UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE COMPUTAÇÃO
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

GIOVANI HENRIQUE BERTUZZO

SERVIÇO PARA LOCALIZAÇÃO EM TEMPO REAL DO TRANSPORTE PÚBLICO COLETIVO

MONOGRAFIA

CAMPO MOURÃO
2019

GIOVANI HENRIQUE BERTUZZO

SERVIÇO PARA LOCALIZAÇÃO EM TEMPO REAL DO TRANSPORTE PÚBLICO COLETIVO

Trabalho de Conclusão de Curso de Graduação apresentado à disciplina de Trabalho de Conclusão de Curso 2, do Curso de Bacharelado em Ciência da Computação do Departamento Acadêmico de Computação da Universidade Tecnológica Federal do Paraná, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

Orientador: Prof. Dr. Rafael Liberato

Coorientador: Prof. Dr. André Luis Schwerz

CAMPO MOURÃO
2019

ATA DE DEFESA DO TRABALHO DE CONCLUSÃO DE CURSO

Às **19:00** do dia **19 de novembro de 2019** foi realizada na sala **E101** da UTFPR-CM a sessão pública da defesa do Trabalho de Conclusão do Curso de Bacharelado em Ciência da Computação do(a) acadêmico(a) **Giovani Henrique Bertuzzo** com o título **Serviço para localização em tempo real do transporte público coletivo**. Estavam presentes, além do(a) acadêmico(a), os membros da banca examinadora composta por: **Prof. Dr. Rafael Liberato Roberto** (orientador(a)), **Prof. Dr. André Luis Schwerz e Prof. Dr. Ivanilton Polato**. Inicialmente, o(a) acadêmico(a) fez a apresentação do seu trabalho, sendo, em seguida, arguido(a) pela banca examinadora. Após as arguições, sem a presença do(a) acadêmico(a), a banca examinadora o(a) considerou _____ na disciplina de Trabalho de Conclusão de Curso 2 e atribuiu, em consenso, a nota _____ (_____). Esse resultado foi comunicado ao (à) acadêmico(a) e aos presentes na sessão pública. A banca examinadora também comunicou ao (à) acadêmico(a) que este resultado fica condicionado à entrega da versão final dentro dos padrões e da documentação exigida pela UTFPR ao professor responsável do TCC no prazo de **onze dias**. Em seguida foi encerrada a sessão e, para constar, foi lavrada a presente Ata que segue assinada pelos membros da banca examinadora, após lida e considerada conforme.

Observações:

_____

_____

_____

_____

Campo Mourão, **19 de novembro de 2019**

_____
Prof. Dr. André Luis Schwerz
Membro 1

_____
Prof. Dr. Ivanilton Polato
Membro 2

_____
Prof. Dr. Rafael Liberato Roberto
Orientador

A ata de defesa assinada encontra-se na coordenação do curso.

Resumo

Bertuzzo, Giovani Henrique. Serviço para localização em tempo real do transporte público coletivo. 2019. 39. f. Monografia (Curso de Bacharelado em Ciência da Computação), Universidade Tecnológica Federal do Paraná. Campo Mourão, 2019.

O transporte público é um dos principais agentes responsáveis por garantir à população urbana seu direito de ir e vir. As rotas e itinerários das linhas dos veículos geralmente são longas e naturalmente podem ocorrer imprevistos durante as viagens, como acidentes no trajeto. Problemas deste âmbito, fazem com que os horários programados das linhas não sejam precisos, com isso alguns usuários acabam chegando atrasados e outros esperam por muito tempo nos pontos de embarque. Em vista disto, este trabalho tem como objetivo, a implementação de um serviço que disponibiliza a localização em tempo real dos veículos do transporte público da cidade de Campo Mourão. Para a construção do serviço, foi realizada a implementação de um sistema distribuído que recebe mensagens de rastreadores, processa estas mensagens, armazena e disponibiliza as informações contidas na mensagem em uma *Application Programming Interface* (API) RESTful. Os resultados dos testes mostram que o serviço atenderia satisfatoriamente à demanda de uma cidade do porte de Campo Mourão em que a frota não passa de 50 veículos.
Palavras-chaves: Mobilidade urbana, Cidades inteligentes, Internet das Coisas

Abstract

Bertuzzo, Giovani Henrique. Real-time location service for collective public transportation. 2019. 39. f. Monograph (Undergradute Program in Computer Science), Federal University of Technology – Paraná. Campo Mourão, PR, Brazil, 2019.

Public transportation is one of the major players responsible for guaranteeing the urban population the right to come and go. The bus routes and itineraries are usually long, as well as, naturally, unforeseen events can occur during the trip, for instance, accidents on the way. Such issues make the scheduled lines inaccurate, which causes some passengers delay or a long time waiting at the boarding platform. Therefore, this work aims the implementation of a service that provides a real-time location of the public transportation bus in the city of Campo Mourão. For the services development, a distributed system has been implemented, which receives messages from set trackers, process them and also it stores and provides the information contained in the message in a RESTful API. The test results show that the service would satisfactorily meet the demand of a city the size of Campo Mourão where the fleet is no more than 50 vehicles.
Keywords: Urban mobility, Smart cities, Internet of Things

Lista de figuras

Lista de tabelas

Siglas

API: *Application Programming Interface*
CPU: *Central Process Unit*
GPRS: *General Packet Radio Services*
GPS: *Global Positioning System*
GSM: *Global System for Mobile Communications*
INCT: Instituto Nacional de Ciência e Tecnologia
IoT: *Internet of Things*
IP: *Internet Protocol*
ITS: *Inteligent Transport Systems and Services*
JSON: *JavaScript Object Notation*
REST: *Representational State Transfer*
TCP: *Transmission Control Protocol*
TIC: Tecnologias da Informação e Comunicação
URBS: Urbanização de Curitiba S/A
UTFPR: Universidade Tecnológica Federal do Paraná

Sumário

Capítulo 1

Introdução

Com a evolução da tecnologia, em especial no contexto de Internet das Coisas, novas soluções têm sido propostas para resolver problemas do mundo contemporâneo. O emprego dessas tecnologias no cenário urbano deu origem ao conceito de Cidades Inteligentes. As cidades inteligentes visam a utilização da tecnologia para promover melhor qualidade de vida para a população, crescimento econômico e sustentabilidade. A mobilidade urbana, especialmente o transporte público, é uma das áreas carentes de soluções inovadoras para melhorar a qualidade de vida dos cidadãos. No Brasil grande parte da população urbana é dependente dos meios de transporte público para realizar as tarefas cotidianas. Entretanto, a maioria das empresas de transporte público não atendem aos requisitos básicos de forma eficiente (RODRIGUES, 2014). As rotas dos ônibus são longas e acontecem diversos imprevistos durante as viagens, como acidentes no trajeto. Estes problemas fazem com que os horários programados das linhas não sejam precisos. Alguns usuários chegam atrasados nos pontos de embarque e perdem o ônibus enquanto outros esperam por muito tempo nos pontos de embarque, sem ter nenhum tipo de informação precisa sobre quando o ônibus irá passar.

Diante deste cenário, este trabalho tem como objetivo desenvolver um serviço que possa ser utilizado para a disponibilização de localização em tempo real dos ônibus do transporte público da cidade de Campo Mourão. Os dados produzidos por este serviço possibilitam a criação de novas soluções tecnológicas que podem contribuir com a mobilidade urbana e qualidade de vida para os cidadãos. Para validar este trabalho, submetemos o serviço a dois testes. No primeiro teste, planejamos diferentes cenários de execução para avaliar qual a quantidade máxima de ônibus que o serviço suportaria. No segundo teste, avaliamos o comportamento do serviço ao ser consumido por uma aplicação real. O segundo teste foi fruto de uma parceria com uma *startup* de Campo Mourão que possui uma aplicação no ramo de transporte público.

O restante do trabalho está organizado como se segue. No Capítulo 2 são apresentados

alguns conceitos teóricos utilizados no trabalho. O Capítulo 3 é responsável por mostrar alguns trabalhos semelhantes a este. O foco deste trabalho está no Capítulo 4, no qual é apresentado uma sequência de passos realizados durante a execução deste trabalho. No Capítulo 5, é apresentado os resultados obtidos na execução de diferentes testes do sistema. Por fim, o Capítulo 6 apresenta as conclusões finais.

Capítulo 2

Fundamentação Teórica

Neste capítulo são apresentados os principais conceitos da área de atuação deste trabalho. Na primeira subseção será abordado o tema *Internet of Things* (IoT), em seguida serão discutidos sobre Cidades Inteligentes, dentro deste tópico serão abordados os conceitos de Mobilidade Urbana e, por fim, afunilando um pouco mais sobre a área de meios de transportes, serão abordadas algumas características dos *Inteligent Transport Systems and Services* (ITS).

2.1. Internet das Coisas (IoT)

Diferente do que se imaginava há alguns anos, a Internet não se tornou apenas uma rede de computadores, mas evoluiu para uma rede de objetos. Esses objetos podem ser de diferentes tipos e tamanhos, como simples sensores, veículos, prédios, câmeras e até mesmo pessoas (PATEL et al., 2016).

Imaginando este cenário onde bilhões de objetos possam detectar, comunicar e compartilhar informações sobre os seus dados, aumentando a capacidade de interação entre os mesmos é que surgiu o conceito de IoT. Segundo (PATEL et al., 2016), a IoT pode ser dividida em 3 categorias, (i) Pessoas para pessoas, (ii) Pessoas para máquinas/coisas e (iii) Máquinas/coisas para máquinas/coisas, todos eles interagindo através da Internet. Vale ressaltar que a comunicação destes objetos, em sua grande maioria é feita sem fio, o que viabiliza ainda mais este ecossistema.

Considerando todos estes objetos trocando informações entre si, com toda informação proveniente de outras coisas/objetos organizada, torna-se mais fácil o desenvolvimento de novos serviços, otimizando o uso dos recursos que já existem e tornando mais inteligente o entendimento e desenvolvimento das cidades.

2.2. Cidades Inteligentes

A crescente migração das zonas rurais para as zonas urbanas ocorrida no último século gerou uma sobrecarga e aglomeração de grande parte da população nas cidades. Esta aglomeração fez com que surgissem diversos problemas, geralmente relacionados à infraestrutura das cidades. Uma alternativa para solucionar estes problemas é a transformação das cidades em espaços mais inteligentes, usando diferentes tecnologias para enfrentar os problemas cruciais ligados à vida urbana, como trânsito, poluição e aglomeração de cidades.

O conceito de cidade inteligente não é definido adequadamente pela literatura, porém alguns estudos afirmam que a inteligência de uma cidade pode ser medida através de seis indicadores, sendo: *smart economy*, *smart mobility*, *smart governance*, *smart environment*, *smart living* e *smart people*, Vanolo (2014).

Segundo Dameri (2013), o principal impulsionador do nascimento e desenvolvimento de uma cidade inteligente é a tecnologia. Especialmente às Tecnologias da Informação e Comunicação (TIC), que permitem conectar diferentes atores na área urbana e fornecer serviços digitais, tanto por instituições privadas como públicas. As universidades, instituições de pesquisa e empresas de alta tecnologia são os principais atores desse modelo: eles são os primeiros a desenvolverem ideias de cidades inteligentes, usando suas competências para planejar e implementar soluções inteligentes.

2.2.1. Mobilidade Urbana

A mobilidade urbana diz respeito às condições de deslocamento da população no espaço geográfico das cidades. O termo é geralmente empregado para referir-se ao trânsito de veículos e também de pedestres, podendo ser através do transporte individual, ou através do uso de transportes coletivos (PENA, 2018).

No Brasil, uma grande transformação na mobilidade começou a acontecer em meados de 1950[1], quando o processo de urbanização intensificou-se, aumentando o uso de veículos motorizados.

Um problema relacionado a mobilidade urbana ocorre quando a proporção do número de automóveis pelo número de habitantes resulta em uma média baixa. No Brasil os números são preocupantes. A má qualidade do transporte público e a redução de impostos por parte do Governo Federal, foi responsável pelo aumento expressivo no número de veículos particulares. Segundo (PENA, 2018), entre os anos de 2002 e 2012, enquanto a população brasileira aumentou 12,2%, o número de veículos registrou um crescimento de 138,6%.

[1] <https://vestibular.uol.com.br/resumo-das-disciplinas/atualidades/mobilidade-urbana-no-brasil-transporte-publico-tem-pouco-investimento-e-a-preferencia-ainda-e-do-carro.htm>

2.2.2. Sistemas de Transportes Inteligentes

As redes urbanas visam facilitar aos usuários uma melhor experiência nos serviços por meio de plataformas inteligentes, como os *Inteligent Transport Systems and Services* (ITS). Os ITS se concentra na aquisição de informações, detecção, controle, processamento de dados e encaminhamentos para dispositivos por meio de interfaces de aplicativos específicas do usuário. A utilidade dos ITS é aprimorada ainda mais através da *Internet of Things* (IoT), que promove uma melhor conectividade com todos dispositivos. Os ITS incluem estratégias para gerenciamento de tráfego, navegação, rastreamento de usuários e processamento coordenado de informações para disponibilizar serviços de qualidade aos usuários (SHARMA et al., 2018).

A evolução da IoT melhorou ainda mais os aplicativos existentes dos ITS, conectando a maioria dos dispositivos aos servidores de suporte a dados (ARASTEH et al., 2016). ITS, em combinação com a IoT, pode suportar conectividade Veículo a Veículo (V2V), bem como Veículo à Infraestrutura (V2I) através de conexões sem fio (LUO et al., 2016). Os ITS podem ser utilizados principalmente para aprimorar a experiência do usuário e aumentar a sua segurança.

Capítulo 3

Trabalhos Relacionados

Este capítulo apresenta alguns serviços semelhantes ao do trabalho realizado. Na primeira subseção será apresentado o sistema Olho Vivo da empresa SPTrans, responsável pelo transporte público da cidade de São Paulo. Em seguida, será apresentado o sistema Itibus, utilizado na cidade de Curitiba. Por fim, é apresentado um projeto de pesquisa colaborativo hospedado pelo Instituto Nacional de Ciência e Tecnologia (INCT) sobre a Internet do Futuro para Cidades Inteligentes, o InterSCity.

3.1. Olho Vivo SPTrans

O Olho Vivo é um sistema que foi desenvolvido pela SPTrans no ano de 2008 na cidade de São Paulo. O objetivo principal do sistema é fornecer informações em tempo real com a localização dos veículos do transporte público da cidade de São Paulo. Dentre suas funcionalidades é possível saber o tempo da viagem, a velocidade média dos veículos, a localização em tempo real dos ônibus e uma estimativa de quando irão passar num ponto específico de embarque (SPTRANS, 2019).

O Olho Vivo oferece ainda uma *Application Programming Interface* (API) para acesso aos dados do sistema, voltada para desenvolvedores. Por meio desta API, podem ser efetuadas consultas que disponibilizam informações aos usuários de outros sistemas. Para utilização da API, é necessário solicitar um *token* de acesso, obtido pós realização de um cadastro junto a SPTrans.

A grande semelhança deste trabalho com o sistema Olho Vivo é o objetivo de disponibilizar uma API com os dados do transporte público coletivo para desenvolvedores poderem produzir aplicações que auxiliem a vida dos usuários.

Figura 3.1. Sistema Olho Vivo SPTrans.

3.2. Curitiba: Itibus

O Itibus é um sistema web que mostra em tempo real a localização dos ônibus do transporte público coletivo da cidade de Curitiba. O ItiBus foi desenvolvido pela área de tecnologia de informação da empresa Urbanização de Curitiba S/A (URBS).

Na tela, um marcador aponta onde você está e mostra todos os pontos de parada num raio de 100 metros, com as linhas de ônibus que passam por cada um deles. No ano de 2017, de janeiro até agosto foram realizadas quase 70 mil consultas ao sistema (URBS, 2017).

Pelo Itibus é possível consultar o trajeto das linhas de três formas: através do nome da linha do ônibus, pelo nome de ruas ou localização espacial, no qual o passageiro seleciona uma área no mapa, e o aplicativo sugere as linhas que circulam nas ruas próximas à região selecionada. Além do trajeto, são mostrados todos os pontos ao longo do percurso com a descrição do endereço e os horários e ainda é possível saber a exata localização do veículo em seu itinerário com seu prefixo (CURITIBA, 2014).

O sistema de transporte coletivo de Curitiba foi um dos primeiros no Brasil a adotar a utilização de diferentes tecnologias para se obter um Sistema de Transportes Inteligente. Uma das consequências geradas pela boa utilização das diferentes tecnologias, foi ser considerado um dos transportes público coletivo mais eficiente do Brasil, o que é comprovado por uma série de prêmios recebidos (CURITIBA, 2018).

3.3. InterSCity

O InterSCity é um projeto de pesquisa colaborativo hospedado pelo INCT sobre a Internet do Futuro para Cidades Inteligentes. O projeto conta com a colaboração de 9 instituições brasileiras e parceiros internacionais. O projeto possui pesquisas em três grandes áreas: Rede

e computação distribuída de alto desempenho, Engenharia de Software para a Internet do Futuro e Análise e modelagem matemática para a Internet do Futuro em cidades inteligentes (Batista et al., 2016).

O projeto visa desenvolver uma plataforma integrada de código aberto que contenha todos os principais componentes para o desenvolvimento de aplicativos robustos, integrados e sofisticados para as cidades inteligentes do futuro. Dentre os seis principais objetivos do projeto, este trabalho se enquadra na implementação e avaliação de uma nova arquitetura de software para sistema distribuído, com foco em larga escala (Batista et al., 2016).

O projeto ainda possui uma seção onde é possível fazer o download de datasets públicos. Um dos datasets é um arquivo de 1.3GB de dados no formato CSV ou XML que contém a simulação de 4 milhões de veículos (carros e ônibus) que se deslocam pela cidade de São Paulo durante um dia. Em outro dataset disponível para download podem ser baixados os dados que representam as viagens realizadas por 2183 linhas de ônibus da cidade de São Paulo coletadas durante 8 dias, contendo informações como a identificação das linhas de ônibus, a frequência de partidas e a velocidade média entre as arestas que ligam dois pontos de ônibus (INCT, 2018).

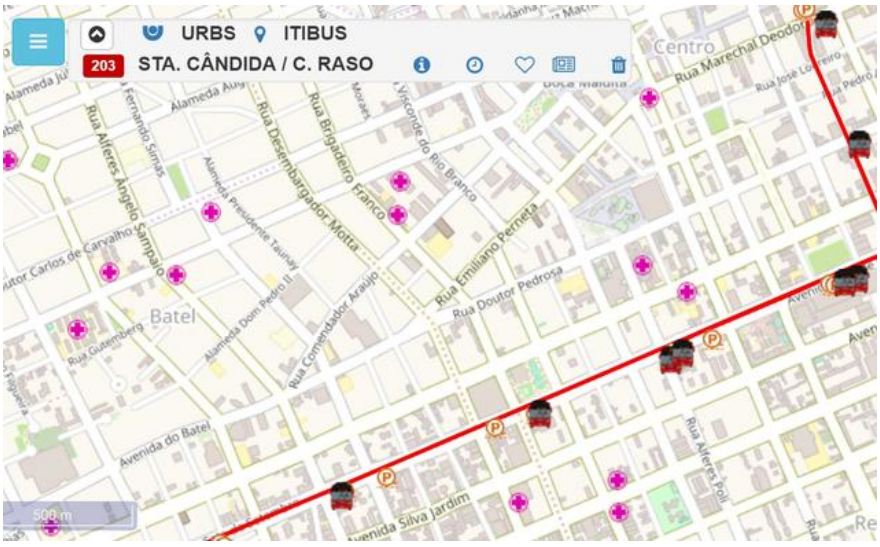

Figura 3.2. Sistema Itibus da URBS.

Capítulo 4

Serviço para Localização em Tempo Real do Transporte Público

Este trabalho consiste no desenvolvimento de um serviço a ser utilizado na coleta e disponibilização de localização em tempo real dos ônibus do transporte público coletivo de Campo Mourão. Em síntese, o serviço possui três responsabilidades: (a) receber as mensagens com informações da localização dos ônibus; (b) processar e armazenar as mensagens; e (c) disponibilizar os dados de localização por meio de uma API. O envio das informações da localização do ônibus deve ser realizado por um equipamento que possua um GPS e seja capaz de enviar as mensagens por meio da rede de telefonia celular. O desenvolvimento deste equipamento está fora do escopo deste trabalho e, por motivos de clareza, o chamaremos de rastreador.

Os dados disponibilizados pelo serviço proposto criam novas oportunidades para o desenvolvimento de soluções tecnológicas voltadas ao transporte público. Um exemplo claro de oportunidade gerada a partir deste serviço é o desenvolvimento de uma aplicação que forneça ao cidadão informações em tempo real sobre o transporte público, tais como os pontos mais próximos, as rotas, localização do ônibus, estimativa de tempo até o próximo ônibus, entre outras.

Durante o desenvolvimento deste trabalho foram estabelecidas parcerias com três empresas, sendo elas: uma empresa de tecnologia no ramo de rastreamento, uma empresa no ramo de transporte público e uma *startup* que possui uma aplicação no ramo de transporte público. Os detalhes dessas parcerias encontram-se nas seções a seguir. O restante do capítulo está organizado da seguinte forma: a Seção 4.1 apresenta a metodologia utilizada no desenvolvimento do trabalho, a Seção 4.2 apresenta os detalhes da implementação do serviço e, por fim, a Seção 4.3 descreve o planejamento dos testes realizados.

4.1. Metodologia

O fluxograma ilustrado na Figura 4.1 apresenta as etapas desenvolvidas para a realização do trabalho.

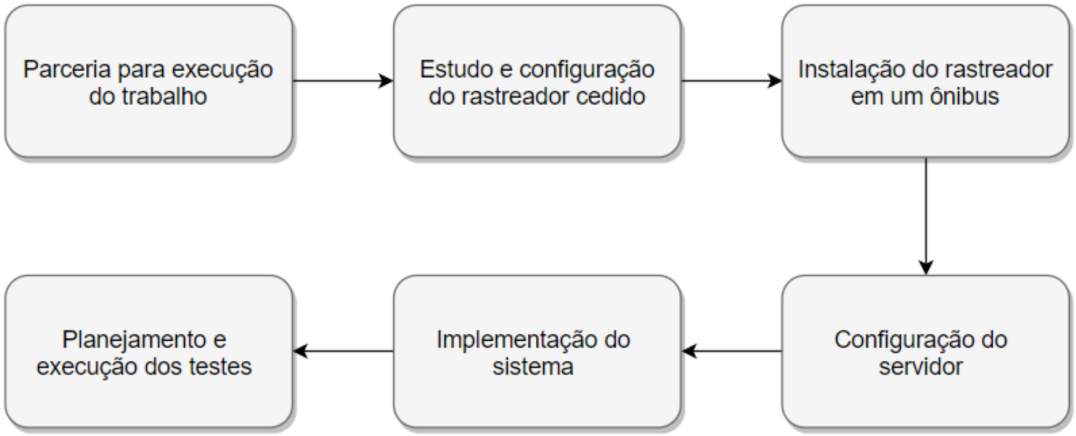

Figura 4.1. Fluxograma da metodologia.

Parcerias para execução do trabalho

A primeira etapa foi buscar parcerias com uma empresa de tecnologia atuante no ramo de rastreamento de veículos e uma empresa de transporte público. Nestas parcerias, a Universidade Tecnológica Federal do Paraná (UTFPR) disponibilizou a infraestrutura necessária para a hospedagem do serviço. A empresa de tecnologia disponibilizou um rastreador utilizado em seu sistema de segurança e a empresa de transporte público permitiu a instalação em um ônibus de sua frota.

Estudo e configuração do rastreador cedido

O modelo de rastreador cedido pela empresa de tecnologia, foi o ST300HD da empresa Suntech. A linha de rastreadores da classe ST300 utiliza principalmente as tecnologias *General Packet Radio Services* (GPRS) e *Global Positioning System* (GPS). O módulo GPS recebe as informações de latitude e longitude dos satélites em órbita terrestre, estas coordenadas são processadas dentro do equipamento unindo com outras informações e enviados seguindo o protocolo de comunicação privado e confidencial determinado pelo fabricante. As mensagens são enviadas ao servidor utilizando pacotes do tipo GPRS com o protocolo de comunicação *Transmission Control Protocol* (TCP) funcionando sobre a estrutura de rede *Global System for Mobile Communications* (GSM). Se não houver cobertura GPRS automaticamente o

módulo armazena estas posições (até 2000 leituras) e descarrega as mesmas assim que a conexão for estabelecida.

Para configuração do rastreador, a empresa disponibilizou o software do fabricante. As principais configurações foram as definições de qual tipo de conexão seria utilizado (TCP ou UDP), o endereço IP e a porta que o rastreador solicita nova conexão e, consequentemente, envia as mensagens de localização. O rastreador também pode ser configurado por meio de mensagens de texto SMS seguindo o protocolo do fabricante.

Destacam-se as seguintes tecnologias utilizadas no rastreador:

1. **GSM:** É uma tecnologia desenvolvida para comunicação com foco em dispositivos móveis sendo usada extensivamente em muitas partes do mundo (WILDE, 2010). O rastreador ST300HD utiliza esta tecnologia de rede para tráfego dos dados.
2. **GPRS:** É um tipo de pacote minimizado, desenvolvido especificamente para dispositivos móveis, que melhora drasticamente a taxa de transferência dos dados em redes GSM (SOININEN, 2003). O rastreador ST300HD utiliza esta tecnologia para realizar a transição dos dados para o servidor.
3. **GPS:** É um mecanismo de posicionamento por satélite, proprietário do governo dos EUA, o GPS fornece aos usuários serviços de posicionamento, retornando a latitude, longitude, data, hora e outras informações. O serviço é dividido em duas categorias, a civil que está disponível gratuitamente para todos os usuários em todo o mundo e a militar, disponível para as forças armadas americanas e aliados (U.S.,GOVERNMENT, 1980).
4. **TCP:** É um dos principais protocolos de comunicação utilizado na Internet. Altamente confiável, o TCP é orientado a conexão e permite que dois dispositivos troquem dados através de um canal ponto a ponto, podendo ser utilizado em redes de comunicação de computador comutadas por pacotes (POSTEL, 1981).

Instalação do rastreador em um ônibus

Após configuração, o rastreador foi instalado em um veículo da frota de ônibus da empresa de transporte público. O veículo disponibilizado, segundo a empresa, foi o com maior frequência de utilização e que diariamente realiza um itinerário dentro da cidade.

O rastreador é ligado diretamente na bateria do veículo, sendo possível consultar as ligações entre os fios da bateria com o do rastreador diretamente no manual do fabricante. A instalação do rastreador no veículo foi realizada por um técnico eletrônico para evitar possíveis problemas. A periodicidade de envio das mensagens de localização foi configurada com o intervalo 30 segundos. No entanto, essa periodicidade pode ser facilmente alterada por meio de uma mensagem SMS para o rastreador.

4.2. Implementação

Esta seção descreve detalhes do desenvolvimento do serviço, inicialmente é apresentada a configuração utilizada no servidor cedido pela UTFPR, posteriormente é apresentada a arquitetura adotada no desenvolvimento do sistema.

4.2.1. Configurações do servidor

Nesta etapa, o Departamento Acadêmico de Computação (DACOM) disponibilizou uma máquina virtual em um de seus servidores com acesso externo para execução do trabalho. Toda configuração foi realizada remotamente por meio de um acesso SSH (do inglês *Secure Socket Shell*). A máquina virtual disponibilizada pelo DACOM veio com o sistema operacional Debian GNU/Linux 9. Nesta máquina virtual, foi instalada a plataforma Docker que divide a arquitetura da aplicação em três *containers*, unidade padrão de software que empacota o código e todas as dependências para que as aplicações sejam executadas de maneira rápida e confiável. A Figura 4.2 ilustra a configuração realizada no servidor.

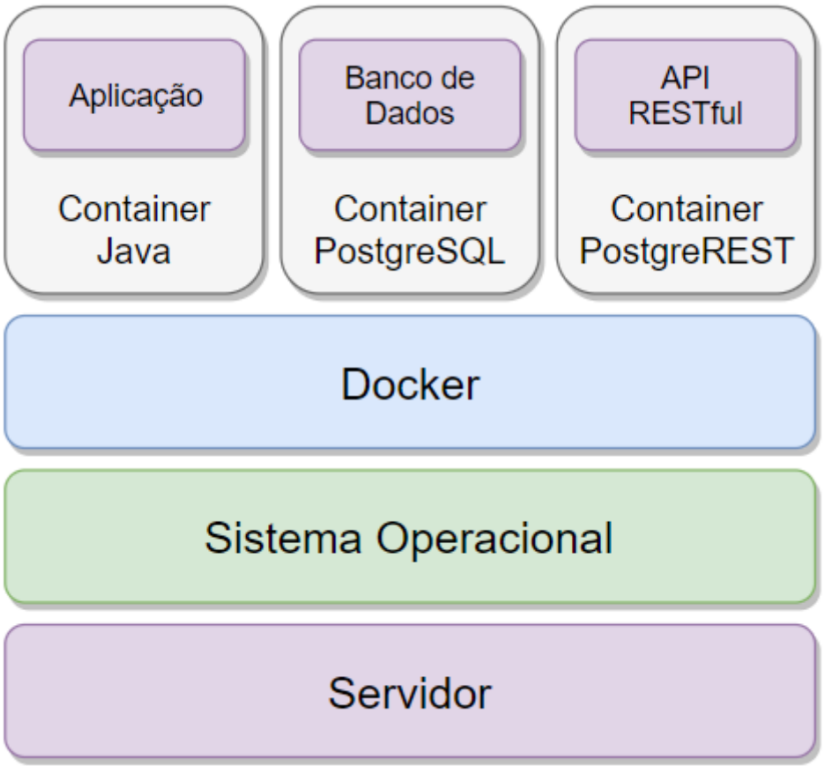

Figura 4.2. Arquitetura do Servidor.

O primeiro *container*, utiliza uma imagem docker Java para o seu funcionamento. É neste *container* que o código da aplicação, desenvolvido na linguagem de programação Java, é executado. O segundo *container*, que é responsável pelo armazenamento dos dados, é um

container que utiliza a imagem docker PostgreSQL. Já o terceiro *container* é responsável pela publicação dos dados recebidos e tratados, gerando uma API RESTful para que os aplicativos possam consumir tais dados. Neste *container* é utilizado uma imagem docker PostgREST.

Cada *container* possui um *Internet Protocol* (IP) próprio, sendo necessário apenas uma comunicação entre os mesmos para que a arquitetura continue a funcionar, ou seja, as três divisões do sistema podem ser desacopladas e continuar funcionando mesmo que em servidores diferentes.

Basicamente o *container* da aplicação é responsável por receber e tratar os dados dos rastreadores e inserir tais dados no banco de dados que fica alocado no segundo *container*. Já o terceiro *container*, é responsável por consultar as informações da base de dados contidas no segundo *container* e gerar uma API que possa ser consultada por outras aplicações.

Segue uma breve descrição das tecnologias utilizadas no servidor:

1. ***Container*:** Unidade padrão de software que empacota o código e todas as suas dependências para que o aplicativo seja executado de maneira rápida e confiável de um ambiente de computação para outro. Uma imagem de *container* do Docker é um pacote de software leve, independente e executável que inclui tudo o que for necessário para execução de um aplicativo: código, tempo de execução, ferramentas do sistema, bibliotecas e configurações do sistema.
2. **Docker:** Plataforma de código aberto, que facilita a criação e administração de ambientes isolados, possibilitando o empacotamento de uma aplicação ou ambiente inteiro dentro de um *container*, fazendo com que o ambiente inteiro torne-se portável para qualquer outro *host* que contenha o Docker instalado.
3. **PostgreSQL:** Sistema gerenciador de banco de dados relacional de código aberto. Existe a mais de 30 anos, sendo fortemente utilizado e atualizado pela sua comunidade. O sistema é conhecido pela robustez dos seus recursos, pelo alto desempenho e confiabilidade dos dados.
4. **PostgREST:** Servidor web independente que transforma qualquer banco de dados PostgreSQL diretamente em uma API RESTful.
5. **Java:** Linguagem de programação orientada a objetos popularmente conhecida por executar em diferentes plataformas. O código gerado nesta linguagem é primeiramente compilado para bytecode e depois executado por uma máquina virtual.

4.2.2. Arquitetura do serviço

A arquitetura do serviço é dividida em três módulos: (i) Módulo de Captura, (ii) Módulo de Processamento e (iii) Módulo de Publicação dos Dados. A Figura 4.3 apresenta uma visão geral da arquitetura projetada em que os dados dos rastreadores são recebidos pelo Módulo de Captura, tratados pelo Módulo de Processamento e disponibilizados pelo Módulo

de Publicação.

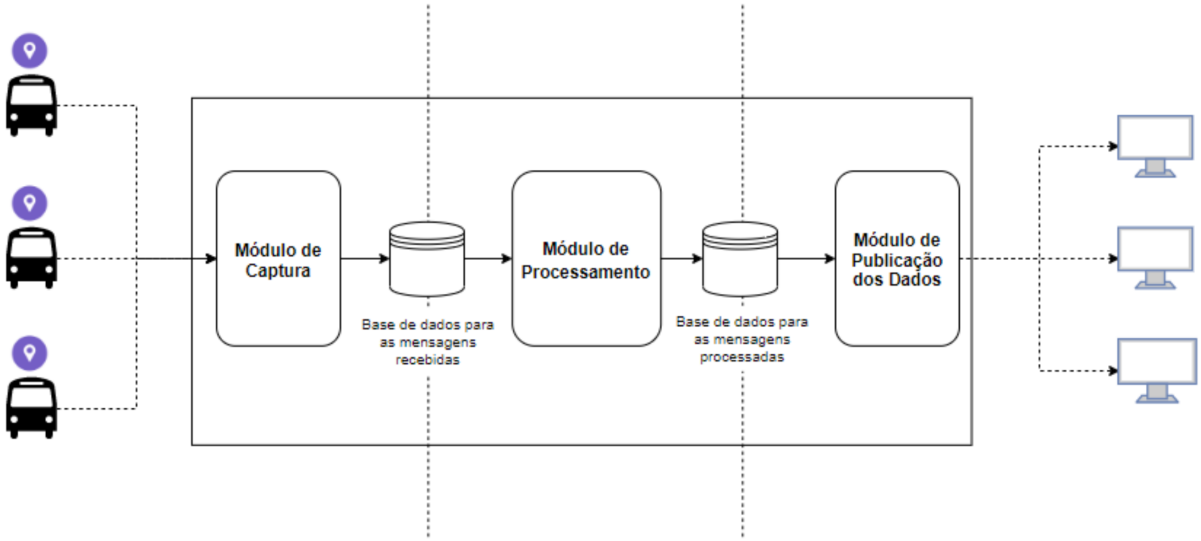

Figura 4.3. Arquitetura Geral do Sistema.

Módulo de Captura

É um *web service multithreading* que estará disponível na internet, ficando alocado no *container* Java conforme explicado na Seção 4.2.1. O serviço é responsável por capturar todas as mensagens enviadas pelos rastreadores (previamente instalados nos ônibus do transporte coletivo e configurados com o IP/porta do servidor).

A principal responsabilidade deste módulo é manter a comunicação com todos os rastreadores. Para isso, o serviço aloca uma *thread* responsável para cada rastreador que solicita uma nova conexão com o serviço. A partir do momento que uma *thread* é instanciada e lhe é atribuída a responsabilidade de receber as mensagens enviadas por um determinado rastreador, um canal de comunicação direto (stream de dados) é mantido entre a *thread* e o rastreador, sem interferência alguma do servidor. A Figura 4.4 ilustra a comunicação direta entre o rastreador e a *thread* após a conexão ser estabelecida. Tal procedimento, evita que o servidor fique com o canal ocupado, deixando-o livre para atender requisições de conexão de novos rastreadores.

Cada *thread* instanciada fica responsável somente por receber as mensagens de um rastreador e adicioná-las em um *array* compartilhado (entre todas as *threads*) em memória. Como cada *thread* recebe muitas mensagens (dependentes da configuração do rastreador) pode ocorrer da mesma não conseguir processar todas mensagens em tempo hábil. Por este motivo foi adotada a estratégia de divisão de responsabilidades no sistema entre os módulos de Captura e Processamento, sendo totalmente desacoplados entre si. Cabendo ao Módulo de Captura, apenas a responsabilidade de receber os dados dos rastreadores e armazená-los

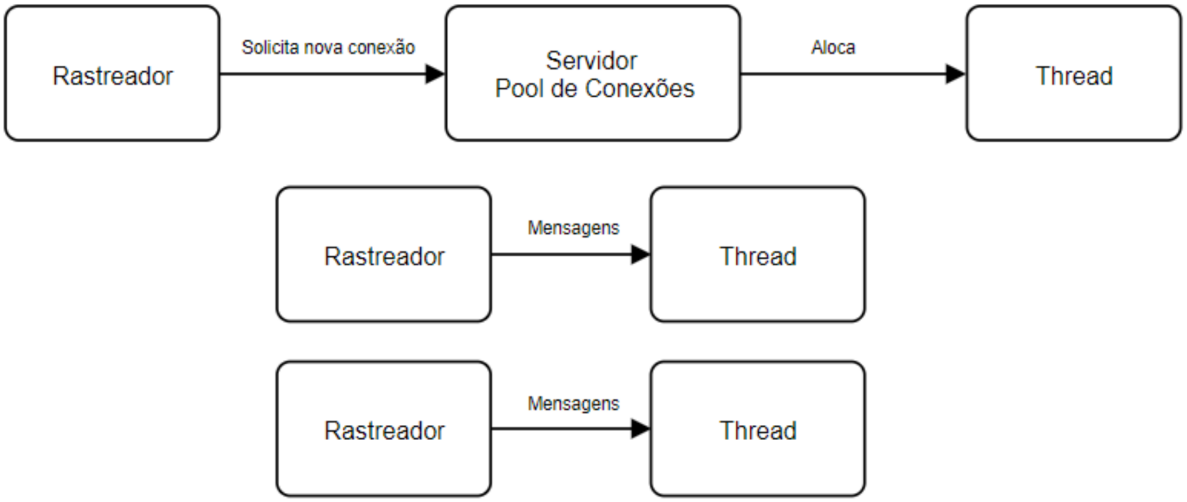

Figura 4.4. Requisição para o Módulo de Captura.

na base de dados para as mensagens recebidas, sem nenhum tipo de pré-processamento.

No próprio Módulo de Captura, existe também uma *thread* única que executa em *loop* infinito, ciclos onde é responsável por retirar todas as mensagens que foram inseridas pelas *threads* no *array* compartilhado, depois faz a inserção no banco em *batch* destas mensagens com o estado "mensagem não processada" (0) na tabela de Mensagens Recebidas, conforme mostra a Figura 4.5. Após a realização de um ciclo completo a *thread* única dorme por N segundos e volta a repetir o ciclo.

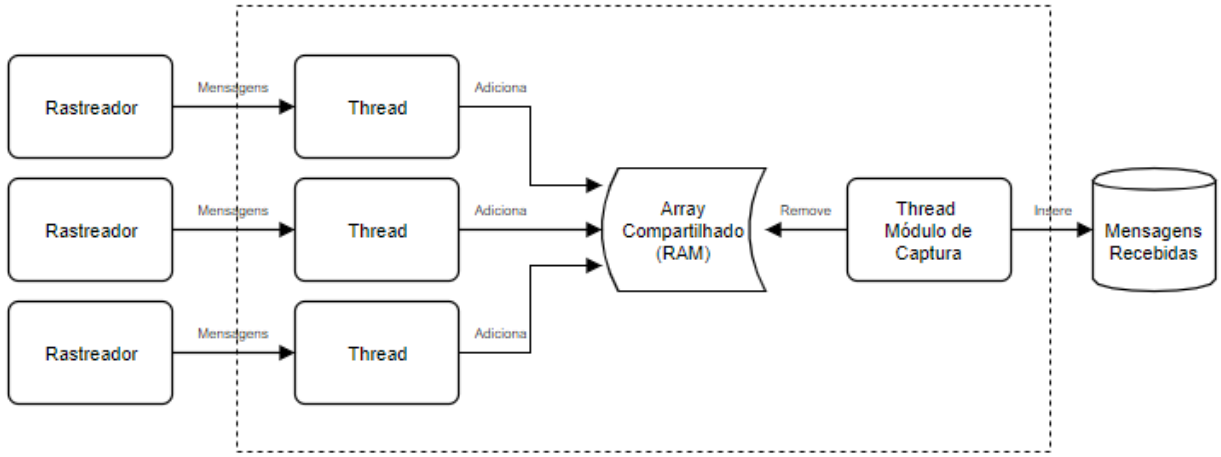

Figura 4.5. Módulo de Captura.

Cada registro da base de dados para as Mensagens Recebidas contém os seguintes atributos:

1. *number_id:* Identificador numérico único para cada mensagem no sistema.
2. *content:* Conteúdo da mensagem recebida.

3. *processed:* Número que informa o estado da mensagem podendo ser: não processada (0), processada (1) ou processando (2).
4. *time:* Data e hora em que a mensagem foi inserida na base de dados.

Módulo de Processamento

Este módulo é responsável pelo processamento das mensagens recebidas no Módulo de Captura e também fica alocado no *container* Java conforme explicado na Seção 4.2.1. Tal módulo é composto principalmente por um *pool* de *threads* que é responsável por alocar uma quantidade suficiente de *threads* processadoras que extraem as informações das mensagens recebidas que ainda não foram processadas. O módulo também é composto por uma *thread* única que remove as mensagens processadas de um *array* compartilhado entre as *threads* processadoras e realiza a inserção em *batch* dessas mensagens na tabela de Mensagens Processadas durante a execução de um ciclo, conforme mostrado na Figura 4.6.

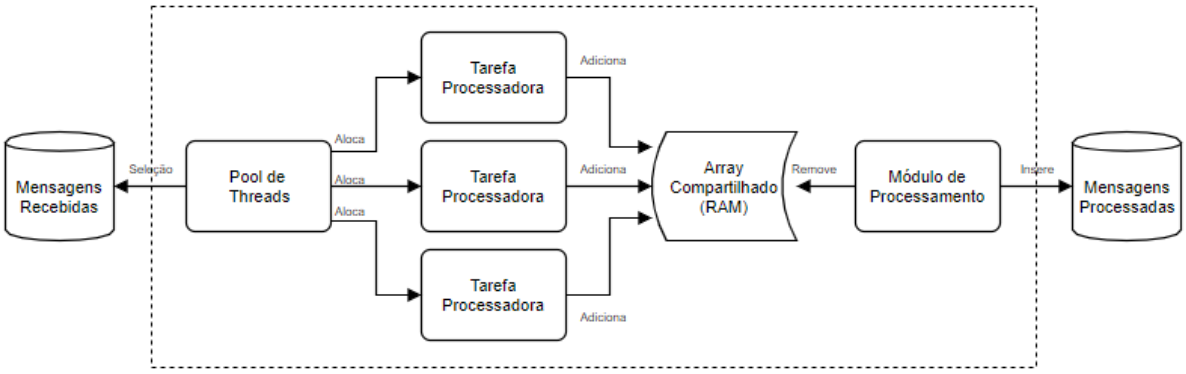

Figura 4.6. Módulo de Processamento.

A ação de processar a mensagem consiste em interpretá-la seguindo o protocolo de comunicação do fabricante do rastreador utilizado. O sistema suporta diferentes modelos de rastreadores, sendo necessário somente a implementação de uma nova classe que estende uma interface genérica, possuindo obrigatoriamente nas mensagens os seguintes atributos: *id* do rastreador, data e hora da mensagem, latitude e longitude.

Durante um ciclo de execução deste módulo são realizadas as seguintes tarefas:

1. **Seleção das mensagens não processadas:** O primeiro passo consiste em consultar quais mensagens da tabela de Mensagens Recebidas ainda não foram processadas (campo processed = 0).
2. **Atualização das mensagens à processar:** Sabendo quais são estas mensagens, é feita uma atualização para o estado processando (campo processed = 2).
3. **Execução das tarefas:** O *pool* de *threads* aloca e/ou reutiliza uma quantidade suficiente de *threads* processadoras para processar todas estas mensagens, após o

processamento as mensagens são inseridas em um *array* que é compartilhado em nível de memória entre as *threads* processadoras e a *thread* única do Módulo de Processamento.

4. **Remoção das mensagens do *array* compartilhado:** Após as três primeiras etapas, a *thread* única do módulo de processamento retira todas as mensagens (que já foram processadas) do *array* compartilhado para a inserção na tabela de Mensagens Processadas.
5. **Atualização e inserção das mensagens processadas:** Com as mensagens removidas do *array* compartilhado é realizada em uma única transação as operações de atualização na tabela de Mensagens Recebidas (campo processed = 1) e inserção na tabela de Mensagens Processadas. A operação é encapsulado por uma transação para evitar inconsistência dos dados no caso de uma falha no sistema.
6. **Sleep para não sobrecarregar o módulo:** Por fim, o módulo dorme pelo tempo de N segundos até repetir um novo ciclo para não utilizar todo o recurso computacional do servidor.

Cada registro da base de dados para as Mensagens Processadas contém os seguintes atributos:

1. *number_id:* Identificador numérico único para cada mensagem processada.
2. *tracker_id:* Identificador numérico único para cada rastreador.
3. *time:* Data e hora contidas na mensagem enviada pelo rastreador.
4. *latitude:* Latitude contida na mensagem enviada pelo rastreador.
5. *longitude:* Longitude contida na mensagem enviada pelo rastreador.
6. *time_receive:* Data e hora em que a mensagem foi processada e inserida na base de dados.

Módulo de Publicação dos Dados

Este módulo tem a responsabilidade de selecionar os dados processados e armazenados pelo Módulo de Processamento, mais especificamente na tabela de Mensagens Processadas, e deixá-los representados de forma transparente em uma API para consulta em tempo real das aplicações, tal módulo está alocado no *container* PostgREST conforme explicado na Seção 4.2.1. Por se tratar de um sistema em tempo real, os dados não podem possuir atrasos. Esta API recebe requisições de aplicações e deve responder corretamente aos dados solicitados, conforme ilustrado na Figura 4.7.

Para realizar este processo é utilizada a ferramenta PostgREST, que transforma qualquer base de dados PostgreSQL em uma API RESTful. Um dos fatores cruciais na decisão por implementar um serviço *Representational State Transfer* (REST), se dá ao fato da necessidade do sistema estar sempre disponível e ter processamento rápido das requisições do cliente. E devido ao fato do REST ser um protocolo sem estado e utilizar diferentes formas

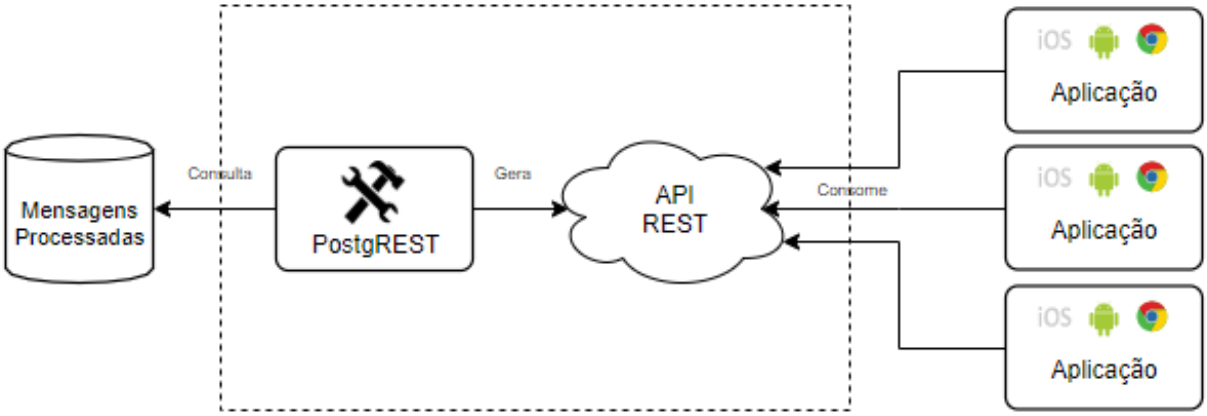

Figura 4.7. Módulo de Publicação dos Dados.

de transição dos dados nas requisições (nesta API é utilizado o formato *JavaScript Object Notation* (JSON) que é bastante leve) foi adotado este modelo arquitetural para compor este módulo (FIELDING, 2000).

Com a intenção de agilizar ainda mais as respostas das requisições, na base de dados foi criada uma *view* que consulta a tabela de Mensagens Processadas, retornando somente a última mensagem processada para cada rastreador diferente, ou seja, não é necessário consultar todos os registros da tabela de Mensagens Processadas para saber qual a mensagem mais recente. Essa *view* pode ser acessada através do endereço http://200.134.18.85:19882/last_update.

4.3. Testes

Por fim, para validação deste trabalho, foram realizados alguns testes no sistema. Os módulos de Captura e Processamento foram testados em conjunto, porém com diferentes configurações. Para sua execução foi implementado um simulador de rastreador utilizando a linguagem de programação Java. Por fim foi desenvolvido um teste final com o sistema completo, onde o serviço foi disponibilizado para ser consumido por uma aplicação real.

Teste de Estresse

O teste de estresse consiste no processo de colocar um sistema em condições extremas para verificar a sua robustez e detectar diversos problemas relacionados à carga, como por exemplo, vazamentos de memória e outros conflitos. Para realização do teste o sistema é submetido a diferentes cargas, iniciando com uma carga normal até uma carga extrema. Durante o teste são avaliados diferentes parâmetros como, a utilização da *Central Process Unit* (CPU), falha no banco de dados e até mesmo a eficiência do projeto de software (JIANG; HASSAN, 2015).

Simulador do Rastreador

Para realizar o teste de estresse, desenvolvemos um simulador do rastreador na linguagem de programação Java, que simula o envio de mensagens idênticas do rastreador ST300HD. O simulador utiliza, como fonte de dados, as mensagens originadas do rastreador ST300HD que foram armazenadas no decorrer do trabalho. A partir desta fonte de dados, o simulador solicita conexão com o servidor e inicia o envio das mensagens, tal como o rastreador. Neste código é possível simular uma quantidade de rastreadores arbitrárias, e também escolher a periodicidade em que as mensagens são enviadas, fator determinante para o funcionamento do sistema.

```
1 String host1 = "200.134.18.85";
2 int porta = 19880;
3 File arquivo1 = new File("LOG_RASTREAMENTO_STATUS2.csv");
4
5 int numThreads = 100; //NUMERO DE THREADS (RASTREADORES SIMULADOS)
6 int timeInterval = 5; //TEMPO DE INTERVALO EM SEGUNDOS
7
8 for(int i=0;i<numThreads;i++){
9     Rastreador r1 = new Rastreador(("Tracker "+i), host1, porta,
          arquivo1, 1000*timeInterval, true);
10    r1.start();
11 }
```

Configurações dos Testes

Para execução dos testes, foram estabelecidas diferentes combinações dos parâmetros para uma melhor perspectiva de comportamento do sistema. No Módulo de Captura e Módulo de Processamento todos os testes foram realizados com a duração de 10 minutos, os parâmetros que se diferem são:

1. *Quantidade de Rastreadores:* A quantidade de rastreadores simulados executando simultaneamente.

Quantidade de Rastreadores
100
1.000
5.000
10.000

2. *Periodicidade das Mensagens:* O intervalo entre uma mensagem e outra em que cada rastreador envia para o servidor.

Periodicidade das Mensagens
5 segundos
10 segundos
30 segundos

3. *Time Sleep (Módulo de Captura):* O tempo em que a *thread* única do Módulo de Captura dorme após encerrar um ciclo.

Time Sleep (Módulo de Captura)
1 segundo
5 segundos
10 segundos

4. *Time Sleep (Módulo de Processamento):* O tempo em que a *thread* única do Módulo de Processamento dorme após encerrar e antes de iniciar um novo ciclo.

Time Sleep (Módulo de Processamento)
1 segundo
5 segundos
10 segundos

As respostas a serem obtidas com os testes no Módulo de Captura são:

- Número total de mensagens recebidas.
- Número total de ciclos executados no período de 10 minutos.
- Tempo total de execução da *thread* única, ignorando o tempo de *sleep*.
- Tempo médio de execução em um ciclo da *thread* única.

As respostas a serem obtidas com os testes no Módulo de Processamento são:

- Número total de mensagens processadas.
- Número total de ciclos executados no período de 10 minutos.
- Tempo total de execução da *thread* única, ignorando o tempo de *sleep*.
- Tempo médio de execução em um ciclo da *thread* única.

Capítulo 5

Resultados e Discussões

O planejamento dos testes do Módulo de Captura e do Módulo de Processamento foram realizados no laboratório da UTFPR, sendo utilizados um total de 14 computadores diferentes. A capacidade de processamento e utilização de memória de cada computador do laboratório era de até 750 rastreadores simulados. Antes de realizar a bateria de testes no Módulo de Captura e Processamento, foi feito um teste incremental para encontrar o ponto de *crash* do sistema. Depois do teste incremental foi realizado a execução das 36 diferentes combinações de testes, conforme explicado na Seção 4.3.

A máquina virtual em que o servidor foi alocado possuía as seguintes configurações: um processador modelo Intel(R) Xeon(R) CPU E5620, com 2.4GHz de frequência, memória RAM de 4GB, e 74GB de disco rígido.

5.1. Teste Incremental Módulo de Captura e Processamento

Antes de executar o planejamento de testes, foi realizado um teste incremental com as seguintes configurações:

- As tabelas de Mensagens Recebidas e Mensagens Processadas do banco de dados foram zeradas antes da execução.
- Após o início do teste, a cada minuto 750 novos rastreadores simulados eram instanciados.
- A duração total do teste foi de 14 minutos.
- No último minuto do teste, 10.500 rastreadores simulados estavam executando.

Na Figura 5.1 é possível observar o gráfico de desempenho da *thread* única no Módulo de Captura e o gráfico de desempenho da *thread* única no Módulo de Processamento.

Durante a execução do teste, também foi observado regularmente a utilização de

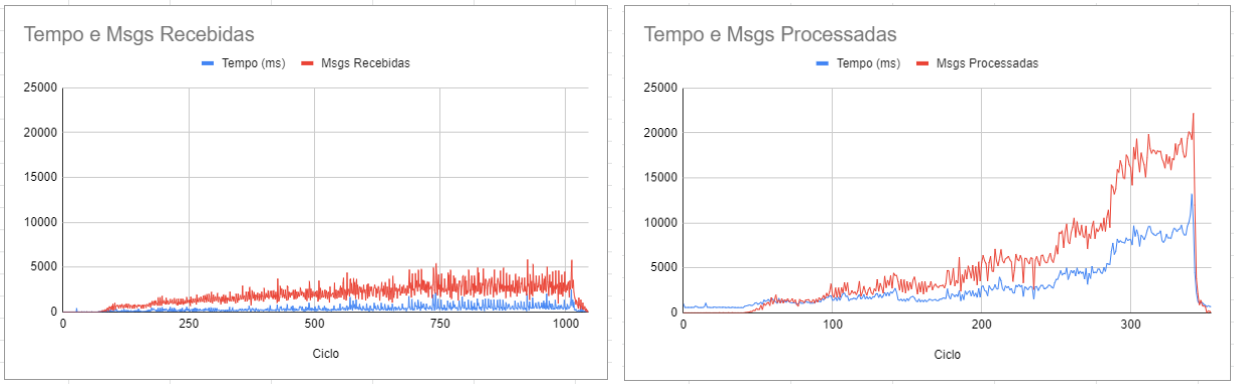

Figura 5.1. Gráfico com resultado do Teste Incremental.

CPU e utilização de memória do servidor. Analisando o desempenho do processo em execução do Módulo de Captura e o processo em execução do Módulo de Processamento, observou-se que no Módulo de Captura a utilização de memória é mais alta, por manter uma grande quantidade de *threads* em conexão com os rastreadores, porém a utilização da CPU não é algo significante. Já no Módulo de Processamento foi observado que com um número elevado de rastreadores simulados (acima de 7000) a *thread* única do Módulo de Processamento começa a apresentar um desempenho ruim, sendo que é notável o aumento significativo de mensagens processadas e principalmente de tempo para execução a cada novo ciclo.

Na Tabela 5.1 é possível observar que o número total de ciclos no Módulo de Captura foi aproximadamente três vezes maior que no Módulo de Processamento, proporção semelhante em relação ao tempo total executado (sem considerar o tempo de *sleep*). Outra informação importante é o tempo médio de execução de um único ciclo, que foi aproximadamente dez vezes maior no Módulo de Processamento.

Parâmetro	Módulo de Captura	Módulo de Processamento
Total de Ciclos	1.046	355
Total Mensagens	1.989.302	1.989.302
Tempo Total Executando	382,82 seg	1.081,16 seg
Tempo Médio Ciclo	365,99 ms	3.045,43 ms

Tabela 5.1. Tabela com resultados do Teste Incremental.

5.2. Testes Módulo de Captura e Processamento

Para melhor compreensão do desempenho do sistema, o teste foi dividido em quatro baterias, considerando o número de rastreadores simulados executados simultaneamente. A primeira bateria foi realizada com 100, a segunda com 1.000, a terceira com 5.000 e a última com 10.000.

A primeira coluna de cada tabela dos resultados contém um identificador para o número do teste, a coluna seguinte é a periodicidade (em segundos) em que as mensagens

eram enviadas por cada rastreador simulado, depois o *time sleep* (em segundos), o número total de mensagens, o número de ciclos executados no período de 10 minutos, o tempo total de execução (em milisegundos) do teste desconsiderando o tempo de *sleep*, e por fim o tempo médio de cada ciclo (em milisegundos).

Antes de iniciar cada um dos testes, algumas configurações eram feitas:

- Restaurar as tabelas de Mensagens Recebidas e Mensagens Processadas do banco de dados.
- Imediatamente após o início dos testes, instanciar todos os rastreadores simulados.
- Após 10 minutos interromper todos rastreadores simulados.

5.2.1. Teste com 100 Rastreadores Simulados

A primeira bateria de testes foi executada sem apresentar nenhum problema de desempenho, o tempo total de execução de cada teste individual ficou dentro dos limites esperados e o tempo médio de execução de um ciclo ficou consideravelmente baixo.

Resultados Módulo de Captura - 100 Rastreadores							
Id	Rastreadores	Period	Sleep	Total Msgs	Ciclos	Tempo Total	Tempo Médio
1	100	5	1	12.145	624	7.436	11,91
2	100	5	5	12.107	128	4.117	32,16
3	100	5	10	12.107	63	2.667	42,33
4	100	10	1	6.110	624	3.403	5,45
5	100	10	5	6.107	126	2.650	21,03
6	100	10	10	6.105	63	1.953	31
7	100	30	1	2.044	616	2.566	4,16
8	100	30	5	2.093	125	1.169	9,35
9	100	30	10	2.073	64	1.138	17,78

Tabela 5.2. Tabela com resultado dos testes com 100 rastreadores no Módulo de Captura.

Resultados Módulo de Processamento - 100 Rastreadores							
Id	Rastreadores	Period	Sleep	Total Msgs	Ciclos	Tempo Total	Tempo Médio
1	100	5	1	12.145	610	19.884	32,59
2	100	5	5	12.107	127	9.781	77,01
3	100	5	10	12.107	63	7.996	126,92
4	100	10	1	6.110	613	13.450	21,94
5	100	10	5	6.107	125	7.955	63,64
6	100	10	10	6.105	62	6.457	104,14
7	100	30	1	2.044	619	10.931	17,65
8	100	30	5	2.093	124	4.639	37,41
9	100	30	10	2.073	64	3.953	61,76

Tabela 5.3. Tabela com resultado dos testes com 100 rastreadores no Módulo de Processamento.

As cidades de médio porte se enquadrariam neste cenário, na qual a frota do transporte público coletivo não ultrapassaria os 100 veículos. Atualmente, a frota do transporte público da cidade de Campo Mourão é de aproximadamente 40 veículos.

Considerando o conjunto de teste número 2 é possível notar por meio da Figura 5.2 que o Módulo de Captura apresenta um padrão na relação entre a quantidade de mensagens recebidas por tempo de processamento do ciclo. No Módulo de Processamento é possível observar na Figura 5.2 alguns picos isolados, onde o tempo de execução de um ciclo ultrapassa os 100 milisegundos, além do grande pico nos ciclos iniciais após a execução do módulo, que foi comum em todos os testes.

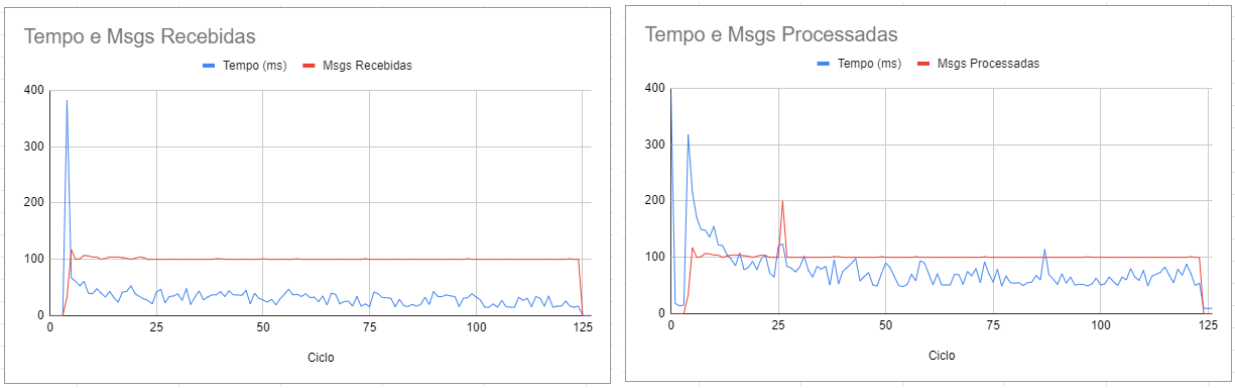

Figura 5.2. Gráfico com resultado do Teste Número 2.

5.2.2. Teste com 1.000 Rastreadores Simulados

A segunda bateria de testes apresentou resultados satisfatórios, mais uma vez o Módulo de Captura ficou com um tempo médio de execução de ciclo muito baixo. Já o Módulo de Processamento começou a apresentar sinais de que o peso do processamento do sistema poderia estar nele, após analisar o tempo médio de ciclo superior a 500 milisegundos no teste número 12.

Resultados Módulo de Captura - 1.000 Rastreadores							
Id	Rastreadores	Period	Sleep	Total Msgs	Ciclos	Tempo Total	Tempo Médio
10	1.000	5	1	119.177	611	15.642	25,6
11	1.000	5	5	120.405	125	10.776	86,2
12	1.000	5	10	119.546	62	9.855	158,95
13	1.000	10	1	60.313	617	11.572	18,75
14	1.000	10	5	59.998	124	6.429	51,84
15	1.000	10	10	60.358	63	5.921	93,98
16	1.000	30	1	20.362	626	5.731	9,15
17	1.000	30	5	20.390	125	3.311	26,48
18	1.000	30	10	20.309	64	2.856	44,62

Tabela 5.4. Tabela com resultado dos testes com 1.000 rastreadores no Módulo de Captura.

Resultados Módulo de Processamento - 1.000 Rastreadores							
Id	Rastreadores	Period	Sleep	Total Msgs	Ciclos	Tempo Total	Tempo Médio
10	1.000	5	1	119.177	571	55.638	97.43
11	1.000	5	5	120.405	120	36.518	304,31
12	1.000	5	10	119.546	60	33.323	555,38
13	1.000	10	1	60.313	591	38.558	65,24
14	1.000	10	5	59.998	122	21.261	174,27
15	1.000	10	10	60.358	62	19.123	308,43
16	1.000	30	1	20.362	612	19.482	31,83
17	1.000	30	5	20.390	124	10.069	81,2
18	1.000	30	10	20.309	63	8.924	141,65

Tabela 5.5. Tabela com resultado dos testes com 1.000 rastreadores no Módulo de Processamento.

Ao analisar o gráfico da Figura 5.3 notamos que mesmo aumentando a quantidade de rastreadores e consequentemente a quantidade de mensagens, o aumento no tempo de processamento não foi tão significativo no Módulo de Captura.

Porém ao analisar a mesma Figura 5.3 na parte do Módulo de Processamento, é possível notar outro sinal de que possivelmente o gargalo estaria no Módulo de Processamento. Existem alguns picos no módulo ao processar duas mil mensagens, e o tempo de execução nos ciclos com estes picos aumenta significativamente, diferente do Módulo de Captura.

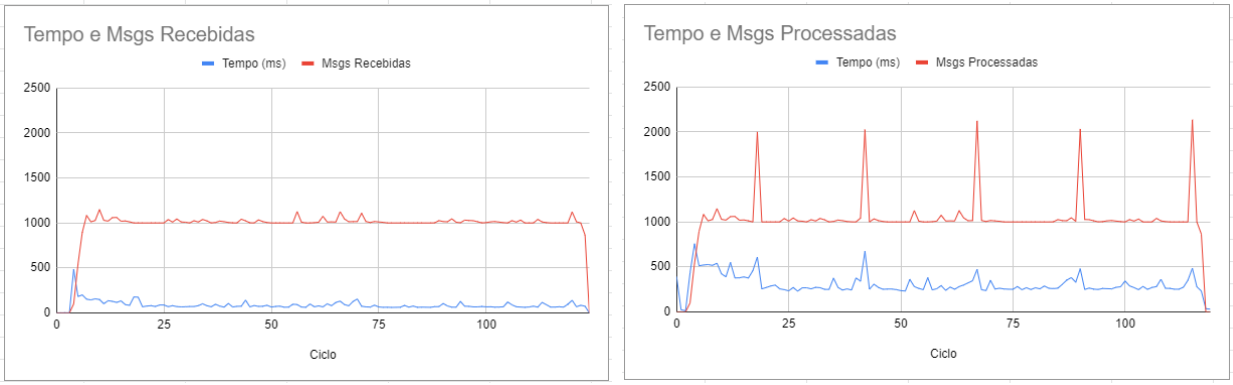

Figura 5.3. Gráfico com resultado do Teste Número 11.

5.2.3. Teste com 5.000 Rastreadores Simulados

Durante os três primeiros testes da terceira bateria, foi possível observar que o sistema, considerando as configurações de hardware do servidor, começava a apresentar indícios de que estava se aproximando do seu limite de rastreadores. Mais uma vez o Módulo de Captura se comportou de forma estável, levando em consideração o número elevado de mensagens recebidas.

No teste de número 19, o Módulo de Processamento ficou com uma média de 590 milisegundos por ciclo, ficando próximo do tempo de *sleep* que era de 1 segundo. No teste

número 20 ultrapassou 2 segundos e no teste número 21 se aproximou dos 4 segundos.

Resultados Módulo de Captura - 5.000 Rastreadores							
Id	Rastreadores	Period	Sleep	Total Msgs	Ciclos	Tempo Total	Tempo Médio
19	5.000	5	1	596.825	561	83.959	149,65
20	5.000	5	5	603.409	118	59.507	504,29
21	5.000	5	10	600.743	59	57.379	972,52
22	5.000	10	1	303.358	609	37.152	61
23	5.000	10	5	302.607	125	26.838	214,7
24	5.000	10	10	302.367	63	24.441	387,95
25	5.000	30	1	101.880	630	19.375	30,75
26	5.000	30	5	107.413	135	11.794	87,36
27	5.000	30	10	102.589	63	10.293	163,38

Tabela 5.6. Tabela com resultado dos testes com 5.000 rastreadores no Módulo de Captura.

Resultados Módulo de Processamento - 5.000 Rastreadores							
Id	Rastreadores	Period	Sleep	Total Msgs	Ciclos	Tempo Total	Tempo Médio
19	5.000	5	1	596.825	407	240.158	590,06
20	5.000	5	5	603.409	92	187.419	2.037,16
21	5.000	5	10	600.743	47	176.036	3.745,44
22	5.000	10	1	303.358	510	135.238	265,17
23	5.000	10	5	302.607	114	86.768	761,12
24	5.000	10	10	302.367	57	80.514	1.412,52
25	5.000	30	1	101.880	592	56.458	95,36
26	5.000	30	5	107.413	130	34.195	263,03
27	5.000	30	10	102.589	61	30.993	508,08

Tabela 5.7. Tabela com resultado dos testes com 5.000 rastreadores no Módulo de Processamento.

Analisando o gráfico da Figura 5.4 do teste número 19, é possível notar que ao aumentar a carga no sistema, o primeiro módulo tem um comportamento aceitável pois a diferença visual na relação entre mensagens recebidas por tempo de execução do ciclo, é algo que cresce proporcionalmente, já no segundo módulo observamos visualmente essa diferença diminuir, ou seja, conforme o número de mensagens processadas aumentam, o tempo de execução aumenta significativamente dando a entender que caso ocorra um maior aumento da carga, o tempo (em milisegundos) de execução do Módulo de Processamento será superior a quantidade de mensagens processadas, algo inviável para o sistema.

5.2.4. Teste com 10.000 Rastreadores Simulados

Na última bateria, os três primeiros testes foram realizados com condições extremas. Máxima carga de rastreadores com periodicidade de envio baixa. O Módulo de Captura conseguiu suprir a necessidade em todos os testes. Já no Módulo de Processamento, provando que os

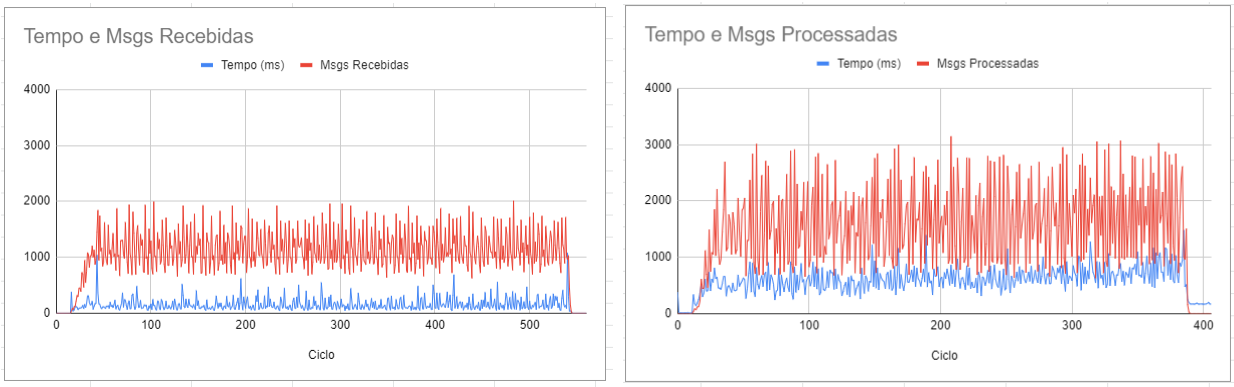

Figura 5.4. Gráfico com resultado do Teste Número 19.

indícios fornecidos visualmente pelos gráficos anteriores não estavam errados, foi observado um desempenho preocupante.

Observando a Tabela 5.9, o teste de número 30 apresentou um tempo médio de ciclo extremamente alto, confrontando com o principal objetivo do sistema que é fornecer a localização em tempo real. Outro caso importante a ser observado é o teste de número 29, onde o Módulo de Processamento não conseguiu processar todas as mensagens recebidas pelo Módulo de Captura no tempo delimitado pelo teste (10 minutos).

Resultados Módulo de Captura - 10.000 Rastreadores							
Id	Rastreadores	Period	Sleep	Total Msgs	Ciclos	Tempo Total	Tempo Médio
28	10.000	5	1	1.201.766	395	262.385	664,26
29	10.000	5	5	1.212.429	95	192.788	2.029,34
30	10.000	5	10	1.208.965	48	190.869	3.976,43
31	10.000	10	1	612.378	524	126.458	241,33
32	10.000	10	5	610.970	121	70.676	584,09
33	10.000	10	10	610.972	59	69.008	1.169,62
34	10.000	30	1	208.534	626	30.115	48,1
35	10.000	30	5	206.067	129	22.167	171,83
36	10.000	30	10	209.883	64	20.873	326,14

Tabela 5.8. Tabela com resultado dos testes com 10.000 rastreadores no Módulo de Captura.

Analisando os gráficos do conjunto de teste 28 e considerando a utilização do mesmo por um período longo, fica nítido que a carga excessiva extrapolou os limites do sistema (considerando as limitações de hardware do servidor) apesar de ter apresentado falhas somente em um teste da bateria. Por meio da Figura 5.5 podemos observar que o Módulo de Processamento apresenta um aumento de tempo muito grande comparado ao aumento do número de mensagens processadas, dando a entender que este aumento aconteceria de forma exponencial caso a duração do teste ultrapassasse os 10 minutos, tempo previamente estipulado.

Resultados Módulo de Processamento - 10.000 Rastreadores							
Id	Rastreadores	Period	Sleep	Total Msgs	Ciclos	Tempo Total	Tempo Médio
28	10.000	5	1	1.201.766	119	541.607	4.551,31
29	10.000	5	5	238.206	12	155.719	12.976,58
30	10.000	5	10	1.208.965	21	474.139	22.578,04
31	10.000	10	1	612.378	297	362.961	1.222,09
32	10.000	10	5	610.970	93	208.502	2.241,95
33	10.000	10	10	610.972	47	190.806	4.059,7
34	10.000	30	1	208.534	554	109.735	198,07
35	10.000	30	5	206.067	119	62.034	521,29
36	10.000	30	10	209.883	61	56.326	923,37

Tabela 5.9. Tabela com resultado dos testes com 10.000 rastreadores no Módulo de Processamento.

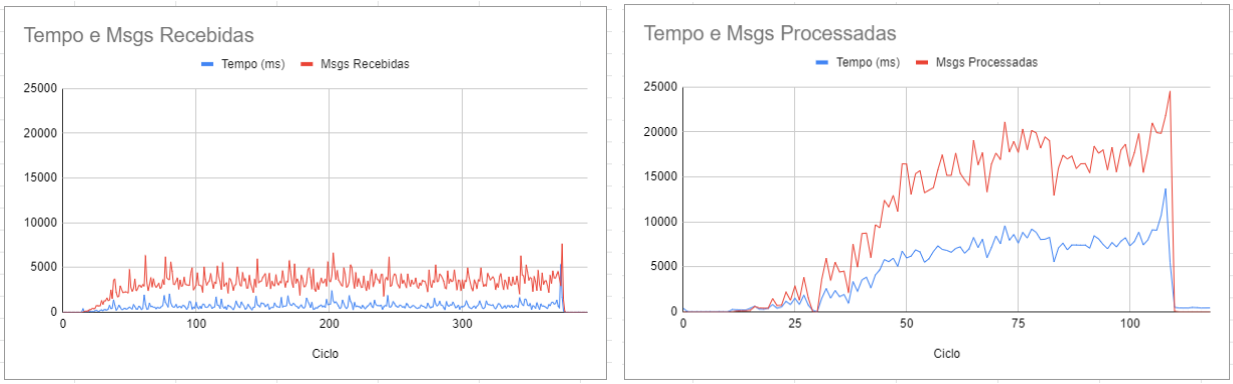

Figura 5.5. Gráfico com resultado do Teste Número 28.

5.3. Teste Completo do Serviço

Para finalizar foi realizado um último teste, avaliando o sistema completo em execução. Observamos o comportamento do serviço no fornecimento de dados para uma aplicação real. Para isso, contamos com a colaboração de uma startup que possui uma aplicação para o transporte público. A startup criou um *branch* de sua aplicação para consumir os dados do nosso serviço. O serviço comportou-se como esperado, permitindo acompanhar o deslocamento em tempo real do ônibus. A Figura 5.6 ilustra a interface da aplicação. As circunferências em azul são os pontos de ônibus e a circunferência verde representa a localização do ônibus monitorado.

5.4. Considerações e Ameaças à validade

Os resultados mostraram que o serviço seria satisfatório para atender a frota do transporte urbano de Campo Mourão que é de aproximadamente 40 veículos. No entanto, para atender cidades de grande porte, seria necessário adotar uma solução escalável, principalmente para o módulo de processamento. Além disso, devemos considerar as limitações do hardware disponibilizado para hospedar o serviço desenvolvido.

Figura 5.6. Aplicação consumindo API.

Como uma ameaça a validade, observamos que existe uma relação entre a quantidade de registros nas tabelas e o desempenho do sistema. Em alguns casos que as tabelas possuíam muitos registros, o tempo de execução de um ciclo do sistema durava o dobro do tempo de execução. Para este fator não influenciar diretamente nos resultados dos testes, foi utilizada a estratégia de restaurar as tabelas do banco, antes de iniciar a execução de qualquer conjunto de teste.

Para solucionar esse problema de desempenho, uma estratégia que poderia ser adotada é a divisão dos dados em duas bases, uma *on-line* e uma *off-line*. A base de dados *on-line* conteria apenas os dados mais recentes (por exemplo da última semana), e os demais dados seriam mantidos na base *off-line* sem intervir no funcionamento do sistema.

Capítulo 6

Conclusão

Este trabalho apresentou o desenvolvimento de uma solução em consonância com o conceito de cidades inteligentes. Mais especificamente, o trabalho apresentou uma contribuição na área de transporte público, propondo um serviço que captura e disponibiliza a localização em tempo real dos veículos do transporte público coletivo.

Os resultados obtidos mostram que a arquitetura proposta atende à demanda de uma cidade do porte de Campo Mourão em que a frota não passa de 50 veículos, além da validação do serviço após ser consumido por uma aplicação real sem apresentar problemas. Os resultados também apontaram a hipótese de que o módulo de processamento necessita ser otimizado para atender frotas com mais de 5 mil veículos. No entanto, para verificar essa hipótese, novos testes deveriam ser realizados em um hardware com mais recursos e capacidade computacional.

Como trabalho futuro, pretendemos verificar o comportamento do serviço desenvolvido em redes de rádio frequência que permitem comunicação a longas distâncias com consumo mínimo de energia.

Referências

ARASTEH, Hamidreza; HOSSEINNEZHAD, Vahid; LOIA, Vincenzo; TOMMASETTI, Aurelio; TROISI, Orlando; SHAFIE-KHAH, Miadreza; SIANO, Pierluigi. Iot-based smart cities: a survey. In: . [S.l.: s.n.], 2016.

Batista, D. M.; Goldman, A.; Hirata, R.; Kon, F.; Costa, F. M.; Endler, M. Interscity: Addressing future internet research challenges for smart cities. In: *2016 7th International Conference on the Network of the Future (NOF)*. [S.l.: s.n.], 2016. p. 1–6.

CURITIBA, Prefeitura Municipal. *Aplicativos mostram localização, rotas, pontos e horários de ônibus*. 2014. Disponível em: <https://www.curitiba.pr.gov.br/noticias/aplicativos-mostram-localizacao-rotas-pontos-e-horarios-de-onibus/32178>. Acesso em: 16 out. 2019.

CURITIBA, Prefeitura Municipal. *Curitiba passa São Paulo e é eleita cidade mais conectada e inteligente do país*. 2018. Disponível em: <https://www.curitiba.pr.gov.br/noticias/curitiba-passa-sao-paulo-e-e-eleita-cidade-mais-conectada-e-inteligente-do-pais/47463>. Acesso em: 16 out. 2019.

DAMERI, Renata Paola. Searching for smart city definition: a comprehensive proposal. *International Journal of Computers Technology*, v. 11, n. 5, p. 2544–2551, 2013.

FIELDING, Roy Thomas. *Architectural Styles and the Design of Network-based Software Architectures*. Tese (Doutorado), 2000. AAI9980887.

INCT, InterSCity. *Open Datasets*. 2018. Disponível em: <http://interscity.org/open_data/>. Acesso em: 16 out. 2019.

JIANG, Zhen; HASSAN, Ahmed E. A survey on load testing of large-scale software systems. *IEEE Transactions on Software Engineering*, v. 41, p. 1–1, 11 2015.

LUO, Yugong; XIANG, Yong; CAO, Kun; LI, Keqiang. A dynamic automated lane change maneuver based on vehicle-to-vehicle communication. *Transportation Research Part C: Emerging Technologies*, v. 62, p. 87–102, 01 2016.

PATEL, Keyur K; PATEL, Sunil M et al. Internet of things-iot: definition, characteristics, architecture, enabling technologies, application & future challenges. *International journal of engineering science and computing*, v. 6, n. 5, 2016.

PENA, Rodolfo F. Alves. *Mobilidade urbana no Brasil*. 2018. [Online; accessed 14-Novembro-2018]. Disponível em: <https://www.significados.com.br/mobilidade-urbana/>.

POSTEL, Jon. *Request for Comment 793*. 1981. Disponível em: <https://tools.ietf.org/html/rfc793>. Acesso em: 16 out. 2019.

RODRIGUES, Alex. *Poder Público recebe milhares de reclamações sobre qualidade do transporte*. 2014. Disponível em: <http://agenciabrasil.ebc.com.br/geral/noticia/2014-03/ii-poder-publico-recebe-milhares-de-reclamacoes-quanto-qualidade-do-transpor>. Acesso em: 16 out. 2019.

SHARMA, Vishal; YOU, Ilsun; PAU, Giovanni; COLLOTTA, Mario; LIM, Jae; KIM, Jeongnyeo. Lorawan-based energy-efficient surveillance by drones for intelligent transportation systems. *Energies*, v. 11, 03 2018.

SOININEN, J. *Request for Comment 3574*. 2003. Disponível em: <https://tools.ietf.org/html/rfc3574>. Acesso em: 16 out. 2019.

SPTRANS, S/A. *Guia de Utilização Olho Vivo*. 2019. Disponível em: <http://olhovivo.sptrans.com.br/files/TutorialNovoOlhoVivo.pdf>. Acesso em: 16 out. 2019.

URBS, S/A. *Guia de Utilização Olho Vivo*. 2017. Disponível em: <http://www.urbs.curitiba.pr.gov.br/noticia/em-menos-de-um-ano-uso-do-itibus-dobra-e-chega-a-70-mil-consultas>. Acesso em: 16 out. 2019.

U.S.,GOVERNMENT, Official. *The Global Positioning System*. 1980. Disponível em: <https://www.gps.gov/systems/gps/>. Acesso em: 16 out. 2019.

VANOLO, Alberto. Smartmentality: The smart city as disciplinary strategy. *Urban Studies*, Sage Publications Sage UK: London, England, v. 51, n. 5, p. 883–898, 2014.

WILDE, E. *Request for Comment 5724*. 2010. Disponível em: <https://tools.ietf.org/html/rfc5724>. Acesso em: 16 out. 2019.