Curso de Ciência da Computação
Universidade Estadual de Mato Grosso do Sul

Protótipo de um Sistema de Informação para Georreferenciar Meios de Transporte Público em Dourados

Leandro Souza da Silva

Nielson Fernandes Silva

Profa. Dra. Glaucia Gabriel Sass (Orientadora)

Dourados – MS
2015

Curso de Ciência da Computação
Universidade Estadual de Mato Grosso do Sul

Protótipo de um Sistema de Informação para Georreferenciar Meios de Transporte Público em Dourados

Leandro Souza da Silva

Nielson Fernandes Silva

Novembro de 2015

Banca Examinadora:

Profa. Dra. Glaucia Gabriel Sass (Orientadora)
Engenharia de Software – UEMS

Prof. Dr. Rubens Barbosa Filho
Inteligência Artificial – UEMS

Profa. MSc. Jéssica Bassani de Oliveira
Engenharia de Software – UEMS

Protótipo de um Sistema de Informação para Georreferenciar Meios de Transporte Público em Dourados

Leandro Souza da Silva
Nielson Fernandes Silva

Este exemplar corresponde à redação final da monografia da disciplina Projeto Final de Curso devidamente corrigida e defendida por Leandro Souza da Silva e Nielson Fernandes Silva, e aprovada pela Banca Examinadora, como parte dos requisitos para a obtenção do título de Bacharel em Ciência da Computação.

Dourados, 22 de Novembro de 2015.

Profa. Dra Glaucia Gabriel Sass (Orientadora)

Dedicamos esse trabalho a Deus, por ser nosso refúgio e fortaleza, socorro bem presente na angústia e as nossas famílias que sempre estiveram nos apoiando.

Os únicos homens verdadeiramente felizes são os que buscam uma maneira de serem úteis aos outros.

Albert Schweitzer

AGRADECIMENTOS

Agradeço primeiramente a Deus, pela sua fidelidade e por me sustentar em todos os momentos da minha vida. Agradeço também a meu pai, Elzir Pereira da Silva e a minha mãe Santa Fernandes Carneiro da Silva, pelo amor, incentivo e apoio incondicional.

Agradeço as minhas irmãs, Lisceanne Fernandes Silva e Niely Fernandes Silva, que sempre estiveram dispostas a me ajudar e me apoiaram durante todo esse trajeto.

Agradeço a professora Dra. Glaucia Gabriel Sass por toda contribuição e apoio a esse trabalho e pela paciência.

Agradeço ao Paulo Edson da Silva, que durante meu período de estágio, prontamente ajudou cedendo para testes alguns equipamentos utilizados nesse trabalho acadêmico.

Agradeço ao meu amigo Leandro Souza da Silva, pela parceria e pelas contribuições no meu trajeto acadêmico.

Agradeço aos meus parentes e amigos, que apoiaram o meu trajeto acadêmico. Agradeço a todos que direta ou indiretamente fizeram parte da minha formação, o meu muito obrigado.

Nielson Fernandes Silva

Agradeço primeiramente aos meus pais, que me deram apoio incondicional em todos os momentos da minha vida acadêmica, Laudelino Queiros da Silva e Vera Lúcia Santos Souza da Silva.

Agradeço aos meus amigos da “República”, que conviveram comigo durante os últimos quatro anos.

Agradeço a professora Dra. Glaucia Gabriel Sass por todo empenho e dedicação em orientar e direcionar o andamento desse trabalho.

Agradeço ao meu amigo Nielson Fernandes Silva, pela contribuição e parceria nesse trabalho e na minha jornada acadêmica.

Agradeço ao meu amigo e professor Yuri Mendes Leite, que contribuiu na tradução do resumo.

Agradeço a todos as pessoas, que de uma forma ou de outra, contribuíram para a minha formação como profissional e cidadão.

Leandro Souza da Silva

RESUMO

O incentivo do governo brasileiro pela aquisição de automóveis a partir da década de 1970 trouxe uma série de consequências para sociedade atual, que enfrenta problemas como: o aumentado do número de carros nas vias pavimentadas, a diminuição do número de ônibus em relação ao número de carros, o aumento de agentes poluentes na atmosfera e a falta de informação sobre o transporte público. Em meio a essas dificuldades surge o conceito de Sistemas de Transportes Inteligentes, que tem a finalidade de informatizar, interligar e instrumentalizar a infraestrutura de transporte público urbano, proporcionando assim, informações mais precisas aos usuários de transporte público (ônibus). Apoiado no conceito de aplicação de tecnologias aos problemas do transporte público, foi desenvolvido um protótipo de um Sistema de Informação capaz de georreferenciar em tempo real os meios de transporte público na cidade Dourados. Este trabalho foi dividido em três etapas: estudo das tecnologias que compõem um Sistema de Informação Geográfica Móvel e trabalhos correlatos, projeto do protótipo do Sistema de informação, e por fim, a implementação e teste do protótipo. Na primeira etapa buscou-se estudar, conhecer e determinar através de testes as tecnologias mais viáveis em termos de custo benefício; Na segunda etapa foi construído o projeto do Sistema e definidas as tecnologias utilizadas de acordo com os estudos realizados na primeira etapa; Na terceira e última etapa, implementou-se e testou-se o protótipo do Sistema de Informação, nessa etapa os testes foram realizados em veículos de uma linha dos meios de transporte público de Dourados. A partir dos estudos realizados elaborou-se três diferentes tipos de arquiteturas: arquitetura com transmissão de dados SMS, *socket*/TCP e HTTP. Através dos testes, análise e comparação dos resultados alcançados entre as arquiteturas, notou-se que, a arquitetura com transmissão HTTP possuía às características mais adequadas para representar o Sistema de informação, algumas dessas características são: grande quantidade de dados enviados por unidade de tempo, ordenação forte dos dados buscados de forma assíncrona em rede e perda de dados nula, desde que haja uma conexão estável com a internet. Ao apresentar um conjunto de características significativas para o Sistema de Informação a arquitetura com transmissão HTTP foi implementada e testada.

Palavras-Chave**:** Georreferenciamento. Sistema de Informação. Transporte público. *Smartphone*.

ABSTRACT

The Brazilian government incentive for the acquisition of automobiles in the 70's brought about a series of consequences to the current society which has to face problems such as: the increase in the number of car on paved ways, the decrease in the number of buses compared to the number of cars, the amount of polluting agents in the atmosphere and the lack of information about public transportation. Amidst this difficulty, the concept of Sistemas de Transporte Inteligentes (Smart Transport Systems) emerges and aims to inform, link and instrumentalize the urban public transport infrastructure by providing more precise information to its users (buses). Based on the concept of technology application to the public transport, it has been developed an Information System prototype which is able to georeference in real time the public means of transportation of Dourados. This work was divided into three steps: the study of the technologies that compose a Mobile Geographical Information System and its correlated works, the project of an Information System Prototype, and finally, the prototype implementation and testing. In the first step, it was aimed to know as well as to determine, through probing, the most viable technologies in terms of cost-efficiency. In the second step, it was built the project of the System and the technologies used were defined following the studies carried out in the first step. In the third and last step, it occurred the testing and implementation of the Information System prototype which were performed in vehicles of one of Dourados public transport lines. Taking the studies carried out as a model, it was elaborated three different types of architecture: an architecture with the transmission of SMS data, socket/TCP and HTTP. By means of the analyses, tests and the comparison of the results reached among the architectures, it has been noticed that the architecture with a HTTP transmission had the most adequate characteristics to represent the Information System, and some of its characteristics are: a large amount of data sent by unit of time, clear ordination of searched data in an asynchronous way in the web with null data loss, provided there is a strong internet connection. After exposing a significant set of characteristics to the Information System the architecture with a HTTP transmission was implemented and then tested.

Keywords: Georeferencing, Information System. Public transport. Smartphone.

SUMÁRIO

LISTA DE SIGLAS XIX
LISTAS DE FIGURAS XXI
LISTAS DE QUADROS XXIII
1 INTRODUÇÃO 25
1.1 Objetivos 26
1.2 Estrutura do trabalho 27
2 REFERÊNCIAL TEÓRICO 29
2.1 Sistema de Informação 29
2.2 Tecnologias para Sistemas de Informações Geográficas Móveis 31
2.2.1 Computação em nuvem 32
2.2.2 Banco de dados em nuvem 35
2.2.3 Protocolo HTTP 36
2.2.4 Servidor web 38
2.2.5 Tecnologias de georreferenciamento e transmissão de dados 41
2.2.6 Tecnologias de desenvolvimento 46
2.2.7 Tecnologias de infraestrutura 48
2.3 Trabalhos correlatos 53
2.3.1 Olho Vivo 53
2.3.2 Vá de Ônibus 54
2.3.3 Cadê o ônibus? (SP) 55
3 MATERIAIS E METODOS 57
4 DESENVOLVIMENTO 59
4.1 Avaliação de tecnologias 59
4.1.1 Arquitetura do SI com transmissão de dados via SMS 59
4.1.2 Arquitetura do SI com transmissão de dados via socket TCP 63
4.1.3 Arquitetura do SI com transmissão de dados via HTTP 66
4.1.4 Escolha da melhor arquitetura 70
4.2 Projeto do protótipo do SI 71
4.2.1 Protótipo do servidor de serviços web 72
4.2.2 Protótipo do aplicativo do usuário 75
4.2.3 Protótipo do aplicativo de rastreamento 77
4.3 Implementação do protótipo do SI 79
4.3.1 Protótipo do servidor de serviços web 79
4.3.2 Protótipo do aplicativo do usuário 79
4.3.3 Protótipo do aplicativo de rastreamento 80
5 RESULTADOS 82
6 CONCLUSÃO 86
6.1 Considerações Finais 86
6.2 Trabalhos Futuros 86
REFERÊNCIAS BIBLIOGRÁFICAS 88

LISTA DE SIGLAS

2G – Segunda Geração

3G – Terceira Geração

3GPP – *3rd Generation Partnership Project*

ANTP – Associação Nacional dos Transportes Públicos

API – *Application Programming Interface*

APN – *Access Point Name*

ASCII – *American Standard Code for Information Interchange*

AT – *Attention*

AVL – *Automatic Vehicle Location*

BSS – *Base Station Subsystem*

DbaaS – *Database-as-a-service*

ETSI – *European Telecommunication Standards Institute*

GAE – *Google App Engine*

GLONASSS – *Globalnaya navigatsionnaya sputnikovaya sistema*

GNSS – *Global Navigation Satellite System*

GPRS – *General Packet Radio Service*

GPS – *Global Position System*

GSM – *Global System for Mobile Communication*

HTTP – *Hyper Text Transfer Protocol*

IaaS – *Infrastructure-as-a-Service*

IDE – *Integrated Development Environment*

IETF – *Internet Engineering Task Force*

IMSI – *International Mobile Subscriber Identity*

IP – *Internet Protocol*

IPEA – Instituto de Pesquisa Econômica Aplicada

ISoc – *Internet Society*

JSON – *JavaScript Object Notation*

MMS – *Multimedia Messaging Service*

MS – *Mobile Station*

NIST – *National Institute of Standards and Technology*

NSS – *Network Subsystem*

PaaS – *Plataform-as-Service*

RAM - *Random Access Memory*

SaaS – *Software-as-a-Service*

SI – Sistema de informação

SIG – Sistema de Informação Geográfica

SIM – *Subscriber Identity Module*

SMS – *Short Message Service*

SO – Sistema Operacional

SOAP – *Simple Object Access Protocol*

SPI – *Service-platform-infrastructure*

SQL – *Strutured Query Language*

STI – Sistemas de Transportes Inteligentes

TCP – *Transmission Control Protocol*

URSS – União das Repúblicas Socialistas Soviéticas

USB – *Universal Serial Bus*

WEB – *Word Wide Web*

LISTAS DE FIGURAS

Figura 2.1 – Papéis das aplicações de negócios dos SI 30
Figura 2.2 – Tipos de SI que dão apoio a estratégias de vantagens competitivas 30
Figura 2.3 – Tipos de SI que dão apoio para processos e operações de negócios 31
Figura 2.4 – Tipos de SI que dão suporte na tomada de decisão 31
Figura 2.5 – As características essenciais de computação em nuvem 32
Figura 2.6 – SPI: Serviços oferecidos pelo modelo de computação em nuvem 33
Figura 2.7 – Estrutura de um banco de dados em nuvem 36
Figura 2.8 – Comportamento de requisição-resposta do HTTP 37
Figura 2.9 – Estrutura básica de um servidor web 39
Figura 2.10 – Comunicação entre processos através de *socket* TCP 41
Figura 2.11 – Representação da distribuição de 24 satélites na órbita terrestre 43
Figura 2.12 – Arquitetura da rede GSM 44
Figura 2.13 – Cartão SIM e um dispositivo móvel. 44
Figura 2.14 – Histórico da evolução da rede móvel a cada geração. 45
Figura 2.15 – GPS TK103-2 48
Figura 2.16 – Entrada do Cartão SIM no GPS TK103-2 49
Figura 2.17 – Comando enviado por SMS para o dispositivo 49
Figura 2.18 – *Modem* GSM 51
Figura 2.19 – *Modem* GSM Profissional 51
Figura 2.20 – Mapa do itinerário da linha 2100-10, Terminal Vila 53
Figura 2.21 – Trajeto entre às Avenidas Pompeu Loureiro e Nossa Senhora de Copacabana, realizado pela linha 121- Copacabana - Central (Circular), Plataforma web 54
Figura 2.22 – Mapa do itinerário da linha 2100-10, Terminal Vila Carrão: Praça da Sé, Plataforma móvel 55
Figura 4.1 – Arquitetura do SI com transmissão de dados via SMS 60
Figura 4.2 – Método principal do programa controlador 61
Figura 4.3 – Classe para envio de mensagens pelo programa controlado 62
Figura 4.4 – Arquitetura do SI com transmissão de dados via *socket* TCP 63
Figura 4.5 – Código da implementação do servidor *socket* TCP 64
Figura 4.6 – Saída da execução da aplicação *socket* TCP 65
Figura 4.7 – Arquitetura do SI com transmissão de dados via HTTP 66

Figura 4.8 – Função para capturar as coordenadas geográficas de um dispositivo móvel de 1 em 1 segundo 68
Figura 4.9 – Função para enviar as coordenadas geográficas de um dispositivo móvel para um servidor web 68
Figura 4.10 – Função do servidor de serviços web para receber coordenadas geográficas 69
Figura 4.11 – Confirmação de recebimento das coordenadas geográficas 69
Figura 4.12 – Diagrama de caso do uso do protótipo do servidor web 72
Figura 4.13 – Diagrama de classes do protótipo do servidor web 74
Figura 4.14 – Diagrama de caso de uso do protótipo do aplicativo do usuário 75
Figura 4.15 – Diagrama de classes do projeto do protótipo do aplicativo do usuário 76
Figura 4.16 – Diagrama de caso de uso do protótipo da aplicação de rastreamento 77
Figura 4.17 – Diagrama de classes do protótipo do aplicativo de rastreamento 78
Figura 5.1 – Tela do aplicativo do usuário, obtida durante o teste 83
Figura 5.2 – Tela do aplicativo do usuário com o trajeto do veículo georreferenciado 84
Figura 5.3 – Tela do aplicativo do usuário com o trajeto do veículo georreferenciado 84
Figura 5.4 – Tela do aplicativo do usuário com o trajeto do veículo georreferenciado 85

LISTAS DE QUADROS

Quadro 2.1 – Os métodos internos de solicitações HTTP 38
Quadro 2.2 – Comandos do rastreador GPS TK103-2 50
Quadro 2.3 – Comandos AT 52
Quadro 2.4: Quadro comparativo das funcionalidades entre as aplicações 55
Quadro 4.1 – Características similares entre as arquiteturas 70
Quadro 4.2 – Descrição do caso de uso: Enviar coordenadas 73
Quadro 4.3 – Descrição do caso de uso: Solicitar coordenadas 74
Quadro 4.4 – Descrição do caso de uso: Visualizar localização 76
Quadro 4.5 – Descrição do caso de uso: Iniciar envio de coordenadas 78
Quadro 5.1 – Tempos obtidos durante os testes. 83

1 INTRODUÇÃO

A mobilidade sempre foi uma necessidade básica à sobrevivência do homem ao longo da história, muitos lugares ao redor do mundo tornaram-se extremamente urbanizados com o advento da indústria, com isso, as locomoções internas e externas de pessoas e mercadorias foram dinamizadas a escalas elevadas, esse fenômeno gerou uma profunda transformação socioespacial, que atinge a forma como o mundo se move hoje.

Segundo o IPEA (2011) a partir da década de 1950, as cidades brasileiras passaram a enfrentar transformações profundas na mobilidade urbana e alguns fatores cruciais levaram a essas mudanças: o intenso processo de urbanização e o aumento da frota de veículos motorizados, tanto dos automóveis quanto dos ônibus.

No período entre 1977 e 2005, as grandes regiões metropolitanas do Brasil tiveram uma redução de 68% para 51%, do total de viagens motorizadas feitas em transporte público, já o percentual de viagens feitas com o automóvel teve um aumento significativo de 32% para 49% (IPEA, 2011).

Mesmo com o crescente número dos transportes individuais, os sistemas de ônibus urbanos e metropolitanos são a modalidade de transporte público mais comum no Brasil, operando em cerca de 85% dos municípios da união (IPEA, 2011). Embora os ônibus estejam presentes na maioria dos municípios brasileiros, isso não garante a qualidade nos serviços prestados à população.

Apesar da abrangência do sistema de transporte público brasileiro, este apresenta algumas dificuldades que atinge a maior parte dos municípios que dependem desse serviço, uma série de itens tem feito com que a qualidade nos serviços seja muito baixa: falta de informação sobre os ônibus, custos tarifários abusivos, superlotação e etc.

Os passageiros cientes dos problemas do transporte público passaram a buscar novas alternativas, como o automóvel, a motocicleta e a bicicleta, que são mais atraentes, pois apresentam as vantagens de privacidade, flexibilidade e comodidade. Com o elevado número de transportes individuais nas vias públicas surgem novos problemas urbanos: poluição congestionamentos e acidentes de trânsito, que contribuem para a redução da qualidade de vida.

Como medidas atenuadoras em relação ao trânsito e a falta de qualidade no transporte público, os Governos das principais cidades do país têm investido em trens, metrôs e corredores de ônibus, essas medidas diminuem o trânsito e o tempo das viagens. Contudo, os sistemas de alta capacidade de trens e metrôs demonstram baixa ocorrência entre as cidades, se restringindo a poucas regiões metropolitanas do país (IPEA, 2011).

Apesar dos esforços públicos e privados para melhorar o sistema rodoviário e o transporte público, como se pode observar nas cidades brasileiras o trânsito e as más condições da infraestrutura de transporte são cada vez mais frequentes.

Das dificuldades encontradas no trânsito e nos sistemas de transporte público surge um novo paradigma em infraestrutura de transporte, que se baseia não apenas em investir dinheiro em ampliação de rodovias, mas também em tornar a infraestrutura existente mais eficiente, inteligente, instrumentalizada[1] e interligada. Uma infraestrutura com essas características requer uma mudança nos modelos de negócios para atrair novas ideias, investimentos, bem como novos usuários.

Os Sistemas de Transportes Inteligentes (STI) se mostram propícios na busca de tais características, esses por sua vez consistem em aplicar tecnologias aos problemas do transporte coletivo, como a falta de informação, congestionamentos e contingências (ANTP, 2012).

1.1 Objetivos

Esse trabalho tem como objetivo o desenvolvimento de um protótipo de um Sistema de Informação, capaz de georreferenciar em tempo real os meios de transporte público na cidade de Dourados. Os objetivos específicos são:

a) Estudar conceitos teóricos e tecnologias que darão suporte ao protótipo Sistema de Informação, estabelecendo e comparando as vantagens e desvantagens das diferentes tecnologias.

b) Projetar um protótipo de um Sistema de Informação capaz de localizar meios de transportes públicos, de acordo com os resultados alcançados nos estudos.

[1] Instrumentalizada: referente aos instrumentos utilizados para compor a arquitetura que torna um dado meio de transporte localizável, exemplo: satélites, câmeras, sensores, rastreadores e etc.

c) Implementação e Desenvolvimento do protótipo do Sistema de Informação para finalidade de testes.

1.2 Estrutura do trabalho

No primeiro capítulo desse trabalho são apresentados a introdução e os objetivos geral e específicos.

O segundo capítulo deste trabalho apresenta um conjunto amplo de definições, conceitos e tecnologias. O início do capítulo define o conceito e as várias classificações de um Sistema de Informação. O capítulo apresenta ainda um conjunto de tecnologias que servem de base para um Sistema de Informação Geográfica Móvel, e por fim, são apresentados um conjunto de trabalhos correlatos e suas principais funcionalidades.

No terceiro capítulo é apresentado a forma como o presente trabalho foi dividido, estruturado e realizado.

O quarto capítulo apresenta o desenvolvimento do Sistema de Informação, composto pelas fases de avaliação de tecnologias, projeto e implementação.

O quinto capítulo é composto por resultados dos testes realizados no protótipo do Sistema de Informação desenvolvido.

Finalmente, o sexto capítulo apresenta as conclusões do presente trabalho e sugestões para trabalhos futuros.

2 REFERÊNCIAL TEÓRICO

2.1 Sistema de Informação

De acordo com Stair (2010), “Sistema de Informação (SI) é composto por conjuntos de elementos ou componentes, que podem ser pessoas, dispositivos, ou até mesmo outros sistemas. Os conjuntos e componentes de um SI são inter-relacionados e tem a finalidade de coletar, manipular, armazenar e disseminar dados e informações, de forma a atingir um objetivo”.

Os tipos de dados manipulados, disseminados, processados como também as informações geradas por um SI, podem definir a classificação de um SI.

Um SI que tem por objetivo o georreferenciamento implica que seus elementos ou componentes manipulem, disseminem e processem dados geográficos. Câmara (2005) define um SI com essas características como Sistema de Informação Geográfica (SIG), que “são sistemas que realizam o tratamento computacional de dados geográficos”.

Câmara (2005) ressalta que a principal diferença de um SIG para um SI convencional é a capacidade de armazenar tanto atributos descritivos como as geometrias dos diferentes tipos de dados geográficos.

A forma de obtenção dos dados através dos elementos ou componentes também podem definir a classificação de um SI.

Um SIG que tem por objetivo o georreferenciamento, e que obtém coordenadas geográficas através de dispositivos móveis, é classificado por Tsou (2004), como Sistemas de Informações Geográficas Móveis (SIG Móveis), Tsou (2004) define os SIG Móveis como “Sistemas de Informação compostos por componentes tecnológicos integrados, cuja finalidade é acessar dados espaciais e de localização, baseado em dispositivos móveis”.

Tsou (2004) ressalta que a implantação de um SIG Móvel deve levar em consideração problemas específicos de domínio, como custo e disponibilidade de tecnologias de transmissão de dados, para os dispositivos móveis.

Um SIG Móvel permite uma funcionalidade importante, que é a localização automática de veículos. As coordenadas geográficas são obtidas automaticamente, através de dispositivos móveis, mantidos em veículos ou em qualquer meio de transporte, esses dados são enviados através de tecnologias de transmissão de dados a um centro de controle, Portilo

(2008), define essa funcionalidade como *Automatic Vehicle Location* (AVL – *Automatic Vehicle Location*).

A classificação de um SI, também pode ser determinada por seu objetivo. A ANTP (2012) define STI (Sistemas de Transporte Inteligente) como sistemas que consistem na aplicação de conjuntos de tecnologias, com a finalidade de solucionar problemas comuns do transporte coletivo.

Nos negócios os SI são classificados de acordo com os papéis ou objetivos nas aplicações de negócios (Figura 2.1).

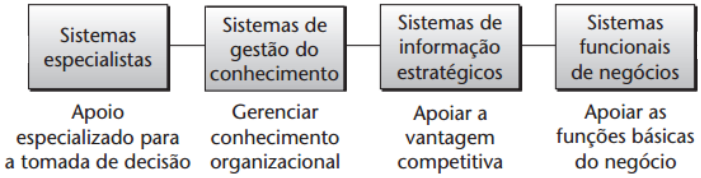
Figura 2.1 – Papéis das aplicações de negócios dos SI

Fonte: O'BRIEN (2011, p. 6).

Os papéis de aplicações dos SI's nos negócios são: suporte de processos e operações de negócios, suporte à tomada de decisões e suporte a estratégias que buscam vantagem competitiva (O'BRIEN, 2011).

Existem diferentes tipos de SI's para cada aplicação de negócios. O'brien (2011) define SI's que dão suporte a estratégias de vantagem competitiva como: Sistemas especialistas, Sistemas de gestão do conhecimento, Sistemas de informação estratégicos e Sistemas funcionais de negócios (Figura 2.2).

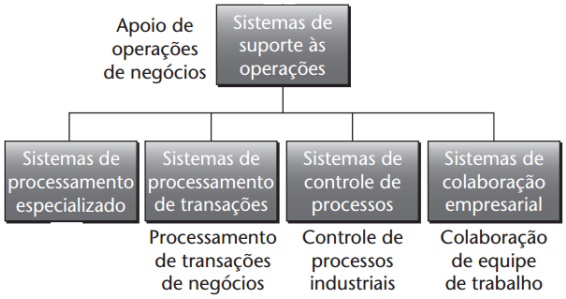
Figura 2.2 – Tipos de SI que dão apoio a estratégias de vantagens competitivas



Fonte: Adaptado de O'BRIEN (2011, p. 11).

De acordo com O'brien (2011), no suporte para processos e operações de negócios, estão definidos os Sistemas de suporte às operações, Sistemas de processamento especializado, Sistemas de processamento de transações, Sistemas de controle de processos e Sistemas de colaboração empresarial (Figura 2.3).

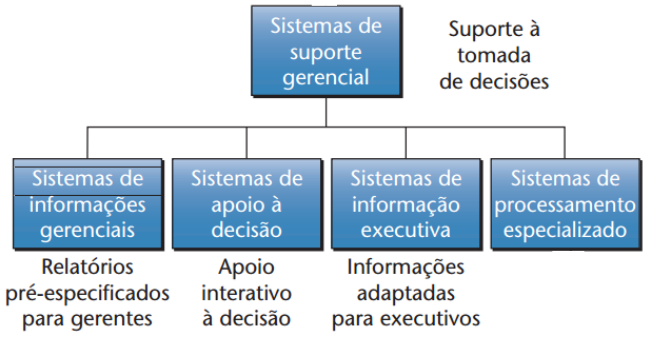
Figura 2.3 – Tipos de SI que dão apoio para processos e operações de negócios



Fonte: Adaptado de O'BRIEN (2011, p. 11).

Por fim, SI's que auxiliam na tomada de decisão são definidos por O'brien (2011) como: Sistemas de suporte gerencial, Sistemas de informações gerenciais, Sistemas de apoio à decisão, Sistemas de informação executiva e Sistemas de processamento especializado (Figura 2.4).

Figura 2.4 – Tipos de SI que dão suporte na tomada de decisão



Fonte: Adaptado de O'BRIEN (2011, p. 11).

Na classificação de SI's definida por O'brien (2011) um SIG Móvel, com funcionalidade AVL é definido como um Sistema especialista, pois em um negócio o SIG Móvel poderá atuar dando suporte especializado na tomada de decisão de uma empresa.

2.2 Tecnologias para Sistemas de Informações Geográficas Móveis

Esta subseção tem como objetivo descrever conceitos teóricos e tecnologias associadas ao desenvolvimento de um SIG Móvel com funcionalidade AVL.

2.2.1 Computação em nuvem

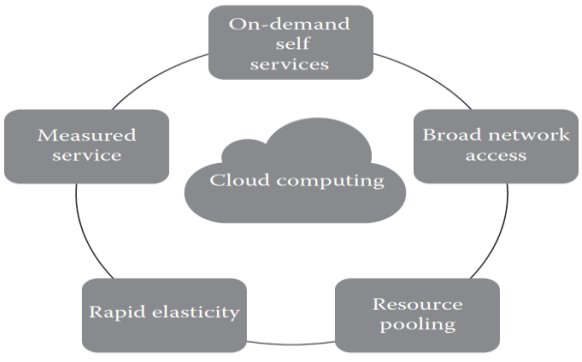
Segundo Chandrasekaran (2015) e Marinescu (2013) a definição formal da computação em nuvem vem do *National Institute of Standards and Technology* (NIST): a computação em nuvem é um modelo para habilitar o acesso por rede ubíqua e sob demanda a um conjunto compartilhado de recursos de computação (como redes, servidores, armazenamento, aplicações e serviços), que possam ser rapidamente disponibilizados e liberados com o mínimo de esforço de gerenciamento ou interação com o provedor de serviços.

Os recursos de computação e infraestrutura como servidores, armazenamento, rede e todas as aplicações disponíveis a partir do fornecedor de nuvem podem ser acessados através da internet, a partir de qualquer local e dispositivo.

Além disso, o uso ou a acessibilidade dos recursos disponibilizados pelo prestador de serviços são cobrados de acordo com o consumo, esse modelo é conhecido como *pay-as-per-use*. Este modelo de nuvem é composto por cinco características essenciais, três modelos de serviços e quatro modelos de implantação.

As cinco características são mostradas na Figura 2.5, o termo essencial significa que, se faltar uma dessas características, então não é a computação em nuvem:

Figura 2.5 – As características essenciais de computação em nuvem



Fonte: CHANDRASEKARAN (2015, p. 14).

A primeira característica é o autosserviço sob demanda (*On-demand self-service*), que permite ao consumidor obter por conta própria recursos de computação, como tempo de servidor e armazenamento em rede de forma automática conforme necessário, sem necessitar de interação humana com cada provedor de serviços.

O amplo acesso por rede (*Broad network access*) é a segunda característica, onde os recursos estão disponíveis através da rede e são acessados através de dispositivos clientes como *smartphones*, *tablets*, *laptops* ou *desktops*.

A terceira característica é o agrupamento de recursos elásticos (*Elastic resource pooling*), onde os recursos de computação do provedor são agrupados para atender a múltiplos consumidores, com diferentes recursos físicos e virtuais dinamicamente atribuídos e reatribuídos conforme a demanda dos consumidores.

Há um senso de independência local em que o cliente geralmente não tem controle ou conhecimento sobre a localização exata dos recursos disponibilizados, mas pode ser capaz de especificar o local em um nível mais alto de abstração (por exemplo, país, estado, ou centro de dados). Exemplos de recursos incluem armazenamento, processamento, memória e largura de banda de rede.

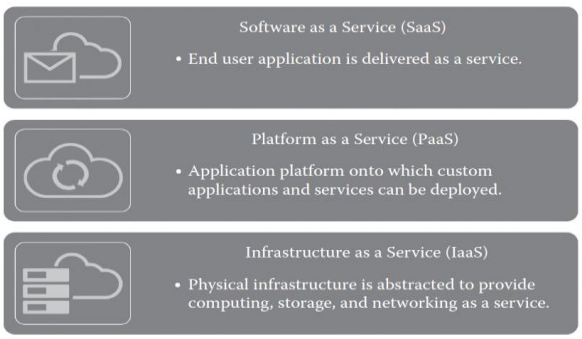
A elasticidade rápida (*Rapid elasticity*) é uma característica que permite a disponibilização e liberação de recursos de computação de forma elástica e automática, os recursos disponíveis podem ser alocados em qualquer quantidade e a qualquer momento.

A última característica é o serviço mensurado (*Measured service*), que permite aos sistemas em nuvem controlar e otimizar de forma automática o uso dos recursos através de medições em tipos de serviços como armazenamento, processamento, comunicação de rede e contas de usuário ativas.

A utilização de recursos pode ser monitorada, controlada e informada, gerando transparência tanto para o fornecedor como para o consumidor do serviço utilizado.

Os três tipos de serviços que disponibilizam recursos de computação baseados em nuvem estão disponíveis para clientes da seguinte forma: *Software-as-a-Service* (SaaS), *Plataform-as-Service* (PaaS) e *Infrastructure-as-a-Serv*ice (IaaS), são também conhecidos como *service-platform-infrastructure* (SPI) que é mostrada na Figura 2.6.

Figura 2.6 – SPI: Serviços oferecidos pelo modelo de computação em nuvem



Fonte: CHANDRASEKARAN (2015, p. 16).

O modelo de serviços SaaS fornece ao consumidor o uso de aplicações fornecidas pelo prestador de serviços em uma infraestrutura de nuvem. As aplicações podem ser acessadas por vários dispositivos clientes, como um navegador, aplicações móveis e entre outros. Alguns exemplos de SaaS são: Google Drive, Google Tradutor, Gmail, Dropbox e Google earth.

Nesse modelo o consumidor não gerencia nem controla a infraestrutura de nuvem subjacente, incluindo rede, servidores, sistemas operacionais, armazenamento, ou mesmo recursos individuais da aplicação, com a possível exceção de configurações limitadas por usuário.

O segundo modelo de serviços denominado PaaS, fornece ao consumidor a possibilidade de instalar seus aplicativos na infraestrutura de nuvem, desenvolvidos com linguagens de programação, bibliotecas, serviços e ferramentas suportadas pelo prestador de serviços.

O consumidor não gerencia nem controla a infraestrutura de nuvem subjacente incluindo rede, servidores, sistema operacional ou armazenamento, mas tem controle sobre as aplicações instaladas e possivelmente configurações do ambiente de hospedagem de aplicações.

O último modelo se serviços conhecido como IaaS, fornece ao consumidor a obtenção de processamento, armazenamento, comunicação de rede e outros recursos de computação fundamentais nos quais o consumidor pode instalar e executar softwares em geral, incluindo sistemas operacionais e aplicativos.

O consumidor não gerencia nem controla a infraestrutura de nuvem subjacente mas tem controle sobre os sistemas operacionais, armazenamento, e aplicativos instalados, e possivelmente um controle limitado de alguns componentes de rede (como *firewalls*).

Os modelos de implantação de nuvem descrevem a maneira como os serviços podem ser implantados ou colocados à disposição de seus clientes, dependendo da estrutura organizacional e da localização. Quatro modelos de implementação são geralmente usados, a saber, privado, comunitário, público e híbrido:

O primeiro modelo de implantação é a nuvem privada, que possui uma infraestrutura que é destinada exclusivamente para uma organização. Pode ser gerida pela organização ou por um terceiro e pode existir dentro ou fora das instalações da organização.

No segundo modelo de implantação, a nuvem comunitária possui uma infraestrutura que é compartilhada por várias organizações e suporta uma comunidade específica que tem preocupações comuns, como por exemplo: missão, requisitos de segurança e política. Pode ser gerida pelas organizações ou por terceiros e podem existir nas instalações ou fora das instalações.

Já no terceiro modelo de implantação, a nuvem pública possui uma infraestrutura que é disponibilizada ao público em geral ou para um grande grupo da indústria e é propriedade de uma organização que vende serviços em nuvem.

E por fim, no último modelo de implantação, a nuvem híbrida possui uma infraestrutura que é uma composição de duas ou mais nuvens (privada, comunitária ou pública), que permanecem entidades únicas, mas estão unidas por tecnologias padronizadas ou proprietárias.

2.2.2 Banco de dados em nuvem

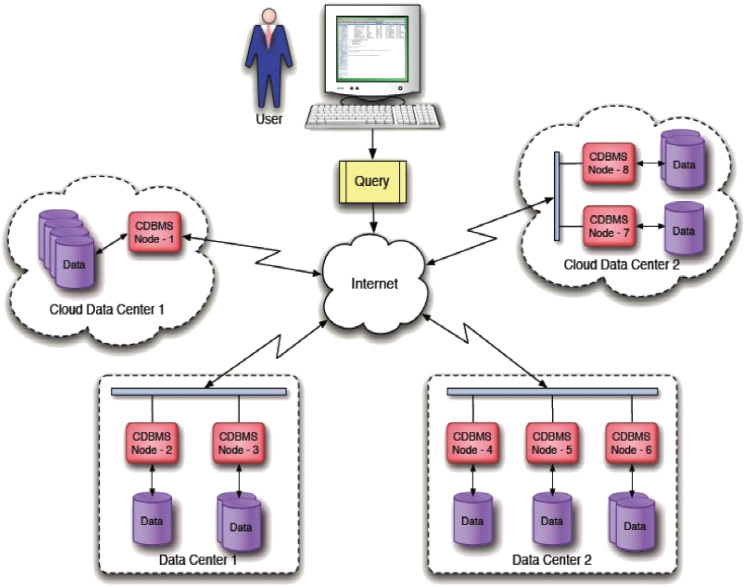
Segundo Shehri (2013) um banco de dados em nuvem pode ser acessado por clientes através da internet a partir do fornecedor de serviços. Um banco de dados em nuvem é implementado utilizando a computação em nuvem, que significa utilizar os recursos de software e hardware do prestador de serviços. Todas as tarefas administrativas e de manutenção são de responsabilidade do prestador de serviços.

Banco de dados em nuvem é usado principalmente como um serviço e, também é chamado de *Database-as-a-service* (DbaaS), neste modelo de serviço, o pagamento pode ser feito de acordo com a capacidade de utilização, bem como as características e utilização das ferramentas de administração do banco de dados.

Um banco de dados em nuvem armazena informações em centros de dados localizados em diferentes regiões, isso faz com que a estrutura do banco de dados em nuvem seja complexa. Há vários nós através de um banco de dados em nuvem, projetados para serviços de consulta em centros de dados que estão localizados em diferentes regiões.

Uma vez que uma consulta é gerada a partir de um usuário através de um terminal, um nó decide primeiro o tipo de consulta e qual nó será melhor para a consulta. Após a consulta ser identificada por um determinado nó, ela é transferida para um nó específico, e em seguida, o nó específico cuida da consulta e responde ao usuário. A Figura 2.7 mostra como um banco de dados em nuvem está disposto.

Figura 2.7 – Estrutura de um banco de dados em nuvem



Fonte: SHEHRI (2013, p. 3).

A seleção de um DBaaS não depende apenas dos serviços prestados pela empresa, mas também das exigências dos usuários que contratam o serviço. Existem certos parâmetros que podem ser tomados como um guia para escolher os melhores DBaas: portabilidade, configurabilidade, acessibilidade, certificação, capacidade de transação, capacidade de armazenamento, integridade dos dados, segurança e armazenamento local.

2.2.3 Protocolo HTTP

O *Hyper Text Transfer Protocol* (HTTP) é um protocolo de comunicação que atua na camada de aplicação da arquitetura TCP/IP, atualmente está na versão 2.0 (HTTP/2) e é definido no [RFC7540] e no [RFC7541]. O RFC é uma publicação do *Internet Engineering Task Force* (IETF) e da *Internet Society* (ISoc), que são as principais entidades que estabelecem padrões de desenvolvimento técnico para a internet.

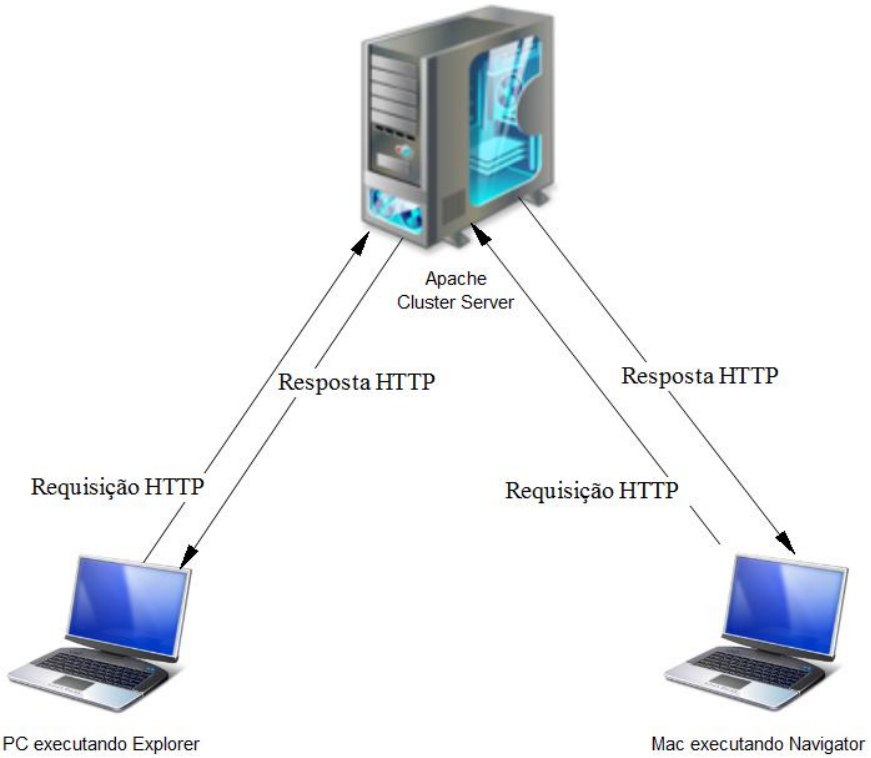
De acordo com Fielding (1999) o HTTP utiliza o modelo cliente-servidor, centrado na ideia de requisição e resposta, que define como um programa(cliente) requisita dados (páginas web) de um programa receptor(servidor) e como esses dados são transferidos para o cliente.

De acordo com Kurose (2010), o HTTP utiliza o *Transmission Control Protocol* (TCP) como seu protocolo de transporte para a transferência de dados. Uma vez que o cliente estabelece uma conexão com um servidor ambos acessam o TCP por meio de suas interfaces *sockets*. No lado do cliente a interface de *socket* é a porta entre o processo (cliente) e a conexão, no lado do servidor, ela é a porta entre o processo (servidor) e a conexão TCP.

Quando um cliente envia uma mensagem para sua interface *socket* a mensagem sai de seus domínios e passa para o domínio do TCP, o que implica que a mensagem será entregue intacta ao servidor visto que o TCP é confiável. De forma semelhante toda mensagem de resposta HTTP emitida pelo servidor chegará intacta ao cliente.

A Figura 2.8 ilustra a comunicação entre clientes e um servidor usando o protocolo HTTP.

Figura 2.8 – Comportamento de requisição-resposta do HTTP



Fonte: KUROSE (2010, p. 73).

Segundo Tanenbaum (2011) o protocolo HTTP foi criado de uma forma mais genérica que o necessário visando às futuras aplicações orientadas a objetos. Por essa razão, são aceitas operações chamadas métodos, diferentes da simples solicitação de uma página da web.

Cada solicitação consiste em uma ou mais linhas de texto American Standard Code for Information Interchange (ASCII), sendo a primeira palavra da primeira linha o nome do método solicitado. Os métodos internos estão listados no Quadro 2.1.

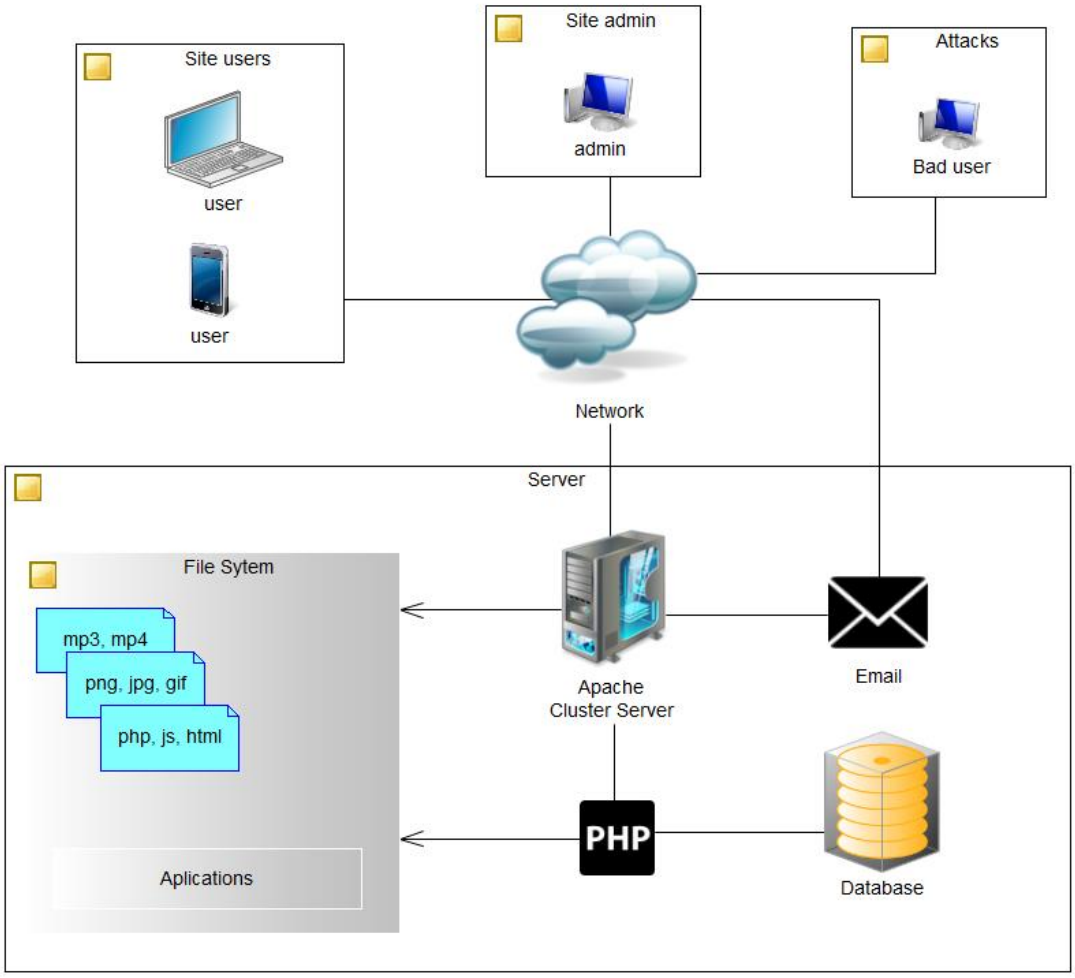
Quadro 2.1 – Os métodos internos de solicitações HTTP

Método	**Descrição**
GET	Solicita a leitura de uma página da Web
HEAD	Solicita a leitura de um cabeçalho de página da Web
PUT	Solicita o armazenamento de um página da Web
POST	Acrescenta a um recurso (por exemplo, uma página da Web)
DELETE	Remove a página da Web
TRACE	Ecoa a solicitação recebida
CONNECT	Conecte-se através de um *proxy*
OPTIONS	Consulta certas opções de uma página

Fonte: Adaptado de TANENBAUM (2011, p. 686).

2.2.4 Servidor web

Segundo Chandramouli (2004) um servidor web é um software que permite que clientes se conectem a ele usando um protocolo bem definido (HTTP). O cliente é geralmente um navegador, como o Internet Explorer ou Mozilla que podem receber opcionalmente dados, que geralmente são páginas web e objetos embutidos tais como: imagens, músicas, vídeos e etc. A estrutura básica de um servidor web está ilustrada na Figura 2.9.



Fonte: BODVOC (2010).

Os servidores costumam ser projetados para atender um grande número de requisições, isso acaba por gerar uma carga para o servidor, devido a esse fator, alguns modelos de processamento baseados em processos ou *threads* são implementados para atender as

requisições de forma a minimizar o impacto da carga no servidor e aumentar a concorrência. Segundo Messias (2007), os servidores web podem ser organizados das seguintes formas:

Servidor iterativo, esse modelo de servidor trata as requisições por ordem de chegada, não possuí um modelo de concorrência baseado em processos ou *threads*, tornando-se muito ineficiente.

Processo por requisição, esse modelo de servidor é concorrente e possuí um processo principal (“pai”) que aguarda por requisições em uma porta pré-definida. Quando uma nova requisição chega, um novo processo (“filho”) é criado e este passa a ter responsabilidade sobre a requisição. Essa abordagem traz consigo como principal desvantagem, o alto custo na criação de processos em Sistemas Operacionais baseados em Unix (Tanenbaum, 2011).

Pool de processos, nesse modelo de servidor um conjunto de processos denominados *workers* e outro denominado *dispatcher* são criados, esses por sua vez constituem um *process pool*. O processo *dispatcher* fica aguardando por requisições de clientes, ao receber uma mensagem, escolhe um dos processos *workers* ociosos da pool e atribui-lhe a tarefa de tratar a mensagem, em seguida o *dispatcher* volta ao seu estado de espera. Os processos *workers* após terminarem suas tarefas não encerram sua execução, continuam esperando novas tarefas, dessa forma podem economizar tempo e diminuir a sobrecarga da criação de novos processos.

Thread por requisição, esse modelo é concorrente e usa *threads* (fluxo de dados dentro de um processo) ao invés de processos, logo a sobrecarga do servidor diminui consideravelmente, já que as *threads* são conhecidas como processos leves e são mais baratas para um sistema operacional do que os processos.

As *threads* podem compartilhar o mesmo espaço de endereçamento dentro de um processo, podem alternar entre si de forma mais eficiente, devido a essas características a concorrência no servidor web pode aumentar de forma a atender um maior número de requisições por unidade de tempo.

Apesar dessa forma de organização ter muitas vantagens sobre processo por requisição, ela apresenta um problema crítico que é a falta de robustez e estabilidade, como as *threads* correm dentro de um mesmo processo e compartilham um mesmo espaço de memória, isso gera um alto grau de dependência entre elas, portanto se uma *thread* estiver com problemas ela pode afetar as demais, deixando o servidor instável.

Pool de *threads*, esse modelo tenta minimizar a sobrecarga causada pela criação das *threads* para cada requisição que chega ao servidor web através da criação de uma *pool* de *threads*, da mesma forma que a *pool* de processos.

TCP/SOCKET, nesse modelo segundo Kurose (2010) para que haja comunicação entre o cliente e o servidor, o cliente tem a tarefa de iniciar o contato com o servidor.

Para que o servidor possa responder ao contato inicial do cliente, ele deve estar pronto, o que implica duas coisas: primeiro, o programa servidor não pode estar inativo, tem de estar executando como um processo antes de o cliente tentar o contato inicial, segundo, o programa tem de ter alguma porta (*socket*) que esteja disponível para o contato inicial do processo do cliente que esteja rodando em uma máquina qualquer.

Estando o processo do servidor em execução, o cliente pode iniciar uma conexão TCP com o servidor, o que é feito no programa cliente pela criação de um *socket*. Ao criar seu *socket* o cliente especifica o endereço de IP e o número de porta do processo servidor. Após a criação do socket no programa cliente, o TCP no programa cliente inicia uma representação de três vias e estabelece uma conexão TCP com o servidor.

Durante a apresentação de três vias, o processo cliente solicita o *socket* de entrada do processo servidor. Quando o servidor atende a solicitação, este cria um novo *socket* exclusivo para o cliente. No fim da fase de apresentação, forma-se uma conexão TCP entre o *socket* do cliente e o novo *socket* do servidor. A Figura 2.10 mostra a comunicação entre processos usando *socket* TCP.

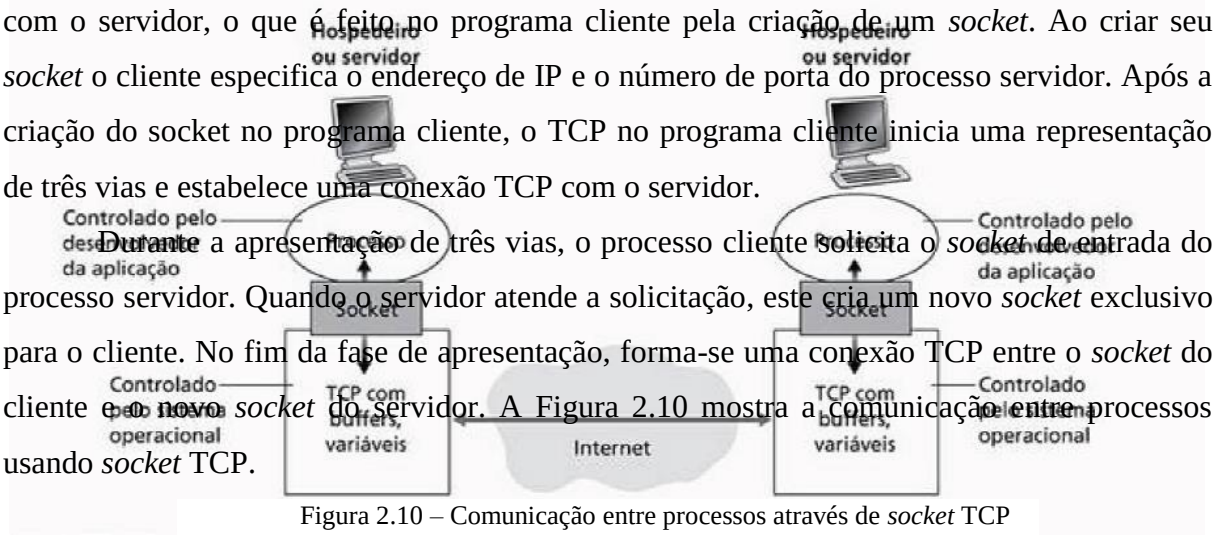


Figura 2.10 – Comunicação entre processos através de *socket* TCP

Fonte: KUROSE (2010, p. 66).

2.2.5 Tecnologias de georreferenciamento e transmissão de dados

As tecnologias de georreferenciamento têm a finalidade de determinar as coordenadas geográficas de dispositivos receptores específicos. Para disponibilizar essas informações entre os componentes do SI é necessário tecnologias de transmissão de dados. As tecnologias de

transmissão de dados são tecnologias que darão suporte a comunicação entre os componentes e elementos do SI.

O conteúdo sobre tecnologias de georreferenciamento descrito nesta subseção foi baseado de acordo com Monico (2008).

Uma importante tecnologia de georreferenciamento é o Sistema Global de Navegação por Satélite ou GNSS (*Global Navigation Satellite System*), que refere-se a uma constelação de satélites, com cobertura global, que fornecem sinais vindo do espaço, esses sinais transmitem dados de localização e cronometria para um receptor. O nome GNSS foi utilizado em 1991, durante a 10ª Conferência de Navegação Aérea.

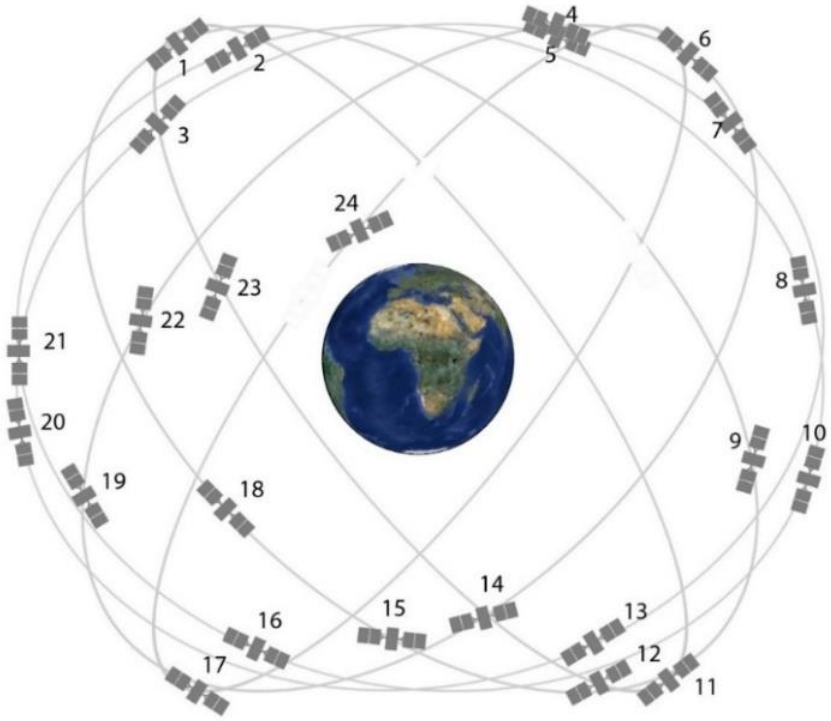
Algumas das tecnologias GNSS já implementadas e em funcionamento são o NAVSTAR-GPS ou apenas GPS (*Global Position System*) e o GLONASSS (*Globalnaya navigatsionnaya sputnikovaya sistema*), ambos são originalmente desenvolvidos para fins militares.

O GPS foi desenvolvido pelo Departamento de Defesa dos Estados Unidos, visando ser o principal sistema de navegação das Forças Armadas norte-americanas e o GLONASS foi desenvolvido na antiga URSS (União das Repúblicas Socialistas Soviéticas), pela *Soviet Union's Scrientific Production Association of Applied Mechanics* e atualmente é mantido e operado *pela Russian Federation Space Forces*.

Em razão da alta acurácia, mesmo em condições climáticas adversas, e do grande desenvolvimento da tecnologia envolvida nos receptores, o GPS é o sistema de navegação mais utilizado, e tem uma grande comunidade de usuários nos mais variados segmentos civis (navegação, posicionamento geodésico, agricultura, controle de frotas e etc.).

O GPS entrou em operação em 27 de abril e 1985, com 24 satélites em órbita (Figura 2.11), e até julho de 2007, havia trinta satélites operacionais.

Figura 2.11 – Representação da distribuição de 24 satélites
na órbita terrestre



Fonte: Adaptado de SPACE (2015).

Com o GPS, um usuário em qualquer local da superfície terrestre tem à sua disposição no mínimo 4 satélites, com este número de satélites é possível determinar em tempo real a localização de um receptor.

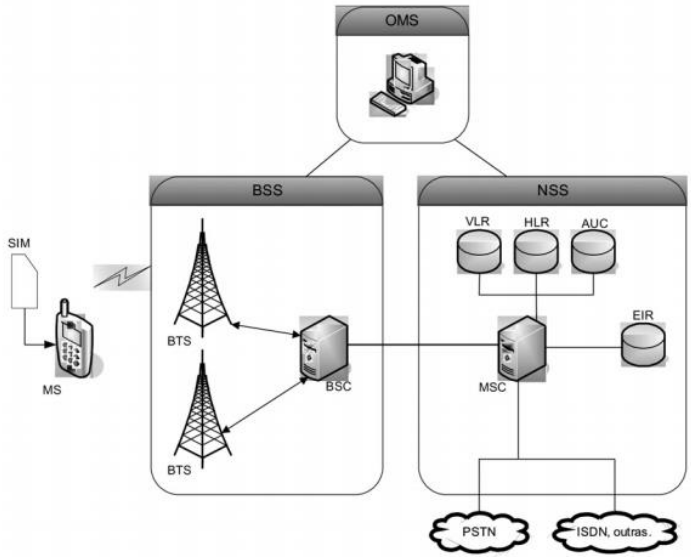
O funcionamento da navegação por GPS, consiste na utilização das distâncias entre os quatro satélites e o receptor GPS. Com as coordenadas dos satélites e as medidas das distâncias entre a antena do receptor e os quatro satélites, em um mesmo instante de tempo, é possível determinar a posição geográfica da antena do receptor.

Para disponibilizar as coordenadas geográficas de um dispositivo receptor GPS, para elementos remotos, é necessário tecnologias de transmissão de dados. A existência de tecnologias de transmissão de dados eficientes dão suporte a funcionalidades importantes como a AVL em um SI.

Atualmente no Brasil, a principal tecnologia de transmissão de dados para dispositivos móveis é o *Global System for Mobile Communication* (GSM), que é um padrão de telefonia celular digital de Segunda Geração (2G), desenvolvido na Europa e padronizado pela *European Telecommunication Standards Institute* (ETSI), sendo adotado na maior parte do mundo (WAGNER, 2009).

De acordo com Tateoki (2007), a arquitetura da rede GSM é composto por várias entidades com funções e interfaces específicas, a rede GSM pode ser dividida em três subsistemas: *Mobile Station* (MS), *Base Station Subsystem* (BSS) e o *Network Subsystem* (NSS), como apresentado na figura 2.12.

Figura 2.12 – Arquitetura da rede GSM



Fonte: PIROTTI; ZUCCOLOTTO (2009, p. 83).

O MS é um dispositivo que faz uso da rede GSM, o dispositivo pode ser um telefone celular ou qualquer outro equipamento. O MS é controlado por um cartão "inteligente" designado de SIM (*Subscriber Identity Module*). A Figura 2.13 exemplifica um cartão SIM.

Figura 2.13 – Cartão SIM e um dispositivo móvel.

Fonte: APPLE (2015).

Com o cartão SIM inserido em um dispositivo o assinante, a partir do dispositivo, pode ter acesso aos serviços da rede GSM. O cartão SIM tem uma identificação única mundial denominada de *International Mobile Subscriber Identity* (IMSI) (TATEOKI, 2007).

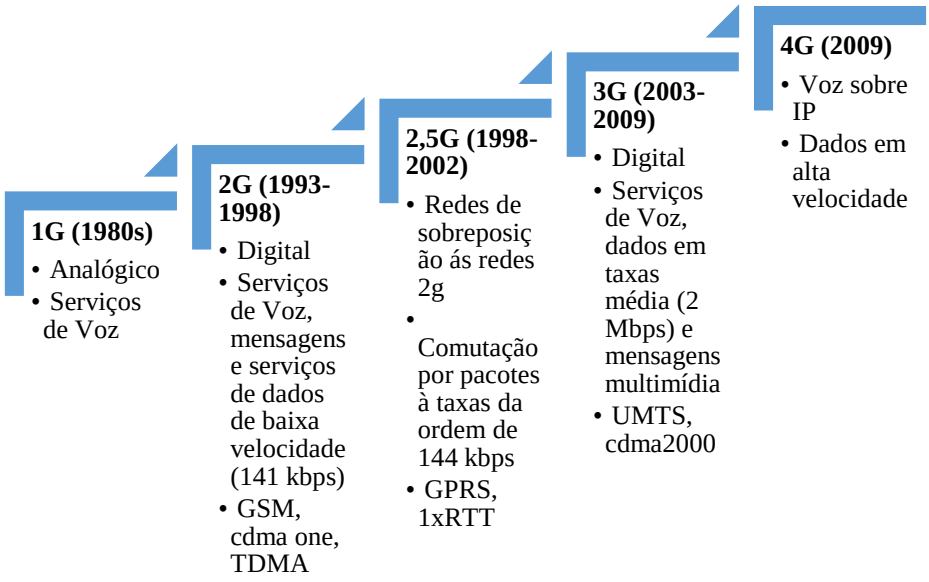
O BSS é responsável pelo controle de ligação rádio da NSS com o MS. O NSS é responsável pela comutação de chamadas e solicitações do MS, adicionalmente providencia toda a funcionalidade necessária para o credenciamento de um assinante móvel (TATEOKI, 2007).

Além do suporte de tráfego de voz, a rede GSM conta com alguns serviços como o *Multimedia Messaging Service* (MMS) e o *Short Message Service* (SMS). O MMS é um serviço de compartilhamento de mensagens multimídias (textos, imagens e áudios) e o SMS é um serviço de compartilhamento de mensagens de texto simples de até 160 caracteres.

Inicialmente as primeiras gerações de redes móveis foram desenvolvidas para o tráfego de voz, porém com o passar do tempo surgiu uma série de esforços para ampliar o tráfego de dados nas redes móveis (WAGNER, 2009).

De acordo com Tateoki (2007), as redes móveis podem transmitir dados em banda larga, graças a evolução da rede móvel a cada geração. A Figura 2.14 exemplifica a evolução da rede móvel a cada geração.

Figura 2.14 – Histórico da evolução da rede móvel a cada geração.



Fonte: Adaptado de MERCADO (2015).

Entre os serviços de transferência de dados de segunda geração pode-se destacar o *General Packet Radio Service* (GPRS), que é um serviço para comunicação de dados que permite conexão com a internet no MS sem a necessidade de estabelecer uma chamada telefônica (WAGNER, 2009).

Ao solicitar uma conexão de dados na rede GPRS é necessário informar a *Access Point Name* (APN). Os endereços da APN são endereços que podem seguir a padronização adotada na internet, como por exemplo operadora.com.br, esses endereços apontam para o servidor que fornece informações que definem quais aplicações, protocolos e serviços o usuário poderá utilizar (SVERZUT, 2005).

A evolução dos serviços de transferência de dados, da rede GSM de segunda geração (GPRS), para os serviços de terceira geração (3G) com altas taxas de dados foi padronizada pela *3rd Generation Partnership Project* (3GPP). Esta evolução modificou a arquitetura da rede GSM, porém manteve a compatibilidade com arquiteturas de gerações anteriores já existentes (WAGNER, 2009).

2.2.6 Tecnologias de desenvolvimento

Esta subseção tem como objetivo descrever os conceitos das tecnologias utilizadas para desenvolver o SI móvel. As tecnologias de desenvolvimento podem ser divididas em auxiliares e implementação.

As tecnologias auxiliares, são aquelas cuja finalidade é dar apoio a implementação, oferecendo um ambiente de desenvolvimento integrado - *Integrated Devolpment Enviornment* (IDE), algumas tecnologias como o Android Studio e o PyCharm (IDE´S) representam as tecnologias auxiliares.

O Android Studio é um IDE oficial para o desenvolvimento de aplicativos para Android. Suas funcionalidades incluem a edição inteligente de códigos, recursos para *design* de interface de usuário e análise de performance (DEVELOPERS, 2015).

O Android Studio tem suporte e atualizações constantes. Com uma IDE oficial, o desenvolvimento de aplicativos para Android é facilitado por conta das documentações existentes.

Uma outra tecnologia de desenvolvimento auxiliar é o PyCharm, que é uma IDE desenvolvida pela empresa JetBrains. A IDE é utilizada para a programação em Python.

De acordo com a JetBrains (2015), o PyCharm oferece depuração gráfica, testador de unidade integrado, integração com sistemas de controle, tem suporte à algumas *frameworkers*

como Flask e Django, *deploy* automático de aplicações Google App Engine e entre outros. O PyCharm é multiplataforma, roda em Windows, Mac OS X e Linux, ele está sob uma licença proprietária, porém existe uma versão *trial*.

As tecnologias de implementação são aquelas usadas diretamente no código fonte das aplicações, ou seja, são independentes de IDE's. Algumas tecnologias como Google Maps, Google App Engine, API NDB e Volley representam as tecnologias de implementação.

Uma importante tecnologia utilizada no desenvolvimento é o Google Maps, que é um serviço desenvolvido pela Google, o serviço oferece suporte a pesquisa de localizações e visualização de mapas. A plataforma de mapeamento da Google também inclui imagens de satélite, *Street View* (vista de rua), relevo, rotas de carro, mapas estilizados, informações demográficas, análises e banco de dados de lugares.

A Google oferece suporte para os desenvolvedores utilizarem o Google Maps através de uma *Application Programming Interface* (API), que é um conjunto de rotinas e padrões de programação.

A API do Google Maps é um conjunto de API's que permite sobrepor dados em um mapa personalizado da Google. Com a API de Mapas do Google o desenvolvedor pode incluir mapas e informações de mapeamento personalizado em aplicativos ou em web Sites (MAPS, 2015).

Segundo a Goolge (2015), A Google App Engine (GAE) é uma plataforma com uma oferta de serviço PaaS, que permite criar e executar aplicativos na infraestrutura da Google. Aplicativos da GAE são fáceis de construir, fácil de manter e fácil de escalar quando o tráfego e o armazenamento de dados mudam. A plataforma suporta desenvolvimento para a seguintes linguagens: Python, PHP, Java e Go. Onde cada linguagem é responsável pelo seu ambiente de execução.

Para a linguagem Python a GAE disponibiliza múltiplas opções de armazenamento de dados para as aplicações:

- *App Engine Datastore*, é um tipo de armazenamento de dados sem esquema, com cache automático e um mecanismo de consulta sofisticado de transações atômicas.
- Google *Cloud SQL*, é um banco de dados relacional *Strutured Query Language* (SQL) para aplicativos *App Engine*, com base no banco de dados MySQL.
- Google *Cloud Storage*, é um serviço de armazenamento de objetos e arquivos, acessível a aplicativos através da biblioteca cliente do Google *Cloud Storage*.

Além disso a GAE disponibiliza um conjunto de API's que facilita a implementação e implantação das aplicações, um exemplo é API NDB para Python, que é usada para armazenamento de dados sem esquemas e é adequada para manipular dados estruturados.

De acordo com a Google (2015), a Volley é uma biblioteca HTTP que torna o trabalho de rede mais fácil e mais rápido para aplicações Android.

A Volley oferece as seguintes funcionalidades: agendamento automático de solicitações de rede, múltiplas conexões de rede simultâneas, disco e memória cache de resposta transparente com coerência de cache HTTP padrão, apoio ao pedido de priorização, solicitação de cancelamento, ordenação forte que torna mais fácil para preencher corretamente a interface do usuário com os dados buscados de forma assíncrona a partir da rede, depuração e ferramentas de rastreamento.

2.2.7 Tecnologias de infraestrutura

Esta subseção tem como objetivo descrever alguns dos dispositivos que utilizam as tecnologias de georreferenciamento e transmissão de dados. Os dispositivos descritos são: rastreador GPS TK103-2, *smartphone* com receptor GPS e *modem* GSM. Esses dispositivos podem ser utilizados em um SI com funcionalidade AVL.

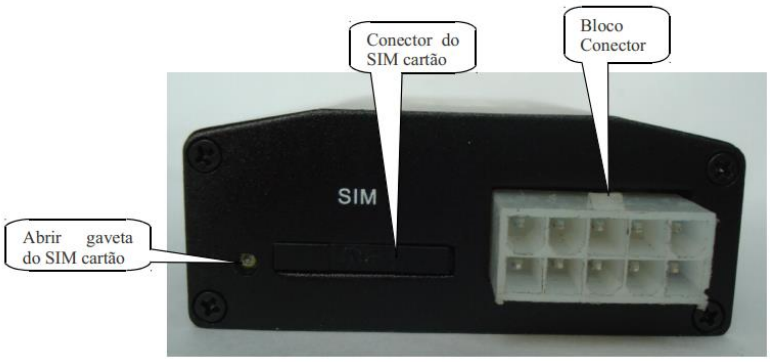
Todo conteúdo sobre o rastreador GPS TK103-2 foi baseado de acordo com o manual do dispositivo (MANUAL GPS TK103-2). Um dispositivo que trabalha com a tecnologia de georreferenciamento GPS é o rastreador GPS TK103-2 (Figura 2.15), utilizado para rastreamento e monitoramento veicular.

Figura 2.15 – GPS TK103-2

Fonte: Elaborado pelos autores.

A transferência de informação de rastreamento ocorre através de serviços da rede móvel GSM, por isso o dispositivo conta com o suporte para instalação de um cartão SIM (Figura 2.16).

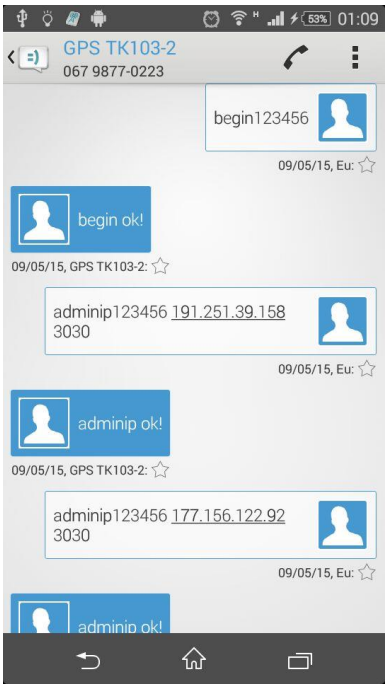
Figura 2.16 – Entrada do Cartão SIM no GPS TK103-2



Fonte: Manual GPS TK103-2.

Para configurar o dispositivo é necessário enviar mensagens SMS com comandos, para o número IMSI do cartão SIM instalado no dispositivo, como mostrado na Figura 2.17.

Figura 2.17 – Comando enviado por SMS para o dispositivo



Fonte: Elaborado pelos autores.

O rastreador pode enviar dados de localização de duas maneiras, por SMS ou por transferências de dados utilizando os serviços de segunda geração (GPRS) da rede móvel. Para iniciar o rastreamento é necessário definir o modo de operação para envio de dados de localização.

Ao configurar o rastreador para enviar informações via SMS é necessário especificar o IMSI (número associado ao cartão SIM) do destinatário. Se o modo de operação de envio de informações for via GPRS é necessário configurar a APN da rede móvel e o destinatário, especificado por um IP e uma Porta (servidor *socket* TCP).

Ao ligar o dispositivo é necessário aguardar de 10 a 40 segundos, esse é o tempo de inicialização necessário para que o rastreador adquiria sinais da rede GSM e do GPS.

O Quadro 2.2 a seguir apresenta os comandos e suas descrições, esses comandos são necessários para configurar e utilizar o rastreador (MANUAL GPS TK103-2).

Quadro 2.2 – Comandos do rastreador GPS TK103-2

COMANDO	**DESCRIÇÃO**	**EXEMPLO**	**RESPOSTA**
begin[SENHA]	Comando para inicializar o dispositivo, deve ser acrescido de uma senha. Por Padrão o dispositivo tem senha 123456.	begin123456	begin ok!
admin[SENHA] + [NUMERO_AUTORIZADO]	Comando que define o número (IMSI) autorizado para configurar e receber dados de georreferenciamento do dispositivo.	admin123456 006799887766	admin ok!
apn[SENHA] + [APN]	Comando que define a APN da rede móvel no dispositivo para uso do GPRS.	apn123456 operadora.com.br	APN ok!
apnuser[SENHA] + [USER]	Comando que define o usuário da APN.	apnuser123456 operadora	APNUSER ok!
apnpasswd[SENHA] + [APN_KEY]	Comando que define a senha da APN.	apnpasswd123456 operadora	APNPASSWD ok!
adminip[SENHA] + [IP] + [PORTA]	Comando que define o IP e a porta do servidor Socket TCP, que receberá os dados de georreferenciamento.	adminip123456 192.168.2.1 3000	ADMINIP ok!
t[TEMPO][TIPO_TEMPO] ***n[SENHA]	Comando que habilita o dispositivo para rastreamento automático, o parâmetro [TEMPO] define o tempo em que o dispositivo deve enviar dados de georreferenciamento automaticamente. O parâmetro [TIPO_TEMPO] define o tipo do tempo que pode ser: s para segundos, m para minutos e h para horas.	t030s***n123456	t030s***n ok!
tracker[SENHA]	Comando que inicia o rastreamento no dispositivo, o dispositivo passa a enviar dados de georreferenciamento para os receptores configurados.	tracker123456	tracker ok!
reboot[SENHA]	Comando que restaura as configurações iniciais do dispositivo.	reboot123456	reboot ok!

Fonte: MANUAL GPS TK103-2.

Outros dispositivos que podem exercer a funcionalidade de rastreamento do GPS TK103-2 são os *smartphones*, isso pois alguns desses dispositivos possuem receptores GPS integrados.

Smartphone designa a categoria de telefones móveis que oferecem recursos avançados que vão além dos recursos oferecidos por um telefone celular típico.

Os *smartphones* rodam um sistema operacional completo, com interface padronizada e suporte para plataformas de desenvolvimento de aplicativos, podem conectar à web através de redes *wireless* e redes móveis de últimas gerações (MORIMOTO, 2009).

Morimoto (2009) ressalta que a funcionalidade de rastreamento em um *smartphone*, que possui receptor GPS, pode ser obtida através de aplicativos específicos executados no sistema operacional do dispositivo.

Um dispositivo que pode ser utilizado em um SI com funcionalidade AVL é o *modem* GSM (Figura 2.18), que é um dispositivo que fornece acesso aos serviços da rede móvel GSM. O *modem* GSM pode ser um dispositivo externo que geralmente é conectado a um computador através de um cabo serial ou via USB (*Universal Serial Bus*) (GSM GPRS, 2015).

Figura 2.18 – *Modem* GSM

Fonte: D-LINK (2015).

Para funcionar e obter os serviços da rede GSM é necessário a instalação do cartão SIM no *modem*. Os principais serviços suportados por um *modem* GSM, são o envio e recebimento de mensagens SMS e MMS, chamadas e acesso à internet da rede móvel (GSM GPRS, 2015).

Os *modems* GSM podem ser utilizados em computadores que atuam como servidores de mensagens SMS, para essa aplicação existe modems profissionais (Figura 2.19) que suportam altas taxas de envio e recebimento de mensagens SMS.

Figura 2.19 – *Modem* GSM
Profissional

Fonte: URMET (2015).

A interação do *modem* GSM com o computador a nível de programação é feita através de comandos AT (abreviação para *Attention*), que são instruções usadas para controlar o *modem* ou outro dispositivo móvel.

Os comandos começam com AT para indicar a atenção do dispositivo para uma instrução de controle. Os comandos são enviados em forma de texto para a porta USB ou serial do dispositivo móvel (INTRODUCTION, 2015). O Quadro 2.3 contém alguns exemplos dos comandos AT.

Quadro 2.3 – Comandos AT

COMANDOS	**DESCRIÇÃO**
AT+CIMI	Obtém informações sobre o número IMSI do dispositivo móvel.
AT+CMGF=1	Define o modo de operação do dispositivo, para enviar mensagens SMS.
AT+CMGS=NUM	Define o número IMSI do destinatário de mensagens SMS. NUM é o número IMSI.
AT+CSAS	Salva as configurações do dispositivo móvel.
AT+CRES	Restaura as configurações do dispositivo. Configurações como o número IMSI do destinatário de mensagens SMS são limpas.

Fonte: INTRODUCTION (2015).

De acordo com a ETSI (2007), originalmente os comandos AT foram desenvolvidos para *modems* de computadores em 1977 pela *Hayes Microcomputer Products*, desde então os comandos apresentaram uma evolução a ponto de se tornar uma plataforma mediadora compreensível, para dispositivos móveis.

Os comandos AT fornecem o controle sobre: chamadas, cartão SIM, informação do dispositivo, configuração do dispositivo, domínio de pacotes e serviços de redes (ETSI, 2007).

Atualmente, um conjunto de comandos AT tem sido padronizado, destes comandos apenas alguns são obrigatórios, além de que muitos dispositivos não têm o conjunto

totalmente implementado. Além disso os fabricantes dos dispositivos móveis podem abrir funções adicionais em seus produtos, implicando assim na criação de novos comandos AT (ETSI, 2007).

2.3 Trabalhos correlatos

Esta subseção tem como finalidade expor e analisar alguns trabalhos correlatos de diferentes regiões e suas principais caraterísticas e funcionalidades.

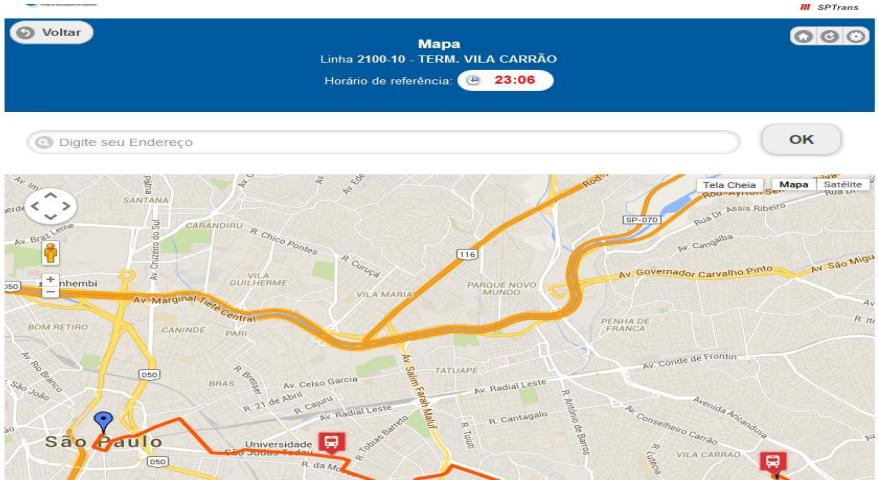
2.3.1 Olho Vivo

Segundo a São Paulo Transporte (SPTrans) (2015), o Olho Vivo é um sistema de monitoramento de transporte (ônibus), que foi lançado em maio de 2008, ele permite que usuários possam verificar na cidade de São Paulo o tempo de viagem, a velocidade média nos principais corredores e vias, localizar os ônibus em tempo real e o horário de quando passarão num determinado ponto. O sistema conta com três opções de consultas:

1) O de olho na linha é a primeira opção, ela permite o usuário digitar o número ou nome da linha e ver ao longo do trajeto quantos ônibus estão circulando no momento e em quais locais estão.
2) A segunda opção é o de olho no ponto, nessa opção o usuário fica sabendo quais ônibus/linhas estão se aproximando de seu ponto.
3) A última alternativa é o de olho na via, nessa opção o usuário pode visualizar como está o desempenho dos principais corredores viários da cidade, com o auxílio de um mapa interativo.

A aplicação utiliza o Google Maps para a visualização do trajeto de uma determinada linha. A Figura 2.20 mostra a utilização do sistema e o itinerário para a linha 2100-10 Terminal Vila Carrão - Terminal Vila Carrão – Praça da Sé.

Figura 2.20 – Mapa do itinerário da linha 2100-10, Terminal Vila



Fonte: SPTrans (2015).

O Olho Vivo é uma aplicação gratuita e está disponível em plataforma web e dispositivos móveis (*smartphones*) com sistema operacional Android e IOS.

2.3.2 Vá de Ônibus

De acordo com a Federação das Empresas de Transporte Coletivo do Estado do Rio de Janeiro (Fetranspor), o Vá de Ônibus é um SI baseado em tecnologia de georreferenciamento, que identifica as melhores opções de deslocamento no Estado do Rio de Janeiro, utilizando o ônibus como meio de transporte.

Segundo a Fetranspor (2015) o sistema permite ao usuário escolher o melhor itinerário para chegar ao destino desejado. Todas as alternativas de viagens são apresentadas, incluindo números e nomes das linhas, eventuais transbordos, distâncias percorridas pelas linhas, trechos a serem percorridos a pé e o valor total a ser pago.

Ao fazer sua escolha, seja ela por custo, distância, caminhada ou transbordos, o usuário terá acesso a todos os detalhes do percurso, ou seja, um passo a passo entre o local de origem informado e o local onde se pretende chegar.

A aplicação utiliza o Google Maps para a visualização do trajeto de uma determinada linha. A Figura 2.21 mostra a utilização do sistema e o itinerário traçado para a busca entre às Avenidas Pompeu Loureiro e Nossa Senhora de Copacabana. São mostrados ainda a distância, o custo e o número da linha de ônibus.

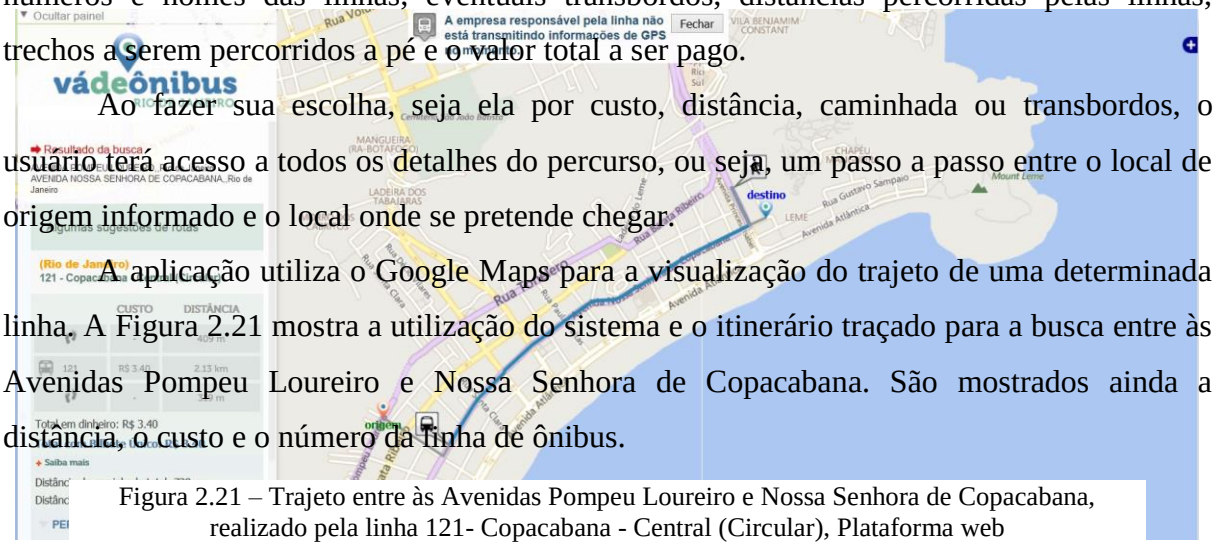


Figura 2.21 – Trajeto entre às Avenidas Pompeu Loureiro e Nossa Senhora de Copacabana, realizado pela linha 121- Copacabana - Central (Circular), Plataforma web

Fonte: Fetranspor (2015).

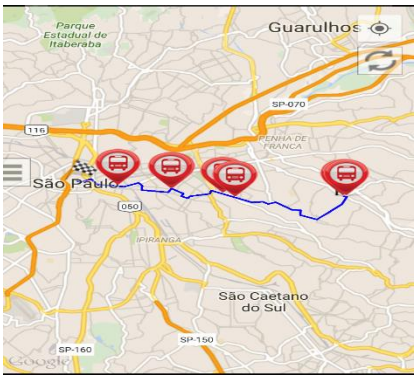
O Vá de Ônibus é uma aplicação gratuita e está disponível em plataforma web e dispositivos móveis (*smartphones*) com sistema operacional Android e IOS.

2.3.3 Cadê o ônibus? (SP)

O Cadê o Ônibus? é um sistema de monitoramento de transporte (ônibus), que foi desenvolvido exclusivamente para celulares. Ele permite que usuários possam verificar a posição geográfica dos ônibus em tempo real, os pontos próximos, o itinerário das linhas, o horário de partida, o trânsito e a administração de linhas favoritas.

A aplicação faz uso do Google Maps para a visualização do trajeto de uma determinada linha. A Figura 2.22 mostra a utilização do sistema e o itinerário para a linha 2100-10 Terminal Vila Carrão – Praça da Sé.

Figura 2.22 – Mapa do itinerário da linha 2100-10, Terminal Vila Carrão: Praça da Sé, Plataforma móvel.



Fonte: Aplicativo Cadê o Ônibus? (SP).

O Cadê o Ônibus? (SP) é uma aplicação gratuita e está disponível nas três principais plataforma móveis, Android, IOS e Windows Phone.

O Quadro 2.4 permite visualizar as funcionalidades disponibilizadas por cada uma das aplicações.

Quadro 2.4: Quadro comparativo das funcionalidades entre as aplicações

Funcionalidades	Olho Vivo	Vá de Ônibus	Cadê o Ônibus ?
Monitoramento em tempo real	X		X
Serviços de ponto de Ônibus	X	X	X
Consulta de itinerário	X	X	X
Consulta de trânsito em tempo real			X
Consulta de linhas	X	X	X
Serviço de QR code		X	

Fonte: Elaborado pelos autores.

O Olho Vivo é uma aplicação que funciona muito bem em sua plataforma web, oferecendo os serviços básicos para os usuários dos meios de transporte de São Paulo. Já a sua versão para a plataforma móvel Android (5.0.2) não funciona.

O Vá de Ônibus é uma aplicação que funciona nas duas plataformas, web e móvel, oferecendo alguns serviços extras como o serviço de QR code, porém não possui um sistema de monitoramento em tempo real, vital para esse tipo de aplicação.

O Cadê o Ônibus? é um sistema que só funciona em plataforma móvel, porém disponibiliza todas as funcionalidades desejáveis para esse tipo de aplicação. Seu diferencial está na interface, ágil e fácil de manusear, oferece monitoramento em tempo real em diversos formatos de mapa e ainda possibilita visualizar a situação do trânsito em um determinado trecho.

Ao analisar as funcionalidades de cada trabalho correlato listadas no Quadro 2.4 é possível compará-las e concluir que, o sistema Cadê o Ônibus é o que apresenta o maior e mais significativo número de funcionalidades, dessa forma o sistema se mostra mais adequado para localizar veículos de transporte em tempo real.

O capítulo seguinte descreverá a forma como o protótipo do SI foi desenvolvido, suas etapas e as tecnologias utilizadas.

3 MATERIAIS E METODOS

O desenvolvimento do protótipo do SI foi dividido em cinco etapas, a primeira etapa consistiu no estudo de conceitos relacionados a um SI, a segunda etapa consistiu no estudo e na avaliação de tecnologias e equipamentos, a terceira etapa consistiu no desenvolvimento do projeto do protótipo do SI, a quarta e quinta etapa consistiu respectivamente na implementação e teste do protótipo do SI.

O estudo de conceitos relacionados a um SI na primeira etapa ocorreu através da pesquisa e revisão de materiais bibliográficos. Nessa etapa foram estudados as diferentes classificações de um SI e as tecnologias associadas um SIG Móvel com funcionalidade AVL.

Na segunda etapa, ocorreu o estudo e a avaliação de tecnologias e equipamentos que poderiam ser utilizados no desenvolvimento do protótipo do SI. A avaliação teve o objetivo de definir as melhores tecnologias a serem utilizadas no protótipo.

Com a escolha das melhores tecnologias para o protótipo do SI na segunda etapa, foi possível desenvolver o projeto do protótipo do SI na terceira etapa. No projeto foram definidas as tecnologias utilizadas e quais as funcionalidades do protótipo. Nessa etapa também foram definidos: os requisitos de sistema, casos de uso e diagramas de classe de projeto.

Na quarta e quinta etapa ocorreram respectivamente a implementação e teste do protótipo do SI. Os testes na quinta etapa foram realizados em veículos de uma linha dos meios de transporte público na cidade de Dourados.

4 DESENVOLVIMENTO

4.1 Avaliação de tecnologias

Esta subseção tem por objetivo detalhar a etapa de estudos e avaliação das tecnologias que antecederam o desenvolvimento do protótipo do SI.

Para avaliar as tecnologias a serem utilizadas no desenvolvimento, foram elaboradas três arquiteturas que pudessem representar o protótipo do SI, e são elas: arquitetura com transmissão de dados via SMS, arquitetura com transmissão de dados via *socket* TCP e arquitetura com transmissão de dados via HTTP.

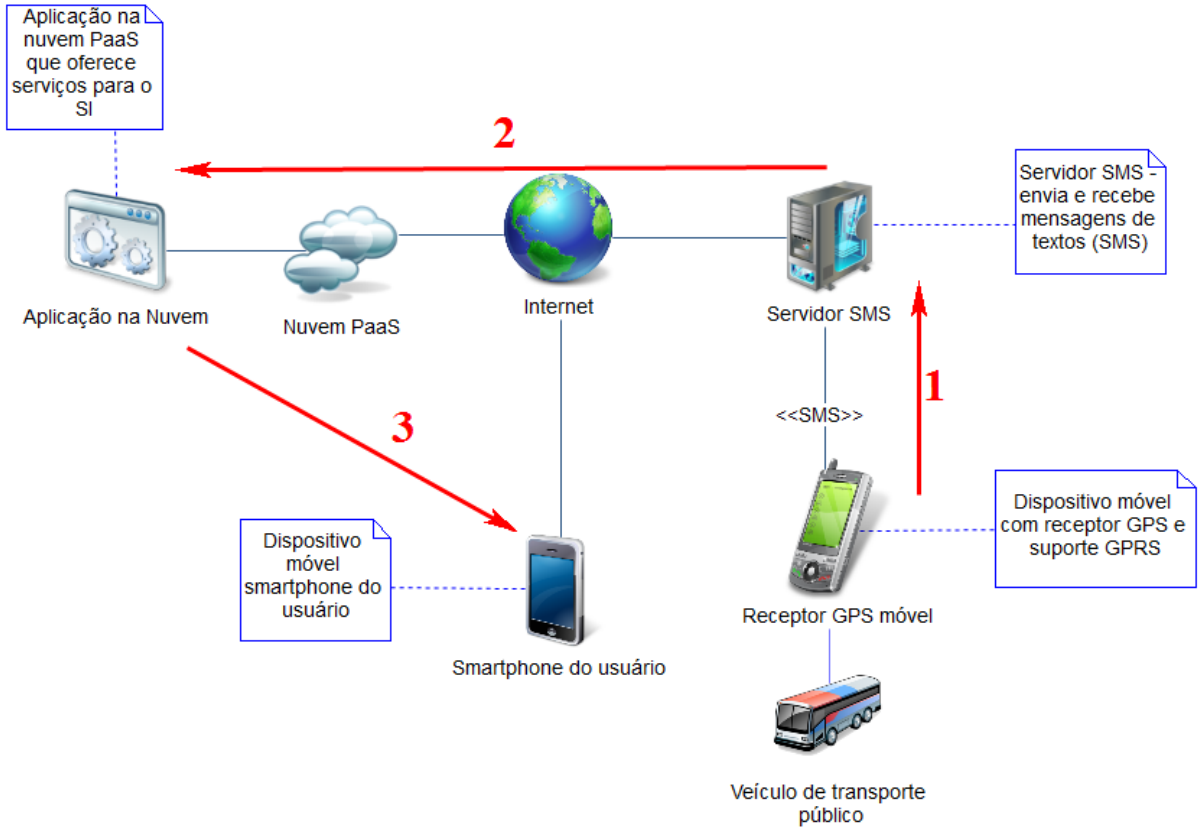
As arquiteturas foram elaboradas para testar as tecnologias estudadas, possibilitando assim a escolha das melhores tecnologias para o protótipo do SI. É importante ressaltar que na etapa de avaliação de tecnologias não ocorreu o desenvolvimento das arquiteturas, mas sim somente estudos e testes específicos nas tecnologias associadas a cada arquitetura.

4.1.1 Arquitetura do SI com transmissão de dados via SMS

A arquitetura do SI com transmissão de dados via SMS foi elaborada para aproveitar o baixo custo apresentado no envio de mensagens SMS.

Nessa arquitetura, as coordenadas geográficas dos veículos devem ser enviadas utilizando mensagens SMS, a Figura 4.1 detalha a arquitetura, as linhas vermelhas numeradas representam o fluxo de envio de coordenadas geográficas entre os componentes do SI.

Figura 4.1 – Arquitetura do SI com transmissão de dados via SMS



Fonte: Elaborado pelos autores.

Nessa arquitetura um SI deve ser composto pelos seguintes elementos: receptor GPS móvel, servidor SMS, *smartphone* do usuário e uma aplicação na nuvem.

O funcionamento do SI com essa arquitetura seria da seguinte forma, um receptor GPS móvel enviaria as coordenadas geográficas do veículo de transporte público, por mensagens SMS para um servidor SMS (linha vermelha número 1 da Figura 4.1).

O servidor SMS, receberia as mensagens SMS através de um *modem* GSM, processaria as informações das mensagens obtendo as coordenadas geográficas e enviaria as coordenadas geográficas para uma aplicação na nuvem (linha vermelha número 2 da Figura 4.1).

A aplicação na nuvem armazenaria as coordenadas geográficas, enviadas pelo servidor SMS. Quando a aplicação cliente no *smartphone* do usuário solicitar por informações, a aplicação na nuvem enviara os dados solicitados para a aplicação solicitante (linha vermelha número 3 da Figura 4.1). Assim o usuário poderá consultar as informações sobre a localização do veículo de transporte público.

Para verificar a viabilidade dessa arquitetura, ocorreram testes através de simulações de envio de mensagem SMS. Nos testes foram utilizados dois *modems* GSM em

computadores, que atuaram como servidores de mensagens SMS. Um dos *modems* atuou como receptor e outro como emissor.

A verificação de mensagens SMS recebidas no *modem* GSM receptor ocorreu através do programa padrão do *modem* que tem uma interface para visualização de mensagens enviadas e recebidas.

Para utilizar o *modem* GSM emissor no computador, que atuou como servidor SMS, foi necessário a implementação de um programa controlador utilizando comandos AT. As Figura 4.2 e 4.3 apresentam os trechos de códigos em linguagem de programação Python da implementação do programa controlador.

Figura 4.2 – Método principal do programa controlador

```
def main():

    # Instanciar o objeto da classe TextMessage
    sms = TextMessage()
    # Conectar ao dispositivo GSM da porta COM (USB ou serial)
    sms.connectPhone()
    # contador
    i = 0

    while(1):
        # Definir Mensagem a ser enviada com o contador
        sms.setContent("Mensagem teste: "+str(i))
        # Enviar Mensagem
        sms.sendMessage()
        print "Mensagem teste: " + str(i) + " Enviada."
        i = i + 1
    # Desconectar Dispositivo GSM
    sms.disconnectPhone()
```

Fonte: Elaborado pelos autores.

Na Figura 4.2 o método *main* envia mensagens SMS continuamente para o receptor, cada mensagem enviada contém um texto e o número de mensagens enviadas anteriormente, desta forma é possível verificar no *modem* GSM receptor a ordem e a chegada de mensagens.

Figura 4.3 – Classe para envio de mensagens pelo programa controlado

```
"""
send.py - Enviar Mensagens SMS.
"""
import serial # Biblioteca para comunicação com o dispositivo
import time
 # Classe para Envio de Mensagens
class TextMessage:

    # construtor da classe
    def __init__(self, recipient="+556799178209", message="MSG Padrao"):
        self.recipient = recipient
        self.content = message
    # Conecta ao dispositivo GSM da porta COM (USB ou serial)
    def connectPhone(self):
        self.ser = serial.Serial('COM17', 460800, timeout=5)
        time.sleep(0.1)

    # Envia Mensagens
    def sendMessage(self):

        self.ser.write('ATZ\r') # restaura o modem para as configurações padrões
        time.sleep(0.5) # Tempo minimo necessário para o modem processar o comando AT
        self.ser.write('AT+CMGF=1\r') # Muda o modo de operação do modem para SMS
        time.sleep(0.5)
        # Define numero IMSI do destinatário
        self.ser.write('''AT+CMGS="''' + self.recipient + '''"\r''')
        time.sleep(0.5)
        self.ser.write(self.content + "\r") # Define Mensagem a ser enviada
        time.sleep(0.5)
        # Envia Ctrl+Z para o modem, finalisando a escrita
        # e enviando a mensagem.
        self.ser.write(chr(26))
        time.sleep(0.5)

    # Desconecta dispositivo GSM
    def disconnectPhone(self):
        self.ser.close()
```

Fonte: Elaborado pelos autores.

A classe *TextMessage*, na Figura 4.3, é responsável por conectar-se no *modem* GSM em uma porta COM (serial ou USB), e por enviar mensagens de texto para um número IMSI de um dispositivo GSM destinatário.

As mensagens SMS são enviadas pelo método *sendMessage*, o tempo de envio para uma mensagem é a soma dos valores nos parâmetros das funções *time.sleep*. Na Figura 4.3 o tempo de envio é de 2,5 segundos por mensagem.

Nos testes realizados, as mensagens SMS enviadas com tempo inferior a 5 segundos chegaram fora de ordem no dispositivo receptor, e algumas mensagens não chegaram.

Em mensagens com tempo de envio superior a 5 segundos, também foi constatado que algumas mensagens não chegaram em ordem, porém todas as mensagens enviadas durante o teste chegaram no destino.

Com os testes foi possível identificar que para utilizar eficientemente a arquitetura de transmissão de dados é necessário o desenvolvimento de um servidor com *modem* GSM eficiente, de modo que receba as mensagens SMS de todos os receptores GPS instalados nos veículos de transporte públicos.

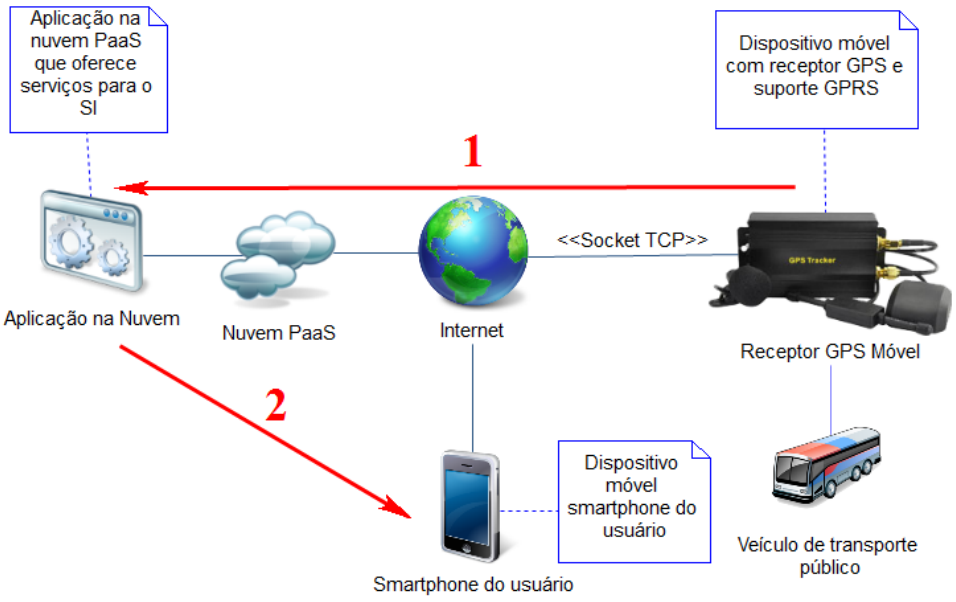
O servidor com *modem* GSM deve tratar a assincronia e a possível perda de mensagens SMS enviadas pelos receptores. O servidor também deve tratar a segurança pois mensagens SMS podem ser enviadas por qualquer dispositivo GSM, podendo ocorrer fraudes ou ataques de negação de serviços.

4.1.2 Arquitetura do SI com transmissão de dados via socket TCP

A arquitetura com transmissão de dados via *socket* TCP foi elaborada para utilizar o rastreador GPS TK103-2, que é um dispositivo próprio para rastreamento veicular.

Nessa arquitetura um rastreador GPS TK103-2 deve atuar na obtenção de coordenadas geográficas dos veículos de transporte público. As coordenadas geográficas dos veículos são enviadas utilizando *socket* TCP, a Figura 4.4 detalha a arquitetura.

Figura 4.4 – Arquitetura do SI com transmissão de dados via *socket* TCP



Fonte: Elaborado pelos autores.

Nessa arquitetura o SI seria composto pelos seguintes elementos: receptor GPS móvel, aplicação na nuvem e *smartphone* do usuário.

Com essa arquitetura o SI funcionária da seguinte forma, o receptor GPS móvel enviaria as coordenadas geográficas para uma aplicação na nuvem, como mostra a linha

vermelha número 1 da Figura 4.4. O envio de dados do receptor GPS móvel, deve utilizar os serviços de transferência de dados da rede móvel, por meio de *socket* TCP.

A aplicação na nuvem receberia as coordenadas geográficas enviadas pelo receptor GPS móvel e armazenaria em um banco de dados em nuvem.

Quando a aplicação cliente, do *smartphone* do usuário, solicitar pela localização de um veículo de transporte público, para a aplicação na nuvem, a coordenada geográfica do veículo será enviada para a aplicação cliente solicitante (linha vermelha número 2 da Figura 4.4). Com isso o usuário da aplicação cliente poderá obter informações de localização de um veículo de transporte público.

Para determinar a viabilidade dessa arquitetura, ocorreram testes no dispositivo receptor GPS TK103-2. Para testar o rastreador foi necessário a implementação de uma aplicação *socket* TCP.

A Figura 4.5 apresenta o código da implementação em Python da aplicação *socket* usada para os testes. A aplicação aceita a conexão de um cliente e imprime os dados enviados pelo mesmo.

Figura 4.5 – Código da implementação do servidor *socket* TCP

```
#!/usr/local/bin/python
import socket
import sys

# Cria um Socket TCP/IP
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# Vincula o Socket a porta, e ao host do computador
server_address = (socket.gethostname(),10000)

print '\n Iniciando servidor em %s na port %s ' % server_address
sock.bind(server_address)

# Define 5 conexoes em fila
sock.listen(5)

while True:

    # Espera por uma conexao
    print >> sys.stderr, '\n Esperando por uma conexao \n'

    connection, client_address = sock.accept()

    try:

        print ' Conexao de: ', client_address, '\n'

        while True:
            # Recebe os dados do cliente
            # Buffer de 1024 bytes
            data = connection.recv(1024)

            print '\n %s' % data

    finally:

        # Finaliza a conexao
        connection.close()
```

Fonte: Elaborado pelos autores.

Com a impressão dos dados enviados pelo cliente foi possível determinar os dados enviados pelo rastreador GPS TK103-2, que foi configurado para ser um cliente da aplicação *socket* TCP implementada.

A Figura 4.6 mostra a saída da execução da aplicação, com a conexão e envio de dados de um cliente, que é o rastreador GPS TK103-2.

Figura 4.6 – Saída da execução da aplicação *socket* TCP

```
Iniciando servidor em 127.0.0.1 na port 10000

Esperando por uma conexao

Conexao de:  ('179.174.75.63', 7996)


150509223439,0556799178209,GPRMC,023439.000,A,2213.0109,S,05448.1614,W,0.00,0.
00,100515,,,A*6B,F,, imei:013227006079663,11,460.6,F:4.
29V,1,139,59068,724,06,0A6B,041C

150509223506,0556799178209,GPRMC,023506.000,A,2213.0109,S,05448.1614,W,0.00,0.
00,100515,,,A*66,F,, imei:013227006079663,10,460.6,F:4.
29V,1,139,5362,724,06,0A6B,041C

150509223509,0556799178209,GPRMC,023509.000,A,2213.0109,S,05448.1614,W,0.00,0.
00,100515,,,A*69,F,, imei:013227006079663,10,460.6,F:4.
29V,1,139,18596,724,06,0A6B,041C

150509223523,0556799178209,GPRMC,023523.000,A,2213.0109,S,05448.1614,W,0.00,0.
00,100515,,,A*61,F,, imei:013227006079663,11,460.6,F:4.
29V,1,139,51556,724,06,0A6B,041C

150509223528,0556799178209,GPRMC,023528.000,A,2213.0109,S,05448.1614,W,0.00,0.
00,100515,,,A*6A,F,, imei:013227006079663,11,460.6,F:4.
29V,1,139,56898,724,06,0A6B,041C

150509223533,0556799178209,GPRMC,023533.000,A,2213.0109,S,05448.1614,W,0.00,0.
00,100515,,,A*60,F,, imei:013227006079663,11,460.6,F:4.
29V,1,139,35674,724,06,0A6B,041C
```

Fonte: Elaborado pelos autores.

A aplicação *socket* TCP foi executada em um computador, com porta aberta para a rede externa (internet). Na figura 4.6 o endereço 127.0.0.1 é o endereço local para o computador, sendo que foi necessário obter o endereço externo através do *modem* de internet do computador. Na Figura 4.6 o endereço 179.174.75.63 é o endereço do dispositivo GPS TK103-2 que conectou-se na aplicação *socket* TCP.

Em cada grupo de dados enviados pelo rastreador GPS TK103-2 (Figura 4.6) ao programa *socket*, é possível determinar as coordenadas geográficas e a identificação do dispositivo pelo número IMSI.

O tamanho de cada grupo de dados enviados pelo rastreador é de 165 bytes. Nos testes o rastreador enviou os dados com um intervalo de 30 segundos para cada mensagem. Com

uma configuração de intervalo de tempo inferior o rastreador não apresentou comunicação com a aplicação *socket*.

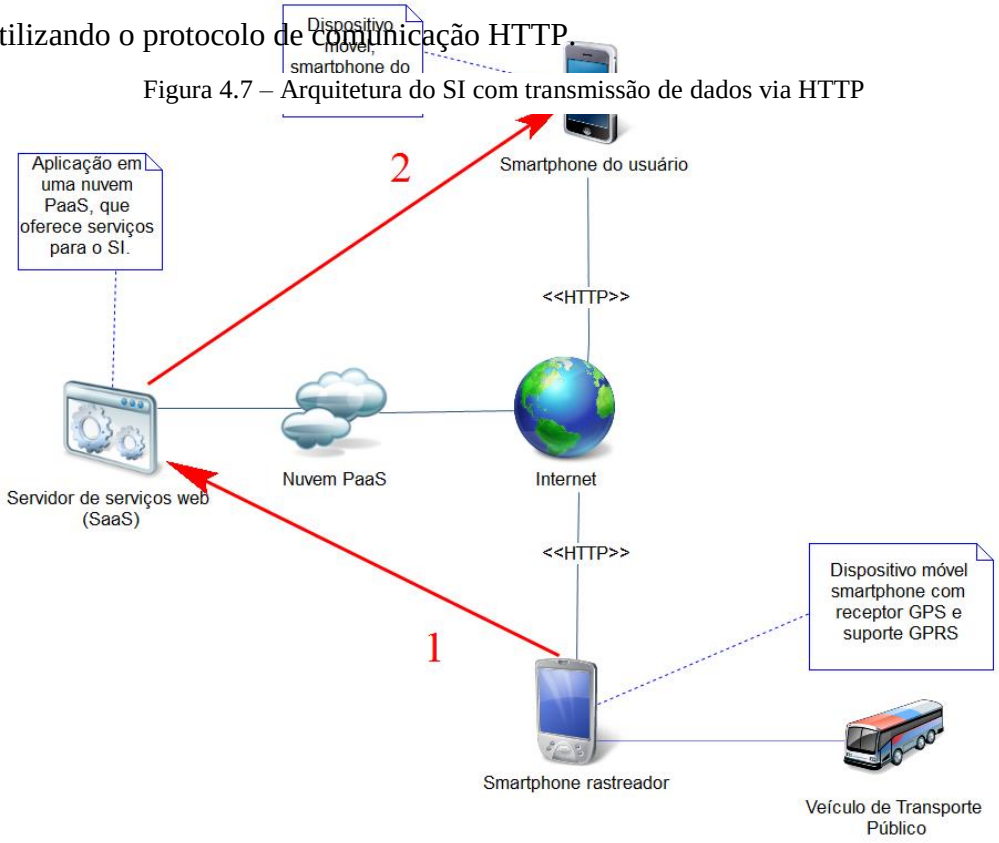
O rastreador GPS TK103-2 apresentou lentidão ou falha ao responder os comandos de configuração. O dispositivo também apresentou lentidão para obter os sinais GPS e da rede móvel GSM. Em local fechado o rastreador perde os sinais saindo de operação e reiniciando frequentemente.

A utilização dessa arquitetura exige o desenvolvimento de uma aplicação *socket* TCP eficiente, que suporte à conexão com segurança dos rastreadores GPS instalados nos veículos de transporte público. A aplicação *socket* TCP deve tratar ataques como o de negação de serviços e conexões não autorizadas.

4.1.3 Arquitetura do SI com transmissão de dados via HTTP

A arquitetura mostrada na Figura 4.7 é constituída por diversos componentes que interagem entre si enviando dados, alguns desses componentes são capazes de transmitir dados utilizando o protocolo de comunicação HTTP.

Figura 4.7 – Arquitetura do SI com transmissão de dados via HTTP



Fonte: Elaborado pelos autores.

O funcionamento da arquitetura mostrada na Figura 4.7 tem início no *samrtphone* rastreador, que obtém as coordenadas geográficas do veículo de transporte público e as envia para o servidor de serviços web, como mostrado pela seta vermelha com o número 1, que liga o *smartphone* rastreador ao servidor de serviços web.

O servidor de serviços web em nuvem, ao receber as coordenadas enviadas pelo *smartphone* rastreador as armazena em um banco de dados em nuvem.

Por fim, ao ser solicitado pelo *smartphone* do usuário o servidor de serviços web retorna as últimas coordenadas geográficas do veículo de transporte para o mesmo, como mostra a seta vermelha com o número 2 na Figura 4.7.

Nessa arquitetura foi realizado um teste com o objetivo de analisar o comportamento de três tecnologias funcionando conjuntamente, o GPS do *smartphone* rastreador, a biblioteca de requisições HTTP Volley e o servidor de serviços web, as variáveis observadas foram: obtenção de coordenadas geográficas, perda de dados e corretude dos dados.

O teste consistiu em: enviar uma requisição do *smartphone* rastreador contendo coordenadas geográficas para o servidor de serviços web.

As Figuras 4.8 a 4.11 compõem o cenário do teste. O início do teste se dá a partir da Figura 4.8, que mostra uma função chamada *startGPS*, cuja responsabilidade é inicializar o GPS para que a função *updateCoordenate* seja chamada de 1 em 1 segundo para obter as coordenadas geográficas.

Após a função *updateCoordenate* obter as coordenadas geográficas, a função *sendToServer* é chamada, ela pode ser vista na Figura 4.9, sua finalidade é enviar as coordenadas para o servidor de serviços web, sendo assim, ela realiza uma requisição HTTP POST enviando um formulário contendo as coordenadas geográficas para o servidor de serviços web.

A Figura 4.10 mostra a atuação do servidor, que ao receber as coordenadas geográficas as armazena em uma variável global se os dados estiverem corretos, dessa forma o valor 1 é retornado indicando sucesso, caso contrário, os dados não são armazenados e o valor 0 é retornado indicando falha.

E por fim, o *smartphone* rastreador pode visualizar o *status* de recebimento das coordenadas geográficas, que pode ser visto na Figura 4.11.

Figura 4.8 – Função para capturar as coordenadas geográficas de um dispositivo móvel de 1 em 1 segundo

```
public void startGPS(){
    LocationManager lManager = (LocationManager)getSystemService(Context.LOCATION_SERVICE);
    LocationListener lListener = new LocationListener() {
        public void onLocationChanged(Location locat) {
            updateCoordenate(locat);
        }
        public void onStatusChanged(String provider, int status, Bundle extras) {}
        public void onProviderEnabled(String provider) {}
        public void onProviderDisabled(String provider) {}
    };
        lManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, 1000, 0, lListener);
}

public void updateCoordenate(Location locat){
    latitude = locat.getLatitude();
    longitude = locat.getLongitude();
    sendDataToServer();
}
```

Fonte: Elaborado pelos autores.

```
public void sendDataToServer(){
```

Figura 4.9 – Função para enviar as coordenadas geográficas de um dispositivo móvel para um servidor web

```
    postReq = new StringRequest(Request.Method.POST, url, new Response.Listener<String>() {
        @Override
        public void onResponse(String response) {
            TextView response_text = (TextView) findViewById(R.id.id_large_text);
            response_text.setText(response);

        }
    }, new Response.ErrorListener() {
        @Override
        public void onErrorResponse(VolleyError error) {
            System.out.println("Error ["+error+"]");

        }

    }){
        @Override
        protected Map<String, String> getParams(){

            Map<String, String> params = new HashMap<String, String>();
            params.put("latitude",   String.valueOf(latitude));
            params.put("longitude",  String.valueOf(longitude));
            return params;
        }

    } ;
    queue.add(postReq);
}
```

Fonte: Elaborado pelos autores.

Figura 4.10 – Função do servidor de serviços web para receber coordenadas geográficas

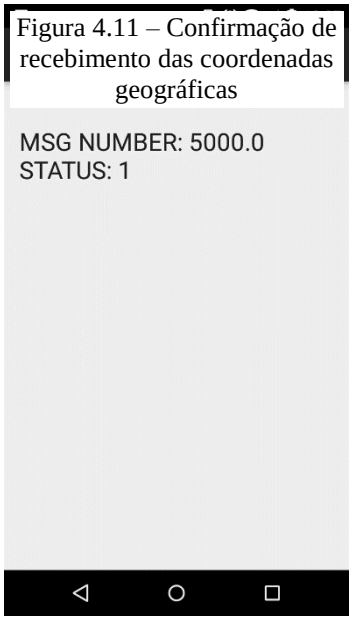
```
@bus_localization_blue_print.route('/data', methods=['GET','POST'])
def data():

    global dic_cord
    try:
        dic_cord['latitude'] = request.form['latitude']
        dic_cord['longitude'] = request.form['longitude']
    except:
        return "0"

    return "1"
```

Fonte: Elaborado pelos autores.

Figura 4.11 – Confirmação de recebimento das coordenadas geográficas



Fonte: Elaborado pelos autores.

O teste foi repetido 5000 vezes, não houve falha na obtenção das coordenadas, não houve perda de dados e os dados se mostraram corretos ao chegarem ao servidor de serviços web. Durante a fase de teste foi possível analisar alguns dos componentes da arquitetura de forma individual.

A biblioteca de requisições Volley possui algumas características como múltiplas conexões de rede simultâneas, agendamento automático de solicitações de rede, cancelamento de requisições e envio de requisições de forma assíncrona.

Apesar do funcionamento da biblioteca Volley ser assíncrono por natureza, ela possui um mecanismo de ordenação forte, que permite realizar requisições e receber respostas em ordem. Devido a sua natureza configurável, a biblioteca Volley se mostra adequada para o tipo de arquitetura que necessita fazer requisições em massa e em um curto período de tempo sem perda de dados.

O servidor de serviços web e o banco de dados são peças fundamentais nessa arquitetura, estando ativos na infraestrutura de nuvem da Google, eles se beneficiam de aspectos como segurança da informação, disponibilidade, escalabilidade e confiabilidade, logo, não se faz necessário a implementação de tais aspectos para a aplicação.

A fonte inicial dos dados e peça central dessa arquitetura é o *smartphone* rastreador, ele apresenta um *hardware* poderoso e robusto, com grande poder de processamento e memória, além de um serviço de GPS preciso, que pode funcionar até mesmo em locais fechados.

Ao utilizar a arquitetura com transmissão HTTP, alguns aspectos como escalabilidade, disponibilidade de serviços e outros são garantidos pelo prestador de serviços, que mantém o servidor de serviços web. Porém, a migração de alguns serviços como o banco de dados é algo que nem todos os prestadores de serviços garantem.

4.1.4 Escolha da melhor arquitetura

Através dos estudos das tecnologias foi possível elaborar três diferentes arquiteturas capazes de representar o protótipo do SI. Durante os testes realizados em cada uma das arquiteturas, várias características foram identificadas.

Cada arquitetura apresentou um conjunto similar de características entre si, que podem ser vistas no Quadro 4.1.

Características	Arq SMS	Arq TCP	Arq HTTP
Custo de implementação da arquitetura	Baixo	Baixo	Médio
Custo para sincronizar o envio dos dados	Alto	Médio	Baixo
Múltiplas conexões simultâneas	Nula	Alta	Alta
Perda de dados	Alta	Nula	Nula
Capacidade de georreferenciar em tempo real	Baixa	Alta	Alta
Custo de implementação de segurança dos dados	Alto	Alto	Médio

Quadro 4.1 – Características similares entre as arquiteturas

Fonte: Autoria própria.

As características do Quadro 4.1 podem ser interpretadas da seguinte forma:

1) O custo de implementação da arquitetura pode ser estimado de acordo com a quantidade de tecnologias utilizadas e o provável número de linhas de código escritas.
2) O custo para sincronizar o envio de dados pode ser estimado conforme os mecanismos disponibilizados pelas tecnologias utilizadas em cada arquitetura e o provável número de linhas de código escritas.
3) Múltiplas conexões simultâneas é um mecanismo que está diretamente relacionado ao conjunto de tecnologias utilizadas em cada arquitetura. As arquiteturas podem ou não dispor desse mecanismo em decorrência das tecnologias utilizadas.
4) A perda de dados diz respeito o quão eficiente é uma arquitetura ao enviar dados de forma que eles não se percam.
5) A capacidade de georreferenciar em tempo real está relacionada ao atraso da arquitetura para enviar os dados do rastreador ao servidor de serviços web, quanto menor for o atraso melhor será a capacidade de georreferenciar em tempo real.
6) O custo de implementação de segurança dos dados pode ser estimado em função das tecnologias utilizadas em cada arquitetura, dos mecanismos que essas dispõem para facilitar a implementação e o provável número de linhas de código escritas.

Analisando o Quadro 4.1 pode-se perceber que para cada característica de uma arquitetura foi atribuído um valor, que varia entre Nulo e Alto. Os valores atribuídos as características são qualitativos, o que permite interpretar o quão bom ou ruim é uma determinada característica de uma arquitetura em comparação as demais.

Ao observar o Quadro 4.1 e comparar os valores das características de cada arquitetura, foi possível concluir por comparação que, a arquitetura com transmissão HTTP se mostrou mais viável para ser implementada dentre as arquiteturas apresentadas. A próxima subseção descreverá o desenvolvimento do projeto do protótipo do SI com a arquitetura com transmissão HTTP.

4.2 Projeto do protótipo do SI

Ao escolher a arquitetura com transmissão HTTP para ser implementada e representar o SI, foram identificados três subsistemas, são eles: servidor de serviços web, aplicativo do usuário e aplicativo de rastreamento.

O protótipo do aplicativo de rastreamento e do usuário foram projetados para a plataforma Android, já o servidor de serviços web foi projetado baseando-se nos padrões da *framework* Flask.

4.2.1 Protótipo do servidor de serviços web

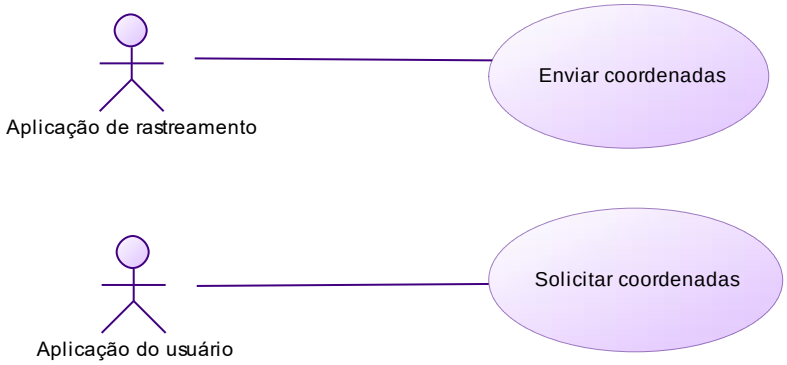
Os requisitos funcionais identificados para o servidor de serviços web são:

RF1 – O servidor de serviços deve armazenar as coordenadas geográficas que a aplicação de rastreamento lhe enviar.

RF2 – O servidor de serviços deverá permitir que a aplicação do usuário solicite as atuais coordenadas geográficas de um veículo de transporte.

Os requisitos funcionais podem ser visualizados no diagrama de caso de uso da Figura 4.12.

Figura 4.12 – Diagrama de caso do uso do protótipo do servidor web



Fonte: Elaborado pelos autores.

O protótipo do servidor web possui dois tipos de usuários: Aplicação do usuário, que poderá solicitar as coordenadas geográficas de um veículo de transporte ao servidor web e a aplicação de rastreamento, que poderá enviar as coordenadas geográficas do dispositivo móvel para o servidor web.

As descrições dos casos de uso do protótipo do servidor web podem ser vistas com detalhes nos Quadros 4.2 e 4.3.

Caso de uso:	Enviar coordenadas.
Ator:	Aplicação de rastreamento
Sumário:	Este caso de uso permite que o ator envie coordenadas geográficas para o servidor de serviços web.
Pré-condição:	
RF: RF1	**RNF:** --
Fluxo Principal:	
1. A aplicação de rastreamento solícita conexão de internet. 2. A aplicação consegue se conectar com a internet. 3. A aplicação de rastreamento envia as coordenadas geográficas ao servidor web. 4. O servidor web retorna uma mensagem confirmando o sucesso no envio das coordenadas geográficas. 5. O caso de uso é encerrado.	
Fluxo Alternativo (1): A aplicação já está conectada à internet.	
a. O sistema avança para o passo 3 do fluxo principal.	
Fluxo Alternativo (2): A aplicação não consegue se conectar à internet.	
a. O sistema retorna ao passo 1 do fluxo principal.	
Fluxo Alternativo (3): A aplicação perde a conexão enquanto tenta enviar dados.	
a. O sistema retorna para o passo 1 do fluxo principal.	
Fluxo Alternativo (4): O servidor está inativo.	
a. O sistema retorna para o passo 3 do fluxo principal.	
Fluxo Alternativo (4): Falha no envio das coordenadas geográficas ou servidor está inativo.	
a. O sistema emite uma mensagem de coordenadas geográficas inválidas para a aplicação de rastreamento. b. O sistema retorna ao passo 3 do fluxo principal.	
Pós condição:	Nenhuma
Relacionamentos: Não há relacionamentos para esse caso de uso	

Quadro 4.2 – Descrição do caso de uso: Enviar coordenadas

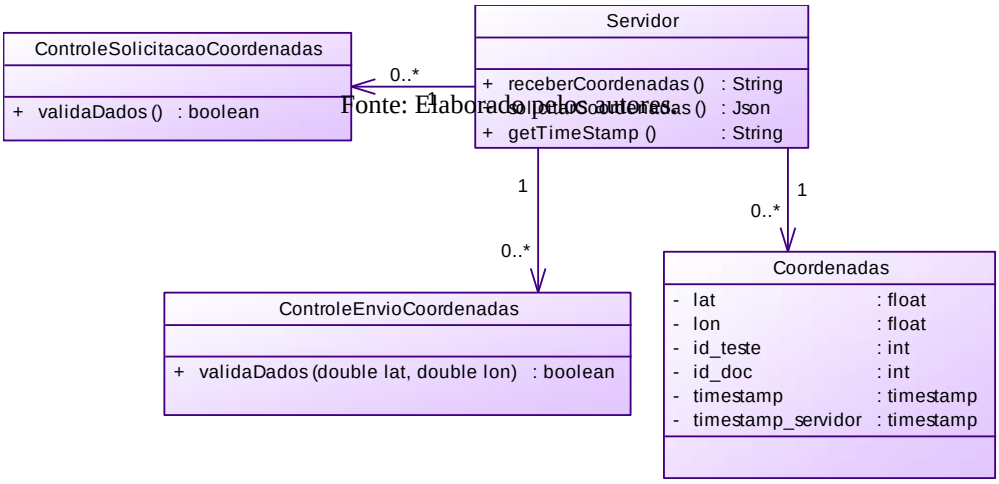
Caso de uso:	Solicitar coordenadas.
Ator:	Aplicação do usuário
Sumário:	Este caso de uso permite que o ator solicite coordenadas geográficas do servidor de serviços web.
Pré-condição:	Nenhuma
RF: RF2	**RNF:** --
Fluxo Principal:	
1. A aplicação do usuário faz uma solicitação de coordenadas geográficas ao servidor web. 2. O servidor web envia para a aplicação do usuário as coordenadas geográficas. 3. O caso de uso é encerrado.	
Pós condição:	Nenhuma.
Relacionamentos: Não há relacionamentos para esse caso de uso.	

Fonte: Elaborado pelos autores.

Quadro 4.3 – Descrição do caso de uso: Solicitar coordenadas

O diagrama de classes que compõem o protótipo do servidor web pode ser visualizado na Figura 4.13.

Figura 4.13 – Diagrama de classes do protótipo do servidor web



Fonte: Elaborado pelos autores.

O diagrama de classes da Figura 4.13 está dividido em quatro classes: *Servidor*, *ControleSolicitacaoCoordenadas*, *ControleEnvioCoordenadas e Coordenadas*.

A classe *Servidor* é responsável por receber as solicitações de serviços, invocar as classes de controle para validar os dados e por fim, enviar os dados para a classe *Coordenadas*. A classe possui três operações, *receberCoordenadas, solicitarCoordenadas e getTimeStamp*.

A operação *receberCoordenadas* permite receber coordenadas geográficas, a operação *solicitarCoordenadas* permite obter coordenadas junto ao servidor e a operação *getTimeStamp* permite obter o *timestamp* do servidor no formato (data, horas, minutos e segundos).

As classes *ControleSolicitacaoCoordenadas* e *ControleEnvioCoordenadas* têm a responsabilidade de controlar os dados vindos da solicitação de serviços da classe *Servidor*. Suas operações permitem validar os dados.

Finalmente, a classe *Coordenadas* tem a responsabilidade de ser um molde para um objeto, cujas informações serão armazenadas em um banco de dados de forma ordenada.

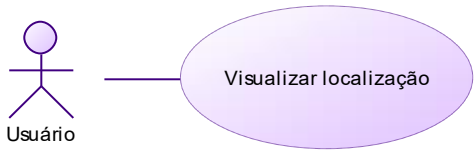
4.2.2 Protótipo do aplicativo do usuário

Com o objetivo de realizar testes do protótipo do SI, o único requisito funcional identificado para o protótipo do aplicativo do usuário foi:

RF1 – Quando o aplicativo do usuário estiver conectado à internet, o usuário poderá visualizar em um mapa a localização de um veículo de transporte em tempo real e o trajeto já realizado pelo veículo.

A Figura 4.14 é o diagrama de caso de uso do para o requisito funcional RF1.

Figura 4.14 – Diagrama de caso de uso do protótipo do aplicativo do usuário



Fonte: Elaborado pelos autores.

O aplicativo terá somente um utilizador, no caso de uso o tipo único de utilizador é chamado de Usuário. A descrição do caso de uso da Figura 4.14 segue na Quadro 4.4 a seguir.

Caso de uso:	Visualizar localização
Ator:	Usuário
Sumário:	Este caso de uso permite que o usuário visualize a localização de um veículo de transporte, em um mapa.
Pré-condição:	Nenhuma.
RF: RF1	**RNF:** Nenhum
Fluxo Principal:	
1. O aplicativo verifica a conexão com a internet. 2. O aplicativo baixa o mapa pela internet e o exibe. 3. O aplicativo tenta obter do servidor de serviços web a localização de um veículo de transporte e o trajeto já realizado pelo veículo. 4. O aplicativo exibe no mapa a última localização e o trajeto do veículo de transporte obtida no servidor de serviços web. 5. O aplicativo espera o usuário encerrar o caso de uso. 6. O usuário não encerra o caso de uso e o aplicativo retorna para o passo 3 do fluxo principal.	
Fluxo Alternativo (1)**:** Não existe conexão com a internet.	
a) O aplicativo espera o usuário encerrar o caso de uso. b) O usuário não encerra o caso de uso e o aplicativo retorna para o passo 1 do fluxo principal.	
Fluxo Alternativo (4)**:** Nenhuma localização de veículo de transporte foi obtida.	
a) O aplicativo espera o usuário encerrar o caso de uso. b) O usuário não encerra o caso de uso e o aplicativo retorna para o passo 3 do fluxo principal.	
Fluxo Alternativo (1b, 4b, 6)**:** O usuário encerra o caso de uso.	
O aplicativo é encerrado.	
Pós condição:	Nenhuma.

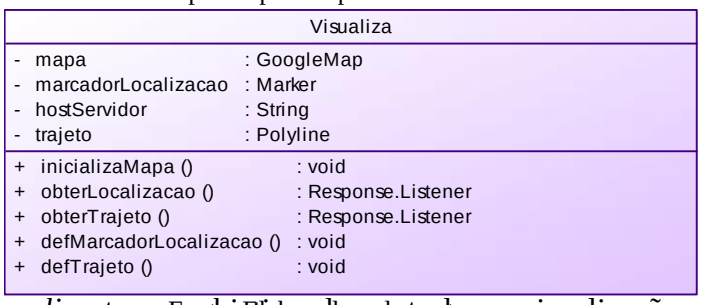
Quadro 4.4 – Descrição do caso de uso: Visualizar localização

Relacionamentos:
Nenhum.

Fonte: Elaborado pelos autores.

A Figura 4.15 é o Diagrama de classe de projeto para o caso de uso Visualizar localização.

Figura 4.15 – Diagrama de classes do projeto do protótipo do aplicativo do usuário



Fonte: Elaborado pelos autores.

A classe *Visualiza* tem o objetivo de controlar a visualização da localização e do trajeto de um veículo em um mapa para o aplicativo do usuário. Os atributos e as operações da classe são: *mapa*, *marcadorLocalizacao*, *hostServidor*, *inicializaMapa*, *trajeto*, *obterLocalizacao*, *defMarcadorLocalizacao* e *defTrajeto*.

O atributo *mapa* é um objeto utilizado pela classe para a criação e visualização de mapas. A localização de um veículo é definida no mapa pelo atributo *marcadorLocalizacao*, o trajeto realizado pelo veículo é definido pelo atributo *trajeto*.

A operação *inicializaMapa* tem por objetivo obter e instanciar o objeto *mapa*. Por conta da tecnologia a ser utilizada, o objeto *mapa* deve ser obtido pela internet.

A obtenção da localização do veículo no servidor de serviços web é de responsabilidade da operação *obterLocalizacao*. A operação *obterTrajeto* tem a responsabilidade de obter o trajeto já realizado pelo veículo.

A operação *defMarcadorLocalizacao* tem a responsabilidade de definir o atributo *marcadorLocalizacao* com a localização, do veículo, obtida na operação *obterLocalizacao*.

O atributo *trajeto* é definido pela operação *defTrajeto* com o trajeto obtido na operação *obterTrajeto*.

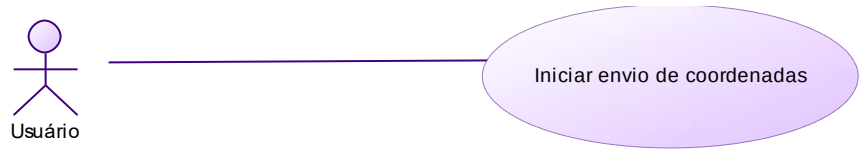
4.2.3 Protótipo do aplicativo de rastreamento

O requisito funcional identificado para o aplicativo de rastreamento é:

RF1 – A aplicação de rastreamento deverá permitir que (usuário) possa iniciar o envio de coordenadas geográficas do dispositivo móvel para o servidor web.

Os requisitos funcionais podem ser visualizados no diagrama de caso de uso da Figura 4.16.

Figura 4.16 – Diagrama de caso de uso do protótipo da aplicação de rastreamento



Fonte: Elaborado pelos autores.

O protótipo da aplicação de rastreamento possui um único usuário, que poderá inicializar o envio de coordenadas geográficas para o servidor web.

A descrição do caso de uso do protótipo do aplicativo de rastreamento pode ser visto com detalhes no Quadro 4.5.

Caso de uso:	Iniciar envio de coordenadas.
Ator:	Usuário
Sumário:	Este caso de uso permite que o ator inicialize o envio de coordenadas geográficas para o servidor web.
Pré-condição:	

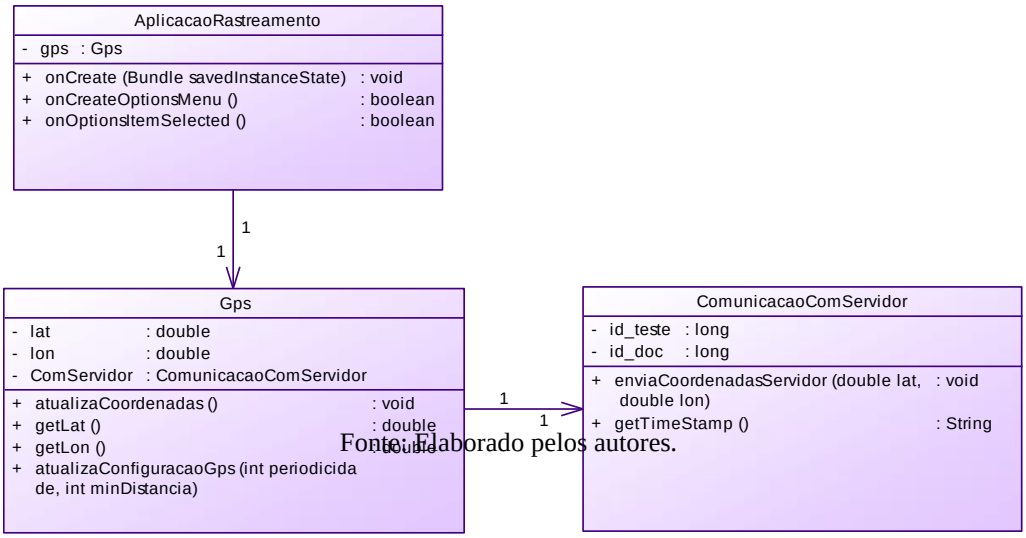
Quadro 4.5 –

RF: RF1	**RNF:** --
Fluxo Principal:	
1. O usuário inicializa a aplicação. 2. A aplicação de rastreamento solícita conexão de internet. 3. A aplicação consegue se conectar com a internet. 4. A aplicação de rastreamento começa o envio de coordenadas geográficas ao servidor web, a aplicação permanece no estado de envio.	
Fluxo Alternativo (2): A aplicação já está conectada à internet.	
a. O sistema avança para o passo 4 do fluxo principal.	
Fluxo Alternativo (3): A aplicação não consegue se conectar à internet.	
a. O sistema retorna ao passo 2 do fluxo principal.	
Fluxo Alternativo (4): A aplicação perde a conexão enquanto tenta enviar dados ou há problemas de conexão com servidor.	
a. O sistema retorna para o passo 2 do fluxo principal.	
Pós condição:	Nenhuma
Relacionamentos: Não há relacionamentos para esse caso de uso	

Descrição do caso de uso: Iniciar envio de coordenadas

O diagrama de classes que compõem o protótipo do aplicativo de rastreamento pode ser visualizado na Figura 4.17.

Figura 4.17 – Diagrama de classes do protótipo do aplicativo de rastreamento



Fonte: Elaborado pelos autores.

Fonte: Elaborado pelos autores.

O diagrama de classes da Figura 4.17 está dividido em três classes, são elas: *AplicacaoRastreamento, Gps e ComunicacaoComservidor*.

A classe *AplicacaoRastreamento* é responsável por criar a tela principal da aplicação, ela dispõem de três operações que dão suporte a manipulação do contexto e de menu.

A classe *Gps* tem a responsabilidade de obter as coordenadas geográficas do dispositivo móvel e repassá-las para a classe *ComunicacaoComServidor*. Suas operações

permitem atualizar as coordenadas geográficas e a configurar a forma como elas são obtidas, ou seja, em função do tempo e do deslocamento do dispositivo móvel.

Por fim, a classe *ComunicacaoComServidor* tem a responsabilidade de se comunicar com o servidor web. A classe possui duas operações, *enviaCoordenadasServidor* e *getTimeStamp*.

A operação *enviaCoordenadasServidor* recebe coordenadas geográficas da classe *Gps e* a operação *getTimeStamp* obtém o *timestamp* no formato (data, horas, minutos e segundos). As informações obtidas pelas operações são enviadas ao servidor, juntamente com o identificador de teste e documento, que são gerados incrementalmente pela classe.

4.3 Implementação do protótipo do SI

A implementação do protótipo do SI foi dividida em três partes: um aplicativo para rastreamento, instalada no *smartphone* rastreador, um aplicativo do usuário, instalada no *smartphone* do usuário e por fim um servidor web na nuvem, que oferece serviços para as duas aplicações anteriores. Os códigos fontes da implementação estão na mídia física solicitada para este trabalho.

4.3.1 Protótipo do servidor de serviços web

O protótipo do servidor web foi implementado com o auxílio da IDE Pycharm, que cria e gerencia projetos baseados em GAE e Flask usando a linguagem de programação Python, além de instalar todas as dependências necessárias para a implementação da aplicação.

Já o armazenamento de dados foi implementado usando a API NDB, para utilizar a API foi necessário a implementação de um classe conhecida como *models*, que mapeia objetos para o banco de dados, a API está disposta no pacote *google.appengine.ex*.

Por fim, a implementação da comunicação com as aplicações foi feita utilizando um documento conhecido como *JavaScript Object Notation* (JSON), que é enviado para as aplicações solicitantes.

4.3.2 Protótipo do aplicativo do usuário

O aplicativo do usuário foi implementado usando a API de mapa Google Maps e a biblioteca de requisições HTTP do Google chamada Volley. O aplicativo foi implementado na IDE Android Studio.

No desenvolvimento do protótipo do aplicativo do usuário a classe de projeto *Visualiza* da Figura 4.16 foi implementada.

Na implementação da operação *obterLocalizacao* e *obterTrajeto* da classe *Visualiza* foi necessário a utilização da biblioteca de requisição HTTP Volley, por conta da comunicação com o servidor de serviços web.

Para instanciar o atributo *mapa*, pela operação *inicializaMapa,* foi utilizado na implementação a classe *GoogleMap*, da API de mapas do Google.

Os atributos e as operações: *marcadorLocalizacao, trajeto, defMarcadorLocalizacao* e *defTrajeto*, foram implementados utilizando a classe *Marker* e *Polyline*, da API de mapas do Google.

4.3.3 Protótipo do aplicativo de rastreamento

O protótipo do aplicativo de rastreamento foi implementado com o auxílio da IDE Android Studio, que cria e gerencia projetos de aplicações para a plataforma Android em linguagem Java.

Para implementar a obtenção das coordenadas geográficas, foram utilizados algumas classes e interfaces presentes no pacote *android.location,* que são: *Location*, *LocationListener* e *LocationManager*

A comunicação feita pela aplicação com servidor web foi implementada usando a biblioteca de requisições HTTP Volley, que utiliza um conjunto de classes para gerenciar uma fila de requisições e o envio de dados em um formulário com o método POST, as classes utilizadas foram: *Request*, *RequestQueue*, *Response* e *VolleyError*.

Os três protótipos implementados conjuntamente, após a fase de projeto, finalizam a construção do protótipo do SI, portanto, o próximo capítulo descreverá os resultados alcançados através dos testes realizados no mesmo.

5 RESULTADOS

O protótipo do SI foi testado através de percursos de uma linha da empresa de transporte público Viação Dourados, que atua em Dourados – Mato Grosso do Sul, Brasil. O objetivo do teste foi verificar o funcionamento do protótipo do SI.

Os testes ocorreram em um veículo da linha cidade universitária, através do aplicativo de rastreamento, instalado no *smartphone* rastreador. O *smartphone* utilizado no teste foi o Motorola G2, com 1 GB de memória RAM e processador *Quad Core* de 1.2 GHz, o sistema operacional usado pelo dispositivo foi o Android versão 5.0.2.

Ocorreram dois testes na linha cidade universitária, o percurso de um dos testes ocorreu no sentido terminal – cidade universitária e outro, no sentido cidade universitária – terminal via florida I.

No percurso onde o veículo saiu do terminal á cidade universitária o *smartphone* rastreador foi configurado para obter as coordenadas geográficas somente pelo GPS do dispositivo. No percurso onde o veículo saiu da cidade universitária ao terminal o dispositivo estava configurado para obter coordenadas geográficas com alta precisão, ou seja, utilizando redes móveis, GPS e *wireless*.

Em cada percurso realizado durante os testes, o aplicativo de rastreamento atuou enviando coordenadas a cada deslocamento de 5 metros. As coordenadas geográficas enviadas pelo aplicativo são as do *smartphone* rastreador, que correspondem com as coordenadas geográficas do veículo.

Para obter informações importantes do teste, os dados das mensagens enviadas pelo aplicativo de rastreamento para o servidor de serviços web, foram: coordenadas geográficas, data, horas, minutos e segundos de obtenção de coordenadas geográficas, número de mensagens já enviadas e identificação do teste.

As mensagens enviadas foram recebidas pelo servidor de serviços web e armazenadas, o servidor também armazenou a data, horas, minutos e segundos em que cada mensagem foi recebida.

Com os dados gerados no teste foi possível determinar o tempo em que cada mensagem levou para ser recebida pelo servidor de serviços web, como também o tempo mínimo de obtenção das coordenadas geográficas pelo aplicativo de rastreamento no *smartphone*.

Em ambos os testes as mensagens enviadas pelo *smartphone* rastreador tiveram um atraso aproximado de dois segundos para chegar no servidor de serviços web. No teste em que o *smartphone* obteve as coordenadas somente pelo GPS, o tempo para obtenção das coordenadas foi de aproximadamente um segundo. Nos testes em que as coordenadas foram obtidas utilizando redes móveis, GPS e *wireless* o tempo aproximado para obtenção das coordenadas geográficas foi de dois segundos. O Quadro 5.1 mostra as diferenças dos tempos de ambos os testes.

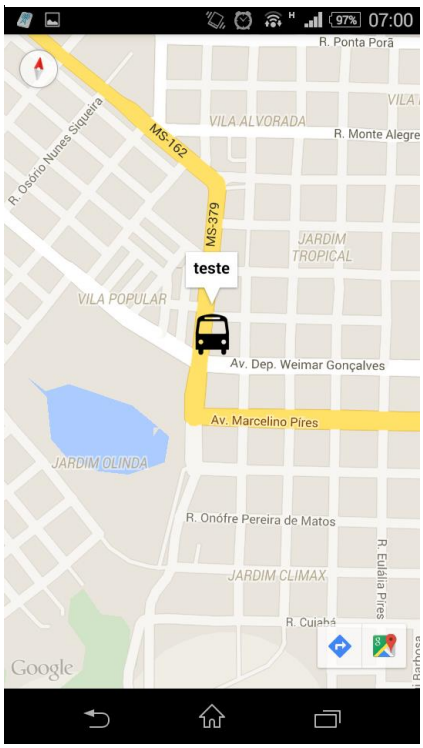
Quadro 5.1 – Tempos obtidos durante os testes.

Modo de obtenção de coordenadas geográficas	**Tempo de atraso das mensagens para chegar no servidor de serviços web**	**Tempo para obtenção das coordenadas geográficas no *smartphone* rastreador**
Somente GPS	2 Segundos	1 Segundo
Redes móveis, GPS e wireless	2 Segundos	2 Segundos

Fonte: Elaborado pelos autores.

Pelo aplicativo do usuário, instalado em outro *smartphone*, foi possível verificar que a localização do veículo mostrada no mapa correspondeu com a localização real do veículo em um momento do teste. A Figura 5.1 é uma imagem de tela do aplicativo do usuário obtida durante o teste.

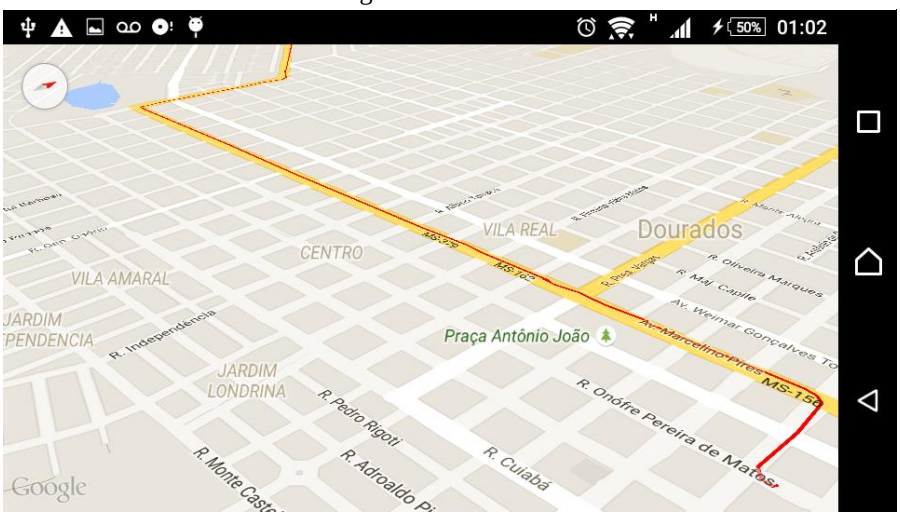
Figura 5.1 – Tela do aplicativo do usuário, obtida durante o teste



Fonte: Elaborado pelos autores.

As coordenadas enviadas pelo aplicativo de rastreamento foram salvas, com a finalidade de verificar o trajeto realizado pela linha do veículo georreferenciado. O trajeto realizado pelo veículo no sentido terminal – cidade universitária é mostrado na Figura 5.2.

Figura 5.2 – Tela do aplicativo do usuário com o trajeto do veículo georreferenciado

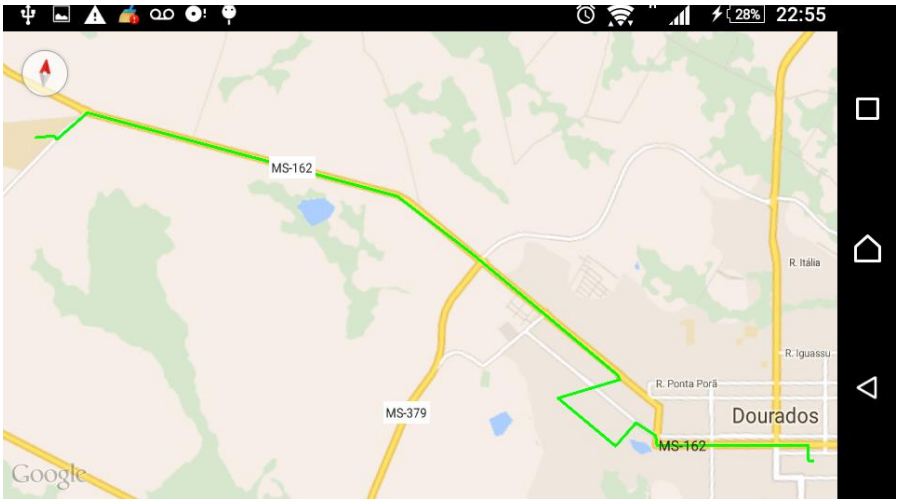


Fonte: Elaborado pelos autores.

Na Figura 5.2, a linha vermelha é o trajeto realizado durante o teste, observa-se que a linha corresponde ao percurso realizado por um veículo da linha cidade universitária.

A Figura 5.3 mostra o trajeto do veículo no sentido cidade universitária – terminal via florida I.

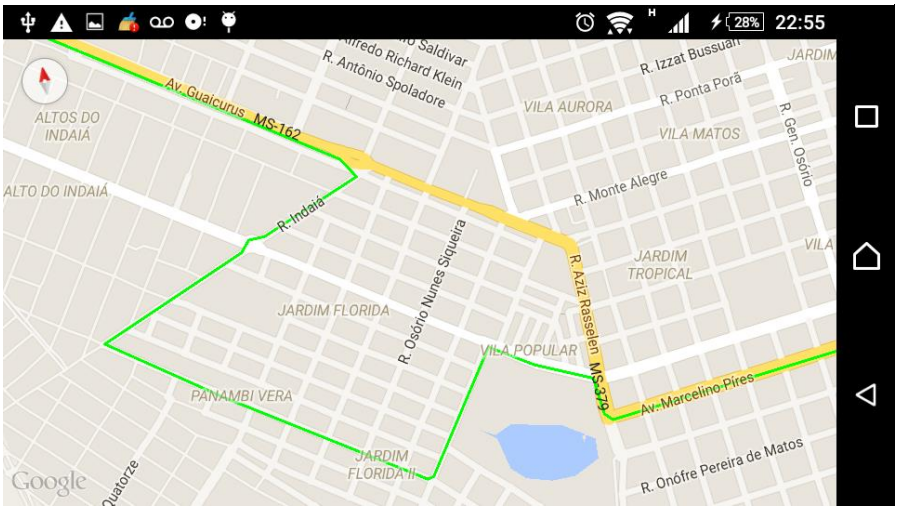
Figura 5.3 – Tela do aplicativo do usuário com o trajeto do veículo georreferenciado



Fonte: Elaborado pelos autores.

Através da Figura 5.4 é possível visualizar o trajeto do veículo no sentido cidade universitária – terminal via florida I, com uma maior aproximação.

Figura 5.4 – Tela do aplicativo do usuário com o trajeto do veículo georreferenciado



Fonte: Elaborado pelos autores.

Pela Figura 5.4, é possível observar a mudança do trajeto do veículo, observa-se também que a linha verde, que representa o trajeto, coincide com as vias, o que demonstra a precisão das coordenadas geográficas obtidas durante o teste.

6 CONCLUSÃO

6.1 Considerações Finais

Devido a aleatoriedade do trânsito, condições climáticas, más condições das vias públicas e etc, os usuários de transporte público da cidade de Dourados têm dificuldades de saber a localização dos ônibus de uma determinada linha. Com o crescente avanço das tecnologias móveis e ubíqua, cresce também as necessidades desses usuários por obter informações com maior rapidez e precisão.

Motivado pelas reais dificuldades e necessidades dos usuários de transporte público da cidade de Dourados, este trabalho buscou criar um protótipo de SI que fosse capaz de georreferenciar um dado veículo de transporte na cidade de Dourados em tempo real.

Para alcançar o objetivo da construção do protótipo do SI, foi necessário estudar diversas tecnologias e conceitos, que deram suporte à criação de modelos (arquiteturas) capazes de expor uma possível solução com riqueza de detalhes.

A partir da construção das arquiteturas, foi possível analisar a viabilidade de cada uma, baseando-se no conceito de custo e benefício. Uma arquitetura com transmissão de dados HTTP foi escolhida dentre as três estudadas nesse trabalho, pois se mostrou mais adequada em termos práticos e de custo.

A construção do protótipo do SI com a arquitetura escolhida exigiu adotar tecnologias de implementação modernas, que foram capazes de agilizar o processo de desenvolvimento, minimizar os custos com implementação, garantir segurança, economizar com infraestrutura de servidor e atender as demandas de *software* em tempo hábil.

Pelos testes realizados o protótipo do SI se mostrou capaz de georreferenciar os veículos de transporte público em tempo real de forma eficiente e sem perdas de informações.

6.2 Trabalhos Futuros

O protótipo do SI desenvolvido nesse projeto está em fase embrionária, para o futuro é possível adicionar novos requisitos ao projeto, de modo que o SI possa ser implantável. Alguns possíveis requisitos são: uma máquina de gerenciamento, capaz de configurar a aplicação do *smartphone* rastreador e o servidor de serviços web; disponibilizar à aplicação do usuário a funcionalidade de monitorar todos os veículos de transporte em atividade de uma

determinada linha; disponibilizar à aplicação do usuário a funcionalidade de monitorar apenas um veículo de transporte de uma determinada linha; disponibilizar à aplicação do usuário a funcionalidade de configurar a frequência de atualização da posição do veículo de transporte; disponibilizar à aplicação do usuário a funcionalidade de encontrar a parada de ônibus mais próxima de si.

Com a adição de novos requisitos, o protótipo do SI se tornará mais robusto, podendo ser utilizado nos negócios como um SI especialista, que dá apoio na tomada de decisão.

REFERÊNCIAS BIBLIOGRÁFICAS

APP Engine: A powerful platform to build web and mobile apps that scale automatically. Disponível em: <https://cloud.google.com/appengine/>. Acesso em: 27 out. 2015.

APPLE e Samsung se unem para acabar com o cartão SIM. Disponível em: <http://www.hardware.com.br/noticias/2015-07/apple-samsung-se-unem-para-acabar-com-cartao-sim.html>. Acesso em: 06 out. 2015.

ASSOCIAÇÃO NACIONAL DE TRANSPORTES PÚBLICOS (ANTP). **Sistemas Inteligentes de Transportes**. The World Bank, maio 2012. Séries de Cadernos Técnicos, volume 8.

BODVOC'S Blog: An Overview of a Web Server. 02 jul. 2010. Disponível em: <https://bodvoc.wordpress.com/2010/07/02/an-overview-of-a-web-server/>. Acesso em: 27 out. 2015.

CADÊ o Ônibus?. Disponível em: <http://www.cadeoonibus.com.br/CoO/SiteV2>. Acesso em: 27 out. 2015.

CHANDRAMOULI, Badrish. **Development and performance evaluation of multithreaded and thread-pool based web servers**. Duke, p. 1 – 2, mai. 2004.

CHANDRASEKARAN, K. **Essentials of cloud computing**. New York: Crc Press, 2015.

DEVELOPERS Android Studio Overview. Disponível em: <http://developer.android.com/tools/studio/index.html>. Acesso em: 06 out. 2015.

D-LINK Modem DWM-157. Disponível em: <http://www.dlink.com.br/produto/dwm-157>. Acesso em: 06 out. 2015.

EUROPEAN TELECOMUNICATIONS STANDARDS INSTITUTE (ETSI). Human Factors: AT Commands for Assistive Mobile Device Interfaces. **ETSI Technical Specification**, Sophia Antipolis, v. 1, n 102 511, 01 de ago. 2007. Disponível em: < http://www.etsi.org/deliver/etsi_ts/102500_102599/102511/01.01.01_60/ts_102511v010101p.pdf>. Acesso em: 06 out. 2015.

FIELDING, Roy T; GETTYS, Jim; MOGUL, Jeffrey; NIELSEN, Henrik Frystyk; BERNERS-LEE, Tim; MASINTER Larry; LEACH Paul. **Hypertext Transfer Protocol -- HTTP/1.1**. Internet RFC 2616, June 1999.

FLASK web development, one drop at a time. Disponível em: <http://flask.pocoo.org/>. Acesso em: 27 out. 2015.

GOOGLE Maps. Disponível em: <https://maps.google.com.br/>. Acesso em: 06 out. 2015.

GSM GPRS Wireless Modems. Disponível em: <http://www.developershome.com/sms/GSMModemIntro.asp>. Acesso em: 06 out. 2015.

INSTITUTO DE PESQUISA ECÔNOMICA APLICADA (IPEA). **Infraestrutura Social e Urbana no Brasil subsídios para uma agenda de pesquisa e formulação de políticas públicas: A mobilidade urbana no Brasil**. IPEA, 25 de mai. 2011. Série Eixos do Desenvolvimento Brasileiro, Comunicados do IPEA, Nº 94.

INTERNET Engineering Task Force: HPACK: Header Compression for HTTP/2. mai 2015. Disponível em: <https://tools.ietf.org/html/rfc7541> Acesso em: 27 out. 2015.

INTERNET Engineering Task Force: Hypertext Transfer Protocol Version 2 (HTTP/2). mai 2015. Disponível em: <https://tools.ietf.org/html/rfc7540>. Acesso em: 27 out. 2015.

INTERNET Society. Disponível em: <http://www.internetsociety.org/>. Acesso em: 27 out. 2015.

INTRODUCTION to AT Commands. Disponível em: <http://www.developershome.com/sms/atCommandsIntro.asp>. Acesso em: 06 out. 2015.

KUROSE, James F. **Redes de computadores e a internet**. 5.ed. São Paulo: Pearson Education do Brasil, 2010.

MANUAL TK103-2 Rastreador Imobilizador Monitoramento GPS/GSM/GPRS. Disponível em: <http://gpsbo.com.br/TK103-2_Portugues_user_manual.pdf>. Acesso em: 06 out. 2015.

MAPS for Work: API do Google Maps for Work. Disponível em: <https://www.google.com/work/mapsearth/products/mapsapi.html>. Acesso em: 06 out. 2015.

MERCADO Telecom Do 1G ao 4G: Infográfico com a evolução das redes móveis. Disponível em: <http://mobilexpert.com.br/mercado-telecom/materias/5432/do-1g-ao-4g-infografico-com-a-evolucao-das-redes-moveis>. Acesso em: 06 out. 2015.

MESSIAS, Valter Rogérios. **Servidor *web* distribuído com diferenciação de serviços – implementação e avaliação de um protótipo**. 2007. Dissertação (Mestrado em Ciência da Computação e Matemática Computacional) – Institudo de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos.

MONICO, João Francisco Galera. **Posicionamento pelo GNSS: descrição, fundamentos e aplicações**. 2 ed. São Paulo: Editora Unesp, 2008.

MORIMOTO, Carlos E. **Smartphones: Guia Prático**. 1 ed. São Paulo: GDH Press e Sul Editores, 2009.

O'BRIEN, James A. **Sistemas de informação: e as decisões gerenciais na era da Internet**. 3. ed. São Paulo: Saraiva, 2011.

OLHOVIVO Sistema de monitoramento de transporte. Disponível em: <http://olhovivo.sptrans.com.br>. Acesso em: 27 out. 2015.

PIROTTI, Rodolfo, P.; ZUCCOLOTTO, Marcos. Transmissão de dados através da telefonia celular: arquitetura das redes GSM e GPRS. **Revista Liberato**. v.10. n.13. p.81-89. Novo Hamburgo. jun. 2009.

PORTILLO, Daniel. Automated Vehicle Location using Global Positioning Systems for First Responders. **IITA Technical Report Series**, Colorado Springs, fev. 2008. Disponível em: < http://www.usafa.edu/df/iita/technical_reports1.cfm>. Acesso em: 06 out. 2015.

PYCHARM: The Most Intelligent Python IDE. Disponível em: <https://www.jetbrains.com/pycharm/>. Acesso em: 27 out. 2015.

SHEHRI, Waleed Al. **Cloud database database as a service**. Sydney, v.5, n.2, p. 1 - 4, abr. 2015.

SPACE Segment: Constellation Arrangement. Disponível em: <http://www.gps.gov/systems/gps/space/>. Acesso em: 06 out. 2015.

SPACE Segment: Constellation Arrangement. Disponível em: <http://www.gps.gov/systems/gps/space/>. Acesso em: 06 out. 2015.

STAIRS, Ralph M; REYNOLDS, Georg W. **Princípios de sistemas de Informação, uma abordagem gerencial**. 9. ed. São Paulo: Cengage, 2010.

SVERZUT, José Umberto. **Redes GSM, GPRS, EDGE e UMTS: Evolução a Caminho da Quarta Geração**. 3 ed. São Paulo: Erica, 2013.

TANENBAUM, Andrew S; David J. Wetherall. **Computer Networks**. 5.ed. Boston: Pearson Education, 2011.

TATEOKI, Getúlio Teruo. **Monitoramento de dados via internet baseado em telefonia celular**, 2007. 123 f. Dissertação (Mestrado) – Universidade Estadual Paulista. Faculdade de Engenharia de Ilha Solteira.

THE Internet Engineering Task Force. Disponível em: <https://www.ietf.org/>. Acesso em: 27 out. 2015.

THE Python NDB Datastore API. Disponível em: <https://cloud.google.com/appengine/docs/python/ndb/>. Acesso em: 27 out. 2015.

TRANSMITTING Network Data Using Volley. Disponível em: <http://developer.android.com/intl/pt-br/training/volley/index.html>. Acesso em: 27 out. 2015.

TSOU, Ming-Hsiang. Integrated Mobile GIS and Wireless Internet Map Server for Environmental Monitoring and Managemant. **Cartography and Geographic Information Science**, Mt. Pleasant, v. 31, n. 3, p. 153-165, mar. 2004.

URMET Daruma Modem MIN210. Disponível em: <http://www.daruma.com.br/pdfs_catgs/Urmet_Daruma_MIN210_3a.pdf>. Acesso em: 06 out. 2015.

VÁDEÔNIBUS Rio de Janeiro. Disponível em: <http://www.vadeonibus.com.br/Vdo/index.php>. Acesso em: 27 out. 2015.

WAGNER, Marcel Stefan. **Influência de protocolos de segurança sobre o desempenho de redes UMTS**, 2009. 141 f. Dissertação (Mestrado) – Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Sistemas Eletrônicos.