

Fabio Sussumu Komori
Rodrigo de Oliveira Morelli
Tiago Bello Torres
Tiago Matos

*XoTransito: Sistema de Geração de Rotas
com Base em Informações do Trânsito*

São Paulo

2008

**Fabio Sussumu Komori
Rodrigo de Oliveira Morelli
Tiago Bello Torres
Tiago Matos**

***XoTransito: Sistema de Geração de Rotas
com Base em Informações do Trânsito***

Projeto de Formatura apresentado à
disciplina PCS2502 - Projeto de Formatura
II, da Escola Politécnica da Universidade de
São Paulo

Área de Concentração:
Engenharia de Computação

Orientador:
Prof. Dr. João José Neto

São Paulo

2008

FICHA CATALOGRÁFICA

Komori, Fabio Sussumu; Morelli, Rodrigo de Oliveira; Matos, Tiago; Torres, Tiago Bello.

XoTransito: Sistema de Geração de Rotas com Base em Informações do Trânsito - São Paulo, 2008.

112 p.

Trabalho de Formatura - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Computação e Sistemas Digitais.

1. Cálculo de rotas 2. Tecnologia adaptativa 3. Análise de trânsito I. Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Computação e Sistemas Digitais II. Título.

Resumo

Neste trabalho, é apresentado o XoTRANSITO, um sistema de planejamento de rotas viárias que leva em consideração o trânsito local e outros fatores relevantes sobre as condições das vias. Para gerar as rotas, é utilizada uma modificação do algoritmo de busca informada A*. Entre os conceitos explorados, encontram-se a execução bidirecional, a utilização de técnicas adaptativas e a consideração da variação das condições de trânsito durante o percurso da rota. A tecnologia adaptativa, que tem como idéia central a capacidade de automodificação de um conjunto de regras, se manifesta no projeto em dois conceitos: a estratificação do mapa e a temporização do trajeto.

O projeto foi desenvolvido no Laboratório de Linguagens e Técnicas Adaptativas sob a orientação do Prof. Dr. João José Neto.

Abstract

The purpose of this document is to describe XOTRANSITO, a route planning system that takes into account the local traffic and other relevant road conditions. The route generation algorithm used in the project is a modification of the A* search algorithm. The following concepts have been proposed to enhance the original algorithm: bidirectional execution, adaptive techniques and the analysis of the variations in traffic during the course. The adaptive technology, which focuses on the ability of a set of rules to modify itself, is the basis of two concepts employed in the project: map layering and course temporization.

The project was developed in the Laboratório de Linguagens e Técnicas Adaptativas, under the supervision of Prof. Dr. João José Neto.

Lista de Abreviaturas e Siglas

AJAX	<i>Asynchronous Javascript and XML</i>
API	<i>Application Programming Interface</i>
CET	Companhia de Engenharia de Tráfego
CSS	<i>Cascading Style Sheets</i>
EAP	Estrutura Analítica de Projeto
EAR	Estrutura Analítica de Riscos
GPS	<i>Global Positioning System</i>
HTML	<i>HyperText Markup Language</i>
Java EE	<i>Java Enterprise Edition</i>
Java SE	<i>Java Standard Edition</i>
JAXB	<i>Java Architecture for XML Binding</i>
JAX-RS	<i>Java API for RESTful Web Services</i>
JPA	<i>Java Persistence API</i>
LTA	Laboratório de Linguagens e Técnicas Adaptativas
MVC	<i>Model-view-controller</i>
OSM	OpenStreetMap
PMBOK	<i>Project Management Body of Knowledge</i>
REST	<i>Representational State Transfer</i>
RUP	<i>Rational Unified Process</i>
SGDB	Sistema de Gerenciamento de Banco de Dados
SOAP	<i>Simple Object Access Protocol</i>
SQL	<i>Structured Query Language</i>
SWOT	<i>Strengths, Weaknesses, Opportunities and Threats</i>
URL	<i>Uniform Resource Locator</i>
XML	<i>Extensible Markup Language</i>

Sumário

1	Introdução	p. 10
2	Aspectos Conceituais	p. 11
2.1	Algoritmos de Roteamento Clássicos	p. 11
2.1.1	Algoritmo A*	p. 13
2.1.2	Algoritmo A* Bidirecional	p. 15
2.2	Tecnologia Adaptativa	p. 16
2.2.1	Estratificação do Mapa	p. 16
2.2.3	Adição e Remoção de Arestas	p. 19
2.3	Aplicações de Geometria Analítica	p. 19
2.3.1	Cálculo da Distância entre Dois Pontos	p. 20
2.3.2	Determinação do Ponto Mais Próximo	p. 22
3	Metodologia de Desenvolvimento	p. 24
3.1	Pesquisa bibliográfica	p. 24
3.2	Documentação	p. 24
3.3	Projeto	p. 26
3.4	Validação do projeto	p. 26
4	Especificações do Projeto	p. 28
4.1	Requisitos Funcionais	p. 28
4.2	Requisitos Não-Funcionais	p. 29
4.3	Casos de Uso	p. 29

4.3.1	Caso de uso 1: Consultar Trânsito	p. 29
4.3.2	Caso de uso 2: Encontrar Melhor Caminho	p. 30
4.3.3	Caso de uso 3: Informar Trânsito	p. 31
4.3.4	Caso de uso 4: Gerenciar Cadastro	p. 31
4.4	Diagrama de Classes	p. 32
4.5	Diagrama de Implantação	p. 33
4.6	Diagrama de Atividades	p. 34
4.7	Diagramas de Seqüência	p. 34
4.8	Gerenciamento do Projeto	p. 36
4.8.1	Estrutura Analítica do Projeto	p. 36
4.8.2	Cronograma	p. 37
4.8.3	Custos	p. 38
4.8.4	Estrutura Analítica de Risco	p. 39
5	Projeto e Implementação	p. 44
5.1	Arquitetura Geral	p. 44
5.1.1	Requisitos para Implantação	p. 45
5.1.2	Resumo das Tecnologias Java e JavaScript Utilizadas	p. 46
5.2	Mapas de Ruas e suas APIs de Visualização	p. 48
5.2.1	OpenStreetMap	p. 48
5.2.2	APIs de Visualização	p. 48
5.3	LibOsm	p. 52
5.3.1	Explicação das Classes e do Funcionamento do Módulo	p. 52
5.4	MapLink	p. 54
5.5	SimCet	p. 56
5.5.1	Base de Dados e Perfis	p. 56
5.5.2	Processador do SimCet	p. 57

5.5.3	WebServices do SimCet	p. 59
5.6	XoTransito	p. 60
5.6.1	Classe Grafo	p. 60
5.6.2	Classe AEstrela	p. 61
5.6.3	WebServices do XoTransito	p. 64
5.6.4	Peso das Arestas	p. 64
6	Testes e Avaliação	p. 66
6.1	Algoritmos	p. 66
6.1.1	Demonstração do Algoritmo	p. 66
6.1.2	Incorporação do Trânsito	p. 75
6.1.3	Primeiro Conjunto de Testes	p. 77
6.1.4	Segundo Conjunto de Testes	p. 85
6.1.5	Análise dos resultados obtidos	p. 94
7	Considerações Finais	p. 95
Anexo A – Diagramas Engenharia de Software		p. 97
Anexo B – Testes Realizados		p. 104
B.1	Teste 1	p. 105
B.1.1	Resultados dos Algoritmos	p. 105
B.2	Teste 2	p. 108
B.2.1	Resultados dos Algoritmos	p. 108
Referências		p. 111

1 *Introdução*

O sistema XOTRANSITO foi proposto e desenvolvido pelos autores como projeto de formatura do curso de Engenharia Elétrica com Ênfase em Computação no ano de 2008. A proposta do trabalho foi criar um sistema que planejasse uma rota entre duas localizações da cidade de São Paulo levando em conta variáveis importantes, como o trânsito e a velocidade permitida nas vias. Esses fatores são considerados durante a execução do algoritmo de roteamento do sistema.

O algoritmo central utilizado no trabalho para a geração de rotas foi uma variação do clássico algoritmo A*, descrito em Russell e Norvig (2003). Foram propostas alterações no algoritmo de roteamento original para torná-lo adaptativo e aproveitar as vantagens que a tecnologia adaptativa oferece para aperfeiçoar a geração de rotas de trânsito, permitindo tratar situações diversas, como congestionamentos e velocidade de vias.

Foi implementada também uma estratégia de estratificação do mapa utilizado no sistema, com a finalidade de criar um algoritmo hierárquico de roteamento, o que permite resolver o problema de forma estruturada, partindo do contexto geral e progredindo para o particular. Na estratificação, as vias da cidade são classificadas em alguns níveis hierárquicos. Essa técnica foi utilizada visando também economia de recursos computacionais, pois, ao classificar as vias do mapa em diversas camadas, apenas as vias de determinada camada (ou nível) são utilizadas pelo algoritmo a cada momento.

Primeiramente, são apresentados os fundamentos teóricos que permeiam o projeto, sendo eles o algoritmo A*, utilizado como base para o algoritmo de roteamento, e as técnicas adaptativas. Após isso, é feita uma descrição do funcionamento do sistema como um todo e de sua arquitetura.

2 Aspectos Conceituais

Neste capítulo, são descritos os fundamentos teóricos que permeiam o desenvolvimento do projeto. Numa primeira seção, são descritos os algoritmos de roteamento clássicos, incluindo o algoritmo de busca A*. Em seguida, é feita uma apresentação sobre a tecnologia adaptativa e os conceitos adaptativos explorados no projeto. Finalmente, é descrito como é feito o cálculo da distância entre dois pontos da cidade (dado que esses pontos são dados em latitude e longitude).

2.1 Algoritmos de Roteamento Clássicos

No projeto, foram analisados diversos algoritmos de roteamento clássicos para serem utilizados como estrutura central do algoritmo adaptativo. Alguns algoritmos pesquisados foram: busca em largura (*breadth-first search*), busca em profundidade (*depth-first search*), busca de custo uniforme, *greedy best-first search* e A* (que também é um algoritmo do tipo *best-first search*). Os algoritmos mencionados estão descritos em Knuth (1997) e Russell e Norvig (2003).

O algoritmo de **busca em largura** começa expandindo o nó raiz e segue expandindo todos os seus sucessores. A seguir, para cada um dos sucessores, expande os sucessores deles. O procedimento é repetido até que se encontre o nó de destino. Ou seja, todos os nós do nível anterior são expandidos antes que algum nó do nível atual seja expandido. Apesar de o algoritmo ser completo, ele é ótimo apenas para grafos sem pesos nas arestas. Além disso, o algoritmo possui complexidade espacial e temporal de $O(b^d)$, em que b é o fator de ramificação do grafo e d é a profundidade da solução. O custo exponencial em termos de espaço e tempo é uma razão forte para a não-adoção da busca em largura no projeto.

O algoritmo de **busca em profundidade** começa explorando o nó raiz e explora cada ramo até o fim antes de realizar retrocesso. Apesar de o algoritmo possuir complexidade

espacial de apenas $O(m)$ (em que m é a profundidade máxima do grafo), sua complexidade temporal também é $O(b^d)$, o que inviabiliza seu uso no projeto.

A **busca de custo uniforme** consiste em uma extensão da busca em largura, conseguindo lidar com grafos ponderados. Durante o processamento, o algoritmo calcula o custo de cada nó, isso é, o somatório dos pesos das arestas do caminho que parte da origem e chega ao nó em questão. Em cada etapa, o algoritmo expande o nó cujo custo seja o menor. Se os pesos das arestas forem positivos, o algoritmo é completo e ótimo. A complexidade espacial e temporal do algoritmo no pior caso é $O(b^{1+\lfloor C^*/\epsilon \rfloor})$, em que C^* é o custo da solução ótima e ϵ é o custo mínimo de cada ação (menor peso das arestas). Observa-se, portanto, que a complexidade pode ser bem maior do que $O(b^d)$, tornando a busca de custo uniforme uma alternativa custosa.

Uma solução para o problema é fornecida por algoritmos de busca informada, os quais usam algum tipo de heurística para guiar a expansão de nós. O algoritmo de busca informada mais simples é o ***greedy best-first search***, que considera apenas um fator na avaliação de um nó: a estimativa do custo do caminho que parte do nó atual e chega ao destino. A cada etapa, o nó de menor estimativa é expandido. O custo estimado é fornecido por uma heurística, que poderia ser a distância em linha reta entre o nó atual e o nó de destino. O algoritmo *greedy best-first search* possui complexidade temporal de $O(b^m)$, em que m é a profundidade máxima do espaço de busca. Um fato importante a ser considerado é que a complexidade pode ser bastante reduzida com a escolha de uma boa heurística.

O algoritmo mais interessante para os propósitos do projeto, entretanto é o **A***, que foi originalmente descrito em Hart, Nilsson e Raphael (1968). O algoritmo A* é completo (sempre encontra uma solução caso exista) e ótimo (encontra a solução com o menor custo possível), desde que a heurística utilizada seja admissível. Além disso, é otimamente eficiente, ou seja, nenhum outro algoritmo ótimo garante visitar um menor número de nós do que A*. Por ser o tipo de busca adotado no projeto, o algoritmo será descrito com mais detalhes a seguir.

Deve-se enfatizar, entretanto, que a aplicação do algoritmo A* em sua forma original não é ideal para este projeto, pois o desempenho é prejudicado para rotas entre pontos muito distantes. Além disso, não é um requisito essencial que a busca encontre o caminho ótimo, já que o fator trânsito, variável e algumas vezes imprevisível, é também considerado.

2.1.1 Algoritmo A*

O algoritmo-base de estabelecimento de rotas utilizado é o A*. Essencialmente, o algoritmo calcula a rota ótima entre dois vértices de um grafo ponderado. Entende-se por rota ótima o caminho cujo somatório de pesos seja mínimo.

Durante o processamento, o algoritmo mantém duas listas de vértices: a **lista fechada**, composta de vértices para os quais já foram encontrados caminhos a partir do vértice de origem, e a **lista aberta**, formada pelos vértices diretamente conectados aos vértices da lista fechada.

A lista aberta do algoritmo é programada como **fila de prioridades**, na qual os elementos são sempre mantidos ordenados segundo algum critério. Para o algoritmo A*, o critério adotado é a função $f(n)$, que estima o custo do caminho que parte da origem e chega ao destino, passando pelo vértice n . Matematicamente, $f(n)$ é calculado da seguinte forma:

$$f(n) = g(n) + h(n)$$

Em que:

- $g(n)$ é o custo acumulado do caminho que parte da origem e chega ao vértice n ;
- $h(n)$ é uma estimativa do custo do caminho entre o vértice n e o vértice de destino.

Desse modo, o algoritmo A* considera não só o custo para atingir o vértice atual, mas também o custo estimado para atingir o destino. De fato, o algoritmo começa analisando os primeiros vértices da fila, ou seja, aqueles que possuem menor valor de $f(n)$ e, portanto, maior probabilidade de conduzir ao vértice de destino através de um caminho de baixo custo.

Uma das vantagens do algoritmo A* é a utilização da heurística $h(n)$, que guia a busca ao vértice de destino, calculando o caminho mais rapidamente do que os algoritmos de força bruta. Dado um vértice qualquer n , a heurística $h(n)$ deve estimar rapidamente o custo entre esse vértice e o vértice de destino. Uma primeira implementação para a heurística é feita com o cálculo da distância em linha reta entre o vértice n e o vértice de destino. Diferentes heurísticas $h(n)$ levam a comportamentos diferentes do algoritmo A*:

- Se $h(n)$ for igual a zero, então todo o trabalho de busca é feito com base na função

$g(n)$. Nesse caso, o algoritmo A* se reduz ao algoritmo de custo uniforme, que garante encontrar o melhor caminho, mas tem alta complexidade computacional.

- Se $h(n)$ for menor ou igual ao custo real do caminho entre o vértice n e o vértice de destino, então é garantido que o algoritmo A* encontre o melhor caminho caso ele exista. Nesse caso, quanto menor for o valor de $h(n)$, mais vértices precisarão ser expandidos, tornando a busca mais lenta.
- Se $h(n)$ for exatamente igual ao custo real do caminho entre o vértice n e o vértice de destino, então o algoritmo A* nunca expandirá vértices que estejam fora do caminho ótimo. Essa situação equivale ao melhor desempenho possível do algoritmo. Entretanto, uma heurística com essas características é dificilmente encontrada na prática. De forma simplificada, se o algoritmo possuir informações perfeitas, ele terá um comportamento ideal.
- Se $h(n)$ for maior do que o custo real do caminho entre o vértice n e o vértice de destino, então não é garantido que o algoritmo encontre o melhor caminho. Essa situação é, portanto, indesejável.
- Se $h(n)$ for muito alto em relação a $g(n)$, então todo o trabalho de busca será feito com base na função $h(n)$. Nesse caso, o algoritmo A* se reduz ao algoritmo *greedy best-first search*.

Cada vez que o algoritmo executa o laço principal, o primeiro vértice da lista aberta (fila de prioridades) é movido para a lista fechada. Esse vértice será chamado de vértice atual. Em seguida, todos os vizinhos do vértice atual são buscados e adicionados à lista fechada (caso já não estejam lá). Cada um desses vértices vizinhos recebe como antecessor o vértice atual. Além disso, cada um deles tem seu custo estabelecido como o custo para atingir o vértice atual mais o custo da aresta que liga o vértice atual ao vértice vizinho em questão.

Se uma heurística imprecisa for utilizada, é possível que um vértice seja fechado mesmo que tenha um custo superior ao custo do menor caminho até ele. Nessa situação, o algoritmo A* ainda é capaz de encontrar o melhor caminho para o nó em questão. Se o vértice atual estiver conectado a um vértice que já está fechado e se o custo do vértice atual somado ao custo da aresta em questão for menor do que o custo registrado anteriormente para esse vértice, então o algoritmo remove esse vértice imperfeito da lista fechada e o adiciona novamente à lista aberta, atualizando seu custo e vértice anterior. Ou seja, o

algoritmo A* é capaz de se recuperar de uma função heurística imperfeita. Vale notar que esse procedimento pode fazer com que um mesmo vértice seja adicionado diversas vezes à lista aberta (a partir de antecessores diferentes). Então, o algoritmo deve sempre verificar se o primeiro vértice da lista aberta (vértice atual) já se encontra na lista fechada antes de adicioná-lo. Se o vértice já estiver lá, então já foi encontrado para ele um custo menor, de forma que o algoritmo pode prosseguir para a próxima iteração sem tentar fechar esse vértice novamente.

2.1.2 Algoritmo A* Bidirecional

Para melhorar o desempenho do algoritmo A* clássico, decidiu-se implementar o algoritmo A* com execução bidirecional (algoritmo de heurística simétrica descrito em Pijlsy e Postz (2006)). A busca bidirecional é basicamente uma aplicação simultânea de duas instâncias do algoritmo A*: uma que parte no estado inicial e outra que parte do estado final. O algoritmo bidirecional é finalizando quando ambas as instâncias se encontram em algum ponto intermédio, como é mostrado na figura 1.

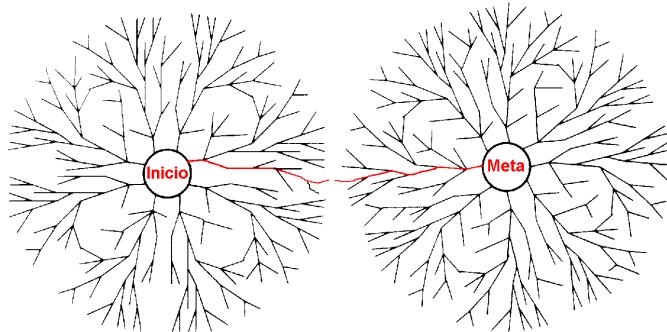


Figura 1: Esquema geral de uma busca bidirecional.

Se a complexidade de uma busca comum (unidirecional) for $O(b^d)$, então cada instância da busca bidirecional percorre apenas metade da profundidade, tendo complexidade de $O(b^{d/2})$. A complexidade das duas instâncias da busca bidirecional é, portanto, $O(b^{d/2} + b^{d/2})$, sendo bem menor do que a complexidade da busca comum.

Outra forma de visualizar o mesmo fato é notar que a busca original expande uma área equivalente a um círculo de raio d ($A = \pi d^2$), ao passo que a busca bidirecional expande uma área equivalente a dois círculos de raio $d/2$ ($A = 2\pi (\frac{d}{2})^2 = \frac{\pi d^2}{2}$).

No caso do problema de geração de rotas, o algoritmo A* seria usado no início do trajeto (tentando chegar ao ponto final) e no final do trajeto (tentando chegar ao início),

o que poderia ocorrer em duas *threads* separadas, compartilhando uma mesma lista de nós. A partir do momento em que uma *thread* ou uma instância do algoritmo encontra um nó já presente na lista fechada da outra instância, tem-se uma solução.

2.2 Tecnologia Adaptativa

A adaptatividade é um conceito que explora a flexibilidade dos algoritmos por meio da automodificação de suas regras de funcionamento. Desse modo, o algoritmo adaptativo pode se comportar de maneira diferente a cada passo da execução, pois suas regras são alteradas em tempo de execução. A tecnologia adaptativa possui diversas aplicações, entre as quais se destacam as áreas de linguagens de programação, processamento de linguagens naturais, computação natural e inteligência artificial. Uma explicação sobre os conceitos básicos de adaptatividade encontra-se em Neto (2004).

Um algoritmo adaptativo se baseia num conjunto de regras, que ditam o seu comportamento. Porém, para cada passo da execução, esse conjunto de regras pode ser modificado, sofrendo inclusões, alterações e exclusões de regras. No caso deste projeto, o conjunto de regras é o mapa de ruas da cidade, que pode sofrer modificações no decorrer da execução do sistema, tais como: remoções temporárias de arestas com muito trânsito e considerações de um subconjunto de arestas num determinado momento da execução, reduzindo o número de possibilidades de busca, além de atualizar dinamicamente os pesos referentes ao trânsito de acordo com a estimativa de tempo da viagem.

No projeto XoTRANSITO, tais idéias resultaram em três conceitos: estratificação do mapa (consideração de diferentes níveis de detalhamento de ruas, com o intuito principal de reduzir o espaço de busca), temporização do trânsito (atualização dinâmica, durante o processamento do algoritmo, do trânsito de acordo com estimativas de tempo e dados históricos registrados) e adição e remoção das arestas, em caso de situações muito ruins de trânsito. Tais idéias são explicadas nas seções subsequentes.

2.2.1 Estratificação do Mapa

Sabendo-se que o mapa da cidade de São Paulo possui um número bastante grande de vias, torna-se custoso o processamento de todas as ruas existentes durante o cálculo das rotas. Além disso, há situações em que a análise de ruas de pequeno porte é simplesmente desnecessária. Por exemplo, no cálculo de uma rota longa, espera-se que grande parte do caminho seja percorrida apenas em grandes avenidas, sendo as ruas menores utilizadas

apenas em pontos próximos da origem e do destino, locais em que uma grande avenida pode não estar diretamente conectada.

Pelo motivo exposto acima, um dos requisitos introduzidos com o objetivo de incorporar os conceitos de adaptatividade é a classificação do mapa em níveis, cada um contendo ruas de portes distintos (estratificação). Quanto maior o nível da observação, menor é o detalhamento do mapa (mapa com menor quantidade de ruas visíveis pelo algoritmo de roteamento do projeto). Inicialmente, dois níveis serão considerados: completo (contendo todas as ruas) e macroscópico (contendo somente as avenidas principais). Deve-se observar, portanto, que a estratificação é uma estratégia que visa incorporar a adaptatividade de forma natural ao projeto.

Considerando a abordagem de estratificação e a execução bidirecional do algoritmo, é possível realizar uma implementação em que o mapa completo é analisado apenas em pontos próximos da origem e do destino, ou seja, no início de cada instância do algoritmo. Ao encontrar uma avenida ou via de nível mais alto, cada instância muda seu nível de observação do grafo, de forma que o algoritmo limita-se a buscar vias desse nível ou superior. A utilização da abordagem estratificada permite uma maior agilidade no processamento do algoritmo, uma vez que não verifica todos os nós possíveis no meio do caminho.

Além disso, o esquema de hierarquia de mapa pode ser expandido para mais de dois níveis, o que mostra a flexibilidade do conceito exposto. Por exemplo, um esquema de três níveis poderia ser construído da seguinte forma: o nível mais alto (nível 3) seria formado pelas vias principais, como as marginais; o nível intermediário (nível 2) incluiria, além das vias de nível 3, as ruas e avenidas de velocidade intermediária, como a Av. Nove de Julho; o nível mais baixo (nível 1) incluiria, além das vias de nível 2, as ruas de baixa velocidade. Novamente, cada nível representa um detalhamento diferente do mapa, estruturando-o em camadas. A alteração de níveis se faz da mesma forma que no caso de dois níveis, sendo possível generalizar a visão do grafo em duas etapas. Resumindo, o algoritmo se utiliza do autômato dado na figura 2 para trocar os níveis da estratificação.

No contexto apresentado, o grafo de ruas pode ser visto como um conjunto de regras: para cada par de vértices, a regra define se existe ou não uma aresta entre eles. Como esse conjunto de regras é alterado dinamicamente, pode-se classificar o procedimento como adaptativo.



Figura 2: Autômato de mudança de nível de ruas. A mudança ocorre sempre que o algoritmo acha em seu processamento uma via de um nível superior.

2.2.1.1 Exemplo de Estratificação

Considera-se uma rota entre o Tatuapé e o Butantã, um caminho relativamente longo, que passa por diversas vias rápidas, como a Marginal Tietê, e avenidas grandes, como a Salim Farah Maluf. Essa rota, caso fosse gerada com o algoritmo A* adaptativo com estratificação e bidirecionalidade, consideraria, na origem e no destino, todas as possibilidades (ruas, avenidas e vias principais). Ao chegar a uma via de nível superior, por exemplo, uma avenida, ele passaria a considerar somente as avenidas e vias principais.

2.2.2 Temporização do Trânsito

Numa abordagem mais precisa, deve-se prever que o caminho gerado a partir da situação de trânsito atual pode não ser bom o suficiente, já que o trânsito é dinâmico e, enquanto o motorista percorre o caminho, a situação do tráfego se altera. Esse tipo de consideração é principalmente útil para caminhos longos ou com trânsito excessivo, nos quais o tempo de percurso torna-se relevante.

Desse modo, é necessário analisar o histórico do trânsito, de forma a se tentar prever como o mesmo estará no momento em que o motorista estiver percorrendo seu caminho. Com isso, pretende-se obter um caminho mais eficiente em cenários reais (dinâmicos). Para se aproximar da realidade, o algoritmo pode considerar uma base de dados estatística, que armazena um conjunto de situações de trânsito para as ruas monitoradas em diferentes horários do dia.

Uma técnica possível para essa abordagem é considerar que o trânsito é um peso que varia com o tempo. Basta, portanto, realizar uma estimativa do tempo de percurso

com base na velocidade da via e no trânsito local. Dispõe-se, a cada instante, dessa estimativa de tempo, pode-se calcular a condição de trânsito real a partir da análise de histórico. Com isso, os pesos utilizados para representar o trânsito passam a ter um significado bastante mais realístico.

Sabendo que os pesos das arestas do grafo (que representam, entre outras coisas, o trânsito) são vistos como regras para o algoritmo, e que essas regras são alteradas pelo algoritmo durante sua execução, conclui-se que a estratégia de temporização também inclui conceitos adaptativos.

2.2.2.1 Exemplo de Temporização do Trânsito

Considera-se uma rota longa, de 25 km de extensão, ligando dois pontos distantes da cidade. É de se supor, que, em São Paulo, essa rota seja percorrida em cerca de 1 hora em horários de trânsito ruim. Novamente, é fácil notar que o perfil de trânsito irá variar diversas vezes no decorrer do trajeto. Caso o caminho seja traçado com o algoritmo A* adaptativo por temporização, o tamanho da rota seria percebido como longo e demorado, e o algoritmo calcularia a rota levando em consideração as previsões de trânsito.

2.2.3 Adição e Remoção de Arestas

Além dos métodos apresentados, o grafo ponderado pode ser submetido a operações de inclusões e exclusões de arestas. Tais ações devem ser realizadas, respectivamente, em casos de vias muito congestionadas e no momento em que essas passam a ter um trânsito melhor. A adição e a remoção de arestas do grafo também podem ser vistas como ações adaptativas, já que o grafo consiste em um conjunto de regras.

2.3 Aplicações de Geometria Analítica

Nesta seção, são explicadas duas aplicações da geometria analítica neste projeto: o cálculo da distância entre dois pontos, dadas suas coordenadas geográficas (latitude e longitude), importante para a determinação de pesos baseados no comprimento de segmentos de reta e a determinação do ponto mais próximo dentre um conjunto de segmentos de reta, utilizado no momento em que o usuário clica no mapa de ruas.

2.3.1 Cálculo da Distância entre Dois Pontos

Como foi explicado, uma implementação simples para a heurística $h(n)$ é considerar o caminho em linha reta entre dois pontos. Entretanto, são conhecidas apenas as coordenadas geográficas dos vértices (latitude e longitude), o que significa que algum cálculo deve ser executado para que a distância efetiva entre dois vértices seja obtida (em metros, por exemplo).

O cálculo executado assume que a Terra é uma esfera perfeita, aproximação considerada satisfatória para os objetivos do projeto ($R_{Terra} = 6.378,137\text{km}$).

Deseja-se calcular, portanto, a distância entre o ponto A , de coordenadas $(\text{lat}_A, \text{long}_A)$ e o ponto B , de coordenadas $(\text{lat}_B, \text{long}_B)$. Os valores de latitude e longitude devem ser primeiramente convertidos de graus para radianos a fim de facilitar os cálculos. A rigor, as coordenadas em questão são esféricas:

$$r = R_{Terra} \quad (2.1)$$

$$\theta = \frac{\pi}{2} - \text{lat} \quad (2.2)$$

$$\varphi = \text{long} \quad (2.3)$$

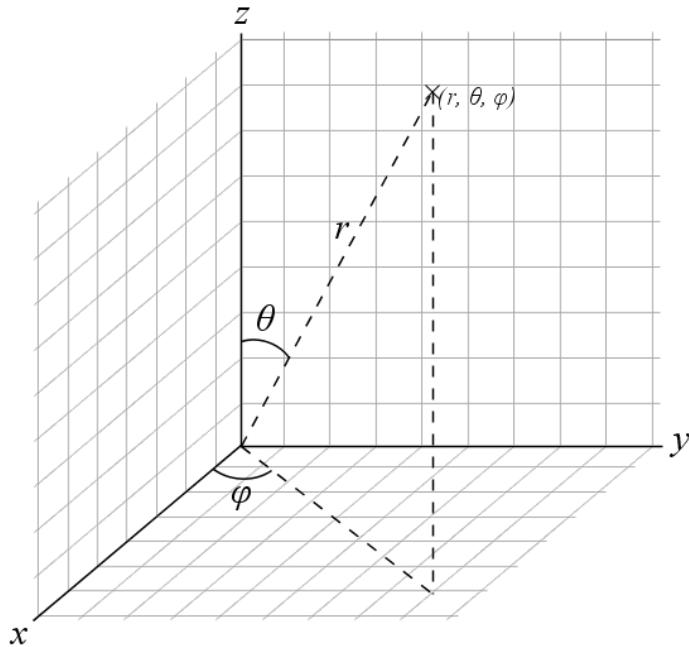


Figura 3: Exemplo de um ponto representado em coordenadas esféricas (r, θ, φ) .

É possível converter as coordenadas esféricas da figura 3 para coordenadas cartesianas com as seguintes fórmulas:

$$x = r \sin \theta \cos \varphi = r \sin \left(\frac{\pi}{2} - \text{lat} \right) \cos(\text{long}) = r \cos(\text{lat}) \cos(\text{long}) \quad (2.4)$$

$$y = r \sin \theta \sin \varphi = r \sin \left(\frac{\pi}{2} - \text{lat} \right) \sin(\text{long}) = r \cos(\text{lat}) \sin(\text{long}) \quad (2.5)$$

$$z = r \cos \theta = r \cos \left(\frac{\pi}{2} - \text{lat} \right) = r \sin(\text{lat}) \quad (2.6)$$

A partir desses cálculos, é possível considerar dois vetores: o vetor $\vec{A} = (x_A, y_A, z_A)$, partindo do centro da Terra e terminando no ponto A , e o vetor $\vec{B} = (x_B, y_B, z_B)$, partindo do centro da Terra e terminando no ponto B . O produto escalar entre os vetores pode ser calculado:

$$\vec{A} \cdot \vec{B} = x_A x_B + y_A y_B + z_A z_B \quad (2.7)$$

Usando a propriedade fundamental do produto escalar, é possível calcular o ângulo α entre os dois vetores:

$$|\vec{A}| |\vec{B}| \cos \alpha = x_A x_B + y_A y_B + z_A z_B \quad (2.8)$$

$$\cos \alpha = \frac{x_A x_B + y_A y_B + z_A z_B}{|\vec{A}| |\vec{B}|} \quad (2.9)$$

$$\cos \alpha = \frac{x_A x_B + y_A y_B + z_A z_B}{R_{\text{Terra}}^2} \quad (2.10)$$

$$\alpha = \cos^{-1} \left(\frac{x_A x_B + y_A y_B + z_A z_B}{R_{\text{Terra}}^2} \right) \quad (2.11)$$

O comprimento do arco determinado pelo ângulo α é justamente a distância entre os pontos A e B :

$$L = \alpha R_{\text{Terra}} \quad (2.12)$$

$$L = \cos^{-1} \left(\frac{x_A x_B + y_A y_B + z_A z_B}{R_{\text{Terra}}^2} \right) R_{\text{Terra}} \quad (2.13)$$

Portanto, pode-se utilizar as conversões apresentadas para calcular a distância L entre dois pontos quaisquer do mapa, o que é a base da heurística $h(n)$ do algoritmo A*.

2.3.2 Determinação do Ponto Mais Próximo

O problema a ser resolvido trata da necessidade de se determinar qual é o ponto existente num mapa de ruas mais próximo da seleção efetuada pelo usuário. Desse modo, um pseudocódigo em alto nível da solução implementada é mostrado a seguir:

- 1 Para cada ponto do mapa fornecido :
- 2 Calcula a distância entre os pontos atual e selecionado ;
- 3 Se a distância for menor do que o limite estabelecido :
- 4 Insere numa lista a rua pertencente ao ponto atual ;
- 5 Para cada rua da lista montada anteriormente :
- 6 Seleciona os dois pontos mais próximos da rua em relação ao ponto selecionado pelo usuário ;
- 7 Calcula a distância desse segmento de reta ao ponto selecionado pelo usuário ;
- 8 Retorna o segmento de reta que possui a menor distância ;

Os conceitos de geometria analítica apresentados a seguir são referentes à linha 7 do pseudocódigo acima. Portanto, a necessidade está baseada no cálculo da distância entre um ponto e um segmento de reta.

Na figura 4, considera-se que o usuário tenha clicado no ponto P e que esteja-se analisando a proximidade entre esse ponto e o segmento de reta (P_1, P_2) , que representa uma aresta do grafo de ruas.

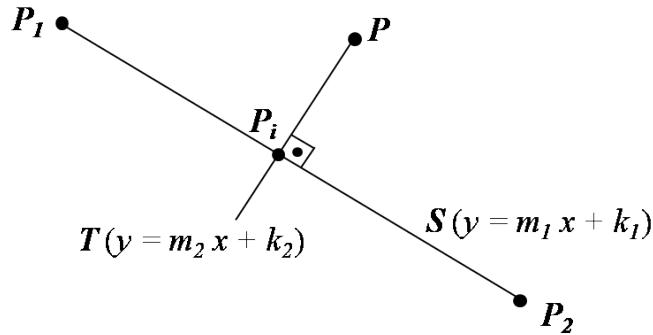


Figura 4: Cálculo da distância entre o ponto P (ponto selecionado pelo usuário) e o segmento de reta (P_1, P_2) , que representa uma aresta no grafo de ruas.

Dados o ponto $P = (x_P, y_P)$ e o segmento de reta definido pelos pontos $P_1 = (x_{P_1}, y_{P_1})$ e $P_2 = (x_{P_2}, y_{P_2})$, pode-se calcular a equação da reta S :

$$y = m_1 x + k_1 \quad (2.14)$$

$$m_1 = \frac{y_{P_2} - y_{P_1}}{x_{P_2} - x_{P_1}} \quad (2.15)$$

$$k_1 = y_{P_1} - m_1 x_{P_1} \quad (2.16)$$

Seja T uma reta perpendicular a S , passando pelo ponto P . O coeficiente angular da reta T vale $m_2 = -\frac{1}{m_1}$ e a equação de tal reta é dada por:

$$y = m_2 x + k_2 \quad (2.17)$$

$$m_2 = -\frac{1}{m_1} \quad (2.18)$$

$$k_2 = y_{P_2} - m_2 x_{P_2} \quad (2.19)$$

O ponto $P_i = (x_i, y_i)$, que representa a intersecção entre as duas retas, é calculado por:

$$P_i = \left(\frac{k_2 - k_1}{m_1 - m_2}, m_1 x_i + k_1 \right) \quad (2.20)$$

Com isso, é possível calcular a distância entre os pontos P e P_i através do método apresentado na seção 2.3.1. Finalmente, deve-se verificar se o ponto de intersecção se encontra dentro do segmento de reta S . Em caso negativo, deve-se calcular as distâncias entre o ponto P e cada uma das extremidades do segmento (pontos P_1 e P_2), escolhendo-se a menor entre essas duas distâncias.

Dados os conceitos teóricos apresentados, foi definida a metodologia de desenvolvimento a ser seguida, explicada na seção seguinte.

3 *Metodologia de Desenvolvimento*

Uma metodologia é um conjunto estruturado de práticas, que pode ser repetível durante o processo de produção de *software*. Nas próximas seções é detalhada toda a metodologia utilizada para o desenvolvimento do projeto.

3.1 Pesquisa bibliográfica

Buscou-se, nesse item, adquirir todo o conhecimento teórico necessário para a implementação do projeto. Para adquirir esse conhecimento, foram realizadas pesquisas sobre a tecnologia adaptativa, sendo consultadas, nesse item, as teses e dissertações publicadas pelo LTA (Laboratório de Linguagens e Técnicas Adaptativas) e material sobre os principais algoritmos de busca, sendo estudados os seguintes algoritmos: busca em largura, busca em profundidade, custo uniforme e o A*.

3.2 Documentação

Nesse item, foram elaborados os relatórios e apresentações intermediários requisitados pela disciplina, além da documentação de entrega final contendo a monografia, o press release e a página web do projeto. Na listagem a seguir são detalhados toda a documentação feita para o projeto.

- Relatórios do primeiro semestre

Elaboração de documento contendo título do projeto, objetivos, resumo e grupo de alunos

Esse documento apresentava a visão inicial do projeto que estava sendo planejado.

Documento entregue em 21 de março de 2008.

Apresentação da evolução da especificação do projeto e entrega de documento

Esse documento apresenta a evolução alcançada na especificação do projeto até a data de sua entrega.

Documento entregue em 20 de abril de 2008.

Apresentação da evolução da especificação do projeto e entrega de documento

Esse documento apresenta a evolução alcançada na especificação do projeto até a data de sua entrega.

Documento entregue em 18 de maio de 2008.

Reunião final (apresentação) e entrega da especificação do projeto de formatura

Esse documento apresentava a especificação do projeto de formatura que seria implementado no segundo semestre pelo grupo. Continha todos os itens necessários à implementação de um projeto: cronograma, custos, diagramas, metodologia de implementação, etc.

Documento entregue em 15 de junho de 2008.

• Relatórios do segundo semestre

Apresentação da evolução do projeto e entrega de documento.

Esse documento apresentava a evolução alcançada na implementação do projeto obtida até a data de sua entrega.

Documento entregue em 18 de agosto de 2008.

Apresentação da evolução do projeto e entrega de documento.

Esse documento apresentava a evolução alcançada na implementação do projeto obtida até a data de sua entrega.

Documento entregue em 4 de novembro de 2008.

Entrega do poster, *press release*, página Web e monografia final.

Nessa etapa, foram entregues o poster do projeto, o *press release*, indicado o local da página Web do projeto e elaborada a monografia final do trabalho.

Documentação entregue no mês de dezembro.

3.3 Projeto

Nesse item, foram realizadas todas as tarefas relacionadas ao planejamento do projeto, sendo que esse planejamento foi uma aproximação do que é proposto pela metodologia RUP (*Rational Unified Process*). Além disso, também foi realizado o gerenciamento do projeto, sendo especificado os itens: EAP, EAR e o cronograma do projeto.

Após a elaboração do escopo e gerenciamento do projeto, foi realizada a sua implementação e a validação de seu resultado.

A listagem a seguir mostra um resumo de toda a metodologia utilizada para a implementação do projeto, sendo que o detalhamento desta é feito nos capítulos 4 e 5.

- Projeto

Especificação dos requisitos dos sistema.

Elaboração dos casos de uso.

Elaboração dos diagramas de classe.

Elaboração do diagrama de implantação.

Elaboração do diagrama de atividades.

Elaboração do diagrama de seqüência.

Construção da EAP (Estrutura Analística de Projeto).

Construção da EAR (Estrutura Analítica de Risco).

Especificação do cronograma do projeto.

Cálculo do custo do projeto.

Elaboração da página Web do projeto.

Implementação do projeto

Realização de testes e validação dos resultados.

3.4 Validação do projeto

Para a validação do projeto foi planejado um conjunto de testes para analisar o desempenho do algoritmo proposto neste trabalho. O conjunto de testes foi dividido em duas frentes: uma que analisa o desempenho do algoritmo adaptativo proposto em comparação

com outros dois algoritmos clássicos em relação à distâncias curtas e outra que analisa a comparação do desempenho para distâncias longas. A análise detalhada do conjunto de testes é feita no capítulo 6, assim como a conclusão a respeito do desempenho do algoritmo proposto no trabalho.

4 Especificações do Projeto

Nesse capítulo, é relatada a especificação do projeto proposto. Numa primeira etapa, são descritos os requisitos funcionais e não funcionais do sistema, os casos de uso elaborados, os diagramas do sistema. Em seguida, são descritos os artefatos relacionados ao gerenciamento do projeto.

4.1 Requisitos Funcionais

Os requisitos funcionais especificam ações que um sistema deve ser capaz de executar, sem levar em consideração restrições físicas. Eles especificam, portanto, o comportamento de entrada e saída de um sistema.

- **F1** - O sistema deve processar os dados em XML do OpenStreetMap e convertê-los em um modelo geográfico de objetos;
- **F2** - O sistema deve converter o modelo geográfico em um modelo matemático, composto de um grafo em que as arestas são as vias e os vértices são os cruzamentos;
- **F3** - O sistema deve atribuir pesos às arestas do grafo, baseados em: informações de trânsito (provindas da CET e dos usuários) e comprimento da via;
- **F4** - O sistema deve traçar rotas no mapa utilizando técnicas adaptativas, alterando o comportamento do algoritmo A*;
- **F5** - O sistema deve receber os dados de origem e destino, através de uma interface gráfica ou por meio da digitação do nome da rua;
- **F6** - O sistema deve permitir o cadastro de usuários e seus locais de interesse, facilitando o uso da interface;
- **F7** - O sistema deve estratificar o mapa, dividindo-o em dois ou mais níveis de detalhamento;

- **F8** - O sistema deve gravar o histórico das informações de trânsito, para otimizar o algoritmo de roteamento;

4.2 Requisitos Não-Funcionais

Requisitos não-funcionais descrevem apenas atributos do sistema ou atributos do ambiente do sistema. São as qualidades globais de um *software*, como manutenibilidade, usabilidade, desempenho, custos e várias outras. Normalmente, esses requisitos são descritos de maneira informal, de maneira controversa (por exemplo, o gerente quer segurança, mas os usuários querem facilidade de uso) e são difíceis de validar.

- **NF1** - O sistema deve possuir uma interface gráfica disponível ao usuário pela Internet;
- **NF2** - O sistema deve ser acessível através de celulares;
- **NF3** - O sistema deve ser implementado usando as tecnologias Java nas versões SE (*Standard Edition*) e EE (*Enterprise Edition*)

4.3 Casos de Uso

Nas seções a seguir são descritos os casos de uso planejados para o *software* do projeto. Esses casos de uso descritos englobam o funcionamento básico que tinha sido previsto para a validação do projeto. Eles correspondem basicamente à ação de traçar uma rota entre dois pontos do mapa, informar o trânsito existente nas ruas da cidade, consultar o trânsito presente nas ruas e gerenciar o cadastro dos usuários do sistema.

4.3.1 Caso de uso 1: Consultar Trânsito

Descrição: Usuário consulta o mapa da cidade, no qual há indicações do grau de congestionamento de cada rua.

Autor: Usuário do sistema

Seqüência de eventos:

1. Usuário solicita a opção de consultar o mapa de congestionamento.

2. Sistema exibe o mapa completo da cidade e a legenda de cores correspondente aos graus de congestionamento.
3. Usuário solicita ampliação ou redução do mapa (*zoom*).
4. Sistema mostra o mapa da região escolhida com o nível de detalhamento adequado.

Observação: Os passos 3 e 4 são opcionais e podem ser repetidos quantas vezes o usuário desejar.

Pós-condição: O caso de uso se encerra quando o usuário não deseja mais repetir os passos 3 e 4.

4.3.2 Caso de uso 2: Encontrar Melhor Caminho

Descrição: Usuário requisita que o sistema trace a rota entre dois pontos da cidade.

Autor: Usuário do sistema

Seqüência de eventos:

1. Usuário informa os pontos de origem e destino, seja por indicação no mapa ou pelo fornecimento do endereço. No caso de indicação no mapa, deve-se seguir o caso de uso 1 antes.
2. Sistema localiza os pontos fornecidos e os exibe no mapa.
3. Sistema calcula a rota entre os dois pontos e a exibe graficamente (mapa) ou textualmente (lista de passos a seguir).

Exceções:

1. No passo 2, o sistema pode não encontrar os pontos solicitados. Nesse caso, uma mensagem de erro é exibida e o sistema volta ao passo 1.
2. No passo 3, o sistema pode não conseguir calcular a rota requisitada. Novamente, uma mensagem de erro é exibida e o sistema volta ao passo 1.

4.3.3 Caso de uso 3: Informar Trânsito

Descrição: Usuário ou CET informa ao sistema o grau de congestionamento de certa rua através de uma classificação.

Autor: Usuário do sistema ou serviço da CET

Seqüência de eventos:

1. Ator informa ao sistema o local do qual deseja informar o grau de congestionamento. O processo pode ser feito por indicação no mapa, fornecimento de endereço ou fornecimento de coordenadas (por exemplo, de um GPS). No caso de indicação no mapa, deve-se seguir o caso de uso 1 antes.
2. Sistema localiza o local fornecido.
3. Ator fornece a classificação do grau de congestionamento através de um número inteiro (nota).

Pós-condição: Banco de dados do sistema atualizado com a nova informação de trânsito.

Exceções:

1. No passo 2, é possível que o sistema não encontre o local indicado. Nessa situação, uma mensagem de erro é exibida e o sistema volta ao passo 1.

4.3.4 Caso de uso 4: Gerenciar Cadastro

Descrição: Usuário se cadastra no sistema e informa os pontos da cidade que mais utiliza.

Autor: Usuário do sistema

Seqüência de eventos:

1. Usuário fornece seu nome e senha.
2. Sistema exibe o mapa da cidade.
3. Usuário seleciona um ponto da cidade, seja por indicação no mapa ou pelo fornecimento do endereço. No caso de indicação no mapa, deve-se seguir o caso de uso 1 antes. O usuário também fornece um nome para esse ponto.

4. Sistema registra o ponto selecionado.

Observação: Os passos 3 e 4 podem ser executados quantas vezes o usuário desejar.

Pós-condição: O caso de uso é finalizado quando o usuário não deseja mais repetir os passos 3 e 4. O banco de dados do sistema é então atualizado com as informações de cadastro.

Exceções:

1. No passo 1, o usuário pode fornecer um nome já existente. Nesse caso, uma mensagem de erro é exibida e o sistema volta ao passo 1.
2. No passo 3, é possível que o sistema não encontre o local indicado. Novamente, um erro é sinalizado e o sistema retorna ao passo 3.

4.4 Diagrama de Classes

A figura 62 ilustra o diagrama de classes do sistema proposto. O diagrama de classes exposto nessa figura contém basicamente três pacotes principais:

- **OSM:** As classes deste pacote modelam o arquivo XML do OpenStreetMap. Desse modo, tem-se:

Node: Representa um determinado ponto no mapa (com informações geográficas de latitude e longitude);

Way: Conjunto ordenado de Nodes, modelando uma via qualquer da cidade e seu sentido;

Tag: Classe do tipo chave/valor, relacionada com Node e Way que apresenta alguns dados adicionais como o nome da rua;

Osm: Contém uma lista de Nodes e uma lista de Ways.

- **MATH:** As classes deste pacote representam um modelo de grafo ponderado genérico e são utilizadas para o processamento do algoritmo de melhor caminho. As arestas do grafo contêm uma lista de pesos para que, neste sistema, possam ser adicionadas informações de trânsito provindas de diferentes fontes.

- **INFO:** As classes deste pacote contém as classes responsáveis por tratar as informações de trânsito, provenientes de diferentes fontes possíveis. Por isso, há a necessidade da interface Fonte, que modela o comportamento de uma fonte de informações de trânsito qualquer.

Para cada um desses pacotes, foi modelada uma classe gerenciadora, responsável por controlar suas respectivas classes e fornecer uma interface simples de acesso. Juntando o comportamento dessas três classes gerenciadoras, existe o integrador, que executa, delegando funções, a principal tarefa do sistema, que é traçar uma rota, dados origem e destino.

4.5 Diagrama de Implantação

O modelo de implantação para o diagrama de instalação se encontra na figura 63. Sendo que o projeto é composto de quatro componentes principais:

1. O servidor da CET, que será o responsável por fornecer as informações do trânsito de São Paulo através de WebService. A conexão com ele é bidirecional, representando as requisições e as respostas.
2. O servidor, que será o computador no qual serão executados o programa de geração de rotas e os dois bancos de dados (o de mapas, responsável por armazenar os dados das vias de toda a cidade, e o geral, responsável por armazenar dados dos usuários, do sistema e dados para o sistema adaptativo). Os bancos de dados poderiam ter sido separados em diferentes máquinas, o que seria o correto e ajudaria no desempenho, mas acarretaria um aumento do custo do aluguel de servidores, o grupo não considerou aceitável.
3. Um ou vários computadores pessoais rodando um *browser* para acesso ao site de geração de rotas.
4. Um ou vários celulares rodando um aplicativo de acesso a Internet para acesso a o site.

4.6 Diagrama de Atividades

Na listagem a seguir são descritos os diagramas de atividades do projeto, assim como os casos de uso relacionados e as figuras pertencentes à esses diagramas. As figuras pertencentes ao diagramas de atividades se encontram no anexo A do relatório.

- **Diagrama de Atividade 1:** Refere-se ao caso de uso 1, ou seja, descreve o processo de solicitação de informações de trânsito para certa região da cidade. Esse diagrama se encontra na figura 64 do anexo do projeto.
- **Diagrama de Atividade 2:** Refere-se ao caso de uso 2, ou seja, descreve o processo de solicitação do melhor caminho entre dois pontos, evitando as ruas congestionadas. Esse diagrama se encontra na figura 65 do anexo do projeto.
- **Diagrama de Atividade 3:** Refere-se ao caso de uso 3, ou seja, descreve o processo de fornecimento de informação de trânsito através do telefone celular. Esse diagrama se encontra na figura 66 do anexo do projeto.
- **Diagrama de Atividade 4:** Refere-se ao caso de uso 4, ou seja, descreve o processo de cadastramento de um local de interesse para um usuário. Esse diagrama se encontra na figura 67 do anexo do projeto.

4.7 Diagramas de Seqüência

Os diagramas de seqüencia referentes aos casos de uso anteriormente citados são descritos nessa seção. Na listagem a seguir são descritos os diagramas de sequência referentes ao casos de uso 1, 2, 3 e 4, respectivamente. As figuras correspondentes à esses diagramas de sequência se encontram no anexo A do relatório.

- **Diagrama de Sequência 1:** Referencia o caso de uso 1. Esse diagrama se encontra na figura 68 do anexo do projeto.
- **Diagrama de Sequência 2:** Referencia o caso de uso 2. Esse diagrama se encontra na figura 68 do anexo do projeto.
- **Diagrama de Sequência 3:** Referencia o caso de uso 3. Esse diagrama se encontra na figura 68 do anexo do projeto.

- **Diagrama de Sequência 4:** Referencia o caso de uso 4. Esse diagrama se encontra na figura 68 do anexo do projeto.

4.8 Gerenciamento do Projeto

A seguir, são explicados os artefatos utilizados pelo grupo para o gerenciamento do projeto.

4.8.1 Estrutura Analítica do Projeto

A figura 5 apresenta o diagrama da EAP do projeto, determinando o escopo do mesmo.

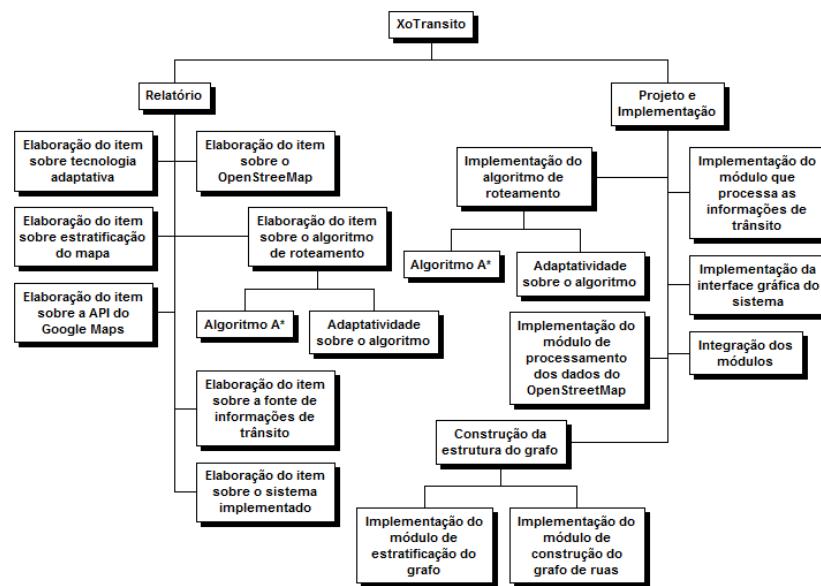


Figura 5: Estrutura Analítica do Projeto.

4.8.2 Cronograma

O cronograma que tinha sido planejado nesse projeto se encontra na figura 6. Nesse cronograma estava previsto o término da implementação das partes essenciais do projeto no mês de setembro e os meses restantes estariam reservados para a melhora do sistema e correção de *bugs*. Em todo o cronograma foi considerado o tempo de desenvolvimento mais testes, pois o desenvolvimento seria sempre acompanhado de testes.

	Task Name	Duration	Start	Finish	Predecessors
1	■ Relatório	14 days	Mon 18/8/08	Tue 2/9/08	11
2	Adaptatividade	3 days	Mon 18/8/08	Fri 22/8/08	
3	Extratificação do mapa	3 days	Mon 18/8/08	Fri 22/8/08	
4	API gráfica	2 days	Mon 18/8/08	Thu 21/8/08	
5	Open Street Maps	2 days	Mon 18/8/08	Thu 21/8/08	
6	■ Algoritmo de roteamento	7 days	Fri 22/8/08	Fri 29/8/08	
7	A-Estrela	3 days	Fri 22/8/08	Sat 23/8/08	3
8	A-Estrela com adaptatividade	4 days	Sat 23/8/08	Fri 29/8/08	7
9	Informação do transito	3 days	Thu 21/8/08	Sat 23/8/08	4
10	Sistema Geral	4 days	Fri 29/8/08	Tue 2/9/08	9;8;2;3;4;5
11	■ Projeto	48 days	Tue 1/7/08	Mon 18/8/08	
12	■ Algoritmo de roteamento	18 days	Tue 1/7/08	Sat 19/7/08	
13	Implementação do A-Estrela	8 days	Tue 1/7/08	Wed 9/7/08	
14	Integração de técnicas adaptativas no algoritmo A-Estrela	10 days	Wed 9/7/08	Sat 19/7/08	13
15	Implementação do módulo de processamento de dados do OSM	5 days	Sat 19/7/08	Fri 25/7/08	13;14
16	Implementação do módulo que pega as informações do transito	5 days	Tue 1/7/08	Sat 5/7/08	
17	Implementação da interface gráfica do sistema	5 days	Fri 25/7/08	Tue 29/7/08	15
18	■ Construção da estrutura da arvore	33 days	Tue 1/7/08	Sat 2/8/08	
19	Implementação do módulo de extratificação do mapa	10 days	Tue 1/7/08	Fri 11/7/08	
20	Implementação do módulo de construção das árvores	5 days	Tue 29/7/08	Sat 2/8/08	19;17
21	Implementação do sistema totalmente integrado	15 days	Sat 2/8/08	Mon 18/8/08	14;15;16;17;18;20

Figura 6: Cronograma para o segundo semestre.

4.8.3 Custos

Por se tratar de um projeto de *software*, em que itens de *hardware*, componentes especiais ou kits de desenvolvimento não são necessários, o custo acaba se tornando baixo. Contabiliza-se apenas o aluguel do servidor e os custos de homem-hora gastos durante o desenvolvimento.

A seguir está o detalhamento do custo que tinha sido calculado em função do cronograma planejado. O valor do homem-hora de cada integrante do grupo custa R\$ 30 e o do orientador custa R\$ 500. Abaixo, os cálculos são mostrados utilizando-se homem-hora como unidade.

1. Custos em homem-hora:

(a) Planejamento

- i. Fabio Sussumu - Criação do diagrama de classes, requisitos e diagrama de pacotes: 10 homem-hora
- ii. Rodrigo Morelli - Criação do diagrama de implantação, análise dos riscos e estimativa dos custos: 10 homem-hora
- iii. Tiago Belo Torres - Criação do diagrama de atividades, casos de usos e descrição do algoritmo: 10 homem-hora
- iv. Tiago Matos - Criação do diagrama de seqüência, organização e gerenciamento do relatório: 10 homem-hora

(b) Discussões sobre idéias, metas e reuniões

- i. Grupo: 4×15 homem-hora

(c) Reuniões com o orientador, 1 hora por semana:

- i. Grupo + orientador: 5×12 homem-hora

(d) Estudo de temas, como adaptatividade, algoritmo A*

- i. Grupo: 4×8 homem-hora

(e) Pesquisa de tecnologias, tais como banco de dados de mapas, API de apresentação gráfica do mapa, informações do trânsito:

- i. Grupo: 4×8 homem-hora

(f) Elaboração de algumas provas de conceito:

- i. Programa para receber informações do trânsito da MapLink: 8 homem-hora

- ii. Programa que gera rotas utilizando o algoritmo A*: 20 homem-hora
 - iii. Implantação da interface gráfica do Google Maps para visualização das rotas traçadas: 5 homem-hora
 - iv. Programa capaz de ler os dados do OpenStreetMap e convertê-lo numa hierarquia de capaz de ser processada: 15 homem-hora
2. Servidor:
- (a) R\$ 20 por mês

4.8.4 Estrutura Analítica de Risco

Riscos de projeto são condições que, caso venham a ocorrer, podem comprometer ou impedir a realização de um dado projeto. Um risco pode ter uma ou mais causas e, se ocorrer, um ou mais impactos.

Para a análise dos riscos do projeto, será utilizado o método de EAR (Estrutura Analítica de Riscos), tendo como base o PMBOK (*Project Management Body of Knowledge*).

Para a identificação dos riscos, a técnica utilizada foi o *brainstorming* em conjunto com a análise **SWOT**. Através dessas variáveis é obtida como saída uma lista de riscos pertencentes ao projeto.

A figura 7 ilustra as categorias e subcategorias nas quais os riscos podem surgir em um projeto típico.

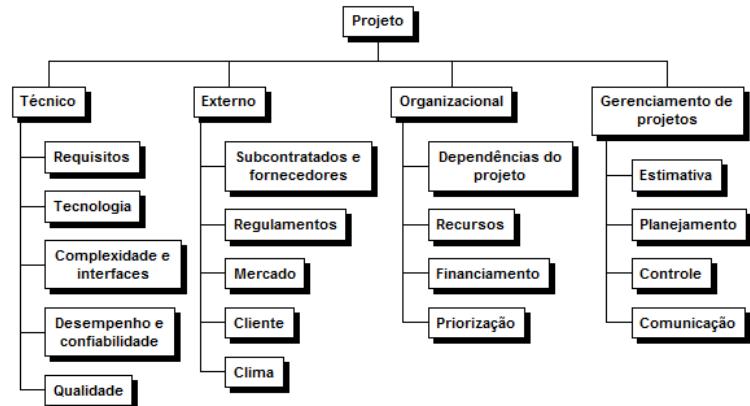


Figura 7: Exemplo de uma EAR de um projeto.

A meta do *brainstorming* é obter uma lista abrangente de riscos do projeto. Em seguida, os riscos são identificados e categorizados por tipo de risco e suas definições são

refinadas.

A análise SWOT é uma ferramenta utilizada para fazer análise de cenário (ou análise de ambiente). O termo SWOT é uma sigla oriunda do idioma inglês, e é um acrônimo de **forças** (*strengths*), **fraquezas** (*weaknesses*), **oportunidades** (*opportunities*) e **ameaças** (*threats*).



Figura 8: Exemplo de diagrama de Análise SWOT.

Essa análise de cenário se divide em ambiente interno (forças e fraquezas) e ambiente externo (oportunidades e ameaças).

4.8.4.1 Identificação dos Riscos

A identificação de riscos ajuda a determinar e armazenar as variáveis (ou fatores) que podem atrapalhar o desenvolvimento do projeto.

Essa atividade de identificação é um processo contínuo e iterativo, pois novos riscos podem ser conhecidos conforme o projeto se desenvolve durante todo o seu ciclo de vida.

Juntamente com a Identificação dos riscos, é feita uma análise qualitativa dos riscos. A análise qualitativa de riscos avalia a prioridade dos riscos identificados usando a probabilidade deles ocorrerem, o impacto correspondente nos objetivos do projeto se os riscos realmente ocorrerem, além de outros fatores, como o prazo e tolerância a risco das restrições de custo, cronograma, escopo e qualidade do projeto.

A avaliação de probabilidade de riscos investiga a probabilidade de cada risco específico ocorrer. A avaliação de impacto de riscos investiga o efeito potencial sobre um objetivo do projeto, como tempo, custo, escopo ou qualidade, inclusive os efeitos negativos das ameaças e os efeitos positivos das oportunidades.

Na análise dos riscos inerentes a este projeto, os mesmos serão classificados de acordo com a listagem a seguir, sendo que ela se encontra em ordem crescente de probabilidade:

- Improvável;
- Pouco provável;
- Provável;
- Muito provável;
- Certeza.

Também foi realizada a classificação do impacto desses riscos no desenvolvimento do projeto. Para classificar o impactos dos riscos foi utilizada a tabela indicada na figura 1. A escala de impacto reflete a importância do impacto, negativa para ameaças ou positiva para oportunidades, em cada objetivo do projeto se ocorrer um risco. As escalas de impacto são específicas do objetivo potencialmente afetado, do tipo e tamanho do projeto, da situação financeira e estratégias da organização e da sensibilidade da organização a impactos específicos. As escalas relativas de impacto são descritores classificados de forma simples, como “muito baixo”, “baixo”, “moderado”, “alto” e “muito alto”, refletindo impactos cada vez maiores conforme definido pela organização.

Tabela 1: Definições de impactos de risco para quatro objetivos diferentes do projeto.

Condições definidas para escalas de impacto de um risco em objetivos importantes do projeto (os exemplos são mostrados somente para os impactos negativos)					
Objetivo do projeto	Muito baixo 0,05	Baixo 0,10	Moderado 0,20	Alto 0,40	Muito alto 0,80
Custo	Aumento de custo não significativo	Aumento de custo < 10%	Aumento de custo de 10 a 20%	Aumento de custo de 20 a 40%	Aumento de custo > 40%
Tempo	Aumento de tempo não significativo	Aumento de tempo < 5%	Aumento de tempo de 5 a 10%	Aumento de tempo de 10 a 20%	Aumento de tempo > 20%
Escopo	Diminuição de escopo quase imperceptível	Áreas menos importantes do escopo afetadas	Áreas importantes do escopo afetadas	Redução do escopo inaceitável para o patrocinador	Item final do projeto sem nenhuma utilidade
Qualidade	Degradação da qualidade quase imperceptível	Somente as aplicações mais críticas são afetadas	Redução da qualidade exige aprovação do patrocinador	Redução da qualidade inaceitável para o patrocinador	Item final do projeto sem nenhuma utilidade

Levantamento dos riscos com base na tabela 1:

1. Técnicos

- (a) Impossibilidade de a equipe de implementar todos os requisitos propostos (fraquezas).
 - i. Probabilidade: provável - 0,2
 - ii. Impacto: moderado
- (b) Não conseguir uma base de dados completa e confiável da cidade de São Paulo sobre informações de ruas e mapas (ameaça).
 - i. Probabilidade: provável - 0,4
 - ii. Impacto: muito alto
- (c) Não conseguir informações de trânsito em tempo real da cidade de São Paulo (ameaça).
 - i. Probabilidade: improvável - 0,05
 - ii. Impacto: moderado
- (d) Desempenho do *software* ser relativamente lento em comparação com os existentes no mercado (fraquezas).
 - i. Probabilidade: muito provável - 0,6
 - ii. Impacto: muito baixo
- (e) O *software* não apresentar qualidade em relação a interface, velocidade e funções extras, se comparado aos existentes no mercado atualmente (fraquezas).
 - i. Probabilidade: muito provável - 0,6
 - ii. Impacto: muito baixo

2. Externos (Ameaças)

- (a) A CET não disponibilizar as informações de trânsito da cidade de São Paulo.
 - i. Probabilidade: provável - 0,3
 - ii. Impacto: moderado
- (b) O MapLink alterar a página de trânsito, impossibilitando a coleta de dados do trânsito.
 - i. Probabilidade: muito provável - 0,6
 - ii. Impacto: moderado
- (c) O MapLink e CET pararem de disponibilizar informações do trânsito da cidade de São Paulo para o grupo.
 - i. Probabilidade: pouco provável - 0,2

- ii. Impacto: muito alto

3. Organizacional

- (a) Alguns dos recursos humanos tiver que se ausentar durante períodos longos (fraquezas).

- i. Probabilidade: pouco provável - 0,2

- ii. Impacto: alto

- (b) O orientador do projeto tiver que se ausentar por períodos longos (ameaça).

- i. Probabilidade: pouco provável - 0,2

- ii. Impacto: muito alto

4. Gerenciamento de Projetos

- (a) Impossibilidade de entregar o projeto completo até a data final de entrega (fraquezas).

- i. Probabilidade: pouco provável - 0,2

- ii. Impacto: muito alto

- (b) Não conseguir seguir o planejamento devido a interferências das aulas, trabalho, casualidades (ameaça).

- i. Probabilidade: provável - 0,4

- ii. Impacto: alto

Como foi visto, os riscos foram ponderados, e a partir da análise cuidadosa desses riscos, foi decidido dar continuidade com o projeto. O próximo capítulo abordará a parte de implementação do projeto, em que serão discutidos os problemas durante a implementação e os riscos que se tornaram realidade.

5 Projeto e Implementação

A seguir, aspectos técnicos de projeto e implementação do sistema são apresentados. Inicialmente, a arquitetura do projeto como um todo é explanada, juntamente com todas as tecnologias de programação envolvidas. Logo em seguida, é apresentado o trabalho referente aos mapas (APIs e fontes de dados). Finalmente, todos os componentes que compõem o projeto final são explicados em maiores detalhes: LIBOSM (tratamento do arquivo XML do OpenStreetMap), MAPLINK (consulta de situações de trânsito), SIMCET (simulador de situações de trânsito) e XOTRANSITO (projeto principal).

5.1 Arquitetura Geral

A arquitetura geral do sistema, incluindo todos os seus componentes, é apresentada na figura 9:

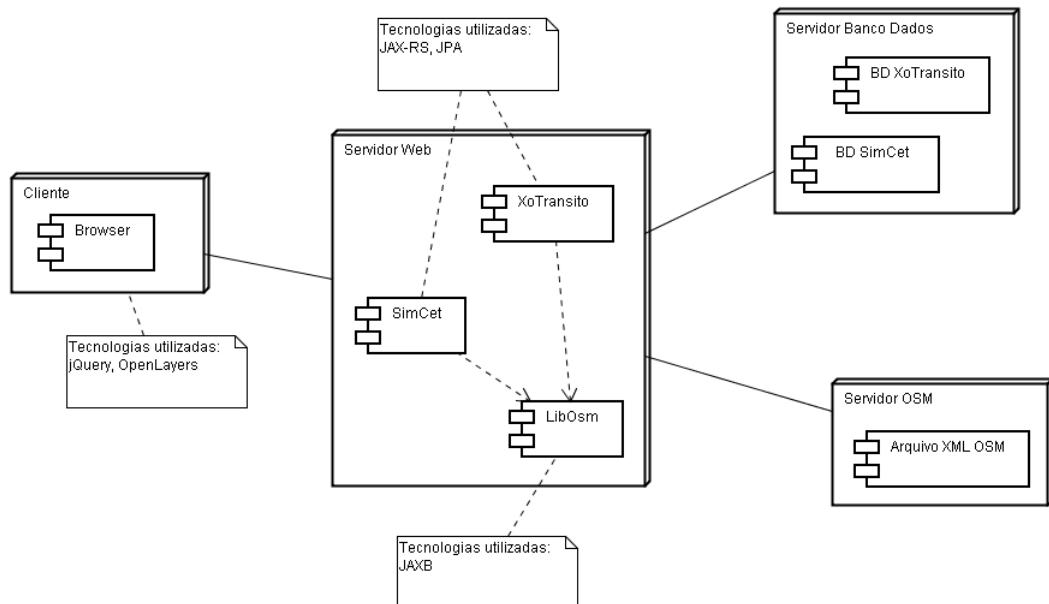


Figura 9: Visão geral dos componentes, incluindo tecnologias utilizadas.

A arquitetura está dividida basicamente em 3 camadas físicas (*tiers*):

- **Servidor de Banco de Dados:** Nó responsável por armazenar os dados persistentes utilizados pelos sistemas XoTRANSITO (dados de usuários e seus locais cadastrados) e SIMCET (dados contendo as situações de trânsito dos vários perfis existentes). O arquivo XML fornecido pelo OpenStreetMap também está incluso nessa camada, embora não seja um banco de dados propriamente dito. Todos os dados que precisam ser recuperados após um reinício do sistema precisam estar nessa camada.
- **Servidor de Aplicações Web:** Nó em que são implantados os sistemas de aplicação propriamente ditos, que compõem o projeto como um todo. São eles: XoTRANSITO (sistema principal, que efetivamente encontra o melhor caminho, dados os pesos de comprimento, trânsito e velocidade das ruas), SIMCET (sistema que simula um fornecedor de dados de trânsito, como a CET) e LIBOSM (biblioteca que processa os dados do arquivo XML provindo do projeto OpenStreetMap).
- **Cliente:** Nó referente aos usuários que acessarão o sistema. O sistema de utiliza da tecnologia Web e, portanto, requer que o cliente possua, em sua máquina, instalado um navegador Web, com suporte a JavaScript.

5.1.1 Requisitos para Implantação

Para ser possível executar o projeto, são necessários os seguintes componentes de *software*:

- Servidor compatível com a especificação *Java 5 Enterprise Edition*, como, por exemplo, JBoss, BEA WebLogic, etc. (recomendável utilizar GlassFish, para evitar problemas de migração).
- Banco de dados compatível com a especificação *Java Persistence API*, como Oracle, PostgreSQL, MySQL, etc. (recomendável utilizar JavaDB).

As recomendações acima foram estabelecidas devido ao fato de que o desenvolvimento dos sistemas foi realizado nos ambientes citados (GlassFish e JavaDB). Porém, seguindo-se a filosofia da tecnologia Java, o projeto pode ser executado em qualquer ambiente que atenda às especificações da Sun, sendo necessária apenas uma configuração para adaptação entre os diferentes ambientes.

5.1.2 Resumo das Tecnologias Java e JavaScript Utilizadas

O projeto foi implementado fazendo-se uso, principalmente, de tecnologias Java e JavaScript. Dado que, atualmente, tais linguagens de programação se apresentam altamente difundidas, porém, com um número muito grande de bibliotecas e estruturas disponíveis para o público com o intuito de facilitar e agilizar o processo de desenvolvimento, uma breve explanação de cada uma das tecnologias usadas se faz necessária. Adotando-se o modelo MVC (*model-view-controller*), há 3 camadas (*layers*) com objetivos diferentes:

- ***Model* (modelo):** camada responsável por realizar as operações básicas que dizem respeito ao núcleo do sistema. Corresponde à parte de um *software* que independe da interface do usuário. Desse modo, o mesmo modelo pode ser reutilizado para vários tipos de interface com o usuário, tais como: *desktop*, Internet, celular, etc.
- ***View* (apresentação):** camada responsável por tratar a interface do usuário propriamente dita, ou seja, cuida da visualização dos dados, sua formatação e o modo como são apresentados ao usuário. Essa camada pode sofrer mudanças significativas e ser adaptada para ser executada em diferentes tipos de dispositivo
- ***Controller* (controladora):** camada responsável por realizar a comunicação e a integração entre o modelo e a apresentação. Trata-se de uma camada simples que executa o trabalho de intermediamento entre as outras duas camadas.

Na parte de apresentação, foram utilizadas tecnologias básicas da Web, como HTML, CSS e JavaScript. Além disso, foram empregadas duas outras bibliotecas para JavaScript, que são detalhadas a seguir:

- **jQuery:** Biblioteca JavaScript usada para facilitar o desenvolvimento de *scripts*. Além disso, possui diversos *plugins* que auxiliam no processo de codificação. Dentre eles, o *plugin* utilizado mais importante é o **Timers**, responsável por executar chamadas AJAX (*Asynchronous JavaScript and XML*) a cada determinado período de tempo pré-estabelecido, recurso usado para exibir a animação dos nós visitados em tempo real no projeto.
- **OpenLayers:** Biblioteca JavaScript usada para exibir os mapas do OpenStreetMap ao usuário e que, portanto, constitui-se na interface básica do projeto. Com essa biblioteca, é possível configurar as fontes das imagens do mapa (do próprio OpenStreetMap, de um servidor próprio contendo as imagens, etc.), fornecer ao

usuário funcionalidades tais como navegação no mapa e controle de *zoom*, assim como mostrar e esconder as camadas apresentadas. No projeto, foram criadas diversas camadas, cada uma com a sua finalidade. Existe a camada base, contendo as ruas propriamente ditas; uma camada contendo os pontos de origem e destino selecionados; outra camada com o caminho traçado e outra com os dados de trânsito.

Na parte da controladora, foi utilizada a nova tecnologia de RESTful WebServices, especificada recentemente para a plataforma Java segundo a JAX-RS:

- **JAX-RS:** A tecnologia RESTful permite a criação de WebServices que ficam atrelados a URLs, sem incorporar a burocracia do protocolo SOAP. Na prática, é uma descompilação dos SOAP WebServices (os WebServices originais), com o novo conceito de utilizar URLs como recursos. Dessa forma, ações são executadas dentro do sistema através de invocações HTTP sobre URLs pré-estabelecidas, retornando, geralmente, dados em formato XML (mas, teoricamente, o tipo de retorno pode ser qualquer formato).

Finalmente, na parte do modelo, foram usadas as seguintes tecnologias da plataforma Java:

- **JPA (Java Persistence API):** trata-se de uma especificação Java com o intuito de lidar com a persistência (comunicação com o banco de dados) de objetos Java. Sua idéia principal é realizar um mapeamento de objetos Java com tabelas de BD e, a partir disso, gerenciar todas as operações que ocorrem com o SGBD (Sistema Gerenciador de Banco de Dados), ou seja, inclusões, alterações, exclusões e consultas. O programador passa a trabalhar somente com os objetos mapeados, não se preocupando com códigos SQL, pois este trabalho é abstraído pelo JPA.
- **JAXB (Java API for XML Binding):** trata-se de uma especificação Java para mapeamento de classes Java para XML, ou seja, é a mesma idéia do JPA, só que aplicado ao XML, ao invés de banco de dados. Desse modo, classes Java são mapeadas em relação a esquemas XML e, assim, essa biblioteca é capaz de realizar a conversão (chamada de *marshalling* e *unmarshalling*) de objetos Java para XML e vice-versa. Esse módulo é intensamente utilizado na biblioteca implementada LIBOSM, que processa o XML provindo do OpenStreetMap.

5.2 Mapas de Ruas e suas APIs de Visualização

Em relação aos mapas de ruas, existem, no mercado, diversas opções que, em sua maioria, são proprietárias e com um custo expressivo. Durante o desenvolvimento deste projeto, foi realizada uma pesquisa a respeito e encontrado um projeto livre chamado OpenStreetMap, explanado na seção seguinte.

Quando se trata de mapas de ruas, é necessário avaliar que existem no mercado diversas opções gráficas (que permitem a visualização simples das imagens de ruas). Porém, para o projeto XoTRANSITO, é necessário, além dessas interfaces, um conjunto de dados referentes às posições das diversas ruas existentes, além de suas conexões.

5.2.1 OpenStreetMap

O OpenStreetMap é um projeto de código-fonte aberto cujo objetivo principal é mapear as ruas por todo mundo. No caso específico da cidade de São Paulo, alvo deste projeto, os dados referentes às vias de trânsito de veículos automotores ainda não estão completos, porém, é possível realizar testes com trechos relativamente grandes na cidade.

Essa fonte de dados foi escolhida por se tratar de uma opção sem custos, uma vez que todas as alternativas pesquisadas mostraram-se inviáveis do ponto de vista financeiro. Além disso, este projeto possui um grande potencial e a tendência é que, com o passar do tempo, seus dados passem a ter uma qualidade mais expressiva. No longo prazo, passará a ser uma grande alternativa para as necessidades de qualquer instituição no que se refere a dados geográficos de ruas.

Na figura 10, é apresentada a tela da página do OpenStreetMap, focada em São Paulo. Como se pode observar, nem todas as ruas estão mapeadas, mas, conforme dito, trate-se de um projeto livre no qual qualquer pessoa pode colaborar. Além disso, o formato dos dados utilizados para o projeto XoTRANSITO é XML, totalmente flexível e que pode sofrer alterações de dados, mudanças estas que são aceitas pelo XoTRANSITO sem grandes transtornos.

5.2.2 APIs de Visualização

Para compor a interface gráfica do sistema, foi realizada outra pesquisa em relação às diversas opções já existentes. A idéia dessas APIs é fornecer, de um modo mais simplificado para o programador, uma interface para Web que exiba o mapa propriamente dito,

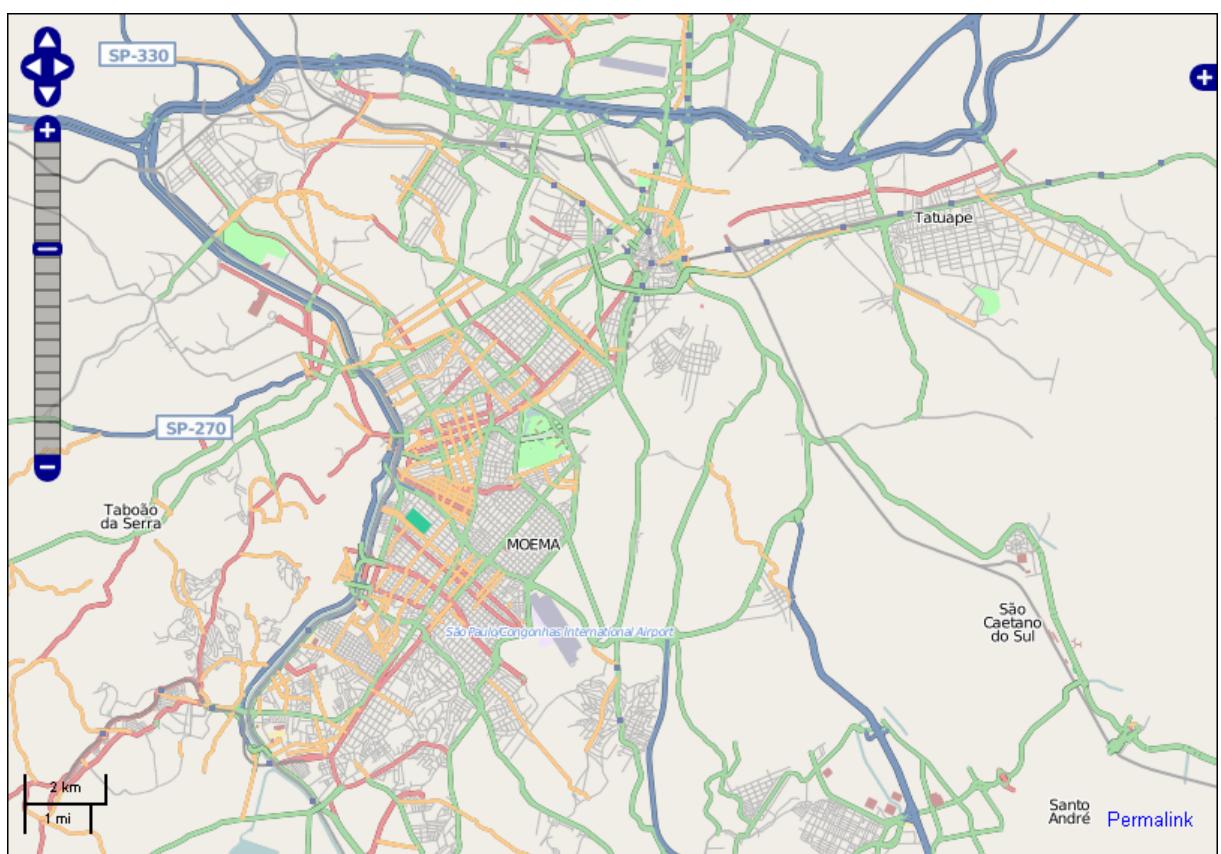


Figura 10: Tela do OpenStreetMap, focada na cidade de São Paulo.

além de prover funcionalidades extras, tais como: *zoom*, movimentação, camadas extras, desenho de figuras geométricas (pontos, linhas e polígonos), etc.

As opções avaliadas são detalhadas a seguir, mostrando-se vantagens e desvantagens que motivaram ou não o seu uso durante a fase de implementação deste projeto.

5.2.2.1 Google Maps

Dentre as opções levantadas, a API do Google Maps proporciona a melhor experiência para o usuário, uma vez que os seus dados apresentam-se bem completos, devido à fonte tratar-se de uma empresa comercial. Por esse motivo, foi utilizada em nosso projeto na sua fase inicial, porém, durante a fase de implementação, foi trocada por outra API.

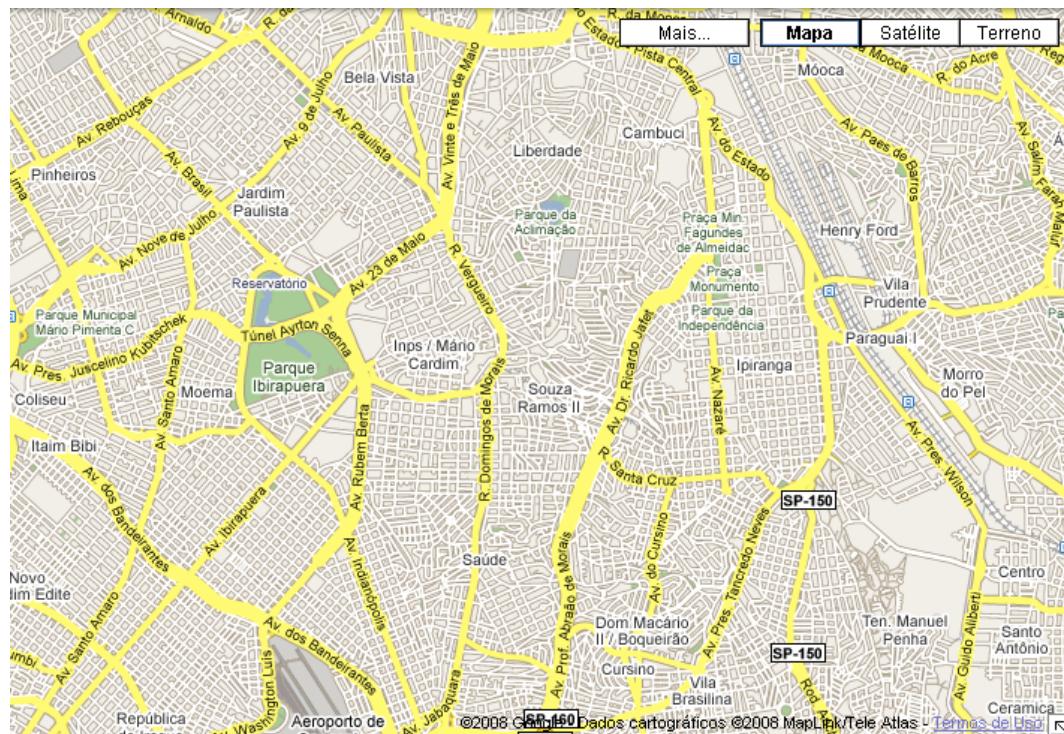


Figura 11: Interface da API do Google Maps.

Tal ocorrência é explicada pelo fato de que muitas ruas apresentadas graficamente pelo Google Maps não estão mapeadas pelo OpenStreetMap e, portanto, não são tratadas pelo algoritmo de roteamento, o que acaba gerando inconsistências entre a interface e a implementação.

5.2.2.2 Mapstraction

Conforme relatado no item anterior, a API do Google Maps foi trocada por essa API, que se trata de um projeto livre cujo objetivo é abstrair as várias APIs já existentes no mercado, criando uma interface única para todas, ou seja, é possível escrever um código único e o mesmo ser compatível com as várias interfaces já existentes, como Google Maps, OpenLayers, etc.

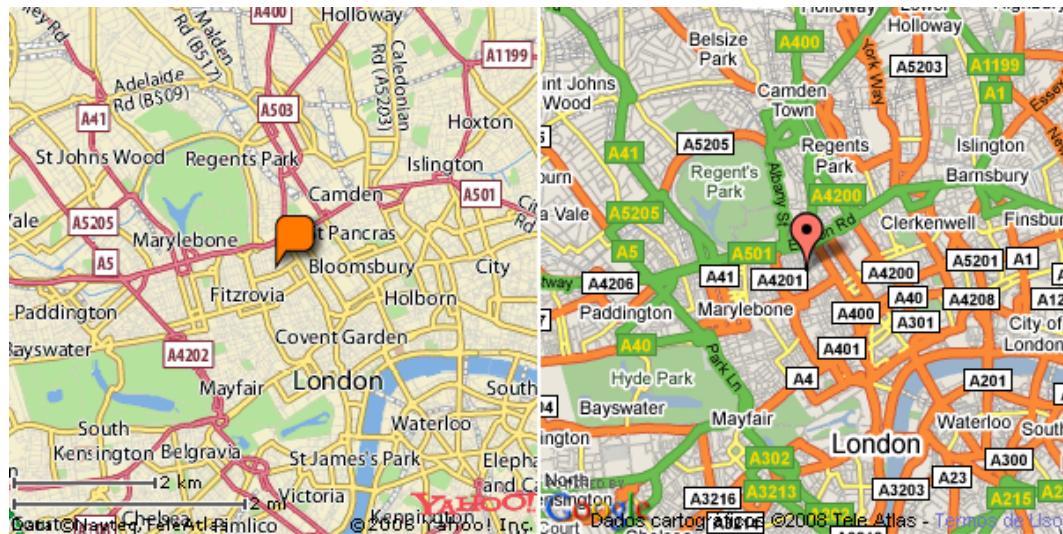


Figura 12: Demonstração da flexibilidade do Mapstraction, para diferentes APIs (Yahoo e Google).

Sua idéia é interessante, porém, foi verificado que esse projeto se encontra bastante incompleto. Num determinado momento da implementação, surgiram funcionalidades, como a criação de camadas independentes da Internet, que ainda não estavam implementadas pelo Mapstraction o que, portanto, inviabilizou o seu uso.

5.2.2.3 OpenLayers

O OpenLayers é a API oficial do OpenStreetMap. Fornece um conjunto de funções mais complexas que o Mapstraction, no entanto, encontra-se num estágio mais avançado de desenvolvimento e, portanto, foi adotado como API para o sistema final. Entre as suas funcionalidades apresentadas, destaca-se a possibilidade de criação de camadas, que podem ser exibidas e escondidas pelo usuário, através de uma janela em sua interface.

O motivo pelo qual foi escolhido entre as diversas opções estudadas é o fato de que é possível configurá-lo para que acesse um determinado servidor de imagens, além do padrão. Essa característica é importante para prover alternativas para o usuário de escolher outro

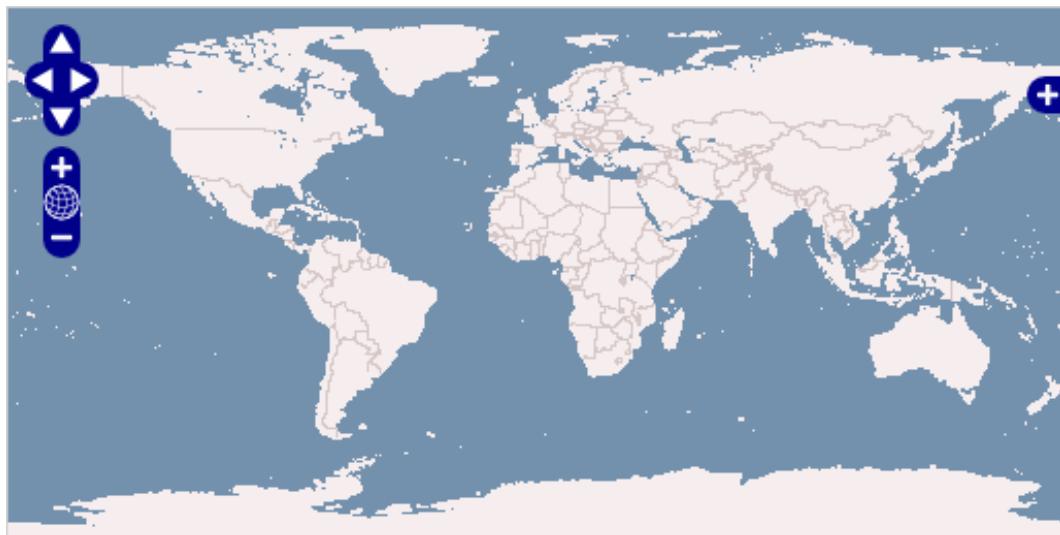


Figura 13: Interface do OpenLayers.

provedor de imagens de mapa, no caso de algum dos servidores estar fora do ar, por exemplo.

5.3 LibOsm

Para tratar a base de dados de mapas fornecida pelo OpenStreetMap, que é utilizada tanto no SIMCET como no XoTRANSITO, foi criado uma biblioteca de código fonte Java que traduz essa base de dados, que é um arquivo XML, em objetos e fornece métodos para trabalhar com esses objetos. Essa biblioteca é o LIBOSM. A figura 14 demonstra, em alto nível, a conversão realizada entre o modelo fornecido pelo OpenStreetMap e o adotado para o algoritmo de roteamento.

5.3.1 Explicação das Classes e do Funcionamento do Módulo

A biblioteca LIBOSM é composta de quatro classes: `Lugar`, `Osm`, `GerenciadorOsm` e `ObjectFactory`. Na figura 15 é mostrado o diagrama de classes dessa biblioteca.

A classe `Lugar` representa um lugar no mundo real, descrito por uma latitude e uma longitude. Os métodos dentro dessa classe são para calcular a distância e a inclinação entre o objeto `Lugar` instanciado e o objeto `Lugar` passado na chamada do método, ou seja, um objeto `Lugar` sabe a distância ou inclinação entre ele e qualquer outro objeto `Lugar`, a partir de fórmulas matemáticas considerando a circunferência do globo.

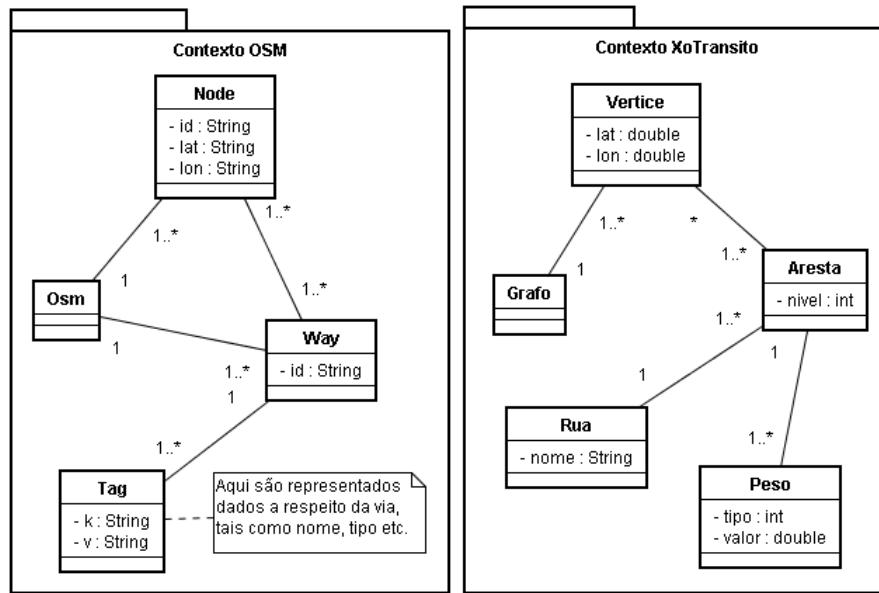


Figura 14: Modelos do OpenStreetMap e do XoTRANSITO, em alto nível.

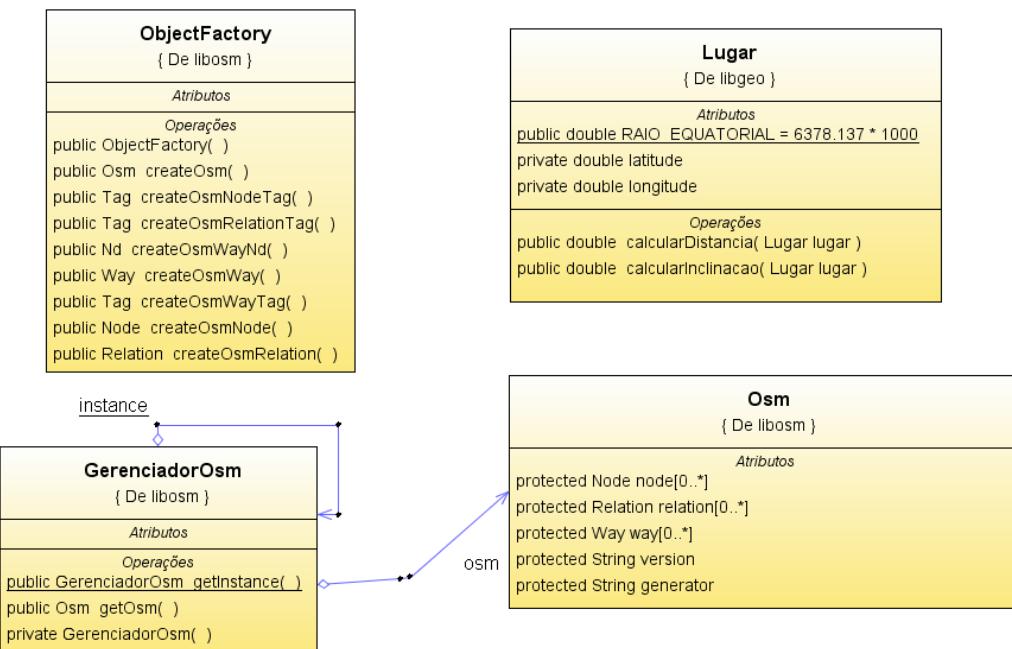


Figura 15: Diagrama de classes do LIBOSM.

A classe `Osm` é a transformação em objetos do XML da base de dados do OpenStreetMap. Essa classe é implementada sobre a arquitetura JAXB. Essa arquitetura faz todo o serviço de converter as *tags* do XML para atributos dentro de um objeto, ou seja, essa classe faz um mapeamento do XML em forma de objetos, de forma que o acesso ocorre com mais facilidade através de métodos `get`, `set` e métodos implementados pelo grupo.

Para completar a classe `Osm`, é gerada uma classe `ObjectFactory`, pelo *framework* JAXB, que contém apenas métodos para criação de novos objetos da própria classe `Osm`. Essas classes, em conjunto, são capazes de traduzir o XML do OpenStreetMap em objetos Java, de fácil utilização.

Passando para a próxima classe, `GerenciadorOsm`, percebesse que no diagrama, ela mesma se instância, isso significa que ela é um *singleton*, ou seja, isso garante que existe apenas uma instancia dela rodando em todo o sistema. Essa classe tem a simples função de instanciar um objeto `Osm` dentro dela e chamar o método do *framework* `JAXBContext.newInstance` para transformar o XML em um objeto `Osm`, com atributos utilizáveis pelo programa.

5.4 MapLink

Para a heurística ser o mais próxima o possível do trânsito cotidiano real, o grupo implementou um programa que pega dados reais de trânsito do site do MapLink. Esses dados são pegos através de `posts` e `gets` em páginas HTML na Internet, nas quais o MapLink disponibiliza esses dados.

Como essa não era uma meta do grupo, e sim apenas uma melhoria, o projeto ainda está em fase final de desenvolvimento. Por enquanto o projeto consegue apenas acessar o site, pegar o trânsito e armazená-lo no banco de dados, mas é necessário um processamento trabalhoso desses dados brutos para serem úteis ao projeto, processamento esse que demanda muito tempo de desenvolvimento. Caso esse projeto fique pronto a tempo da apresentação, ele contribuirá com dados reais de trânsito, que durante o percurso do algoritmo A* temporizado, trará mais realismo às simulações.

Para isso, foi criada uma base de dados especial para a heurística, que vai ser brevemente descrita, no qual é utilizado perfis diferentes do SIMCET, com base em horários restritos, com hora e dia da semana bem definidos para aproveitamento do algoritmo temporizado. A situação do trânsito também foi alterada para salvar o nome da rua a que se refere, ao invés de códigos de vértices. E por fim, foram criadas tabelas espe-

ciais para o mapeamento do OSM no MAPLINK e uma tabela para a média do trânsito de determinadas ruas, em determinados horários com o passar do tempo, para cálculos heurísticos.

Na figura 16 é mostrado o diagrama de classes dessa implementação, que também é o modelo relacional da base de dados.

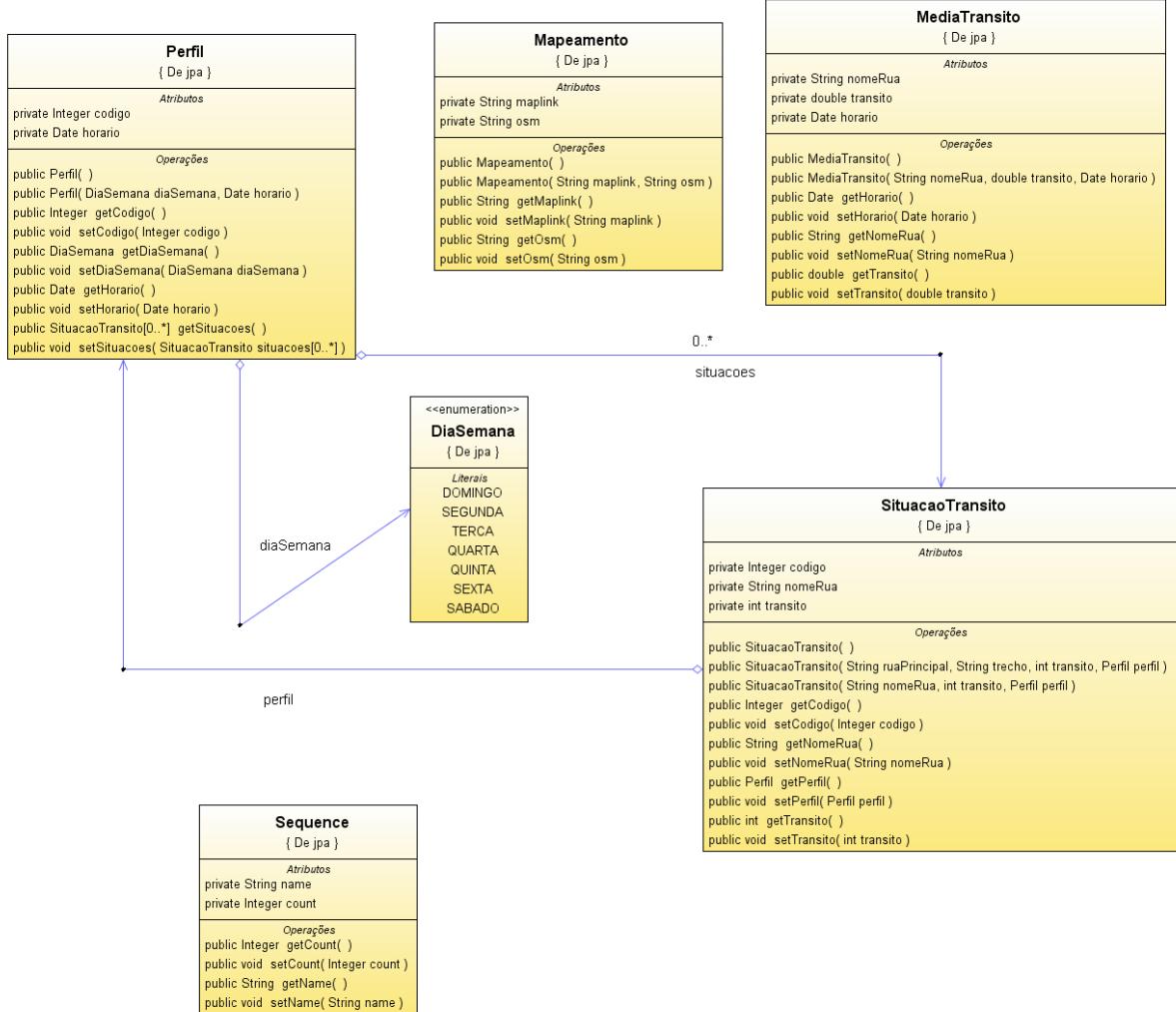


Figura 16: Diagrama de classes do MAPLINK.

A parte interessante desse projeto de extensão, é justamente a base de dados desenvolvida, que juntamente com métodos de *data mining* eficientes, é capaz de relacionar médias de trânsito, com o trânsito atual e determinados trânsitos históricos salvos no banco de dados para prever de forma muito precisa o trânsito de um momento próximo, e gerar uma heurística capaz de desviar desse trânsito.

5.5 SimCet

Devido à dificuldade de contato com a CET (Companhia de Engenharia de Tráfego), foi implementado um simulador de CET. Esse simulador é composto de uma base de dados estatística, contendo diversos perfis, um processador para essa base de dados e um módulo que fornece a comunicação via WebService.

Na figura 17, é apresentada uma visão geral do simulador de trânsito SIMCET.

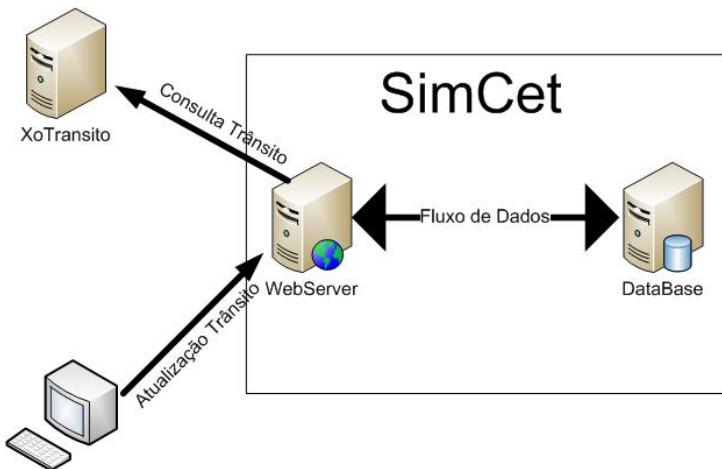


Figura 17: Visão geral do simulador de trânsito SIMCET.

5.5.1 Base de Dados e Perfis

Para a simulação da CET ser o mais próximo possível do real, foi implementada uma base de dados estatísticos que armazenam os dados referentes a certos perfis. Perfis são situações de trânsito salvas pelo próprio usuário. Cada situação de trânsito é associada a um perfil.

Na figura 18, é mostrado o diagrama de classe e o modelo relacional da base de dados estatística do SIMCET.

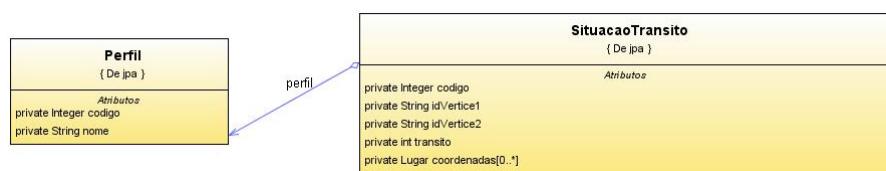


Figura 18: Diagrama de classe e relacional da base de dados de informação de trânsito.

Com base na figura 18, verifica-se que cada situação de trânsito, que corresponde a um conjunto de: código da situação, um número único para garantir unicidade no banco de dados, uma nota para esse trânsito, variando de zero a dez, dois IDs de vértices, que representam os dois pontos que compõem uma aresta, e coordenadas, que são transientes e portanto não persistidos no banco de dados. Cada situação de trânsito também está relacionada a um perfil, ou seja, para cada perfil, podem existir várias situações de trânsito, mas o contrário não é verdade. A situação de trânsito é salva em um banco de dados relacional e recuperada toda vez que se deseja utilizar uma heurística ou situação de testes.

5.5.2 Processador do SimCet

Quanto ao processador do SIMCET, ele se baseia de uma interface Web e WebServices para acesso do XoTRANSITO, e um simulador de um fornecedor de dados reais que não pertence ao mesmo projeto.

A interface Web é componente que permite ao usuário visualizar o trânsito e dar sua própria contribuição para a base de dados, adicionando uma nota para o trânsito de uma ou várias vias. Essa interface Web foi desenvolvida em Java e acessa o banco de dados através de JPA. Sempre que algum usuário altera algum dado relacionado ao trânsito, o SIMCET chama automaticamente um WebService do XoTRANSITO e atualiza a situação de trânsito do mesmo.

Na figura 19, é mostrada a interface Web inicial do SIMCET. Nessa interface temos campos para que o usuário selecione uma rua e de uma nota, assim como veja o trânsito atual ou carregue um perfil salvo no banco de dados.

Na figura 20, é apresentado o SIMCET ao se clicar na aba “Trânsito”. Ela abre um menu no qual é possível escolher as opções de dar uma nota ao trânsito, interpolar, que é calcular a média de trânsito entre dois pontos com trânsito, limpar e mostrar o trânsito atual.

Já na figura 21, clicou-se na aba “Perfis”. Dentro dessa aba se encontram as opções de salvar e carregar um perfil. Como foi discutido anteriormente, um perfil de trânsito é salvo de acordo com um nome dado pelo usuário.

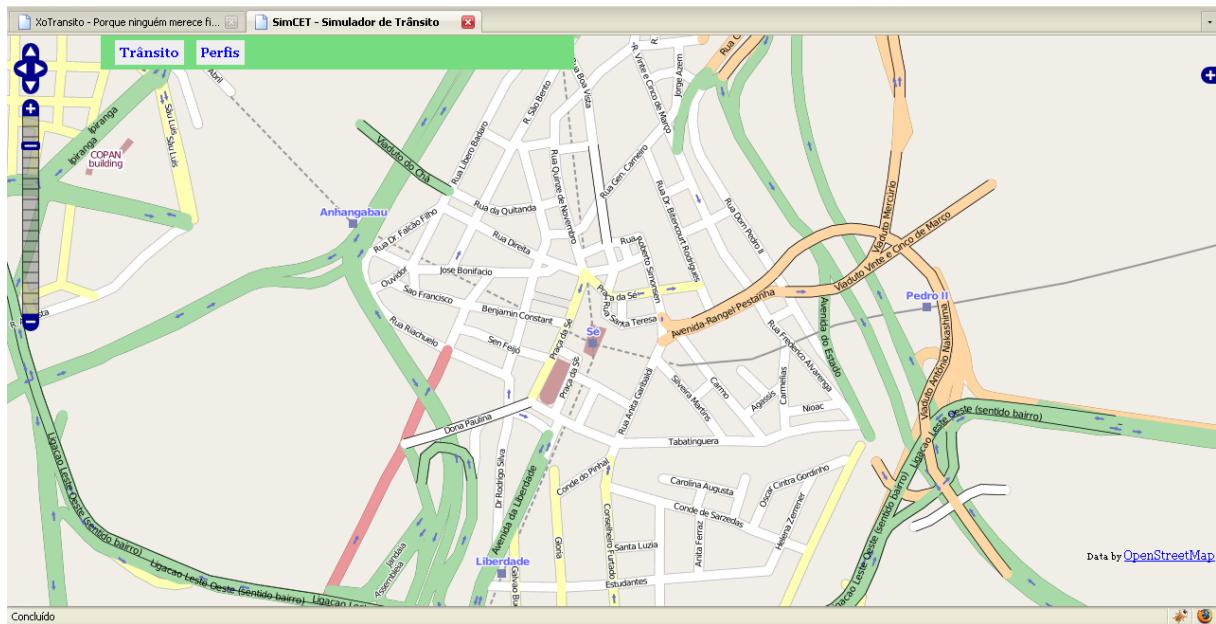


Figura 19: Interface Web inicial do SIMCET.

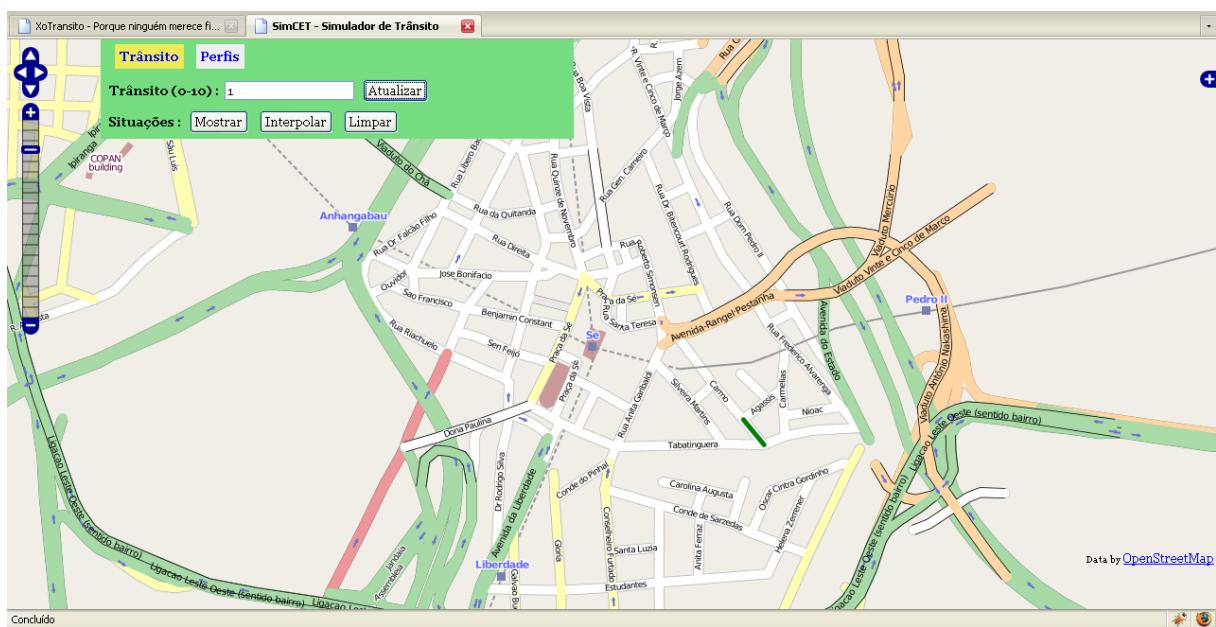


Figura 20: Exemplo aplicado ao SIMCET.

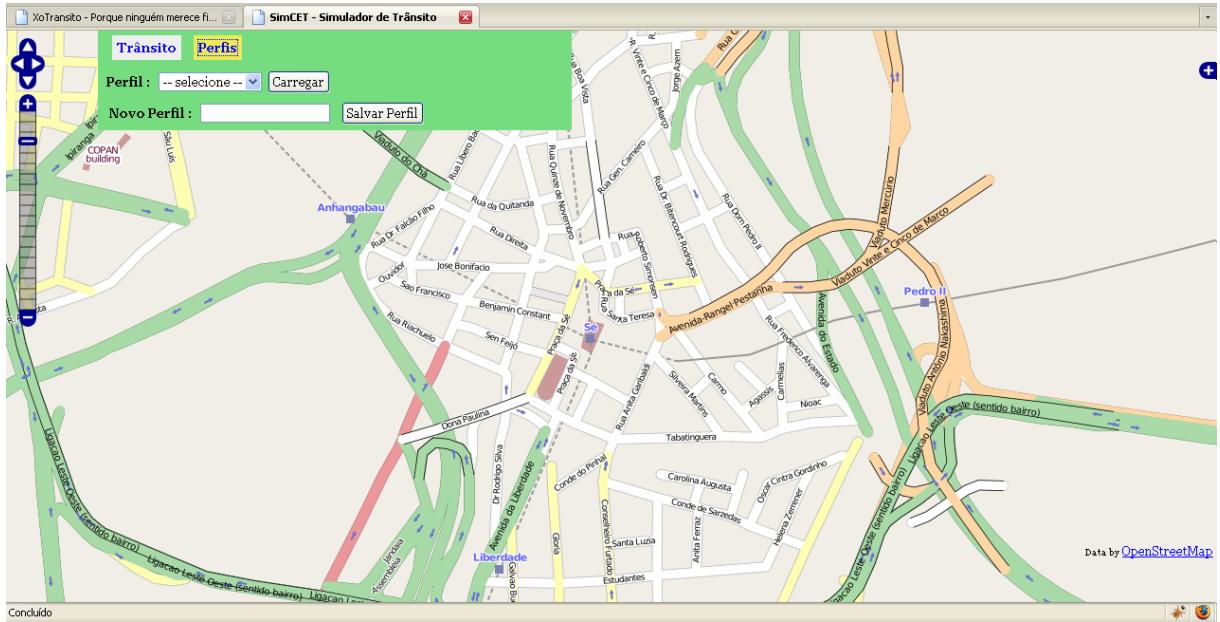


Figura 21: Exemplo aplicado ao SIMCET.

5.5.3 WebServices do SimCet

Dentre os diversos WebServices criados no projeto do SIMCET, abaixo são citados os principais e explicadas suas principais características de funcionamento no contexto do projeto XOTRANSITO.

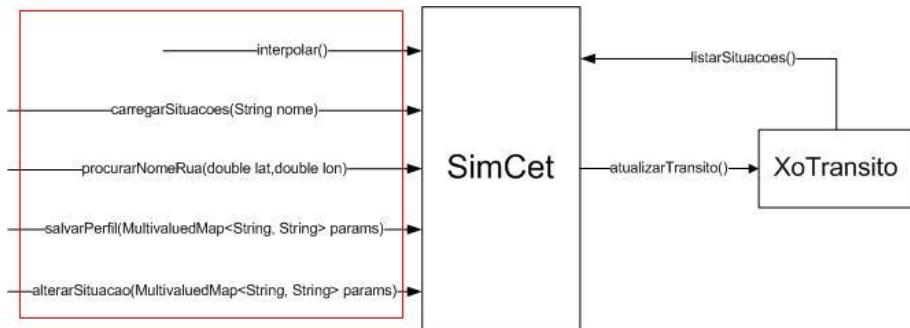


Figura 22: WebServices do SIMCET.

Como pode ser visto na figura 22, existem diversos WebServices que são disponibilizados para a própria interface Web do SIMCET, os quais estão representados do lado esquerdo da figura, dentro de um quadrado em vermelho. Como se pode ver, eles são WebServices que fornecem serviços relacionados ao banco de dados, como perfis e situações de trânsito. Os WebServices `carregarSituacoes` e `alterarSituacao` carregam e alteram, respectivamente, situações de trânsito na base de dados. Já o `salvarPerfil` altera os

perfis da base de dados. Por fim, o `procurarNomeRua` devolve um nome de rua dado uma latitude e uma longitude. O `interpolar` interpola o trânsito entre duas arestas de uma mesma rua que possuem uma nota de trânsito, mas as arestas entre essas têm trânsito desconhecido, a partir disso usamos uma heurística em que entre essas duas arestas o trânsito desconhecido é interpolado, no próprio sentido da palavra, ou seja, é calculada a diferença entre as duas notas conhecidas, essa diferença é dividida pelo número de nós com notas desconhecidas, e a cada nó, é iniciado no menor valor e somado essa fração até o nó de nota conhecida de maior valor. Esses últimos acessam apenas a base de dados de mapa.

O WebService `atualizarTransito` está disponível no XoTRANSITO, mas é intimamente ligado ao SIMCET, pois toda vez que algum usuário altera a situação do trânsito, esse WebService é chamado e faz com que o XoTRANSITO tenha conhecimento dessa mudança.

Para finalizar, o `listarSituacoes` é um WebService disponível para o XoTRANSITO e qualquer outro produto que queira saber a situação atual do trânsito. O mesmo devolve uma lista com o nome das ruas e os trechos sobre os quais se tenha conhecimento do trânsito, representado por uma nota.

5.6 XoTransito

O módulo XoTRANSITO é o componente principal do projeto, tendo como objetivo controlar todos os aspectos do cálculo de rotas, seja o mesmo realizado com ou sem informações de trânsito. Nesse contexto, todos os algoritmos de roteamento estão contidos nesse módulo. Ainda, o módulo implementa uma modelagem específica para o grafo de ruas, conveniente para a aplicação dos algoritmos. Finalmente, o módulo contém a interface Web para a geração de rotas, através da qual o usuário seleciona os pontos de origem e destino desejados.

5.6.1 Classe Grafo

A classe `Grafo` contém dois atributos, que representam o mesmo grafo de duas maneiras:

- `vertices`: Representação estática do grafo de ruas. Na prática, é apenas um conjunto de vértices, que permite encontrar as arestas relacionadas e os pesos refe-

rentes aos atributos de comprimento, velocidade e nível, ou seja, aqueles que não variam durante a execução.

- **transitos**: Representação da parte dinâmica do grafo de ruas. Na prática, é um mapeamento de cada aresta do grafo às respectivas informações de trânsito, que podem ser alteradas durante a execução do algoritmo.

Os métodos mais importantes da classe são descritos abaixo:

- **incluirVertice**: Inclui o vértice passado como parâmetro na coleção **transitos**.
- **incluirAresta**: Inclui a aresta passada como parâmetro (representada através dos dois vértices que a definem) na coleção **transitos**. São passados alguns dados da aresta, incluindo a rua a que ela pertence.
- **melhorCaminho**: Calcula o melhor caminho chamando os métodos da classe **AEstrela**. Além disso, passa alguns parâmetros para o algoritmo de roteamento (mais detalhes na explicação da classe **AEstrela**).
- **verticeMaisProximo**: Encontra o vértice do grafo mais próximo do ponto que possui a latitude e a longitude passadas como parâmetro. Esse método é utilizado para interpretar os cliques do usuário na interface Web.
- **atualizarTransito**: Atualiza o trânsito da aresta iniciada no vértice de **idVertice1** e finalizada no vértice de **idVertice2**. O trânsito passado como parâmetro varia de 0 (sem congestionamento) a 10 (congestionamento máximo).

5.6.2 Classe AEstrela

A classe **AEstrela** é responsável por implementar o algoritmo A* e suas diversas variações. Para tanto, a classe dispõe de uma lista de parâmetros, os quais podem ser ativados ou não pelo cliente. Por exemplo, pode-se escolher ou não a ativação dos parâmetros **BIDIRECIONAL** e **ESTRATIFICADO**, o que produziria variantes diferentes do algoritmo.

Além disso, a classe contém uma representação interna de vértice (classe **AEstrelaVertice**), a qual inclui todos os parâmetros necessários para a execução do algoritmo, incluindo o vértice anterior, o custo real do vértice ($f(n)$), o custo estimado do vértice ($h(n)$) e o nível de estratificação da aresta entre **anterior** e o vértice atual:

```

private class AEstrelaVertice {
    private Vertice vertice;
    private AEstrelaVertice anterior;
    private double custo;
    private double estimativa;
    private Nivel nivel;
}

```

O método mais importante dessa classe, entretanto, é a própria implementação do algoritmo A*, a qual foi realizada de modo a se comportar de forma diferente dependendo da lista de parâmetros passada. A seguir, é apresentada a lógica de alto nível do algoritmo, incluindo trechos de pseudocódigo.

No nível mais genérico, o comportamento do algoritmo geral é o seguinte:

- 1 Se a lista aberta não está vazia:
- 2 Visita o primeiro vértice da lista aberta;
- 3 Verifica se o caminho foi encontrado;
- 4 Se o caminho foi encontrado:
 - 5 Reconstrói o caminho encontrado;
 - 6 Retorna o caminho reconstruído;
- 7 Se a lista fechada não contém o vértice visitado:
 - 8 Adiciona o vértice visitado à lista fechada;
 - 9 Para cada aresta que contém o vértice visitado:
 - 10 Analisa o vértice que completa a aresta;
 - 11 Se esse vértice não estiver na lista fechada OU se sua função f for menor do que a já existente na lista fechada:
 - 12 Adiciona ou substitui o vértice na lista aberta;

Notas sobre o algoritmo:

- **Linha 3.** No A* original, encontrar o caminho significa que o nó sendo visitado é o destino. No A* bidirecional, encontrar o caminho significa que o nó sendo visitado já está contido na lista fechada do outro sentido de percurso.
- **Linha 5.** No A* original, reconstruir o caminho significa percorrer todas as referências ao nó anterior a partir do nó último nó visitado (destino). No A* bidirecional,

deve-se levar em conta que existem duas listas abertas (uma partindo da origem e outra partindo do destino) e, portanto, o caminho é reconstruído a partir do vértice de encontro das instâncias em direção a cada um dos extremos.

- **Linha 9.** Em todos os casos (exceto bidirecional no sentido inverso), consideram-se apenas as arestas que partem do vértice visitado. No caso estratificado, somente as arestas do nível atual são selecionadas.
- **Linha 12.** Nesse ponto, os novos atributos do vértice analisado são calculados: f , h , a referência para o vértice anterior e o nível de estratificação.

O A* original se baseia, principalmente, na função de avaliação f , resultante da soma dos pesos até o vértice atual e da função heurística que estima o peso do percurso do vértice atual até o destino. A cada momento, o vértice visitado é aquele que apresenta maiores chances de compor o melhor caminho, uma vez que a lista de prioridades utilizada é ordenada pela função f . Uma vez que existem muitas possibilidades de rota, muitas delas com custos muito parecidos, o algoritmo acaba por expandir muitos vértices. Tal problema é a principal motivação de pesquisa por métodos mais eficazes no projeto XOTRANSITO.

Uma primeira modificação introduzida no algoritmo foi a inclusão do conceito de bidirecionalidade: ao invés de o algoritmo se iniciar da origem e processar até o destino, pode-se construí-lo de forma a realizar, alternativamente, os mesmos passos, porém, começando da origem e também do destino. Portanto, o algoritmo se encerra encontrando um ponto comum entre os dois trechos de rotas processados. Essa alteração resultou na duplicação das listas, tanto a aberta como a fechada. Além disso, o algoritmo processa um passo partindo da origem, outro do destino, outro da origem e assim por diante, alternativamente. O caminho, desse modo, é construído com base nas informações das duas listas fechadas, observando-se o sentido do percurso (na lista fechada iniciada da origem, o caminho é reconstruído do meio até o começo e na lista fechada iniciada do destino, do meio até o fim).

A segunda alteração inserida no algoritmo A* original foi a inclusão do conceito de estratificação: as várias ruas existentes da cidade podem pertencer a diferentes níveis de detalhe. Assim, em certos momentos do algoritmo, são desconsideradas certas ruas para diminuir o espaço de busca do algoritmo. A modificação propriamente dita é realizada no projeto foi a alteração do modo como a lista de arestas do vértice visitado é montada: somente são retornadas arestas correspondentes ao nível atual do algoritmo. Utilizando a bidirecionalidade, na parte inicial do algoritmo, são consideradas todas as ruas. Porém,

quando ruas de um menor nível de detalhamento são atingidas (por exemplo, avenidas e vias de trânsito rápido), o nível é alterado e o algoritmo passa a considerar uma menor quantidade de vias. A seguir, é explicado como são calculados os pesos das arestas para o algoritmo de roteamento calcular o custo de cada percurso.

5.6.3 WebServices do XoTransito

Como explicado, toda a comunicação entre os módulos XoTRANSITO e SIMCET é feita através de WebServices. Além disso, os eventos JavaScript das telas (causados por ações de usuários) fazem requisições aos WebServices.

A seguir, são descritos os principais WebServices do módulo XoTRANSITO:

- **atualizarTransito:** Atualiza as informações de trânsito do grafo do XoTRANSITO a partir de informações do SIMCET.
- **tracarRota:** Traça uma rota, dados os parâmetros de origem e destino. Também recebe os parâmetros ESTRATIFICADO e BIDIRECIONAL, que permitem a escolha de uma variante do algoritmo.
- **login:** Recebe os dados cadastrais do usuário (*login* e senha) e verifica se as informações fornecidas estão cadastrados no banco de usuários. Em caso positivo, realiza a autenticação do usuário.
- **listar:** Dado um usuário, lista os pontos de interesse cadastrados pelo mesmo.
- **incluir:** Inclui um novo ponto de interesse para o usuário autenticado.

5.6.4 Peso das Arestas

O peso das arestas foi modelado de forma a considerar diversos fatores que influenciam no tempo gasto para percorrer um determinado trecho da cidade, através de suas ruas e avenidas. Neste projeto, foram considerados, como parâmetros de avaliação dos pesos das arestas, os seguintes itens:

- **Comprimento**, medido em quilômetros: constitui-se no peso inicial de todo grafo que modele um conjunto de vias e seja objeto do traçado de rotas. O valor do peso, dado em quilômetros, é calculado no carregamento do sistema através do algoritmo

de cálculo de distância entre dois pontos geográficos (definidos por uma latitude e uma longitude), explicado anteriormente.

- **Velocidade**, medida em km/h: fator importante para o sistema, uma vez que existe uma grande variação das velocidades máximas entre as diversas vias da cidade. Segundo o código de trânsito brasileiro, a velocidade máxima, onde não existir sinalização, é de: 30 km/h para vias locais, 40 km/h para vias coletoras, 60 km/h para vias arteriais e 80 km/h para vias de trânsito rápido.
- **Trânsito**, medida de 0 a 10: esse fator refere-se à situação de trânsito aferido na aresta referente. Pode assumir valores de 0 (totalmente livre) até 10 (totalmente congestionado), indicando o nível de congestionamento apurado no local.

Notar que os itens acima são apenas algumas das possibilidades de fatores a serem considerados. Outros itens, tais como segurança e inclinação de rua, poderiam ser adicionados sem prejuízo da lógica do algoritmo, com o intuito de melhorar ainda mais a modelagem dos fatores relevantes na situação real.

Os fatores citados (comprimento, velocidade e trânsito) são agrupados em um peso único usando a equação abaixo:

$$\text{peso} = \text{comprimento} \left(\frac{1}{\text{velocidade}} \right) \left(1 + \frac{\text{transito}}{10} \right) \quad (5.1)$$

A utilização dessa fórmula auxilia no cômputo do tempo necessário para percorrer a aresta em questão. O peso no final, que possui como unidade de medida, a hora, representa, consequentemente, o tempo necessário para passar por esse trecho, considerando-se um automóvel com uma velocidade dada pela junção dos efeitos da velocidade máxima e trânsito do local.

6 Testes e Avaliação

A seguir, encontram-se alguns testes práticos do projeto. Deve-se notar, entretanto, que os testes não têm relevância estatísticas, tendo como propósito apenas fornecer uma idéia das diferenças de desempenho entre os algoritmos.

6.1 Algoritmos

Para a comparação da eficiência do algoritmo adaptativo implementado no projeto foram realizados testes de roteamento entre duas localizações da cidade de São Paulo. Nesse conjunto de testes, foi analisado o desempenho (em relação ao número de nós visitados e ao tempo de processamento despendido) do algoritmo adaptativo em relação a outros dois algoritmos clássicos: o algoritmo A* original e o A* bidirecional.

Num primeiro conjunto de testes foram analisados os desempenhos desses algoritmos para distâncias consideradas curtas (de 4 a 50 quarteirões de distância) entre a origem e o destino. No segundo conjunto de testes foram analisados os desempenhos destes algoritmos para distâncias consideradas médias (de 50 a 100 quarteirões entre a origem e o destino).

Na sessão que segue, será demonstrado o comportamento dos algoritmos para um entendimento melhor por parte do leitor das diferenças entre eles, e em seguida serão feitos os testes descritos acima.

O conjunto completo dos resultados obtidos nos testes realizados se encontram no anexo B.

6.1.1 Demonstração do Algoritmo

Durante a implementação do algoritmo, foi pensado em se fazer um modo *debug* em que seja possível visualizar os passos do algoritmo, e também sua árvore de expansão de

nós. Esse modelo foi implementado, e abaixo é mostrada uma série de figuras, representando passos intermediários da animação, a fim de demonstrar o comportamento dos algoritmos.

6.1.1.1 Algoritmo A* Convencional

Para começar a demonstração, é utilizado o algoritmo A* tradicional, que não possui nenhuma das melhorias implementadas pelo grupo. As figuras que seguem são a demonstração do modo *debug* desse algoritmo original.

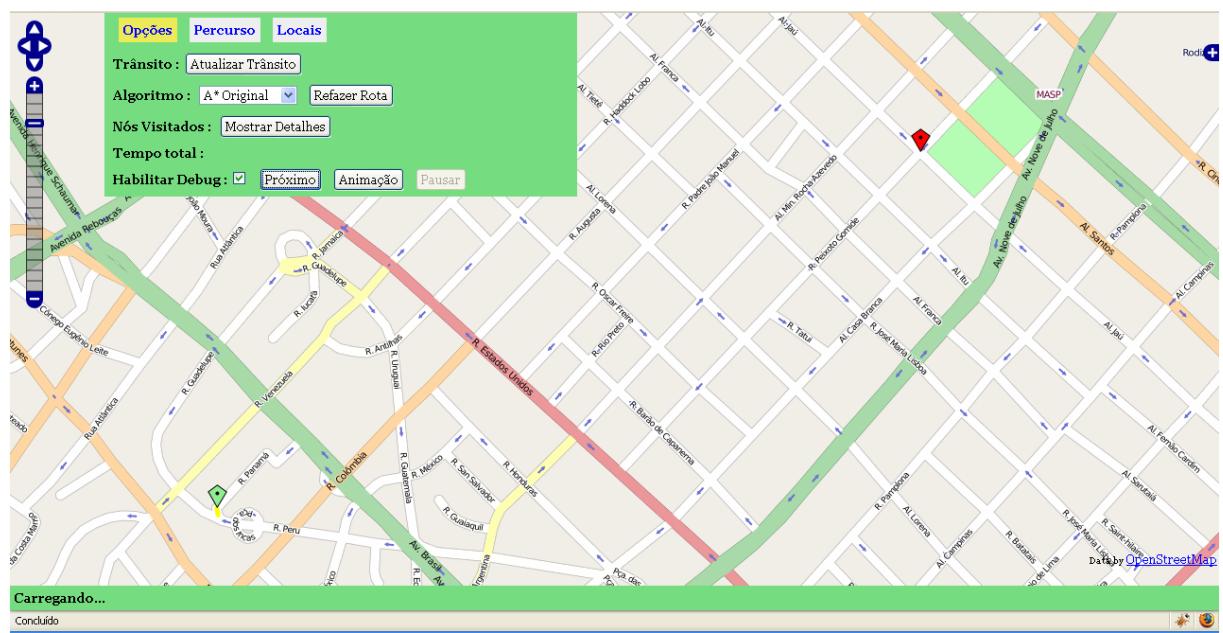


Figura 23: Modo inicial do algoritmo original.

Como pode ser visto nas figuras 23 a 27, a árvore de nós se expandiu além do esperado pelo grupo, o que os levou a tomar a decisão de implementar o algoritmo que segue, com conceitos de bidirecionalidade.

6.1.1.2 Algoritmo A* Bidirecional

O algoritmo bidirecional, como explicado em 2.1.2, ele parte da origem e destino simultaneamente, se encontrando em um ponto no intermédio do caminho. A idéia de utilizar esse algoritmo visa diminuir a árvore de nós expandidos.

Abaixo, seguem as figuras do percurso do algoritmo bidirecional.

Como é visualizado nas figuras 28 a 31, a meta do grupo de diminuir os nós visitados

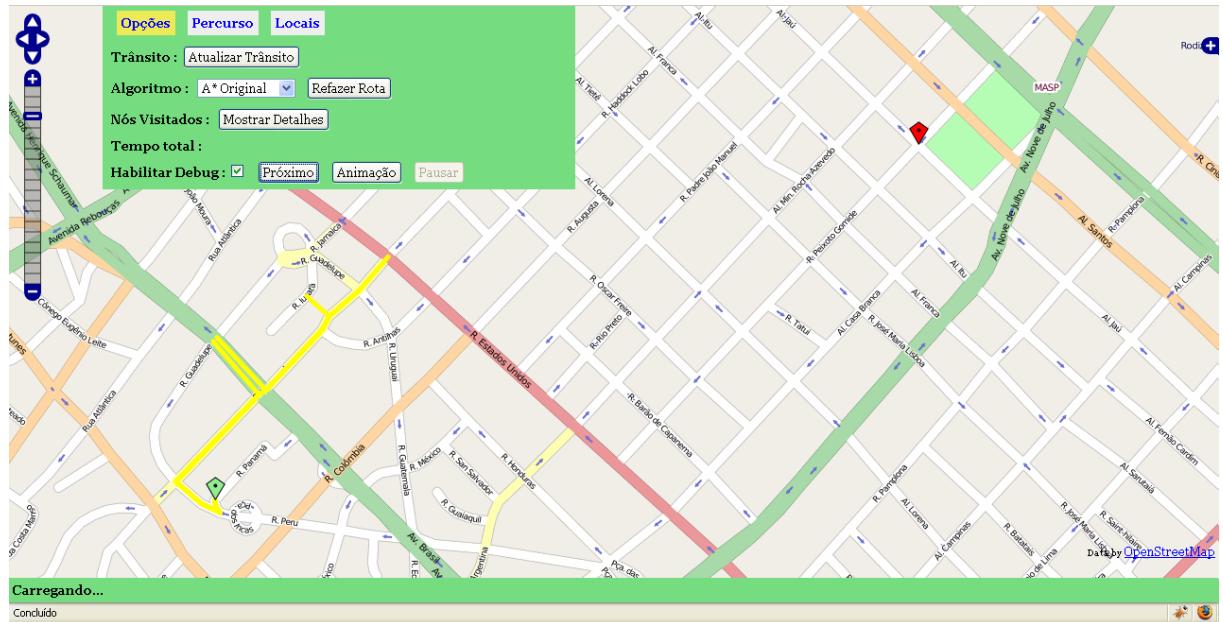


Figura 24: Primeiros passos do algoritmo original.

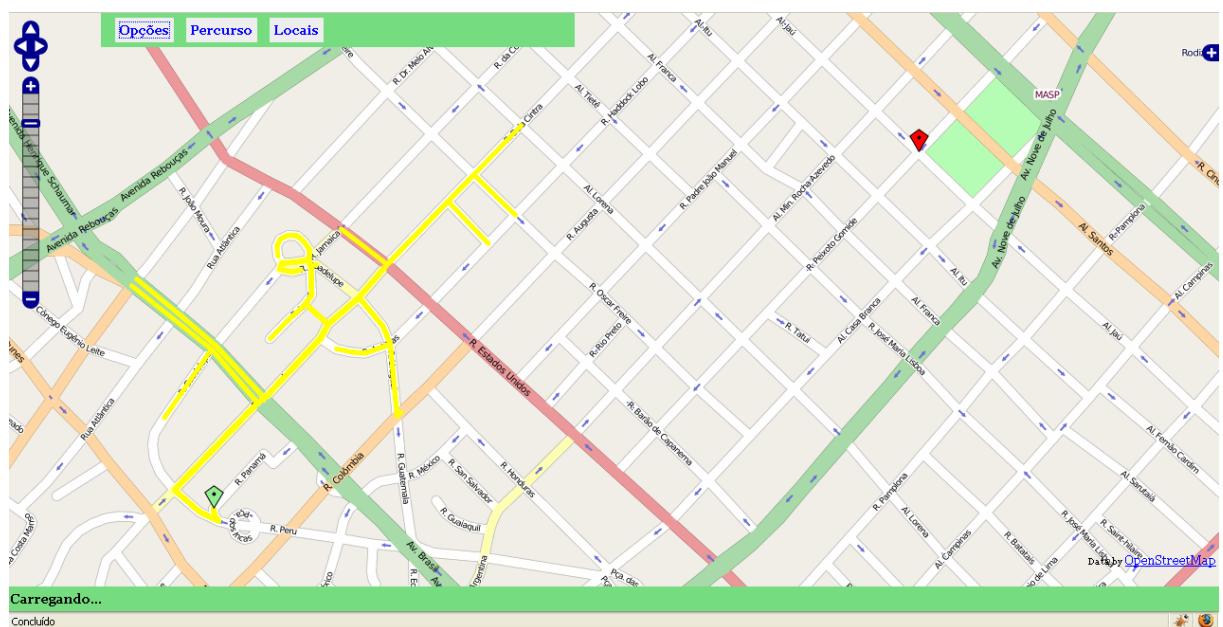


Figura 25: Passos intermediários do algoritmo original.

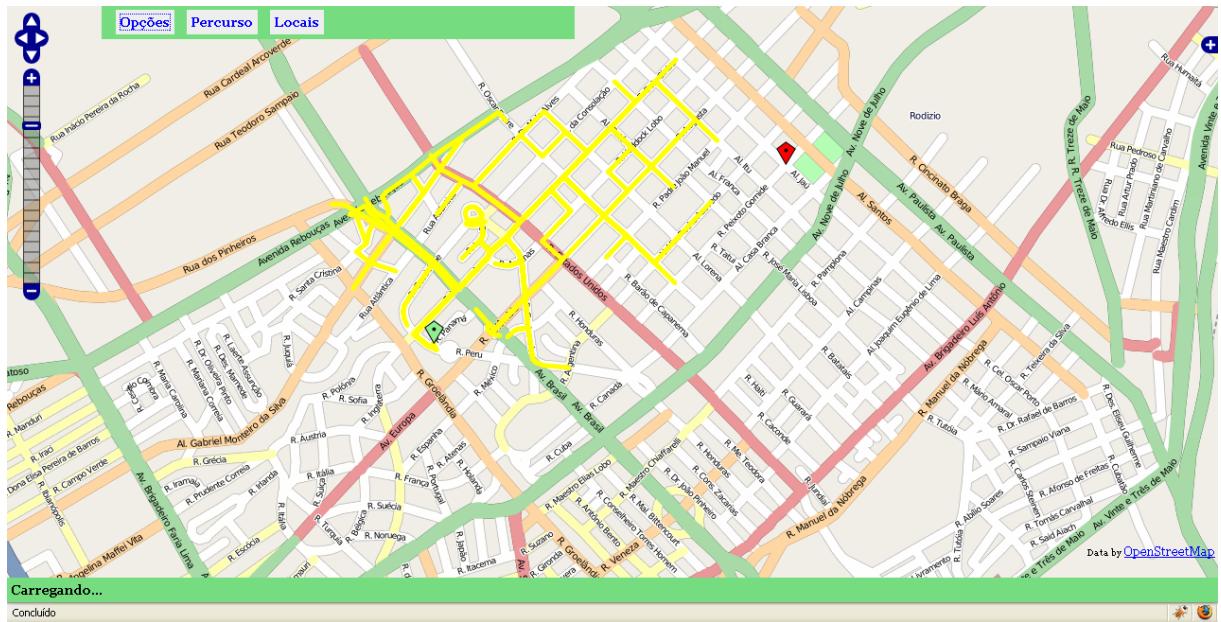


Figura 26: Passos semi-finais do algoritmo original.

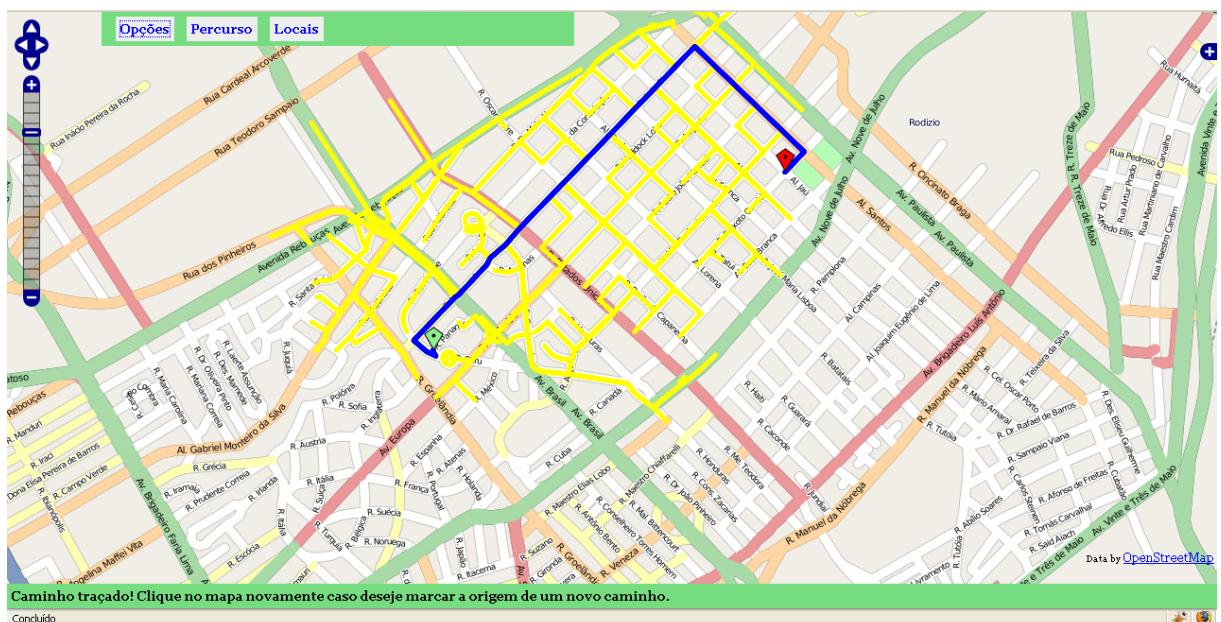


Figura 27: Algoritmo original finalizado.

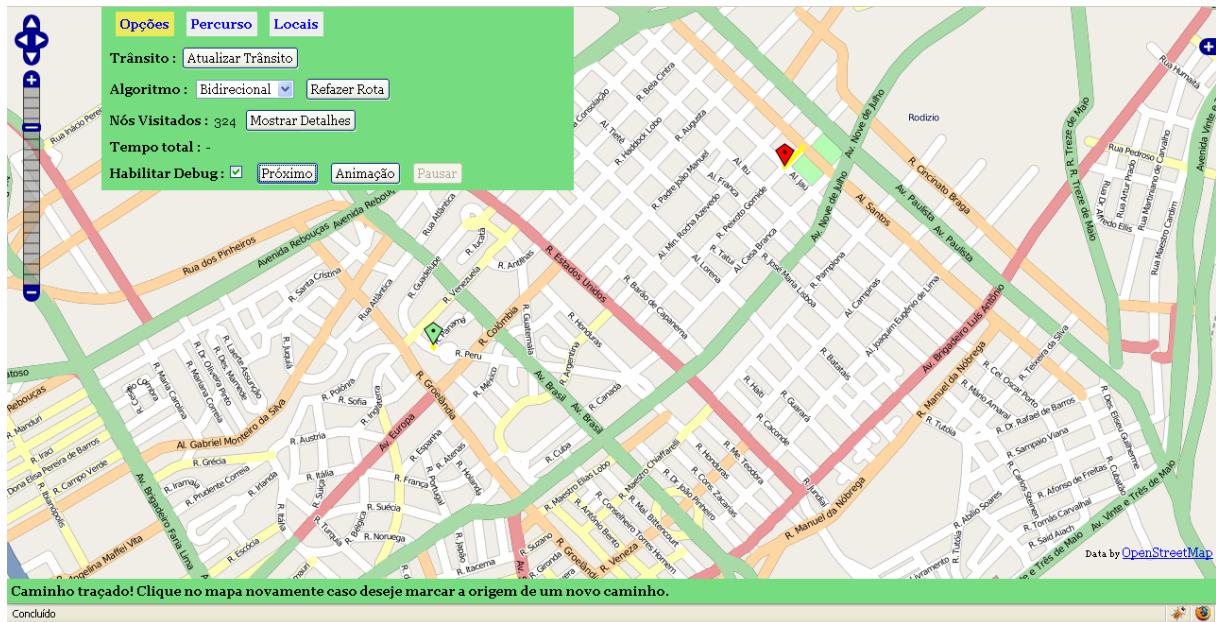


Figura 28: Algoritmo bidirecional inicializando.

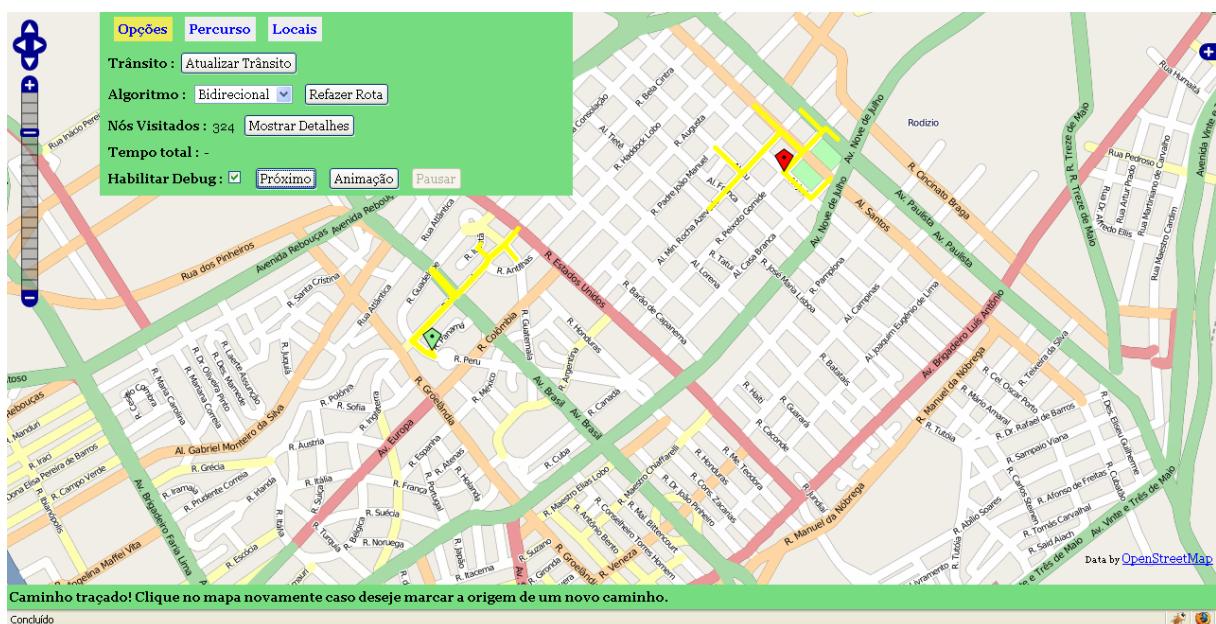


Figura 29: Primeiros passos do algoritmo bidirecional.

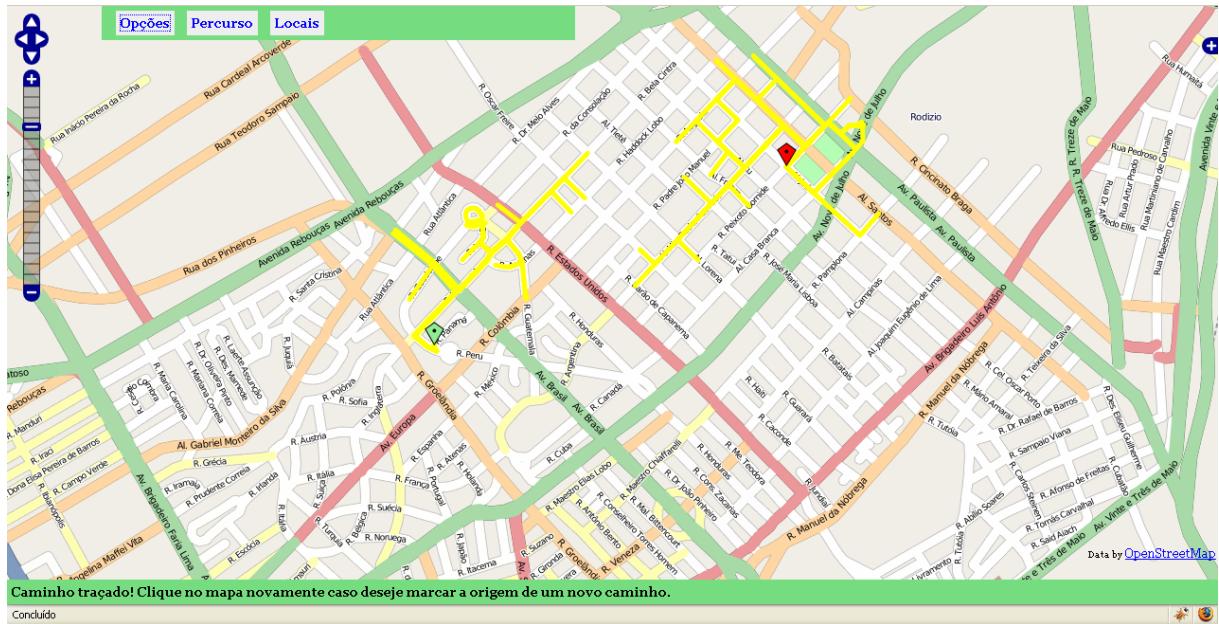


Figura 30: Passos semi-finais do algoritmo bidirecional.

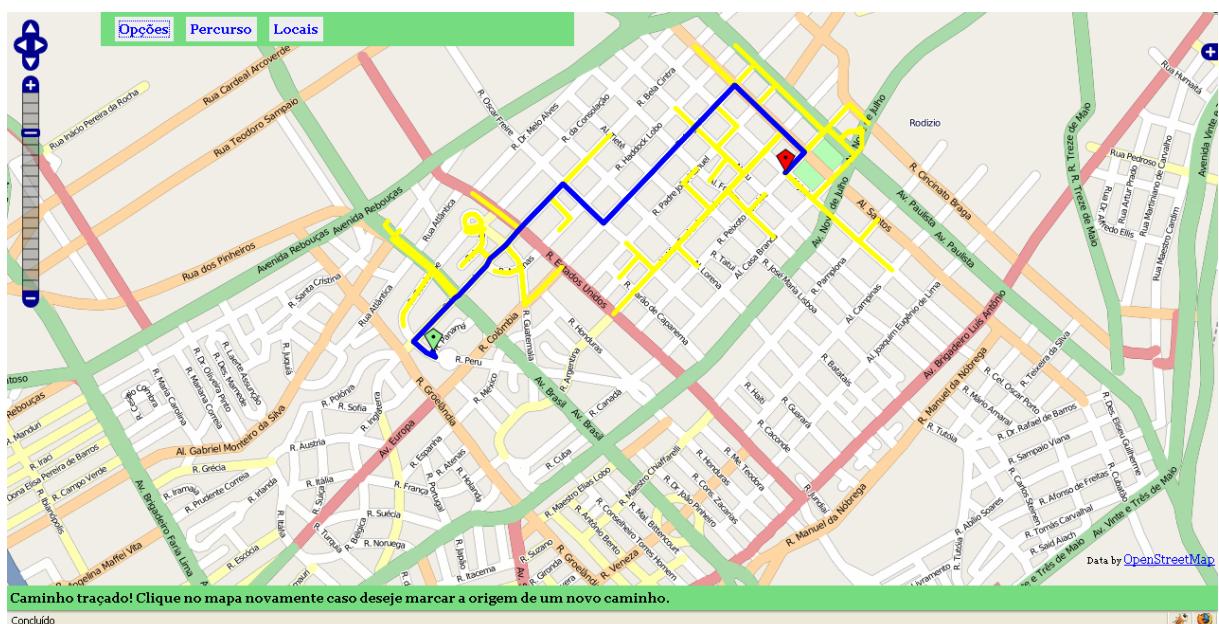


Figura 31: Finalização do algoritmo bidirecional.

foi atingida com sucesso. Mesmo assim, foi decidido implementar um algoritmo capaz de melhorar ainda mais essa quantidade de nós visitados, e ele é explicado na próxima sessão.

6.1.1.3 Algoritmo A* Bidirecional e Estratificado

Para melhorar o desempenho e o número de nós visitados pelo algoritmo, e ainda incluir de forma natural a adaptatividade no projeto proposto, foi desenvolvido o algoritmo bidirecional estratificado. Como explicado em 2.2.1, esse algoritmo parte da origem e destino ao mesmo tempo, em instâncias diferentes de processamento, e ainda considera os níveis das vias, sendo possível de alterar o grafo a qualquer momento.

Abaixo, seguem figuras dos passos do algoritmo estratificado.

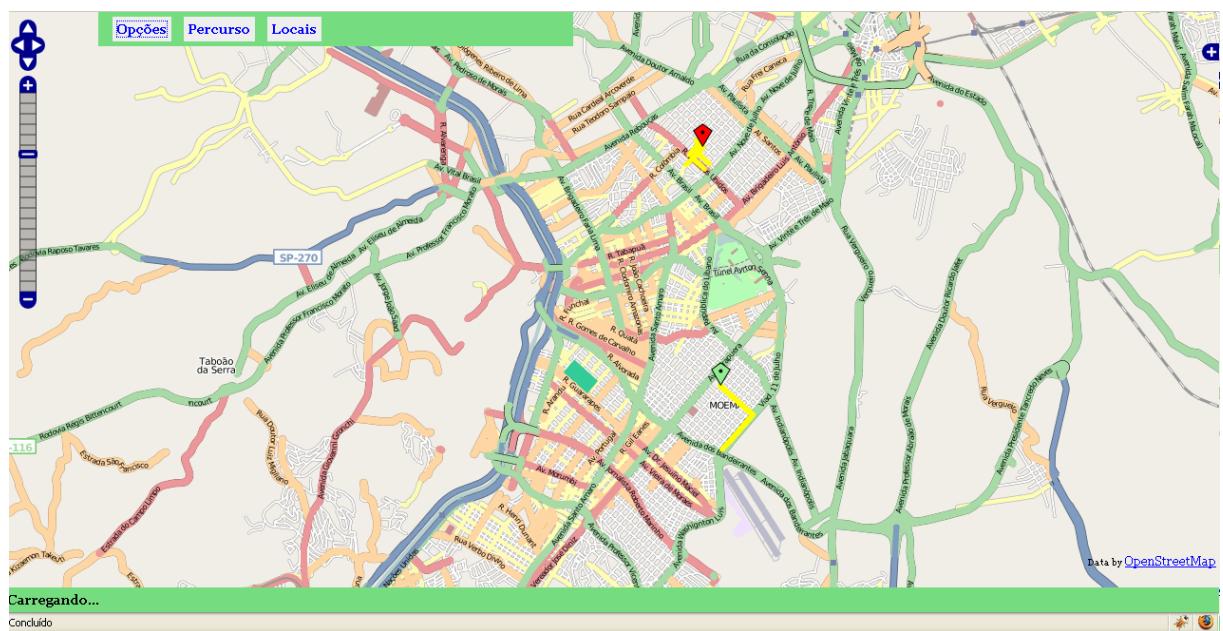


Figura 32: Inicialização do algoritmo estratificado.

Como foi visto nessa execução, percebe-se nas figuras 32 e 36, que o algoritmo, ao chegar a uma avenida de nível mais alto, passa a ignorar as ruas de nível inferior, o que resulta uma árvore de visitação de nós muito menor e um desempenho muito maior. Para fins de comparação, foi colocada uma instância do algoritmo A* original executando o mesmo trajeto na figura 37.

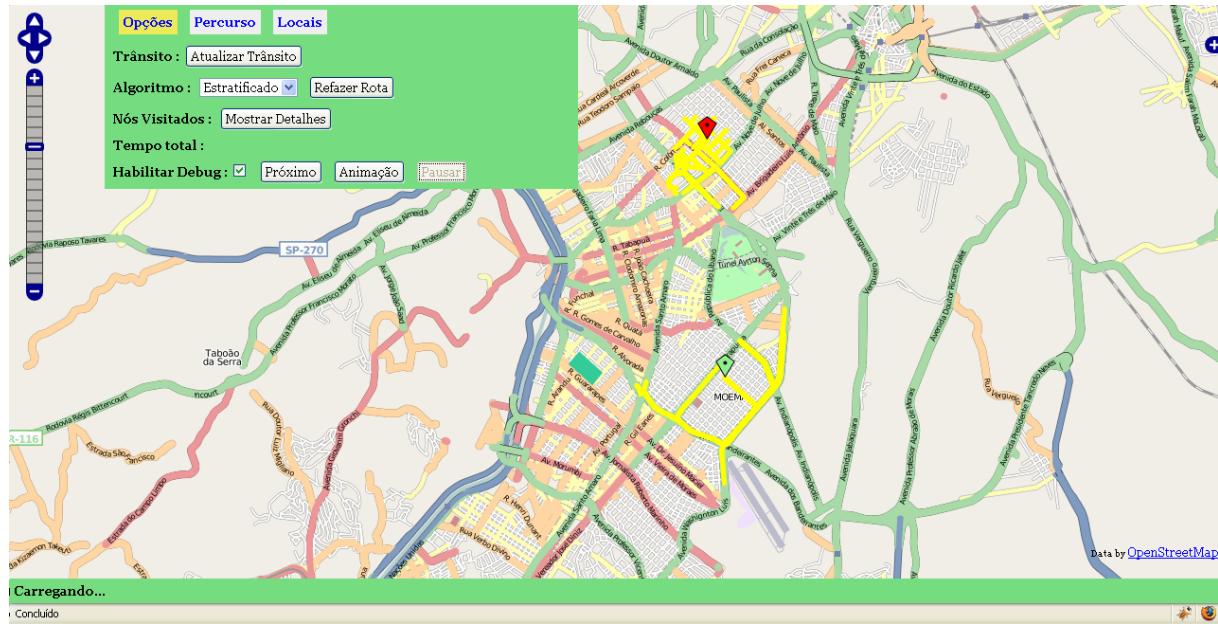


Figura 33: Primeiros passos do algoritmo estratificado.

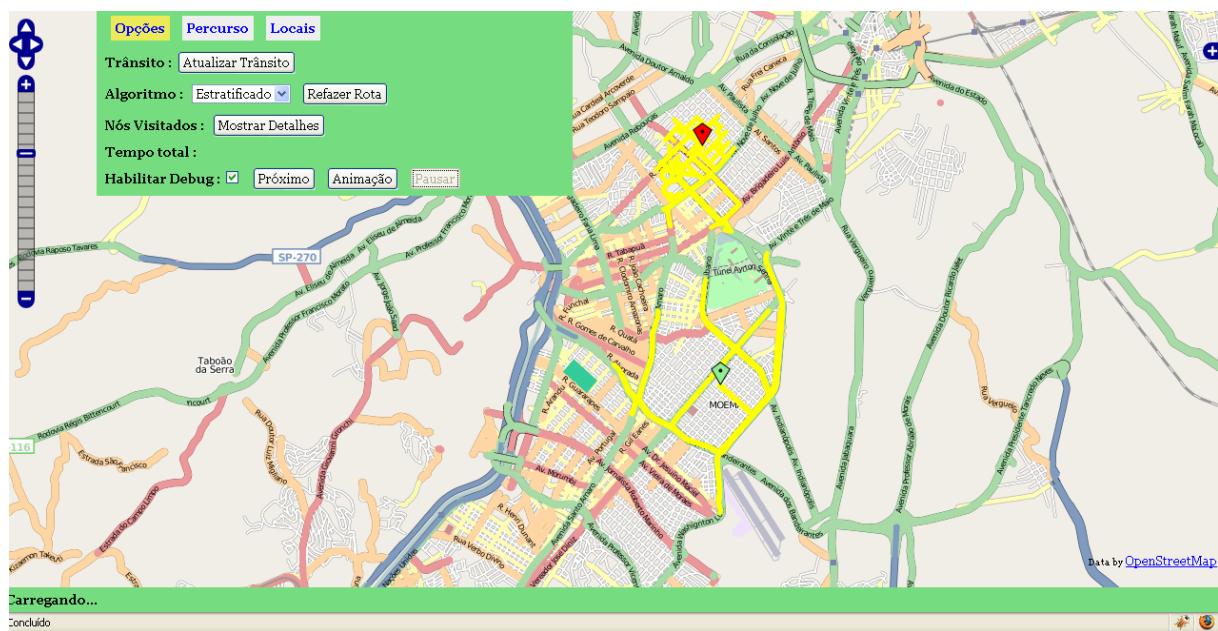


Figura 34: Passos intermediários do algoritmo estratificado.

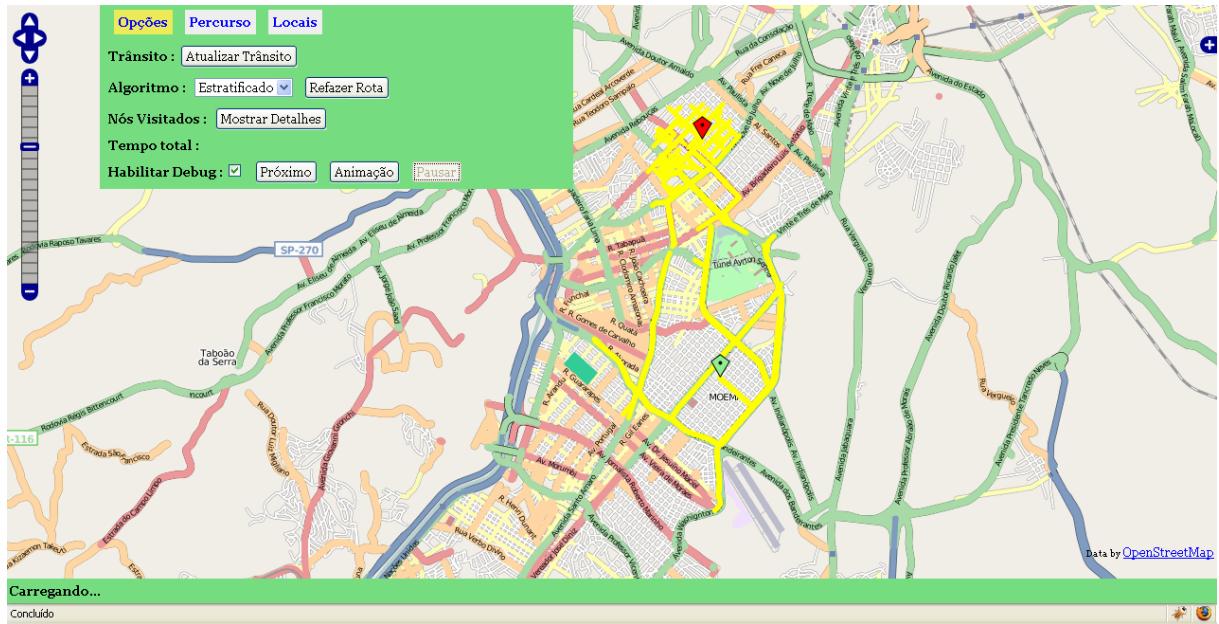


Figura 35: Passos semi-finais do algoritmo estratificado.

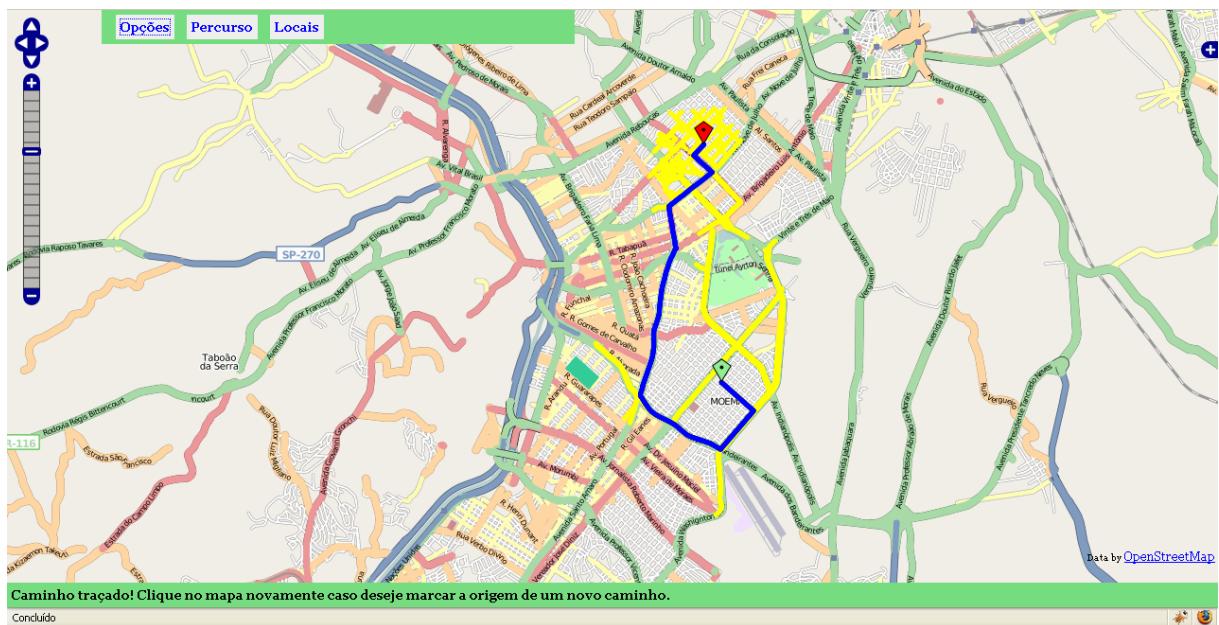


Figura 36: Final do algoritmo estratificado.

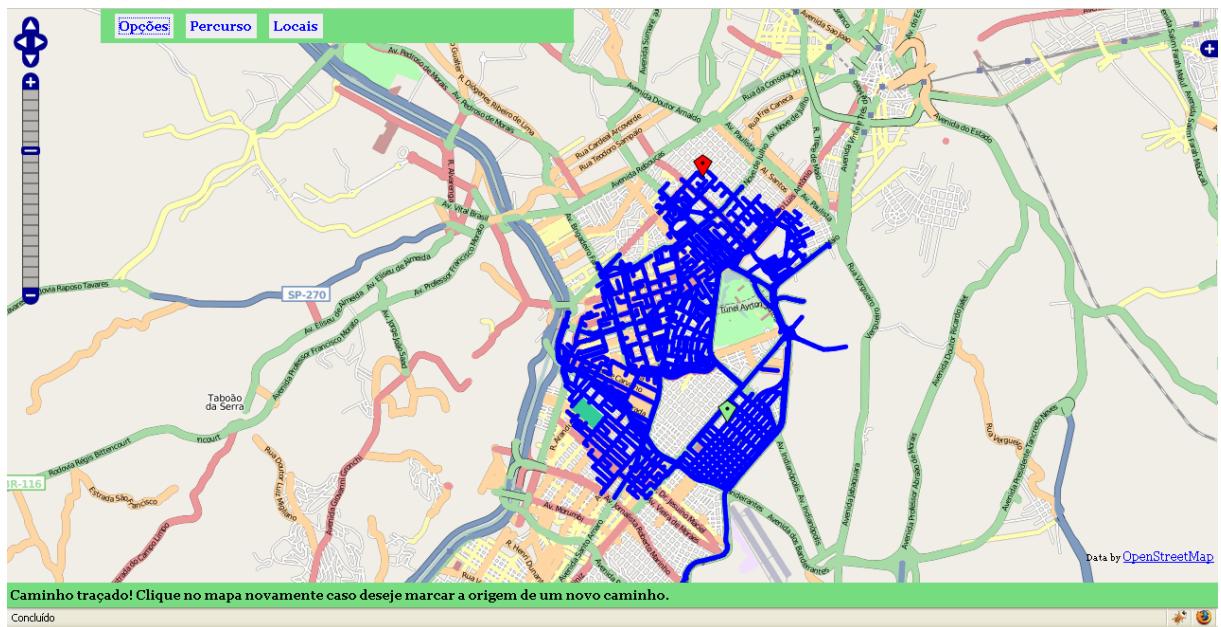


Figura 37: Comparação entre o algoritmo estratificado e o original. Nessa figura, é apresentada a expansão do original.

6.1.2 Incorporação do Trânsito

A seguir, é mostrada a execução do algoritmo incorporando informações de trânsito, que são representadas pelas cores de um semáforo tradicional: verde para bom, amarelo para razoável e vermelho para congestionado. A rota gerada pelo projeto demonstra a preferência do algoritmo de roteamento em percorrer caminhos com um bom trânsito:

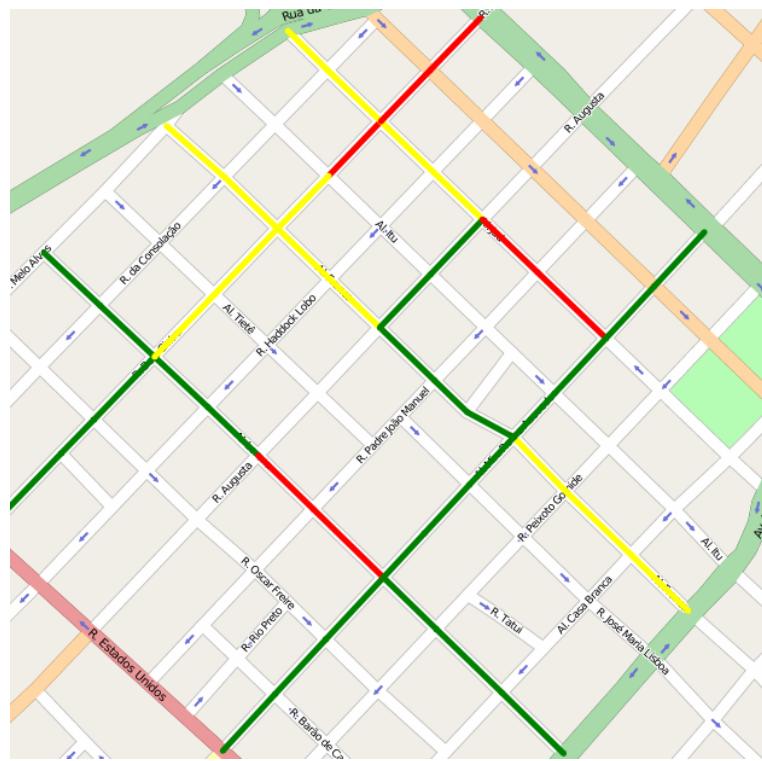


Figura 38: Informações de trânsito no mapa.

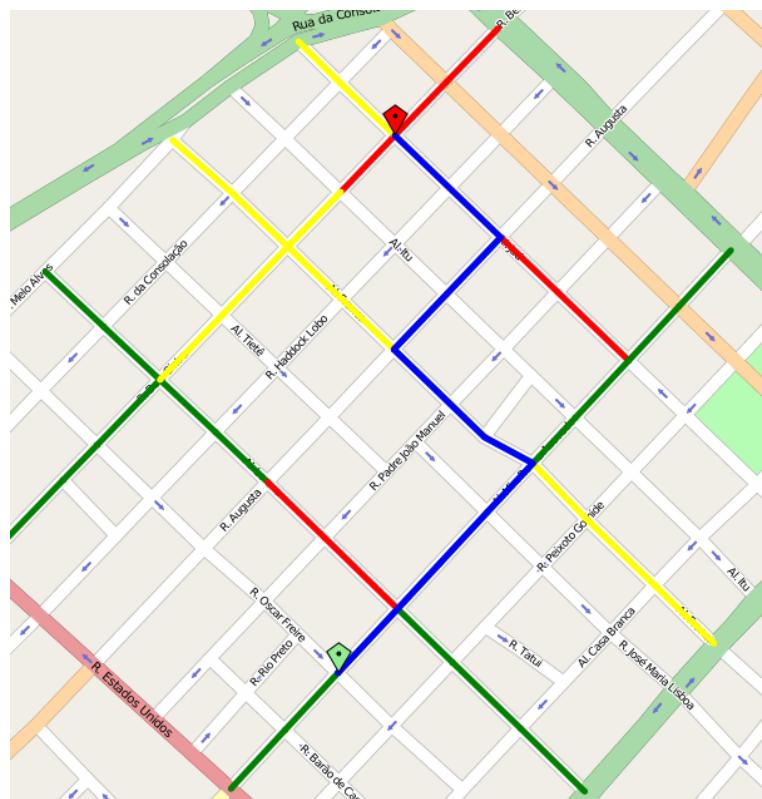


Figura 39: Rota traçada com base no trânsito.

6.1.3 Primeiro Conjunto de Testes

No primeiro conjunto de testes foi traçado uma rota entre duas ruas próximas da cidade de São Paulo. As ruas utilizadas nesse teste foram a Rua Darwin e a Rua Farrapos conforme indicado na figura 40. Cada um dos algoritmos implementados no projeto foi utilizado para traçar uma rota entre esses dois pontos. Nas seções a seguir são comentadas as rotas traçadas para cada um dos algoritmos e numa última seção é analisado o desempenho obtido pelos algoritmos. Conforme indicado nos resultados apresentados em anexo as ruas pelas quais os algoritmos roteiam foram as mesmas, sendo que apenas ocorreu diferenciação no desempenho (em relação ao número de nós utilizados e ao tempo decorrido) dos algoritmos.

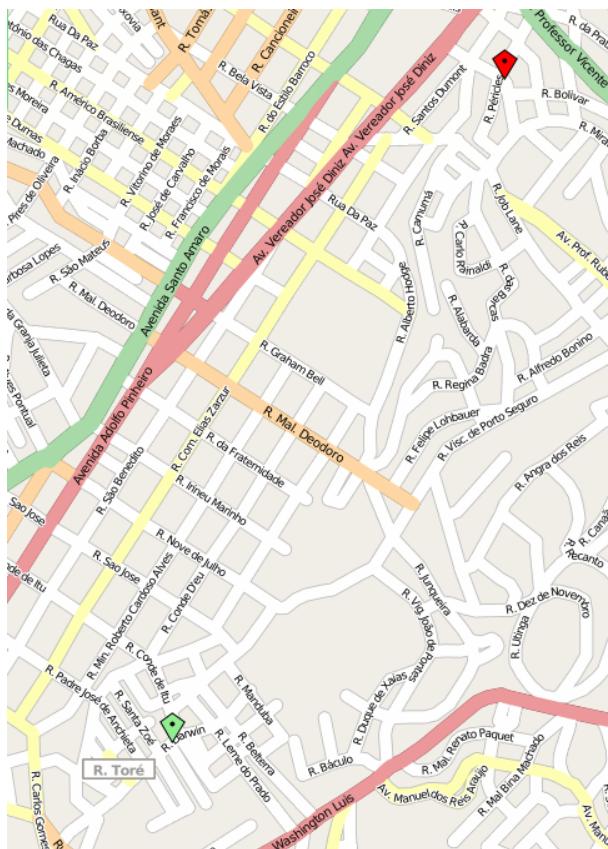


Figura 40: Pontos de origem e destino do teste 1.

6.1.3.1 Algoritmo A*

As figuras 41 e 42 ilustram respectivamente o caminho traçado e a árvore de expansão percorrida pelo algoritmo A* original. Conforme se pode observar da análise da figura 42 o algoritmo tendeu a realizar uma grande expansão na árvore de busca, muitas vezes indo para caminhos que não possuem utilidade para alcançar o objetivo desejado.



Figura 41: Rota traçada para o teste 1 usando o algoritmo A*.

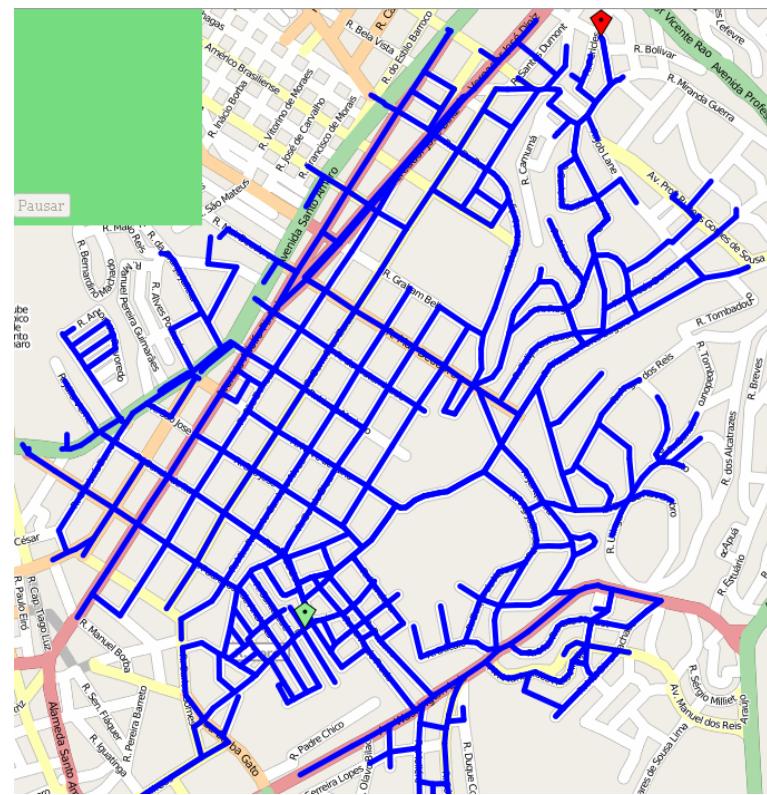


Figura 42: Detalhes do caminho traçado para o teste 1 usando o algoritmo A*.

6.1.3.2 Algoritmo A* Bidirecional

As figuras 43 e 44 ilustram respectivamente o caminho traçado e a árvore de expansão percorrida pelo algoritmo A* bidirecional. Conforme se pode observar da análise da figura 44 o algoritmo A* bidirecional tendeu a sofrer uma menor expansão na árvore de busca, conforme era esperado devido à utilização de dois algoritmos A* originais percorrendo a árvore de busca em sentido inversos.

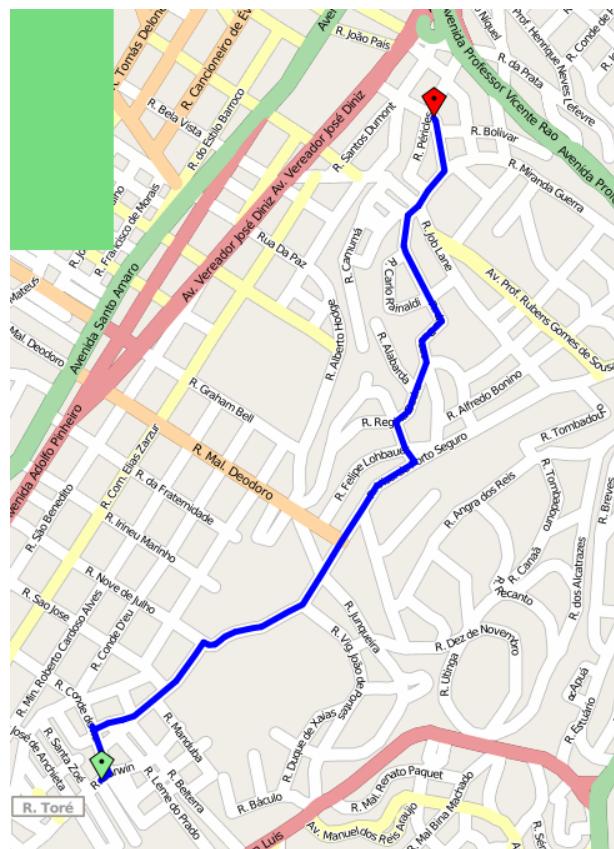


Figura 43: Rota traçada para o teste 1 usando o algoritmo A* bidirecional.



Figura 44: Detalhes do caminho traçado para o teste 1 usando o algoritmo A* bidirecional.

6.1.3.3 Algoritmo Adaptativo

As figuras 45 e 46 ilustram, respectivamente, o caminho traçado e a árvore de expansão percorrida pelo algoritmo adaptativo proposto no projeto. Como se pode observar na análise da figura 44, o algoritmo adaptativo tendeu a se comportar de uma maneira análoga ao algoritmo A* bidirecional com relação à expansão dos nós da árvore de busca.

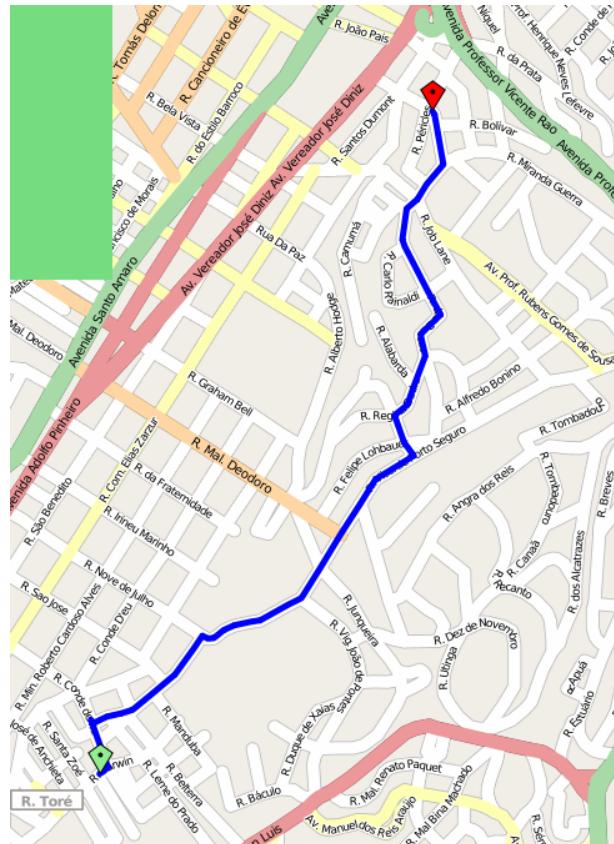


Figura 45: Rota traçada para o teste 1 usando o algoritmo adaptativo com mapa estratificado.

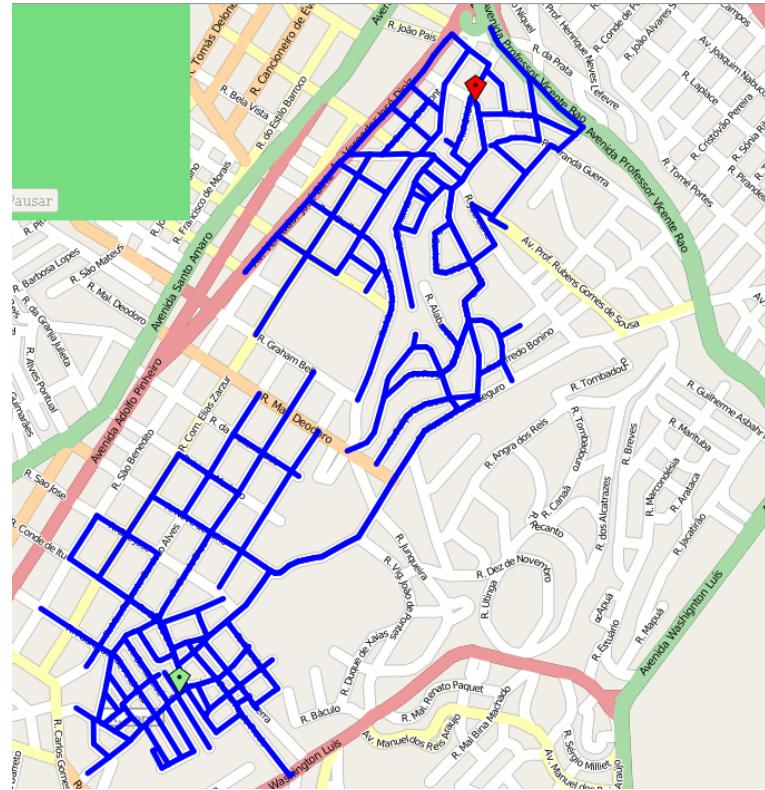


Figura 46: Detalhes do caminho traçado para o teste 1 usando o algoritmo adaptativo com mapa estratificado.

6.1.3.4 Análise do Desempenho para Distâncias Curtas

As figura 47 e 48 ilustram os resultados obtidos para o roteamento entre as ruas Darwin e Farrapos da cidade de São Paulo. Na figura 47 é mostrado o número de nós visitados por cada um dos algoritmos de roteamento testados e na figura 48 é mostrado o tempo de processamento de cada um desses.

Conforme as figuras 49 e 50, para distâncias curtas, o desempenho do algoritmo adaptativo implementado no projeto é equivalente ao desempenho do algoritmo A* bidirecional. A taxa de redução no número de nós visitados chega a aproximadamente 50% para o algoritmo bidirecional e de 51% para o algoritmo proposto. Já na quantidade de tempo despendido, a taxa de redução chega a 66% do valor gasto pelo algoritmo A* original.



Figura 47: Nós visitados para distâncias curtas.

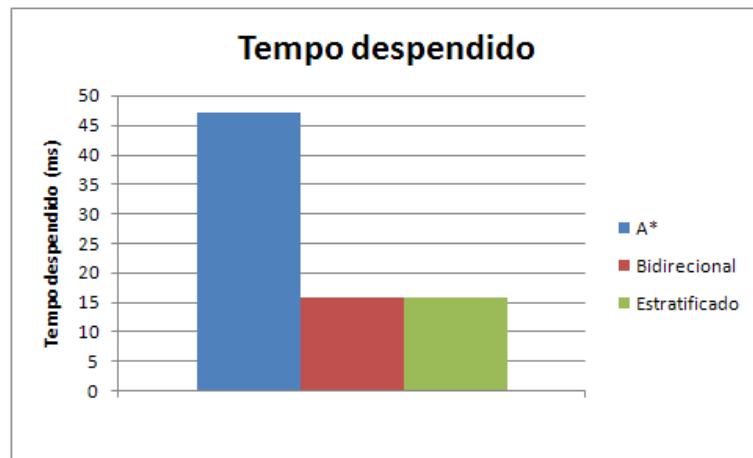


Figura 48: Tempo despendido pelos algoritmos.

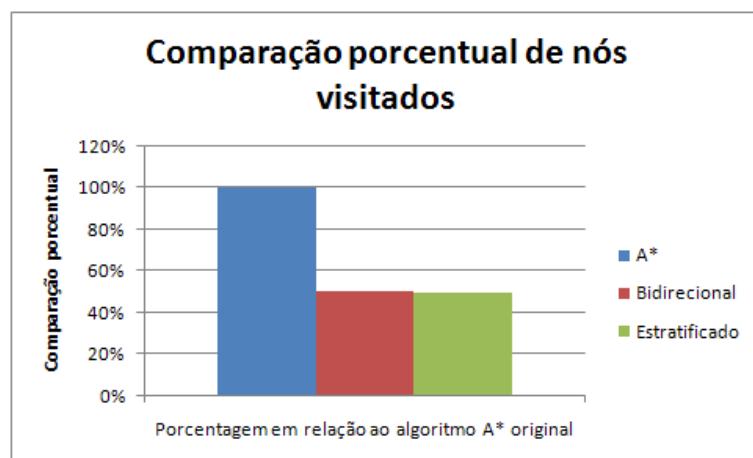


Figura 49: Comparação porcentual do número de nós utilizados em relação ao algoritmo A* original.

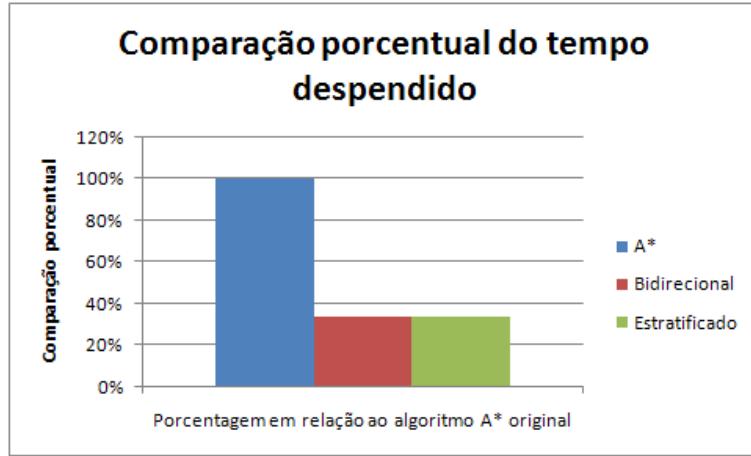


Figura 50: Comparação porcentual do tempo gasto em relação ao algoritmo A* original.

6.1.4 Segundo Conjunto de Testes

No segundo conjunto de testes foi traçado uma rota entre duas ruas localizada a uma distância média uma da outra. As ruas utilizadas nesse teste foram a Rua Darwin e a Rua Bela Cintra conforme indicado na figura 51. Igualmente como foi feito no primeiro conjunto de testes, cada um dos algoritmos implementados no projeto foi utilizado para traçar a rota entre essas duas ruas. Nas seções a seguir são comentadas as rotas traçadas para cada um dos algoritmos e na última seção será analisado o desempenho obtido pelos algoritmos. Assim como ocorreu no primeiro conjunto de testes, as ruas pelas quais os algoritmos rotaram foram as mesmas para todos os algoritmos, conforme pode-se observar nas informações presentes no anexo B, sendo que novamente a diferença apresentada entre eles foram em relação ao número de nós visitados e ao tempo despendido.

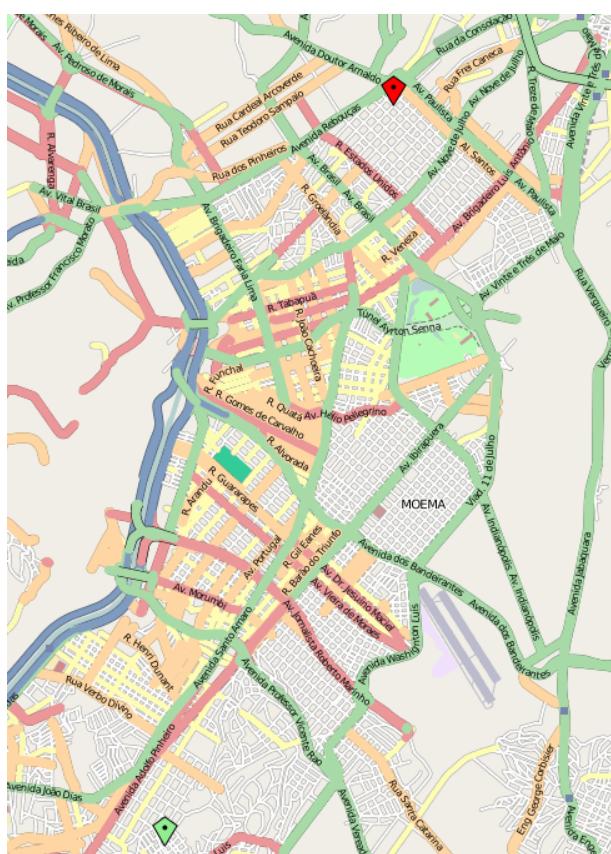


Figura 51: Pontos de origem e destino do teste 2.

6.1.4.1 Algoritmo A*

As figuras 52 e 53 ilustram respectivamente o caminho traçado e a árvore de expansão percorrida pelo algoritmo A* original. Conforme se pode observar da análise da figura 53 o algoritmo tendeu a realizar uma grande expansão na árvore de busca, muitas vezes indo para caminhos que não possuem utilidade para alcançar o objetivo desejado. Nesse segundo teste pode-se observar que o algoritmo A* original tende a sofrer uma enorme expansão no conjunto de nós filhos existentes entre a origem e o destino do mapa.



Figura 52: Rota traçada para o teste 2 usando o algoritmo A*.

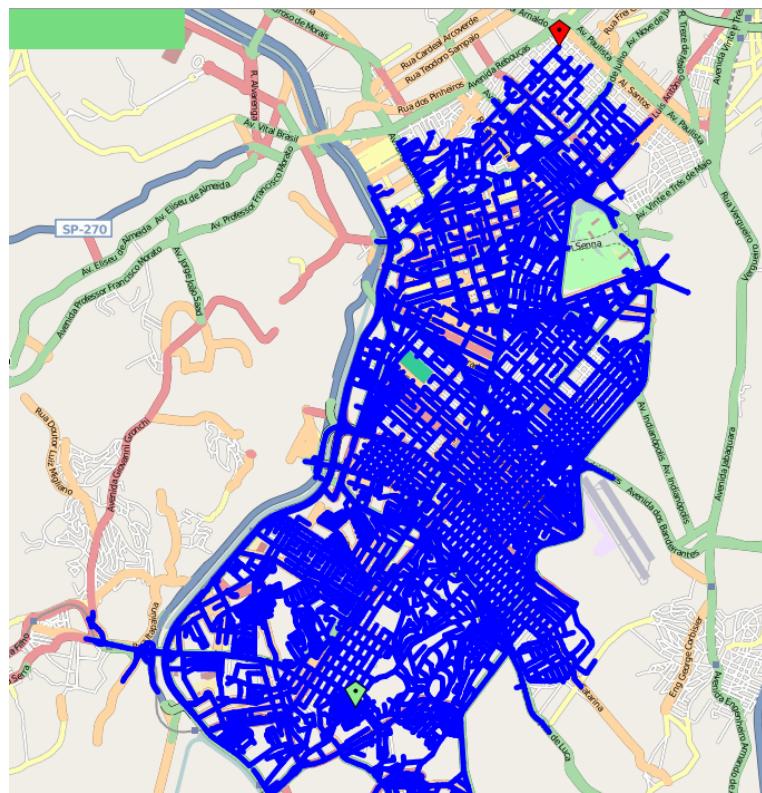


Figura 53: Detalhes do caminho traçado para o teste 2 usando o algoritmo A*.

6.1.4.2 Algoritmo A* Bidirecional

As figuras 54 e 55 ilustram respectivamente o caminho traçado e a árvore de expansão percorrida pelo algoritmo A* bidirecional. Conforme se pode observar da análise da figura 55 o algoritmo A* bidirecional tendeu a sofrer uma menor expansão na árvore de busca, conforme era esperado devido à utilização de dois algoritmos A* originais percorrendo a árvore de busca em sentido inversos. Nesse segundo conjunto de teste pode-se visualizar melhor, conforme a figura 55 que apesar de sofrer uma menor expansão na árvore de busca o algoritmo A* bidirecional ainda sofre uma grande expansão no conjunto de nós filhos nas regiões próximas tanto da origem quanto do destino.

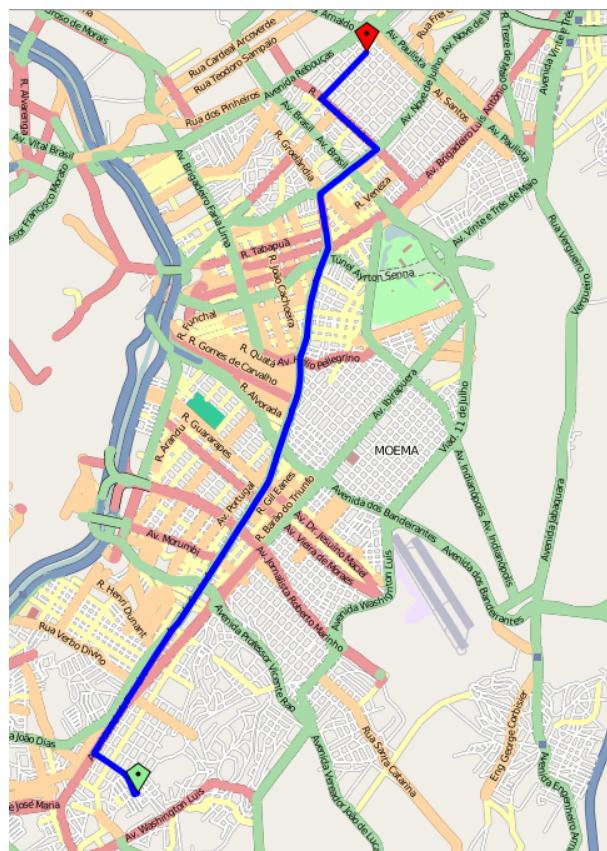


Figura 54: Rota traçada para o teste 2 usando o algoritmo A* bidirecional.

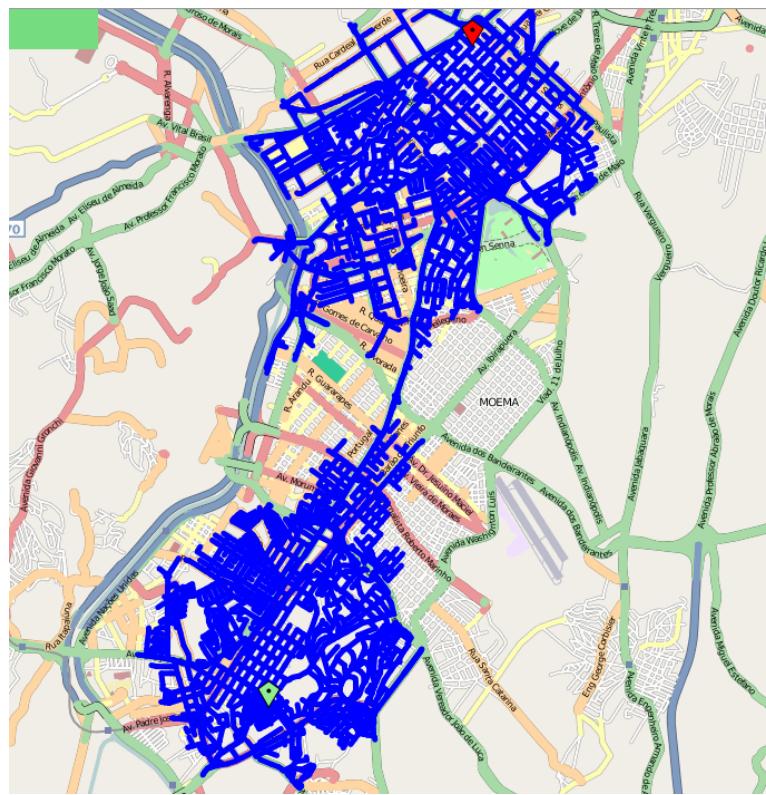


Figura 55: Detalhes do caminho traçado para o teste 2 usando o algoritmo A* bidirecional.

6.1.4.3 Algoritmo Adaptativo

As figuras 56 e 57 ilustram respectivamente o caminho traçado e a árvore de expansão percorrida pelo algoritmo adaptativo proposto no projeto. Como se pode observar na análise da figura 57, o algoritmo adaptativo tendeu a se comportar de uma maneira melhor do que os outros dois algoritmos utilizados no teste, pois esse sofreu uma menor expansão no conjunto de nós da árvore de busca.

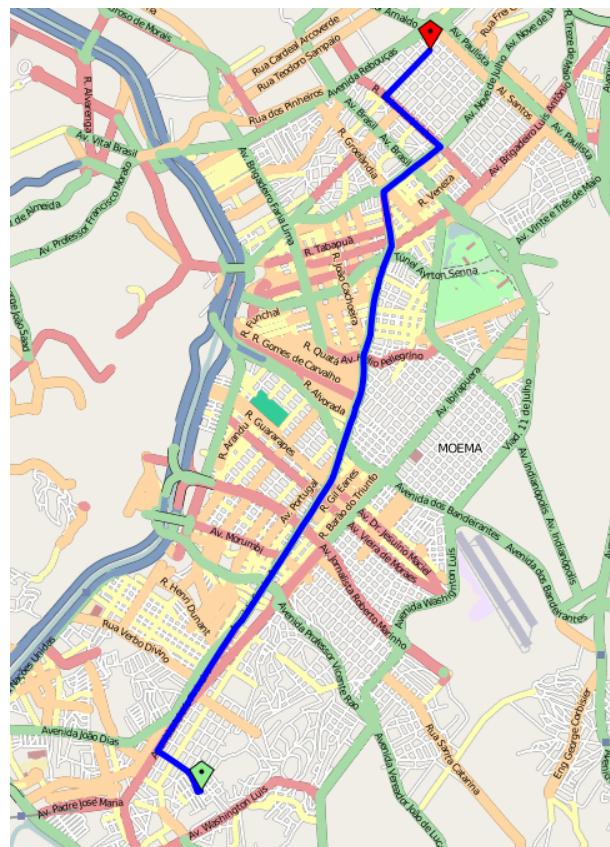


Figura 56: Rota traçada para o teste 2 usando o algoritmo adaptativo com mapa estratificado.

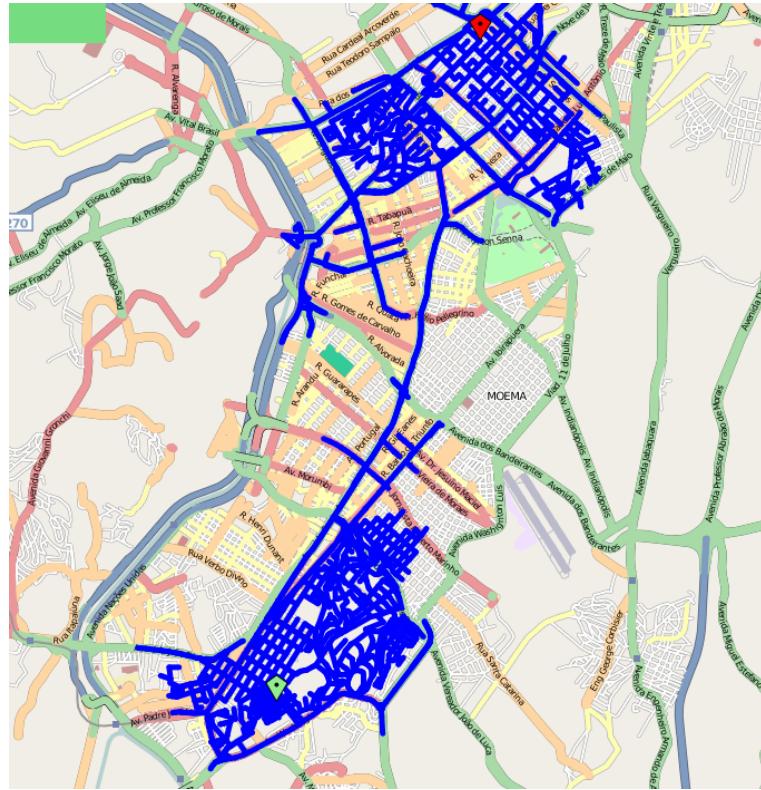


Figura 57: Detalhes do caminho traçado para o teste 2 usando o algoritmo adaptativo com mapa estratificado.

6.1.4.4 Análise do Desempenho para Distâncias Médias

As figuras 58 e 59 ilustram os resultados obtidos para o roteamento entre as ruas Darwin e Bela Cintra da cidade de São Paulo. Na figura 58 é mostrado o número de nós visitados por cada um dos algoritmos de roteamento testados e na figura 59 é mostrado o tempo de processamento de cada um desses.

Conforme as figuras 60 e 61, para distâncias médias, o desempenho do algoritmo adaptativo implementado no projeto é superior ao desempenho dos algoritmos A* original e A* bidirecional. A taxa de redução no número de nós visitados, em relação ao algoritmo A* original, chega a aproximadamente 42% para o algoritmo bidirecional e de 67% para o algoritmo proposto. Já na quantidade de tempo despendido pelo algoritmo proposto, a taxa de redução chega a 94% do valor gasto pelo algoritmo A* original.



Figura 58: Nós visitados para distâncias médias.



Figura 59: Tempo despendido pelos algoritmos.

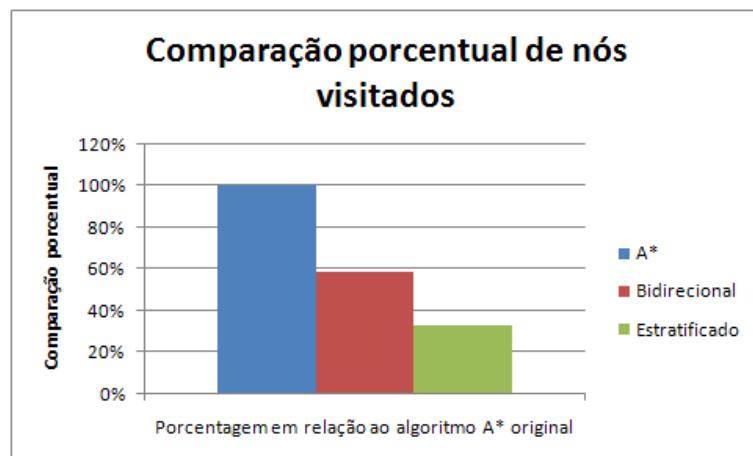


Figura 60: Comparação porcentual do número de nós utilizados em relação ao algoritmo A* original.

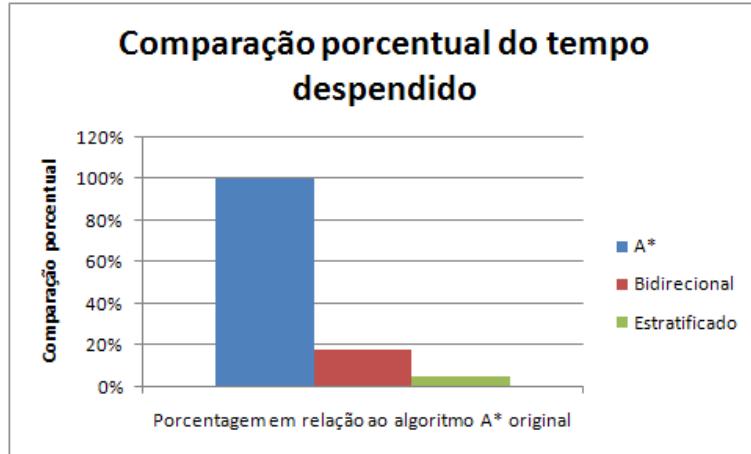


Figura 61: Comparação porcentual do tempo gasto em relação ao algoritmo A* original.

6.1.5 Análise dos resultados obtidos

Como se pode observar nas análises dos testes realizados, o algoritmo adaptativo proposto neste projeto apresenta um desempenho equiparável e em alguns casos superior tanto ao algoritmo A* original quanto ao algoritmo A* bidirecional. Essa superioridade se comprova pela quantidade de nós expandidos na geração de uma rota entre dois pontos do mapa, sendo que a redução na quantidade de nós utilizados pelo algoritmo adaptativo girou entre aproximadamente 50 a 67% da quantidade de nós utilizados pelo algoritmo A* original.

Apesar dos resultados satisfatórios apresentados, será necessário um conjunto de testes maior e mais abrangente para avaliar e comprovar a superioridade do algoritmo proposto. Esse conjunto de testes será necessário para aumentar a abrangência da análise, como também evitar uma conclusão precipitada e não fundamentada em uma quantidade satisfatória de resultados a respeito da eficiência do algoritmo.

7 Considerações Finais

O projeto XOTRANSITO conseguiu atingir o seu principal objetivo, que foi implementar um sistema de geração de rotas considerando-se, além do peso convencional de comprimento, aspectos de trânsito. Foi possível aplicar conceitos de adaptatividade com o intuito de aperfeiçoar o algoritmo A*, através do uso da estratificação de ruas. Porém, devido ao tempo disponível, a idéia do trânsito em tempo real não foi completamente concretizada e, desse modo, entra na lista de futuros avanços.

O estudo dos principais algoritmos de roteamento e suas variantes foi importante no decorrer deste trabalho, assim como a tecnologia adaptativa, apresentada pelo orientador deste trabalho, Prof. Dr. João José Neto. Todo esse conhecimento foi agrupado e reunido com o intuito da criação de um algoritmo que atendesse às necessidades do projeto. A partir disso, foi elaborado o algoritmo A* bidirecional e estratificado, conforme descrito anteriormente.

O algoritmo em questão atende aos requisitos do projeto na medida em que incorpora as qualidades das diversas idéias nele embutidas: o A* original, que serve como base para o roteamento, a bidirecionalidade, que incorpora um melhor desempenho, e a estratificação, um uso mais inteligente da estrutura de ruas e avenidas de uma cidade. Portanto, esta é a contribuição deste trabalho para a comunidade científica: uma possível abordagem para os problemas de roteamento em mapas de ruas, considerando-se diversos aspectos, tais como comprimento, velocidade, trânsito e nível de estratificação.

Além disso, merece destaque a forma como a tecnologia adaptativa foi aplicada ao projeto. Apesar de, historicamente, os dispositivos adaptativos terem surgido como complementos a reconhecedores formais de linguagem (por exemplo, autômatos), a aplicação da adaptatividade certamente não se limita a essa abordagem. De fato, as ações adaptativas no projeto XOTRANSITO atuaram sobre um grafo (mapa de ruas), que pode ser visto como um conjunto de regras (para cada dois vértices, a regra define se existe ou não uma aresta que os une). Essa utilização mostra que a adaptatividade pode se manifestar

em diversos mecanismos subjacentes, sendo eles reconhecedores de linguagem tradicionais ou estruturas de dados vistas como conjunto de regras.

*ANEXO A – Diagramas Engenharia de
Software*

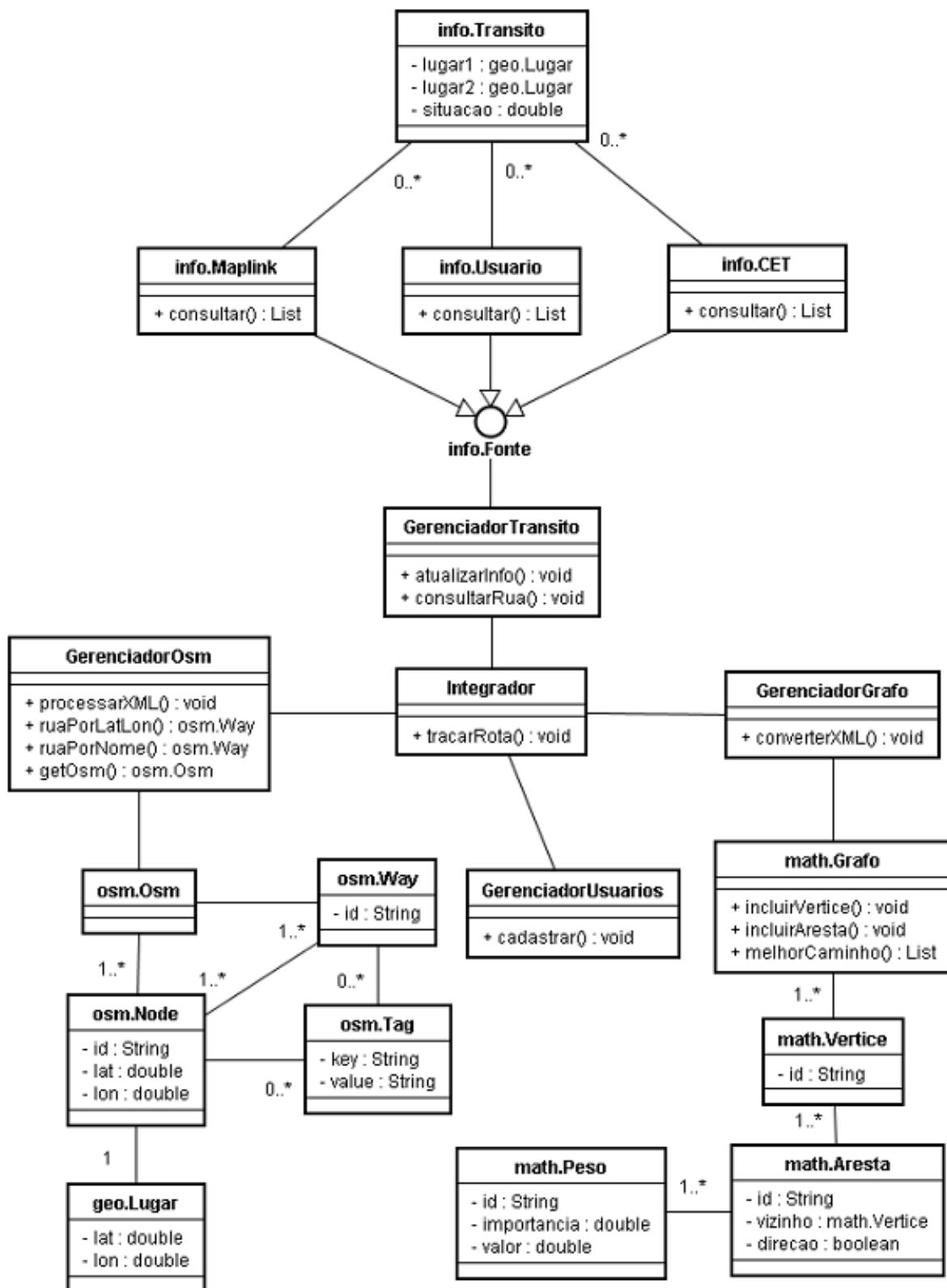


Figura 62: Diagrama de classes do sistema proposto.

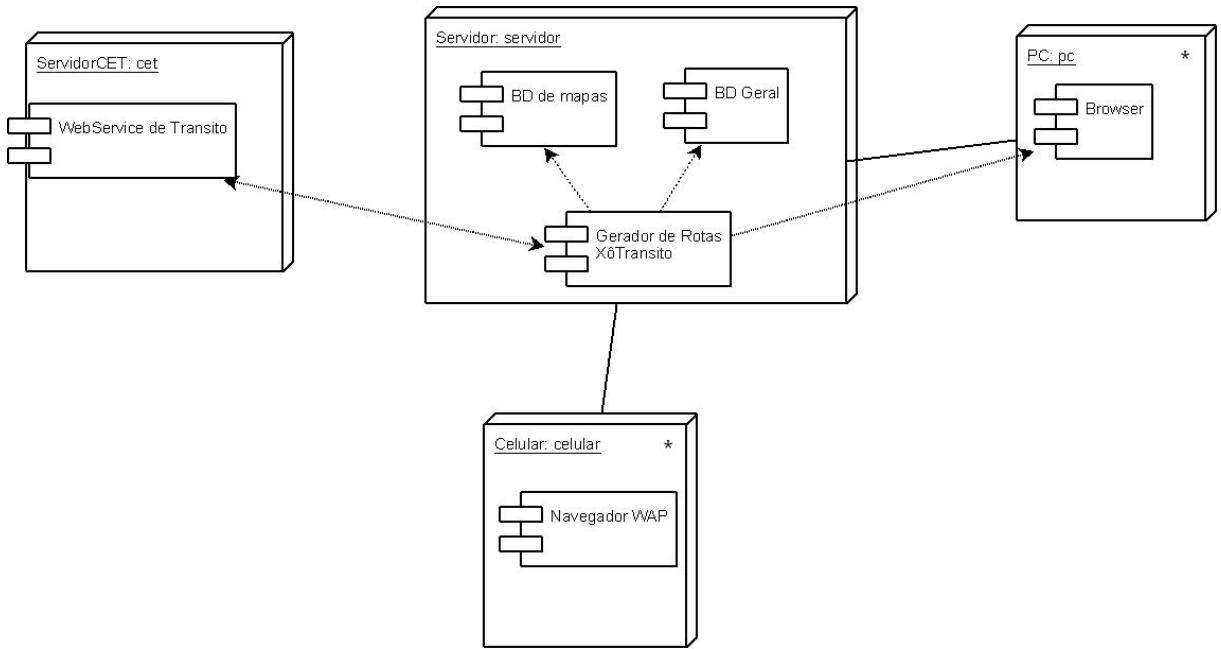


Figura 63: Diagrama de implantação do sistema proposto.

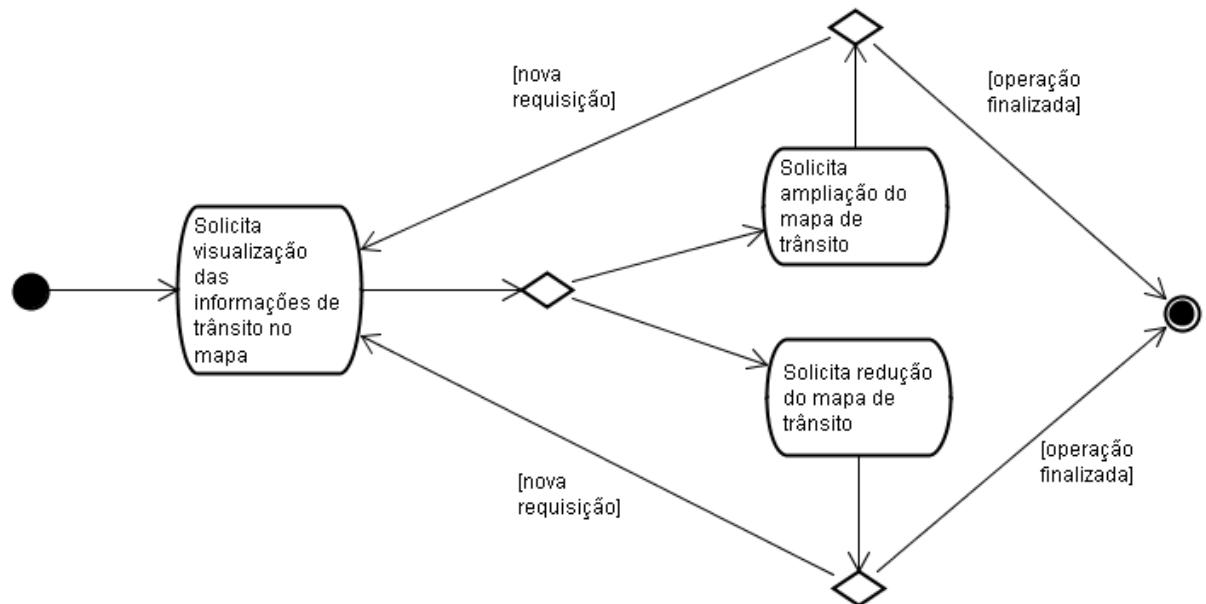


Figura 64: Diagrama de atividade 1.

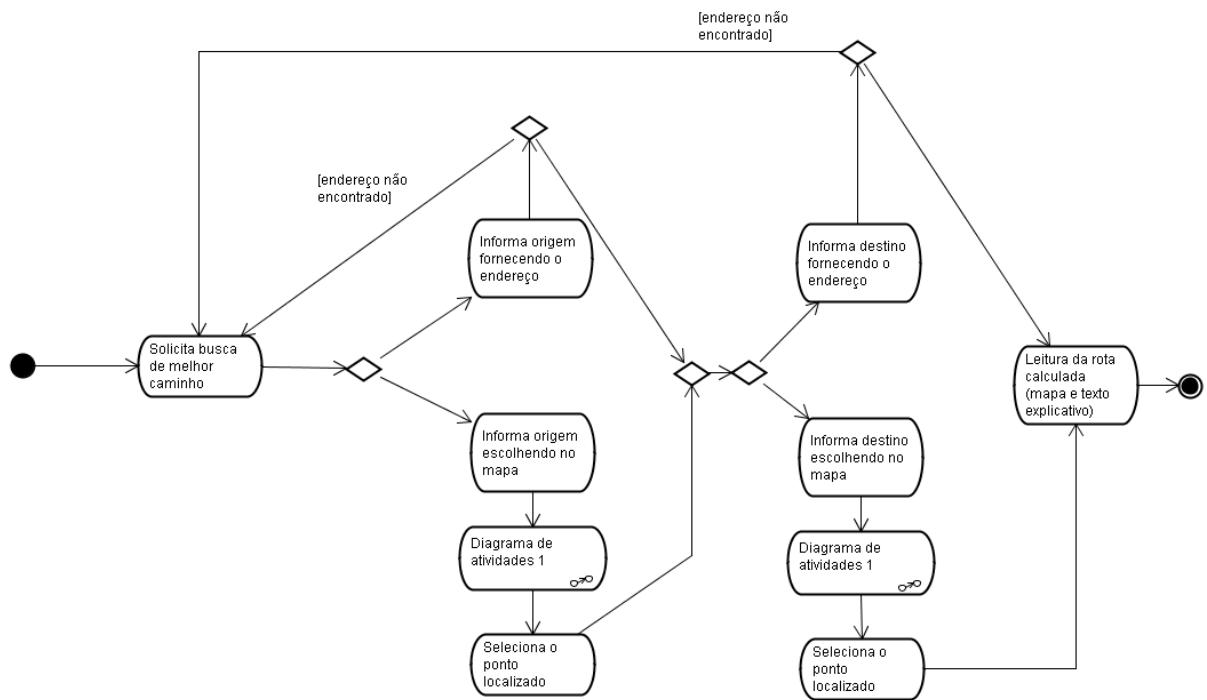


Figura 65: Diagrama de atividade 2.

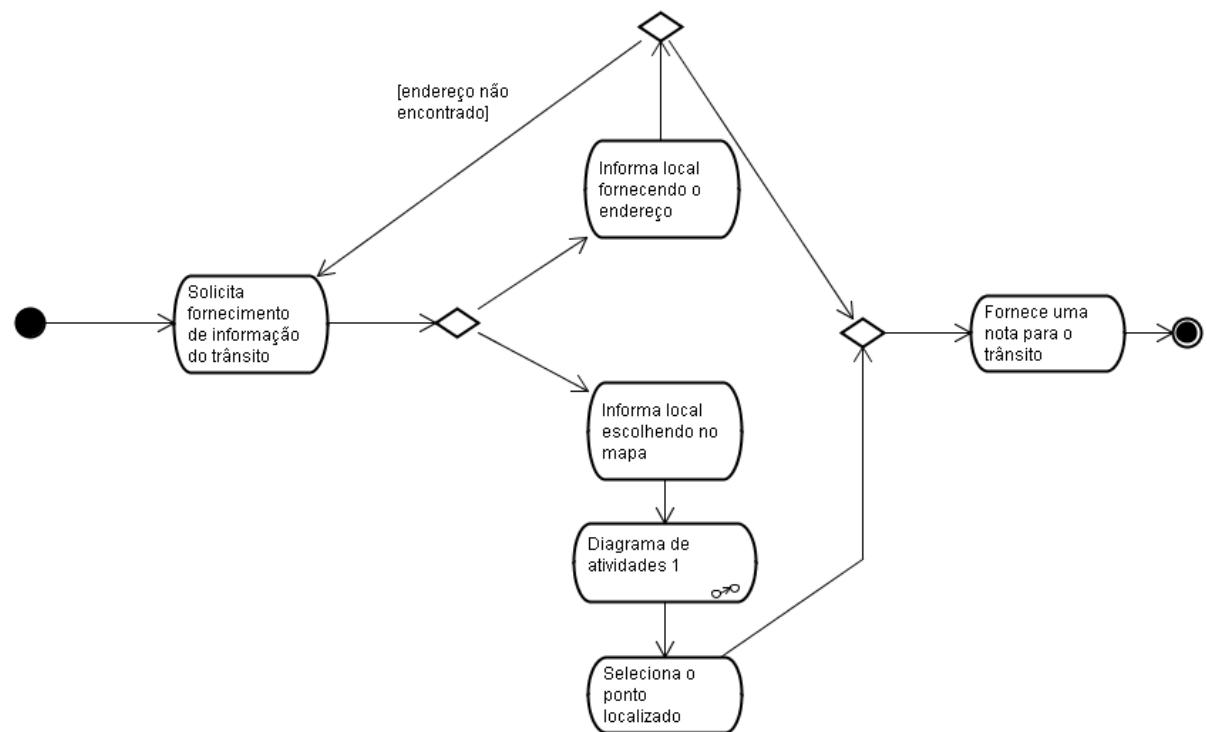


Figura 66: Diagrama de atividade 3.

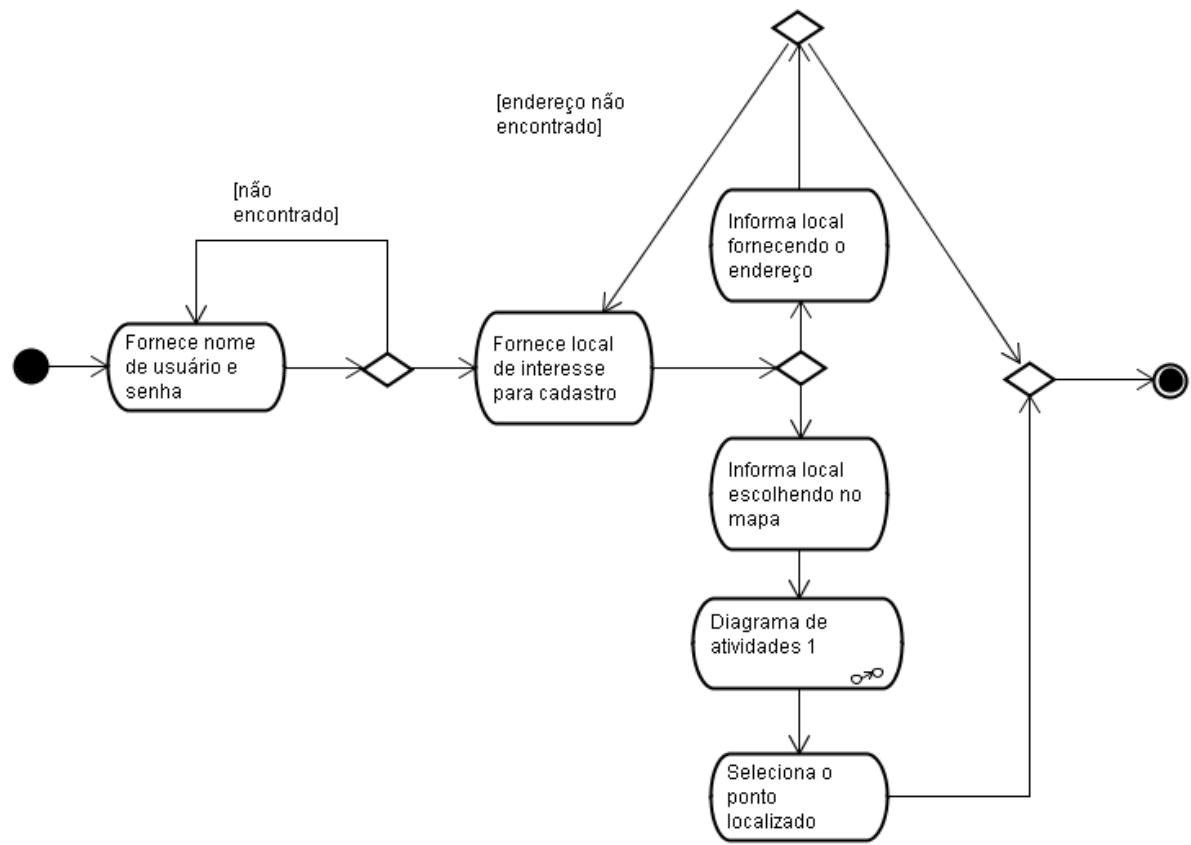


Figura 67: Diagrama de atividade 4.

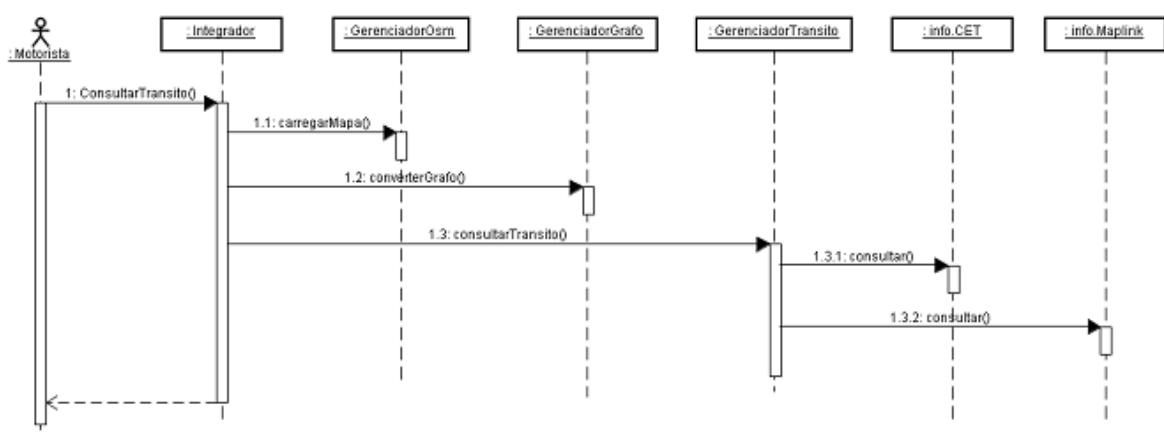


Figura 68: Diagrama de seqüência 1.

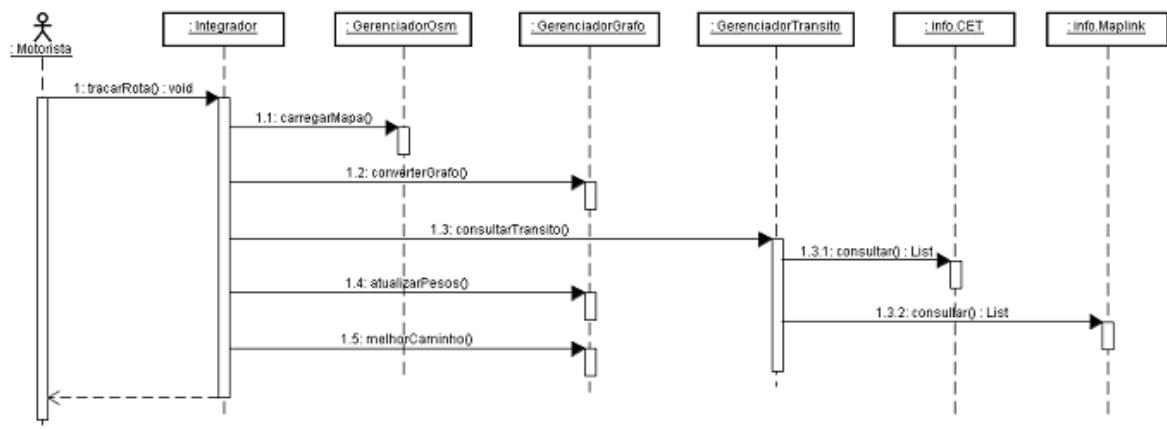


Figura 69: Diagrama de seqüência 2.

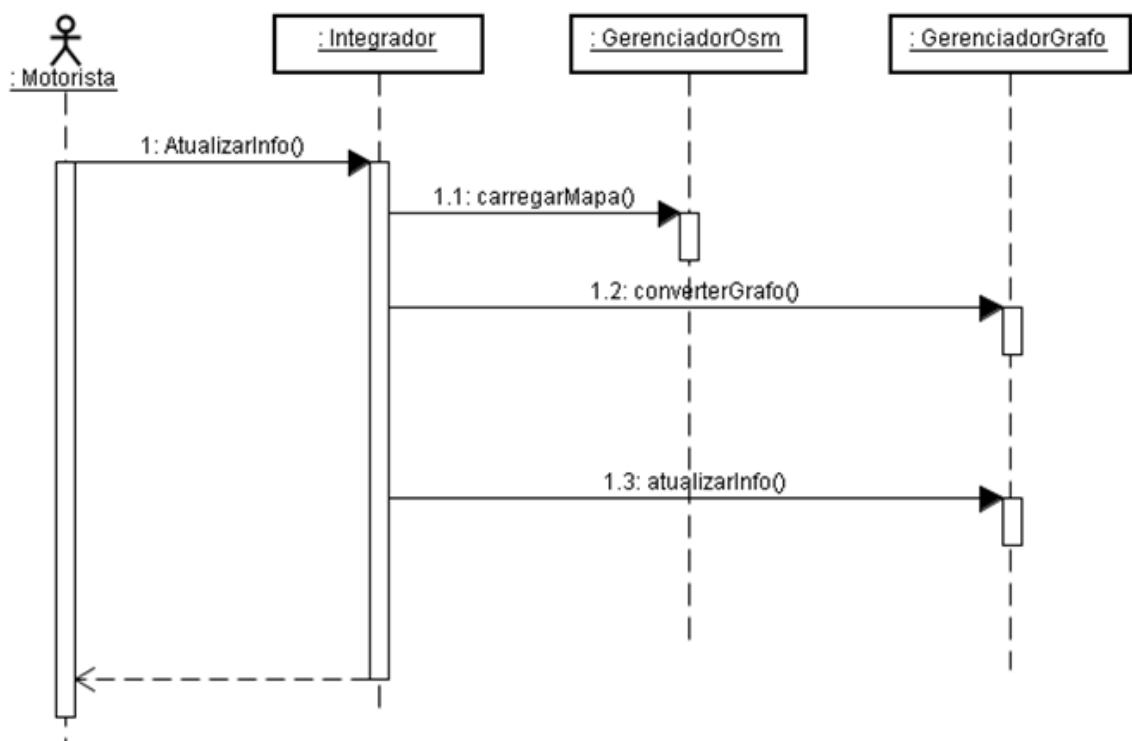


Figura 70: Diagrama de seqüência 3.

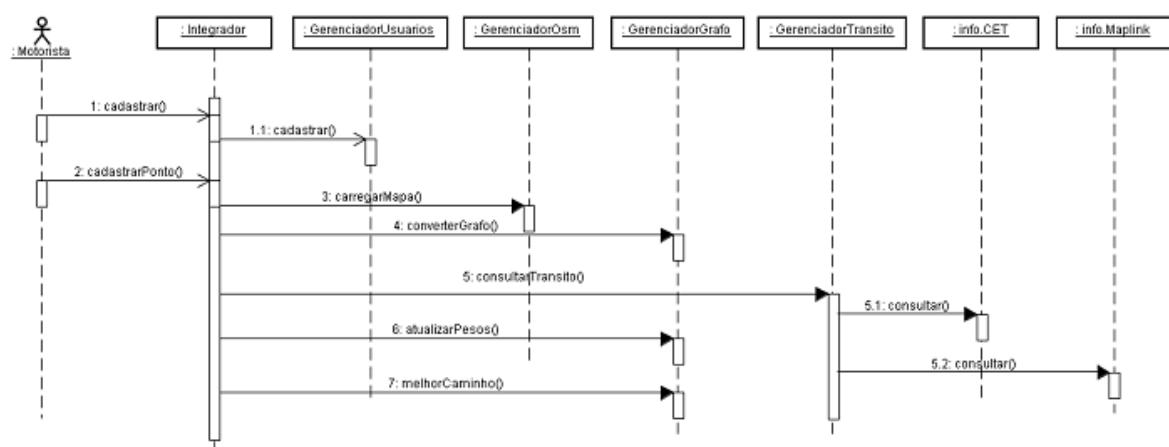


Figura 71: Diagrama de seqüência 4.

ANEXO B - Testes Realizados

B.1 Teste 1

Tipo de rota: rota curta

Tipo de caminho: longe de avenidas principais

Endereço de origem: Rua Darwin

Endereço de destino: Rua Farrapos

B.1.1 Resultados dos Algoritmos

B.1.1.1 A*

Número de nós visitados: 764

Tempo decorrido: 47ms

Percorso:

1. R. Darwin
2. R. Conde de Itu
3. R. Visc. de Porto Seguro
4. R. Aurélia Perez Alvarez
5. R. Regina Badra
6. R. das Barcas
7. R. Miranda Guerra
8. R. Farrapos

Ilustrações dos Resultados:

Figura 40: pontos de origem destino

Figura 41: rota traçada

Figura 42: detalhes do caminho traçado

B.1.1.2 Bidirecional

Número de nós visitados: 382

Tempo decorrido: 16ms

Percorso:

1. R. Darwin
2. R. Conde de Itu
3. R. Visc. de Porto Seguro
4. R. Aurélia Perez Alvarez
5. R. Regina Badra
6. R. das Barcas
7. R. Miranda Guerra
8. R. Farrapos

Figura 43: rota traçada

Figura 44: detalhes do caminho traçado

B.1.1.3 Estratificado

Número de nós visitados: 376

Tempo decorrido: 16ms

Percorso:

1. R. Darwin
2. R. Conde de Itu
3. R. Visc. de Porto Seguro
4. R. Aurélia Perez Alvarez
5. R. Regina Badra

6. R. das Barcas
7. R. Miranda Guerra
8. R. Farrapos

Figura 45: rota traçada

Figura 46: detalhes do caminho

B.2 Teste 2

Tipo de rota: rota média

Tipo de caminho: longe de avenidas principais

Endereço de origem: Rua Darwin

Endereço de destino: Rua Bela Cintra

B.2.1 Resultados dos Algoritmos

B.2.1.1 A*

Número de nós visitados: 8448

Tempo decorrido: 8203ms

Percorso:

1. R. Darwin
2. R. Conde de Itu
3. Avenida Adolfo Pinheiro
4. Avenida Santo Amaro
5. Túnel São Gabriel
6. Av. São Gabriel
7. Av. Nove de Julho
8. R. Estados Unidos
9. R. Bela Cintra

Ilustrações dos Resultados:

Figura 51: pontos de origem e destino

Figura 52: rota traçada

Figura 53: detalhes do caminho

B.2.1.2 Bidirecional

Número de nós visitados: 4921

Tempo decorrido: 1484ms

Percorso:

1. R. Darwin
2. R. Conde de Itu
3. Avenida Adolfo Pinheiro
4. Avenida Santo Amaro
5. Túnel São Gabriel
6. Av. São Gabriel
7. Av. Nove de Julho
8. R. Estados Unidos
9. R. Bela Cintra

Figura 54: rota traçada

Figura 55: detalhes do caminho

B.2.1.3 Estratificado

Número de nós visitados: 2782

Tempo decorrido: 438ms

Percorso:

1. R. Darwin
2. R. Conde de Itu
3. Avenida Adolfo Pinheiro
4. Avenida Santo Amaro

5. Túnel São Gabriel

6. Av. São Gabriel

7. Av. Nove de Julho

8. R. Estados Unidos

9. R. Bela Cintra

Figura 56: rota traçada

Figura 57: detalhes do caminho

Referências

- A Guide to the Project Management Body of Knowledge (PMBOK Guide). 3. ed. [S.l.]: Project Management Institute, 2004.
- GOOGLE Maps API. Disponível em: <<http://code.google.com/apis/maps>>. Acesso em: 8 jun. 2008.
- HART, P. E.; NILSSON, N. J.; RAPHAEL, B. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions of Systems Science and Cybernetics*, v. 4, n. 2, 1968.
- JESUS, L. de et al. Adaptools 2.0: Aspectos de implementação e utilização. *Revista IEEE América Latina*, v. 5, n. 7, 2007.
- KNUTH, D. E. *The Art Of Computer Programming*. 3. ed. Boston: Addison-Wesley, 1997.
- LESTER, P. *A* Pathfinding para Iniciantes*. Disponível em: <http://www.policyalmanac.org/games/aStarTutorial_port.htm>. Acesso em: 26 nov. 2008.
- MAPLINK: Guia de ruas e mapas rodoviários, com rotas e localizador de endereço. 2008. Disponível em: <<http://maplink.uol.com.br>>. Acesso em: 13 nov. 2008.
- NETO, J. J. Tecnologia adaptativa (roteiro de estudos). 2004. Disponível em: <http://www.pcs.usp.br/lta/download/roteiro_estudo.pdf>. Acesso em: 12 abr. 2008.
- NETO, J. J. Adaptatividade e tecnologia adaptativa: Tutorial baseado no wta 2007. *Revista IEEE América Latina*, v. 5, n. 7, 2007.
- OPENSTREETMAP. Disponível em: <<http://www.openstreetmap.org>>. Acesso em: 8 jun. 2008.
- PIJLSY, W.; POSTZ, H. Bidirectional a*: Comparing balanced and symmetric heuristic methods. *Econometric Institute Report EI 2006-41*, 2006.
- PISTORI, H. Tecnologia adaptativa em engenharia de computação: Estado da arte e aplicações. *Tese de Doutorado*, São Paulo, 2003.
- PISTORI, H.; NETO, J. J. Adaptree: Proposta de um algoritmo para indução de Árvores de decisão baseado em técnicas adaptativas. *Conferência Latino Americana de Informática (CLEI)*, Montevideo, 2002.
- RUSSELL, S.; NORVIG, P. *Artificial Intelligence: A modern approach*. 2. ed. New Jersey: Prentice Hall, 2003. 97-101 p.

SIMMETRICS API. Disponível em:
<<http://www.dcs.shef.ac.uk/sam/simmetrics.html>>. Acesso em: 8 jun. 2008.

TCHEMRA, A. H. Aplicação da tecnologia adaptativa em sistemas de tomada de decisão. *Revista IEEE América Latina*, v. 5, n. 7, 2007.

WIKIPEDIA. *A* Star Search Algorithm*. Disponível em:
<http://en.wikipedia.org/wiki/A_star>. Acesso em: 26 set. 2008.

WIKIPEDIA. *Best-First Search*. Disponível em: <http://en.wikipedia.org/wiki/Best-first_search>. Acesso em: 2 set. 2008.

WIKIPEDIA. *Bidirectional Search*. Disponível em:
<http://en.wikipedia.org/wiki/Bidirectional_search>. Acesso em: 2 set. 2008.

WIKIPEDIA. *Breadth-First Search*. Disponível em:
<http://en.wikipedia.org/wiki/Breadth-first_search>. Acesso em: 22 set. 2008.

WIKIPEDIA. *Depth-First Search*. Disponível em: <http://en.wikipedia.org/wiki/Depth-first_search>. Acesso em: 25 jun. 2008.

WIKIPEDIA. *Depth-Limited Search*. Disponível em:
<http://en.wikipedia.org/wiki/Depth-limited_search>. Acesso em: 22 ago. 2008.

WIKIPEDIA. *Uniform-Cost Search*. Disponível em:
<http://en.wikipedia.org/wiki/Uniform-cost_search>. Acesso em: 2 set. 2008.