

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

**ANÁLISE E IMPLEMENTAÇÃO DE UM SISTEMA DE OFERTAS DE ESTÁGIOS E  
CURRÍCULOS PARA A WEB**

**ANÁLISE E IMPLEMENTAÇÃO DE UM SISTEMA DE OFERTAS DE ESTÁGIOS E  
CURRÍCULOS PARA WEB**

Trabalho apresentado ao curso de  
Graduação em Sistemas de Informação da  
Faculdade Natalense para o Desenvolvimento  
do Rio Grande do Norte para obtenção do grau  
de Bacharel

**José Wilton Pinheiro Galvão Junior**

Orientador:

Felipe Araújo Aleixo, MSc.

**NATAL/RN**

**2003**

**JOSÉ WILTON PINHEIRO GALVÃO JUNIOR**

FACULDADE NATALENSE PARA O DESENVOLVIMENTO DO RIO GRANDE DO  
NORTE

BACHARELADO EM SISTEMAS DE INFORMAÇÃO

ANÁLISE E IMPLEMENTAÇÃO DE UM SISTEMA DE OFERTAS DE ESTÁGIOS E  
CURRÍCULOS PARA A WEB

**ANÁLISE E IMPLEMENTAÇÃO DE UM SISTEMA DE OFERTAS DE ESTÁGIOS E  
CURRÍCULOS PARA A WEB**

Monografia apresentada e aprovada em \_\_\_\_\_ de \_\_\_\_\_ de 2003 pela banca  
examinadora composta pelos seguintes membros:

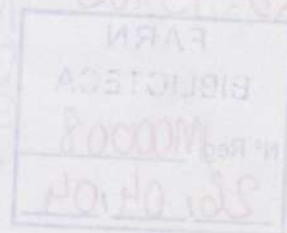
  
Felipe Araújo Aleixo, MsC.  
Monografia apresentada ao curso de  
Graduação em Sistemas de Informação da  
Faculdade Natalense para o Desenvolvimento  
do Rio Grande do Norte para obtenção do grau  
de Bacharel em Sistemas de Informação.

**Orientador:**

Felipe Araújo Aleixo, MsC.

**NATAL/RN**

**2003**



Galvão Junior, José Wilton Pinheiro

Análise e implementação de um sistema de ofertas de estágios e currículos para a web / José Wilton Pinheiro Galvão Junior. \_\_\_\_ . Natal: FARN, 2003.  
90 p.

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

Orientador: Fellipe Araújo Aleixo, MsC.

1. Sistema de gerenciamento de dados – Monografia
2. Internet – Monografia
3. Banco de currículo – Monografia

RN/FARN/BC

CDU – 004.65



M  
004.65  
9171a  
2003

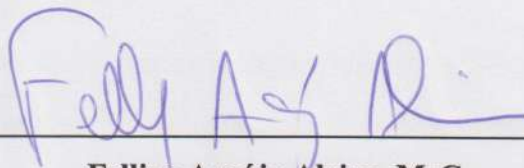
**LIGA DE ENSINO DO RIO GRANDE DO NORTE**  
**FACULDADE NATALENSE PARA O DESENVOLVIMENTO DO RIO GRANDE DO**  
**NORTE**

**BACHARELADO EM SISTEMAS DE INFORMAÇÃO**

**ANÁLISE E IMPLEMENTAÇÃO DE UM SISTEMA DE OFERTAS DE ESTÁGIOS E**  
**CURRÍCULOS PARA A WEB**

**José Wilton Pinheiro Galvão Junior**

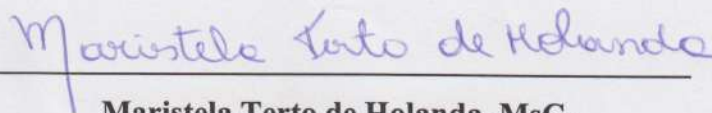
Monografia apresentada e aprovada em \_\_\_\_ de \_\_\_\_ de 2003 pela banca  
examinadora composta pelos seguintes membros:



**Fellipe Araújo Aleixo, MsC.**  
Orientador



**Antônio de Pádua de Miranda Henriques, MsC.**  
Membro examinador



**Maristela Terto de Holanda, MsC.**  
Membro examinador

Natal, 10 de dezembro de 2003

## AGRADECIMENTOS

Agradeço primeiramente a Deus pelo dom da vida e aos meus pais pelo apoio e sacrifício feito para que concluísse mais essa etapa da minha vida.

Ao meu orientador Felliipe Araújo Aleixo, pela paciência e pelo incentivo dado ao meu trabalho. Aos professores Luiz Augusto Mendes Filho e Ytalo Rosendo pela ajuda durante o desenvolvimento deste trabalho. Ao professor Antônio de Pádua de Miranda Henriques que me incentivou a continuar a caminhada acadêmica.

As minhas amigas Kaline Medeiros, Cristiane Teixeira e Lara Vanessa Caldas que tornaram as noites de faculdade mais alegres.

Agradeço também a todos que de alguma forma contribuíram para a conclusão de mais essa etapa da minha vida.

## RESUMO

Cada vez mais o mercado de trabalho exige que o profissional que se candidata a um estágio ou emprego tenha conhecimento de todas as áreas da empresa. Para isso, as empresas estão buscando jovens que tenham a capacidade de aprender e ajudar a empresa a se tornar mais competitiva. Esses jovens são, na maioria das vezes, universitários que querem uma oportunidade para ingressar no mercado e poder pôr em prática tudo que aprendem na universidade. Este trabalho apresenta a análise e implementação de um sistema de ofertas de estágios e currículos para a web que servirá como ponte direta entre os alunos e as empresas. A análise do sistema foi desenvolvida utilizando a UML, e mais especificamente os diagramas de casos de uso, de classes e de colaboração. As interfaces do sistema foram criadas em HTML. Para a implementação das classes do sistema foi utilizada a linguagem de programação Java. O banco de dados foi desenvolvido no Microsoft Access 2000. Além de fazer uso de JSP para criação dos scripts, responsáveis pela interação entre as interfaces e as classes que farão a conexão com o banco de dados do sistema.

**Palavras-chave:** programação web, currículos, orientação a objetos, Java.

## ABSTRACT

Each time more the work market demands that the professional who if candidate to a period of training or job has knowledge of all the areas of the company. For this, the companies are searching young that they have the capacity to learn and to help the company if to become more competitive. These young ones are, in the majority of the times, colleges student who want a chance to enter the market and power to put in practical everything that learn in the university. This work presents the analysis and implementation of a system of offers of periods of training and resumes for web that it will serve as direct bridge between the pupils and the companies. The analysis of the system was developed using the UML, and more specifically the diagrams use-cases, class and contribution. The interfaces of the system had been created in HTML. For the implementation of the class of the system the programming language Java was used. The data base was developed in the Microsoft Access 2000. Besides making use of JSP for creation of scripts, responsible for the interaction between the interfaces and the class that will make the connection the data base of the system.

**Key-words:** programming web, resumes, objects orientation, Java.

|                                                  |    |
|--------------------------------------------------|----|
| 2.1.2 Polimorfismo                               | 18 |
| 2.2 UML (UNITED MODELING LANGUAGE)               | 19 |
| 2.2.1 Diagramas de Casos de Uso                  | 20 |
| 2.2.2 Diagramas de Classes                       | 21 |
| 2.2.3 Diagramas de Colaboração                   | 22 |
| 3 JAVA                                           | 23 |
| 3.1 JAVABEANS                                    | 25 |
| 3.2 JAVA DATABASE CONNECTIVITY (JDBC)            | 26 |
| 3.3 SERVLETS                                     | 27 |
| 3.4 JSP SERVER PAGES                             | 28 |
| 4 HTML                                           | 30 |
| 4.1 Formulários                                  | 32 |
| 5 INSTRUMENTAS UTILIZADAS PARA O DESENVOLVIMENTO | 33 |
| 5.1 Model-Gen for UML 1.2                        | 33 |
| 5.2 Microsoft FrontPage 2000                     | 33 |
| 5.3 Microsoft Access 2000                        | 33 |
| 5.4 Microsoft IIS                                | 34 |

## SUMÁRIO

|                                                           |           |
|-----------------------------------------------------------|-----------|
| <b>1. INTRODUÇÃO .....</b>                                | <b>12</b> |
| 1.1. PROBLEMA DE PESQUISA .....                           | 12        |
| 1.2. OBJETIVOS .....                                      | 13        |
| 1.3. JUSTIFICATIVA .....                                  | 13        |
| 1.4. ESTRUTURA DO TRABALHO .....                          | 14        |
| <b>2. REFERENCIAL TEÓRICO .....</b>                       | <b>15</b> |
| 2.1. ORIENTAÇÃO A OBJETOS .....                           | 15        |
| 2.1.1. Abstração .....                                    | 15        |
| 2.1.2. Encapsulamento .....                               | 16        |
| 2.1.3. Objetos .....                                      | 16        |
| 2.1.4. Atributos e Métodos .....                          | 16        |
| 2.1.5. Classe .....                                       | 17        |
| 2.1.6. Herança .....                                      | 17        |
| 2.1.7. Polimorfismo .....                                 | 18        |
| 2.1.8. Generalização .....                                | 19        |
| 2.2. UML (UNIFIED MODELING LANGUAGE) .....                | 19        |
| 2.2.1. Diagramas de Casos de Uso .....                    | 20        |
| 2.2.2. Diagramas de Classes .....                         | 21        |
| 2.2.3. Diagramas de Colaboração .....                     | 22        |
| 2.3. JAVA .....                                           | 23        |
| 2.4. JAVABEANS .....                                      | 25        |
| 2.5. JAVA DATABASE CONNECTIVITY (JDBC) .....              | 26        |
| 2.6. SERVLETS .....                                       | 27        |
| 2.7. JAVASERVER PAGES .....                               | 28        |
| 2.8. SQL .....                                            | 30        |
| 2.9. HTML .....                                           | 31        |
| 2.9.1. Formulários .....                                  | 32        |
| 2.10. FERRAMENTAS UTILIZADAS PARA O DESENVOLVIMENTO ..... | 33        |
| 2.10.1. Poseidon for UML 1.2 .....                        | 33        |
| 2.10.2. Microsoft FrontPage 2000 .....                    | 33        |
| 2.10.3. Microsoft Access 2000 .....                       | 34        |
| 2.10.4. JCreator LE .....                                 | 34        |

|                                                              |           |
|--------------------------------------------------------------|-----------|
| 2.10.5. Resin 2.1.5 .....                                    | 34        |
| <b>3. METODOLOGIA.....</b>                                   | <b>35</b> |
| <b>4. ANÁLISE E IMPLEMENTAÇÃO DO SISTEMA.....</b>            | <b>36</b> |
| 4.1. ANÁLISE DO SISTEMA DE ESTÁGIOS E CURRÍCULOS.....        | 36        |
| 4.1.1. Análise de Requisitos .....                           | 37        |
| 4.1.2. Diagramas de Casos de Uso .....                       | 39        |
| 4.1.3. Diagrama Conceitual .....                             | 41        |
| 4.1.4. Diagramas de Colaboração e Cenários.....              | 41        |
| 4.1.5. Diagramas de Classes .....                            | 52        |
| 4.2. IMPLEMENTAÇÃO DO SISTEMA DE ESTÁGIOS E CURRÍCULOS ..... | 54        |
| 4.2.1. Interface do Sistema .....                            | 55        |
| 4.2.1.1. Módulo administrador .....                          | 55        |
| 4.2.1.1.1. Login.....                                        | 55        |
| 4.2.1.1.2. Cadastrar aluno .....                             | 56        |
| 4.2.1.1.3. Atualizar aluno .....                             | 58        |
| 4.2.1.2. Módulo aluno.....                                   | 60        |
| 4.2.1.2.1. Login.....                                        | 61        |
| 4.2.1.2.2. Cadastrar currículo .....                         | 63        |
| 4.2.1.2.3. Atualizar currículo .....                         | 66        |
| 4.2.1.2.4. Pesquisar ofertas de estágios .....               | 67        |
| 4.2.1.3. Módulo empresa .....                                | 69        |
| 4.2.1.3.1. Login.....                                        | 70        |
| 4.2.1.3.2. Cadastrar empresa.....                            | 71        |
| 4.2.1.3.3. Cadastrar oferta.....                             | 72        |
| 4.2.1.3.4. Pesquisar currículos.....                         | 74        |
| 4.2.1.3.5. Atualizar cadastro .....                          | 75        |
| 4.2.1.3.6. Atualizar oferta .....                            | 78        |
| 4.2.1.3.7. Excluir oferta .....                              | 81        |
| 4.2.2. Modelo de Dados.....                                  | 83        |
| 4.2.3. WAR (Web Application).....                            | 84        |
| <b>5. CONCLUSÃO.....</b>                                     | <b>85</b> |
| <b>REFERÊNCIAS BIBLIOGRÁFICAS .....</b>                      | <b>86</b> |
| <b>BIBLIOGRAFIA CONSULTADA .....</b>                         | <b>88</b> |
| <b>ANEXO A – QUESTIONÁRIO .....</b>                          | <b>90</b> |

## LISTA DE FIGURAS

|                                                                                        |    |
|----------------------------------------------------------------------------------------|----|
| FIGURA 1 – Exemplo de herança simples .....                                            | 17 |
| FIGURA 2 – Exemplo de herança múltipla .....                                           | 18 |
| FIGURA 3 – Polimorfismo .....                                                          | 18 |
| FIGURA 4 – Representação da notação utilizada para casos de uso .....                  | 20 |
| FIGURA 5 – Caso de uso expandido – cenário.....                                        | 21 |
| FIGURA 6 – Notação UML para representação de uma classe .....                          | 22 |
| FIGURA 7 – Diagrama de classes.....                                                    | 22 |
| FIGURA 8 – Diagrama de colaboração .....                                               | 23 |
| FIGURA 9 – Exemplo de um JavaBean.....                                                 | 26 |
| FIGURA 10 – Arquitetura JDBC .....                                                     | 27 |
| FIGURA 11 – Suporte de recursos JSP para a separação de apresentação e implementação.. | 30 |
| FIGURA 12 – Sintaxe básica das instruções SQL .....                                    | 31 |
| FIGURA 13 – Formato genérico de uma tag .....                                          | 32 |
| FIGURA 14 – Estrutura de um documento HTML.....                                        | 32 |
| FIGURA 15 – Sintaxe de um formulário em HTML .....                                     | 33 |
| FIGURA 16 – Estrutura de uma tag INPUT .....                                           | 33 |
| FIGURA 17 – Dados a serem disponibilizados no currículo do aluno .....                 | 38 |
| FIGURA 18 – Operações disponíveis no sistema .....                                     | 39 |
| FIGURA 19 – Diagrama de casos de uso do módulo empresa .....                           | 39 |
| FIGURA 20 – Diagrama de casos de uso do módulo aluno.....                              | 40 |
| FIGURA 21 – Diagrama de casos de uso do módulo administrador .....                     | 40 |
| FIGURA 22 – Modelo conceitual do sistema de estágios e currículos .....                | 41 |
| FIGURA 23 – Cenário “cadastrar aluno” .....                                            | 42 |
| FIGURA 24 – Diagrama de colaboração “cadastrar aluno” .....                            | 42 |
| FIGURA 25 – Cenário “logar” .....                                                      | 43 |
| FIGURA 26 – Diagrama de colaboração “logar” .....                                      | 43 |
| FIGURA 27 – Cenário “atualizar currículo” .....                                        | 44 |
| FIGURA 28 – Diagrama de colaboração “atualizar currículo” .....                        | 44 |
| FIGURA 29 – Cenário “excluir currículo” .....                                          | 45 |
| FIGURA 30 – Diagrama de colaboração “excluir currículo” .....                          | 45 |
| FIGURA 31 – Cenário “pesquisar ofertas” .....                                          | 46 |
| FIGURA 32 – Diagrama de colaboração “pesquisar ofertas por curso” .....                | 46 |

|                                                                                      |    |
|--------------------------------------------------------------------------------------|----|
| FIGURA 33 – Diagrama de colaboração “pesquisar ofertas por área” .....               | 46 |
| FIGURA 34 – Diagrama de colaboração “pesquisar ofertas por curso e período” .....    | 47 |
| FIGURA 35 – Cenário “cadastrar empresa” .....                                        | 47 |
| FIGURA 36 – Diagrama de colaboração “cadastrar empresa” .....                        | 47 |
| FIGURA 37 – Cenário “cadastrar ofertas de estágios” .....                            | 48 |
| FIGURA 38 – Diagrama de colaboração “cadastrar ofertas de estágios” .....            | 48 |
| FIGURA 39 – Cenário “atualizar oferta” .....                                         | 48 |
| FIGURA 40 – Diagrama de colaboração “atualizar oferta” .....                         | 49 |
| FIGURA 41 – Cenário “excluir oferta” .....                                           | 49 |
| FIGURA 42 – Diagrama de colaboração “excluir oferta” .....                           | 49 |
| FIGURA 43 – Cenário “atualizar cadastro” .....                                       | 50 |
| FIGURA 44 – Diagrama de colaboração “atualizar cadastro de aluno” .....              | 50 |
| FIGURA 45 – Cenário “cadastrar aluno” .....                                          | 50 |
| FIGURA 46 – Diagrama de colaboração “cadastrar aluno” .....                          | 51 |
| FIGURA 47 – Cenário “pesquisar currículos” .....                                     | 51 |
| FIGURA 48 – Diagrama de colaboração “pesquisar currículos por curso” .....           | 52 |
| FIGURA 49 – Diagrama de colaboração “pesquisar currículos por área” .....            | 52 |
| FIGURA 50 – Diagrama de colaboração “pesquisar currículos por curso e período” ..... | 52 |
| FIGURA 51 – Diagrama de classes do sistema de estágios e currículos .....            | 53 |
| FIGURA 52 – Visão modular do sistema de estágios e currículos .....                  | 54 |
| FIGURA 53 – Tela principal do módulo administrador.....                              | 55 |
| FIGURA 54 – Seqüência de operações para login do administrador.....                  | 55 |
| FIGURA 55 – Servlet loginAdministrador.jsp.....                                      | 56 |
| FIGURA 56 – Método autenticarAdministrador.....                                      | 56 |
| FIGURA 57 – Seqüência de operações da opção cadastrar aluno .....                    | 57 |
| FIGURA 58 – Método cadastrarAluno .....                                              | 57 |
| FIGURA 59 – Servlet cadastrarAluno.jsp.....                                          | 58 |
| FIGURA 60 – Seqüência de operações da opção atualizar aluno .....                    | 59 |
| FIGURA 61 – Servlet buscarAluno.jsp.....                                             | 59 |
| FIGURA 62 – Método pesqAlunoMatricula.....                                           | 60 |
| FIGURA 63 – Servlet atualizarAluno.jsp.....                                          | 60 |
| FIGURA 64 – Método atualizarAluno .....                                              | 60 |
| FIGURA 65 – Tela principal da opção atualizar aluno .....                            | 60 |
| FIGURA 66 – Tela principal do módulo aluno.....                                      | 61 |

|                                                                       |    |
|-----------------------------------------------------------------------|----|
| FIGURA 67 – Seqüência de operações da opção login de aluno .....      | 62 |
| FIGURA 68 – Servlet loginCurriculo.jsp.....                           | 62 |
| FIGURA 69 – Método autenticarCurriculo.....                           | 62 |
| FIGURA 70 – Seqüência de operações da opção cadastrar currículo ..... | 63 |
| FIGURA 71 – loginAluno.htm.....                                       | 64 |
| FIGURA 72 – Servlet loginAluno.jsp .....                              | 64 |
| FIGURA 73 – Método autenticarAluno .....                              | 65 |
| FIGURA 74 – Métodos de inserção de dados do currículo.....            | 65 |
| FIGURA 75 – cadastrarCurriculo.htm .....                              | 66 |
| FIGURA 76 – Seqüência de operações da opção atualizar currículo.....  | 66 |
| FIGURA 77 – Seqüência de operações da opção pesquisar ofertas.....    | 67 |
| FIGURA 78 – Método pesqOfertasCurso .....                             | 68 |
| FIGURA 79 – Método pesqOfertasArea .....                              | 68 |
| FIGURA 80 – Método pesqOfertasCursoPeriodo.....                       | 69 |
| FIGURA 81 – Servlet pesqOfertasCursoPeriodo.....                      | 69 |
| FIGURA 82 – Tela principal do módulo empresa .....                    | 70 |
| FIGURA 83 – Servlet loginEmpresa.jsp .....                            | 70 |
| FIGURA 84 – Método autenticarEmpresa .....                            | 71 |
| FIGURA 85 – Esquema de operações da opção cadastrar empresa.....      | 71 |
| FIGURA 86 – Servlet cadastrarEmpresa.jsp.....                         | 72 |
| FIGURA 87 – Método cadastrarEmpresa .....                             | 72 |
| FIGURA 88 – Esquema de operações da opção cadastrar oferta.....       | 73 |
| FIGURA 89 – Servlet cadastrarOferta.jsp.....                          | 73 |
| FIGURA 90 – Método cadastrarOferta .....                              | 73 |
| FIGURA 91 – Método pesqCodigoOferta.....                              | 74 |
| FIGURA 92 – Esquema de operações da opção pesquisar currículos .....  | 74 |
| FIGURA 93 – Método pesqCurriculoCurso.....                            | 75 |
| FIGURA 94 – Servlet pesqCurriculoCurso.jsp .....                      | 75 |
| FIGURA 95 – Esquema de operações da opção atualizar cadastro .....    | 76 |
| FIGURA 96 – Servlet buscarEmpresaLogin.jsp .....                      | 76 |
| FIGURA 97 – Método pesqEmpresaLoginSenha .....                        | 77 |
| FIGURA 98 – Servlet atualizarEmpresa.jsp.....                         | 77 |
| FIGURA 99 – Método atualizarEmpresa .....                             | 78 |
| FIGURA 100 – Seqüência de operações da opção atualizar oferta.....    | 79 |

|                                                                        |    |
|------------------------------------------------------------------------|----|
| FIGURA 101 – Servlet pesqOfertaCodigoAtualizar.jsp .....               | 79 |
| FIGURA 102 – Método pesqOfertaCodigo.....                              | 80 |
| FIGURA 103 – Servlet atualizarOferta.jsp .....                         | 80 |
| FIGURA 104 – Método atualizarOferta.....                               | 81 |
| FIGURA 105 – Seqüência de operações da opção excluir oferta .....      | 81 |
| FIGURA 106 – pesqOfertaCodigoExcluir.jsp .....                         | 82 |
| FIGURA 107 – excluirOferta.jsp .....                                   | 83 |
| FIGURA 108 – Método excluirOferta.....                                 | 83 |
| FIGURA 109 – Modelo lógico do banco de dados .....                     | 83 |
| FIGURA 110 – Estrutura do WAR do sistema de estágios e currículos..... | 84 |

empresas que servem de intermediária entre a empresa que deseja contratar estagiários e os estudantes, e com contato direto com a faculdade, o que muitas vezes não é eficaz, pois nem todos os alunos têm acesso a essas informações.

Um meio de popularizar a divulgação dessas informações é disponibilizá-las através da Internet.

A Internet hoje funciona como o maior e mais rápido meio de comunicação, além de possibilitar acesso em qualquer parte onde esteja conectado o usuário, tendo um custo relativamente baixo. O que possibilita ao usuário rapidez e facilidade no acesso as informações disponíveis nesse meio.

Este trabalho propõe a análise e implementação de um protótipo de um sistema de disponibilização de ofertas de estágios e currículos para o site da Faculdade Natalense para o Desenvolvimento do Rio Grande do Norte – FARN. Este sistema estará disponível para que empresas possam publicar ofertas de estágios para alunos desta instituição, e também para que os alunos disponibilizem seus currículos a fim de buscar oportunidades de estágios.

## 2. O PROBLEMA DE PESQUISA

Uma das grandes dificuldades encontradas pelos alunos da FARN para encontrar estágios é a falta de divulgação das ofertas existentes no mercado. Muitas vezes o estudante tem que se deslocar até agências de estágios para ter acesso a essas informações, já que atualmente não existem sites que disponibilizem ofertas de estágios em Natal.

Em alguns casos as empresas interessadas entram em contato com a faculdade e expõem suas ofertas, mas a forma de divulgação não é suficiente. Isso se dá através de avisos fixados nos corredores da faculdade, não havendo uma centralização da informação.

## 1. INTRODUÇÃO

Cada vez mais o mercado de trabalho exige que o empregado tenha conhecimento em todas as áreas da empresa. Por isso, as empresas estão buscando jovens que tenham a capacidade de aprender e ajudar a empresa a se tornar mais competitiva.

Segundo o Centro de Integração Empresa Escola, 76% desses jovens são universitários que possuem interesse em colocar em prática todo o conhecimento adquirido na universidade.

Os métodos mais comuns de buscar esses jovens são: através de agências de estágios, como Instituto Euvaldo Lodi - IEL, Centro de Integração Empresa Escola - CIEE, que são empresas que servem de intermédio entre a empresa que deseja contratar estagiários e os estudantes, e com contato direto com a faculdade, o que muitas vezes não é eficaz, pois nem todos os alunos têm acesso a essas informações.

Um meio de popularizar a divulgação dessas informações é disponibilizá-las através da Internet.

A Internet hoje funciona como o maior e mais rápido meio de comunicação, além de possibilitar acesso em qualquer parte onde esteja conectado o usuário, tendo um custo relativamente baixo. O que possibilita ao usuário rapidez e facilidade no acesso as informações dispostas nesse meio.

Este trabalho propõe a análise e implementação de um protótipo de um sistema de disponibilização de ofertas de estágios e currículos para o site da Faculdade Natalense para o Desenvolvimento do Rio Grande do Norte – FARN. Este sistema estará disponível para que empresas possam publicar ofertas de estágios para alunos desta Instituição, e também para que esses alunos disponibilizem seus currículos a fim de buscar oportunidades de estágios.

### 1.1. PROBLEMA DE PESQUISA

Uma das grandes dificuldades encontradas pelos alunos da FARN para encontrar estágios é a falta de divulgação das ofertas existentes no mercado. Muitas vezes o estudante tem que se deslocar até agências de estágios para ter acesso a essas informações, já que praticamente não existem sites que disponibilizem ofertas de estágios em Natal.

Em alguns casos as empresas interessadas entram em contato com a faculdade e expõe suas ofertas, mas a forma de divulgação não é suficiente. Esta se dá através de avisos fixados nos corredores da faculdade, não havendo uma centralização da informação.

Como centralizar as ofertas de estágios propostas e tornar sua divulgação eficiente de forma que todos os alunos da FARN tenham acesso a estas? Como possibilitar às empresas acesso aos alunos que possuem interesse em estagiar?

## 1.2. OBJETIVOS

O objetivo geral deste trabalho trata-se da implementação de um sistema de ofertas de estágios e currículos voltado para a web.

Este trabalho foi dividido nos seguintes objetivos específicos:

- Realização de uma pesquisa sobre análise e projeto orientado a objetos;
- Realização de um estudo sobre a linguagem Java a fim de utilizar suas ferramentas para acesso a banco de dados (JDBC) e para a construção de componentes (JavaBeans);
- Realização de uma pesquisa sobre Java Server Pages – JSP;
- Levantamento de dados para análise de requisitos;
- Análise de dados para construção dos diagramas de casos de uso, de classes e de colaboração, utilizando a notação UML;
- Construção do modelo lógico de banco de dados;
- Construção das interfaces gráficas utilizando HTML;
- Construção das classes que irão manipular os dados utilizando a linguagem Java;
- Ligação das interfaces com as classes, criando conteúdo dinâmico com JSP.

## 1.3. JUSTIFICATIVA

O sistema cria uma ponte entre os alunos da FARN e as empresas interessadas em contratar estagiários que estudam nesta instituição.

Este sistema estará disponível ao público através do site da FARN, tornando possível a interação com o mesmo em qualquer lugar, facilitando o acesso às informações publicadas. Neste, o aluno poderá divulgar seu currículo, além de visualizar ofertas expostas por empresas, tornando mais fácil o acesso ao mercado de trabalho. As empresas, além de expor vagas de estágios, poderão buscar profissionais que melhor se encaixem ao seu perfil, através da análise dos currículos cadastrados pelos alunos.

Dessa forma, haverá uma maior proximidade entre empresas e alunos, contribuindo tanto para a empresa que poderá fazer uma seleção detalhada, adquirindo dessa forma o

melhor profissional, tanto para o aluno que terá acesso a todas as oportunidades que o mercado de trabalho oferece.

## 2. ORIENTAÇÃO A OBJETOS

### 1.4. ESTRUTURA DO TRABALHO

Este trabalho está dividido em cinco capítulos básicos onde serão abordados os itens: introdução, referencial teórico, metodologia, análise de resultados que corresponde à descrição da análise e a implementação do sistema e a conclusão.

No primeiro capítulo, descreve-se a introdução do trabalho dando uma visão geral do trabalho, além do problema que este propõe e os objetivos atingidos pelo mesmo.

O segundo capítulo, referencial teórico, mostra todas as tecnologias e ferramentas que foram utilizadas para a criação deste sistema, dando uma visão geral sobre estas.

O terceiro capítulo, metodologia, descreve todos os passos referentes a construção do trabalho, além da classificação do trabalho seguindo os tipos de pesquisa utilizados.

No quarto capítulo, “análise e implementação do sistema”, estão descritos todos os passos e as especificações da construção do sistema, começando pela idéia do trabalho, passando pela análise de requisitos, construção da análise orientada a objetos com seus diagramas, descrição das interfaces e das operações que o mesmo propõe, finalizando com o modelo lógico do banco de dados.

A conclusão relata a idéia principal, os objetivos alcançados e as perspectivas futuras para a continuação do trabalho.

## 2. REFERENCIAL TEÓRICO

### 2.1. ORIENTAÇÃO A OBJETOS

Segundo Lócus (2003), orientação a objetos é o estado da arte para o desenvolvimento de sistemas e aplicações em geral. Sua principal característica é a de permitir uma grande produtividade de código e de garantir uma interface homogênea do desenvolvimento de sistemas.

A proposta da orientação a objetos é representar o mais fielmente as situações do mundo real nos sistemas computacionais.

Para Tafner e Correia (2001, p.7),

a orientação a objetos consiste em conceber um sistema computacional como um todo orgânico formado por objetos que se relacionam entre si. Esse enfoque pode ser aplicado tanto à análise de sistemas quanto à programação e essa é uma das principais vantagens da orientação a objetos: a mesma metodologia serve tanto para a definição lógica quanto para a sua implementação.

O processo de análise consiste em identificar quais objetos existem dentro do universo que se pretende automatizar, e quais são os atributos e ações desses objetos.

A programação orientada a objetos consiste em utilizar estruturas de dados que simulem o comportamento dos objetos, facilitando a programação pelo uso de uma metodologia unificada para a análise e a programação. Os mesmos objetos que o analista vier a identificar no seu diagrama podem ser implementados exatamente como foram projetados, sem necessidade de fazer uma “tradução” dos diagramas de análise para estruturas de dados e de programação como acontece na análise estruturada.

Coleman *et al.* (1996) afirma que as principais vantagens do desenvolvimento orientado a objetos são: abstração de dados, compatibilidade, flexibilidade, reutilização, extensibilidade e manutenção.

Para entender melhor a orientação a objetos, Tafner e Correia (2001) definem seis princípios básicos para a mesma: abstração, objetos, encapsulamento, atributos e métodos, classe e herança. Page-Jones (2001) acrescenta a esses princípios os conceitos de polimorfismo e generalização.

### 2.1.1. Abstração

O termo abstração, aplicado ao desenvolvimento de sistemas, significa a grosso modo que só deve ser representado aquilo que vai ser usado. (TAFNER e CORREIA, 2001)

Pelo princípio da abstração, nós isolamos os objetos que queremos representar do ambiente complexo em que se situam, e nesses objetos representamos somente as características que são relevantes para o problema em questão.

### 2.1.2. Encapsulamento

Segundo Page-Jones (2001, p. 9), “encapsulamento é o agrupamento de idéias afins em uma unidade, conceito esse que pode então ser informado em uma só palavra”.

Para Tafner e Correia (2001), o encapsulamento é um dos grandes trunfos da programação orientada a objetos em relação à programação estruturada. A vantagem é que o encapsulamento disponibiliza o objeto com toda sua funcionalidade sem que haja a necessidade de conhecer o seu funcionamento interno, nem como armazena os dados requisitados.

Outro ponto importante é o fato de permitir que se faça modificações internas em um objeto, acrescentando métodos sem afetar o funcionamento dos outros componentes do sistema que utilizam esse mesmo objeto.

### 2.1.3. Objetos

Objeto é a unidade com encapsulamento que tem uma identidade única, separada de seu estado - uma instância de uma classe.

### 2.1.4. Atributos e Métodos

Pode-se dizer que os atributos dos objetos são variáveis ou campos que armazenam os diferentes valores que as características dos objetos podem conter.

Os atributos de um objeto somente mudam de valor através de estímulos externos ou internos. A única forma de modificar os atributos dos objetos é disparando eventos que provocam a transição desses estados nos objetos.

Os métodos são procedimentos ou funções que realizam as ações próprias dos objetos, Assim, os métodos são as ações que o objeto pode realizar.

#### 2.1.5. Classe

Page-Jones (2001, p. 435) define classe como “uma construção de desenho/programação que define as características (atributos e métodos) dos objetos – as instâncias dessa classe – que serão efetivamente criadas em tempo de execução”.

#### 2.1.6. Herança

Segundo Siqueira (2003), herança é um mecanismo para derivar novas classes a partir de classes existentes através de um processo de refinamento. Uma classe derivada herda a representação de dados e operações de sua classe base, mas pode seletivamente adicionar novas operações, estender a representação de dados ou redefinir a implementação de operações já existentes. Uma classe base proporciona a funcionalidade que é comum a todas as suas classes derivadas, enquanto que uma classe derivada proporciona a funcionalidade adicional que especializa o seu comportamento.

A herança pode ser simples ou múltipla. A herança é denominada simples quando uma classe herda características de apenas uma superclasse.

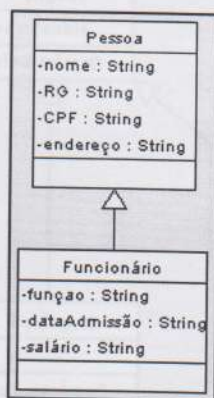


Figura 1 - Exemplo de herança simples

A herança é considerada múltipla quando uma classe herda características de duas ou mais superclasses.

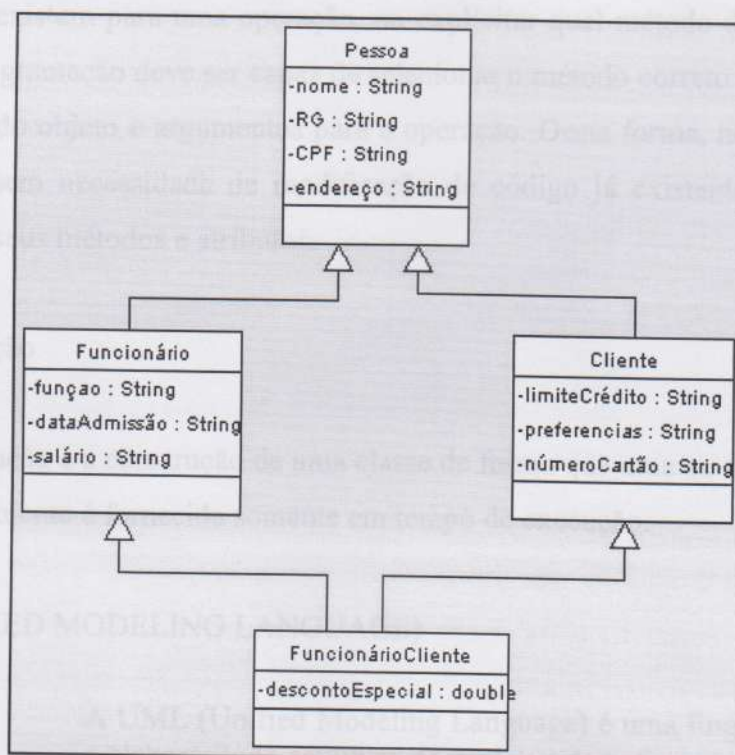


Figura 2 – Exemplo de herança múltipla

### 2.1.7. Polimorfismo

Para Page-Jones (2001, p. 39), “polimorfismo é a habilidade pela qual uma única operação ou nome de atributo pode ser definido em mais de uma classe e assumir implementações diferentes em cada uma dessas classes”.

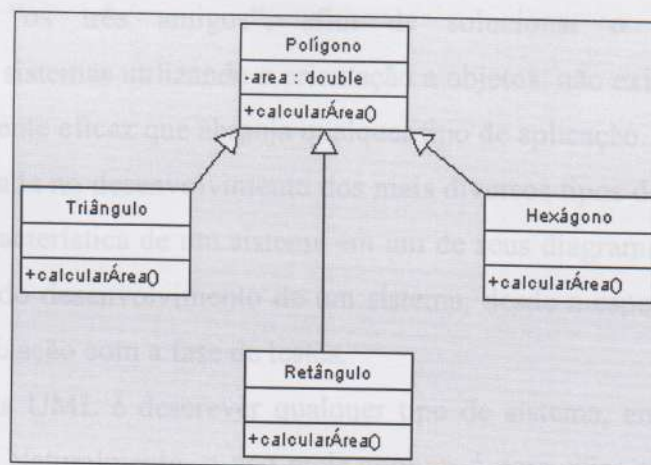


Figura 3 - Polimorfismo

Segundo Ricarte (2003), polimorfismo implica que uma operação de uma mesma classe pode ser implementada por mais de um método. O usuário não precisa saber quantas implementações existem para uma operação, ou explicitar qual método deve ser utilizado: a linguagem de programação deve ser capaz de selecionar o método correto a partir do nome da operação, classe do objeto e argumentos para a operação. Desta forma, novas classes podem ser adicionadas sem necessidade de modificação de código já existente, pois cada classe apenas define os seus métodos e atributos.

#### 2.1.8. Generalização

Generalização é a construção de uma classe de forma que uma ou mais das classes que ela utiliza internamente é fornecida somente em tempo de execução.

### 2.2. UML (UNIFIED MODELING LANGUAGE)

A UML (Unified Modeling Language) é uma linguagem-padrão para a elaboração da estrutura de projetos de software. A UML poderá ser empregada para a visualização, a especificação, a construção e a documentação de artefatos que façam uso de sistemas complexos de software. (BOOCH, RUMBAUGH e JACOBSON, 2000, p. 13)

A UML é uma tentativa de padronizar a modelagem orientada a objetos de uma forma que qualquer sistema, seja qual for o tipo, possa ser modelado corretamente, com consistência, fácil de se comunicar com outras aplicações, simples de ser atualizado e compreensível.

UML foi desenvolvida por Grady Booch, James Rumbaugh e Ivar Jacobson, que são conhecidos como “os três amigos”, afim de solucionar o grande problema do desenvolvimento de sistemas utilizando a orientação a objetos: não existência de uma notação padronizada e realmente eficaz que abranja qualquer tipo de aplicação.

A UML é usada no desenvolvimento dos mais diversos tipos de sistemas. Ela abrange sempre qualquer característica de um sistema em um de seus diagramas e é também aplicada em diferentes fases do desenvolvimento de um sistema, desde a especificação da análise de requisitos até a finalização com a fase de testes.

O objetivo da UML é descrever qualquer tipo de sistema, em termos de diagramas orientado a objetos. Naturalmente, o uso mais comum é para criar modelos de sistemas de

software, mas a UML também é usada para representar sistemas mecânicos sem nenhum software.

A UML é composta por nove tipos de diagramas: diagrama de casos de uso, de classes, de objeto, de estado, de sequência, de colaboração, de atividade, de componente e de execução. Mas, neste trabalho, só serão utilizados três tipos: diagrama de casos de uso, de classes e de colaboração.

### 2.2.1. Diagrama de Casos de Uso

Segundo Page-Jones (2001), o diagrama de casos de uso é uma interação definida e intencional entre um sistema e um ser humano, ou um ator não humano, que está desempenhando um papel específico fora do sistema.

A modelagem de um diagrama de casos de uso é uma técnica usada para descrever e definir os requisitos funcionais de um sistema. Eles são escritos em termos de atores externos, casos de uso e o sistema modelado.

Os atores representam o papel de uma entidade externa ao sistema como um usuário, um hardware, ou outro sistema que interage com o sistema modelado. Os atores iniciam a comunicação com o sistema através dos casos de uso, onde o caso de uso representa uma sequência de ações executadas pelo sistema e recebe do ator que lhe utiliza dados de um tipo ou formato já conhecido, e o valor de resposta da execução de um caso de uso também já é de um tipo conhecido, tudo isso é definido juntamente com o caso de uso através da descrição dos cenários.

A notação usada pelo diagrama de casos de uso é:

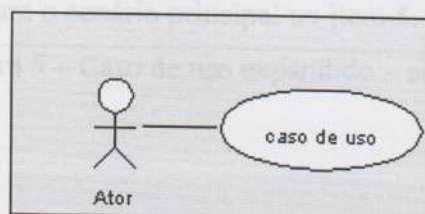


Figura 4 – Representação da notação utilizada para casos de uso

Para Cabral e Araújo (2003), o ator representa qualquer entidade que interage com o sistema. Pode ser uma pessoa, um sistema, etc. Algumas de suas características são descritas abaixo:

- Ator não é parte do sistema. Representa os papéis que o usuário do sistema pode desempenhar;
- Ator pode interagir ativamente com o sistema;

- Ator pode ser um receptor passivo de informação;
- Ator pode representar um ser humano, uma máquina ou outro sistema.

As características principais de um caso de uso são:

- Um caso de uso modela o diálogo entre atores e o sistema;
- Um caso de uso é iniciado por um ator para invocar uma certa funcionalidade do sistema;
- Um caso de uso é um fluxo de eventos completo e consistente;
- O conjunto de todos os casos de uso representa todas as situações possíveis de utilização do sistema.

O caso de uso exige que haja uma documentação para proporcionar um maior entendimento do sistema. A documentação é composta de uma descrição textual resumida e dos fluxos de eventos.

CASO DE USO: Comprar produto  
 ATOR: Cliente  
 SEQUÊNCIA NORMAL:  
 0. O cliente navega pelo catálogo e seleciona os itens a serem comprados  
 1. Cliente vai para o checkout  
 2. O cliente preenche formulário da remessa  
 3. O sistema apresenta as informações total do faturamento incluindo a remessa, os preços e impostos  
 4. O cliente preenche informações do cartão de crédito  
 5. O sistema autoriza a compra  
 6. O sistema confirma imediatamente a venda  
 7. O sistema manda a confirmação da venda por e-mail  
 SEQUÊNCIA ALTERNATIVA: Falha na autorização  
 No item 5, o sistema falha na autorização da compra por crédito. Permite ao cliente resubmeter a informação de cartão de crédito e entrar novamente.  
 SEQUÊNCIA ALTERNATIVA: Cliente regular  
 2a. O sistema mostra a informação atual da remessa. A informação de preço e os quatro últimos dígitos do cartão de crédito.  
 2b. O cliente pode aceitar ou escrever por cima destas opções por omissão. Retorna para o cenário principal no item 5.

Figura 5 – Caso de uso expandido – cenário

## 2.2.2. Diagrama de Classes

Classe é uma descrição de um grupo de objetos com atributos, comportamentos, relacionamentos com outros objetos e semântica comuns. A notação usada pela UML para representar uma classe é:

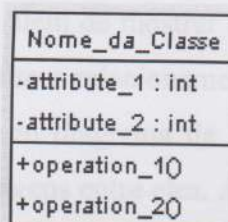


Figura 6 – Notação UML para representação de uma classe

Page-Jones (2001) define diagrama de classes como um diagrama que retrata uma série de elementos estáticos (essencialmente classes), juntamente com suas associações, estruturas de superclasse/subclasse e outros inter-relacionamentos estáticos. Opcionalmente, um diagrama desse tipo inclui, para cada classe, os seus atributos, as operações e as assinaturas das operações.

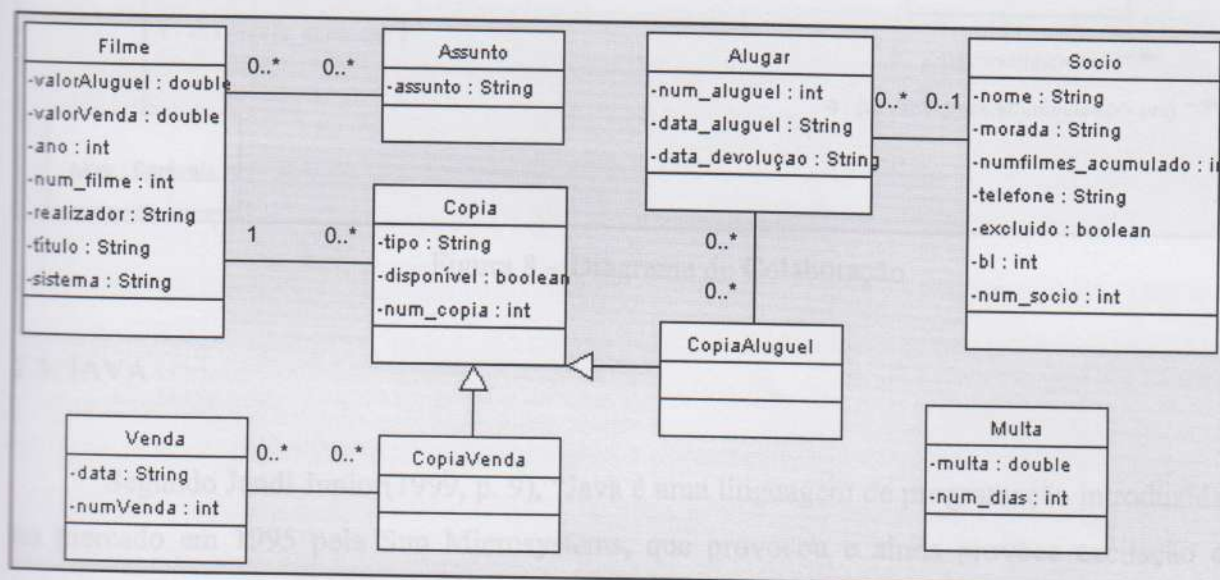


Figura 7 – Diagrama de Classes

Um sistema normalmente possui alguns diagramas de classe, já que não são todas as classes que estão inseridas em um único diagrama e uma certa classe pode participar de vários diagramas de classes – visão modular do problema.

Uma classe num diagrama pode ser diretamente implementada utilizando-se uma linguagem de programação orientada a objetos que tenha suporte direto para construção de classes. Para criar um diagrama de classes, as classes têm que estar identificadas, descritas e relacionadas entre si.

### 2.2.3. Diagrama de Colaboração

Para Page-Jones (2001), um diagrama de colaboração é o diagrama que mostra as mensagens passadas entre objetos em run-time, juntamente com os vínculos entre os objetos.

No diagrama de colaboração, além de mostrar a troca de mensagens entre os objetos, percebe-se também os objetos com os seus relacionamentos.

As setas de mensagens, em um diagrama de colaboração, são desenhadas entre os objetos para mostrar o fluxo de mensagens entre eles. As mensagens são nomeadas, que entre outras coisas mostram a ordem em que as mensagens são enviadas. Também podem mostrar condições, interações, valores de resposta, e etc. O diagrama de colaboração também pode conter objetos ativos, que executam paralelamente com outros.

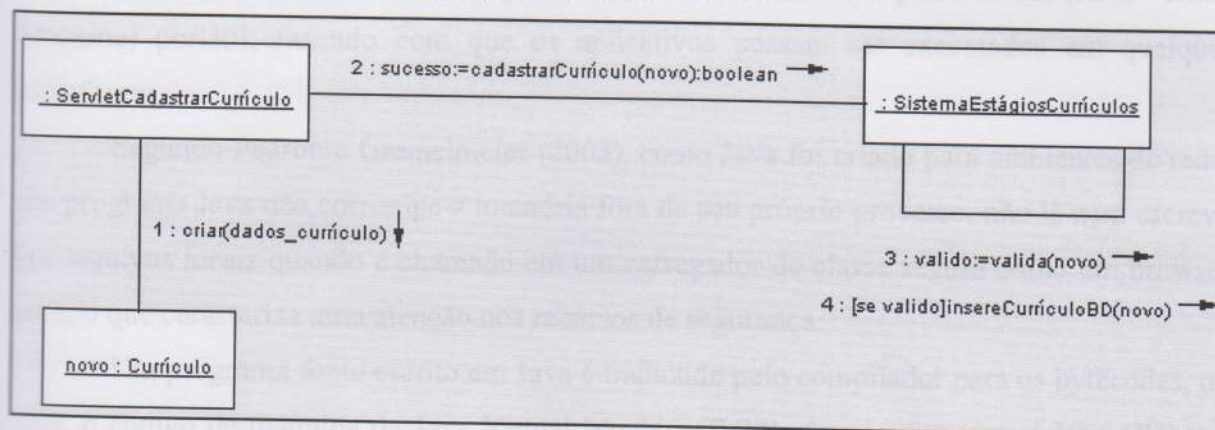


Figura 8 – Diagrama de Colaboração

### 2.3. JAVA

Segundo Jandl Junior(1999, p. 9), “Java é uma linguagem de programação introduzida no mercado em 1995 pela Sun Microsystems, que provocou e ainda provoca excitação e entusiasmo em programadores, analistas e projetistas de software”.

Essa reação é provocada pelo fato de Java ser o resultado de um trabalho consistente de pesquisa e desenvolvimento de todo um ambiente de desenvolvimento e execução de programas que exhibe as facilidades proporcionadas pela orientação a objetos, pela portabilidade do código produzido, pelas características de segurança e pela facilidade de sua integração a outros ambientes.

A linguagem Java exhibe importantes características que diferenciam-na de outras linguagens como: puramente orientada a objetos, independente de plataforma, não utiliza ponteiros, performance, segurança e permissão de múltiplas linhas de execução (multithreading).

Para Horstmann e Cornell (2001), a principal vantagem de Java é a existência de uma biblioteca em tempo de execução que visa proporcionar independência de plataforma, sendo

capaz de usar o mesmo código em diferentes sistemas operacionais, o que é extremamente necessário para a programação na Internet.

Segundo Lemay e Cadenhead (1999), a linguagem Java também é independente de plataforma no código fonte. Os programas Java são salvos como arquivo de texto antes de serem compilados, os quais podem ser criados em qualquer plataforma que ofereça suporte a Java.

Java resolve o problema interplataforma através de sua máquina virtual (Java Virtual Machine) portátil, fazendo com que os aplicativos possam ser executados em qualquer plataforma.

Segundo Pedron e Gremelmeier (2003), como Java foi criado para ambientes de rede, um programa Java não corrompe a memória fora de seu próprio processo, não lê nem escreve em arquivos locais quando é chamado em um carregador de classe seguro como um browser web, o que caracteriza uma atenção nos recursos de segurança.

Um programa fonte escrito em Java é traduzido pelo compilador para os bytecodes, ou seja, o código de máquina da Java Virtual Machine(JVM). A máquina virtual Java (JVM) é um programa capaz de interpretar os bytecodes produzidos pelo compilador, fazendo com que um programa Java possa ser executado em qualquer plataforma, desde que possua uma JVM.

O interpretador de Java pode executar os bytecodes do Java diretamente em qualquer máquina para o qual o interpretador tenha sido transportado.

Para Pearson e Gremelmeier (2003), essa é uma vantagem enquanto desenvolve-se uma aplicação, mas ela é claramente mais lenta que um código compilado. A vantagem é que ela diminui a preocupação dos desenvolvedores com problemas de versões erradas, comuns na maioria dos ambientes de desenvolvimento.

Java foi projetada para adaptar-se a um ambiente em evolução. As bibliotecas podem adicionar livremente novos métodos e variáveis de instância sem qualquer efeito sobre seus clientes. Em Java, a descoberta do tipo de dado em tempo de execução é imediata. Esta é uma característica importante nas situações em que o código precisa ser adicionado a um programa em execução. Um exemplo é o código que é transmitido através da Internet para ser executado no browser, o que caracteriza Java como uma linguagem mais dinâmica do que C ou C++.

## 2.4. JAVABEANS

“JavaBeans são classes Java que podem ser facilmente reutilizadas e reunidas em aplicações. Qualquer classe Java que siga determinadas convenções de design pode ser um componente JavaBean”. (BODOFF e GREEN, 2002, p. 246)

Segundo Fields e Kolb (2000, p. 16), “JavaBeans são objetos escritos em Java, cuja implementações estão de acordo com um conjunto de convenções projetadas para promover modularidade e reusabilidade”.

JavaBeans usa um estilo de programação que conduz a pedaços de código de programa independentes de coisas externas, que encapsulam comportamento, funcionalidade ou dados relacionados, que podem ser usados e reutilizados em múltiplos contextos em que seja preciso conhecer os detalhes de sua operação interna. Como resultado, os objetos JavaBeans podem prontamente ser conectados uns com outros e combinados a fim de fornecer capacidades específicas de aplicações ou capacidades mais sofisticadas.

Segundo Bodoff e Green (2002, p. 247), “as convenções de design de componentes JavaBeans administram as propriedades da classe e os métodos públicos que dão acesso às propriedades”.

Uma propriedade de componente JavaBean pode ser: leitura/gravação, somente leitura ou somente gravação ou simples, o que significa que ele contém um valor único ou indexado, representando uma matriz de valores.

Não há nenhuma exigência para que uma propriedade seja implementada por uma variável de instância; a propriedade deve ser acessível usando métodos públicos que estejam de acordo com determinadas convenções:

- Para cada propriedade legível, o bean deve ter um método do formato `PropertyClass getProperty() { ... }`
- Para cada propriedade gravável, o bean deve ter um método do formato `setProperty(PropertyClass pc) { ... }`

Além dos métodos das propriedades, um componente JavaBean deve definir um construtor que não tenha nenhum parâmetro.

A figura 9 mostra um exemplo da estrutura de um bean.

```

import java.io.Serializable;

public class MyBean implements Serializable{
    private int property1;
    private boolean property2;
    public MyBean( ){ }
    public void setProperty1(int property1){
        this.property1 = property1;
    }
    public void setProperty2(boolean property2){
        this.property2 = property2;
    }
    public int getProperty1( ){ return property1; }
    public boolean getProperty2( ){ return property2; }
}

```

Figura 9 – Exemplo de um JavaBean

## 2.5. JAVA DATABASE CONNECTIVITY (JDBC)

Segundo Pedron e Gremelmeier (2003), o maior incremento à programação Java, é a possibilidade de acesso a banco de dados. O mesmo programa de banco de dados desenvolvido com Java pode rodar em um PC, uma estação de trabalho ou um terminal de Java. Para atender a estes requisitos, a primeira biblioteca que a Sun forneceu foi o Java Database Connectivity (JDBC).

“O JDBC é um conjunto de classes para integração da linguagem Java com banco de dados relacionais. O Java Database Connectivity é uma API que possibilita a adição de comandos SQL em uma aplicação Java”. (DISCOVER TECHNOLOGY, 2003)

Com JDBC, os programadores podem escrever comandos SQL de forma única para tipos de banco de dados totalmente diferentes, sendo que os fabricantes dos banco de dados devem prover o driver de JDBC para seus produtos.

Para Pedron e Gremelmeier (2003), a idéia por trás do JDBC é criar suporte para utilização do SQL padrão em Java. Pode-se passar comandos em forma de String para o driver do banco de dados e receber deste os resultados de forma que toda a aplicação seja transparente ao programador. Este precisa apenas processar os comandos SQL e receber os resultados, ficando toda particularidade do banco, gerenciada pelo driver do fabricante e comandada pelas rotinas internas do JDBC.

A estrutura do JDBC inclui:

- O objetivo do JDBC é ser uma interface independente de qualquer sistema de gerenciamento de banco de dados (SGBD), permitir um acesso genérico a banco de dados através do SQL e ter uma interface uniforme para diferentes fontes de dados.
- O programador deve escrever apenas uma interface para interagir com um banco de dados e, utilizando o JDBC, o programador poderá acessar qualquer fonte de dados sem ter uma programação adicional para realizar estas operações.

A figura 10 mostra a arquitetura do JDBC. A classe DriverManager é utilizada para abrir uma conexão com um banco de dados através de um driver JDBC, que precisa estar registrado antes de realizar uma conexão. Quando uma conexão é iniciada, o DriverManager escolhe um driver de uma lista de drivers disponíveis para efetuar a conexão. Uma vez estabelecida a conexão com sucesso, as chamadas para consultas e busca de resultados serão feitas diretamente no driver JDBC.

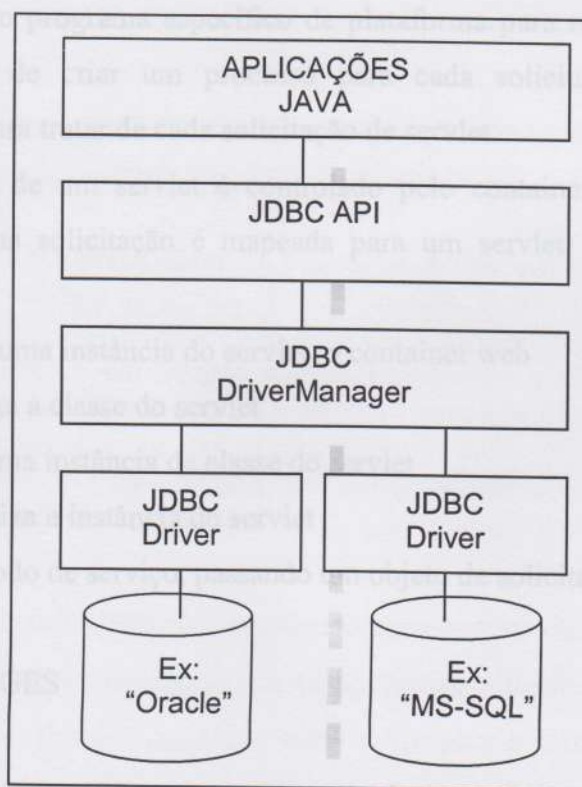


Figura 10 – Arquitetura JDBC

## 2.6. SERVLETS

“Servlet é uma classe da linguagem de programação Java usada para estender a capacidade dos servidores que hospedam aplicações acessadas via um modelo de programação do tipo solicitação-resposta”. (BODOFF e GREEN, 2002, p. 191)

Servlet foi introduzido em 1996 pela Sun Microsystems como pequenas aplicações baseadas em Java para adicionar funcionalidade dinâmica a servidores da web.

Embora os servlets possam responder a qualquer tipo de solicitação, eles são mais usados para estender as operações hospedadas em servidor web. Para tais aplicações, a tecnologia Java Servlet define classes servlet específicas para HTTP.

Segundo Fiels e Kolb (2000), os servlets Java têm um modelo de programação similar aos scripts de CGI, na medida em que eles recebem uma solicitação de HTTP de um navegador da web e espera-se que localizem e/ou construam o conteúdo apropriado para a resposta do servidor.

Diferentemente dos programas tradicionais de CGI que necessitam da criação de um novo processo para tratar de cada nova solicitação, todos os servlets associados com um servidor da web rodam dentro de um único processo. Este processo roda uma Java Virtual Machine (JVM), que é o programa específico de plataforma para rodar programas de Java compilados. Ao invés de criar um processo para cada solicitação, a JVM cria um encadeamento de Java para tratar de cada solicitação de servlet.

O ciclo de vida de um servlet é controlado pelo container no qual o servlet foi implantado. Quando uma solicitação é mapeada para um servlet, o container executa as seguintes etapas:

1º) Se não existe uma instância do servlet, o container web

- a) Carrega a classe do servlet
- b) Cria uma instância da classe do servlet
- c) Inicializa a instância do servlet

2º) Chama o método de serviço, passando um objeto de solicitação e um de resposta.

## 2.7. JAVA SERVER PAGES

“Java Server Pages – JSP – é uma tecnologia baseada em Java que simplifica o processo de desenvolvimento de sistemas da web dinâmicos”. (FIELDS e KOLB, 2000, p. 2)

Para Solís (2003), “uma página JSP não é mais que uma página web normal que contém porções de código Java e porções de código em HTML junto com outros elementos que proporcionam informação adicional ao terminal no qual a página vai ser visualizada”.

Java Server Pages foi desenvolvida pela Sun Microsystems em 1998 como resposta a aparição das páginas ASP (Active Server Pages) da Microsoft.

Telemaco Neto (2003), afirma que JSP oferece a vantagem da portabilidade de plataforma podendo ser executado em outros sistemas operacionais. Ela possibilita ao desenvolvedor de sistemas produzir aplicações que permitam o acesso a banco de dados, o acesso a arquivos-texto, a captação de informações a partir de formulários, a captação de informações a partir de informações sobre o visitante e sobre o servidor, o uso de variáveis e loops entre outras coisas.

Segundo Fields e Kolb (2000), a JSP possibilita aos designer da web e programadores incorporar rapidamente elementos dinâmicos em páginas da web usando Java embutido e algumas tags que fornecem ao designer de HTML um meio de acesso de dados e lógica de negócios armazenados em objetos Java sem ter que dominar as complexidades do desenvolvimento de aplicações.

A primeira vez que uma página JSP é carregada pelo container JSP o código Java é compilado gerando um servlet que é executado, gerando uma página HTML que é enviada para o navegador. As chamadas subseqüentes são enviadas diretamente ao servlet gerado na primeira requisição, não ocorrendo mais as etapas de geração e compilação do servlet.

Fields e Kolb (2000, p. 12) afirmam que

JSP oferece diversos benefícios como um sistema para geração de conteúdo dinâmico. Primeiramente, sendo uma tecnologia baseada em Java, ela se aproveita de todas as vantagens que a linguagem Java fornece em relação a desenvolvimento e acionamento. Como uma linguagem orientada a objetos com forte digitação, encapsulamento, tratamento de exceções e gerenciamento de memória automático, o uso de Java conduz a uma produtividade aumentada do programador e a código mais robusto.

A própria JSP oferece diversas vantagens como um modelo de programação que enfatiza o design de aplicações centradas em componentes e performance aperfeiçoada na CGI. Este modelo de programação permite o uso de JSP para manter uma rígida separação entre os elementos de apresentação de uma aplicação da web e sua implementação básica.

O JSP fornece um meio bem simples de manter essa separação, seguindo três princípios básicos: o primeiro é a não existência de código de implementação misturado com código de apresentação; o segundo é a não existência de código HTML no objeto JavaBeans; e o terceiro especialmente dedicado à geração programática de HTML usando-se código de Java.

A figura 11 mostra o papel do mecanismo de extensão de tags da especificação JSP, que permite que programadores Java implementem novas tags JSP e as empacotem em bibliotecas específicas de aplicação que podem então ser importadas para páginas JSP individuais.

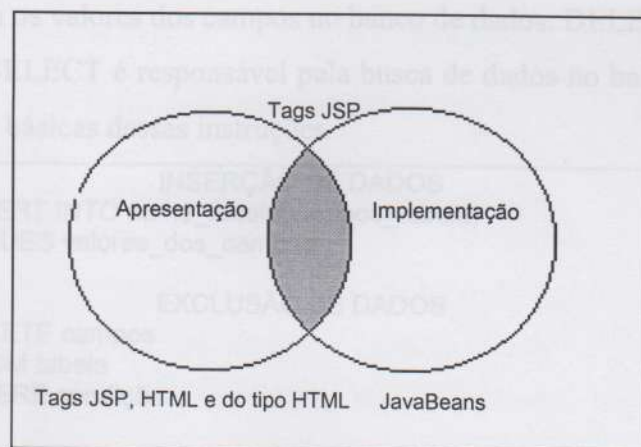


Figura 11 – Suporte de recursos JSP para a separação de apresentação e implementação

Segundo Melo (2002), o benefício da separação destes dois aspectos, é que as modificações em um podem ser realizadas sem a necessidade de qualquer alteração no outro. Essa tecnologia permite a utilização de tags HTML de JSP para o acesso as funcionalidades dos componentes JavaBeans. Se nenhuma das tags disponíveis fornecer a funcionalidade necessária desejada, pode-se escrever uma biblioteca de tags específicas de aplicação (tags personalizadas), usando o mecanismo de extensão de tags JSP.

## 2.8. SQL

Silberschatz, Korth e Sudarshan (1999), referem-se a linguagem SQL como uma linguagem de consulta, mas ela possui muitos outros recursos além da consulta ao banco de dados, como meios para a definição da estrutura de dados, para modificação de dados no banco de dados e para especificação de restrições de segurança.

A SQL foi desenvolvida pelo Departamento de Pesquisas da IBM como forma de interface para o sistema de banco de dados relacional denominado SYSTEM R, no início dos anos 70. Em 1986, o American National Standard Institute (ANSI), publicou um padrão SQL.

A linguagem SQL tem como grandes virtudes sua capacidade de gerenciar índices, sem a necessidade de controle individualizado de índice corrente, algo muito comum nas linguagens de manipulação de dados do tipo registro a registro. Outra característica muito importante disponível em SQL é sua capacidade de construção de visões, que são formas de

visualizar os dados na forma de listagens independentes das tabelas e organização lógica dos dados.

A SQL têm como comandos básicos: INSERT, UPDATE, DELETE e SELECT. A instrução INSERT é responsável pela inserção de dados no banco de dados. UPDATE é a instrução que atualiza os valores dos campos no banco de dados. DELETE exclui informações do banco de dados. SELECT é responsável pela busca de dados no banco de dados. A figura 12 mostra as sintaxes básicas dessas instruções.

|                                                                                                             |
|-------------------------------------------------------------------------------------------------------------|
| <p><b>INSERÇÃO DE DADOS</b></p> <pre>INSERT INTO nome_tabela(campos_tabela) VALUES valores_dos_campos</pre> |
| <p><b>EXCLUSÃO DE DADOS</b></p> <pre>DELETE campos FROM tabela WHERE condição</pre>                         |
| <p><b>PESQUISA DE DADOS</b></p> <pre>SELECT campos FROM tabela WHERE condição</pre>                         |
| <p><b>ATUALIZAÇÃO DE DADOS</b></p> <pre>UPDATE campo FROM tabela SET novos_valores WHERE condição</pre>     |

Figura 12 – Sintaxe básica das instruções SQL

## 2.9. HTML

HTML é a linguagem usada para criar páginas web, que por sua vez será composta de textos e comandos especiais denominados tags. A HTML tem como finalidade básica formatar texto e imagens exibidos e criar ligações entre páginas web, criando assim documentos com o conceito de hipertexto.

Para Alvarez (2003), a HTML é uma linguagem universal destinada à elaboração de páginas com hipertexto. O conceito de hipertexto está associado a uma nova forma de documentos, compostos por texto formatado e ligações a outros documentos, os quais poderão ser do tipo hipertexto, ou mesmo imagens, sons, vídeo, etc.

Segundo Beceiro (2003), uma página HTML é um conjunto de comandos e textos, digitado no formato de texto puro que determinam aos browsers a maneira de apresentar a página no microcomputador.

Severo (2003) afirma que a HTML não foi criada para controlar a aparência dos documentos, ao contrário dos processadores de texto e programas de layout da página.

Os documentos HTML possuem algumas características básicas como: formatação de documentos em vários estilos de letras, cores, tamanhos; formatação de tabelas; formatação de listas; criação de mapas clicáveis, relacionados a hiperlinks; e, formatação de formulários.

Os comandos em HTML, tags, irão “dizer” ao browser como o texto, a informação e as imagens serão exibidos.

As tags são identificadas por serem envolvidas pelos sinais < > ou < / >. Entre os sinais < > são especificados as tags propriamente ditas. No caso de tags que precisam envolver um texto, a sua finalização deverá ser feita usando a barra de divisão “/”, indicando que a tag está finalizando a marcação de um texto, o que faz as tags normalmente andarem em pares.

```
<NOME DO TAG> Texto </NOME DO TAG>
```

Figura 13 – Formato genérico de um tag

A estrutura de um documento HTML apresenta os seguintes componentes:

```
<HTML>
  <HEAD><TITLE>Título do documento</TITLE></HEAD>
  <BODY>
    texto,
    imagem,
    links,
    ...
  </BODY>
</HTML>
```

Figura 14 – Estrutura de um documento HTML

Os documentos HTML se dividem em duas seções principais: a seção <HEAD> e a <BODY>. A <HEAD> contém informações sobre o documento; o elemento <TITLE>, por exemplo, define um título que é mostrado no alto do browser. Na seção <BODY>, está contido o que será mostrado na janela principal do browser, sendo apresentado ao leitor. <BODY> pode conter cabeçalhos, parágrafos, tabelas, formulários, links para outros documentos e imagens.

### 29.1. Formulários

Um formulário é um modelo para a entrada de um conjunto de dados. O primeiro passo para fazer formulários é aprender as etiquetas que desenham as janelinhas de entrada de

dados, para depois trabalharmos com os *scripts*, que são os programas que tratam esses dados, oferecendo os serviços desejados (acesso a banco de dados, envio de e-mail, etc.).

O elemento `<FORM>` delimita um formulário e contém uma sequência de elementos de entrada e de formatação do documento.

```
<FORM ACTION="URL_de_script" METHOD=método>...</FORM>
```

Figura 15 – Sintaxe de formulário em HTML

Para entrada de dados é utilizado a tag `<INPUT>`, conforme estrutura mostrada na figura 16, delimita diversos tipos como campos de texto, senha, caixas de seleção e botões.

```
<INPUT TYPE="tipo_entrada" NAME="nome_campo">
```

Figura 16 – Estrutura de uma tag INPUT

## 2.10. FERRAMENTAS UTILIZADAS PARA O DESENVOLVIMENTO

### 2.10.1. Poseidon for UML 1.2

Poseidon for UML 1.2 é a ferramenta de modelagem da empresa Gentleware AG, representada comercialmente no Brasil pela Mondo Technologies, fazendo parte da sua suíte de produtos de apoio a projetos de software Mondo Methods.

O Poseidon é uma evolução da ferramenta de código-aberto Argo UML. Seu principal foco está na facilidade de uso que a torna simples de aprender e usar.

Esta ferramenta foi utilizada para a construção da análise orientada a objeto: os diagramas de caso de uso, diagrama de classes e diagrama de colaboração.

### 2.10.2. Microsoft FrontPage 2000

O Microsoft FrontPage 2000 é uma ferramenta da Microsoft Corporation, disponível para PC nos sistemas operativos MS-Windows que permite a gestão de informação na web. Uma das grandes vantagens do FrontPage é a sua publicação direta sem necessidade de usar, por exemplo, o protocolo FTP, afirma o Centro de Informática e Educação (2003). Neste trabalho, o FrontPage 2000 foi utilizado para construção das interfaces HTML que compõem o sistema.

### 2.10.3. Microsoft Access 2000

Segundo Mink e Aníbal Junior (2000), o Access 2000 é um aplicativo, desenvolvido pela Microsoft Corporation, que faz parte da última geração de programas de bancos de dados, voltado para o sistema operacional Windows. Como aplicativo do Office 2000 faz amplo uso da interface gráfica do sistema, bem como de modelos assistentes que facilitam a criação e manipulação de banco de dados. O Access 2000 foi o aplicativo onde foi desenvolvido o banco de dados que compõe o sistema.

### 2.10.4. JCreator LE

HYPERLINK (2003) afirma que o JCreator LE é uma IDE Java para Windows de fácil operação da Xinox. O JCreator é um software gratuito que para seu funcionamento requer apenas a instalação do pacote Java SDK da Sun. Este proporciona edição e compilação de código Java, sendo o primeiro a não utilizar a tela do MS-DOS para compilação.

O JCreator foi utilizado para construção das classes Java que manipulam os dados no banco de dados e as outras operações decorrentes do sistema.

### 2.10.5. Resin 2.1.5

Segundo Rocha Neto (2002), o servidor Resin 2.1.5 foi desenvolvido pela empresa Caucho, possui distribuição livre. O Resin é de fácil instalação e operacionalização. Para iniciá-lo, basta acessar a pasta BIN do servidor e executar o arquivo httpd.exe. Para acessar as páginas, basta digitar na barra de endereço do browser o seguinte endereço: [http://localhost:8080/nome\\_da\\_página](http://localhost:8080/nome_da_página). Este é o responsável pela execução de todas as interfaces e scripts JSP do sistema.

O Resin é um servidor que suporta scripts JSP. Neste trabalho foi utilizado para criar o conteúdo dinâmico do sistema, ou seja, interligar o banco de dados com os scripts e as interfaces.

### 3. METODOLOGIA

O presente trabalho classifica-se como uma pesquisa aplicada, proporcionando uma solução para o problema da descentralização e da divulgação das ofertas de estágios para os alunos da FARN, utilizando teorias como orientação a objetos, UML, Java e JavaServer Pages. Oliveira (1998) afirma que a pesquisa aplicada tem por objetivo pesquisar, comprovar ou rejeitar hipóteses sugeridas pelos modelos teóricos e fazer a sua aplicação às diferentes necessidades humanas.

Para o desenvolvimento deste trabalho foram seguidos alguns passos a fim de possibilitar uma seqüência lógica de atividades que permitam alcançar o objetivo principal deste estudo: a implementação de um protótipo de um sistema de ofertas de estágios e currículos para o site da FARN. Primeiramente, foi realizada uma pesquisa bibliográfica sobre as tecnologias a serem utilizadas para a construção do sistema.

O segundo passo se deu através de entrevistas com os coordenadores da FARN afim de levantar dados que sirvam como requisitos para a construção da análise do sistema e expressar sua viabilidade, caracterizando uma pesquisa de campo.

Os dados coletados neste estudo classificam-se como pesquisa qualitativa. Neste trabalho foi analisada cada situação, identificando relações, causas, efeitos e conseqüências necessários a compreensão do problema proposto. Segundo Marconi e Lakatos (1999), na pesquisa qualitativa, o sentido prioritário, depende, conseqüentemente, da natureza do problema, suas causas e seus efeitos e do material que os métodos permitem coletar. A pesquisa qualitativa tem como objetivo situações complexas ou estritamente particulares.

Em seguida, foi desenvolvida a análise do sistema utilizando os conceitos da orientação a objetos e o desenvolvimento dos diagramas de casos de uso, de classes e de colaboração.

Finalizando, realizou-se a implementação do sistema, obedecendo quatro etapas: projeto de interface usando HTML, projeto lógico e físico do banco de dados, desenvolvimento das classes Java (beans) e programação dinâmica para web com JSP, possibilitando os testes de usabilidade do sistema.

## 4. ANÁLISE E IMPLEMENTAÇÃO DO SISTEMA

### 4.1. ANÁLISE DO SISTEMA DE ESTÁGIOS E CURRÍCULOS

A proposta do desenvolvimento do sistema de estágios e currículos surgiu no segundo semestre do ano de 2002 como parte do projeto multidisciplinar para os alunos do então 3º ano do curso de Bacharelado em Sistemas de Informação.

Este projeto foi inicialmente criado como forma de avaliação das disciplinas Tópicos Avançados em Programação, Tópicos Avançados em Internet e Banco de Dados II, sob a orientação dos professores Kaiser Magalde, Fellipe Araújo Aleixo e Sérgio Badiali, tendo como base às tecnologias ministradas nas referidas disciplinas: Java, JSP e SQL, respectivamente.

Daí então surgiu a necessidade de desenvolver um sistema que aplicasse essas tecnologias, além do desejo de deixar para a instituição algo positivo que tivesse realmente uma aplicação prática e que trouxesse algum benefício para a instituição e para os alunos que fazem parte desta. Já que o propósito do projeto multidisciplinar não foi totalmente atingido por não haver uma integração entre os módulos desenvolvidos.

Portanto, surgiu o desejo de criar este trabalho, já que o projeto não foi implementado totalmente, deixando a desejar sobre o seu funcionamento. A maioria das operações propostas neste não foram implementadas, o que inviabilizou a implantação deste.

Para o desenvolvimento deste trabalho, aproveitou-se apenas a idéia. Toda a parte de análise e implementação do sistema, foi feita a partir do começo. Todos os dados referentes a outros módulos não foram pesquisados nem aproveitados já que a idéia era recomeçar o processo de desenvolvimento, criando assim, um sistema homogêneo que atende as reais necessidades impostas.

Inicialmente, para o desenvolvimento da análise do sistema, foram aplicados questionários aos coordenadores e professores a fim de coletar dados para a análise de requisitos e para verificar a usabilidade do sistema. Esses dados serviram de base para a construção da análise orientada a objetos que consiste do diagrama de casos de uso, diagrama de classes e diagramas de colaboração, utilizando a UML e a ferramenta Poseidon for UML

1.2. A aluna Renata Barros afirma que uma das grandes vantagens do sistema é a facilidade no acesso às informações relativas às oportunidades de estágios, não necessitando percorrer as agências de estágios e as empresas.

#### 4.1.1. Análise de Requisitos

Para construção dessa análise, não foi aproveitado nenhum dado ou especificação de outros módulos do projeto. A análise foi recomeçada pelo fato de não haver uma documentação real do desenvolvimento dos outros módulos propostos no projeto multidisciplinar e também pelo fato de que cada desenvolvedor possui uma idéia diferente para seu módulo, não criando assim um sistema homogêneo.

A análise de requisitos se deu através de aplicação de questionário, cujo modelo encontra-se no anexo A, com coordenadores e professores dos cursos de Administração, Administração com Habilitação em Marketing, Bacharelado em Sistemas de Informação e Licenciatura em Computação, alunos da instituição e empresas interessadas.

A primeira questão levantada no questionário foi a importância do desenvolvimento de um sistema de estágios e currículos para a instituição.

Segundo a professora Livia Maria, coordenadora dos cursos de Bacharelado em Sistemas de Informação e Licenciatura em Computação, uma das maiores dificuldades dos alunos é saber onde ou quais as empresas que necessitam de estagiários, mesmo com a presença de agências de estágios como o IEL e o CIEE. O sistema quando implantado facilitará o acesso dos alunos às oportunidades lançadas no mercado.

Além disso, a professora Catarina Souza, coordenadora dos cursos de Administração e Administração com Habilitação em Marketing, afirma que com a criação de um local específico para coordenar as informações referentes a ofertas de estágios e currículos haverá um avanço com relação a oferta e concretização de estágios.

O segundo ponto questionado foi em relação as vantagens que o sistema poderia trazer para os alunos da instituição. Para o professor Carlos Antônio Moreira, professor do curso de Administração, o sistema tornará público as informações relativas a cada estudante, quanto à formação acadêmica, experiência profissional, habilidades, qualificações, entre outras, facilitando o acesso das empresas ao profissional que mais se adapte ao perfil procurado.

A professora Catarina Souza ressalta também como vantagens: a rapidez do cadastramento tanto das ofertas de estágios quanto dos currículos dos alunos, verificação do perfil exigido pelo mercado e adequação dos alunos às necessidades do mercado.

A aluna Renata Barros afirma que uma das grandes vantagens do sistema é a facilidade no acesso as informações relativas às oportunidades de estágios, não necessitando percorrer as agências de estágios e as empresas.

Para Allyson Medeiros, diretor da empresa Infonatal Informática e Serviços, o sistema facilitará a vida das empresas interessadas em contratar estagiários, já que não será necessário deslocar-se até as agências de estágios para disponibilizar vagas de estágios, além de possibilitar a pesquisa de currículos, simplificando o acesso a mão-de-obra mais adequada ao perfil procurado pela empresa.

Segundo o professor Luiz Augusto Mendes, professor dos cursos de Administração e Bacharelado em Sistemas de Informação, o sistema se implantado no site da instituição, facilitará o acesso às informações contidas neste, já que os alunos utilizam o site para obter informações sobre a faculdade, serviço de email e acesso ao conteúdo das disciplinas ministradas.

Outra questão lançada diz respeito as desvantagens que o sistema poderia acarretar. A única citada foi quanto ao gerenciamento do sistema, pela professora Catarina Souza que afirma: “o sistema deve ser bem gerenciado, pois do contrário, as falhas recairão sobre a imagem da instituição”.

O próximo questionamento se deu quanto aos dados que seriam disponibilizados no currículo do aluno. A figura 17 apresenta o resultado desse questionamento.

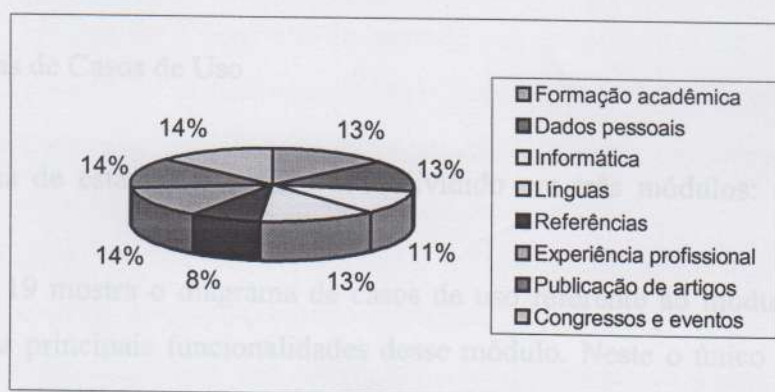


Figura 17 – Dados a serem disponibilizados no currículo do aluno

Segundo o professor Carlos Antônio Moreira, os dados pessoais dos alunos devem ficar restritos ao nome, email e telefone fixo ou celular, já que a Internet não é um meio totalmente seguro de tráfego de informações. Além da publicação de referências que deve ser verificada anteriormente sobre a permissão da divulgação das empresas ou pessoas que serão citadas.

Finalizando, a última questão foi quanto as operações que o sistema iria disponibilizar para os usuários. A figura 18 demonstra os resultados obtidos por este questionamento.

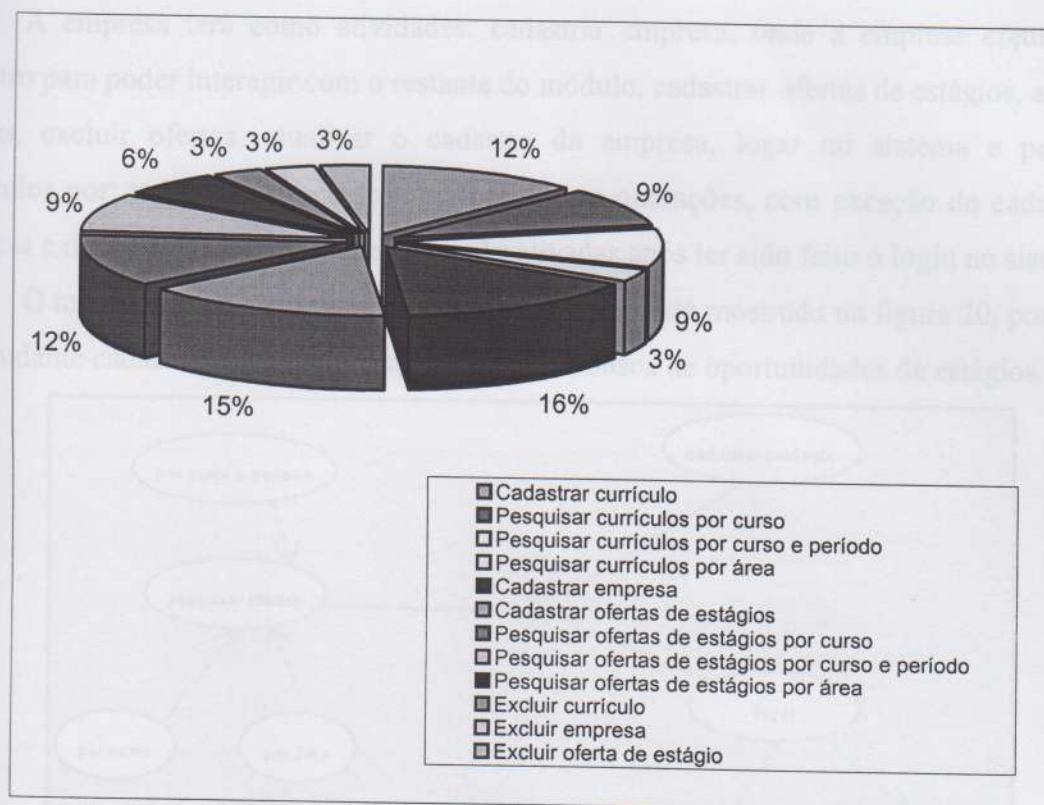


Figura 18 – Operações disponíveis no sistema

#### 4.1.2. Diagramas de Casos de Uso

O sistema de estágios e currículos é dividido em três módulos: empresa, aluno e administrador.

A figura 19 mostra o diagrama de casos de uso referente ao módulo empresa, onde estão descritas as principais funcionalidades desse módulo. Neste o único ator que interage com o sistema é a empresa por intermédio de um responsável através de um login e de uma senha informados pelo mesmo no ato do cadastro da empresa.

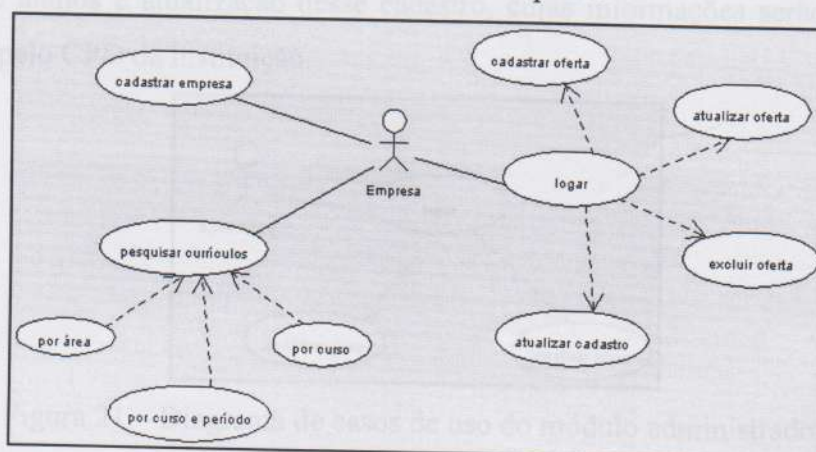


Figura 19 – Diagrama de casos de uso do módulo empresa

A empresa terá como atividades: cadastrar empresa, onde a empresa efetuará um cadastro para poder interagir com o restante do módulo, cadastrar ofertas de estágios, atualizar ofertas, excluir ofertas, atualizar o cadastro da empresa, logar no sistema e pesquisar currículos por: curso, área e curso e período. Estas operações, com exceção do cadastro da empresa e da pesquisa de currículos, só serão ativadas após ter sido feito o login no sistema.

O módulo aluno, cujo diagrama de casos de uso está mostrado na figura 20, possibilita ao estudante cadastrar e gerenciar seu currículo e a busca de oportunidades de estágios.

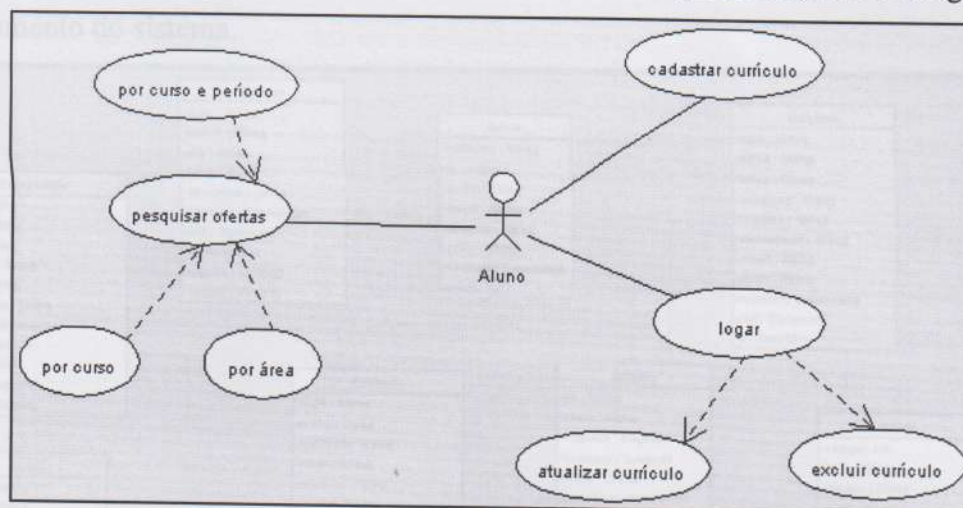


Figura 20 – Diagrama de casos de uso do módulo aluno

No módulo aluno estão incluídas as seguintes opções: cadastrar currículo, logar no sistema, atualizar currículo, excluir currículo e pesquisar ofertas por: curso, área e curso e período.

Assim como no módulo empresa, o aluno só terá acesso as opções, com exceção de pesquisar ofertas, após ter efetuado o login no sistema.

Para gerenciar os alunos que interagem com o sistema, o módulo administrador, conforme diagrama de casos de uso encontra-se na figura 21, dispõe as operações referentes ao cadastro de alunos e atualização desse cadastro, cujas informações serão informadas ao administrador pelo CPD da instituição.

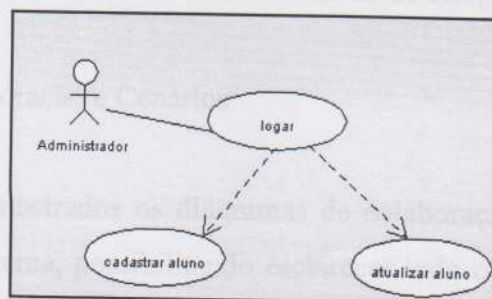


Figura 21 – Diagrama de casos de uso do módulo administrador

Todas as operações contidas no sistema serão descritas no item 4.1.4. através dos cenários dos casos de uso presentes na análise.

#### 4.1.3. Diagrama Conceitual

Nesse item será mostrado o diagrama conceitual do sistema, onde serão descritos as classes, os relacionamentos e os atributos das classes, possibilitando uma visão geral sobre o funcionamento do sistema.

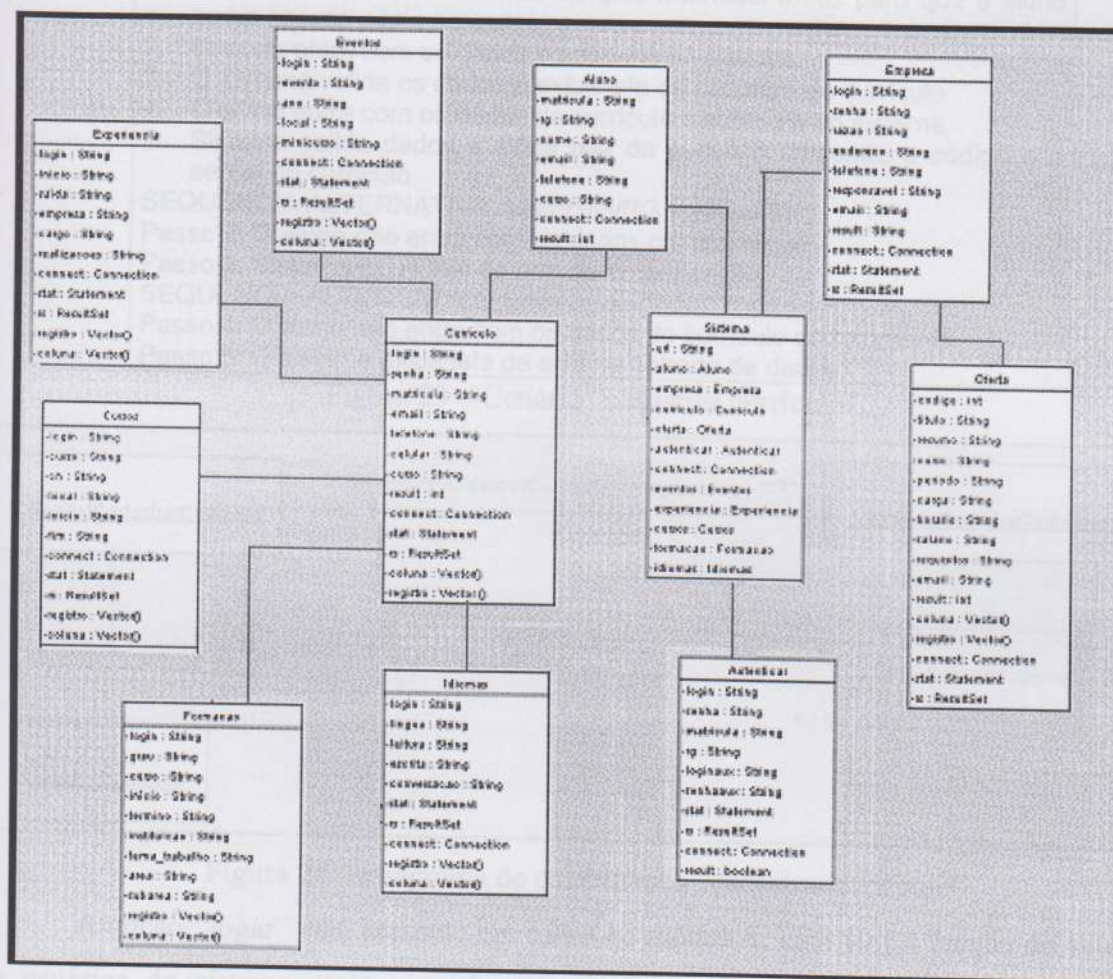


Figura 22 – Modelo conceitual do sistema de estágios e currículos

#### 4.1.4. Diagramas de Colaboração e Cenários

Neste item serão mostrados os diagramas de colaboração e os cenários de todas as operações contidas no sistema, possibilitando esclarecer todo o funcionamento do sistema e criar uma documentação consistente do mesmo.

A primeira operação a ser descrita é “cadastrar currículo” que faz parte do módulo aluno. Nesta o aluno cadastra seu currículo, desde que esteja cadastrado anteriormente no sistema. Para que o aluno acesse o formulário de cadastro de currículo, é necessário que antes tenha digitado a matrícula e o rg para verificar se realmente faz parte da instituição, conforme passos descritos na figura 23.

CASO DE USO: Cadastrar currículo  
 ATOR: Aluno  
 SEQUÊNCIA NORMAL:  
 0. O aluno seleciona a opção “Cadastrar currículo” no menu principal  
 1. O sistema exibe tela com os campos matrícula e RG para que o aluno tenha acesso a tela de cadastro  
 2. O aluno entra com os dados e submete ao sistema  
 3. O sistema valida os dados e exibe tela de cadastro de currículo  
 4. O aluno entra com os dados do currículo e submete ao sistema  
 5. Sistema valida dados e exibe tela de sucesso contendo o código e a senha do currículo  
 SEQUÊNCIA ALTERNATIVA: Matrícula/RG inválido(s)  
 Passo 2: O aluno não entra com os dados corretamente  
 Passo 3: Sistema exibe tela de erro de autenticação  
 SEQUÊNCIA ALTERNATIVA: Entrada de dados inválida  
 Passo 4: O aluno não entra com os dados do currículo corretamente  
 Passo 5: O sistema exibe tela de erro na entrada de dados

Figura 23 – Cenário “cadastrar currículo”

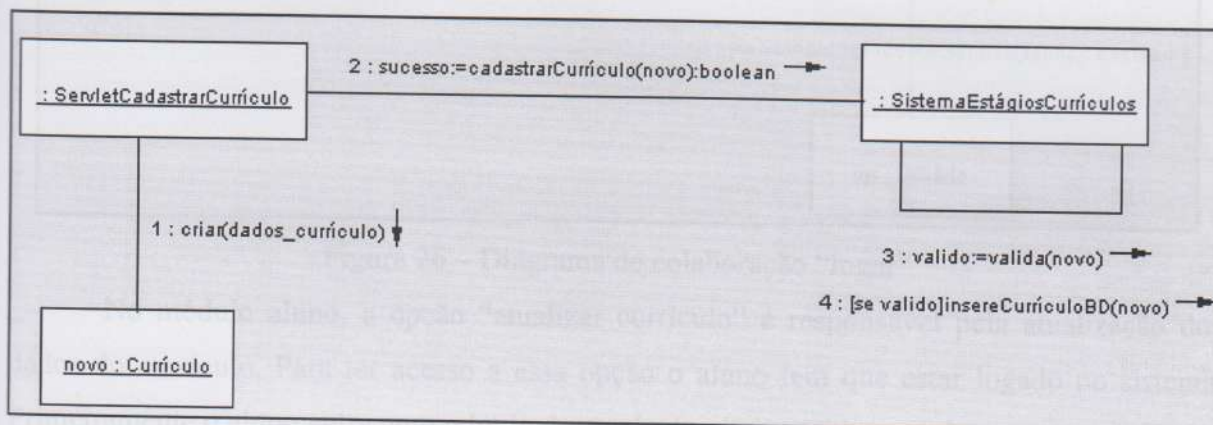


Figura 24 – Diagrama de colaboração “cadastrar currículo”

A opção “logar” está presente em todos os módulos. Esta tem a função de autenticar os usuários do sistema, pois só após esta autenticação é que serão ativadas as opções principais do sistema.

**CASO DE USO: Logar**  
**ATOR:** Aluno/Empresa/Administrador  
**SEQUÊNCIA NORMAL:**  
 0. O ator escolhe a opção "Logar" no menu principal  
 1. O sistema exibe tela de login  
 2. O ator entra com os dados de login e submete ao sistema  
 3. O sistema valida dados e exibe tela de sucesso  
**SEQUÊNCIA ALTERNATIVA: Dados de login inválidos**  
 Passo 2: O ator entra com os dados de login incorretos  
 Passo 3: O sistema exibe tela de erro de login  
**SEQUÊNCIA ALTERNATIVA: Esqueceu login/senha**  
 Passo 2: O ator seleciona a opção "Esqueceu login/senha"  
 Passo 3: O sistema exibe tela para o ator entrar com o seu email  
 Passo 4: O ator entra com o email e submete ao sistema  
 Passo 5: O sistema valida email, busca no banco de dados o login e a senha correspondentes ao email informado e exibe tela com o login e a senha do ator.

Figura 25 – Cenário "logar"

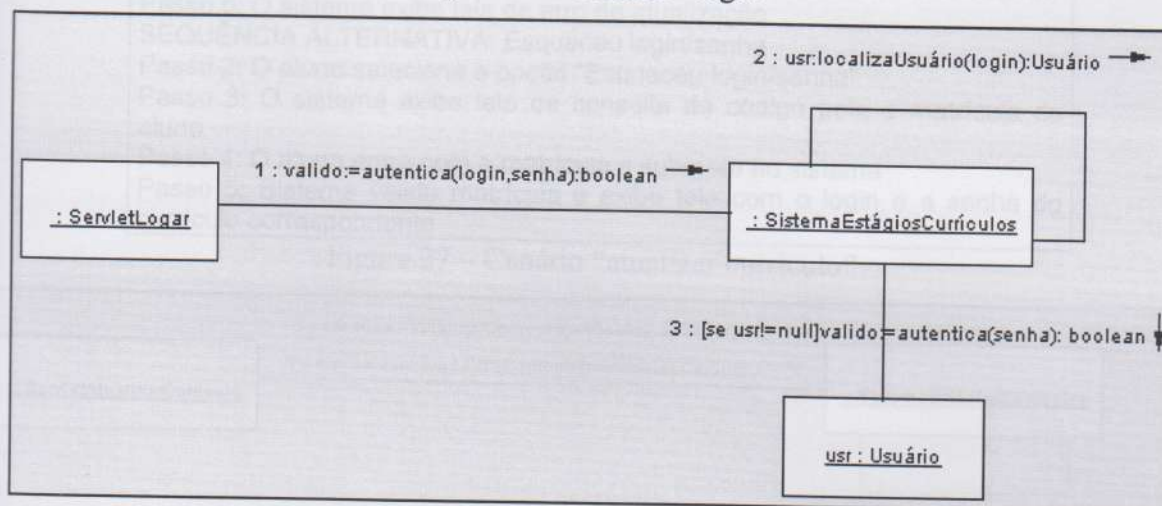


Figura 26 – Diagrama de colaboração "logar"

No módulo aluno, a opção "atualizar currículo" é responsável pela atualização dos dados do currículo. Para ter acesso a essa opção o aluno tem que estar logado no sistema. Primeiramente o aluno entra com o login do currículo para ser feita uma busca que retorna um formulário com todos os dados para atualização se for um login válido. Posteriormente, o aluno digita os dados a serem atualizados e envia os novos dados para serem salvos no banco de dados, conforme sequência de passos que estão descritos na figura 27.

CASO DE USO: Atualizar currículo

ATOR: Aluno

SEQUÊNCIA NORMAL;

0. O aluno seleciona a opção "Atualizar currículo" após ter efetuado login no sistema
1. O sistema exibe tela de consulta pelo login e a senha do currículo
2. O aluno entra com o login e a senha do currículo e submete ao sistema
3. Sistema valida login e senha, busca as informações do currículo e exibe estas informações na forma de formulário
4. O aluno entra com os dados a serem atualizados e submete ao sistema
5. O sistema valida dados, atualiza o currículo no banco de dados e exibe tela de sucesso

SEQUÊNCIA ALTERNATIVA: Login e senha de currículo inexistente

Passo 2: O aluno entra com login e senha do currículo inexistente

Passo 3: O sistema exibe mensagem: "Não existe currículo com os dados informados"

SEQUÊNCIA ALTERNATIVA: Entrada de dados inválida

Passo 4: O aluno não entra com os dados a serem atualizados corretamente

Passo 5: O sistema exibe tela de erro de atualização

SEQUÊNCIA ALTERNATIVA: Esqueceu login/senha

Passo 2: O aluno seleciona a opção "Esqueceu login/senha"

Passo 3: O sistema exibe tela de consulta de código pela a matrícula do aluno

Passo 4: O aluno entra com a matrícula e submete ao sistema

Passo 5: Sistema valida matrícula e exibe tela com o login e a senha do currículo correspondente

Figura 27 – Cenário "atualizar currículo"

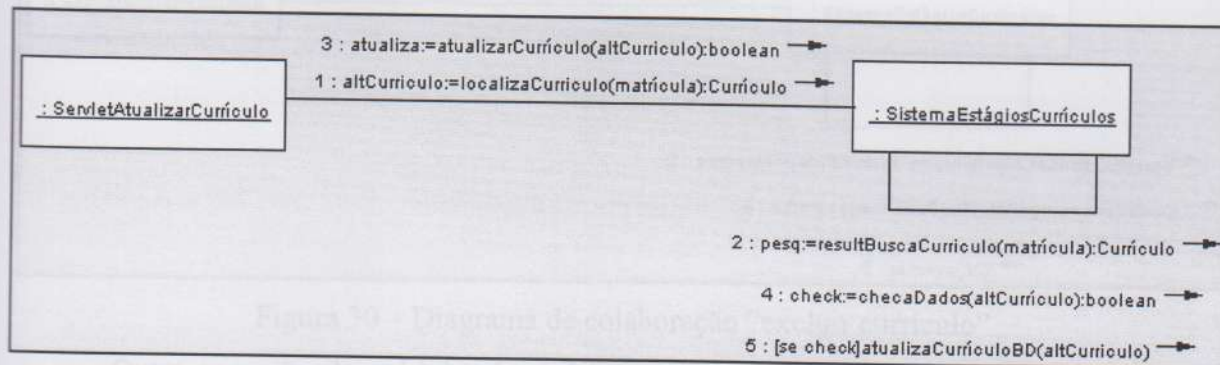


Figura 28 – Diagrama de colaboração "atualizar currículo"

A opção "excluir currículo" também faz parte do módulo aluno. Nela o aluno, após ter logado no sistema, entra com o login cadastrado no currículo para que o sistema faça uma busca e retorne os dados do currículo para confirmação de exclusão. Só após esta confirmação é que o sistema exclui os dados desse currículo no banco de dados.

**CASO DE USO: Excluir currículo****ATOR: Aluno****SEQUÊNCIA NORMAL:**

0. O aluno seleciona a opção "Excluir currículo" após ter efetuado login no sistema
1. Sistema exibe tela de consulta com o login e a senha do currículo
2. O aluno entra com os dados e submete ao sistema
3. O sistema busca os dados referentes ao currículo e retorna-os para confirmação de exclusão do aluno.
4. O aluno confirma exclusão e submete ao sistema
5. O sistema valida dados, exclui o currículo do banco de dados e exibe tela de sucesso

**SEQUÊNCIA ALTERNATIVA: Login e senha de currículo inexistente**

Passo 2: O aluno entra com login e senha do currículo inexistente

Passo 3: O sistema exibe mensagem: "Não existe currículo com os dados informados"

**SEQUÊNCIA ALTERNATIVA: Esqueceu login/senha**

Passo 2: O aluno seleciona a opção "Esqueceu login/senha"

Passo 3: O sistema exibe tela de consulta de código pela a matrícula do aluno

Passo 4: O aluno entra com a matrícula e submete ao sistema

Passo 5: Sistema valida matrícula e exibe tela com o login e a senha do currículo correspondente

Figura 29 – Cenário "excluir currículo"

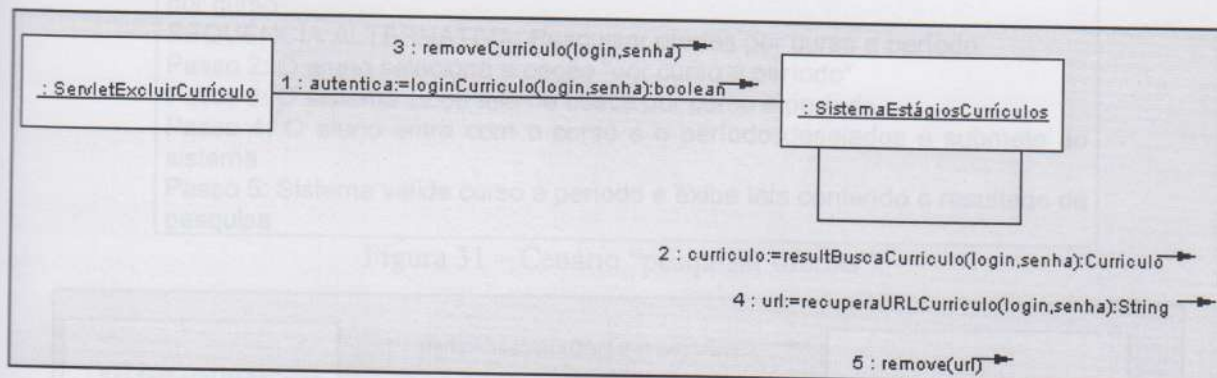


Figura 30 – Diagrama de colaboração "excluir currículo"

Outra operação do módulo aluno é "pesquisar ofertas de estágios". Para essa opção não é necessário o aluno estar logado no sistema. O sistema possibilita três opções de busca ao aluno: por curso, por área e por curso e período. O aluno entra com os dados referentes a busca escolhida e o sistema retorna uma página com o resultado da pesquisa.

Figura 31 – Diagrama de colaboração "pesquisar ofertas por área"

CASO DE USO: Pesquisar ofertas

ATOR: Aluno

SEQUÊNCIA NORMAL:

0. O aluno seleciona a opção "Pesquisar ofertas" no menu principal
1. O sistema exibe tela de pesquisa com três opções: por curso, por área e por curso e período
2. O aluno seleciona a opção desejada
3. O sistema exibe tela de consulta com a opção desejada
4. O aluno entra com os dados da pesquisa e submete ao sistema
5. Sistema valida dados e exibe tela com o resultado da pesquisa

SEQUÊNCIA ALTERNATIVA: Entrada de dados incorreta

Passo 4: O aluno não entra com os dados da pesquisa corretamente

Passo 5: Sistema exibe tela de erro de entrada de dados

SEQUÊNCIA ALTERNATIVA: Pesquisar ofertas por área

Passo 2: O aluno seleciona a opção "por área"

Passo 3: O sistema exibe tela de busca por área

Passo 4: O aluno entra com a área desejada e submete ao sistema

Passo 5: Sistema valida área e exibe tela contendo o resultado da pesquisa por área

SEQUÊNCIA ALTERNATIVA: Pesquisar ofertas por curso

Passo 2: O aluno seleciona a opção "por curso"

Passo 3: O sistema exibe tela de busca por curso

Passo 4: O aluno entra com o curso desejado e submete ao sistema

Passo 5: Sistema valida curso e exibe tela contendo o resultado da pesquisa por curso

SEQUÊNCIA ALTERNATIVA: Pesquisar ofertas por curso e período

Passo 2: O aluno seleciona a opção "por curso e período"

Passo 3: O sistema exibe tela de busca por curso e período

Passo 4: O aluno entra com o curso e o período desejados e submete ao sistema

Passo 5: Sistema valida curso e período e exibe tela contendo o resultado da pesquisa

Figura 31 – Cenário "pesquisar ofertas"

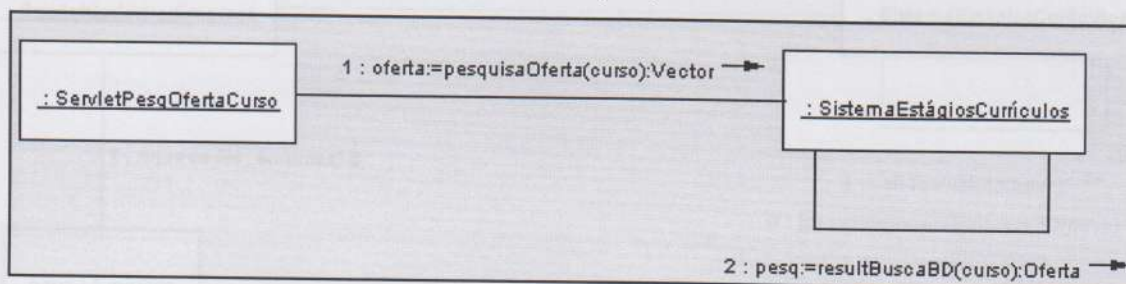


Figura 32 – Diagrama de colaboração "pesquisar ofertas por curso"

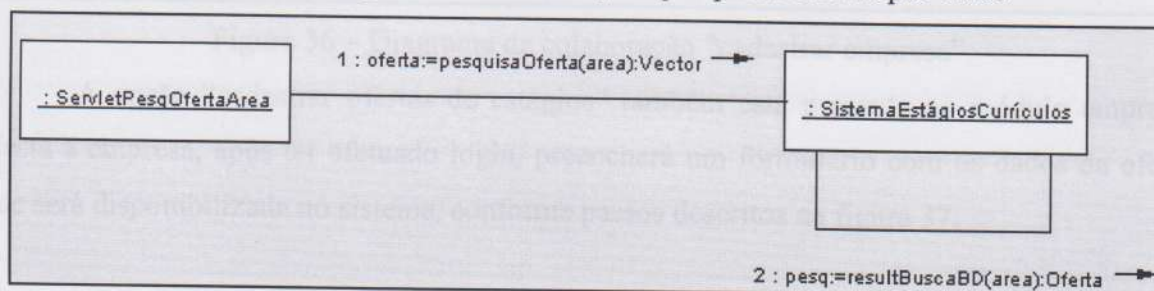


Figura 33 – Diagrama de colaboração "pesquisar ofertas por área"

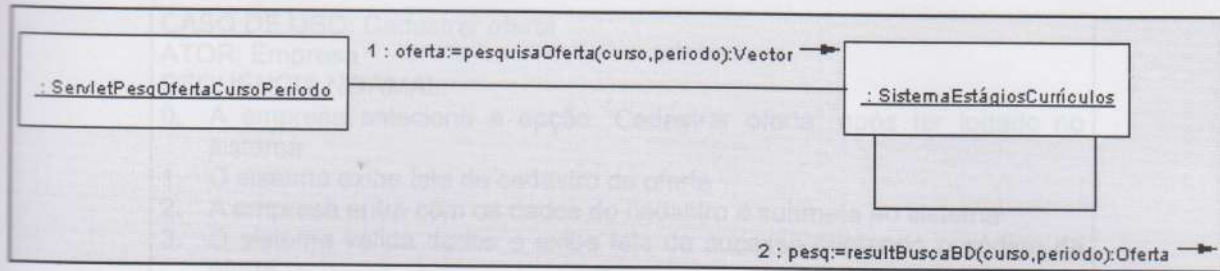


Figura 34 – Diagrama de colaboração “pesquisar ofertas por curso e período”

“Cadastrar empresa” faz parte do módulo empresa. Esta operação consiste apenas em efetuar um cadastro da empresa e do responsável por esta que irá interagir diretamente com o sistema e que será responsável por todas as operações que a empresa irá efetuar.

CASO DE USO: Cadastrar empresa  
 ATOR: Empresa  
 SEQUÊNCIA NORMAL:  
 0. A empresa seleciona a opção “Cadastrar empresa” no menu principal  
 1. O sistema exibe tela de cadastro de empresa  
 2. A empresa entra com os dados do cadastro e submete ao sistema  
 3. Sistema valida dados e exibe tela de sucesso  
 SEQUÊNCIA ALTERNATIVA: Entrada de dados incorreta  
 Passo 2: Empresa não entra com os dados corretamente  
 Passo 3: Sistema exibe tela de erro de cadastro  
 SEQUÊNCIA ALTERNATIVA; Login já existente  
 Passo 2: A empresa entra com um login já existente  
 Passo 3: Sistema exibe tela de erro de login

Figura 35 – Cenário “cadastrar empresa”

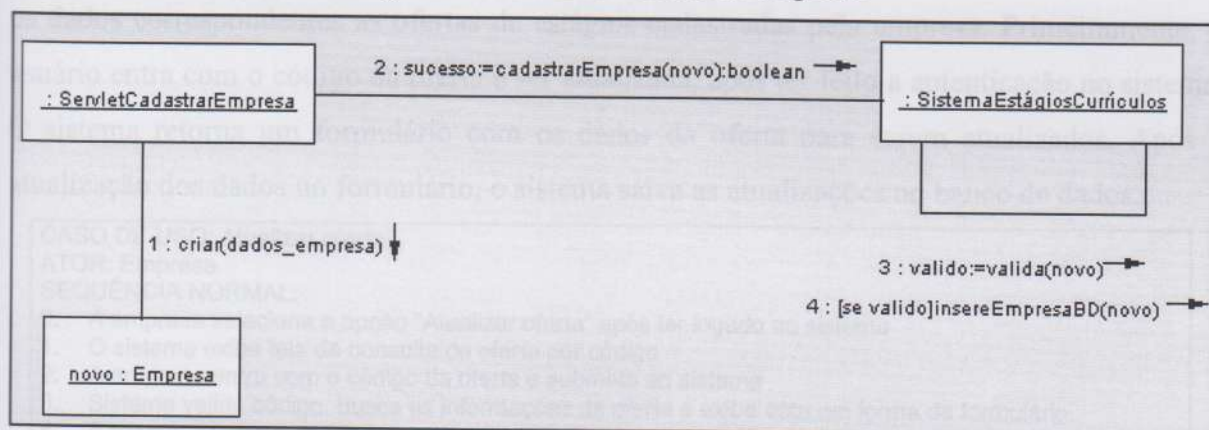


Figura 36 – Diagrama de colaboração “cadastrar empresa”

A opção “cadastrar ofertas de estágios” também está presente no módulo empresa. Nesta a empresa, após ter efetuado login, preencherá um formulário com os dados da oferta que será disponibilizada no sistema, conforme passos descritos na figura 37.

Figura 39 – Cenário “atualizar oferta”

CASO DE USO: Cadastrar oferta

ATOR: Empresa

SEQUÊNCIA NORMAL:

0. A empresa seleciona a opção "Cadastrar oferta" após ter logado no sistema
1. O sistema exibe tela de cadastro de oferta
2. A empresa entra com os dados do cadastro e submete ao sistema
3. O sistema valida dados e exibe tela de sucesso contendo o código da oferta

SEQUÊNCIA ALTERNATIVA: Entrada de dados incorreta

Passo 2: Empresa não entra com os dados corretamente

Passo 3: Sistema exibe tela de erro de cadastro

Figura 37 – Cenário "cadastrar ofertas de estágios"

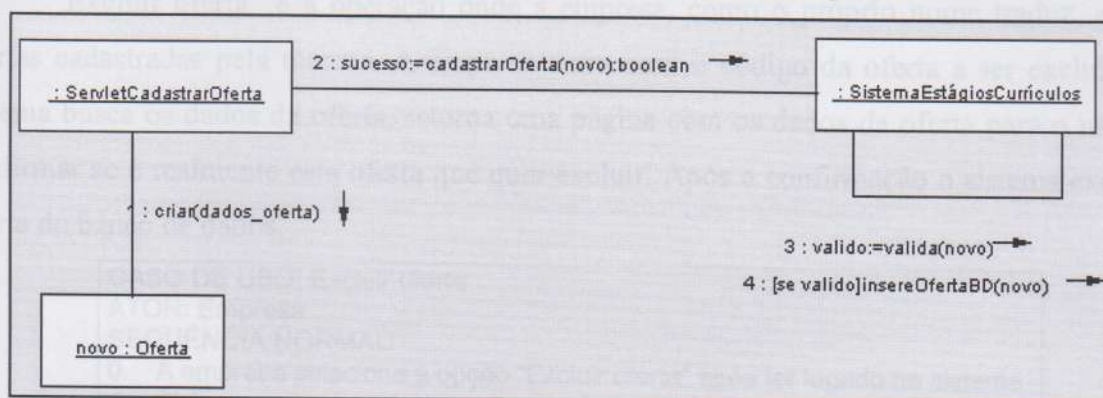


Figura 38 – Diagrama de colaboração "cadastrar ofertas de estágios"

"Atualizar oferta" é uma operação do módulo empresa onde poderão ser atualizados os dados correspondentes as ofertas de estágios cadastradas pela empresa. Primeiramente, o usuário entra com o código da oferta a ser atualizada, após ter feito a autenticação no sistema. O sistema retorna um formulário com os dados da oferta para serem atualizados. Após a atualização dos dados no formulário, o sistema salva as atualizações no banco de dados.

CASO DE USO: Atualizar oferta

ATOR: Empresa

SEQUÊNCIA NORMAL:

0. A empresa seleciona a opção "Atualizar oferta" após ter logado no sistema
1. O sistema exibe tela de consulta de oferta por código
2. A empresa entra com o código da oferta e submete ao sistema
3. Sistema valida código, busca as informações da oferta e exibe esta em forma de formulário
4. A empresa entra com os dados a serem atualizados e submete ao sistema
5. O sistema valida dados, atualiza os dados da oferta e exibe tela de sucesso

SEQUÊNCIA ALTERNATIVA: Entrada de dados inválida

Passo 4: A empresa não entra com os dados a serem atualizados corretamente

Passo 5: O sistema exibe tela de erro de entrada de dados

SEQUÊNCIA ALTERNATIVA: Esqueceu código

Passo 2: A empresa seleciona a opção "Esqueceu código"

Passo 3: O sistema exibe tela de consulta de código por email

Passo 4: A empresa entra com o email e submete ao sistema

Passo 5: O sistema valida email e exibe tela com o código da oferta

Figura 39 – Cenário "atualizar oferta"

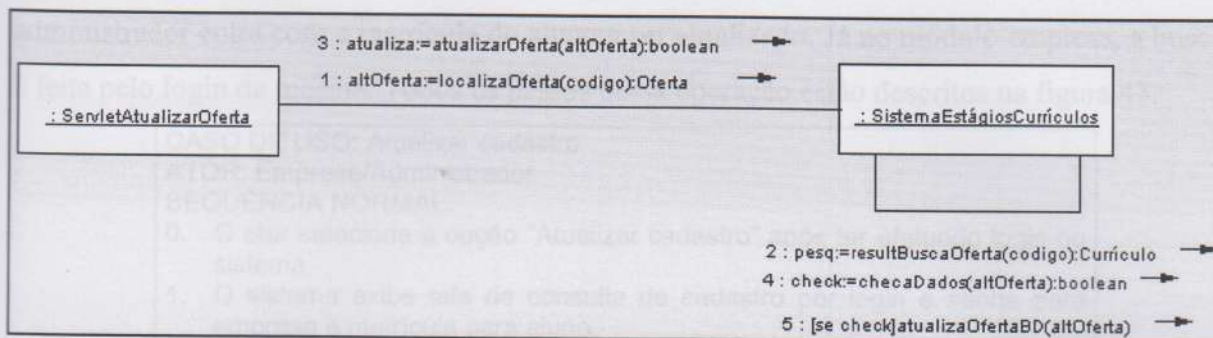


Figura 40 – Diagrama de colaboração “atualizar oferta”

“Excluir oferta” é a operação onde a empresa, como o próprio nome traduz, exclui ofertas cadastradas pela mesma. A empresa entra com o código da oferta a ser excluída, o sistema busca os dados da oferta, retorna uma página com os dados da oferta para o usuário confirmar se é realmente esta oferta que quer excluir. Após a confirmação o sistema exclui a oferta do banco de dados.

**CASO DE USO:** Excluir oferta  
**ATOR:** Empresa  
**SEQUÊNCIA NORMAL:**  
 0. A empresa seleciona a opção “Excluir oferta” após ter logado no sistema  
 1. Sistema exibe tela de consulta de oferta por código  
 2. A empresa entra com o código da oferta e submete ao sistema  
 3. O sistema valida código, exclui a oferta do banco de dados e exibe tela de sucesso  
**SEQUÊNCIA ALTERNATIVA:** Código de oferta incorreto  
 Passo 2: A empresa entra com o código da oferta incorreto  
 Passo 3: O sistema exibe tela de erro de código  
**SEQUÊNCIA ALTERNATIVA:** Esqueceu código  
 Passo 2: A empresa seleciona a opção “Esqueceu código”  
 Passo 3: O sistema exibe tela de consulta de código por email da empresa  
 Passo 4: A empresa entra com o email e submete ao sistema  
 Passo 5: O sistema valida email e exibe tela com o código da oferta

Figura 41 – Cenário “excluir oferta”

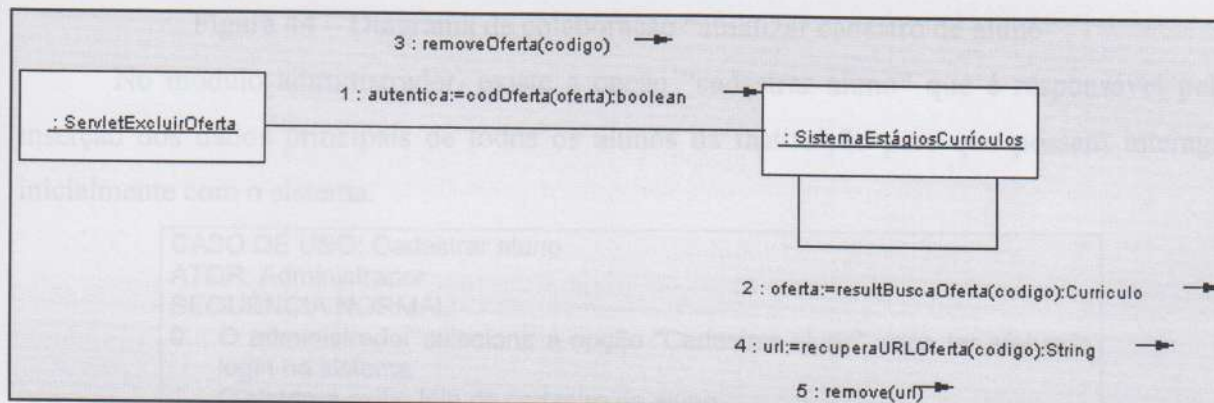


Figura 42 – Diagrama de colaboração “excluir oferta”

A opção “atualizar cadastro” está presente em dois módulos: empresa e administrador. No módulo administrador, a opção refere-se ao cadastro dos alunos da instituição. O

administrador entra com a matrícula do aluno a ser atualizado. Já no módulo empresa, a busca é feita pelo login da mesma. Todos os passos dessa operação estão descritos na figura 43.

CASO DE USO: Atualizar cadastro  
 ATOR: Empresa/Administrador  
 SEQUÊNCIA NORMAL:  
 0. O ator seleciona a opção "Atualizar cadastro" após ter efetuado login no sistema  
 1. O sistema exibe tela de consulta de cadastro por login e senha para empresa e matrícula para aluno.  
 2. O ator entra com os dados e submete ao sistema  
 3. O sistema valida dados e exibe os dados na forma de formulário  
 4. O ator entra com os dados a serem atualizados e submete ao sistema  
 5. Sistema valida dados, atualiza os dados no banco de dados e exibe tela de sucesso  
 SEQUÊNCIA ALTERNATIVA: Login/senha incorretos  
 Passo 3: O ator não entra com o login e a senha corretos  
 Passo 4: Sistema exibe tela de erro de login/senha  
 SEQUÊNCIA ALTERNATIVA: Matrícula incorreta  
 Passo 3: O ator não entra com a matrícula do aluno corretamente  
 Passo 4: Sistema exibe tela de erro de matrícula  
 SEQUÊNCIA ALTERNATIVA: Entrada de dados a serem atualizados incorreta  
 Passo 4: O ator não entra com os dados a serem atualizados corretamente  
 Passo 5: Sistema exibe tela de erro de atualização

Figura 43 – Cenário "atualizar cadastro"

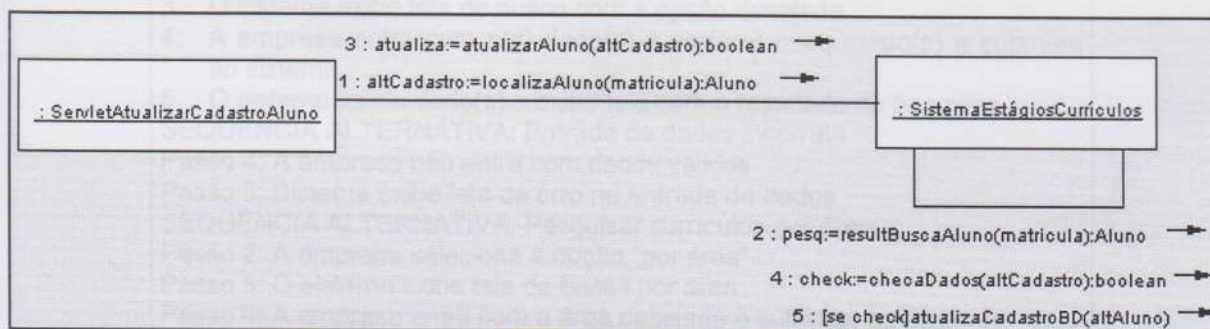


Figura 44 – Diagrama de colaboração "atualizar cadastro de aluno"

No módulo administrador, existe a opção "cadastrar aluno" que é responsável pela inserção dos dados principais de todos os alunos da instituição para que possam interagir inicialmente com o sistema.

CASO DE USO: Cadastrar aluno  
 ATOR: Administrador  
 SEQUÊNCIA NORMAL:  
 0. O administrador seleciona a opção "Cadastrar aluno" após ter efetuado login no sistema  
 1. O sistema exibe tela de cadastro de aluno  
 2. Administrador entra com os dados do aluno e submete ao sistema  
 3. Sistema valida dados e exibe tela de sucesso  
 SEQUÊNCIA ALTERNATIVA: Entrada de dados incorreta  
 Passo 2: O administrador não entrada com os dados do aluno corretamente  
 Passo 3: Sistema exibe tela de erro de cadastro

Figura 45 – Cenário "cadastrar aluno"

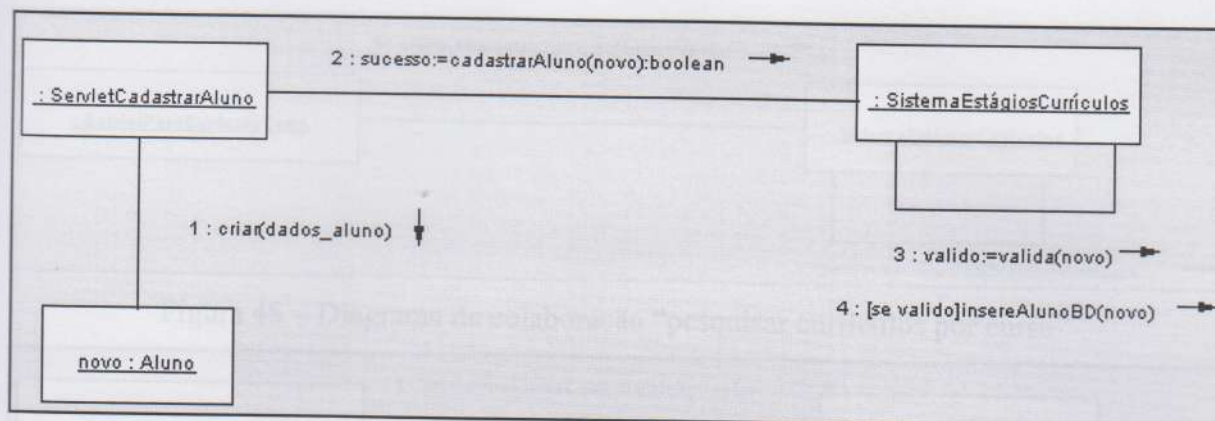


Figura 46 – Diagrama de colaboração “cadastrar aluno”

A operação “pesquisar currículos” possui o mesmo funcionamento e opções de busca da operação “pesquisar ofertas de estágios”. Os passos a serem seguidos nesta operação estão descritos na figura 47.

CASO DE USO: Pesquisar currículos  
 ATOR: Empresa  
 SEQUÊNCIA NORMAL:

0. A empresa seleciona a opção “Pesquisar currículos”
1. O sistema exibe tela de pesquisa com três opções: por curso, por área e por curso e período
2. A empresa seleciona a opção desejada
3. O sistema exibe tela de busca com a opção desejada
4. A empresa entra com o(s) dado(s) a ser(em) pesquisado(s) e submete ao sistema
5. O sistema valida dado(s) e exibe tela com o resultado da pesquisa

SEQUÊNCIA ALTERNATIVA: Entrada de dados incorreta

Passo 4: A empresa não entra com dados válidos

Passo 5: Sistema exibe tela de erro na entrada de dados

SEQUÊNCIA ALTERNATIVA: Pesquisar currículos por área

Passo 2: A empresa seleciona a opção “por área”

Passo 3: O sistema exibe tela de busca por área

Passo 4: A empresa entra com a área desejada e submete ao sistema

Passo 5: Sistema valida área e exibe tela contendo o resultado da pesquisa por área

SEQUÊNCIA ALTERNATIVA: Pesquisar currículos por curso

Passo 2: A empresa seleciona a opção “por curso”

Passo 3: O sistema exibe tela de busca por curso

Passo 4: A empresa entra com o curso desejado e submete ao sistema

Passo 5: Sistema valida curso e exibe tela contendo o resultado da pesquisa por curso

SEQUÊNCIA ALTERNATIVA: Pesquisar currículos por curso e período

Passo 2: A empresa seleciona a opção “por curso e período”

Passo 3: O sistema exibe tela de busca por curso e período

Passo 4: A empresa entra com o curso e o período desejados e submete ao sistema

Passo 5: Sistema valida curso e período e exibe tela contendo o resultado da pesquisa

Figura 47– Cenário “pesquisar currículos”

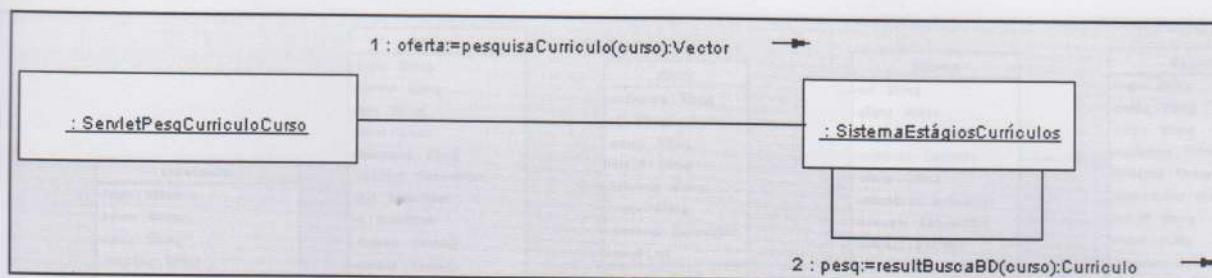


Figura 48 – Diagrama de colaboração "pesquisar currículos por curso"

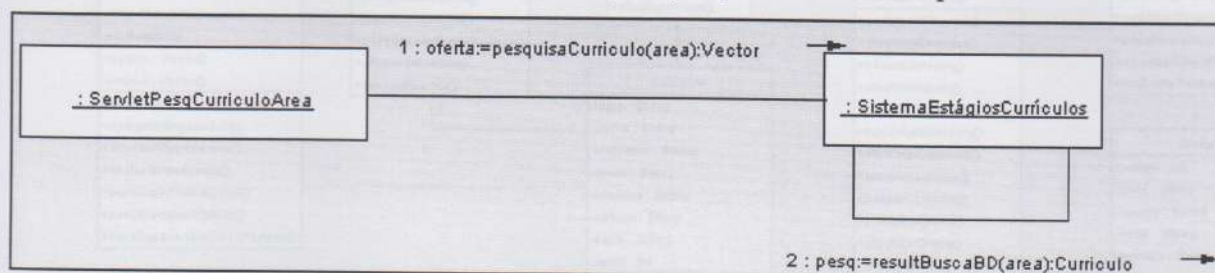


Figura 49 – Diagrama de colaboração "pesquisar currículos por área"

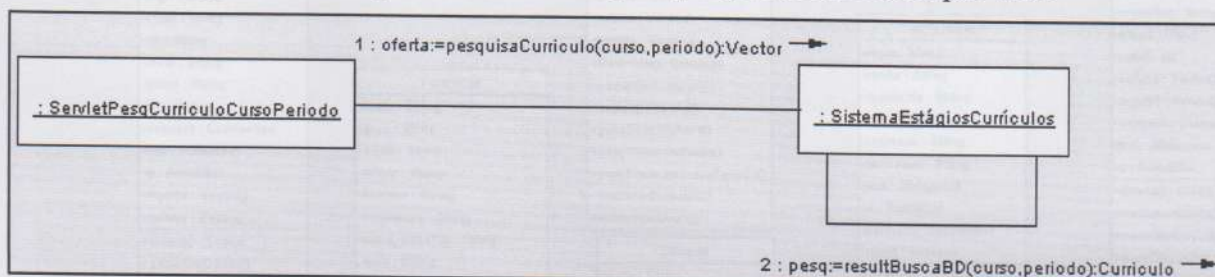


Figura 50 – Diagrama de colaboração "pesquisar currículos por curso e período"

#### 4.1.5. Diagrama de Classes

O diagrama de classes do sistema, mostrado na figura 51, descreve todas as classes que fazem parte do sistema, bem como todos os atributos e métodos das classes, além da cardinalidade das mesmas.

Figura 51 – Diagrama de classes do sistema de estágios e currículos

A classe `Sistema` é responsável pela conexão com o banco de dados, além de servir como ponte de ligação de todas as classes. É a classe principal do sistema, onde todas as mensagens são invocadas a partir de objetos criados das outras classes.

A classe `Aluno` é a classe responsável pelo cadastro e pela atualização dos dados dos alunos.

`Curriculo`, `Evento`, `Formação`, `Experiência`, `Cursos e Informa` são responsáveis pelo cadastro, pela atualização, pela exclusão e pela busca dos dados dos currículos no banco de dados.

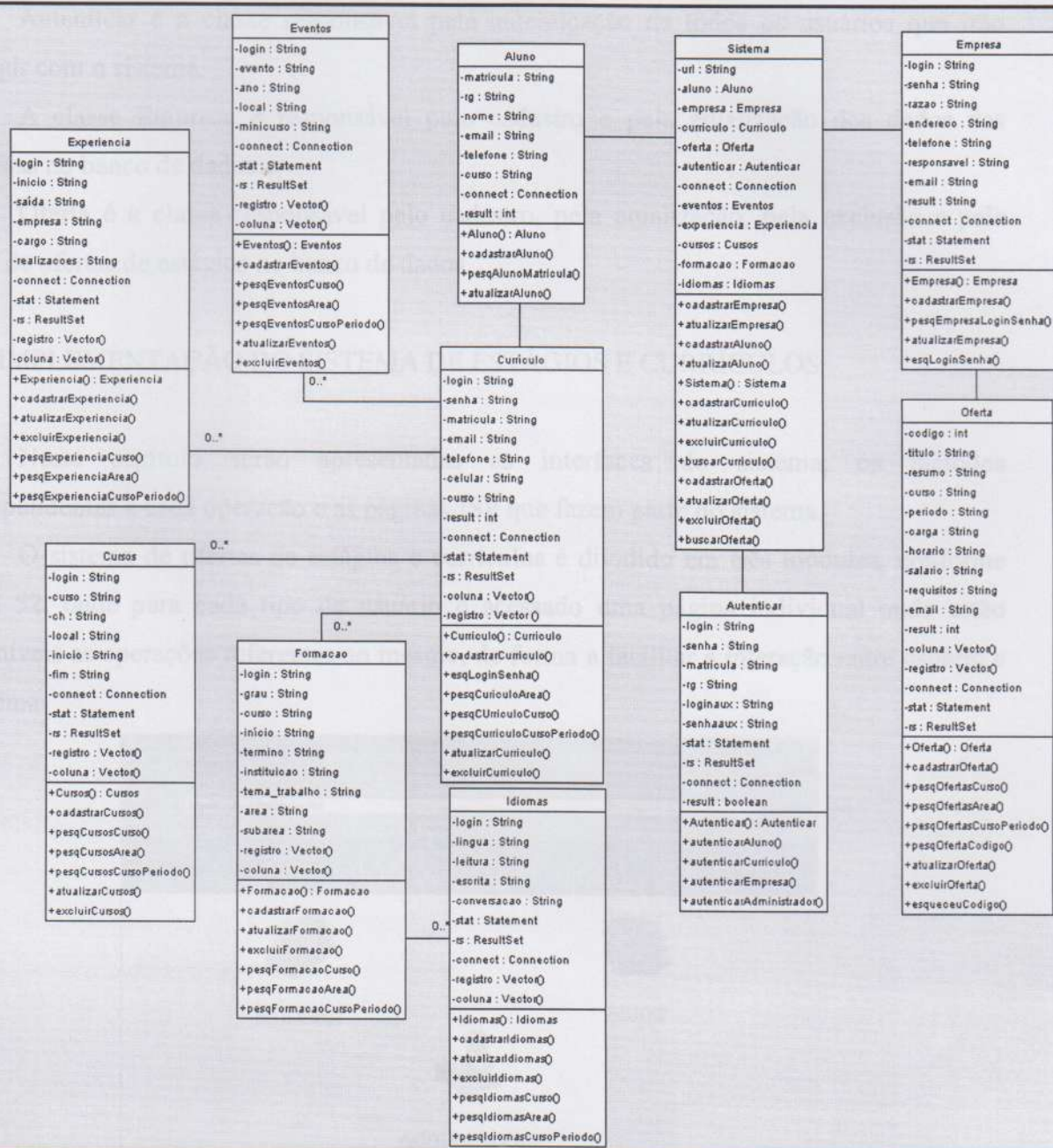


Figura 51 – Diagrama de classes do sistema de estágios e currículos

A classe Sistema é responsável pela conexão com o banco de dados, além de servir como ponte de ligação de todas as classes. É a classe principal do sistema, onde todas os métodos são invocados a partir de objetos criados das outras classes.

A classe Aluno é a classe responsável pelo cadastro e pela atualização dos dados dos alunos.

Curriculum, Events, Formation, Experience, Courses and Languages are responsible for the registration, for the update, for the exclusion and for the search of the data of the curricula in the data bank.

Autenticar é a classe responsável pela autenticação de todos os usuários que irão interagir com o sistema.

A classe Empresa é responsável pelo cadastro e pela atualização dos dados das empresas no banco de dados.

Oferta é a classe responsável pelo cadastro, pela atualização, pela exclusão e pela busca de ofertas de estágios no banco de dados.

## 4.2. IMPLEMENTAÇÃO DO SISTEMA DE ESTÁGIOS E CURRÍCULOS

Neste capítulo serão apresentadas as interfaces do sistema, os métodos correspondentes a cada operação e as páginas JSP que fazem parte do sistema.

O sistema de ofertas de estágios e currículos é dividido em três módulos, conforme figura 52, onde para cada tipo de usuário é acessado uma página individual onde estão disponíveis as operações referentes ao mesmo, de forma a facilitar a interação entre usuário e o sistema.

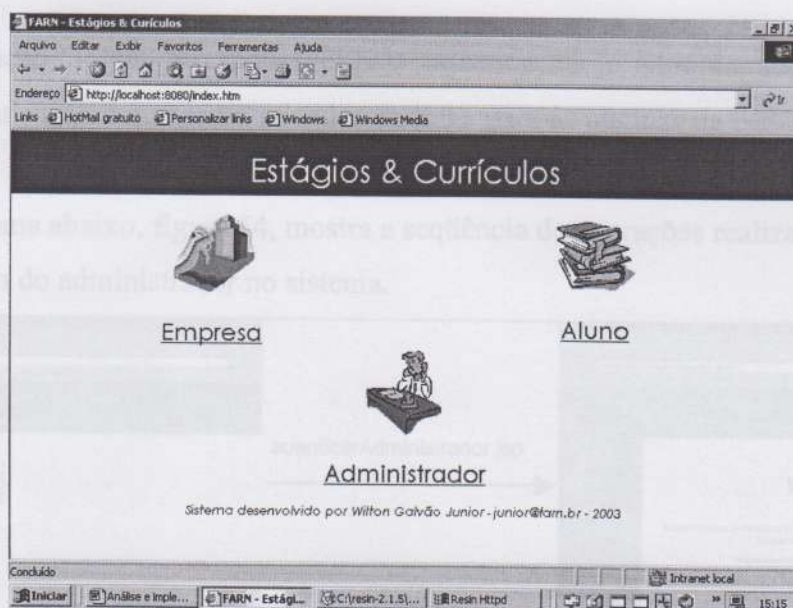


Figura 52 – Visão modular do sistema de estágios e currículos

### 4.2.1. Interfaces do Sistema

#### 4.2.1.1. Módulo administrador

O módulo administrador é responsável pelo gerenciamento dos alunos que irão interagir com o sistema. Suas operações são: cadastrar aluno, atualizar aluno e login.

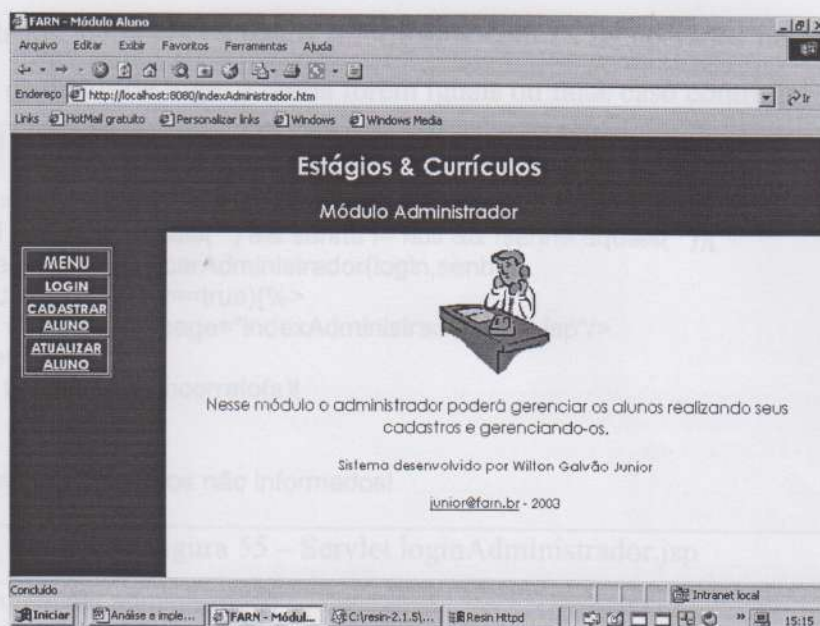


Figura 53 – Tela principal do módulo administrador

#### 4.2.1.1.1. Login

A operação “login” é responsável pela autenticação do administrador no sistema. Só após essa autenticação é que serão ativados os links para as páginas de cadastro e atualização de aluno.

O esquema abaixo, figura 54, mostra a seqüência de operações realizadas para que seja efetuado o login do administrador no sistema.

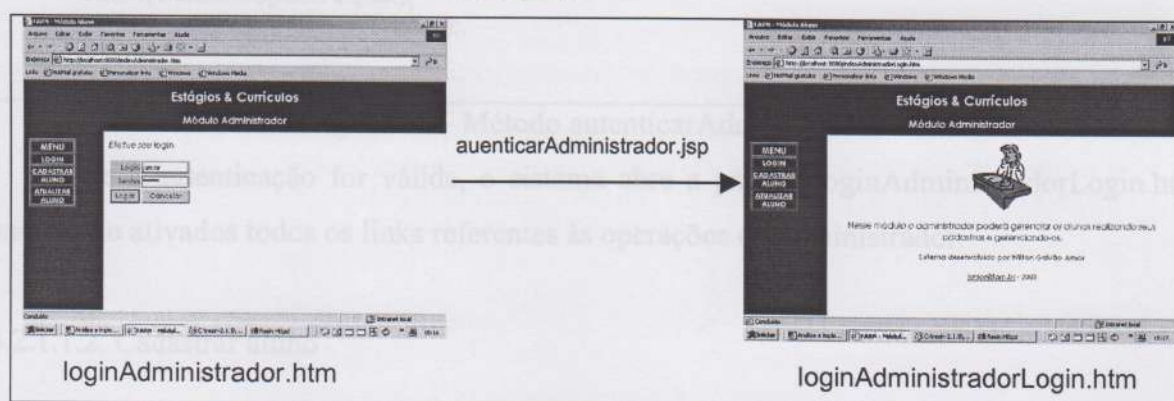


Figura 54 – Seqüência de operações para login do administrador

Ao selecionar a opção login no menu do módulo, o sistema abre a página loginAdministrador.htm que contém um formulário com os campos login e senha. Após entrar com os dados, o sistema ativa o servlet loginAdministrador.jsp (figura 55) que invoca o método autenticarAdministrador da classe Autenticar (figura 56) que busca o login e a senha

no banco de dados, compara os dados informados com os recuperados no banco de dados e retorna true se os dados de login e senha forem iguais ou false caso contrário.

```
<% Sistema sist = new Sistema();
String login = request.getParameter("login");
String senha = request.getParameter("senha");
if(login != null && !login.equals("") && senha != null && !senha.equals("")){
    sist.autenticar.autenticarAdministrador(login,senha);
    if(sist.autenticar.result==true){%>
        <jsp:forward page="indexAdministradorLogin.jsp"/>
    }else{%>
        Login/senha incorreto(s)!
    }
} else { %>
    Parâmetros obrigatórios não informados!
}%>
```

Figura 55 – Servlet loginAdministrador.jsp

```
//Método para autenticar administrador
public void autenticarAdministrador(String login,String senha){
    try{
        this.login = login;
        this.senha = senha;
        stat = connect.createStatement();
        rs = stat.executeQuery("SELECT login,senha FROM Administrador WHERE
            login='"+login+"'");
        if( rs.next() ){
            loginaux = rs.getString(1).trim();
            senhaaux = rs.getString(2).trim();
            result = autenticacao();
        } else {
            result = false;
        }
        stat.close();
    }
    catch(SQLException sqllex){
        sqllex.printStackTrace();
    }
}
```

Figura 56 – Método autenticarAdministrador

Se a autenticação for válida, o sistema abre a página loginAdministradorLogin.htm onde estão ativados todos os links referentes às operações do administrador.

#### 4.2.1.1.2. Cadastrar aluno

Na operação “cadastrar aluno”, o administrador entra com os dados do aluno para serem cadastrados no banco de dados, para permitir que o mesmo possa interagir com o sistema. Os alunos só poderão cadastrar seus currículos após terem feito a autenticação no sistema com a matrícula e o RG que serão cadastrados nessa operação.

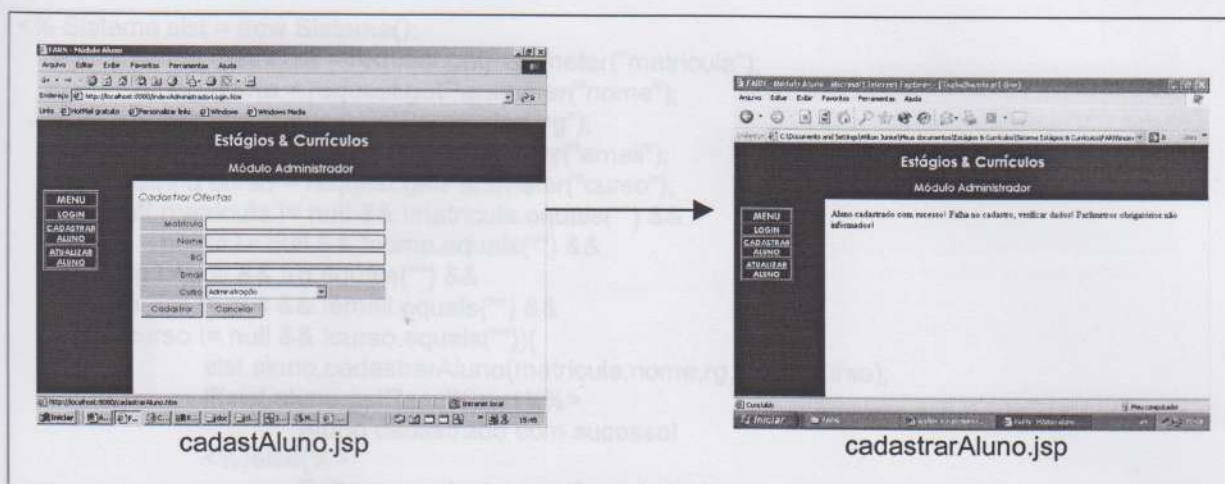


Figura 57 – Sequência de operações da opção cadastrar aluno

O administrador seleciona a opção “cadastrar aluno” no menu principal. O sistema executa a página `cadastrarAluno.htm` para que o administrador entre com os dados referentes ao aluno. Após digitar os dados, é acionado o servlet `cadastrarAluno.jsp` que utiliza o método `cadastrarAluno` da classe `Aluno` (figura 58). Esse método possui como parâmetros: matrícula, nome, RG e email.

O servlet `cadastrarAluno.jsp` (figura 59) recebe os dados informados no formulário da página de cadastro, testa se todos os parâmetros foram digitados e aciona o método da classe `Aluno`, se todos forem digitados corretamente. O método `cadastrarAluno` armazena os dados no banco de dados e guarda o resultado da operação na variável `result`.

O servlet recupera o valor dessa variável através do método `getResult()` e dependendo do resultado mostra a mensagem de sucesso ou falha.

```
//Método para cadastrar um aluno no banco de dados
public void cadastrarAluno(String matricula,String nome,String email,String telefone,String curso){
    try{
        stat = connect.createStatement();
        result = stat.executeUpdate("INSERT INTO Aluno(matricula,nome,email,
            telefone,curso) VALUES ('"+matricula+"','"+nome+"','"+email+"',
            '"+telefone+"','"+curso+"')");
        stat.close();
    }
    catch(SQLException sqlex){
        sqlex.printStackTrace();
    }
}
}
```

Figura 58 – Método `cadastrarAluno`

```

<% Sistema sist = new Sistema();
String matricula = request.getParameter("matricula");
String nome = request.getParameter("nome");
String rg = request.getParameter("rg");
String email = request.getParameter("email");
String curso = request.getParameter("curso");
if( matricula != null && !matricula.equals("") &&
    nome != null && !nome.equals("") &&
    rg != null && !rg.equals("") &&
    email != null && !email.equals("") &&
    curso != null && !curso.equals("")){
    sist.aluno.cadastrarAluno(matricula,nome,rg,email,curso);
    if(sist.aluno.getResult()==1){%>
        Aluno cadastrado com sucesso!
    }else{%>
        Falha no cadastro, verificar dados!
    }
} else {%>
    Parâmetros obrigatórios não informados!
}%> %>

```

Figura 59 – Servlet cadastrarAluno.jsp

#### 4.2.1.1.3. Atualizar aluno

A operação “atualizar aluno” é responsável pela atualização de dados dos alunos. Ao selecionar a opção de atualização é aberta uma página de pesquisa de aluno por matrícula, `pesqAlunoMatricula.htm`, onde o administrador entra com a matrícula correspondente ao aluno. O sistema aciona o servlet `buscarAluno.jsp` (figura 61) que aciona o método `pesqAlunoMatricula` (figura 62) da classe `Aluno`. Esse método busca os dados do aluno relativo a matrícula informada e retorna-os em forma de formulário. No formulário esses dados são acessados através dos métodos `get` da classe `aluno`.

Após a atualização dos dados, é invocado o servlet `atualizarAluno.jsp` (figura 63) que aciona o método `atualizarAluno` (figura 64) da mesma classe, que recebe como parâmetros os dados digitados no formulário e os envia ao banco de dados para serem atualizados conforme matrícula informada, retornando uma mensagem de sucesso ou falha.

```

<input type="text" name="matricula" value="<%=sist.aluno.getMatricula()%>" />
<input type="text" name="nome" value="<%=sist.aluno.getNome()%>" />
<input type="text" name="rg" value="<%=sist.aluno.getRg()%>" />
<input type="text" name="email" value="<%=sist.aluno.getEmail()%>" />
<input type="text" name="curso" value="<%=sist.aluno.getCurso()%>" />
<input type="button" value="Atualizar" />
<input type="button" value="Cancelar" />

```

Figura 61 – Servlet buscarAluno.jsp

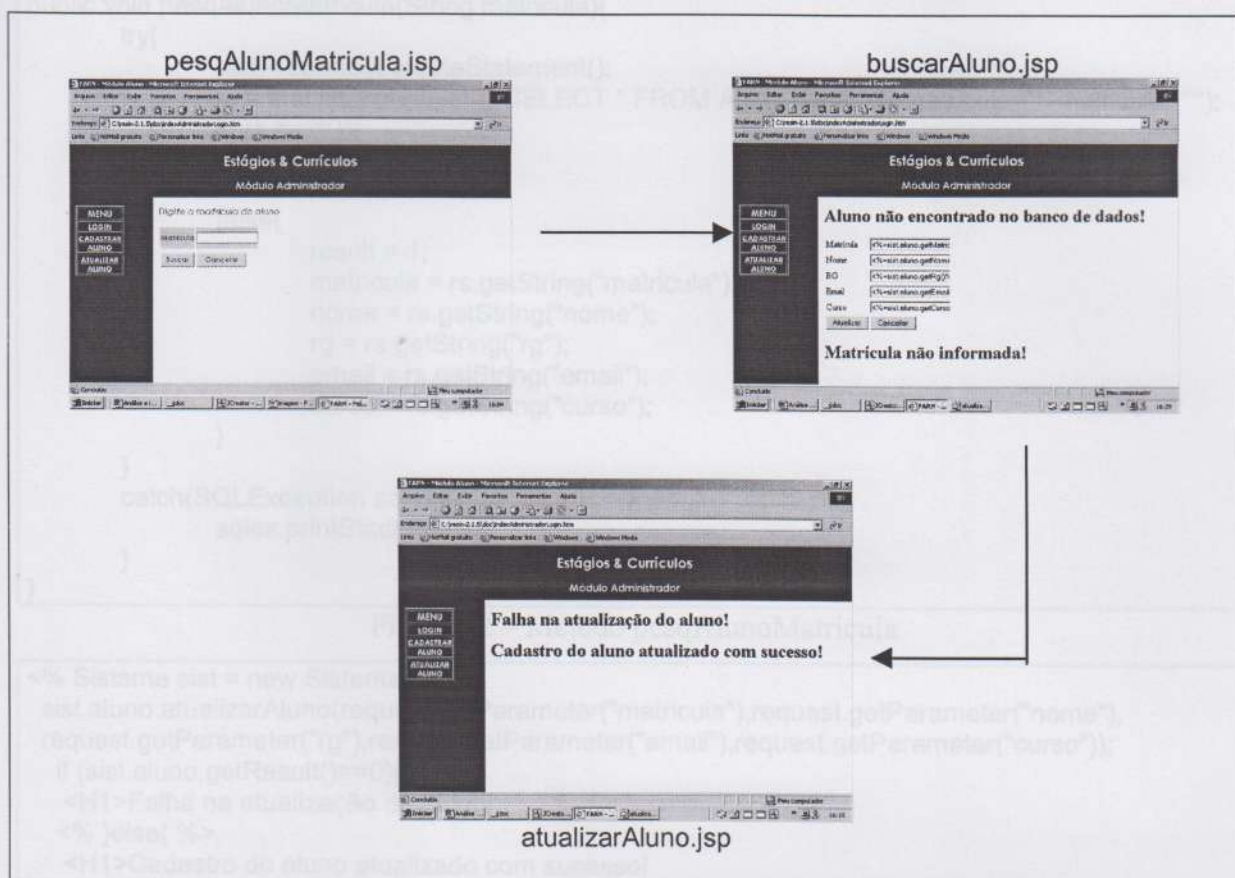


Figura 60 – Sequência de operações da opção atualizar aluno

```

<% Sistema sist = new Sistema();
String matricula = request.getParameter("matricula");
if (matricula != null && !matricula.equals("")){
    sist.aluno.pesqAlunoMatricula(matricula);
    if (sist.empresa.getResult()==0){ %>
        <H1>Aluno não encontrado no banco de dados!
    }else{ %>
        <form method="post" action="atualizarAluno.jsp">
            <table>
                <tr><td>Matrícula</td>
                <td><input type="text" name="matricula" value="<%=sist.aluno.getMatricula()%>"></td>
            <tr><td>Nome</td>
                <td><input type="text" name="nome" value="<%=sist.aluno.getNome()%>"></td>
            <tr><td>RG</td>
                <td><input type="text" name="rg" value="<%=sist.aluno.getRg()%>"></td>
            <tr><td>Email</td>
                <td><input type="text" name="email" value="<%=sist.aluno.getEmail()%>"></td>
            <tr><td>Curso</td>
                <td><input type="text" name="curso" value="<%=sist.aluno.getCurso()%>"></td>
            <tr><td><input type="submit" value="Atualizar"></td>
            <td><input type="reset" value="Cancelar"></td>
            </tr>
            </table>
        </form>
    } else {%>
        Matrícula não informada!
    } %>
} else {%>
    Matrícula não informada!
} %>

```

Figura 61 – Servlet buscarAluno.jsp

```
//Método que pesquisa um aluno pela matrícula
public void pesqAlunoMatricula(String matricula){
    try{
        stat = connect.createStatement();
        rs = stat.executeQuery("SELECT * FROM Aluno WHERE matricula='"+matricula+"'");
        if(!rs.next()){
            result = 0;
            return;
        }else{
            result = 1;
            matricula = rs.getString("matricula");
            nome = rs.getString("nome");
            rg = rs.getString("rg");
            email = rs.getString("email");
            curso = rs.getString("curso");
        }
    }
    catch(SQLException sqlex){
        sqlex.printStackTrace();
    }
}
```

Figura 62 – Método pesqAlunoMatricula

```
<% Sistema sist = new Sistema();
sist.aluno.atualizarAluno(request.getParameter("matricula"),request.getParameter("nome"),
request.getParameter("rg"),request.getParameter("email"),request.getParameter("curso"));
if (sist.aluno.getResult()==0){ %>
    <H1>Falha na atualização do aluno!
<% }else{ %>
    <H1>Cadastro do aluno atualizado com sucesso!
<% } %>
```

Figura 63 – Servlet atualizarAluno.jsp

```
//Método para atualizar o cadastro de um aluno
public void atualizarAluno(String matricula,String nome,String rg,String email,String curso){
    try{
        stat = connect.createStatement();
        result = stat.executeUpdate("UPDATE Aluno SET (matricula='"+matricula+"',
nome='"+nome+"',rg='"+rg+"',email='"+email+"',curso='"+curso+"') WHERE
matricula='"+matricula+"'");
        stat.close();
    }
    catch(SQLException sqlex){
        sqlex.printStackTrace();
    }
}
```

Figura 64 – Método atualizarAluno

#### 4.2.1.2.Módulo aluno

Nesse módulo, o único ator a interagir com o sistema é o aluno. Neste o aluno poderá cadastrar e gerenciar seu currículo, após autenticação, além de pesquisar ofertas de estágios disponibilizadas pelas empresas. As principais operações desse módulo são: login, cadastrar

currículo, atualizar currículo, excluir currículo e buscar ofertas de estágios por: curso, área e curso e período.

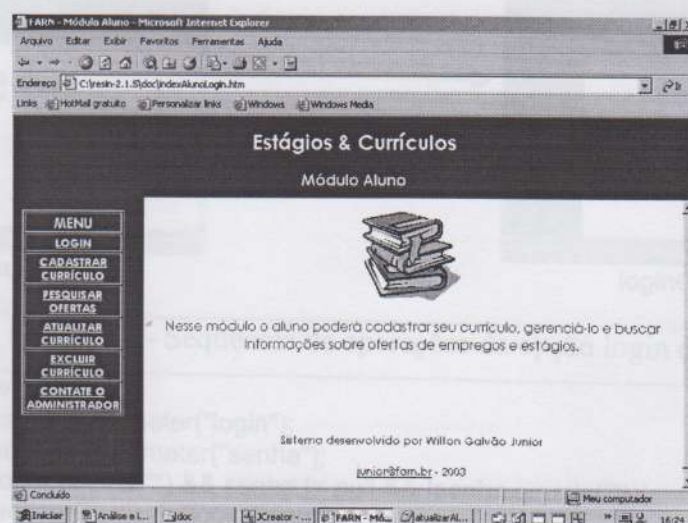


Figura 66 – Tela principal do módulo aluno

#### 4.2.1.2.1. Login

Assim como no módulo administrador, a operação “login” no módulo aluno, possui a mesma sequência de passos, alterando somente o nome do método de autenticação.

Primeiramente, o aluno seleciona a opção “login” no menu. O sistema aciona a página loginCurriculo.htm que possui os mesmos parâmetros do login de administrador. Essa página aciona o servlet loginCurriculo.jsp (figura 68) que contém o método autenticarCurriculo (figura 69) da classe Autenticar. O servlet primeiro verifica se os dados foram digitados para só após acionar o método da classe.

Assim como no login de administrador, o sistema compara os dados informados com os recuperados do banco de dados, retornando true se comparação for verdadeira, false se não. Se a comparação for verdadeira, o sistema aciona a página loginCurriculoLogin.htm onde estão acionados todos os links referentes às operações que o aluno pode fazer no seu currículo.

Figura 69 – Método autenticarCurriculo

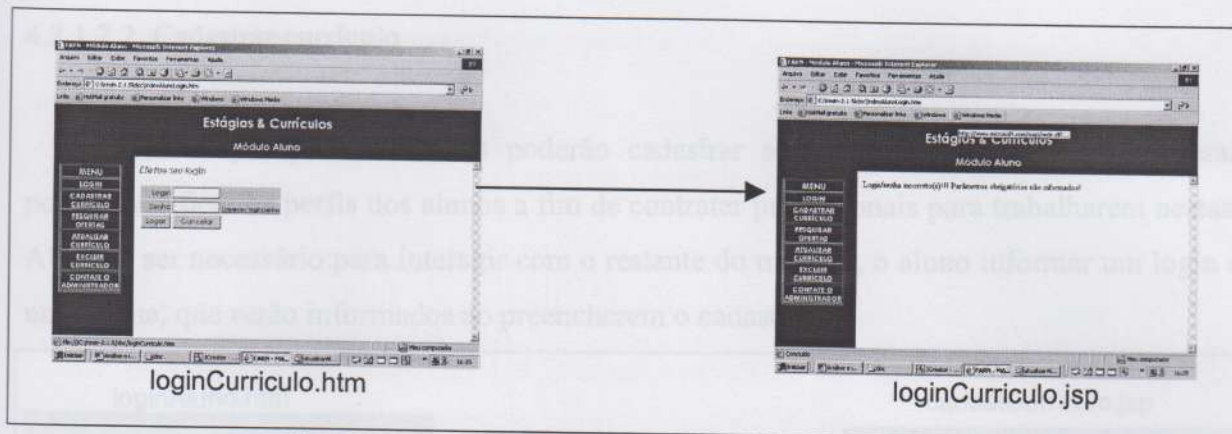


Figura 67 – Sequência de operações da opção login de aluno

```
<% Sistema sist = new Sistema();
String login = request.getParameter("login");
String senha = request.getParameter("senha");
if (login != null && !login.equals("") && senha != null && !senha.equals("")){
    sist.autenticar.autenticarCurriculo(login,senha);
    if(sist.autenticar.result==true){ %>
        <jsp:forward page="indexAlunoLogin.htm"/>
    <% }else{ %>
        Login/senha incorreto(s)!!!
    <% }
}else{ %>
    Parâmetros obrigatórios não informados!
<% } %>
```

Figura 68 – Servlet loginCurriculo.jsp

```
//Método para autenticar currículo
public void autenticarCurriculo(String login,String senha){
    try{
        this.login = login;
        this.senha = senha;
        stat = connect.createStatement();
        rs = stat.executeQuery("SELECT login,senha FROM Curriculo WHERE login="
            +login+"");
        if( rs.next() ){
            loginaux = rs.getString(1).trim();
            senhaaux = rs.getString(2).trim();
            result = autenticacao();
        } else {
            result = false;
        }
        stat.close();
    }
    catch(SQLException sqlex){
        sqlex.printStackTrace();
    }
}
```

Figura 69 – Método autenticarCurriculo

#### 4.2.1.2.2. Cadastrar currículo

Nessa operação, os alunos poderão cadastrar seus currículos para que empresas possam verificar os perfis dos alunos a fim de contratar profissionais para trabalharem nestas. Além de ser necessário para interagir com o restante do módulo, o aluno informar um login e uma senha, que serão informados ao preencherem o cadastro.

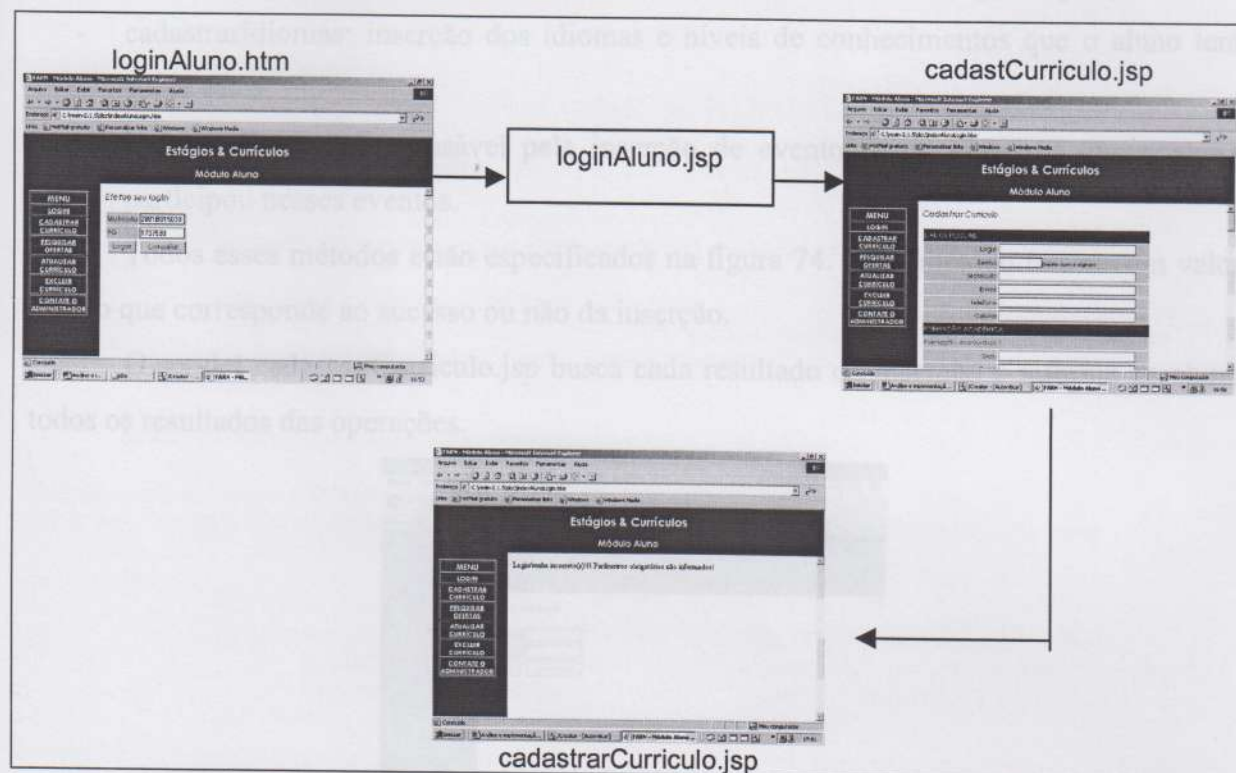


Figura 70 – Sequência de operações da opção cadastrar currículo

Inicialmente, ao acessar a opção “cadastrar currículo”, o sistema abre a página loginAluno.htm (figura 71). Esta página é responsável pela autenticação do aluno para que o mesmo possa preencher os dados referentes ao seu currículo e para garantir que apenas os alunos da instituição, cadastrados pelo administrador, tenham acesso a esta operação. O aluno entra com a matrícula e o rg na tela de autenticação. Logo após o sistema aciona o servlet loginAluno.jsp (figura 72) que chama o método autenticarAluno (figura 73) da classe autenticar que possui o mesmo funcionamento dos outros métodos de login desta classe.

O servlet loginAluno.jsp busca o resultado da autenticação e se for verdadeiro executa a página cadastrarCurriculo.htm (figura 75), onde o aluno preencherá todos os dados referentes ao seu currículo. Após o preenchimento, é acionado o servlet cadastrarCurriculo.jsp que executa os métodos referentes ao cadastro do currículo no banco de dados, testando se os parâmetros obrigatórios foram todos preenchidos.

Os métodos de cadastro de currículo são:

- cadastrarDados: responsável pelo cadastramento de dados pessoais do aluno no banco de dados;
- cadastrarFormacao: responsável pela inserção da formação acadêmica do aluno no banco de dados;
- cadastrarExperiencia: cadastramento das experiências profissionais do aluno;
- cadastrarCurso: responsável pelos cursos de extensão que o aluno participou;
- cadastrarIdiomas: inserção dos idiomas e níveis de conhecimentos que o aluno tem sobre estes;
- cadastrarEvento: responsável pela inserção de eventos e os micursos que o aluno participou nesses eventos.

Todos esses métodos estão especificados na figura 74. Cada método retorna um valor inteiro que corresponde ao sucesso ou não da inserção.

O servlet cadastrarCurriculo.jsp busca cada resultado da inserção e informa ao aluno todos os resultados das operações.

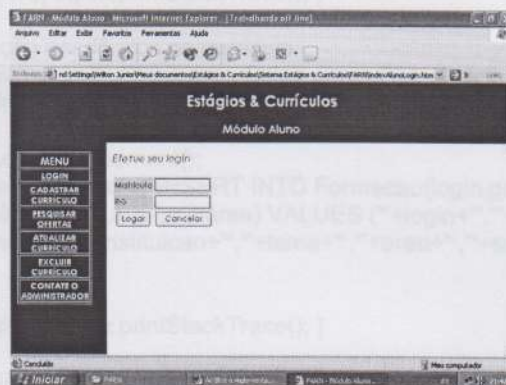


Figura 71 – loginAluno.htm

```
<% Sistema sist = new Sistema();
String matricula = request.getParameter("matricula");
String rg = request.getParameter("rg");
if (matricula != null && !matricula.equals("") && rg != null && !rg.equals("")){
    sist.autenticar.autenticarAluno(matricula,rg);
    if(sist.autenticar.result==true){%>
        <jsp:forward page="cadastrarCurriculo.htm"/>
    }else{ %>
        Matrícula e/ou RG inválido(s)!
    } %>
}else{ %>
    Parâmetros obrigatórios não informados!
}%>
```

Figura 72 – Servlet loginAluno.jsp

```

public void autenticarAluno(String matricula,String rg){
    try{
        login = matricula;
        senha = rg;
        stat = connect.createStatement();
        rs = stat.executeQuery("SELECT matricula,rg FROM Aluno WHERE
            matricula='"+login+"'");
        if( rs.next() ){
            loginaux = rs.getString(1).trim();
            senhaux = rs.getString(2).trim();
            result = autenticacao();
        } else { result = false; }
        stat.close(); }
        catch(SQLException sqlex){ sqlex.printStackTrace();}
    }
}

```

Figura 73 – Método autenticarAluno

```

//Método para cadastrar os dados pessoais no currículo
public void cadastrarDados(String log,String s,String mat,String mail,String tel,String cel){
    try{
        stat = connect.createStatement();
        result = stat.executeUpdate("INSERT INTO curriculo(login,senha,matricula,email,telefone,
            celular) VALUES ('"+log+"','"+s+"','"+mat+"','"+mail+"','"+tel+"','"+cel+"')");
        stat.close();
    }
    catch(SQLException sqlex){ sqlex.printStackTrace(); }
}

//Método para cadastrar a formação acadêmica no currículo
public void cadastrarFormacao(String login,String grau,String curso,String inicio,String termino,String
    instituicao,String tema,String area,String subarea){
    try{
        stat = connect.createStatement();
        result = stat.executeUpdate("INSERT INTO Formacao(login,grau,curso,inicio,termino,
            instituicao,tema,area,subarea) VALUES ('"+login+"','"+grau+"','"+curso+"','"+inicio+"',
            '"+termino+"','"+instituicao+"','"+tema+"','"+area+"','"+subarea+"')");
        stat.close();
    }
    catch(SQLException sqlex){ sqlex.printStackTrace(); }
}

//Método para cadastrar a experiência profissional no currículo
public void cadastrarExperiencia(String login,String inicio,String saida,String empresa,String cargo,String
    realizacoes){
    try{
        stat = connect.createStatement();
        result = stat.executeUpdate("INSERT INTO Experiencia(login,inicio,saida,empresa,
            cargo,realizacoes) VALUES ('"+login+"','"+inicio+"','"+saida+"','"+empresa+"',
            '"+cargo+"','"+realizacoes+"')");
        stat.close();
    }
    catch(SQLException sqlex){ sqlex.printStackTrace(); }
}

//Método para cadastrar cursos de extensão no currículo
public void cadastrarCurso(String login,String curso,String ch,String local,String inicio,String fim){
    try{
        stat = connect.createStatement();
        result = stat.executeUpdate("INSERT INTO Cursos(login,curso,ch,local,inicio,fim) VALUES
            ('"+login+"','"+curso+"','"+ch+"','"+local+"','"+inicio+"','"+fim+"')");
        stat.close();
    }
    catch(SQLException sqlex){ sqlex.printStackTrace(); }
}
}

```

Figura 74 – Métodos de inserção de dados do currículo

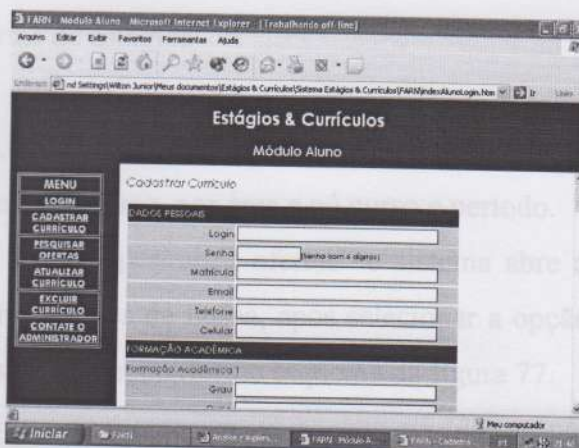


Figura 75 - cadastrCurrículo.jsp

#### 4.2.1.2.3. Atualizar currículo

A opção “atualizar currículo” é responsável pela atualização dos dados no banco de dados referente ao currículo do aluno. Inicialmente, o aluno seleciona a opção “atualizar currículo”. O sistema executa a página `pesqCurrículoAtualizar.htm` onde o aluno deve digitar o login e a senha correspondente ao currículo. Após digitar os dados, o sistema executa o servlet “`atualizarCurrículo.jsp`” que invoca o método `atualizarCurrículo` da classe `Currículo` que insere as alterações no banco de dados.

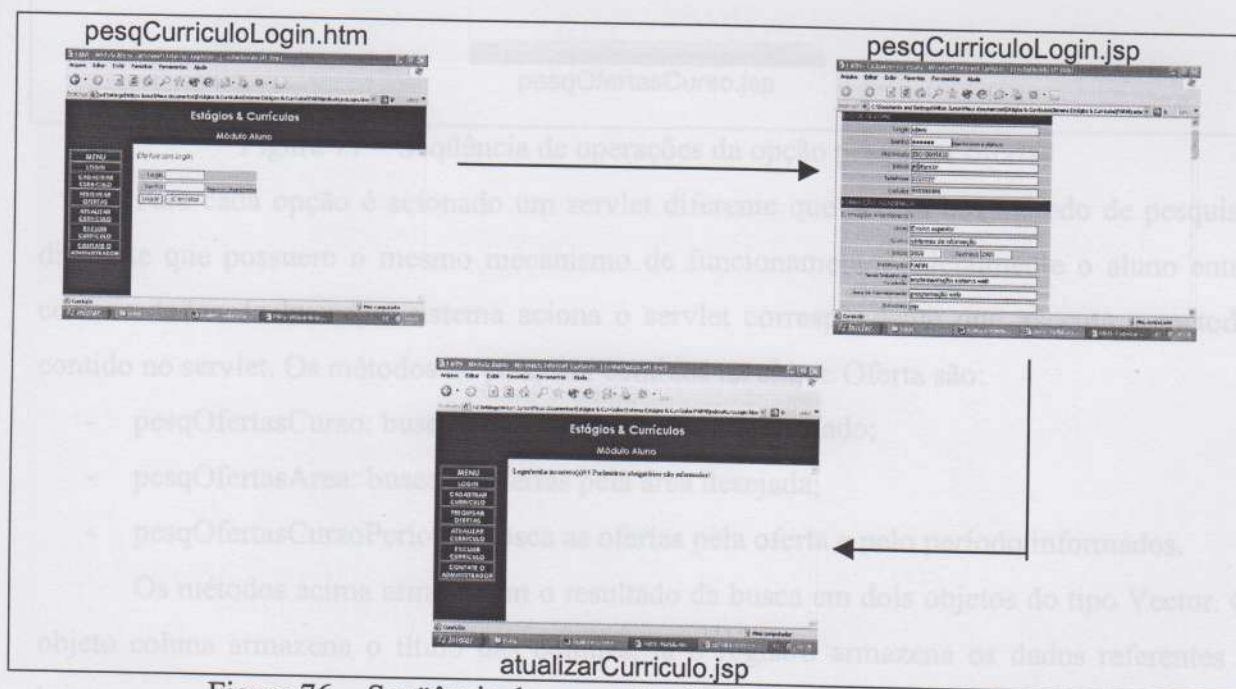


Figura 76 - Sequência de operações da opção atualizar currículo

#### 4.2.1.2.4. Pesquisar ofertas de estágios

Nessa operação, o aluno pode pesquisar ofertas de estágios disponibilizadas pelas empresas, em três opções: por curso, por área e pó curso e período.

Ao selecionar a opção “pesquisar ofertas” o sistema abre a página `pesqOfertas.htm` onde estão listadas as três opções de busca, após selecionar a opção o sistema abre a página com o formulário correspondente, conforme esquema da figura 77.

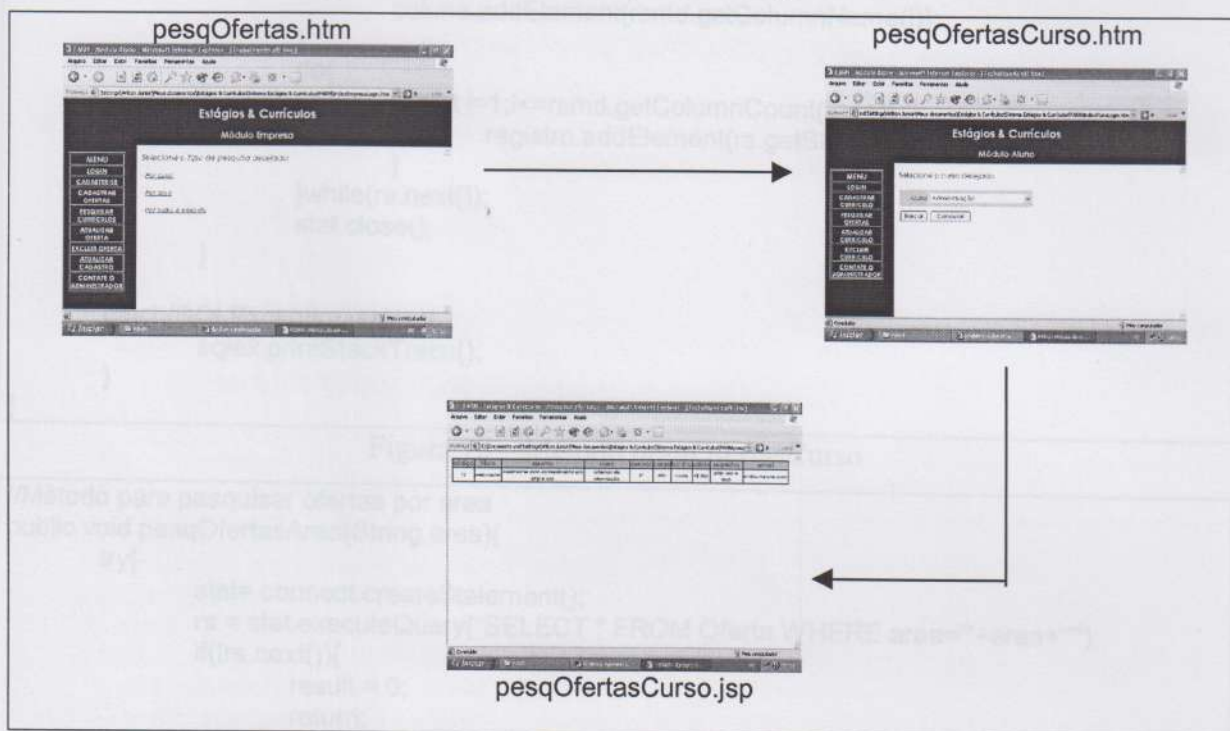


Figura 77 – Sequência de operações da opção pesquisar ofertas

Para cada opção é acionado um servlet diferente que acessa um método de pesquisa diferente que possuem o mesmo mecanismo de funcionamento. Inicialmente o aluno entra com os dados da busca, o sistema aciona o servlet correspondente que executa o método contido no servlet. Os métodos de pesquisa contidos na classe `Oferta` são:

- `pesqOfertasCurso`: busca as ofertas pelo curso informado;
- `pesqOfertasArea`: busca as ofertas pela área desejada;
- `pesqOfertasCursoPeriodo`: busca as ofertas pela oferta e pelo período informados.

Os métodos acima armazenam o resultado da busca em dois objetos do tipo `Vector`. O objeto `coluna` armazena o título das colunas, já o registro armazena os dados referentes a busca.

```

//Método para pesquisar ofertas por curso
public void pesqOfertasCurso(String curso){
    try{
        stat = connect.createStatement();
        rs = stat.executeQuery("SELECT * FROM Oferta WHERE curso='"+curso+"'");
        if(!rs.next()){
            result = 0;
            return;
        }else{
            result = 1;
            ResultSetMetaData rsmd = rs.getMetaData();
            for(int i=1;i<=rsmd.getColumnCount();i++){
                coluna.addElement(rsmd.getColumnName(i));
            }
            do{
                for(int i=1;i<=rsmd.getColumnCount();i++){
                    registro.addElement(rs.getString(i));
                }
            }while(rs.next());
            stat.close();
        }
    }
    catch(SQLException sqlex){
        sqlex.printStackTrace();
    }
}

```

Figura 78 – Método pesqOfertasCurso

```

//Método para pesquisar ofertas por área
public void pesqOfertasArea(String area){
    try{
        stat= connect.createStatement();
        rs = stat.executeQuery("SELECT * FROM Oferta WHERE area='"+area+"'");
        if(!rs.next()){
            result = 0;
            return;
        }else{
            result = 1;
            ResultSetMetaData rsmd = rs.getMetaData();
            for(int i=1;i<=rsmd.getColumnCount();i++){
                coluna.addElement(rsmd.getColumnName(i));
            }
            do{
                for(int i=1;i<=rsmd.getColumnCount();i++){
                    registro.addElement(rs.getString(i));
                }
            }while(rs.next());
            stat.close();
        }
    }
    catch(SQLException sqlex){
        sqlex.printStackTrace();
    }
}

```

Figura 79 – Método pesqOfertasArea

```
//Método para pesquisar ofertas por curso e período
public void pesqOfertasCursoPeriodo(String curso,String periodo){
    try{
        stat = connect.createStatement();
        rs = stat.executeQuery("SELECT * FROM Oferta WHERE curso='"+curso+"',
            periodo='"+periodo+"'");
        if(!rs.next()){
            result = 0;
            return;
        }else{
            result = 1;
            ResultSetMetaData rsmd = rs.getMetaData();
            for(int i=1;i<=rsmd.getColumnCount();i++){
                coluna.addElement(rsmd.getColumnName(i));
            }
            do{
                for(int i=1;i<=rsmd.getColumnCount();i++){
                    registro.addElement(rs.getString(i));
                }
            }while(rs.next());
            stat.close();
        }
    }
    catch(SQLException sqllex){
        sqllex.printStackTrace();
    }
}
```

Figura 80 – Método pesqOfertasCursoPeriodo

```
<% Sistema sist = new Sistema();
String curso = request.getParameter("curso");
sist.oferta.pesqOfertasCurso(curso);
int soma = sist.oferta.getColuna().size();
if (sist.oferta.getResult() != 0){ %>
    <table border="1" bordercolor="#000080">
    <th><% = sist.oferta.getColuna().elementAt(i)
    } %>
    <tr>
    <for int j=0;j<sist.oferta.getRegistro().size();j++){%>
    <td><% = sist.oferta.getRegistro().elementAt(j) %>
    <% if (soma == i+1){
        soma +=11; %>
    <tr><%
    }
    } %>
```

Figura 81 – Servlet pesqOfertasCurso

#### 4.2.1.3. Módulo empresa

Nesse módulo, o único ator é a empresa. Nele a empresa poderá efetuar e gerenciar seu cadastro e cadastrar e gerenciar ofertas de estágios a serem disponibilizadas aos alunos, além de buscar profissionais através dos currículos disponibilizados pelos alunos.

As principais operações deste módulo são: login, cadastrar empresa, cadastrar oferta, pesquisar currículos, atualizar cadastro, atualizar oferta e excluir oferta.



Figura 82 – Tela principal do módulo empresa

#### 4.2.1.3.1. Login

Assim como em todas as operações de login presentes no sistema, o ator seleciona a opção “login” no menu do módulo correspondente. O sistema exibe a tela de login, o ator, no caso a empresa, entra com o login e a senha. O sistema aciona o servlet responsável pela autenticação, loginEmpresa.jsp (figura 83), que executa o método correspondente, autenticarEmpresa (figura 84), que compara os dados informados com os recuperados no banco de dados e retorna true ou false dependendo do resultado da comparação. Se o resultado for true o sistema abre a página com todos os links referentes às operações da empresa ativados.

```
<% Sistema sist = new Sistema();
String login = request.getParameter("login");
String senha = request.getParameter("senha");
if(login != null && !login.equals("") && senha != null && !senha.equals("")){
    sist.autenticar.autenticarEmpresa(login,senha);
    if(sist.autenticar.result==true){%>
        <jsp:forward page="indexEmpresaLogin.htm"/>
    }else{%>
        Login/senha incorreto(s)!
    }
}%>
<% }else{ %>
    Parâmetros obrigatórios não informados!
}%>
```

Figura 83 – Servlet loginEmpresa.jsp

```
//Método para autenticar empresa
public void autenticarEmpresa(String login,String senha){
    try{
        this.login = login;
        this.senha = senha;
        stat = connect.createStatement();
        rs = stat.executeQuery("SELECT login,senha FROM Empresa WHERE login="
            +login+""");
        if ( rs.next() ) {
            loginaux = rs.getString(1).trim();
            senhaux = rs.getString(2).trim();
            result = autenticacao();
        } else {
            result = false;
        }
        stat.close();
    }
    catch(SQLException sqlex){
        sqlex.printStackTrace();
    }
}
```

Figura 84 – Método autenticarEmpresa

#### 4.2.1.3.2. Cadastrar empresa

Essa operação é responsável pelo cadastro da empresa no sistema. É através do login e da senha informados no cadastro que a empresa tem acesso às outras opções do sistema. Inicialmente a empresa seleciona a opção “cadastrar empresa”, o sistema abre a página cadastEmpresa.htm, onde serão informados todos os dados da empresa, inclusive o nome e o email do representante da empresa no sistema, que será o responsável pela empresa e por todas as informações que a empresa vier a disponibilizar no sistema.

Após a entrada dos dados, o sistema aciona o servlet cadastrarEmpresa.jsp (figura 86) que contém o método cadastrarEmpresa (figura 87). O servlet verifica se todos os dados foram digitados para poder acionar o método. Esse método é responsável pela inserção da oferta no banco de dados.

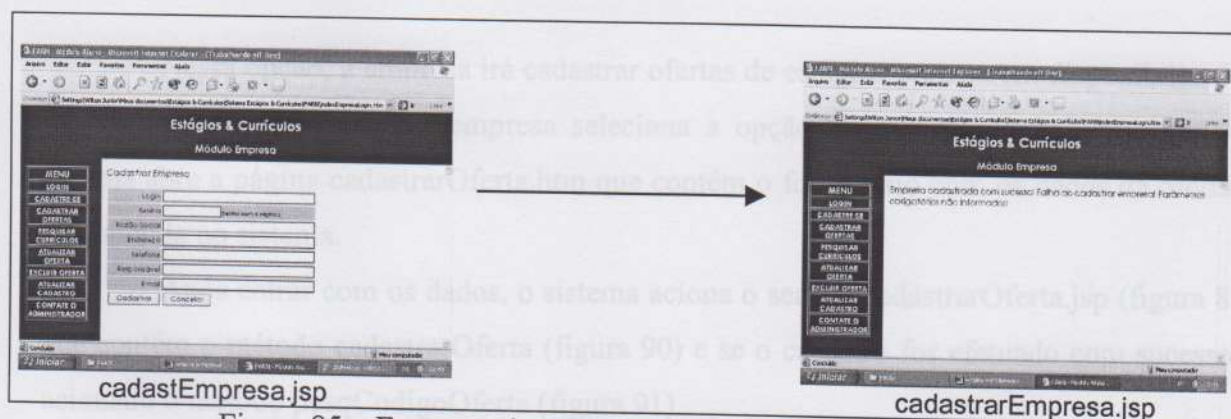


Figura 85 – Esquema de operações da opção cadastrar empresa

```

<% Sistema sist = new Sistema();
String login = request.getParameter("login");
String senha = request.getParameter("senha");
String razao = request.getParameter("razao");
String end = request.getParameter("endereco");
String fone = request.getParameter("telefone");
String resp = request.getParameter("responsavel");
String email = request.getParameter("email");
if ( login != null && !login.equals("") &&
    senha != null && !senha.equals("") &&
    razao != null && !razao.equals("") &&
    end != null && !end.equals("") &&
    fone != null && !fone.equals("") &&
    resp != null && !resp.equals("") &&
    email != null && !email.equals("") ) {
    sist.empresa.cadastrarEmpresa(login,senha,razao,end,fone,resp,email);
    if(sist.empresa.getResult()==1){%>
        <font color="#000080" face="Century Gothic">Empresa cadastrada com sucesso!
    <%}else{%>
        <font color="#000080" face="Century Gothic">Falha ao cadastrar empresa!
    <%}%>
    <% } else { %>
        <font color="#000080" face="Century Gothic">Parâmetros obrigatórios não preenchidos!
    <% } %>
}

```

Figura 86 – Servlet cadastrarEmpresa.jsp

```

//Método para cadastrar uma empresa no banco de dados
public void cadastrarEmpresa(String login,String senha,String razao,String endereco,String
telefone,String responsavel,String email){
    try{
        stat = connect.createStatement();
        result = stat.executeUpdate("INSERT INTO Empresa(login,senha,razao,
            endereco,telefone,responsavel,email) VALUES ('"+login+"','"+senha+"',
            '"+razao+"','"+endereco+"','"+telefone+"','"+responsavel+"','"+email+"')");
        stat.close();
    }
    catch(SQLException sqllex){
        sqllex.printStackTrace();
    }
}

```

Figura 87 – Método cadastrarEmpresa

#### 4.2.1.3.3. Cadastrar oferta

Nessa opção, a empresa irá cadastrar ofertas de estágios para serem disponibilizadas no sistema. Primeiramente, a empresa seleciona a opção “cadastrar ofertas” no menu. O sistema abre a página cadastrarOferta.htm que contém o formulário com os dados da oferta a ser incluída no sistema.

Após entrar com os dados, o sistema aciona o servlet cadastrarOferta.jsp (figura 89) que contém o método cadastrarOferta (figura 90) e se o cadastro for efetuado com sucesso é acionado o método pesqCodigoOferta (figura 91).

O servlet inicialmente testa se todos os parâmetros foram preenchidos corretamente, aciona o método `cadastrarOferta` que inclui a oferta no banco de dados e guarda o resultado inteiro referente a inserção da oferta na variável `result`. Este recupera o resultado através do método `getResult()`, e verifica se o resultado for igual a 1, executa o método `pesqCodigoOferta` que retorna o código da oferta no banco de dados, já que este campo é autonumerado pelo banco.

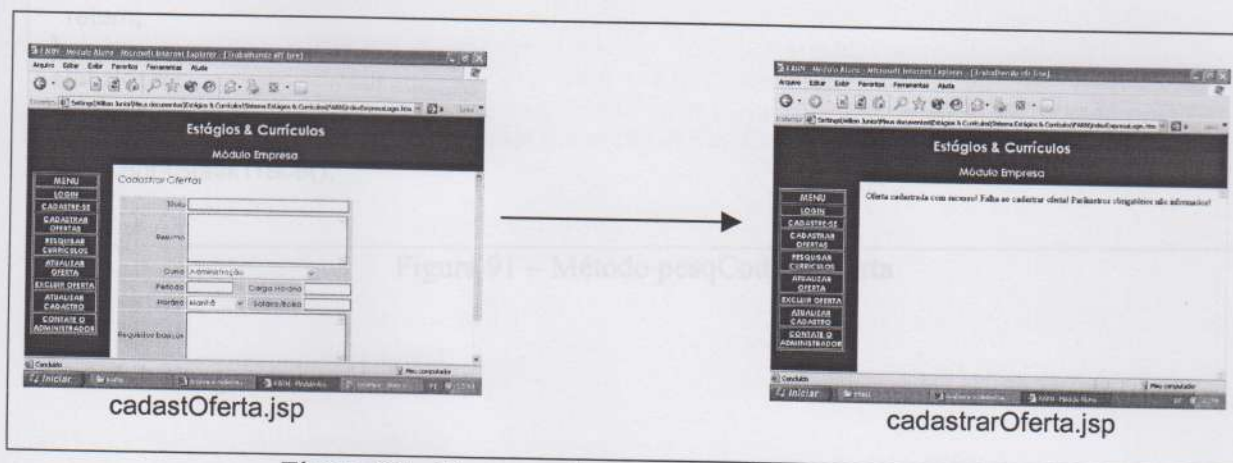


Figura 88 – Esquema de operações da opção cadastrar oferta

```
<% Sistema sist = new Sistema();
String titulo = request.getParameter("titulo"), resumo = request.getParameter("resumo");
String curso = request.getParameter("curso"), periodo = request.getParameter("periodo");
String carga = request.getParameter("carga"), horario = request.getParameter("horario");
String salario = request.getParameter("salario"), requisitos = request.getParameter("requisitos");
String email = request.getParameter("email");
if(titulo != null && !titulo.equals("") && resumo != null && !resumo.equals("") &&
curso != null && !curso.equals("") && periodo != null && !periodo.equals("") &&
carga != null && !carga.equals("") && horario != null && !horario.equals("") &&
salario != null && !salario.equals("") && requisitos != null && !requisitos.equals("") &&
email != null && !email.equals("")){
sist.oferta.cadastrarOferta(titulo,resumo,curso,periodo,carga,horario,salario,requisitos,email);
if(sist.oferta.getResult()!=0){ %>
Oferta cadastrada com sucesso!
<%}else{%> Falha ao cadastrar oferta! <%}
} else {%> Parâmetros obrigatórios não informados! <% } %>
```

Figura 89 – Servlet cadastrarOferta.jsp

```
//Método para cadastrar uma oferta
public void cadastrarOferta(String titulo,String resumo,String curso,String periodo,String carga,
String horario,String salario,String requisitos,String email){
try{
stat = connect.createStatement();
result = stat.executeUpdate("INSERT INTO Oferta(titulo,resumo,curso,periodo,carga ,
horario,salario,requisitos,email) VALUES ('"+titulo+"','"+resumo+"','"+curso+"',
 '"+periodo+"','"+carga+"','"+horario+"','"+salario+"','"+requisitos+"','"+email+"')");
stat.close();
}
catch(SQLException sqllex){
sqllex.printStackTrace();
}
}
```

Figura 90 – Método cadastrarOferta

```

//Método que busca o código da oferta pelo título
public void pesqCodigo(String titulo){
try{
    stat = connection.createStatement();
    rs = stat.executeQuery("SELECT codigo FROM Oferta WHERE titulo='"+titulo+"'");
    if(rs.next()){
        codigo = rs.getInt(1);
        result = 1;
    } else {
        result = 0;
        return;
    }
    stat.close();
}
catch(SQLException sqllex){
    sqllex.printStackTrace();
}
}

```

Figura 91 – Método pesqCodigoOferta

#### 4.2.1.3.4. Pesquisar currículos

Essa operação consiste em buscar currículos de alunos que estejam no banco de dados. A empresa seleciona a opção “pesquisar currículos” no menu principal, o sistema abre a página “pesquisarCurriculos.jsp” com três opções de busca: por curso, por curso e período e por área. A empresa seleciona a busca desejada, o sistema abre formulário para entrada de dados referentes a pesquisa, como por exemplo a página “pesqCurriculosCurso.jsp”, a empresa entra com os dados referentes ao currículo desejado e o sistema através do servlet pesqCurrículoCurso.jsp (figura 94) busca os dados dos currículos desejados e retorna-os em forma de tabela.

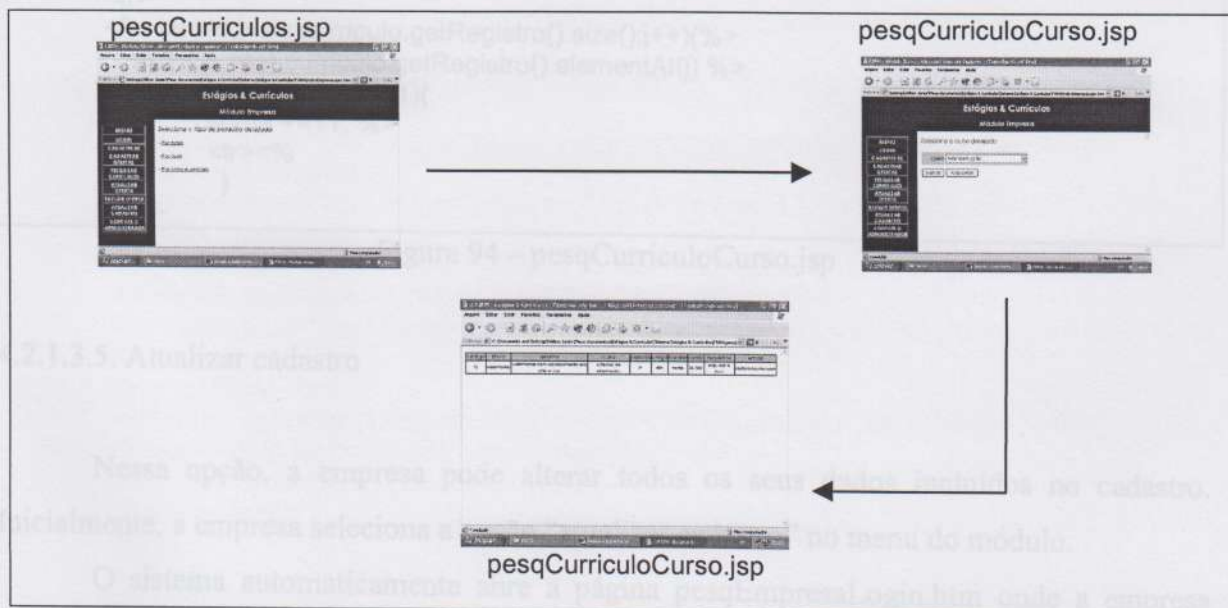


Figura 92 – Esquema de operações da opção pesquisar currículos

```
//Método para pesquisar currículos por curso
public void pesqCurrículoCurso(String curso){
    try{
        stat = connect.createStatement();
        rs = stat.executeQuery("SELECT * FROM Currículo,Eventos,Experiencia,Cursos,
                               Formacao,Idiomas WHERE (SELECT login FROM Cursos WHERE
                               Cursos.curso='"+curso+"'");
        if(!rs.next()){
            result = 0;
            return;
        }else{
            result = 1;
            ResultSetMetaData rsmd = rs.getMetaData();
            for(int i=1;i<=rsmd.getColumnCount();i++){
                coluna.addElement(rsmd.getColumnName(i));
            }
            do{
                for(int i=1;i<=rsmd.getColumnCount();i++){
                    registro.addElement(rs.getString(i));
                }
            }while(rs.next());
            stat.close();
        }
    }
    catch(SQLException sqlex){
        sqlex.printStackTrace();
    }
}
```

Figura 93 – Método pesqCurrículosCurso

```
<% Sistema sist = new Sistema();
String curso = request.getParameter("curso");
sist.curriculo.pesqOfertasCurso(curso);
int soma = sist.curriculo.getColuna().size();
if (sist.curriculo.getResult() != 0){ %>
    <table border="1" bordercolor="#000080">
        <% for(int i=0;i<soma;i++){ %>
            <th><% = sist.curriculo.getColuna().elementAt(i)
            %>
        </th>
        <tr>
            <for int j=0;j<sist.curriculo.getRegistro().size();j++){%>
                <td><% =sist.curriculo.getRegistro().elementAt(j) %>
                <% if (soma == i+1){
                    soma +=11; %>
                </td><%
                %>
            }
        </tr>
    } %>
    %>
```

Figura 94 – pesqCurrículoCurso.jsp

#### 4.2.1.3.5. Atualizar cadastro

Nessa opção, a empresa pode alterar todos os seus dados incluídos no cadastro. Inicialmente, a empresa seleciona a opção “atualizar cadastro” no menu do módulo.

O sistema automaticamente abre a página pesqEmpresaLogin.htm onde a empresa entrará com o seu login e sua senha e logo após é acionado o servlet buscarEmpresaLogin.jsp

(figura 96) responsável por buscar as informações referentes a empresa usando o método `pesqEmpresaLoginSenha` (figura 97). O resultado da busca é retornado em um formulário onde os valores dos dados são recuperados através de métodos `get`, das variáveis onde foram armazenados os resultados da pesquisa.

Após atualizados, os dados no formulário, o sistema aciona o servlet `atualizarEmpresa.jsp` (figura 98) que atualiza os dados no banco de dados através do método `atualizarEmpresa` (figura 99) da classe `Empresa`. Esse servlet retorna a mensagem referente ao resultado da atualização do cadastro da empresa.



Figura 95 – Esquema de operações da opção atualizar cadastro

```
<% Sistema sist = new Sistema();
String login = request.getParameter("login");
String senha = request.getParameter("senha");
if(login != null && !login.equals("") && senha != null && !senha.equals("senha")){
sist.empresa.pesqEmpresaLoginSenha(login,senha);
if (sist.empresa.getResult()==0){ %>
<H1>Empresa não encontrada!
<% }else{ %>
<form method="post" action="atualizarEmpresa.jsp">
<table>
<tr><td>Login</td>
<td><input type="text" name="login" value="<%=sist.empresa.getLogin()%>"></td>
<tr><td>Senha</td>
<td><input type="password" name="senha" value="<%=sist.empresa.getSenha()%>"></td>
<tr><td>Razão Social</td>
<td><input type="text" name="razao" value="<%=sist.empresa.getRazao()%>"></td>
<tr><td>Endereço</td>
<td><input type="text" name="endereco" value="<%=sist.empresa.getEndereco()%>"></td>
<tr><td>Telefone</td>
<td><input type="text" name="telefone" value="<%=sist.empresa.getTelefone()%>"></td>
<tr><td>Responsável</td>
<td><input type="text" name="responsavel" value="<%=sist.empresa.getResponsavel()%>"></td>
<tr><td>Email</td>
<td><input type="text" name="email" value="<%=sist.empresa.getEmail()%>"></td>
<tr><td><input type="submit" value="Atualizar"></td>
<td><input type="reset" value="Cancelar"></td>
</tr>
</table></form>
<% }
} else {> Login e/ou senha não informado(s)!
<% } %>
```

Figura 96 – Servlet `buscarEmpresaLogin.jsp`

```

//Método que busca uma empresa por login e senha
public void pesqEmpresaLoginSenha(String login,String senha){
try{
    stat = connect.createStatement();
    rs = stat.executeQuery("SELECT * FROM Empresa WHERE login='"+login+"',senha=
        '"+senha+"'");
    if(rs.next()){
        result=1;
        login = rs.getString("login");
        senha = rs.getString("senha");
        razao = rs.getString("razao");
        endereco = rs.getString("endereco");
        telefone = rs.getString("telefone");
        responsavel = rs.getString("responsavel");
        email = rs.getString("email");
    }else{
        result=0;
        return;
    }
    stat.close();
}
catch(SQLException sqllex){
    sqllex.printStackTrace();
}
}

```

Figura 97– Método pesqEmpresaLoginSenha

```

<% Sistema sist = new Sistema();
String login = request.getParameter("login");
String senha = request.getParameter("senha");
String razao = request.getParameter("razao");
String endereco = request.getParameter("endereco");
String telefone = request.getParameter("telefone");
String responsavel = request.getParameter("responsavel");
String email = request.getParameter("email");
if (login != null && !login.equals("") && senha != null && !senha.equals("") &&
    razao != null && !razao.equals("") && endereco != null && !endereco.equals("") &&
    telefone != null && !telefone.equals("") && responsavel != null &&
    !responsavel.equals("responsavel")
    email != null && !email.equals("")){
    sist.empresa.atualizarEmpresa(login,senha,razao,endereco,telefone,responsavel,email);
    if (sist.empresa.getResult()==0){ %>
        <H1>Falha na atualização da empresa!
    <% }else{ %>
        <H1>Empresa atualizada com sucesso!
    <% }
} else {%>
    Parâmetros obrigatórios não informados!
<% } %>

```

Figura 98 – Servlet atualizarEmpresa.jsp

```

//Método que atualiza uma empresa no banco de dados pela razão social
public void atualizarEmpresa(String login,String senha,String razao,String endereco,String
telefone,String responsavel,String email){
    try{
        stat = connect.createStatement();
        result = stat.executeUpdate("UPDATE Empresa SET login='"+login+"', senha=
        '"+senha+"',razao='"+razao+"', endereco='"+endereco+"',telefone='"+telefone+"',
        responsavel='"+responsavel+"',email='"+email+"'WHERE razao='"+razao+"'");
        stat.close();
    }
    catch(SQLException sqllex){
        sqllex.printStackTrace();
    }
}

```

Figura 99 – Método atualizarEmpresa

#### 4.2.1.3.6. Atualizar oferta

Nessa opção, a empresa poderá atualizar os dados de uma oferta cadastrada segundo seu código.

A empresa seleciona a opção “atualizar oferta” no menu. O sistema abre a tela `pesqOfertaCodigoAtual.jsp`, onde a empresa entra com o código da oferta a ser atualizada. O sistema executa o servlet `pesqOfertaCodigoAtualizar.jsp` (figura 101) que busca os dados referentes a oferta que possui o código informado através do método `pesqOfertaCodigo` (figura 102) da classe `Oferta` que busca os dados de uma oferta pelo código e os armazena em variáveis locais da classe. O servlet mostra os dados da oferta em um formulário onde os valores são acessados através de métodos `get` das variáveis onde estão armazenados os resultados da pesquisa.

A empresa atualiza os dados no formulário, o sistema executa o servlet `atualizarOferta.jsp` (figura 103) que aciona o método `atualizarOferta` (figura 104), responsável por atualizar a oferta no banco de dados, retornando 0 ou 1 para variável inteira `result`, dependendo do resultado da atualização. O servlet recupera o valor da variável `result` e imprime a mensagem referente ao resultado da atualização.

```

<input type="text" name="senha" value="<%=oferta.getSenha()%>" />
<input type="text" name="razao" value="<%=oferta.getRazao()%>" />
<input type="text" name="endereco" value="<%=oferta.getEndereco()%>" />
<input type="text" name="telefone" value="<%=oferta.getTelefone()%>" />
<input type="text" name="responsavel" value="<%=oferta.getResponsavel()%>" />
<input type="text" name="email" value="<%=oferta.getEmail()%>" />
<input type="button" value="Atualizar" />
<input type="button" value="Cancelar" />
</form>
</div>

```

Figura 101 – Servlet `pesqOfertaCodigoAtualizar.jsp`

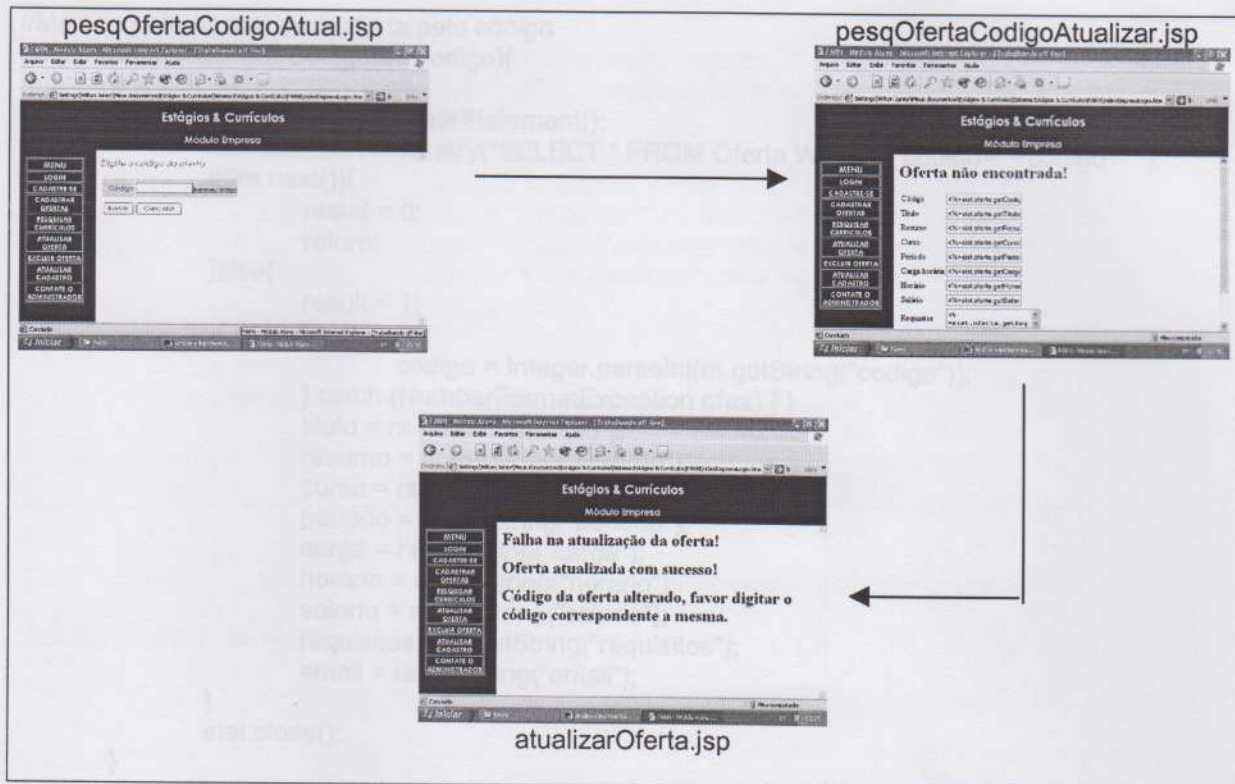


Figura 100 – Sequência de operações da opção atualizar oferta

```
<% Sistema sist = new Sistema();
int codigo = Integer.parseInt(request.getParameter("codigo"));
sist.oferta.pesqOfertaCodigo(codigo);
if (sist.oferta.getResult()==0){ %>
<H1>Oferta não encontrada!
<% }else{ %>
<form method="post" action="atualizarOferta.jsp">
<table>
<tr> <td>Código</td>
<td><input type="text" name="codigo" value="<%=sist.oferta.getCodigo()%>"></td></tr>
<tr> <td>Título</td>
<td><input type="text" name="titulo" value="<%=sist.oferta.getTitulo()%>"></td> </tr>
<tr> <td>Resumo</td>
<td><input type="text" name="resumo" value="<%=sist.oferta.getResumo()%>"></td></tr>
<tr> <td>Curso</td>
<td><input type="text" name="curso" value="<%=sist.oferta.getCurso()%>"></td></tr>
<tr> <td>Período</td>
<td><input type="text" name="periodo" value="<%=sist.oferta.getPeriodo()%>"></td></tr>
<tr> <td>Carga horária</td>
<td><input type="text" name="carga" value="<%=sist.oferta.getCarga()%>"></td></tr>
<tr> <td>Horário</td>
<td><input type="text" name="horario" value="<%=sist.oferta.getHorario()%>"></td></tr>
<tr> <td>Salário</td>
<td><input type="text" name="salario" value="<%=sist.oferta.getSalario()%>"></td> </tr>
<tr> <td>Requisitos</td><td><textarea name="requisitos"><%=sist.oferta.getRequisitos()%>
</textarea></td></tr>
<tr> <td>Email</td>
<td><input type="text" name="email" value="<%=sist.oferta.getEmail()%>"></td></tr>
<tr> <td><input type="submit" value="Atualizar"></td>
<td><input type="reset" value="Cancelar"></td>
</tr></table></form>
<% } %>
```

Figura 101 – Servlet pesqOfertaCodigoAtualizar.jsp

```

//Método para buscar uma oferta pelo código
public void pesqOfertaCodigo(int codigo){
    try{
        stat = connect.createStatement();
        rs = stat.executeQuery("SELECT * FROM Oferta WHERE codigo='"+codigo+"'");
        if(!rs.next()){
            result = 0;
            return;
        }else{
            result = 1;
            try {
                codigo = Integer.parseInt(rs.getString("codigo"));
            } catch (NumberFormatException nfex) { }
            titulo = rs.getString("titulo");
            resumo = rs.getString("resumo");
            curso = rs.getString("curso");
            periodo = rs.getString("periodo");
            carga = rs.getString("carga");
            horario = rs.getString("horario");
            salario = rs.getString("salario");
            requisitos = rs.getString("requisitos");
            email = rs.getString("email");
        }
        stat.close();
    }
    catch(SQLException sqlex){
        sqlex.printStackTrace();
    }
}

```

Figura 102 – Método pesqOfertaCodigo

```

<% Sistema sist = new Sistema();
int codigo = Integer.parseInt(request.getParameter("codigo"));
String titulo = request.getParameter("titulo");
String resumo = request.getParameter("resumo");
String curso = request.getParameter("curso");
String periodo = request.getParameter("periodo");
String carga = request.getParameter("carga");
String horario = request.getParameter("horario");
String salario = request.getParameter("salario");
String requisitos = request.getParameter("requisitos");
String email = request.getParameter("email");
if(codigo != 0){
    sist.oferta.atualizarOferta(codigo,titulo,resumo,curso,periodo,carga,horario,salario,requisitos,
    email);
    if (sist.oferta.getResult()==0){ %>
        <H1>Falha na atualização da oferta!
    }else{ %>
        <H1>Oferta atualizada com sucesso!
    }
} else { %>
    <H1>Código da oferta alterado, favor digitar o código correspondente a mesma.
} %>

```

Figura 103 – Servlet atualizarOferta.jsp

```
//Método para atualizar uma oferta
public void atualizarOferta(int codigo,String titulo,String resumo,String curso,String periodo,
String carga,String horario,String salario,String requisitos,String email){
    try{
        stat = connect.createStatement();
        result = stat.executeUpdate("UPDATE Oferta SET(titulo='"+titulo+"',
        resumo='"+resumo+"',curso='"+curso+"',periodo='"+periodo+"',
        carga='"+carga+"',horario='"+horario+"',salario='"+salario+"',
        requisitos='"+requisitos+"',email='"+email+"') WHERE codigo=
        '"+codigo+"'");
        stat.close();
    }
    catch(SQLException sqlex){
        sqlex.printStackTrace();
    }
}
```

Figura 104 – Método atualizarOferta

#### 4.2.1.3.7. Excluir oferta

Assim como na atualização, a opção “excluir oferta” inicia-se com a pesquisa da oferta pelo código. Ao selecionar esta opção o sistema executa a página `pesqOfertaCodigoExcluir` onde a empresa entra com o código correspondente. Logo após o sistema executa o servlet `pesqOfertaCodigoExcluir.jsp` (figura 106) que invoca o método `pesqOfertaCodigo` da classe `Oferta`, conforme mostrado no item anterior. O sistema retorna os dados da oferta para que a empresa confirme a exclusão. Confirmada a exclusão, o sistema executa o servlet `excluirOferta.jsp` (figura 107) que invoca o método `excluirOferta` (figura 108) da classe `Oferta`.

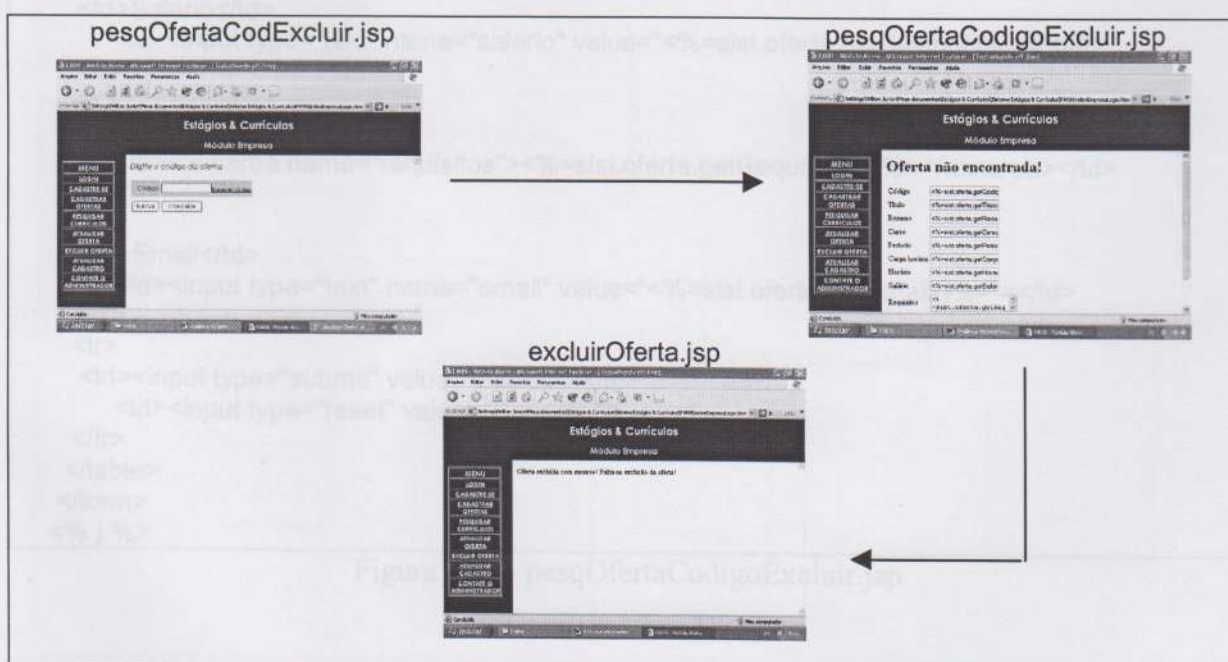


Figura 105 – Sequência de operações da opção excluir oferta

```

<% Sistema sist = new Sistema();
int codigo = Integer.parseInt(request.getParameter("codigo"));
sist.oferta.pesqOfertaCodigo(codigo);
if (sist.oferta.getResult()==0){ %>
  <H1>Oferta não encontrada!
<% }else{ %>
  <form method="post" action="excluirOferta.jsp">
  <table>
  <tr>
  <td>Código</td>
  <td><input type="text" name="codigo" value="<%=sist.oferta.getCodigo()%>"></td>
  </tr>
  <tr>
  <td>Título</td>
  <td><input type="text" name="titulo" value="<%=sist.oferta.getTitulo()%>"></td>
  </tr>
  <tr>
  <td>Resumo</td>
  <td><input type="text" name="resumo" value="<%=sist.oferta.getResumo()%>"></td>
  </tr>
  <tr>
  <td>Curso</td>
  <td><input type="text" name="curso" value="<%=sist.oferta.getCurso()%>"></td>
  </tr>
  <tr>
  <td>Período</td>
  <td><input type="text" name="periodo" value="<%=sist.oferta.getPeriodo()%>"></td>
  </tr>
  <tr>
  <td>Carga horária</td>
  <td><input type="text" name="carga" value="<%=sist.oferta.getCarga()%>"></td>
  </tr>
  <tr>
  <td>Horário</td>
  <td><input type="text" name="horario" value="<%=sist.oferta.getHorario()%>"></td>
  </tr>
  <tr>
  <td>Salário</td>
  <td><input type="text" name="salario" value="<%=sist.oferta.getSalario()%>"></td>
  </tr>
  <tr>
  <td>Requisitos</td>
  <td><textarea name="requisitos"><%=sist.oferta.getRequisitos()%></textarea></td>
  </tr>
  <tr>
  <td>Email</td>
  <td><input type="text" name="email" value="<%=sist.oferta.getEmail()%>"></td>
  </tr>
  <tr>
  <td><input type="submit" value="Excluir"></td>
  <td><input type="reset" value="Cancelar"></td>
  </tr>
  </table>
  </form>
  <% } %>

```

Figura 106 – pesqOfertaCodigoExcluir.jsp

```

<% Sistema sist = new Sistema()
    int codigo = Integer.parseInt(request.getParameter("codigo"));
    sist.oferta.excluirOferta(codigo);
    if(sist.aluno.getResult()==1){%>
        Oferta excluída com sucesso!
    <%}else{%>
        Falha na exclusão da oferta!
    <%}%>

```

Figura 107 – excluirOferta.jsp

```

//Método para excluir uma oferta
public void excluirOferta(int codigo){
    try{
        stat = connect.createStatement();
        result = stat.executeUpdate("DELETE FROM Oferta WHERE codigo='"+codigo+"'");
        stat.close();
    }
    catch(SQLException sqllex){
        sqllex.printStackTrace();
    }
}

```

Figura 108 – Método excluirOferta

#### 4.2.2. Modelo de Dados

Nesse tópico, é mostrado o modelo lógico do banco de dados do sistema (figura 109). O banco de dados do sistema foi implementado utilizando a ferramenta Microsoft Access 2000.

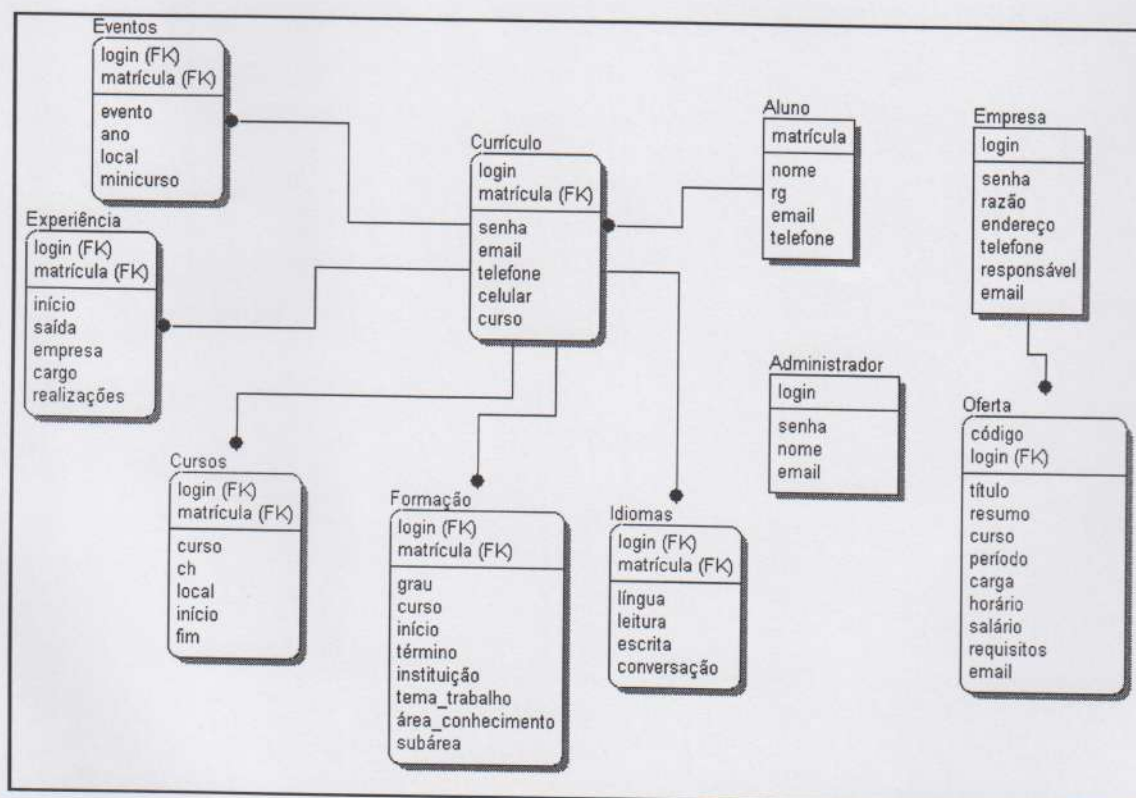


Figura 109 – Modelo lógico do banco de dados

#### 4.2.3.WAR (Web Application)

Neste item é mostrado o WAR referente ao sistema de ofertas de estágios e currículos. O diretório sistemaEstagiosCurriculos-WAR é onde estão armazenados todos os scripts, classes e interfaces do sistema. Neste diretório estão armazenados as páginas JSP e HTML, além do mesmo conter o subdiretório, além do subdiretório WEB-INF onde estão armazenados as classes, no subdiretório WEB-INF/classes e o descritor da aplicação web.xml.

O esquema abaixo (figura 110) mostra a estrutura do WAR do sistema.

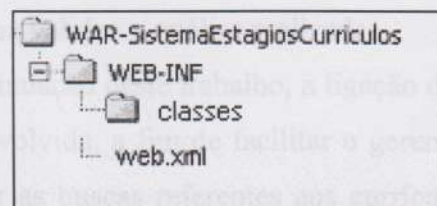


Figura 110 – Estrutura do WAR do sistema de estágios e currículos

## 5. CONCLUSÃO BIBLIOGRÁFICAS

O sistema de ofertas de estágios e currículos funcionará como uma ponte direta entre os alunos e as empresas que desejem contratar estes. No sistema, o aluno poderá cadastrar seu currículo, gerenciá-lo e buscar ofertas de estágios cadastrados pela empresa. Para a empresa as opções são: cadastrar ofertas de estágios, cadastrar-se, pesquisar currículos de alunos e gerenciar as ofertas cadastradas e seu cadastro.

Toda a funcionalidade proposta para o sistema na análise foi implementada. O sistema foi testado na prática e verificou-se que o mesmo atendeu aos requisitos levantados na análise. A implementação, desta forma validou a análise realizada.

Propõe-se como continuação deste trabalho, a ligação do sistema com a base de dados acadêmica, da instituição envolvida, a fim de facilitar o gerenciamento dos alunos que farão uso desse sistema e detalhar as buscas referentes aos currículos e as ofertas deixando-as o mais sucintas possíveis.

Este trabalho estará disponível para toda a comunidade, através do site <http://www.estagiosecurriculos.com>.

FIELDS, D. K.; KOLD, M. A. *Desenvolvendo na web com Java Server Pages*. Rio de Janeiro: Ciência Moderna Ltda., 2000.

HORSTMANN, C. S.; CORNELL, G. *Core Java 2*. São Paulo: Makron Books, 2001.

HYPERLINE. Ferramentas de desenvolvimento em Java. Disponível em: <http://www.mhvaiz.com.br/links/prop/java/java-tool.html>. Acesso em: 03 out. 2003.

JANDI, JUNIOR, P. *Introdução ao Java*. São Francisco: Universidade de São Francisco, 1999.

LEMAY, L.; CADENHEAD, R. *Aprenda em 21 dias Java 2*. Rio de Janeiro: Campus, 1999.

LOCUS. Orientação a objetos. Disponível em: <http://locus.com.br/corporativo/tecnologia/objetos.htm>. Acesso em: 16 abr. 2003.

MARCONI, M. de A.; LAKATOS, E. M. *Técnicas de pesquisa: planejamento e execução de pesquisas, amostragens e técnicas de pesquisa, elaboração, análise e interpretação de dados*, 4. ed. São Paulo: Atlas, 1999.

MELO, R. B. de A. *Biblioteca digital: implementação de um protótipo utilizando JSP*, 2002. Monografia (Bacharelado em Sistemas de Informação) - Faculdade Natalense para o Desenvolvimento do Rio Grande do Norte.

MINK, C.; ANIBAL JUNIOR. *Acesso 2000*. São Paulo: Makron Books, 2000.

## REFERÊNCIAS BIBLIOGRÁFICAS

- ALVAREZ, M. A. **Qué es HTML**. Disponível em: <<http://www.htmlstaff.org/html09.php>>. Acesso em: 08 jul. 2003.
- BECEIRO, F. P. **Tutorial HTML**. Disponível em: <<http://www.abcdicas.com.br>>. Acesso em: 08 jul. 2003.
- BODOFF, S.; GREEN, D. **J2EE: tutorial**. Rio de Janeiro: Campus, 2002.
- BOOCH, G.; RUMBAUGH, J.; JACOBSON, I. **UML**, guia do usuário. Rio de Janeiro: Campus, 2000.
- CABRAL, A. M. de O.; ARAÚJO, L. G. G. de. **UML – Unified Modeling Language. Cursos de Informática**, São Paulo, ano 1, n. 3, 2003. 1 CD-ROOM.
- CENTRO DE INFORMÁTICA E EDUCAÇÃO. **FrontPage 2000**. Disponível em: <<http://www.cic.ua.pt/gc/www/frontpage>>. Acesso em: 10 out. 2003.
- COLEMAN, D. *et al.* **Desenvolvimento orientado a objetos: o método fusion**. Rio de Janeiro: Campus, 1996.
- DISCOVER TECHNOLOGY. **Java Database Connectivity**. Disponível em: <<http://www.discover.com.br/artigos/jdbc.htm>>. Acesso em: 08 abr. 2003.
- FIELDS, D. K.; KOLD, M. A. **Desenvolvendo na web com Java Server Pages**. Rio de Janeiro: Ciência Moderna Ltda., 2000.
- HORSTMANN, C. S.; CORNELL, G. **Core Java 2**. São Paulo: Makron Books, 2001.
- HYPERLINK. **Ferramentas de desenvolvimento em Java**. Disponível em: <<http://www.mhavila.com.br/link/prog/java/java-tool.html>>. Acesso em: 05 out. 2003.
- JANDL JUNIOR, P. **Introdução ao Java**. São Francisco: Universidade de São Francisco, 1999.
- LEMAY, L.; CADENHEAD, R. **Aprenda em 21 dias Java 2**. Rio de Janeiro: Campus, 1999.
- LOCUS. **Orientação a objetos**. Disponível em: <<http://locus.com.br/corporativo/tecnologia/objetos.htm>>. Acesso em 16 abr. 2003.
- MARCONI, M. de A.; LAKATOS, E. M. **Técnicas de pesquisa: planejamento e execução de pesquisas, amostragens e técnicas de pesquisa, elaboração, análise e interpretação de dados**. 4. ed. São Paulo: Atlas, 1999.
- MELO, R. B. de A. **Biblioteca digital: implementação de um protótipo utilizando JSP**. 2002. Monografia (Bacharelado em Sistemas de Informação) – Faculdade Natalense para o Desenvolvimento do Rio Grande do Norte.
- MINK, C.; ANIBAL JUNIOR. **Access 2000**. São Paulo: Makron Books, 2000.

OLIVEIRA, S. L. de. **Tratado de metodologia científica: projetos de pesquisa, TGI, TCC, monografias, dissertações e teses.** São Paulo: Pioneira, 1998.

PAGE-JONES, M. **Fundamentos do desenho orientado a objetos com UML.** São Paulo: Pearson Education do Brasil, 2001.

PEDRON, S. L.; GREMELMEIER, C. H. **Utilização da linguagem de programação Java com banco de dados relacionais.** Disponível em: <<http://www.inf.uri.com.br/utilizacao.htm>>. Acesso em: 25 mar. 2003.

RICARTE, I. L. M. **Introdução a Orientação a Objetos.** Disponível em: <<http://www.dca.fee.unicamp.br/courses/PooJava/polimorf>>. Acesso em: 29 out. 2003.

ROCHA NETO, T. F. da. **Implementação de um sistema acadêmico na web utilizando JSP.** 2002. Monografia (Bacharelado em Sistemas de Informação) – Faculdade Natalense para o Desenvolvimento do Rio Grande do Norte.

SEVERO, C. **Curso básico de HTML.** Disponível em: <<http://www.terra.com.br/informatica/ajuda/estiloweb/webdesign/manual1.htm>>. Acesso em: 08 jul. 2003.

SILBERSCHATZ, A.; KORTH, H. F.; SUDARSHAN, S. **Sistema de banco de dados.** 3. ed. São Paulo: Makron Books, 1999.

SIQUEIRA, M. F. **Herança.** Disponível em: <<http://www.dct.ufms.br/~marcelo/teaching/too/texts/aula04.pdf>>. Acesso em: 29 out. 2003.

SOLIS, S. M. **Introdução ao JSP e WML.** Disponível em: <<http://br.wmlclub.com/articulos/jsp.htm>>. Acesso em: 25 mar. 2003.

TAFNER, M. A.; CORREIA, C. H. **Análise orientada a objetos.** Florianópolis: Visual Books, 2001.

TELEMACO NETO, U. **O que é JSP.** Disponível em: <<http://crieseuwebsite.com/tutjsp01.shtml>>. Acesso em: 05 mai. 2003.

MANGAN, M. A. S.; VARGAS, P. K.; AZZOLIN, D. **Técnicas para desenvolvimento de aplicações orientadas a objetos utilizando a linguagem Java.** Disponível em: <<http://www.portajava.com/home/modules.php?name=News&file=article&id=52>>. Acesso em: 25 mar. 2003.

SUTOS, A. V. de. **UML: prático e descomplicado.** 2. ed. São Paulo: Foca, 2002.

MAYER, A. C. **Otimizando a performance de bancos de dados relacionais.** Rio de Janeiro: Atlas Books do Brasil, 2001.

OLIVEIRA, A. de P. **Apostila Servlet/JSP.** Viçosa/MG: Universidade Federal de Viçosa, 2001.

## BIBLIOGRAFIA CONSULTADA

ANDRADE, S.; OLIVEIRA, A. K. P. de. **Manual para apresentação de trabalhos acadêmicos** projetos de pesquisa, relatórios finais, monografias e artigos. 1. ed. rev. amp. Natal: FARN, 2003.

COLLA, E. C. **Servlets**, Java ao lado do servidor. Disponível em: <<http://www.insite.com.br/docs/develop-servlet95.htm>>. Acesso em: 22 abr. 2003.

CURSO HTML. Disponível em: <<http://www.detetivez.hpg.ig.com.br/html/oquehtml.htm>>. Acesso em: 08 jul. 2003.

DEITEL, H. M.; DEITEL, D. J. **Java**, como programar. 3. ed. Porto Alegre: Bookman, 2001.

FIGUEIREDO, M. L. de. **Análise orientada a objetos de um sistema acadêmico na web usando UML**. 2002. Monografia (Bacharelado em Sistemas de Informação) – Faculdade Natalense para o Desenvolvimento do Rio Grande do Norte.

FOWLER, M.; SCOTT, K. **UML essencial**: um breve guia para a linguagem-padrão de modelagem de objetos. 2. ed. Porto Alegre: Bookman: 2000.

GEARY, D. M. **Dominando Java Server Pages avançado**. Rio de Janeiro: Ciência Moderna Ltda., 2002.

HAGGAR, P. **Java**: guia prático de programação. Rio de Janeiro: Campus, 2000.

INTERFACE ENTRE JAVA E BANCOS DE DADOS RELACIONAIS. Disponível em: <<http://www.dca.fee.unicamp.br/courses/poojava/javadb/jdbc.html>>. Acesso em: 24 mar. 2002.

INTRODUÇÃO A PROGRAMAÇÃO SQL. Disponível em: <<http://www.portaldaprogramacao.com/artigo2.asp?n=263>>. Acesso em: 22 mai. 2003.

INTRODUÇÃO AO MUNDO DO JAVA. Disponível em: <<http://www2.fundao.pro.br/articles.asp?cod=58>>. Acesso em: 22 abr. 2003.

MANGAN, M. A. S.; VARGAS, P. K.; AZZOLIN, D. **Técnicas para desenvolvimento de aplicações orientadas a objetos utilizando a linguagem Java**. Disponível em: <<http://www.portaljava.com/home/modules.php?name=News&file=article&sid=52>>. Acesso em: 25 mar. 2003.

MATOS, A. V. de. **UML**: prático e descomplicado. 2. ed. São Paulo: Érica, 2002.

MAYER, R. C. **Otimizando a performance de bancos de dados relacionais**. Rio de Janeiro: Axcel Books do Brasil, 2001.

OLIVEIRA, A. de P. **Apostila Servlet/JSP**. Viçosa/MG: Universidade Federal de Viçosa, 2001.

\_\_\_\_\_, D. V. de. **JDBC**. Videira/SC: UNOESC, 2001.

ROMÃO, B. **Introdução à linguagem JSP**. Disponível em: <http://www.portaldaprogramacao.com/artigos2.asp?n=190>>. Acesso em: 05 mai. 2003.

SOUZA, E. C. de. **Programação orientada a objetos com Java**. Florianópolis: Relativa, 2002.

TUTORIAL HTML. Disponível em: <http://www.icmc.usp.br/manuals/HTML/intro.html>>. Acesso em: 22 mai. 2003.

UNIVERSIDADE DE AVEIRO. **Introdução ao HTML**. Acesso em: <http://www.cic.ua.pt/gc/www/introhtml.htm>>.

## ANEXO A – QUESTIONÁRIO

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

**QUESTIONÁRIO PARA LEVANTAMENTO DE DADOS PARA CONSTRUÇÃO DA  
ANÁLISE DE UM SISTEMA DE ESTÁGIOS E CURRÍCULOS PARA A WEB**

1. Você acha importante o desenvolvimento de um sistema de ofertas de estágios e currículos para o site da FARN? Justifique sua resposta.

---

---

---

---

2. Quais seriam as vantagens que este sistema poderia trazer para os alunos da FARN?

---

---

---

3. Quais as desvantagens acarretadas pela implantação desse sistema?

---

---

---

4. Na sua opinião, quais os dados dos alunos que deveriam ser divulgados no currículo:

- ☐ FORMAÇÃO ACADEMICA  
☐ CONHECIMENTO EM INFORMATICA  
☐ LINGUAS  
☐ DADOS PESSOAIS  
☐ REFERENCIA PROFISSIONAL  
☐ PUBLICAÇÃO DE ARTIGOS  
☐ CONGRESSOS E EVENTOS  
☐ OUTROS: \_\_\_\_\_  
☐ OUTROS: \_\_\_\_\_

5. Sugestões de funcionalidades para o sistema:

- ☐ CADASTRAR CURRICULOS  
☐ PESQUISAR CURRICULOS POR CURSO  
☐ PESQUISAR CURRICULOS POR CURSO E PERIODO  
☐ CADASTRAR EMPRESAS  
☐ CADASTRAR OFERTAS DE ESTAGIOS E EMPREGOS  
☐ PESQUISAR OFERTAS POR CURSO  
☐ PESQUISAR OFERTAS POR CURSO E PERIODO  
☐ OUTROS: \_\_\_\_\_  
☐ OUTROS: \_\_\_\_\_  
☐ OUTROS: \_\_\_\_\_