UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Guilherme Pereira de Souza

Central de Gerenciamento de Controle de Tráfego

Uberlândia, Brasil

2017

UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Guilherme Pereira de Souza

Central de Gerenciamento de Controle de Tráfego

Trabalho de conclusão de curso apresentado à Faculdade de Computação da Universidade Federal de Uberlândia, Minas Gerais, como requisito parcial exigido à obtenção do grau de Bacharel em Sistemas de Informação.

Orientador: Rafael Pasquini

Universidade Federal de Uberlândia – UFU

Faculdade de Ciência da Computação

Bacharelado em Sistemas de Informação

Uberlândia, Brasil

2017

Guilherme Pereira de Souza

Central de Gerenciamento de Controle de Tráfego

Trabalho de conclusão de curso apresentado à Faculdade de Computação da Universidade Federal de Uberlândia, Minas Gerais, como requisito parcial exigido à obtenção do grau de Bacharel em Sistemas de Informação.

Rafael Pasquini
Orientador

Luis Fernando Faina

Rodrigo Sanches Miani

Uberlândia, Brasil
2017

Agradecimentos

Agradeço primeiramente àquela que sempre esteve ao meu lado me apoiando e me tornando um reflexo de sua bondade, educação e dedicação, minha querida mãe. Àquela que me acolheu e apoiou na dedicação aos estudos, minha querida avó Ilda; Ao meu padrinho Edvaldo pelo apoio e incentivo a querer seguir a área da computação; A minha madrinha Rosana por sempre estar dísponivel a compartilhar o seu conhecimento, ajudando do vestibular até o presente trabalho de conclusão de curso. Ao meu professor e orientador Rafael Pasquini por todo ensinamento e motivação necessária que foi dada; E por fim, aos grandes amigos que fiz neste período de faculdade e que sempre me alegraram e incentivaram.

"Vamos pensar o impensável, vamos fazer o impossível. Vamos nos preparar para lidar com o próprio inefável, e ver se não podemos expressá-lo depois." - Douglas Adams

Resumo

Este trabalho busca explicar o desenvolvimento de uma central de gerenciamento de controle de tráfego da arquitetura de gestão de sinalização de trânsito proposta por Everton Rocha Lira em sua dissertação de mestrado. O trabalho consiste em mostrar o escopo do sistema, trabalhos relacionados, os materiais e métodos utilizados, a modelagem do sistema e uma apresentação do mesmo. Visando ao final, unir-se ao desenvolvimento proposto no mestrado em arquitetura de gestão de sinalização de trânsito e criar a arquitetura como um todo.

Palavras-chave: *Arquitetura para Gerenciar a Sinalização de Trânsito, Central de Gerenciamento de Controle de Tráfego, Transmissor de Sinal de Trânsito, Receptores de Unidades de Veículos*

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

Ajax	*Asynchronous JavaScript e XML*
C-ITS	*Cooperative Intelligent Transport Systems* Inteligente
CSS	*Cascading Style Sheets*
HTTP	*HyperText Transfer Protocol*
IPv4	*Internet Protocol version 4*
ITS	*Intelligent Trasportation System*
JSON	*JavaScript Object Notation*
LED	*Light Emitting Diode*
RFID	*Radio-Frequency IDentification*
SGBD	Sistema Gerenciador de Banco de Dados
SO	Sistema Operacional
TCMC	*Traffic Control Management Center*
TCP/IP	*Transmission Control Protocol/Internet Protocol*
TSMA	*Traffic Sign Management Architecture*
TST	*Traffic Sign Transmitters*
URL	*Uniform Resource Locator*
V2I	*Vehicle-to-Infrastructure*
V2V	*Vehicle-to-Vehicle*
VANET	*Vehicular Ad hoc Networks*
VURs	*Vehicular Unit Receivers*
VMS	*Variable Message Sign*
XHTML	*eXtensible HyperText Markup Language*
XML	*eXtensible Markup Language*

Sumário

1 Introdução

Uma das causas de mortalidade em alta evidência no Brasil é referente a acidentes envolvendo veículos automotivos, causa esta que, segundo (SIVAK, 2014), é responsável por 3.5% do total de mortes contabilizadas anualmente no Brasil e 2.1% em escala global. Para isto, os estudos mais recentes apontam como um bom investimento a melhoria e atualização da forma convencional de apresentação da sinalização aos condutores, reduzindo assim a taxa de mortalidade (ZHANG T.; DELGROSSI, 2012).

Atualmente a forma convencional de apresentação da sinalização de trânsito aos condutores é por meio de placas fixas de metal, adesivadas com uma sinalização de trânsito, conforme podemos ver na Figura 1 que mostra a sinalização de "Pare" em diversos países. Essa forma de apresentação da sinalização, embora relativamente funcional e prática, é problemática pois, além de não ser possível alterar a sinalização exibida sem ter o esforço manual de substituir a placa presencialmente, ainda existem diversas causas para o não entendimento do motorista em relação as sinalizações de trânsito.

Uma destas causas está relacionada ao aumento da longevidade da população. (GRACA, 1986) aponta que um dos três principais motivos de acidentes iniciados por pessoas de idade avançada está relacionado a dificuldades na interação com a sinalização de trânsito.

Figura 1 – Sinalização de "Pare" em diversos países.

Adaptado de: "Various stop signs found around the world.". Disponibilizado em: <http://images.roadtrafficsigns.com/img/art/stopsigns-L.jpg>, acesso em Jun. 2017

Outra causa evidente envolvendo o modelo atual de sinalização de trânsito é o aumento na dificuldade de enxergar as placas durante o período noturno, fator que, apesar de ser mais presente em pessoas com problemas de visão, afeta toda a população em menor ou maior grau (PETERMAN, 2013).

Mas a causa principal ou, no mínimo, um dos maiores motivadores de acidentes automobilísticos é a desatenção do condutor. Em 2006 foi divulgado um estudo sobre o papel da desatenção como causa de acidentes, nele constavam os resultados de um experimento envolvendo motoristas de ambos os gêneros e diversas faixas de idade. Constatou-se que o fator desatenção, em suas diversas formas, fez parte das causas em 78% dos acidentes automobilísticos (KLAUER et al., 2006).

Para isto, foi proposto em (LIRA et al., 2016), uma arquitetura para gerenciar a sinalização de trânsito (*TSMA - Traffic Sign Management Architecture*), tendo em vista dar suporte a uma alteração dinâmica da sinalização exibida nas placas de trânsito. Isso é feito por meio de um projeto que fornece aos profissionais responsáveis pela gestão do tráfego a possibilidade de atualizar remotamente os dispositivos que cumprem a função de uma "placa de trânsito" no TSMA.

Como mostrado na Figura 2, existem 3 componentes essenciais para a arquitetura:

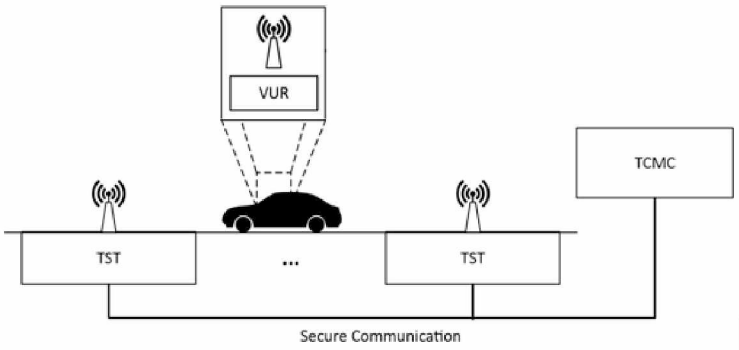

Figura 2 – Comunicação da Sinalização de Trânsito no TSMA.

Fonte: (LIRA et al., 2016)

- **TCMC:** A Central de Gerenciamento de Controle de Tráfego define as mensagens de sinalização rodoviária, e, através de um canal de comunicação seguro, envia mensagens de sinalização rodoviária para os Transmissores de Sinal de Tráfego (*TST-Traffic Sign Transmitters*) em sua área de atuação;

- **TST:** Os Transmissores de Sinal de Tráfego são o equivalente as "placas de trânsito" na forma convencional e são responsáveis por transmitir essas mensagens de

sinalização rodoviária através da tecnologia Wi-Fi para os Receptores de Unidade Veiculares (*VURs- Vehicular Unit Receivers*);

- **VUR:** O Receptor de Unidade Veicular recebe as mensagens de sinalização rodoviária e as apresenta para o motorista.

Este trabalho visa dar enfoque no desenvolvimento da Central de Gerenciamento de Controle de Tráfego (*TCMC*) proposta na Arquitetura para Gerenciar a Sinalização de Trânsito (*TSMA*) definida por (LIRA et al., 2016).

1.1 Objetivos

Os objetivos gerais deste projeto são:

- Desenvolver uma interface gráfica web da central de gerenciamento de controle de tráfego (TCMC);
- Desenvolver um sistema controlador da central de gerenciamento de controle de tráfego (TCMC);
- Permitir comunicação da sinalização entre a central de gerenciamento de controle de tráfego (TCMC) com o transmissor de sinal de trânsito (TST).

Os objetivos específicos deste projeto são:

- Apresentar um mapa virtual que exiba a posição de cada um dos dispositivos TST localizados na região de cobertura de um TCMC;
- Permitir a visualização, inclusão, alteração e exclusão das localidades, sinalizações, redes e atribuições dos TSTs.

1.2 Justificativa

Parte dos trabalhos nesta temática de sistemas de transporte inteligente tem o objetivo de prover informações ao motorista para ajudá-lo a tomar decisões seguras ou até mesmo diminuir a atuação do elemento humano no trânsito. É importante ressaltar que a grande maioria dos trabalhos relacionados à sinalização de trânsito é direcionada a propor formas de melhorar o controle e apresentação dos semáforos. Assim, poucos são os trabalhos que tendem a dar enfoque na explicação e desenvolvimento de uma central de gerenciamento de controle de tráfego que permite atualizações remotas da infraestrutura de sinalização.

1.3 Metodologia

Neste trabalho é proposto o desenvolvimento de uma interface web para o recebimento dos parâmetros da localização e sinalização dos TSTs pelos operadores internos. Assim como o gerenciamento (Inserção, Leitura, Atualização e Exclusão) dos dados recebidos no banco de dados. Posteriormente será desenvolvido um sistema controlador que fará a comunicação entre TCMC e TST via socket TCP/IP, passando a atual sinalização do transmissor.

1.4 Resultados Esperados

Este trabalho espera mostrar o funcionamento de uma central de gerenciamento de controle de tráfego através de um protótipo, além de junto com alguns dispositivos raspberry, testar a arquitetura de gestão de sinalização de trânsito. Tudo está sendo realizado em conjunto com o trabalho de mestrado de (LIRA et al., 2016) que implementa as interações dos TSTs para os VURs , ao final, espera-se mostrar um protótipo, juntando os dois trabalhos e apresentando a arquitetura como um todo.

1.5 Organização do Trabalho

Este trabalho é organizado em capítulos, dos quais este é o primeiro e apresenta a ideia do sistema, incluindo seu objetivo e justificativa. O Capítulo 2 apresenta a revisão bibliográfica referente a trabalhos relacionados ao TSMA no âmbito do TCMC. No Capítulo 3 estão os materiais e métodos utilizados para a modelagem e implementação do sistema. O Capítulo 4 apresenta o desenvolvimento deste trabalho, mostrando o escopo, a modelagem, a apresentação e a implementação do sistema desenvolvido. No Capítulo 5 são apresentados as conclusões do trabalho. Por fim, são apresentados as referências bibliográficas e códigos da implementação do TCMC no anexo.

2 Revisão Bibliográfica

Segundo (DOWNS, 2004) há uma forte tendência de que o número de congestionamentos, entre outras consequências do aumento na densidade do tráfego urbano, continue a aumentar. Algumas das formas de minimizar estas consequências, bem como aumentar a segurança e praticidade de atividades relacionadas ao trânsito vem dos Sistemas de Transporte Inteligentes (ITS - *Intelligent Trasportation System*). Os ITS descrevem a tecnologia aplicada aos transportes e infraestrutura para transferir informações entre sistemas, visando maior segurança, produtividade e desempenho ambiental. Isso inclui aplicações autossuficientes como sistemas de tráfego de gestão, informação e sistemas de alerta instalados em veículos individuais, bem como a Cooperação de Sistemas de Transporte Inteligente (C-ITS - *Cooperative Intelligent Transport Systems*) que envolvem comunicações entre veículo e infraestrutura (V2I - *Vehicle-to-Infrastructure*) e comunicações entre veículo e veículo (V2V - *Vehicle-to-Vehicle*). Cada uma das seções a seguir contém a descrição de um trabalho envolvendo tópicos que estão relacionados ao TSMA, ao qual faremos comparações com a utilização da central de gerenciamento de controle de tráfego proposta neste trabalho.

2.1 VMS - *Variable Message Signs*

Conforme explicado em (LIRA et al., 2016), os painéis de mensagens variáveis, são painéis que utilizam predominantemente a tecnologia LED (*Light Emitting Diode* - Diodo Emissor de Luz) para a exibição de textos ou sinalizações de trânsito. Um exemplo de painel de mensagens variáveis pode ser visto na Figura 3.

A vantagem deste tipo de exibição é bem próxima do TSMA, pois ambas permitem a atualização remota da sinalização de trânsito via uma central de gerenciamento. A principal diferença desta forma de exibição para o TSMA é a comunicações entre veículo e infraestrutura que não é definida no VMS. Além disto, no modelo TSMA, optou-se pela utilização da tecnologia E-ink para a exibição da sinalização, visto que o consumo energético desta tecnologia é bem menor e o seu custo é mais barato do que a tecnologia LED.

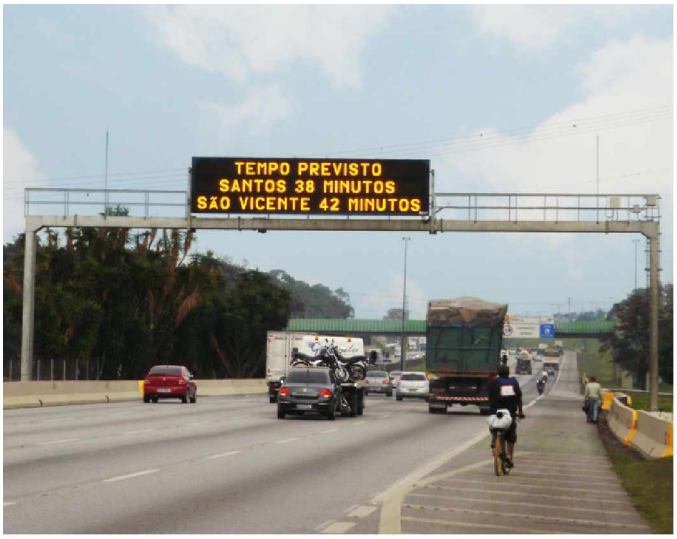

Figura 3 – Exemplo de painel de mensagens variáveis.

Fonte: https://goo.gl/mWdEGh

2.2 RFID - *Radio-Frequency IDentification*

Já na proposta de (SATO; MAKANAE, 2006), além de manter a placa física, o motorista também verá a sinalização através de um dispositivo veicular com tela. Esta proposta tem como tecnologia principal o RFID (*Radio-Frequency IDentification*) ou Identificação por Radiofrequência, que é uma tecnologia sem fio destinada a coleta de dados via ondas de rádio. Um sistema RFID contem 4 componentes básicos, são eles:

- Transpônder ou Etiqueta (Tag): É no microchip deste componente que são armazenados os dados para coleta. Além disto, a etiqueta também possui uma antena que permite o recebimento de requisições e envio dos dados;
- Antena de leitura: Detecta as etiquetas via radiofrequência, gerando assim um meio de comunicação entre a etiqueta e o leitor;
- Transceptor ou Leitor: Este componente é responsável por processar os dados recebidos da antena e envia-los ao controlador;
- Controlador: Aplicação que realiza o tratamento dos dados recuperados da etiqueta.

Portanto, conforme podemos ver na Figura 4, a implementação de (SATO; MAKANAE, 2006) consistem armazenar dados da sinalização em uma etiqueta RFID próxima da placa física. Para a leitura desta etiqueta, uma antena de leitura RFID deverá ser instalada na parte de baixo do veículo. Assim, quando uma etiqueta entrar no alcance da antena, esta irá gerar um meio de comunicação entre etiqueta e leitor. O leitor por sua

vez irá processar os dados da sinalização e encaminhar para o controlador exibi-la na tela do dispositivo veicular.

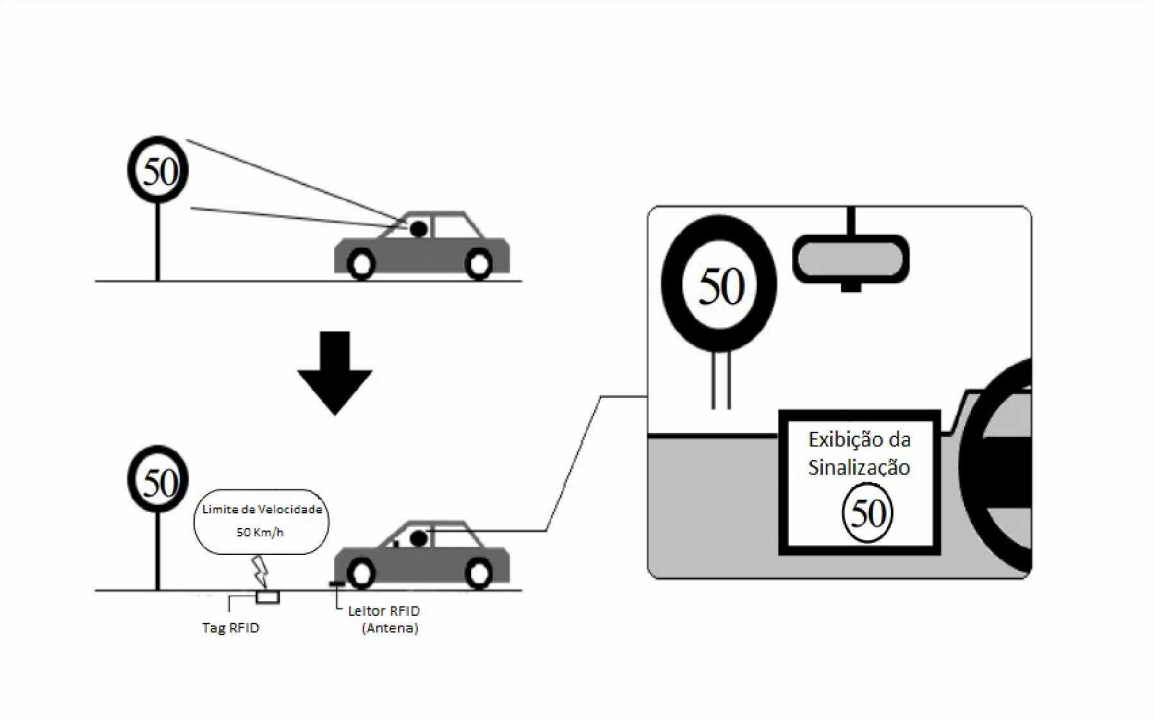

Figura 4 – Funcionamento da proposta utilizando sistema RFID.

Adaptado de: (SATO; MAKANAE, 2006)

As vantagem desta implementação é comparável com o TSMA no que diz respeito a comunicação entre veiculo e infraestrutura. Porém, no sistema de (SATO; MAKANAE, 2006), haverá um esforço manual de substituir a etiqueta RFID presencialmente caso seja necessário alterar a sinalização. Já na arquitetura TSMA, é previsto a central de gerenciamento de controle de tráfego (TCMC) justamente para realizar a alteração remota da sinalização.

2.3 VANET - *enabled in-vehicle traffic signs*

A grande maioria das pesquisas sobre o desenvolvimento de melhores dispositivos de controle de tráfego concentra-se em semáforos, enquanto muito poucos trabalhos falam sobre o assunto específico de melhorar os sinais de trânsito comuns. (FERNANDES, 2009) é um dos poucos trabalhos que propõem um sistema baseado em GPS que constantemente varre o local atual de um determinado veículo através de redes ad hoc veiculares (VANETs - *Veicular Ad hoc Networks*) e compara a sua posição relativa a uma base de dados de sinais de trânsito incorporado em uma unidade de veículos. Ou seja, o trabalho de (FERNANDES, 2009) vem da ideia principal de que os sinais de tráfego em veículos

podem ser alcançados através da colaboração dos veículos locais, criando sinais virtuais de trânsito e de modo sincronizado, mas sem uma infraestrutura de controle centralizado. Já na proposta de arquitetura de gestão da sinalização de trânsito (TSMA) de (LIRA et al., 2016), existe uma infraestrutura de controle centralizado, a Central de Gerenciamento de Controle de Tráfego (TCMC), que gera mensagens de sinal de trânsito que são enviados para os Transmissores de Sinal de Tráfego (TST), e por sua vez, os TSTs transmitem as mensagens de sinalização rodoviária para os Receptores de Unidade Veicular (VUR) localizados no interior dos veículos circulantes. Como resultado, a abordagem TSMA é mais dinâmica, uma vez que o TCMC é capaz de modificar as mensagens de sinalização rodoviária do TST, sempre que necessário, enquanto que em (FERNANDES, 2009), cada veículo deve ter a sua base de dados atualizados a fim de reagir a mudanças na infraestrutura.

2.4 Comparação

A Tabela 1 apresenta uma comparação dos trabalhos relacionados com a arquitetura do TSMA referente a prever a existência de uma central de gerenciamento de controle de tráfego (TCMC).

Tabela 1 – Comparação entre os trabalhos relacionados ao TSMA.

	VMS	SATO	VANETs	TSMA
Prevê atualização remota da sinalização?	Sim	Não	Sim	Sim
Mantém placa de trânsito externa?	Sim	Sim	Não	Sim
Recebimento da sinalização no veículo?	Não	Sim	Sim	Sim
Prevê central de gerenciamento de controle? (TCMC)	Sim	Não	Não	Sim

3 Materiais e Métodos

Este capítulo apresenta os materiais e métodos utilizados para a realização deste trabalho. Os materiais estão relacionados às tecnologias e ferramentas utilizadas e o método apresenta a sequência das principais atividades realizadas.

3.1 Materiais

A Tabela 2 apresenta as ferramentas e tecnologias utilizadas para a criação do TCMC.

Tabela 2 – Ferramentas e tecnologias utilizadas no desenvolvimento do TCMC.

Ferramentas e Tecnologias	**Versão**	**Finalidade**	**Referência**
StarUML	2.8	Criação de diagramas	http://staruml.io/
Bizagi	3.1	Criação da modelagem de processos do projeto	https://www.bizagi.com/pt
Notepad++	7.4	Ambiente de desenvolvimento	https://notepad-plus-plus.org
WampServer	3.0	Aplicação que instala o ambiente de desenvolvimento web (PHP, Apache, Mysql e PHPMyAdmin)	http://www.wampserver.com/en/
PHP	5.6	Linguagem de programação da interface gráfica	http://php.net/
Apache	2.4	Servidor Web	https://httpd.apache.org/
Mysql	5.7.14	Gerenciador de banco de dados	http://www.mysql.com/
PhpMyAdmin	4.6	Para gerenciamento do banco de dados	https://www.phpmyadmin.net/

XHTML	1.0	Linguagem de marcação utilizada para produzir páginas na web	https://www.w3schools.com/html/html_xhtml.asp
JQuery	1.7	Biblioteca JavaScript de código aberto	http://jquery.com/
CSS	3	Linguagem de estilo utilizada para definir a apresentação de documentos escritos em uma linguagem de marcação	http://www.w3schools.com/css/
Cupcake Theme	1.4	Template utilizado na interface gerencial web	https://themeforest.net/item/cupcake-premium-admin-template-mobile-theme
Google Chrome	59.0.3	Navegador utilizado para visualização da interface web	https://www.google.com/chrome
C++	Padrão 14	Linguagem de programação do controlador	http://www.cplusplus.com/
Code::Blocks	16.1	IDE de desenvolvimento C++	http://www.codeblocks.org/
GNU GCC Compiler	3.4	Compilador C++	http://gcc.gnu.org/
Raspberry Pi	1 Modelo B e 2	Ferramenta utilizada para os testes do VUR e TST respectivamente	https://www.raspberrypi.org/
Tightvncserver	1.3.9	Aplicação para gerenciamento remoto do raspberry pi	http://www.tightvnc.com
VNC Viewer	6.1.1	Ferramenta que permite a visualização da tela do raspberry pi em outro computador	https://www.realvnc.com

Sistema operacional windows	8.1	SO utilizado no desenvolvimento do TCMC	https://www.microsoft.com/pt-br/windows/
Sistema operacional raspbian	4.9.28	SO utilizado na criação do componente do servidor TST e testes dos componentes do TST e VUR criados por (LIRA et al., 2016)	https://www.raspbian.org/

3.1.1 Ajax

Ajax (*Asynchronous JavaScript e XML*) é um conjunto de técnicas de JavaScript e XML(*eXtensible Markup Language*) (entre outras tecnologias) nomeada por (GARRETT et al., 2005) que permite deixar a experiência do navegador com o usuário mais interativa.

No modelo clássico de aplicação web a comunicação entre cliente e servidor é baseado em um sistema síncrono, ou seja, a interface do usuário envia uma solicitação HTTP(*Hypertext Transfer Protocol*) e espera a resposta do servidor, este então processa a solicitação e retorna uma página HTML. A Figura 5 mostra de forma visual o processo deste modelo.

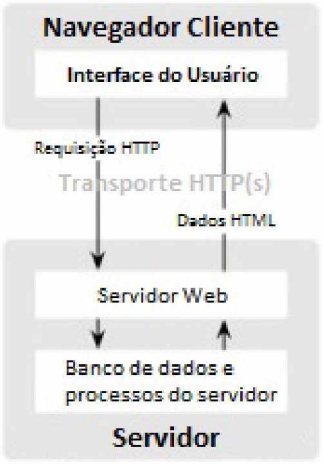

Figura 5 – Modelo clássico de aplicação web.

Adaptado de: (GARRETT et al., 2005)

Já no modelo de aplicação utilizando Ajax, também chamado de modelo interativo, a comunicação entre cliente e servidor web é baseada em um sistema assíncrono, pois a ferramenta Ajax é inserida como uma nova camada entre a interface do usuário e o servidor web, ou seja, o Ajax é ativado através de uma chamada JavaScript pela interface do usuário e envia uma solicitação HTTP ao servidor, este então processa a solicitação e retorna os dados em XML (ou outras formas de formatação como JSON (*JavaScript Object Notation)* e texto simples). O ajax pode então processar os dados recebidos e carregar apenas a parte pertinente a solicitação na interface do usuário. A Figura 6 mostra de forma visual o processo do modelo utilizando Ajax.

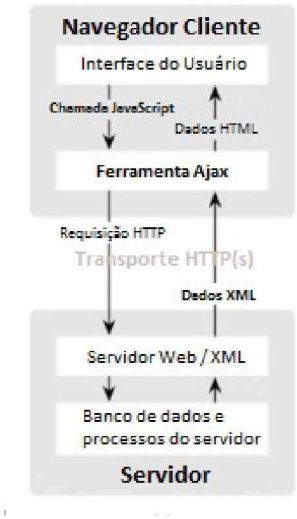

Figura 6 – Modelo Ajax de aplicação web.

Adaptado de: (GARRETT et al., 2005)

Na implementação do TCMC foi utilizado a técnica Ajax com a biblioteca Jquery para tratar melhor a experiência do usuário. Com ela foi possível:

- Utilizar a API do google maps para mostrar a localização dos TSTs na área do TCMC;
- Exibir as mensagens de erro e sucesso dos formulários sem ter que recarregar a página.

3.2 Métodos

As etapas para a modelagem e a implementação da central de gerenciamento de controle de tráfego seguiram o modelo sequencial linear proposto por (PRESSMAN, 2005). A utilização desse modelo é justificado porque o sistema é simples e uma versão inicial

dos requisitos do mesmo já foram pré-definidos na dissertação de mestrado de (LIRA et al., 2016). As etapas definidas foram:

1. **Levantamento dos requisitos**: O levantamento dos requisitos iniciou com o orientador fornecendo a visão geral do sistema já pré-definida no inicio da dissertação de mestrado de (LIRA et al., 2016), mostrando os principais conceitos envolvidos na definição das funcionalidades pretendidas para o sistema. Dessa visão foram extraídos os principais requisitos e outros foram identificados posteriormente;

2. **Análise e projeto**: Com os requisitos levantados, foi modelada a arquitetura inicial e a base de dados do TCMC;

3. **Implementação**: Na implementação, inicialmente foi definida a base de dados. A criação da base MySql foi realizada por meio da ferramenta Phpmyadmin. Posteriormente, a implementação da interface gerencial web foi criada. Por fim, o controlador do TCMC foi implementado;

4. **Testes**: Os testes foram realizados pelo autor deste trabalho visando identificar erros de codificação e testes para verificar se as funcionalidades de cada um dos componentes foram devidamente implementadas.

4 Desenvolvimento

Este capítulo apresenta o resultado deste trabalho que é a implementação de uma central de gerenciamento de controle de tráfego (TCMC).

4.1 Escopo do Sistema

Na arquitetura TSMA, o TCMC é "uma entidade local que é responsável pela gestão de todos os TSTs, dispositivos equivalentes às atuais placas de trânsito sob a ótica do TSMA, em sua área de atuação predefinida. Os TCMCs são previstos como sendo extensões do órgão nacional de trânsito"(LIRA et al., 2016). Ou seja, é nesta entidade que será gerenciada as sinalizações, localizações e redes de cada um dos TSTs em uma determinada área. Assim como os órgãos municipais de trânsito gerenciam as placas de trânsito convencionais, os operadores internos do TCMC gerenciam os TSTs de sua respectiva abrangência.

Esta área de abrangência do TCMC não é definida por limites municipais, mas, sim, conforme for mais conveniente para o órgão nacional de trânsito. Ou seja, dependendo da quantidade de TSTs ou funcionários disponíveis, um município pode ter um ou mais TCMCs em sua área. A única recomendação feita por (LIRA et al., 2016) é que não haja sobreposição entre as áreas de atuação caso o município tenha dois ou mais TCMCs.

O TCMC é separado em 3 componentes principais, são eles:

- **Interface Gerencial Web**: É através deste componente que ocorre a interação entre homem e computador, ou seja, é através da interface gerencial web que o operador poderá realizar a implantação e manutenção dos TSTs;
- **Banco de dados**: Este componente é responsável por armazenar os dados recebidos pelo operador através da interface gerencial web;
- **Controlador**: Este componente tem como responsabilidade verificar se há alguma nova atribuição de TST na base de dados, e caso houver, realizar a comunicação com o determinado TST via socket TCP.

  Podemos verificar os processos de cada um dos componentes do TCMC, TST e VUR na Figura 7. É importante ressaltar que o desenvolvimento deste trabalho é definido pelos componentes "Interface Gerencial Web", "Banco de Dados", "Controlador" e "Servidor TST". Os componentes "Módulo TST" e "Módulo VUR" foram desenvolvidos por (LIRA et al., 2016) e seus detalhes podem ser verificados na dissertação de mestrado do mesmo.

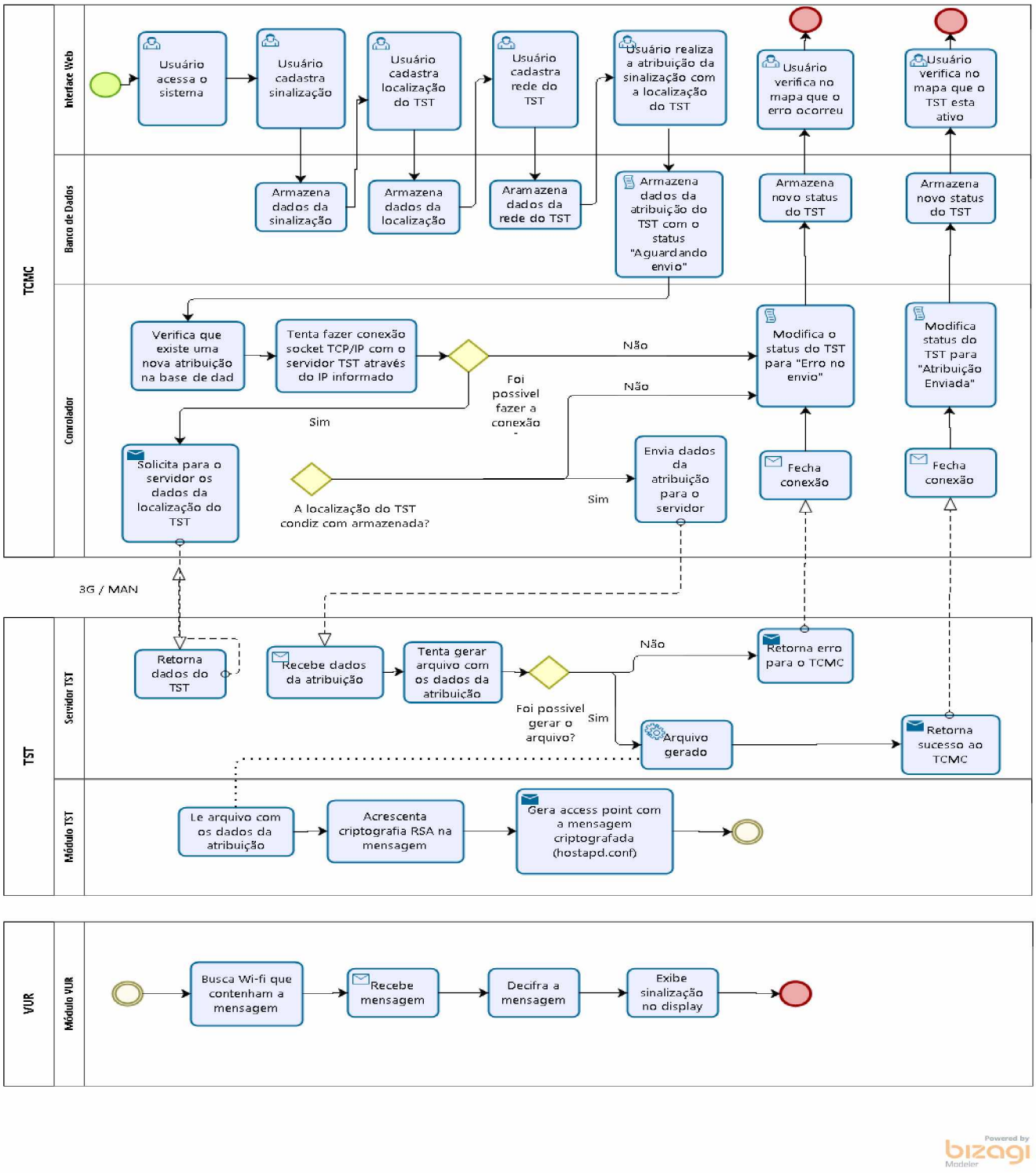

Figura 7 – Fluxograma de processos do TSMA.

4.1.1 Interface Gerencial Web

A interface gerencial web é uma aplicação web desenvolvida utilizando tecnologia XHTML(*eXtensible Hypertext Markup Language*), CSS(*Cascading Style Sheets*) e JavaScript no lado do cliente e linguagem de programação PHP(*PHP: Hypertext Preprocessor*) no lado do servidor.

O lado do cliente web da interface gerencial do TCMC foi desenvolvida sob a lógica do modelo interativo descrito na seção 3.1.1. Portanto, as solicitações HTTP da interface do usuário para o servidor web são realizadas de forma assíncrona, possibilitando uma atualização parcial da página.

4.1.2 Banco de Dados

O banco de dados do TCMC foi desenvolvido utilizando o sistema gerenciador de banco de dados (SGBD) MySQL visto a fácil utilização do mesmo com as linguagens de programação da interface gerencial (PHP) e da linguagem de programação do controlador (C++).

Além disto, em termos de autenticação de banco de dados, o MySQL fornece bons mecanismos para garantir que somente usuários autorizados têm acesso para o servidor de banco de dados.

4.1.3 Controlador

Os módulos do TST e VUR implementados por (LIRA et al., 2016) foram criados na linguagem de programação C++, portanto, para manter a padronização e eventuais manutenções, o componente do controlador do TCMC foi criado utilizando a mesma linguagem de programação.

É através deste componente do TCMC que é realizado a comunicação com o TST via socket TCP/IP.

4.2 Modelagem do Sistema

A Tabela 3 representa os requisitos funcionais existentes para o desenvolvimento do TCMC.

Tabela 3 – Requisitos funcionais do sistema.

Código	**Nome**	**Requisito**
RF001	Manter Usuário	**COMO** Administrador,
		QUERO ter a possibilidade de incluir, visualizar, alterar e excluir usuários do sistema,
		PARA gerenciar os acessos ao TCMC.
		Um usuário possui: ID, Nome, Sobrenome, E-mail, Senha e Nível de Permissão
RF002	Manter Sinalização	COMO Administrador ou Operário,
		QUERO ter a possibilidade de incluir, visualizar, alterar e excluir as sinalizações no sistema,
		PARA gerenciar as sinalizações ativas ou não do TCMC.
		Uma sinalização possui: Código, Caminho da imagem, Nome da imagem e Tipo
RF003	Manter Localização	**COMO** Administrador ou Operário,
		QUERO ter a possibilidade de incluir, visualizar, alterar e excluir as localizações de cada TST no sistema,
		PARA gerenciar as localizações ativas ou não dos TSTs.
		Uma localização possui Endereço, Latitude, Longitude, Código IBGE e MAC Address
RF004	Manter Redes	COMO Administrador ou Operário,
		QUERO ter a possibilidade de incluir, visualizar, alterar e excluir as redes definidas para cada localização dos TSTs nos sistema,
		PARA gerenciar as redes definidas nos TSTs.
		Uma rede possui Localização e IP
RF005	Manter Atribuição	**COMO** Administrador ou Operário,
		QUERO ter a possibilidade de incluir, visualizar, alterar e excluir as atribuições realizadas entre localização do TST e Sinalizações,
		PARA gerenciar os TSTs no TCMC.
		Uma atribuição possui, Localização, atribuição, usuário, TTL, Timestamp, Observação e Status

RF006	Visualizar Usuário	**COMO** Analista,
		QUERO ter a possibilidade de apenas visualizar os usuários existentes e suas respectivas funções no TCMC,
		PARA acionar o devido responsável caso ocorra alguma inconsistência no TST.
RF007	Visualizar Sinalização	**COMO** Analista,
		QUERO ter a possibilidade de apenas visualizar as sinalizações existentes no TCMC
		PARA ficar ciente das sinalizações do TCMC.
RF008	Visualizar Localização	**COMO** Analista,
		QUERO ter a possibilidade de apenas visualizar as localizações dos TSTs (ativos ou não) no TCMC,
		PARA ter conhecimento das localizações ja cadastradas no sistema, estando elas atribuídas ou não.
RF009	Visualizar Redes	**COMO** Analista,
		QUERO ter a possibilidade de apenas visualizar as redes de cada um dos TSTs (se houverem) no TCMC,
		PARA ter conhecimento das redes já cadastradas no sistema, tendo a localização sendo atribuída ou não.
RF010	Visualizar Mapa	**COMO** usuário do sistema (Administrador, Operador ou Analista),
		QUERO ter a possibilidade de visualizar o mapa contendo todos os TSTs atribuídos no TCMC,
		PARA realizar a análise dos parâmetros e status de cada TST.
RF011	Definir tipos de sinalização	Os Tipos de sinalizações são: Regulamentação, Alerta, Indicação, Turístico e Obras
RF012	Definir Status da atribuição	Os status da atribuição são: 1- Aguandando envio, 2 - Atribuição enviada, 3- Erro no envio
RF013	Realizar comunicação com o TST	O controlador do sistema deve conseguir se comunicar via socket com o TST
RF014	Enviar informações da atribuição	O controlador só irá enviar as informações da atribuição caso confirme a localidade do TST

A Tabela 4 representa os requisitos não funcionais existentes para o desenvolvimento do TCMC.

Tabela 4 – Requisitos não funcionais do sistema.

Código	**Nome**	**Requisito**
RNF001	Definição de login e senha	Por padrão o login é o e-mail do usuário e a senha é padrão.
RNF002	Armazenamento da senha	A senha deverá ser armazenada no banco cifrada
RNF003	Acesso ao sistema	O usuário precisa logar-se para ter acesso ao sistema.
RNF004	Portabilidade	A interface gerencial web deve se desenvolvida para executar qualquer ambiente e o Controlador do TCMC deve ser desenvolvido para executar em ambiente windows.

A partir dos requisitos funcionais e não funcionais, foi criado o diagrama de caso de uso do sistema, conforme podemos verificar na Figura 8.

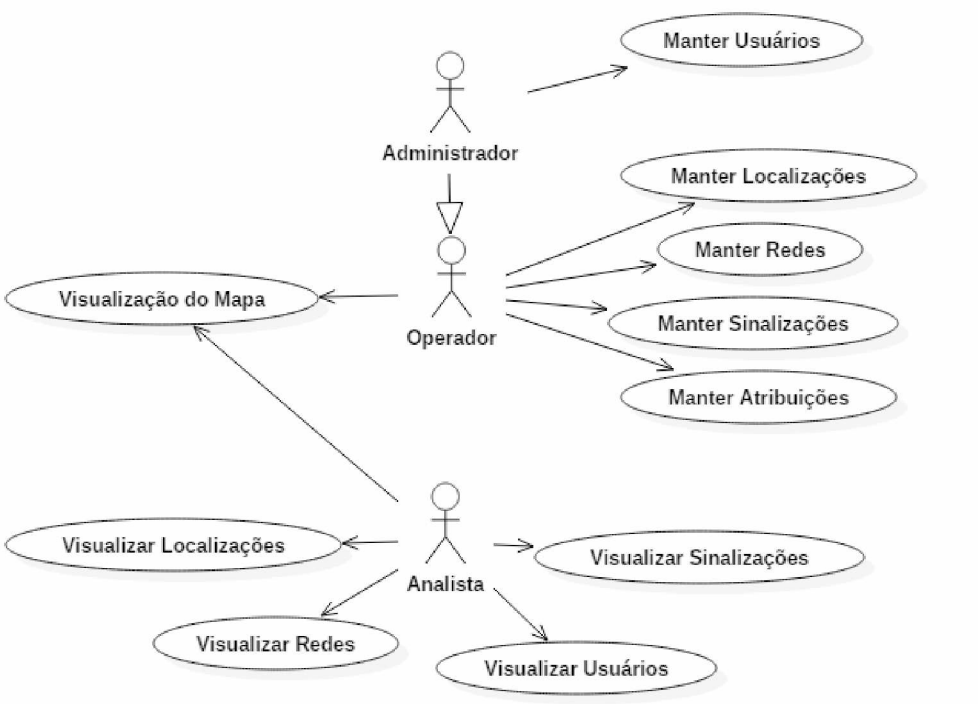

Figura 8 – Diagrama de Caso de Uso do TCMC.

As documentações dos casos de uso “Manter Usuário”, “Manter Sinalização”, “Manter Localização”, “Manter Rede” e “Manter Atribuição” podem ser verificadas nas Tabelas 5, 6, 7, 8 e 9, respectivamente.

Tabela 5 – Caso de Uso - Manter Usuário.

Identificador do caso de uso:
Manter Usuário

Objetivo
O objetivo deste caso de uso é detalhar o fluxo de inserção, alteração, visualização e exclusão de usuários.

Atores
Administrador.

Pré-condição:
O usuário deve estar logado no TCMC.

Fluxo Principal:
1. TCMC: Exibe tela home;
2. Usuário: Clica em usuários; [FA1]
3. TCMC: Verifica na base de dados os níveis de permissão existentes.
4. TCMC: Exibe tela de usuários;
5. Usuário: Preenche as informações restantes;
6. Usuário: Clica em enviar;
7. TCMC: Verifica na base de dados que o usuário ainda não está cadastrada; [FA2]
8. TCMC: Inclui o usuário na base de dados;
9. TCMC: Exibe mensagem "Dados inseridos com sucesso".

Fluxo Alternativo 1 – Usuário clica em buscar usuários;
1. Usuário: Clica em buscar usuário;
2. TCMC: Verifica na base de dados os usuários existentes;
3. TCMC: Exibe tabela com os usuários existentes.
4. TCMC: Caso o nível de permissão do usuário for Administrador ou Operador, exibe botão deletar.
5. Usuário: Clica em deletar;
6. TCMC: Exclui na base de dados o usuário selecionado.

Fluxo Alternativo 2 – Usuário existe na base de dados
1. TCMC: Verifica na base de dados que o usuário já está cadastrado
2. TCMC: Exibe mensagem "Já existem instâncias deste usuário na base de dados, deseja altera-lo? Sim / Não"
3. Usuário: Clica em Sim; [FA3]
4. TCMC: Altera os parâmetros do usuário na base de dados;
5. TCMC: Exibe mensagem: "Dados alterados com sucesso".

Fluxo Alternativo 3 – Usuário clica em Não
1. Usuário: Clica em Não;
2. TCMC: Exibe mensagem: "Dados mantidos com sucesso".

Tabela 6 – Caso de Uso - Manter Sinalização.

Identificador do caso de uso:
Manter Sinalização

Objetivo:
O objetivo deste caso de uso é detalhar o fluxo de inserção, alteração, visualização e exclusão de sinalizações.

Atores:
Administrador e Operador.

Pré-condição:
O usuário deve estar logado no TCMC.

Fluxo Principal:
1. TCMC: Exibe tela home;
2. Usuário: Clica em Sinalização; [FA1]
3. TCMC: Verifica na base de dados os tipos de sinalização existente.
4. TCMC: Exibe tela de sinalizações;
5. Usuário: Seleciona a imagem da sinalização ;
6. Usuário: Seleciona o tipo da sinalização;
7. Usuário: Preenche as informações restantes;
8. Usuário: Clica em enviar;
9. TCMC: Verifica na base de dados que a sinalização ainda não está cadastrada; [FA2]
10. TCMC: Inclui a sinalização na base de dados;
11. TCMC: Exibe mensagem "Dados inseridos com sucesso".

Fluxo Alternativo 1 – Usuário clica em buscar sinalização;
1. Usuário: Clica em buscar sinalização;
2. TCMC: Verifica na base de dados as sinalizações existentes;
3. TCMC: Exibe tabela com as sinalizações existentes e botão deletar em cada uma das instâncias.
4. Usuário: Clica em deletar;
5. TCMC: Exclui na base de dados a localização selecionada.

Fluxo Alternativo 2 – Sinalização já existe na base de dados
1. TCMC: Verifica na base de dados que a sinalização já está cadastrada
2. TCMC: Exibe mensagem "Já existem instâncias desta sinalização na base de dados, deseja altera-lo? Sim / Não"
3. Usuário: Clica em Sim; [FA3]
4. TCMC: Altera os parâmetros da sinalização na base de dados;
5. TCMC: Exibe mensagem: "Dados alterados com sucesso".

Fluxo Alternativo 3 – Usuário clica em Não
1. Usuário: Clica em Não;
2. TCMC: Exibe mensagem: "Dados mantidos com sucesso".

Tabela 7 – Caso de Uso - Manter Localização.

Identificador do caso de uso:
Manter Localização

Objetivo:
O objetivo deste caso de uso é detalhar o fluxo de inserção, alteração, visualização e exclusão de localizações.

Atores:
Administrador e Operador

Pré-condição:
O usuário deve estar logado no TCMC.

Fluxo Principal:
1. TCMC: Exibe tela home;
2. Usuário: Clica em localizações; [FA1]
3. TCMC: Exibe tela de localizações;
4. Usuário: Digita o endereço ou CEP da localização do TST;
5. Usuário: Clica em buscar;
6. TCMC: Chama API Google Maps passando o endereço digitado;
7. API Google Maps: Retorna Json com os dados do endereço encontrado;
8. TCMC: Exibe Google Maps com a localização informada;
9. Usuário: Preenche as informações restantes;
10. Usuário: Clica em enviar;
11. TCMC: Verifica na base de dados que a localização ainda não está cadastrada; [FA2]
12. TCMC: Inclui a localização na base de dados;
13. TCMC: Exibe mensagem "Dados inseridos com sucesso".

Fluxo Alternativo 1 – Usuário clica em buscar localizações;
1. Usuário: Clica em buscar localizações;
2. TCMC: Verifica na base de dados as localizações existentes;
3. TCMC: Exibe tabela com as localizações existentes e botão deletar em cada uma das instâncias.
4. Usuário: Clica em deletar;
5. TCMC: Exclui na base de dados a localização selecionada.

Fluxo Alternativo 2 – Localização já existe na base de dados
1. TCMC: Verifica na base de dados que a localização já está cadastrada
2. TCMC: Exibe mensagem "Já existem instâncias desta localização na base de dados, deseja altera-lo? Sim / Não"
3. Usuário: Clica em Sim; [FA3]
4. TCMC: Altera os parâmetros da localização na base de dados;
5. TCMC: Exibe mensagem: "Dados alterados com sucesso".

Fluxo Alternativo 3 – Usuário clica em Não
1. Usuário: Clica em Não;
2. TCMC: Exibe mensagem: "Dados mantidos com sucesso".

Tabela 8 – Caso de Uso - Manter Redes.

Identificador do caso de uso:
Manter Redes

Objetivo:
O objetivo deste caso de uso é detalhar o fluxo de inserção, alteração, visualização e exclusão da rede da localização do TST.

Atores:
Administrador e Operador.

Pré-condição:
O usuário deve estar logado no TCMC.

Fluxo Principal:
1. TCMC: Exibe tela home;
2. Usuário: Clica em redes; [FA1]
3. TCMC: Verifica na base de dados as localizações existentes;
4. TCMC: Exibe tela de redes;
5. Usuário: Seleciona a localização;
6. Usuário: Preenche o IP da localização;
7. Usuário: Clica em enviar;
8. TCMC: Verifica na base de dados que a rede da localização ainda não está cadastrada; [FA2]
9. TCMC: Inclui a rede da localização na base de dados;
10. TCMC: Exibe mensagem "Dados inseridos com sucesso".

Fluxo Alternativo 1 – Usuário clica em buscar rede;
1. Usuário: Clica em buscar rede;
2. TCMC: Verifica na base de dados as redes existentes;
3. TCMC: Exibe tabela com as redes existentes e botão deletar em cada uma das instâncias.
4. Usuário: Clica em deletar;
5. TCMC: Exclui na base de dados a rede selecionada.

Fluxo Alternativo 2 – Rede da localização já existe na base de dados
1. TCMC: Verifica na base de dados que a rede da localização já está cadastrada
2. TCMC: Exibe mensagem "Já existem instâncias da rede desta localização na base de dados, deseja altera-la? Sim / Não"
3. Usuário: Clica em Sim; [FA3]
4. TCMC: Altera os parâmetros da rede da localização na base de dados;
5. TCMC: Exibe mensagem: "Dados alterados com sucesso".

Fluxo Alternativo 3 – Usuário clica em Não
1. Usuário: Clica em Não;
2. TCMC: Exibe mensagem: "Dados mantidos com sucesso".

Tabela 9 – Caso de Uso - Manter Atribuição.

Identificador do caso de uso:
Manter Atribuição

Objetivo:
O objetivo deste caso de uso é detalhar o fluxo de inserção, alteração, visualização e exclusão de atribuições dos TSTs.

Atores:
Administrador, Operador.

Pré-condição:
O usuário deve estar logado no TCMC.

Fluxo Principal:
' 1. Usuário: Clica em Atribuições; [FA1]
2. TCMC: Verifica na base de dados as localizações que possuem rede e os tipos de sinalizações existentes;
3. TCMC: Exibe tela de atribuições;
4. Usuário: Seleciona a localização;
5. TCMC: Insere o marcador da localização no google maps;
6. TCMC: Exibe mapa da localização;
7. Usuário: Seleciona o tipo de sinalização;
8. TCMC: Verifica na base de dados quais são as localizações que pertencem ao tipo selecionado;
9. TCMC: Monta dinamicamente a combo box de sinalização com somente as sinalizações pertencentes ao tipo predefinido;
10. Usuário: Seleciona a sinalização e preenche o restante dos dados;
11. Usuário: Clica em enviar;
12. TCMC: Verifica na base de dados que a atribuição ainda não está cadastrada; [FA2]
13. TCMC: Inclui a atribuição na base de dados;
14. TCMC: Exibe mensagem "Dados inseridos com sucesso".

Fluxo Alternativo 1 – Usuário clica em mapa;
1. Usuário: Clica em buscar mapa;
2. TCMC: Verifica na base de dados as atribuições existentes;
3. TCMC: Insere os marcadores da localização das atribuições no google maps;
4. TCMC: Exibe mapa com as atribuições existentes e seus respectivos status;
5. Usuário: Clica em uma marcação;
6. TCMC: Exibe janela de informação contendo o endereço, sinalização e eventual erro ocorrido;
7. TCMC: Caso o nível de permissão do usuário for Administrador ou Operador, exibe botão deletar;
8. Usuário: Clica em deletar;
9. TCMC: Exclui na base de dados a atribuição selecionada.

Fluxo Alternativo 2 – Atribuição já existe na base de dados
1. TCMC: Verifica na base de dados que a atribuição já está cadastrada
2. TCMC: Exibe mensagem "Já existem instâncias desta atribuição na base de dados, deseja altera-la? Sim / Não";
3. Usuário: Clica em Sim; [FA3]
4. TCMC: Altera os parâmetros da atribuição na base de dados;
5. TCMC: Exibe mensagem: "Dados alterados com sucesso".

Fluxo Alternativo 3 – Usuário clica em Não
1. Usuário: Clica em Não;
2. TCMC: Exibe mensagem: "Dados mantidos com sucesso".

Os casos de uso "Visualizar Usuário", "Visualizar Sinalização", "Visualizar Localização", "Visualizar Rede" e "Visualizar Mapa" seguem o mesmo padrão de documentação, mudando apenas o ícone que é clicado pelo usuário e os respectivos parâmetros de cada classe. Um exemplo deste padrão pode ser visto na Tabela 10.

Tabela 10 – Caso de Uso - Visualizar usuário.

Identificador do caso de uso:
Visualizar Usuário

Objetivo:
O objetivo deste caso de uso é detalhar o fluxo de visualização de usuários.

Atores:
Analista.

Pré-condição:
O usuário deve estar logado no TCMC.

Fluxo Principal:
1. TCMC: Exibe tela home;
2. Usuário: Clica em Buscar Usuários;
3. TCMC: Verifica na base de dados os usuários existentes.
4. TCMC: Exibe tela com a tabela de usuários do TCMC;

Para a criação do banco de dados foi modelado o diagrama de entidade e relacionamento visto na Figura 9. Nele podemos verificar cada uma das tabelas do TCMC (Tipo das sinalizações, sinalizações, nível de permissão do usuário, usuários, redes dos TSTs, localizações dos TSTs, possíveis status dos TSTs e as atribuições realizadas) e suas respectivas relações. É importante ressaltar que cada rede, localização, sinalização e atribuição tem a identificação do respectivo usuário que realizou o cadastro.

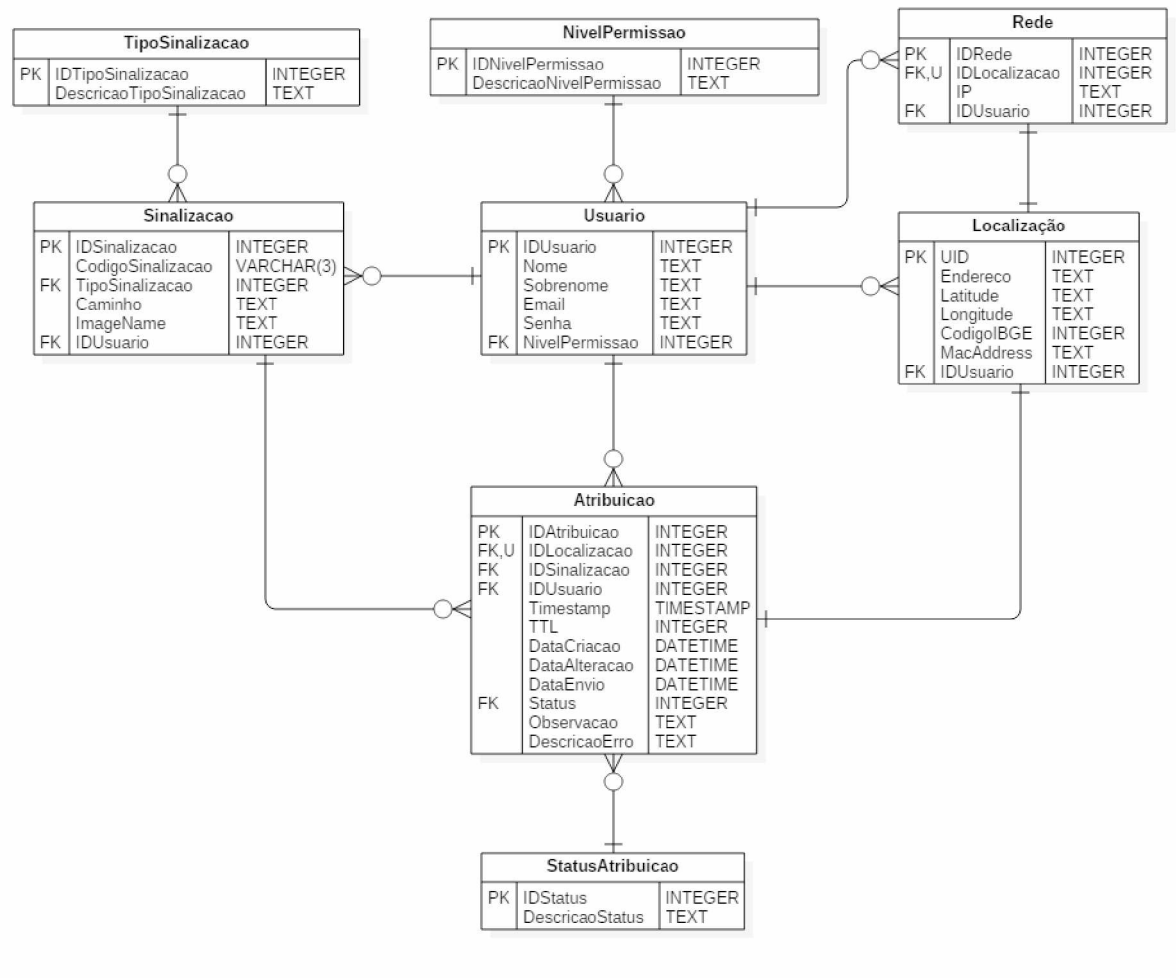

Figura 9 – Diagrama de entidade e relacionamento do TCMC.

Na Tabela 11 é apresentado os parâmetros da tabela atribuição. Esta tabela será manipulada tanto pela interface gerencial web quanto pelo controlador.

A interface gerencial web cadastra a atribuição da sinalização e localização do TST e define a data de criação, alteração e status desta atribuição. O status da atribuição definido pela interface gerencial web será igual a 1 (Aguardando Envio).

Já o controlador define a data de envio e status da atribuição, que pode ser 2 (TST enviado) caso o envio das informações tenha sido enviada com sucesso ou 3 (Erro no envio) caso o ocorra algum erro no envio das informações. Neste ultimo caso, o controlador também irá definir o campo descrição erro com uma explicação do erro ocorrido.

Tabela 11 – Campos da tabela atribuição.

Campo	**Tipo**	**Nulo**	**Chave primária**	**Chave estrangeira**	**Observação**
IDAtribuição	Integer	Não	Sim	Não	
IDLocalização	Integer	Não	Sim	Não	Vem da tabela Localizacao
IDSinalização	Integer	Não	Não	Sim	Vem da tabela Sinalizacao
IDUsuario	Integer	Não	Não	Sim	Vem da tabela Usuario
Timestamp	Timestamp	Não	Não	Não	
TTL	Não	Não	Não	Não	
DataCriacao	Não	Não	Não	Não	
DataAlteracao	Não	Não	Não	Não	Default 1970-01-01T00:00:00
DataEnvio	Não	Não	Não	Não	Default 1970-01-01T00:00:00
Status	Integer	Não	Não	Sim	Vem da tabela StatusAtribuicao
Observacao	Text	Sim	Não	Não	
DescricaoErro	Text	Sim	Não	Não	

4.3 Apresentação do Sistema

O sistema desenvolvido tem por objetivo auxiliar os operadores internos do TCMC em relação ao controle dos TSTs de sua área de atuação.

O layout do sistema é composto por três setores:

- **Setor superior**: Contém o logotipo do sistema e uma área de exibição dos dados do perfil do usuário. Ao clicar no ícone do perfil, o botão de logout do sistema é exibido;
- **Setor lateral esquerdo**: Contém o menu lateral fixo do sistema;
- **Setor central**: Contém o conteúdo da página que esta sendo navegada.

O setor lateral esquerdo será apresentado de forma diferente dependendo do nível de permissão do usuário. A Figura 10 apresenta os tipos diferentes de menus por nível de permissão, onde o analista pode apenas visualizar cada uma das funcionalidades, o operador pode realizar o gerenciamento das redes, sinalizações, localizações e atribuições, e por fim, o administrador que além de ter as permissões do operador, também tem a permissão de gerenciar os usuários.

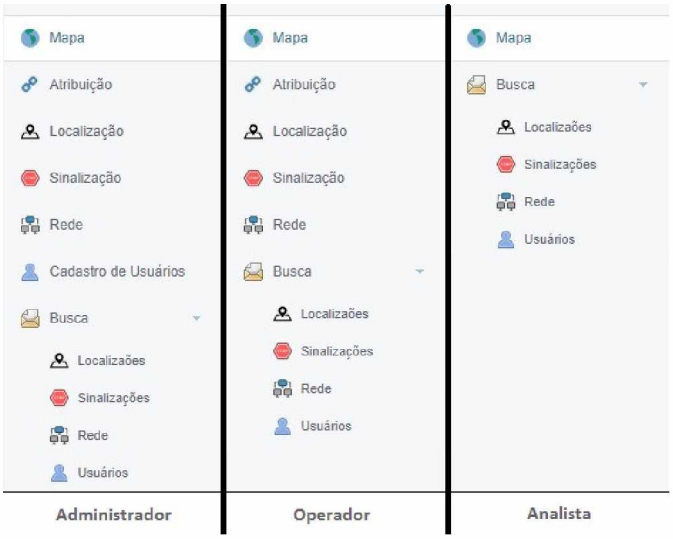

Figura 10 – Menus por nível de permissão do usuário.

A Figura 11 apresenta a tela inicial do sistema sob a visão do administrador, é nesta tela que é exibido o mapa contendo todos os TSTs da área de atuação do TCMC. Para mostrar o status atual do TST na região, o mapa é atualizado a cada 30 segundos (valor arbitrário) e a data e hora da última atualização é exibida acima do mesmo.

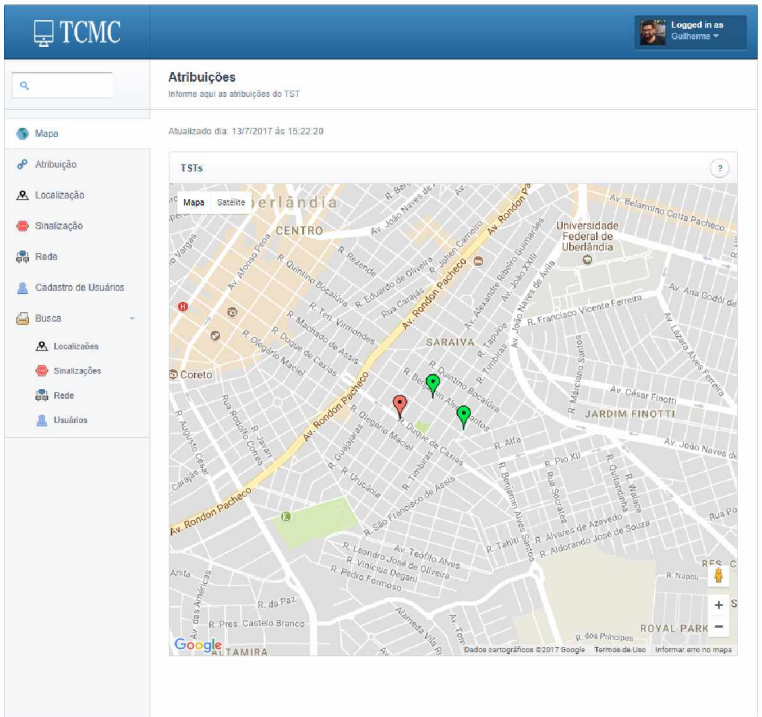

Figura 11 – Tela inicial do TCMC.

A cor dos marcadores do mapa variam conforme o status da atribuição, a Tabela 12 representa esta relação.

Tabela 12 – Relação dos marcadores do mapa com os status da atribuição.

Marcador	**Status**
	Erro de envio (3)
	Aguardando Envio (1)
	Atribuição enviada (2)

Caso o usuário do sistema clique em um destes marcadores ele poderá verificar uma janela de informações contendo o endereço, a sinalização e algum eventual erro existente. Além disto, caso o usuário tiver nível de permissão de administrador ou operador, poderá deletar o TST exibido. A Figura 12 apresenta um exemplo de janela de informações sob a visão de um administrador.

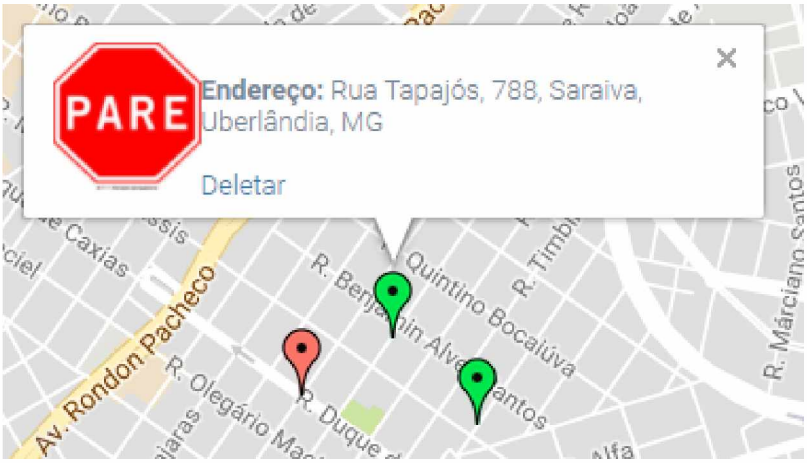

Figura 12 – Janela de informações do TST no mapa da tela inicial.

Conforme vimos na Figura 10, existem diferentes tipos de menu dependendo do nível de permissão do usuário. Contudo, caso aconteça de um usuário de nível analista ou operador descobrir uma URL (*Uniform Resource Locator*) que não possui permissão e tentar entrar nesta página, o mesmo será redirecionado para a página de erro de permissão visto na Figura 13.

Figura 13 – Página de erros do sistema.

A Figura 14 apresenta a página de formulário de cadastro de uma atribuição, nela podemos definir qual a localidade do TST, a sua sinalização, o horário em que a sinalização deverá começar a ser exibida (Timestamp), a duração em segundos da sinalização a partir do Timestamp (TTL) e uma observação sobre o TST.

As páginas de formulários de "Usuário", "Sinalização", "Localização" e "Rede" seguem a mesma lógica da página de "Atribuição" vista na Figura 14, mudando apenas os parâmetros de cada formulário. O fluxo destas páginas podem ser vistos nas Tabelas 5, 6, 7, 8 e 9 da seção 4.1.1.

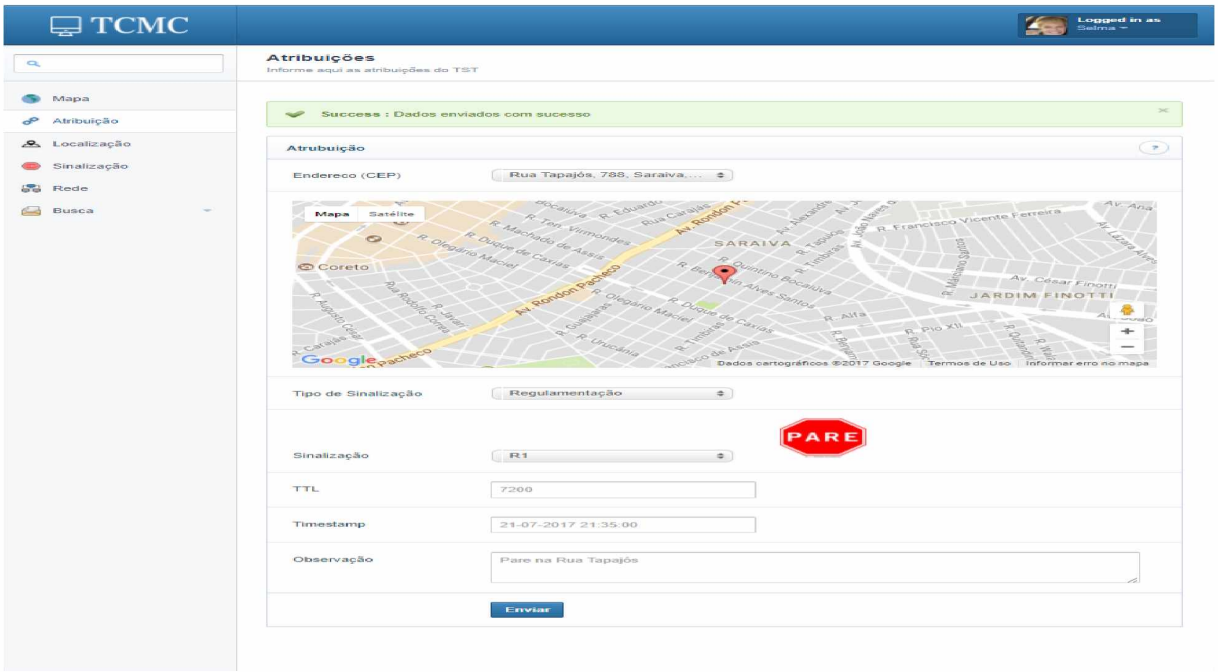

Figura 14 – Formulário de atribuição do TST.

Após o usuário clicar em enviar, os parâmetros são enviados para o servidor web através do Ajax. O servidor web então armazena os parâmetros na base de dados e retorna que o armazenamento foi feito com sucesso ao cliente. Com isso, utilizando o Ajax, é possível mostrar as mensagens de sucesso (ou erro) realizados no cadastro sem a necessidade de recarregar a página inteira. A Figura 15 mostra através da ferramenta phpMyAdmin, os dados da Figura 14 agora armazenados no banco de dados.

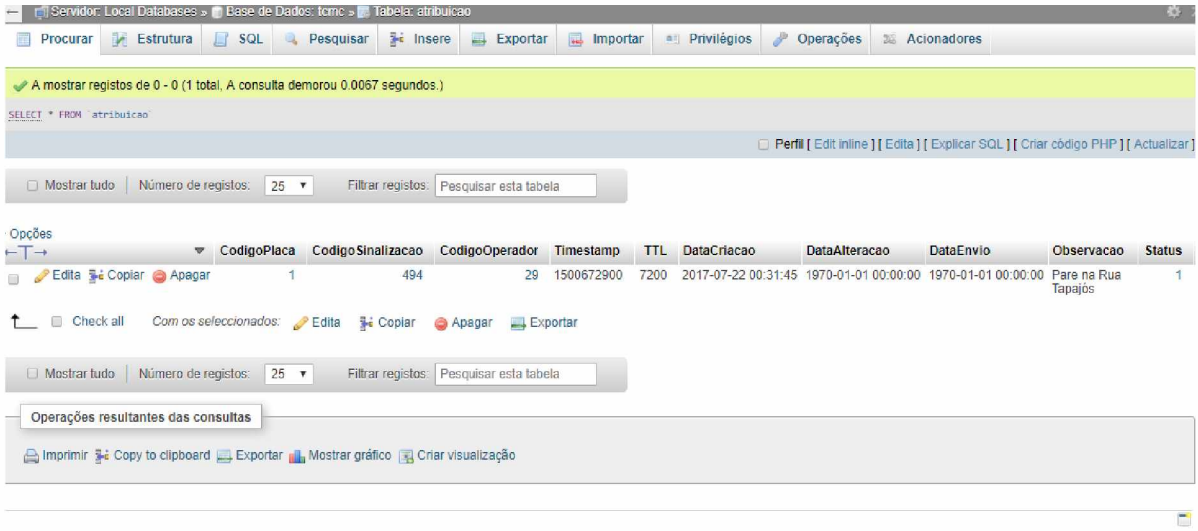

Figura 15 – Banco de dados visualizado através da ferramenta phpMyAdmin.

Com os dados armazenados, o controlador verificará se existe alguma atribuição com o status "Aguardando Envio" na base de dados, caso exista, tentará realizar a comunicação TCP com o IP definido na tabela de redes do TST.

Caso a conexão aconteça, o controlador pedirá a localização do servidor TST. Se o TST estiver exatamente na localização definida na base de dados, o controlador enviará as informações da atribuição para o servidor TST. Este então tentará criar um arquivo texto com os dados recebidos. Em caso positivo, retorna para o controlador que tudo ocorreu conforme o esperado. O controlador por fim atualiza o status da atribuição para "Atribuição Enviada" e a data de envio com a data atual.

Caso ocorra algum erro de iniciação da biblioteca de socket, como: criação do socket, erro ao se conectar, conexão perdida, falha na criação do arquivo de atribuições do TST ou o TST não pertence a localização definida, o controlador irá alterar o status da atribuição para "Erro de Envio" e armazenar a descrição do erro na tabela.

A Figura 16 mostra os dados armazenados na Figura 14 agora encontrados pelo controlador e enviados para o servidor TST.

O módulo TST criado por (LIRA et al., 2016) lê o arquivo gerado pelo servidor e gera o ponto de acesso com os dados da atribuição.

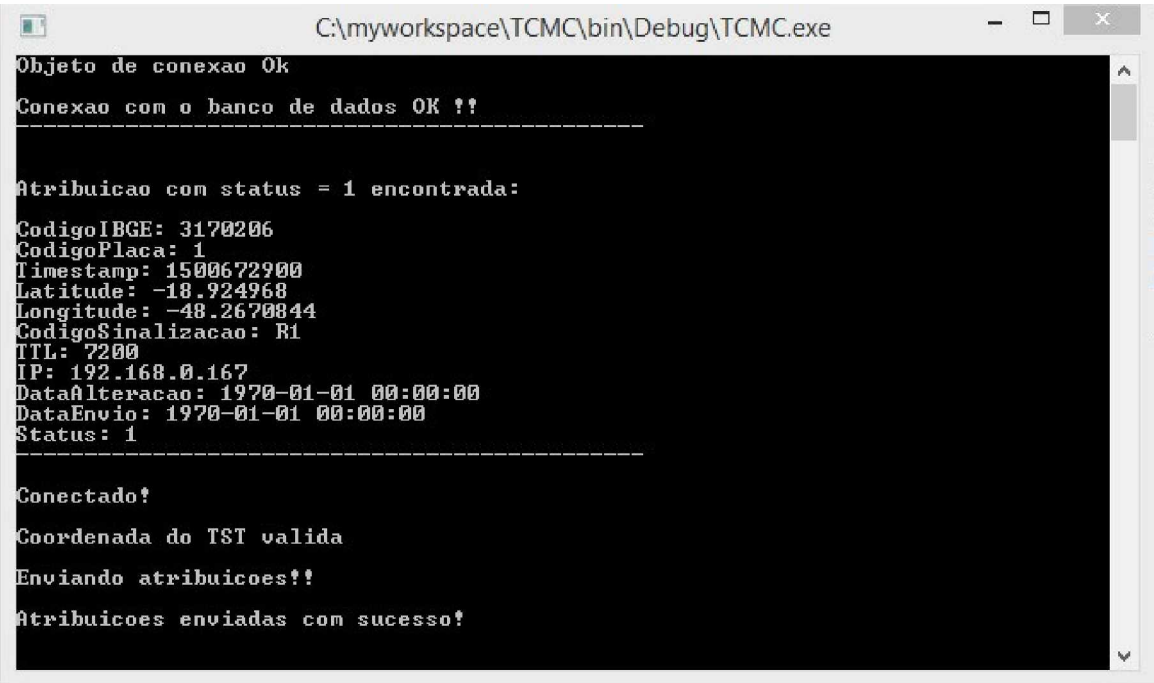

Figura 16 – Display do Controlador com os dados enviados para o TST.

Por fim, o módulo VUR criado por (LIRA et al., 2016), busca os pontos de acessos que estão na arquitetura do TSMA e exibe a sinalização no display do veículo.

4.4 Implementação do Sistema

Nesta seção são apresentados códigos com o objetivo de exemplificar a implementação de algumas funcionalidades da central de gerenciamento de controle de tráfego.

4.4.1 Mapa das atribuições

Para criar o mapa de atribuições da tela inicial foi necessário utilizar a técnica Ajax para solicitar a localização de todos os TSTs do banco de dados. Tendo todas as localizações, foi definido os marcadores no mapa e suas eventuais alterações conforme o status do TST.

```
1  function ajaxCall(){
2    //localização inicial
3     var id = "";
4      //Chama serviço via Ajax para retornar os dados da atribuição
5      $.post("MapaAux.php", {Localizacao:id}, function(data){
6        //Converte Json para Array
7        var locations = data_to_array(data);
8
9        //Definição inicial do mapa caso não tenha nenhum marcador
10       var myOptions = {
```

```
11          // Posição do centro do mapa
12          center: new google.maps.LatLng(33.890542, 151.274856),
13          //Tamanho do Zoom
14          zoom: 13,
15          //Tipo do Mapa = Padrão
16          mapTypeId: google.maps.MapTypeId.ROADMAP
17
18        };
19        //Define o mapa pelo id da div no html
20        var map = new google.maps.Map(document.getElementById("default"),
21          myOptions);
22
23        //Chama função para incluir os marcadores
24        setMarkers(map,locations)
25      },"json");
26
27      //Chama função para mostrar o horário da ultima atualização
28      MostraHorario();
29
30  }
```

O retorno do serviço "MapaAux" são os dados do TST em JSON, retornando a latitude, longitude, o endereço, o caminho da imagem da sinalização, o código da localização, o status da atribuição e algum eventual erro.

```
1  [{
2    "Latitude": "-18.924968",
3    "Longitude": "-48.2670844",
4    "Endereco": "Rua Tapajós, 788, Saraiva, Uberlândia, MG",
5    "Caminho": "img\/simple\/Regulamentacao\/R-1.jpg",
6    "CodigoPlaca": "1",
7    "Status": "2",
8    "DescricaoErro": ""
9  }]
```

Com estes dados, a função "setMarkers" irá gerar o marcador na latitude e longitude informada, alterar a cor dependendo do status e informar na janela de informações o endereço, a imagem definida pelo identificador "Caminho" e o erro retornado em "DescricaoErro". O código completo da criação do mapa de atribuições pode ser verificado no Anexo A no final deste trabalho.

4.4.2 Comunicação com o TST

Já a comunicação com o TST foi feita via socket TCP/IP em linguagem C++. Como o sistema operacional utilizado na criação do controlador foi o Windows 8.1, utilizamos a biblioteca winsock. Para a comunicação, foi criada uma função chamada "ClientSocket" que tem como parâmetros de entrada a região da memória que armazena a mensagem com os dados da atribuição (buffer), o IP do TST armazenado na base de dados (host), e as coordenadas do TST (coordinate). O controlador irá verificar se existe alguma nova atribuição no banco de dados e caso houver, irá chamar a função passando os dados, o IP e a coordenada inserida no banco. Os passos da função são:

- Definir as variáveis;

```
1  //Estrutura utilizada para armazenar informações de inicialização
       dos sockets do windows.
2  WSADATA data;
3
4  //Cria um Socket
5  SOCKET winsock;
6
7  //Cria uma estrutura SOCKADDR_IN, ao qual serão definidos os parâ
       metros do endereço do socket
8  SOCKADDR_IN sock;
9
10 //região da memória para armazenar a mensagem de retorno
11 char buffer2[1024];
12
13 //região da memória para armazenar a mensagem de pedido de
       coordenada
14 char coordinateRequest[15];
15
16 //variável para armazenar o número de bytes recebidos
17 int bytes;
```

- Inicializar a biblioteca winsock;

```
1  if (WSAStartup(MAKEWORD(1, 1), & data) == SOCKET_ERROR) {
2
3      // Se der erro ao iniciar, exibe mensagem e retorna código do
           erro
4      printf("Erro ao inicializar o winsock\n\n");
5      return 99;
6    }
```

A função WSAStartup() serve para inicializar a biblioteca winsock e recebe como parâmetros a sua versão e o ponteiro para a estrutura que armazena as informações do socket no windows.

- Inicializar o socket;

```
1 if ((winsock = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP)) ==
    SOCKET_ERROR) {
2
3     //Se der erro ao criar, exibe mensagem e retorna código do erro
4     printf("Erro ao criar socket\n\n");
5     return 98;
6   }
```

A função socket() serve para inicializar o socket e tem como parâmetros de entrada a família de protocolo, o tipo de socket e o protocolo a ser utilizado. No caso do TCMC, utilizamos os valores AF_INET que representa a família de endereços IPv4 (*Internet Protocol version 4*), o tipo SOCK_STREAM que define a utilização do protocolo de controle de transmissão (TCP) e o protocolo IPPROTO_TCP que define o protocolo TCP.

- Definir os parâmetros do endereço do socket;

```
1 //Define a família do endereço do Socket
2   sock.sin_family = AF_INET;
3
4   //Define o número da porta
5   sock.sin_port = htons(3000);
6
7   //Define o IP ao qual o socket será conectado
8   sock.sin_addr.s_addr = inet_addr(host);
```

- Conectar o socket;

```
1 //Conecta o socket
2   if (connect(winsock, (SOCKADDR * ) & sock, sizeof(sock)) ==
      SOCKET_ERROR) {
3
4     // Se ocorrer erro, exibe mensagem e retorna código do erro
5     printf("Erro ao se conectar\n\n");
6     return 97;
7   }
```

A função connect() tem por finalidade conectar o socket ao endereço definido na estrutura SOCKADDR e tem como parâmetros de entrada o socket iniciado, um ponteiro para a estrutura do endereço e o comprimento da estrutura sockaddr apontada no 2º argumento.

- Pedir a localização do TST;

```
1 //Define valor do buffer de pedido de coordenada
2  strcpy(coordinateRequest,"CoordenadasTST");
3 //Envia mensagem pedindo as coordenadas do TST
4   send(winsock, coordinateRequest, strlen(coordinateRequest), 0);
```

A função send() tem por finalidade enviar as informações para o servidor e tem como parâmetros de entrada o socket iniciado, o buffer ao qual será armazenado as informações a ser enviada, o tamanho deste buffer e uma flag indicando como será o modo de envio (no caso do TCMC, definimos como 0 pois não necessitamos de nenhuma ação especial no envio).

- Receber a localização do TST;

```
1  //Limpa a variavel de retorno
2   memset(buffer2, 0, 1024);
3
4   //Recebe as coordenadas do TST
5   bytes = recv(winsock, buffer2, 1024, 0);
6
7   //Se ocorreu algum erro no recebimento, exibe mensagem de conexão
        perdida e retorna código do erro
8   if (bytes == -1) {
9     printf("Conexão perdida\n\n");
10    return 96;
11  }
```

A função recv() tem por finalidade receber as informações enviadas pelo servidor e tem como parâmetros de entrada o socket iniciado, o buffer ao qual será armazenado as informações recebidas, o tamanho deste buffer e uma flag indicando como será o modo de recebimento (no caso do TCMC, também definimos a variável como 0 pois não necessitamos de nenhuma ação especial no envio).

- Verificar se a localização do TST é igual a localização atribuída e em caso positivo, enviar os dados da atribuição;

```
1  if(strcmp(buffer2,coordinate)==0){
2
3         //Exibe mensage de validação
4         printf("Coordenada do TST valida\n\nEnviando atribuicoes!!\
              n\n");
5
6         //Envia os atributos da atribução
7         send(winsock, buffer, strlen(buffer), 0);[...]
```

- Receber o parecer do servidor em relação a criação do arquivo texto com os dados da atribuição. Em caso positivo, a função retorna 1, em caso negativo, exibe mensagem de erro e retorna o código de erro;

```
          //Limpa a variável de retorno
        memset(buffer2, 0, 1024);

        //Recebe o retorno do TST
        bytes = recv(winsock, buffer2, 1024, 0);

        // Se ocorreu algum erro no recebimento, exibe mensagem de
           conexão perdida e retorna código do erro
            if (bytes == -1) {
            printf("Conexão perdida\n\n");
            return 96;
        }
        //Se ocorreu alguma falha na criação do arquivo, exibe
           mensagem de falha e retorna código do erro
        if (strcmp(buffer2,"Falha")==0){
            printf("Falha ao abrir o arquivo\n\n");
            return 95;
        }
        //Se tudo ocorreu bem retorna 1
        else if(strcmp(buffer2,"Enviado")==0){
              return 1;
        }
}
```

- Fechar o socket e finalizar as operações sockets no windows.

```
    //Fecha o socket
    closesocket(winsock);
    //Finaliza operações sockets no windows
    WSACleanup();
```

Assim, conseguimos realizar a conexão com o TST por meio da comunicação via socket TCP. O código completo da conexão com o TST pode ser verificado no anexo B no final deste trabalho.

5 Conclusão

A aplicação modelada e implementada neste projeto mostra uma forma simples e eficiente dos operadores do órgão nacional de trânsito gerenciarem os TSTs em uma central de gerenciamento de controle de tráfego (TCMC). O resultado do trabalho torna possível a atualização remota da infraestrutura de sinalização dentro da arquitetura proposta por (LIRA et al., 2016).

A utilização de ferramentas como raspberry, técnica Ajax, programação com sockets, desenvolvimento web, entre outros, foram muito importantes no desenvolvimento pessoal do autor. O mesmo utilizou técnicas aprendidas no curso de bacharelado em sistemas de informação, técnicas aprendidas com a orientação do professor e pesquisas de trabalhos relacionados.

Em relação a trabalhos futuros, podemos verificar:

- Implementar a criptografia RSA na transferência do TCMC para o TST;
- Criar uma área para gerenciar funcionalidades do TCMC, como por exemplo, o tempo de espera para a atualização do mapa contendo todos os TSTs;
- Definir uma "hierarquia entre os TCMCs, de forma que um TCMC de 1º nível seja responsável apenas pela gestão lógica da política de atualização das sinalizações e atendimento ao contato externo, enquanto TCMCs de 2º nível seriam responsáveis por dar manutenção aos equipamentos e implementar a atualização da sinalizações conforme os pedidos do TCMC de 1º nível. Este modelo poderia ser aplicado a nível de grandes cidades" segundo propõe (LIRA et al., 2016) em sua dissertação de mestrado.

Referências

DOWNS, A. Why traffic congestion is here to stay.... and will get worse. *ACCESS Magazine*, v. 1, n. 25, 2004. Citado na página 15.

FERNANDES, R. J. Vanet-enabled in-vehicle traffic signs. 2009. Citado 2 vezes nas páginas 17 e 18.

GARRETT, J. J. et al. Ajax: A new approach to web applications. 2005. Citado 2 vezes nas páginas 21 e 22.

GRACA, J. L. Driving and aging. clinics in geriatric medicine. In: . [S.l.: s.n.], 1986. Citado na página 11.

KLAUER, S. G. et al. *The impact of driver inattention on near-crash/crash risk: An analysis using the 100-car naturalistic driving study data.* [S.l.], 2006. Citado na página 12.

LIRA, E. R. et al. Tsma: uma arquitetura para gerenciar a sinalização de trânsito. Universidade Federal de Uberlândia, 2016. Citado 12 vezes nas páginas 12, 13, 14, 15, 18, 21, 23, 24, 26, 41, 42 e 48.

PETERMAN, D. R. Ensuring that traffic signs are visible at night: Federal regulations. 2013. Citado na página 12.

PRESSMAN, R. S. *Software engineering: a practitioner's approach.* [S.l.]: Palgrave Macmillan, 2005. Citado na página 22.

SATO, Y.; MAKANAE, K. Development and evaluation of in-vehicle signing system utilizing rfid tags as digital traffic signs. *International Journal of ITS Research*, v. 4, n. 1, 2006. Citado 2 vezes nas páginas 16 e 17.

SIVAK, M. Mortality from road crashes in 193 countries: A comparison with other leading causes of death. University of Michigan, Ann Arbor, Transportation Research Institute, 2014. Citado na página 11.

ZHANG T.; DELGROSSI, L. Vehicle safety communications: protocols, security, and privacy. 2012. Citado na página 11.

Anexos

ANEXO A – Código para criação do mapa de atribuições

```
1  //Função que inicaliza a chamada
2  function initialize() {
3    //Chama a função ajax
4    ajaxCall();
5
6    //Aguarda 30 segundos (Valor arbitrário) e chama a função novamente
7      setInterval(ajaxCall,30000);
8
9
10 }
11
12 function ajaxCall(){
13   //localização inicial
14    var id = "";
15     //Chama serviço para retornar os dados da atribuição
16     $.post('MapaAux.php', {Localizacao:id}, function(data){
17       //Converte Json para Array
18       var locations = data_to_array(data);
19
20       //Definição inicial do mapa caso não tenha nenhum marcador
21       var myOptions = {
22         // Posição do centro do mapa
23         center: new google.maps.LatLng(33.890542, 151.274856),
24         //Tamanho do Zoom
25         zoom: 13,
26         //Tipo do Mapa = Padrão
27         mapTypeId: google.maps.MapTypeId.ROADMAP
28
29       };
30       //Define o mapa pela div html
31       var map = new google.maps.Map(document.getElementById("default"),
32         myOptions);
33
34       //Chama função para incluir os marcadores
35       setMarkers(map,locations)
36     },'json');
37
38     //Chama função para mostrar o horário da ultima atualização
39     MostraHorario();
40
```

```
41 }
42
43   function setMarkers(map,locations){
44   //Define as variaveis do marcador e do loop
45     var marker, i
46
47   //loop enquanto existir localizações
48   for (i = 0; i < locations.length; i++)
49   {
50
51     //Variavel da latitude
52      var lat = locations[i]["Latitude"]
53
54     //Variavel da longitude
55     var long = locations[i]["Longitude"]
56
57     //Variavel do endereço
58     var add =  locations[i]["Endereco"]
59
60     //Variavel da sinalização
61     var sinal = locations[i]["Caminho"]
62
63     //Variavel do ID da localização
64     var UID = locations[i]["CodigoPlaca"]
65
66     //Variavel do Status do TST
67     var status= locations[i]["Status"]
68
69     //Variavel da descrição do erro
70     var descErro = locations[i]["DescricaoErro"]
71
72     //Define uma coordenada
73     latlngset = new google.maps.LatLng(lat, long);
74
75     //Define o marcador com a coordenada
76     marker = new google.maps.Marker({
77         map: map , position: latlngset
78       });
79     //Define a posição do centro do mapa
80     map.setCenter(marker.getPosition())
81
82
83     //Define uma janela de informações
84     var infowindow = new google.maps.InfoWindow()
85
86     //Se status for igual a 1: Muda a cor do marcador para amarela
87     if(status==1){
```

```
88        marker.setIcon('http://maps.google.com/mapfiles/ms/icons/yellow-
              dot.png');
89        var content = '<IMG BORDER="0" ALIGN="Left" width="70" height="70"
               SRC="'+sinal+'"><br/><b>Endereço:</b> ' +add+ '<br/><br/><a
              href=?apagar="'+UID+'">Deletar</a>'
90      }
91      //Se status for igual a 2: Muda a cor do marcador para verde
92      else if(status==2){
93        marker.setIcon('http://maps.google.com/mapfiles/ms/icons/green-dot
              .png');
94        var content = '<IMG BORDER="0" ALIGN="Left" width="70" height="70"
               SRC="'+sinal+'"><br/><b>Endereço:</b> ' +add+ '<br/><br/><a
              href=?apagar="'+UID+'">Deletar</a>'
95      }
96      //Se status for igual a 3: Muda a cor do marcador para vermelho.
97      else if (status==3){
98        marker.setIcon('http://maps.google.com/mapfiles/ms/icons/red-dot.
              png');
99        var content = '<IMG BORDER="0" ALIGN="Left" width="70" height="70"
               SRC="'+sinal+'"><br/><b>Endereço:</b> ' +add+ '<br/><br/><b>
              Erro:</b> '+descErro+'<br/><br/><a href=?apagar="'+UID+'">
              Deletar</a> / <a href=?tentar="'+UID+'">Tentar Novamente</a>'
100     }
101
102     //Define a janela de informações quando clicar no marcador
103     google.maps.event.addListener(marker,'click', (function(marker,
            content,infowindow){
104       return function() {
105         infowindow.setContent(content);
106         infowindow.open(map,marker);
107       };
108     })(marker,content,infowindow));
109
110   }
111 }
```

ANEXO B – Código para comunicação com o TST

```
1  #include <stdio.h>
2  #include <winsock.h>
3  #include <conio.h>
4  #include <windows.h>
5  #include <string.h>
6  //Função que inicaliza a comunicação com o TST via socket TCP/IP
7  //ClientSocket (região da memória que armazena a mensagem com os dados
       da atribuição, IP do TST, Coordenadas do TST)
8  int ClientSocket(char buffer[1024],char host[12],char coordinate[50]) {
9
10 //Estrutura utilizada para armazenar informações de inicialização do
       Windows Sockets
11 WSADATA data;
12
13 //Cria um Socket
14 SOCKET winsock;
15
16 //Cria uma estrutura SOCKADDR_IN
17 SOCKADDR_IN sock;
18
19 //região da memória para armazenar a mensagem de retorno
20 char buffer2[1024];
21
22 //variável para armazenar o número de bytes recebidos
23 int bytes;
24
25 //Inicializa a biblioteca winsock
26   if (WSAStartup(MAKEWORD(1, 1), & data) == SOCKET_ERROR) {
27
28     // Se der erro ao inicar, exibe mensagem e retorna código do erro
29     printf("Erro ao inicializar o winsock\n\n");
30     return 99;
31   }
32
33 //Conecta o socket connect(Socket, ponteiro para a estrutura do endereço
       , comprimento da estrutura sockaddr apontada pelo argumento do endere
       ço)
34   if (connect(winsock, (SOCKADDR * ) & sock, sizeof(sock)) ==
         SOCKET_ERROR) {
35
```

```
36      // Se ocorrer erro, exibe mensagem e retorna código do erro
37      printf("Erro ao se conectar\n\n");
38      return 97;
39    }
40
41    Sleep(1);
42
43    //Envia mensagem pedindo as coordenadas do TST
44    send(winsock, "CoordenadasTST", 14, 0);
45
46    //Limpa a variavel de retorno
47    memset(buffer2, 0, 1024);
48
49    //Recebe as coordenadas do TST
50    bytes = recv(winsock, buffer2, 1024, 0);
51
52    //Se ocorreu algum erro no recebimento, exibe mensagem de conexão
          perdida e retorna código do erro
53    if (bytes == -1) {
54      printf("Conexão perdida\n\n");
55      return 96;
56    }
57    //valida se as coordenadas recebidas são iguais as coordenadas da base
          de dados
58     if(strcmp(buffer2,coordinate)==0){
59
60          //Exibe mensage de validação
61          printf("Coordenada do TST valida\n\nEnviando atribuicoes!!\n\n")
                ;
62
63          //Envia os atributos da atribução
64          send(winsock, buffer, strlen(buffer), 0);
65
66          //Limpa a variavel de retorno
67          memset(buffer2, 0, 1024);
68
69          //Recebe o retorno do TST
70          bytes = recv(winsock, buffer2, 1024, 0);
71
72          // Se ocorreu algum erro no recebimento, exibe mensagem de conex
                ão perdida e retorna código do erro
73              if (bytes == -1) {
74              printf("Conexão perdida\n\n");
75              return 96;
76          }
77          //Se ocorreu alguma falha na criação do arquivo, exibe mensagem
                de falha e retorna código do erro
```

```
78          if (strcmp(buffer2,"Falha")==0){
79              printf("Falha ao abrir o arquivo\n\n");
80              return 95;
81          }
82          //Se tudo ocorreu bem retorna 1
83          else if(strcmp(buffer2,"Enviado")==0){
84                return 1;
85          }
86      //Se a coordenada não bate, exibe mensagem e retorna código do erro
87     }else{
88          printf("Coordenada do TST não condiz com a especificada!\n\n");
89          return 94;
90     }
91     //Fecha o socket
92      closesocket(winsock);
93      //Finaliza operações soquetes no windows
94      WSACleanup();
95 }
```