UNIVERSIDADE DO EXTREMO SUL CATARINENSE – UNESC
CURSO DE CIÊNCIA DA COMPUTAÇÃO

GUILHERME DE BONA ROMANCINI

LOGICINI: UM MODULO DE LOGÍSTICA DE ERP UTILIZANDO A TEORIA DE GRAFOS APLICADA A EMPRESA DE TRANSPORTE PARA O DESPACHO DE PRODUTOS CERÂMICOS

CRICIUMA, JULHO DE 2011

GUILHERME DE BONA ROMANCINI

LOGICINI: UM MODULO DE LOGÍSTICA DE ERP UTILIZANDO A TEORIA DE GRAFOS APLICADA A EMPRESA DE TRANSPORTE PARA O DESPACHO DE PRODUTOS CERÂMICOS

Trabalho de Conclusão apresentado para obtenção do Grau de Bacharel em Ciência da Computação da Universidade do Extremo Sul Catarinense.

Orientador: Prof. Msc.Paracelso Oliveira Caldas

CRICIUMA, JULHO DE 2011

GUILHERME DE BONA ROMANCINI

Logicini: Um Módulo de Logística de ERP Utilizando a Teoria de Grafos Aplicada a Empresa de Transporte para o Despacho de Produtos Cerâmicos

Submetido ao corpo docente do Curso de Ciência da Computação da Universidade do Extremo Sul Catarinense como um dos requisitos para obtenção do grau de Bacharel em Ciência da Computação.

Profa. MSc. Ana Claudia Garcia Barbosa
Coordenadora do Curso de Ciência da Computação

Banca Examinadora:

Prof. MSc. Paracelso Oliveira Caldas (UNESCO)
Orientador

Prof. MSc. Gustavo Bisognin (UNESC)

Prof. MSc. Kristian Madeira (UNESC)

AGRADECIMENTOS

A meus Pais e meu Irmão, que acreditarem em mim e no desenvolvimento profissional que adquiri durante o curso, sempre estando comigo em todos os momentos da vida.

A minha Namorada por ter a compressão e paciência que precisei para me dedicar a esse trabalho e por ter me apoiado em todas as fases do projeto.

Aos meus Amigos e Colegas que sempre estiveram comigo desde o inicio do curso, proporcionando bons momentos, espero contar sempre com vocês.

Ao meu orientador pelo aprendizado e responsabilidade passados durante todas as etapas de desenvolvimento do meu projeto.

Muito obrigado a todos vocês!

“A Ciência não estuda ferramentas. Ela estuda como nós as utilizamos, e o que descobrimos com elas.”

Edsger Dijkstra

RESUMO

Os sistemas de gestão empresarial são uma realidade de qualquer empresa de pequeno, médio e grande porte atualmente. Cada empreendimento tem uma particularidade, diferenciando-a, das demais existentes, e os analistas devem implantar o sistema para que funcione da maneira que o cliente trabalha, minimizando os problemas culturais da mesma.

Esse trabalho visa solução para realização do despacho dos produtos fabricados em sua sede para uma gama de clientes espalhados pelas várias regiões do país, de maneira que o veiculo que transportará a mercadoria percorra a menor distância possível entre as entregas. As entregas são feitas utilizando veículos de 14 até 44 toneladas de peso liquido definido pelo usuário que opera o sistema.

O sistema de despacho e entregas é feito utilizando os algoritmos de menor caminho dentro da Teoria de Grafos, nesse trabalho foram abordados os algoritmos de Dijkstra, Bellman-Ford, e Floyd-Warshall. Comparando cada um deles para além de obter um resultado eficaz, faze-lo de forma eficiente.

As informações que fazem parte do grafo são armazenadas em um banco de dados e após processamento pelo sistema esses dados constituem a montagem do grafo.

Como resultado principal o software gera a lista de cidades que o veiculo transportador irá seguir, junto com os pedidos para serem entregues aos clientes da empresa produtora.

Palavras-Chave: Sistemas ERP, Engenharia de Software, Teoria de Grafos, Algoritmos de menor caminho.

ABSTRACT

The business management systems are a reality of any business small, medium and large companies today. Each project has a special feature, distinguishing it, the other existing and analysts must deploy the system to work the way the client works to minimize the cultural problems of the same.

This work aims to achieve the solution to order the products manufactured at its headquarters for a range of customers spread across various regions of the country, so that the vehicle transporting the goods the shortest distance possible between deliveries. Deliveries are made using 14 vehicles to 44 tons of net weight defined by the user who operates the system.

The system dispatch and delivery is done using the shortest path algorithms in the graph theory, in this work were discussed Dijkstra algorithms, Bellman-Ford and Floyd-Warshall. Comparing each addition to achieve an effective, do it efficiently.

The information that is part of the graph are stored in a database and after processing by the system these data are mounting the graph.

As a main result the software generates a list of cities that the carrier vehicle will follow, together with applications to be delivered to customers of the producer.

Keywords: ERP systems, software engineering, graph theory, shortest path algorithms.

LISTA DE ABREVIATURAS E SIGLAS

ERP *Enterprise Resource Planning*

FK *Foreign Key*

GHZ *Gigahertz*

GPL *General Public License*

HD *Hard Disk*

IDE *Integrated Development Environment*

ISO *International Organization for Standartization*

MHZ *Megahertz*

MS Milissegundos

NBR Normas Brasileiras

NN *Not Null*

OMG *Object Management Group*

PCV Problema do caixeiro viajante

RPM Rotações Por Minuto

PK *Primary Key*

SC Santa Catarina

SGBD Sistema Gerenciador de Banco de Dados

SIGE Sistemas Integrados de Gestão Empresarial

UML *Unified Modeling Language*

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE QUADROS

SUMÁRIO

1 INTRODUÇÃO

Uma transportadora atua no ramo de transportes rodoviários de cargas, suas principais fontes de cargas provêm das indústrias cerâmicas da região sul de Santa Catarina. Quando um cliente solicita os serviços da transportadora, ela indica a quantidade de matérias primas que serão carregadas, e os destinos de cada mercadoria a ser entregue, a rota a ser seguida fica a critério da empresa de transportes.

O setor de logística é responsável pelo fluxo dos materiais desde o fornecedor até o cliente final. Seu objetivo é entregar os produtos adquiridos por clientes nas condições, prazos, quantidades, locais, clientes e custos solicitados (SEVERO, 2006). Visa-se percorrer a menor distancia possível, utilizar rodovias asfaltadas, aproveitar rotas com menor número de pedágios, minimizar gastos com oficinas. Deve-se organizar a matéria transportada, ou seja, deve se pensar na maneira de como será distribuída a carga antes do carregamento para posteriormente facilitar a descarga para os destinatários, seguindo sempre um plano de organização de cargas e rotas.

O sistema de rotas precisará incorporar um algoritmo baseado na teoria de grafos para analisar o caminho a ser percorrido por cada veículo visando a menor trajetória por meio de grafos valorados. Além da distância cada aresta precisa de um valor, que será o custo total da rodovia, ou seja, para ir do ponto x ao y passando pela aresta xy, esta terá um valor que seguirá uma fórmula pré-definida, onde deve constar o valor do frete, valor do pedágio, distância entre os pontos a ser seguidos, e a qualidade da estrada naquele trecho.

O objeto de estudo fará se aos problemas de roteamento em grafos, tal como árvores de custo mínimo, menor caminho e Problema do Caixeiro Viajante (PCV) bem como o desenvolvimento da aplicação para solução do mesmo. Árvore de custo mínimo é encontrar uma árvore que contenha todos os vértices do grafo e cuja soma das arestas seja mínima. O

problema do menor caminho tem como objetivo obter um percurso mínimo entre dois ou mais vértices de um grafo. O algoritmo de Dijkstra, tem como objetivo obter o menor caminho entre um dado vértice fixo e todos os demais vértices do grafo. No algoritmo de Floyd o percurso é feito entre todos os pares de ponto. O algoritmo de Ford encontra caminhos mínimos contendo pesos negativos, porém para este problema não poderá existir ciclos negativos, senão invalidará a noção de distância.

A montagem da carga é feita considerando o peso permitido para cada veículo, o peso entre eixos, a disponibilidade da entrega, primeiras cargas a serem descarregadas serão carregadas nos lugares com maiores facilidades para descarregamento. Uma carga mal distribuída e desorganizada aumenta em muito o tempo para retirada do material transportado, ocasionando atraso na entrega de diversas cargas, aumento do preço do frete em relação ao tempo parado, e transtorno para quem deseja receber a carga.

Necessariamente a montagem da carga e o sistema de roteamento são partes de um mesmo sistema interligado por suas necessidades, ou seja para desenvolvimento da rota é imprescindível a implementação do algoritmo de montagem da carga, que é a base para poder fazer o roteamento com a descarga de cada mercadoria que será transportada.

Faz-se necessário um módulo que se integre com o ERP da empresa para gerenciamento eficaz, para isso será imprescindível à criação do projeto de acordo com as normas da engenharia de software. Segundo Pressman (2006) a natureza de aplicação deve ditar a abordagem a ser tomada. Ao combinarmos abordagens, o todo pode ser maior do que a soma das partes.

1.1 OBJETIVO GERAL

Desenvolver um protótipo de módulo para gerenciamento das rotas e montagem das cargas utilizando a teoria de grafos aplicados à logística de transportes de cargas terrestres para sistemas ERP.

1.2 OBJETIVOS ESPECÍFICOS

Os objetivos específicos deste projeto são:

a) descrever os aspectos para criação do projeto de software;
b) comparar algoritmos da teoria de grafos na tarefa do melhor caminho para utilização no sistema;
c) programar o sistema utilizando um banco de dados;
d) criação de uma interface gráfica;
e) demonstrar o funcionamento do software;
f) desenvolvimento do módulo de logística para sistemas ERP para gerenciamento das rotas e montagem das cargas utilizando teoria dos grafos.

1.3 JUSTIFICATIVA

Os Sistemas Integrados de Gestão Empresarial (SIGE) ou ERP tradicionalmente conhecido, são sistemas de informação que integram todos os dados e processos de uma organização em um único sistema, fornecem rastreamento e visibilidade global da informação de qualquer parte da empresa e de sua cadeia de suprimento, o que possibilita decisões inteligentes (CHOPRA; MEINDL, 2003).

Para desenvolvimento do projeto de software será considerado primordialmente essencial às fases de análise de requisitos, projeto, implementação, testes e manutenção. Análise de requisitos, segundo Braude (2004) é o processo de entender e colocar no papel, uma declaração do que uma aplicação destina se a fazer depois de construída. Projeto define como a aplicação deverá ser construída, descrevendo as partes envolvidas e como elas devem ser montadas, consiste em um conjunto de diagramas, podendo usar uma linguagem de notação como a Linguagem Unificada de Modelagem (UML). Para Guedes (2008) UML é uma linguagem visual utilizada para modelar sistemas computacionais por meio de orientação de objetos, sendo que essa linguagem tem como objetivo auxiliar engenheiros de softwares a definirem as características de seus projetos. Implementação é a parte que todo o projeto será codificado para uma linguagem de programação e após tem se a parte que fará o testes desse software já codificado, entrando em um ciclo que fica em programação e testes.

O módulo desse sistema será construído na linguagem de programação, JAVA, pois é uma linguagem promissora que constantemente vem crescendo no mercado, tem se tornado muito popular nos últimos anos e é flexível, pode se programar em qualquer sistema operacional que contenha a maquina virtual.

Utilizar-se-á um banco de dados para armazenar toda a informação cadastrada no sistema que será usado também para fazer estatísticas e relatórios.

Rotas farão uso de algoritmos da teoria de grafos que é uma teoria matemática das relações entre elementos de conjuntos discretos, para definir o uso do algoritmo adequado fará se uma análise de algoritmos de caminho mínimo para implantação no módulo.

Serão comparados os algoritmos de Djkstra, Floyd e Ford. Para a análise de qual será implantado, o comparativo dará se em relação do tempo para descoberta do menor caminho, menor rota encontrado, custo de memória e processamento, dentre outros requisitos que serão

desenvolvidos durante os testes. Todos os algoritmos descritos acima farão parte desse software, para permitir a análise prática de cada um deles.

Djkstra, Floyd e Ford, são algoritmos de menor caminho que procuram em um grafo, o menor caminho a ser percorrido entre dois ou mais pontos. Djkstra trabalha apenas com grafos valorados com valores positivos, usa duas listas uma chamada F(fechado) e A(aberto), F é a lista das vértices que já conhece um caminho mínimo e A que não conhecem. Ford pode trabalhar com ciclos negativo ou não, a cada iteração ele calcula a distancia de um ponto inicial a todos os demais pontos. No algoritmo de Floyd são feitas inúmeras iterações que corresponde ao número de nós no grafo, em cada iteração corresponde uma matriz cujos valores são modificados.

O sistema terá como base o Problema do Caixeiro Viajante (PCV), caixeiro viajante é uma profissão em que deve se levar os produtos da cidade que foram produzidos para fora percorrendo o menor caminho possível. Conforme Netto (1996) o PCV tem grande importância teórica, dada sua relação com outros problemas de otimização combinatória, a pesquisa de algoritmos para a busca de soluções de boa qualidade tem sido um linha de pesquisa de grande difusão e sucesso.

O propósito desse software é a criação de um sistema que determine as rotas que deverão ser percorridas, de acordo com os produtos que precisam ser entregues ao destinatário, observado a distância entre as localidades. Ao inicio da viagem, as rotas serão devidamente planejadas e dará se então a posição de volume a de revestimentos cerâmicos ordenados conforme o planejamento das entregas.

1.4 ESTRUTURA DO TRABALHO

O Trabalho está dividido em seis capítulos, o primeiro descreve a introdução do problema, objetivos e justificativas para elaboração do trabalho e solução do problema. O segundo capítulo fala dos sistemas ERPs, ciclos, fases e técnicas para construção de um software.

O terceiro capítulo desse projeto consiste em abordar a engenharia de software, seus métodos de desenvolvimento, ferramentas e técnicas. O quarto capítulo descreve a teoria de grafos e seus algoritmos para aplicação no problema inicial.

O quinto capítulo, mostra os trabalhos correlatos desenvolvidos na área, e o sexto capítulo apresenta a modelagem do protótipo, o banco de dados, o desenvolvimento dos algoritmos, e os resultados obtidos.

2 SISTEMAS ERP

No início da década de 90, os ERPs passaram a ser amplamente utilizados pelas empresas. Eram extremamente caros, e adequados somente para empresas grandes. No decorrer dessa década, as grandes empresas fizeram suas escolhas sobre os sistemas a serem adquiridos e implantados, carregando assim o mercado das grandes empresas e reduzindo as possibilidades de negócio para os fornecedores de ERPs nesse segmento empresarial (Corrêa, 1998). Segundo Chopra e Meindl (2003), com a evolução dos ERPs, houve também uma mudança na tecnologia adotada pelas empresas, que passaram de plataformas tipo mainframe para cliente e servidor.

Sistemas integrados de Gestão Empresarial *(Enterprise Resouce Planning* - ERP*)* é um tipo de sistemas que tem como principal característica integrar os processos internos da empresa. Esses sistemas operam desde a produção até o financeiro, interligando todas as partes em um único sistema subdivido em módulos que utilizam uma grande base de dados.

Segundo Martins (2007), ERP é um conjunto de softwares que permitem a integração de dados de sistemas de informações transacionais e os processos de negócios da empresa. Para Rezende (2005) os ERPs devem possuir uma base de dados única que manipulam e gerar informações operacionais e gerencias para todas as organizações.

Pode se concluir que um sistema ERP é um conjunto de sistemas que compartilham a mesma base de dados, e são interligados para troca de informações entre as diversas áreas da empresa.

O objetivo do ERP é integrar todos os departamentos e funções da organização em um sistema unificado no intuito de obter as informações de forma eficaz e oportuna (SOUZA; ZWICKER, 2000).

Pode se definir que um sistema ERP é um conjunto de sistemas que compartilham a mesma base de dados, e interligam-se para troca de informações entre os setores, criando uma solução empresarial.

Os ERP são uns dos principais sistemas de informação, e seu valor se dá ao fato de ser uma solução de gestão empresarial muito reconhecida na sociedade empresarial. No entanto, sua adoção requer grandes investimentos, e sua implantação requer consultorias especializadas e em consequência maiores gastos para a empresa.

Para Schroeder (2002) aquisição de uma solução integrada deve-se observar algumas características: melhorar o acompanhamento e o controle dos processos da empresa, filias ou depósitos; reduzir os custos operacionais internos; aumentar a previsibilidade operacional; aumentar a visibilidade e transparência do fluxo de informações; aumentar a velocidade da empresa como um todo; aperfeiçoar os processos de trabalho visando aumentar a qualidade; tomar decisões mais rápidas levando em consideração informações online ou estatísticas; estreitar e melhorar o relacionamento com seus clientes e trabalhar com diretrizes e metas empresariais integradas as pessoas e ao processo de trabalho.

Uma empresa que contrata esses sistemas espera conseguir ter total controle de seu negócio, melhorar a produtividade, aumentar os lucros. Para Corrêa (1998) um ERP deve disponibilizar a informação certa e boa na hora certa, nos pontos de tomada de decisão ao longo de todo empreendimento principalmente em termos do fluxo logístico.

2.1 CICLO DE VIDA DE ERP

O ciclo de vida do ERP, demostra as etapas desde a criação do sistema até sua utilização, estando sempre em constante desenvolvimento e aprimoramento.

Souza e Zwicker (2000) demonstram um modelo para o ciclo de vida de sistemas ERP conforme figura 1, seguindo as etapas:

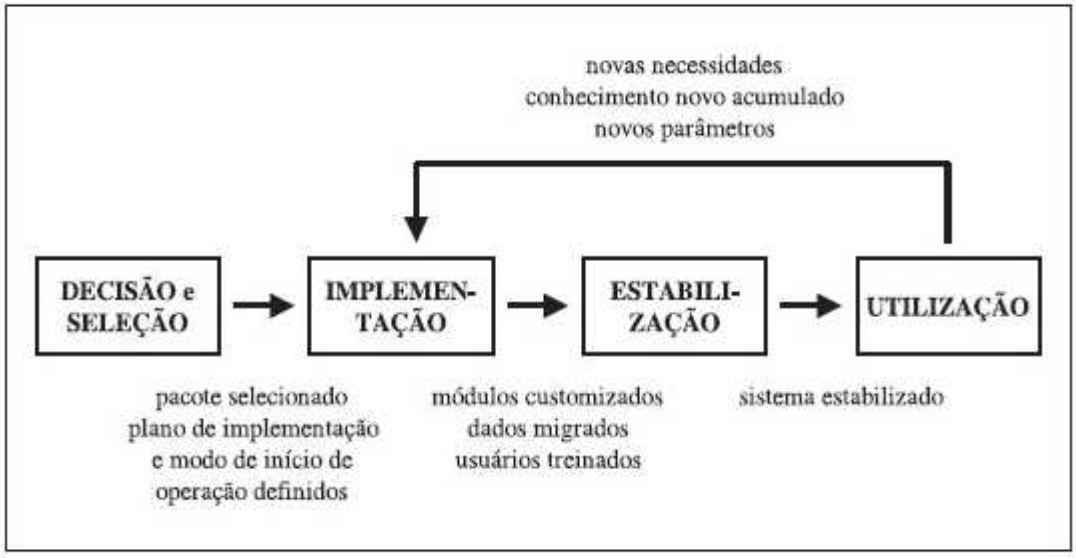


Figura 1 – Ciclo de vida do ERP
Fonte: SOUZA, C.; ZWICKER, R. (2000, p.35)

a) decisão e seleção descreve a solução para o sistema ERP a ser implantado na empresa de acordo com as necessidades do cliente, envolve o planejamento, regras e prazos a serem cumpridos, e as métricas do projeto;
b) implementação, define-se o processo que irá deixar o sistema em funcionamento, fazem parte dessa etapa, as configurações de hardware e software, customização do sistema, e treinamento e suporte aos usuários;
c) estabilização, nessa etapa o ERP começa a trabalhar dentro da empresa, aqui ocorre as maiores dificuldades do usuário, possíveis problemas de treinamento, erros no software, para serem corrigidos na próxima etapa;
d) utilização, chegando nessa etapa o ERP já está funcional para a empresa, não representa que todos os erros e problemas foram corrigidos, por isso que desta

etapa volta se para a implementação, pois ao longo do tempo, podem ocorrer mudança e aperfeiçoamentos no software.

Para o desenvolvimento e implantação de um sistema ERP, segundo Chambers (2007, p.51) o sistema deve seguir algumas regras:

a) Administração do ERP é avaliar o campo de missão e visão da empresa para atingir suas metas;
b) Geração de informação do ERP é onde será alimentado o sistema com dados;
c) Controle e avaliação, as informações são analisadas ajudando o usuário na tomada de decisão;
d) Disseminação dos dados e informações corresponde a parte onde distribui a informação para o responsável da área correspondente;
e) Utilização das informações da empresa, toda informação é composta a fim de incorporar dados para ajudar o usuário no processo decisório;
f) Retroalimentação ou realimentação ou *feedback* das informações, estrutura o processo decisório, afim de melhorar as necessidades de informações da empresa.

2.2 TECNICAS DE IMPLEMENTAÇÃO ERP

Seguindo a estrutura do ciclo de vida do ERP, a etapa de implementação apresenta técnicas que serão usadas para modelar o ambiente usual às necessidades da empresa. Na etapa de desenvolvimento do sistema podem ocorrer problemas imprevisíveis ao modelo que foi projetado, podendo ocorrer mudanças na estrutura inicial do software.

Wood e Caldas (1999), ressaltam que é a etapa de implementação é uma das mais críticas para o desenvolvimento do ERP. Segundo Laudon e Laudon (1996) a implementação é definida como as atividades organizacionais realizadas em direção à adoção, gerenciamento e roteirização de uma inovação.

2.2.1 Técnicas para Gestão de Projeto

A gestão de projeto é definida como o planejamento, a programação e o controle de uma série de tarefas de forma a atingir seus objetivos com êxito, para beneficio dos

participantes do projeto (KERZNER, 2004). As técnicas para gestão de projetos são metodologias baseadas no conceito da NBR ISO 10006:2000, que descreve as diretrizes para qualidade no gerenciamento do projeto.

2.2.2 Técnicas para Análise de Processo de Negócio

O processo de negócio é onde o projeto é tratado como um produto a ser negociado, não visa esclarecer assuntos técnicos como banco de dados, linguagem de programação, hardware, rede, etc. Procura- se adequar novas práticas de negocio agregando e incorporando novos pontos de controle ao software. Segundo SOUZA e ZWICKER (2000), para adequar o processo de negocio ao ERP necessita ajustar a empresa ao sistema, o sistema a empresa, ou a combinação parcial de ambas.

2.2.3 Técnicas para Gestão de Mudança

A implantação do ERP apresenta a empresa uma nova forma de organização de acordo com a estrutura do sistema. Segundo Gambôa e Filho (2003), dentro da gestão de mudança planejada, implementa se o plano de capacitação para usuários finais do ERP, e por fim definir a estratégia de implementação e de conversão dos dados a serem adotadas.

2.2.4 Técnicas para Gestão de Qualidade

Dentro todos os requisitos do projeto de software, para ter total sucesso, não é apenas necessário que satisfaçam todos os itens funcionais para a empresa, mas que o projeto esteja de acordo com seu cronograma, obedecendo todos os prazos estipulados. Seguindo os custos planejados pela empresa, desde o projeto até a implantação e manutenção do sistema. Ostrenga et al. (1993), provê uma base que a gestão de qualidade reduz o tempo do ciclo e também os custos.

2.2.5 Técnicas para a Gestão de Riscos

Segundo Gusmão (2007), a gestão de riscos tem como objetivo a abordagem de riscos a ser utilizada e planejar as atividades de gerência que serão executadas para o projeto. Sendo assim é de fundamental importância que haja um acompanhamento dos riscos associados ao sistema para identificar e minimizar os problemas durante o projeto.

3 ENGENHARIA DE SOFTWARE

A engenharia de software abrange a gestão do projeto e as técnicas de desenvolvimento de um software, é de fundamental importância para um projeto de sucesso que siga as corretamente as etapas e os conceitos definidos por ela.

Antes de conceituar se engenharia de software deve conhecer outros quatro conceitos, como apresenta Rezende (2005):

a. engenharia é a arte de construir com base no conhecimento cientifico e empírico;
b. engenhar é inventar produzir algo, idealizar;
c. sistema é o conjunto de partes que se integram afim de realizar um objetivo;
d. software é o subsistema de um sistema computacional, ou seja os programas de computador.

Para Sommerville (1992), a engenharia de software envolve as questões técnicas e não técnicas, pode se citar a aquisição do conhecimento, as técnicas para implementar o projeto e a gestão do projeto.

> Engenharia de software é a metodologia de desenvolvimento e manutenção de sistemas modulares, com as seguintes características: processo (roteiro) dinâmico, integrado e inteligente de soluções tecnológicas; adequação aos requisitos funcionais do negocio do cliente e seus respectivos procedimentos pertinentes; efetivação de padrões de qualidade, produtividade e afetividade em suas atividades e produtos; fundamentação na tecnologia da informação disponível, viável, oportuna e personalizada; planejamento e gestão de atividades, recursos, custos e datas (REZENDE, 2005, p. 2).

Segundo Pressman (2006) a engenharia de software é a criação e a utilização de sólidos princípios de engenharia a fim de obter softwares que sejam confiáveis e que trabalhem eficientemente em máquinas reais.

De modo geral, a engenharia de software prove a qualidade do software, atendendo aos requisitos do software com eficiência.

3.1 ENGENHARIA DE REQUISITOS

A engenharia de requisitos é o processo que reúne todas as atividades fundamentais para a produção de um projeto de software afim de atingir os requisitos e dar manutenção do software ao longo do tempo.

Segundo Rezende (2005) os requisitos do software devem especificar efetivamente o Sistema de Informações, suas funções, desempenho, interface e restrições, conforme as fases e subfases da metodologia de desenvolvimento de sistemas.

Para Shaw (2001) os requisitos descrevem um acordo ou contrato entre duas partes: cliente e contratante, especificando o que o software deverá fazer para ser aprovando em um teste de aceitação.

Pode se observar que a engenharia de requisitos é a primeira parte de um ciclo de vida do projeto de software, nele são descritos o comportamento do software e todas as funções que ele será responsável, e sendo aceito pelo cliente, este é o conjunto de documentos que servirá para firmar contrato entre cliente e desenvolvedor.

O processo de engenharia de requisitos pode ser realizado por sete funções distintas: concepção, levantamento, elaboração, negociação, especificação, validação e gestão.

A concepção segundo Pressman (2006) é como o projeto de software será realizado, na maioria dos casos o engenheiro de software nessa etapa, faz uma série de perguntas livre de contexto, para entender o problema do cliente.

A fase do levantamento é onde acontece a interação do desenvolvedor com os usuários do sistema, definindo seus objetivos e necessidades.

Na elaboração as informações as informações obtidas pelos clientes nas etapas anteriores são refinadas a fim de produzir um modelo de análise que define o domínio do problema.

A negociação implica no confronto dos requisitos obtidos pelo cliente e o modelo de analise proposto pelo desenvolvedor conciliando-os para inicio, são definidos custos e prazos de entrega satisfazendo cada uma das partes.

Especificação é o produto final do engenheiro de requisitos, é o documento que irá descrever as funções do software, desempenho e restrições.

A validação prevê se na etapa de especificação todos os requisitos foram declarados de forma não ambígua, e sem erros, para garantir a qualidade do software.

E por fim tem se a gestão dos requisitos para controlar e encontrar possíveis erros, modificações de requisitos em qualquer tempo de desenvolvimento do projeto.

3.2 MODELOS DE PROCESSO DE SOFTWARE

O modelo de processo de software descreve como se desenvolverá as etapas do ciclo de vida do projeto, do inicio até sua manutenção. E de fundamental importância ter isso bem declarado e posto em pratica, pois implica na organização dos processos e de como ocorrerá à iteratividade entre as etapas de elaboração do software. Existem vários modelos que projetam fases para desenvolver um sistema seguindo suas politicas e normas de uso.

Um modelo de processo de software é uma descrição abstrata do processo de software. As informações são integradas no modelo de processo para indicar, quem, quando, onde e porque as etapas serão realizadas (LONCHAMP, 1993).

Segundo Pressman (2006) o melhor modelo de processo é aquele que se aproxima do pessoal que fará o serviço, e a existência de um modelo de processo não é a garantia que o software será entregue no seu prazo.

3.2.1 Modelo em Cascata

O modelo em cascata é conhecido também como ciclo de vida clássico, segue um processo sistemático e sequencial. Segundo Reis (2003), o modelo cascata apresenta claramente uma distinção entre as atividades de desenvolvimento do projeto e a passagem para o estágio de manutenção.

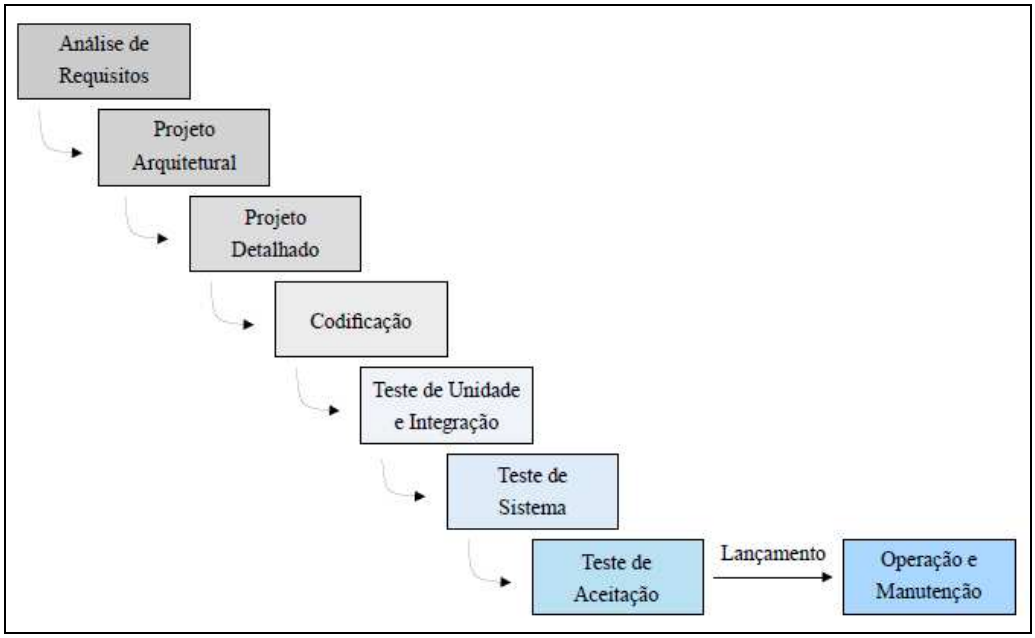


Figura 2. Ciclo de vida do modelo cascata,
Fonte: Reis, I. (2003, p. 26)

O modelo cascata é simples e de fácil entendimento, tanto para desenvolver como cliente. O ciclo de vida começa com a analise de requisitos imposto pelo cliente, o desenvolver recolhendo os requisitos e cria um modelo de processo que servirá de base para apresentação ao cliente.

Com os requisitos do software em mãos, cria-se então o projeto detalhado que envolvem todas as características do sistema, após vem a fase de codificação, tornando o projeto em produto, com o sistema implementado vem as fase de testes e testes de aceitação, e ao final a operação e a manutenção sempre andando paralelamente durante todo o ciclo de vida do software.

3.2.2 Prototipagem

Na Prototipagem o cliente define um conjunto de objetivos gerais para o software, mas não identifica detalhadamente requisitos de entrada, processamento ou saída, são inicialmente mostrados telas do sistema, e suas relações só serão construída a medida que o protótipo é apresentado e avaliado pelo cliente.

Segundo Rezende (2005) a prototipagem é o processo de construir um sistema experimental rapidamente e de baixo custo para demonstração e avaliação, de modo que os futuros clientes e usuários do sistema possam melhor determinar seus requisitos.

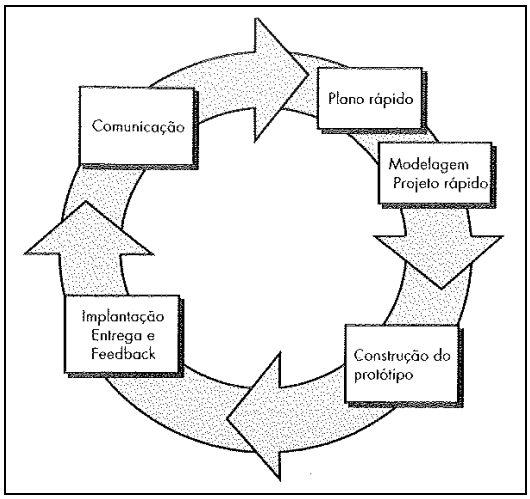


Figura 3. Ciclo de vida da prototipagem
Fonte: Pressman, R. (2006, p.43)

Esse modelo de sistema tem como propósito iniciar o ciclo de vida do processo com a comunicação entre o desenvolvedor e o cliente para definir os objetivos do software, a partir dessa etapa, cria – se um projeto rápido, que nada mais é do que o *layout* e algumas telas de entrada e saída de dados, este é construído e implementado onde novamente é apresentado ao cliente, estando sempre em um ciclo (Pressam, 2006).

3.2.3 Modelo Espiral

Esse processo permite uma série de iterações entre as fases de desenvolvimento. Cada iteração implica em uma volta no ciclo espiral. Nesse modelo o desenvolvedor pode obter produtos ou resultados em prazos mais curtos (REZENDE, 2005).

Segundo Horstimann(2003) as fases iniciais do modelo espiral centram em criação de protótipos para interagir com o cliente familiarizando-o com o sistema. Pode criar protótipos para validar interfaces com sistemas externos, testar o desempenho etc.

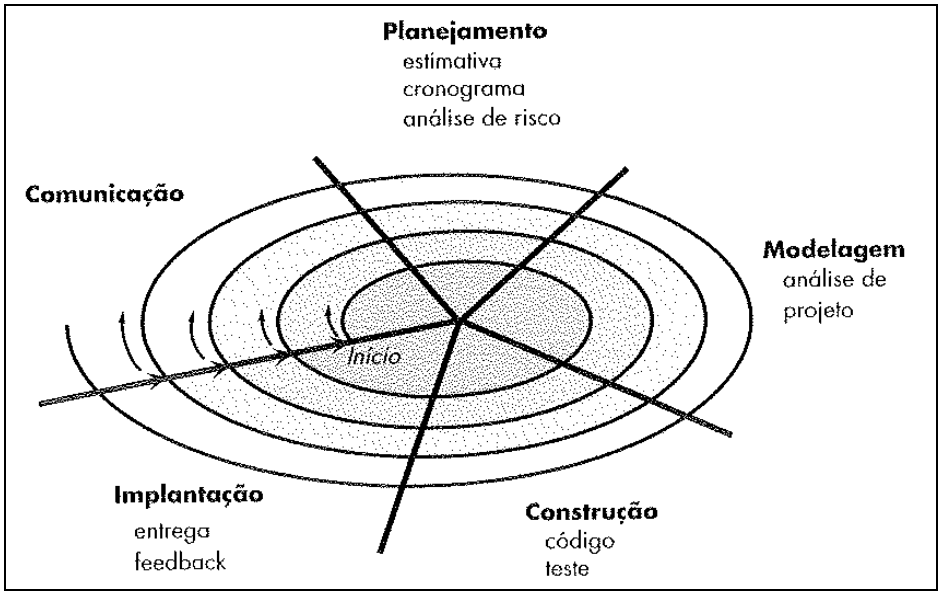


Figura 4. Ciclo de vida do modelo espiral
Fonte: Pressan, R. (2006, p.45)

O modelo espiral é divido em conjunto de atividades. Cada uma das atividades representa um segmento do caminho espiral, a medida que esse processo começa a equipe realiza atividades que são indicadas por um circuito (PRESSMAN, 2006).

O modelo espiral é geralmente usado para sistemas de grande porte, pois o sistema avança a medida que se completam os ciclos, e tanto o desenvolvedor como cliente passam a entender mais o sistema.

3.3 Arquitetura de Software

A arquitetura de software determina a organização em partes de um sistema. Cada parte pode conter um serviço próprio ou de terceiro que será integrada com no projeto afim de obter seu objetivo. Segundo Martins (2007) a engenharia de software ao estabelecer um tipo de arquitetura para o projeto, deve se analisar os elementos que compõe o software, que serão armazenados em pacote, e cada pacote pode conter um ou mais elementos.

Para Braude (2004) a arquitetura para o software é o projeto em alto nível para que possa permitir que os engenheiros de software ganhem clareza no seu projeto facilitando o entendimento, deve decompor o projeto em pequenos números de classe.

Existem vários formas, ou tipos de arquiteturas de sistemas, Fowler (2004) relata que não existe apenas um modo de especificar a arquitetura de software,

Os tipos de arquiteturas são classificados em camadas, colunas, objetos ou monolíticos, conforme a Figura 5:

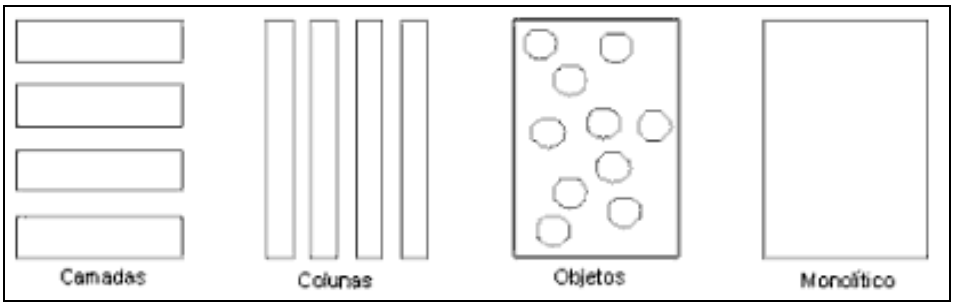


Figura 5. Tipos de arquiteturas de software
Fonte: Martins J. (2007, p.18)

Na arquitetura, o software é dividido em camadas cada uma com uma função específica, essa divisão pode ser de um a infinito. Para Martins (2007), o mais comum é dividir em cinco camadas: interface usuário, controle, domínio, persistência, interface externa.

a) interface usuário: faz o controle e interação como usuário do sistema, é responsável pela aparência e navegação;
b) controle: o controle é responsável por fazer as transações de negócio e integridade das informações;
c) domínio: realiza a chamada de dados e contem as funcionalidades do sistema;
d) persistência: armazena e recupera os dados do domínio;
e) interface externa: faz a integração com sistemas de terceiros.

A quantidade de camadas é determinada pelo arquiteto de software, e nem sempre incorpora todas essas camadas, porém a camada de domínio sempre precisa estar em qualquer projeto, pois é nela que são implementadas todas as funcionalidades do sistema.

Na arquitetura em colunas divide-se o sistema em módulos, sendo que dentro de cada módulo é implementado todas as funções do software, e cada modulo pode se subdividir em camadas do tipo: controle, domínio, persistência etc.

A arquitetura em objetos, segundo Martins (2007), cada conjunto de objeto implementa apenas uma parte das funções do software. Por fim a arquitetura monolítica, é organizada em uma única parte que contempla todos os itens do sistema.

Para Fowler (2003) a parte mais difícil encontrada na arquitetura de software, é escolher quais partes do sistema serão necessárias construir e o que cada uma delas devem implementar em suas subdivisões.

3.4 FERRAMENTAS

No decorrer da fase de projeto e implementação do software, é necessário que haja ferramentas e técnicas que ajudem o engenheiro de software a documentar e esclarecer de forma visível todas as etapas de construção e implementação do sistema. As ferramentas e técnicas para a engenharia de software tem grande relevância no desenvolvimento de sistemas e são fundamentais para gerenciar qualquer tipo de projeto.

3.4.1 Linguagem de Modelagem Unificada

Segundo Fowler (2004) UML é um conjunto de notações gráficas para auxilio no projeto de sistemas, é um padrão aberto controlado pela OMG, foi desenvolvida em 1997, e até hoje é usada para auxiliar o desenvolvimento de projetos de software.

> UML serve para especificar, construir, visualizar e documentar os artefatos de um sistema de software. (...) A ênfase da UML é na definição de uma linguagem de modelagem padrão (standart), e, por conseguinte, independente de linguagem de programação, de ferramentas case, bem como dos processos de desenvolvimento. O objetivo da UML é, dependendo do tipo de projeto, das ferramentas de suporte, ou da organização envolvida, poder adotar diferentes processos/metodologias, mantendo-se, contudo, a utilização de uma única linguagem de modelagem (RAMOS, 2006, p.8).

A UML define uma notação e um metamodelo, notação é o material gráfico usado para a modelagem e matamodelo é um padrão ou exemplo de um modelo para ser seguido, já que a notação pode ser implementada da forma que bem intender, o metamodelo propõe certa padronização para criação do material gráfico.

3.4.1.1 Diagramas

Segundo Fowler (2004) a UML descreve treze tipos de diagramas oficiais e podem ser usados da maneira que se necessite:

a) diagrama de atividades: útil para visualizar fluxos de trabalho e processos de negócio;
b) diagrama de classes: visualizar características do objeto e relacionamentos;
c) diagramas de comunicação: descreve como será realizada a integração entre os objetos;
d) diagrama de componentes: como realizar a estrutura e a maneira que será construída a conexão entre os componentes;
e) diagrama de estruturas compostas: mostra como ocorrerá a decomposição de uma classe;
f) diagrama de distribuição: como será distribuído os nós no projeto;
g) diagrama de visão geral da interação: tem por objetivo conciliar os diagramas se sequencia e atividade afim de uma melhor interação dos dados;
h) diagrama de objeto: configuração e funções variáveis de cada objeto;
i) diagrama de pacotes: mostra a estrutura na sua forma hierárquica;
j) diagrama sequencial: permite visualizar os passos sequenciais de um objeto;
k) diagrama de maquina de estados: define por quais eventos os objetos podem passar no decorrer do ciclo de vida;
l) diagrama de sincronismo: mostra a interação dos objetos, porém com ênfase no sincronismo;
m) diagrama de casos de uso: como os usuários farão a interação com o sistema.

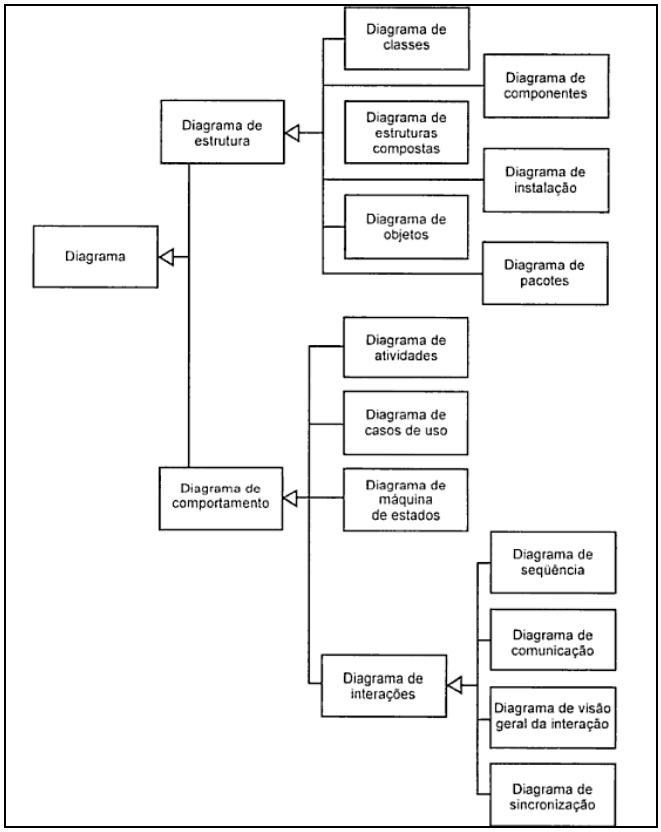


Figura 6. Tipos de diagrama
Fonte: FOWLER, M. (2004 p.34)

4 TEORIA DE GRAFOS

Segundo Coelho (2007) a teoria de grafos representa matematicamente um objeto na forma computacional, o grafo é definido por um conjunto de vértices e outro de arestas. O grafo pode conter algumas propriedades, dentre elas:

a) podem ser direcionados ou não;

b) o número de arestas implica no tamanho do grafo;

c) vizinhos são grafos unidos por uma aresta;

d) caminho é o percurso de um nó do grafo até outro;

e) ciclo é o caminho que começa e termina no mesmo nó, e acíclico é um grafo sem ciclos.

Grafos dirigidos podem ser usados para melhorar matematicamente as relações de um conjunto de grafos. Para Anton e Rorres (2000) um grafo dirigido é um conjunto de elementos finitos com uma relação também finita de pares ordenados de elementos distintos.

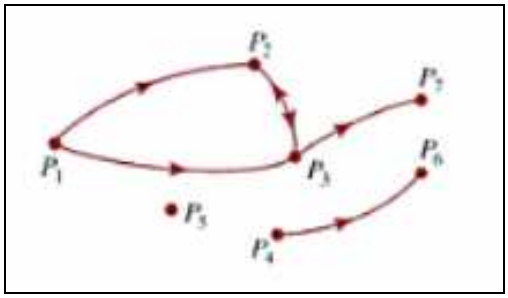


Figura 7. Grafo Dirigido
Fonte: Anton e Rorres (2000, p.397)

A Figura 7 demonstra que no mesmo grafo pode ocorrer de haver componentes separados de vértices ou sem estar conectados a nenhum outro nó.

4.1 PROBLEMA DO CAIXEIRO VIAJANTE

Segundo Brookshear (2003) o problema do caixeiro viajante é visitar seus clientes em diferentes cidades, e com o mínimo de gastos na viajem, a dificuldade do caixeiro é montar a trajetória que irá percorrer sem exceder a quilometragem determinada.

Johnson (2001) descreve que é quase impossível resolver o PCV, por existirem inúmeras rotas mesmo que com poucas cidades, a melhor estratégia por ele descrita seria a criação de roteiros que sejam relativamente pequenos e não necessariamente os mais curtos.

Mesmo não existindo mais caixeiros viajantes no século XXI, o problema continua, em meios como transportes rodoviários, correios, rede de internet, que ainda devem se preocupar em calcular a rota mais curta entre os nós de uma rede.

4.2 ALGORITMOS

Algoritmos são sequencias de etapas descritas de forma lógica que podem ou não levar a solução de um problema computacional.

Na análise de algoritmos podemos compará-los afim de avaliar seu desempenho, segundo GoodRich e Tamassia (2002), para determinar o tempo de execução de um algoritmo é necessário executar entradas com uma série de dados e registrar o desempenho para cada análise, o tipo de hardware e software também influencia do tempo de execução.

Em grafos os algoritmos podem ser implementados para vários domínios diferente, como explica GoodRick e Tamassia (2006), são usados em sistemas de informação geográfica, sistemas de transportes aéreos, rodoviários e marítimos, circuitos elétricos, conexões de internet, etc.

4.2.1 Algoritmo de Dijkstra

O algoritmos de Djstra é algoritmo com método guloso, para solução do problema do menor caminho. Goodrich e Tamassia (2006) pressupõe que para simplificar esse algoritmo fica necessário que o grafo de entrada não seja dirigido e simples.

> O pseudocódigo do algoritmo de Dijkstra é fornecido no código abaixo.
> Entrada: um grafo simples, ponderado e não dirigido G, e um vértice v de G.
> Saída: um rótulo D[u] para cada vértice u de G tal que D[u] seja o comprimento do caminho mínimo de v para u em G.
> Inicialize D[v] ← 0 e D[u] ← + ∞ para todo u ≠ v.
> Crie um fila de prioridade Q contendo todos os vértices de G usando rótulos como chave enquanto Q não for vazia faça:
> u ← Q remove o mínimo
> para cada vértice z adjacente a u tal que z esteja em Q faça:
> se D[u] + w(u,z) < D[z] então
> D[z] ← D[u] + w(u,z)
> Altere para D[z] a chave vértice de z em Q
> Retorna o rótulo D[u] de cada vértice em u (Goodrich e Tamassia, 2006, p.543).

O algoritmo percorre todos os nós do grafos, começando pelo nó de entrada, ou seja, em um mapa, se estamos em Criciúma e precisa-se ir até Florianópolis, passando por Araranguá, Imbituba e Palhoça, como representa a Figura 8.

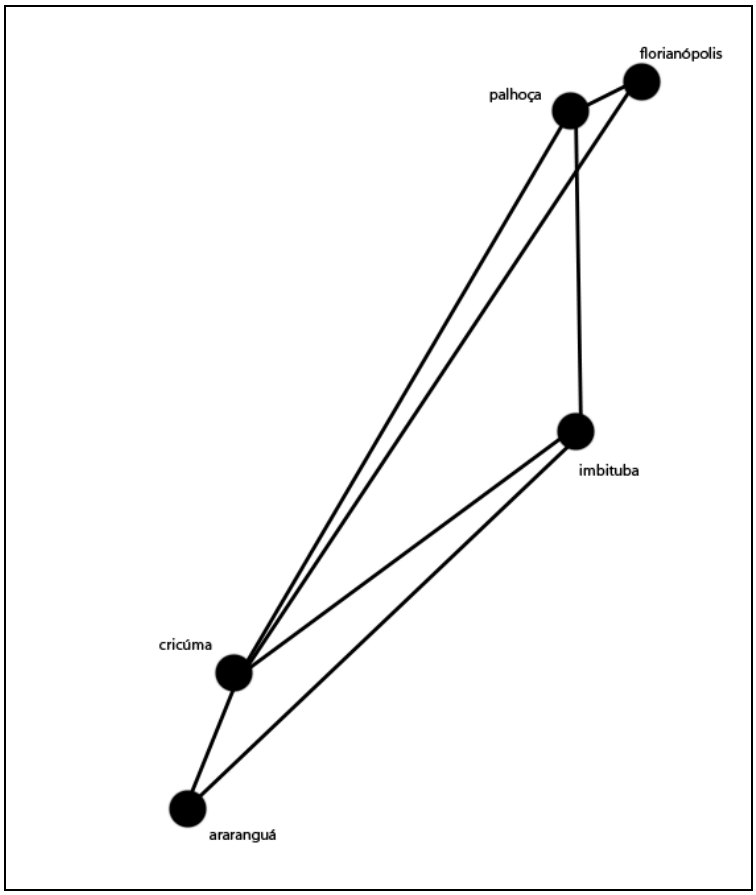


Figura 8. Exemplo algoritmo Dijkstra

O objetivo inicial é encontrar os nós mais próximos percorrendo todos do grafo, uma vez definidos aplica-se o algoritmo para definir o menos caminho baseado na distância do primeiro com os demais, e o retorno do algoritmo é conforme mostrado por Goodrich e Tamassia (2006), um vetor contendo os nós para percorrer o menos caminho.

4.2.2 Algoritmo de Floyd

O algoritmo de Floyd foi proposto por Robert W. Floyd para resolução do problema do menos caminho. Segundo Lopes (1999) o algoritmo constrói uma matriz de adjacência onde cada elemento recebe um valor do menor caminho entre os vértices.

A seguir é descritos passos para implementação do algoritmo.

Dados de entrada: Grafo G.
Dados de saída: C com menor custo de vértice de G.
Big O: $O(N^3)$
Passo 1: Para i = 1 até N (inicializar a matriz C)
Passo 1.1: Para j = 1 até N;
Passo 1.1.1: Se G(i,j) é diferente de 0 então:
Passo 1.1.1.1: Copie G(i,j) para C(i,j).
Passo 1.1.2: Se não:
Passo 1.1.2.1: Copie a para C(i,J).
Passo 2: Para i = 1 até N (a diagonal não interessa)
Passo 2.1: Copie 0 para C(i,j).
Passo 3: Para k = 1 até N:
Passo 3.1 Para i =1 até N:
Passo 3.1.1: Para j =1 até N:
Passo 3.1.1.1: Se (C(i,k) + C(k,j)) < C(i,j), então:
Passo 3.1.1.1.1: Copie C(i,k) + C(k,j) para C(i,j) (Lopes, 1999, p.370).

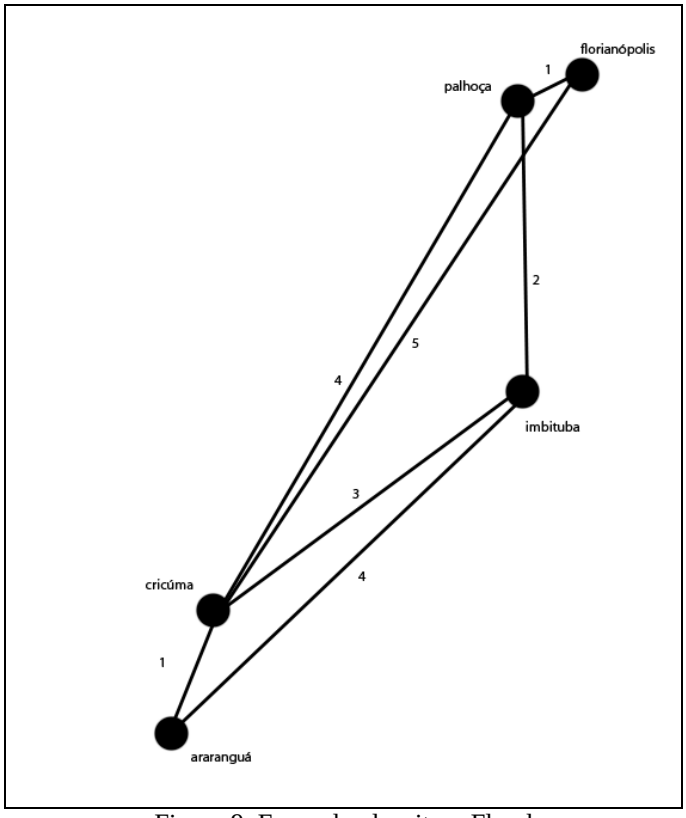


Figura 9. Exemplo algoritmo Floyd

Implementado o exemplo, segundo o algoritmo de Floyd descrito por Lopes, deve-se inicialmente inicializar a matriz de adjacência, conforme Tabela 1.

Tabela 1. Matriz de pesos do caminho mais curto na primeira iteração

Cidades	Criciúma	Araranguá	Imbituba	Palhoça	Florianópolis
Criciúma	-	1	3	4	5
Araranguá	1	-	4	-	-
Imbituba	3	4	-	2	-
Palhoça	4	-	2	-	1
Florianópolis	5	-	-	1	-

Como mostra no algoritmo deve se popular a matriz de vértices com os pesos descritos no grafo inicial, conforme a Tabela 2.

Tabela 2. Matriz de vértices antes da primeira iteração

Cidades	Criciúma	Araranguá	Imbituba	Palhoça	Florianópolis
Criciúma	0	1	1	1	1
Araranguá	2	0	2	-	-
Imbituba	3	3	0	3	-
Palhoça	4	-	4	0	0
Florianópolis	5	-	-	5	0

A Tabela 3 mostra a saída da execução do algoritmo de Floyd, tem se então a matriz do peso mínimo que indica a distância de uma cidade até a outra, e a Tabela 4 de vértices.

Tabela 3. Resultado da matriz do peso mínimo mais curto

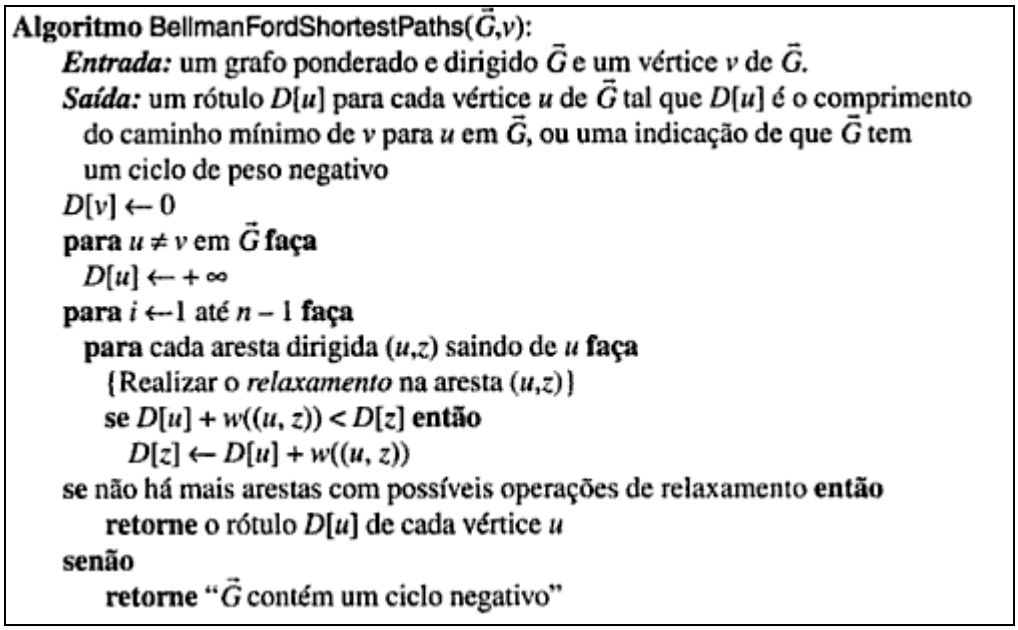
Cidades	Criciúma	Araranguá	Imbituba	Palhoça	Florianópolis
Criciúma	0	1	3	4	5
Araranguá	1	0	4	5	6
Imbituba	3	4	0	2	3
Palhoça	4	5	2	0	1
Florianópolis	5	6	3	1	0

Tabela 4. Resultado da matriz de vértices

Cidades	Criciúma	Araranguá	Imbituba	Palhoça	Florianópolis
Criciúma	0	1	1	1	1
Araranguá	2	0	2	1	1
Imbituba	3	3	0	3	4
Palhoça	4	1	4	0	4
Florianópolis	5	1	4	0	5

4.2.3 Algoritmo de Bellman Ford

O algoritmo de Ford segundo Goodrich e Tamassia(2006), encontra caminhos mínimos em suas arestas contendo pesos negativos, o grafo deve ser dirigido pois senão, isto originará um ciclo negativo, invalidando a informação da distância do grafo. A Figura 10 mostra a implementação do algoritmo:



Algoritmo BellmanFordShortestPaths($\vec{G}$,v):
Entrada: um grafo ponderado e dirigido $\vec{G}$ e um vértice v de $\vec{G}$.
Saída: um rótulo $D[u]$ para cada vértice u de $\vec{G}$ tal que $D[u]$ é o comprimento do caminho mínimo de v para u em $\vec{G}$, ou uma indicação de que $\vec{G}$ tem um ciclo de peso negativo
$D[v] \leftarrow 0$
para $u \neq v$ em $\vec{G}$ **faça**
  $D[u] \leftarrow +\infty$
para $i \leftarrow 1$ até $n - 1$ **faça**
  para cada aresta dirigida (u,z) saindo de u **faça**
    {Realizar o *relaxamento* na aresta (u,z)}
    se $D[u] + w((u, z)) < D[z]$ **então**
      $D[z] \leftarrow D[u] + w((u, z))$
se não há mais arestas com possíveis operações de relaxamento **então**
  retorne o rótulo $D[u]$ de cada vértice u
senão
  retorne "$\vec{G}$ contém um ciclo negativo"

Figura 10. Algoritmo de Bellman-Ford
Fonte: GoodRich e Tamassia (2006, p.352)

No ambiente funcional este algoritmo cria uma lista de distancias, que inicia num dado ponto, denominado início, que calcula a distancia deste aos demais pontos do grafo e retorna uma lista contendo as distancias.

5. TRABALHOS CORRELATOS

Com o avanço da tecnologia de informação, dos desenvolvedores de sistemas e da alta demanda na procura de softwares que resolvam problemas, organizando e computando dados e transformando em informações, inúmeros trabalhos surgem nessa área, segue um breve contexto dos que foram usados para embasamento da pesquisa.

5.1 ESTUDO E IMPLEMENTAÇÃO DE ALGORITMOS DE ROTEAMENTO

Trabalho de conclusão de curso do Instituto Tecnológico da Aeronáutica (ITA), autor Rudini Menezes Sampaio, aborda os problemas de roteamento com grafos, como menor caminho e arvore de custo mínimo, carteiro chinês e carteiro viajante, e desenvolve um software para solução destes problemas com a linguagem de programação Visual C++ 5.

5.2 PROTÓTIPO DE UM SISTEMA PARA PLANEJAMENTO E CONTROLE DA PRODUÇÃO EM PEQUENAS E MICRO EMPRESAS

Trabalho de conclusão de curso da Universidade do Sul Catarinense, apresentado em 2005, autor Reginaldo Kjhelin Coelho, esse trabalho visa o desenvolvimento de um plano de PCP utilizando a UML como ferramenta de modelagem para um sistema de apoio a pequena e micro empresas.

5.3 VISUALIZAÇÃO DE INFORMAÇÃO APLICADA A GERÊNCIA DE SOFTWARE

Esse trabalho é uma dissertação de mestrado apresentado a Universidade Federal do Rio Grande do Sul elaborado por Cristina Cemin, consiste no desenvolvimento de visualizador de informações, incluindo visualização e analise de grafos.

A ferramenta desenvolvida é um visualisador de informações, os grafos consistem em representar chamadas de programas, e demontram o grafo com visualização gráfica através da ferramenta Dotty.

6. LOGICINI: UM MODULO DE LOGÍSTICA DE ERP UTILIZANDO A TEORIA DE GRAFOS

O sistema proposto nesse trabalho usará as técnicas descritas no projeto, afim de integrá-las para o desenvolvimento de um software que seja capaz de determinar o menor caminho a ser percorrido por um veiculo de transporte, e analisar quais algoritmos se comportam melhor nessa função.

6.1 METODOLOGIA

Durante o ciclo de desenvolvimento do projeto foi realizado o levantamento bibliográfico com base nos assuntos supracitados. Os materiais foram encontrados na biblioteca da UNESC, sites de universidades que disponibilizam seu acervo pela internet.

Também houve um trabalho na coleta de dados em transportadoras e empresas com despacho de produtos cerâmicos para realizar a viabilidade de solução do problema.

O modulo para logística é usado para gerenciar a demanda de entrada e saída de produtos de uma empresa, bem como prever situações para aperfeiçoar a rota de um veiculo e diminuir os custos com transporte de mercadorias.

O sistema será composto de dois módulos principais: Cadastros e Logística. Os cadastros são as entradas de dados necessárias para o funcionamento do modulo de logística, o modulo da logística terá dois elementos fundamentais: montagem de rota e montagem de carga.

A montagem da carga é o recurso em que o software varre o banco de dados a procura de pedidos que estão em aberto para carregamento, dentro esses pedidos é feita uma pré-seleção por áreas de cobertura e peso máximo de carregamento. As áreas de cobertura são

divididas por estados, e cada estado tem suas cidades principais a para definir uma rota padrão. O peso máximo que o veiculo pode carregar é determinado a partir de um cadastro de veículos que será usado como base para montagem da carga.

6.2 REGRAS DO NEGÓCIO

Uma empresa X produz revestimentos cerâmicos e atende diversas regiões do Brasil, cada estado tem uma cidade principal onde é necessário a conferencia e o despacho da mercadoria para as demais cidades.

A empresa Y que compra os produtos da empresa X tem um tempo de 1 a 30 dias para receber a mercadoria adquirida. Quando o pedido de compra da empresa Y para a X é feito o estoque de produtos acabados da empresa X não é alterado, será modificado somente quando o sistema selecionar o pedido para distribuição.

A empresa X possui vários veículos cadastrados em seu sistema, tais que podem carregar desde 14 até 44 toneladas de produtos, sendo que o sistema precisará carregar gerar os pedidos até chegar no máximo de peso carregado possível.

Os transportadores designados neste modelo como Z recebem a programação de sua carga para serem carregados na empresa X com os pedidos pré-selecionados de acordo com as entregas para os Ys, como exemplo: carga com cinco pedidos de acordo com a tabela abaixo:

Tabela 5. Pedidos

Código do Pedido	Cliente	Peso	Cidade	Estado
1	Cliente 1	7 ton	Florianópolis	SC
2	Cliente 2	3 ton	Tubarão	SC
3	Cliente 3	5 ton	Florianópolis	SC
4	Cliente 4	8 ton	Criciúma	SC
5	Cliente 5	9 ton	Içara	SC

O sistema aplica algoritmos de grafo para resolver a situação e deve apresentar a seguinte solução:

Tabela 6. Ordenação dos pedidos

Código do Pedido	Cliente	Peso	Cidade	Estado
1	Cliente 1	7 ton	Florianópolis	SC
3	Cliente 3	5 ton	Florianópolis	SC
2	Cliente 2	3 ton	Tubarão	SC
5	Cliente 5	9 ton	Içara	SC
4	Cliente 4	8 ton	Criciúma	SC

Ou seja, primeiramente o veiculo deve passar na cidade principal para despacho das mercadorias que serão as capitais de cada estado e depois a partir destas cidades fazer as demais entregas.

6.3 REQUISITOS

Os requisitos abstraídos nesse sistema foram retirados pelo acadêmico com base em pesquisa realizada em transportadoras e empresas do ramo que utilizam o sistema de

distribuição de pedidos fracionados direto ao cliente. Dentre os principais usados no modulo de logística:

a) Cadastro de cidades vizinhas

b) Seleção dos pedidos para carregamento

c) Montagem da rota e ordem de carregamento da carga

O Quadro 1 mostram os requisitos principais usados para elaboração do protótipo do módulo de logística, os demais encontram-se em apêndice.

F3. Seleção de Pedidos a serem Entregues Oculto ( )
Descrição: Deverá selecionar os pedidos para ser coletados e entregá-los
Requisitos Funcionais

Nome	Restrição	Categoria	Desejável	Permanente
NF 3.1 Acesso a função	Usuários com perfil de logística podem acessar essa função	Segurança	( )	(X)
NF 3.2 Buscar Pedidos em aberto	O sistema deverá buscar na base de dados os pedidos que estão em aberto para o carregamento	Especificação	(X)	( )
NF 3.3 Peso do Veiculo	Deverá selecionar os pedidos de carregamento de acordo com o peso do veículo que transportará a carga	Especificação	(X)	( )
NF 3.4 Busca dos Pedidos	O sistema deve selecionar automaticamente os pedidos que serão carregados com base na data e no peso do pedido.	Especificação	(X)	( )
NF 3.5 Salvar Pedidos da Carga	Os pedidos que farão parte da carga serão arquivados no banco de dados para definir a rota de acordo com os algoritmos determinados	Especificação	(X)	( )

Quadro 1. Requisitos funcionais

A seleção de pedidos é determinada pelas regras do negocio, para especificar e selecionar os pedidos que serão carregados e despachados no mesmo veículo.

Os pedidos que fazer parte de uma mesma rota, são processados por meio dos algoritmo de Dijkstra, Floyd ou Ford, como resultado o sistema mostrará a melhor rota para o veiculo fazer a viajem.

6.4 FUNCIONALIDADES DO PROTÓTIPO

Os diagramas de casos de uso proporcionam uma melhor integração entre o usuário e o analista de sistema, pois é nele que são apresentadas as funcionalidades dos softwares. Esse sistema foi dividido em dois perfis principais, usuário e logística. O perfil usuário tem acesso as funcionalidades essenciais para que o modulo de logística funcione de forma integra. Abaixo os diagramas de casos de uso para os perfis:

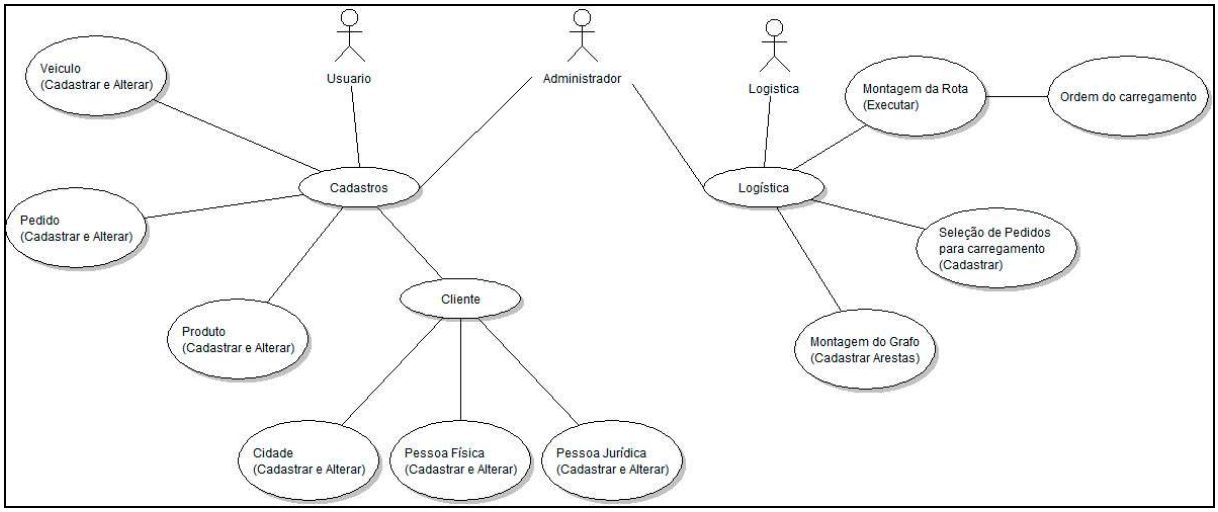


Figura 11. Diagrama de Casos de Uso

6.4.1 Fluxo de Informações

Os diagramas de atividades mostram o fluxo de informações que serão necessárias para determinada funcionalidade do software. Com ele consegue-se claramente visualizar as interações que o usuário terá no sistema. A seguir os diagramas de atividades:

A atividade de cadastro de cidades vizinhas é responsável por unir duas cidades em um grafo através da distancia. Pode existir mais de um caminho entre duas localidades, ou nenhum caminho entre elas.

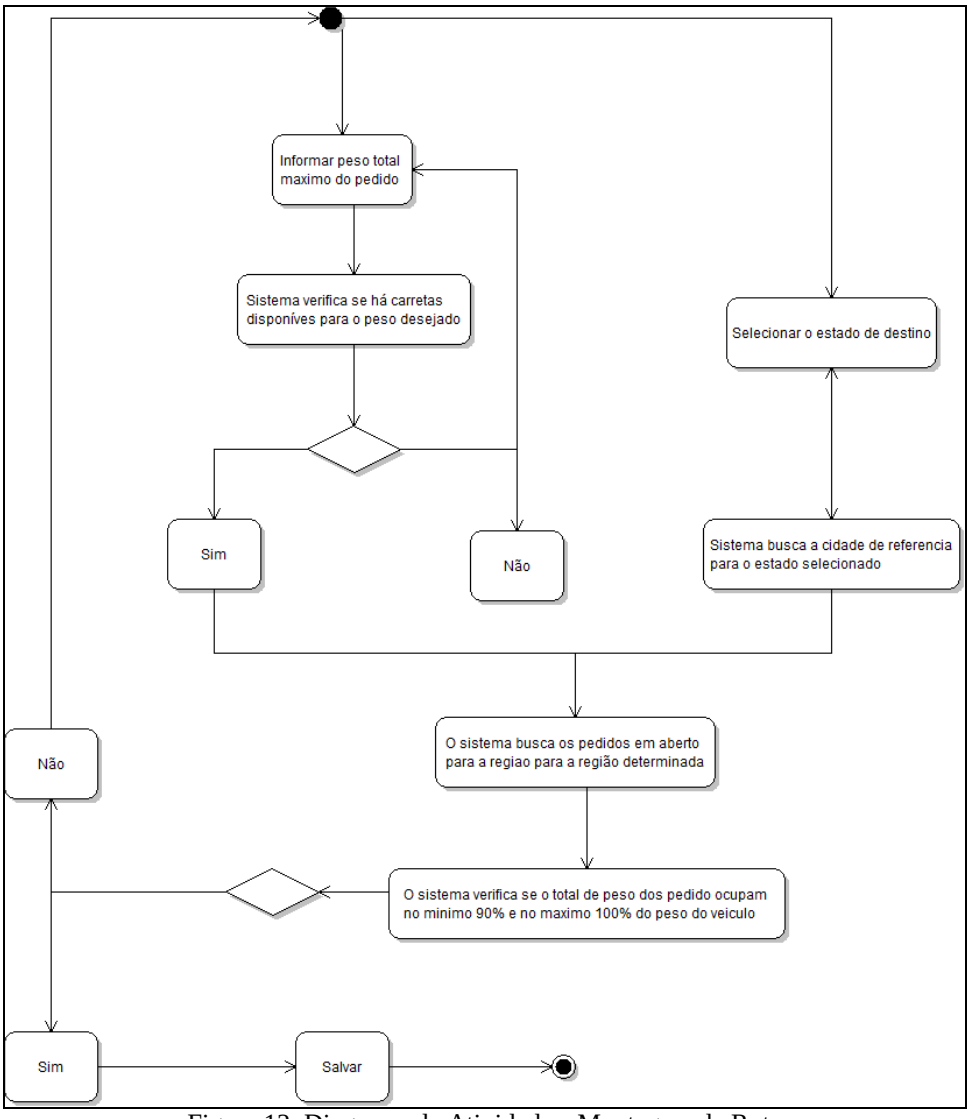


Figura 12. Diagrama de Atividades: Montagem da Rota

Montagem da Rota, é a funcionalidade referente a escolha dos pedidos que farão parte do mesmo carregamento. Tal qual deverá ser processado para encontrar o menor percurso de uma as demais cidades.

Para processar a rota basta escolher uma da lista que não foram processadas, e selecionar o algoritmo da teoria de grafos que melhor realiza o processamento da montagem da carga.

6.5 BANCO DE DADOS

O sistema de gerenciamento de dados (SGBD) é responsável por armazenar com eficácia os dados da aplicação de maneira segura, retirando da aplicação a preocupação dos dados de uma empresa.

O banco de dados utilizado nesse projeto é o MySql versão 5.5.8 com licença GPL, foi escolhido por ser um banco de dados estável , com bom desempenho e estabilidade e comporta a linguagem de programação JAVA.

6.5.1 Dicionário de dados

O dicionário de dados é uma forma de organização e visualização dos dados pertencentes ao banco. Para implementação do sistema foram usadas às tabelas que constam em apêndice. Dentre elas, destaca-se as tabelas de rota e rota_pedidos, pois nelas que se concentra o armazenamento de informações que constituem o inicio e os destinos da rota.

Tabela 7. Dicionário de Dados Tabela Rota

Tb_rota: tabela que contém informações sobre uma rota				
Chave	Atributo	Tipo	Valor Padrão	Comentário
PK	Cod_rota	INTEGER	NN	Código da Rota único, sequencial, iniciando em 0
FK	Cod_cidade_incial	INTEGER	NN	Código da Cidade que será o ponto de partida da rota
FK	Cod_veiculo	INTEGER	NN	Código do Veiculo que fará o transporte

Tabela 8. Dicionário de Dados Tabela Rota_Pedidos

Tb_rota_pedidos: tabela que contém os pedidos que são despachados no mesmo veículo				
Chave	Atributo	Tipo	Valor Padrão	Comentário
PK	Cod_rota_pedidos	INTEGER	NN	Código da Rota_Pedidos único, sequencial, iniciando em 0
FK	Cod_rota	INTEGER	NN	Código da rota
FK	Cod_pedido	INTEGER	NN	Código do pedido

A segunda tabela é a responsável por armazenar e unir os pedidos que fazer parte da mesma rota.

6.6.1 Armazenamento

Com o diagrama de classes é possível visualizar de forma genérica a estrutura usada para o armazenamento persistente no desenvolvimento da aplicação, e como se dará o relacionamento entre as classes.

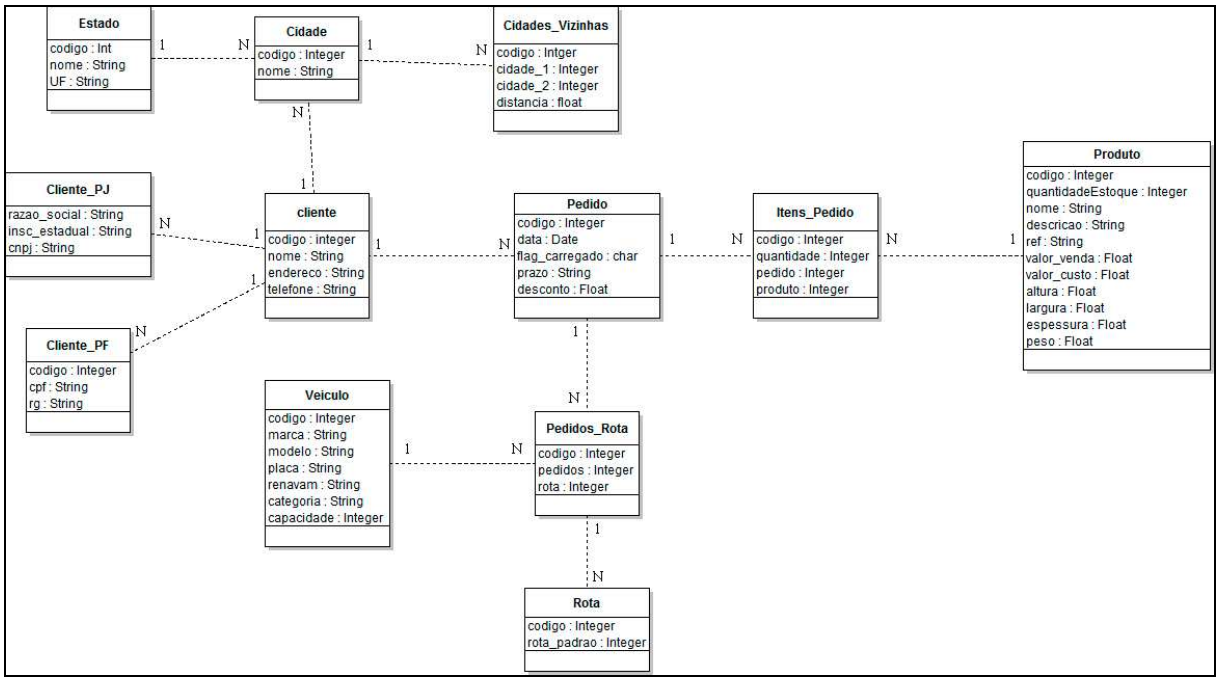


Figura 13. Diagrama de Classes

A classe Cliente se estende a outras duas classes Cliente_PF (pessoas físicas) e Cliente_PJ (pessoas jurídicas). A classe de Cidade é de fundamental importância no estudo desse software, pois é nela que obterá os dados para o processamento das rotas. A classe de Cidades Vizinhas é utilizada para armazenar as informações entre duas cidades.

Produtos e Pedidos são classes auxiliares para obtenção de dados necessária a Classe de Rotas. Rotas é classe responsável por organizar quais pedidos serão carregado num mesmo veiculo.

6.7 PROTÓTIPO

Com a documentação pronta e apta para começo do desenvolvimento dá-se inicio a fase de produção do protótipo. Nesta fase são definidas as telas do sistema, é criado o banco de dados, e a programação utilizando os algoritmos de menor caminho da Teoria de Grafos.

6.7.1 Interface

Com base em todas os dados supra citados foi desenvolvido o sistema para aplicação desktop, utilizando a linguagem de programação JAVA e o banco de dados MYSQL.

A tela inicial do sistema, é a tela de acesso, dividido em três níveis:

a) Usuario padrão: acesso apenas aos módulos de cadastro;

b) Usuario logística: acesso apenas ao módulo de logística;

c) Administrador: acesso a todas as funções do sistema.

Figura 14. Tela de Login

Após efetuar o Login o usuário entra na tela inicial do sistema, dependendo do seu tipo terá acesso as suas funções.

Figura 15. Tela Principal

O módulo de cadastro realiza as seguintes funções:

a) Cadastro e alteração de Clientes Pessoa Física;

b) Cadastro e alteração de Clientes Pessoa Jurídica;

c) Cadastro e alteração de Cidades;

d) Cadastro e alteração de Produtos;

e) Cadastro e alteração de Pedidos;

f) Cadastro e alteração de Veículos.

O modulo de logística é responsável por:

a) Seleção de pedidos para uma carga

b) Cadastro e alteração de distância entre cidades;

c) Montar a Rota para um veículo;

d) Mostrar a ordem de descarregamento.

Na tela de Seleção de pedidos o usuário escolhe o peso do veiculo que realizará o pedido e a cidade principal. A função procurar os pedidos por cidade, na cidade principal o volume de entregas tradicionalmente é maior, e as próximas cidades serão suas vizinhas. A figura 16 mostra a tela de Seleção de pedidos:

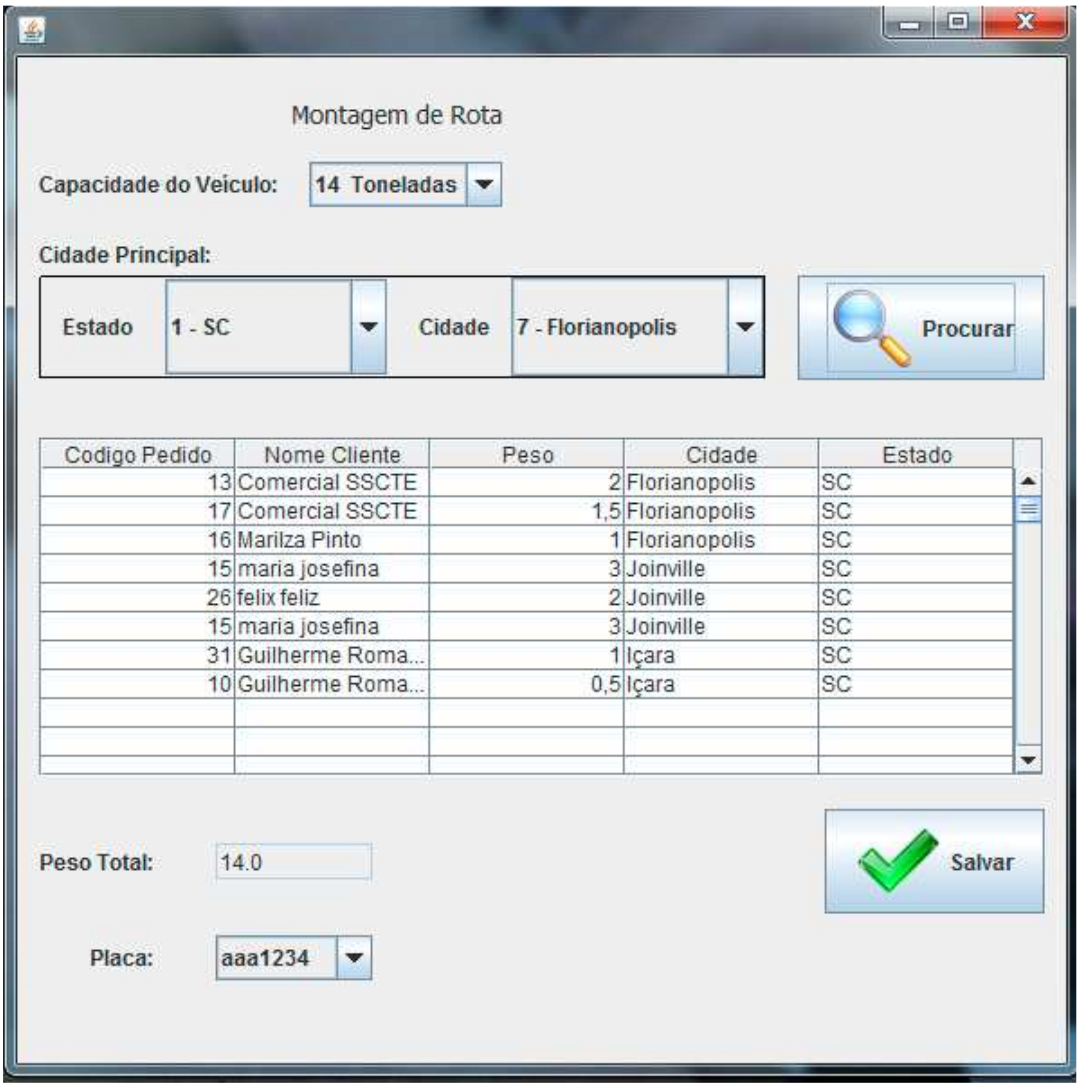


Figura 16, Tela de Seleção de Pedidos

Os pedidos selecionados em uma carga são armazenados no banco de dados e na tela de montagem da rota estes pedidos são ordenados por entregas. Podendo ser processados por meio dos algoritmos de Bellman-Ford, Floyd-Warshall, e Dijkstra.

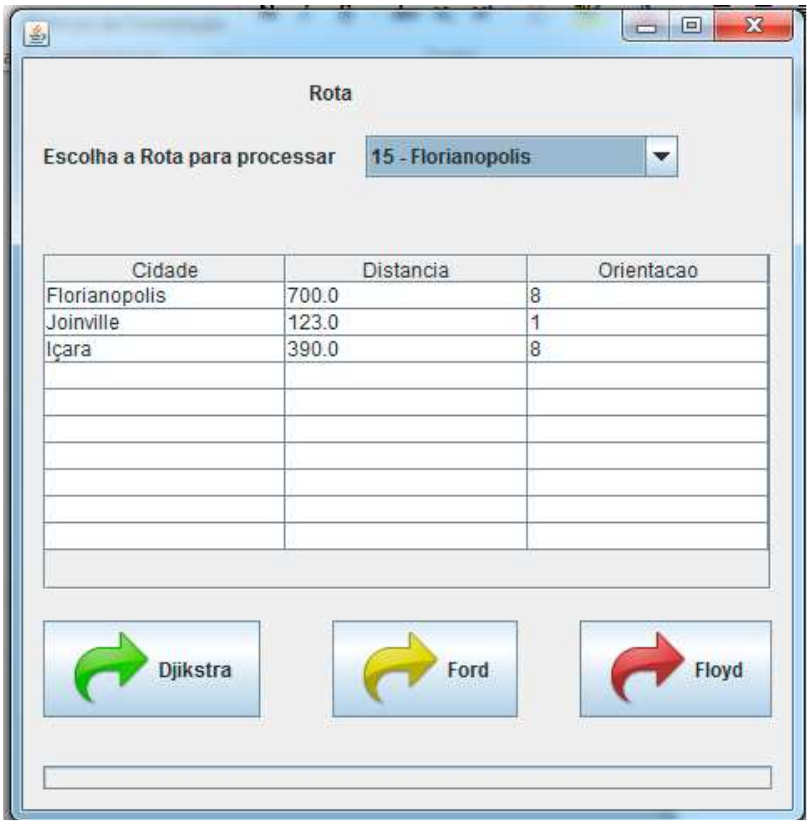


Figura 17. Tela do Processamento da Rota

6.7.2 Processamento das Rotas

O processamento da rota pode ocorrer utilizando os três tipos de algoritmos de menor caminho da teoria de grafos: Dijkstra, Floyd-Warshall e Belmann-Ford. Todos os três algoritmos cumpriram com eficácia o roteamento das cidades.

O maior problema encontrado com todos os algoritmos foi encontrar a rota para um conjunto de cidades, e não a melhor rota de uma cidade para outra. Outra dificuldade encontrada foi a de que o grafo teria que ser dinâmico, ou seja, de acordo com as cidades

cadastradas no banco de dados, e a distancia entre elas. Pelo fato do grafo ser dinâmico não podia usar o código da cidade para identificação de um ponto no mesmo, para esse problema criou-se um Array chamado identificação que guarda o código da cidade e gera um código de 0 a N para esse ponto.

Os algoritmos supracitados são definidos como uma função da classe Grafo. Para execução de cada é mandado uma matriz [n][n] e um Arraylist contendo a lista das cidades buscadas.

A figura x demostra o problema a ser resolvido, um veiculo está na empresa para despacho de sua mercadoria, e deseja encontrar a rota com menor caminho para fazer suas entregas. O ponto inicial é a sede da empresa, os destinos são os pontos K,X,Z.

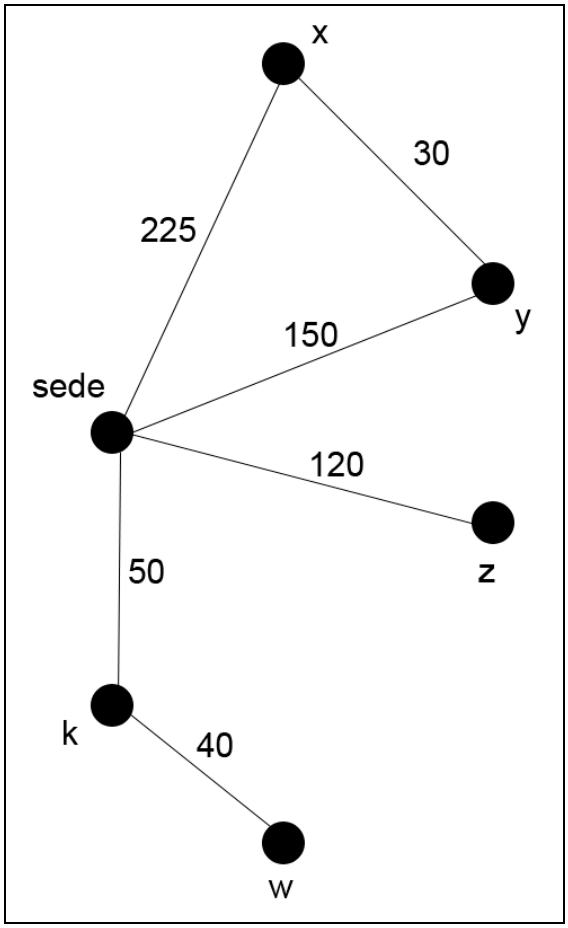


Figura 18. Grafo

Pode existir mais de uma solução para o problema. Cada algoritmo tem certa forma de tratar o roteamento entre as cidades, porém o resultado, em termos de eficiência é muito próximo. A cada subtítulo a seguir será demostrado por meios gráficos como acontece o processamento.

6.7.2.1 Algoritmo de Dijkstra

O algoritmo de Dijkstra encontra o menor caminho entre duas cidades no grafo. Por esse motivo para encontrar uma lista de cidades o processamento é feito baseando na cidade principal até uma das cidades buscadas, a menor distancia para uma destas cidades será o caminhamento no grafo.

	X	Z	K
Empresa	180 km	120	50

Menor distância: K

	X	Z
Loja K	230 km	170 km

Menor distância: Z

	X
Loja Z	300 km

Trajeto: K, Z, X.

Quadro 2. Solução Dijkstra

6.7.2.2 Algoritmo de Floyd-Warshall

O algoritmo de Floyd Warshall gera uma matriz com as distancia entra todos os pontos do grafo. De acordo com essa matriz a próxima cidade a ser caminhada no grafo é aquela que tem a menor distancia do ponto inicial a qualquer outra da lista de cidades buscadas.

	Sede	X	Y	Z	W	K
Sede	0	180	150	120	90	50
X	180	0	30	200	270	230
Y	150	30	0	270	240	200
Z	120	200	270	0	210	170
W	90	270	240	210	0	40
K	50	230	200	170	40	0

Pedidos: X,Z,K.
Menor Distancia Sede: K
Menor Distancia K: Z
Menor Distancia Z: X

Rota: K, Z, X

Quadro 3. Solução Floyd-Warshall

6.7.2.3 Algoritmo de Bellman-Ford

O algoritmo de Bellman-Ford cria a partir do ponto inicial uma lista de distancia entre todos os pontos do mapa. Como descrito nos algoritmos anteriores a logica para este não é diferente, a partir da menor distancia encontrada na lista, será o próximo ponto inicial e vai ser removida da lista de cidades procuradas até a lista ficar vazia. As cidades encontradas são armazenadas em outra lista na ordem que foram encontradas.

	Sede	X	Y	Z	W	K
Sede	INF	180	150	120	90	50

Menor distância com pedido: K

	Sede	X	Y	Z	W	K
Loja K	50	230	200	170	40	INF

Menor distância com pedido: Z

	Sede	X	Y	Z	W	K
Loja Z	120	200	270	INF	210	170

Menor distância com pedido: X

Rota: K, Z, X

Quadro 4. Solução Bellman-Ford

6.8 ANÁLISE DOS ALGORITMOS

Todos os algoritmos acima descritos foram implementados em JAVA, e para cada qual teve um custo de processamento diferente. A configuração do computador que realizou o teste é:

a) Processador Intel Core 2 duo 7200 2.53 Ghz;

b) Memoria 4GB 800mhz em dual channel;

c) HD 500 Gb 7200 RPM;

d) Sistema operacional Windows 7 64 bits;

e) Java 1.6.0_25;

f) Banco de dados MYSQL 5.5

No primeiro teste foi realizada uma busca com dezessete pedidos distribuídos entre sete cidades em um grafo com vinte e três cidades. Cada algoritmo foi processado dez vezes excluindo o maior e o menor resultado de cada processamento com base nos dados:

a) Cidades mapeadas: 23

b) Cidades buscadas: 7

c) Pedidos buscados: 17

Tabela 9. Análise dos algoritmos (Primeiro Teste)

Iteração	Dijkstra	Bellman-ford	Floyd-warshal
1	~~3 ms~~	18 ms	58 ms
2	2 ms	18 ms	~~231 ms~~
3	1 ms	~~65 ms~~	48 ms
4	1 ms	17 ms	45 ms
5	~~0 ms~~	30 ms	84 ms
6	2 ms	17 ms	76 ms
7	2 ms	15 ms	39 ms
8	2 ms	~~13 ms~~	38 ms
9	1 ms	19 ms	70 ms
10	0 ms	14 ms	~~37 ms~~
Média	**1,375 ms**	**18,5 ms**	**57,25 ms**

Nessa tabela pode se observar que o algoritmo de Dijkstra é relativamente mais eficiente que os de Ford e Floyd para este caso. A figura 19, mostra a linha do tempo entre as iterações e o tempo gasto para processamento.

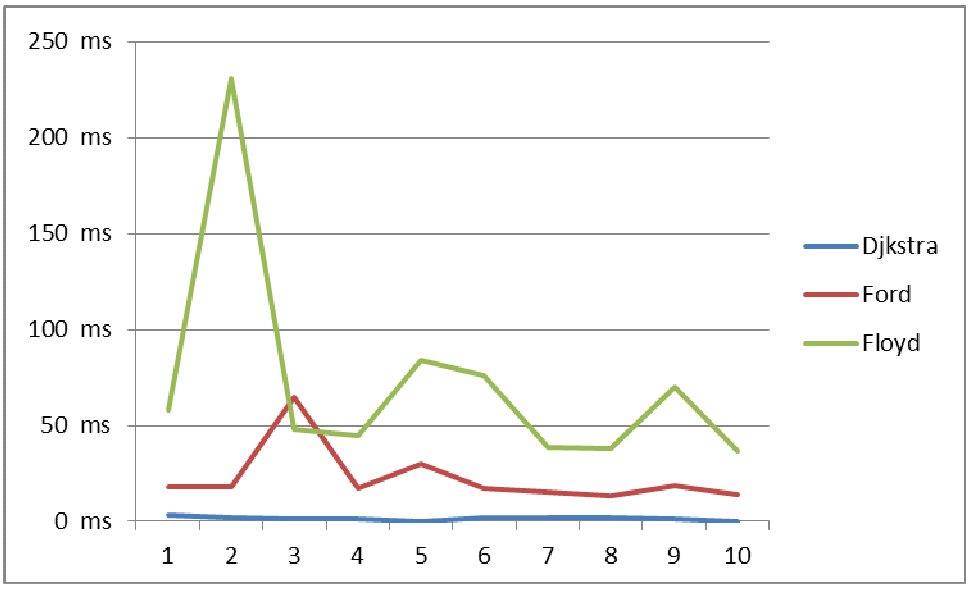


Figura 19. Comparativo dos algoritmos (Primeiro Teste)

O algoritmo de Djkstra foi constante em todo o tempo de análise variando em no máximo 1 milissegundo. Ford foi constante a partir da 4ª iteração com variações de 13 à 19 milissegundos. O algoritmo de Floyd teve altos e baixos durante todo o ciclo de análise.

Com um ciclo de processamento um dez vez maior, usando os seguintes dados:

a) Cidades no grafo: 23

b) Cidades buscadas: 70

c) Pedidos buscados: 170

Tabela 10. Análise dos algoritmos (Segundo Teste)

Iteração	Dijkstra	Bellman-ford	Floyd-warshal
1	39 ms	270 ms	42 ms
2	18 ms	268 ms	~~37 ms~~
3	37 ms	231 ms	49 ms
4	19 ms	212 ms	~~73 ms~~
5	13 ms	216 ms	38 ms
6	38 ms	234 ms	43 ms
7	17 ms	~~194 ms~~	63 ms
8	~~11 ms~~	202 ms	42 ms
9	12 ms	~~274 ms~~	39 ms
10	~~49 ms~~	204 ms	59 ms
Média	**24,125 ms**	**227,375 ms**	**46,875 ms**

O custo de processamento de Floyd não foi alterado devido que seu processamento é feito apenas uma vez entre todos os pontos do grafo, compondo uma matriz de distancia, e a partir desta é encontrado as rotas. Os algoritmos de Dijkstra e Ford aumentar proporcionalmente ao tamanho da complexidade em dez vezes. A figura 20 mostra a linha do tempo em execução.

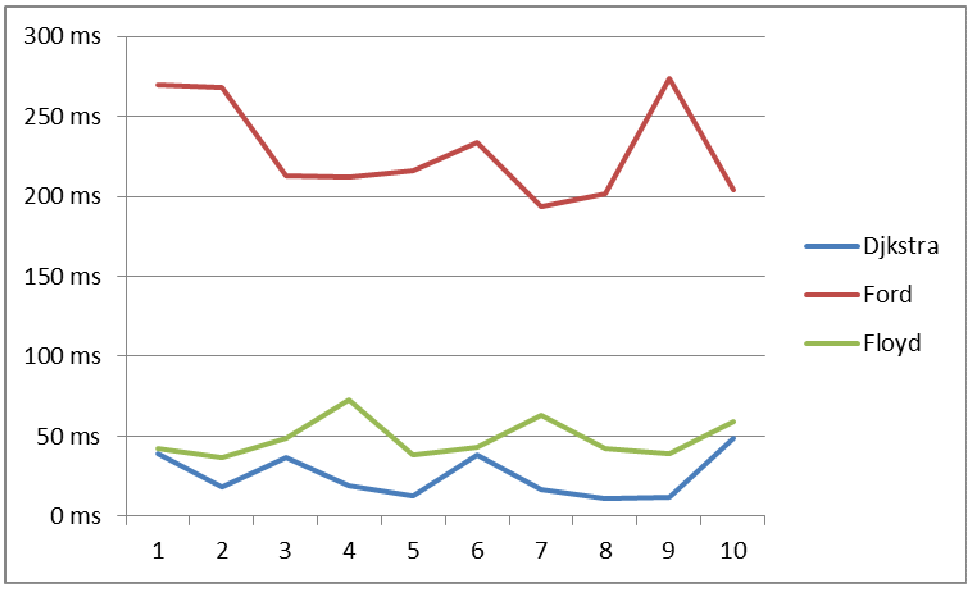


Figura 20. Comparativo dos algoritmos (Segundo Teste)

Para esse caso, o algoritmo de Dijkstra ainda se mostrou mais eficiente que os demais, porém podemos perceber que em uma aplicação que exija um maior roteamento do que números de pontos no grafo, o algoritmo de Floyd consegue maior agilidade, dependendo do grau de complexidade do problema.

RESULTADOS OBTIDOS

Os resultados obtidos por meio deste trabalho de pesquisa foram a analise comparativo dos algoritmos de menor caminho: Dijkstra, Floyd-Warshall, e Bellman-Ford sendo que foi desenvolvido um protótipo para servir de base para implementação dos algoritmos.

Foi implementado o módulo auxiliar de cadastros (cliente, produtos, pedidos, cidades) que dão suporte, e tornam funcional o módulo de logística. Dentro deste foi desenvolvido uma função para busca no banco de dados dos pedidos que compõe a mesma rota. E a partir desta seleção, foram abstraídas as cidades para processamento.

Para processar uma lista de cidades foi gerado um grafo que pode ser gerado dinamicamente de acordo com a necessidade do usuário. Tendo a lista de cidades(destinos) e o mapa(grafo), utilizou os algoritmos de Dijkstra, Bellman-Ford e Floyd-Warshall para processar a rota.

A cada processamento foi analisado o que cada algoritmo poderia proporcionar para cada caso. O algoritmo de Dijkstra foi eficiente em todos os casos testados, porém quando aumentado o numero de destinos este teve um aumento relativo de tempo para processamento, caso que não ocorreu com o algoritmo de Floyd, pois este usa apenas uma matriz que é executada uma única vez que compõe as distancia entre todas as cidades, e por isso independe o numero de buscas. O algoritmo de Ford em todos os testes não teve um bom desempenho.

CONCLUSÃO

Atualmente os sistemas ERPs, são parte constituinte de qualquer tipo de empresa, para facilitar o controle e documentação das informações que passam por ela. Trabalhar sem ter um sistema que dê apoio à empresa é uma tarefa que está entrando em extinção, devido a grande gama de softwares que circula no mercado, e das facilidades que estes proporcionam ao cliente.

A Teoria de Grafos estuda a relação entre objetos, a relação que foi abordada no trabalho é a de menor caminho entre os pontos. Para tal foram estudados os algoritmo de Dijkstra, Bellman-Ford e Floyd-Warshall.

Neste trabalho foi desenvolvido um protótipo de um modulo que pode ser implantado em um sistema ERP para despacho de produtos cerâmicos. O problema de entregar vários pedidos para várias cidades.

Um grande obstáculo encontrado durante o ciclo de desenvolvimento do protótipo foi de como encontrar uma rota que contemple todas as cidades buscadas, já que os algoritmos de menor caminho encontram o menor percurso dados dois pontos no grafo, e não uma lista de destinos.

Outra grande dificuldade foi de fazer o grafo dinâmico, partindo de cidades cadastradas no banco de dados, com a informação da distancia entre elas. Para isso foi desenvolvido um método que busca as cidades que são vizinhas, atribuem a elas um número de identificação no grafo de zero à N, e a partir desse índice é atribuído o valor de uma aresta a outra.

O algoritmo de Dijkstra anda em um grafo a procura do menor caminho de um ponto a qualquer outro do grafo, valorando cada ponto de chegada com o valor de sua aresta. Floyd-Warshall percorre todos os nos do grafo e monta uma matriz de distancia, a qual está

contida todas as informações do menor caminho de um ponto a qualquer outro. Bellman-Ford, para cada ponto inicial ele cria uma lista de distancia deste para os demais pontos do grafo.

Conseguiu-se analisar o poder de cada algoritmo, todos cumprem o seu dever, com algumas diferenças de processamento entre si. Nos testes Dijkstra se mostrou com melhor desempenho, em seguida de Floyd, e Ford. Podendo se observar que a medida que se aumenta a busca Ford passa a ter um melhor desempenho.

Para implementação desse protótipo em uma organização deve-se atualizá-lo para as reais necessidades da organização a fim de melhorá-lo já que nesse trabalho foram implementadas apenas suas funcionalidades principais.

Todos os objetivos do trabalho foram alcançados utilizando as ferramentas descritas no trabalho, os algoritmo foram implementados num protótipo para análise de desempenho e funcionalidade. Criou se também uma seleção de pedidos para formar a montagem da rota de um veículo no módulo de logística e apresentou a solução de melhor caminho.

Ao fim deste trabalho, algumas sugestões para trabalhos futuros:

a) Desenvolvimento de um protótipo para diminuir a distancia entre paradas de ônibus utilizando a Teoria de Grafos;

b) Teste de qualidade no protótipo Logicini;

c) Analise e viabilidade de implantação de software em empresas de transporte de cargas para indústria cerâmica.

REFERÊNCIAS

ANTON; RORRES. **Algebra linear:** com aplicações85. 8. ed. Sao Paulo: Artmed, 2000.

BRAUDE, Eric. **Projeto de software:** da programação a arquitetura: uma abordagem baseada em java. Porto Alegre: Artmed, 2004. 625 p.

BROOKSHEAR, J. Glenn. **Ciência da computação:** uma visão abrangente. 7. ed. Sao Paulo: Artmed, 2003.

CHAMBERS, Stuart; JOHSTON Robert; SLACK Nigel; **Administração da produção**. 2 ed. São Paulo: Atlas, 2007.

CHOPRA, S.; MEINDL, P. **Gerenciamento da Cadeia de Suprimentos** – Estratégia, Planejamento e Operação. Prentice Hall, 2003.

COELHO, Flavio. **Computação científica com python.** Petrópolis: Flavio Codeço Coelho, 2007.

CORRÊA, H. L. ***ERPs:*** por que as implantações são tão caras e raramente dão certo? In: SIMPÓSIO DE ADMINISTRAÇÃO DA PRODUÇÃO, LOGÍSTICA E OPERAÇÕES INDUSTRIAI**s**, 1. *Anais...* São Paulo: FGV-SP, 1998..

FOWLER, Martim. **Padrões de arquitetura de aplicações corporativas.** Porto Alegre: Artmed, 2003.

FOWLER, Martin. **UML essencial:** um breve guia para a linguagem padrão de desenvolvimento de objetos. 3. ed. Porto Alegre: Artmed, 2004.

GAMBÔA, F e Filho, E.B. **Fatores Críticos De Sucesso Na Implementação De Sistemas Integrados De Gestão De Recursos**. X Simpósio de Engenharia de Produção (SIMPEP), UNESP, Bauru. 2003.

GOODRICH, Michael; TAMASSIA, Roberto. **Estrutura de dados e algoritmos em java.** 4. ed. Porto Alegre: Artmed, 2006.

GOODRICH, Michael; TAMASSIA, Roberto. **Projeto de algoritmos:** fundamentos, análises e exeplos da internet. Porto Alegre: Artmed, 2002.

GUEDES, Gilleanes T. A.. **UML:** Uma Abordagem Pratica. 3. ed. São Paulo: Novatec, 2008. 236 p.

GUSMAO C.M.G e MOURA H.P.; **Riscos para ambientes de múltiplos projetos de desenvolvimento de software**. Tese de Doutorado. UFPE, Recife, 2007.

HORSTMANN, Cay. **Conceitos de computação com o essencial de JAVA:** 3. ed. Sao Paulo: Artmed, 2003.

JOHNSON, Steven. **Emergência:** a dinâmica de redes em formigas, cérebros, cidades e softwares. Rio de Janeiro: Jorge Zahar, 2001.

KERZNER, Harold; **Gestão de Projetos**: as melhores práticas. 2ed. São Paulo: Artmed, 2004.

LAUDON, Kenneth C. e LAUDON, Jane P. Management Information Systems. 4 ed. Upper Saddle River. Prentice Hall, 1996.

LONCHAMP, J. A. **Structured Conceptual and Terminological Frameword for Software Process Enginnering.** International conference on software process. Berlin, Los Alamitos, 1993.

LOPES, Arthur Vargas. **Estrutura de dados:** para a construção de software. Canoas: Ulbra, 1999.

MARTINS, José Carlos Cordeiro. **Tecnicas para desenvolvimento de projeto de software:** Rio de Janeiro: Brasport, 2007. 465 p.

NETTO, P. O. Boaventura. **Grafos:** teoria, modelos e algoritmos. Sao Paulo: Adegar Blucher, 1996.

OSTRENGA, M. R.; OZAN, T. R.; McILHATTAN, R. D.; HARWOOD, M. D., **Guia da Ernst & Young para a gestão total de custos**. Rio de Janeiro, ed. 1, ed. Record, 1993.

PRESSMAN, R. S. **Engenharia de Software**. São Paulo: Makron Books, 2006.

RAMOS, Ricardo Argenton. **Treinamento prático em UML:** desenvolva e gerencie seu projetos com essa sensecional ferramenta. Sao Paulo: Universo Dos Livros, 2006. REZENDE, Denis Alcides. **Engenharia de software e sistemas de informação:** 3. ed. Rio de Janeiro: Brasport, 2005.

REIS, Christian. **Caracterização de um Modelo de Processo para Projetos de Software Livre.** 3. ed. São Paulo: 2003. 236 p.

SEVERO FILHO, Jõao. **Administração de Logística Integrada:** Materiais, PCP e Marketing. 2. ed. Rio de Janeiro: E-papers, 2006. 310 p.

SHAW, Alan C.. **Sistemas e Softwares de tempo real:** Porto Alegre: Artmed, 2001.

SCHROEDER, Isley Roberto. **O paradigma da informática:** gerar lucro para as empresas. Sao Paulo: Ampub, 2002.

SOMMERVILLE, Ian. **Software Enginnering:** 4. ed. England: Addison Wesley, 1992.

SOMMERVILLE, Ian; **Engenharia de Software**. São Paulo: Pearson, 2004.

SOUZA, Cesar e ZWICKER, Ronaldo. Sistemas Integrados de Gestão Empresarial: Estudos de casos de implementação de Sistemas ERP. Tese de Mestrado. USP, São Paulo, 2000.

WOOD JR., T., CALDAS, M. P. **Modas e modismos em gestão**: pesquisa exploratória sobre adoção e implementação de ERP. In: ENCONTRO NACIONAL DOS PROGRAMAS DE PÓS-GRADUAÇÃO EM ADMINISTRAÇÃO, Foz do Iguaçu. Anais...Rio de Janeiro: Anpad, 1999.

Apêndice A – Requisitos Funcionais

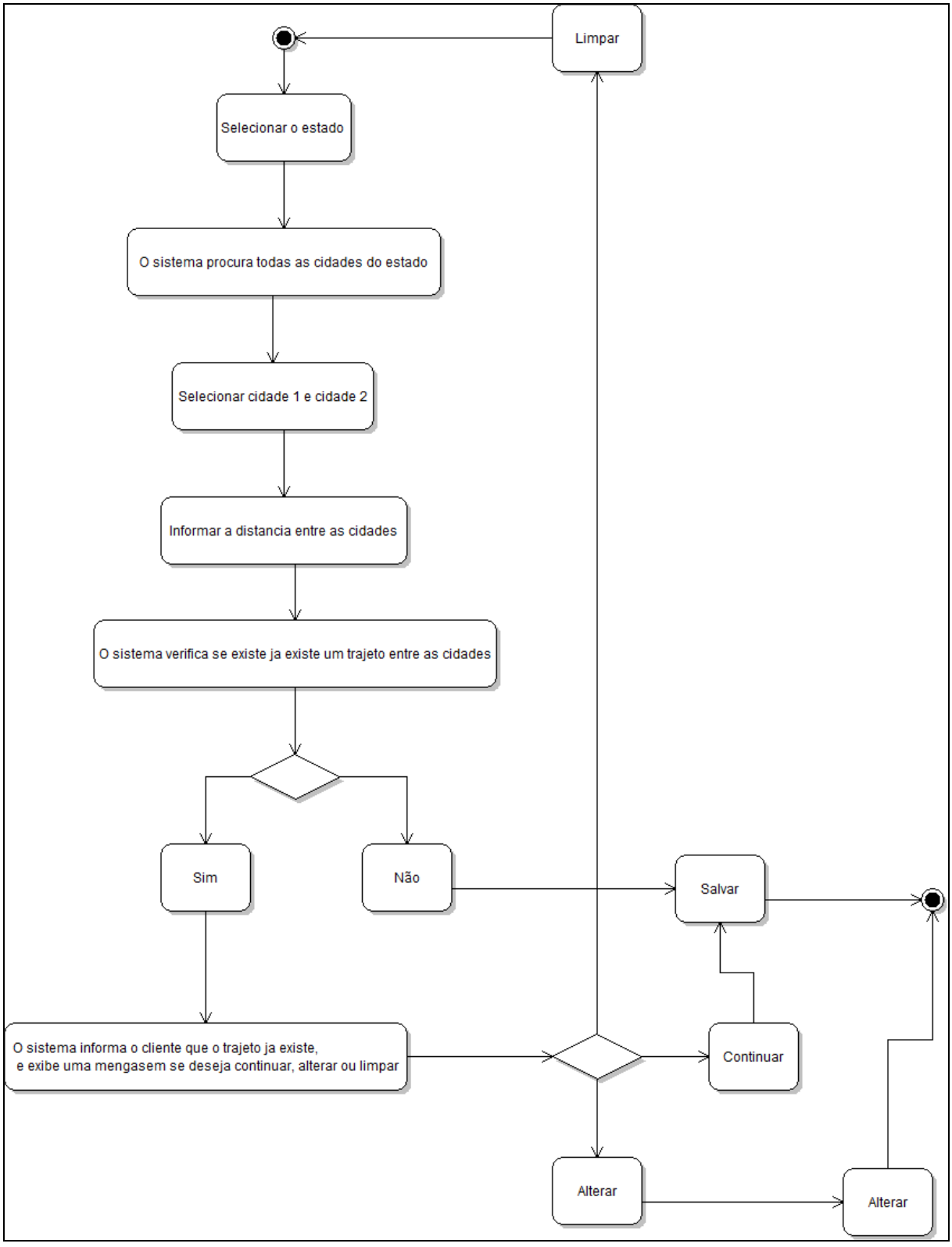
F1. Cadastro de Cidades				Oculto ( )
Descrição: O Sistema deverá cadastrar uma cidade				
Requisitos Não Funcionais				
Nome	Restrição	Categoria	Desejavel	Permanente
NF 1.1 Cadastro de cidades	Todos os usuarios podem acessar essa função	segurança	( )	(X)
NF 1.2 Codigo da cidade	O codigo da cidade deverá ser gerado automaticamente	especificação	(X)	( )
NF 1.3 Nome da cidade	Não pode existir nome de cidades iguais no mesmo estado	especificação	(X)	( )

F2. Cadastro de Cidades Vizinhas				Oculto ( )
Descrição: O Sistema deverá cadastrar uma distancia entre duas cidades				
Requisitos Não Funcionais				
Nome	Restrição	Categoria	Desejavel	Permanente
NF 2.1 Cadastro de cidades Vizinhas	Usuarios com perfil de logística podem acessar essa função	segurança	( )	(X)
NF 2.2 Codigo da cidade	O codigo da cidade deverá ser gerado automaticamente	especificação	(X)	( )
NF 2.3 Distancia entre cidades	Pode existir mais de uma distancia entre duas cidades	especificação	(X)	( )
NF 2.4 Cidades sem distancia	Não pode ter nenhuma cidade sem caminho para qualquer outra	especificação	(X)	( )

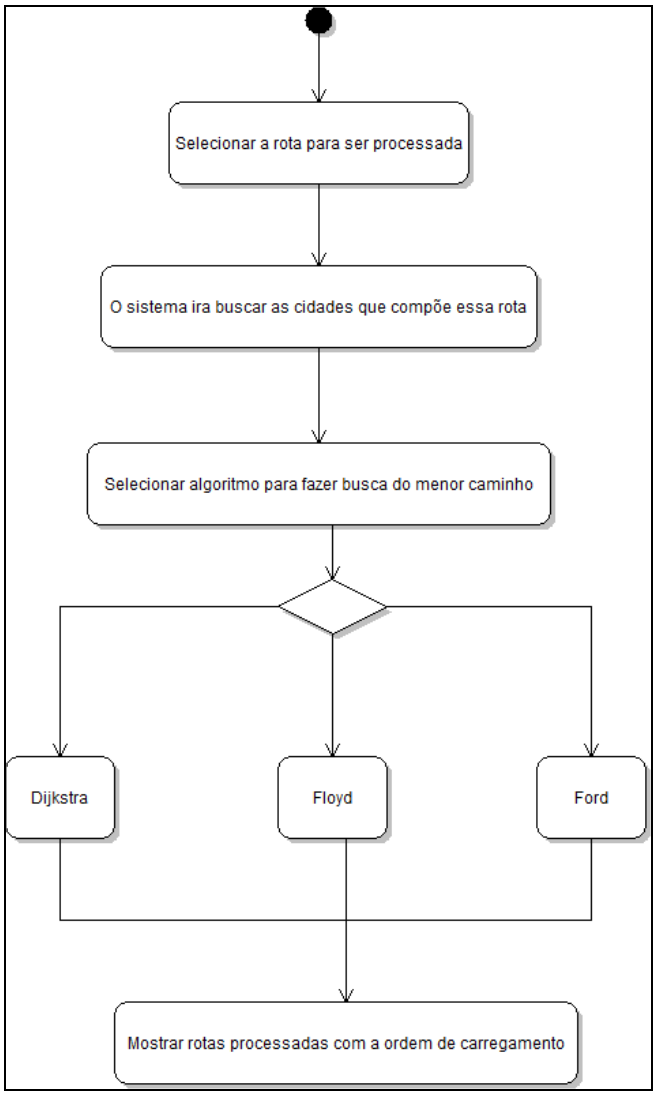
F4. Estabelecer a Rota				Oculto ( )
Descrição: O sistema deverá montar a rota de acordo com os pedidos da carga				
Requisitos Não Funcionais				
Nome	Restrição	Categoria	Desejavel	Permanente
NF 4.1 Acesso a função	Usuarios com perfil de logística podem acessar essa função	segurança	( )	(X)
NF 4.2 Buscar Pedidos da Carga e aberto	O sistema deverá buscar na base de dados os pedidos de uma carga em aberto	especificação	(X)	( )
NF 4.3 Prerapar Processamento	O processamento ocorrerá com base nos pedidos da carga selecionados e as respectivas cidades que farão parte da rota	especificação	(X)	( )
NF 4.4 Processamento	Independente do algortimo utilizado, o sistema deverá mostra a ordem das cidades para distribuição	especificação	(X)	( )
NF 3.5 Apresentação	O sistema imprime os pedidos para fazer a coleta em ordem de carregamento/descarregamento	apresentação	(X)	( )

Apêndice B – Diagramas de Atividades

Cadastro de Cidades vizinhas:



Processamento das Rotas:


Selecionar a rota para ser processada
O sistema ira buscar as cidades que compõe essa rota
Selecionar algoritmo para fazer busca do menor caminho
Dijkstra
Floyd
Ford
Mostrar rotas processadas com a ordem de carregamento

Apêndice C – Dicionário de Dados

tb_cidade: tabela que armazena as informações de uma cidade				
Chave	**Atributo**	**Tipo**	**Valor Padrão**	**Comentário**
PK	cod_cidade	INTEGER	NN	Codigo da cidade único, sequencial, inciando em 0
	cidade	VARCHAR (40)	NN	Nome da cidade
FK	cod_estado	INTEGER	NN	Codigo do estado

tb_cidade_vizinha: tabela que contém informações sobre a distancia entre as cidades				
Chave	**Atributo**	**Tipo**	**Valor Padrão**	**Comentário**
PK	cod_cidade_vizinha	INTEGER	NN	Codigo da cidade_vizinha único, sequencial, inciando em 0
FK	cod_cidade_1	INTEGER	NN	Codigo da cidade
FK	cod_cidade_2	INTEGER	NN	Codigo da cidade
	distancia	FLOAT	NN	Distancia entre a cidade 1 e a cidade 2

tb_cliente: tabela que armazena as informações de um cliente				
Chave	**Atributo**	**Tipo**	**Valor Padrão**	**Comentário**
PK	cod_cliente	INTEGER	NN	Codigo do Cliente único, sequencial, inciando em 0
	nome	INTEGER	NN	Nome do Cliente
	endereco	INTEGER	NN	Endereço do Cliente
	telefone	FLOAT		Telefone do Cliente
FK	cod_cidade	INTEGER	NN	Codigo da Cidade do Cliente

tb_cliente_pf: tabela que armazena as informações de um cliente pessoa física				
Chave	**Atributo**	**Tipo**	**Valor Padrão**	**Comentário**
PK	cod_cliente_pf	INTEGER	NN	Codigo do Cliente Pessoa Física único, sequencial, inciando em 0
FK	cod_cliente	INTEGER	NN	Codigo do Cliente
	cpf	VARCHAR(11)	NN	CPF do Cliente Pessoa Física sem formatação
	rg	VARCHAR(10)		RG do Cliente Pessoa Física sem formatação

tb_cliente_pj: tabela que armazena as informações de um cliente pessoa jurídica				
Chave	**Atributo**	**Tipo**	**Valor Padrão**	**Comentário**
PK	cod_cliente_pj	INTEGER	NN	Codigo do Cliente Pessoa Jurídica único, sequencial, inciando em 0
FK	cod_cliente	INTEGER	NN	Codigo do Cliente
	razao_social	VARCHAR(50)	NN	Razao Social da Empresa
	cnpj	VARCHAR(14)	NN	CNPJ da Empresa sem formatação
	ie	VARCHAR(13)		Inscrição Estadual da Empresa sem formatação

tb_estado: tabela que armazena as informações de um estado				
Chave	**Atributo**	**Tipo**	**Valor Padrão**	**Comentário**
PK	cod_estado	INTEGER	NN	Codigo do Estado único, sequencial, inciando em 0
	nome	VARCHAR(40)	NN	Nome do Estado

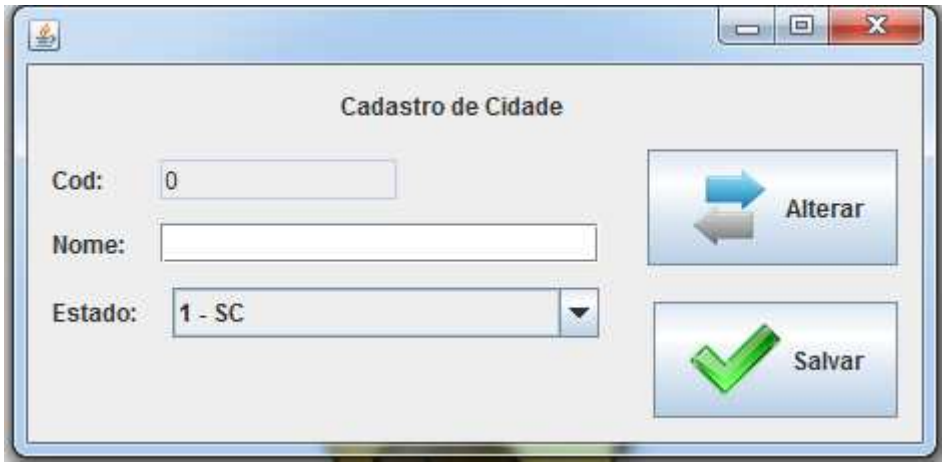
tb_pedido: tabela que armazena as informações de pedido				
Chave	**Atributo**	**Tipo**	**Valor Padrão**	**Comentário**
PK	cod_pedido	INTEGER	NN	Codigo do Pedido único, sequencial, inciando em 0
FK	cod_cliente	INTEGER	NN	Codigo do Cliente
	data_inicial	DATE	NN	Data que foi realizado o pedido
	desp	VARCHAR(1)		Flag que informa se o pedido foi despachado (s) ou não (n)
	sit	VARCHAR(1)		Flag que informa a situação do pedido ativo (a), cancelado (c), entregue (e)
	data_entrega	DATE		Data que foi feita a entrega no Cliente
	desc	VARCHAR(100)		Descrições diversas sobre o pedido

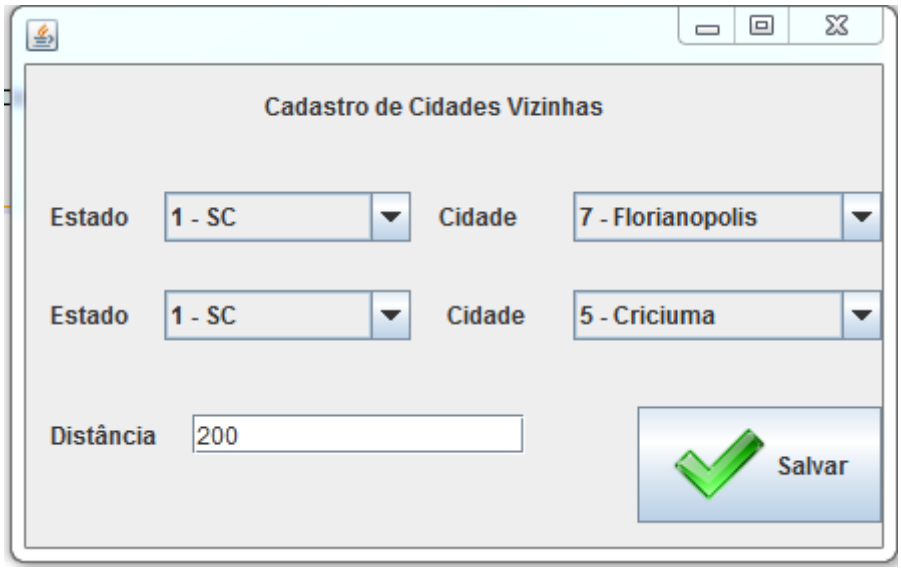
tb_pedidos_produtos: tabela que contem os itens do pedido				
Chave	**Atributo**	**Tipo**	**Valor Padrão**	**Comentário**
PK	cod_pedidos_produtos	INTEGER	NN	Codigo da relação Pedido com Produtos único, sequencial, inciando em 0
FK	cod_pedido	INTEGER	NN	Código do Pedido
FK	cod_produto	INTEGER	NN	Código do Produto
	quantidade	VARCHAR(1)	NN	Quantidade de determinado produto para este pedido

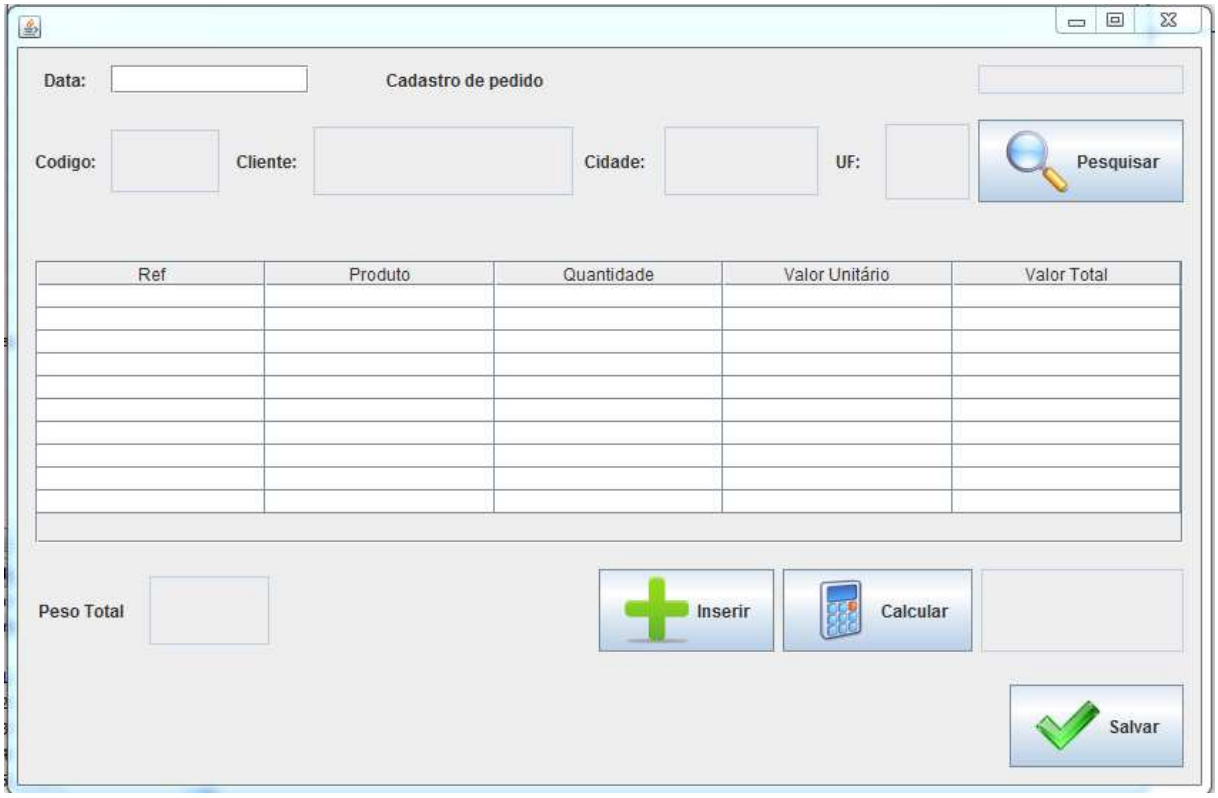
tb_produto: tabela que armazena as informações do produto				
Chave	**Atributo**	**Tipo**	**Valor Padrão**	**Comentário**
PK	cod_produto	INTEGER	NN	Codigo do Produto único, sequencial, inciando em 0
	nome	INTEGER	NN	Nome do produto
	desc	VARCHAR(100)		Descrições diversas sobre o produto
	ref	VARCHAR(10)	NN	Referencia do produto
	qnt_estoque	INT	NN	Quantidade deste produto que está no estoque
	valor_custo	FLOAT		Valor do custo do produto
	valor_venda	FLOAT	NN	Valor de venda do produto
	altura	FLOAT		altura do produto
	largura	FLOAT		largura do produto
	espessura	FLOAT		espessura do produto
	peso	FLOAT	NN	peso do produto

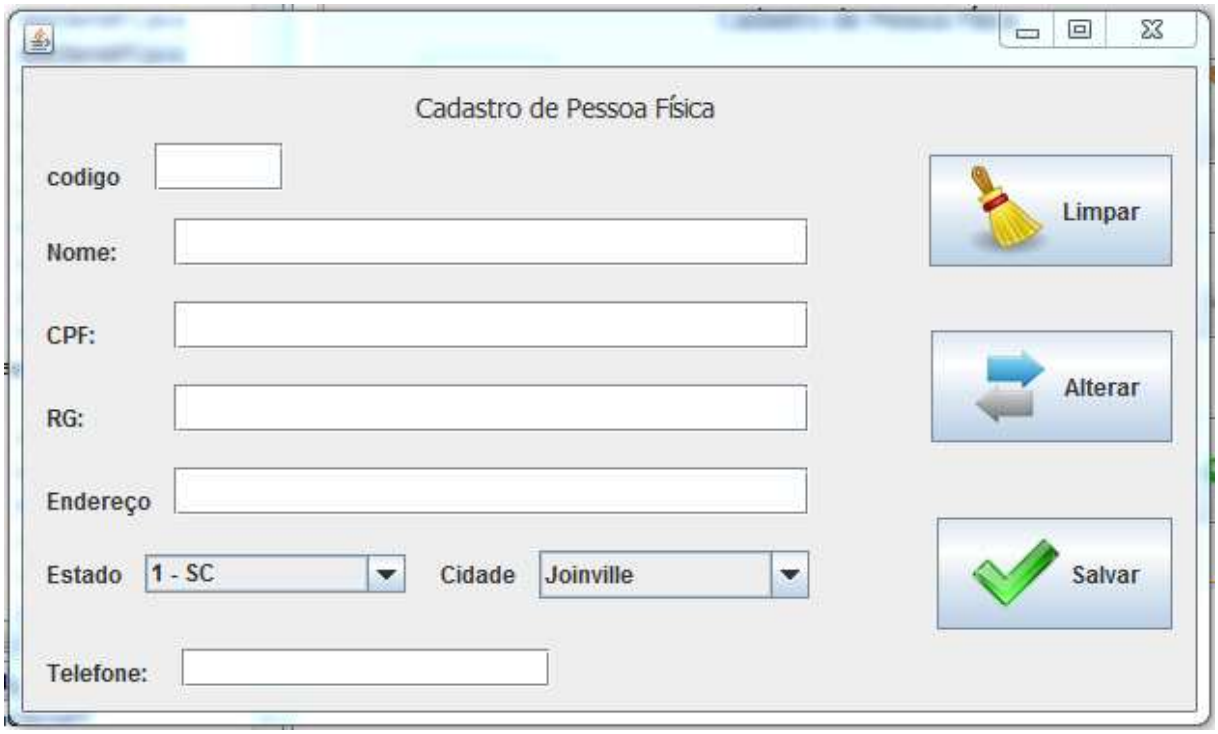
tb_veiculo: tabela que contém as informações de um veículo				
Chave	**Atributo**	**Tipo**	**Valor Padrão**	**Comentário**
PK	cod_veiculo	INTEGER	NN	Codigo do Veículo único, sequencial, inciando em 0
	marca	VARCHAR(20)	NN	Marca do veículo
	modelo	VARCHAR(20)	NN	modelo do veículo
	placa	VARCHAR(20)		placa do veículo
	renavam	VARCHAR(20)		renavam do veículo
	capacidade	INTEGER		capacidade de carga máxima do veículo
	categoria	VARCHAR(1)		Categiria que o veiculo pertence Graneleiro(g) Baú(b) Sider(s)

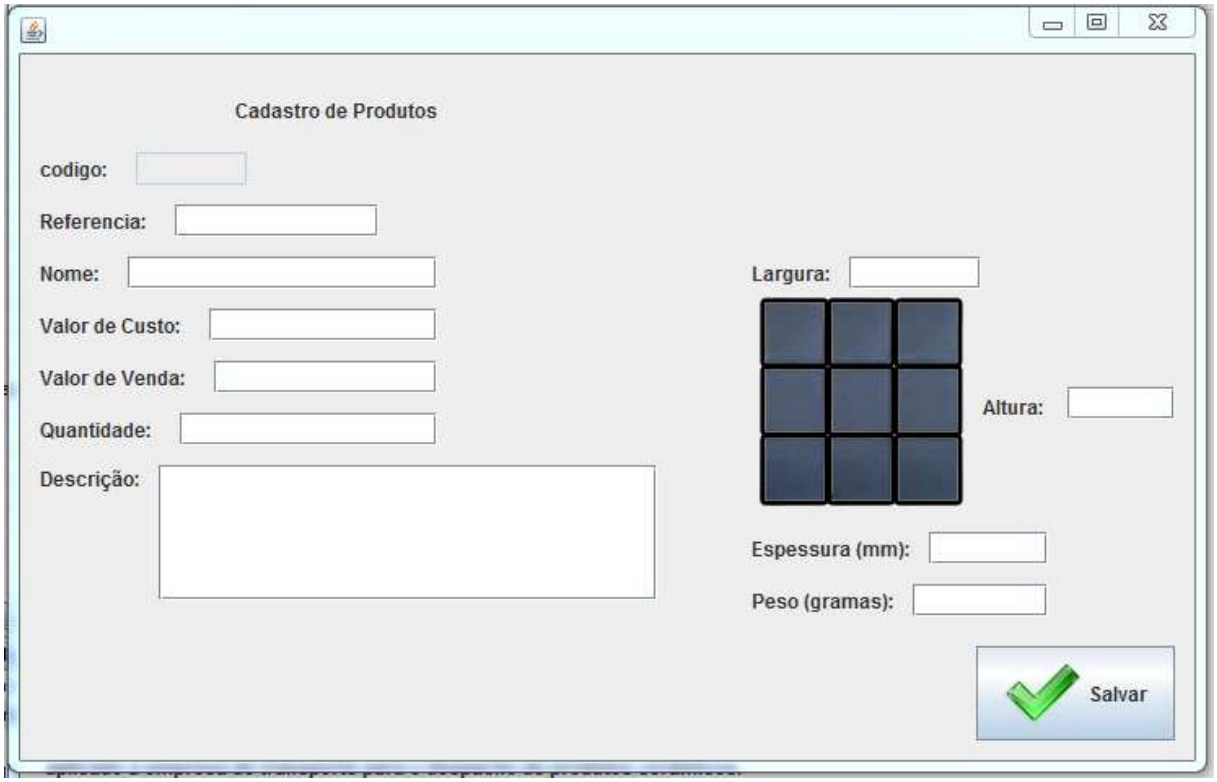
Apêndice D – Telas do Sistema

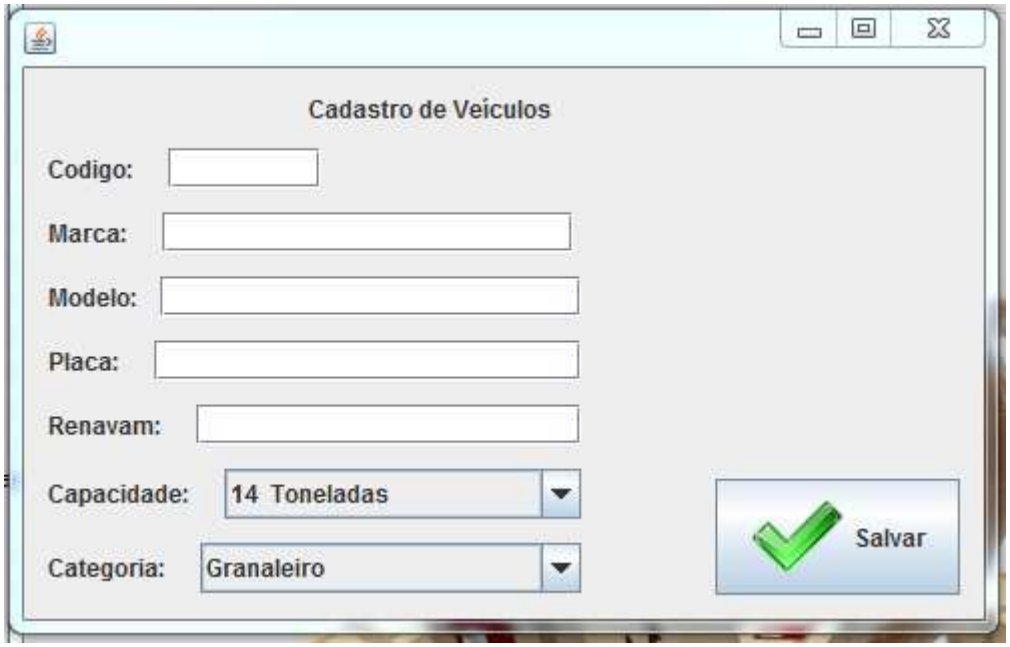





Data:
Cadastro de pedido
Codigo:
Cliente:
Cidade:
UF:
Pesquisar
Ref
Produto
Quantidade
Valor Unitário
Valor Total
Peso Total
Inserir
Calcular
Salvar


Cadastro de Pessoa Física
codigo
Nome:
CPF:
RG:
Endereço
Estado
1 - SC
Cidade
Joinville
Telefone:
Limpar
Alterar
Salvar


Cadastro de Produtos
codigo:
Referencia:
Nome:
Valor de Custo:
Valor de Venda:
Quantidade:
Descrição:
Largura:
Altura:
Espessura (mm):
Peso (gramas):
Salvar


Cadastro de Veículos
Codigo:
Marca:
Modelo:
Placa:
Renavam:
Capacidade:
14 Toneladas
Categoria:
Granaleiro
Salvar

Apêndice E – Artigo

LOGICINI: UM MODULO DE LOGÍSTICA PARA ERP UTILIZANDO A TEORIA DE GRAFOS

Guilherme B. Romancini[1], Paracelso O. Caldas[2]

1 Acadêmico do curso de Ciência da Computação – Unidade Acadêmica de Ciência, Engenharias e Tecnologias – Universidade do Extremo Sul Catarinense (UNESC) - Criciúma, SC - Brasil

2 Professor do curso de Ciência da Computação – Unidade Acadêmica de Ciência, Engenharias e Tecnologias – Universidade do Extremo Sul Catarinense (UNESC) - Criciúma, SC - Brasil

{guilhermeromancini@hotmail.com, poc@unesc.net

Resumo. *Esse trabalho visa solução para realização do despacho dos produtos fabricados em sua sede para uma gama de clientes espalhados pelas várias regiões do país, de maneira que o veiculo que transportará a mercadoria percorra a menor distância possível entre as entregas, utilizando a teoria de grafos para realizar o processamento das rotas com o uso dos algoritmos de Djikstra, Floyd e Ford.*

Palavras-Chave. *Sistemas ERP, Engenharia de Software, Teoria de Grafos, Algoritmos de menor caminho.*

1. INTRODUÇÃO

Uma empresa de comercio de materiais cerâmicos localizados no extremo Sul de Santa Catarina tem clientes em vários lugares do país, deseja-se entregar a mercadoria para seus destinatários no menor tempo possível, fazendo com que seus veículos percorram a menor distancia entre as cidades. O tamanho de cada pedido é variável, sendo que poderá existir cargas de 1 até N entregas em N cidades diferentes.

Para tanto será necessário o desenvolvimento de módulo de logística para atuar em um sistema ERP que utilizará algoritmos da teoria de grafos para resolver o problema. Dentre os algoritmo usados, deverá se analisar cada um deles, para demonstrar qual apresenta um processo mais ágil.

Deve – se existir dois processos principais, a montagem da carga para o veiculo que fará o transportes e o processamento das rotas dos pedidos. Para fazer a montagem da carga o sistema deverá analisar todos os pedidos que estão em aberto, baseando se na data do pedido, e após a seleção desta primeira cidade, realizar a procura de pedidos nas cidades mais próximas a esta. O processamento das rotas irá fazer a busca de uma carga que esteja montada, ou seja, com as cidades e os pedidos cadastrados em um mesmo veiculo, e por meio da Teoria de Grafos e seus algoritmos montará uma possível solução.

2. ERP

O objetivo do ERP é integrar todos os departamentos e funções da organização em um sistema unificado no intuito de obter as informações de forma eficaz e oportuna (SOUZA; ZWICKER, 2000).

Pode se definir que um sistema ERP é um conjunto de sistemas que compartilham a mesma base de dados, e interligam-se para troca de informações entre os setores, criando uma solução empresarial.

Os ciclos de vida de um ERP demonstram as fases da criação de um sistema, para esse projeto foi adotado o método em cascata, que visa fazer um levantamento dos requisitos do software, modelar o projeto estrutural, detalhar o projeto e após entra na fase de codificação, testes, e um ciclo entre manutenção e desenvolvimento conforme mostra a figura a seguir.

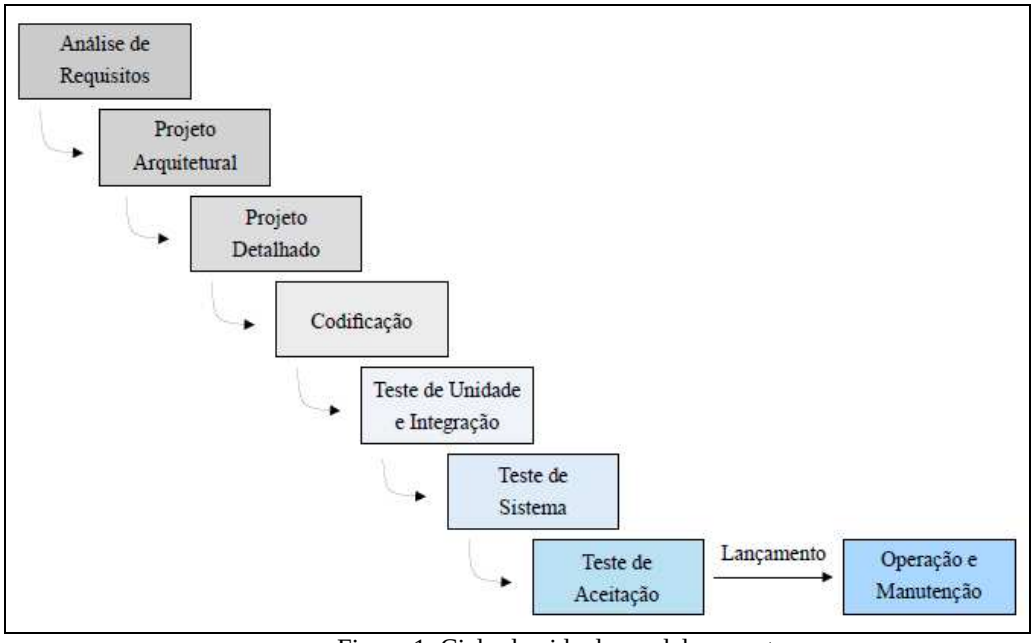


Figura 1. Ciclo de vida do modelo cascata,
Fonte: Reis, I. (2003, p. 26)

3. TEORIA DE GRAFOS

Segundo Coelho (2007) a teoria de grafos representa matematicamente um ou mais objetos na forma computacional, o grafo é definido por um conjunto de vértices e outro de arestas.

3.1 DJIKSTRA

O algoritmo de Djikstra, é um método guloso para solução do problema do menor caminho. O algoritmo percorre o grafo, primeiramente passando por aqueles que tem o menos peso, até encontrar o nó procurado. Goodrich e Tamassia (2006) pressupõe que para simplificar esse algoritmo fica necessário que o grafo de entrada não seja dirigido e simples.

3.2 FLOYD

O algoritmo de Floyd constrói uma matriz de adjacência entre todos os nós do grafo e cada elemento da matriz recebe um valor que é a distancia entre os pares de nós.

3.3 FORD

O algoritmo de Ford segundo Goodrich e Tamassia(2006), encontra caminhos mínimos em suas arestas contendo pesos negativos, o grafo deve ser dirigido pois senão, isto originará um ciclo negativo, invalidando a informação da distância do grafo.

4. SISTEMA DESENVOLVIDO

O sistema será composto de dois módulos principais: Cadastros e Logística. Os cadastros são as entradas de dados necessárias para o funcionamento do modulo de logística, o modulo da logística terá dois elementos fundamentais: montagem de rota e montagem de carga.

A montagem da carga é o recurso em que o software varre o banco de dados a procura de pedidos que estão em aberto para carregamento, dentro esses pedidos é feita uma pré-seleção por áreas de cobertura e peso máximo de carregamento. As áreas de cobertura são divididas por estados, e cada estado tem suas cidades principais a para definir uma rota padrão. O peso máximo que o veiculo pode carregar é determinado a partir de um cadastro de veículos que será usado como base para montagem da carga.

4.1 FUNCIONALIDADES DO PROTÓTIPO

Esse sistema foi dividido em dois perfis principais, usuário e logística. O perfil usuário tem acesso as funcionalidades essenciais para que o módulo de logística funcione de forma integra. Abaixo os diagramas de casos de uso para os perfis:

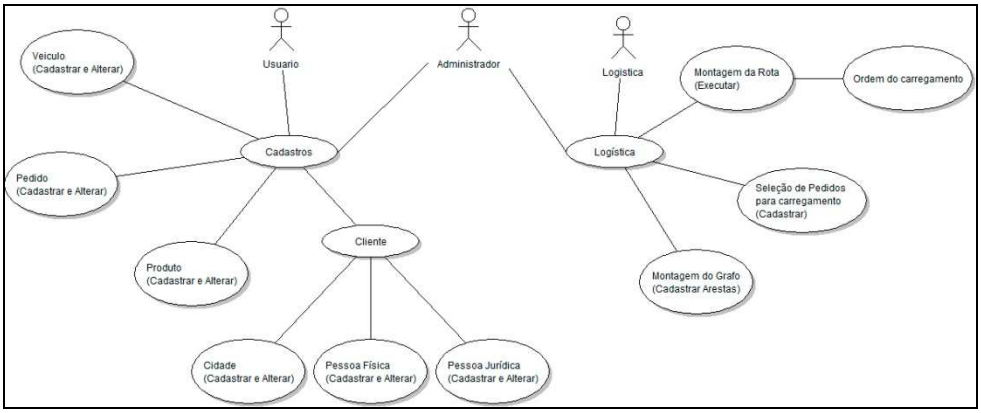


Figura 2. Diagrama de Casos de Uso

4.2 BANCO DE DADOS E LINGUAGEM DE PROGRAMAÇÃO

O banco de dados utilizados para implementação deste protótipo de modulo de logística foi o MySql versão 5.5.8 por apresentar bom desempenho e integração com a linguagem de programação utilizada JAVA, utilizando a IDE NetBeans versão 6.9.

4.3 CLASSES E ARMAZENAMENTO

Com o diagrama de classes é possível visualizar de forma genérica a estrutura usada para o armazenamento persistente no desenvolvimento da aplicação, e como se dará o relacionamento entre as classes.

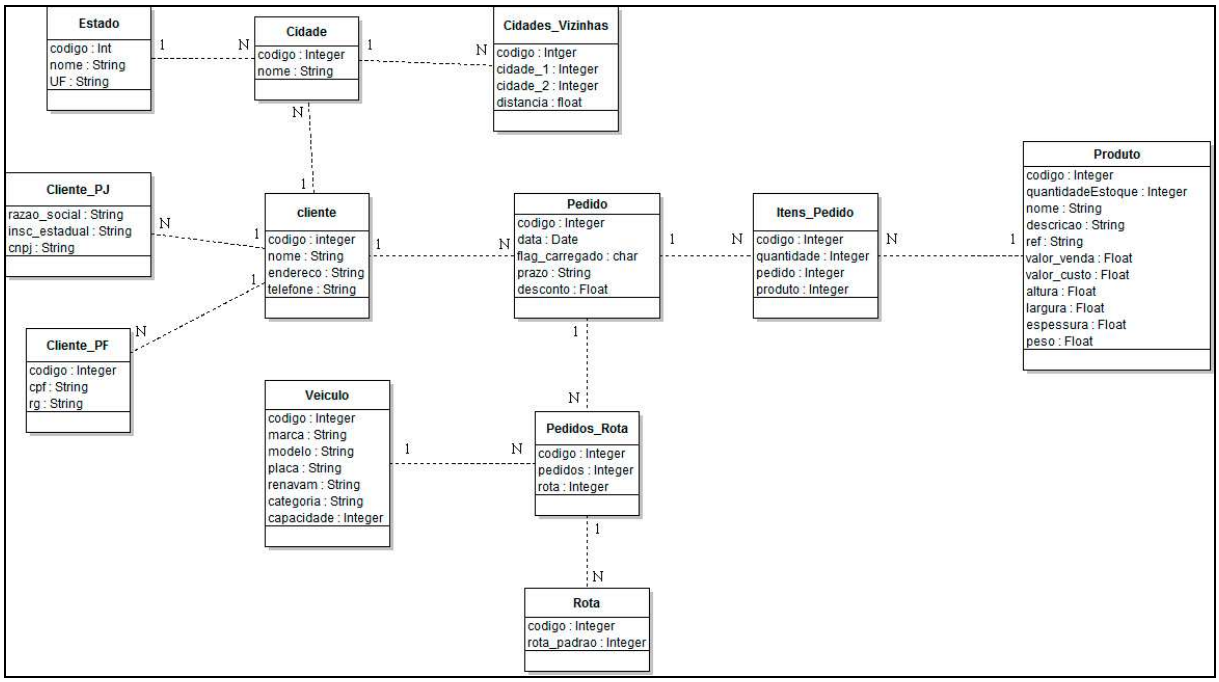


Figura 3. Diagrama de Classes

4.4 INTERFACE

A tela de Seleção de pedidos faz a montagem da carga, o usuário escolhe o peso do veiculo que realizará o pedido e a cidade principal. A função procurar os pedidos por cidade, na cidade principal o volume de entregas tradicionalmente é maior, e as próximas cidades serão suas vizinhas. A figura 4 mostra a tela de Seleção de pedidos:

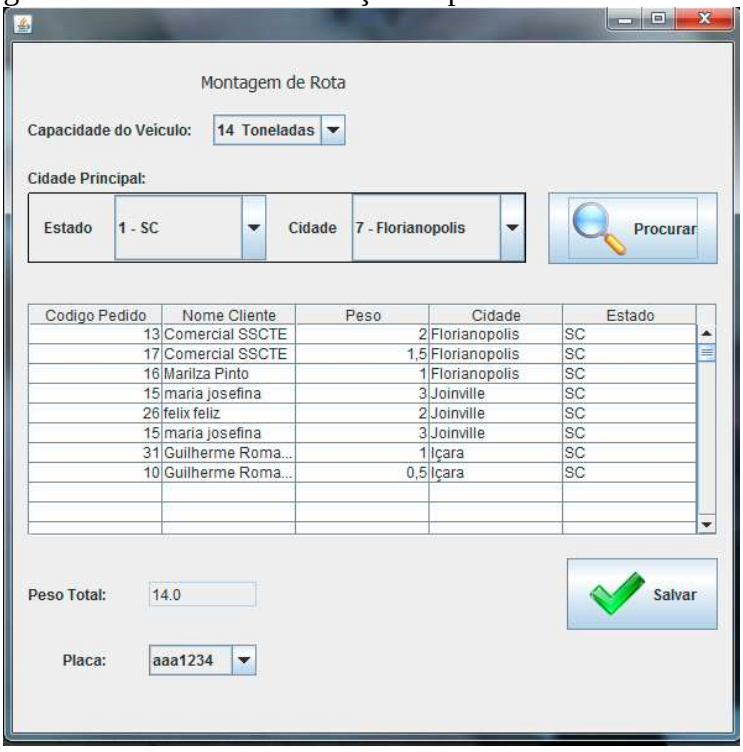


Figura 4, Tela de Seleção de Pedidos

Os pedidos selecionados em uma carga são armazenados no banco de dados e na tela de montagem da rota estes pedidos são ordenados por entregas. Podendo ser processados por meio dos algoritmos de Bellman-Ford, Floyd-Warshall, e Dijkstra.

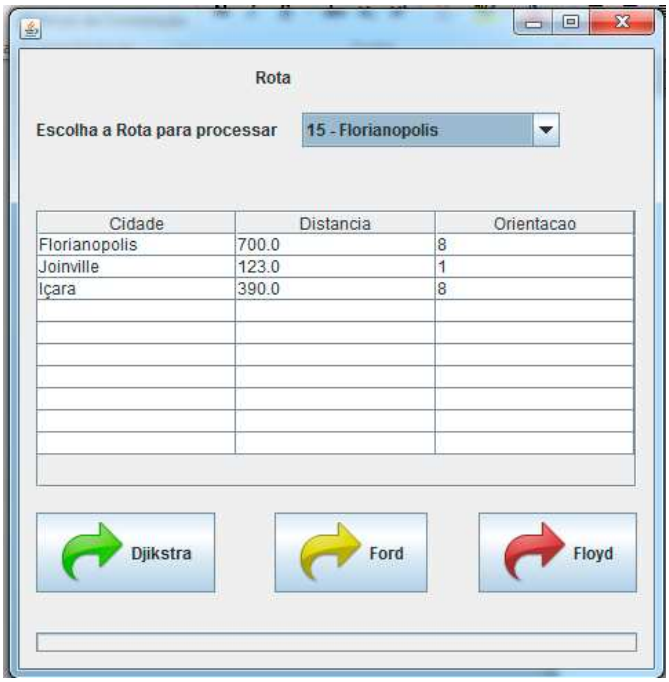


Figura 5. Tela do Processamento da Rota

5. PROCESSAMENTO E ANÁLISE COM GRAFOS

O processamento da rota pode ocorrer utilizando os três tipos de algoritmos de menor caminho da teoria de grafos: Dijkstra, Floyd-Warshall e Belmann-Ford. Todos os três algoritmos cumpriram com eficácia o roteamento das cidades.
os algoritmos foram implementados em JAVA, e para cada qual teve um custo de processamento diferente. A configuração do computador que realizou o teste é:

a) Processador Intel Core 2 duo 7200 2.53 Ghz;
b) Memoria 4GB 800mhz em dual channel;
c) HD 500 Gb 7200 RPM;
d) Sistema operacional Windows 7 64 bits;
e) Java 1.6.0_25;
f) Banco de dados MYSQL 5.5

No primeiro teste foi realizada uma busca com dezessete pedidos distribuídos entre sete cidades em um grafo com vinte e três cidades. Cada algoritmo foi processado dez vezes excluindo o maior e o menor resultado de cada processamento com base nos dados:

a) Cidades mapeadas: 23
b) Cidades buscadas: 7
c) Pedidos buscados: 17

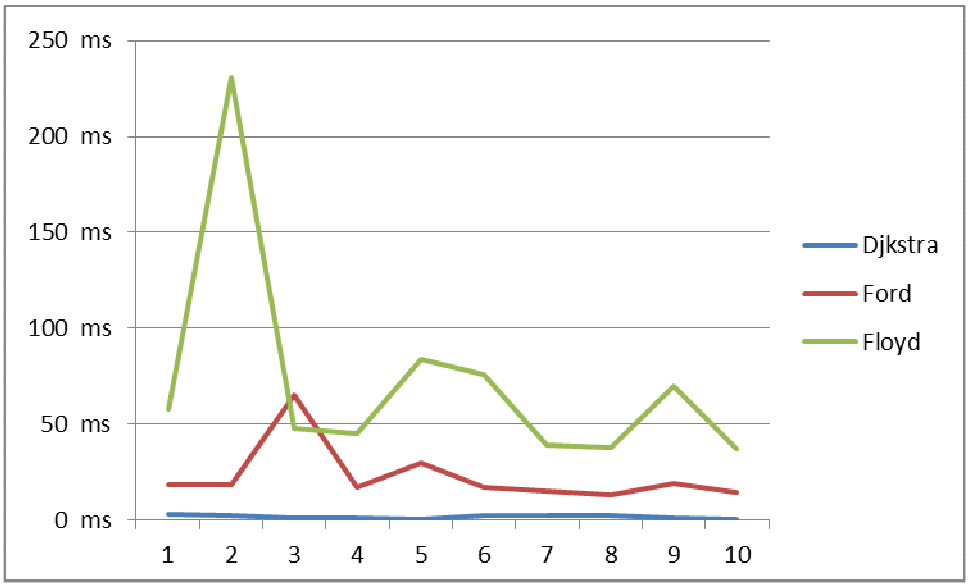


Figura 6. Comparativo dos algoritmos

O algoritmo de Djkstra foi constante em todo o tempo de análise variando em no máximo 1 milissegundo. Ford foi constante a partir da 4ª iteração com variações de 13 à 19 milissegundos. O algoritmo de Floyd teve altos e baixos durante todo o ciclo de análise.

Após analise deste teste multiplicando por 10 o numero de cidades buscadas e o numero de pedidos encontrados, temo o seguinte gráfico:

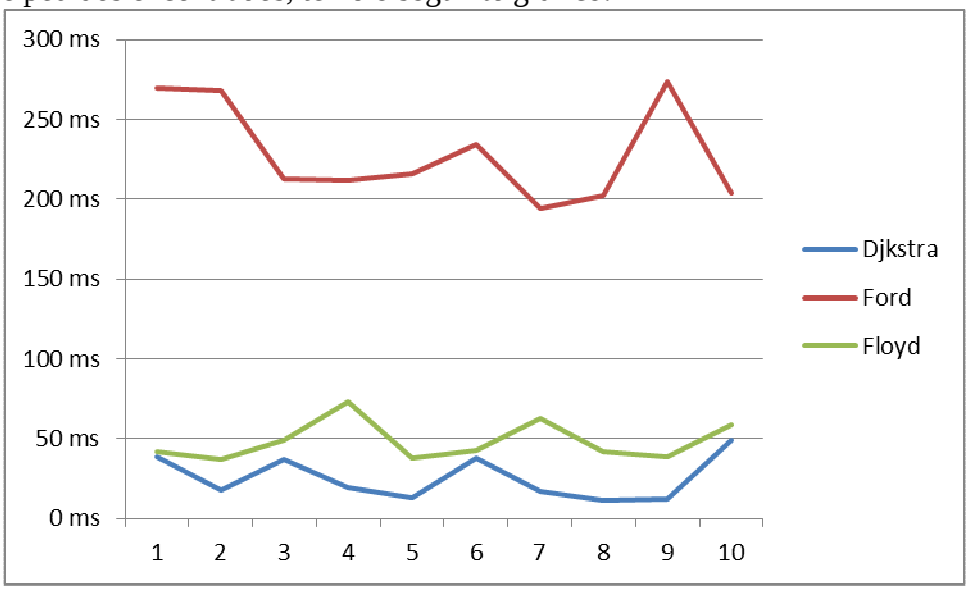


Figura 7. Comparativo dos algoritmos

Para esse caso, o algoritmo de Dijkstra ainda se mostrou mais eficiente que os demais, porém podemos perceber que em uma aplicação que exija um maior roteamento do que números de pontos no grafo, o algoritmo de Floyd consegue maior agilidade, dependendo do grau de complexidade do problema.

6. CONCLUSÃO

Esse artigo apresentou o protótipo Logicini, seu desenvolvimento, e o comparativo entre os algoritmos de menor caminho utilizados no software.
Para escolher um entre esses algoritmos, deve se primeiro analisar o problema base, para depois optar por uma destas implementações, porém todos funcionaram corretamente e apresentaram a solução desejada.
Tendo em vista um numero pequeno de nós em um grafo a melhor solução é o algoritmo de Djikstra, porém se necessitar um grande numero de buscas em um grafo a medida do tempo o algoritmo de Floyd vai se tornando mais eficiente.

REFERÊNCIAS

SOUZA, C. e ZWICKER, R. (2000) Sistemas Integrados de Gestão Empresarial: Estudos de casos de implementação de Sistemas ERP. USP.
REIS, C. (2003) **Caracterização de um Modelo de Processo para Projetos de Software Livre.** São Paulo.
COELHO, F. (2007) **Computação científica com python.** Petrópolis.
GOODRICH, M; TAMASSIA, R. (2006) **Estrutura de dados e algoritmos em java.** Artmed.