



Treinamento    Android    Build    Apps  
Conteúdo Teórico/Prático

www.trainning

## Sumário

Arquitetura da plataforma .....	5
Kernel do Linux.....	5
Camada de abstração de hardware (HAL).....	6
Android Runtime .....	6
Bibliotecas C/C++ nativas .....	7
Estrutura da Java API.....	7
Aplicativos do sistema.....	8
Padrões de Desenvolvimento na plataforma Android.....	10
Separação de conceitos.....	10
Arquitetura multicamadas .....	10
Model-View-Controller (MVC) .....	12
Model-View-Presenter (MVP) .....	13
Definição das camadas:.....	14
Versões Android - Timeline .....	15
Fundamentos de aplicativos .....	25
Componentes de aplicativo.....	27
Activity.....	27
Services.....	27
Content Provider .....	28
Broadcast Receiver.....	28
Ativação de componentes.....	30
O arquivo Manifest.xml.....	31
Declaração de componentes.....	32
Declaração de recursos de componentes .....	33
Recursos do aplicativo.....	34
A IDE Android Studio .....	35
Sistema de compilação Gradle .....	41
Ferramentas de depuração e criação de perfil .....	42
Depuração em linha .....	42
Monitores de desempenho.....	43
Módulos .....	44
Arquivos de projetos .....	47
Configurações de estrutura de projetos .....	50

Developer Services .....	51
Módulos .....	51
Atualizar o IDE e o SDK Tools .....	52
Pacotes recomendados .....	54
Gradle para Android.....	57
O arquivo de configurações do Gradle.....	59
Arquivos de propriedade do Gradle.....	61
Como sincronizar o projeto com arquivos do Gradle .....	62
Criar projeto .....	64
Dispositivo virtual (Emulador).....	73
Layouts .....	81
Programação do XML.....	82
Carregamento do recurso XML.....	82
Atributos.....	83
ID .....	83
Parâmetros do layout.....	84
Posição do layout .....	85
Tamanho, preenchimento e margens.....	85
Layouts comuns.....	86
Criação de layouts com um adapter .....	87
Atributos de Layout.....	88
Identificação de um componente dentro de uma View .....	91
Aplicativos – criação.....	92
Comece um novo projeto.....	92
Importar um projeto .....	94
<b>Controles de entrada .....</b>	<b>94</b>
LinearLayout.....	96
Projeto LinearLayout .....	98
Relative Layout .....	104
Projeto Relative Layout .....	105
ListView (Visualização em lista).....	110
Projeto ListView .....	111
GridView Layout .....	113

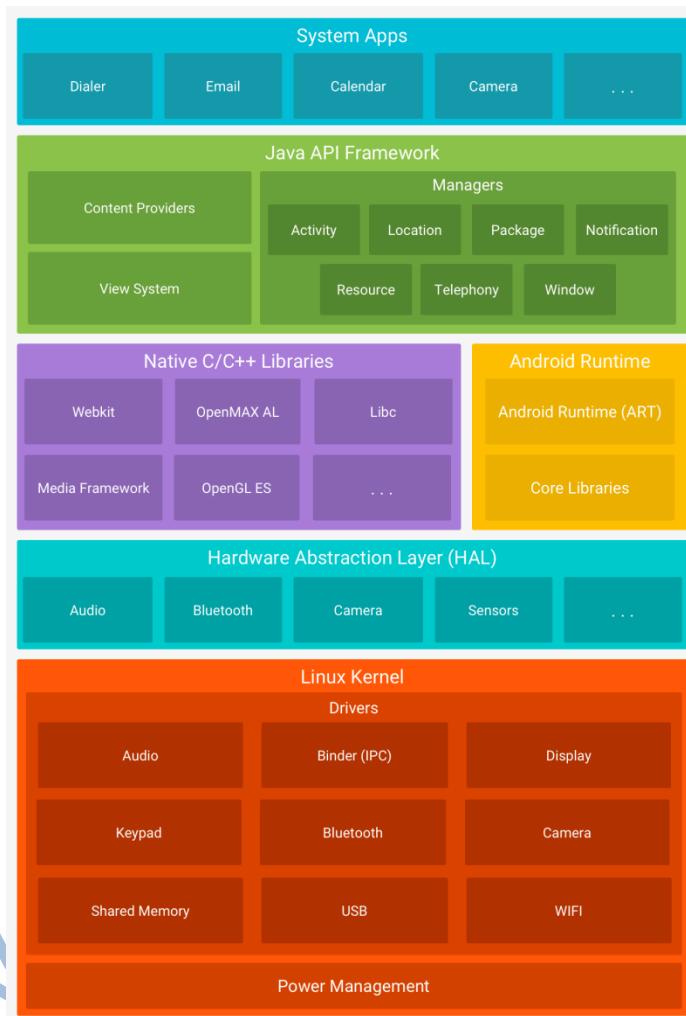
Projeto GridView:.....	113
RecyclerView .....	119
Animações.....	120
Adicionar dependências.....	121
Projeto Navigation Drawer.....	121
TabView Layout.....	138
Projeto Tab Layout .....	138
CardView .....	146
Projeto Card Layout.....	147
Acessando a câmera.....	158
Projeto Camera Access.....	160
Captura de Audio .....	166
Projeto Audio Capture.....	168
Android Services.....	173
Projeto Service .....	176
Aplicando BroadCastReceiver .....	181
Criando BroadCast Receivers .....	182
Registrando BroadcastReceiver .....	182
Broadcasting Custom Intents .....	183
Projeto BroadCastRec .....	184
SQLite Utilizando Content Provider .....	187
Banco de dados - Package.....	187
Banco de Dados - Criação.....	187
Banco de Dados - Inserção .....	188
Banco de dados - Fetching .....	189
Base de Dados – Helper Class.....	190
Content Provider.....	190
Content URIs .....	191
Criar Content Provider .....	192
Content Provider .....	193
Projeto SQLite Content Provider.....	193
Notification Android.....	200
Implementação passo a passo .....	201

Consumo de APIs - Retrofit .....	206
Criando Novo Projeto.....	207
Referencias.....	212

www.trainning.com.br

## Arquitetura da plataforma

O Android é uma pilha de software com base em Linux de código aberto criada para diversos dispositivos mobile. O diagrama a seguir mostra a maioria dos componentes da plataforma Android.



A pilha de software do Android.

### Kernel do Linux

A base da plataforma Android é o kernel do linux. Por exemplo: o Android Runtime (ART) confia no kernel do Linux para cobrir funcionalidades como encadeamento e gerenciamento de memória de baixo nível.

Usar um kernel do Linux permite que o Android aproveite os recursos de segurança principais e que os fabricantes dos dispositivos desenvolvam drivers de hardware para um kernel conhecido.

## Camada de abstração de hardware (HAL)

A camada de abstração de hardware (HAL) fornece interfaces padrão que expõem as capacidades de hardware do dispositivo para a estrutura da Java API de maior nível. A HAL consiste em módulos de biblioteca, que implementam uma interface para um tipo específico de componente de hardware, como o módulo de câmera ou bluetooth. Quando uma Framework API faz uma chamada para acessar o hardware do dispositivo, o sistema Android carrega o módulo da biblioteca para este componente de hardware.

## Android Runtime

Para dispositivos com Android versão 5.0 (API nível 21) ou mais recente, cada aplicativo executa o próprio processo com uma instância própria do Android Runtime (ART). O ART é projetado para executar várias máquinas virtuais em dispositivos de baixa memória executando arquivos DEX, um formato de bytecode projetado especialmente para Android, otimizado para oferecer consumo mínimo de memória. Construa cadeias de ferramentas, como Jack, e compile fontes Java em bytecodes DEX, que podem ser executadas na plataforma Android.

- Alguns dos recursos principais de ART são:
- Compilação "ahead-of-time" (AOT) e "just-in-time" (JIT)
- Coleta de lixo (GC) otimizada

Melhor compatibilidade de depuração, inclusive um gerador de perfil de exemplo, exceções de diagnóstico detalhadas e geração de relatórios de erros, além da capacidade de definir pontos de controle para monitorar campos específicos

Antes do Android versão 5.0 (API nível 21), o Dalvik era o tempo de execução do Android. Se o seu aplicativo executa o ART bem, deve funcionar no Dalvik também, mas talvez não vice-versa.

O Android também contém um conjunto das principais bibliotecas de tempo de execução que fornecem a maioria da funcionalidade da linguagem de programação Java, inclusive alguns recursos de linguagem Java 8 que a estrutura da Java API usa.

## Bibliotecas C/C++ nativas

Vários componentes e serviços principais do sistema Android, como ART e HAL, são implementados por código nativo que exige bibliotecas nativas programadas em C e C++. A plataforma Android fornece as Java Framework APIs para expor a funcionalidade de algumas dessas bibliotecas nativas aos aplicativos. Por exemplo, é possível acessar OpenGL ES pela Java OpenGL API da estrutura do Android para adicionar a capacidade de desenhar e manipular gráficos 2D e 3D no seu aplicativo.

Se estiver desenvolvendo um aplicativo que exige código C ou C++, você pode usar o Android NDK para acessar algumas dessas bibliotecas de plataforma nativa diretamente do seu código nativo.

## Estrutura da Java API

O conjunto completo de recursos do SO Android está disponível pelas APIs programadas na linguagem Java. Essas APIs formam os blocos de programação que você precisa para criar os aplicativos Android simplificando a reutilização de componentes e serviços de sistema modulares e principais, inclusive:

Um sistema de visualização rico e extensivo útil para programar a IU de um aplicativo, com listas, grades, caixas de texto, botões e até mesmo um navegador da web incorporado

Um gerenciador de recursos, fornecendo acesso a recursos sem código como strings localizadas, gráficos e arquivos de layout

Um gerenciador de notificação que permite que todos os aplicativos exibam alertas personalizados na barra de status

Um gerenciador de atividade que gerencia o ciclo de vida dos aplicativos e fornece uma pilha de navegação inversa

Provedores de conteúdo que permite que aplicativos accessem dados de outros aplicativos, como o aplicativo Contatos, ou compartilhem os próprios dados

Os desenvolvedores têm acesso completo às mesmas Framework APIs que os aplicativos do sistema Android usam.

## Aplicativos do sistema

O Android vem com um conjunto de aplicativos principais para e-mail, envio de SMS, calendários, navegador de internet, contatos etc. Os aplicativos inclusos na plataforma não têm status especial entre os aplicativos que o usuário opta por instalar. Portanto, um aplicativo terceirizado pode se tornar o navegador da Web, o aplicativo de envio de SMS ou até mesmo o teclado padrão do usuário (existem algumas exceções, como o aplicativo Configurações do sistema).

Os aplicativos do sistema funcionam como aplicativos para os usuários e fornecem capacidades principais que os desenvolvedores podem acessar pelos próprios aplicativos. Por exemplo: se o seu aplicativo quiser enviar uma mensagem SMS, não é necessário programar essa funcionalidade — é possível invocar o aplicativo de SMS que já está instalado para enviar uma mensagem ao destinatário que você especificar.

Maioria dos apps Android possui arquitetura incorreta. Muitas não estão otimizadas para o melhor desempenho e podem usar menos o equipamento de radiocomunicações e as transferências de dados

O desempenho na rede é uma das coisas mais importantes que a maioria dos apps faz mal", afirmou o programador da Google, Colt McAnlis, durante a conferência Google I/O, realizada na semana passada. Quase todas têm arquiteturas incorretas para essas operações, explicou McAnlis a uma multidão de programadores.

Ao estruturar a forma como as aplicações móveis acessam as redes de maneira ineficiente, lembrou McAnlis, impõem-se aos usuários custos desnecessários em termos de desempenho e duração da bateria. "As más operações em rede custam dinheiro", enfatizou.

"O usuário paga um preço alto por cada pedido falso que o app faz, todos os pacotes dessincronizados, cada imagem de dois bits solicitada desnecessariamente", ilustrou McAnlis.

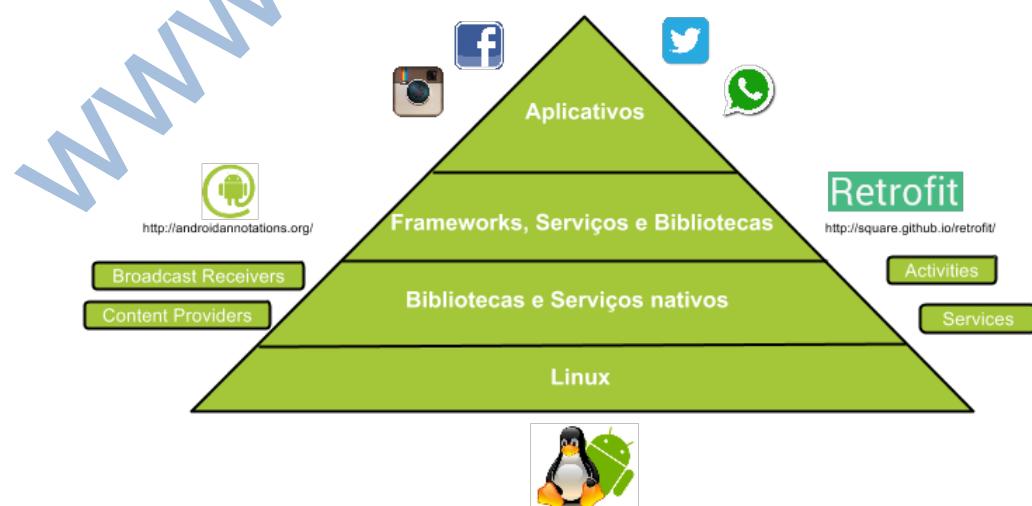
O programador propõe uma solução: usar menos o equipamento de radiocomunicações e as transferências de dados. Uma maneira de suportar isso é fazer “loteamentos” de dados.

Ou seja, arquitetar a aplicação de modo que aos dados de baixa prioridade sejam enviados quando o hardware de rede do dispositivo for ativado por outro sistema, minimizando a quantidade de tempo e energia utilizada na comunicação por rádio. Desencadear processos de pré busca de dados é outra técnica importante, para suavizar a utilização da rede pelas aplicações Android, assinalou.

“Quando a aplicação ‘sabe’ que vai fazer seis ou sete pedidos no futuro, não deve esperar que o dispositivo ‘adormeça’ para depois acordá-lo novamente”, ironiza o programador. Para ele, faz mais sentido tirar proveito do fato de o chip estar em atividade e fazer logo os pedidos.

McNalis sugeriu aos programadores a utilização do serviço Cloud Messaging, do Google, para as atualizações, em vez do sistema de hierarquização de servidores. “Este é um desperdício de tempo para o usuário”, alerta.

Segundo o programador, “cada vez que um app consulta o servidor e retribui um pacote nulo, dizendo que não há novos dados, o usuário paga por isso”.



## **Padrões de Desenvolvimento na plataforma Android**

Dentro da plataforma Android é possível desenvolver aplicativos considerando alguns padrões de desenvolvimento. O mais comum é o padrão MVC (Model-View-Controller). No entanto, um outro padrão é, também, comumente usado para o desenvolvimento de aplicativos; esse padrão chama-se: MVP(Model-View-Presenter).

Obs: não há nenhuma semelhança com a outra sigla MVP(Minimal Value Proposition – Proposta Minima de Valor)

Apesar do MVP ter se tornado mais conhecido recentemente devido a aplicação mais adequada a APPs mobile, digo, mais adequada em frente ao uso do conhecido MVC, o MVP veio bem antes do desenvolvimento mobile que temos hoje.

### **Separação de conceitos**

O termo Separação de Conceitos, a princípio, foi cunhado primeiramente por Edsger Wybe Dijkstra. Dijkstra foi pioneiro em várias áreas da computação.

Resumidamente Dijkstra disse que o estudo isolado de um aspecto, um conceito, mesmo quando sabendo da importância dos outros e ainda assim os mantendo como irrelevantes para o conceito atual em estudo, esse comportamento é um comportamento inteligente, pois assim é possível se aprofundar no assunto, aspecto, e decifra-lo, resolve-lo, de maneira eficiente.

Dijkstra citou isso ainda na década de 70, naquela época os softwares já tinham problemas em serem "amontoados". Separação de conceitos foi o primeiro passo para os estudos da construção de softwares em camadas.

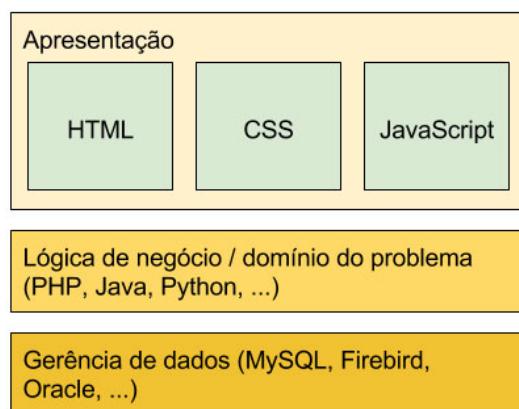
### **Arquitetura multicamadas**

O tema arquitetura em multicamadas não possui uma definição explícita de quais são as camadas que certamente devem ter em um software, somente que é uma melhor opção defini-lo, em camadas, nem mesmo o número mínimo e máximo é seguro informar.

Com a divisão do software em layers programadores iniciam seu desenvolvimento com foco em uma “*clean software architecture*” (software de arquitetura limpa).

Arquitetura limpa em aplicativos Android é algo muito maior do que somente a utilização do MVP. Esse assunto já vem sendo discutido a algum tempo e há excelentes artigos sobre.

A seguir um exemplo comum de arquitetura multicamadas, a arquitetura de um Web site:



Com o software em camadas é possível melhorar, em inúmeros aspectos, a reutilização de código e manutenção e evolução dele. Ressaltando que o trabalho de softwares em camadas é algo que complementa a estratégia de código limpo juntamente com técnicas de escrita limpa de algoritmos e com o uso de padrões de projeto.

Importante notar que o desenvolvimento em camadas não é uma exclusividade da programação orientada a objetos, qualquer paradigma de desenvolvimento pode trabalhar nessa linha.

Outro ponto importante é que as camadas mais acima na arquitetura tendem a ser dependentes das camadas abaixo, sempre: ou somente da próxima camada abaixo ou da próxima e de mais algumas.

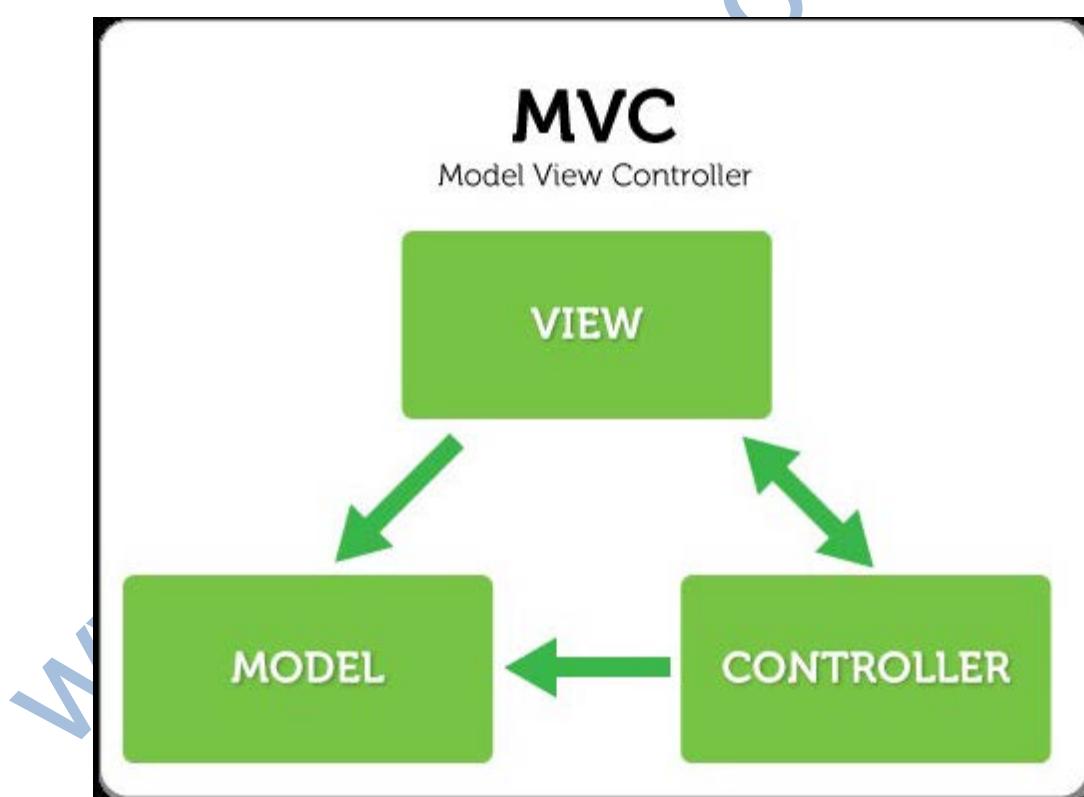
Assim podemos prosseguir a dois conhecidos e muito utilizados padrões de arquitetura: MVC (somente para relembrar) e MVP.

### Model-View-Controller (MVC)

MVC é um padrão de arquitetura e provavelmente um dos padrões da computação mais conhecidos, isso ao lado do Singleton e do Factory. Note que um padrão de arquitetura tende a ser definido antes do início da codificação, tanto que não é comum encontrarmos métodos de refatoração para esse tipo de padrão.

O MVC foi criado com o objetivo de dividir as camadas que estavam fortemente acopladas a camada de visualização de softwares gráficos, ou seja, veio para resolver um problema com a interface de usuário.

Este padrão é bem simples. Ele está mencionado aqui para que seja possível entender a origem do MVP e compará-lo ao MVC. Abaixo o diagrama comum do MVC:



As camadas:

View (Visualização): contém as entidades gráficas, entidades que permitem a saída de dados (tendo como origem o Model ou o Controller) e a entrada de dados (usuário);

Controller (Controlador): camada responsável por aceitar as entradas de dados (View) e também as saídas (Model) e trabalha-las para que estas cheguem corretamente as camadas de destino;

Model (Modelo): responsável pela lógica e domínio do problema, incluindo a manipulação de dados, mesmo que em seu princípio esse último aspecto não era tratado, persistência de dados.

Quando se tratando de padrão de arquitetura, mais precisamente do padrão MVC, na figura acima, é apenas uma representação dele, pois é possível e viável algumas vezes, balancear a lógica entre o Model e o Controller, por exemplo, para melhorar a comunicação entre as camadas ou até mesmo para obter maior performance sem sair do padrão de arquitetura escolhido.

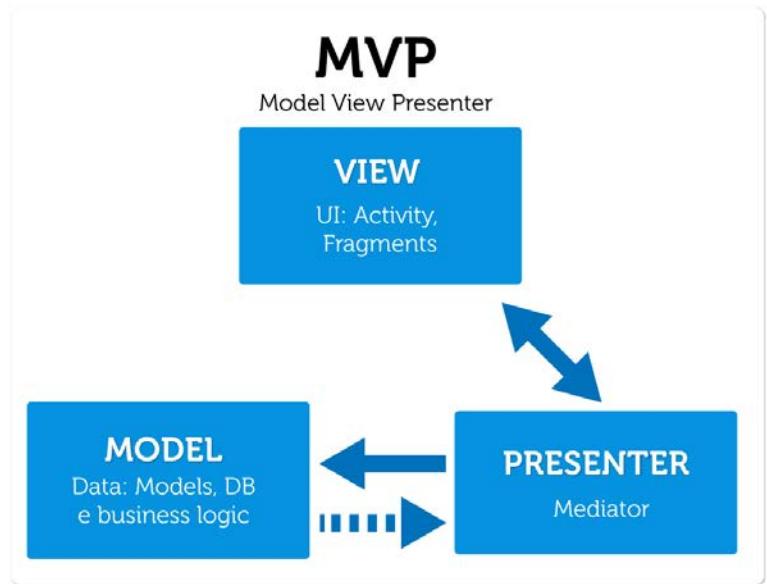
Com isso é possível seguir com o MVP, o derivado do MVC.

### **Model-View-Presenter (MVP)**

O MVP não existe devido a uma adaptação necessária do MVC ao Android, pois o MVP é muito anterior ao Android, precisamente, da década de 90, onde a união Apple, HP e IBM fez com esse padrão de arquitetura surgisse devido a necessidades que o MVC não cobria.

Alguns conteúdos informavam que o MVP não era um padrão de arquitetura. Provavelmente é possível encontrar essa informação em outros artigos.

O MVP(Model-View-Presenter) permite uma divisão melhor em camadas quando o ponto crítico é isolar camadas superiores de camadas inferiores. Necessariamente, permitir que a camada diretamente relacionada com a interface do usuário somente se comunique (seja dependente) com a camada diretamente abaixo dela, aqui a camada Presenter. O diagrama ilustra essa dependência:



### **Definição das camadas:**

View (Visualização): como no MVC, responde a saída e entrada de dados, porém a saída vem do Presenter, a entrada normalmente vem do usuário;

Presenter (Apresentador): Camada responsável por responder as invocações da camada de visualização e invocações da camada de modelo, além de também poder invocar ambas as camadas. O Presenter trabalha a formatação dos dados que entram em ambas as camadas paralelas e também pode incluir parte da lógica de negócio que alguns programadores podem pensar que deveria estar somente na camada de modelo;

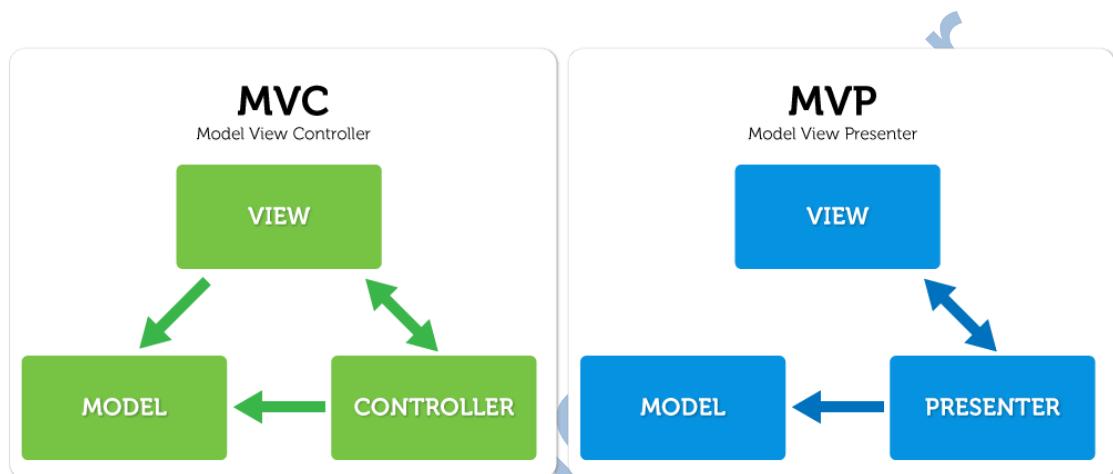
Model (Modelo): camada fornecedora de dados além de conter a lógica de negócio do domínio do problema.

A maior diferença entre o MVP e o MVC é a não-possibilidade de comunicação direta entre a camada View e a camada Model. Mesmo que com o objetivo de dividir o software em camadas, há um problema imenso na evolução do software quando essas camadas podem se comunicar diretamente.

Porém, como acontece com o MVC, no MVP também podemos balancear as coisas. Parte da lógica de negócio pode sim ser movida para a camada Presenter.

A recomendação é utilizar lógica sempre abaixo da camada de visualização, mesmo sabendo que seu balanceamento na aplicação do MVP, em seu software, pode induzi-lo a colocar ao menos alguns trechos de lógica na camada de visualização. Evite isso.

O diagrama a diferenças entre os padrões:



## Versões Android - Timeline

Desde o seu início, em 2008, até agora o sistema operacional da Google já recebeu 15 versões, a última delas foi lançada em agosto deste ano: o Android Oreo ou 8.0. Essa é uma característica muito interessante: todas as versões recebem nomes de doces, seguindo por ordem alfabética. Quando questionado o motivo, a Google se esquiva do assunto e continua mantendo segredo.



Essa é a linha do tempo das versões do sistema operacional Android – a partir da versão 1.5.

### **Android cupcake (1.5)**



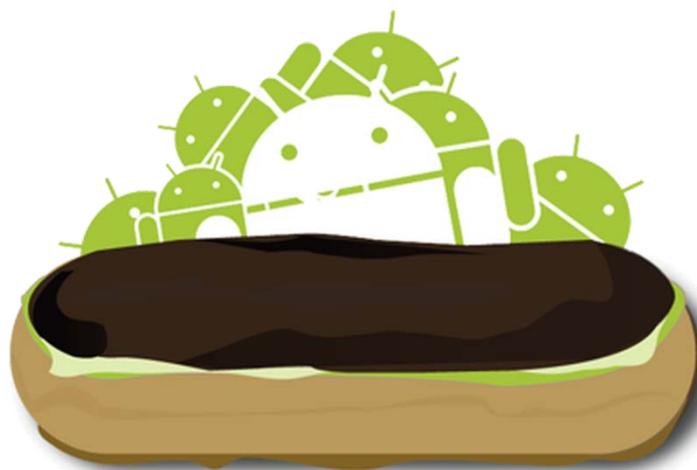
O Cupcake foi a primeira versão oficial do sistema (não é à toa que o nome começa com a letra “C”, a terceira do alfabeto). O 1.5 era extremamente básico, comparado com as versões de agora e tinha as funções de copiar e colar textos, usar widgets, Youtube e algumas animações básicas, além de gravar vídeos com a câmera. O Cupcake foi descontinuado assim que as versões mais novas chegaram ao mercado e hoje não tem mais smartphones para comprar com esse sistema.

### **Android donut (1.6)**



Esta versão foi lançada no final de 2009 e contava com mais telas e a resoluções maiores, até 480x800 pixels. Além disso, o sistema ganhou uma nova interface para os aplicativos de câmera e recursos de pesquisa por voz.

### **Android eclair (2.0 – 2.01)**



O doce “Eclair” no Brasil é conhecido como bomba de chocolate, uma massa fofinha, com recheio cremoso e coberta por calda de chocolate endurecida. Esta versão do sistema foi lançada há apenas um mês do Android Donut e trouxe muitas mudanças, entre elas o multi-touch, planos de fundo animados, maior contraste e resoluções de tela e claro, novas funcionalidades da câmera.

### **Android froyo (2.2)**



**Froyo**  
Android 2.2/2.2.3

O Froyo ou Frozen Yogurt, foi lançado na metade de 2010 e a sua principal característica era a velocidade do sistema, que estava 450% mais rápido. A partir desta versão o Android começa a ficar mais evoluído, com capacidade de compartilhar conexão 3G, WiFi ou USB, atualizações automáticas dos aplicativos e claro, a grande novidade: o suporte da tecnologia Flash, da Adobe. Hoje o FroYo é uma das versões mais populares do mercado.

#### **Android gingerbread (2.3)**



Lançada no final de 2010, essa versão do sistema era mais fácil e rápida de utilizar, dando maior suporte para a câmera frontal. Além disso, os smartphones com Android Gingerbread vinham com NFC, uma tecnologia que permite a troca de informações sem fio e de forma segura entre dispositivos que estejam próximos um do outro.

#### **Android honeycomb (3.0 – 3.1 – 3.2)**



Esta versão foi lançada em 2011 exclusivamente para tablets, por isso tinha um poder de processamento maior que as outras edições. O Honeycomb foi desenvolvido para dar suporte às telas maiores, além de permitir abas no navegador e duas barras: uma em cima, para gerenciar o sistema e outra em baixo com relógio e botões de atalho. A câmera e os aplicativos também receberam ajustes.

#### **Android ice cream sandwich (4.0)**



**ANDROID 4.0**  
Ice Cream Sandwich

O Ice Cream Sandwich foi Lançado em novembro de 2011, mas foi anunciado juntamente com o Google Music. Essa versão veio para unificar os tablets e smartphones em uma única versão do Android, em vez de operarem em versões separadas. As novidades do sistema foram: controle de tráfego na internet, edição simples de vídeos e fotos, destravamento de tela por reconhecimento facial, entre outros recursos.

### Android jelly bean (4.1 – 4.2 – 4.3)



O Jelly Bean trouxe um novo momento do Android, marcado pelo design moderno e sistema com melhor desempenho. Ele foi lançado em 2012 e contemplou uma série de novidades como widgets na tela de bloqueio, notificações com botões de ação, atalhos na área de notificações e muito mais. Foi esta versão do sistema que inaugurou o recurso de inteligência Google Now e permitiu o uso de mais de uma conta nos dispositivos.

### Android kit kat (4.4)



Lançado oficialmente no final de 2013, o Android 4.4 trouxe melhorias de desempenho e segurança, além do Google Now, que permitiu realizar mais

tarefas por meio de comandos de voz, como enviar mensagens de texto, reproduzir músicas ou pesquisar rotas.

### **Android lollipop (5.0)**



Android 5.0, Lollipop

O Lollipop foi lançado em 2014 e se tornou compatível com diversas telas: smartphones, tablets, relógios, TVs e carros. A aparência ganhou destaque nesta versão, que passou a contemplar sombras e movimento, além de mostrar notificações nas telas de bloqueio.

### **Android marshmallow (6.0)**



O Marshmallow é uma das mais recentes versões do Android e traz atualizações no NOW, que executa comandos sem precisar interromper o uso. Além disso, possibilita gerenciar os apps que acessam os recursos do seu smartphone, economizando mais bateria.

### **Android nougat (7.0)**



O Android Nougat foi lançado em 2016 e é a penúltima versão do sistema operacional do Google até agora. Ele se destaca por novidades como a abertura de dois aplicativos ao mesmo tempo e as novas notificações. Características como otimização da carga de bateria, definições de aplicativos padrões e economia do plano de dados também fazem parte desta versão.

### **Android oreo (8.0)**



Esta versão é a mais recente do sistema, lançada em 2017 e com recursos e funcionalidades atualizados. O Oreo garante duas vezes mais velocidade ao ligar, função de preenchimento automático e o picture in picture, que permite realizar duas funções ao mesmo tempo.

### **Android 9.0 Pie**



O Google lançou a primeira prévia do desenvolvedor da próxima grande atualização do Android, Android 9.0 P, em 7 de março de 2018. Em 6 de agosto de 2018, a empresa lançou oficialmente a versão final do Android 9.0, com o codinome oficial “Pie”. Incluiu uma série de novos recursos e mudanças importantes. Um deles trocou os botões de navegação tradicionais em favor de um botão alongado no centro, que se tornou o novo botão home. Deslizar para cima abre a Visão geral, com seus aplicativos usados mais recentemente, uma barra de pesquisa e cinco sugestões de aplicativos na parte inferior. Você pode deslizar para a esquerda para ver todos os aplicativos abertos recentemente ou pode arrastar o botão home para a direita para rolar rapidamente pelos aplicativos.

Também incluiu alguns novos recursos projetados para ajudar a estender a vida útil da bateria do seu smartphone. Isso foi conseguido com o uso de aprendizado de máquina no dispositivo, que prevê quais aplicativos você usará agora e quais aplicativos não usará até mais tarde. O Pie também tem Shush,

um recurso que coloca automaticamente o telefone no modo Não perturbe quando você vira a tela do telefone para baixo em uma superfície plana. Há também Slices, que fornece uma versão menor de um aplicativo instalado dentro da Pesquisa Google, oferecendo certas funções de aplicativo sem abrir o aplicativo completo.

Como de costume, o Android 9.0 Pie foi lançado oficialmente primeiro para os telefones Pixel do Google, mas também foi lançado no Essential Phone ao mesmo tempo.

### Android 10

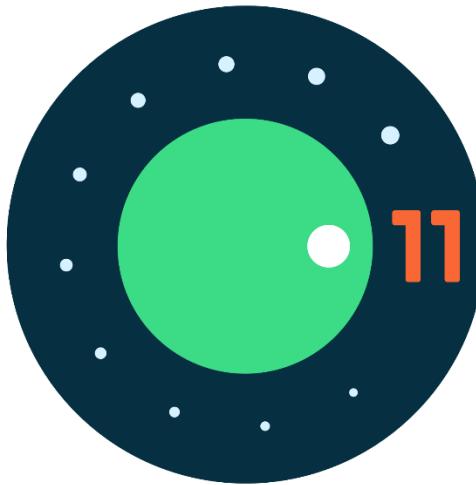


10 anos após o lançamento do sistema operacional, obtivemos outro marco importante na história do Android. O Google lançou a primeira prévia oficial do Android Q para desenvolvedores em 13 de março de 2019. Em 22 de agosto de 2019, o Google anunciou uma grande atualização da marca Android . Isso incluía um novo logotipo e, mais importante, a decisão de abandonar o nome da sobremesa tradicional para a próxima versão. Como resultado, o Android Q é oficialmente conhecido apenas como Android 10. Ele foi lançado oficialmente em 3 de setembro de 2019, para dispositivos Pixel do Google.

Como de costume com qualquer nova versão do Android, o Android 10 teve uma série de novos recursos e melhorias, bem como uma série de novas APIs. Isso incluía suporte para os telefones dobráveis que estavam por vir . O Android 10 também introduziu um modo escuro em todo o sistema, junto com novos controles de navegação por gestos, um menu de compartilhamento mais

eficiente, recursos de resposta inteligente para todos os aplicativos de mensagens e mais controle sobre as permissões baseadas no aplicativo.

### Android 11



Em 18 de fevereiro, o Google lançou o primeiro Developer Preview para Android 11. Depois que vários outros betas públicos foram lançados, a versão final do Android 11 foi lançada em 8 de setembro de 2020.

O Android 11 chegou com muitos recursos novos. Isso inclui uma nova categoria de notificação de conversas, onde todos os seus chats de vários aplicativos são coletados em um só lugar. Você também tem a opção de salvar todas as notificações que apareceram em seu telefone nas últimas 24 horas. Um novo recurso permite gravar a tela do seu telefone, completa com áudio, sem a necessidade de um aplicativo de terceiros. Há também uma nova seção do Android 11 dedicada ao controle de dispositivos domésticos inteligentes.

### Fundamentos de aplicativos

Os aplicativos do Android são programados em linguagem de programação Java. As ferramentas do Android SDK compilam o código — em conjunto com todos os arquivos de dados e recursos — em um APK — um pacote Android, que é um arquivo de sufixo .apk. Os arquivos de APK contêm todo o conteúdo de um aplicativo do Android e são os arquivos que os dispositivos desenvolvidos para Android usam para instalar o aplicativo.

Depois de instalado em um dispositivo, cada aplicativo do Android é ativado na própria sandbox:

O sistema operacional Android é um sistema Linux multiusuário em que cada aplicativo é ou possui um usuário diferente.

Por padrão, o sistema atribui a cada aplicativo um ID de usuário do Linux exclusivo (o ID é usado somente pelo sistema e é desconhecido para o aplicativo). O sistema define permissões para todos os arquivos em um aplicativo, de modo que somente o ID de usuário atribuído àquele aplicativo pode acessá-los.

Cada processo tem a própria máquina virtual (VM), portanto, o código de um aplicativo é executado isoladamente de outros aplicativos.

Por padrão, cada aplicativo é executado no próprio processo do Linux. O Android inicia o processo quando é preciso executar algum componente do aplicativo; em seguida, encerra-o quando não mais é necessário ou quando o sistema precisa recuperar memória para outros aplicativos.

Assim, o sistema Android implementa o princípio do privilégio mínimo. Ou seja, cada aplicativo, por padrão, tem acesso somente aos componentes necessários para a execução do seu trabalho e nada mais. Isso cria um ambiente muito seguro em que o aplicativo não pode acessar partes do sistema para o qual não tem permissão.

No entanto, sempre existe uma maneira de um aplicativo compartilhar dados com outros aplicativos e acessar serviços do sistema:

É possível fazer com que dois aplicativos compartilhem o mesmo ID de usuário do Linux, caso em que eles são capazes de acessar os arquivos um do outro. Para preservar os recursos do sistema, os aplicativos com o mesmo ID de usuário também podem ser combinados para serem executados no mesmo processo Linux e compartilharem a mesma VM (também é preciso atribuir o mesmo certificado aos aplicativos).

Um aplicativo pode solicitar permissão para acessar dados de dispositivo como contatos do usuário, mensagens SMS, o sistema montável (cartão SD),

câmera, Bluetooth etc. O usuário precisa conceder essas permissões de forma explícita.

Essas são as informações básicas de como um aplicativo do Android existe dentro do sistema.

### **Componentes de aplicativo**

Componentes de aplicativo são os blocos de construção de um aplicativo Android. Cada componente é um ponto diferente pelo qual o sistema pode entrar em seu aplicativo. Nem todos os componentes são pontos de entrada reais para o usuário e alguns dependem uns dos outros, mas cada um existe como uma entidade independente e desempenha uma função específica — cada um é um bloco de construção exclusivo que ajuda a definir o comportamento geral do aplicativo.

Há quatro tipos de componentes de aplicativo. Cada tipo tem uma finalidade distinta e tem um ciclo de vida específico que define a forma pela qual o componente é criado e destruído.

Esses são os quatro tipos de componentes de aplicativos:

#### **Activity**

Atividades representam uma tela única com uma interface do usuário. Por exemplo, um aplicativo de e-mail pode ter uma atividade que mostra uma lista de novos e-mails, outra atividade que compõe um e-mail e outra ainda que lê e-mails. Embora essas atividades funcionem juntas para formar uma experiência de usuário coesa no aplicativo de e-mail, elas são independentes entre si. Portanto, um aplicativo diferente pode iniciar qualquer uma dessas atividades (se o aplicativo de e-mails permitir). Por exemplo, um aplicativo de câmera pode iniciar a atividade no aplicativo de e-mail que compõe o novo e-mail para que o usuário compartilhe uma foto.

Uma “atividade” é implementada como uma subclasse de Activity.

#### **Services**

Services são componentes executados em segundo plano para realizar operações de execução longa ou para realizar trabalho para processos remotos.

Eles não apresentam uma interface do usuário. Por exemplo, um serviço pode tocar música em segundo plano enquanto o usuário está em um aplicativo diferente ou buscar dados na rede sem bloquear a interação do usuário com uma activity. Outro componente, como uma activity, pode iniciar o serviço e deixá-lo executar ou vincular-se a ele para interagir. Um serviço é implementado como uma subclasse de Service.

### **Content Provider**

Content Providers gerenciam um conjunto compartilhado de dados do aplicativo. É possível armazenar os dados no sistema de arquivos, em um banco de dados SQLite ou em qualquer local de armazenamento persistente que o aplicativo possa acessar. Por meio do Content Provider, outros aplicativos podem consultar ou até modificar os dados (se o Content Provider permitir). Por exemplo, o sistema Android oferece um Content Provider que gerencia os dados de contato do usuário. Assim, qualquer aplicativo com as permissões adequadas pode consultar parte do Content Provider (como *ContactsContract.Data*) para ler e gravar informações sobre uma pessoa específica.

Os Content Providers são úteis para ler e gravar dados privados no aplicativo e não compartilhados. Por exemplo, o aplicativo Note Pad usa um Content Provider para salvar notas.

Um Content Provider é implementado como uma subclasse de ContentProvider e precisa implementar um conjunto padrão de APIs que permitem a outros aplicativos realizar transações.

### **Broadcast Receiver**

Broadcast Receivers são componentes que respondem a anúncios de transmissão por todo o sistema. Muitas transmissões se originam do sistema — por exemplo, uma transmissão que anuncia que uma tela foi desligada, a bateria está baixa ou uma tela foi capturada. Os aplicativos também podem iniciar transmissões — por exemplo, para comunicar a outros dispositivos que alguns dados foram baixados no dispositivo e estão disponíveis para uso. Embora os Broadcast Receivers não exibam nenhuma interface do usuário, eles podem criar uma notificação na barra de status para alertar ao usuário quando ocorre uma transmissão. Mais comumente, no entanto, um Broadcast Receiver

é somente um "portal" para outros componentes e realiza uma quantidade mínima de trabalho. Ele pode iniciar um service para executar um trabalho baseado no evento.

Os Broadcast Receiver são implementados como subclasses de BroadcastReceiver e cada transmissão é entregue como um objeto Intent.

Um aspecto exclusivo do projeto do sistema Android é que qualquer aplicativo pode iniciar um componente de outro aplicativo. Por exemplo, se você quiser que o usuário capture uma foto com a câmera do dispositivo, provavelmente haverá outro aplicativo que faz isso e seu aplicativo poderá usá-lo, ou seja, não será necessário desenvolver uma atividade para capturar uma foto. Não é necessário incorporar nem mesmo vinculá-lo ao aplicativo da câmera ao código. Em vez disso, basta iniciar a activity no aplicativo da câmera que captura uma foto. Quando concluída, a foto é retornada ao aplicativo em questão para ser usada. Para o usuário, parece que a câmera é realmente parte do aplicativo.

Quando o sistema inicia um componente, ele inicia o processo daquele aplicativo (se ele já não estiver em execução) e instancia as classes necessárias para o componente.

Exemplo: se o aplicativo iniciar a atividade no aplicativo da câmera que captura uma foto, aquele aplicativo é executado no processo que pertence ao aplicativo da câmera e não no processo do aplicativo. Portanto, ao contrário dos aplicativos na maioria dos outros sistemas, os aplicativos do Android não têm nenhum ponto de entrada único (não há a função main(), por exemplo).

Como o sistema executa cada aplicativo em um processo separado com permissões de arquivo que restringem o acesso a outros aplicativos, o aplicativo não pode ativar diretamente um componente de outro aplicativo. No entanto, o sistema Android pode fazer isso. Portanto, para ativar um componente em outro aplicativo, é preciso enviar uma mensagem ao sistema que especifique o intent de iniciar um componente específico. Em seguida, o sistema ativa o componente.

## Ativação de componentes

Três dos quatro tipos de componente — atividades, serviços e receptores de transmissão — são ativados por uma mensagem assíncrona chamada intent. Os intents vinculam componentes individuais entre si em tempo de execução (como mensageiros que solicitam uma ação de outros componentes), seja o componente pertencente ao aplicativo ou não.

O intent é criado com um objeto Intent, que define uma mensagem para ativar um componente específico ou um tipo específico de componente — os intents podem ser explícitos ou implícitos, respectivamente.

Para activities e services, os intents definem a ação a executar (por exemplo, "exibir" ou "enviar" algo) e podem especificar a URI dos dados usados na ação (entre outras coisas que o componente a iniciar precisa saber). Um intent pode transmitir uma solicitação de uma atividade para exibir uma imagem ou abrir uma página da Web. Em alguns casos, é preciso iniciar uma activity para receber um resultado; nesse caso, a atividade também retorna o resultado em um Intent (por exemplo, é possível emitir um intent para que o usuário selecione um contato pessoal e retorne-o a você — o intent de retorno contém uma URI que aponta para o contato selecionado).

Para Broadcast Receivers, o intent simplesmente define o anúncio que está sendo transmitido (por exemplo, uma transmissão para indicar que a bateria do dispositivo está acabando contém uma string de ação conhecida que indica "nível baixo da bateria").

O outro tipo de componente - o Content Provider - não é ativado por intents. Em vez disso, ele é ativado por uma solicitação de um ContentResolver. O “resolvedor” de conteúdo trata todas as transações diretas com o provedor de conteúdo para que o componente que executa as transações com o provedor não precise e, em vez disso, chama os métodos no objeto ContentResolver. Isso deixa uma camada de abstração entre o provedor de conteúdo e o componente que solicita informações (por segurança).

Há dois métodos para ativar cada tipo de componente:

É possível iniciar uma activity (ou dar-lhe algo novo para fazer) passando uma Intent a startActivity() ou startActivityForResult()(para que, quando desejado, a atividade retorne um resultado).

É possível iniciar um dispositivo (ou dar novas instruções a um serviço em andamento) passando uma Intent a startService(). Também é possível vincular ao serviço passando uma Intent a bindService().

Você pode iniciar uma transmissão passando uma Intent para métodos como sendBroadcast(),sendOrderedBroadcast()ou sendStickyBroadcast().

É possível iniciar uma consulta a um provedor de conteúdo chamando query() em um ContentResolver.

### O arquivo Manifest.xml

Antes de o sistema Android iniciar um componente de aplicativo, é preciso ler o arquivo AndroidManifest.xml (o arquivo de “manifesto”) do aplicativo para que o sistema saiba que o componente existe. O aplicativo precisa declarar todos os seus componentes nesse arquivo, que deve estar na raiz do diretório do projeto do aplicativo.

O Manifest faz outras coisas além de declarar os componentes do aplicativo, por exemplo:

Identifica todas as permissões do usuário de que o aplicativo precisa, como acesso à internet ou acesso somente leitura aos contatos do usuário.

Declara o nível de API mínimo exigido pelo aplicativo com base nas APIs que o aplicativo usa.

Declara os recursos de hardware e software usados ou exigidos pelo aplicativo, como câmera, serviços de Bluetooth ou tela multitoque.

As bibliotecas de API às quais o aplicativo precisa se vincular (outras além das APIs de estrutura do Android), como a biblioteca Google Maps.

## Declaração de componentes

A principal tarefa do manifesto é informar ao sistema os componentes do aplicativo. Por exemplo, um arquivo de manifesto pode declarar uma atividade da seguinte forma:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest ... >
    <application android:icon="@drawable/app_icon.png" ...
    >
        <activity
            android:name="com.example.project.ExampleActivity"
            android:label="@string/example_label" ...
        >
            </activity>
            ...
        </application>
    </manifest>
```

No elemento `<application>`, o atributo `android:icon` aponta para recursos de um ícone que identifica o aplicativo.

No elemento `<activity>`, o atributo `android:name` especifica o nome da classe totalmente qualificada da subclasse de Activity e o atributo `android:label` especifica uma string a usar como rótulo da atividade visível ao usuário.

É preciso declarar todos os componentes desta forma:

Elementos `<activity>` para atividades

Elementos `<service>` para serviços

Elementos `<receiver>` para receptores de transmissão

Elementos `<provider>` para provedores de conteúdo

Activities, services e content providers incluídos no código-fonte, mas não declarados no manifesto, não ficam visíveis para o sistema e, consequentemente, podem não ser executados. No entanto, Broadcast Receiver podem ser declarados no Manifest dinamicamente no código (como

objetos BroadcastReceiver) e registrados no sistema chamando-se registerReceiver().

### **Declaração de recursos de componentes**

Conforme abordado em *Ativação de componentes*, é possível usar uma Intent para iniciar activities, services e Broadcast Receiver. Isso pode ser feito nomeando-se explicitamente o componente-alvo (usando o nome da classe do componente) no intent. No entanto, a verdadeira força dos intents reside no conceito de intents implícitos. Os intents implícitos descrevem simplesmente o tipo de ação a executar (e, opcionalmente, os dados em que a ação deve ser executada) e permitem ao sistema encontrar e iniciar um componente no dispositivo que pode executar a ação. Se houver mais de um componente que possa executar a ação descrita pelo intent, o usuário selecionará qual deles usar.

Para o sistema identificar os componentes que podem responder a um intent, compara-se o intent recebido com os filtros de intent fornecidos no arquivo de manifesto de outros aplicativos no dispositivo.

Ao declarar uma activity no Manifest do aplicativo, pode-se incluir filtros de intents que declarem os recursos da atividade para que ela responda a intents de outros aplicativos. Para declarar um filtro de intents nos componentes, adiciona-se um elemento <intent-filter> como filho do elemento de declaração do componente.

Exemplo: se o desenvolvimento for para um aplicativo de e-mail com uma activity para compor um novo e-mail, é possível declarar um filtro de intents para responder a intents "enviar" (para enviar um novo e-mail), da seguinte forma:

```
<manifest ... >
    ...
    <application ... >
        <activity
            android:name="com.example.project.ComposeEmailActivity">
            <intent-filter>
                <action
                    android:name="android.intent.action.SEND" />
                <category
                    android:name="android.intent.category.DEFAULT" />
                <data android:type="*/*" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

```
        </intent-filter>
    </activity>
</application>
</manifest>
```

Em seguida, se outro aplicativo criar um intent com a ação ACTION\_SEND e passá-la para startActivity(), o sistema poderá iniciar a atividade de forma que o usuário possa rascunhar e enviar um e-mail.

### Recursos do aplicativo

Os aplicativos Android são compostos de mais que somente códigos — eles exigem recursos separados do código-fonte, como imagens, arquivos de áudio e tudo o que se relaciona com a apresentação visual do aplicativo. Por exemplo, você deve definir animações, menus, estilos, cores e o layout das interfaces do usuário da atividade com arquivos XML. O uso de recursos de aplicativo facilita a atualização de diversas características do aplicativo sem a necessidade de modificar o código e — fornecendo conjuntos de recursos alternativos — permite otimizar o aplicativo para diversas configurações de dispositivo (como idiomas e tamanhos de tela diferentes).

Para todo recurso incluído no projeto Android, as ferramentas de programação SDK definem um ID inteiro exclusivo que o programador pode usar para referenciar o recurso do código do aplicativo ou de outros recursos definidos no XML. Por exemplo, se o aplicativo contiver um arquivo de imagem de nome logo.png (salvo no diretório res/drawable/), as ferramentas de SDK gerarão um código de recurso chamado R.drawable.logo, que pode ser usado para referenciar a imagem e inseri-la na interface do usuário.

Um dos aspectos mais importantes de fornecer recursos separados do código-fonte é a capacidade de fornecer recursos alternativos para diferentes configurações de dispositivo. Por exemplo, ao definir strings de IU em XML, é possível converter as strings em outros idiomas e salvá-las em arquivos separados. Em seguida, com base em um qualificador de idioma acrescentado ao nome do diretório do recurso (como res/values-fr/ para valores de string em francês) e a configuração de idioma do usuário, o sistema Android aplica as strings de idioma adequadas à IU.

O Android aceita vários qualificadores para recursos alternativos. O qualificador é uma string curta incluída no nome dos diretórios de recurso para definir a configuração do dispositivo em que esses recursos serão usados.

Outro exemplo: deve-se criar diferentes layouts para as activities conforme a orientação e o tamanho da tela do dispositivo. Exemplo: quando a tela do dispositivo está em orientação retrato (portrait - vertical), pode ser desejável um layout com botões na vertical, mas, quando a tela está em orientação paisagem (landscape - horizontal), os botões devem estar alinhados horizontalmente. Para alterar o layout conforme a orientação, pode-se definir dois layouts diferentes e aplicar o qualificador adequado ao nome do diretório de cada layout. Em seguida, o sistema aplica automaticamente o layout adequado conforme a orientação atual do dispositivo.

## A IDE Android Studio

O Android Studio é o ambiente de desenvolvimento integrado (IDE) oficial para o desenvolvimento de aplicativos Android e é baseado no IntelliJ IDEA . Além do editor de código e das ferramentas de desenvolvedor avançados do IntelliJ, o Android Studio oferece ainda mais recursos para aumentar sua produtividade na criação de aplicativos Android, como:

Um sistema de compilação flexível baseado no Gradle

Um emulador rápido com inúmeros recursos

Um ambiente unificado para você poder desenvolver para todos os dispositivos Android

Instant Run para aplicar alterações a aplicativos em execução sem precisar compilar um novo APK

Modelos de códigos e integração com GitHub para ajudar a criar recursos comuns dos aplicativos e importar exemplos de código

Ferramentas e estruturas de teste cheias de possibilidades

Ferramentas de verificação de código suspeito para detectar problemas de desempenho, usabilidade, compatibilidade com versões e outros

## Compatibilidade com C++ e NDK

Compatibilidade embutida com o Google Cloud Platform, facilitando a integração do Google Cloud Messaging e do App Engine.

É possível organizar a janela principal para obter mais espaço na tela ocultando ou movendo barras e janelas de ferramenta. Também é possível usar atalhos de teclado para acessar a maioria dos recursos do IDE.

A qualquer momento, é possível pesquisar o código-fonte, os bancos de dados, as ações, os elementos da interface do usuário e assim por diante pressionando duas vezes a tecla Shift ou clicando na lupa no canto superior direito da janela do Android Studio. Isso pode ser muito útil quando, por exemplo, você quiser localizar uma determinada ação de IDE e esqueceu a forma de acionamento.

## Janelas de ferramentas

Em vez de usar perspectivas predefinidas, o Android Studio segue o contexto e exibe automaticamente as janelas de ferramentas relevantes de acordo com o trabalho. Por padrão, as janelas de ferramentas mais comuns são fixas na barra de janelas de ferramentas nas bordas da janela de aplicativos.

- Para expandir ou recolher uma janela de ferramenta, clique no nome da ferramenta na barra de janelas de ferramentas. Também é possível arrastar, fixar, desafixar, anexar e desanexar janelas de ferramentas.
- Para voltar ao layout padrão atual da janela de ferramentas, clique em **Window > Restore Default Layout** ou personalize o layout padrão clicando em **Window > Store Current Layout as Default**.
- Para mostrar ou ocultar toda a barra de janelas de ferramentas, clique no ícone da janela  no canto inferior esquerdo da janela do Android Studio.
- Para localizar uma janela de ferramenta específica, passe o cursor sobre o ícone da janela e selecione a janela da ferramenta no menu.

Também é possível usar atalhos de teclado para abrir janelas de ferramentas.

A tabela 1 lista os atalhos para as janelas mais comuns.

Janela da ferramenta	Windows e Linux	Mac
Projeto	<b>Alt + 1</b>	<b>Command + 1</b>
Controle de versão	<b>Alt + 9</b>	<b>Command + 9</b>
Executar	<b>Shift + F10</b>	<b>Control + R</b>
Depurar	<b>Shift + F9</b>	<b>Control + D</b>
Android Monitor	<b>Alt + 6</b>	<b>Command + 6</b>
Voltar ao editor	<b>Esc</b>	<b>Esc</b>
Esconder todas as janelas de ferramentas	<b>Control + Shift + F12</b>	<b>Command + Shift + F12</b>

Se você quiser ocultar todas as barras de ferramentas, janelas de ferramentas e guias do editor, clique em **View > Enter Distraction Free Mode**. O *Distraction Free Mode* será ativado. Para sair do *Distraction Free Mode*, clique em **View > Exit Distraction Free Mode**.

Você pode usar *Speed Search* para pesquisar e filtrar dentro da maioria das janelas de ferramentas no Android Studio. Para usar *Speed Search*, selecione a janela de ferramentas e digite a consulta de pesquisa.

### Preenchimento automático de código

O Android Studio tem três tipos de preenchimento automático de código, que podem ser acessados usando atalhos de teclado.

Tipo	Descrição	Windows e Linux	Mac
Preenchimento básico	Exibe sugestões básicas de variáveis, tipos, métodos, expressões e outros. Se você chamar a conclusão básica duas vezes seguidas, verá mais resultados, incluindo membros privados e membros estáticos não importados.	<b>Control + Space</b>	<b>Control + Space</b>
Preenchimento inteligente	Exibe opções relevantes para o contexto em questão. A conclusão inteligente conhece os tipos e os fluxos de dados esperados. Se você chamar a conclusão inteligente duas vezes seguidas, verá mais resultados, incluindo cadeias.	<b>Control + Shift + Space</b>	<b>Control + Shift + Space</b>
Preenchimento de declaração	Preenche a declaração atual para você, adicionando parênteses, colchetes, chaves, formatação e outros elementos que eventualmente estejam faltando.	<b>Control + Shift + Enter</b>	<b>Shift + Command + Enter</b>

**Tabela 2.** Atalhos de teclado para preenchimento automático de código.

Também é possível executar correções rápidas e mostrar as ações de intenção pressionando **Alt+Enter**.

## Encontrar exemplos de código

O Buscador de exemplos de código do Android Studio ajuda a encontrar exemplos de código Android de alta qualidade oferecidos pelo Google de acordo com o símbolo em destaque no momento no seu projeto.

## Navegação

Algumas dicas para ajudá-lo na navegação do Android Studio.

- Alterne entre os arquivos acessados recentemente com a ação *Recent Files*. Pressione **Control+E** (**Command+E** no Mac) para chamar a ação Recent Files. Por padrão, o último arquivo acessado é selecionado. Também é possível acessar qualquer janela de ferramentas por meio da coluna esquerda dessa ação.
- Veja a estrutura do arquivo atual com a ação *File Structure*. Chame a ação File Structure pressionando **Control+F12** (**Command+F12** no Mac). Essa ação permite navegar rapidamente para qualquer parte do arquivo atual.
- Pesquise e navegue para uma classe específica no projeto com a ação *Navigate to Class*. Chame a ação pressionando **Control+N** (**Command+O** no Mac). A ação Navigate to Class é compatível com expressões sofisticadas, incluindo maiúsculas intermediárias (camel humps), caminhos, navegar para linha e correspondência do nome do meio, entre muitas outras. Se você chamar a ação duas vezes seguidas, ela mostrará os resultados obtidos das classes do projeto.
- Navegue para um arquivo ou pasta com a ação *Navigate to File*. Chame a ação Navigate to File pressionando **Control+Shift+N** (**Command+Shift+Ono** Mac). Para pesquisar pastas em vez de arquivos, adicione uma / ao final da expressão.
- Navegue para um método ou campo por nome com a ação *Navigate to Symbol*. Chame a ação Navigate to Symbol pressionando **Control+Shift+Alt+N** (**Command+Shift+Alt+O** no Mac).
- Encontre todos os fragmentos de código que referenciam a classe, o método, o campo, o parâmetro ou a declaração na posição atual do cursor pressionando **Alt+F7**.

## Estilo e formatação

Durante a edição, o Android Studio aplica automaticamente formatação e estilos como especificado nas configurações de estilo do código. Você pode personalizar as configurações de estilo do código de acordo com a linguagem de programação, incluindo a especificação de convenções

para tabulação e identação, espaços, quebras de linha, chaves e linhas em branco. Para personalizar as configurações de estilo do código, clique em **File > Settings > Editor > Code Style (Android Studio > Preferences > Editor > Code Style no Mac.)**

Embora o IDE aplique automaticamente a formatação durante a edição, também é possível chamar explicitamente a ação *Reformat Code* pressionando **Control+Alt+L (Opt+Command+L no Mac)** ou identar automaticamente todas as linhas pressionando **Control+Alt+I (Alt+Option+I no Mac)**.

```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    mActionBar = getSupportActionBar();
    mActionBar.setDisplayHomeAsUpEnabled(true);
```

**Figura 5.** Código antes da formatação.

```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    mActionBar = getSupportActionBar();
    mActionBar.setDisplayHomeAsUpEnabled(true);
    // Get reference to the drawer layout and set event listener
```

**Figura 6.** Código depois da formatação.

## Conceitos básicos do controle de versão

O Android Studio é compatível com diversos sistemas de controle de versões (VCSs), incluindo Git, GitHub, CVS, Mercurial, Subversion e Google Cloud Source Repositories.

Após importar o aplicativo para o Android Studio, use as opções do menu VCS do Android Studio para ativar a compatibilidade de VCS com o sistema de controle de versão desejado, criar um repositório, importar os novos arquivos para o controle de versão e executar outras operações:

1. No menu **VCS** do Android Studio, clique em **Enable Version Control Integration**.
2. No menu suspenso, selecione o sistema de controle de versão a ser associado à raiz do projeto e clique em **OK**.

Agora, o menu VCS exibe diversas opções de controle de versão, de acordo com o sistema selecionado.

**Observação:** Também é possível usar a opção de menu **File > Settings > Version Control** para definir e modificar as configurações do controle de versão

### Sistema de compilação Gradle

O Android Studio usa o Gradle como o sistema de compilação de base, com outros recursos específicos do Android sendo disponibilizados pelo *Android Plugin for Gradle*. Esse sistema de compilação é executado como uma ferramenta integrada no menu do Android Studio e de forma independente na linha de comando. Você pode usar os recursos do sistema de compilação para fazer o seguinte:

Personalize, configure e amplie o processo de compilação.

Crie diversos APKs do seu aplicativo com diferentes recursos usando o mesmo projeto e os mesmos módulos.

Reutilize código e recursos nos conjuntos de origem.

O uso da flexibilidade do Gradle permite que você faça tudo isso sem modificar os arquivos de origem principais do aplicativo. O nome dos arquivos de compilação do Android Studio é **build.gradle**. Esses são arquivos de texto simples que usam Groovy para configurar a compilação com os elementos fornecidos pelo Android Plugin para Gradle. Cada projeto tem um arquivo de compilação de nível superior para todo o projeto e arquivos de compilação de módulo separados para cada módulo. Quando você importa um projeto existente, o Android Studio gera automaticamente os arquivos de compilação necessários.

### Variantes de compilação

O sistema de compilação pode ajudar a criar versões diferentes do mesmo aplicativo de um único projeto. Isso é útil quando existe uma versão gratuita e uma versão paga do aplicativo ou você quer distribuir vários APKs para configurações de dispositivo diferentes no Google Play.

## Divisões de APK

As divisões de APK permitem criar eficientemente vários APKs de acordo com a densidade de tela ou o ABI. Por exemplo, as divisões de APK permitem criar versões hdpi e mdpi separadas de um aplicativo sem que sejam consideradas variações diferentes e possibilitem o compartilhamento de configurações de aplicativo de teste, javac, dx e ProGuard.

## Redução de recursos

A redução de recursos no Android Studio remove automaticamente recursos não utilizados do aplicativo empacotado e das dependências de biblioteca. Por exemplo, se o aplicativo usar serviços do Google Play para acessar funcionalidades do Google Drive e não usar o Google Sign-In, a remoção de recursos poderá remover os diversos ativos drawable dos botõesSignInButton.

**Observação:** a redução de recursos trabalha em conjunto com ferramentas de redução de código, como ProGuard.

## Gerenciamento de dependências

As dependências do projeto são especificadas por nome no arquivo build.gradle. O Gradle se encarrega de encontrar as dependências e disponibilizá-las na compilação. Você pode declarar dependências de módulos, dependências binárias remotas e locais no arquivo build.gradle. O Android Studio configura os projetos para usarem por padrão o repositório central do Maven. (Essa configuração está incluída no arquivo de compilação de nível superior para o projeto.)

## Ferramentas de depuração e criação de perfil

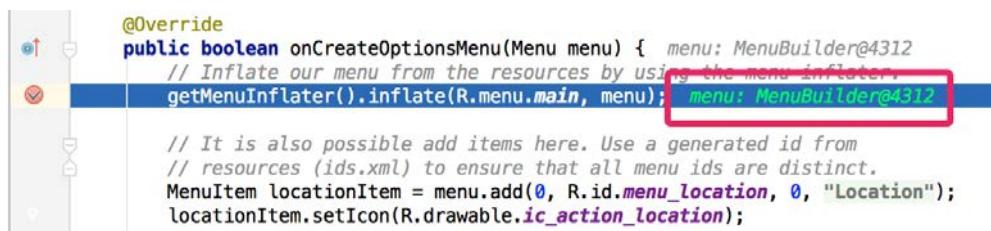
O Android Studio ajuda a depurar e melhorar o desempenho do código, incluindo ferramentas de depuração em linha e análise de desempenho.

### Depuração em linha

Use a depuração em linha para melhorar o acompanhamento do código na visualização do depurador com a verificação em linha de referências, expressões e valores de variáveis. As informações de depuração em linha incluem:

- Valores de variáveis em linha
- Objetos referenciadores que referenciam um determinado objeto

- Valores de retorno dos métodos
- Lambda e expressões do operador
- Valores de dica de contexto



O valor de uma variável em linha.

Para usar a depuração em linha, na janela **Debug**, clique em **Settings** e marque a caixa de seleção **Show Values Inline**.

### Monitores de desempenho

O Android Studio oferece monitores de desempenho para que você possa facilmente acompanhar o uso de memória e CPU do aplicativo, encontrar objetos desalocados, localizar vazamentos de memória, otimizar desempenho de gráficos e analisar solicitações de rede. Em um aplicativo executando em um dispositivo ou emulador, abra a janela de ferramentas **Android Monitor** e clique na guia **Monitors**.

### Captura de pilha

Durante o monitoramento do uso de memória no Android Studio, é possível, ao mesmo tempo, iniciar a coleta de lixo e despejar a pilha Java para um instantâneo de pilha em um arquivo de formato binário HPROF específico do Android. O visualizador de HPROF exibe classes, instâncias de cada classe e uma árvore de referência para ajudar a acompanhar o uso de memória e localizar vazamentos de memória.

### Rastreador de alocação

O Android Studio permite rastrear a alocação de memória durante o monitoramento do uso de memória. O rastreamento de alocação de memória permite monitorar onde os objetos são alocados quando determinadas ações são executadas. O conhecimento dessas alocações possibilita a otimização do desempenho e do uso de memória do aplicativo com ajustes de chamadas de métodos relacionados a essas ações.

### Acesso aos arquivos de dados

As ferramentas do Android SDK, como Systrace, logcat e Traceview, geram dados de desempenho e depuração para análises detalhadas do aplicativo.

Para visualizar os arquivos de dados gerados disponíveis, abra a janela de ferramentas Captures. Na lista de arquivos gerados, clique duas vezes em um arquivo para exibir os dados. Clique com o botão direito do mouse em qualquer arquivo .hprof para convertê-lo para o formato de arquivo .hprof padrão.



## Módulos

*Módulo* é uma coleção de arquivos de origem e configurações de compilação que permitem dividir o projeto em unidades distintas de funcionalidade. O projeto pode ter um ou mais módulos. Um módulo pode usar outro módulo como dependência. Cada módulo pode ser individualmente compilado, testado e depurado.

Muitas vezes, os módulos adicionais são úteis para criar bibliotecas de código no projeto ou quando é necessário criar conjuntos de código e recursos diferentes para tipos de dispositivos distintos, como telefones e wearables, mas mantendo todos os arquivos no escopo do mesmo projeto e compartilhando algum código.

Para adicionar um novo módulo ao projeto, clique em **File > New > New Module**.

O Android Studio oferece alguns tipos diferentes de módulo:

### Módulo de aplicativo Android

Oferece um contêiner para o código-fonte, os arquivos de recursos e configurações do aplicativo, como o arquivo de compilação do módulo e o arquivo de manifesto do Android. Ao criar um novo projeto, o nome padrão do módulo é "app".

Na janela **Create New Module**, o Android Studio oferece os seguintes módulos de aplicativos:

- Módulo de telefones e tablets
- Módulo de Android Wear
- Módulo de Android TV
- Módulo de Glass

Cada um deles fornece arquivos essenciais e alguns modelos de código adequados ao aplicativo ou tipo de dispositivo correspondente.

### Módulo de biblioteca

Fornece um contêiner para o código reutilizável, que pode servir de dependência em outros módulos de aplicativos ou importado para outros projetos. Em termos de estrutura, o módulo de biblioteca é igual ao módulo de aplicativo. No entanto, quando compilado, cria um arquivo de arquivamento de código em vez de um APK, por isso não pode ser instalado em um dispositivo.

Na janela **Create New Module**, o Android Studio oferece os seguintes módulos de biblioteca:

- Biblioteca Android: esse tipo de biblioteca contém todos os tipos de arquivos permitidos em um projeto Android, inclusive código-fonte, recursos e arquivos de manifesto. A compilação gera um arquivo Android Archive (AAR) que pode ser adicionado como dependência dos módulos de aplicativos Android.
- Biblioteca Java: esse tipo de biblioteca contém apenas arquivos de origem Java. O resultado da compilação é um arquivo Java Archive (JAR) que você pode adicionar como dependência de módulos de aplicativos Android ou outros projetos Java.

### Módulo de Google Cloud

Oferece um contêiner para o código de back-end do Google Cloud. O módulo adiciona o código e dependências necessários para um back-end do Java App Engine que usa HTTP simples, Cloud Endpoints e Cloud Messaging para conexão com o aplicativo. Você pode desenvolver o seu back-end para fornecer os serviços de nuvem necessários para o aplicativo.

O uso do Android Studio para criar e desenvolver o módulo de Google Cloud permite gerenciar o código do aplicativo e do back-end no mesmo projeto. Também é possível executar e testar o código de back-end localmente e usar o Android Studio para implantar o módulo de Google Cloud.

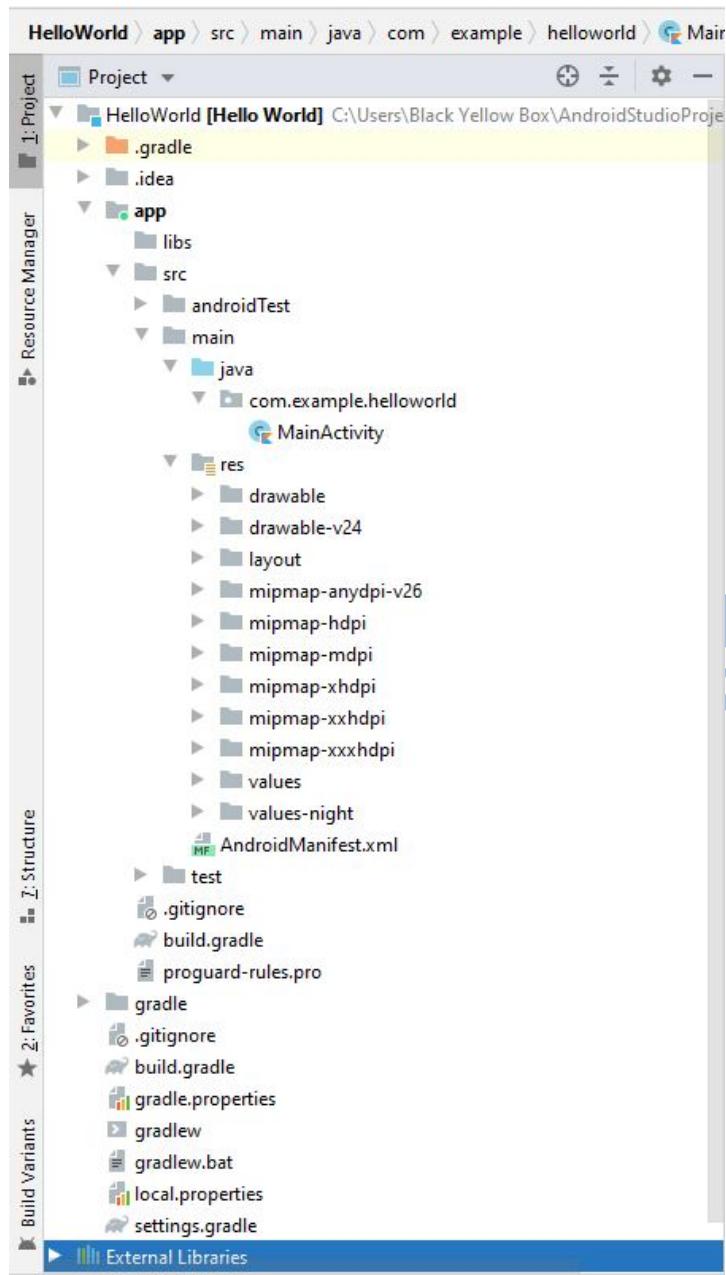
Para obter mais informações sobre a adição de um módulo de Google Cloud, consulte Adicionar um módulo de Servlet Java do App Engine. Para obter

mais informações sobre a execução e implantação de um módulo de Google Cloud, consulte Executar, testar e implantar o back-end.

Algumas pessoas também usam o termo subprojeto em vez de módulo. Isso não representa um problema, pois o Gradle também faz referência a módulos como projetos. Por exemplo, quando você cria um módulo de biblioteca e quer adicioná-lo como dependência a um módulo de aplicativo Android, é necessário declará-lo da seguinte forma:

```
dependencies {  
    implementation project(':my-library-module')  
}
```

## Arquivos de projetos



Por padrão, o Android Studio exibe os arquivos de projetos na visualização Android. Essa visualização não reflete a hierarquia de arquivos real no disco, mas é organizada por módulos e tipos de arquivo para simplificar a navegação entre os principais arquivos de origem do projeto, ocultando alguns arquivos ou diretórios pouco usados. Veja a seguir algumas mudanças estruturais em relação à estrutura no disco:

- Mostra todos os arquivos de configuração da compilação do projeto em um grupo Gradle Script no nível superior.

- Mostra todos os arquivos de manifesto de cada módulo em um grupo no nível de módulos (quando existem arquivos de manifesto diferentes para variações e tipos de compilação de produto diferentes).
- Mostra todos os arquivos de recursos alternativos em um único grupo, em vez de pastas separadas por qualificador de recurso. Por exemplo, todas as versões de densidade do ícone inicializador podem ser vistas lado a lado.

Em cada módulo de aplicativo Android, os arquivos são mostrados nos seguintes grupos:

#### **manifests**

Contém o arquivo `AndroidManifest.xml`.

#### **java**

Contém os arquivos de código-fonte do Java, separados por nome de pacote, inclusive o código de teste do JUnit.

#### **res**

Contém todos os recursos que não são código, como layouts XML, strings de IU e imagens em bitmap, divididos em subdiretórios correspondentes.

Para ver a estrutura de arquivos real do projeto, incluindo todos os arquivos ocultos na visualização do Android, selecione Project no menu suspenso na parte superior da janela Project.

Quando você seleciona a visualização Project, pode ver um número muito maior de arquivos e diretórios. Os mais importantes são:

#### ***module-name/***

#### ***build/***

Contém saídas de compilação.

#### ***libs/***

Contém bibliotecas privadas.

`src/`

Contém todos os arquivos de código e recursos do módulo nos seguintes subdiretórios:

`androidTest/`

Contém o código dos testes de instrumentação executados em um dispositivo Android.

`main/`

Contém os arquivos do conjunto de origem "main": o código Android e os recursos compartilhados por todas as variações de compilação (os arquivos para outras variações de compilação residem em diretórios irmãos, como `src/debug/` para o tipo de compilação de depuração).

`AndroidManifest.xml`

Descreve a natureza do aplicativo e de cada um de seus componentes. Para obter mais informações, consulte a documentação do `AndroidManifest.xml`.

`java/`

Contém os códigos-fonte Java.

`gen/`

Contém os arquivos Java gerados pelo Android Studio, como o arquivo `R.java` e as interfaces criadas de arquivos AIDL.

`res/`

Contém recursos de aplicativos, como arquivos `drawable`, arquivos de `layout` e `strings` de IU. Consulte Recursos de aplicativos para obter mais informações.

**assets/**

Contém o arquivo que deve ser compilado para um arquivo .apk no estado em que está. Você pode navegar nesse diretório da mesma forma que em um sistema de arquivos normal usando URIs e ler arquivos como stream de bytes usando o AssetManager . Por exemplo, esse é um bom local para texturas e dados de jogos.

**test/**

Contém código para testes locais executados na JVM host.

**build.gradle (módulo)**

Define as configurações de compilação do módulo.

**build.gradle (projeto)**

Define a configuração de compilação que se aplica a todos os módulos. Esse arquivo faz parte do projeto, portanto deve ser mantido no controle de revisões em conjunto com todo o código-fonte restante.

### **Configurações de estrutura de projetos**

Para alterar várias configurações do projeto do Android Studio, clique em File > Project Structure para abrir a caixa de diálogo Project Structure. Ela contém as seguintes seções:

SDK Location: define a localização do JDK, do Android SDK e do Android NDK usados pelo projeto.

Project: define a versão do Gradle e do Android Plugin para Gradle, bem como o nome da localização do repositório.

Developer Services: contém configurações de componentes complementares do Google ou de terceiros para o Android Studio.

Modules: permite editar configurações de compilação de módulos, incluindo o SDK pretendido e o SDK mínimo, a assinatura do aplicativo e as dependências de biblioteca.

## Developer Services

A seção Developer Services da caixa de diálogo Project Structure contém páginas de configuração para diversos serviços que podem ser usados para o aplicativo. Essa seção contém as seguintes páginas:

AdMob: permite ativar o componente AdMob do Google, que ajuda a entender os usuários e a exibir a eles publicidade personalizada.

Analytics: permite ativar o Google Analytics, que ajuda a medir interações do usuário com o aplicativo em diversos dispositivos e ambientes.

Authentication: permite que os usuários utilizem o Google Sign-In para fazer login no aplicativo com suas contas Google.

Cloud: permite ativar os serviços Firebase baseados em nuvem para o aplicativo.

Notifications: permite usar o Google Cloud Messaging para comunicação entre o aplicativo e o servidor.

A ativação de qualquer um desses serviços pode fazer com que o Android Studio adicione as dependências e permissões necessárias ao aplicativo. Cada página de configuração lista essas e outras ações tomadas pelo Android Studio na ativação do serviço associado.

## Módulos

A seção de configurações Modules permite alterar opções de configuração de cada um dos módulos do projeto. A página de configurações de cada módulo é dividida nas seguintes guias:

Properties: especifica as versões do SDK e das ferramentas de compilação usadas para compilar o módulo.

Siging: especifica o certificado usado para assinar o APK.

Flavors: permite criar várias variações, onde cada variação especifica um conjunto de definições de configuração, como a versão mínima e pretendida de SDK do módulo e o código e nome da versão. Por exemplo, é possível definir

uma variação que tem um SDK com versão mínima 15 e versão pretendida 21 e outra variação que tem um SDK com versão mínima 19 e versão pretendida 23.

**Build Types:** permite criar e modificar configurações de compilação, como descrito em Configuração de compilações do Gradle. Por padrão, cada módulo tem os tipos de compilação debug e release, mas tipos adicionais podem ser definidos, se necessário.

**Dependencies:** lista as dependências de biblioteca, arquivo e módulo deste módulo. É possível adicionar, modificar e excluir dependências nesse painel. Para obter mais informações sobre dependências de módulos, consulte Configuração de compilações do Gradle.

### Atualizar o IDE e o SDK Tools

Após instalar o Android Studio, é fácil manter o IDE do Android Studio e as ferramentas do Android SDK atualizados com atualizações automáticas e o Android SDK Manager.

### Atualizar o IDE e alterar os canais

O Android Studio usa uma pequena caixa de diálogo em balão para notificar a disponibilidade de atualizações para o IDE. No entanto, você pode verificar manualmente a existência de atualizações clicando em **Help > Check for Update** (no Mac, **Android Studio > Check for Updates**).

As atualizações do Android Studio estão disponíveis nos canais de versões a seguir:

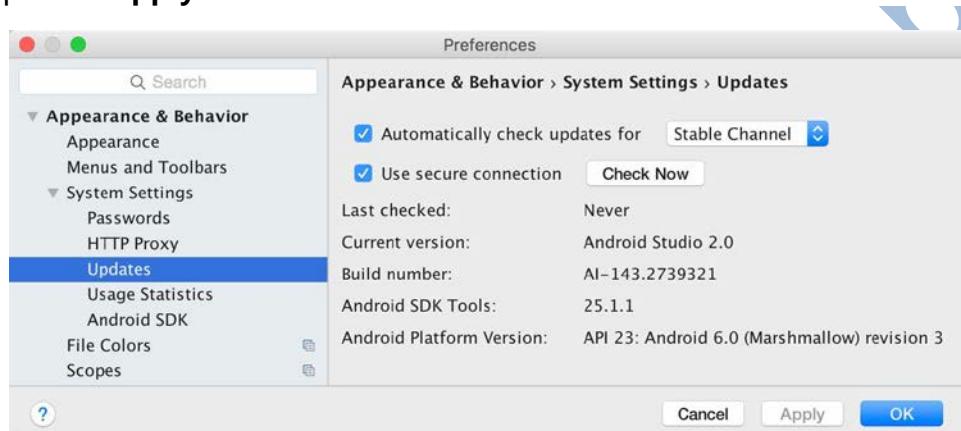
- **Canal Canary:** versões de vanguarda absoluta, atualizadas aproximadamente uma vez por semana. Embora essas compilações estejam sujeitas a mais erros, elas são testadas e queremos oferecer acesso antecipado para que você possa experimentar novos recursos e enviar seus comentários. Esse canal **não é recomendado para desenvolvimento de produção**.
- **Canal do desenvolvedor:** compilações canary, escolhidas individualmente e aprovadas em uma rodada completa de testes internos.
- **Canal beta:** compilações candidatas a lançamento baseadas em compilações canary, lançadas para obter comentários antes de serem disponibilizadas no canal estável.
- **Canal estável:** a versão estável oficial disponível para download em [developer.android.com/studio](http://developer.android.com/studio).

Por padrão, o Android Studio oferece atualizações do canal estável. Se você quiser tentar uma das outras versões do Android Studio, conhecidas

coletivamente como canais de visualização, poderá optar por receber atualizações de um desses canais.

Para alterar o canal de atualização, faça o seguinte:

1. Abra a janela **Preferences** clicando em **File > Settings** (no Mac, **Android Studio > Preferences**).
2. No painel à esquerda, clique em **Appearance & Behavior > System Settings > Updates**.
3. Verifique se a opção **Automatically check for updates** está marcada e selecione um canal na lista suspensa (consulte a figura 1).
4. Clique em **Apply** ou **OK**.



As preferências de atualização do Android Studio.

Se você quiser experimentar um dos canais de visualização (canary, desenvolvimento ou beta) sem deixar de usar a compilação estável para os projetos de produção do Android, poderá instalar com segurança uma segunda versão do Android Studio baixando a versão de visualização de [tools.android.com](http://tools.android.com).

### Atualizar as ferramentas com o SDK Manager

O Android SDK Manager oferece ferramentas, plataformas e outros componentes do SDK necessários para desenvolver aplicativos.

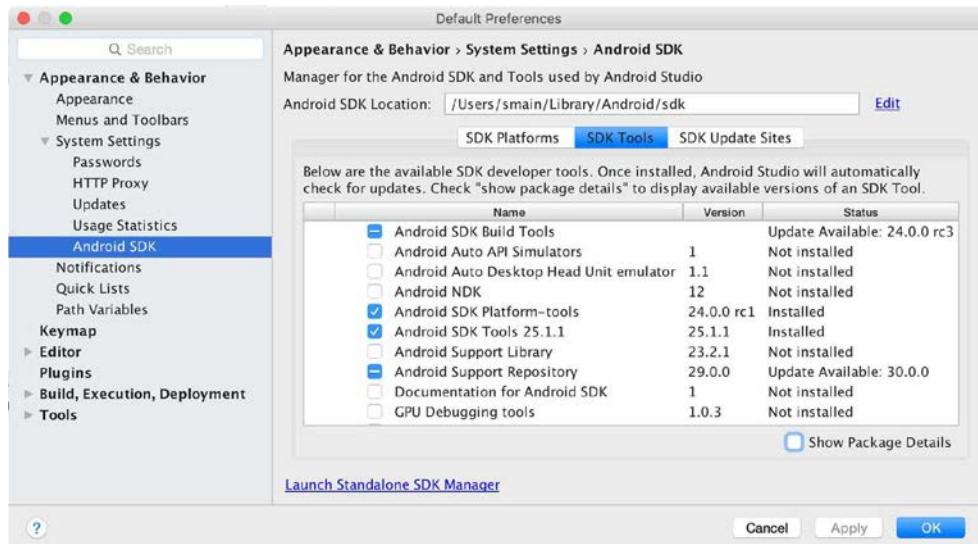
Para abrir o SDK Manager, clique em **Tools > Android > SDK Manager** ou clique em **SDK Manager**  na barra de ferramentas.

Quando uma atualização estiver disponível para um dos seus pacotes, será exibido um traço  na caixa de seleção ao lado do pacote.

- Para atualizar um item ou instalar um item novo, clique na caixa de seleção para que ela exiba uma marca de verificação.
- Para desinstalar um pacote, clique na caixa de seleção para desmarcá-la.

As atualizações pendentes são indicadas na coluna esquerda com um ícone de download . As remoções pendentes são indicadas com uma cruz vermelha .

Para atualizar os pacotes selecionados, clique em **Apply** ou **OK** e concorde com os contratos de licença.



O Android SDK Manager.

## Pacotes recomendados

Você deve considerar cuidadosamente as ferramentas a seguir na guia SDK Tools:

### Android SDK Build Tools

Obrigatório. Inclui ferramentas para compilar aplicativos Android.

### Android SDK Platform-tools

Obrigatório. Inclui diversas ferramentas necessárias para a plataforma Android, incluindo a ferramenta adb.

### Android SDK Tools

Obrigatório. Inclui ferramentas essenciais como Android Emulator e ProGuard.

### Android Support Repository

Recomendado. Inclui o repositório Maven local para as bibliotecas de suporte, que oferecem um conjunto ampliado de APIs compatível com a maioria

das versões do Android. O repositório é necessário para produtos como Android Wear, Android TV e Google Cast. Para obter mais informações, leia Biblioteca de suporte.

### **Google Repository**

Recomendado. Inclui o repositório Maven local para as bibliotecas do Google, que oferecem diversos recursos e serviços para aplicativos, incluindo Firebase, Google Maps e conquistas e placares de líderes de jogos, entre muitos outros.

Na guia SDK Platforms, também é necessário instalar pelo menos uma versão da plataforma Android. Cada versão oferece diversos pacotes diferentes. Para baixar apenas os pacotes necessários, clique na caixa de seleção ao lado do nome da versão.

Para ver todos os pacotes disponíveis para cada plataforma Android, clique em Show Package Details na parte inferior da janela. Em cada versão de plataforma, você encontrará os pacotes a seguir:

### **Android SDK Platform**

Obrigatório. Pelo menos uma plataforma é necessária no seu ambiente para que você possa compilar o aplicativo. Para fornecer a melhor experiência do usuário nos dispositivos mais recentes, use a última versão de plataforma como alvo de compilação. Ainda será possível executar o aplicativo em versões anteriores. No entanto, para execução na versão mais recente do Android, será necessário compilá-lo com a versão mais recente da plataforma para usar os novos recursos.

### **Intel ou ARM System Images**

Recomendado. A imagem de sistema é necessária para executar o Android Emulator. Cada versão de plataforma contém as imagens de sistema compatíveis. Você também pode baixar imagens de sistema posteriormente durante a criação de Android Virtual Devices (AVDs) no AVD Manager. Selecione Intel ou ARM de acordo com o processador do computador de desenvolvimento.

Observação: Se você pretende usar APIs do Google Play Services, deve usar a Google APIs System Image.

A lista acima não é abrangente e você pode adicionar outros sites para baixar pacotes adicionais de terceiros.

Em alguns casos, um pacote SDK pode exigir uma revisão mínima específica de outra ferramenta. Nesse caso, o SDK Manager exibe uma notificação com um aviso e adiciona as dependências à sua lista de downloads.

Dica: você também pode personalizar o arquivo build.gradle para que cada projeto use uma cadeia de compilação e opções de compilação específicas. Para obter mais informações, consulte Configuração de compilações do Gradle.

### **Editar ou adicionar sites de ferramentas do SDK**

Para gerenciar em que sites de SDK o Android Studio verifica a existência de atualizações de ferramentas do Android e de terceiros, clique na guia SDK Update Sites. Você pode adicionar outros sites que hospedam ferramentas próprias e baixar pacotes desses sites.

Por exemplo, uma operadora de celular ou fabricante de dispositivos pode oferecer bibliotecas de API adicionais que são compatíveis com dispositivos próprios que executam Android. Para desenvolver usando essas bibliotecas, você pode instalar o pacote do Android SDK desses terceiros adicionando o URL de suas ferramentas de SDK ao SDK Manager na guia SDK Update Sites.

Se uma operadora ou fabricante de dispositivos hospedar um arquivo repositório de complementos do SDK em seu próprio site, siga estas etapas para adicionar esse site ao Android SDK Manager:

Clique na guia SDK Update Sites.

Clique em Add  na parte inferior da janela.

Insira o nome e o URL do site do terceiro e clique em OK.

Verifique se a caixa de seleção está marcada na coluna Enabled.

Clique em Apply ou OK.

Os pacotes de SDK disponíveis no site aparecerão na guia SDK Platforms ou SDK Tools, conforme o caso.

### **Gradle para Android**

O sistema de compilação do Android compila os recursos e o código-fonte do aplicativo e os empacote em APKs que você pode testar, implantar, assinar e distribuir. O Android Studio usa o Gradle, um kit de ferramentas de compilação avançado, para automatizar e gerenciar o processo de compilação, permitindo que você defina configurações de compilação personalizadas e flexíveis. Cada configuração de compilação pode definir seu próprio conjunto de códigos e recursos, reutilizando as partes comuns a todas as versões do aplicativo. O Android Plugin for Gradle funciona com o kit de ferramentas de compilação para fornecer processos e configurações ajustáveis que são específicas para compilar e testar aplicativos Android.

O Gradle e o plug-in do Android são executados de forma independente do Studio. Isso significa que você pode compilar seus aplicativos Android pelo Android Studio, pela linha de comando do seu computador ou em computadores nos quais o Android Studio não esteja instalado (como servidores de integração contínua). Se você não estiver usando o Android Studio, veja como compilar e executar seu aplicativo pela linha de comando. A saída da compilação é a mesma, esteja você compilando o projeto em uma linha de comando, em um computador remoto ou usando o Android Studio.

**Observação:** como o Gradle e o plug-in do Android são executado de forma independente do Android Studio, você precisa atualizar as ferramentas de compilação separadamente. Leia as notas da versão para saber como atualizar o Gradle e o plug-in do Android.

A flexibilidade do sistema de compilação do Android permite realizar configurações de compilação personalizadas sem modificar os arquivos dos recursos principais do aplicativo. Esta seção ajuda a entender como o sistema de compilação do Android funciona e como ele pode ajudar a personalizar e automatizar várias configurações de compilação. Se quiser saber mais saber como implantar seu aplicativo, consulte Como programar e executar no Android Studio. Para começar a criar configurações de compilação personalizadas imediatamente usando o Android Studio, consulte Como configurar variantes de compilação.

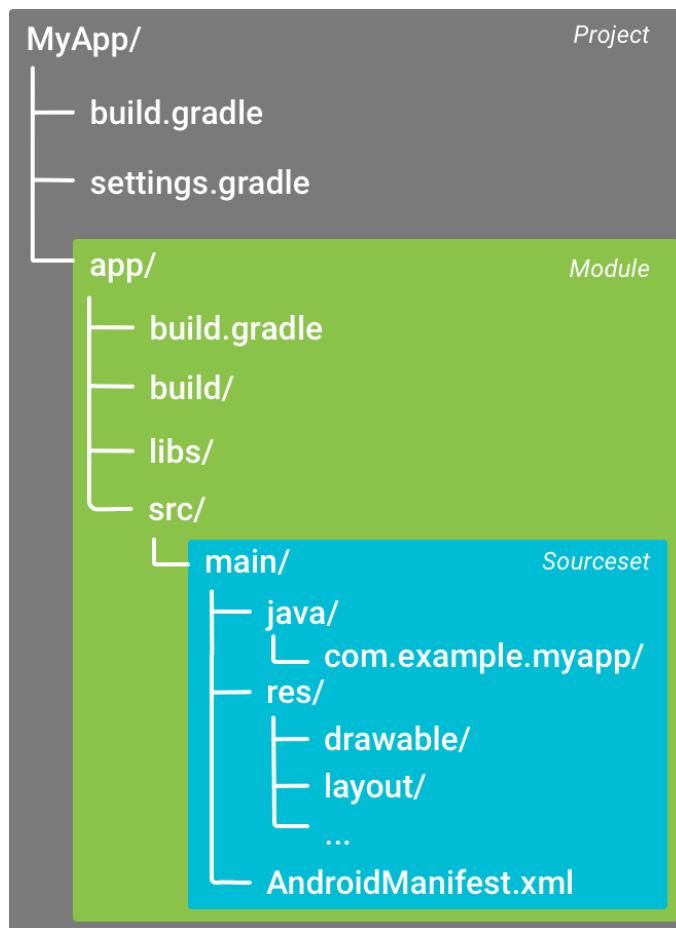
## Divisões de APK

O sistema de compilação permite compilar automaticamente diferentes APKs, cada um contendo apenas os códigos e recursos necessários para uma densidade de tela ou interface binária de aplicativo (ABI) específica.

### Arquivos de configuração de compilação

Para criar configurações de compilação personalizadas, é necessário fazer alterações em um ou mais arquivos de configuração de compilação ou arquivos build.gradle. Esses arquivos de texto sem formatação usam a Domain Specific Language (DSL) para descrever e manipular a lógica de compilação que usa Groovy, que é uma linguagem dinâmica para a máquina virtual Java (JVM). Não é preciso conhecer Groovy para começar a configurar sua compilação, pois o Android Plugin for Gradle introduz a maioria dos elementos DSL de que você precisará..

Ao iniciar um novo projeto, o Android Studio automaticamente cria alguns desses arquivos para você, conforme é mostrado na figura abaixo, e os preenche com base em padrões razoáveis.



A estrutura de projeto padrão para um módulo de aplicativo Android.

Existem alguns arquivos de configuração de compilação do Gradle que fazem parte da estrutura de projeto padrão de um aplicativo Android. Antes de começar a configurar sua compilação, é importante entender o escopo e o propósito de cada um desses arquivos e os elementos DSL que eles devem definir.

### O arquivo de configurações do Gradle

O arquivo `settings.gradle`, localizado no diretório-raiz do projeto, informa ao Gradle quais módulos ele deve incluir ao compilar seu aplicativo. Para a maioria dos projetos, o arquivo é simples e inclui apenas o elemento a seguir:

```
include ':app'
```

Entretanto, projetos com vários módulos devem especificar cada módulo que deve ser incluído na compilação final.

## O arquivo de compilação de nível mais alto

O arquivo build.gradle de nível mais alto, localizado no diretório-raiz do projeto, define configurações de compilação que se aplicam a todos os módulos do seu projeto. Por padrão, o arquivo de compilação de nível mais alto usa o bloco buildscript {} para definir os repositórios e as dependências do Gradle que são comuns a todos os módulos do projeto. O exemplo de código a seguir descreve as configurações padrão e os elementos DSL que podem ser encontrados no arquivo build.gradle de nível mais alto após a criação de um novo projeto.

```
// Top-Level build file where you can add configuration options common to all
// sub-projects/modules.
buildscript {
    ext.kotlin_version = "1.4.21"
    repositories {
        google()
        jcenter()
    }
    dependencies {
        classpath "com.android.tools.build:gradle:4.1.1"
        classpath "org.jetbrains.kotlin:kotlin-gradle-plugin:$kotlin_version"

        // 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
}
```

## O arquivo de compilação de módulo

O arquivo build.gradle de módulo, localizado em cada diretório <project>/<module>/, permite definir configurações de compilação para o módulo específico no qual ele se encontra. Definir essas configurações de compilação permite fornecer opções de empacotamento personalizadas, como tipos de compilação e variações de produtos adicionais, além de modificar as configurações no manifesto de aplicativo main/ ou do arquivo build.gradle de nível mais alto.

Este arquivo `build.gradle` de módulo do exemplo de aplicativo Android descreve alguns dos elementos DSL básicos e das configurações que você deve conhecer.

```
// Top-level build file where you can add configuration
options common to all sub-projects/modules.
buildscript {
    ext.kotlin_version = "1.4.21"
    repositories {
        google()
        jcenter()
    }
    dependencies {
        classpath "com.android.tools.build:gradle:4.1.1"
        classpath "org.jetbrains.kotlin:kotlin-gradle-
plugin:$kotlin_version"

        // 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
}
```

## **Arquivos de propriedade do Gradle**

O Gradle também contém dois arquivos de propriedades, localizados no diretório-raiz do seu projeto, úteis para especificar configurações para o próprio kit de ferramentas de compilação do Gradle:

`gradle.properties`

Nesse arquivo, é possível definir configurações do Gradle para todo o projeto, como o tamanho máximo de heap do daemon do Gradle. Para saber mais, consulte O ambiente de compilação.

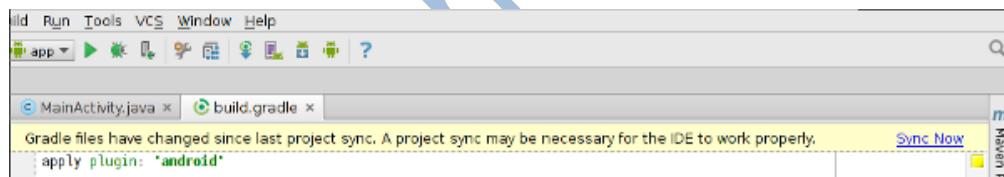
`local.properties`

Esse arquivo configura as propriedades de ambiente local do sistema de compilação, como o caminho da instalação do SDK. Como o conteúdo desse arquivo é gerado automaticamente pelo Android Studio e é específico ao ambiente de desenvolvedor local, você não deve modificá-lo manualmente ou inseri-lo em seu sistema de controle de versões.

### **Como sincronizar o projeto com arquivos do Gradle**

Quando você faz alterações nos arquivos de configuração de compilação do seu projeto, o Android Studio exige a sincronização dos arquivos do projeto para permitir a importação das alterações e a execução de algumas verificações para garantir que as configurações não criem erros de compilação.

Para sincronizar os arquivos do projeto, clique em **Sync Now** na barra de notificações exibida ao fazer alterações, conforme é mostrado na figura 3, ou clique em **Sync Project** na barra de menu. Se o Android Studio perceber erros em sua configuração, por exemplo, se o código-fonte usar recursos de API que só estão disponíveis em um nível de API superior à sua `compileSdkVersion`, a janela **Messages** será exibida com uma descrição do problema.



**Figura 3.** Sincronizando o projeto com arquivos de configuração de compilação no Android Studio.

### **Conjuntos de origem**

O Android Studio agrupa o código-fonte e os recursos de maneira lógica para cada módulo, criando conjuntos de origem. O conjunto de origem main/de um módulo contém os códigos e os recursos usados por todas as suas variantes de compilação. Diretórios de conjuntos de origem adicionais são opcionais, e o Android Studio não os cria automaticamente quando você configura novas variantes de compilação. Entretanto, a criação de conjuntos de origem, semelhante ao main/, ajuda a organizar os arquivos e recursos que o Gradle só deve usar ao compilar certas versões do seu aplicativo:

`src/main/`

Esse conjunto de origem inclui os códigos e recursos comuns a todas as variantes de compilação.

`src/<buildType>/`

Crie esse conjunto de origem para incluir os códigos e recursos exclusivos de um tipo de compilação específico.

`src/<productFlavor>/`

Crie esse conjunto de origem para incluir os códigos e recursos exclusivos de uma variação de produto específica.

`src/<productFlavorBuildType>/`

Crie esse conjunto de origem para incluir os códigos e recursos exclusivos de uma variante de compilação específica.

Por exemplo, para gerar a versão "fullDebug" do seu aplicativo, o sistema de compilação mescla códigos, configurações e recursos dos seguintes conjuntos de origem:

- `src/fullDebug/` (o conjunto de origem da variante de compilação)
- `src/debug/` (o conjunto de origem do tipo de compilação)
- `src/full/` (o conjunto de origem da variação de produto)
- `src/main/` (o conjunto de origem principal)

Observação: ao criar um novo arquivo ou diretório no Android Studio usando as opções de menu File > New, você pode criá-lo para um conjunto de origem específico. Os conjuntos de origem a escolher se baseiam nas configurações de compilação, e o Android Studio cria os diretórios necessários automaticamente se eles ainda não existirem.

Se conjuntos de origem diferentes contiverem versões diferentes do mesmo arquivo, o Gradle usará a seguinte ordem de prioridade ao decidir qual arquivo usar (os conjuntos de origem à esquerda modificam os arquivos e as configurações dos conjuntos de origem à direita):

variante de compilação > tipo de compilação > variação de produto > conjunto de origem principal > dependências de biblioteca

Isso permite que o Gradle use arquivos específicos à variante de compilação que você está tentando compilar reutilizando atividades, lógica de aplicativo e recursos comuns a outras versões do aplicativo. Ao mesclar diversos manifestos, o Gradle usa a mesma ordem de prioridade, portanto cada variante de compilação pode definir diferentes componentes ou permissões no manifesto final.

## Criar projeto

### Visão geral do Android Studio a partir do aplicativo-padrão

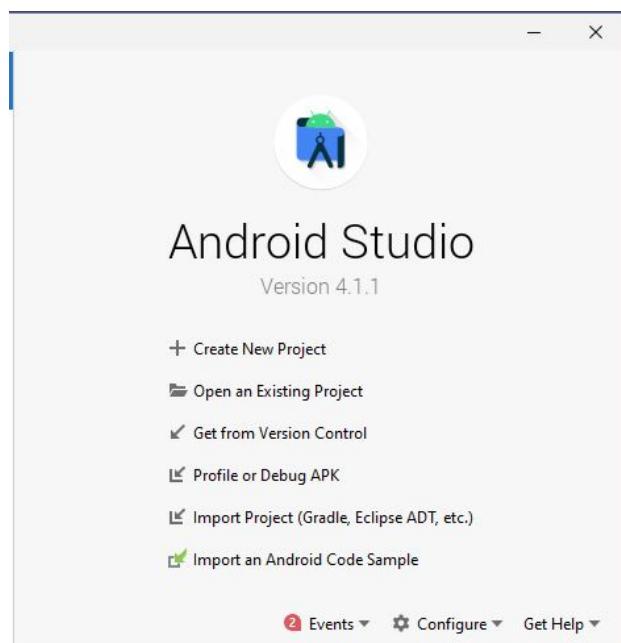
O aplicativo HelloWorld exibe a string "Hello World" na tela de um dispositivo virtual ou dispositivo físico Android. Aqui está o que o aplicativo parece:



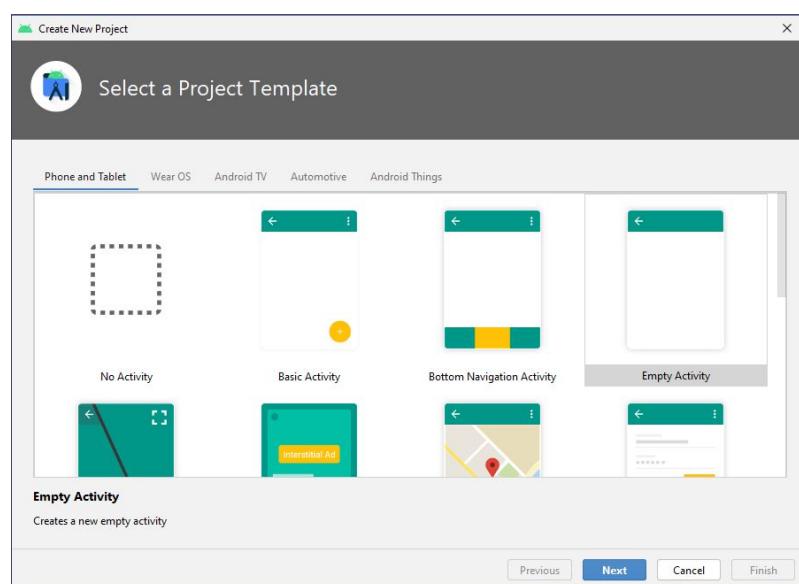
## Criando o projeto HelloWorld

Nesta tarefa, você cria um novo projeto de aplicativo para verificar se o Android Studio está instalado corretamente.

1. Abra o Android Studio, se ainda não estiver aberto.
2. Na caixa de diálogo principal **Welcome to Android Studio**, clique em **Start a new Anndroid Studio Project**.



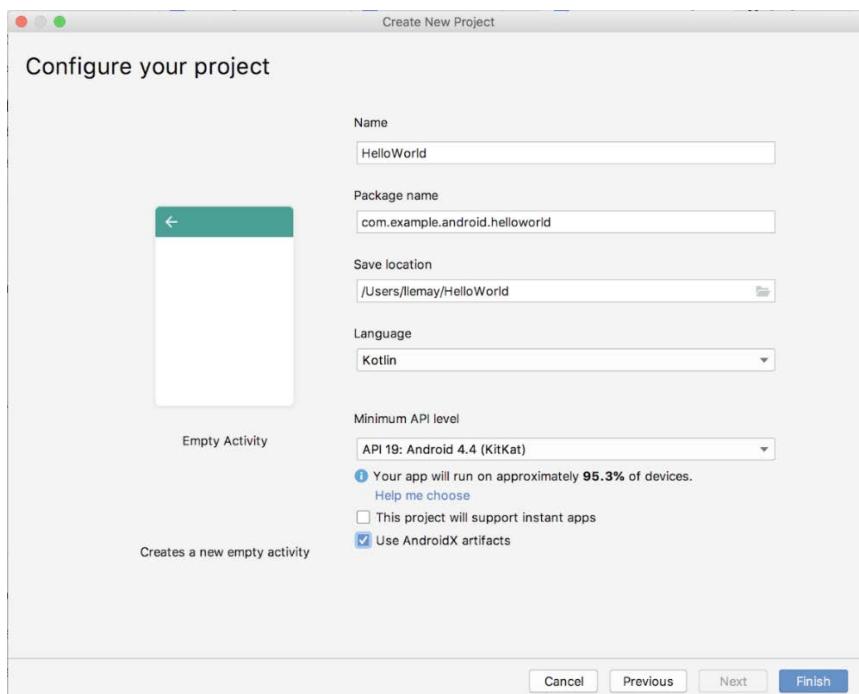
A caixa de diálogo **Choose your project** é exibida. Selecione **Activity Vazia** conforme mostrado abaixo e clique em **Avançar**.



Uma Activity é uma estrutura única e focada que o usuário pode fazer. Cada

aplicativo deve ter pelo menos uma activity como ponto de entrada. Pense nessa activity de ponto de entrada como a função *main()* em outros programas. Uma activity geralmente possui um layout associado a ela que define como os elementos da interface com o usuário (UI) aparecem em uma tela. O Android Studio fornece vários Activity templates para ajudar você a começar.

Na caixa de diálogo **Configure your project**, insira "HelloWorld" para o **nome**:



Aceite o **android.example.com** padrão para o **domínio da Empresa** ou crie um domínio exclusivo da empresa. Esse valor mais o nome do aplicativo é o nome do pacote do seu aplicativo. Se você não estiver planejando publicar seu aplicativo, aceite o padrão. Você pode alterar o nome do pacote do seu aplicativo mais tarde, mas é um trabalho extra.

Verifique se o **local** padrão de **salvamento** é onde você deseja armazenar seu aplicativo. Se não, mude a localização para o seu diretório preferido.

Certifique-se de que o **linguagem** seja Kotlin.

Certifique-se de que o nível mínimo da API seja **API 19: Android 4.4 (KitKat)**. Com esse nível de API, o aplicativo seria executado em aproximadamente 95,3% dos dispositivos.

Marque a caixa de seleção **Use AndroidX artifacts – caso não esteja pré-selecionada**.

Deixe todas as outras caixas de seleção desmarcadas e clique em **Finish**. Se seu projeto exigir mais componentes para o SDK de destino escolhido, o Android Studio os instalará automaticamente, o que pode demorar um pouco. Siga os prompts e aceite as opções padrão.

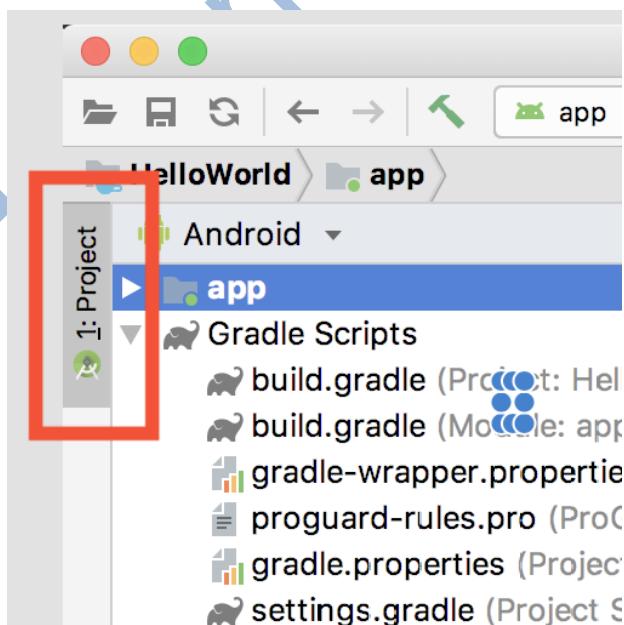
O Android Studio agora cria seu projeto, o que pode levar algum tempo. Você não deve receber nenhum erro. Se você receber algum aviso, ignore-o.

### Características do Android Studio

Nesta tarefa, você explora o projeto HelloWorld no Android Studio e aprende os fundamentos do desenvolvimento com o Android Studio.

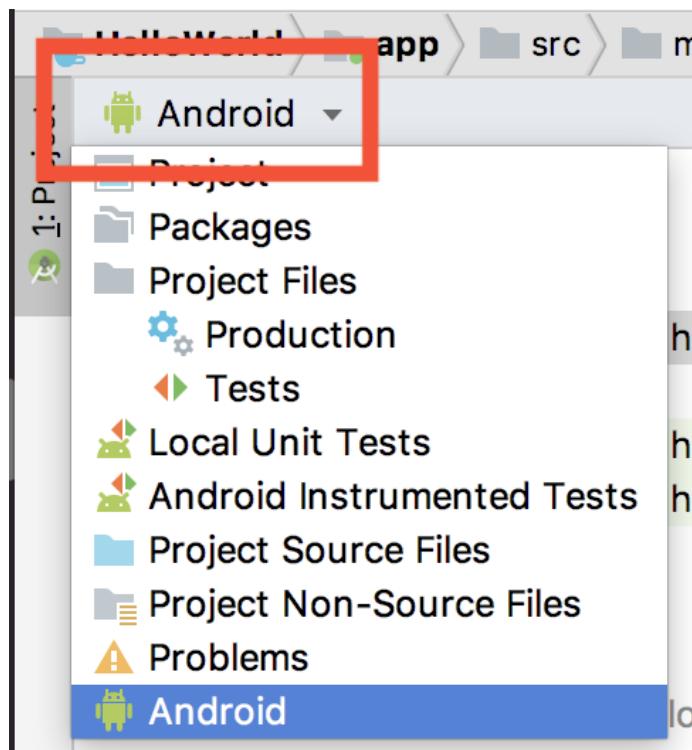
#### Painel do projeto

Se a guia **Project** ainda não estiver selecionada, selecione-a. Essa guia está na coluna da guia vertical no lado esquerdo da janela do Android Studio. Observe a imagem abaixo:



Para visualizar o projeto como uma hierarquia de projeto-padrão do Android, selecione **Android** no menu suspenso na parte superior do painel Projeto. (Android é o padrão.) Você pode visualizar os arquivos do projeto de

várias maneiras, incluindo a visualização dos arquivos como eles aparecem na hierarquia do sistema de arquivos. No entanto, o projeto é mais fácil de usar usando a visualização do Android.

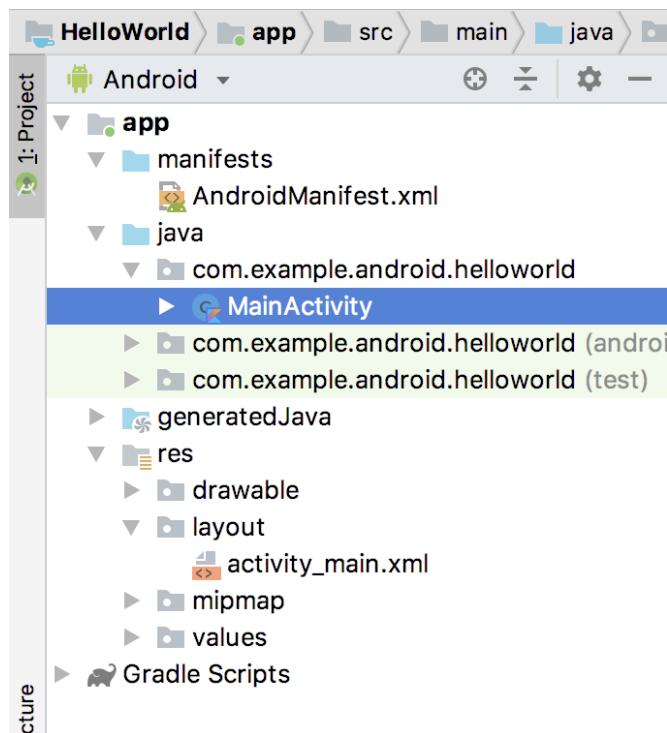


*Observação:* a referência ao painel Projeto é feita, quando definido como **Android**, como o painel **Project>Android**.

### Pasta app

Todo o código e recursos do seu aplicativo estão localizados na pasta **app**.

- No painel **Project>Android**, expanda pasta do aplicativo . Dentro da app pasta são quatro subpastas: manifests, java, generatedJava, e res.
- Expanda a pasta java e, em seguida, expanda a pasta **com.example.android.HelloWorld** para ver o arquivo **MainActivity Kotlin**.



A pasta **java** contém todo o código principal do Kotlin para um aplicativo Android. Existem razões históricas pelas quais seu código Kotlin aparece na pasta java. Essa convenção permite que o Kotlin interaja perfeitamente com código escrito na linguagem de programação Java, mesmo no mesmo projeto e aplicativo.

Os arquivos de classe do seu aplicativo estão contidos em três subpastas, como mostra a figura acima. A pasta **com.example.hello.helloworld** (ou o nome do domínio que você especificou) contém todos os arquivos para um pacote de aplicativos. Em particular, o arquivo *MainActivity.class* é o principal ponto de entrada do seu aplicativo. As outras duas pastas na pasta java são usadas para códigos relacionados ao teste, como testes de unidade.

**Nota:** No sistema de arquivos, seus arquivos Kotlin possuem uma **.kt** extensão e um ícone **K**. Na visualização Projeto, o Android Studio mostra o nome da classe (*MainActivity*) sem a extensão.

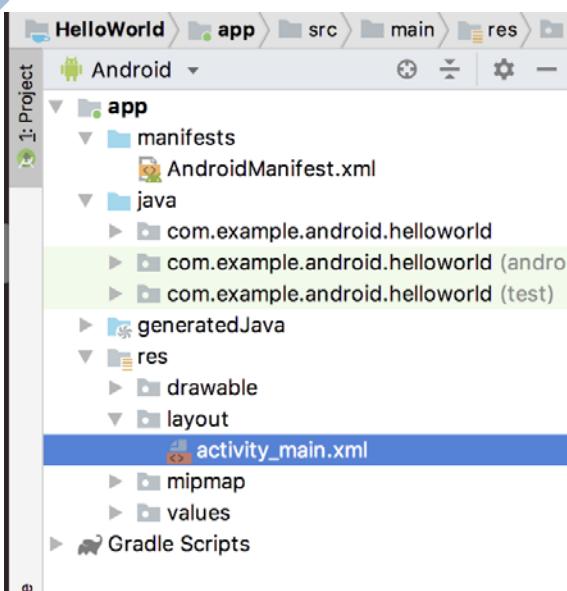
Observe a pasta generatedJava . Esta pasta contém arquivos que o Android Studio gera quando cria o aplicativo. Não edite nada nessa pasta, porque suas alterações podem ser substituídas quando você reconstruir o aplicativo. Mas é útil saber sobre essa pasta quando você precisa examinar esses arquivos durante a depuração.

### Pasta res

No painel **Project > Android** , expanda a pasta res

- A pasta res contém recursos. Recursos no Android são conteúdos estáticos usados em seus aplicativos. Os recursos incluem imagens, sequências de texto, layouts de tela, estilos e valores, como cores hexadecimais ou dimensões padrão.
- Aplicativos Android separam o código e os recursos do Kotlin o máximo possível. Isso torna muito mais fácil encontrar todas as strings ou ícones usados na interface do usuário do aplicativo. Além disso, quando você altera um desses arquivos de recursos, a alteração entra em vigor em todos os lugares em que o arquivo é usado no aplicativo.

Dentro da pasta res, expanda a pasta layout para ver o arquivo **activity\_main.xml**.

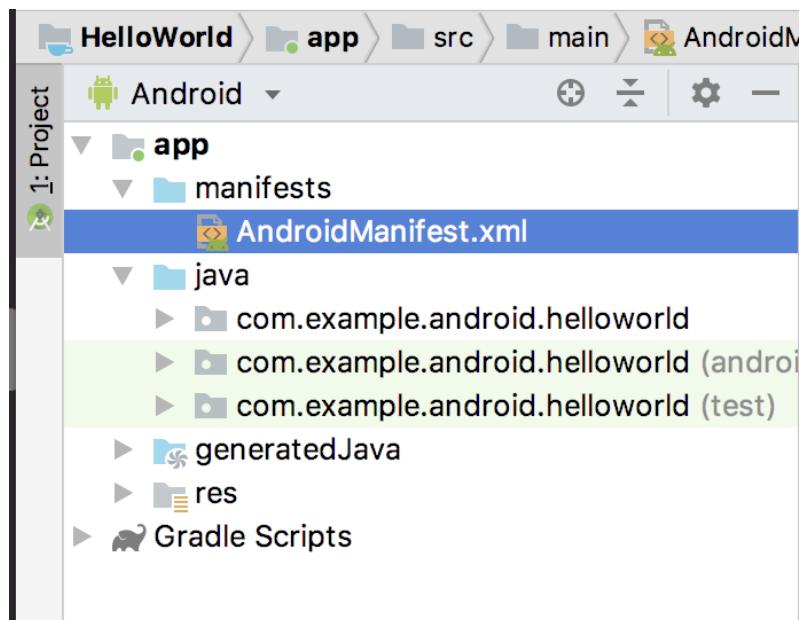


Seu **Activity** é geralmente associado com um arquivo de layout da interface do usuário, definido como um arquivo XML no diretório **res/layout**. Esse arquivo de layout geralmente é nomeado após sua activity. Nesse caso, o nome da activity é **MainActivity**, portanto, o layout associado é **activity\_main**.

### Pasta manifests e o AndroidManifest.xml

A pasta **manifests** contém arquivos que fornecem informações essenciais sobre o seu aplicativo para o sistema Android.

Expanda a pasta **manifests** e clique duas vezes em **AndroidManifest.xml** para abri-lo. O arquivo **AndroidManifest.xml** inclui detalhes que o sistema Android precisa para executar seu aplicativo, incluindo quais activitys fazem parte do aplicativo.



Observe que **MainActivity** é referenciado no elemento **<activity>**. Qualquer **Activity** em seu aplicativo deve ser declarado no manifest. Aqui está um exemplo para **MainActivity**:

```

<activity android:name=".MainActivity">
    <intent-filter>
        <action
            android:name="android.intent.action.MAIN" />

        <category
            android:name="android.intent.category.LAUNCHER" />

```

```
</intent-filter>
</activity>
```

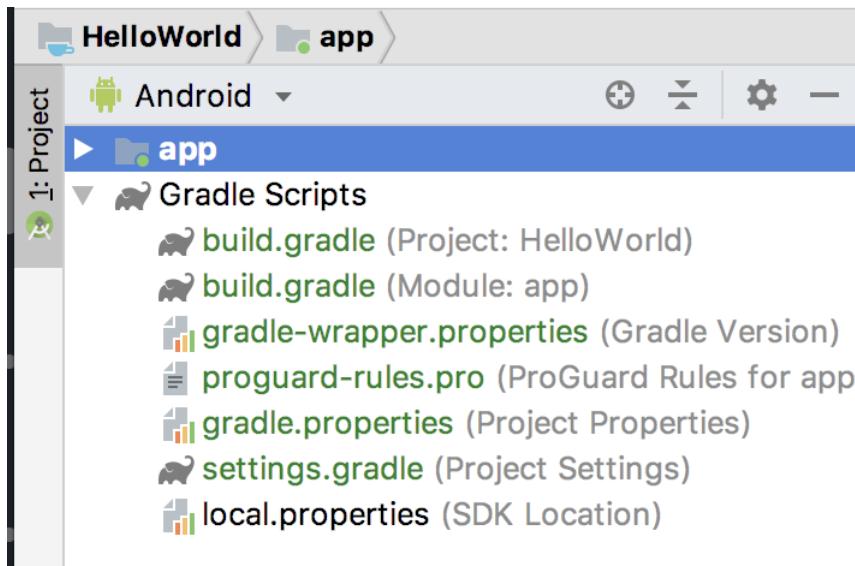
Observe o elemento **<intent-filter>** dentro **<activity>**. Os elementos **<action>** e **<category>** neste intent-filter informam ao Android por onde iniciar o aplicativo quando o usuário clica no ícone do inicializador.

O arquivo **AndroidManifest.xml** também é o lugar onde você define as permissões necessárias para o seu aplicativo. As permissões incluem a capacidade do aplicativo de ler contatos do telefone, enviar dados pela Internet ou acessar hardware, como a câmera do dispositivo.

### Pasta Gradle Scripts

O Gradle é um sistema de automação de construção que usa uma linguagem específica do domínio para descrever a estrutura, a configuração e as dependências do projeto do aplicativo. Quando você compila e executa seu aplicativo, você vê informações sobre a construção do Gradle em execução. Você também vê informações sobre o Kit de Pacotes do Android (APK) sendo instalado. ( APK é o formato de arquivo de pacote que o sistema operacional Android usa para distribuir e instalar aplicativos móveis.)

Expanda a pasta **Gradle Scripts**. No painel **Project > Android**, esta pasta contém todos os arquivos necessários para o sistema de compilação.



Procure o arquivo ***build.gradle* (Project: HelloWorld)**:

- Este arquivo contém as opções de configuração comuns a todos os módulos que compõem o seu projeto. Cada projeto do Android Studio contém um único arquivo de compilação Gradle de nível superior. Este arquivo define os repositórios e dependências do Gradle que são comuns a todos os módulos no projeto.

Procure o arquivo **build.gradle (Module: app)**:

- Além do arquivo build.gradle no nível do projeto , cada módulo possui um arquivo build.gradle próprio. O arquivo build.gradle de nível de módulo permite que você defina configurações de construção para cada módulo. (O aplicativo HelloWorld tem apenas um módulo, o módulo para o próprio aplicativo.) Esse arquivo build.gradle é o que se costuma editar com mais frequência ao alterar configurações de compilação no nível do aplicativo. Por exemplo, você edita esse arquivo build.gradle quando altera o nível do SDK ao qual seu aplicativo oferece suporte ou quando declara novas dependências na seção dependencies.

## **Dispositivo virtual (Emulador)**

### **Executar o aplicativo em um dispositivo virtual (emulador)**

Nesta tarefa, você usa o gerenciador do dispositivo virtual Android (AVD) para criar um dispositivo virtual (um emulador). O dispositivo virtual simula a configuração de um tipo específico de dispositivo Android. Então você usa esse dispositivo virtual para executar o aplicativo.

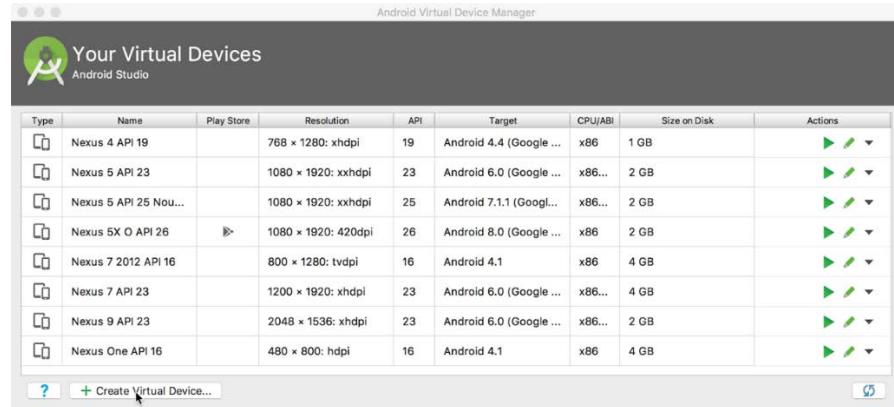
O Android Emulator é um aplicativo independente e possui seus próprios requisitos de sistema. Dispositivos virtuais podem usar muito espaço em disco. Se você encontrar algum problema, consulte Executar aplicativos no emulador do Android.

### **criar um dispositivo virtual Android (AVD)**

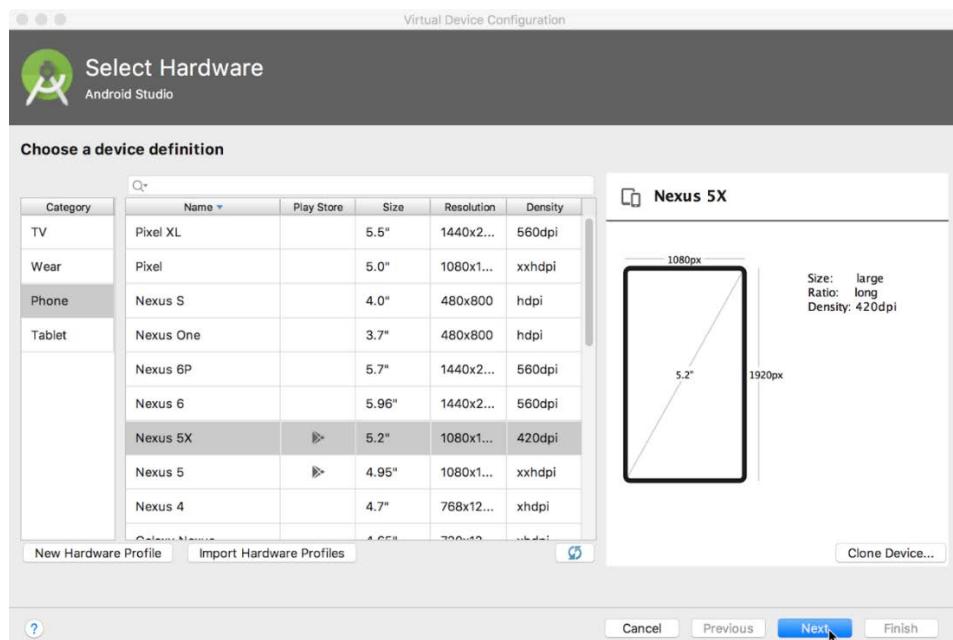
Para executar um emulador no seu computador, você precisa criar uma configuração que descreva o dispositivo virtual.

- No Android Studio, selecione **Ferramentas> Gerenciador de AVD** ou clique no ícone do **Gerenciador de AVD**  na barra de

ferramentas. A caixa de diálogo Seus dispositivos virtuais é exibida. Se você já criou dispositivos virtuais, a caixa de diálogo mostrará os mesmos (conforme mostrado na figura abaixo). Caso contrário, você verá uma lista em branco.

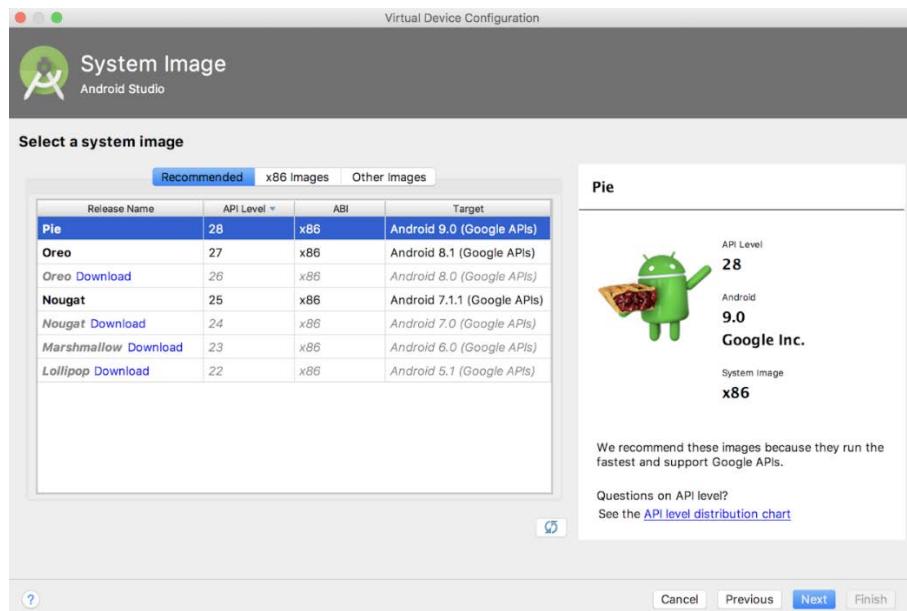


Clique em **+ Create a new virtual device** na parte inferior esquerda da caixa de diálogo. A caixa de diálogo **Select Hardware** é exibida, mostrando uma lista de dispositivos de hardware pré-configurados. Para cada dispositivo, a tabela fornece uma coluna para seu tamanho de exibição diagonal ( **Tamanho** ), resolução da tela em pixels ( **Resolução** ) e densidade de pixels ( **Densidade** ).



Selecione um dispositivo como **Nexus 5x** ou **Pixel XL** e clique em **Next**. A caixa de diálogo **Systgem image** é exibida.

Clique na guia **Recommended** e escolha qual versão do sistema Android deve ser executada no dispositivo virtual (como, por exemplo, **Pie**, **Oreo** ou **Nougat**).



**Nota:** muitas outras versões do sistema Android estão disponíveis do que as mostradas na guia **Recomendado**. Para vê-las, veja as guias **Imagens x86** e **Outras Imagens**.

Essas imagens usam muito espaço em disco, portanto, apenas algumas fazem parte da instalação original. Se um link de **download** estiver visível ao lado de uma imagem do sistema que você deseja usar, essa imagem não será instalada. Clique no link para iniciar o download, o que pode levar muito tempo. Quando o download estiver concluído, clique em **Finish**.

Depois de escolher uma imagem do sistema, clique em **Next**. A caixa de diálogo **Dispositivo Virtual Android (AVD)** é aberta. Verifique sua configuração e clique em **Finish**.

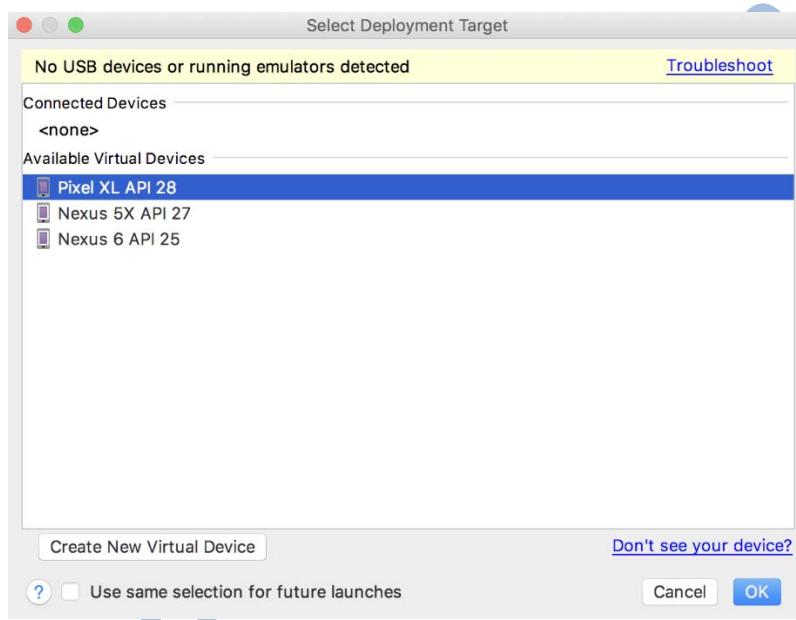
### executar o aplicativo no dispositivo virtual

Nesta tarefa, você finalmente executa seu novo aplicativo.

No Android Studio, selecione **Executar> Executar aplicativo** ou clique no ícone **Run**  na barra de ferramentas. A caixa de diálogo **Selecionar**

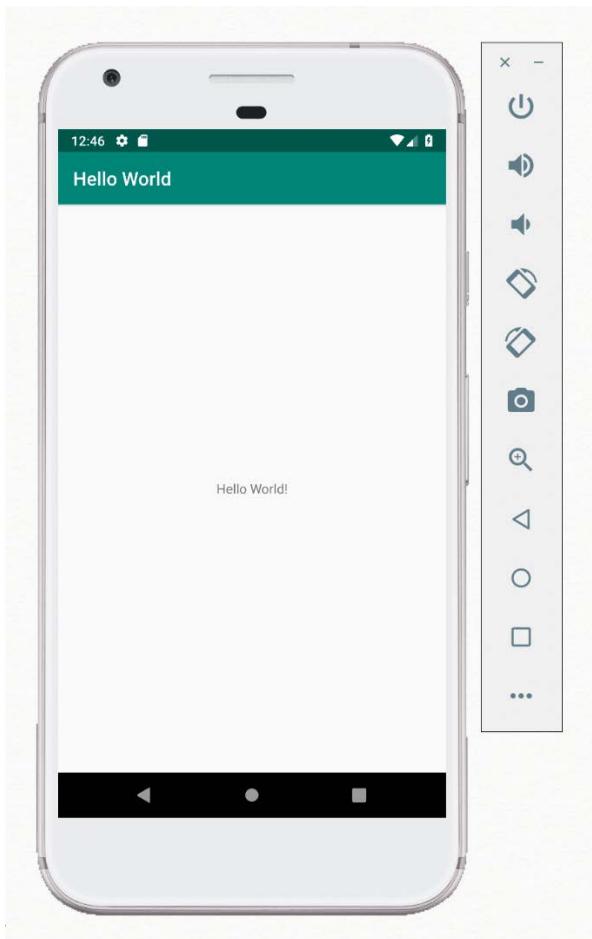
**Destino de Implementação** é exibida e avisa que nenhum dispositivo está disponível. Você verá esse aviso se não tiver um dispositivo físico conectado ao seu computador de desenvolvimento ou se ainda não tiver lançado um dispositivo virtual.

No diálogo **Selecionar Destino de Implementação**, em **Dispositivos Virtuais Disponíveis**, selecione o dispositivo virtual que você criou. Clique **OK**.



O emulador é iniciado e inicializado como um dispositivo físico. Dependendo da velocidade do seu computador, esse processo pode demorar um pouco. Seu aplicativo é criado e, quando o emulador está pronto, o Android Studio carrega o aplicativo APK no emulador e o executa.

É possível ver o aplicativo HelloWorld, conforme mostrado na figura a seguir.



## Executar o aplicativo em um dispositivo físico

Nesta tarefa, você executa seu aplicativo em um dispositivo móvel físico, como um smartphone ou tablet, se tiver um. Sempre teste seus aplicativos em dispositivos virtuais e físicos.

O que você precisa:

- Um dispositivo Android, como um smartphone ou tablet.
- Um cabo de dados USB para conectar seu dispositivo Android ao seu computador através da porta USB.
- Se você estiver usando um sistema Linux ou Windows, talvez seja necessário executar etapas extras. Consulte a documentação Executar aplicativos em um dispositivo de hardware . Você também pode precisar instalar o driver USB

apropriado para o seu dispositivo. Para drivers USB baseados no Windows, consulte Instalar drivers USB do OEM .

### ativar a depuração USB

Para permitir que o Android Studio se comunique com seu dispositivo Android, você deve ativar a depuração USB nas configurações de opções do desenvolvedor do dispositivo.

No Android 4.2 (Jellybean) e superior, as configurações das opções do desenvolvedor ficam ocultas por padrão. Para mostrar as opções do desenvolvedor e habilitar a depuração USB:

1. No seu dispositivo, abra **Configurações**, pesquise **Sobre o telefone**, toque em **Sobre o telefone** e toque em **Criar número** sete vezes.
2. Retornar para a página anterior ( **Configurações / Sistema** ). **Opções do desenvolvedor** aparecem na lista. Toque nas **opções do desenvolvedor** .
3. Selecione a **depuração USB** .

### execute seu aplicativo no dispositivo Android

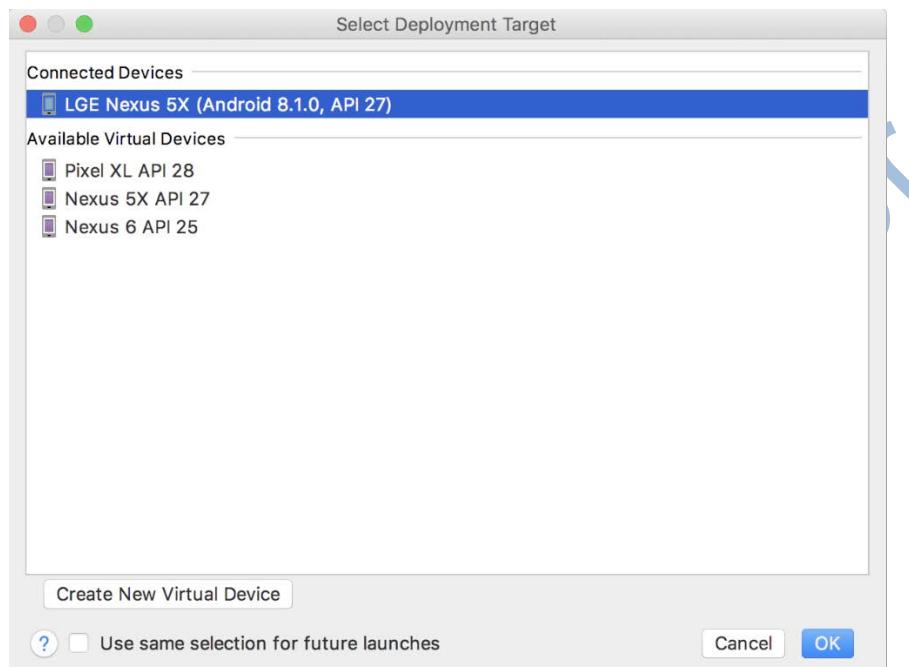
Agora você pode conectar seu dispositivo e executar o aplicativo no Android Studio.

Conecte o dispositivo Android à sua máquina de desenvolvimento com um cabo USB. Uma caixa de diálogo deve aparecer no dispositivo, pedindo para permitir a depuração USB.



Selecione a opção **Always allow** para lembrar deste computador. Toque em **OK** .

No seu computador, na barra de ferramentas do Android Studio, clique no botão **Run**  . A caixa de diálogo **Available virtual devices** é aberta com a lista de emuladores e dispositivos conectados disponíveis. Você deve ver seu dispositivo físico junto com quaisquer emuladores.



Selecione seu dispositivo e clique em **OK** . O Android Studio instala o aplicativo no seu dispositivo e o executa.

### Solução de problemas

Se o Android Studio não reconhecer seu dispositivo, tente o seguinte:

Desconecte o cabo USB e conecte-o novamente.

Reinic peace o Android Studio.

Se o computador ainda não encontrar o dispositivo ou declarar "não autorizado", siga estas etapas:

1. Desconecte o cabo USB.
2. No dispositivo, abra as **opções do desenvolvedor** no aplicativo **Configurações** .
3. Toque em **Revogar autorizações de depuração USB** .

4. Volte a ligar o dispositivo ao seu computador.
5. Quando solicitado, conceda autorizações.

Pode ser necessário instalar o driver USB apropriado para o seu dispositivo. Veja os aplicativos Executar em um dispositivo de hardware .

## Resumo

- Para instalar o Android Studio, acesse o Android Studio e siga as instruções para fazer o download e instalá-lo.
- Para ver a hierarquia do Android de um aplicativo no painel Projeto, clique na guia **Projeto** na coluna da guia vertical. Em seguida, selecione **Android** no menu suspenso na parte superior.
- Quando você precisar adicionar novas dependências ao seu projeto ou alterar as versões de dependência, edite o arquivo **build.gradle(Module:app)**.
- Todo o código e recursos de um aplicativo estão localizados nas pastas **app** e **res**. A pasta **java** inclui activitys, testes e outros componentes no código-fonte Kotlin ou Java (ou ambos). A pasta **res** contém recursos, como layouts, seqüências de caracteres e imagens.
- Para adicionar recursos, componentes e permissões ao seu aplicativo para Android, edite o arquivo **AndroidManifest.xml**. Todos os componentes do aplicativo, como activitys adicionais, devem ser declarados neste arquivo XML.
- Para criar um dispositivo virtual Android (um emulador) para executar seu aplicativo, use o Gerenciador de AVD .
- Para executar seu aplicativo em um dispositivo Android físico usando o Android Studio, ative a depuração USB no dispositivo. Para fazer isso, abra **Configurações > Sobre o telefone** e toque em **Criar número** sete vezes. Em seguida, abra **Configurações > Opções do desenvolvedor** e selecione **Depuração USB** .

## Layouts

O layout define a estrutura visual para uma interface do usuário, como a IU de uma activity ou de um widget de aplicativo. É possível declarar um layout de dois modos:

- **Declarar elementos da IU em XML.** O Android fornece um vocabulário XML direto que corresponde às classes e subclasses de View, como as de widgets e layouts.
- **Instanciar elementos do layout em tempo de execução.** O aplicativo pode criar objetos View e ViewGroup (e processar suas propriedades) programaticamente.

A estrutura do Android dá a flexibilidade de usar um desses métodos ou ambos para declarar e gerenciar a IU do aplicativo. Por exemplo, você pode declarar os layouts padrão do aplicativo em XML, incluindo os elementos da tela que aparecerão neles e em suas propriedades. Em seguida, você poderia adicionar código ao aplicativo que modificaria o estado dos objetos da tela, inclusive os declarados em XML, em tempo de execução.

- Pode-se usar também a ferramenta de Hierarchy Viewer para depurar layouts — ela revela os valores de propriedades do layout, desenha o esqueleto do layout com indicadores de preenchimento/margens e exibições totalmente renderizadas e a depuração pode ocorrer no emulador ou no dispositivo.
- A ferramenta layoutopt permite a rápida análise dos layouts e das hierarquias com relação a ineficiências e outros problemas.

A vantagem de declarar a IU em XML é separar melhor a apresentação do aplicativo do código que controla seu comportamento. As descrições da IU são externas ao código do aplicativo, ou seja, é possível modificá-las ou adaptá-las sem modificar o código-fonte e recompilar. Por exemplo, é possível criar layouts XML para diferentes orientações de tela, diferentes tamanhos de tela de dispositivos e diferentes idiomas. Além disso, a declaração de layout em XML facilita a exibição da estrutura da sua IU, o que facilita a depuração de problemas. Assim sendo, este documento se concentrará em ensinar a declarar o layout em XML. Se você estiver interessado em instanciar objetos View em tempo de execução, consulte as referências das classes [ViewGroup](#) e [View](#).

Em geral, o vocabulário XML para declarar elementos da IU segue rigorosamente a estrutura e a atribuição de nome às classes e aos métodos, em que os nomes de elementos correspondem a nomes de classe e nomes de atributos correspondem a métodos. Na verdade, a correspondência normalmente é tão direta que é possível supor qual atributo XML corresponde a determinado método de classe ou supor qual classe corresponde a certo elemento XML. Contudo, observe que nem todo vocabulário é idêntico. Em alguns casos, há pequenas diferenças de nome. Por exemplo, o elemento `EditText` tem um atributo `text` que corresponde a `EditText.setText()`.

## Programação do XML

Usando o vocabulário XML do Android, é possível projetar rapidamente layouts de IU e os elementos de tela intrínsecos do mesmo modo que se cria páginas Web em HTML — com uma série de elementos aninhados.

Cada arquivo de layout deve conter exatamente um elemento raiz, que deve ser um objeto View ou ViewGroup. Com o elemento raiz definido, é possível adicionar objetos ou widgets de layout extras como elementos filho para construir gradualmente uma hierarquia de View que define o layout. Por exemplo, eis um layout XML que usa um `LinearLayout` vertical para conter uma `TextView` e um `Button`:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical" >
    <TextView android:id="@+id/text"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello, I am a TextView" />
    <Button android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello, I am a Button" />
</LinearLayout>
```

Após declarar o layout no XML, salve o arquivo com uma extensão `.xml` no diretório `res/layout/` do projeto do Android para compilá-lo adequadamente.

## Carregamento do recurso XML

Ao compilar o aplicativo, cada arquivo de layout XML é compilado em um recurso `View`. Deve-se carregar o recurso de layout do código do aplicativo na implementação de retorno de chamada `Activity.onCreate()`. Para isso, chame `setContentView()`, passando a referência para o recurso de layout na forma: `R.layout.layout_file_name`. Por exemplo, se o layout XML for salvo como `main_layout.xml`, será necessário carregá-lo para a Atividade desta forma:

```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main_layout);
}
```

O método de retorno de chamada `onCreate()` na Atividade é chamado pela estrutura do Android quando ela é inicializada (veja a discussão sobre ciclos de vida no documento Activities ).

## Atributos

Cada objeto View e ViewGroup aceita uma variedade própria de atributos XML. Alguns atributos são específicos de um objeto View (por exemplo, `TextView` aceita o atributo `textSize`), mas esses atributos também são herdados de objetos View que possam estender essa classe. Alguns são comuns a todos os objetos View porque são herdados da classe View raiz (como o atributo `id`). Além disso, outros atributos são considerados "parâmetros do layout", que são os que descrevem determinadas orientações de layout do objeto View conforme definido pelo objeto ViewGroup pai daquele objeto.

### ID

Qualquer objeto View pode ter um ID de número inteiro associado para identificar exclusivamente o View dentro da árvore. Ao compilar o aplicativo, esse ID é referenciado como um número inteiro, mas o ID normalmente é atribuído no arquivo XML do layout como uma string, no atributo `id`. É um atributo XML comum a todos os objetos View (definido pela classe `View`) e você o usará com frequência. A sintaxe de um ID, dentro de uma tag XML, é:

```
android:id="@+id/my_button"
```

Um símbolo de arroba (@) no início da string indica que o analisador XML deve analisar e expandir o restante da string de ID e identificá-la como um recurso de ID. O símbolo de mais (+) significa que é um novo nome de recurso que precisa ser criado e adicionado aos recursos (no arquivo `R.java`). Há diversos outros recursos de ID oferecidos pela estrutura do Android. Ao referenciar um ID de recurso do Android, não é necessário ter o símbolo de mais, mas deve-se adicionar o espaço de nome do pacote `android` da seguinte maneira:

```
android:id="@android:id/empty"
```

Com o espaço de nome do pacote `android` em vigor, podemos referenciar um ID da classe de recursos `android.R` em vez de um da classe de recursos locais.

Para criar exibições e referenciá-las a partir do aplicativo, um modo padrão comum é:

1. Definir uma vista/widget no arquivo de layout e atribuir um ID exclusivo a ele/ela:

```
<Button android:id="@+id/my_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/my_button_text" />
```

2. Em seguida, crie uma instância do objeto de exibição e capture-a do layout (normalmente no método `onCreate()`):

```
Button myButton = (Button)
findViewById(R.id.my_button);
```

Definir IDs para objetos de exibição é importante ao criar um `RelativeLayout`. Em um layout relativo, exibições irmãs podem definir o layout relativo para outra exibição irmã referenciada pelo ID exclusivo.

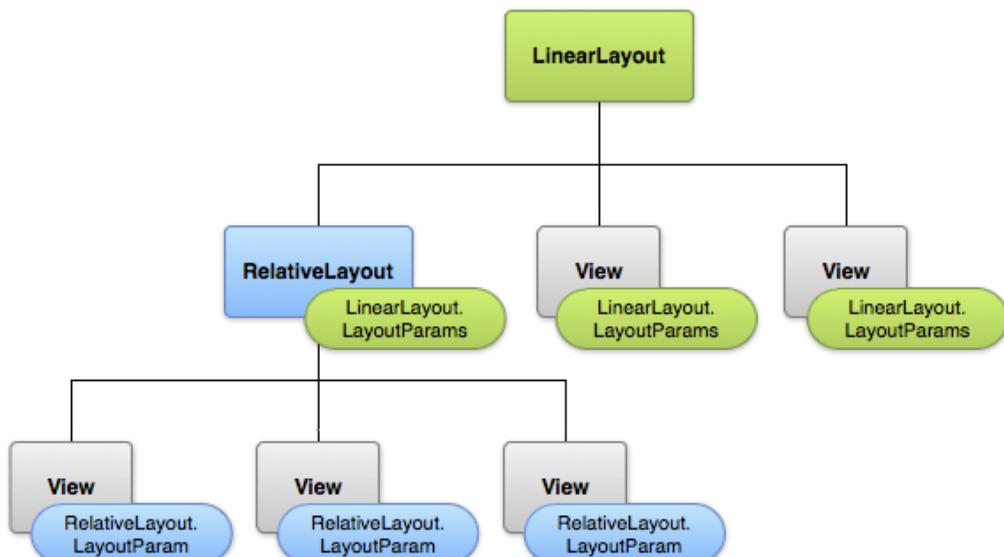
Os IDs não precisam ser exclusivo por toda a árvore, mas devem ser exclusivos dentro da parte da árvore em que se está procurando (que pode, com frequência, ser toda a árvore, portanto, é preferível ser totalmente exclusivo sempre que possível).

### Parâmetros do layout



Os atributos do layout XML chamados `layout_something` definem parâmetros para a View apropriados para a ViewGroup em que reside.

Cada classe ViewGroup implementa uma classe aninhada que estende `ViewGroup.LayoutParams`. Essa subclasse contém tipos de propriedade que definem o tamanho e a posição de cada exibição filha, conforme necessário para o grupo de exibições. Como se pode ver na figura 1, um grupo de exibições pais define parâmetros de layout para cada exibição filha (incluindo o grupo de exibições filhas).



**Figura 1.** Visualização de uma hierarquia de exibições com parâmetros de layout associados a cada uma delas.

Observe que cada subclasse LayoutParams tem a própria sintaxe para definir valores. Cada elemento filho deve definir LayoutParams apropriados para o pai, embora possa também definir diferentes LayoutParams para os próprios filhos.

Todos os grupos de exibições contêm largura e altura (`layout_width` e `layout_height`) e cada exibição é obrigatória para defini-las. Muitos LayoutParams também contêm margens e bordas opcionais.

É possível especificar largura e altura com medidas exatas, embora não seja recomendável na maioria dos casos. Em geral, usa-se uma destas constantes para definir a largura e a altura:

- **`wrap_content`** instrui a exibição a se redimensionar de acordo com as medidas exigidas pelo conteúdo.
- **`match_parent`** instrui a exibição a assumir o maior tamanho permitido pelo grupo de exibições pais.

Em geral, a especificação de largura e altura de um layout com unidades absolutas, como pixels, não é recomendada. Em vez disso, o uso de medidas relativas como unidades de pixel independentes de densidade (**`dp`**), **`wrap_content`** ou **`match_parent`** é uma abordagem melhor porque ajuda a garantir que o aplicativo exiba o conteúdo adequadamente nos diversos tamanhos de tela de dispositivos.

## Posição do layout

As views têm uma localização, expressa como um par de coordenadas *left* e *top* e duas dimensões, expressas como largura e altura. A unidade de localização e de dimensões é o pixel.

É possível recuperar a localização de uma exibição chamando os métodos `getLeft()` e `getTop()`. O primeiro retorna a coordenada esquerda, ou X, do retângulo que representa a exibição. O último retorna a coordenada top, ou Y, do retângulo que representa a exibição. Esses métodos retornam a localização da exibição em relação ao pai correspondente. Por exemplo, quando `getLeft()` retorna 20, significa que a exibição se localiza 20 pixels à direita da borda esquerda do seu pai direto.

Adicionalmente, diversos métodos de conveniência são oferecidos para evitar cálculos desnecessários, chamados `getRight()` e `getBottom()`. Esses métodos retornam as coordenadas das bordas direita e inferior do retângulo que representa a exibição. Por exemplo, chamar `getRight()` é semelhante ao seguinte cálculo: `getLeft() + getWidth()`.

## Tamanho, preenchimento e margens

O tamanho de uma exibição é expresso por largura e altura. As exibições, na verdade, têm dois pares de valores de largura e altura.

O primeiro par é conhecido como *largura medida* e *altura medida*. Essas dimensões definem o tamanho que a exibição terá dentro da exibição pai. As dimensões medidas podem ser obtidas chamando `getMeasuredWidth()` e `getMeasuredHeight()`.

O segundo par é simplesmente conhecido como *largura* e *altura* ou, às vezes, *largura do desenho* e *altura do desenho*. Essas dimensões definem o tamanho real da exibição na tela, em tempo de desenho e após o layout. Esses valores podem diferir da largura e da altura medidas. Os valores de largura e altura podem ser obtidos chamando `getWidth()` e `getHeight()`.

Para medir as dimensões, a exibição leva em conta o preenchimento. O preenchimento é expresso em pixels para a esquerda, a direita e as partes de cima e de baixo da exibição. O preenchimento pode ser usado para compensar o conteúdo da exibição por um número específico de pixels. Por exemplo, um preenchimento à esquerda de 2 empurrará o conteúdo da exibição em 2 pixels para a direita da borda esquerda. O preenchimento pode ser definido usando o método `setPadding(int, int, int, int)` e consultado chamando `getPaddingLeft()`, `getPaddingTop()`, `getPaddingRight()` e `getPaddingBottom()`.

Mesmo que cada exibição possa definir um preenchimento, ela não fornece nenhuma compatibilidade com margens. No entanto, os grupos de exibições oferecem essa compatibilidade.

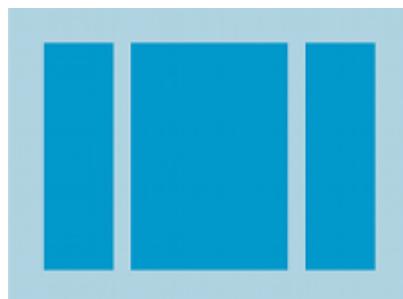
## Layouts comuns



Cada subclasse da classe `ViewGroup` fornece um modo exclusivo de exibir as exibições aninhadas dentro dela. Abaixo estão alguns dos tipos de layout mais comuns criados na plataforma Android.

**Observação:** Embora seja possível aninhar um ou mais layouts em outro layout para obter o projeto de IU, deve-se procurar manter a hierarquia do layout a menos profunda possível. O layout carrega mais rápido se tiver menos layouts aninhados (uma hierarquia de exibições grande é melhor do que uma hierarquia de exibições profunda).

### Linear Layout



Layout que organiza os filhos em uma única linha horizontal ou vertical. Ele cria uma barra de rolagem se o comprimento da janela excede o comprimento da tela.

### *Relative Layout*



Permite especificar a localização de objetos filhos relativos entre si (filho A à esquerda do filho B) ou relativos aos pais (alinhados no topo do pai).

### *WebView*



Exibe páginas da Web.

## **Criação de layouts com um adapter**

Quando o conteúdo do layout é dinâmico ou não predeterminado, é possível usar um layout que torne **AdapterView** uma subclasse para preencher o layout com exibições em tempo de execução. Uma subclasse da classe **AdapterView** usa um **Adapter** para agrupar dados ao seu layout. O **Adapter** se comporta como um intermediário entre a fonte dos dados e o layout do **AdapterView** — o **Adapter** recupera os dados (de uma fonte como uma matriz ou uma consulta de banco de dados) e converte cada entrada em uma exibição que pode ser adicionada ao layout do **AdapterView**.

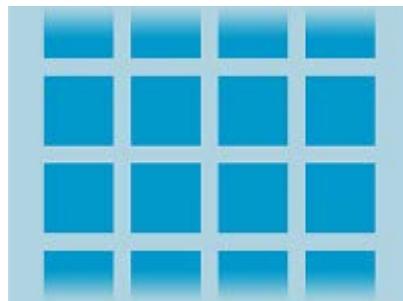
Alguns layouts comuns retornados por um adaptador:

### *ListView Layout*



Exibe uma lista de rolagem de coluna única.

## GridView Layout



Exibe uma grade de rolagem de colunas e linhas.

### Atributos de Layout

Cada layout tem um conjunto de atributos que definem suas propriedades visuais. Existem alguns atributos comuns entre todos os layouts e alguns são atributos específicos para cada layout. Estes são os atributos comuns aplicados a todos os layouts:

Attributo	Descrição
android:id	Essa é a ID que identifica de forma única qualquer componente dentro de uma view.
android:layout_width	Largura do layout.
android:layout_height	Altura do layout
android:layout_marginTop	Declara espaço entre o componente e o topo do layout.
android:layout_marginBottom	Declara espaço entre o componente e o rodapé do layout.
android:layout_marginLeft	Declara espaço entre o componente e o lado esquerdo do layout.
android:layout_marginRight	Declara espaço entre o componente e o lado direito do layout.
android:layout_gravity	Especifica como os Views-filhos podem ser posicionadas
android:layout_weight	Especifica quanto do espaço extra no layout deve ser destinada à View.
android:layout_x	Especifica as coordenadas no eixo x do layout

android:layout_y	Especifica as coordenadas no eixo y do layout
android:paddingLeft	paddingLeft (espaço interno de um componente em relação a sua margem) preenchido para o layout no lado esquerdo
android:paddingRight	paddingRight (espaço interno de um componente em relação a sua margem) preenchido para o layout no lado direito
android:paddingTop	paddingTop (espaço interno de um componente em relação a sua margem) preenchido para o layout no topo
android:paddingBottom	paddingBottom (espaço interno de um componente em relação a sua margem) preenchido para o layout no rodapé.

Aqui largura e altura são a dimensão do layout / view que pode ser especificado em termos de DP (Pixels independente de densidade), SP (Pixels independente de escala), PT (pontos que é 1/72 de polegada), PX (pixels), MM (milímetros) e finalmente em (polegadas).

É possível especificar a largura e altura com medidas exatas, mas frequentemente, você vai usar uma dessas constantes para definir a largura ou a altura:

- **android: layout\_width = wrap\_content** diz a sua opinião sobre o tamanho próprio para as dimensões exigidas pelo seu conteúdo.
- **android: layout\_width = fill\_parent** diz a sua view para se tornar tão grande quanto a sua view principal.

O atributo gravity desempenha papel importante no posicionamento do objeto de exibição e pode possuir um ou mais (separados por '|') dos seguintes valores constantes.

Constante	Valor	Descrição
top	0x30	Empurra objeto para o topo de seu contêiner, não alterando seu tamanho.
bottom	0x50	Empurra objeto para o baixo de seu contêiner, não alterando seu tamanho.
left	0x03	Empurra objeto para a esquerda de seu contêiner, não alterando seu tamanho.
right	0x05	Empurra objeto para a direita de seu contêiner, não alterando seu tamanho.
center_vertical	0x10	Coloca o objeto no centro vertical do seu contêiner, não alterando seu tamanho.
fill_vertical	0x70	Aumenta o tamanho vertical do objeto, se for necessário para que ele preencha completamente o seu contêiner.
center_horizontal	0x01	Coloca o objeto no centro horizontal de seu contêiner, não alterando seu tamanho.
fill_horizontal	0x07	Aumenta o tamanho horizontal do objeto, se for necessário para que ele preencha completamente o seu contêiner.
center	0x11	Coloca o objeto no centro do contêiner, tanto o eixo vertical e horizontal, sem alterar o seu tamanho.
fill	0x77	Aumenta o tamanho horizontal e vertical do objeto, se for necessário para que ele preencha completamente o seu contêiner.
clip_vertical	0x80	Opção adicional que pode ser configurado para ter a parte superior e / ou bordas inferiores da view-filho cortada

		nos limites do seu contêiner. O clip será baseado na gravidade vertical: uma gravidade-topo cortará a borda inferior, uma gravidade-inferior cortará a borda superior, e nenhuma aplicada cortará ambas as bordas.
clip_horizontal	0x08	Opção adicional que pode ser configurado para ter a esquerda e / ou bordas direita da view-filha cortada nos limites do seu contêiner. O clip será baseado na gravidade horizontal: a gravidade cortará a borda direita, a gravidade direita cortará a borda esquerda, e nenhuma delas aplicada cortará ambas as bordas.
start	0x00800003	Empurra o objeto para o início de seu contêiner, não alterando seu tamanho.
end	0x00800005	Empurra o objeto para o final de seu contêiner, não alterando seu tamanho.

## Identificação de um componente dentro de uma View

Um objeto na View pode ter um ID único atribuído a ele que irá identificá-lo exclusivamente da árvore na View. A sintaxe para uma ID, dentro de uma tag XML é:

```
android:id="@+id/my_button"
```

Uma breve descrição dos sinais @ e + :

- O (@) at-símbolo no início da cadeia indica que o parser XML deve analisar e expandir o resto da cadeia ID e identificá-lo como um recurso ID.

O símbolo de mais (+) significa que este é um novo nome do recurso que deve ser criada e adicionada aos nossos recursos.

## Aplicativos – criação

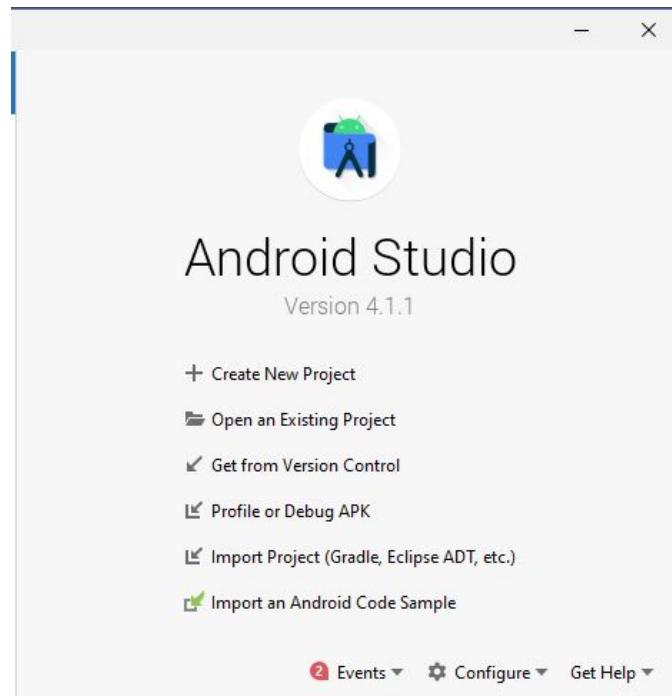
Um projeto do Android Studio contém um ou mais módulos que mantêm seu código organizado em unidades de funcionalidade discretas. Esta página mostra como começar um novo projeto ou importar um existente.

### Comece um novo projeto

O Android Studio facilita a criação de aplicativos Android para diversos tipos de dispositivo, como celulares, tablets, TV, Wear e Google Glass. O assistente **New Project** permite escolher os tipos de dispositivo para o seu aplicativo e inserir na estrutura do projeto tudo o que é necessário para começar. Siga as etapas abaixo para criar um novo projeto.

### Etapa 1: iniciar e configurar o projeto

Se você não tem um projeto aberto, o Android Studio mostra a tela de boas-vindas. Para criar um novo projeto, clique em **Start a New Android Studio project**.

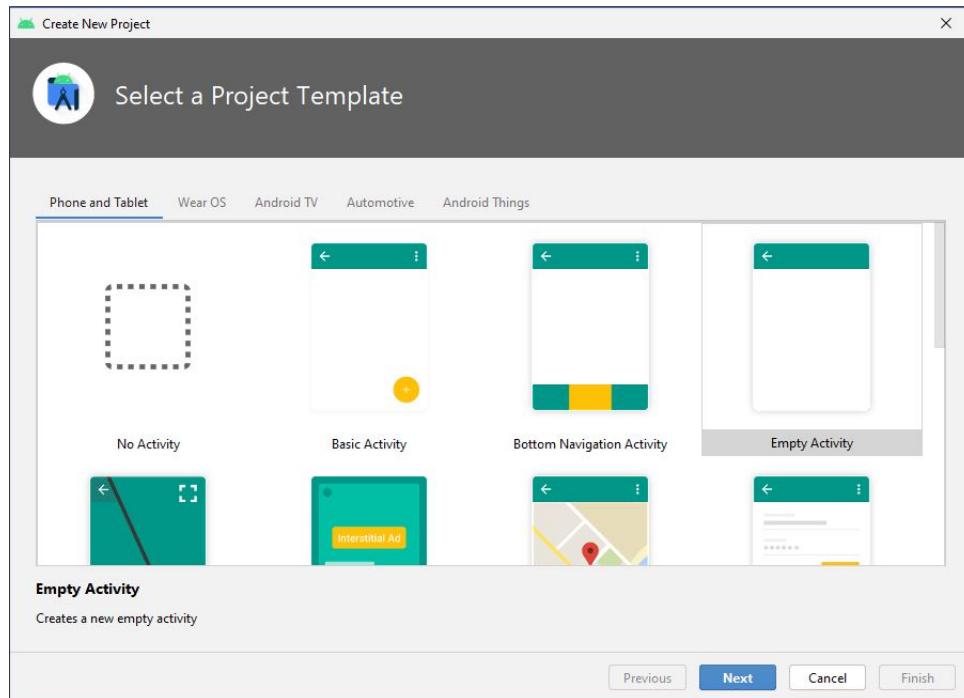


Se você já tem um projeto aberto, o Android Studio mostra o ambiente de desenvolvimento. Para criar um novo projeto, clique em **File > New > New Project**.

A janela seguinte permite configurar o nome do aplicativo, o nome do pacote e o local do seu projeto.

### Etapa 2: adicionar uma activity

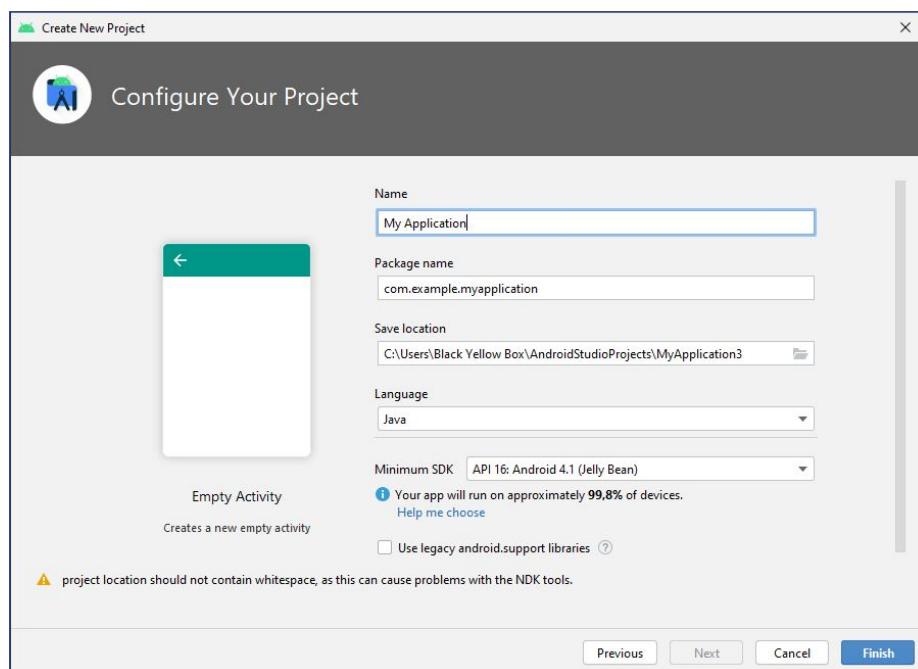
A tela seguinte permite selecionar um tipo de atividade para adicionar ao aplicativo, conforme mostrado na figura. Essa tela exibe um conjunto de atividades diferentes para cada tipo de dispositivo selecionado na etapa anterior.



Escolha um tipo de activity e clique em **Next**.

### Etapa 3: configurar a activity

A próxima tela é voltada para a configuração da atividade a adicionar ao aplicativo, como mostrado na figura.

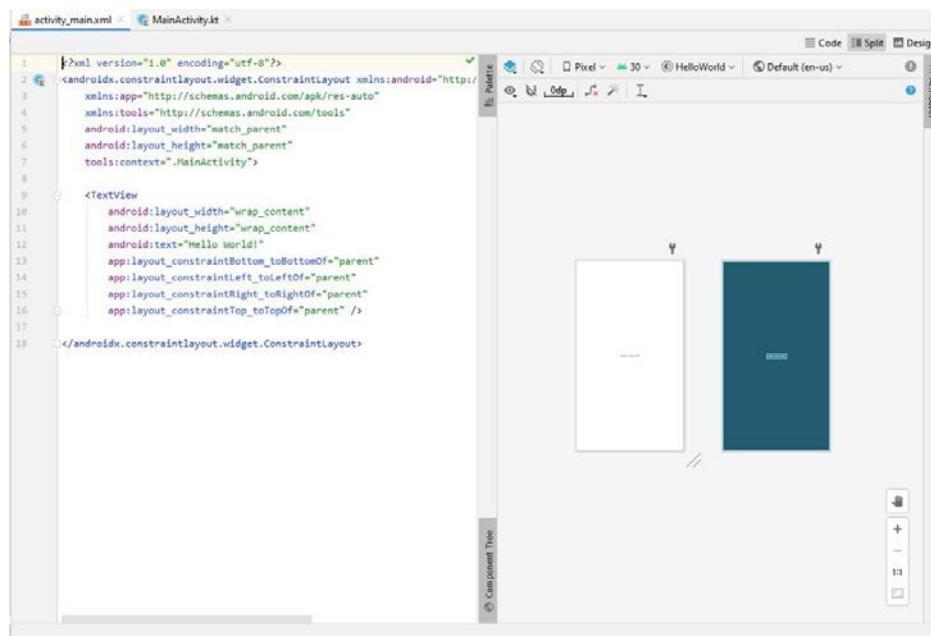


A tela **Configure the Activity**.

Insira o nome da atividade, o nome do layout e o título da atividade. Em seguida, clique em **Finish**.

#### **Etapa 4: desenvolver o aplicativo**

O Android Studio cria a estrutura padrão do seu projeto e abre o ambiente de desenvolvimento. Se o aplicativo oferecer suporte a mais de um tipo de dispositivo, o Android Studio criará uma pasta de módulos com os arquivos-fonte completos para cada um deles, como mostrado na figura.



**Figura 6.** Estrutura do projeto para um aplicativo recentemente criado.

#### **Importar um projeto**

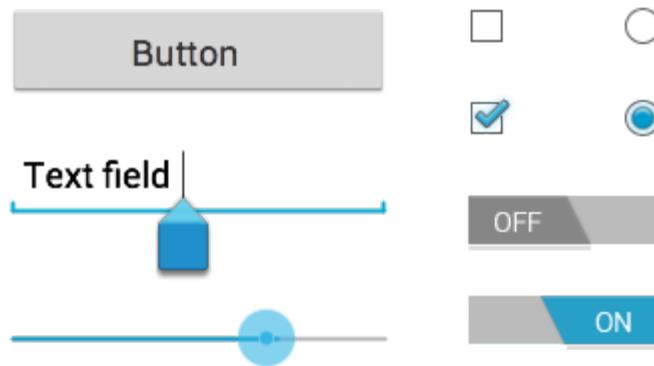
Para importar um projeto no Android Studio, faça o seguinte:

1. Clique em **File > New > Import Project**.
2. Na janela **Select Eclipse or Gradle Project to Import** que aparece, navegue para o diretório-raiz do projeto que quer importar.
3. Clique em **OK**.

Com isso, o Android Studio abre o projeto em uma nova janela do IDE.

Se estiver importando um projeto do controle de versão, use o menu **File > New > Project from Version Control**.

#### **Controles de entrada**



Controles de entrada são os componentes interativos na interface do usuário de seu aplicativo. O Android oferece uma ampla variedade de controles que podem ser usados na IU, como botões, campos de texto, barras de busca, caixas de seleção, botões de zoom, botões de alternância e muito mais.

Adicionar um controle de entrada à IU é tão simples quanto adicionar um elemento XML ao [layout\\_XML](#). Por exemplo, eis um layout com um campo de texto e um botão:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="horizontal">
    <EditText android:id="@+id/edit_message"
        android:layout_weight="1"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:hint="@string/edit_message" />
    <Button android:id="@+id/button_send"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/button_send"
        android:onClick="sendMessage" />
</LinearLayout>
```

Cada controle de entrada tem suporte para um conjunto específico de eventos de entrada, portanto, você pode processar eventos como quando o usuário digita texto ou toca em um botão.

### Controles comuns

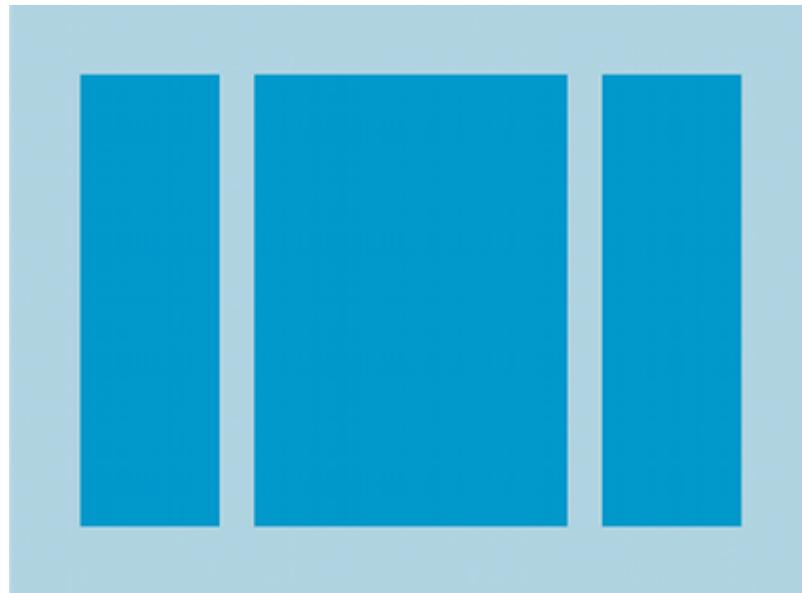
Eis uma lista de alguns dos controles comuns que você pode usar no aplicativo. Siga os links para saber mais sobre o uso de cada um deles.

**Observação:** O Android fornece muitos outros controles além dos listados aqui. Navegue no pacote `android.widget` para descobrir mais. Se o aplicativo precisa de um tipo específico de controle de entrada, você pode criar os próprios **componentes personalizados**.

Tipo de controle	Descrição	Classes relacionadas
<code>Botão</code>	Botão de pressão que pode ser pressionado ou clicado pelo usuário para realizar uma ação.	<code>Button</code>
<code>Campo_de_texto</code>	Campo de texto editável. É possível usar o widget <code>AutoCompleteTextView</code> para criar um widget de entrada de texto que forneça sugestões para preenchimento automático	<code>EditText, AutoCompleteTextView</code>
<code>Caixa de seleção</code>	Chave liga/desliga que pode ser alternada pelo usuário. Use caixas de seleção ao apresentar aos usuários um grupo de opções selecionáveis que não sejam mutualmente exclusivas.	<code>CheckBox</code>
<code>Botão de opção</code>	Similar às caixas de seleção, exceto que somente uma opção pode ser selecionada no grupo.	<code>RadioGroup, RadioButton</code>
<code>Botão de alternar</code>	Botão liga/desliga com um indicador de luz.	<code>ToggleButton</code>
<code>Sz\hnbbnSeletores</code>	Caixa de diálogo para que os usuários selecionem um valor para um conjunto usando botões para cima/para baixo ou via gesto de deslizar. Use um widget <code>DatePicker</code> para inserir os valores para a data (mês, dia, ano) ou um widget <code>TimePicker</code> para inserir valores para um horário (hora, minuto, AM/PM), que serão formatados automaticamente para a localidade do usuário.	<code>DatePicker, TimePicker</code>

## LinearLayout

`LinearLayout` é um grupo de exibições que alinha todos os filhos em uma única direção vertical ou horizontal. Você pode especificar a direção do layout com o atributo `android:orientation`.



Todos os filhos de um LinearLayout são empilhados um após o outro. Portanto, uma lista vertical terá apenas um filho por linha, independentemente de sua largura, e uma lista horizontal terá altura de apenas uma linha (a altura do filho mais alto, mais preenchimento). Um LinearLayout respeita margens entre filhos e a gravidade (alinhamento à direita, no centro ou à esquerda) de cada filho.

### Ponderação do layout

#### Filhos com a mesma ponderação

Para criar um layout linear em que cada filho usa a mesma quantidade de espaço na tela, defina a `android:layout_height` de cada exibição "importância" a uma exibição em termos de quanto espaço ela deve ocupar na tela. Um valor de ponderação maior permite que a exibição se expanda para preencher todo o espaço restante na exibição pai. As exibições filhas podem especificar um valor de ponderação. Com isso, todo o espaço restante no grupo de exibições será designado aos filhos na proporção de sua ponderação declarada. A ponderação padrão é zero.

Por exemplo, para três campos de texto, dois com ponderação 1 e o terceiro sem ponderação, esse terceiro campo de texto sem ponderação não aumentará e ocupará apenas a área necessária para seu conteúdo. Os outros dois expandirão igualmente para preencher o espaço restante depois que os três campos forem medidos. Se a ponderação 2 (em vez de 0) for atribuída ao terceiro campo, ele será mais importante que os outros dois e, portanto, ficará com metade do espaço total restante e os primeiros compartilharão o resto igualmente.

## Atributos Linear Layout

Estes são os atributos importantes, específicos, para Linear Layout:

Atributo	Descrição
android:id	Essa é a ID que identifica de forma única qualquer componente dentro de uma view.
android:baselineAligned	Este deve ser um valor booleano - "verdadeiro" ou "falso" - e impede o layout de alinhar as linhas de base de seus objetos-filhos.
android:divider	Isto proporcionar desenhar um linha como um separador vertical entre botões. Você usa um valor de cor, na forma de "#rgb", "#argb", "#rrggbb", ou "#aarrggbb".
android:gravity	Especifica como um objeto deve ser posicionado, em ambos os eixos X e Y. Os valores possíveis são parte top, bottom, left, right, center, center_vertical, center_horizontal etc.
android:orientation	Isso especifica o sentido de disposição do layout; você vai usar "horizontal" para uma linha, "vertical" para uma coluna. O padrão é horizontal.

## Projeto LinearLayout

O LinearLayout é o gerenciador de layout mais básico fornecido pelo Android. Ele organiza as visualizações- filho, tanto horizontal quanto verticalmente, com base na propriedade de orientação especificada. O valor da propriedade de orientação pode ser horizontal ou vertical.

1	Você vai usar o Android Studio IDE para criar um aplicativo para Android e nomeá-lo como LinearLayoutApp.
2	Modificar o conteúdo padrão da pasta res /layout / activity_main.xml para incluir botões e caixas de texto no Linear Layout.
3	Definir as constantes requeridas Username, Password, Enviar, Botão 1, Botão 1, Botão 2, Botão 3 em res /values/ strings.xml

4

Execute o aplicativo para iniciar o emulador Android e verificar o resultado das mudanças feitas na aplicação.

Abaixo está a captura de tela do resultado final:



Este é o trecho de código do conteúdo do arquivo de layout localizado na res /layout / values/strings.xml. Implemente esse código em seu projeto:

```
<resources>
    <string name="app_name">LinearLayout</string>
    <string name="login">Login</string>
    <string name="username">UserName</string>
    <string name="password">Password</string>
    <string name="enviar">Enviar</string>
    <string name="button1">Botão 1</string>
    <string name="button2">Botão 2</string>
    <string name="button3">Botão 3</string>
</resources>
```

Este é o trecho de código do conteúdo do arquivo de layout localizado na res /layout / activity\_main.xml. Implemente esse código em seu projeto:

```
<ScrollView
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal">

    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:layout_gravity="center"
        android:orientation="vertical"
        tools:context=".LinearLayout">

        <LinearLayout
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical">

            <TextView
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:background="#009688"
                android:gravity="center"
                android:padding="5dp"
                android:text="@string/login"
                android:textColor="#ffffff" />

            <LinearLayout
                android:layout_width="fill_parent"
                android:layout_height="wrap_content"
                android:orientation="horizontal"
                android:paddingTop="10dp">

                <TextView
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:paddingLeft="10dp"
                    android:paddingTop="10dp"
                    android:text="@string/username" />

                <EditText
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:layout_marginLeft="40dp"
                    android:layout_weight="0.50"
                    android:paddingTop="10dp" />
            
```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:paddingLeft="10dp"
        android:paddingTop="10dp"
        android:text="@string/password" />

    <EditText
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:layout_marginLeft="40dp"
        android:layout_weight="0.50"
        android:paddingTop="10dp" />
</LinearLayout>

<Button
    android:layout_width="wrap_content"
    android:layout_height="30dp"
    android:layout_gravity="center"
    android:layout_marginBottom="5dp"
    android:layout_marginTop="10dp"
    android:background="#009688"
    android:paddingLeft="15dp"
    android:paddingRight="15dp"
    android:text="@string/enviar"
    android:textColor="#ffffff"
    android:textSize="15sp" />

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="0.1dp"
    android:background="#009688"
    android:orientation="horizontal" />

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginBottom="10dp"
    android:layout_marginTop="10dp"
    android:orientation="horizontal" >

    <Button
        android:layout_width="100dp"
        android:layout_height="wrap_content"
        android:layout_marginBottom="2dp"
        android:layout_marginLeft="5dp"
        android:layout_marginRight="2dp"
        android:layout_marginTop="2dp"
        android:background="#009688"
        android:text="@string/button1"
        android:textColor="#fff"
        android:textStyle="bold" />

    <Button
        android:layout_width="100dp"
        android:layout_height="wrap_content"
        android:layout_margin="2dp"
        android:background="#009688"
        android:text="@string/button2"
        android:textColor="#fff"
        android:textStyle="bold" />
```

```
<Button
    android:layout_width="100dp"
    android:layout_height="wrap_content"
    android:layout_margin="2dp"
    android:background="#009688"
    android:text="@string/button3"
    android:textColor="#fff"
    android:textStyle="bold" />
</LinearLayout>

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="0.1dp"
    android:background="#009688"
    android:orientation="horizontal" />

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginBottom="10dp"
    android:layout_marginTop="10dp"
    android:orientation="vertical">

    <Button
        android:layout_width="100dp"
        android:layout_height="wrap_content"
        android:layout_margin="2dp"
        android:background="#009688"
        android:text="@string/button1"
        android:textColor="#fff"
        android:textStyle="bold" />

    <Button
        android:layout_width="100dp"
        android:layout_height="wrap_content"
        android:layout_margin="2dp"
        android:background="#009688"
        android:text="@string/button2"
        android:textColor="#fff"
        android:textStyle="bold" />

    <Button
        android:layout_width="100dp"
        android:layout_height="wrap_content"
        android:layout_margin="2dp"
        android:background="#009688"
        android:text="@string/button3"
        android:textColor="#fff"
        android:textStyle="bold" />
</LinearLayout>

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="0.1dp"
    android:background="#009688"
    android:orientation="horizontal" />

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
```

```

        android:layout_marginBottom="10dp"
        android:layout_marginTop="10dp"
        android:orientation="vertical">

        <Button
            android:layout_width="100dp"
            android:layout_height="wrap_content"
            android:layout_gravity="left"
            android:layout_margin="2dp"
            android:layout_weight="1"
            android:background="#009688"
            android:text="@string/button1"
            android:textColor="#fff"
            android:textStyle="bold" />

        <Button
            android:layout_width="100dp"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:layout_margin="2dp"
            android:layout_weight="2"
            android:background="#009688"
            android:text="@string/button2"
            android:textColor="#fff"
            android:textStyle="bold" />

        <Button
            android:layout_width="100dp"
            android:layout_height="wrap_content"
            android:layout_gravity="right"
            android:layout_margin="2dp"
            android:layout_weight="3"
            android:background="#009688"
            android:text="@string/button3"
            android:textColor="#fff"
            android:textStyle="bold" />
    </LinearLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="0.1dp"
        android:background="#009688"
        android:orientation="horizontal" />
</LinearLayout>

</ScrollView>

```

 Este é o trecho de código do conteúdo do arquivo de layout localizado na main /java / nome.do.seu.pacote/MainActivity. Implemente esse código em seu projeto:

```

import android.os.Build
import android.support.v7.app.AppCompatActivity
import android.os.Bundle
import android.view.Window
import android.view.WindowInsets
import android.view.WindowManager

```

```

class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)

        setContentView(R.layout.activity_main)
        @Suppress("DEPRECATION")
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.R) {

            window.insetsController?.hide(WindowInsets.Type.statusBars())
        } else {
            window.setFlags(
                WindowManager.LayoutParams.FLAG_FULLSCREEN,
                WindowManager.LayoutParams.FLAG_FULLSCREEN
            )
        }
    }
}

```

Execute seu projeto. Confira o resultado.

## Relative Layout

RelativeLayout é um ViewGroup que exibe visualizações filhas em posições relativas. A posição de cada view pode ser especificada em relação aos “elementos-irmãos” (como a esquerda ou abaixo de outra view) ou em posições relativas à área relativa ao RelativeLayout (tal como alinhado à parte inferior, esquerda ou central).



Um RelativeLayout é um utilitário muito poderoso para projetar uma interface de usuário, pois pode eliminar views group aninhadas e manter sua hierarquia de layout plana, o que melhora o desempenho. Caso o projeto esteja usando vários grupos LinearLayout aninhados, você poderá substituí-los por um único RelativeLayout.

## Posicionando Views

RelativeLayout permite que as “views-filhas” especifiquem a posição relativa à view principal ou a outra (especificada por ID). Então, é possível alinhar dois elementos pela borda direita, ou fazer um abaixo do outro,

centralizado na tela, centralizado à esquerda e assim sucessivamente. Por padrão, todas as “views-filhas” são desenhadas na parte superior esquerda do layout, portanto, você deve definir a posição de cada view usando as várias propriedades de layout disponíveis no `RelativeLayout.LayoutParams`.

Algumas das muitas propriedades de layout disponíveis para views em um `RelativeLayout` incluem:

Android: `layout_alignParentTop`

Se “`true`”, faz com que a borda superior da view coincida com a borda superior da “view-pai”.

`android:layout_centerVertical`

Se “`true`”, centraliza esta “view-filha” verticalmente em seu “view-pai”.

`android:layout_below`

Posiciona a borda superior da view abaixo da view especificada com um ID de recurso.

Android: `layout_toRightOf`

Posiciona a margem esquerda da view à direita da view especificada com um ID de recurso.

Estes são apenas alguns exemplos. Todos os atributos de layout estão documentados em `RelativeLayout.LayoutParams`.

O valor para cada propriedade de layout é um booleano para habilitar uma posição de layout em relação ao parent `RelativeLayout` ou uma ID que faz referência a outra exibição no layout contra a qual a vista deve ser posicionada.

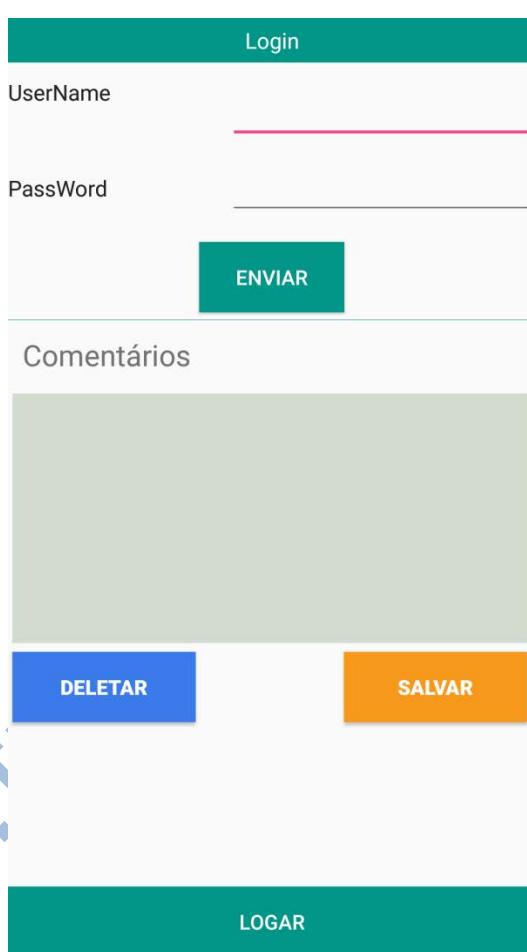
No layout XML do seu projeto, as dependências contra outras views no layout podem ser declaradas em qualquer ordem. Exemplo: você pode declarar que “view1” seja posicionado abaixo de “view2”, mesmo que “view2” seja a última view declarada na hierarquia. O exemplo abaixo demonstra esse cenário.

## Projeto Relative Layout

1	Você vai usar o Android Studio IDE para criar um aplicativo para Android e nomeá-lo como <code>RelativeLayoutApp</code> .
2	Modificar o conteúdo padrão da pasta <code>res /layout / activity_main.xml</code> para incluir botão e caixas de texto no <code>RelativeLayout</code> .

	3 Definir as constatens requeridas Login, UserName, PassWord, Enviar, Comentários, Deletar, Salvar em res /values/ strings.xml
4	Execute o aplicativo para iniciar o emulador Android e verificar o resultado das mudanças feitas na aplicação.

Abaixo está a captura de tela do resultado final:



Este é o trecho de código do conteúdo do arquivo de layout localizado na res /layout / values/string.xml. Implemente esse código em seu projeto:

```
<resources>
    <string name="app_name">RelativeLayoutApp</string>
    <string name="login">Login</string>
    <string name="username">UserName</string>
    <string name="password">PassWord</string>
    <string name="enviar">Enviar</string>
```

```

<string name="comentarios">Comentários</string>
<string name="deletar">Deletar</string>
<string name="salvar">Salvar</string>
<string name="logar">Logar</string>
</resources>

```

Este é o trecho de código do conteúdo do arquivo de layout localizado na res /layout / activity\_main.xml. Implemente esse código em seu projeto:

```

<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/text1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_centerInParent="false"
        android:background="#009688"
        android:gravity="center"
        android:padding="5dp"
        android:text="@string/login"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:textColor="#ffffff"
        android:textSize="15sp" />

    <EditText
        android:id="@+id/editText1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignLeft="@+id/editText2"
        android:layout_alignParentRight="true"
        android:layout_alignTop="@+id/textView1"
        android:ems="10" />

    <Button
        android:id="@+id/btnSubmit"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="false"
        android:layout_below="@+id/editText2"
        android:layout_centerInParent="true"
        android:layout_marginTop="15dp"
        android:background="#009688"
        android:paddingBottom="5dp"
        android:paddingLeft="25dp"
        android:paddingRight="25dp"
        android:paddingTop="5dp"
        android:text="@string/enviar"
        android:textColor="#ffffff" />

    <RelativeLayout
        android:id="@+id/cxcx"

```

```

        android:layout_width="match_parent"
        android:layout_height="0.1dp"
        android:layout_below="@+id/btnSubmit"
        android:layout_marginTop="5dp"
        android:background="#009688"
        android:orientation="horizontal"/>>

    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_below="@+id/cxcx">

        <TextView
            android:id="@+id/lblComments"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentLeft="true"
            android:layout_alignParentTop="true"
            android:padding="10dp"
            android:text="@string/comentarios"
            android:textSize="20dp"/>

        <EditText
            android:id="@+id/txtComments"
            android:layout_width="match_parent"
            android:layout_height="170dp"
            android:layout_alignLeft="@+id/lblComments"
            android:layout_below="@+id/lblComments"
            android:layout_centerHorizontal="true"
            android:layout_margin="3dp"
            android:background="#3362"
            android:textSize="18sp"/>

        <Button
            android:id="@+id/btnSave"
            android:layout_width="125dp"
            android:layout_height="wrap_content"
            android:layout_alignRight="@+id/txtComments"
            android:layout_below="@+id/txtComments"
            android:layout_margin="3dp"
            android:background="#f7991c"
            android:padding="5dp"
            android:text="@string/salvar"
            android:textColor="#fff"
            android:textStyle="bold"/>

        <Button
            android:id="@+id/button1"
            android:layout_width="125dp"
            android:layout_height="wrap_content"
            android:layout_alignBaseline="@+id/btnSave"
            android:layout_alignBottom="@+id/btnSave"
            android:layout_alignParentLeft="true"
            android:layout_margin="3dp"
            android:background="#3b7bea"
            android:padding="5dp"
            android:text="@string/deletar"
            android:textColor="#fff"
            android:textStyle="bold"/>
    </RelativeLayout>

```

```

<Button
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_alignParentBottom="true"
    android:layout_centerHorizontal="true"
    android:background="#009688"
    android:padding="8dp"
    android:text="@string/logar"
    android:textColor="#ffffff" />

<EditText
    android:id="@+id/editText2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignBaseline="@+id/textView2"
    android:layout_alignBottom="@+id/textView2"
    android:layout_alignParentRight="true"
    android:ems="10"
    android:inputType="textPassword">
</EditText>

<TextView
    android:id="@+id/textView2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentLeft="true"
    android:layout_below="@+id/editText1"
    android:layout_marginTop="20dp"
    android:text="@string/password"
    android:textAppearance="?android:attr/textAppearanceLarge"
    android:textSize="15sp" />

<TextView
    android:id="@+id/textView1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentLeft="true"
    android:layout_below="@+id/text1"
    android:layout_marginTop="10dp"
    android:text="@string/username"
    android:textAppearance="?android:attr/textAppearanceLarge"
    android:textSize="15sp" />

</RelativeLayout>

```

Este é o trecho de código do conteúdo do arquivo de layout localizado na main/java / nome.do.seu.pacote/MainActivity. Implemente esse código em seu projeto:

```

import android.os.Build
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.view.WindowInsets
import android.view.WindowManager

class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
    }
}

```

```
@SuppressLint("NewApi")
if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.R) {
    window.insetsController?.hide(WindowInsets.Type.statusBars())
} else {
    window.setFlags(
        WindowManager.LayoutParams.FLAG_FULLSCREEN,
        WindowManager.LayoutParams.FLAG_FULLSCREEN
    )
}
```

Rode seu projeto. Confira o resultado

## **ListView (Visualização em lista)**

A ListView é um grupo de exibições que exibe uma lista de itens roláveis. Os itens da lista são inseridos automaticamente na lista usando um Adapter que obtém conteúdo de uma origem como uma matriz ou consulta de banco de dados e converte cada resultado de item em uma exibição, que é colocada na lista.



## Visão Geral do Custom Adapter no do Android ListView

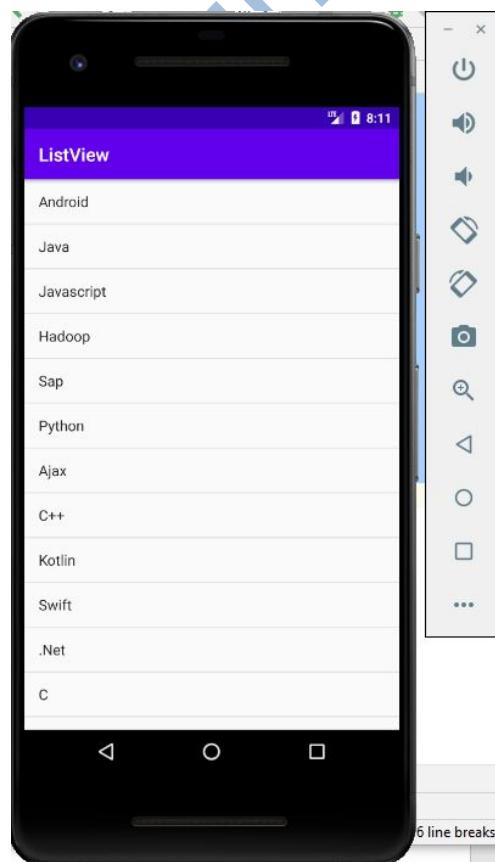
O adapter mais simples para preencher uma view a partir de um ArrayList é o ArrayAdapter. É isso que será implementado nesse exercício. Existem também outros adapters, como o CursorAdapter que se liga diretamente a um conjunto de resultados de um banco de dados SQLite local e usa um Cursor como fonte de dados.

## Projeto ListView

1	Você vai usar o Android Studio IDE para criar um aplicativo para Android e nomeá-lo como ListViewApp.
2	Modificar o conteúdo padrão da pasta <code>java /nome.do.seu.pacote.listviewapp/</code> para incluir os códigos que serão responsáveis pelo funcionamento do app.
3	Modificar o conteúdo do arquivo <code>xml</code> em <code>res/values/strings.xml</code>
4	Execute o aplicativo para iniciar o emulador Android e verifique o resultado das mudanças feitas na aplicação.

Neste projeto, será desenvolvido um aplicativo implementando o ListView Android , simples. Será criado um listview e, na sequência, será implementada uma nova activity para exibir o item – em tela única - que foi selecionado no listview.

Abaixo está a captura de tela do resultado final:



Agora, é necessário modificar o arquivo strings.xml para armazenar todos os labels dos itens da lista que será criada. Vá até a pasta values/strings.xml e abra o arquivo. Em seguida implemente o código abaixo:

```
<resources>
    <string name="app_name">ListView</string>
    <string-array name="technology_list">
        <item>Android</item>
        <item>Java</item>
        <item>Javascript</item>
        <item>Hadoop</item>
        <item>Sap</item>
        <item>Python</item>
        <item>Ajax</item>
        <item>C++</item>
        <item>Kotlin</item>
        <item>Swift</item>
        <item>.Net</item>
        <item>C</item>
    </string-array>
</resources>
```

Na sequência, abra o arquivo **MainActivity.kt** e implemente o código abaixo. No código a seguir, é importado todos os dados dos recursos xml e armazenando-os em um array.

```
import android.support.v7.app.AppCompatActivity
import android.os.Bundle
import android.widget.AdapterView
import android.widget.ArrayAdapter
import android.widget.ListView
import android.widget.Toast
//import kotlinx.android.synthetic.main.activity_main.*

class MainActivity : AppCompatActivity() {

    // val language =
    arrayOf<String>("C" , "C++" , "Java" , ".Net" , "Kotlin" , "Ruby" , "Rails" , "Python" , "Java Script" , "Php" , "Ajax" , "Perl" , "Hadoop" )

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        val language:Array<String> =
    resources.getStringArray(R.array.technology_list)
        val arrayAdapter =
    ArrayAdapter<String>(this , android.R.layout.simple_list_item
```

```
_1,language)
    val listView: ListView =
findViewById(R.id.listView)
    listView.setAdapter(arrayAdapter)

    listView.adapter = arrayAdapter
    listView.setOnItemClickListener =
AdapterView.OnItemClickListener { adapterView, view,
position, id ->
        val selectedItem =
adapterView.getItemAtPosition(position) as String
        val itemIdAtPos =
adapterView.getItemIdAtPosition(position)

        Toast.makeText(applicationContext,"click item
$selectedItem its position
$itemIdAtPos",Toast.LENGTH_SHORT).show()
    }
}
```

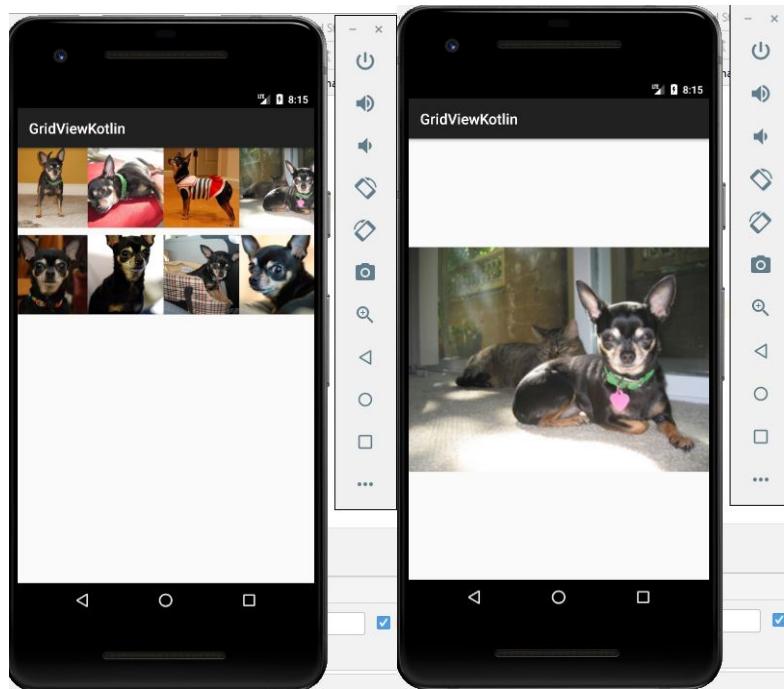
## GridView Layout

Layout do GridView é extremamente útil no Android. Principalmente quando os dados, imagens ou qualquer tipo de informação precisam ser exibidos em formato grade – estrutura de iconização. Este layout pode ser usado para criar aplicativos como visualizadores de imagens, leitores de áudio ou de vídeo.

## Projeto GridView:

- 1 Você vai usar o Android Studio IDE para criar um aplicativo para Android e nomeá-lo como GridViewApp.
- 2 Modificar o conteúdo padrão da pasta java /nome.do.seu.pacote.gridviewapp / criando duas novas classes ImageAdapter.kt e FullImage.kt para incluir os códigos que serão responsáveis pelo funcionamento do app.
- 3 Definir um novo arquivo xml em res /layout/ full\_image.xml
- 4 Execute o aplicativo para iniciar o emulador Android e verificar o resultado das mudanças feitas na aplicação.

Abaixo está a captura de tela do resultado final:



Seu arquivo `activity_main.xml` tem de se parecer com este abaixo.  
Implemente em seu projeto:

```
<?xml version="1.0" encoding="utf-8"?>
<GridView
    xmlns:android="http://schemas.android.com/apk/res/android"
        android:id="@+id/grid_view"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:numColumns="auto_fit"
        android:columnWidth="90dp"
        android:horizontalSpacing="10dp"
        android:verticalSpacing="10dp"
        android:gravity="center"
        android:stretchMode="columnWidth" >

</GridView>
```

Crie uma nova Classe clicando com o botão direito do mouse (clique direito) **src** ⇒ **pasta do pacote** ⇒ **New** ⇒ **Kotlin Class/File** e nomeie sua classe como `ImageAdapter.kt`

Sua classe ImageAdapter.java precisa se extender da classe BaseAdapter. O código para o ImageAdapter.kt é este:

```

import android.content.Context
import android.view.View
import android.view.ViewGroup
import android.widget.AbsListView
import android.widget.BaseAdapter
import android.widget.ImageView

class ImageAdapter      // Construtor
    (private val mContext: Context) : BaseAdapter() {
    // mantem todas as imagens dentro do array
    var mThumbIds = arrayOf<Int>(
        R.drawable.sample_0, R.drawable.sample_1,
        R.drawable.sample_2,
        R.drawable.sample_3, R.drawable.sample_4,
        R.drawable.sample_5, R.drawable.sample_6,
        R.drawable.sample_7
    )

    override fun getCount(): Int {
        return mThumbIds.size
    }

    override fun getItem(position: Int): Any {
        return mThumbIds[position]
    }

    override fun getItemId(position: Int): Long {
        return 0
    }

    override fun getView(position: Int, convertView: View?,
parent: ViewGroup): View {
        val imageView = ImageView(mContext)
        imageView.setImageResource(mThumbIds[position])
        imageView.scaleType = ImageView.ScaleType.CENTER_CROP
        imageView.setLayoutParams(AbsListView.LayoutParams(290,
290))
        return imageView
    }
}

```

Mostrando imagem selecionada do GridView em uma nova tela (Activity Tela cheia)

Até esse momento foram exibidas apenas imagens em GridView. É interessante quando podemos acrescentar funcionalidades como, por exemplo, mostrar a imagem selecionada (clicada) em uma nova tela cheia. Para isso, precisamos passar o ID do recurso de imagem do GridView para essa nova Activity.

Para isso, é necessário criar um novo arquivo XML na pasta de layout. Nomeie-o como full\_image.xml e implemente o seguinte código:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical" >

    <ImageView android:id="@+id/full_image_view"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent" />

</LinearLayout>
```

Agora, é necessário implementar uma nova classe java. Crie uma nova activity clicando com o botão direito do mouse em **src ⇒ pasta do pacote ⇒ New ⇒ Kotlin Class/File** e nomeie sua classe como FullImageActivity.kt e implemente o seguinte código:

```
import android.app.Activity
import android.os.Bundle
import android.view.View
import android.widget.ImageView

class FullImageActivity : Activity() {
    public override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.full_image)

        // get intent data
        val i = intent

        // Selected image id
        val position = i.extras!!.getInt("id")
        val imageAdapter = ImageAdapter(this)
```

```

        val imageView =
findViewById<View>(R.id.full_image_view) as ImageView

imageView.setImageResource(imageAdapter.mThumbIds.get(posit
ion))
    }
}

```

Sua MainActivity deve ser igual a essa. Implemente o código abaixo na sua MainActivity.kt:

```

import android.app.Activity
import android.content.Intent
import android.os.Bundle
import android.view.View
import android.widget.AdapterView.OnItemClickListener
import android.widget.GridView

class MainActivity : Activity() {
    public override fun onCreate(savedInstanceState:
Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        val gridView = findViewById<View>(R.id.grid_view)
as GridView

        // Instance of ImageAdapter Class
        gridView.adapter = ImageAdapter(this)
        /**
         * On Click event for Single Gridview Item
         */
        gridView.setOnItemClickListener = OnItemClickListener
        { parent, v, position, id -> // Sending image id to
FullScreenActivity
            val i = Intent(applicationContext,
FullImageActivity::class.java)
            // passing array index
            i.putExtra("id", position)
            startActivity(i)
        }
    }
}

```

Abra seu arquivo AndroidManifest.xml e adicione a nova activity implementada da seguinte forma. Este trecho de código refere-se a sua

aplicação. Isso significa que o cabeçalho de seu arquivo AndroidManifest.xml deve permanecer sem alteração:

```
<application
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name" >
    <activity
        android:label="@string/app_name"
        android:name=".MainActivity" >
        <intent-filter >
            <action android:name="android.intent.action.MAIN" />

            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
<!-- FullImageActivity -->
<activity android:name=".FullImageActivity"></activity>
</application>
```

Este é o código do arquivo AndroidManifest.xml deste exercício:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
    xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.gridviewkotlin">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.GridViewKotlin">
        <activity android:name=".MainActivity" >
            <intent-filter>
                <action
                    android:name="android.intent.action.MAIN" />

                <category
                    android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <!-- FullImageActivity -->
        <activity
            android:name=".FullImageActivity"></activity>
    </application>
</manifest>
```

Execute seu projeto e veja o resultado.

## RecyclerView

O widget `RecyclerView` é uma versão mais avançada e flexível do `ListView`. Esse widget é um contêiner para exibir grandes conjuntos de dados que podem ser rolados com muita eficiência ao manter um número limitado de visualizações. Use o widget `RecyclerView` quando tiver coletas de dados cujos elementos mudam durante a execução baseados na ação do usuário ou em eventos de rede.

A classe `RecyclerView` simplifica a exibição e o tratamento de grandes conjuntos de dados oferecendo:

- Gerenciadores de layout para posicionar itens
- Animações padrão para operações de item comuns, como remoção ou adição de itens

Você também tem a flexibilidade de definir gerenciadores de layout e animações personalizadas para widgets `RecyclerView`.



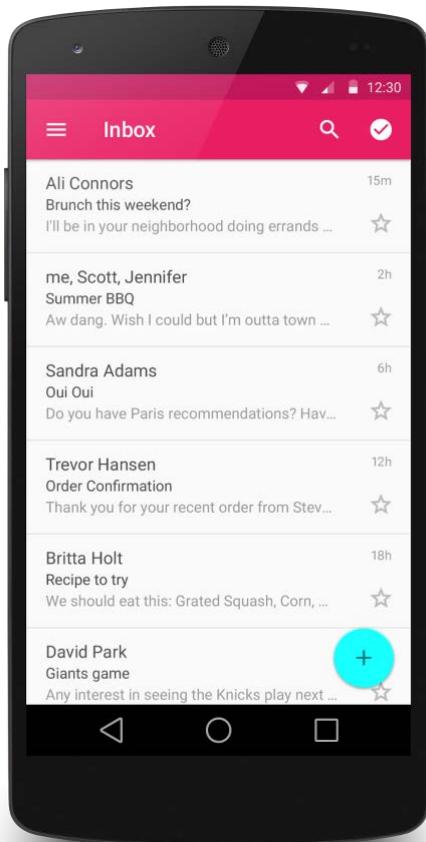
**Figura 1.** O widget `RecyclerView`.

Para usar o widget `RecyclerView`, você deve especificar um adaptador e um gerenciador de layout. Para criar um adaptador, amplie a classe `RecyclerView.Adapter`. Os detalhes da implementação dependem das especificações do conjunto de dados e do tipo de visualização.

Usando `RecyclerView`, listas e grades podem ser criadas muito facilmente.

Neste exercício, será implementado o widget `RecyclerView` simples com um layout que você pode personalizar. Também faremos uso de manipulação de uma classe de adaptadores(adapters).

Essa é a estrutura do widget `RecyclerView` com os atributos necessários:



Listas com `RecyclerView`.



Um **gerenciador de layout** posiciona as visualizações de item dentro de um `RecyclerView` e determina quando reutilizar visualizações de item que não estão mais visíveis ao usuário. Para reutilizar (ou *reciclar*) uma visualização, um gerenciador de layout pode solicitar ao adaptador a substituição do conteúdo da visualização com um elemento diferente do conjunto de dados. Visualizações recicladas dessa maneira aprimoram o desempenho ao evitar a criação de visualizações desnecessárias ou a realização de pesquisas `findViewById()` caras.

`RecyclerView` fornece esses gerenciadores de layout embutidos:

- `LinearLayoutManager` exibe itens em uma lista de rolagem vertical ou horizontal.
- `GridLayoutManager` exibe itens em uma grade.
- `StaggeredGridLayoutManager` exibe itens em uma grade escalonada.

Para criar um gerenciador de layout personalizado, amplie a classe `RecyclerView.LayoutManager`.

## Animações

As animações para a adição e remoção de itens são habilitadas, por padrão, em `RecyclerView`. Para personalizá-las, amplie a

classe `RecyclerView.ItemAnimator` e use o método `RecyclerView.setItemAnimator()`.

### Adicionar dependências

O widget `RecyclerView` é parte da Biblioteca de Suporte v7. Para usar esse widget no projeto, adicione estas dependências do Gradle ao módulo do aplicativo. Abra o arquivo Gradle do seu projeto localizado na pasta app: Nome do Projeto/app/build.gradle e adicione a dependência para o widget `RecyclerView`. Essa é a dependência a ser adicionada:

```
dependencies {
    //RecyclerView View
    implementation 'androidx.recyclerview:recyclerview:1.0.0'
}
```

### Projeto Navigation Drawer

Material Design foi introduzido na versão do Android Lollipop (versão 5.0). Em Material Design, muitas coisas novas foram introduzidas como Material Theme, novos widgets, sombras personalizadas – como mencionado acima, desenhos desenhados em vetor e animações personalizadas.

Neste projeto, serão implementadas as principais etapas do desenvolvimento do Material Design, ou seja, escrever o tema personalizado e implementar o Navigation Drawer usando o RecyclerView.

1	Você vai usar o Android Studio IDE para criar um aplicativo para Android e nomeá-lo como NavDrawerApp.
2	Modificar o conteúdo padrão da pasta java /nome.do.seu.pacote.recyclerviewapp / criando duas novas classes necessárias para incluir os códigos que serão responsáveis pelo funcionamento do app.
3	Definir e trabalhar com novos arquivo xml
4	Execute o aplicativo para iniciar o emulador Android e verificar o resultado das mudanças feitas na aplicação.

O Material Design fornece um conjunto de propriedades para personalizar o tema Material Design Color. Usamos cinco atributos principais para personalizar o tema geral que será usado para implementar o aplicativo.

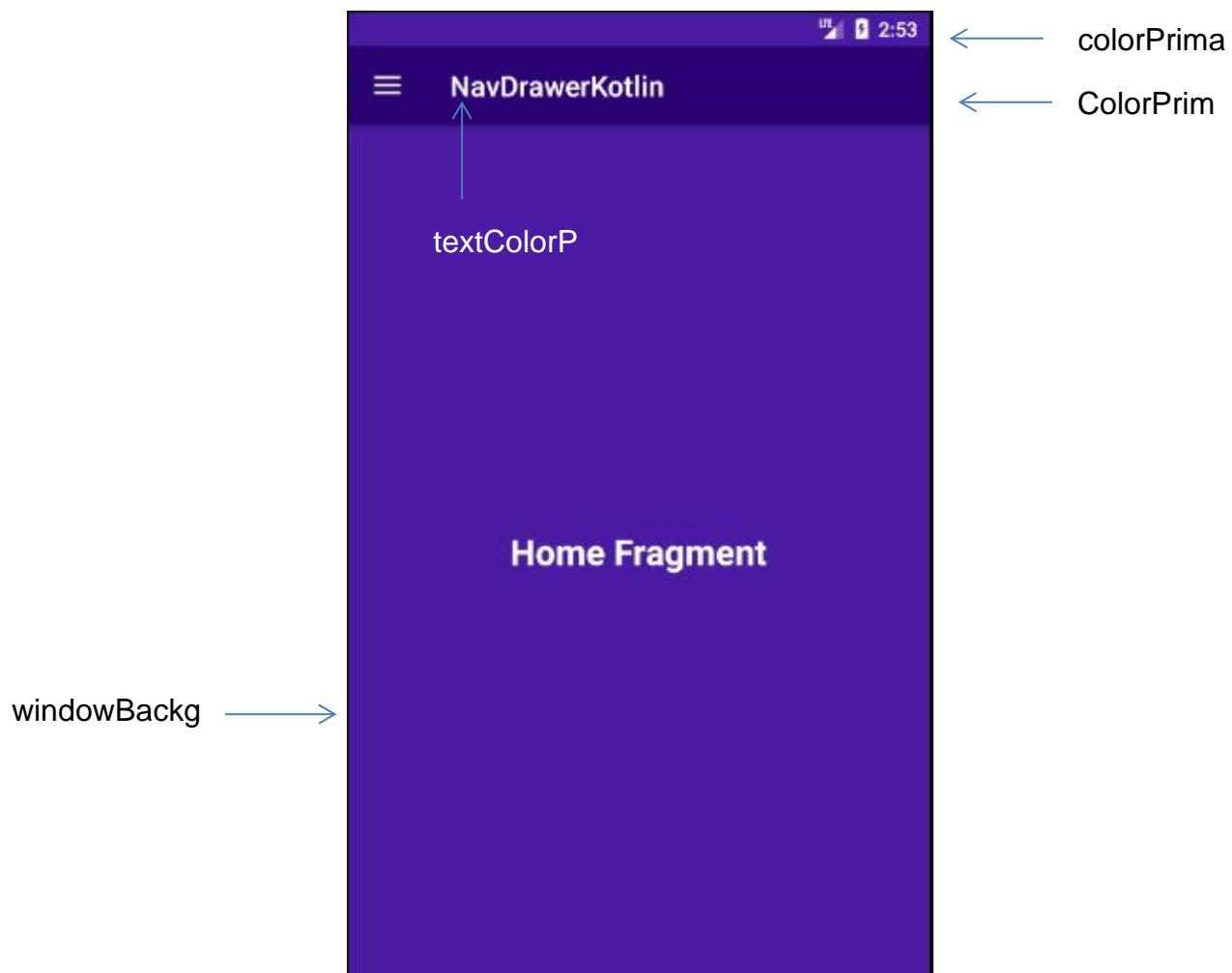
`colorPrimaryDark` - Esta é a cor primária mais escura (geralmente) do aplicativo aplica-se principalmente ao fundo da barra de notificação.

`ColorPrimary` - Esta é a cor principal do aplicativo. Essa cor será aplicada como fundo da barra de ferramentas.

`textColorPrimary` - Esta é a cor principal do texto. Isso se aplica ao título da barra de ferramentas.

`windowBackground` - Esta é a cor de fundo padrão do aplicativo.

`navigationBarColor` - Esta cor define a cor de fundo da barra de navegação do rodapé. (visível, na maioria das vezes, nos emuladores Android dentro do Android Studio)



Om o Android Studio aberto e seu projeto criado navegue até `res ⇒ values ⇒ strings.xml` e implemente o código abaixo:

### Adicionando dependências

Acesse `build.gradle` de nível de aplicativo e adicione as duas dependências a seguir:

```
dependencies {
    implementation "org.jetbrains.kotlin:kotlin-stdlib:$kotlin_version"
```

```

implementation 'androidx.core:core-ktx:1.3.2'
implementation 'androidx.appcompat:appcompat:1.2.0'
implementation
'com.google.android.material:material:1.2.1'
    implementation
'androidx.constraintlayout:constraintlayout:2.0.4'
        testImplementation 'junit:junit:4.+'
        androidTestImplementation
'androidx.test.ext:junit:1.1.2'
        androidTestImplementation
'androidx.test.espresso:espresso-core:3.3.0'

    implementation
'com.google.android.material:material:1.2.1'
    implementation
'androidx.recyclerview:recyclerview:1.1.0'
}

apply plugin: 'com.android.application'
apply plugin: 'kotlin-android'
apply plugin: 'kotlin-android-extensions'

```

## Criação do layout Navigation Drawer

Navegue até layout da MainActivity (*activity\_main.xml*) e adicione o DrawerLayout com um RecyclerView

```

<?xml version="1.0" encoding="utf-8"?>
<androidx.drawerlayout.widget.DrawerLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/drawer_layout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:openDrawer="start">

    <!-- Main Activity - Inicio -->
    <LinearLayout
        android:id="@+id/main_activity_content_id"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:fitsSystemWindows="true"
        android:orientation="vertical"
        tools:context=".MainActivity">

        <com.google.android.material.appbar.AppBarLayout
            android:id="@+id/activity_main_appbarlayout"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"

            android:theme="@style/ThemeOverlay.AppCompat.Dark.ActionBar">

            <androidx.appcompat.widget.Toolbar
                android:id="@+id/activity_main_toolbar"

```

```

        android:layout_width="match_parent"
        android:layout_height="?attr/actionBarSize"
        android:background="?attr/colorPrimary"
        android:weightSum="1"
        app:contentInsetStart="0dp"
        app:popupTheme="@style/ThemeOverlay.AppCompat.Light">

    <TextView
        android:id="@+id/activity_main_toolbar_title"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:ellipsize="marquee"
        android:fadingEdge="horizontal"
        android:focusable="true"
        android:focusableInTouchMode="true"
        android:marqueeRepeatLimit="marquee_forever"
        android:scrollHorizontally="true"
        android:singleLine="true"
        android:textColor="#ff4500"
        android:textSize="30sp" />

    </androidx.appcompat.widget.Toolbar>

</com.google.android.material.appbar.AppBarLayout>

<RelativeLayout
    android:id="@+id/activity_main_content_id"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/colorPrimaryDark"
    android:clipToPadding="false" />

</LinearLayout>
<!-- Main Activity Fim -->

<!-- Custom Navigation Drawer Inicio -->
<LinearLayout
    android:id="@+id/navigation_layout"
    android:layout_width="@dimen/navigation_drawer_width"
    android:layout_height="match_parent"
    android:layout_gravity="start"
    android:background="@color/colorPrimary"
    android:orientation="vertical">

    <ImageView
        android:id="@+id/navigation_header_img"
        android:layout_width="match_parent"
        android:layout_height="@dimen/nav_header_height"
        android:layout_marginStart="3dp"
        android:layout_marginLeft="3dp"
        android:layout_marginEnd="3dp"
        android:layout_marginRight="3dp"
        android:layout_marginBottom="5dp"
        android:padding="10dp"
        android:scaleType="fitCenter"
        android:src="@drawable/ic_batman"
        tools:ignore="ContentDescription" />

    <androidx.recyclerview.widget.RecyclerView

```

```

        android:id="@+id/navigation_rv"
        android:layout_width="@dimen/navigation_drawer_width"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:scrollbarHorizontal="@null"
        android:scrollbarVertical="@null">

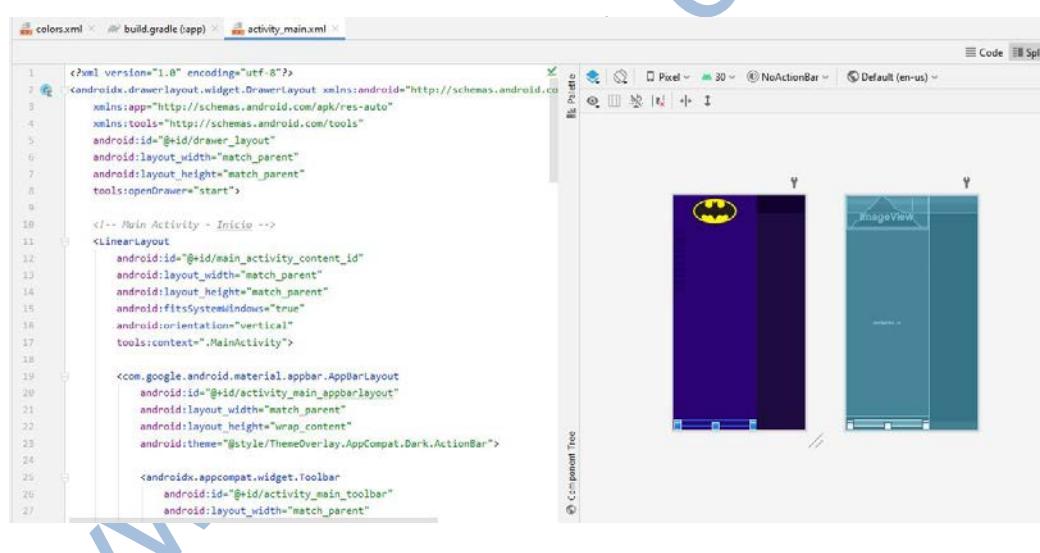
    </androidx.recyclerview.widget.RecyclerView>

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:layout_margin="12sp"
        android:textAlignment="center"
        android:textColor="@android:color/white"
        android:textStyle="bold" />

</LinearLayout>
<!-- Custom Navigation Drawer - Fim -->

</androidx.drawerlayout.widget.DrawerLayout>

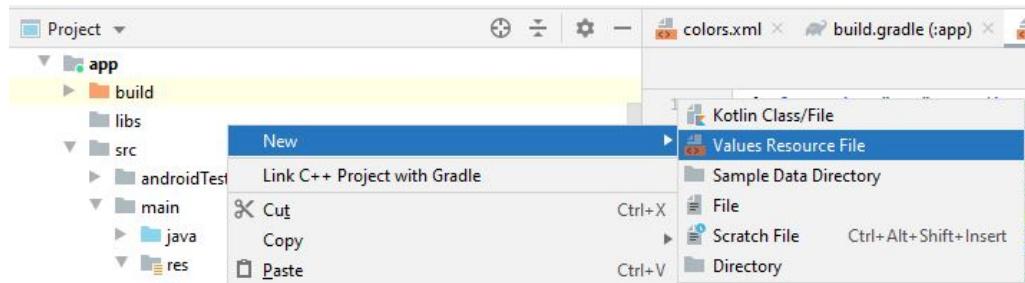
```



**A largura do drawer** muda de acordo com a largura da tela do dispositivo que o aplicativo está executando.

Para tablets, a **largura da drawer** é de **240 dp** e **260 dp** para telefones

Vá para **res> pressione clique com o botão direito na pasta de values> New> Values resource files**



Implemente o código abaixo:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <dimen name="nav_header_height">100dp</dimen>
        <!-- Per the design guidelines, navigation drawers should be
between 240dp and 320dp:
    https://developer.android.com/design/patterns/navigation-
drawer.html -->
    <dimen name="navigation_drawer_width">260dp</dimen>
</resources>
```



Crie um novo arquivo XML (**clique com o botão direito na pasta de layout > New > Layout Resource File**) para o layout de linha do RecyclerView (`row_nav_drawer.xml`)

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="44dp"
    android:orientation="horizontal"
    android:padding="3sp"
    android:paddingStart="10dp"
    android:paddingEnd="0dp"
    android:paddingLeft="10dp"
    android:paddingRight="0dp">

    <ImageView
        android:id="@+id/navigation_icon"
        android:layout_width="35sp"
        android:layout_height="35sp"
        android:layout_gravity="center_vertical"
        android:scaleType="centerInside"
        android:src="@android:drawable/btn_star_big_on"
        tools:ignore="ContentDescription" />

    <TextView
        android:id="@+id/navigation_title"
        android:layout_width="match_parent"
        android:layout_height="35sp"
        android:layout_gravity="center_vertical"
```

```

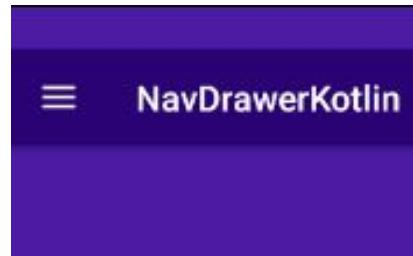
        android:layout_marginStart="7sp"
        android:gravity="center_vertical"
        android:textSize="18sp"
        android:layout_marginLeft="7sp" />

</LinearLayout>

```

Agora, temos que criar um novo tema.

Neste tema, definimos a cor do botão Drawer Toggle (ícone do menu de hambúrguer)



Vá para `res> values> styles.xml` e implemente o seguinte tema:

```

<resources>

    <!-- Base application theme. -->
    <style name="AppTheme"
parent="Theme.AppCompat.Light.DarkActionBar">
        <!-- Customize your theme here. -->
        <item name="colorPrimary">@color/colorPrimary</item>
        <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
        <item name="colorAccent">@color/colorAccent</item>
    </style>

    <style name="AppTheme.NoActionBar">
        <item name="windowActionBar">false</item>
        <item name="windowNoTitle">true</item>
        <item
            name="android:actionMenuTextColor">@android:color/white</item>
            <item name="drawerArrowStyle">@style/DrawerArrowStyle</item>
        </style>

        <style name="DrawerArrowStyle"
parent="@style/Widget.AppCompat.DrawerArrowToggle">
            <item name="spinBars">true</item>
            <item name="color">@android:color/white</item>
        </style>

    </resources>

```

Substitua o `theme`, no `AndroidManifest.xml`. Implemente o código abaixo:

```

<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.navdrawerkotlin">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme.NoActionBar">
        <activity android:name=".DemoActivity" />
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

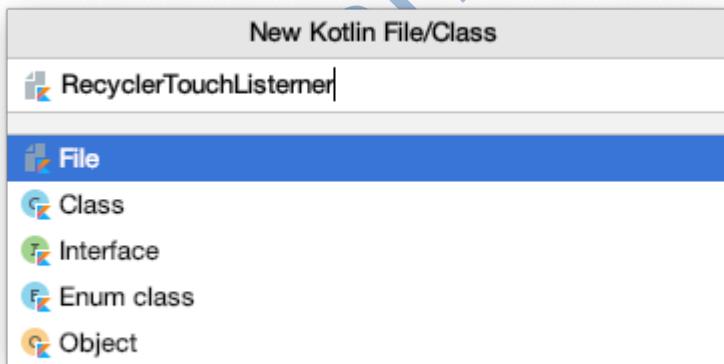
                <category
                    android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>

```

## Criando o Listener do RecyclerView

Crie um novo arquivo Kotlin com o nome RecyclerTouchListener



Vamos usar essa classe para detectar qual linha do RecyclerView está selecionada.

```

import android.content.Context
import android.view.GestureDetector
import android.view.MotionEvent
import android.view.View
import androidx.recyclerview.widget.RecyclerView

class RecyclerTouchListener internal constructor(
    context: Context,

```

```

    private val clickListener: ClickListener?
) : RecyclerView.OnItemTouchListener {
    private val gestureDetector: GestureDetector =
        GestureDetector(context, object :
        GestureDetector.SimpleOnGestureListener() {
            override fun onSingleTapUp(e: MotionEvent): Boolean {
                return true
            }
        })
    override fun onInterceptTouchEvent(rv: RecyclerView, e:
    MotionEvent): Boolean {
        val child = rv.findChildViewUnder(e.x, e.y)
        if (child != null && clickListener != null &&
        gestureDetector.onTouchEvent(e)) {
            clickListener.onClick(child,
            rv.getChildAdapterPosition(child))
        }
        return false
    }
    override fun onTouchEvent(rv: RecyclerView, e: MotionEvent) {
    }

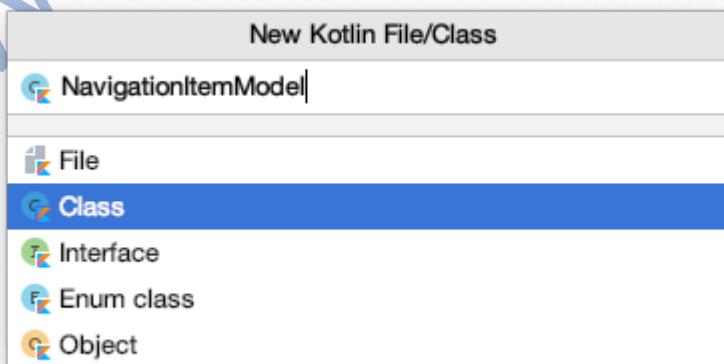
    override fun
    onRequestDisallowInterceptTouchEvent(disallowIntercept: Boolean) {
    }
}

internal interface ClickListener {
    fun onClick(view: View, position: Int)
}

```

## Criação de model NavDrawer

Crie um novo arquivo Kotlin , dê esse nome NavigationItemModel.kt



Implemente o código abaixo:

```
data class NavigationItemModel(var icon: Int, var title: String)
```

### Criação do adapter RecyclerView Drawer

Crie um novo arquivo Kotlin com um nome *NavigationRVAdapter.kt* e parâmetros uma ArrayList dos itens de menu e a posição atual do item pressionado:

- Declare o contexto no início da aula
- Crie o ViewHolder
- Defina o número de itens para o ReyclerView
- E passar os dados para as visualizações (ícone de navegação e título)

```
import android.content.Context
import android.graphics.Color
import android.graphics.PorterDuff
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.core.content.ContextCompat
import androidx.recyclerview.widget.RecyclerView
import kotlinx.android.synthetic.main.row_nav_drawer.view.*

class NavigationRVAdapter(private var items: ArrayList<NavigationItemModel>, private var currentPos: Int) : RecyclerView.Adapter<NavigationRVAdapter.NavigationItemViewHolder>() {

    private lateinit var context: Context

    class NavigationItemViewHolder(itemView: View) : RecyclerView.ViewHolder(itemView)

    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): NavigationItemViewHolder {
        context = parent.context
        val navItem =
            LayoutInflater.from(parent.context).inflate(R.layout.row_nav_drawer, parent, false)
        return NavigationItemViewHolder(navItem)
    }

    override fun getItemCount(): Int {
        return items.count()
    }

    override fun onBindViewHolder(holder: NavigationItemViewHolder, position: Int) {
        // To highlight the selected Item, show different
    }
}
```

```

background color
    if (position == currentPos) {

holder.itemView.setBackgroundColor(ContextCompat.getColor(context,
        R.color.colorPrimaryDark))
    } else {

holder.itemView.setBackgroundColor(ContextCompat.getColor(context,
        android.R.color.transparent))
    }

holder.itemView.navigation_icon.setColorFilter(Color.WHITE,
        PorterDuff.Mode.SRC_ATOP)

holder.itemView.navigation_title.setTextColor(Color.WHITE)
        //val font = ResourcesCompat.getFont(context,
        R.font.mycustomfont)
        //holder.itemView.navigation_text.typeface = font

//holder.itemView.navigation_text.setTextSize(TypedValue.COMPLEX_UNIT_SP,
        20.toFloat())

        holder.itemView.navigation_title.text =
        items[position].title

holder.itemView.navigation_icon.setImageResource(items[position].
        icon)
    }
}

```

## Configurando o drawer

No MainActivity.kt, declare o DrawerLayout , o adaptador do RecyclerView e crie um ArrayList dos itens de menu com o NavigationItemModel que fizemos anteriormente:

- Defina a barra de ferramentas , o layout da recycler e adicione o listener
- Se você notar, depois de selecionar um item do menu, fechamos o drawer após 200 milissegundos. Esse atraso ajuda a ter uma animação de fechamento suave se o fragment ou activity que você escolheu carregar muitos dados.
- Agora, adicione o item de menu padrão . Este item de menu será selecionado o primeiro que você verá quando o aplicativo iniciar; o item de menu padrão é o fragment 'Home'. Assim, destacamos o

item, que está na posição 0, e substituímos o layout do conteúdo da Activity pelo fragmento.

- Se o fragment ou activity usa o teclado, vamos adicionar o código para fechar o teclado ao abrir / fechar o drawer
- Na MainActivity, serão, também, adicionadas as seguintes linhas para definir uma imagem de header-background no drawer
- Por último, quando você pressiona o botão Voltar no dispositivo, verifique se o Drawer está aberto ou não, e o número de fragmentos em sua pilha

Implemente o código abaixo:

```
import android.content.Context
import android.content.Intent
import android.net.Uri
import android.os.Bundle
import android.os.Handler
import android.view.View
import android.view.inputmethod.InputMethodManager
import androidx.appcompat.app.ActionBarDrawerToggle
import androidx.appcompat.app.AppCompatActivity
import androidx.appcompat.widget.Toolbar
import androidx.core.content.ContextCompat
import androidx.core.view.GravityCompat
import androidx.drawerlayout.widget.DrawerLayout
import androidx.recyclerview.widget.LinearLayoutManager
import androidx.recyclerview.widget.RecyclerView
import kotlinx.android.synthetic.main.activity_main.*

class MainActivity : AppCompatActivity() {

    lateinit var drawerLayout: DrawerLayout
    private lateinit var adapter: NavigationRVAdapter

    private var items = arrayListOf(
        NavigationItemModel(R.drawable.ic_home, "Home"),
        NavigationItemModel(R.drawable.ic_music, "Música"),
        NavigationItemModel(R.drawable.ic_movie, "Filmes"),
        NavigationItemModel(R.drawable.ic_book, "Livros"),
        NavigationItemModel(R.drawable.ic_profile, "Perfil"),
        NavigationItemModel(R.drawable.ic_settings, "Configurações"),
    )

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        drawerLayout = findViewById(R.id.drawer_layout)

        // Set the toolbar
        setSupportActionBar(toolbar)
    }
}
```

```

// Setup Recyclerview's Layout
navigation_rv.layoutManager = LinearLayoutManager(this)
navigation_rv.setHasFixedSize(true)

// Add Item Touch Listener

navigation_rv.addOnItemTouchListener(RecyclerTouchListener(this,
object : ClickListener {
    @SuppressLint("DEPRECATION")
    override fun onClick(view: View, position: Int) {
        when (position) {
            0 -> {
                // # Home Fragment
                val bundle = Bundle()
                bundle.putString("fragmentName", "Home
Fragment")
                val homeFragment = DemoFragment()
                homeFragment.arguments = bundle
                supportFragmentManager.beginTransaction()
                    .replace(R.id.activity_main_content_id,
homeFragment).commit()
            }
            1 -> {
                // # Musica Fragment
                val bundle = Bundle()
                bundle.putString("fragmentName", "Musica
Fragment")
                val musicFragment = DemoFragment()
                musicFragment.arguments = bundle
                supportFragmentManager.beginTransaction()
                    .replace(R.id.activity_main_content_id,
musicFragment).commit()
            }
            2 -> {
                // # Filmes Fragment
                val bundle = Bundle()
                bundle.putString("fragmentName", "Filmes
Fragment")
                val moviesFragment = DemoFragment()
                moviesFragment.arguments = bundle
                supportFragmentManager.beginTransaction()
                    .replace(R.id.activity_main_content_id,
moviesFragment).commit()
            }
            3 -> {
                // # Livros Fragment
                val bundle = Bundle()
                bundle.putString("fragmentName", "Livros
Fragment")
                val booksFragment = DemoFragment()
                booksFragment.arguments = bundle
                supportFragmentManager.beginTransaction()
                    .replace(R.id.activity_main_content_id,
booksFragment).commit()
            }
            4 -> {
                // # Perfil Activity
                val intent = Intent(this@MainActivity,
DemoActivity::class.java)
                intent.putExtra("activityName", "Perfil

```

```

Activity")
    startActivity(intent)
}
5 -> {
    // # Configurações Fragment
    val bundle = Bundle()
    bundle.putString("fragmentName",
"Configurações Fragment")
    val settingsFragment = DemoFragment()
    settingsFragment.arguments = bundle
    supportFragmentManager.beginTransaction()
        .replace(R.id.activity_main_content_id,
settingsFragment).commit()
}

}
// Don't highlight the 'Profile' and 'Like us on
Facebook' item row
if (position != 5 && position != 4) {
    updateAdapter(position)
}
Handler().postDelayed({
    drawerLayout.closeDrawer(GravityCompat.START)
}, 200)
})
})

// Update Adapter with item data and highlight the default
menu item ('Home' Fragment)
updateAdapter(0)

// Set 'Home' as the default fragment when the app starts
val bundle = Bundle()
bundle.putString("fragmentName", "Home Fragment")
val homeFragment = DemoFragment()
homeFragment.arguments = bundle
supportFragmentManager.beginTransaction()
    .replace(R.id.activity_main_content_id,
homeFragment).commit()

// Close the soft keyboard when you open or close the Drawer
val toggle: ActionBarDrawerToggle = object :
ActionBarDrawerToggle(this, drawerLayout, activity_main_toolbar,
R.string.navigation_drawer_open, R.string.navigation_drawer_close) {
    override fun onDrawerClosed(drawerView: View) {
        // Triggered once the drawer closes
        super.onDrawerClosed(drawerView)
        try {
            val inputMethodManager =
                getSystemService(Context.INPUT_METHOD_SERVICE)
        as InputMethodManager
        inputMethodManager.hideSoftInputFromWindow(currentFocus?.windowToken,
0)
            } catch (e: Exception) {
                e.stackTrace
            }
    }
    override fun onDrawerOpened(drawerView: View) {
}
}

```

```

        // Triggered once the drawer opens
        super.onDrawerOpened(drawerView)
        try {
            val inputMethodManager =
                getSystemService(Context.INPUT_METHOD_SERVICE)
            as InputMethodManager

            inputMethodManager.hideSoftInputFromWindow(currentFocus!!.windowToken,
                0)
            } catch (e: Exception) {
                e.stackTrace
            }
        }
    }
    drawerLayout.addDrawerListener(toggle)

    toggle.syncState()

    // Set Header Image
    navigation_header_img.setImageResource(R.drawable.ic_batman)

    // Set background of Drawer

    navigation_layout.setBackgroundColor(ContextCompat.getColor(this,
        R.color.colorPrimary))
}

private fun updateAdapter(highlightItemPos: Int) {
    adapter = NavigationRVAdapter(items, highlightItemPos)
    navigation_rv.adapter = adapter
    adapter.notifyDataSetChanged()
}

override fun onBackPressed() {
    if (drawerLayout.isDrawerOpen(GravityCompat.START)) {
        drawerLayout.closeDrawer(GravityCompat.START)
    } else {
        // Checking for fragment count on back stack
        if (supportFragmentManager.backStackEntryCount > 0) {
            // Go to the previous fragment
            supportFragmentManager.popBackStack()
        } else {
            // Exit the app
            super.onBackPressed()
        }
    }
}
}

```

Crie um novo arquivo XML (**clique com o botão direito na pasta de layout > New > Layout Resource File**) e nomeie como **fragment\_demo.xml**. Implemente código abaixo:

```

<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/colorPrimaryDark"

```

```

    android:gravity="center"
    tools:context=".DemoFragment">

    <TextView
        android:id="@+id/fragment_name"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Fragment Name"
        android:textColor="@android:color/white"
        android:textSize="25sp"
        android:textStyle="bold" />

</RelativeLayout>

```

Crie, novamente, um novo arquivo XML (**clique com o botão direito na pasta de layout> New> Layout Resource File**) e nomeie como **activity\_demo.xml**. Implemente código abaixo:

```

<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/colorPrimaryDark"
    android:gravity="center"
    tools:context=".DemoActivity">

    <TextView
        android:id="@+id/activity_name"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Activity Name"
        android:textColor="@android:color/white"
        android:textSize="25sp"
        android:textStyle="bold" />

</RelativeLayout>

```

Agora, crie uma nova classe Kotlin – dentro do pacote `com.example.navdrawerkotlin` – nomeie-o como `DemoFragment.kt`.

Implemente o código abaixo:

```

import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.fragment.app.Fragment
import kotlinx.android.synthetic.main.main_fragment_demo.view.*

```

```

class DemoFragment : Fragment() {

```

```

override fun onCreateView(
    inflater: LayoutInflater,
    container: ViewGroup?,
    savedInstanceState: Bundle?
): View? {
    val rootView = inflater.inflate(R.layout.fragment_demo,
    container, false)

    val fragmentName = arguments?.getString("fragmentName")

    rootView.fragment_name.text = fragmentName

    return rootView
}
}

```

Neste passo, crie uma nova classe Kotlin - dentro do pacote com.example.navdrawerkotlin – nomeie-o como DemoActivity.kt.

Implemente o código abaixo:

```

import android.os.Bundle
import androidx.appcompat.app.AppCompatActivity
import kotlinx.android.synthetic.main.activity_demo.*

class DemoActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_demo)

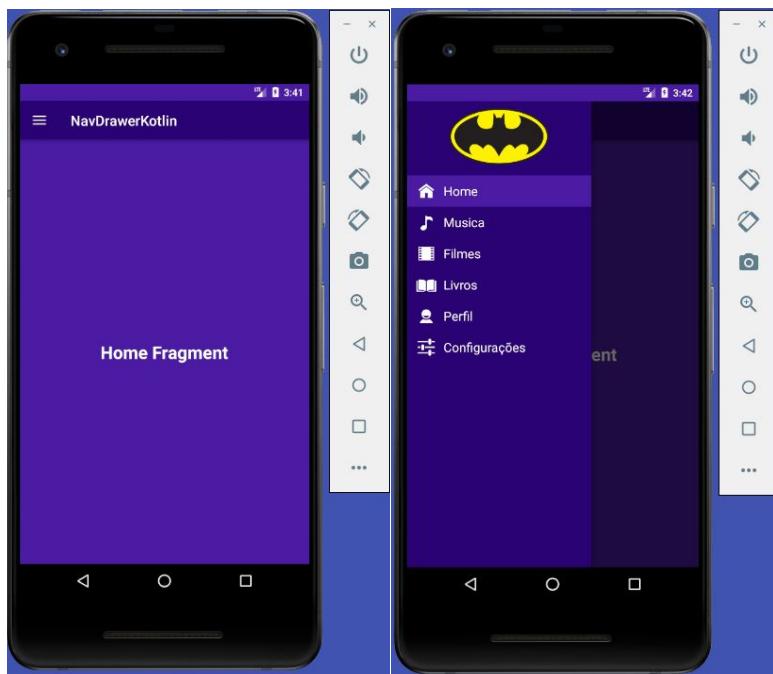
        val activityName = intent.getStringExtra("activityName")

        activity_name.text = activityName
    }
}

```

Execute seu projeto. Observe o resultado:





## TabView Layout

O modo de navegação tipo tabulação é um padrão de design muito comum entre as aplicações Android. Desde a versão 5.0 do Android, o Material Design tornou-se o padrão visual do Android – como mencionado, anteriormente – e todas as outras plataformas da Google. Com isso, inúmeras mudanças ocorreram. Várias APIs do sistema operacional Android sofreram atualizações. O resultado de uma dessas atualizações foi a depreciação da ActionBar.

Uma nova API Android, a Toolbar – como visto no exercício anterior - do Android foi lançada para substituí-la. Devido a essa mudança, as novas APIs para guias do Android também foram lançadas recentemente através da biblioteca de suporte de design(support library).

A classe principal usada para exibir Tabs através dessas novas APIs é o Android TabLayout. Neste exercício, será criado uma tela com três tabs usando essa nova API com Fragments e um ViewPager.

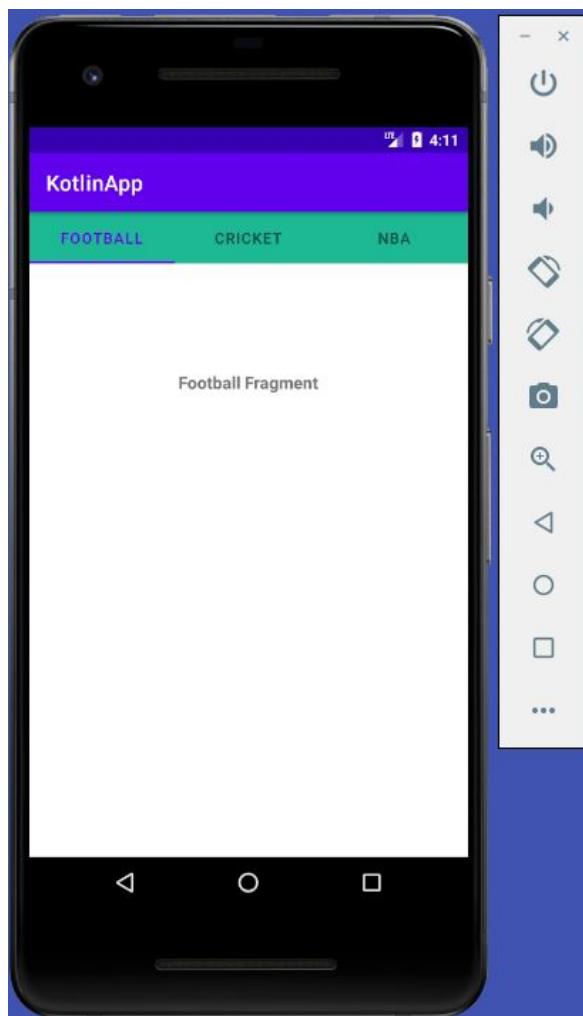
### Projeto Tab Layout

1

Você vai usar o Android Studio IDE para criar um aplicativo para Android e nomeá-lo como TabLayoutApp.

	Modificar o conteúdo padrão da pasta java /nome.do.seu.pacote.tabviewapp / criando novas classes necessárias para incluir os códigos que serão responsáveis pelo funcionamento do app.
3	Definir e trabalhar com novos arquivo xml
4	Execute o aplicativo para iniciar o emulador Android e verificar o resultado das mudanças feitas na aplicação.

Abaixo está a captura de tela do resultado final:



Após a criação do projeto, com o Android Studio aberto Abra seu arquivo build.gradle; seu código deve ser semelhante a este. Faça as alterações necessárias:

```
dependencies {
    implementation "org.jetbrains.kotlin:kotlin-stdlib:$kotlin_version"
    implementation 'androidx.core:core-ktx:1.3.2'
    implementation 'androidx.appcompat:appcompat:1.2.0'
    implementation 'com.google.android.material:material:1.2.1'
```

```

implementation 'androidx.constraintlayout:constraintlayout:2.0.4'
testImplementation 'junit:junit:4.+'
androidTestImplementation 'androidx.test.ext:junit:1.1.2'
androidTestImplementation 'androidx.test.espresso:espresso-
core:3.3.0'
}

```

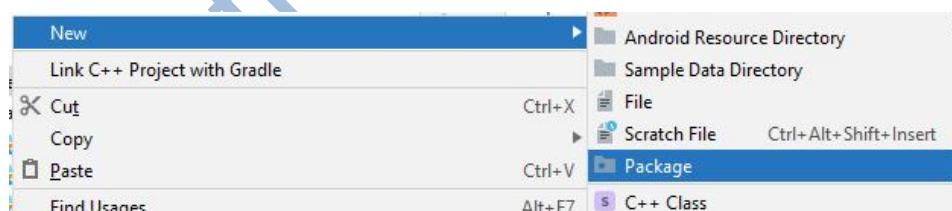
Abra o arquivo res/values/themes.xml e implemente o código abaixo:

```

<resources xmlns:tools="http://schemas.android.com/tools">
    <!-- Base application theme. -->
    <style name="Theme.TabLayoutKotlin"
parent="Theme.MaterialComponents.DayNight.DarkActionBar">
        <!-- Primary brand color. -->
        <item name="colorPrimary">@color/purple_500</item>
        <item name="colorPrimaryVariant">@color/purple_700</item>
        <item name="colorOnPrimary">@color/white</item>
        <!-- Secondary brand color. -->
        <item name="colorSecondary">@color/teal_200</item>
        <item name="colorSecondaryVariant">@color/teal_700</item>
        <item name="colorOnSecondary">@color/black</item>
        <!-- Status bar color. -->
        <item name="android:statusBarColor"
tools:targetApi="l">?attr/colorPrimaryVariant</item>
        <!-- Customize your theme here. -->
    </style>
</resources>

```

Na sequencia, vamos cirar as Tabs do Android com fragments e ViewPager. Defina três fragments simples e seus respectivos arquivos kotlin. Dentro do pacote com.example.tablayoutkotlin crie um novo package e nomeie-o como **fragments**.



*fragment\_basket.xml*

```

<?xml version="1.0" encoding="utf-8"?>
<FrameLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".fragments.Basket">
    <!-- TODO: Update blank fragment layout -->
    <TextView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:text="Basket Ball"

```

```

        android:textAlignment="center"
        android:textSize="16sp"
        android:textStyle="bold"
        android:gravity="center_horizontal" />
</FrameLayout>
```

Dentro do novo package criado – **fragments** – crie o arquivo **Basket.kt**

### **Basket.kt**

```

import android.os.Bundle
import androidx.fragment.app.Fragment
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import com.example.tablayoutkotlin.R

class Basket : Fragment() {
    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        // Inflate the layout for this fragment
        return inflater.inflate(R.layout.fragment_basket, container,
false)
    }
}
```

### **fragment\_football.xml**

```

<?xml version="1.0" encoding="utf-8"?>
<FrameLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".fragments.Football">
    <!-- TODO: Update blank fragment layout -->
    <TextView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:text="Football Fragment"
        android:textAlignment="center"
        android:textSize="16sp"
        android:textStyle="bold"
        android:gravity="center_horizontal" />
</FrameLayout>
```

Dentro do novo package criado – **fragments** – crie o arquivo **Football.kt**

### **Football.kt**

```

import android.os.Bundle
import android.view.LayoutInflater
```

```

import android.view.View
import android.view.ViewGroup
import androidx.fragment.app.Fragment
import com.example.tablayoutkotlin.R

class Football : Fragment() {
    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        // Inflate the layout for this fragment
        return inflater.inflate(R.layout.fragment_football, container,
false)
    }
}

```

### **fragment\_soccer.xml**

```

<?xml version="1.0" encoding="utf-8"?>
<FrameLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".fragments.Soccer">
    <!-- TODO: Update blank fragment layout -->
    <TextView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:text="Soccer Fragment"
        android:textAlignment="center"
        android:textSize="16sp"
        android:textStyle="bold"
        android:gravity="center_horizontal" />
</FrameLayout>

```

Dentro do novo package criado – **fragments** – crie o arquivo **Soccer.kt**

### **Soccer.kt**

```

import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.fragment.app.Fragment
import com.example.tablayoutkotlin.R

class Soccer : Fragment() {
    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        // Inflate the layout for this fragment
        return inflater.inflate(R.layout.fragment_soccer, container,
false)
    }
}

```

```

    }
}
```

Agora o projeto possui todos os fragments de Tabs definidos; é necessário definir um adapter de page view para o recurso de swipe tab. Navegue até a pasta `java/ nome.do.seu.pacote/` siga os passos já realizados em exercícios anteriores para criar uma classe e nomeie-a como ***MyAdapter.kt***. Quando criar a classe, implemente o código abaixo:

```

import android.content.Context
import androidx.fragment.app.Fragment
import androidx.fragment.app.FragmentManager
import androidx.fragment.app.FragmentPagerAdapter
import com.example.tablayoutkotlin.fragments.Soccer
import com.example.tablayoutkotlin.fragments.Football
import com.example.tablayoutkotlin.fragments.Basket

@SuppressLint("NewApi")
internal class MyAdapter(
    var context: Context,
    fm: FragmentManager,
    var totalTabs: Int
) : FragmentPagerAdapter(fm) {
    override fun getItem(position: Int): Fragment {
        return when (position) {
            0 -> {
                Football()
            }
            1 -> {
                Soccer()
            }
            2 -> {
                Basket()
            }
            else -> getItem(position)
        }
    }
    override fun getCount(): Int {
        return totalTabs
    }
}
```

 Na classe ***MyAdapter***, inicializa-se os fragments conforme sua localização. Em seguida, é necessário definir o layout para a atividade principal, onde todas essas tabs serão exibidas. Implemente o código abaixo na sua ***activity\_main.xml***:

```

<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
    <com.google.android.material.tabs.TabLayout
```

```

        android:id="@+id/tabLayout"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#1db995">
    </com.google.android.material.tabs.TabLayout>
<androidx.viewpager.widget.ViewPager
    android:id="@+id/viewPager"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@id/tabLayout"
    android:layout_centerInParent="true"
    android:layout_marginTop="100dp"
    tools:layout_editor_absoluteX="8dp" />
</RelativeLayout>

```

As tabs anteriores foram adicionadas à action bar através do código. Mas, como é possível perceber no exercício, os separadores do Android estão acima, na tela; usamos a Toolbar e o TabLayout separadamente para exibir guias. Além disso, note que também é adicionado um ViewPager, que é anexado a este TabLayout na activity. Abra seu arquivo **MainActivy.kt** e implemente o código abaixo:

```

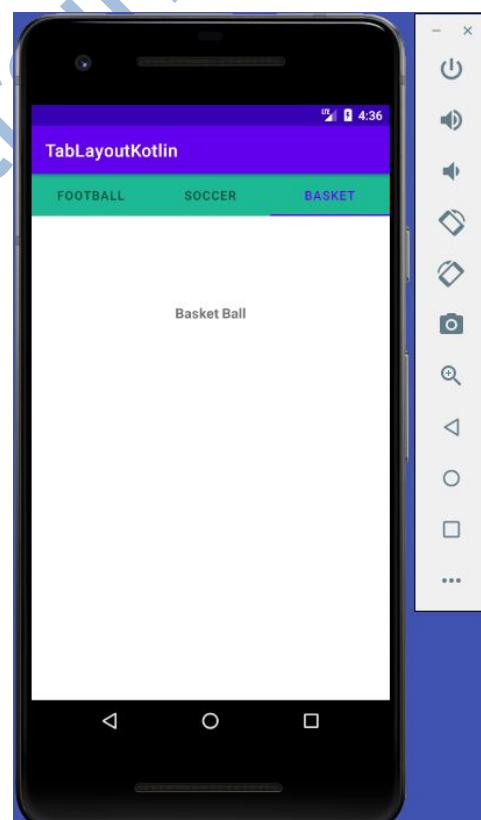
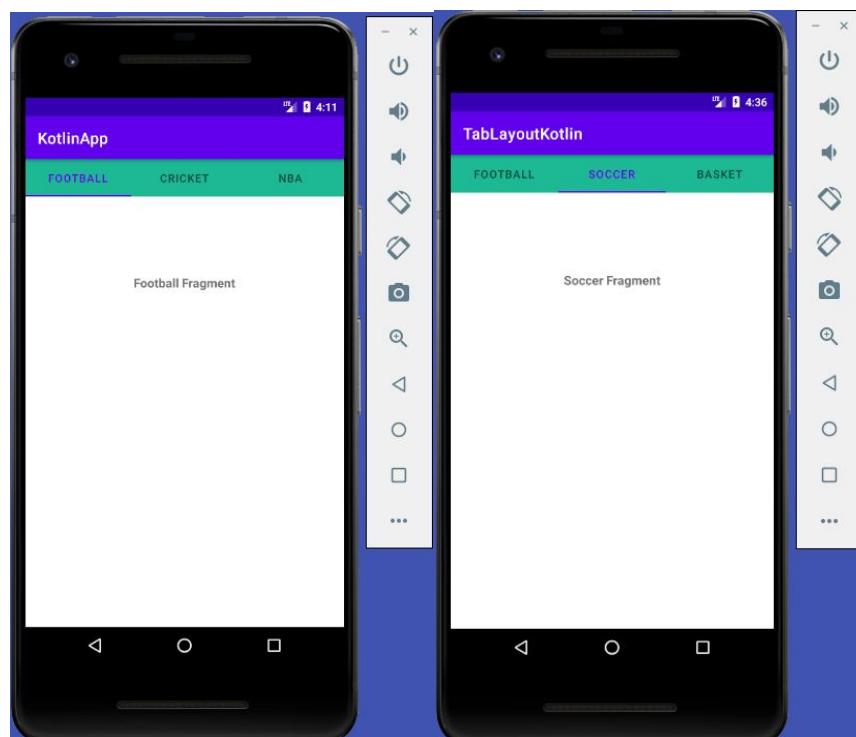
import android.os.Bundle
import androidx.appcompat.app.AppCompatActivity
import androidx.viewpager.widget.ViewPager
import com.google.android.material.tabs.TabLayout
import
com.google.android.material.tabs.TabLayout.OnTabSelectedListener
import
com.google.android.material.tabs.TabLayoutOnPageChangeListener
class MainActivity : AppCompatActivity() {
    lateinit var tabLayout: TabLayout
    lateinit var viewPager: ViewPager
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        title = "TabLayoutKotlin"
        tabLayout = findViewById(R.id.tabLayout)
        viewPager = findViewById(R.id.viewPager)
        tabLayout.addTab(tabLayout.newTab().setText("Football"))
        tabLayout.addTab(tabLayout.newTab().setText("Soccer"))
        tabLayout.addTab(tabLayout.newTab().setText("Basket"))
        tabLayout.tabGravity = TabLayout.GRAVITY_FILL
        val adapter = MyAdapter(this, supportFragmentManager,
            tabLayout.tabCount)
        viewPager.adapter = adapter

        viewPager.addOnPageChangeListener(TabLayoutOnPageChangeListener(tabLayout))
        tabLayout.addOnTabSelectedListener(object :
            OnTabSelectedListener {
            override fun onTabSelected(tab: TabLayout.Tab) {
                viewPager.currentItem = tab.position
            }
        })
    }
}

```

```
        }  
        override fun onTabUnselected(tab: TabLayout.Tab) {}  
        override fun onTabReselected(tab: TabLayout.Tab) {}  
    })  
}
```

Rode seu aplicativo. Confira o resultado:



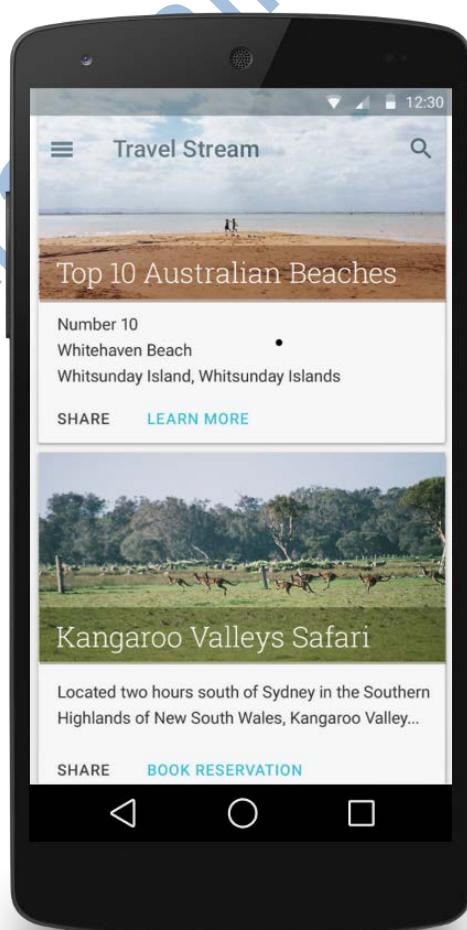
## CardView

CardView amplia a classe FrameLayout e permite que você exiba informações dentro dos cartões, que têm uma aparência coerente nas diversas plataformas. Os widgets CardView podem ter sombras e bordas arredondadas.

Para criar um cartão com sombra, use o atributo `card_view:cardElevation`. CardView usa elevação real e sombras dinâmicas no Android 5.0 (API de nível 21) e posteriores e volta para uma implementação de sombra programática em versões anteriores.

Use estas propriedades para personalizar a aparência do widget CardView:

- Para definir o raio do canto nos layouts, use o atributo `card_view:cardCornerRadius`.
- Para definir o raio do canto no seu código, use o método `CardView.setRadius`.
- Para definir a cor de segundo plano de um cartão, use o atributo `card_view:cardBackgroundColor`.



### CardView Overlapping Corners:

Java: `setPreventCornerOverlap (boolean)`

XML: `cardView: cardPreventCornerOverlap`

Propriedade única do Android CardView; seus cantos são arredondados.

A parte interessante sobre esta propriedade é essa, ela limpa os cantos do conteúdo automaticamente. Infelizmente, esta propriedade, a partir de agora, não é suportada em dispositivos com API 20 ou baixos. Em vez disso, quando uma imagem que cobre o CardView completo é exibida nesses dispositivos, um preenchimento é adicionado à imagem para evitar os cantos sobrepostos.

### CardView Content Padding:

Java: `setContentPadding (int, int, int, int)`

XML: `cardView: contentPadding`

Quando um card é exibido na tela, geralmente várias visualizações são adicionadas dentro dele. Felizmente, o Android CardView suporta esta propriedade única, através da qual o preenchimento do conteúdo CardView pode ser definido de uma só vez. A vantagem desta propriedade é que não é necessário definir margem ou preenchimento para cada visualização dentro do CardView individualmente para que ela fique adequada.

### CardView Elevation:

Java: `setCardElevation (float)`

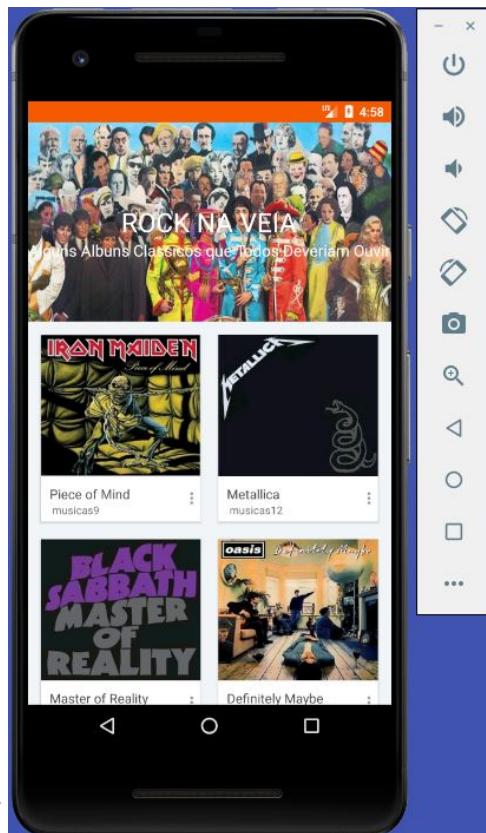
XML: `cardView: cardElevation`

## Projeto Card Layout

1	Você vai usar o Android Studio IDE para criar um aplicativo para Android e nomeá-lo como CardLayoutApp.
2	Modificar o conteúdo padrão da pasta <code>java /nome.do.seu.pacote.cardviewapp</code> / criando novas classes necessárias para incluir os códigos que serão responsáveis pelo funcionamento do app.

3	Definir e trabalhar com novos arquivos xml
4	Execute o aplicativo para iniciar o emulador Android e verificar o resultado das mudanças feitas na aplicação.

Abaixo está a captura de tela do resultado final:



Neste projeto, será usado o RecyclerView para mostrar uma lista de CardViews.

Após a criação do projeto, com o Android Studio aberto Abra seu arquivo build.gradle; seu código deve ser semelhante a este. Faça as alterações necessárias:

```
dependencies {
    implementation "org.jetbrains.kotlin:kotlin-stdlib:$kotlin_version"
    implementation 'androidx.core:core-ktx:1.3.2'
    implementation 'androidx.appcompat:appcompat:1.2.0'
    implementation 'com.google.android.material:material:1.2.1'
    implementation 'androidx.constraintlayout:constraintlayout:2.0.4'
    testImplementation 'junit:junit:4.+'
    androidTestImplementation 'androidx.test.ext:junit:1.1.2'
    androidTestImplementation 'androidx.test.espresso:espresso-
```

```

core:3.3.0'

//dependencias

//cardview
implementation 'androidx.cardview:cardview:1.0.0'
//recyclerview
implementation 'androidx.recyclerview:recyclerview:1.0.0'

// Glide
implementation 'com.github.bumptech.glide:glide:3.7.0'
}

```

Na sequencia, dentro do arquivo **activity\_main.xml**, implemente o código abaixo:

```

<?xml version="1.0" encoding="utf-8"?>
<androidx.coordinatorlayout.widget.CoordinatorLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:id="@+id/main_content"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@android:color/white"
    android:fitsSystemWindows="true">

    <com.google.android.material.appbar.AppBarLayout
        android:id="@+id/appbar"
        android:layout_width="match_parent"
        android:layout_height="@dimen/detail_backdrop_height"
        android:fitsSystemWindows="true"
        android:theme="@style/ThemeOverlay.AppCompat.Dark.ActionBar">

        <com.google.android.material.appbar.CollapsingToolbarLayout
            android:id="@+id/collapsing_toolbar"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:fitsSystemWindows="true"
            app:contentScrim="?attr/colorPrimary"
            app:expandedTitleMarginEnd="64dp"
            app:expandedTitleMarginStart="48dp"

            app:expandedTitleTextAppearance="@android:color/transparent"
            app:layout_scrollFlags="scroll|exitUntilCollapsed">

            <RelativeLayout
                android:layout_width="wrap_content"
                android:layout_height="wrap_content">

                <ImageView
                    android:id="@+id/backdrop"
                    android:layout_width="match_parent"
                    android:layout_height="match_parent"
                    android:fitsSystemWindows="true"
                    android:scaleType="centerCrop"
                    app:layout_collapseMode="parallax" />

                <LinearLayout

```

```

        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true"
        android:gravity="center_horizontal"
        android:orientation="vertical">

        <TextView
            android:id="@+id/love_music"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/backdrop_title"
            android:textColor="@android:color/white"
            android:textSize="@dimen/backdrop_title" />

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/backdrop_subtitle"
            android:textColor="@android:color/white"
            android:textSize="@dimen/backdrop_subtitle" />

    </LinearLayout>
</RelativeLayout>

<androidx.appcompat.widget.Toolbar
    android:id="@+id/toolbar"
    android:layout_width="match_parent"
    android:layout_height="?attr/actionBarSize"
    app:layout_collapseMode="pin"
    app:popupTheme="@style/ThemeOverlay.AppCompat.Light"
/>

</com.google.android.material.appbar.CollapsingToolbarLayout>

</com.google.android.material.appbar.AppBarLayout>

<include layout="@layout/content_main" />

</androidx.coordinatorlayout.widget.CoordinatorLayout>

```

Navegue até a pasta ***java/nome.do.seu.pacote/res/layout*** siga os passos já realizados em exercícios anteriores para criar uma arquivo e nomeie-a como ***álbum\_card.xml***. Quando criar esse arquivo de layout, implemente o código abaixo:

```

<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:card_view="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

    <androidx.cardview.widget.CardView
        android:id="@+id/card_view"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_gravity="center">

```

```

        android:layout_margin="@dimen/card_margin"
        android:elevation="3dp"
        card_view:cardCornerRadius="@dimen/card_album_radius">

    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <ImageView
            android:id="@+id/thumbnail"
            android:layout_width="match_parent"
            android:layout_height="@dimen/album_cover_height"

        android:background="?attr/selectableItemBackgroundBorderless"
            android:clickable="true"
            android:scaleType="fitXY" />

        <TextView
            android:id="@+id/title"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_below="@id/thumbnail"
            android:paddingLeft="@dimen/album_title_padding"
            android:paddingRight="@dimen/album_title_padding"
            android:paddingTop="@dimen/album_title_padding"
            android:textColor="@color/album_title"
            android:textSize="@dimen/album_title" />

        <TextView
            android:id="@+id/count"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_below="@id/title"

        android:paddingBottom="@dimen/songs_count_padding_bottom"
            android:paddingLeft="@dimen/album_title_padding"
            android:paddingRight="@dimen/album_title_padding"
            android:textSize="@dimen/songs_count" />

        <ImageView
            android:id="@+id/overflow"
            android:layout_width="@dimen/ic_album_overflow_width"

        android:layout_height="@dimen/ic_album_overflow_height"
            android:layout_alignParentRight="true"
            android:layout_below="@id/thumbnail"

        android:layout_marginTop="@dimen/ic_album_overflow_margin_top"
            android:scaleType="centerCrop"
            android:src="@drawable/ic_dots" />

    </RelativeLayout>
</androidx.cardview.widget.CardView>
</LinearLayout>

```

Navegue até a pasta `java/nome.do.seu.pacote/res/layout` siga os passos já realizados em exercícios anteriores para criar uma arquivo e nomeie-a como **`conteudo_main.xml`**. Quando criar esse arquivo de layout, implemente o código abaixo:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/viewBg"
    app:layout_behavior="@string/appbar_scrolling_view_behavior"
    tools:context="br.com.aluno.cardviewapp.MainActivity"
    tools:showIn="@layout/activity_main">

    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/recycler_view"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:clipToPadding="false"
        android:scrollbars="vertical" />

</RelativeLayout>
```

Navegue até a pasta `java/ nome.do.seu.pacote/` siga os passos já realizados em exercícios anteriores para criar uma classe e nomeie-a como **`Album.kt`**. Quando criar a classe, implemente o código abaixo:

```
class Album {
    var name: String? = null
    var numSongs = 0
    var thumbnail = 0

    constructor() {}
    constructor(name: String?, numSongs: Int, thumbnail: Int) {
        this.name = name
        this.numSongs = numSongs
        this.thumbnail = thumbnail
    }
}
```

Em seguida, Navegue até a pasta `java/ nome.do.seu.pacote/` siga os passos já realizados em exercícios anteriores para criar uma classe e nomeie-a como **`AlbumsAdapter.kt`**. Quando criar a classe, implemente o código abaixo. Esta classe mostra como incluir um widget Android CardView em um RecyclerView:

```

import android.content.Context
import android.view.*
import android.widget.ImageView
import android.widget.TextView
import android.widget.Toast
import androidx.appcompat.widget.PopupMenu
import androidx.recyclerview.widget.RecyclerView
import com.bumptech.glide.Glide

class AlbumsAdapter(private val mContext: Context, private val
albumList: List<Album>) :
    RecyclerView.Adapter<AlbumsAdapter.MyViewHolder>() {
    inner class MyViewHolder(view: View) :
        RecyclerView.ViewHolder(view) {
        var title: TextView
        var count: TextView
        var thumbnail: ImageView
        var overflow: ImageView

        init {
            title = view.findViewById<View>(R.id.title) as TextView
            count = view.findViewById<View>(R.id.count) as TextView
            thumbnail = view.findViewById<View>(R.id.thumbnail) as
            ImageView
            overflow = view.findViewById<View>(R.id.overflow) as
            ImageView
        }
    }

    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): MyViewHolder {
        val itemView = LayoutInflater.from(parent.context)
            .inflate(R.layout.album_card, parent, false)
        return MyViewHolder(itemView)
    }

    override fun onBindViewHolder(holder: MyViewHolder, position: Int) {
        val album = albumList[position]
        holder.title.text = album.name
        holder.count.setText(" musicas" + album.numOfSongs)

        // loading album cover using Glide library
        Glide.with(mContext).load(album.thumbnail).into(holder.thumbnail)
        holder.overflow.setOnClickListener {
            showPopupMenu(holder.overflow)
        }
    }

    /**
     * Showing popup menu when tapping on 3 dots
     */
    private fun showPopupMenu(view: View) {
        // inflate menu
        val popup = PopupMenu(mContext, view)
        val inflater = popup.menuInflater
        inflater.inflate(R.menu.menu_album, popup.menu)
        popup.setOnMenuItemClickListener(MyMenuItemClickListener())
        popup.show()
    }
}

```

```

    /**
     * Click listener for popup menu items
     */
    internal inner class MyMenuItemClickListener :
        PopupMenu.OnMenuItemClickListener {
        override fun onMenuItemClick(menuItem: MenuItem): Boolean {
            when (menuItem.itemId) {
                R.id.action_add_favourite -> {
                    Toast.makeText(mContext, "Adicionado aos
favoritos", Toast.LENGTH_SHORT).show()
                    return true
                }
                R.id.action_play_next -> {
                    Toast.makeText(mContext, "Adicionado a fila",
Toast.LENGTH_SHORT).show()
                    return true
                }
                else -> {
                }
            }
            return false
        }
    }

    override fun getItemCount(): Int {
        return albumList.size
    }
}

```

A activity abaixo mostra a lista CardView. Abra seu arquivo **MainActivity.kt** e implemente o código abaixo:

```

import android.graphics.Rect
import android.os.Bundle
import android.util.TypedValue
import android.view.View
import android.widget.ImageView
import androidx.appcompat.app.AppCompatActivity
import androidx.appcompat.widget.Toolbar
import androidx.recyclerview.widget.DefaultItemAnimator
import androidx.recyclerview.widget.GridLayoutManager
import androidx.recyclerview.widget.RecyclerView
import androidx.recyclerview.widget.RecyclerView.ItemDecoration
import com.bumptech.glide.Glide
import com.google.android.material.appbar.AppBarLayout
import
com.google.android.material.appbar.AppBarLayout.OnOffsetChangedListener
r
import com.google.android.material.appbar.CollapsingToolbarLayout
import java.util.*

class MainActivity : AppCompatActivity() {
    private var recyclerView: RecyclerView? = null
    private var adapter: AlbumsAdapter? = null
    private var albumList: MutableList<Album>? = null
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        val toolbar = findViewById<View>(R.id.toolbar) as Toolbar

```

```

        setSupportActionBar(toolbar)
        initCollapsingToolbar()
        recyclerView = findViewById<View>(R.id.recycler_view) as
RecyclerView
        albumList = ArrayList()
        adapter = AlbumsAdapter(this, albumList as ArrayList<Album>)
        val layoutManager: RecyclerView.LayoutManager =
GridLayoutManager(this, 2)
        recyclerView!!.layoutManager = layoutManager
        recyclerView!!.addItemDecoration(GridSpacingItemDecoration(2,
dpToPx(10), true))
        recyclerView!!.itemAnimator = DefaultItemAnimator()
        recyclerView!!.adapter = adapter
        prepareAlbums()
        try {
            Glide.with(this).load(R.drawable.cover)
                .into(findViewById<View>(R.id.backdrop) as ImageView)
        } catch (e: Exception) {
            e.printStackTrace()
        }
    }

    /**
     * Initializing collapsing toolbar
     * Will show and hide the toolbar title on scroll
     */
    private fun initCollapsingToolbar() {
        val collapsingToolbar =
            findViewById<View>(R.id.collapsing_toolbar) as
CollapsingToolbarLayout
        collapsingToolbar.title = " "
        val appBarLayout = findViewById<View>(R.id.appbar) as
AppBarLayout
        appBarLayout.setExpanded(true)

        // hiding & showing the title when toolbar expanded &
collapsed
        appBarLayout.addOnOffsetChangedListener(object :
OnOffsetChangedListener {
            var isShow = false
            var scrollRange = -1
            override fun onOffsetChanged(appBarLayout: AppBarLayout,
verticalOffset: Int) {
                if (scrollRange == -1) {
                    scrollRange = appBarLayout.totalScrollRange
                }
                if (scrollRange + verticalOffset == 0) {
                    collapsingToolbar.title =
getString(R.string.app_name)
                    isShow = true
                } else if (isShow) {
                    collapsingToolbar.title = " "
                    isShow = false
                }
            }
        })
    }

    /**
     * Adding few albums for testing
     */

```

```

private fun prepareAlbums() {
    val covers = intArrayOf(
        R.drawable.album1,
        R.drawable.album2,
        R.drawable.album3,
        R.drawable.album4,
        R.drawable.album5,
        R.drawable.album6,
        R.drawable.album7,
        R.drawable.album8,
        R.drawable.album9,
        R.drawable.album10,
        R.drawable.album11,
        R.drawable.album12,
        R.drawable.album13,
        R.drawable.album14,
        R.drawable.album15,
        R.drawable.album16,
        R.drawable.album17,
        R.drawable.album18,
        R.drawable.album19,
        R.drawable.album20
    )
    var a = Album("Piece of Mind", 9, covers[0])
    albumList!! .add(a)
    a = Album("Metallica", 12, covers[1])
    albumList!! .add(a)
    a = Album("Master of Reality", 8, covers[2])
    albumList!! .add(a)
    a = Album("Definitely Maybe", 12, covers[3])
    albumList!! .add(a)
    a = Album("BadMotorFinger", 12, covers[4])
    albumList!! .add(a)
    a = Album("Elvis Presley", 12, covers[5])
    albumList!! .add(a)
    a = Album("Dirt", 13, covers[6])
    albumList!! .add(a)
    a = Album("Muddy Waters at Newport", 9, covers[7])
    albumList!! .add(a)
    a = Album("Let it Be", 12, covers[8])
    albumList!! .add(a)
    a = Album("Misfits Collection ", 20, covers[9])
    albumList!! .add(a)
    a = Album("Nevermind ", 13, covers[10])
    albumList!! .add(a)
    a = Album("London Calling ", 19, covers[11])
    albumList!! .add(a)
    a = Album("Led Zeppelin II ", 13, covers[12])
    albumList!! .add(a)
    a = Album("Some Girls ", 10, covers[13])
    albumList!! .add(a)
    a = Album("Aladdin Sane ", 15, covers[14])
    albumList!! .add(a)
    a = Album("Ramones ", 14, covers[15])
    albumList!! .add(a)
    a = Album("Burn ", 12, covers[16])
    albumList!! .add(a)
    a = Album("Morrison Hotel ", 11, covers[17])
    albumList!! .add(a)
    a = Album("Van Halen ", 11, covers[18])
    albumList!! .add(a)
}

```

```

        a = Album("Pendulum ", 10, covers[19])
        albumList!!.add(a)
        adapter!!.notifyDataSetChanged()
    }

    /**
     * RecyclerView item decoration - give equal margin around grid
     * item
     */
    inner class GridSpacingItemDecoration(
        private val spanCount: Int,
        private val spacing: Int,
        private val includeEdge: Boolean
    ) :
        ItemDecoration() {
        override fun getItemOffsets(
            outRect: Rect,
            view: View,
            parent: RecyclerView,
            state: RecyclerView.State
        ) {
            val position = parent.getChildAdapterPosition(view) // item position
            val column = position % spanCount // item column
            if (includeEdge) {
                outRect.left =
                    spacing - column * spacing / spanCount // spacing
                - column * ((1f / spanCount) * spacing)
                outRect.right =
                    (column + 1) * spacing / spanCount // (column + 1)
                * ((1f / spanCount) * spacing)
                if (position < spanCount) { // top edge
                    outRect.top = spacing
                }
                outRect.bottom = spacing // item bottom
            } else {
                outRect.left = column * spacing / spanCount // column
                * ((1f / spanCount) * spacing)
                outRect.right =
                    spacing - (column + 1) * spacing / spanCount // spacing
                - (column + 1) * ((1f / spanCount) * spacing)
                if (position >= spanCount) {
                    outRect.top = spacing // item top
                }
            }
        }
    }

    /**
     * Converting dp to pixel
     */
    private fun dpToPx(dp: Int): Int {
        val r = resources
        return Math.round(
            TypedValue.applyDimension(
                TypedValue.COMPLEX_UNIT_DIP,
                dp.toFloat(),
                r.displayMetrics
            )
        )
    }
}

```

```
}
```

Rode seu aplicativo. Confira o resultado.

## Acessando a câmera

Acessar o hardware de câmera do device Android é uma das features mais usadas por uma significativa parte dos aplicativos existente dentro do ecossistema Android. Para armazenar as fotos/vídeos obtidas por através dos aplicativos será usado MediaStore.ACTION\_IMAGE\_CAPTURE para iniciar a aplicação de câmera existente instalada em seu telefone. Sua sintaxe é dada abaixo:

```
Intent intent = new Intent(android.provider.MediaStore.ACTION_IMAGE_CAPTURE);
```

Além da Intent exibida acima, existem outras intents disponíveis fornecidos pela MediaStore. Eles são listados da seguinte forma:

Sr.No	Tipo de Intent e sua descrição
1	<b>ACTION_IMAGE_CAPTURE_SECURE</b> Retorna a imagem capturada da câmera, quando o dispositivo está protegido
2	<b>ACTION_VIDEO_CAPTURE</b> chama o aplicativo de vídeo existente no Android para capturar o vídeo
3	<b>EXTRA_SCREEN_ORIENTATION</b> Ele é usado para definir a orientação da tela como vertical ou paisagem
4	<b>EXTRA_FULL_SCREEN</b> usado para controlar a interface do usuário do ViewImage
5	<b>INTENT_ACTION_VIDEO_CAMERA</b>

	Intent usada para iniciar a câmera no modo de vídeo
6	<b>EXTRA_SIZE_LIMIT</b> usado para especificar o limite de tamanho de captura de vídeo ou imagem

Na sequencia será usado a função `startActivityForResult ()` para iniciar esta activity e aguardar o resultado. Sua sintaxe é dada abaixo:

```
startActivityForResult(intent, 0)
```

Este método foi definido na activity class. É invocado a partir da `MainActivity.java`. Existem métodos definidos na activity class que produzem o mesmo resultado, quando não se está chamando da activity principal, mas de outro lugar. Eles estes:

Sr.No	Activity function description
1	<b>startActivityForResult(Intent intent, int requestCode, Bundle options)</b> inicia uma activity, mas pode ter um pacote adicional de opções com ela
2	<b>startActivityFromChild(Activity child, Intent intent, int requestCode)</b> inicia a activity quando sua activity é filho de qualquer outra activity
3	<b>startActivityFromChild(Activity child, Intent intent, int requestCode, Bundle options)</b> It work same as above , but it can take extra values in the shape of bundle with it
4	<b>startActivityFromFragment(Fragment fragment, Intent intent, int requestCode)</b>

	Funciona da mesma forma exemplificada acima, mas pode possuir valores extras na forma de pacote
5	<b>startActivityFromFragment(Fragment fragment, Intent intent, int requestCode, Bundle options)</b> Ele não apenas lança a activity do fragment, mas pode ter valores extras com ele

Independentemente da função usada para iniciar a activity, todos retornarão o resultado(captura de imagem ou video). O resultado pode ser obtido substituindo a função onActivityResult.

### Projeto Camera Access

	<b>Descrição</b>
1	Você usará Android Studio IDE para criar um aplicativo Android e nomeá-lo como CameraApp
2	Modifique o arquivo src / MainActivity.java para adicionar código Intent para iniciar a câmera.
3	Modifique o arquivo de layout XML file res/layout/activity_main.xml
4	Adicione a permissão da Câmera e execute o aplicativo; escolha um dispositivo Android executável e instale o aplicativo e verifique os resultados.

Abra seu arquivo **res/layout/activity\_main.xml** e implemente o código abaixo:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:app="http://schemas.android.com/apk/res-auto"
  xmlns:tools="http://schemas.android.com/tools"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  tools:context=".MainActivity">
```

```

<RelativeLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintBottom_toBottomOf="parent"
    android:id="@+id/relative"
    >

    <ImageView
        android:id="@+id/ivImage"
        android:layout_width="250dp"
        android:layout_height="250dp"
        android:layout_centerInParent="true"
        tools:src="@tools:sample/avatars" />
    <VideoView
        android:layout_width="match_parent"
        android:layout_height="250dp"
        android:id="@+id/vvVideo"
        />

</RelativeLayout>

<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    app:layout_constraintTop_toBottomOf="@id/relative"
    app:layout_constraintStart_toStartOf="@id/relative"
    app:layout_constraintEnd_toEndOf="@id/relative"
    android:layout_marginTop="5dp"
    android:id="@+id	btnTakePhoto"
    android:text="Tirar foto"
    />

<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    app:layout_constraintTop_toBottomOf="@id	btnTakePhoto"
    app:layout_constraintStart_toStartOf="@id/relative"
    app:layout_constraintEnd_toEndOf="@id/relative"
    android:layout_marginTop="5dp"
    android:id="@+id	btnVideo"
    android:text="Video"
    />

<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    app:layout_constraintTop_toBottomOf="@id	btnVideo"
    app:layout_constraintStart_toStartOf="@id/relative"
    app:layout_constraintEnd_toEndOf="@id/relative"
    android:layout_marginTop="5dp"
    android:id="@+id	btnSavePhoto"
    android:text="salvar foto"
    />
</androidx.constraintlayout.widget.ConstraintLayout>

```

Abra seu arquivo **src>MainActivity.kt**. Implemente o código abaixo:

```

import android.Manifest
import android.app.Activity
import android.content.DialogInterface
import android.content.Intent
import android.content.pm.PackageManager
import android.graphics.Bitmap
import android.net.Uri
import android.os.Environment
import android.provider.MediaStore

import android.os.Bundle
import android.util.Log
import android.view.View
import android.widget.Button
import android.widget.EditText
import android.widget.ImageView
import android.widget.MediaController
import android.widget.Toast
import android.widget.VideoView
import androidx.appcompat.app.AlertDialog
import androidx.appcompat.app.AppCompatActivity
import androidx.core.app.ActivityCompat
import androidx.core.content.ContextCompat

import java.io.File
import java.io.FileOutputStream

class MainActivity : AppCompatActivity() {

    internal lateinit var image: ImageView
    internal lateinit var btnTakePhoto: Button
    internal lateinit var btnSavePhoto: Button
    internal lateinit var btnVideo: Button
    internal lateinit var videoView: VideoView
    internal var cameraPermission: Int = 0
    internal var writePermission: Int = 0
    internal var bitmap: Bitmap? = null

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        //region permissões
        cameraPermission = ContextCompat.checkSelfPermission(this,
Manifest.permission.CAMERA)
        writePermission = ContextCompat.checkSelfPermission(this,
Manifest.permission.WRITE_EXTERNAL_STORAGE)
        if (cameraPermission == PackageManager.PERMISSION_DENIED || writePermission == PackageManager.PERMISSION_DENIED)
            permission()
        //endregion

        //region bind
        btnSavePhoto = findViewById(R.id.btnSavePhoto)
        btnTakePhoto = findViewById(R.id.btnTakePhoto)
        btnVideo = findViewById(R.id.btnVideo)
        image = findViewById(R.id.ivImage)
    }
}

```

```

videoView = findViewById(R.id.vvVideo)
//endregion

//visibility
videoView.visibility = View.INVISIBLE
image.visibility = View.INVISIBLE
btnSavePhoto.visibility = View.INVISIBLE
//endregion

//adiciona um media controle ao video
val mediaController = MediaController(this)
videoView.setMediaController(mediaController)

//region eventos clique
btnTakePhoto.setOnClickListener {
    val takePhoto = Intent(MediaStore.ACTION_IMAGE_CAPTURE)
    startActivityForResult(takePhoto, COD_IMAGE_PHOTO)
}

btnSavePhoto.setOnClickListener(View.OnClickListener {
    if (bitmap == null) {
        Toast.makeText(this@MainActivity, "Tire uma foto
primeiro", Toast.LENGTH_SHORT).show()
        return@OnClickListener
    }

    val builder = AlertDialog.Builder(this@MainActivity)
    builder.setTitle("Salvar")
    val edName = EditText(this@MainActivity)
    builder.setView(edName)
    builder.setMessage("De um nome para a foto")
    builder.setPositiveButton("OK") { dialog, which ->
        val sName = edName.text.toString()
        save(sName)
    }
    builder.setNegativeButton("Cancel") { dialog, which ->
        dialog.cancel()
        builder.show()
    }
}
btnVideo.setOnClickListener {
    val intent = Intent(MediaStore.ACTION_VIDEO_CAPTURE)
    startActivityForResult(intent, COD_VIDEO)
}
//endregion

}

//metodo para salvar a imagem no celular
@SuppressLint("DEPRECATION")
fun save(name: String) {

    val root =
Environment.getExternalStorageDirectory().absolutePath //pega o
caminho interno
    val myDir = File("$root/images") //cria um caminho para a
pasta
    myDir.mkdirs() // cria a pasta
    val fname = "$name.jpg" // da um nome para um arquivo imagem
}

```

```

        val file = File(myDir, fname) //cria o arquivo na pasta
        if (file.exists()) file.delete() //verifica se j o existe um
arquivo com o nome se sim ele deleta
        try {
            val out = FileOutputStream(file)
            bitmap!!.compress(Bitmap.CompressFormat.JPEG, 100, out)
            out.flush()
            out.close()
            val b = AlertDialog.Builder(this)
            b.setTitle("Mensagem")
            b.setMessage("Arquivo salvo no celular no caminho " +
file.getAbsolutePath())
            b.setPositiveButton("Ok") { dialog, which ->
dialog.dismiss() }
            b.show()

        } catch (e: Exception) {
            e.printStackTrace()
        }
    }

    override fun onActivityResult(requestCode: Int, resultCode: Int,
data: Intent?) {
        super.onActivityResult(requestCode, resultCode, data)
        if (requestCode == COD_IMAGE_PHOTO && resultCode ==
Activity.RESULT_OK) {
            if (image.visibility == View.INVISIBLE) {
                image.visibility = View.VISIBLE
                btnSavePhoto.visibility = View.VISIBLE
                videoView.visibility = View.INVISIBLE
            }
            //pega imagem
            val extras = data!!.extras
            bitmap = extras!![ "data" ] as Bitmap
            image.setImageBitmap(bitmap)

        }

        if (requestCode == COD_VIDEO && resultCode ==
Activity.RESULT_OK) {
            if (videoView.visibility == View.INVISIBLE) {
                videoView.visibility = View.VISIBLE
                image.visibility = View.INVISIBLE
                btnSavePhoto.visibility = View.INVISIBLE
            }

            val videoUri = data!![ "data" ] //pega o caminho do video que
foi salvo
            val alert = AlertDialog.Builder(this)
            alert.setTitle("Mensagem")
            alert.setMessage("Video salvo em: " +
videoUri!!.toString())
            videoView.setVideoURI(videoUri) //inicia no player e video
            videoView.start()

        }
    }
}

```

```

//metodo que pede permissões
fun permission() {
    ActivityCompat.requestPermissions(this,
arrayOf(Manifest.permission.CAMERA,
Manifest.permission.WRITE_EXTERNAL_STORAGE,
Manifest.permission.READ_EXTERNAL_STORAGE), COD_PERMISSION)
}

override fun onRequestPermissionsResult(requestCode: Int,
permissions: Array<String>, grantResults: IntArray) {
    super.onRequestPermissionsResult(requestCode, permissions,
grantResults)

    if (requestCode == COD_PERMISSION) {
        if (cameraPermission == PackageManager.PERMISSION_DENIED
|| writePermission == PackageManager.PERMISSION_DENIED)
            permission()
    }
}

override fun onPause() {
    super.onPause()
    //pausa o video se entrar em segundo plano
    if (videoView.visibility == View.VISIBLE) {
        videoView.pause()
    }
}

override fun onPostResume() {
    super.onPostResume()
    //reinicia o video ao voltar para o primeiro plano
    if (videoView.visibility == View.VISIBLE) {
        videoView.start()
    }
}

companion object {
    internal val COD_IMAGE_PHOTO = 1
    internal val COD_PERMISSION = 1
    internal val COD_VIDEO = 2
}
}

```

 Na sequencia, abra seu arquivo **AndroidManifest.xml** e as modificações necessárias:

```

<uses-feature android:name="android.hardware.camera"
    android:required="true" />

<uses-permission android:name="android.permission.CAMERA" />
<uses-permission
    android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission
    android:name="android.permission.READ_EXTERNAL_STORAGE" />

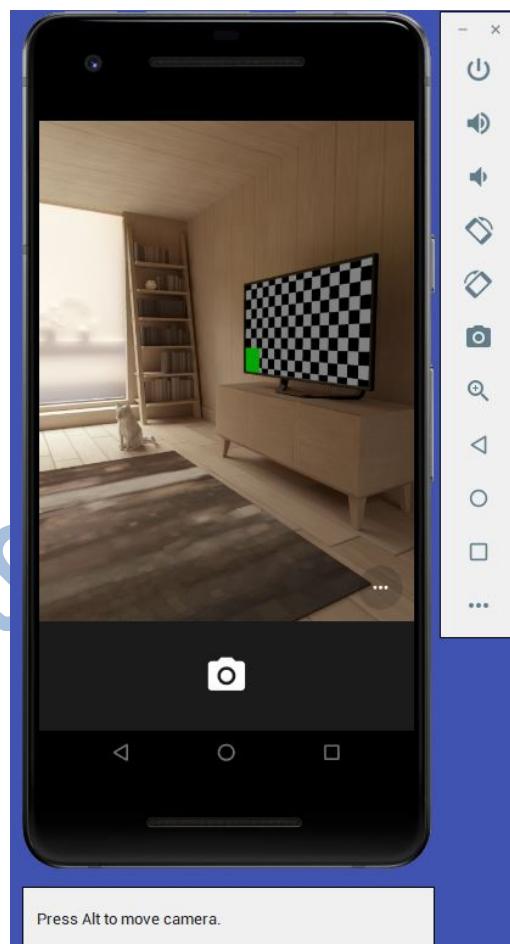
<application
    android:allowBackup="true"

```

```
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:roundIcon="@mipmap/ic_launcher_round"
    android:supportsRtl="true"
    android:theme="@style/Theme.CameraKotlin">
    <activity android:name=".MainActivity">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />

            <category android:name="android.intent.category.LAUNCHER"
/>
        </intent-filter>
    </activity>
</application>
```

Rode seu aplicativo. Confira o resultado.



## Captura de Audio

O Android possui um microfone embutido através do qual é possível capturar áudio e armazená-lo. Existem muitas maneiras de fazer isso, mas a mais comum é através da classe MediaRecorder.

O Android oferece a classe MediaRecorder para gravar áudio ou vídeo. Para usar a classe MediaRecorder, é preciso criar uma instância da classe MediaRecorder.

Agora, é preciso definir o formato de origem, saída, codificação e o arquivo de saída.

Depois de especificar a fonte de áudio, o formato e o seu arquivo de saída, é possível chamar os dois métodos básicos para preparar e começar a gravar o áudio.

Além desses métodos, existem outros métodos listados na classe MediaRecorder que permitem exercer mais controle sobre a gravação de áudio e vídeo.

Métodos e descrição	
1	<b>set AudioSource()</b> Este método especifica a fonte de áudio a ser gravada
2	<b>set Video Source()</b> Este método especifica a origem do vídeo a ser gravado
3	<b>set Output Format()</b> Este método especifica o formato de áudio no qual o áudio deve ser armazenado
4	<b>set Audio Encoder()</b> Este método especifica o codificador de áudio a ser usado
5	<b>set Output File()</b> Este método configura o caminho para o arquivo no qual o áudio gravado deve ser armazenado
6	<b>stop()</b>

	Esse método interrompe o processo de gravação.
7	<b>release()</b> Este método deve ser chamado quando a instância do gravador for necessária.

## Projeto Audio Capture

Nesste exercício será criada a classe MediaRecorder para capturar áudio e, em seguida, a classe MediaPlayer para reproduzir o áudio gravado.

	<b>Descrição</b>
1	Você usará Android Studio IDE para criar um aplicativo Android e nomeá-lo como AudioCapture
2	Modificar o arquivo src/MainActivity.kt para adicionar o código de AudioCapture
3	Modifica o arquivo layout XML file res/layout/activity_main.xml e adicionar os components de IU requeridos.
4	Modificar o arquivo AndroidManifest.xml para adicionar as permissões necessárias.
5	Execute o aplicativo e escolha um dispositivo Android executável e instale o aplicativo e verifique os resultados.

Este é o arquivo AndroidManifest.xml. Implemente o código abaixo:

```

<uses-permission
    android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.RECORD_AUDIO" />
<uses-permission android:name="android.permission.STORAGE" />

<application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:roundIcon="@mipmap/ic_launcher_round"
    android:supportsRtl="true"
    android:theme="@style/Theme.AudioKotlin">
    <activity android:name=".MainActivity">
```

```

<intent-filter>
    <action android:name="android.intent.action.MAIN" />

    <category android:name="android.intent.category.LAUNCHER"
/>
</intent-filter>
</activity>
</application>

```

Este é o arquivo **dimens.xml**. Se o Android Studio não a criou de maneira automática proceda dessa forma: navegue até a pasta res/values e, com o botão direito do mouse, clique na pasta values. Aponte o cursor do mouse em New; na próxima janela clique em Values resource files e nomeie-o o arquivo como **dimens.xml**. Esse é o conteúdo do arquivo. Implemente o código abaixo:

```

<resources>
    <!-- Default screen margins, per the Android Design guidelines. -->
    <dimen name="activity_horizontal_margin">16dp</dimen>
    <dimen name="activity_vertical_margin">16dp</dimen>
</resources>

```

Este é o arquivo **activity\_main.xml**. Implemente o código abaixo:

```

<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin">

    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/imageView"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Gravar"
        android:id="@+id/button"
        android:layout_below="@+id/imageView"
        android:layout_alignParentLeft="true"
        android:layout_marginTop="37dp"
        />

```

```

<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Parar"
    android:id="@+id/button2"
    android:layout_alignTop="@+id/button"
    android:layout_centerHorizontal="true"
    />

<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Play"
    android:id="@+id/button3"
    android:layout_alignTop="@+id/button2"
    android:layout_alignParentRight="true"
    android:layout_alignParentEnd="true"
    />

<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="PARAR A GRAVAÇÃO"
    android:id="@+id/button4"
    android:layout_below="@+id/button2"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="10dp"
    />
</RelativeLayout>

```

Este é o arquivo **MainActivity.kt**. Implemente o código abaixo:

```

import android.Manifest.permission
import android.content.pm.PackageManager
import android.media.MediaPlayer
import android.media.MediaRecorder
import android.os.Bundle
import android.os.Environment
import android.os.Environment.getExternalStorageDirectory
import android.view.View
import android.widget.Button
import android.widget.Toast
import androidx.appcompat.app.AppCompatActivity
import androidx.core.app.ActivityCompat
import androidx.core.content.ContextCompat
import java.io.IOException
import java.util.*

@SuppressLint("DEPRECATION")
class MainActivity : AppCompatActivity() {
    var buttonStart: Button? = null
    var buttonStop: Button? = null
    var buttonPlayLastRecordAudio: Button? = null
    var buttonStopPlayingRecording: Button? = null
    var AudioSavePathInDevice: String? = null
    var mediaRecorder: MediaRecorder? = null

```

```

var random: Random? = null
var RandomAudioFileName = "ABCDEFGHIJKLMNP"
var mediaPlayer: MediaPlayer? = null
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)
    buttonStart = findViewById<View>(R.id.button) as Button
    buttonStop = findViewById<View>(R.id.button2) as Button
    buttonPlayLastRecordAudio = findViewById<View>(R.id.button3) as
Button
    buttonStopPlayingRecording = findViewById<View>(R.id.button4) as
Button
    buttonStop!!.isEnabled = false
    buttonPlayLastRecordAudio!!.isEnabled = false
    buttonStopPlayingRecording!!.isEnabled = false
    random = Random()
    buttonStart!!.setOnClickListener {
        if (checkPermission()) {
            AudioSavePathInDevice =
                getExternalStorageDirectory().absolutePath + "/" +
                    CreateRandomAudioFileName(5) + "AudioRecording.3gp"
            MediaRecorderReady()
            try {
                mediaRecorder!!.prepare()
                mediaRecorder!!.start()
            } catch (e: IllegalStateException) {
                // TODO Auto-generated catch block
                e.printStackTrace()
            } catch (e: IOException) {
                // TODO Auto-generated catch block
                e.printStackTrace()
            }
            buttonStart!!.isEnabled = false
            buttonStop!!.isEnabled = true
            Toast.makeText(this@MainActivity, "Gravação iniciada",
                Toast.LENGTH_LONG).show()
        } else {
            requestPermission()
        }
    }
    buttonStop!!.setOnClickListener {
        mediaRecorder!!.stop()
        buttonStop!!.isEnabled = false
        buttonPlayLastRecordAudio!!.isEnabled = true
        buttonStart!!.isEnabled = true
        buttonStopPlayingRecording!!.isEnabled = false
        Toast.makeText(this@MainActivity, "Gravação terminada",
            Toast.LENGTH_LONG).show()
    }
    buttonPlayLastRecordAudio!!.setOnClickListener {
        buttonStop!!.isEnabled = false
        buttonStart!!.isEnabled = false
        buttonStopPlayingRecording!!.isEnabled = true
        mediaPlayer = MediaPlayer()
        try {
            mediaPlayer!!.setDataSource(AudioSavePathInDevice)
            mediaPlayer!!.prepare()
        } catch (e: IOException) {
            e.printStackTrace()
        }
    }
}

```

```

        }
        mediaPlayer!!.start()
        Toast.makeText(this@MainActivity, "Tocando a gravação",
            Toast.LENGTH_LONG).show()
    }
    buttonStopPlayingRecording!!.setOnClickListener {
        buttonStop!!.isEnabled = false
        buttonStart!!.isEnabled = true
        buttonStopPlayingRecording!!.isEnabled = false
        buttonPlayLastRecordAudio!!.isEnabled = true
        if (mediaPlayer != null) {
            mediaPlayer!!.stop()
            mediaPlayer!!.release()
            MediaRecorderReady()
        }
    }
}

fun MediaRecorderReady() {
    mediaRecorder = MediaRecorder()
    mediaRecorder!!.setAudioSource(MediaRecorder.AudioSource.MIC)
    mediaRecorder!!.setOutputFormat(MediaRecorder.OutputFormat.THREE_GPP)
    mediaRecorder!!.setAudioEncoder(MediaRecorder.OutputFormat.AMR_NB)
    mediaRecorder!!.setOutputFile(AudioSavePathInDevice)
}

fun CreateRandomAudioFileName(string: Int): String {
    val stringBuilder = StringBuilder(string)
    var i = 0
    while (i < string) {

stringBuilder.append(RandomAudioFileName[random!!.nextInt(RandomAudioFileName
.length)]))
        i++
    }
    return stringBuilder.toString()
}

private fun requestPermission() {
    ActivityCompat.requestPermissions(this@MainActivity,
        arrayOf(permission.WRITE_EXTERNAL_STORAGE, permission.RECORD_AUDIO),
        RequestPermissionCode)
}

override fun onRequestPermissionsResult(requestCode: Int,
                                         permissions: Array<String>,
                                         grantResults: IntArray) {
    when (requestCode) {
        RequestPermissionCode -> if (grantResults.size > 0) {
            val StoragePermission = grantResults[0] ==
                PackageManager.PERMISSION_GRANTED
            val RecordPermission = grantResults[1] ==
                PackageManager.PERMISSION_GRANTED
            if (StoragePermission && RecordPermission) {
                Toast.makeText(this@MainActivity, "Permissão garantida",
                    Toast.LENGTH_LONG).show()
            } else {
                Toast.makeText(this@MainActivity, "Permissão negada",
                    Toast.LENGTH_LONG).show()
            }
        }
    }
}

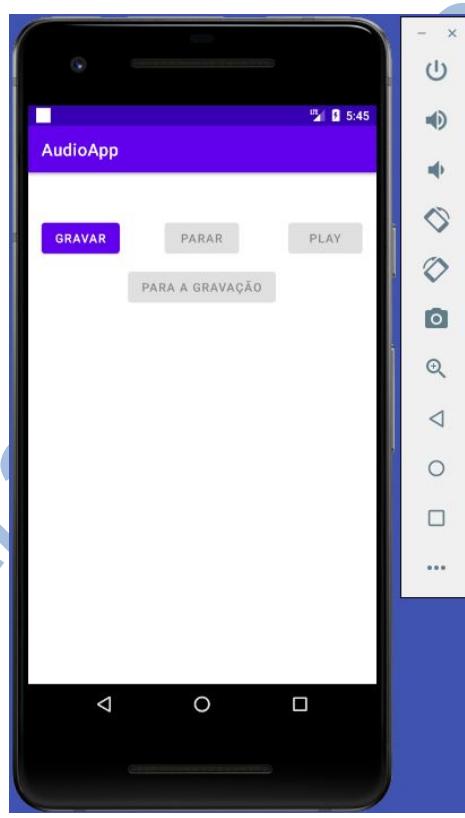
```

```
        }
    }
}

fun checkPermission(): Boolean {
    val result = ContextCompat.checkSelfPermission(applicationContext,
        permission.WRITE_EXTERNAL_STORAGE)
    val result1 = ContextCompat.checkSelfPermission(applicationContext,
        permission.RECORD_AUDIO)
    return result == PackageManager.PERMISSION_GRANTED &&
        result1 == PackageManager.PERMISSION_GRANTED
}

companion object {
    const val RequestPermissionCode = 1
}
}
```

Rode seu aplicativo. Confira o resultado:



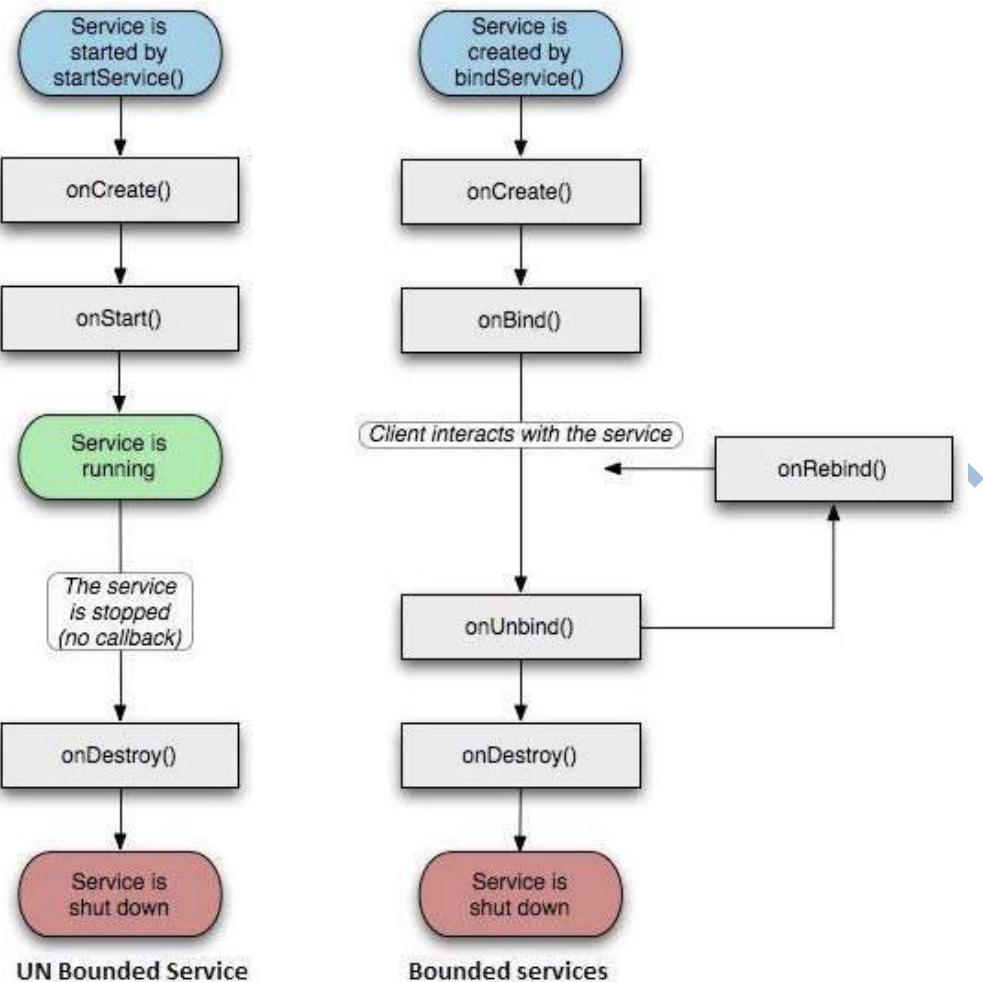
## Android Services

Um serviço(service) é um componente que é executado em segundo plano para operações de longa duração sem precisar interagir com o usuário e

funciona mesmo se o aplicativo for destruído(on destroy). Um serviço pode, essencialmente, levar dois estados:

	<b>Estado e Descrição</b>
1	<p><b>Started</b></p> <p>Um serviço é iniciado quando um componente de aplicativo, como uma activity, o inicia chamando startService (). Uma vez iniciado, um serviço pode ser executado em segundo plano indefinidamente, mesmo que o componente que o iniciou seja destruído.</p>
2	<p><b>Bound</b></p> <p>Um serviço é vinculado quando um componente de aplicativo se liga a ele através do bindService (). Um serviço vinculado oferece uma interface cliente-servidor que permite aos componentes interagir com o serviço, enviar pedidos, obter resultados e até mesmo fazê-lo em todos os processos com comunicação interprocesso (IPC).</p>

Um service possui métodos de retorno; retorno do ciclo de vida que é possível implementar para monitorar mudanças em seu estado; também é possível executar o trabalho no estágio apropriado. O seguinte diagrama mostra o ciclo de vida quando o service é criado com startService () e o diagrama ao lado mostra o ciclo de vida quando o service é criado com bindService (): (imagem cortesia: android.com)



Para criar um service, é criada uma classe Java que estende a classe base do Service ou uma das suas subclasses existentes. A classe base do service define vários métodos de retorno de chamada e os mais importantes são dados abaixo. Não é necessário implementar todos os métodos de retorno de chamada. No entanto, é importante que fique claro que cada um deles e implemente aqueles que assegurem que seu aplicativo se comporte da maneira que os usuários esperam.

Callback e Descrição	
1	<b>onStartCommand()</b> O sistema chama esse método quando outro componente, como uma activity, solicita que o service seja iniciado, chamando <code>startService()</code> . Implementando

	este método, é sua responsabilidade parar o service quando seu trabalho for feito, chamando os métodos stopSelf () ou stopService ().
2	<b>onBind()</b> O sistema chama esse método quando um outro componente quer se ligar ao service chamando bindService (). Implementando este método, é necessário fornecer uma interface que os clientes usem para se comunicar com o service, retornando um objeto IBinder. Também é necessário sempre implementar este método, mas caso a ligação não precise ser permitida, basta torná-lo nulo.
3	<b>onUnbind()</b> O sistema chama esse método quando todos os clientes se desconectaram de uma interface específica publicada pelo serviço.
4	<b>onRebind()</b> O sistema chama esse método quando novos clientes se conectaram ao serviço, depois de ter sido previamente notificado de que todos tinham se desconectado no seu OnUnbind (Intent).
5	<b>onCreate()</b> O sistema chama este método quando o serviço foi criado pela primeira vez usando onStartCommand () ou onBind (). Esta chamada é necessária para executar uma configuração única.
6	<b>onDestroy()</b> O sistema chama esse método quando o serviço não é mais usado e está sendo destruído. Seu serviço deve implementar isso para limpar todos os recursos, como threads, listeners registrados, receivers, etc.

## Projeto Service

	<b>Descrição</b>
1	Você usará Android Studio IDE para criar um aplicativo Android e nomeá-lo como ServiceApp

2	Modificar seu arquivo <i>MainActivity.kt</i> para adicionar os métodos <i>startService()</i> and <i>stopService()</i> .
3	Criar um novo arquivo java <i>MyService.kt</i> . Este arquivo terá implementação de métodos relacionados ao Service Android.
4	Definir o Service no arquivo <i>AndroidManifest.xml</i> usando a tag <i>&lt;service ... /&gt;</i> . Um aplicativo pode ter um ou mais services sem restrições.
5	Modificar o conteúdo padrão do arquivo <i>res / layout / activity_main.xml</i> para incluir dois botões no <i>LinearLayout</i> .
6	Não é necessário alterar quaisquer constantes no arquivo <i>res / values / strings.xml</i> .
7	Execute o aplicativo para iniciar o aplicativo e verifique o resultado das alterações feitas no aplicativo.

Neste passo está o conteúdo do arquivo de activity principal modificado *MainActivity.kt*. Este arquivo pode incluir cada um dos métodos fundamentais do ciclo de vida. Aqui, adiciona-se os métodos *startService ()* e *stopService ()* para iniciar e parar o service.

Este é o arquivo, com conteúdo modificado, do arquivo *AndroidManifest.xml*. Aqui, adiciona-se a tag *<service ... />* para incluir o servissee. Implemente o código abaixo:

```

<application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:roundIcon="@mipmap/ic_launcher_round"
    android:supportsRtl="true"
    android:theme="@style/Theme.ServiceKotlin">
    <activity android:name=".MainActivity">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />

            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
    <service android:name=".MyService" />

```

```
</application>
```

Este é o conteúdo do arquivo **res/layout/activity\_main.xml** para incluir dois botões. Implemente o código abaixo:

```

<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Services"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:textSize="30dp" />

    <TextView
        android:id="@+id/textView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Services Android Kotlin"
        android:textColor="#000080"
        android:textSize="30dp"
        android:layout_above="@+id/imageButton"
        android:layout_centerHorizontal="true"
        android:layout_marginBottom="40dp" />

    <ImageButton
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/imageButton"
        android:layout_centerVertical="true"
        android:layout_centerHorizontal="true" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/button2"
        android:text="Iniciar Services"
        android:onClick="startService"
        android:layout_below="@+id/imageButton"
        android:layout_centerHorizontal="true" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Parar Services"
        android:id="@+id/button"
        android:onClick="stopService"
        android:layout_below="@+id/button2"
        android:layout_alignLeft="@+id/button2"
```

```

        android:layout_alignStart="@+id/button2"
        android:layout_alignRight="@+id/button2"
        android:layout_alignEnd="@+id/button2" />

    </RelativeLayout>

```

Este é o arquivo **dimens.xml**. Se o Android Studio não a criou de maneira automática proceda dessa forma: navegue até a pasta res/values e, com o botão direito do mouse, clique na pasta values. Aponte o cursor do mouse em New; na próxima janela clique em Values resource files e nomeie-o o arquivo como **dimens.xml**. Esse é o conteúdo do arquivo. Implemente o código abaixo:

```

<resources>
    <!-- Default screen margins, per the Android Design guidelines. -->
    <dimen name="activity_horizontal_margin">16dp</dimen>
    <dimen name="activity_vertical_margin">16dp</dimen>

</resources>

```

Está o conteúdo do **MyService.kt**. Este arquivo pode ter a implementação de um ou mais métodos associados ao service com base em requisitos. Por enquanto, será implementado apenas dois métodos *onStartCommand()* e *onDestroy()*:

```

import android.app.Service
import android.content.Intent
import android.os.IBinder
import android.widget.Toast
import androidx.annotation.Nullable

class MyService : Service() {
    @Nullable
    override fun onBind(intent: Intent): IBinder? {
        return null
    }

    override fun onStartCommand(intent: Intent, flags: Int, startId: Int): Int {
        // Permite continuar rodando até que seja parado
        Toast.makeText(this, "Service Iniciado",
        Toast.LENGTH_LONG).show()
        return START_STICKY
    }

    override fun onDestroy() {
        super.onDestroy()
        Toast.makeText(this, "Service Destruido",
        Toast.LENGTH_LONG).show()
    }
}

```

Finalizando, neste passo está o conteúdo do arquivo de activity principal modificado *MainActivity.kt*. Este arquivo pode incluir cada um dos métodos fundamentais do ciclo de vida. Aqui, adiciona-se os métodos *startService ()* e *stopService ()* para iniciar e parar o service.

```
import android.app.Activity
import android.content.Intent
import android.os.Bundle
import android.util.Log
import android.view.View

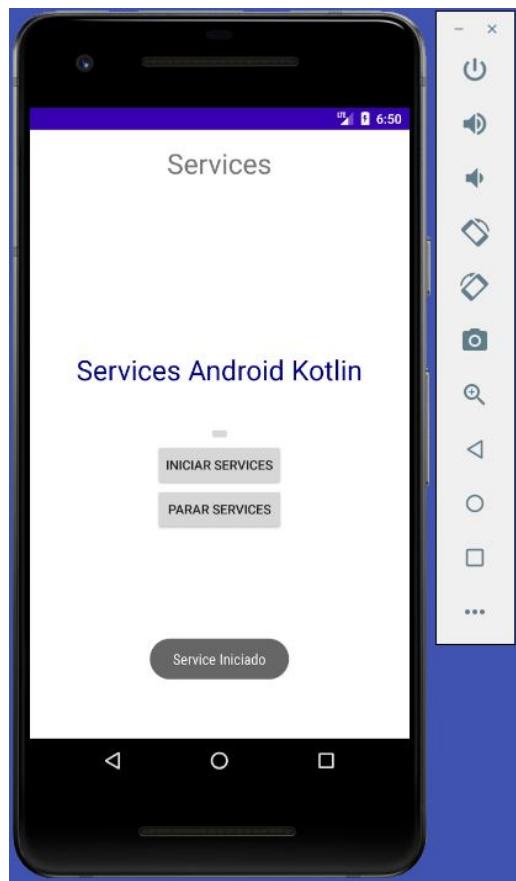
class MainActivity : Activity() {
    var msg = "Android : "

    /**Chamado quando o service é criado. */
    public override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        Log.d(msg, "The onCreate() event")
    }

    fun startService(view: View?) {
        startService(Intent(baseContext, MyService::class.java))
    }

    // Método para parar o Service
    fun stopService(view: View?) {
        stopService(Intent(baseContext, MyService::class.java))
    }
}
```

Rode seu aplicativo. Confira o resultado.



## Aplicando BroadCastReceiver

Como mostrado no em Fundamentos de App, os broadcast receivers simplesmente respondem a mensagens de transmissão de outros aplicativos ou do próprio sistema. Essas mensagens são algumas vezes chamadas de eventos ou intents. Por exemplo, os aplicativos também podem iniciar transmissões para permitir que outros aplicativos saibam que alguns dados foram baixados para o dispositivo e está disponível para serem usados, então este é um broadcast receiver que irá interceptar esta comunicação e iniciará a ação apropriada.

Estas são duas etapas importantes para que o BroadcastReceiver funcione para o sistema transmitido por intenções

- Criando o Host Broadcast Receiver;
- Registrando o Receptor de Transmissão;

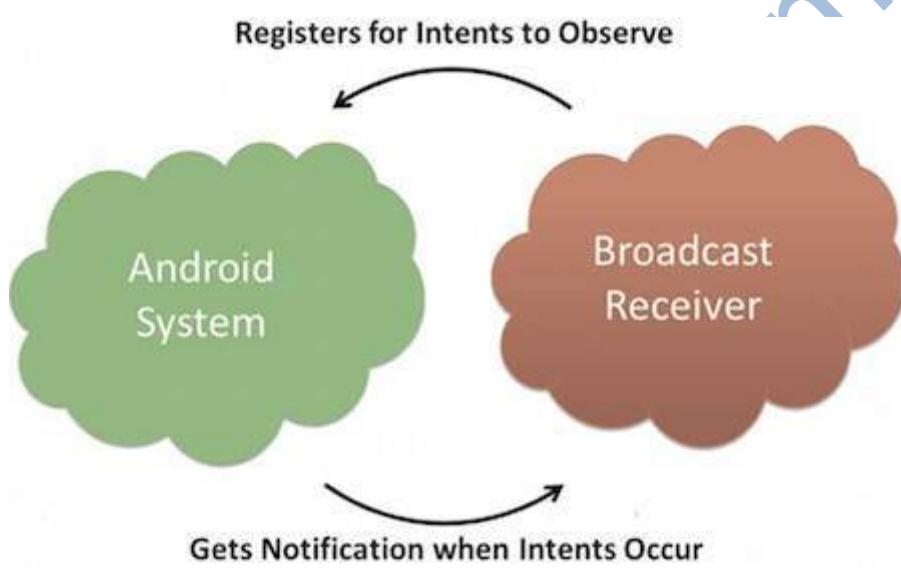
Há um passo adicional para implementar as intents personalizadas, entã, será necessário criar e transmitir essas intents.

## Criando BroadCast Receivers

Um BroadCast Receiver é implementado como uma subclasse da classe BroadcastReceiver e substituindo o método onReceive () em que cada mensagem é recebida como um parâmetro de objeto Intent.

## Registrando BroadcastReceiver

Um aplicativo “escuta” as intents de transmissão específicas registrando um broadcast receiver no arquivo AndroidManifest.xml. Considere que será registrado uma intente chamada MyReceiver para o evento gerado pelo sistema ACTION\_BOOT\_COMPLETED que é acionado pelo sistema assim que o sistema Android tiver concluído o processo de inicialização.



Agora, sempre que o dispositivo Android for inicializado, ele será interceptado pelo BroadcastReceiver MyReceiver e a lógica implementada dentro do Receive () será executada.

Existem vários eventos gerados pelo sistema definidos como campos estáticos finais na classe Intent. A tabela a seguir lista alguns eventos importantes do sistema:

	Constantes de Eventos e Descrição
1	<code>android.intent.action.BATTERY_CHANGED</code>

	Broadcast fixa contendo o estado de carga, o nível e outras informações sobre a bateria.
2	<b>android.intent.action.BATTERY_LOW</b> Indica condição de bateria fraca no dispositivo.
3	<b>android.intent.action.BATTERY_OKAY</b> Indica que a bateria está ok agora depois de estar baixa.
4	<b>android.intent.action.BOOT_COMPLETED</b> Isso é transmitido uma vez, depois que o sistema terminou a inicialização.
5	<b>android.intent.action.BUG_REPORT</b> Mostrar activity para relatar um bug.
6	<b>android.intent.action.CALL</b> Executa uma chamada para algo especificado pelos dados.
7	<b>android.intent.action.CALL_BUTTON</b> O usuário pressionou o botão "chamar" e acessar o discador ou outra interface do usuário apropriada para fazer uma chamada.
8	<b>android.intent.action.DATE_CHANGED</b> A data mudou.
9	<b>android.intent.action.REBOOT</b> Reboot do dispositivo.

### Broadcasting Custom Intents

Se é necessário que seu próprio aplicativo emita e envie intents personalizadas, necessariamente terão de ser criadas e enviadas essas intents usando o método sendBroadcast () dentro da sua activity class. Se o método

usado for SendStickyBroadcast (Intent), significa que a Intent que está enviando permanece “próxima” após a transmissão ser completada.

### Projeto BroadCastRec

Este exercício irá explicar como criar BroadcastReceiver para interceptar intents personalizadas. Uma vez que seja possível conhecer uma intenção personalizada, será possível programar seu aplicativo para interceptar as intents geradas pelo sistema.

	<b>Descrição</b>
1	Você usará Android Studio para criar um aplicativo Android e nomeá-lo como BroaCastReceiverApp
2	Modificar o arquivo principal <i>MainActivity.kt</i> e adicionar o método <i>broadcastIntent()</i> ;
3	Crie um novo arquivo java chamado <i>MyReceiver.kt</i> sob o pacote package <i>teste.app.com.broadcastreceiverapp</i> para definir um BroadcastReceiver.
4	Um aplicativo pode processar uma ou mais intents personalizadas e de sistema sem restrições. Todas as intents que deseja interceptar devem estar registradas em seu arquivo <i>AndroidManifest.xml</i> usando a tag <receiver ... />
5	Modificar o conteúdo padrão do arquivo <i>res / layout / activity_main.xml</i> para incluir um botão para transmitir a intent.
6	Não é necessário modificar o arquivo de seqüência de caracteres, Android studio cuida do arquivo <i>string.xml</i> .
7	Executar o aplicativo e verificar o resultado das alterações realizadas.

Este é o conteúdo do arquivo ***res / layout / activity\_main.xml*** para incluir um botão para transmitir a intent personalizada. Implemente o código abaixo:

```

<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="BroadcastReceiverApp"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:textSize="30dp" />

    <TextView
        android:id="@+id/textView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Broadcast personalizado "
        android:textColor="#000080"
        android:textSize="30dp"
        android:layout_below="@+id/textView1"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="70dp" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/button2"
        android:text="Broadcast Intent"
        android:onClick="broadcastIntent"
        android:layout_below="@+id/textView2"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="107dp" />

</RelativeLayout>

```

Este é o conteúdo do arquivo da activity principal modificado; *MainActivity.kt*. Este arquivo pode incluir cada um dos métodos fundamentais do ciclo de vida. É adicionado o método `broadcastIntent()` para transmitir uma intent personalizada. Implemente o código abaixo:

```

import android.app.Activity
import android.content.Intent
import android.os.Bundle
import android.view.View

class MainActivity : Activity() {
    /**Chamado quando a primeira activity é criada */
    public override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
    }
}

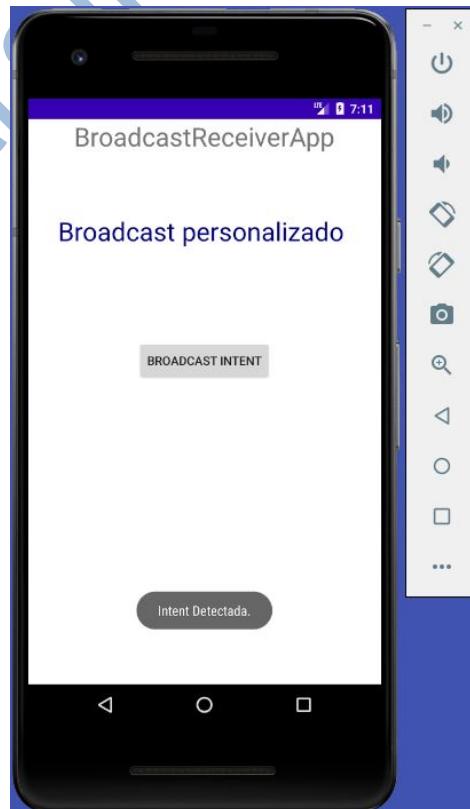
```

```
// broadcast intent customizado.  
fun broadcastIntent(view: View?) {  
    val intent = Intent()  
    intent.action = "com.example.broadcastreleckotlin.CUSTOM_INTENT"  
    sendBroadcast(intent)  
}  
}
```

Este é o arquivo AndroidManifest.xml para este projeto. Faça as alterações necessáris como se segue abaixo:

```
<activity android:name=".MainActivity">  
    <intent-filter>  
        <action android:name="android.intent.action.MAIN" />  
  
        <category android:name="android.intent.category.LAUNCHER"  
    />  
    </intent-filter>  
</activity>  
  
<receiver android:name=".MyReceiver" >  
    <intent-filter>  
        <action  
    android:name="com.example.broadcastreleckotlin.CUSTOM_INTENT" />  
    </intent-filter>  
</receiver>  
</application>
```

Rode seu aplicativo. Confira o resultado.



## SQLite Utilizando Content Provider

O SQLite é um banco de dados SQL de código-fonte aberto que armazena dados em um arquivo de texto em um dispositivo. O Android vem com a implementação de banco de dados SQLite embutida.

O SQLite suporta todos os recursos da base de dados relacionais. Para acessar este banco de dados, você não precisa estabelecer nenhum tipo de conexões para ele, como JDBC, ODBC e.t.c.

### Banco de dados - Package

O pacote principal é android.database.sqlite que contém as classes para gerenciar seus próprios bancos de dados.

### Banco de Dados - Criação

Para criar um banco de dados, é necessário apenas chamar o método openOrCreateDatabase com database name e modo, como parâmetro. Ele retorna uma instância do banco de dados SQLite que você deve receber em seu próprio objeto. Sua sintaxe é dada abaixo:

```
QLiteDatabase mydatabase = openOrCreateDatabase("your database
name", MODE_PRIVATE, null);
```

Além disso, existem outras funções disponíveis no pacote de banco de dados, que faz esse trabalho. Eles estão listados abaixo:

Sr.No	Métodos e Descrição
1	<b>openDatabase(String path, SQLiteDatabase.CursorFactory factory, int flags, DatabaseErrorHandler errorHandler)</b>  Este método apenas abre o banco de dados existente com o modo de sinalização apropriado. O modo de sinalizadores comuns pode ser OPEN_READWRITE OPEN_READONLY

2	<b>openDatabase(String path, SQLiteDatabase.CursorFactory factory, int flags)</b>
	<p>É semelhante ao método acima, pois também abre o banco de dados existente, mas não define nenhum manipulador para lidar com os erros dos bancos de dados</p>
3	<b>openOrCreateDatabase(String path, SQLiteDatabase.CursorFactory factory)</b> <p>Ele não só abre, mas crie o banco de dados se ele não existir. Esse método é equivalente ao método openDatabase.</p>
4	<b>openOrCreateDatabase(File file, SQLiteDatabase.CursorFactory factory)</b> <p>Este método é semelhante ao método acima, mas é preciso o objeto Arquivo como um caminho, em vez disso, em uma seqüência de caracteres. É equivalente a file.getPath()</p>

### Banco de Dados - Inserção

É possível criar tabelas ou inserir dados na tabela usando o método execSQL definido na classe SQLiteDatabase. Sua sintaxe é dada abaixo:

```
mydatabase.execSQL("CREATE TABLE IF NOT EXISTS YourDataBaseName(Username
VARCHAR,Password VARCHAR);");
mydatabase.execSQL("INSERT INTO YourDataBaseName VALUES('admin','admin');");

```

Isso irá inserir alguns valores na tabela dentro do banco de dados. Outro método que também faz o mesmo trabalho, mas leva algum parâmetro adicional, é dado abaixo:

Métodos e Descrição	
1	<b>execSQL(String sql, Object[] bindArgs)</b>

Este método não apenas insere dados, mas também é usado para atualizar ou modificar dados já existentes no banco de dados usando argumentos de ligação

## Banco de dados - Fetching

É possível recuperar qualquer coisa do banco de dados usando um objeto da classe Cursor. O método é chamado desta classe - chamado rawQuery - e retornará um conjunto de resultados com o cursor apontando para a tabela. É possível mover o cursor para a frente e recuperar os dados.

```
Cursor resultSet = mydatabase.rawQuery("Select * from YourDataBaseName",null);
resultSet.moveToFirst();
String username = resultSet.getString(0);
String password = resultSet.getString(1);
```

Existem outras funções disponíveis na classe Cursor que nos permite efetivamente recuperar os dados. Isso inclui:

	Métodos e Descrição
1	<b>getColumnCount()</b> Este método retorna o número total de colunas da tabela.
2	<b>getColumnIndex(String columnName)</b> Este método retorna o número de índice de uma coluna, especificando o nome da coluna
3	<b>getColumnName(int columnIndex)</b> Este método retorna o nome da coluna especificando o índice da coluna
4	<b>getColumnNames()</b> Esse método retorna a matriz de todos os nomes das colunas da tabela.
5	<b>getCount()</b>

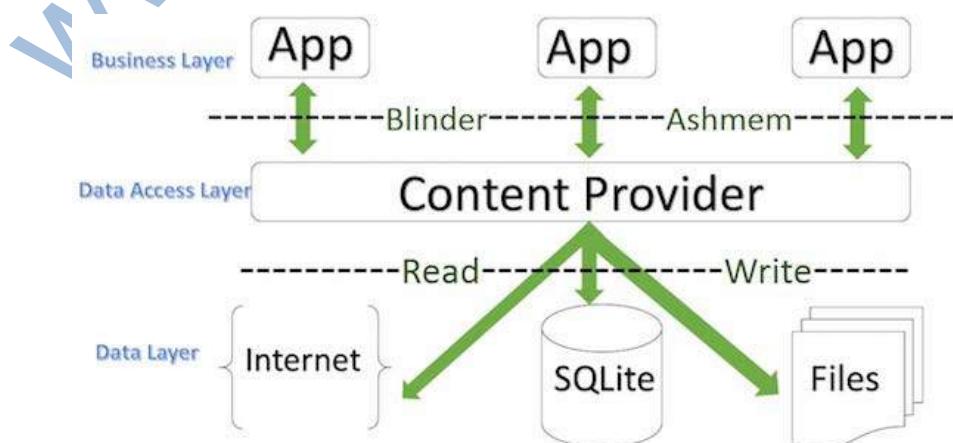
	Esse método retorna a matriz de todos os nomes das colunas da tabela.
6	<b>getPosition()</b> Este método retorna a posição atual do cursor na tabela
7	<b>isClosed()</b> Esse método retorna verdadeiro se o cursor estiver fechado e retornar falso caso contrário

### Base de Dados – Helper Class

Para gerenciar todas as operações relacionadas ao banco de dados, uma classe auxiliar foi fornecida e é chamada `SQLiteOpenHelper`. Ele gerencia automaticamente a criação e atualização do banco de dados. Sua sintaxe é dada abaixo:

#### Content Provider

Um componente de provedor de conteúdo fornece dados de um aplicativo para outros, mediante solicitação. Esses pedidos são tratados pelos métodos da classe `ContentResolver`. Um fornecedor de conteúdo pode usar diferentes maneiras de armazenar seus dados e os dados podem ser armazenados em um banco de dados, em arquivos ou mesmo em uma rede.



Às vezes é necessário compartilhar dados entre aplicativos. É aqui que os provedores de conteúdo se tornam muito úteis.

Os provedores de conteúdo permitem centralizar o conteúdo em um só lugar em que muitas aplicações diferentes acessem, conforme necessário. Um Content Provider se comporta muito como um banco de dados onde é possível consultar, editar seu conteúdo, além de adicionar ou excluir conteúdo usando os métodos `insert ()`, `update ()`, `delete ()` e `query ()`. Na maioria dos casos, esses dados são armazenados em um banco de dados SQLite.

Um Content Provider é implementado como uma subclasse da classe `ContentProvider` e deve implementar um conjunto padrão de APIs que permitem que outros aplicativos executem transações.

### **Content URIs**

Para consultar um provedor de conteúdo, é necessário especificar a seqüência de consulta na forma de um URI que tenha o seguinte formato:

```
<prefix>://<authority>/<data_type>/<id>
```

Aqui estão alguns detalhes de várias partes do URI:

Detalhes e Descrição	
1	<b>prefix</b> Isso sempre é configurado para o conteúdo: //
2	<b>authority</b> Isso especifica o nome do provedor de conteúdo, por exemplo, contatos, navegador, etc. Para provedores de conteúdo de terceiros, este pode ser o nome totalmente qualificado, como <i>teste.app.com.sqlitecontprovapp.StudentsProvider</i>
3	<b>data_type</b>

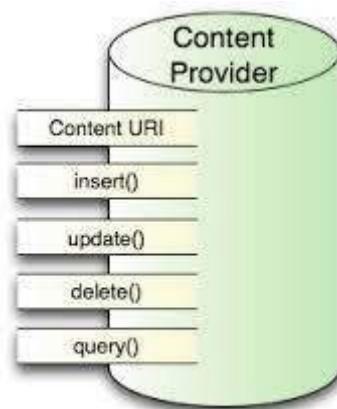
	<p>Isso indica o tipo de dado fornecido por esse provedor particular. Por exemplo, se você estiver obtendo todos os contatos do provedor de conteúdo de contatos, o caminho de dados seria people e o URI se pareceria com esse conteúdo: // contacts / people</p>
4	<p><b>id</b></p> <p>Isso especifica o registro específico solicitado. Por exemplo, se você estiver procurando o contato número 5 no provedor de conteúdo de contatos, o URI ficaria assim <i>content://contacts/people/5</i>.</p>

## Criar Content Provider

Isso envolve um número de etapas simples para criar seu próprio provedor de conteúdo.

- Antes de tudo, é necessário criar uma classe de Provedor de Conteúdo que estenda o ContentProviderbaseclass.
- Em segundo lugar, é necessário definir seu endereço URI do provedor de conteúdo que será usado para acessar o conteúdo.
- Em seguida, será preciso criar seu próprio banco de dados para manter o conteúdo. Geralmente, o Android usa o banco de dados SQLite e a estrutura precisa substituir o método onCreate () que usará o método SQLite Open Helper para criar ou abrir o banco de dados do provedor. Quando seu aplicativo é lançado, o manipulador onCreate () de cada um dos seus Provedores de Conteúdo é chamado no segmento principal do aplicativo.
- Em seguida, será preciso implementar consultas do Content Provider para executar diferentes operações específicas do banco de dados.
- Finalmente, registre o Content Provider no seu arquivo activity usando a tag <provider>.

Aqui está a lista de métodos necessária para substituir na classe Content Provider para que seu Content Provider esteja funcionando:



### **Content Provider**

- `onCreate ()` Este método é chamado quando o provedor é iniciado.
- `query ()` Este método recebe uma solicitação de um cliente. O resultado é retornado como um objeto Cursor.
- `insert ()` Este método insere um novo registro no provedor de conteúdo.
- `delete ()` Este método exclui um registro existente do provedor de conteúdo.
- `update ()` Este método atualiza um registro existente do provedor de conteúdo.
- `getType ()` Este método retorna o tipo MIME dos dados no URI fornecido.

### **Projeto SQLite Content Provider**

Este exercício mostrará como criar seu próprio ContentProvider.

	<b>Descrição</b>
1	Você usará Android Studio IDE para criar um aplicativo Android e nomeá-lo como SQLiteContProvApp.

2	Modificar o arquivo <i>MainActivity.java</i> e adicionar dois novos métodos <i>onClickAddName()</i> e <i>onClickRetrieveStudents()</i> .
3	Criar um novo arquivo java chamado <i>StudentsProvider.java</i> dentro de <i>teste.app.com.sqlitecontprovapp</i> e defini o Content Provider e os métodos necessarios.
4	Registrar o content provider no arquivo <i>AndroidManifest.xml</i> usando <i>&lt;provider.../&gt;</i> tag
5	Modificar o conteúdo do arquivo <i>res/layout/activity_main.xml</i> para incluir os components de IU necessarios.
6	Execute o aplicativo e verifique o resultado das alterações.

Este é o conteúdo do arquivo ***res/layout/activity\_main.xml***. Implemente o código abaixo:

```

<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="SQLiteContent provider App"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:textSize="30dp" />

    <TextView
        android:id="@+id/textView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Recuperar e dispor ao Content Provider "
        android:textColor="#000080"
        android:textSize="15dp"
        android:layout_below="@+id/textView1"
        android:layout_centerHorizontal="true" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/button2"
        android:layout_alignBottom="@+id/textView2"
        android:layout_centerHorizontal="true" />

```

```

        android:text="Insira o Nome"
        android:onClick="onClickAddName"
        android:layout_above="@+id/button"
        android:layout_centerHorizontal="true"
        android:layout_marginBottom="24dp" />

<EditText
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/editText2"
    android:hint="Name"
    android:textColorHint="@android:color/holo_blue_light"
    android:layout_above="@+id/editText3"
    android:layout_alignLeft="@+id/editText3"
    android:layout_alignStart="@+id/editText3"
    android:layout_marginBottom="24dp"
    android:layout_alignParentRight="true"
    android:layout_alignParentEnd="true" />

<EditText
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/editText3"
    android:hint="Disciplina"
    android:textColorHint="@android:color/holo_blue_bright"
    android:layout_marginBottom="62dp"
    android:layout_above="@+id/button2"
    android:layout_alignLeft="@+id/textView1"
    android:layout_alignStart="@+id/textView1"
    android:layout_alignParentRight="true"
    android:layout_alignParentEnd="true" />

<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Recuperar estudante"
    android:id="@+id/button"
    android:onClick="onClickRetrieveStudents"
    android:layout_alignParentBottom="true"
    android:layout_centerHorizontal="true"
    android:layout_marginBottom="57dp" />
</RelativeLayout>
```

Crie o novo arquivo dentro de java/nome do pacote e nomeie como  
**StudentsProvider.kt**. Implemente o código abaixo:

```

import android.content.*
import android.database.Cursor
import android.database.SQLException
import android.database.sqlite.SQLiteDatabase
import android.database.sqlite.SQLiteOpenHelper
import android.database.sqlite.SQLiteQueryBuilder
import android.net.Uri
import android.text.TextUtils
import java.util.*

class StudentsProvider() : ContentProvider() {
```

```

companion object {
    val PROVIDER_NAME =
        "teste.app.com.sqlitecontprovapp.StudentsProvider"
    val URL = "content://" + PROVIDER_NAME + "/students"
    val CONTENT_URI = Uri.parse(URL)
    val _ID = "_id"
    val NAME = "name"
    val GRADE = "grade"
    private val STUDENTS_PROJECTION_MAP: HashMap<String, String>?
        = null
    val STUDENTS = 1
    val STUDENT_ID = 2
    var uriMatcher: UriMatcher? = null
    val DATABASE_NAME = "College"
    val STUDENTS_TABLE_NAME = "students"
    val DATABASE_VERSION = 1
    val CREATE_DB_TABLE = " CREATE TABLE " + STUDENTS_TABLE_NAME +
        " (_id INTEGER PRIMARY KEY AUTOINCREMENT, " +
        " name TEXT NOT NULL, " +
        " grade TEXT NOT NULL);"

    init {
        uriMatcher = UriMatcher(UriMatcher.NO_MATCH)
        uriMatcher!!.addURI(PROVIDER_NAME, "students", STUDENTS)
        uriMatcher!!.addURI(PROVIDER_NAME, "students/#",
            STUDENT_ID)
    }
}

/**
 * Constantes específicas do Banco de Dados SQLite
 */
private var db: SQLiteDatabase? = null

/**
 * Helper class que cria e gerencia o repositorio de dados
 Provider
 *
 */
private class DatabaseHelper internal constructor(context:
Context?) :
    SQLiteOpenHelper(context, DATABASE_NAME, null,
DATABASE_VERSION) {
    override fun onCreate(db: SQLiteDatabase) {
        db.execSQL(CREATE_DB_TABLE)
    }

    override fun onUpgrade(db: SQLiteDatabase, oldVersion: Int,
newVersion: Int) {
        db.execSQL("DROP TABLE IF EXISTS " + STUDENTS_TABLE_NAME)
        onCreate(db)
    }
}

override fun onCreate(): Boolean {
    val context = context
    val dbHelper = DatabaseHelper(context)
    /**
     * Cria um banco de dados com capacidade de gravação que
     desencadeie sua
     * criação se ainda não existir.

```

```

        */
        db = dbHelper.writableDatabase
        return if (db == null) false else true
    }

    override fun insert(uri: Uri, values: ContentValues?): Uri? {
        /**
         * Adiciona uma nova gravação de dados Estudante
         */
        val rowID = db!!.insert(STUDENTS_TABLE_NAME, "", values)
        /**
         * Aqui, se a gravação foi adicionada com sucesso
         */
        if (rowID > 0) {
            val _uri = ContentUris.withAppendedId(CONTENT_URI, rowID)
            context!!.contentResolver.notifyChange(_uri, null)
            return _uri
        }
        throw SQLException("Failed to add a record into $uri")
    }

    override fun query(
        uri: Uri, projection: Array<String>?,
        selection: String?, selectionArgs: Array<String>?, sortOrder:
        String?
    ): Cursor? {
        var sortOrder = sortOrder
        val qb = SQLiteQueryBuilder()
        qb.tables = STUDENTS_TABLE_NAME
        when (uriMatcher!!.match(uri)) {
            STUDENTS -> qb.projectionMap =
                STUDENTS_PROJECTION_MAP
            STUDENT_ID -> qb.appendWhere(_ID + " = " +
                uri.pathSegments[1])
            else -> {
            }
        }
        if (sortOrder == null || sortOrder == "") {
            /**
             * Por padrão, organiza o nome dos estudantes
             */
            sortOrder = NAME
        }
        val c = qb.query(
            db, projection, selection,
            selectionArgs, null, null, sortOrder
        )
        /**
         * registro que observar se um conteúdo de URI mudou
         */
        c.setNotificationUri(context!!.contentResolver, uri)
        return c
    }

    override fun delete(uri: Uri, selection: String?, selectionArgs:
    Array<String>?): Int {
        var count = 0
        when (uriMatcher!!.match(uri)) {
            STUDENTS -> count = db!!.delete(STUDENTS_TABLE_NAME,
            selection, selectionArgs)
            STUDENT_ID -> {

```

```

        val id = uri.pathSegments[1]
        count = db!!.delete(
            STUDENTS_TABLE_NAME,
            "_ID + " + id +
                if (!TextUtils.isEmpty(selection)) " AND
                ($selection)" else "",
                selectionArgs
            )
        }
        else -> throw IllegalArgumentException("Unknown URI $uri")
    }
    context!!.contentResolver.notifyChange(uri, null)
    return count
}

override fun update(
    uri: Uri, values: ContentValues?,
    selection: String?, selectionArgs: Array<String>?
): Int {
    var count = 0
    when (uriMatcher!!.match(uri)) {
        STUDENTS -> count = db!!.update(STUDENTS_TABLE_NAME,
values, selection, selectionArgs)
        STUDENT_ID -> count = db!!.update(
            STUDENTS_TABLE_NAME, values,
            "_ID + " + uri.pathSegments[1] +
                if (!TextUtils.isEmpty(selection)) " AND
                ($selection)" else "", selectionArgs
            )
        else -> throw IllegalArgumentException("Unknown URI $uri")
    }
    context!!.contentResolver.notifyChange(uri, null)
    return count
}

override fun getType(uri: Uri): String? {
    when (uriMatcher!!.match(uri)) {
        STUDENTS -> return
        "vnd.android.cursor.dir/vnd.example.students"
        STUDENT_ID -> return
        "vnd.android.cursor.item/vnd.example.students"
        else -> throw IllegalArgumentException("Unsupported URI:
$uri")
    }
}
}

```



Este é o conteúdo do arquivo de atividade principal modificado **MainActivity.kt**. Este arquivo pode incluir cada um dos métodos fundamentais do ciclo de vida. Foram adicionados dois novos métodos em `onClickAddName()` e `onClickRetrieveStudents()` para manipular a interação do usuário com o aplicativo. Implemente o código abaixo:

```

import android.app.Activity
import android.content.ContentValues
import android.net.Uri

```

```

import android.os.Bundle
import android.view.View
import android.widget.EditText
import android.widget.Toast

class MainActivity() : Activity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
    }

    fun onClickAddName(view: View?) {
        //record Adiciona uma nova gravação
        val values = ContentValues()
        values.put(StudentsProvider.NAME,
            (findViewById<View>(R.id.editText2) as
EditText).text.toString())
        values.put(StudentsProvider.GRADE,
            (findViewById<View>(R.id.editText3) as
EditText).text.toString())
        val uri = contentResolver.insert(
            StudentsProvider.CONTENT_URI, values)
        Toast.makeText(baseContext,
            uri.toString(), Toast.LENGTH_LONG).show()
    }

    fun onClickRetrieveStudents(view: View?) {
        //Recupera os dados gravados
        val URL =
"content://teste.app.com.sqlitecontprovapp.StudentsProvider"
        val students = Uri.parse(URL)
        val c = managedQuery(students, null, null, null, "name")
        if (c.moveToFirst()) {
            do {
                Toast.makeText(this,
                    c.getString(c.getColumnIndex(StudentsProvider._ID)) +
                    ", " +
                    c.getString(c.getColumnIndex(StudentsProvider.NAME)) +
                    ", " +
                    c.getString(c.getColumnIndex(StudentsProvider.GRADE))), +
                    Toast.LENGTH_SHORT).show()
            } while (c.moveToNext())
        }
    }
}

```



Este é o conteúdo modificado do arquivo AndroidManifest.xml. Aqui, foi adicionando a tag <provider ... /> para incluir o Content Provider:

```

<application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:roundIcon="@mipmap/ic_launcher_round"
    android:supportsRtl="true"
    android:theme="@style/Theme.DbSQLiteKotlin">

```

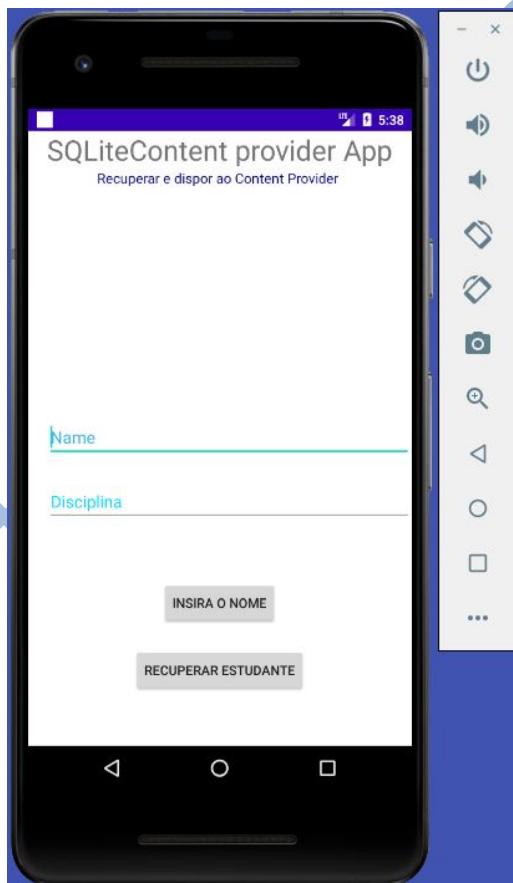
```
<activity android:name=".MainActivity">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />

        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>

<!-- declaração do Provider-->
<provider android:name="StudentsProvider"
    android:authorities="teste.app.com.sqlitecontprovapp.StudentsProvider"
/>

</application>
```

Rode seu aplicativo. Confira o resultado:



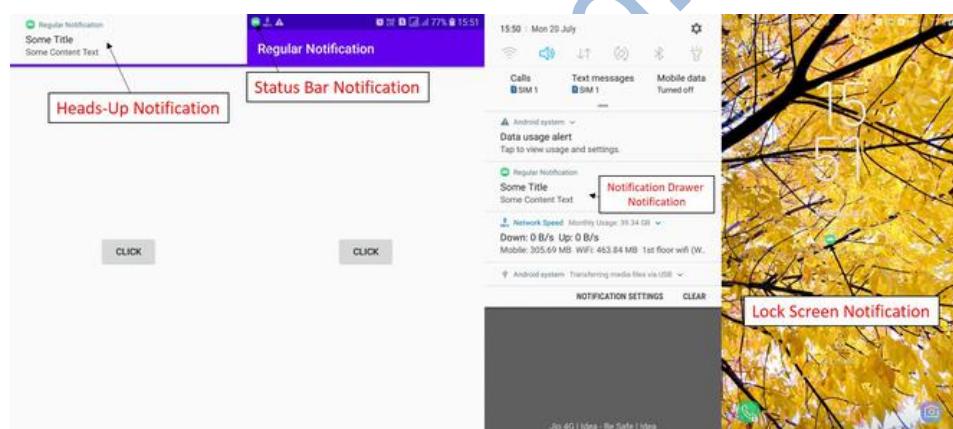
## Notification Android

Uma notificação é uma mensagem que você pode exibir para o usuário fora da UI normal do seu aplicativo. Quando você diz ao sistema que emita uma

notificação, primeiro aparece como um ícone na área de notificação. Para ver os detalhes da notificação, o usuário abre a gaveta de notificação. Tanto a área de notificação como a gaveta de notificação são áreas controladas pelo sistema que o usuário pode visualizar a qualquer momento.

As notificações podem ser de vários formatos e designs, dependendo do desenvolvedor. Em geral, deve-se ter testemunhado esses quatro tipos de notificações:

- Notificação da barra de status (aparece no mesmo layout da hora atual, porcentagem da bateria)
- Gaveta de notificação Notificação (aparece no menu suspenso)
- Notificação de alerta (aparece na tela de sobreposição, por exemplo: notificação do Whatsapp, mensagens OTP)
- Notificação de tela de bloqueio (acho que você sabe)



## Implementação passo a passo

### Etapa 1: Criar um novo projeto

Para criar um novo projeto no Android Studio, repetimos aqui os passos executados anteriormente para outros projetos; selecione Kotlin como a linguagem de programação.

### Etapa 2: Trabalhar com o arquivo **activity\_main.xml**

Acesse o arquivo **activity\_main.xml** e consulte o código a seguir. Nesta etapa, vamos projetar nossa página de layout. Aqui, usaremos o RelativeLayout

para obter a Visualização de rolagem do arquivo Kotlin. Abaixo está o código para o arquivo **activity\_main.xml**.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <Button
        android:id="@+id	btn"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true"
        android:text="Enviar Notificação" />

</RelativeLayout>
```

### Etapa 3: criar uma nova activity

Navegue até a pasta **java/nome.do.seu.pacote** e crie uma nova classe kotlin; nomeie a atividade como **NotificationView.kt**. Quando alguém clicar na notificação, esta activity será aberta em nosso aplicativo, e o usuário será redirecionado para esta página.

Implemento código abaixo:

```
import android.os.Bundle
import android.widget.TextView
import androidx.appcompat.app.AppCompatActivity

class NotificationView : AppCompatActivity() {
    var textView: TextView? = null
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_notification)
    }
}
```

No código acima, estamos chamando um novo arquivo de layout. Navegue até a pasta **java/res/layout** e crie-o. Nomeie como **activity\_notification.xml**. Abaixo está o código para implementar no arquivo recém-criado:

```

<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".NotificationView">

    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true"
        android:text="Olá! Essa é sua notificação funcionando"
        android:textSize="15sp"
        android:textStyle="bold" />

</RelativeLayout>

```



**Nota 1 :** Sem configurar Notification Channels, você não pode construir notificação para aplicativos com Android API > = 26. Para eles, a geração de um canal de notificação é obrigatória. Aplicativos com API <26 não precisam de canais de notificação, apenas precisam do criador de notificações. Cada canal deve ter um comportamento particular que será aplicável a todas as notificações que fazem parte dele. Cada canal terá, portanto, um ID de canal que basicamente atuará como um identificador exclusivo para este canal, o que será útil se o usuário quiser diferenciar um canal de notificação específico. Em contraste, o construtor Notification fornece uma maneira conveniente de definir os vários campos de uma Notificação e gerar visualizações de conteúdo usando o modelo de layout de notificação da plataforma, mas não é capaz de direcionar um canal de notificação específico.

**Nota 2:** Se você referiu qualquer outra documentação ou qualquer outro blog antes deste, você deve ter notado que eles apelam para implementar a seguinte dependência “**com.android.support:support-compat:28.0.0**”. O que experimentei pessoalmente é que não há necessidade de implementá-lo, as coisas vão longe e vão bem sem isso também.

### Etapa 5: Trabalhar com o arquivo MainActivity.kt

Acesse o arquivo **MainActivity.kt** e abaixo está o código para que deve ser implementado. Os comentários são adicionados dentro do código para entender o código em mais detalhes.

```

import android.app.Notification
import android.app.NotificationChannel
import android.app.NotificationManager
import android.app.PendingIntent
import android.content.Context
import android.content.Intent
import android.graphics.BitmapFactory
import android.graphics.Color
import android.os.Build
import android.os.Bundle
import android.widget.Button
import android.widget.RemoteViews
import androidx.appcompat.app.AppCompatActivity

class MainActivity : AppCompatActivity() {

```

```

// declaring variables
lateinit var notificationManager: NotificationManager
lateinit var notificationChannel: NotificationChannel
lateinit var builder: Notification.Builder
private val channelId = "i.apps.notifications"
private val description = "Test notification"

@Suppress("DEPRECATION")
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)

    // accessing button
    val btn = findViewById<Button>(R.id.btn)

    // it is a class to notify the user of events that happen.
    // This is how you tell the user that something has happened
    in the
    // background.
    notificationManager =
        getSystemService(Context.NOTIFICATION_SERVICE) as NotificationManager

    // onClick listener for the button
    btn.setOnClickListener {

        // pendingIntent is an intent for future use i.e after
        // the notification is clicked, this intent will come into
        action
        val intent = Intent(this, NotificationView::class.java)

        // FLAG_UPDATE_CURRENT specifies that if a previous
        // PendingIntent already exists, then the current one
        // will update it with the latest intent
        // 0 is the request code, using it later with the
        // same method again will get back the same pending
        // intent for future reference
        // intent passed here is to our afterNotification class
        val pendingIntent = PendingIntent.getActivity(this, 0,
            intent, PendingIntent.FLAG_UPDATE_CURRENT)

        // RemoteViews are used to use the content of
        // some different layout apart from the current activity
        layout
        val contentView = RemoteViews(packageName,
            R.layout.activity_notification)

        // checking if android version is greater than oreo(API
        26) or not
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
            notificationChannel = NotificationChannel(channelId,
                description, NotificationManager.IMPORTANCE_HIGH)
            notificationChannel.enableLights(true)
            notificationChannel.lightColor = Color.GREEN
            notificationChannel.enableVibration(false)

            notificationManager.createNotificationChannel(notificationChannel)

            builder = Notification.Builder(this, channelId)
                .setContent(contentView)
                .setSmallIcon(R.drawable.ic_launcher_background)
        }
    }
}

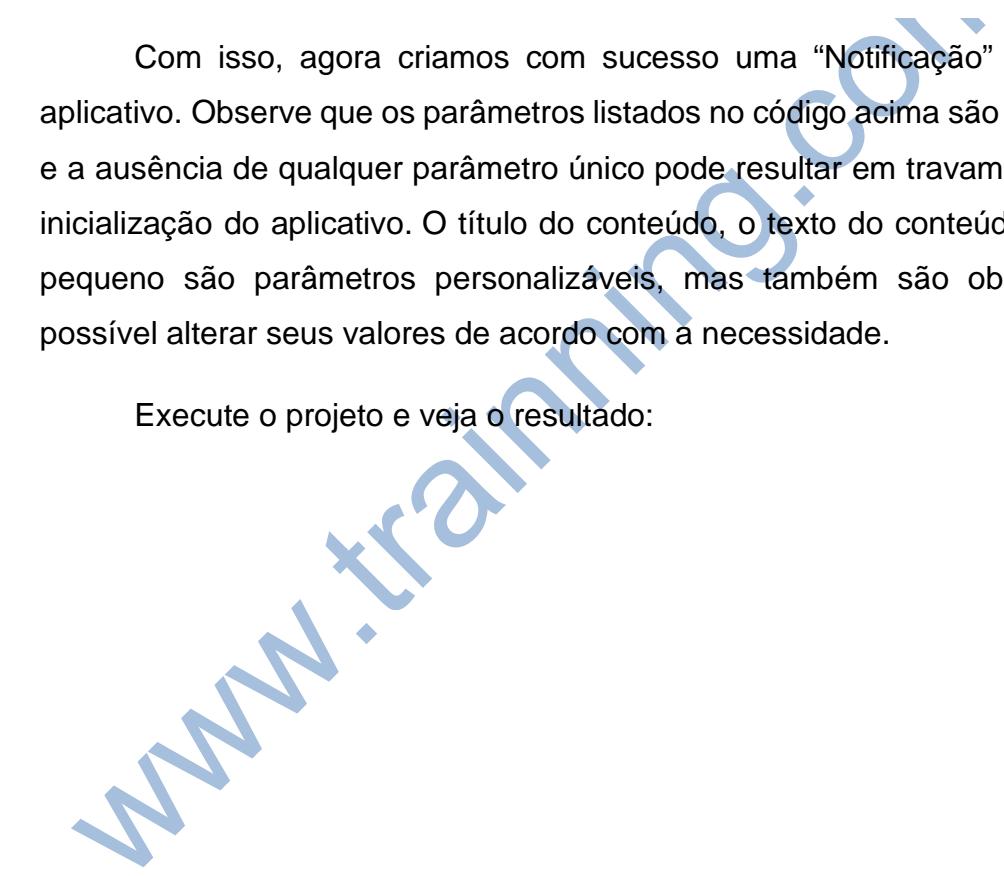
```

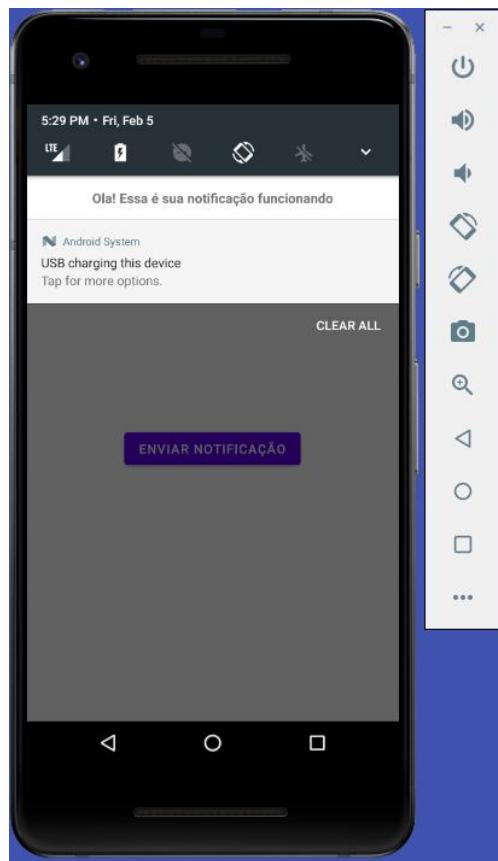
```
.setLargeIcon(BitmapFactory.decodeResource(this.resources,
R.drawable.ic_launcher_background))
    .setContentIntent(pendingIntent)
} else {
    builder = Notification.Builder(this)
        .setContent(contentView)
        .setSmallIcon(R.drawable.ic_launcher_background)

.setLargeIcon(BitmapFactory.decodeResource(this.resources,
R.drawable.ic_launcher_background))
    .setContentIntent(pendingIntent)
}
notificationManager.notify(1234, builder.build())
}
}
```

Com isso, agora criamos com sucesso uma “Notificação” para nosso aplicativo. Observe que os parâmetros listados no código acima são obrigatórios e a ausência de qualquer parâmetro único pode resultar em travamento ou não inicialização do aplicativo. O título do conteúdo, o texto do conteúdo e o ícone pequeno são parâmetros personalizáveis, mas também são obrigatórios. É possível alterar seus valores de acordo com a necessidade.

Execute o projeto e veja o resultado:

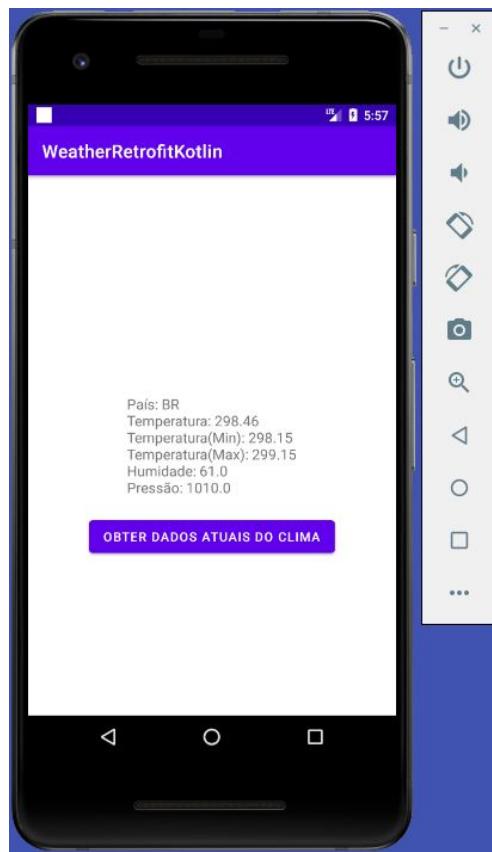




## Consumo de APIs - Retrofit

A biblioteca Retrofit é massivamente usada para fazer as chamadas http dentro da plataforma Android. O Retrofit é a melhor alternativa - em relação a biblioteca Volley - em termos de facilidade de uso, desempenho, extensibilidade e outras coisas. É um cliente REST tipo-seguro para o Android, construído pela Square. Usando esta ferramenta, o desenvolvedor Android pode tornar todas as coisas da rede/conexão muito mais fáceis.

Este é o aplicativo que será desenvolvido utilizando a biblioteca Retrofit.



Para este projeto, será usada uma API "<http://api.openweathermap.org/>". Para usar esta API, é necessário obter a API Key. Neste Link - "<http://api.openweathermap.org/>" é possível ver como obter a API Key. Em suma, é necessário se registrar e fazer login para obter a chave.

### Criando Novo Projeto

Crie um novo projeto no Android Studio a partir do File ⇒ New Project. Quando for solicitado a selecionar a activity padrão, selecione Empty Activity e continue.

Abra **build.gradle** e adicione as dependências Retrofit, Gson:

```
dependencies {
    implementation "org.jetbrains.kotlin:kotlin-stdlib:$kotlin_version"
    implementation 'androidx.core:core-ktx:1.3.2'
    implementation 'androidx.appcompat:appcompat:1.2.0'
    implementation 'com.google.android.material:material:1.2.1'
    implementation 'androidx.constraintlayout:constraintlayout:2.0.4'
    testImplementation 'junit:junit:4.+'
    androidTestImplementation 'androidx.test.ext:junit:1.1.2'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.3.0'
```

```
//dependencias inseridas
implementation 'com.squareup.retrofit2:retrofit:2.0.0'
implementation 'com.squareup.retrofit2:converter-gson:2.0.0'
}
```

Ao se trabalhar com operações/conexão de rede de internet, é necessário adicionar permissões de INTERNET no arquivo **AndroidManifest.xml**.

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

<application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:roundIcon="@mipmap/ic_launcher_round"
    android:supportsRtl="true"
    android:theme="@style/Theme.WeatherRetrofitKotlin">
    <activity android:name=".MainActivity">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />

            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
</application>
```

### Criando a interface WeatherService.kt

Primeiro, é preciso saber qual o tipo de resposta JSON que será recebida pelo aplicativo. Neste exercício, está sendo usado a API como exemplo e fonte de dados; essa API mostra como criar objetos Java que serão capazes de analisar os filmes mais recentes. Com base na resposta JSON retornada para esta chamada de API, será definido como uma representação básica:

Crie a interface e nomeie-a como **WeatherService.kt** dentro do pacote principal:

```
import retrofit2.Call
import retrofit2.http.GET
import retrofit2.http.Path
import retrofit2.http.Query

interface WeatherService {
    @GET("data/2.5/weather?")
    fun getCurrentWeatherData(@Query("lat") lat: String, @Query("lon") lon: String, @Query("APPID") app_id: String): Call<WeatherResponse>
}
```

Também será preciso criar uma carquivo kotlin chamado **WeatherResponse.kt**. Esta arquivo contém as informações oriundas da API. Crie o arquivo dentro do pacote do principal.

```
import com.google.gson.annotations.SerializedName

class WeatherResponse {

    @SerializedName( "coord" )
    var coord: Coord? = null
    @SerializedName( "sys" )
    var sys: Sys? = null
    @SerializedName( "weather" )
    var weather = ArrayList<Weather>()
    @SerializedName( "main" )
    var main: Main? = null
    @SerializedName( "wind" )
    var wind: Wind? = null
    @SerializedName( "rain" )
    var rain: Rain? = null
    @SerializedName( "clouds" )
    var clouds: Clouds? = null
    @SerializedName( "dt" )
    var dt: Float = 0.toFloat()
    @SerializedName( "id" )
    var id: Int = 0
    @SerializedName( "name" )
    var name: String? = null
    @SerializedName( "cod" )
    var cod: Float = 0.toFloat()
}
```

## Fazendo a Requisição

Vamos fazer o primeiro pedido da nossa MainActivity. Quando queremos consumir a API de forma assíncrona, o serviço é chamado da seguinte forma. Abra o **MainActivity.kt** e faça as alterações abaixo:

```
import android.graphics.Typeface
import android.os.Bundle
import android.view.View
import android.widget.TextView
import androidx.appcompat.app.AppCompatActivity

import retrofit2.Call
import retrofit2.Callback
import retrofit2.Response
import retrofit2.Retrofit
import retrofit2.converter.gson.GsonConverterFactory

class MainActivity : AppCompatActivity() {

    private var weatherData: TextView? = null
```

```

override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)

    weatherData = findViewById(R.id.textView)

    findViewById<View>(R.id.button).setOnClickListener {
        getCurrentData()
    }
}

internal fun getCurrentData() {
    val retrofit = Retrofit.Builder()
        .baseUrl(BaseUrl)
        .addConverterFactory(GsonConverterFactory.create())
        .build()
    val service = retrofit.create(WeatherService::class.java)
    val call = service.getCurrentWeatherData(lat, lon, AppId)
    call.enqueue(object : Callback<WeatherResponse> {
        override fun onResponse(call: Call<WeatherResponse>,
        response: Response<WeatherResponse>) {
            if (response.code() == 200) {
                val weatherResponse = response.body() !!

                val stringBuilder = "País: " +
                    weatherResponse.sys!! .country +
                    "\n" +
                    "Temperatura: " +
                    weatherResponse.main!! .temp +
                    "\n" +
                    "Temperatura(Min): " +
                    weatherResponse.main!! .temp_min +
                    "\n" +
                    "Temperatura(Max): " +
                    weatherResponse.main!! .temp_max +
                    "\n" +
                    "Humidade: " +
                    weatherResponse.main!! .humidity +
                    "\n" +
                    "Pressão: " +
                    weatherResponse.main!! .pressure

                weatherData!!.text = stringBuilder
            }
        }

        override fun onFailure(call: Call<WeatherResponse>, t:
        Throwable) {
            weatherData!!.text = t.message
        }
    })
}

companion object {

    var BaseUrl = "http://api.openweathermap.org/"
    var AppId = "e6fcc2c16925a31d1a28d86c19a445a2"
    var lat = "-23.62304"
    var lon = "-47.11994"
}
}

```

O Retrofit irá baixar e analisar os dados da API em um segmento de plano background e, em seguida, retornar os resultados ao segmento UI através do método onResponse ou onFailure.

Execute seu aplicativo. Confira o resultado.

www.trainning.com.br

## Referencias

<https://developer.android.com/guide/platform/index.html?hl=pt-br>  
<http://www.androidpro.com.br/recursos-do-sistema-android/>  
<http://computerworld.com.br/maioria-dos-apps-android-possui-arquitetura-incorrecta>  
<https://thiengo.com.br/mvp-android>  
<http://www.positivoinformatica.com.br/doseujeito/tecnologia/descubra-qual-e-a-sua-versao-do-android/>  
<http://www.tinmegali.com/en/model-view-presenter-android-part-1/>  
<http://www.tinmegali.com/en/model-view-presenter-mvp-in-android-part-2/>  
<http://www.tinmegali.com/en/model-view-presenter-mvp-in-android-part-3/>  
<https://guides.codepath.com/android/Architecture-of-Android-Apps#migrating-to-clean-architecture>  
[http://konmik.com/post/introduction\\_to\\_model\\_view\\_presenter\\_on\\_android/](http://konmik.com/post/introduction_to_model_view_presenter_on_android/)  
<http://thefinestartist.com/android/mvp-pattern>  
<https://antonioleiva.com/mvp-android/>  
[https://en.wikipedia.org/wiki/Separation\\_of\\_concerns](https://en.wikipedia.org/wiki/Separation_of_concerns)  
[https://en.wikipedia.org/wiki/Multitier\\_architecture](https://en.wikipedia.org/wiki/Multitier_architecture)  
<https://en.wikipedia.org/wiki/Model%20view%20controller>  
<https://en.wikipedia.org/wiki/Model%20view%20presenter>  
<http://antonioleiva.com/mvp-android/>  
<http://hannesdorfmann.com/android/mosby>  
<http://www.codeproject.com/Articles/288928/Differences-between-MVC-and-MVP-for-Beginners>  
<https://github.com/konmik/konmik.github.io/wiki/Introduction-to-Model-View-Presenter-on-Android>  
<https://en.wikipedia.org/wiki/Model%20view%20presenter>  
<https://developer.android.com/studio/intro/index.html?hl=pt-br>  
<https://developer.android.com/studio/intro/keyboard-shortcuts.html?hl=pt-br>  
<https://developer.android.com/studio/intro/update.html?hl=pt-br>  
<https://developer.android.com/studio/build/index.html?hl=pt-br>  
<https://developer.android.com/studio/build/shrink-code.html?hl=pt-br>  
<https://developer.android.com/studio/debug/index.html?hl=pt-br>  
<https://developer.android.com/studio/test/index.html?hl=pt-br>  
<https://developer.android.com/guide/topics/ui/declaring-layout.html?hl=pt-br>  
<https://developer.android.com/studio/projects/create-project.html?hl=pt-br>  
<https://developer.android.com/guide/topics/ui/controls.html?hl=pt-br>  
<https://developer.android.com/guide/topics/ui/layout/linear.html?hl=pt-br>

<http://www.tutorialsee.com/layout/linearLayout>

<http://www.tutorialsee.com/layout/relativeLayout>

<https://www.androidhive.info/2012/02/android-gridview-layout-tutorial/>

BY RAVI TAMADA

<https://www.androidhive.info/2011/10/android-listview-tutorial/>

By ravi tamada

<https://developer.android.com/training/material/lists-cards.html?hl=pt-br>

<https://www.androidhive.info/2016/01/android-working-with-recycler-view/>

by ravi tamada

<https://www.androidhive.info/2016/01/android-working-with-recycler-view/>

by ravi tamada

<https://material.io/guidelines/material-design/introduction.html?hl=pt-br#introduction-principles>

[https://pt.wikipedia.org/wiki/Plain\\_Old\\_Java\\_Objects](https://pt.wikipedia.org/wiki/Plain_Old_Java_Objects)

<https://www.androidhive.info/2015/04/android-getting-started-with-material-design/>  
por ravi tamada

<http://www.truiton.com/2015/06/android-tabs-example-fragments-viewpager/>  
por mohit gupt

<https://developer.android.com/training/material/lists-cards.html?hl=pt-br>  
<http://www.truiton.com/2015/03/android-cardview-example/>  
por mohit gupt

<http://www.truiton.com/2015/06/android-snackbar-example/>  
por mohit gupt

<https://www.androidhive.info/2015/12/android-material-design-floating-action-button/>  
por ravi tamada

<https://johncodeos.com/how-to-create-a-custom-navigation-drawer-in-android-using-kotlin/>

<https://www.geeksforgeeks.org/notifications-in-kotlin/>