Roteiro de Laboratório 02

José Cassiano Gunji

Dando continuidade ao desenvolvimento de aplicativos Android, vamos agora criar um aplicativo com interface responsiva e que realiza alguns cálculos.

1. Calculadora de Gorjetas

Vamos criar um aplicativo que calcule rapidamente o valor de gorjetas típicas (5, 10 e 15%) assim como uma gorjeta personalizada. Desta vez, nosso aplicativo vai possuir comportamento além do *layout* da *activity*. Vamos separar o processo de elaboração do aplicativo em duas partes. Primeiro vamos criar o projeto gráfico do aplicativo, o qual será responsivo. A seguir, vamos criar o código que fará com que o aplicativo execute as suas tarefas.

> É importante sempre criar interfaces gráficas responsivas, ou seja, que são capazes de se adequar aos diversos dispositivos em que serão utilizadas, nas diversas condições e cenários de uso. Por exemplo, uma interface gráfica responsiva deve funcionar igualmente bem seja ela utilizada em um smartfone, em um *tablet*, em um *notebook*, na orientação paisagem ou retrato, usando tela de toque, caneta digitalizadora, tela de toque ou o que mais for desenvolvido.

1.1. *Layout* do aplicativo Calculadora de Gorjetas

Inicie o seu novo projeto no Android Studio do mesmo modo que fez no projeto anterior. Crie um novo projeto com o modelo *Empty Activity*, defina o nome do projeto como “Calculadora de Gorjeta”, o nome do pacote como “br.unip.calculadoraDeGorjeta”, mantenha a linguagem como “Java” e certifique-se de que o SDK mínimo seja API 16 (*Jelly Bean*).

O assistente cria uma *Activity* com o *TextView “Hello World!”*. Selecione o *TextView* no editor gráfico ou na árvore de componentes e exclua-o.

Nossa *Activity* poderá ser grande demais para ser apresentada por inteiro na tela de dispositivos menores ou se o dispositivo for girado para a orientação paisagem. Assim, vamos criar um projeto gráfico que utiliza uma barra de rolagem vertical. Arraste o Container *ScrollView* para o *RelativeLayout* da árvore de componentes como mostrado na figura.

Ao criar seu aplicativo, selecione como SDK mínimo a API 16. Afinal, você vai testar seu aplicativo em um emulador que está usando justamente a API 16.

Ao criar seu aplicativo, selecione como SDK mínimo a API 16. Afinal, você vai testar seu aplicativo em um emulador que está usando justamente a API 16.

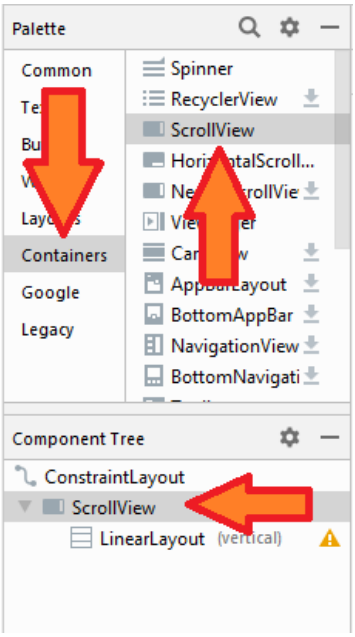

Figura 1 – Inserindo uma barra de rolagem ao *layout* do aplicativo

Para criar a interface gráfica, vamos utilizar o leiaute *TableLayout*, que organiza os componentes como uma tabela. Para tanto, comece apagando o *LinearLayout* criado automaticamente ao inserir o *ScrollView*. A seguir, arraste o *TableLayout* para dentro do *ScrollView* na árvore de componentes.

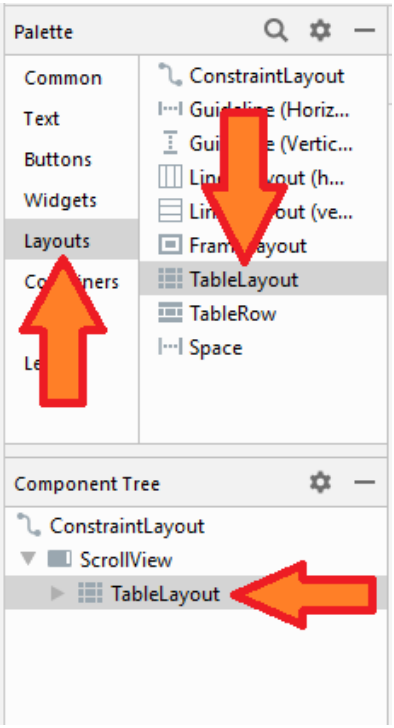

Figura 2 – Definindo o *layout* da interface gráfica como *TableLayout*

Deve-se sempre criar interfaces gráficas responsivas, não importa se o aplicativo será *desktop*, móvel ou *web*. Todo aplicativo deve se adaptar às mais variadas modalidades de visualização. Isso é particularmente importante para dispositivos móveis, pois eles se apresentam em uma variedade enorme de tamanhos, resoluções e combinações entre os dois. Um aplicativo deve se comportar de maneira semelhante em qualquer dispositivo compatível.

O componente *TableLayout* irá redimensionar e reposicionar os componentes gerenciados por ele de maneira automática sempre que a tela mudar de tamanho (ao ser instalado em dispositivos diferentes) ou se a tela mudar de orientação (entre retrato e paisagem). Para tanto, precisamos indicar ao *TableLayout* quais as colunas ele pode esticar para ocupar toda a sua área disponível. Primeiro, com o *TableLayout* selecionado, expanda a seção *"All Attributes"* para exibir todos os atributos. A seguir, vamos modificar sua propriedade *strechColumns* para o valor "1, 2, 3" para indicar que a segunda, terceira e quarta colunas podem ser esticadas (a numeração das colunas inicia em zero, como em qualquer *array* ou coleção em Java).

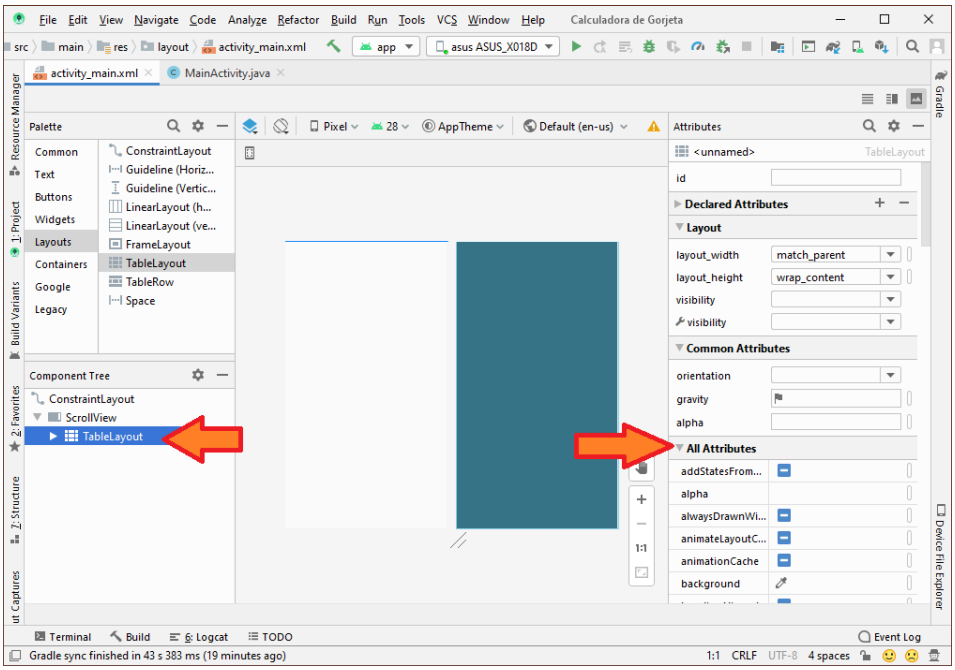

Figura 3 – Exibindo todos os atributos do *TableLayout*

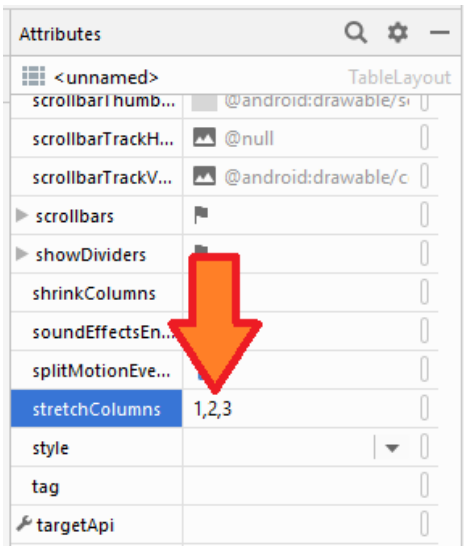

Figura 4 – Definindo as colunas que podem ser esticadas no leiaute

A interface gráfica que projetamos apresenta 7 linhas de componentes. Assim, vamos arrastar sete *TableRow* para dentro do *TableLayout* na árvore de componentes. Também vamos alterar a propriedade *id* de cada *TableRow* para atribuir um nome único a cada um. Atribua os nomes *tableRow1*, *tablerRow2*, ..., *tableRow7*. Sua árvore deve apresentar a estrutura mostrada na figura:

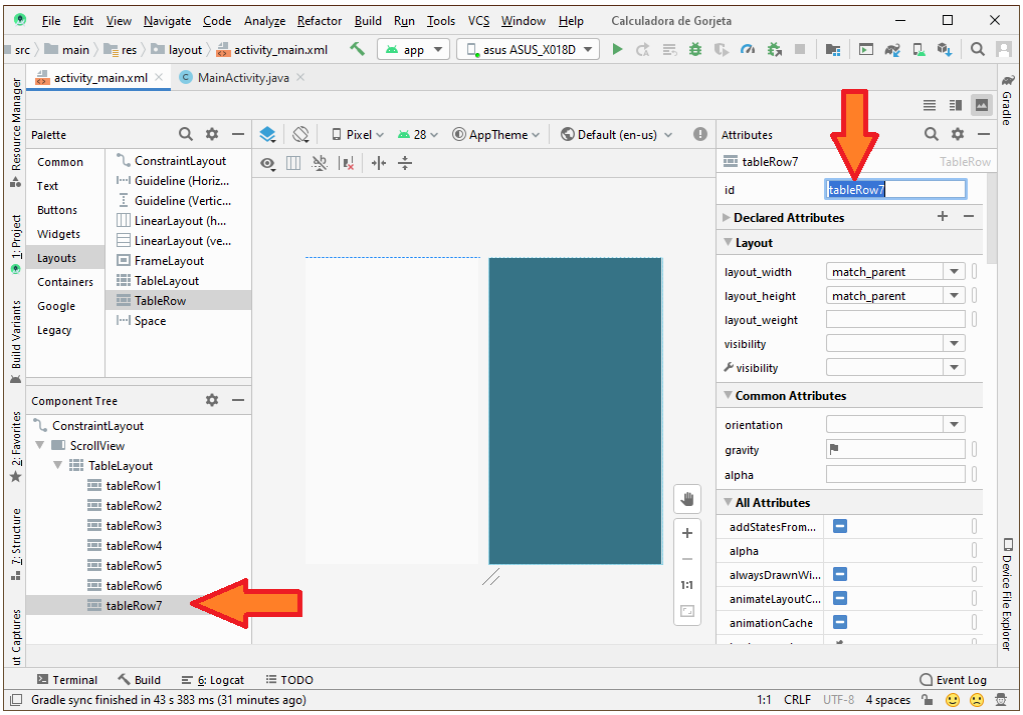

Figura 5 – Inserindo sete linhas ao *layout* de tabela

Na *tableRow1*, adicione um *TextView*. Em sua propriedade *text*, ao invés de definir o texto “Conta” diretamente, vamos criar uma entrada no arquivo *string.xml* com o valor que será apresentado no *TextView*. Para tanto, clique no botão ao lado do espaço de texto na propriedade *text.*

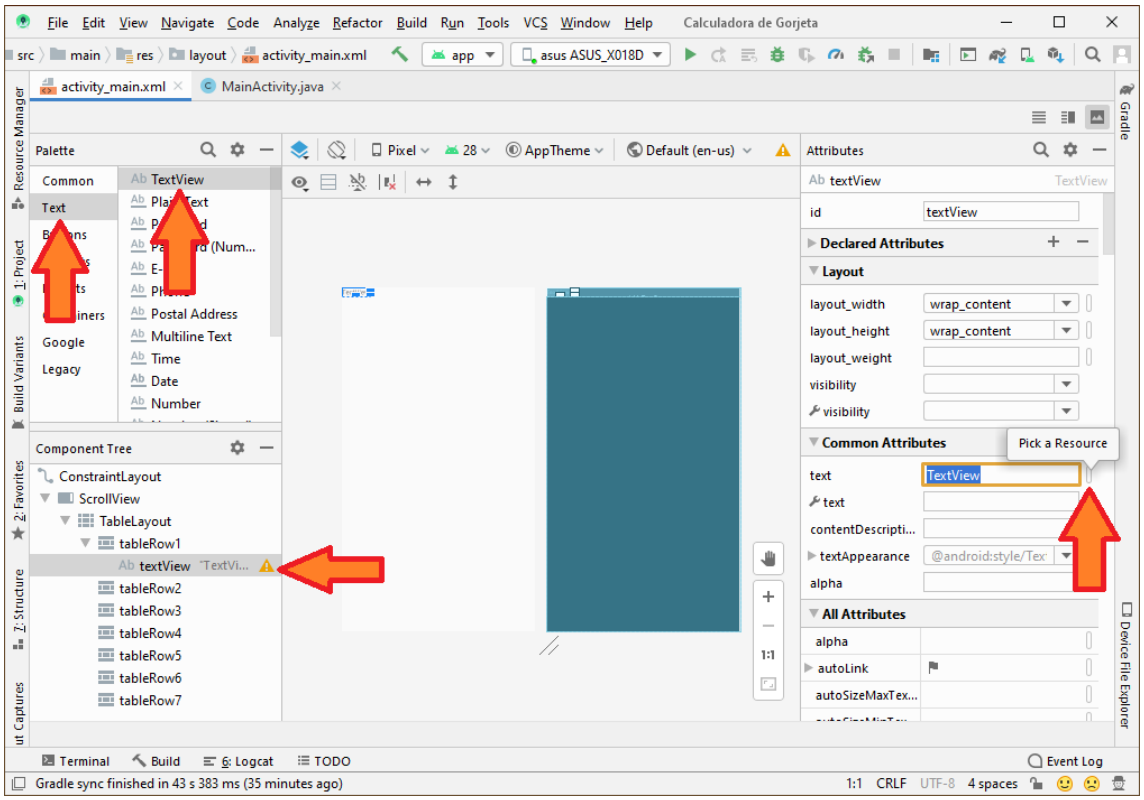

Figura 6 – Definindo um recurso de texto ao *TextView*

Na janela que se abre, clique no botão “+” e selecione *“String value”*.

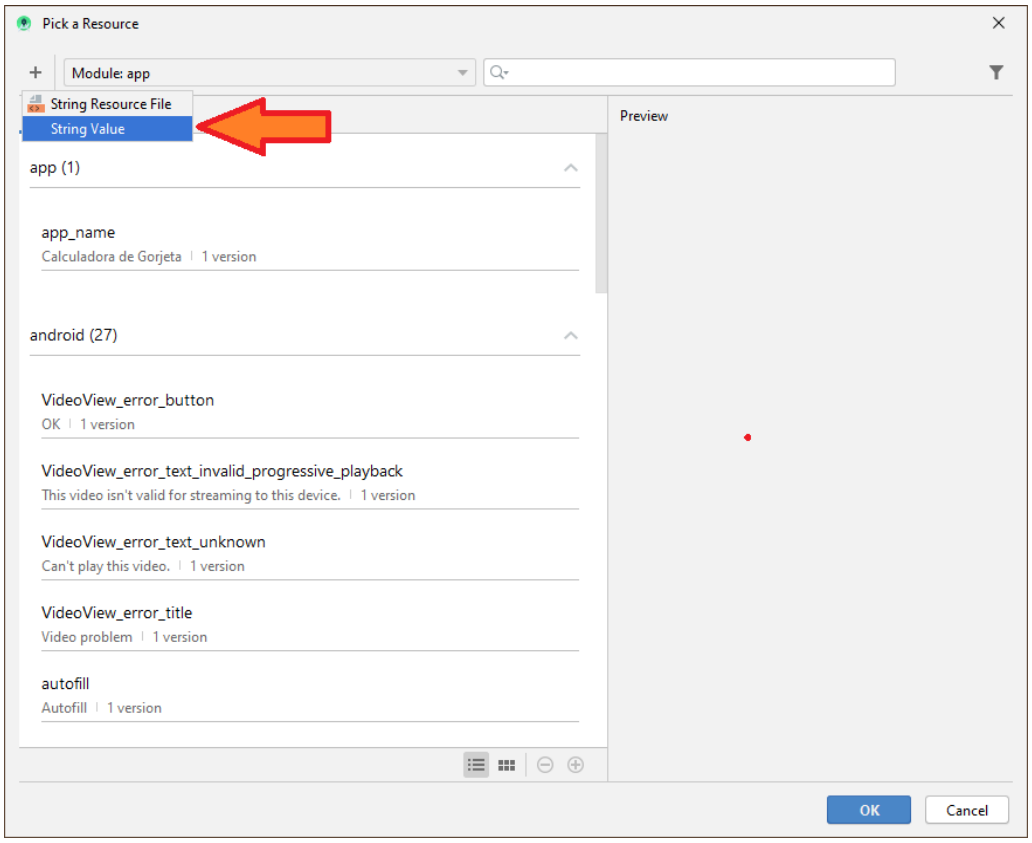

Figura 7 – Acrescentando um novo recurso de texto

Na janela que se abre, defina o nome do recurso como “conta” e o valor do recurso com “Conta”.

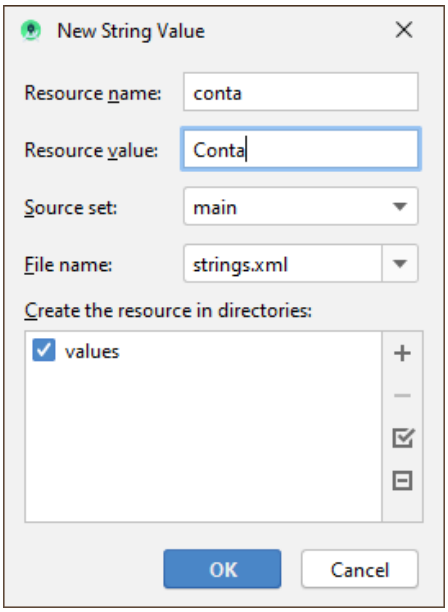

Figura 8 – Definindo recursos de texto

Clique em *“OK”* duas vezes e veja o resultado de sua edição. À primeira vista pode parecer uma maneira complicada e desajeitada de se definir o texto de um *TextView*. A razão para se fazer isso é que o *TextView* só precisa saber o nome do recurso de texto que irá apresentar, que no caso é *“@string/conta”*. Já o valor deste recurso, por enquanto é “Conta”, definido no arquivo *“string.xml”*. A vantagem, agora, é que é bastante fácil criar traduções dos textos presentes neste arquivo para diversos idiomas diferentes. Quando isto é feito, o seu aplicativo irá se configurar automaticamente para o idioma do sistema operacional onde será instalado, claro, desde que a tradução para tal idioma tenha sido feita.

Abra o arquivo *“string.xml”* e confira o recurso de texto que você acabou de criar.

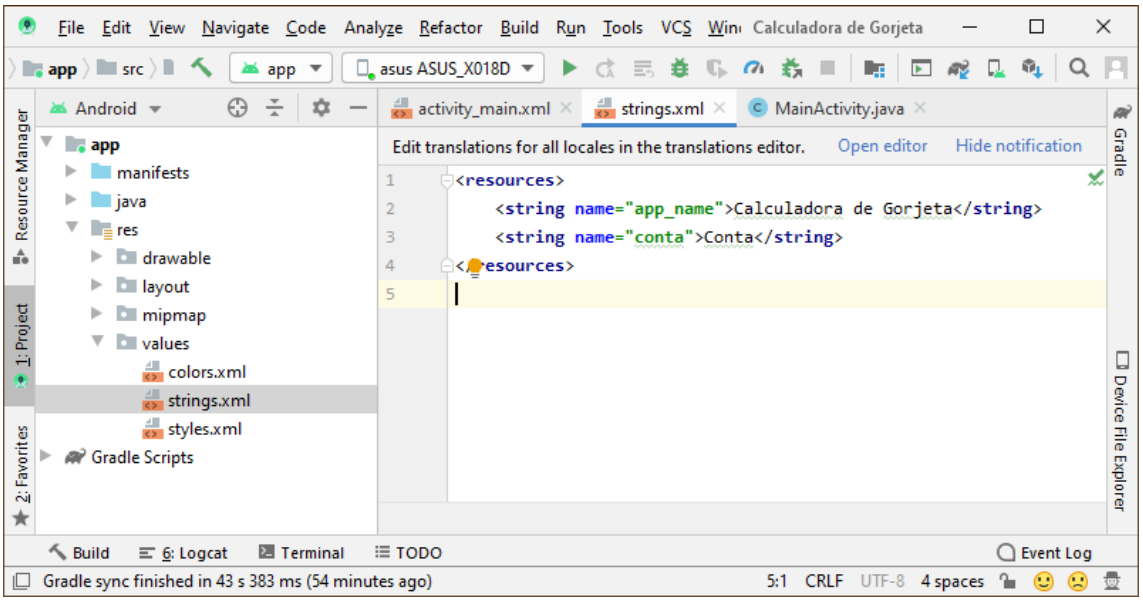

Figura 9 – Exibindo recursos de texto no arquivo *"strings.xml"*

Adicione agora ao *tableRow1* um *Tex Number* (*Decimal*). Defina sua *id* como "*contaEditText*". Defina também sua propriedade *ems* para nenhum valor. Um valor na propriedade *ems* indica um tamanho fixo para o *EditText*, o que acaba fazendo com que o layout acabe ultrapassando os limites da tela

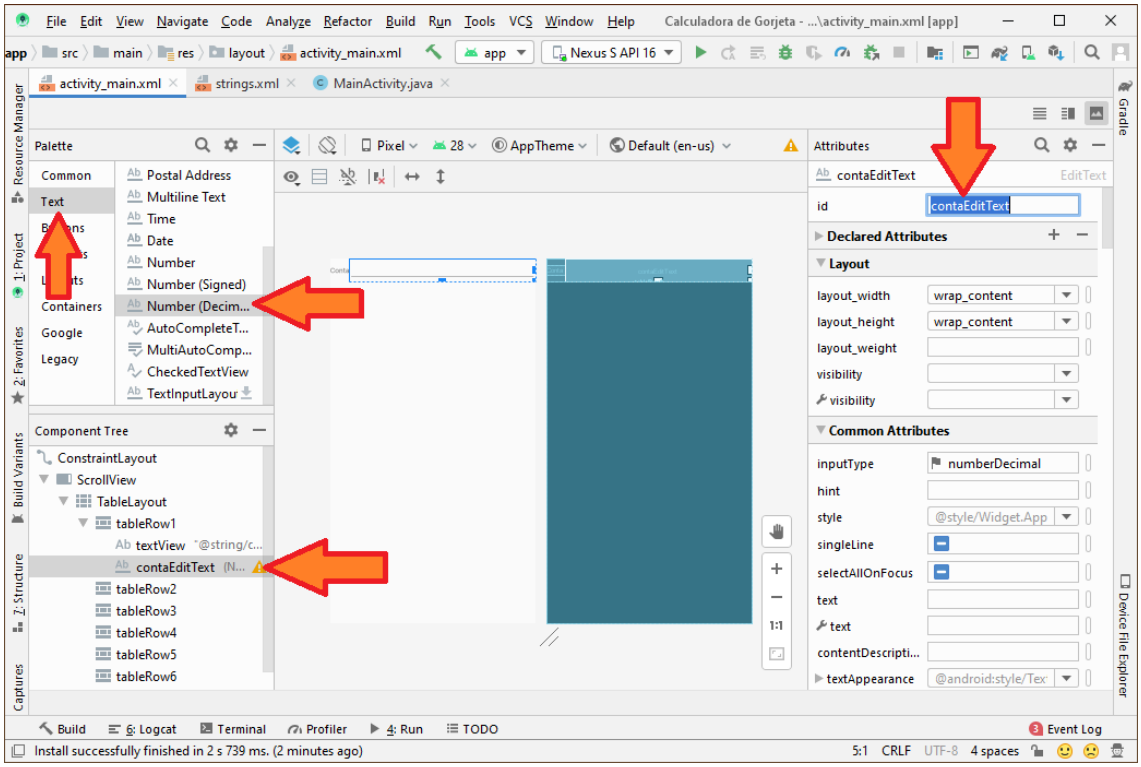

Figura 10 – Configurando o campo de texto numérico para receber o valor da conta

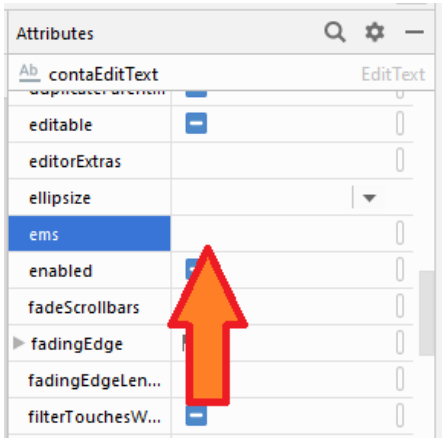

Figura 11 – Definindo o atributo *ems* de *contaEditText*

Além disso, queremos que o *contaEditText* ocupe o espaço total das três colunas restantes da tabela. Assim, vamos definir a sua propriedade *layout_span* para o valor 3.

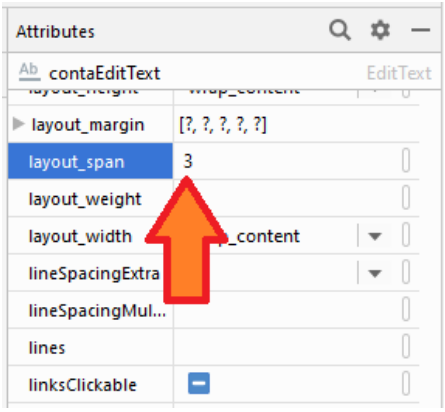

Figura 12 – Definindo a abrangência da célula *contaEditText*

Na *tableRow2*, adicione um *textView*. Em sua propriedade *text*, crie uma nova entrada no arquivo *strings.xml* chamada *"explicacao1"* com o valor "Abaixo estão os valores típicos de gorjeta". Defina sua propriedade *layout_span* para 4, de modo que ela ocupe as quatro colunas da tabela.

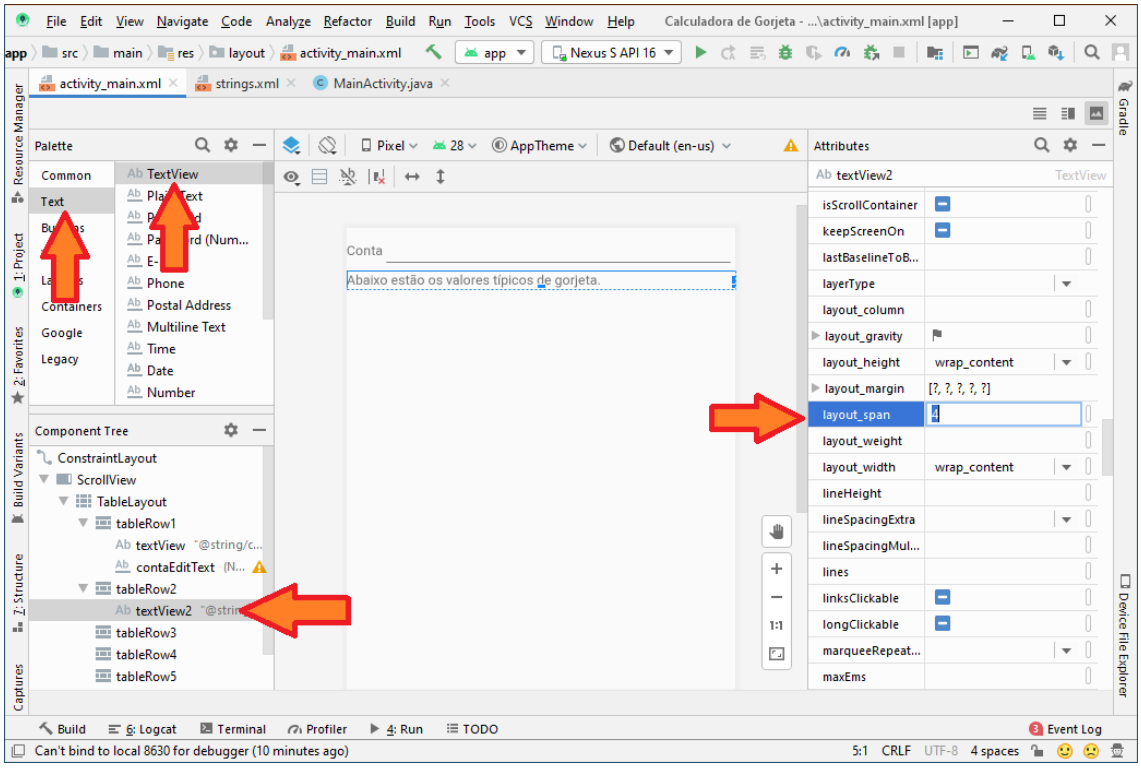

Figura 13 – Inserindo o primeiro texto explicativo

Na *tableRow3*, adicione um *TextView* com uma nova entrada no arquivo strings.xml chamada “percentual” com o valor “Percentual”.
Adicione três *Text Fields Number (Decimal)*, com os textos “5”, “10” e “15”. Estes textos podem ser definidos diretamente na propriedade text de cada EditText, pois nunca seriam traduzidos.

Selecione os três *EditText* clicando neles na árvore de componentes mantendo a tecla CTRL pressionada. Isso permite que você selecione mais de um componente simultaneamente. Com isso, a palheta de propriedades exibe apenas as propriedades em comum que os componentes possuem. Altere a propriedade *ems* das três para nada.

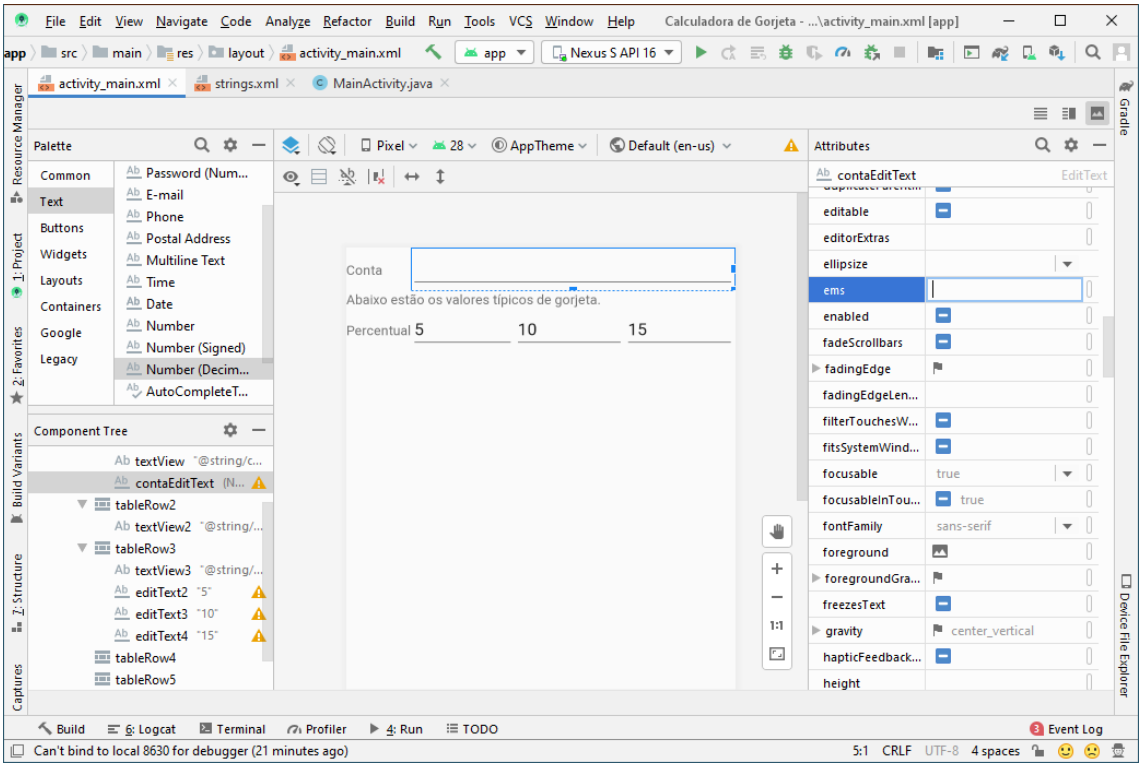

Figura 14 – Inserindo os elementos de *tableRow3*

Na *tableRow4*, crie um *TextView* com uma nova entrada no arquivo *strings.xml* chamada “gorjeta” com o valor “Gorjeta”. Adicione *três Text Fields Number (Decimal)*. Eles receberão os valores calculados das gorjetas padrão, por isso devem ser nomeados como ngorjeta5EditText, gorjeta10EditText e gorjeta15EditText. Não se esqueça de apagar o valor da propriedade *ems* deles.

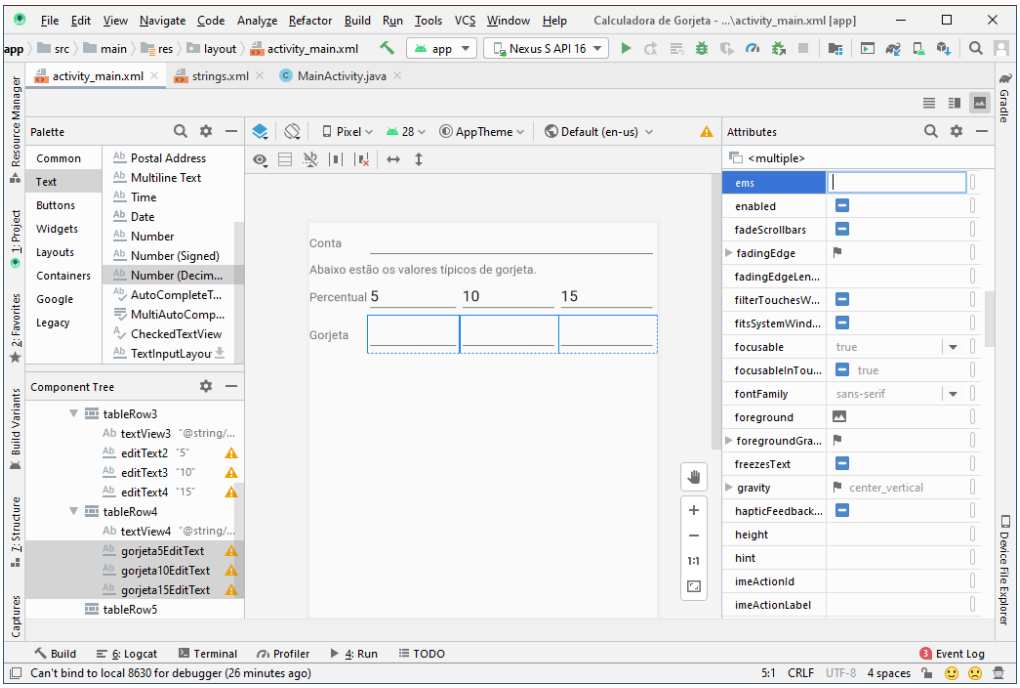

Figura 15 – Inserindo os elementos de *tableRow4*

Ao *tableRow5*, adicione um *TextView* com uma nova entrada no arquivo *strings.xml* de nome “explicacao2” e valor “Defina uma gorjeta de valor personalizado”. Defina seu layout:span para 4.

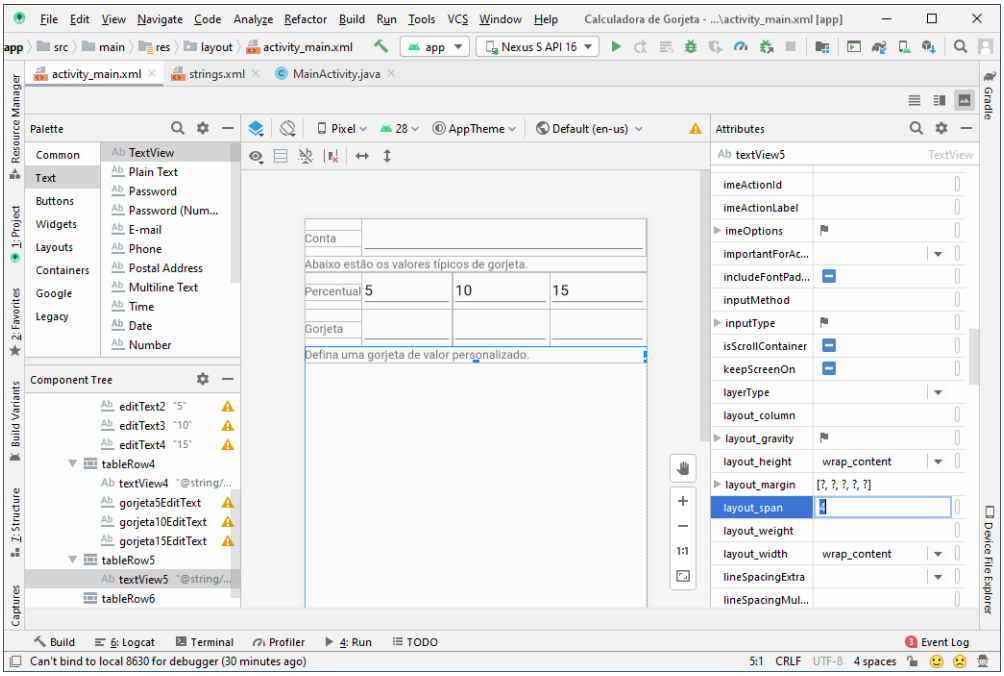

Figura 16 – Inserindo a segunda linha de explicação

Adicione ao *tableRow6* um *TextView*. Desta vez não é preciso definir uma nova entrada no arquivo *strings.xml*. Ela já existe e vamos reutiliza-la. Clique no botão ao lado da propriedade *text*. Escolha a entrada “percentual” de seu arquivo *strings.xml* do seu projeto.

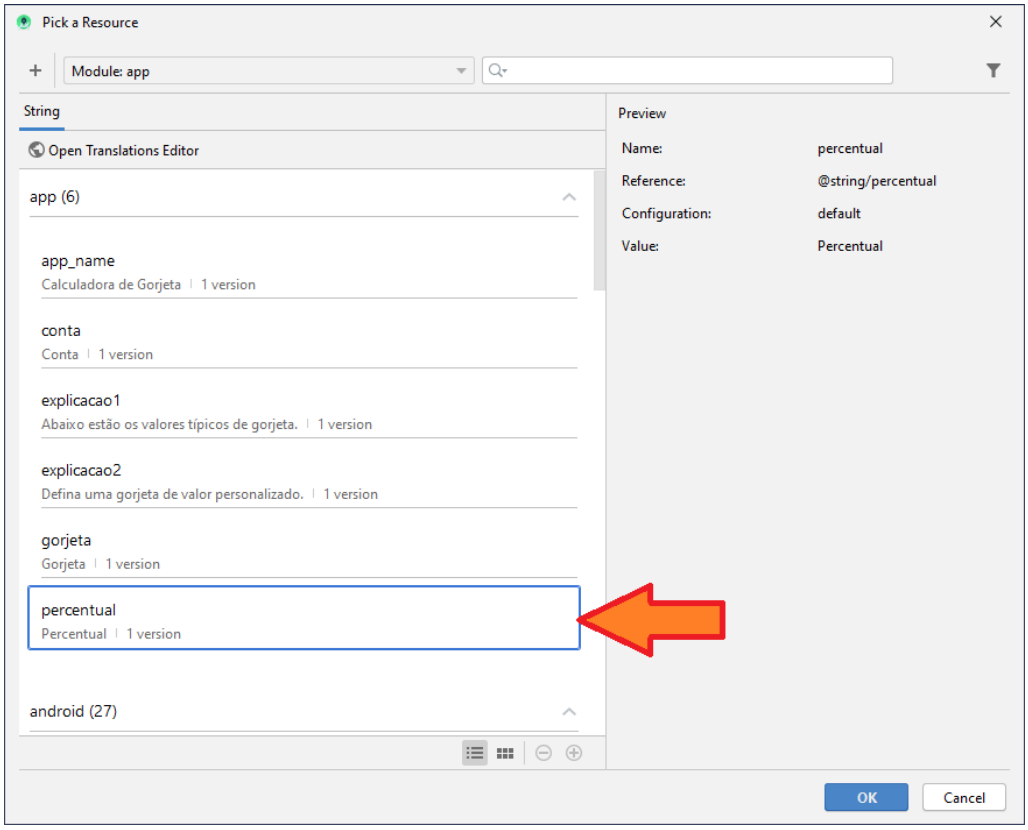

Figura 17 – Selecionando um recurso de texto preexistente

Adicione um *Widgets:SeekBar* ao *tableRow6*. Defina sua *id* como *percentualSeekBar*. Altere sua propriedade *layout_span* para 2.

Adicione um *Text Field Number (Decimal)*. Defina sua *id* para *percentualEditText* e apague o valor de sua propriedade *ems*.

Na *tableRow7*, adicione um *TextView* com a entrada “gorjeta” do arquivo *strings.xml*. Adicione um *Text Field Number (Decimal)*. Defina seu *layout_span* para 3 e apague sua propriedade *ems*. Defina seu id para *gorjetaEditText*.

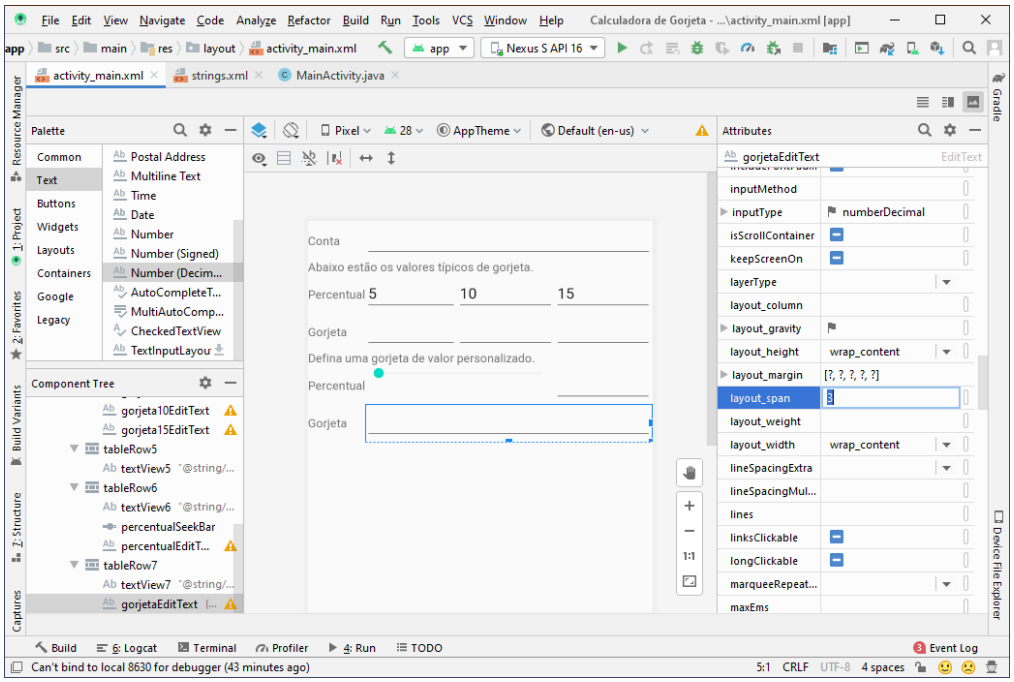

Figura 18 – Completando o *layout* do aplicativo

Vamos mudar a cor e o tamanho dos *textViews*. Selecione-os todos mantendo a tecla [CTRL] pressionada. Altere a propriedade *textColor* para #000000 (preto) e a propriedade *textSize* para 18sp. Como cada dispositivo Android possui um tamanho de tela diferente e uma densidade de pixels diferente, é muito difícil definir tamanhos por pixels absolutos. Assim, o Android define dois tipos de pixels que não dependem da resolução da tela. Eles são os pixels independentes da escala (sp), que são úteis para definir tamanhos de fontes e os pixels independentes da densidade (dp), que são úteis para todo o resto. Assim, com estes tipos de pixels, sua interface gráfica terá a mesma aparência em qualquer dispositivo.

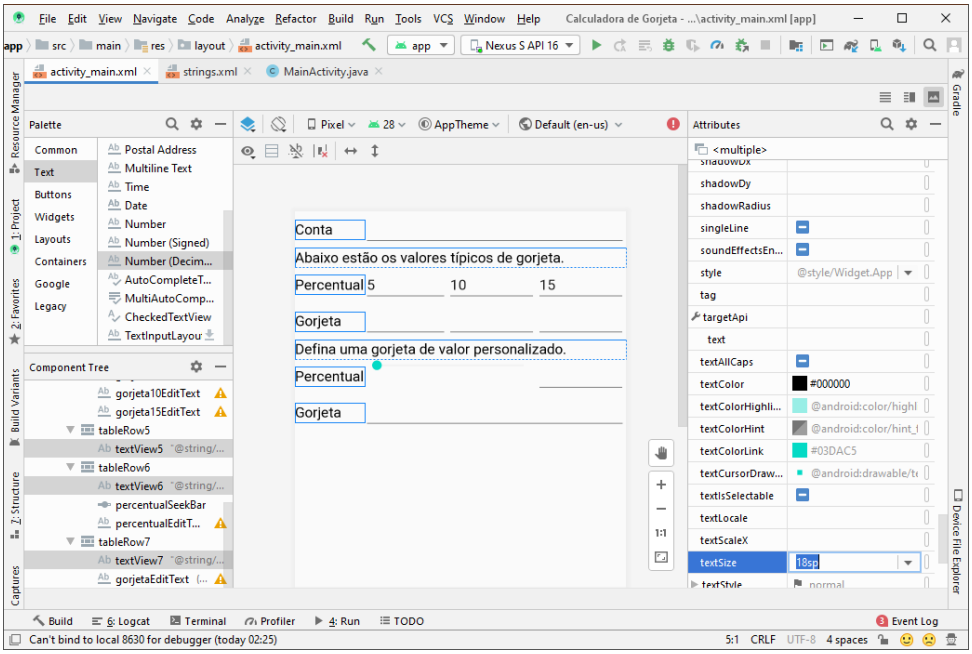

Figura 19 – Definindo a cor e o tamanho dos *textViews* simultaneamente

As unidades de *pixel* do Android são outra maneira que a plataforma oferece para facilitar o trabalho do desenvolvedor ao criar interfaces gráficas responsivas, coerentes e consistentes. As unidades *dp* e *sp* são as mais importantes, mas não são as únicas. Você pode saber mais nestes documentos:

COMPATIBILIDADE com densidades de pixel diferentes. *Developers*. [s.d.]. Disponível em: https://developer.android.com/training/multiscreen/screendensities?hl=pt-br. Acesso em: 10 ago. 2020.

DIFFERENCE Between dp, dip, sp, px, in, mm, pt in Android. *Javapapers*. 2014. Disponível em: https://javapapers.com/android/difference-between-dp-dip-sp-px-in-mm-pt-in-android/. Acesso em: 10 ago. 2020.

Selecione todos os *editTexts* e altere a sua propriedade *gravity* para *center*.

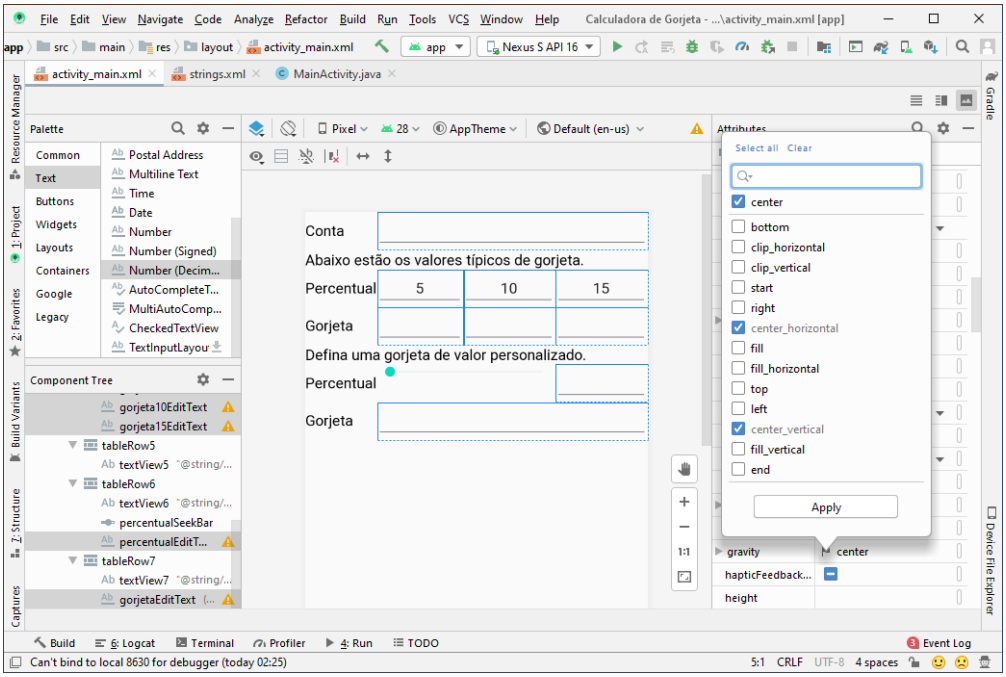

Figura 20 – Alterando a gravidade dos *editTexts*

Conforme você for criando o seu *layout*, tenha certeza de que a visualização está saindo como esperado. Se seu visualizador estiver diferente, vale a pena conferir os últimos passos para ver se você se enganou ou se esqueceu de algum ponto.

Remova da seleção o *contaEditText* e defina o atributo *focusable* como *false*. Isso evita que o usuário seja capaz de alterar os valores de todos os *editTexts* menos o primeiro.

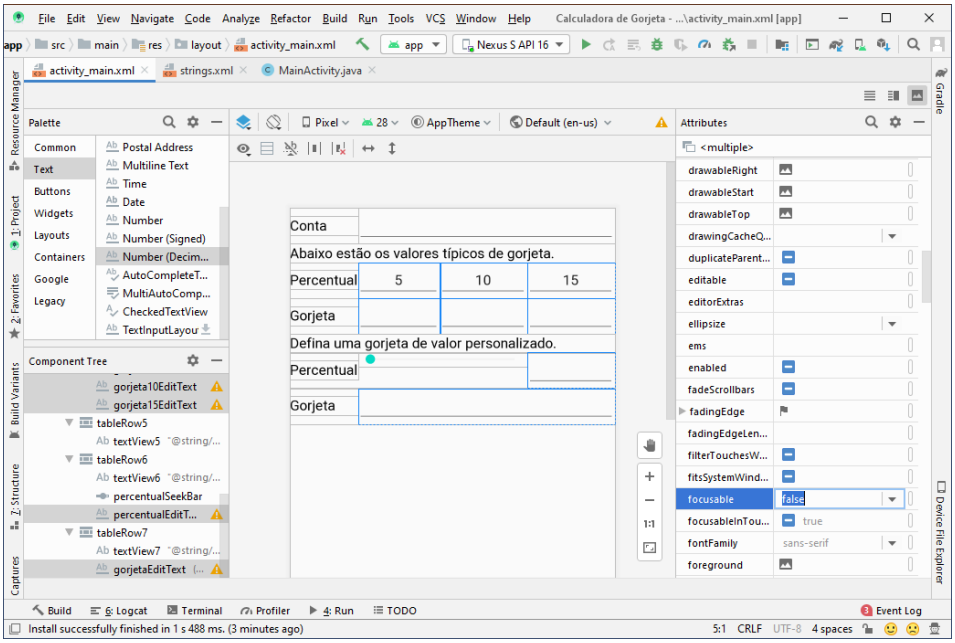

Figura 21 – Protegendo os *editTexts* de edições do usuário

Selecione o *percentualSeekBar* e altere o seu atributo *layout_gravity* para *center_vertical*, de modo que ele fique centralizado na direção vertical da linha do leiaute.

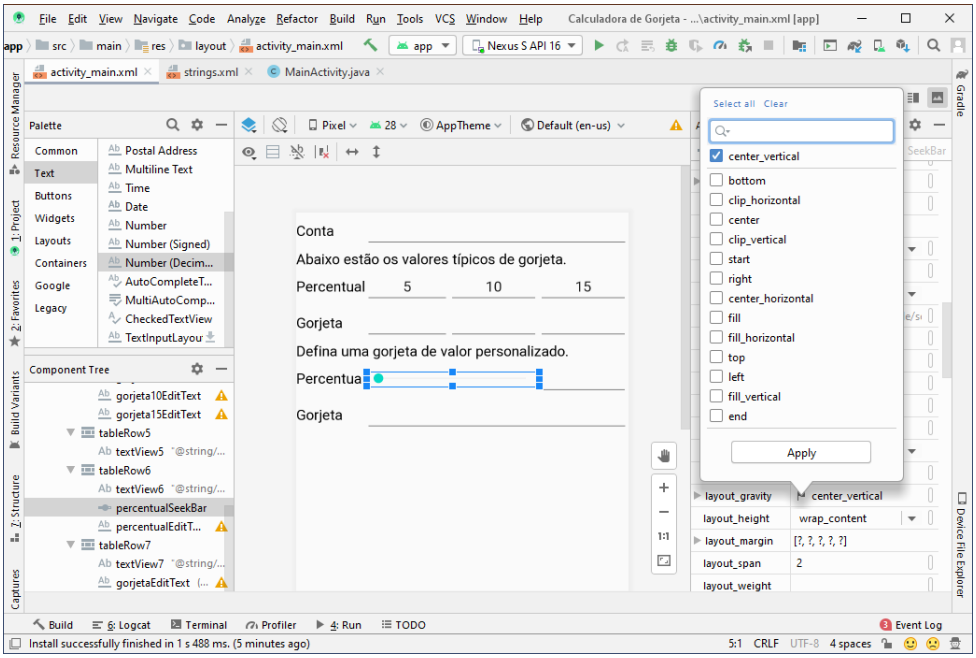

Figura 22 – Centralizando o *percentualSeekBar*

Clique no botão para mudar a visualização para a orientação paisagem e verifique se seu leiaute se comporta de maneira responsiva.

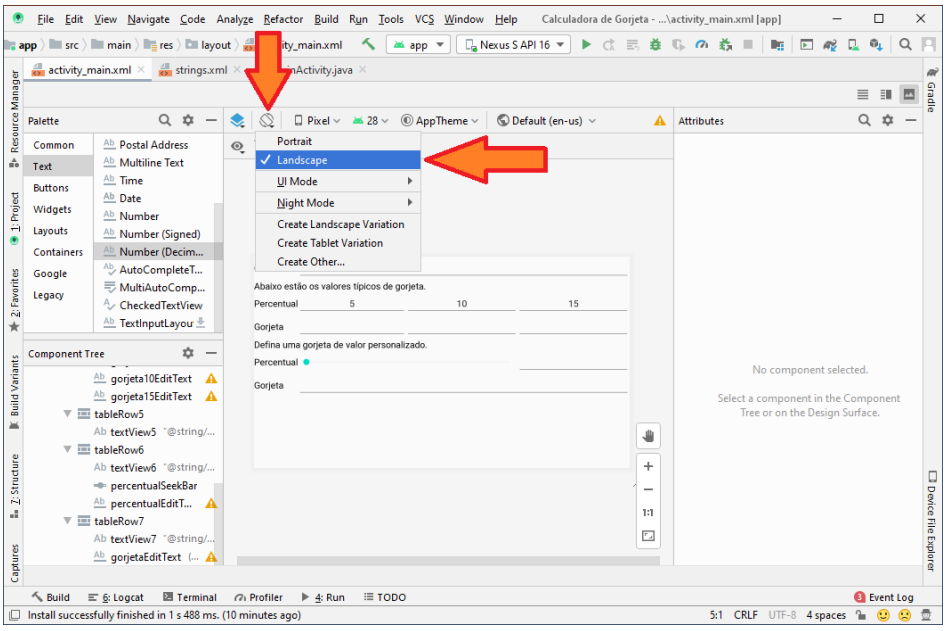

Figura 23 – Testando a responsividade de seu aplicativo