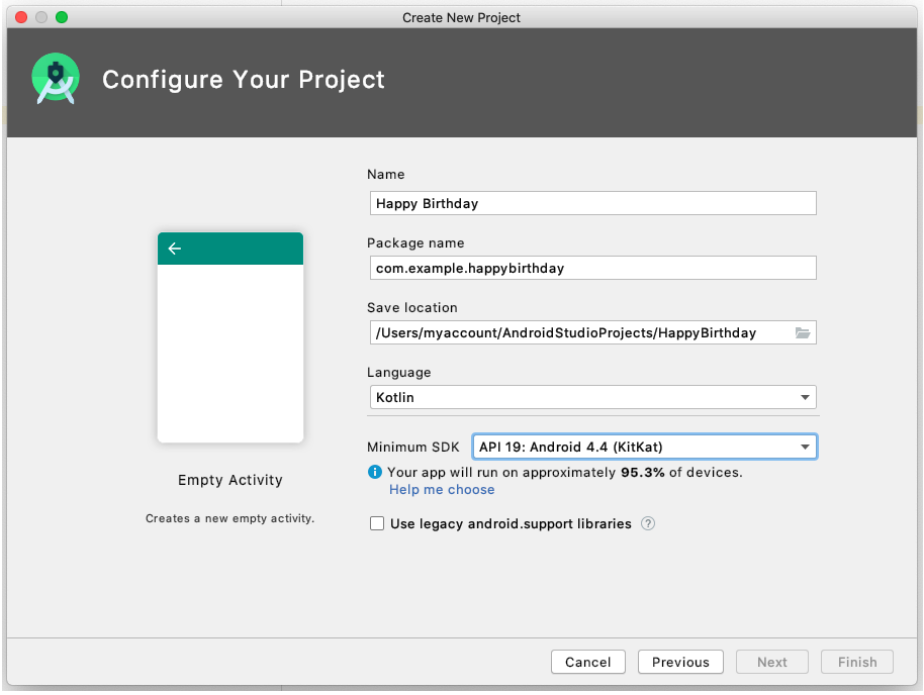
2. Configurar o app Happy Birthday

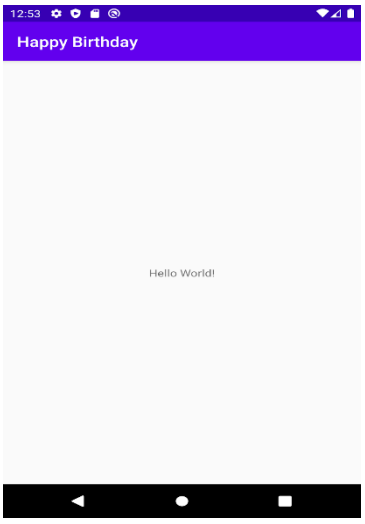
Criar um projeto Empty Activity

1. Para começar, crie um novo projeto Kotlin no Android Studio usando o modelo **Empty Activity**.
2. Dê o nome "Happy Birthday" ao app e defina o nível mínimo da API como 19 (KitKat).

Importante: caso você não saiba como criar um novo projeto no Android Studio, consulte Criar e executar seu primeiro app Android para mais informações.



3. Execute o app. Ele ficará parecido com a captura de tela abaixo.



Quando você cria o app Happy Birthday com o modelo Empty Activity, o Android Studio configura recursos para um app Android básico, incluindo uma mensagem "Hello World!" no meio da tela. Neste codelab, você aprende

como essa mensagem é mostrada, como mudar o texto dela para uma mensagem de aniversário e como adicionar e formatar outras mensagens.

Sobre as interfaces do usuário

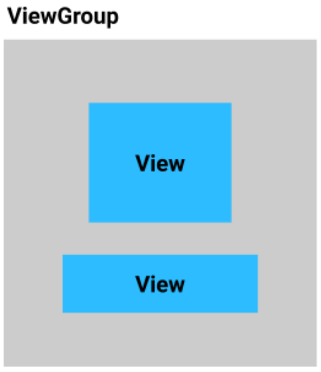
A interface do usuário (IU) de um app é o que você vê na tela: texto, imagens, botões e muitos outros tipos de elementos. Essa é a forma com que o app mostra informações ao usuário e como o usuário interage com o app.

Cada um desses elementos é o que chamamos de `View`. Quase tudo o que você vê na tela do seu app é uma `View`. As `Views` podem ser interativas, como um botão clicável ou um campo de entrada editável.

Neste codelab, você vai trabalhar com um tipo de `View` que serve para mostrar texto e é conhecida como `TextView`.

As `Views` em um app Android não flutuam na tela sozinhas. As `Views` são relacionadas umas com as outras. Por exemplo, uma imagem pode aparecer ao lado de um texto, e os botões podem formar uma fileira. Para organizar as `Views`, coloque-as em um contêiner. Um `ViewGroup` é um contêiner em que objetos `View` podem ficar. Ele é responsável por organizar as `Views` dentro de si. A organização, ou o *layout*, pode mudar dependendo do tamanho e da proporção da tela do dispositivo Android em que o app é executado. O layout pode se adaptar à orientação do dispositivo, ou seja, modo retrato ou paisagem.

O `ConstraintLayout` é um tipo de `ViewGroup`, que ajuda a organizar as `Views` de forma flexível.



Sobre o Layout Editor

Criar a interface do usuário organizando as `Views` e os `ViewGroups` é uma grande parte da criação de um app Android. O Android Studio oferece uma ferramenta que ajuda você a fazer isso, chamada **Layout Editor**. Você usará o **Layout Editor** para mudar o texto "Hello World!" para "Happy Birthday!" e, mais tarde, para definir o estilo do texto.

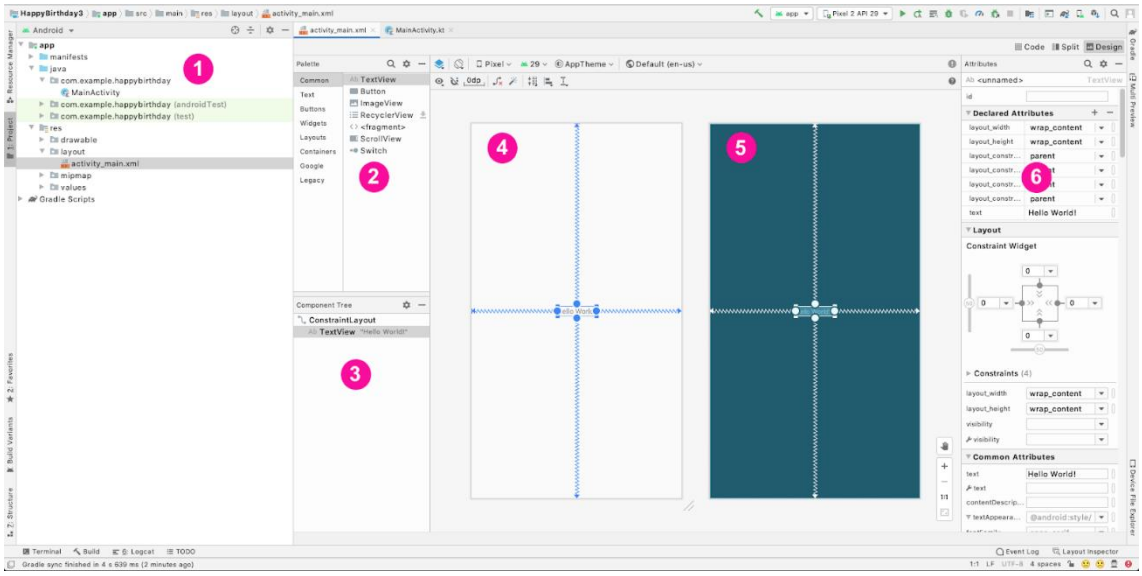
Quando o **Layout Editor** for aberto, ele exibirá vários elementos. Este codelab vai te ensinar a usar a maioria deles. Use a captura de tela com comentários abaixo para entender as janelas no **Layout Editor**. Você aprenderá mais sobre cada elemento à medida que fizer mudanças no app.

- À esquerda, (1) é a janela **Project**, que você já viu. Ela lista todos os arquivos que compõem o projeto.

- No centro, você pode conferir os desenhos (4) e (5), que representam o layout da tela do app. A representação (4) à esquerda é uma estimativa de como o app ficará ao ser executado. Ela é chamada de visualização **Design**.
- A representação (5) à direita é a visualização **Blueprint**, que pode ser útil para operações específicas.
- A **Palette** (2) contém listas de diferentes tipos de `Views` que podem ser usadas no app.
- A **Component Tree** (3) é uma representação diferente das visualizações da tela. Ela lista todas as visualizações da tela.
- No canto direito (6), encontra-se a parte **Attributes**. Ela mostra os atributos de uma `View` e permite mudá-los.

Leia mais sobre o **Layout Editor** e como configurá-lo no guia de referência do desenvolvedor.

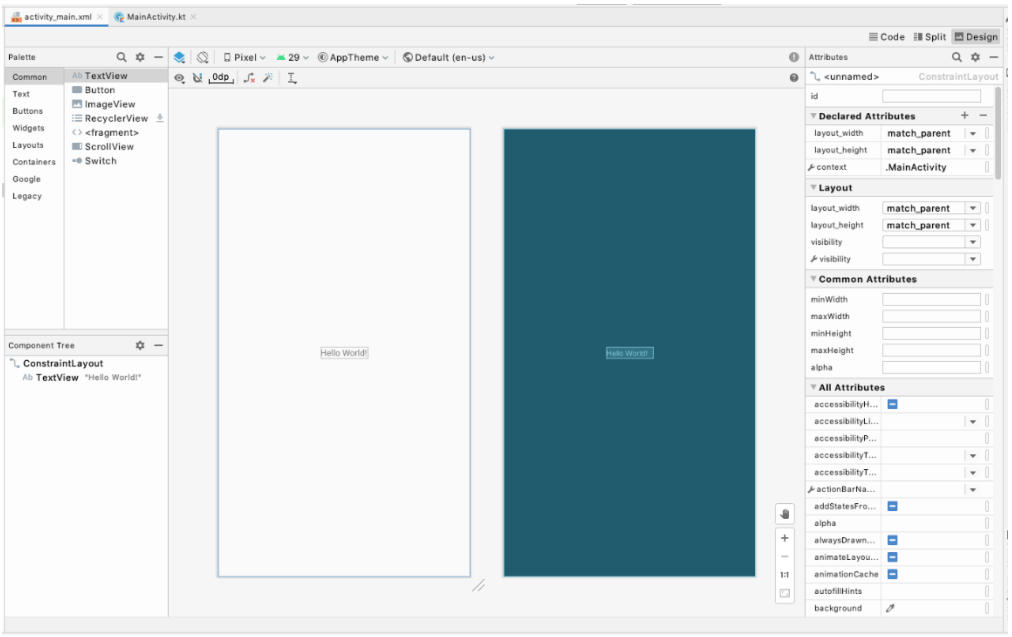
Captura de tela com comentários de todo o **Layout Editor**:



Agora, vamos mudar algumas coisas no **Layout Editor** para deixar o app mais parecido com um cartão de aniversário.

Mudar a mensagem "Hello World"

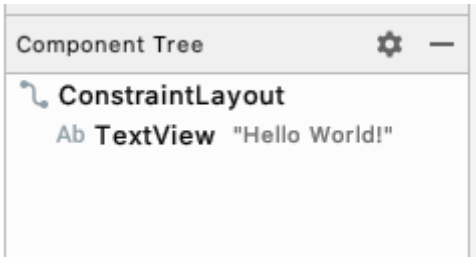
1. No Android Studio, localize a janela **Project** à esquerda.
2. Preste atenção a estes arquivos e pastas: a pasta **app** contém a maioria dos arquivos que você mudará. A pasta **res** é destinada a recursos, como imagens ou layouts de tela. A pasta **layout** é destinada a layouts de tela. O arquivo `activity_main.xml` contém uma descrição do layout da tela.
3. Abra a pasta **app**, depois a **res** e, então, a **layout**.
4. Clique duas vezes em `activity_main.xml`. Isso abre o arquivo `activity_main.xml` no **Layout Editor** e mostra o layout descrito na visualização **Design**.



Observação: nestes codelabs, com frequência você precisará abrir um arquivo, como nas etapas anteriores. Isso pode ser resumido como: abrir **activity_main.xml** (**res** > **layout** > **activity_main.xml**) em vez de listar cada etapa separadamente.

5. Consulte a lista de visualizações na **Component Tree**. Observe que há um `ConstraintLayout`, assim como uma `TextView` abaixo dele. Eles representam a IU do app. A `TextView` está recuada porque está dentro do `ConstraintLayout`. À medida que você adicionar mais `Views` ao `ConstraintLayout`, elas vão ser adicionadas a essa lista.

6. A mensagem **Hello World!**, que é o texto que você encontra ao executar o app, aparece do lado da `TextView`.

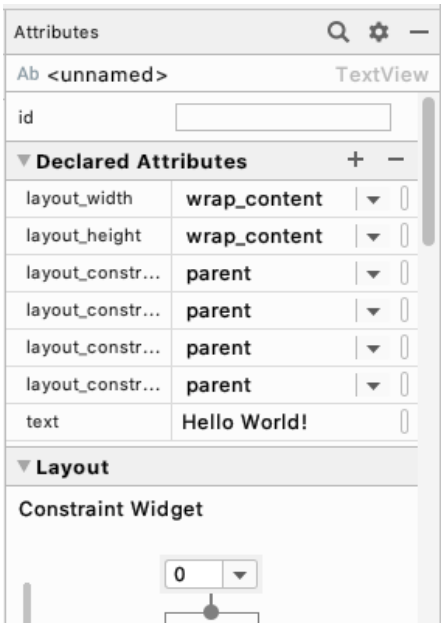


7. Na **Component Tree**, clique em `TextView`.

8. Localize **Attributes** à direita.

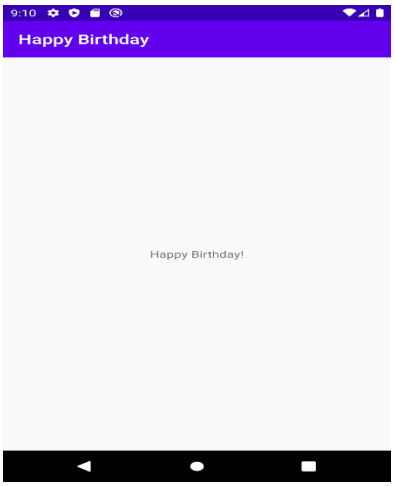
9. Localize a seção **Declared Attributes**.

10. Observe que o atributo **text** na seção **Declared Attributes** contém a mensagem **Hello World!**.



O atributo **text** mostra o texto exibido dentro da `TextView`.

11. Clique no atributo **text** em que o texto **Hello World!** se encontra.
12. Mude o texto para **Happy Birthday!** e pressione a tecla **Enter**. Por enquanto, não se preocupe caso você receba um alerta sobre uma string fixada no código. Você aprenderá a se livrar desse alerta no próximo codelab.
13. O texto mudou na **Design View**. Isso é incrível, porque permite visualizar as mudanças na hora.
14. Execute o app, que agora vai mostrar o texto **Happy Birthday!**



Bom trabalho! Você fez suas primeiras mudanças em um app Android.

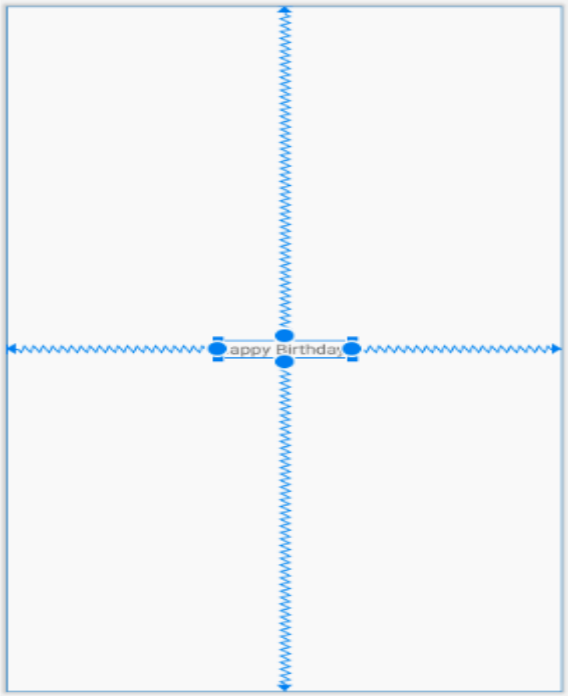
3. Adicionar TextViews ao layout

Agora, o cartão de aniversário que você está criando ficou diferente do texto que aparece no app. Em vez do texto pequeno no centro, você precisará de duas mensagens maiores: uma no canto superior esquerdo e outra no canto

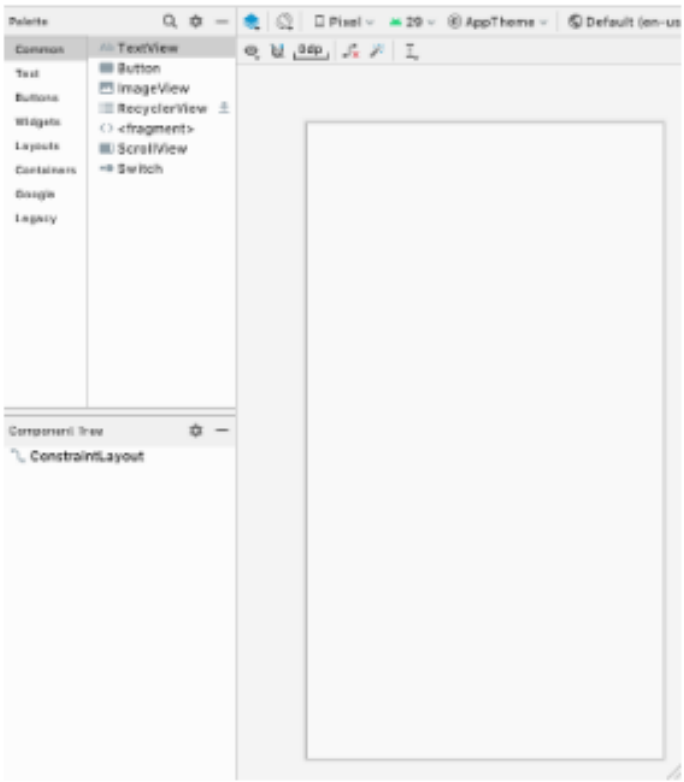
inferior direito. Nesta tarefa, você vai excluir a `TextView` existente e adicionar duas `TextViews` novas. Você também aprenderá a posicioná-las no `ConstraintLayout`.

Excluir a TextView atual

1. No **Layout Editor**, clique para selecionar a `TextView` no centro do layout.



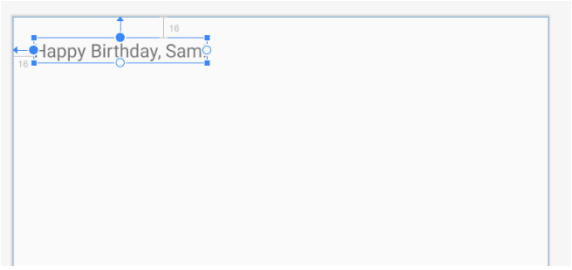
2. Pressione a tecla **Delete**. O Android Studio vai excluir a `TextView`, e o app passará a mostrar somente um `ConstraintLayout` no **Layout Editor** e na **Component Tree**.



Dica: caso queira aumentar o zoom no layout, use os controles no canto inferior direito do **Layout Editor** para ajustar o tamanho. Clique no último ícone para retornar a um nível de zoom em que o layout inteiro caiba na tela.

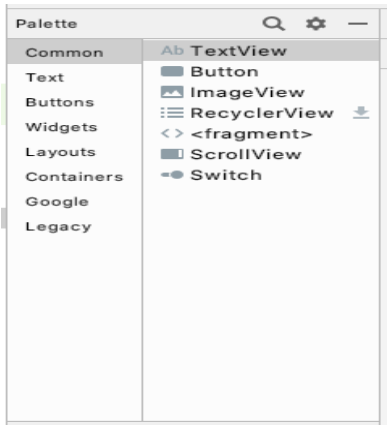
Adicionar uma TextView

Nesta etapa, você vai adicionar uma `TextView` no canto esquerdo de cima do app para conter a mensagem de aniversário.

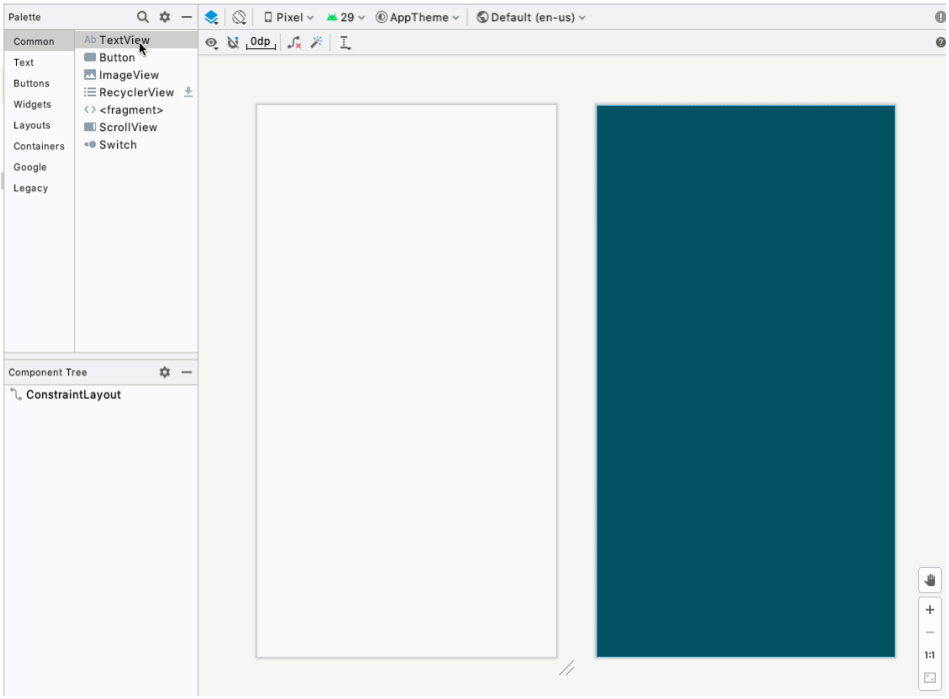


A **Palette** no canto superior esquerdo do **Layout Editor** contém listas de diferentes tipos de `Views`, organizadas por categoria, que podem ser adicionadas ao app.

1. Localize a `TextView`. Ela aparece tanto na categoria **Common** quanto na categoria **Text**.



2. Arraste uma `TextView` da **Palette** para o canto superior esquerdo da plataforma de design no **Layout Editor** e solte. Não é necessário fazer um posicionamento exato, basta soltar a TextView perto do canto esquerdo de cima.

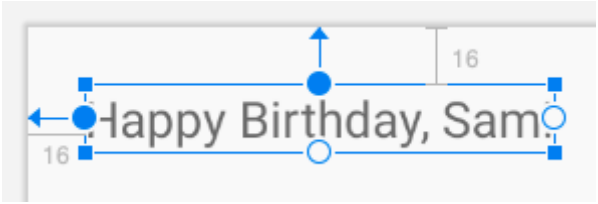


3. Observe que existe uma TextView adicionada e um ponto de exclamação vermelho na **Component Tree**.
4. Passe o cursor sobre o ponto de exclamação para conferir uma mensagem de alerta informando que a visualização não está restringida e vai mudar para uma posição diferente quando o app for executado. Vamos corrigir isso na próxima etapa.

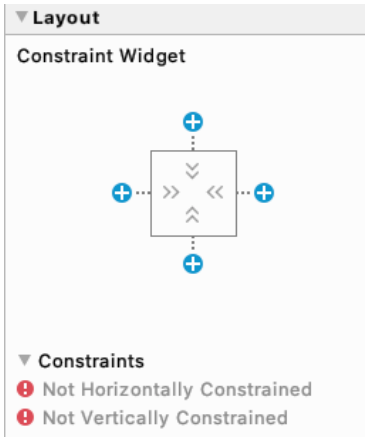
Posicionar a TextView

Para o cartão de aniversário, a TextView precisa estar perto do canto esquerdo de cima, com um pouco de espaço sobrando ao redor dela. Para corrigir o alerta, você vai adicionar algumas restrições à TextView, instruindo o app sobre a posição. Restrições são direções e limitações para o local em que uma View pode ficar no layout.

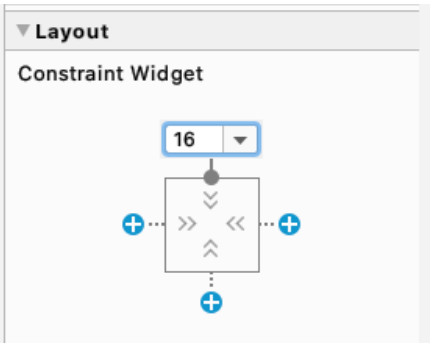
As restrições adicionadas à parte superior e à esquerda terão margens. A margem especifica a distância entre uma View e a borda do contêiner dela.



1. Em **Attributes**, à direita, localize o **Constraint Widget** na seção **Layout**. O quadrado representa a visualização.



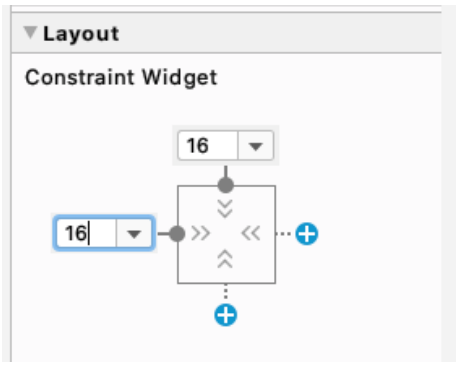
2. Clique em + na parte de cima do quadrado. Isso se aplica à restrição entre a parte de cima da TextView e a borda de cima do ConstraintLayout.
3. Um campo com um número é exibido para configurar a margem superior. A margem é a distância entre a `TextView` e a borda do contêiner, o `ConstraintLayout`. O número exibido varia dependendo da área em que você soltou a `TextView`. Ao definir a margem de cima, o Android Studio também adiciona automaticamente uma restrição da parte de cima da visualização de texto até a parte de cima do `ConstraintLayout`.



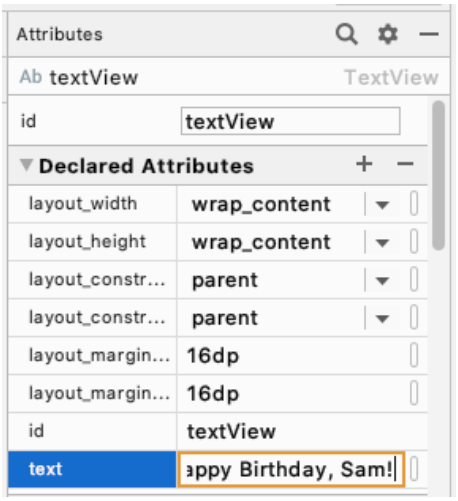
4. Mude a margem superior para 16.

Observação: a unidade de medida para margens e outras distâncias na IU é *pixels de densidade independente* (dp, na sigla em inglês). Essa unidade é como centímetros ou polegadas, mas é usada para distâncias na tela. O Android converte esse valor para o número adequado de pixels reais para cada dispositivo. Como valor de referência, 1 dp equivale a cerca de 1/160 de uma polegada, mas pode ser maior ou menor para alguns dispositivos.

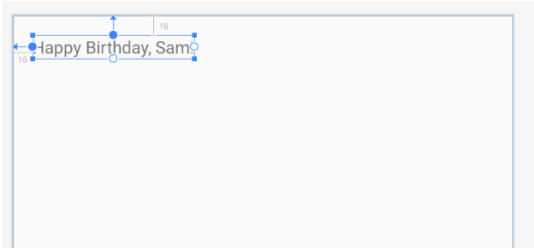
5. Faça o mesmo para a margem esquerda.



6. Defina um **text** para desejar feliz aniversário ao seu amigo, como "Happy Birthday, Sam!", e pressione **Enter**.



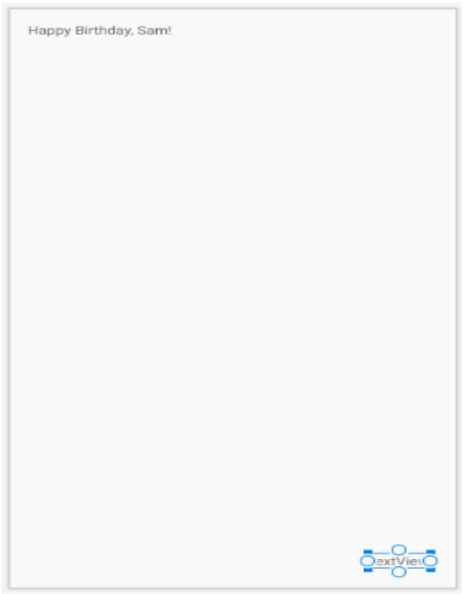
7. Observe que a visualização **Design** é atualizada para mostrar como o app vai ficar.



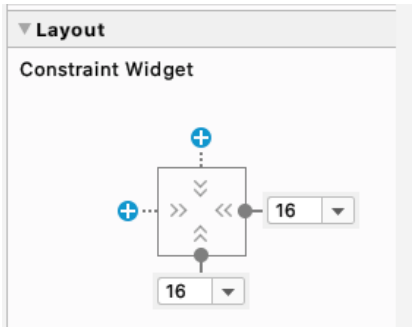
Adicionar e posicionar outra TextView

Seu cartão de aniversário precisa de uma segunda linha de texto no canto direito de baixo, que será adicionada nesta etapa da mesma forma que na tarefa anterior. Quais você acredita que seriam as margens dessa `TextView`?

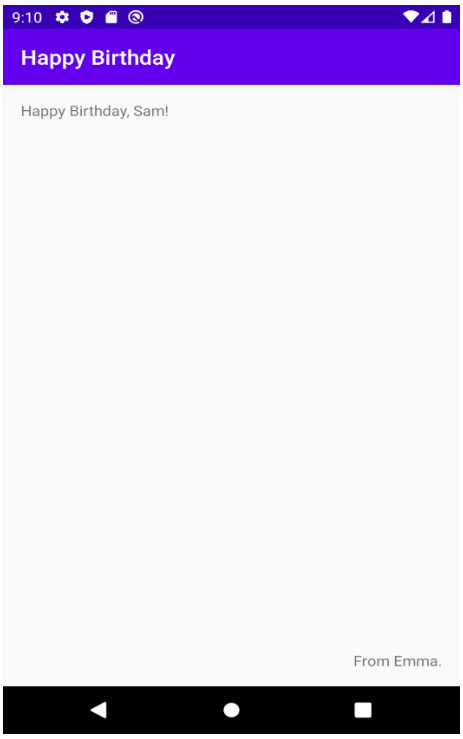
1. Arraste uma nova `TextView` da **Palette** e solte perto do canto direito de baixo da visualização do app no Layout Editor.



2. Defina a margem direita como 16.
3. Defina a margem de baixo como 16.



4. Em **Attributes**, defina o atributo **text** para assinar o cartão, por exemplo, "From Emma".
5. Execute o app. Sua mensagem de aniversário aparece no canto esquerdo de cima e a assinatura no canto direito de baixo.

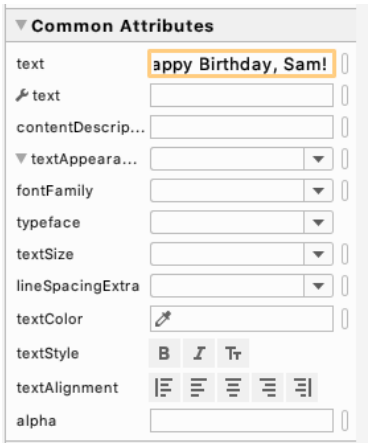


Parabéns! Você adicionou e posicionou alguns elementos de IU no seu app.

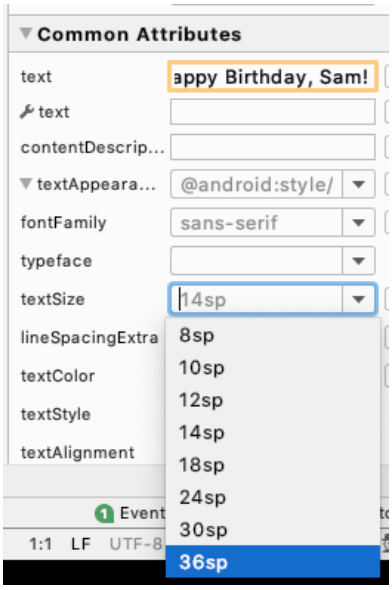
4. Adicionar estilo ao texto

Você adicionou texto à interface do usuário, mas ele ainda não está com a aparência do app final. Nesta tarefa, você vai aprender a mudar o tamanho e a cor do texto, além de outros atributos que afetam a aparência da `TextView`. Você também pode testar fontes diferentes.

1. Clique na primeira `TextView` na **Component Tree** e localize a seção **Common Attributes** da janela **Attributes**. Role para baixo até encontrar essa opção.
2. Observe os diversos atributos, incluindo **fontFamily**, **textSize** e **textColor**.

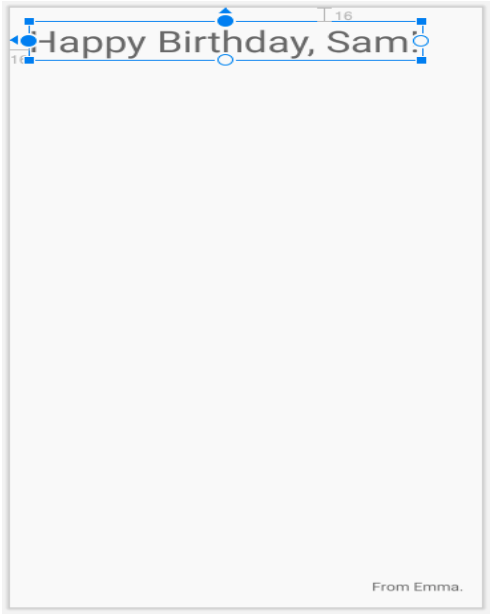


3. Localize **textAppearance**.
4. Caso **textAppearance** não esteja expandido, clique no triângulo de cabeça para baixo.
5. Em **textSize**, defina **textSize** como **36sp**.

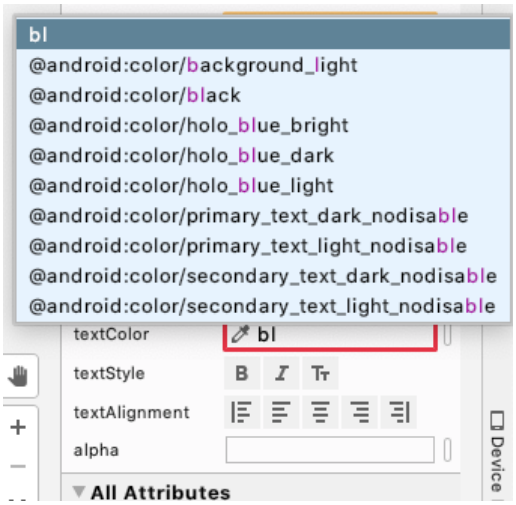


Observação: assim como dp é uma unidade de medida para distâncias na tela, **sp** é uma unidade de medida para o tamanho da fonte. Os elementos da IU em apps Android usam duas unidades de medida diferentes: *pixels de densidade independente* (**dp**), usada anteriormente para o layout, e *pixels escalonáveis* (**sp**, na sigla em inglês), usada ao definir o tamanho do texto. Por padrão, um sp é do mesmo tamanho que um dp, mas é redimensionado com base no tamanho de texto preferido do usuário.

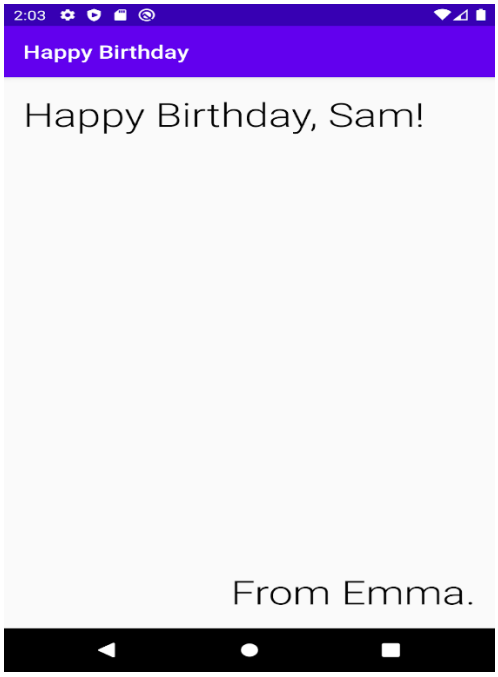
6. Observe a mudança no **Layout Editor**.



7. Mude **fontFamily** para **casual**.
8. Teste algumas fontes diferentes para saber como elas ficam. É possível encontrar ainda mais opções de fontes na parte de baixo da lista, em **More Fonts...**.
9. Quando terminar de testar fontes diferentes, defina **fontFamily** como **sans-serif-light**.
10. Clique na caixa de edição do atributo **textColor** e comece a digitar **black**. Observe que, à medida que você digita, o Android Studio mostra uma lista de cores contendo o texto digitado até o momento.



11. Selecione **@android:color/black** na lista de cores e pressione **Enter**.
12. Na `TextView` contendo sua assinatura, mude **textSize**, **textColor** e **fontFamily** para que sejam iguais.
13. Execute o app e confira o resultado.



Parabéns! Você aprendeu o básico para criar um app de cartão de aniversário.