CAPÍTULO 5

TRADUTOR ONLINE

Neste capítulo, vamos desenvolver um aplicativo que realiza a tradução de textos do português para outros idiomas selecionados por você. Habilitaremos o reconhecimento de voz de seu dispositivo para informar o texto a ser traduzido pelo app. Além da tradução, você poderá compartilhar o resultado com seus contatos, ou simplesmente ouvir o texto que foi traduzido.

> Vá ao link de download dos projetos do livro (http://nelfabbri.com/appinventor/appinventor.zip) e baixe os arquivos que serão utilizados nesse aplicativo.

O app que desenvolveremos fará a tradução de uma palavra ou frase do português para o inglês, italiano, alemão ou espanhol, utilizando o componente de tradução `Yandex Translate`. O usuário poderá digitar as palavras a serem traduzidas na caixa de texto ou simplesmente usar o reconhecimento de voz que transformará o que foi falado em texto escrito. Para isso, veremos o componente `SpeechRecognizer`.

O aplicativo também possibilitará a leitura do texto que foi

E-book gerado especialmente para Carla Fabiane Calixto - carlacalixto16@gmail.com

traduzido através do componente `TextToSpeech`, e o compartilhamento do texto traduzido pelo aplicativo WhatsApp, por meio do componente `ActivityStarter`. Este transmitirá informações do nosso app para outro que está instalado em nosso dispositivo móvel. O aplicativo consumirá dados de internet, pois a tradução é realizada através de uma API externa.

> Uma API (*Application Programming Interface*) é um conjunto de padrões de programação disponíveis para que outros aplicativos utilize seus resultados.

5.1 APP TRADUTOR

Para desenvolver nosso tradutor, vamos iniciar um novo projeto no App Inventor. Clique na opção `Start New Project` e dê o nome de **Tradutor**.

Para o desenvolvimento do design do app, necessitaremos de cinco botões. Veja a seguir as funções de cada um:

1. O primeiro botão realizará o reconhecimento da voz do usuário;
2. O segundo vai efetivar a tradução;
3. O terceiro lerá o texto que foi traduzido;
4. O quarto botão limpará os textos da tela para realizar uma nova tradução;

E-book gerado especialmente para Carla Fabiane Calixto - carlacalixto16@gmail.com

5. O último botão vai possibilitar o compartilhamento da tradução com algum de seus contatos do WhatsApp.

Vamos conhecer o componente `ListPicker`. Ele é responsável por listar todos os idiomas que cadastraremos para realizar a tradução. Usaremos também `Labels` para exibir as mensagens e duas `TextBoxs`, que servirão para a digitação do texto a ser traduzido e para a exibição da tradução.

A tela do aplicativo é demonstrada na figura a seguir:

Figura 5.1: Tela do aplicativo tradutor online

Para começar, vamos realizar algumas modificações com a tela de fundo do nosso aplicativo. Na área `Components` do ambiente

E-book gerado especialmente para Carla Fabiane Calixto - carlacalixto16@gmail.com

de desenvolvimento do App Inventor, selecione a `Screen1` para habilitar as suas propriedades.

Figura 5.2: Habilitando as propriedades da Screen1

A tabela a seguir demonstra os componentes que deveremos inserir, o local onde os encontramos, as propriedades que deveremos alterar e os novos valores de cada uma.

Componente	Pallete	Propriedade	Alterar valor para
Screen1	- -	APPName	Tradutor
		BackgroundColor	Gray
		Icon	FALAR.png
		ScreenOrientation	Portrait
		TitleVisible	Desmarcar, pois criaremos a nossa barra de título personalizada.

A propriedade `APPName` indicará o nome do aplicativo que será exibido quando o leitor o instalar em seu dispositivo. A `BackgroundColor` altera a cor do fundo da tela, que aqui definimos como `Gray` (cinza). Na propriedade `Icon`, a imagem que utilizamos deverá ser exibida como ícone de seu aplicativo após a instalação em um dispositivo móvel. Faça o upload da imagem `FALAR.png`, conforme vimos em capítulos anteriores.

Definimos a propriedade `ScreenOrientation` para `Portrait`, para que a tela de seu aplicativo permaneça sempre na posição de retrato, ou seja, fique sempre em pé, independente de

E-book gerado especialmente para Carla Fabiane Calixto - carlacalixto16@gmail.com

deitarmos ou não nosso dispositivo. Já a propriedade `TitleVisible`, deve ser desmarcada para que o título padrão do App Inventor não apareça, pois desejamos criar uma área de título personalizada. Sua `Screen1` ficará conforme a figura:

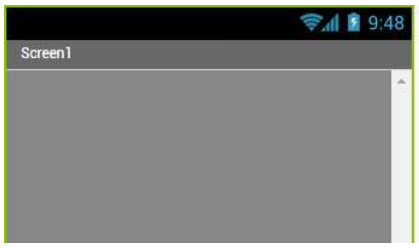

Figura 5.3: Screen1

Vamos agora criar o título de nosso aplicativo, juntamente com um botão para a escolha da língua para a qual o seu texto deverá ser traduzido. Como desejamos ter dois componentes na mesma linha, torna-se necessário usar um componente que os agrupe, o `HorizontalArrangement`, e você poderá encontrá-lo na `Pallete` dos `Layouts`. Localize-o e o insira em sua `Screen1`. Note que ele já se posiciona na parte superior de sua tela.

A tabela a seguir mostra as propriedades que deverão ser alteradas e seus novos valores.

Componente	Pallete	Propriedade	Alterar valor para
HorizontalArrangement	Layout	AlignHorizontal	Right
		AlignVertical	Center
		BackGroundColor	Dark Gray
		Height	35 pixels
		Width	Fill Parent

E-book gerado especialmente para Carla Fabiane Calixto - carlacalixto16@gmail.com

A propriedade `AlignHorizontal` realiza o alinhamento dos componentes que ficarão em seu interior. Aqui, selecionamos o alinhamento para que todos os componentes fiquem alinhados à direita, por isso foi marcada a opção `Right` . Mas para um alinhamento referente ao seu espaço vertical dentro do `HorizontalArrangement` , a propriedade `AlignVertical` foi indicada como `Center` .

A propriedade `BackGroundColor` indica a cor de fundo da barra de título que está sendo desenvolvida, logo deverá ficar com uma cor diferente da usada no fundo. Escolhemos a cor `Dark Gray` (cinza escuro). Precisamos também alterar os tamanhos da altura e da largura que o `HorizontalArrangement` ocupará dentro da `Screen1` : a propriedade `Height` terá o valor de `35 pixels` e a `Width` terá a opção `Fill Parent` marcada, já que ocupará todo o espaço horizontal do seu dispositivo.

Dentro do componente `HorizontalArrangement1` , vamos inserir uma `Label` para exibir o título do aplicativo. Veja quais propriedades deveremos alterar:

Componente	Pallete	Propriedade	Alterar valor para
Label	User Interface	FontBold	Ativar
		FontSize	18
		Width	Fill Parent
		Text	Traduz Tudo
		TextAlign	Center
		TextColor	White

A propriedade `FontBold` habilitará o efeito de **negrito** e, em

E-book gerado especialmente para Carla Fabiane Calixto - carlacalixto16@gmail.com

`FontSize` , alteramos o **tamanho da fonte** para 18. Em `Width` , desejamos que o título ocupe todo o espaço interno do `HorizontalArrangement` . Já a propriedade `Text` mudará o nome de exibição, de `Text for Label1` para o texto sugerido: **Traduz Tudo**.

Como gostaríamos que o título ficasse centralizado na linha, alteramos a propriedade `TextAlign` para `Center` . A última formatação para a `Label` do título foi elegermos a cor do texto (`TextColor`), branco (`White`).

Ainda dentro da área de título no `HorizontalArrangement1` , devemos inserir o componente `ListPicker` , que exibirá os idiomas à disposição do usuário para a tradução. Insira o componente ao lado direito da `Label` que indica o título do aplicativo. A tabela seguinte mostra as propriedades e os valores do `ListPicker` :

Componente	Pallete	Propriedade	Alterar valor para
ListPicker	User Interface	ElementsFromString	Alemão, Espanhol, Inglês, Italiano
		Height	35 pixels
		Width	35 pixels
		Image	configuracao.png
		Text	Apagar todo o texto

A propriedade `ElementsFromString` exibe as possibilidades de idiomas selecionados para tradução. Fica a critério do leitor acrescentar ou excluir opções, mas, neste momento, sugiro manter as escolhas devido à programação dos blocos que veremos adiante.

E-book gerado especialmente para Carla Fabiane Calixto - carlacalixto16@gmail.com

Definimos o tamanho do botão, tanto sua altura (`Height`) como seu comprimento (`Width`), em `35 pixels` . A propriedade `Image` exibe uma imagem de fundo no componente `ListPicker` e, para isso, é preciso realizar o upload da imagem `configuracao.png` , conforme já demonstrado. Como queremos exibir apenas uma imagem no `ListPicker` , devemos deletar o texto de seu interior, apagando todo o conteúdo da propriedade `Text` .

Veja como deverá estar a visualização da barra de título, após as configurações feitas até aqui:

Figura 5.4: Barra de título

Precisamos inserir mais uma `Label` abaixo da área de título para informar ao usuário o que ele deverá fazer.

Componente	Pallete	Propriedade	Alterar valor para
Label	User Interface	Text	INSIRA O TEXTO PARA TRADUZIR
		TextColor	White

A figura a seguir exibe o resultado das alterações realizadas na `Label` .

E-book gerado especialmente para Carla Fabiane Calixto - carlacalixto16@gmail.com

Figura 5.5: Estágio atual da Screen1

Necessitamos de um espaço para a digitação do texto a ser traduzido. O componente utilizado para isso, como já vimos, é a `TextBox`. Insira-o imediatamente abaixo da `Label` com o texto **INSIRA O TEXTO PARA TRADUZIR**. Veja na seguinte tabela as configurações que a `TextBox` deverá sofrer.

Componente	Pallete	Propriedade	Alterar valor para
TextBox	User Interface	Height	80 Pixels
		Width	Fill Parent
		Hint	Digite seu texto
		Multiline	Deixar marcado
		Rename	Texto

Como já demonstramos as propriedades `Height` e `Width`, não repetiremos sua explicação, pois elas realizarão as mesmas funções. Já a propriedade `Hint` teve seu valor alterado para **Digite seu texto**. Essa informação será exibida dentro da `TextBox` quando seu aplicativo for executado e, ao escrever dentro dele, ela sumirá automaticamente, funcionando apenas como um lembrete do que deverá ser digitado em seu interior.

A propriedade `Multiline`, quando selecionada, permite que a frase contida no componente seja exibida em várias linhas.

E-book gerado especialmente para Carla Fabiane Calixto - carlacalixto16@gmail.com

Alteramos o nome da `TextBox` para `Texto`, pois futuramente será mais fácil identificar o componente pelo nome `Texto`. Veja o atual estágio em que a `Screen1` está em relação ao layout desenvolvido:

Figura 5.6: Layout atualizado

Para dar um espaçamento entre a `TextBox` que receberá o texto para ser traduzido e a próxima `Label` que indicará o texto já traduzido, precisamos inserir uma `Label` que terá apenas a função de separar esses dois componentes. Ela não conterá nenhuma informação, é apenas uma questão de ajuste de layout do app.

Insira-a logo abaixo da `TextBox` e realize as modificações em suas propriedades:

Componente	Pallete	Propriedade	Alterar valor para
Label	User Interface	Height	20 pixels
		Text	Apagar o texto

Agora, precisamos inserir a informação de que o texto apresentado logo abaixo é o resultado da tradução realizada. Para tanto, insira mais uma `Label` imediatamente abaixo da que

E-book gerado especialmente para Carla Fabiane Calixto - carlacalixto16@gmail.com

deixamos vazia e altere a propriedade que exibe o texto ao usuário (`Text`) para **TRADUÇÃO**. A propriedade que define a cor do texto (`TextColor`) deverá ser `White` .

Veja na figura o atual estágio da tela:

Figura 5.7: Layout atualizado

Necessitamos de um componente que exibirá a tradução realizada. Vamos utilizar o `TextBox` . Insira-o imediatamente abaixo da `Label` com o texto **Tradução** e realize as alterações em suas propriedades:

Componente	Pallete	Propriedade	Alterar valor para
TextBox	User Interface	BackgroundColor	Light Gray
		Enabled	ativar
		Height	80 Pixels
		Width	Fill Parent
		Hint	Texto Traduzido
		Multiline	Deixar marcado
		Rename	TRADUÇÃO

E-book gerado especialmente para Carla Fabiane Calixto - carlacalixto16@gmail.com

A única propriedade que está na tabela e ainda não foi apresentada é a `Enabled`. Quando ativada, ela desabilita a digitação em seu interior. Utilizaremos esse recurso, pois essa `TextBox` servirá apenas para exibir o resultado da tradução, não sendo necessário digitar nenhum conteúdo em seu interior.

Após as alterações, veja na figura a seguir como estará sua tela:

Figura 5.8: Layout atualizado

Novamente por questões de estética, vamos colocar mais uma `Label`, logo abaixo da `TextBox` do texto traduzido, para usar como um espaço entre os componentes. Veja na tabela a seguir as suas propriedades:

Componente	Pallete	Propriedade	Alterar valor para
Label	User Interface	Height	20 pixels
		Text	Deixar sem texto

E-book gerado especialmente para Carla Fabiane Calixto - carlacalixto16@gmail.com

Abaixo dessa última `Label` de espaçamento, vamos escrever no aplicativo o nome do seu desenvolvedor. Então, insira mais uma `Label` e realize as alterações em suas propriedades.

Componente	Pallete	Propriedade	Alterar valor para
Label	User Interface	FontBold	Selecionar
		Width	Fill Parent
		Text	Desenvolvido pelo Prof. Nelson Fabbri Gerbelli

Veja na figura a seguir como estará sua tela.

Figura 5.9: Layout atualizado com o nome do desenvolvedor

Abaixo do nome do desenvolvedor, prepararemos a área na qual os botões deverão ficar. Primeiramente, vamos inserir mais um componente que realizará sua organização horizontal, o já conhecido `HorizontalArrangement`, e realizar as seguintes

E-book gerado especialmente para Carla Fabiane Calixto - carlacalixto16@gmail.com

alterações:

Componente	Pallete	Propriedade	Alterar valor para
HorizontalArrangement	Layout	AlignVertical	Center
		BackgroundColor	Gray
		Width	Fill Parent

Para finalizarmos a criação da tela do nosso app, devemos inserir no interior do `HorizontalArrangement` cinco `Buttons`, cada um com sua função bem definida. Veja a tabela a seguir com as propriedades que devem ser alteradas e seus novos valores.

Componente	Pallete	Propriedade	Alterar valor para
Button	User Interface	Height	35 pixels
		Width	35 pixels
		Image	Microfone.png
		Text	Apagar o texto
		Rename	btn_reconhece_voz
Button	User Interface	Height	35 pixels
		Width	Fill Parent
		Text	TRADUZIR
		TextAlign	Center
		Rename	btn_Traduzir
Button	User Interface	Height	35 pixels
		Width	Fill Parent
		Text	Falar
		TextAlign	Center
		Rename	btn_Falar

E-book gerado especialmente para Carla Fabiane Calixto - carlacalixto16@gmail.com

Button	User Interface	Height	35 pixels
		Width	Fill Parent
		Text	LIMPAR
		TextAlign	Center
		Rename	btn_Limpar
Button	User Interface	Height	35 pixels
		Width	Fill Parent
		Image	whats.png
		TextAlign	Center
		Rename	btn_whats
		Text	Apagar o texto

Realize o upload das imagens indicadas na propriedade `Image` para a exibição no botão.

> Vale lembrar que as imagens `Microfone.png` e `whats.png`, assim como todas as outras usadas neste livro, estão disponíveis para download conforme indicado no começo do capítulo.

Devemos inserir também cinco componentes não visíveis que nos ajudarão a realizar as atividades propostas. São eles:

Objeto	Pallete	Função
Yandex Translate	Media	Uma API (Interface de Programação de Aplicativos) gratuita da empresa Yandex que está incorporada ao App Inventor, e é responsável pela tradução de mais de 70

E-book gerado especialmente para Carla Fabiane Calixto - carlacalixto16@gmail.com

		idiomas. Saiba mais através do endereço http://api.yandex.com/translate/.
TextToSpeech	Media	Componente que realiza a leitura do texto selecionado, ou seja, faz a vocalização do texto traduzido.
SpeechRecognizer	Media	Um componente de reconhecimento de voz para ouvir o que o usuário está falando e converter sua fala em texto. Utiliza o recurso de reconhecimento de fala do sistema Android.
ActivityStarter	Connectivity	Componente que realiza a abertura de outro aplicativo. No nosso caso, o nosso tradutor abrirá o aplicativo WhatsApp para envio e compartilhamento da tradução realizada.
Notifier	User Interface	Exibe uma mensagem de aviso ao usuário caso ele não possua o aplicativo **WhatsApp** instalado em seu dispositivo.

Veja como ficarão disponibilizados os componentes não visíveis em sua tela:

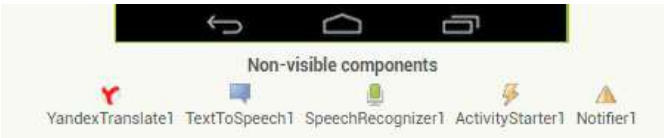

Figura 5.10: Componentes não visíveis

5.2 BLOCOS DO TRADUTOR

Com o design realizado e configurado, vamos clicar sobre o botão `Blocks` para começarmos a programar os blocos. Como teremos quatro possibilidades de idiomas para realizar a tradução, precisamos definir uma variável para receber e armazenar essa informação. Nesse caso, definiremos uma linguagem padrão para a tradução, pois se o usuário não selecionar nenhuma, o app vai traduzir automaticamente do português para o inglês.

E-book gerado especialmente para Carla Fabiane Calixto - carlacalixto16@gmail.com

Inicialize uma variável com o nome `idioma` e atribua um bloco de texto vazio. Dentro, escreva em letras minúsculas `pt-en`, indicando que vamos realizar uma tradução do idioma português para o inglês. Veja na próxima figura como deverá ficar definido o bloco da variável.

> Cada língua possui seu próprio código, que é atribuído por letras ou números como sendo seu identificador. Em nosso aplicativo, vamos usar os seguintes códigos de línguas que indicam as traduções:
>
> - **pt-en** — Português para inglês;
> - **pt-de** — Português para alemão;
> - **pt-es** — Português para espanhol;
> - **pt-it** — Português para italiano.

Figura 5.11: Variável idioma configurada com valor padrão

Veremos a seguir a programação de cada botão de nosso aplicativo.

5.3 RECONHECIMENTO DE VOZ

Teremos um botão que fará o reconhecimento da voz do usuário e depois escreverá o texto na tela. Após clicar no botão `btn_reconhece_voz`, será ativado o reconhecimento de voz de

E-book gerado especialmente para Carla Fabiane Calixto - carlacalixto16@gmail.com

seu celular através do componente `SpeechRecognizer1` e este escreverá o que foi dito por você na `TextBox` .

Vamos programar esse botão. Para isso, na guia `Blocks` , localize o componente `btn_reconhece_voz` . Selecione o bloco `When btn_reconhece_voz.click` e arraste-o para a área `Viewer` .

Para o aplicativo reconhecer a sua fala, precisamos usar o bloco do componente `Call SpeechRecognizer1.GetText` . Localize-o em `Blocks` e encaixe-o no botão `btn_reconhece_voz` , pois somente assim será ativada a função de reconhecimento de voz do seu dispositivo. Veja na figura a seguir como deverá ficar o bloco `btn_reconhece_voz` .

Figura 5.12: Bloco de reconhecimento de voz

Agora que o aplicativo já identificou sua fala, devemos exibir o resultado no formato de texto dentro da `TextBox` que realizará a tradução. Vamos usar o evento que será executado após o recebimento do texto, o `AfterGettingText` . Então, inserimos o bloco `When SpeechRecognizer1.AfterGettingText` da guia de componentes. Veja como fica o bloco que receberá a fala:

E-book gerado especialmente para Carla Fabiane Calixto - carlacalixto16@gmail.com

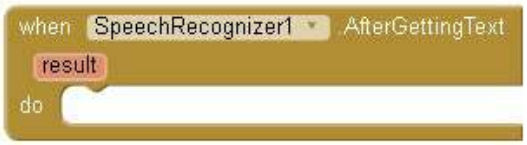

Figura 5.13: Bloco que recebe o texto falado

Para escrever o texto que foi ditado, necessitamos indicar o local onde ele será apresentado. Para a exibição, utilizaremos o componente `TextBox` que renomeamos como **Texto**. Insira um bloco `set Texto.Text to` da área de componentes no interior do `When SpeechRecognizer1.AfterGettingText`, pois é no componente `Texto.Text` que estará o texto a ser traduzido. Veja como está o desenvolvimento até o momento:

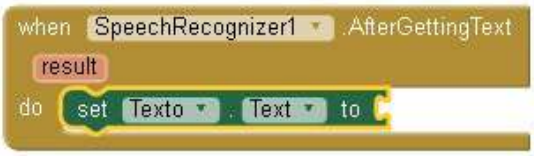

Figura 5.14: Preparando a TextBox para o recebimento do texto

Precisamos agora receber do aplicativo o texto para a exibição. O texto escrito a partir do que foi falado encontra-se na variável `result` do bloco `When SpeechRecognizer1.AfterGettingText`. Encontramos a variável `result` ao posicionarmos o ponteiro do mouse sobre a opção `result` no bloco principal e arrastando a opção `get result` para complementar o bloco `set Texto.Text to`. Veja o processo para encontrar a variável `get result` na figura seguinte.

E-book gerado especialmente para Carla Fabiane Calixto - carlacalixto16@gmail.com

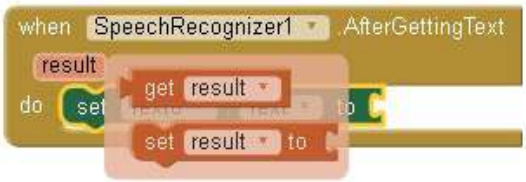

Figura 5.15: Selecionando a opção get result

Figura 5.16: Exibindo o texto falado

No reconhecimento de voz, devido ao barulho externo do seu ambiente, o aplicativo poderá não reconhecer sua fala perfeitamente. Nesse caso, será escrita alguma palavra errada, bastando apenas corrigi-la pelo teclado para não ocorrer erros na execução do aplicativo.

5.4 TRADUZINDO O TEXTO

Temos um botão que realiza a tradução do texto que já está visível na `TextBox` do aplicativo. Quando clicarmos nele para traduzir, vamos chamar a API do YandexTranslate e enviar todas as informações para que seja feita a tradução.

Vamos começar a programação inserindo um bloco do componente `btn_Traduzir.click` e, no seu interior, encaixar o bloco que realiza a chamada do componente `YandexTranslate1`, o `call YandexTranslate1.RequestTranslation`, para

E-book gerado especialmente para Carla Fabiane Calixto - carlacalixto16@gmail.com

requisitar a tradução. Dentro desse bloco, arraste a variável de idioma para informar de qual e para qual idioma o texto deverá ser traduzido, e encaixe-a na opção `languageToTranslateTo` . Vale lembrar que, na criação da variável idioma, nós definimos o valor **pt-en** como padrão.

Para finalizarmos esse bloco, informe o texto a ser traduzido, que se encontra no componente `TEXTO.text` , inserindo-o na opção `textToTranslate` . Note como deverá ficar seu bloco do botão `btn_Traduzir` na figura a seguir.

Figura 5.17: Bloco btn_Traduzir finalizado

Após a execução do bloco `call YandexTranslate1.RequestTranslation` , que enviou os dados para a tradução do texto, precisamos recebê-la e exibi-la no local correto. Para tanto, insira o bloco `When YandexTranslate1.GotTranslation` , que recebe o texto já traduzido. Porém, é preciso indicar o local para a exibição da tradução. Insira um bloco do componente `set TRADUCAO.Text to` para exibir a tradução que virá da API.

Para recebê-la, descanse o ponteiro do mouse sobre a opção `translation` e arraste o bloco `get translation` para completar a exibição:

E-book gerado especialmente para Carla Fabiane Calixto - carlacalixto16@gmail.com

Figura 5.18: Recebendo a tradução e exibindo-a no app

5.5 VOCALIZAÇÃO DA TRADUÇÃO

Temos um botão que poderá ler o texto que já foi traduzido. O componente responsável pela leitura é o `TextToSpeech`. Devemos chamar o `TextToSpeech` e indicar qual texto deverá ser lido.

Vamos inserir o bloco do componente `When btn_Falar.click`, pois a leitura será inicializada somente após o clique nele. Depois, arraste o bloco do componente `call TextToSpeech1.Speak` para iniciá-la.

Precisamos indicar qual texto deverá ser lido e encaixar na opção `message` o componente `TRADUCAO.TEXT`. A figura a seguir demonstra o botão de vocalização completo:

Figura 5.19: Botão que realiza a leitura do texto traduzido

Para realizar uma nova tradução, precisamos apagar os textos

E-book gerado especialmente para Carla Fabiane Calixto - carlacalixto16@gmail.com

exibidos na tela do app. Para isso, basta clicar no `btn_Limpar` que realizará tal tarefa.

Para realizarmos os blocos do botão limpar, é necessário encaixar espaços em branco em cada componente. Ou seja, adicionaremos uma caixa de texto sem preenchimento para as duas `TextBoxs` `set Texto.text to` e `set TRADUCAO.Text to`, conforme demonstrado na próxima imagem.

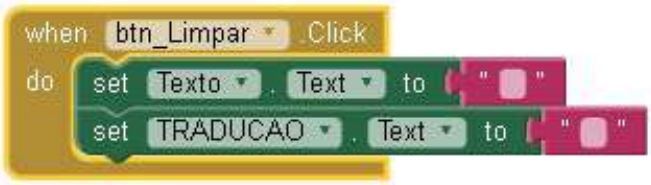

Figura 5.20: Botão que limpa os dados da tela

5.6 SELECIONANDO O IDIOMA PARA A TRADUÇÃO

Como nosso aplicativo poderá traduzir o texto para uma das línguas cadastradas no `ListPicker`, precisamos identificar qual foi a seleção do usuário. Quando o usuário clicar no ícone de configuração (`ListPicker1`), aparecerão as 4 opções já inseridas na tela de design (Alemão, Espanhol, Inglês e Italiano). Quando ele escolher o idioma desejado, o aplicativo deverá identificar qual índice do `ListPicker` foi selecionado.

Conseguimos identificar esse índice pela propriedade `SelectionIndex` do `ListPicker1`, e devemos guardar esse valor na variável `idioma`. A partir deste ponto, temos de realizar uma série de decisões para identificar qual língua foi escolhida. Se

E-book gerado especialmente para Carla Fabiane Calixto - carlacalixto16@gmail.com

for selecionada a primeira opção, teremos o índice 1; se a opção for Espanhol, o índice será o número 2, e assim sucessivamente.

Como o aplicativo também lerá a tradução feita, devemos realizar os ajustes necessários do idioma para a tradução e leitura do texto traduzido, pois se não o alterarmos, a leitura será feita com um sotaque. Por exemplo, para a leitura de um texto em alemão, estaríamos usando um leitor no idioma inglês, e não queremos isso.

Na prática, vamos inserir primeiramente um bloco indicando que, após a seleção de um idioma constante no `When ListPicker1.AfterPicking`, devemos realizar todas as tarefas descritas anteriormente. Em seu interior, vamos inserir a variável `set global idioma to` para receber o número do índice selecionado no `ListPicker1`. Ele é identificado através do bloco `ListPicker1.SelectionIndex`, que deverá ser posicionado na variável.

Figura 5.21: Variável recebendo o idioma selecionado

Agora que a variável já recebeu a identificação do idioma selecionado, devemos realizar as verificações para o ajuste da tradução através do bloco `if`. Insira um bloco de controle `if`, e nele encaixe um bloco de comparação com o sinal de igual, da guia `Built-in` na seção `Logic`. Dentro dele, é necessário inserir a variável que contém o número do idioma selecionado para ser comparado com o número do primeiro idioma que definimos

E-book gerado especialmente para Carla Fabiane Calixto - carlacalixto16@gmail.com

anteriormente.

Caso essa comparação seja verdadeira, faremos as configurações do código da língua para o app realizar a tradução. Insira um bloco de variável `set global idioma to` para receber o bloco de texto com o valor `pt-de` que traduzirá do português para o alemão. Para configurarmos o leitor com voz nesse idioma, necessitamos inserir os blocos `Language` e `Country` do `TextToSeech1` , e neles colocar um bloco de texto com o código `DEU` para a vocalização.

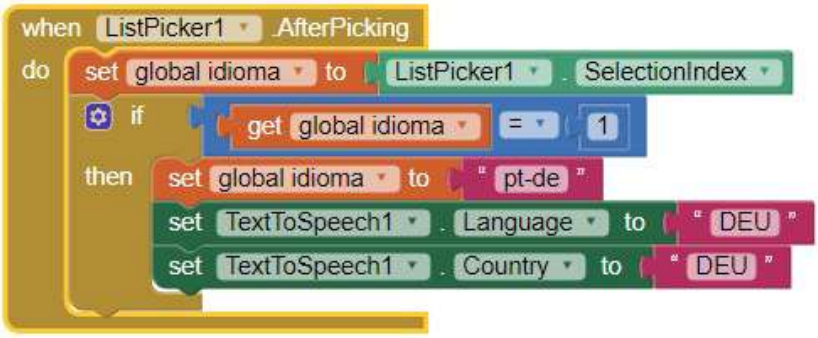

Figura 5.22: Idioma alemão selecionado

A prática demonstrada anteriormente deverá ser realizada para os demais idiomas disponíveis. Veja na tabela a seguir os dados para a configuração dos idiomas e realize as configurações dos blocos faltantes.

Idioma	Código	Linguagem	País
Espanhol	pt-es	esp	ESP
Inglês	pt-en	en	USA
Italiano	pt-it	it	ITA

E-book gerado especialmente para Carla Fabiane Calixto - carlacalixto16@gmail.com

Após a inserção dos blocos, veja como deverá ficar seu bloco de seleção de idioma:

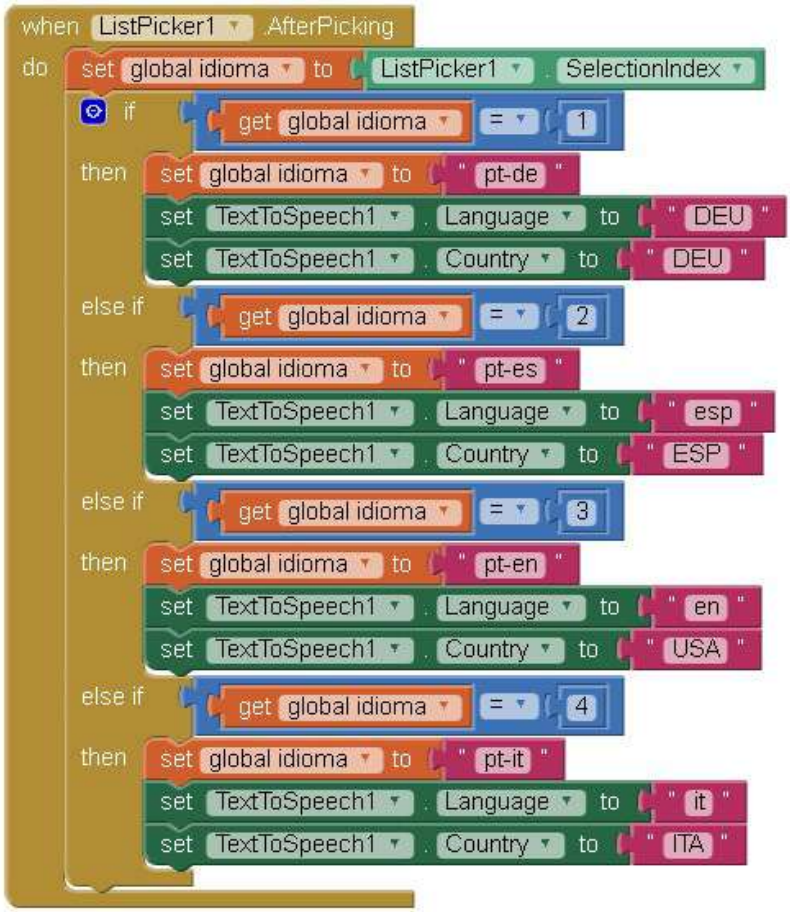

Figura 5.23: ListPicker que seleciona o idioma para tradução

Uma observação importante para esse momento é que, após inserir o bloco de controle `if`, temos de colocar os blocos de `else if`, que realizarão uma outra decisão caso a anterior não

E-book gerado especialmente para Carla Fabiane Calixto - carlacalixto16@gmail.com

seja verdadeira. Precisaremos testar as quatro opções de idioma possíveis, dependendo da escolha feita pelo usuário.

Para isso, clique no ícone de configuração ao lado esquerdo do bloco `if` (na cor azul) e arraste o bloco `else if` para dentro do controle:

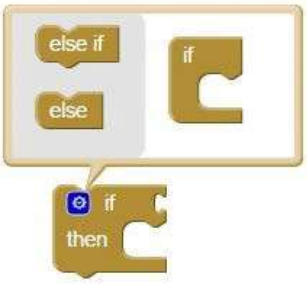

Figura 5.24: Botão de configuração do IF

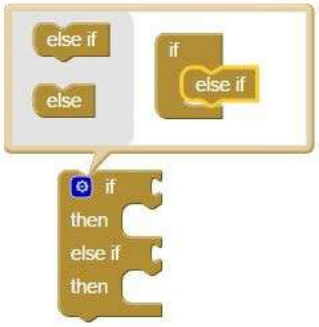

Figura 5.25: If com a opção ELSE IF

5.7 COMPARTILHANDO SUA TRADUÇÃO

O componente `ActivityStarter` permite que você combine aplicativos fazendo com que um inicie outras aplicações. É uma

E-book gerado especialmente para Carla Fabiane Calixto - carlacalixto16@gmail.com

maneira fundamental de ampliar os recursos do App Inventor, aproveitando outras ferramentas, sejam elas aplicativos criados com o App Inventor ou não. Eles também podem ser aplicativos como Câmera e Maps, que estão pré-instalados no dispositivo.

Como dissemos no início deste capítulo, teremos um botão que vai compartilhar a tradução com um de seus contatos do aplicativo externo WhatsApp. Como se trata do uso de um aplicativo que não foi desenvolvido por nós, o nosso deverá enviar informações para ele. Para essa tarefa, precisaremos realizar alguns ajustes no componente `ActivityStarter`, pois é ele que ativará o WhatsApp.

Esses ajustes ocorrerão quando o usuário clicar no botão `btn_whats`. Então, é necessário inserir um bloco do componente `When btn_whats.Click` e, no seu interior, vamos ter vários blocos de configuração do `ActivityStarter1`. A tabela a seguir exibe os blocos que deverão ser inseridos e seus valores.

Linha	Propriedade	Valor para digitação
1	Action	android.intent.action.SEND
2	ActivityPackage	com.whatsapp
3	ActivityClass	com.whatsapp.ContactPicker
4	DataType	text/plain
5	ExtraKey	Android.intent.extra.TEXT
6	ExtraValue	Bloco com o texto traduzido

Vamos entender cada **propriedade** do `ActivityStarter1` e seus **valores** que foram alterados:

1. Define a ação a ser realizada pelo seu aplicativo. Nesse caso,

E-book gerado especialmente para Carla Fabiane Calixto - carlacalixto16@gmail.com

gostaríamos de enviar os dados, por isso, utilizamos a opção `SEND`.

2. Indica o nome do aplicativo que queremos usar.
3. Indica qual a funcionalidade do aplicativo que desejamos usar. Aqui foi indicado que utilizaremos a opção de abrir a lista de contatos `ContactPicker`.
4. Indica o tipo de informação que vamos enviar para o aplicativo. Nesse caso, a informação é do tipo texto.
5. O ExtraKey representa qualquer informação adicional. Nesse caso, estamos dizendo que vamos usar o envio de um texto para o WhatsApp.
6. Indica qual texto será enviado. Em nosso caso, vamos encaminhar o conteúdo do componente que contém o texto traduzido, ou seja, o `TRADUCAO.TEXT`.

Veja como deverá ficar momentaneamente configurado o bloco do botão `btn_whats`.

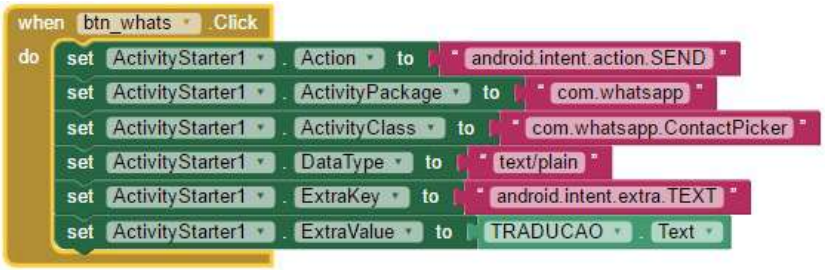

Figura 5.26: Configurações do ActivityStarter

Após as configurações do `ActivityStarter1` que preparam

E-book gerado especialmente para Carla Fabiane Calixto - carlacalixto16@gmail.com

o seu app para enviar dados, teremos de verificar se o usuário tem esse aplicativo instalado em seu dispositivo. Para tanto, é necessário inserir um bloco `if` e realizar a verificação de sua existência.

Caso o WhatsApp não esteja instalado, informaremos ao usuário por meio de um bloco do `notifier` ; caso contrário, ativaremos a chamada do WhatsApp utilizando o comando `Call ActivityStarter1.StartActivity` . Veja a seguir como deverá ficar o bloco `if` com a verificação da instalação do WhatsApp.

Figura 5.27: Verificando se o WhatsApp está instalado

A seguir, vemos uma imagem com o bloco completo do `btn_whats` .

E-book gerado especialmente para Carla Fabiane Calixto - carlacalixto16@gmail.com

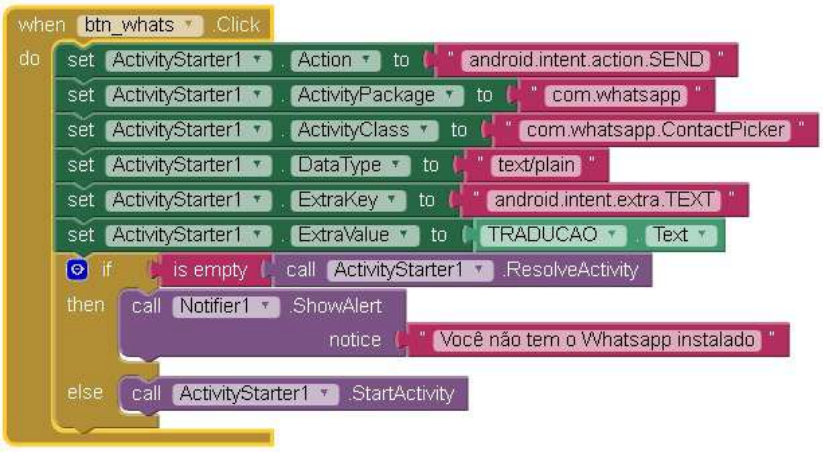

Figura 5.28: btn_whats completo

5.8 TESTANDO O APLICATIVO

Chegamos ao momento de testarmos o nosso tradutor. Clique para emular e instale-o no seu dispositivo físico, pois assim você poderá testar todos os recursos do aplicativo. A figura a seguir exibe a tela do reconhecimento de voz ativado pelo botão `btn_reconhece_voz` .

E-book gerado especialmente para Carla Fabiane Calixto - carlacalixto16@gmail.com

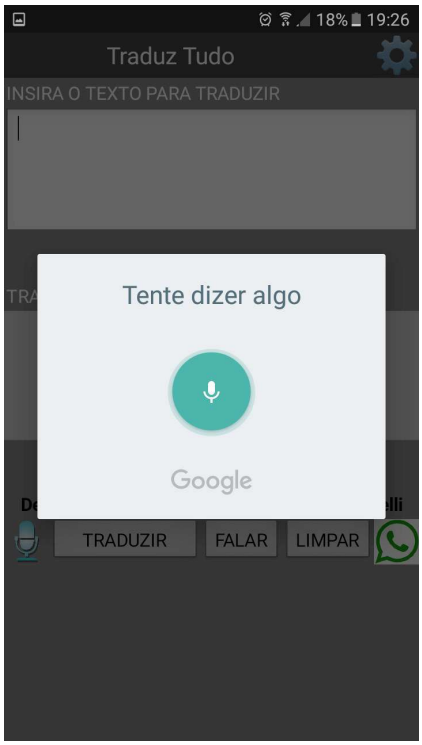

Figura 5.29: Reconhecimento de voz ativado

Após o usuário digitar ou falar um texto, será realizada uma tradução do português para o inglês, ao clicarmos no botão `btn_Traduzir` . Seu resultado pode ser visto a seguir.

E-book gerado especialmente para Carla Fabiane Calixto - carlacalixto16@gmail.com

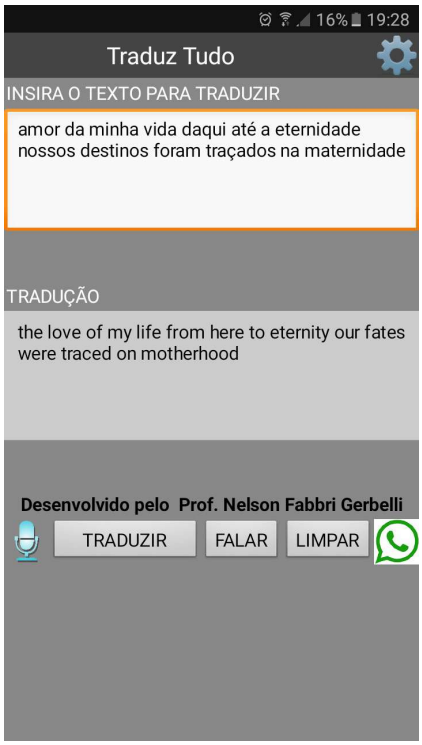

Figura 5.30: Tradução já realizada pelo app

Como a tradução é realizada por uma API, será necessário que o leitor possua em seu dispositivo móvel uma conexão com a internet; caso contrário, não será possível realizar a tradução.

Teste cada um dos botões que foram programados e, caso ocorra algum erro, volte para verificar os blocos do botão que o ocasionou, corrigindo-os se necessário. Uma dica importante em caso de erro é verificar os comandos que foram digitados. Veja se estão da maneira que indicamos, respeitando as letras maiúsculas e minúsculas, pois esse pequeno detalhe é uma das grandes causas de vários problemas na execução do aplicativo.

E-book gerado especialmente para Carla Fabiane Calixto - carlacalixto16@gmail.com

Deixo aqui uma sugestão e, ao mesmo tempo, um desafio para o leitor que concluiu o app de tradução: você poderá trocar ou inserir novas línguas conforme a sua necessidade.

5.9 RESUMINDO

No próximo capítulo, o leitor vai produzir um aplicativo que o ajudará a encontrar o local onde deixou seu carro estacionado. Utilizando o geolocalizador do dispositivo em conjunto com uma API da Google, poderemos traçar a rota para chegar até ele.

E-book gerado especialmente para Carla Fabiane Calixto - carlacalixto16@gmail.com

CAPÍTULO 6

ONDE ESTACIONEI?

Como toda solução nasce de um problema, desenvolvi este aplicativo pensando nas pessoas que, assim como eu, acabam por ter dificuldades de lembrar onde estacionaram seu carro e como encontrá-lo novamente. O app **Onde Estacionei?** nos ajudará traçando a rota de volta até o veículo. Este app também poderá ser útil em outras oportunidades, como voltar a um ponto de partida de um lugar pouco conhecido, geralmente em viagens.

O aplicativo funcionará recebendo o endereço e as coordenadas do satélite através do componente `LocationSensor` somente quando solicitado pelo usuário. Também armazenará as informações do local do estacionamento pelo componenete `TinyDB` . Quando desejarmos retornar ao ponto inicial, deveremos clicar no botão para localizar as atuais coordenadas do satélite, e solicitar que uma API exiba o mapa e trace a rota até o destino.

Por ser um componente que recebe valores do satélite, o `LocationSensor` funcionará apenas a céu aberto. Haverá consumo de dados de internet apenas para o uso do mapa, pois a rota é realizada através da API externa do Google Maps.

O leitor deverá ter instalado em seu dispositivo o aplicativo

E-book gerado especialmente para Carla Fabiane Calixto - carlacalixto16@gmail.com

Google Maps. Caso não o possua, vá até a *Play Store* e realize o download e a instalação do app *GPS e transporte público*.

6.1 INICIANDO O LAYOUT DO APP

Vamos criar um novo projeto no App Inventor, dando-lhe o nome de `Onde_Parei`. Antes de começarmos a desenvolver o layout do projeto, podemos verificar na figura a seguir como ficará a tela do nosso aplicativo.

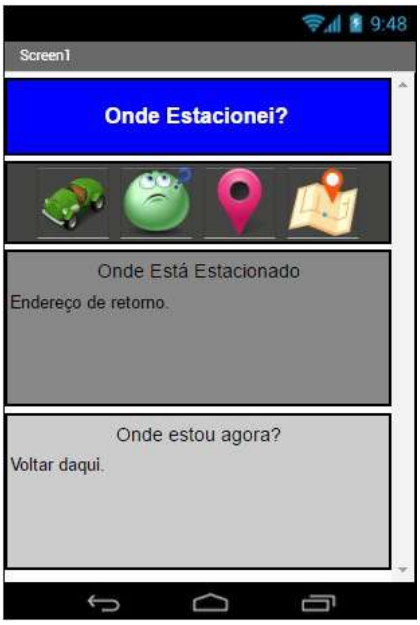

Figura 6.1: Tela do aplicativo Onde Estacionei?

Para o desenvolvimento dessa tela, vamos precisar de dois `HorizontalArrangement`. O primeiro servirá para a barra de títulos de seu aplicativo, e o segundo receberá os botões que terão as funções de localização do endereço do estacionamento. Teremos

E-book gerado especialmente para Carla Fabiane Calixto - carlacalixto16@gmail.com

algumas `Labels` que serão responsáveis pela exibição de informações e dois `VerticalArrangements` que farão parte da estética do aplicativo, servindo apenas para a organização das informações internas.

A tabela a seguir demonstra os componentes que deveremos inserir, o local onde os encontramos e as propriedades que devemos alterar. Vamos começar configurando a sua tela. Sem inserir nenhum componente ainda, realize as modificações:

Componente	Pallete	Propriedade	Alterar valor para
Screen	- -	AppName	Onde Estacionei?
		Icon	car.png
		TitleVisible	Desmarcar

Como já vimos em capítulos anteriores, alteramos o nome de exibição do aplicativo quando ele for instalado em `AppName`, realizamos o upload do ícone `car.png` e desabilitamos o título padrão da `Screen` para podermos criar nossa própria barra de título.

Para criar a nossa barra de título personalizada, vamos novamente utilizar o componente `HorizontalArrangement` para, em seu interior, inserir uma `Label`.

Componente	Pallete	Propriedade	Alterar valor para
HorizontalArrangement	Layout	AlignHorizontal	Center
		AlignVertical	Center
		BackgroundColor	Blue
		Height	60 pixels

E-book gerado especialmente para Carla Fabiane Calixto - carlacalixto16@gmail.com

		Width	Fill Parent
Label	User Interface	FontBold	Marcar
		FontSize	20
		Text	Onde Estacionei?
		TextColor	White

Para o componente `HorizontalArrangement` , realizamos o alinhamento de seu conteúdo ao centro, tanto na propriedade `AlignHorizontal` como na `AlignVertical` . Deixamos a cor de fundo selecionada como azul (`Blue`), e alteramos sua altura e seu comprimento horizontal.

Já com a `Label` que exibirá o título do aplicativo, mudamos o tamanho da fonte para `20` . Alteramos a propriedade `Text` para exibir a informação `Onde Estacionei?` e deixamos a cor do texto marcada como `White` .

A figura a seguir exibe a tela desenvolvida até o momento:

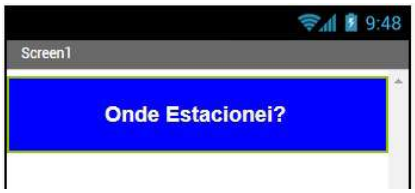

Figura 6.2: Estágio atual do desenvolvimento

Após configurar a barra de título, necessitamos de um `HorizontalArrangement` para organizar os quatro botões que realizarão todo o trabalho do app. São eles:

E-book gerado especialmente para Carla Fabiane Calixto - carlacalixto16@gmail.com

- **1º botão:** localiza suas coordenadas de estacionamento e salva em seu dispositivo.
- **2º botão:** exibe o endereço do local do estacionamento.
- **3º botão:** localiza o ponto de partida para retornar ao local de estacionamento.
- **4º botão:** traça a rota de retorno ao carro e exibe-a em um mapa.

Vamos preparar essa nova parte da `Screen` : insira logo abaixo da barra de título um componente `HorizontalArrangement` e configure-o conforme demonstrado na tabela a seguir. Após essas configurações, insira os quatro botões de trabalho no interior da `Screen` , não se esquecendo de configurá-los com os dados da tabela.

Componente	Pallete	Propriedade	Alterar valor para
HorizontalArrangement	Layout	AlignHorizontal	Center
		BackgroundColor	Dark Gray
		Width	Fill Parent
Button	User Interface	Height	60 pixels
		Width	60 pixels
		Image	save.png
		Text	apagar o texto
		Rename	Btn_Guardar
Button	User Interface	Height	60 pixels
		Width	60 pixels

E-book gerado especialmente para Carla Fabiane Calixto - carlacalixto16@gmail.com

		Image	onde.png
		Text	apagar o texto
		Rename	Btn_Exibir
Button	User Interface	Height	60 pixels
		Width	60 pixels
		Image	aqui.png
		Text	apagar o texto
		Rename	Btn_EstouAqui
Button	User Interface	Height	60 pixels
		Width	60 pixels
		Image	mapa.png
		Text	apagar o texto
		Rename	Btn_Chegar

Com o `HorizontalArrangement`, deixamos a propriedade `AlignHorizontal` marcada como `Center`, com a intenção de alinhar os botões, além de indicar que o componente utilizará todo o espaço horizontal. Para padronizar o tamanho dos botões, marcamos as propriedades `Height` e `Width` como `60` pixels.

Realize o upload das imagens baixadas. Temos uma imagem específica para identificar o que fará cada botão. Como queremos que neles apareçam apenas as figuras, apagamos o texto da propriedade `Text`. Alteramos também cada nome dos componentes `Button` para uma melhor identificação no momento da programação dos blocos.

A figura a seguir exibe a tela desenvolvida com os botões de

E-book gerado especialmente para Carla Fabiane Calixto - carlacalixto16@gmail.com

controle:

Figura 6.3: Estágio atual do desenvolvimento

Agora prepararemos um espaço para a exibição do local onde você estacionou seu veículo. Vamos utilizar um `VerticalArrangement` para organizar as informações em duas linhas. A primeira vai conter uma `Label` indicando o texto **Onde está estacionado**, e a segunda mostrará o endereço em que você estacionou.

Vamos lá. Insira abaixo do `HorizontalArrangement` dos botões um componente `VerticalArrangement` e, dentro dele, insira duas `Labels` . Realize as configurações indicadas na tabela:

Componente	Pallete	Propriedade	Alterar valor para
VerticalArrangement	Layout	BackgroundColor	Gray
		Heigth	30 percent
		Width	Fill Parent
Label	User Interface	FontBold	Marcar
		FontSize	18
		Width	Fill Parent

Componente	Pallete	Propriedade	Alterar valor para

E-book gerado especialmente para Carla Fabiane Calixto - carlacalixto16@gmail.com

		Text	Onde está estacionado
		TextAlignment	Center
Label	User Interface	FontSize	16
		Text	Endereço de retorno.
		Rename	Lbl_Carro

No `VerticalArrangement` , alteramos sua cor de fundo, e deixamos sua altura e seu comprimento horizontal definidos. Na primeira `Label` , deixamos a formatação em negrito, alteramos o tamanho da fonte, e seu comprimento ocupará todo o espaço da linha em que se encontra. Escrevemos o texto `Onde está estacionado` na propriedade `Text` que ficará centralizada na linha. Já para a segunda `Label` , alteramos o tamanho da fonte, mudamos o texto de exibição para `Endereço de retorno.` e também a renomeamos para `Lbl_Carro` .

Figura 6.4: Estágio atual do desenvolvimento

Da mesma maneira que inserimos o `VerticalArrangement` anterior com duas `Labels` , necessitamos repetir o mesmo

E-book gerado especialmente para Carla Fabiane Calixto - carlacalixto16@gmail.com

processo para exibir o endereço atual para retornar ao veículo estacionado. Insira outro componente abaixo do `VerticalArrangement` e, em seu interior, mais duas `Labels`, com as seguintes configurações:

Componente	Pallete	Propriedade	Alterar valor para
VerticalArrangement	Layout	BackgroundColor	Light Gray
		Heigth	30 percent
		Width	Fill Parent
Label	User Interface	FontBold	Marcar
		FontSize	18
		Width	Fill Parent
		Text	Onde estou agora?
		TextAlignment	Center
Label	User Interface	FontSize	16
		Text	Voltar daqui..
		Rename	Lbl_Estou

Veja a tela final do aplicativo:

E-book gerado especialmente para Carla Fabiane Calixto - carlacalixto16@gmail.com

Figura 6.5: Tela final do aplicativo

A seguir, inseriremos os componentes não visíveis que farão parte do projeto:

Componente	Pallete	Propriedade	Alterar valor para
LocationSensor	Sensors	Enabled	Desmarcar
		Rename	Location_Parado

O componente `LocationSensor` , encontrado na `Pallete Sensors` , identifica a latitude e a longitude do satélite, bem como o endereço correspondente a essas coordenadas. Após sua inserção na `Screen` , desmarcamos a sua propriedade com `Enabled` , pois ele só será ativado quando houver uma solicitação do usuário. Alteramos também seu nome para `Location_Parado` para melhor identificação no momento em que estivermos trabalhando

E-book gerado especialmente para Carla Fabiane Calixto - carlacalixto16@gmail.com

com os blocos de programação.

Para armazenar as informações do local do estacionamento e para que elas fiquem disponíveis mesmo após o encerramento da execução do app, necessitamos de um componente da guia `Storage` : o `TinyDB` . Basta inseri-lo e nenhuma alteração nas propriedades será necessária.

Insira também o componente não visível `Notifier` , responsável por transmitir ao usuário a solicitação se queremos ou não salvar o endereço do estacionamento. Não se preocupe ainda com a resolução lógica do aplicativo, pois mais adiante veremos os procedimentos passo a passo.

Agora, necessitamos de um componente que realize a conexão do nosso aplicativo com o app *Google Maps*. Insira um componente `ActivityStarter` da guia `Connectivity` , e realize as seguintes alterações nas propriedades:

Linha	Propriedade	Alterar valor para
1	action	android.intent.action.VIEW
2	Activity Class	com.google.android.maps.MapsActivity
3	Activity Package	com.google.android.apps.maps

1. Define a ação a ser realizada pelo seu aplicativo. Nesse caso, gostaríamos de ter uma visão das informações, por isso utilizamos a opção `VIEW` .
2. Indica o nome do aplicativo que queremos utilizar, ou seja, o `MapsActivity` .
3. Indica qual a funcionalidade do aplicativo que desejamos

E-book gerado especialmente para Carla Fabiane Calixto - carlacalixto16@gmail.com

usar. Aqui indicamos que utilizaremos a opção de exibição do mapa, através da `maps` .

Da mesma maneira que inserimos o componente `Location_Parado` , deveremos inserir mais um `LocationSensor` , agora usando-o para a identificação do local de retorno. Após sua inclusão na `Screen` , altere os seus valores conforme demonstra a tabela:

Componente	Pallete	Propriedade	Alterar valor para
LocationSensor	Sensors	Enabled	Desmarcar
		Rename	Location_EstouAqui

A figura a seguir exibe os componentes não visíveis inseridos na área de design:

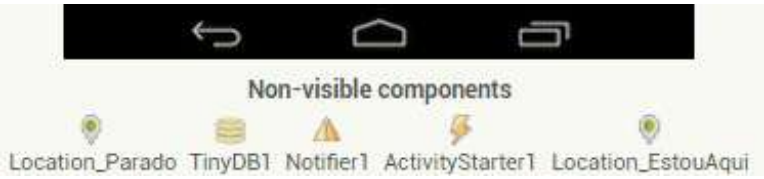

Figura 6.6: Componentes não visíveis

6.2 IDENTIFICANDO A LATITUDE E A LONGITUDE DO ESTACIONAMENTO

Para identificarmos o local em que estacionamos o veículo, precisamos armazenar em variáveis as coordenadas da latitude e da longitude logo que estacionarmos o carro. Você consegue identificá-las, utilizando o componente `LocationSensor` . Lembre-se de que ele foi renomeado para `Location_Parado` .

E-book gerado especialmente para Carla Fabiane Calixto - carlacalixto16@gmail.com

O componente `LocationSensor` ativa a geolocalização em seu dispositivo diretamente do satélite, não necessitando de uso de dados da internet. Porém, é preciso que o leitor tenha esse recurso em seu dispositivo.

Note também que desmarcamos a propriedade `Enabled` do componente `LocationSensor`, pois ele será habilitado para receber os dados da sua localização somente quando clicarmos no botão `Btn_Guardar`.

Primeiramente precisamos criar as variáveis para guardar as coordenadas de origem e também do ponto final, para quando quisermos voltar para encontrar o veículo estacionado. Inicialize quatro variáveis da guia `Built-in` e nomeie-as como `Latitude_car`, `Longitude_car`, `Latitude_atual` e `Longitude_atual`, em que as `Latitude_car` e `Longitude_car` indicarão o local onde você estacionou o veículo, e as `Latitude_atual` e `Longitude_atual` indicarão o ponto do retorno.

Deixe as quatro variáveis criadas como numéricas; elas só poderão receber números. Para isso, encaixe um bloco numérico da guia `Math` em cada uma. Em caso de dúvidas quanto à criação de variáveis, retome a leitura do capítulo *Uma simples calculadora*.

E-book gerado especialmente para Carla Fabiane Calixto - carlacalixto16@gmail.com

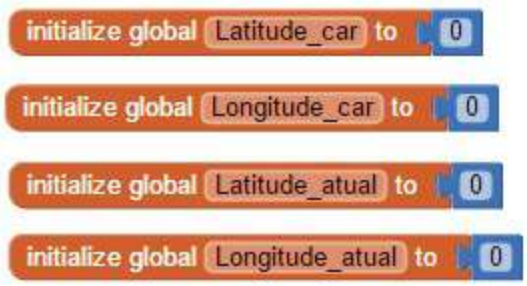

Figura 6.7: Variáveis do aplicativo

Precisamos ativar o componente `LocationSensor` para que ele receba a latitude e a longitude de onde acabamos de estacionar. Conseguimos isso apenas alterando a propriedade `Enabled` de `false` para `true`. Isso acontecerá quando o usuário clicar no `Btn_Guardar`.

Para que isso aconteça, insira um bloco `When Btn_Guardar.Click` na sua área `Viewer` e, dentro desse bloco, colocaremos um comando para ativar o localizador. Selecione o comando `set Location_Parado.Enable to` e, logo na sequência, encaixe um bloco lógico `true`. A figura a seguir exibe o `Btn_Guardar` com o localizador ativado.

Figura 6.8: Btn_Guardar ativando o geolocalizador

Quando ativarmos o geolocalizador, ele receberá as coordenadas e também o endereço do local onde estamos

E-book gerado especialmente para Carla Fabiane Calixto - carlacalixto16@gmail.com

estacionando. Devemos exibir o endereço em uma `Label` e guardar as coordenadas nas variáveis. Porém, antes de guardar os valores, precisamos ter certeza de que o `LocationSensor` já recebeu os valores do satélite, pois às vezes demora alguns segundos para essa tarefa ser executada.

Como não temos como saber o tempo exato que isso tomará, precisamos verificar constantemente. Teremos uma variável para guardar a informação assim que o endereço for exibido. Se essa variável de controle estiver marcada como verdadeira, podemos continuar e armazenar os dados nas variáveis `Latitude_car` e `Longitude_car` ; caso contrário, necessitaremos aguardar mais um tempo.

Precisamos criar a variável que verifica se o endereço foi recebido. Crie mais uma variável, atribua o nome de `verifica` e o valor `false` , pois só vamos continuar quando a variável tiver um valor verdadeiro `true` .

Figura 6.9: Criação da variável verifica

O componente que recebe as informações da localização que devemos inserir na área `Viewer` é o `when Location_Parado.LocationChanged` . A figura a seguir mostra o bloco para recebimento da latitude e longitude.

E-book gerado especialmente para Carla Fabiane Calixto - carlacalixto16@gmail.com

Figura 6.10: Recebendo os dados do localizador

Dentro desse bloco, necessitamos atribuir o endereço recebido pelo localizador para a `Lbl_Carro` . Primeiramente insira um `set Lbl_Carro.Text to` para receber o valor do endereço que está em `Location_Parado.CurrentAddress` , que deverá ser encaixado no `Llb_Carro` . Veja a seguir o bloco que exibe o endereço recebido pelo localizador.

Figura 6.11: Exibindo o endereço na Lbl_Carro

Após a exibição do endereço, devemos marcar a variável `verifica` como `true` , e assim guardar as informações nas variáveis. Insira um bloco para alterar o valor da variável. A próxima imagem exibe o bloco `when Location_Parado.LocationChanged` até o momento.

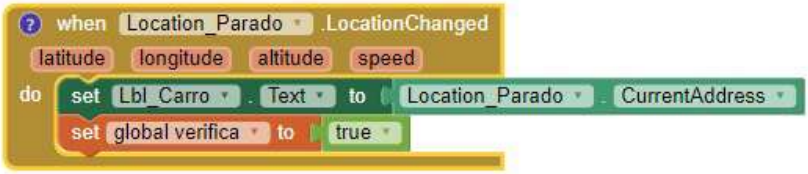

Figura 6.12: Recebendo todos os dados do localizador

E-book gerado especialmente para Carla Fabiane Calixto - carlacalixto16@gmail.com

Precisamos sempre confirmar se a variável `verifica` está com o valor `true`, pois é a partir desse ponto que armazenaremos os dados nas `Latitude_car` e `Longitude_car`. Existe um bloco que realiza uma verificação constante, o `While`. Sempre após esse bloco, existe uma comparação e, se o resultado dela for verdadeiro, será dada sequência aos comandos. A figura a seguir exibe o bloco `While test do` que deverá ser inserido:

Figura 6.13: Bloco While

Note que, após o comando `While`, existe uma opção de `test`, e é nesse espaço que deveremos testar se a variável `verifica` está com o valor verdadeiro. Insira um bloco lógico de comparação da guia `Built-in` logo após a opção `test`. Dentro desse bloco, insira `set global verifica` e, após o símbolo de igualdade, encaixe um bloco lógico `true`.

Figura 6.14: Bloco While com o teste de verificação

Quando o teste anterior for verdadeiro, precisaremos alterar o valor da variável `verifica` para `false`, para futura utilização do aplicativo. As variáveis `Latitude_car` e `Longitude_car` deverão receber os valores referentes às coordenadas do estacionamento do veículo, e é neste ponto que isso acontece.

E-book gerado especialmente para Carla Fabiane Calixto - carlacalixto16@gmail.com

Insira as variáveis `set global Latitude_car to` e `set global Longitude_car to` da guia `Built-in` . Para a variável que armazena a **latitude**, selecione a opção `get latitude` apenas posicionando o ponteiro do mouse sobre a opção e arrastando-a para encaixar na variável. Repita o mesmo procedimento para o encaixe da longitude na variável `Longitude_car` .

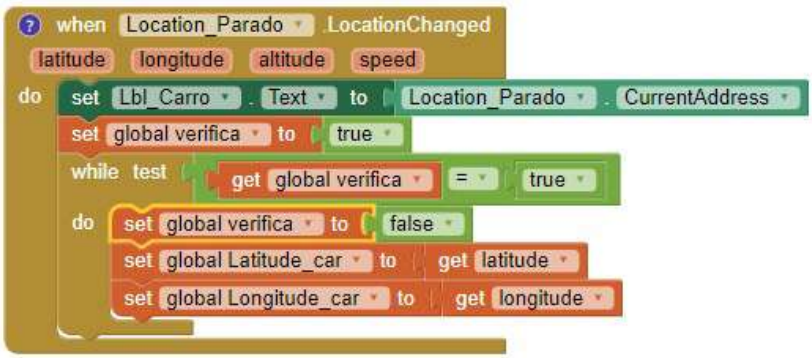

Figura 6.15: Recebendo os dados do localizador

Antes de realmente salvarmos a localização do carro, é preciso perguntar ao usuário do aplicativo se ele realmente deseja guardar os dados. Caso responda que **Sim**, efetivaremos a gravação do local; mas se o usuário responder **Não** ou **Cancelar**, nada será executado.

6.3 ARMAZENANDO A LATITUDE E A LONGITUDE

Necessitamos criar uma caixa de diálogo com a pergunta **Salvar endereço atual?**, e os botões de resposta **Sim**, **Não** e **Cancelar**. Insira um bloco do componente

E-book gerado especialmente para Carla Fabiane Calixto - carlacalixto16@gmail.com

`Notifier.ShowChooseDialog` para realizar essa tarefa.

Note as opções de encaixe: `message`, `title`, `button1Text`, `button2Text` e `cancelable`. Nelas deverão ser encaixados blocos de texto com as seguintes informações: em `message`, digite **Salvar endereço atual?**. Em `title`, indicando o que será exibido na caixa de mensagem, digite **Atualizar?**. Já nos `button1Text` e `button2Text`, insira **Sim** e **Não**, respectivamente, para nomear os botões de resposta nos quais o usuário deverá clicar.

Na opção `cancelable`, já temos um bloco lógico definido como `true`. Ou seja, o botão de **cancelar** já vem configurado para exibição, mas, caso não queira que ele apareça, altere seu valor para `false`. Veja na figura como ficará o bloco do botão `when Location_Parado.LocationChanged` quando finalizado:

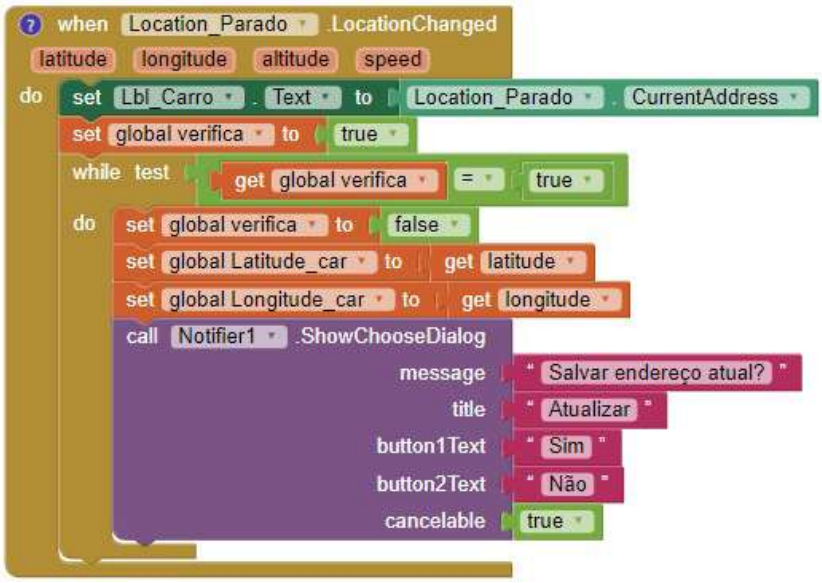

Figura 6.16: Location_Parado finalizado

E-book gerado especialmente para Carla Fabiane Calixto - carlacalixto16@gmail.com

Ainda não configuramos nenhuma ação no clique dos botões do `Notifier`, que deseja saber se vamos salvar ou não as informações de parada. Para identificar a resposta do usuário, precisamos inserir o componente `When Notifier1.AfterChoosing`. Dentro do `Notifier`, colocaremos a decisão responsável por verificar se o usuário clicou no botão **Sim**.

Para isso, insira um bloco de controle `if` e, logo em seguida, selecione um bloco de texto que realize a comparação do botão clicado com o que virá do aplicativo. Para receber a escolha do usuário, posicione o ponteiro do mouse sobre `choice` e selecione a opção que surgirá `get choice` para, logo após, encaixar a decisão `compare texts`.

Finalmente, insira uma caixa de texto e, dentro dela, o nome do botão que você deseja tratar. Nesse caso, queremos programar o botão **Sim**, que gravará os dados da localização do usuário. A próxima figura exibe como fica a configuração do bloco `if`.

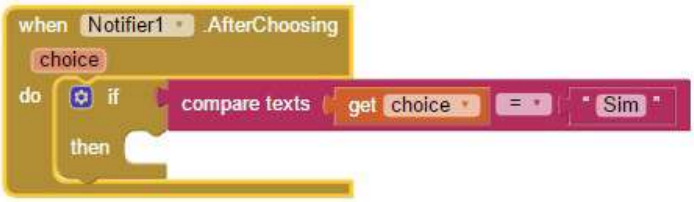

Figura 6.17: Verificando a escolha do usuário

Como é necessário guardar os valores da latitude, da longitude e do endereço do veículo estacionado, usaremos o componente `TinyDB`. Com ele, é possível armazenar dados e posteriormente recuperá-los. Essa é uma solução simples para armazenamento de

E-book gerado especialmente para Carla Fabiane Calixto - carlacalixto16@gmail.com

informações, mas atende exatamente ao que desejamos para o aplicativo.

O bloco `Call TinyDB1.StoreValue` , encontrado na guia de `Blocks` ao clicar sobre o componente `TinyDB1` , armazenará no dispositivo a `tag` (um nome de variável) e o valor através da opção `valueToStore` . Teremos de guardar o endereço, latitude e longitude do local onde estacionamos o veículo. Um único componente `TinyDB` pode armazenar várias informações, basta ir criando várias `tags` diferentes.

Incluiremos um bloco `Call TinyDB1.StoreValue` logo após o comando `then` do bloco `if` . Na opção `tag` , devemos colocar um bloco de texto com o nome da variável que armazenará o conteúdo. Na opção `valueToStore` , encaixe a `Label` que exibe o endereço recebido pelo componente `LocationSensor` , ou seja, a `Lbl_Carro.text` . A figura exibe o bloco que grava o endereço no componente `TinyDB1` :

Figura 6.18: Salvando o endereço no TinyDB1

Necessitamos salvar também a latitude e a longitude. Vamos usar o mesmo componente `TinyDB1` para armazenar esses dados, porém em `tags` diferentes.

Vamos inserir mais dois blocos do `Call TinyDB1.StoreValue` para realizar essa tarefa. No primeiro, na

E-book gerado especialmente para Carla Fabiane Calixto - carlacalixto16@gmail.com

opção `tag` , insira um bloco de texto e, dentro dele, defina o nome da variável `latitude` . Na opção `valueToStore` , encaixe a variável `get global Latitude_car` . Repita o procedimento com os dados da longitude, pois devemos salvar também sua coordenada.

Figura 6.19: Salvando a latitude e a longitude no TinyDB1

Após salvar os dados em seu dispositivo, precisamos exibir uma notificação de que tudo ocorreu bem para o usuário. Prepare um bloco `Notifier.ShowAlert` para exibir um texto com a informação **Salvo com sucesso**. A próxima figura demonstra essa tarefa.

Figura 6.20: Exibindo a confirmação

Já utilizamos o componente de localização e precisamos desabilitá-lo. Caso não o façamos, o componente ficará buscando a cada momento a posição do usuário, mesmo não havendo mais

E-book gerado especialmente para Carla Fabiane Calixto - carlacalixto16@gmail.com

necessidade, pois já estacionamos o veículo e armazenamos os dados. Então, selecione um bloco `set Location_Parado.Enabled to` e encaixe um bloco lógico `false`, conforme a figura:

Figura 6.21: Desabilitando o localizador

A próxima figura exibe o bloco completo do `Notifier.AfterChoosing`:

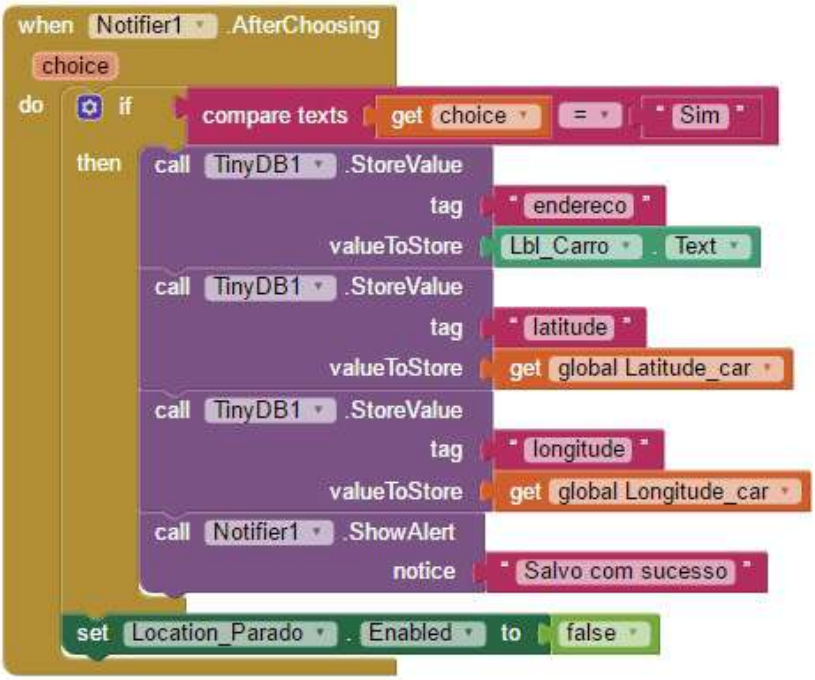

Figura 6.22: Bloco Notifier.AfterChoosing completo

E-book gerado especialmente para Carla Fabiane Calixto - carlacalixto16@gmail.com

6.4 EXIBINDO O ENDEREÇO DO ESTACIONAMENTO

Para rever o endereço do veículo que está gravado em seu app, bastar clicar no `Btn_Exibir` que as informações salvas no `TinyDB1` serão recuperadas e mostradas na `Label` de nome `Lbl_Carro.text` . Os valores da latitude e longitude não serão exibidos para o usuário e deverão ficar armazenados nas variáveis `Latitude_car` e `Longitude_car` .

Vamos inserir primeiramente um `When Btn_Exibir.Click` para, na sequência, incluir um componente `set Lbl_Carro.text to` para receber o endereço do `TinyDB1` . Para recuperar o endereço, inclua um componente `call TinyDB1.GetValue` .

Na opção `tag` , encaixe um bloco de texto e digite o nome da `tag` que contém o endereço salvo anteriormente. Nesse caso, digite apenas **endereco**. Na opção `valueIfTagNotThere` , coloque uma caixa de texto vazia. A sua função é de não exibir nada na `Label` se a tag `endereco` não existir. A figura a seguir demonstra o bloco que exibe o endereço recuperado do `TinyDB1` .

Figura 6.23: Recuperando o endereço do TinyDB1

Como até agora exibimos apenas o endereço, falta recuperar sua latitude e sua longitude. Como as salvamos no TinyDB1, devemos atribuir seus valores às variáveis `Latitude_car` e

E-book gerado especialmente para Carla Fabiane Calixto - carlacalixto16@gmail.com

`Longitude_car` .

Insira o bloco da variável `set global Latitude_car to` para receber o valor e, na sequência, encaixe um componente `call TinyDB1.GetValue` com o texto `latitude` na opção da `tag` . Em `valueIfTagNotThere` , coloque também uma caixa de texto vazia como demonstrado anteriormente. Necessitamos repetir esse processo para a variável `Longitude_car` .

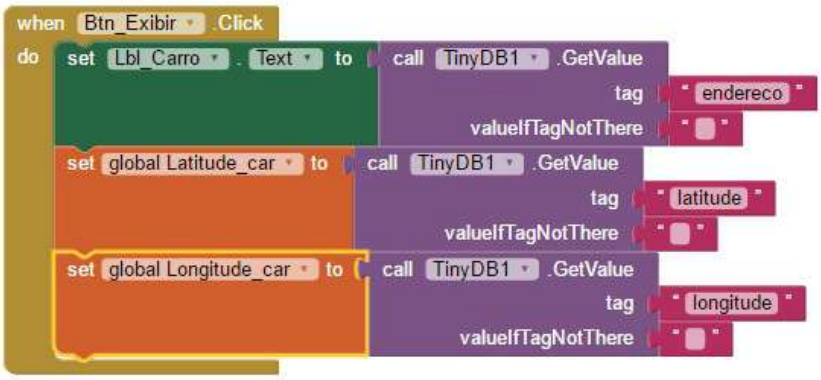

Figura 6.24: Btn_Exibir

6.5 LOCALIZANDO O PONTO DE PARTIDA

Para encontrar o ponto de retorno para o seu veículo, precisamos clicar no botão `Btn_EstouAqui` que identificará as suas coordenadas atuais de localização. Clicando nele, será ativado o geolocalizador `Location_EstouAqui` que então receberá os dados da latitude e longitude para armazenamento nas variáveis `Latitude_atual` e `Longitude_atual` . Futuramente, ele vai traçar e exibir o caminho de chegada, pois precisaremos disso para elaborar a rota entre dois pontos.

E-book gerado especialmente para Carla Fabiane Calixto - carlacalixto16@gmail.com

Iniciaremos inserindo o bloco do componente `When Btn_EstouAqui.Click` para então habilitar o geolocalizador `Location_EstouAqui`, inserindo o bloco `set Location_EstouAqui to` e encaixando o comando lógico `true` para habilitá-lo. A próxima figura exibe o bloco de ativação do `Location_EstouAqui`.

Figura 6.25: Btn_EstouAqui ativando o Location_EstouAqui

Após a sua ativação, automaticamente o componente de localização é executado e os dados são recebidos pelo aplicativo. O bloco que representa esse processo e que deverá ser inserido é o `when Location_EstouAqui.LocationChanched`.

Figura 6.26: Bloco Location_EstouAqui.LocationChanched

Como necessitamos exibir o endereço atual da sua localização indicado pelo bloco `Location_Parado` na `Lbl_Estou`, coloque um componente `Lbl_Estou.Text to`, e nele encaixe o bloco que identificou esse endereço, o `Location.Parado.CurrentAdress`. A figura a seguir mostra o bloco exibindo o endereço atual.

E-book gerado especialmente para Carla Fabiane Calixto - carlacalixto16@gmail.com

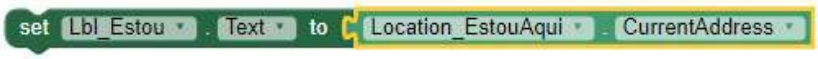

Figura 6.27: Exibindo o endereço atual

O endereço atual exibido serve apenas de informação para o usuário, porém o aplicativo deverá armazenar também a latitude e a longitude nas variáveis `Latitude_atual` e `Longitude_atual` , já que é com esses valores que traçaremos o caminho de volta.

Insira as variáveis `set global Latitude_atual` e `set global Longitude_atual` para receber os valores `get latitude` e `get longitude` , respectivamente. Veja como fica:

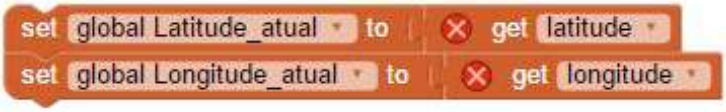

Figura 6.28: Variáveis recebendo os valores da localização atual

Com tudo já preparado, precisamos desabilitar o geolocalizador `Location_EstouAqui` , inserindo um bloco `set Location_EstouAqui to` e encaixando o comando lógico `false` . A próxima figura exibe o bloco de recebimento dos dados `Location_EstouAqui.LocationChanced` .

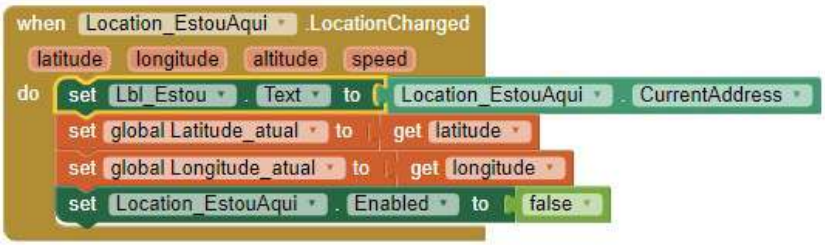

Figura 6.29: Bloco Location_EstouAqui.LocationChanched finalizado

E-book gerado especialmente para Carla Fabiane Calixto - carlacalixto16@gmail.com

6.6 TRAÇANDO A ROTA

Com todos os dados já preparados, precisamos programar o botão que exibirá o caminho de volta até o local do estacionamento do veículo. Aqui usaremos o componente `ActivityStarter1` para chamar a execução da API externa do Google Maps, que exibirá o caminho de retorno. Para isso, basta informar a URL da API, as coordenadas atuais e as iniciais, e depois inicializar o `ActivityStarter` para exibir o mapa.

Insira primeiramente o botão que executará todos os demais comandos, o `When Btn_Chegar.Click` . Dentro dele, necessitamos de um bloco `set ActivityStarter1.DataUri to` para informarmos o endereço da API do Google Maps e os pontos de partida e chegada.

Como temos várias informações para passar à API, é preciso colocar um bloco de texto `join` para juntar todos os dados. Para isso, acesse a guia `Blocks` e, na seção `Built-in` , selecione a guia dos blocos de `Text` para expandir suas opções e encontrar o bloco de texto `join` . Note que ele já vem configurado para receber duas informações.

No primeiro espaço da `join` , insira um bloco de texto e, dentro dele, informe o endereço `http://maps.google.com/maps?saddr=` . No segundo espaço, insira a variável `get global Latitude_atual` .

Configure sua `join` para receber mais seis opções de entradas de textos, pois necessitamos informar as coordenadas iniciais e finais, separadas por uma vírgula, e o comando que indica o ponto de destino: `daddr` .

E-book gerado especialmente para Carla Fabiane Calixto - carlacalixto16@gmail.com

A latitude e a longitude sempre serão separadas por vírgula (`,`). Adicione um bloco de texto e, dentro dele, insira essa pontuação.Na próxima entrada de texto da `join` , acrescente a variável `get global Longitude_atual` .

Já temos o ponto de partida, precisamos configurar agora o ponto de chegada, informando as coordenadas iniciais de onde estacionamos nosso veículo. Para dizer à API que informaremos o ponto de chegada, devemos acrescentar um bloco de texto com a informação `&daddr=` e, na sequência da entrada de textos, encaixar em cada opção as variáveis do destino `get global Latitude_car` e `get global Longitude_car` , não se esquecendo de colocar uma vírgula entre os blocos das variáveis.

Atenção

- `saddr=` : serve para definir o ponto de partida para as pesquisas de rotas. Esse ponto pode ser uma latitude, uma longitude ou um endereço formatado para consulta.
- `daddr=` : define o ponto de chegada para as pesquisas de rotas. Tem o mesmo formato e comportamento de `saddr=` .

E-book gerado especialmente para Carla Fabiane Calixto - carlacalixto16@gmail.com

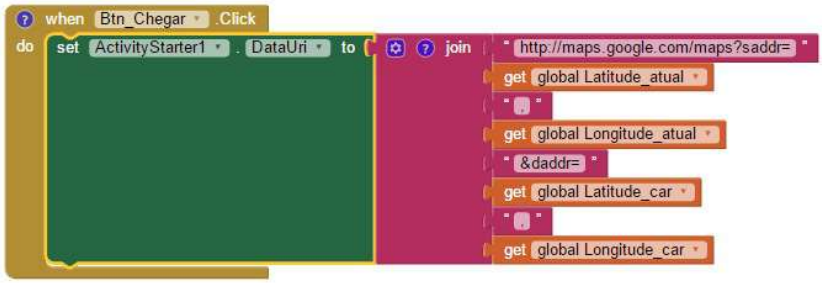

Figura 6.30: Btn_Chegar com o ActivityStarter configurado

Já está tudo pronto para a exibição da API, basta agora solicitar a execução. Coloque o bloco que realiza a chamada da API, o `call ActivityStarter1.StartActivity` . A figura a seguir exibe o botão `Chegar` com todos os códigos necessários.

Figura 6.31: Btn_Chegar com todos os códigos

6.7 TESTANDO SEU APLICATIVO

Para testar esse aplicativo, será preciso instalá-lo em seu dispositivo. Essa operação é necessária devido à utilização do geolocalizador que está presente em seu celular. Vale lembrar que, para a exibição do mapa com a rota traçada, você precisa ter acesso à internet.

E-book gerado especialmente para Carla Fabiane Calixto - carlacalixto16@gmail.com

Após a instalação, abra o aplicativo e clique no primeiro botão à esquerda. Quando o localizador encontrar o endereço, este será exibido na tela e, na sequência, solicitará a sua decisão para salvar ou não o endereço do estacionamento.

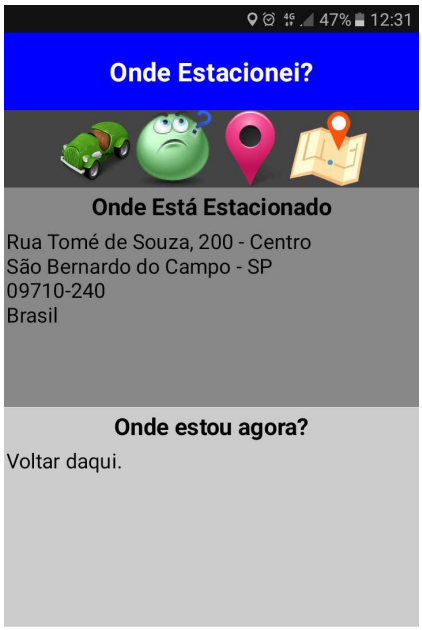

Figura 6.32: Localizando o endereço do estacionamento

O segundo botão da direita para a esquerda exibe o local atual onde você se encontra. Serve para traçar a rota de volta até o local de estacionamento. A figura a seguir exibe um exemplo de local para iniciar o retorno.

E-book gerado especialmente para Carla Fabiane Calixto - carlacalixto16@gmail.com

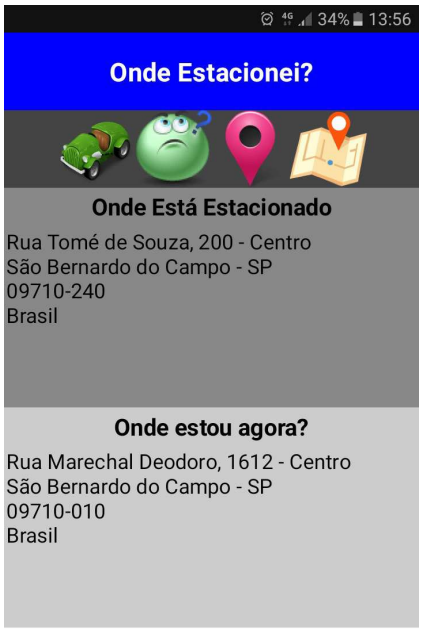

Figura 6.33: Exibindo um local para iniciar o retorno

Quando clicar no primeiro botão à direita, será exibido um mapa com a rota para você chegar até seu veículo. A próxima imagem mostra o mapa com a rota de volta ao ponto de partida.

E-book gerado especialmente para Carla Fabiane Calixto - carlacalixto16@gmail.com

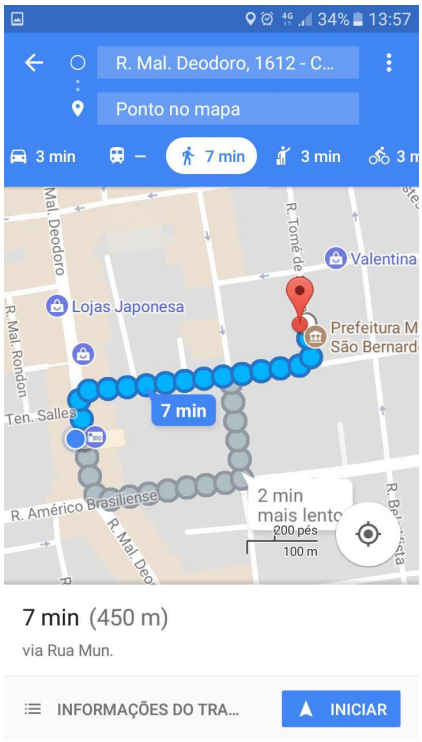

Figura 6.34: Exibindo o mapa com a rota

6.8 RESUMINDO

Neste capítulo, aprendemos a trabalhar com o geolocalizador do seu dispositivo, salvar os valores em um `TinyDB` e utilizar a API do Google Maps para traçar uma rota.

No próximo capítulo, iniciaremos o projeto que fará o nosso aplicativo acessar um banco de dados em MySQL. Veremos primeiramente a configuração do **ambiente de desenvolvimento web** *Apache* em seu computador. Assim, o App Inventor se conectará ao gerenciador de banco de dados *PhpMyAdmin* contido

E-book gerado especialmente para Carla Fabiane Calixto - carlacalixto16@gmail.com

nesse servidor.

E-book gerado especialmente para Carla Fabiane Calixto - carlacalixto16@gmail.com