UNIVERSIDADE FEDERAL RURAL DE PERNAMBUCO

UNIDADE ACADÊMICA DE SERRA TALHADA

BACHARELADO EM SISTEMAS DE INFORMAÇÃO

INFORME.CITY: plataforma para informar e gerenciar problemas de infraestrutura urbana

Por

Natan de Souza Albuquerque

Serra Talhada,
Agosto/2018

UNIVERSIDADE FEDERAL RURAL DE PERNAMBUCO
UNIDADE ACADÊMICA DE SERRA TALHADA
CURSO DE BACHARELADO EM SISTEMAS DE INFORMAÇÃO

NATAN DE SOUZA ALBUQUERQUE

INFORME.CITY: plataforma para informar e gerenciar problemas de infraestrutura urbana

Trabalho de Conclusão de Curso apresentado ao Curso de Bacharelado em Sistemas de Informação da Unidade Acadêmica de Serra Talhada da Universidade Federal Rural de Pernambuco como requisito parcial à obtenção do grau de Bacharel.

Orientador: Dr. Richarlyson Alves D'Emery
Coorientador: Me. Glauber Magalhães Pires

Serra Talhada,
Agosto/2018

Dados Internacionais de Catalogação na Publicação (CIP)
Sistema Integrado de Bibliotecas da UFRPE
Biblioteca da UAST, Serra Talhada - PE, Brasil.

A345i Albuquerque, Natan de Souza

INFORME.CITY: plataforma para informar e gerenciar problemas de infraestrutura urbana / Natan de Souza Albuquerque. – Serra Talhada, 2018.

102 f.: il.

Orientador: Richarlyson Alves D'Emery
Coorientador: Glauber Magalhães Pires

Trabalho de Conclusão de Curso (Graduação em Bacharel em Sistema de Informação) – Universidade Federal Rural de Pernambuco. Unidade Acadêmica de Serra Talhada, 2018.

Inclui referências e apêndice.

1. Aplicações web. 2. Informática. 3. Software – Desenvolvimento. I. D'Emery, Richarlyson Alves, orient. II. Pires, Glauber Magalhães, coorient. III. Título.

CDD 004

UNIVERSIDADE FEDERAL RURAL DE PERNAMBUCO
UNIDADE ACADÊMICA DE SERRA TALHADA
BACHARELADO EM SISTEMAS DE INFORMAÇÃO

NATAN DE SOUZA ALBUQUERQUE

INFORME.CITY: plataforma para informar e gerenciar problemas de infraestrutura urbana

Trabalho de Conclusão de Curso julgado adequado para obtenção do título de Bacharel em Sistemas de Informação, defendida e aprovada por unanimidade em 05/09/2018 pela banca examinadora.

Banca Examinadora:

Prof. Dr. Richarlyson Alves D'Emery
Orientador
Universidade Federal Rural de Pernambuco

Prof. Me Hidelberg Oliveira Albuquerque
Universidade Federal Rural de Pernambuco

Prof. Me Héldon José Oliveira Albuquerque
Universidade Federal Rural de Pernambuco

A todos aqueles que de alguma forma estiveram e estão próximos de mim, em especial à minha família, professores, e amigos, sem os quais não teria alcançado essa etapa da minha formação.

AGRADECIMENTOS

A Deus por ter me dado saúde e a força para superar as dificuldades.

A univerdiade, em especial à coordenação do curso, pela chance extra que recebi para completar minha formação.

Ao meu orientador Richarlyson D'Emery e coorientador Glauber Pires, por todo o suporte e contribuições fundamentais para a realização desse trabalho.

A minha mãe, pelo amor, incentivo, e principalmente pela paciência que teve comigo em todos esses anos de minha formação.

Aos meus amigos e colegas por seus incentivos e companheirismo ao longo do curso, em especial a Wermeson, Jonhontham, Diogo, Murilo, e Arthur.

E a todos que direta ou indiretamente contribuíram para minha formação, o meu muito obrigado.

RESUMO

Introdução: Nos últimos anos houve um aumento significativo da participação popular na administração pública devido ao uso de tecnologias da informação e ao desenvolvimento de mecanismos de comunicação entre população e governo. Essas tecnologias aplicadas na gestão pública deu origem ao governo eletrônico. Com o surgimento de novos dispositivos como os *smartphones* e o crescente aumento de seus usuários, permitiu o surgimento de uma nova abordagem de administração pública participativa, a qual ficou conhecida como governo móvel. Dentre os focos da gestão urbana, têm-se os serviços de infraestrutura urbana, com sua natureza técnica caracterizada pelos sistemas e subsistemas técnicos. Esses sistemas estão sujeitos ao aparecimento de problemas que podem comprometer seu funcionamento, é de responsabilidade do governo municipal resolvê-los nos ambientes das cidades. A principal dificuldade do governo nesse processo é identificar o surgimento desses problemas. Uma abordagem para minimizar esse desafio é estimular a sociedade para que ela possa ajudar nesse processo, reportando os problemas encontrados, por fazerem parte do seu dia-a-dia. Essa abordagem vem sendo utilizada há muitos anos, porém através de meios limitados, como centrais telefônicas e formulários, que dificultam o processo para ambas as partes desse quadro. **Objetivo:** O objetivo desse trabalho foi desenvolver uma plataforma de *software* que facilite o processo de divulgação, monitoramento e controle de problemas de infraestrutura urbana. **Método:** Foi baseado em etapas comuns aos processos de desenvolvimento de software, como: levantamento de requisitos, planejamento dos casos de uso, modelagem dos componentes do sistema utilizando a *Unified Modeling Language*, implementação, e testes para validação. A plataforma foi desenvolvida na arquitetura cliente-servidor, através de um *Web service* no formato arquitetural *Representational State Transfer* (REST) e duas aplicações cliente. O *Web service* foi desenvolvido utilizando a linguagem de programação Go e o MongoDB como banco de dados. A primeira aplicação cliente, serviu para gerenciamento dos problemas informados na plataforma. Ela foi desenvolvida para ser uma aplicação de página única, utilizando-se a linguagem TypeScript com o *framework* Angular 6, utilizou a GoogleMaps JavaScript API v3 e foi estilizada com o *framework* CSS Bootstrap. A segunda aplicação cliente foi desenvolvida na linguagem TypeScript utilizando os *frameworks* Ionic e PhoneGap, foi desenvolvida para executar nas plataformas Andoid, iOS e Window Phone. Esse aplicativo fez uso da GoogleMaps JavaScript API v3. Ao término do desenvolvimento, as aplicações foram testadas. O *Web service* foi validado através de testes unitários caixa-branca e teste de desempenho de carga, caixa-preta, utilizando nesse último a ferramenta Vegeta. As duas aplicações cliente foram testadas por teste funcional baseado em casos de uso e casos de teste. Após execução desses testes as aplicações foram validadas por atenderem aos seus requisitos. **Conclusão:** Após o desenvolvimento e a validação dos sistemas, constatou-se que esses atenderam aos requisitos e objetivos levantados nesse trabalho. Desse forma, foi desenvolvida uma plataforma de *software* que facilita a população reportar problemas através de seus *smartphones* e que fornece ao governo uma ferramenta para gerenciar esses problemas.

Palavras-chave: Infraestrutura urbana, m-Gov, Aplicativo *Web*, REST, PhoneGap, GNSS, Go.

ABSTRACT

Introduction: In recent years there has been a significant increase in popular participation in public administration due to the use of information technology and the development of communication mechanisms between population and government. These technologies applied in public management gave rise to e-government. With the emergence of new devices such as smartphones and the increasing increase of its users, it allowed the emergence of a new approach to participatory public administration, which became known as mobile government. Among the focuses of urban management are urban infrastructure services, with their technical nature characterized by systems and technical subsystems. These systems are subject to the appearance of problems that can compromise its operation, it is the responsibility of the municipal government to solve them in the environments of the cities. The main difficulty of the government in this process is to identify the emergence of these problems. One approach to minimizing this challenge is to stimulate society so that it can help in this process by reporting the problems encountered because they are part of their day-to-day lives. This approach has been used for many years, but through limited means such as telephone exchanges and forms, which hinder the process for both parties. **Objective:** The objective of this work was to develop a software platform that facilitates the process of dissemination, monitoring and control of urban infrastructure problems. Method: It was based on steps common to software development processes, such as: requirements gathering, use case planning, modeling of system components using Unified Modeling Language, implementation, and validation tests. The platform was developed in the client-server architecture, through a Web service in the architectural form Representational State Transfer (REST) and two client applications. The Web service was developed using the Go programming language and MongoDB as the database. The first client application was used to manage the problems reported on the platform. It was developed to be a single-page application, using the TypeScript language with the Angular framework 6, used the GoogleMaps JavaScript API v3 and was styled with the CSS Bootstrap framework. The second client application was developed in the TypeScript language using the Ionic and PhoneGap frameworks, and was designed to run on the Andoid, iOS and Window Phone platforms. This application made use of the GoogleMaps JavaScript API v3. At the end of the development, the applications were tested. The Web service was validated through unit testing, white box and load performance test, black box, using the latter the Vegeta tool. The two client applications were tested by functional test based on use cases and test cases. After running these tests the applications were validated because they met your requirements. **Conclusion:** After the development and validation of the systems, it was verified that they met the requirements and objectives raised in this work. In this way, a software platform has been developed that facilitates the population to report problems through their smartphones and that provides the government with a tool to manage these problems.

Keywords: Urban Infrastructure, m-Gov, Web Application, REST, PhoneGap, GNSS, Go.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

API	*Application Programming Interface*
CSS	*Cascade Style Sheets*
e-Gov	Governo Eletrônico
m-Gov	Governo Móvel
FFI	Interface de Função Extrangeira
GLONASS	*Global'naya Navigatsionnaya Sputnikovaya Sistema*
GNSS	Sistemas de Navegação Global por Satélite
GPS	Sistema de Posicionamento Global
HTML	*Hypertext Markup Language*
HTTP	*Hypertext Transfer Protocol*
JSON	*JavaScript Object Notation*
MVC	*Model, View, and Control*
REST	*Representational State Transfer*
SGBD	Sistema Gerenciador de Banco de Dados
TI	Tecnologia da Informação
URI	*Uniform Resource Identifier*
XML	*Extensible Markup Language*
Web API	*Web Application Programming Interface*

SUMÁRIO

1 Introdução

Neste capítulo são apresentadas a contextualização, motivação e justificativa, bem como seus objetivos e método utilizado. Na Seção 1.1 expõe-se brevemente a contextualização do problema. Na Seção 1.2 discorre-se sobre a motivação e a justificativa para sua elaboração. Na Seção 1.3 descrevem-se os objetivos deste trabalho. A Seção 1.4 discorre sobre o método utilizado. Pro fim, na Seção 1.5, é fornecida uma visão geral dos capítulos que compõe este documento.

1.1 Contextualização e Problema

Nos últimos anos houve um aumento significativo da participação popular na administração pública, essa participação é legitimada por lei e os governos devem incentivá-la (SANABIO; SANTOS; DAVID, 2013). Um dos fatores que mais influenciam na participação popular é o desenvolvimento dos mecanismos de comunicação entre população e governo.

O uso de Tecnologia da Informação (TI) aplicada à administração pública trouxe melhorias como maior transparência de serviços públicos, ajuste de burocracias, diminuição de custos e melhorou, principalmente, a comunicação entre o cidadão e a administração pública. Essas tecnologias vêm sendo aplicadas com sucesso principalmente na gestão a nível municipal e foram enquadradas no que ficou conhecido como governo eletrônico (e-GOV) (FUNAI; REZENDE, 2012; SOUSA et al., 2015).

A Constituição Federal Brasileira de 1988 definiu que é responsabilidade da gestão de cada município brasileiro zelar pela sua infraestrutura urbana, garantindo sua operabilidade e a qualidade de vida dos seus cidadãos (BRASIL, 1988). Infraestrutura urbana pode ser compreendida como o conjunto de serviços básicos essenciais à atividade humana no ambiente das cidades, tais como iluminação pública, água, esgoto, transporte, entre outros (BERTEI et al., 2014). Os serviços de infraestrutura têm sua natureza técnica caracterizada pelos sistemas e subsistemas técnicos provenientes da engenharia urbana (MASCARÓ; YAOSHINAGA, 2005). Devido a essa natureza, é comum o surgimento de problemas nesses serviços, esses prejudicam, ou até mesmo comprometem, o seu funcionamento.

Dada a importância dessa infraestrutura para a vida do cidadão urbano, o governo municipal deve incluí-los como uma prioridade no seu plano de gestão. Algumas das principais funções da gestão pública são reconhecer, analisar e solucionar problemas (CONTABILIDADE, 2011). Essas três funções fundamentam as atividades de gestão municipal dos recursos de infraestrutura urbana.

A função de reconhecer problemas quando esses podem surgir em qualquer lugar no ambiente das cidades é um grande desafio para os governos municipais, agravando-se com o crescimento das zonas urbanas ao longo do tempo (URBANO; HORIZONTE, 2002).

O processo comumente adotado para esse fim, consiste em oferecer mecanismos de denúncia aos cidadãos, geralmente através de telefonia, para contatarem o governo municipal e reportarem o problema identificado. Outro modo é através de fiscalização de toda a extensão territorial urbana, processo que é dispendioso e que se agrava proporcionalmente ao tamanho dessa área. A partir das informações recolhidas dessas atividades, o governo municipal pode analisar os problemas identificados e em seguida planejar e executar soluções para os mesmos.

Utilizar ligações telefônicas e fiscalização as cegas como meio para informar problemas de infraestrutura urbana é um processo que perdurou ao longo dos anos. Mas, com o surgimento de novas mídias e meios de comunicação mais fáceis e acessíveis à população, essa forma de identificar problemas tornou-se ineficiente e dispendiosa. Isso fica mais evidente quando é analisado o modo como os antigos processo foram implementados. Em regras gerais, através do atendimento ao público nas centrais telefônicas municipais e ouvidorias, onde evidencia-se dificuldade de prestar um bom atendimento ao público (CARRIJO; ALVARENGA, 2011) e o baixo grau de qualidade das informações levantadas, que são difíceis de serem auditadas e acompanhas pela população ao longo dos demais processos.

1.2 Motivação e Justificativa

Frente as dificuldades apontadas, em específico as presentes nos processos de identificação, monitoramento e controle de problemas de infraestrutura urbana; é importante levantar uma solução que supere essas dificuldades. Solução essa que possa oferecer maior facilidade, eficiência e transparência nesses processos.

O surgimento e a expansão de novas tecnologias que demostram viabilidade para superar tais dificuldades são fatores motivantes para propor uma solução baseada nessas tecnologias.

Assim, este trabalho verifica o desenvolvimento de uma solução de *software* que supra tais necessidades.

Para isso, optou-se pelo desenvolvimento de um sistema embasado em soluções tecnológicas existentes para os seguintes campos: Sistema de Navegação por Satélite (GNSS, do inglês *Global Navigation Satellite System*), computação móvel em dispositivos *smartphones* e arquitetura cliente-servidor com base em *Web*/*Hypertext Transfer Protocol* (HTTP).

É justificável utilizar soluções nessas áreas devido à capacidade que elas têm demonstrado para cumprirem os requisitos levantados pela natureza dos problemas de infraestrutura urbana. Utilizar sistemas de geolocalização, através de *GNSS*, para demarcar problemas de infraestrutura tem se apresentado, não somente como uma ótima solução para localizá-los e monitorá-los, mas também como uma ótima ferramenta de gestão pública (URBANO; HORIZONTE, 2002). Essa abordagem vem sendo utilizada com sucesso em aplicações práticas para outras áreas desse tipo de gestão, como a área da saúde, a exemplo do trabalho de Barros et al. (2014), que realizaram o monitoramento e controle de focos de dengue no território urbano.

O uso da computação móvel através de *smartphone* é uma boa abordagem quando pretende-se estimular o cidadão a informar os problemas de infraestrutura no local em que esses são identificados. Lanza (2011) levanta os benefícios da utilização dessa abordagem para a administração pública municipal no que tange à participação popular no governo e classifica a utilização desses dispositivos como uma nova forma de governança eletrônica, complementar ao e-GOV, anunciada como Governo Móvel (m-GOV).

Os *smartphones* são dispositivos telefônicos com capacidades muito além dos telefones celulares clássicos, que oferecem maior poder computacional e agregam vários outros serviços, como, por exemplo, GNSS, capacidade de capturar imagens, diversas formas de conexão com a *Internet*, entre outras (CHMIELARZ, 2015; ZHONG, 2013). Além disso, no Brasil, há uma ampla utilização desses dispositivos por parte da população, em 2015 o uso de *smartphones* conectados à *Internet* superou o de computadores (MEIRELLES, 2015).

Com o desenvolvimento de tecnologias *Web*, surgiram novas formas de disponibilizar serviços de software através da *Internet* para acesso remoto em diversas plataformas e dispositivos, como *smartphones*, por exemplo. Essas novas soluções são resumidas nos *Web services* e *Web Application Programming Interface* (Web API), que são soluções para oferecer interoperabilidade entre sistemas distribuídos através de conexões de rede utilizando HTTP e padrões estabelecidos para plataforma *Web* (KUROSE; ROSS, 2010; TIDWELL; SNELL; KULCHENKO, 2001).

1.3 Objetivos

1.3.1 Geral

Desenvolver uma plataforma de *software* que permita a divulgação, monitoramento e controle de problemas de infraestrutura urbana.

1.3.2 Específicos

Para alcançar este objetivo, têm-se os seguintes objetivos específicos:

- Identificar, classificar e delimitar quais os tipos de problemas de infraestrutura urbana que serão abordados pela plataforma;
- Prover um mecanismo para a população poder informar problemas utilizando uma aplicação *Web* ou *mobile*;
- Desenvolver uma ferramenta que permita ao governo municipal visualizar e gerenciar os problemas informados;
- Desenvolver uma API *Web* que permita a interoperabilidade e integração do sistema proposto a outros sistemas.

1.4 Método

O método adotado nesse projeto consistiu em dividir em etapas os processos de projeto e desenvolvimento de *software*, baseando-se em Pressman (2006). Essas etapas foram organizadas de forma sequencial, de maneira que cada etapa serviu como base para a seguinte. A seguir, são listadas essas etapas na ordem estabelecida para seu desenvolvimento:

A **primeira etapa**, consistiu no levantamento de referências bibliografias e **trabalhos relacionados**, dos quais foram analisados para identificar características que pudessem ser utilizadas conforme o propósito deste trabalho.

Os trabalhos relacionados foram encontrados através de pesquisas na *Internet* com mecanismos de buscas com base na similaridade temática, levou-se ainda em consideração

características como: abordar área de problemas de infraestrutura urbana ou similares, utilizar soluções de *software* baseadas em aplicações para *Web* ou dispositivos móveis e utilizar ferramentas e tecnologias semelhantes às que foram empregadas para o desenvolvimento da solução apresentada nesse trabalho.

A**segunda etapa** tratou de definir quais os **tipos de problemas de infraestrutura urbana** que o sistema daria suporte. Também foram analisados problemas abordados nos trabalhos relacionados levantados na primeira etapa.

A **terceira etapa** consistiu no levantamento e formalização dos **requisitos funcionais** e **não funcionais**, dos principais **casos de uso** do sistema. Foram definidas quais abordagens e tipos de testes seriam utilizados a cada componente do sistema.

A partir dos requisitos funcionais, foram elaborados os diagramas de caso de uso em *Unified Modeling Language* (UML). Também foram elaborados **casos de teste**, que formaram o mecanismo de validação utilizado para determinar se o sistema conseguiu cumprir os requisitos propostos. Utilizou-se duas abordagens de teste, a abordagem de caixa-preta e a abordagem de caixa-branca. Essa última foi utilizada para a aplicação servidora que compreende ao *Web service*. Para ele também foi elaborado um teste não funcional de carga. Os casos de teste foram derivados dos casos de uso, esses últimos serviram para definir como a solução se comportaria em cenários de interação com seus atores.

Na **quarta etapa** está a modelagem das classes de domínio, através do uso da *Unified Modeling Language* (UML), utilizando a Orientação à Objetos (OO). Também nessa etapa foi definida a arquitetura cliente-servidor da plataforma, levando em conta os seus subsistemas, que correspondem à aplicação de servidor (*back-end*) e duas aplicações clientes (*front-end*).

A **quinta etapa** consistiu na implementação do *Web service*. Abrangeu a escolha das tecnologias utilizadas, como: linguagem de programação, *frameworks*, Sistema Gerenciador de Banco de Dados (SGBD) e protocolos de segurança. Definiu-se a API de serviço para o formato *Representational State Transfer* (REST) com base em HTTP, tomando como base a modelagem estabelecida na quarta etapa.

Foi utilizada a linguagem de programação Go, versão 1.10.3, com uso do *framework* Gin, versão 1. Go foi escolhida por permitir desenvolvimento para aplicações *Web* com alto desempenho em velocidade e em relação ao volume de processos simultâneos, conforme Valkov, Chechina e Trinder (2018). Foi utilizado como banco de dados o MongoDB por oferecer capacidades de escalabilidade, mecanismo para armazenamento de arquivos chamado *Mongo File System* (MongoFS), e implementar armazenamento indexado e operadores de consulta para

dados de geolocalização nativamente.

A **sexta etapa** compreendeu na implementação de duas aplicações clientes. Apesar disso, houve variação das plataformas destino para as aplicações clientes. O sistema para gerenciamento dos problemas informados (utilizado pelo governo e órgão responsáveis) e o sistema para reportar problemas (utilizado pela população).

A primeira aplicação cliente desenvolvida foi a aplicação para administração dos problemas relatados. Esse cliente foi desenvolvido nas linguagens *Hypertext Markup Language* (HTML), *Cascade Style Sheets* (CSS), TypeScript e JavaScript. Foi utilizado o *framework* JavaScript, Angular 6. Esse *framework* utiliza o padrão *Model, View, and Control* (MVC) no lado do cliente (navegador *Web*), para facilitar a construção e manutibilidade da aplicação. A aplicação possui integração com o Google Maps, foram utilizadas a Google Maps API JavaScript, versão 3, para acesso aos serviços de Mapas Dinâmicos e de Rotas. Também foi utilizada a API Google Charts para a apresentação de dados utilizando gráficos.

A segunda aplicação cliente ficou responsável por informar os problemas encontrados pela população. Ela foi desenvolvida utilizando-se as mesmas linguagens e tecnologias da aplicação de administração, além dessas tecnologias foram utilizados os *frameworks* Ionic e PhoneGap para permitir a criação de executáveis em diversas plataformas de dispositivos móveis de maneira integrada a essas plataformas. Essa aplicação cliente utilizou os serviços de GNSS e da câmera dos dispositivos móveis, para aquisição de dados sobre localização e imagens, respectivamente.

Na **sétima e última etapa**, foram aplicados testes funcionais e um caso de teste não funcional nos sistemas desenvolvidos, esses casos de teste foram os levantados na terceira etapa. Cada caso de uso relevante teve um respectivo caso de teste que serviu para validação dos sistemas.

1.5 Organização do Trabalho

As demais partes desse trabalho estão organizadas da seguinte forma:

- No Capítulo 2 apresenta-se o referencial teórico desse trabalho, explanam-se os principais assuntos relacionados a temática e as tecnologias utilizadas na elaboração desse projeto. Explanam-se sobre infraestrutura urbana, gestão urbana, Sistemas

de Navegação Global por Satélites, Google Maps, PhoneGap, plataforma *Web* e arquitetura REST e os trabalhos relacionados.

- O Capítulo 4 discorre sobre a plataforma desenvolvida, abrangendo as etapas estabelecidas no método, da concepção da plataforma até a sua validação.
- O Capítulo 5 abrange a conclusão e as considerações finais desse trabalho.
- Por fim, têm-se as Referências, seguidas de Apêndices que complementam a monografia.

2 Fundamentação Teórica

Neste capítulo é apresentada uma breve explanação sobre os principais assuntos e tecnologias relacionados a temática e ao desenvolvimento do projeto de software. Na Seção 2.1 explana-se Infraestrutura Urbana. A Seção 2.2 descreve brevemente a Gestão Urbana e as práticas de utilização de TI nesse tipo de gestão. Na Seção 2.3 têm-se os Sistemas de Navegação Global por Satélite, com ênfase nos sistemas GPS e GLONASS. A Seção 2.4 descreve o serviço de mapas na Web Google Maps. Na Seção 2.5 descreve-se o framework para construção de aplicativos móveis chamado PhoneGap. Por fim, a Seção 2.6 oferece um resumo da plataforma Web, sua arquitetura e linguagens, além de explanar brevemente o modelo arquitetural para construção de Web Service denominado REST.

2.1 Infraestrutura Urbana

A infraestrutura urbana pode ser compreendida como o conjunto de serviços básicos essenciais para à atividade humana no ambiente das cidades, tais como iluminação pública, água, esgoto, transporte, entre outros (BERTEI et al., 2014). O termo "serviço de infraestrutura" advém dos resultados finais obtidos através do uso dos sistemas técnicos de engenharia urbana, que são de natureza física e dão suporte à essas características de serviços. Dessa forma, pretende-se integrar ao conceito de sistema técnico: sua função dentro do meio urbano, através do serviço prestado à população, e seus equipamentos de rede física (MASCARÓ; YAOSHINAGA, 2005). Esses sistemas de infraestrutura são frutos da Engenharia e têm sua origem na antiguidade junto ao surgimento dos primeiros núcleos urbanos, a exemplo do sistema viário do império romano.

Devida a sua natureza física, esses sistemas estão sujeitos a desgastes, advindos de fatores físicos, químicos e defeitos de projeto ou construção, levando ao surgimento de problemas que podem prejudicar, ou até comprometer, o seu funcionamento. Dessa forma, é importante a rápida identificação e solução desses problemas para que eles não cheguem a comprometerem o funcionamento do sistema e os seus serviços prestados.

Mascaró e Yaoshinaga (2005) levantam alguns exemplos de subsistemas de infraestru-

tura urbana: subsistema viário (vias urbanas); subsistema de drenagem pluvial; subsistema de abastecimento de água; subsistema de esgotos sanitários; subsistema energético e subsistema de comunicação. Todos esses interligam-se funcionalmente e formam a infraestrutura urbana.

2.2 Gestão Urbana

O processo de gestão, do ponto de vista da administração, está relacionado com a utilização de recursos e a aplicação de atividades com o objetivo de utilizá-los eficientemente. Dessa forma, o processo de gestão é uma função básica da administração que estabelece a importância e a forma como serão desenvolvidos os demais processos, ela é o ato de fazer administração nas organizações (CHIAVENATO, 2014).

Diante disso, Rezende e Frey (2005) apontam que o processo de gestão é importante na administração pública das cidades, devida a natureza de que esse tipo de administração baseia-se em estipular metas e utilizar recursos de forma eficiente e em setores estratégicos para alcançá-las, orientadas para o benefício da sociedade.

Ainda segundo os autores,

> A cidade é um organismo dinâmico e complexo. Esse organismo pode ser caracterizado por grandes diversidades e múltiplos contrastes, gerando inúmeras dificuldades ao gestor público. Nesse sentido a gestão urbana deve desempenhar um papel relevante para contribuir na diminuição desses contrastes, dificuldades e conflitos e também na solução dos múltiplos problemas enfrentados. (REZENDE; FREY, 2005, p. 53).

Nesse contexto, a gestão urbana tem ênfase na utilização dos processos e práticas, advindos da administração pública, aplicados na organização das cidades com o intuito de melhorar a qualidade de vida dos cidadãos. Sob o ponto de vista prático, esses processos baseiam-se em investimentos e melhorias na infraestrutura de serviços urbanos.

2.2.1 Uso de Tecnologia de Informação na Gestão Urbana

Com o aumento do uso de TI em organizações, no fim da década de 70 houve um grande movimento mundial dentro dos governos para realizar a reforma do aparelho do estado, com o objetivo de trazer mais eficiência para burocracias estatais, bem como a prestação de serviços sem a necessidade de contato presencial (LIMA; RIBEIRO, 2012).

Nesse cenário, surgiu o e-Gov que tem como objetivo utilizar novas tecnologias de informação e comunicação aplicadas em uma ampla quantidade de funções do governo, em especial, na comunicação deste para com a sociedade. As principais relações sustentadas pelo governo eletrônico são: aplicações *Web* com foco para o segmento governo-negócio (G2B); aplicações *Web* voltadas para a relação governo-cidadão (G2C); e aplicações *Web* referentes à estratégias governo-governo (G2G) (REZENDE; FREY; BETINI, 2005). Para Rezende, Frey e Betini (2005), o e-Gov é como uma oportunidade de se repensar a forma como governos prestam serviços aos cidadãos, atendendo as demandas e necessidades dos usuários de informação governamental.

Além de possibilitar o surgimento do e-Gov, a expansão da Internet também acompanhou o aumento do uso de celulares no Brasil, que em 2011 ultrapassou a quantidade de habitantes e, em 2015, passou a corresponder a grande maioria dos dispositivos conectados a essa rede. Esse fenômeno fez com que governos começassem a pensar em prestar esses serviços públicos a partir destes dispositivos, dando origem ao Governo Móvel (m-Gov) (LIMA; RIBEIRO, 2012; MEIRELLES, 2015).

Lanza (2011) refere-se ao m-Gov como uma estratégia de disponibilização de serviços públicos através de plataformas móveis para os cidadãos e à sociedade, dessa maneira permite que esses acessem informações públicas e interajam com esses serviços em qualquer momento e lugar.

O m-Gov surge para facilitar a interação entre os cidadãos e os órgãos públicos de maneira mais ágil, possibilitando que essa comunicação ocorra independente da localização ou tempo, além disso, permite encontrar soluções mais rápidas para problemas que dependam da participação popular para serem resolvidos.

2.3 Sistemas de Navegação Global por Satélite

Sistemas de Navegação Global por Satélite (GNSS, do inglês *Global Navigation Satellite Systems*) são sistemas projetados para determinar a localização de um determinado ponto na superfície da Terra. Eles são fundamentados no uso de satélites artificiais, cuja localização são conhecidas ao longo do tempo e servem de base para calcular, através de algoritmos de triangulação, a localização de um determinado ponto num espaço tridimensional (HOFMANN-WELLENHOF; LICHTENEGGER; WASLE, 2007). Essa localização é representada cartograficamente através

da latitude, longitude, e altitude de um ponto. Hofmann-Wellenhof, Lichtenegger e Wasle (2007) explanam os principais sistemas modernos de navegação global baseados em satélites artificiais: Sistema de Posicionamento Global (GPS, do inglês *Global Positioning System*) dos Estados Unidos da América, *Global Navigation Satellite System* (GLONASS, do russo *Global'naya Navigatsionnaya Sputnikovaya Sistema*) da Rússia e *Galileo* da União Europeia.

Os sistemas GPS e GLONASS são os mais utilizados atualmente, pois cobrem tanto o segmento civil quanto o militar e contam com suas constelações de satélites completas (SIMONSEN; PRUDENTE, 2015). Esses dois sistemas vêm sendo utilizados em conjunto em muitos dos receptores do segmento de usuário, pois melhoram a precisão da localização devido ao aumento do número de satélites disponíveis quando somadas suas constelações (HOFMANN-WELLENHOF; LICHTENEGGER; WASLE, 2007). As subseções a seguir explanam com maiores detalhes os sistemas GPS e GLONASS.

2.3.1 *Global Positioning System*

GPS é composto por uma constelação de vinte e quatro satélites artificias que orbitam em torno do planeta Terra a aproximadamente 20.200 km acima do nível do mar. Esses satélites permitem que receptores conheçam sua posição em qualquer lugar sobre a Terra com uma notável precisão (PEREIRA, 2014).

Seu desenvolvimento teve início em 1973 pelo Departamento de Defesa dos Estados Unidos, esse projeto tinha como objetivo construir um sistema que permitisse localizar com exatidão navios, aeronaves, tropas militares terrestres e ajudasse no lançamento de mísseis. O projeto recebeu o nome de *NAVSTAR – Navigation Satellite Timing and Ranging*, nome oficial dado ao GPS pelo Departamento de Defesa dos Estados Unidos.

Dentre as principais características do GPS, destacam-se: a alta precisão de posição para usuários militares, boa precisão para usuários civis, independência de fatores climáticos, fácil acesso através de equipamentos simples. Além disso, o GPS utiliza uma técnica de navegação que é feita através da medição das distâncias entre os satélites.

Segundo Baroni (2006, p. 39),

> As distâncias com relação a cada satélite são medidas fazendo-se a correlação entre o sinal recebido e uma réplica gerada pelo usuário. O usuário é capaz de diferenciar cada satélite através da porção do código PRN (*Pseudorandom noise*) que cada satélite transmite. Medindo a distancia com relação a quatro

> satélites o usuário pode determinar as quatro incógnitas: três coordenadas de posição e a correção no relógio.

O GPS é estruturado em três segmentos, são eles: segmento espacial, segmento de controle e segmento de usuário.

- Seguimento Espacial - É formado por uma constelação com 24 (vinte e quatro) satélites que giram em torno da Terra, em seis orbitas diferentes, com quatro satélites por órbita. Os satélites percorrem a órbita a cada 12 horas e estão espaçados de forma que a área de visão de um receptor GPS esteja coberta por pelo menos seis satélites, isso é mais que suficiente para determinação de uma posição, uma vez que apenas quatro satélites são necessários para isso. "Cada satélite possui painéis solares para fornecimento de energia, quatro relógios atômicos, equipamentos de computação e comunicação e baterias recarregáveis para os períodos de eclipse" (TOLENTINO, 2003, p. 79). Com isso eles são capazes de transmitir sinais de rádio constantemente chegando a um receptor GPS em centésimos de segundo.
- Seguimento de Controle - Todos os satélites são controlados e monitorados por estações espalhadas pela Terra, a principal delas é a estação mestre de controle - MCS, que é auxiliada por outras cinco estações. Essas estações além de garantir o funcionamento dos satélites realizando atualizações de informações regularmente, também pode fazer modificações nos mesmos.
- Seguimento de Usuário - É formado pelos receptores GPS, que pode variar de tamanho, fabricante e qualidade de recepção e tem a capacidade de converter sinais de rádio vindos dos satélites para determinar a posição geográfica de um ponto na Terra, de forma a orientar a navegação aérea, terrestre ou náutica (VICENTE; BERNARDI, 2002).

  Para Baroni (2006, p. 39),

  > O segmento usuário consiste basicamente de receptores militares e civis projetados especialmente para decodificar e processar os sinais que recebem dos satélites. O receptor computadorizado grava as transmissões de vários satélites e aplica algoritmos de solução para obtenção de posição, velocidade e tempo (solução de navegação).

  Os receptores GPS podem ser bem simples e portáteis, por isso estão presentes em *smartphones*, *tablets*, carros, entre outros. Segundo Tolentino (2003), além

de decodificar os sinais dos satélites, os receptores também permitem (devido a microprocessadores, memória e *software* internos), utilizar sistemas de medidas, sistemas de coordenadas, armazenamento de dados, troca de dados com outros receptores ou com um computador, etc.

2.3.2 *Global Navigation Satellite System*

O GLONASS começou a ser desenvolvido pela antiga União Soviética, em meados de 1970. Com o fim da União Soviética o projeto foi continuado pela Rússia. Assim como o GPS, o GLONASS foi projetado com propósitos militares e também civis. Seu primeiro satélite foi lançado em 1982 e teve sua constelação operacional de satélites atingida pela primeira vez em 1995. Porém, com o fim da União Soviética, o projeto ficou em esquecimento e decaiu ao longo dos anos até chegar, no ano de 2001, a uma constelação de somente seis satélites. Nesse período teve-se início o projeto de modernização do sistema, responsável pela reconstrução da constelação de satélites, introdução de novos sinais e atualizações no segmento de controle. Sua atualização atingiu a operacionalidade global no ano de 2011, onde sua constelação alcançou os 24 (vinte e quatro) satélites (GROVES, 2013).

Assim como o GPS, o GLONASS é composto por três seguimentos, o segmento espacial, seguimento de controle e segmento de usuário.

- Segmento Espacial - O segmento espacial é formado por 24 satélites que orbitam a Terra ocupando três planos orbitais, com altitude de raio de aproximadamente 25.500 km. Ao longo de seu desenvolvimento foram lançados vários modelos de satélites, sendo eles GLONASS-M, GLONASS-K1, GLONASS-K2 (em planejamento) (GROVES, 2013).
- Segmento de Controle - O segmento de controle é composto pelas estações terrestres que controlam e monitoram os satélites, corrigindo as órbitas e os relógios de cada satélite GLONASS. Esse segmento está diretamente subordinado à Força Espacial Russa.
- Segmento de Usuário - O segmento de usuário corresponde aos dispositivos terrestres capazes de receber e interpretar o sinal GLONASS. Para o cálculo de uma posição, uma receptor GLONASS precisa receber o sinal de quatro satélites, três para determinação das coordenadas e um para determinação do tempo (GROVES, 2013; HOFMANN-

WELLENHOF; LICHTENEGGER; WASLE, 2007).

2.4 Google Maps

O Google Maps é um serviço de visualização de mapas através da *Web* fornecido pela empresa Google Inc. Foi criado por Lars Rasmussen e Jens Rasmussen na companhia Where 2 Technologies, adquirida em outubro de 2004 pela Google Inc. Após essa aquisição, a Google lançou esse serviço com o nome de Google Maps em 07 de fevereiro de 2005 (SVENNERBERG, 2010).

A Google oferece uma API do Google Maps para as plataformas: *Web*, Android, e iOS; permitindo que desenvolvedores de *software* possam agregar esse serviços às suas aplicações. A plataforma *Web* foi a primeira plataforma e é a que mais utiliza esse serviço, sua API é fornecida na linguagem de programação JavaScript e em 2015 foi lançada a versão 3 (DINCER; URAZ, 2013; GOOGLE, 2018; SVENNERBERG, 2010).

As principais funcionalidades do Google Maps são fornecer mapas interativos, com possibilidade de mudança de escala (*zoom*), mudança de estilos de mapa, permite sobreposição de camadas de informações (imagens de satélite, mapa de elevação, camadas de imagens), movimentação livre através de interação do usuário e criação de marcadores de localização customizáveis. Todas essas opções podem ser manipuladas e reconfiguradas através da API.

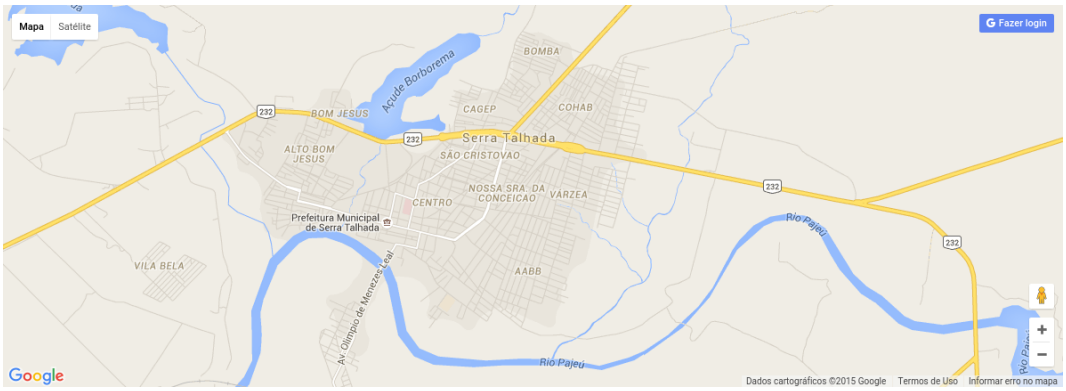
A Figura 2.1, demostra um exemplo de mapa do Google Maps, esse tipo de mapa costuma ser embutido em páginas *Web* e permite interatividade com o usuário.

Através da Google Maps JavaScript API v.3 é possível que desenvolvedores de *software* possam criar mapas com propriedades customizadas, e manipular essas propriedades através de código JavaScript. Um dos principais recursos existentes na API é a possibilidade de se demarcar pontos no mapa.

A Figura 2.2 demostra o uso da API JavaScript do Google Maps, esse trecho de código, na linguagem JavaScript, cria um mapa centralizado na coordenada geográfica de latitude -34.397 e longitude 150.644 no padrão WGS841[1] e *zoom* de nível 8.

[1] *World Geodetic System* (WGS) é uma norma para cartografia de origem geocêntrica definida em 1984, é utilizada por GNSS, incluindo o GPS

Figura 2.1 – Exemplo de mapa do Google Maps no estilo padrão



Fonte: Elaborada pelo autor

Figura 2.2 – Trecho de código demostrando o uso da Google Maps JavaScript API v.3

```
var map;

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

Fonte: Elaborada pelo autor

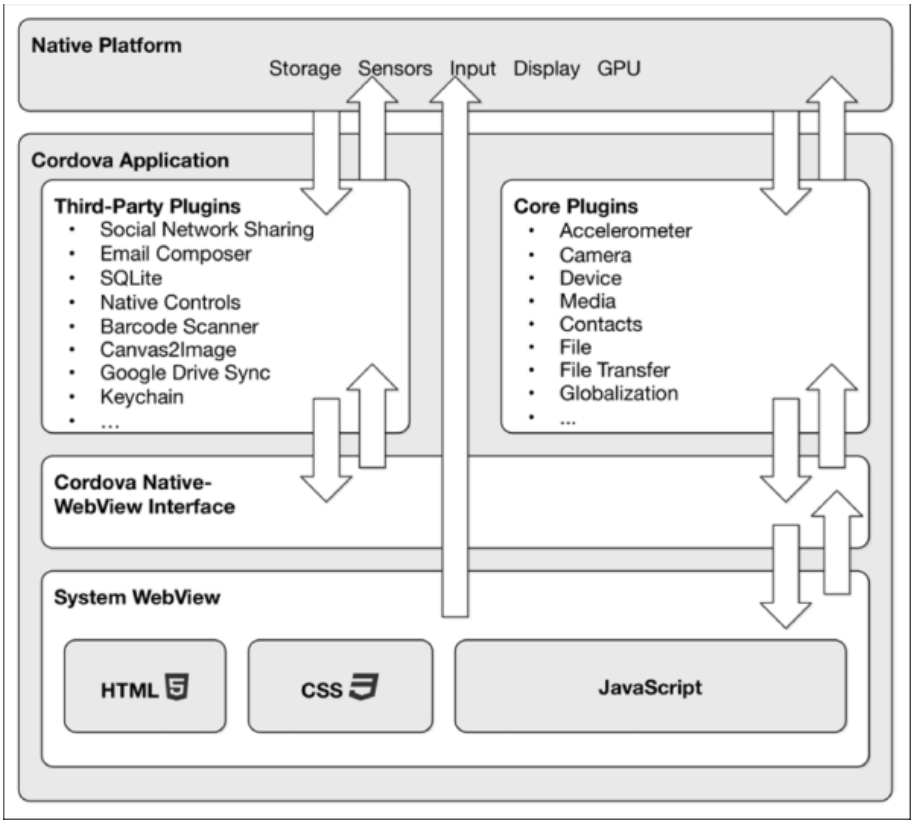
2.5 PhoneGap

PhoneGap é um projeto de código livre usado para construir aplicativos móveis. Ele utiliza código JavaScript, HTML e CSS, os empacota em um programa executável capaz de rodar em diversas plataformas de dispositivos móveis. Esse *framework* permite o acesso a recursos nativos do dispositivo, como GNSS e câmera, através do uso de Interface de Função Estrangeira (FFI, do inglês *Foreign Function Interface*). Essa interface permite chamar código nativo da plataforma utilizando a linguagem de programação JavaScript. Além disso PhoneGap permite a construção de aplicações nativas através do seu sistema de *plugins*. Esse sistema permite que desenvolvedores adicionem recursos personalizados às suas aplicações. Muitos *plugins* são disponibilizados e mantidos pelo time de desenvolvedores do PhoneGap em conjunto com a comunidade do código livre (MYER, 2012).

O projeto do PhoneGap nasceu em um *hackathon* (maratona de programação) no ano de 2008 na conferência iPhoneDevCamp, mais tarde renomeada para iOSDevCamp. Nos anos seguintes a empresa Nitobi completou o desenvolvimento desse *software*. No ano de 2011, a empresa Nitobi foi adquirida pela empresa Adobe, após a aquisição a Adobe doou parte do

código fonte do projeto para *Apache Software Foundation* (ASF). Desde então essa fundação é responsável pelo desenvolvimento e manutenção dos componentes núcleo do PhoneGap, permitindo que esses componentes permaneçam sob código livre. Esse projeto é nomeado de Apache Cordova. Dessa maneira, o PhoneGap é o projeto Apache Cordova acrescido de alguns serviços e extensões da Adobe que melhoram ainda mais as suas capacidades (FERNANDEZ; ALBER, 2015).

Figura 2.3 – Visão geral da arquitetura de uma aplicação do PhoneGap/Apache Cordova



Fonte: Shotts (2014, p. 13)

O PhoneGap utiliza da característica de que a plataforma *Web* é independente e permite que essa independência vá a um nível além do natural, ele permite que a programação para *Web* possa interagir com os serviços do dispositivo móvel, graças ao seu sistema de *plugins* e FFI.

A Figura 2.3 mostra como estão organizados os componentes que compõe a arquitetura de uma aplicação PhoneGap. A aplicação é construída com as linguagens JavaScript, HML e CSS e comunica-se com serviços nativos da plataforma de dispositivo móvel através de *plugins*, e das interfaces fornecidas entre essas camadas. Essas interfaces são oferecidas pelo núcleo Cordova, através de uma camada de código nativo de cada plataforma, que se comunica com a *WebView* utilizando o mecanismo de JavaScript que ela fornece. Assim a aplicação desenvolvida executa na *WebView* e pode acessar através do JavaScript essa camada de código nativo por uma

interface interoperável.

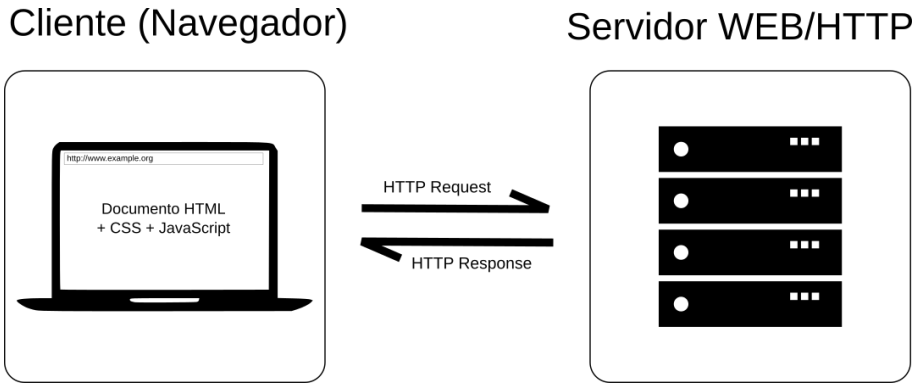
2.6 Plataforma *Web* e REST

A *World Wide Web* (WEB) ou somente *Web* surgiu em 1989 no Centro Europeu para Pesquisa Nuclear (CERN), foi projetada pelo físico britânico Tim Berners-Lee (BERNERS-LEE, 1989). Inicialmente como uma plataforma para compartilhamento de documentos através do seu próprio protocolo da camada de aplicação HTTP. Em 1994, o CERN e o Instituto de Tecnologia de Massachusetts (MIT) assinaram um acordo criando o World Wide Web Consortium (W3C), essa organização é direcionada para o desenvolvimento da *Web*, e tem a responsabilidade de criar os padrões e especificações de protocolos e linguagens dessa plataforma (KUROSE; ROSS, 2010).

Nessa plataforma, os documentos de hipertexto são visualizados através do aplicativo cliente chamado de navegador *Web*, o navegador é responsável por se conectar aos servidores *Web* e, através do protocolo HTTP, baixar os documentos e recursos, interpretá-los e exibi-los ao usuário. Esse processo é feito com base na arquitetura cliente-servidor, sendo assim, o navegador *Web* corresponde ao cliente e o servidor *Web* ao servidor (TANENBAUM, 2003).

A Figura 2.4 mostra essa distribuição dos componentes em cliente e servidor. Os documentos de hipertexto da *Web* são escritos na linguagem de marcação HTML, através do uso de CSS pode-se aplicar estilos personalizados para exibição desses documentos. Os navegadores também suportam a linguagem de programação JavaScript, utilizada para construção de pequenos programas que executam no navegador, geralmente acrescentando mais funcionalidades na interação entre usuário e página *Web* (documento em hipertexto) (TANENBAUM, 2003).

Figura 2.4 – Arquitetura da *Web*, modelo cliente servidor utilizando o protocolo HTTP



Fonte: Elaborada pelo autor

O servidor *Web* HTTP é responsável por receber as requisições do cliente (navegador *Web*), processá-las e retornar o conteúdo solicitado novamente ao cliente através da resposta HTTP. Dessa maneira, o servidor HTTP pode executar aplicações que tratem uma requisição e com base nela possa realizar determinados serviços, como consultas a base de dados e montar uma resposta específica para a solicitação, isso caracteriza o que se conhece como *Web Service* (TIDWELL; SNELL; KULCHENKO, 2001).

Baseado no modelo arquitetural cliente-servidor através de HTTP, Fielding (2000) estabelece uma arquitetura para construção de sistemas de hipermídia, esse estilo de arquitetura recebe o nome de *Representational State Transfer* (REST). A Seção 2.6.1 descreve esse estilo arquitetural.

2.6.1 *Representational State Transfer*

Transferência de Estado Representacional é uma abstração da arquitetura da *Web*, consiste em um conjunto coordenado de restrições arquiteturais aplicadas a componentes, conectores e elementos de dados dentro de um sistema distribuído de hipermídia. Esse estilo ignora os detalhes da implementação dos componentes e a sintaxe do protocolo HTTP, objetivando focar somente os papéis desses componentes. Esse estilo arquitetural pode ser aplicado no desenvolvimento de *Web Service*. Baseia-se nos seguintes princípios para implementação, a luz de Richardson e Ruby (2007):

- Protocolo cliente-servidor sem estado - cada mensagem HTTP contém toda a informação necessária para compreender o pedido. Dessa forma, nem o cliente e nem o servidor precisam guardar os estados das comunicações entre essas mensagens. Muitas aplicações baseadas em HTTP utilizam *cookies* e outros mecanismos para manter o estado da sessão;
- Conjunto de operações bem determinadas que se aplicam a todos os recursos de informação - o HTTP já define algumas operações, como POST, GET, PUT e DELETE. Geralmente essas operações são combinadas com operações do tipo CRUD (do inglês, *Create, Read, Update, and Delete*) para a persistência de dados;
- Sintaxe universal para identificar cada recursos. Assim, cada recurso é unicamente direcionado através do seu *Uniform Resource Identifier* (URI); e

- Uso de hipermídia na representação dos recursos da aplicação e para as transições de estado: a representação deste estado nesse sistema são tipicamente HTML, *Extensible Markup Language* (XML) e *JavaScript Object Notation* (JSON).

2.7 Resumo do Capítulo

Nesse Capítulo foram explanados os principais tópicos relacionados à temática do projeto. Na Seção 2.1 descreveu-se sobre infraestrutura urbana, que pode ser compreendida como o conjunto de serviços básicos essenciais para à atividade humana no ambiente das cidades, tais como iluminação pública, água, esgoto, transporte, entre outros. Devida a sua natureza física, esses serviços estão sujeitos a desgastes, advindos de fatores físicos, químicos, e defeitos de projeto ou construção, levando ao surgimento de problemas que podem prejudicar, ou até comprometer, o seu funcionamento. É importante a rápida identificação e conserto desses problemas para que eles não cheguem a comprometerem o funcionamento do sistema e os seus serviços prestados.

Na Seção 2.2 foram retratados os principais conceitos relacionados ao processo de Gestão Urbana, essa tem muita importância na administração pública das cidades, devida a sua natureza de basear-se na estipulação de metas e utilização de recursos eficientemente aplicados a setores estratégicos. Com o aumento do uso de Tecnologia de Informação (TI) em organizações, no fim da década de 70 houve um grande movimento mundial dentro dos governos para realizar a reforma do aparelho do estado. Dessa forma, a utilização de TI nesse setor originou os que ficou conhecido como e-Gov. Com a utilização de dispositivos móveis aplicados nas práticas de gestão urbana surgiu o m-Gov, ou governo móvel.

A Seção 2.3 tratou de Sistemas de Navegação Global por Satélites (GNSS), que são sistemas utilizados para calcular a localização de um determinado objeto no globo terrestre. Entre esses sistemas destacam-se o GPS e o GLONASS. Ambos são construídos com base em três segmentos, o segmento espacial, o segmento de controle, e o segmento de usuário. O segmento espacial corresponde à constelação de satélites que orbitam o globo terrestre. O segmento de controle é representado pelas estações de controle responsáveis pelo ajuste de parâmetros como rota e relógio dos satélites. O segmento de usuário corresponde aos diversos modelos de dispositivos capazes de interpretar os sinais dos satélites e calcular a sua localização com base nesses.

Na Seção 2.4 explanou-se o Google Maps, que é um serviço de visualização de mapas através da web fornecido pela empresa Google Inc. E cujas principais funcionalidades do são fornecer mapas interativos, com possibilidade de mudança de escala (*zoom*), mudança de estilos de mapa, permite sobreposição de camadas de informações (imagens de satélite, mapa de elevação, camadas de imagens), movimentação livre através de interação do usuário, e criação de marcadores de localização customizáveis. Além disso mostrou-se brevemente as capacidades da sua API JavaScript, cujo nome oficial é Google Maps JavaScript API v.3.

A Seção 2.5 descreveu o *framework* para construção de aplicações para dispositivos móveis, PhoneGap, que utiliza código JavaScript, HTML, e CSS e os empacota em um programa executável capaz de rodar em diversas plataformas de dispositivos móveis. Além disso mostrou como estão organizados os componentes que compõe a arquitetura de uma aplicação PhoneGap.

Por fim, a Seção 2.6 discutiu sobre a plataforma web, que é uma plataforma para compartilhamento de documentos através do protocolo *Hypertext Transfer Protocol* (HTTP). Nessa plataforma, os documentos de hipertexto são visualizados através do aplicativo cliente chamado de navegador web, o navegador é responsável por se conectar aos servidores web e, através do protocolo HTTP, baixar os documentos e recursos, interpretá-los e exibi-los ao usuário. Os documentos de hipertexto são escritos na linguagem de marcação *HyperText Markup Language* (HTML), pode-se aplicar estilos personalizados de exibição através de *Cascade Style Sheets* (CSS), e seus navegadores também suportam a linguagem de programação JavaScript. A Seção também mostrou uma arquitetura para construção de sistemas de hipermídia que recebe o nome de *Representational State Transfer* (REST). Essa arquitetura é uma abstração da arquitetura da web, consiste num conjunto coordenado de restrições arquiteturais aplicadas a componentes, conectores, e elementos de dados dentro de um sistema distribuído de hipermídia, geralmente é utilizada no desenvolvimento de *Web services*.

3 Trabalhos Relacionados

Neste capítulo são apresentados trabalhos relacionados ao proposto nesta monografia. A Seção 3.1 apresenta o aplicativo móvel Take Vista, uma aplicação para denúncia de problemas baseada em hashtags. A seção 3.2 explana a plataforma Colab, uma rede social colaborativa que aborda vários problemas urbanos. A Seção 3.3 apresenta a plataforma Onde Fui Roubado, uma plataforma voltada para a área de segurança pública. Por fim, a Seção 3.4 faz um comparativo entre os trabalhos relacionados e a plataforma proposta nesse trabalho em relação às plataformas suportadas, às áreas abordadas e a disponibilização de dados.

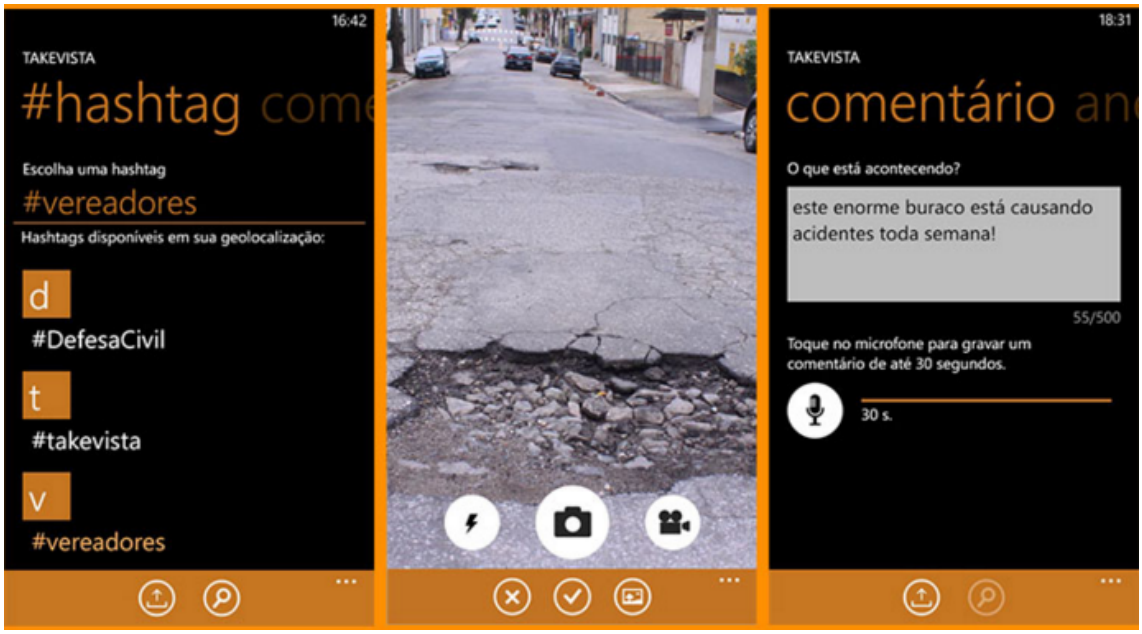
3.1 Take Vista

O Take Vista é um aplicativo para Android, Windows Phone e iOS, foi desenvolvido pela empresa Sisplus no fim do ano de 2014. Permite a comunidade reportar problemas urbanos por meio de fotos, vídeos, textos e áudios e encaminhem as reclamações para os órgãos responsáveis. Desse modo, incentiva as pessoas a participarem ativamente na cobrança por melhoria nas condições de suas cidades.

A Figura 3.1 ilustra algumas das telas do aplicativo que exemplificam a atividade de reportar um problema urbano, segue o seguinte fluxo: o usuário cria um evento contendo as informações, seleciona uma *hashtag* relacionada ao órgão responsável e submete sua denuncia para esse órgão. Após receber a denúncia, o órgão responsável mantém o usuário informado sobre as providências tomadas até a resolução do problema ou a conclusão do evento. É necessário que o usuário esteja no local do problema, na hora de relatar a ocorrência, uma vez que o endereço é adquirido através do GPS do *smartphone* do usuário.

As ocorrências são separadas por *hashtags* que funcionam como uma padronização, possibilitando maior eficiência de busca para o aplicativo, no entanto, isso gera a limitação de não atender todos os tipos de ocorrências. O Take Vista possui um aplicativo para a plataforma *Web* (*Web site*), que pode ser acessado através do navegador *Web*, nele o usuário pode acompanhar suas ocorrências, porém não permite que ele possa reportar uma nova ocorrência através dessa

Figura 3.1 – Telas de submissão de ocorrência no Take Vista



Fonte: Nascimento (2015)

plataforma. Isso dificulta a utilização da ferramenta para pessoas que não possuem *smartphones* compatíveis com a aplicação móvel. Além disso, é uma ferramenta paga que deve ser contratada pelo município.

3.2 Colab

Colab é uma solução de *software*, criada em 2013, que facilita a comunicação entre a população e os governos municipais, permite que a população possa intervir de maneira colaborativa, fiscalizando problemas urbanos, propondo novos projetos e avaliando os serviços realizados (GUSMÃO, 2013).

O Colab é uma rede social, *Web* (*Web site*) e aplicativos móveis para *smartphones* e *tablets* suportando as plataformas Android e iOS. Ao utilizar o Colab a população pode identificar problemas nas suas cidades e levantar cobranças para providências do governo municipal e órgãos responsáveis. Isso é possível através das relações de compartilhamento entre os usuários do sistema em suas redes de colaboração (COLAB, 2018).

A Figura 3.2 ilustra a tela do aplicativo para dispositivo móvel, pode-se observar um exemplo de denúncia, o usuário Caio Oliveira denuncia a falta de um trecho da calçada na rua Tira Destes em Niterói.

Figura 3.2 – Exemplo de postagem de problema, aplicativo móvel Colab

Fonte: (COLAB, 2018)

As postagens são classificadas em alguma área abrangida pela administração municipal e depois é realizada uma triagem dessas informações. O sistema utiliza georreferenciamento, assim os problemas são localizados mais facilmente dentro da cidade.

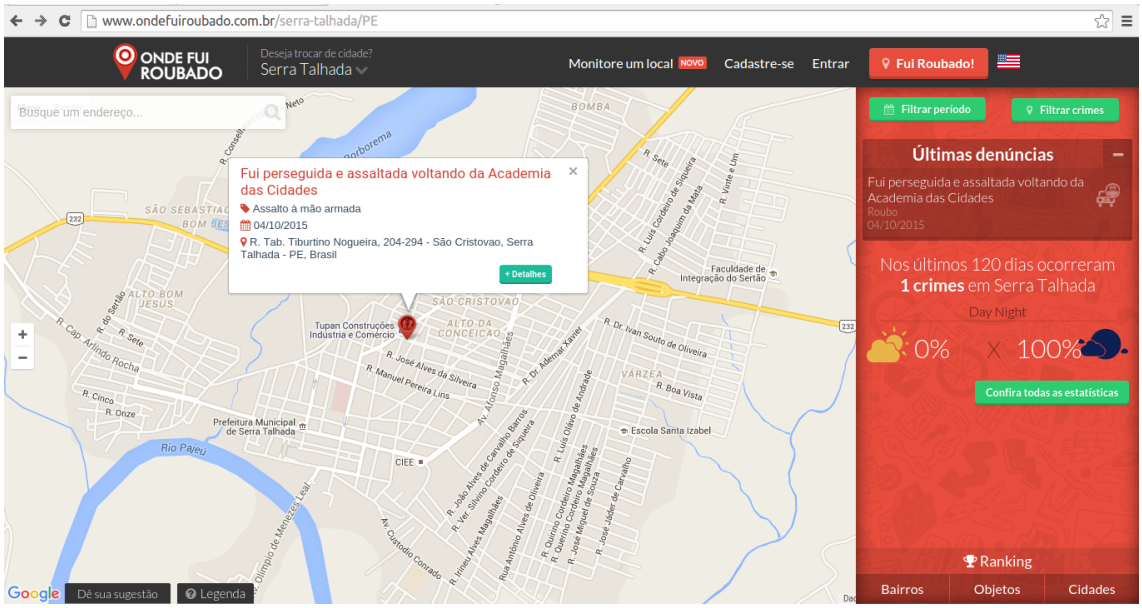
Colab é uma ferramenta que para ser implantada necessita de parceria comercial entre a companhia Colab e o governo municipal. Desse modo, a ferramenta está limitada a alguns poucos municípios que firmaram essa parceria.

3.3 Onde Fui Roubado

A ferramenta intitulada “Onde Fui Roubado” foi desenvolvida em 2013 por dois estudantes da Universidade Federal da Bahia, Márcio Vicente e Fellipe Norton, é composta por um aplicativo *Web* (*Web site*) e um aplicativo para *smartphone*, sua função é mapear locais onde foram registradas ocorrências policiais, como furto e roubo.

A Figura 3.3 ilustra um exemplo de denúncia de assalto na cidade de Serra Talhada – PE. Nela é possível observar a localização do acontecimento no mapa da cidade. Através dessa plataforma, o usuário pode registrar furtos e roubos na sua cidade, alimentando um banco

Figura 3.3 – Visualização de uma ocorrência de assalto no Onde fui roubado, aplicativo *Web*



Fonte: Onde Fui Roubado (2018)

de dados que armazena essas ocorrências. Os dados registrados são principalmente: o tipo da ocorrência, o objeto roubado e a localização do acontecimento. Para a captura desse último, o aplicativo utiliza o GPS do *smartphone*.

É importante salientar que as ocorrências registradas pelo aplicativo são provenientes exclusivamente de usuários que divulgam essas informações no aplicativo, esse não possui vínculo com instituições responsáveis pela Segurança Pública, desse modo as informações coletadas não são redirecionadas para os órgãos competentes para que medidas formais possam ser tomadas.

3.4 Análise dos Trabalhos Relacionados

O levantamento desses trabalhos relacionados foram utilizados critérios como similaridade temática, colaboração entre população e governo, tecnologias utilizadas para desenvolvimento da aplicação, plataformas de acesso suportadas e relevância social. Então, foram criados quadros comparativos para melhor visualização das similaridades entre os trabalhos relacionados.

3.4.1 Plataformas Suportadas

O Quadro 3.1 compara as plataformas suportadas por cada trabalho relacionado, destacando a aplicação e as plataformas suportadas.

Quadro 3.1 – Comparativo dos trabalhos relacionados quanto às plataformas suportadas

Trabalho Relacionado	*Web*	Android	iOS	Windows Phone
Take Vista	N	S	S	S
Colab	S	S	S	N
Onde Fui Roubado	S	S	S	N
Proposta desse Trabalho	S	S	S	S

Fonte: Elaborado pelo autor

No Quadro 3.1 é possível perceber que todos os trabalhos relacionados suportam parcialmente as principais plataformas móveis de acesso. Destacando-se o Colab, que só não dá suporte para a plataforma Windows Phone.

3.4.2 Área de Abordagem dos Problemas

Elaborou-se o Quadro 3.2 para ilustrar quais categorias de problemas cada plataforma propõe-se a atender, identificando as áreas de abordagem dos problemas.

Quadro 3.2 – Comparativo dos trabalhos relacionados quanto à área de abordagem dos problemas

Trabalho Relacionado	Área de Atuação	Categorizados
Take Vista	Todas as áreas	N
Colab	Todas as áreas	S
Onde Fui Roubado	Segurança Pública	S
Proposta desse Trabalho	Infraestrutura	S

Fonte: Elaborado pelo autor

Ao analisar o Quadro 3.2 é possível observar que os trabalhos Take Vista e Colab abordam tipos de problemas de todas as áreas, enquanto que o trabalho Onde Fui Roubado e a solução proposta nesse trabalho focam em áreas específicas. O fato de abordar muitas áreas de problemas faz com que as soluções sejam genéricas a ponto de abstrair funcionalidades específicas para determinadas áreas. Muitas dessas funcionalidades podem ser de grande ajuda aos Governo e Órgãos para resolver determinados tipos de problemas.

Ao focar em uma determinada área de problemas o trabalho proposto nesse projeto tenta oferecer funcionalidades específicas de alguns problemas da área de infraestrutura para con-

tribuir de forma positiva na resolução desses mesmos. Por exemplo, funcionalidade envolvendo mecanismos de otimização de rotas geográficas nos planos de resolução dos problemas, que permitem diminuir custos com transporte e alocação de recursos, aproveitando-se de maneira inteligente das distâncias entre os locais de execução das obras.

Entretanto, vale ressaltar, que apesar da definição de escopo para uma única classe de problema, o projeto proposto permite que novos problemas sejam cadastrados. Esse tipo extensibilidade foi previsto nas definições dos requisitos.

3.4.3 Integração com Outros Sistemas e Dados Abertos

Uma premissa importante quando desenvolve-se soluções de utilidade pública é a questão da transparência, nesse caso, a publicação dos dados devem obedecer boas práticas já estabelecidas, para que a sociedade possa usufruir desses dados.

Com base na análise dos trabalhos relacionados, é mostrado a seguir o Quadro 3.3 comparando questões como permissividade de integração com outros sistemas e também a questão da publicação de dados abertos, uma vez que todos os trabalhos focam em ferramentas de utilidade pública.

Quadro 3.3 – Comparativo dos trabalhos relacionados quanto publicação de dados abertos e permissividade de integração com outros sistemas

Trabalho Relacionado	Disponibiliza dados	Fornece API para integração
Take Vista	N	N
Colab	N	N
Onde Fui Roubado	N	N
Proposta desse Trabalho	S	S

Fonte: Elaborado pelo autor

No Quadro 3.3, observa-se a utilização de boas práticas para caracterizá-los como aberto, como, por exemplo, licença e formato dos dados; e permissividade de cada trabalho relacionado em integrar-se e incentivar essa integração com outros sistemas e ferramentas.

Com base nessa análise é possível notar que todos os trabalhos relacionados levantados não permitem a prática da publicação de dados abertos além de não incentivarem e disponibilizarem interfaces para integração de outros sistemas. Dessa maneira, a solução proposta nesse trabalho pretende publicar dodos não sensíveis em formato aberto e permitir a integração de outros sistemas através de API em *Web service*.

3.5 Considerações Finais

Nesse capítulo foram explanados os trabalhos relacionados. Os aplicativos Take Vista, Colab, e Onde Fui Roubado. Mostrando detalhes como suas plataformas de execução, áreas de abordagem e algumas das suas funcionalidades.

Destaca-se as diferentes abordagem de gerenciamentos de problemas, a exemplo de *hashtags*, cobrança de soluções.

Apesar da positividade das ferramentas, nenhuma incentiva a integração de novos sistemas e ou funcionalidades, sendo esta característica um diferencial do trabalho proposto.

4 Informe.city - Plataforma para Informar e Gerenciar Problemas de Infraestrutura Urbana

Neste capítulo é apresentada a plataforma Informe.city, bem como seus componentes integrantes. Na seção 4.1 são apresentados os tipos de problemas de infraestrutura que foram abordados pela plataforma. Na Seção 4.2 é tratada a etapa de levantamento de requisitos das aplicações. Na Seção 4.3 é mostrada a etapa de elaboração e descrição dos casos de uso das aplicações. A Seção 4.4 descreve os tipos e as categorias de testes que foram utilizadas para validar as aplicações. Na Seção 4.5 são apresentadas as classes do modelo da aplicação. A Seção 4.6 mostra como foi projetada a arquitetura da plataforma. Na Seção 4.7 é explanado o desenvolvimento dos sistemas que compõe a plataforma, especificando cada componente em suas subseções. Por fim, na Seção 4.8 descrevem-se os testes para validação da plataforma.

4.1 Definição dos Problemas Abordados

Seguida a ordem estabelecida pelo método desse projeto, Seção 1.4, a primeira etapa consistiu na definição dos problemas de infraestrutura que foram abordados pela plataforma.

Os problemas foram determinados a partir das análises dos trabalhos relacionados e do domínio da aplicação. Dessa maneira, foram identificados vários problemas de infraestrutura urbana, esses foram agrupados em categorias baseando-se em suas similaridades temáticas. Essa abordagem teve como objetivo oferecer maior clareza para os usuários das aplicações em identificar e manipular os problemas. A seguir são listadas as categorias de problemas que foram estabelecidas para serem cobertas inicialmente pela plataforma de software desenvolvida nesse trabalho:

- Água e Esgoto;
- Luz e Energia;
- Pedestres;

- Transporte Público;
- Urbanismo; e
- Vias e Trânsito.

Essas categorias são compostas de vários problemas de infraestrutura que são comuns às cidades. Assim cada categoria abriga uma lista de tipos de problemas, a exemplo a categoria Água e Esgoto, que agrupa problemas como: bueiro entupido, bueiro sem tampa, esgoto a céu aberto, falta de água, ponto de inundação e cano estourado. Dessa maneira, um usuário pode identificar um problema na cidade e enquadrá-lo no tipo que melhor se aproxime da realidade. A lista com todas as categorias e seus problemas é apresentada no Apêndice A.

Ainda nessa etapa do projeto, identificou-se a necessidade de permitir que o governo ou órgãos responsáveis pela resolução dos problemas pudessem acrescentar novas categorias e tipos de problemas conforme sua necessidade. Em consequência disso foi elaborado um requisito funcional para a aplicação cliente, responsável pelo gerenciamento de problemas. Esse requisito será explanado a seguir, na Seção 4.2, seguindo a ordem do método adotado.

4.2 Levantamento de Requisitos

Levou-se em consideração o requisito inicial de que a plataforma seria composta por múltiplos subsistemas, pois abrange plataformas móveis e a *Web*, disponibilizando um *Web service* e duas aplicações cliente. O *Web service* reúne, coordena e fornece dados para as aplicações cliente. As duas aplicações cliente servem para gerenciar e informar problemas, respectivamente.

Sendo assim, por convenção e seguindo boas práticas de Engenharia de *Software* a luz de Pressman (2006), os requisitos serão apresentados neste trabalho partindo da perspectiva das necessidades dos atores que interagirão com cada parte da plataforma. Esses atores foram identificados e definidos conforme o exposto no Quadro 4.1 a seguir:

Quadro 4.1 – Identificação dos atores para cada subsistema da plataforma

Subsistema da Plataforma	**Atores das Interações**
Web service	Aplicação para Reporte, Aplicação para Gerenciamento, Sistemas de Terceiros
Aplicação para Gerenciar Problemas	Governo/Órgão (Administrador)
Aplicação para Reportar Problemas	Usuário Comum (Cidadão)

Fonte: Elaborado pelo autor

Conforme o Quadro 4.1, a aplicação para gerenciamento dos problemas é utilizada pelo governo ou órgãos, a aplicação para reportar problemas é utilizada pelo cidadão e o *Web service* interage com vários sistemas clientes: aplicação para gerenciamento de problemas, aplicação para informar problemas e outros possíveis sistemas que possam vir a utilizar seus serviços, pois trata-se de uma aplicação servidora.

Como exemplo de prováveis sistemas que possam utilizar os serviços do *Web service*, podem-se citar: portal de dados abertos, mecanismos de busca e sistemas autônomos para monitoramento da infraestrutura urbana. Todavia, esse projeto limitou-se a desenvolver somente as duas aplicações clientes já explanadas.

Também, nessa etapa, foram levantados os requisitos, utilizando-se mecanismos como: a análise dos trabalhos relacionados, observação do ambiente urbano e métodos para monitoramento e controle de problemas. Utilizando buscas na literatura de referência e na *Internet* como fonte dessas informações. Dessa maneira foi elaborado o documento de requisitos para a plataforma, fazendo-se separação dos mesmos pelos subsistemas do quais se tratam.

Para fins de demonstração, serão apresentados somente alguns requisitos para exemplificar como esses foram elaborados. Esses requisitos podem ser funcionais, não funcionais ou de domínio, servindo como amostra do conjunto levantado. A lista completa com todos os requisitos pode ser consultada no Apêndice B.

4.2.1 Requisitos da Aplicação Servidora (*Web Service*)

A seguir são listados alguns dos requisitos para aplicação servidora, o *Web service*.

- [**RF01**] O sistema deve oferecer uma API em formato REST consumindo e retornando valores codificados em JSON;
- [**RF03**] O sistema deve oferecer acesso público para leitura dos seguintes recursos: Problema, Categoria de Problema, e Tipo de problema, não necessitando de autenticação para tal;
- [**RNF01**] O mecanismo de autenticação deve ser por meio de chave de acesso a API para os pontos de acesso restrito. Essas chaves devem ser codificadas no formato *JSON WebTokens*;

- [**RNF02**] O sistema deve suportar uma carga mínima de 200 requisições por segundo.
- [**RNF03**] *Web service* - deve oferecer acesso seguro através do uso do protocolo HTTP sob *Transport Layer Security* (TLS), conhecido como HTTPS;

4.2.2 Requisitos da Aplicação para Gestão de Problemas

A seguir são listados alguns dos requisitos definidos para a aplicação responsável pelo gerenciamento de problemas:

- [**RF01**] O sistema deve representar visualmente a localização dos problemas em um mapa virtual utilizando as informações de localização;
- [**RF02**] O sistema deve disponibilizar estatísticas de problemas com base nas relações: problemas por tipo, problemas por categoria e problemas por situação;
- [**RF03**] O sistema deve calcular uma rota de navegação otimizada que percorra os problemas recém informados para as suas fiscalizações e resoluções, buscando a economia nos custos de deslocamento totais;
- [**RF04**] O sistema deve permitir o cadastro de novas categorias e de novos tipos de problemas.
- [**RNF01**] O sistema deve apresentar interface responsiva para vários tamanhos de telas, compatíveis com dispositivos como *smartphones*, *tablets*, *notebooks* e *desktops*, os quais permitam acesso à *Web*;

4.2.3 Requisitos da Aplicação para Reporte de Problemas

A seguir, exemplificam-se alguns dos requisitos para a aplicação utilizada para reportar problemas de infraestrutura à plataforma.

- [**RF03**] O sistema deve possibilitar ao usuário escolher como informar a localização do problema, se deseja apontar uma localização geográfica manualmente ou utilizar a localização do dispositivo através do GNSS do mesmo.

- [**RF04**] O sistema deve possibilitar que o usuário selecione o tipo de problema de infraestrutura que deseja reportar. A aplicação deve buscar intuitividade a seleção utilizando a abordagem de apresentar os tipos de problemas em categorias, agrupados pela similaridade temática.

- [**RF05**] O sistema deve permitir ao usuário acompanhar um problema a partir do momento em que for informado, através da mudança na sua situação, que pode ser classificado como: informado, andamento ou solucionado.

- [**RNF01**] O sistema deve funcionar em *smartphones* com as plataformas Android, iOS, e se possível Windows Phone.

4.3 Planejamento de Casos de Uso e de Testes

Baseando-se nos requisitos já levantados, foram elaborados os casos de uso para cada um dos subsistemas que compõe a plataforma. Cada caso de uso abstrai como determinada funcionalidade será explanada pela aplicação e quais atores interagirão com ela. Seguindo-se a convenção estabelecida anteriormente, foram elaborados diagramas de caso de uso para cada um dos subsistemas da plataforma, como exposto no Quadro 4.1.

Para elaboração desses diagramas foi dada em UML.

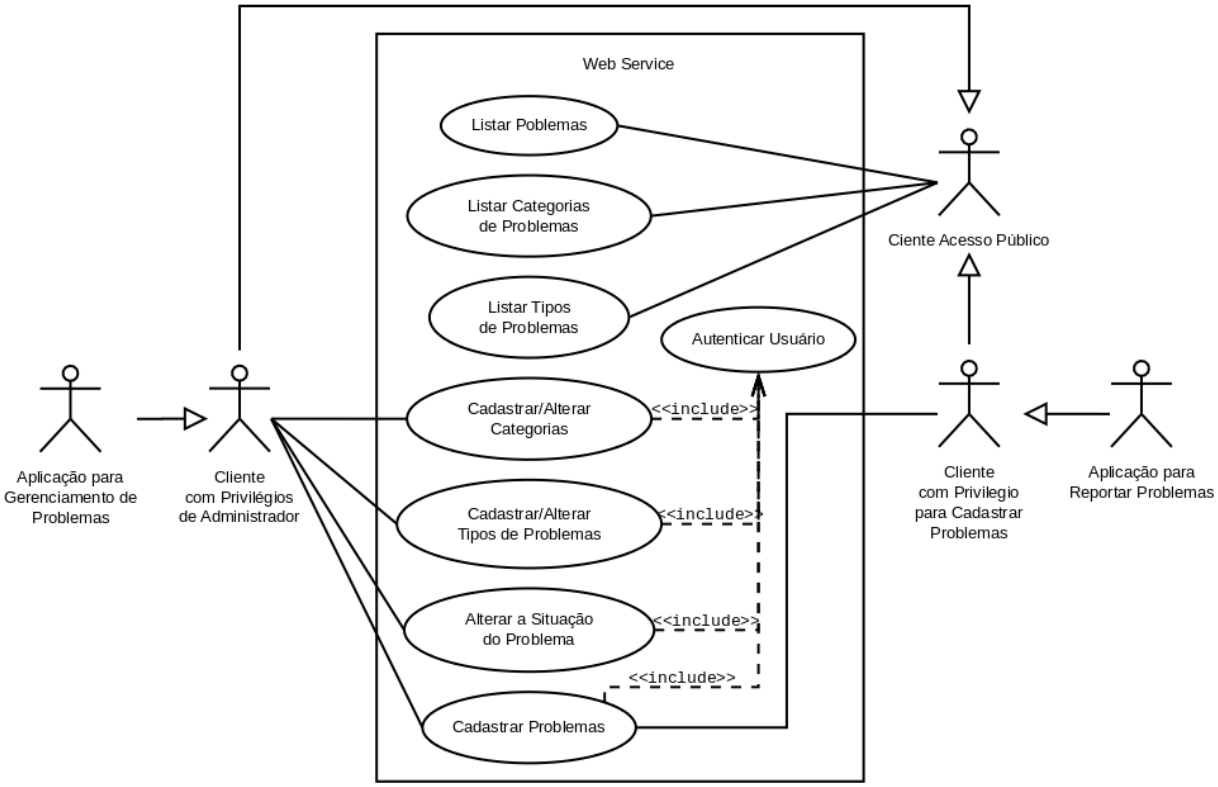
4.3.1 Casos de Uso do *Web Service*

O diagrama de caso de uso simplificado e apresentado na Figura 4.1. Percebem-se os principais casos de uso para o *Web service* da plataforma.

Na Figura 4.1 é possível observar os atores que interagem com esse componente. Os atores representados são abstrações de aplicações cliente que consumirão os serviços ofertados pelo servidor. Cada ator tem um conjunto de privilégios determinado pelas credenciais no caso de uso de autenticação. A rejeição e a permissão de acesso a um serviço define qual função uma aplicação cliente exercerá na plataforma.

Os casos de uso “Listar Problemas”, “Listar Categorias de Problemas” e “Listar Tipos de Problemas” são de acesso público, pois permitem que aplicações cliente consumam seus

Figura 4.1 – Diagrama de caso de uso do *Web service*



Fonte: Elaborada pelo autor

serviços sem a necessidade de autenticação. Esses casos de uso estão de acordo com os requisitos de publicação dos dados de domínio público.

Os casos de uso “Cadastrar/Alterar Categorias”, “Cadastrar/Alterar Tipos de Problemas” e “Alterar a Situação do Problema” explanam operações cujo acesso são de exclusividade de atores com privilégio de administração. O caso de uso “Alterar a Situação do Problema” tem sua função auto-explicada pelo seu nome.

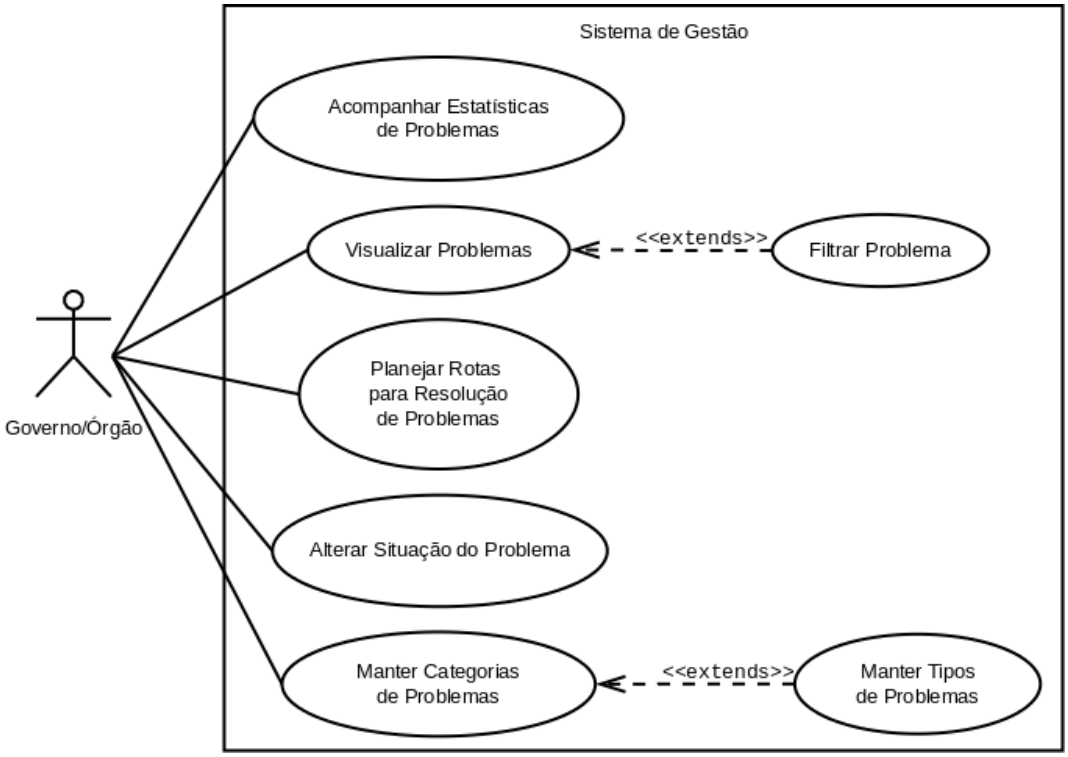
O caso de uso “Autenticar Usuário” é responsável pelo mecanismo de liberação ou bloqueio de acesso aos serviços privados do sistema.

O caso de uso “Cadastrar Problemas” é o principal caso de uso do serviço, pois suas operações têm alguma relação, direta ou indireta, com os demais casos de uso. Além disso, é responsável por cumprir o principal requisito da plataforma. Esse requisito permite que os problemas sejam informados. A partir disso, persiste os dados dos problemas e compartilha-os com as demais aplicações.

4.3.2 Casos de Uso da Aplicação para Gestão de Problemas

A Figura 4.2 apresenta o diagrama simplificado de casos de uso da aplicação para gerenciamento dos problemas informados. Nela pode-se identificar os principais casos de uso e o ator que interage com o sistema, nesse caso a aplicação para gestão de problemas. Esse ator está identificado como governo ou órgão. Na prática ele corresponderá a um ou mais servidores públicos ou funcionário de algum órgão competente. Por sua vez, ele terá responsabilidades como: acompanhar problemas, informar as mudanças de estado dos problemas ao longo do seu processo de resolução, elaborar plano de rotas para a resolução dos problemas, bem como cadastrar novas categorias e tipos de problemas que possam surgir em decorrência das suas necessidades e dos cidadãos. Esse usuário deve possuir privilégio de gestor para ter acesso à tais casos de uso, pois tem potencial de alterar significativamente o estado e a qualidade da plataforma como um todo.

Figura 4.2 – Diagrama simplificado de casos de uso da aplicação para gerenciar problemas



Fonte: Elaborada pelo autor

O caso de uso “Acompanhar Estatísticas de Problemas” oferece informações úteis para o gestor sobre os problemas existentes na cidade. Informações como quais tipos de problemas são mais frequentes e o quantitativo de problemas em relação as suas situações. Dessa forma, o

gestor tem uma visão ampla dos processos de reporte e resolução de problemas.

O caso de uso “Visualizar Problemas” oferece uma maneira rápida e amigável para o gestor encontrar os problemas informados pelos cidadãos, podendo ainda “Filtrar Problemas” pelas suas situações, seus tipos e suas categorias.

O caso de uso “Planejar Rotas para Resolução de Problemas” oferece um mecanismo de planejamento otimizado para rotas de navegação, que é útil para economia de recursos no processo logístico da resolução de problemas de infraestrutura. Ele permite que, a partir das informações de georreferenciamento dos problemas e do ponto de origem dos recursos, seja calculada uma rota otimizada no tocante a distância percorrida para se chegar aos problemas. Permite a economia de recursos como combustível e tempo, além de fornecer maior controle para o gestor, que terá de antemão um plano preestabelecido de fiscalização/resolução de problemas.

O caso de uso “Alterar Situação do Problema” permite que o gestor registre as mudanças de situação que ocorram nos problemas, desde o momento que seja informado, quando seja encaminhado para resolução ou quando finalmente estiver resolvido. Permitindo, consequentemente, a população acompanhar esse ciclo de vida e o gestor ter ciência e controle sobre o andamento dos processos no tocante a administração de problemas de infraestrutura.

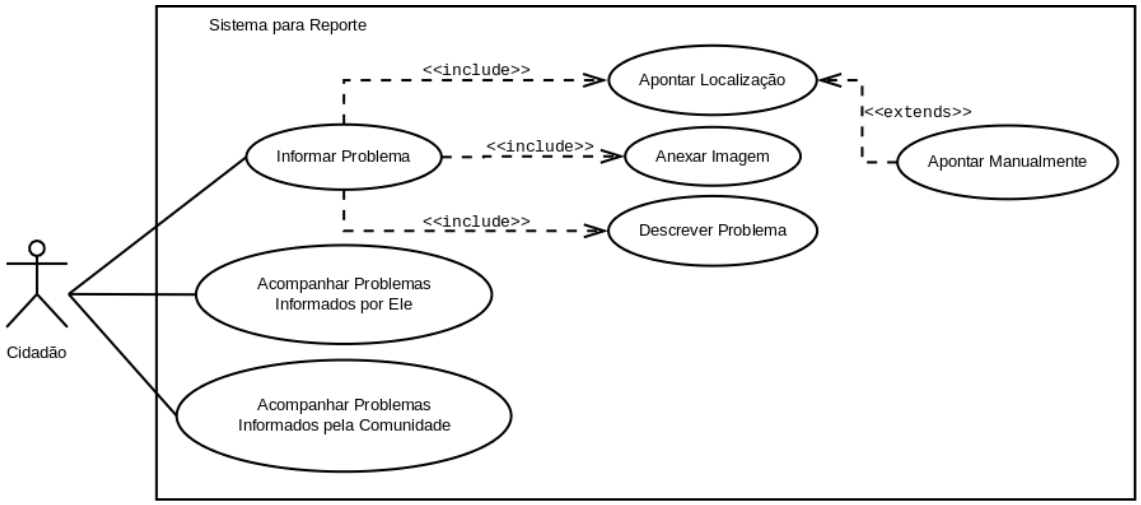
Os casos de uso “Manter Categoria de Problemas” e “Manter Tipos de Problemas” surgiram pelo acréscimo de um requisito funcional que define a possibilidade de ampliar o escopo da plataforma. Nesse trabalho foi escolhido o escopo de problemas de infraestrutura urbana, estabelecidos na primeira etapa do método, porém com essa funcionalidade a plataforma proposta nesse projeto poderá cobrir outras áreas de atuação, como, por exemplo, saúde e segurança pública. Dessa forma, esse caso de uso permite a plataforma estender sua aplicabilidade em demandas futuras. É importante acrescentar que apesar dessa viabilidade esse projeto não aplicou em seu escopo tais possibilidades.

4.3.3 Casos de Uso da Aplicação para Reporte de Problemas

A Figura 4.3 apresenta uma simplificação do diagrama de caso de uso da aplicação para reportar problemas. Essa aplicação é disponibilizada para a população de maneira a permitir que um cidadão possa informar ocorrência de algum problema na cidade, utilizando para tal as principais plataformas móveis, a saber: Android, iOS e Windows Phone.

De acordo com a Figura 4.3, o ator, identificado como “Cidadão”, poderá ter acesso as

Figura 4.3 – Diagrama de casos de uso da aplicação para reporte de problemas



Fonte: Elaborada pelo autor

funcionalidades abstraídas em cada caso de uso, a saber: poder informar problemas encontrados na cidade apontando sua localização, anexando uma imagem para comprovar a denúncia, além de complementar com informações que auxiliem o governo municipal ou órgãos competentes no processo de resolução do problema. Outros casos de uso incluem o acompanhamento dos problemas informados desde sua denúncia até sua resolução.

Os casos de uso para acompanhamento de problemas proporcionam transparência nos processos públicos, servindo como meio de fiscalização cidadã para as operações de resolução de problemas do governo e de órgãos competentes.

O caso de uso “Apontar Localização” atribui qualidade e precisão para a atividade de identificação de problemas e serve como uma boa solução no aspecto relacionado aos meios convencionais de denúncia, como, por exemplo, a telefonia, pois as informações de localização prestadas nesses serviços são imprecisas e, muitas vezes, incompletas, já o georreferenciamento é preciso rápido e completo.

O caso de uso “Anexar Imagem” serve como principal mecanismo de validação e autenticidade do problema pois ele determina que o usuário comprove a existência do problema através de uma imagem real, dificultando a criação de falsas denúncias por cidadãos mal-intencionados.

Esses casos de uso abstraíram problemas existentes e ofereceram soluções viáveis e mais intuitivas para os cidadãos quando comparados aos mecanismos tradicionais de denúncia.

4.3.4 Descrição dos Casos de Uso

Foram elaboradas descrições para os principais casos de uso identificados nas seções anteriores. De modo a elaborar os possíveis cenários e seus fluxos e ainda determinar como se dá o comportamento do sistema ao interagir com os atores. Essas especificações de casos de uso foram fundamentais para a elaboração de casos de teste funcionais para o processo de validação da plataforma, seguindo o exposto por Pressman (2006, p. 120).

Nesse projeto, optou-se por seguir o modelo de descrição numerada baseando-se em Bezerra (2006, p. 47). De forma sucinta, e somente para fins de exemplo, foram demonstradas no Apêndice C algumas das descrições de caso de uso elaboradas.

4.4 Planejamento dos Testes

Os testes foram derivados dos requisitos funcionais, não funcionais e dos casos de uso para cada aplicação. Também foi estabelecido qual abordagem de teste cada subsistema utilizou, como apresentado no Quadro 4.2.

Quadro 4.2 – Abordagens de teste definidas para os subsistemas da plataforma

Aplicação alvo dos testes	Abordagem caixa-preta	Abordagem caixa-branca
Web service	S	S
Cliente para Gestão de Problemas	S	N
Cliente para Reportar Problemas	S	N

Fonte: Elaborado pelo autor

Conforme o Quadro 4.2, foi estabelecido que a aplicação *Web service* fosse testada em ambas abordagens.

Nessa aplicação foi definido a realização de um teste de desempenho, que segue a abordagem caixa-preta, conhecido como teste de carga. Apesar de não ser um teste funcional ele é importante para se verificar a capacidade da aplicação em relação ao seu desempenho e consumo de recursos, conhecendo-se assim as suas limitações. Estabeleceu-se o uso da ferramenta Vegeta, versão 11.0, para realização desses testes.

Ainda para a aplicação *Web service*, definiu-se a elaboração de testes unitários para abordagem caixa-branca. Os mesmos foram implementados durante o desenvolvimento da aplicação utilizando-se as ferramentas para testes da linguagem Go. Objetivou-se avaliar e

validar o funcionamento das rotinas de código da REST API a partir desses testes.

Na Figura 4.4 é ilustrado um trecho de código Go para exemplificar como foram elaborados os testes unitários caixa-branca.

Figura 4.4 – Trecho de código Go que implementa um caso de teste para o serviço de cadastro de usuário da REST API

```
var (
	eng       = gin.New()
	usuarioID = ""
)

func init() {
	eng.POST("usuario", CreateUsuarioCtrl)
}

...

func TestCreateUsuarioCtrl(t *testing.T) {

	t.Run("NoContentType", func(t *testing.T) {
		r, err := http.NewRequest(http.MethodPost, "/usuario", nil)
		if err != nil {
			t.Fatal(err)
		}
		w := httptest.NewRecorder()
		eng.ServeHTTP(w, r)
		if w.Code != http.StatusBadRequest {
			t.Fail()
		}
	})
...
```

Fonte: Elaborada pelo autor

Conforme a Figura 4.4, pode-se observar o teste da unidade de *software* responsável por realizar o cadastro de usuários no sistema. A rotina serviu para verificar o comportamento da requisição quando é omitido o cabeçalho HTTP que especifica o tipo de conteúdo enviado, ou seja, o *Content-Type*. Conforme o caso de uso dessa funcionalidade, o sistema cancela a requisição e retorna o código 401, ou seja, que houve uma requisição mal elaborada.

O teste unitário avalia o resultado obtido e verifica se está de acordo com o estabelecido, em caso de desacordo o teste informa a falha.

O Quadro 4.2 ainda permite observar a abordagem caixa-preta para as duas aplicações clientes. Decidiu-se realizá-los seguindo o fluxo de ações dos requisitos funcionais e seus valores de classes de equivalência para as entradas de dados. As classes de equivalência são intervalos e exemplos de valores permitidos e não permitidos como entrada e saída do sistema. Um exemplo é o uso de valores aceitos no campo *"e-mail"* do formulário de cadastro de usuário. Esse campo

deve aceitar somente valores válidos como *e-mail* e os inválidos devem ser recusados.

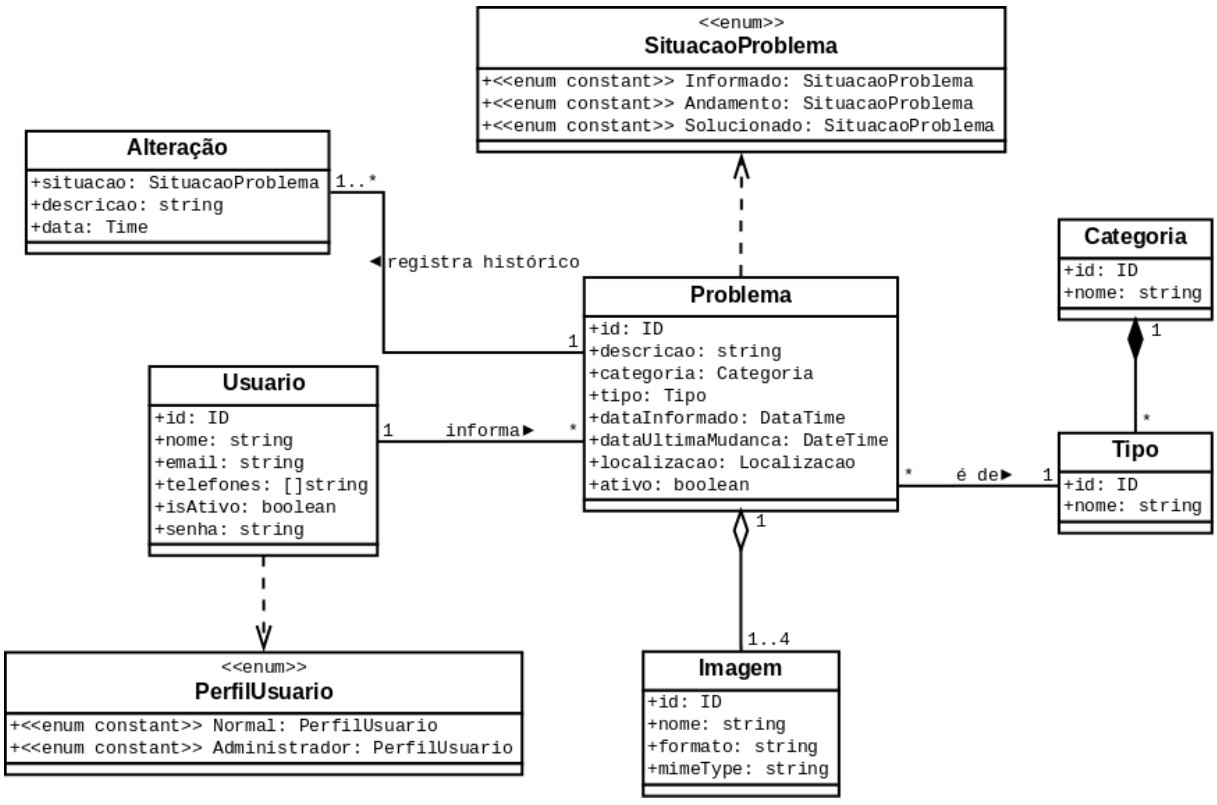
Foi estabelecido que os testes de usabilidade fossem executados ao longo do desenvolvimento do sistema, atentando para o fluxo de exceções identificados e possíveis falhas, sendo corrigidos de imediato. Após o desenvolvimento dessas aplicações os testes seriam executados novamente. A execução dos testes e seus resultados são explanados na Seção 4.8.

4.5 Modelagem da Plataforma

Utilizando o Desenvolvimento de Software baseado no Paradigma de Orientação a Objetos, essa etapa consistiu em extrair, a partir dos requisitos e dos casos de uso, quais classes compõem o modelo das aplicações e suas inter-relações.

Tratando-se de uma plataforma única, composta por subsistemas que funcionam sobre um mesmo domínio de negócio, as classes de modelo identificadas são comuns a todos os subsistemas da plataforma, como *Web service*, cliente de gestão e cliente para reportar problemas. Na Figura 4.5 é apresentado um diagrama de classe simplificado mostrando classes do modelo de negócio da plataforma .

Figura 4.5 – Diagrama simplificado de classes para os modelos do domínio da aplicação



Fonte: Elaborada pelo autor

Observando-se o modelo simplificado da aplicação na Figura 4.5, é possível observar a existência de quatro classes principais e outras auxiliares na composição do modelo. As principais classes identificadas foram:

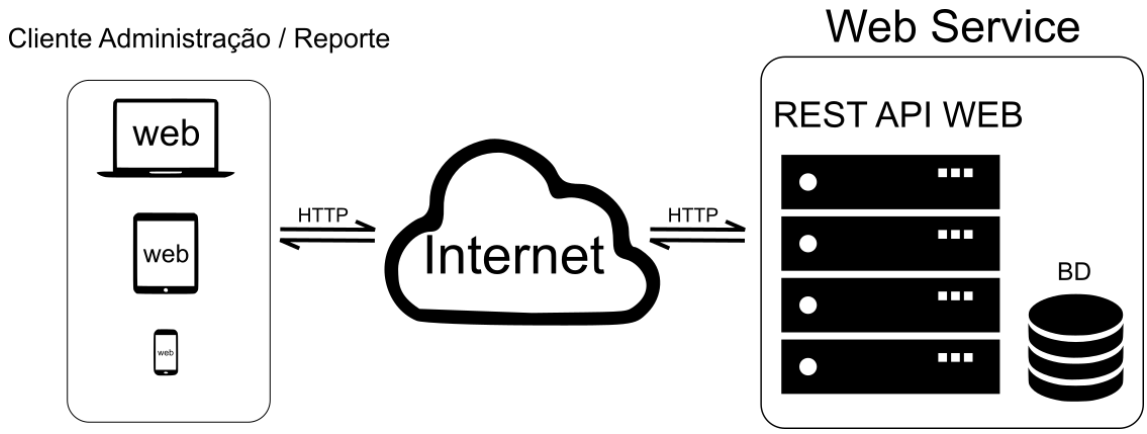
- **Usuário** - classe que abstrai informações dos usuários que terão acesso aos sistemas. Está diretamente ligada ao perfil de acesso que um usuário tem ao interagir com a plataforma. Relaciona-se com a classe Problema, uma vez que usuários de perfil normal (cidadão) informarão problemas na plataforma.
- **Problema** - é a principal classe, pois engloba as informações necessárias para permitir os processos de gerenciamento. É a classe de domínio da aplicação, relaciona-se com a classe Alteração para registrar as mudanças de situação do problema; e relaciona-se com a classe Tipo, que indica a natureza do problema retratado.
- **Tipo** - é a classe responsável por abstrair as informações dos tipos de problemas de infraestrutura encontrados nas cidades. Relaciona-se por composição com a classe Categoria.
- **Categoria** - representa um agrupamento lógico de tipos de problemas.

4.6 Arquitetura da Plataforma

A arquitetura do sistema foi estabelecida com base no levantamento de requisitos, casos de uso, modelagem das classes do domínio e padrões consolidados utilizados em sistemas similares. A adoção de uma arquitetura cliente-servidor é uma regra geral em aplicações dessa natureza e têm demonstrado ao longo dos anos ótima capacidade em atender os requisitos de distribuição e escalabilidade (TANENBAUM, 2003). A seguir, a Figura 4.6 apresentada um diagrama que mostra de maneira simplificada a arquitetura adotada pela plataforma desenvolvida desse projeto.

A Figura 4.6 mostra uma visão geral da arquitetura do sistema e seus componentes, enfatizando onde cada componente será executado. Nela, pode-se observar que o *Web service* (REST API) executa no servidor HTTP e tem integração com o SGBD. Os dois clientes se comunicam com o *Web service* através do protocolo HTTP utilizando a *Internet* como rede. O cliente de gestão é um aplicativo *Web* que executa no navegador e pode ser acessado em qualquer dispositivo que dê suporte a essa plataforma. O cliente para “Reporte de Problemas”

Figura 4.6 – Arquitetura da plataforma com seus subsistemas e dispositivos de execução



Fonte: Elaborada pelo autor

é um aplicativo móvel para as plataformas suportadas pelo *framework* PhoneGap, a princípio Android, iOS, e Windows Phone.

4.7 Desenvolvimento da Plataforma

Esse seção discorre sobre o desenvolvimento das aplicações que compõem a solução de *software* proposta nesta monografia. Abrangendo a quinta e sexta etapa estabelecidas pelo método adotado.

4.7.1 *Web Service* e REST API

O primeiro sistema desenvolvido foi o *back-end* das aplicações ou *Web service*, cuja principal incumbência foi servir uma API seguindo o modelo arquitetural REST. Esse *Web service* funciona em conjunto com o servidor de banco de dados para persistir os dados dos modelos da aplicação, conforme a arquitetura apresentada na seção anterior, presente na Figura 4.6.

O *Web service* foi desenvolvido utilizando-se a linguagem de programação Go versão 1.10.3 e o *framework* Gin versão 1. A escolha dessa linguagem foi motivada pela suas características de alta concorrência entre processos, suporte nativo para desenvolvimento de serviços HTTP através de sua biblioteca padrão e, acima de todas elas, a comprovada eficiência no desenvolvimento de serviços para *Internet*.

Foi utilizado o banco de dados MongoDB, versão 3.4.16, que é um banco de dados NoSQL, cuja abordagem difere significativamente dos banco de dados relacionais, pois é orientado a documentos. MongoDB armazena seus dados em coleções de documentos sem a necessidade de estruturação e normalização utilizadas nos banco de dados para a *Structured Query Language* (SQL). O principal motivo para sua utilização, deve-se ao fato de MongoDB oferecer alto grau de escalabilidade, possuir um mecanismo para armazenamento de arquivos chamado *Mongo File System* (MongoFS), utilizado para armazenar as imagens dos problemas reportados, e implementar armazenamento e operadores de consulta para dados geográficos nativamente, utilizado para os dados para georreferenciamento dos problemas.

A partir das classes de modelo levantadas na quarta etapa do método estabelecido, foram definidos os recursos ofertados pela REST API. Esses recursos são: Categoria, Tipo de Problema, Imagem, Problema e Usuários.

Cada recurso é unicamente identificado por um URI que serve como mecanismo de acesso ao mesmo através de chamadas HTTP. Uma interface com o mesmo URI de recurso pode reagir de forma diferente dependendo do método HTTP utilizado na requisição e também conforme parâmetros e conteúdo enviados na mesma.

Todos os serviços implementados na REST API estão de acordo com os casos de uso e suas restrições de acesso estabelecidas na etapa de planejamento.

O Quadro 4.3 expõe interfaces para os serviços que foram implementados. Pode-se observar o serviço ofertado de acordo o URI do recurso e do método utilizado na solicitação HTTP.

Conforme a arquitetura REST, interfaces para consulta de dados foram desenvolvidas para responderem à requisições HTTP do tipo GET. Interfaces de serviços para atualização de dados dos recursos foram implementados para serem acessados por meio do método PUT. Por fim, as interfaces de serviços para criação de recursos foram desenvolvidas para serem acessadas através do método POST.

Essa REST API foi desenvolvida atendendo os requisitos e os casos de uso estabelecidos. Interfaces para consulta podem aceitar parâmetros nas suas requisições. Esses parâmetros alteram o comportamento padrão dos serviços e servem como mecanismos para configurar ou alterar suas saídas. Interfaces para alteração e criação de recursos necessitam que os dados a serem alterados ou inseridos sejam informados no corpo da requisições. Esses dados são utilizados para substituírem os antigos (atualização), ou para serem persistidos (criação) no recurso destino da solicitação. Para esse projeto, optou-se pela utilização do formato JSON para codificação dos

Quadro 4.3 – Interfaces de acesso aos serviços (API do *Web service*)

Serviço	**Método**	***Uniform Resource Identifier* (URI)**
Realizar login	POST	https://informe.city/login
Renovar chave de acesso	GET	https://informe.city/auth/refresh_token
Listar categorias	GET	https://informe.city/api/v1.0/categoria
Criar uma categoria	POST	https://informe.city/api/v1.0/categoria
Consultar uma categoria	GET	https://informe.city/api/v1.0/categoria/{id_categoria}
Atualizar dados da categoria	PUT	https://informe.city/api/v1.0/categoria/{id_categoria}
Listar tipos de problemas da categoria	GET	https://informe.city/api/v1.0/categoria/{id_categoria}/tipos
Criar tipo de problema em uma categoria	POST	https://informe.city/api/v1.0/categoria/{id_categoria}/tipos
Consultar um tipo problema	GET	https://informe.city/api/v1.0/categoria/{id_categoria}/tipos/{id_tipo}
Atualizar dados do tipo de problema	PUT	https://informe.city/api/v1.0/categoria/{id_categoria}/tipos/{id_tipo}
Download de imagem	GET	https://informe.city/media/{id_imagem}
Upload de imagens	POST	https://informe.city/media/upload
Listar problemas	GET	https://informe.city/api/v1.0/problema
Criar um problema	POST	https://informe.city/api/v1.0/problema
Consultar um poblema	GET	https://informe.city/api/v1.0/problema/{id_problema}
Atualizar dados do problema	PUT	https://informe.city/api/v1.0/problema/{id_problema}
Listar usuários	GET	https://informe.city/api/v1.0/usuario
Criar usuário	POST	https://informe.city/api/v1.0/usuario
Consultar um usuário	GET	https://informe.city/api/v1.0/usuario/{id_usuario}
Consultar o usuário dono da chave de acesso	GET	https://informe.city/api/v1.0/me
Atualizar dados do usuário	PUT	https://informe.city/api/v1.0/usuario/{id_usuario}

Fonte: Elaborado pelo autor

dados em todas as interfaces do *Web service*.

O mecanismo para controle de acesso foi desenvolvido utilizando-se o modelo de chave de acesso à API. As chaves são codificadas no modelo *JSON Web Tokens*. Essa chave de acesso é gerada através do URI “https://informe.city/login” para contas de usuários ativas. E possuem um prazo de validade que corresponde a um período de seis meses. Passado esse período de tempo, a chave fica inutilizável. O *Web service* permite a renovação de chaves de acesso, gerando-se uma nova chave a partir de uma pré-existente e válida, por meio do URI “https://informe.city/auth/refresh_token”.

Chaves de acesso contém todos os dados necessários para a identificação de um usuário encriptados e armazenados em si mesmas. Somente o *Web service* gerador da chave tem acesso a seus dados sensíveis.

Em solicitações a recursos não públicos, seguindo-se as especificações dos requisitos, deve-se informar a chave de acesso para que o *Web service* a avalie e identifique se o usuário da chave tem permissão para acessar os recursos. Dessa forma, garantiu-se que os recursos só são acessados ou modificados por usuários com as permissões para tal.

A seguir será mostrado, para fins de demonstração, o funcionamento do processamento de execução de uma requisição no *Web service* desenvolvido. Neste exemplo, foi escolhido o URI “https://informe.city/api/v0.1/usuario” e o método GET. Essa interface, quando utilizado o método GET, oferece acesso aos recursos que correspondem aos usuários cadastrados no sistema. As requisições para esse serviço são autorizadas somente para usuários com privilégios de administrador, dessa forma, faz-se necessário que previamente um usuário com tal permissão tenha gerado uma chave válida de acesso a API.

Essa chave é então informada através do cabeçalho “*Authorization*” para que o mecanismo de validação possa processá-la.

Figura 4.7 – Representação de requisição HTTP 1.1 para a REST API desenvolvida. A chave de acesso foi encurtada por questão de legibilidade

```
GET /api/v1.0/usuario HTTP/1.1
Host: informe.city
Accept: application/json
Authorization: Bearer eyJhbGciOiJI ... tglKhzclCqyTba1EPLa5Z_480
```

Fonte: Elaborada pelo autor

A solicitação, representada na Figura 4.7, é encaminhada através da Internet até o *Web service* desenvolvido, depois ele extraí da chave de acesso as informações do usuário e verifica se o mesmo tem permissão para tal. Autorizada a requisição, ela é encaminhada para o procedimento responsável por processá-la. A seguir, na Figura 4.8, é ilustrado um trecho de código na linguagem Go que trata a requisição exemplificada.

O trecho de código, da Figura 4.8, serve como demonstração de como foram implementados os procedimentos para execução dos serviços no *Web service*. Procedimentos similares foram desenvolvidos para cada uma das interfaces de acesso mostradas no Quadro 4.3. Cada procedimento implementou as operações necessárias para atender os requisitos e casos de uso da aplicação.

Ainda com base no procedimento desenvolvido, ilustrado na Figura 4.8, é apresentada uma possível resposta do *Web service* gerada pelo mesmo. Para melhor legibilidade, foram desconsiderados os cabeçalhos da resposta à requisição HTTP e é mostrado apenas o conteúdo do seu corpo.

Como ilustrado na Figura 4.9, a API codifica os dados dos recursos no formato JSON. A principal vantagem desse formato é em relação a sua legibilidade tanto por máquina quanto por humanos e seu tamanho reduzido quando comparado a outras formatações utilizadas, a exemplo de XML.

Figura 4.8 – Código desenvolvido na linguagem Go responsável por processar a requisição exposta na Figura 4.7

```
func ListUsuariosCtrl(ctx *gin.Context) {

	pg, err := response.NewPageFromGinContext(ctx)
	if err != nil {
		statusBadRequest(ctx)
		return
	}

	usuarios := []model.Usuario{}

	q := database.Get().C("usuario").Find(nil)
	total, err := q.Count()
	if err != nil {
		statusIntenalServerError(ctx)
		return
	}

	err = q.Skip(pg.Skip()).Limit(pg.Top()).All(&usuarios)
	if err != nil && err != database.ErrNotFound {
		statusIntenalServerError(ctx)
		return
	}

	resp := response.NewCollection().Using(pg)

	resp.SetValue(total, len(usuarios), usuarios)

	ctx.JSON(http.StatusOK, resp)
}
```

Fonte: Elaborada pelo autor

Ao fim de seu desenvolvimento, o *Web service* foi posto em ambiente de produção utilizando-se o serviço de hospedagens em nuvem DigitalOcean, em uma máquina com 1 Gigabyte de Memória de Acesso Aleatório (RAM, do inglês *Random Access Memory*), 25 Gigabyte de armazenamento em disco de estado sólido, que utiliza o sistema operacional GNU/Linux distribuição Ubuntu versão 16.04.4, para a arquitetura x64.

O *Web service* funciona com conexões seguras através de *Transport Layer Security* (TLS), implementada na versão 1.2 desse padrão. Seus certificados são validados pela autoridade certificadora *Let's Encrypt*. Dessa forma, toda comunicação é encriptado de ponta a ponta garantindo segurança na tramitação dos dados.

Figura 4.9 – Exemplo de resposta da REST API para uma solicitação utilizando o método GET ao URI https://informe.city/api/v1.0/usuario.


```
{
  "total":1,
  "length":1,
  "value":[
    {
      "email": "admin@informe.city",
      "id": "5b6522bd5a93147egc57c7b0",
      "nome": "Administrador",
      "perfil": "admin"
    }
  ]
}
```


Fonte: Elaborada pelo autor

4.7.2 Informe Admin - Aplicação para Gerenciamento de Problemas

Seguindo os requisitos levantados, os casos de uso e a modelagem do sistema, foi desenvolvida a aplicação cliente responsável pelo gerenciamento dos problemas de infraestrutura reportados pela população.

Para seu desenvolvimento foram utilizados o *framework* Angular versão 6.0.8, com TypeScript 2.7.2. Angular foi escolhido devida as facilidades para desenvolvimento de aplicações *Web* de página única, em que toda aplicação é executada em uma mesma página HTML, tendo todas as suas operações de comunicação executadas através de chamadas HTTP assíncronas. Além disso, esse *framework* segue o modelo de três camadas. Nesse modelo há a divisão lógica e funcional dos componentes do sistema seguindo o padrão arquitetural *Model, View and Controller* (MVC). Utilizar essa abordagem na aplicação cliente ofereceu melhor organização e manutibilidade do código.

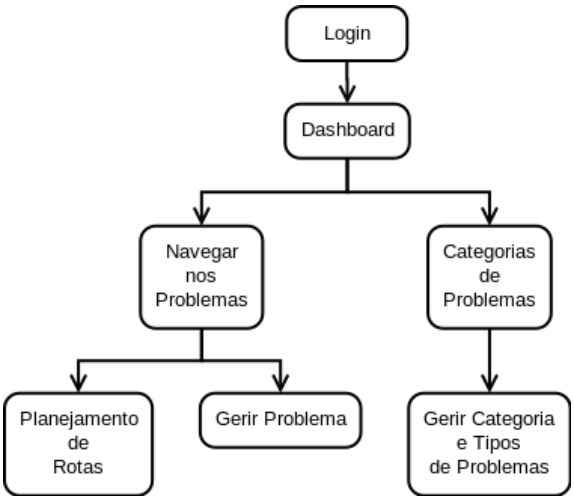
Também foi utilizado para customização da aparência da aplicação o *framework* CSS Bootstrap, versão 4.0. Bootstrap permitiu construir uma interface gráfica agradável e responsiva para a aplicação. Dessa forma, a aplicação adapta-se a diferentes tamanhos de tela, modificando a experiência de usabilidade conforme o dispositivo utilizado na navegação.

Por se tratar de uma aplicação com interface gráfica serão apresentadas as principais funcionalidades a partir dessa perspectiva.

A Figura 4.10 mostra como foram distribuídas as telas da aplicação e como se dá a navegação. Serão explanadas as principais funcionalidades seguindo o fluxo de navegação.

A Figura 4.11 apresenta a tela de *login* da aplicação. É responsável pela autenticação

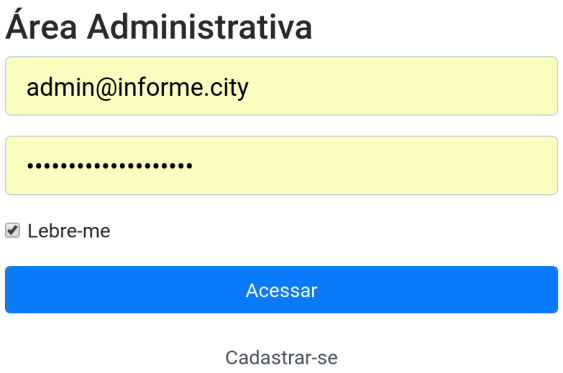
Figura 4.10 – Identificação das telas e do fluxo de navegação da aplicação para gerenciamento de problemas.



Fonte: Elaborada pelo autor

do usuário a partir da confirmação das credenciais de acesso ao sistema, o usuário administrador poderá acessá-las.

Figura 4.11 – Tela de *login* da aplicação para gerenciamento de problemas



Fonte: Elaborada pelo autor

A Figura 4.12 apresenta a tela de *dashboard* da aplicação. Essa tela contêm estatísticas úteis para o processo de gerenciamento dos problemas informados na plataforma. Possibilita que o administrador obtenha rapidamente informações sobre a situação atual dos problemas na plataforma. Foram utilizados recursos visuais para facilitar a apresentação dos dados da plataforma, através da biblioteca de gráficos dinâmicos para a *Web* da Google, conhecida como Google Charts.

O gráfico de problemas por situação mostra a relação entre os problemas que foram informados, os em andamento e os já solucionados. O gráfico problemas por categoria mostra

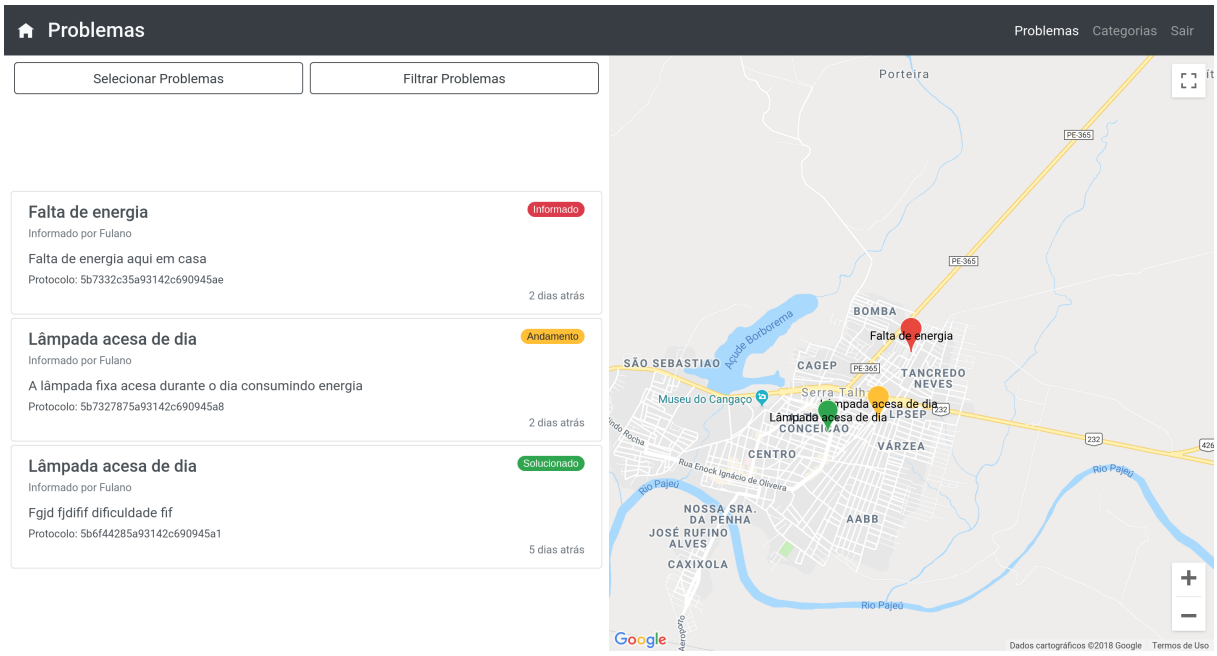
Figura 4.12 – Tela de *dashboard* da aplicação para gerenciamento de problemas.

Fonte: Elaborada pelo autor

distribuição dos problemas pelas categorias da plataforma, com isso o gestor pode identificar quais categorias se sobressaem e tomar decisões a partir dessas informações. O gráfico de problemas por tipo faz a mesmo mecanismo do anterior porém para um nível com maior granulação de informações, no caso os tipos de problemas.

Na Figura 4.13 tem-se a tela de navegação dos problemas informados na plataforma.

Figura 4.13 – Tela de navegação nos problemas informados da aplicação para gerenciamento de problemas.



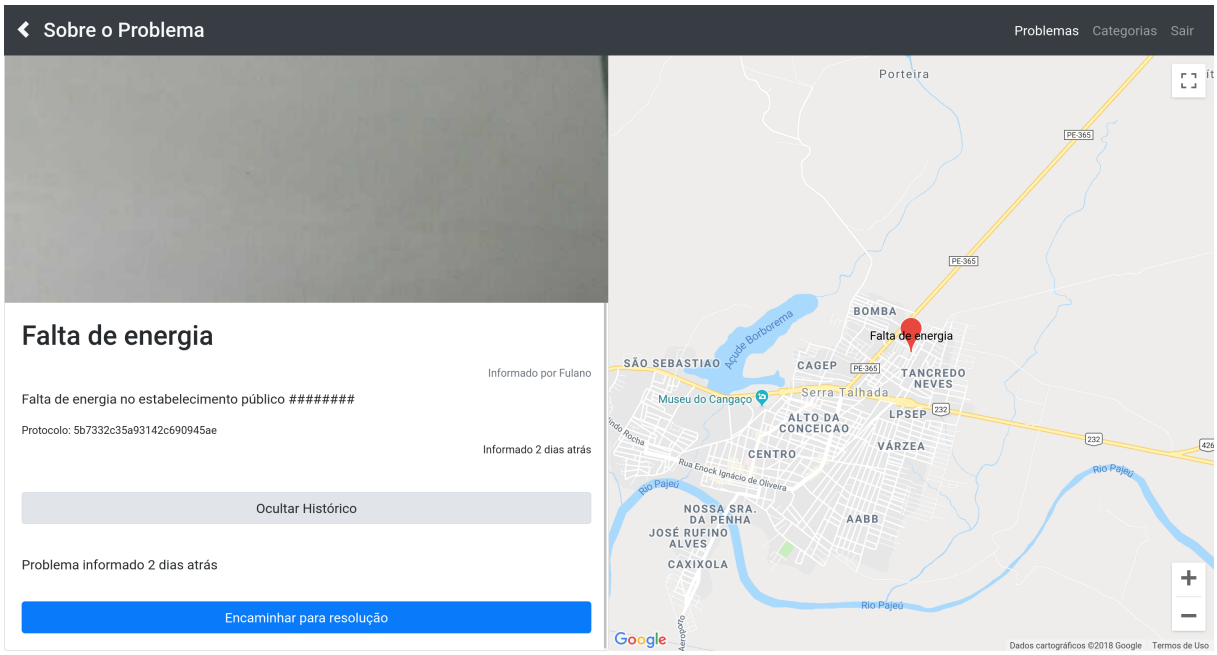
Fonte: Elaborada pelo autor

Essa figura, corresponde à principal tela de administração da plataforma, sendo o ponto de partida para as funcionalidades ofertadas relacionadas aos problemas. Podem ser observadas, no quadrante superior esquerdo, as funcionalidades de “Selecionar Problemas” e

“Filtrar Problemas”, ambas voltadas para a manipulação dos problemas visualizados nessa tela. A primeira função, “Selecionar Problemas”, serve para o usuário apontar um ou mais problemas e, a partir disso, realizar operações como o planejamento e otimização de rotas entre eles.

A segunda função, “Filtrar Problemas”, serve como mecanismo de busca pela especificação de características das quais o usuário tenha interesse. Características como a categoria, o tipo e a situação atual que encontram-se esses problemas. Dessa maneira, pode-se reduzir ou aumentar o conjunto de problemas listados para o usuário.

Figura 4.14 – Tela para gerenciar um problema da aplicação de gerenciamento

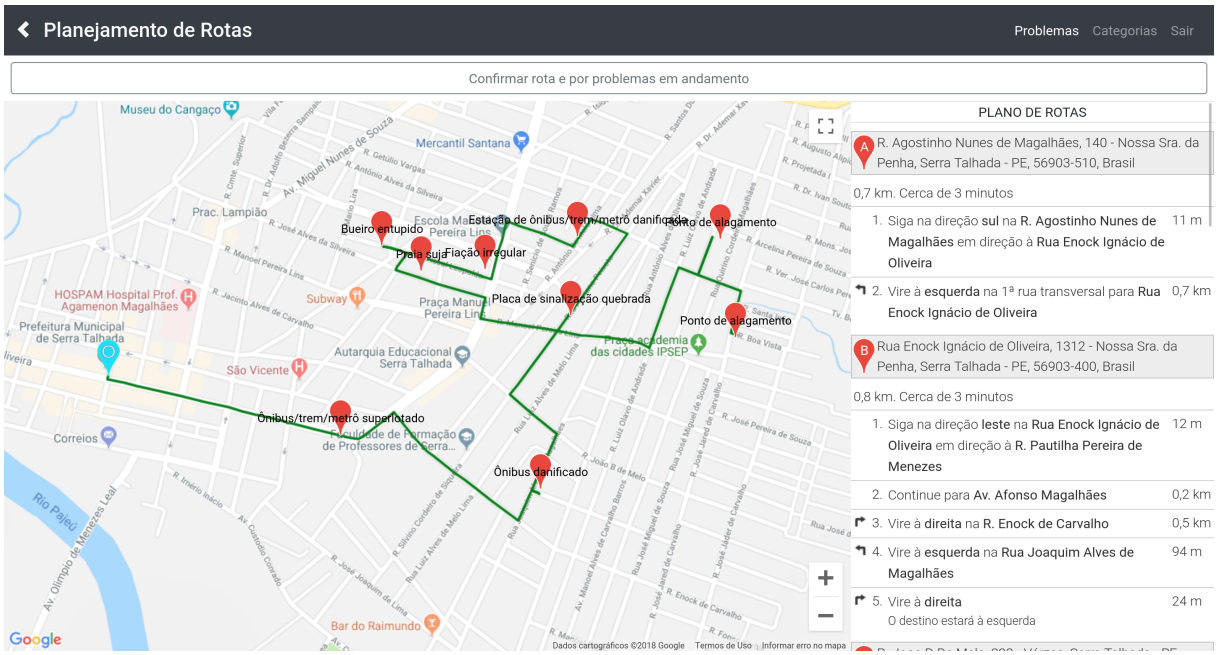


Fonte: Elaborada pelo autor

Na Figura 4.14 tem-se a tela para gerir um problema. O usuário administrador tem acesso a todos os detalhes do problema em questão. Observam-se ainda mais detalhes sobre um dado problema reportado à plataforma. Também é disponibilizado o histórico de alterações do problema, visualização da linha de tempo com as mudanças que o problema sofreu ao longo do seu processo de resolução. Dependendo da situação em que se encontra o problema, é fornecida uma opção para mudança de sua situação. Dessa forma, ao gerenciar um problema em situação de informado, há a possibilidade de encaminhá-lo para resolução. Para um problema em via de resolução, está a opção de marcá-lo como resolvido.

A Figura 4.15 demostra a tela para planejamento otimizado de rotas de acesso aos problemas. Exemplifica-se uma situação hipotética de um plano de rota otimizado em relação à distância do percurso entre um ponto de origem, na figura de localização azul e demarcado com a letra “O”, até um problema. Essa funcionalidade foi elaborada para oferecer ao governo ou

Figura 4.15 – Tela para planejamento de rotas otimizadas para acesso aos problemas da aplicação para gerenciamento



Fonte: Elaborada pelo autor

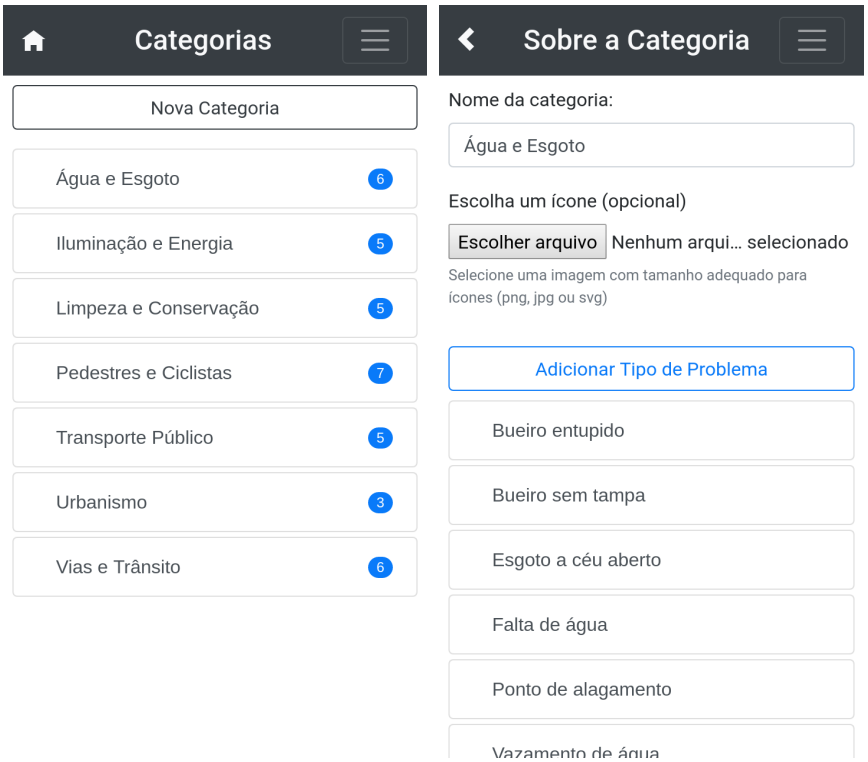
órgãos um melhor aproveitamento de recursos atrelados à logística, recursos como combustível, gasto com deslocamento de insumos e pessoal, e ainda de tempo.

Para otimização da rota, foi utilizado o serviço de roteamento da Google através de sua API JavaScript, conhecido como Google Routs. Esse serviço ofereceu a possibilidade de otimização de rotas entre pontos, é limitado a no máximo 26 pontos por solicitação. Devida a essa limitação, foram implementados procedimentos para tratar planos com mais de 26 problemas. Esses procedimentos consistem em agrupar os problemas em conjuntos que contenham até 26 itens. Em seguida, são feitas requisições para cada conjunto de problemas e os seus resultados são atrelados ao plano geral.

Na Figura 4.16 são apresentadas as telas: Gerenciamento de Categorias (Figura 4.16a) e de Gerenciamento de Tipos (Figura 4.16b). De acordo com a Figura 4.16a é possível observar a opção para adicionar uma nova categoria à plataforma. Em relação a Figura 4.16b, pode-se notar a opção para adicionar um tipo de problema a uma categoria. Essas funcionalidades permitem ao usuário administrador modificar ou adicionar categorias de problemas conforme o surgimento dessa necessidade. Nesse projeto foram cadastrados as categorias e os tipos para os problemas de infraestrutura urbana que foram levantados na Seção 4.1, conforme o método adotado.

Ainda é possível observar, na Figura 4.16, a capacidade da aplicação em se adaptar a diferentes tamanhos de telas. Ilustra-se a perspectiva de um usuário acessando a aplicação a

Figura 4.16 – Tela de gerenciamento de categorias e gerenciamento de tipos da aplicação para gerenciamento



(a) Gerenciamento de Categorias **(b) Gerenciamento de Tipos**

Fonte: Elaborada pelo autor

partir de um dispositivo móvel com tamanho reduzido de tela.

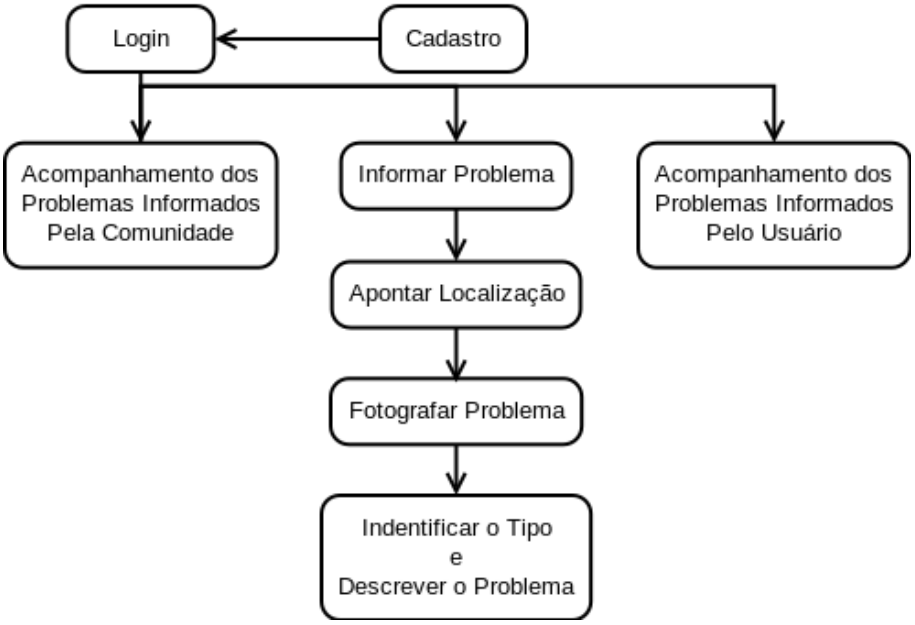
4.7.3 Informe.city - Aplicativo móvel para reportar problemas

Conforme os requisitos estabelecidos na segunda etapa do método, foi desenvolvida a plicação para dispositivos móveis que permite os cidadãos informarem problemas de infraestrutura encontrados na sua cidade. Foram utilizados para seu desenvolvimento as linguagens de programação TypeScript (versão 3.0.1) e JavaScript, versão padrão ECMAScript 2015, bem como a linguagem de marcação HTML (versão 5) e folhas de estilo em cascata CSS (versão 3).

Utilizou-se o *framework* Ionic (versão 4.0.6) para o desenvolvimento dessa aplicação. Ionic é baseado no Angular e utiliza o PhoneGap para promover o desenvolvimento de aplicativos móveis. Também disponibiliza vários outros recursos próprios, como, por exemplo, componentes e serviços adaptáveis às plataformas destino da aplicação. A seguir, identificam-se as telas que compõe o aplicativo e o fluxo de navegação.

Conforme a Figura 4.17, serão explanadas as principais telas que compõe a aplicação

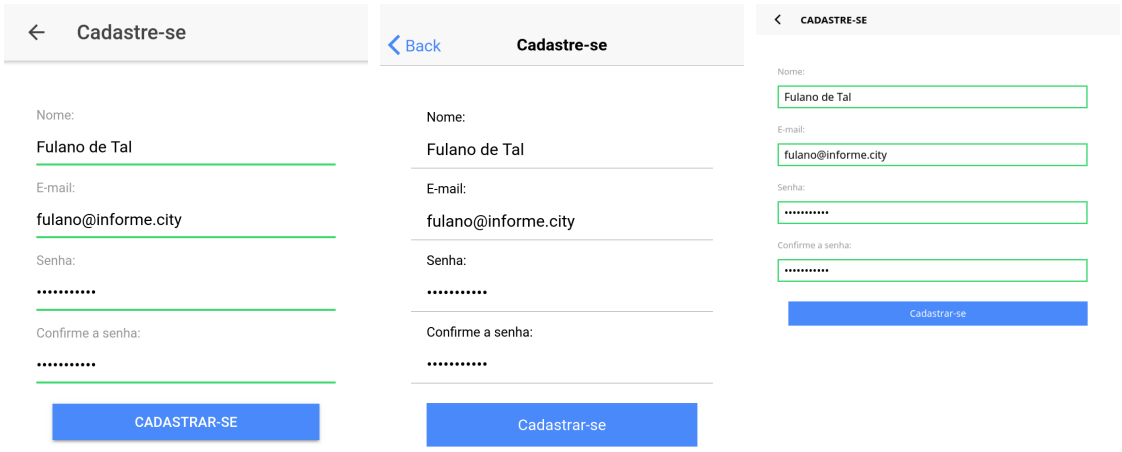
Figura 4.17 – Identificação das telas e do fluxo de navegação da aplicação para reportar problemas



Fonte: Elaborada pelo autor

desenvolvida nessa etapa. Mais adiante, é mostrada a tela de cadastro desenvolvida com as ferramentas citadas. Na Figura 4.18, tem-se a integração da aplicação à plataforma móvel que a executa.

Figura 4.18 – Tela de cadastro

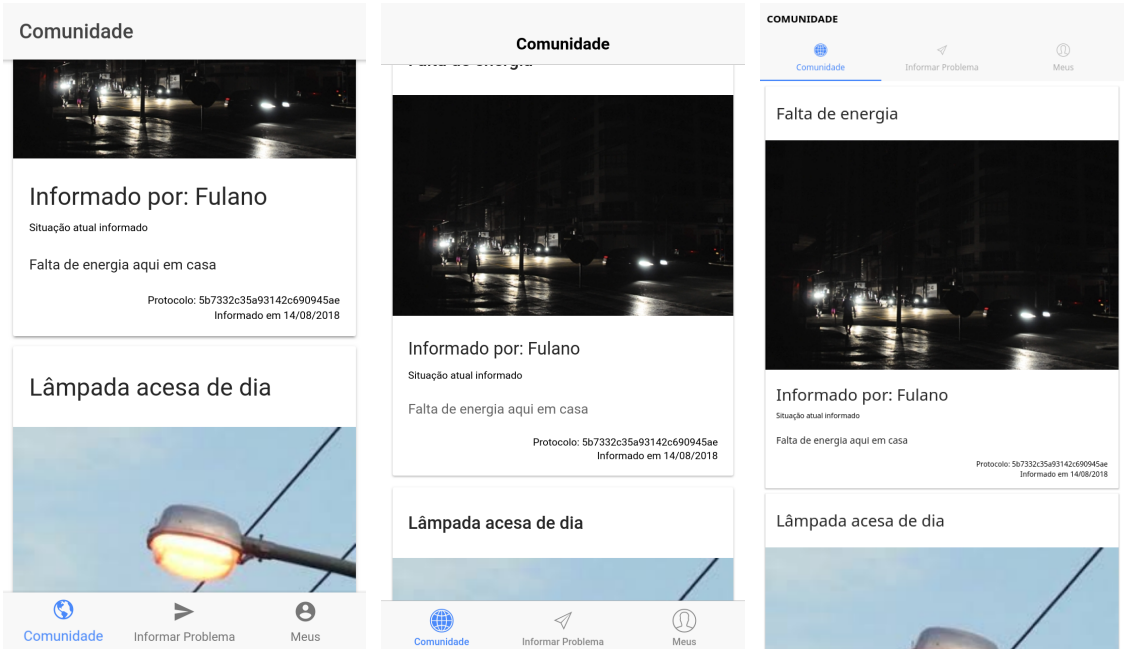


(a) Android **(b) iOS** **(c) Windows Phone**

Fonte: Elaborada pelo autor

As Figuras 4.18a, 4.18b e 4.18c ilustram as telas de cadastro de usuário em diferentes

Figura 4.19 – Tela para acompanhamentos dos problemas informados na plataforma



(a) Android **(b) iOS** **(c) Windows Phone**

Fonte: Elaborada pelo autor

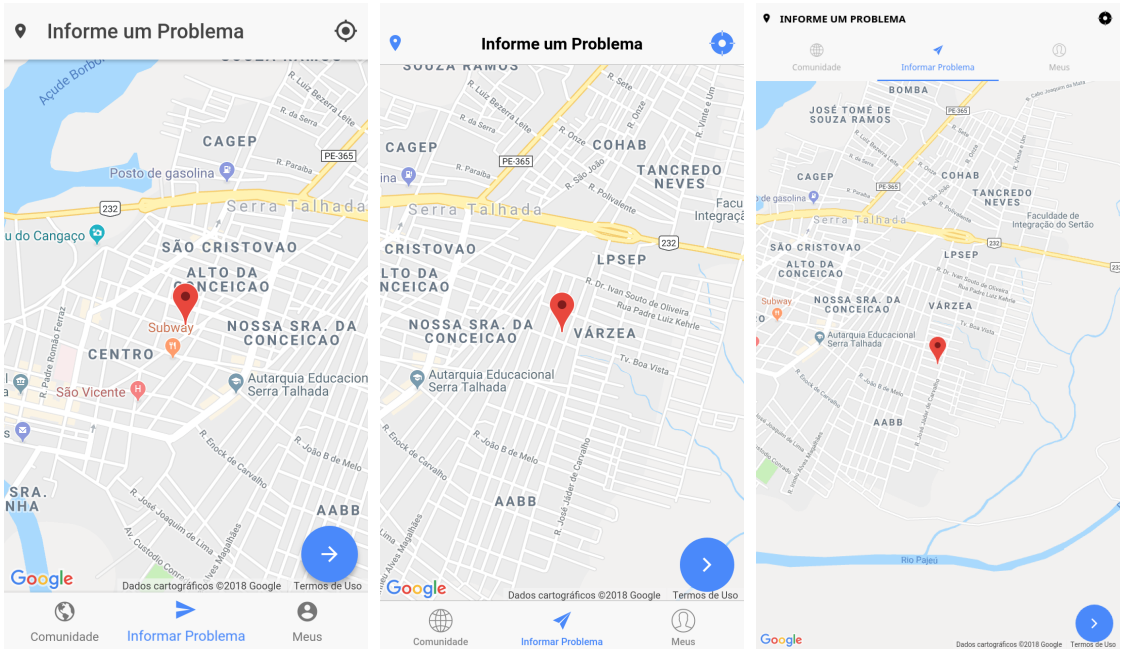
plataformas, ou seja, Android, IOS e Windows Phone, respectivamente

Ainda em relação a Figura 4.18, o cidadão pode cadastrar-se preenchendo os campos de entrada aceitos. Esses valores passam por validação para que somente valores pertencentes as classes de equivalência definidas sejam aceitos. Essas classes de equivalência foram estabelecidas na etapa de planejamento, explanada na Seção 4.4

Na Figura 4.19 tem-se a tela para acompanhamento dos problemas informados na plataforma. Na tela para acompanhamento dos problemas da comunidade, que abrange todos os problemas informados, o usuário pode ver os problemas que surgem e o seus andamentos. Para isso, observam-se as suas alteração de situação ao longo do tempo. Esse recurso permite que o cidadão tenha acesso à informações públicas de maneira rápida e prática, diferindo das abordagens clássicas. Nessas abordagens, a exemplo da telefonia, o usuário estava sujeito a burocracias que desestimulavam o acompanhamento dos problemas. Dessa forma, acredita-se que a aplicação forneça maior transparência para a população, tornando-a um agente ativo no processos de identificação e resolução de problemas, contribuindo para a cidadania conforme os princípios do m-Gov.

A tela para acompanhamento do problemas informados pelo usuário da aplicação é similar a apresentada na Figura 4.19, diferindo somente quanto aos problemas exibidos; uma vez que nela são exibidos somente os problemas informados pelo usuário do aplicativo. Essa tela de

Figura 4.20 – Tela para apontar a localização do problema a ser informado



(a) Android **(b) iOS** **(c) Windows Phone**

Fonte: Elaborada pelo autor

acompanhamento não será explanada nesse documento.

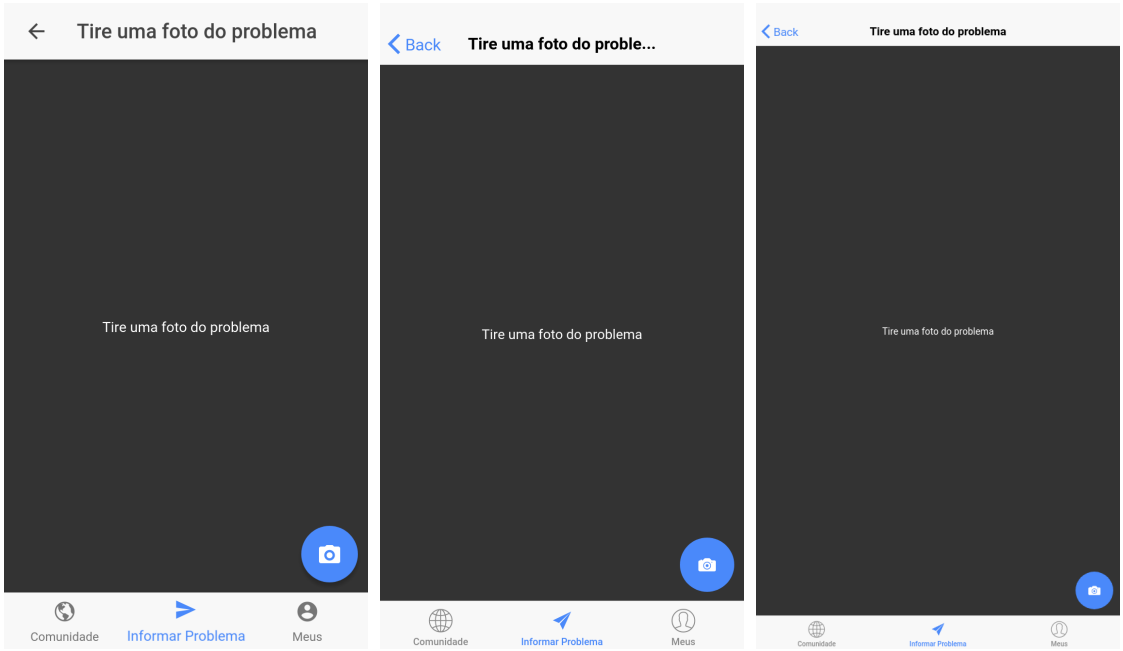
A Figura 4.20 expõe a primeira tela do fluxo responsável por informar um problema. Essa consiste na tela para apontar a localização do problema a ser denunciado. Pode-se observar duas maneiras para informar uma localização. A primeira maneira é habilitada por padrão, que consiste em receber a localização atual do problema através do GNSS do dispositivo utilizado. A segunda forma é através do apontamento manual de uma localização do mapa exibido.

Dessa maneira os dados de localização dos problemas são precisos quando comparadas às descrições verbais ou por formulário comumente utilizadas pelas abordagens clássicas para reportar problemas.

Seguindo o fluxo de navegação, Figura 4.17, há a tela para fotografar o problema, Figura 4.21. A partir dessa tela o usuário acessa a câmera do dispositivo e captura uma imagem do problema. O usuário pode ajustar a imagem recortando-a para às dimensões estabelecidas para o aplicativo. Após o envio da imagem, o usuário é redirecionado para a página de detalhamento do problema, ilustrada na Figura 4.22.

Sendo assim, o usuário pode escolher o tipo de problema que melhor corresponda à realidade. É possível observar que os problemas são distribuídos conforme suas categorias de maneira a facilitar sua identificação. Após selecionado um tipo de problema, a aplicação solicita, ainda nessa tela, que o usuário informe uma breve descrição do problema. Essa descrição serve

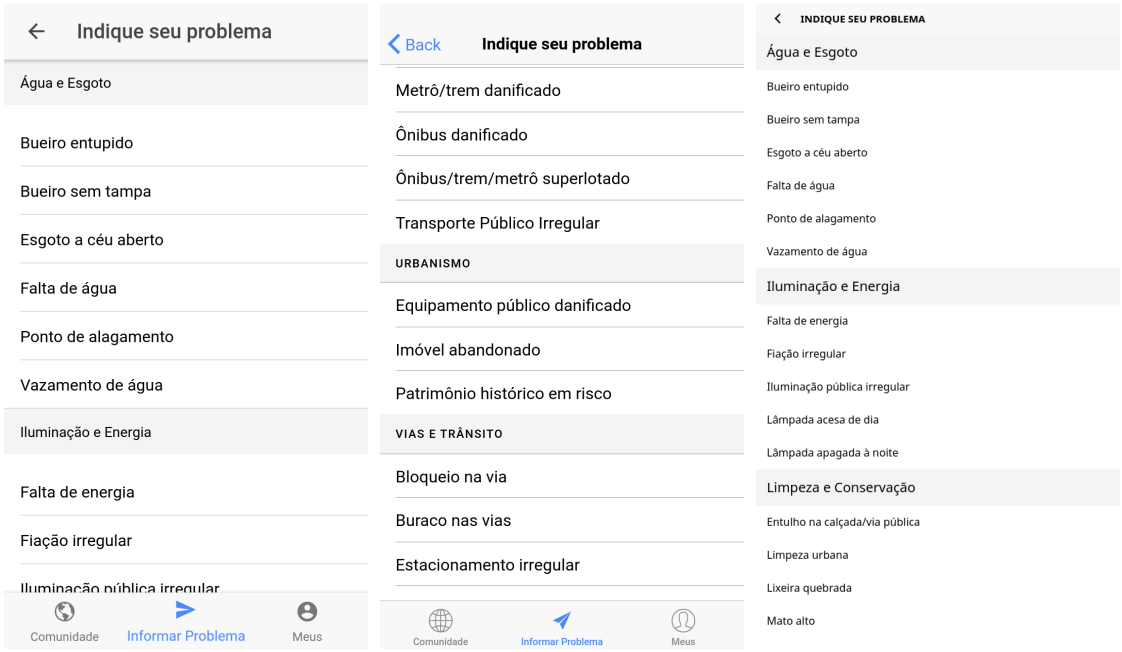
Figura 4.21 – Tela para fotografar o problema



(a) Android **(b) iOS** **(c) Windows Phone**

Fonte: Elaborada pelo autor

Figura 4.22 – Tela para descrição do problema



(a) Android **(b) iOS** **(c) Windows Phone**

Fonte: Elaborada pelo autor

para o cidadão registrar observações e informar detalhes sobre o problema em questão, assim ele tem um mecanismo de comunicação direta com o governo. Após a descrição ser informada, o sistema reporta o problema e volta para a tela de acompanhamento. Todas essas etapas estão de acordo com os requisitos e casos de uso estabelecidos para a aplicação.

4.8 Execução dos Testes e Validação da Plataforma

Essa fase do projeto consistiu na aplicação dos testes que foram definidos na terceira etapa do método, Seção 4.4. Foram executas as abordagens conforme planejado para cada uma das aplicações desenvolvidas.

4.8.1 Aplicação *Web Service*

A aplicação de *Web service* foi validada através das duas abordagens de testes conforme o planejado na etapa descrita na Seção 4.4. Utilizaram-se testes unitários para a abordagem caixa-branca e teste de carga para a caixa-preta.

4.8.1.1 Abordagem Caixa-branca

Os testes unitários foram executados ao longo do desenvolvimento da aplicação, para as principais unidades de código do sistema, utilizando-se a ferramenta oficial para testes da linguagem Go, com o comando “go test” para vários pacotes do código fonte da aplicação. Para demonstração, foi escolhido somente um caso de teste pra ser exposto nesse documento. O caso escolhido foi o mesmo apresentado na Figura 4.4 da etapa de definição e planejamento dos dos testes. A Figura 4.23, aborda o processo de execução daquele teste unitário (Figura 4.4).

Conforme a Figura 4.23, foi executado o teste da rotina de *software* responsável por criar um usuário, esse caso de teste foi descrito em detalhas na seção planejamento de testes. Nela é possível analisar que o teste foi bem sucedido e a aplicação realizou o fluxo de execução de forma correta. Nesse teste foram executados 1,6% do código do pacote “control”. A ferramenta “cover” da linguagem Go mostra visualmente as linhas de código que foram executadas nesse caso de teste em específico.

Figura 4.23 – Execução de caso de teste unitário pela ferramenta “go test”

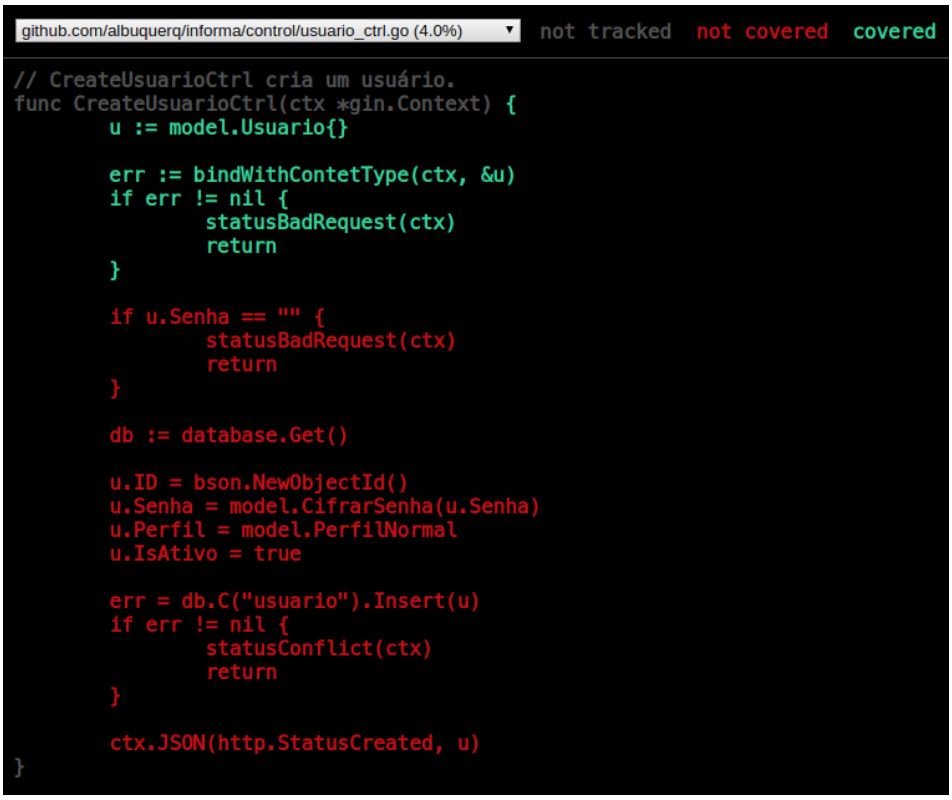
```
> go test ./control -v -run TestCreateUsuarioCtrl/NoContentType \\
-coverprofile=saida.prof

[GIN-debug] [WARNING] Running in "debug" mode. Switch to "release" mode in
production.
 - using env:   export GIN_MODE=release
 - using code:  gin.SetMode(gin.ReleaseMode)

[GIN-debug] POST   /usuario                  -->
github.com/albuquerq/informa/control.CreateUsuarioCtrl (1 handlers)
=== RUN   TestCreateUsuarioCtrl
=== RUN   TestCreateUsuarioCtrl/NoContentType
--- PASS: TestCreateUsuarioCtrl (0.00s)
    --- PASS: TestCreateUsuarioCtrl/NoContentType (0.00s)
PASS
coverage: 1.6% of statements
ok      github.com/albuquerq/informa/control    0.012s  coverage: 1.6% of
statements
```

Fonte: Elaborada pelo autor

Figura 4.24 – Análise do código coberto pelo teste unitário



Fonte: Elaborada pelo autor

A Figura 4.24 exibe o resultado do comando “go tool cover -html=saida.prof”. Foram testados os trechos de código que implementaram o fluxo de exceção estabelecido na etapa de planejamento dos teste, conforme a Figura 4.4 na fase de definição dos testes, Seção 4.4. Pode-se observar que foram testados 4% do código das rotinas que tratam do modelo de domínio “Usuário”, levantado na etapa de modelagem do sistema. Essa abordagem de teste foi executada

para as principais rotinas do *Web service*.

4.8.1.2 Abordagem Caixa-preta

Como abordagem de teste caixa-preta foi realizado o teste de carga, conforme o estabelecido na Seção 4.4, para validação da aplicação servidora. Seguindo o planejamento, foi utilizada a ferramenta Vegeta para sua realização.

O cenário elaborado para teste, avaliou a capacidade da aplicação servidora em atender às requisições das aplicações clientes no tocante ao serviço para informar problemas. Esse serviço é ofertado pela REST API através do URI "https://informe.city/api/v1.0/problema" utilizando o método POST.

Foi testado um cenário em que fosse requirida a criação de 15.000 problemas no período de 30 segundos. As requisições foram feitas a uma taxa constante de 500 requisições por segundo durante 30 segundos, considerando o município de Serra Talhada-PE, por um computador com processador Intel Core i7 modelo 2640M de 2,80 GHz com 2 núcleos, 8 Gigabytes de RAM utilizando-se uma conexão ADSL de 2 Mbit/s, para um computador servidor localizado em *San Francisco*, *California* (EUA), com processador Intel Xeon modelo E5-2650L V3 de 1,80 GHz com 1 núcleo, e 1 Gigabyte de RAM. Os resultados do teste são mostrados na Figura 4.25.

Figura 4.25 – Valores de saída da ferramenta Vegeta para o teste de carga.

```
Requests      [total, rate]            15000, 500.03
Duration      [total, attack, wait]    5m18.444671726s, 29.998001103s, 4m48.446670623s
Latencies     [mean, 50, 95, 99, max]  2m26.679466174s, 2m24.328590559s, 4m43.377289123s,
5m1.817383403s, 5m17.457121579s
Bytes In      [total, mean]            13763289, 917.55
Bytes Out     [total, mean]            8314659, 554.31
Success       [ratio]                  97.13%
Status Codes  [code:count]             201:14569  0:431
Error Set:
Post https://informe.city/api/v1.0/problema: net/http: TLS handshake timeout
Post https://informe.city/api/v1.0/problema: dial tcp 0.0.0.0:0->138.68.8.243:443: i/o timeout
```

Fonte: Elaborada pelo autor

Na Figura 4.25, observa-se o resultado mostrado pela ferramenta Vegeta. É possível verificar que o teste teve duração total de 5 minutos e 18 segundos. Desse intervalo, 30 segundos (arredondado a partir de 29,998) foram realizando as solicitações e 4 minutos e 48 segundos foram aguardando os resultados. A partir desses tempos, é possível identificar que a rede na qual foi realizado o teste possui uma latência alta. Das 15.0000 solicitações realizadas 97,13% foram bem sucedidas, apenas de 2,87% delas não foram obtidas respostas. Dessa forma foram obtidos

14569 respostas com o status 201 do HTTP (Recurso Criado) conforme a REST API e 431 sem código de status. Ao observar o conjunto de erros pode-se identificar que essas 431 respostas mal sucedidas não podem ser classificadas como erro por parte do servidor, pois foram oriundas de *timeout* de conexão, que ocorre quando o tempo da resposta ultrapassa o tempo determinado para esperá-la, de modo que o cliente abandona a conexão. Isso deve-se principalmente à latência da rede em que o teste foi realizado.

A partir da taxa de sucesso obtida, 97,13%, o *Web service* demonstrou conseguir atender demandas superiores às esperadas para um cenário real. Uma vez que, em um cenário real, a quantidade de problemas de infraestrutura identificados em uma cidade dificilmente alcançará uma taxa de 500 problemas por por segundo. Portanto, o *Web service* atendeu aos requisitos estabelecidos conforme finalidade de sua utilização e a desse projeto.

4.8.2 Aplicação para Gerenciamento de Problemas

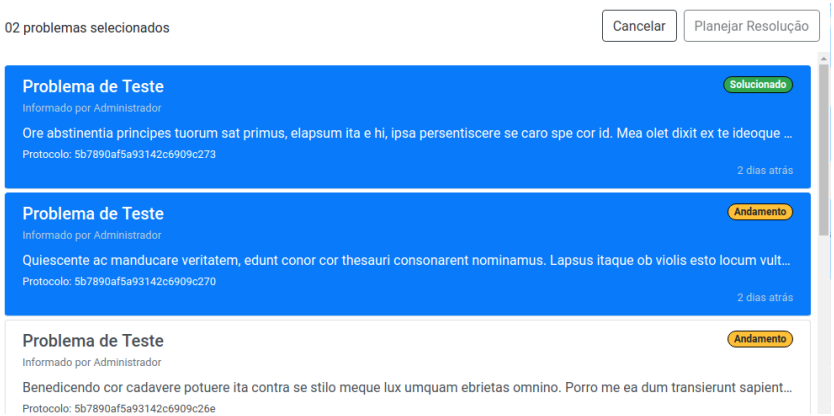
Para a validação da aplicação cliente, responsável pelo gerenciamento dos problemas reportados, foram executados os testes conforme o planejado na terceira etapa do método adotado, explanada na Seção 4.4. Esses testes foram realizados utilizando-se o navegador Chromium, versão 68.0.3440.106.

Foram realizados testes de funcionalidade em abordagem caixa-preta, os quais consistiram em seguir os casos de teste estabelecidos e acompanhar os fluxos principais, alternativos e de exceções. Foram realizados na aplicação desenvolvida observando se esses fluxos eram seguidos de acordo com o planejado. Nesse mesmo processo foram testados os valores estabelecidos pelas suas classes de equivalência para validar se a aplicação tratava as entradas corretamente, permitindo que somente valores válidos fossem aceitos.

A Figura 4.26 ilustra um exemplo de mecanismo utilizado pela aplicação para permitir somente valores válidos como entrada. Nela é representado um cenário em que existem problemas selecionados que estão em situação de “atendido” ou “em andamento” e espera-se realizar o planejamento de rotas.

Uma vez que não faz sentido planejar a resolução de um problema que encontra-se já resolvido ou em fase de resolução, a aplicação não disponibiliza essa funcionalidade para tais problemas selecionados. Somente problemas recém informados, que não foram encaminhados para resolução ou resolvidos, podem ser utilizados no planejamento de rotas. Dessa maneira,

Figura 4.26 – Mecanismo para validação de valores de entrada utilizado na aplicação para gerenciamento de problemas



Fonte: Elaborada pelo autor

atendendo os valores estipulados nas classes de equivalência.

A partir dos testes dos valores e ações de entrada, observaram-se as saídas da aplicação. Essas últimas foram comparadas às classes de equivalência para ver se correspondiam.

Com base nos seus resultados, comprovou-se que a aplicação desenvolvida comportou-se de maneira planejada, tratando as entradas de forma correta e executando seus processos conforme tal. Retornando os valores de saída corretos para cada fluxo de execução pré-determinado, em conformidade com os casos de uso e de teste estabelecidos.

Conclui-se que a aplicação para gerenciamento de problemas atende aos requisitos propostos e comprovou-se apta para ser implantada em ambiente de produção.

4.8.3 Aplicação para Reportar Problemas

Para a validação do aplicativo responsável por informar os problemas, desenvolvido para plataformas de dispositivos móveis, foram utilizados as mesmas abordagens utilizadas para o teste da aplicação para gerenciamento de problemas, conforme a Seção 4.8.2. Consistindo na execução de teste de funcionalidade seguindo os fluxos dos casos de uso levantados e os valores de entrada e saída conforme as classes de equivalência estabelecidas. A aplicação foi testada para a plataforma Android, utilizando a versão 6.0. Por limitação de recursos, as demais plataformas, iOS e Windows Phone, foram testadas utilizando-se emulador ou simulador.

Nos testes realizados, foram analisados valores e ações de entrada e saída da aplicação em relação aos estabelecidos nos casos de uso e nas classes de equivalências. Desse modo,

valores e ações foram aplicados como entrada no sistema, observaram-se as saídas da mesma conforme cada teste. Essas saídas foram comparadas as saídas planejadas.

Após essa fase, constatou-se que a aplicação desenvolvida atendeu a todos os requisitos levantados em sua concepção. Comprovando-se ser eficaz em realizar as funções a que se propõe.

4.9 Consideraçõe Finais

Nesse capítulo foram explanadas as etapas para planejamento, elaboração e validação dos sistemas que compõe à plataforma Informe.city. Na Seção 4.1 foram identificados e classificados a partir da análise dos trabalhos relacionados, pesquisas e observações, quais os tipos de problemas de infraestrutura urbana a plataforma deu suporte. Esses problemas identificados foram agrupados em categorias conforme suas similaridades temáticas. Nessa análise foi observada a possibilidade de permitir ao gestor expandir o escopo de problemas suportados pela plataforma.

Na Seção 4.2, descreveu-se a etapa para levantamento dos requisitos dos sistemas, explanando brevemente alguns dos principais requisitos levantados, fazendo distinção quanto aos subsistemas que compõem a plataforma: a aplicação para gerenciar problemas, a aplicação para reportar problemas e o *Web service*. Alguns requisitos foram importantes para oferecer funcionalidades que agregam relevância à plataforma desenvolvida, como, por exemplo, requisitos de permissividade de expansão dos tipos de problemas abordados, advindo da constatação na etapa anterior. Esse requisito abriu a possibilidade de futuro crescimento da plataforma para outras áreas de gestão além da infraestrutura urbana.

Nessa etapa de levantamento de requisitos, foram desconsideradas a abordagem de se utilizar outras mídias além de imagens na funcionalidade de reportar problemas. A possibilidade de usar mídias como áudio e vídeo em conformidade com outro trabalho relacionado foi descartada inicialmente. O principal motivo foi o fato dessa abordagem alavancar dificuldades oriundas de desafios computacionais relacionados à extração de informações a partir desses tipos de mídia e de compressão de dados. Ao utilizar áudio para informar problemas limitar-se-ia o entendimento da natureza do mesmo pela plataforma e sobrecarregaria o usuário gestor com essa tarefa. Dessa forma, o processo de reportar problemas poderia ser comprometido pela inviabilidade de superar tais desafios no cenário dessa pesquisa.

Apesar desses problemas, abordar outros tipos de mídia é uma funcionalidade que

exigirá mais estudos e análises, dessa maneira esse trabalho a sugere como trabalho futuro, dadas as limitações de tempo e escopo da pesquisa.

A Seção 4.3 tratou da etapa de planejamento dos casos de uso e das abordagens de teste. Demonstraram-se os diagramas de caso de uso simplificados, utilizando a UML, para cada um dos subsistemas que compõem a plataforma. Também foram elaboradas algumas descrições de casos de uso para fins de demonstração, expostas no Apêndice C. Na definição das abordagens de teste foi estabelecido que o *Web service* seria testado utilizando-se testes unitários através da abordagem de caixa-branca e teste de carga com a abordagem de caixa-preta. Os testes unitários foram desenvolvidos utilizando-se a abordagem de testes da linguagem de programação Go. Também foi estabelecido que o teste de carga seria realizado utilizando-se a ferramenta Vegeta, com a finalidade de garantir o devido funcionamento da plataforma em grandes cidades, onde proporcionalmente surgem mais problemas de infraestrutura urbana.

Para as aplicações de reportar problemas e a de gerenciamento foi estabelecido que seriam testadas utilizando-se teste de funcionalidade baseados nos casos de uso e nos requisitos definidos, além disso utilizando-se classes de equivalência para os valores de entrada e saída.

Na Seção 4.5, foram demonstradas as classes do domínio da aplicação, essas classes são comuns a todos os subsistemas que compõem a plataforma. As principais classes levantadas foram: Usuário, Problema, Tipo e Categoria.

A arquitetura da plataforma foi descrita na seção 4.6. Consistiu na adesão do modelo cliente-servidor em que o *Web service* funciona como aplicação servidora e os demais subsistemas funcionam como clientes. Essa arquitetura permite a interoperabilidade de sistemas através do uso da *Web API* em REST. Dessa forma, outras aplicações poderão consumir e informar dados sobre os problemas da plataforma, abrindo assim uma gama de possibilidades. Como, por exemplo, pode-se idealizar uma rede de sensores capazes de identificar problemas de infraestrutura urbana que surjam nos sistemas de infraestrutura de uma cidade. Essa rede de sensores poderia utilizar a plataforma para informar de maneira automatizada o surgimento desses problemas. Outras possibilidades idealizadas consistem na abordagem de análise dos dados, uma vez que os dados dos problemas podem ser acessados publicamente, ferramentas de análise podem agregar mais valor a esses dados utilizando-se mecanismos de Inteligência Artificial. Essas duas possibilidades em um cenário onde existem inúmeras outras.

Na Seção 4.7 é abordada a etapa do desenvolvimento da plataforma. A explanação desse desenvolvimento foi segmentada conforme os subsistemas que a compõem. Assim, o desenvolvimento do *Web service* foi explanado na subseção 4.7.1, a aplicação para gerenciamento

de problemas foi explanado na subseção 4.7.2 e, por fim, o desenvolvimento da aplicação para reporte de problemas foi abordado na subseção 4.7.3.

O desenvolvimento do *Web service* foi voltado para construir um sistema de alto desempenho e baixo consumo de recursos computacionais. Essa concepção guiou todas as escolhas e métodos no desenvolvimento, desde a linguagem de programação utilizada ao sistema de banco de dados.

Para o desenvolvimento da aplicação para gerenciamento de problemas foram utilizadas tecnologias e práticas já estabelecidas no mercado de desenvolvimento de aplicações *Web*. A escolha de utilizar uma aplicação de página única trás consigo vantagens como baixo consumo de dados, velocidade na realização das operações, uma vez que o navegador *Web* não precisa baixar uma nova página da Internet e renderizá-la, permite que as operações sejam realizadas de forma assíncrona sem comprometer o funcionamento da aplicação. Outro fato importante em relação a essa aplicação é quanto a sua capacidade de adaptação à variações de tamanho de tela, conhecida como responsividade. Essa característica oferece o benefício de acesso à aplicação a partir de dispositivos móveis, algo que pode ser positivo para o usuário.

Aplicação também forneceu um mecanismo de otimização de rotas, construído através de serviços de geoprocessamento de dados da Google, utilizados em escala global por milhares de sistemas e têm comprovada eficácia no que se propõe a realizar.

Para o desenvolvimento do aplicativo para reporte de problemas foram utilizadas ferramentas que agilizam esse processo, como apresentado na subseção 4.7.3. O uso do *framework* Ionic com o PhoneGap permitiu que uma mesma base de códigos executasse em três plataformas móveis distintas. Integrando-se a essas plataformas de maneira harmônica, fato que facilita a usabilidade da mesma para seus usuários.

A etapa de execução dos testes, explanada na Seção 4.8, foi fundamental para comprovar a eficácia dos subsistemas e da plataforma como um todo.

Como levantado nos requisitos, o *Web service* deveria ter a capacidade de atender a uma carga de requisições estipulada para uma cidade de grande porte, como constando no requisito não funcional RNF02, subseção B.2.2.2 do Apêndice B, estipulado em 200 requisições por segundo. Essa capacidade foi superada conforme a execução do teste de carga, alcançando 500 requisições por segundo no cenário abordado pelo teste. O que comprovou sua capacidade em atender as necessidades concebidas.

O cenário de teste utilizado não foi ideal para medir os limites desse sistema, um teste de carga distribuído em várias redes e com valores mais altos para as taxas de requisição

seria um ambiente mais adequado para tais mediadas. Esse possível cenário é sugerido para ser implementado em trabalhos futuros.

Para as aplicações cliente, os testes foram realizados conforme as funcionalidades e os casos de uso estabelecidos, observando-se as classes de equivalências predeterminadas. Esses testes comprovaram que ambas as aplicações clientes cumpriram os requisitos funcionais e não funcionais estabelecidos.

Por limitações do tempo da pesquisa, deixou-se de ser elaboradas outras abordagens de testes que ajudariam a identificar através estimativas mensuráveis quantitativamente qual o grau em que elas atendem os requisitos e também abordagens qualitativas para determina-se a aceitação e a usabilidade das aplicações por parte de seus usuários. Essas abordagens de teste podem ser realizas em trabalhos futuros.

No geral, ao fim do desenvolvimento da plataforma, foi observado que ela enquadrou-se nas propostas preestabelecidas em projeto. Atendeu aos requisitos estipulados e de modo abrangente é funcional podendo ser utilizada em produção.

5 Conclusão

Neste capítulo são apresentadas as considerações finais sobre o trabalho desenvolvido nesta monografia. Na Seção 5.1 estão as considerações finais. Na Seção 5.2 descrevem-se as contribuições desta monografia e na Seção 5.3 são apresentadas algumas propostas para trabalhos futuros.

5.1 Considerações Finais

Ao analisar a premissa básica da democracia, de tornar o indivíduo parte integrante e igualitária da sociedade com responsabilidade nas decisões e ações do seu Governo e os avanços alcançados nas tecnologias da informação, que diminuem cada vez mais a distância entre os indivíduos e mundo ao seu redor, permite-se contemplar a possibilidade de fortalecimento dos sistemas democráticos através da participação ativa dos indivíduos por meio do exercício de sua cidadania através de uma abordagem tecnológica.

Se levada em consideração a capacidade da tecnologia em diminuir significativamente alguns dos maiores desafios técnicos enfrentados no mundo real, governos enfrentam o desafio de tentarem identificar problemas emergentes nos sistemas de infraestrutura urbana. Pode-se estimar que soluções tecnológicas nessas áreas tenham uma relevância considerável para a sociedade.

Consoante esse cenário, a proposta apresentada nesta monografia buscou contribuir positivamente para esses dois campos, através do desenvolvimento de uma solução que viabiliza a realização dessas possibilidades advindas da tecnologia. De modo geral, esse trabalho é mais um exemplo que serve como evidência de que esse tipo de abordagem é viável para a construção de um governo mais participativo, em que o indivíduo passa a ser mais ativo nos processos governamentais.

Para o desenvolvimento desse trabalho, houve a necessidade de diminuição do escopo da abordagem temática, pois a temática de identificar e gerenciar problema é muito abrangente, mesmo quando adotada a ideia de abordar somente a área de Administração Pública e Cidadania. Advindo dessas limitações de cenário, esse trabalho abrangeu os problemas de infraestrutura urbana, deixando aberto um macro conjunto de possibilidades a serem exploradas, como, por

exemplo, problemas relacionados a Saúde, Segurança Pública e Educação.

Destaca-se a diversidade de tecnologias comercializadas hoje em dia e cada uma dotada de sistemas operacionais cujas aplicações precisam estar em conformidade. Nesse sentido, propostas que envolvem o desenvolvimento de sistemas, deveriam preocupar-se com a amplitude da utilização desses sistemas. Nesse contexto, a aplicação proposta pode ser utilizada em diversas plataformas móveis, Andoid, iOS e Windows Phone.

Essa aplicação permite que cidadãos possam utilizá-la para informar problemas de infraestrutura.

No processo de informar um problema, a plataforma desenvolvida não abordou utilizar conteúdo de mídias em formatos de áudio e vídeo como realizado por outros trabalhos relacionados. Nesse trabalho, só foram utilizados imagens e texto. Essa escolha adveio de dificuldades oriundas de desafios computacionais relacionados à extração de informações a partir daqueles tipos de mídia. A má utilização desses poderia comprometer a funcionalidade para informar problemas. Para utilizá-los deve-se realizar estudos que explorem em detalhes essa viabilidade, como, por exemplo, otimização através de compressão de dados. Dessa maneira, esse trabalho deixa em aberto esse estudo que poderá ser realizado em trabalhos futuros.

A plataforma foi testada e validada seguindo abordagens também explanadas nesse trabalho, Seção 4.8.3. Após a validação por meio dos testes, se mostrou apta a cumprir sua função e, nesse sentido, destaca-se o teste de carga aplicado.

Da execução desse teste foram obtidos resultados de desempenho acima dos estabelecido em requisito não funcional. Foi estabelecida em requisito uma taxa mínima de 200 requisições por segundo para a atividade de informar problemas. Nos resultados obtidos do teste de carga foi constatado que os sistema pôde atender a taxas de 500 requisições por segundo. Atestando-se assim sua potencialidade em cenários mais rigorosos.

Outra característica positiva está na oferta de módulos distintos, tanto para um gestor público, quanto para o cidadão.

O módulo para gerenciamento de problemas, utilizado pelo gestor, permite controlar os problemas informados, podendo planejar suas resoluções e acompanhar seus ciclos de vida, desde o momento de identificação até o momento das suas resoluções (Seção 4.7.2).

Além dessas funcionalidades, esse sistema oferece um mecanismo de otimização de rotas. Esse mecanismo foi construído através de serviços de geoprocessamento de dados da Google e permite redução de gastos por parte do governo, uma vez que os processos de logística são beneficiados pela otimização das rotas de acesso aos problemas.

Outro módulo, planejado e desenvolvido para atuar como um *Web service* (Seção 4.7.1), permite que a aplicação tenha interoperabilidade com outros sistemas. Essa funcionalidade acrescenta muitas possibilidades à plataforma no que tange à outras abordagens na utilização dos dados sobre problemas e à abrangência de sistemas que poderão interagir. Algumas dessas possibilidades foram demonstradas na Seção 4.9.

A pesar da limitação de escopo, a plataforma permite o acréscimo de outros tipos de problemas mediante os casos de uso "Manter Categoria de Problemas" e "Manter Tipos de Problemas", expostos na Seção C.2. Dessa maneira, pode-se estender a outras áreas de atuação, como, por exemplo, Saúde e Segurança Pública. Isso permitiria que ela pudesse, por exemplo, abranger problemas como focos de dengue; indicação de áreas com epidemias; infestações de vetores de doenças, como roedores; dentre outros tipos de problemas na área de Saúde Pública. Em Segurança Pública, poderia funcionar como mecanismo de denúncia para ocorrências criminosas como assaltos, roubos e venda de drogas ilícitas. Esses são alguns potenciais exemplos em um cenário maior. Apesar dessa possibilidade de expansão, nessa monografia foi abordada a área de infraestrutura urbana, pelos motivos já expostos.

Sendo assim, o conjunto dos sistemas desenvolvidos quando integrados, compõem uma plataforma de *software* que agrega um conjunto de funcionalidades que permite a divulgação, o monitoramento e o controle de problemas de infraestrutura urbana.

5.2 Contribuições da Monografia

Esse trabalho contribui com alguns artefatos gerados ao longo da pesquisa, a saber:

- Compilação de problemas de infraestrutura urbana;
- Distribuição de um aplicativo móvel, para vários sistemas operacionais, que promove a integração de participação pública em gestão governal, em especial, para problemas de infraestrutura urbana;
- Sistema administrativo para a gestão de problemas, permitindo a elaboração de plano de rotas otimizado;
- *Web sevice* em REST API que permite a interoperabilidade e integração com outras plataformas; e
- Apresentação de uma metodologia de desenvolvimento de software que pode servir de referências a futuros trabalhos acadêmicos ou até comerciais.

5.3 Propostas para Trabalhos Futuros

Visando a continuidade desta pesquisa, novas abordagens e aplicabilidades poderiam ser utilizadas, visando o enriquecimento do trabalho. Entre as possibilidades, destacam-se:

- Como exposto nas Considerações Finais, Seção 5.1, esse trabalho explanou um pequeno segmento do conjunto de possibilidades que o tema oferece. Dessa forma, incentiva-se o desenvolvimento de outros trabalhos que possam explanar um domínio maior de tipos de problemas e outras categorias, a exemplo da Saúde, Segurança Pública e Educação. Podendo abranger outros ambientes além do urbano;
- Desenvolvimento de outros métodos de otimização para os planos de resolução de problemas, visando a economicidade do dinheiro público;
- Realização de outros tipos de teste para as aplicações desenvolvidas, como: teste de estresse distribuído, testes de segurança como *Pentest* e testes de usabilidade;
- Integração da plataforma desenvolvida com sistemas automatizados de monitoramento de problemas utilizando-se a *Web API* desenvolvida;
- Análise da viabilidade para utilização de outros recursos de mídia no processo de reporte de problemas, utilizando áudio e vídeo e como extrair informações sobre a natureza do problema a partir dessas fontes;
- Desenvolvimento e aplicação de algoritmos de compressão para as mídias utilizadas na notificação de problemas; e
- Desenvolvimento de novas funcionalidades para estimular a participação social na plataforma, como, por exemplo, uma abordagem de rede social.

REFERÊNCIAS

BARONI, L. *Análise de algoritmos de navegação para um sistema GPS diferencial em tempo real*. Tese (Mestrado) — Ministério da Ciência e Tecnologia, 2006. Disponível em: <http://mtc-m17.sid.inpe.br/col/sid.inpe.br/jeferson/2004/09.21.09.01/doc/publicacao.pdf>.

BARROS, D. M. d. S.; MORAIS, P. S. G. de; PAIVA, J. C. de; LIMA, J. R. F. de; SILVA, J. L. R. da. Observatório Nacional Da Dengue - Sistema Para Monitoramento De Casos De Dengue. *Revista Brasileira de Inovação Tecnológica em Saúde ISSN: 2236-1103*, v. 3, n. 4, p. 1–14, 2014. ISSN 2236-1103. Disponível em: <http://ufrn.emnuvens.com.br/reb/article/view/3534>.

BERNERS-LEE, T. *Information management: A proposal*. W3C, 1989. 1–16 p. Disponível em: <http://www.citeulike.org/group/16652/article/5821442http://www.w3.org/History/1989/proposal.html>.

BERTEI, R. M.; PANDOLFO, A.; BARBACOVI, N. E.; MORO, L. D.; GOMES, A. P.; MORO, P. D.; BERTICELLI, R.; PANDOLFO, L. M. *Desenvolvimento de um Sistema de Informação para o Gerenciamento de Redes de Infraestrutura Urbana*. 2014. 300–313 p.

BEZERRA, E. *Princínpios da análise e projeto de sistemas com UML*. 2ª. ed. Rio de Janeiro: Elsevier, 2006. 392 p. ISBN 8535216960.

BRASIL. Constituição da República Federativa do Brasil. *Texto constitucional originalmente publicado no Diário Oficial da União de 5 de outubro de 1988.*, 1988. ISSN 10282092.

CARRIJO, C. V.; ALVARENGA, L. I. de. *Qualidade do Atendimento Prestado ao Cliente-Cidadão da Secretaria Municipal de Regulação Urbana de Aparecida de Goiânia: Um Estudo Exploratório e Propositivo*. 2011. Disponível em: <http://www.convibra.com.br/dwp.asp?id=3104{&}ev>.

CHIAVENATO, I. *Introdução À Teoria Geral da Administração*. 9ª. ed. Tamboré - SP: MANOLE, 2014. ISBN 9788520436691.

CHMIELARZ, W. Study of Smartphones Usage from the Customer's Point of View. *Procedia Computer Science*, Elsevier Masson SAS, v. 65, p. 1085–1094, 2015. ISSN 18770509. Disponível em: <http://linkinghub.elsevier.com/retrieve/pii/S1877050915028756>.

COLAB. *Colab*. 2018. Disponível em: <http://www.colab.re/>.

CONTABILIDADE, C. F. de. *Gestão pública responsável: uma abordagem do Sistema CFC/CRCs*. Brasília: [s.n.], 2011. 108 p.

DINCER, A.; URAZ, B. *Google Maps JavaScript API Cookbook*. [S.l.]: PACKT PUBLISHING, 2013. ISBN 9781849698825.

FERNANDEZ, W.; ALBER, S. *Beginning App Development With Parse and PhoneGap*. New York: Apress, 2015. ISBN 9781484202357.

FIELDING, R. T. Architectural Styles and the Design of Network-based Software Architectures. *Building*, v. 54, p. 162, 2000. Disponível em: <http://www.ics.uci.edu/{~}fielding/pubs/dissertation/top.h>.

FUNAI, M. T.; REZENDE, D. A. Avaliação da gestão integrada de serviços, processos e informações do E-Gov de uma prefeitura paranaense. *Revista Paranaense de Desenvolvimento*, n. 118, p. 67–85, 2012. ISSN 2236-5567. Disponível em: <http://www.ipardes.pr.gov.br/ojs/index.php/revistaparanaense/article/view/244/370>.

GOOGLE. *Google Maps Javascript API*. 2018. Disponível em: <https://cloud.google.com/maps-platform/maps/>.

GROVES, P. D. *Principles of GNSS, Inertial, and Multisensor Integrated Navigation Systems*. 2ª. ed. Boston, Londom: Artech House, 2013. 800 p. ISBN 978-1-60807-005-3. Disponível em: <https://books.google.com.br/books?id=t94fAgAAQBAJ{&}printsec=frontcover{&}dq=gnss{&}hl=pt-BR{&}sa=X{&}redir{_}esc=y{\#}v=o>.

GUSMÃO, G. *Criador do Colab explica sua "rede social para a cidadania"*. 2013. Disponível em: <https://exame.abril.com.br/tecnologia/criador-do-colab-explica-sua-rede-social-para-a-cidadania/>.

HOFMANN-WELLENHOF, B.; LICHTENEGGER, H.; WASLE, E. *GNSS – Global Navigation Satellite Systems: GPS, GLONASS, Galileo, and more*. New York: Springer Science & Business Media, 2007. 518 p. ISBN 3211730176, 9783211730171. Disponível em: <https://books.google.com.br/books?id=Np7y43HU{_}m8C{&}printsec=frontcover{&}dq=gnss{&}hl=pt-BR{&}sa=X{&}ved=0CC8Q6AEwAGoVChMIk67q1IeEyQIVA42QCh2Psghf{\#}v=o>.

KUROSE, J. S.; ROSS, K. W. *Rede de Computadores e a Internet: Uma Abordagem Top-Down*. 5ª. ed. São Paulo: Pearson, 2010. ISBN 978-85-88639-97-3.

LANZA, B. B. B. *Dinâmica de Relacionamento Entre Atores de Projetos Governamentais: O caso do M-GOV do Paraná*. 149 p. Tese (Mestrado) — Pontifícia Universidade Católica do Paraná, 2011.

LIMA, P. C. A. de; RIBEIRO, T. T. Mobile Government: Utililização de dispositivos móveis para aproximação do cidadão e governo. *Mobile Government: An Emerging Direction in e-Government*, n. 5, p. 155–170, 2012. Disponível em: <http://www.scopus.com/inward/record.url?eid=2-s2.0-67249089590{&}partnerID=tZOtx3y1http://repositorio.fjp.mg.gov.br/consad/handle/123456789/>.

MASCARÓ, J. L.; YAOSHINAGA, M. *Infra-estrutura urbana*. Porto Alegre: Masquatro, 2005.

MEIRELLES, F. d. S. *Tecnologia da Informação - 26ª Pesquisa Anual do Uso de TI*. [S.l.], 2015. v. 26, n. 1, 1–24 p. Disponível em: <http://eaesp.fgvsp.br/ensinoeconhecimento/centros/cia/pesquisa>.

MYER, T. *Begining Phonegap*. Indianapolis, Indiana: John Wiley & Sons, 2012. 388 p. ISBN 9781118156650.

NASCIMENTO, A. *Os 10 melhores aplicativos Windows Phone da semana - 28/05/2015*. 2015. Disponível em: <http://canaltech.com.br/dica/windows-phone/os-10-melhores-aplicativos-windows-phone-da-semana-28052015/{\#}ixzz3rFUjA>.

PEREIRA, E. H. U. *A Matemática do GPS*. 30 p. Tese (Doutorado) — Universidade Federal do Piauí, 2014. Disponível em: <http://bit.profmat-sbm.org.br/xmlui/handle/123456789/1390>.

PRESSMAN, R. *Engenharia de software*. [S.l.]: McGraw-Hill, 2006. ISBN 9788586804571.

REZENDE, D. A.; FREY, K. ADMINISTRAÇÃO ESTRATÉGICA E GOVERNANÇA ELETRÔNICA NA GESTÃO URBANA. *eGesta – Revista Eletrônica de Gestão de Negócios*, v. 1, n. 1, p. 51–59, 2005.

REZENDE, D. A.; FREY, K.; BETINI, R. C. *Governança e democracia eletrônica na gestão urbana*. 2005.

RICHARDSON, L.; RUBY, S. *RESTful Web Services*. [S.l.: s.n.], 2007. 446 p. ISBN 978-0-596-52926-0.

ROUBADO, O. F. *Onde Fui Roubado*. 2018. Disponível em: <http://www.ondefuiroubado.com.br>.

SANABIO, M. T.; SANTOS, G. J. dos; DAVID, M. V. *Administração Pública Contemporânea: Política, Democracia e Gestão*. Juiz de Fora: UFJF, 2013. 246 p. ISBN 9788576721666.

SHOTTS, K. *PhoneGap for Enterprise*. Birmingham: Packt Publishing, 2014. 192 p. ISBN 9781783554751. Disponível em: <http://css.dzone.com/articles/enterprise-android-apps>.

SIMONSEN, R. R.; PRUDENTE, P. Análise do uso combinado gps/glonass no posicionamento sob efeito de cintilação ionosférica. *Revista Brasileira de Cartografia*, v. 67, n. 1, p. 201–214, 2015. Disponível em: <http://www.lsie.unb.br/rbc/index.php/rbc/article/view/949/772>.

SOUSA, R. G. de; WRIGHT, G. A.; PAULO, E.; MONTE, P. A. do. A JANELA QUE SE ABRE: UM ESTUDO EMPÍRICO DOS DETERMINANTES DA TRANSPARÊNCIA ATIVA NOS GOVERNOS DOS ESTADOS BRASILEIROS. *Revista Ambiente Contábil*, v. 7, n. 1, p. 176–195, 2015. Disponível em: <www.periodicos.ufrn.br/ambiente/article/view/5484>.

SVENNERBERG, G. *Beginning Google Maps API 3*. New York: Apress, 2010. 329 p. ISSN 09196072. ISBN 978-1-4302-2802-8.

TANENBAUM, A. S. *Redes de Computadores*. 4. ed. [S.l.]: CAMPUS, 2003. 968 p.

TIDWELL, D.; SNELL, J.; KULCHENKO, P. *Programming Web Services with SOAP*. 1ª. ed. [S.l.]: OŔeilly, 2001. 216 p. ISBN 0-596-00095-2.

TOLENTINO, R. J. V. GPS (GLOBAL POSITIONING SYSTEM) - SISTEMA DE POSICIONAMENTO GLOBAL. *PRETEXTO*, v. 4, n. 1, p. 77–100, 2003. Disponível em: <http://www.fumec.br/revistas/pretexto/article/view/395/391>.

URBANO, P.; HORIZONTE, B. *Geoprocessamento como ferramenta de gestão urbana*. 2002.

VALKOV, I.; CHECHINA, N.; TRINDER, P. *Comparing languages for engineering server software*. 2018. 218–225 p. Disponível em: <http://dl.acm.org/citation.cfm?doid=3167132.3167144>.

VICENTE, J.; BERNARDI, E. Aplicação Do Sistema De Posicionamento Global (Gps) Na Coleta De Dados. *Dados*, p. 31, 2002. Disponível em: <http://www.rc.unesp.br/igce/aplicada/textodi.html>.

ZHONG, B. From smartphones to iPad: Power users' disposition toward mobile media devices. *Computers in Human Behavior*, Elsevier Ltd, v. 29, n. 4, p. 1742–1748, 2013. ISSN 07475632. Disponível em: <http://dx.doi.org/10.1016/j.chb.2013.02.016>.

APÊNDICE A – Categorias e Tipos de Problemas Abordados

Nesse apêndice são expostas as categorias e os tipos de problemas abordados nesse projeto pela plataforma.

- Água e Esgoto
  - Bueiro entupido
  - Bueiro sem tampa
  - Esgoto a céu aberto
  - Falta de água
  - Ponto inundação
  - Cano estourado
- Luz e Energia
  - Apagão
  - Cabos ou fios irregulares
  - Iluminação pública defeituosa
  - Lâmpada acesa de dia
  - Lâmpada queimada
- Pedestres
  - Calçada inexistente
  - Calçada irregular
  - Faixa de pedestre irregular
  - Ausência acessibilidade
- Transporte Público
  - Estação de transporte público danificada
  - Transporte público defeituoso
  - Superlotação

  - Transporte público ilegal

- Urbanismo

  - Dano em equipamento público
  - Imóvel abandonado

- Vias e Trânsito

  - Via bloqueada
  - Buraco nas vias
  - Estacionamento irregular
  - Placa de sinalização danificada
  - Semáforo danificado

APÊNDICE B – Requisitos dos Sistemas

B.1 Requisitos de Domínio

Nesta Seção é apresentado um exemplo de requisito de domínio de negócio para a plataforma.

B.1.1 [RD01] Situação do Problema

A plataforma deve fazer distinção entre problemas recém informados, problemas em solução e problemas solucionados. Essas situações correspondem ao ciclo de vida de um problema. Essa distinção servirá para oferecer um mecanismo de acompanhamento do processo de resolução do problema, objetivando facilitar esse acompanhamento para o gestor, na aplicação de gerenciamento para o cidadão e na aplicação para reportar problemas.

Prioridade: ☒ Essencial ☐ Importante ☐ Desejável

B.2 Requisitos do *Web Service* e da REST API

A seguir são mostrados os requisitos funcionais e não funcionais para o *Web service*.

B.2.1 Requisitos Funcionais

B.2.1.1 [RF01] API e Formato de Respostas

O sistema deve oferecer uma API em formato REST consumindo e retornando valores codificados em JSON. Deve permitir paginação e limitação para os recursos retornados o número de itens retornados em respostas com múltiplos itens. Esse mecanismo de paginação deve permitir que em uma resposta haja no máximo 50 itens e no mínimo 10, quando o resultado da requisição ultrapassar esses limites.

Prioridade: ☐ Essencial ☒ Importante ☐ Desejável

B.2.1.2 [RF02] Recursos Ofertados

A *Web* API deve permitir o acesso e manipulação para os dados dos seguintes recursos: Usuário, Problema, Categoria de Problema, Tipo de Problema e Media. Podendo servir: Notificação, Órgão e Localização de Implantação. O acesso aos dados deve respeitar os limites dos privilégios dos atores que interagem com a aplicação.

Prioridade: ☒ Essencial ☐ Importante ☐ Desejável

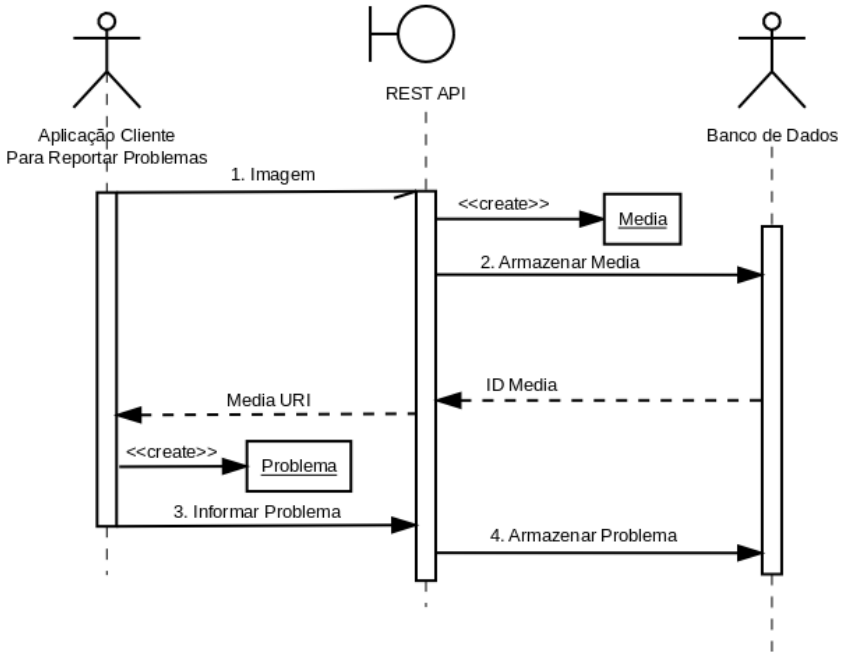
B.2.1.3 [RF03] Controle de Acesso

A aplicação deve oferecer acesso público para leitura dos seguintes recursos: Problema, Categoria de Problema e Tipo de problema, não necessitando de autenticação para tal. O mecanismo de autenticação deve ser implementado pelo próprio sistema mediante uma operação de *login* com as informações: *e-mail* e senha para o acesso.

Prioridade: ☒ Essencial ☐ Importante ☐ Desejável

B.2.1.4 [RF04] Recursos de Mídia e Arquivos Estáticos

Figura B.1 – Diagrama de sequência para o envio de uma imagem e o registro de um problema



Fonte: Elaborada pelo autor

O *Web service* deve atender aos padrões de controle de *cache* para arquivos de mídia utilizados pelos navegadores para diminuir o consumo de dados e otimizar o carregamento desses

recursos nas aplicações. Os arquivos de mídia e imagens, que forem enviados sem relação com os dados de algum problema devem ser removidos de forma automática, com um prazo de 24 horas a partir de seu envio. Evitando que se acumulem imagens de problemas que não foram informados. Esses casos acontecem quando a operação 3 do diagrama de sequência mostrado na Figura B.1 não é executada. Além disso, os arquivos de imagens devem ser armazenados no banco de dados. Por conveniência, o *Web service* deverá servir os arquivos estáticos das aplicações *Web* desenvolvidas.

Prioridade: ☐ Essencial ☒ Importante ☐ Desejável

B.2.2 Requisitos Não Funcionais

B.2.2.1 [RNF01] Acesso aos Recursos

A API deve permitir acesso *CORS Origem* para requisições. O mecanismo de autenticação deve ser por meio de chave de acesso a API para os pontos de acesso restrito. Essas chaves devem ser codificadas no formato *JSON Web Tokens*.

Prioridade: ☒ Essencial ☐ Importante ☐ Desejável

B.2.2.2 [RNF02] Desempenho de Carga

O sistema pode suportar uma carga mínima de 200 requisições por segundo. Essa carga mínima permitira que a plicação atenda à cidades de grande porte, onde o número de problemas emergentes é mais alto.

Prioridade: ☐ Essencial ☐ Importante ☒ Desejável

B.2.2.3 [RNF03] Reação à Falhas e Segurança

É importante que o sistema reaja à falhas de maneira a não prejudicar o funcionamento da plataforma por longos períodos de tempo. Para isso é desejável que o sistema se reinicialize em caso de eventuais interrupções.

Também é importante que o sistema ofereça mecanismos de segurança na comunicação através da Internet para prevenir fraudes e acesso indevido a dados sensíveis dos usuários da plataforma. Para isso ele deve utilizar o protocolo HTTP sob *Transport Layer Security* (TLS), conhecido como HTTPS.

Prioridade: ☐ Essencial ☐ Importante ☒ Desejável

B.3 Requisitos da Aplicação para Gerenciamento de Problemas

A seguir serão listados os requisitos funcionais e não funcionais para a aplicação responsável pelo gerenciamento dos problemas reportados à plataforma.

B.3.1 Requisitos Funcionais

B.3.1.1 [RF01] Apresentação das Informações Geográficas

Para melhor usabilidade e facilitar o entendimento dos dados a ferramenta para administração, deverá representar visualmente, através de mapa virtual, as localizações dos problemas da plataforma.

Prioridade: ☒ Essencial ☐ Importante ☐ Desejável

B.3.1.2 [RF02] Estatísticas

O sistema deve disponibilizar estatísticas de problemas com base nas relações: problemas por tipo, problemas por categoria e problemas por situação. Essas estatísticas apresentarão para o gestor informações importantes para a tomada de decisões. Assim, poderá ver quais tipos de problema são mais reportados pela população e também como estão distribuídos na plataforma os problemas em relação às suas situações no processo de resolução.

Prioridade: ☐ Essencial ☒ Importante ☐ Desejável

B.3.1.3 [RF03] Otimização de Rotas

O sistema deve calcular uma rota de navegação otimizada que percorra os problemas recém informados. Essas rotas aparecerão como um plano para ser percorrido em processos como fiscalizações e para a resolução desses problemas. A otimização em relação à distância percorrida aparece como uma solução que busca economizar custos nesses processos. Esses custos estão relacionados ao transporte e a logística envolvida nas atividades para resolução de problemas de infraestrutura urbana.

Prioridade: ☐ Essencial ☒ Importante ☐ Desejável

B.3.1.4 [RF04] Expansão da Abordagem de Problemas

Observando a análise e o levantamento dos tipos de problemas abordados inicialmente pela plataforma, foi identificada a possibilidade de expansão de tipos de problemas conforme o surgimento dessa necessidade. Dessa maneira, o sistema deve permitir o cadastro de novas categorias e de novos tipos de problemas.

Prioridade: ☐ Essencial ☐ Importante ☒ Desejável

B.3.1.5 [RF05] Navegação nos Problemas

O sistema deve possibilitar que o gestor filtre entre todos os problemas os que ele tiver interesse. Esse filtro deve ser através dos atributos categoria, tipo e situação. Também podem permitir filtro por meio do raio de localização em relação a um ponto e por usuário informante do problema.

Prioridade: ☒ Essencial ☐ Importante ☐ Desejável

B.3.2 Requisitos Não Funcionais

B.3.2.1 [RNF01] Responsividade

O sistema deve apresentar interface responsiva para vários tamanhos de telas, compatíveis com dispositivos como *smartphones*, *tablets*, *notebooks* e *desktops* com permissão de acesso à *Web*.

Prioridade: ☐ Essencial ☐ Importante ☒ Desejável

B.4 Requisitos da Aplicação para Reporte de Problemas

Aqui são litados os requisitos funcionais e não funcionais para a aplicação responsável por reportar problemas para a plataforma.

B.4.1 Requisitos Funcionais

B.4.1.1 [RF01] Registro de Usuários

O sistema deve permitir cadastro de usuários. O usuário deve entrar com as suas informações de conta para ter acesso às funcionalidades do sistema. As informações devem ser: nome, *e-mail* e a senha para acesso ao sistema.

Prioridade: ☒ Essencial ☐ Importante ☐ Desejável

B.4.1.2 [RF02] Facilidade para Identificação de Problema

O sistema deve possibilitar ao usuário selecionar o tipo de problema de infraestrutura que deseja reportar. A aplicação deve buscar intuitividade do usuário nessa seleção uma abordagem de apresentar os tipos de problemas em categorias, agrupados por similaridade temática.

Prioridade: ☒ Essencial ☐ Importante ☐ Desejável

B.4.1.3 [RF03] Apontamento de Localização do Problema

O sistema deve possibilitar ao usuário escolher como informar a localização do problema, optando entre apontar uma localização geográfica manualmente ou utilizar a localização do dispositivo através do GNSS.

Prioridade: ☐ Essencial ☐ Importante ☒ Desejável

B.4.1.4 [RF04] Descrever Problema

O sistema deve permitir ao usuário acrescentar uma descrição em texto às informações para envio de problema. Além de texto, o sistema deve permitir o envio de imagem do problema, servindo de mecanismo antifraude, pois uma imagem real deve ser capturada a partir do dispositivo que está notificando o problema.

Prioridade: ☐ Essencial ☒ Importante ☐ Desejável

B.4.1.5 [RF05] Acompanhamento de Problema

O sistema deve permitir ao usuário acompanhar um problema a partir do momento em que for informado, através da mudança na sua situação, que pode ser informado, andamento

ou solucionado. Essas mudanças devem ser registradas em forma de histórico para posterior verificação. Esse histórico é uma linha do tempo onde são informados os momentos das alterações e uma breve descrição, permitindo a transparência do processo.

Prioridade: ☒ Essencial ☐ Importante ☐ Desejável

B.4.1.6 [RF06] Publicidade e Transparência

O sistema deve permitir ao usuário visualizar os problemas recentes informados por outros usuários da sua cidade. Isso permite a ciência de problemas existentes na sua cidade e possa acompanhar o andamento dos processos de resolução dos problemas de uma forma ampla, transmitindo um senso de cidadania participativa.

Prioridade: ☒ Essencial ☐ Importante ☐ Desejável

B.4.2 Requisitos Não Funcionais

B.4.2.1 [RNF01] Plataformas Móveis

O sistema deve funcionar em *smartphones* nas plataformas Android, iOS e, se possível, Windows Phone.

Prioridade: ☐ Essencial ☐ Importante ☒ Desejável

APÊNDICE C – Descrição de Casos de Uso

C.1 *Web Service*

A aplicação de servidora (*Web service*) é apresentada na Tabela C.1 a descrição do caso de uso “Cadastrar Problema”, também é exposta a descrição do caso de uso “Autenticar Usuário” (Tabela C.2).

C.2 Aplicação para Gestão dos Problemas

Para fins de demonstração, apresenta-se uma descrição de caso de uso para a aplicação responsável por gerenciar os problemas relatados na plataforma. Essa descrição é exposta na Tabela C.3 e descreve o caso de uso “Alterar Situação do Problema”.

C.3 Aplicação para Reporte de Problemas

Para a aplicação de gerenciamento de problemas, apresenta-se a descrição do caso de uso “Informar Problema” (Tabela C.4) e os casos de uso incluídos, a saber: “Apontar Localização” (Tabela C.5), “Anexar Imagem” (Tabela C.6) e “Descrever Problema” (Tabela C.7).

Tabela C.1 – Descrição do caso de uso Cadastrar Problema

Caso de Uso:	Apontar Localização
Sumário:	O *Web service* cadastra um novo problema na base de dados.
Ator Principal:	Cliente com privilégio para cadastrar problemas.
Precondição:	O cliente ter estabelecido uma conexão HTTP com o *Web service*.
Fluxo Principal:	FP01. O cliente envia uma requisição HTTP utilizando o método POST com as informações do problema codificadas no formato JSON; FP02. Incluir e executar o caso de uso Autenticar Usuário; FP03. O *Web service* checa a validade das informações apresentadas. Caso as informações estejam erradas, o sistema executa o fluxo de exceção FE01; se corretas, o sistema cadastra o problema na base de dados e retorna as informações do problema codificadas no formato JSON e com o código de status 201 do HTTP.
Fluxo de Exeção	FE01 [FP03] Informações inválidas 1. O sistema interrompe o fluxo principal, retorna a requisição com o código de status 400 do HTTP e fecha a conexão.
Pós-condição:	O problema foi cadastrado com sucesso.

Fonte: Elaborada pelo autor

Tabela C.2 – Descrição do caso de uso Autenticar Usuário

Caso de Uso:	Autenticar Usuário
Sumário:	O *Web service* checa se o cliente tem a permissão necessário para seguir o fluxo de execução.
Ator Principal:	Aplicação cliente do *Web service*.
Precondição:	O cliente ter estabelecido uma conexão HTTP com o *Web service*.
Fluxo Principal:	FP01. O cliente informa a chave de acesso. FP02. O sistema extrai as informações do cliente com base na chave de acesso. Em seguida verifica se o cliente tem permissão. Caso o cliente não tenha permissão o sistema executa o fluxo alternativo FA01. Caso seja autorizado, o sistema continua o fluxo de execução no caso de uso que incluiu o atual.
Fluxo Alternativo	FA01 [FP02] Acesso negado 1. O sistema identifica que o cliente não tem permissão de acesso. Interrompe o fluxo de execução que o incluiu e retorna para o cliente o código 403 do HTTP fechando a conexão estabelecida.
Pós-condição:	O usuário foi autenticado e autorizado com sucesso.

Fonte: Elaborada pelo autor

Tabela C.3 – Descrição do caso de uso Alterar Situação do Problema

Caso de Uso:	Alterar Situação do Problema
Sumário: **Ator Principal:** **Precondição:**	O administrador alterará a situação do problema. Governo/Órgão (Usuário Administrador) O usuário administrador estar logado no sistema.
Fluxo Principal:	FP01. O sistema apresenta as informações do problema, que são: tipo do problema, situação, imagem do problema, descrição, identificador único, data da criação do problema, e histórico das alterações; FP02. O administrador seleciona a opção desejada para alterar a situação do problema, escolhendo uma ação, que pode ser "Encaminhar para Resolução" (FA01) ou "Problema Concluído" (FA02).
Fluxo Alternativo	FA01 [FP02] Encaminhar para Resolução 1. O sistema muda a situação do problema para "Andamento" e registra essa mudança no histórico do problema. FA02 [FP02] Problema Concluído 1. O sistema muda a situação do problema para "Solucionado" e registra essa mudança no histórico do problema.
Pós-condição:	A situação do problema foi alterada com sucesso.

Fonte: Elaborada pelo autor

Tabela C.4 – Descrição do caso de uso Informar Problema

Caso de Uso:	Informar Problema
Sumário: **Ator Principal:** **Precondição:**	O usuário informará um problema de infraestrutura. Usuário (Cidadão). O dispositivo estar conectado a internet e o usuário estar logado no sistema.
Fluxo Principal:	FP01. O usuário seleciona informar problema; FP02. Incluir e executar o caso de uso: Apontar Localização; FP03. O sistema reporta o problema.
Pós-condição:	O problema foi reportado com sucesso.

Fonte: Elaborada pelo autor

Tabela C.5 – Descrição do caso de uso Apontar Localização

Caso de Uso:	Apontar Localização
Sumário: **Ator Principal:** **Precondição:**	O usuário (Cidadão) selecionará a localização do problema. Usuário (Cidadão). Executado o passo 1 do fluxo principal do caso de uso Informar Problema (Tabela C.3).
Fluxo Principal:	FP01. O sistema mostra o local atual do dispositivo conforme o GNSS; FP02. O usuário aponta um local desejado ou confirma a localização indicada; FP03. Incluir e executar o caso de uso: Anexar Imagem.
Fluxo de Exceção:	FE01 [FP01] Sem permissão de acesso ao GNSS 1. O sistema reporta o erro e solicita permissão de acesso ao GNSS do dispositivo; 2. O usuário permite o sistema acessar o GNSS do dispositivo; 3. O sistema volta para o fluxo principal (FP01).
Pós-condição:	A localização do problema foi apontada.

Fonte: Elaborada pelo autor

Tabela C.6 – Descrição do caso de uso Anexar Imagem

Caso de Uso:	Anexar Imagem
Sumário: **Ator Principal:** **Precondição:**	O usuário (Cidadão) fotografa o problema. Usuário (Cidadão). Execução bem sucedida do caso de uso Apontar Problema (Tabela C.5).
Fluxo Principal:	FP01. O sistema exige que o usuário retire uma foto do local do problema; FP02. O usuário fotografa o problema utilizando a câmera do dispositivo; FP03. O sistema exige confirmação da fotografia retirada; FP04. O usuário confirma; FP05. Incluir e executar o caso de uso: Descrever Problema.
Pós-condição:	A imagem do problema foi enviada.

Fonte: Elaborada pelo autor

Tabela C.7 – Descrição do caso de uso Descrever Problema

Caso de Uso:	Descrever Problema
Sumário:	O cidadão seleciona o tipo de problema que será informado, e descreve textualmente o problema.
Ator Principal:	Usuário (Cidadão).
Precondição:	Execução bem sucedida do caso de uso Anexar Imagem (Tabela C.6).
Fluxo Principal:	FP01. O sistema mostra os tipos de problemas por categoria; FP02. O usuário seleciona um tipo de problema; FP03. O sistema exige que o usuário escreva uma descrição para o problema; FP04. O usuário descreve o problema e informa que concluiu.
Pós-condição:	O tipo do problema foi escolhido e a descrição informada.

Fonte: Elaborada pelo autor