EXACTA Engenharia de Produção eISSN 1983-9308
https://doi.org/10.5585/ExactaEP.v17n3.8444



# Desenvolvimento de um software educacional para apoio ao ensino de localização e roteirização

# Development of educational software to support location and routing teaching

José Aurir Gonçalves de Almeida Junior[1]
Heráclito Lopes Jaguaribe Pontes[2]
Marcos Ronaldo Albertin[3]

**Resumo**
Os processos educacionais passam por constantes alterações para se adaptarem às novas realidades digitais, oriundas da existência de um desenvolvimento tecnológico incessante, causador de profundas transformações sociais. Neste contexto, softwares apresentam-se como ferramentas importantes na educação há décadas. Considerando esta importância, a motivação deste trabalho parte da identificação de uma oportunidade de contribuir para o ensino de métodos de localização e roteirização em disciplinas de logística, através de um software desenvolvido com foco educacional. Para tanto, este trabalho inicia pela definição de tecnologias adequadas ao seu desenvolvimento, utilizando-as para a elaboração de algoritmos e de uma interface gráfica capazes de proporcionar ao usuário a aplicação de métodos de localização e roteirização, complementando o ensino teórico destes temas. O sistema desenvolvido é então verificado em comparação com outros sistemas, como o LogWare®, Google® Maps e Excel®, permitindo sua validação como software educacional, mostrando suas vantagens, limitações e oportunidades de melhoria.

**Palavra-chave**: Ensino; Localização; Roteirização; Software Educacional.

**Abstract**
Educational approaches deal with constant changes in themselves to adapt in the new digital realities that emerges from the existence of an unstoppable technologic development, prompter of deep social transformations. In this context, software's show themselves as important tools in education for decades. Considering this importance, the motivation for this works arises from the identification of an opportunity to contribute to the education of location and routing problems in logistics courses, through the development of software with educational focus. This work starts with the definition of some technologies, adequate to its development, using them to elaborate algorithms and a graphical user interface, able of offer to the end user the application of location and routing methods, complementing the theoretical teaching of these subjects. The developed system is, then, verified through the comparison with some other reference systems, like LogWare®, Google® Maps and Excel®, allowing the validation as educational software, showing its advantages, limitations and opportunities for improvement.

**Keywords**: Teaching; Location; Routing; Educational Software.

[1]Graduado em Engenharia de Produção Mecânica, Universidade Federal do Ceará.
aurirjr@gmail.com

[2]Doutor em Engenharia Mecânica/Manufatura, Professor do Departamento de Engenharia de Produção, Universidade Federal do Ceará.
hjaguaribe@ufc.br

[3]Doutor em Engenharia de Produção, Professor do Departamento de Engenharia de Produção, Universidade Federal do Ceará.
albertin@ot.ufc.br

## 1 Introdução

O rápido desenvolvimento tecnológico vem sendo impulsionador de profundas transformações culturais, econômicas e sociais. Os processos educacionais também precisam adaptar-se às novas realidades e possibilidades oferecidas pela tecnologia. Segundo Cruz e Neri (2014, p.9), "docentes têm em suas aulas alunos com uma profunda percepção de tecnologias, de forma que precisam estar preparados para utilizar tais tecnologias e para se beneficiar do uso das mesmas".

Neste contexto de evolução tecnológica, percebe-se o surgimento de diversos recursos audiovisuais que oferecem novas possibilidades de explorar os sentidos do aluno em busca de maior efetividade do aprendizado.

Neste ambiente tecnológico, os computadores são importantes engrenagens do mundo digital. Além dos computadores, os *softwares* são elementos fundamentais das tecnologias digitais, que passam a ter significava relevância na dinâmica educacional.

"A utilização de ambientes informatizados, empregando-se *softwares* educativos avaliados previamente pelo professor, acompanhados de uma didática construtiva e evolutiva, pode ser uma solução interessante para os diversos problemas de aprendizagem em diferentes níveis" (Magedanz, 2004. p.6).

Dentre os muitos temas de ensino passíveis de apoio educacional por *softwares*, o presente trabalho explora a área de logística. A respeito do ensino dessa disciplina, identificou-se uma deficiência na oferta de *softwares*, com foco educacional e gratuito, que tratem de estratégias de roteirização e localização de instalações. Georges e Seydell (2008) complementam que *softwares* da área de roteirização e localização são capazes de resolver uma gama relativamente importante de problemas de natureza logística. Contudo, além de serem dispendiosos, muitas vezes são oferecidos em versões acadêmicas limitadas ou que acompanham livros-texto, como o, por exemplo, o programa LogWare® do livro de Ballou (2006).

Com isso, este trabalho tem por objetivo o desenvolvimento e a verificação de um *software* educacional para apoiar o ensino de roteirização e localização de instalações em disciplinas de logística, disponibilizando um recurso didático para o aprendizado.

O trabalho é composto por mais três seções: a segunda seção é a fundamentação teórica das técnicas e teorias utilizadas, a terceira seção aborda o desenvolvimento do sistema e os resultados obtidos, e, a última seção aborda as conclusões do trabalho.

## 2 Fundamentação teórica

### 2.1 Localização de instalações

De acordo com Ballou (2006), as decisões sobre localização envolvem a determinação do número, local e proporções das instalações a serem usadas. Essas instalações incluem pontos nodais da rede, como fábricas, portos, armazéns, pontos de varejo e pontos centrais de serviços na cadeia de suprimentos.

#### 2.1.1 Métodos de localização de instalações

Existem vários modelos que teorizam a problemática da localização de instalações. Segundo Fitzsimmons e Fitzsimmons (2004), apesar da

seleção do local ser influenciada por fatores qualitativos, um estudo baseado em métodos quantitativos, tais como custos e distância, pode dar uma direção mais assertiva e útil.

De acordo com Ballou (2006), os problemas de localização podem ser classificados segundo alguns critérios, como o de força direcionadora, segundo o qual a localização de instalações é determinada por um fator fundamental, geralmente de forte teor econômico.

Conforme Ballou (2006), o número de instalações é um critério para classificação dos métodos de localização, caracterizando-se em decidir a localização de somente um ponto ótimo ou de selecionar múltiplos locais dependendo de certas variáveis. Segundo Wanke (2003), localizar apenas uma ou várias instalações são problemas consideravelmente diferentes. Na localização única, evita-se a necessidade de considerar fatores como os custos fixos de operações.

De acordo com Slack, Chambers e Johnston (2002), o método mais utilizado para localização de uma planta única, terminal ou armazém, é o chamado Centro de Gravidade Exato (CGE), método de grade ou método centroide. O modelo é classificado matematicamente como um modelo estático de localização. Para Bowersox e Closs (2001), o método CGE é uma alternativa para buscar a melhor localização geográfica. Este centro pode ser relacionado a vários tipos de taxas, como pesos, volumes e distâncias, para selecionar a alternativa de menor custo.

Dentre possíveis métodos exatos de localização múltipla, Ballou (2006) destaca o método do múltiplo centro de gravidade e o método *p*-mediana.

O método do múltiplo centro de gravidade utiliza o método centroide, que calcula o centro e gravidade exato, em conglomerados de vértices, definidos segundo algum critério. Uma maneira de abordar tal problema é configurar os conglomerados mediante a concentração dos pontos mais próximos entre si. Depois de se encontrar as localizações de centro de gravidade, os pontos são redistribuídos à estas localizações. Novas localizações de centro de gravidade são encontradas para os conglomerados revisados. O processo continua até que não se encontre mais mudança. Isso completa as computações para um número específico de instalações a ser localizado. E pode ser repetido com diferentes números de instalações (Ballou, 2006).

Já em relação ao método p-mediana, Lorena, Senne, Paiva e Pereira (2001) afirmam ser um problema clássico de localização e consiste em localizar p centros (medianas) em uma rede de modo a minimizar a soma das distâncias de cada vértice ao centro mais próximo.

## 2.2 Roteirização

A roteirização é o processo logístico que tem como objetivo a melhoria nos trajetos que um veículo deve percorrer, geralmente a fim de minimizar o tempo ou a distância. É um dos meios fundamentais para se reduzir os custos e proporcionar melhorias na prestação dos serviços, de forma a reduzir o tempo de transporte e cumprir as metas impostas no processo. Assim, a melhor utilização da frota reflete num menor número de veículos e em menores custos operacionais (Ballou, 2006).

Segundo Laporte, Gendreau e Semet (2002), a importância da roteirização para a redução de custos e melhoria do nível de serviço é muito relevante, pois, através do planejamento, evitam-se os desperdícios, reduzem-se os custos, aperfeiçoa-

se o desempenho operacional e, consequentemente, melhora-se o nível de serviço.

A problemática de roteirização pode ser definida em três fatores fundamentais: decisões, que dizem respeito à programação e ao planejamento de clientes a serem visitados, motoristas e veículos a serem utilizados e sequência das entregas a cada cliente; objetivos, que procuram elevar o nível do serviço, mas com custos baixos; e restrições, que devem ser estabelecidas, como os horários, a jornada de trabalho, o tamanho dos veículos nas vias públicas, dentre outros (Novaes, 2007).

### 2.2.1 Métodos de roteirização

De acordo com Novaes (2007), os métodos de roteirização são utilizados para efetuar o planejamento das rotas de maneira eficiente, para que sua execução possa ser realizada da melhor forma possível.

Sistemas de roteirização são sistemas computacionais que, através de algoritmos e base de dados, são capazes de obter soluções para problemas de roteirização com resultados satisfatórios, consumindo tempo e esforço relativamente pequenos quando comparados aos gastos nos tradicionais métodos manuais (Cunha, 1997).

> Embora sejam muitas as variações dos problemas de roteirização, é possível reduzi-los a alguns modelos básicos. Existe o problema de encontrar uma rota ao longo de uma rede em que o ponto de origem seja diferente do ponto de destino. Há um problema similar sempre que se apresentam múltiplos pontos de origem e de destino. Mais complexo ainda é o problema de fazer itinerários quando os pontos de origem e destino são os mesmos. (Ballou, 2006, p.191).

De acordo com Von Atzingen, Da Cunha, Nakamoto, Ribeiro e Schardong, (2012), o Problema de Caminho Mínimo (PCM) consiste em determinar um caminho entre dois vértices tal que a somatória dos custos unitários dos arcos que compõem este caminho seja o mínimo.

O algoritmo de Dijkstra resolve o PCM de uma única origem em um grafo orientado. Ele encontra o menor caminho entre quaisquer dois nós da rede, quando todos os arcos têm comprimento não-negativo. Nele é utilizado um procedimento iterativo, determinando, na iteração 1, o nó mais próximo do nó 1, na segunda iteração, o segundo nó mais próximo do nó 1, e assim sucessivamente, até que em alguma iteração o nó destino seja atingido. (Arenales, Armentano, Morabito & Yanasse, 2007).

Além de problemas de roteirização com um ponto de origem e outro de destino, Ballou (2006) destaca também os problemas com pontos de origem e destino múltiplos, que ocorrem quando há mais de um vendedor, fábrica ou armazém para servir a mais de um cliente com o mesmo produto. Um tipo de algoritmo de Programação Linear (PO), o método do transporte, é aplicado a este problema.

O problema de transporte é uma classe especial de problemas de PO que trata do envio de uma mercadoria de origens para destinos, onde o objetivo é a determinação de uma programação que minimize o custo total e satisfaça os limites de fornecimento e demanda (Taha, 2008).

Ballou (2016) aborda outra situação de roteirização na qual o ponto de origem e o de destino são os mesmos, caracterizando o Problema do Caixeiro Viajante (PCV). Para Gomes (2008), o PCV pode ser entendido como o problema de um vendedor que deseja visitar um conjunto de cidades, passando exatamente uma vez por cada uma, voltando ao ponto de partida no final do seu percurso.

Para Cunha (1997), problemas de roteirização de veículos são muitas vezes definidos como problemas de um ou mais caixeiros viajantes que incluem restrições adicionais de capacidade, além de outras que dependem de cada aplicação. Ballou (2006) aborda dois métodos dentre as inúmeras abordagens para enfrentar problemas dessa complexidade. Um deles é simples (o método da varredura), e o outro, mais complexo, enfrentando elementos mais práticos e produzindo soluções de maior qualidade, (o método das economias).

### *2.3 Desenvolvimento de software*

Para Pressman e Maxin (2016), software abrange programas executáveis em computador, conteúdos (apresentados à medida que os programas são executados), informações descritivas, tanto na forma impressa quanto na virtual, abrangendo a mídia eletrônica. Segundo Fernandes (2002), é uma sentença escrita em uma linguagem computável, para a qual existe uma máquina (computável) capaz de interpretá-la. Ao interpretar o software, a máquina computável é direcionada à realização de tarefas específicas.

Bass, Clements e Kazman (2003) afirmam que a arquitetura de software constitui um modelo relativamente pequeno e compreensível, de como o sistema é estruturado e como seus elementos trabalham juntos.

"A escolha de um padrão de arquitetura deve ser guiada pelas propriedades gerais da aplicação a ser desenvolvida. Antes de escolher um padrão específico, deve-se explorar várias alternativas, pois diferentes padrões de arquitetura implicam em diferentes estruturas, consequentemente, diferentes sistemas". (Leite, 2007, p.91).

A seguir, são explorados diversos elementos presentes na arquitetura deste projeto: HTML, CSS, JavaScript, Angular, Google Maps, GitHub e Firebase.

1. HTML: é a sigla em inglês para *HyperText Markup Language*, que, em português, significa Linguagem para Marcação de Hipertexto. Brooks (2007) ressalta que é uma linguagem de marcação usada para especificar a estrutura de um documento. Um navegador de internet é um *software* que interpreta estas marcações de estrutura, então, constrói uma página *web* com recursos de hipermídia com os quais o usuário pode interagir. Dentre as diversas marcações possíveis nesta linguagem, este trabalho faz uso do *Scalable Vector Graphics* (SVG). Conforme Mozilla (2017), gráficos vetoriais escaláveis podem ser embutidos diretamente no HTML para descrever de forma vetorial desenhos e gráficos bidimensionais de forma estática, dinâmica ou animada.

2. CSS: Outra linguagem de marcação relevante no desenvolvimento *Web* é o *Cascading Style Sheets* (CSS). O Mozilla (2017) define como uma linguagem de estilo usada para descrever a apresentação de um documento escrito em HTML, descrevendo como elementos são mostrados na tela, no papel, no discurso ou em outras mídias. É uma linguagem de estilo usada para especificar a aparência dos vários elementos de um documento que foi definido por uma linguagem de marcação, de forma separada.

3. *JavaScript:* é uma linguagem de programação interpretada, utilizada em *browser*, que permite aos scripts interagirem com o usuário, controlarem o browser, ou alterarem o documento sendo apresentado pelo browser (Flanagan, 2006). Bortolossi (2012) apresenta *JavaScript* como uma linguagem que permite modificar e integrar, de forma dinâmica e interativa, o conteúdo e a

aparência dos vários elementos que compõem uma aplicação elaborada com HTML e CSS.

4. Angular: é um framework para construção de aplicações web utilizando HTML, CSS e TypeScript. Um framework é uma abstração e implementação para uma aplicação em um dado domínio do problema. Uma parte que pode ser utilizada em diversos projetos e que, em conjunto com outros componentes, completam uma aplicação (Korva, 2016). Angular tem com uma arquitetura Model-View-Controller (MVC) e baseada em componentes. O MVC é dividido em três componentes: model (modelo) encapsula o núcleo de dados e de funcionalidade da aplicação; view (visualização) apresenta informações oriundas do modelo para o usuário e; controller (controlador) manuseia as informações segundo eventos acionados pelo usuário (Bubeck, 1992). A principal linguagem de programação utilizada em Angular é o TypeScript.

5. Google Maps: é um serviço online de pesquisa e visualização de mapas e imagens de satélite da Terra, gratuito até certos níveis de utilização. O serviço foi iniciado em 2005, revolucionando a apresentação de mapas em páginas web, ao permitir os usuários, de forma gratuita, navegar por mapas de elevada integridade. A Google® permite a utilização destes mapas em outras aplicações, através de uma API (Application Programming Interface). Esta API consiste em HTML, CSS e JavaScript funcionando juntos, realizando requisições de informações em servidores à medida que se navega no mapa. A API fornece classes, com métodos e propriedades, que podem ser usadas para controlar o comportamento dos mapas.

6. GitHub: é um serviço de hospedagem para projetos de *software* livre, permitindo a publicação do código-fonte do sistema e o desenvolvimento colaborativo. Além disso, o GitHub permite a distribuição de *softwares web* para o usuário.

7. Firebase: é um serviço *online* que provém uma API para desenvolvedores de *software* armazenar e sincronizar informação através de múltiplos clientes. O *Firebase* utiliza o JSON como estrutura de dados.

## 3. Desenvolvimento do *software* proposto

### *3.1 Etapas de desenvolvimento*

O desenvolvimento se inicia com a definição da arquitetura do software, através da escolha das tecnologias envolvidas e resumo de suas interações, que formam as bases de funcionamento do sistema. Em seguida, é apresentada uma elaboração detalhada da interface gráfica e das funcionalidades gerais do sistema, como manipulação de mapa e grafo, seleção de elementos, abrir e salvar problemas, entre outras.

Após as funcionalidades gerais, segue-se o desenvolvimento de funcionalidades específicas, relacionadas aos métodos de localização e roteirização utilizados e seus algoritmos respectivos. Por último o software é validado comparando os resultados de um problema específico com outros aplicativos, como o LogWare®, o Google® Maps e o Excel®, e disponibilizado para alunos da disciplina. A Figura 1 mostra essas etapas.

**Figura 1**: Etapas do desenvolvimento


Fonte: Autores (2019).

### *3.2 Definição da arquitetura*

A definição das tecnologias a serem utilizadas é uma das primeiras decisões de desenvolvimento. Na escolha de uma linguagem para implementação dos algoritmos, considerou-se que os problemas nesse trabalho não são complexos, sendo possível sua implementação em diversas linguagens. Os problemas são de pequena escala, com foco no aprendizado, e não problemas reais que exigem muito processamento e são melhores implementados em tecnologias de alto desempenho.

Portanto, a capacidade da linguagem de programação a ser escolhida, para implementação dos algoritmos, não é um critério muito relevante. Por outro lado, considerando as características pedagógicas e de interface como requisitos de qualidade, foi necessária uma tecnologia que fornecesse liberdade no desenvolvimento da interface gráfica, e, ao mesmo tempo, robustez, para ser possível explorar ao máximo recursos visuais.

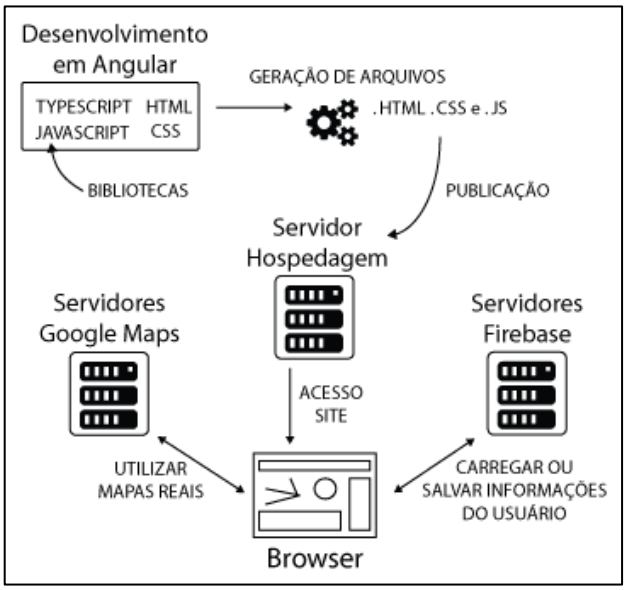
Não são todas as tecnologias de software que fornecem esta liberdade, muitas delas fornecem um conjunto limitado de elementos visuais, com baixo grau de customização. Assim, certos comportamentos desejados não poderiam ser implementados neste trabalho, ou em novas funcionalidades futuras. Outro critério utilizado na escolha das tecnologias é o custo de desenvolvimento, que consiste no tempo e recursos dispendidos. Em seguida, foi definido analisado o requisito portabilidade entre os sistemas operacionais. Após definir uma tecnologia, existe um dispêndio de tempo ao percorrer sua curva de aprendizado. Assim, considerando os critérios abordados foi escolhido no ecossistema Web as tecnologias de HTML, CSS e JavaScript.

No próximo passo foram escolhidos os frameworks ou bibliotecas que serão utilizados. Um mesmo sistema pode ser desenvolvido, com resultado semelhante, em diversos frameworks diferentes. Por ser um framework bastante utilizado e maduro, o Angular foi escolhido para este projeto, acreditando-se ser uma escolha que mantenha o sistema compatível e atualizável por muitos anos. De forma semelhante, o Firebase foi o serviço escolhido para realizar o armazenamento de informações do usuário, como problemas modelados. É um serviço gratuito, baseado em tecnologia Web, que permite armazenar informações no formato JSON.

O GitHub foi utilizado como repositório gratuito de código aberto, o que permite futuros trabalhos colaborativos, além de permitir também a publicação da aplicação direto para o usuário final, sem necessitar de outro serviço de hospedagem de páginas ou contratação de domínios. Outro serviço utilizado foi o Google Maps, que acrescenta funcionalidades interessantes no sistema, ao permitir utilizar mapas reais. É um serviço gratuito para pequenas e médias demandas, que fornece

uma API em JavaScript para sua utilização. A Figura 2 apresenta a arquitetura e as tecnologias definidas.

**Figura 2** - Arquitetura geral do sistema



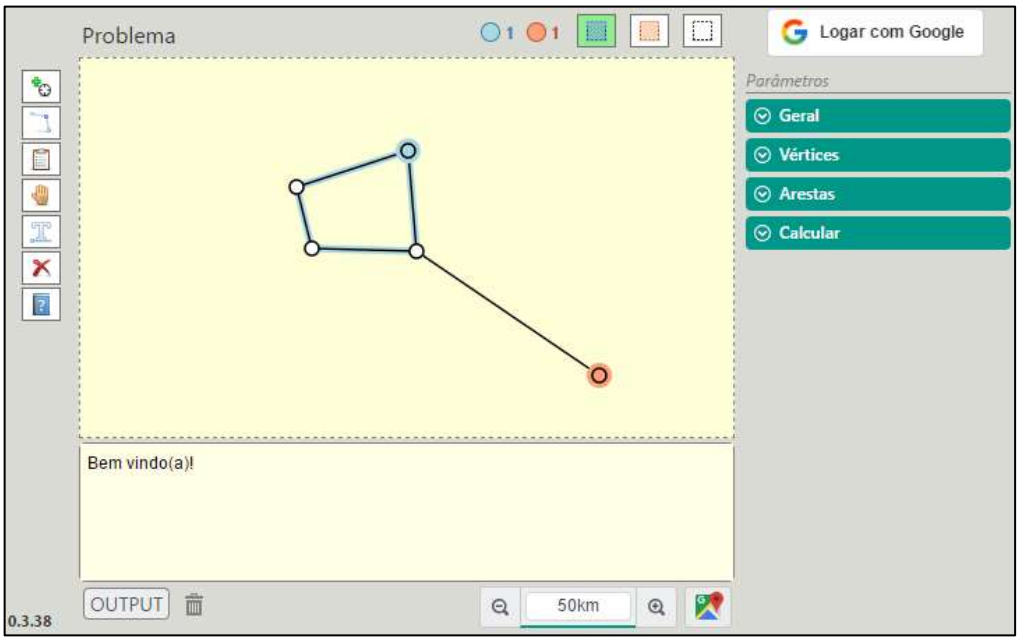

Fonte: Autores (2019)

### *3.3 Interface gráfica e funcionalidades gerais*

#### 3.3.1 Visão geral

Os problemas de localização e roteirização envolvem diversas variáveis de entrada, como custos, taxas, volumes, coordenadas, etc. Dentre essas variáveis, uma estrutura se destaca como elemento central desses problemas, que são as coordenadas das localidades e as relações que estas guardam entre si. Estas informações podem ser representadas visualmente através de grafos, onde os vértices são posicionados de acordo com suas coordenadas geográficas em um mapa imaginário. Assim, este mapa foi definido como elemento central da interface, permitindo ao usuário modelar o problema e visualizar os seus resultados tendo este mapa como referência. Na Figura 3 ele possui fundo amarelo, ao centro, apresentando um grafo de exemplo em seu interior.

**Figura 3** - Visão geral da interface gráfica

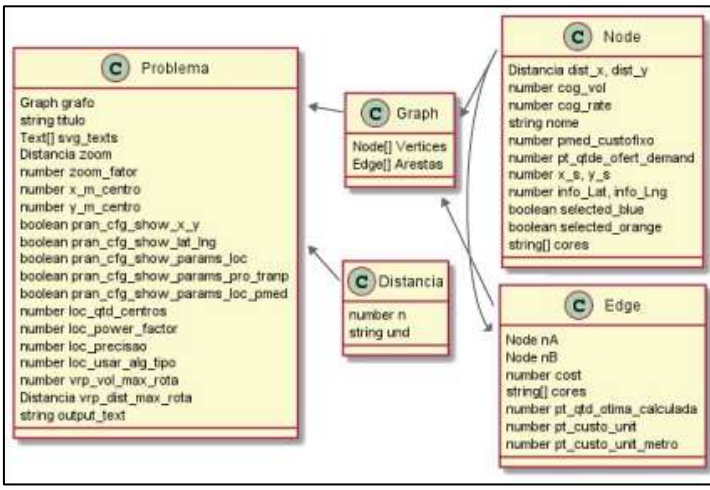



Fonte: Autores (2019).

Para o desenvolvimento da interface, foram utilizadas marcações SVG, que estão sincronizadas com as informações do problema. Assim, se o problema possui um vértice com certas coordenadas, marcações SVG propriamente configuradas irão desenhar um círculo no local correto. Essa sincronização entre as informações do problema e sua representação visual é possível graças às funcionalidades de Data Binding do Angular.

A Figura 4 apresenta um esboço do esquemático geral da entidade. É nesse objeto que são mantidas várias informações do problema modelado, como nome do problema, grafo, vértices, arestas, parâmetros, etc. Várias dessas informações são mantidas em sincronização com a interface gráfica.

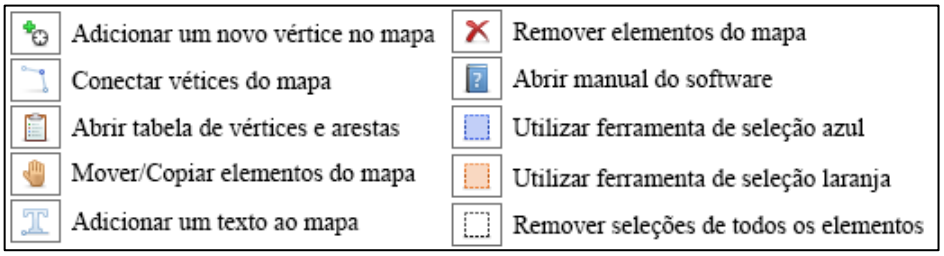
**Figura 4 -** Esquemático geral da entidade




Fonte: Autores (2019).

Para modelar o problema através do mapa e seus elementos, a interface foi desenhada para fornecer, ao usuário, acesso à diversas ferramentas. Utilizando um padrão consagrado de disposição dos elementos visuais, manteve-se o objeto em edição (o mapa) ao centro, enquanto as possíveis ferramentas de edição encontram-se na periferia da interface. A Figura 5 lista os botões e um resumo de suas funcionalidades.

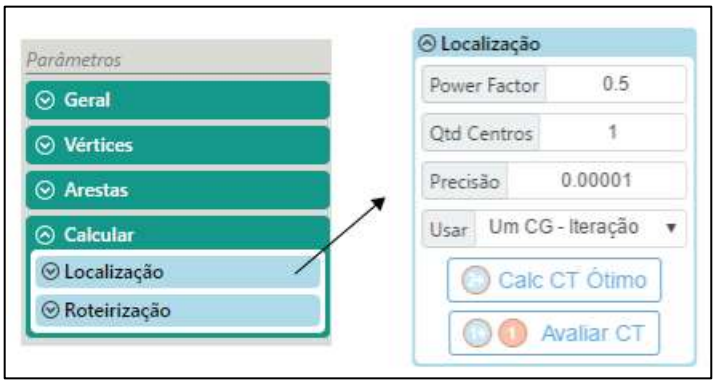
**Figura 5 -** Algumas ferramentas e suas funcionalidades




Fonte: Autores (2019).

O sistema ainda permite outras ações úteis, como movimentação de vários elementos, duplicação de elementos, seleções invertidas, etc. Além das ações executadas por essas ferramentas, existem vários parâmetros configuráveis para se resolver um determinado problema. Determinaram-se duas estruturas na interface com esse propósito. Na primeira, através dos elementos na direita da interface, o usuário pode configurar diferentes parâmetros do problema e executar diferentes ações, como mostra a Figura 6.

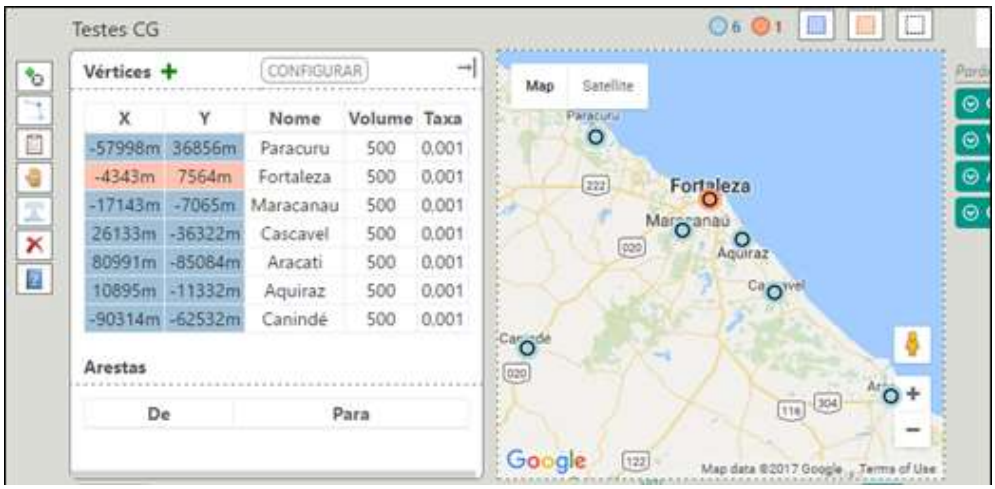
**Figura 6 -** Elementos visuais para configuração de parâmetros




Fonte: Autores (2019)

A segunda estrutura elaborada para permitir a configuração de parâmetros de um problema é uma estrutura de tabelas, que listam todos os vértices e arestas do problema conforme demonstra a Figura 7.

**Figura 7 -** Tabela de vértices e arestas

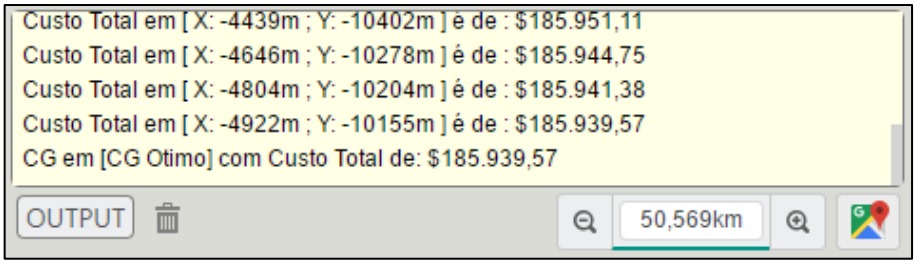



Fonte: Autores (2019).

Seu tamanho horizontal é ajustável para não prejudicar a visualização das informações em problemas de maior escala, assim como as colunas que cada tabela utiliza também são configuráveis de acordo com a aplicação do sistema. Percebe-se que essa abordagem reaproveita toda a estrutura da interface gráfica para as diversas funcionalidades do sistema, permitindo que uma mesma modelagem do mapa possa ser usada para executar diferentes algoritmos e resolver múltiplos problemas.

Na parte inferior da interface, conforme Figura 8, foi colocado um campo de saída (Output), onde o sistema pode imprimir informações variadas, como informações relevantes durante a execução dos algoritmos e seus resultados. Este elemento pode ser ocultado da aplicação se não for utilizado.

**Figura 8 -** Informações de saída




Fonte: Autores (2019).

### 3.3.2 Manipulação do mapa

Algumas funcionalidades envolvendo o mapa central foram adicionadas por questões ergonômicas e educacionais. Dentre elas, está a disponibilização de mapas reais, através da implementação da API do Google® Maps. O usuário pode, a qualquer momento, sobrepor um mapa real ao seu mapa modelado, que compartilha das mesmas coordenadas do grafo do problema. A Figura 9 demonstra essa funcionalidade.

Figura 9 - 1Modelando o problema com mapas reais


Fonte: Autores (2019)

Dessa forma, o usuário pode comparar rotas e localidades, definidas na modelagem do problema, com rotas e localidades reais, ou, ainda, modelar um problema real. A representação em tela utilizada pelo Google® Maps é uma projeção cilíndrica da superfície do planeta, denominada Mercator. Como deseja-se, por motivos educacionais, utilizar um mapa teórico de coordenadas lineares que permite, a qualquer instante, comparações com mapas reais, torna-se necessário uma correlação precisa entre as localidades nos dois mapas, bem como uma representação visual consistente.

Dependendo das dimensões do problema sendo modelado, o espaço disponível em tela pode não ser suficiente para visualização adequada dos elementos do mapa. Assim, para a utilização do sistema, a funcionalidade de enfoque ajustável (zoom), que altera a escala. Além do zoom, elaborou-se um mecanismo de translação (pan) da porção visível do mapa. Através dessas duas funcionalidades, o usuário pode ter livre navegação e visualização do mapa. O zoom e o pan podem ser alterados utilizando diferentes ferramentas, como o botão de rolar do mouse ou alteração direta da escala.

Para implementar essas funcionalidades, é preciso criar um relacionamento entre as coordenadas do modelo, em metros, e as coordenadas dos elementos na interface, em pixels. A visualização em tela do mapa depende do tamanho deste elemento na interface, que é determinado pelo tamanho do dispositivo, da janela do browser, e de como o usuário ajustou os elementos da interface. Essa visualização é apenas uma porção do mapa modelado, que tem dimensões infinitas já que se trata de um plano cartesiano livre, onde um vértice pode ser adicionado em qualquer coordenada.

As Equações 1 e 2 representam as relações entre essas coordenadas, utilizadas na programação do sistema. Assim, sempre que um usuário adiciona um novo vértice em alguma posição $(Xd, Yd)$ do dispositivo, é possível determinar qual a posição $(Xp, Yp)$ correspondente no mapa do problema. Estas equações dependem de variáveis que controlam o zoom e o pan, já que são com elas que o usuário manipula a porção visível, no dispositivo, do mapa do problema.

$$Xp = (Xd - Xdc) * FatorZoom + Xpan \quad (1)$$

$$Yp = (Ydc - Yd) * FatorZoom + Ypan \quad (2)$$

Onde:
$Xp\ e\ Yp$ são coordenadas do problema, em metros;
$Xd\ e\ Yd$ são coordenadas do dispositivo, em pixels;
$Xdc\ e\ Ydc$ são coordenadas, em pixels, do centro da porção visível do mapa, no dispositivo;
$FatorZoom$ é a variável que controla o *zoom*;
$Xpan\ e\ Ypan$ são coordenadas de referência para controlar a translação da porção visível do mapa, no dispositivo.

### 3.3.3 Salvar e carregar problemas

Contribuindo para o objetivo educacional do software, optou-se por viabilizar o armazenamento das informações de um problema modelado para posterior utilização. Essa funcionalidade pode ser útil em diversas situações, como problemas complexos que demoram a serem construídos, ou na reutilização do problema, por um aluno, para eventual apresentação, ou, ainda, a criação, por parte de um professor, de problemas públicos para serem utilizados por alunos.

Salvar informações em arquivos é algo trivial no universo digital, mas a utilização da Web como repositório é uma tendência que foi adotada neste sistema. Assim, ao salvar ou carregar um problema salvo, o sistema faz uso de um serviço online, o Firebase, dispensando os usuários da utilização de arquivos e mídias digitais de gravação.

Por esses elementos da interface, o usuário realiza uma autenticação no sistema, que permite o acesso aos problemas salvos previamente. Estes problemas podem ser excluídos a qualquer momento. Também é possível carregar problemas públicos, configurados através do Firebase, que só podem ser alterados por quem administra o software, recurso com utilidade voltada aos professores.

## *3.4 Métodos de localização*

### 3.4.1 Localização única

Utilizando a estrutura genérica de mapa e grafos foi possível associar diferentes algoritmos objetivando a resolução de problemas de localização. O primeiro foi de localização de instalação única, utilizando o Método do Centro de Gravidade.

Foi realizada a implementação do algoritmo utilizando TypeScript. Como parâmetros de entrada do problema, têm-se um conjunto de vértices selecionados de azul, valores de volume e taxa configurados na tabela de vértices, e parâmetros gerais do problema, definidos na direita da interface

O algoritmo é iterativo e tem como critério de parada a precisão definida nos parâmetros. O resultado do problema consiste em um novo vértice, selecionado de vermelho, adicionado automaticamente ao problema, na localização ótima encontrada pelo método centroide, além de informações, no Output, referentes ao passo a passo da resolução do problema e o valor do custo final. Além de calcular o custo ótimo, o software permite avaliar um custo total para certo vértice específico, utilizando as mesmas taxas e volumes definidos na tabela de vértices, que pode ser comparado com o custo ótimo encontrado.

### 3.4.2 Localização múltipla

Dois métodos foram implementados para localização de instalação múltiplas. O primeiro deles, do múltiplo centro de gravidade, calcula uma quantidade p de centros de gravidade, utilizando o método centroide anterior, para p agrupamentos de vértices. As informações de entrada são os vértices selecionados de azul, a precisão de parada do método centroide, e a quantidade p desejada.

Estes agrupamentos são realizados de acordo com as proximidades entre os vértices, que são calculadas e ordenadas de forma decrescente. A

cada iteração do algoritmo os vértices mais próximos são agrupados, se já não tiverem sido agrupados anteriormente, sendo substituídos por um centro de gravidade. Novas distâncias são calculadas e ordenadas, repetindo o processo de clustering até que restem, somente, p vértices. Estes p vértices finais, que guardam em Array a informação de quais vértices originais lhe deram origem, acabam por determinar quais agrupamentos devem ser feitos. Finalmente, todos os vértices de cada agrupamento são submetidos, pelo algoritmo, ao método centroide, que calcula cada uma das p localidades desejadas.

Outro método disponibilizado foi o da p-mediana que analisa, dentre todos os vértices selecionados de azul, quais p vértices devem ser escolhidos como instalações de origem. A resolução deste método passa pela modelagem e resolução de um problema de Programação Linear (PL). Diferentemente dos métodos anteriores, este método não foi completamente implementado em TypeScript, necessitando de uma biblioteca de código aberto, a jsLPSolver, em JavaScript, que possui esse propósito. Portanto, resta ao código desenvolvido em TypeScript a modelagem do problema de PL de acordo com os requisitos da biblioteca, além da interação com a interface para definir os parâmetros de entrada e apresentar os resultados.

## *3.5 Métodos de roteirização*

### 3.5.1 Melhor rota

O primeiro problema de roteirização implementado consistiu em encontrar a melhor rota entre dois vértices do grafo. A solução deste problema só é possível se existir pelo menos uma rota possível entre estes dois vértices. O algoritmo implementado em TypeScript foi o de Dijkstra. O algoritmo de Dijkistra permite considerar qualquer variável como custo para percorrer uma aresta. Assim, é possível modelar situações mais realistas, considerando estradas desgastadas ou pedágios, e não somente as distâncias cartesianas.

### 3.5.2 Problema do transporte

Neste problema, é preciso definir quantidades transportadas entre origens, com ofertas, e destinos, com demandas, buscando minimizar o custo total. Entre cada origem e destino, existe uma aresta. Assim, vários vértices e arestas, e suas propriedades, são os parâmetros de entrada do problema.

### 3.5.3 Método da varredura

Como parâmetros de entrada deste problema têm-se vértices de seleção azul, determinando os destinos, e um único vértice vermelho, como origem. Além disso, o usuário pode opcionalmente definir restrições de volume máximo e distância máxima percorrida que não podem ser ultrapassados em cada roteiro.

O algoritmo mostra o comportamento do método, de “varrer”, em sentido horário ou anti-horário, todos os vértices, conectando-os, tomando como centro de rotação a origem. Ele calcula para cada vértice de destino, o valor da função $Math.atan2(X_d - X_o\ ,\ Y_d - Y_o)$, onde $(X_o\ ,\ Y_o)$ $e$ $(X_d\ ,\ Y_d)$ representam as coordenadas da origem e do destino, respectivamente. Esse método permitiu definir a inclinação da reta que passa entre cada combinação origem/destino. Ordenando os valores encontrados em ordem crescente ou decrescente, permite simular o comportamento proposto de “varredura”.

### 3.5.4 Método das economias

O método das economias teve como parâmetros de entrada o volume máximo e distância máxima para cada rota. O algoritmo cria uma rota inicial para cada vértice de destino (azul), contendo somente o respectivo vértice e a origem (vermelho). Em seguida, calcula-se, para todas as combinações 2 a 2 de vértices de destino, a suposta economia que se teria ao agregá-los em uma mesma rota. Essas economias formam uma lista, ordenada em forma decrescente. O algoritmo então percorre essa lista, agrupando os vértices, quando possível, segundo algumas restrições do método e as restrições de rota configuradas pelo usuário. É possível mostrar visualmente o funcionamento de cada passo do algoritmo, para que o usuário observe o método durante a formação das novas rotas.

### *3.6 Verificações*

As funcionalidades logísticas apresentadas anteriormente passaram por testes de verificação para atestar o funcionamento do programa. Eventualmente, alguns erros foram encontrados, levando a correções na programação dos algoritmos ou da lógica da interface. A metodologia de verificação do programa utilizada foi a comparação com os sistemas LogWare®, Google® Maps e Excel® a partir da resolução de problemas educacionais selecionados de livro-texto. Esses programas são bastante utilizados na área educacional do ensino de logística ao longo dos últimos anos. A partir deste ponto, o software proposto será referenciado como kLog, para facilitar a leitura do texto.

#### 3.6.1 Métodos de localização

Para os métodos de localização, problemas solucionados do livro-texto do Ballou (2006) e comparados com o programa LogWare®. O programa, na sua versão 6.0, é distribuído com problemas de exemplo que utilizam o métodos de localização.

Durante a resolução do Método do Centroide (COG), os dois softwares apresentam algumas diferenças. O LogWare® pergunta ao usuário, a cada iteração do algoritmo, se este deseja continuar sua execução, até que esteja satisfeito com a precisão do resultado. Já o kLog recebe esta precisão como um parâmetro de entrada, sendo este o critério de parada para a execução do algoritmo. Para o problema exemplo aqui tratado, a precisão foi definida como 0,00000001.

Os dois softwares obtiveram resultados de forma diferente, mas com valores satisfatoriamente próximos, mesmo neste problema com volumes elevados. As coordenadas do centro de gravidade convergem para um ponto comum, ao passo que o custo total difere em poucas unidades após ciclos de iteração.

Outra diferença entre os dois sistemas está nas possibilidades de ação do usuário após a resolução do problema. No LogWare®, o usuário fica limitado à visualização dos valores e coordenadas finais. No kLog, pode-se reaproveitar estas coordenadas em ações subsequentes, como busca de um centro de gravidade viável.

De forma semelhante, problemas exemplos foram comparados entre o kLog e LogWare®, para verificação do Método do Múltiplo Centro de Gravidade (MULTICOG).

O Método da p-mediana teve a verificação realizada também por comparação de resultados entre kLog e LogWare®. Uma diferença observada entre os dois sistemas, quando o LogWare® resolve problemas utilizando latitudes e longitudes, diz respeito a consideração da taxa. Enquanto o kLog considera uma taxa unitária por metro, o LogWare® considera uma taxa unitária por milha. Assim, a

comparação neste caso só foi possível ao considerar uma conversão de 1.609,34 metros por milha.

### 3.6.2 Métodos de Roteirização

Para atestar a funcionalidade do Método do Caminho Mínimo, desenvolvido com o algoritmo de Djikistra, foram feitas comparações manuais entre os resultados do kLog e de outros sistemas, como o *LogWare*® e o *Google*® *Maps*, retornando resultados satisfatórios. No teste de verificação foram utilizadas as principais cidades da região Norte do Estado do Ceará. Observou-se a semelhança de roteiro e no cálculo da melhor rota entre a cidade de Sobral-CE e o porto do Mucuripe, em Fortaleza-CE.

A confiabilidade do kLog na resolução do Problema do Transporte está diretamente ligada à confiabilidade da biblioteca jsLPSolver, a solução escolhida para resolver problemas de Programação Linear (PL) utilizando JavaScript. Assim, resta ao kLog atestar a qualidade do método através da correta modelagem do problema e do funcionamento adequado da interface gráfica, além da correta utilização da biblioteca.

O software utilizado como referência de comparação foi o Excel®, que é capaz de resolver, através do suplemento Solver, problemas de PL, como o problema do transporte.

Parte da verificação do Método de Varredura se deu de forma manual e visual. Em qualquer problema modelado, pode-se determinar, visualmente, qual a ordem de conexões que serão feitas entre os vértices, dada a natureza simples do método. A outra parte da verificação diz respeito as restrições de volume total e distância total de cada roteiro, que foram respeitadas nos testes modelados.

De forma semelhante, a verificação das restrições nos testes com o Método das Economias foi feita através da análise dos resultados e constatação de rotas dentro dos limites configurados. Contudo, o roteiro definido por este método não pode ser analisado de forma manual, dado sua complexidade, precisando ser comparado com fontes confiáveis. O LogWare® utiliza o mesmo algoritmo em seu módulo ROUTESEQ. Porém, neste módulo, ele não aceita parâmetros de restrição, limitando-se a gerar um caminho que é a solução heurística do problema do caixeiro viajante. Dado que as restrições já foram verificadas através de métodos manuais, limitou-se a comparar o resultado das rotas geradas pelo kLog, sem restrições, com aquelas geradas pelo LogWare®.

### 3.6.3 Discussão e resultados das verificações

Os vários testes de verificação realizados estão dentro das limitações de recursos deste trabalho. Eles foram importantes para apresentar problemas que foram corrigidos durante o desenvolvimento. Algumas imperfeições encontradas ainda precisam ser corrigidas em versões futuras, mas as funcionalidades foram mantidas, pois os resultados foram considerados estáveis e satisfatórios o suficiente para a versão inicial do sistema.

**Quadro 1 -** Resumo dos resultados das verificações realizadas

| Método | Resultados das Verificações Realizadas |
|---|---|
| Método de Localização Única (LogWare® *Versus* klog) | Os dois programas apresentaram os resultados de forma diferente, mas com valores satisfatoriamente próximos. As coordenadas do centro de gravidade convergem para um ponto comum, ao passo que o custo total difere em poucas unidades após alguns ciclos de iteração. |

| | |
|---|---|
| Método de Localização Múltipla (LogWare® *Versus* klog) | Foi possível perceber diferenças nos resultados encontrados pelos sistemas, apesar de mínimas. O custo total diferiu em torno de 0,01%, enquanto as coordenadas variaram entre 0,08% (4.996m e 5km) e 2,06% (1.021m e 1km), aproximadamente. O kLog tem precisão de 1 metro para todas suas distâncias, e tem a precisão critério de parada do método centroide configurada pelo usuário. Os detalhes do algoritmo interno do *LogWare*® são desconhecidos, e ele pode estar efetuando arredondamentos. Contudo, como as diferenças encontradas em repetidos testes se mantiveram mínimas, a funcionalidade foi mantida no kLog, passível de melhorias futuras. |
| Método de Localização da p-mediana (LogWare® *Versus* klog) | Na comparação, em ambos os programas foram escolhidas as mesmas cidades locais de instalações de fornecimento para as demais. O agrupamento realizado também foi idêntico. Contudo, o custo final calculado diferiu em quase 17%. Essa diferença elevada se deve a dois fatores: o primeiro, menos relevante, foi o método de localização das cidades para formulação do problema equivalente no kLog. Enquanto o *LogWare*® definiu latitudes e longitudes específicas, a localização no kLog foi feita de forma visual pelo nome das cidades. A segunda diferença é a limitação já comentada, no kLog, quando se utiliza o *Google® Maps* em elevadas latitudes, como neste problema exemplo. Dado essas duas fontes de erro, as coordenadas utilizadas pelo algoritmo ficaram bem distorcidas em relação às coordenadas do *LogWare*®, mas não distorcidas para alterar a alocação dos vértices. Contudo, outros testes comparativos realizados com menores latitudes e uma quantidade menor de vértices diminuíram consideravelmente as diferenças. Além disso, esse problema não é encontrado quando se trabalha, em ambos os sistemas, com coordenadas lineares, não necessitando do *Google® Maps* para localização visual. |
| Roteirização Método do Caminho Mínimo (*Google® Maps Versus* klog) | Nos resultados obtidos, os mapas do *Google® Maps* apresentaram elevada continuidade e precisão. Não são somente as vias principais (modeladas no kLog) que podem ser utilizadas, mas quaisquer vias mapeadas pela Google. As vias possuem sinuosidades e elevações reais, consideradas nos algoritmos internos do *Google® Maps*. Por outro lado, a modelagem do kLog é simplificada, limitando-se a vértices que representam as principais cidades, interligadas por arestas totalmente retas. Somente as principais vias estaduais e federais foram modeladas. Mesmo com estas limitações, a rota calculada pelo kLog foi bem semelhante à do *Google® Maps*, contendo praticamente as mesmas cidades e vias utilizadas. |
| Roteirização Problema do Transporte (Excel® *Versus* klog) | Os valores obtidos de quantidade de oferta e demanda e o custo total mínimo em ambos os programas Excel® e kLog foram iguais. |
| Roteirização Método da Varredura (LogWare® *Versus* klog) | Os dois programas obtiveram resultados similares com relação as restrições de volume total e distância total de cada roteiro obtido. |
| Roteirização Método das Economias (LogWare® *Versus* klog) | Foi possível perceber algumas diferenças no roteiro, e o resultado no kLog é levemente melhor (rota aproximadamente 1,56% menor) que o resultado do *LogWare*® para o mesmo problema. Este é um resultado dentro da normalidade, assim como outros que foram encontrados. O método das economias é uma heurística, não necessariamente encontrando o resultado ótimo, e possuindo algumas fontes de aleatoriedade. |

Fonte: Autores (2019)

## 4. Conclusão

Este trabalho propôs um software educacional para apoiar o ensino de localização e roteirização em disciplinas de logística. A justificativa deste objetivo foi apresentada frente a relevância das tecnologias digitais nos processos de ensino.

Disponibilizou-se um embasamento teórico relacionado aos assuntos abordados, iniciando-se com a apresentação dos métodos de localização e

roteirização que foram utilizados, seguindo-se da conceituação das tecnologias necessárias ao desenvolvimento do sistema.

Inicialmente foram determinadas quais tecnologias de desenvolvimento de software eram as mais adequadas, foi possível através de uma ponderação das opções de tecnologias disponíveis e de uma tomada de decisão alinhada com requisitos de qualidade que considerem o objetivo educacional do sistema. Em seguida foram implementados os algoritmos capazes de resolver problemáticas comumente tratadas no ensino de localização e roteirização através da elaboração de algoritmos em TypeScript, com eventual utilização da biblioteca jsLPSolver, baseando-se em formulações detalhadas destes métodos fornecidas pelos autores referenciados no embasamento teórico. Posteriormente foi desenvolvida uma interface gráfica. O desenvolvimento dessa interface foi constantemente orientado pela utilização educacional do sistema, apresentando como elemento principal um mapa central no qual um grafo pode ser modificado utilizando-se diversas ferramentas desenvolvidas de forma ergonômica e lúdica. Utilizando-se estes elementos visuais, pode-se modelar problemas de roteirização e localização resolvidos através dos algoritmos implementos sendo os resultados devidamente apresentados na interface.

Por último foi realizada a verificação dos algoritmos implementados através da aplicação dos métodos de localização e roteirização na resolução de problemas com soluções previamente conhecidas. Para tanto, utilizou-se de variadas fontes de problemas, como aqueles que acompanham o LogWare®. As soluções obtidas com o sistema desenvolvido foram comparadas com estas fontes (LogWare®, Google® Maps e Excel®) e as vantagens e limitações do sistema foram ressaltadas.

Dado o foco de utilização educacional do sistema, algumas imprecisões encontradas não foram de grande impacto em sua qualidade, já que não há a pretensão do seu uso em situações reais críticas e profissionais, que envolvem centenas ou milhares de variáveis. Encontra-se maior relevância para a qualidade, nas verificações, ao se comparar a existência de recursos visuais e funcionalidades com foco didático, em relação a outros softwares que também resolvem os problemas modelados.

O foco deste trabalho foi o desenvolvimento e a verificação do programa, a próxima etapa será a realização de pesquisas de validação com usuários do sistema, capazes de investigar a percepção de alunos e professores quanto à utilidade lúdica do software, possibilitando a correção de problemas, melhoraria de limitações e desenvolvimento de novas funcionalidades.

## Referências

Arenales, M., Armentano, V., Morabito, R., & Yanasse, H. (2007). Pesquisa Operacional para cursos de Engenharia. Rio de Janeiro: Campus.

Ballou, R. H. (2006). Gerenciamento da Cadeia de Suprimentos/Logística Empresarial. 5 ed. Porto Alegre: Bookman.

Bass, L., Clements, P., & Kazman, R. (2003). Software Architecture in Practice. 2 ed. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Bortolossi, H. J. (2012). Criando conteúdos educacionais digitais interativos em matemática e estatística com o uso integrado de tecnologias: GeoGebra, JavaView, HTML, CSS, MathML e JavaScript. Revista do Instituto GeoGebra Internacional de São Paulo. 1(1), XXVIII - XXXVI.

Bowersox, D. J. & Closs, D. J. (2001). Logística empresarial: o processo de integração da cadeia de suprimentos. São Paulo: Atlas.

Brooks, D. R. (2007). An Introduction to HTML and JavaScript for Scientists and Engineers. London: Springer-Verlag.

Bubeck, S. (1992). Applications Programming in Smaltalk-80: How to use Model-View-Controller. Smalltalk-80. 2(5), 01-11.

Cruz, A. G., & Neri, D. F. (2014). A inserção de tablets em escolas da rede pública estadual na cidade de Petrolina-PE: uma percepção dos educadores educandos. Revista de Educação do Vale do São Francisco - Revasf, Petrolina, 4(6), pp.06-26.

Cunha, C. B. (1997). Uma contribuição para o problema de roteirização de veículos com restrições operacionais. São Paulo. Tese (Doutorado). Escola Politécnica da Universidade de São Paulo.

Fernandes, J. (2002). O que é um programa (Software). Recuperado em 25 de Junho, 2017: <http://www.cic.unb.br/~jhcf/MyBooks/iess/Software/oqueehsoftware.html>.

Fitzsimmons, J., & Fitzsimmons, M. (2004). Administração de serviços: operações, estratégias e tecnologia da informação. 4 ed. Porto Alegre –RS: Bookman.

Flanagan, D. (2006). JavaScript: the definitive guide. O'Reilly Media, Inc.

Georges, M. R. R, & Seydell, M. R. R. (2008). Dificuldades no ensino da logística. In: V CONVIBRA–Congresso Virtual Brasileiro de Administração.

Korva, J. (2016). Developing a web application with Angular 2: Graphical editor for Happywise's Cove Trainer.

Laporte, G., Gendreau, M., & Semet, J. Y. P. F. (2002). Classical and modern heuristics for the vehicle routing problem, International Transactions in Operational Research, 7(5), 285-300.

Leite, R. C. G. (2007). Um framework para automação/integração do processo de desenvolvimento de projetos de estruturas reticuladas tridimensionais. Belo Horizonte. Tese (Doutorado). Escola de Engenharia da Universidade Federal de Minas Gerais.

Lorena, L. A. N., Senne, E. L. F., Paiva, A. C., & Pereira M. A. (2001). Integração de modelos de localização a sistemas de informações geográficas. Revista Gestão & Produção, 8(2), 180 - 191.

Magedanz, A. (2004). Computador: Ferramenta de trabalho no Ensino (de Matemática). 14f. Curso de Pós-Graduação Lato Sensu Univates. Recuperado em 20 de Junho, 2017: < http://ensino.univates.br/~magedanza/pos/artigo_final_adriana_magedanz.pdf >.

Mozilla. Developer Network. (2017). Recuperado em 25 de Maio, 2017: < https://developer.mozilla.org > .

Novaes, A. G. (2007). Logística e gerenciamento da cadeia de distribuição: estratégia, operação e avaliação. 3 ed. Rio de Janeiro: Elsevier.

Pressman, R, & Maxim, B. (2016). Engenharia de Software. 8 ed. Brasil: McGraw Hill.

Slack, N., Chambers, S., & Johnston, R. (2002). Administração da Produção. 2 ed. In: São Paulo: Editora Atlas.

Taha, H. A. (2008). Sistemas de Filas. 8 ed. São Paulo: Person Prentice Hall.

Von Atzingen, J., Da Cunha, C. B., Nakamoto, F. Y., Ribeiro, F. R., & Schardong, A. (2012). Análise comparativa de algoritmos eficientes para o problema de caminho mínimo. Universidade de São Paulo (USP). São Paulo. Escola Politécnica.

Wanke, P. (2003). Logística e Gerenciamento da Cadeia de Suprimentos. São Paulo. Editora Atlas.

**Recebido em: 07 mar. 2018 / Aprovado em: 19 abr. 2018**

**Para referenciar este texto**

Almeida, J. A. G. de Jr., Pontes, H. L. J. P., & Albertini, M. R. (2019). Desenvolvimento de um software educacional para apoio ao ensino de localização e roteirização. *Exacta*, *17*(3), 81-99. https://doi.org/10.5585/ExactaEP.v17n3.8444.