

**LIGA DE ENSINO DO RIO GRANDE DO NORTE
FACULDADE NATALENSE PARA O DESENVOLVIMENTO DO RN
BACHARELADO EM SISTEMAS DE INFORMAÇÃO**

**DESENVOLVIMENTO DE UMA FERRAMENTA COM ARQUITETURA ABERTA
PARA O APOIO AO ESTUDO DA TEORIA DOS GRAFOS**

Patrício Santos Carneiro

Orientador:

Antônio de Pádua de M. Henriques

**NATAL/RN
2003**

LIGA DE ENSINO DO RIO GRANDE DO NORTE
FACULDADE DE ENSINO DE GRADUAÇÃO
PATRICIO SANTOS CARNEIRO
BACHARELADO EM SISTEMAS DE INFORMAÇÃO

DESENVOLVIMENTO DE UMA FERRAMENTA COM ARQUITETURA ABERTA
PARA O APOIO AO ESTUDO DA TEORIA DOS GRAFOS



Patricia Santos Carneiro

DESENVOLVIMENTO DE UMA FERRAMENTA COM ARQUITETURA ABERTA PARA O APOIO AO ESTUDO DA TEORIA DOS GRAFOS

Monografia apresentada e aprovada em ____/____/____
Pela banca examinadora composta pelos seguintes membros:

Projeto de Monografia apresentado ao
curso de Bacharelado em Sistemas de
Informação da Faculdade Natalense
para o Desenvolvimento do Rio Grande
do Norte como requisito parcial para a
obtenção do título em Bacharel em
Sistemas de Informação

Orientador:

Antônio de Pádua de M. Henriques

**NATAL/RN
2003**



m
004.78:37
C29d
2003
ex 01

Carneiro, Patricio Santos

Desenvolvimento de uma ferramenta com arquitetura aberta para o apoio ao estudo da teoria dos grafos. Patricio Santos Carneiro. ____ Natal: FARN, 2003 49 p.

Monografia (Bacharelado em Sistemas de Informação – Faculdade Natalense para o Desenvolvimento do Rio Grande do Norte)

Orientador: Antônio de Pádua de M. Henriques

1. Sistema de Computação – Monografia
2. Software Educativo – Monografia
3. Teoria dos Grafos I. Desenvolvimento de uma ferramenta com arquitetura aberta para o apoio ao estudo da teoria dos grafos.

RN/FARN/BC

CDU 004.78:37

**LIGA DE ENSINO DO RIO GRANDE DO NORTE
FACULDADE NATALENSE PARA O DESENVOLVIMENTO DO RN
BACHARELADO EM SISTEMAS DE INFORMAÇÃO**

**DESENVOLVIMENTO DE UMA FERRAMENTA COM ARQUITETURA ABERTA
PARA O APOIO AO ESTUDO DA TEORIA DOS GRAFOS**

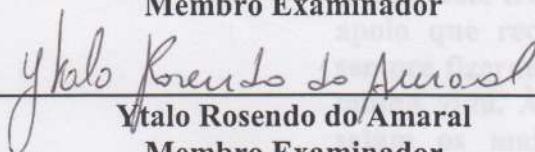
Patrício Santos Carneiro

Monografia apresentada e aprovada em ____ / ____ / ____,
Pela banca examinadora composta pelos seguintes membros.



Antônio de Pádua de M. Henriques
Orientador

Robson Santana Pacheco
Membro Examinador



Ytalo Rosendo do Amaral
Membro Examinador

AGRADECIMENTOS

Primeiramente agradeço a Deus por ter me capacitado a desenvolver este projeto, que por sua vez, foi elaborado com o auxílio e esforço de várias pessoas e instituições, não cabendo aqui citar a todas, dentre estas pessoas foi de caráter indispensável a contribuição dada por Anderson Morby pela ajuda e crítica no tratamento das imagens.

Meus sinceros agradecimentos para todos os professores que fizeram parte de toda a minha vida acadêmica, desde meu primeiro dia de vida escolar até os dias de hoje.

Realizo meus agradecimentos ao Professor Robson Santana Pacheco que sempre incentivou e acompanhou de perto o desenvolvimento de todos os meus projetos nunca se negando a ajudar quando solicitado.

Ao Provedor de Internet ACT, NatalSoft e FARN pela disponibilidade de equipamento e das ferramentas de apoio para a elaboração e conclusão desta investigação.

Por fim, aos meus professores revisores e orientadores, em especial o Professor Antônio de Pádua de M. Henriques pela sua orientação, dedicação e suporte prestado durante a realização e conclusão deste projeto.

Afinal venho agradecer a todos que de uma maneira ou de outra, contribuíram para a realização deste projeto.

Dedico este trabalho aos meus pais, pelo apoio que recebi e pelos esforços que sempre fizeram por mim durante toda a minha vida. À todos os meus parentes, sejam os mais próximos ou os mais distantes, que, de alguma forma, contribuíram para a sua realização.

AGRADECIMENTOS

Primeiramente agradeço a Deus por ter me capacitado a desenvolver este projeto, que por suas vez, foi elaborado com o auxílio e esforço de várias pessoas e instituições, não cabendo aqui citar a todas, dentre estas pessoas foi de caráter indispensável a contribuição dada por Anderson Morhy pela ajuda e critica no tratamento das imagens.

Meus sinceros agradecimentos para todos os professores que fizeram parte de toda a minha vida acadêmica, desde meu primeiro dia de vida escolar até os dias de hoje.

Enfatizo meus agradecimentos ao Professor Robson Santana Pacheco que sempre incentivou e acompanhou de perto o desenvolvimento de todos os meus projetos nunca se negando a ajudar quando solicitado.

Ao Provedor de Internet ACT, NatalSoft e FARN pela disponibilidade de equipamento e das ferramentas de apoio para a elaboração e conclusão desta investigação.

Por fim, aos meus professores revisores e orientadores, em especial o Professor Antônio de Pádua de M. Henriques pela sua orientação, dedicação e suporte prestado durante a realização e conclusão deste projeto.

Assim venho agradecer a todos que de uma maneira ou de outra, contribuíram para a realização deste projeto.

Aprendizagem	25
1.1- GRAFOS	26
1.1.1 Grafo direcionado	26
1.1.2 Grau de um vértice	27
1.1.3 Árvores	28
1.1.4 Adjacência	28
1.1.5 Percursos, Ciclos e Circuitos	29
1.1.6 Matriz de Adjacência	29
1.1.7 Matriz de Incidência	30
1.1.8 Alguns problemas clássicos envolvendo grafos	31
1.2 APLICAÇÃO	34
1.2.1 Arquitetura da aplicação	34
1.2.2 Formato do arquivo gerado pelo FARN Graph	40
1.2.3 Funcionamento da XMI	40
1.2.4 Características de XML	40
1.2.5 Exemplo do armazenamento de um grafo em XML	41
1.2.6 Qual a vantagem de usar o XML?	42
1.2.7 Recursos implementados	44
2 METODOLOGIA	45
3 ANÁLISE DE RESULTADOS	47
4 CONCLUSÃO	48
5 REFERÊNCIAS BIBLIOGRÁFICAS	49

SUMÁRIO

1	INTRODUÇÃO.....	11
1.1	PROBLEMA DE PESQUISA	12
1.2	OBJETIVOS	12
1.3	JUSTIFICATIVA	13
2	REFERÊNCIAL TEÓRICO	14
2.1	PROGRAMAÇÃO ORIENTADA AO OBJETO	14
2.2	OBJECT PASCAL	15
2.3	O QUE É DELPHI.....	15
2.3.1	Objetos.....	16
2.3.2	Eventos	16
2.3.3	Forms	17
2.3.4	Object Inspector	17
2.4	SOFTWARE EDUCATIVO	23
2.4.1	Quanto aos princípios da aprendizagem e os paradigmas educacionais.....	23
2.4.2	Quanto ao uso da tecnologia aplicada ao processo educacional.....	24
2.4.3	Ensino X Aprendizagem.....	25
2.5	GRAFOS.....	26
2.5.1	Grafo direcionado	26
2.5.2	Grau de um vértice.....	27
2.5.3	Árvores	28
2.5.4	Adjacência	28
2.5.5	Percursos, Ciclos e Circuitos	29
2.5.6	Matriz de Adjacência	29
2.5.7	Matriz de Incidência	30
2.5.8	Alguns problemas clássicos envolvendo grafos	31
2.6	APLICAÇÃO	34
2.6.1	Arquitetura da aplicação	34
2.6.2	Formato do arquivo gerado pelo FARN Graph.	40
2.6.3	Funcionamento da XML.....	40
2.6.4	Características de XML	40
2.6.5	Exemplo do armazenamento de um grafo em XML.....	41
2.6.6	Qual a vantagem de usar o XML?	42
2.6.7	Recursos implementados	44
3	METODOLOGIA.....	45
4	ANÁLISE DE RESULTADOS.....	47
5	CONCLUSÃO.....	48
6	REFERÊNCIAS BIBLIOGRÁFICAS.....	49

LISTA DE FIGURAS

FIGURA 1: Formulário e componentes	17
FIGURA 2: Propriedades do Object Inspector	18
FIGURA 3: Eventos do Object Inspector	19
FIGURA 4: Barra de Componentes	20
FIGURA 5: Editor de Código fonte	20
FIGURA 6: Barra de acesso rápido	21
FIGURA 7: Gerenciador de Projetos	21
FIGURA 8: Barra de menu	22
FIGURA 9: Help On-Line – Padrão do Windows	22
FIGURA 10: Tipos e modalidade de softwares educativos x Paradigma Educacional	25
FIGURA 11: Representação de alguns grafos	26
FIGURA 12: Representação de grafos conectados e desconectados	27
FIGURA 13: Grau de um vértice em um Grafo G	27
FIGURA 14: Árvores com quatro vértices de Ordem 4	28
FIGURA 15: Demonstração de Percursos Simples e Elementar de um grafo G	29
FIGURA 16: Representação da Matriz de Adjacência de um Grafo G	30
FIGURA 17: Representação da Matriz de Incidência de um Grafo G	30
FIGURA 18: Exemplo de um grafo Euleriano	31
FIGURA 19: Desenho da criança - Grafo Euleriano	31
FIGURA 20: Grafo gerado pelo problema das pontes de Königsberg	32
FIGURA 21: Exemplos de grafos Hamiltonianos	33
FIGURA 22: Visualização das camadas de exibição	35
FIGURA 23: Visualização final	36
FIGURA 24: Grafo de Exemplo	41

LISTA DE TABELAS

TABELA 1: Relação entre os paradigmas educacionais, suas características e.....	25
TABELA 2: Opções de interação da aplicação	37
TABELA 3: Opções de Configuração	38
TABELA 4: Classe TDadosForma	38
TABELA 5: Propriedades da classe TDadosForma	38
TABELA 6: Classe TDadosAresta	39
TABELA 7: Propriedades da classe TDadosAresta	39
TABELA 8: Visualizando o XML em um Web Browser	41
TABELA 9: Código em XML	42

RESUMO

Neste projeto, apresenta-se uma ferramenta base extensível que se propõe a auxiliar o estudo da teoria dos grafos. Todo o processo de análise e implementação desta ferramenta visa à possibilidade de um aprimoramento futuro, utilizando padrões como a Programação Orientada a Objetos e formato padronizado de armazenamento (XML - eXtensible Markup Language). Desta forma, garantimos uma arquitetura aberta, possibilitando o desenvolvimento distribuído por equipes de diferentes níveis de conhecimento e maturidade na área de grafos e da ciência da computação. A principal meta deste projeto é oferecer subsídios para a melhoria da qualidade da aprendizagem, garantindo a efetiva utilização do computador no ensino da teoria dos grafos.

In this project, a tool is presented extensible base that if considers to assist it the study of the theory of the graphs. All the process of analysis and implementation of this tool aims at to the possibility of a future improvement, using standards as the Object-Oriented Programming and standardized format of storage (XML - Extensible markup language). Of this form, we guarantee an opened architecture, making possible the development distributed for teams of different levels of knowledge and maturity in the area of graphs and the computer science. The main goal of this project is to offer to subsidies for the improvement of the quality of the learning, guaranteeing the effective use of the computer in the education of the theory of the graphs.

descreve o caso específico) de um certo tipo em um grafo. A dificuldade de todos os problemas de Grafos está em desenvolver algoritmos eficientes que encontrem a configuração ótima desejada.

Segundo GOLDBERG e LUNA (2009) o conceito de grafos é bastante simples e intuitivo, onde se pode considerar que um Grafo, nada mais é que uma representação gráfica de interdependência entre elementos que são representados por nós, simbolicamente unidos através de um traço denominado aresta.

Qualquer um que tentar desenvolver um software de grafos, inicialmente enfrentará a dificuldade de exibi-los graficamente, dado a sua complexidade computacional, e perante a isso, serão implementados os padrões utilizados na Teoria dos Grafos para que seja possível a visualização quase que total das abstrações de problemas distintos e de diferentes graus de dificuldade.

A proposta desse projeto consiste na implementação de uma ferramenta base de software, com o propósito de auxiliar o aluno a compreender visualmente a abstração de um problema e resolvê-lo com facilidade através de uma interface prática e simples. Foram implementados nessa ferramenta, algoritmos eficientes de alta relevância para o estudo da Teoria dos Grafos em sala de aula.

1 INTRODUÇÃO

A área de *Teoria dos Grafos* trata de abstrações de certos problemas práticos encontrados em uma infinidade de ramos totalmente distintos tais como projetos de chips, administração de frotas de veículos, sistemas operacionais de computadores, sistemas de posicionamento geográfico, etc, como também em outras áreas da matemática e da pesquisa operacional. Muitos destes problemas podem ser descritos sobre um grafo.

Os problemas consistem em encontrar uma configuração ótima (máxima ou mínima, conforme o caso específico) de um certo tipo em um grafo. A dificuldade de todos os problemas de Grafos está em desenvolver algoritmos eficientes que encontrem a configuração ótima desejada.

Segundo GOLDBARG e LUNA (2000) o conceito de grafos é bastante simples e intuitivo, onde se pode considerar que um Grafo, nada mais é que uma representação gráfica de interdependência entre elementos que são representados por nós, simbolicamente unidos através de um traço denominados aresta.

Qualquer um que tentar desenvolver um software de grafos, inicialmente enfrentará a dificuldade de exibi-los graficamente, dado a sua complexidade computacional, e perante a isso, serão implementados os padrões utilizados na *Teoria dos Grafos* para que seja possível a visualização quase que total das abstrações de problemas distintos e de diferentes graus de dificuldade.

A proposta desse projeto consistiu na implementação de uma ferramenta base de software, com o propósito de auxiliar o aluno a compreender visualmente a abstração de um problema e resolvê-lo com facilidade através de uma interface prática e simples. Foram implementados nessa ferramenta, algoritmos eficientes de alta relevância para o estudo da *Teoria dos Grafos* em sala de aula.

1.1 PROBLEMA DE PESQUISA

Concretamente, uma das principais dificuldades da compreensão de problemas que envolvem grafos é dada pela abstração específica de cada problema. A ciência da computação é de fato o que proporciona uma robustez capaz de absorver o rápido desenvolvimento da tecnologia de forma mais eficiente possível, resolvendo problemas de grande complexidade computacional em um tempo cada vez menor. Portanto, levanta-se o seguinte problema: O que poderia ser implementado para facilitar o entendimento do aluno ao estudar a *Teoria dos Grafos*?

1.2 OBJETIVOS

O objetivo deste projeto é contribuir ao estudo da *Teoria dos Grafos*. Foi aplicada a ciência da computação através de investigações de uma grande coleção de problemas envolvendo grafos.

- Investigação de problemas estruturais e algoritmos em grafos.
- Identificação da linguagem de implementação, uma vez que sua complexidade computacional é alta.
- Modelar toda a estrutura, analisando de forma minuciosa, para que essa estrutura chegue a suprir todos os requisitos do escopo da ferramenta em questão.
- Desenvolver uma interface simples, confortável e de fácil entendimento ao usuário, tanto para a entrada dos dados como também para os resultados exibidos.
- Identificar e implementar os algoritmos de alta relevância para o estudo da disciplina Teoria dos Grafos.
- Implementar uma arquitetura aberta para possibilitar a incorporação de novos recursos desenvolvidos pelo próprio aluno.

Após todos os seus resultados estiverem precisos e confiáveis esta ferramenta atingirá seu objetivo geral.

1.3 JUSTIFICATIVA

Justifica-se este projeto pela reconhecida importância da *Teoria dos Grafos* na teoria da computação. Problemas desse caráter ocorrem freqüentemente em várias áreas da matemática, e são essencialmente fundamentais em todas as áreas da teoria da computação.

A motivação que gerou esta proposta reside na necessidade de facilitar a compreensão, observação e visualização da abstração de um problema envolvendo grafos, proporcionando um melhor aproveitamento da turma, uma vez que o software seja utilizado em uma sala de aula.

A meta deste projeto é de disseminar um pouco mais o conhecimento sobre a *Teoria dos Grafos*, através de uma cooperação entre seus investigadores e a instituição patrocinadora, formalizando em conjunto uma pesquisa de excelência.

A primeira linguagem de programação a implementar os conceitos de OOP foi a linguagem SIMULA-68; em seguida surgiu a linguagem Smalltalk - desenvolvida pela Xerox, que podendo ser considerada a linguagem incentivadora onde popularizou o emprego da OOP por ser uma linguagem de pura linguagem Orientada ao Objeto(OO), onde implementa todos os conceitos de OO, o que não ocorre com as linguagens OO híbridas onde implementam apenas alguns conceitos de orientação ao objeto.

O emprego da OOP foi uma saída protagonizada pelos desenvolvedores para minimizar os custos dos sistemas, com ênfase aos custos relativos às manutenções corretivas.

Como princípio, a OOP utiliza a mesma concepção da engenharia de hardware que projeta novos equipamentos usando os mesmos componentes básicos como transistores, resistores, fusíveis, diodos, chips etc. Os "objetos" já existentes são utilizados para produzir outros novos "objetos", tornando essa metodologia mais poderosa que as metodologias tradicionais de desenvolvimento de sistemas.

Fundamentalmente o que se deseja com esta "nova" metodologia de desenvolvimento de sistemas são basicamente duas características: reutilização de código e modularidade de escrita; na qual a ideia de OOP é imbatível quando comparada com as metodologias antigas, pelo fato da OOP empregar um conceito mais real, mais concreto, que é o de Objeto.

2 REFERENCIAL TEÓRICO

2.1 PROGRAMAÇÃO ORIENTADA AO OBJETO

Formalmente, Programação Orientada a Objetos é a programação através da qual os programas recorrem a áreas ou partições de memória onde encontram objetos (dados + procedimentos) representativos de cada realidade.

Podemos considerar a Programação Orientada ao Objeto (OOP - Object-Oriented Programming) como uma extensão quase natural da Programação Modular nos últimos anos. Na realidade, não podemos fazer tal afirmação pelo fato da OOP ter sido concebida há muito tempo atrás (no início da década de 70). Sua origem vem da linguagem Simula (Simula Language), criada na Noruega no início da década de 60, como seu nome indica, foi criada para fazer simulações; entretanto, seu uso alavancou o conceito da similaridade com o mundo real.

A primeira linguagem de programação a implementar os conceitos de OOP foi a linguagem SIMULA-68; em seguida surgiu a linguagem Smalltalk -desenvolvida pela Xerox, que podendo ser considerada a linguagem incentivadora onde popularizou o emprego da OOP por ser uma linguagem de pura linhagem Orientada ao Objeto(OO), poderosíssima, que implementa todos os conceitos de OO, o que não ocorre com as linguagens OO híbridas onde implementam apenas alguns conceitos de orientação ao objeto.

O emprego da OOP foi uma saída protagonizada pelos desenvolvedores para minimizar os custos dos sistemas, com ênfase aos custos relativos às manutenções corretivas.

Como princípio, a OOP utiliza a mesma concepção da engenharia de hardware que projeta novos equipamentos usando os mesmos componentes básicos como transistores, resistores, fusíveis, diodos, chips, etc. Os "objetos" já existentes são utilizados para produzir outros novos "objetos", tornando essa metodologia mais poderosa que as metodologias tradicionais de desenvolvimento de sistemas.

Fundamentalmente o que se deseja com esta "nova" metodologia de desenvolvimento de sistemas são basicamente duas características: reutilização de código e modularidade de escrita; na qual a idéia de OOP é imbatível quando comparada com as metodologias antigas, pelo fato da OOP empregar um conceito mais real, mais concreto, que é o de Objeto.

Segundo PRICE(1997) Apud Leite & Rahal Júnior, um objeto em particular é como um processador com memória própria e independente de outros objetos. Em termos de implementação, objeto é um bloco de dados privados envolvidos por código, de maneira que o acesso a ele só pode ser feito sob condições especiais. Todo o comportamento desse "ente" encapsulado é descrito através de rotinas que manipulam seus dados, sendo que o seu estado corrente está em seus próprios dados; em outras palavras, cada objeto tem suas próprias características, moldadas a partir de uma matriz.

2.2 OBJECT PASCAL

Neste projeto o software em questão será implementado na linguagem ObjectPascal no qual possui todo o poder da OOP e utilizado como ferramenta de desenvolvimento o ambiente DELPHI no qual possui alguns tópicos básicos de Orientação a Objetos.

2.3 O QUE É DELPHI

O Delphi é um ambiente de desenvolvimento de aplicações, orientado a objeto, criada pela Borland (atualmente chamada de Inprise) em 1995 que permite o desenvolvimento de poderosas aplicações visuais onde incorpora modernos conceitos de programação visual combinados aos mais poderosos recursos da linguagem Object Pascal baseadas em diversos sistemas operacionais tais como o MS Windows e Linux. Pode-se utilizar o conjunto de componentes e código gerado para transformar seus protótipos em aplicações robustas que satisfaçam nossas necessidades. O Delphi também oferece ferramentas de bancos de dados que permite o desenvolvimento de aplicações Client/Server e relatórios. As ferramentas de bancos de dados permitem que você visualize seus dados dinamicamente durante o desenvolvimento possibilitando a visualização rápida do resultados.

Esta ferramenta possui uma arquitetura baseada em componentes onde asseguram que as aplicações Delphi sejam robustas, reutilizáveis e de fácil manutenção. O ambiente consiste de vários elementos, ferramentas de design e de banco de dados para auxiliá-lo a desenvolver e testar aplicações de forma rápida e intuitiva, isso é possível por possuir um construtor visual de interface com o usuário no qual permite criar visualmente aplicações através da seleção de componentes na paleta.

A facilidade em alternar entre formulários e seu código permite aos desenvolvedores trabalhar tanto na edição de texto como no modo de design visual através de uma sincronização do código fonte com a representação visual. Outra característica dessa ferramenta é que tem uma arquitetura aberta onde permite adicionar componentes e ferramentas personalizadas e de terceiros.

2.3.1 Objetos

Um objeto pode ser rapidamente conceituado como uma unidade dinâmica, composta por um estado interno privativo (estrutura de dados) e um comportamento (conjunto de operações), ou seja, um "ente" ativo dotado de certas características que o tornam "inteligente", a ponto de tomar certas decisões quando devidamente solicitado.

Objetos são os controles visuais do Windows, como caixas de mensagem, menu de opções, imagens, entre outros. O Delphi fornece uma grande variedade de objetos prontos que você pode usar na criação do seu programa.

2.3.2 Eventos

Os programas feitos em Delphi são orientados a eventos. Um evento nada mais é do que uma ação que foi gerada pelo usuário. Clicar com o mouse ou pressionar uma tecla são exemplos de eventos. Segue abaixo alguns dos eventos mais usados no Delphi:

OnClick: Ocorre quando o usuário dá um clique sobre o objeto

OnDblClick: Ocorre quando o usuário dá um duplo clique sobre o objeto

OnKeyDown: Ocorre quando o usuário pressiona uma tecla enquanto o objeto está selecionado.

OnKeyUp: Ocorre quando o usuário solta uma tecla enquanto o objeto está selecionado.

OnKeyPress: Ocorre quando o usuário pressiona uma tecla.

OnMouseDown: Ocorre quando o usuário pressiona um botão do mouse.

OnMouseUp: Ocorre quando o usuário solta o botão do mouse.

OnMouseMove: Ocorre quando o usuário move o ponteiro do mouse.

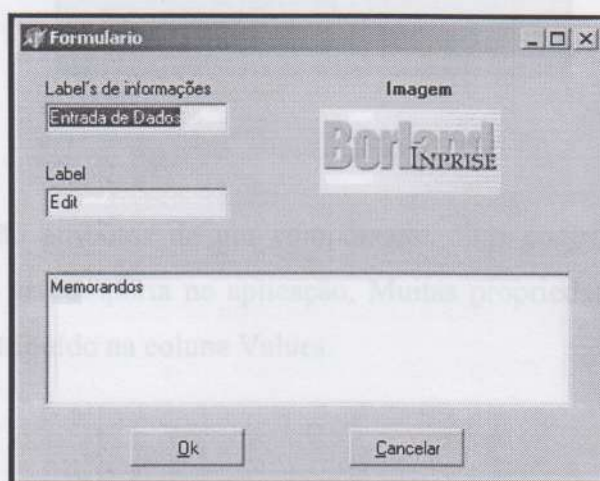
O ambiente é composto por **Integrated Development Environment (IDE)** que consiste de diversos elementos ou ferramentas visuais. Uma vez acostumados a estes elementos, pode-se começar a desenvolver aplicações. Tais elementos são:

- Form
- Component Palette
- Object Inspector
- Code Editor
- SpeedBar
- ProjectManager
- SpeedMenus
- On-Line Help

2.3.3 Forms

São a característica central das aplicações onde o desenvolvedor vai adicionando itens chamados de componentes ao seu form. Um componente é um objeto Delphi, tal como um label, box ou edit.

FIGURA 1: Formulário e componentes



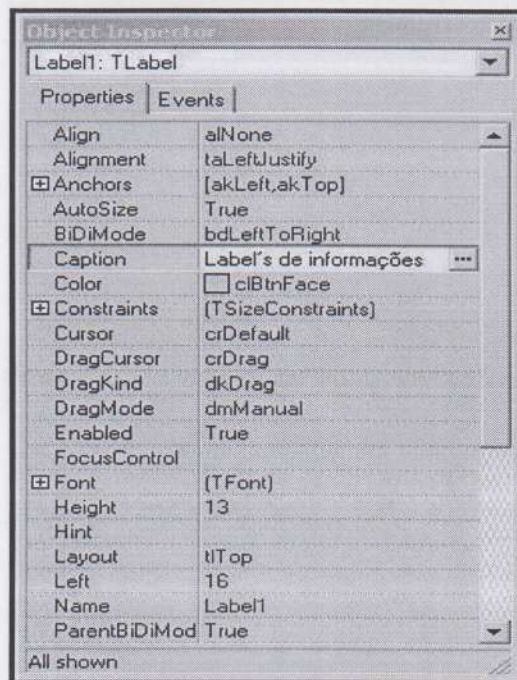
2.3.4 Object Inspector

O Object Inspector trabalha com a Component Palette. Uma vez selecionado um componente da Component Palette é adicionado ao form, o Object Inspector automaticamente exibe as propriedades e eventos que podem ser utilizados como o componente. As propriedades e eventos (exibidos no formato de menu) permitem personalizar os componentes visualmente sem a necessidade de codificação.

Os menus são dinâmicos no tocante em que somente as propriedades e eventos que se aplicam aos componentes selecionados aparecerão. Se múltiplos componentes são

selecionados, somente as propriedades e eventos compartilhados por todos os componentes aparecerão no Object Inspector.

FIGURA 2: Propriedades do Object Inspector



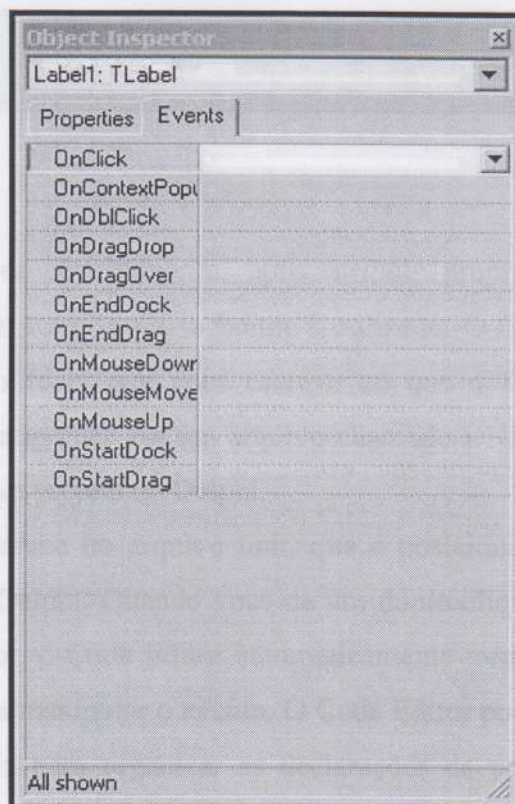
Properties Page

Propriedades são atributos de um componente. Elas controlam a forma como um componente aparece e se comporta na aplicação. Muitas propriedades na coluna Properties têm um valor default atribuído na coluna Values.

Componente Palette

A Componente Palette exibe uma seleção de componentes agrupados por função ou utilização. Um ponto chave do Delphi é que ele permite a criação de seus próprios componentes e personalização da Componente Palette. Você pode adicionar ou remover componentes da paleta, bem como criar uma nova página na paleta. A Componente Palette aparece abaixo da barra de menu, à direita, e consiste de botões (com os componentes) e páginas dividindo a funcionalidade de cada componente.

FIGURA 3: Eventos do Object Inspector



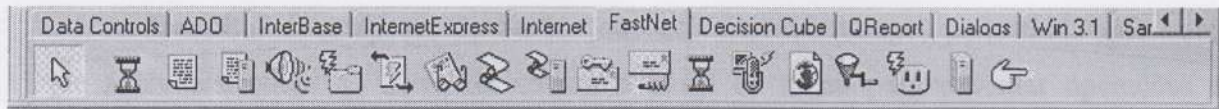
Events Page

A página Events do Object Inspector lista os eventos que um componente pode reconhecer. No Delphi, você escreve procedures chamadas event handlers, e as utiliza na página Events para associar o nome de um event handler com o evento. Eventos são ações do usuário ou ocorrências de sistema que o componente pode reconhecer.

Componente Palette

A Componente Palette exibe uma seleção de componentes agrupados por função ou utilização. Um ponto chave do Delphi é que ele permite a criação de seus próprios componentes e personalização da Componente Palette. Você pode adicionar ou remover componentes da paleta, bem como criar uma nova página na paleta. A Componente Palette aparece abaixo da barra de menu, a direita, e consiste de botões (com os componentes) e páginas dividindo a funcionalidade de cada componente.

FIGURA 4: Barra de Componentes

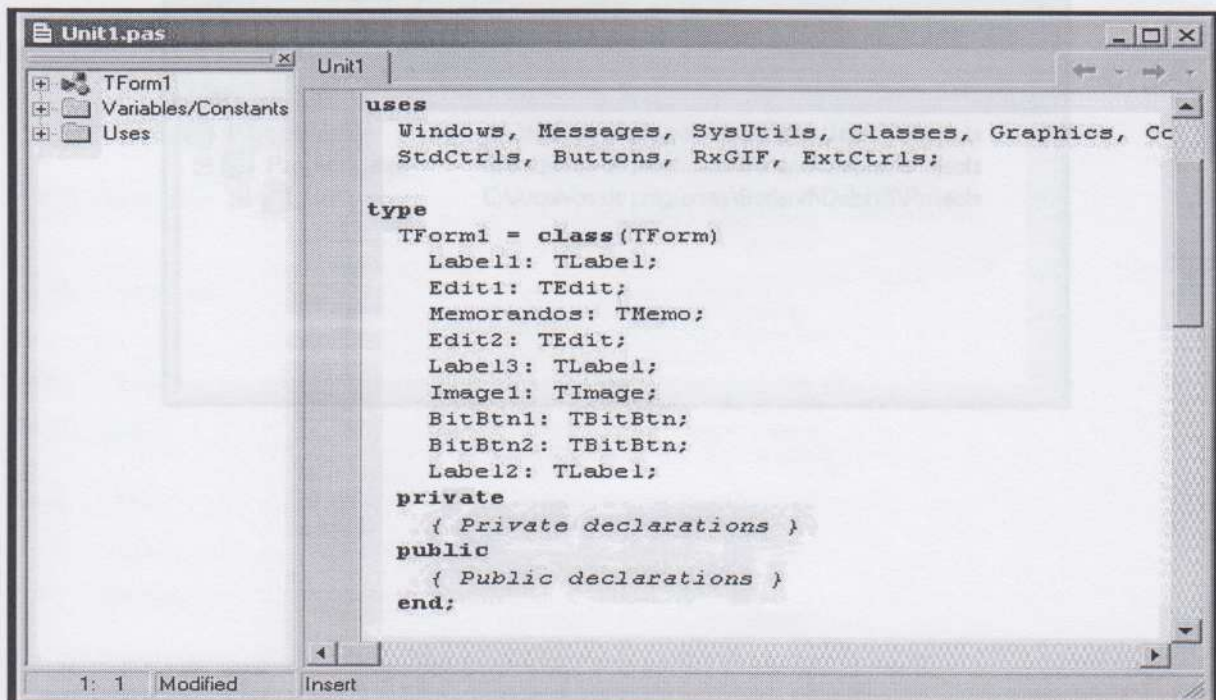


Code Editor

No processo de desenvolvimento, após atribuir propriedades e eventos a cada componente adicionado ao form, o Code Editor é utilizado. O Code Editor é um editor de texto que exibe o código fonte que você escreve ou que o Delphi gera para criar uma aplicação. O código fonte aparece em um arquivo chamado UNIT.PAS que é um dos tipos de arquivos que compõe um projeto do Delphi.

O Code Editor aparece no arquivo unit, que é posicionado atrás do Form1 default quando você inicializa o Delphi. Quando você dá um duplo clique em um evento na página Events do Object Inspector, o Code Editor automaticamente torna-se a janela para que você possa digitar o código para manipular o evento. O Code Editor pode exibir múltiplos arquivos unit. Uma unit é utilizada para organizar as declarações de programação do Delphi. Por exemplo, procedures utilitárias podem ser agrupadas como uma unit.

FIGURA 5: Editor de Código fonte



SpeedBar

A SpeedBar é composta de botões que economizam tempo, permitindo que você clique sobre um botão ao invés de utilizar a barra de menu para executar operações e comandos mais comuns.

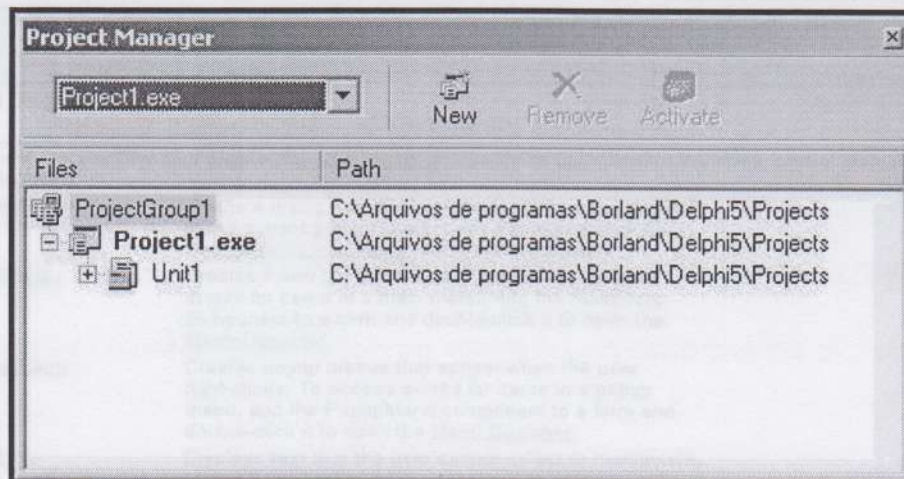
FIGURA 6: Barra de acesso rápido



Project Manager

No Delphi, os arquivos que compõem uma aplicação, form e units, são organizados como um projeto. O Project Manager lista cada arquivo de unit e form em sua aplicação e permite navegar entre elas. Visualizando a Janela do Project Manager aparece com o nome do projeto. Se o projeto não tiver um nome, o arquivo default chamado Project1, aparece como segue:

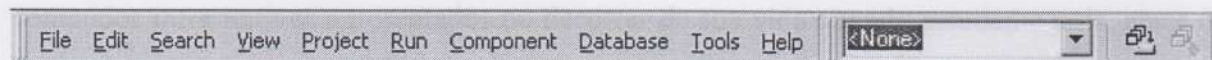
FIGURA 7: Gerenciador de Projetos



Speed Menu

Um elemento importante do IDE do Delphi é o SpeedMenu. No qual são menus que oferecem acesso rápido aos comandos disponíveis no momento. Eles são sensíveis ao Contexto.

FIGURA 8: Barra de menu

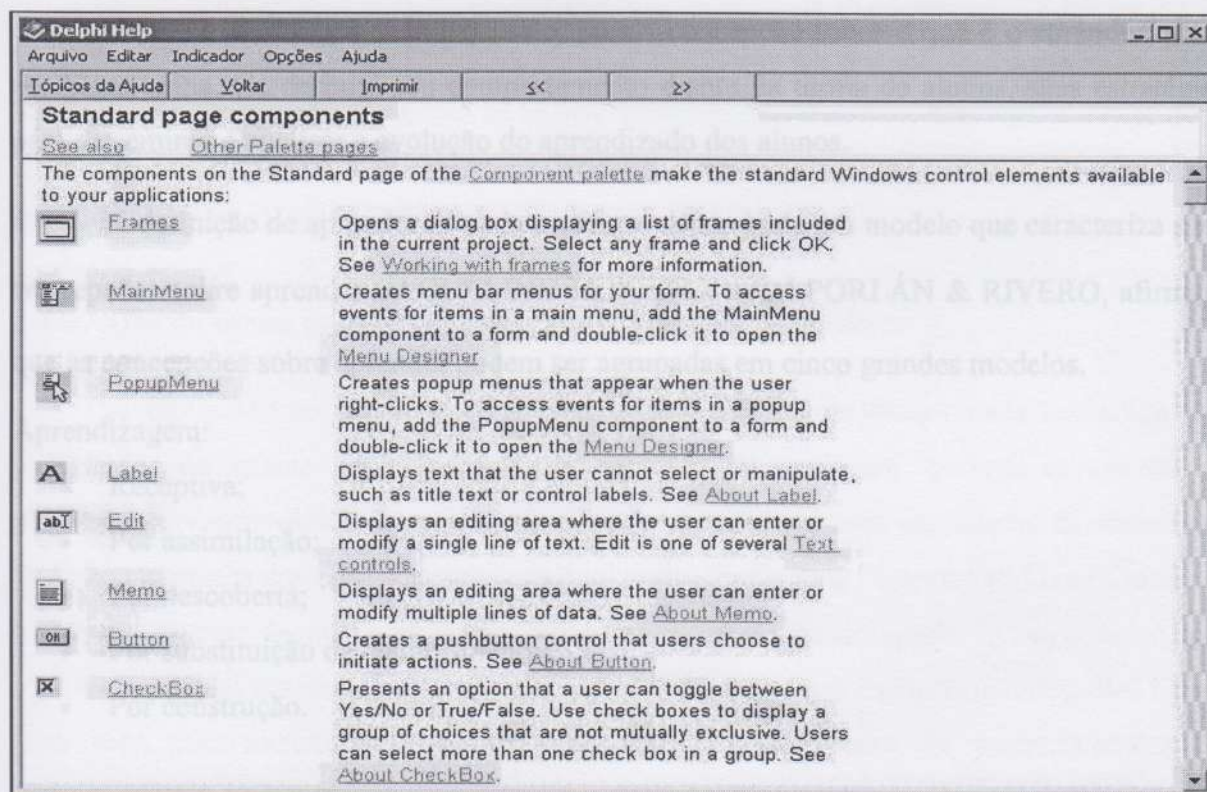


Help On-Line

O Help On-Line é uma grande fonte de referência sobre o Delphi. No menu Help, os itens Contents e Topic Search são comuns as aplicações MS Windows, em particular, inclui um glossário, que é útil para uma referência rápida sobre o significado de termos.

O Help é sensível ao contexto, se torna disponível quando o Code Editor for exibido e oferece auxílio na sintaxe da linguagem, palavras reservadas, e assim por diante.

FIGURA 9: Help On-Line – Padrão do Windows



2.4 SOFTWARE EDUCATIVO

Num contexto em que a quantidade de informações existentes no mundo, em que a cada instante novas fontes são descobertas, excede em muito o que se pode ser absorvida por uma pessoa durante sua vida inteira, fazendo-se necessário varias formas de potencializar o aprendizado através de ferramentas apropriadas.

Chega a ser injusto na realidade atual, as pessoas serem levadas a absorver excessivos conteúdos informativos apresentados no decorrer de sua vida quando estudante pois, antes de qualquer coisa, as atividades propostas pelos seus professores deveriam estimulá-los ao desenvolvimento de habilidades para utilizar, relacionar, analisar e avaliar tais informações.

O professor de acordo com a sua concepção de aprendizagem utilizará-se de suas estratégias para cumprir um determinado conteúdo a desenvolver com a turma, acompanhar e analisar a evolução do aprendizado de seus alunos, e enfim, direcionar todos os seus esforços para chegar em um determinado objetivo.

2.4.1 Quanto aos princípios da aprendizagem e os paradigmas educacionais

O que é aprender? Para as pessoas que atuam como professores, estão totalmente envolvidas com a resposta a essa questão, pois a concepção sobre o que é o aprender que o professor tenha, ele definirá seu comportamento diante da turma de alunos, suas estratégias para disseminar e analisar a evolução do aprendizado dos alunos.

A definição de aprender de cada professor depende de um modelo que caracteriza suas concepções sobre aprender. JAQUELINE TEIXEIRA apud PORLÁN & RIVERO, afirmam que as concepções sobre aprender podem ser agrupadas em cinco grandes modelos.

Aprendizagem:

- Receptiva;
- Por assimilação;
- Por descoberta;
- Por substituição de conhecimentos;
- Por construção.

No modelo de aprendizagem receptiva o processo de aprendizado é caracterizado por uma vasta memorização e acumulação de conhecimentos. Onde a aprendizagem se dá a partir da fala do professor, e nesse caso, os conhecimentos anteriores dos alunos são desconsiderados ou considerados errôneos.

O modelo de aprendizagem por assimilação dá ênfase naquilo que se foi aprendido anteriormente, conectando e relacionando seus conhecimentos escolares, valorizando a aprendizagem significativa.

A aprendizagem por descoberta enfatiza a descoberta dos alunos, no qual se dá uma busca intensa de conceitos e princípios por meio do esforço do próprio aluno, valorizando a experimentação e o trabalho prático como sua principal estratégia e fatores facilitadores do aprendizado.

Na aprendizagem por substituição de conhecimentos já existentes, considera-se que o aluno durante todo o seu processo de desenvolvimento adquiriu vários conhecimentos, que a princípio, podem ser considerados errôneos, aprender neste modelo, consiste na troca dessas informações erradas ou espontâneas pelos conhecimentos científicos.

Já no modelo de aprendizagem por construção, consiste na aprendizagem por evolução dos seus conhecimentos, valorizando os conhecimentos prévios dos alunos, tomando-os como ponto de partida para a construção de novos conhecimentos. Assim, aprender é tornar mais complexo o conhecimento cotidiano, acarretado por mudanças conceituais, no sentido de evolução dos conceitos já existentes.

2.4.2 Quanto ao uso da tecnologia aplicada ao processo educacional

A figura abaixo exibe os elementos existentes em uma determinada sociedade. As instituições de ensino atentam-se às necessidades da sociedade perante as constantes mudanças de desenvolvimento, e como os próprios membros dessas instituições de ensino são também elementos dessa tal sociedade, na qual estão diretamente comprometidos na formação e aperfeiçoamento daqueles que foram colocados à sua responsabilidade. O computador, que aqui representa as diversas tecnologias de informática existentes e usadas na educação. Nesse caso toda essa tecnologia pode ser usada como amplificador de potencialidades na capacitação dos alunos, dos professores e da própria instituição de ensino.

2.5 Para JAQUELINE TEIXEIRA apud PORLÁN & RIVERO a relação entre Softwares Educativos e Paradigma Educacional pode ser representado pela figura e a tabela abaixo.

FIGURA 10: Tipos e modalidade de softwares educativos x Paradigma Educacional



TABELA 1: Relação entre os paradigmas educacionais, suas características e

Paradigma Educacional	Visão da natureza humana	Quanto a atividade do aprendiz.	Quanto ao direcionamento na utilização do software	Modalidades de Software Educacional
Comportamentalista	Empirista e racionalista	Algorítmico	Dura	Tutoriais, Exercitação e prática
Construtivista	Interacionalista	Heurístico	Branda	Simulação, Jogos

2.4.3 Ensino X Aprendizagem

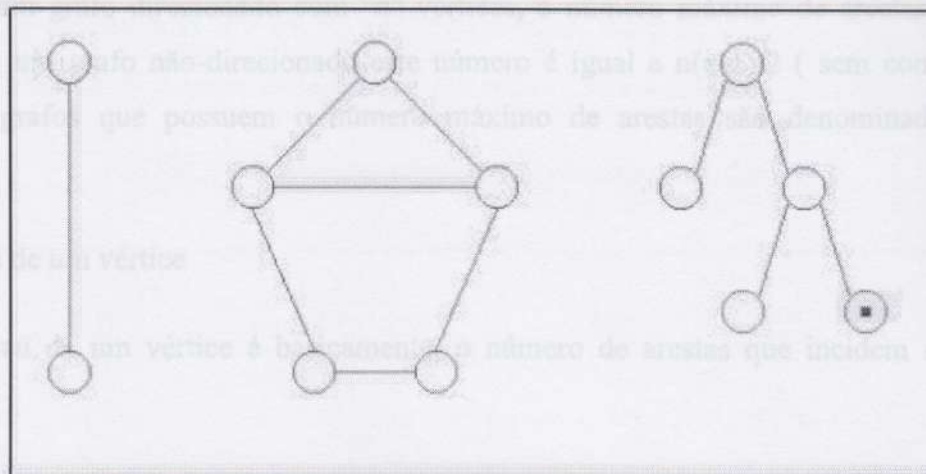
A utilização de Software Educativo pode auxiliar o desenvolvimento do pensar crítico e do aprender a aprender nos alunos. Entende-se que independente da modalidade de software desenvolvido e utilizado num determinado contexto, é a concepção de aprender que o professor tenha, que direcionará o uso da ferramenta. Desta forma, numa abordagem construtivista, a utilização de Software Educativo pode contribuir para tratar de propostas intelectuais que dificilmente seriam possíveis de serem criadas nas suas melhores formas sem a utilização do computador.

2.5 GRAFOS

Segundo GOLDBARG e LUNA (2000), pode-se considerar que um Grafo, nada mais é que uma representação gráfica de interdependência entre elementos que são representados por nós, simbolicamente unidos através de um traço denominado aresta.

“Um grafo G consiste de um conjunto não vazio de elementos, chamado vértices, e uma lista de pares destes elementos não ordenados, chamados arestas. Se v e w são vértices de G então um arco vw ou wv é dito ligar v a w .” (Henriques apud Wilson e Watkins 1990).

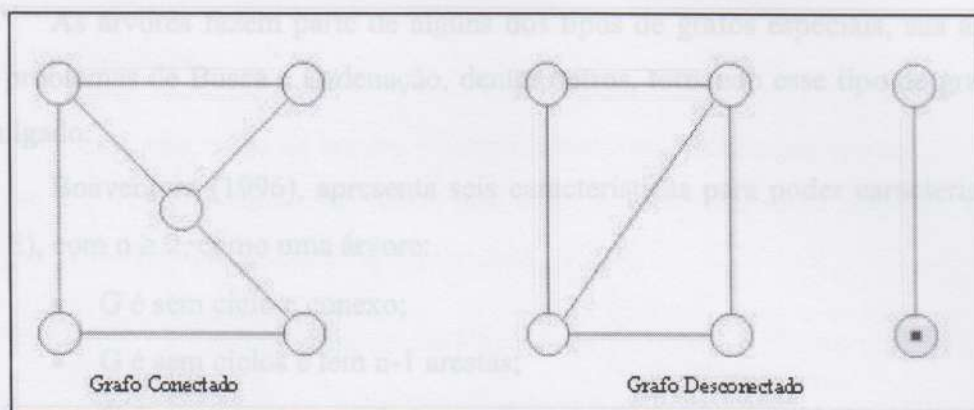
FIGURA 11: Representação de alguns grafos



2.5.1 Grafo direcionado

“Um Grafo Direcionado denotado por $G_D(V, E)$, ou basicamente Digrafo, consiste de um conjunto V de elementos chamados vértices e de um conjunto de pares ordenados na forma (i, j) , $i, j \in V$, chamados arcos, ou arestas direcionadas, de G . O vértice “ i ” é chamado de vértice inicial e o vértice “ j ” de vértice final, juntos são chamados pontos finais.” (Henriques apud Chen 1990).

2.5.3 FIGURA 12: Representação de grafos conectados e desconectados

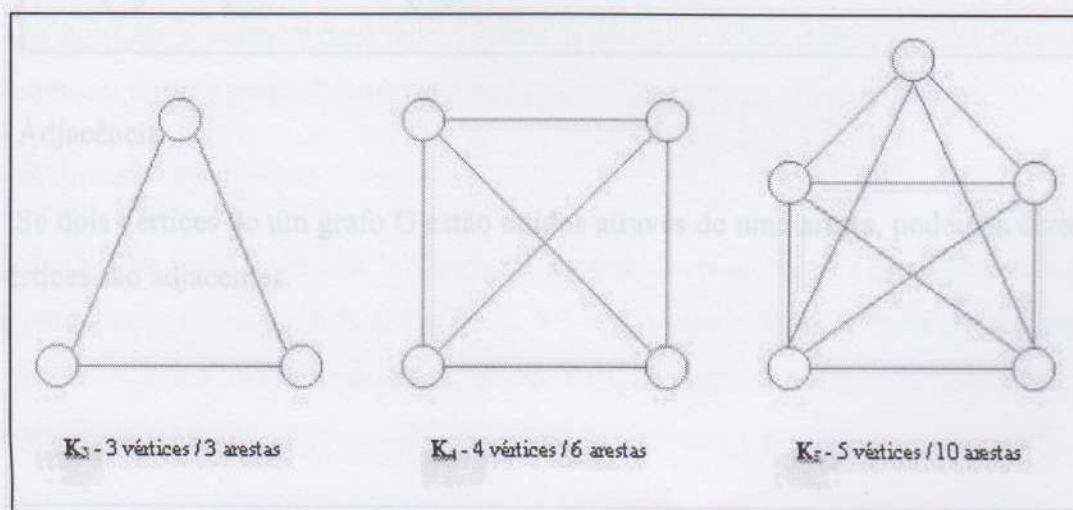


Em um grafo direcionado com " n " vértices, o número máximo de arestas é igual a $n(n-1)$ e em um grafo não-direcionado este número é igual a $n(n-1)/2$ (sem considerar os laços). Os grafos que possuem o número máximo de arestas são denominados grafos completos.

2.5.2 Grau de um vértice

O grau de um vértice é basicamente, o número de arestas que incidem sobre este vértice.

FIGURA 13: Grau de um vértice em um Grafo G



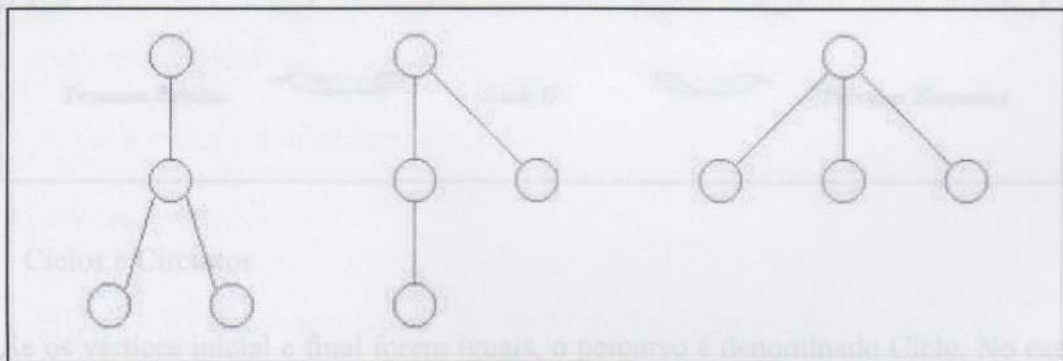
2.5.3 Árvores

As árvores fazem parte de alguns dos tipos de grafos especiais, sua aplicação é feita em problemas de Busca e Ordenação, dentre outros, tornando esse tipo de grafo amplamente divulgado.

Boaventura (1996), apresenta seis características para poder caracterizar um grafo G (V, E) , com $n \geq 2$, como uma árvore:

- G é sem ciclo e conexo;
- G é sem ciclos e tem $n-1$ arestas;
- G é conexo e tem $n-1$ arestas;
- G é sem ciclos e a adição de uma aresta cria um ciclo único;
- G é conexo, mas $G' = (V, E - \{e\})$ é desconexo $\forall e \in E$;
- Todo par de vértices de G é unido por uma cadeia única.

FIGURA 14: Árvores com quatro vértices de Ordem 4



2.5.4 Adjacência

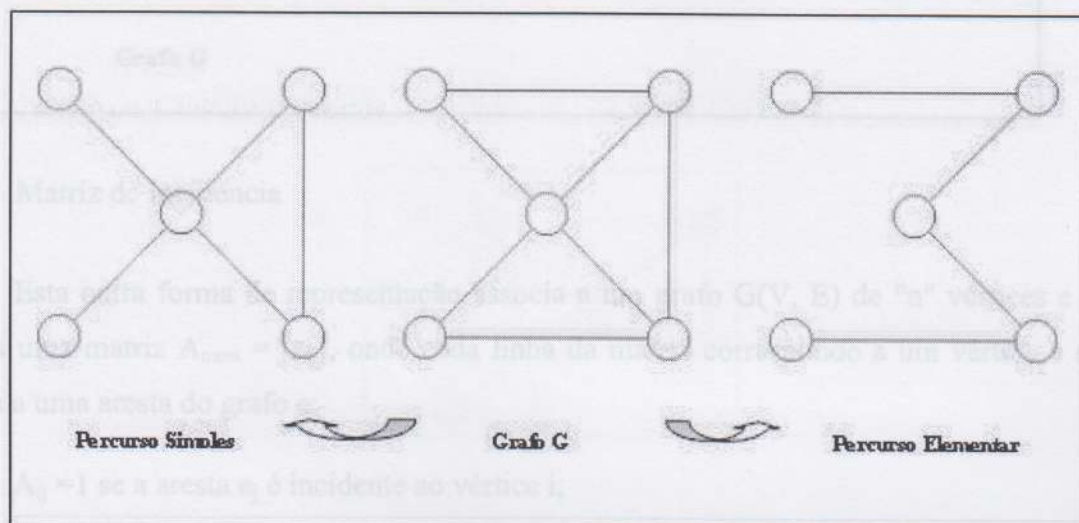
Se dois vértices de um grafo G estão unidos através de uma aresta, podemos dizer que estes vértices são adjacentes.

2.5.5 Percursos, Ciclos e Circuitos

2.5.5.1 Percursos

É uma sequência de arestas sucessivamente adjacentes que vão de um vértice V_i a um vértice V_f . Um percurso que não repete arestas é dito Percurso Simples, e se não repete vértice é dito Percurso Elementar (Henriques - 2003).

FIGURA 15: Demonstração de Percursos Simples e Elementar de um grafo G



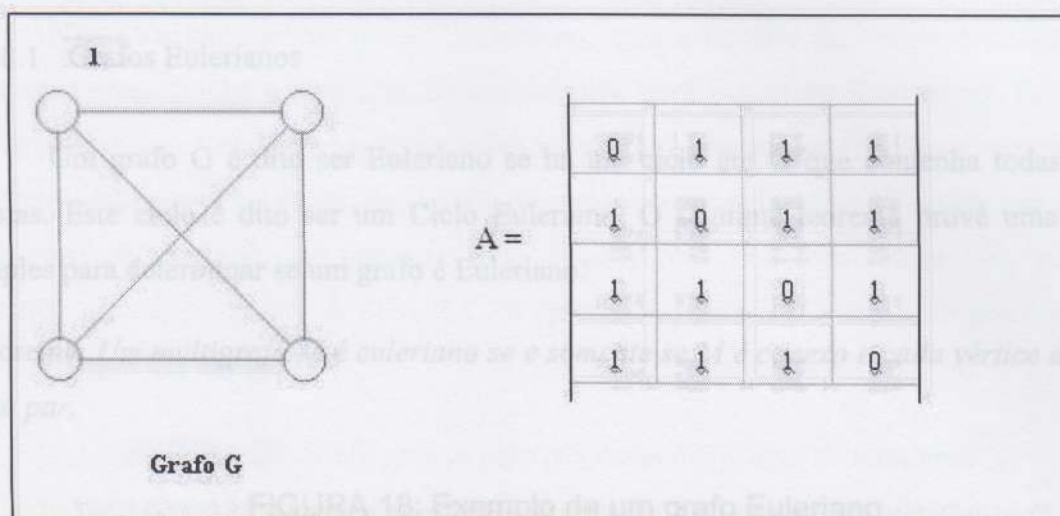
2.5.5.2 Ciclos e Circuitos

Se os vértices inicial e final forem iguais, o percurso é denominado Ciclo. No caso do Ciclo acontecer em um grafo direcionado, é chamado Circuito.

2.5.6 Matriz de Adjacência

Se G é um grafo de ordem " n ", a matriz de adjacência necessária para representar este grafo é uma matriz de ordem $n \times n$. Seja $A_{n \times n} = [a_{ij}]$ a matriz de adjacência de um grafo $G(V, E)$, se $a_{ij} = 1$, então existe uma aresta ligando i a j , ao contrário se $a_{ij} = 0$, então não existe uma aresta ligando i a j .

FIGURA 16: Representação da Matriz de Adjacência de um Grafo G



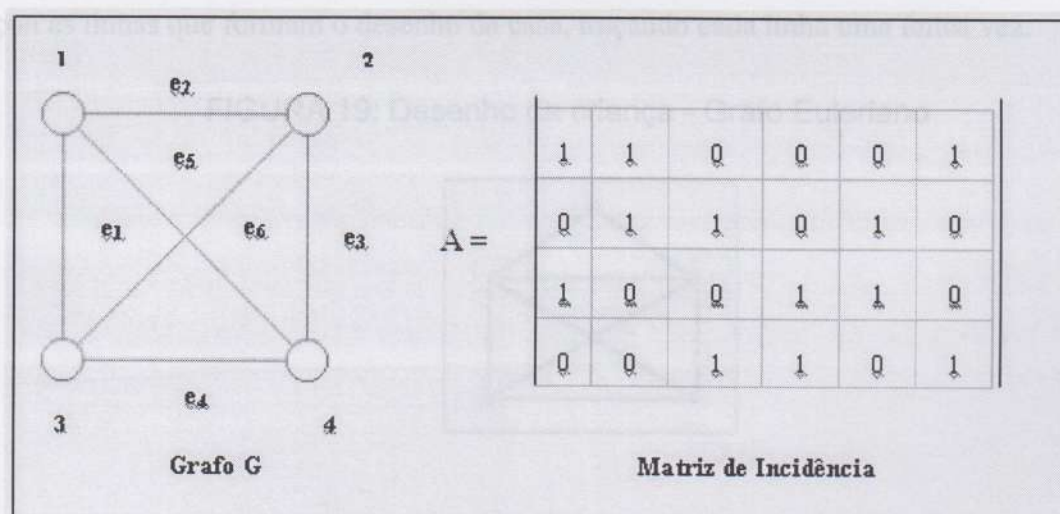
2.5.7 Matriz de Incidência

Esta outra forma de representação associa a um grafo $G(V, E)$ de " n " vértices e " m " arestas uma matriz $A_{n \times m} = [a_{ij}]$, onde cada linha da matriz corresponde a um vértice e cada coluna a uma aresta do grafo e:

$A_{ij} = 1$ se a aresta e_j é incidente ao vértice i ;

$A_{ij} = 0$ caso contrário.

FIGURA 17: Representação da Matriz de Incidência de um Grafo G



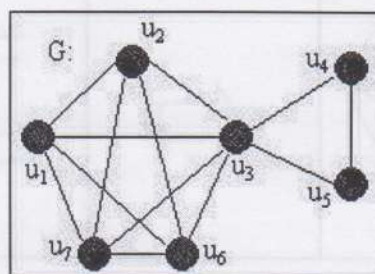
2.5.8 Alguns problemas clássicos envolvendo grafos

2.5.8.1 Grafos Eulerianos

Um grafo G é dito ser Euleriano se há um ciclo em G que contenha todas as suas arestas. Este ciclo é dito ser um Ciclo Euleriano. O seguinte teorema provê uma solução simples para determinar se um grafo é Euleriano:

Teorema: *Um multigrafo M é euleriano se e somente se M é conexo e cada vértice de M tem grau par.*

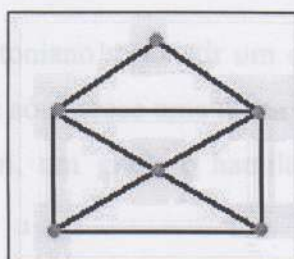
FIGURA 18: Exemplo de um grafo Euleriano



2.5.8.1.1 Problema do desenho da casa

A figura abaixo exibe o desenho de uma criança que diz ter posto a ponta do lápis numa das bolinhas e com movimentos contínuos (sem levantar e sem retroceder o lápis) traçou as linhas que formam o desenho da casa, traçando cada linha uma única vez.

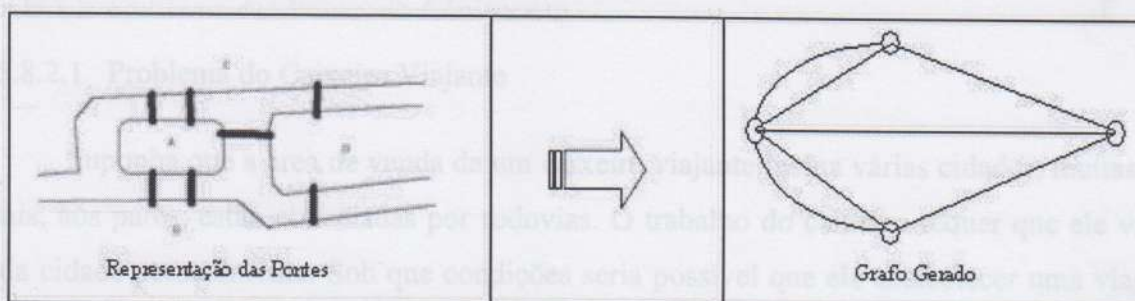
FIGURA 19: Desenho da criança - Grafo Euleriano



2.5.8.1.2 Problema das Pontes de Königsberg

Não só a história dos Grafos Eulerianos, mas a história da Teoria dos Grafos, tem como seu ponto inicial a resolução do mistério das setes pontes de Königsberg. Em 1936, o matemático suíço Leonardo Euler, apresentou uma solução matemática para o problema. O problema das pontes, na época, fazia parte do cotidiano dos habitantes da cidade que tentavam encontrar um caminho que percorresse as setes pontes exatamente uma vez, partindo e voltando ao mesmo ponto. A FIGURA 20 apresenta uma representação do problema e sua transformação em um grafo.

FIGURA 20: Grafo gerado pelo problema das pontes de Königsberg

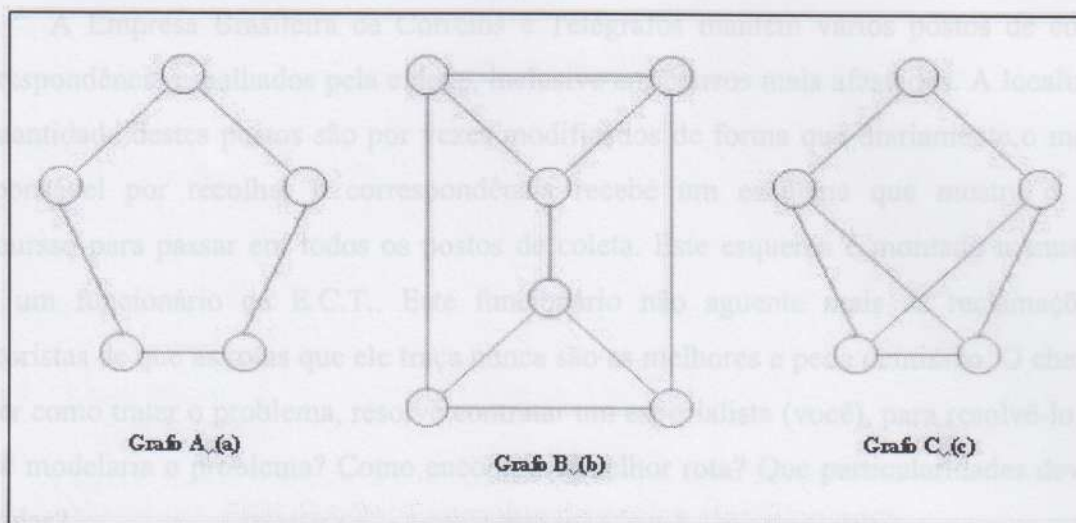


As regiões da cidade, inclusive a ilha de Kneiphof, foram transformadas nos vértices do grafo G , e as pontes que interligam as regiões transformaram-se nas arestas, gerando como produto final um multigrafo G . O problema das pontes de Königsberg foi transformado em um problema de grafo, que consiste em encontrar um caminho em um grafo G , que contenha todas as arestas do grafo uma única vez. Este caminho é conhecido, hoje, como Caminho Euleriano.

2.5.8.2 Grafos Hamiltonianos

Um grafo G é dito ser hamiltoniano se existir um ciclo em G que contenha todos os seus vértices, sendo que cada vértice só aparece uma única vez no ciclo. Este ciclo é chamado de ciclo hamiltoniano. Sendo assim, um grafo é hamiltoniano se ele contiver um ciclo hamiltoniano.

FIGURA 21: Exemplos de grafos Hamiltonianos



2.5.8.2.1 Problema do Caixeiro Viajante

Suponha que a área de venda de um caixeiro viajante inclua várias cidades, muitas das quais, aos pares, estão conectadas por rodovias. O trabalho do caixeiro requer que ele visite cada cidade pessoalmente. Sob que condições seria possível que ele estabelecer uma viagem circular (que o leve ao ponto de partida) de forma a que ele visite cada cidade exatamente uma vez?

Este problema pode ser modelado por um grafo $G(V,A)$, onde:

$$V = \{ c \mid c \text{ é uma cidade} \}$$

$$A = \{ (c_1, c_2) \mid \text{há uma estrada que conecta as cidades } c_1 \text{ e } c_2, \text{ sendo que ela não passa por nenhuma outra cidade neste trajeto} \}$$

Modelado desta forma, a solução deste problema passa por verificar se o grafo G é hamiltoniano.

Qualquer um que tenta implementar problemas envolvendo grafos irá se deparar com uma das principais dificuldades que se tem sempre que desejamos envolver a exibição das abstrações de um problema em formas fáceis de visualização, tornando-se bem mais

2.5.8.3 O Problema da Coleta de Correspondências

A Empresa Brasileira de Correios e Telégrafos mantém vários postos de coleta de correspondência espalhados pela cidade, inclusive em bairros mais afastados. A localização e a quantidade destes postos são por vezes modificados de forma que diariamente o motorista responsável por recolher a correspondência recebe um esquema que mostra o melhor percurso para passar em todos os postos de coleta. Este esquema é montado manualmente por um funcionário da E.C.T.. Este funcionário não aguenta mais as reclamações dos motoristas de que as rotas que ele traça nunca são as melhores e pede demissão. O chefe, sem saber como tratar o problema, resolve contratar um especialista (você), para resolvê-lo. Como você modelaria o problema? Como encontrar a melhor rota? Que particularidades devem ser tratadas?

2.6 APLICAÇÃO

2.6.1 Arquitetura da aplicação

A arquitetura da aplicação, basicamente consiste na manipulação entre dois objetos principais, que representaram os vértices e suas ligações denominadas de arestas. Tais objetos tiveram suas estruturas modeladas de forma a se obter a qualquer momento, informações sobre um vértice e/ou uma aresta qualquer de um grafo.

A idéia principal desse projeto, é a criação de um software com fins pedagógicos e de incentivo a pesquisa, induzindo a utilização da computação, aplicada a teoria dos grafos, onde, a cada momento em que o aluno estiver estudando a disciplina, poderá sem tantas dificuldades, incorporar novas rotinas e funcionalidades a aplicação. Isso se dá possível pela estruturação inicial modelada durante essa investigação.

Como resultado teremos no mínimo, como aparelhar os alunos com os recursos da teoria dos grafos para resolução de problemas e dar-lhes oportunidade de desenvolver suas habilidades de programação, elaborando novas rotinas ou programas para a manipulação de grafos.

Qualquer um que tenta implementar problemas envolvendo grafos irá se deparar com uma das principais dificuldades que se tem sempre que desejamos envolver a exibição das abstrações de um problema em formas fáceis de visualização, tornando-se bem mais

complexo, quando se necessita do armazenamento de informações tais como posicionamento de limites, regiões, tamanho, distancia dentre outros.

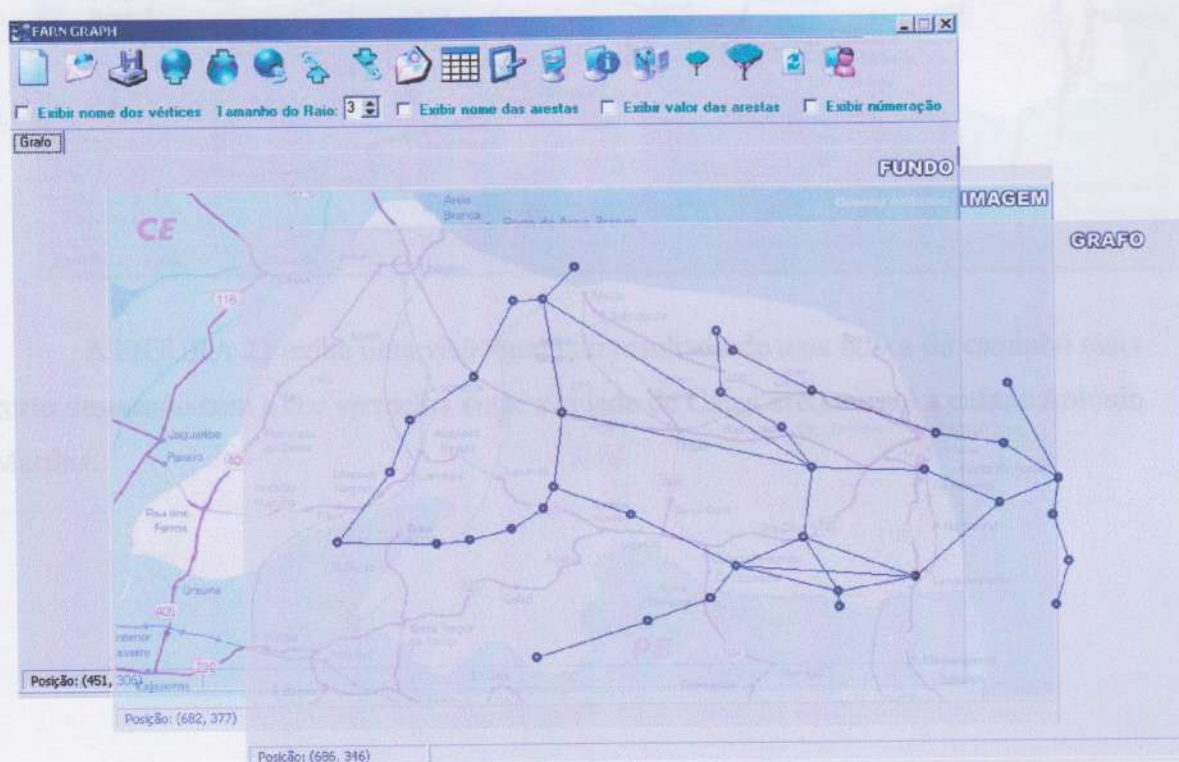
Para entendermos melhor esse projeto iremos dividi-lo em quatro partes:

- Visualização
- Interface
- Informações
- Armazenamento

2.6.1.1 Visualização

A figura abaixo demonstra a forma em que se foi divididos as camadas existentes para a exibição do grafo.

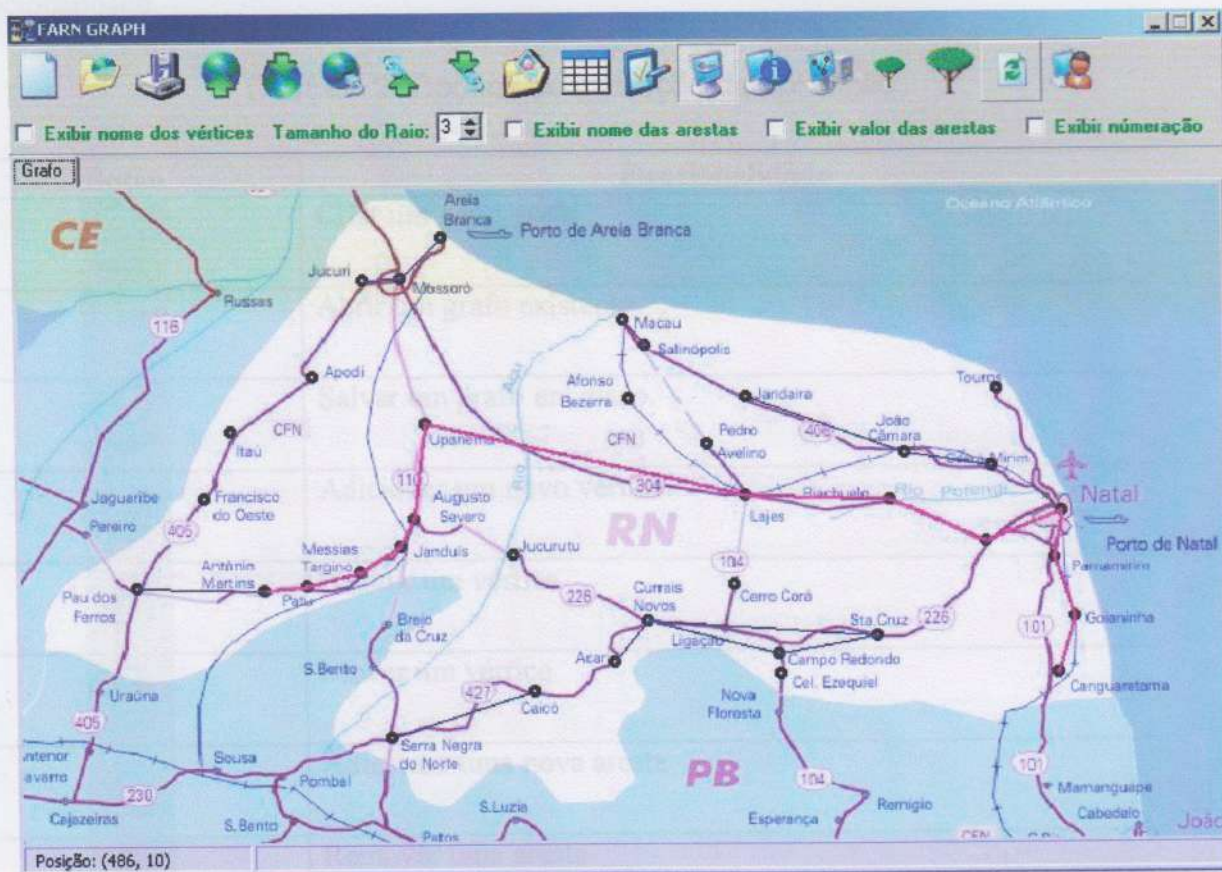
FIGURA 22: Visualização das camadas de exibição



- . A camada *fundo* faz parte do próprio formulário da aplicação.
- . A camada *imagem* tem a função de auxílio no mapeamento e na construção do grafo.
- . A camada *grafo* será responsável pela exibição do grafo de fato.

2.6 Com a sobreposição dessas camadas a aplicação ficará com a seguinte apresentação:

FIGURA 23: Visualização final



A FIGURA 23 exibe uma visão geral do resultado de uma busca do caminho mais curto destacado com a cor vermelha entre a cidade de Canguaretama até a cidade Antônio Martins.

2.6.1.2 Interface

A interface de interação com o usuário se dá através de botões para cada funcionalidade.

TABELA 2: Opções de interação da aplicação

Botão	Funcionalidade
	Criar um novo grafo
	Abrir um grafo existente.
	Salvar um grafo em disco.
	Adicionar um novo vértice
	Excluir um vértice
	Mover um vértice
	Adicionar uma nova aresta
	Remover uma aresta
	Abrir uma imagem de fundo
	Exibir o grafo em forma de uma matriz de adjacência
	Analisar o grafo
	Exibir e Ocultar imagem de fundo
	Exibir as informações de um vértice selecionado
	Executa o algoritmo do caminho mais curto
	Força a atualização da tela
	Informações sobre a aplicação

TABELA 3: Opções de Configuração

Opções de configuração do grafo	
☐ Exibir nome dos vértices	Seleção da visualização dos nomes dos vértices
Tamanho do Raio: 3	Tamanho do raio de cada vértice
☐ Exibir nome das arestas	Seleção da visualização dos nomes das arestas
☐ Exibir valor das arestas	Seleção da visualização dos valores das arestas
☐ Exibir numeração	Exibe a numeração de cada vértice e aresta

2.6.1.3 Informações

A recuperação das informações contidas no grafo será feita através das propriedades internas de cada instancia dos objetos da aplicação conforme as estruturas de cada objeto.

Logo abaixo, um detalhamento da estruturação de cada um dos objetos:

TABELA 4: Classe TDadosForma

Bloco de declaração da Classe TDadosForma
<pre>TDadosForma = class TipoForma, Nome, Informacoes: String; X, Y, Tamanho, TamCaneta, NumForma: integer; CorCaneta: TColor; end;</pre>

TABELA 5: Propriedades da classe TDadosForma

Propriedade	Descrição
TipoForma	Será armazenado o tipo da forma geométrica em que se deseja pintar os pontos numa determinada área da tela o qual representará um vértice, lembrando que, nesse protótipo de software só foi implementada a forma de círculo. Exemplo: Círculo, Retângulo...
Nome	Nessa propriedade será armazenado o nome da localização em que o vértice irá representar. Exemplo: Natal, João, Alemanha e outros...
Informações	Em informações será armazenado o dado referente ao vértice. Exemplo: Acari é considerada a cidade mais limpa do Brasil, no qual foi habitada primitivamente pelo índios cariris e fica a 209 Km de Natal/RN.

X	Coluna em que será exibido o vértice.
Y	Linha em que será exibido o vértice.
Tamanho	Tamanho do Raio do vértice. Exemplo: por padrão a aplicação iniciará cada grafo com o tamanho 3 podendo ser modificado pelo usuário.
TamCaneta	Largura em que será contornada a forma geométrica que será exibido o vértice. Exemplo: 1, 2, 3... Nesse protótipo, por padrão a aplicação, essa propriedade iniciará com o valor 1 não podendo ser alterado.
NumForma	Número seqüencial da forma que servirá como identificador de cada vértice. Exemplo: 1, 2, 3

TABELA 6: Classe TDadosAresta

Bloco de declaração da Classe TDadosAresta

```
TDadosAresta = class
    Origem, Destino, TamCaneta, NumForma, Valor: integer;
    CorCaneta: TColor;
    Nome: String;
end;
```

TABELA 7: Propriedades da classe TDadosAresta

Propriedade	Descrição
Origem	Identificará o vértice de origem de uma ligação entre dois vértices.
Destino	Identificará o vértice de destino de uma ligação entre dois vértices.
TamCaneta	Largura em que será pintado o traço que determinará uma aresta. Exemplo: 1, 2, 3... Nesse protótipo, por padrão a aplicação, essa propriedade iniciará com o valor 1 mudando de tamanho após a execução de algum algoritmo de busca.

NumForma	Número sequencial da forma que servirá como identificador de cada aresta. Exemplo: 1, 2, 3
Valor	Valor correspondente a ligação entre os dois pontos.
Cor	Cor na qual será pintado o traço representante de cada aresta. Exemplo: clBlack, clBlue, clWhite... Nesse protótipo, por padrão, a aplicação ira utilizar o valor clBlue, essa propriedade poderá ser mudada, quando houver a execução de algum algoritmo de busca.
Nome	Nessa propriedade será armazenado o nome da aresta de ligação entre dois vertices. Exemplo: Natal - Acari, João - Fernando, Alemanha - Japão, e outros...

2.6.2 Formato do arquivo gerado pelo FARN Graph.

O formato do arquivo de armazenamento dos grafos desta ferramenta é o padrão XML (eXtensible Markup Language), ou seja, Linguagem de Marcação Extensível na qual é um dos métodos padrão de identificação e descrição de dados na Web.

2.6.3 Funcionamento da XML

É largamente implementada e de fácil organização, onde, “Tags” particulares habilitam a definição, transmissão, validação e interpretação de dados entre aplicações e organizações.

2.6.4 Características de XML

Troca de dados estruturados para não estruturados: XML possibilita uma syntax padrão universal para troca de dados. XML especifica um rigoroso meio de representar a estrutura inerente de dados baseado em texto para que possam ser reconhecidos e interpretados sem ambigüidade. É simples, sendo baseada em “tags”, ela equilibra a familiaridade dos desenvolvedores com HTML, mas providencia um mecanismo flexível e extensível que pode suportar uma gama de tipos de dados diferentes desde registros de banco de dados estruturados até documentos desestruturados e tudo que estiver no meio.

2.6.5 Exemplo do armazenamento de um grafo em XML

Na FIGURA 24 mostramos um exemplo simples de um grafo e logo após na FIGURA 25 o seu código de armazenamento XML.

FIGURA 24: Grafo de Exemplo

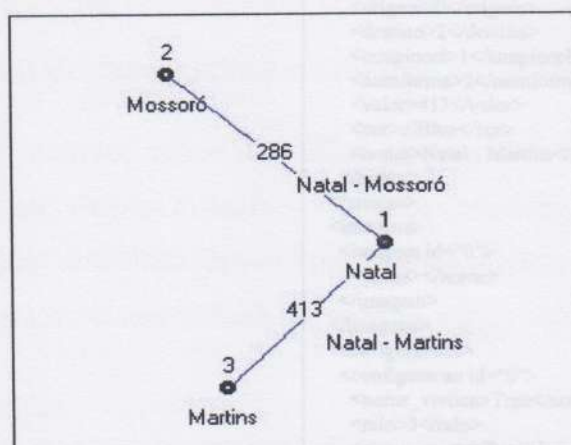


TABELA 8: Visualizando o XML em um Web Browser

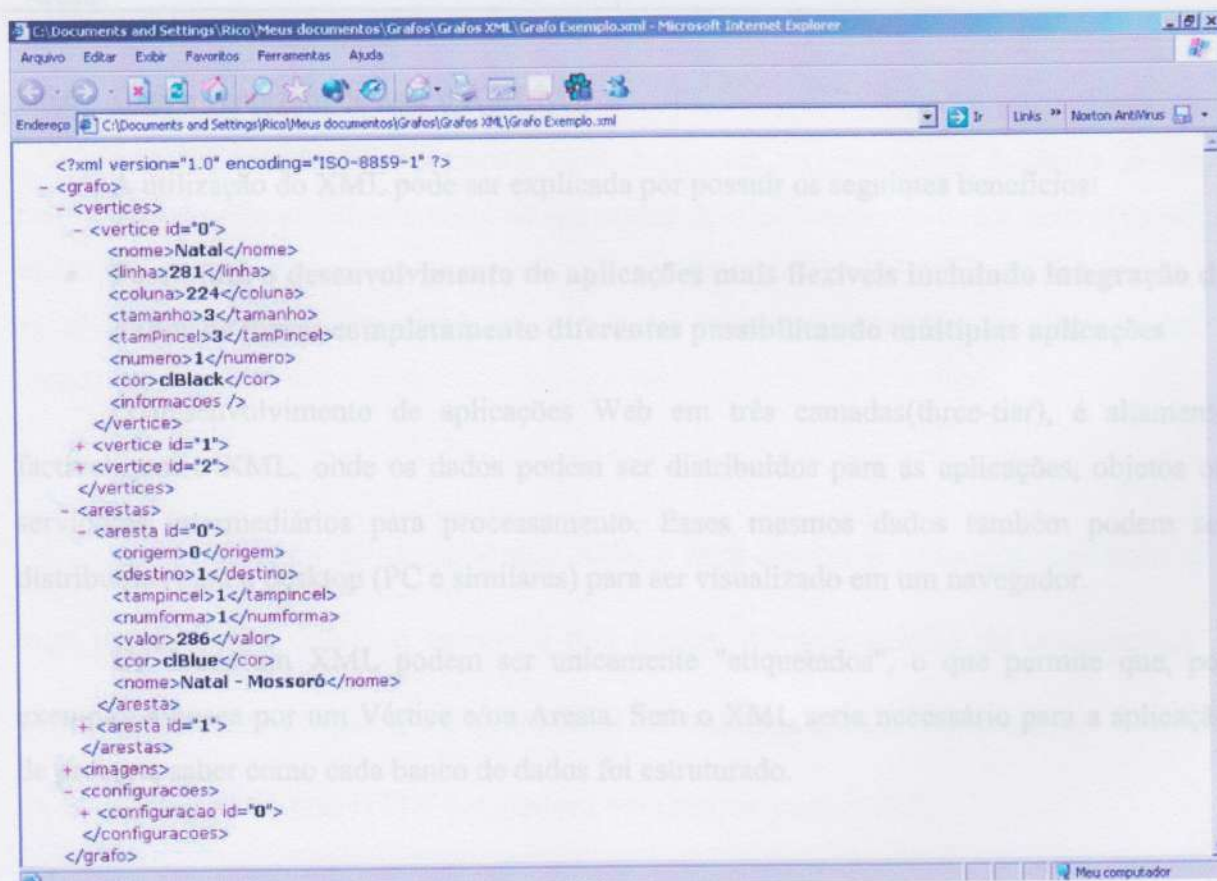


TABELA 9: Código em XML

<pre> <?xml version="1.0" encoding="ISO-8859-1" ?> <grafo> <vertices> <vertice id="0"> <nome>Natal</nome> <linha>281</linha> <coluna>224</coluna> <tamanho>3</tamanho> <tamPincel>3</tamPincel> <numero>1</numero> <cor>clBlack</cor> <informacoes></informacoes> </vertice> <vertice id="1"> <nome>Mossoró</nome> <linha>173</linha> <coluna>123</coluna> <tamanho>3</tamanho> <tamPincel>3</tamPincel> <numero>2</numero> <cor>clBlack</cor> <informacoes></informacoes> </vertice> <vertice id="2"> <nome>Martins</nome> <linha>202</linha> <coluna>310</coluna> <tamanho>3</tamanho> <tamPincel>3</tamPincel> <numero>3</numero> <cor>clBlack</cor> <informacoes></informacoes> </vertice> </vertices> <arestas> </pre>	<pre> <aresta id="0"> <origem>0</origem> <destino>1</destino> <tampincel>1</tampincel> <numforma>1</numforma> <valor>286</valor> <cor>clBlue</cor> <nome>Natal - Mossoró</nome> </aresta> <aresta id="1"> <origem>0</origem> <destino>2</destino> <tampincel>1</tampincel> <numforma>2</numforma> <valor>413</valor> <cor>clBlue</cor> <nome>Natal - Martins</nome> </aresta> </arestas> <imagens> <imagem id="0"> <nome></nome> </imagem> </imagens> <configuracoes> <configuracao id="0"> <nome_vertice>True</nome_vertice> <raio>3</raio> <nome_aresta>True</nome_aresta> <valor_aresta>True</valor_aresta> <numeracao>True</numeracao> </configuracao> </configuracoes> </grafo> </pre>
---	---

2.6.6 Qual a vantagem de usar o XML?

A utilização do XML pode ser explicada por possuir os seguintes benefícios:

- **Possibilita o desenvolvimento de aplicações mais flexíveis incluindo integração de dados de fontes completamente diferentes possibilitando múltiplas aplicações**

O desenvolvimento de aplicações Web em três camadas(three-tier), é altamente factível com o XML, onde os dados podem ser distribuídos para as aplicações, objetos ou servidores intermediários para processamento. Esses mesmos dados também podem ser distribuídos para o desktop (PC e similares) para ser visualizado em um navegador.

Os dados em XML podem ser unicamente "etiquetados", o que permite que, por exemplo, a busca por um Vértice e/ou Aresta. Sem o XML seria necessário para a aplicação de procura, saber como cada banco de dados foi estruturado.

- **Computação e manipulação local dos dados**

Os dados XML recebidos por um cliente são analisados e podem ser editados e manipulados de acordo com o interesse do usuário. Ao contrário de somente visualizar os dados, os usuários podem manipulá-los de várias formas. Os recursos disponíveis do Document Object Model (DOM) permitem que os dados sejam manipulados via scripts ou outra linguagem de programação.

- **Múltiplas formas de visualização e atualização do conteúdo**

A separação da interface visual dos dados propriamente ditos permite a criação de aplicações mais poderosas, simples e flexíveis. Os dados recebidos em XML por um usuário podem ser visualizados de diferentes formas uma vez que o XML define somente os dados e não o visual. A interpretação visual poderia ser dada de várias maneiras diferentes, de acordo com as aplicações.

- **Distribuição dos dados via rede de forma mais comprimida e escalável**

Assim como o HTML, o XML, por ser um formato baseado em texto aberto, pode ser distribuído via HTTP sem necessidade de modificações nas redes existentes.

- **Escalabilidade**

Devido ao fato dos documentos XML separarem completamente os dados da forma com a qual são visualizados, autores de aplicações de visualização de dados podem torná-las muito poderosas e interativas, permitindo ao usuário visualizar os dados da forma que lhe agrade. Dessa forma, a interatividade, em termos, não dependeria tanto da aplicação que o gerou.

- **Compressão**

A compressão de documentos XML é fácil devido à natureza repetitiva das tags usadas para definir a estrutura dos dados. A necessidade de compressão é dependente da aplicação e da quantidade de dados a serem manipulados. Padrões de compressão do tipo HTTP 1.1 podem ser usados para o XML.

2.6.7 Recursos implementados

O propósito deste projeto era de criar uma estrutura base para que futuros alunos pudessem ampliá-la, implementando os seus próprios recursos uma vez a preocupação com a interface não fosse necessária, pois já estaria desenvolvida.

Tal propósito foi totalmente atendido onde podemos comprovar seu pleno funcionamento através dos recursos extras já implementados, tais como:

- ✓ Criação de novos grafos não direcionados utilizando apenas o mouse, clicando nas regiões que se deseja adicionar e/ou manipular os vértices e Arestas.
- ✓ Alteração e modificação de grafos já existentes, criados pela aplicação anteriormente.
- ✓ Meios de manipulação com Vértices e Arestas, possibilitando a inclusão, associação e exclusão entre os Vértices e Arestas.
- ✓ Adicionar e recuperar informações independentes referentes a cada vértice criado.
- ✓ Criar um grafo a partir de um mapeamento feito utilizando-se como base visual uma imagem de fundo.
- ✓ Geração da Matriz de Adjacência.
- ✓ Habilitar e desabilitar a exibição da imagem de fundo de um Grafo.

Alem dos recursos citados anteriormente, também foram implementados alguns algoritmos abordados durante o estudo da Teoria dos Grafos, tais como:

- ✓ Caminho mais curto;
- ✓ Arvore Geradora Mínima;
- ✓ Arvore Geradora Máxima.

3 METODOLOGIA

A área de Teoria dos Grafos trata de abstrações de certos problemas práticos encontrados em uma infinidade de ramos totalmente distintos tais como projetos de chips, administração de frotas de veículos, sistemas operacionais de computadores, sistemas de posicionamento geográfico, etc, como também em outras áreas como a da pesquisa operacional.

A metodologia aplicada que será proposta nesse projeto surge de vários tipos de pesquisas distintas, nos quais suas características classificam-se da seguinte forma:

Um dos tipos de pesquisa abordada será a *pesquisa aplicada*, pois o projeto irá tratar de problemas freqüentes ocorridos na nossa realidade, tais problemas chegam a envolver a teoria dos grafos e várias áreas da matemática discreta para obtenção de seus resultados.

Segundo OLIVEIRA (2000), a *pesquisa aplicada* requer várias teorias ou leis mais amplas como um ponto de partida, tendo como objetivo pesquisar, comprovar, induzir ou rejeitar hipóteses sugeridas pelos modelos teóricos e fazer sua aplicação as nossas diferentes humanas.

É correto dizer que uma das principais dificuldades da compreensão de problemas que envolvem grafos é dada pela abstração específica de cada problema. Todo problema existente possui suas particularidades e é através dessa grande variedade de características implícitas que será abordada nesse projeto a *Pesquisa Exploratória*.

A área de Teoria dos Grafos proporciona uma consistência de informações sobre as mais variadas áreas, tais como sistemas operacionais de computadores, chips, rotas de veículos, etc. Enfim, necessitamos da Teoria dos Grafos para delimitarmos e explicarmos a grande relevância desses problemas que serão investigados.

CERVO (1996) fala que a *Pesquisa Exploratória* é responsável em demonstrar maiores informações sobre vários assuntos; Facilita a parametrização de um tema de trabalho; Determina os objetivos ou formula as hipóteses de uma pesquisa ou ainda descobre novo tipo de enfoque para o trabalho que se esta imaginando. Através dessas pesquisas exploratórias é que se avalia a possibilidade de desenvolver uma boa pesquisa sobre determinado assunto.

Portanto, a investigação da Teoria dos Grafos só se tornará possível mediante de uma gama de conhecimentos extraídos de livros, revistas, artigos e outros meios relacionados a formas de pesquisas.

ANDRADE (1998) afirma que, a *Pesquisa Bibliográfica* procura explicar um problema a partir de tais referências teóricas publicadas em documentos. A *Pesquisa Bibliográfica* poderá acontecer independentemente ou fazer parte da pesquisa descritiva ou experimental. Em ambos os casos, ela procura conhecer e analisar as contribuições culturais ou científicas do passado existentes sobre um determinado assunto, tema ou problema.

4 ANÁLISE DE RESULTADOS

Este projeto atingiu todo o seu propósito firmado, uma vez que uma ferramenta base extensível foi implementada, testada e que certamente poderá ser usada como um ponto de apoio ao estudo e incentivo à investigação da teoria dos grafos, existindo toda a possibilidade de um aprimoramento futuro. A escolha da linguagem e do padrão de armazenamento dos grafos, neste caso, à Programação Orientada a Objetos e (XML - eXtensible Markup Language), foram de grande relevância o qual resultou numa arquitetura aberta possibilitando o desenvolvimento distribuído por equipes de diferentes níveis de conhecimento na área de grafos e da ciência da computação.

5 CONCLUSÃO

Este projeto contribuiu para uma investigação de alto nível, disseminando uma gama de novos conhecimentos em distintas áreas, tais como: Metodologias de ensino, Meios de aprendizagem, Software Educacional, Distribuição de dados na Internet e Programação Orientada a Objetos. Desta forma, se for bem direcionado, este projeto pode vir a ser um embrião de uma ferramenta Open Source (Código Aberto) através de uma cooperação entre seus desenvolvedores e a instituição de ensino, formalizando em conjunto uma pesquisa de excelência.

CANTO, Marco. Dominando o Delphi 2. São Paulo: MAKRON Books, 1996.

CERVO, A. L.; BERVAN, P. A. Metodologia Científica. 4^{ta} Ed. São Paulo: MAKRON Books, 1996.

FERREIRA, C. E. Combinatória e Grafos. Disponível em: http://www.ima.usp.br/~procomb/proj_submetido/node5.html. Acesso em: 05/04/2003.

GOLDBARG, M. C.; LUNA, H. P. L. Otimização Combinatória e Programação Linear. Rio de Janeiro: Campus, 2000.

HENRIQUES, A. P. M. Teoria dos Grafos e Sistema de Informação Geográfica: Uma alternativa de integração. 2000. Dissertação (Mestrado em Sistema de Computação) – Universidade Federal do Rio Grande do Norte, Natal.

LEITE, M.; RAHAL JÚNIOR, N. S. Programação Orientada ao objeto: uma abordagem didática. Disponível em: http://www.revista.unicamp.br/infocce/artigos/leite_rahal.html. Acesso em: 05/04/2003.

LINGUAGENS ORIENTADAS A OBJETOS. Disponível em: http://www.webinformation.hpg.ig.com.br/prog_OOP.htm. Acesso em: 05/04/2003.

LISTA DV-DELPHI. Disponível em: http://www.geocities.com/vv_delphi/manuals.htm. Acesso em: 03/08/2003.

OLIVEIRA, S. L. Tratado de Metodologia Científica. São Paulo: Pioneira, 1997.

SANTANA, Gladiston. O que é Delphi e qual a sua diferença quando comparado a outros RADs. Disponível em: http://www.portaldeinformatica.com.br/artigos_delphi_o_q_eh.htm. Acesso em: 03/08/2003.

6 REFERÊNCIAS BIBLIOGRÁFICAS

ANDRADE, M. M. **Introdução à Metodologia do Trabalho Científico**. 3ªed. São Paulo: Atlas, 1998.

A TEORIA dos grafos aplicada na computação. Disponível em: <<http://www.icmc.sc.usp.br/~grad/SCE/sce127.html>>. Acesso em: 05/04/2003.

BOAVENTURA NETTO, P. O. **Grafos: Teoria, Modelos, Algoritmos**. São Paulo: Edgard Blücher, 1996.

_____. **Grafos: Teoria, Modelos, Algoritmos**. 2ª Ed. São Paulo: Edgard Blücher, 2001.

CANTÚ, Marco. **Dominando o Delphi 2**. São Paulo: MAKRON Books, 1996.

CERVO, A. L.; BERVAN, P. A. **Metodologia Científica**. 4ªed. São Paulo: MAKRON Books, 1996.

FERREIRA, C. E. **Combinatória e Grafos**. Disponível em: <http://www.ime.usp.br/~procomb/proj_submetido/node6.html>. Acesso em: 05/04/2003.

GOLDBARG, M. C.; LUNA, H. P. L. **Otimização Combinatória e Programação Linear**. Rio de Janeiro: Campus, 2000.

HENRIQUES, A. P. M. **Teoria dos Grafos e Sistema de Informação Geográfica: Uma alternativa de integracao**. 2000. Dissertação (Mestrado em Sistema de Computação) – Universidade Federal do Rio Grande do Norte, Natal.

LEITE, M.; RAHAL JÚNIOR, N. S. **Programação Orientada ao objeto: uma abordagem didática**. Disponível em: <http://www.revista.unicamp.br/infotec/artigos/leite_rahall.html>. Acesso em: 05/04/2003.

LINGUAGENS ORIENTADAS A OBJETOS. Disponível em: <http://www.webinformation.hpg.ig.com.br/prog_OOP.htm>. Acesso em: 05/04/2003.

LISTA DV-DELPHI. Disponível em: <http://www.geocities.com/vv_delphi/manuais.htm>. Acesso em: 03/08/2003.

OLIVEIRA, S. L. **Tratado de Metodologia Científica**. São Paulo: Pioneira, 1997.

SANTANA, Gladiston. **O que é Delphi e qual a sua diferença quando comparado a outros RADs**. Disponível em: <http://www.portaldeinformatica.com.br/artigos_delphi_o_q_eh.htm> Acesso em: 03/08/2003.

TEIXEIRA, J. F. **Uma discussão sobre a classificação de software educacional.** Disponível em: < <http://www.revista.unicamp.br/infotec/artigos/jacqueline.html>>. Acesso em: 25/08/2003.