ADOP - Alternativa Digital às Ouvidorias Públicas

Jheison Maciel Ines

PROJETO DE CONCLUSÃO DE CURSO

Data de Depósito:

Assinatura: _____

ADOP - Alternativa Digital às Ouvidorias Públicas

Jheison Maciel Ines

Cristina Maria Valadares de Lima

Monografia apresentada ao Instituto Superior de Educação do UNIFOR/MG, como requisito parcial para obtenção do título de bacharel em Ciência da Computação, sob a orientação da Prof Me:. Cristina Maria Valadares de Lima.

Unifor-MG - Formiga
Novembro/2018

Dedico este Trabalho a meus pais e minha irmã por todo apoio durante essa jornada.

Agradecimentos

Primeiramente quero agradecer a Deus e N. Sra. Aparecida por toda proteção concedida em todos esses anos nas viagens entre minha casa e a faculdade. A minha grande família, mas especial a minha irmã Jheilla e meus pais Adercio e Maria por estarem ao meu lado, ajudaram como puderam, ficamos sempre juntos pois o sonho de me formar nunca foi só meu. Eles foram minha força e inspiração para seguir lutando, meus melhores exemplos que levarei pelo resto da minha vida.

Gostaria de agradecer do fundo do meu coração aos meus Professores de todas escolas que passei, se cheguei até aqui, foi graças a cada um de vocês que hoje fazem parte dessa vitória. Aos meus professores do Unifor-MG, foram 4 anos de conhecimento que me acompanharão onde quer que eu vá. Nunca fui o aluno mais dedicado, mas me esforcei muito para não decepcioná-los.

A minha orientadora Prof Me. Cristina Maria Valadares de Lima por todos esses anos, se tornou uma grande amiga, apesar das turbulências sempre me lembrarei das vezes que me deu apoio, me ajudou, das piadas de nerds (rsrs), enfim, muito obrigado. Espero ter sido um bom aluno/amigo, e que nossa amizade continue, pois admiro muito você.

Várias pessoas tiveram participação direta nesse trabalho, em especial 4 pessoas. A minha amiga Prof Me. Cristina Maria Valadares pelas orientações dadas em todas etapas do projeto. A minha irmã Jheilla Maciel Ines e meu tio Prof Esp. Eucilio Graciano Maciel por terem me ajudado com a parte escrita, dando sugestões e revisando todo texto. Por último e não menos importante meu grande amigo Bel. Saulo Henrique Gomes Castro, sem você não teria conseguido desenvolver a aplicação da forma desejada.

Aos meus queridos amigos de faculdade que me ensinaram muito, serão sempre lembrados, que nossa amizade vá além deste projeto de vida que iniciamos à 4 anos. Que daqui alguns anos possamos reunir nossas famílias para aquele churrasco e uma boa cerveja gelada e relembrar os velhos tempos. Agradeço e peço desculpas aos meus amigos por todas vezes que estive ausente nas reuniões e festa que vocês me convidaram, saibam que quis muito ser mais presente, um abraço enorme pela força que me deram até aqui.

"*Nada como perceber que o mundo é mais louco que você mesmo*"

Thor: The Dark World

Sumário

1 Introdução **1**

1.1 Considerações Iniciais 1

1.2 Objetivos do Trabalho 2

1.3 Estrutura da Monografia 2

2 Trabalhos Relacionados **3**

2.1 Considerações iniciais 3

2.2 Informação Geográfica Voluntária no Pantanal: Um sistema Web colaborativo utilizando a API Google Maps 3

2.3 Desenvolvimento e Integração de Mapas Dinâmicos Georreferenciados para o Gerenciamento e Vigilância em Saúde 4

2.4 Mapeamento criminal por meio da plataforma Google Maps 4

2.5 Desenvolvimento WEB de Sistema de Movimentação Financeira Para Empresas de Pequeno Porte 5

2.6 Web design responsivo – Bootstrap 5

2.7 Considerações finais 5

3 Materiais e Métodos **7**

3.1 Considerações Iniciais 7

3.2 Ouvidorias Públicas 7

3.3 Aplicações Web 8

3.4 Programação Web 8

3.4.1 Web Services 8

3.4.2 HTML5 9

3.4.3 CSS3 9

3.4.4 PHP 9

3.4.5 JavaScript 10

3.4.6 Jquery 10

3.4.7 Ajax . . . 10
3.5 Design Responsivo . . . 10
3.5.1 Frameworks de Design Responsivo . . . 11
3.5.2 Framework Bootstrap . . . 11
3.6 WampServer . . . 12
3.7 Georreferenciamento . . . 12
3.7.1 Google Maps . . . 12
3.7.2 Api Google Maps JavaScript . . . 12
3.7.3 Api Geocoding . . . 13
3.8 Banco de Dados . . . 13
3.8.1 MySQL . . . 13
3.9 Metodologia Adotada . . . 14
3.9.1 Fases do Projeto . . . 14
3.9.2 Considerações Finais . . . 16

4 Desenvolvimento e Resultados **17**
4.1 Considerações Iniciais . . . 17
4.2 Especificação de Requisitos . . . 17
4.3 Preparação do Ambiente . . . 19
4.4 Api's Google Maps . . . 20
4.5 Implementações . . . 21
4.5.1 Grid Bootstrap . . . 21
4.5.2 Desenvolvimento da index . . . 21
4.5.3 Página Inicial . . . 24
4.5.4 Página Mapeamento . . . 28
4.5.5 Página Sobre . . . 33
4.5.6 Página Contato . . . 33
4.5.7 Resultados . . . 34

5 Conclusões **39**
5.1 Considerações Finais . . . 39
5.2 Contribuições . . . 40
5.3 Trabalhos Futuros . . . 40

A Gráficos dos Resultados **45**

Lista de Figuras

3.1 Mapa Mental 14
3.2 Marcador com informações 16

4.1 DER das tabelas do banco 20
4.2 Aspectos da grid Bootstrap 22
4.3 Exemplo da construção das páginas 22
4.4 Cabeçalho e menu sanduíche 23
4.5 Rodapé 23
4.6 Botão Flutuante 24
4.7 Modal das matérias 25
4.8 Cards com notícias 26
4.9 Formulário de queixas 27
4.10 Exemplo requisição ao ViaCEP 28
4.11 Mensagens de alerta 28
4.12 Código emissor do XML 29
4.13 Mapa com Marcadores 30
4.14 Filtro de cidade 31
4.15 Código para filtrar cidade 32
4.16 Funcionamento da função Ajax 32
4.17 Recorte da página sobre 33
4.18 Formulário de contato 34
4.19 Marcação feita local errado 35
4.20 Acesso a aplicação por diferentes dispositivos 36

A.1 Gráfico da pergunta 1 45
A.2 Gráfico da pergunta 2 46
A.3 Gráfico da pergunta 3 46
A.4 Gráfico da pergunta 4 46
A.5 Gráfico da pergunta 5 47

A.6 Gráfico da pergunta 6 47
A.7 Gráfico da pergunta 7 47

Lista de Tabelas

3.1 API's utilizadas 14

4.1 Requisitos Funcionais 18
4.2 Requisitos Não-Funcionais 19
4.3 Exemplo da disposição dos dados na tabela de municípios 20
4.4 Exemplo da disposição dos dados na tabela de queixas 20

Lista de Siglas

ADOP - Alternativa Digital as Ouvidorias Públicas
AJAX - *Asynchronous Javascript and XML*
API - Interface de Programação de Aplicativos
BD - Banco de Dados
CDN - Rede de Distribuição de Conteúdo
CRUD - *Create, Read, Update e Delete*
CSS - Folha de Estilo em Cascatas
DER - Diagrama Entidade Relacionamento
DOM - *Document Object Model*
HTML - Linguagem de Marcação de Hipertexto
HTTP - Protocolo de Transferência de Hipertexto
IBGE - Instituto Brasileiro de Geografia e Estatística
JHI - *Journal of Healt Informatics*
JS - *Javascript*
JSON - *JavaScript Object Notation*
JSP - *Java Server Page*
MVC - *Model View Controller*
PDF - *Portable Document Format*
PHP - *Hypertext Preprocessor*
PX - *Pixels*
RAM - *Random Access Memory*
RF - Requisito Funcional
RNF - Requisito Não-Funcional
SGBD - Sistema de Gerenciamento de Banco de Dados
SMTP - Protocolo Simples de Transferência de E-mail
SQL - Linguagem de Consulta Estruturada
SSD - *Solid-State Drive*

UF - Unidade da Federação
URL - Localizador Uniforme de Recursos
VGI - *Volunteered Geographic Information*
XML - *eXtensible Markup Language*

Resumo

Ines, J. M. *ADOP - Alternativa Digital as Ouvidorias Públicas.* Monografia (Graduação) — Centro Universitário de Formiga – Unifor-MG – Formiga, 2018.

Hoje em dia é comum ver pessoas se queixando umas para as outras de problemas sociais e estruturais dos locais em que vivem, mas poucas realmente buscam uma comunicação mais próxima com governantes. O projeto desenvolvido nesta monografia apresenta uma aplicação *web* como uma possível solução para aproximar quem se queixa, dos que realmente tem poder de aplicar mudanças, onde todas as queixas são registradas em um banco de dados *MySQL* que utiliza a linguagem *SQL* e posteriormente são mapeadas para facilitar que pontos com problemas sejam identificados. Todo mapeamento é feito usando duas *API's Google Maps JavaScript* e *Google Geocoding.* Com uso *framework bootstrap* todas as páginas da aplicação mantêm padrão responsivo, para que se adapte a qualquer dispositivo, independente do tamanho de tela do mesmo. Ele provém um conjunto de estruturas prontas, com classes *HTML* já estilizadas por padrão possibilitando ganho significativo de produtividade. ADOP vem como uma alternativa às ouvidorias públicas, sendo capaz de coletar, armazenar e mapear queixas em todo território nacional, podendo ser acessada de qualquer dispositivo que esteja conectado a Internet.

Palavras-chave: ADOP. Ouvidorias Públicas. Google Maps JavaScript. Bootstrap.

Abstract

Ines, J. M. *ADOP - Digital Alternative to Public Ombudsmen.* Monograph (Graduation) — Centro Universitário de Formiga – Unifor-MG – Formiga, 2018.

Nowadays it is common to see people complaining to one another about social and structural problems of the places where they live, but few of them really seek closer communication with our rulers. The project developed in this monograph presents a web application as a possible solution to approach who complains about those who really have the power to make changes, in which all complaints are registered in a MySQL database that uses SQL and are subsequently mapped to make it easier for problem points to be identified. All mapping is done using two Google Maps JavaScript and Google Geocoding APIs. With use of the bootstrap framework, all pages of the application maintain responsive standard, so that it adapts to any device, regardless of screen size. It provides with a set of ready-made structures, with HTML classes already stylized by default, allowing to achieve significant productivity gains. ADOP comes as an alternative to public ombudsmen, being able to collect, store and map complaints throughout the national territory, and can be accessed from any device that is connected to the Internet.

Keywords: ADOP. Public Ombudsmen. Google Maps Javascript. Bootstrap.

Capítulo

1

Introdução

1.1 Considerações Iniciais

As ouvidorias públicas são importantes para que a população em geral possa se comunicar por meio de canais diretos aos governantes. O brasileiro fala sobre políticas públicas cada vez mais, principalmente aquelas que afetam diretamente o lugar onde vivem. A ânsia por melhorias é algo normal que parte de cada um, pessoas de bem, que pensam em prol de todos. As ouvidorias públicas proporcionam benefícios como a inclusão o que contribui positivamente na sociedade.

Inclusão social envolve diversos campos da vida em sociedade: educação, saúde, segurança, distribuição de renda, entre outros. A prática de políticas públicas inclusivas tem por objetivo melhorar as condições de vida para toda população, a fim de contribuir com a igualdade de oportunidades. Sendo assim, a inclusão social tem papel de reforçar a construção e consolidação dos valores sociais na democracia brasileira, onde todos podem ser úteis no desenvolvimento do país (RICHE, 2010).

Sob essa visão, as ouvidorias públicas surgem como uma possibilidade de que cada cidadão possa fazer sua parte, contribuindo socialmente com sua visão dos problemas sociais, econômicos, estruturais e de segurança por exemplo, para que governantes possam analisar e posteriormente aplicar práticas que tragam melhorias a toda sociedade. As ouvidorias públicas oferecem condições de influenciar e alavancar melhorias na prestação dos serviços públicos e a geração de igualdade de oportunidades, permitindo que cada pessoa possa fazer sua parte, tenha voz e vez dentro das administrações públicas (CARDOSO, 2010).

Para contribuir com essa questão, ferramentas que facilitem essa comunicação entre população e governos são válidas. Essa relação hoje é feita majoritariamente por telefone ou e-mail. Dessa forma, a criação de uma aplicação *web* responsiva, acessível de qualquer dispositivo que esteja conectado à internet, onde é possível expor parte desses problemas de forma rápida e

simples, contribui de forma positiva para tomadas de decisões futuras por parte dos governantes.

Atualmente desenvolver sites e aplicações responsivos que se adaptam a diversos tamanhos de telas diferente se tornou algo normal, visto a quantidade de dispositivos disponíveis no mercado e sua disparidade heterogênea. Acessar uma página que não se adapte a esses dispositivos é desagradável e frustrante para seus usuários, a falta de uma adequação correta ao *layout* e de seus componentes, faz com que o conteúdo seja visualizado de forma incompleta, tenha navegação prejudicada e o desempenho comprometido (PROSTT, 2013).

A técnica para criação de site responsivo, se aplica devido a *layout* fluído, que consiste em transformar unidades de medida fixa em relativas, ou seja, para diferentes tipos de tela, o *layout* se adapta de forma relativa. É feito um cálculo que considera o tamanho da tela e usa uma medida percentual para se adaptar a qualquer dispositivo (MARTINS; ROMANI,).

1.2 Objetivos do Trabalho

Com esse embasamento, o presente trabalho de modo geral visa desenvolver uma aplicação *web* responsiva para melhorar a comunicação entre sociedade e órgãos públicos, usando tecnologias *open source* como *PHP, Bootstrap e API's do Google*. Para facilitar e ampliar as formas que os cidadãos têm de colaborar com o meio em que vive, no âmbito municipal, estadual e federal.

De forma mais específica o trabalho mostra a criação desta aplicação usando as *API's Maps JavaScript e Geocoding* do *Google*, para marcar pontos de georreferenciamento nos endereços declarados através de um formulário. Todas as queixas registradas são exibidas em um mapa na página de mapeamento, facilitando a visualização dos locais onde há necessidade de melhorias. Por meio deste formulário presente na aplicação, todas as informações passadas pelos usuários, são coletadas e armazenadas em um banco de dados, que servirá como tomada de decisões por parte das autoridades competentes. A manipulação do banco *MySQL* é feita por meio do *PHPMyadmim* e as linguagens *HTML, CSS, JavaScript* também são utilizadas em todo processo de desenvolvimento da aplicação.

1.3 Estrutura da Monografia

Esta monografia está organizada da seguinte forma, no capítulo1 foi descrita a introdução, com as considerações iniciais sobre o trabalho e os objetivos do mesmo. Já no capítulo 2 é apresentado o estudo da arte com alguns projetos relacionados, que serviram de inspiração para elaboração da aplicação *web* ADOP. No capítulo 3 são expostos materiais e métodos, contendo as definições dos principais conceitos, tecnologias e metodologias utilizados neste trabalho. O capítulo 4 mostra o desenvolvimento e os resultados do atual trabalho a partir das técnicas e métodos apresentados anteriormente. E, o capítulo 5 traz as conclusões deste trabalho e suas contribuições, além de potenciais trabalhos futuros.

Capítulo

2

Trabalhos Relacionados

2.1 Considerações iniciais

Neste tópico alguns trabalhos relacionados serão apresentados, tais como ferramentas e tecnologias usadas nos projetos, com intuito de demonstrar o que será usado no desenvolvimento da aplicação *web* de prestação de queixas.

2.2 Informação Geográfica Voluntária no Pantanal: Um sistema Web colaborativo utilizando a API Google Maps

O projeto desenvolvido teve o objetivo de desenvolver uma aplicação web, o VGI-Pantanal que coleta informações provenientes dos usuários e demonstra esses dados geográficos do pantanal brasileiro. O usuário possui nível de acesso que lhe dá possibilidade de gerenciar e atualizar a base. A aplicação segue critérios éticos, políticos e de qualidade. Qualquer cidadão ou empresa pode acessar a página do VGI-Pantanal e tem a possibilidade de coletar informações da base, que forem úteis para contribuir com outros projetos e pesquisas (SOUZA et al., 2012).

As ferramentas de desenvolvimento *web* são usadas e integradas a uma *API* do *Google Maps*, para demonstrar locais e informações provenientes da mesma. Dentre algumas tecnologias utilizadas para o desenvolvimento da aplicação estão o Servidor *Apache*, a API propriamente já citada, linguagens de programação como *HTML, PHP*, para fazer o *back-end, XML, AJAX, jQuery e Javascript*.

Esta é uma aplicação *web* com a possibilidade de que terceiros possam contribuir de forma direta e voluntária, trazendo grandes benefícios. Todos os dados coletados podem servir como auxiliadores nas tomadas de decisões futuras, que empresas ou instituições governamentais possam vir a necessitar, ou simplesmente para pesquisas de cidadãos comuns buscando conhecer mais sobre a região do pantanal (SOUZA et al., 2012).

2.3 Desenvolvimento e Integração de Mapas Dinâmicos Georreferenciados para o Gerenciamento e Vigilância em Saúde

O artigo publicado no JHI, descreve a criação da aplicação web Integra-GIS, voltada para área da saúde, com objetivo de integrar informações como monitoramento e controle de determinado evento da saúde a uma base geográfica, das regiões da baixada Santista e Ribeirão Preto, no estado de São Paulo. Conta com diversos filtros para as informações, afim de poder usá-las como auxílio nas tomadas de decisões pelos profissionais responsáveis (NETO et al., 2014).

Como o sistema trabalha com diversas camadas e níveis de escala diferentes, os autores optaram por trabalhar com servidor de mapas gratuitos, o *Geoserver*, usando bases de dados alocadas no SGBD *PostgreSQL, API's do Google Maps e OpenLayers*. Foi usado aplicativo em JSP em um servidor *Tomcat*, para traduzir endereços em pontos no mapa. O resultado da criação desse mapa é exibido ao usuário através de um aplicativo criado com a *API Openlayers*, e demonstrado no ambiente do navegador *web*.

A utilização de sistemas *web*, baseados em ferramentas gratuitas, tem contribuído de forma significativa e positiva com profissionais de diversas áreas de atuação. Com Integra-GIS a comunicação é feita com diversas aplicações, contendo informações sobre epidemiologia com o georreferenciamento dos casos de tuberculose, dengue, hanseníase, mortalidade infantil entre outros dados, integrando todos eles em sua base (NETO et al., 2014).

2.4 Mapeamento criminal por meio da plataforma Google Maps

A criação de um mapa criminal para departamento de polícia da região de Blumenau-SC, utilizando-se da *API do Google Maps*, veio para substituir os mapas com alfinetes. De acordo com BORNHOFEN e TENFEN (2009), o projeto foi elaborado com intuito de suprir essa demanda por um sistema atualizado, para mais agilidade no processo de demarcação de locais onde ocorressem crimes, fazendo uso da *API* para marcar pontos no mapa. A ferramenta não gerou custos para os cofres públicos, e o código fonte foi disponibilizado, para uso posterior por outras instituições (BORNHOFEN; TENFEN, 2009).

Por meio do uso de banco de dados relacional *MySQL*, foram criadas tabelas contendo informações sobre os crimes e ícones personalizados para cada tipo diferente. Foi criada também uma tabela com descrição completa do ocorrido. Com uso da linguagem *PHP*, os dados são trazidos da base no formato, incorporado ao mapa com função do *JavaScript* e exibido em uma página *HTML*, estilizada com *CSS*. A atualização da base é feita por meio de outro *software*, que emite arquivos *XML*, para que seja transformado em instruções *SQL*, fazendo com que o *MySQL* adicione essas informações à sua base.

A aplicação do mapa criminal da região de Blumenau-SC, foi criada sem gastos, um grande

passo para compartilhamento das informações para as polícias e também para os cidadãos, de forma que sejam facilmente entendidas. Para que entre em produção e seja usado para tomar decisões quanto ao posicionamento do policiamento efetivo diariamente, a aplicação ainda precisa de ajustes. Com disponibilização do código fonte, outros profissionais capacitados poderão fazer melhorias e correções, contribuindo para um sistema cada vez mais robusto (BORNHOFEN; TENFEN, 2009).

2.5 Desenvolvimento WEB de Sistema de Movimentação Financeira Para Empresas de Pequeno Porte

Este artigo retrata a realidade de pequenas empresas que não podem investir alto em *software* de gestão financeira do negócio. O trabalho apresenta a criação de uma aplicação *web* de baixo custo que atenda às necessidades de empresas de pequeno porte (BODOT; AGNER, 2014).

Foi proposta a criação de um sistema que funcione direto do navegador, podendo ser executado em qualquer dispositivo que tenha um navegador web instalado. Dentre as ferramentas utilizadas no desenvolvimento estão *HTML, CSS, PHP, SGBD* que usa linguagem SQL, no ambiente de desenvolvimento *NetBeans*, servidor *web XAMPP*. Todo projeto foi estruturado usando arquitetura *MVC* , com ciclo de vida incremental, que visa implantar novas versões a cada nova ferramenta desenvolvida (BODOT; AGNER, 2014).

O artigo visa criar um *software web* de baixo custo que atenda às necessidades de pequenas empresas e seus fluxos de caixa, gravando todas movimentações em um banco de dados, podendo ser usadas como base para tomadas de decisões por meio de monitoramento do fluxo.

2.6 Web design responsivo – Bootstrap

Este artigo retrata a importância de se usar páginas *web* responsivas, ou seja, *layouts* que se adaptem a diferentes resoluções de telas, como as de *smartphones* com telas de 320px e até mesmo em tela 4k que alcançam 8.294.400px (ALMEIDA, 2011). No processo de desenvolvimento é usado o *framework Bootstrap*, para auxiliar e facilitar na criação dos elementos da tela, acarretando em um ganho de produtividade (TOMAZINI; LOPES, 2015).

Para obter resultados satisfatórios em sites responsivos, é necessário aplicar tecnologias como o *framework Bootstrap, HTML* e *CSS*, com *layout* baseados em *grid*, grade com linhas e colunas com *pixels* maleáveis (TOMAZINI; LOPES, 2015). O artigo descreve basicamente o processo de desenvolvimento usando o *framework* e práticas necessárias para criação de telas que se adaptam a diferentes dispositivos e telas.

2.7 Considerações finais

Todos trabalhos citados foram desenvolvidos com base em tecnologias e linguagens voltadas para *web* em sua maioria, similares aos objetivos apresentados na sessão 1 para desenvolvimento da aplicação ADOP.

É possível observar a importância de aplicações responsivas, visto a vasta gama de dispo-

sitivos de diversas marcas e modelos, cada um com suas peculiaridades com relação as suas telas. Também foram apresentados *APIs* e *frameworks* para integração com as aplicações, algo benéfico durante as fases de desenvolvimento, pelo fato da reutilização de códigos, ferramentas prontas, na sua maioria *open source*.

Capítulo

3

Materiais e Métodos

3.1 Considerações Iniciais

Nesta seção é possível identificar e entender alguns conceitos sobre ouvidorias públicas, Aplicações *web* responsivas, *API Google Maps JavaScript*, o banco de dados, linguagens de programação utilizadas na interação com banco e criação de aplicações *web*.

3.2 Ouvidorias Públicas

As ouvidorias tem se tornado um importante meio de comunicação entre as instituições e seus públicos, além de atuar de forma estratégica nas gestões administrativas e facilitando os diálogos para construção de soluções cotidianas de conflitos internos e externos. Atualmente, colabora também como instrumento de combate à corrupção dentro das organizações. Na esfera pública, as ouvidorias ampliam os espaços democráticos por meio da inclusão social das pessoas, possibilitando uma maior colaboração junto às administrações públicas. Essa participação da população tem objetivo de explicitar prioridades, alternativas de ações e soluções possíveis, para problemas estruturais e cada vez mais presentes (CYRILLO; ALVES; OLIVEIRA, 2018).

Nesse canal de comunicação entre o poder público e o cidadão, as ouvidorias públicas possuem a missão de envolver e trazer os diversos públicos para dentro da instituição, de forma cooperativa. Por meio dessa participação extrair importantes informações que servirão de base para as próprias instituições públicas traçarem suas prioridades de atuação e os ajustes necessários (PEREIRA, 2018).

A constante geração de indicadores através das ouvidorias, permitem aos gestores mais atentos uma mudança de estratégia e o constante aperfeiçoamento do seu agir. Da mesma forma, quando o cidadão percebe que a sua colaboração é importante, e que sua manifestação é levada em conta dentro das instituições públicas, introduz uma relação de parceria e confiança

beneficiando a todos de forma geral (GUILHERME, 2018).

É preciso reconhecer o importante papel das ouvidorias e suas contribuições no processos de captação e produção de dados estratégicos para a gestão pública, seja pelo meios tradicionais ou criando alternativas como a apresentada neste trabalho, alternativas que estreitem a relação população e gestões públicas.

3.3 Aplicações Web

Aplicação *Web* é uma aplicação desenvolvida no formato de site e não é disponibilizado nas lojas virtuais de dispositivos móveis como *Play Store (Google) e App Store (Apple)*. Possui uma programação responsiva que reconhece e se adapta ao dispositivo utilizado pelo usuário. Toda sua programação é feita utilizando linguagens como *HTML5, CSS e Javascript*. Podendo ser acessado de qualquer sistema operacional, por meio de um navegador web, desde que possua conexão com a internet (MENDES, 2017).

Em uma aplicação web é possível, por exemplo, cadastrar informações em um banco de dados, fazer requisições a *API's*, automatizar processos, entre outros. O grande foco de uma aplicação é solucionar um problema utilizando a programação para isso.

3.4 Programação Web

A abundância de dispositivos eletrônicos existentes com acesso à Internet faz com que o desenvolvimento web aumente a cada dia, se tornando mais um atrativo para desenvolvedores (BODOT; AGNER, 2014). No desenvolvimento de aplicações web, as tecnologias mais utilizadas são *HTML, CSS, JavaScript* e algumas outras linguagens que fazem a ligação entre cliente (páginas *web*) e servidor como o PHP (LOUDON, 2018).

3.4.1 Web Services

Os *Web services* foram criados com o intuito de facilitar a comunicação entre dispositivos de diferentes plataformas, através de protocolos, independente da linguagem de programação utilizada nessas plataformas. Os Web services funcionam com qualquer sistema operacional, *hardwares* ou linguagem de programação de suporte *web*. (KRATZ et al., 2007)

Web Services são componentes acessíveis através da rede, que trabalham trocando mensagens no formato *XML* ou *JSON*. A disponibilização das operações e a descrição do serviço também ocorrem através do padrão *XML*. O arquivo do serviço a ser usado possui todas as informações necessárias para que outras aplicações possam interagir com o mesmo, o que inclui o formato das mensagens para as chamadas aos métodos do serviço, protocolos de comunicação e as formas de localização do serviço. Utilizar um *Web Service* é vantajoso e permite que recurso seja acessado independentemente da plataforma de *hardwares* ou *softwares* na qual foi implementado (HENKELS, 2011).

Um dos *webservices* a ser usado é o ViaCEP[1], voltado para consultar Códigos de Endereçamento Postal (CEP) do Brasil. Trata-se de serviço gratuito que neste trabalho é usado

[1] https://viacep.com.br/

para consultar endereços pelo CEP fornecido pelo usuário, em posse da resposta ao serviço a aplicação preenche campos rua, bairro, cidade, uf, do seu formulário de criação de queixas automaticamente.

3.4.2 HTML5

O *HTML* é uma linguagem amplamente utilizada na construção de páginas *web*, que são interpretadas pelos navegadores (*browser*). Trata-se de linguagem de marcação, que utiliza *tags*, delimitadas por <> e </>, para marcar e definir a estrutura de páginas *web*. A forma com que todo conteúdo é exibido, são feitas por códigos em *CSS* (BORTOLOSSI, 2012).

Com uso do *HTML5* é possível criar documentos com estruturas como títulos, links, tabelas, formulários, elementos de vídeos, imagens, animações, entre outros, diretamente ao usuário. Também pode ser usado em conjunto com outras linguagens incorporadas entre suas tags, como por exemplo, *API's, JavaScript* e o *PHP*, que fazem a interatividade e ligação com banco de dados, entre usuário e servidor (FLATSCHART, 2011).

3.4.3 CSS3

A linguagem *CSS* é utilizada para estilizar páginas *HTML*, dar uma aparência aos elementos, definir cores, efeitos, posicionamento, etc. *CSS* auxilia o desenvolvedor a poupar tempo na implementação de uma página, também facilita posteriormente na edição e até mesmo fazendo carregamento mais rápido (MARTINS; SILVA; FLÔRES, 2017).

O *CSS3* ilustra como os elementos em *HTML* devem ser apresentados nas telas. Resumidamente, é o *CSS* que determina o visual das páginas, desde o tamanho da fonte até a imagem de fundo, tudo pode ser alterado com o *CSS* (OLIVEIRA; ELER, 2015).

Devido ao *CSS3*, todos os elementos de páginas web ganham cores, posicionamento, tamanho e com uso da linguagem fica mais fácil e ágil fazer o *re-design* dos sites, o código de formatação pode ser isolado em um único arquivo, podendo aplicar o mesmo estilo para vários elementos que tenham classes idênticas de uma só vez.

3.4.4 PHP

O PHP foi criado inicialmente como uma linguagem de *script* estruturada, ao longo dos anos, vários recursos foram sendo incorporados tornando-a mais robusta, possibilitando uso de classes e paradigma de programação orientada a objetos (MINETTO, 2007).

Trata-se de uma linguagem de programação *openSource*, com documentação vasta, exemplos práticos e disponíveis em seu site oficial[2]. Por dar dinamicidade aos sites é amplamente usada hoje em dia na construção de páginas *web*, (NIEDERAUER, 2004).

Com *PHP* é possível criar sites dinâmicos em conjunto ao *HTML*. Porém todos *scripts* escritos usando a linguagem, são executados no servidor. Após isso, os resultados são incorporados ao *HTML* e exibidos na tela, não consumindo recursos de *hardware* do cliente (CHAVES; SILVA, 2008).

[2]http://www.php.net/

Um dos pontos favoráveis que leva este projeto usar a linguagem *PHP*, é que tem uma sintaxe simples e amigável, o que torna o processo de aprendizado mais produtivo e célere.

3.4.5 JavaScript

É executada diretamente nos navegadores *web* (*Chrome, Mozilla, Edge*, entre outros), sendo possível manipular diversos elementos escritos em *HTML*, criando interações dinâmicas e fluídas dentro de páginas web (BASSO et al., 2015).

JavaScript permite a criação de pequenos programas que são colocados junto ao código *HTML*, tornando possível diversas aplicações que processam dados, criar e alterar elementos e animações em *HTML*, validação de formulários, comunicar-se com servidor e algumas outras possibilidades. Trabalhando de forma independente, todos *scripts* são processados pelo dispositivo do próprio usuário (GRILLO; FORTES,).

Ao utilizar *JavaScript* à algumas vantagens em encontrar *plug-ins* para diversas finalidades, como por exemplo a criação de gráficos e principalmente a interação com a *API do Google Maps*, que são incorporados às aplicações *web* e as tornam ainda mais dinâmicas e ricas em funcionalidades.

3.4.6 Jquery

Biblioteca *JavaScript* criada em 2005, com intuito de simplificar a forma como códigos em JS são escritos, tornando assim desenvolvimento mais prático e ágil. É possível manipular eventos, elementos em *HTML, CSS*, animações, fazer interações com *AJAX* e maior facilidade ao manipular com poucas linhas o *DOM* (SANTANA; ROMANI; OTAVIAN,).

3.4.7 Ajax

Para que páginas web possam ser mais dinâmicas, atualizando elementos e informações sem que as mesmas sofram recarregamento, é usado *Ajax*, uma mistura de *JavaScript* e *XML*. Ele pode transferir *XML, JSON* e outros dados textuais de um servidor Web para uma página Web, usando o protocolo *HTTP* por meio do estabelecimento de um canal de comunicação assíncrona entre páginas Web do lado do cliente e do lado do servidor (ANDRIOLO, 2018).

Neste projeto o *Ajax* é usado para fazer requisições do tipo *POST* ou *GET* (Métodos de comunicação do protocolo *HTTP*)[3] a páginas em *PHP*, que por sua vez faz o que for necessário e retorna os dados ao *Ajax*, que se encarrega de atualizar os elementos *HTML* ou simplesmente exibir mensagens informativas para o usuário.

3.5 Design Responsivo

Design responsivo tem por definição, criar páginas na internet, que adaptem o seu *layout* a qualquer dispositivo, com telas de diferentes resoluções, ou seja, se adaptando a telas de *smartphones* até TVs de última geração, que chegam a ter resoluções acima de 4k (SILVA, 2014).

[3]Uma requisição GET é enviada como string anexada a URL, enquanto que a requisição POST é encapsulada junto ao corpo da requisição HTTP e não pode ser vista, o que a torna um meio mais seguro. Fonte: https://www.devmedia.com.br/entendendo-os-metodos-get-e-post-no-php/10470

A nomenclatura 4k se refere à quantidade de *pixels* horizontais e verticais (4096x2160) na tela. *Design* responsivo tem o objetivo de garantir uma melhor experiência do usuário, permitindo navegação e leitura agradável sem afetar o conteúdo (ALMEIDA, 2011).

O *layout* define como será a estética das páginas *web*, que podem ser estáticos ou fluídos, baseados em largura e comprimento. Na opinião dos autores, o uso dos *layouts* fluídos é o mais recomendado, por se adaptar a diferentes resoluções de telas e aproveitar de forma mais eficiente todos espaços, distribuindo elementos em uma página na internet de forma agradável (SCHULENBURG et al., 2013).

Este formato de *layout* se caracteriza por não especificar tamanhos e quaisquer outras medidas de forma fixa. São usadas porcentagens, podendo assim se adequar de acordo com a proporção da tela utilizada (ROSA; SILVA, 2018).

Para que elementos como imagens e vídeos sejam flexíveis e possam se adaptar ao *layout*, é utilizada uma linha de código em *CSS*, declarando que determinado elemento deve ter largura dada por porcentagem. Com isso, ele é redimensionado de forma automática, seguindo proporcionalmente a resolução do dispositivo em que está sendo exibido (FRANÇA, 2015).

3.5.1 Frameworks de Design Responsivo

Desenvolver um *web* site responsivo e intuitivo requer dedicação e tempo. Pensando nisso, diversos profissionais recorrem ao uso de *frameworks* específicos para *design* responsivo. Com utilização de tal ferramenta, desenvolvedores conseguem ser mais ágeis, devido à praticidade na criação das páginas *web* (SILVA; VALENTE, 2015).

Diversas estruturas responsivas vêm prontas e algumas, são escritas de maneira mais simples usando *Frameworks*. Caso o desenvolvedor já tenha conhecimento em *HTML* e *CSS*, a curva de aprendizado se torna maior (BORTOLI; RUFINO,).

3.5.2 Framework Bootstrap

Este *framework* dedicado à criação de interfaces tem seu código aberto e foi desenvolvido em 2010 por Mark Otto e Jacob Thornton. Sua concepção se deu para uso interno da rede social, o Twitter (SANTOS; RUFINO,).

O *Bootstrap* vai além de um *framework* de *design* responsivo. Dentre o conteúdo do *Bootstrap* está uma pasta com diversos arquivos *CSS* e outra contendo *JavaScript* que podem ser incorporados ao projeto, criando elementos como caixas de login, barra de menus, entre outros (MIGUEL; COSTA, 2015).

Com a quantidade de dispositivos que contam com características diferentes um dos outros, o *Bootstrap* veio para oferecer adaptação de telas entre estes dispositivos, contando com quantidade significativa de componentes, que facilitam a vida de desenvolvedores em todo mundo, usando diferentes linguagens de programação voltadas para *web* responsiva (TECHIO; CHICON, 2016).

A versão mais recente do *Bootstrap* funciona em diversos navegadores, tanto na plataforma móvel e *desktops*. Alguns elementos das páginas podem ser exibidos de forma inesperada em navegadores antigos ou desatualizados. Diariamente o *framework* recebe contribuições em seu

código fonte (SILVA; VALENTE, 2015).

O *framework Bootstrap* será utilizado para implementação das páginas em *HTML* da aplicação. Diversos elementos de páginas são fornecidos pela ferramenta já respeitando o design responsivo. A motivação de uso deste *framework* está diretamente ligada ao fato dele ser gratuito e intuitivo de usar e conta com materiais de auxilio, que são encontrados facilmente na internet.

3.6 WampServer

Trata-se de um programa que reúne várias outras ferramentas como *MySql, PHP e Apache*. Através do *WampServer* é possível usar um ambiente de desenvolvimento *web*, ele faz papel de servidor local responsável pela comunicação da aplicação (GIRALDI; PESSOA et al.,).

Por meio da instalação do programa *WampServer*, automaticamente instalam-se ferramentas citadas anteriormente, necessárias na fase de desenvolvimento e deixa disponível outros itens como o *PhpMyAdmin* para administrar banco de dados e outras opções de configuração e gerenciamento dos projetos.

3.7 Georreferenciamento

O Georreferenciamento consiste em poder fazer marcações em mapas, por meio de coordenadas reais, definidas por latitude e longitude. Com a *API Google Maps JavaScript*, é possível obter coordenadas de um determinado local, pelo endereço existente e até mesmo georreferenciar demais pontos usando latitude e longitude (ARAUJO et al., 2009).

A partir dessa possibilidade a aplicação busca no banco de dados todas queixas já registradas, através das latitudes e longitudes de cada uma delas e é colocado um marcador personalizado de acordo com tipo da queixa nas coordenadas fornecidas.

3.7.1 Google Maps

Trata-se de um serviço disponibilizado pela empresa *Google*, que exibe imagens de satélite da Terra de forma gratuita. É possível visualizar vários tipos de mapas através da ferramenta, observando nomes de ruas, rodovias, pontos específicos como empresas, locais turísticos. Também dá a possibilidade de criar rotas detalhadas, desde quilometragem até mesmo tipos de vias que a rota possui (BASTOS; JAQUES, 2010).

3.7.2 Api Google Maps JavaScript

Esta *API* gratuita do Google, oferece ao desenvolvedor ferramentas que possibilitam a implementação de mapas em *web* sites e aplicações, com recursos para manipulação do mapa, como, marcação de pontos, obtenção de rotas, informações sobre locais, entre outras (MIYAHARA; MENA-CHALCO; CESAR-JR, 2011).

A *API Google Maps* foi construída para ser utilizada usando a linguagem *JavaScript*. Existem diversas estruturas prontas que fornecem uma interface, o que faz com que desenvolvedor possa construir mapas dentro de suas aplicações (OLIVEIRA; NETO; SANTOS, 2010).

Por se tratar de uma *API* que se mantém gratuita mesmo com número considerável de acessos, cerca de 25.000 requisições por dia, ela será utilizada para marcar todos pontos onde já existam queixas registradas pelos usuários.

3.7.3 Api Geocoding

A API Google Geocoding é uma forma de utilizar um geo-codificador através de um protocolo *HTTP* ou *HTTPs*. A geocodificação consiste em converter endereços em coordenadas por meio de requisições pela url[4] que responde com um arquivo JSON. Também é possível realizar a operação inversa, transformando coordenadas em endereços (ALVES; ROMANI; OTAVIAN,).

Existe a possibilidade de utilizar o protocolo *HTTP* ou *HTTPS*. O segundo é recomendado quando os parâmetros informados são confidenciais uma vez que o protocolo *HTTPS* provê a proteção das informações passadas por parâmetro. Independente do protocolo escolhido, o formato dos parâmetros de envio ou o conteúdo da resposta será o mesmo (GOLDSTEIN, 2014).

No projeto proposto a *API* será usada para converter endereço fornecido pelo usuário por meio de um formulário em coordenadas geográficas, que são gravadas no banco de dados e posteriormente usadas para marcar pontos no mapa.

3.8 Banco de Dados

Comumente é definido como uma técnica onde dados são armazenados de forma organizada e padronizada, assim como em uma agenda telefônica, onde junto a cada nome, também é armazenado número de celular, telefone fixo, *e-mail*, endereço, entre outros. Um banco de dados pode ser armazenado localmente (diretamente no dispositivo do usuário) ou externamente em um servidor por exemplo (FERREIRA; TAKAI; FINGER, 2009).

3.8.1 MySQL

O *MySQL* é um servidor e gerenciador de banco de dados relacional, completo e muito popular, por se tratar de uma ferramenta gratuita, rápida e poderosa no que diz respeito a manipulação de dados e informações, amplamente usado no meio acadêmico, mas também em aplicações comerciais (MILANI, 2007).

Com seu uso é possível realizar instruções, para gerir o banco de dados, realizando tarefas de *CRUD*. Em outras palavras, os dados podem ser criados, lidos, atualizados e excluídos da base de dados (CAGNIN; PENTEADO, 1999).

Por estar mais familiarizado com banco de dados relacionais, foi mais viável usar o *MySQL* na produção deste trabalho, outro fator importante na escolha foi por ele ser gratuito. Na aplicação ele tem papel de manter duas tabelas, a de "queixas", que sofrerá inserções e consultas e a tabela "municipios", usada para auto completar campo filtro de cidade para centralizar o mapa no local desejado.

[4]Exemplo: https://maps.googleapis.com/maps/api/geocode/json?address=ENDEREÇOkey=API_KEY

3.9 Metodologia Adotada

Esta sessão tem por objetivo demonstrar como o projeto foi elaborado, desde sua concepção até finalização do desenvolvimento, a partir da fundamentação teórica retratada anteriormente.

3.9.1 Fases do Projeto

Nas fases iniciais do projeto foi feita uma especificação de requisitos funcionais e não-funcionais para a aplicação e criado um esboço por meio de um mapa mental visto na Figura 3.1, do que seria necessário durante o desenvolvimento. Foi iniciada a preparação do ambiente de trabalho, com a instalação de ferramentas como *PHP, MySQL* e servidor local *Apache*, e feita com *WampServer*, que possibilita o criação da aplicação *web*.

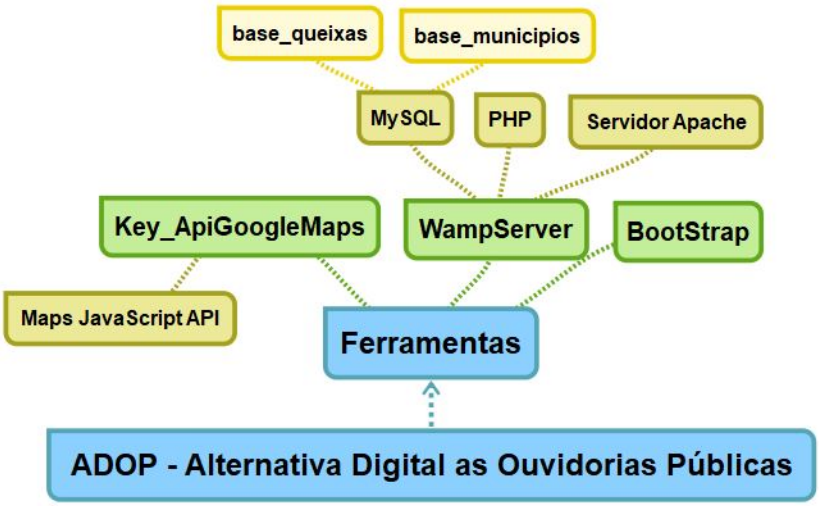

Figura 3.1: Mapa Mental
Fonte: Próprio autor

A segunda etapa envolveu a busca por uma base de dados que contenha os nomes de todas cidades brasileiras e dados de georreferenciamento (latitude e longitude). A mesma foi encontrada em um repositório do *GitHub*, disponível no link[5], já no formato *.sql*, pronta para ser importada para banco de dados *MySQL* "municipios_br", com a ferramenta *phpMyAdmin* incluída durante instalação do *WampServer*. Na Tabela 3.1 as duas API's Google Maps usadas no projeto e suas descrições retiradas do site oficial da API.

Tabela 3.1: API's utilizadas
Fonte: Próprio autor

API	Descrição
Google Maps JavaScript API	Incorpore um mapa do Google em sua página da web usando JavaScript. Manipule o mapa e adicione conteúdo com a ajuda de vários serviços.
Geocoding API	A Geocoding API é um serviço que fornece geocodificação e geocodificação reversa de endereços.

[5] https://github.com/kelvins/Municipios-Brasileiros;

Foi criada uma conta do Google e por meio do *site*[6], foram selecionadas as API's usadas, para disponibilização de uma chave/credencial única de acesso aos recursos da *API*. Essa credencial é usada toda vez que uma solicitação é feita para a ferramenta.

Vários componentes do *Bootstrap* exigem o uso de JavaScript para funcionar. Especificamente, eles exigem jQuery , Popper.js e os próprios plug-ins de JavaScript. Essa inclusão do *Bootstrap* e suas dependências necessárias ao projeto é feita via *BootstrapCDN* com *<scripts>* que fazem requisição aos arquivos necessários aos servidores em que ficam alocados.

Usando *phpMyAdmin* foi criada tabela a de queixas e feita a exportação da tabela de municípios baixada anteriormente para a base de dados da aplicação, possibilitando assim que todos dados possam ser escritos, acessados e entregues dentro da aplicação, já no formato *XML* por exemplo, e em um segundo momento usado pelo JavaScript para exibir de forma gráfica marcadores em um mapa e informações provenientes do local que podem ser lidas ao clicar no marcador.

As etapas seguintes exemplificarão o desenvolvimento da aplicação, o que inclui criar as 4 páginas da aplicação, são elas a Inicial, Mapeamento, Sobre e Contato. A tela Inicial conta uma imagem ilustrativa de um mapa, botão para criação das queixas, *cards* com algumas informações sobre ouvidorias públicas e como a população pode influenciar em uma sociedade e formas como isso pode ser feito. Menu superior e rodapé são exibidos dinamicamente em todas telas.

Quanto à tela responsável em coletar reclamações, é apresentado um formulário com campos obrigatórios, tais como cidade em que reside, nome completo, endereço em que deseja contribuir com alguma queixa, tipo de problema a ser descrito e um campo para descrição. Ao final todas essas informações são armazenadas em uma tabela no banco de dados, para consultas posteriores. Com auxílio da *API Google Maps JavaScript*, um novo marcador é exibido em um mapa, no devido local informado na queixa, já com a descrição do problema ali existente, que pode ser observado ao clicar sobre o mesmo. Exemplo de georreferenciamento com informações no marcador na Figura 3.2.

[6]https://console.developers.google.com/apis

Figura 3.2: Marcador com informações
Fonte: Próprio autor

Todas marcações das reclamações ficarão acessíveis na página de Mapeamento, onde é possível centralizar mapa na cidade desejada por meio de um filtro de cidade brasileiras. Para expor informações sobre o projeto proposto, da ferramenta desenvolvida fica a cargo da página Sobre e também botão de acesso a esta monografia . Para usuários que desejarem acessar a base de dados na íntegra, sem fins lucrativos, visando pesquisas e tomadas de decisões por governantes, poderá fazer pedido pela página de contato da aplicação web.

A última etapa foi colocar a aplicação *web online*, para que possa estar acessível para o público geral por meio da *url* http://adop.tech/. Ao final de cada inserção de uma queixa, o usuário será convidado a participar de uma pesquisa de satisfação, através de um questionário, para verificar qualitativamente a satisfação de uso da ferramenta, disponível no botão *feedback* presente em todas páginas da aplicação.

3.9.2 Considerações Finais

Esta sessão abordou a fundamentação teórica com diversas tecnologias e fundamentos e também foram relatadas de forma superficial algumas etapas necessárias para o desenvolvimento da aplicação *web* ADOP. No capítulo seguinte cada uma dessas etapas serão mostradas de forma clara e didática.

Capítulo

4

Desenvolvimento e Resultados

4.1 Considerações Iniciais

Esta sessão apresentará detalhadamente como as tecnologias foram usadas e como se deu o desenvolvimento da aplicação ADOP, subdividindo esta sessão em algumas outras que mostram documentação de auxílio, preparação e instalação das ferramentas, desenvolvimento *front-end* das páginas em que usuário poderá navegar, e *back-end* de cada uma delas. E ao fim resultados que irão exemplificar a aplicação e seu funcionamento.

4.2 Especificação de Requisitos

Para um bom desenvolvimento é importante identificar os requisitos, pois a partir desta fase podem surgir muitos erros pela falta de planejamento, que se não corrigidos a tempo impactarão em tempo e custo de desenvolvimento. Na especificação pode conter requisitos funcionais e não-funcional e até mesmo um diagrama de caso de uso ou prototipação de parte do produto. São descritos o passo a passo de cada funcionalidade bem como suas devidas restrições.

De um modo geral, o conjunto de requisitos de um sistema é definido durante as fases iniciais do processo de desenvolvimento. Tal conjunto de requisitos é visto como uma especificação do que deve ser implementado. Referem-se a funcionalidades do sistema, são as funções que o sistema/aplicativo deve possuir para atender ao negócio e suas regras.

Um RF especifica uma função que o sistema ou componente deve ser capaz de realizar. Estes são requisitos de software que definem o comportamento do sistema. Com base nos objetivos descritos da aplicação, foram levantados os requisitos mostrados na Tabela 4.1.

Tabela 4.1: Requisitos Funcionais
Fonte: Próprio autor

Código	Requisito	Descrição
RF-001	Manter queixas	Ao criar queixa preenchendo todo o formulário, todas informações deverão ser gravadas no banco de dados da aplicação.
RF-002	Criar marcadores	Mapa deverá exibir todas queixas georreferenciadas por pontos no mapa.
RF-003	Filtrar cidades	Usuário ao filtrar pelo nome da cidade na tela de mapeamento o mesmo deverá ser recarregado centralizando o mapa na cidade informada.
RF-004	Exibir informações	Ao clicar nos marcadores que indicam o local de cada queixa, deverá ser exibida um caixa contendo as informações detalhadas da queixa existente.
RF-005	Contatar desenvolvedor	Encaminhar *e-mail* com identificação e a mensagem do usuário ao desenvolvedor por meio do e-mail da aplicação disponível no menu Contato.

Também foram especificados alguns RNF na Tabela 4.2. São requisitos relacionados ao uso da aplicação em termos de desempenho, usabilidade, confiabilidade, segurança, interoperabilidade, disponibilidade, compatibilidade, manutenção e tecnologias envolvidas.

Tabela 4.2: Requisitos Não-Funcionais
Fonte: Próprio autor

Código	Requisito	Tipo	Prioridade
RNF-001	Disponível nos principais navegadores *web*(*Chrome, Mozilla e Edge*)	Compatibilidade	Essencial
RNF-002	Manter padrão de cores em todas páginas	Usabilidade	Requerida
RNF-003	*Navbar* (Barra superior) fixa no topo do site sobrepondo outros elementos	Usabilidade	Alta
RNF-004	Uso de Design responsivo nas interfaces gráficas	Usabilidade	Essencial
RNF-005	Exibir mapa em 100% da largura de tela	Usabilidade	Alta
RNF-006	Indicar campos obrigatórios para formulário aplicados	Usabilidade	Média
RNF-007	Exibir alertas personalizados com ícones para todas ações realizadas	Usabilidade	Essencial
RNF-009	Integração com a *API do Google Maps* para marcação de pontos e conversão de endereços	Interoperabilidade	Essencial
RNF-010	Integração com *API* do Via Cep para auto completar campos de formulários	Interoperabilidade	Essencial
RNF-011	Cores de fundo da aplicação em degradê azul	Usabilidade	Requerido
RNF-012	Ícone da aplicação no formato de um marcador (Pino de Mapa)	Usabilidade	Média

4.3 Preparação do Ambiente

Para que fosse possível desenvolver a aplicação localmente e aplicar testes, foi usado notebook com processador *Intel(R) Core(TM)* i7-5500U, 8 GB(*Giga bits*) de memória *RAM*, SSD ou unidade de estado sólido de 480 GB. O sistema operacional usado foi *Windows 10 Home* com arquitetura de 64 *bits*.

Com a instalação do *Wampserver* toda parte de configuração do servidor local *Apache 3.0.6* é feita automaticamente, assim como instalação das seguintes tecnologias: *PHP* nas versões 5.6.31, 7.0.23, 7.1.9 – *MySQL* 5.7.19 – *PhpMyAdmin* 4.7.4, essenciais para desenvolvimento do projeto.

De posse da tabela que conta com 5.570 registros de todos municípios brasileiros e estão em conformidade aos parâmetros do IBGE (Instituto Brasileiro de Geografia e Estatística) e também já no formato .sql, conforme texto detalhado no capítulo 3 tópico 3.7.1, a extensão .sql se refere a tipos de arquivos reconhecidos pelo gerenciador do banco de dados *PhpMyAdmin* entre outros, simplesmente foi criado um banco de dados "municipios_brasileiros"e feito *import* da tabela "municipios"para ele. Logo em seguida foi feita a criação da tabela "queixas", responsável por armazenar todas queixas que serão informadas pelos usuários, o *DER* das duas tabelas podem ser observadas na Figura 4.1.

As duas tabelas não mantém relacionamento entre si, cada uma foi usada para finalidade específica que não necessitariam serem relacionadas. Na tabela "municípios", existem colunas que se referem a código IBGE, nome do município, código UF, UF, estado, latitude e longitude, como pode ser observado na Tabela 4.3, a disposição dos dados da mesma.

Figura 4.1: DER das tabelas do banco
Fonte: Próprio autor

Tabela 4.3: Exemplo da disposição dos dados na tabela de municípios
Fonte: Próprio autor

Código IBGE	Nome do Município	Código UF	UF	Estado	Latitude	Longitude
5200050	Abadia de Goiás	52	GO	Goiás	-16.7573	-49.4412
3100104	Abadia dos Dourados	31	MG	Minas Gerais	-18.4831	-47.3916
5200100	Abadiânia	52	GO	Goiás	-16.1970	-48.7057

Já a tabela de "queixas", foi criada para armazenar dados informados pelos usuários via formulário. No ato da criação de cada queixa, é registrado o nome do usuário, nome da cidade, estado, cep, bairro, rua, número, tipo de queixa, descrição dos problemas informados, latitude/longitude e data, onde os 3 últimos campos da tabela são preenchidos automaticamente pela aplicação no momento da solicitação. Disposição dos dados na Tabela 4.4.

Tabela 4.4: Exemplo da disposição dos dados na tabela de queixas
Fonte: Próprio autor

Nome	Município	UF	CEP	Bairro	Rua	Número	Tipo
Sebastião	Formiga	MG	37256000	Vila	A	125	Saneamento
Maria	Arcos	MG	36564000	Centro	B	64	Iluminação
Daniela	Japaraíba	MG	38152000	Esperança	C	256	Sinalização

4.4 Api's Google Maps

É um serviço público e gratuito que qualquer pessoa pode usar em seus sites e aplicações, desde que o usuário final não seja cobrado pelo uso do serviço. Na verdade o *Google Maps* possui várias *API's* que podem ser incorporadas ao site/aplicação dependendo de cada caso. Para usar a *Google Maps JavaScript e Geocoding*, foi necessário obter uma chave seguindo as

Descrição	Latitude	Longitude	Data
Esgoto a céu aberto	-50.3804	-20.1190	2018-09-18 22:10:00
Lâmpadas queimadas	-16.7573	-49.2837	2018-09-22 17:03:10
Falta faixa de pedestres	-20.6589	-49.4412	2018-09-30 13:22:54

instruções da própria *Google*, por meio do link[1] que poderá ser adicionada ao seu aplicativo ou *website*. A chave é usada para rastrear solicitações de *API* associadas ao projeto para uso.

4.5 Implementações

A partir desta seção todo conteúdo é dedicado a demonstrar o desenvolvimento das telas da aplicação web, tanto no *front-end* usando classes do *framework Bootstrap* para agilizar a criação dos elementos e fazendo com que a aplicação seja responsiva, se adaptando a qualquer tamanho de tela, também é mostrado a parte de *back-end* que faz comunicação com servidor *web*, banco de dados e requisições a *webservices e API's*.

4.5.1 Grid Bootstrap

Como foi proposto ao longo da monografia que a aplicação seria responsiva para se adaptar a diversos dispositivos, todos elementos criados para as páginas são implementados dentro usando um *grid* de referência, a do *Bootstrap*. O sistema de grade do Bootstrap baseia-se em 12 colunas e usa uma série de contêineres, linhas e alinhamento do conteúdo.

Este grid inclui cinco camadas de classes predefinidas para criar *layouts* responsivos. Sendo possível personalizar o tamanho de suas colunas em dispositivos extra pequenos, pequenos, médios, grandes ou extra grandes, para cada tamanho de tela diferente é definido quantas colunas cada elemento *HTML* irá ocupar.

Exemplo: Dois blocos de texto lado a lado definidos pela classe .col-md-6 onde cada um deles usará 6 colunas totalizando as 12 que abrangem a tela toda, isso em dispositivos de tela mediana maiores que 767px de largura. Imagine esses mesmos dois blocos em uma tela muito inferior, os textos ficariam em blocos muito pequenos e difíceis de ler, por esse motivo para os mesmo blocos é usada a classe .col-sm-12, assim quando for visualizado em tela menor, o primeiro bloco ocupa toda a tela fazendo com que o segundo bloco seja mostrado abaixo ocupando 12 colunas também. Aspectos de grid para diferentes tamanhos de telas na Figura 4.2.

4.5.2 Desenvolvimento da index

Qualquer acesso às páginas da aplicação se dá a partir do arquivo em *PHP index*, construído da seguinte forma, foi declarada uma variável que fica encarregada de receber via *GET* o conteúdo do parâmetro 'i' contendo o nome da página a ser acessada. É atribuído a essa variável por padrão a palavra "*home*", para carregar a página inicial pela primeira vez.

A *index* faz inclusão do cabeçalho e rodapé automaticamente ao corpo da página solicitada, além de um botão flutuante que fica visível em toda aplicação. A cada nova solicitação, a *index*

[1] https://cloud.google.com/maps-platform/

	Extra pequeno <576px	Pequeno ≥576px	Médio ≥768px	Grande ≥992px	Extra grande ≥1200px
Largura máxima do contêiner	Nenhum (auto)	540 px	720 px	960 px	1140px
Prefixo de classe	`.col-`	`.col-sm-`	`.col-md-`	`.col-lg-`	`.col-xl-`

Figura 4.2: Aspectos da grid Bootstrap
Fonte: Site oficial do Bootstrap

redireciona o usuário para uma nova tela. Exemplo visto na Figura 4.3.

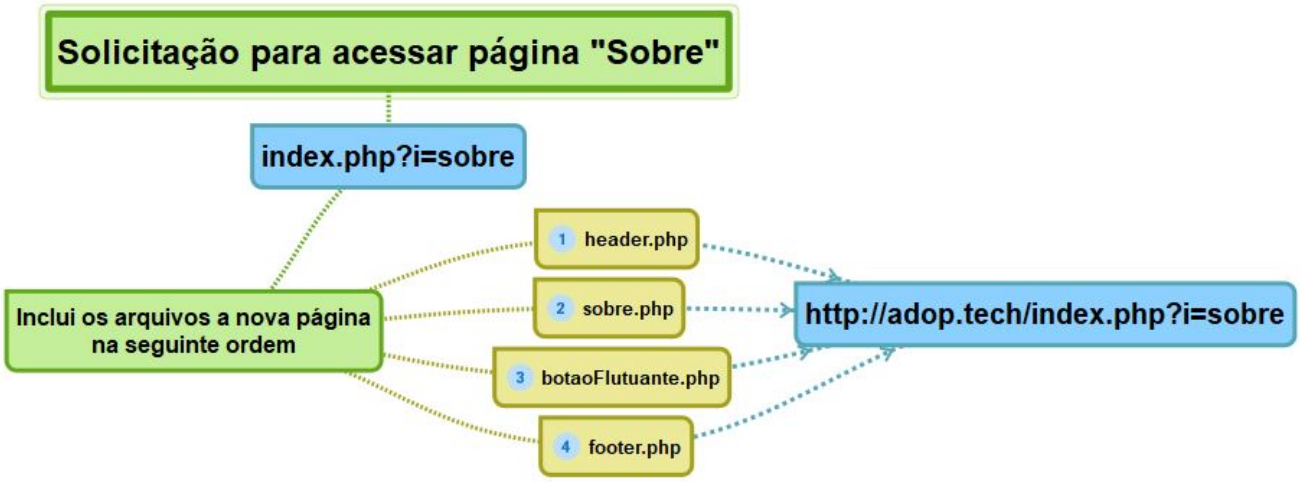

Figura 4.3: Exemplo da construção das páginas
Fonte: Próprio autor

Cabeçalho

Para que todas as páginas tenham um cabeçalho, é feito *include* do *arquivo header.php* que contém o código que o implementa fixado no topo com logo e lista das páginas disponíveis para navegação, em telas de tamanho inferior a 425px de largura essa lista é transformada em um ícone conhecido como menu sanduíche, contendo a lista. Resultado na Figura 4.4.

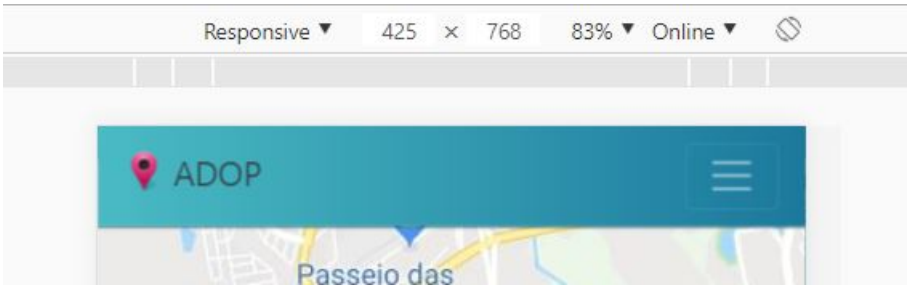

Figura 4.4: Cabeçalho e menu sanduíche
Fonte: Próprio autor

Rodapé

Assim como o cabeçalho está presente em todas as páginas da aplicação, o mesmo acontece com o rodapé que tem seu arquivo *footer.php* também incluído na *index*, contando com algumas informações como nome de projeto e do desenvolvedor, além do nome da faculdade e curso exercido. Para que usuários possam encontrar o autor do projeto nas redes sociais, foram adicionados botões para o *LinkedIn, Instagram e Whatsapp*. Os botões em formato circular se expandem quando a seta do mouse passa por eles, isso acontece devido aos estilos aplicados com *CSS*. Resultado na Figura 4.5.

Figura 4.5: Rodapé
Fonte: Próprio autor

Botão Flutuante

Este botão está disponível em qualquer tela da aplicação, para que o usuário possa criar queixas sem a necessidade de voltar a página de início, local onde originalmente fica o botão principal de criação de queixas. Para não atrapalhar a experiência de navegação ele está posicionado flutuando no canto inferior direito de todas as páginas. O botão *feedback* logo acima dele, tem a função de levar o usuário a uma pesquisa de uso da aplicação que será mostrada no fim deste capítulo. Resultado na Figura 4.6.

Figura 4.6: Botão Flutuante
Fonte: Próprio autor

4.5.3 Página Inicial

A tela inicial desenvolvida é ligeiramente simples contando com poucos elementos e textos que expõem a ideia da aplicação. Na parte superior foi adicionada uma imagem ilustrativa que remete à função central que é mapear problemas citados pelos usuários. Essa imagem foi fixada no topo da *tag* <*body*> de forma que todo o restante do site role sobre, dando efeito chamado de *Parallax*[2]. Um bom exemplo desse efeito é o jogo Super Mario, onde plano de fundo se move lentamente enquanto o personagem Mario se desloca rapidamente.

[2]Técnica de mover imagens de fundo em uma velocidade mais lenta do que os demais conteúdos de primeiro plano, causando uma ilusão de profundidade nas interfaces.

Sobre a imagem existem o nome da aplicação em formato de sigla e o botão principal de criação de queixas.

Para reforçar esse vínculo social da aplicação e informar os usuários, foram adicionados 3 *cards* com matérias relacionadas à comportamento, relacionamento em sociedade e ouvidorias públicas. No topo de cada um dos cards foi colocada uma imagem que ilustra o que o texto tem a dizer, logo abaixo o título e um resumo, um *link* de "Leia mais", foi adicionado ao rodapé, o leitor ao clicar, tem acesso ao conteúdo completo da matéria por meio de um *modal* (nome da classe que compõe uma caixa flutuante), onde podem ser colocados diversos outros elementos. A Figura 4.7 mostra como esse *modal* é exibido na tela.

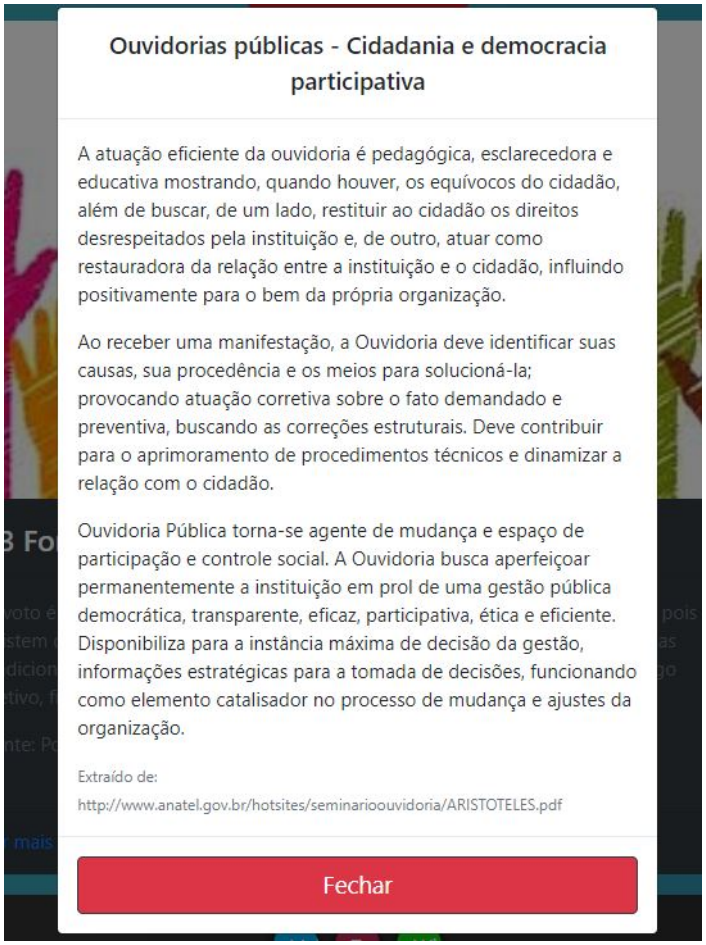

Figura 4.7: Modal das matérias
Fonte: Próprio autor

Como pode ser observado na Figura 4.8, os 3 cards estão posicionados na mesma linha criada com a classe *row*, um ao lado do outro com uma pequena margem entre si, onde dentro da *row* foi inserido um bloco de conteúdos com a *tag div class="card-group"* e antes de seu fechamento foram colocados 3 blocos que dão origem aos *cards*. Ao usar *cards-group*, todos cards colocados dentro do bloco se auto redimensionam e fazem quebra de linha, fazendo com que cada *card* fique abaixo de outro, dependendo do tamanho de tela em que o conteúdo esteja sendo exibido.

Figura 4.8: Cards com notícias
Fonte: Próprio autor

O formulário mostrado na Figura 4.9, faz parte de um *modal* que é exibido com ação de clique do *mouse* sobre os botões de criação de queixas, tanto o principal quanto o flutuante citado anteriormente. Cada um dos campos seguem padrão de criação para que sejam responsivos, e as bordas dos campos obrigatórios que usam o atributo "*required*", do próprio *HTML*, ficam vermelhas com uso da classe *was-validated* do *Bootstrap*, até que o usuário entre com dados nos campos, tonando-os verdes quando validados.

Figura 4.9: Formulário de queixas
Fonte: Próprio autor

Pensando em facilitar o preenchimento do formulário a aplicação usa *Ajax* para trazer dados do *webservice* ViaCEP assim que o campo CEP é preenchido. Por meio deste serviço é possível auto completar os campos cidade e UF em cidades com número de CEP único, mas caso exista um por rua, os campos bairro e rua também são completados. A aplicação faz uma requisição quando o campo CEP perde o foco, ou seja, ao clicar em outro campo.

Uma função em *Javascript* captura o conteúdo do campo CEP e faz a requisição via *GET* com *Ajax*, usando uma *url* do serviço, com o formato desejado no retorno, no caso *JSON*. Se ocorrer um erro, uma mensagem é mostrada pedindo a revisão do CEP informado, exemplo da requisição e retorno na Figura 4.10.

Em caso de sucesso na requisição, o *Ajax* preenche os campos citados automaticamente usando objetos *JSON* com todas informações necessárias. Quando todo formulário for completado e o usuário clicar no botão criar, a queixa é coletada por outra função *Javascript/Ajax*, passando os dados ao arquivo grava_queixas.php, que faz a gravação na tabela queixas presente no BD e informa ao *Ajax* se ouve ou não algum erro que posteriormente disparará uma mensagem ao usuário, como pode ser visto na Figura 4.11.

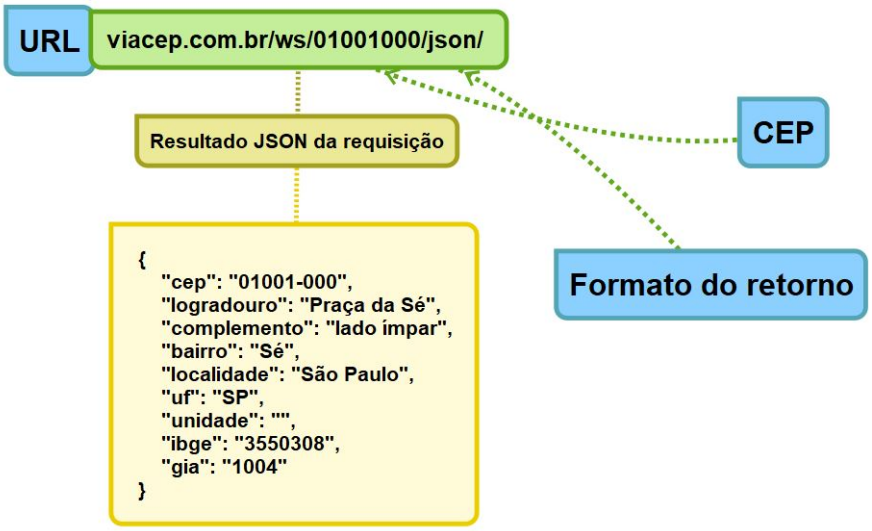

Figura 4.10: Exemplo requisição ao ViaCEP
Fonte: Próprio autor

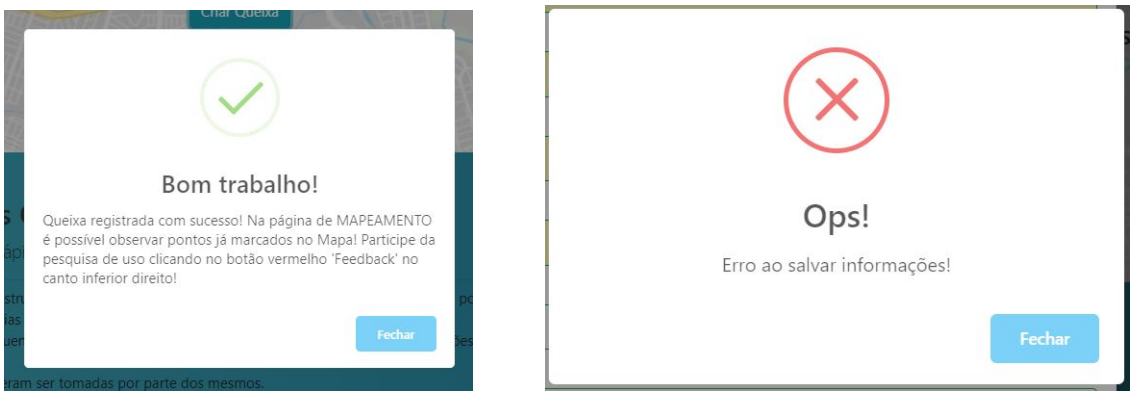

Figura 4.11: Mensagens de alerta
Fonte: Próprio autor

Toda aplicação usa mensagens de alerta *SweetAlert*, uma biblioteca *Javascript* carregada via *CDN*, para usa-lá utilizamos o seguinte *script* adicionado dentro corpo da *tag* <*head*>.

src="https://unpkg.com/sweetalert/dist/sweetalert.min.js"

4.5.4 Página Mapeamento

A tela de mapeamento foi desenvolvida com um mapa de elemento principal e um formulário com o campo cidades, para que fosse possível centralizar o mesmo em qualquer cidade de todo território nacional. Para desenhar o mapa, basta seguir basicamente três passos, incorporar a biblioteca da *API do Google*, designar um elemento <*div id="mapa"* no HTML, onde será desenhado o mapa, inserir o trecho de código *JavaScript* para definir as opções do mapa instanciando no objeto map, por fim, criar os *marker's* (marcadores) das queixas com seus respectivos *infoWindow*[3], que são adicionados ao mapa.

[3]Pequenas caixas ou balões de conteúdo, normalmente textos informativos que podem ser acessados clicando nos marcadores.

A incorporação da biblioteca é feita através de uma *url da API*, que recebe como parâmetro a chave/*Key1* de acesso obrigatória. No trecho de código em *JavaScript* uma função "initMap" chamada ao incorporar a *API* no projeto, é criado objeto mapa na instância de "*google.maps.Map*", que recebe o id="mapa", do elemento e as opções iniciais do mapa. O atributo "center" recebe as variáveis latitude e longitude pelo parâmetro da função, que são coordenadas que centralizam o mapa, por padrão elas recebem valores referentes a localização da cidade de Cristais-MG. O "zoom" define a altura inicial da visualização, várias definições podem ser aplicadas, mas até aqui somente as citadas são usadas.

Marker's são pontos no mapa, definidos por uma posição geográfica e o ícone, sendo que este último é personalizado de acordo com tipo da queixa registada. Mas para criá-los juntamente com as *infoWindow* são necessários os dados gravados no BD. A função *downloadUrl* executa chamada a um arquivo *PHP*, responsável por buscar os dados e montar XML, que é entregue dentro do *Javascript*. A Figura 4.12 mostra parcialmente o código emissor do XML.

```php
// #### Usar funções DOM do PHP para emitir XML
$dom = new DOMDocument("1.0", "UTF-8");
$node = $dom->createElement("markers");
$parnode = $dom->appendChild($node);

header("Content-type: text/xml");
while($row = $resultado_query->fetch(PDO::FETCH_ASSOC)) {
  $node = $dom->createElement("marker");
  $newnode = $parnode->appendChild($node);
  $newnode->setAttribute("Cidade", $row['nomeCidade']);
  $newnode->setAttribute("Estado", $row['nomeEstado']);
  $newnode->setAttribute("Bairro", $row['nomeBairro']);
  $newnode->setAttribute("Rua", $row['nomeRua']);
  $newnode->setAttribute("Numero", $row['numero']);
  $newnode->setAttribute("Tipo", $row['tipoQueixa']);
  $newnode->setAttribute("Descricao", $row['descricaoQueixa']);
  $newnode->setAttribute("Data", $row['dataQueixa']);
  $newnode->setAttribute("lat", $row['lat']);
  $newnode->setAttribute("lng", $row['lng']);
}
echo $dom->saveXML();
```

Figura 4.12: Código emissor do XML
Fonte: Próprio autor

Por meio de um laço de repetição todas informações como cidade, estado, bairro, rua, número, tipo, descrição, data, além de latitude e longitude de cada queixa são passadas para variáveis, incluindo a definição dos ícones personalizados e criação da *infoWindow*. Com todos valores já atribuídos, a variável *marker* define para cada um dos marcadores, informações como posição geográfica em que serão aplicados e uma ação de clique para visualizar informações dos locais registrados pelos usuários. Assim que a aplicação faz a requisição à *API*, o mapa é criado e todos pontos georreferenciados são exibidos, como pode ser visto na Figura 4.13.

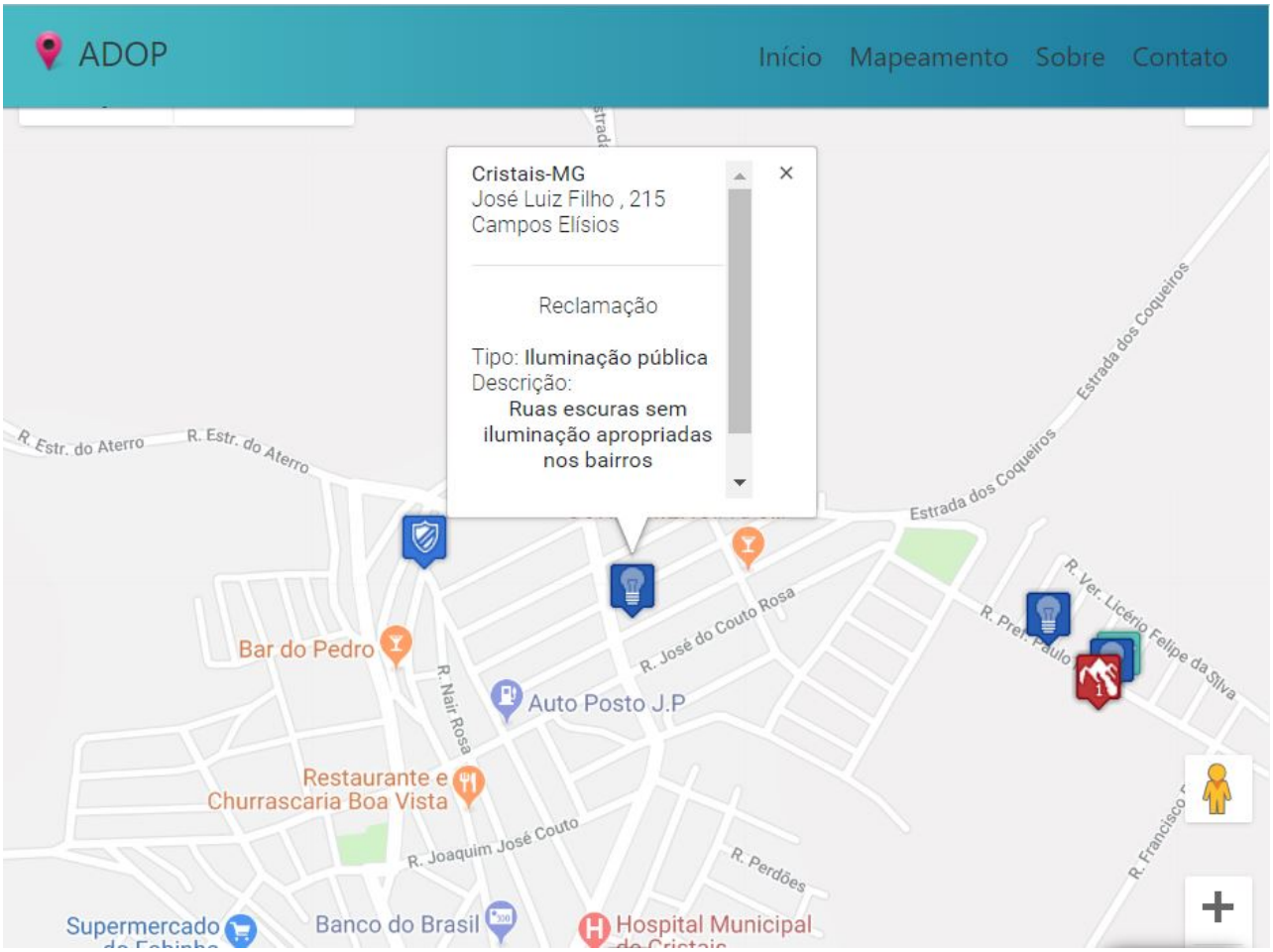

Figura 4.13: Mapa com Marcadores
Fonte: Próprio autor

Outro elemento importante existente na página de mapeamento é o formulário com campo de texto e o botão "Filtrar", para inserir uma cidade e centralizar o mapa naquela região. Este formulário como todos outros usados na aplicação segue padrão dos oferecidos pelas classes do *Bootstrap*. No campo disponível para usuário ao digitar 4 letras referentes ao nome da cidade, a aplicação exibe uma lista com várias cidades para completar o campo selecionando uma delas. Na Figura 4.14 é possível ver o recurso sendo usado.

Figura 4.14: Filtro de cidade
Fonte: Próprio autor

Para realizar essa pesquisa possibilitando dar ao usuário a opção de completar o nome da cidade com apenas um clique, a aplicação realiza uma busca no banco de dados, especificamente na tabela de municípios. Como foi dito anteriormente, a busca (*SELECT* no banco de dados) só é feita a partir da quarta letra digitada, para não sobrecarregar a comunicação entre aplicação e servidor, mostrando opções mais condizentes ao que o usuário deseja.

Esse recurso atualiza um elemento posicionado logo abaixo do formulário no *HTML*, mostrando uma nova lista sem que seja necessário o recarregamento da página, devido ao uso de *Javascript e PHP*. Na Figura 4.15 é possível ver a função responsável por implementar a funcionalidade, por meio do método *.getJSON*, um classe em *PHP* é chamada e retorna um objeto com os dados da tabela, que logo em seguida são incluídos em um *array* (vetor) e mostrados ao usuário.

```
//AUTO COMPLETA CIDADE
$(document).ready(function() {
    // Captura o retorno do buscando.php
    $.getJSON('buscando.php', function(data){
      //array
      var dados = [];

      // Armazena no array somente o nome da cidade
      $(data).each(function(key, value) {
        dados.push(value.nome_municipio);
      });

      // Seta o elemento HTML com array por meio do id
      $('#cidade').autocomplete({ source: dados, minLength: 4});
    });
});
```

Figura 4.15: Código para filtrar cidade
Fonte: Próprio autor

Para que o mapa seja recarregado e já centralizado sobre a cidade informada no campo de filtro, a aplicação utiliza *Ajax*. Assim apenas o elemento onde o mapa é carregado é atualizado, não necessitando o recarregamento total da página, reduzindo consumo de dados e melhorando a usabilidade do sistema.

Ao clicar no botão Filtrar, uma função em *Javascript* é disparada, que captura a cidade informada no campo de filtro e encaminha via *POST* pelo *Ajax* ao arquivo em *PHP*, que por sua vez busca no banco de dados por meio de um *SELECT*, compara a cidade informada pelo usuário com as existentes e responde com um objeto *JSON*, com o status, latitude e longitude.

O *Ajax* tem funções padrão para cada tipo de retorno das requisições, mas somente duas são usadas no filtro para cidades, *success* e *error* como visto na Figura 4.16.

Figura 4.16: Funcionamento da função Ajax
Fonte: Próprio autor

Na função **success**, o campo formulário é limpo possibilitando uma nova consulta, a latitude e longitude retornados pelo *PHP* são passados para variáveis do JS e colocadas no parâmetro da

função *initMap(lat,long)* que invoca o mapa novamente, fazendo com que ele atualize na posição da cidade desejada caso o status seja verdadeiro (*true*). Nos casos em que status retorna como falso (*false*), o usuário é informado de que a cidade não foi encontrada por meio de um alerta que segue os mesmos padrões mostrados na sessão 4.5.3.

Se a função **error** for chamada é sinal de que o *PHP* não conseguiu realizar a consulta com a cidade informada, então um outro alerta de erro é disparado na tela, para que o usuário tente novamente.

4.5.5 Página Sobre

Criada com intuito de falar sobre a idealização do projeto, desenvolvedor, suas propostas e objetivos. Títulos e blocos com textos são dispostos na página de forma responsiva. Todo o conteúdo sofre quebras de linhas se adaptando a diferentes tamanhos de telas. A página conta também com um botão "leia mais", que dá acesso a essa monografia no formato PDF, que é aberto em uma nova guia do navegador. Parte da tela pode ser vista na Figura 4.17.

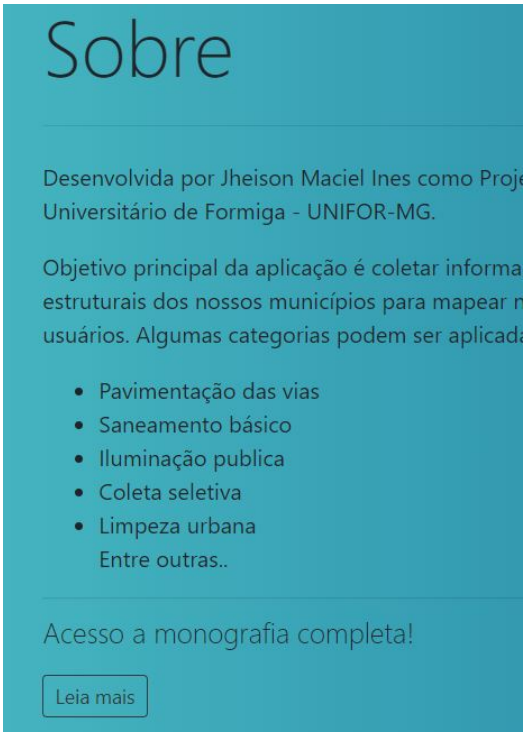

Figura 4.17: Recorte da página sobre
Fonte: Próprio autor

4.5.6 Página Contato

Na página contato o intuito é manter interação mais direta com pessoas que fazem o acesso à aplicação, por meio da troca de *e-mails*. A tela conta com um pequeno formulário visto na Figura 4.18, com os campos nome, *e-mail* e mensagem, que o usuário pode usar para enviar suas sugestões ou críticas e até mesmo expressar interesse no projeto para algum fim que não seja lucrativo.

Figura 4.18: Formulário de contato
Fonte: Próprio autor

Ao preencher os campos obrigatórios e clicar no botão Enviar, uma função JS é iniciada e prepara os dados do formulário para serem enviados por uma requisição *Ajax* ao *PHP*, que faz comunicação com servidores de *e-mail* usando biblioteca *PHPMailer*.

PHPMailer é uma biblioteca *PHP* de código aberto para o envio de *e-mails*, com suporte a HTML, diferentes protocolos e níveis de autenticação. Por possuir uma implementação do protocolo *SMTP* integrada, essa biblioteca pode ser utilizada em plataformas que não possuem um servidor para envio de *e-mails* nativos, como o *Windows*. Para usar as classes da PHPMailer, o código fonte da biblioteca foi baixado de um repositório oficial no *GitHub*[4] e colocado na pasta do projeto.

No arquivo *PHP*, responsável por enviar o *e-mail* e responder a requisição *Ajax* feita anteriormente, é feita inclusão de um dos arquivos da biblioteca, o *autoload.php*. A partir daqui já é possível criar uma instância da classe *PHPMailer* e configurar os dados do envio.

Caso algum erro venha ocorrer no retorno do *PHP*, a função *error* do *Ajax* se encarrega de mostrar alerta ao usuário. Para sucesso no envio, a função *success* informa que a mensagem foi enviada ao *e-mail* da aplicação. Como já foi citado, *Ajax* tem diversas funções para tratar retorno das requisições, no caso do envio de mensagens também foi usada a função *beforeSend*.

Ela é executada entre envio e resposta do servidor, já que o processo requer tempo maior de espera, dentro dela é desabilitado o botão de envio, evitando que usuário tente enviar novamente, duplicando o *e-mail* na caixa de entrada.

4.5.7 Resultados

Depois de finalizar desenvolvimento, foi criado um domínio http://adop.tech/ online para a aplicação e feita a hospedagem na empresa Hostinger, tornando acessível em quaisquer dispositivos com acesso a *Internet*. As propostas apresentadas no primeiro capítulo foram concluídas

[4]https://github.com/PHPMailer/PHPMailer

em um nível satisfatório, dando ao usuário a possibilidade de expor suas queixas, em sua maioria da rua onde moram de forma simples e rápida, já que todo processo não leva mais de um minuto.

Algumas ressalvas podem ser feitas quanto ao mapeamento das queixas, pois várias delas não obtiveram êxito na marcação exata ou o mais aproximada possível do ponto informado no ato da criação da queixa. Esse erro ocorreu em alguns casos onde o próprio usuário digitou um endereço errado ou incompleto, mas também ocorreram falhas da *API Google Geocoding* que respondeu com coordenadas geográficas diferentes das corretas para o endereço fornecido. A Figura 4.19 exemplifica melhor o erro ao mapear a queixa, o marcador está fixado em um ponto fora da rua, porém foi colocado no mesmo bairro indicado pelo endereço.

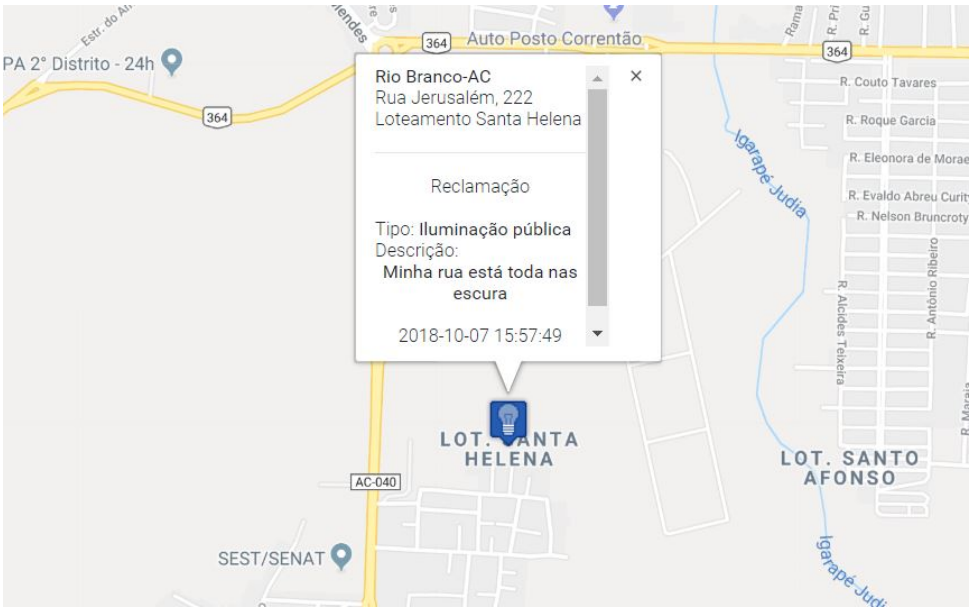

Figura 4.19: Marcação feita local errado
Fonte: Próprio autor

Levando em consideração que todos informações ficam armazenadas no banco de dados da aplicação, o fato de algumas queixas não terem sido mapeadas corretamente pode de certa forma ser relevado. O mapa que mostra todas as queixas aos usuários se comportou de forma responsiva assim como todo restante da ferramenta.

O conceito de responsividade também é outro ponto positivo que obteve nível satisfatório na aplicação *web*. Todas telas se comportaram bem, tanto nos testes feitos durante o desenvolvimento quanto nos feitos posteriormente a aplicação estar disponível na *Internet*. Na Figura 4.20 ilustrativa é possível ver exemplo como ela se comporta em diferentes dispositivos e tamanhos de telas.

Figura 4.20: Acesso a aplicação por diferentes dispositivos
Fonte: Próprio autor

Figura 4.20(A) - ***Tablet***

Representada por um *IPad* de 768x1024px que mostra a aplicação em um formato reduzido, com conteúdos em proporção menor, já sofrendo alterações e se mantendo responsiva.

Figura 4.20(B) - ***Mobile***

Mostra uma tela bem menor de um *IPhone* com 320x480px, onde se pode notar uma mudança significativa no layout, é notada presença de um ícone na parte superior do lado direito e a imagem ilustrativa da aplicação ocupando quase toda tela.

Figura 4.20(C) - ***Desktop***

A melhor visualização de componentes como textos, mapa, formulários, *cards* entre outros pode ser notada em *desktop's* que contam com resoluções similares a que foi disposta a aplicação, ocupando 1600x992px de um *Mac*.

Figura 4.20(D) - **Notebooks**

Em dispositivos que contam com resolução aproximada a 1280x802px de *MacBook's* a visualização de todas telas da aplicação se mantém sem nenhuma alteração de *layout*, possibilitando uma boa interação com a aplicação.

A aplicação criada com intuito de se tornar uma alternativa as ouvidorias públicas tradicionais, foi acessada e testada por diversas pessoas de diferentes regiões do país, com representação de vários estados. Assim que criada uma queixa, o usuário foi convidado a participar de um questionário online com 7 questões de múltipla escolha que avaliam o uso da aplicação *web*.

Resultados da avaliação de uso da ADOP aplicado a 30 usuários.

A 1ª questão a ser respondida pelos usuários, perguntava qual dispositivo ele usou para utilizar a aplicação, a maior parte somando 77% - 23 acessos, foi por meio dos Smartphones seguido dos computadores com 23% - 7 acessos. Apêndice A Figura A.1

Na 2ª questão foi perguntado qual navegador foi usado para acessar a aplicação, o Google Chrome foi o destaque ao ser usado em 83% dos casos, totalizando 25 vezes, seguido pelo Safari com 10% - 3 acessos e o Mozilla Firefox 7% - 2 acessos. Apêndice A Figura A.2

Na 3ª questão foi perguntando se durante a navegação pelas páginas o layout se comportou de forma estranha mostrando elementos desalinhados, em 100% das respostas os usuários responderam Não, atestando que a aplicação se adaptou tanto nos smartphones quanto nos computadores independente do navegador e dispositivo usado. Apêndice A Figura A.4

Ao serem questionados sobre mensagens de erros na tela, nenhum usuário relatou problemas, porém quando a marcação correta da queixa no endereço fornecido, 4 delas não obtiveram êxito, nos outro 26 casos foi relatado sucesso na marcação. Apêndice A Figura A.5 e Figura A.6

Em uma nota com escala de 1 a 5 referente à facilidade de uso da aplicação, 92% das 30 avaliações, cerca de 27 deram nota 5 o outros 8% deram nota 4 para a aplicação. Já quando o quesito era desempenho no carregamento das páginas 29 avaliações deram nota 5 e apenas uma avaliou com nota 4. Apêndice A Figura A.7 e Figura A.4

Os resultados atestaram o desempenho da aplicação em diversos aspectos em situações reais de uso, a aplicação não foi submetida a nenhuma ferramenta de teste quanto a performance, para expor todos dados foi utilizado como base apenas a opinião de 30 usuários.

Capítulo

5

Conclusões

5.1 Considerações Finais

Como foi descrito no início desta monografia pelo capitulo 1, é importante que todos cidadãos tenham participação ativa na sociedade. Uma das formas de promover essa interação é por meio das ouvidorias públicas. Quando todos participam contribuindo positivamente no meio em que vivem, as transformações e decisões tomadas pelo governantes são benéficas à sociedade em sua maioria.

Para possibilitar que a população em geral possa ter poder de participação maior, a aplicação *web* ADOP (Alternativa Digital as Ouvidorias Públicas) foi idealizada e desenvolvida utilizando ferramentas e tecnologias com foco em dispositivos com acesso a *Internet*. Como o próprio nome já sugere, ela tem o intuito de ser uma alternativa às ouvidorias públicas. A aplicação é capaz de coletar as reclamações frequentes da população por meio de um formulário onde cada um pode deixar quantas contribuições quiser.

Todas esses queixas são armazenadas em um banco de dados *MySQL* que posteriormente pode ser requisitado e usado em prol das mudanças necessárias, qualquer pessoa ao acessar a aplicação tem acesso a um mapa com todas queixas mapeadas. Os marcadores que indicam o ponto onde existe uma queixa contém informações sobre o mesmo, podendo ser visualizadas apenas com um clique no marcador desejado.

Durante o desenvolvimento os objetivos propostos foram atingidos, a aplicação obteve êxito ao possibilitar que fosse acessível de qualquer dispositivo, independente do tamanho da tela ou modelo, dependendo apenas de uma conexão com a *Internet*, facilitando a criação das queixas de forma consideravelmente rápida e simples. Como foi proposto o mapeamento das queixas, todas podem ser visualizadas por meio de um mapa no menu de mapeamento.

Assim como toda aplicação, a ADOP pode evoluir com implementação de diversas melhorias no seu funcionamento, o sistema de mapeamento por exemplo marca algumas queixas no

mapa sem precisão necessária, fazendo com que os locais não estejam de acordo com endereço informado pelo usuário. Novas funcionalidades podem ser atribuídas ao projeto, tais como, vincular a aplicação diretamente aos bancos de dados das prefeituras, fazer com que as queixas possam ser acessadas e compartilhadas em redes sociais, entre algumas outras possibilidades.

5.2 Contribuições

A desenvolvimento da aplicação trás benefícios importantes para todos que venham a usa-la, expondo problemas do meio em que vive e também para aqueles que buscam conhecimentos específicos de algumas tecnologias apresentadas neste trabalho.

De certo modo, é muito difícil que alguém ou algum grupo social esteja totalmente excluído de toda a sociedade, geralmente isso ocorre sobre uma parte dela. Inclusão social é democratizar os diferentes espaços para aqueles que não possuem acesso direto a eles, por meio da aplicação isso é possível com a inserção de pessoas na promoção de melhorias para toda sociedade.

5.3 Trabalhos Futuros

Trabalhos futuros poderão ser propostos e implementados em torno da aplicação visando melhorias no mapeamento, atribuição de novas funcionalidades ao sistema como a possibilidade de compartilhar as queixas criadas em redes sociais, integração com base de dados governamentais ou até criação de uma *API* da aplicação para disponibilização de recursos e dados a outros sistemas. Ampliar a aplicação *web* para que possa ser executada em plataformas nativas como a do *Android e IOs*, possibilitando o seu uso sem a necessidade de conexão com a Internet.

Referências Bibliográficas

ALMEIDA, J. de. Oito milhões de pixels em imagens de quatro quilates: 4k. *Imagem Digital. Imprensa Oficial, São Paulo*, p. 222–231, 2011.

ALVES, B.; ROMANI, L. A. S.; OTAVIAN, A. F. Agritempogis: um aplicativo para auxiliar agricultores em processos de tomada de decisão. In: IN: MOSTRA DE ESTAGIÁRIOS E BOLSISTAS DA EMBRAPA INFORMÁTICA AGROPECUÁRIA, 12., 2016, CAMPINAS. RESUMOS EXPANDIDOS... BRASÍLIA, DF: EMBRAPA, 2016. *Embrapa Informática Agropecuária-Artigo em anais de congresso (ALICE)*. [S.l.].

ANDRIOLO, E. L. Interação na web: um estudo sobre interfaces e conceitos de desenvolvimento. 2018.

ARAUJO, R. et al. Georreferenciamento de fotografias aéreas e análise da variação da linha de costa. *Métodos en Teledetección Aplicada a la Prevención de Riesgos Naturales en el Litoral*, p. 123–138, 2009.

BASSO, M. et al. Mb engine: Game engine para a construção de jogos em html 5. *Anais do VI EATI: VI Encontro Anual de Tecnologia da Informação. Instituto Federal Farroupilha: Frederico Westphalen*, 2015.

BASTOS, R.; JAQUES, P. A. Antares: Um sistema web de consulta de rotas de ônibus como serviço público. *Revista Brasileira de Computação Aplicada*, v. 2, n. 1, p. 41–56, 2010.

BODOT, J. W.; AGNER, W. R. F. Desenvolvimento web de sistema de movimentação financeira para empresas de pequeno porte. *Semanaacademica. com, Guarapuava*, v. 1, n. 53, p. 1–28, 2014.

BORNHOFEN, P. R.; TENFEN, E. Mapeamento criminal por meio da plataforma google maps. *SEGURANÇA PÚBLICA*, p. 82, 2009.

BORTOLI, N. d. S. de; RUFINO, R. R. Conceito para o desenvolvimento web utilizando spring boot, bootstrap e angular js.

BORTOLOSSI, H. J. Criando conteúdos educacionais digitais interativos em matemática e estatística com o uso integrado de tecnologias: Geogebra, javaview, html, css, mathml e javascript. *Revista do Instituto GeoGebra Internacional de São Paulo. ISSN 2237-9657*, v. 1, n. 1, 2012.

CAGNIN, M. I.; PENTEADO, R. A. *Avaliação das vantagens quanto à facilidade de manutenção e expansão de sistemas legados sujeitos à engenharia reversa e segmentação*. Tese (Doutorado) — Dissertação de Mestrado, DC-UFSCar, 1999.

CARDOSO, A. S. R. *Ouvidoria pública como instrumento de mudança*. [S.l.], 2010.

CHAVES, A. M.; SILVA, G. proposta de uma arquitetura de software e funcionalidades para implementação de um ambiente integrado de desenvolvimento para a linguagem php. i jornada científica e vi fipa do cefet bambuí. *Centro Federal de Educação Tecnológica de Bambuí. Bambuí*, p. 5, 2008.

CYRILLO, R. M.; ALVES, V. C.; OLIVEIRA, A. L. L. de. Ouvidoria e participação cidadã. *Composição do CNMP*, p. 71, 2018.

FERREIRA, J. E.; TAKAI, O. K.; FINGER, M. Banco de dados. *Instituo Federal do Espírito Santo, Vitória*, 2009.

FLATSCHART, F. *HTML 5-Embarque Imediato*. [S.l.]: Brasport, 2011.

FRANÇA, S. dos S. Web design responsivo: caminhos para um site adaptável. *Interfaces Científicas-Exatas e Tecnológicas*, v. 1, n. 2, p. 75–84, 2015.

GIRALDI, B.; PESSOA, M. et al. Sistema informatizado de gerenciamento dos processos de introdução de pragas quarentenárias e de bioagentes exóticos de controle-gerprocquarentena. In: IN: CONGRESSO INTERINSTITUCIONAL DE INICIAÇÃO CIENTÍFICA, 6., 2012, JAGUARIÚNA.[ANAIS...] JAGUARIÚNA: EMBRAPA MEIO AMBIENTE, 2012. 1 CD ROM. Nº 12414. *Embrapa Meio Ambiente-Artigo em anais de congresso (ALICE)*. [S.l.].

GOLDSTEIN, S. *Criação de plataforma de geocoding baseada em serviços Google Maps*. Tese (Doutorado), 2014.

GRILLO, F. D. N.; FORTES, R. P. D. M. *Aprendendo JavaScript*.

GUILHERME, F. C. A atuação das ouvidorias municipais na região metropolitana do estado do rio de janeiro. *Gestão publica-Unisul Virtual*, 2018.

HENKELS, B. Crawler webservice para auxílio ao monitoramento de bolsas e recursos para instituições de ensino superior. 2011.

KRATZ, R. de A. et al. Fabrica de adequação de objetos de aprendizagem. *Brazilian Journal of Computers in Education*, v. 15, n. 3, 2007.

LOUDON, K. Desenvolvimento de grandes aplicações web. *Revista Telfract*, v. 1, n. 1, 2018.

MARTINS, H. M.; ROMANI, L. A. S. Design responsivo aplicado ao website agritempo. In: IN: MOSTRA DE ESTAGIÁRIOS E BOLSISTAS DA EMBRAPA INFORMÁTICA AGROPCUÁRIA, 13., 2017, CAMPINAS. RESUMOS EXPANDIDOS... BRASÍLIA, DF: EMBRAPA, 2017. *Embrapa Informática Agropecuária-Artigo em anais de congresso (ALICE)*. [S.l.].

MARTINS, J.; SILVA, J. M. C. da; FLÔRES, O. C. A linguagem de estilo css: um exemplo de plano de aula* integrando as disciplinas de língua portuguesa e introdução à linguagem de programação. In: *Anais do Encontro Virtual de Documentação em Software Livre e Congresso Internacional de Linguagem e Tecnologia Online*. [S.l.: s.n.], 2017. v. 6, n. 1.

MENDES, J. F. d. S. *Advogados na Web*. Tese (Doutorado), 2017.

MIGUEL, F. d. A. M.; COSTA, J. L. Desenvolvimento de sites responsivos utilizando o framework bootstrap com aplicação de user experience. *São Bernardo do Campo*, 2015.

MILANI, A. *MySQL-guia do programador*. [S.l.]: Novatec Editora, 2007.

MINETTO, E. L. Frameworks para desenvolvimento em php. *São Paulo: Novatec*, 2007.

MIYAHARA, E. K.; MENA-CHALCO, J. P.; CESAR-JR, R. M. Genealogia acadêmica lattes. *São Paulo: Universidade de São Paulo*, 2011.

NETO, V. C. et al. Desenvolvimento e integração de mapas dinâmicos georreferenciados para o gerenciamento e vigilância em saúde. *Journal of health informatics*, v. 6, n. 1, 2014.

NIEDERAUER, J. Desenvolvendo websites com php. *São Paulo: Novatec*, 2004.

OLIVEIRA, A. D. A.; ELER, M. M. Acessibilidade em governo eletrônico: um estudo sobre a aplicação de padrões web em sítios gov. br. *XI Simpósio Brasileiro de Sistemas de Informação*, p. 683–690, 2015.

OLIVEIRA, J. C. de; NETO, W. P. de S.; SANTOS, A. d. P. dos. Aplicando api do google maps para criar mapa interativo. estudo de caso: Campus-viçosa. In: *XIV Simposio Internacional SELPER*. [S.l.: s.n.], 2010.

PEREIRA, O. B. O papel das ouvidorias no estado democrático de direito. *Composição do CNMP*, p. 31, 2018.

PROSTT, M. E. Interface web utilizando design responsivo: um estudo de caso aplicado a smartphones, tablets, computadores e televisores. Universidade Tecnológica Federal do Paraná, 2013.

RICHE, C. A. Ouvidoria, caminho para a civilidade e o diálogo. *Organicom*, v. 7, n. 12, p. 180–182, 2010.

ROSA, D. da; SILVA, T. L. da. Adaptação de interfaces para dispositivos móveis com html5. 2018.

SANTANA, J. de; ROMANI, L.; OTAVIAN, A. Análise comparativa das bibliotecas jquery e dojo. *Embrapa Informática Agropecuária-Comunicado Técnico (INFOTECA-E)*, Campinas: Embrapa Informática Agropecuária, 2011.

SANTOS, L. G. dos; RUFINO, R. R. O impacto no desenvolvimento de produtos computacionais utilizando angular js, spring framework e java.

SCHULENBURG, H. R. W. et al. Arquitetura da informação e as metas de experiência do usuário no contexto da práxis de construção dos elementos gráficos de interface para web. *Projetica*, v. 4, n. 1, p. 179–198, 2013.

SILVA, A. D. A. P. D. Design responsivo: técnicas, frameworks e ferramentas. 2014.

SILVA, P. H. R.; VALENTE, W. A. G. Análise de aplicações web responsivas para múltiplos dispositivos. *Caderno de Estudos em Sistemas de Informação*, v. 2, n. 2, 2015.

SOUZA, W. et al. Informação geográfica voluntária no pantanal: um sistema web colaborativo utilizando a api google maps. *Simpósio de Geotecnologias no Pantanal, Bonito, MS. Proceedings of Simpósio de Geotecnologias no Pantanal (GeoPantanal)*, v. 4, p. 763–772, 2012.

TECHIO, G. B.; CHICON, P. M. M. Implementação dos frameworks bootstrap e foundation aplicados na construção de um objeto de aprendizagem para o ensino da engenharia de software. v. 6, n. 1, p. 37–44, 2016.

TOMAZINI, M.; LOPES, L. F. B. *Web design responsivo–Bootstrap*. 2015.

Apêndice

A

Gráficos dos Resultados

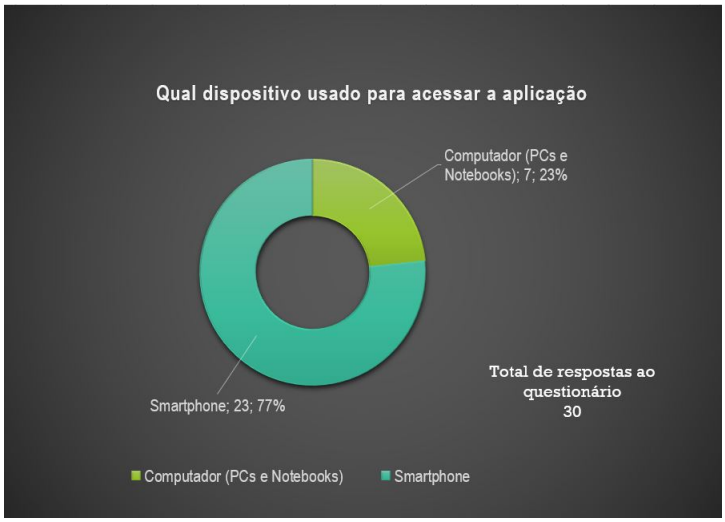

Figura A.1: Gráfico da pergunta 1
Fonte: Próprio autor

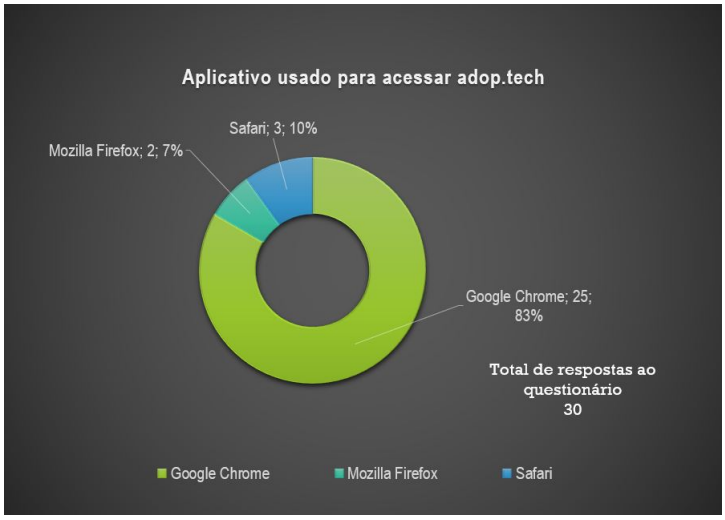

Figura A.2: Gráfico da pergunta 2
Fonte: Próprio autor

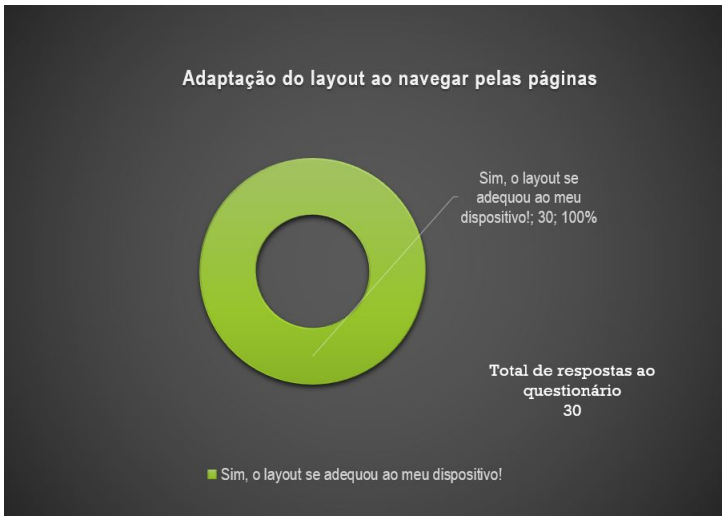

Figura A.3: Gráfico da pergunta 3
Fonte: Próprio autor

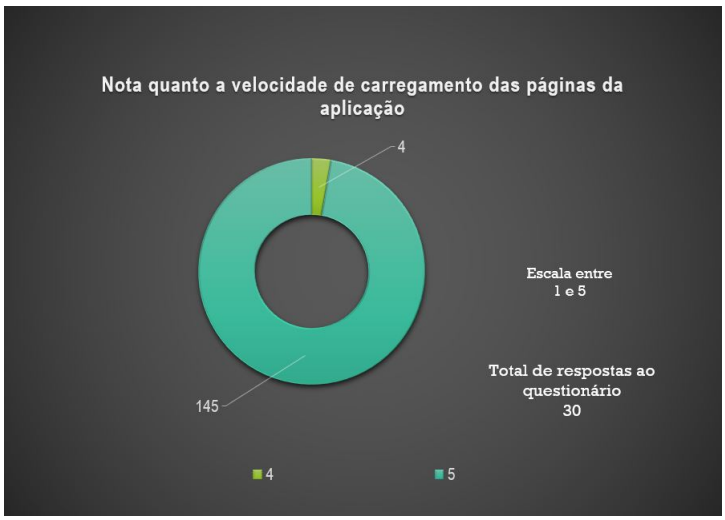

Figura A.4: Gráfico da pergunta 4
Fonte: Próprio autor

Figura A.5: Gráfico da pergunta 5
Fonte: Próprio autor

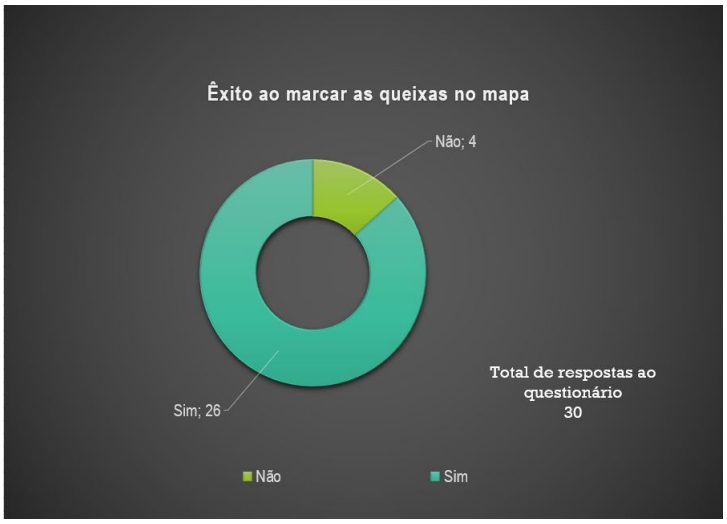

Figura A.6: Gráfico da pergunta 6
Fonte: Próprio autor

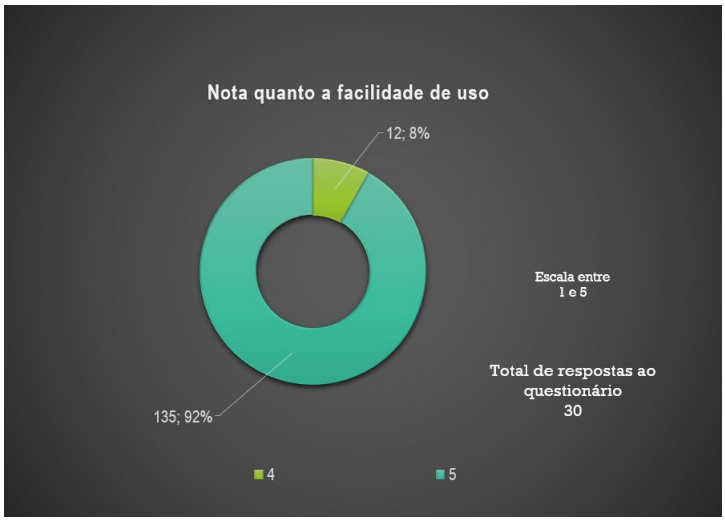

Figura A.7: Gráfico da pergunta 7
Fonte: Próprio autor