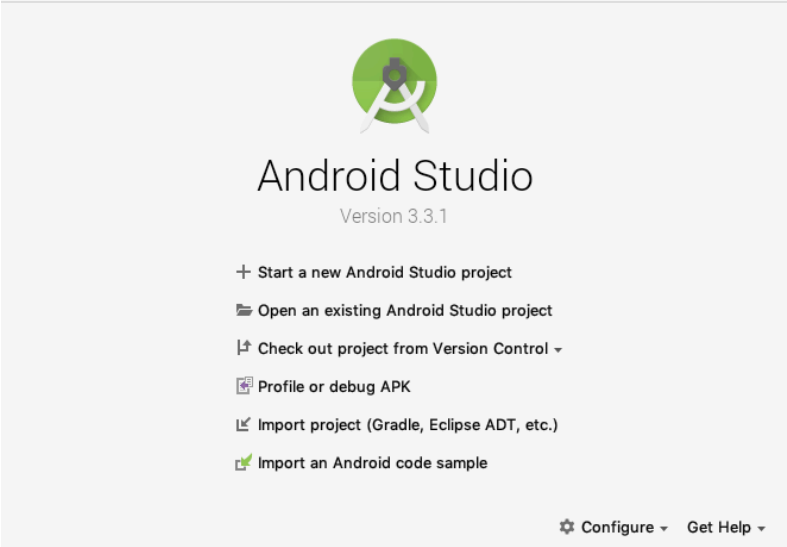
Este material baseia-se no Capítulo 2 do livro [1]. Iremos construir uma aplicação de boas vindas (uma espécie de Hello World para Android) ilustrando diversos detalhes fundamentais. Para prosseguir, providencie o download e instalação do Android SDK e do IDE Android Studio. Eles podem ser obtidos a partir do Link 1.

Link 1
https://developer.android.com/studio/

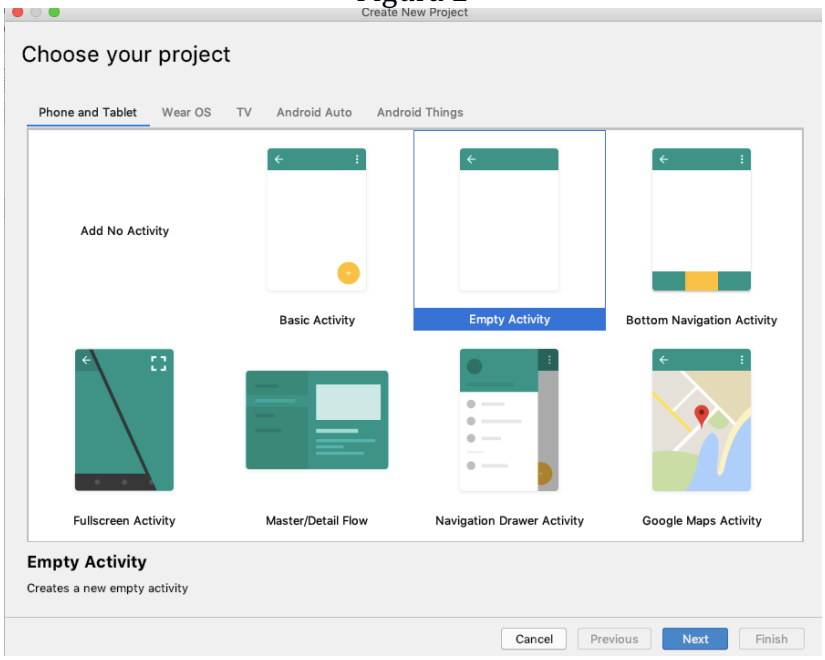
Passo 1. Para criar uma nova aplicação para Android, abra o Android Studio e escolha a opção Start a new Android Studio Project, como mostra a Figura 1. Observe que, conforme novas versões da ferramenta são liberadas, as interfaces aqui exibidas podem ser diferentes.

Figura 1



Passo 2. Na tela exibida pela Figura 2, escolha o template a ser utilizado para a criação da aplicação. O Android Studio irá gerar diversos arquivos necessários para o funcionamento da aplicação. Dependendo do template escolhido, ele adicionará certas funcionalidades. Nosso objetivo é construir uma aplicação ˜hello world˜, por isso, escolha o template Empty Activity.

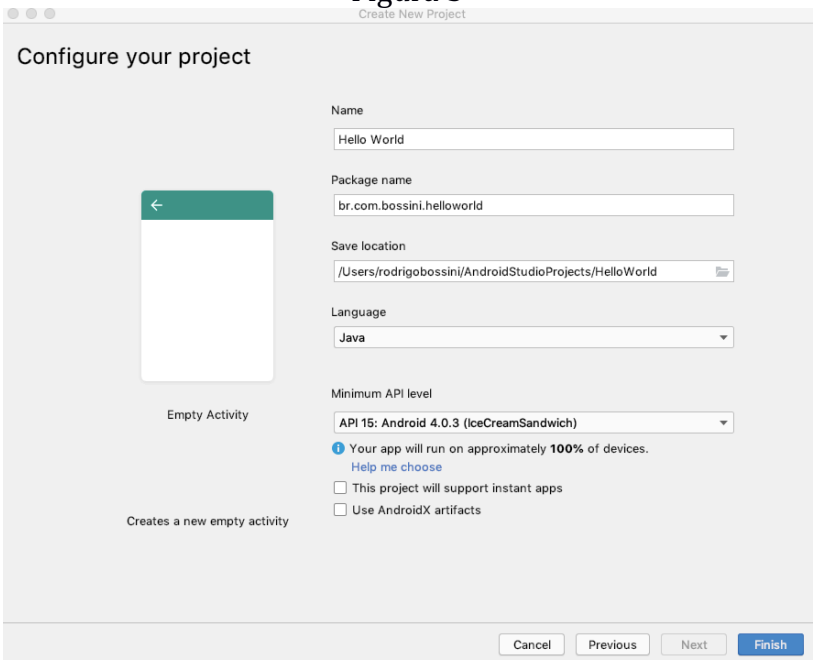
Figura 2



Passo 3. A seguir, como mostra a Figura 3, preencha os campos com os seguintes valores:

Name: Hello World
Package Name: br.com.bossini.helloworld
Save Location: Mantenha o padrão
Language: Java
Minimum API Level: API 15

Figura 3

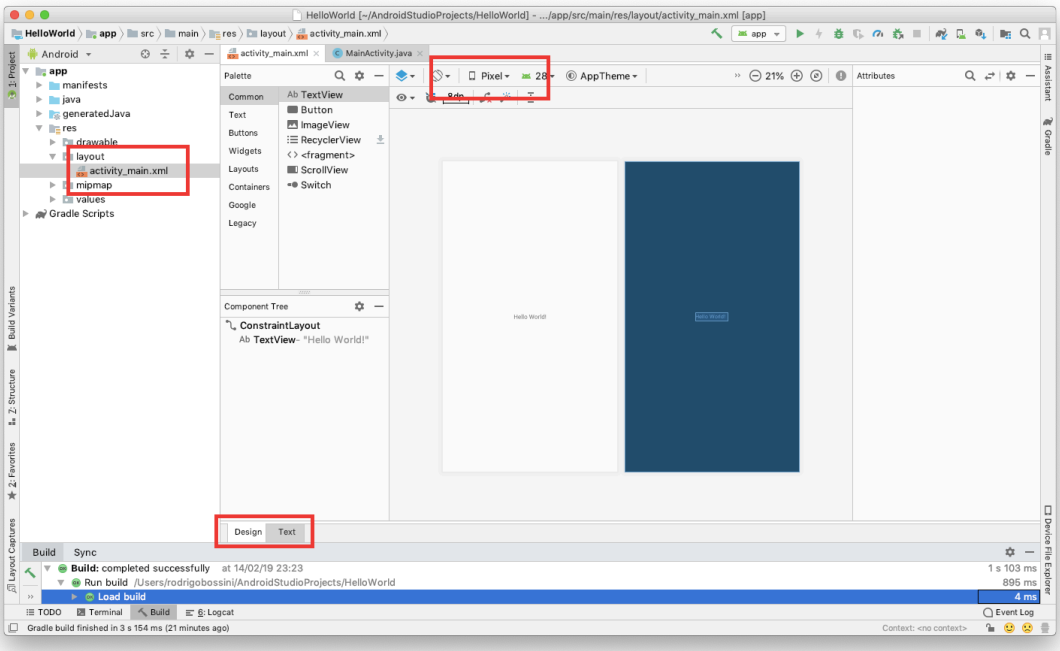


Note que o Android Studio pega o inverso do domínio e o coloca como pacote principal da aplicação. Neste exemplo, portanto, o pacote principal será br.com.bossini.helloworld. Você pode escolher um nome que achar interessante, lembrando que a parte br.com é bastante comum na indústria. Depois disso, basta clicar em Finish e aguardar. Note que, ao criar a primeira aplicação logo depois da instalação do Android Studio, o tempo de espera pode ser de alguns minutos. Isso ocorreo pois o Android Studio pode fazer o download do Gradle neste momento.

Observe que o Android Studio já criou uma aplicação Hello World bastante simples. Faremos agora diversos ajustes ilustrando funcionalidades muito importantes.

Passo 4. A Figura 4 mostra o editor visual do Android Studio. Observe que o arquivo activity_main.xml está selecionado e é possível visualizá-lo de duas formas: Design (para fazer edições graficamente, arrastando e soltando) e Text (para editar o arquivo xml textualmente, o que pode ser conveniente dependendo do que é preciso).

Figura 4



Ainda na tela exibida pela Figura 4, escolha Nexus 6 como modelo (na parte superior, um pouco acima do dispositivo).

Passo 5. Agora iremos adicionar uma figura ao projeto, que deve ser colocada na pasta drawable. Faça o download da figura bug.png do site da disciplina e a seguir basta copiá-la e colá-la na pasta drawable. Esta figura será colocada na tela principal da aplicação.

Passo 6. Também trocaremos o ícone da aplicação, aquele que o usuário utiliza para iniciar a aplicação. Para isso, clique com o direito na pasta res e selecione New > Image Asset. Utilize a figura DeitelOrange.png que também pode ser obtida no site da disciplina.

Passo 7. Por padrão, o arquivo activity_main.xml contém um **ConstraintLayout** (ou algum outro, dependendo da versão de seu Android Studio) que é um gerenciador de layout e que contém um TextView, o componente que exibe o texto Hello World neste exemplo.

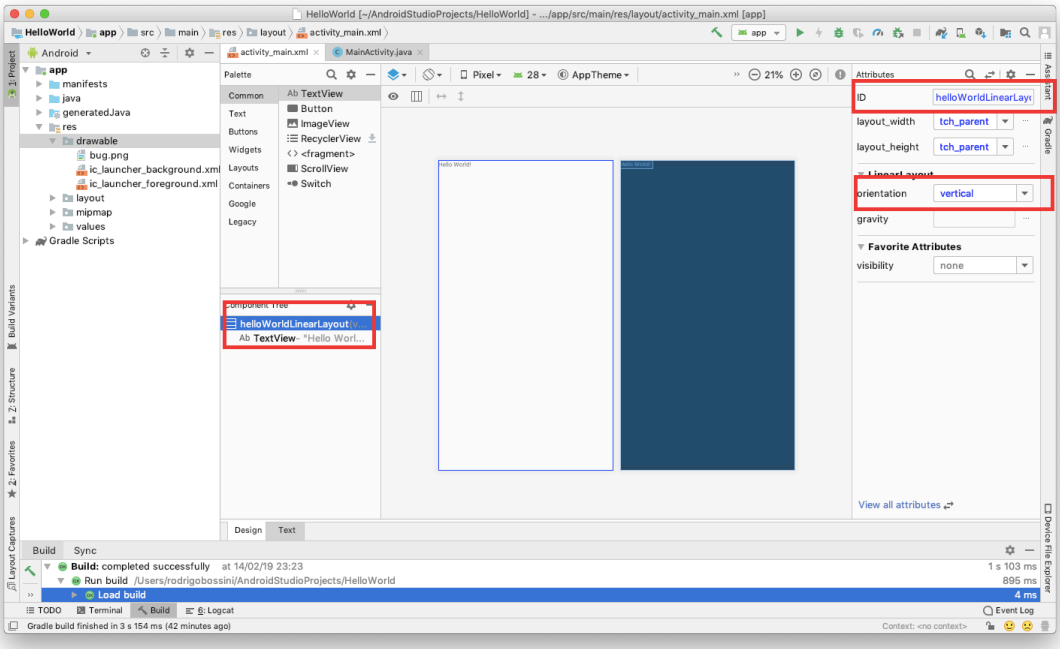
7.1 Substituiremos o ConstraintLayout por um **LinearLayout**, um gerenciador de layout que exibe seus componentes horizontal ou verticalmente. Para isso, clique em Text (aquele da Figura 4, ao lado de Design). Troque a primeira ocorrência de ConstraintLayout, incluindo o pacote, para LinearLayout. Observe que a segunda ocorrência (lá embaixo) já é alterada automaticamente.

7.2 Remova também as propriedades **absoluteX** e **absoluteY** do TextView.

7.3 Em geral, é interessante atribuir um nome a cada componente visual (incluindo os gerenciadores de layout), o que pode ser feito utilizando-se a propriedade id. Para fazer essa alteração, volte à opção Design, selecione o LinearLayout na árvore e troque seu id no menu à direita, como mostra a Figura 5.

7.4 Vamos, ainda, trocar a orientação do layout. Ele ficará na vertical, ou seja, seus filhos serão colocados um abaixo do outro. A Figura 5 também mostra essa alteração.

Figura 5



Note que, conforme edições são feitas graficamente, o conteúdo textual vai sendo alterado automaticamente pelo IDE. O conteúdo até então é exibido na Listagem 1.

Listagem 1

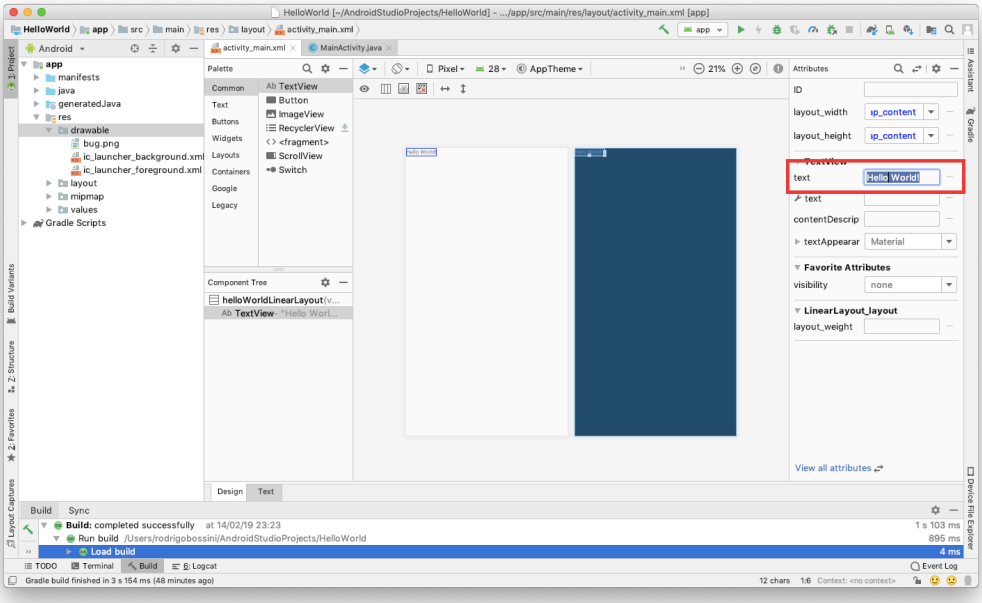
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/helloWorldLinearLayout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MainActivity">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!" />
</LinearLayout>
```

Passo 8. A aplicação já possui um componente visual do tipo TextView que exibe o texto Hello World. Siga o mesmo procedimento para alterar seu id para helloWorldTextView.

Passo 9. Esse TextView possui uma característica indesejada: o texto que exibe está localizado diretamente no arquivo activity_main.xml. Isso é ruim pois dessa forma a aplicação não pode ser internacionalizada facilmente. A boa prática consiste em colocar o texto da aplicação em um arquivo XML apropriado somente para isso. Como veremos, isso torna muito simples a **internacionalização**.

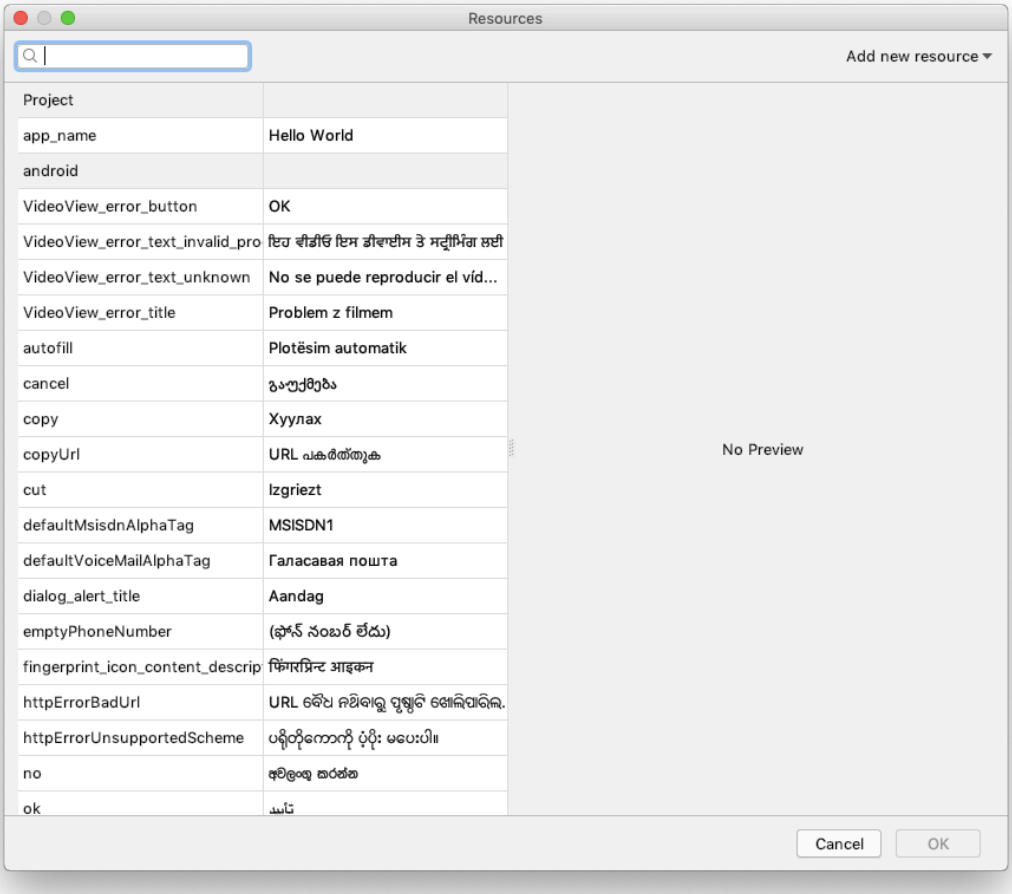
9.1 Na pasta res/values há um arquivo strings.xml que consiste de uma coleção de pares chave/valor que representam os textos da aplicação. Para adicionar o texto Hello World neste arquivo, selecione o TextView no editor visual e encontre sua propriedade text no menu à esquerda inferior (Figura 6).

Figura 6



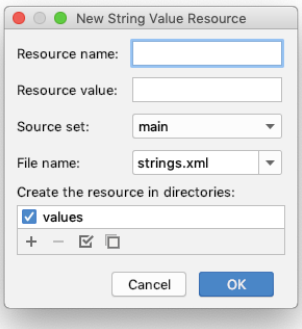
9.2 Agora clique nos três pontos ao lado do texto para obter a tela exibida pela Figura 7.

Figura 7



9.3 Clique em Add new Resource >> New String Value para obter a tela exibida pela Figura 8.

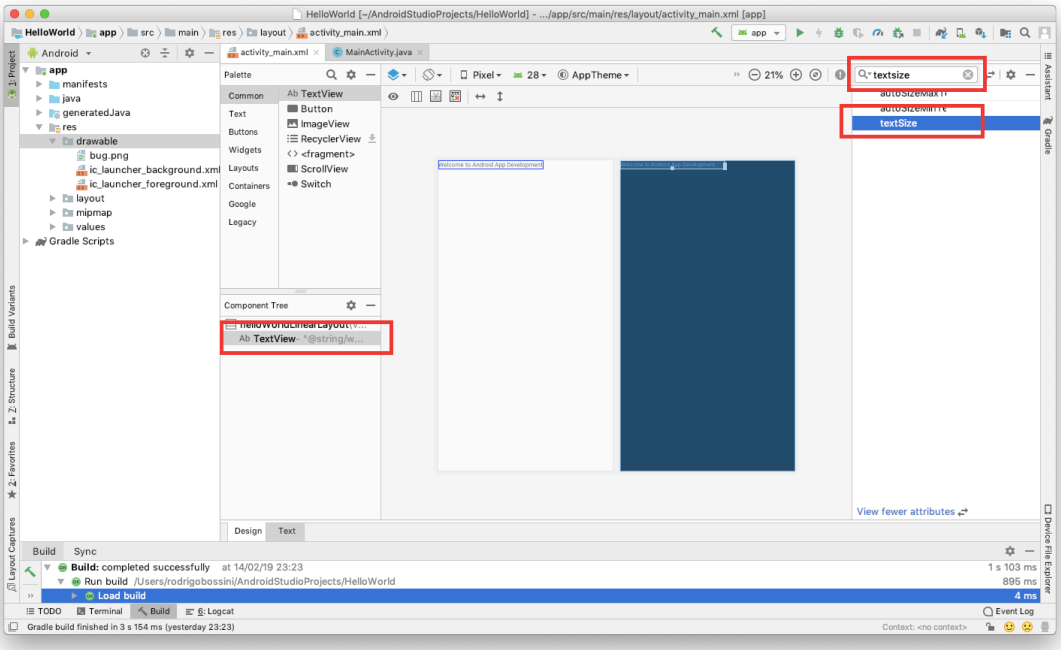
Figura 8



9.4 Preencha Resource name com **welcome** e Resource value com **Welcome to Android App Development**. Observe que o valor associado ao TextView agora é "@string/welcome". Essa notação quer dizer que o texto exibido por esse TextView será aquele associado ao name "welcome", localizado no arquivo de recursos strings.xml, aquele localizado na pasta res/values.

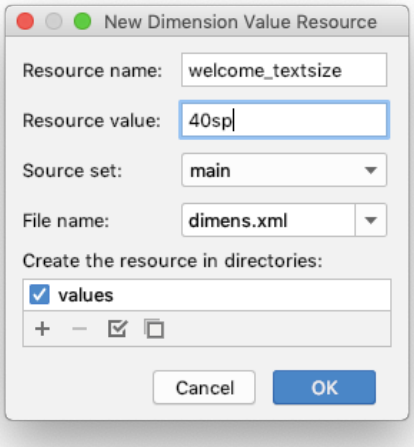
Passo 10. Agora iremos configurar a propriedade textSize do TextView. Isso pode ser feito diretamente no arquivo activity_main.xml mas uma configuração assim pode ser reutilizada por diversos componentes visuais e, por essa razão, pode ser interessante criar a configuração em um arquivo à parte e então utilizá-la. É o que faremos usando o arquivo dimens.xml, também na pasta res/values. Com o TextView selecionado, encontre a propriedade textSize, como na Figura 9. Pode ser necessário clicar em View all attributes antes. Use a lupa para buscar pelo campo desejado, digitando seu nome (nesse caso, textSize).

Figura 9



10.1 Clique nos três pontos e escolha New Resource > New Dimen Value, de maneira análoga à criação de uma string. Configure os valores como na Figura 10.

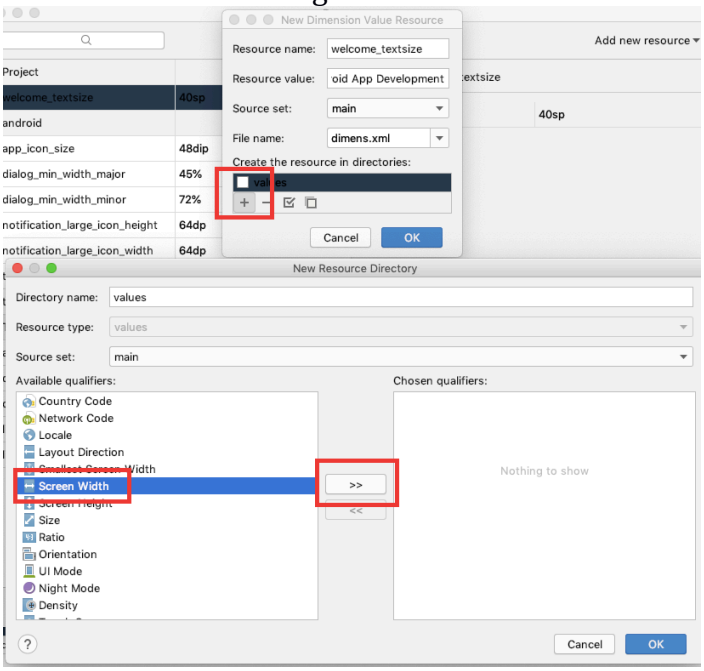
Figura 10



Há algumas unidades de medida disponíveis, como px, dp (ou dip) e sp. Para que o Android possa "escalar" os elementos visuais de acordo com o dispositivo sendo utilizado, opte por dp ou sp. Ambos servem, mas sp também permite que a plataforma considere a preferência do usuário para tamanho de fonte, configurada no sistema operacional. Evite utilizar a unidade de medida px, pois sua aplicação não se comportará adequadamente em dispositivos com telas de tamanho e resolução diferentes.
Observe que o valor associado ao textSize ficou sendo "@dimen/welcome_textsize". Isso quer dizer que o valor a ser utilizado será aquele associado ao name welcome_textsize, localizado no arquivo dimens.xml na pasta res/values.

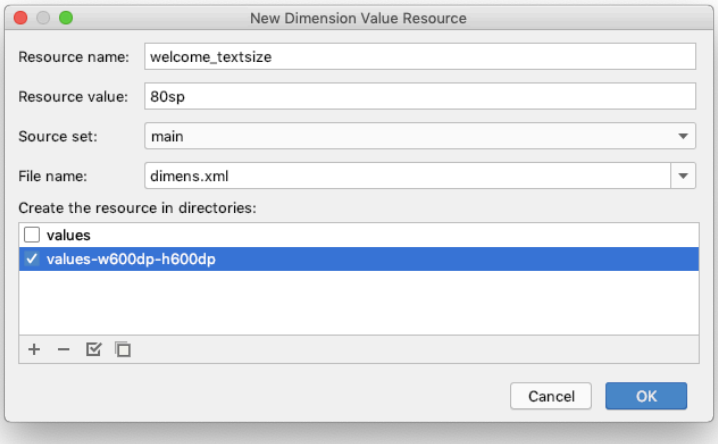
Passo 11. Agora criaremos uma configuração apropriada para tablets que possuam pelo menos 600 DPI. Para isso, abra novamente o editor da Figura 10, coloque o mesmo name (welcome_textsize) e 80sp de value. Desmarque a checkbox values e clique no botão +. Como na Figura 11, escolha Screen Width e clique em >>.

Figura 11



11.1 Preencha o campo Screen Width com 600 e clique OK. Siga os mesmos passos para o campo Screen Height. **Antes de sair, marque as caixas values-w600dp e h600dp** e clique em Ok. Veja a Figura 12.

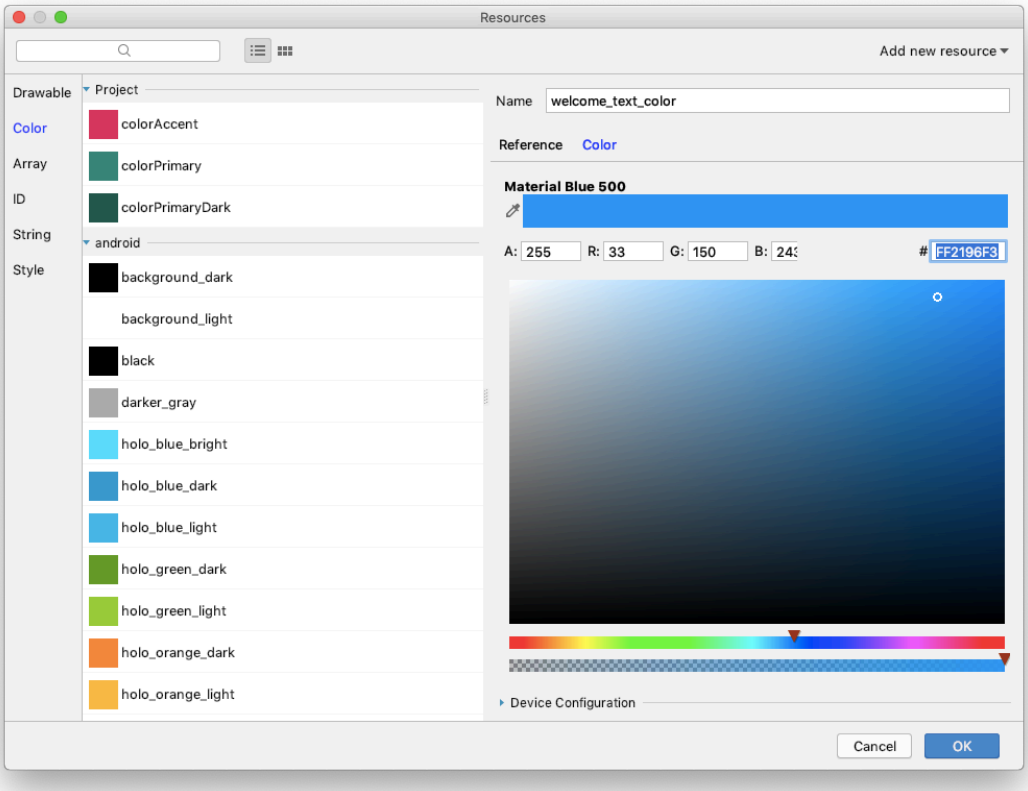
Figura 12



Observe que agora existem novas pastas dimens na pasta res/values. A configuração adequada será escolhida automaticamente pelo Android com base no dispositivo que estiver executando a aplicação.

Passo 12. Agora vamos trocar a propriedade textColor do TextView. Faça de maneira análoga à criação de string e dimensões. Encontre a propriedade e crie um novo recurso com os valores name welcome_text_color e value #FF2196F3. Veja a Figura 13.

Figura 13



Passo 13. Agora iremos centralizar o texto da TextView usando a propriedade gravity. Para tal, expanda-a e escolha “center”.

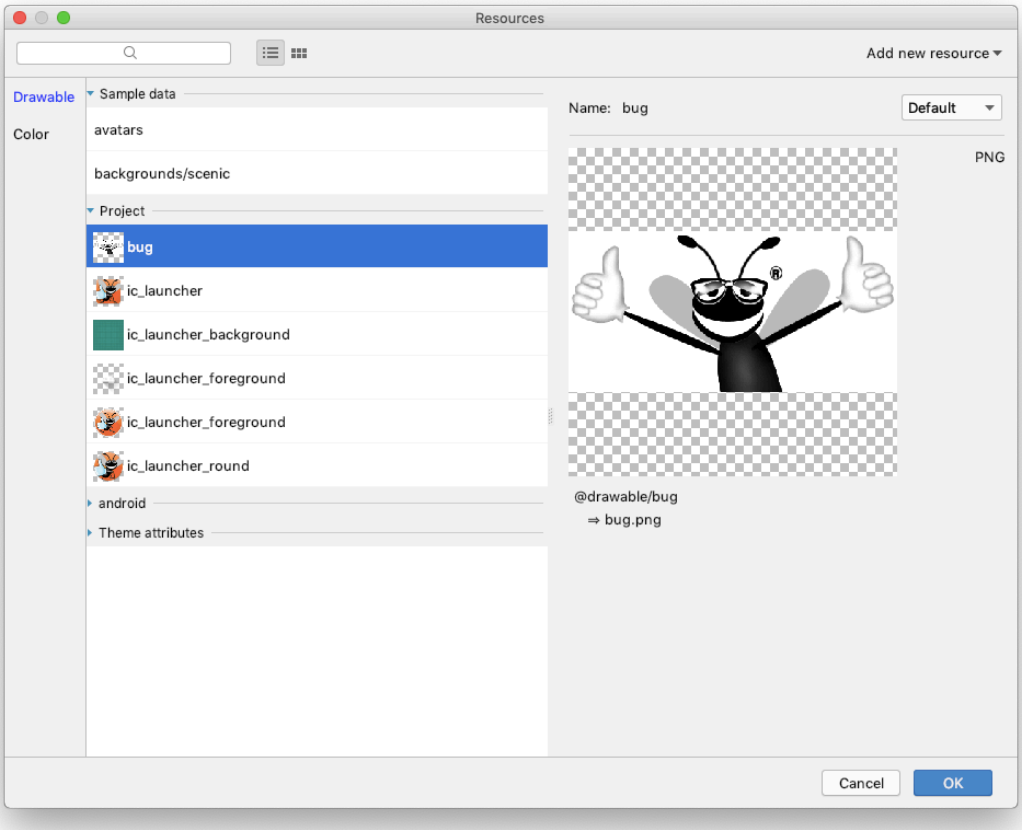
Passo 14. Também é possível controlar o posicionamento do componente visual dentro do layout em que ele está, usando a propriedade layout:gravity. Neste exemplo, expanda a propriedade, selecione center e então horizontal.

Passo 15. Faça esse passo na aba Text. Um LinearLayout é capaz de posicionar seus componentes visuais proporcionalmente, usando a propriedade weight. Por padrão, essa propriedade tem valor 0. Desejamos que o TextView e a imagem que será adicionada ocupem metade (verticalmente) da tela cada. Para isso basta configurar a propriedade weight de cada uma com valores iguais.

15.1 Coloque o valor 1 na propriedade layout_weight do TextView. Após alguns segundos aparecerá uma lâmpada amarela ao lado de layout:height. Trata-se de um componente chamado Android Lint, responsável por identificar aspectos da aplicação que podem ser melhorados, por exemplo visando melhor desempenho. Neste caso, o Android Lint está sugerindo usar 0dp no lugar de wrap_content, o que segundo ele poderá melhorar o desempenho. Vá em frente e faça a troca.

Passo 16. Agora vamos adicionar um componente visual do tipo ImageView para exibir a imagem bug.png. Para isso, arraste e solte um ImageView da paleta para o canvas, posicionando-o abaixo do TextView. Na tela exibida pela Figura 14, expanda project e escolha bug.

Figura 14



16.1 Troque também o id do ImageView, para bugImageView.

Passo 17. No Android, há um recurso chamado TalkBack que basicamente fala um texto para que pessoas com deficiência visual possam utilizar uma aplicação. Esse texto pode ser aquele exibido na tela ou pode ser um texto qualquer configurado pelo programador. Um usuário interessado em utilizar esse recurso só precisa habilitá-lo nas configurações de seu aparelho. Para um TextView, o dispositivo irá falar o seu texto. Para uma figura, o programador pode fornecer um texto descritivo usando a propriedade contentDescription. Lembra-se da lâmpada amarela que o Android Lint exibiu quando adicionamos o ImageView. Era a isso que ele se referia!
Encontre a propriedade **contentDescription** do ImageView e adicione um texto (usando o arquivo strings.xml, como feito antes!). O name deve ser deitel_logo e o value deve ser Deitel's bug's double thumbs up.

Passo 18. Agora iremos internacionalizar a aplicação, ou seja, oferecê-la em diferentes idiomas. Digamos que queremos oferecer a aplicação em alemão. Para isso, basta criar uma pasta chamada values-de. O trecho "de" refere-se a um padrão ISO para a identificação de países e idiomas. Neste caso estamos especificando somente o idioma. A variação values-de_DE identifica o idioma e o país. Neste caso, estamos nos referindo ao alemão falado na Alemanha. Para adicionar a tradução, abra o arquivo strings.xml e clique em Open Editor. No canto superior esquerdo,

clique no globo para adicionar um novo Locale. Faça então as traduções (ok, pode usar o Google Tradutor :)).

Bibliografia

[1] Deitel, P., Deitel, H., Wald, A. Android 6 for Programmers – An App-Driven Approach. 3rd edition. Pearson, 2016.