# Desenvolvimento de um Aplicativo Móvel de Avaliação do Transporte Coletivo Urbano

**Davi Pires da Silveira, Romualdo Monteiro de Resende Costa**

Curso de Bacharelado em Sistemas de Informação – Centro de Ensino Superior de Juiz de Fora (CESJF) – Campus Academia
36016-000 – Juiz de Fora– MG– Brasil

{davipsilveira@gmail.com, romualdomrc@gmail.com}

***Abstract:*** *With the popularization of mobile devices and increasing their dependence on everyday activities, they have become a great form of interaction between service providers and users, allowing the reception of data of the consumed product practically in real time. To take advantage of this situation, an application was developed for the Android operating system, which aims to evaluate the quality of the service offered in urban transport. This article describes the features in the application, the steps of its development using the Delphi programming language with the Firemonkey framework and sending the information to Google Firebase.*

***Resumo:*** *Com a popularização dos dispositivos móveis e o aumento da dependência dos mesmos nas atividades cotidianas da população, eles se tornaram uma grande forma de interação entre prestadores de serviço e usuários, permitindo a recepção de dados do produto consumido praticamente em tempo real. Para tirar proveito dessa situação, foi desenvolvido um aplicativo, para o sistema operacional Android, que visa avaliar a qualidade do serviço oferecido no transporte coletivo. Este artigo descreve as funcionalidades presentes no aplicativo, os passos do seu desenvolvimento usando a linguagem de programação Delphi com o framework Firemonkey e o envio das informações para o Google Firebase.*

## 1. Introdução

Atualmente, os dispositivos móveis representam a maior parte dos clientes que consomem serviços e acessam sistemas de informação através da Internet (CISCO, 2017). A partir dessa realidade, é fundamental que os serviços sejam desenvolvidos buscando atender prioritariamente a esses clientes e, não apenas, aos acessos tradicionais realizados a partir dos desktops. Adicionalmente, os dispositivos móveis trazem facilidades que podem tornar as aplicações mais dinâmicas e funcionais, como o uso imediato dos serviços, independente da localização do usuário. Esses dispositivos também, frequentemente, contam com recursos como a possibilidade de registrar eventos através de fotos e de reportar a localização através de geoposicionamento, o que pode contribuir ainda mais para a elaboração de aplicações funcionais.

Existem diversos cenários que podem ser beneficiados pela facilidade de acesso das aplicações através dos dispositivos móveis, bem como pelos recursos adicionais que esses dispositivos oferecem a essas aplicações. Um desses cenários consiste na avaliação

informal de serviços onde, através dos dispositivos móveis, o usuário pode reportar informações sobre o serviço durante a sua utilização. Nesse caso, o prestador de serviço, além de receber o retorno sobre o serviço prestado em um menor intervalo de tempo, pode receber um número maior de informações, em razão da esperada facilidade de postagem que o ambiente móvel oferece e com maior qualidade, caso sejam utilizados os recursos desses dispositivos, como câmeras e geoposicionamento.

Nesse contexto, este trabalho propõe o desenvolvimento de um aplicativo móvel para avaliação do transporte coletivo urbano. Através desse aplicativo, usuários poderiam realizar denúncias, reclamações, solicitações, realizar elogios, fazer sugestões ou, simplesmente, avaliar a experiência de transporte que estão vivenciando. Dessa forma, a empresa responsável pelo serviço poderia contar com informações sobre o serviço realizado e, eventualmente, responder as notificações realizadas pelos usuários que poderiam acompanhar o processamento da sua notificação. Além da empresa prestadora de serviço, outros usuários poderiam ser beneficiados pelas informações submetidas para, eventualmente, tomar decisões sobre a utilização ou não do serviço oferecido baseado nas notificações dos demais usuários.

Adicionalmente, a fim de aproveitar os recursos disponíveis nos equipamentos móveis, a aplicação proposta permite aos usuários registrar fotos das notificações submetidas e registrar a localização do usuário durante a postagem através do geoposicionamento do equipamento. Essa informação sobre a localização permite, inclusive, verificar, ainda que parcialmente, a veracidade de algumas das informações submetidas pois pode associar o usuário ao trajeto percorrido pelo coletivo.

O Capítulo 2 tece maiores comentários sobre o aplicativo desenvolvido com destaque para os seus principais recursos para notificações. Os capítulos seguintes são dedicados aos principais recursos da aplicação que são explorados em detalhes, como a utilização de mapas e recursos de imagens, que são abordados no Capítulo 3. A seguir, o Capítulo 4 trata do armazenamento distribuído das informações, que precisam ser compartilhadas entre os diversos usuários da aplicação. Por fim, o último capítulo trata das conclusões.

# 2. Aplicação Móvel de Avaliação do Transporte Urbano

### 2.1. Ferramenta de Desenvolvimento: Firemonkey

O Firemonkey (TETI, 2014) é um framework de desenvolvimento multi plataforma da Embarcadero Technologies que torna possível usar o mesmo código fonte para gerar um executável nativo para Windows (32 e 64-bits), Android, iOS (32 e 64-bits), OS X (32 e 64-bits) e Linux (aplicação servidor) usando a linguagem de programação Delphi (TETI, 2014) ou C++ (STROUSTRUP, 2014). O aplicativo objeto deste artigo foi desenvolvido usando a linguagem Delphi.

Lançado em 2011 na versão XE2 do RAD Studio[1], IDE[2] focada no desenvolvimento rápido de aplicações, originalmente o Firemonkey só suportava a compilação para as plataformas 32-bits: Windows, iOS e OS X usando a linguagem Delphi. Em 2013, na versão XE5, foi incorporado o desenvolvimento para aplicações Android e, finalmente, em 2014, o desenvolvedor passou a ter a opção de implementar o código usando a linguagem de programação C++.

A versão XE8, lançada em 2015 trouxe como grande novidade o desenvolvimento IoT[3] (Internet of Things), algo que estava se destacando no mercado com o aumento do uso de dispositivos com tecnologias de comunicação como Bluetooth[4], NFC (Near Field Communication)[5], RFID (Radio Frequency Identification)[6], entre outros. No início de 2017, foi disponibilizada a compilação para Linux, mesmo que apenas para aplicações no servidor[7].

Desde então, versões futuras da IDE foram focadas em adicionar os novos recursos que surgiram nos sistemas operacionais suportados. A versão mais recente, o RAD Studio 10.3 Rio, lançado em novembro de 2018, oferece suporte às seguintes versões:

| Sistema Operacional | Versão |
|---|---|
| Android | 5, 6, 7, 8 e 9 |
| iOS | 10, 11 e 12 |
| OS X | macOS Sierra, High Sierra e Mojave |
| Windows | 7, 8.1, 10, Server 2012 e Server 2016 |
| Linux | Ubuntu 14.04 LTS, 16.04 LTs e 18.04 LTS; RedHat Enterprise Linux (versão 7) |

**Tabela 1: Sistemas operacionais que a ferramenta oferece suporte (EMBARCADERO. Platform Status, 2019)**

Em 2018, a Embarcadero Technologies anunciou a versão gratuita para desenvolvimento, utilizando tanto Delphi quanto C++. Até então, essas ferramentas eram

[1] https://www.embarcadero.com/br/products/rad-studio/faq
[2] https://pt.wikipedia.org/wiki/Ambiente_de_desenvolvimento_integrado
[3] https://www.forbes.com/sites/jacobmorgan/2014/05/13/simple-explanation-internet-things-that-anyone-can-understand/#4edb611f1d09
[4] https://www.infowester.com/bluetooth.php
[5] https://www.infowester.com/nfc.php
[6] https://pt.wikipedia.org/wiki/Identifica%C3%A7%C3%A3o_por_radiofrequ%C3%AAncia
[7] http://docwiki.embarcadero.com/RADStudio/Rio/en/Linux_Application_Development

comercializadas em versões paga. A versão gratuita, conhecida como *Community Edition* (EMBARCADERO. Delphi: Community Edition, 2019) possui todos os recursos para desenvolvimento de aplicativos Android e pode ser usada gratuitamente por estudantes, empresas com até 5 desenvolvedores ou receita máxima de US$ 5.000,00 anuais, o que facilitou bastante a utilização dessa ferramenta como solução de desenvolvimento a ser adotada, inclusive, em cursos relacionados à computação.

Entre os recursos do Firemonkey, que podem ser encontrados, inclusive na sua versão gratuita, merecem ser destacados o fato de que a aplicação final consiste em um código binário nativo da plataforma escolhida, o que elimina a necessidade de instalação de qualquer outra biblioteca para o seu funcionamento. Nessas aplicações, a interface oferece recursos avançados como a criação de animações 2D e 3D, que serão processadas pela GPU (ANDROID AUTHORITY, 2016), se disponível, diminuindo assim o processamento do processador e aumentando a fluidez das animações. Nas aplicações as interfaces podem ser do tipo drag and drop e os controles visuais existentes respeitam a identidade visual padrão de cada plataforma, sendo possível alterar a plataforma destino durante o desenvolvimento e verificar o resultado final sem a necessidade de executar a aplicação em um dispositivo emulador. Também é possível criar leiautes específicos para cada tamanho diferente de tela. Existe a opção de configurar, via interface, as permissões de acesso, ícones do aplicativo e animações de inicialização.

### 2.2. Funcionalidades do aplicativo

A tela inicial do aplicativo exibe a lista com as ocorrências previamente registradas por todos os usuários, agrupadas pelo tipo. A descrição é composta pela data em que o usuário registrou, o evento, a linha e o título da ocorrência. Logo acima da lista, é possível pesquisar algum registro específico, informando os dados desejados no campo pesquisar.

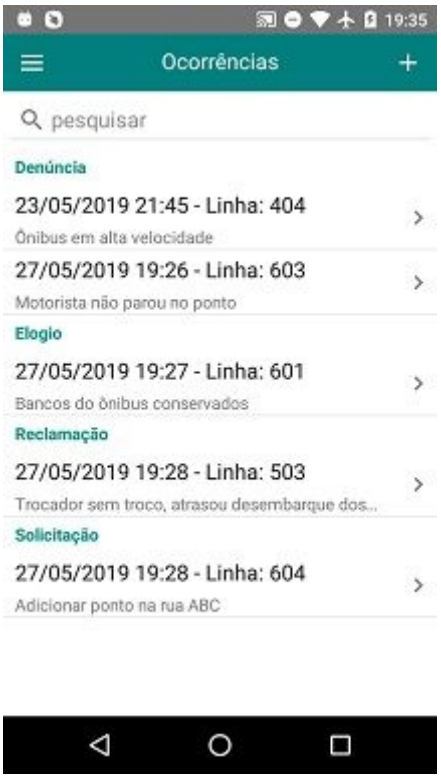


**Figura 1: Tela inicial do aplicativo (do autor).**

O botão localizado no canto superior esquerdo da tela, permite que o usuário faça um filtro nos dados exibidos na lista de ocorrências da tela principal. São três opções:

| Tipo de Filtro | Opções |
|---|---|
| Linha | Todas ou<br>uma em específico |
| Data | Todas ou<br>registradas hoje ou<br>registradas nos últimos 7 dias ou<br>registradas nos últimos 30 dias |
| Tipo | Todos ou<br>um em específico: denúncia, elogio, reclamação ou solicitação |

**Tabela 2: Tipos de filtro disponíveis ao acessar menu da tela principal.**

Os filtros são combinados e podem ser aplicados clicando em ‘Filtrar’. Ao aplicar o filtro de exemplo da Figura 2, serão exibidas as ocorrências de todas as linhas (A), que foram registradas nos últimos 30 dias (B) de todos os tipos (C).

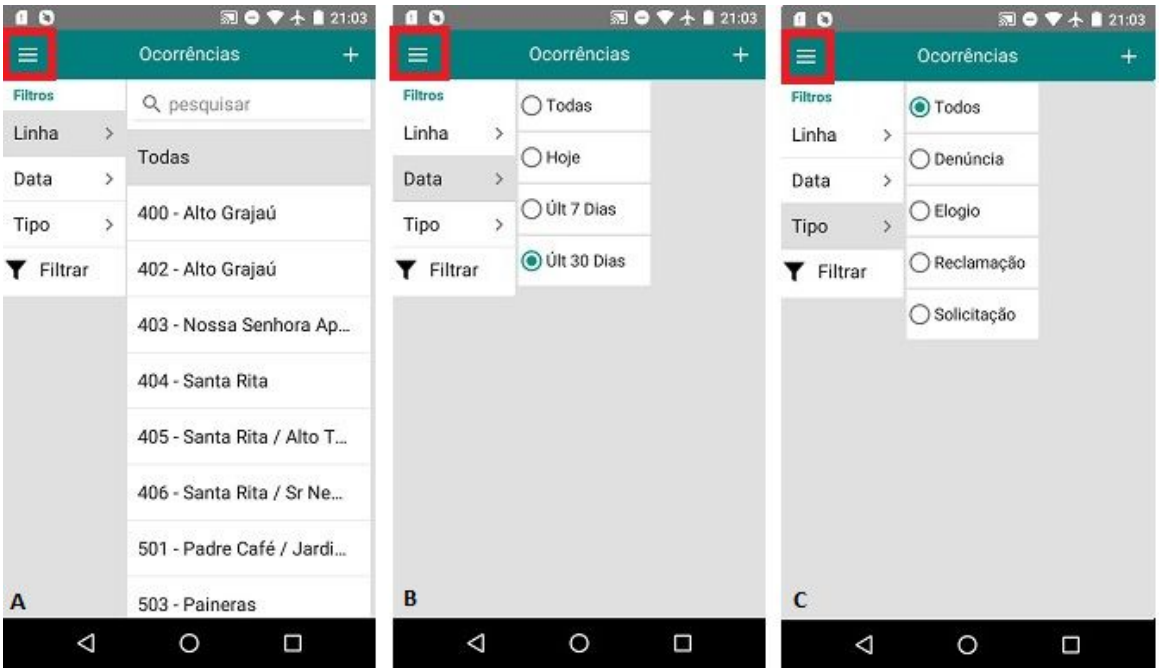


**Figura 2: Exemplo das opções de filtro (do autor).**

Ao clicar no botão para registrar uma nova ocorrência, localizado no canto superior direito, serão exibidos os tipos de ocorrência disponíveis: Denúncia, Elogio, Reclamação ou Solicitação. O usuário deve selecionar o tipo que mais se adequa ao seu relato. A escolha correta do tipo é importante, pois ele é usado para categorizar a informação. A Figura 3 apresenta as imagens das opções disponíveis. O usuário escolhe o tipo desejado através da seleção de uma dessas imagens.

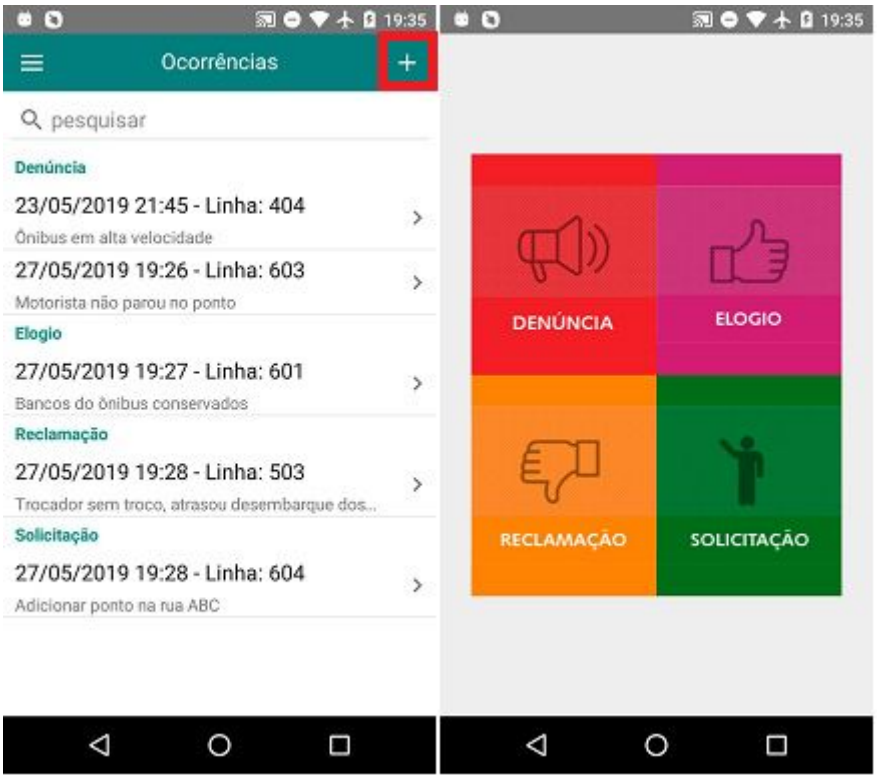


**Figura 3: Cadastro de nova ocorrência e seleção do tipo (do autor).**

Após a seleção do tipo da ocorrência, inicia-se o processo de preenchimento das informações obrigatórias de registro. A Figura 4 (A) apresenta os elementos constantes deste cadastro, os mesmos para todos os tipos de ocorrência. Nessa figura, a aba "DADOS" possui os campos relacionados à data, ao título escolhido pelo usuário e da linha correspondente ao objeto da ocorrência. Por padrão, e para facilitar a inserção dos dados, os campos referentes a data e a hora assumem o valor do horário configurado no dispositivo. O título deve ser preenchido com um breve resumo do fato. A linha será selecionada entre uma opção de valores já cadastrados no aplicativo, bastando clicar no item “Linha” para que seja exibida uma tela de pesquisa das linhas, conforme exemplificado na Figura 4 (B).

O usuário pode navegar entre as três abas livremente. Ao finalizar o preenchimento de todas as informações, a ocorrência deve ser salva clicando no botão localizado no canto superior direito. Uma vez salva, a ocorrência só poderá ser edita pelo usuário que a cadastrou.

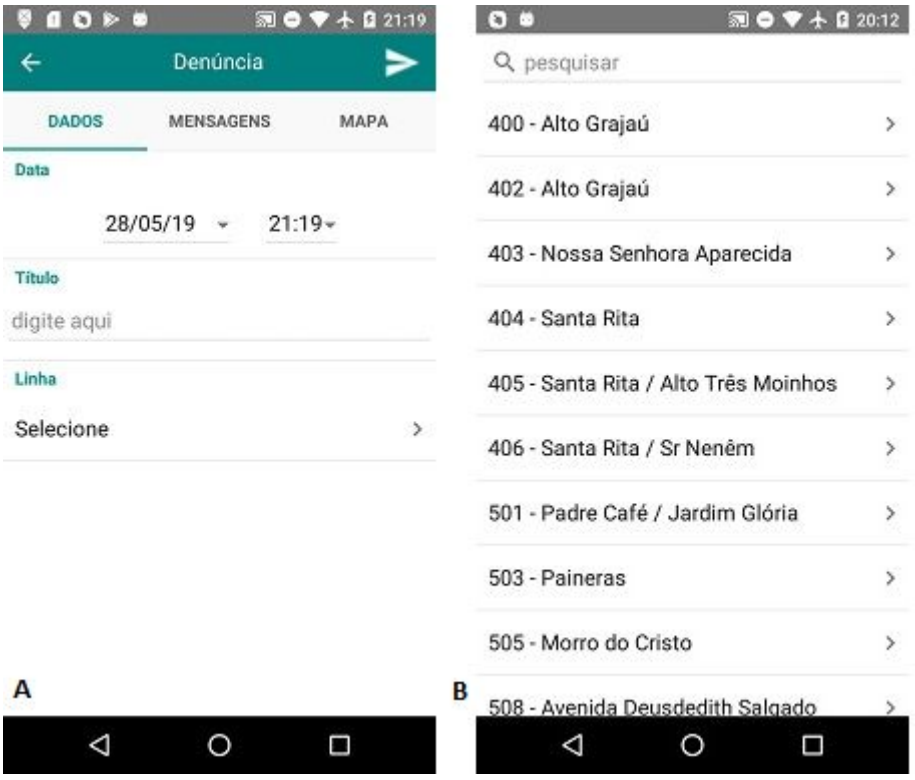


**Figura 4: Aba dados e tela para seleção da linha (do autor).**

# 3. Recursos de Imagem e Utilização de Mapas

### 3.1. Recursos de Imagem

Para adicionar uma imagem e/ou mensagem de texto, o usuário deve navegar até a aba "MENSAGENS", conforme apresentado na Figura 5. Ao clicar no ícone da câmera fotográfica, no canto inferior esquerdo, são exibidas duas opções para a seleção da imagem: tirar uma nova foto ou selecionar uma que já esteja salva no álbum de fotos do dispositivo. As imagens são adicionadas na ordem cronológica e, junto com as mensagens, auxiliam na descrição e comprovação dos fatos.

O aplicativo possui o recurso de aumentar o tamanho da imagem para facilitar a visualização, basta clicar sobre a imagem para que ela seja ampliada. Clicando novamente na imagem, ele retorna para sua posição original. Recurso exemplificado na Figura 6.


**Figura 5: Opções para inserir uma imagem (do autor).**

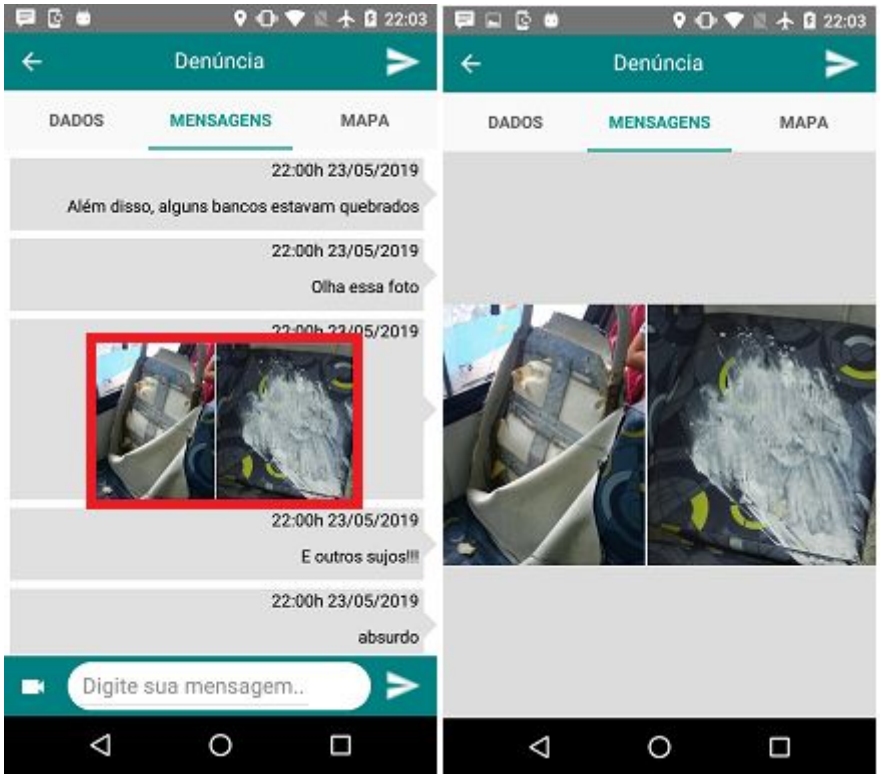


**Figura 6: Opção de zoom na imagem (do autor).**

### 3.2.1. Configurações na ferramenta de desenvolvimento

Algumas configurações foram realizadas pela IDE para que a aplicação possa acessar a câmera do dispositivo, salvar as novas fotos registradas e acessar o álbum. Essas configurações são refletidas diretamente no arquivo conhecido como Manifesto na plataforma Android (*AndroidManifest.xml*). Esse é o arquivo que contém as informações necessárias para que o Android possa executar o código do aplicativo.

A configuração é realizada na IDE acessando o menu 'Project/Options', item 'Application', subitem 'Uses Permissions' e marcando as opções: 'Camera' para acesso a câmera, 'Read external storage' para ler as fotos salvas no dispositivo e 'Write external storage' para permitir salvar novas fotos no aparelho, conforme apresentado na Figura 7.

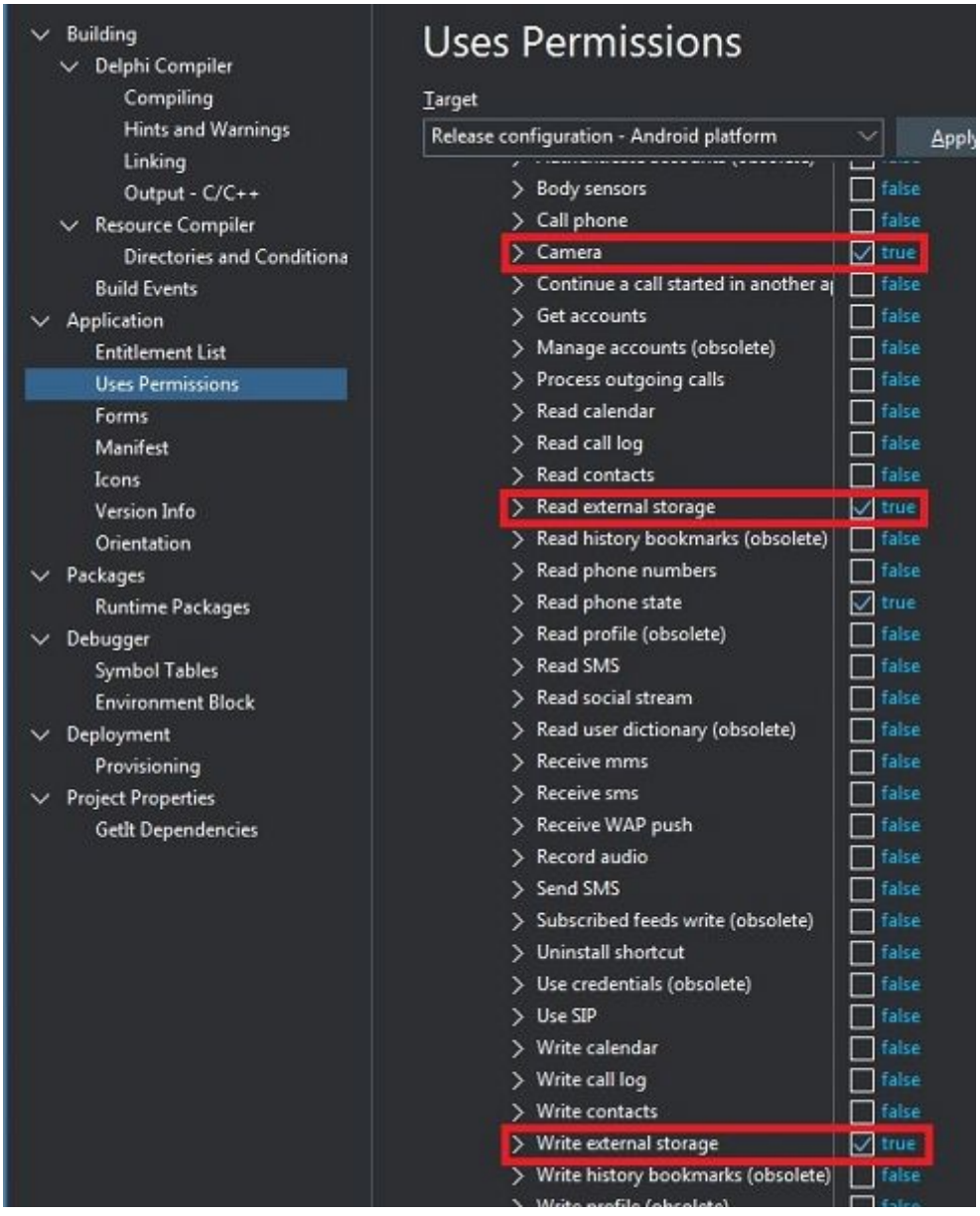


**Figura 7: Permissões para acesso a câmera (do autor).**

O componente TActionList[8] é usado para acessar a câmera e o álbum de fotos do dispositivo. Ele é a forma mais simples de interagir com a câmera, sendo necessárias poucas linhas de código para exibir na tela a foto tirada ou selecionada pelo usuário. Para isso, foram criadas duas ações da categoria 'Media Library' no TActionList, conforme Tabela 3. Ao finalizar a execução de cada uma das ações, o evento OnDidFinishTaking[9], configurado para cada ação, é disparado. Ele possui como parâmetro um objeto do tipo TBitmap[10] contento a foto que será adicionada na aba de "MENSAGENS".

| Nome da Ação | Definição | Fim de execução |
|---|---|---|
| TTakePhotoFromCameraAction | abre a câmera para ser registrada uma nova foto | Confirmar a foto registrada |
| TTakePhotoFromLibraryAction | abre álbum para seleção de uma imagem salva no dispositivo | Confirmar a foto selecionada no álbum |

**Tabela 3: Ações disponíveis do TActionList**

### 3.2. Mapa e Geoposicionamento

Por ser um aplicativo para avaliação de um meio de transporte, o local em que a ocorrência sucedeu é de extrema importância. Com os dados extraídos do aplicativo, seria possível detectar as áreas da cidade com maiores índices de ocorrências ou ainda fazer uma análise das informações, visando, por exemplo, a tomada de decisões para prevenção de problemas futuros.

Para selecionar o local da ocorrência, o usuário pode navegar até a aba "MAPA". Através do sistema de geoposicionamento, um marcador vermelho será automaticamente posicionado na localização do usuário no momento de registro da ocorrência. Essa posição inicial, apresentada na Figura 8, pode ser alterada pelo usuário, visto que o registro não precisa ser, obrigatoriamente, efetuado na mesma localização que o fato ocorreu. Para alterar a posição da ocorrência, basta clicar no local desejado dentro do mapa que o marcador vermelho será movido para o novo local. O marcador azul exibe o local atual do dispositivo, lido do próprio serviço de mapa da plataforma, no caso, do Google.

[8] http://docwiki.embarcadero.com/Libraries/Rio/en/FMX.ActnList.TActionList
[9] http://docwiki.embarcadero.com/Libraries/Rio/en/FMX.MediaLibrary.TOnDidFinishTaking
[10] http://docwiki.embarcadero.com/Libraries/Rio/en/FMX.Graphics.TBitmap


**Figura 8: Aba Mapa: dados de geoposionamento do usuário (do autor).**

### 3.1.1. Configurações na ferramenta de desenvolvimento

Para a execução da aplicação, especificamente, é necessário conceder permissão ao aplicativo para acessar os dados de posicionamento, lidos através do sensor GPS (Global Position System)[11] do dispositivo. A configuração foi realizada na IDE acessando o menu 'Project/Options', item 'Application', subitem 'Uses Permissions' e selecionando as opções 'Access coarse location' e 'Access fine location', conforme apresentado na Figura 9.

A exibição do mapa é encapsulada pelo componente TMapView[12], que exibe a biblioteca de mapa padrão do sistema operacional, no caso do Google, o Google Maps. Dentro desse componente é possível configurar o que será exibido no mapa: botões de zoom, bússola e minha localização. Outra alteração importante na navegação é a inclusão dos pontos de interesse, prédios, localização atual e tráfego. Todas as configurações mencionadas são feitas apenas ativando as opções desejadas nas propriedades visuais do componente, conforme detalhado na Figura 10.

[11] https://pt.wikipedia.org/wiki/Sistema_de_posicionamento_global
[12] http://docwiki.embarcadero.com/Libraries/Rio/en/FMX.Maps.TMapView

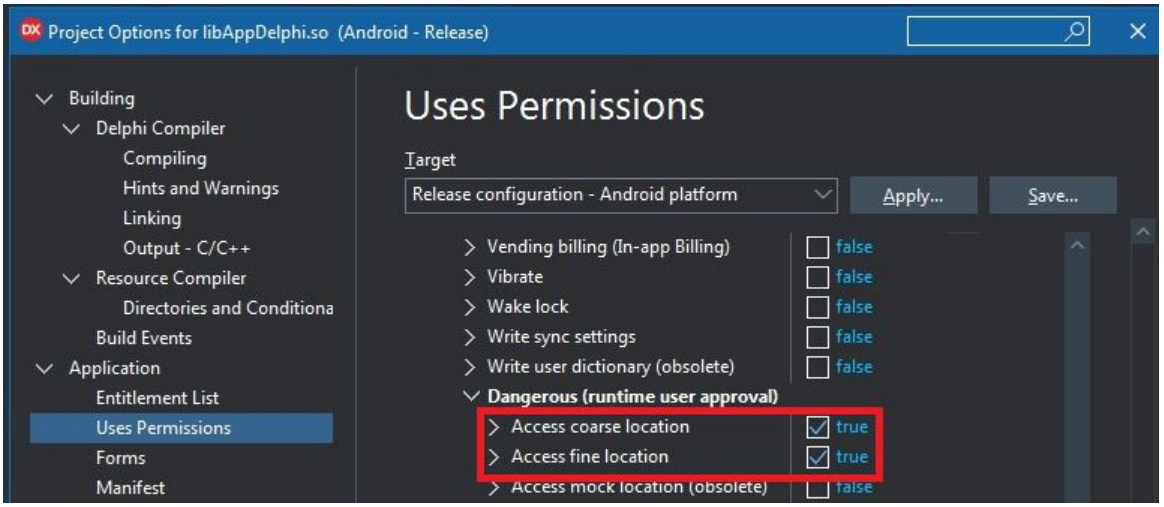


**Figura 9: Configuração da solicitação de permissão de acesso à localização do dispositivo (do autor).**

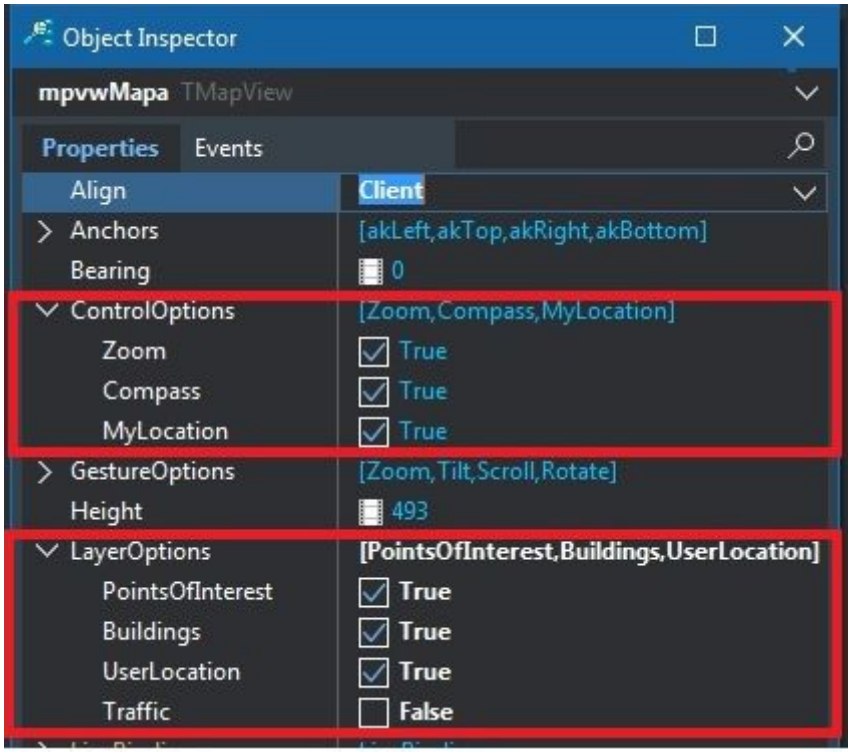


**Figura 10: Configuração dos itens a serem exibidos no mapa através do componente TMapView (do autor).**

Outras duas configurações são necessárias para o funcionamento do mapa dentro do aplicativo. A primeira, apresentado na Figura 11, consiste na ativação do uso do serviço de mapas do Android. Para isso, dentro da IDE deve ser acessado o menu ‘Project/Options’, item ‘Application’, subitem ‘Entitlement List’, opção ‘Maps Service’.

O segundo passo necessário é a configuração da chave de API[13] para uso do Google Map. A Embarcadero oferece uma documentação detalhada de como realizar esse processo[14].

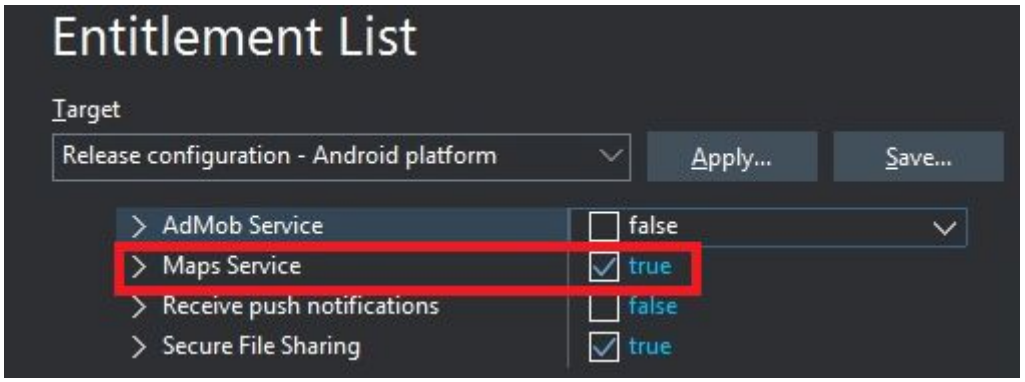


**Figura 11: Configuração de acesso a localização pelo serviço de mapa da plataforma (do autor).**

## 4. Armazenamento Distribuído das Informações

O armazenamento distribuído das informações permite que as ocorrências sejam acessadas de qualquer lugar e a qualquer momento pelas empresas de transporte. As informações podem ser armazenadas em vários computadores interligados via Internet ao redor do mundo, garantindo o backup dos dados e a alta disponibilidade do sistema.

Todas as ocorrências registradas no aplicativo são armazenadas no próprio dispositivo e enviadas, posteriormente, para um servidor. O usuário não necessita de conexão com a Internet no momento de registro da ocorrência, pois o envio é realizado logo que o aplicativo detectar que o dispositivo possui conexão com a Internet. Desta forma, o funcionamento do aplicativo não fica dependendo de conexão com rede, seja ela sem fio, ou através do sinal da operadora de telefone móvel.

Foi implementado o envio e recepção de todas as informações salvas no banco SQLite[15] da aplicação para o Firebase Realtime Database[16], serviço gratuito da Google que disponibiliza um banco de dados NoSQL (NOSQL, 2019) para armazenamento em tempo real das informações geradas no aplicativo. Basta ter um email do Gmail[17] para criar seu projeto e iniciar o uso.

---

[13] https://pt.wikipedia.org/wiki/Interface_de_programa%C3%A7%C3%A3o_de_aplica%C3%A7%C3%B5es

[14] http://docwiki.embarcadero.com/RADStudio/Rio/en/Configuring_Android_Applications_to_Use_Google_Maps

[15] https://www.sqlite.org/index.html

[16] https://firebase.google.com/?hl=pt-BR

[17] https://www.gmail.com

A comunicação com o Firebase é realizada através de um cliente HTTPS (Hyper Text Transfer Protocol Secure) (GOOGLE, 2019). A autenticação é opcional e pode ser configurada nos parâmetros “auth” e “uid” do cabeçalho da requisição. Os dados são trafegados no formato JSON (JavaScript Object Notation) (JSON, 2019), conforme exibido na Figura 12.

O JSON salvo no Firebase contém uma lista de ocorrências e cada ocorrência é identificada pelo IMEI (ANATEL, 2019) do dispositivo em que foi registrada, concatenado de um identificador único gerado por cada dispositivo para cada registro (destacado na figura). O item possui um par de chave/valor dentro do JSON para cada informação preenchida no aplicativo: data, latitude, longitude, linha, código do tipo e título. As mensagens são armazenadas em uma lista de objetos e cada mensagem possui um par com chave/valor: identificador (id), foto (codificada em Base64[18]), data e texto da mensagem.

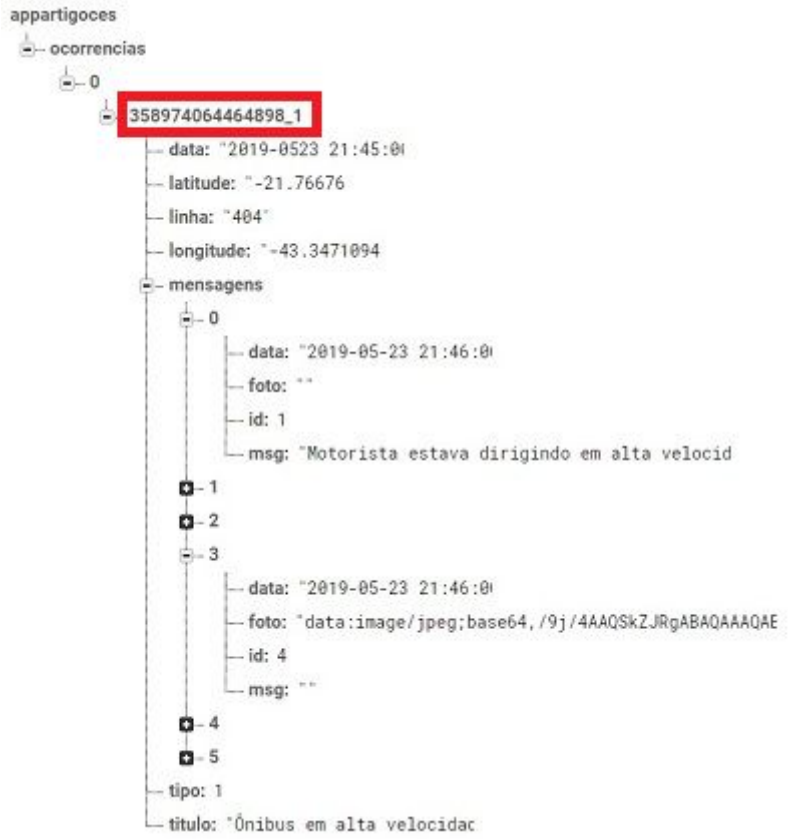


**Figura 12: Formato dos dados salvos no Firebase (do autor).**

A recepção dos dados ocorre ao iniciar o app e pode ser solicitado manualmente na tela principal, botão localizado no canto superior esquerdo, opção “Atualizar”. O dispositivo faz download de todos as ocorrências e as salva no banco local. A Figura 13 mostra a função criada para leitura das informações salvas no Firebase. Ela instancia um

[18] https://pt.wikipedia.org/wiki/Base64

objeto para fazer a chamada HTTPS, o vIdHTTP, configura os dados de autenticação no cabeçalho da requisição e configura o tipo do dado de retorno, no caso JSON. Faz a leitura dos dados usando o método GET (MOZILLA, 2019).

O envio ocorre logo após o usuário finalizar o cadastro da ocorrência. A Figura 14 mostra a função de envio, semelhante ao da recepção. A diferença fica no método da requisição usado, no caso, o PUT (MOZILLA, 2019). Um array de bytes com os dados do JSON da ocorrência a ser salva é passado como parâmetro no método.

```
function TThreadFirebaseObj.LerDados(const pURl, pAuth, pUID: String): String;
var
  vIdHTTP: THTTPClient;
  vResposta: IHTTPResponse;
begin
  Result := '';

  vIdHTTP     := nil;
  try
    vIdHTTP                       := THTTPClient.Create;
    //dados de autentificação configurados no console do Firebase
    vIdHTTP.CustomHeaders['auth'] := pAuth;
    vIdHTTP.CustomHeaders['uid']  := pUID;
    vIdHTTP.ContentType           := 'application/json';

    try
      vResposta := vIdHTTP.Get(pURl);
      Result    := vResposta.ContentAsString;
    except
      on e: Exception do
        LogMsg('TThreadFirebaseObj.LerDados', e);
    end;
  finally
    FreeAndNil(vIdHTTP);
  end;
end;
```

**Figura 13: Leitura dos dados salvos no Firebase (do autor).**

```
function TThreadFirebaseObj.EnviarDados(const pURl, pAuth, pUID, pJsonGravar: String): Boolean;
var
  vJsonStream: TStringStream;
  vIdHTTP: THTTPClient;
begin
  Result := False;

  vIdHTTP     := nil;
  vJsonStream := nil;
  try
    //JSON com as informações da ocorrência a ser gravado no Firebase
    vJsonStream := TStringStream.Create(UTF8Encode(pJsonGravar));

    vIdHTTP                       := THTTPClient.Create;
    //dados de autentificação configurados no console do Firebase
    vIdHTTP.CustomHeaders['auth'] := pAuth;
    vIdHTTP.CustomHeaders['uid']  := pUID;
    vIdHTTP.ContentType           := 'application/json';

    try
      vIdHTTP.Put(pURl, vJsonStream);
      Result := True;
    except
      on e: Exception do
        LogMsg('TThreadFirebaseObj.EnviarDados', e);
    end;
  finally
    FreeAndNil(vJsonStream);
    FreeAndNil(vIdHTTP);
  end;
end;
```

**Figura 14: Envio de dados para o Firebase (do autor).**

## 5. Conclusões

O desenvolvimento do aplicativo possibilitou uma avaliação da ferramenta escolhida. Apesar do Delphi não ser uma linguagem muito utilizada nos últimos anos, algo que pode mudar com o lançamento da versão gratuita da IDE, ela se apresentou como uma boa opção para o desenvolvimento rápido de aplicativos Android, principalmente para quem já domina a linguagem.

A criação das telas ficam bem simples com o recurso de arrastar e soltar componentes e com a opção de criar um leiaute específico para cada tamanho de tela do dispositivo, seja no modo retrato ou paisagem. O uso de componentes visuais torna possível a configuração do mapa, câmera e sensor GPS com quase nenhuma linha de código. Outro ponto positivo é a interface para alteração das configurações no Manifesto, eliminando a edição manual do arquivo XML. O ponto negativo da ferramenta fica na parte da documentação, em que muitas vezes, componentes e métodos não possuem qualquer descrição de seu funcionamento.

O aplicativo para avaliação do transporte coletivo urbano foi desenvolvido com uma interface simples para atender a todos os públicos e tipos de dispositivos. Possui o intuito de centralizar as informações para que seja possível melhorar a qualidade do serviço baseado em ocorrências reais, relatadas pelos próprios usuários. O fato de ser um aplicativo para dispositivo móvel facilita o lançamento das informações, que pode ser

realizado durante o uso do transporte, e se beneficia dos seus recursos, como registro de fotos e localização. O código fonte do aplicativo foi disponibilizado gratuitamente na plataforma GitHub, podendo ser acessado no endereço https://github.com/artigoces/AvaliacaoTransporteColetivo.

## 6. Referências


CISCO. **Cisco visual networking index: Global mobile data traffic forecast update, 2017-2022 White Paper**. Disponível em https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-738429.html. Acesso em Maio 2019

TETI, Daniele. **Delphi Cookbook: 50 hands-on recipes to master the power of Delphi for cross-platform and mobile development on Windows, Mac OS X, Android, and iOS**. Packt Publishing, 2014

STROUSTRUP, Bjarne. **Programming: Principles and Practice Using C++**. 2 ed. Editora Addison-Wesley Professional, 2014

EMBARCADERO. **Platform Status**. Disponível em http://docwiki.embarcadero.com/PlatformStatus/en/Main_Page. Acesso Março de 2019

EMBARCADERO. **Delphi: Community Edition**. Disponível em https://www.embarcadero.com/products/delphi/starter. Acesso Março de 2019

JSON. **Introducing JSON**. Disponível em: http://json.org/. Acesso em Junho de 2019

NOSQL. **O que é NoSQL?** Disponível em: https://aws.amazon.com/pt/nosql/. Acesso em Junho de 2019

ANDROID AUTHORITY. **What is a GPU and how does it work?.** Disponível em https://www.androidauthority.com/what-is-a-gpu-gary-explains-693542/. Acesso em Junho de 2019

GOOGLE. **Por que usar HTTPS?** Disponível em https://developers.google.com/web/fundamentals/security/encrypt-in-transit/why-https?hl=pt-br. Acesso em Junho de 2019

ANATEL. **Identificação Internacional de Equipamento Móvel**. Disponível em http://www.anatel.gov.br/celularlegal/imei. Acesso em Junho de 2019

MOZILLA. **Métodos de Requisição HTTP**. Disponível em https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Methods. Acesso em Junho de 2019