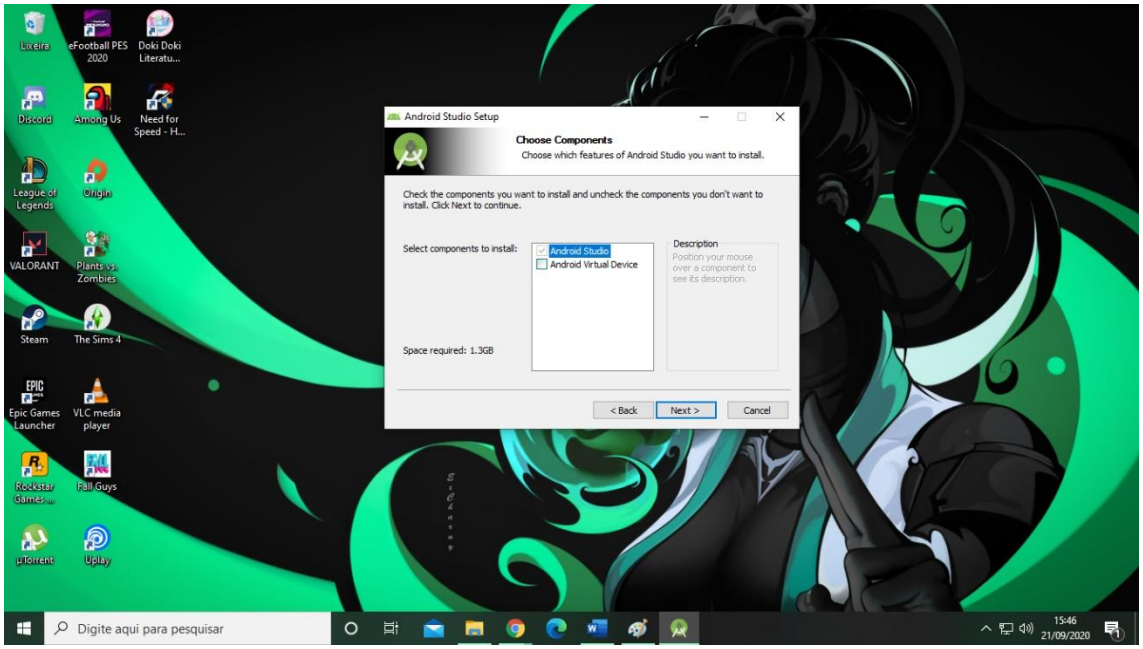
Link para acesso à versão:

https://onedrive.live.com/?authkey=%21ACuiICQotxPtDkc&cid=B531722A90B4601A&id=B531722A90B4601A%21729068&parId=B531722A90B4601A%21729064&action=locate

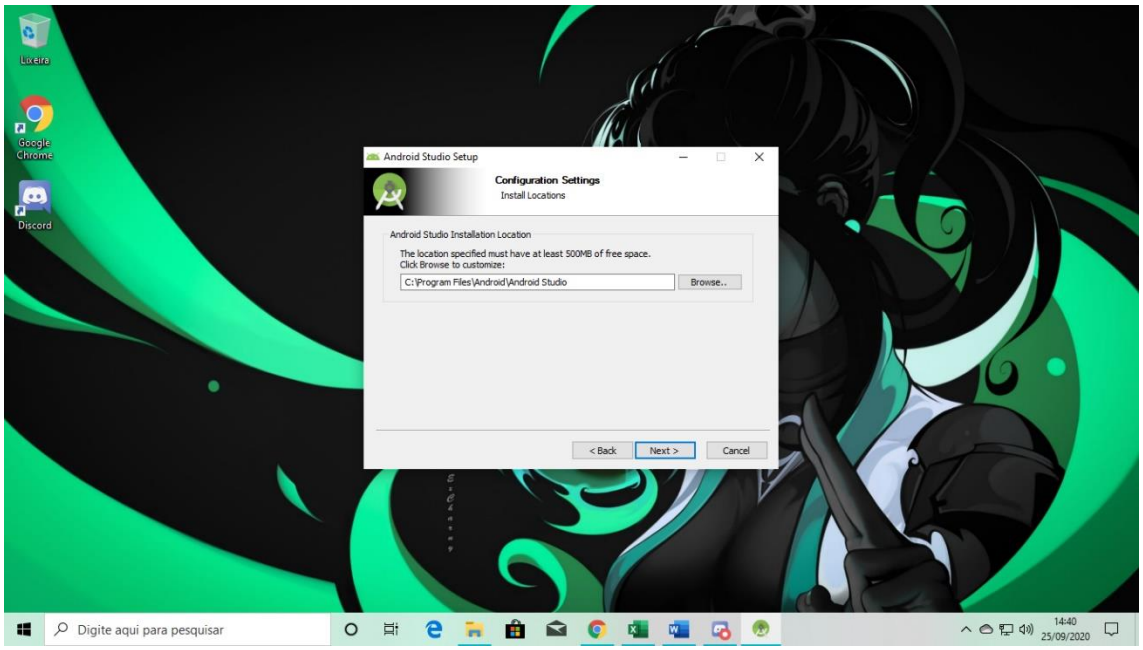


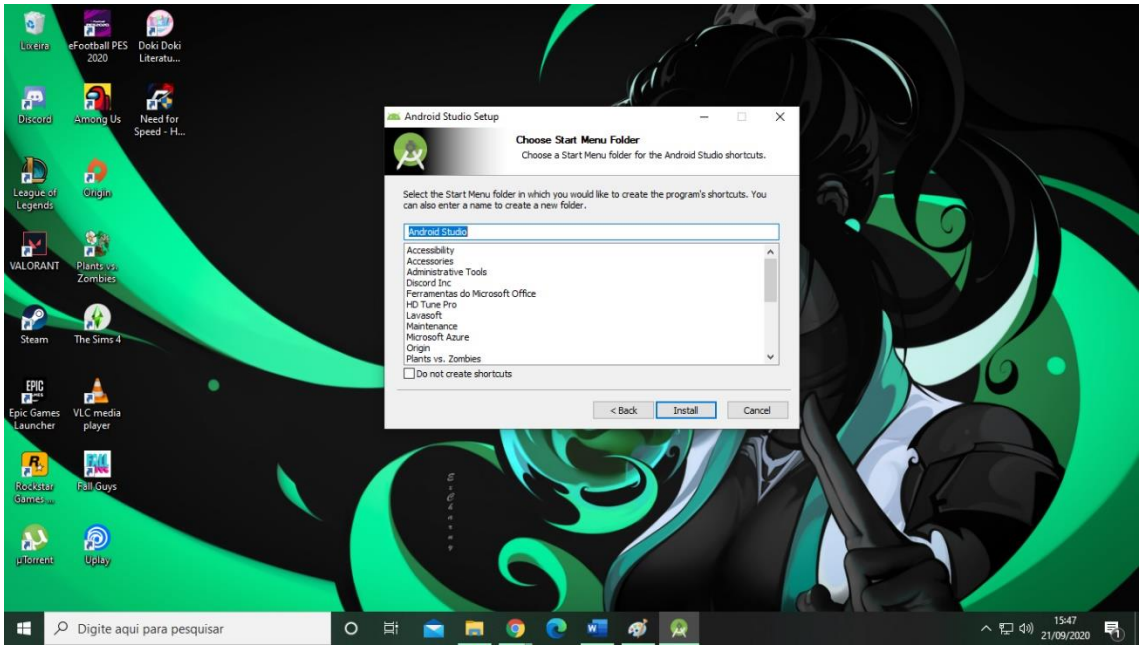
Clicando no arquivo, o programa baixará na sua pasta de download ou na pasta em que você optar. Abra a pasta por meio de um duplo clique no arquivo executável.

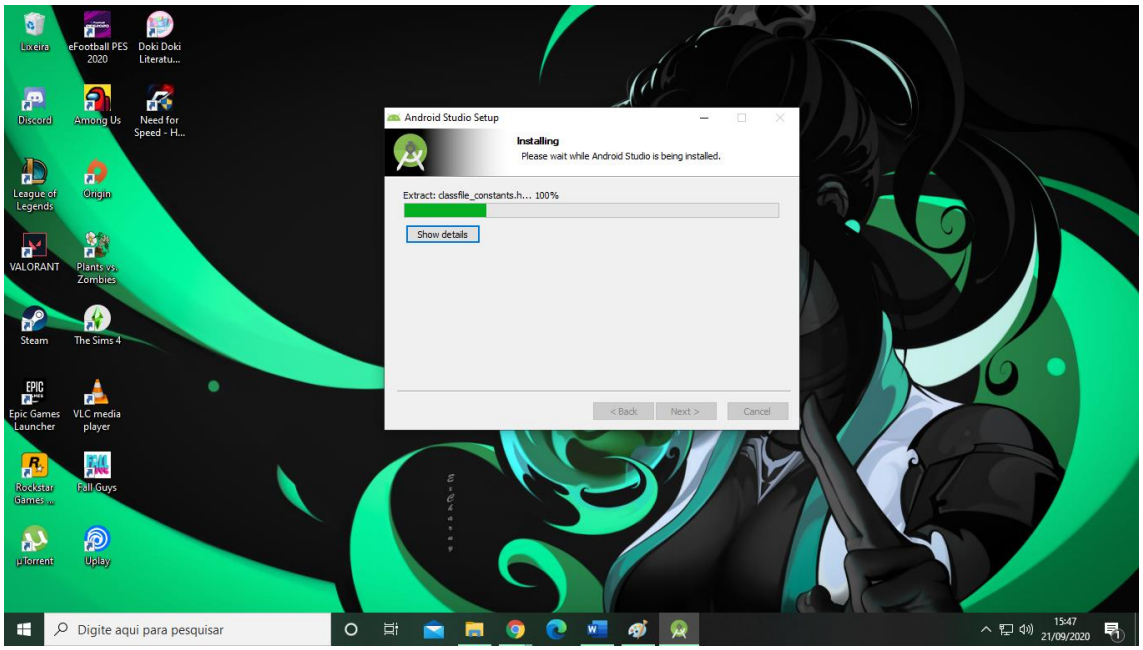


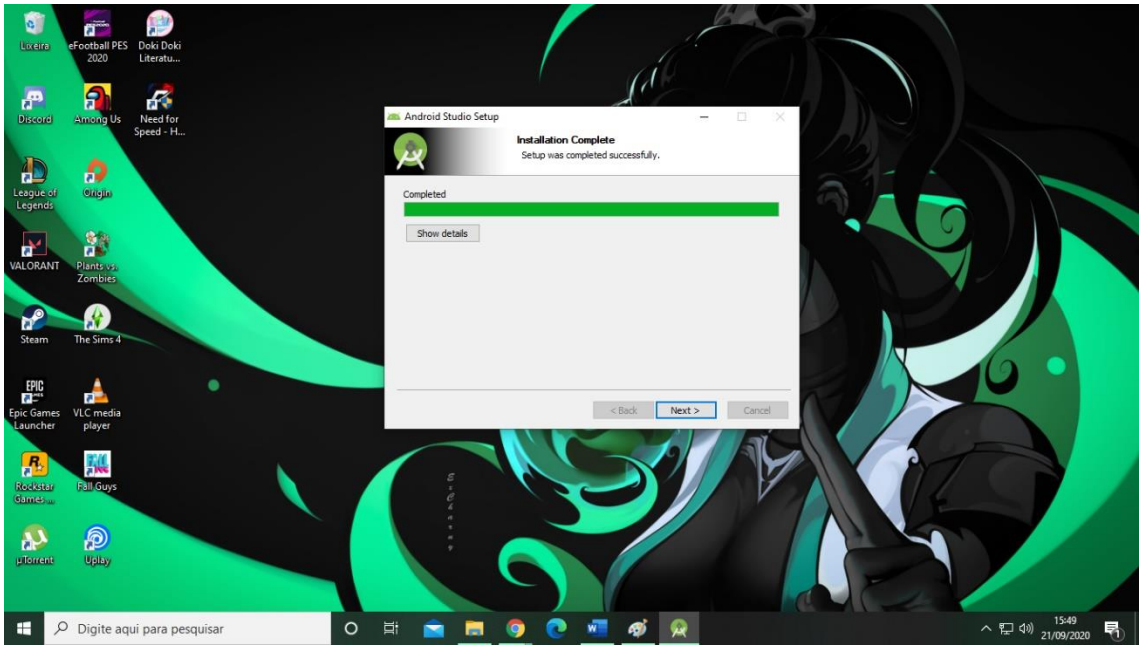


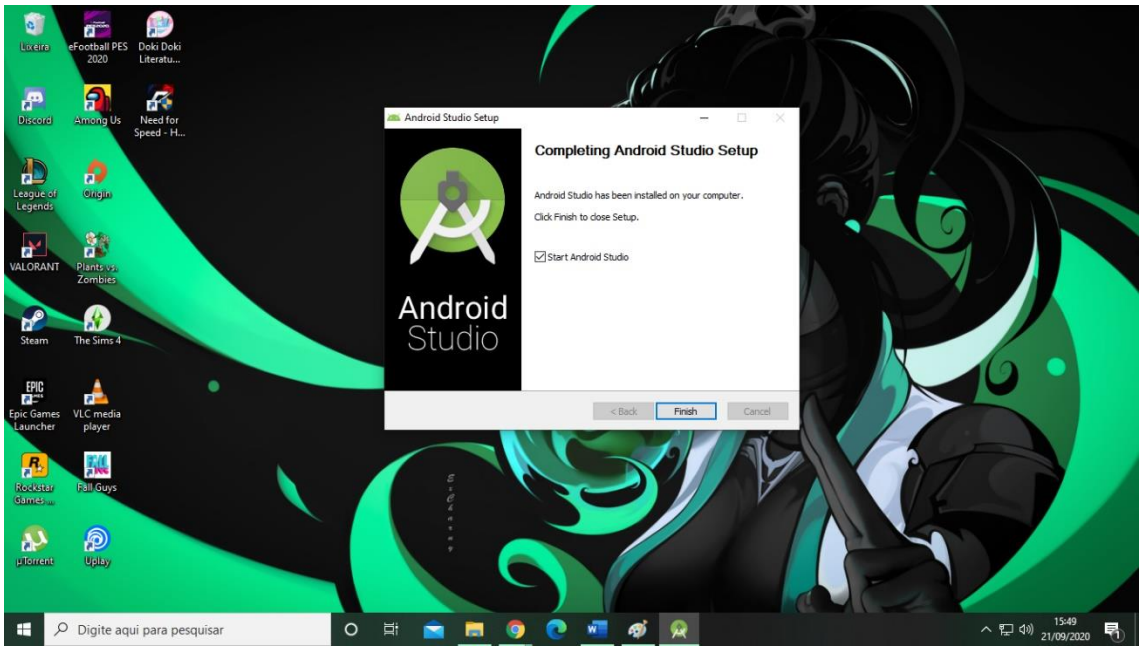
Na próxima opção, selecione a pasta/diretório que prefere instalar. É importante ressalvar que a pasta não pode conter espaços ou caracteres especiais, como: “$”, “%”, “-“, “!”, etc.


Lixeira
Google Chrome
Discord
Android Studio Setup
Configuration Settings
Install Locations
Android Studio Installation Location
The location specified must have at least 500MB of free space.
Click Browse to customize:
C:\Program Files\Android\Android Studio
Browse...
< Back
Next >
Cancel
Digite aqui para pesquisar
14:40
25/09/2020


Lixeira
eFootball PES 2020
Doki Doki Literatu...
Discord
Among Us
Need for Speed - H...
League of Legends
Origin
VALORANT
Plants vs. Zombies
Steam
The Sims 4
Epic Games Launcher
VLC media player
Rockstar Games ...
Fall Guys
µTorrent
Uplay
Android Studio Setup
Choose Start Menu Folder
Choose a Start Menu folder for the Android Studio shortcuts.
Select the Start Menu folder in which you would like to create the program's shortcuts. You can also enter a name to create a new folder.
Android Studio
Accessibility
Accessories
Administrative Tools
Discord Inc
Ferramentas do Microsoft Office
HD Tune Pro
Lavasoft
Maintenance
Microsoft Azure
Origin
Plants vs. Zombies
Do not create shortcuts
< Back
Install
Cancel
Digite aqui para pesquisar
15:47
21/09/2020

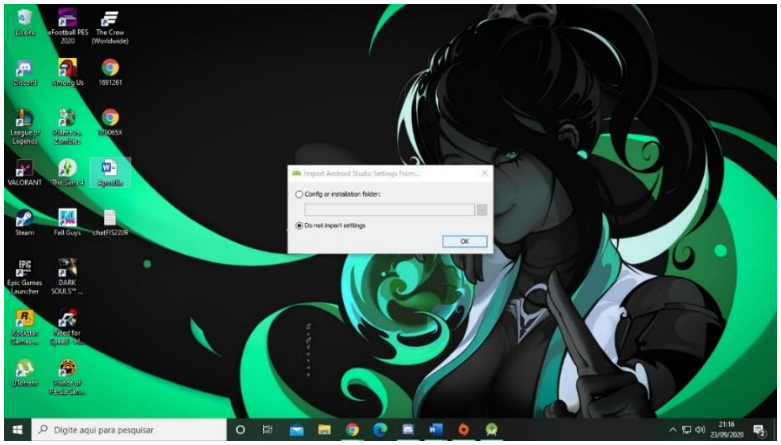

Lixeira
eFootball PES 2020
Doki Doki Literatu...
Discord
Among Us
Need for Speed - H...
League of Legends
Origin
VALORANT
Plants vs. Zombies
Steam
The Sims 4
Epic Games Launcher
VLC media player
Rockstar Games ...
Fall Guys
µTorrent
Uplay
Android Studio Setup
Installing
Please wait while Android Studio is being installed.
Extract: classfile_constants.h... 100%
Show details
< Back
Next >
Cancel
Digite aqui para pesquisar
15:47
21/09/2020


Lixeira
eFootball PES 2020
Doki Doki Literatu...
Discord
Among Us
Need for Speed - H...
League of Legends
Origin
VALORANT
Plants vs. Zombies
Steam
The Sims 4
Epic Games Launcher
VLC media player
Rockstar Games ...
Fall Guys
µTorrent
Uplay
Android Studio Setup
Installation Complete
Setup was completed successfully.
Completed
Show details
< Back
Next >
Cancel
Digite aqui para pesquisar
15:49
21/09/2020

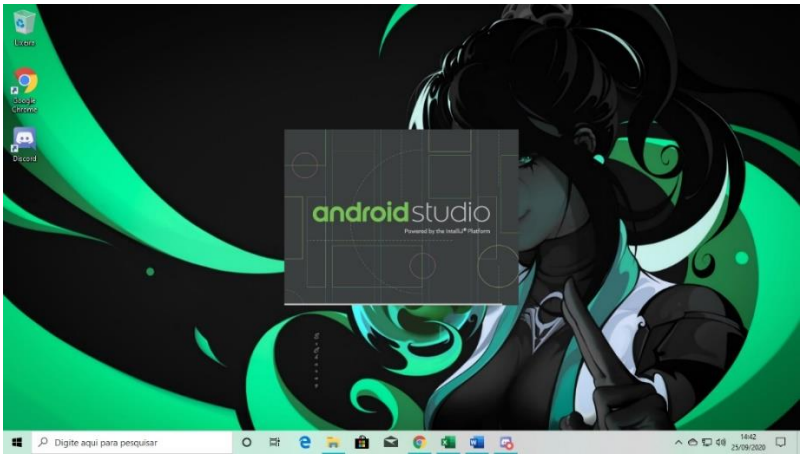


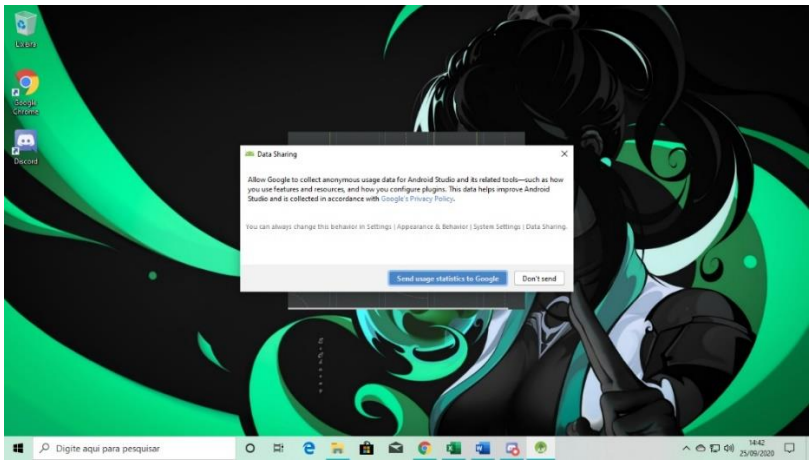
Se deixar a caixa de texto selecionada, o programa iniciará assim que finalizar, se não desejar abrir o AS, desmarque a seleção.

Configurando o Android Studio



Caso já tenha instalado o AS em outras versões e queira importar suas configurações, escolha a opção “Config or installation folder” e a selecione.



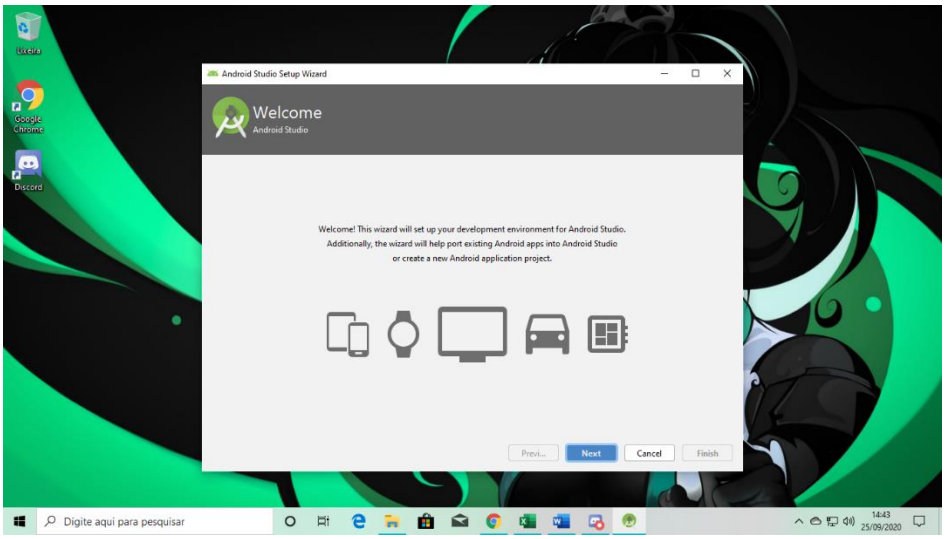


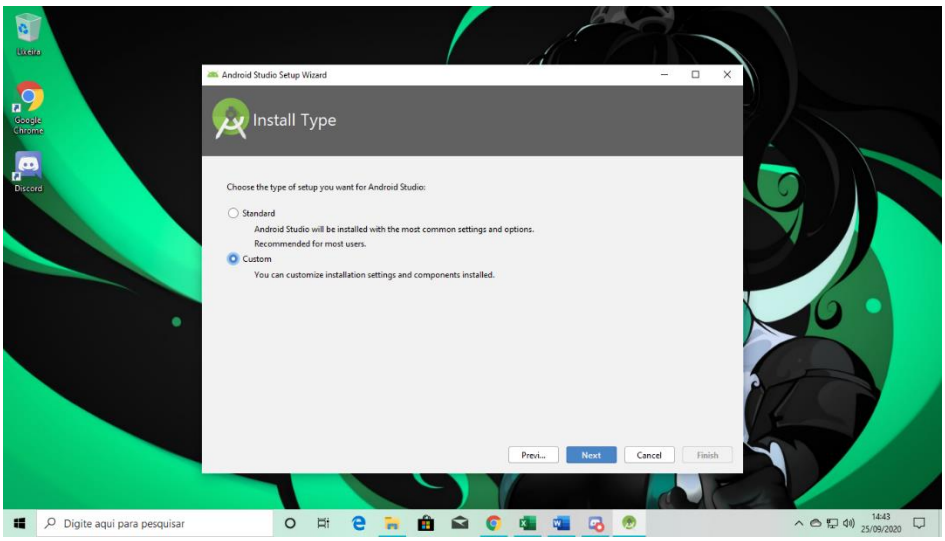
Está opção é para perguntar se você gostaria que o Android Studio envie estatística para o Google.

Configuração do SDK (Software Development Kit).

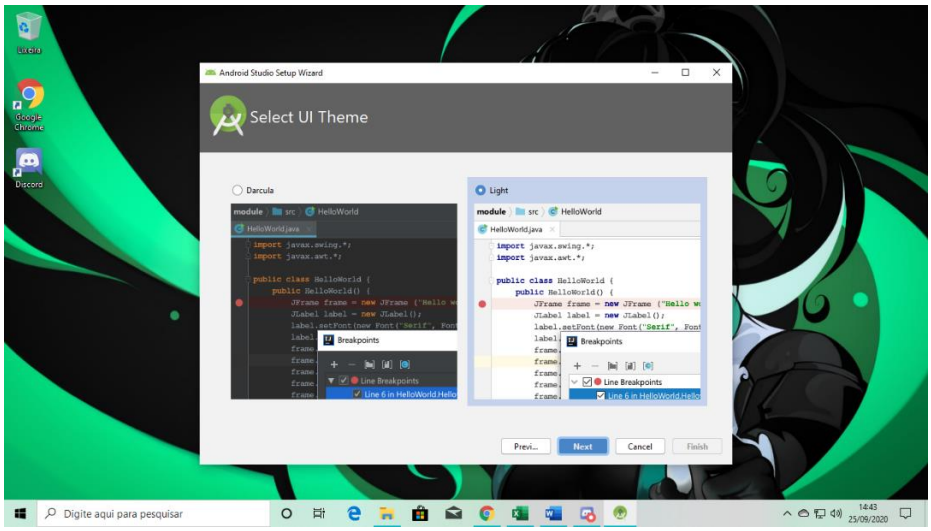
O que é SDK?

SDK é um kit de ferramentas de desenvolvimento, ou seja, um conjunto de recursos úteis para se desenvolver um software. Um SDK instalado na máquina do programador é comparável a um carpinteiro ter em sua caixa de ferramentas um martelo, pregos, lixa de madeira e um serrote.

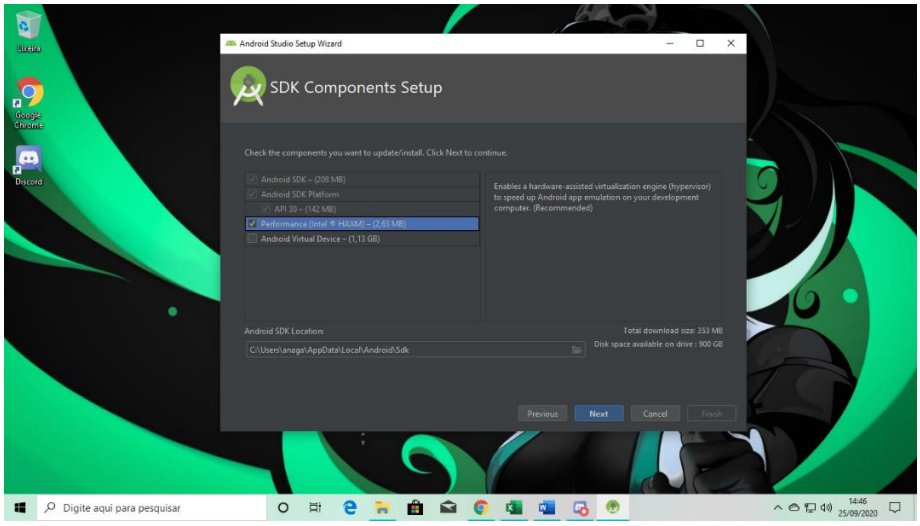




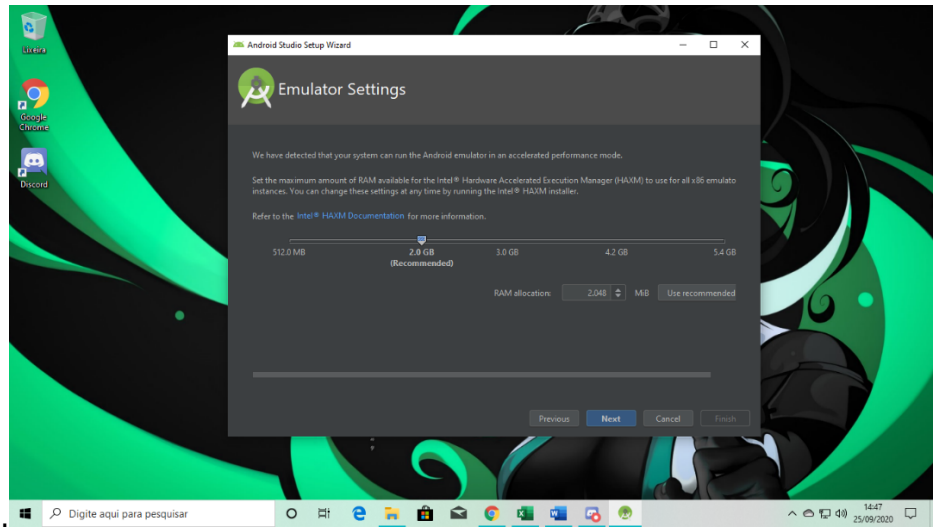
A opção “Standard” instala as configurações padrão do SDK;



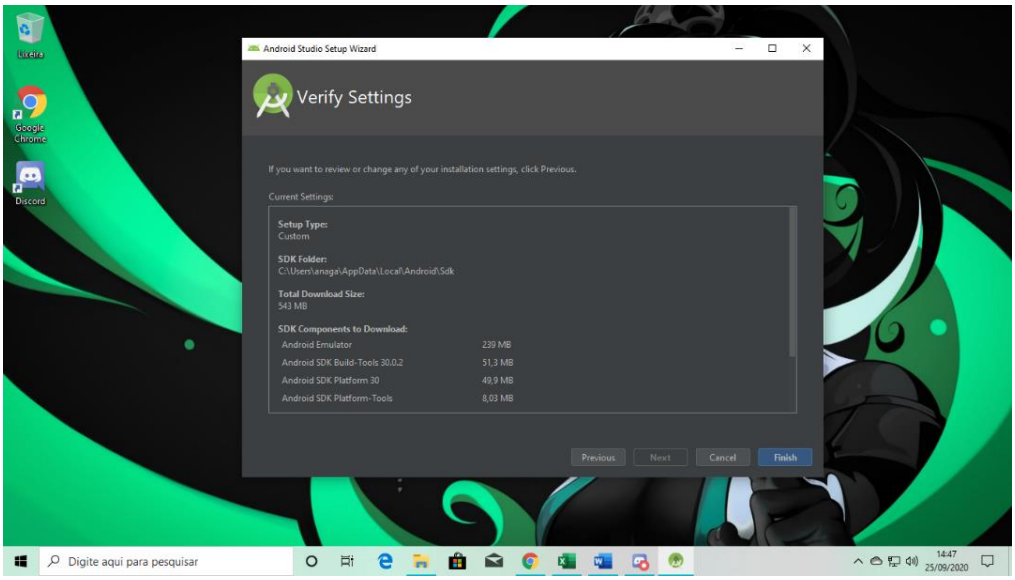
Aqui você pode selecionar o tema que mais lhe agrada.

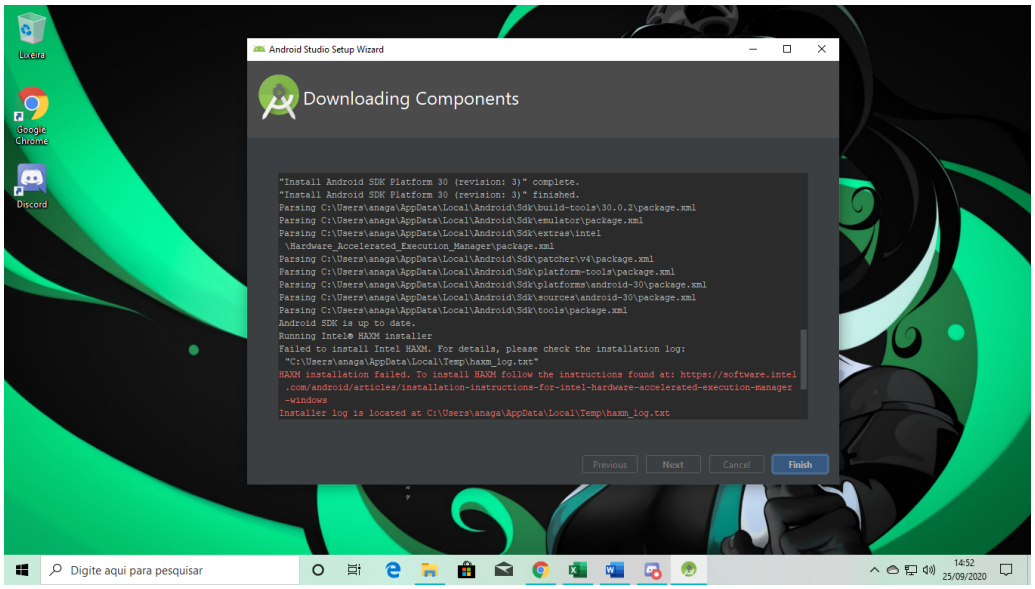


O HAXM é uma ferramenta exclusiva dos processadores Intel, que serve para aumentar a velocidade operacional do emulador do Android, portanto é recomendável prosseguir com essa opção selecionada desde que possua um processador desta linha

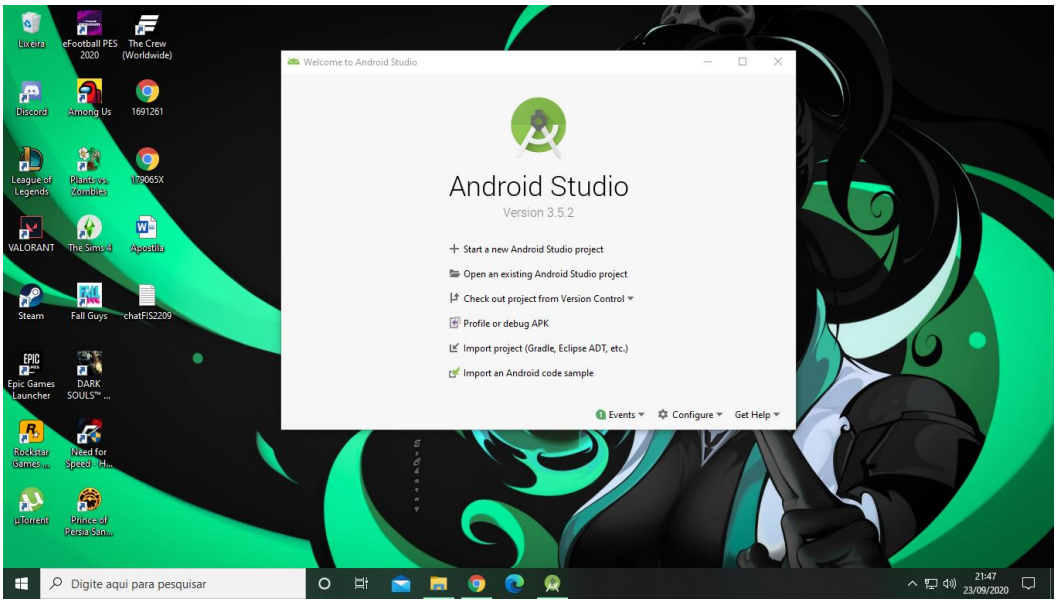


Caso seu processador tenha pouca memória ao baixar o HAXM, priorize definir entre 2GB a 1 GB.

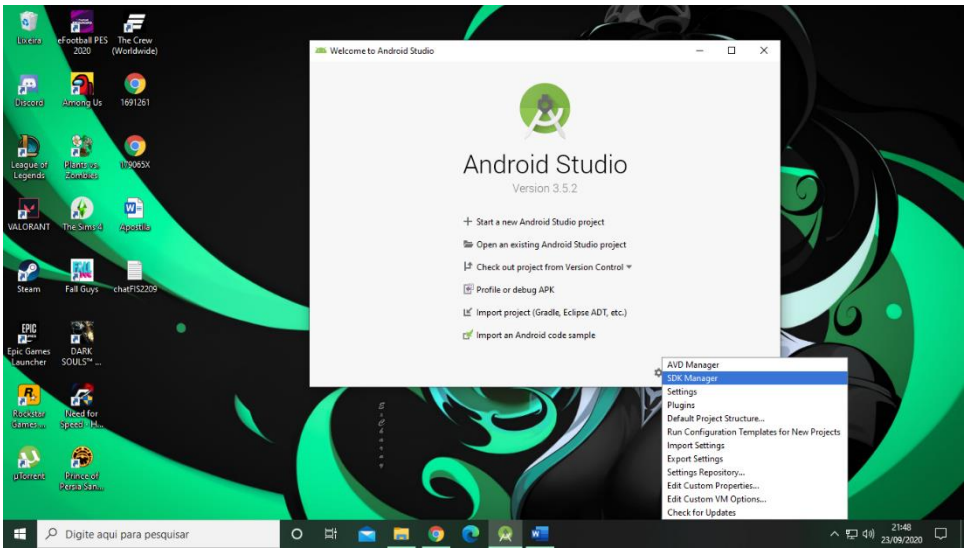


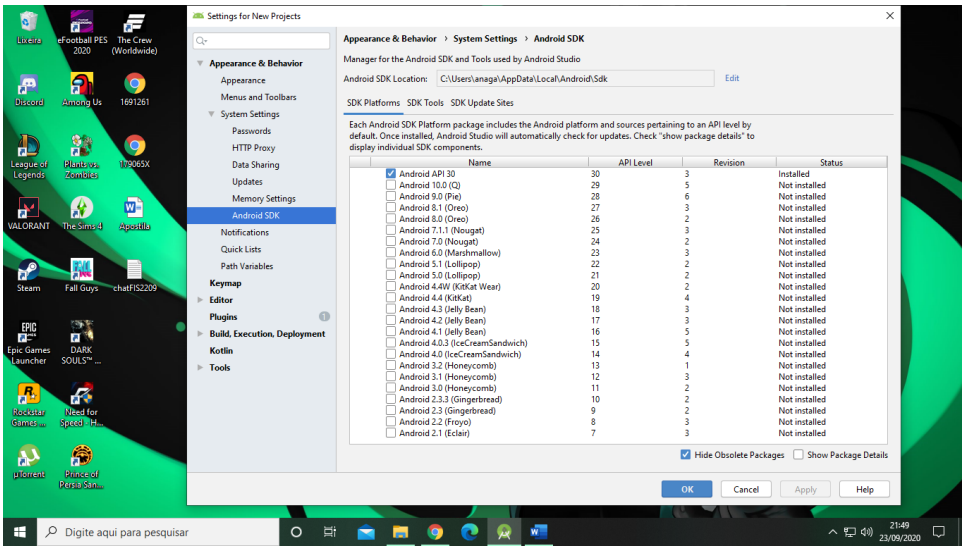


No meu caso não é possível instalar o HAXM, visto que meu processador é AMD.

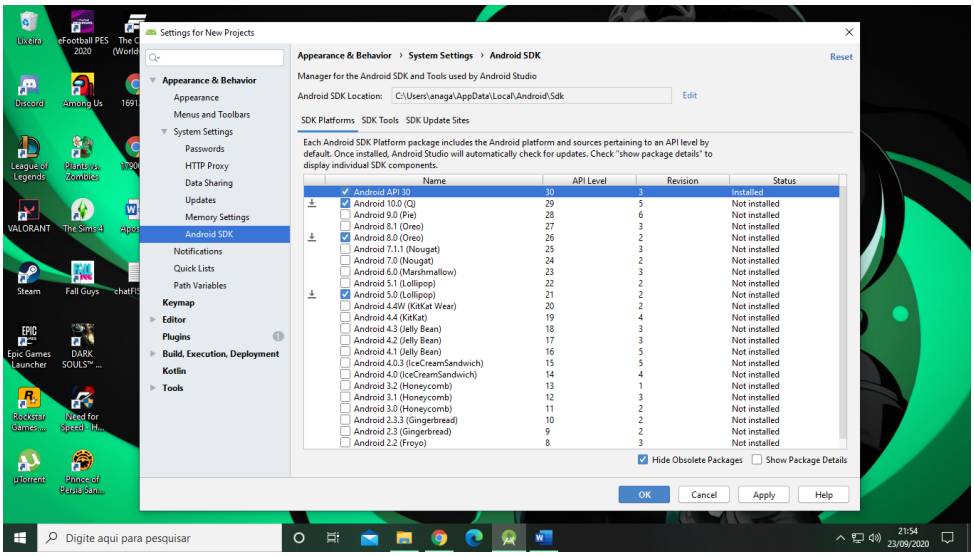


Atualize as configurações do SDK e faça o download dos Androids que serão utilizados:

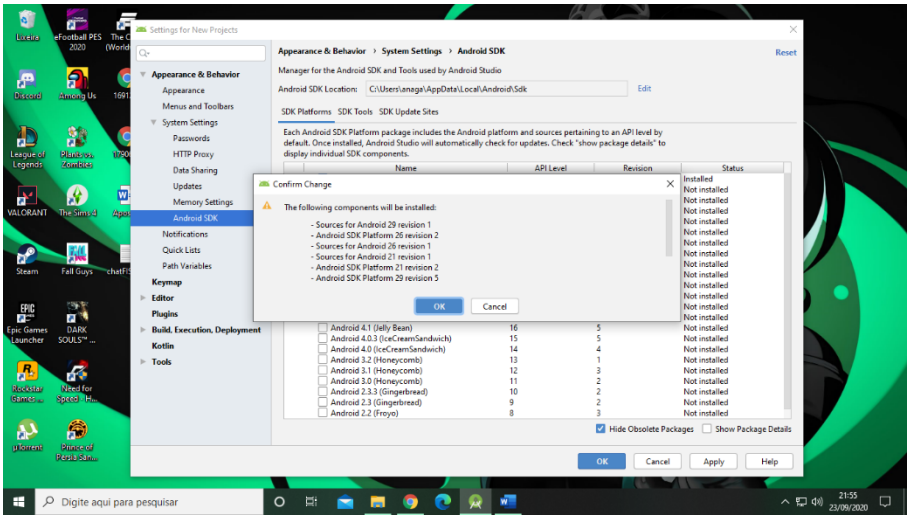


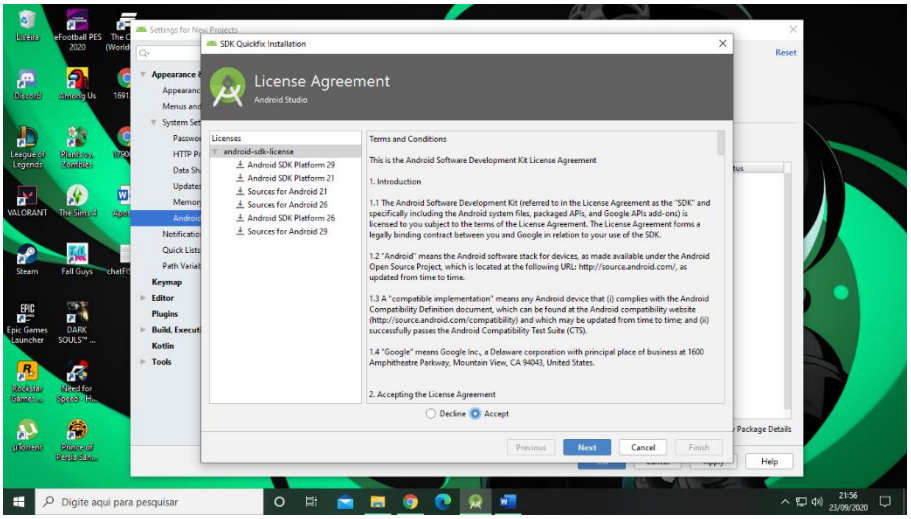


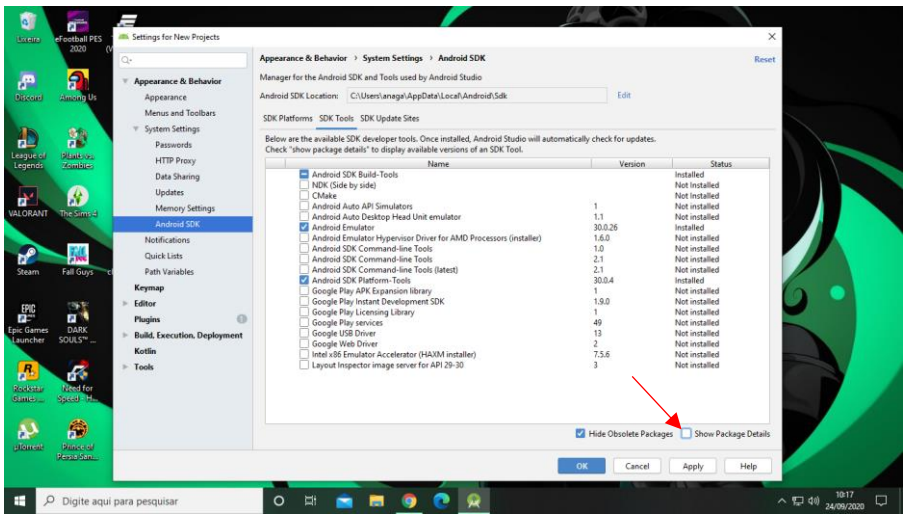
Não é indicado utilizar a versão mais recente do Android em razão desta versão apresentar possíveis avarias e/ou incompatibilidades.



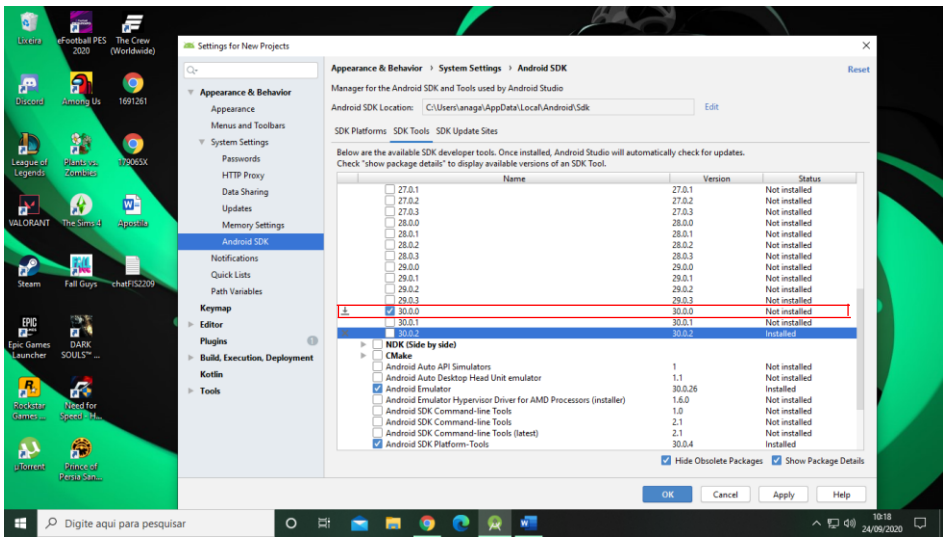
Clique em “Apply” para salvar suas alterações.

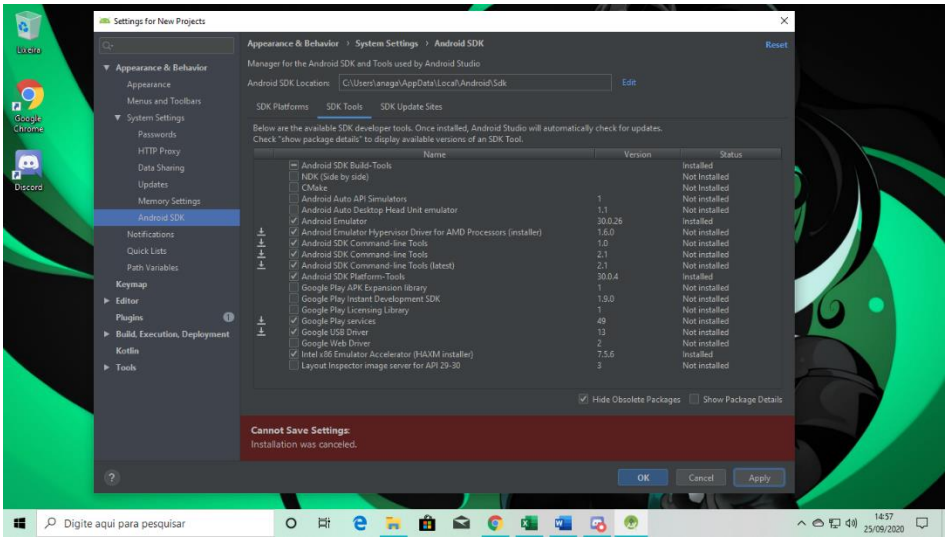




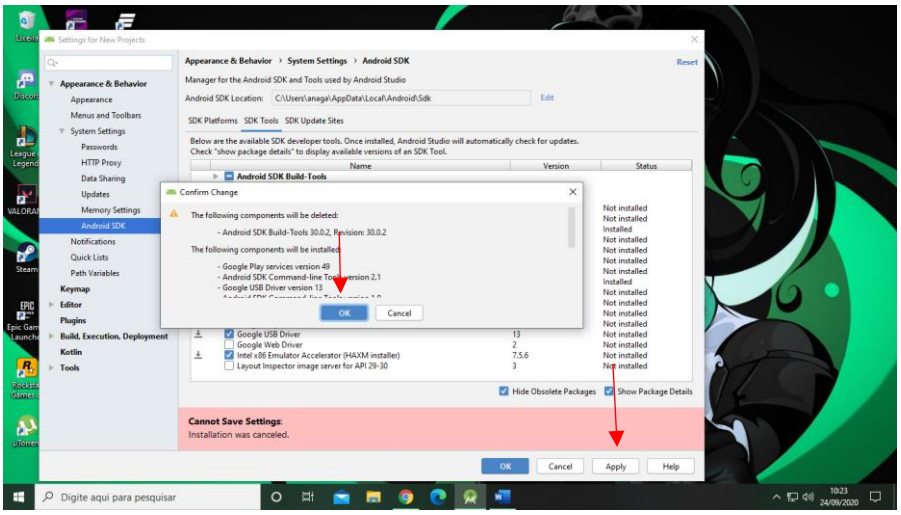


Ao expandir, na Opção “Android SDK Build Tools”, selecione a opção 30.0.0 e desmarque a 30.0.2.

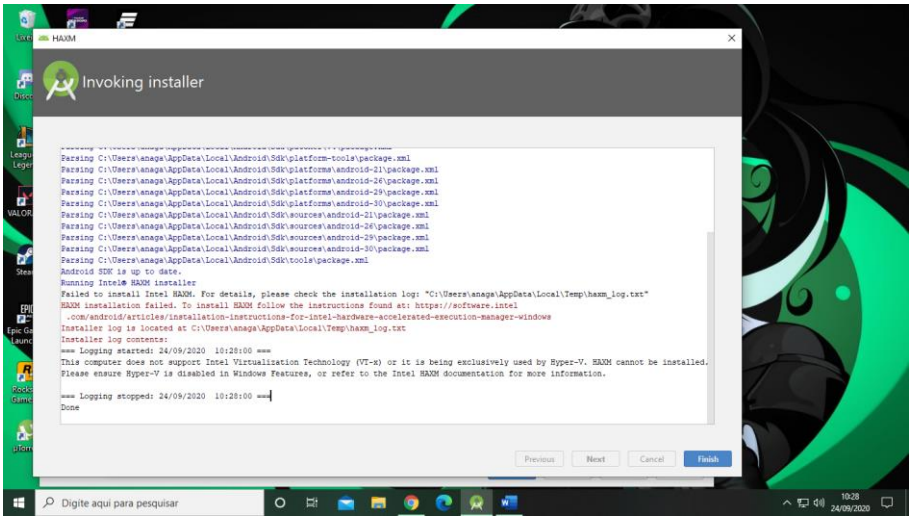




Caso não tenha instalado o HAXM no instante da instalação, apenas eleja a opção “Intel x86 Emulator Accelerator (HAXM Installer)”

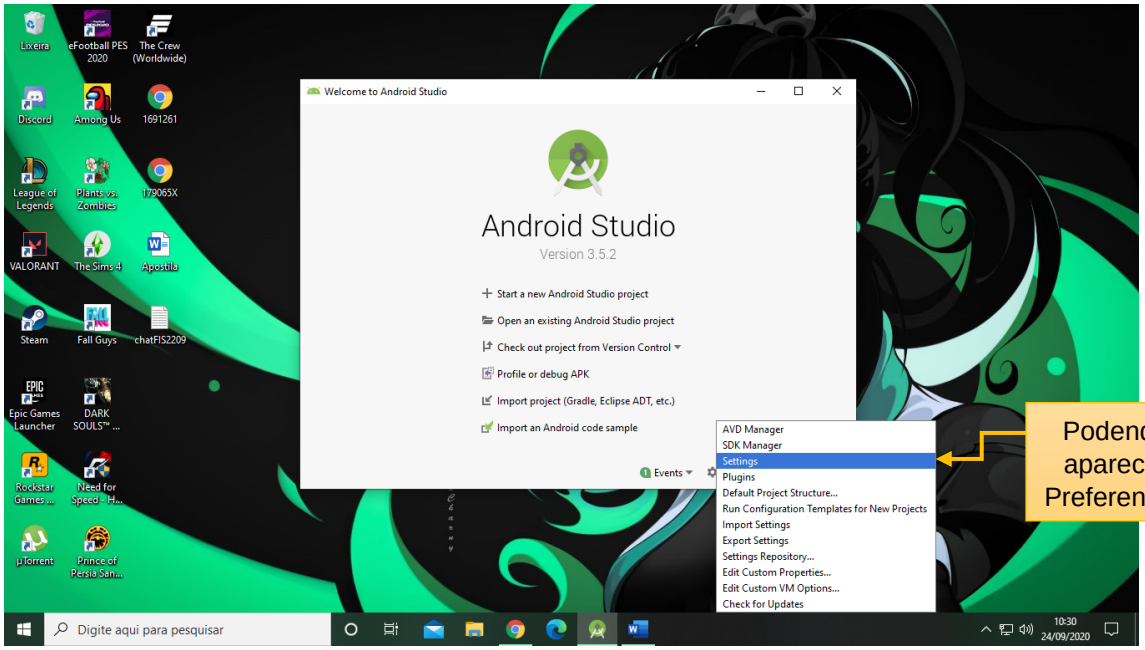


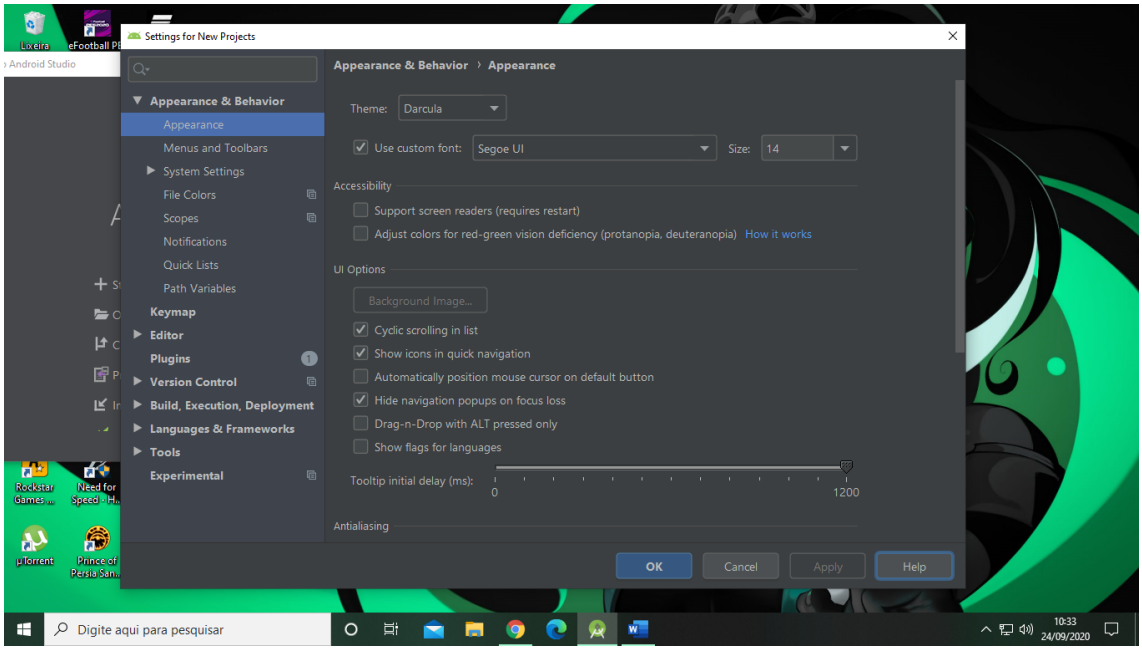
Click em “Apply” e em seguida “OK” para as devidas modificações de seu interesse.



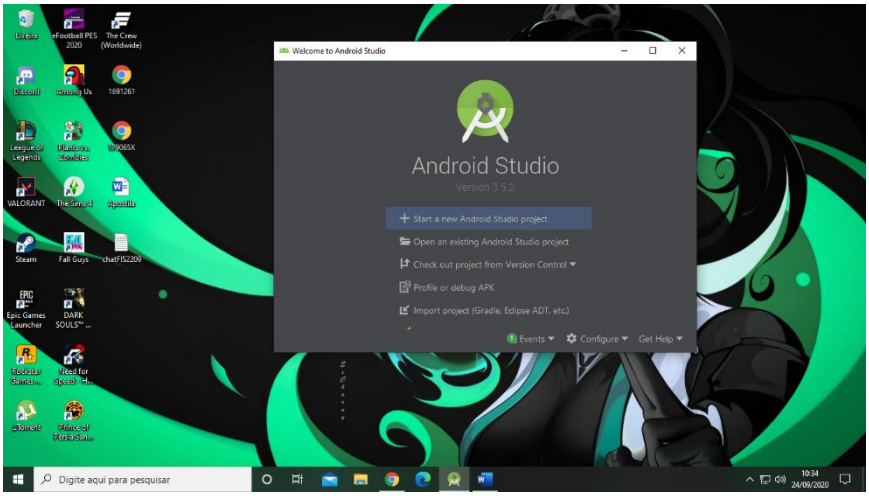
Nesta tela serão listados os recursos instalados e os possíveis erros apresentados no processo de instalação.

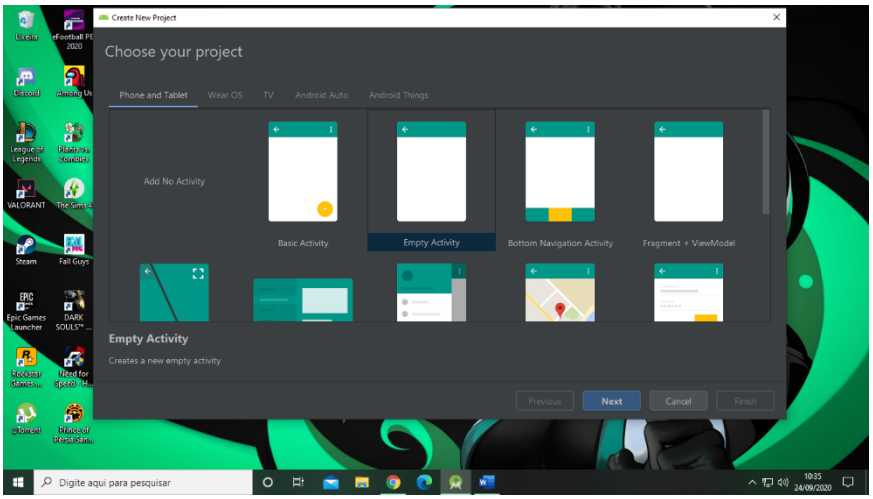
Personalização do Android Studio:

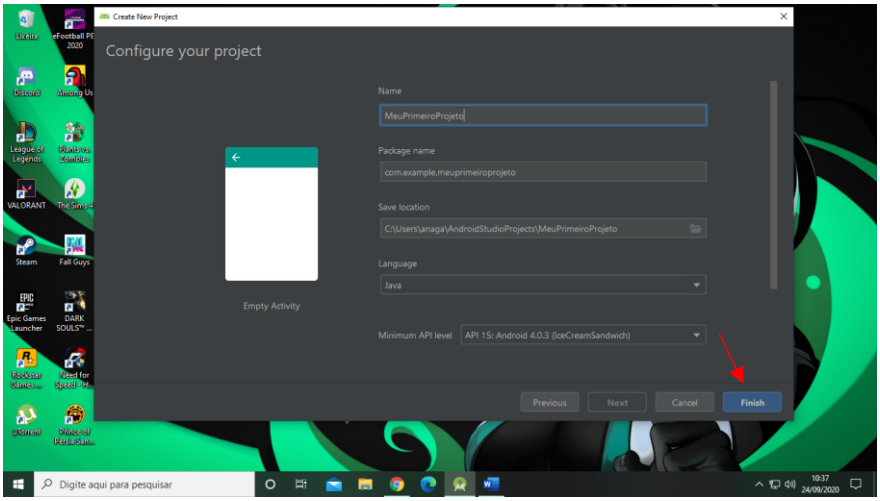


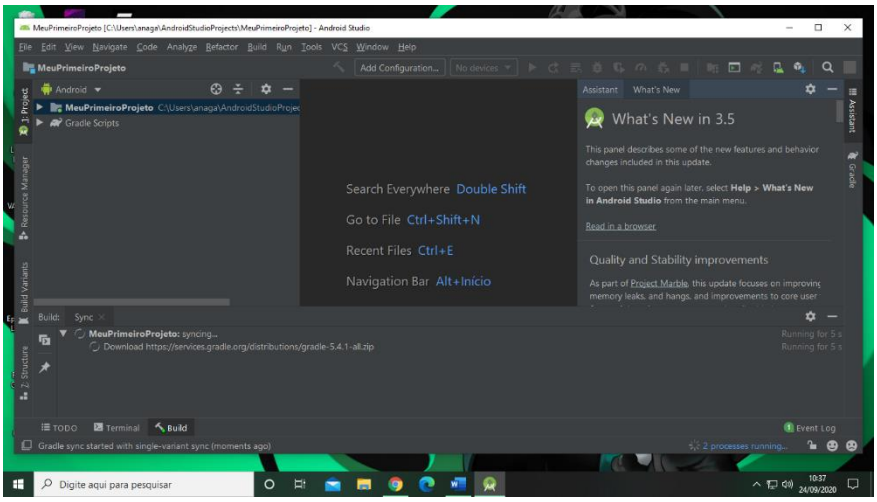


Primeiro programa

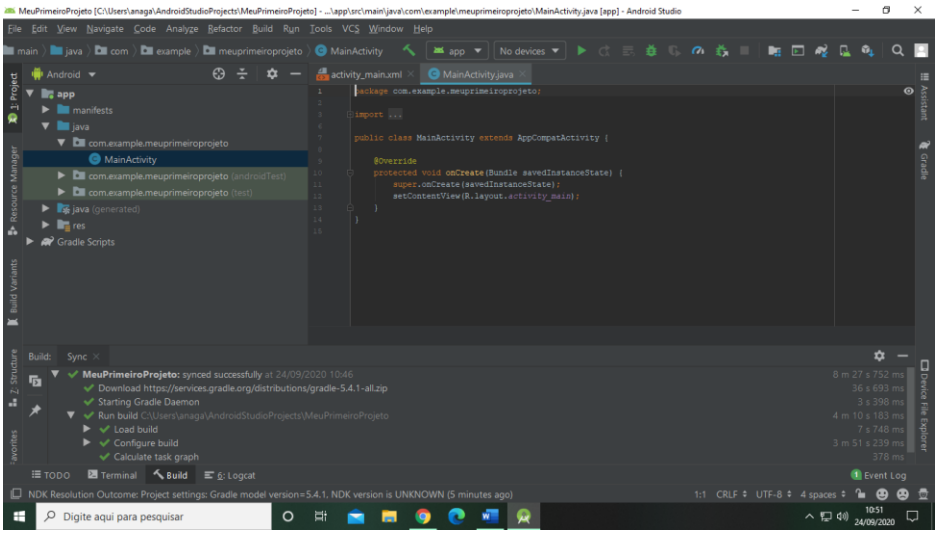




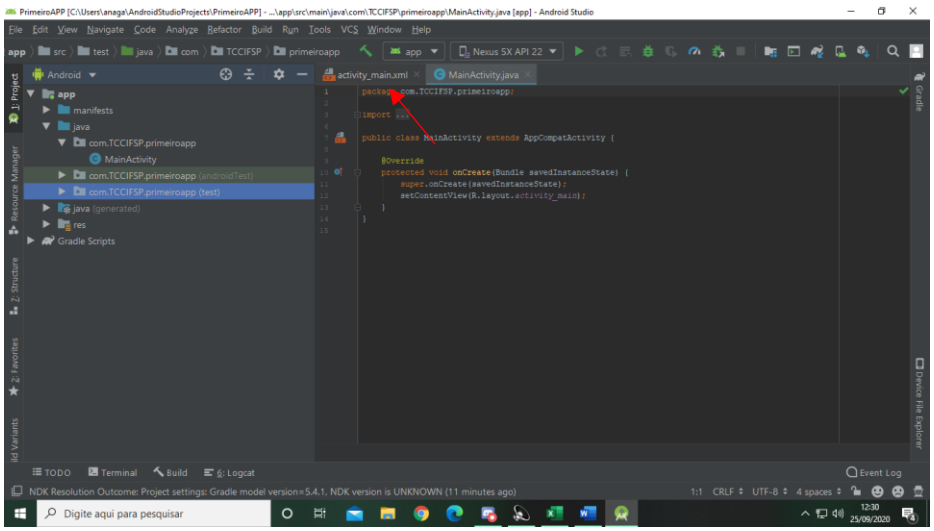




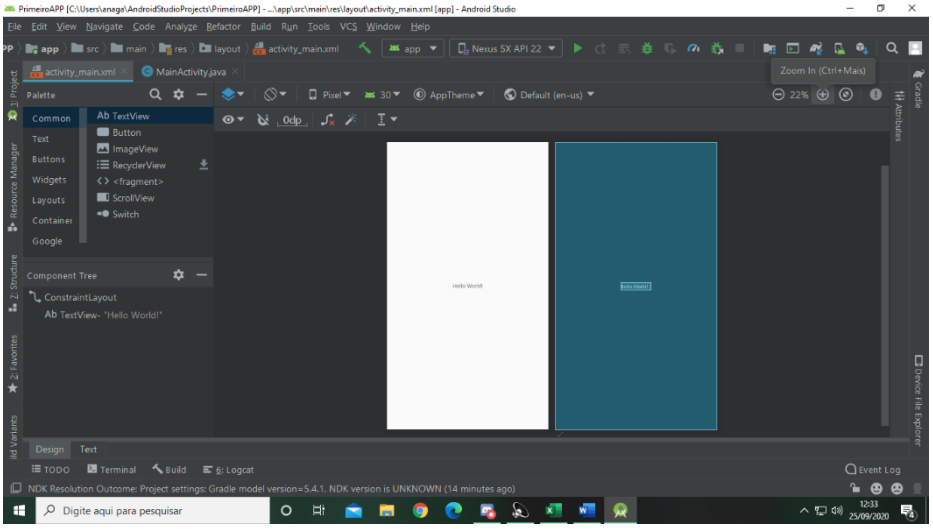
Após carregar todos os recursos apresentados na barra inferior do AS, a próxima tela abrirá automaticamente.

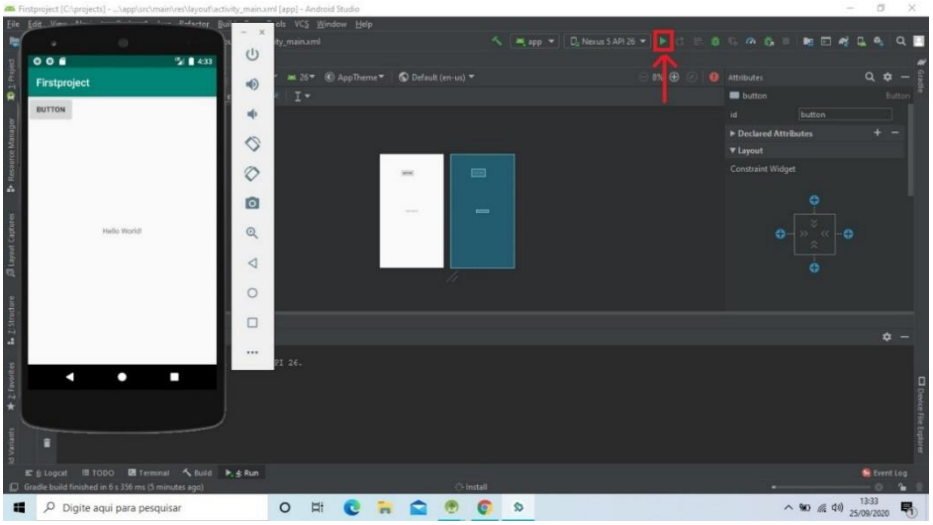


MaisActivity é a interface do app; um modo de remodelar a tela do seu App.



Para adicionar “button” ou TextView”, selecione-o e arraste até a telinha branca.



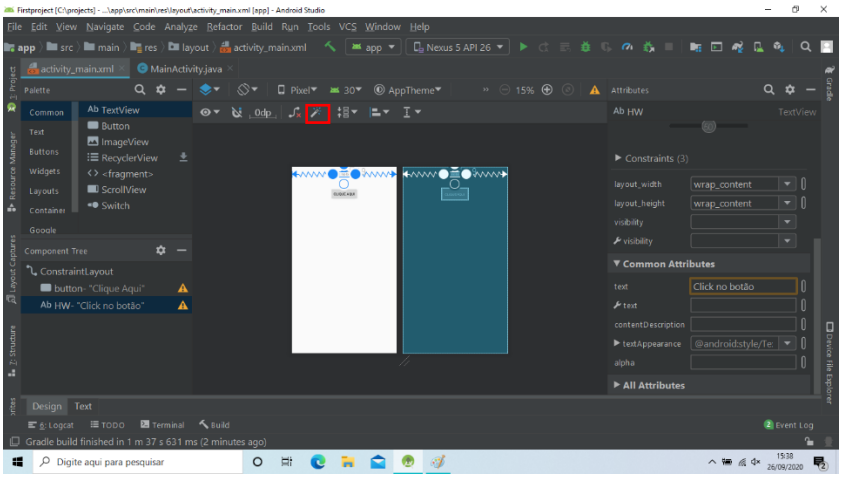


Se já criou um emulador, clique no botão marcado acima para a execução de seu programa.

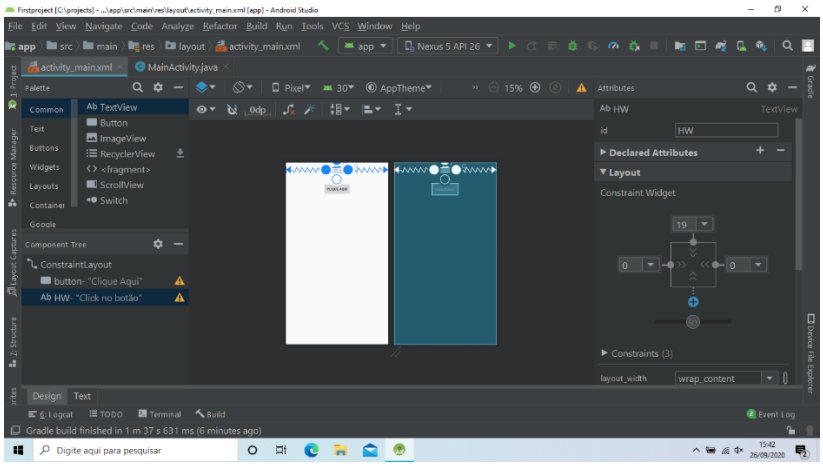
Criando um aplicativo que através de um botão exiba “Hello World”:

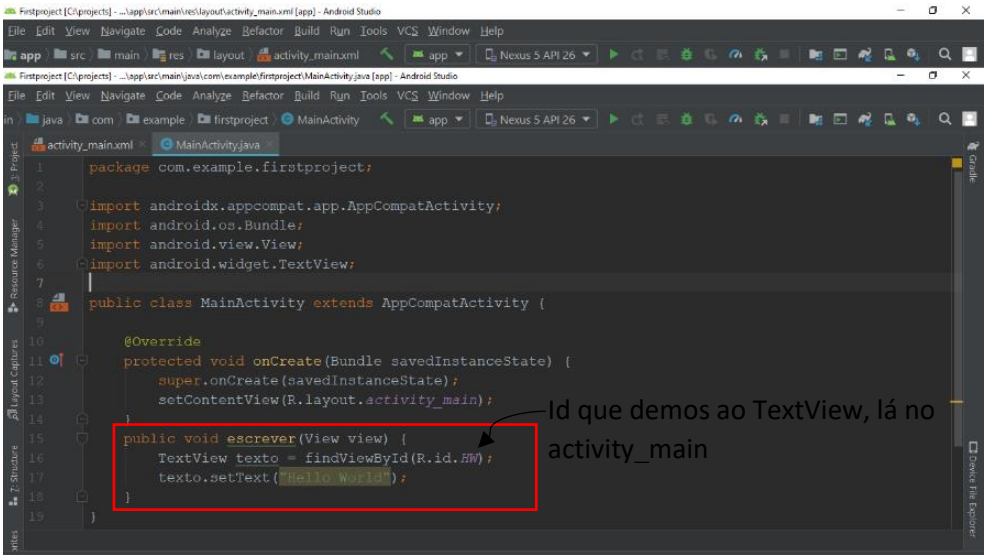
Clicando no item desejado, neste caso, o “TextView” e “Button”, arraste para o retângulo em branco.

Para que eles fiquem na posição preterida quando executado, deve-se clicar sobre a varinha com uns efeitos em azul destacado em vermelho, conforme imagem abaixo. Já para definir o que será exibido na TextView, vá em Attributes>Common Attrinutes>Text e digite o que deseja exibir (está grifado em laranja na imagem).



Ainda em “Attributes”, estabeleça um id para poder identificar este elemento no código.





Dentro do Método, foi criado uma ação para o botão colocado na interface do nosso App, chamada “escrever” que irá mudar o texto para “Hello World”.

Criar uma referência:

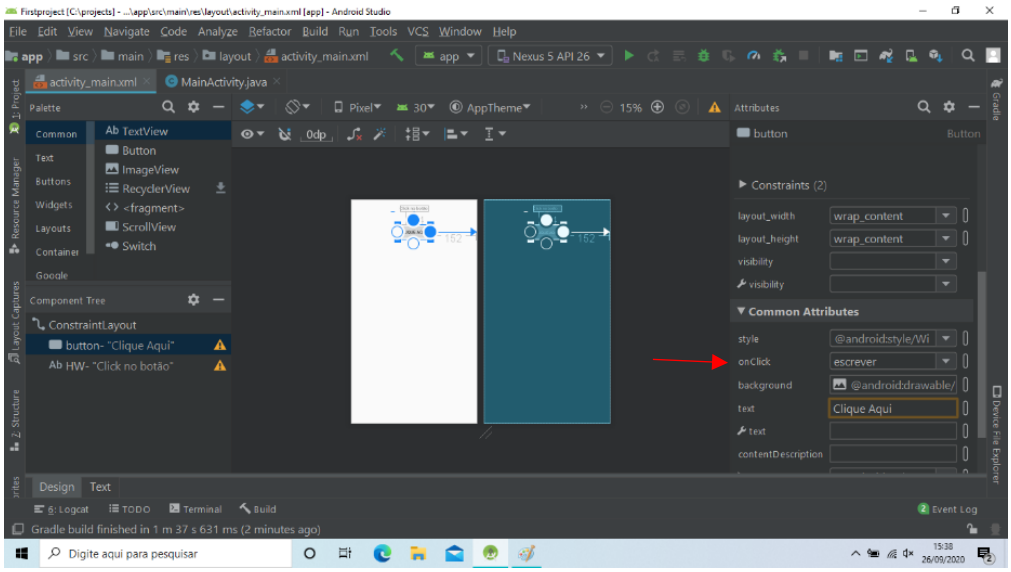
TextView texto = findViewById(R.id.HW);

Texto é nossa denominação para utilizarmos dentro do código;

findViewById -> encontre um componente de interface (Button, TextView, etc);

Utilizando nossa referência que criamos, ou seja, “Texto”:

setText = Configurar um texto (dar um valor, neste caso)



Ao criar dentro da class, voltamos ao “button” e suas propriedades, vamos em “onClick” e retomamos a ação que criamos anteriormente.

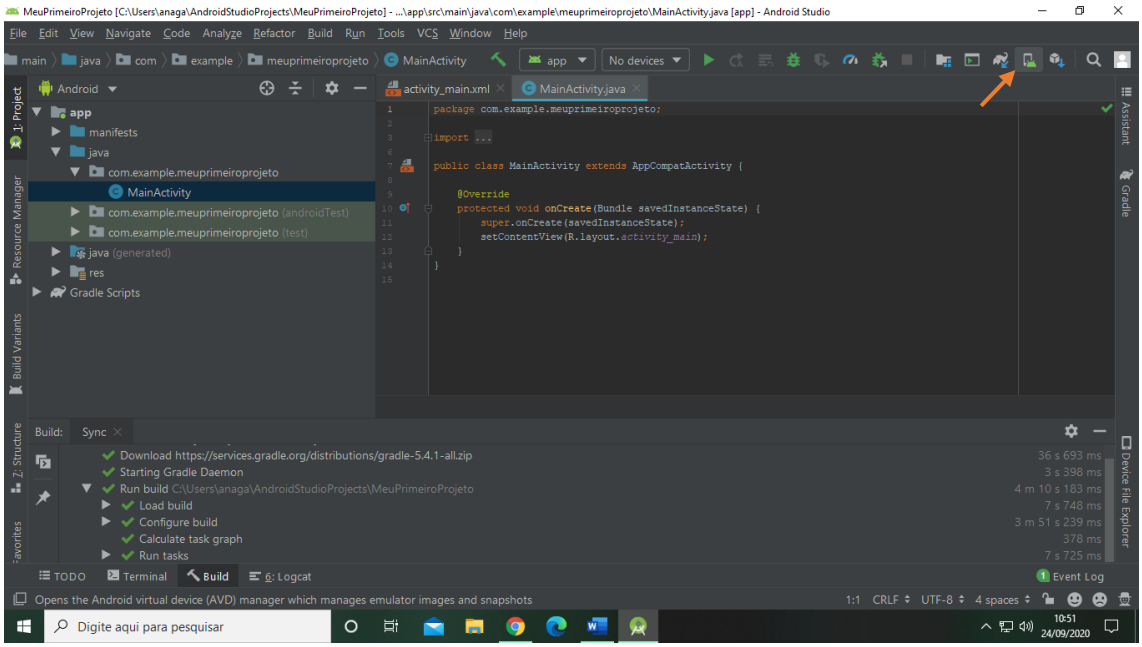
Ao clicar no botão apareceu nosso “Hello World”.

Criando o emulador

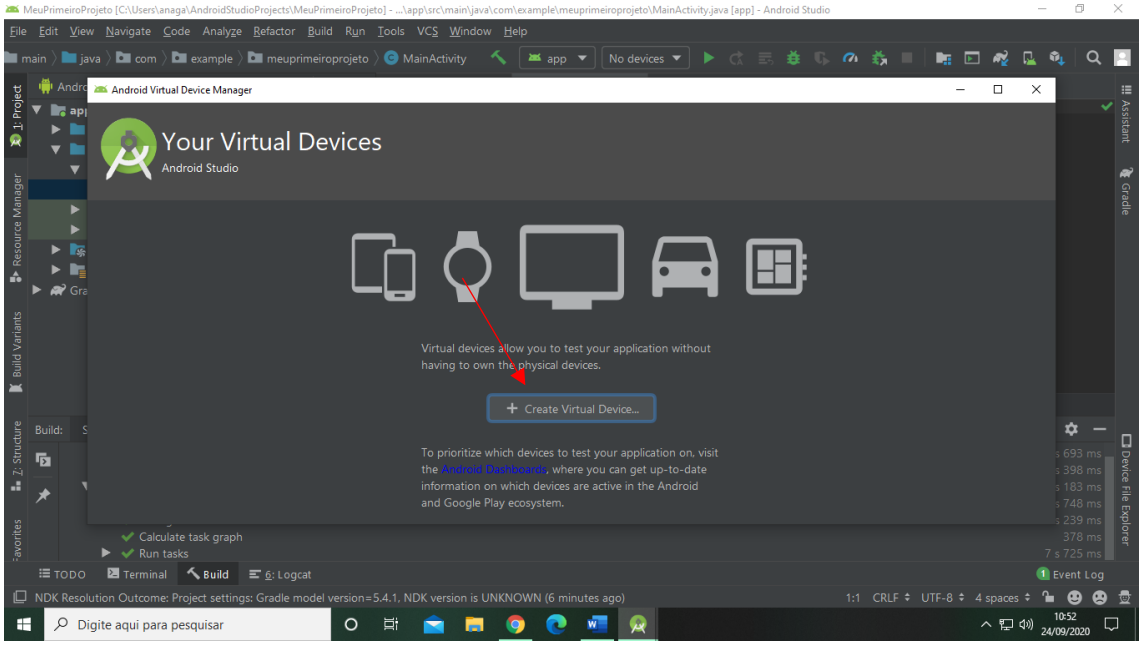
O que é virtualização?

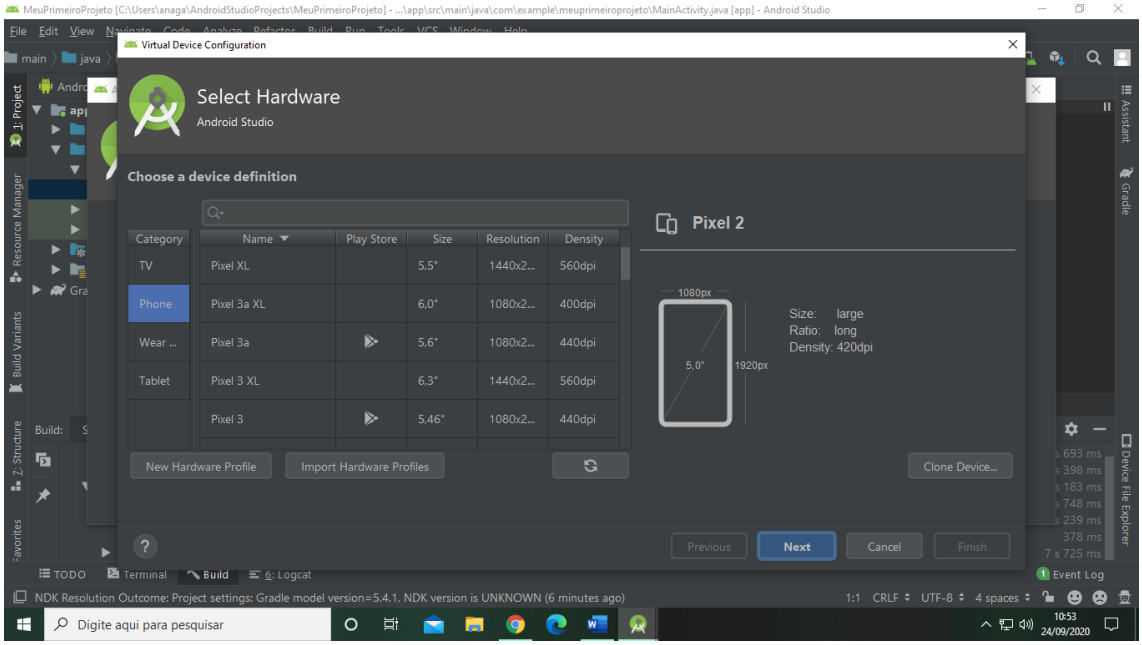
Uma tecnologia que torna processadores modernos executarem softwares desenvolvidos para processadores mais antigos, por meio de criar um ambiente virtual emulando o processador antigo. Caso esta opção não esteja habilitada na máquina, é necessário acessar a BIOS do dispositivo e ativar manualmente a virtualização navegando pelo menu de configuração da BIOS.

- Virtualização nos processadores Intel é chamado VT-X (Pode estar desabilitado);
- Virtualização nos processadores AMD é chamado AMD-V (Sempre habilitado).

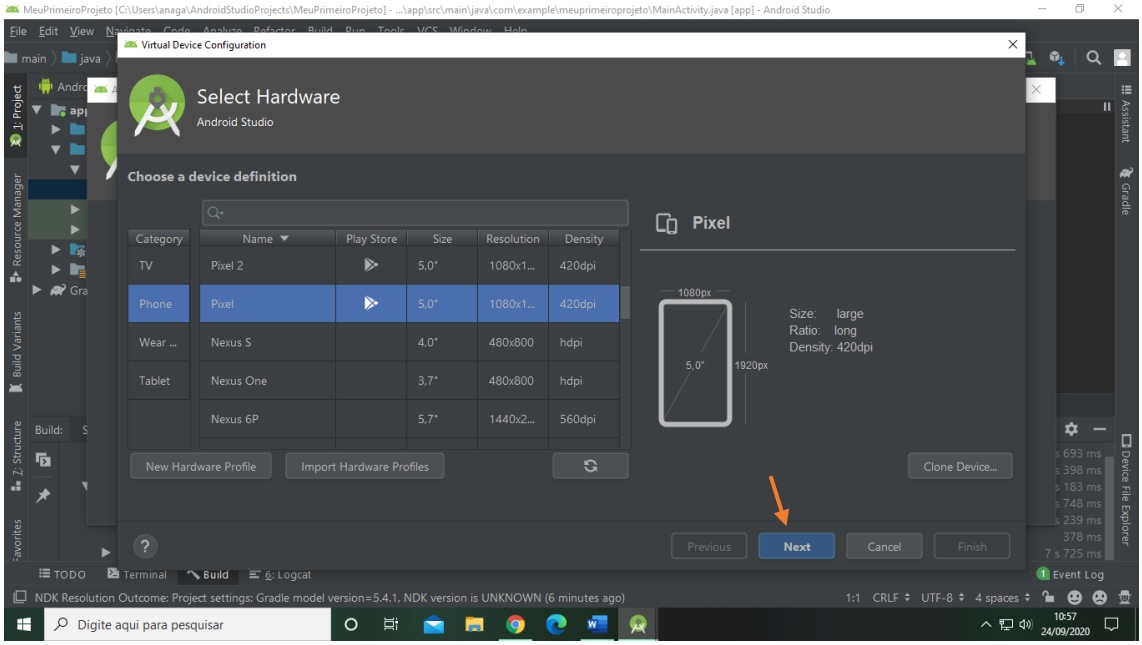


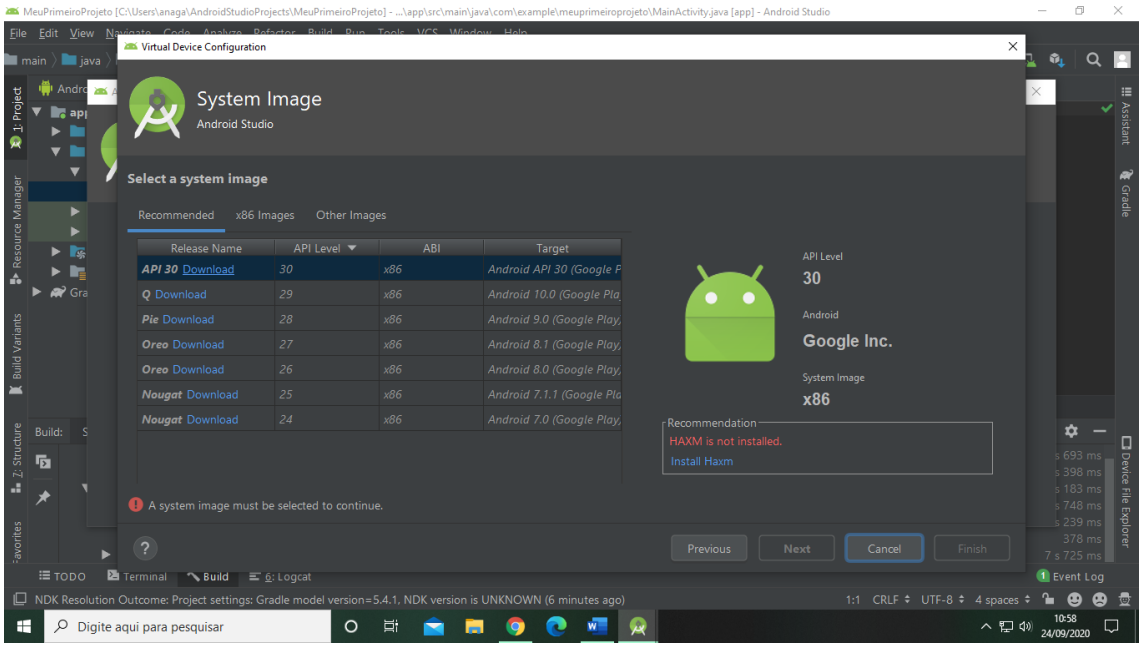
Clique no celular com o símbolo do Android;

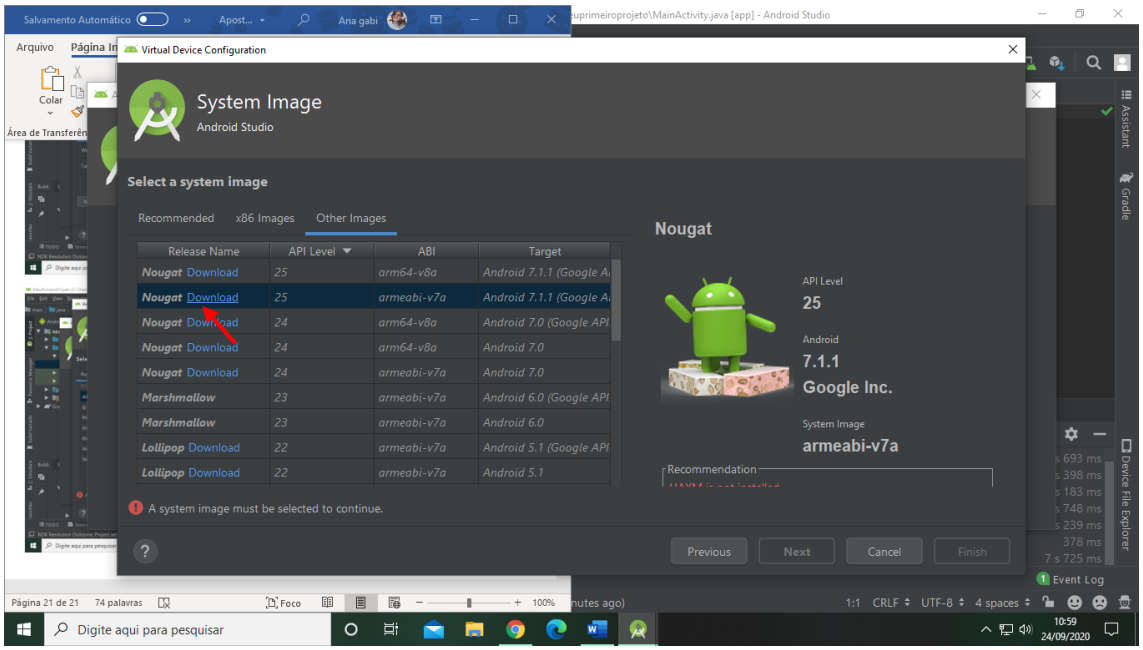




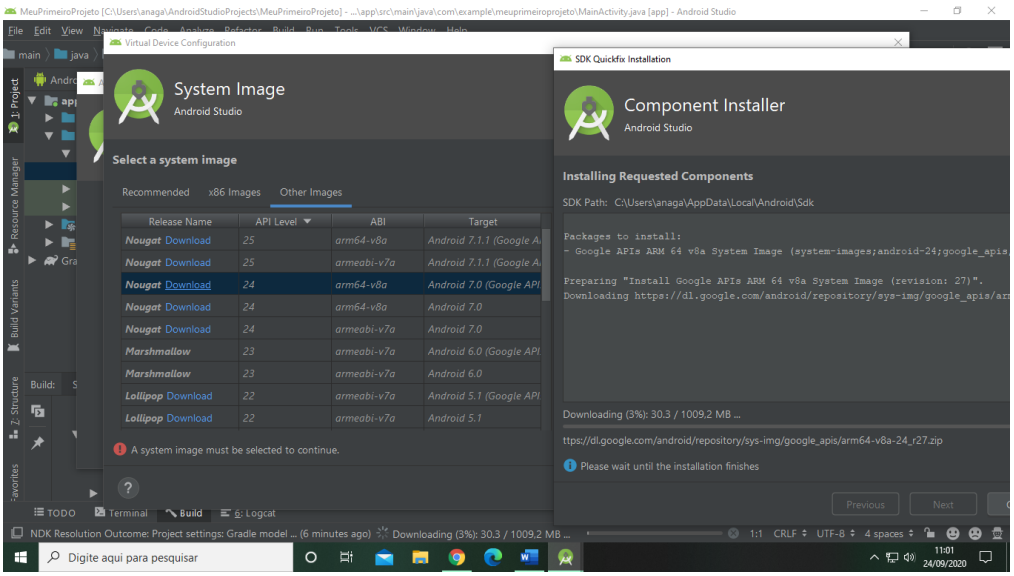
Escolha um modelo de celular;

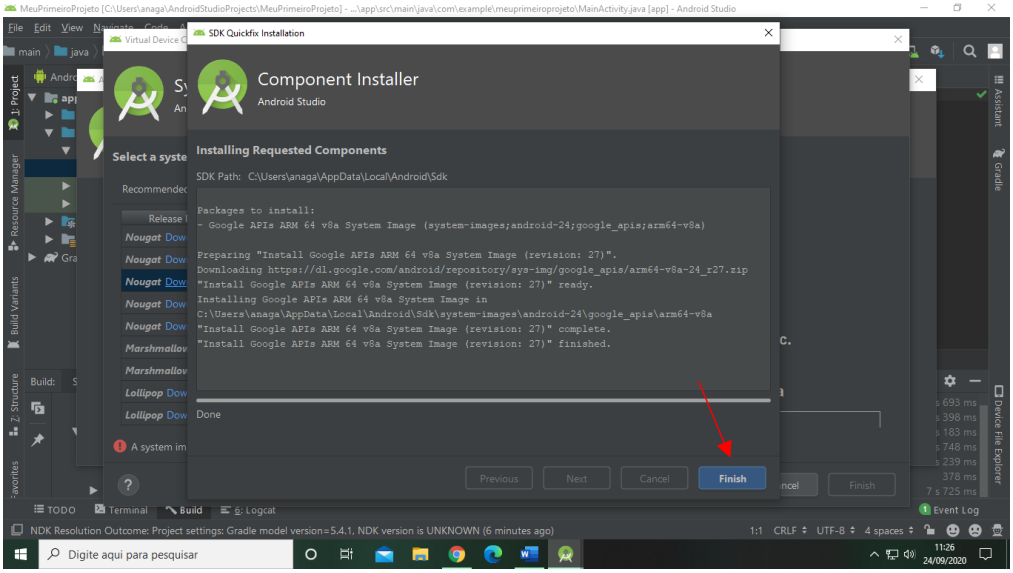

Virtual Device Configuration
Select Hardware
Android Studio
Choose a device definition
Category
Name
Play Store
Size
Resolution
Density
TV
Pixel 2
5.0"
1080x1...
420dpi
Phone
Pixel
5.0"
1080x1...
420dpi
Wear ...
Nexus S
4.0"
480x800
hdpi
Tablet
Nexus One
3.7"
480x800
hdpi
Nexus 6P
5.7"
1440x2...
560dpi
Pixel
1080px
1920px
5.0"
Size: large
Ratio: long
Density: 420dpi
New Hardware Profile
Import Hardware Profiles
Clone Device...
Previous
Next
Cancel
Finish
NDK Resolution Outcome: Project settings: Gradle model version=5.4.1, NDK version is UNKNOWN (6 minutes ago)
Digite aqui para pesquisar

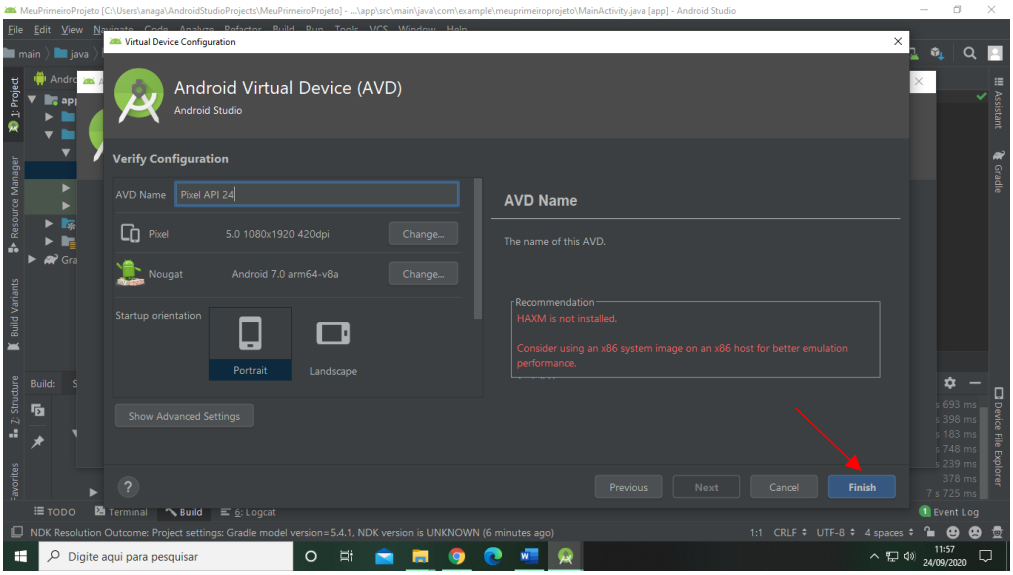

Virtual Device Configuration
System Image
Android Studio
Select a system image
Recommended
x86 Images
Other Images
Release Name
API Level
ABI
Target
API 30 Download
30
x86
Android API 30 (Google
Q Download
29
x86
Android 10.0 (Google Pla
Pie Download
28
x86
Android 9.0 (Google Play
Oreo Download
27
x86
Android 8.1 (Google Play
Oreo Download
26
x86
Android 8.0 (Google Play
Nougat Download
25
x86
Android 7.1.1 (Google Pla
Nougat Download
24
x86
Android 7.0 (Google Play
API Level
30
Android
Google Inc.
System Image
x86
Recommendation
HAXM is not installed.
Install Haxm
A system image must be selected to continue.
Previous
Next
Cancel
Finish
NDK Resolution Outcome: Project settings: Gradle model version=5.4.1, NDK version is UNKNOWN (6 minutes ago)
Digite aqui para pesquisar

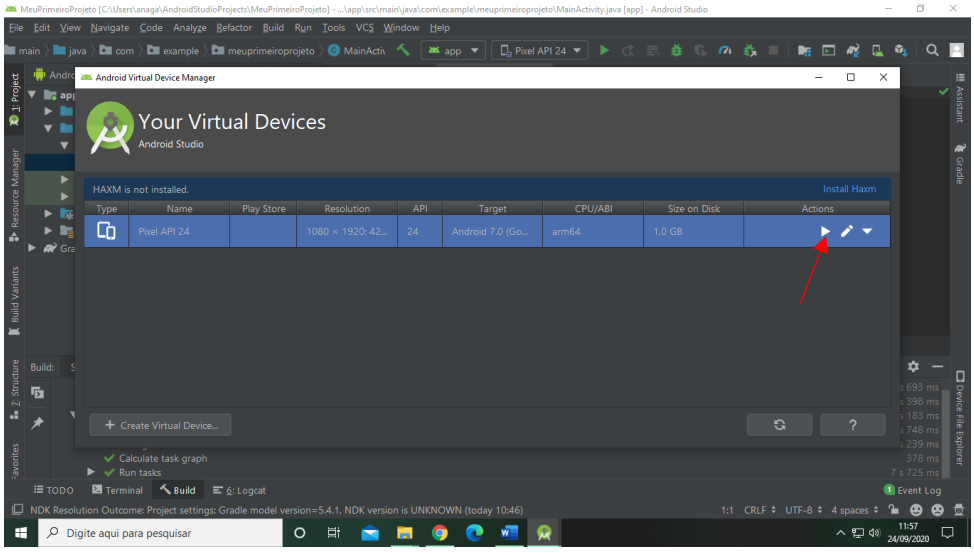


Caso tenha o HAXM, pode-se usar o x86 Images (e é o mais indicado), caso não tenha, como foi o meu caso, use “other images”. Ao escolher o Android que irá utilizar, clique em “Download”, aguarde o tempo de carregamento;

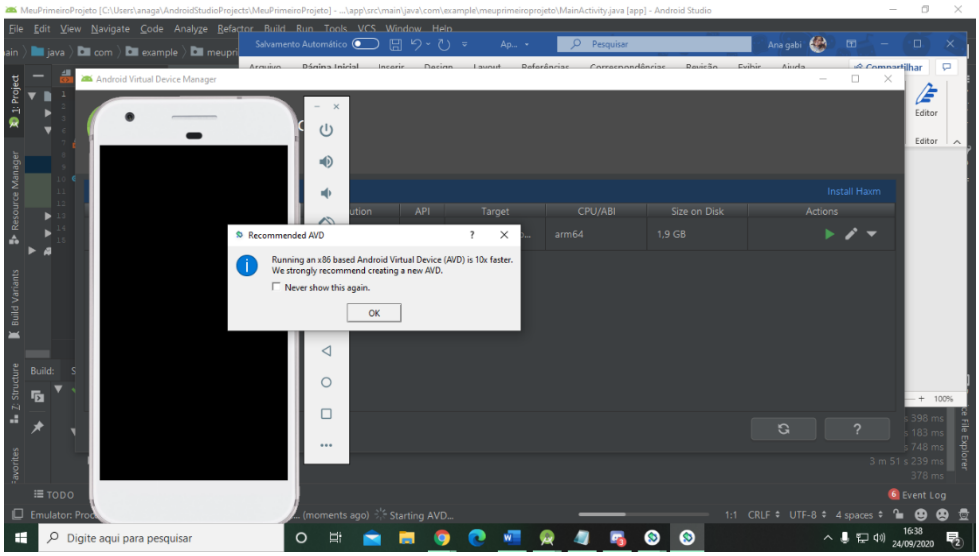






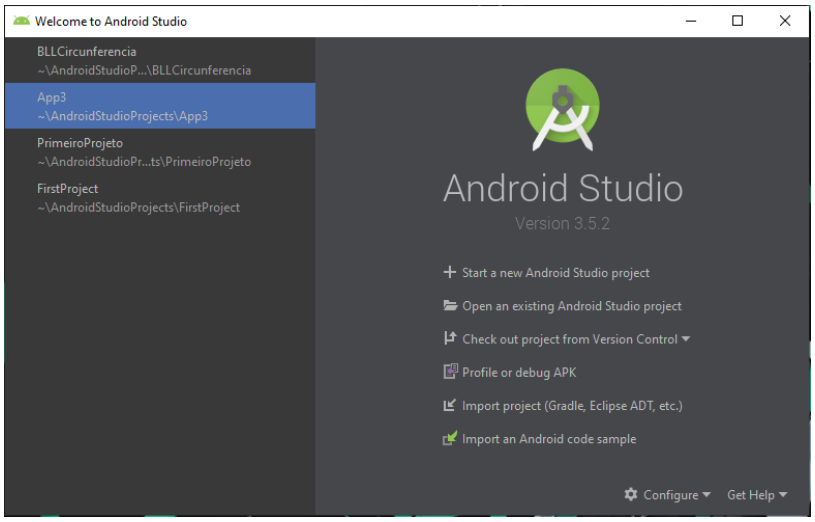


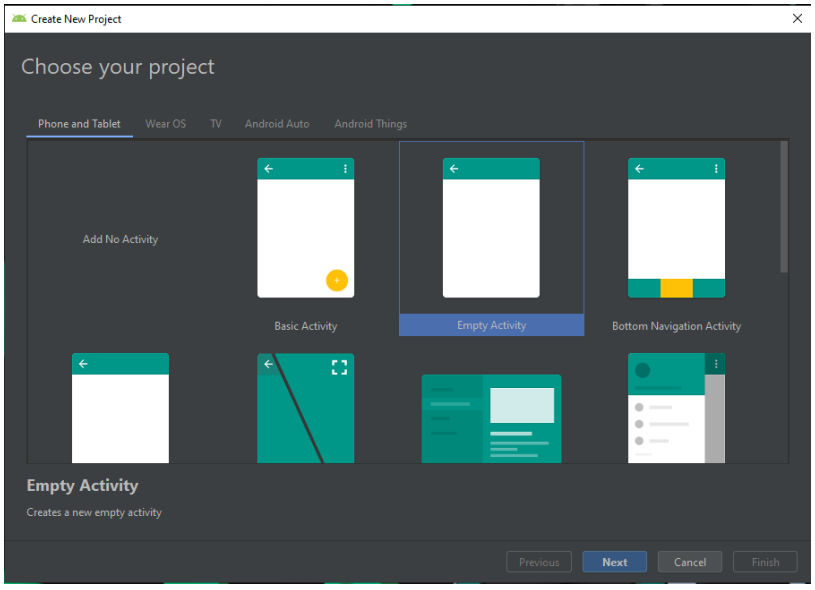
Clique no triangulo para executar seu emulador.

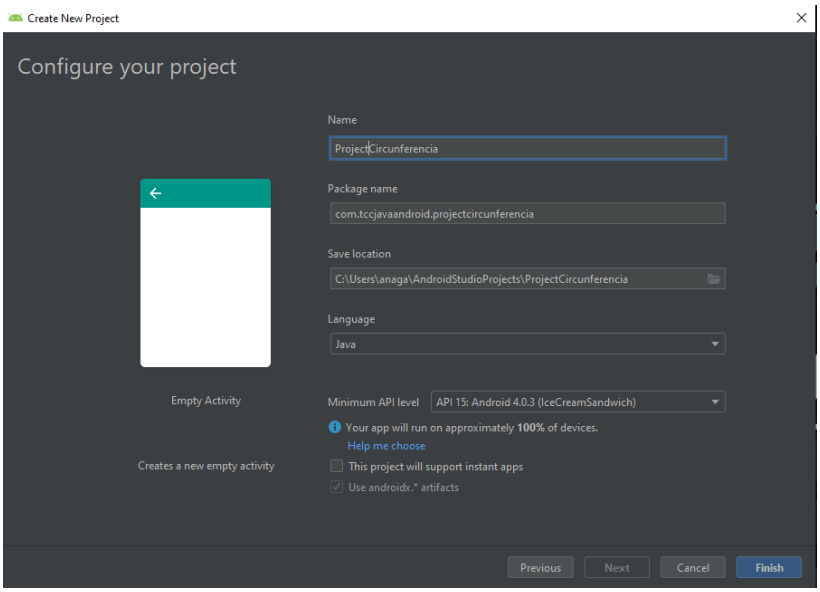

Android Virtual Device Manager
Recommended AVD
Running an x86 based Android Virtual Device (AVD) is 10x faster.
We strongly recommend creating a new AVD.
Never show this again.
OK
Install Haxm
API
Target
CPU/ABI
Size on Disk
Actions
arm64
1,9 GB
Digite aqui para pesquisar

Aplicativo 01: Calcular circunferência

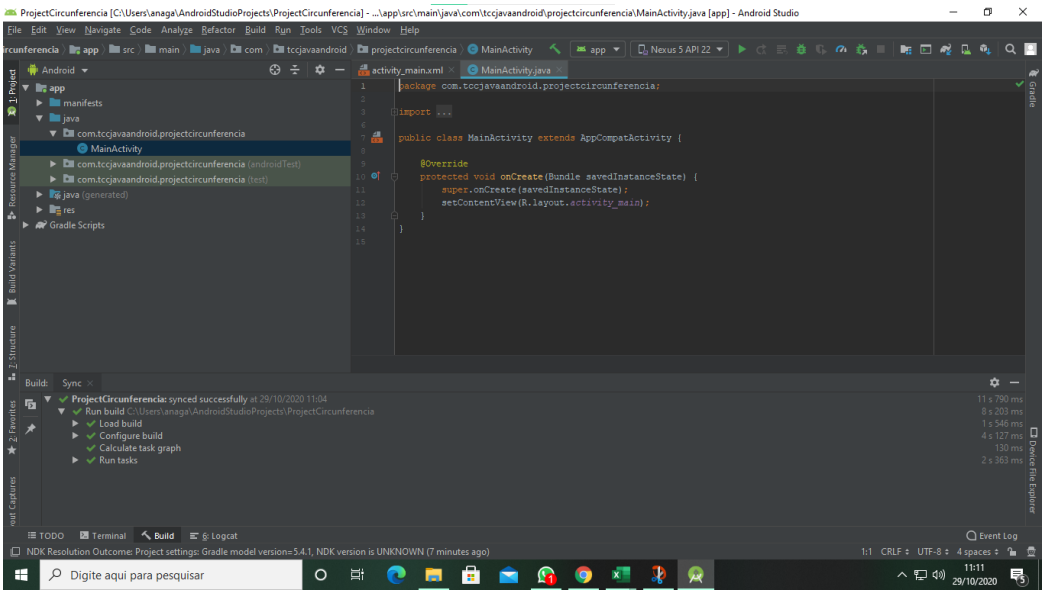
O primeiro passo é criar o projeto.

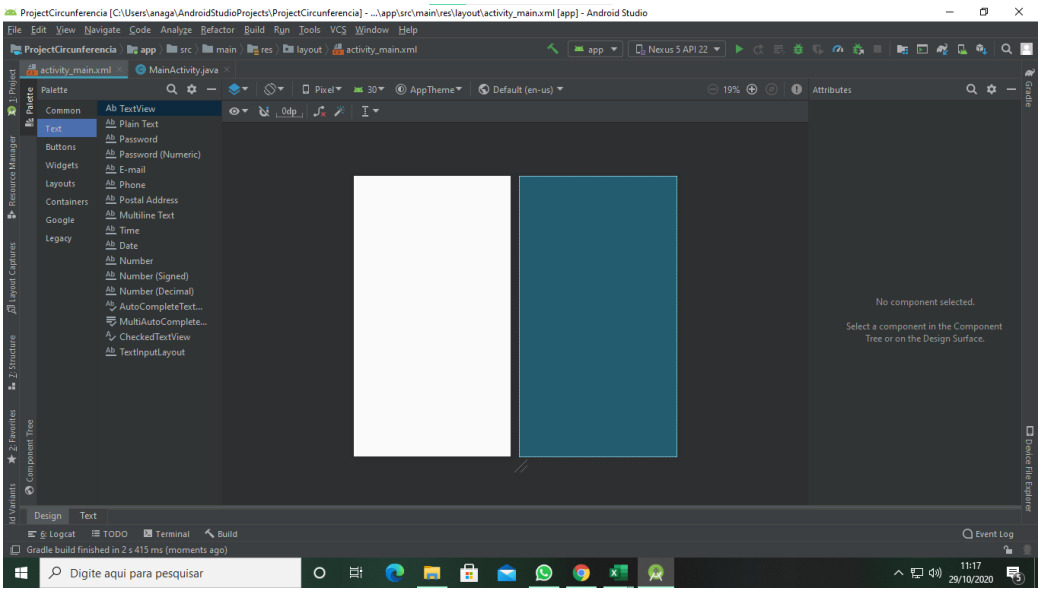


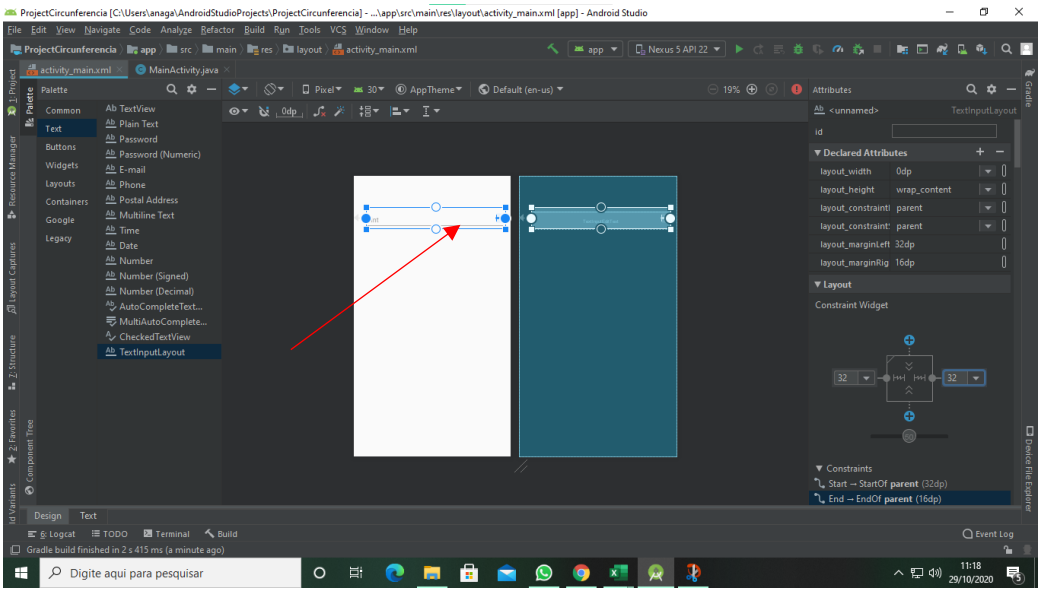


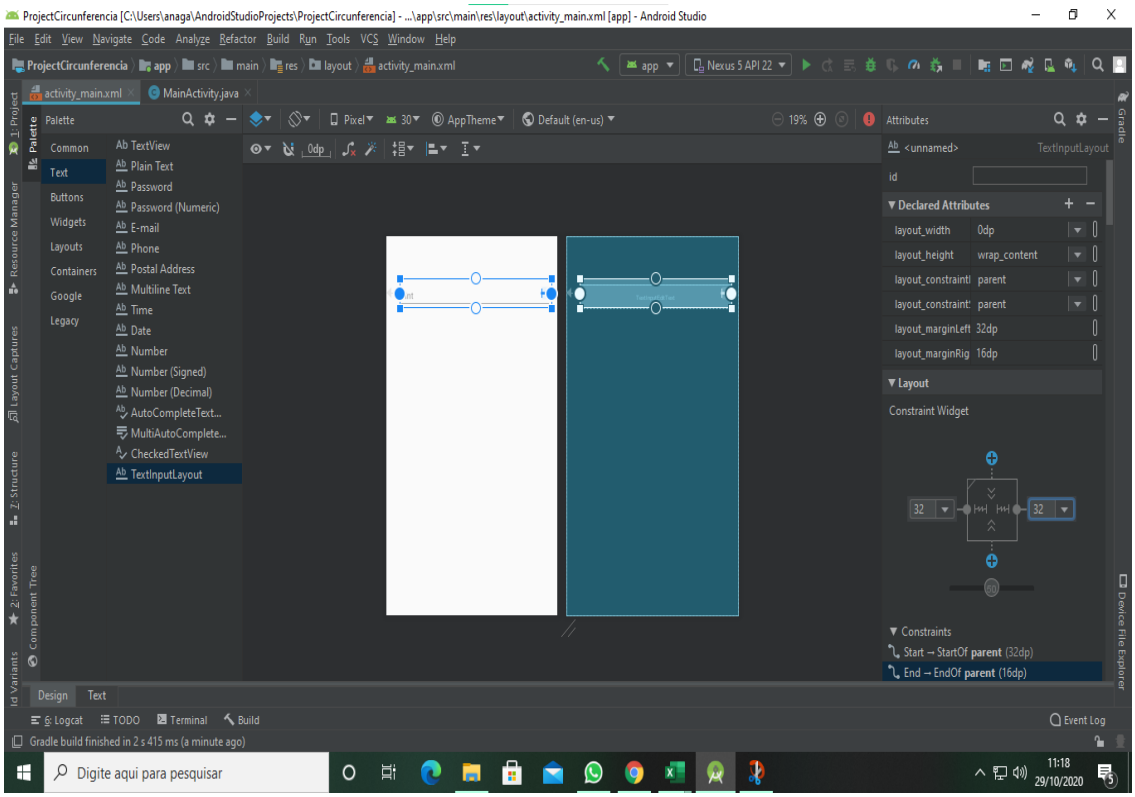


O segundo passo, com o projeto já criado, é ir em “activity_main.xml” para adicionar os campos de textos e botões.




ProjectCircunferencia [C:\Users\anaga\AndroidStudioProjects\ProjectCircunferencia] - ...\app\src\main\res\layout\activity_main.xml [app] - Android Studio
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
activity_main.xml
MainActivity.java
Palette
Common
Text
Buttons
Widgets
Layouts
Containers
Google
Legacy
Ab TextView
Plain Text
Password
Password (Numeric)
E-mail
Phone
Postal Address
Multiline Text
Time
Date
Number
Number (Signed)
Number (Decimal)
AutoCompleteText...
MultiAutoComplete...
CheckedTextView
TextInputLayout
Attributes
No component selected.
Select a component in the Component Tree or on the Design Surface.
Design
Text
Logcat
TODO
Terminal
Build
Event Log
Gradle build finished in 2 s 415 ms (moments ago)
Digite aqui para pesquisar
11:17
29/10/2020

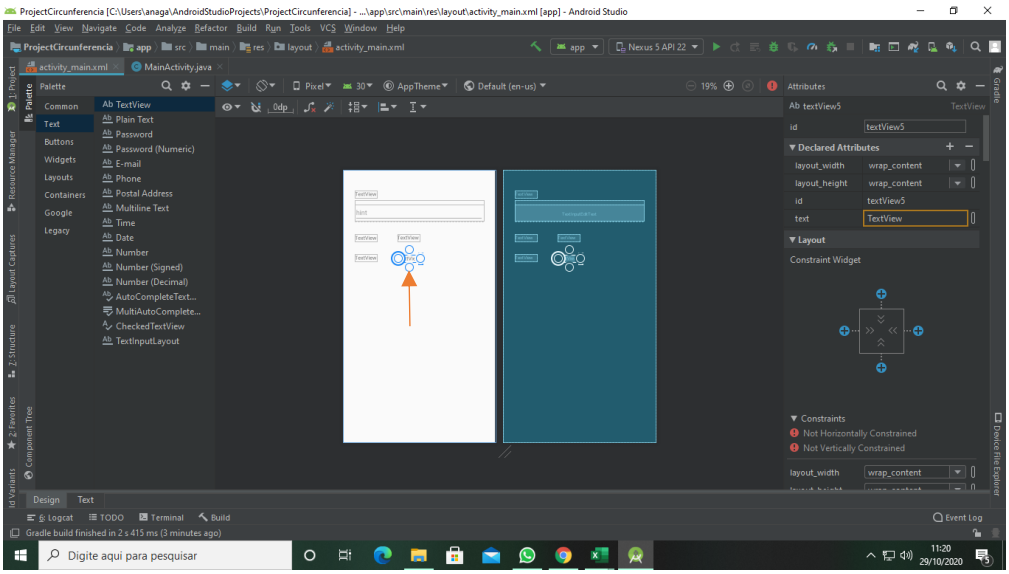

ProjectCircunferencia [C:\Users\anaga\AndroidStudioProjects\ProjectCircunferencia] - ...\app\src\main\res\layout\activity_main.xml [app] - Android Studio
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
activity_main.xml
MainActivity.java
Palette
Common
Text
Buttons
Widgets
Layouts
Containers
Google
Legacy
Ab TextView
Plain Text
Password
Password (Numeric)
E-mail
Phone
Postal Address
Multiline Text
Time
Date
Number
Number (Signed)
Number (Decimal)
AutoCompleteText...
MultiAutoComplete...
CheckedTextView
TextInputLayout
Attributes
<unnamed>
TextInputLayout
id
Declared Attributes
layout_width 0dp
layout_height wrap_content
layout_constraint parent
layout_constraint parent
layout_marginLeft 32dp
layout_marginRig 16dp
Layout
Constraint Widget
32
32
Constraints
Start → StartOf parent (32dp)
End → EndOf parent (16dp)
Design
Text
Logcat
TODO
Terminal
Build
Event Log
Gradle build finished in 2 s 415 ms (a minute ago)
Digite aqui para pesquisar
11:18
29/10/2020



No terceiro passo, como indicado pela seta vermelha, araste o objeto desejado até a tela do app para adicioná-lo ao projeto, este objeto servirá para a entrada do valor do Raio que permitirá calcular o perímetro e a área.

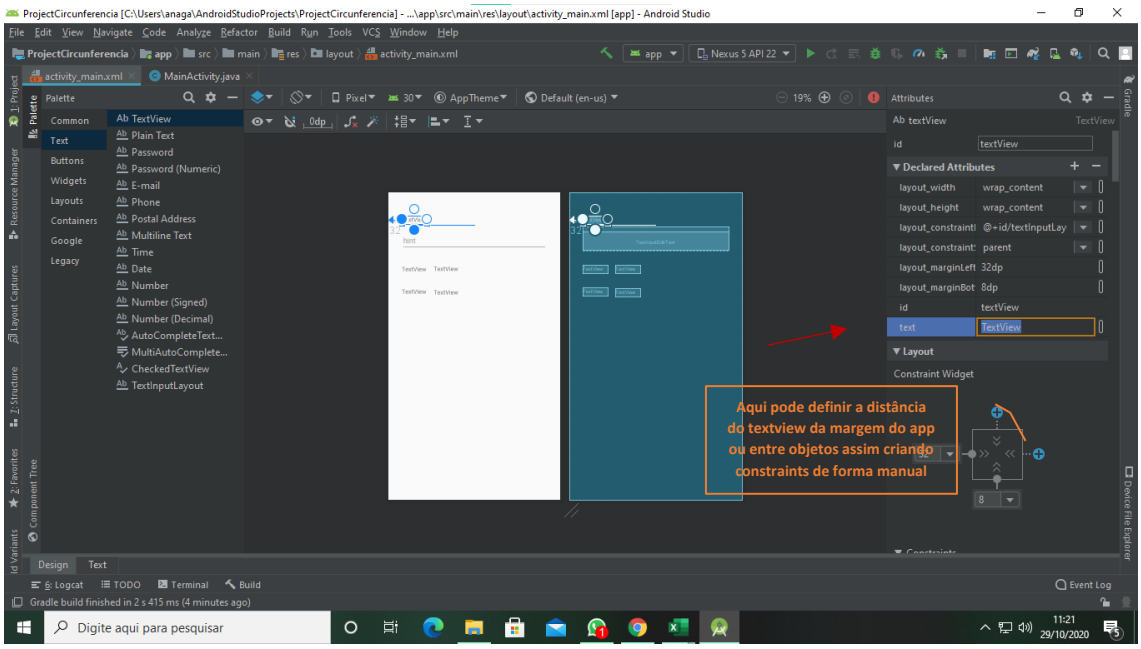
Observação: O “TextImputLayout” não vem no pacote, é preciso clicar no mesmo para que apareça a opção de baixá-lo.

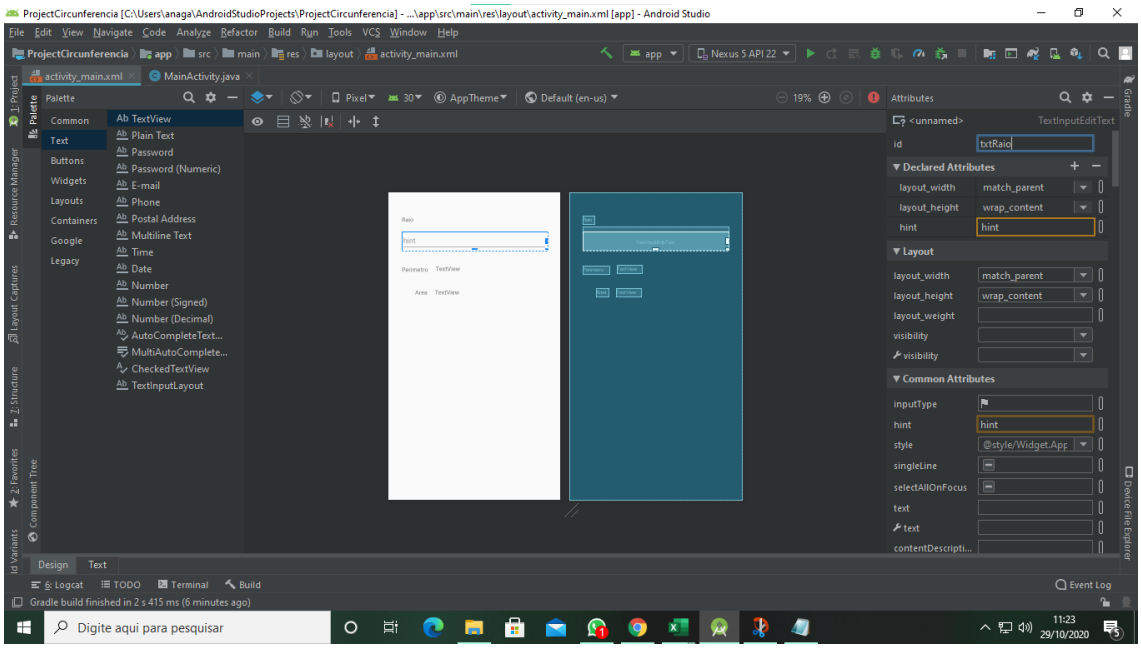
Na próxima tela adicionamos os TextViews em nosso projeto para indicar cada variável, além de exibir o valor do cálculo do perímetro e da área.



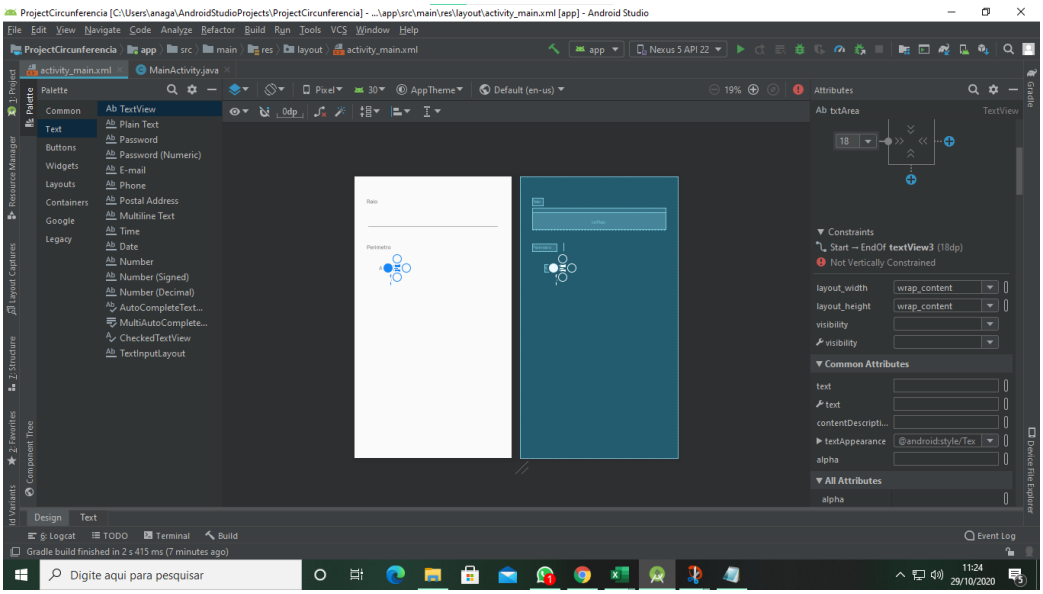
Quarto passo. A seta laranja indica a “left constraint”, se arrastar esta bolinha ao outro textview criará uma constraint, ou seja, um alinhamento entre os dois objetos.

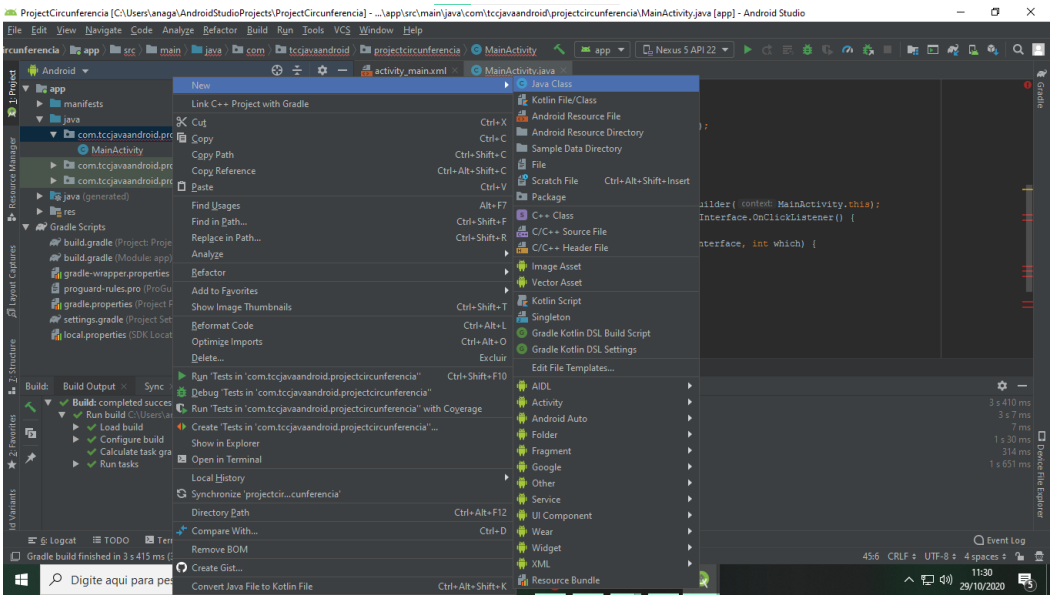
Em “Attributes” podemos alterar propriedades dos nossos objetos colocados na tela e em “text” pode-se alterar o valor que será exibido na tela, como a textview abaixo, na qual iremos atribuir o valor “Raio” para exibir na tela do nosso aplicativo.

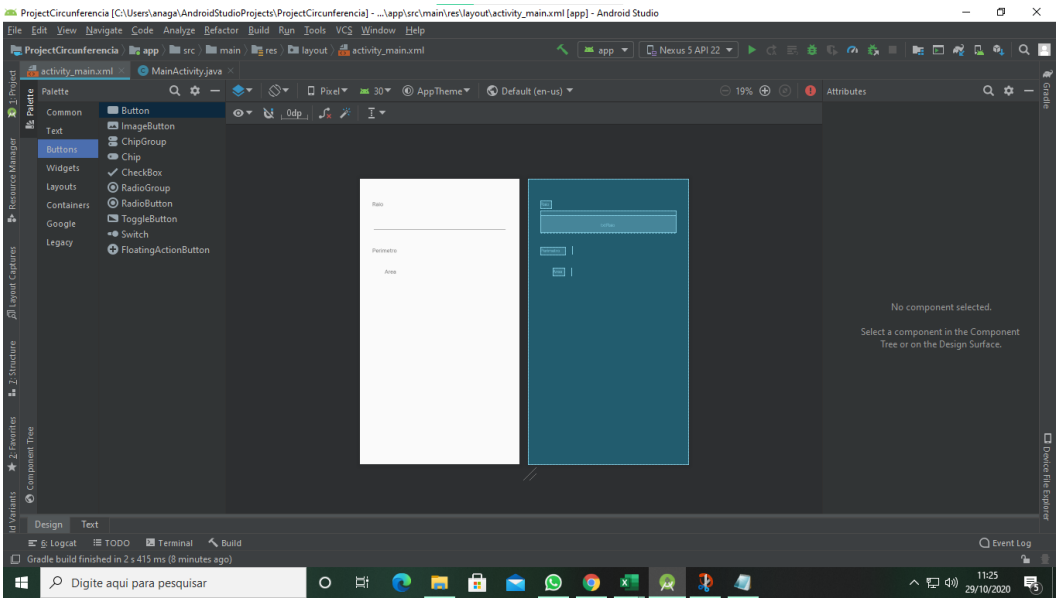


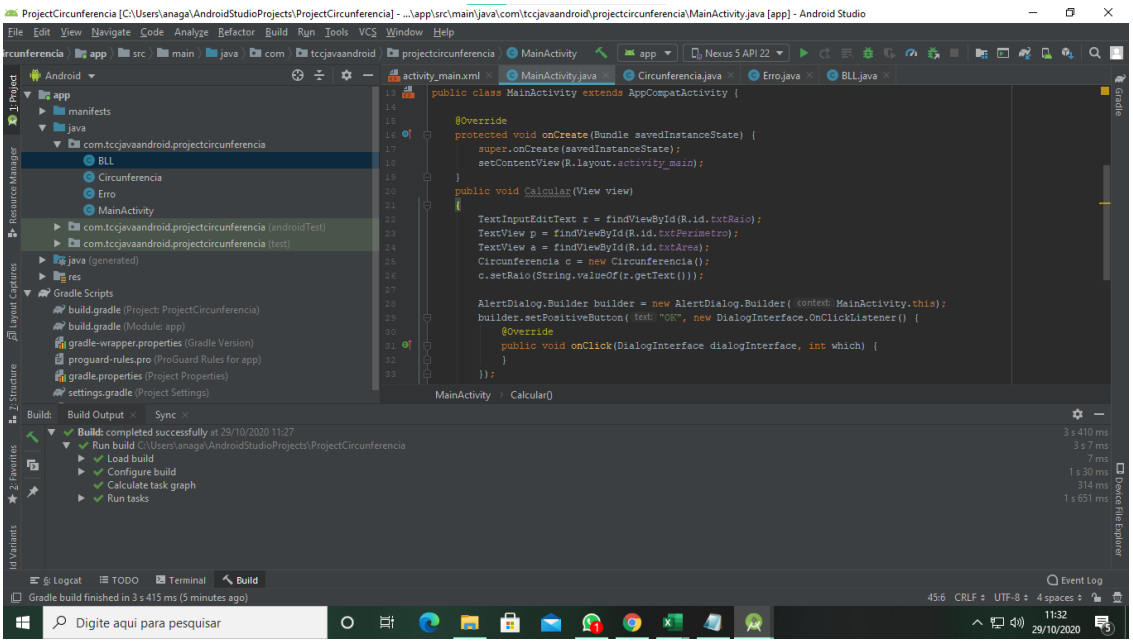


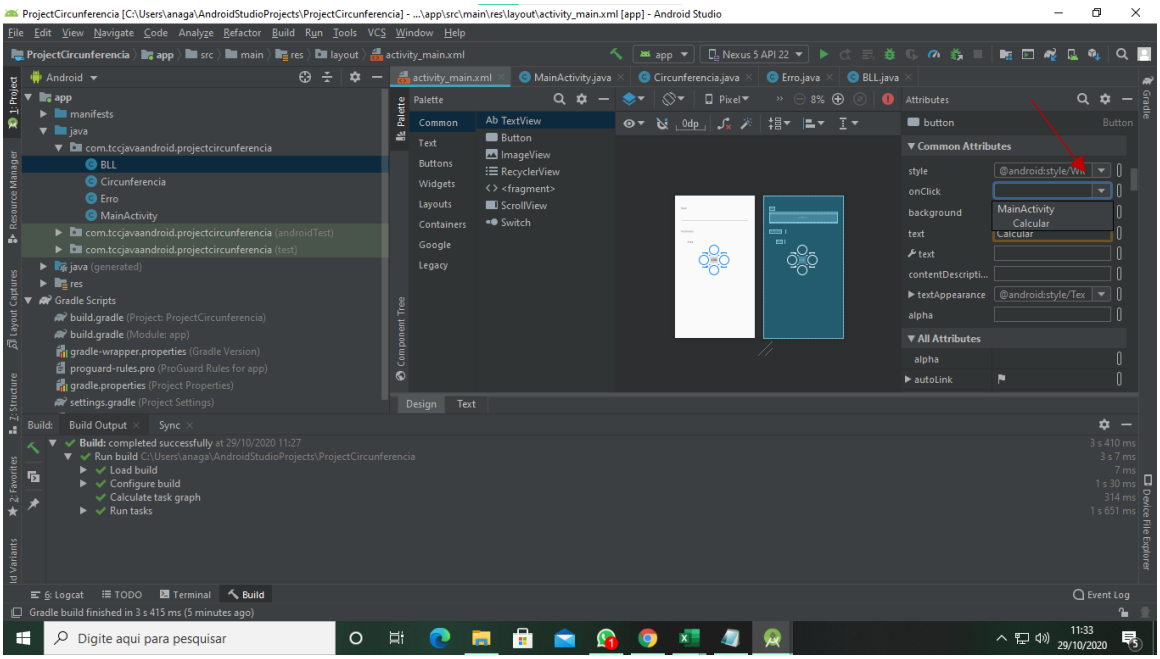
No quinto passo definimos o “id”, txtRaio, permitindo manipular a entrada e saída de dados através desses objetos, pois neste campo será possível digitar um número que será utilizado para exibir mais dois dados. E é através do id na linha de código “TextInputEditText r = findViewById(R.id.***txtRaio***);“ que será possível usar o valor digitado pelo usuário.



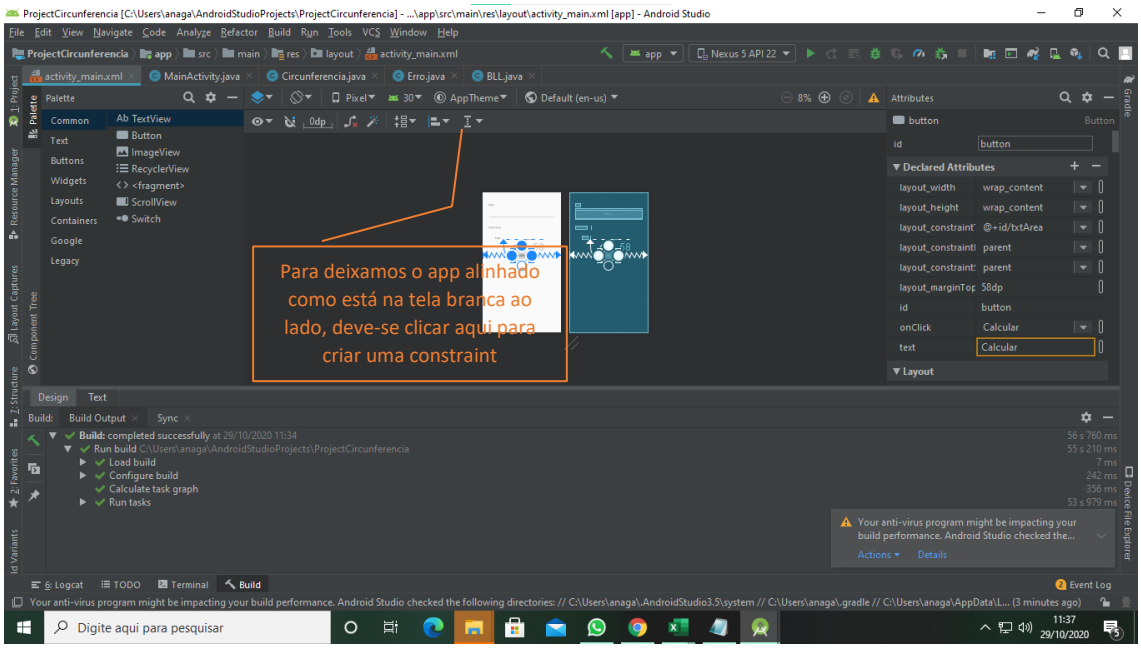



ProjectCircunferencia [C:\Users\anaga\AndroidStudioProjects\ProjectCircunferencia] - ...\app\src\main\res\layout\activity_main.xml [app] - Android Studio
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
activity_main.xml
MainActivity.java
Palette
Common
Text
Buttons
Widgets
Layouts
Containers
Google
Legacy
Button
ImageButton
ChipGroup
Chip
CheckBox
RadioGroup
RadioButton
ToggleButton
Switch
FloatingActionButton
Attributes
No component selected.
Select a component in the Component Tree or on the Design Surface.
Design
Text
Gradle build finished in 2 s 415 ms (8 minutes ago)
Digite aqui para pesquisar
11:25
29/10/2020


ProjectCircunferencia [C:\Users\anaga\AndroidStudioProjects\ProjectCircunferencia] - ...\app\src\main\java\com\tccjavaandroid\projectcircunferencia\MainActivity.java [app] - Android Studio
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
activity_main.xml
MainActivity.java
Circunferencia.java
Erro.java
BLL.java
Android
app
manifests
java
com.tccjavaandroid.projectcircunferencia
BLL
Circunferencia
Erro
MainActivity
Gradle Scripts
public class MainActivity extends AppCompatActivity {
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
}
public void Calcular(View view)
{
TextInputEditText r = findViewById(R.id.txtRaio);
TextView p = findViewById(R.id.txtPerimetro);
TextView a = findViewById(R.id.txtArea);
Circunferencia c = new Circunferencia();
c.setRaio(String.valueOf(r.getText()));
AlertDialog.Builder builder = new AlertDialog.Builder(context: MainActivity.this);
builder.setPositiveButton(text: "OK", new DialogInterface.OnClickListener() {
@Override
public void onClick(DialogInterface dialogInterface, int which) {
}
});
MainActivity › Calcular()
Build: completed successfully at 29/10/2020 11:27
Run build C:\Users\anaga\AndroidStudioProjects\ProjectCircunferencia
Load build
Configure build
Calculate task graph
Run tasks
Gradle build finished in 3 s 415 ms (5 minutes ago)
Digite aqui para pesquisar
11:32
29/10/2020

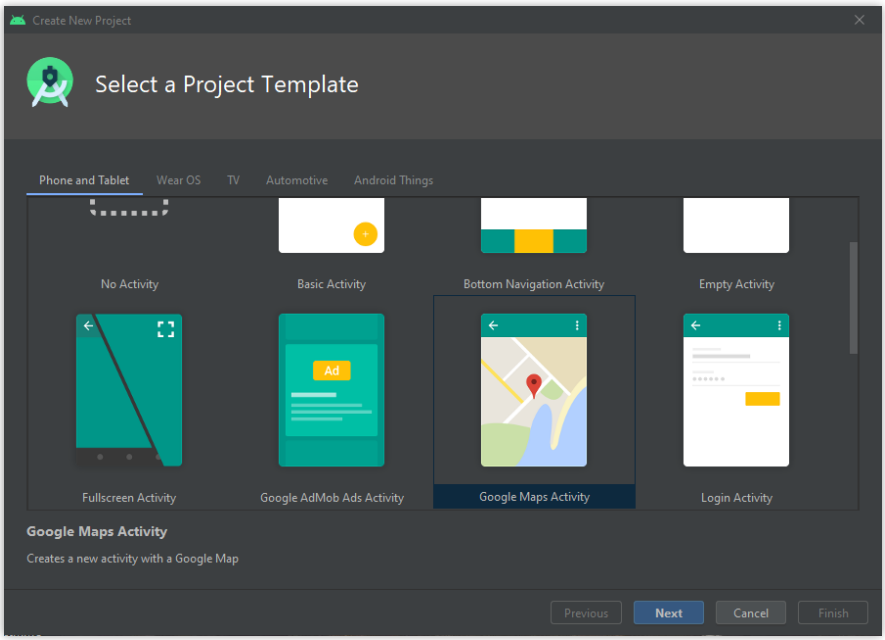


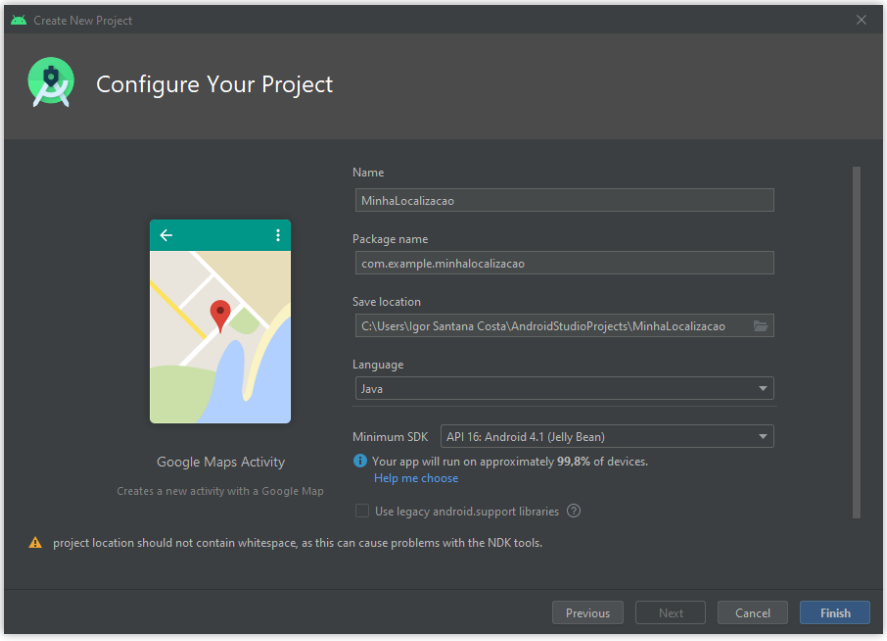
No sexto e último passo, em “Attributes” do botão iremos definir em “onClick” o método “calcular, criado na classe principal MainActivity.java, para assim definirmos a função do botão.



Aplicativo 02: Minha localização com Google Maps

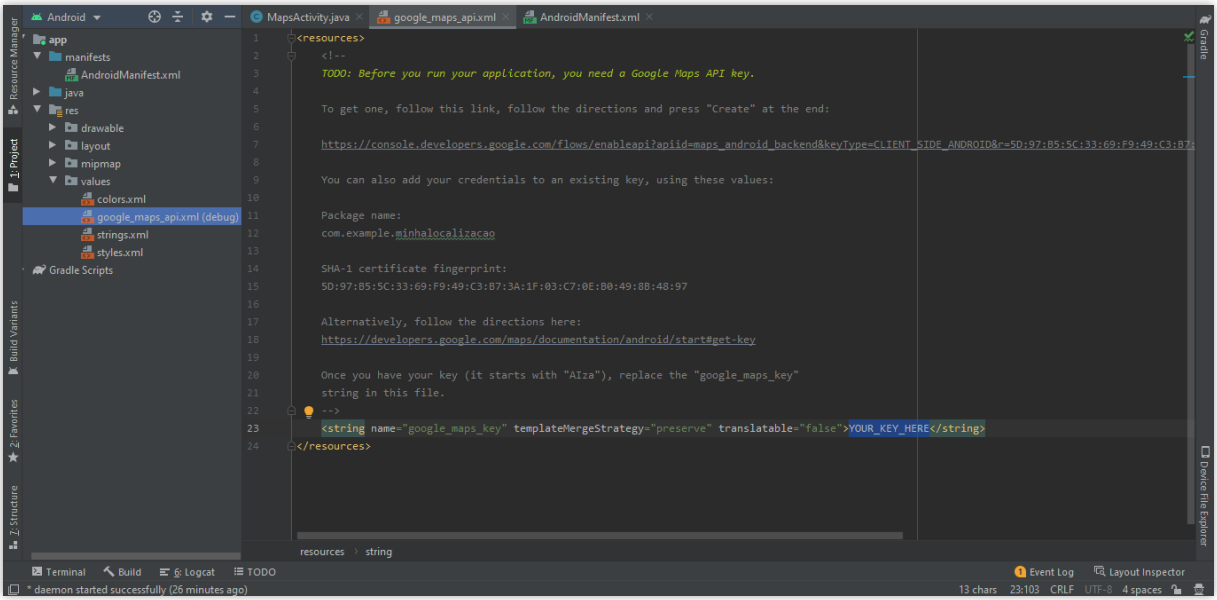
1. Crie um projeto com *Google Maps Activity*. Nele já estará implementado a *API* do *Google MAPS*





2. Antes de rodar a aplicação é necessário configurar a chave de *API* do *Google Maps*. Por isso, logo após a criação do projeto é aberto o “google_maps_api.xml”. Dentro desse arquivo terá um link personalizado, copie-o e cole-o em seu navegador. Em seguida acesse a sua conta Google e siga as instruções para criar

a sua chave de desenvolvedor. Por fim, copie o código e insira-o na *string* “google_maps_key”, substituindo completamente “YOUR_KEY_HERE”.



3. Para que o projeto funcione serão necessárias permissões para a localização geográfica do usuário e o uso da internet. Portanto, adicione as seguintes permissões em “AndroidManifest.xml”:

```
<uses-permission
android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.INTERNET" />
```

4. Com tudo devidamente configurado, enfim a *Activity* do *Google Maps* poderá ser modificada. Para tanto, abra o “MapsActivity”, que está dentro do pacote do seu programa – no caso do exemplo, “com.example.minhalocalizacao”.

5. Note que há um objeto *mMap* que é instância da classe *GoogleMap*. É ele quem possui os atributos do nosso mapa e será utilizado para configurar as nossas preferências.

6. No método *onMapReady()* está predefinido que logo após a aplicação iniciar será definido um novo marcador e a câmera moverá para a Austrália, com coordenadas em Sydney. Tais códigos não serão úteis para a nossa aplicação e podem ser removidos.

7. Para que seja disponibilizado o botão “My Location” (“Minha Localização”, em português), que direciona o usuário à sua localização no mapa, ele deverá ser definido como verdadeiro para o nosso mapa (no caso, o objeto *mMap*). Então podemos usar:

```
mMap.setMyLocationEnabled(true);
```

Observe que ocorrerá um erro, pois para ativar o botão “Minha Localização” é preciso ser garantido que o usuário tenha dado permissão para que seus dados geográficos sejam acessados pela aplicação. Então, com o cursor do mouse sobre a linha pressione *Alt* + *Shift* + *Enter* para que automaticamente seja gerada a estrutura de checagem, resultando em:

```
if (ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION) !=
PackageManager.PERMISSION_GRANTED &&
ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_COARSE_LOCATION) !=
PackageManager.PERMISSION_GRANTED) {
    // TODO: Consider calling
    //    ActivityCompat#requestPermissions
    // here to request the missing permissions, and then overriding
    //   public void onRequestPermissionsResult(int requestCode, String[]
permissions,
    //                                          int[] grantResults)
    // to handle the case where the user grants the permission. See the
documentation
    // for ActivityCompat#requestPermissions for more details.
    return;
}
mMap.setMyLocationEnabled(true);
```

O *if* verifica se o dispositivo **não** tem as permissões necessárias. Se for esse o caso, então a permissão deverá ser requisitada. Para isso, substituímos as linhas de comentário com:

```
ActivityCompat.requestPermissions(this, new String[] {
        Manifest.permission.ACCESS_FINE_LOCATION,
        Manifest.permission.ACCESS_COARSE_LOCATION}, 1);
```

ActivityCompat.requestPermissions() é o método que faz a requisição das permissões necessárias (*Manifest.permission.ACCESS_FINE_LOCATION* e *Manifest.permission.ACCES_COARSE_LOCATION*). Os seus argumentos são, respectivamente: atividade (*this*), permissões (citadas acima) e código de requisição (pode ser qualquer número inteiro, nesse programa foi escolhido o 1).

Porém, a estrutura ainda está incompleta. Perceba que "mMap.setMyLocationEnabled(true);" será executado mesmo se o usuário negar a permissão, gerando um erro. Assim, deve-se coloca-lo dentro de um *else*. Dessa forma, primeiro será checado se o dispositivo não tem permissão, se ele não tiver será requisitada; ou se ele já tiver (*else*) o botão "Minha Localização" será habilitado. Logo, o código será o seguinte:

```
if (ActivityCompat.checkSelfPermission(this,
        Manifest.permission.ACCESS_FINE_LOCATION) !=
PackageManager.PERMISSION_GRANTED
        && ActivityCompat.checkSelfPermission(this,
        Manifest.permission.ACCESS_COARSE_LOCATION) !=
PackageManager.PERMISSION_GRANTED) {

    ActivityCompat.requestPermissions(this, new String[] {
            Manifest.permission.ACCESS_FINE_LOCATION,
            Manifest.permission.ACCESS_COARSE_LOCATION}, 1);
} else {
mMap.setMyLocationEnabled(true);
}
```

8. Entretanto, mesmo que o usuário aceite a requisição, o botão será ativado apenas ao reabrir a aplicação. Desse modo, podemos utilizar o método *onRequestPermissionsResult()* para que o botão seja habilitado logo ao aceitar a permissão (digite o nome do método e pressione *Enter* ou *Tab* para cria-lo

automaticamente). Para tanto, basta copiar o *if* gerado automaticamente na etapa anterior e alterar as lógicas de comparação "!=" para "==". Se a sentença for verdadeira, então deve-se habilitar o botão, caso contrário pode-se exibir uma mensagem informando que a permissão foi negada:

```
public void onRequestPermissionsResult(int requestCode, @NonNull String[]
permissions, @NonNull int[] grantResults) {
    if (ContextCompat.checkSelfPermission(this,
            Manifest.permission.ACCESS_FINE_LOCATION) ==
PackageManager.PERMISSION_GRANTED) {
        mMap.setMyLocationEnabled(true);
    }
    else
        Toast.makeText(this, "Permissão negada!",
Toast.LENGTH_SHORT).show();
    return;
}
```

Onde:

- ***Toast.makeText()*** é o método da classe *Toast* usado para mostrar a mensagem. Ele recebe como parâmetros, respectivamente: contexto (*this*), o texto a ser exibido ("Permissão negada") e o tempo de exibição (apenas *Toast.LENGTH_SHORT* ou *Toast.LENGTH_LONG*);
- ***.show()*** é o método que faz a janela de texto da classe *Toast* ser exibida.

9. Para que as coordenadas do usuário sejam exibidas a ele quando clicar em "Minha Localização", precisamos de um método que é chamado quando tal evento ocorrer. Dessa maneira, podemos implementar na classe *MapsActivity* o *listener* "GoogleMap.OnMyLocationButtonClickListener", ficando:

```
public class MapsActivity extends FragmentActivity implements
        OnMapReadyCallback,
        GoogleMap.OnMyLocationButtonClickListener
{
```

10. Em seguida pressione *Ctrl+i* para implementar o método *onMyLocationButtonClick()*. Note que ele é booleano e retorna
    a. ***false*** se o comportamento padrão deve ocorrer (mover a câmera de forma centralizada no local do usuário); ou
    b. ***true*** se o comportamento padrão não deve ser realizado.

    Não esqueça de inserir "mMap.setOnMyLocationButtonClickListener(this);" no método *onMapReady()* para habilitar o *listener* no mapa.

11. Por fim, vamos programar para que ao clicar no botão "Minha Localização" apareça a latitude e longitude do local do usuário dentro de uma janela de mensagem da classe *Toast*:

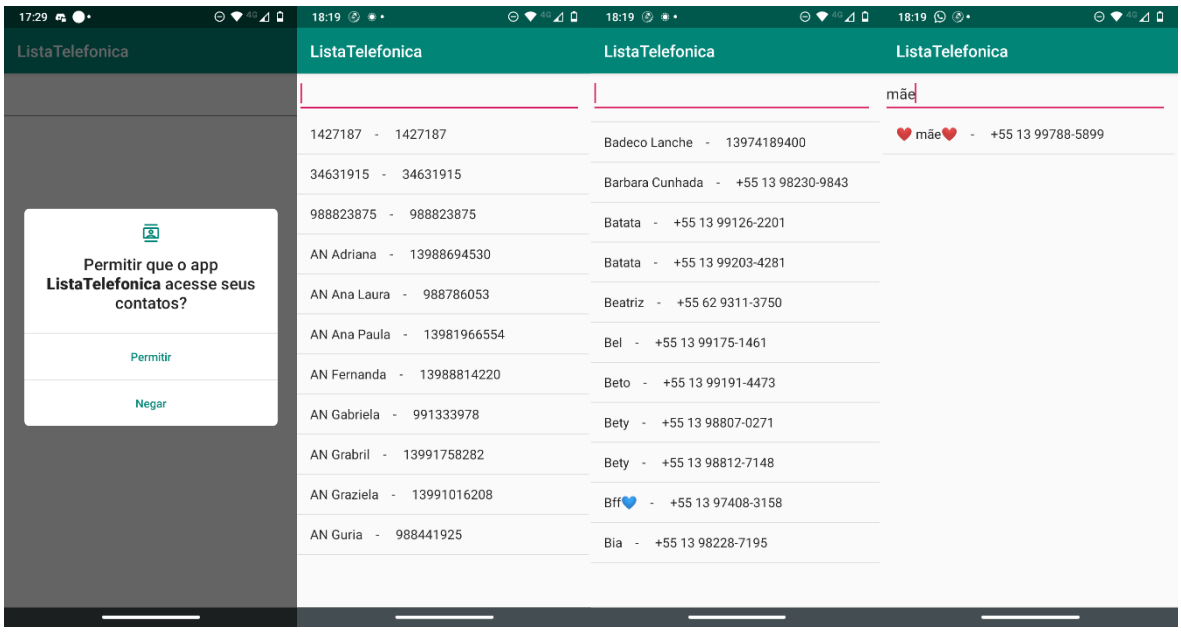
```
public boolean onMyLocationButtonClick() {
    String lat, lng, info;

    mMap.clear();

    lat = String.valueOf(mMap.getMyLocation().getLatitude());
    lng = String.valueOf(mMap.getMyLocation().getLongitude());
    info = "Latitude: " + lat + "\nLongitude: " + lng;

    Toast.makeText(this, info, Toast.LENGTH_LONG).show();
    MarkerOptions marker = new MarkerOptions();

    marker.position(new LatLng(Double.parseDouble(lat),
Double.parseDouble(lng)))
            .title("Você está aqui!");

    mMap.addMarker(marker);

    return false;
}
```

Onde:

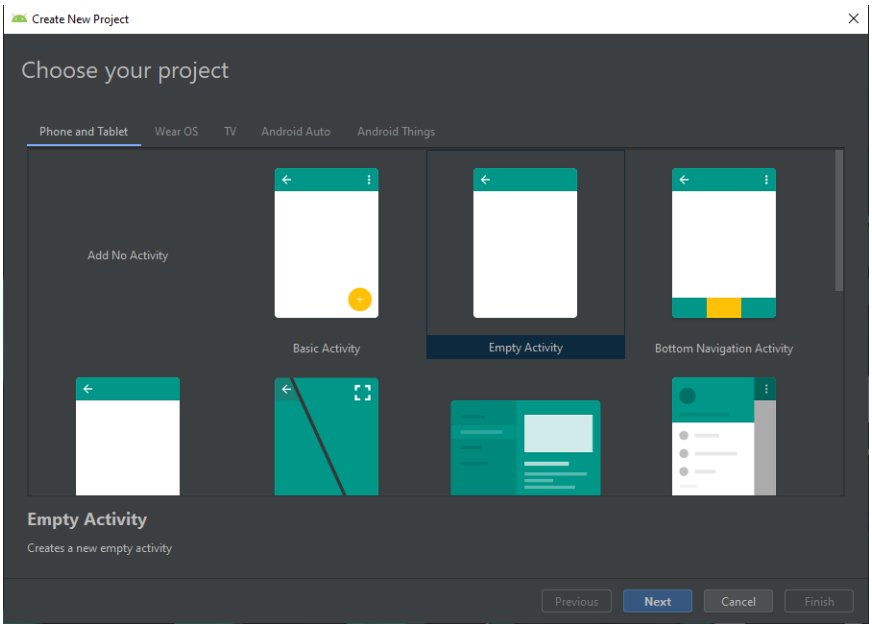
- ***lat*** é a *String* que recebe o valor da latitude do usuário, o qual é recuperado pelo método *mMap.getMyLocation().getLatitude()* e convertido de *double* para *String*;
- ***lng*** é a *String* que contém o valor da longitude do usuário, o qual é recuperado pelo método *mMap.getMyLocation().getLongitude()* e convertido de *double* para *String*;
- ***info*** é a *String* que recebe o valor das *Strings* acima;
- ***mMap.clear()*** é o método utilizado para remover todos os marcadores do mapa, evitando assim que fiquem registrados diversos deles;
- ***Toast.makeText()*** recebe como parâmetros, respectivamente: contexto (*this*), o texto a ser exibido (*info*) e o tempo de exibição (Toast.*LENGTH_LONG*);
- ***.show()*** é o método que faz a janela de texto da classe *Toast* ser exibida;
- ***marker***, instância de *MarkerOptions*, é o objeto que carrega os atributos de um marcador a ser adicionado no mapa;
- ***marker.position()*** é o método que define a coordenada de um marcador. Ele recebe uma instância da classe *LatLng*, que no caso possui a latitude *lat* e longitude *lng*, ambas convertidas de String para double;
- ***marker.title()*** é o método que determina o título a ser exibido quando o usuário clicar no marcador;
- ***mMap.addMarker()*** é o método que adiciona um marcador no mapa – no caso, é adicionado um de acordo com as opções configuradas para o objeto *marker*;
- é retornado ***false*** para que o comportamento padrão do botão seja executado.

Aplicativo 03: Lista Telefônica

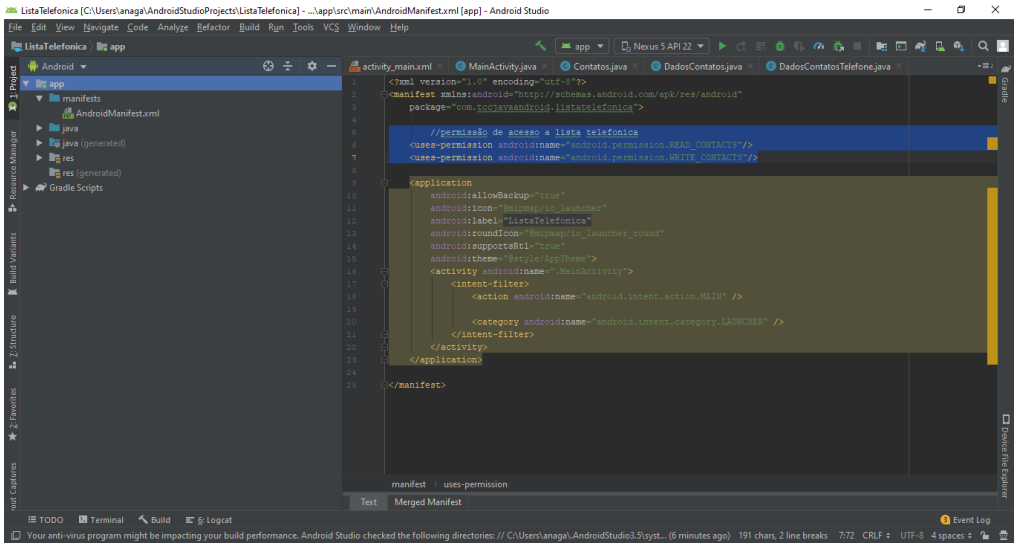
No seguinte projeto é trabalhado o acesso da lista telefônica do celular e filtragem de dados. A ideia do projeto é exibir a lista telefônica do telefone, exibindo nome e número, e permitir a filtragem dos contatos da lista.



1 Crie um projeto com modelo em branco no *Android Studio*.



2 Para poder ler os contatos precisamos de permissão do Sistema Operacional e para isso devesse acessar o arquivo AndroidManifest.xml e adicionar a permissão android.permission.READ_CONTACTS e android.permission.WRITE_CONTACT, como a imagem abaixo.



3 Iremos utilizar um EditText para o usuário entrar com os dados que deseja filtrar e um ListView para exibir os contatos da lista telefônica.

4 Criaremos as entidades Dados Contatos e Telefone, que irá armazenar as informações como id, nome e Telefone.

```
package com.tccjavaandroid.listatelefonica;
import java.util.List;

public class DadosContatos {
    private String ID;
    private String Nome;
    private List Telefones;

    public String getID() {return ID;}

    public void setID(String string) {
        ID = string;
    }

    public String getNome() {
```

```
        return Nome;
    }

    public void setNome(String nome) {
        Nome = nome;
    }

    public List getTelefones() {
        return Telefones;
    }

    public void setTelefones(List telefones) {
        Telefones = telefones;
    }

    //Metodo sobreescrito para que não aparece o nome do componente
    //na listView
    @Override
    public String toString() {

        return Nome + "    -     " + Telefones.get(0);
    }
}
```

```
package com.tccjavaandroid.listatelefonica;

public class DadosContatosTelefone {
    private String Telefone;

    public String getTelefone() {
        return Telefone;

    }
```

```
    public void setTelefone(String telefone) {
        Telefone = telefone;
    }
    @Override
    public String toString() {

        return "" + Telefone;
    }
}
```

5 Em seguida criaremos a classe Contatos que fará o preenchimento das entidades criadas anteriormente.

```
package com.tccjavaandroid.listatelefonica;

import android.content.Context;
import android.database.Cursor;
import android.provider.ContactsContract;
import android.widget.EditText;

import java.util.ArrayList;
import java.util.List;

public class Contatos {

    private Context ctx;

    public Contatos(Context contexto)
    {
        this.ctx = contexto;
    }
```

```
    public List getContatos(String pesquisa)
    {
        Cursor C_Contatos
                = null;
        if (android.os.Build.VERSION.SDK_INT >=
android.os.Build.VERSION_CODES.ECLAIR) {
            C_Contatos =
this.ctx.getContentResolver().query(ContactsContract.Contacts.CONTENT_URI,
                    null, null, null, ContactsContract.Contacts.DISPLAY_NAME);
        }
        //pega os index das colunnas
        int IndexID = C_Contatos.getColumnIndex(ContactsContract.Contacts._ID);
        int IndexTemTelefone =

C_Contatos.getColumnIndex(ContactsContract.Contacts.HAS_PHONE_NUMBER);

        int IndexName =

C_Contatos.getColumnIndex(ContactsContract.Contacts.DISPLAY_NAME);

        List Contatos = new ArrayList();
        DadosContatos Contato;
        while(C_Contatos.moveToNext())
        {
            Contato = new DadosContatos();
            Contato.setID(C_Contatos.getString(IndexID));
            Contato.setNome(C_Contatos.getString(IndexName));
            //verifica se o contato tem telefone
            if(Integer.parseInt(C_Contatos.getString(IndexTemTelefone))>0)
            {
                Telefone _Telefone = new Telefone(Contato.getID(), this.ctx);
                Contato.setTelefones(_Telefone.getTelefones());
            }
```

```
        if (Contato.getNome().contains(pesquisa))
            Contatos.add(Contato);
    }
    C_Contatos.close();
    return Contatos;
  }
}
```

6 Caso contato possua telefone, então será preenchido na a classe Telefone mostrada abaixo.

```
7  package com.tccjavaandroid.listatelefonica;
   import android.content.Context;
   import android.database.Cursor;
   import android.provider.ContactsContract;
   import java.util.ArrayList;
   import java.util.List;

   public class Telefone {

      private String _IDContato;
      private Context _ctx;
      public Telefone(String IDContato, Context Contexto)
      {
         this._IDContato = IDContato;
         this._ctx = Contexto;
      }
      public List getTelefones()
      {
         Cursor C_Telefones
   =this._ctx.getContentResolver().query(ContactsContract.CommonDataKinds.P
   hone.CONTENT_URI, null,
   ContactsContract.CommonDataKinds.Phone.CONTACT_ID + " = " +
   _IDContato, null, null);
```

```
            int IndexTelefone;
            List Telefones = new ArrayList();
            while(C_Telefones.moveToNext())
            {
                DadosContatosTelefone Telefone = new DadosContatosTelefone();
                IndexTelefone =
        C_Telefones.getColumnIndex(ContactsContract.CommonDataKinds.Phone.N
        UMBER);
                Telefone.setTelefone(C_Telefones.getString(IndexTelefone));
                Telefones.add(Telefone);
            }
            C_Telefones.close();
            return Telefones;
        }
    }
```

7. Para ler os contatos da lista telefônica e escrever em uma List view utiliza se o código abaixo.

```
 if (ActivityCompat.checkSelfPermission(this,
       Manifest.permission.READ_CONTACTS) !=
PackageManager.PERMISSION_GRANTED &&
ActivityCompat.checkSelfPermission(this,
Manifest.permission.WRITE_CONTACTS) !=
PackageManager.PERMISSION_GRANTED) {
   ActivityCompat.requestPermissions(this, new String[]{
          Manifest.permission.WRITE_CONTACTS,
          Manifest.permission.READ_CONTACTS}, 1);
   return;
}
final ListView listaPessoas = (ListView) findViewById(R.id.ListaContatosTelefonicos);

final List[] ListaContatos = {new ArrayList()};
```

```
final Contatos Contato = new Contatos(this);
```

- o if é para checar se a permissão de acesso aos contatos do celular foi dada, caso seja verdadeiro, ira ler e escrever os contatos na listView.
- **PackageManager.PERMISSION_GRANTED**, indica que a permissão de acesso foi dada.

8. O código abaixo tem como função filtrar os dados conforme o que o usuário escrever no EditTextP.

```
((EditText)findViewById(R.id.EditTextP)).setOnEditorActionListener(
        new EditText.OnEditorActionListener() {
            @Override
            public boolean onEditorAction(TextView v, int actionId, KeyEvent event) {
                if (actionId == EditorInfo.IME_ACTION_SEARCH ||
                        actionId == EditorInfo.IME_ACTION_DONE ||
                        event != null &&
                                event.getAction() == KeyEvent.ACTION_DOWN &&
                                event.getKeyCode() == KeyEvent.KEYCODE_ENTER) {
                    if (event == null || !event.isShiftPressed()) {
                        ListaContatos[0] =
Contato.getContatos(EditTextP.getText().toString());

                        // adapter que sera o source para a listview
                        ArrayAdapter adapter = new ArrayAdapter(MainActivity.this,
                                android.R.layout.simple_list_item_1, ListaContatos[0]);

                        //seta o adapter para o listview
                        listaPessoas.setAdapter(adapter);

                        return true; // consumir.
                    }
                }
```

```
            return false; // passar para outros listeners
        }
    }
);
```

O código completo da Main_Activity

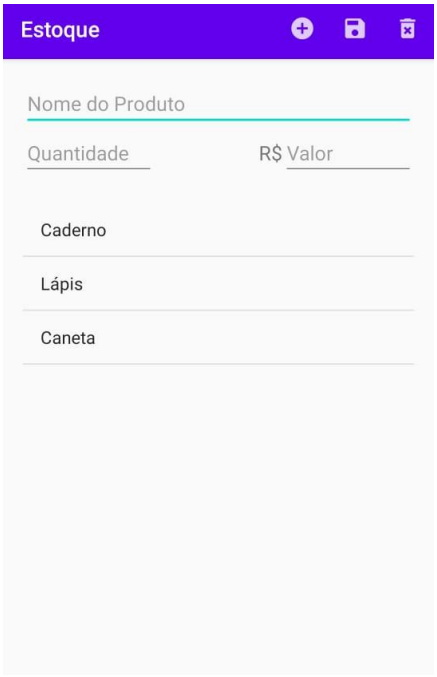
```
package com.tccjavaandroid.listatelefonica;

import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import androidx.core.app.ActivityCompat;
import android.Manifest;
import android.annotation.SuppressLint;
import android.content.pm.PackageManager;
import android.view.KeyEvent;
import android.view.inputmethod.EditorInfo;
import android.widget.ArrayAdapter;
import android.widget.EditText;
import android.widget.ListView;
import android.widget.TextView;

import java.security.spec.ECParameterSpec;
import java.util.ArrayList;
import java.util.List;

public class MainActivity extends AppCompatActivity {

    EditText EditTextP;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
```

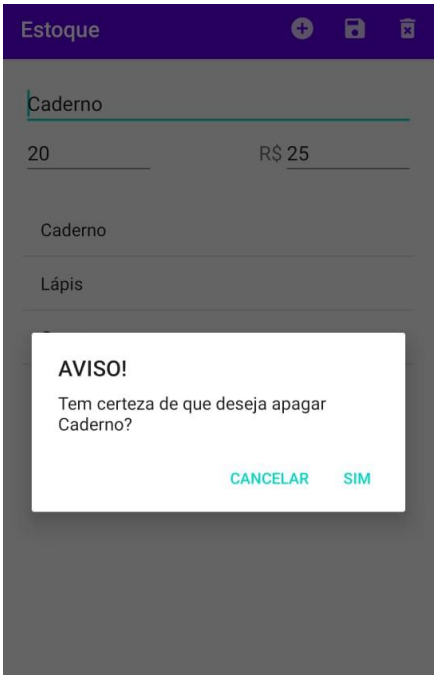
```
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        EditTextP = findViewById(R.id.EditTextP);

        if (ActivityCompat.checkSelfPermission(this,
                Manifest.permission.READ_CONTACTS) !=
PackageManager.PERMISSION_GRANTED &&
ActivityCompat.checkSelfPermission(this,
Manifest.permission.WRITE_CONTACTS) !=
PackageManager.PERMISSION_GRANTED) {
            ActivityCompat.requestPermissions(this, new String[]{
                    Manifest.permission.WRITE_CONTACTS,
                    Manifest.permission.READ_CONTACTS}, 1);
            return;
        }
        final ListView listaPessoas = (ListView)
findViewById(R.id.ListaContatosTelefonicos);

        final List[] ListaContatos = {new ArrayList()};

        final Contatos Contato = new Contatos(this);

        ((EditText)findViewById(R.id.EditTextP)).setOnEditorActionListener(
                new EditText.OnEditorActionListener() {
                    @Override
                    public boolean onEditorAction(TextView v, int actionId, KeyEvent event)
{
                        if (actionId == EditorInfo.IME_ACTION_SEARCH ||
                                actionId == EditorInfo.IME_ACTION_DONE ||
                                event != null &&
                                        event.getAction() == KeyEvent.ACTION_DOWN &&
                                        event.getKeyCode() == KeyEvent.KEYCODE_ENTER) {
```

```
                    if (event == null || !event.isShiftPressed()) {
                        ListaContatos[0] =
Contato.getContatos(EditTextP.getText().toString());

                        // adapter que sera o source para a listview
                        ArrayAdapter adapter = new ArrayAdapter(MainActivity.this,
                                android.R.layout.simple_list_item_1, ListaContatos[0]);

                        //seta o adapter para o listview
                        listaPessoas.setAdapter(adapter);

                        return true; // consumir.
                    }
                }
                return false; // passar para outros listeners
            }
        }
    );}}
```

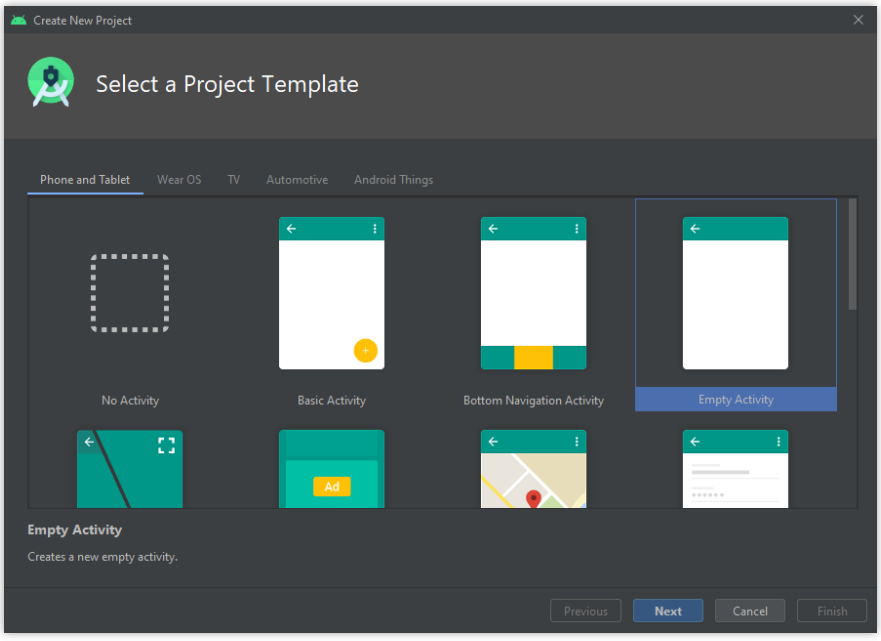
Aplicativo 04: CRUD com Firebase

No seguinte projeto é trabalhado operações *CRUD* (*Create, Read, Update* e *Delete*) num banco de dados do *Google Firebase*. A ideia da base de dados consiste em armazenar informações acerca de mercadorias em um estoque. Para tanto, cada mercadoria terá atributos como identificação, nome do produto, quantidade e valor.





1- Crie um novo projeto com modelo em branco no *Android Studio*.



2- Acesse o site do *Firebase* e entre em uma conta *Google*. Clique em “Ir para o console” e crie um novo projeto. Nessa aplicação não será necessário ativar o recurso do *Google Analytics*.

3- Para adicionar o *Firebase* ao aplicativo, clique no ícone do Android , insira o nome do pacote e, opcionalmente, o nome da aplicação e o *hash SHA-1* do seu certificado de assinatura – para consegui-lo execute o comando *gradlew signingReport* no terminal da IDE.

4- Faça o *download* do arquivo de configuração *.json* e siga as instruções para implementá-lo ao seu projeto:

5- Adicione o *SDK* do *Firebase*. Para isso, insira as seguintes linhas de código em:

- *build.gradle* no nível do projeto (*<project>/build.gradle*):

```
buildscript {
    repositories {
        // Verifique se tem essa linha (se não, adicione-a):
        google()  // Repositório Maven do Google
    }
    dependencies {
```

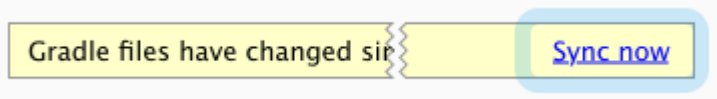
```
        ...
        // Insira esta linha
        classpath 'com.google.gms:google-services:4.3.4'
    }
}

allprojects {
    ...
    repositories {
        // Verifique se tem essa linha (se não, adicione-a):
        google()  // Repositório Maven do Google
        ...
    }
}
```

- *build.gradle* no nível do *app* (*<project>/<app-module>/build.gradle*):

```
apply plugin: 'com.android.application'
// Insira esta linha
apply plugin: 'com.google.gms.google-services'

dependencies {
    // Importe o Firebase BoM
    implementation platform('com.google.firebase:firebase-bom:26.2.0')

    // Declare a dependência da biblioteca do banco de dados do Firebase
    implementation 'com.google.firebase:firebase-database:19.6.0'
}
```

Note que também foi implementada a dependência do *Firebase-database*, uma vez que tal recurso será usado na aplicação. A referência foi retirada do repositório do *Google Firebase*.

6- Pressione "Sincronizar agora" na barra que aparece na IDE:



7- Voltando à página do *Firebase* em seu navegador, acesse “Realtime Database” no menu à esquerda de título “Criação”. Clique em “Criar banco de dados”, selecione o local mais próximo e inicie no modo de teste.

8- Para finalizar a configuração do projeto será necessário implementar uma permissão de uso da internet, possibilitando a alteração no banco de dados. Portanto, adicione-a em “AndroidManifest.xml”:

```
<uses-permission android:name="android.permission.INTERNET" />
```

9- Partindo para o design da aplicação, serão usadas três caixas de textos com *hints* dos nomes dos atributos da mercadoria (nome do produto, quantidade e valor) a ser manipulada. Também conterá uma *List View* para visualizar a lista de produtos registrados. Para facilitar, será utilizado um *layout* linear de orientação vertical.

- Começando pela alteração do *layout*, vá na *activity_main.xml* (*app* > *res* > *layout*) e substitua “androidx.constraintlayout.widget.ConstraintLayout” por “LinearLayout”;
- Insira, dentro da *tag* do *LinearLayout*, “android:orientation=’vertical’”;
- Apague o *TextView* “Hello World!”;
- Implemente um objeto *Plain Text* para o nome do produto e um *Number* para quantidade e o último *Number (decimal)* para o atributo valor. Para ficar mais agradável, os campos quantidade e valor podem ficar dentro de um *Linear layout* horizontal e pode ser inserido um *Text View* com texto de “R$” à frente do campo do valor;
- Por fim, adicione um *List View*.

O design deve ficar algo como:

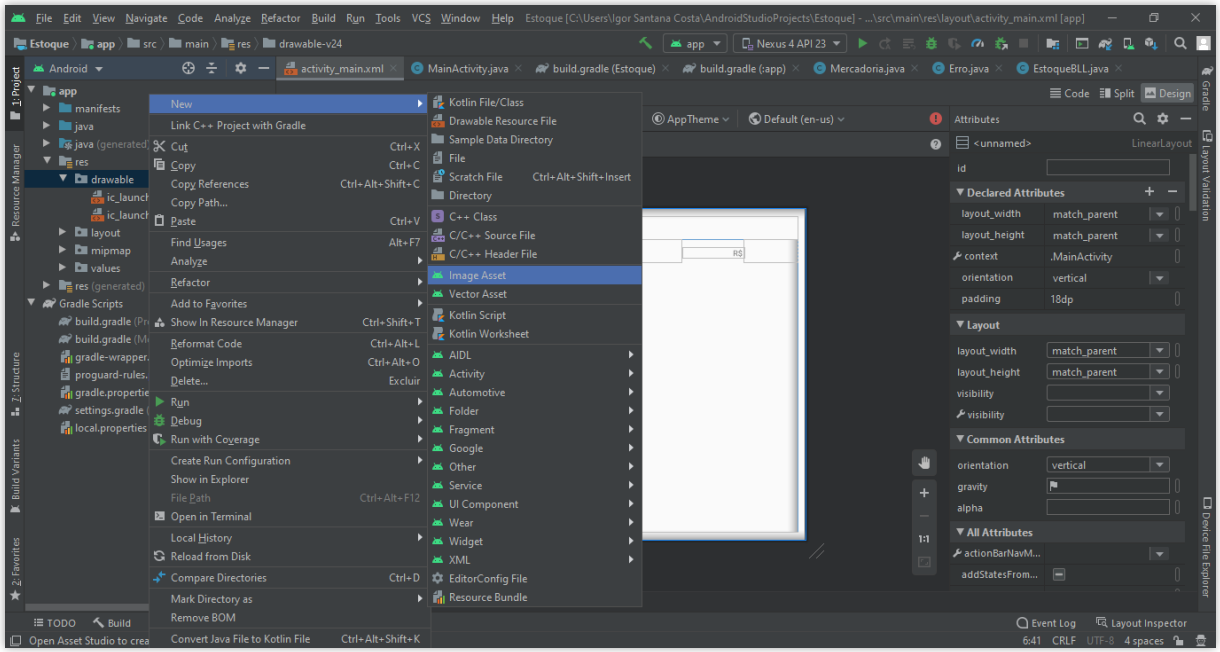
```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
```

```
android:layout_height="match_parent"
android:orientation="vertical"
android:padding="18dp"
tools:context=".MainActivity">

<EditText
    android:id="@+id/editProduto"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:ems="10"
    android:hint="Nome do Produto"
    android:inputType="textPersonName" />

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="68dp"
    android:orientation="horizontal">

    <EditText
        android:id="@+id/editQtd"
        android:layout_width="120dp"
        android:layout_height="wrap_content"
        android:ems="10"
        android:hint="Quantidade"
        android:inputType="number" />

    <TextView
        android:id="@+id/txtCambio"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:gravity="right"
        android:text="R$"
```

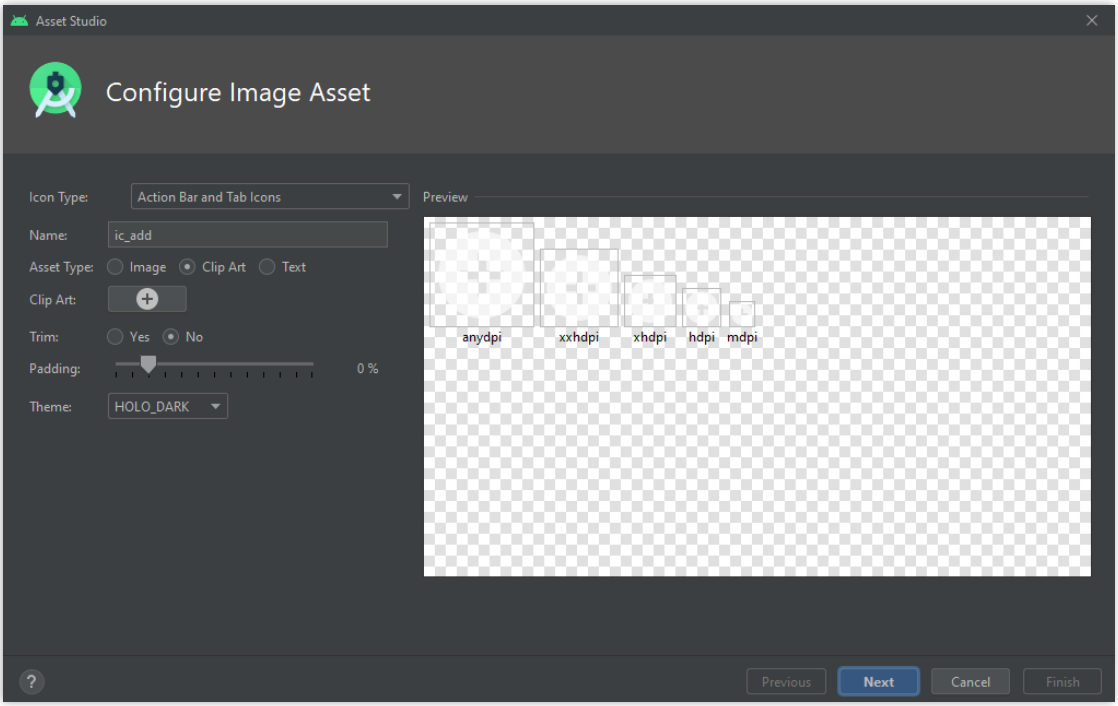
```
        android:textSize="18sp" />

    <EditText
        android:id="@+id/editValor"
        android:layout_width="120dp"
        android:layout_height="wrap_content"
        android:ems="10"
        android:hint="Valor"
        android:inputType="numberDecimal" />
  </LinearLayout>

  <ListView
    android:id="@+id/listV_produtos"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />
</LinearLayout>
```

10- Concluindo a parte de design, será feito um menu com botões para registrar novos produtos, atualizar dados de mercadorias já existentes e deletar produtos.

- Primeiro serão selecionados os ícones dos botões:



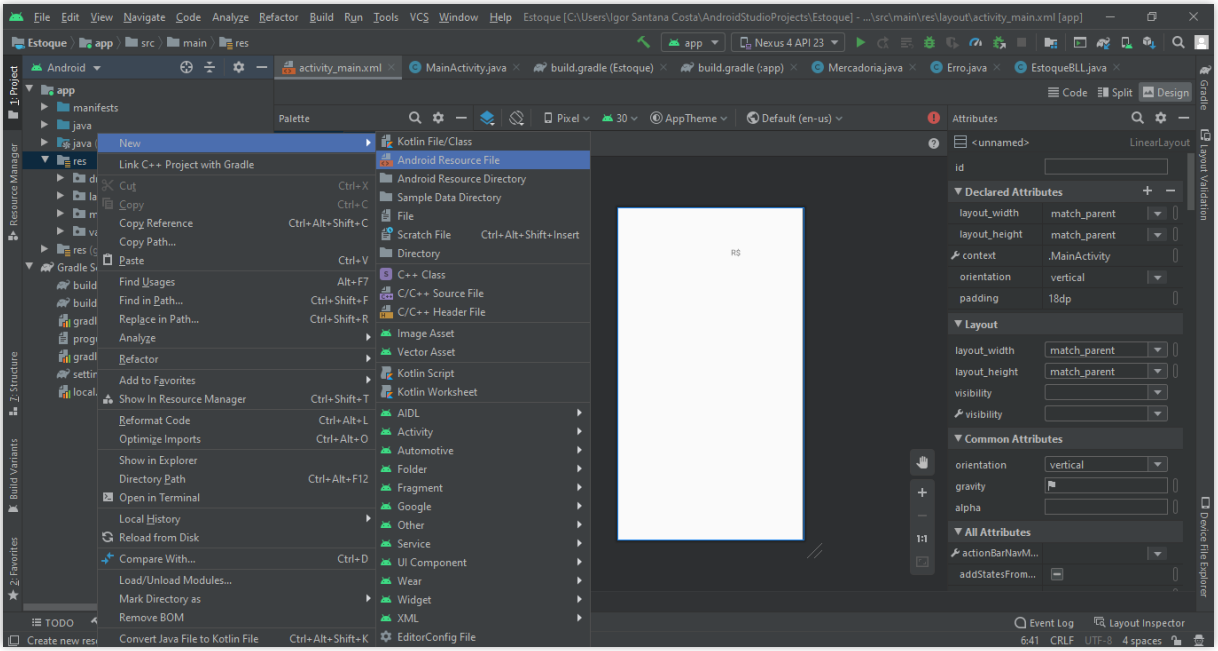
Dentro de *res*, clique com o botão direito do *mouse* no diretório *drawable*, vá em *new* e selecione *image asset*. Em seguida altere o tipo de ícone para “Action Bar and Tab Icons” e escolha o ícone em *Clip Art*. Não esqueça de modificar o nome. Tal processo será repetido mais duas vezes para os outros botões.



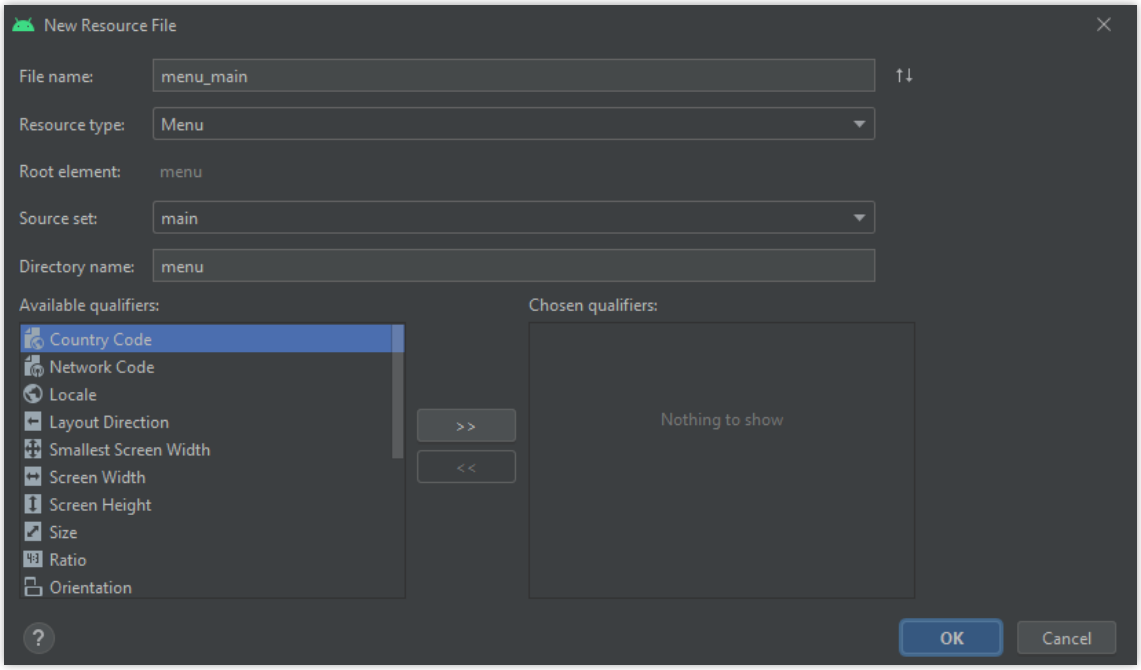
No exemplo serão definidos:

Nome	Clip Art	Tema
ic_add	add circle	HOLO_DARK
Ic_save	save	HOLO_DARK
ic_delete	delete forever	HOLO_DARK

Em seguida será criado o menu:



Clique com o botão direito do *mouse* no diretório *res*, selecione *new* e vá em *Android Resource File*. Depois altere o *Resource type* para “menu”, nomeie-o e clique em *OK*.

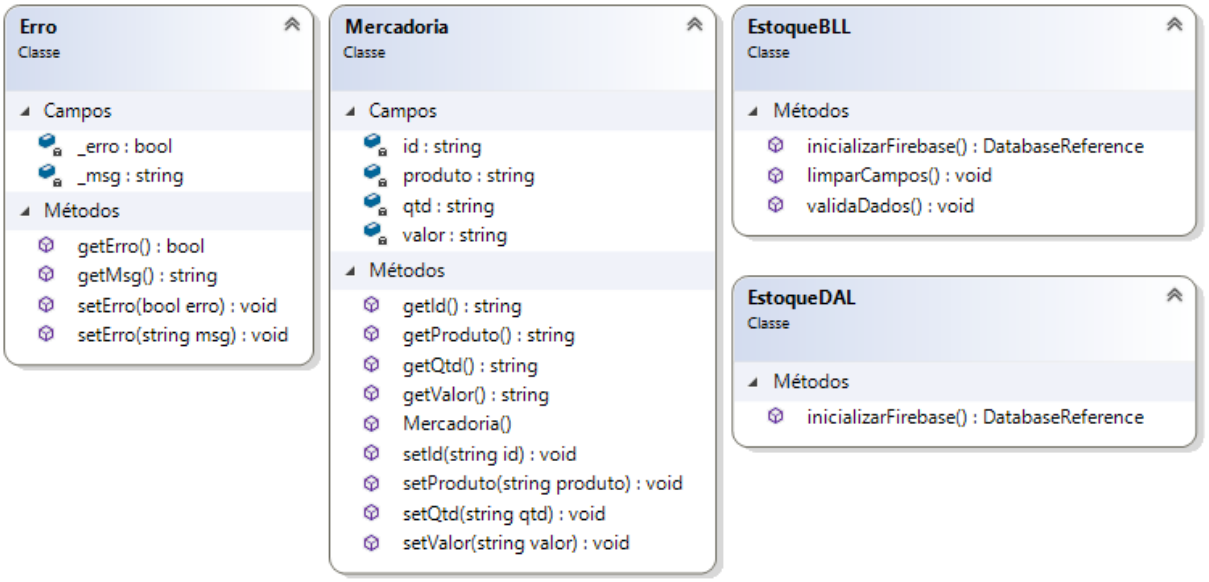


Agora serão implementados os itens (botões) no menu. Para tanto, basta inserir na aba design três objetos de itens de menu e configurar seus atributos para aparecerem como ícones. O *menu_main.xml* deverá ficar parecido com:

```
<menu xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:android="http://schemas.android.com/apk/res/android">

    <item
        android:id="@+id/menu_add"
        android:icon="@drawable/ic_add"
        android:title="Adicionar"
        app:showAsAction="always" />
    <item
        android:id="@+id/menu_save"
        android:icon="@drawable/ic_save"
        android:title="Salvar alterações"
        app:showAsAction="always" />
    <item
        android:id="@+id/menu_delete"
        android:icon="@drawable/ic_delete"
        android:title="Deletar"
```

```
        app:showAsAction="ifRoom" />
</menu>
```

11- As classes abaixo serão implementadas no projeto:



Lembre-se: construtores, *getters* e *setters* podem ser gerados automaticamente ao pressionar *alt + insert*.

12- A classe *EstoqueDAL* conterá apenas um método para fazer a conexão com o banco de dados *Firebase*:

```
public static DatabaseReference inicializarFirebase(Context context,
FirebaseDatabase firebaseDatabase) {
    FirebaseApp.initializeApp(context);
    firebaseDatabase = FirebaseDatabase.getInstance();
    firebaseDatabase.setPersistenceEnabled(true);

    return firebaseDatabase.getReference();
}
```

Onde:

- ***FirebaseApp*** é o ponto de entrada dos *SDKs* do *Firebase*. Assim, ao usar o *initializeApp()* é inicializada a instância *FirebaseApp* padrão usando valores de recurso de *string* – preenchidos em *google-services.json*;

- ***FirebaseDatabase.getInstance()*** recupera uma instância do banco de dados usando getInstance() e fazendo uma referência ao local onde você quer gravar.
- ***setPersistenceEnabled()*** é usado para aplicar alterações feitas *offline* quando o dispositivo for conectado à internet;
- ***firebaseDatabase.getReference()*** é retornada o objeto *DatabaseReference* para ler ou gravar dados no banco de dados.

13- Já a classe *EstoqueBLL* conterá:

- Um método para inicializar a conexão com o *Firebase* – isto é, executar o método da *DAL* (*Data access layer*) do projeto:

```
public static DatabaseReference inicializarFirebase(Context context,
FirebaseDatabase firebaseDatabase) {
    return EstoqueDAL.inicializarFirebase(context, firebaseDatabase);
}
```

- Um método para validar se os dados foram preenchidos corretamente:

```
public void validaDados(Mercadoria m) {

    Erro.setErro(false);

    if (m.getProduto() == null || m.getProduto().isEmpty()) {
        Erro.setErro("Preencha o campo do nome do produto!");
        Erro.setErro(true);
        return;
    }

    if (m.getQtd() == null || m.getQtd().isEmpty()) {
        Erro.setErro("Preencha o campo da quantidade da mercadoria!");
        Erro.setErro(true);
        return;
    }

    if (m.getValor() == null || m.getValor().isEmpty()) {
```

```
        Erro.setErro("Preencha o campo do valor da mercadoria!");
        Erro.setErro(true);
        return;
    }
}
```

Assim, se o usuário deixar algum campo vazio será passada à classe *Erro* a respectiva mensagem de erro e o valor *true* (que será útil em breve). É desnecessário verificar se foi colocado um valor numérico nas áreas "Quantidade" e "Valor" porque os *Edit Texts* possuem *flags* que impedem isso.

- E por fim um método para limpar os campos e tirar o foco deles:

```
public static void limparCampos(EditText edtProduto, EditText edtQtd, EditText
edtValor) {
    edtProduto.setText(null);
    edtQtd.setText(null);
    edtValor.setText(null);
    edtProduto.clearFocus();
    edtQtd.clearFocus();
    edtValor.clearFocus();
}
```

14- Agora será feita a conexão com o banco de dados. Para isso, primeiro serão construídos objetos da *FirebaseDatabase* e *DatabaseReference* na *MainActivity*:

```
public class MainActivity extends AppCompatActivity {

    FirebaseDatabase firebaseDatabase;
    DatabaseReference databaseReference;

    ...
}
```

Em seguida será usado o *EstoqueBLL.inicializarFirebase()* no *onCreate() da Main Activity*:

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    databaseReference = EstoqueBLL.inicializarFirebase(MainActivity.this,
        firebaseDatabase);


}
```

O *databaseReference* é o ponto de partida para todas as operações do banco de dados.

15- Desenvolvendo a operação *Create* (botão adicionar do menu):

- Antes de tudo, serão chamados os objetos da tela que serão identificados no *onCreate()*:

```
public class MainActivity extends AppCompatActivity {

    EditText edtProduto, edtQtd, edtValor;
    ListView listV_produtos;

    ...

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        edtProduto = (EditText) findViewById(R.id.editProduto);
        edtQtd = (EditText) findViewById(R.id.editQtd);
        edtValor = (EditText) findViewById(R.id.editValor);
        listV_produtos = (ListView) findViewById(R.id.listV_produtos);
```

```
        databaseReference = EstoqueBLL.inicializarFirebase(MainActivity.this,
        firebaseDatabase);


    }

    ...
}
```

- Para que o menu apareça será necessário sobrescrever o método *onOptionsMenuCreate()* utilizando o *MenuInflater*, o qual irá ler o *menu_main.xml* e cria-lo conforme os objetos que configuramos:

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.menu_main,menu);
    return super.onCreateOptionsMenu(menu);
}
```

  Não esqueça que basta escrever “onCreateOptionsMenu” que será sugerido o *override* do método automaticamente.

- Quando um item do menu de opções é selecionado, o sistema chama o método *onOptionsItemSelected()* da atividade. Esse método passa o *MenuItem* selecionado. É possível identificar o item chamando *getItemId()*, que retorna o código exclusivo para o item de menu. Assim, pode-se sobrescrever o *onOptionsItemSelected()* e fazer um *switch* com os *ids* dos itens para cada operação. Começando pelo item *menu_add* (botão adicionar):

```
@Override
public boolean onOptionsItemSelected(@NonNull MenuItem item) {
    int id = item.getItemId();

    final Mercadoria m = new Mercadoria();

    AlertDialog.Builder builder = new AlertDialog.Builder(MainActivity.this);
    builder.setPositiveButton("OK", new DialogInterface.OnClickListener() {
```

```
        @Override
        public void onClick(DialogInterface dialog, int which) {
            // Apenas para fechar o dialog
        }
    });

    switch (id) {
        case R.id.menu_add: {
            m.setId(UUID.randomUUID().toString());
            m.setProduto(edtProduto.getText().toString().trim());
            m.setQtd(edtQtd.getText().toString().trim());
            m.setValor(edtValor.getText().toString().trim());

            EstoqueBLL.validaDados(m);
            if (Erro.getErro()) {
                builder.setMessage(Erro.getMsg()).setTitle("ERRO!");
                builder.create().show();
            } else {
                databaseReference.child("Mercadoria").child(m.getId()).setValue(m);
                EstoqueBLL.limparCampos(edtProduto, edtQtd, edtValor);
            }

            return true;
        }
        default:
            return super.onOptionsItemSelected(item);
    }
}
```

Onde:

- ***onOptionsItemSelected*** é *booleano* e quando processar um item corretamente deve retornar *true*. Caso não processe o item de menu, deve-se chamar a sua implementação de superclasse (a implementação padrão retornará falso);

- ***builder*** é o objeto que carrega os atributos de preferências do *AlertDialog*. Esse objeto será usado nos três itens de menu para exibir mensagens de erro ou confirmação. Dessa forma, ele é instanciado e seu *PositiveButton* é alterado para ter valor de “OK” e não é executado nada para que, ao clicar sobre esse botão, o diálogo apenas feche. Observe que a sua mensagem e título só são alterados dentro do *switch*, recebendo valores respectivos a um eventual erro verificado pelo método *validaDados()*;
- ***m*** é o objeto de *Mercadoria* que receberá os valores dos *Edit Texts*, com exceção do seu *id*, o qual é gerado aleatoriamente através de *UUID.randomUUID()*. Caso os dados de *m* sejam válidos, os seus valores serão passados ao banco de dados através do *databaseReference.setValue()*. Note que antes disso é usado o método *child()*, encarregado por definir um caminho base de dados. Assim, pode-se dizer que primeiro é criada a tabela “Mercadoria”, abaixo dela estará uma mercadoria determinada pelo seu *id* e abaixo dele terão todos os atributos de *m* e seus respectivos valores. Exemplo da estrutura:

- ***trim()*** serve para remover espaços em branco do início e do fim do texto;
- ***limparCampos()*** é um método da *BLL* (*Business Logic Layer*) do projeto que limpará os campos e removerá o foco de qualquer um deles.

16- Desenvolvendo a operação *Read* (*select* da *List View*):

- Primeiro será preciso instanciar um *ArrayList* de tipo *Mercadoria* e construir o seu *adapter*. Além disso, para atualizar os campos com os dados do objeto selecionado e auxiliar nas próximas operações, deve-se criar um outro objeto da classe *Mercadoria*:

```
public class MainActivity extends AppCompatActivity {
  ...

  private List<Mercadoria> listMercadoria = new ArrayList();
  private ArrayAdapter<Mercadoria> arrayAdapterMercadoria;

  Mercadoria mSelecionada;

}
```

- Para que seja implementada na *List View* todas as alterações feitas no banco de dados será necessário adicionar um *EventListener* à tabela *Mercadoria*:

```
private void eventoDatabase() {
   databaseReference.child("Mercadoria").addValueEventListener(new
ValueEventListener() {
      @Override
      public void onDataChange(@NonNull DataSnapshot snapshot) {
         listMercadoria.clear();
         for (DataSnapshot objSnapshot: snapshot.getChildren()) {
            Mercadoria m = objSnapshot.getValue(Mercadoria.class);
            listMercadoria.add(m);
         }
         arrayAdapterMercadoria = new ArrayAdapter(MainActivity.this,
               android.R.layout.simple_list_item_1, listMercadoria);
         listV_produtos.setAdapter(arrayAdapterMercadoria);
      }

      @Override
      public void onCancelled(@NonNull DatabaseError error) {
```

```
        }
    });
}
```

Onde:

- As classes que implementam essa interface **addValueEventListener** podem ser usadas para receber eventos sobre alterações de dados em um local. Fixe o ouvinte a um addValueEventListener localização do usuário (new ValueEventListener).
- Já para passar os valores da mercadoria selecionada aos campos será preciso definir um *onItemClickListener()* no *ListV_produtos*. Para tanto, deve-se sobrescrever o *onItemClick()* do *AdapterView* para que os dados do item selecionado sejam recuperados à *mSelecionada*, enfim passando seus valores aos *Edit Texts*:

```
listV_produtos.setOnItemClickListener(new AdapterView.OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> parent, View view, int position, long
id) {
        mSelecionada = (Mercadoria)parent.getItemAtPosition(position);
        edtProduto.setText(mSelecionada.getProduto());
        edtQtd.setText(mSelecionada.getQtd());
        edtValor.setText(mSelecionada.getValor());
    }
});
```

- Por questões estéticas, pode-se sobrescrever o *toString()* de *Mercadoria* para retornar apenas o nome do objeto, isto é, o atributo *produto*. Então basta ir à classe *Mercadoria*, pressionar *Alt+Insert*, selecionar *toString(),* clicar em “OK” e substituir para retornar apenas *produto*:

```
@Override
public String toString() {
    return produto;
}
```

17- Desenvolvendo a operação *Update* (botão salvar do menu):

- As lógicas do botão adicionar e salvar são semelhantes. A principal diferença é que o id da mercadoria não é gerado automaticamente, mas sim recuperada da mSelecionada. Logo, ao registrar no banco de dados e utilizar o mesmo id é feita apenas uma substituição das informações:

```
@Override
public boolean onOptionsItemSelected(@NonNull MenuItem item) {

    ...

    switch (id) {
        case R.id.menu_add: {...}
        case R.id.menu_save: {
            try {
                m.setId(mSelecionada.getId());
                m.setProduto(edtProduto.getText().toString().trim());
                m.setQtd(edtQtd.getText().toString().trim());
                m.setValor(edtValor.getText().toString().trim());

                EstoqueBLL.validaDados(m);
                if (Erro.getErro()) {
                    builder.setMessage(Erro.getMsg()).setTitle("ERRO!");
                    builder.create().show();
                } else {
                    databaseReference.child("Mercadoria").child(m.getId())
                            .setValue(m);
                    EstoqueBLL.limparCampos(edtProduto, edtQtd, edtValor);
                }
            } catch (Exception e) {
                builder.setMessage("Selecione um produto!").setTitle("ERRO!");
                builder.create().show();
            }

            return true;
```

```
        }
        default:
            return super.onOptionsItemSelected(item);
    }
}
```

Observe que é utilizado um try-catch para garantir que o usuário tenha selecionado alguma mercadoria.

18- Desenvolvendo a operação *Delete* (botão deletar do menu):

- Antes de realizar a eliminação de uma mercadoria é mais adequado perguntar ao usuário se ele tem certeza de que deseja fazer isso. Assim, pode-se definir um dialog perguntando isso e apresentando um botão negativo para cancelar (e não fazer nada) e outro para confirmar (e deletar):

```
@Override
public boolean onOptionsItemSelected(@NonNull MenuItem item) {

    ...

    switch (id) {
        case R.id.menu_add: {...}
        case R.id.menu_save: {...}
        case R.id.menu_delete: {
            try {
                EstoqueBLL.validaDados(mSelecionada);

                mSelecionada.getId();

                builder.setTitle("AVISO!").setMessage("Tem certeza de que deseja
apagar " +
                        mSelecionada.getProduto() + "?");
                builder.setNegativeButton("Cancelar", new
DialogInterface.OnClickListener() {
```

```
                @Override
                public void onClick(DialogInterface dialog, int which) {
                    // apenas para fechar o dialog
                }
            }).setPositiveButton("Sim", new DialogInterface.OnClickListener() {
                @Override
                public void onClick(DialogInterface dialog, int which) {
                    m.setId(mSelecionada.getId());

databaseReference.child("Mercadoria").child(m.getId()).removeValue();
                    EstoqueBLL.limparCampos(edtProduto, edtQtd, edtValor,
mSelecionada);
                }
            }).create().show();
        } catch (Exception e) {
            builder.setMessage("Selecione um produto!").setTitle("ERRO!");
            builder.create().show();
        }

        return true;
    }
    default:
        return super.onOptionsItemSelected(item);
  }
}
```

Obviamente, nesse botão são apenas recuperados os valores da mSelecionada. Em seguida, ao invés de ser usado setValue() no databaseReference, é utilizado removeValue() para apagar os dados.

- Para finalizar, pode-se alterar o método limparCampos() para melhorar o projeto. Note que os dados de um mSelecionada permanecem armazenados mesmo ao, por exemplo, salvar alterações de uma mercadoria. Assim, se o usuário selecionar uma mercadoria, alterar suas informações e, mesmo após os campos serem limpos, clicar no botão

deletar, será questionado se ele deseja deletar a última mercadoria manipulada. Portanto, podemos implementar em limparCampos():

```
public static void limparCampos(EditText edtProduto, EditText edtQtd, EditText
edtValor, Mercadoria mSelecionada) {

    ...

    try {
        mSelecionada.setId(null);
    } catch (Exception e) {
        /* Se cair aqui significa que o objeto pessoaSelecionada sequer foi
instanciado e então
        não é preciso fazer nada */
    }
}
```

Dessa forma, o id da mSelecionada será limpo. Entretanto, deve-se também adicionar uma condição no método validaDados() para verificar se o id a ser selecionado será valido. Desse modo, basta acrescentar:

```
public static void validaDados(Mercadoria m) {

    Erro.setErro(false);

    if (m.getId() == null || m.getId().isEmpty()) {
        Erro.setErro("Selecione um produto!");
        Erro.setErro(true);
        return;
    }

    if (m.getProduto() == null || m.getProduto().isEmpty()) {...}

    if (m.getQtd() == null || m.getQtd().isEmpty()) {...}
```

```
    if (m.getValor() == null || m.getValor().isEmpty()) {...}
}
```

E por fim resta modificar as linhas em que *limparCampos()* é chamado para:

```
EstoqueBLL.limparCampos(edtProduto, edtQtd, edtValor, mSelecionada);
```

Referências

SCHUMAN, Deivis. “Você conhece a história do Android? Confira aqui toda sua trajetória”. Disponível em: < https://www.nextpit.com.br/historia-do-android >. Acesso em: 15 de setembro de 2020.

MEYER, Maximiliano. “A história do Android”. Disponível em: < https://www.oficinadanet.com.br/post/13939-a-historia-do-android >. Acesso em: 16 de setembro de 2020.

TecMundo. “A história do Android, o robô que domina o mercado mobile”. Disponível em: <https://www.tecmundo.com.br/ciencia/120933-historia-android-robo-domina-o-mercado-mobile-video.htm >. Acesso em: 16 de setembro de 2020.

GeekHunter. “Programando em Java Android”. Disponível em: < https://blog.geekhunter.com.br/java-android/#:~:text=%C3%89%20de%20conhecimento%20geral%20que,Android%2C%20por%20isso%20essa%20jun%C3%A7%C3%A3o >. Acesso em: 16 de setembro de 2020.

TechTudo. “Android 4.4 kitkat”. Disponível em: <https://www.techtudo.com.br/tudo-sobre/android-4-4.html >. Acesso em: 17 de setembro de 2020.

IBest. “Android 11 lançado! Conheça todas as novidades dessa nova versão”. Disponível em: <https://canaltech.com.br/android/android-11-atualizacao-funcoes-novidades-159556/>. Acesso em: 17 de setembro de 2020.

RINALDI, Camila. “Testamos o Android Q: veja o que esperar da nova versão do Android”. Disponível em: <https://olhardigital.com.br/noticia/android-10-q-funcoes-lancamentos-e-recursos/83641https://olhardigital.com.br/noticia/android-10-q-funcoes-lancamentos-e-recursos/83641>. Acesso em: 17 de setembro de 2020.

TecMundo. “Confira 7 novidades do Android 9.0 Pie”. Disponível em: <https://www.tecmundo.com.br/software/132937-7-novidades-android-9-pie.htm>. Acesso em: 17 de setembro de 2020.

HIGA, Paulo. “As melhores novidades do Android 8.0 Oreo”. Disponível em: <https://tecnoblog.net/211151/android-8-o-novidades-recursos/#:~:text=Nova%20vers%C3%A3o%20do%20Android%20tem,seguran%C3

%A7a%2C%20notifica%C3%A7%C3%B5es%2C%20%C3%A1udio%20e%20mais HYPERLINK "https://tecnoblog.net/211151/android-8-o-novidades-recursos/"& HYPERLINK "https://tecnoblog.net/211151/android-8-o-novidades-recursos/"text=O%20Google%20lan%C3%A7ou%20o%20Android,uma%20s%C3%A9rie%20de%20refinamentos%20internos.>. Acesso em: 18 de setembro de 2020.

TecMundo. “Linha do tempo: por dentro da evolução do Android”. Disponível em: <https://www.tecmundo.com.br/android/82344-linha-tempo-dentro-evolucao-do-sistema-android.htm#:~:text=O%20Android%20%C3%A9%20um%20sistema,e%20das%20prefer%C3%AAncias%20do%20propriet%C3%A1rio.>. Acesso em: 18 de setembro de 2020.

Principais sites utilizados para desenvolvimento dos aplicativos.

- https://developer.android.com/?hl=pt-br
- https://firebase.google.com/?hl=pt
- https://www.devmedia.com.br/