UNIVERSIDADE FEDERAL DE PERNAMBUCO

CENTRO DE INFORMÁTICA

CURSO DE BACHARELADO EM ENGENHARIA DA COMPUTAÇÃO

ANDERSON FELIX DA SILVA

DESENVOLVIMENTO DE APLICATIVO MÓVEL PARA MAPEAMENTO PARTICIPATIVO DOS FOCOS DO AEDES AEGYPTI USANDO GAMIFICAÇÃO

Recife

2018

ANDERSON FELIX DA SILVA

DESENVOLVIMENTO DE APLICATIVO MÓVEL PARA MAPEAMENTO PARTICIPATIVO DOS FOCOS DO AEDES AEGYPTI USANDO GAMIFICAÇÃO

Trabalho apresentado ao Programa de Graduação em Engenharia da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para conclusão do Curso de Engenharia da Computação, orientado pelo Prof. Dr. Wellington Pinheiro dos Santos.

Recife
2018

AGRADECIMENTOS

Agradeço pelo apoio da minha família, especialmente aos meus pais, Ângela e Adenilson, que por todos esses anos sempre estiveram ao meu lado, incentivando-me nos momentos difíceis, principalmente no início do curso.

Também sou grato a minha companheira Camila Bernardo, que teve uma participação importante durante grande parte da minha graduação. Foi a minha principal orientadora nos momentos mais complicados, contribuindo para o meu crescimento profissional e cidadão.

Sou grato a todos os professores os quais contribuíram para que eu chegasse a esse momento, professores do Ensino Médio da ETE Aderico Alves de Vasconcelos e o conjunto de professores da UFPE. Em especial, sou grato ao Prof. Dr. Wellington Pinheiro dos Santos, com quem trabalho desde 2015 e tive o prazer de ser orientado nesse projeto. Agradece-o por sua generosidade e ensinamentos, não só profissionais, mas também humanísticos.

Agradeço a todos os companheiros de trabalho do Núcleo de Tecnologia da Informação da UFPE, onde pude ter os meus primeiros contatos com o ambiente profissional. Agradeço pela receptividade e paciência que tiveram para comigo no início do meu estágio.

Agradeço aos companheiros de curso, e futuros colegas de profissão, por todos os momentos compartilhados.

A todos que direta ou indiretamente fizeram parte da minha formação, o meu muito obrigado.

RESUMO

As arboviroses acometem as Américas desde a era dos descobrimentos. Inicialmente, por meio do vírus da febre amarela, depois pelo vírus da dengue, e agora, diante das alterações climáticas e o crescente número de voos internacionais, dois novos arbovírus tornaram-se endêmicos nas Américas: os vírus Chikungunya e Zika. Essas endemias têm em comum o seu tipo de vetor transmissor, o Aedes aegypti. A prevenção de arbovírus depende prioritariamente da eliminação do vetor: os focos do Aedes aegypti. Isto requer uma atuação estratégica por parte dos órgãos de saúde pública para controle desse vetor. No Brasil, há os Agentes de Controle de Endemias (ACEs), que atuam tanto no controle do vetor transmissor quanto em medidas educativas junto à população. Deste modo, levando em consideração que os ACEs trabalham em regiões pobres e de difícil acesso, como ajudá-los no melhor combate aos focos de mosquitos? Com a popularização dos *smartphones* e, diante dos benefícios da computação móvel a área multiprofissional, aplicações para a área de saúde voltadas ao apoio das atividades dos ACEs, podem contribuir para um maior controle de epidemias como as arboviroses. O objetivo desse trabalho é projetar um aplicativo para dispositivos móveis para mapeamento e vigilância participativa por GPS dos lugares infectados pelo Aedes Aegypti, de forma a fornecer informações para especialistas em saúde pública e os gestores públicos nos municípios de Recife, Olinda, Jaboatão dos Guararapes e Campina Grande, bem como em nível nacional e da OMS OPAS. O aplicativo móvel, intitulado Geo-Saúde, foi desenvolvido utilizando o framework Ionic 3. O Ionic é um framework para desenvolvimento de aplicações para dispositivos móveis de forma rápida e fácil. A ferramenta proposta melhora o fluxo das informações coletadas durante as visitas dos Agentes, contribuindo para uma melhor comunicação entre laboratórios comerciais e estaduais e órgãos de saúde pública. Por conseguinte, ajudando na previsão de fatores facilitadores da propagação da doença e da ocorrência de surtos.

Palavras-chave: Aedes argypti. Zika. Arboviroses. ACEs. Aplicativo móvel. Controle Epidêmico. Geo-Saúde.

ABSTRACT

Arboviruses have been affecting the Americas since the age of discoveries. Initially, through the yellow fever virus, then the dengue virus, and now, in the face of climate change and the growing number of international flights, two new arboviruses have become endemic in the Americas: the Chikungunya and Zika viruses. These endemics have in common their type of transmitting vector, the Aedes aegypti. The prevention of arbovirus depends primarily on the elimination of the vector: the Aedes aegypti outbreaks. This requires a strategic action by the public health organs to control this vector. In Brazil, there are the Endemic Control Agents (ECAs), which act both in the control of the transmitting vector and in educational measures with the population. So, taking into account that ECAs work in poor and hard-to-reach regions, how to help them better fight mosquitoes? With the popularization of smartphones and, given the benefits of mobile computing in the multiprofessional area, health applications aimed at supporting the activities of the ECAs can contribute to a greater control of epidemics such as arboviruses. The objective of this work is to design a mobile application for GPS mapping and participatory surveillance of sites infected by Aedes Aegypti, in order to provide information to public health experts and public managers in the cities of Recife, Olinda, Jaboatão dos Guararapes and Campina Grande, as well as at the national level and the WHO PAHO. The mobile application, titled Geo-Sáude, was developed using the framework Ionic 3. Ionic is a framework for developing applications for mobile devices quickly and easily. The proposed tool improves the flow of information collected during the visits of the Agents, contributing to better communication between commercial and state laboratories and public health agencies. Therefore, helping to predict factors that facilitate the spread of disease and the occurrence of outbreaks.

Keywords: Aedes argypti. Zika Arboviroses. ECAs. Mobile application. Epidemic Control. Geo-Sáude.

ÍNDICE DE FIGURAS

SUMÁRIO

1. INTRODUÇÃO

A dengue ganhou destaque entre as enfermidades reemergentes, sendo considerada a mais importante das doenças virais transmitidas por artrópodes, tornando-se a arbovirose mais difundida do mundo (BRAGA; VALLE, 2007). A dengue manifesta-se, clinicamente, sob duas formas principais: a dengue clássica, chamada de febre de dengue, e a forma hemorrágica, conhecida ora como febre hemorrágica de dengue (FHD), ora como síndrome de choque de dengue (FHD/SCD). Desde a década de 1970, a Organização Mundial da Saúde (OMS) tem atuado ativamente no desenvolvimento e no fomento de estratégias de tratamento e controle da doença [1]. A dengue é transmitida por mosquitos do gênero Aedes, sendo o Aedes aegypti seu principal vetor. O Aedes aegypti é encontrado, principalmente, nos ambientes urbanos, em depósitos de armazenamento de água [2].

Devido às alterações climáticas propícias à dispersão de vetores e suas doenças e ao crescente número de voos internacionais, favoráveis à movimentação de pessoas doentes ou infectadas em período de incubação, vive-se no Brasil a introdução e um rápido processo de dispersão rumo a se tornar endêmicos dois novos arbovírus para as Américas: o vírus Chikungunya, introduzido entre julho e agosto de 2014, após ter entrado no Caribe em dezembro de 2013 e, anteriormente, ter causado grandes epidemias no continente africano e na Ásia desde 2014; e o vírus Zika, possivelmente introduzido no mesmo período durante a Copa do Mundo realizada em 2014 no Brasil [3].

O SUS, Sistema Único de Saúde, é dividido em três níveis de atenção à saúde: primária, secundária e terciária. A atenção primária à saúde é organizada por unidades de saúde da família, por pequenas equipes de médicos, enfermeiros e agentes comunitários de saúde. Os agentes comunitários de saúde são os pilares deste sistema, eles atuam como meios para dar às pessoas acesso ao nível inicial de cuidados de saúde, como também para adquirir informações importantes para os

formuladores de políticas de saúde e especialistas em saúde pública, responsáveis pelo planejamento de cuidados de saúde e controle de epidemias. Por conseguinte, o uso de ferramentas tecnológicas pode agregar rapidez e conforto às atividades dos profissionais da atenção básica de saúde, como também contribuir para uma maior eficácia nas estratégias traçadas pelos gestores e supervisores da saúde pública, ajudando na previsão de surtos endêmicos.

O presente trabalho é parte de um projeto de pesquisa apoiado pela Facepe, British Council, Newton Fund e Institutional Links Vírus Zika, que envolve a colaboração entre a University College London (Reino Unido) e uma equipe do Brasil (UFPE e parceiro da UFCG e UPE), cujo título é *A gamified m-training app for health professional on protocols and participatory surveillance associated with Zika virus*. Este trabalho foi desenvolvido considerando a região metropolitana de Recife, capital do Estado de Pernambuco, Brasil. Essa região é composta por 14 municípios, envolvendo uma população de quase 4 milhões de pessoas. Recife é a cidade mais rica e bem desenvolvida das regiões nordeste e norte do Brasil. Porém, Recife carrega condições climáticas tropicais e apresenta profundas contradições sociais que a tornam mais suscetível a epidemias de arbovírus, como dengue, febre Chikungunya, e principalmente Zika. No entanto, essas doenças transmitidas por vetores permitirão a futura escalabilidade para outras doenças. Os Agentes de Controle de Endemias (ACEs), chamados de Agentes de Saúde Ambiental e Controle de Endemias (ASACEs) em Recife, têm papel fundamental no controle do vetor de doenças como as arboviroses, combatendo os fatores ambientais e assumindo o papel de educadores populares de saúde para, junto à população, realizar ações em centros comunitários e escolas públicas, de forma a contribuir para o aumento da consciência a respeito das arboviroses e outras doenças e para o engajamento de cidadãos e cidadãs no combate ao vetor.

Tendo em vista que o crescimento do mercado de dispositivos móveis tem gerado oportunidades comerciais e sociais em diversas áreas. Esse dispositivo, que é considerado um computador de bolso, possui acesso a milhões de aplicativos. Além disso, na área da saúde, a computação móvel está sendo aplicada em

diversas subáreas. Entre as principais aplicações, destacam-se o monitoramento remoto, o apoio ao diagnóstico e o apoio à tomada de decisão [5].

O objetivo principal deste trabalho é desenvolver um piloto de um aplicativo móvel para auxiliar na tarefa de mapeamento e vigilância participativa das áreas infectadas pelo mosquito Aedes aegypti como possíveis locais de transmissão da Dengue, da febre Chikungunya, e de proliferação do vírus Zika, bem como realizar pré-diagnóstico e fornecer informações para especialistas e gestores da saúde pública nos municípios de Recife, Olinda, Campina Grande e Jaboatão dos Guararapes. Os protótipos do aplicativo móvel serão avaliados em ambientes controlados por um grupo focal, onde os participantes poderão testar a robustez e a aplicabilidade da ferramenta desenvolvida.

Os primeiros beneficiados imediatos do projeto serão os ACEs e ASACEs, que terão ao seu dispor uma ferramenta dinâmica para auxiliar suas atividades diárias. Em segundo lugar, serão beneficiados os especialistas em saúde pública e os governos locais e nacionais, que terão acesso às informações geográficas para melhorar as estratégias de combate ao vírus Zika e a outros arbovírus. Por fim, as mulheres, os bebês e as comunidades locais serão os principais beneficiários, uma vez que o Zika tem consequências mais graves em populações de baixa renda que não possuem acesso aos cuidados básicos de saúde, podendo gerar inclusive paralisia parcial (síndrome de Guillan-Barré) e morte.

Portanto, a presente monografia explica no capítulo de Metodologia as etapas do planejamento, que envolve os estudos sobre os principais atores presentes no sistema de vigilância sanitária e epidemiológica, do levantamento dos requisitos, da arquitetura do sistema, e das tecnologias escolhidas para desenvolver o aplicativo. No capítulo de Resultados são mostrados os principais aspectos da fase de desenvolvimento do aplicativo, bem como as principais funcionalidades presentes no aplicativo desenvolvido. Logo em seguida, no capítulo de Conclusões e Trabalhos Futuros serão mostrados os principais desafios do projeto, as suas contribuições e os próximos trabalhos envolvendo a ferramenta tecnológica criada.

2. METODOLOGIA

2.1. Planejamento

Nesta etapa inicial do projeto, foram realizados encontros com profissionais da área de saúde dos municípios de Recife, Olinda e Campina Grande. Esses encontros foram fundamentais para a definição do escopo do projeto. Diante disso, foram identificados os atores principais envolvidos no sistema de vigilância sanitária e epidemiológica, em seguida, foram levantados os requisitos principais do projeto, definindo a arquitetura do projeto e, por fim, especificando alguns casos de uso para o aplicativo desenvolvido.

2.1.1. Identificação dos principais atores

O sistema de vigilância sanitária e epidemiológica considera três tipos de perfis profissionais no controle dos vetores: coordenador, supervisor e agente de saúde.

Esses profissionais têm responsabilidades diferentes e interdependentes. O conhecimento sobre essas responsabilidades foi muito importante para definirmos a estrutura do projeto. De modo a esclarecer os papeis de cada profissional, os tópicos abaixo elencam as principais atribuições sobre eles.

Atribuições do Coordenador:

- Buscar apoio e sustentabilidade para realização do Levantamento de Índices Rápidos do Aedes aegypti (LIRAa);
- Estratificar e calcular o número de imóveis a pesquisar;

- Definir os estratos (são os bairros), considerando-se as características socioambientais;
- Definir os quarteirões a serem trabalhados, utilizando-se do sistema informatizado disponibilizado;
- Definir os recursos humanos que estão disponíveis: supervisores, agentes, laboratoristas e digitadores;
- Definir os materiais necessários: veículos, equipamentos e material de campo (componentes da bolsa de trabalho de campo);
- Definir os laboratórios de apoio;
- Planejar ações de supervisão durante a realização do LIRAa;
- Digitar os resultados no sistema;
- Analisar dados e elaborar relatório final;
- Planejar ações necessárias, após o levantamento, para as áreas consideradas mais críticas: operações de campo, ações de informação, educação e comunicação social.

Atribuições do Supervisor:

- Organizar e distribuir os agentes na área de trabalho;
- Abastecer os agentes de saúde com os insumos necessários;
- Supervisionar as atividades dos agentes de saúde;
- Receber e conferir os boletins: Boletim de Campo e Laboratório (BCL);
- Consolidar os dados no BCL e preencher o Resumo Parcial (RP);
- Encaminhar ao laboratório os BCL e resumos parciais com as amostras coletadas;
- Enviar ao setor de digitação o BCL e RP por estrato;

Atribuições do Agente de Saúde:

- Visitar de 20 a 25 imóveis por dia;
- Realizar minuciosa pesquisa larvária nos imóveis definidos no estrato;

- Coletar e preencher os rótulos dos tubitos (tubos de ensaio para coleta de larvas);
- Registrar as informações no formulário BCL;
- Repassar, ao final do dia, o BCL devidamente preenchido ao supervisor.

2.2. Proposta

Nesta etapa do projeto, a partir do conhecimento sobre os atores envolvidos e dos seus respectivos contextos de atuação, foram definidos os conteúdos a serem abordados na aplicação móvel, bem como, os módulos e as tecnologias utilizadas para o desenvolvimento de cada um deles.

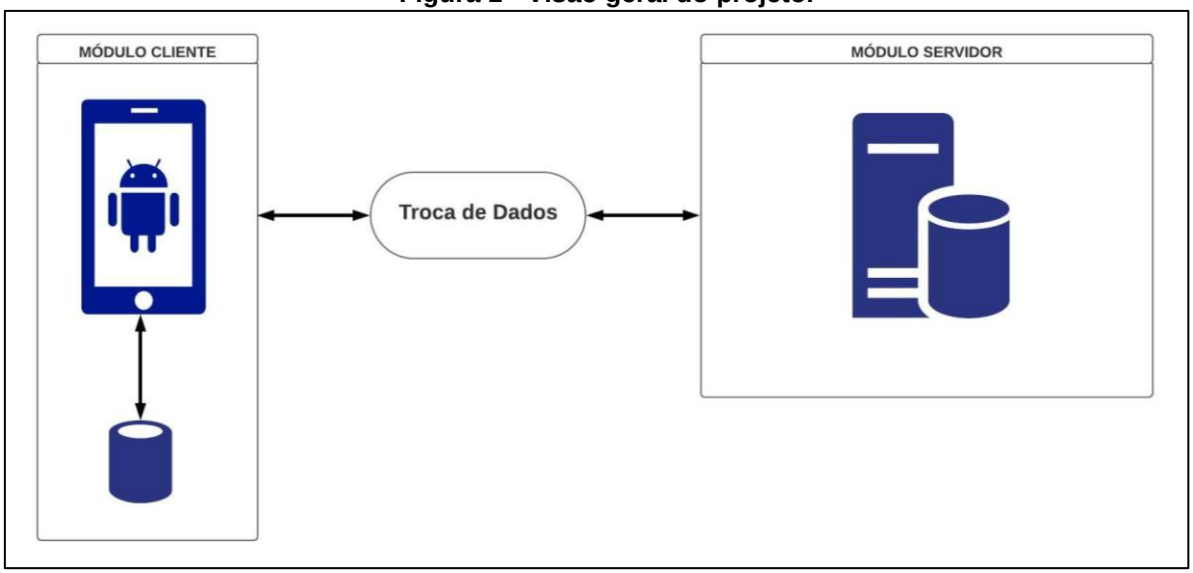
Os Agentes de Controle de Endemias realizam vistorias de residências, depósitos, terrenos baldios e estabelecimentos comerciais para buscar focos endêmicos. As informações coletadas durante as visitas são anotadas em formulários de papel. Os formulários preenchidos irão ser entregues aos supervisores, para análise e consolidação dos dados coletados. Em seguida, esses dados serão encaminhados ao setor de digitação. Todo esse fluxo impede que sejam tomadas decisões, de forma célere, para casos emergências.

Isto posto, de forma a aumentar a celeridade do fluxo das informações coletadas durante as visitas, foi decidido a criação de dois módulos no projeto completo: módulo cliente e módulo servidor. O modulo cliente tem suas funcionalidades focadas às atividades dos agentes de saúde, ele consiste em um aplicativo móvel para plataforma Android; o módulo servidor tem suas funcionalidades voltadas às atividades dos supervisores e coordenadores, ele consiste de um website responsivo.

Na Figura 1, temos uma visão geral do projeto completo. Entretanto, neste trabalho o foco é detalhar o módulo cliente, o aplicativo móvel para plataforma

Android. Não obstante, é importante pontuar que há outros módulos envolvidos no fluxo dos dados coletados.

Figura 1 - Visão geral do projeto.



Fonte: Figura criada pelo autor.

2.2.1. Módulo cliente: aplicativo móvel

O aplicativo, nomeado de Geo-Saúde, tem seu foco voltado às atividades do Agente de Saúde, no caso, o Agente de Controle de Endemias. O Geo-Saúde tem como objetivo substituir as versões impressas do Formulário de Visita Diária, ou Boletim de Campo e Laboratório, voltado ao controle de arboviroses, utilizados pelos ACEs (Agentes de Controle de Endemias). O aplicativo contém dois tipos de formulários: LIRAa e PSA. O formulário LIRAa – Levantamento Rápido de Índices para Aedes Aegypti – tem a vantagem de apresentar, de maneira rápida e segura, os índices de infestações larvários para o mosquito do gênero Aedes aegypti, podendo redirecionar e/ou intensificar algumas intervenções, ou ainda, alterar as estratégias de controle adotadas. Já o formulário de visita diária PSA, tem como objetivo coletar informações tanto sobre o mosquito Aedes aegypti quanto sobre o

mosquito Cúlex. Foi decidido utilizar um banco de dados local no aplicativo, com a finalidade de garantir uma persistência local dos dados quando não houver conexão com a internet.

Para construção do Geo-Saúde, foi decidido utilizar o *framework* Ionic. Essa escolha foi realizada tomando como base as duas principias características do Ionic: possibilidade de desenvolver aplicações híbridas e o uso de tecnologias web no desenvolvimento (sendo essa a característica principal para a adoção desse *framework*). O desenvolvimento de aplicações híbridas promove a interoperabilidade entre os principais sistemas operacionais móveis disponíveis na atualidade: Android e iOS. Portanto, o desenvolvedor poderá construir o aplicativo utilizando um único código-fonte, que será processado e compilado em diferentes linguagens de programação. O desenvolvimento de aplicativos móveis no Ionic, é realizado baseando-se em elementos do padrão de arquitetura de software MVC (Model-View-Controller), por meio das linguagens CSS (*Cascading Style Sheets*), HTML (*HyperText Markup Language*) e TypeScript. Assim, pessoas com conhecimento em desenvolvimento web, podem construir um aplicativo utilizando uma abordagem híbrida.

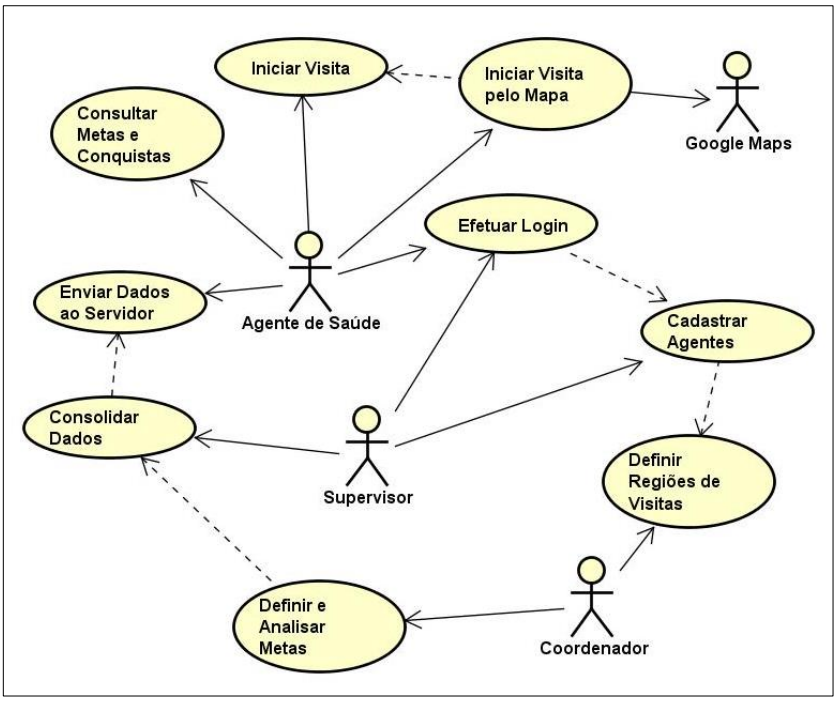
2.2.2. Módulo servidor: website responsivo

O módulo do servidor, um website responsivo, aborda todas as atividades referentes aos Coordenadores e Supervisores dos Agentes de Saúde. Por meio desse website, eles realizam o cadastro de novos Agentes, realizam a alocação de cada Agente nas regiões distritais as quais eles irão trabalhar. Esse lado do servidor, responde ainda pelo armazenamento de todos os dados coletados pelos agentes durante as visitas. Após o agente realizar a coletas das informações e retornar a um ponto que tenha internet, os dados serão enviados ao servidor. Esses dados serão processados e gerarão os relatórios consolidados. Os arquivos dos consolidados gerados serão encaminhados aos coordenadores para análise e tomada de decisão, caso haja necessidade.

2.2.3. Descrição dos casos de uso para o aplicativo Geo-Saúde

Nesta sessão, vamos abordar os principais casos de uso relacionados ao aplicativo Geo-Saúde. Os casos de uso foram criados, buscando esclarecer mais precisamente cada funcionalidade do aplicativo. Na Figura 2, podemos ver o Diagrama de Caso de Uso Geral, ilustrando as principais atividades para cada ator envolvido.

Figura 2 - Diagrama de caso de uso geral.



Fonte: diagrama criado no software astah UML pelo autor.

Como o foco desse projeto é no módulo cliente, vamos detalhar cinco casos de usos atrelados ao Agente de Saúde.

UC01: Login do agente de saúde	
Descrição Sumária	Este caso de uso é responsável por realizar o login do agente no aplicativo Geo-Saúde.
Atores	Agentes de saúde.
Pré-condições	O agente estar cadastrado no sistema.
Pós-condições	Agente conseguiu entrar no aplicativo.
Fluxo Principal: 1. Agente abre o aplicativo Geo-Saúde; 2. Digita o nome do usuário e a senha; 3. Clica em entrar; 4. Verifica se o usuário existe no sistema do supervisor; 5. Se existir, persiste o login do usuário localmente; 6. Realiza o login.	
Fluxo Secundário: • No passo 3, caso o usuário deixe algum campo em branco, exibe uma mensagem para exigir o preenchimento do respectivo campo; • No passo 4, caso não haja um cadastro do usuário no sistema do supervisor, será exibida uma mensagem informando sobre essa inexistência.	

UC02: Iniciar anotações sobre as visitas	
Descrição Sumária	Este caso de uso é responsável por descrever o fluxo para o agente iniciar as anotações sobre as visitas.
Atores	Agentes de Saúde.
Pré-condições	O agente ter entrado no aplicativo.
Pós-condições	Informações coletas e armazenadas no banco de dados local ou remoto.
Fluxo Principal: 1. Agente acessa a aba de Atividades; 2. Escolhe a atividade de Iniciar Visita; 3. Clica em um dos estratos disponíveis; 4. Verifica se o endereço informado pelo GPS está correto. Caso não, deve-se corrigir o endereço a ser visitado; 5. Escolher o formulário a ser preenchido; 6. Acessa cada campo do formulário clicando sobre cada um deles; 7. Preenche as informações dos respectivos campos; 8. Clica em Finalizar Visita e confirma a finalização apertando em sim; 9. Caso haja conexão com a	

internet, os dados são enviados ao servidor; 10. Exibe uma mensagem informando sobre a finalização da visita e envio dos dados ao servidor.	
Fluxo Secundário: 1. No passo 9, caso não haja conexão com a internet, os dados são armazenados em um banco de dados local.	

UC03: Iniciar uma visita a partir da tela de mapa	
Descrição Sumária	Este caso de uso é responsável por descrever o fluxo para o agente iniciar as anotações sobre as visitas a partir da tela de mapa.
Atores	Agentes de Saúde.
Pré-condições	O agente ter entrado no aplicativo.
Pós-condições	Informações coletas e armazenadas no banco de dados local.
Fluxo Principal: 1. Agente acessa a tela de Mapa; 2. Navega pelo Mapa e escolhe um ponto que ainda não foi visitado; 3. Verifica se o endereço do imóvel	

a ser visitado está correto; 4. Escolhe o formulário a ser preenchido; 5. Acessa cada campo do formulário clicando sobre cada um deles; 6. Clica em Finalizar Visita e confirma a finalização apertando em sim; 7. Caso haja conexão com a internet, os dados serão enviados ao servidor.	
Fluxo secundário: • No passo 7, caso não haja conexão com a internet, armazena os dados localmente.	

UC04: Acesso às metas e conquistas	
Descrição Sumária	Este caso de uso é responsável por mostrar o acesso à tela de conquista.
Atores	Agentes de Saúde
Pré-condições	O agente ter entrado no aplicativo.
Pós-condições	O agente saberá as metas atingidas e as metas ainda a atingir.
Fluxo Principal: 1. Agente acessa a aba Conquistas; 2. Visualiza os gráficos que exibem	

um resumo sobre cada meta do agente.	

UC05: Enviar dados ao servidor	
Descrição Sumária	Este caso de uso é responsável por mostrar o envio de dados ao servidor.
Atores	Agentes de Saúde
Pré-condições	O agente ter entrado no aplicativo. O aplicativo ter dados no banco local.
Pós-condições	Os dados serem enviado ao servidor e serem removidos do banco de dados local.
Fluxo Principal: 1. Agente acessa a aba de Atividades; 2. Clica na opção de Enviar Dados; 3. Os dados serão obtidos do banco de dados local e enviados ao servidor; 4. Os dados serão removidos do banco de dados local; 5. Exibe mensagem confirmando o envio ao servidor.	

2.3. Tecnologias Utilizadas

Nessa seção iremos descrever as principais características das tecnologias utilizadas para o desenvolvimento do aplicativo. Essas tecnologias são pré-requisitos para o desenvolvimento do aplicativo móvel.

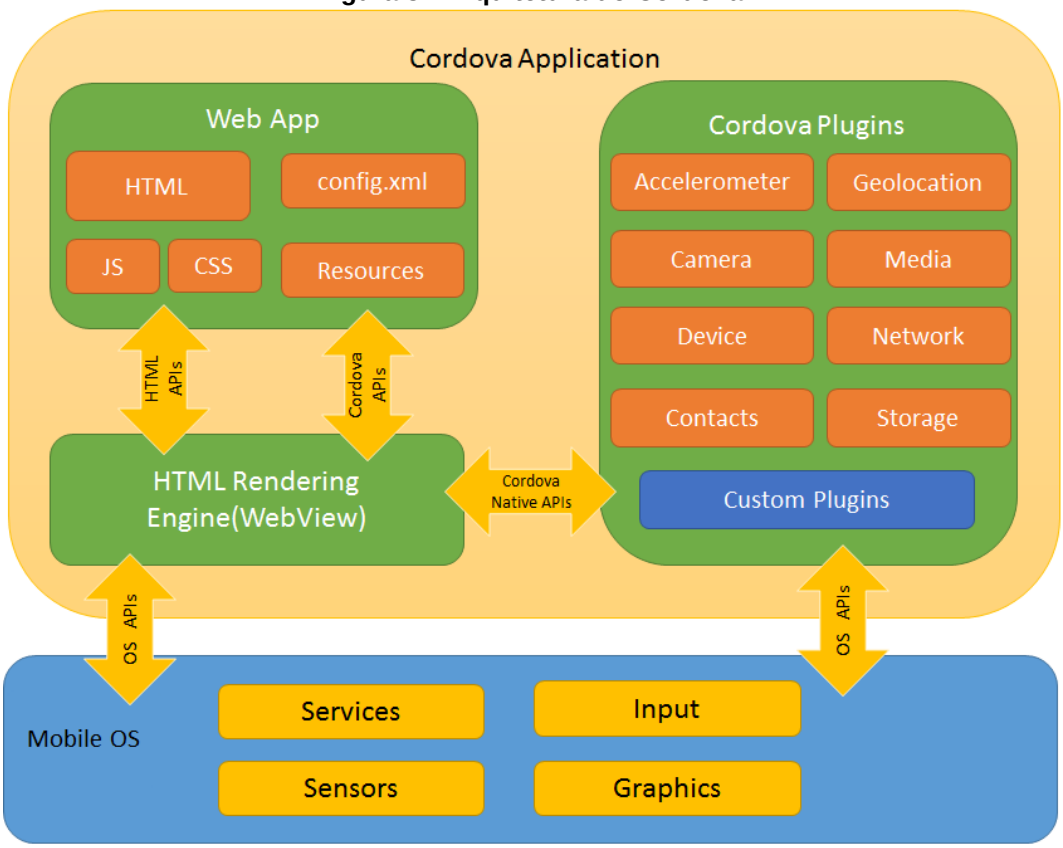
2.3.1. *Framework* Ionic 3

Ionic é uma plataforma gratuita para desenvolvimento de softwares para sistemas móveis, ele fornece uma estrutura bastante prática, acessível e fácil para o desenvolvimento de aplicações multiplataformas. O Ionic utiliza o padrão de arquitetura de software MVC (modelo, visão e controle). Esse padrão separa a aplicação em 3 camadas: a camada de interação do usuário (view), a camada de manipulação dos dados (model) e a camada de controle (controller). A escolha para o uso do Ionic nesse projeto deve-se a pequena curva de aprendizagem para utilizá-lo, uma vez que ele utiliza tecnologias web: HTML, CSS e TypeScript, aos quais são responsáveis pela estrutura, estilização e ações do aplicativo [7]. Além do mais, o Ionic apresenta em sua estrutura, duas tecnologias chaves: Apache Cordova e Angular.

2.3.2. Apache Cordova

É uma tecnologia que faz a integração do aplicativo - que por ser uma aplicação web não tem acesso diretamente ao hardware - aos recursos nativos dos diferentes dispositivos móveis, viabilizando o acesso do aplicativo aos recursos do aparelho no qual está instalado, como por exemplo, o GPS, câmera e etc [7]. Portanto, o *Apache Cordova* é utilizado pelo Ionic para acessar as funcionalidades nativas dos aparelhos móveis, como o acelerômetro, sensor de impressão digital e geolocalização [8].

Figura 3 - Arquitetura do Cordova.



Fonte: https://cordova.apache.org/docs/en/latest/guide/overview/index.html.

A Figura 3 mostra os principais componentes da arquitetura do Cordova. São eles:

- WebView

Componente que fornece um mecanismo de renderização para o aplicativo. A webview é um tipo especial de “browser”, contendo os recursos necessários para que o HTML, CSS e JavaScript funcionem junto ao sistema operacional móvel. Por conta disto, os aplicativos desenvolvidos sobre o cordova, são ditos híbridos; onde a parte relacionada ao uso de recursos do sistema é desenvolvida com código nativo, e a outra parte, relacionada ao uso de recursos visuais é feita com código não

nativo. Além disso, faz uso de uma interface JavaScript, que possibilita o acesso aos recursos nativos do dispositivo móvel, como a câmera, microfone e o acelerômetro.

- Web App

Esse componente é o responsável por concentrar os arquivos contendo todo o código web desenvolvido: HTML, CSS e JavaScript. Como qualquer página web, há o arquivo padrão *index.html* referenciando o CSS, JavaScript, *images*, arquivos de mídia e outros arquivos necessários para execução. A aplicação web é executada sobre uma WebView. Vale ressaltar a importância do arquivo *config.xml*, que oferece informações sobre o aplicativo e especifica os parâmetros que afetam o seu funcionamento, como exemplo, no que se refere a mudança de orientação da tela [8].

- Plugins

Os Plugins são os responsáveis por estabelecer uma interface entre o Cordova e os componentes nativos do dispositivo móvel, por meio deles é possível se comunicar com as *APIs* do sistema operacional. O *Core Plugins* é um conjunto de plugins utilizado pelo Cordova para acessar os recursos do dispositivo móvel, como bateria, câmera, contatos e etc. O Cordova também permite o uso de plugins customizados, assim você pode integrar plugins externo ao seu projeto [8].

2.3.3. Angular

É um *framework* voltado para criação de aplicações web modernas, tomando como base a característica de uma página *HTML5* que permite trabalhar com conteúdo dinâmico (também chamado de *Single Page Applications* ou *SPAs*), utilizando ligação direta e bidirecional dos dados (*two-way data-binding*) que permite sincronização imediata entre *models* e *views*, tornando a aplicação mais rápida, pois o carregamento da *view* é feito apenas uma vez, modificando o conteúdo que é exibido nas páginas de forma dinâmica, não precisando alocar uma nova página

[10]. O Angular é utilizado pelo Ionic para controlar os objetos e configurar as telas do aplicativo móvel [9].

Devido à facilidade em encapsular certos comportamentos e regras da interface, o Angular adotou em suas aplicações, o modelo baseado em componentes. Esse modelo também permite que um componente possa encapsular outros componentes, contribuindo para a reutilização de um componente em vários outros lugares da aplicação. Cada componente do Angular é composto de três itens principais: template HTML, CSS e TypeScript (classe para gerenciamento do componente).

2.3.4. Node.js

É um interpretador de código JavaScript com o código aberto que permite construir de forma leve e fácil, aplicações de rede, rápidas e escaláveis. O Node.js possui um dos maiores gerenciadores de pacotes, o NPM (Node Package Manager) [11]. Essa tecnologia, NPM, foi utilizada para instalar alguns pacotes necessários para o desenvolvimento do aplicativo móvel. Inclusive, o próprio Ionic foi instalado a partir do NPM.

Foi necessário, utilizando o NPM, instalar os seguintes pacotes/módulos:

- *Framework* Ionic versão 3 – é a ferramenta principal, onde ficou concentrado todo o desenvolvimento da aplicação;
- *Geolocation* - módulo utilizado para captura das coordenadas geográficas, latitude e longitude; permitindo acessarmos o GPS do dispositivo móvel, de forma a contribuir com o requisito de geolocalização; também foi por meio do *Geolocation* que utilizamos a ferramenta de mapas da Google;
- *Storage* - módulo responsável por acessar os recursos de armazenamento de dados do *smartphone*, viabilizando a persistência

de toda informação que transita pelo aplicativo, desde informações sobre o login até as informações coletadas pelos agentes de saúde;

- *Sqlite* – módulo instalado de forma a dar uma maior assistência ao módulo *Storage*, no que concerne à priorização dos modos de armazenamento, assim, o *Sqlite* foi definido como o armazenamento local prioritário;
- *Http* – esse módulo é fundamental para a comunicação com o servidor, utilizando o protocolo HTTP, permitindo a troca de dados entre cliente e servidor;
- *Network* – possui a responsabilidade de monitorar quando há conexão com internet no dispositivo móvel.

2.3.5. Android SDK

O Android SDK – Kit de Desenvolvimento de Software para Android – fornece ferramentas para auxiliar no desenvolvimento de aplicativos para todas as versões dos sistemas operacionais Android. Um importante benefício ao utilizar o Android SDK, é a possibilidade de realizar testes do aplicativo, seja em um software emulador do android ou em um *smartphone* com o sistema android.

2.3.6. Google Chrome

O Ionic permite utilizar o navegador *Google Chrome* para realizar testes de maneira local. Ao executarmos o comando *ionic serve*, o Ionic inicia um servidor web local com o nosso aplicativo em execução, abrindo o navegador logo em seguida. Dessa forma, poderemos utilizar as ferramentas de *debug* do navegador para detectarmos problemas e resoluções de problemas. No *Google Chrome*, essas ferramentas podem ser acessadas ao clicarmos na tecla F12. Dentre as funcionalidades providas pelas ferramentas de desenvolvimento do navegador *Google Chrome,* destacam-se: edição de qualquer elemento do código do projeto,

acesso aos recursos da aplicação (armazenamento de dados e cookies, por exemplo) e monitoramento de requisições AJAX (*Asynchronous Javascript and XML*) [12].

3. RESULTADOS

Nesta seção são abordados os resultados obtidos a partir da fase de desenvolvimento do aplicativo Geo-Saúde. São mostrados os passos iniciais de preparação do ambiente de desenvolvimento, bem como as funcionalidades presentes em cada parte do aplicativo desenvolvido.

3.1. Desenvolvimento do Aplicativo

Após a definição do Ionic como plataforma de desenvolvimento, o próximo passo foi preparar o ambiente para o início do desenvolvimento de um protótipo do aplicativo Geo-Saúde. Nesta etapa também avaliamos a estrutura de cada tela do aplicativo, verificando os layouts e componentes disponíveis pelo Ionic. Por fim, devido à necessidade de manter a persistência dos dados coletados pelos agentes de saúde, definimos os modos para armazenamento local e remoto dos dados.

3.1.1. Preparando o Ambiente

O Ionic pode ser instalado em Windows, Linux ou Mac. Todos os comandos do Ionic são realizados diretamente no terminal por meio de CLI (*Command-Line Interface*) chamado de Ionic CLI. O Node.js é a tecnologia responsável por dar suporte ao Ionic CLI para executar suas tarefas. Portanto, antes de instalar o Ionic, foi necessário instalar o Node.js e o Github para controle de versão. Em seguida, o Ionic foi instalado por meio do console de comandos do Node.js. Finalmente, foi instalado o editor de código Visual Studio Code, onde foi escrito todo o código da aplicação.

3.1.2. Aplicativo Geo-Saúde

O aplicativo Geo-Saúde foi desenvolvido utilizando a tecnologia para construção de aplicações móveis multiplataformas, o Ionic, que por sua vez, disponibiliza vários recursos de estilização e estrutura, como campos de texto, botões, ícones e cartões estilizados que tornam mais fácil o trabalho do desenvolvedor. No site do *framework*, são informados todos os passos iniciais necessários para instalação e configuração do projeto. Por meio de comandos executados no terminal, é gerado um novo projeto, com as pastas já criadas automaticamente, ficando disponível toda a estrutura necessária para trabalhar com os recursos do HTML, CSS e TypeScript. Em virtude de o Ionic ter como base o Angular, o desenvolvimento de aplicativos assimila-se ao desenvolvimento de um *website*, essa semelhança é bem evidente na estrutura de páginas e diretórios. No entanto, o Ionic comparado ao Angular, utiliza alguns componentes a mais, o que deixa o aplicativo com a aparência mais nativa para o sistema operacional móvel ao qual ele está sendo desenvolvido. Isto também impacta na forma como o Ionic utiliza a página do navegador para executar o aplicativo, a página reorganiza os componentes, deixando o navegador semelhante ao sistema nativo do aplicativo. O resto da página do navegador fica em branco, onde se podem utilizar as ferramentas de debug de código disponibilizadas pelo navegador, de forma a detectar algum problema.

- **Funcionalidades e Telas do Aplicativo**

O Geo-Saúde é um aplicativo piloto que possui as funcionalidades necessárias para auxiliar nas atividades dos ACSs e ACEs, além disso, ele utiliza métodos dinâmicos para acompanhamento das atividades, exibindo de maneira interativa as metas diárias/semanais/mensais para cada agente. Um dos primeiros desafios para estruturar as telas, foi decidir a forma de representá-las e quais as

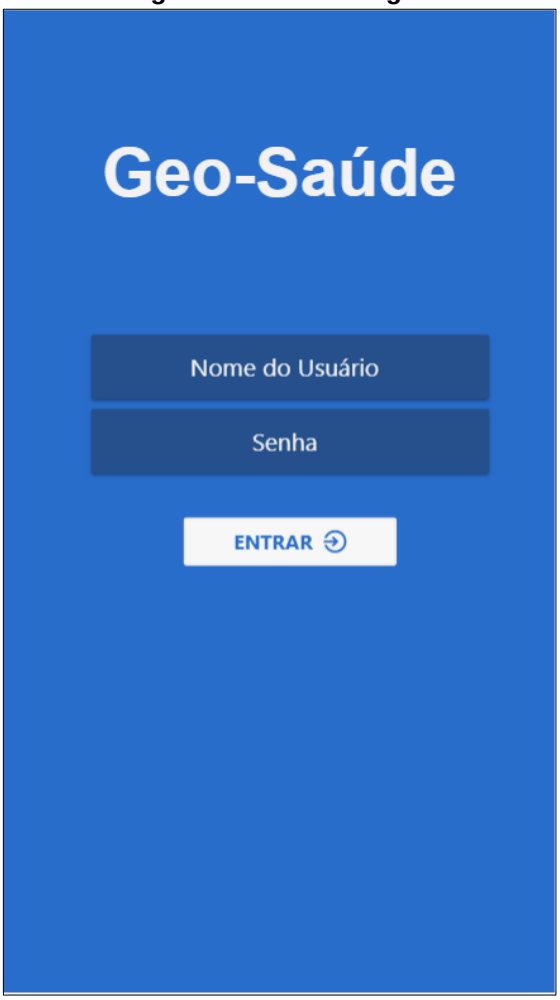
seções principais do aplicativo. Diante disso, analisando o contexto do aplicativo, foi decidido quatro seções como sendo as mais importantes para o aplicativo: perfil, conquistas, atividades e mapa. Para representar essas seções, priorizando o conforto do usuário em alternar entre as seções de forma rápida e fácil, foi decidido utilizar *Tabs Fixas*. Essas *Tabs* ficam na parte inferior do aplicativo, deixando os itens para cada tela sempre a vista.

Vamos agora ver as funcionalidade principais presentes no aplicativo, que estão enumeradas logo abaixo, ilustradas com figuras capturadas da tela do sistema.

- **Autenticação no Sistema**

Com a finalidade de concentrar as atividades para cada ACE, foi decidido que cada agente terá o seu perfil. Desta maneira, para que o agente possa realizar o *login*, é necessário que o seu supervisor tenha previamente cadastrado o agente no sistema do administrador, logo após, o agente deverá ser informado sobre as suas credenciais de acesso e poderá começar a usar o aplicativo Geo-Saúde. Na Tela de Login (Figura 4), o agente preenche os campos com o nome do seu usuário e senha.

Figura 4 - Tela de Login.



Fonte: print screen da aplicação do Geo-Saúde.

- **Tela de Perfil**

Após efetuar o *login*, o usuário será direcionado a Tela de Perfil (Figura 5). Nesta tela estarão as principais informações do agente de saúde: foto do perfil, nome completo, cargo ocupado, localidade de trabalho e as principais conquistas atingidas de acordo com a abordagem gamificada. A tela de perfil, também servirá como um identificador nas visitas feitas pelos agentes. As medalhas conquistadas pelos agentes serão exibidas à medida que o agente for completando suas metas

diárias, semanais e mensais. Nesta mesma tela, será permitido ao agente inserir uma foto sua, de modo a promover a customização do seu perfil.

Figura 5 - Tela de perfil.

Fonte: print screen da aplicação do Geo-Saúde.

- **Realizando as Atividades Principais dos Agentes**

O supervisor será o responsável por distribuir os agentes geograficamente nos distritos sanitários. Cada distrito sanitário conterá vários estratos. Os estratos geralmente são os bairros de uma determinada região. A modelagem de uma atividade para o agente é feita pelo supervisor por meio da atribuição de um distrito sanitário ao agente.

- **Tela de Atividades**

Na Tela de Atividades, Figura 6, do aplicativo Geo-Saúde, serão listadas as atividades disponíveis aos agentes de saúde. Há dois botões que direcionam para determinadas atividades: Iniciar Visita e Enviar Dados.

Figura 6 - Tela de atividades.

Fonte: print screen da aplicação do Geo-Saúde.

O botão Iniciar Visita direciona o usuário ao início do fluxo da coleta das informações de uma visita. Já o botão Enviar Dados ficará disponível quando houver dados armazenados no banco de dados local e que precisam ser enviados ao

servidor. O botão Enviar Dados envia todos os dados presentes no banco de dados local ao servidor, logo em seguida esses dados são apagados do banco local.

- **Tela de Início da Visita**

Após a escolha da opção Iniciar Visita, será aberta a Tela de Início da Visita (Figura 7). Nessa tela serão listados todos os estratos pertencentes ao distrito sanitário ao qual o agente está responsável. O agente deverá escolher um dos estratos de acordo com a localidade que ele está iniciando a visita.

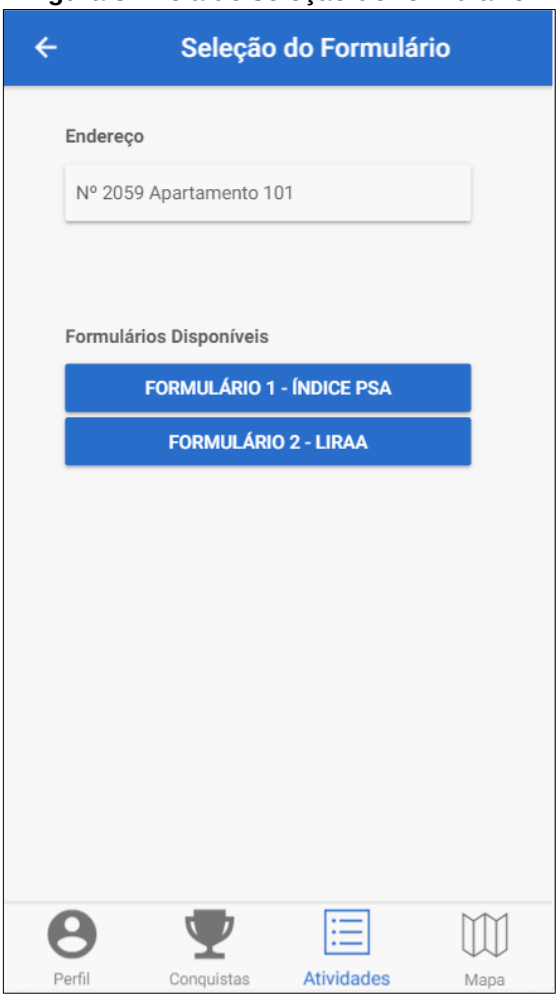
Figura 7 - Tela de início da visita.

Fonte: print screen da aplicação do Geo-Saúde.

- **Tela de Seleção do Formulário**

Quando o agente escolher um dos estratos, a Tela de Seleção do Formulário será aberta (Figura 8). Nessa tela há um campo para preenchimento do endereço do imóvel a ser visitado, bem como para escolha dos formulários de visitas disponíveis: PSA e LIRAa.

Figura 8 - Tela de seleção do formulário.



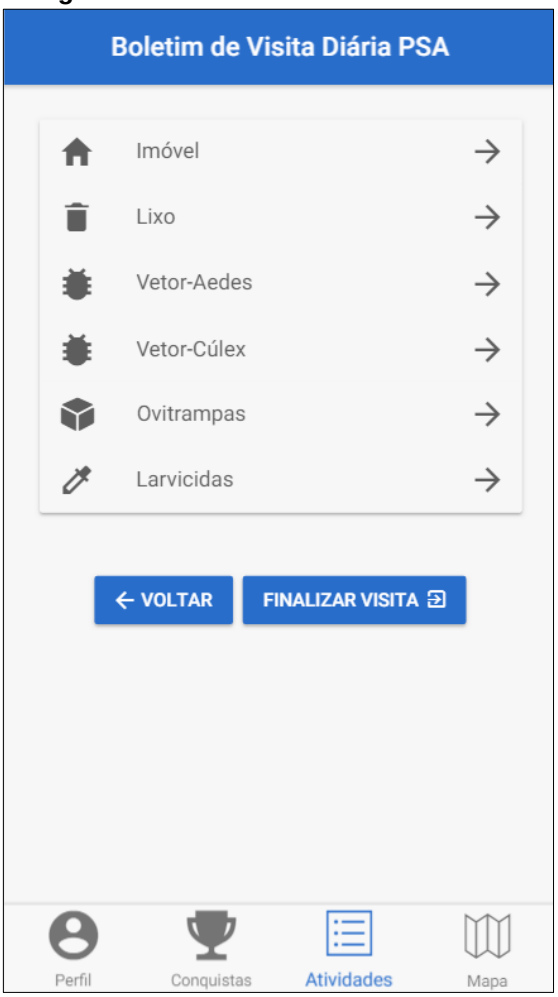
Fonte: print screen da aplicação do Geo-Saúde.

- **Tela de Boletim de Visita Diária PSA e do Índice LIRAa**

Após a escolha de um dos formulários, será aberta a tela para o formulário escolhido. Para a escolha do Formulário 1 – Índice PSA, será aberta a Tela de Boletim de Visita Diária PSA (Figura 9), já para escolha do Formulário 2 – LIRAA, será aberta a Tela de Boletim do Índice LIRAa (Figura 10).

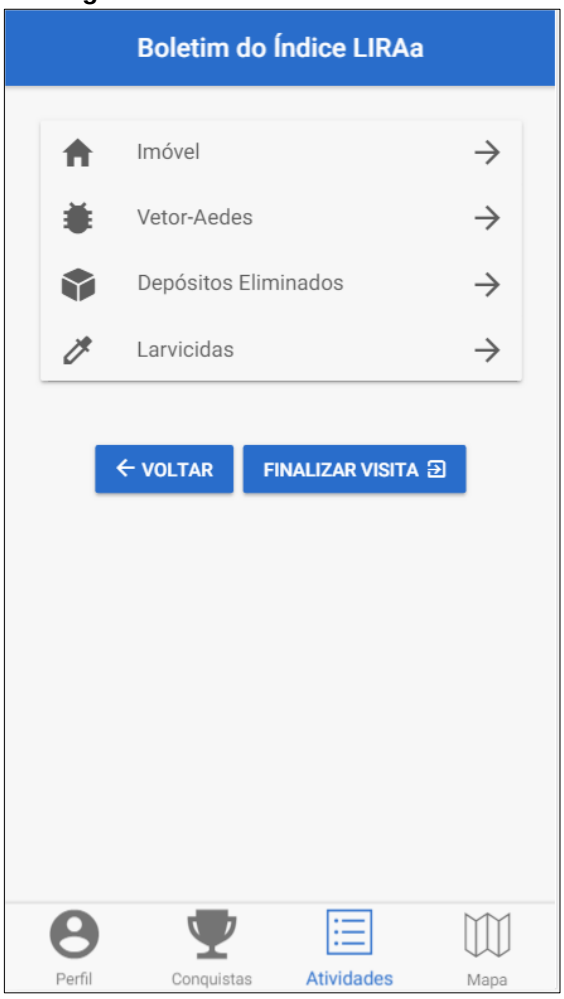
A Tela de Boletim de Visita Diária PSA e a Tela de Boletim do Índice LIRAa, representam os formulários em formato digital. O Formulário de Visita Diária PSA, Figura 9, disponibiliza os campos para preenchimento das principais informações sobre o imóvel (tipo do imóvel, situação e data da visita), lixo encontrado (tipo e acondicionamento do lixo), vetor Cúlex (tipos de criadouros e tratamentos), vetor Aedes (tipo de criadouros e tratamentos), número de ovitrampas (armadilhas) e a quantidade de larvicidas utilizados. O Formulário do Índice LIRAa, Figura 10, disponibiliza os campos para preenchimento das informações sobre o imóvel (tipo do imóvel e data da visita), vetor Aedes (tipos de criadouros), número de depósitos eliminados e a quantidade de larvicidas utilizados.

Figura 9 - Boletim de visita diária PSA.



Fonte: print screen da aplicação do Geo-Saúde.

Figura 10 - Boletim do índice LIRAa

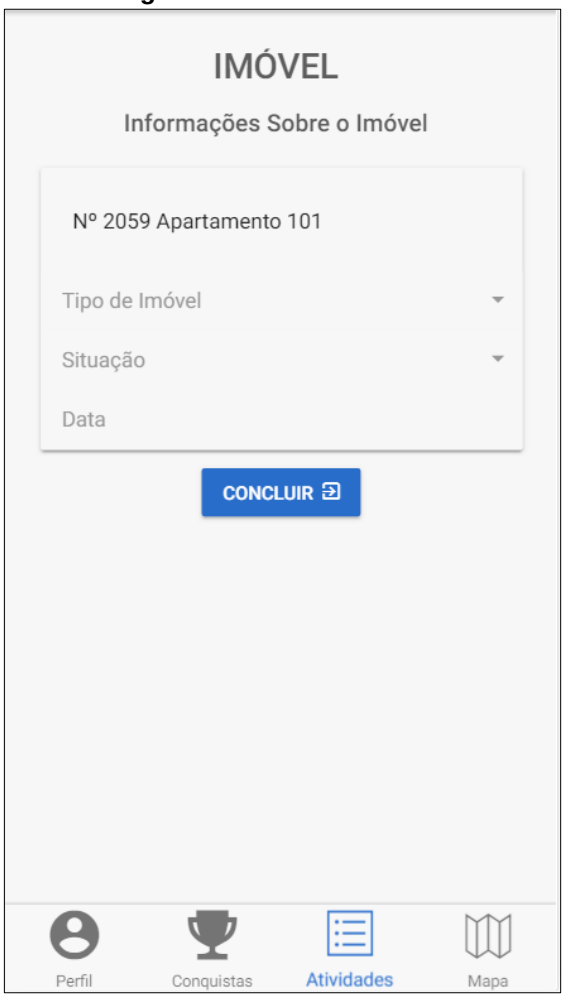


Fonte: print screen da aplicação do Geo-Saúde.

- **Tela de Imóvel**

Na tela de imóvel, Figura 11, o agente vai informar qual o endereço exato do imóvel, o tipo do imóvel (residência, comércio, terreno ou outros), a situação do imóvel (inspecionado, não inspecionado ou recuperado) e a data da visita. O campo para informar sobre a situação do imóvel, só está presente no Boletim de Visita Diária PSA. Ao terminar de preencher os campos sobre o imóvel, bastar clicar em Concluir para prosseguir com o preenchimento do formulário.

Figura 11 - Tela de imóvel.



Fonte: print screen da aplicação do Geo-Saúde.

- **Tela de Informações Sobre o Lixo**

A próxima tela refere-se ao lixo encontrado durante a visita. Nessa tela, Figura 12, serão informados o tipo do lixo (domiciliar ou especial), o acondicionamento (adequado, inadequado ou não se aplica) e o seu destino caso seja do tipo domiciliar (coletado diretamente, coletado indiretamente, queimado e enterrado, jogado em barreira, jogado no rio, jogado no canal, coleta seletiva ou outros). Essa tela só está presente no Boletim de Visita Diária PSA.

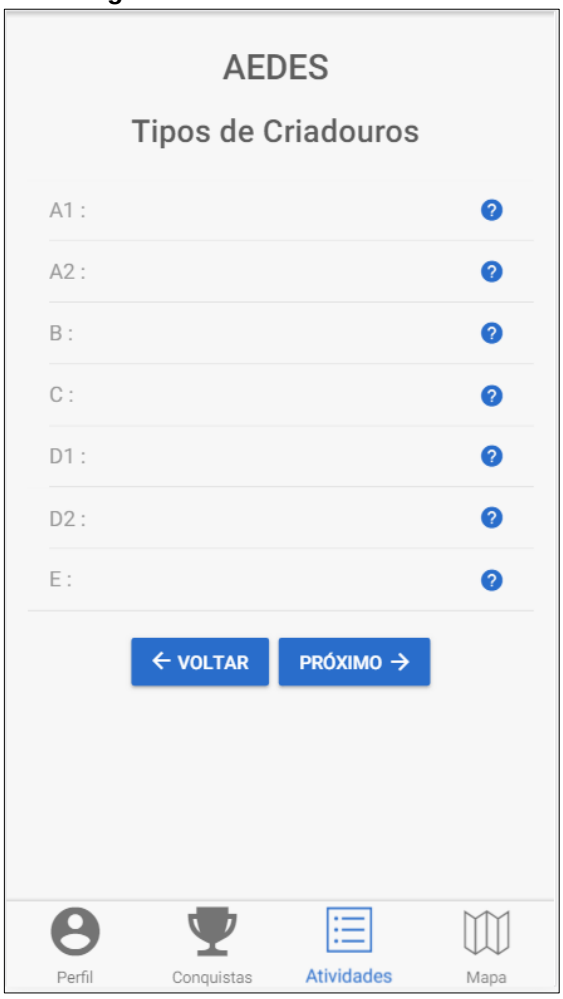
Figura 12 - Tela de informações sobre o lixo.

Fonte: print screen da aplicação do Geo-Saúde.

- **Tela do Vetor Aedes**

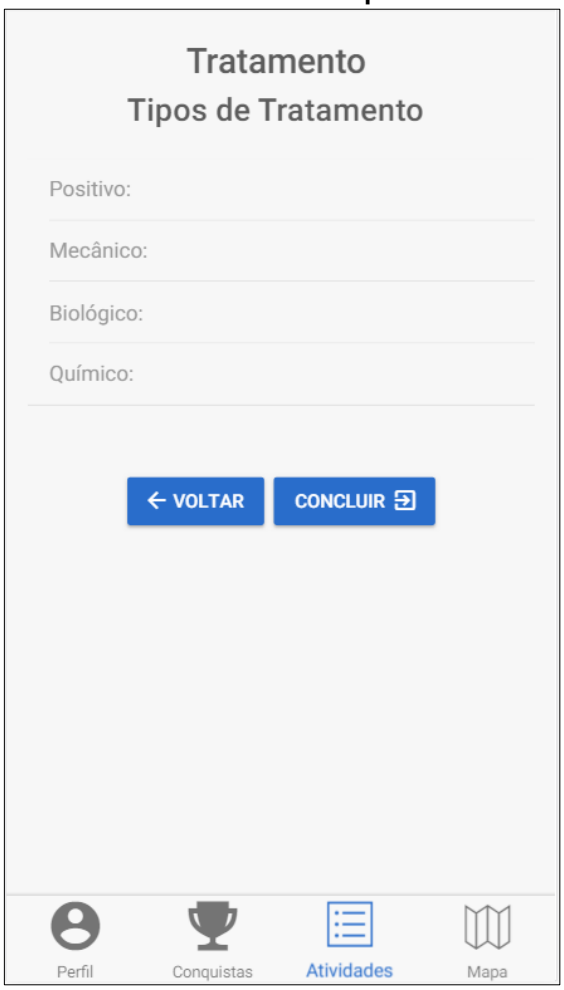
Nessa tela, Figura 13, serão informados os tipos de criadouros (caixa d'água, vasos, pneus, axilas de folhas, lixo e outros) onde foram encontrados focos do mosquito Aedes aegypti. Para o Boletim de Visita Diária PSA, é necessário ainda informar os tipos de tratamentos utilizados (positivo, mecânico, biológico e químico) para cada foco encontrado, Figura 14.

Figura 13 - Tela do vetor aedes.



Fonte: print screen da aplicação do Geo-Saúde.

Figura 14 - Tela de tratamento os campos do aedes e do cúlex.

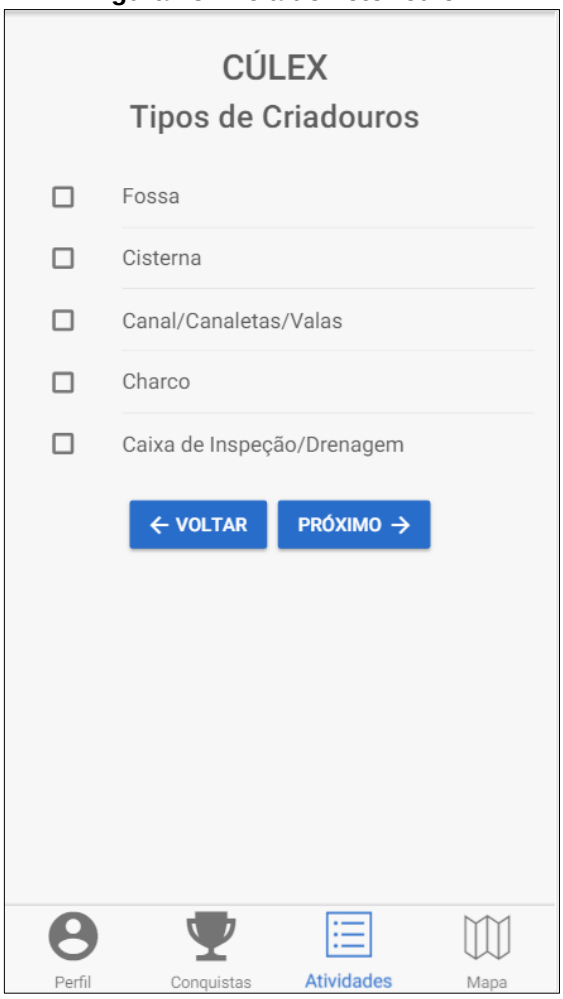


Fonte: print screen da aplicação do Geo-Saúde.

- **Tela do Vetor Cúlex**

Essa tela está presente apenas no Boletim de Visita Diária PSA. Por meio dela serão informados os tipos de criadouros encontrados (Figura 15). Em seguida, já na tela de tratamento (Figura 14), deverão ser informados os tipos de tratamentos utilizados no combate aos focos encontrados.

Figura 15 - Tela do vetor cúlex.

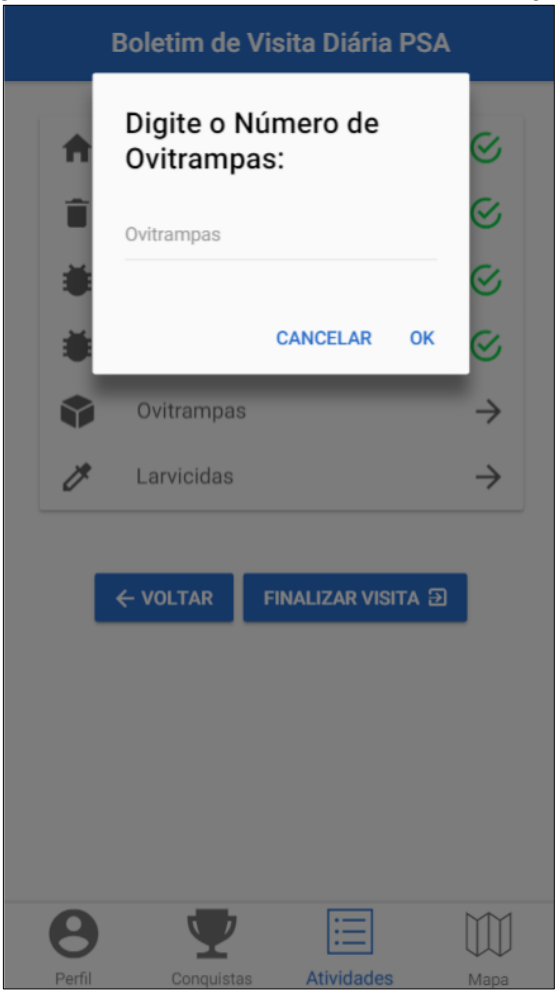


Fonte: print screen da aplicação do Geo-Saúde.

- **Informando o Número de Ovitrampas**

As ovitrampas são armadilhas utilizadas para monitorar a quantidade de mosquitos em uma determinada região. Promovendo ações mais rápidas no combate a proliferação do mosquito. No Boletim do PSA é necessário informar o número de ovitrampas utilizadas na visita (Figura 16).

Figura 16 - Informando o número de ovitrampas.

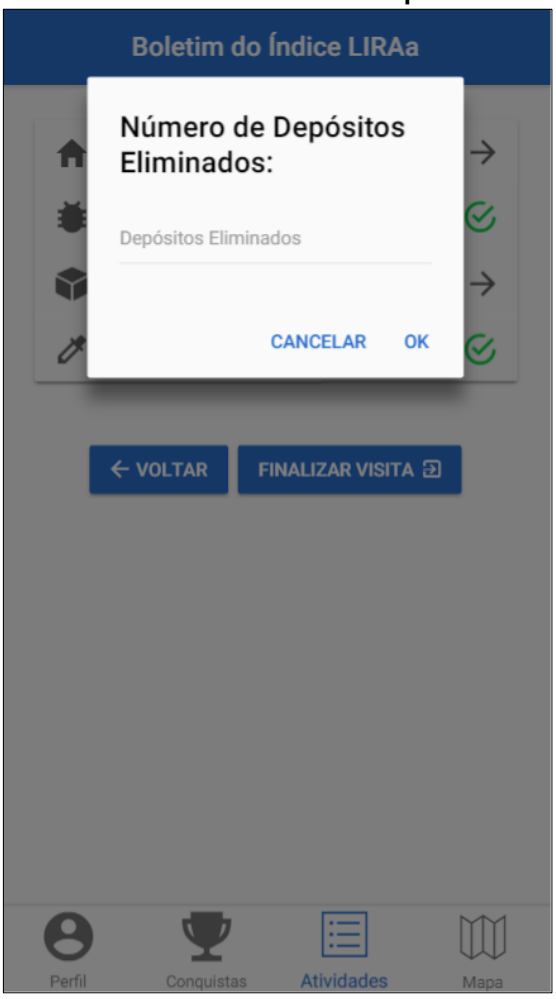


Fonte: print screen da aplicação do Geo-Saúde.

- **Informando o Número de Depósitos Eliminados**

No Boletim do LIRAa é necessário informar o número de depósitos de mosquitos eliminados durante a visita (Figura 17).

Figura 17 - Informando o número de depósitos eliminados



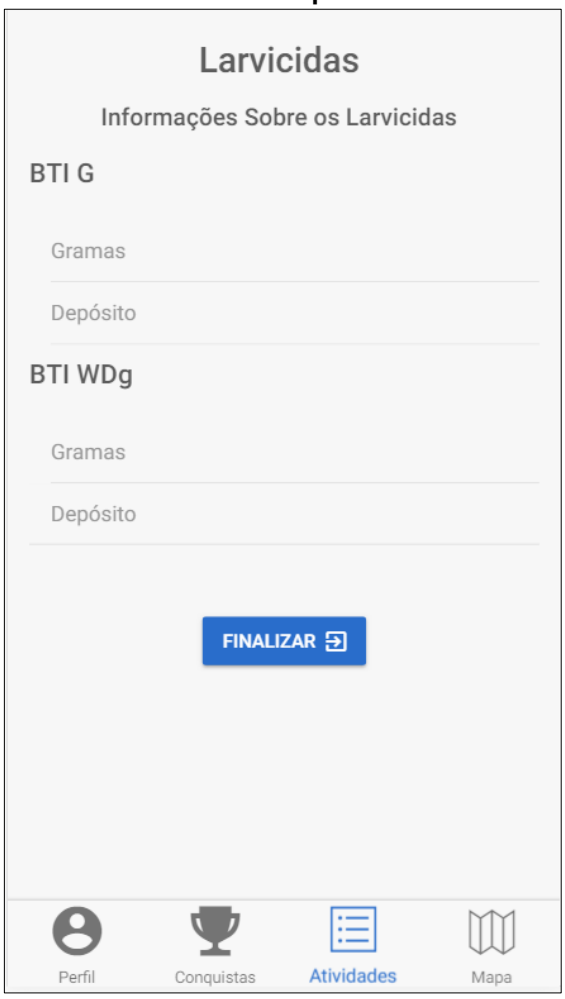
Fonte: print screen da aplicação do Geo-Saúde.

- **Tela de Larvicidas**

Em ambos os boletins é necessário informar sobre os larvicidas utilizados, indicando a quantidade utilizada para cada tipo de larvicida. Os larvicidas disponíveis para os agentes são:

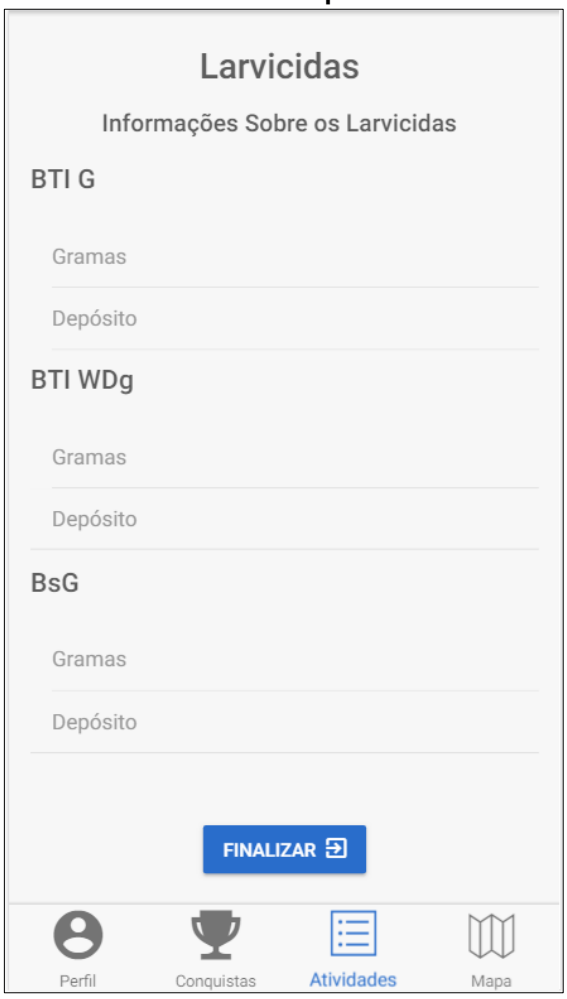
- BTI G e BTI WDg, para o Boletim do LIRAa (Figura 18);
- BTI G, BTI WDg e BsG, para o Boletim do PSA (Figura 19);

Figura 18 - Tela de larvicidas para o boletim do LIRAa.



Fonte: print screen da aplicação do Geo-Saúde.

Figura 19 - Tela de larvicidas para o boletim do PSA.



Fonte: print screen da aplicação do Geo-Saúde.

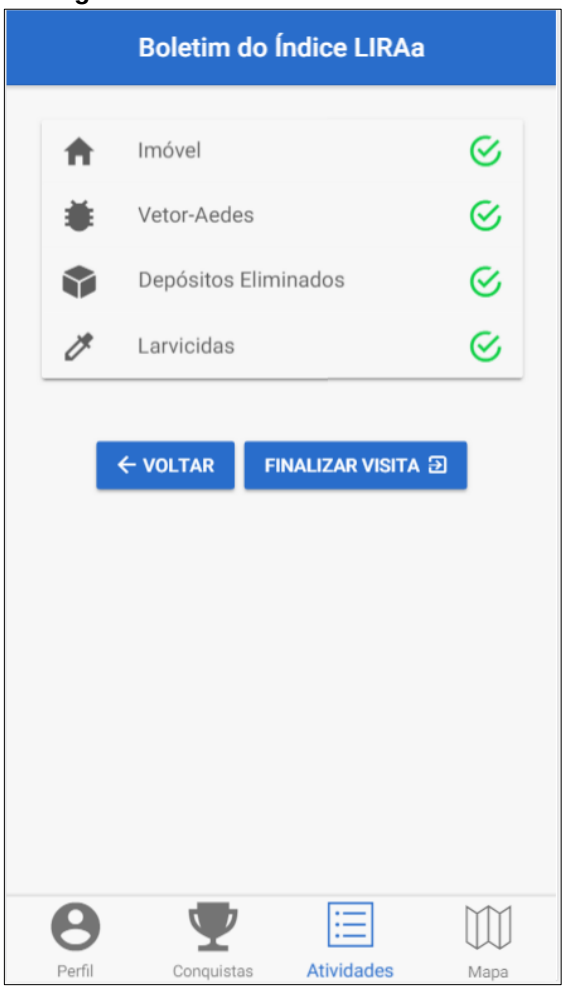
- **Finalizando a Visita**

Após o preenchimento de cada campo dos formulários, conforme mostrado nas figuras Figura 21 e Figura 20, o usuário poderá clicar sobre o botão Finalizar para finalizar a visita e guardar as informações coletadas (Figura 22).

Figura 20 - Boletim PSA concluído.

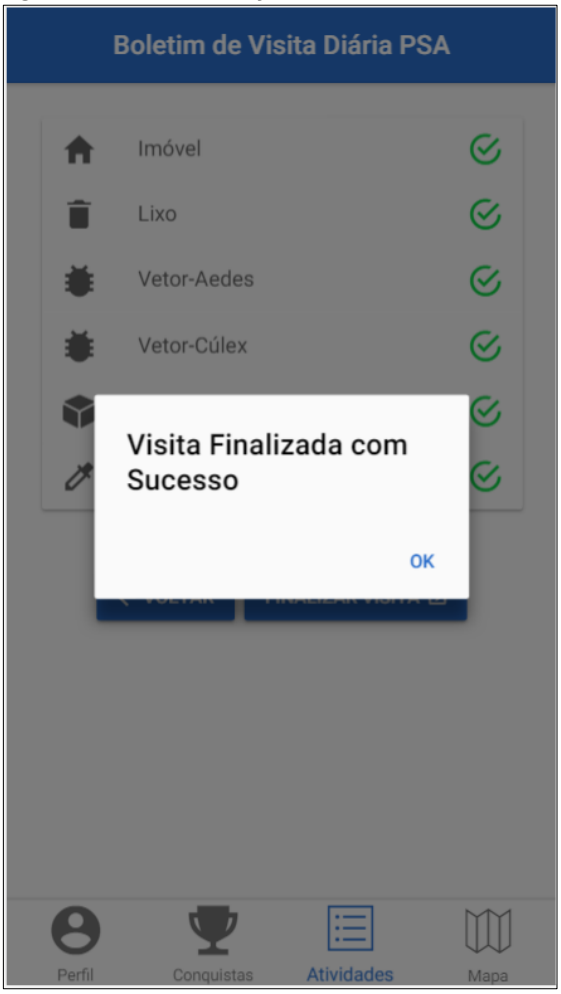
Fonte: print screen da aplicação do Geo-Saúde.

Figura 21 - Boletim LIRAa concluído.



Fonte: print screen da aplicação do Geo-Saúde.

Figura 22 - Confirmação do término da visita.



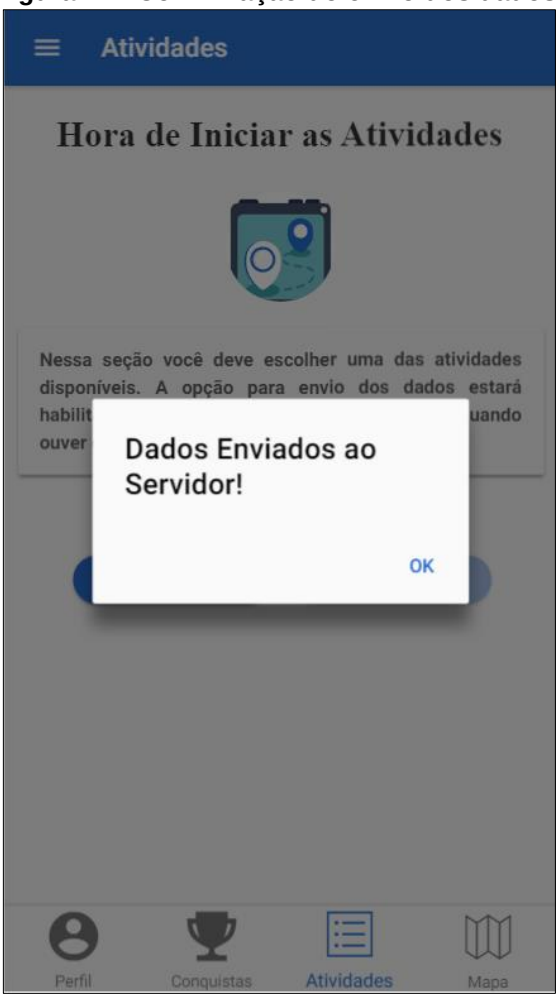
Fonte: print screen da aplicação do Geo-Saúde.

Ao finalizar uma visita, o aplicativo verifica se há conexão com a internet, caso haja, ele envia os dados coletados para o servidor de forma automática. Caso a internet esteja desconectada, os dados poderão ser enviados mais tarde por meio do botão Enviar Dados que fica na tela de atividades. Esse botão será sempre ativado quando houver dados que ainda não foram enviados ao servidor (Figura 23). Quando os dados forem enviados ao servidor após o clique no botão Enviar Dados, será exibida uma mensagem de confirmação (Figura 24).

Figura 23 - Botão enviar dados está ativo.

Fonte: print screen da aplicação do Geo-Saúde.

Figura 24 - Confirmação do envio dos dados.

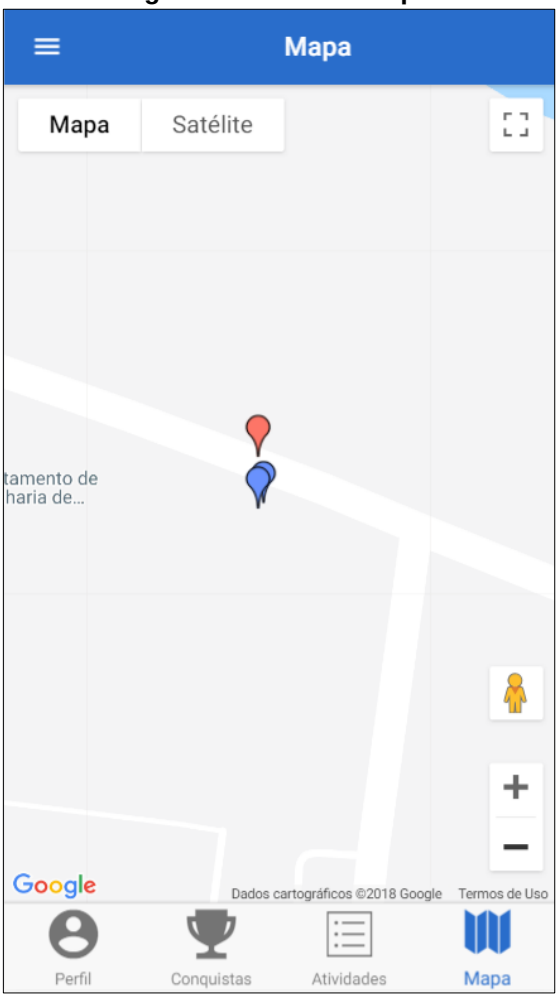


Fonte: print screen da aplicação do Geo-Saúde.

- **Tela de Mapa**

De modo a oferecer uma orientação ao agente, foi decidido utilizar a API do Google Maps para mostrar a localização geográfica para cada localidade, ou seja, de cada distrito e seus respectivos endereços. A Tela de Mapa (Figura 25) sinaliza a localização dos imóveis já visitados (representados pelos pinos da cor azul) e a localização dos imóveis a visitar (representados pelos pinos da cor vermelha), podendo o agente iniciar uma visita clicando nos pinos vermelhos.

Figura 25 - Tela de mapa.

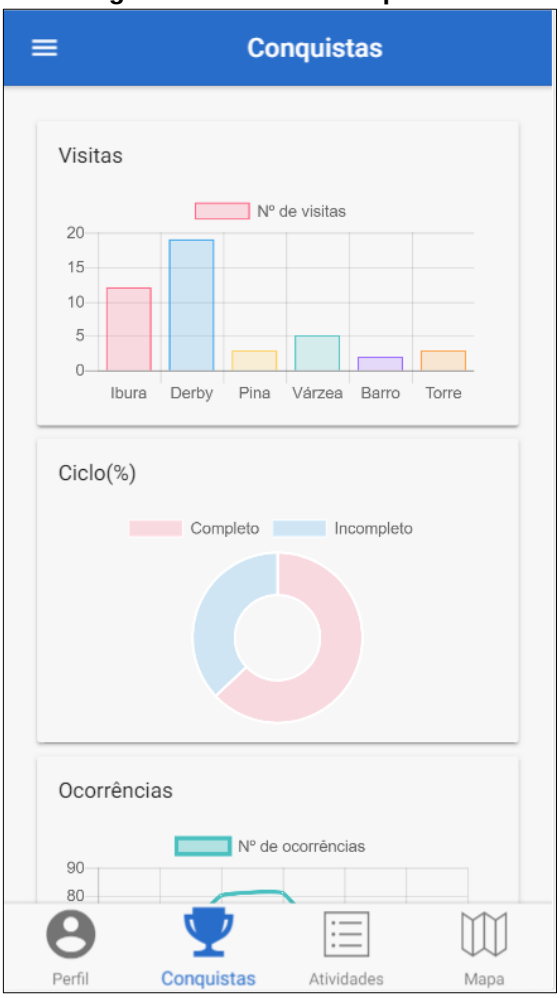


Fonte: print screen da aplicação do Geo-Saúde.

- **Tela de Conquistas**

Na Tela de Conquistas (Figura 26) é possível ver um resumo das atividades do agente. Foi decidido o uso de gráficos em barra, coluna, linha e pizza para fornecer de maneira agradável ao agente uma visão abrangente sobre o seu trabalho. É possível ver as estatísticas sobre as atividades já realizadas e também sobre as atividades que ainda faltam realizar para cumprirem as metas previamente definidas.

Figura 26 - Tela de conquistas.



Fonte: print screen da aplicação do Geo-Saúde.

- **Armazenamento dos Dados do Geo-Saúde**

A partir das oficinas de validação no CVA (Centro de Vigilância Ambiental) de Recife; e CEVAO (Centro de Vigilância Ambiental de Olinda) de Olinda, os ACEs puderam dar algumas sugestões de correções e, assim, participar do desenvolvimento. O ponto de maior destaque foi no tocante a necessidade de estar conectado à Internet para usar o aplicativo e sincronizar os dados coletados com a base de dados remota, sendo esse um cenário improvável em várias regiões populares, seja por conta de áreas de sombra, não cobertas por sinais de

comunicações, seja por questões ligadas ao custo extra que o governo teria que ter para contratar planos de internet junto às operadoras de telefonia móvel. Portanto, foi decidido utilizar um banco de dados local, off-line, para registro das informações dos formulários, atualizando-se o banco de dados na web somente quando na presença de internet.

- **Armazenamento de Dados Locais no Ionic**

O Ionic possui um recurso para armazenamento de dados localmente, o *Storage*. O Ionic *Storage* é uma maneira simples para armazenar pares chave/valor e objetos JSON de forma responsiva. Essa responsividade deve-se ao fato do *Storage* utilizar uma variedade de *engines* de armazenamento, escolhendo a melhor opção dependendo da plataforma ao qual o aplicativo está sendo executado. Essas *engines* são para PWA (*Progressive Web Apps*) ou um aplicativo nativo. Quando executado em um contexto de um aplicativo nativo, o *Storage* priorizará o uso do SQLite, pois é um dos bancos de dados mais estáveis e amplamente usados em aplicativos móveis, evitando algumas armadilhas presentes em outros modos de persistência de dados, como o *LocalStorage* e *IndexedDB*, onde o SO pode decidir limpar os dados armazenados nestes últimos modos de persistência, devido ao pouco espaço na memória de armazenamento do *smartphone* [7].

Destarte, diante das vantagens apresentadas pelo *Storage*, foi decidido utilizá-lo como a forma para armazenamento dos dados de forma local no aplicativo Geo-Saúde.

Para abranger a persistência dos dados tanto do Boletim de Visita Diária PSA quanto do Boletim do Índice LIRAa, foram criados dois *providers*, um para cada tipo de formulário. *Provider* é um serviço do Ionic, com a finalidade de ser utilizado para o desenvolvimento do código de armazenamento de dados. Desse modo, esses *providers* utilizam os modelos estruturais do banco de dados para cada formulário, bem como a manipulação desses modelos por meio de um CRUD. O CRUD é um

acrônimo para as quatro operações básicas em um banco de dados: *Create* (*insert*), *Read* (*select*), *Update* (*update*) e *Delete* (*delete*). Assim sendo, as operações do CRUD vão permitir que realizemos todas as operações junto ao modo de persistência *Storage*.

- **Armazenamento de Dados Remotamente no Ionic**

O desenvolvimento do aplicativo Geo-Saúde, tem como um de seus objetivos acelerar o fluxo dos dados coletados pelos agentes de saúde. Assim, espera-se que os órgãos de saúde pública atuem de forma mais célere no combate às endemias. Diante desse contexto, foi necessário ter uma integração do Ionic com um sistema de *Back-end* ou servidor. Portanto, foi preciso programar o envio de dados do aplicativo Geo-Saúde para o sistema de *Back-end*, para que ele armazenasse os dados e, repassasse-os aos supervisores dos agentes de saúde.

Foi criada uma *API* no aplicativo, com a finalidade de realizar a autenticação do agente e enviar os dados coletados para cada formulário. Nessa *API* é informado o endereço do servidor (*back-end*), assim, quando o agente for entrar no aplicativo, as suas credencias, login e senha, serão verificados nesse servidor. Caso o usuário já esteja cadastrado, será liberado o acesso ao aplicativo. Como o agente não terá internet nos momentos das visitas, o Geo-Saúde persistirá o login do agente no aplicativo, mesmo quando ele estiver sem internet. A outra responsabilidade dessa *API*, é enviar ao servidor as informações dos formulários, LIRAa e PSA, quando finalizados. Toda essa troca de informações entre o aplicativo e o servidor será feita por meio de objetos *JSON*.

No lado do servidor, temos um banco de dados não relacional, o *MongoDB*. Esse tipo de banco é orientado a documentos, ele fornece um alto desempenho, alta disponibilidade e auto escalabilidade. O registro no *MongoDB* é um documento, onde a estrutura de dados é composta de campos com pares de valores. Esses registros em documentos assemelham-se aos objetos *JSON*. Os valores de cada

campo também podem incluir outros documentos. Há três vantagens principais em um banco de dados baseado em documentos: muitas linguagens de programação utilizam documentos como tipos nativos; ao usar documentos diminui-se a complexidade de *joins*; e o suporte ao polimorfismo devido aos esquemas dinâmicos [12].

- **Gerando o APK (Android Package)**

Após o desenvolvimento do aplicativo, foi necessário criar um APK para ele. O APK é um arquivo que nos permite instalar o aplicativo nos smartphones com sistema Android. Para geração do APK, foram realizados os seguintes procedimentos:

i. Primeiramente, entramos no diretório do projeto do aplicativo pela linha de comando (para o sistema operacional Windows utilizamos o Prompt de Comando ou o Windows PowerShell). Em seguida, adicionamos a plataforma na qual o aplicativo será instalado. Para isso, foi executado o comando abaixo:

```
> ionic platform add android
```

ii. Por fim, o APK foi gerado por meio do comando:

```
> ionic build android
```

Logo depois da criação do APK, bastou copiá-lo para o *smartphone android* e instalá-lo. Antes da instalação, devido ao fato do aplicativo não está sendo instalado por meio da loja Google Play Store, foi necessário fazer uma intervenção nas configurações do dispositivo, para habilitar a instalação de aplicativos vindos de fontes desconhecidas. Isso não é recomendado, pois o usuário pode baixar outros

APKs de fontes desconhecidas e que estejam infectados. Portanto, para fins práticos e de testes, habilitamos essa opção, mas logo após a instalação, desabilitamos.

Outra observação que vale ser feita, é com relação à geração do APK com assinatura. Uma assinatura digital para o APK do aplicativo é um dos pré-requisitos para publicar um aplicativo na loja da Google. Para o nosso caso não fizemos isso, pois o objetivo, pelo menos inicialmente, é instalar o aplicativo nos *smartphones* para os profissionais de saúde de regiões específicas.

4. CONCLUSÕES E TRABALHOS FUTUROS

4.1. Desafios

Para o desenvolvimento do aplicativo Geo-Saúde, tivemos algumas dificuldades: atraso na aquisição dos dados e definição do *framework* para desenvolvimento Android. O Centro de Vigilância Ambiental do Recife compartilhou conosco os dados do bairro de Campo Grande, correspondentes a dois meses de coleta; esses dados estavam em formulários impressos em papel, que tinham diversos ruídos inerentes ao quotidiano dos agentes de saúde ambiental e controle de endemias, fazendo com que dispendêssemos bastante tempo na digitalização e correção desses dados para nos auxiliar na definição da estrutura do formulário digital que ficaria no aplicativo. Já no desenvolvimento do aplicativo, inicialmente foi decidido trabalhar com o Android Studio, mas a ferramenta se mostrava muito pesada e burocrática no desenvolvimento. Então decidimos migrar para o *framework* Ionic 3, pela sua facilidade em tempo de desenvolvimento.

Há outros desafios encontrados juntos aos Agentes de Saúde durante as oficinas realizadas: a não familiaridade de alguns agentes com aplicativos para *smartphones*, a falta de segurança para uso de celulares em ambientes públicos e a falta de um plano de dados para o envio mais rápido dos dados ao servidor.

4.2. Contribuições

Neste trabalho foi apresentado um aplicativo voltado aos agentes de controle de endemias, o Geo-Saúde. Esse aplicativo auxilia os agentes em suas atividades diárias, por meio de formulários digitais que facilitam o preenchimento das informações coletadas durante as visitas. Para tanto foi construído um banco de

dados local para o armazenamento de dados enquanto o celular estiver sem internet, e um banco de dados remoto para concentrar todos os dados recebidos após o celular conectar-se à internet. Além do mais, foi utilizada uma abordagem inicial para a gamificação, baseada nas metas de visitas de cada agente, concedendo medalhas aos usuários que cumprirem suas metas.

Por fim, a principal contribuição deste projeto, é fornecer um retorno social com a adição de uma ferramenta que possa contribuir para melhorar a atenção aos cuidados básicos de saúde às populações de baixa renda.

4.3. Trabalhos Futuros

Como trabalho futuro, pretende-se aprimorar a dinâmica de gamificação presente no Geo-Saúde, utilizando métodos de conquistas mais refinados. Também se pretende concluir o desenvolvimento da aplicação *web* no nível do supervisor, para concentração dos dados enviados pelo aplicativo, gerenciamento das contas dos agentes de saúde, e geração de relatórios consolidados em Json, para posterior inserção nas aplicações do Ministério da Saúde.

Além disso, a partir do levantamento dos bancos de dados de informações georreferenciadas de casos de arboviroses e de locais infectados, pretende-se elaborar modelos de predição para possíveis surtos endêmicos em determinadas regiões, e em determinadas épocas do ano. Consequentemente contribuindo na definição de estratégias mais eficientes dos gestores da saúde pública.

Por fim, após a plena adoção do Geo-Saúde pelos órgãos de saúde pública dos municípios de Recife, Olinda, Jaboatão dos Guararapes e Campina Grande, pretende-se continuar expandindo o uso do aplicativo para outras cidades do Brasil.

BIBLIOGRAFIA

[1] BRAGA, I.A.; LIMA, J. B. P.; SOARES, S. d. S.; VALLE, D. Aedes aegypti resistance to temephos during 2001 in several municipalities in the states of Rio de Janeiro, Sergipe, and Alagoas, Brazil. *Memórias do Instituto Oswaldo Cruz*, SciELO Brasil, 99, n. 2, p. 199-203, 2004.

[2] BRAGA, I. A.; VALLE, D. Aedes aegypti: histórico do controle no Brasil. *Epidemiologia e serviços de saúde,* Coordenação-Geral de Desenvolvimento de Epidemiologia em Serviços/Secretaria da Vigilância em Saúde/Ministério da Saúde, 16, n. 2, p. 113-119, 2007.

[3] VASCONCELOS, P. F. d. C. Doença pelo vírus Zika: um novo problema emergente nas Américas? Revista Pan-Amazônica de Saúde, Instituto Evandro Chagas/Secretaria de Vigilância em Saúde/Ministério da Saúde, 6, n. 2, p. 9–10, 2015.

[4] BELTRÁN, J. D.; BOSCOR, A.; SANTOS, W. P. dos; MASSONI, T.; KOSTKOVA, P. ZIKL: A new system to empower health workers and local communities to improve surveillance protocols by e-learning and to forecast Zika virus in real time in Brazil. In: ACM Proceedings of the 8th International Digital Health Conference. [S.I.: s.n.], 2018.

[5] OLIVEIRA, T. R. de; COSTA, F. M. R. da. Desenvolvimento de aplicativo móvel de referência sobre vacinação no Brasil. Journal of Health Informatics, 2012; 4(1): 23-7.

[6] TIBES, C. M. d. S.; DIAS, J. D.; ZEM-MASCARENHAS, S. H. Aplicativos móveis desenvolvidos para a área da saúde no Brasil: revisão integrativa da literatura. Rev. Min. Enferm 2014; 18(2):471-478.

[7] Ionic framework, versão 3. Disponível em: https://ionicframework.com. Acessado em: 02/09/2018.

[8] Apache Cordova. Disponível em: https://cordova.apache.org. Acessado em: 02/09/2018.

[9] JOSÉ, F. R. da S. Ionic Framework Essencial. Versão 1.0.1. 06/2016.

[10] Angular. Disponível em: https://angular.io. Acessado em: 03/09/2018.

[11] Node.js. Disponível em: https://nodejs.org. Acessado em: 03/09/2018.

[12] MongoDB. Disponível em: https://docs.mongodb.com. Acessado em 09/09/2018.

[13] LEANDRO, Debugando o seu código com Google Chrome. DevMedia, 2013. Disponível em: https://www.devmedia.com.br/debugando-seu-codigo-com-google-chrome/29137. Acesso em: 09/09/2018.

APÊNDICE A – CÓDIGO DESENVOLVIDO NO PROJETO

Neste Apêndice encontram-se as principais partes do código fonte do aplicativo Geo-Saúde, bem como a explicação sobre cada funcionalidade presente nessas partes.

O código do aplicativo encontra-se no seguinte repositório do GitLab: https://gitlab.com/anderson.ffelixsilva/geo-saude.git.

- **Arquivo onde são declarados os módulos do projeto**

Todos os componentes do aplicativo são referenciados no arquivo */app/app.module.ts*, conforme podemos ver na Figura 27.

Figura 27 - Código de app.module.ts.

```
// ------------- Importando Pages ---------------------------------------
import { LoginPage } from '../pages/login/login';
import { PerfilPage } from '../pages/perfil/perfil';
import { AchievementsPage } from '../pages/achievements/achievements';
...

// ------------- Importando Providers -----------------------------------
import { PsaFormProvider } from '../providers/psa-form/psa-form';
import { LiraaFormProvider } from '../providers/liraa-form/liraa-form';
import { NetworkProvider } from '../providers/network/network';
import  {  ApiSendFormsProvider  }  from  '../providers/api-send-forms/api-
send-forms';

...

// ------------- Importando Plugins -------------------------------------
import { Camera } from '@ionic-native/camera';
import { Network } from '@ionic-native/network';
import { IonicStorageModule } from '@ionic/storage';
import { HttpModule } from '@angular/http';

@NgModule({
...
imports: [
    IonicModule.forRoot(MyApp),
    HttpModule,
    IonicStorageModule.forRoot()
  ],
...
providers: [
      LiraaFormProvider,
      NetworkProvider,
      PsaFormProvider,
...
  ]
})
```

Fonte: Código fonte do aplicativo Geo-Saúde desenvolvido pelo autor.

- **Código dos Providers**

No código do *app.module.ts*, temos os seguintes *providers*: PsaFormProvider, LiraaFormProvider, NetworkProvider e ApiSendFormProvider. O objetivo principal de cada *provider* é realizar operações com os dados transitados pelo aplicativo, fazendo uma separação bem definida entre a regra de negócio da aplicação e suas páginas ou componentes. Assim sendo, eles possuem as seguintes responsabilidades:

PsaFormProvider

Esse *provider* cria um banco de dados local para armazenar dados provenientes do boletim de visita do PSA. O banco local é criado utilizando o plugin *storage* do Ionic 3. Além disto, esse *provider* contém o código para realizar as operações de CRUD. Deste modo, sendo o responsável por realizar as operações de escrita, leitura, exclusão e atualização dos dados do boletim de visita do PSA no banco local. O PsaFormProvider é detalhado na Figura 28 e Figura 29.

Figura 28 – Fragmento de código do provider psa-form.ts.

```
@Injectable()
export class PsaFormProvider {

...
// ----------------- Construtor -----------------------------------------

  constructor(
    private datepipe: DatePipe,
    private eventsCtrl: Events) {

    // Criando novo storage.
    this.psaFormDb = new Storage({
      name: '__psadb',
      storeName: '__psaForm',
      driverOrder: ['indexeddb', 'sqlite', 'websql ']
    });
  }

// ----------------- Métodos -----------------------------------------

  // Insere dados no banco local.
  public insert(visitInformationPSA: VisitInformationPSA) {
    // Gerando a chave de armazenamento.
    let key = this.datepipe.transform(new Date(), "ddMMyyyyHHmmss");
    return this.save(key, visitInformationPSA);
  }

// Atualiza dados do banco local.
  public update(key: string, visitInformationPsa: VisitInformationPSA) {
    return this.save(key, visitInformationPsa);
  }


  // Salva dados no banco local.
  private save(key: string, visitInformationPSA: VisitInformationPSA) {
    return this.psaFormDb.set(key, visitInformationPSA);
  }


  // Remove dados do banco local.
  public remove(key: string) {
    return this.psaFormDb.remove(key);
  }
```

Fonte: Código fonte do aplicativo Geo-Saúde desenvolvido pelo autor.

Figura 29 - Fragmento de código do provider psa-form.ts

```
  // Busca todos os dados do banco local.
   public getAll() {
    // Declarando a lista que guardará as informações que serão retornadas
    let visitsInformations: VisitInformationPSAList[] = [];

    // Lendo cada linha da estrutura do storage.
    return    this.psaFormDb.forEach((value:    VisitInformationPSA,    key:    string,
iterationNumber: Number) => {

      // Publica um evento para avisar que há dados no storage que precisam ser
enviados ao servidor.
      this.eventsCtrl.publish('haveNewFormsToSend', true);

     // Declarando objeto que receberá o valor dos parâmetros de cada linha.
      let visit = new VisitInformationPSAList(null, null);
      visit.setKey(key);                                  // Recebe a chave.
      visit.setVisitInformation(value);                   // Recebe o valor, neste caso é
o objeto VisitInformationPSA.

      visitsInformations.push(visit);                          // Coloca objeto com as
informações das visitas na lista.
    })
      .then(() => {
       // Retorna lista com todos os formulários do banco local.
        return Promise.resolve(visitsInformations);
      })
      .catch((error) => {
        // Retorna o erro, caso aconteça.
        return Promise.reject(error);
      });
  }
}
```

Fonte: Código fonte do aplicativo Geo-Saúde desenvolvido pelo autor.

LiraaFormProvider

O *provider* para o formulário LIRAa, possui responsabilidades semelhantes ao do formulário PSA. Ele é responsável por criar o banco de dados local para armazenar dados provenientes do boletim de visita do LIRAa, bem como por realizar as operações de CRUD nesse banco local utilizando o plugin *storage* do Ionic 3. O LiraaFormProvider é detalhado na Figura 30 e Figura 31.

Figura 30 - Fragmento do código do provider liraa-form.ts.

```
@Injectable()
export class LiraaFormProvider {
  // -------------- Construtor -----------------------------
  constructor(
    private datepipe: DatePipe,
    private eventsCtrl: Events) {

    // Criando novo banco local.
    this.liraaFormDb = new Storage({
      name: '__liraadb',
      storeName: '__liraaForm',
      driverOrder: ['indexeddb', 'sqlite', 'websql ']
    });
  }

...
  // -------------- Métodos -------------------------

  // Insere novo dados no banco local.
  public insert(visitInformationLIRAa: VisitInformationLIRAa) {
    // Gerando a chave de armazenamento.
    let key = this.datepipe.transform(new Date(), "ddMMyyyyHHmmss");
    return this.save(key, visitInformationLIRAa);
  }

  // Atualiza dados do banco local.
  public update(key: string, visitInformationLIRAa: VisitInformationLIRAa){
    return this.save(key, visitInformationLIRAa);
  }

  // Salva dados no banco local.
  private save(key: string, visitInformationLIRAa: VisitInformationLIRAa){
    return this.liraaFormDb.set(key, visitInformationLIRAa);
  }

  // Deleta dados do banco local.
  public remove(key: string) {
    return this.liraaFormDb.remove(key);
  }
```

Fonte: Código fonte do aplicativo Geo-Saúde desenvolvido pelo autor.

Figura 31 - Fragmento do código do provider liraa-form.ts.

```
  // Busca todos os dados que estão no banco local.
  public getAll() {
    // Declarando a lista que guardará as informações que serão retornadas
    let visitsInformations: VisitInformationLIRAaList[] = [];

    // Lendo cada linha da estrutura do storage.
    return   this.liraaFormDb.forEach((value:   VisitInformationLIRAa,   key:   string,
iterationNumber: Number) => {
      // Publica um evento caso avisando que há dados para ser enviados ao servidor.
      this.eventsCtrl.publish('haveNewFormsToSend', true);

      // Declarando objeto que receberá o valor dos parâmetros de cada linha.
      let visit = new VisitInformationLIRAaList(null, null);
      visit.setKey(key);                    // Recebe a chave.
      visit.setVisitInformation(value);       // Recebe o valor, neste caso é o objeto
VisitInformationLIRAa.

      visitsInformations.push(visit);         // Coloca objeto com as informações das
visitas na lista.
    })
      .then((msg) => {
        // Retorna lista com todos os formulários liraa do banco local.
        return Promise.resolve(visitsInformations);
      })
      .catch((error) => {
        // Retorna o erro, caso ocorra.
        return Promise.reject(error);
      });
  }
}
```

Fonte: Código fonte do aplicativo Geo-Saúde desenvolvido pelo autor.

NetworkProvider

Devido à falta de um plano de dados móveis aos agentes de saúde, é necessário, conforme já foi exposto anteriormente nesse trabalho, o armazenamento de dados de forma local, com o envio desses dados ao servidor quando houver conexão com a internet. Portanto, o NetwordProvider, Figura 32, é o responsável por monitorar se há conexão com a internet.

Figura 32 - Código do provider network.ts.

```
@Injectable()
export class NetworkProvider {

  // -------------- Atributos -------------------------
  // Guarda o estado de conexão da rede (offline ou online).
  public status: string;


  // -------------- Construtor -------------------------

  constructor(
    public network: Network,
    public eventCtrl: Events,
    public toast: ToastController) {

    this.status = "online";
  }


  // -------------- Métodos -------------------------
  // Inicializa evento de rede.
  public initializeNetworkEvents(): void {

    // Quando a internet for desligada.
    this.network.onDisconnect().subscribe(() => {
      if (this.status == "online") {
        // Publica evento avisando que a rede está offline.
        this.eventCtrl.publish('network:offline');

        // Altera estado da rede para offline.
        this.status = "offline";
      }


    });

    // Quando a internet for ligada.
    this.network.onConnect().subscribe(() => {
      if (this.status === "offline") {

        if (this.network.type === 'wifi' || this.network.type == "3g" ||
this.network.type == "4g") {
          // Publica evento avisando que a rede está online.
          this.eventCtrl.publish('network:online');
          // Altera estado da rede para online.
          this.status = "online";
        }
      }

    });
  }

  // Retorna o estado da rede.
  public getNetworkStatus() {
    return this.status;
  }
}
```

Fonte: Código fonte do aplicativo Geo-Saúde desenvolvido pelo autor.

ApiSendFormProvider

Esse *provider* é responsável por enviar os dados ao servidor, tanto do formulário LIRAa quanto do formulário PSA. Desta forma, o endereço do servidor é informado nesse componente. No escopo desse *provider* (Figura 33 e Figura 34), há dois métodos principais para cada formulário, um para enviar os dados para apenas uma visita, e outro para enviar os dados de várias visitas realizadas. O primeiro método é utilizado assim que o agente finaliza uma visita, sob a condição de ter conexão com a internet, os dados da visita finalizada são automaticamente enviados. Já o segundo método é utilizado após a realização de diversas visitas, sob a condição de não ter conexão com a internet no momento das visitas, sendo assim necessário armazenar os dados localmente, e enviá-los posteriormente utilizando esse segundo método do *provider* ApiSendFormProvider. É por meio desse *provider* que a autenticação do usuário é realizada (Figura 35).

Figura 33 - Fragmento de código do provider api-send-forms.ts.

```
@Injectable()
export class ApiSendFormsProvider {

  // -------------- Atributos --------------------------
  private urlServer = 'http://localhost:8080/';
  private endUrlFormPsa = 'psa-form/';
  private endUrlFormLiraa = 'liraa-form/';

  // -------------- Construtor --------------------------
  constructor(
    public http: Http,
    public psaProvider: PsaFormProvider,
    public liraaProvider: LiraaFormProvider) {
    console.log('Hello ApiSendFormsProvider Provider');
  }

// -------------- Métodos --------------------------

// Envia os dados do boletim PSA para uma visita.

  sendFormPsa(formPsaVisited) {
    return new Promise((resolve, reject) => {
      let hds = new Headers();
      hds.append('Content-Type', 'application/json');

      console.log(formPsaVisited);

      this.http.post(this.urlServer                +                this.endUrlFormPsa,
JSON.stringify(formPsaVisited), { headers: hds })
        .map(res => res.json())
        .subscribe(data => {
          resolve(data);

        }, error => {
          reject(error);
        });
    });
  }

// Envia os dados do boletim LIRAa para uma visita.

    sendFormLiraa(formLiraaVisited) {
    return new Promise((resolve, reject) => {
      let hds = new Headers();
      hds.append('Content-Type', 'application/json');

      this.http.post(this.urlServer               +               this.endUrlFormLiraa,
JSON.stringify(formLiraaVisited), { headers: hds })
        .map(res => res.json())
        .subscribe(data => {
          resolve(data);

        }, error => {
          reject(error);
        });
    });
  }
...
```

Fonte: Código fonte do aplicativo Geo-Saúde desenvolvido pelo autor.

Figura 34 - Fragmento de código para o provider api-send-forms.ts.

```
...

   // Envia os dados do boletim LIRAa para várias visitas.

   sendAllFormsLiraa(formsLiraaVisited) {
    return new Promise((resolve, reject) => {
      let hds = new Headers();
      hds.append('Content-Type', 'application/json');

      console.log(formsLiraaVisited);

      formsLiraaVisited.forEach((value: VisitInformationLIRAa) => {
        var visitLiraaInfo = value.visitInformationLIRAa;

        this.http.post(this.urlServer              +              this.endUrlFormLiraa,
JSON.stringify(visitLiraaInfo), { headers: hds })
          .map(res => res.json())
          .subscribe(data => {
            resolve(data);
            // Deleta todos os registros locais.
            this.liraaProvider.remove(value.key);

          }, error => {
            reject(error);
          });
      });
    });
  }

// Envia os dados do boletim PSA para várias visitas.

  sendAllFormsPsa(formsPsaVisited) {
    return new Promise((resolve, reject) => {
      let hds = new Headers();
      hds.append('Content-Type', 'application/json');

      console.log(formsPsaVisited);

      formsPsaVisited.forEach((value: VisitInformationPSA) => {
        var visitPSAInfo = value.visitInformationPSA;

        this.http.post(this.urlServer               +               this.endUrlFormPsa,
JSON.stringify(visitPSAInfo), { headers: hds })
          .map(res => res.json())
          .subscribe(data => {
            resolve(data);
            // Deleta todos os registros locais.
            this.psaProvider.remove(value.key);

          }, error => {
            reject(error);
       });
      });
    });
  }
...
```

Fonte: Código fonte do aplicativo Geo-Saúde desenvolvido pelo autor.

Figura 35 - Fragmento de código do provider api-send-forms.ts.

```
...

// Realiza a autenticação do usuário comunicando-se com o servidor.

public sendAuthenticationData(username: string, senha: string) {
    //Realizando chamada HTTP direcionada ao Back-End
    return this.http.post(this.urlServer + this.authEndPoint, { 'username': username,
'senha': senha });

  }
}
```

Fonte: Código fonte do aplicativo Geo-Saúde desenvolvido pelo autor.