AGENDA 1

UTILIZANDO O ANDROID STUDIO PARA DESENVOLVIMENTO MOBILE

GEEaD - Grupo de Estudos de Educação a Distância

Centro de Educação Tecnológica Paula Souza

GOVERNO DO ESTADO DE SÃO PAULO

EIXO TECNOLÓGICO DE INFORMAÇÃO E COMUNICAÇÃO

CURSO TÉCNICO EM DESENVOLVIMENTO DE SISTEMAS

PROGRAMAÇÃO MOBILE I

Expediente

Autor:

GUILHERME HENRIQUE GIROLLI

Revisão Técnica:

Eliana Cristina Nogueira Barion

Revisão Gramatical:

Juçara Maria Montenegro Simonsen Santos

Editoração e Diagramação: Flávio Biazim

São Paulo – SP, 2020

MERGULHANDO NO TEMA...

Como todos já sabem, um dispositivo computacional, seja ele qual for, é composto pela sua parte física, denominada Hardware e sua parte lógica, chamada de Software.

Não é diferente com os dispositivos Mobile, como celulares ou Smartphones, Tablets, etc. Não adiantaria tanta potência e design, sem um sistema operacional para fazer o dispositivo ganhar vida!

Com a evolução dos equipamentos, a indústria necessitava de um sistema genérico, que funcionasse em diversos *Hardwares* diferentes, e que estivesse em constante manutenção e evolução, assim como os dispositivos físicos.

O sistema operacional Android, tem seu código aberto pela Google, e muitas empresas fabricantes de aparelhos, já trabalham alterando esse código, customizando e otimizando para se adequar aos seus produtos. Alguns dispositivos hoje, são lançados contendo uma versão do sistema que possui seu código aberto e parte sendo proprietário da empresa do aparelho, em virtude das customizações e otimizações.

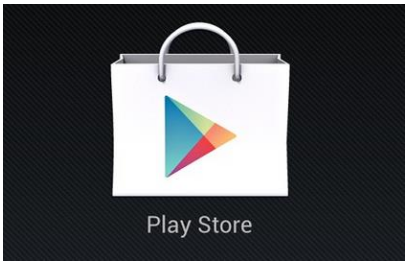


Figura 3 – Aplicativo de compras da loja da empresa Google, chamada de Play Store.

Com sua grande utilização, o sistema operacional Android abriu as portas de um novo mercado, o mercado de desenvolvimento Mobile. O Android, possui uma loja de aplicativos, chamada de "Play Store", que são comercializados aos seus usuários que buscam ferramentas para o lazer, trabalho, estudo, etc. E com a popularização da loja virtual da empresa Google, muitos desenvolvedores, tiveram a oportunidade de publicar seus trabalhos e ferramentas para todo o mundo, onde muitos saíram do anonimato e se tornando milionários. Uma outra grande forma de trabalho para o desenvolvedor, é criar projetos para fins particulares, ou seja, desenvolver aplicativos para empresas como bancos, empresas de vendas, etc.

Para atender cada vez mais a necessidade do mercado de desenvolvimento, foi criada a ferramenta Android Studio, para auxiliar e facilitar o desenvolvimento de aplicativos, seja ele para comercialização na loja da empresa Google, ou para atender empresas específicas.

No tópico a seguir, apresentamos o Android Studio e assim já estaremos aptos para instalar e executar o nosso primeiro projeto desenvolvido para o sistema operacional Android!

Com o surgimento e aceitação do sistema operacional Android, o mercado necessitava de ferramentas para atender o desenvolvimento de aplicativos, gerados a partir da linguagem Java.

Por algum tempo, os desenvolvedores utilizavam a ferramenta Eclipse. Para conseguir operar essa ferramenta, era necessário muitos ajustes e configurações. Muitos desenvolvedores abandonaram os estudos nessa área, pelo simples fato de não conseguir, configurar o ambiente e produzir o seu primeiro projeto. Tornando uma experiência frustrante, seu primeiro contato com esse universo da programação Mobile.

Em maio de 2013, a empresa Google fez o lançamento da sua ferramenta de desenvolvimento para a plataforma Android, o Android Studio. Uma ferramenta de fácil instalação e configuração, utilizando a linguagem de marcação XML (*Extensible Markup Language*) para a criação das telas do aplicativo, as chamadas *Activity's*. E para o desenvolvimento operacional do aplicativo é utilizado a linguagem de programação Java.

O Android Studio é a ferramenta oficial para desenvolvimento exclusivo para a plataforma Android, ou seja, **esta ferramenta não desenvolve para outros sistemas operacionais**.

Atualmente, no mercado contamos com outras ferramentas de desenvolvimento, algumas permitem a programação de um único projeto, e exportação deste projeto para trabalhar/executar nas principais plataformas de dispositivos móveis. Essas ferramentas de desenvolvimento são chamadas de **multiplataforma**.

Muitas empresas e desenvolvedores, defendem a ideia de que para a criação de um bom aplicativo, é fundamental que ele tenha desempenho e baixa utilização dos recursos do dispositivo Mobile.

Desta forma, levantam a bandeira de que o desenvolvimento ideal é aquele realizado na ferramenta exclusiva da plataforma, pois essa ferramenta não trabalha com conversões, ela é nativa e desenvolvida para atender as necessidades de um único sistema operacional. É válido lembrar que isso não é uma lei, é apenas a opinião de uma parcela de desenvolvedores.

Por outro lado, o mercado de ferramentas de desenvolvimento multiplataforma cresce, pois otimizam o tempo gasto na produção de aplicativos.

Devidamente apresentada nossa ferramenta de desenvolvimento Android Studio, chegou a hora de efetuar a instalação dela, que além de robusta e sensacional, conta com a licença de uso no nível Apache2.0, ou seja, ela é de uso gratuito! Vamos aproveitar essa oportunidade ao máximo?

Acesse o conteúdo do manual Android Studio aqui.

Activity

A ***Activity* é um componente do aplicativo**, responsável por fornecer uma tela para o usuário, desta forma gerando uma interface com ele. O aplicativo pode conter uma ou mais *Activity's*, de acordo com a sua necessidade.

Nos aplicativos com mais de uma tela, ou seja, mais de uma *Activity*, uma delas é escolhida como principal. É esta *Activity* principal que será exibida inicialmente ao usuário, e assim, será a responsável por chamar as outras telas do projeto.

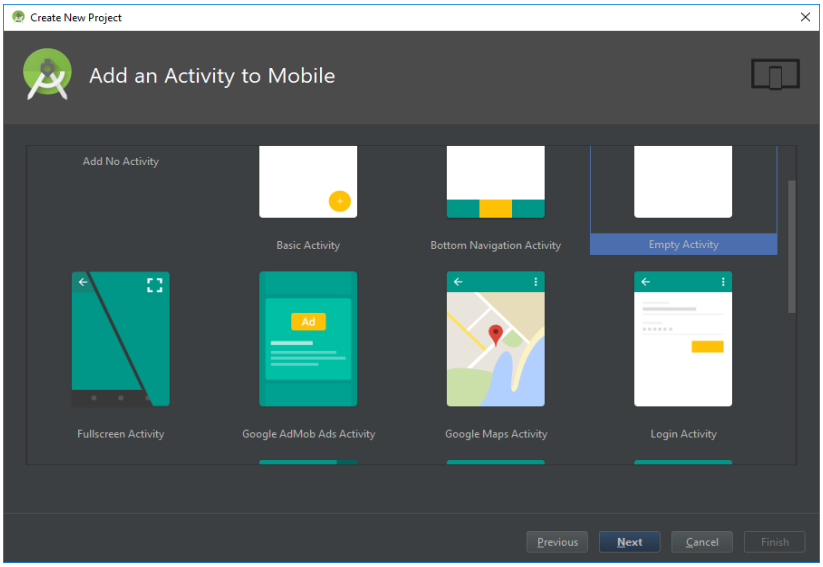


Figura 16 – Tela de sugestões de Activity.

Para o desenvolvimento da *Activity*, utilizamos o modo "Design" ou "XML". No modo "Design" trabalhamos com uma paleta de componentes, onde arrastamos para a nossa tela, tornando o processo mais rápido, pois ele gera automaticamente o XML.

Já no modo "XML", utilizamos a linguagem de marcação para codificar nossa interface, e por exte processo o usuário necessita conhecer bem seus códigos e funções.

Na criação de um projeto no Android Studio, nossa IDE fornece alguns modelos de *Activity's*, previamente desenvolvidas para ajudar os desenvolvedores Mobile no seu trabalho. A figura 16, mostra alguns exemplos de telas, para usos diversos, como menus, mapas, login, etc. Essas telas ficam disponíveis para utilização ou alteração do desenvolvedor, fazendo assim ele ganhar tempo no desenvolvimento de aplicativos.

Em nosso projeto inicial, vamos utilizar a "***Empty Activity***", ou seja, é uma *Activity* vazia! Utilizada para a criação de projetos específicos, que necessita de uma tela neutra.

Marcelo, vamos criar nosso primeiro projeto no Android Studio?

Agora que nossa bagagem de conhecimento já está completa, podemos juntamente com Marcelo, desenvolver nosso primeiro projeto no Android Studio! É fundamental que neste momento você esteja na tela inicial do Android Studio. Vamos iniciar clicando em "***Star a new Android Studio project***".

Quando solicitamos a criação de um novo projeto, uma tela de assistente é aberta, para auxiliar nesta etapa.

Agora, chegou a hora de escolher nossa **Activity**, isto é, nossa primeira tela do projeto. O *Android Studio* permite desenvolvimento para os mais diferentes tipos de dispositivos mobile, desde relógios, passando por smartphones, até chegar em carros.

É nesta etapa que escolhemos para quais dispositivos programar. Nosso foco inicial é os smartphones, então vamos até a guia **"Phone and Tablet"**.

Como combinamos anteriormente, vamos escolher a **"Empty Activity"**, e aplicar **"Next"**, como mostra a figura abaixo.

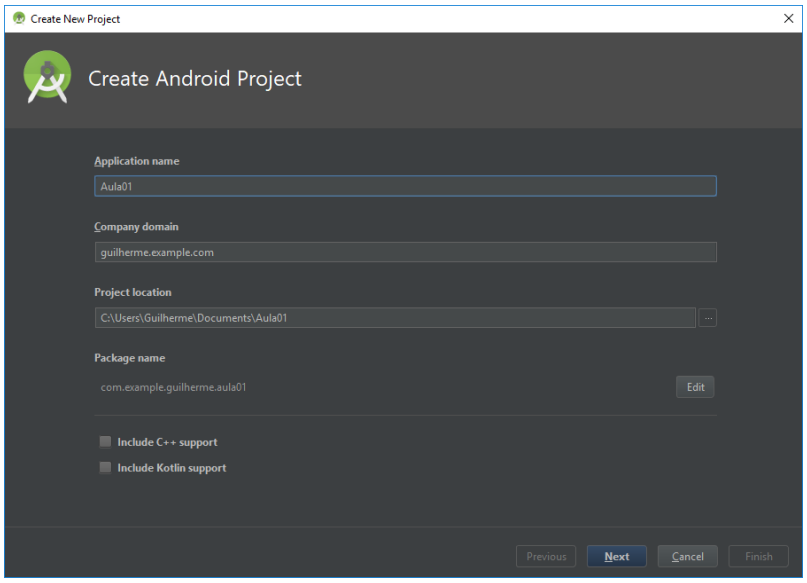


Figura 17 – Início da criação do projeto Aula01.

São solicitados dados do projeto, como o Nome da Aplicação, o domínio da empresa ou pessoa que está desenvolvendo e a localização para salvar os arquivos do projeto. As informações anteriores são alteradas de acordo com a sua necessidade. Não alteraremos o **"Package name"**, ou nome do pacote do projeto, e a linguagem Java.

É importante lembrar que o Java e o *Android Studio* utilizam "Case Sensitive", ou seja, faz diferenciação entre letras maiúsculas e minúsculas. Então, a forma como foi escrito o nome do projeto tem total importância no decorrer da programação!

Indique qual é a API mínima de trabalho. No nosso caso, vamos escolher a **"API 25: Android 7.1.1 (Nougat)"**. Lembre-se de instalar essa API no processo de download do SDK.

O nosso primeiro projeto leva o nome de **"Aula01"**. Veja na figura 6 como ficou nossa tela do assistente. Depois, é só clicar no botão **"Next"**.

É importante lembrar que o Java e Android Studio utiliza "Case Sensitve", ou seja, faz diferenciação com letras maiúsculas e minúsculas, note na figura 17, como foi escrito o nome do projeto, isso faz total importância no decorrer da sua programação!

O Android Studio, permite desenvolvimento para os mais diferentes tipos de dispositivos Mobile, desde relógios, passando por Smartphones, até chegar nos carros. É nesta etapa que vamos escolher, para quais dispositivos vamos programar. Nosso foco inicial são os Smartphones, então vamos desmarcar qualquer outra opção, deixando apenas "***Phone and Tablet***" selecionado.

Indique qual é a API mínima de trabalho, no nosso caso, vamos escolher a "***API 24: Android 7.0 (Nougat)***", que foi o SDK instalado anteriormente através do SDK Manager.

Uma observação importante, vista na figura 18, é que ao escolher a API, o Android Studio nos informa uma estimativa de dispositivos compatíveis com o aplicativo que será desenvolvido, neste caso 8.1 % de todos os dispositivos do mundo. Isso é importante, pois nos auxilia no processo de escolha da API mínima em projetos futuros, durante a sua promissora carreira de desenvolvedor Mobile.

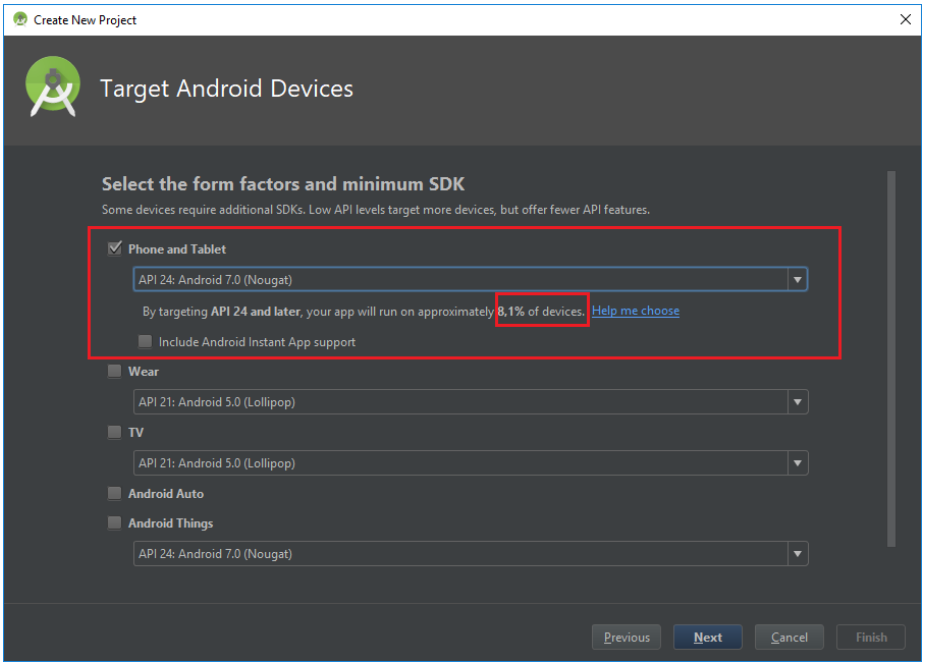


Figura 18 – Escolha dos dispositivos e API para programação.

A opção "***Include Android Instant App support***" deve ser desmarcada, pois não vamos oferecer nosso aplicativo através dessa tecnologia, que é basicamente uma versão de execução fora do método convencional, oferecido pela loja de aplicativos da Google. Ou seja, nossa aplicativo poderia ser executado em navegadores, não sendo necessário à sua instalação no aparelho.

Após a escolha do dispositivo e API, vamos prosseguir com o botão "Next" para a próxima etapa. Agora chegou a hora de escolher nossa *Activity*, ou seja, nossa primeira tela do projeto.

Como combinamos anteriormente, vamos escolher a "***Empty Activity***", e aplicar "***Next***", como mostra a figura 19.

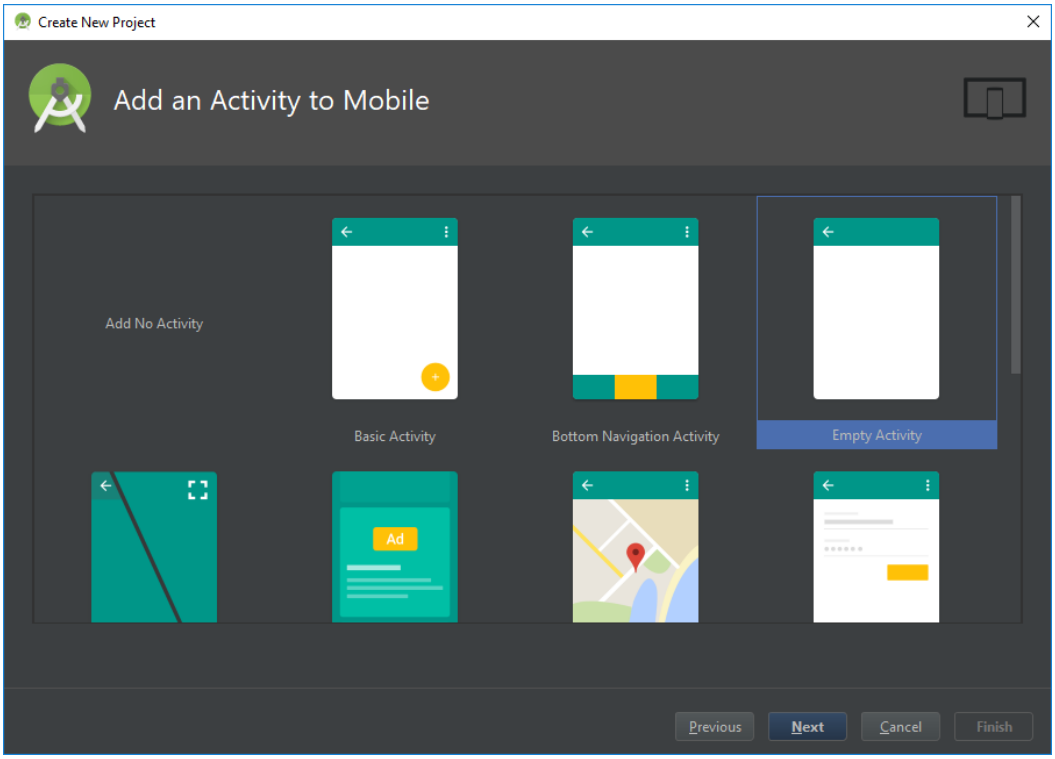


Figura 19 – Escolha da Activity.

Para finalizar, vamos configurar nossa Activity, informando o nome dela, e o nome de Layout. Ficou confuso? Calma que vou explicar!

O nome da *Activity* é o nome da classe Java, responsável pelos métodos e atributos da nossa tela ou interface, ou seja, responsável pelas ações da tela, como exemplo, efetuar uma soma, limpar um campo com o toque de um botão, etc. Já o nome do *Layout* é o nome do arquivo XML responsável por gerar a parte visual da nossa tela, ou seja, o que o usuário vai visualizar, como cores, botões, caixas de texto, etc.

Vamos colocar o nome da nossa *Activity* como "**Calculadora**" atenção as letras maiúsculas e minúsculas, e automaticamente o nome do Layout é criado como "***activity_calculadora***". Deixe a opção "**Backwards compatibility (AppCompat)**" desmarcada, para que nossa *Activity* trabalhe com compatibilidade em exibições em versões anteriores. Após as configurações clique em "***Finish***" (Figura 20) e aguarde o carregamento do projeto.

Esta última tela, de configuração da *Activity*, não é padrão, para cada tipo de *Activity* é solicitado opções diferentes de configurações.

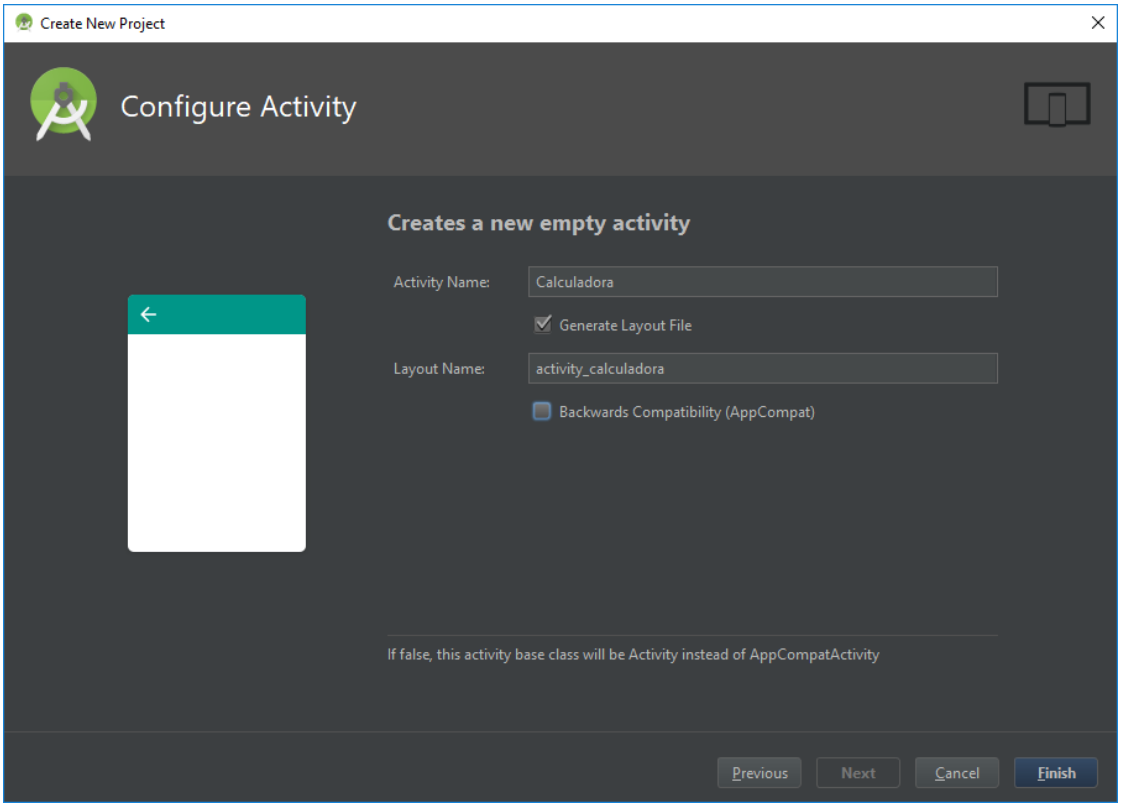


Figura 20 – Configuração da Activity.

Conhecendo a IDE (Integrated Development Environment) do **Android Studio**

Após todas as etapas anteriores, e criação do nosso primeiro projeto, chegou a hora de conhecer alguns detalhes importantes sobre a nossa ferramenta. Precisamos alinhar alguns termos e locais que utilizaremos com mais frequência em nossa IDE (Integrated Development Environment) ou Ambiente Integrado de Desenvolvimento. Assim, nosso processo de evolução se torna mais rápido, pois não vamos perder tempo com orientações repetidas para acesso aos principais recursos do ambiente.

A figura 21, retirada do site oficial do Android Studio, identifica todas as áreas da nossa IDE, com uma breve explicação sobre cada grupo de itens. Vamos acompanhar!

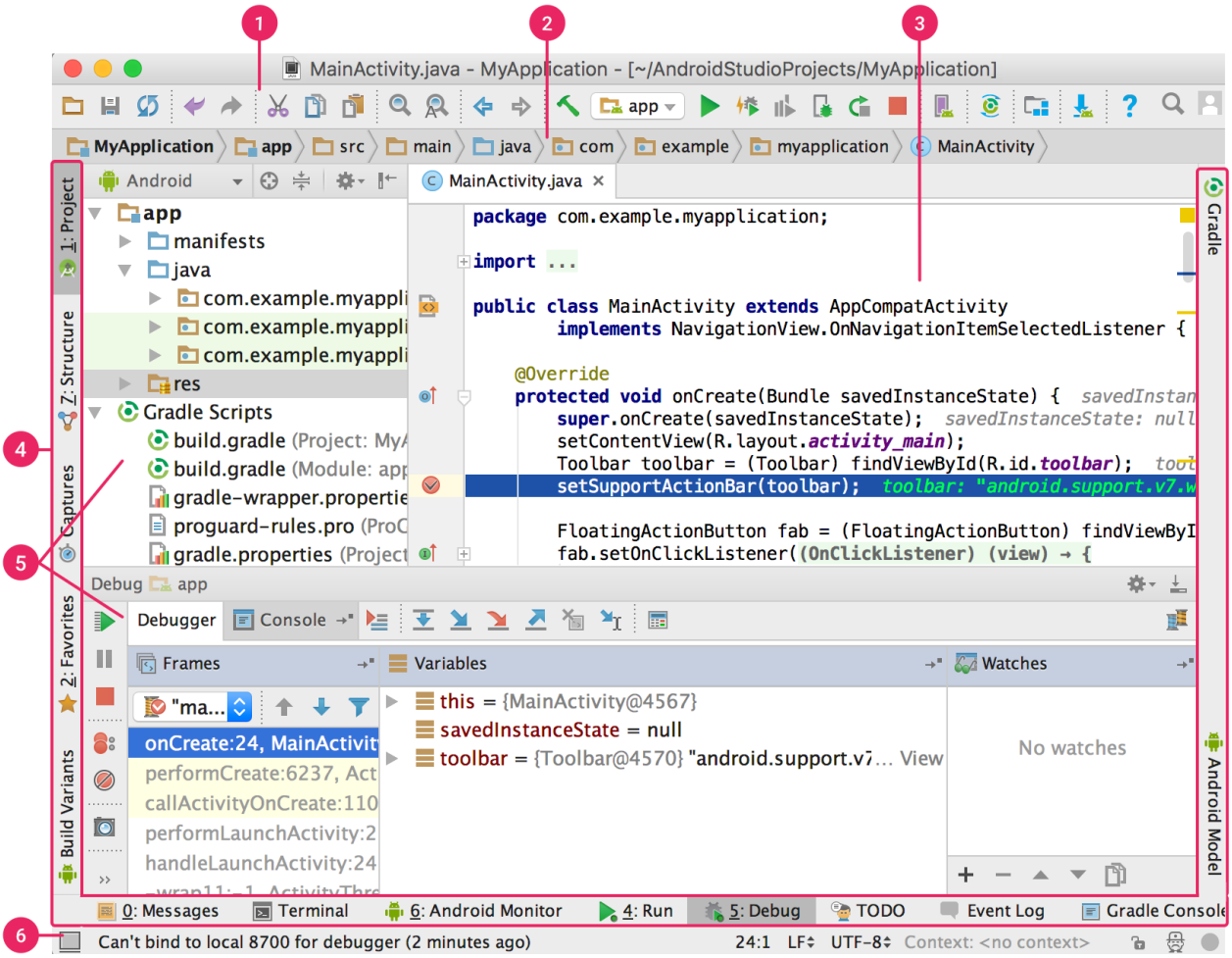


Figura 21 – Explicação sobre as áreas da IDE do Android Studio. Disponível em: https://developer.android.com/studio/intro/?hl=pt-br

1. A **barra de ferramentas** permite executar diversas ações, incluindo executar aplicativos e inicializar ferramentas do Android.
2. A **barra de navegação** ajuda na navegação pelo projeto e na abertura de arquivos para edição. Ela oferece uma visualização mais compacta da estrutura visível na janela **Project**.
3. A **janela do editor** é o local em que você cria e modifica código. Dependendo do tipo de arquivo atual, o editor pode mudar. Por exemplo, ao visualizar um arquivo de layout, o editor abre o Editor de layout.
4. A **barra de janela de ferramentas** fica fora da janela do IDE e contém os botões que permitem expandir ou recolher a janela de cada ferramenta.
5. A **janela das ferramentas** dá acesso a tarefas específicas, como gerenciamento de projetos, busca, controle de versão e muitos outros. Você pode expandi-las e recolhê-las.
6. A **barra de status** mostra o status do projeto e do próprio IDE, além de advertências e mensagens.

A estrutura do projeto é exibida pela ferramenta "***Project***", nela encontramos os arquivos Java, arquivos de configuração do projeto, arquivos XML que geram a interface, etc.
Para acesso aos arquivos Java da aplicação, devemos localizar o pacote criado, que fica em "***App -> java -> com.example.*.aula01***", conforme figura 22, e clicar duas vezes sobre o arquivo que deseja abrir.

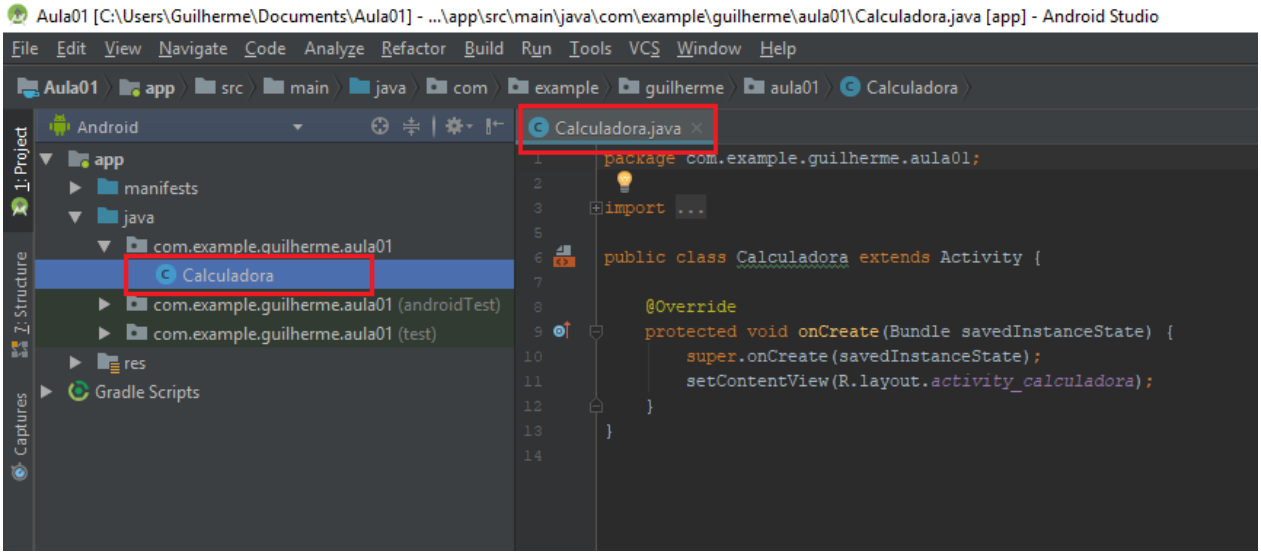


Figura 22 – Arquivos Java do projeto.

Para acesso aos arquivos de *Layout* das *Activity's* que geram nossas telas, basta procurar na ferramenta "***Project***" a pasta *layout*, que fica em "***App -> res -> layout***". Verifique a figura 23, e de um duplo clique para abrir o arquivo.

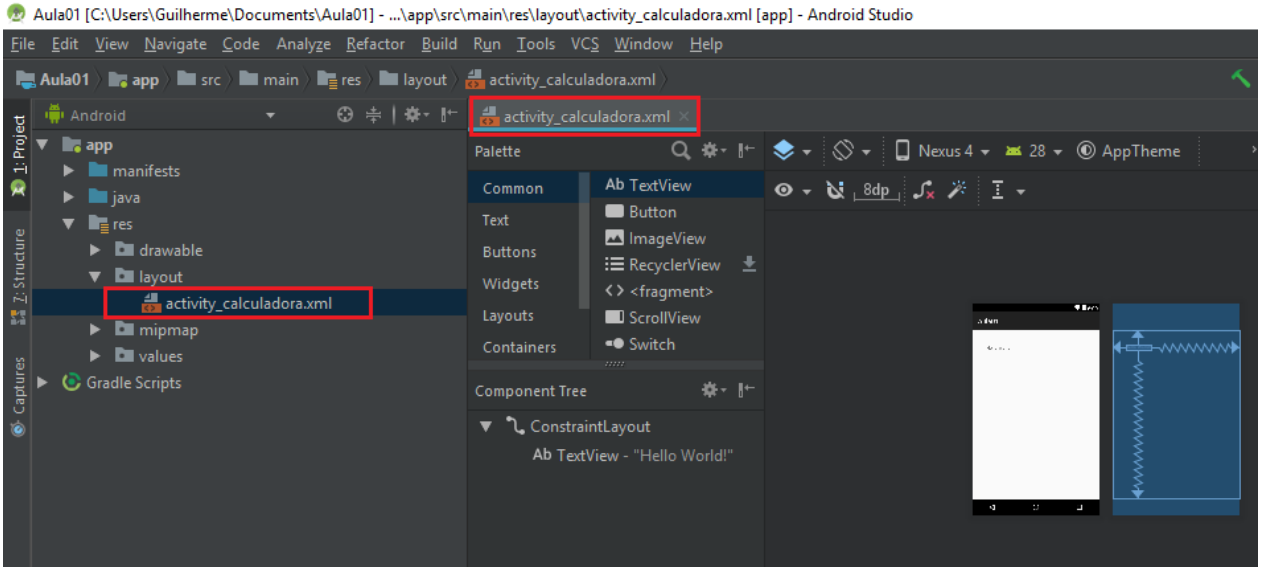


Figura 23 – Arquivos XML do projeto.

Na barra de ferramentas do Android Studio, é importante nesse primeiro momento também conhecer alguns botões responsáveis pela execução da aplicação e outras ferramentas do Android Studio, verifique na imagem 24 os itens.

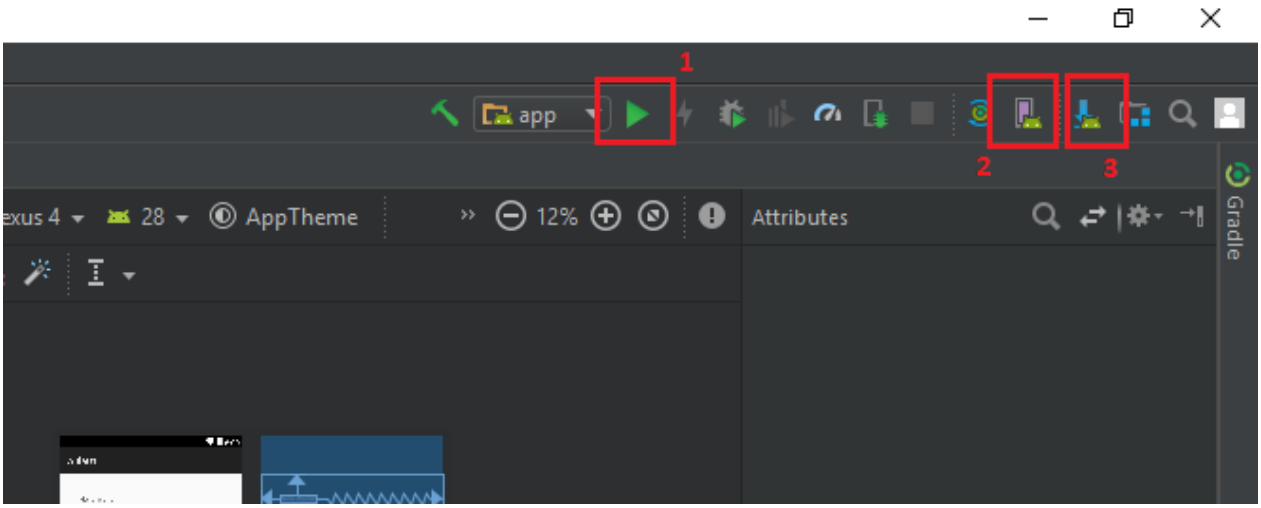


Figura 24 – Barra de ferramentas do Android Studio.

Item 1 – Run 'app', ou seja, execução do aplicativo para teste.

Item 2 – AVD (Android Virtual Device) Manager.

Item 3 – Segunda forma para acessar o SDK Manager, visto anteriormente neste material.

A seguir vamos conhecer a AVD (Android Virtual Device), necessária para executar nossos aplicativos durante a fase de desenvolvimento.

AVD (Android Virtual Device)

Como verificamos, o Android Studio desenvolve aplicativos nativos para o sistema operacional Android. Nosso ambiente de laboratório e estudos utiliza o sistema operacional Windows 10, o que gerou uma certa dúvida em Marcelo. Como vamos fazer para testar nosso aplicativo durante o seu desenvolvimento?

Assim, conhecemos a AVD (*Android Virtual Device*), ou Dispositivo Virtual Android. Com o nome já deu para ter uma noção do que se trata não é verdade! Ela é uma ferramenta presente no Android Studio, para emular um dispositivo Android em nosso sistema operacional Windows 10, essa ferramenta nada mais é que uma máquina virtual que roda nosso sistema operacional Linux o Android.

Também é permitido a execução do aplicativo em dispositivos físicos conectados ao Android Studio, por meio do cabo USB, porém esse dispositivo necessita que seu Android tenha o modo de desenvolvedor habilitado. Vamos aprender no futuro como fazer isso. Só que a AVD, torna mais flexível o processo de testes do aplicativo, pois podemos gerar um número infinito de equipamentos, e assim executar nossa aplicação em diversos tipos de equipamento, o que as vezes não seria possível com os equipamentos físicos de alto custo, como *Smartphone* de última geração. Os desenvolvedores iniciantes e empresas pequenas, não contam com recursos financeiros altos para compra de equipamentos, ainda mais para testes, não é verdade!

A figura 25, mostra a tela da AVD Manager, e seu botão "***Create Virtual Device...***", onde vamos criar o nosso primeiro dispositivo de testes para finalmente executar nosso primeiro projeto.

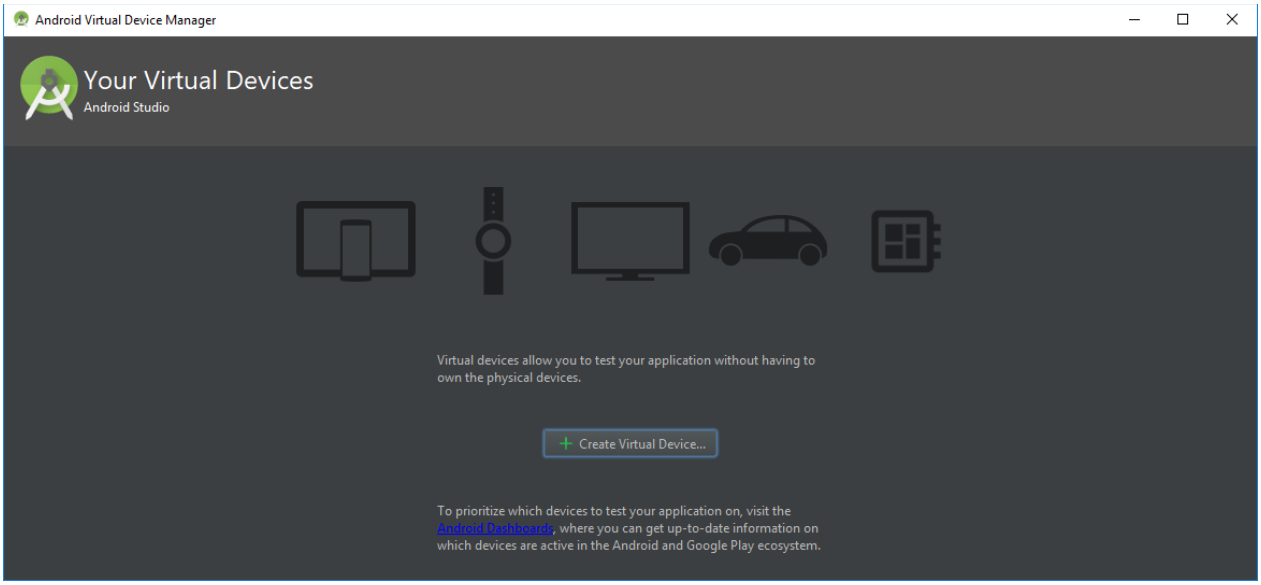


Figura 25 – AVD Manager do Android Studio.

Na primeira tela, vamos escolher o *Hardware* do nosso dispositivo. Contamos com uma infinidade de modelos e tamanhos de tela, para Smartphones, TV's, relógios e Tablets. E podemos customizar o nosso equipamento, ou seja, criar um *Hardware* diferente dos exibidos nas listas.

Como estamos desenvolvendo um aplicativo para Smartphone, vamos escolher um que já esteja disponível na lista, vamos trabalhar com o "***Nexus S***", selecione ele e clique em "***Next***", igual na figura 26.

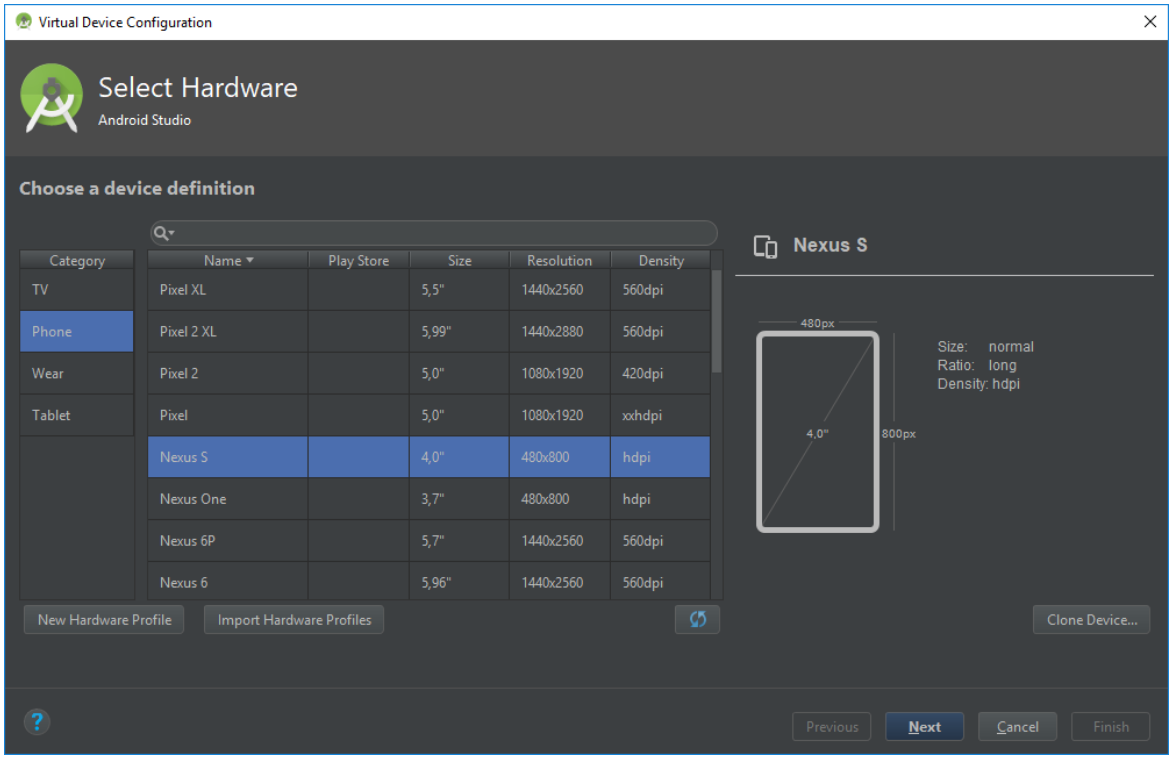


Figura 26 – Seleção do Hardware na AVD Manager.

Após escolher o *Hardware*, vamos escolher qual versão do sistema operacional Android vamos instalar nesse equipamento virtual.

Verifique que ainda não temos um sistema operacional, vamos ter que clicar em *Download* (Figura 27), para efetuar a instalação do sistema em nosso dispositivo. Como estamos trabalhando com a API 24 em nosso projeto, obrigatoriamente vamos ter que instalar o sistema Android 24 ou superior. Vamos escolher para nosso laboratório a versão Android 7.0 Nougat com nível 24 de API.

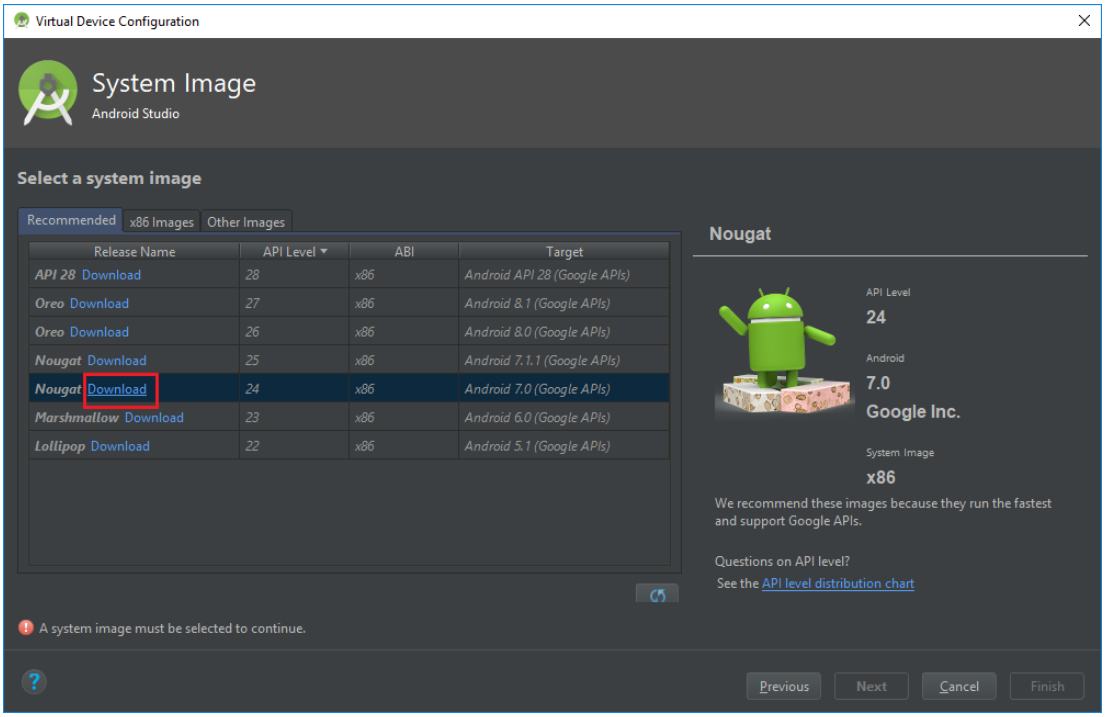


Figura 27 – *Download* do Android Nougat API 24 na AVD Manager.

Esta etapa requer um pouco mais de tempo, pois vamos efetuar o Download da imagem do Android e efetuar a sua instalação, e assim como em outras etapas do processo de instalação e configuração do Android Studio, um item que influencia é a sua velocidade de Internet contratada com seu provedor de acesso.

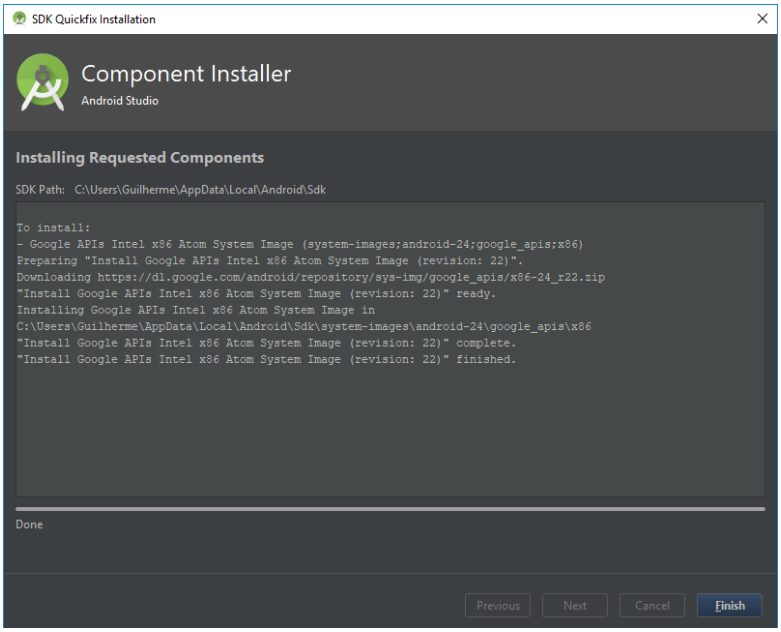
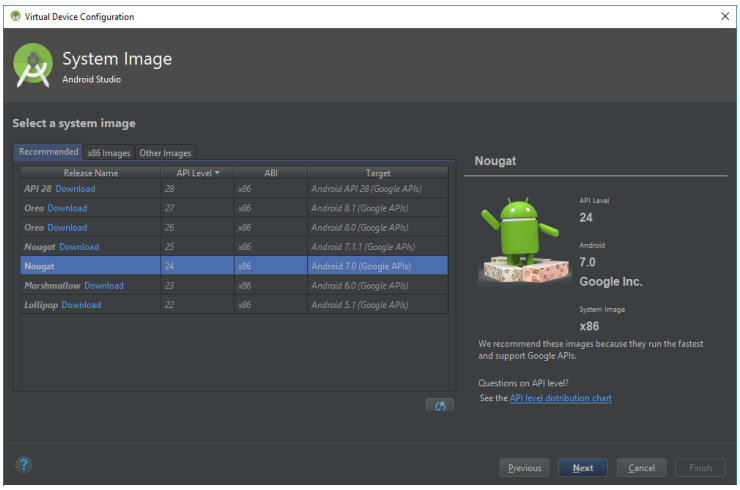


Figura 28 – Instalação completa do Android Nougat API 24 na AVD Manager.

Não se preocupe, todas essas etapas são necessárias na primeira execução do Android, pois digamos que ele veio sem esses recursos de fábrica! Depois, sua IDE já estará pronta para uso, e desenvolvimento de vários projetos, tenho certeza!

A figura 28, mostra a finalização da etapa de *Download* e instalação, vamos clicar em "***Finish***". A Figura também mostra que agora a versão 24 já está instalada, vamos selecionar ela, e clicar em "***Next***".



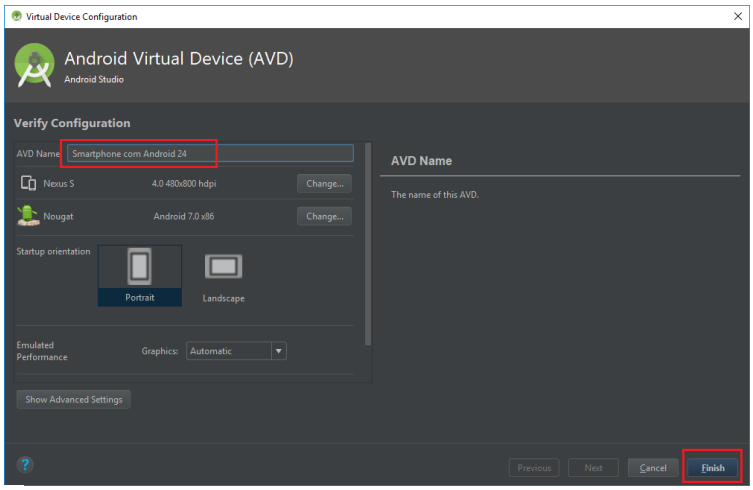


Figura 29 – Finalização do processo de criação da AVD.

Na janela a seguir, é a verificação final do equipamento virtual que vamos criar para teste, revise as configurações e principais características, e escolha um nome. Escolhemos o nome "Smartphone com Android 24" e vamos clicar em "***Finish***" (Figura 29).

Nossa AVD foi criada com sucesso! Para efetuar sua primeira execução é necessário instalar um último programa, para ajudar no processo de desempenho da nossa máquina virtual. O Intel® Hardware Accelerated Execution Manager ou Intel® HAXM. Ele pode ser adquirido através do site: https://software.intel.com/en-us/articles/intel-hardware-accelerated-execution-manager-intel-haxm.

Vamos descarregar o arquivo ZIP em nosso computador e efetuar descompactação e a sua instalação, clique no link conforme figura 30 para efetuar o *Download*.

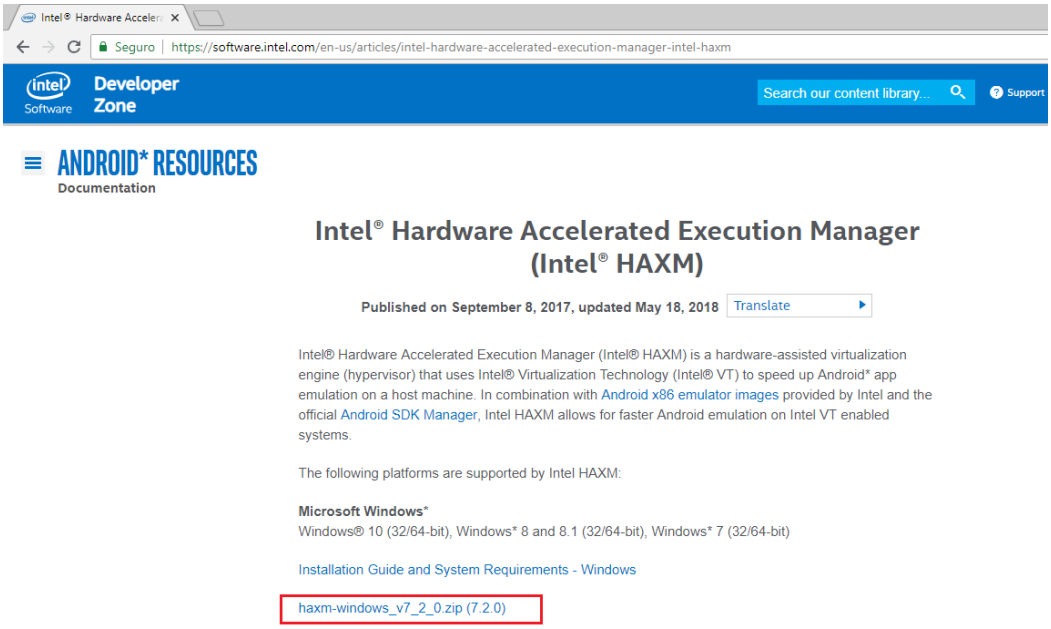


Figura 30 – Site da Intel HAXM.

Após o *Download,* descompacte o arquivo ZIP, e execute o arquivo de instalação igual a figura 31. Clique em "**Sim**" para a pergunta de segurança do Windows 10, e após abrir o assistente de instalação, clique em "***Install***".

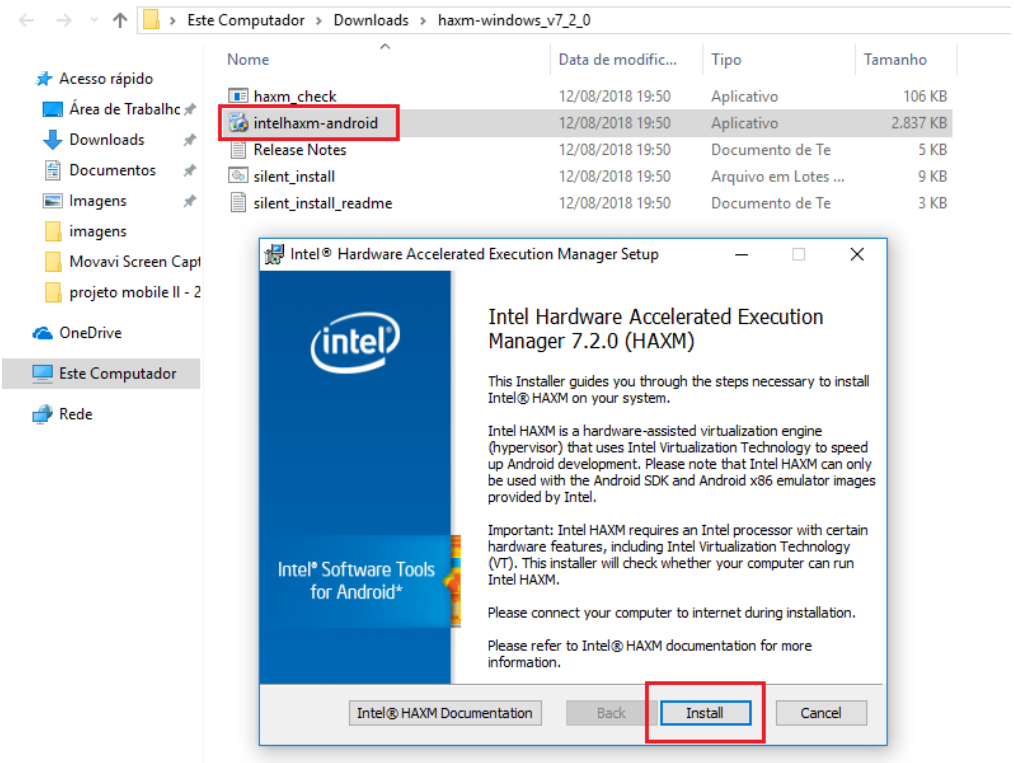


Figura 31 – Instalação do Intel HAXM.

Após o sucesso da instalação, desmarque a opção "***Launch Intel HAXM Documentation***" para não abrir a sua documentação e clique em "***Finish***"(Figura 32) e volte ao Android Studio.

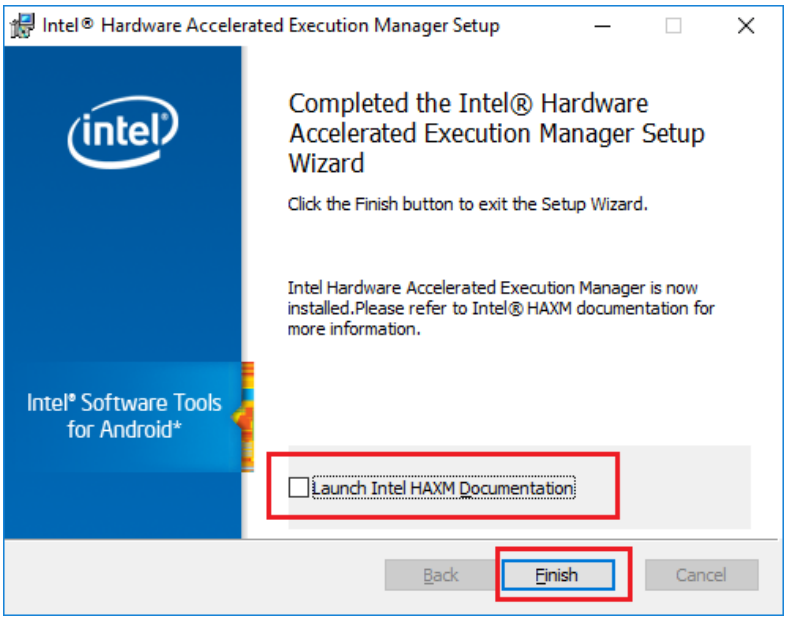


Figura 32 – Sucesso na instalação do Intel HAXM.

Agora podemos executar nosso dispositivo virtual, para verificar se ele está funcionando, clique em "***Launch this AVD in the emulator***" conforme figura 33. E aguarde até que seu dispositivo virtual seja executado e carregado em tela, igual a figura 34.

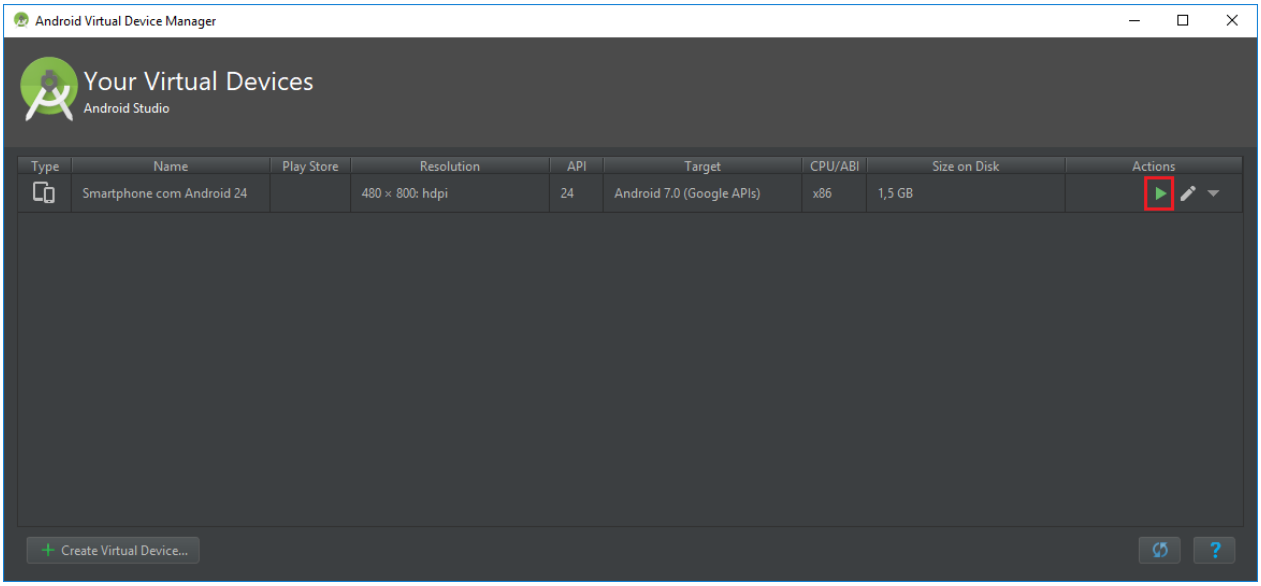


Figura 33 – Inicialização do nosso dispositivo virtual.

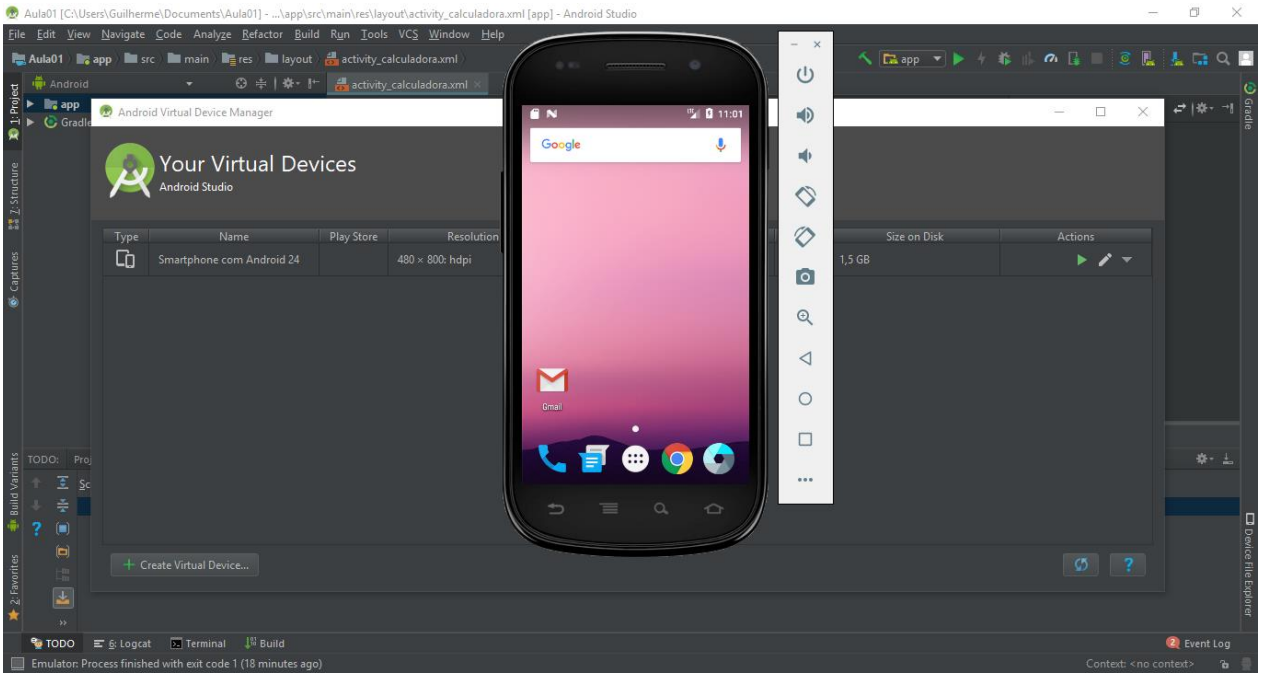


Figura 34 – AVD executada com sucesso na tela do nosso computador de laboratório.

Se você chegou até aqui, meus parabéns! Você e Marcelo estão progredindo juntos com o Android Studio! E agora a sua IDE já está pronta para encarar e executar nosso primeiro projeto, e muitos outros que estão por vir.

Caso não conseguiu efetuar o processo, revise as etapas e consulte o manual do Android Studio, disponível no seu site oficial: https://developer.android.com/studio/intro/
Uma outra dica é verificar se o seu computador possui o processador compatível com sistemas de virtualização, efetue uma pesquisa nos manuais do seu processador e placa mãe e verifique como ativar essa função.
Se mesmo verificando as dicas anteriores, você não conseguiu executar a AVD, uma última solução é executar os aplicativos em um aparelho Android, através do cabo USB e com o modo de desenvolvedor ativo. Vamos ensinar esse processo no decorrer deste material.

É interessante que a AVD fique sempre ligada, sendo minimizada quando não estiver sendo utilizada. Assim não vamos perder muito tempo, aguardando ela iniciar toda vez que executar nossa aplicação para teste.

Executando o projeto

Com o projeto criado e com a AVD em execução, podemos executar nosso projeto. Nossa *Activity* atual possui um *TextView* com o texto "Hello World". Vamos executar nosso aplicativo, para verificar o resultado e completar o teste.

Clique no botão "***Run app"*** da figura 35 e escolha a AVD que está aberta e clique em **"OK"** para que ela receba a instalação do aplicativo.

O processo de execução do aplicativo pode demorar alguns segundos, dependendo da configuração do seu computador. É importante você verificar na barra de status as etapas que estão sendo executadas pelo Android Studio. A figura 36 mostra a barra de status e uma barra de progressão em uma das etapas da execução do aplicativo.

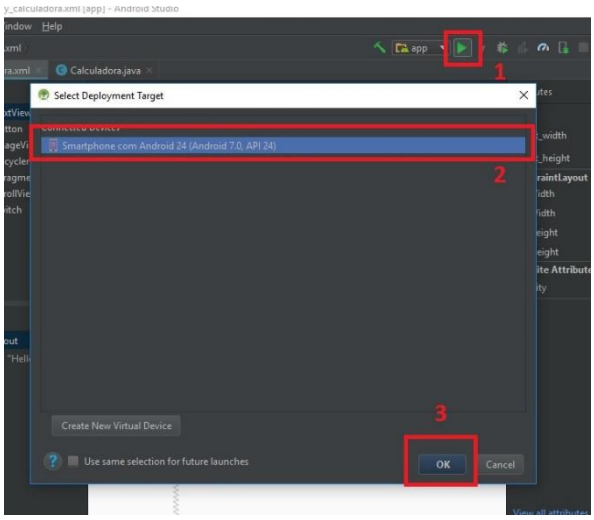


Figura 35 – Execução do aplicativo.

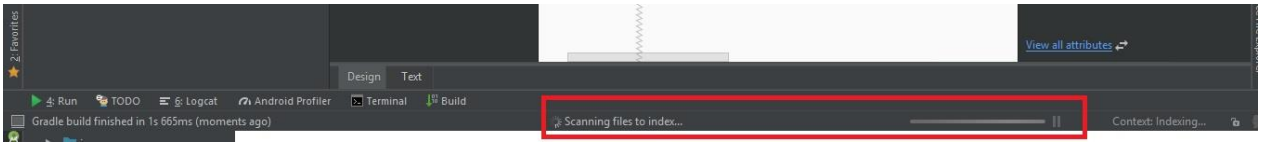


Figura 36 – Barra de Status do Android Studio.

Não é necessário configurar nada na AVD, após a execução do aplicativo, o mesmo aparece automaticamente na tela do nosso celular virtual, como demonstra a figura 37.

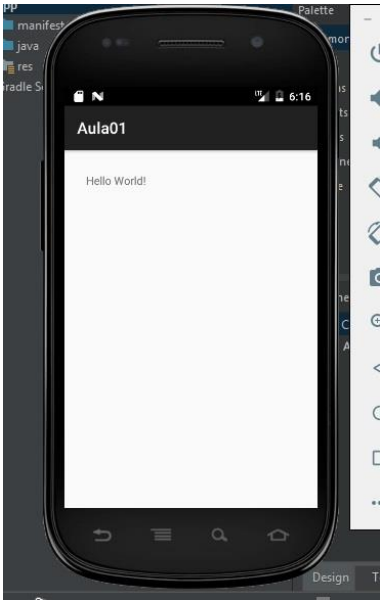


Figura 37 – AVD executando o aplicativo.

Desenvolvendo o projeto Calculadora de soma entre dois valores

Vamos seguir mergulhando nos estudos do Android Studio. Vamos pausar nossa execução do aplicativo através do botão *"Stop app"* igual a figura 38. Vale lembrar que não é necessário fechar a AVD, ela pode ficar em execução minimizada em seu computador, sem interferir no desenvolvimento do projeto.

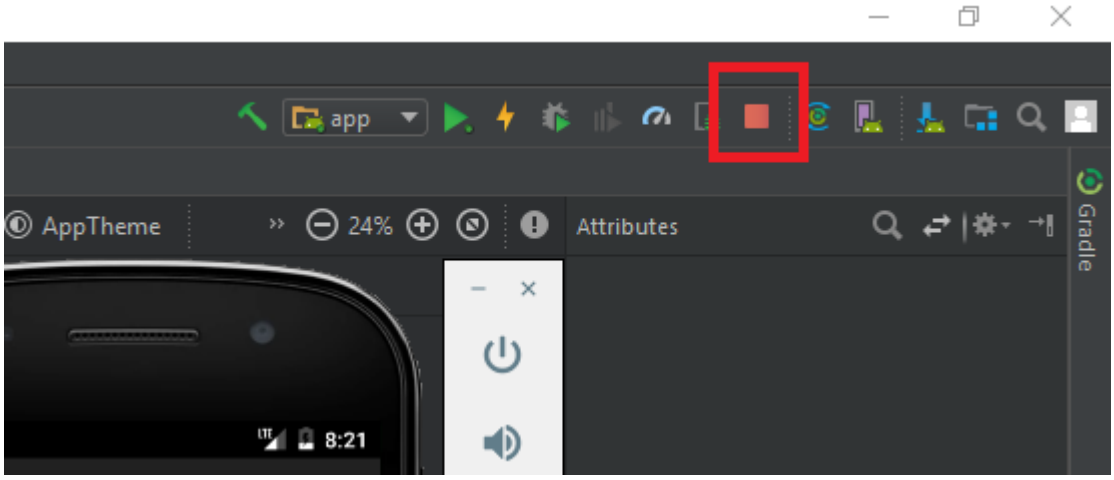


Figura 38 – Botão "Stop app".

Para desenvolver o projeto calculadora, é necessário popular nossa *Activity* com alguns componentes. Vamos utilizar os seguintes componentes para este projeto: *TextView, EditText e Button.* Cada componente com seus respectivos *ID e Text.*

O *ID* é o identificador do componente, ou seja, o nome dele. E este identificador é único em seu projeto, assim o Android Studio não permite a inserção de dois componentes com o mesmo *ID.* Ele é utilizado para "chamar" a utilização de um componente na programação Java do aplicativo.

A figura 39, mostra o *Layout* do aplicativo, a nossa interface gráfica com o usuário. Para desenvolver este trabalho assista ao vídeo a seguir que aborda a construção do *layout* da aplicação. Esta etapa utiliza o arquivo ***"activity _calculadora.xml".***

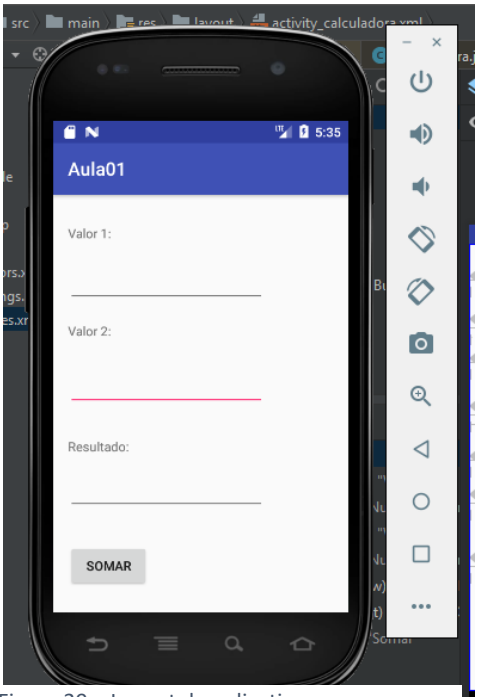


Figura 39 – Layout do aplicativo.

Após o desenvolvimento da *Activity,* é necessário efetuar a programação da classe Java, responsável pelas ações dos nossos componentes presentes na interface gráfica. O arquivo "***Calculadora.java***" contém a classe responsável pelas ações da nossa interface e de seus componentes. A figura 40 apresenta a codificação inicial da nossa classe.

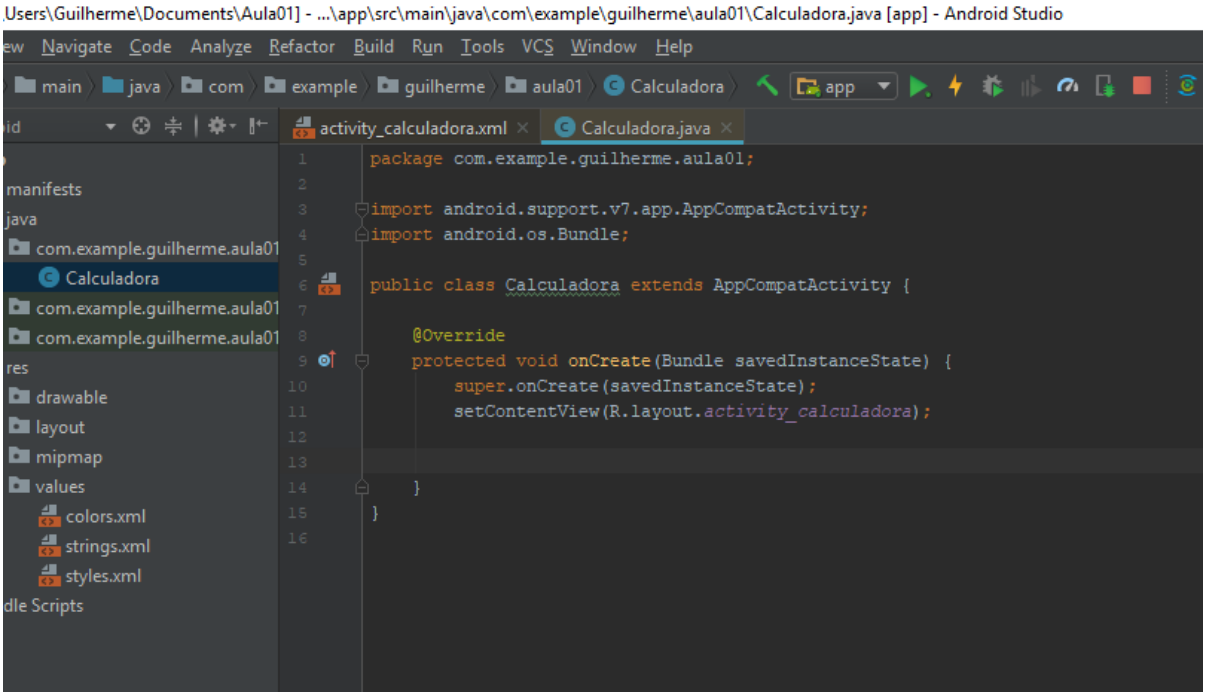


Figura 40 – Classe "Calculadora.java".

A primeira codificação realizada em nosso projeto, é inserir os *Import's* necessários para utilização de alguns componentes em nossa aplicação. Essa codificação é inserida em baixo do *Package* da nossa classe. A figura 41 exibe todos os *Import's* utilizados no projeto.

Agora, é necessário desenvolver uma espécie de ligação, entre nosso componente da interface gráfica com um componente criado na classe. Essa ligação é utilizada apenas nos componentes que demonstra uma interatividade com o usuário, ou seja, aqueles que tem uma comunicação diretamente com o usuário. Um exemplo é o *TextView* que recebe ou exibe um determinado valor para o usuário.

O código é realizado dentro do método ***onCreate***. A figura 41 mostra a implementação do código e os comentários em cada linha para aprendizagem das funções de cada um dos códigos presentes na classe.

O método ***onCreate*** é responsável basicamente pela criação e configuração da nossa *Activity* na execução do aplicativo. É nele que vamos implementar o método que captura o evento de clique no botão "Somar" da interface, e converte esse clique através do método ***onClick*** em ações.

```
package com.example.guilherme.aula01;

//Import's utilizados pelos componentes do projeto
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;

public class Calculadora extends AppCompatActivity {
    //Sobrecarga do método onCreate.
    //Bundle savedInstanceState: Recebe o estado da Activity.
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        //Utilização do onCreate da classe "AppCompatActivity", para que a base
        //da Activity seja criada.
        super.onCreate(savedInstanceState);
        //Configura o layout da interface XML definindo todos os componentes.
        setContentView(R.layout.activity_calculadora);

        //Ligação do componente da classe com os componentes da interface.
        Button btnSomarProg = (Button) findViewById(R.id.btnSomar);

        //Código responsável pela captura do evento de Clique no botão
        //e execução da ação realizada após o clique.
        btnSomarProg.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View arg0) {
                //Ligação dos componentes da classe com os componentes da interface.
                EditText edtValor1Prog = (EditText) findViewById(R.id.edtValor1);
                EditText edtValor2Prog = (EditText) findViewById(R.id.edtValor2);
                EditText edtResultadoProg = (EditText) findViewById(R.id.edtResultado);
            }
        });
    }
}
```

Figura 41 – Classe "Calculadora.java".

Para efetuar e mostrar a soma entre o valor 1 e o valor 2, é necessário efetuar uma conversão. O *editText* entrega os números na forma de caracteres, desta forma, é necessário converter para o tipo *Double.* Vamos utilizar também as variáveis para guardar nossos valores iniciais e o resultado, todas as três do tipo *Double*. Após a leitura e armazenamento dos valores, efetuamos a soma, e na sequência voltamos o resultado para nossa interface. Esse retorno é utilizado uma conversão, já que nosso resultado se encontra no formato *Double*, e nosso *editText* recebe apenas valores do tipo texto (*String*). A figura 42 mostra o código final como deve ficar, e após a sua implementação podemos executar e testar nosso aplicativo na AVD.

```
package com.example.guilherme.aula01;

//Import's utilizados pelos componentes do projeto
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;

public class Calculadora extends AppCompatActivity {
    //Sobrecarga do método onCreate.
    //Bundle savedInstanceState: Recebe o estado da Activity.
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        //Utilização do onCreate da classe "AppCompatActivity", para que a base
        //da Activity seja criada.
        super.onCreate(savedInstanceState);
        //Configura o layout da interface XML definindo todos os componentes.
        setContentView(R.layout.activity_calculadora);

        //Ligação do componente da classe com os componentes da interface.
        Button btnSomarProg = (Button) findViewById(R.id.btnSomar);

        //Método responsável pela captura do evento de Clique no botão
        //e execução da ação realizada após o clique.
        btnSomarProg.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View arg0) {
                //Ligação dos componentes da classe com os componentes da interface.
                EditText edtValor1Prog = (EditText) findViewById(R.id.edtValor1);
                EditText edtValor2Prog = (EditText) findViewById(R.id.edtValor2);
                EditText edtResultadoProg = (EditText) findViewById(R.id.edtResultado);

                //Conversão e armazenamento dos caracteres obtidos na interface
                //gráfica.
                double num1 = Double.parseDouble(edtValor1Prog.getText().toString());
                double num2 = Double.parseDouble(edtValor2Prog.getText().toString());
                //Soma e armazenamento do resultado.
                double resultado = num1 + num2;
                //Retorno para a interface gráfica do resultado.
                edtResultadoProg.setText(String.valueOf(resultado));
            }
        });
    }
}
```

Figura 42 – Código final da classe Calculadora.

A figura 43 mostra o resultado obtido com nosso aplicativo, sendo executado na AVD.

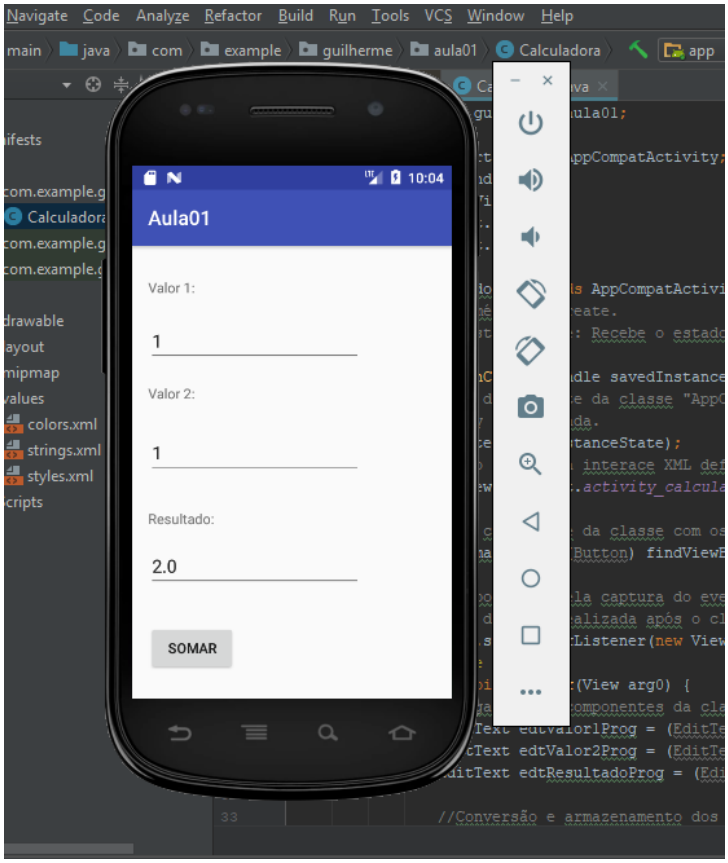


Figura 43 – AVD executando o projeto final.

Acompanhe a “Atividade Online” para colocar em prática todo o conteúdo da Agenda 1.