Dispositivos em Rede: Uma Aplicação Desenvolvida Em Java

Everton Bruno Silva dos Santos[1]

[1]Universidade Estadual de Feira de Santana (UEFS)
Caixa Postal 44036-900 – Feira de Santana – BA – Brazil
Av. Transnordestina, s/n, Novo Horizonte

evertonbrunosds@gmail.com

Resumo. *Visando modernizar o modo como uma empresa de tecnologia atua no âmbito das rede computacionais, uma a aplicação foi implementada para que tenha uma interface gráfica simples, intuitiva e de fácil entendimento afim de seguir e atender todas as diretrizes estabelecidas na descrição do problema enfrentado pela WillFall.*

1. Introdução

A medida em que o mundo se moderniza é natural que as pessoas passem a usufruir dos aparatos tecnológicos que encontram-se a disposição no mercado. Cada vez mais, tornam-se nítidos os efeitos de um mundo globalizado, tendo em vista que a transmissão de informação nunca foi tão instantânea como nos dias de hoje. Dentre vários métodos disponíveis de se efetuar essa tarefa é indiscutível que o mais utilizado seja a internet, pois graças a ela podemos nos comunicar com qualquer pessoa localizada em qualquer lugar do mundo.

Ainda que tenham indivíduos que façam parte disso utilizando-se desses recursos afim de atender suas necessidades de natureza pessoal, deve-se destacar que as organizações empresariais também fazem parte disso, para tal, em ambos os casos o processo se dá por meio de dispositivos interligados à alguma rede computacional, seja ela uma rede local, seja ela a rede mundial de computadores.

Assim, a empresa WilFall, prestadora de serviços ligados a rede de dispositivos computacionais, preocupada com a alta competitividade existente em seu ramo de atuação decidiu facilitar a forma como seu negócio é realizado. Para tal, a mesma pretende por meio de um sistema detentor de uma interface gráfica de fácil uso, realizar a criação e manutenção de redes computacionais afim de atender principalmente o público das organizações empresariais.

Em meio a esse cenário que compreende as redes de computadores, é comum que o número de dispositivos conectados seja indeterminado e variável, porém, no caso da WillFall, em todos os seus trabalhos o número mínimo de computadores é sempre de 30 *nodes*, podendo se estender para algo superior a isso. Ademais, algo relevante que merece destaque é o modo como os *nodes* se comportam na rede em que fazem parte. Assim, roteadores exercem função única, a medida em que atuam como pontes de acesso que visam promover a comunicação entre os terminais, enquanto que os terminais além de exercerem a mesma função dos roteadores, podem atuar não só como ponto de partida, mas também de chegada de dados, ou seja, os terminais exercem função dupla.

Este relatório, tem por objetivo descrever a aplicação que foi desenvolvida na linguagem de programação Java para efetuar criação e manutenção das redes computacionais descritas, afim de atender as demandas da empresa WillFall.

2. Metodologia

Ao longo do desenvolvimento do projeto houve um considerável número de sessões em sala, essas que puderam contribuir significativamente para a implementação desse projeto. Nesse aspecto, dentre todas a discussões a mais relevante trata do núcleo do problema, que nesse caso refere-se ao uso e implementação de uma estrutura de dados conhecida como grafo, logo, desvendar do que se trata essa estrutura de origem matemática foi algo extremamente essencial.

No decorrer do desenvolvimento da aplicação descrita inicialmente na introdução desse relatório, formam utilizados métodos de implementação que visam reduzir grandes tarefas em várias outras de menor complexidade, portanto, esse processo foi efetuado a cada etapa de desenvolvimento da solução até que a mesma tivesse sido concluída por completo. Assim, o processo de implementação adotado na resolução das demandas apresentadas podem ser descritas da seguinte forma:

1. Implementação de um grafo.
2. Implementação do algorítimo de menor caminho.
3. Implementação do algorítimo de distância euclidiana.
4. Compreensão e elaboração de interfaces gráficas em java.

Compreensão e implementação de um grafo: ao longo das sessões a estrutura de dados grafo apresentou-se como uma ótima forma de solucionar o problema que compreende a estabelecer uma rede computacional. Contudo, deve-se ter em mente que essa estrutura não se limita a isso, portanto, a partir do momento em que ela busca trabalhar com pontos interligados em um plano, a mesma pode ser aplicada a uma diversidade de outras situações, dentre elas, uma muito comum gira entorno de sua utilização em mapas [Lafore 2004].

Na implementação do grafo surgiram duas possibilidades para se armazenar os incontáveis vértices e arestas, desse modo, ficou estabelecido que seriam usadas estruturas dinâmicas disponíveis na API do java, assim a princípio foi utilizada a *ArrayList* que logo foi substituída pelo *HashMap*. Essa decisão foi tomada devido vantagem contida no *HashMap* que consiste em tornar-se dispensáveis a criação de métodos de busca necessários para se trabalhar em certas situações com os *ArrayList*.

Nesse sentido, a implementação do grafo ocorreu da seguinte forma: o grafo contém em si incontáveis vértices, os vértices por sua vez possuem incontáveis arestas e as arestas possuem um peso que se dá por meio de um valor inteiro e uma referência para um outro vértice. Ademais, foram implementados métodos que nos permitem adicionar e remover vértices e arestas, bem como métodos que visam marcar por meio de uma propriedade *"boleana"* as arestas que fazem parte do melhor caminho entre dois vértice, entre outros métodos.

Implementação do algorítimo de menor caminho: No que se refere a solução de um determinado problema, é comum nos depararmos com uma situação em que podemos

nos utilizar de uma diversidade de resoluções pré-existentes. Contudo, nada nos impede de buscarmos nossas próprias alternativas em um processo que pode ser compreendido por muitos como o "reinvento da roda". Uma vez que, qualquer solução pode ser repensada afim de atender uma determinada demanda específica.

Assim, impulsionado pelo ideal de autonomia e liberdade em repensar uma resolução própria para o problema, o algoritmo de menor caminho foi implementado tendo como base a observação de grafos, ou seja, em um primeiro momento a solução se deu por meio de um modo de desenvolvimento marcado pelo empirismo, onde era necessário descobrir todos os caminhos possíveis entre dois pontos, para só então avaliar qual deles era o menos custoso.

Nesse sentido, após ser implementado, o método de menor caminho apresentou-se efetivo em sua funcionalidade encontrando o menor caminho em todos os testes realizados, porém, o mesmo sofria de sérios problemas ligados a eficiência, uma vez que, a medida em que o número de conexões aumentava, o tempo necessário para se achar o menor caminho crescia exponencialmente. Então, afim de resolver esse problema, o algoritmo de Dijkstra foi adotado como solução definitiva.

Implementação do algorítimo de distância euclidiana: visando atender mais um dos requisitos do problema descrito ao longo do desenvolvimento desse relatório, foi necessário elaborar uma forma de se calcular a distância euclidiana entre dois dispositivos contidos na interface gráfica. Porém, para nos certificarmos de que esse cálculo seria efetuado de maneira correta, foram realizadas pesquisas em fontes de nível acadêmico que tratassem do assunto.

Assim, por meio de uma pesquisa foi descoberto que, a distância euclidiana pode ser calculada por meio da aplicação do teorema de Pitágoras conforme descrito em [Tartaglia 2016]. Desse modo, para dois pontos em um espaço com duas dimensões, bastou-se obter a diferença entre os eixos X e Y de dois dispositivos exibidos na interface da aplicação para se ter um triângulo, para só então aplicar o teorema.

Ademais, na aplicação desenvolvida o processo descrito se dá por meio da seleção de dois dispositivos, para tal, por questões de coerência só é possível exibir o painel de opções no caso de haver mais de um dispositivo disponível em rede. Além disso, um outro ponto de relevância, refere-se ao fato de que na aplicação, por questão de coerência, não é possível selecionar o mesmo dispositivo como ponto de partida e chegada afim de realizar o cálculo euclidiano.

Compreensão e elaboração de interfaces gráficas em java: o desenvolvimento das interfaces gráficas no java podem ser feitas por meio do auxilio de ferramentas disponíveis em algumas IDE's da linguagem. Desse modo, o NetBeans se faz extremamente útil nesse aspecto, uma vez, que nos oferece recursos de componentes visuais prémoldados sem que haja a necessidade de implementa-los.

Assim, por meio de pesquisas realizadas na documentação oficial da [Oracle 2020], foi possível determinar os modos de se usar componentes como JOptionPane, JMenu, JFrame, JPane e entre outros úteis a aplicação. Contudo, na construção do grafo, houve uma situação em que não foram encontrados componentes que pudessem representar com fidelidade dos roteadores e terminais da rede.

Para tal, foi desenvolvida uma classe que herda de uma JLabel todos os seus métodos, porém, conta com a capacidade de receber ícones que correspondem visualmente ao que cada *node* representa na rede no momento de seu instanciamento. Desse modo, através da interface o usuário pode interagir com a aplicação e por meio de seus ícones é possível diferenciar os roteadores dos demais dispositivos conectados a rede.

3. Resultados e Discussões

A solução desenvolvida conta com a possibilidade de realizar todas as tarefas demandadas pela WillFall na descrição do problema, contudo, antes de mais nada é necessário que no computador em que a aplicação será usada é preciso haver o JDK8 ou superior devidamente instalado, sendo essa uma de suas condições de uso.

Referente a isso, uma outra condição, diz respeito a integridade dos arquivos carregados para memória, pois no caso de algum deles não seguir o padrão proposto da descrição do problema a aplicação apresentará falhas. Contudo, graças aos arquivos citados, o software conta com a possibilidade de carrega-los e grava-los em memória. Caracterizando-se como algo que permitirá que aos usuários a capacidade armazenar todo o progresso alcançado na criação e manutenção de uma determinada rede.

Ao longo da excussão dos testes de natureza manual, verificou-se que a aplicação está preparada para suportar diversos casos de uso, dentre eles, um gira entorno da possibilidade do usuário digitar letras ou valores incorretos no momento de estabelecer o peso de um dada conexão assim, o processo descrito se dá por meio do tratamento de exceções.

Ademais, o grafo, por questões de coerência e praticidade é exibido com seus vértices de forma circular, afim de evitar que uma aresta cruze um vértice que não lhe pertence. Essa ideia foi extensamente debatida nas discussões em sala, contudo, outros colegas decidiram adotar outros métodos de resolução, mas que não foram implementados na aplicação descrita nesse relatório tendo em vista a dificuldade de se trabalhar com mais de um arquivo. Assim, ao fim do projeto o resultado da interface gráfica pode ser observado na (figura 1).

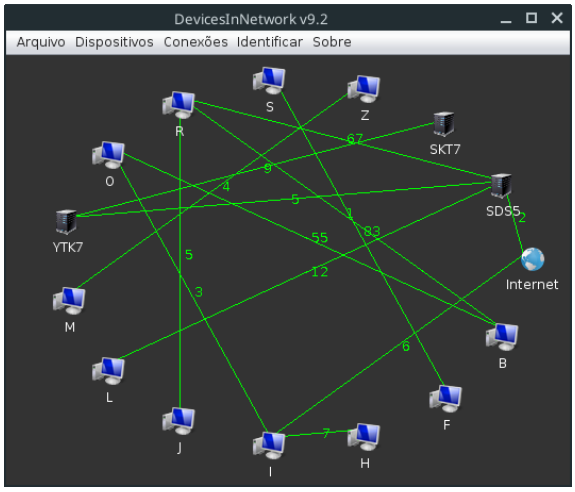

Figura 1. Grafo Circular Representando Rede Computacional.

Um outro ponto relevante da aplicação, refere-se ao modo como o algoritmo de Dijkstra é exibido, sendo assim, no que se refere a exibição do menor caminho entre dois

pontos, ao invés de adotar a ideia padrão de se exibir o melhor caminho destacando-se a melhor rota entre dois dispositivos, por questões de conveniência, adotou-se a mudança de cor das arestas como forma de destaque. Algo que pode ser observado na (figura 2). Já a exibição de menor caminho de um dado terminal para todos os demais ocorreu por meio do uso de um *JOptionPane* que é exibido assim que o usuário clina em algum dos dispositivos conforme a (figura 3).

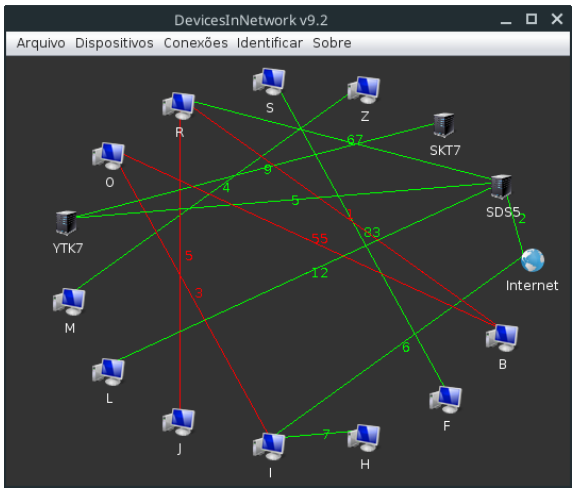

Figura 2. Exibição de Menor Rota Entre Dois Dispositivos.

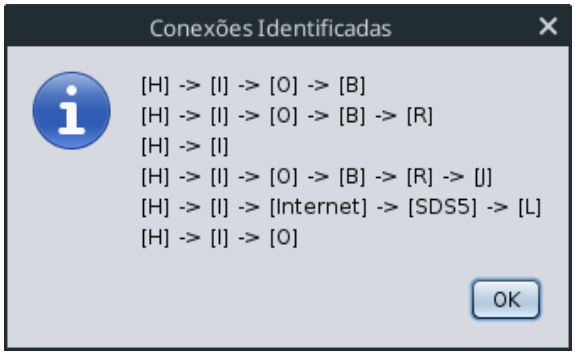

Figura 3. Exibição de Menor Rota Entre Um Dispositivo e Todos os Demais.

4. Conclusões

Este relatório abordou trechos da implementação desenvolvida visando atender as demandas da empresa WillFall que atua na área de criação e manutenção de redes computacionais, esta empresa solicitou o serviço visando facilitar o trabalho de seus funcionário ao fazer com que os mesmos passem a trabalhar com interfaces gráficas ao desempenharem suas tarefas. Para que por meio disso, ao fazer parte dos contemplados pela praticidade proporcionada pela modernidade a WillFall possa se manter líder de mercado.

Assim, a solução implementada conforme mostrado ao longo do relatório conta com uma janela única onde ficam disponíveis todas as suas funcionalidades, essas que exercem suas tarefas com eficiência e praticidade por meio de uma interface gráfica. Ademais, testes foram realizados extensivamente afim de comprovar a efetividade de cada

uma das funções disponíveis na aplicação, afim de evitar não só o retrabalho, mas também, problemas a empresa que solicitou o serviço.

Os pontos positivos desse software correspondem ao modo como os dados são importados, processados, armazenados e exibidos aos usuários, uma vez que, considerando em cada ação as demandas solicitadas pela WillFall em seu problema descritivo. Todavia, os pontos de melhoria desse software giram em torno do fato de que trabalhar com arquivos a primeira vista um boa ideia, porém, atualmente o mercado da programação já dispõem de soluções mais eficazes, a exemplo disso, os arquivos podem ser substituídos por um baco de dados no futuro.

Referências

Lafore, R. (2004). *Estrutura de Dados Algoritmos em Java*. Ciência Moderna.

Oracle (2020). Java documentation. Disponível em: <https://docs.oracle.com/en/java>. Acesso em: 14 Fev 2020.

Tartaglia, L. (2016). Teorema de pitágoras : Aplicações de demonstrações em sala de aula. Disponível em: <https://repositorio.ufscar.br/bitstream/handle/ufscar/8566/DissLTF.pdf?sequence=1&isAllowed=y>. Acesso em: 22 Ago 2020.