Conexão de terminais e roteadores

Evelyn Suzarte Fernandes

Engenharia da Computação – Universidade Estadual de Feira de Santana (UEFS)  
Avenida Transnordestina s/n – 44036-900 – Feira de Santana – BA – Brasil

evelynsuzarte@hotmail.com

**Resumo.** Este relatório tem como objetivo descrever como foi desenvolvido o software para a empresa WillFall, que tem como função principal fazer as conexões numa grande rede entre terminais e roteadores, adicionando e removendo pontos através da interface gráfica ou através de um arquivo de texto onde fica armazenado as configurações dos terminais.

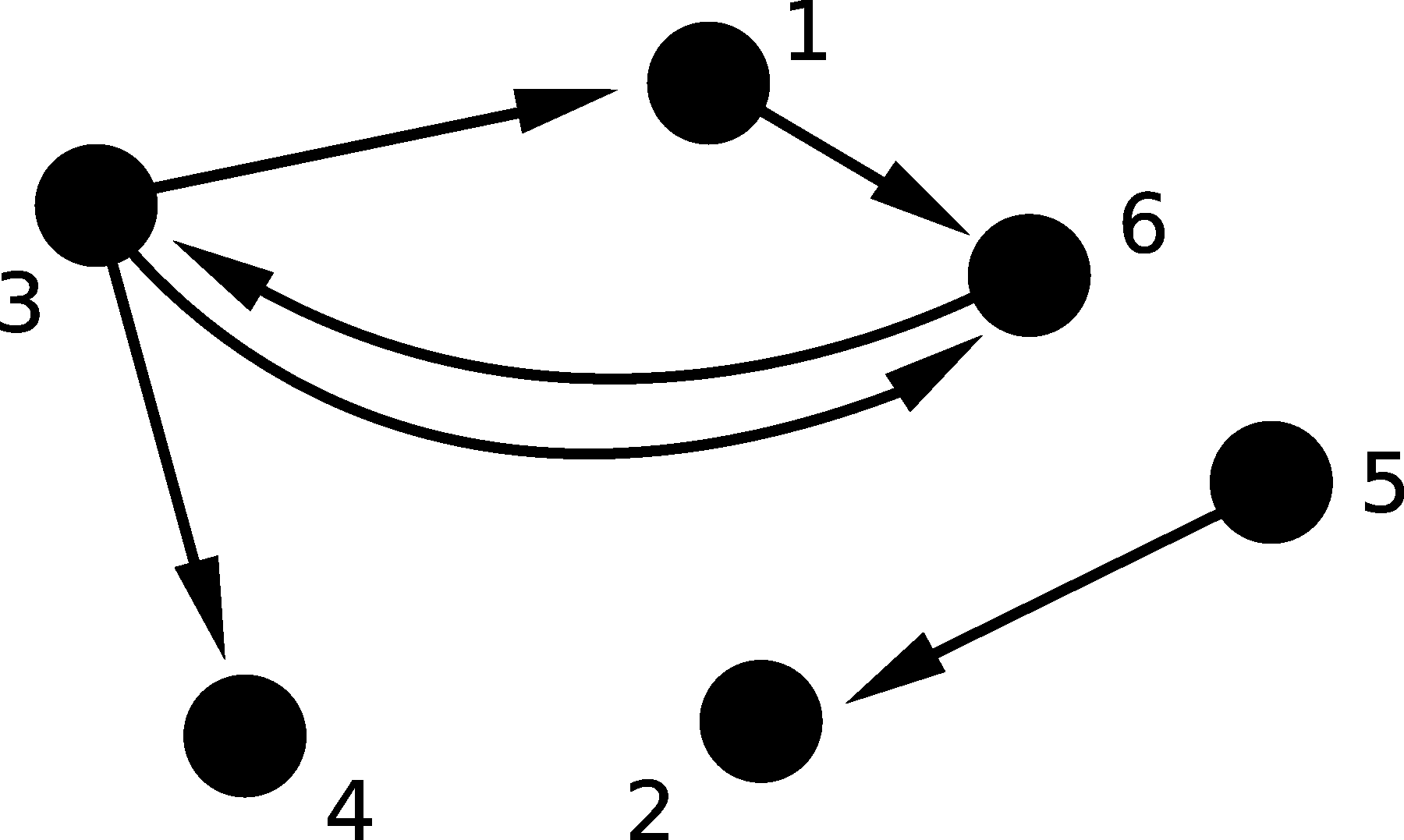
# 1. Introdução

A WillFall atua como prestadora de serviços na área de redes, montando, configurando e otimizando redes, principalmente empresariais. Ela busca um software inovador onde deseja que toda rede mantida pela empresa utilize roteadores com um firmware doméstico e computadores com softwares de rede personalizado.

Sendo assim, a empresa pediu para que os alunos do MI Programação desenvolvessem um software onde os roteadores executem sua função única de encaminhadores (encaminhar pacotes) e os computadores tenham dupla função (encaminhar e receber pacotes.

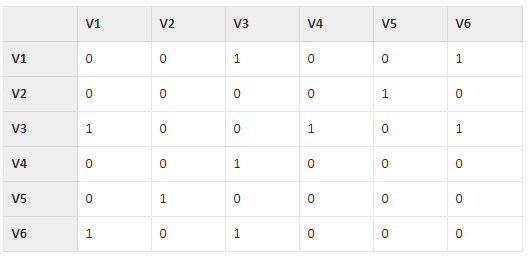
# 2. Fundamentação teórica

A principal estrutura de dados utilizada neste software foi o Grafo. “Um grafo é uma coleção de vértices(V) e uma coleção de arcos (E) constituídos por pares de vértices. É uma estrutura usada para representar um modelo em que existem relações entre os objetos de uma certa coleção” (ARAÚJO, 2007). O grafo é representado como mostrado na Figura 1.

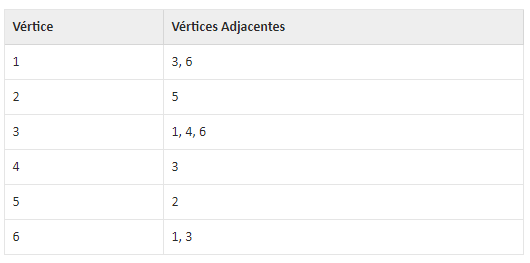


**Figura 1. Representação do grafo**

Para representar um grafo na programação, podemos usar uma Matriz de adjacência ou Listas de adjacência. A Matriz de adjacência “consiste em uma matriz N x N (onde N é um número de vértices) onde (i,j) indica se existe uma ligação do vértice i para o vértice j”(ARAÚJO, 2007), se houver ligação é colocado o número 1,caso contrário, é colocado o número 0, como mostrado na Tabela 1. As Listas de adjacência guardam as arestas incidentes de cada vértice, segundo ARAÚJO (2007), isto pode ser feito usando um vector de tamanho V (número de vértices), onde v[i] guardará a lista dos arcos ou vértices conectados ao vértice i, como mostrado na Tabela 2.



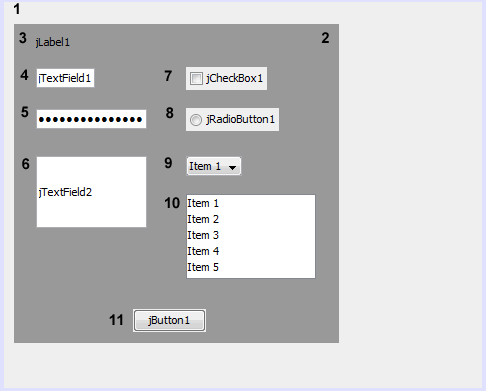
**Tabela 1. Matriz de adjacência**



**Tabela 2. Listas de adjacência**

Para fazer a interface gráfica foi utilizado o pacote Swing do Java para a criação das telas. A Interface Gráfica, também chamada de *Graphical User Interface* (GUI), “é um conceito da forma de interação entre o usuário do computador e um programa por meio de uma tela ou representação gráfica, visual, com desenhos, imagens, etc. Geralmente é entendido como a “tela” de um programa” (ELIAS, 2010).

O pacote *Swing* é importado da seguinte forma: javax.swing.\*. Segundo DEV MEDIA (2010), este pacote é uma evolução do pacote *Abstract Windows Toolkit* (AWT), todos seus componentes começam com a letra “J” para poder diferenciar dos componentes do pacote AWT já que o Swing possui quase todos componentes que existiam no pacote. “Os componentes do tipo swing possuem uma grande diferença dos Componentes AWT, pois os componentes Swing são implementados com nenhum código nativo, ou seja, apesar de serem sensivelmente mais lentos que os componentes nativos, eles permitem maior liberdade” DEV MEDIA (2006). O pacote Swing possui janelas, botões, campos de texto para entrada e saída de dados, painel de rolagem, seletor de dados e entre outros, como mostrado na Figura 2.



**Figura 2. Componentes do Java Swing**

# 3. Metodologia

Para a resolução deste problema foi feito um software em Linguagem Java pelo programa NetBeans, utilizando Grafo como a estrutura de dados principal, importação e exportação de arquivos, e interface gráfica para o sistema operacional Windows.

Já a parte de lógica e planejamento foi utilizada a metodologia *Problem Based Learning* (PBL), traduzido para o português como Aprendizagem Baseada Em Problemas. “A metodologia do PBL enfatiza o aprendizado autodirigido, centrado no estudante. Grupos de até 12 estudantes se reúnem com um docente (tutor ou facilitador) duas ou três vezes por semana. O professor não "ensina" da maneira tradicional, mas facilita a discussão dos alunos, conduzindo-a quando necessário e indicando os recursos didáticos úteis para cada situação.” (UNIFESP,2020)

Para a organização estrutural do software foi utilizado o padrão de projeto *Model-View-Controlle*r (MVC). Segundo MACORATTI(2020), esta arquitetura fornece uma maneira de dividir a funcionalidade envolvida na manutenção e apresentação dos dados de uma aplicação. O MVC possui três componentes, são elas:

* “O Model que representa os dados e não deve incluir detalhes de implementação podendo ter muitas Views associadas;
* A View que representa um componente de interface de usuário que está vinculado a um Model. Ela pode exibir os dados e permitir que a modificação dos dados pelo usuário [...].
* O Controller que fornece um mecanismo para o usuário interagir com o sistema definindo como a interface do usuário vai reagir a ação do usuário. Ele é responsável por trocar e interpretar mensagens entre a View e o Model.” (MACORATTI, 2014)

# 4. Resultados e discussão

Os requisitos pedidos pela empresa foram: adicionar e remover nós da rede de conexões, calcular distancia euclidiana entre os locais, mostrar caminhos menos custosos entre as conexões, mostrar a rede de conexões na tela e importar e exportar arquivos com as configurações dos dados. Como mostrado anteriormente, para resolução dos requisitos, o código do software desenvolvido está estruturado de acordo com o padrão de projeto MVC, sendo sete classes no total, onde é um para o Controller, quatro para o Model e dois para a View.

A classe Controller do pacote Controller, faz a chamada dos métodos da Model para que ela seja chamada na View. O Controller é formado por métodos estáticos que fazem a chamada dos métodos do Grafo onde o mesmo também é instanciado, já que é a estrutura de dados principal do software e a todo momento estará sendo manipulada através do Controller.

No pacote Model possui as classes Aresta, Vértice, Grafo e Manipular Arquivos. A classe Aresta possui como atributos os vértices de origem e destino, um peso representado por um número inteiro positivo e uma variável booleana chamada “visitado”, e como métodos, possui os métodos básicos como os *getters* e *setters*.

A classe Vértice possui como atributos o nome da vértice, uma distância representada por um número inteiro que irá auxiliar em alguns métodos, vértice pai, visitado e terminal como variáveis booleanas, uma lista do tipo vértice para indicar os vértices vizinhos e uma lista do tipo aresta com a lista de arestas incidentes, já que para a resolução deste problema estamos usando a lista de adjacência, e como métodos possui os básicos *getters* e *setters* , e métodos de adicionar incidentes e adicionar vizinhos.

A classe Manipular Arquivos tem a função de exportar e importar os arquivos de texto em formato .txt do tipo *Comma-separated values* (CSV). “O CSV é uma implementação particular de arquivos de texto separados por um delimitador, que usa a vírgula e a quebra de linha para separar os valores” (FUNÇÃO EXCEL, 2020). Nesta classe possui apenas os métodos estáticos de escrita e salvamento de um arquivo, que tem como principal parâmetro o caminho dos arquivos. Ao exportar um arquivo, todos os dados contidos no Grafo serão passados para o arquivo de texto, já ao importar, todos os dados lidos serão carregados no Grafo.

A classe Grafo tem como base o grafo MOTA(2014) para construção do código, ela possui como atributos uma lista do tipo aresta e uma lista do tipo vértice para armazenar as arestas e as vértices que o pertencem, e uma variável booleana para indicar se há ciclo. Ela possui os métodos básicos de *get* e *set*, métodos auxiliares e os métodos para realizar os requisitos, como o método de adicionar e de remover. No método de adicionar, o usuário insere os dados (ponto inicial, ponto final, peso da conexão, se é terminal ou não) através da interface para criar as conexões, e no método de remoção é pedido o nome da vértice que se quer apagar.

Para fazer a ligação entre o usuário e o sistema, o pacote View fica responsável por isso, ela possui uma classe principal chamada “Minha tela” que fará toda exibição ao usuário através dos *Listeners* e dos eventos que são acionados. Segundo DEV MEDIA (2014), “*Listeners*” em português significa “Ouvinte”, um objeto tem vários dependentes que são anexados a ele, quando este objeto muda de estado todos os seus dependentes são avisados. O método de calcular a distância euclidiana necessita da captura das coordenadas de dois pontos que o usuário clicar, para mostrar o caminho menos custoso entre dois pontos o usuário deve digitar o nome dos dois pontos que deseja ver o caminho, e para ver o menor caminho de um ponto para todos os outros basta digitar o nome de apenas um ponto.

Para calcular o menor caminho foi utilizado o Algoritmo de Dijkstra, “num dado vértice (nó) no grafo, o algoritmo localiza o caminho com a menor custo (isto é, o caminho mais curto) entre esse vértice e todos os outros vértices, recorrendo ao peso/custo da aresta” (PERES,2016).

Para apresentar na tela o desenho das conexões é necessária a classe Desenhar Grafo, ela vai desenhar o grafo em formato de círculo, para evitar que as vértices sejam colocadas no mesmo ponto, ela tem como atributos o grafo, o raio que terá o círculo, um valor base para a coordenada X e para a coordenada Y, um valor inteiro e para o tamanho que eu quero que seja a bolinha que representará a vértice e uma variável booleana chamada “menorCaminho”. Essa classe tem dois tipos de construtores, um para apresentar o grafo completo e o outro para traçar apenas o menor caminho de um ponto.

# 5. Conclusões

Alguns requisitos pedidos para a produção do software não foram cumpridos, sendo eles o requisito de calcular a distância euclidiana e o de calcular o menor caminho.

Os pontos fortes do software é a interação do usuário através da interface gráfica, já o ponto fraco, os métodos da interface poderiam ser mais otimizados, o Algoritmo de Dijkstra começou a dar problema depois de certo tempo, alguns métodos que eram pra ser acionados a partir do clique do *mouse* na tela e não foram feitos desta forma ,e faltou o tratamento de erros através das Exceções.

A partir do desenvolvimento do software passei a entender o funcionamento de uma interface gráfica, como é feita a interação do usuário, como podemos personalizar a interface através do Java *Swing* e aprendi como funciona o padrão de projeto MVC.

# Referências

Macoratti, J. C. (2014) “Entendendo o padrão MVC – Model – View – Controller”, http://www.macoratti.net/14/05/net\_mvc.htm, Março.

Macoratti, J. C. (2020) “Padrões de Projeto: O modelo MVC – Model – View – Controller”, http://www.macoratti.net/vbn\_mvc.htm, Março.

Unifesp (2020) “Aprendizado Baseado em Problemas”, http://www2.unifesp.br/centros/cedess/pbl/, Março.

Araújo, M (2007) “GRAFOS – 1ª PARTE”, https://www.revista-programar.info/artigos/grafos-1a-parte/, Março

Elias, M (2010) “O que é interface gráfica?”, http://www.explorando.com.br/o-que-e-interface-grafica/, Março

Dev Media (2006) “AWT x Swing – Código Nativo x Código Interpretado...”, https://www.devmedia.com.br/awt-x-swing-codigo-nativo-x-codigo-interpretado/3212, Março

Mota, G. (2014) “Algortimos.grafo”, https://gist.github.com/divanibarbosa/819e7cfcf1b9bae48c4e0f5bd74fb658, Março.

Dev Media (2010) “SWING - Componentes mais importantes e suas propriedades”, https://www.devmedia.com.br/swing-componentes-mais-importantes-e-suas-propriedades/16113, Março

Função Excel (2020) “Arquivo CSV”, https://www.funcaoexcel.com.br/arquivo-csv/, Março

Dev Media (2014) “Java Listeners: Trabalhando com ActionListener e KeyListener em Java”, https://www.devmedia.com.br/java-listeners-trabalhando-com-actionlistener-e-keylistener-em-java/31850, Março

Peres, R (2016) “Algoritmo De Dijkstra”, https://www.revista-programar.info/artigos/algoritmo-de-dijkstra/, Março