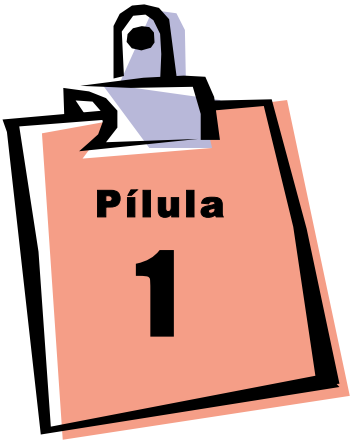


Conceitos Básicos e Práticos do Android

I
OBJETIVOS

- Conhecer o Sistema Operacional Móvel Android.
- Entender a composição do Android diferenciando suas funcionalidades.
- Conceber configuração básica do Android através de outras ferramentas de software.
- Manipular módulos básicos do Android.

II
TEMAS A TRATAR

- O que é o Adroid?
- Arquitetura Android.
- Componentes de uma Aplicação Android.
- Ciclo de Vida de uma Activity.
- Ferramenta Android SDK.
- Criando Primeiro Projeto.

III
MARCO TEÓRICO

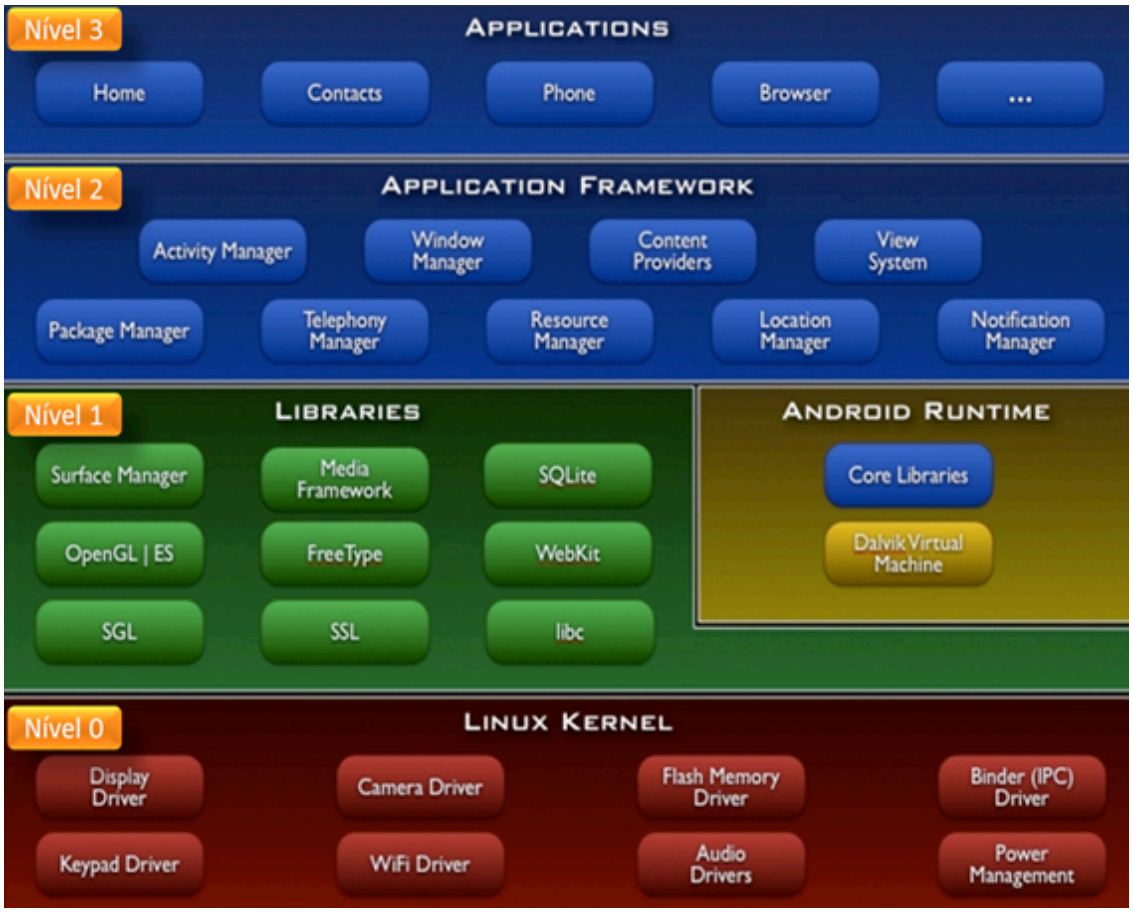
I. O QUE É O ANDROID?

O Android é um sistema operacional para dispositivos móveis, baseado em uma plataforma de código aberta sob a licença apache, permitindo que os fabricantes possam modificar seu código fonte, essa liberdade permite adicionar novos recursos e incorporá-los ao sistema, desta forma o software evolui constantemente. O sistema possui cerca de 12 milhões de linhas de código, sendo: 3 milhões em XML; 2.8 milhões em C; 2.1 milhões em Java; e 1.75 milhões em C++.

Ao contrário do que muita gente pensa, o Android não é desenvolvido apenas pela Google, existe um grupo de empresas como a Motorola, Samsung, LG, Sony Ericsson, HTC, China Mobile, Intel, Asus e muitas outras, que forma o Open Handset Alliance (OHA) com o objetivo de desenvolver a plataforma.

II. ARQUITETURA ANDROID

A arquitetura do sistema operacional Android é uma pilha de programas agrupados em camadas. Podemos dividir essas camadas em níveis zero, um, dois e três, conforme figura abaixo.



Arquitetura Android em níveis:

Nível Zero

No nível zero, temos a base da pilha, ou seja, o Kernel (Linux Kernel), para desenvolvê-la foi utilizado à versão 2.6 do Sistema Operacional Linux. Nele encontraremos os programas de gerenciamento de memória, configurações de segurança e vários drivers de hardware.

Nível Um

No nível um, temos as camadas de bibliotecas (Libraries) e tempo de execução (Android RunTime).
A camada de biblioteca é um conjunto de instruções que dizem ao dispositivo como lidar com diferentes tipos de dados, incluindo um conjunto de biblioteca C / C++ usadas por diversos componentes do sistema e são expostas a desenvolvedores através da estrutura de aplicativo Android.
A camada de tempo de execução inclui um conjunto de bibliotecas do núcleo Java (Core Libraries). Para desenvolver aplicações para o Android, os programadores utiliza a linguagem de programação Java, nesta camada encontraremos a Máquina Virtual Dalvik (DVM).
O Android usa a máquinas virtuais Dalvik para rodar cada aplicação com seu próprio processo. Isso é importante por algumas razões: nenhuma aplicação é dependente de outra e se uma aplicação parar, ela não afeta quaisquer outras aplicações rodando no dispositivo e isso simplifica o gerenciamento de memória, pois a máquina virtual está baseada em registradores e desenvolvida de forma otimizada para requerer pouca memória e permitir que múltiplas instâncias executem ao mesmo tempo.
Ao contrário do que se afirma, que a Dalvik é uma máquina virtual Java, isso não é verdadeiro, pois ela executa seu próprio tipo de bytecodes.

Nível Dois

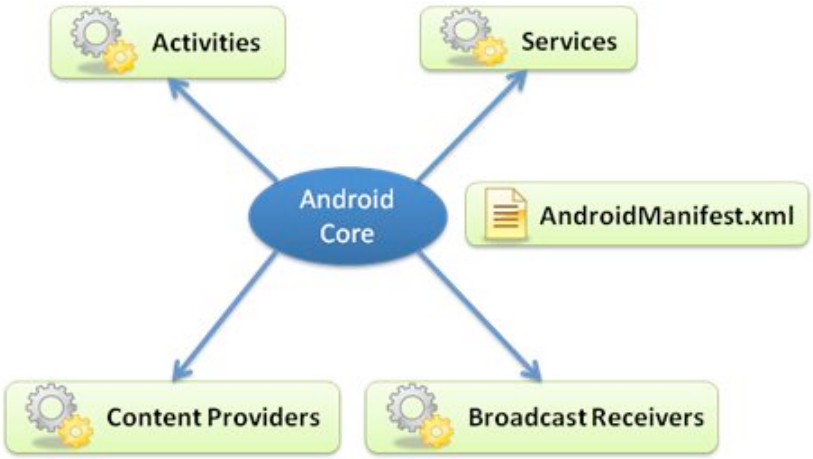
No nível dois, temos a camada de framework de aplicação (Application Framework), programas que gerenciam as aplicações básicas do telefone. Os desenvolvedores têm acesso total ao framework como um conjunto de ferramentas básicas com o qual poderá construir ferramentas mais complexas.

Nível Três

No nível três, temos a camada de aplicações e as funções básicas do dispositivo. Esta é a camada de interação entre o usuário e o dispositivo móvel, nela encontramos aplicativos cliente de e-mail, programa de SMS, calendário, mapas, navegador, contatos entre outros.

III. COMPONENTES DE UMA APLICAÇÃO ANDROID

Desenvolver aplicações para Android significa compor uma série de componentes para que o objetivo final da aplicação seja atingido. A figura a seguir mostra estes componentes.



Activities são as representantes das telas da aplicação. Associada a uma *activity* normalmente existe uma *view*, que define como será feita a exibição visual para o usuário. As *activities* são responsáveis por gerenciar os eventos de tela e também coordenam o fluxo da aplicação.

Services, são códigos que executam em segundo plano. Normalmente são utilizados para tarefas que demandam um grande tempo de execução.

Content Providers (provedores de conteúdos), são a maneira utilizada pela plataforma para compartilhar dados entre as aplicações que executam no dispositivo. Um exemplo bem claro disto é a aplicação de gerenciamento de contatos do Android, que é nativa. Aplicações desenvolvidas por terceiros podem utilizar um content provider a fim de ler os contatos armazenados no dispositivo de forma simples.

Broadcast Receivers, são componentes que ficam "escutando" a ocorrência de determinados eventos, que podem ser nativos ou disparados por aplicações. Uma aplicação pode, por exemplo, utilizar um *broadcast receiver* para ser avisada quando o dispositivo estiver recebendo uma ligação e, com base nessa informação, realizar algum tipo de processamento.

Junto os estes componentes, existe o arquivo de manifesto **AndroidManifest.xml**. Ele é obrigatório e único para cada aplicação. É nele que são feitas as configurações gerais da aplicação e dos componentes que fazem parte dela. E, juntando tudo isto, existe a figura do **Android Core**, que na verdade não é um componente específico, mas sim a plataforma Android

propriamente dita. É ele quem proporciona a interação entre os componentes e as aplicações e torna possível a execução do código.

Activity

Uma **Activity** (usarei o termo em inglês com letra maiúscula, cujo plural é **Activities**, para já irmos nos acostumando) é uma única "coisa" que o usuário pode fazer. Parece estranho ou óbvio demais, mas é isso mesmo! Uma **Activity** é uma atividade ou ação da sua aplicação.

Toda aplicação Android é formada por uma ou mais Activities e só uma pode rodar por vez. Quase todas as Activities possuem interação com o usuário, desse modo, a classe Activity do Android cuida da criação de uma janela na qual você (desenvolvedor) vai jogar sua UI com o método **setContentView(View)** (falaremos disso com mais detalhes na próxima aula). Mas isso não é tudo: apesar de a maioria das Activities construírem sua UI em uma tela cheia, há casos de janelas flutuantes (num tema com windowIsFloating setado) e Activities dentro de outra Activity (ActivityGroup).

Fragment

A partir da versão HONEYCOMB a classe Activity implementou a classe Fragment pensando na modularização, na construção de User Interface sofisticados para telas maiores e em ajudar a expandir a sua aplicação entre telas pequenas e grandes. De forma simples:

Imagine que sua User Interface não vem mais na Activity, mas em uma "coisa" separada e você pode chamar esse código em várias Activities de acordo com sua vontade (ou com o Ciclo de Vida da Aplicação). Esse é o conceito base dos Fragments (fragmentos). E essa gerência é feita através do FragmentManager (assunto que veremos na prática, mais para frente).

Mobile vs Desktop

Como Desenvolvedor Android, você deve entender que **um dispositivo móvel não possui a mesma capacidade de processamento de um Desktop**, por exemplo. Então, diferente das aplicações Desktop, que podem rodar simultaneamente (alternando apenas com o Alt+Tab), as Activities dificilmente serão executadas em simultâneo (nem mesmo no nosso querido Android, que é um

dos melhores sistemas operacionais para dispositivos móveis). É por isso, que temos a **Activity Stack** e o **Ciclo de Vida** de uma Activity.

Activity Stack

A forma mais fácil de identificar uma Activity é uma tela. Não é regra, mas geralmente se o App possui uma tela, possui uma Activity e duas telas significam 2 Activities. Além das Activities da sua aplicação, temos as Activities do próprio Android. E como (geralmente) só uma Activity é executa por vez, a **Activity Stack** ou **Pilha de Atividades** é quem faz esse gerenciamento.

Na Prática

Imagine um aplicativo de notícias. Estamos lendo uma notícia e essa "tela" representa uma Activity (em azul, na imagem abaixo), afinal toda a UI foi montada nela e os dados impressos na View. Ao terminar de ler, você quer compartilhar essa notícia, certo? Aí você *toca* no botão de Compartilhar e então uma nova Activity (em verde) surge sobre a anterior, mas ainda podemos vê-la por baixo (na web isso seria algo como uma lightbox ou um pop-up). Então você decide compartilhar por e-mail e ao escolher essa opção o App do Gmail abre por cima, ou seja, uma nova Activity (em laranja) foi para o topo da pilha.

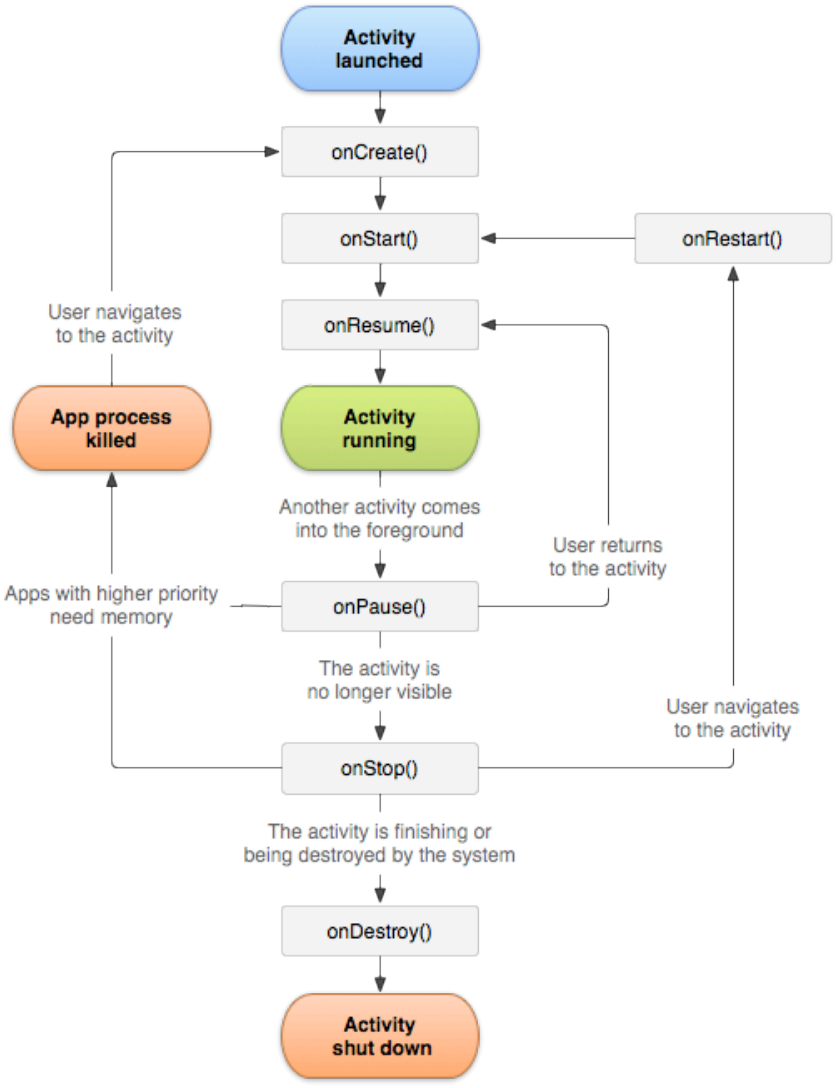
A escolha da Activity que está no topo é feita pela Activity Stack e toda essa mudança de estados faz parte do Ciclo da Vida da Activity. Sobre a Activity Stack já falamos, agora é hora do **Ciclo da Vida** ou **Life Cycle** de uma Activity no Android.

IV. CICLO DE VIDA DE UMA ACTIVITY

Na imagem anterior, a Activity 1 ficou no estado **pausado** (on Pause state) enquanto a Activity 2 apareceu, pois ela ainda era visível. No momento em que a Activity 3 foi chamada, a 2 deixou de aparecer, sendo assim ela mudou para o estado **parado** (on Stop state). Isso ocorreu também com a Activity 2, mas como ela já cumpriu seu papel, depois do**on Stop** ela vai para o estado de **destruída** (on Destroy state) e é finalizada.

Após o envio do e-mail (ou por ação do usuário) a Activity 1 pode voltar para o topo, sendo assim, ela iria ser **reiniciada** (onRestart ocorre), já que estava no estado parado. Se o usuário tivesse cancelado a ação de compartilhar (no segundo passo) a Activity 2 teria sido destruída e a 1 retornaria para o 1° plano saindo do estado pausado, então ela não seria mais reiniciada, mas sim **retomada** (onResume ocorre).

Esse ciclo pode ser visto melhor na imagem abaixo:



Sendo assim, podemos identificar 3 ciclos:

O **ciclo completo** de vida da Activity ocorre entre a primeira chamada no **onCreate(Bundle)** até a única chamada de**onDestroy()**. Sendo assim, uma Activity irá executar tudo o que for "global" no onCreate() e liberar todos os recursos no onDestroy().
O **ciclo visível** de vida da Activity ocorre entre o **onStart()** até o **onStop()** correspondente (depois de um *onStop*, podemos ter outro *onStart*, depois de um *onRestart*). Durante esse tempo o usuário pode ver a UI gerada pela Activity na tela, mesmo que não possa interagir. Entre esses dois métodos você pode manter os recursos necessários para mostrar a Activity para o usuário. Por exemplo, você pode registrar um BroadcastReceiver no **onStart()** para monitorar as mudanças de dados que impactam na sua UI (atualização constante de dados, por exemplo) e retirar esse registro no **onStop()**, quando o usuário não consegue mais ver sua UI. Isso pode acontecer várias vezes enquanto a Activity se torna visível e invisível para o usuário, assim você economiza recurso (o BroadcastReceiver), quando ele não for necessário.
O **ciclo de primeiro plano** da vida da Activity acontece entre o **onResume()** até o seu **onPause()** correspondente (a mesma coisa do *onStop* e *onStart*, mas dessa vez, não teremos um método no meio, como o *onRestart*, mas sim a Activity vai direto, como você viu na imagem). Durante esse tempo a Activity está no topo da tela e o usuário pode interagir com ela. Uma Activity pode ir frequentemente entre o estado Paused (pausado) e Resumed (resumido), por tal motivo o código aqui deve ser leve: nada de colocar todo o seu código aqui!

Os Métodos

Voltando à imagem (que eu tirei da documentação do Android, claro!) podemos ver que determinados métodos são chamados em cada mudança de estado e isso é legal, pois o Desenvolvedor pode manipular sua aplicação com isso. Por exemplo, ele pode salvar os dados antes da Activity ser parada ou destruída. Vamos ver cada método:

- **onCreate()** – Executado quando uma Activity é criada. Geralmente é o método responsável por carregar os layouts (XML) e outras operações de inicialização. Só é executado 1 vez durante o Clico de Vida da Activity.
- **onStart()** – É chamado logo depois do **onCreate()** ou quando a Activity que estava em background volta a ter foco. (Depois dela temos **onResume**, caso a Activity esteja em primeiro plano ou **onStop** se ela não for visível).

• **onResume()** – Método chamado quando a Activity vai começar a interagir com o usuário (ou retoma foco). Sempre vem depois do **onStart**, caso a Activity tenha sido iniciada ou se ela estava parada (**onStop**), ou pode ocorrer direto (vindo depois do **onPause**) caso a Activity, que não estava em primeiro plano, mas ainda visível, volte para o topo.

• **onPause()** – Ocorre logo antes da Activity perder o foco, ou seja, quando ir para o Background, mas não foi (ainda) destruída. Geralmente é usada para parar animações e recursos que estejam consumindo processamento e persistir dados não salvos, desta forma, se (por falta de recursos) a Activity for destruída, não perderemos essas informações. Tenha em mente que a Activity que está tomando o foco não será criada até que esse método retorne, sendo assim, o código aqui também deve ser leve.

• **onStop()** – Ocorre assim que a Activity deixa de aparecer totalmente. Pode ser seguida por **onRestart** caso a Activity volte ao topo ou por **onDestroy** caso ela passe dessa para uma melhor.

• **onRestart()** – É chamado quando uma Activity que estava parada volta ao foco. Bem antes do **onStart**.

• **onDestroy()** – Esse método ocorre logo antes da Activity ser destruída/finalizada (alguém chamou finish() ou o sistema está temporariamente a destruindo para salvar recursos, você pode verificar isFinishing() para distinguir essas situações).

Observação: Os métodos **onDestroy**, **onStop** e **onPause** (esse último apenas antes da versão HONEYCOMB) são marcados na documentação como "killable", ou seja, nesses estados a aplicação pode ser destruída pelo sistema sem executar nenhuma linha de código, sendo assim, use o método **onPause** para gravar todos os dados que devem ser persistidos, caso a aplicação seja destruída. Além disso, o método onSaveInstanceState(Bundle) é chamado antes desses casos para que alguma informação de estado (Bundle) seja guardada e possa ser resguardada usando o método **onCreate(Bundle)**, caso a Activity retorne. É um pouco complicado, mas você vai entender mais na prática. Enquanto isso, a leitura da documentação é válida.

V. FERRAMENTA ANDROID SDK

A maneira mais fácil para você começar a desenvolver aplicativos Android é fazer o download do Android SDK que é a principal ferramenta do desenvolvimento de aplicações baseadas em Android, esta é distribuída como um arquivo zip que é descompactado em um diretório em sua unidade de disco rígido. Como já houve várias atualizações do SDK, é recomendado que você manten. O SDK inclui:

android.jar

O arquivo Java archive contendo todas as classes do Android SDK necessárias para a construção do seu aplicativo.

documention.html e diretório de documentos

A documentação do SDK é fornecida localmente e na Web. Ela tem, em grande parte, forma de JavaDocs, facilitando a navegação em vários pacotes no SDK. A documentação também inclui um Guia de Desenvolvimento de alto nível e links para a comunidade mais ampla do Android.

Diretório de amostras

O subdiretório de amostras contém código de origem completo para uma variedade de aplicativos, incluindo ApiDemo, que exercita muitas APIs. O aplicativo de amostra é um excelente lugar para você explorar quando começar a desenvolver seu aplicativo Android.

Diretório de ferramentas

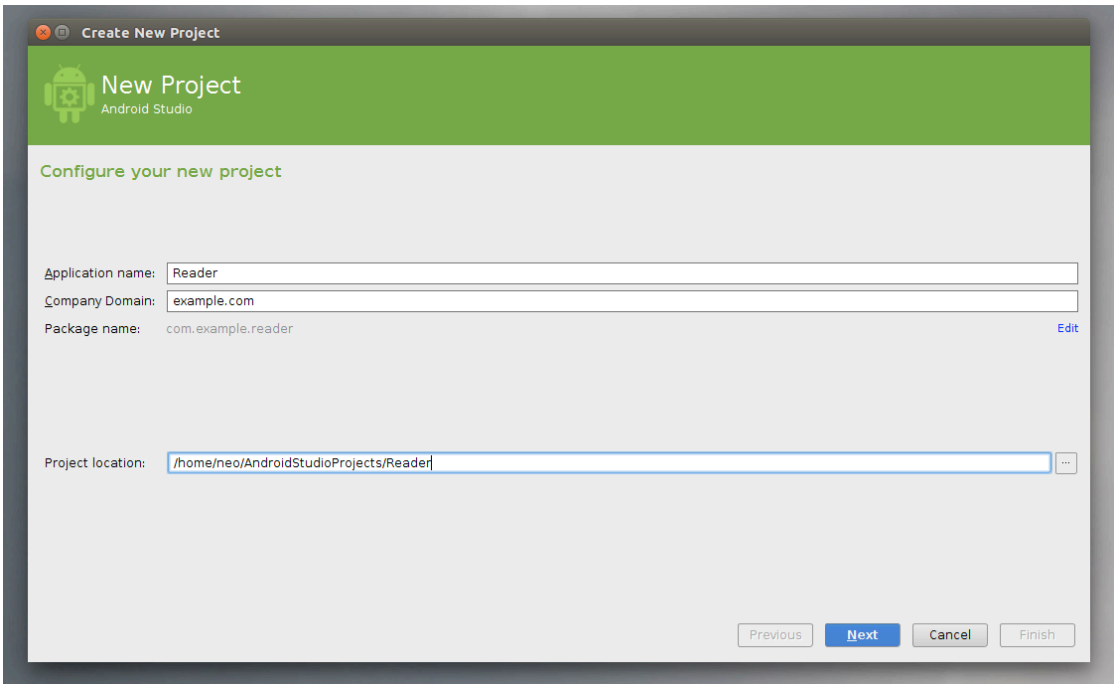
Contém todas as ferramentas de linha de comando para construir aplicativos Android. A ferramenta mais útil e usada com mais frequência é o utilitário adb (Android Debug Bridge).

usb_driver

Diretório contendo os drivers necessários para conectar o ambiente de desenvolvimento a um dispositivo ativado por Android, como G1 ou o telefone de desenvolvimento desbloqueado Android Dev 1. Esses arquivos só são necessários para desenvolvedores que utilizam a plataforma Windows.

VI. CRIANDO PRIMEIRO PROJETO

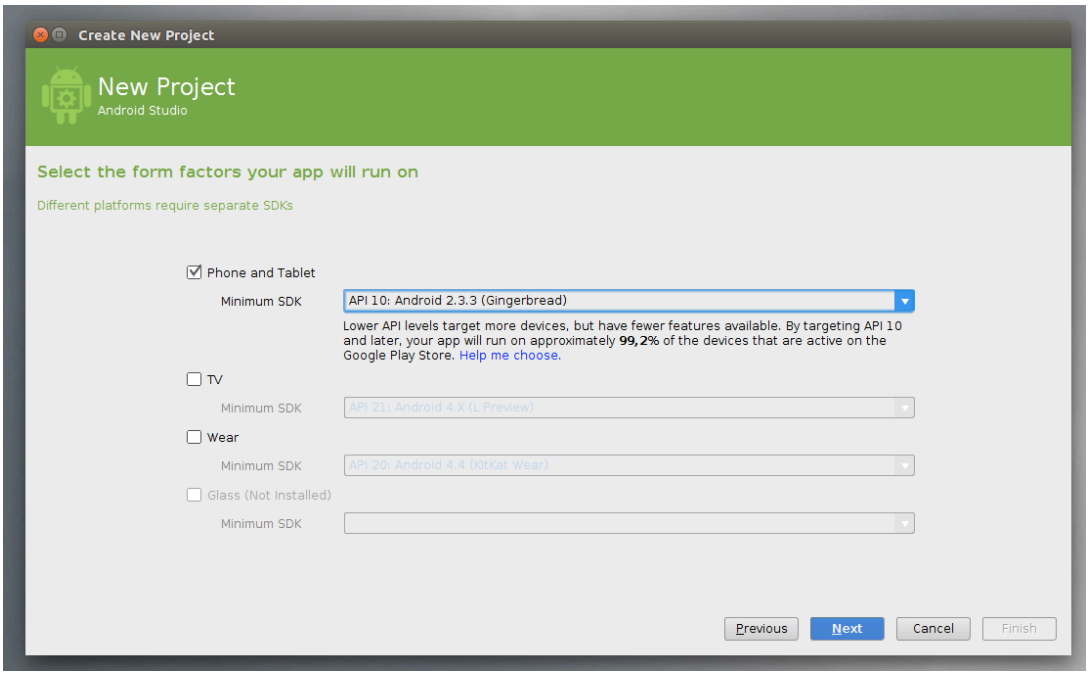
Com o Android Studio aberto, você precisa clicar em **New Project…** e a próxima tela será apresentada:



Essa é a hora de escolher o nome do aplicativo: **Application Name**.
Abaixo dele você escolhe precisa informar o nome da sua companhia. Eu informei **mariovalney.com** porque isso já irá gerar um **package name** bem legal, mas você pode colocar seu nome ou qualquer outro nome que ache legal.
O **Package name** já deve estar preenchido, mas você pode editá-lo independentemente das informações já escritas até agora. Lembre-se que ele deve ser **único** no sistema Android (principalmente se você for submeter seu aplicativo para o Google Play, pois todo mundo terá acesso a ele), além disso ele deve seguir as mesmas regras dos pacotes em Java.

É recomendável utilizar a url do seu site (quando você for desenvolver algum aplicativo vai querer divulgá-lo na internet, certo? Nada melhor do que ter um site para isso!) ou então **example.com**, já que estamos estudando.
Por último temos a localização do Projeto no seu computador.

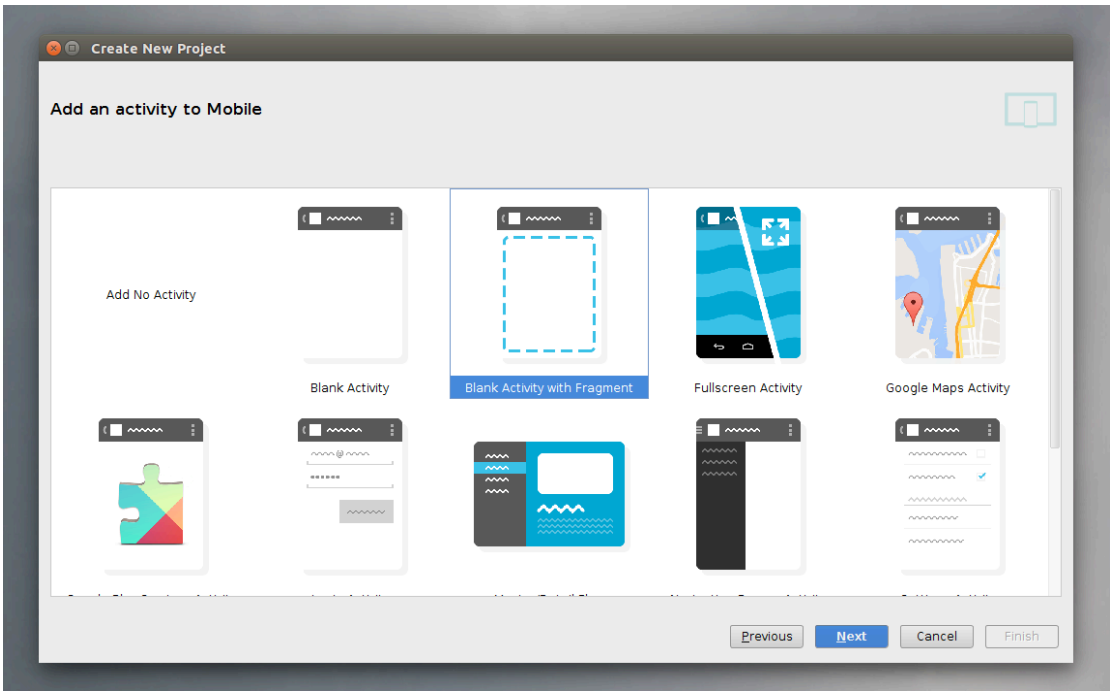
Com tudo OK basta clicar em **Next**.



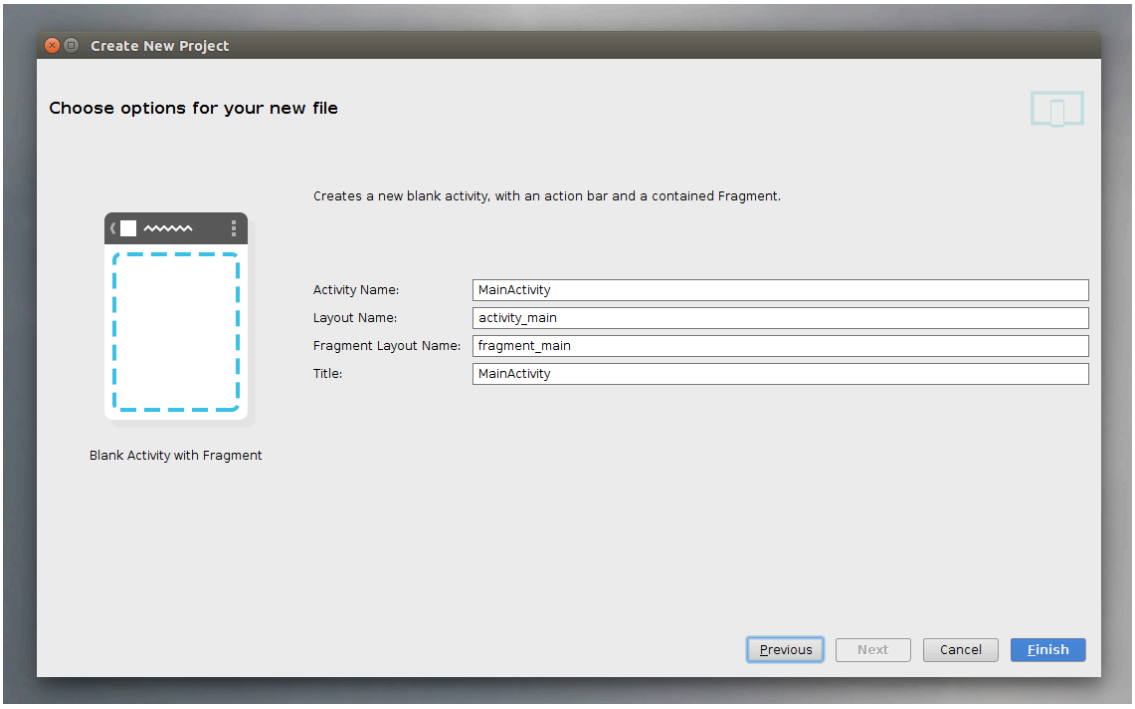
Na tela que apareceu, você precisa escolher para o que você vai desenvolver. No nosso caso, iremos desenvolver para celulares e tablets então marcamos a primeira checkbox.

Além disso, iremos escolher também qual a SDK mínima para a nossa aplicação (leia a próxima seção para entender mais, por enquanto, escolha a API 10).

Clique em **Next** de novo, para ir para a próxima tela:



O Android Studio facilita demais nossa vida e essa tela mostra bem isso: você pode escolher o que vai querer fazer e ele já cria algumas coisas. Por exemplo, meu projeto final foi um aplicativo com mapa (integração com a API do Google Maps) então eu escolhi **Google Maps Activity**. O **Reader** não terá isso e será mais simples do as outras sugestões dessa tela, então a melhor escolha será **Blank Activity with Fragment** (falaremos mais tarde sobre Activity e sobre Fragments, atividades e fragmentos).
Clique em **Next**, para prosseguir:



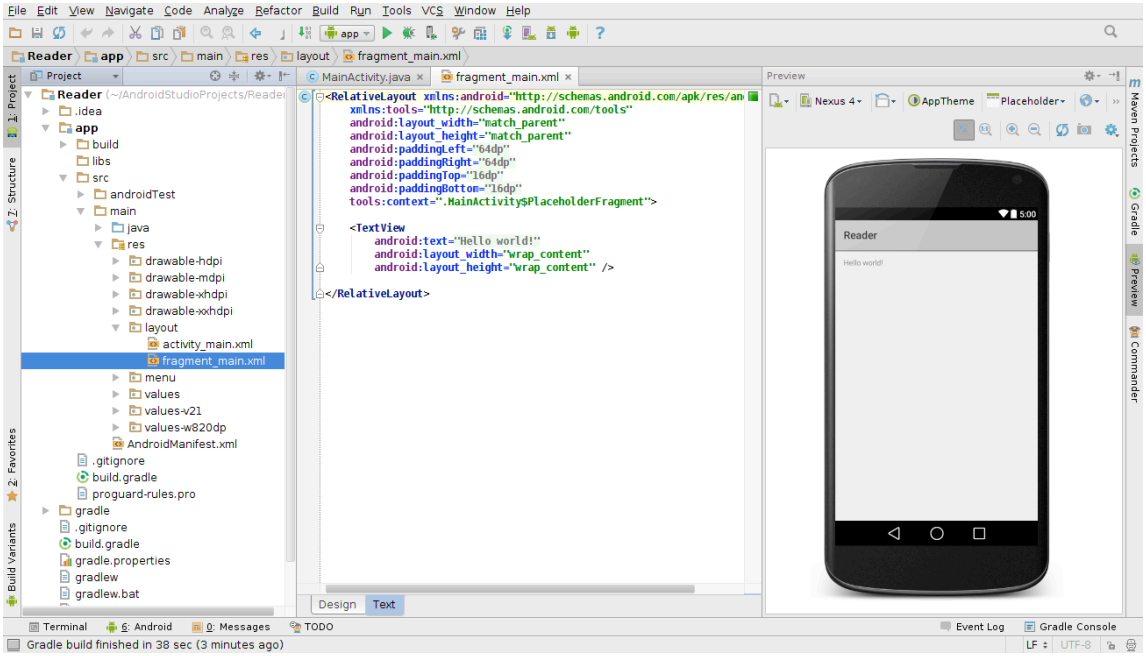
Chegou a hora de escolher o nome da sua primeira atividade (o Android Studio já vai criá-la), o nome do arquivo de layout da atividade e do fragmento.

Para a atividade, escolhemos **MainActivity**, pois será nossa atividade principal. Para o arquivo de layout, escolhemos **activity_main** e para o layout do fragmento **fragment_main**. Além disso o título da atividade também será MainActivity, por enquanto.
Iremos falar mais sobre layout, UI e a estrutura dos arquivos do Projeto depois. Contudo, já que estamos no clima, vou explica o que escolhemos ali em cima:

- O **nome da atividade** é também o nome da classe (e do arquivo java).
- O **layout** é um arquivo XML que fica numa pasta específica e será usado para o layout da atividade.

- O **layout do fragmento** também é um arquivo de layout, contudo será usado pelo fragmento.
- O **título da atividade** é o texto que ficará a ActionBar (barra de cima do app).

Agora é só clicar em **Finish** e esperar o Gradle terminar:



Pronto! Seu projeto está criado com um Hello World.

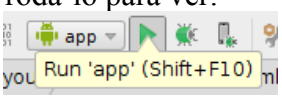
Minimum SDK

O arquivo de manifesto descreve detalhes sobre o seu app e identifica quais versões do Android ele suporta. Especificamente, podemos definir a SDK mínima **minSdkVersion** e a SDK alvo **targetSdkVersion**. Essa informação descreve, respectivamente, a última versão do Android que o seu app suporta e a versão mais atual na qual ele foi testado. Por padrão a *target SDK* deve ser a última, afinal é boa prática testar seu APP para a nova versão do Android toda vez que sair uma nova.

Rodando seu APP

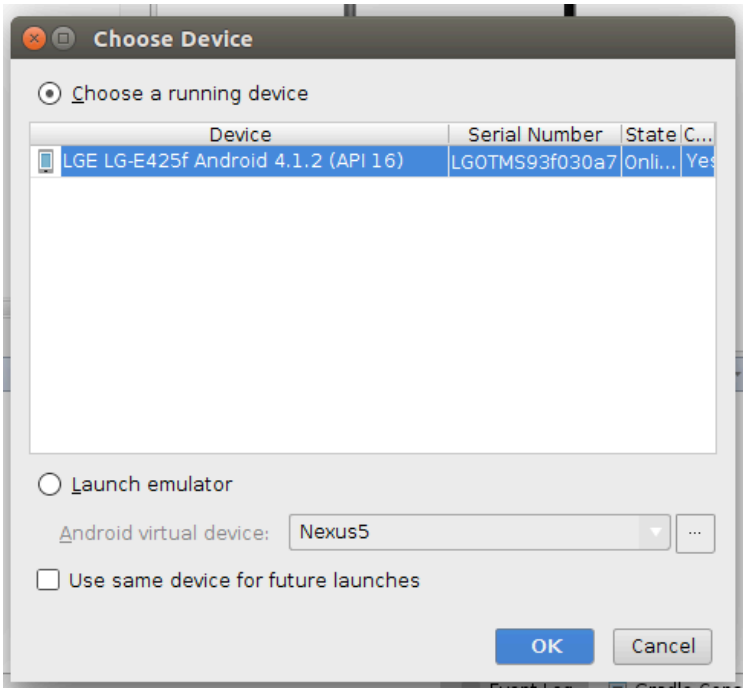
O Android Studio já cria um projeto com o **Hello World**. Iremos rodá-lo para ver:

Primeiro, vá na barra de ferramentas e clique no ícone de **Play**.



Na janela que irá abrir, você seleciona qual dispositivo irá rodar o aplicativo. O Android Studio vem com um emulador muito bom, mas iremos focar no desenvolvimento

utilizando um dispositivo físico, OK? Então é importante que tenha um celular rodando Android conectado através do cabo USB ao seu computador. Ele irá aparecer assim:



Se seu dispositivo **não aparecer** na listagem acima, provavelmente você ainda não o configurou para testes (leia a próxima seção antes de continuar).
Com o dispositivo selecionado, basta clicar em **OK** e esperar.
Se você não tiver um dispositivo rodando Android, você pode aprender sobre como usar o emulador do Android Studio aqui.

Ativando o DEV Mode

Você precisa ativar o modo de **depuração USB** do seu dispositivo para poder utilizá-lo para testes. Para isso, vá nas configurações e procure por **Developer Options** (Opções do Programador).
Não achou? Se você tem um dispositivo com Android 4.2 ou superior, essa opção estará escondida por padrão. Para achá-la, vá no menu About (Sobre) e toque 7 vezes na opção **Build Number** (fica no final).

Manifesto

O **AndroidManifest.xml** é o arquivo mais importante do seu aplicativo. Duvida? É nele que o sistema encontra o **nome** do aplicativo, **ícone** principal, nome do pacote, as atividades (activities) do app, as permissões (lembra que você precisa autorizar algumas coisas, quando instala alguns aplicativos?), a lista de serviços e outras informações relevantes.

O manifesto do Android é um arquivo XML, sendo assim, segue a estrutura normal:

```
1   <?xml version="1.0" encoding="utf-8"?>
2
3   <manifest>
4
5       <uses-permission />
6       <permission />
7       <permission-tree />
8       <permission-group />
9       <instrumentation />
10      <uses-sdk />
11      <uses-configuration />
12      <uses-feature />
13      <supports-screens />
14      <compatible-screens />
15      <supports-gl-texture />
16
17      <application>
18
19          <activity>
20              <intent-filter>
21                  <action />
22                  <category />
23                  <data />
24              </intent-filter>
25              <meta-data />
26          </activity>
27
28          <activity-alias>
29              <intent-filter> . . . </intent-filter>
30              <meta-data />
31          </activity-alias>
32
33          <service>
34              <intent-filter> . . . </intent-filter>
35              <meta-data/>
36          </service>
37
38          <receiver>
39              <intent-filter> . . . </intent-filter>
40              <meta-data />
41          </receiver>
42
43          <provider>
44              <grant-uri-permission />
45              <meta-data />
46              <path-permission />
47          </provider>
48
49          <uses-library />
50
51      </application>
52
53  </manifest>
```

IV

(O trabalho prático tem uma duração de 1 hora) **ATIVIDADES**

01. Crie um novo projeto para aplicação Mobile intitulado pelo seu nome pessoal. A função do aplicativo será apresentar na tela do dispositivo seus dados a seguir: (a) Nome completo, (b) Data de Nascimento e (c) Bairro de Residência.

V

PERGUNTAS

1. ¿Qual é a vantagem do Android ter seu código aberto?
2. ¿O que é uma Activity?
3. ¿O que contém o arquivo Manifesto?