Universidade Federal Fluminense
Instituto de Ciência e Tecnologia
Bacharelado em Ciência da Computação

Lucas Simas de Andrade Santos

Aplicativo móvel para localização de Unidades de Saúde

Rio das Ostras-RJ
2017

LUCAS SIMAS DE ANDRADE SANTOS

APLICATIVO MÓVEL PARA LOCALIZAÇÃO DE UNIDADES DE SAÚDE

Monografia apresentada ao Curso de Bacharelado em Ciência da Computação do Instituto de Ciência e Tecnologia da Universidade Federal Fluminense, como requisito parcial para obtenção do Grau de Bacharel.

Orientador: Prof. Dr. CARLOS BAZILIO MARTINS

Rio das Ostras-RJ

2017

LUCAS SIMAS DE ANDRADE SANTOS

APLICATIVO MÓVEL PARA LOCALIZAÇÃO DE UNIDADES DE SAÚDE

Monografia apresentada ao Curso de Bacharelado em Ciência da Computação do Instituto de Ciência e Tecnologia da Universidade Federal Fluminense, como requisito parcial para obtenção do Grau de Bacharel.

Aprovada em DEZEMBRO de 2017.

BANCA EXAMINADORA

Prof. Dr. CARLOS BAZILIO MARTINS - Orientador
UFF

Prof. Ms. EDUARDO MARQUES
UFF

Prof. Dr. ADRIANA PEREIRA DE MEDEIROS
UFF

Rio das Ostras-RJ
2017

Ao meu pai Josenir Pereira dos Santos, minha mãe Zeruia Simas de Andrade Santos e minha irmã Marihá Simas de Andrade Santos, por todo apoio e carinho durante esta jornada.

Agradecimentos

Agradeço primeiramente a Deus, por ter me dado saúde e força para sustentar este sonho. Aos amigos que fiz nesta caminhada e que não estaria completa sem eles, que em todos os momentos que estiveram comigo, sendo eles, bons ou ruins. Por todas ou quaisquer palavras de conselho dadas. E apesar de serem muitos irei citar alguns em especial. Aos meus companheiros e amigos de república em primeiro lugar: Caio Chames, Eduardo Mello, Gabriel Rangel, Michael Rocha e Pedro Campos, por toda paciência, brincadeira, estudo e confiança que tivemos.

À minha família que inquestionavelmente me apoiou. Não existem palavras suficientes para agradecer. Aos professores deste campus que se esforçam para passar todo este conhecimento. E se dedicam com todo seu coração à este trabalho.

Lista de Figuras

Lista de Tabelas

Sumário

Resumo

Neste projeto será apresentada uma solução capaz de reunir informações sobre unidades de saúde e entregá-las aos usuários de maneira útil e simplificada. Esta solução não será restrita à unidades de saúde públicas ou privadas, irá cobrir ambas. Após a apresentação do problema, foi planejado o sistema através da criação de diagramas e descrição de seus requerimentos. Ao final, foi construído um protótipo do sistema, baseado nos diagramas, com o objetivo de fazer testes das funcionalidades projetadas e de verificar se a solução proposta era válida para resolver parcialmente ou completamente o problema.

Palavras-chave: Android, Saúde, Aplicativo.

Abstract

This project will present a solution capable of gathering information about health units and delivering them to users in a useful and simplified way. This solution will not be restricted to public or private health units, will cover both. After the presentation of the problem, the system was planned through the creation of diagrams and description of its requirements. In the end, a prototype of the system was built, based on the diagrams, with the objective of testing the designed features and verifying that the proposed solution was valid to partially or completely solve the problem.

Keywords: Android, Health, App.

Capítulo 1

Introdução

É cada vez mais comum para os cidadãos usarem a internet para obter informações como locais para comer, atendimento de algum serviço, compras de produtos específicos e outros. Porém, nem tudo que é buscado na internet pode ser encontrado com a mesma facilidade. Por exemplo, na área de saúde não seria muito complicado encontrar um hospital na cidade que você se encontra. Porém seria mais complicado encontrar um hospital na cidade que atenda a uma especialidade específica. Mais complicado ainda, seria procurar nas cidades vizinhas. Geralmente lugares diferentes fornecem informação de maneira diferente, e você teria que depender da facilidade que a informação está disposta para encontrar o que precisa.

A tecnologia da informação pode ser definida como um conjunto de soluções que informatizam uma determinada área. Em geral, ao permitir o gerenciamento de informações em um computador você está informatizando um sistema. Para exemplificar, o governo começou a informatizar o sistema do SUS com formulários de perguntas sobre atendimentos em unidades UBS (unidade básica de saúde) através do sistema e-SUS.

A disponibilização da informação na internet é, na maioria dos casos, simples de se implementar por ser um serviço barato e fácil de ser contratado. Como exemplo do serviço de armazenamento barato pode-se ver o Google Cloud Storage que possui o preço de 0,01 dolares por GB/mês [1]. O problema ocorre na hora de filtrar esta rede de informações, ou seja, como localizar uma informação em um espaço tão abrangente.

O grau de dificuldade para selecionar o conteúdo desejado é o que faz diferença entre um software e outro. Conseguir uma melhor precisão dos dados buscados ou melhores meios de adquirir uma informação, por mais específica que seja, pode definir o software com melhores chances de ser utilizado. Isto pode fazer com que dois programas com o mesmo objetivo tenham popularidades diferentes. Esta definição não considera características muito importantes como a usabilidade e interface amigável.

1.1 Motivação

A motivação inicial aconteceu ao estudar melhor os programas já existentes na área de saúde como: Busca Saúde(SP) [3], Bradesco Seguros [4], Help Saúde [5], Amil [6], Unimed [7], Busca AMS [8] e Google Maps

[9]. Alguns destes sistemas serão melhor explanados no capítulo 2. Foi notado que grande parte destes possuem as informações necessárias para que o usuário possa se localizar. Porém, nenhum deles oferece conteúdo para várias localidades (municípios, estados, regiões,, etc.) ou tipos de unidades diferentes. Os aplicativos vistos, Busca AMS [8] e Busca Saúde [5], possuem escopo limitado a uma rede (grupo empresarial) e localidade respectivamente. No caso, estes aplicativos englobam unidades de saúde do tipo pública ou privada ou até unidades muito específicas como: UBS, UPA e SUS. Ou seja, o usuário não vai conseguir obter todas as informações que ele deseja com um único aplicativo.

Atualmente também é possível localizar unidades de saúde através do Google Maps [9]. Porém, ele enxerga a unidade como um local apenas, não existem informações como especialidades de atendimento, médicos ou exames possíveis de se realizar no local. Ainda assim, pode haver uma dificuldade maior na busca de locais como clínicas ou outros tipos de unidades menores.

Foi estudado qual seria o sistema para que este programa fosse melhor utilizado. Considerando duas opções principais: sistema móvel ou um sistema web. Pensando no sistema web pode-se definir sem um estudo que seria a melhor opção devido à sua abrangência que chega à ser maior que a dos dispositivos móveis, considerando que este pode servir tanto aos próprios dispositivos quanto à desktops ou notebooks. Porém, sistemas web não são tão fáceis quanto programas para serem utilizados, pois para um usuário é muito mais fácil instalar um aplicativo do que ficar decorando quais sites são utilizados para o que ele precisa, além de que um aplicativo pode funcionar sem internet mesmo que de modo limitado. Também não parecia ser tão atrativo ter um sistema web apenas para ser acessado por computadores, desde que o usuário estaria impossibilitado de checar o local destino a qualquer momento. Mais um ponto levado em consideração foi o aumento do número de usuários de dispositivos móveis e a constante criação de aplicativos voltados para as necessidades do usuário. Esta informação pode ser vista através de duas pesquisas. A primeira é uma estimativa feita pela Zenith Media [11], que verificou o aumento do uso de internet feito por aparelhos móveis de 65% em 2016 para 70% em 2017, e estimou que chegará até 73% em 2018 . A segunda pesquisa é da comScore feita nos Estados Unidos onde o número de usuários da internet apenas para desktop vem diminuindo enquanto usuários apenas para móvel vem aumentando. Além disso no ano de 2015 pela primeira vez os dispositivos móveis passaram a frente conforme a figura 1.1 [12]. De acordo com os argumentos citados um aplicativo móvel é a melhor escolha, pois alcança um maior número de usuários o que implica em uma maior chance de ser utilizado.

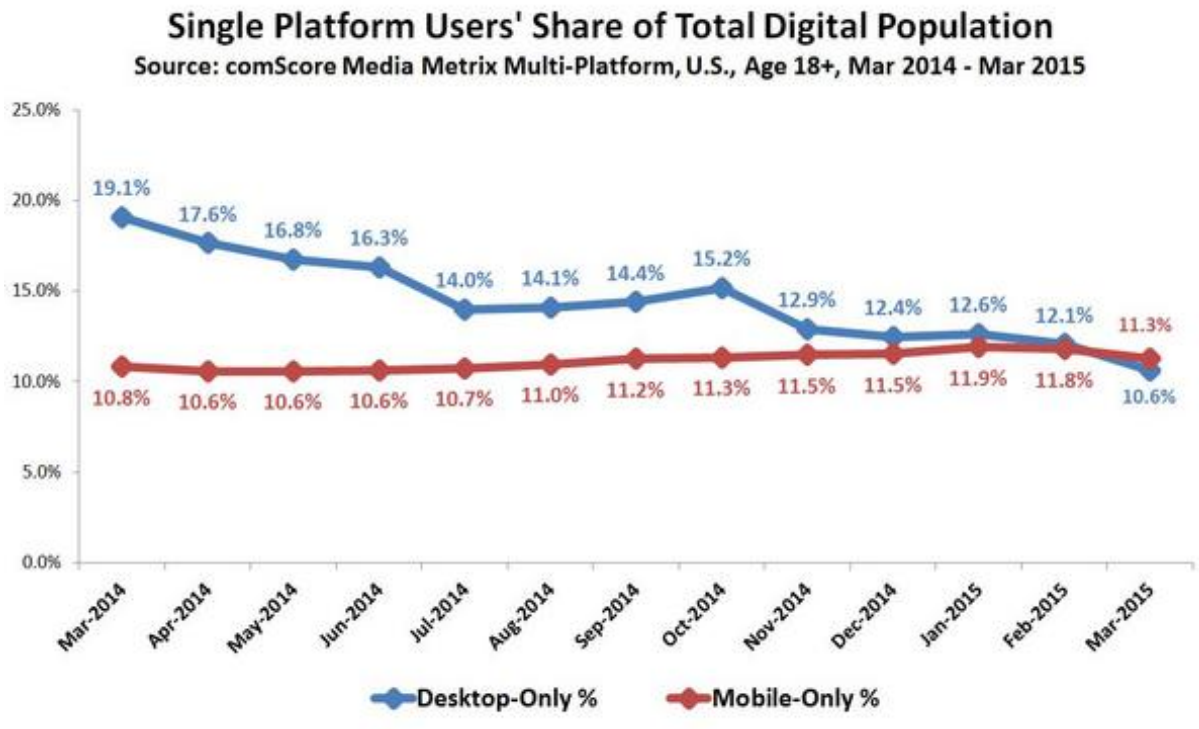


Figura 1.1: Utilização de internet por plataforma única (apenas desktop e apenas móvel)

Atualmente, é possível ver vários sites que passaram a ter um aspecto de aplicativo, pelo menos quando acessados por dispositivos móveis. Por isto e pelo fato da escolha do sistema ser um aplicativo, por ser mais simples de ser utilizado e mais popular, pode-se dizer que o planejamento do sistema foi fortemente influenciado pelo PWA (Progressive Web App)[2]. Este termo é utilizado para sites que utilizam as mais novas tecnologias de desenvolvimento. Estes sites geralmente possuem algumas características em comum, como: responsividade de acordo com a resolução da tela do usuário, interação com o usuário como se fosse um aplicativo e progressivo. O termo progressivo é utilizado para sites que possuem um núcleo simples que funcionaria em qualquer navegador e também uma parte mais desenvolvida que seria oferecida para certos tipos de navegadores que dão suporte. O termo PWA será melhor explicado na seção de tecnologias utilizadas.

Mesmo com esta quantidade enorme de aplicativos que se tem acesso, existem áreas onde é preciso desenvolver novas ideias. Pode-se afirmar que uma destas ideias é dar ao usuário a possibilidade de atender muitas das suas necessidades com um único aplicativo. Por exemplo, suponha que uma pessoa viaja muito do Rio de Janeiro para São Paulo e para cada cidade exista um aplicativo com as informações sobre a condução (meios de transporte). Então esta pessoa deveria manter dois aplicativos em seu dispositivo para poder fazer uso deles, ou seja, mesmo se forem simples eles estão restringindo o usuário à uma área que eles alcançam. Seria isso um problema? Para alguns casos, dois aplicativos que fazem a mesma coisa instalados no mesmo dispositivo vão gerar um certo desconforto e dificuldade de uso para quem usa. Além disso, fica evidente que recursos do dispositivo estão sendo utilizados inutilmente ao manter dois programas que possuem a mesma função com diferença apenas nos dados que eles contém.

Foi notado na área de saúde uma grande deficiência no campo de tecnologia da informação. Não

existe um meio que concentra todas essas informações de unidades de saúdes, tipos de atendimentos, especialidades médicas atendidas. Ainda que existam alguns aplicativos com esse tipo de informação, os mesmos estão situados com um limite (ex: por tipo de unidade de saúde, região, etc).

Também foi pensado em uma maneira na qual os próprios usuários do aplicativo iriam fornecer informações de locais de saúde que não estão mapeados no sistema e assim obter informações de todas as áreas possíveis. Essa seria uma ótima maneira de ter um sistema atualizado, através de um banco de dados colaborativo.

1.2 Objetivo

Desenvolver um aplicativo que reúna informações de unidades de saúde tanto para consultas quanto atendimentos emergenciais para que o usuário possa ser encaminhado para o local correto com base na sua atual necessidade, além de permitir aos próprios usuários do sistema atualizá-lo.

Capítulo 2

Ferramentas Similares

Neste capítulo serão apresentados softwares que possuem um ou mais objetivos iguais aos do desenvolvido neste projeto, mostrando suas funcionalidades.

É de grande importância na hora do desenvolvimento de um programa, obter conhecimento de ferramentas parecidas, com o intuito de melhorar seu próprio aplicativo. Por isso, serão citados alguns aplicativos que têm um objetivo igual ou bem próximo ao do que esta sendo apresentado, juntamente com suas ideias e funcionalidades.

Vários aplicativos e sites foram encontrados como: Busca Saúde(SP) [3], Bradesco Seguros [4], Help Saúde [5], Amil [6], Unimed [7], Busca AMS [8] e Google Maps [9]. Além de listados, serão descritos abaixo os sistemas que mais se assemelham ao projeto proposto.

2.1 Google Maps

O Google Maps oferece vários serviços de busca para vários tipos de estabelecimento (hospitais, restaurantes, lanchonetes, postos de gasolina, etc). No caso, ele próprio permite que o usuário busque unidades de saúde como: clínicas, hospitais, e outros. Entretanto qualquer estabelecimento tratado pelo Google não possui uma diferenciação, ou seja, sendo ele um restaurante ou hospital as informações fornecidas são as mesmas; sendo estas: endereço, contato, dias e horários de atendimento. O que é um problema quando relacionamos ao projeto sugerido, pois não traz informações especializadas do hospital.

Com um exemplo na imagem 2.1 vemos uma busca de hospital feita no aplicativo do Google Maps. Nesta pesquisa, além de permitir buscar com texto assim como na imagem, abaixo do campo de texto pode-se escolher visualizar apenas os hospitais e clínicas que estão abertos no momento, através de um botão abaixo da barra de busca ”Abertos agora”.

Figura 2.1: Tela do resultado de busca de hospital do Google Maps

Contudo por ser um aplicativo mais genérico abrangendo todo o planeta, ele possui algumas falhas. Essas falhas vem do grau de incerteza de suas informações. Como seus dados são gerados de maneira colaborativa, vindos dos seus usuários, não podemos afirmar que qualquer uma de suas informações estão corretas. Como exemplo usamos a Clínica Mater Dei e o Pronto Socorro da cidade de Rio das Ostras - RJ, os quais não estão classificados como hospitais ou clínicas em uma busca feita pelo API do Google Maps no dia 13/08/2017, e isto confirma o grau de incerteza dos dados.

2.2 Help Saúde

O Help Saúde é um site que é especializado em buscas de médicos. Ele possui vários aspectos sugeridos no projeto como a busca por especialidade e é possível buscar em todo o Brasil (ou um país). Porém no Help Saúde se busca por médicos com o intuito do usuário marcar consulta para os mesmos. Para um médico entrar na sua lista ele deve pagar um valor específico, tornando um sistema de divulgação para médicos apenas. Por ser um site no qual o médico tem que pagar seu anuncio, não seria possível encontrar profissionais de unidades públicas, pois existiria pouco interesse destes médicos pagarem seus próprios anúncios. Como a página inicial sugere, é uma busca por profissionais de saúde, como na figura 2.2, não chega a abranger mais do que isto. Em alguns casos existem usuários usando o sistema colocando uma clínica como se fosse um médico, porém são dados de uma clínica como se fosse um médico só (ou uma

pessoa).

Figura 2.2: Tela inicial para buscar por médico do Help Saúde

2.3 Busca AMS

É um aplicativo bem similar ao que esta sendo desenvolvido neste projeto com algumas diferenças em especial[8]. Este aplicativo foi publicado pela Petrobras S.A., e tem como objetivo permitir ao usuário consultar todos os médicos credenciados à AMS - Assistência Multidisciplinar de Saúde. Ou seja, a diferença em questão é que este aplicativo tem foco nos médicos credenciados de um grupo particular enquanto o que está em desenvolvimento no projeto são clínicas médicas tanto públicas quanto particulares.

São disponibilizadas as seguintes funcionalidades no aplicativo: armazenar no aplicativo os dados de credenciados escolhidos pelo beneficiário; acesso rápido ao call center da AMS e link para o site da AMS; permite adicionar dados dos profissionais e instituições de saúde à agenda do celular; visualização através de mapa, da localização geográfica do credenciado e sugestões de trajetos a partir da localização do seu dispositivo; Emergência: localiza os credenciados mais próximos à sua localização, que realizam atendimento de emergência; busca informações dos credenciados a partir de palavras chave; guarda as últimas buscas realizadas para consultas futuras; o usuário pode limpar o histórico das buscas realizadas.

As figuras 2.3 e 2.4 representam as telas de exibição de resultados de buscas feitas no aplicativo Busca AMS.

Figura 2.3: Tela de resultado da busca do aplicativo Busca AMS

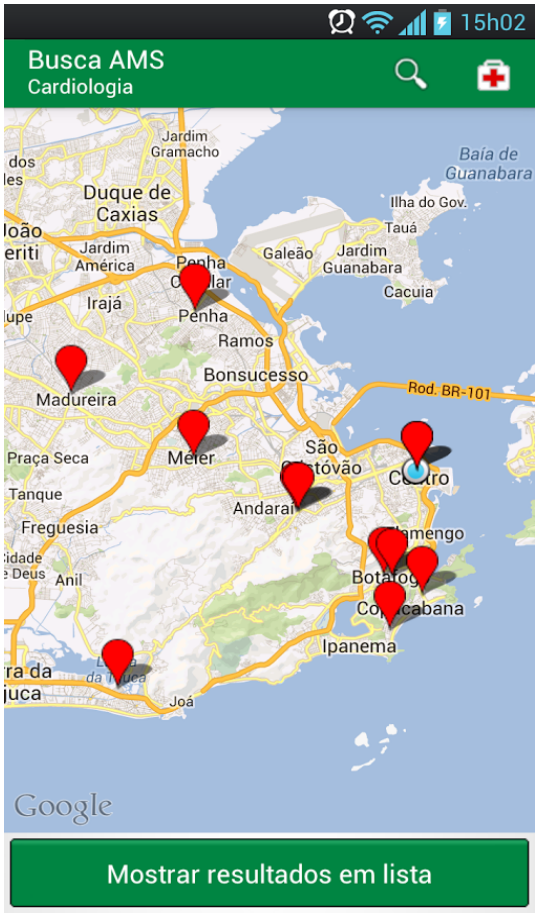


Figura 2.4: Tela de exibição de mapa com credenciados do aplicativo Busca AMS

2.4 Conclusão sobre as ferramentas

Foi possível notar a partir das ferramentas aqui apresentadas, que nenhuma delas soluciona o problema já descrito. Através de uma verificação geral destas ferramentas, nota-se que na maioria dos casos elas se limitam a uma área de atuação específica como: unidades credenciadas à um plano de saúde, unidades dispostas em uma região geográfica específica, unidades sem informações dos tipos de atendimentos possíveis e ao tipo da unidade (particular e pública). Existem dois desafios importantes para a solução dos problemas citados acima, são eles: conseguir informações de todas as unidades do país (localização, tipos de atendimento, telefone, horário de atendimento, etc) e manter estas informações atualizadas.

Capítulo 3

Tecnologias Utilizadas

Neste capítulo serão apresentadas as tecnologias que foram utilizadas para implementação do sistema. Também será descrito com qual finalidade cada uma destas tecnologias foi usada. De acordo com a imagem 3.1 é possível verificar de maneira simples qual utilização cada uma destas tecnologias teve no desenvolvimento do sistema. A pilha de software da imagem divide as tecnologias em dois grupos: Cliente e Servidor representando respectivamente as tecnologias que foram utilizadas para o desenvolvimento do servidor de processamento de dados e do ambiente de interação do usuário.

]

Figura 3.1: Imagem representativa da pilha de softwares utilizada no desenvolvimento do sistema. Segue em ordem: Cordova, HTML, Bootstrap, Javascript, PHP, Google API, MariaDB. O Sheer não possui logo definido, contudo entraria tanto no servidor quanto no aplicativo.

3.1 PHP

O PHP é uma linguagem de programação de script código aberto que atualmente é muito utilizada para desenvolvimento de sistemas web [13]. O motivo desta grande utilização em ambientes web, é porque ele pode ser embutido dentro de páginas HTML. Ele foi utilizado para o desenvolvimento do servidor do sistema. Em especial, foi usado para criação das funcionalidades que irão tratar dados e fazer modificações em banco.

Para a utilização da linguagem PHP é necessário um servidor HTTP para processar as requisições. No caso do projeto foi utilizado o Apache através do Xampp que é um ambiente de desenvolvimento para a criação de sistemas web. Após a instalação do Apache será possível acessar seu sistema de maneira local pelo navegador através do endereço 127.0.0.1.

Como o PHP tem uma interação fácil com HTML e pelo servidor HTTP ser acessado pelo navegador, para o servidor saber quando um script PHP deve ser processado ele irá buscar pela inicialização do script que é "<?php" e será processado até finalização do script que é "?>", a partir da inicialização do script, todo o código será processado como PHP. Com isso é permitido que um arquivo tenha HTML

e PHP nele.

Contudo, é muito complexo criar um sistema do zero pois seria necessário criar vários controles que são padrões em muitos sistemas para depois de todos estes controles partir para o projeto em si. Mas para solucionar este problema, é possível fazer a utilização de frameworks que foram desenvolvidos para ter uma base de controles aonde o desenvolvedor pode começar seu projeto apenas utilizando estas funcionalidades, podendo focar apenas no seu projeto. De acordo com as tabelas 3.1 e 3.2 é possível verificar as funcionalidades que o framework Sheer possui e compará-las com os outros frameworks PHP conhecidos. Para a construção destas tabelas foram utilizados alguns frameworks PHP mais conhecidos ou utilizados. Estas informações dos frameworks (excluindo o Sheer) foram retiradas da wikipedia na página de comparação de frameworks [17].

Framework	Ajax	MVC	i18n ou L10n	ORM	Teste	Migração de Banco
Cake PHP 3	Sim	Sim	Sim	Sim	Sim	Sim
Drupal	Sim	Não	Sim	Sim	Sim	Sim
Laravel	Sim	Sim	Sim	Sim	Sim	Sim
CodeIgniter	Sim	Sim	Parcial	Não	Sim	Sim
Zend2	Sim	Sim	Sim	Sim	Sim	Sim
Sheer	Sim	Sim	Não	Sim	Não	Sim

Tabela 3.1: Tabela 1 de comparação de Frameworks PHP com o Sheer incluso

Framework	Template	Database	Validação Form.	RAD	Mobile
Cake PHP 3	Sim	Sim	Sim	Sim	Sim
Drupal	Sim	Sim	Sim	Não	Sim
Laravel	Sim	Sim	Sim	Sim	Não
CodeIgniter	Sim	Sim	Sim	Sim	Sim
Zend2	Sim	Sim	Sim	N/S	N/S
Sheer	Sim	Sim	Sim	Sim	Sim

Tabela 3.2: Tabela 2 de comparação de Frameworks PHP com o Sheer incluso

Uma explicação sobre as colunas das tabelas será feita agora para melhor entendimento. Ajax é um script do lado do cliente que entra em contato ou recebe informações do servidor. MVC é o modelo de projeto com base em três tipos de objetos que são: Modelo, classe e Controle. A coluna i18n e L10n significam que os frameworks dão suporte a várias linguagens padrões como pt (Português - Portugal) ou pt-br (Português - Brasil). A coluna ORM, esta representa uma técnica de desenvolvimento utilizada para reduzir a não pendência da programação orientada aos objetos utilizando bancos de dados relacionais [18]. A coluna teste, significa se o framework possui algum teste automatizado. A coluna de migração de banco verifica se o framework permite mudança no schema utilizado no projeto. A coluna template verifica se é permitido a criação de páginas dinamicamente de acordo com o conteúdo. A coluna Database verifica é feito conexão com banco de dados ou qualquer tipo de sistema para guardar informações. A

coluna RAD verifica se o framework é preparado para desenvolvimento mais rápido, como exemplo, um software que tem menos enfase em planejamento e mais em desenvolvimento. A coluna Mobile verifica se o framework vai oferecer tratamento diferente para acessos ao sistema vindos de dispositivos mobile. Mais sobre o framework Sheer será falado na seção dele.

Com base nas tabelas, pode-se notar que existem vários frameworks com a mesma capacidade ou até maior do que o escolhido para desenvolvimento do projeto. Porém, é visível que o Sheer não tem grandes perdas quando relacionado aos outros. Portanto, por conhecer melhor o Sheer e pelas características dele mostradas nas tabelas 3.1 e 3.2, foi escolhido para ser o framework do projeto.

3.2 MariaDB

O MariaDB é um sistema de gerenciamento de banco de dados que surgiu de uma ramificação do projeto Mysql [14]. Ele foi utilizado no sistema para a manutenção do banco de dados do servidor. A principio o Mysql tinha sido escolhido para ser o gerenciador de banco de dados do sistema, porém após relatos do MySql ser mais inconsistente que o sua ramificação MariaDB foi decidido dar continuidade com ele.

Para utilização do banco de dados MariaDB é necessário um servidor MySql, assim como no caso do PHP foi utilizado o Xampp que é um ambiente de desenvolvimento que já trás o MariaDB. A utilização do banco pode ser feita através de um gerenciador de banco de dados como um MySql Workbench ou por qualquer sistema que tenha capacidade de acessar o servidor MySql. O próprio PHP fornece métodos para se conectar a um banco MySql / MariaDB como: mysql_connect(). Após a criação da conexão é possível utilizar o banco de maneira normal utilizando comandos como CREATE, INSERT, UPDATE, DELETE e outros.

3.3 Javascript

O Javascript é uma linguagem de programação interpretada. Originalmente foi implementado para rodar no lado do cliente sem passar por processamento no servidor através dos navegadores [15]. No sistema esta linguagem foi utilizada para tratar a interação do usuário para com o sistema.

Para utilização do Javascript basta ter um arquivo HTML que possa ser interpretado por um navegador e neste incluir um script Javascript através da tag <script></script>. Nesta tag você pode passar o endereço de um string ou colocar o script dentro dela. Através do Javascript é possível selecionar, manipular e inserir elementos HTML na página atual, além de permitir modificações de ações padrões do navegador como o que acontece quando se clica em um link <a>.

Através do Javascript também é possível fazer requisições a outros servidores ou páginas. Como no exemplo, uma página que possui uma lista de produtos a venda e estes produtos são atualizados sem que o usuário tenha que fazer qualquer coisa, isto ocorreria através do Javascript fazendo requisições a outra página para poder atualizar as informações que estão sendo visualizadas pelo usuário.

3.4 Bootstrap

O Bootstrap é uma biblioteca, composta por CSS e Javascript, para auxiliar na construção de websites com finalidade de serem acessados tanto por dispositivos desktop quanto mobile [16]. Foi utilizado apenas o seu módulo CSS no desenvolvimento do sistema para que o site tenha aparência de um aplicativo móvel.

Para utilização do Bootstrap é necessário apenas a tag <link> como esta: ”<link rel=”stylesheet” href=”https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-beta/css/bootstrap.min.css”>”, a partir disto é possível utilizar as classes do Bootstrap. Para explicar como o Bootstrap colabora no desenvolvimento de páginas para dispositivos móveis, será demonstrado um pouco das classes de divisão de tela. Com a biblioteca é possível separar a tela em 12 partes através das classes col-XX-1 até col-XX-12, com esta separação é possível criar a interface já planejada em um celular e ela funcionar igualmente bem em um tablet ou até um desktop aonde a resolução é maior. Na figura 3.2 é possível visualizar melhor a divisão de telas. As letras do meio da classe representadas na figura como ”md”, podem variar em: ”xs” para resoluções muito pequenas (smartphones), ”sm” para resoluções pequenas (alguns smartphones e tablets), ”md” para resoluções médias (alguns tablets e desktops) e ”lg” para resoluções grandes (desktops).

Figura 3.2: Exemplo das classes de divisão do Bootstrap

3.5 Google APIs

A Google disponibiliza várias APIs para utilização de desenvolvedores que facilitam e melhoram algumas áreas dos sistemas dos mesmos. Como exemplo, este projeto utilizou duas APIs da Google que são: Google Maps e Google Place / Geocode, para exibição e interação do usuário com o mapa e localização dos pontos respectivamente.

Para utilizar o Google Maps API Javascript [19] é necessário inserir um script na página ou nas páginas que serão utilizadas ”<script src=”https://maps.googleapis.com/maps/api/js” async defer></script>” este irá carregar as funcionalidades que podem ser utilizadas no Google maps, de exibir um mapa até inserir pontos ou caminhos no mesmo.

Para inserir um mapa na página basta chamar o código da figura 3.3, neste caso o primeiro parâmetro do método é o elemento HTML onde o mapa será inicializado. O retorno do método é o objeto do Google Maps, que no caso é retornado para a variável ”map”. Esta variável poderá ser chamada durante a execução do código para realizar alterações no mapa, como: inserção de marcadores, exibição de caminhos até locais específicos, modificação de zoom, posição de visualização do mapa, tipo de terreno exibido e outros.

```
map = new google.maps.Map(document.getElementById('map'), {
  center: {lat: -34.397, lng: 150.644},
  zoom: 8
});
```

Figura 3.3: Código para inicialização do Google Maps em uma página

Para utilizar os métodos do Google Geocode [20], é necessário apenas fazer uma requisição HTTP para o endereço "https://maps.googleapis.com/maps/api/geocode/json" com o parametro address (Exemplo: Rio+das+Ostras+RJ) para buscar por algum local. Para utilizar o Google Places [21], é necessário apenas fazer uma requisição para "https://maps.googleapis.com/maps/api/place/nearbysearch/json" com os parametros location (latitude;longitude), radius(raio de busca) ou rankby(distance - para os mais próximos); no caso o places também é possível passar o parametro "type" para definir que tipo de estabelecimento esta procurando.

3.6 Sheer

O Sheer é um framework PHP desenvolvido pela empresa BG Studios. Um dos intuitos do framework é transformar todas as funcionalidades desenvolvidas pelo programador em uma API. Ou seja, todas as funcionalidades desenvolvidas para serem utilizadas pelo framework podem ser utilizadas através de requisições http. Além disso, o framework colabora para o desenvolvimento do projeto no formato MVC (modelo, visão e controle). Como já apresentado na seção do PHP as tabelas de framework 3.1 e 3.2, é possível ver que apesar de ser um framework desconhecido, ele não sai perdendo em comparação aos mais utilizados.

O Sheer também ajuda no controle de páginas, e na simplificação de acesso / modificação das informações do banco de dados. Através de métodos padrões, permite ao programador fazer acessos e modificações.

Através do Sheer todos as classes criadas no projeto podem estender as diversas classes padrões do framework, e estas tem por definição serem utilizadas como API. Existem três tipos de classes: Action, Renderer e DataProvider, que representam respectivamente controle, visualização e módulo. Quando uma classe criada estender uma destas classes do framework, ela automaticamente poderá ser acessada como uma API. Explicando melhor com um exemplo, ao criar a classe ClasseTesteA e estender à classe Action do framework ela poderá ser chamada através de uma requisição http como esta "http://127.0.0.1/Action/ClasseTesteA". Esta requisição irá retornar o resultado em formato de JSON.

3.7 Cordova

O Cordova é um sistema que pode produzir um aplicativo para vários sistemas operacionais mobile diferentes através de um site [23]. Este sistema roda no node.js que é um Javascript multiplataforma executado no servidor (e não no cliente como o normal) que permite reproduzir scripts Javascript diretamente sem o uso do navegador.

Para a utilização do cordova é necessário que instale o node.js e depois execute o ”comando npm install -g cordova” no cmd. Posteriormente é possível criar projetos através do comando ”cordova create”. Após isto é possível inserir vários dispositivos no projeto cordova, para que ao compilar sejam geradas versões do aplicativo para estes dispositivos, como exemplo: ”cordova add platform android” irá adicionar a plataforma Android. Após a criação do projeto, o cordova irá gerar uma pasta com os arquivos básicos para geração do seu site em forma de aplicativo. É possível inserir na pasta ”www” os arquivos do site. Ao finalizar a inserção dos arquivos, só será necessário rodar o comando ”cordova build” para gerar o aplicativo para ser usado no dispositivo móvel.

Uma informação útil é que o cordova não processa arquivos PHP. O motivo de não processar é porque ele não é um servidor PHP ou não contém um nele. O que ele faz para ”transformar” um site em aplicativo, na verdade, é chamar o navegador padrão do dispositivo (como exemplo no Android o WebKit) e irá exibir suas páginas que estavam na pasta ”www” neste navegador. Apesar disto, ainda é possível utilizar javascript e com isso permitir ao desenvolvedor obter informações que um PHP conseguiria através de requisições HTTP.

3.8 PWA

O termo Progressive Web App é usado para representar páginas web e sites inteiros que se parecem como aplicativos de sistemas móveis, segue na figura 3.4 uma representação de elementos importantes para sites baseados no PWA, os elementos não são obrigatórios para o funcionamento. Estas páginas não pegam apenas a aparências de aplicativos, mas também procuram ter velocidade ao responder requisições do usuário. Devem ser responsivos o suficiente para funcionar num celular, tablet ou até mesmo desktop. Deve-se funcionar normalmente mesmo sem internet, carregando os dados previamente para caso existir uma instabilidade na internet o aplicativo dar continuidade a requisição do usuário. Procura sempre ter as tecnologias mais novas para facilitar o uso do seu site, assim como: camera, microfone, leitor biométrico (que alguns dispositivos já possuem), entre muitas outras tecnologias que estão disponíveis em celulares. Com todas estas melhorias para que o usuário tenha uma utilização do site mais fluida, o sistema em questão, além de poder ser acessado por várias plataformas diferentes, vai ter menos chances de que o usuário saia dele sem cumprir seu objetivo.

Figura 3.4: Gráfico de elementos encontrados em sites baseados no PWA

Capítulo 4

Projeto

Neste capítulo será introduzido o projeto do sistema de buscas para unidades de saúde. Este sistema irá permitir ao usuário buscar unidades de saúde de acordo com sua necessidade e localização . Algumas particularidades do sistema serão apresentadas, assim como: arquitetura, modelagem e os requisitos do sistema.

Primeiramente será apresentado o aplicativo proposto para solucionar o problema. Logo após será apresentado a modelagem do sistema, onde será possível observar os seus requisitos e a construção dos diagramas referentes à ele.

4.1 Visão Geral do Sistema

Será explanado aqui o que foi planejado para o projeto de controle e exibição de informações sobre unidades de saúde através de um sistema colaborativo. Este tem como principal objetivo satisfazer as necessidades do usuário em relação a localizações de unidades de saúde (clínicas, hospitais, laboratórios e outros). Além disso, o sistema vai precisar do usuário para fazer atualizações através do banco colaborativo, passando as modificações para outros usuários.

O projeto se baseia em dois sistemas: um é o sistema web que vai fazer a organização e controle das informações (unidades de saúde e especializações); o outro é um sistema móvel que o usuário vai interagir fazendo suas buscas e customizações, sendo respectivamente o servidor e cliente.

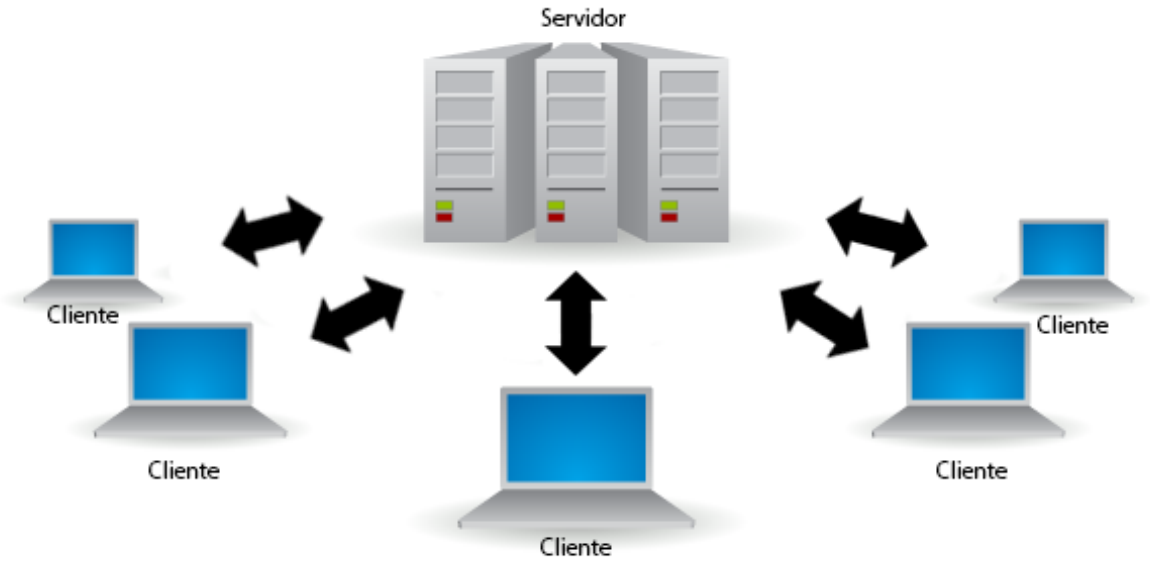
4.1.1 Servidor Web

O servidor web será responsável por fazer toda a gestão do conteúdo inserido e a base de dados. A base contém as informações que serão utilizadas no aplicativo. A principio o preenchimento do banco vai acontecer de acordo com a localidade dos usuários ativos no aplicativo. Na primeira vez que o usuário buscar informações de uma cidade que não possui registros, será cadastrado no servidor as informações de hospitais encontrados no Google Maps. Este cadastro ocupa um tempo insignificante, como será

explicado no capítulo sobre a implementação. Portanto não vai gerar insatisfação para o usuário. Esta funcionalidade foi realizada apenas no servidor web.

O servidor deve estar pronto para receber informações do aplicativo como modificações em unidades cadastradas ou novas unidades através de requisições web. Além disso, irá prover os usuários com a mesma funcionalidade que existe no aplicativo móvel, deixando assim de ser apenas uma API de controle de banco de dados. Isto significa que o servidor também poderá ser acessado por usuários, porém no formato de um site.

Na figura 4.1 é demonstrado como funciona o banco colaborativo do servidor através do fluxo de dados representado pelas setas. Esta é uma demonstração de como os usuários (cliente) irão tanto pegar informação quanto escreve-la no servidor. Com isso, concluímos que todos os usuários irão poder visualizar as modificações realizadas por outros usuários.



]

Figura 4.1: Representação de distribuição de informações entre Servidor e Clientes

4.1.2 Aplicativo Móvel

O aplicativo é um sistema que o usuário fará uso direto (ele mesmo vai interagir com o sistema). Este usará as informações das unidades de saúde cadastradas diretamente do servidor. Além disso, o aplicativo fornecerá as funcionalidades de busca e customização das unidades de saúde. O aplicativo permitirá que o usuário insira ou modifique informações diretamente no servidor.

4.2 Especificação de Requisitos

Nesta seção serão apresentadas as especificações dos requisitos para o desenvolvimento do sistema de buscas de unidades de saúde. As especificações do sistema serão demonstradas através de requisitos funcionais e não-funcionais, diagrama de classe de análise e casos de uso do sistema.

4.2.1 Requisitos

Serão apresentados os requisitos utilizados para planejamento e desenvolvimento do sistema. Serão apresentados nas tabelas 4.1 e 4.2 todos os requisitos funcionais e não-funcionais.

Requisitos	Descrição
Localidade atual do usuário	Através de coordenadas dadas pelo GPS exibir o local em que o usuário se encontra.
Localidade das unidades de saúde	Permitir ao usuário visualizar em tela o local e informações sobre o endereço da unidade de saúde.
Exibir caminho pelo mapa até o destino	O sistema deverá mostrar o caminho de onde o usuário se encontra, através de coordenadas fornecidas pelo sistema e GPS, até uma possível localidade destino.
Exibir informações da unidade de saúde	Permitir ao usuário à qualquer momento de suas buscas ou visualização de informação conseguir obter informações (Nome, endereço, telefone) das unidades de saúde em sua tela.
Busca manual pelo mapa	Será possível para o usuário visualizar todas as unidades cadastradas através do aplicativo por uma exibição de mapa.
Busca automática por proximidade	Este requisito será conhecido pelo usuário como busca emergencial, esta funcionalidade tratará de localizar uma unidade com atendimento emergencial mais próxima possível.
Busca por especialidade	Neste requisito o usuário terá acesso a uma busca por unidades próximas ou qualquer uma que possua a especialidade buscada.
Customização de informações	Fazer atualização ou cadastro de unidades de saúde diretamente no servidor.

Tabela 4.1: Tabela de requisitos funcionais.

Usabilidade	O uso do aplicativo para o usuário deve acontecer de maneira simples e intuitiva. Deve-se levar em conta a exibição de forma limpa das informações e de ações que o usuário pode fazer com o aplicativo. As telas devem ser feitas para que o usuário não demore mais de 10 segundos para encontrar o que deseja.
Tratamento de erros	O sistema deve ser capaz de tratar qualquer erro encontrado durante sua execução, não deixando o usuário em espera. Não pode demorar mais de 30 segundos para interromper uma execução em falha e dar retorno ao usuário.

Tabela 4.2: Tabela de requisitos não funcionais.

4.2.2 Diagrama de casos de uso

O diagrama de casos de uso auxilia a comunicação entre quem está criando o sistema com quem vai utilizá-lo através da descrição de um cenário que irá mostrar a interação do usuário com o sistema. O diagrama de casos de uso feito para o projeto segue na figura 4.2.

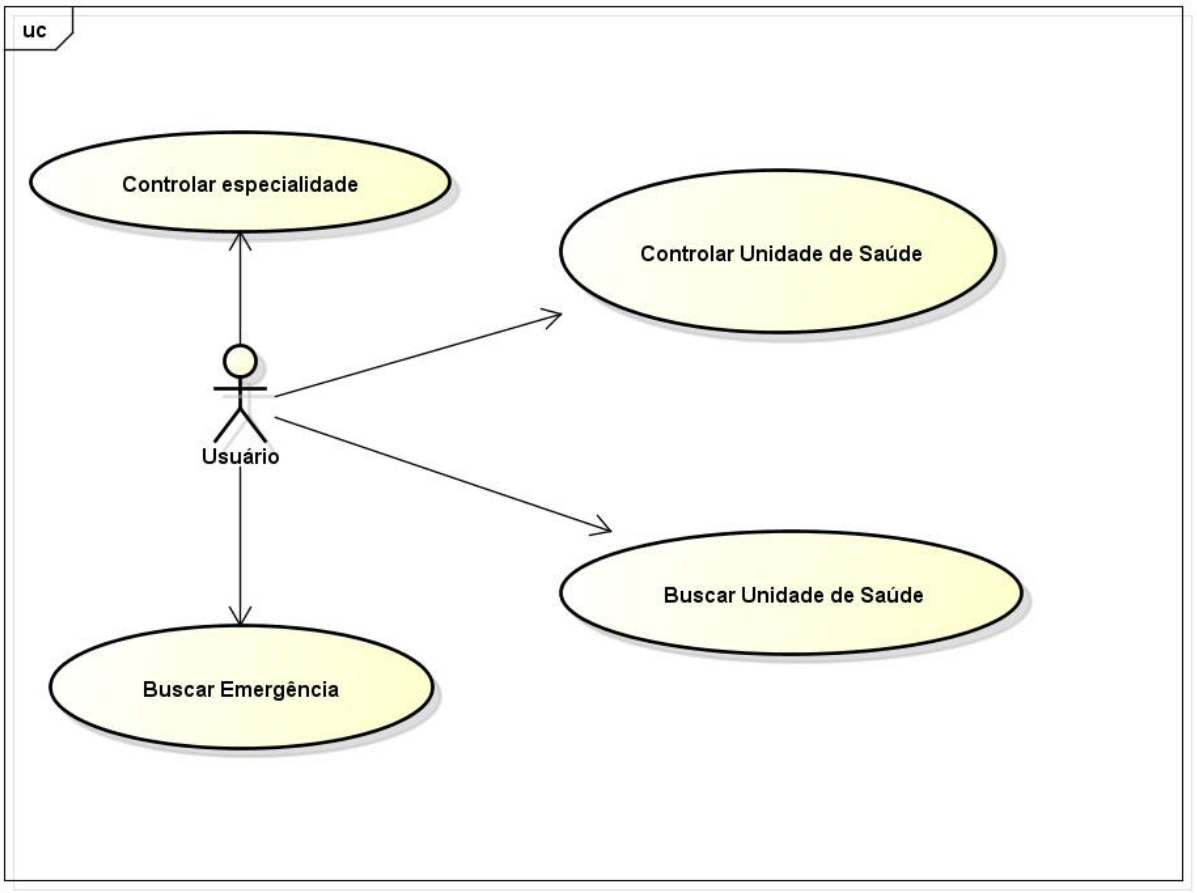


Figura 4.2: Diagrama de casos de Uso

Agora serão descritos os casos de uso do projeto, levando em conta a figura 4.2 já apresentada. Controlar especialidade é a funcionalidade que vai permitir ao usuário adicionar / remover uma especialidade a uma unidade de saúde, ou seja, vincular uma especialidade à uma unidade. É importante relatar que novos tipos de especialidades não podem ser inseridos por usuários.

Controlar especialidade

Ator	Usuário
Fluxo Principal	1. O usuário escolhe adicionar / remover uma ou várias especialidade à uma unidade 2. O sistema exibe as especialidades disponíveis e as já adicionadas 3. O usuário escolhe as especialidades que ficaram vinculadas à unidade 4. O sistema valida e atualiza as informações 5. O sistema retorna o usuário para o mapa

Controlar Unidade de Saúde é a funcionalidade que vai permitir ao usuário adicionar novas unidades de saúde.

Controlar unidade de saúde

Ator	Usuário
Fluxo Principal	1. O usuário escolhe a opção de adicionar uma nova unidade de saúde 2. O sistema exibe o mapa para o usuário indicar o local da unidade 3. O usuário indica o local que a unidade de saúde está 4. O sistema exibe as possíveis informações (nome da unidade, endereço, especialidades, emergência) a serem inseridas 5. O usuário preenche e envia as informações 6. O sistema valida e atualiza as informações 7. O sistema retorna o usuário para o mapa

Buscar unidade de saúde representa todos os tipos de buscas abertas ao usuário em que ele tem que ter alguma interação na escolha das opções, como por exemplo, escolher o tipo de especialidade, ou a localidade em que ele deseja buscar.

Buscar unidade de saúde

Ator	Usuário
Fluxo Principal	1. O usuário escolhe sua cidade e uma especialidade 2. O sistema exibe as possíveis unidades de saúde com suas informações (local no mapa, endereço escrito, telefone e nome da unidade)

Buscar Emergência é uma busca para o usuário encontrar a unidade emergencial mais próxima. Neste ele deve possuir apenas uma interação com o sistema, com a preocupação de ser mais rápido para

o usuário.

Buscar Emergência

Ator	Usuário
Fluxo Principal	1. O usuário pede para buscar uma unidade emergencial 2. O sistema exibe as unidades emergenciais mais próximas, com base na localização atual do usuário 3. O usuário escolhe para qual deseja ir 4. O sistema exibe o caminho até o local escolhido

4.2.3 Diagrama de classes de análise

O diagrama de classes de análise tem como objetivo demonstrar o domínio do projeto através das classes. Sendo isso, este tipo de diagrama explica o funcionamento dos elementos (objetos) do sistema, quais ligações e informações que o sistema irá gerenciar.

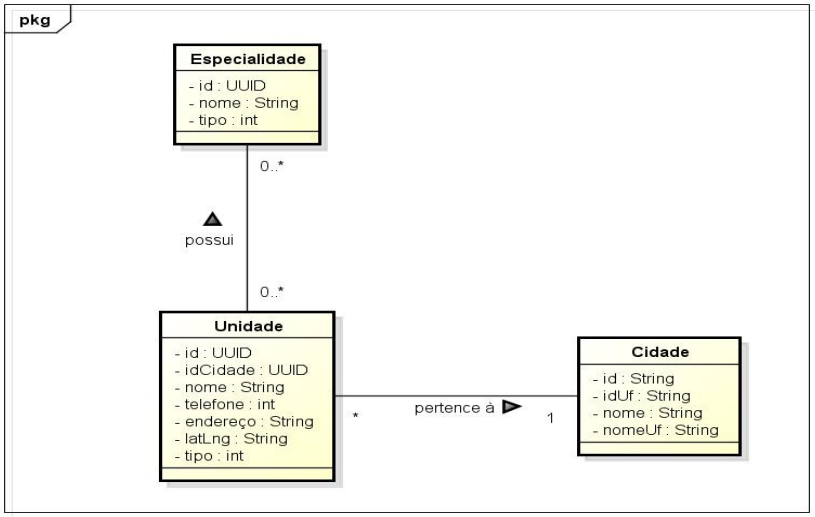


Figura 4.3: Diagrama de classe de análise

4.3 Projeto de Software

4.3.1 Diagrama de classes de projeto

O diagrama de classes de projeto tem como objetivo demonstrar as classes de um sistema abordando vários detalhes da implementação dele próprio. Assim como seus atributos, métodos, modelo utilizado e os relacionamentos entre os objetos.

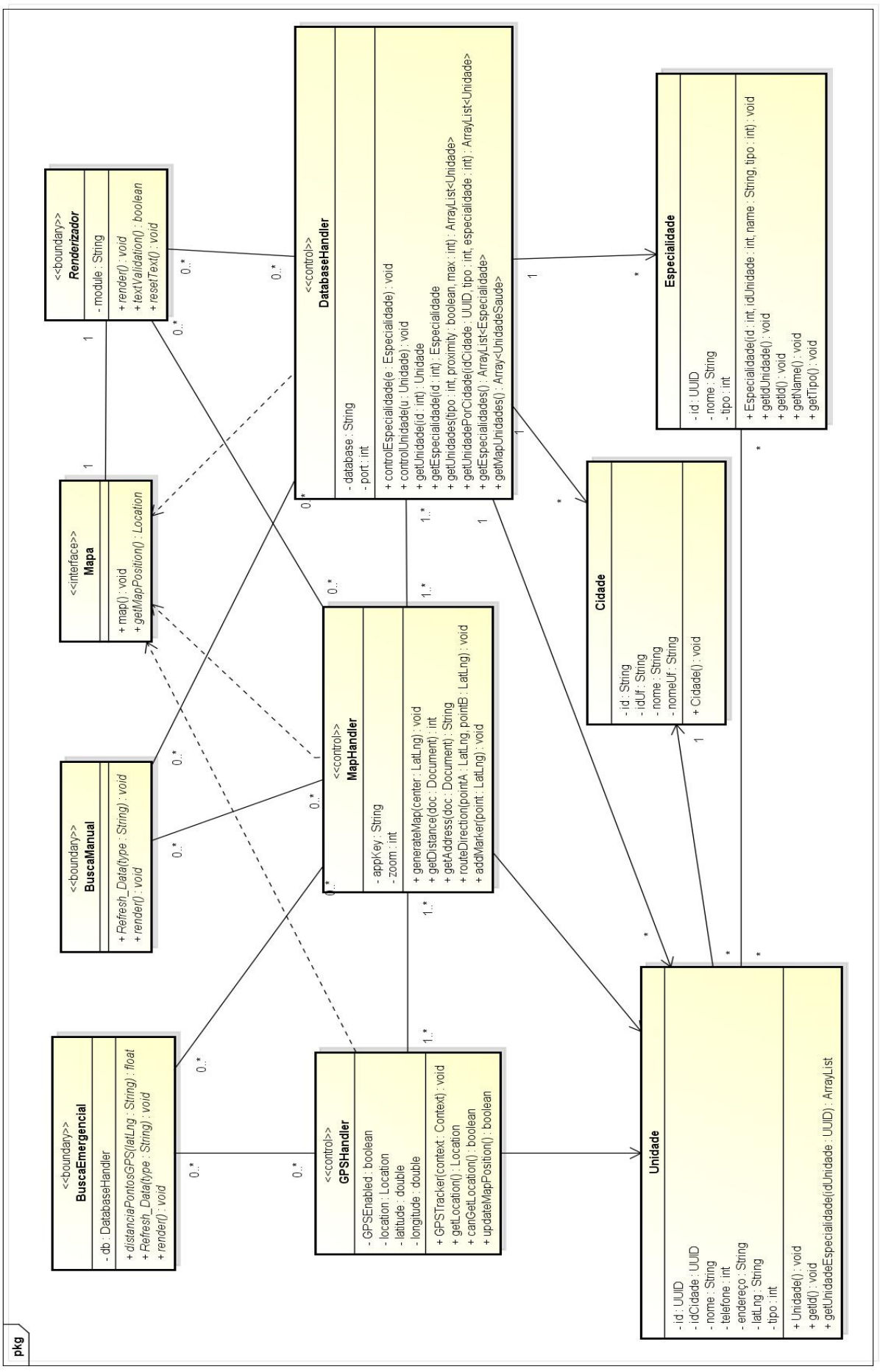


Figura 4.4: Diagrama de classe de projeto

O modelo MVC (Model View Controller) foi utilizado no desenvolvimento do projeto. Ele consiste em três tipos de classes, que são: Models, Views e Controllers. O uso deste modelo tem como objetivo geral melhorar o reúso de código e a formatação do código. Explicando os tipos das classes:

Função do Controller: Responsável por identificar e indicar às classes de model e view o que devem fazer, em relação ao o que o usuário irá visualizar e com que dados irá interagir. São as classes: GPSHandler, MapHandler e DatabaseHandler.

Classes de View

Agora, serão explicadas as classes do tipo view do projeto. Todas as classes deste tipo possuem o mesmo objetivo que é ter o controle da exibição e iteração do usuário com a tela. Todas as classes deste tipo contarão com um método chamado render, que tem como objetivo exibir a tela ao usuário. Além de outros métodos que irão variar de acordo com a página e suas necessidades.

Alguns dos outros métodos importantes servem para: exibição do mapa e exibição de formulários. As classes do tipo view do projeto são: BuscaEmergencial, BuscaManual, Mapa, e Renderizador.

Classes de Model

Agora serão explicadas as classes do tipo model do projeto. Todas as classes deste tipo, assim como as classes do tipo view, possuem um mesmo objetivo compartilhado entre si que neste caso é a manipulação e validação dos dados. Cada classe model é referente a uma estrutura diferente, por exemplo: unidade, especialidade e cidade. Cada uma delas sabe como adicionar ou alterar suas próprias informações.

As classes do tipo model do projeto são: Unidade, Especialidade e Cidade.

Classes de Control

Agora, serão explicadas as classes do tipo control do projeto. As classes deste tipo tem como objetivo indicar às classes do tipo model e view o que devem fazer, o que o usuário vai visualizar ou com que dados ele pode interagir.

Databasehandler: Classe necessária para fazer a comunicação entre o servidor e o aplicativo, para buscas e modificações de dados do servidor.
Serão listados na tabela 4.3 os métodos de importância da classe de controle de banco de dados:

Função	Descrição
controlUnidade	Método para controle de dados em banco das tabelas de unidade e unidadeEspecialidade.
controlEspecialidade	Método para controle de dados em banco da tabela de especialidade.
GetUnidades	Função para buscar unidades de um tipo específico. É possível procurar apenas as mais próximas. Também é possível limitar o resultado.
GetUnidadesPorCidade	Função para buscar unidades de uma cidade específica. É possível procurar de acordo com uma especialidade. Outra capacidade deste método é que, ao buscar uma cidade à qual não possui unidade cadastrada, o sistema automaticamente buscará no Google Maps por hospitais através de buscas com a palavra ”hospital”ou ”clínica”. E então adicionará no sistema para que o usuário possua uma base de informações minimamente usável. E assim garantindo que o sistema vai ter pelo menos as informações que o Google também possui.

Tabela 4.3: Tabela de métodos da classe Databasehandler.

GPSHandler: Classe criada para buscar informações da posição do usuário, como latitude e longitude. Serão listados na tabela 4.4 os métodos de importância da classe de controle do GPS.

Função	Descrição
getLocation	Função responsável para a busca de informações da posição do usuário atual através do GPS. Além disso esta função é capaz de tratar erros, caso não haja disponibilidade de internet e GPS.
canGetLocation	Função que verifica se é possível utilizar o GPS, ou seja, verifica se o GPS e a internet estão habilitados.

Tabela 4.4: Tabela de métodos da classe GPSTracker.

MapHandler: Classe criada para fazer a comunicação com os servidores da Google para a geração do mapa e a utilização das funcionalidades da API. Além disso é capaz de obter informações como nome da rua, locais próximos ou até nome do local.
Serão listados na tabela 4.5 os métodos de importância da classe de criar rotas.

Função	Descrição
generateMap	Método para chamar a função de inserção do mapa na interface.
routeDirection	O aplicativo faz uma requisição HTTP ao servidor da Google com a rota de uma origem e um destino. Este documento (como é chamada a resposta da requisição) possui as informações em um objeto javascript (JSON). Essas informações são baseadas em pontos do mapa (GPS) que quando ligados formam o caminho até o destino.
getDistance	Função que pega a informação da distancia do caminho entre os pontos.
getAddress	Função que pega a informação do nome do endereço de um ponto do GPS.
addMarker	Função para inserir um marcador no mapa, necessário para exibição de unidades ou da localização do usuário.

Tabela 4.5: Tabela de métodos da classe RouteDirection.

4.3.2 Diagrama Entidade Relacionamento

Será apresentado o diagrama entidade relacionamento lógico e físico que foi planejado e desenvolvido para o banco de dados. O diagrama da figura 4.5 apresenta as seguintes entidades que serão explicadas na tabela 4.6:

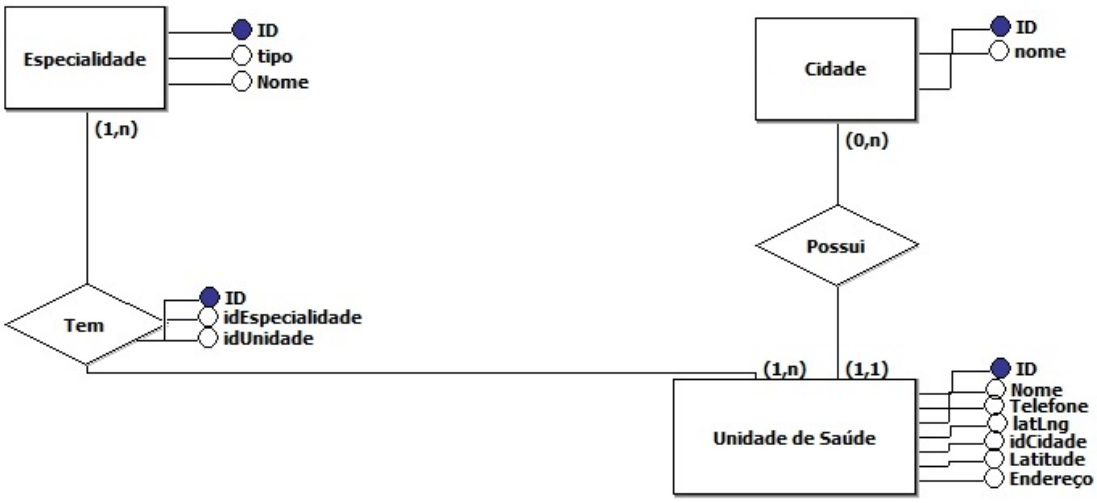


Figura 4.5: Diagrama entidade relacionamento - Lógico

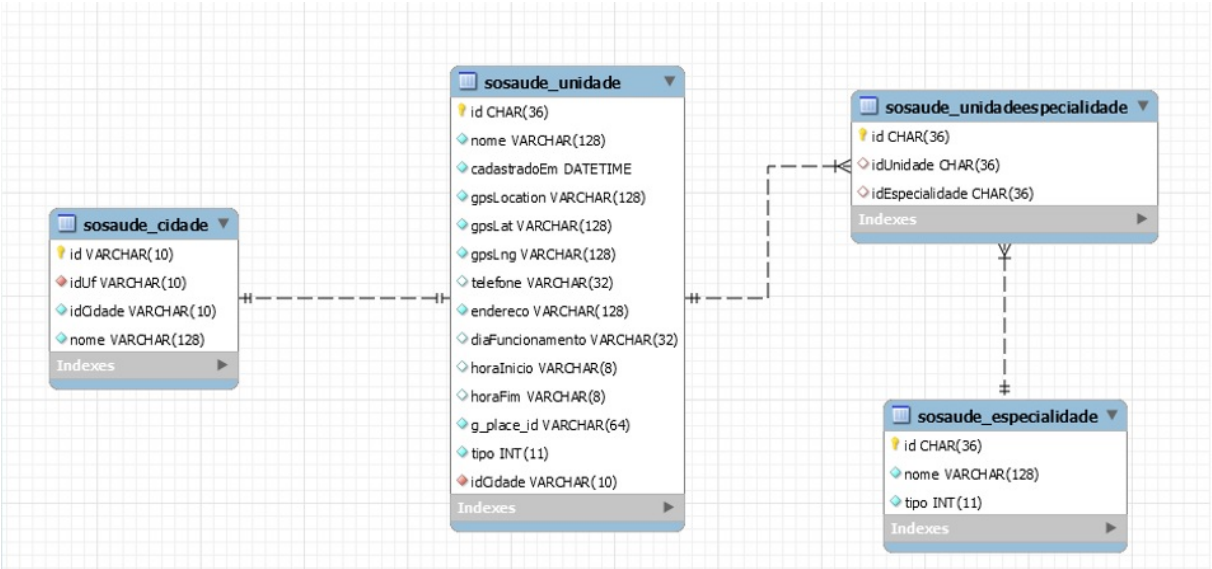


Figura 4.6: Diagrama entidade relacionamento - Físico

Entidade	Descrição
Unidade de Saúde	Esta entidade representa, como o nome sugere, as unidades de saúde. Cada linha levará as informações de posição de acordo com o GPS, seus telefones, endereço, nome e cidade aonde se localiza.
Especialidade	Esta irá manter todas as informações das especialidades cadastradas no sistema. As especialidades de uma unidade são quaisquer tipo de atendimento que pode ser considerado diferenciado. Como exemplo: alergista é uma especialidade de atendimento médico, exame de sangue é uma especialidade do tipo exame.
EspecialidadeUnidade	Representa a relação entre as tabelas Unidade de Saúde e Especialidade, e irá guardar informação de quais especialidades estão à disposição em uma unidade médica.
Cidade	Representa um conjunto de dados que possuem os nomes das cidades e estado (o qual a cidade pertence). Estes dados foram tirados do site do IBGE através de uma planilha EXCEL fornecida por eles.

Tabela 4.6: Tabela de descrição de entidades do banco de dados.

Capítulo 5

Implementação

Neste capítulo serão introduzidos detalhes sobre a implementação do sistema. Em geral, as funcionalidades mais complexas utilizadas no sistema serão explicadas e detalhadas nas seções abaixo. Além disso, serão explicados os testes realizados para decidir melhor como implementar as funcionalidades. Todas estas informações servem para a melhor compreensão do sistema.

5.1 Google APIs

O sistema foi planejado para que os próprios usuários pudessem atualiza-lo. Porém, foi possível perceber que se o sistema não possui unidades para que os usuários possam buscar, não vai existir motivação para o usuário entrar no sistema. E com isso, não haverá informação inserida pelo usuário. Portanto, chegou-se a conclusão que o aplicativo precisa de uma base inicial. Para a base inicial foi utilizado a Google, que fornece APIs para busca de locais baseados em posições no GPS, tendo categorização dos locais (restaurante, hospital, lanchonete, etc) e facilitando a busca por unidades médicas para inicialização do sistema. As APIs da Google utilizados no projeto foram: Google Places, Google Maps JS, Google Geocode.

Além disso, como será utilizado um mapa para melhor exibição de locais para os usuários, também neste ponto foi pensado no Google. O Google Maps é utilizado na maioria dos aplicativos que buscam exibir o mapa que o usuário possa interagir, como os aplicativos Waze e Uber, devido as funcionalidades que sua API fornece. Portanto, o Google também será usado para visualização do mapa e exibição das unidades.

5.2 Inserção inicial da base de dados

O método de inicialização acontece principalmente (isto é dito pois ele também é utilizado em outros casos) quando o usuário abre o sistema pela primeira vez. O sistema recupera a localização atual do usuário e inicializa o método, que verifica se na região deste usuário existe alguma unidade cadastrada no banco de dados. Caso não exista o sistema irá fazer uma requisição HTTP ao Google Maps buscando

as unidades mais próximas considerando algumas palavras chave: clínica, hospital e unidade; as unidades retornadas pelo Google Maps serão inseridas no sistema e assim irá se formar a base inicial.

Um outro lugar aonde este método é utilizado é na busca por unidade de saúde, que irá chama-lo caso o usuário busque unidade em alguma cidade que não foi inicializada. Com isso o usuário dificilmente irá fazer uma busca sem retorno do sistema.

Como este método é chamado sob demanda do usuário, não foi necessário fazer a automatização do mesmo para preencher o banco com unidades de todas as cidades. Isto irá remover a chance do usuário encontrar unidades desatualizadas, por terem sido cadastradas no sistema antes de serem necessárias.

Para a busca no Google Maps API das unidades de saúde é utilizada a requisição HTTP para https://maps.googleapis.com/maps/api/place/nearbysearch/json [21]. Nesta requisição é possível buscar as unidades mais próximas do usuário de acordo com a posição do GPS, um tipo definido pelo programador, além de uma palavra chave com os parâmetros location, type e keyword respectivamente. O resultado é dado na figura 5.1. Para verificar se ocorreu com sucesso a requisição basta verificar se o campo status retornou com um "OK". Explicando o resultado da requisição, temos o campo results com vários locais e suas informações sendo que algumas delas serão usadas pelo sistema como: nome, local, endereço, tipo e place_id. O place_id será usado como um marcador para não cadastrar a mesma unidade várias vezes.

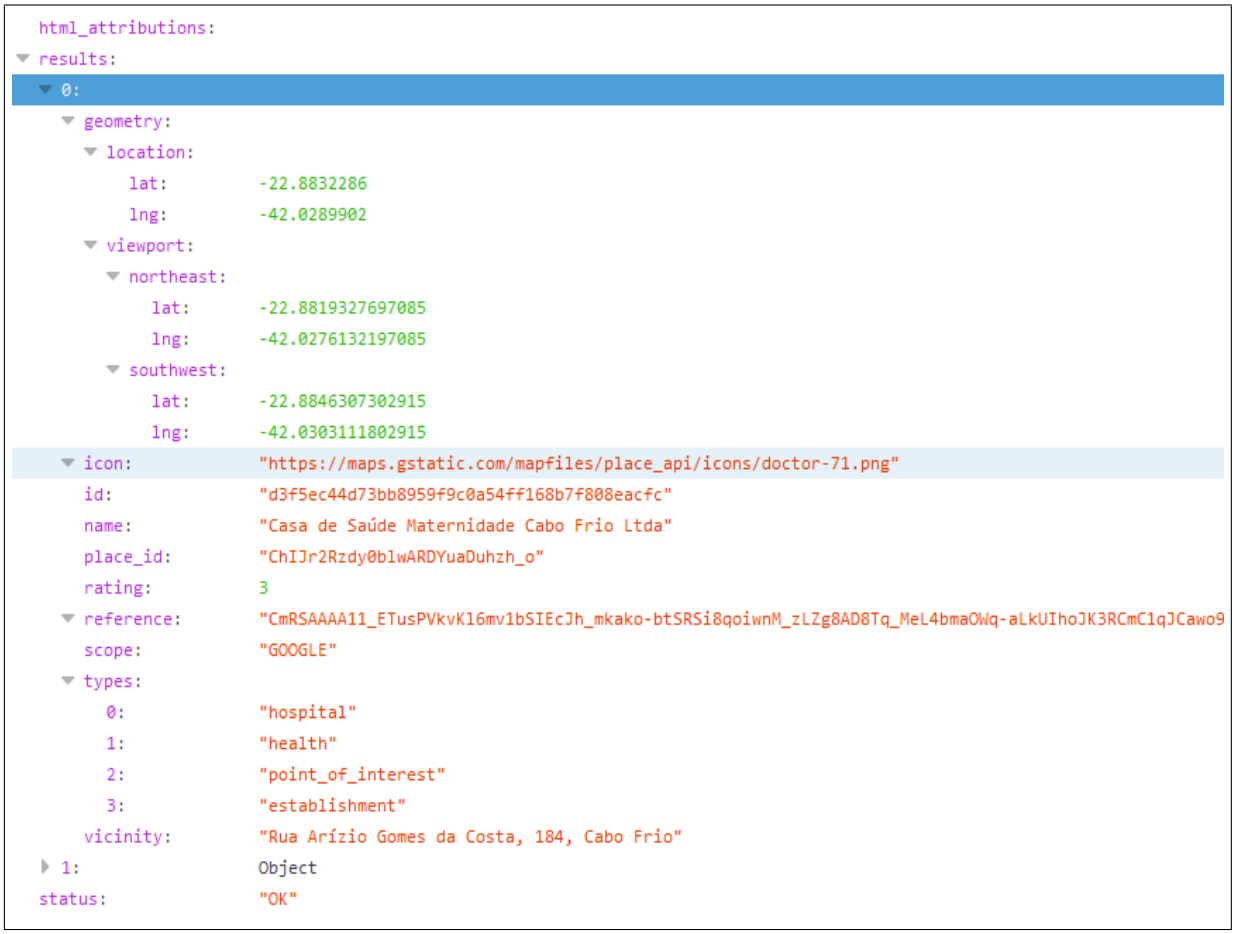


Figura 5.1: Resultado de uma requisição JSON feita para o Google Maps usando a API de nearby search

Para verificar o tempo de resposta do método de inicialização inteiro, foram feitas várias requisições através do console do Google Chrome e Mozilla Firefox. Com isso, foi possível verificar o tempo de resposta destas requisições que ficaram na média de 1,2 segundos para requisições feitas em locais que não possuem unidades de saúde cadastradas. Explicando melhor o teste, foram feitas 10 requisições com o banco de dados limpo para a cidade de Cabo Frio - RJ e a média das requisições foi de 1,2 segundos. O tempo referido nos testes, é um tempo bom para que o usuário não fique frustrado esperando o sistema dar retorno.

5.3 Método de busca emergencial

O método de busca emergencial serve para que o usuário consiga encontrar a unidade que atenda uma emergência mais próxima. Para isso foi entendido que é um método que não pode demorar. Porém, para explicar melhor é preciso mencionar que a exibição do caminho do usuário até uma unidade é feita pelo Google Maps através de uma requisição HTTP. Com estas informações já é possível demonstrar o problema, pois para saber qual unidade emergencial no banco do sistema é a mais próxima do usuário seriam necessárias várias requisições ao Google e isto tornaria o método muito lento. Com isso foi decidido que o sistema deveria calcular a distancia do usuário para as unidades por conta própria em linha reta (sem considerar ruas e caminhos possíveis para o usuário) para diminuir a quantidade de unidades próximas (em até 5) e então exibi-las para o usuário escolher por conta própria e exibindo o caminho pontualmente.

Contudo, para melhor calcular a distancia entre o usuário e a unidade de saúde foram selecionados dois métodos diferentes: a fórmula da distância entre dois pontos (distância euclidiana) e e fórmula de distância entre dois pontos em uma esfera (Haversine) [22]. Por ser mais correto, foi escolhida a formula de Haversine que calcula a distancia entre dois pontos a partir de suas latitudes e longitudes. Mesmo assim foi feito um cálculo para verificar a diferença entre os dois métodos para melhor demonstrar.

Na figura 5.2 está a representação no Google Maps dos dois pontos que serão utilizados no cálculo da distancia entre dois pontos. A distancia entre os dois pontos no mapa é de 1.2 km e consiste em uma reta que se localiza próximo ao município de Arraial do Cabo - RJ, as coordenadas serão aplicadas nas duas fórmulas para representação. Os pontos para o teste são: -22.9557103; -42.0374938 e -22.9495668; -42.0470306. Usando a fórmula da distancia euclidiana como dado na figura 5.3 é possível obter um valor de 1.113 km. Usando a fórmula de distancia de haversine como dado na figura 5.4 é possível obter um valor de 1.192 km. Levando em consideração os dois resultados, é possível verificar que a fórmula de haversine chega mais próximo ao resultado desejado.

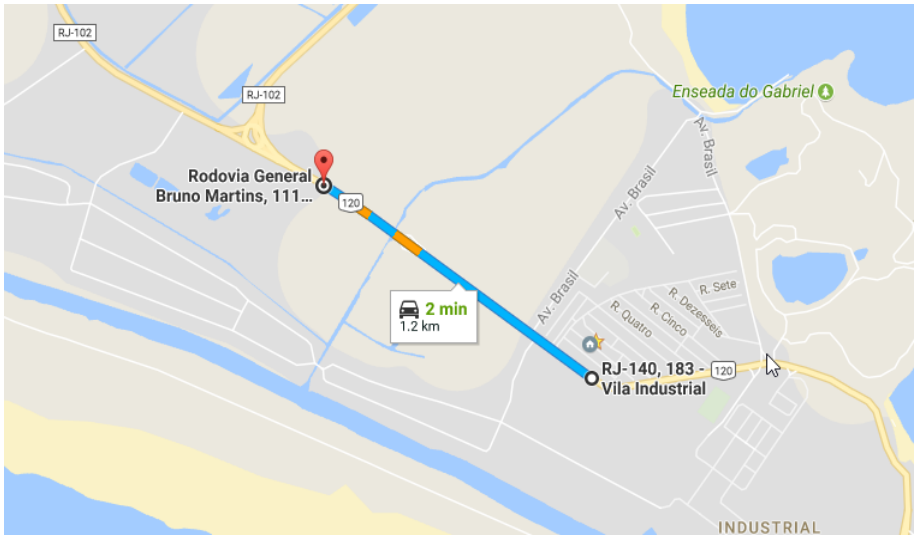


Figura 5.2: Local de teste para o cálculo de distancia entre dois pontos

$$\mathrm{d}(\mathbf{p}, \mathbf{q}) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2}.$$

Figura 5.3: Fórmula de cálculo da distancia euclidiana em 2 dimensões

$$= 2r \arcsin\left(\sqrt{\sin^2\left(\frac{\varphi_2 - \varphi_1}{2}\right) + \cos(\varphi_1)\cos(\varphi_2)\sin^2\left(\frac{\lambda_2 - \lambda_1}{2}\right)}\right)$$

Figura 5.4: Fórmula de cálculo da distancia em uma esfera de haversine

5.4 Servidor PHP

Para implementação do servidor do sistema foi usada a linguagem PHP pois já fornece uma facilidade de interação com o HTML onde foi desenvolvido a interface do sistema. O servidor foi planejado para que pudesse ter suas funções chamadas por requisições HTTP como se fosse uma API, por este motivo foi utilizado o framework Sheer que já automatiza isto, pois todas as funcionalidades criadas no projetos podem ser acessadas por requisições HTTP. Com o servidor em formato de API será possível futuramente desenvolver um aplicativo onde a interface seja feita em qualquer linguagem, pois a utilização das funcionalidades ocorrerão do mesmo jeito.

5.5 Criação do aplicativo como site

A criação do aplicativo como um site ocorreu pelos motivos que serão explanados agora. Primeiramente, como foi citado e explicado no capítulo da introdução (Capítulo 1), existe um maior interesse de se fazer um aplicativo para dispositivos móveis por ser de mais fácil acesso e uso para o usuário, também

lembrando da quantidade crescente de usuários para este tipo de dispositivo. Além disso, existe o fato do servidor ter sido feito em PHP e a facilidade de interação entre o PHP e o HTML. Com este pensamento, foi decidido que a interface para o usuário seria feita em um site. Porém, para o usuário este acesso ao site deve ser feito por um aplicativo de maneira transparente.

Com isso, e através do conhecimento da ferramenta Cordova [23], que é uma IDE para desenvolvimento de aplicativos que permite criar um aplicativo que simplesmente abre um site específico nele, surgiu a ideia de fazer um site que fosse chamado pelo aplicativo. A ideia do Cordova é ter um aplicativo que simule um navegador o qual o usuário vai sempre ver um site escolhido pelo desenvolvedor. Com esta decisão bastou criar o site com a interface feita para um dispositivo móvel utilizando a biblioteca de CSS Bootstrap e criar em JAVA um aplicativo Android que chamasse o módulo de WebView ou utilizar o próprio Cordova para acessar este site e assim o usuário passa a ver um site como se fosse um aplicativo. Para o sistema deste projeto foi utilizado o Cordova por ser mais simples de usar.

A vantagem da utilização de um site como aplicativo é que o usuário não precisa fazer a atualização do aplicativo para te-lo atualizado, pois assim como sites toda vez que são acessados têm seus scripts e páginas baixadas novamente. Já a desvantagem é que dificilmente será desenvolvido algum conteúdo para ser utilizado de maneira offline pelo usuário, pois a utilização deste tipo de conteúdo se torna complexa.

Juntamente com a linguagem HTML foi utilizado o Javascript para controlar o uso das páginas e chamar as funções / métodos do sistema através de requisições ao servidor PHP. Como informado na subseção do servidor PHP esta chamada de métodos seria possível em qualquer linguagem que a interface fosse desenvolvida, devido ao servidor funcionar como uma API e aceitar requisições HTTP.

5.6 Testes funcionais

Foram realizados testes durante o desenvolvimento do sistema. Estes serão descritos para melhor entendimento. As funcionalidades que terão os testes explanados são: Inicialização do banco, cadastro de uma unidade, atualização de especialidades, busca emergencial. O intuito destes testes foi documentar o funcionamento dos métodos de acordo com possíveis entradas e os resultados obtidos. Através dos testes também é possível entender como aguardar a resposta dos métodos.

5.6.1 Inicialização do Banco

#	Parâmetros	Resposta Esperada	Resposta Obtida
1	gpsLocation = -22.896502,-42.0398797	Minimamente cadastro de 4 unidades de saúde de Cabo Frio - RJ	Cadastro de 4 unidades de saúde de Cabo Frio - RJ
2	gpsLocation = -22.948524,-42.0514557	Minimamente cadastro de 1 unidade de saúde de Arrraial do Cabo - RJ	Cadastro de 1 unidade de saúde de Arraial do Cabo - RJ e 3 unidades de saúde de Cabo Frio - RJ
3	gpsLocation = -22.855381,-42.0564127	Minimamente cadastro de 1 unidade de saúde de São Pedro da Aldeia - RJ	Cadastro de 1 unidade de saúde de São Pedro da Aldeia - RJ e 1 unidade de saúde de Cabo Frio - RJ

Tabela 5.1: Tabela de testes do método de cadastro de unidades

5.6.2 Cadastro de uma Unidade

#	Parâmetros	Resposta Esperada	Resposta Obtida
1	gpsLocation = -22.8828222;-42.026725 nome = ultrasom	Unidade cadastrada com as informações enviadas, dando prioridade para informações da Google	Unidade cadastrada com as informações: nome = Ultrassonografia Cabo Frio gpsLocation = -22.8828222;-42.026725 endereço = Avenida Júlia Kubitscheck, 35 Centro - Sala 305 - Centro
2	gpsLocation = -22.5166942;-41.9435853 nome = null	Unidade cadastrada com as informações enviadas, dando prioridade para informações da Google	Unidade cadastrada com as informações: nome = Hospital Municipal Drł Naelma Monteiro gpsLocation = -22.8828222;-42.026725 endereço = Rua Neudon Lustosa, s/n - Parque Zabulao
3	gpsLocation = -22.5142941;-41.9488929 nome = Casa de Saúde 1	Unidade cadastrada com as informações enviadas, dando prioridade para informações da Google. Neste caso é um ponto que o Google não possui informação.	Unidade cadastrada com as informações: nome = Casa de Saúde 1 gpsLocation = -22.5142941;-41.9488929 endereço = null

Tabela 5.2: Tabela de testes do método de inicialização da base de dados

5.6.3 Atualização de Especialidades

#	Parâmetros	Resposta Esperada	Resposta Obtida
1	unidade = Unidade A especialidade = Pediatria, Ortopedia	Todas as especialidades escolhidas serão inseridas	Unidade com as especialidades: Pediatria, Ortopedia
2	unidade = Unidade A especialidade = null	Todas as especialidades escolhidas serão inseridas	Unidade sem especialidades

Tabela 5.3: Tabela de testes do método de inicialização da base de dados

5.6.4 Busca Emergencial

#	Parâmetros	Resposta Esperada	Resposta Obtida
1	gpsLocation = -22.948524,-42.0514557	A unidade mais próxima. Neste caso em Arraial do Cabo - RJ (Hospital Geral de Arraial do Cabo)	Unidade de Arraial do Cabo - RJ, gpsLocation = -22.9758547;-42.0281234
2	gpsLocation = -22.855381,-42.0564127	A unidade mais próxima. Neste caso em São Pedro da Aldeia - RJ (Hospital Maria Isabel dos Santos Silva)	Unidade de São Pedro da Aldeia - RJ, gpsLocation = -22.840685;-42.1034839

Tabela 5.4: Tabela de testes do método de inicialização da base de dados

5.6.5 Conclusão sobre testes

A partir dos testes realizados e documentados através dos casos de teste foi possível concluir que as funcionalidades foram desenvolvidas corretamente. Com isso será dado um breve resumo sobre o que foi notado sobre as funcionalidades.

Os testes realizados na inserção e atualização das unidades de saúdes foram feitas com sucesso. Como exemplo, o método de inicialização de banco que utiliza tanto a inserção quanto a atualização de unidade de saúde foi utilizado em várias cidades e todas ocorreram com sucesso. Alguns casos de inserção podem ser mais demorados, quando realizados pelo sistema e em grande escala. A busca manual consegue verificar corretamente as unidades nas cidades procuradas e especialidades desejadas.

O método de inicialização automática feita pelo servidor é executado quando existe uma busca em um local onde não existem registros de unidades de saúde, o método está funcionando corretamente. Com um porém, que é o caso explicado anteriormente que a inserção de muitas unidades pode se tornar lenta. Isto foi verificado que em cidades grandes como Rio de Janeiro - RJ o tempo de execução do método aumentou de 1,2 segundos para 2 segundos em média.

A busca automática emergencial conseguiu exibir os locais mais próximos corretamente e exibir o caminho até o local.

Capítulo 6

Sistema

Neste capítulo serão apresentadas informações sobre o sistema implementado. Será falado sobre as telas do sistema, seus fluxos serão descritos e os testes realizados serão explicados.

6.1 Sistema criado

Através de testes no sistema desenvolvido foi possível a verificação tanto das funcionalidades implementadas quanto da facilidade ou dificuldade de interação que o usuário pode ter com as telas. Com isso, foi possível notar o que é necessário para melhorar a interface para a utilização dos usuários.

A partir do sistema desenvolvido foram feitos testes como: envio de unidades para o servidor e controle deste envio apenas com internet ativa; funcionamento da gestão de dados relacionados à unidades (como especialidades relacionadas) e busca de unidade a partir de um ponto por proximidade ou por cidade cadastrada. A seguir serão descritas as telas principais do aplicativo móvel.

Todas as telas possuem duas opções iguais, que são os ícones de recarregar tela (seta dando volta) e o de voltar a tela inicial (casa). A primeira, na figura 6.1, o usuário vai entrar na tela inicial do aplicativo e terá algumas opções. Esta tela serve apenas para que o usuário decida o que vai fazer. As opções que o usuário tem são: Visualizar, Emergência, Busca Especialidade e Nova Unidade.

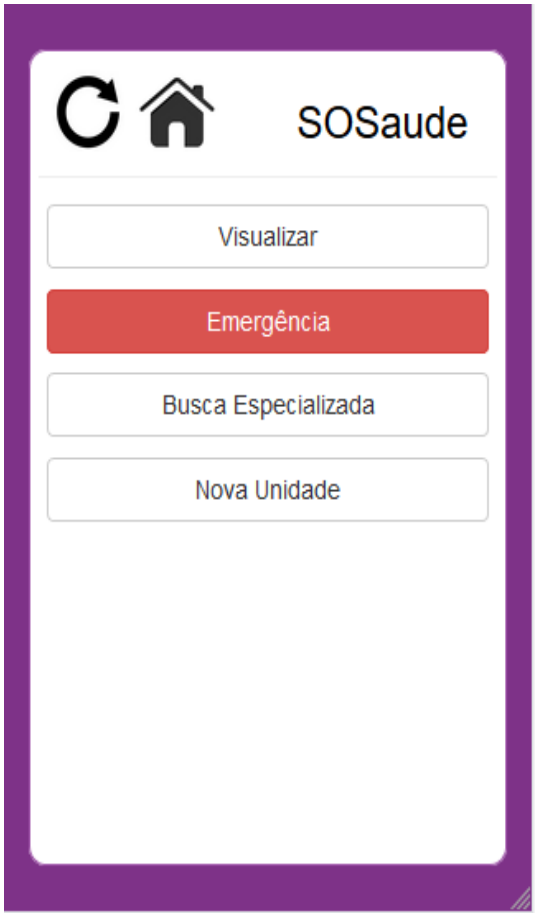


Figura 6.1: Tela Inicial

Caso o usuário escolha o terceiro botão que é Busca Especializada o usuário será levado à tela 6.2. Estas telas representam uma busca manual onde é possível para o usuário escolher a cidade e a especialidade que ele precisa. Após a escolha de ambos o sistema vai exibir no mapa as unidades que correspondem ao filtro selecionado. Também é possível para o usuário visualizar as informações de endereço da unidade ao clicar no ícone de alguma delas como exibido na figura 6.3.

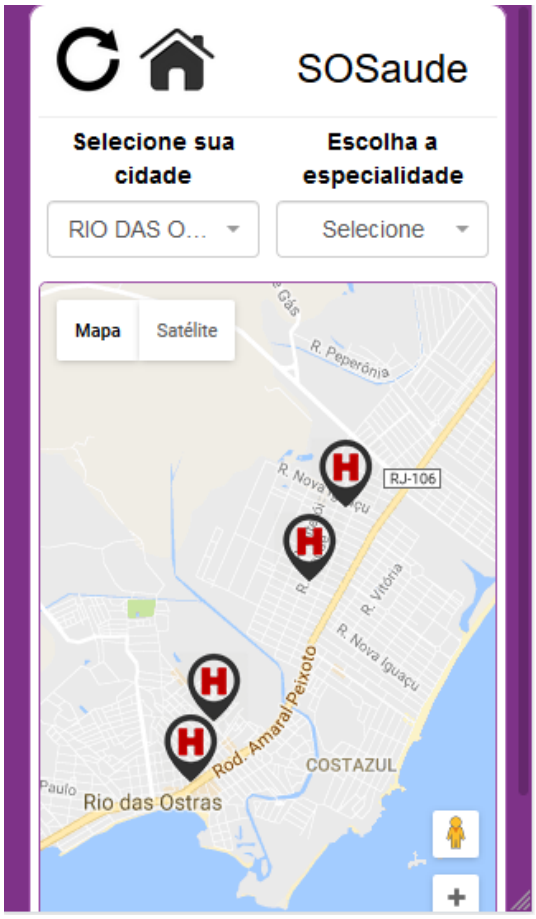


Figura 6.2: Tela de Busca - inicio

Figura 6.3: Tela de Busca - visualização de uma unidade

Caso o usuário, após clicar para visualizar as informações da unidade como na figura 6.3, ele pode escolher clicar na opção de atualizar unidade. Ao clicar nesta opção, o usuário vai para a tela de atualizar especialidades da unidade, como na figura 6.4. Nesta página existe a possibilidade de atualizar as especialidades da unidade.

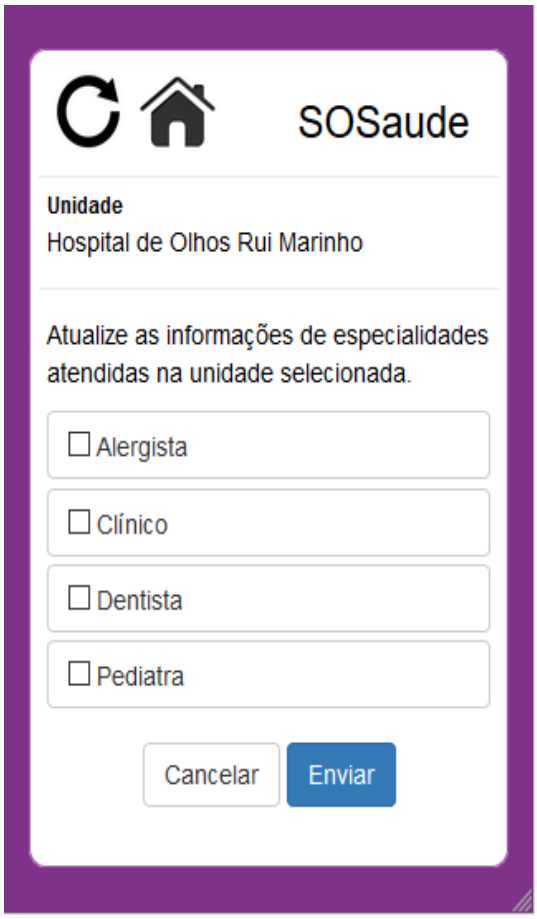


Figura 6.4: Tela para controle de unidade de saúde

Na tela inicial, caso o usuário escolha a opção de Emergência, ele será levado à figura 6.5 que é a tela busca automática (de emergência). Nesta tela, a única interação com o programa será apertar o botão e o aplicativo irá buscar a unidade mais próxima com atendimento à emergência e já direcionar este usuário para o local, demonstrando o caminho até o local, com visualização pelo mapa. Esta imagem foi tirada de um exemplo com apenas um hospital emergencial, caso exista mais de um usuário vai poder clicar e escolher o caminho para uma das opções.

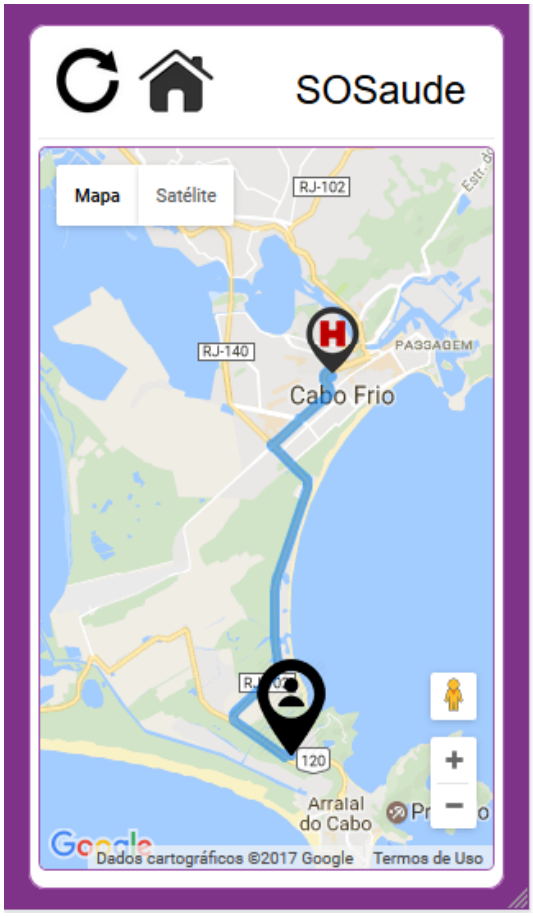


Figura 6.5: Tela de busca emergencial

Além das telas demonstradas ainda existe o fluxo de adicionar uma nova unidade. Neste caso, o usuário vai clicar em algum ponto do mapa aonde ele deseja adicionar uma unidade, e poderá escrever o nome desta. Estas serão as únicas informações obrigatórias ao usuário (localidade e nome).

Capítulo 7

Conclusão

Com o objetivo do projeto definido, como a construção de um aplicativo que a principio vai permitir um usuário de um sistema operacional móvel ter acesso à informações de unidades de saúde de todos os lugares possíveis, foi iniciada a analise e planejamento do projeto. Após o planejamento do projeto do sistema estar bem definido e documentado, foi implementado o aplicativo. Durante todo este processo foram feitas algumas decisões que serão apresentadas abaixo.

A escolha do modelo MVC para o desenvolvimento do projeto permitiu constatar a reutilização de código e facilitou visualmente a construção do projeto em relação aos códigos. A partir desta escolha foi possível criar as classes de visão e controle para subdividir o código em módulos onde cada um teria sua função dentro do projeto e assim facilitando ver aonde poderia acontecer uma reutilização de uma funcionalidade do sistema. O resultado desta utilização foi a redução nas linhas de código.

Ao iniciar a implementação do aplicativo foram notados pré-requisitos para o funcionamento do sistema (obrigatoriedade de internet e GPS). Estes pré-requisitos variam de acordo com as funcionalidades do sistema. Como exemplo, a busca emergencial exige tanto internet quando GPS para poder encontrar a unidade emergencial mais próxima do usuário. Por outro lado, a busca manual por cidade ou especialidade não têm necessidade de utilizar GPS, porém continua precisando de internet.

Com o desenvolvimento do aplicativo do projeto foram encontrados alguns problemas. Entre eles um que pode ser utilizado como um grande aprendizado, é o planejamento do projeto. Muito se foi pensado em produzir, porém, com o surgimento de problemas acabou havendo atrasos no desenvolvimento.

Com a criação de um banco único para todos os usuário, facilitou a atualização deste banco. Através de ajuda da inserção de informações vinda de usuários, o banco pode-se manter atualizado. O que seria impossível de se fazer por não existir uma limitação de distancia física para as unidades cadastradas no sistema.

O preenchimento automático do banco para cidades que não foram utilizadas ainda através de dados vindos do Google Maps, também permite com que o sistema possa ser usado sem problemas não importando o local do usuário. Através desta funcionalidade foi possível garantir que o sistema pode ser utilizado em qualquer momento e buscando unidades em qualquer local.

A construção do servidor PHP como uma API também irá facilitar qualquer tipo de trabalho

futuro neste sistema. Como qualquer método pode ser acessado da mesma maneira, através de requisições HTTP, não irá existir dificuldades de dar continuidade ao projeto em interfaces feitas em outras linguagens e consequentemente em outros tipos de interfaces.

Ao fim pode ser visto que o problema principal proposto no trabalho, a possibilidade de buscar unidades de saúde e permitir ao usuário saber onde ela se localiza visualmente, pode ser solucionado através deste aplicativo. Porém como algumas outras propostas não foram implementadas, é possível que exista uma continuidade para este projeto. Entre as propostas não implementadas estão:

- União de unidades de saúde que são iguais. Já existe uma inteligencia básica para verificar duplicidades, que é através do identificador do Google. Porém, nem todos os casos podem ser detectados pelo sistema. Por isso, seria necessário que um usuário do sistema pudesse reportar uma duplicidade de uma unidade.

- Remoção de unidades de saúde que não existem mais ou nunca existiram. O sistema não tem como verificar se uma unidade já existente no sistema deixou de existir ou nunca existiu. Por isso, seria necessário a possibilidade de que um usuário pudesse informar a invalidade de uma unidade.

- Realização de testes com usuários. O sistema desenvolvido não foi testado por usuários e este tipo de teste é importante para identificar possíveis problemas de usabilidade do sistema.

Referências Bibliográficas

[1] Google Cloud Storage (30/08/2016) *https://cloud.google.com/storage/pricing*

[2] Progressive apps (30/11/2017) *https://medium.com/@slightlylate/progressive-apps-escaping-tabs-without-losing-our-soul-3b93a8561955*

[3] Busca Saúde (SP) (12/08/2017) *http://buscasaude.prefeitura.sp.gov.br/*

[4] Bradesco Seguros (12/08/2017) *http://www.bradescoseguros.com.br/wps/portal/TransforDigital/Site/Atendimento*

[5] Help Saúde (12/08/2017) *http://www.helpsaude.com/*

[6] Amil (12/08/2017) *https://www.amil.com.br/portal/web/servicos/saude/rede-credenciada/amil/busca-avancada*

[7] Unimed (12/08/2017) *https://play.google.com/store/apps/details?id=br.com.UnimedRio.Associado*

[8] Busca AMS (12/08/2017) *https://play.google.com/store/apps/details?id=br.com.petrobras.rj.android.buscaams*

[9] Google (12/08/2017) *https://www.google.com/maps*

[10] Busca UBS. (28/08/2017) *http://www3.prefeitura.sp.gov.br/buscaubsweb/forms/frmSobre.aspx*

[11] Zenith Media (30/11/2017) *https://www.zenithmedia.com/smartphone-penetration-reach-66-2018/*

[12] comScore (30/08/2016) *https://www.comscore.com/Insights/Blog/Number-of-Mobile-Only-Internet-Users-Now-Exceeds-Desktop-Only-in-the-U.S*

[13] PHP (12/08/2017) *http://php.net/manual/en/intro-whatis.php.*

[14] MariaDB (12/08/2017) *https://mariadb.com/.*

[15] Javascript (12/08/2017) *https://www.javascript.com/.*

[16] Bootstrap (12/08/2017) *http://getbootstrap.com/.*

[17] Wikipedia - comparação de frameworks (03/09/2017) *https://en.wikipedia.org/wiki/Comparison_of_web_framewor*

[18] Hibernate - What is an ORM (03/09/2017) *http://hibernate.org/orm/what-is-an-orm/*

[19] Google Maps API - Documentação (13/09/2017) *https://developers.google.com/maps/documentation/javascript/t*

[20] Google Geocode API - Documentação (13/09/2017) *https://developers.google.com/maps/documentation/geocoding*

[21] Google Places API - Documentação (13/09/2017) *https://developers.google.com/places/web-service/search*

[22] van Brummelen, Glen Robert (2013) *Heavenly Mathematics: The Forgotten Art of Spherical Trigonometry*

[23] Apache Cordova (27/08/2017) *https://cordova.apache.org/*