

Roteiro de Laboratório 01

José Cassiano Gunji

1. Introdução ao Android

Tradicionalmente, tecnologias de desenvolvimento como o Java são empregadas para o desenvolvimento de sistemas de diversos tamanhos para serem empregados em microcomputadores ou equipamentos maiores. Até poucas décadas atrás, era necessário empregar tecnologias específicas para se desenvolver *software* para sistemas móveis ou embarcados, como dispositivos portáteis (telefones e agendas eletrônicas) e centrais de controle (controle de ignição de automóveis, módulos de automação industrial, entre outros). Em Java, podia-se usar a distribuição Java ME, como discutido na Unidade I. Afinal, era necessária uma distribuição específica do Java para se adequar às limitações do hardware móvel ou embarcado (pouca memória RAM, pouco espaço de armazenamento e baixo poder de processamento).

Lembrete:

Você pode conhecer e baixar as diversas distribuições do Java e ferramentas oficiais no portal oficial da Oracle: <https://www.oracle.com/java/technologies/>

Com o progresso da tecnologia desses dispositivos, hoje já não é necessário utilizar tecnologias de desenvolvimento de *software* específicas. A maioria dos *smartphones* atuais é várias vezes mais poderosa em todos os aspectos do que os primeiros microcomputadores que executavam Java SE. Assim, usamos o próprio Java SE em dispositivos móveis.

Observação:

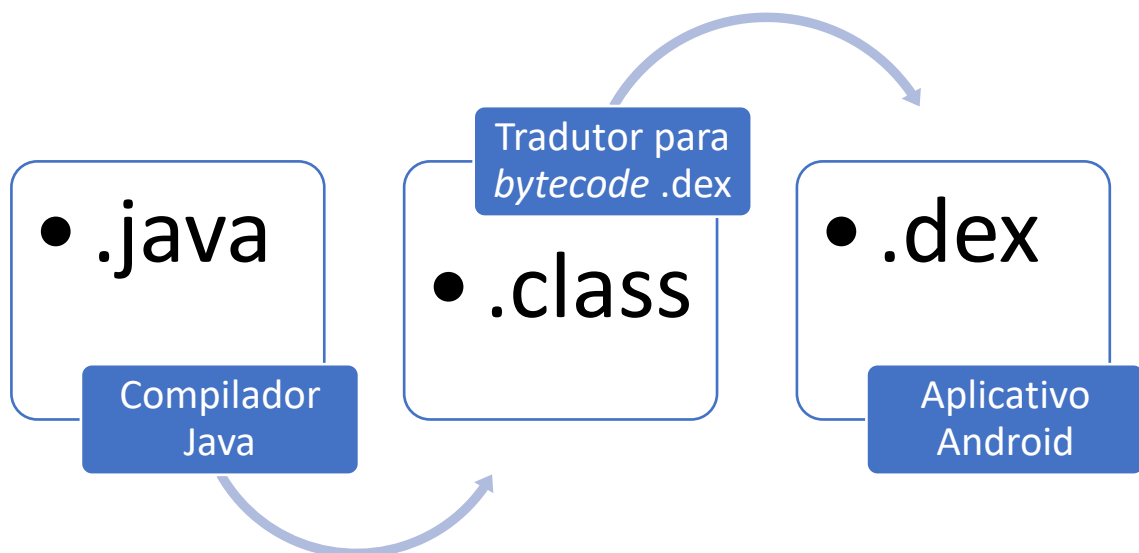
Apesar do Java ME não ser mais necessário para dispositivos móveis, ele continua disponível, principalmente para dar alicerce a outras distribuições para aplicações que ainda apresentam restrições de capacidade e desempenho, como *smart cards* e aplicações multimídia, como discos Blu-Ray.

Uma das principais tecnologias de desenvolvimento para dispositivos móveis é o Android. O Android é primariamente um Sistema Operacional desenvolvido pelo Google para ser empregado em dispositivos móveis, tipicamente *smartphones*, *tablets*, *smartwatches*, *smart-TVs*, centrais multimídia de automóveis entre outras aplicações que surgem constantemente.

2. Arquitetura Android

O Android é uma bifurcação (*fork*) do Sistema Operacional Linux que emprega como principal tecnologia de programação uma bifurcação (*fork*) do Java SE. Entretanto, o S.O. não é equipado com uma máquina virtual Java tradicional, principalmente porque elas não são desenvolvidas para utilizar a tela de toque como principal meio de entrada do usuário, nem os diversos sensores disponíveis na maioria dos dispositivos móveis. Em seu lugar, o Android emprega uma máquina virtual própria, chamada **Dalvik Virtual Machine**. A Máquina Virtual Dalvik não executa *bytecodes* convencionais do Java (extensão *.class*). Ela executa *bytecodes* de extensão *.dex* (*Dalvik Executable*). Assim, o processo de compilação para Android possui um passo a mais, a tradução do *bytecode* Java para *bytecode* Dalvik.

Figura 1: Processo de compilação de aplicativos Android.



Fonte: O autor.

Para se desenvolver aplicativos para Android com Java, rigorosamente falando, são necessários apenas os seguintes kits de desenvolvimento:

- Java SE SDK;
- Android SDK.

Tendo os dois kits de desenvolvimento acima, já é possível desenvolver aplicativos Android usando um editor de arquivos-texto e o compilador por linha de comando. Entretanto, como a estrutura de um aplicativo Android é razoavelmente complexa, vale muito a pena utilizar uma IDE. Originalmente, o Google, atual desenvolvedor do Android, disponibilizou um plugin para a IDE Eclipse, o ADT (Android Development Tools – Ferramentas de Desenvolvimento Android). Logo a comunidade de desenvolvimento adotou esta solução. Mas em

junho de 2015, o Google anunciou o fim do suporte ao ADT, sendo substituído pela sua IDE própria, especializada em desenvolvimento para esta plataforma, o Android Studio. Assim, vamos utilizar esta IDE.

Observação:

O Android Studio é a ferramenta oficial do Google e, provavelmente, a mais completa para se desenvolver aplicativos para Android, mas está longe de ser a única. Para aplicações ou desafios de desenvolvimento específicos, existem alternativas ao Android Studio. Algumas podem ser vistas nos seguintes endereços:

<https://www.businessofapps.com/marketplace/app-development-software/research/android-development-tools/>

<https://www.mockplus.com/blog/post/android-developer-tool>

<https://www.androidauthority.com/best-android-developer-tools-671650/>

O Android Studio é distribuído gratuitamente (em código aberto) pelo Google em diversas versões a partir de sua página <http://developer.android.com/studio>. Em sua página de downloads, pode-se baixar o SDK para as plataformas Windows, Linux, Mac OS e Chrome OS.

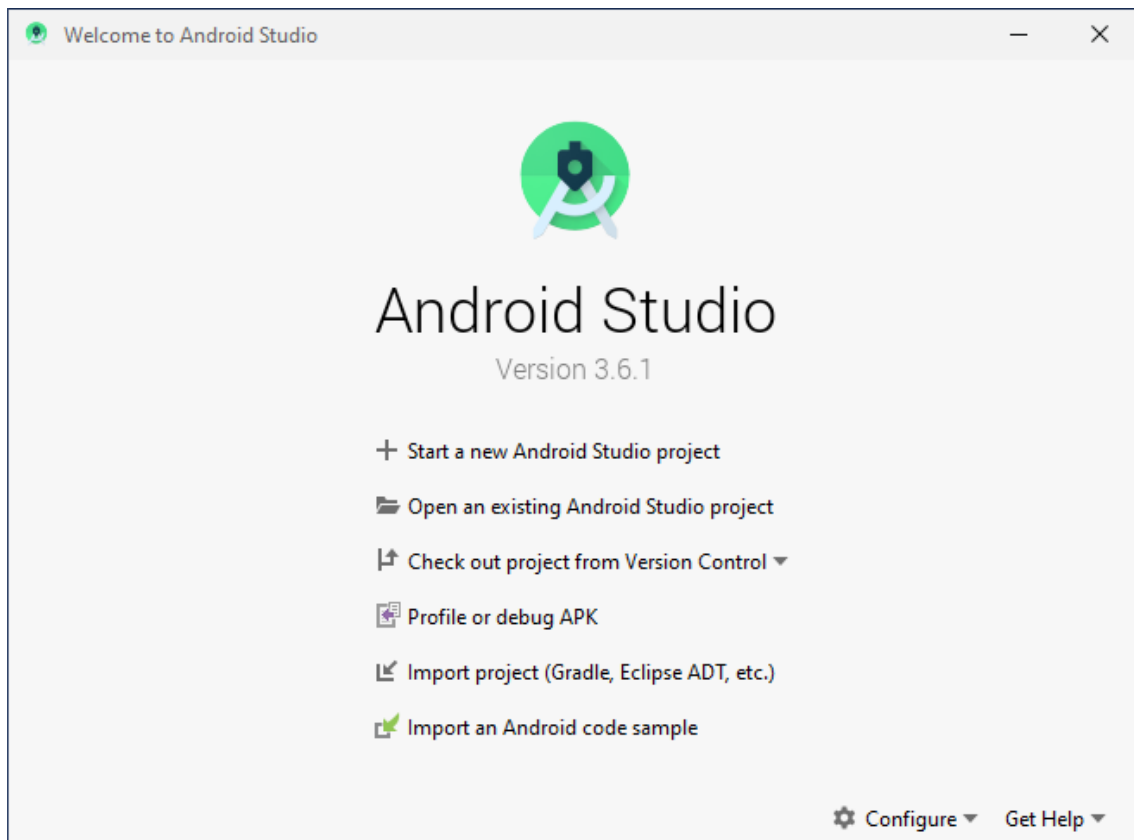
Observação:

Também é possível programar em C++ e, a partir da versão 3.0 do Android Studio, programar também em Kotlin. Por enquanto e, provavelmente, por algum tempo ainda, a principal linguagem de desenvolvimento para Android continua sendo o Java.

3. Instalando e Utilizando o Android Studio

Escolha a versão correta do Android Studio para seu sistema operacional, faça o download e instale-o, seguindo as instruções apresentadas na tela. Você provavelmente não irá precisar se desviar das opções padrão do processo de instalação. Quando a instalação estiver completa, será apresentada a tela de boas-vindas, semelhante à figura abaixo.

Figura 2: Tela de boas vindas do Android Studio.



Fonte: O autor.

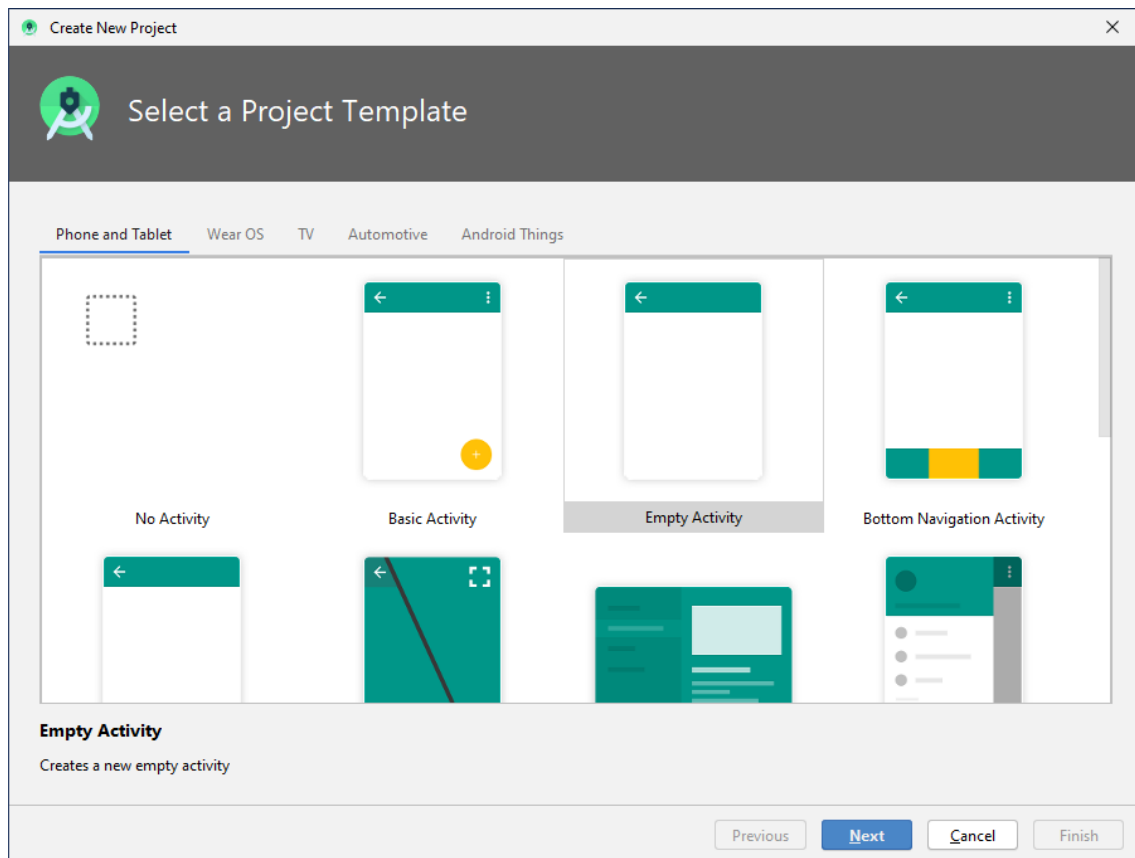
Observação:

O ritmo com o qual as tecnologias relacionadas aos dispositivos móveis (e de computação em geral) evolui implica que, quase certamente, a sua versão do Android Studio será mais nova do que a usada ao elaborar este livro-texto. Não se preocupe com isso. Use sempre a versão mais nova disponível. Quase sempre as mudanças de uma versão para outra envolvem aspectos mais avançados da plataforma. Em geral, estes aspectos não são abordados em textos introdutórios, como este. Quando você estiver avançado no assunto o suficiente para essas diferenças se tornarem relevantes, você já estará fluente na tecnologia o suficiente para perceber sozinho as mudanças e como lidar com elas.

3.1. Aplicativo “Olá mundo!”

Selecione a opção “Start a new Android Studio Project”. Neste momento, o Android Studio apresenta uma série de *templates* de aplicativos para diversas modalidades de dispositivos compatíveis.

Figura 3: Opções de *templates* de aplicativos Android.



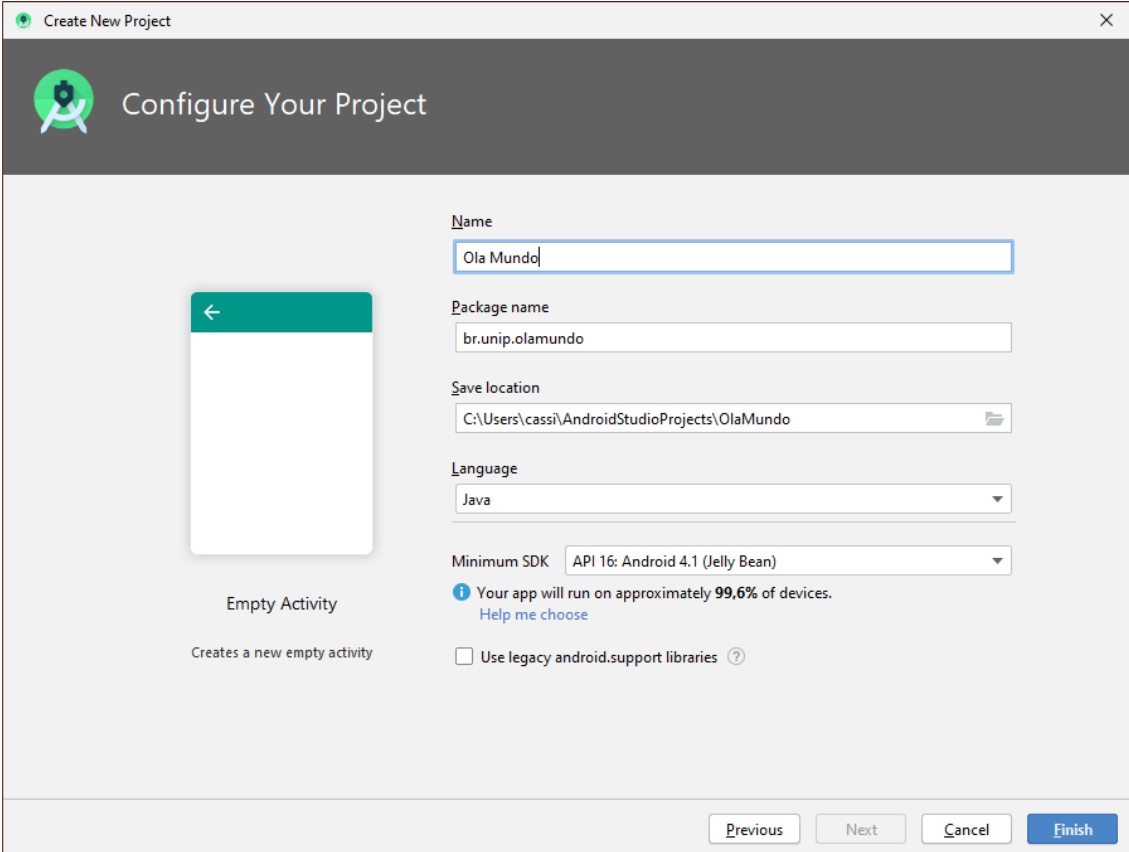
Fonte: O autor.

Aplicativos Android são organizados em quatro principais elementos: *Activity*, *Intent*, *Broadcast Receivers* e *Services*.

- **Activity:** É o principal elemento de interação de um aplicativo. Cada atividade costuma modelar uma interface gráfica do aplicativo.
- **Intent:** É um componente que permite que o aplicativo se comunique com o sistema operacional e outros aplicativos. Por exemplo, o aplicativo pode se comunicar com a câmera, o receptor GPS, o aplicativo de mapas, etc.
- **Broadcast Receivers:** Com este componente, o aplicativo pode receber informações transmitidas pelo sistema operacional ou outros aplicativos. Por exemplo, o S.O. pode informar que a bateria está em nível crítico, que a conexão 3G foi estabelecida, que os fones de ouvido foram conectados, etc.
- **Services:** Com este componente o aplicativo pode continuar executando tarefas mesmo que não esteja mais ativo em primeiro plano.

Aproveite este momento e explore os tipos de dispositivos e os *templates* de *activities* para cada um. Quando estiver satisfeito, escolha em “*Phone and Tablet*” o *template* “*Empty Activity*”. Clique em “*Next*”.

Figura 4: Configurações de um novo projeto Android.



The screenshot shows the 'Create New Project' dialog in Android Studio. The dialog is titled 'Configure Your Project' and has a green header bar. On the left, there is a preview of the 'Empty Activity' template, which is a white card with a green header bar and a back arrow. Below the preview, it says 'Empty Activity' and 'Creates a new empty activity'. On the right, there are several input fields and a dropdown menu. The 'Name' field is filled with 'Ola Mundo'. The 'Package name' field is filled with 'br.unip.olamundo'. The 'Save location' field is filled with 'C:\Users\cassi\AndroidStudioProjects\OlaMundo'. The 'Language' dropdown is set to 'Java'. The 'Minimum SDK' dropdown is set to 'API 16: Android 4.1 (Jelly Bean)'. Below these fields, there is a blue information icon and a message: 'Your app will run on approximately 99,6% of devices.' with a link 'Help me choose'. There is also a checkbox for 'Use legacy android.support libraries' which is unchecked. At the bottom, there are four buttons: 'Previous', 'Next', 'Cancel', and 'Finish'.

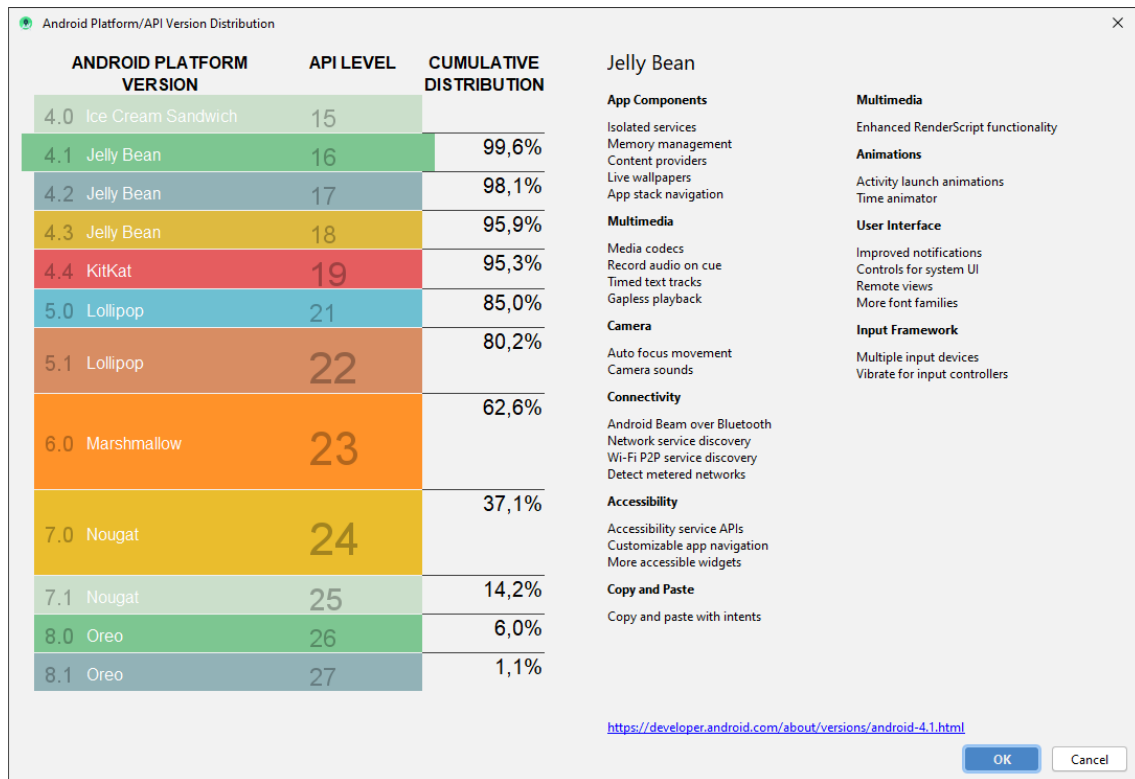
Fonte: O autor.

Nas configurações de seu projeto que surgem em seguida, defina o nome de seu projeto como “Ola Mundo”. Não use acentos ou caracteres especiais. Defina o nome do pacote (“*Package name*”) para “br.unip.olamundo”. Na vida real, costuma-se definir o nome do pacote como o endereço de Internet da empresa desenvolvedora (ao contrário), como fizemos acima. Altere também a linguagem do projeto para “Java”.

Uma das últimas opções que você tem para escolher é o “*Minimum SDK*”, que significa qual será a versão mínima necessária do Sistema Operacional Android para que ele consiga executar o seu projeto. Nesta opção é importante selecionar uma opção de compatibilidade de maximize a quantidade de dispositivos que serão compatíveis com o seu aplicativo. Quanto mais antiga a API, menos recursos estarão disponíveis. Por outro lado, maior será o número de dispositivos compatíveis. Com a seleção da API 16 (Android 4.1 Jelli Bean), o próprio assistente indica que 99,6% dos dispositivos ativos na loja GooglePlay poderão utilizar o seu aplicativo. Clicando no link “Help me choose” é apresentado um gráfico com as taxas de utilização de cada nível de API e quais as suas principais inovações. Este gráfico é atualizado em tempo real e mostra

quais são os recursos adicionados por cada nível de API em relação ao nível anterior.

Figura 5: Quadro comparativo entre os diversos níveis de API.



Fonte: O autor.

O nível de API é um termo técnico adotado para simplificar a vida do desenvolvedor. Eles são sempre números inteiro incrementados de um em um. Com isso, o desenvolvedor não precisa se lembrar dos nomes comerciais de cada versão de SDK (*Jelly Bean*, *KitKat*, *Lollipop*, etc.) muito menos da versão de cada plataforma.

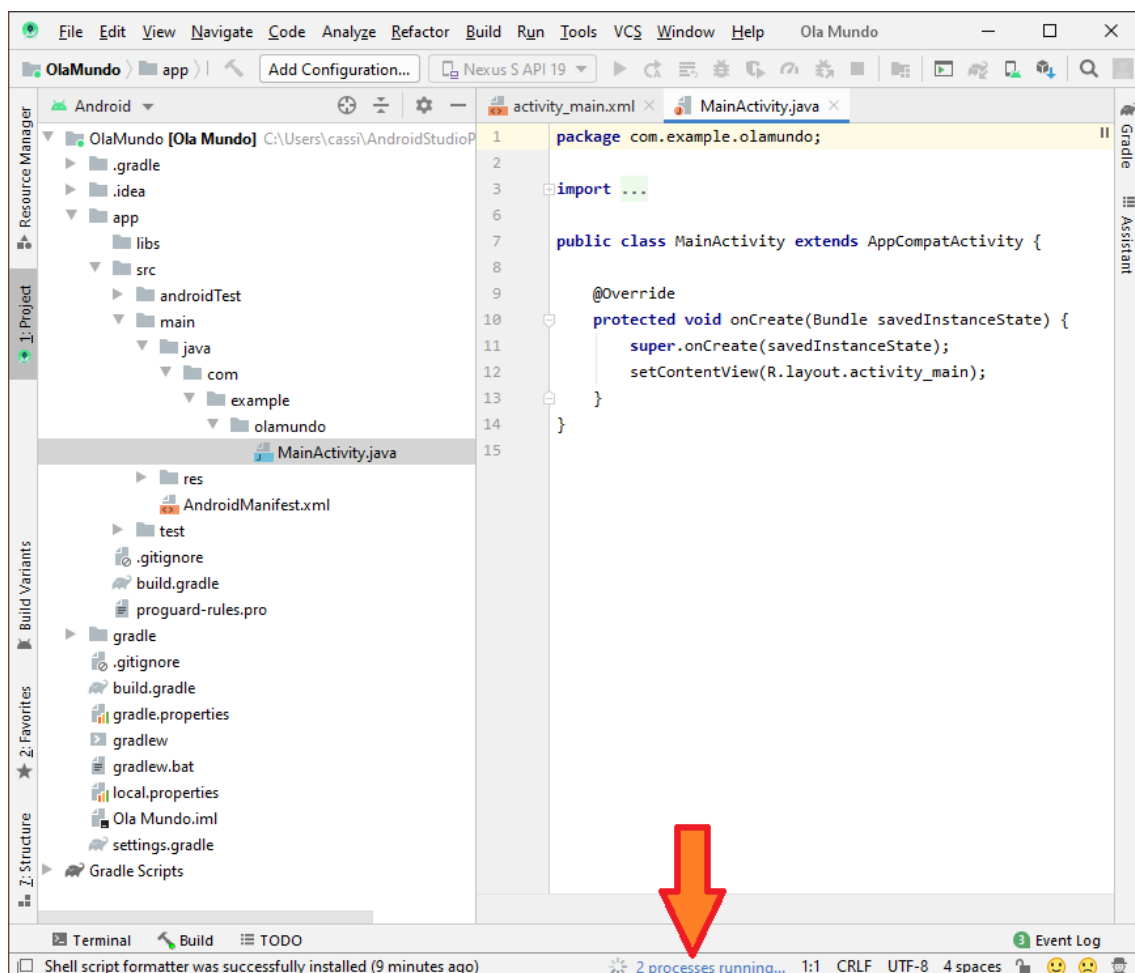
Lembrete:

O sistema operacional Android foi desenvolvido para ser compatível tanto com versões anteriores quanto futuras. Isso significa que, ao definir a versão mínima do Android para seu aplicativo para, por exemplo, nível de API 15 (Ice Cream Sandwich), você garante que seu aplicativo poderá ser executado por qualquer aparelho com um Android do mesmo nível de API ou superior, inclusive os futuros. Não importa se seu aplicativo foi compilado usando um SDK de nível de API 25, ele poderá ser executado por um Android de nível 26 ou superior, mesmo que ainda não disponível. Qualquer aplicativo sempre será compatível com um Android de nível de API mais recente do que aquele em que foi compilado.

Ao clicar em “*Finish*”, o Android Studio cria seu projeto e inicia um processo de preparo automatizado de seu projeto. Este processo costuma ser

um tanto demorado, principalmente se o computador que esteja usando seja limitado em memória RAM, se ele usa disco rígido ao invés de um SSD e se o processador for particularmente antigo. Este processo é realizado principalmente por uma ferramenta chamada *Gradle*, que é uma ferramenta de automação de projetos. Com ela, o Android Studio prepara, entre outras coisas, a estrutura de arquivos do projeto e uma série de arquivos necessários a ele e que não devem ser editados manualmente pelo desenvolvedor. Ainda mais, se esta for a primeira vez que esteja executando o Android Studio, ou se ele foi recentemente atualizado, ou ainda se os *scripts Gradle* foram atualizados, muita coisa será baixada em segundo plano. Tenha paciência. Observe a barra de *status* na parte de baixo da janela do Android Studio. Seu projeto estará pronto quando não houver qualquer “ampulheta” girando e nenhuma barra de progresso avançando. A figura a seguir mostra um exemplo de processo sendo executado em segundo plano.

Figura 6: Exemplo de processos sendo executados em segundo plano.

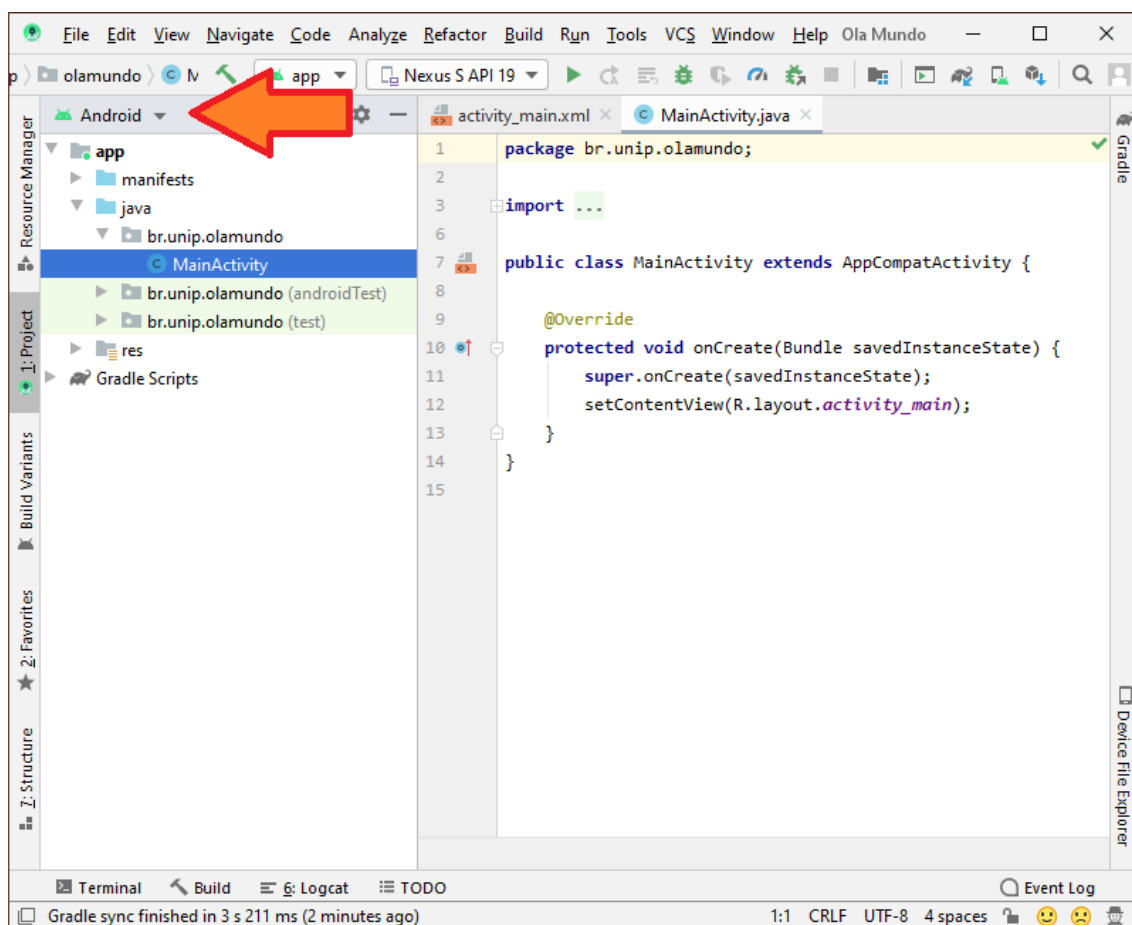


Fonte: O autor.

Não se preocupe. Nas próximas vezes que abrir ou criar um novo projeto, o processo será bem mais rápido.

Quando o projeto estiver terminado, você deve ter uma janela semelhante à mostrada na figura a seguir.

Figura 7: Android Studio pronto para ser usado.



Fonte: O autor.

No painel à direita é apresentada a área de edição do elemento selecionado. No painel à esquerda é apresentada a estrutura do projeto. Esta estrutura pode ser configurada de acordo com diversos padrões. O padrão Android selecionado é obviamente o mais apropriado no momento, mas é um padrão diferente do padrão de um aplicativo Java. Você pode selecionar outro padrão, como o de aplicativo Java, para reconhecer a estrutura do sistema, semelhante à estrutura mostrada pelo Eclipse, mas retorne ao padrão Android antes de prosseguir.

No pacote “br.unip.olamundo” estarão as classes e interfaces criadas para o projeto. Este pacote está na pasta “Java”.

Na pasta “res” (resource – recurso) estão os recursos do aplicativo. Em “drawable” estarão as imagens utilizadas no aplicativo. Em “layout” os arquivos XML que definem as interfaces gráficas. Aqui já está o arquivo de configuração da atividade “MainActivity” criada automaticamente pelo assistente. Na pasta “mipmap” estão arquivos de ícones em diversas resoluções para serem usados dependendo da resolução do dispositivo e do local onde o ícone será exibido (na

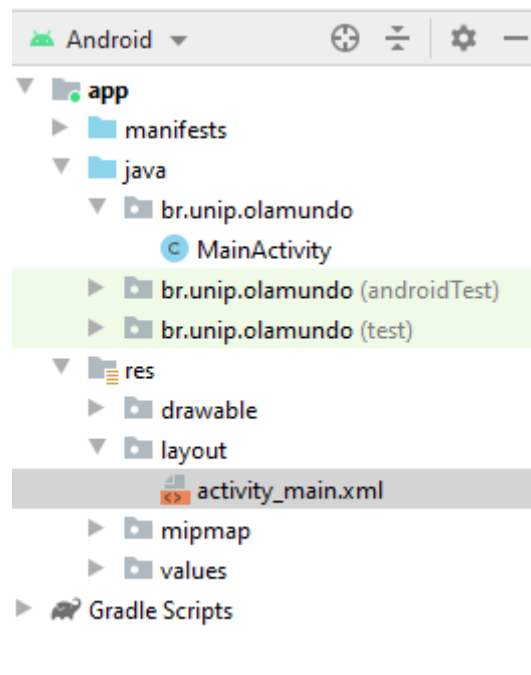
gaveta de aplicativos, na loja, nos destaques da loja, etc.). Nas pasta “*values*” estão arquivos de configuração XML para diversos aspectos do aplicativo, como suas cores, seus *strings* (pode haver um conjunto de *strings* para cada língua suportada pelo aplicativo) entre outros. Isto facilita bastante o trabalho de traduzir seu aplicativo para outras línguas. Essa tradução pode ser feita contratando-se tradutores profissionais.

Observação:

Houve um tempo em que era possível usar o Tradutor do Google de dentro do próprio Android Studio para gerar versões localizadas de um aplicativo. Mas, devido aos resultados ruins desta prática, o Google removeu este recurso.

Se a aba “*activity_main.xml*” não estiver aberta, abra-a clicando duas vezes nela no navegador de projeto mostrado abaixo.

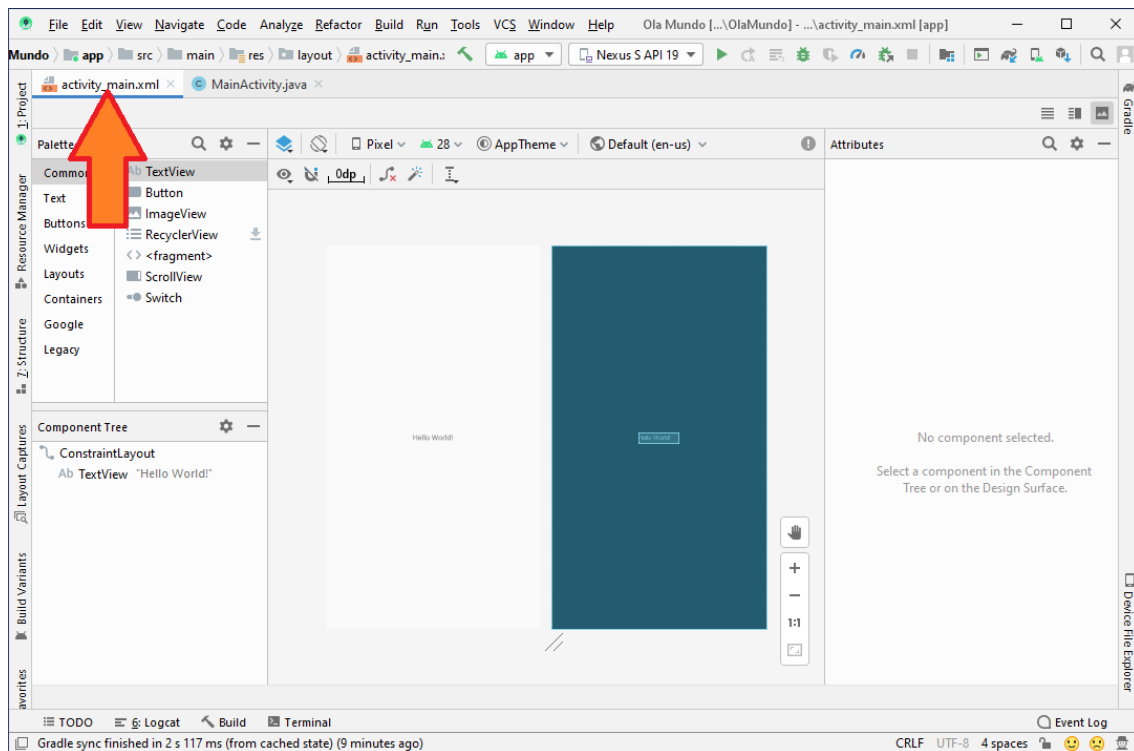
Figura 8: Atividade principal no navegador de projeto.



Fonte: O autor.

Se a janela de seu Android Studio ainda não estiver maximizada, maximize-a agora. Dependendo das configurações de seu computador, pode ser interessante ocultar as janelas de ferramentas que não esteja usando. Neste caso, evite fechar as janelas diretamente, pois você pode ter dificuldades em trazê-las de volta mais tarde. Em vez disso, dê um clique duplo na aba da janela que deseja maximizar. Para trazer as demais janelas de volta, dê um clique duplo novamente na aba da janela.

Figura 9: Maximizando a janela de edição.



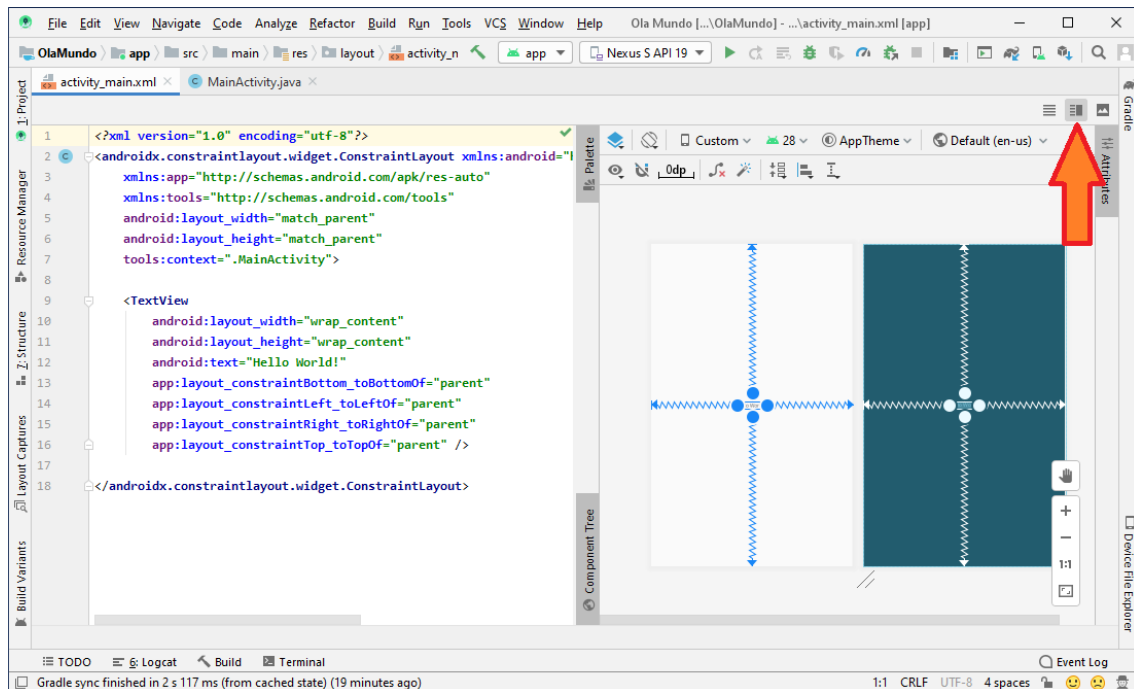
Fonte: O autor.

Lembrete:

Caso você tenha fechado algumas janelas e não saiba exatamente como trazê-las de volta, você pode usar o comando de menu *[Window] -> [Restore Default Layout]*. Este comando retorna a IDE a um modo padrão de exibição.

O leiaute de sua aplicação pode ser editado tanto pelo editor de leiaute quanto pelo editor de código XML. Você pode mudar o modo de edição clicando nos botões mostrados na figura abaixo.

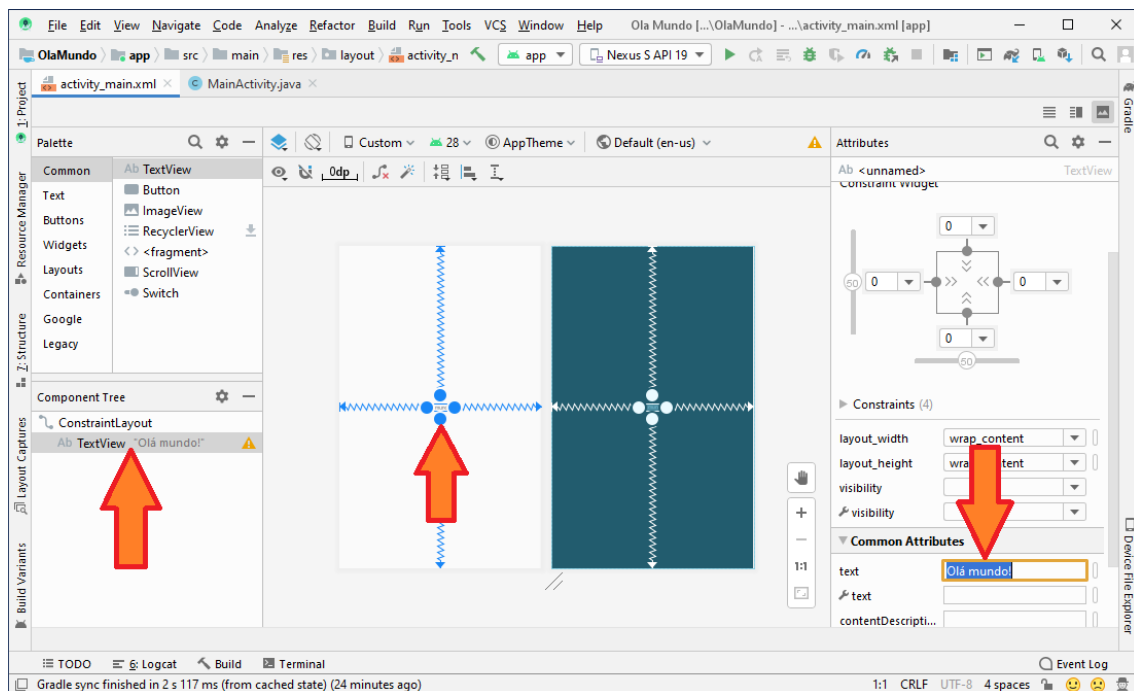
Figura 10: Escolhendo a exibição de leiaute ou de código.



Fonte: O autor.

Escolha a visualização de leiaute novamente. Na árvore de componentes ou na visualização do leiaute, clique no componente `TextView`. A seguir, na janela de atributos à direita, altere o valor do atributo `text` para “Olá mundo!”. Desta vez, por se tratar de um texto que será apresentado na interface gráfica, o texto pode conter qualquer caractere especial e acentuação.

Figura 11: Alterando o texto do *TextView*.



Fonte: O autor.

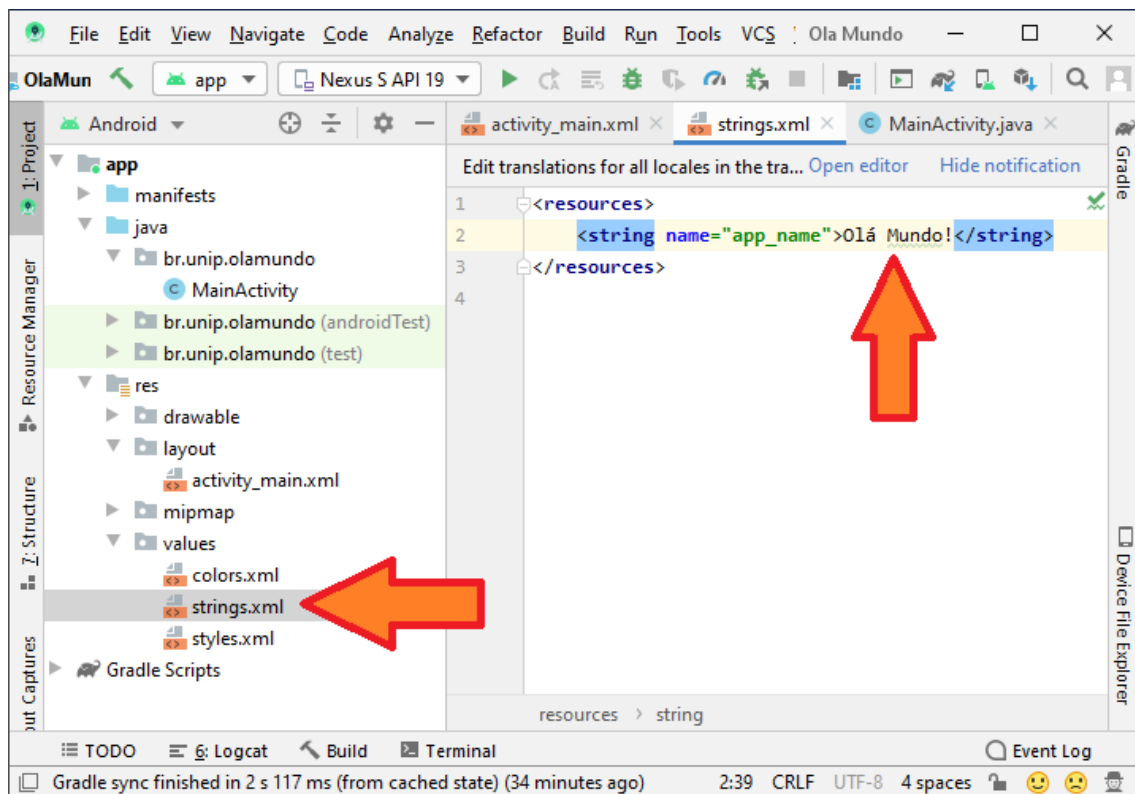
Observação:

O atributo *text* indica qual o texto que será apresentado pelo *TextView* no aplicativo quando ele for iniciado.

Já o atributo *text* marcado com uma chave de porca (logo abaixo) indica que é um atributo de ferramenta de edição. Isso significa que o valor que for inserido nele será apresentado apenas no visualizador em tempo de edição, mas não no aplicativo final. Os atributos de ferramenta são úteis quando um valor é preenchido apenas em tempo de execução por código mas, para facilitar o projeto visual em tempo de projeto, o valor do atributo de ferramenta é utilizado, permitindo uma pré-visualização do atributo.

Quando criamos o aplicativo, não podíamos usar caracteres especiais nem acentos. Entretanto, queremos que nosso aplicativo seja apresentado em português correto na gaveta de aplicativos. Para tanto, vamos alterar o texto de apresentação do aplicativo. Abra o arquivo de *strings* na pasta *values* e modifique o nome do aplicativo.

Figura 12: Corrigindo o nome do aplicativo.



Fonte: O autor.

Observação:

É uma boa prática de programação definir todos os textos usados no aplicativo em um arquivo separado XML. Assim, a manutenção e a tradução do aplicativo para outras linguagens pode ser feita por pessoas sem conhecimento de programação, apenas editando o arquivo XML. O mesmo é válido para aplicativos *desktop* em Java, .Net, C++, etc. No próximo exemplo iremos adotar esta prática.

3.2.Criação e Configuração de um Emulador Android

É óbvio que não podemos executar um aplicativo Android em um computador pessoal. Os sistemas operacionais são diferentes e incompatíveis. Para testar nosso aplicativo, temos duas opções: utilizar um dispositivo Android real conectado ao computador de desenvolvimento por um cabo USB ou utilizar um dispositivo emulado no próprio computador de desenvolvimento. Ambas as modalidades de execução possuem vantagens e desvantagens.

Para se executar o aplicativo em um dispositivo real, primeiro é necessário que ele esteja preparado para tanto. Para fazer isso, habilite as opções de desenvolvedor. O procedimento exato para tanto muda de versão para versão de Android assim como de fabricante para fabricante de dispositivos. Em geral, o processo inicia no menu de configurações do dispositivo, na categoria "Sobre". Procure alguma informação sobre o Número da Versão do seu aparelho. Clique-

a diversas vezes rapidamente. Após ao menos sete cliques, você receberá a mensagem “Você é um desenvolvedor”. Se o Número de Versão de seu aparelho não surtir este efeito, tente clicar repetidamente em outras informações sobre seu dispositivo. Eventualmente as opções de desenvolvedor serão desbloqueadas.

Lembrete:

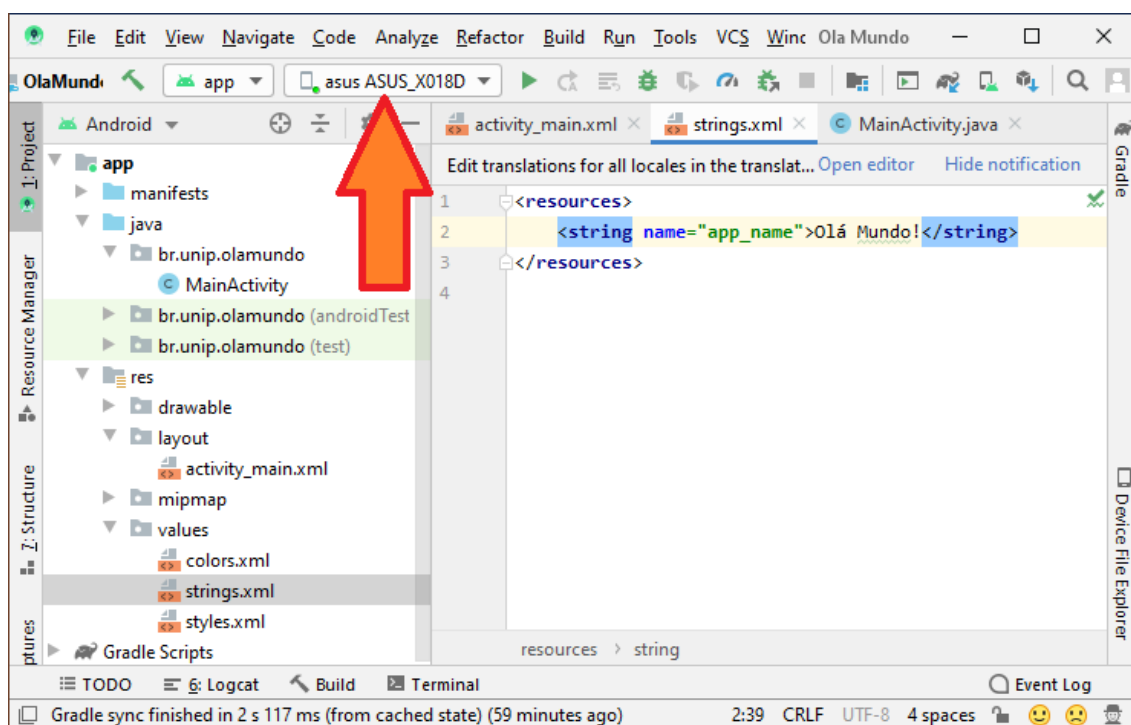
Você pode bloquear novamente as opções de desenvolvedor se assim o desejar. Isso costuma ser feito no próprio menu de opções de desenvolvedor.

Observação:

As opções de desenvolvedor não possuem qualquer relação com o desbloqueio de acesso *root* do dispositivo. Você pode habilitar as opções de desenvolvedor sem correr o risco de perder a garantia do dispositivo. Apenas tome cuidado com as opções que utilizar e ao conectar seu dispositivo por cabo USB em portas desconhecidas, pois seu dispositivo estará habilitado para ser programado pelo cabo.

A seguir, volte para a raiz do menu de configurações e entre no menu de Desenvolvedor. Lá, habilite a opção “Depuração USB”. Agora seu aparelho está configurado para receber, executar e depurar seu aplicativo. Conecte seu dispositivo ao computador de desenvolvimento com um cabo USB. Após a instalação dos drivers de seu dispositivo no seu computador, o Android Studio irá reconhecer seu aparelho.

Figura 13: Dispositivo real disponível para executar o aplicativo.



Fonte: O autor.

Observação:

No momento da escrita deste livro-texto, ainda é possível usar o truque que será explicado abaixo para executar um emulador em uma máquina de 4GB de memória RAM. Entretanto, a cada nova versão do Android Studio, mais difícil tem sido fazer com que este truque funcione de maneira aceitável. Caso este seja seu caso, recomenda-se usar um dispositivo real, pois ele não irá consumir memória da máquina de desenvolvimento.

Outra solução (por enquanto) é executar o Android Studio em Linux. Como este sistema operacional consome muito menos memória que os demais, mais memória permanece disponível tanto para o Android Studio quanto para o emulador.

Outra maneira de executar, testar e depurar seu aplicativo é em um dispositivo Android virtual, que pode ser totalmente emulado ou virtualizado, dependendo das capacidades de seu computador.

Neste ponto, devemos planejar como configurar tal emulador. Em primeiro lugar, um dispositivo emulado **sempre** terá um desempenho pior do que o de um dispositivo real. O quanto ele será pior depende de o emulador estar em modo emulado ou virtualizado. Em modo emulado, o dispositivo é interpretado inteiramente por *software*. Esta é a situação de menor desempenho possível, mas pode ser a única opção. Isto ocorre, por exemplo, se seu computador não for compatível com a tecnologia de virtualização utilizada pelo Android Studio. Já em modo virtualizado, o processador de seu computador consegue executar “diretamente” as instruções do sistema operacional Android emulado. Esta é uma situação de melhor desempenho. Além disso, caso seu computador também possua recursos de GPU (placa de vídeo discreta ou mesmo incorporada), estes poderão ser usados para acelerar o desempenho de seu emulador.

Entretanto, o fator mais importante para se considerar ao configurar um emulador Android é quanta memória RAM você tem disponível em seu computador de desenvolvimento. Caso possua 8GB ou mais, você poderá emular qualquer dispositivo com qualquer versão de Android (talvez até mais de um emulador simultaneamente). Já se você possuir entre 4 e 8GB, vai precisar planejar como configurar seu emulador. Afinal, alguns dispositivos Android, principalmente os mais caros, dispõem de 2, 3, 4, até 16Gb de memória RAM. Se tentarmos emular um dispositivo destes, a memória dele deve ser reservada da memória RAM do computador de desenvolvimento. Ou seja, se você estiver usando um computador de 4GB e tentar emular um dispositivo Pixel de 2GB de RAM, você acabará usando mais de 2GB do seu computador apenas para emular o dispositivo. Isso não irá funcionar. O que iremos fazer é configurar um emulador de um dispositivo razoavelmente antigo, mas que ocupa pouca memória da máquina de desenvolvimento.

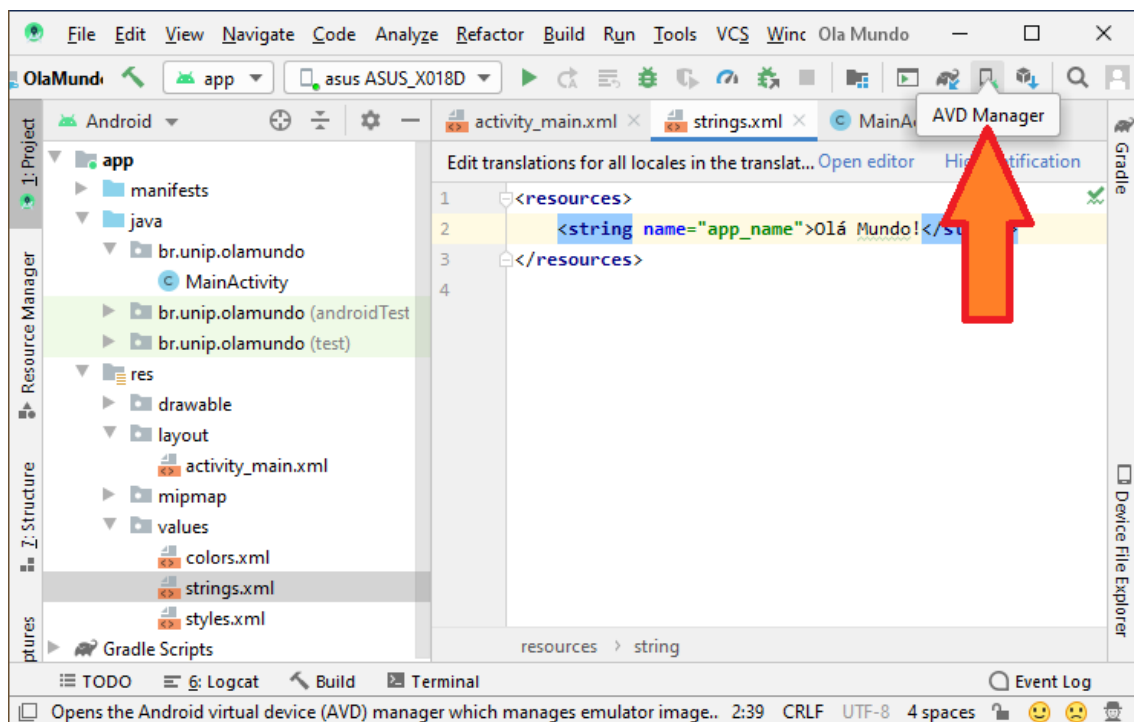
Saiba mais:

A disponibilidade de recursos de aceleração por *hardware* assim como a sua configuração depende da versão do Android Studio que estiver utilizando e do sistema operacional do computador de desenvolvimento. Caso você encontre dificuldades, você pode consultar o guia atualizado do Android Studio, que no momento da escrita deste livro-texto encontra-se no seguinte endereço:

<https://developer.android.com/studio/run/emulator-acceleration>

Abra o Gerenciador de Dispositivos Android Virtuais (“AVD Manager”).

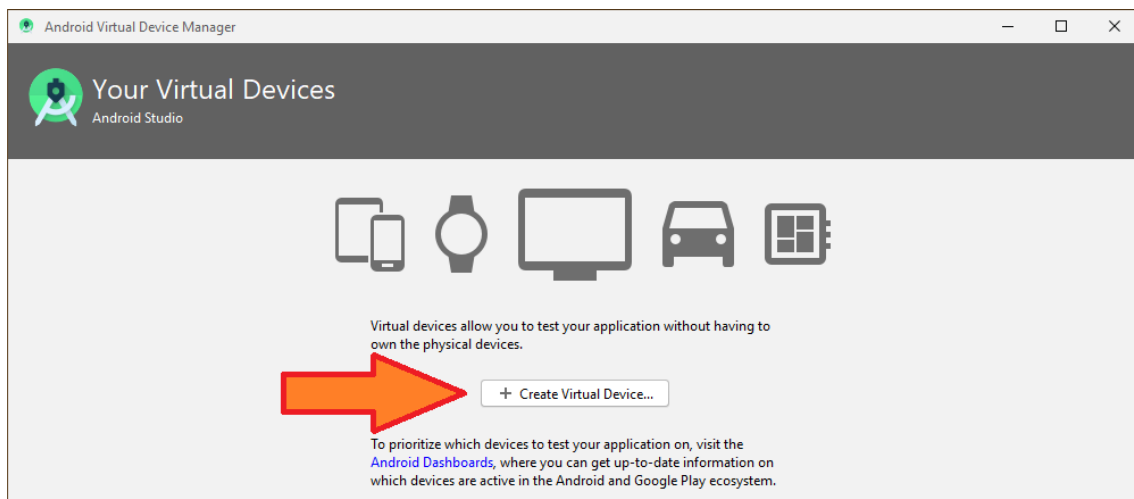
Figura 14: Abrindo o Gerenciador de Dispositivos Android Virtuais.



Fonte: O autor.

Crie um novo dispositivo virtual.

Figura 15: Crie um novo dispositivo virtual.



Fonte: O autor.

Selecione o modelo de telefone “Nexus S” (Nexus Ésse, não escolha o Nexus Cinco, pois o segundo utiliza muito mais memória RAM!). Clique em “Next”. A seguir, vamos escolher a versão do Sistema Operacional Android que será instalado no emulador.

É claro que não podemos instalar com sucesso um sistema operacional moderno em um *hardware* antigo. Isso é verdade tanto nos dispositivos reais quanto nos dispositivos emulados. Devemos escolher uma versão contemporânea ao *hardware* que vamos emular. Neste caso, vamos escolher o Android Jelly Bean, nível de API 16.

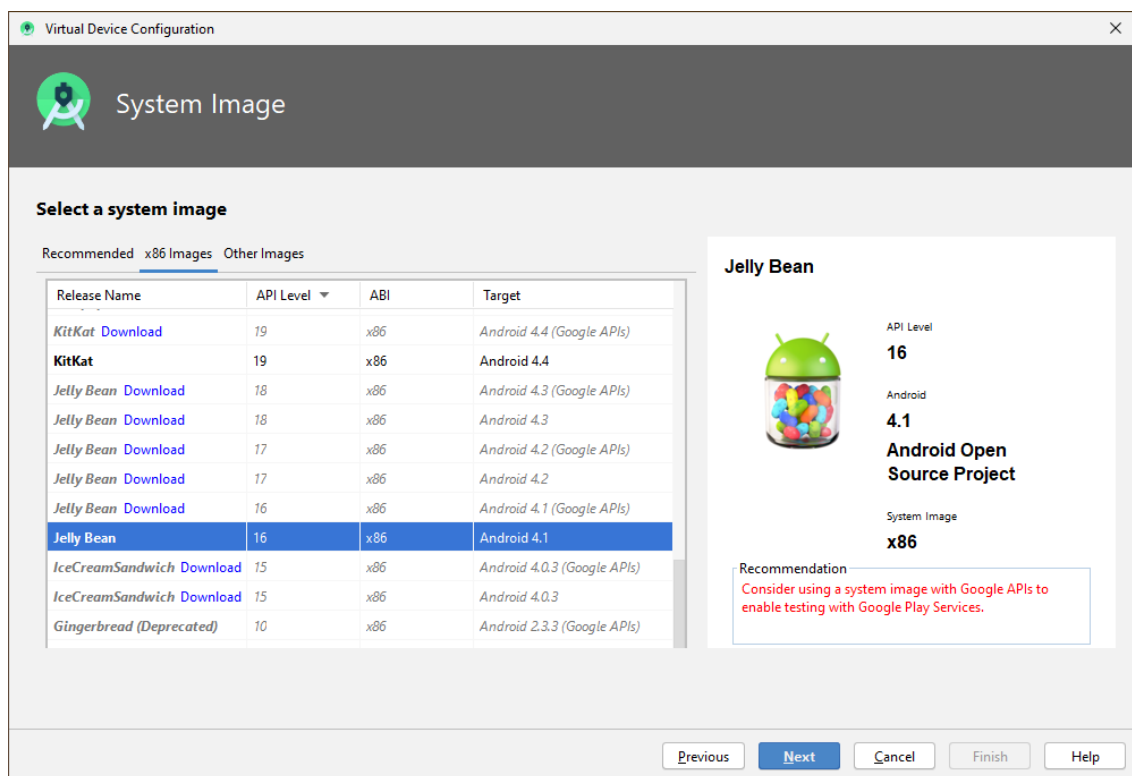
Lembrete:

No começo do capítulo configuramos nosso aplicativo para o nível mínimo de API 16. Assim, se escolhermos um emulador com no mínimo este nível de API, ele conseguirá executar nosso aplicativo.

Outro detalhe que devemos nos atentar é quanto ao processador usado pelo emulador. No momento da escrita deste livro-texto, há duas opções: ARM (32 e 64 bits) e x86 (32 e 64 bits). Apesar do processador de arquitetura ARM ser o mais popular nos dispositivos reais, ele não pode ser virtualizado por um computador com processador de arquitetura x86. Assim, vamos escolher uma imagem para a arquitetura x86. Além disso, escolha uma imagem de 32 bits, mesmo que seu computador esteja executando um sistema operacional de 64 bits. O que importa é a quantidade de memória RAM do dispositivo emulado. Só devemos usar imagens de 64 bits se o dispositivo emulado possuir mais do que 3 GB ou quando os dispositivos reais já não mais usarem S. O. de 32 bits.

Escolha a imagem Jelly Bean de nível de API 16 de arquitetura x86 conforme mostrado a seguir.

Figura 16: Escolha do sistema operacional do dispositivo emulado.



Fonte: O autor.

Observação:

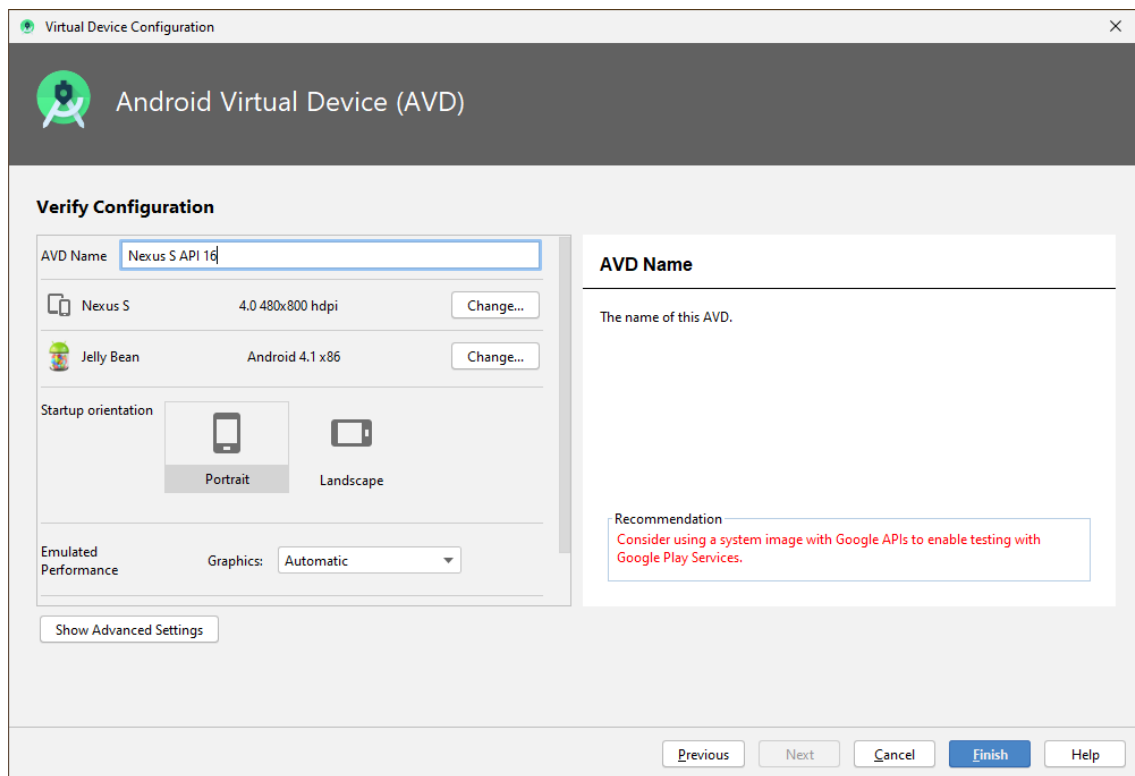
Na primeira vez que utilizar uma imagem, ela ainda não estará instalada. Clique no *link "Download"* ao lado da imagem, aguarde que a imagem seja baixada, então selecione-a.

Observação:

O assistente provavelmente irá sugerir que você use uma imagem com as APIs do Google. Elas permitem, entre outras coisas, que seu aplicativo utilize recursos da loja de aplicativos do Google. Como não iremos fazer isso agora, vamos escolher uma imagem sem estas APIs para diminuir o tamanho do download.

Você pode mudar o nome de sua máquina virtual se quiser. A seguir, clique em *"Finish"*.

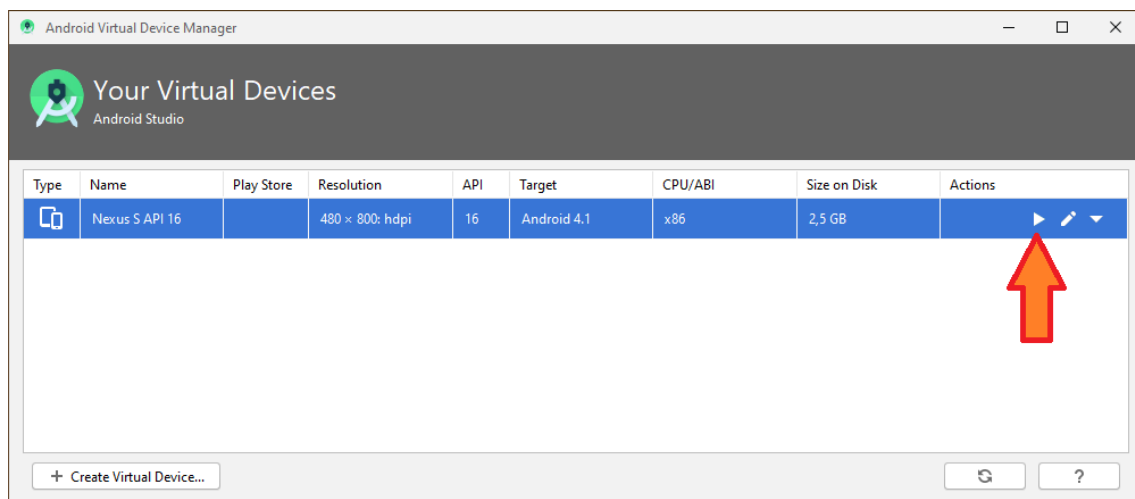
Figura 17: Concluindo a criação do emulador.



Fonte: O autor.

Neste ponto, inicie o seu emulador clicando no botão “*Play*”. Agora, o emulador irá iniciar o primeiro *boot*, o qual costuma ser mais demorado. Depois que o emulador abrir, você não precisa desliga-lo. Mantenha-o ligado e disponível como se fosse um dispositivo real.

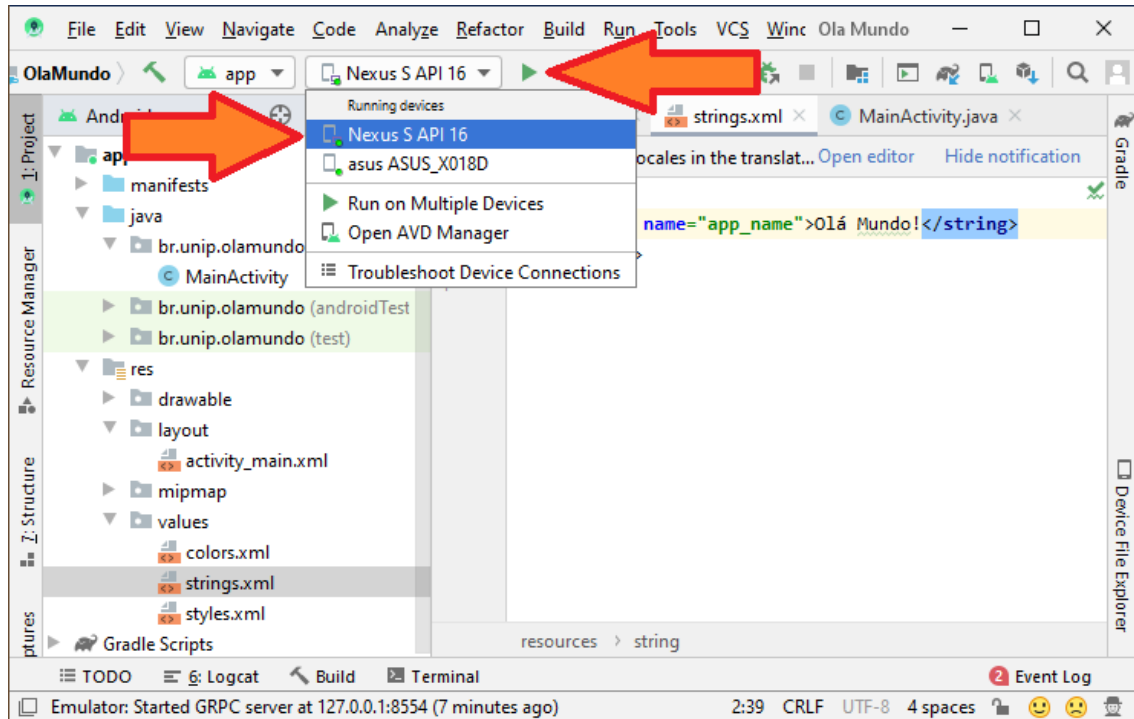
Figura 18: Iniciando o emulador.



Fonte: O autor.

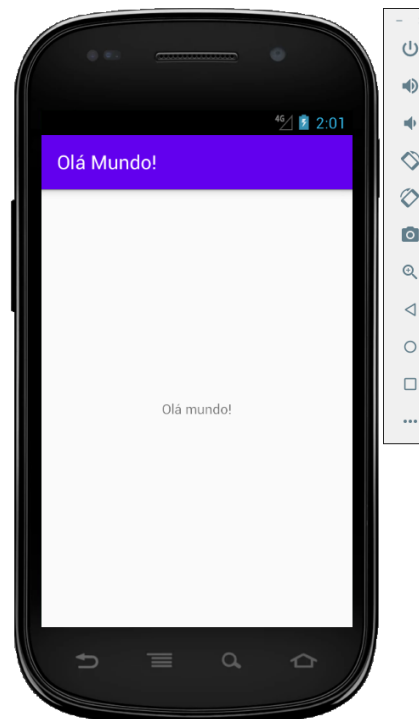
Neste ponto você já pode selecionar um dispositivo que esteja rodando ou um que esteja configurado, mas ainda não esteja rodando, compilar, instalar e executar seu programa.

Figura 19: Compilar, instalar e executar o aplicativo.



Fonte: O autor.

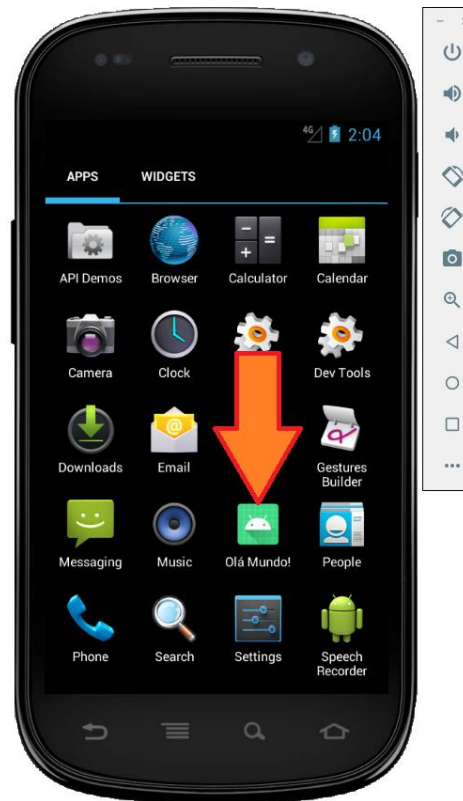
Figura 20: Aplicativo "Olá Mundo!" em execução no emulador.



Fonte: O autor.

Volte à tela inicial do Android e entre na gaveta de aplicativos. Note como o nome do aplicativo que você acabou de criar é mostrado com acento.

Figura 21: Nome do aplicativo com acentos e caracteres especiais.



Fonte: O autor.