Capítulo 4

ETANOL OU GASOLINA?

Se você possui um veículo *flex*, deve ter reparado na diferença flutuante entre os valores dos combustíveis etanol e gasolina, e deve ter se perguntado com qual combustível deveria abastecer. Sabendo que, com um litro de etanol, você roda uma quilometragem menor do que com um litro de gasolina, qual seria o mais vantajoso? Neste capítulo, construiremos um aplicativo para lhe ajudar a decidir entre abastecer com etanol ou gasolina.

Nesse aplicativo, teremos uma tela de interação para a digitação dos valores dos dois combustíveis e um botão para realizarmos os cálculos e exibirmos qual é a melhor opção de abastecimento. Além disso, teremos também um botão para limpar os dados digitados e um para acessar uma segunda tela, para que o usuário saiba como é feito o cálculo para a escolha do combustível.

4.1 TELA DO APLICATIVO

Veja na figura a seguir como será a tela final do nosso aplicativo:

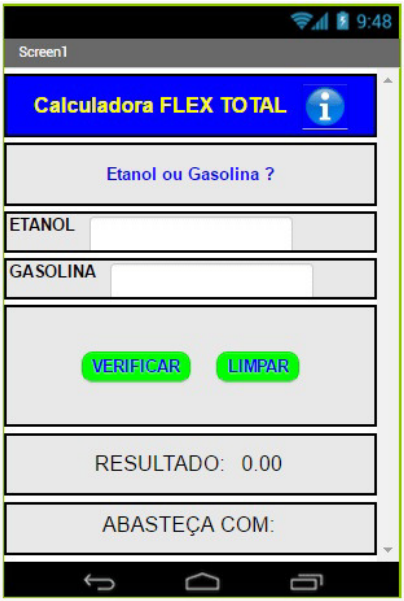

Figura 4.1: Tela final do aplicativo Calculadora Flex Total

Para começar, vamos criar um novo projeto. Para isso, clique no menu `Projects`, selecione a opção `Start New Project`, e dê o nome de `Etanol_ou_Gasolina`.

Vamos começar a desenvolver nossa tela inicial. Como vimos na figura, necessitamos de três **botões**: um para mostrar a fórmula do cálculo; outro para realizar os cálculos e exibir o resultado desejado; e um para limpar os dados da tela para realizarmos um novo cálculo.

Necessitamos também de duas **caixas de texto** para a digitação dos valores do etanol e da gasolina, que serão utilizadas para a realização dos cálculos. Por último, vamos precisar também de algumas `Labels` para exibir os valores resultantes das contas e da

decisão realizada pelo aplicativo.

Veja na tabela adiante os componentes que devemos inserir, a guia em que eles se encontram e as propriedades que deverão ser alteradas para conseguirmos construir a tela do aplicativo.

Observe que, na coluna **Componente**, estão todos os itens que devemos arrastar para a tela do aplicativo. Já a coluna **Pallete** nos indica o local onde encontramos os componentes, que estão agrupados por categorias de funcionalidades. Por exemplo, para acessar a `Pallete` `Layout` e encontrar um componente `HorizontalArrangement`, basta clicar sobre o nome dessa `Pallete` e todos os componentes responsáveis pelo ajuste do layout estarão visíveis.

A coluna **Propriedade** indica qual propriedade deverá ser alterada, pois nem sempre precisamos alterar todas as propriedades de um componente. Já a coluna **Alterar valor** indicará qual o novo valor que cada uma deverá receber para chegarmos à tela final do nosso aplicativo.

Antes de inserir os componentes, perceba que já os utilizamos e explicamos no capítulo anterior. Com isso, o leitor já deverá estar mais familiarizado com o ambiente de desenvolvimento. Mas vamos começar a desenvolver o layout de nosso app.

Com a sua `Screen1` selecionada, altere a propriedade `AppName` para **Etanol ou Gasolina?**, pois este será o nome exibido no aplicativo quando ele estiver instalado em seu dispositivo. Criaremos a nossa própria área de título do aplicativo, então para o App Inventor não exibir o título padrão, desmarque a opção na propriedade `TitleVisible`.

Com essas propriedades já alteradas, poderemos começar a desenvolver a barra de título de nosso app. Ela terá uma `Label` com o nome do aplicativo e um `Button` que abrirá uma segunda tela, com as informações sobre como é feito esse cálculo para a decisão entre os combustíveis. Então, para exibir dois ou mais componentes na mesma linha, o App Inventor necessita que insiramos um controle `HorizontalArrangement`. É ele quem faz o **arranjo horizontal** dos componentes na linha.

Após inserir um componente `HorizontalArrangement`, precisaremos realizar alguns ajustes em suas propriedades. Veja na tabela a seguir as propriedades que deverão ser alteradas e seus novos valores.

Componente	Pallete	Propriedade	Alterar valor
HorizontalArrangement	Layout	AlignHorizontal	Center
		AlignVertical	Center
		BackgroundColor	Blue
		Height	50 pixels
		Width	Fill Parent

A propriedade `AlignHorizontal` realiza o alinhamento dos componentes que ficarão em seu interior. Aqui selecionamos o alinhamento para que todos os componentes fiquem alinhados no centro da tela, por isso foi marcada a opção `Center`. Agora para um alinhamento referente ao seu espaço vertical, dentro do `HorizontalArrangement`, a propriedade `AlignVertical` também é marcada como `Center`.

A propriedade `BackgroundColor`, que indicará a cor de fundo da barra de título desenvolvida, deverá ficar com uma cor

diferente da utilizada no fundo, então escolhemos a cor `Blue` . Necessitamos também alterar os tamanhos da altura e largura que o `HorizontalArrangement` ocupará dentro da `Screen1` . Para alterar a altura da barra de título, usaremos na propriedade `Height` o valor de `50 pixels` e, para o seu comprimento, na propriedade `Width` , selecione a opção `Fill Parent` que a deixará ocupando todo o espaço horizontal do seu dispositivo.

Preparado o espaço para o título, devemos inserir em seu interior uma `Label` para exibir o título do aplicativo e um botão que, ao ser pressionado, exibirá uma outra tela com as informações do aplicativo. A tabela seguinte indica a ordem em que devemos inserir os componentes e as propriedades que deverão ser alteradas.

Componente	Pallete	Propriedade	Alterar valor
Label	User Interface	FontBold	Marcar
		FontSize	20
		Text	Calculadora FLEX TOTAL
		TextColor	Yellow
Button	User Interface	Height	40 pixels
		Width	40 pixels
		image	info.png
		Text	Apagar o texto
		Renomear	Btn_Informacoes

Para a `Label` , selecionamos a propriedade `FontBold` , pois ela deixará o título com o efeito de **negrito**. Para aumentar o tamanho do texto, modificamos a propriedade `FontSize` para o

valor de 20. A propriedade que exibe o texto que desejamos exibir é a `Text`, onde foi digitado o título **Calculadora FLEX TOTAL**. Em `TextColor`, alteramos a cor do texto para `Yellow`.

Já para o componente `Button`, foi definida a altura do botão para `40 pixels` através da propriedade `Height`. Para a largura do botão, também deixamos a propriedade `Width` para `40 pixels`. Inserimos a imagem `info.png` conforme demonstrado no capítulo anterior. Apagamos também a propriedade `Text` do botão para ele possuir somente a imagem. Não se esqueça de que alteramos o nome do componente para `Btn_Informacoes`.

Veja como ficou a barra de título de seu aplicativo:

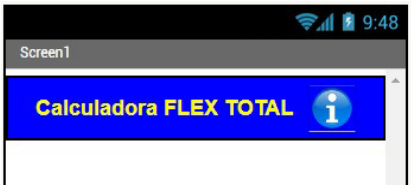

Figura 4.2: Barra de título do aplicativo

Continuando o desenvolvimento do layout, vamos agora exibir uma mensagem para o usuário: **Etanol ou Gasolina?**. Devemos primeiramente inserir um componente `HorizontalArrangement` logo abaixo da barra de título e, dentro dele, uma `Label` para realizar a exibição da mensagem.

Após inserir os objetos na sua `Screen1`, realize as configurações conforme demonstrado a seguir.

Componente	Pallete	Propriedade	Alterar valor
HorizontalArrangement	Layout	AlignHorizontal	Center
		AlignVertical	Center
		Height	50 pixels
		Width	Fill Parent
Label	User Interface	FontBold	Marcar
		FontSize	16
		Text	Etanol ou Gasolina?
		TextColor	Blue

A figura a seguir exibe o atual estágio da tela do app:

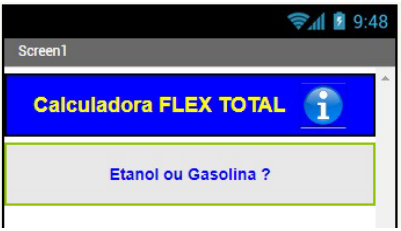

Figura 4.3: Mensagem: Etanol ou Gasolina?

Devemos agora preparar a digitação do valor do **Etanol**. Precisaremos de uma `Label` para informar a legenda do que devemos digitar e de um componente `TextBox` que será responsável por receber o valor digitado pelo usuário. Não podemos nos esquecer que tanto a `Label` como a `Textbox` deverão ser inseridas dentro de um `HorizontalArrangement` .

Comece inserindo o componente `HorizontalArrangement` da guia `Layout` logo abaixo do `HorizontalArrangement` que contém a mensagem *Etanol ou Gasolina?*, e realize as configurações que estão na tabela a seguir.

Componente	Pallete	Propriedade	Alterar valor
HorizontalArrangement	Layout	Width	Fill Parent
		AlignVertical	Center
Label	User Interface	FontBold	Marcar
		FontSize	16
		Text	ETANOL
TextBox	User Interface	Hint	Preço do Etanol
		NumbersOnly	Marcar
		TextAlignment	Right
		Renomear	Txt_Etanol

Vamos comentar apenas as propriedades que ainda não utilizamos, pois as demais continuam a realizar as mesmas funções já vistas anteriormente. No componente `HorizontalArrangement`, alteramos a propriedade `AlignVertical` que centralizará verticalmente todos os demais componentes que estiverem em seu interior. A propriedade `NumbersOnly` do componente `TextBox`, quando marcada, ativará apenas o teclado numérico do telefone ou dispositivo em que o aplicativo estiver instalado, pois o teclado alfanumérico torna-se desnecessário nesse momento, já que desejamos digitar apenas números. Alteramos também a propriedade `TextAlignment` para exibir as informações alinhadas ao lado direito do componente `TextBox`. A figura seguinte exibe a tela

com o espaço para a digitação do valor do etanol.

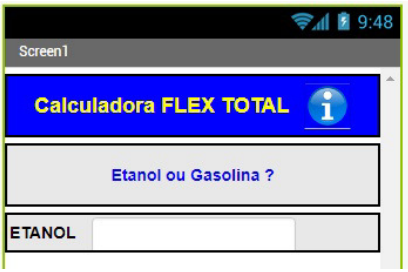

Figura 4.4: Digitação do valor do Etanol

Seguindo essa mesma ideia, realize a inserção e configuração dos componentes para a digitação do preço da **Gasolina**. Veja na tabela os componentes que deverão ser inseridos e as propriedades com os valores para alterar:

Componente	Pallete	Propriedade	Alterar valor
HorizontalArrangement	Layout	Width	Fill Parent
		AlignVertical	Center
Label	User Interface	FontBold	Marcar
		FontSize	16
		Text	GASOLINA
TextBox	User Interface	Hint	Preço da Gasolina
		NumbersOnly	Marcar
		TextAlignment	Right
		Renomear	Txt_Gasolina

A figura a seguir exibe a tela do app após inserirmos os componentes descritos anteriormente.

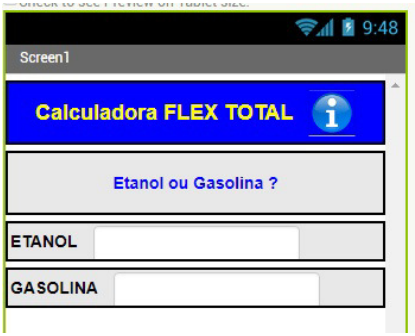

Figura 4.5: Tela com os botões de controle

Necessitamos agora inserir mais um `HorizontalArrangement` abaixo do espaço reservado para a digitação do preço da gasolina, pois ele vai conter dois botões de controle em seu interior. O primeiro realizará o cálculo para informar qual combustível é mais vantajoso para se abastecer, e o segundo realizará a limpeza das informações.

Entre eles, vamos inserir uma `Label` que não terá informação nenhuma, com a finalidade de inserir um espaçamento estético entre os botões de verificar e limpar.

A tabela a seguir exibe a ordem em que deveremos inserir os componentes descritos anteriormente e as propriedades que precisamos alterar juntamente com os novos valores.

Componente	Pallete	Propriedade	Alterar valor
HorizontalArrangement	Layout	AlignVertical	Center
		AlignHorizontal	Center
		Height	100 pixels
		Width	Fill Parent

Componente	Pallete	Propriedade	Alterar valor
Button	User Interface	BackgroundColor	Green
		FontSize	16
		FontBold	Marcar
		Shape	Rounded
		Text	VERIFICAR
		TextAlignment	Center
		TextColor	Blue
		Renomear	Btn_Verifica
Label	User Interface	Text	Inserir 5 espaços
Button	User Interface	BackgroundColor	Green
		FontSize	16
		FontBold	Marcar
		Shape	Rounded
		Text	LIMPAR
		TextAlignment	Center
		TextColor	Blue
		Renomear	Btn_Limpar

A única propriedade que ainda não utilizamos é a `Shape`, que está presente na configuração dos componentes `Buttons`. A seleção da opção `Rounded` é responsável por exibir um contorno arredondado do botão, trocando o modelo padrão retangular.

Também é preciso destacar que deixamos a propriedade `Text` da `Label` com cinco espaços (pressione 5 vezes a barra de espaço de seu teclado). Utilizamos este recurso com a finalidade de inserir um espaçamento estético entre os botões de verificar e limpar.

A figura a seguir exibe a tela após inserirmos os botões e configurarmos as suas propriedades.

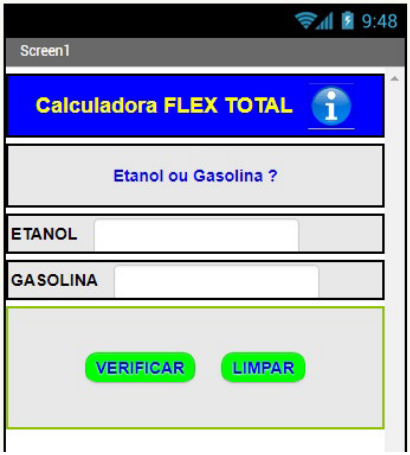

Figura 4.6: Tela com os botões de controle

Quando o usuário pressionar o botão `Btn_Verifica`, precisaremos de uma `Label` para exibir o resultado da conta que será demonstrado mais adiante. Teremos um `HorizontalArrangement` e duas `Labels` em seu interior: uma para exibir a palavra **Resultado** e outra para exibir o resultado do cálculo que efetuaremos mais adiante, neste capítulo.

A tabela seguinte indica a ordem em que os componentes deverão ser inseridos e as suas propriedades com os novos valores para alteração.

Componente	Pallete	Propriedade	Alterar valor
HorizontalArrangement	Layout	AlignVertical	Center
		AlignHorizontal	Center
		Height	50 pixels

		Width	Fill Parent
Label	User Interface	FontSize	20
		Text	RESULTADO
Label	User Interface	FontSize	20
		Text	0.00

Note que não há nenhuma nova propriedade, e isso permitirá ao leitor realizar esta tarefa como um pequeno teste de seu aprendizado. A figura a seguir exibe a tela após inserirmos a `Label` de resultado.

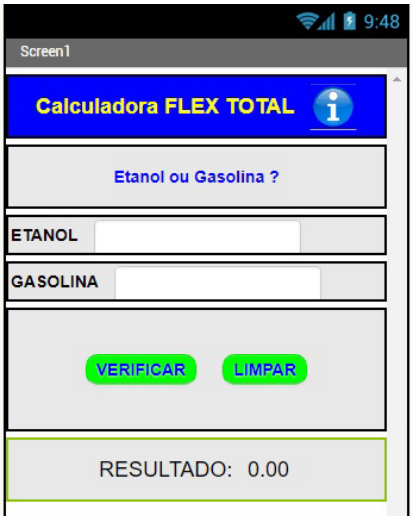

Figura 4.7: Tela com Labels de resultado

Agora, necessitamos informar qual combustível será mais vantajoso após realizarmos os cálculos. Para tanto, vamos inserir novamente mais duas `Labels` : a primeira para exibir a mensagem **Abasteça com** e a segunda para indicar uma das opções, **Etanol** ou

Gasolina. Porém, para demonstrar um novo componente, em vez do `HorizontalArrangement`, veremos o componente `VerticalArrangement`. Ele tem a função de organizar dois ou mais componentes verticalmente em sua tela.

Insira o `VerticalArrangement` abaixo do último `HorizontalArrangement` e, dentro dele, insira as duas `Labels`. A tabela a seguir indica as propriedades com os novos valores para alteração.

Componente	Pallete	Propriedade	Alterar valor
VerticalArrangement	Layout	AlignVertical	Center
		AlignHorizontal	Center
		Height	Fill Parent
		Width	Fill Parent
Label	User Interface	FontSize	20
		Text	ABASTEÇA COM
Label	User Interface	FontBold	Marcar
		FontSize	50
		TextColor	Red
		Text	--
		Renomear	Lbl_Resultado

Figura 4.8: Tela principal finalizada

Para o aplicativo emitir uma mensagem ao usuário, no caso dele se esquecer de digitar algum valor, vamos acrescentar um componente que realiza essa tarefa, o `Notifier`. Você encontra-o na guia `User Interface` e não precisa alterar nenhuma propriedade. Esse componente é chamado de componente não visível, pois ele não será exibido na sua `Screen1` durante o desenvolvimento do layout, ficando localizado na seção de `Non-visible components`.

Antes de programar os blocos — que são representações gráficas de comandos que executam determinadas tarefas no aplicativo —, veja a seguir a fórmula que usaremos para calcular qual combustível é mais vantajoso para abastecer:

Fórmula

Resultado = Etanol / Gasolina

Se o resultado obtido for menor ou igual a 0.70, isso indicará que abastecer com etanol é mais vantajoso; caso contrário, o app deverá mostrar que a gasolina será mais indicada economicamente.

4.2 INSERINDO OS BLOCOS DE CONTROLE

Agora vamos para a área de programação. Para isso, basta clicar no botão `Blocks` no canto superior direito de sua tela:

Figura 4.9: Botão para acionar a área de blocos

Precisamos definir três variáveis numéricas para receber e armazenar os valores digitados do preço do etanol e da gasolina, além do resultado do cálculo obtido através da fórmula apresentada.

Como vimos no capítulo anterior, acesse a área `Blocks` e, na guia `Built-in`, selecione três `initialize global name to` na opção `Variables`, conforme demonstra a figura a seguir. Esses blocos têm por finalidade declarar as variáveis que usaremos.

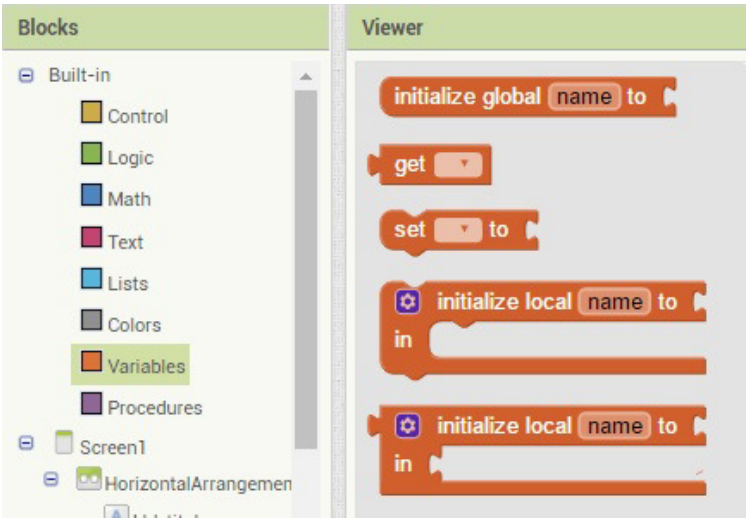

Figura 4.10: Inserindo variáveis

Vamos trocar os nomes das variáveis criadas em cada um de nossos blocos. Clique onde você lê a palavra `name` e altere para **Etanol**, e faça esse procedimento para os outros dois blocos, digitando os nomes **Gasolina** e **Resultado**. Veja os três blocos das variáveis sem os nomes alterados na próxima figura.

Figura 4.11: Blocos das variáveis

Encaixe o valor zero (0) em cada uma das variáveis criadas, para indicar que elas serão numéricas:

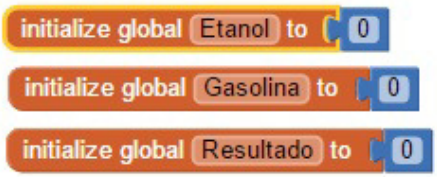

Figura 4.12: Variáveis do app

Vamos agora trabalhar com o botão que realiza os cálculos e a verificação de qual combustível é mais vantajoso. Precisamos inserir um bloco que represente o clique no `Btn_Verifica`, para que os cálculos sejam executados. Devemos primeiramente selecionar na guia de `Blocks` o componente `Btn_Verifica`.

Note que, ao clicar sobre o `Btn_Verifica`, um leque de ações que esse botão pode executar se abrirá ao lado direito. Selecione então o bloco `when Btn_Verifica.Click`, conforme demonstra a figura a seguir. Esse bloco de ação — ou como chamamos na programação, esse **evento** — nos indica que, quando o botão de verificar for clicado, serão executados os demais blocos que estão em seu interior.

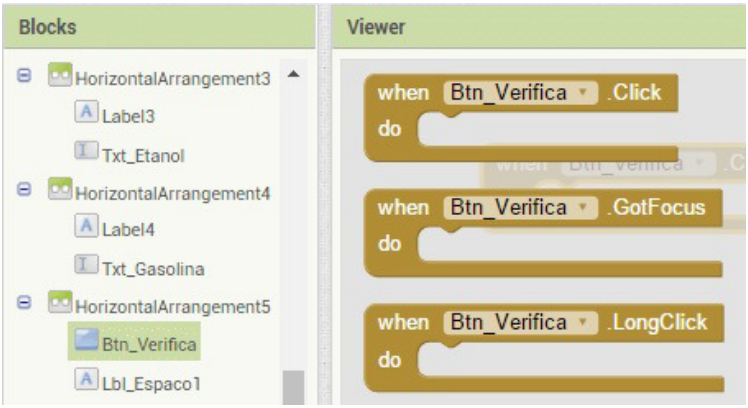

Figura 4.13: Selecionando o bloco Btn_Verifica.Click

Para que o aplicativo identifique o clique do botão `Btn_Verifica`, devemos selecioná-lo e arrastá-lo para a área de programação, ou seja, a área `Viewer` do App Inventor, como demonstrado na figura:

Figura 4.14: Bloco Btn_Verifica

4.3 TOMANDO A DECISÃO

Para o App Inventor poder decidir qual combustível é mais vantajoso, precisamos de um bloco de controle que faça essa verificação. Esse processo é realizado pelo bloco `if`. Junto a este bloco, sempre devemos anexar uma ou mais comparações. No caso de a decisão ser verdadeira, será executada a opção ou opções que vierem encaixadas na frente do comando `then`. No caso de a condição ser falsa, nenhum bloco será executado e a programação continuará apenas com os blocos que estiverem abaixo do bloco `if`.

O bloco `if`, ou estrutura de decisão, possui duas maneiras de trabalhar: o método simples, que é o que acabamos de demonstrar e trata apenas de realizar as tarefas quando a verificação for verdadeira; e o método composto que, além de tratar a condição verdadeira, também executa alguma ação se a comparação do `if` for falsa. Para isso, devemos inserir uma configuração no bloco `if`, ou seja, incluir a opção `else`.

Dentro do `Btn_Verifica`, arraste um bloco de controle `if`.

Para isso, selecione na guia `Built-in` a opção `Control` que, ao lado direito, aparecerá todos os controles disponíveis, então selecione e arraste a opção de decisão `if` para dentro do bloco do `Btn_Verifica`. Veja na figura a seguir como inserir o bloco `if`.

Figura 4.15: Bloco if

Esse bloco terá a finalidade de verificar se o usuário digitou os valores do etanol e da gasolina. Caso o usuário não digitar algum dos dados, o aplicativo emitirá uma informação para que ele digite todos os valores; caso contrário, os cálculos serão realizados. Para isso, precisamos configurar o bloco `if` para aceitar a opção contrária (`else`).

Veja com inserir o `else` em um bloco `if`:

1. Primeiro, clique no ícone de cor azul ao lado esquerdo do comando `if`:

Figura 4.16: Configurando o else

2. Clique no comando `else`, segure e arraste-o para o lado esquerdo dentro do `if`. A figura a seguir exibe o comando `else`, após clicar no ícone de configuração do bloco `if`.

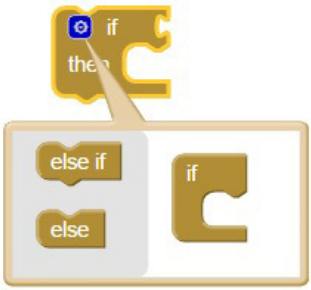

Figura 4.17: Inserindo o bloco else

3. Veja como deverá ficar seu bloco `if/else` finalizado:

Figura 4.18: Bloco if/else finalizado

O bloco `Btn_Verifica` deverá ficar com o `if` inserido desta forma:

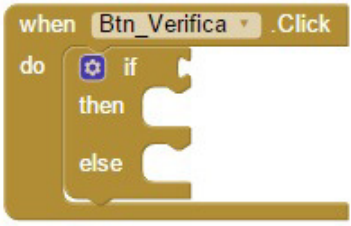

Figura 4.19: Btn_Verifica com o bloco if

Com esse bloco de decisão já posicionado na área `Viewer`, precisamos de um bloco para decidir se a primeira **ou** a segunda opção está sem receber um valor. Para isso, vamos inserir um bloco `or`, da guia `Logic` da seção `Built-in`, e encaixá-lo no bloco `if`. Veja o resultado na figura a seguir:

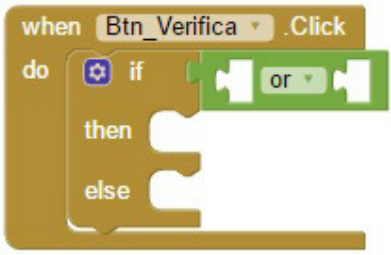

Figura 4.20: Bloco lógico or encaixado no if

Já temos o bloco `if` que realiza a decisão, porém falta ainda inserir seus critérios. Como queremos verificar se não foi digitado algum valor nos campos do etanol ou da gasolina, precisamos inserir um bloco `is empty` da guia `Text`, ou seja, verificar se os campos dos valores estão vazios. Posicione-o para dentro do bloco lógico `or`, conforme vemos a seguir.

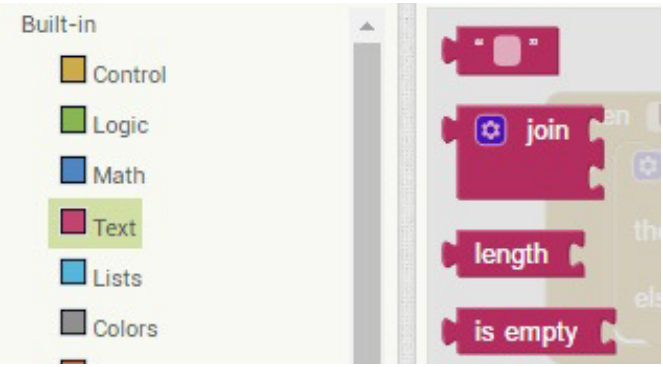

Figura 4.21: Selecionando o bloco is empty

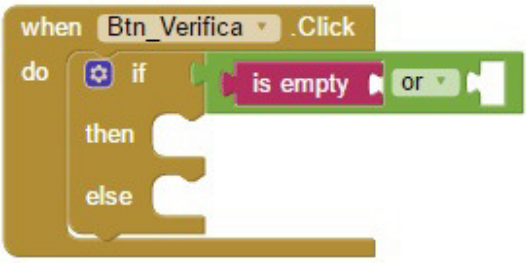

Figura 4.22: Bloco is empty encaixado no if

Agora que já inserimos o bloco `is empty`, devemos verificar qual componente não poderá ficar vazio. Então selecione o componente `Txt_Etanol.Text` e encaixe-o logo após o bloco `is empty`. Com a junção desses blocos, conseguimos verificar se não foi digitado um valor para o campo etanol. Não podemos esquecer de realizar a mesma verificação para o valor da gasolina, para isso, repita o procedimento inserindo o bloco do componente `Txt_Gasolina.Text` após o `is empty`. A figura a seguir demonstra a linha do `if` finalizada.

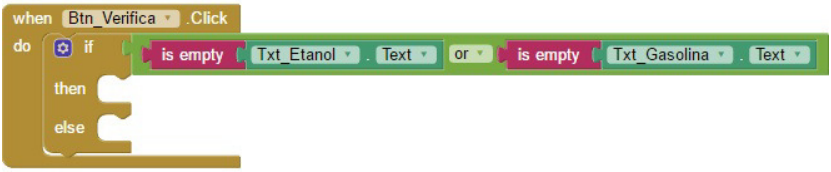

Figura 4.23: Verificando se os valores não foram digitados

Após essa verificação, e se ela for verdadeira, devemos exibir uma notificação ao usuário para que digite as informações necessárias. Existe um componente que realiza essa tarefa, o `notifier` . Note que já inserimos esse componente no design do projeto. Então, vamos adicionar um bloco `call notifier.ShowAlert` para encaixar na frente da opção `then` do `if` , pois este é o local que devemos programar quando a decisão testada por esse bloco for considerada verdadeira.

Na sequência do `Notifier1` , insira um bloco de texto e digite a informação que desejamos exibir para o usuário: `Digite todos os valores!` .

Figura 4.24: Exibindo uma notificação ao usuário

Caso o usuário tenha digitado os dois valores, as variáveis etanol e gasolina deverão recebê-los. Para transferir os valores digitados em tela para as variáveis, devemos realizar uma atribuição de valores, isto é, fazer cada uma delas receber o seu

valor específico.

Então, devemos primeiramente inserir um bloco para receber o valor. Para isso, posicione o ponteiro do mouse sobre a variável `Etanol` para aparecer a opção `set global Etanol to` , já que a opção `set global` nos indica que a variável receberá um valor. Depois, arraste-o para encaixar na frente da opção `else` do bloco `if` , pois só poderemos receber os valores se não estiverem vazios (`else`). A figura a seguir mostra como exibir as opções da variável `Etanol` .

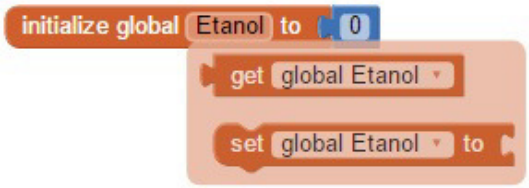

Figura 4.25: Exibindo as opções da variável Etanol

Encaixe-o na variável em que você inseriu um componente `Txt_Etanol.text` , pois é nele que se encontra o valor digitado pelo usuário, e é nesse exato momento que ocorre a atribuição de valor da informação digitada pelo usuário para a variável do aplicativo. Você precisará realizar o mesmo procedimento com a variável `Gasolina` , encaixando nela o bloco `Txt_Gasolina` . Veja na próxima figura o resultado parcial dos blocos.

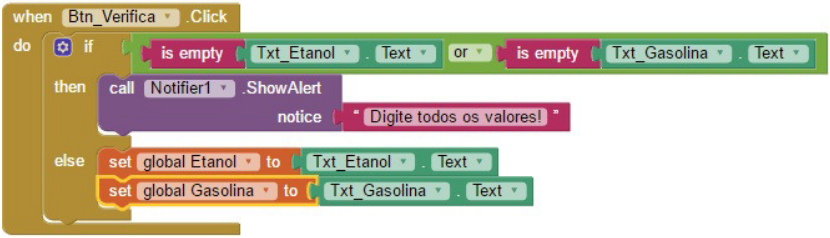

Figura 4.26: Movendo os dados para as variáveis

Já temos as variáveis com os valores digitados. Agora precisamos realizar a conta com a fórmula especificada, em que a variável do resultado receberá a divisão dos valores das variáveis `Etanol` pela `Gasolina`. Para podermos usar o valor que está atribuído a uma variável, devemos utilizar o comando `get global`.

> **Resumindo**
>
> O comando `set` indica que a variável está recebendo um valor. Já o comando `get` indica que queremos utilizar o valor atribuído à variável.

Logo após os blocos de recebimento dos valores digitados, colocaremos um bloco da variável para receber o resultado da conta. Encaixe um bloco da variável `set global resultado to`.

Precisamos de um bloco que realize uma operação matemática de divisão. Note que, na guia `Built-in`, temos uma opção chamada `Math`. Nela encontramos todas as operações matemáticas disponíveis no App Inventor. Localize nessa opção

um bloco de divisão para encaixar na frente da variável resultado. Dentro dos espaços no bloco de divisão, insira um bloco de variável `get global Etanol` e um da `get global Gasolina`. Estes dois últimos você encontra posicionando novamente o ponteiro do mouse sobre a definição da variável, conforme já demonstrado. A próxima figura exibe o bloco que calcula a divisão dos preços.

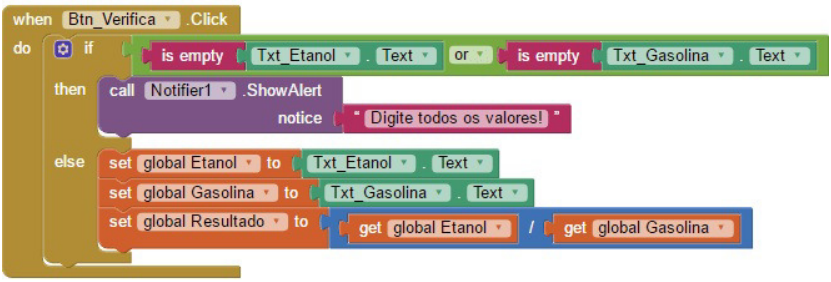

Figura 4.27: Cálculo da divisão do preço do etanol pelo da gasolina

Já calculamos a divisão dos preços do etanol pela gasolina, agora precisamos exibir essa informação no aplicativo para que o usuário possa vê-la. Anteriormente, inserimos um componente `Lbl_Porcento.Text` para isso. Vamos inserir o bloco do componente `set Lbl_Porcento.Text to` da guia `Blocks` para receber o resultado do cálculo.

Como o resultado da divisão pode conter várias casas decimais após a vírgula, seria muito interessante, por questão apenas de estética, que formatássemos a exibição para mostrar apenas dois números depois dela. Para isso, devemos inserir um bloco que vai configurar a exibição do resultado com duas casas decimais.

Selecione na `Built-in` da guia `Math` a opção `format as`

`decimal number` , responsável pela formatação que queremos. Encaixe a variável `get global Resultado` para exibição. E na opção `Places` , responsável pela formatação da quantidade de números que teremos após a vírgula, arraste um bloco numérico, também da guia `Math` , e informe dentro dele a quantidade de casas decimais desejadas. No nosso caso, duas casas após a vírgula. Veja na figura seguinte o bloco finalizado.

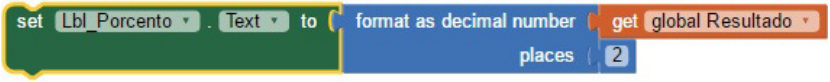

Figura 4.28: Exibição do resultado formatado com duas casas decimais

O ponto mais importante do nosso app, a decisão entre abastecer com etanol ou gasolina, será visto agora. Arraste mais um bloco `if` da guia `Controls` e configure-o para ter também uma opção `else` . Dentro desse bloco, encaixe um de comparação relacional **menor que** (com o símbolo `<`) da guia `Math` . No primeiro espaço, coloque a variável do resultado e, ao lado, encaixe um bloco numérico e insira nele o valor `0.7` .

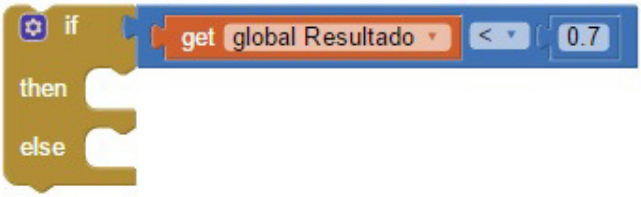

Figura 4.29: Verificando o resultado calculado

Caso a decisão seja verdadeira, devemos exibir no `Lbl_Resultado.Text` a informação **Etanol**; caso contrário, a mesma `Label` exibirá o texto **Gasolina**. Após o `then` , encaixe o bloco do `Lbl_Resultado.Text` e um bloco de texto e, dentro

deste, digite `Etanol` . Caso a decisão do bloco `if` não seja verdadeira, precisamos tratar a segunda opção, o `else` .

Novamente usaremos o bloco `Lbl_Resultado.Text` junto com um bloco de texto com o conteúdo `Gasolina` . Veja na figura a seguir o resultado:

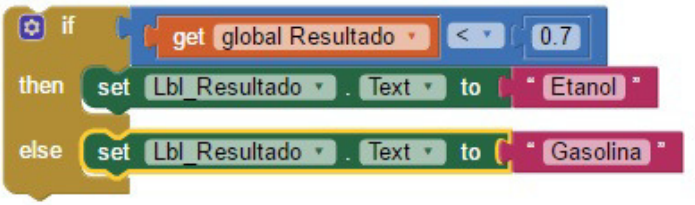

Figura 4.30: Bloco if com a verificação do resultado

Ao término dos blocos, para que o teclado do dispositivo não fique visível após a digitação dos valores, devemos inserir um bloco para escondê-lo. Logo, vamos usar um comando que chamará (`call`) a função de esconder o teclado. A função `call Txt_Gasolina.HideKeyboard` tem essa finalidade, e você encontra-a na guia do componente `Txt_Gasolina` .

Veja na figura a seguir como deverá ficar todo o bloco `Btn_Verifica` .

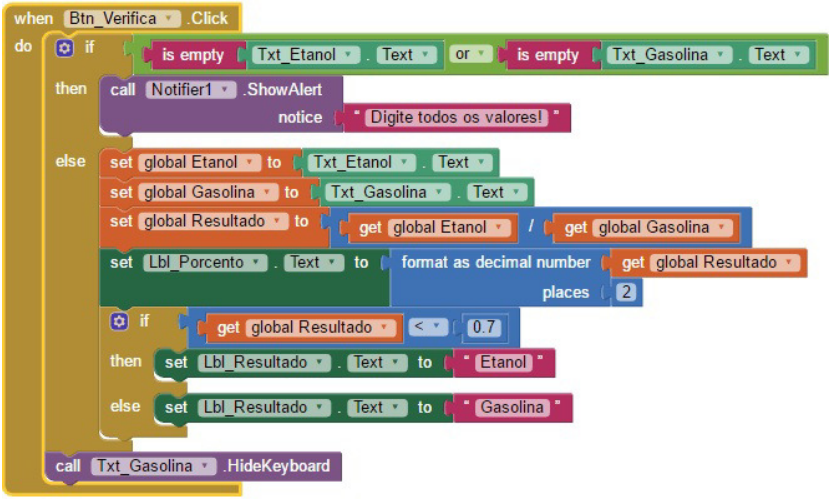

Figura 4.31: Visualização do Btn_Verifica

Vamos supor que o usuário digitou algum valor errado e necessita apagar os dados da tela, ou mesmo que já tenha utilizado uma vez o aplicativo e queira realizar outros cálculos. É para essas ocasiões que preparamos o botão `Btn_Limpar`, que terá a função de apagar todos os dados visíveis nas `TextBoxs` usadas e deixar as `Labels` com os dados iniciais: `Lbl_Resultado` que receberá o valor `--`, e `Lbl_Porcento` que receberá o valor `0.00`.

Para executarmos o que foi descrito, insira o bloco `Btn_Limpar.Click` e, logo a seguir, coloque os blocos `set Lbl_Resultado to`, `set Lbl_Porcento to`, `set Txt_Etanol to` e `set Txt_Gasolina to`. Atribua um bloco de texto vazio para cada componente, depois acrescente os caracteres `--` para o bloco `Lbl_Resultado`, e para o `Lbl_Porcento`, os caracteres `0.00`.

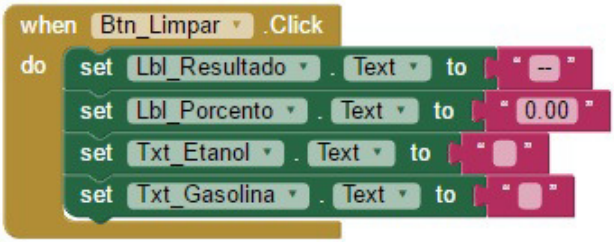

Figura 4.32: Bloco Limpar

Além de mover os valores demonstrados anteriormente, precisamos também apagar os valores contidos nas variáveis para um futuro uso.

Então, arraste as três variáveis `set global etanol to` , `set global gasolina to` e `set global resultado to` e encaixe em cada uma um bloco **numérico** da guia `Math` com o valor zero. Confira na figura a seguir o resultado final do `Btn_Limpar` .

Figura 4.33: Btn_Limpar

4.4 TELA DE INFORMAÇÃO

Precisamos de um botão para exibir uma segunda tela do aplicativo que informará ao usuário como é feito o cálculo. Vamos trabalhar com o botão `Btn_Informacoes` para realizar essa tarefa. Primeiramente, necessitamos inserir uma nova tela. Clique no botão `Add Screen` na parte superior do App Inventor.

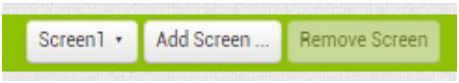

Figura 4.34: Inserindo uma nova Screen

Após o clique, será exibida a janela da figura seguinte, indicando que uma nova `Screen` (tela) será criada. Deixe o nome `Screen2` como padrão sugerido pelo App Inventor e clique no botão `OK`.

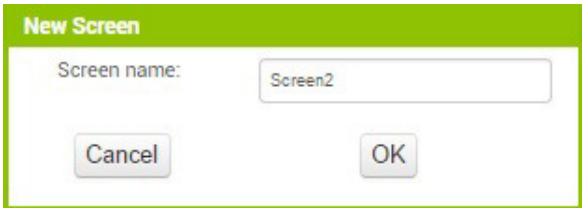

Figura 4.35: Criando uma nova Screen

Será apresentada uma nova tela vazia para nosso aplicativo. Vamos configurá-la para que apresente a visualização conforme a figura a seguir nos apresenta.

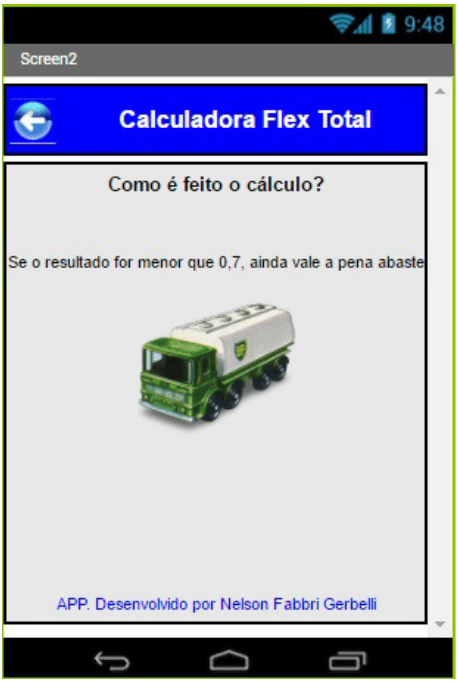

Figura 4.36: Tela de informações do app

Perceba que quase todos os componentes que vamos usar na `Screen2` já foram vistos e comentados anteriormente. Caso persista alguma dúvida, não evite em retomar a leitura.

Como construímos uma barra de título na `Screen1`, também vamos repetir o mesmo procedimento para a `Screen2`. Então, na propriedade `TitleVisible`, clique para desmarcar a opção padrão. Construiremos a barra de título com um botão para retornar à tela principal e uma `Label` para exibir propriamente o título do app.

A tabela a seguir apresenta os objetos que devemos inserir e as propriedades a serem alteradas. Vale lembrar que primeiramente

deverá ser inserido o componente `HorizontalArrangement` e, depois das configurações das suas propriedades, preenchê-lo com os componentes `Button` e `Label`.

Componente	Pallete	Propriedade	Alterar valor para
HorizontalArrangement	Layout	AlignVertical	Fill Parent
		BackgroundColor	Blue
		Height	50 pixels
		Width	Fill Parent
Button	User Interface	Height	35 pixels
		Width	35 pixels
		Image	Voltar.png
		Text	Apagar
		Renomear	Btn_Voltar
Label	User Interface	FontBold	Marcar
		FontSize	20
		Text	Calculadora Flex Total
		TextColor	White
		TextAlignment	Center
		Width	Fill Parent

Veja o resultado da barra de título da `Screen2`:

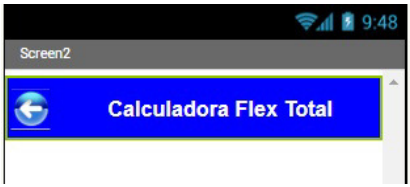

Figura 4.37: Barra de título

Vamos agora preparar a parte central da tela de informações. Abaixo da barra de título, coloque um componente `VerticalArrangement` da guia `Layout`. Como já falamos anteriormente, ele ajusta verticalmente os componentes que ficam em seu interior. Precisamos ajustar o seu comprimento através da propriedade `Width` para `Fill Parent`, pois assim o componente ocupará todo o espaço horizontal do dispositivo. Também queremos deixar centralizados todos os componentes que ainda vamos inserir nele, então, altere as propriedades `AlignVertical` e `AlignHorizontal` para a opção `Center`.

Agora precisamos pôr duas `Labels` no interior da `VerticalArrangement`. Insira uma abaixo da outra, e realize as alterações sugeridas na tabela.

Componente	Pallete	Propriedade	Alterar valor para
Label	User Interface	FontBold	Marcar
		FontSize	16
		Text	Como é feito o cálculo?
		TextAlignment	Center
		Height	50 pixels
		Width	Fill Parent

Componente	Pallete	Propriedade	Alterar valor para
Label	User Interface	FontBold	Marcar
		Text	Divida o valor do litro do etanol pelo da gasolina. Se o resultado for menor que 0.7, ainda vale a pena abastecer com etanol. Se for maior, opte pela gasolina.

Note na imagem a seguir o resultado da configuração das duas `Labels` inseridas, e observe que o texto explicativo de como é feito o cálculo não está sendo exibido por inteiro. Mas não se preocupe, pois, quando você estiver testando seu app, as informações aparecerão por inteiro em seu dispositivo.

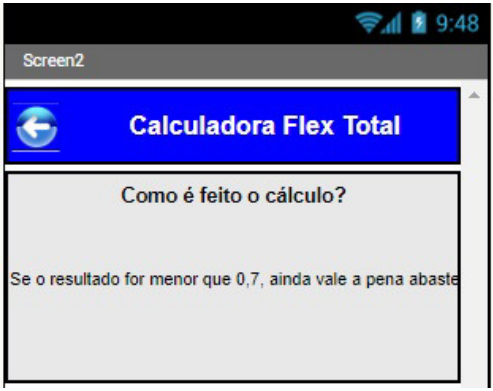

Figura 4.38: Labels configuradas

Para ilustrar a tela de informações, vamos inserir uma imagem com um caminhão tanque. Localize na guia `User Interface` um componente `Image`, responsável pela exibição de imagens, e coloque-o abaixo da `Label` que contém o texto da explicação dos

cálculos. Para inserir uma imagem, vá até a propriedade `Picture` e realize o upload do arquivo `Leyland-Petrol-Tanker-icon.png`.

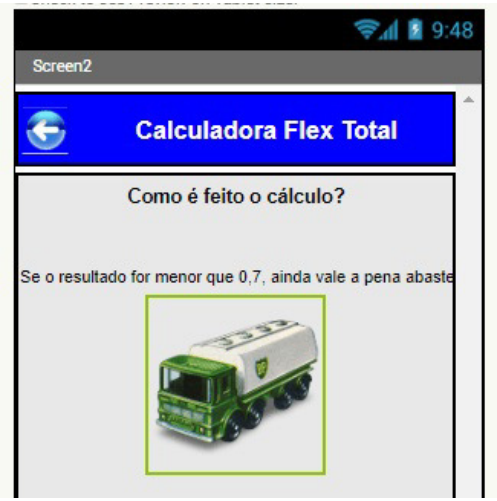

Figura 4.39: Imagem exibida

Para terminarmos a criação da tela de informações, só falta inserir duas `Labels`: a primeira servirá apenas para dar um espaçamento entre a imagem e a `Label` que vai conter a informação do desenvolvedor do app. Veja na tabela a seguir as propriedades que devem ser alteradas e seus novos valores.

Componente	Pallete	Propriedade	Alterar valor para
Label	User Interface	Height	90 pixels
		Text	Apagar
Label	User Interface	Width	Fill Parent
		Text	App desenvolvido por Nelson

			Fabbri Gerbelli
		TextColor	Blue

Com o design pronto, precisamos agora realizar a programação do botão voltar, que tem como finalidade fechar a tela de informações e retornar para a tela principal do aplicativo.

Na área dos blocos, selecione o componente `When Btn_Voltar.Click` . Agora precisamos de um comando para fechar a tela de informações. Para isso, acesse a opção `Control` na guia `Built-in` e selecione um `close screen` , posicionando-o dentro do `When Btn_Voltar.Click` . Veja na figura a seguir o resultado do `Btn_Voltar` .

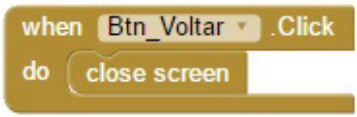

Figura 4.40: Btn_Voltar

Ainda na `Screen1` , é preciso realizar a chamada para a exibição da segunda `Screen` . Clique no botão `Screen2` no topo do App Inventor e selecione a `Screen1` , para retornar ao desenvolvimento da tela principal.

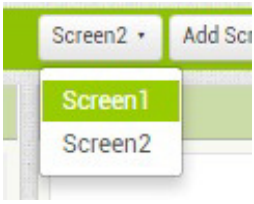

Figura 4.41: Acessando a Screen1

Selecione o componente `Btn_informações.Click` e arraste-o para a área de desenvolvimento. Agora na guia `Built-in`, na opção `Control` selecione um bloco `open another screen screenName`, que realiza a abertura e exibição da outra tela. Encaixe um bloco de texto, e digite internamente o nome da sua segunda tela do aplicativo — no nosso caso, digite apenas `Screen2`. Atenção para as letras maiúsculas e minúsculas; é preciso digitar igual ao nome que foi dado durante a criação da segunda tela.

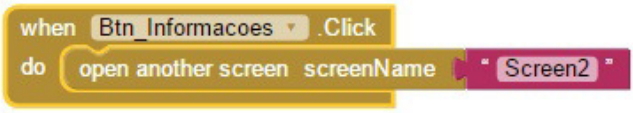

Figura 4.42: Btn_Informacoes para exibir a Screen2

4.5 TESTANDO O APLICATIVO

Para finalizarmos este capítulo, sugerimos que você emule seu aplicativo. Utilize uma das maneiras para emular, demonstradas no capítulo anterior.

Após a visualização da tela no emulador, digite os valores para o preço do litro do etanol e da gasolina e, ao final, clique no botão de calcular para ver o resultado de qual combustível é mais vantajoso. Não se esqueça de clicar no botão de informação para acessar a `Screen2` do app. As figuras a seguir exibem as telas do aplicativo em funcionamento.

Figura 4.43: Aplicativo em funcionamento

Figura 4.44: Exibindo a tela de informações

4.6 RESUMINDO

Neste capítulo, aprendemos a criar uma nova tela, utilizar as variáveis, realizar contas e a configurar a exibição de casas decimais. Vimos também decisões com o bloco `if/else` , como

limpar os dados já visualizados, e criamos um app que você poderá utilizar no dia a dia para decidir qual é o combustível que mais compensa quando for abastecer seu veículo flex.

No próximo capítulo, o leitor vai produzir um aplicativo que realizará a tradução de uma palavra, ou de um texto digitado ou falado, para um dos idiomas disponíveis. Além disso, será possível compartilhar a tradução por aplicativo externo, ou se desejar, ouvir o que foi traduzido através da leitura realizada pelo aplicativo.