# Etapa 3: atualize o código

1. Aberto MainActivity. Esta é a rollDice()aparência da função até agora:

private fun rollDice() {  
   val randomInt = (1..6).random()  
   val resultText: TextView = findViewById(R.id.result\_text)  
   resultText.text = randomInt.toString()  
}

1. Use findViewByID()para obter uma referência ao novo ImageViewno layout por ID ( R.id.dice\_image) e atribua essa visualização a uma nova diceImagevariável:

val diceImage: ImageView = findViewById(R.id.dice\_image)

val drawableResource = when (randomInt) {  
   1 -> R.drawable.dice\_1  
   2 -> R.drawable.dice\_2  
   3 -> R.drawable.dice\_3  
   4 -> R.drawable.dice\_4  
   5 -> R.drawable.dice\_5  
   else -> R.drawable.dice\_6  
}

[6. Tarefa: entender os níveis de API e compatibilidade](https://developer.android.com/codelabs/kotlin-android-training-images-compat#5)

Uma das melhores coisas sobre o desenvolvimento para Android é o grande número de dispositivos em que seu código pode ser executado - do Nexus One ao Pixel, a fatores de forma como tablets, Pixelbooks, relógios, TVs e carros.

Ao escrever para o Android, você não escreve aplicativos completamente separados para cada um desses diferentes dispositivos - mesmo os aplicativos que são executados em formatos radicalmente diferentes, como relógios e TVs, podem compartilhar códigos. Mas ainda existem restrições e estratégias de compatibilidade das quais você precisa estar ciente para oferecer suporte a tudo isso.

Nesta tarefa, você aprenderá como direcionar seu aplicativo para níveis específicos de API do Android (versões) e como usar as bibliotecas do Android Jetpack para oferecer suporte a dispositivos mais antigos.

Etapa 1: **explore os níveis de API**

No codelab anterior, ao criar seu projeto, você indicou o nível de API Android específico que seu aplicativo deve suportar. O sistema operacional Android tem diferentes números de versão com nomes de guloseimas saborosas que estão em ordem alfabética. Cada versão do sistema operacional vem com novos recursos e funcionalidades. Por exemplo, o Android Oreo foi lançado com suporte para [aplicativos Picture-in-picture](https://developer.android.com/guide/topics/ui/picture-in-picture) , enquanto o Android Pie [introduziu Slices](https://developer.android.com/guide/slices/) . Os níveis de API correspondem às versões do Android. Por exemplo, API 19 corresponde ao Android 4.4 (KitKat).

Devido a uma série de fatores, incluindo o que o hardware pode suportar, se os usuários optam por atualizar seus dispositivos e se os fabricantes oferecem suporte a diferentes níveis de sistema operacional, os usuários inevitavelmente acabam com dispositivos que executam diferentes versões de sistema operacional.

Ao criar seu projeto de aplicativo, você especifica o nível mínimo de API que seu aplicativo suporta. Ou seja, você especifica a versão mais antiga do Android compatível com o seu aplicativo. Seu aplicativo também tem um nível para o qual é compilado e um nível ao qual se destina. Cada um desses níveis é um parâmetro de configuração nos arquivos de compilação do Gradle.

1. Expanda a pasta **Gradle Scripts** e abra o **arquivo build.gradle (Módulo: app)** .

Este arquivo define parâmetros de compilação e dependências específicas para o módulo de aplicativo. O **arquivo build.gradle (Project: DiceRoller)** define os parâmetros de construção para o projeto como um todo. Em muitos casos, seu módulo de app é o único módulo em seu projeto, então essa divisão pode parecer arbitrária. Mas se seu aplicativo se tornar mais complexo e você o dividir em várias partes, ou se seu aplicativo suportar plataformas como o Android watch, você poderá encontrar módulos diferentes no mesmo projeto. 2. Examine a androidseção na parte superior do build.gradlearquivo. (O exemplo abaixo não é a seção inteira, mas contém o que você está mais interessado neste codelab.)

android {  
   compileSdkVersion 28  
   defaultConfig {  
       applicationId "com.example.android.diceroller"  
       minSdkVersion 19  
       targetSdkVersion 28  
       versionCode 1  
       versionName "1.0"  
   }

1. Examine o compileSdkVersionparâmetro.

compileSdkVersion 28

Este parâmetro especifica o nível de API do Android que o Gradle deve usar para compilar seu aplicativo. Esta é a versão mais recente do Android que seu aplicativo pode suportar. Ou seja, seu aplicativo pode usar os recursos de API incluídos neste nível de API e inferior. Nesse caso, seu aplicativo oferece suporte à API 28, que corresponde ao Android 9 (Pie).

1. Examine o targetSdkVersionparâmetro, que está dentro da defaultConfigseção:

targetSdkVersion 28

Este valor é a API mais recente com a qual você testou seu aplicativo. Em muitos casos, é o mesmo valor que compileSdkVersion.

1. Examine o minSdkVersionparâmetro.

minSdkVersion 19

Este parâmetro é o mais importante dos três, pois determina a versão mais antiga do Android na qual seu aplicativo será executado. Dispositivos que executam o sistema operacional Android mais antigo do que este nível de API não podem executar seu aplicativo.

Escolher o nível mínimo de API para seu aplicativo pode ser um desafio. Defina o nível da API muito baixo e você perderá os recursos mais recentes do sistema operacional Android. Defina um valor muito alto e seu aplicativo só poderá ser executado em dispositivos mais novos.

Quando você configurar seu projeto e chegar ao ponto em que define o nível mínimo de API para seu aplicativo, clique em **Ajude-me a escolher** para ver a caixa de diálogo **Distribuição de versão** da **API** . A caixa de diálogo fornece informações sobre quantos dispositivos usam diferentes níveis de sistema operacional e recursos que foram adicionados ou alterados nos níveis de sistema operacional. Você também pode verificar as notas de versão da documentação do Android e o [painel](https://developer.android.com/about/dashboards/) , que contém mais informações sobre as implicações do suporte a diferentes níveis de API.

Etapa 2: **explore a compatibilidade**

Escrever para diferentes níveis de API do Android é um desafio comum que os desenvolvedores de aplicativos enfrentam, portanto, a equipe de estrutura do Android fez muito trabalho para ajudá-lo.

Em 2011, a equipe lançou a primeira biblioteca de suporte, uma biblioteca desenvolvida pelo Google que oferece classes compatíveis com versões anteriores e funções úteis. Em 2018, o Google anunciou o Android Jetpack, que é uma coleção de bibliotecas que inclui muitas das classes e funções anteriores da biblioteca de suporte, além de expandir a biblioteca de suporte.

1. Aberto MainActivity.
2. Observe que sua MainActivityclasse não se estende a partir de Activitysi mesma, mas a partir de AppCompatActivity.

class MainActivity : AppCompatActivity() {   
...

AppCompatActivity é uma classe de compatibilidade que garante que sua atividade tenha a mesma aparência em diferentes níveis de sistema operacional de plataformas.

1. Clique no símbolo **+** próximo à linha que começa com importpara expandir as importações de sua classe. Observe que a AppCompatActivityclasse é importada do androidx.appcompat.apppacote. O namespace para as bibliotecas do Android Jetpack é androidx.
2. Abra **build.gradle (Módulo: app)** e role para baixo até a seção de dependências.

dependencies {  
   implementation fileTree(dir: 'libs', include: ['\*.jar'])  
   implementation"org.jetbrains.kotlin:kotlin-stdlib-jdk7:$kotlin\_version"  
   implementation 'androidx.appcompat:appcompat:1.0.0-beta01'  
   implementation 'androidx.core:core-ktx:1.0.1'  
   implementation 'androidx.constraintlayout:constraintlayout:1.1.2'  
   testImplementation 'junit:junit:4.12'  
   androidTestImplementation 'androidx.test:runner:1.1.0-alpha4'  
   androidTestImplementation   
        'androidx.test.espresso:espresso-core:3.1.0-alpha4'  
}

Observe a dependência da appcompatbiblioteca, que faz parte androidxe contém a AppCompatActivityclasse.

**Dica:** em geral, se seu aplicativo *pode* usar uma classe de compatibilidade das bibliotecas Jetpack, ele *deve* usar uma dessas classes, porque essas classes fornecem suporte para o maior número possível de recursos e dispositivos.

Etapa 3: **adicionar compatibilidade para drawables vetoriais**

Você usará seu novo conhecimento sobre namespaces, Gradle e compatibilidade para fazer um ajuste final em seu aplicativo, que otimizará o tamanho do aplicativo em plataformas mais antigas.

1. Expanda a pasta **res** e expanda **drawable** . Clique duas vezes em uma das imagens de dados.

Como você aprendeu antes, todas as imagens de dados são, na verdade, arquivos XML que definem as cores e formas dos dados. Esses tipos de arquivos são chamados de *drawables vetoriais* . O bom dos drawables vetoriais em comparação aos formatos de imagem de bitmap como PNG é que os drawables vetoriais podem ser redimensionados sem perder qualidade. Além disso, um drawable vetorial é geralmente um arquivo muito menor do que a mesma imagem em formato de bitmap.

Uma coisa importante a se notar sobre drawables vetoriais é que eles são suportados na API 21 em diante. Mas o SDK mínimo do seu aplicativo é definido como API 19. Se você tentar seu aplicativo em um dispositivo ou emulador API 19, verá que o aplicativo parece ser compilado e executado perfeitamente. Então, como isso funciona?

Quando você cria seu aplicativo, o processo de compilação do Gradle gera um arquivo PNG de cada um dos arquivos vetoriais, e esses arquivos PNG são usados ​​em qualquer dispositivo Android abaixo de 21. Esses arquivos PNG extras aumentam o tamanho do seu aplicativo. Aplicativos desnecessariamente grandes não são ótimos - eles tornam os downloads mais lentos para os usuários e ocupam mais do espaço limitado de seus dispositivos. Aplicativos grandes também têm uma chance maior de serem desinstalados e de os usuários não conseguirem fazer o download ou cancelar os downloads desses aplicativos.

A boa notícia é que existe uma biblioteca de compatibilidade do Android X para drawables vetoriais desde o nível de API 7. 2. Abra **build.gradle (Módulo: app)** . Adicione esta linha à defaultConfigseção:

vectorDrawables.useSupportLibrary = true

1. Clique no botão **Sincronizar agora** . Cada vez que um build.gradlearquivo é modificado, você precisa sincronizar os arquivos de construção com o projeto.
2. Abra o activity\_main.xmlarquivo de layout. Adicione este namespace à <LinearLayout>tag raiz , abaixo do toolsnamespace:

xmlns:app="http://schemas.android.com/apk/res-auto"

O appnamespace é para atributos que vêm de seu código customizado ou de bibliotecas, e não da estrutura principal do Android.

1. Altere o android:srcatributo no <ImageView>elemento a ser app:srcCompat.

app:srcCompat="@drawable/empty\_dice"

O app:srcCompatatributo usa a biblioteca Android X para oferecer suporte a drawables vetoriais em versões mais antigas do Android, de volta ao nível 7 da API.

1. Crie e execute seu aplicativo. Você não verá nada diferente na tela, mas agora seu aplicativo não precisa usar arquivos PNG gerados para as imagens dos dados, não importa onde eles sejam executados, o que significa um arquivo de aplicativo menor.

## [8. Desafio de codificação](https://developer.android.com/codelabs/kotlin-android-training-images-compat#7)

**Observação:** todos os desafios de codificação são opcionais.

**Desafio:** Modifique o aplicativo DiceRoller para ter dois dados. Quando o usuário toca no botão **Lançar** , cada dado deve ter um valor independente do outro.

**Dica:** Crie uma nova função privada para obter uma imagem de drawable aleatória e retornar um número inteiro para o recurso drawable. Use essa função para cada uma das imagens de dados.

private fun getRandomDiceImage() : Int { ... }

## ****Código de solução de desafio de codificação****

Projeto Android Studio: [DiceRollerFinal-challenge](https://github.com/google-developer-training/android-kotlin-fundamentals-apps/tree/master/DiceRollerFinal-challenge" \t "_blank)

## [7. Código da solução](https://developer.android.com/codelabs/kotlin-android-training-images-compat#6)

Projeto Android Studio: [DiceRollerFinal](https://github.com/google-developer-training/android-kotlin-fundamentals-apps/tree/master/DiceRollerFinal" \t "_blank)

[9. Resumo](https://developer.android.com/codelabs/kotlin-android-training-images-compat#8)

Recursos do aplicativo:

* Os recursos do seu aplicativo podem incluir imagens e ícones, cores padrão usadas no aplicativo, strings e layouts XML. Todos esses recursos são armazenados na respasta.
* A drawablepasta de recursos é onde você deve colocar todos os recursos de imagem para seu aplicativo.

Usando drawables vetoriais em visualizações de imagens:

* Drawables vetoriais são imagens descritas no formato XML. Drawables de vetor são mais flexíveis do que imagens de bitmap (como arquivos PNG) porque podem ser redimensionados para qualquer tamanho ou resolução.
* Para adicionar um drawable ao layout do seu aplicativo, use um <ImageView>elemento. A fonte da imagem está no android:srcatributo. Para referir-se à pasta de recursos drawable, use @drawable, por exemplo "@drawable/image\_name".
* Use a ImageViewvisualização em seu MainActivitycódigo para a imagem. Você pode usar setImageResource()para alterar a imagem da visualização para um recurso diferente. Use R.drawablepara se referir a drawables específicos, por exemplo setImageResource(R.drawable.image\_name).

A lateinitpalavra-chave:

* Minimize as chamadas para findViewById()em seu código declarando os campos para manter essas visualizações e inicializando os campos em onCreate(). Use a lateinitpalavra - chave para o campo para evitar a necessidade de declará-lo anulável.

O toolsnamespace para atributos de tempo de design:

* Use o tools:srcatributo no <ImageView>elemento em seu layout para exibir uma imagem apenas na visualização ou no editor de design do Android Studio. Você pode então usar uma imagem vazia android:srcpara o aplicativo final.
* Use o toolsnamespace no arquivo de layout do Android para criar conteúdo de espaço reservado ou dicas para layout no Android Studio. Os dados declarados por toolsatributos não são usados ​​no aplicativo final.

Níveis de API:

* Cada sistema operacional Android tem um número e nome de versão oficial (por exemplo, Android 9.0, "Pie") e um nível de API (API 28). Use os níveis de API nos arquivos Gradle do seu aplicativo para indicar as versões do Android que seu aplicativo suporta.
* O compileSdkVersionparâmetro no build.gradlearquivo especifica o nível de API do Android que o Gradle deve usar para compilar seu aplicativo.
* O targetSdkVersionparâmetro especifica o nível de API mais recente em relação ao qual você testou seu aplicativo. Em muitos casos, esse parâmetro tem o mesmo valor que compileSdkVersion.
* O minSdkVersionparâmetro especifica o nível de API mais antigo em que seu aplicativo pode ser executado.

Android Jetpack:

* O Android Jetpack é uma coleção de bibliotecas, desenvolvida pelo Google, que oferece classes compatíveis com versões anteriores e funções úteis para oferecer suporte a versões mais antigas do Android. O Jetpack substitui e expande o conjunto de bibliotecas anteriormente conhecido como Android Support Library.
* As classes importadas do androidxpacote referem-se às bibliotecas Jetpack. As dependências do Jetpack em seu build.gradlearquivo também começam com androidx.

Compatibilidade com versões anteriores para drawables vetoriais:

* Drawables vetoriais são suportados apenas nativamente em versões do Android superiores à API 21. Em versões mais antigas, o Gradle gera imagens PNG para esses drawables quando seu aplicativo é criado.
* Você pode especificar que a Android Support Library deve ser usada para drawables vetoriais em versões mais antigas da API com o vectorDrawables.useSupportLibrary = trueparâmetro de configuração no build.gradlearquivo.
* Depois de habilitar a biblioteca de suporte para drawables vetoriais, use o app:srcCompatatributo no <ImageView>elemento (em vez de android:src) para especificar a origem do drawable vetorial para essa imagem.

O appnamespace:

* O appnamespace em seu arquivo de layout XML é para atributos que vêm de seu código customizado ou de bibliotecas, não da estrutura Android principal.

## [11. Lição de casa](https://developer.android.com/codelabs/kotlin-android-training-images-compat#10)

Esta seção lista as possíveis tarefas de casa para os alunos que estão trabalhando neste codelab como parte de um curso ministrado por um instrutor. Cabe ao instrutor fazer o seguinte:

* Atribua lição de casa, se necessário.
* Comunique aos alunos como enviar os trabalhos de casa.
* Avalie as tarefas de casa.

Os instrutores podem usar essas sugestões tão pouco ou quanto quiserem e devem se sentir à vontade para designar qualquer outro dever de casa que considerem adequado.

Se você estiver trabalhando neste codelab por conta própria, sinta-se à vontade para usar essas tarefas de lição de casa para testar seus conhecimentos.

## ****Alterar um aplicativo****

Adicione um botão **Limpar** ao aplicativo DiceRoller que define a imagem do dado de volta para a imagem vazia.

## ****Responda estas perguntas****

### **Questão 1**

Qual <ImageView>atributo indica uma imagem de origem que deve ser usada apenas no Android Studio?

* android:srcCompat
* app:src
* tools:src
* tools:sourceImage

### **Questão 2**

Qual método altera o recurso de imagem para um ImageViewem código Kotlin?

* setImageResource()
* setImageURI()
* setImage()
* setImageRes()

### **Questão 3**

O que a lateinitpalavra - chave em uma declaração de variável indica no código Kotlin?

* A variável nunca é inicializada.
* A variável só é inicializada no tempo de execução do aplicativo.
* A variável é inicializada automaticamente para null.
* A variável será inicializada posteriormente. Eu prometo!

### **Questão 4**

Qual configuração do Gradle indica o nível de API mais recente com o qual seu aplicativo foi testado?

* minSdkVersion
* compileSdkVersion
* targetSdkVersion
* testSdkVersion

### **Questão 5**

Você vê uma linha de importação em seu código que começa com androidx. O que isto significa?

* A classe faz parte das bibliotecas do Android Jetpack.
* A classe está em uma biblioteca externa que será carregada dinamicamente quando o aplicativo for executado.
* A aula é "extra" e opcional para sua classe.
* A aula faz parte do suporte XML do Android.

## Envie seu aplicativo para avaliação

Verifique se o aplicativo tem o seguinte:

* O layout do aplicativo deve incluir uma visualização de imagem e dois botões.
* O código do aplicativo deve definir dois manipuladores de clique, um para cada botão.
* O manipulador de cliques do botão **Limpar** deve definir a imagem do dado como R.drawable.empty\_dice.

[1. Bem-vindo](https://developer.android.com/codelabs/kotlin-android-training-available-resources#0)

Este codelab faz parte do curso Android Kotlin Fundamentals. Você obterá o máximo valor deste curso se trabalhar com os codelabs em sequência. Todos os codelabs do curso estão listados na [página de destino dos codelabs](https://codelabs.developers.google.com/android-kotlin-fundamentals/) do [Android Kotlin Fundamentals](https://codelabs.developers.google.com/android-kotlin-fundamentals/) .

Introdução

Neste codelab, você aprende sobre os recursos que são úteis para os desenvolvedores do Kotlin Android, incluindo modelos, documentação, vídeos e aplicativos de amostra.

O que você já deveria saber

* O fluxo de trabalho básico do Android Studio.
* Como usar o Layout Editor no Android Studio.

O que você aprenderá

* Onde encontrar informações e recursos para desenvolvedores Kotlin e Android.
* Como alterar o ícone do iniciador em um aplicativo.
* Como procurar ajuda ao criar aplicativos Android usando Kotlin.

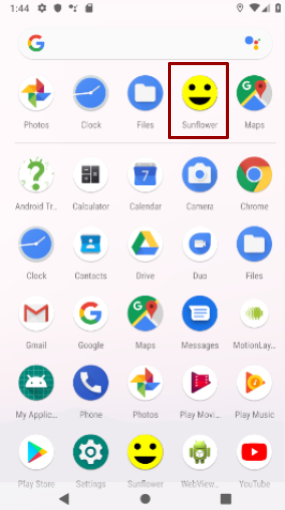
O que você vai fazer

* Explore alguns dos recursos disponíveis para desenvolvedores Kotlin Android de todos os níveis.
* Baixe e explore um aplicativo de amostra do Kotlin Android.
* Altere o ícone do iniciador de um aplicativo.

## [2. Visão geral do aplicativo](https://developer.android.com/codelabs/kotlin-android-training-available-resources#1)

Neste codelab, você aprende sobre modelos, exemplos, documentação e outros recursos que estão disponíveis para desenvolvedores Kotlin Android.

Primeiro, você cria um aplicativo simples a partir de um modelo do Android Studio e modifica o aplicativo. Em seguida, você baixa e explora o aplicativo de amostra Android Sunflower. Você substitui o ícone do iniciador do aplicativo de amostra (um girassol) por um recurso de imagem de clip-art que está disponível no Android Studio (um rosto sorridente).



## [3. Tarefa: Use modelos de projeto](https://developer.android.com/codelabs/kotlin-android-training-available-resources#2)

O Android Studio fornece modelos para projetos de atividades e aplicativos comuns e recomendados. Os modelos integrados economizam tempo e ajudam a seguir as melhores práticas de design.

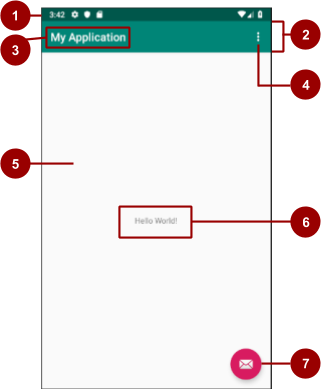
Cada modelo incorpora um esqueleto de atividade e interface de usuário. Você já usou o modelo de Atividade Vazia neste curso. O modelo de atividade básica tem mais recursos e incorpora recursos de aplicativos recomendados, como o menu de opções que aparece na barra de aplicativos em dispositivos Android.

## Etapa ****1: explorar a arquitetura de atividades básicas****

1. No Android Studio, crie um projeto.
2. Na caixa de diálogo **Escolha seu projeto** , selecione o modelo de Atividade Básica e clique em **Avançar** .
3. Na caixa de diálogo **Configure your project** , nomeie o aplicativo como quiser. Selecione **Kotlin** para o idioma e marque a caixa de seleção **Usar artefatos AndroidX** . Clique em **Concluir** .
4. Crie o aplicativo e execute-o em um emulador ou dispositivo Android.
5. Identifique as peças etiquetadas na figura e tabela abaixo. Encontre seus equivalentes na tela do seu dispositivo ou emulador. Inspecione o código Kotlin correspondente e os arquivos XML descritos na tabela.

Estar familiarizado com o código-fonte Kotlin e os arquivos XML o ajudará a estender e personalizar o modelo de atividade básica de acordo com suas necessidades.

### **Arquitetura do modelo de atividade básica**



|  |  |  |
| --- | --- | --- |
| **#** | **UI description** | **Referência de código** |
| 1 | Barra de status, que o sistema Android fornece e controla. | Não visível no código do modelo, mas você pode acessar a barra de status de sua atividade. Por exemplo, você pode adicionar código MainActivity.ktpara [ocultar a barra de status](http://developer.android.com/training/system-ui/status.html) , se necessário. |
| 2 | A barra de aplicativos, também chamada de barra de ação , fornece estrutura visual, elementos visuais padronizados e navegação. | Em activity\_main.xml, procure [Toolbar](http://developer.android.com/reference/android/support/v7/widget/Toolbar.html) dentro do elemento [AppBarLayout](http://developer.android.com/reference/android/support/design/widget/AppBarLayout.html" \t "_blank) . Para compatibilidade com versões anteriores, o AppBarLayoutno modelo incorpora um Toolbarque tem a mesma funcionalidade de uma [ActionBar](https://developer.android.com/reference/android/app/ActionBar.html" \t "_blank) . Para alterar a aparência da barra de aplicativos, altere os atributos da barra de ferramentas. Para tutoriais da barra de aplicativos, consulte [Adicionar a barra de aplicativos](https://developer.android.com/training/appbar/) . |
| 3 | O nome do aplicativo é inicialmente derivado do nome do seu pacote, mas você pode alterá-lo para o que quiser. | Em AndroidManifest.xml, procure android:label="@string/app\_name".Especifique a app\_namestring em strings.xml. |
| 4 | O botão flutuante do menu de opções contém itens de menu para a atividade. O botão de estouro também contém opções de menu globais, como **Pesquisa** e **Configurações** do aplicativo. Seus itens de menu de aplicativo vão para este menu. | Em MainActivity.kt, o onOptionsItemSelected()método implementa o que acontece quando o usuário seleciona um item de menu. Para ver os itens do menu de opções, abra o res/menu/menu\_main.xmlarquivo. Apenas o item de menu **Configurações** é especificado neste modelo. |
| 5 | O [CoordinatorLayout](http://developer.android.com/reference/android/support/design/widget/CoordinatorLayout.html" \t "_blank) ViewGroup é um layout que fornece mecanismos para que os elementos da IU interajam. A IU do seu aplicativo vai dentro do content\_main.xmlarquivo, que está incluído nele ViewGroup. | Em activity\_main.xml, procure a include layoutinstrução. Nenhuma vista é especificada neste layout; em vez disso, o layout inclui o content\_mainlayout, onde as visualizações do aplicativo são especificadas. As visualizações do sistema são mantidas separadas das visualizações exclusivas do seu aplicativo. |
| 6 | O modelo usa um [TextView](http://developer.android.com/reference/android/widget/TextView.html" \t "_blank) para exibir "Hello World". Você substitui isso TextViewpelos elementos de IU do seu aplicativo. | A visualização de texto "Hello World" está no content\_main.xmlarquivo. Todos os elementos de IU do seu aplicativo devem ser definidos neste arquivo. |
| 7 | Botão de ação flutuante (FAB) | Em activity\_main.xml, procure o FloatingActionButtonelemento. O FAB é definido como um elemento de interface do usuário usando um ícone de clip-art. MainActivity.ktinclui um esboço onCreate()que define um onClick()ouvinte para o FAB. |

## ****Etapa 2: personalizar o aplicativo que o modelo produz****

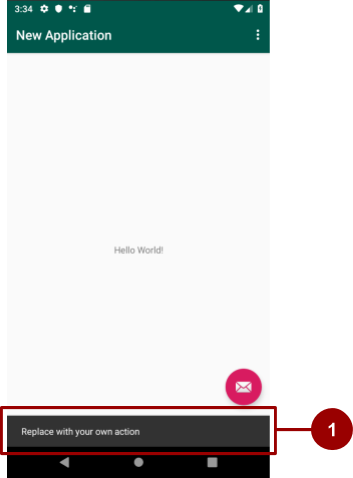
Altere a aparência do aplicativo produzido pelo modelo de atividade básica. Por exemplo, você pode alterar a cor da barra de aplicativos para coincidir com a barra de status. (Em alguns dispositivos, a barra de status é um tom mais escuro da mesma cor primária que a barra de aplicativos usa.)

1. Altere o nome do aplicativo que a barra de aplicativos exibe. Para fazer isso, altere o app\_namerecurso de string no arquivo **res> values> strings.xml** para o seguinte:

<string name="app\_name">New Application</string>

1. Altere a cor da barra de aplicativos ( Toolbar) em **res> layout> activity\_main.xml** alterando o android:backgroundatributo para "?attr/colorPrimaryDark". Este valor define a cor da barra do aplicativo para uma cor primária mais escura que corresponde à barra de status:

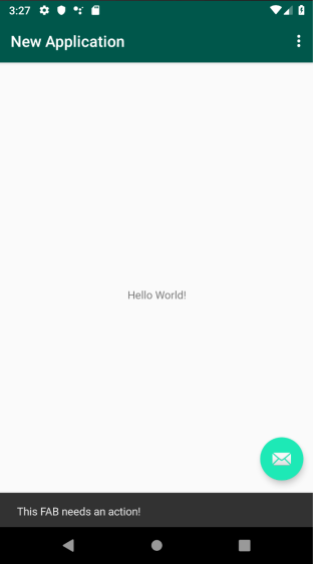
android:background="?attr/colorPrimaryDark"

1. Execute o aplicativo. O novo nome do aplicativo aparece na barra de status e a cor de fundo da barra de aplicativos é mais escura e corresponde à cor da barra de status. Quando você clica no FAB, uma barra de ferramentas aparece, mostrada como 1 na imagem abaixo.
2. Altere o texto da barra de ferramentas. Para fazer isso, abra MainActivitye procure o código de stub onCreate()que define um onClick()ouvinte para o botão. Mude "Replace with your own action"para outra coisa. Por exemplo:

fab.setOnClickListener { view ->  
   Snackbar.make(view, "This FAB needs an action!", Snackbar.LENGTH\_LONG)  
       .setAction("Action", null).show()  
}

1. O FAB usa a cor de destaque do aplicativo, portanto, uma maneira de alterar a cor do FAB é alterando a cor de destaque. Para alterar a cor de **destaque** , abra o arquivo **res> values> colors.xml** e altere o colorAccentatributo, conforme mostrado abaixo. (Para obter ajuda na escolha das cores, consulte o [sistema de cores](https://material.io/design/color/the-color-system.html) do [Material Design](https://material.io/design/color/the-color-system.html) .)

<color name="colorAccent">#1DE9B6</color>

1. Execute o aplicativo. O FAB usa a nova cor e o texto da barra de ferramentas mudou.

**Dica:** Para obter detalhes sobre a sintaxe XML para acessar recursos, consulte [Acessando seus recursos de aplicativo](https://developer.android.com/guide/topics/resources/providing-resources#Accessing) .

## Etapa ****3: explore como adicionar atividades usando modelos****

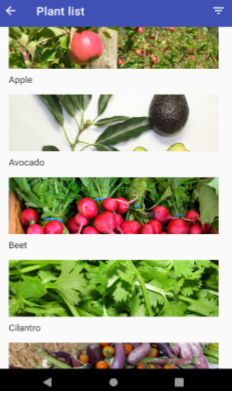
Para os codelabs deste curso até agora, você usou os modelos Empty Activity e Basic Activity para iniciar novos projetos. Você também pode usar modelos de atividades ao criar atividades após a criação do projeto.

1. Crie um projeto de aplicativo ou escolha um projeto existente.
2. No painel **Projeto> Android** , clique com o botão direito na pasta **java** .
3. Selecione **Novo> Atividade> Galeria** .
4. Adicione uma atividade ao aplicativo selecionando um dos Activitymodelos. Por exemplo, selecione **Atividade de gaveta de navegação** para adicionar um Activityque tenha uma gaveta de navegação.
5. Para exibir a atividade no editor de layout, clique duas vezes no arquivo de layout da atividade (por exemplo activity\_main2.xml). Use a guia **Design** e a guia **Texto** para alternar entre a visualização do layout da atividade e o código do layout.

## [4. Tarefa: Aprenda com o código de amostra](https://developer.android.com/codelabs/kotlin-android-training-available-resources#3)

Os repositórios de [amostras do Google](https://github.com/googlesamples) no GitHub fornecem amostras de código Kotlin Android que você pode estudar, copiar e incorporar em seus projetos.

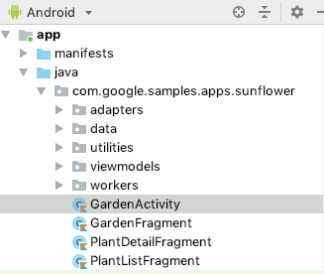
## Etapa ****1: faça download e execute uma amostra de código Kotlin Android****

1. Em um navegador, navegue até [github.com/android](https://github.com/android) .
2. Para **Idioma** , selecione **Kotlin** .
3. Selecione um aplicativo de amostra do Kotlin Android que foi modificado recentemente e baixe o código do projeto do aplicativo. Para este exemplo, baixe o arquivo zip para o aplicativo [android-sunflower](https://github.com/android/android-sunflower" \t "_blank) , que demonstra alguns dos componentes do Android Jetpack.
4. No Android Studio, abra o projeto android-sunflower-master.
5. Aceite todas as atualizações recomendadas pelo Android Studio e execute o aplicativo em um emulador ou dispositivo Android. 

**Observação:** as amostras contidas no repositório de amostras do Google no GitHub são um ponto de partida para desenvolvimento posterior. Nós o encorajamos a projetar e construir suas próprias ideias nesses exemplos.

## ****Etapa 2: explorar um exemplo de código Kotlin Android****

Agora que você tem o aplicativo de amostra Android Sunflower aberto no Android Studio, aprenda sobre o aplicativo e explore seus arquivos de projeto.

1. Para obter informações sobre o que um aplicativo de amostra está demonstrando, visite o arquivo README do aplicativo no GitHub. Para este exemplo, consulte o [README](https://github.com/googlesamples/android-sunflower#readme) do [Android Sunflower](https://github.com/googlesamples/android-sunflower#readme) .
2. No Android Studio, abra um dos arquivos de atividade Kotlin no aplicativo, por exemplo GardenActivity.kt.
3. Em GardenActivity.kt, encontre uma classe, tipo ou procedimento com o qual você não está familiarizado e procure na documentação do desenvolvedor Android. Por exemplo, para saber mais sobre o setContentView()método, pesquise em developer.android.com para encontrar [setContentView()](https://developer.android.com/reference/android/app/Activity" \l "setContentView(android.view.View,%20android.view.ViewGroup.LayoutParams)" \t "_blank).

## Etapa 3: alterar o ícone do iniciador

Nesta etapa, você altera o ícone do iniciador para o aplicativo de amostra Android Sunflower. Você adiciona uma imagem de clip-art e a usa para substituir o ícone atual do iniciador Android Sunflower.

### **Ícones do iniciador**

Cada aplicativo que você cria com o Android Studio começa com um ícone de iniciador padrão que representa o aplicativo. Os ícones do iniciador às vezes são chamados de ícones de aplicativos ou ícones de produtos .

Se você publicar um aplicativo no Google Play, o ícone do iniciador do aplicativo aparecerá na lista do aplicativo e nos resultados de pesquisa na Google Play Store.

Depois que um aplicativo é instalado em um dispositivo Android, o ícone do iniciador do aplicativo aparece na tela inicial do dispositivo e em qualquer outro lugar no dispositivo. Por exemplo, o ícone do iniciador do aplicativo Android Sunflower aparece na janela **Search Apps** do dispositivo , mostrado como 1 na captura de tela abaixo. O ícone do iniciador padrão, mostrado como 2 abaixo, é usado inicialmente para todos os projetos de aplicativos que você cria no Android Studio.

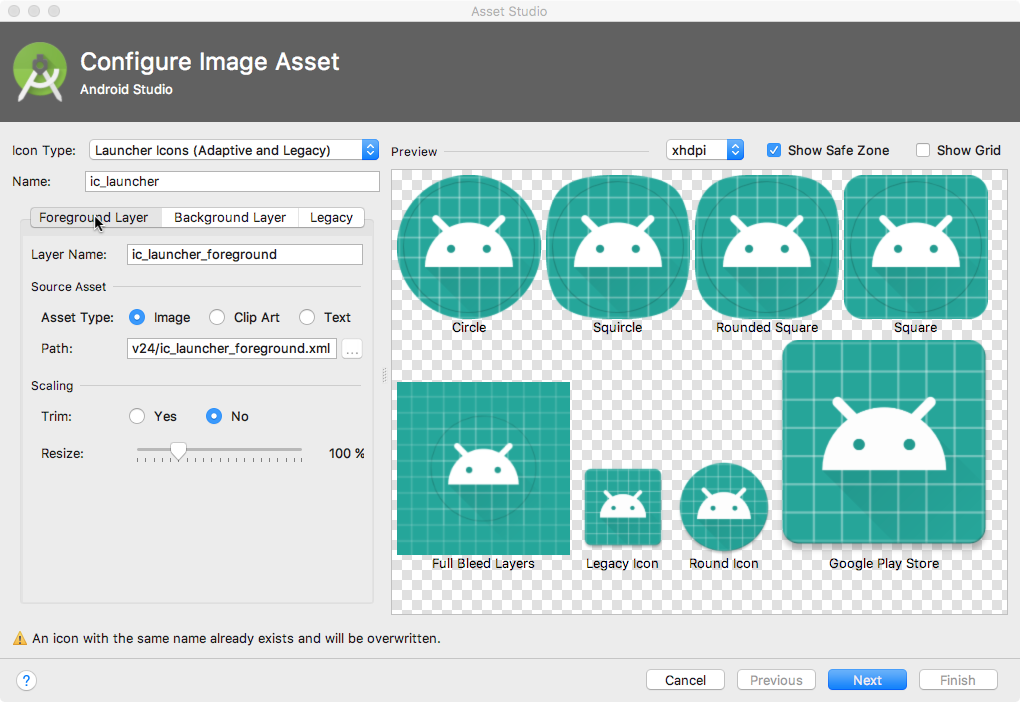


### **Alterando o ícone do iniciador**

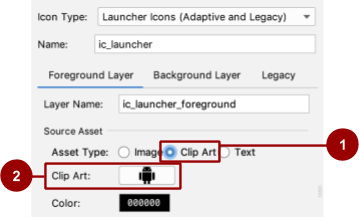
Passar pelo processo de alteração do ícone do iniciador apresenta os recursos de ativos de imagem do Android Studio.

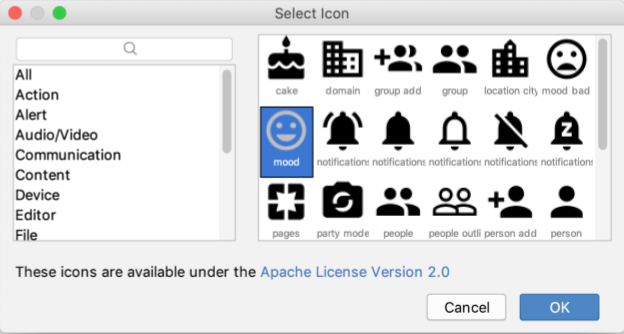
No Android Studio, veja como alterar o ícone de inicialização do aplicativo Android Sunflower:

1. No painel **Projeto> Android** , clique com o botão direito (ou Control + clique) na pasta **res** . Selecione **Novo> Ativo de imagem** . A caixa **de** diálogo **Configure Image Asset é** exibida.



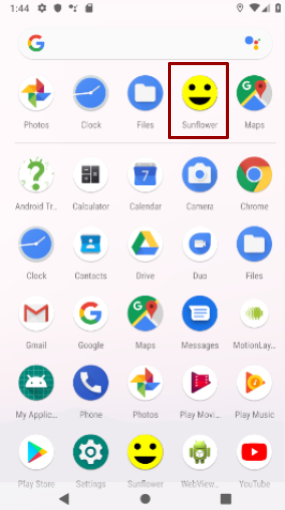
1. No campo **Icon Type** , selecione **Launcher Icons (Adaptive & Legacy)** se ainda não estiver selecionado. Clique na guia **Foreground Layer** .
2. Para o **tipo de ativo** , selecione **Clip Art** , mostrado como 1 na captura de tela abaixo.

4. Clique no ícone do robô no campo **Clip Art** , mostrado como 2 na captura de tela acima. A caixa de diálogo **Selecionar ícone é** exibida, mostrando o conjunto de ícones do Material Design. 5. Navegue na caixa de diálogo **Selecionar ícone** ou pesquise um ícone por nome. Selecione um ícone, como o ícone de **humor** , para sugerir um bom humor. Clique **OK** .



1. Na caixa **de** diálogo **Configure Image Asset** , clique na guia **Background Layer** . Para o **tipo de ativo** , selecione **cor** . Clique no chip de cor e selecione uma cor para usar como camada de fundo para o ícone.
2. Clique na guia **Legado** e revise as configurações padrão. Confirme que você deseja gerar ícones legados, redondos e da Google Play Store. Clique em **Avançar** .
3. A caixa de diálogo **Confirmar caminho do ícone é** exibida, mostrando onde os arquivos de ícone estão sendo adicionados e substituídos. Clique em **Concluir** .
4. Execute o aplicativo em um emulador AVD ou dispositivo Android.

O Android Studio adiciona automaticamente as imagens do iniciador aos diretórios **mipmap** para as diferentes densidades de tela. O aplicativo Sunflower para Android agora usa o novo ícone de clip-art como ícone de inicialização.



**Dica** : para aprender sobre como projetar ícones de iniciador eficazes, consulte o guia de [ícones de produto](https://material.io/design/iconography/) do Material Design .

1. Execute o aplicativo novamente. Certifique-se de que o novo ícone do iniciador apareça na tela Search Apps.

[5. Tarefa: Explorar documentos e outros recursos](https://developer.android.com/codelabs/kotlin-android-training-available-resources#4)

Etapa 1: explore a documentação oficial do Android

Explore alguns dos sites de documentação Android mais úteis e familiarize-se com o que está disponível:

1. Acesse [developer.android.com](http://developer.android.com/index.html) . Esta documentação oficial do desenvolvedor Android é mantida atualizada pelo Google.
2. Acesse [developer.android.com/design/](http://developer.android.com/design/) . Este site oferece diretrizes para projetar a aparência e a funcionalidade de aplicativos Android de alta qualidade.
3. Acesse [material.io](https://material.io/) , que é um site sobre design de materiais. Material Design é uma filosofia de design conceitual que descreve como todos os aplicativos, não apenas os aplicativos Android, devem parecer e funcionar em dispositivos móveis. Navegue pelos links para saber mais sobre Material Design. Por exemplo, para aprender sobre o uso de cores, clique na guia **Design** e selecione **Cor** .
4. Acesse [developer.android.com/docs/](http://developer.android.com/docs/) para encontrar informações de API, documentação de referência, tutoriais, guias de ferramentas e exemplos de código.
5. Acesse [developer.android.com/distribute/](http://developer.android.com/distribute/) para encontrar informações sobre como publicar um aplicativo no [Google Play](https://developer.android.com/distribute/googleplay/index.html) . Google Play é o sistema de distribuição digital do Google para aplicativos desenvolvidos com o Android SDK. Use o [Google Play Console](https://developer.android.com/distribute/console/index.html) para aumentar sua base de usuários e começar a [ganhar dinheiro](https://developer.android.com/distribute/best-practices/earn/) .

**Etapa 2: explore o conteúdo da equipe do Android e da Pesquisa Google**

1. Explore o [canal Android Developer no YouTube](https://www.youtube.com/user/androiddevelopers) , que é uma ótima fonte de tutoriais e dicas.
2. Visite o [blog oficial do Android](https://blog.google/products/android/) , onde a equipe do Android publica notícias e dicas.
3. Insira uma pergunta na Pesquisa Google e o mecanismo de Pesquisa Google coleta resultados relevantes de vários recursos. Por exemplo, use a Pesquisa Google para fazer a pergunta: "Qual é a versão do sistema operacional Android mais popular na Índia?" Você pode até inserir mensagens de erro na Pesquisa Google.

Etapa 3: **Pesquisar no Stack Overflow**

Stack Overflow é uma comunidade de programadores que ajudam uns aos outros. Se você tiver um problema, é provável que alguém já tenha postado uma resposta.

1. Vá para [Stack Overflow](http://stackoverflow.com/" \t "_blank) .
2. Na caixa de pesquisa, digite uma pergunta como "Como faço para configurar e usar o ADB via Wi-Fi?" Você pode pesquisar no Stack Overflow sem se registrar, mas se quiser postar uma nova pergunta ou responder a uma pergunta, você precisa se registrar.
3. Na caixa de pesquisa, digite [android]. Os []colchetes indicam que você deseja pesquisar postagens marcadas como sendo sobre Android.
4. Você pode combinar tags e termos de pesquisa para tornar sua pesquisa mais específica. Experimente estas pesquisas:

* [android] and [layout]
* [android] "hello world"

**Dica:** para saber mais sobre as várias maneiras de pesquisar no Stack Overflow, consulte a [Central de Ajuda](http://stackoverflow.com/help/searching) do [Stack Overflow](http://stackoverflow.com/help/searching" \t "_blank) .

[6. Resumo](https://developer.android.com/codelabs/kotlin-android-training-available-resources#5)

* A documentação oficial do desenvolvedor Android está em [developer.android.com](http://developer.android.com/index.html) .
* *Material Design* é uma filosofia de design conceitual que descreve como os aplicativos devem ser exibidos e funcionar em dispositivos móveis. Material Design não é apenas para aplicativos Android. As diretrizes do Material Design estão em [material.io](https://material.io/) .
* O Android Studio fornece modelos para projetos de atividades e aplicativos comuns e recomendados. Esses modelos oferecem código funcional para casos de uso comuns.
* Ao criar um projeto, você pode escolher um modelo para sua primeira atividade.
* Enquanto estiver desenvolvendo seu aplicativo, você pode criar atividades e outros componentes de aplicativo a partir de modelos integrados.
* [Amostras do Google](https://github.com/googlesamples) contém amostras de código que você pode estudar, copiar e incorporar em seus projetos.