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

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.

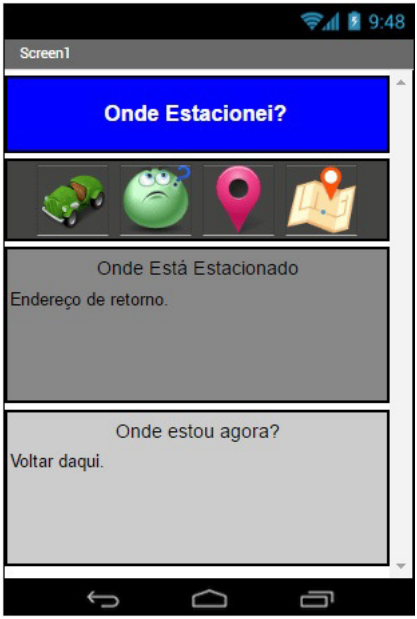

Figura 6.1: Tela do aplicativo Onde Estacionei?

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

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

		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:

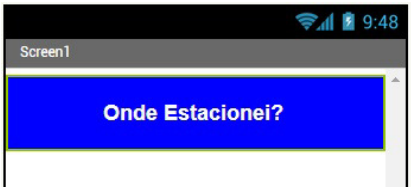

Figura 6.2: Estágio atual do desenvolvimento

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

- **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

		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

controle:

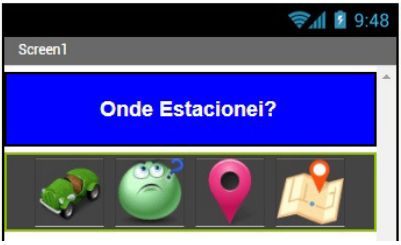

Figura 6.3: Estágio atual do desenvolvimento

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

		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

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:

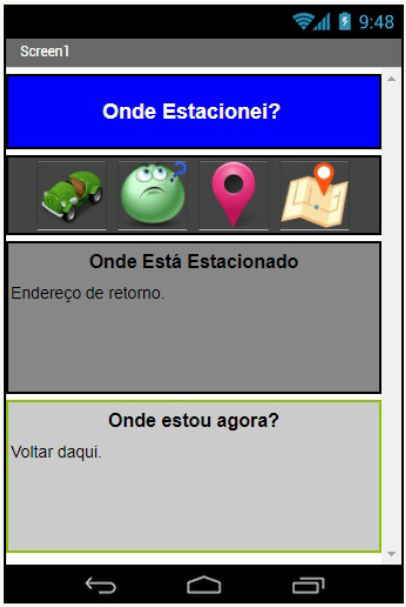

Figura 6.5: Tela final do aplicativo

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

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

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:

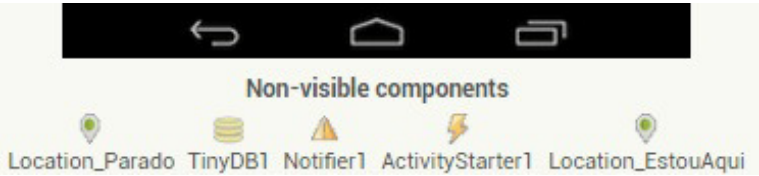

Figura 6.6: Componentes não visíveis

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`.

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*.

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

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.

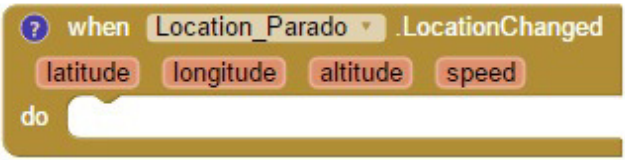

Figura 6.10: Recebendo os dados do localizador

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.

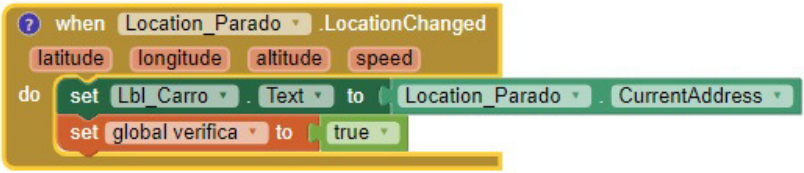

Figura 6.12: Recebendo todos os dados do localizador

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`.

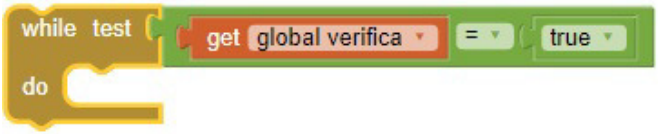

Figura 6.14: Bloco While com o teste de verificação

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.

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`.

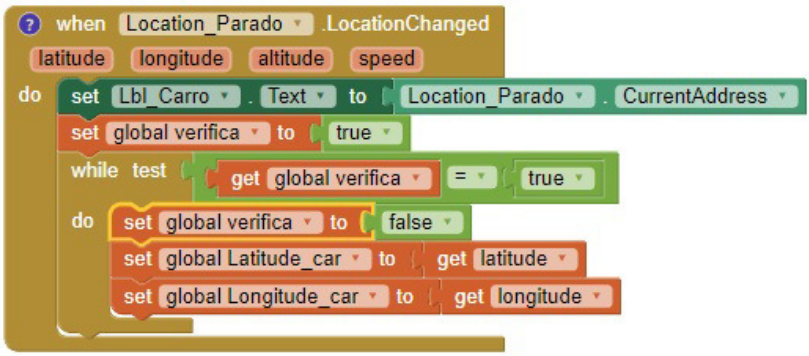

Figura 6.15: Recebendo os dados do localizador

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

`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:

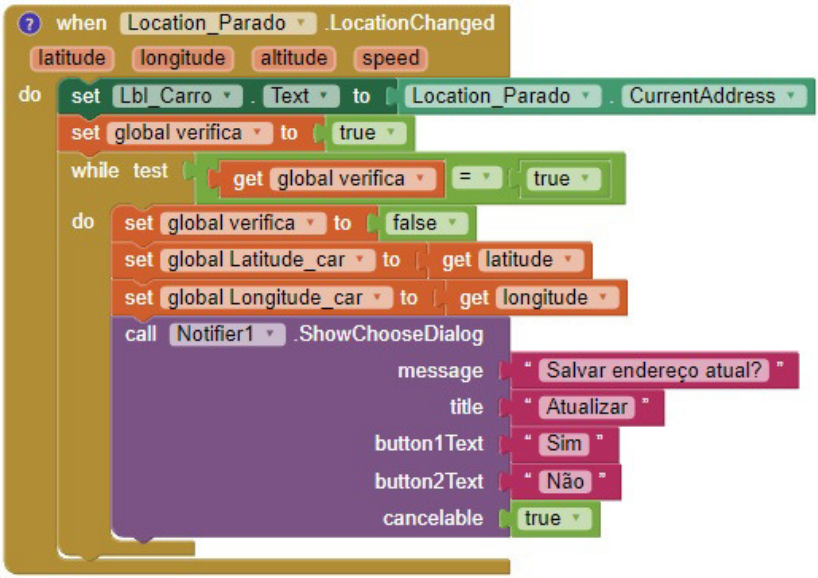

Figura 6.16: Location_Parado finalizado

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`.

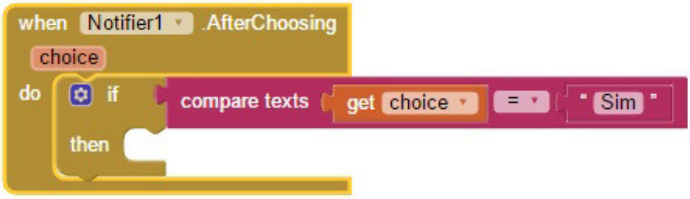

Figura 6.17: Verificando a escolha do usuário

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

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

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.

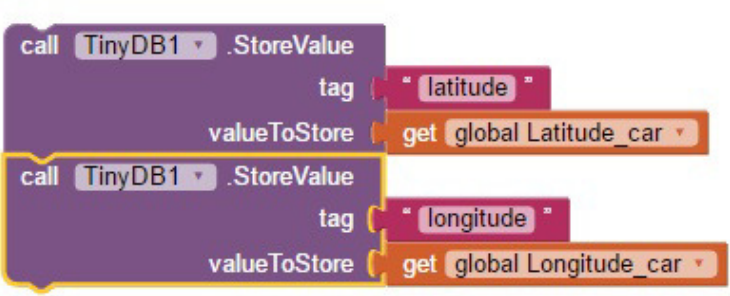

Figura 6.19: Salvando a latitude e a longitude no TinyDB1

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

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`:

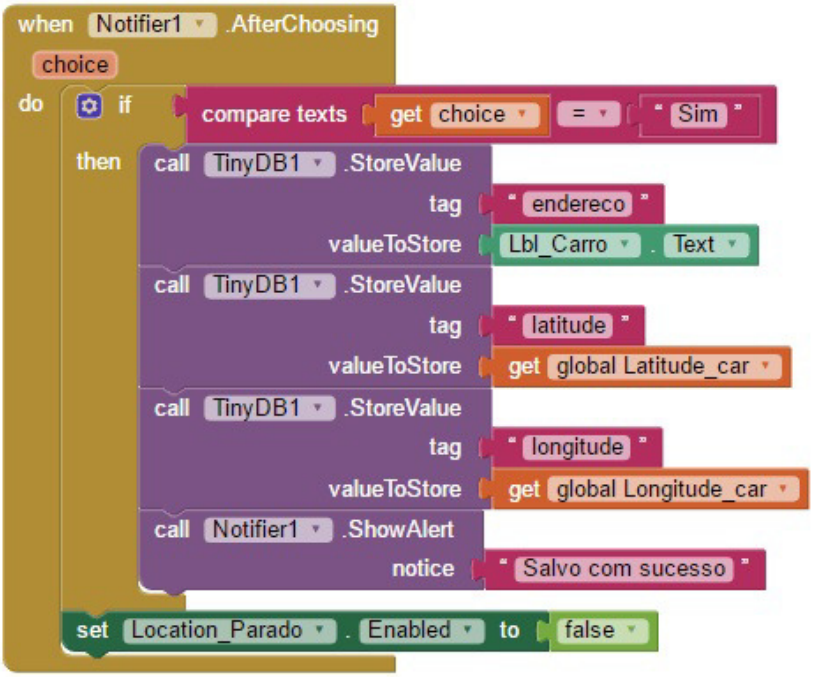

Figura 6.22: Bloco Notifier.AfterChoosing completo

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`.

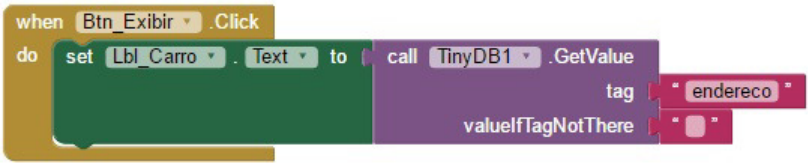

Figura 6.23: Recuperando o endereço do TinyDB1

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

`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` .

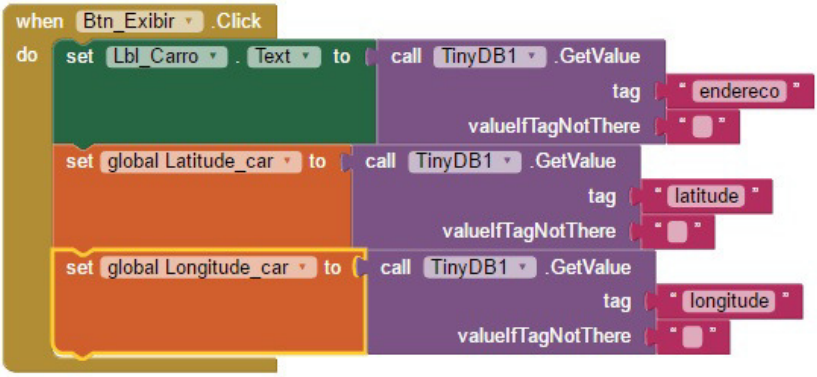

Figura 6.24: Btn_Exibir

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.

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`.

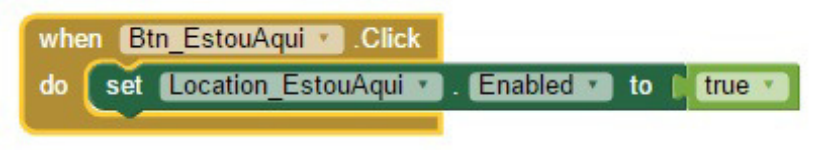

Figura 6.25: Btn_EstouAqui ativando o Location_EstouAqui

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`.

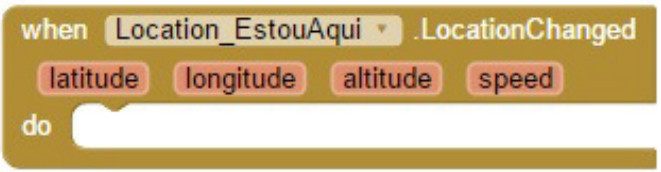

Figura 6.26: Bloco Location_EstouAqui.LocationChanched

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.

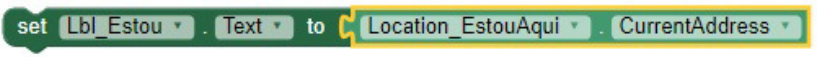

Figura 6.27: Exibindo o endereço atual

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:

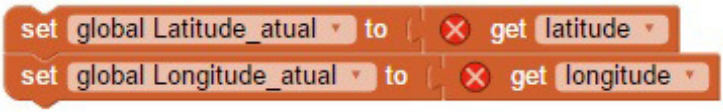

Figura 6.28: Variáveis recebendo os valores da localização atual

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`.

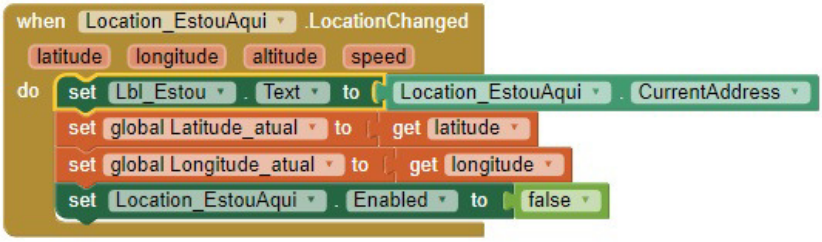

Figura 6.29: Bloco Location_EstouAqui.LocationChanched finalizado

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` .

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=` .

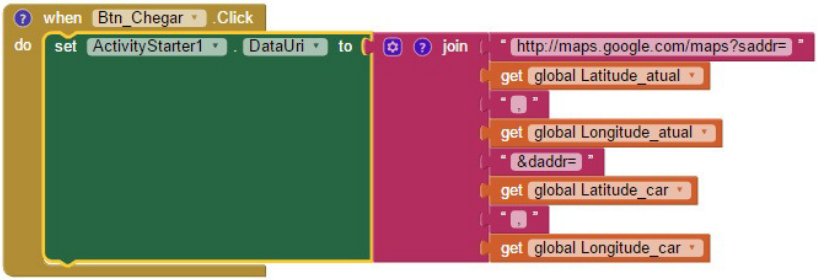

Figura 6.30: Btn_Chegar com o ActivityStarter configurado

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.

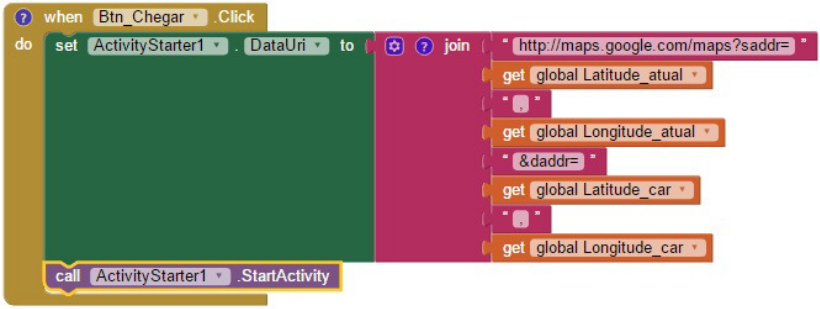

Figura 6.31: Btn_Chegar com todos os códigos

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.

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.

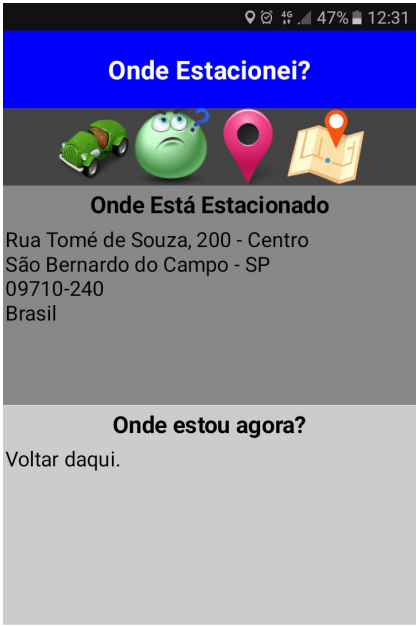

Figura 6.32: Localizando o endereço do estacionamento

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.

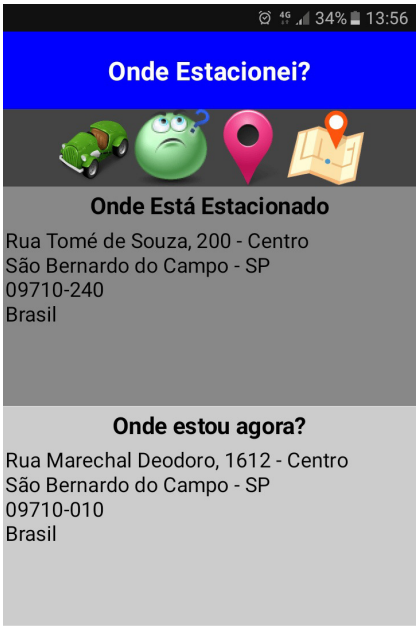

Figura 6.33: Exibindo um local para iniciar o retorno

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.

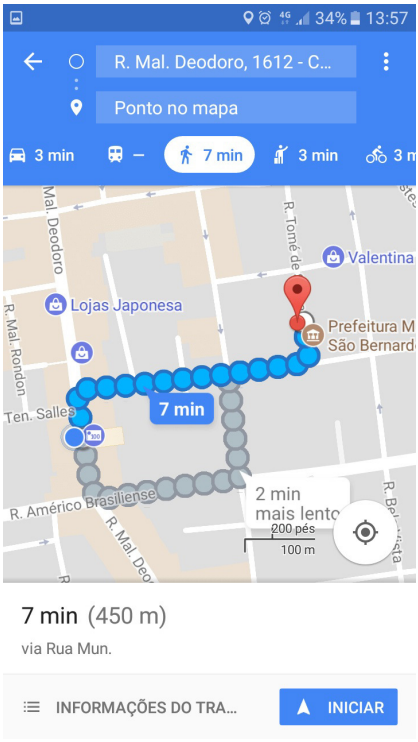

Figura 6.34: Exibindo o mapa com a rota

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