Android Studio: Instalação, Configuração e Primeiro App

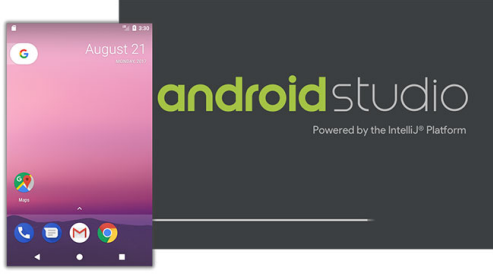
Android com Java

Flávio Augusto de Freitas



Android Studio

Neste tutorial vamos passo a passo estudar o Android Studio. Sua origem, o que era utilizado anteriormente, a instalação e configuração deste IDE, otimização para máquinas menos robustas e um primeiro projeto Android.



Caso você já seja um desenvolvedor Android que utilize alguma das linguagens oficiais, porém devido a lentidão do Android Studio em sua máquina você tenha optado por utilizar algum outro IDE, não deixe de acompanhar o tutorial, ou vídeo, até o final para então tentar as configurações de otimização que serão apresentadas para assim voltar aos trabalhos de desenvolvimento com o principal IDE Android.

Histórico

O Android já estava disponível para os programadores desde 2007, na verdade estava disponível um plugin para o IDE (Ambiente de Desenvolvimento Integrado) Eclipse que permitia que o Android SDK (*Software Development Kit*) fosse utilizado para o desenvolvimento de aplicativos deste sistema operacional.

O plugin era o ADT, Android Development Kit Tools.

Até antes da liberação da primeira versão estável do Android Studio o Eclipse dominou o mundo do dev Android.

Hoje o ADT não mais é mantido devido a inúmeros bugs e falhas de segurança, segundo o site oficial do Android. A recomendação é que todos os desenvolvedores façam a migração imediata para o Android Studio.

Ainda há desenvolvedores que trabalham com o Eclipse, muitos deles provavelmente porque desconhecem que é possível configurar o Android Studio para que este IDE rode ao menos tão leve quanto o Eclipse roda nas máquinas deles.

No decorrer do tutorial vamos abordar o passo a passo para trabalhar a performance do Android Studio em máquinas mais simples, 4 GiB de memória ou menos.

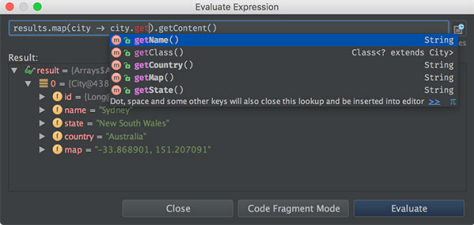
2

Outros IDEs para desenvolvimento de aplicativos Android

Apesar do Eclipse ter dominado por muito tempo como o principal IDE para desenvolvimento de aplicativos Android, muitos outros IDEs eram / são utilizados.

Um que veio ganhando força até o momento do lançamento oficial do Android Studio 1.0 em 2014 foi o IDE base deste, o IntelliJ IDEA da JetBrains.

O IntelliJ IDEA é um IDE JVM que permite, além de outras coisas, o uso do Android SDK para que aplicativos desse SO mobile sejam desenvolvidos. A seguir uma imagem simples de uso do IntelliJ:

*Então é a JetBrains que desenvolve o Android Studio?*

Não, na verdade o IntelliJ IDEA foi somente utilizado como IDE base, as configurações para que o IDE Android Studio seja um IDE exclusivo para desenvolvimento Android, essas configurações foram e são desenvolvidas pelos profissionais do Google.

Alias este é um ponto fraco do Eclipse IDE que também foi um dos motivos da criação do Android Studio: ele era muito genérico, atendia a muitas linguagens, tendo assim várias funcionalidades que não eram úteis aos desenvolvedores Android.

A seguir listo alguns outros IDEs também utilizados para desenvolvimento Android: • NetBeans e plugin Android;

• Xamarin, este é para desenvolvimento de app multiplataforma, porém utilizando a linguagem C#. IDE ainda mantido pela Microsoft;

• Delphi;

• Basic4Android. Acredite, há developers que desenvolvem excelentes aplicativos utilizando este IDE Basic;

3

• IntellXDK para aqueles que preferem tecnologias Web no mundo mobile, mais precisamente o HTML5;

• Corona Labs para criação de games. Terá de utilizar a linguagem Lua;

• Adobe PhoneGap, com este você utilizará também tecnologias Web: CSS, HTML e JavaScript.

Na lista acima coloquei os IDEs mais populares depois do Android Studio, Eclipse e IntelliJ IDEA. Existem alguns outros que permitem o dev Android.

Note que o objetivo foi apenas apresentar alguns outros IDEs, isso, pois para desenvolvimento Android o Android Studio é a melhor opção, muito por ser um IDE especialista somente para este tipo de desenvolvimento.

Lançamento

Em maio de 2013 o Google anunciou seu novo IDE para desenvolvimento de apps Android, utilizando a licença Apache 2 o Android Studio, baseado no IntelliJ IDEA e com o software de automação de compilação Gradle, foi liberado em sua versão de preview.

A seguir a imagem de como era a inicialização da primeira versão desse IDE:



Em dezembro de 2014 foi liberada a versão 1.0, a primeira estável do Android Studio. Fazendo assim com que o Google adotasse este IDE como o prioritário para desenvolvimento Android.

Com o lançamento da versão estável o Google passou a incentivar os desenvolvedores a migrarem de outros IDEs, principalmente o Eclipse, para o Android Studio. Mesmo assim manteve por um bom tempo a atualização do plugin ADT para Eclipse para posteriormente deprecia-lo e então prosseguir somente com a evolução do novo IDE oficial.

4

Objetivos

Acelerar, em muito, o desenvolvimento de aplicativos para o SO mobile do Google e se tornar o principal IDE, em números de uso, para desenvolvimento de apps deste SO.

Com este IDE temos todas as ferramentas para o desenvolvimento e testes completos dos projetos de aplicativos. É possível desenvolver aplicativos também para Tablet, TV, relógio, Google auto e Google Glass.

Caso você seja um desenvolvedor Java, Kotlin ou C/C++ é possível prosseguir sem problemas com o Android Studio, pois para as linguagens oficiais de desenvolvimento, ou você já pode iniciar o software e utilizar a linguagem de seu conhecimento ou apenas terá de instalar um outro kit de desenvolvimento, como o NDK que é para dev em C/C++.

De qualquer forma, para a linguagem oficial de sua escolha você terá todo o suporte do IDE para desenvolvimento e testes.

Com as novas versões estáveis que são liberadas de tempos em tempos, o Android Studio vem ficando cada vez mais simples e facilitando ainda mais a programação de apps, com a compra da Fabric pelo Google podemos esperar ainda mais melhorias ao menos na integração de APIs populares, como está ocorrendo com as APIs Firebase.

Instalação e configurações essenciais

A partir daqui vamos direto ao uso do IDE. Posteriormente vamos a apresentação de algumas das principais características junto a criação de um projeto inicial simples.

Prosseguindo deste ponto do conteúdo é importante que você siga na prática cada uma das seções apresentadas, assim você conseguirá já configurar seu ambiente de desenvolvimento, até mesmo se sua máquina tiver pouca memória.

Pré-requisitos

No "mundo ideal" é esperado, segundo a documentação, que sua máquina Windows / Linux / Mac esteja com o SO atualizado e com ao menos os seguintes requisitos de hardware:

• 4 GiB de memória, onde somente o emulador utiliza 1 GiB e o restante será eventualmente utilizado pelo IDE;

• 2 GiB espaço livre em disco, onde 500 MiB é para o IDE e 1.5 GiB é para as imagens de sistema do emulador Android, na verdade o emulador pode vir a utilizar 1.5 GiB de espaço;

• 1.280 x 800 é o mínimo para resolução de tela.

Falei no "mundo ideal", pois como já informado anteriormente, é possível ajustar / desativar algumas características e funcionalidades desse IDE para poder utiliza-lo em máquinas ainda mais simples.

5

Download e instalação

Aqui vamos trabalhar a versão mais atual disponível do IDE, na construção deste tutorial a versão era a 3.0 Beta 2 (Canary 8). Apesar de ser uma versão beta, eu ainda utilizo em produção a última versão estável do IDE, nesta época era a versão 2.3.3.

Recomendo que tenha o mesmo comportamento para seus projetos que seguirão a loja de aplicativos: mantenha o uso da última versão estável do IDE.

Isso, pois esta já passou por inúmeros testes e você não terá dores de cabeça com, por exemplo: pop-ups de informação que não fecham ou encerramentos repentinos do IDE.

Para download e instalação, siga os passos abaixo:

• Entre no link a seguir e faça o download:

https://developer.android.com/studio/preview/index.html?hl=pt-br;

• Extraia o software do arquivo **.zip** e então inicie o instalador:

• Dependendo do seu SO será solicitado o local adequado para a instalação do IDE, se possível siga sempre com as configurações padrões;

• No MAC não há a opção de inclusão ou não do Android SDK, logo, se em seu SO houver este questionamento de inclusão de SDK, isso ainda no passo de instalação, marque para que o Android SDK seja adicionado.

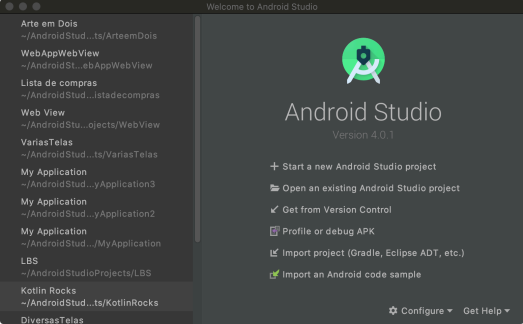
Preze sempre por realizar o download do Android Studio no site oficial, digo isso, pois há inúmeros outros sites que até mesmo colocam que é gratuito o download por eles. Isso é verdade também no site do Android, pois este IDE é gratuito.

Ao final da instalação acione o novo software, terá de aguardar o carregamento inicial (que pode demorar um pouco):

6



Logo depois, caso não haja nenhum projeto aberto anteriormente no Android Studio, o menu inicial é apresentado:

À esquerda temos a lista de projetos recentes.

7

JDK, Java Development Kit

Como o Android Studio é um IDE Java, precisamos do kit de desenvolvimento desta linguagem. Porém provavelmente sua máquina já tem algum JDK instalado, logo isso nos poupa tempo, pois automaticamente o IDE passa a utiliza-lo.

Mas com a versão do Android Studio que instalamos é recomendado que tenhamos a versão mais atual do kit Java, o JDK 8, na época deste conteúdo.

Caso você não tenha ou esteja utilizando uma versão anterior em sua máquina, temos a opção de utilizar o Open JDK que vem embutido ao IDE.

Assim, com o menu inicial aberto, siga:

• Clique em **Configure**;

• Logo depois em **Project Defaults**;

• Então clique em **Project Structure**;

• Na tela **SDK Location**, mais precisamente na seção **JDK Location**, dê um check ✔ em **Use embedded JDK (recommended)**;

• Clique em **Apply** ou **Ok**.

Com isso você não mais deverá ter problemas com a configuração de JDK para o Android Studio, o que está embutido é a versão estável mais atual do Open JDK.

*Já que o Open JDK é o recomendado e já vem embutido no IDE, por que este não é utilizado por padrão desde o inicio?*

Provavelmente por questões comerciais que envolvem a Oracle, proprietária do Java e não controladora do Open JDK.

SDK Tools

São necessários alguns kits de desenvolvimento, software development kit (SDK), para que seja possível prosseguir com o desenvolvimento de apps no Android Studio. Pode ser que sua instalação de IDE não inclua algumas ferramentas desde o início.

Para isso, ainda no menu inicial, siga:

• Clique em **Configure**;

• Logo depois em **SDK Manager**;

• Na nova janela aberta clique na aba **SDK Tools**;

8

• Dê um check ✔ ao menos nas seguintes ferramentas (caso elas já não estejam instaladas em sua versão de IDE):

• **Android SDK Build-Tools**;

• **Android SDK Tools**;

• **Android SDK Platform-Tools**;

• **Android Support Repository**, sub-item de **Support Repository**;

• **Google Repository**, também um sub-item de **Support Repository**;

• **Intel x86 Emulator Accelerator (HAXM Installer)**. Este é para a execução eficiente do emulador em sua máquina. Pode ser que a opção disponível seja a **ARM**, instale esta então, pois o accelerator apresentado é de acordo com a máquina de desenvolvimento.

• Por fim clique em **Apply** ou **Ok**, aceite os termos de licença para as SDKs selecionadas, caso algum seja apresentado, e então aguarde a instalação finalizar. Não esqueça de reiniciar o Android Studio ao final.

9

Depois de instaladas as ferramentas, o Android Studio se encarrega de lhe informar sobre atualizações delas quando disponíveis, mas não deixe de entrar em **SDK Manager** > **SDK Tools** para poder você mesmo acionar as atualizações ainda pendentes e, por algum motivo, não informadas.

Note que dependendo do projeto que você for desenvolver será necessário a instalação de ainda mais ferramentas.

SDK Platforms

Ao menos uma plataforma SDK Android é necessária para que você consiga prosseguir com a criação de projetos no Android Studio.

O recomendado é que sempre a mais atual esteja instalada em seu IDE. Pois está tende a ser a API alvo de nossos projetos, isso porque ela já incluirá todas as APIs e funcionalidades das versões anteriores do Android.

Novamente no menu inicial, siga os passos:

• Clique em **Configure**;

• Logo depois em **SDK Manager**;

• Clique na aba **SDK Platforms**;

10

• Selecione a mais atual, deixando ela com um check ✔. Na época deste tutorial a mais atual era a SDK **Android 10.0 (Q)**;

• Clique em **Ok**, aceite os termos caso surja algum e aguarde a instalação;

• Ao final reinicie o Android Studio.

Como com as SDK Tools, depois de selecionada a SDK Platform e instalada, o Android Studio se encarregará de informar sobre novas atualizações.

É possível, e provavelmente você fará isso em pouco tempo, ter mais de um SDK de plataforma instalado

SDK Update Sites

Por padrão o Android Studio já tem uma série de fontes de atualização para os seus SDKs. Eu particularmente ainda não me vi na necessidade de adicionar ainda mais fontes e recomendo que inicialmente você não modifique a configuração inicial para está parte do IDE.

Vale ressaltar que posteriormente, com o desenvolvimento de novos conteúdos Android aqui no Blog e canal, nós estaremos configurando ainda mais itens ao IDE e consequentemente indo mais a fundo em algumas características dele.

11

Canais de atualização de versão do IDE

Para atualizar o Android Studio não há necessidade de entrarmos no site oficial com frequência e então mantermos o download da última versão.

O próprio IDE nos informa sobre novas versões, digo, informa de acordo com o canal de atualização escolhido. Os possíveis são:

• **Canary**: passou por poucos testes e tem periodicidade semanal.

• **Dev**: teve uma bateria de testes maior, mas ainda apresenta alguns bugs não filtrados pelos feedbacks da versão canary. O delay de liberação também é maior do que no canal canary.

• **Beta**: quase em produção, com vários bugs já corrigidos, mas ainda em ajustes.

• **Stable Channel** / **Canal Estável**: versão de produção, recomendada para a construção de seus aplicativos Android que vão para as lojas de apps.

O canal configurado como padrão é de acordo com a versão de seu IDE, caso seja uma versão estável o canal **Stable Channel** será o default. Para mudar isso, no menu inicial, siga os passos abaixo:

• Acione **Configure**;

• Em seguida acione **Preferences**;

• Logo depois vá em **Appearance & Behavior**;

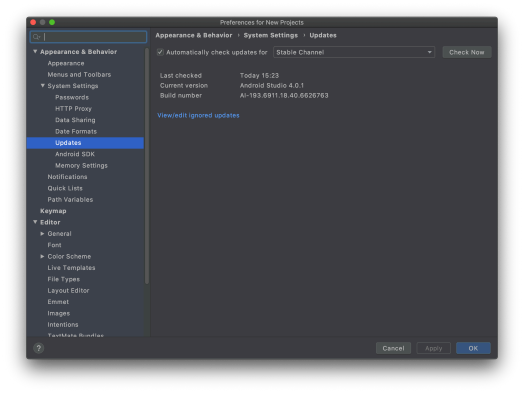
• Expanda **System Settings**;

• Vá em **Updates**;

• Deixe marcada, ✔, a opção **Automatically check updates for**. No menu drop down ao lado, escolha o canal que você deseja manter os avisos de atualização;

• Por fim clique em **Apply** ou **Ok**.

12



Primeiro projeto e principais características

Com as configurações iniciais realizadas, podemos prosseguir com a criação do primeiro projeto. Vamos iniciar um projeto Java. Discutiremos também a estrutura de projeto e a ferramenta de compilação.

Criando o projeto

Com o menu inicial aberto, siga os passos a seguir:

• Clique em **Start a new Android Studio project**;

• Escolha **Phone and Tablet**, depois **Empty Activity** e clique em **Next**.

13

• Na página seguinte serão solicitadas algumas configurações iniciais de projeto: • Em **Name** coloque o nome de seu aplicativo, aqui vamos seguir com "App01";

• Em **Package name** é comum colocar a URL invertida da página da empresa do aplicativo, isso caso você tenha um domínio público. Eu utilizo frequentemente a URL do meu Blog acrescido do nome do app (por exemplo, com.blogspot.flavioaf.app01), mas pode deixar "br.com.app01" por simplicidade. Você pode utilizar o que preferir, mas este é o padrão utilizado. Além disso, cada app deve ter esse nome único para ser publicado na Play Store;

• Em **Save location** você deve informar onde o projeto se encontrará em sua máquina. O que faço é criar um diretório **/Applications** e assim colocar todos os meus projetos Android nele, isso, pois tenho mais de um IDE em minha máquina e também porque fica melhor organizado;

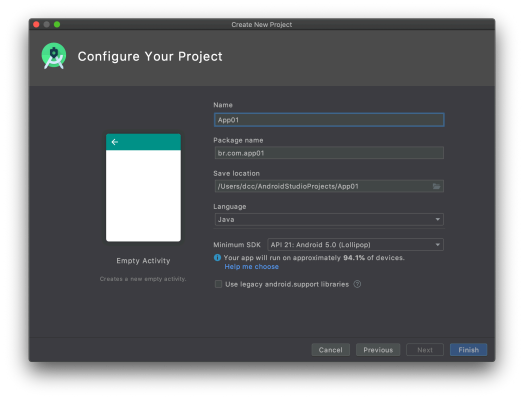
• Em **Package name** por padrão será utilizado os seus **Package name** e **Name** sem caracteres especiais e espaço em branco;

• Em **Language** escolha Java.

14

• Em **Minimum SDK** você pode escolher API mínima, escolha a API 15, **Android 4.0.3, Ice Cream Sandwich**. Caso você não tenha a API 15 como opção, selecione a menor possível, pois posteriormente, no código do projeto, poderemos atualizar essa versão facilmente, mas sugiro deixar como está.;

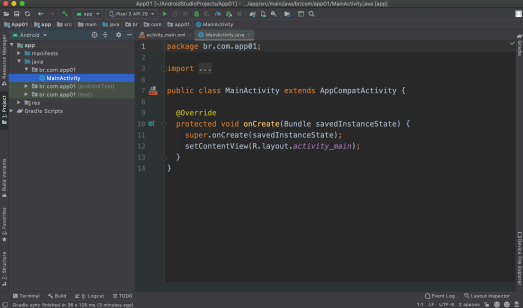
• Clique em **Finish**.



Essa inicialização pode demorar um pouco, é normal até o momento de trabalharmos as configurações do Android Studio para que ele rode de maneira eficiente em sua máquina.

Por fim temos o projeto aberto:

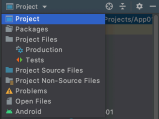
15



Estrutura de projeto

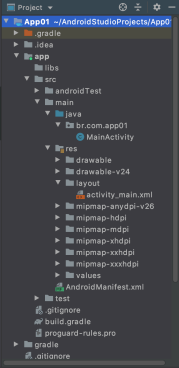
Inicialmente é comum termos a janela de ferramenta **Project** aberta em nosso projeto. Por ela conseguimos navegar por todos os arquivos.

De início o modo de visualização desta janela é **Android**. Há outros modos.



Caso você queira ver como verdadeiramente é a estrutura de arquivos e folders de seu projeto em disco mude a visualização para **Project**:

16



Depois deste pequeno teste, volte com a visualização em **Android**.



Todo projeto no Android Studio contém ao menos um módulo e cada módulo tem os seguintes diretórios:

• **/manifests**: aqui se encontra o arquivo AndroidManifest.xml, arquivo de configuração que é útil ao Android SO e as lojas de aplicativos que disponibilizam apps Android;

• **/java**: aqui se encontram todos os arquivos de programação de seu projeto, incluindo os arquivos de teste que são gerados automaticamente depois da criação do project. Mesmo quando o projeto é Kotlin o folder de programação é **java**.

17

• **/res**: todos os recursos estáticos ficam neste folder, recursos como: imagens, vídeos, áudios, layouts XML, Strings internas, entre outros.

A seguir a imagem da estrutura de projeto expandida:



Indo mais no detalhe em **/res**, temos:

• **/drawable**: onde ficarão as imagens, ícones e alguns XMLs responsáveis por efeitos como gradiente e sombra. Este folder tende a ter ainda mais versões dele, são elas: **mdpi**, **hdpi**, **xhdpi** e **xxhdpi**. Todas essas referentes a densidade de pixels em tela;

• **/layout**: aqui ficam todos os layouts XML de seu projeto. Como em **/drawable**, é possível ter outras versões deste folder para suporte, por exemplo, versões para devices em orientação horizontal;

• **/mipmap**: aqui entram os ícones de seu aplicativo, digo, o ícone de lançamento dele, o que ficará na área de aplicativos do device. O modelo de trabalho aqui é similar ao do folder **/drawable**, exceto que das versões solicitadas para o ícone há mais uma, a **xxxhdpi**;

• **/values**: neste folder ficam as Strings, as especificações de tema, cores e outros itens, como arrays de conteúdos, que podem ser carregados no projeto. Como acontece com outros folders em **/res**, **/values** também pode ter outras versões. Para arquivos a rodar no projeto somente em devices com o Android 21 ou superior, teríamos, por exemplo, **/values-v21**.

18

Note que "Gradle Scripts" não faz parte da estrutura de projeto, o Gradle é uma ferramenta de automação de compilação, que falaremos um pouco mais na próxima seção.

Gradle

Se você é um desenvolvedor Java "das antigas" deve saber o que é ter de carregar manualmente as APIs externas necessárias em seu projeto, incluindo a necessidade de compilação na mão, utilizando o prompt de comando.

Com o Eclipse isso já não mais era necessário, a compilação e outras necessidades para a execução de um projeto já eram automatizadas no próprio IDE, fazendo uso da ferramenta de automação Apache Ant.

O Gradle é exatamente isso, uma ferramenta de automação de compilação. A seguir a lista de algumas funcionalidades delegadas ao Gradle:

• Carregar automaticamente APIs externas;

• Realizar testes nos algoritmos do projeto;

• Compilar o projeto para a geração de um arquivo APK (pacote de aplicativo Android);

• Permitir a geração de mais de um arquivo APK, flavors. Um para a versão gratuita e outro para a versão paga, por exemplo;

• Assinar o APK com chaves de debug para execução em ambiente de desenvolvimento; • Redução de recurso, passo realizado para aumentar a performance do APK;

• Permitir a integração com o ProGuard para melhor desempenho e segurança no código do aplicativo.

Apesar dos poucos itens citados, somente no "Compilar o projeto para a geração de um arquivo APK" tem trabalho suficiente para demandar alguns minutos de seu tempo de produção.

Como curiosidade, o Gradle faz uso da sintaxe do Groovy, uma linguagem desenvolvida para plataformas Java.

Quando criamos um projeto temos uma série de arquivos gerados para essa ferramenta de compilação, porém somente dois são, a princípio, importantes e passíveis de atualização na maioria dos casos:

• **build.gradle** em nível de projeto, que serve para todos os módulos do projeto; • **build.gradle** em nível de módulo, cada módulo tem o seu.

A seguir os arquivos **build.gradle** do projeto que acabamos de criar.

Primeiro o arquivo de nível de projeto:

19

// Top-level build file where you can add configuration options common to all sub-projects/modules. buildscript **{**

repositories **{**

google()

jcenter()

**}**

dependencies **{**

classpath "com.android.tools.build:gradle:4.0.1"

// NOTE: Do not place your application dependencies here; they belong

// in the individual module build.gradle files

**}**

**}**

allprojects **{**

repositories **{**

google()

jcenter()

**}**

**}**

task clean(type: Delete) **{**

delete rootProject.buildDir

**}**

É importante saber que o Gradle é uma ferramenta de compilação independente do Android Studio e da plataforma Android. Ou seja, a atualização dele no Android Studio é também independente, pode ser que tenha uma nova versão do Gradle quando ainda não há uma nova versão do próprio IDE.

A seguir o arquivo **build.gradle** de nível de módulo do único módulo que temos neste projeto inicial, o módulo **app**:

apply plugin: 'com.android.application'

android **{**

compileSdkVersion 29

defaultConfig **{**

applicationId "br.com.app01"

minSdkVersion 21

targetSdkVersion 29

versionCode 1

versionName "1.0"

testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"

**}**

buildTypes **{**

release **{**

minifyEnabled false

proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pro' **}**

**}**

**}**

20

dependencies **{**

implementation fileTree(dir: "libs", include: ["\*.jar"])

implementation 'androidx.appcompat:appcompat:1.2.0'

implementation 'androidx.constraintlayout:constraintlayout:2.0.2'

testImplementation 'junit:junit:4.12'

androidTestImplementation 'androidx.test.ext:junit:1.1.2'

androidTestImplementation 'androidx.test.espresso:espresso-core:3.3.0'

**}**

Acima coloquei os comentários nos pontos que mais importam em um projeto inicial.

Saiba que é possível ter mais de um módulo e que os outros podem ser, além do de aplicativo, de biblioteca e de engine do Google.

Note que o Gradle necessita de conexão com a Internet para poder funcionar sem problemas, principalmente porque ele mantém a verificação de atualizações na rede e também necessita carregar APIs externas, inclusive para seu funcionamento.

Estes são também os porquês de o Android Studio necessitar de conexão com a Internet para funcionar 100%. De qualquer forma, caso você tenha limitação em sua banda de Internet, é possível trabalhar com o Gradle em Offline Mode.

Editor de código e layout

A janela do lado direito no IDE é para edição de código, o modelo de apresentação de sintaxe desta janela se ajusta de acordo com o tipo de arquivo aberto.

Inicialmente você já deve ter visto como fica com o código Java. Agora abra o arquivo XML **/res/layout/activity\_main.xml**. Assim terá algo como:

21

Isso assumindo que a aba **Code**, no topo direito da janela de código, está selecionada.

Note que os códigos de layout são em linguagem de marcação XML, mas podemos construí-los em tempo de execução com o Kotlin / Java.

Eu particularmente prefiro construir todos os layouts em XML e pelo editor XML, mas é possível utilizar a ferramenta de design *drag & drop* do Android Studio. Para isso selecione no topo à direita da janela de código a opção **Design** e terá algo como:

22

• A esquerda, um menu dos itens que podem ser arrastados e colocadas no layout;

• Ao centro, a visualização do layout, incluindo a blue-print-screen para facilitar a visualização da estrutura;

• A direita, o menu de atributos de alguma das visualizações selecionadas no centro da ferramenta de design;

• No topo, além de várias funcionalidades possíveis, há a ferramenta de definição de device para apresentação do layout no centro. No teste acima estamos com o device Pixel.

Caso você queira atualizar as definições de cores e marcações do editor de código, vá ao menu do Android Studio, logo depois em **Preferences** (ou em **Settings**, depende de seu SO), então em **Editor** e assim você terá uma série de configurações que poderão ser alteradas.

Emulador e execução do projeto

Diferente de outras versões do Android Studio e até mesmo de outros IDEs, desde o Android Studio 2.3.3 o emulador nativo, ou Android Virtual Device (AVD), está definido de maneira eficiente, ou seja, não mais há a necessidade de uso de plugins de emulador como o Genymotion.

Com o AVD Manager é possível criar vários emuladores para posteriores testes, incluindo emuladores de relógio, TV e tablet.

23

O que recomendo, ao menos para os testes finais de seu app, é a criação de um emulador com a mesma API level de alvo de compilação de seu projeto. Aqui nossa API é a 26. E também um outro emulador com a API level mínima atendida pelo seu app, aqui nossa API level mínima é a 15.

Ambos com densidade média de pixels, xhdpi.

Criando um emulador

No menu do Android Studio acesse **Tools**, logo depois acesse **AVD Manager**:



Na próxima tela clique em **+ Create Virtual Device...** . Logo depois, na coluna **Category**, selecione **Phone**. Na coluna central selecione o tipo de device que deseja utilizar. Aqui vamos prosseguir com o Pixel 3, densidade 440dpi:

24

Clique em **Next**.

Na próxima tela temos de definir a API level, imagem de sistema, que estará no device. Essa tela já abre com a aba **Recommended** selecionada. Aqui vamos criar um device com a API 29, Android 10.0, e com o Google APIs já instalado, isso para permitir uso de mapa e outras funcionalidades que requisitariam o Google Play Services no device.

Você provavelmente terá de instalar essa imagem de sistema do Android 10.0, clique em download e aguarde o carregamento.

É recomendado que você utilize uma imagem de sistema de acordo com o processador de sua máquina, x86 para 32bits e x86\_64 para 64 bits:

25

Clique em **Next** e assim estaremos na última tela.

Nela você verifica o que já foi definido e ainda pode mudar algumas outras características, como o nome desse novo emulador. Aqui vou prosseguir com o nome já presente "Pixel 3 API 29":26

Clicando em **Show advanced settings** é possível mudar ainda mais configurações, mas vamos seguir com todas as configurações padrões, ou seja, deixe esta tela como está. Clique em **Finish**. Emulador criado.

Executando o projeto

Acesse o AVD Manager novamente e coloque seu novo emulador para rodar. Na coluna **Actions** clique no ícone verde de play na linha dele:

Depois que ele estiver iniciado por completo, como na figura abaixo:



Na barra de tarefas do Android Studio, clique no ícone verde de execução de projeto:

27

O programa será executado no emulador recém-criado, pois é o único que existe. No caso de haver mais de um emulador criado, para escolher em qual emulador o app deve ser executado, selecione o menu **Run** e depois **Select device...** . No menu que abre, selecione o emulador que deseja que execute o app, como na figura abaixo.



O menu fecha assim que você clicar no emulador escolhido. Depois na barra de tarefas do Android Studio, clique no ícone verde de execução de projeto, como na figura anterior. O app será executado no emulador escolhido.

Assim o aplicativo de exemplo será apresentado no emulador:



Note que ao lado do emulador há uma barra de controles e a última opção (**...**) é um menu onde é possível acionar outras funcionalidades nele, incluindo muitos dos sensores que o Android SO pode aceitar em um device real. A seguir a imagem do menu aberto:

28

Seu mouse é equivalente ao seu dedo em um device real. O teclado lhe permitirá entrar com dados em campos de formulários. 

Arrastando e soltando arquivos binários e outros APKs em seu emulador, respectivamente, salvará eles na área de arquivos ou instalará quando um APK.

Executando em um aparelho real

Siga os passos abaixo para conseguir executar seus aplicativos do Android Studio direto em um device real: 1. Conecte o dispositivo real ao computador pelo cabo USB;

2. No device acesse **Settings**;

3. Ainda no device acione **Developer options** para que o **USB debugging** seja ativado e você possa executar os aplicativos no aparelho;

Caso esteja com uma versão do Android acima da 4.2, as opções de desenvolvedores estão ocultas. Sendo assim siga, depois de já ter conectado o device ao computador:

1. No aparelho, acesse **Settings**;

2. Ainda no aparelho acesse **About phone**;

29

3. Toque sete vezes em **Build number** e volte a tela anterior;

4. Agora acione **Developer options**.

Quando for executar o aplicativo você terá seu device real como opção no dialog de **Connected devices**.

Pode ser necessário antes da execução

Caso mesmo depois de suas atualizações o aplicativo não demonstre elas, pode ser que o Android Studio não esteja atualizando o projeto por algum motivo interno, pouca memória disponível, por exemplo.

É até mesmo possível que exceções sejam geradas devido a não atualização de estrutura de projeto.

Para isso tente o rebuild do projeto indo, no menu do Android Studio, em **Build** e então em **Rebuild Project**:

Logo depois tente novamente a execução.

Otimização do IDE para máquinas limitadas

Há dois tipos de otimização a se realizar com o Android Studio:

• Um para quando sua máquina tem 8 GiB ou mais de memória e não apresenta lentidão com o Android Studio;

• Outro para quando sua máquina apresenta lentidão na execução deste IDE.

Aqui vamos dar foco no caso dois, pois em um momento dele será possível abordar como melhorar ainda mais a performance do Android Studio em máquinas robustas, mais precisamente na parte de configuração de tamanho de pilha.

30

Diminuir o tamanho máximo da pilha disponível para o IDE

O valor padrão de memória disponível ao IDE é de 1280 MiB. Com sua máquina apresentando lentidão o recomendado é diminuir essa pilha para 512 MiB.

Para isso, no menu do Android Studio vá em **Help**, logo depois em **Edit Custom VM Options...**.

Caso o arquivo **studio.vmoptions** ainda não tenha sido criado, será solicitado a ti que permita essa criação. Pode deixá-la ocorrer sem problemas.

Dentro do arquivo coloque a seguinte definição:

-Xmx512m

Logo depois reinicie o IDE para que a nova configuração seja definida.

Para confirmar, acione o prompt de comando de seu SO e coloque **jps -lvm**. Na tela de resultado busque por **-Xmx512m**. 

Para máquinas mais robustas é recomendado o aumento do tamanho da pilha, para 2 GiB, por exemplo: -Xmx2g

Desative o processamento paralelo

É possível que o Android Studio compile módulos independentes em paralelo. Em caso de lentidão isso é um ponto fraco, logo, certifique-se que seu IDE não tem essa opção ativada.

Para verificar / desativar o processamento paralelo, siga os passos abaixo:

• Vá em **File** (ou **Android Studio** caso esteja com um Mac);

• Depois em **Settings** (no Mac é **Preferences**);

• No menu esquerdo expanda **Build, Execution, Deployment**;

31

• Então clique em **Compiler**;

• Certifique-se de que a opção **Compile independent modules in parallel** está desmarcada, caso não, desmarque-a;

• Clique em **Apply** ou **Ok**.

Sempre utilize as versões mais atuais do Gradle e Android Plugin

O Android Studio vai lhe informar sobre essas atualizações, de qualquer forma, não deixe de sempre atualiza-las assim que informado, pois dessa maneira seu IDE obtém o melhor em desempenho que foi colocado na nova versão do Gradle e do Android Plugin para o Gradle.

Somente inclua dependências necessárias

É comum colocar no **build.gradle** de módulo de aplicativo uma ou outra dependência que depois de alguns testes deixa de ser utilizada. Um outro problema é colocar uma dependência muito genérica que acaba incluindo uma série de APIs que nunca serão utilizadas em seu projeto.

Quando trabalhando com APIs do Google Play Services isso tende a ocorrer caso você opte por utilizar a referência genérica, **'com.google.android.gms:play-services:11.2.0'**, quando na verdade precisaria somente da referente à Location API, **'com.google.android.gms:play-services-location:11.2.0'**, por exemplo.

Revise todas as dependências de seu projeto e seja específico nas declarações delas, somente utilize a versão genérica caso não tenha outro caminho para as APIs necessárias em seu app.

Utilize um device real

Evite o uso do emulador virtual em caso de lentidão, mesmo você configurando um número de memória disponível a ele abaixo dos comuns 1.5 GiB, ainda assim é recomendado que você utilize um device real via USB, como explicado anteriormente nas seções de emulador.

Com isso seu ambiente de desenvolvimento utilizará menos memória e será possível até mesmo de aproveitar esse não uso de emulador para liberar um pouco mais do que 512 MiB ao IDE em **studio.vmoptions**.

Caso não tenha um device real, trabalhe as configurações do emulador em **Advanced settings**, principalmente a memória, e então utilize ele.

Desative algumas configurações do Lint

O Lint é o responsável pelas verificações *on the fly* em seu editor de código no Android Studio, isso é pesado quando se tem pouca memória disponível.

Acesse a configuração dessa ferramenta para poder desativar o que não é necessário para você

32

Algumas outras melhorias

Todas as dicas anteriores são relacionadas a melhoria no uso da memória de sua máquina de ambiente de desenvolvimento, assim poderá acabar com a lentidão do IDE.

É possível também trabalhar a eficiência no uso da banda de rede. Veja ainda mais dicas no link a seguir: Configurar o IDE para máquinas com pouca memória.

Outras ferramentas

Caso queira se aprofundar ainda mais no Android Studio, isso sabendo que dominar este IDE vai lhe ajudar a ser um desenvolvedor Android ainda mais eficiente, a seguir listo algumas outras ferramentas que, ou já foram abordadas aqui no Blog e canal, ou ainda vamos falar sobre. Segue:

• InstantRun, para execuções rápidas, sem novas compilações;

• LogCat, ferramenta de live report;

• Android Profiler, para verificação de consumo de memória no device de testes; • VCS, para integração com aplicativos de controle de versão;

• Android Image Asset Studio, para criação dos ícones do projeto.

Não deixe de praticar o uso do IDE, pois assim é que você conseguirá fixar o aprendizado.

Conclusão

Apesar dos sérios problemas com lentidão em máquinas até mesmo com 8 GiB de memória, o Android Studio vem melhorando consideravelmente e se tornando mais eficiente a cada nova versão estável.

O IDE é sem sombra de dúvidas a melhor opção para aqueles que desenvolvem, ou vão desenvolver, aplicativos Android com as linguagens oficiais (Java, Kotlin, C, C++). Isso principalmente por não ser um IDE genérico atendendo somente a plataforma Android.

Podemos esperar melhorias consideráveis, digo, em termos de inclusão de código, mesmo sabendo que já é possível adicionar até mesmo as APIs do Firebase com poucos cliques.

Com a compra do Fabric pelo Google, provavelmente a integração com APIs populares e externas deverá ser ainda mais facilitada, além do já fácil acesso via referência em Gradle.

33