UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
COTSI - TECNOLOGIA EM SISTEMAS PARA INTERNET
CURSO DE TECNOLOGIA EM SISTEMAS PARA INTERNET

WESLEY OZEBE DOS SANTOS

UM APLICATIVO MÓVEL PARA AUXILIAR NO COMBATE Á DENGUE

TRABALHO DE CONCLUSÃO DE CURSO

TOLEDO
2021

WESLEY OZEBE DOS SANTOS

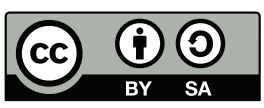
UM APLICATIVO MÓVEL PARA AUXILIAR NO COMBATE Á DENGUE

A Mobile Application to Help Fight Dengue

Trabalho de Conclusão de Curso apresentado ao Curso de Tecnologia em Sistemas para Internet da Universidade Tecnológica Federal do Paraná, como requisito parcial para a obtenção do título de Tecnólogo.

Orientador: Prof. (Me.) Marcelo Alexandre da Cruz Ismael

TOLEDO
2021



WESLEY OZEBE DOS SANTOS

UM APLICATIVO MÓVEL PARA AUXILIAR NO COMBATE À DENGUE

Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Tecnológo em Sistemas para Internet da Universidade Tecnológica Federal do Paraná (UTFPR).

Data de aprovação: 01/dezembro/2021

_____

Marcelo Alexandre da Cruz Ismael
Mestre
Universidade Tecnológica Federal do Paraná

_____

Fabio Alexandre Spanhol
Doutor
Universidade Tecnológica Federal do Paraná

_____

Edson Tavares de Camargo
Doutor
Universidade Tecnológica Federal do Paraná

TOLEDO

2021

Aos meus pais, Valdir e Thatiane, que sempre me apoiaram e me incentivaram. Também aos meus amigos que me ajudaram nos meus momentos difíceis.

AGRADECIMENTOS

Expresso meus sinceros agradecimentos primeiramente a Deus, pelo dom da vida e por ter me ajudado a chegar até aqui. Aos meus familiares e amigos que me apoiaram nos momentos difíceis. Agradeço também aos professores, seres iluminados, que conseguiram transmitir conhecimento e sabedoria através de suas aulas.

E sucedeu que, terminados os dias de seu ministério, voltou para sua casa.(Lucas 1:23).

RESUMO

OZEBE, Wesley. UM APLICATIVO MÓVEL PARA AUXILIAR NO COMBATE Á DENGUE. 2021. 71 f. Trabalho de Conclusão de Curso – Curso de Tecnologia em Sistemas para Internet, Universidade Tecnológica Federal do Paraná. Toledo, 2021.

Com o advento do aumento dos casos de dengue no Brasil, em especial na cidade de Toledo-PR, a rápida e prática comunicação entre o cidadão e os agentes de combates a endemias da prefeitura municipal, ajuda a diminuir os focos de dengue. Dessa forma, o objetivo central do trabalho é o desenvolvimento de um aplicativo que utiliza uma API de geolocalização para auxiliar no combate a dengue, agilizando a identificação de possíveis focos, permitindo uma maior participação da sociedade no combate ao mosquito trasmissor da dengue. O desenvolvimento do aplicativo móvel foi realizado utilizando-se do Android Studio como IDE, em conjunto de APIs como: Fused Location Provider, Maps SDK e Firebase. Tendo como resultado final um aplicativo android onde é possível realizar a denúncia e acompanhamento de focos de dengue por parte do usuário, inclusão de visitas a focos de dengue e visitas a residências por parte do agente de combate a endemias e a extração de relatórios e manutenção do sistema por parte do administrador.

Palavras-chave: API. Geolocalização. Android. Combate a dengue. *Frameworks*.

ABSTRACT

OZEBE, Wesley. A MOBILE APP TO ASSIST IN THE FIGHT AGAINST DENGUE. 2021. 71 f. Trabalho de Conclusão de Curso – Curso de Tecnologia em Sistemas para Internet, Universidade Tecnológica Federal do Paraná. Toledo, 2021.

With the advent of the increase in dengue cases in Brazil, especially in the city of Toledo-PR, the rapid and practical communication between the citizen and the municipality agents for endemics, helps to reduce the outbreaks of dengue. Thus, the central objective of work is the development of an application that uses a geolocation API to help combat dengue, speeding up the identification of possible outbreaks, allowing a greater participation of society in the fight against the dengue transmitting mosquito. The development of the application was carried out using Android Studio as an IDE, together with APIs such as: Fused Location Provider, Maps SDK and Firebase. Having as final result an android application where it is possible to perform the denunciation and monitoring of dengue outbreaks by the user, inclusion of visits to dengue outbreaks and visits to homes by agents for endemics and the extraction of reports and maintenance of the system by the administrator.

Keywords: API. Geolocation. Android. Fight against dengue. *Frameworks*.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

API	*Application Programmin Interface*
JS	JavaScript
JSON	JavaScript Object Notation
DBA	*Database Administrator*
DDOS	*Distributed Denial-of-Service*
SDK	*Software Development Kit*
IDE	*Integrated Development Environment*

SUMÁRIO

1 INTRODUÇÃO

A dengue continua sendo um problema para o Brasil, principalmente porque aqui o mosquito transmissor encontra condições extremamente favoráveis para sua proliferação. Dentre tais condições pode-se citar: temperaturas entre 20 e 40 graus celsius na maior parte do território, problemas de saneamento básico e também um crescimento urbano desordenado (VEIGA, 2019). No ano de 2020 foram registrados mais de 900 mil casos, com mais de 500 mortes (VENAGLIA, 2020). No Paraná, tivemos mais de 250 mil casos (SAÚDE, 2021), já em Toledo, tivemos mais de cinco mil casos notificados até a vigésima oitava semana epidemiológica de 2020, representando mais de dois porcento do total estadual (SAÚDE, 2020).

Atualmente o controle dos focos de dengue é realizado com ações preventivas, como vistorias nas casas e quintais, conscientização através de panfletos, vídeos e notícias(TOLEDO, 2020b)(TOLEDO, 2020a).

O desafio que encontramos ao lidar com a dengue não é somente de cunho sanitário, mas também educacional, já que para que ocorra a reciclagem e o descarte correto de lixo, como plásticos e outros, a população precisa ser conscientizada do perigo e do que pode ser realizado para combater a dengue, pois com o acúmulo desse tipo de resíduo acaba-se por armazenar água das chuvas, tornando-se criadouros do mosquito transmissor da dengue. Desta forma esse trabalho visa desenvolver um aplicativo que permita reportar os focos de dengue, e também realize o controle de casas vistoriadas relizada pelos agentes de combate a endemias, sendo utilizado tanto pelo cidadão quanto pelos agentes de combate a endemias. Com o objetivo de diminuir a proliferação do mosquito da Dengue.

1.1 JUSTIFICATIVA

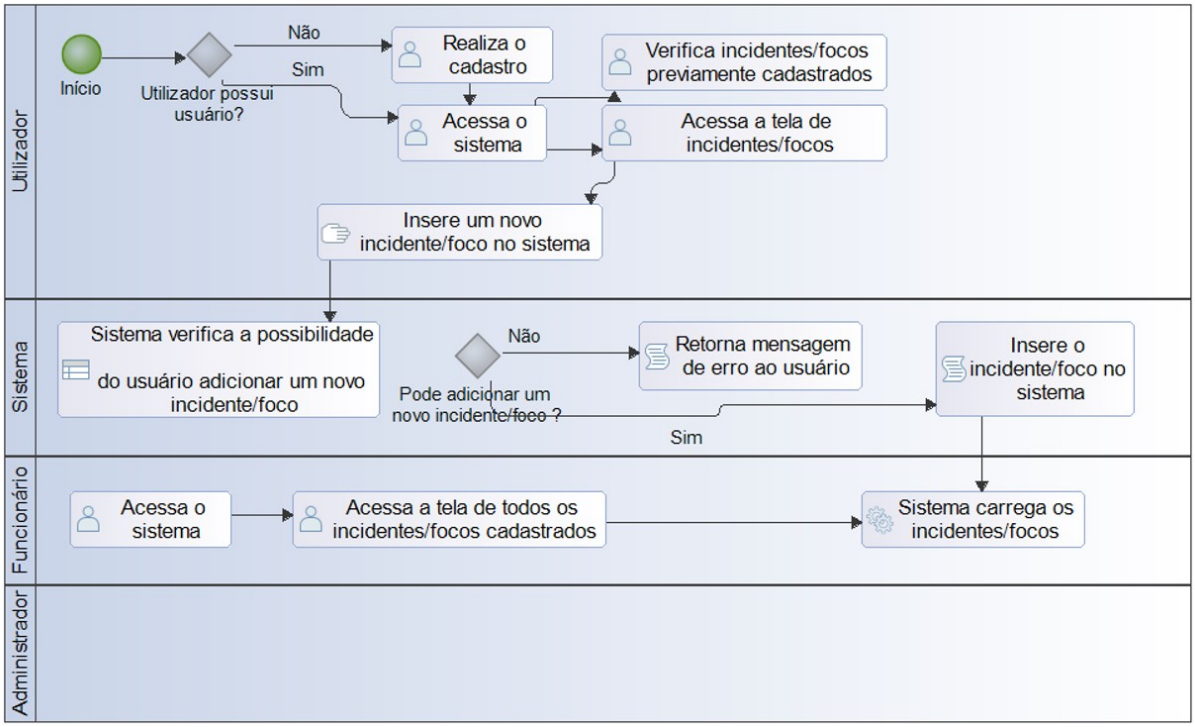
Espera-se que com o desenvolvimento do aplicativo seus usuários possam reportar os focos de dengue por meio de geolocalização, com o objetivo de diminuir os casos, auxiliando também o profissional de combate a dengue na rápida identificação e atuação sobre o local. Além de permitir relatórios detalhados sobre a quantidade de incidentes por localidade e período, permitindo assim que esforços possam ser direcionados de maneira mais efetiva no combate a dengue.

1.2 OBJETIVOS

O objetivo geral desse trabalho é desenvolver um aplicativo para identificar o foco e monitorar sua resolução, que torne simples e eficaz a comunicação entre o cidadão comum e a Prefeitura Municipal, que atue como um intermediador, sendo realizado a partir do uso de geolocalização e imagens, para auxílio do profissional de combate a dengue. A Figura 1

mostra o fluxo do tratamento de um foco de dengue, desde seu cadastro no aplicativo até a efetiva eliminação.

Figura 1 – Fluxo de inserção e atendimento de um foco de dengue no sistema



Fonte: Autoria própria (2021).

Para alcançar tal objetivo buscaram-se os seguintes objetivos específicos:

- Utilização de API (do inglês, *Application Programming Interface*) de Geolocalização.
- Criação de um aplicativo Android através do qual possa ser cadastrado o possível foco de dengue, sendo inseridas informações relevantes como localização, descrição e imagem do local.
- Armazenar os dados em um banco de dados não relacional.
- Utilização de um serviço de autenticação de usuários.

2 REVISÃO DE LITERATURA

Neste capítulo são apresentados assuntos fundamentais para o desenvolvimento deste aplicativo. Na seção de computação em nuvem é apresentado de maneira sucinta conceitos básicos que a definem, além da contextualização com o aplicativo a ser desenvolvido. Já no tópico de serviço de geolocalização são apresentados seus conceitos e também sua contextualização e uso no desenvolvimento do aplicativo. Na seção Seção 2.3 são apresentados de forma sucinta os conceitos de API além de apresentar a API de geolocalização utilizada.

2.1 COMPUTAÇÃO EM NUVEM

A computação em nuvem pode ser definida como um modelo para permitir acesso á uma rede sob demanda de recursos compartilhados, a qual consistem em uma coleção de computadores interconectados e virtualizados, seus recursos podem ser rapidamente provisionados e liberados com mínimo esforço de gerenciamento ou interação com o provedor de serviços.(THE. . . , 2010)(BUYYA et al., 2009)(B., 2016).

Atualmente, com a introdução da computação em nuvem quanto menor for o número de máquinas e parques desatualizados melhor para a produção e o negócio. Com o parque de máquinas atualizado, existe a necessidade do uso crescente de recursos computacionais avançados que muitas vezes as empresas não têm condição de investir. A computação em nuvem permite o acesso a estes recursos sem que haja investimento na aquisição, e ela também permite que aconteça a expansão desses recursos de forma *on-demand* e com grande flexibilidade (PAZ; LOOS, 2020).

A computação em nuvem também é uma grande aliada de startups, ajudando a lidar mais facilmente com várias questões, como a falta de espaço para a alocação de servidores, prevenção de ciberataques como ataque DDoS e diminuição de uso de profissionais externos especializados, como por exemplo o DBA (NADE; ELECTRONIC; PROJECTS, 2021).

Desse modo, a utilização de serviços descentralizados em nuvem torna-se um aliado, sendo possível a utilização de recursos de forma simples e eficaz. Assim, no desenvolvimento desse aplicativo foram utilizados vários serviços baseados em nuvem, como o serviço de geolocalização, serviço de autenticação e armazenamento de dados, o qual serão abordados nas próximas seções.

2.2 SERVIÇO DE GEOLOCALIZAÇÃO

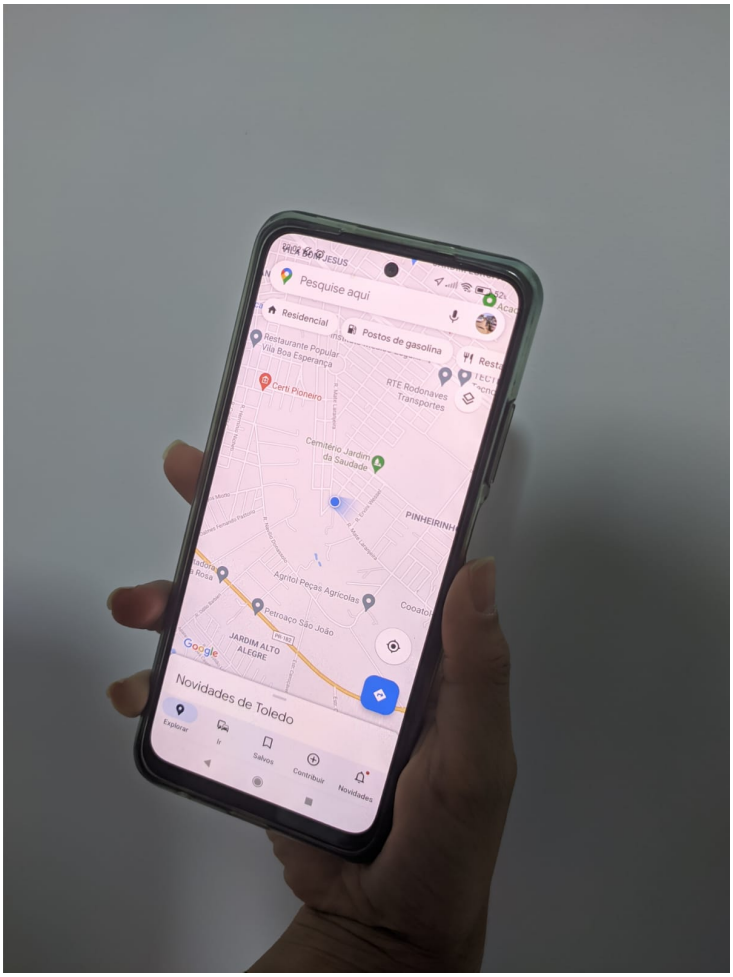
A geolização pode ser definida como o processo de determinar ou estimar a posição geográfica de um objeto(GEOGRAPHIC. . . , 2018).

A geolocalização pode ser dividida em várias categorias, como (ZREIK, 2019):

- Geocodificação, informações de localização para lugares específicos em um mapa.
- Georreferenciamento, informações sobre a localiação física de uma pessoa ou objeto por meio de dados de GPS.
- Geotagging, informações de localização adicionadas a um objeto, como uma postagem de mídia social.

Os dados de geolocalização são utilizados diariamente por muitas pessoas (ZREIK, 2019). Sendo muito importantes para a utilização de aplicativos de mídias sociais e várias outras finalidades, como estimativas de desmatamento e mudanças de florestas, pesquisas de marketing a fim de identificar grupos de usuários específicos com a finalidade de descrever e aumentar a base de clientes (KORTUM; ACEMYAN, 2019), entre outros. A Figura 2 mostra um uso cotidiano da geolocalização, um aplicativo para visualização de mapas em tempo real, no exemplo, trata-se do Google Maps.

Figura 2 – Exemplo de uso da geolocalização no cotidiano, no Google Maps



Fonte: Autoria própria (2021).

No aplicativo desenvolvido foi utilizado os dados de geolocalização com o fim de auxiliar a prefeitura a identificar a localização do foco de dengue, além de facilitar para

o usuário a visualização de possíveis focos. Sendo recuperada a localização por meio de *APIs* de geolocalização, a qual veremos na próxima seção.

2.3 *API* DE GEOLOCALIZAÇÃO

API pode ser definida como uma conexão entre sistemas. Ela fornece uma abstração para um problema e especifica como deve-se interagir com os componentes de *software* que implementam uma solução para esse problema (MARTIN, 2011). A *API* não precisa saber como esses *softwares* foram implementados, dessa forma acarreta em uma simplificação do desenvolvimento gerando uma diminuição de custos e menor tempo de desenvolvimento (REDHAT, 2021).

Alguns motivos para a utilização de *API's* no desenvolvimento da aplicação (MARTIN, 2011), (ZHONG; MEI, 2019):

- Fácil otimização
- Redução de duplicação de código.
- Reutilização de código.
- Aumenta a longevidade do sistema.

Como os objetivos contemplam a utilização de um sistema de armazenamento em banco de dados não relacional, utilização de geolocalização, além de um serviço de autenticação de usuários e necessidade de uso mobile, foi necessário encontrar *API's* que se enquadrassem nesses requisitos.

Para tal, antes de mais nada foi necessário verificar a usabilidade da *API*, podemos definir a usabilidade e dimensões de uma *API* utilizando alguns princípios (BORE; BORE, 2005):

- Especificidade
  - É a porcentagem de elementos da *API* que tratam da funcionalidade do aplicativo.
- Simplicidade
  - Mede a facilidade com que o usuário pode traduzir a funcionalidade necessária do aplicativo em usos de elementos de API.
- Clareza
  - Mede o quão óbvio é o propósito do nome da *API*.

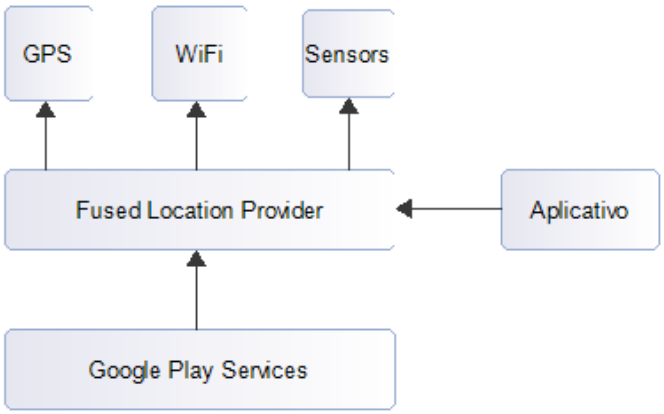
Nesta seção abordaremos o uso da *API* de geolocalização, já as outras *API's* utilizadas serão abordadas no Capítulo 3, Seção 3.3.

Para a geolocalização foi utilizado a *API* gratuita do Maps SDK(GOOGLE, 2021h) em conjunto com a *API Fused Location Provider*(DEVELOPERS, 2021a) ambas da Google, a qual fornecem todos os dados necessários para requisição da localização atual do usuário do aplicativo e também a visualização dessa localização em um mapa.

A Figura 3 demonstra o fluxo de uso da *Fused Location Provider API* onde o aplicativo solicita a localização e a *API* é responsável por recupera-la, todo o processo

de recuperação da localização é realizado de modo automático pela *API*, mas podem ser repassados alguns parâmetros como a prioridade e nível de uso de bateria, no Capítulo 3 na Seção 3.5 será mostrado como foi realizado o desenvolvimento da aplicação e consequentemente a configuração de tais parâmetros.

Figura 3 – diagrama de acesso do *Fused Location Provider API*

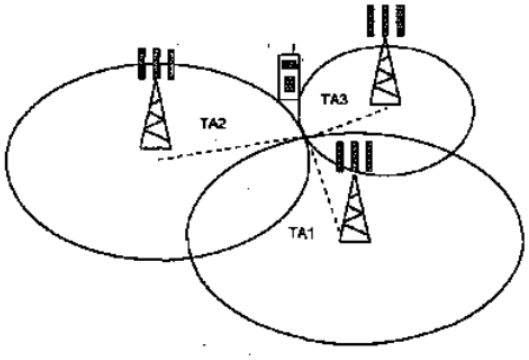


Fonte: Developers (2021b)

No âmbito da recuperação da localização do utilizador do aplicativo, o *Fused Location Provider API* realiza os acessos aos sensores necessários para requisitar a localização do *smartphone*. Um dos métodos utilizados é a localização via sensor GPS, nela é recuperado a localização atual do *smartphone* pela triangulação de sinal entre satélites, onde o sinal é encaminhado ao *smartphone* e o atraso entre o envio e o recebimento é calculado, estimando a localização atual do *smartphone*. Um outro método alternativo, mas que possui o funcionamento semelhante para determinar a localização do *smartphone*, é estimar distância entre três torres de celular, também conhecidas como ERB (Estação Rádio Base) próximas. A distância entre o *smartphone* e cada antena pode ser estimada baseado no tempo de atraso entre o momento que a torre envia o *ping* para o *smartphone* e recebe a resposta de volta (SMITH, 2008), (HUSSAIN et al., 2007). Ambos os resultados, dependendo da disponibilidade de sinal, são muito precisos, auxiliando no desenvolvimento do aplicativo. Uma representação da recuperação da localização do usuário por triangulação de ERB (Estação Rádio Base) pode ser verificada na Figura 4.

Para fins explicativos a realização da triangulação da localização do dispositivo por meio de torres de celular é realizada em três passos. O primeiro passo é recuperar a força do sinal captado pela estação transmissora, o segundo passo envolve aproximar a distância da estação transmissora correspondente com a ajuda da força do sinal recebido, já no último passo é encontrado a localização geográfica real do dispositivo. Para o último passo é realizado uma intersecção de dois ou mais círculos, para tal, (a,b) e (c,d) são o centro dos círculos, com raio (r) e (s), estes círculos podem ser definidos pela Equação 1

Figura 4 – Exemplo de triangulação da localização do usuário por meio de ERB (Estação Rádio Base)



Fonte: (HUSSAIN et al., 2007)

(HUSSAIN et al., 2007):

$$(x-a)^2 + (y-b)^2 = r^2$$
$$(x-c)^2 + (y-d)^2 = s^2$$

A Equação 1 é uma equação do segundo grau e para ser resolvida ela precisa ser convertida para uma forma linear. Entre muitas soluções, uma das soluções mais convenientes envolve a rotação do eixo para encontrar os pontos de intersecção. O algoritmo é realizado conforme se segue (HUSSAIN et al., 2007):

Os pontos de intersecção $(x1, y1)$ e $(x2, y2)$ dos dois círculos são calculados da seguinte maneira:

$$e = c - a$$
$$f = d - b$$
$$p = \sqrt{(e^2 + f^2)}$$
$$k = \frac{(p^2 + r^2 - s^2)}{2p}$$

Dessa forma, os pontos são:

$$x1 = a + \frac{ek}{p} + \frac{f}{p} * \sqrt{(r^2 - k^2)}$$
$$y1 = b + \frac{fk}{p} + \frac{e}{p} * \sqrt{(r^2 - k^2)}$$

e respectivamente

$$x2 = a + \frac{ek}{p} + \frac{f}{p} * \sqrt{(r^2 - k^2)}$$
$$y2 = b + \frac{fk}{p} + \frac{e}{p} * \sqrt{(r^2 - k^2)}$$

A intersecção de três círculos é apenas uma extensão dessa fórmula, e envolve encontrar a intersecção dos três círculos e determinar o ponto mais viável, que é conhecido como epicentro.

Para visualização em mapa da latitude e longitude recuperada pela *Fused Location Provider API* foi utilizado o Maps SDK. A Figura 5 mostra um exemplo de uso do Maps SDK, sendo utilizado para visualizar um ponto no mapa.

Figura 5 – Exemplo de uso do Maps SDK

Fonte: Autoria própria (2021).

3 METODOLOGIA

Este trabalho refere-se a implementação de uma ferramenta para auxiliar no registro dos focos de dengue além de auxiliar o profissional de combate a endemias a registrar suas visitas.

O trabalho apresentado conta com a abordagem quantitativa e teve como foco os dados de casos de dengue no Brasil.

Neste capítulo são apresentados assuntos fundamentais quanto a metodologia utilizada. Na seção de projeto são apresentados os requisitos funcionais e as regras de negócio utilizadas no desenvolvimento do aplicativo, além dos diagramas de casos de uso e diagramas de sequência, na seção de modelagem de entidades é apresentado a modelagem das classes e suas interações, além do diagrama UML do aplicativo, em *API's* utilizadas no aplicativo é apresentado quais *API's* foram utilizadas. Já na seção de Banco de dados não relacional é apresentado de forma resumida seus conceitos e usos, além de sua utilização conjunta do aplicativo desenvolvido. Em seguida na seção de desenvolvimento do aplicativo móvel é apresentado o desenvolvimento do aplicativo.

3.1 PROJETO

O conjunto de requisitos do aplicativo é apresentado no Quadro 1.

Quadro 1 – Requisitos Funcionais.

ID	**Descrição**
RF1	O aplicativo deve permitir que o usuário cadastre-se no sistema.
RF2	O aplicativo deve permitir e garantir que apenas os usuários cadastrados consigam inserir novos focos/incidentes no sistema.
RF3	O aplicativo deve permitir que os usuários insiram sua localização atual no momento da inserção do foco/incidente.
RF4	O aplicativo deve permitir que o usuário anexe uma imagem ao inserir o foco/incidente.
RF5	O aplicativo deve permitir que o usuário insira uma descrição do incidente/foco.
RF6	O aplicativo deve permitir que o usuário insira uma localização secundária do incidente/foco, sendo selecionada em um mapa.
RF7	O aplicativo deve permitir a inserção de administradores e funcionários.
RF8	O aplicativo deve permitir e garantir que apenas os administradores e funcionários excluam focos/incidentes que estejam com status diferente de aberto.
RF9	O aplicativo deve permitir e garantir que funcionários consigam cadastrar visitas aos focos/incidentes ou realizar a inserção de uma visita manualmente.
RF10	O aplicativo deve permitir e garantir que as alterações que forem feitas quando o dispositivo não estiver conectado a internet sejam sincronizadas ao ficar online novamente.
RF11	O aplicativo deve permitir que o administrador cadastre um limite de focos/incidentes abertos pelo usuário que não estejam em atendimento, evitando Spam.
RF12	O aplicativo deve permitir e garantir que as mudanças realizadas em um foco/incidente sejam salvas em um histórico, podendo ser vista tanto pelo usuário solicitante quando por outros funcionários e administradores.
RF13	O aplicativo deve permitir que o administrador visualize relatórios de bairros com maior quantia de focos/incidentes, mês do ano com maior quantia de focos/incidentes incluídos além de relatórios de visualização de quantia de atendimento de focos/incidentes por funcionário em um determinando período.

Fonte: Autoria própria (2021).

O conjunto de regras de negócio do aplicativo é apresentado no Quadro 2.

Quadro 2 – Regras de negócio.

ID	Descrição
RN1	O usuário realiza o login no aplicativo com seu usuário e senha.
RN2	O usuário acessa a tela de inserção de foco/incidente, insere as informações e aciona o botão de salvar.
RN3	O aplicativo verifica se o usuário possui mais cadastros de focos/incidentes que o limite de cadastro aberto permite, evitando a inclusão de incidentes repetidos e SPAM.
RN4	O aplicativo verifica se a descrição do problema foi preenchida além da localização e do tipo do foco/incidente.
RN5	O funcionário realiza o login no aplicativo com seu usuário e senha.
RN6	O funcionário acessa a tela de visualização de focos/incidentes cadastrados.
RN7	O aplicativo mostra apenas os focos/incidentes que não estão sendo tratados por outros funcionários.
RN8	O funcionário assume o foco/incidente, selecionando seu usuário como responsável e adicionando um status novo, na tela de exibição de detalhes do foco/incidente e aciona o botão salvar.
RN9	Após realizar o atendimento o funcionário modifica o status do foco/incidente.
RN10	Caso o funcionário queira também pode incluir uma visita ao foco/incidente, selecionando sua data e horário, o que foi realizado e também caso queira, observações.
RN11	Após realizar os devidos tratamentos, o funcionário altera o status do foco/incidente para solucionado.
RN12	O usuário verifica o histórico de alterações e de visitas ao foco/incidente que cadastrou.

Fonte: Autoria própria (2021).

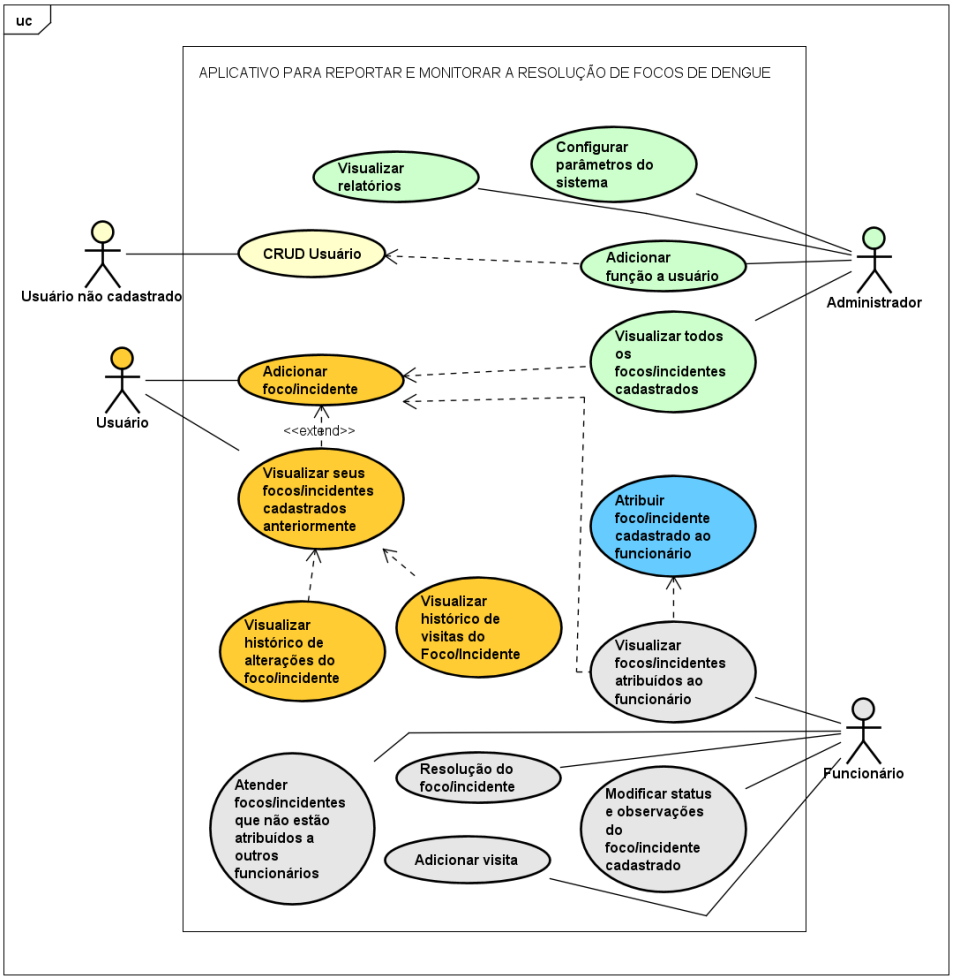
A partir dos requisitos foram desenvolvidos os diagramas de casos de uso, sequência e classe. Esses diagramas documentam os fluxos e entidades envolvidas nos processos desempenhados pelo usuário, administrador e funcionário. A Figura 6 demonstra o diagrama de Casos de Uso geral do aplicativo.

O `usuário não cadastrado` precisa se cadastrar para se tornar um `usuário`. A partir disso, ele poderá adicionar um novo foco/incidente, visualizar os seus focos/incidentes cadastrados ou visualizar os que estejam visíveis para todos.

Para realizar a inclusão de um foco/incidente é necessário que o `usuário` acesse a tela de inclusão de foco/incidente, preencha a descrição, o tipo do foco/incidente , utilize a sua localização atual ou selecione um local no mapa que será a localização do foco/incidente e também tem a escolha de incluir uma foto da galeria de seu dispositivo. Esse fluxo está representando na Figura 7.

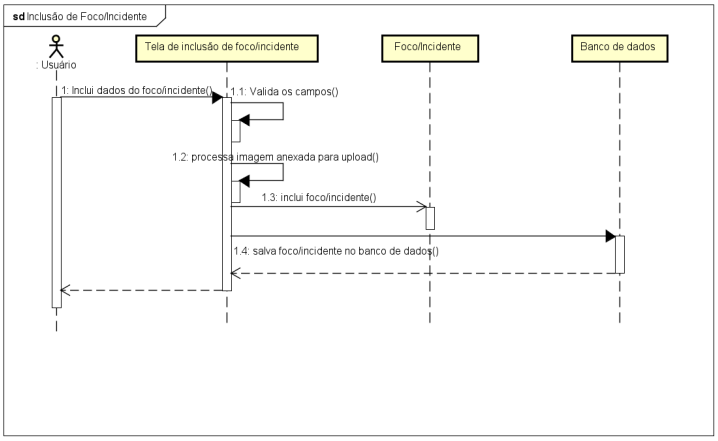
Após a ação do `usuário` de adicionar um foco/incidente o `funcionário` pode visualiza-lo na lista de focos/incidentes, a partir desse momento o `funcionário` pode realizar as alterações necessárias no foco/incidente, sendo ele visível para todos os `funcionários` até

Figura 6 – Diagrama de casos de uso Geral



Fonte: Autoria própria (2021).

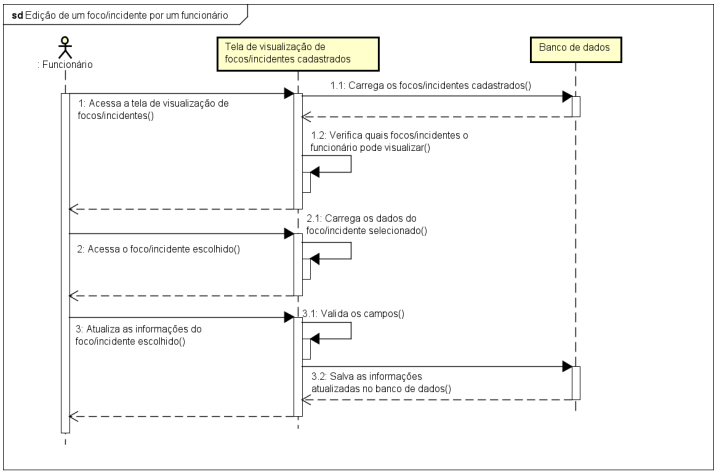
Figura 7 – Diagrama de sequência para inclusão de foco/incidente



Fonte: Autoria própria (2021).

que seja alterado o `Atendente para o incidente`, a partir disso apenas o `funcionário` selecionado conseguirá visualiza-lo, além do `usuário` que realizou a inclusão do foco/incidente e o `administrador`. O fluxo de edição de um foco/incidente está representado na Figura 8.

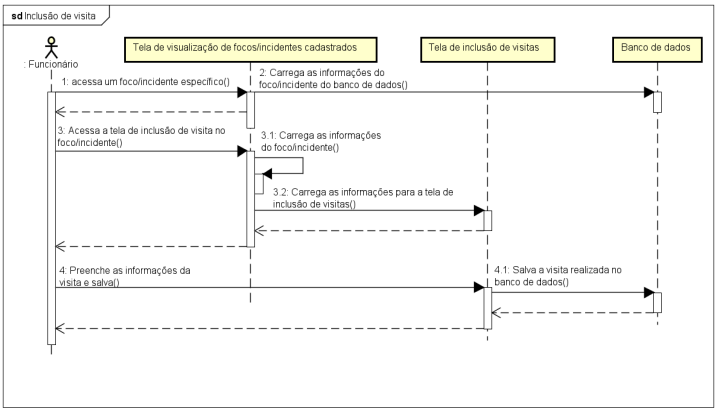
Figura 8 – Diagrama de sequência para edição de um foco/incidente realizada por um funcionário



Fonte: Autoria própria (2021).

O `funcionário` também pode realizar a inclusão de uma visita após a inclusão de um foco/incidente, ou realizar a inclusão de uma visita sem ser para um foco/incidente específico, tal fluxo está representado na Figura 9.

Figura 9 – Diagrama de sequência para a inclusão de uma visita, realizada por um funcionário

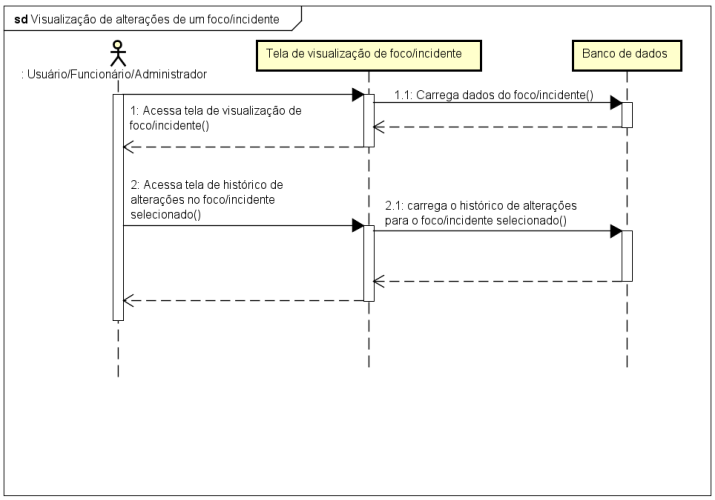


Fonte: Autoria própria (2021).

A partir da visualização de um foco/incidente é possível verificar as alterações realizadas, na opção de histórico de alterações, todas as alterações no foco/incidente são gravadas em formato de lista, onde é possível verificar quem alterou, o que foi alterado e quando foi alterado, tanto o `usuário`, `administrador` e `funcionário` conseguem verificar

esse histórico de alterações. O fluxo de visualização do histórico de alterações de um foco/incidente está representando na Figura 10.

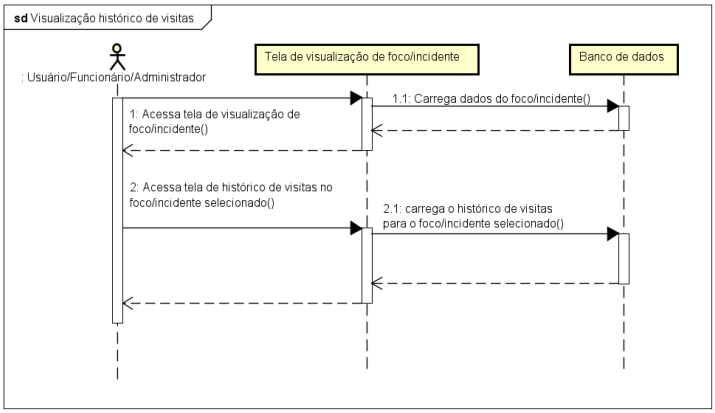
Figura 10 – Diagrama de sequência para visualização do histórico de alterações realizadas em um foco/incidente



Fonte: Autoria própria (2021).

Na visualização de um foco/incidente também é possível verificar o histórico de visitas adicionadas pelo `funcionário`, sendo possível verificar, caso existam visitas adicionadas, a atividade realizada na visita, a data e hora da visita, o email do `funcionário` que realizou a visita, as observações e também a data de inclusão da visita. O fluxo de visualização do histórico de visitas realizadas em um foco/incidente esta representando na Figura 11.

Figura 11 – Diagrama de sequência para visualização do histórico de visitas realizadas a um foco/incidente

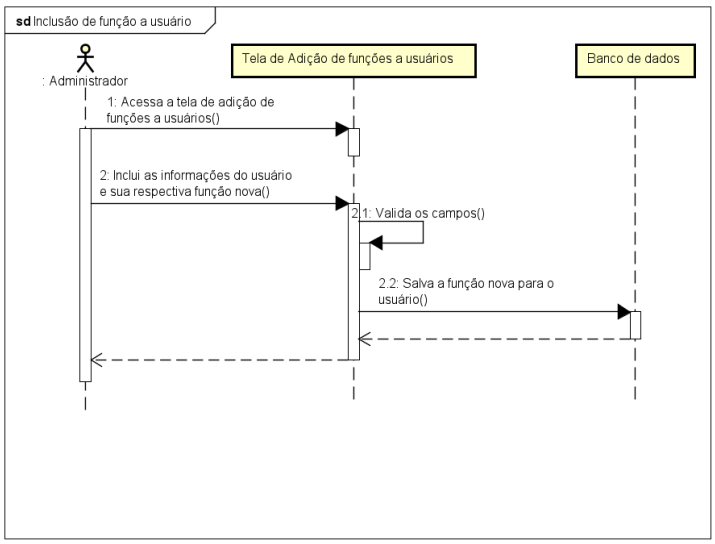


Fonte: Autoria própria (2021).

O papel do `administrador` é responsável por adicionar a função de `funcionário` ou `administrador` a um `usuário` previamente cadastrado, sendo tal ação realizada por

meio da tela de `Administração do sistema`, em `Adicionar funções a usuários`, para tal é necessário que o `administrador` adicione o email do `usuário`, insira seu nome completo e também adicione a sua função/atribuição nova. Nessa mesma tela é possível verificar os cadastros de funções para usuários previamente cadastrados, e caso seja da vontade do `administrador` exclui-las. O fluxo de inclusão está representado na Figura 12, já o fluxo de visualização e exclusão de função a usuários está representado na Figura 13.

Figura 12 – Diagrama de sequência para inclusão de função a usuário



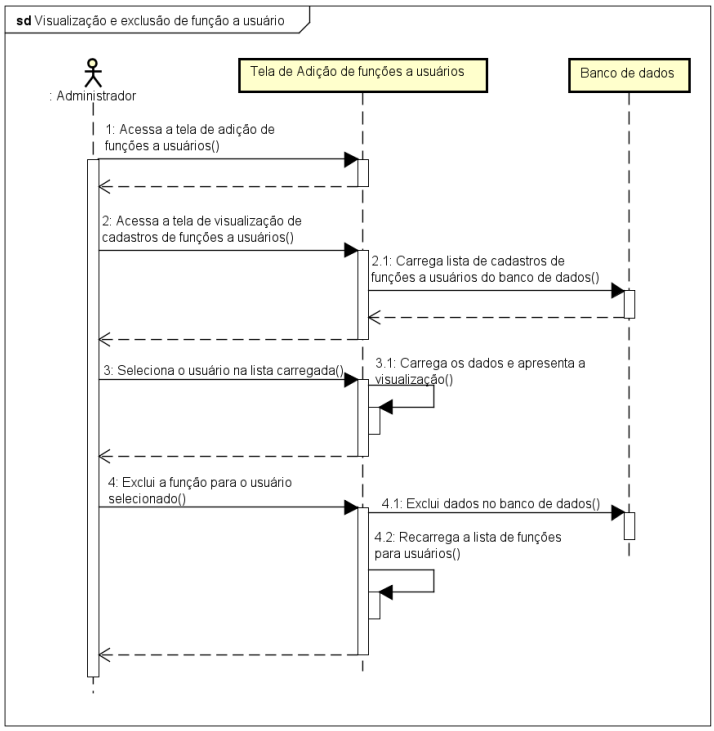
Fonte: Autoria própria (2021).

O papel do `administrador` também é responsável por configurar os parâmetros do sistema, como por exemplo a quantia máxima de focos/incidentes cadastrados por um `usuário`, o fluxo de configuração dos parâmetros do sistema está representado na Figura 14.

O papel do `administrador` tem a possibilidade de realizar a exclusão em massa de focos/incidentes, isso é possível para evitar armazenamento de dados desnecessários no banco, sendo possível realizar a exclusão pela quantia de dias que um foco/incidente foi solucionado ou a quantia de dias que foi incluído, ignorando seu status. Tal fluxo de exclusão de focos/incidentes em massa está representado na Figura 15.

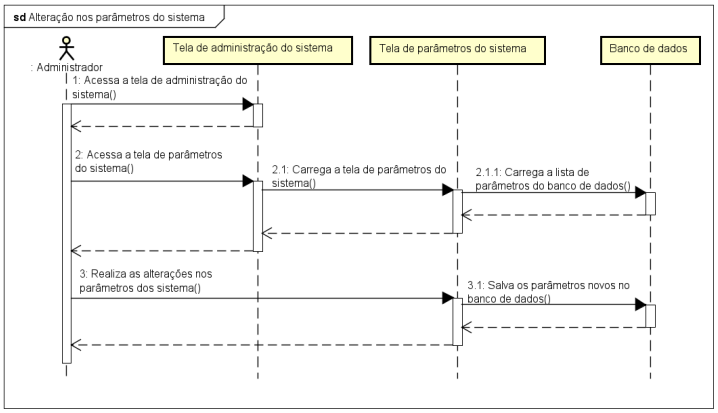
Após a inclusão dos focos/incidentes, o `administrador` pode verificar vários relatórios na tela de `dashboard do administrador` em `administração do sistema`, são eles: Bairro com maior quantia de focos/incidentes em um período específico, focos/incidentes cadastrados nos últimos 7 dias, mês que possui mais focos/incidentes em um determinado ano, quantia de focos/incidentes atendidos por um `funcionário` em um determinado período e a quantia de visitas realizadas por um `funcionário` em um determinado período. Tais relatórios possuem informações, que podem ser verificadas acionando o botão `sobre` em cada um de seus cards, o fluxo de visualização dos relatórios está representado na

Figura 13 – Diagrama de sequência para inclusão de função a usuário



Fonte: Autoria própria (2021).

Figura 14 – Diagrama de sequência para alteração dos parâmetros dos sistema



Fonte: Autoria própria (2021).

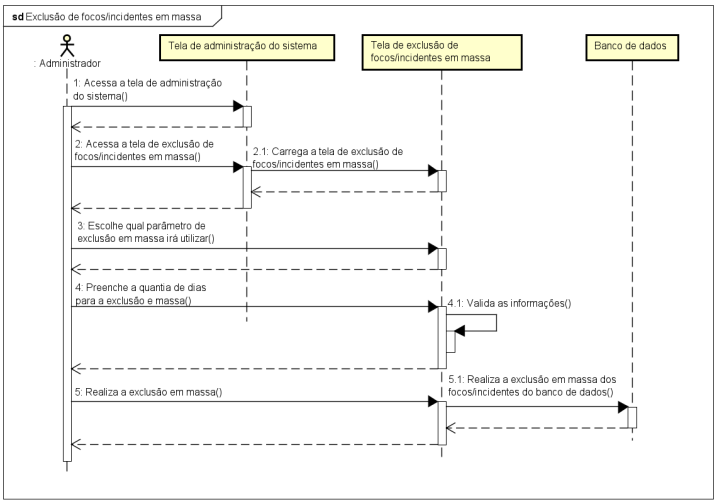
Figura 16.

3.2 MODELAGEM DE ENTIDADES

As *Activityes* do sistema estão vinculadas a pelo menos uma das classes representada na Figura 17.

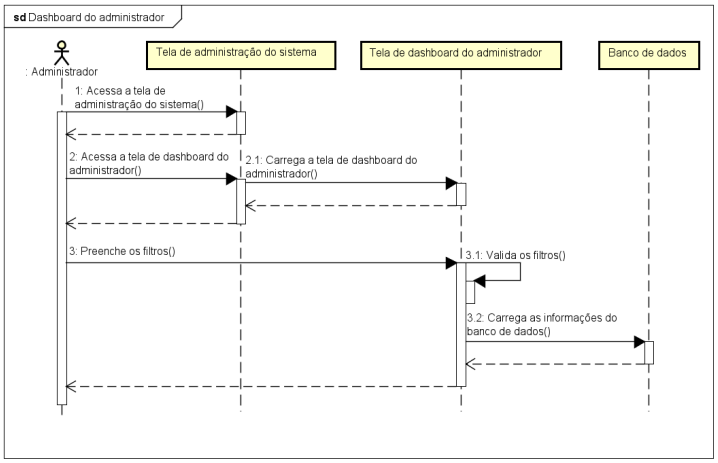
Todas as classes que possuem campos de identificação geram esses campos

Figura 15 – Diagrama de sequência para exclusão em massa de focos/incidentes



Fonte: Autoria própria (2021).

Figura 16 – Diagrama de sequência para a visualização do dashboard do administrador



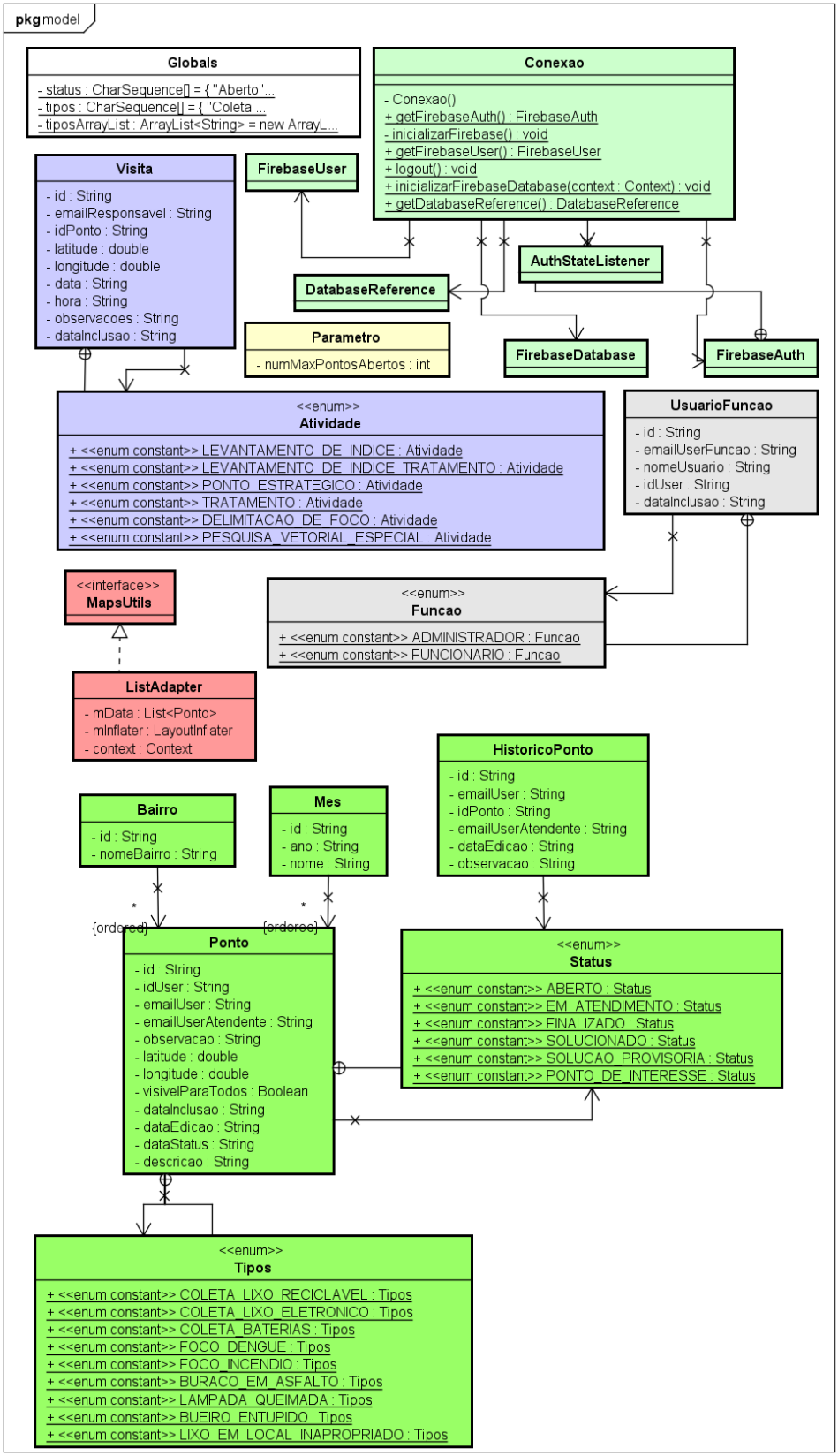
Fonte: Autoria própria (2021).

baseando-se na data atual e em um valor randômico, e ao serem persistidos no banco armazanam esses valores para serem recuperados posteriormente.

A classe/tabela `Ponto` é a principal do aplicativo, em sua referência no banco de dados é que são armazenados os focos/incidentes, possui esse nome genérico no aplicativo para facilitar a identificação, e está vinculada a outras, como o `HistoricoPonto` onde são armazenados os históricos de alterações realizados no foco/incidente. Para fins de relatórios ela está vinculada também a classe `Bairro` e a classe `Mes`, que são respectivamente utilizadas para extração de relatório de focos/incidentes por bairro e focos/incidentes por mês de um determinado ano.

O enum `Tipos` serve para a identificação do foco/incidente, foram adicionadas várias opçoes, mas a utilizada nesse trabalho é `Foco de dengue`, os demais são para testes

Figura 17 – Diagrama de classes



Fonte: Autoria própria (2021).

e também para trabalhos futuros.

Já o enum `Status` serve para a verificação do status do foco/incidente, quando incluído, é utilizado o status padrão `Aberto`, o status `Em atendimento` é para quando o foco/incidente está sendo tratado por algum `funcionário`. A principal diferença do status `solucionado` para `finalizado` é que o `solucionado` representa que o foco/incidente foi solucionado, já o `finalizado` representa que ele foi finalizado sem uma trativa, por exemplo, quando faltam informações no foco/incidente cadastrado ou as informações são conflitantes.

A classe/tabela `Visita` armazena as visitas realizadas por um `funcionário`, ela pode armazenar o id de um foco/incidente a partir do `idPonto`, mas não depende do foco/incidente, sendo possível adicionar uma visita avulsa. O enum `Atividade` que está vinculado a `visita` serve para elucidar o que foi realizado na visita, as descrições das atividades foram extraídas conforme a ficha de visita disponível no Apêndice A.

A classe/tabela `UsuarioFuncao` armazena as funções atribuídas a determinados `usuários`, sendo que quando um `usuário` se cadastra no sistema ele não é incluído nessa tabela, apenas quando o `administrador` adiciona uma função para ele, seja de `administrador` ou `funcionário`, o enum `Funcao` armazena as funções disponíveis, sendo ela `administrador` e `funcionário`.

A classe/tabela `Parametro` armazena os parâmetros cadastrados, sendo essencial para o `administrador`"regular certas funções do aplicativo, como a quantia máxima de focos/incidentes incluídos por um `usuário` que estejam com status `Aberto`.

Na classe/tabela `Globals` existem alguns métodos públicos úteis no desenvolvimento, reduzindo acoplamento.

Por último, sendo umas das mais importantes, temos a classe `Conexao`, a qual não é armazenada no banco de dados, sendo utilizada para realizar a conexão com o banco de dados com `DatabaseReference` e `FirebaseDatabase` e também realizar a autenticação com `FirebaseAuth`, `AuthStateListener` e `FirebaseUser`.

3.3 *APIs* UTILIZADAS NO APLICATIVO

Para se atingir os objetivos de autenticação de usuário e armazenamento de dados foi utilizado o Firebase.

O Firebase não é apenas uma *API*, mas também um MBaaS (do inglês, *backend as a service*), que vem consolidando-se no mercado, criada em 2012, ela é mantida pela Google, sendo utilizada por grandes empresas como Alibaba.com, The New York Times, OneFootball, The Economist e entre várias outras (GOOGLE, 2021g). Trata-se de uma *API* robusta com uma vasta documentação (GOOGLE, 2021c), e seu uso pode-se adequar para várias situações, como necessidades de autenticação de usuário (GOOGLE, 2021d), armazenamento de dados (GOOGLE, 2021f) além de uso *mobile* com desenvolvimento Android ou iOS, quanto desenvolvimento *WEB* com *JS* (GOOGLE, 2021c).

Desse modo, o Firebase *API* enquadra-se nos princípios de usabilidade de *API* (BORE; BORE, 2005), pois é específica para ferramenta a ser desenvolvida, sendo simples e clara e possuindo vasta documentação.

O Firebase possui alguns planos, o primeiro, chamado de Plano Spark, possui várias funcionalidades, como possibilidade de autenticação de usuários por vários provedores, como provedor de email, conta do Google e outros (GOOGLE, 2021d), além de acesso a um banco de dados não relacional, com nome de *Realtime database*(GOOGLE, 2021f), onde os dados são salvos em formato JSON, com capacidade total de 1 GB. Mesmo 1 GB parecendo pouco, lembre-se que um banco de dados não relacional consome menos armazenamento que um banco de dados relacional (JOSE; ABRAHAM, 2017) (CONTI, 2020). A Figura 18 mostra um exemplo da estrutura de um objeto armazenado no Firebase *Realtime database*.

Figura 18 – Exemplo de estrutura do Firebase Realtime Database

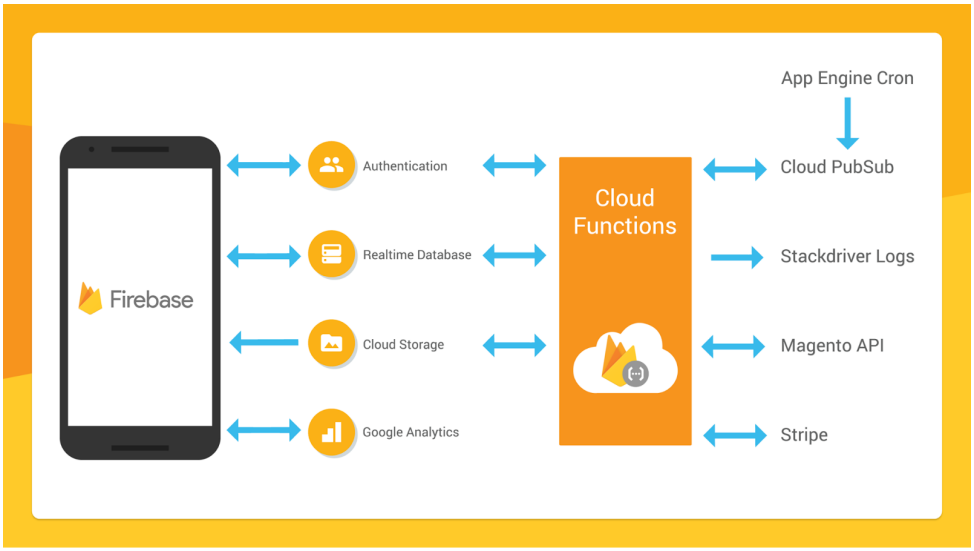
Fonte: Puffelen (2016)

A autenticação de usuários é um serviço a parte da *API* não havendo limites de login para métodos de autenticação comum como Email, já o armazenamento de arquivos como fotos também é um serviço a parte, com nome de *Cloud Storage*(GOOGLE, 2021a) havendo capacidade de armazenamento de 5gb (GOOGLE, 2021e). A Figura 19 mostra como é dividido a arquitetura do Firebase, mostrando a comunicação entre o aplicativo e as API's que foram utilizadas no desenvolvimento, como *Authentication*, *Realtime Database* e *Cloud Storage*.

3.4 BANCO DE DADOS NÃO RELACIONAL

O banco de dados não relacional é um tipo de banco de dados que não utiliza um esquema de tabelas, linhas e colunas encontrado na maior parte dos bancos de dados tradicionais. Os bancos de dados não relacionais usam um modelo de armazenamento otimizado para os tipos de dados que estão sendo armazenados, sendo muito utilizados para o armazenamento de grandes volumes de dados (VERA-OLIVERA et al., 2021). Por

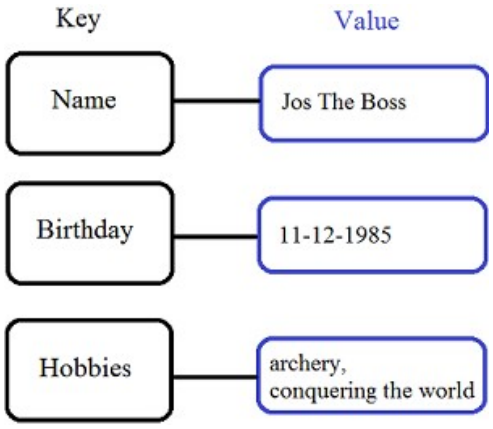
Figura 19 – Arquitetura do Firebase



Fonte: devs (2017)

exemplo, os dados podem ser armazenados como pares chave/valor como em documentos JSON (AZURE, 2018) (VERA-OLIVERA et al., 2021), a Figura 20 mostra a estrutura de um banco de dados NoSQL a qual utiliza a forma de armazenamento chave/valor, como é o caso do Firebase *Realtime Database* (GOOGLE, 2021f).

Figura 20 – Estrutura de um banco *NOSQL* do tipo chave/valor



Fonte: Meysman (2016)

Sabedo dessas informações será utilizado um banco de dados não relacional para desenvolvimento do aplicativo pelos seguintes motivos:

- Desempenho maior em inserções, atualizações e exclusões que nos bancos de dados relacionais comuns (CONTI, 2020).
- Armazenamento facilmente escalável, sendo necessário pouco gasto para armazenar mais dados (JOSE; ABRAHAM, 2017).

- Baixa necessidade de administração por um *DBA*, por ter pouca necessidade de manutenção (JOSE; ABRAHAM, 2017).

Desse modo, como o Firebase contempla um banco de dados não relacional, baseado em chave/valor para armazenamento de dados, que já foi abrangido na Seção 3.3, foi decidido utilizar o serviço presente na *API* para desenvolvimento do armazenamento.

3.5 DESENVOLVIMENTO DO APLICATIVO MÓVEL

Para o desenvolvimento do aplicativo mobile foi utilizado a *IDE* Android Studio. Após ser criado um aplicativo inicial, sem telas, foi adicionado a configuração do Firebase (GOOGLE, 2021b). Para a estruturação do projeto do aplicativo foi utilizado uma simplificação do modelo MVC (do inglês, *Model View Controller*), onde as *Activityes* agem como a View, onde é apresentado o conteúdo mas também agem como `Controller`, pois nelas que são solicitados as informações presentes no banco de dados, o banco de dados em nuvem, do Firebase age como `Model`, que recupera as informações e transmite para o `Controller` atualizar as informações nas `Views`.

Logo após foi incluído as telas de autenticação no sistema, conforme Figura 24 e Figura 25. A partir da utilização do FirebaseAuth (GOOGLE, 2021d) foi possível construir a autenticação do aplicativo, no Algoritmo 1 é possível verificar que toda a autenticação é realizada pelo método login utilizando o objeto de autenticação FirebaseAuth.

Após o usuário realizar o *login* no aplicativo ele é redirecionado para a tela principal. O aplicativo verifica se ele possui alguma função como ”administrador”ou ”funcionario”e realiza a liberação das telas, sendo que a tela principal do usuário possui as informações básicas conforme Figura 26, já a tela do administrador possui as telas básicas e também as telas de administração, conforme Figura 27, já a tela inicial do funcionário possui as opções liberadas ao funcionário, conforme Figura 28.

Uma opção do usuário ao acessar a tela inicial é a inclusão de um novo Foco/incidente, ao acessá-la ele se depara com algumas opções, conforme Figura 21.

Para obter a localização atual do usuário foi utilizado o objeto fusedLocationProviderClient, que como explanado anteriormente, na Seção 2.3 atua como uma *API*, sendo requisitado de forma simplificada e funcional a localização atual do usuário, a parte do método referente a obtenção da localização atual do usuário está disponível na Algoritmo 2, para seu uso como visto, é apenas necessário setar o intervalo de atualização da localização atual e também setar a prioridade.

Para a obtenção da localização via mapa foi utilizado o Maps SDK, e seu uso é prático pois é necessário apenas instanciar a tela e capturar a localização selecionada pelo usuário

Ao inserir as informações pertinentes e acionar o botão ”Adicionar foco/incidente”, após as validações de campos, é gerado um Id para o foco/incidente incluído e salvo no banco de dados, a parte do método referente á inclusão do foco/incidente no banco de dados

Algoritmo 1: Método de autenticação na tela de login

```
1  private void login(String email, String senha) {
2    progressBarLogin.setVisibility(View.VISIBLE);
3    auth.signInWithEmailAndPassword(email, senha).addOnCompleteListener(
4    Login.this,
5      new OnCompleteListener < AuthResult > () {
6        @Override
7        public void onComplete(@NonNull Task < AuthResult > task) {
8          if (task.isSuccessful()) {
9            Intent i = new Intent(Login.this, MainActivity.class);
10           startActivity(i);
11           finish();
12         } else {
13           try {
14             throw task.getException();
15           } catch (FirebaseAuthInvalidCredentialsException e) {
16             edtLoginEmail.setError("Login e/ou senha incorreto");
17             edtLoginEmail.requestFocus();
18             progressBarLogin.setVisibility(View.INVISIBLE);
19             btnLogin.setEnabled(true);
20             btnNovoUsuario.setEnabled(true);
21             ...
22           }
23         }
24       }
25     });
26   }
```

Fonte: Autoria própria (2021).

está disponível na Algoritmo 3 . Após o preenchimento das informações, conforme exemplo na Figura 29, e as informações do foco/incidente serem salvas, é processado a imagem anexada para upload, para tal é realizado a tentativa de compressão da imagem, utilizando a biblioteca Zetbaitsu/Compressor(ZETBAITSU, 2021) e posteriormente, realizado o upload da imagem utilizando o *Firebase Storage Reference*.

Todas as demais classes que são persistidas em banco como `UsuarioFuncao`, `HistoricoPonto`, `Parametro` e `Visita` utilizam do mesmo padrão, sendo criado o objeto e persistido, utilizando o objeto `databaseReference`. Por exemplo, no evento de edição de um foco/incidente, é criado também um objeto da classe `HistoricoPonto`, para armezenar as modificações, como pode ser verificado na Algoritmo 4.

Na tela de visualizaçao de focos/incidentes, disponível tanto para usuário (Figura 30, funcionário (Figura 31) e administrador (Figura 32) podem ser realizados filtragens nos focos/incidentes visíveis, os seguintes filtros estão disponíveis: Filtrar por data de

Algoritmo 2: Parte do método responsável pela obtenção da localização atual do usuário

```
locationRequest = new LocationRequest();

//prioridade de gasto de energia e etc.
//modo mais acurado - usa GPS
locationRequest.setPriority(LocationRequest.PRIORITY_HIGH_ACCURACY);

//a cada quanto tempo a localização verifica a acuração ?
locationRequest.setInterval(1000 * DEFAULT_UPDATE_INTERVAL);

//a cada quanto tempo é verificado a acuração setado para o mais rápido ?
locationRequest.setFastestInterval(1000 * FAST_UPDATE_INTERVAL);

fusedLocationProviderClient.requestLocationUpdates(
locationRequest, locationCallBack, null);
```

Fonte: Autoria própria (2021).

Algoritmo 3: Método responsável pela inclusão de um foco/incidente

```
String descricao = edtDescPonto.getEditText().getText().toString();
String dataInclusao = Globals.getDateFormatdddmmyyyyhhmmss().
format(calendar.getTime());
String idUserLogado = firebaseUser.getUid();
String emailUserLogado = firebaseUser.getEmail();
Ponto.Tipos tipo = Globals.getPontoTipoByCharSequence(
dropdownTipos.getText().toString());

Ponto p = new Ponto();
p.setId(uuid);
p.setIdUser(idUserLogado);
p.setEmailUser(emailUserLogado);
p.setVisivelParaTodos(false);
p.setTipo(tipo);
p.setStatus(Ponto.Status.ABERTO);
p.setLatitude(localizacaoAtual.getLatitude());
p.setLongitude(localizacaoAtual.getLongitude());
p.setDataInclusao(dataInclusao);
p.setDescricao(descricao);

databaseReference.child("Ponto").child(uuid).setValue(p);

processaImagemParaUpload();
```

Fonte: Autoria própria (2021).

Figura 21 – Tela de inclusão de um Foco/Incidente

Fonte: Autoria própria (2021).

inclusão, a qual filtra os focos/incidentes pela data de inclusão, da menor a maior data, Filtrar por status, onde é possível filtrar pela lista de status disponíveis, por exemplo, filtrar apenas os focos/incidentes com status `Solucionado` e por último também é possível Filtrar por tipo do foco/incidente.

Um dos principais componentes do aplicativo é a área de administração, sendo responsável pela manutenação dos usuários e respectivas funções, relatórios, exclusões de focos/incidentes e configurações de parâmetros globais. A tela principal de administração pode ser verificada na Figura 22.

Mesmo possuindo várias atribuições, a forma com que lida com os dados do banco de dados é a mesma, os objetos são mapeados e armazenados em arrays dentro do aplicativo, para serem analisados e processados, um exemplo disso é o método responsável por filtrar a quantia de focos/incidentes abertos nos últimos 7 dias, disponível dentro da *activity* Dashboard do Administrador (Figura 42), tal método pode ser visualizado na Algoritmo 5.

As telas disponíveis para o `funcionário`, assim como todas as outras telas, foram feitas para serem utilizadas de forma a assegurar que os dados persistidos localmente sejam integrados assim que o utilizador, seja ele `usuário`, `funcionário` ou `administrador` conectar-se a internet, possibilitando que o utilizador faça uso do aplicativo sem que esteja conectado a internet, um exemplo é a inclusão de uma visita por parte do `funcionário`, a

Algoritmo 4: Parte do método responsável pela inclusão de um historico de alteração na edição de um foco/incidente

```
1  HistoricoPonto historicoPonto = new HistoricoPonto();
2  historicoPonto.setId(geraUUID());
3  historicoPonto.setEmailUser(firebaseUser.getEmail());
4  historicoPonto.setEmailUserAtendente(ponto.getEmailUserAtendente());
5  historicoPonto.setIdPonto(ponto.getId());
6  historicoPonto.setDataEdicao(dataStatus);
7  historicoPonto.setStatus(status);
8  historicoPonto.setObservacao(edtObsIncidente.getEditText().getText().toString());
9
10 databaseReference.child("HistoricoPonto").child(
11 historicoPonto.getId()).setValue(historicoPonto);
12 finish();
```

Fonte: Autoria própria (2021).

tela de inclusão de visita por parte do `funcionário` pode ser visualizada na Figura 23.

A sincronização independente de conexão é realizado pela API do Firebase, não sendo necessário configurações primordias, desse modo pode ser utilizada em todos os âmbitos do sistema.

As demais telas do sistema podem ser encontradas na seção de Apêndices.

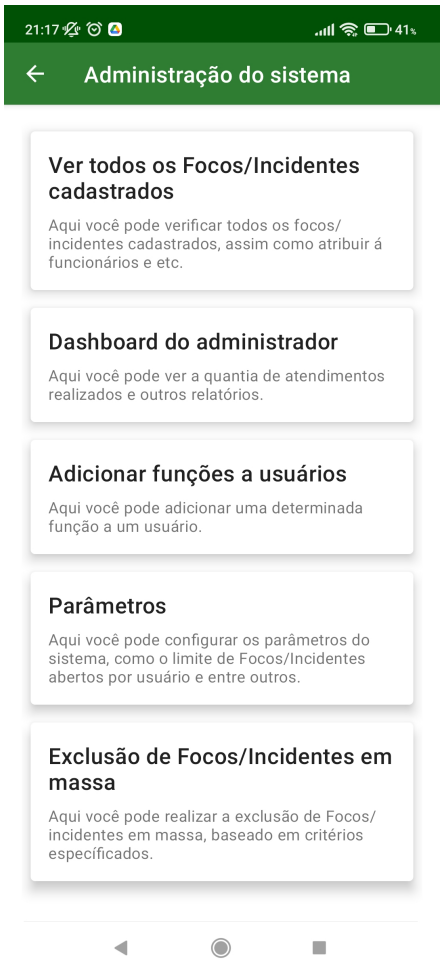
O projeto foi versionado utilizando a ferramenta GIT, do GitHub. Atualmente seu repositório é privado, disponível na conta Github do autor, em: https://github.com/ozebe.

Algoritmo 5: Método responsável por contar a quantia de focos/incidentes abertos nos últimos 7 dias

```
1  private void aplicaFiltroQtdFocosUltimos7Dias() {
2      databaseReference.child("Ponto").addListenerForSingleValueEvent(
3      new ValueEventListener() {
4          @Override
5          public void onDataChange(@NonNull DataSnapshot snapshot) {
6              int qtd = 0;
7              for (DataSnapshot objSnapshot : snapshot.getChildren()) {
8                  Ponto p = objSnapshot.getValue(Ponto.class);
9                  try {
10                     Date dtInclusaoPonto = Globals.getDateFormatdddmmyyyy()
11                     .parse(p.getDataInclusao());
12                     Date dataAtual = new Date();
13
14                     Calendar cal = Calendar.getInstance();
15                     cal.setTime(dataAtual);
16                     cal.add(Calendar.DATE, -7);
17                     Date dataAtualMenos7Dias = cal.getTime();
18                     //se a data de inclusão é depois ou igual a data atual
19                     //menos 7 dias e se a data de inclsão é antes
20                     // ou igual a data atual.
21                     if ((dtInclusaoPonto.compareTo(dataAtualMenos7Dias)) >= 0
22                     && (dtInclusaoPonto.compareTo(dataAtual) <= 0)){
23                         qtd++;
24                     }
25                 } catch (ParseException e) {
26                     Globals.snackbarErro(findViewById(android.R.id.content),
27                     DashboardAdministrador.this,
28                             "Não foi possível gerar o relatório. \n" +
29                             e.getMessage());
30                 }
31             }
32             QtdFocosCadastradosUltimos7Dias.setText(""+qtd);
33         }
34
35         @Override
36         public void onCancelled(@NonNull DatabaseError error) {
37         }
38     });
39 }
```

Fonte: Autoria própria (2021).

Figura 22 – Opções disponíveis na tela de administração do aplicativo na visão do administrador.



Fonte: Autoria própria (2021).

Figura 23 – Tela de inclusão de visita a um foco/incidente, disponível para o funcionário e administrador

Fonte: Autoria própria (2021).

4 CONCLUSÃO

A dengue ainda é um problema sério no Brasil, a forma encontrada pelo autor para auxiliar, por meio da tecnologia, envolve o desenvolvimento de aplicativo que visa facilitar o trabalho do profissional de saúde e também aumentar as taxas de denúncia de possíveis focos de dengue, sendo assim essencial para atuar na diminuição de futuros casos de dengue.

Tendo como base, o objetivo geral do trabalho. Foi desenvolvido um aplicativo que atua como uma interface entre o cidadão e a prefeitura, além de automatizar o trabalho dos agentes de combate a endemias, quanto a um controle mais efetivo sobre as casas e ambientes passiveis de formação de focos do mosquito transmissor da dengue.

Assim os objetivos secundários, contemplam a utilização de API's para a implementação do serviço de geolocalização, armazenamento de dados em banco de dados não relacional. A API utilizada para implementação dos serviços de geolocalização foi o *Fused Location Provider*, da Google, que é capaz de entregar resultados no âmbito da localização atual do usuário, sendo uma API com uma documentação extensa e bastante eficaz. Após isso foi utilizado o *Maps SDK* da Google o qual fornece um ambiente completo para visualização e uso de mapas.

Tendo como resultado final o desenvolvimento de um aplicativo android onde é possível realizar a denúncia e acompanhamento de focos de dengue por parte do usuário, inclusão de visitas a focos de dengue e visitas a residências por parte do funcionário e a extração de relatórios e manutenção do sistema por parte do administrador.

A partir desse desenvolvimento, o aplicativo será disponibilizado para a Prefeitura de Toledo. Podendo assim, em conjunto com a sociedade, prefeitura e tecnologia, diminuir os casos de dengue, considerado um problema crônico não apenas em Toledo mas em todo o Brasil.

4.1 TRABALHOS FUTUROS

O aplicativo pode ser aperfeiçoado para futuros trabalhos que buscam facilitar a eficiência dos serviços públicos da cidade, entre eles é possível destacar:

- Uso de alertas SMS para indicar a mudança de estado para cada fase do foco/incidente
- implementar o serviço de chat para a comunicação entre o usuário e o funcionário.

Referências

AZURE. **Dados não relacionais e NoSQL**. 2018. Disponível em: <https://docs.microsoft.com/pt-br/azure/architecture/data-guide/big-data/non-relational-data>. Acesso em: 26 de setembro de 2021. Citado na página 33.

B., R. N. **Cloud Computing.** The MIT Press, 2016. (The MIT Press Essential Knowledge Series). ISBN 9780262529099. Disponível em: <https://search.ebscohost.com/login.aspx?direct=true&db=e000xww&AN=1238001&lang=pt-br&site=eds-live&scope=site>. Citado na página 15.

BORE, C.; BORE, S. Profiling software api usability for consumer electronics. In: **2005 Digest of Technical Papers. International Conference on Consumer Electronics, 2005. ICCE.** [S.l.: s.n.], 2005. p. 155–156. Citado 2 vezes nas páginas 17 e 32.

BUYYA, R. et al. Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility. **Future Generation Computer Systems**, v. 25, n. 6, p. 599 – 616, 2009. ISSN 0167-739X. Disponível em: <https://search.ebscohost.com/login.aspx?direct=true&db=edselp&AN=S0167739X08001957&lang=pt-br&site=eds-live&scope=site>. Citado na página 15.

CONTI, J. C. da S. K. S. G. Estudo e análise de desempenho de banco de dados relacionais e nosql. **Seminário de Iniciação Científica e Tecnológica da UTFPR; XXV Seminário de Iniciação Científica e Tecnológica da UTFPR**, 2020. Disponível em: <https://search.ebscohost.com/login.aspx?direct=true&db=ir01449a&AN=pecutf.paper.6198&lang=pt-br&site=eds-live&scope=site>. Citado 2 vezes nas páginas 32 e 33.

DEVELOPERS, G. **Fused Location Provider**. 2021. Disponível em: <https://developers.google.com/android/reference/com/google/android/gms/location/FusedLocationProviderClient.html>. Acesso em: 11 de novembro de 2021. Citado na página 17.

DEVELOPERS, G. **Fused Location Provider Diagram**. 2021. Disponível em: <https://developers.google.com/location-context/fused-location-provider>. Acesso em: 11 de novembro de 2021. Citado na página 18.

DEVS, G. **Hamilton App Takes the Stage**. 2017. Disponível em: <https://www.googblogs.com/hamilton-app-takes-the-stage/>. Acesso em: 05 de outubro de 2021. Citado na página 33.

GEOGRAPHIC information — Imagery sensor models for geopositioning — Part 1: Fundamentals. [S.l.], 2018. v. 2018. Citado na página 15.

GOOGLE. **Cloud Storage para Firebase**. 2021. Disponível em: <https://firebase.google.com/docs/storage>. Acesso em: 26 de setembro de 2021. Citado na página 32.

GOOGLE. **Creating a Firebase Project**. 2021. Disponível em: <https://console.firebase.google.com/u/0/>. Acesso em: 16 de outubro de 2021. Citado na página 34.

GOOGLE. **Firebase API Reference**. 2021. Disponível em: <https://firebase.google.com/docs/reference>. Acesso em: 26 de setembro de 2021. Citado na página 31.

GOOGLE. **Firebase Authentication**. 2021. Disponível em: <https://firebase.google.com/docs/auth>. Acesso em: 26 de setembro de 2021. Citado 3 vezes nas páginas 31, 32 e 34.

GOOGLE. **Firebase Pricing**. 2021. Disponível em: <https://firebase.google.com/pricing>. Acesso em: 26 de setembro de 2021. Citado na página 32.

GOOGLE. **Firebase Realtime Database**. 2021. Disponível em: <https://firebase.google.com/docs/database>. Acesso em: 26 de setembro de 2021. Citado 3 vezes nas páginas 31, 32 e 33.

GOOGLE. **Firebase Use Cases**. 2021. Disponível em: <https://firebase.google.com/use-cases>. Acesso em: 26 de setembro de 2021. Citado na página 31.

GOOGLE. **SDK do Maps para Android**. 2021. Disponível em: <https://developers.google.com/maps/documentation/android-sdk/overview>. Acesso em: 01 de outubro de 2021. Citado na página 17.

HUSSAIN, S. et al. Positioning a mobile subscriber in a cellular network system based on signal strength. **IAENG International Journal of Computer Science**, v. 34, 11 2007. Citado 2 vezes nas páginas 18 e 19.

JOSE, B.; ABRAHAM, S. Exploring the merits of nosql: A study based on mongodb. In: **2017 International Conference on Networks Advances in Computational Technologies (NetACT)**. [S.l.: s.n.], 2017. p. 266–271. Citado 3 vezes nas páginas 32, 33 e 34.

KORTUM, P.; ACEMYAN, C. Z. The impact of geographic location on the subjective assessment of system usability. **International Journal of Human-Computer Interaction**, v. 35, n. 2, p. 123 – 130, 2019. ISSN 10447318. Disponível em: <https://search.ebscohost.com/login.aspx?direct=true&db=iih&AN=133653957&lang=pt-br&site=eds-live&scope=site>. Citado na página 16.

MARTIN, R. **API Design for C++.** Morgan Kaufmann, 2011. ISBN 9780123850034. Disponível em: <https://search.ebscohost.com/login.aspx?direct=true&db=e000xww&AN=355589&lang=pt-br&site=eds-live&scope=site>. Citado na página 17.

MEYSMAN, A. **Have you met the Realtime Database?** 2016. Disponível em: <https://dzone.com/articles/nosql-database-types-1>. Acesso em: 06 de outubro de 2021. Citado na página 33.

NADE, G.; ELECTRONIC, T.; PROJECTS. How can startups make use of cloud services. **Electronic Theses, Projects, and Dissertations**, 2021. Disponível em: <https://search.ebscohost.com/login.aspx?direct=true&db=edsbas&AN=edsbas.90881860&lang=pt-br&site=eds-live&scope=site>. Citado na página 15.

PAZ, A. C. M.; LOOS, M. J. A importância da computação em nuvem para a indústria 4.0. 2020. ISSN 1808-0448. Disponível em: <https://search.ebscohost.com/login.aspx?direct=true&db=ir01446a&AN=pdpcdu.article.9317&lang=pt-br&site=eds-live&scope=site>. Citado na página 15.

PUFFELEN, F. van. **Have you met the Realtime Database?** 2016. Disponível em: <https://firebase.googleblog.com/2016/07/have-you-met-realtime-database.html>. Acesso em: 06 de outubro de 2021. Citado na página 32.

REDHAT. **O que é API?** 2021. Disponível em: <https://www.redhat.com/pt-br/topics/api/what-are-application-programming-interfaces>. Acesso em: 26 de setembro de 2021. Citado na página 17.

SAÚDE, M. da. **Monitoramento dos casos de arbovirosesurbanas causados por vírus transmitidospor Aedes (dengue, chikungunya e zika),semanas epidemiológicas 1 a 53, 2020**. Volume 52. Edifício PO700, 7º andarCEP: 70.719-040 – Brasília/DF, 2021. Citado na página 13.

SAÚDE, S. de Estado da. **Situação da Dengue, Chikungunya e Zika Vírus no Paraná**. 2020. Disponível em: <http://www.dengue.pr.gov.br/sites/dengue/arquivos_restritos/files/documento/2020-11/boletimdengue43_2020.pdf>. Acesso em: 02 de outubro de 2021. Citado na página 13.

SMITH, C. S. **Cell Phone Triangulation Accuracy Is All Over The Map**. 2008. Disponível em: <https://searchengineland.com/cell-phone-triangulation-accuracy-is-all-over-the-map-14790>. Acesso em: 17 de novembro de 2021. Citado na página 18.

THE NIST Definition of Cloud Computing. **ACM Queue**, p. 6 – 7, 2010. ISSN 15427730. Disponível em: <https://search.ebscohost.com/login.aspx?direct=true&db=iih&AN=50733041&lang=pt-br&site=eds-live&scope=site>. Citado na página 15.

TOLEDO, P. de. **DENGUE MATA: TOLEDO CONTRA A DENGUE!** 2020. Disponível em: <https://www.toledo.pr.gov.br/noticia/dengue-mata-toledo-contra-a-dengue>. Acesso em: 05 de outubro de 2021. Citado na página 13.

TOLEDO, P. de. **Toledo - Todos contra a dengue**. 2020. Disponível em: <https://www.toledo.pr.gov.br/dengue/>. Acesso em: 02 de outubro de 2021. Citado na página 13.

VEIGA, E. **Por que o Brasil não consegue vencer a dengue?** 2019. Disponível em: <https://www.dw.com/pt-br/por-que-o-brasil-n%C3%A3o-consegue-vencer-a-dengue/a-50450637>. Acesso em: 24 de setembro de 2021. Citado na página 13.

VENAGLIA, G. **Brasil tem quase 1 milhão de casos de dengue em 2020, diz Ministério da Saúde**. 2020. Disponível em: <https://www.cnnbrasil.com.br/saude/brasil-tem-quase-1-milhao-de-casos-de-dengue-em-2020-diz-ministerio-da-saude/>. Acesso em: 24 de setembro de 2021. Citado na página 13.

VERA-OLIVERA, H. et al. Data modeling and nosql databases - a systematic mapping review. **ACM Computing Surveys**, v. 54, n. 6, p. 1 – 26, 2021. ISSN 03600300. Disponível em: <https://search.ebscohost.com/login.aspx?direct=true&db=iih&AN=151906779&lang=pt-br&site=eds-live&scope=site>. Citado 2 vezes nas páginas 32 e 33.

ZETBAITSU, Z. **Compressor - An android image compression library.** 2021. Disponível em: <https://github.com/zetbaitsu/Compressor>. Acesso em: 15 de novembro de 2021. Citado na página 35.

ZHONG, H.; MEI, H. An empirical study on api usages. **IEEE Transactions on Software Engineering**, v. 45, n. 4, p. 319–334, 2019. Citado na página 17.

ZREIK, J.-P. Geo-location, location, location. **Rutgers Computer Technology Law Journal**, v. 45, n. 2, p. 135 – 168, 2019. ISSN 07358938. Disponível em: <https://search.ebscohost.com/login.aspx?direct=true&db=iih&AN=144292730&lang=pt-br&site=eds-live&scope=site>. Citado na página 16.

Apêndices

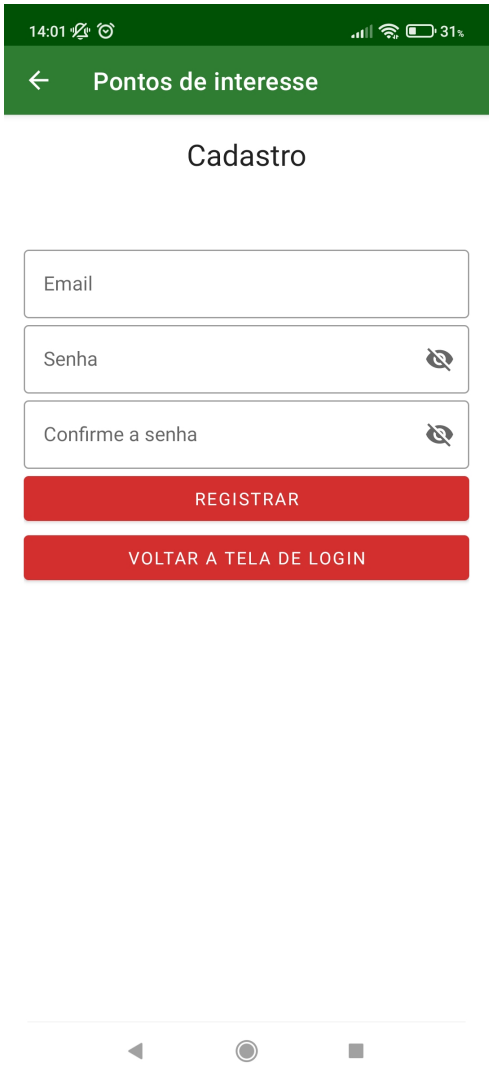
APÊNDICE A – Telas do aplicativo Android

Neste apêndice são ilustradas as principais telas do sistema desenvolvido neste TCC, mostrando as interfaces dos processos de cadastro (Figura 24) e acesso (Figura 25) ao aplicativo, inclusão (Figura 29) e atualização de incidentes/focos, assim como a visualização de tal por parte do usuário (Figura 33), funcionário/administrador (Figura 34), também é mostrado a tela inicial da lista de focos/incidentes na visão do usuário (Figura 30), funcionário (Figura 31) e administrador (Figura 32),além da visualização da lista de histórico de alterações no foco/incidente (Figura 38) e lista de histórico de visitas no foco/incidente (Figura 39), assim como a visualização de uma visita específica ao foco/incidente (Figura 40).

Também são motradas as principais telas de administração, como a tela inicial de administração (Figura 41), a tela de inclusão de funções a usuários (Figura 44), a tela de exclusão de focos/incidentes em massa (Figura 43), a tela inicial da visualização dos relatórios (Figura 42) e também a tela de configuração dos parâmetros do aplicativo (Figura 45).

Todas as imagens são capturas de tela realizadas pelo autor do trabalho a partir da execução do aplicativo.

Figura 24 – Tela de cadastro

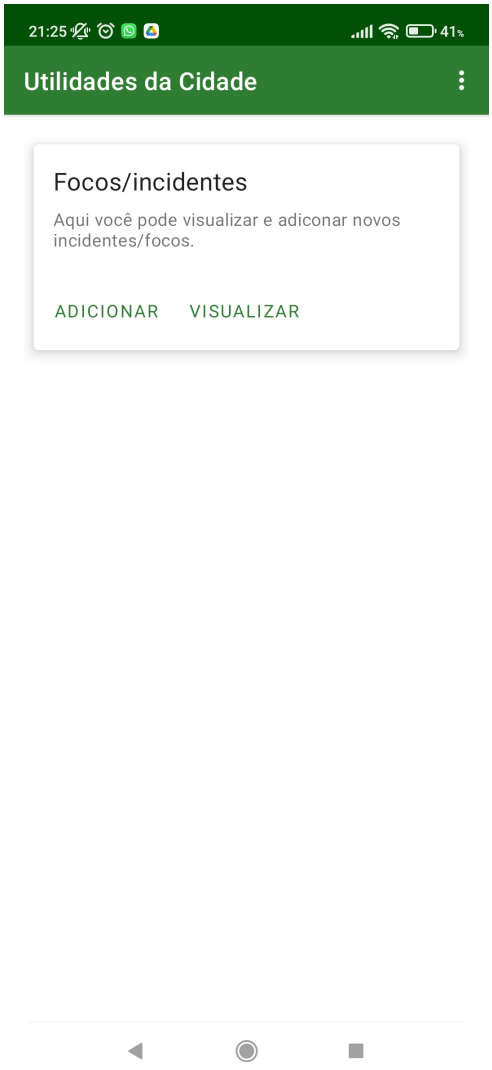


Fonte: Autoria própria (2021).

Figura 25 – Tela de Login

Fonte: Autoria própria (2021).

Figura 26 – Tela Inicial do usuário

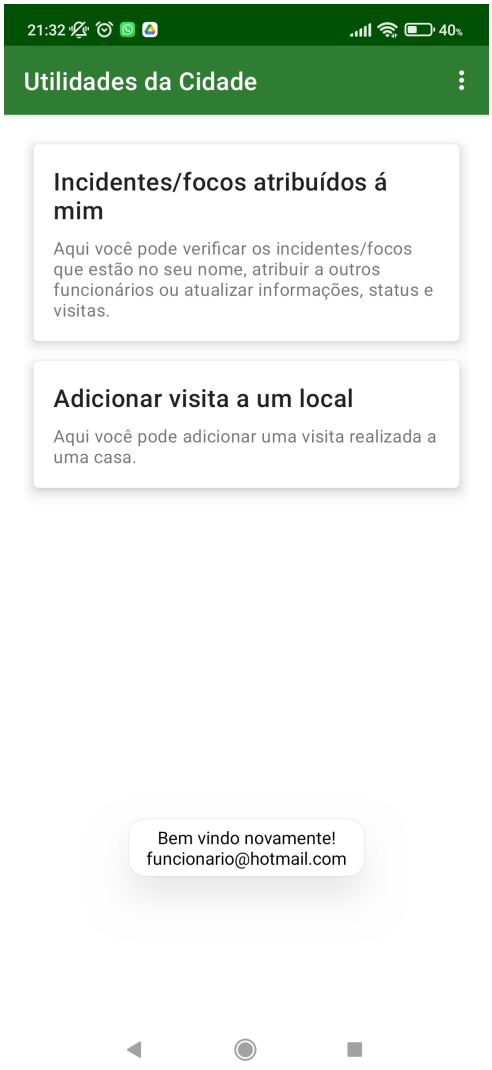


Fonte: Autoria própria (2021).

Figura 27 – Tela Inicial do administrador

Fonte: Autoria própria (2021).

Figura 28 – Tela Inicial do funcionário



Fonte: Autoria própria (2021).

Figura 29 – Tela de inclusão de um Foco/Incidente preenchida

Fonte: Autoria própria (2021).

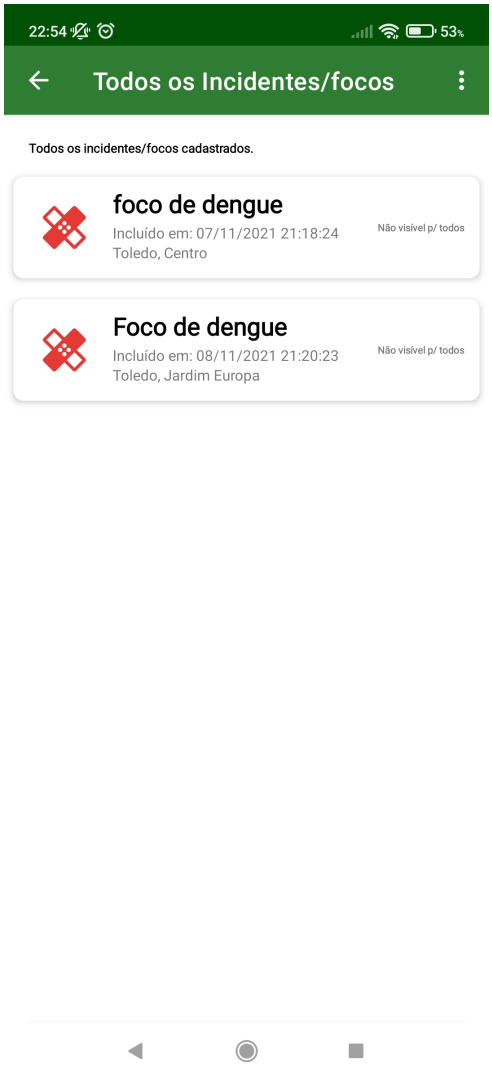
Figura 30 – Tela Inicial de visualização de focos/incidentes cadastrados pelo usuário

Fonte: Autoria própria (2021).

Figura 31 – Tela Inicial de visualização de focos/incidentes na visão do funcionário

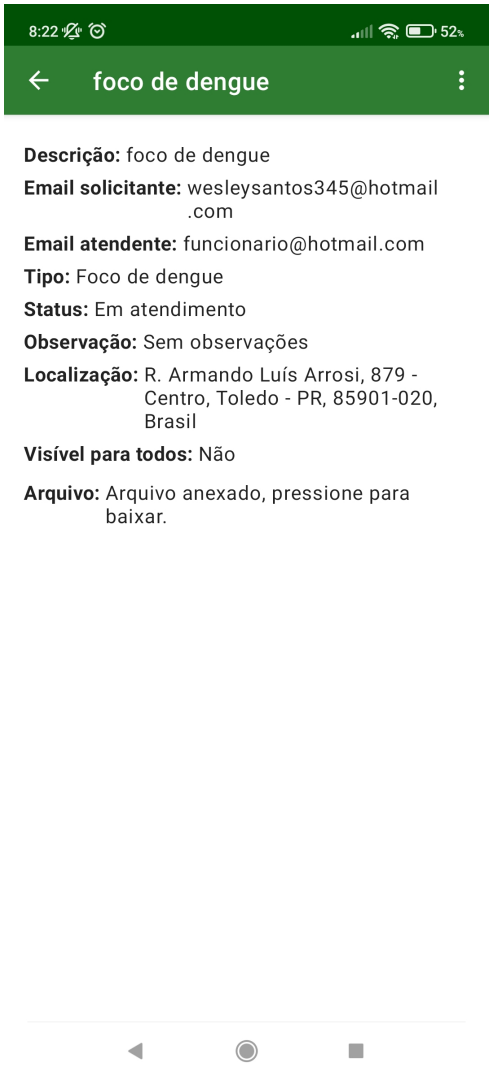
Fonte: Autoria própria (2021).

Figura 32 – Tela Inicial de visualização de focos/incidentes na visão do administrador



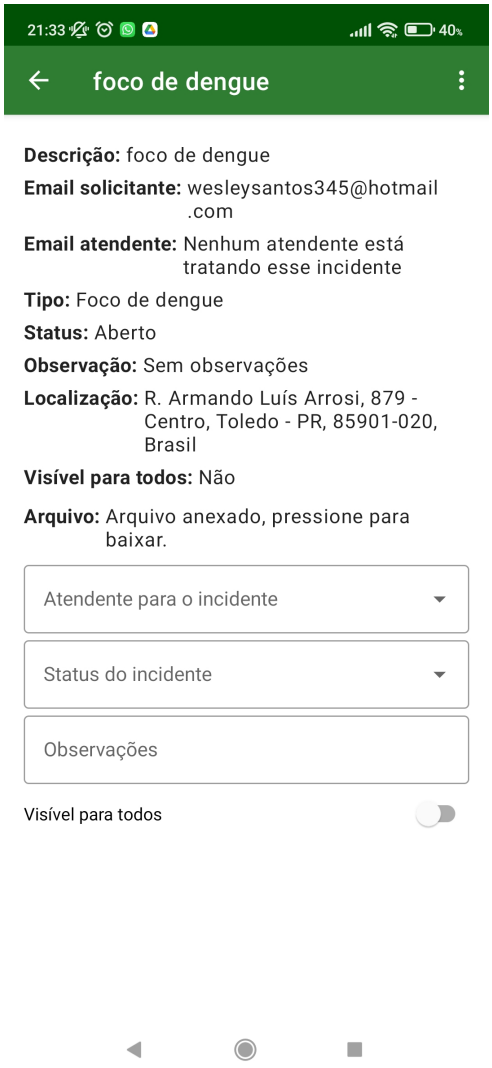
Fonte: Autoria própria (2021).

Figura 33 – Tela de visualização de um foco/incidente cadastrado pelo usuário



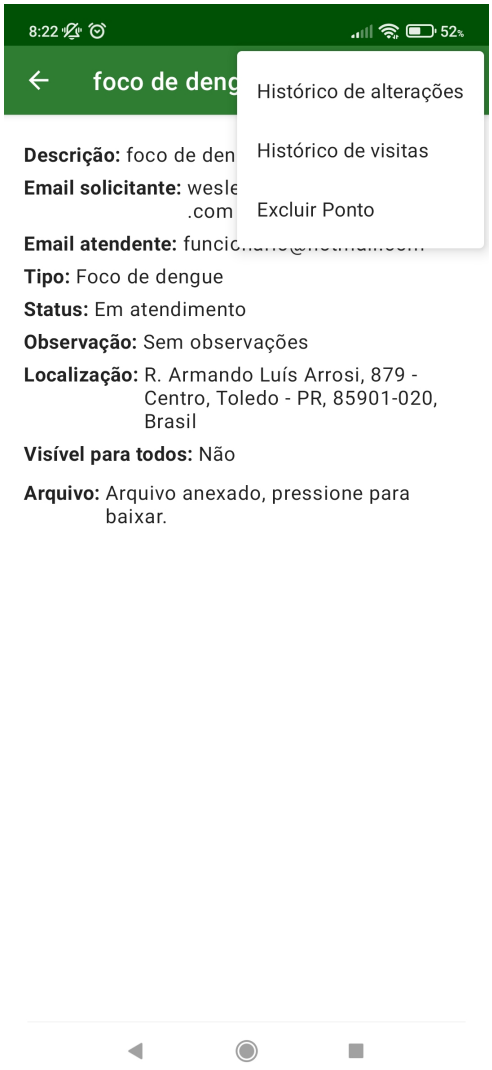
Fonte: Autoria própria (2021).

Figura 34 – Tela de visualização de um foco/incidente na visão do funcionário ou administrador



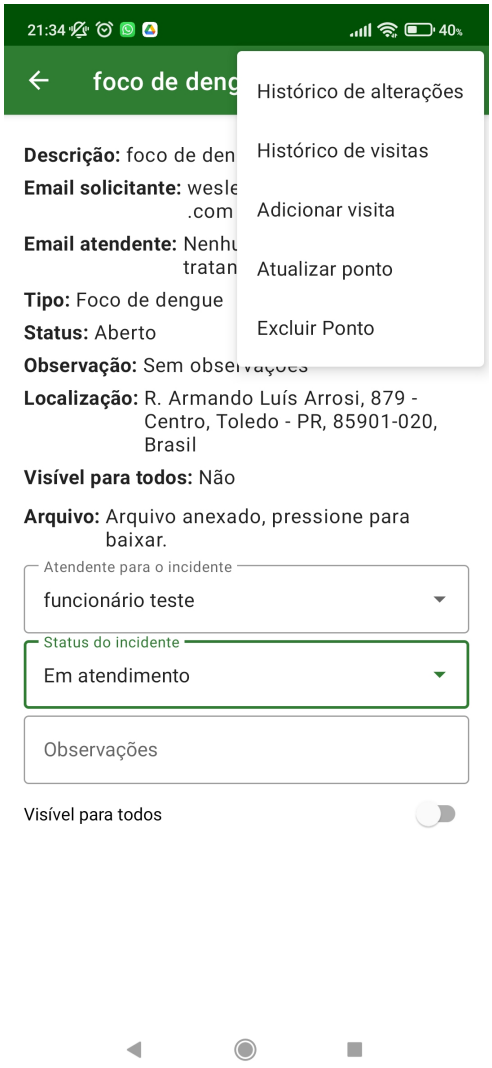
Fonte: Autoria própria (2021).

Figura 35 – Opções disponíveis na tela de visualização de um foco/incidente cadastrado pelo usuário



Fonte: Autoria própria (2021).

Figura 36 – Opções disponíveis na tela de visualização de um foco/incidente na visão do funcionário ou administrador



Fonte: Autoria própria (2021).

Figura 37 – Tela de inclusão de visita a um foco/incidente, disponível para o funcionário e administrador

Fonte: Autoria própria (2021).

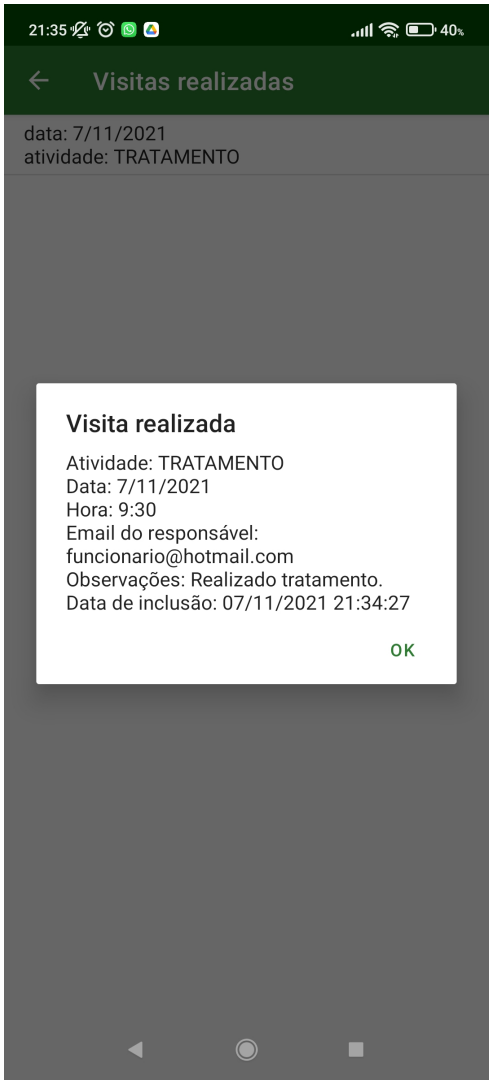
Figura 38 – Tela de visualização de histórico de alterações no foco/incidente, disponível para todos os usuários.

Fonte: Autoria própria (2021).

Figura 39 – Tela de visualização de visitas no foco/incidente, disponível para todos os usuários.

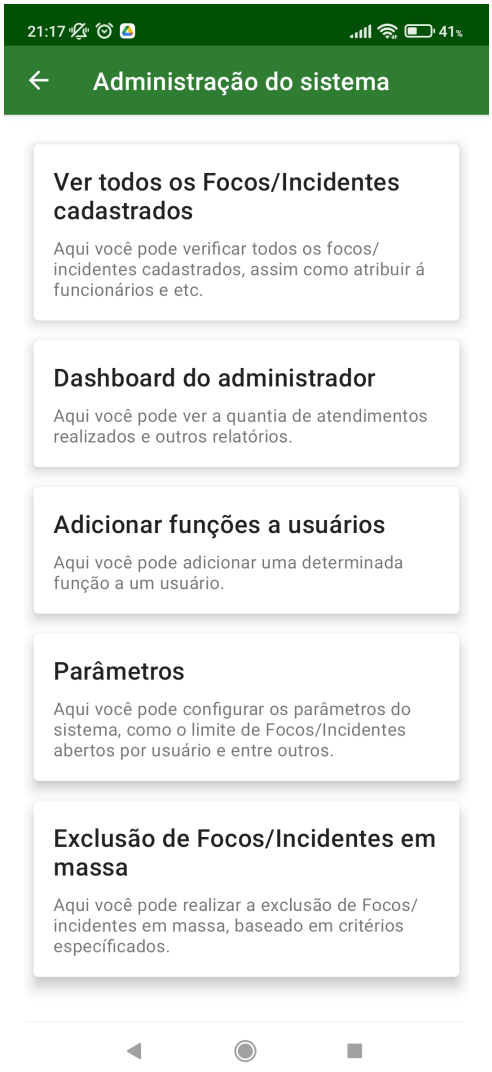
Fonte: Autoria própria (2021).

Figura 40 – Tela de visualização de uma visita específica no foco/incidente, disponível para todos os usuários.



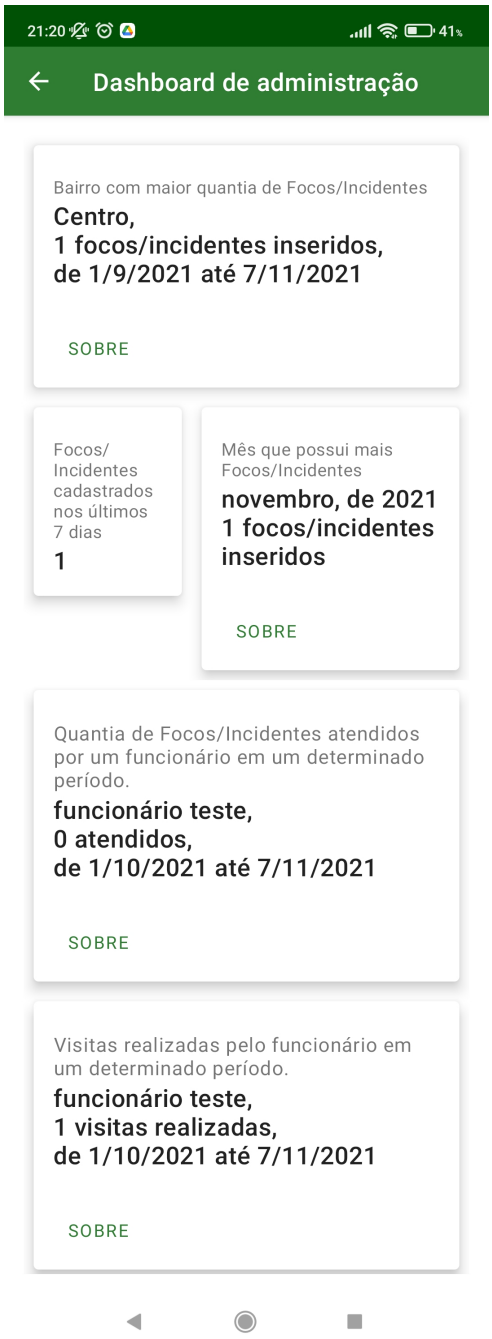
Fonte: Autoria própria (2021).

Figura 41 – Opções disponíveis na tela de administração do sistema na visão do administrador.



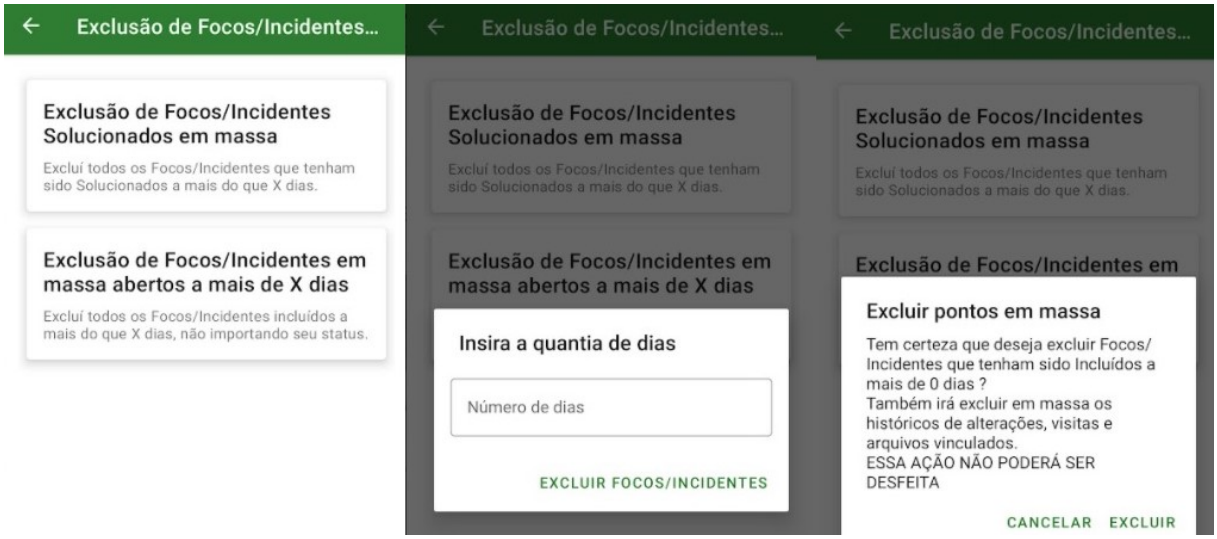
Fonte: Autoria própria (2021).

Figura 42 – Relatórios disponíveis na tela ”Dashboard do administrador”na visão do administrador.



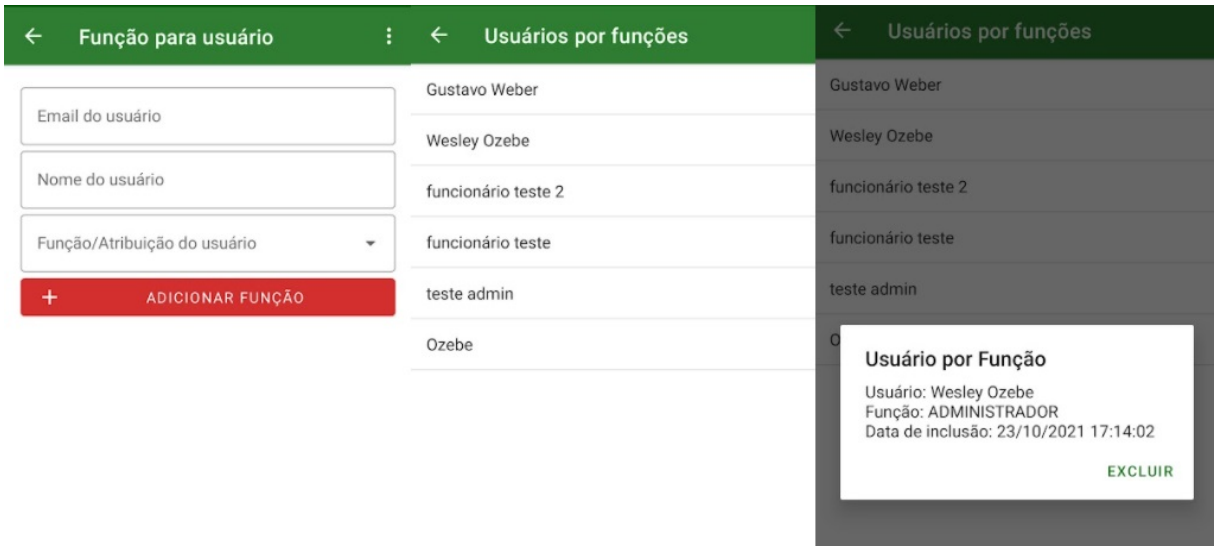
Fonte: Autoria própria (2021).

Figura 43 – Opções disponíveis na tela de exclusão de focos/incidentes em massa, assim como a ação resultante do acionamento da segunda opção, exemplificando o uso pelo administrador.



Fonte: Autoria própria (2021).

Figura 44 – Opções disponíveis na tela de adicionar funções a usuários, assim como a ação resultante do acionamento da visualização dos cadastros, exemplificando o uso pelo administrador.



Fonte: Autoria própria (2021).

Figura 45 – Tela de visualização e edição de parâmetros do sistema, na visão do administrador.

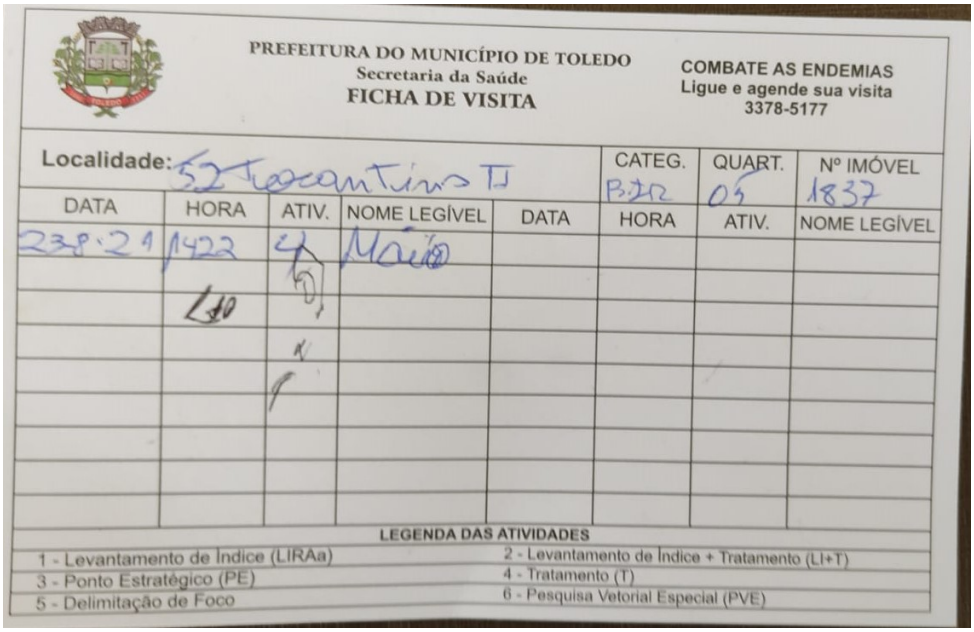
Fonte: Autoria própria (2021).

Anexos

ANEXO A – Ficha de visita de dengue

Ficha de visita de dengue extraída de uma casa onde o agente de saúde passou, sendo possível verificar o que é preenchido e os campos disponíveis.

Figura 46 – Ficha de visita de dengue



PREFEITURA DO MUNICÍPIO DE TOLEDO
Secretaria da Saúde
FICHA DE VISITA

COMBATE AS ENDEMIAS
Ligue e agende sua visita
3378-5177

Localidade: 52 Tocantins TJ | CATEG. B.Z12 | QUART. 05 | Nº IMÓVEL 1837

DATA	HORA	ATIV.	NOME LEGÍVEL	DATA	HORA	ATIV.	NOME LEGÍVEL
23.8.21	1422	4	Mário				

LEGENDA DAS ATIVIDADES

1 - Levantamento de Índice (LIRAa)	2 - Levantamento de Índice + Tratamento (LI+T)
3 - Ponto Estratégico (PE)	4 - Tratamento (T)
5 - Delimitação de Foco	6 - Pesquisa Vetorial Especial (PVE)

Fonte: Autoria própria (2021).