

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO RIO
GRANDE DO NORTE

CAMILA BARBOSA GOMES DE ARAÚJO

**ELECTROSEARCH: UM SISTEMA PARA PESQUISA DE COMPONENTES
ELETRÔNICOS NO AMBIENTE DE LABORATÓRIO**

CURRAIS NOVOS/RN
2014

CAMILA BARBOSA GOMES DE ARAÚJO

**ELECTROSEARCH: UM SISTEMA PARA PESQUISA DE COMPONENTES
ELETRÔNICOS NO AMBIENTE DE LABORATÓRIO**

Trabalho de conclusão de curso apresentado à Coordenação do Curso Técnico de nível médio em Informática na forma integrada do Campus Currais Novos do Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte, em cumprimento às exigências legais como requisito parcial à obtenção do título de Técnico Integrado em Informática.

Orientador: Prof. Esp. Alfredo Rodrigues de Lima

Co-orientador: Prof. Esp. Pedrina Célia Brasil

CURRAIS NOVOS/RN
2014

CAMILA BARBOSA GOMES DE ARAÚJO

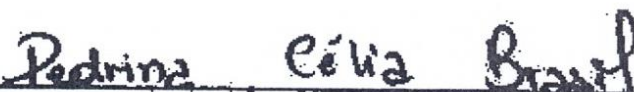
**ELECTROSEARCH: UM SISTEMA PARA PESQUISA DE COMPONENTES
ELETRÔNICOS NO AMBIENTE DE LABORATÓRIO**

Trabalho de conclusão de curso apresentado à
Coordenação do Curso Técnico de nível médio em
Informática na forma integrada do Campus Currais
Novos do Instituto Federal de Educação, Ciência e
Tecnologia do Rio Grande do Norte, em
cumprimento às exigências legais como requisito
parcial à obtenção do título de Técnico Integrado
em Informática.

Trabalho de Conclusão de Curso aprovado em 19/01/2015: 95



Prof. Esp. Alfredo Rodrigues de Lima - Orientador
1856443



Prof. Esp. Pedrina Célia Brasil - Co-orientadora
2131716



Prof. Msc. Marcílio Meira - Coordenador dos Cursos Técnicos em Informática
1831023

RESUMO

Este trabalho de conclusão de curso propõe uma solução ao problema verificado no Laboratório de Eletricidade e Eletrônica do Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte – Campus Currais Novos. Nesse ambiente de aulas, tanto os alunos quanto os professores não possuem consciência precisa de quais são os componentes existentes no laboratório, o que torna mais lenta a realização de trabalhos e projetos práticos. Logo, é proposta a elaboração de um sistema desktop que resolva tal conflito. A partir desse, será realizada a gestão dos componentes existentes no laboratório. A meta é prover um sistema em que docentes e discentes possam encontrar os componentes mecânicos e eletrônicos presentes no laboratório, bem como suas especificações. Com a implantação do sistema, almeja-se facilitar a rotina de projetos e práticas das disciplinas ministradas no ambiente laboratorial. O programa foi desenvolvido utilizando a linguagem Java, o padrão UML 2.0, o Sistema de Gerenciamento de Banco de Dados (SGBD) MySQL Workbench e a IDE do NetBeans 8.0. Tais tecnologias tiveram auxílio de técnicas de desenvolvimento tais como MVC – Modelo, Visão, Controle – e possibilitaram um desfecho mais organizado, bem estruturado e de fácil manutenção do software.

Palavras-chave: Laboratório de Eletricidade e Eletrônica. Componentes mecânicos e eletrônicos. Ambiente desktop. Java. MySQL Workbench.

ABSTRACT

This course conclusion paper proposes the solution for the problem situation verified on the Laboratory of Electricity and Electronic of Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte – Campus Currais Novos. At the place of classes and practices, as students and teachers do not have total conscience of what are the accessible components on the laboratory, which make slower the work and project's realization. By the way, it's proposed the elaboration of a software to solve that conflict. The application should be implemented to a desktop environment and intend to keep constant updates at the Institute's local network. The goal is turn possible a simple research to the database, where teachers and students may find plenty of mechanical and electronical's components placed at the laboratory, as well as your specifications. With the system implantation, it's an achievement to make projects, classes and practices' routine easier to the matters taught at the work environment. The application was made using Java, the pattern UML 2.0, the MySQL WorkBench and the NetBeans IDE 8.0. These technologies were supported by object-oriented programming's techniques and architecture MVC (Model, View, Controller), the same ones turned possible an organized, well done end and an easier maintenance of the software.

Keywords: Laboratory of Electricity and Electronic. Mechanical and electronical's components. Desktop environment. Java. MySQL Workbench.

LISTA DE ILUSTRAÇÕES

- Quadro 1 – Síntese de Carga horária e Atividades
- Quadro 2 – Representação dos Requisitos funcionais do sistema
- Quadro 3 – Representação dos Requisitos não funcionais do sistema
- Quadro 4 – Representação dos Requisitos Suplementares do sistema
- Quadro 5 – Representação dos atores do diagrama de casos de uso
- Quadro 6 – Representação dos casos de uso
- Quadro 7 – Descrição do CDU 1
- Quadro 8 – Descrição do CDU 2
- Quadro 9 – Descrição do CDU 3
- Quadro 10 – Descrição do CDU 4
- Quadro 11: Contratos de operação do diagrama de sequência da Imagem 5
- Imagem 1 – Exemplo genérico de um diagrama de casos de uso
- Imagem 2 – Diagrama de Casos de Uso
- Imagem 3 – Diagrama do Modelo Conceitual
- Imagem 4 – Diagrama de Classes
- Imagem 5 – Diagrama de Sequência
- Imagem 6 – Representação ilustrada da arquitetura MVC
- Imagem 7 – Projeto Arquitetural do sistema
- Imagem 8 – Print da implementação dos padrões Singleton e Facade no sistema ElectroSearch
- Imagem 9 – Print da tela do sistema onde o padrão observer foi implementado
- Imagem 10 – Print do código referente a implementação do padrão observer
- Imagem 11: Print da tela de login do sistema
- Imagem 12: Print da tela responsável por manter itens
- Imagem 13: Print da tela que exibe a pesquisa
- Imagem 14: Print da tela de cadastro de itens
- Imagem 15: Tela de edição de itens
- Imagem 16: Painel inicial do usuário Professor
- Imagem 17: Tela de configuração de usuários
- Imagem 18: Tela de cadastro de usuários
- Imagem 19: Tela de cadastro de usuários

SUMÁRIO

1 INTRODUÇÃO	7
1.1 JUSTIFICATIVA	8
1.2 OBJETIVOS	8
1.2.1 Objetivos Gerais	8
1.2.2 Objetivos Específicos	9
1.3 METODOLOGIA	9
2 DADOS GERAIS DA PESQUISA/EXTENSÃO	10
2.1 SÍNTESE DE CARGA HORÁRIA E ATIVIDADES	10
3 FUNDAMENTAÇÃO TEÓRICA	12
3.1 A LINGUAGEM DE MODELAGEM UML	12
3.2 O JAVA COMO LINGUAGEM DE PROGRAMAÇÃO	13
3.3 O BANCO DE DADOS: A LINGUAGEM SQL	15
3.4 A CONEXÃO JAVA COM O BANCO DE DADOS: JDBC	16
3.5 A COMUNICAÇÃO NA REDE: BANCO DE DADOS E SISTEMA	17
4 DESENVOLVIMENTO	18
4.1 PROJETO	18
4.1.1 Documento de Visão Geral do Sistema	18
4.1.2 Análise de Requisitos	19
4.1.3 Casos de Uso	21
4.1.4 Modelagem Conceitual	27
4.1.5 Diagrama de Classes	28
4.1.6 Diagrama de Sequência	29
4.1.7 Contratos de Operação	30
4.1.8 Projeto Arquitetural	31
4.1.9 Padrões de Projeto	33
6 CONCLUSÃO	47
REFERÊNCIAS BIBLIOGRÁFICAS	48

1 INTRODUÇÃO

A tecnologia da informação tornou-se indispensável para o ambiente acadêmico. Do setor administrativo às salas de aula, há inúmeras ferramentas que facilitam a ação de diretores, pedagogos, coordenadores, professores e alunos. São soluções – equipamentos e softwares – que agilizam o dia a dia dos corpos docente e discente. À proporção que esses elementos se tornam regulares, a necessidade por eles também aumenta.

Os laboratórios do IFRN encontram-se neste contexto. Há no Instituto a necessidade de explorar o desenvolvimento de um sistema de gestão de laboratório, voltado para o gerenciamento de equipamentos do Laboratório de Eletricidade e Eletrônica. Apesar de não ser o único a carecer de tal suporte, o laboratório é o que mais exige da criatividade dos alunos, por ser o que propõe projetos os quais pleiteiam o conhecimento amplo dos componentes ali presentes.

A rotina do laboratório de eletrônica é tomada por projetos, aulas práticas e atividades que exigem do aluno um planejamento dentro e fora do ambiente laboral. Através desse conhecimento, foi identificada a necessidade de software capaz de auxiliar a pesquisa e desenvolvimento de projetos no perímetro do campus. Para tal, o presente trabalho propõe o desenvolvimento de um sistema de cadastro e pesquisa de componentes eletrônicos para os laboratórios do IFRN.

Dentre as tecnologias utilizadas no desenvolvimento do sistema para gerenciamento do laboratório de eletrônica estão Java 8 up.25, MySQL Workbench 6.2, padrão UML 2.0 e o NetBeans IDE 8.0.

Visando atingir seus objetivos, este trabalho foi organizado da seguinte maneira: 1- Introdução, onde é exposto o contexto, objetivos e motivação do trabalho; 2- Fundamentação teórica necessária a implementação do sistema, onde há a descrição dos aspectos importantes e tecnologias utilizadas no processo de desenvolvimento; 3- Secção para descrição do desenvolvimento do software, na qual todos os aspectos do projeto são apresentados, como também um capítulo destinado ao manual do sistema para pesquisa de componentes eletrônicos em ambiente de laboratório – ElectroSearch.

1.1 JUSTIFICATIVA

Este trabalho se justifica na necessidade de um software de controle, gerenciamento e pesquisa de componentes para o ambiente de Laboratório de Eletricidade e Eletrônica do IFRN – Campus Currais Novos. Como apresentado na secção anterior, a execução de projetos e aulas práticas tornar-se-ia facilitada com esta sistema.

As disciplinas que usufruem do laboratório para suas práticas costumam realizar projetos para conclusão da matéria, culminando todo o conteúdo ministrado numa aplicação útil no cotidiano. Para isso, os alunos usufruem tanto dos componentes providos pelo Instituto, quanto compram novos componentes adequados a seu projeto.

O princípio da implementação do software é agilizar o processo de realização desses projetos, permitindo consulta e gerenciamento dos componentes presentes no laboratório em um computador conectado na rede local da instituição de ensino.

1.2 OBJETIVOS

Esta secção apresenta os objetivos do trabalho, conforme descritos a seguir.

1.2.1 Objetivos Gerais

Este Trabalho de Conclusão de Curso (TCC) visa modelar e desenvolver um software com finalidade de controle, gerenciamento e pesquisa de componentes eletrônicos do laboratório de eletricidade e eletrônica do IFRN – Campus Currais Novos. Esse sistema deve atuar na rede local do Instituto e permitir consultas no seu perímetro.

1.2.2 Objetivos Específicos

Os objetivos específicos deste Trabalho de Conclusão de Curso são:

- Compreender as necessidades do laboratório de eletricidade e eletrônica;
- Auxiliar a realização de projetos e de aulas práticas no laboratório de eletricidade e eletrônica;
- Tornar a localização e manutenção dos componentes do laboratório mais rápida.
- Automatizar e centralizar a gestão dos componentes do laboratório.

1.3 METODOLOGIA

A metodologia utilizada para a realização deste trabalho está dividida nas seguintes etapas: (i) Análise de requisitos, (ii) Modelagem, (iii) Implementação, e (iv) Documentação.

Durante a análise de requisitos do sistema foram realizadas reuniões e encontros com um dos responsáveis pelo laboratório de eletricidade e eletrônica para auxiliar a compreensão do software a ser desenvolvido.

Na etapa de modelagem, o projeto do sistema foi elaborado. Através do modelo UML foram gerados diagramas com o propósito de descrição e melhor entendimento do projeto.

Durante o período de implementação, ocorreu a programação do software, na linguagem Java e com o banco de dados SQL. Na última etapa, a documentação, desenvolvida durante todo o processo, inclui-se a redação dos textos produzidos que compõem este trabalho de conclusão e curso.

2 DADOS GERAIS DA PESQUISA/EXTENSÃO

TÍTULO DO PROJETO: ElectroSearch: um sistema para pesquisa de componentes eletrônicos em laboratório

PERÍODO DE REALIZAÇÃO: 12 meses

TOTAL DE DIAS: 180 dias

TOTAL DE HORAS: 720 horas

2.1 SÍNTESE DE CARGA HORÁRIA E ATIVIDADES

Quadro 1: Síntese de Carga horária e Atividades

MÊS	CARGA HORÁRIA	ATIVIDADES DESENVOLVIDAS
Março	80 horas	- Levantamento de requisitos;
Abril	80 horas	- Elaboração dos casos de uso e do documento de visão geral do sistema; - Análise e correção dos casos de uso e do documento de visão geral do sistema;
Maio	80 horas	- Elaboração da modelagem conceitual; - Visualização das primeiras telas do sistema; - Análise e correção da modelagem conceitual e dos outros documentos já produzidos;
Junho	80 horas	- Elaboração do diagrama de classes; - Estruturação da arquitetura do sistema; - Estruturação do banco de dados; - Início da implementação: tela de cadastro, exclusão e edição de itens; - Testes e correções do diagrama de classes e dos documentos já produzidos;
Julho	80 horas	- Elaboração do diagrama de sequência; - Implementação da tela de configuração dos usuários; - Testes e correções do diagrama de sequência e dos documentos já produzidos;
Agosto	80 horas	- Elaboração do diagrama de pacotes; - Implementação do sistema de login; - Testes e correções do diagrama de pacotes e dos documentos já produzidos; - Correções de erros e tratamento de exceções;
Setembro	80 horas	- Correções de erros e tratamento de exceções; - Estudo sobre Padrões de Projeto;
Outubro	80 horas	- Ajuste da arquitetura do sistema; - Correções de erros e tratamento de exceções; - Ajuste dos documentos produzidos ao sistema quase elaborado;

		<ul style="list-style-type: none">- Alteração no banco de dados do sistema;- Alteração no documento de visão geral do sistema;
Novembro	80 horas	<ul style="list-style-type: none">- Adequação do sistema às mudanças recentes;- Implementação da função que possibilita a comunicação em rede;- Realização de testes de sistema

Fonte: Elaborado pelos autores

3 FUNDAMENTAÇÃO TEÓRICA

Este capítulo visa apresentar os conceitos e terminologias relacionadas com este trabalho. Primeiramente é abordada a linguagem de modelagem utilizada para analisar e projetar o software. Em seguida, é abordado o ambiente de programação Java, seguido pela linguagem de manipulação de banco de dados (BD) SQL. Por fim, são apresentadas as outras tecnologias que permitiram a concretização dos objetivos deste trabalho (o Driver JDBC e a atualização do BD em rede).

3.1 A LINGUAGEM DE MODELAGEM UML

As linguagens gráficas de modelagem existem há muito tempo, entretanto existe uma enorme controvérsia sobre seu papel na indústria do software. Essas controvérsias afetam diretamente o modo como percebem a UML em si (FOWLER, 2005, p. 25).

“A UML (Unified Modeling Language) é uma linguagem-padrão para a elaboração da estrutura de projetos de software”, segundo Booch, Rumbaugh e Jacobson (2005, p. 13). Pode ser empregada para a visualização, construção, especificação e documentação de artefatos do software a ser desenvolvido.

A UML pode ser tratada de três maneiras, segundo Larman (2007, p. 39): como rascunho, como planta de software e como linguagem de programação. No primeiro caso, há a elaboração de diagramas incompletos e informais criados para explorar a gama de soluções para o sistema, conquanto explorando o poder das linguagens visuais. No segundo caso, há a elaboração de diagramas de projeto relativamente detalhados usados tanto para entender o problema, quanto para geração do código do software. Na terceira e última maneira, ocorre a interpretação da UML como a especificação executável completa de um sistema de software nesta linguagem.

Para Booch, Rumbaugh e Jacobson (2005, p. 13):

A UML é adequada para modelagem de sistemas, cuja abrangência poderá incluir sistemas de informação corporativos a serem distribuídos a

aplicações baseadas em Web e até sistemas complexos embutidos de tempo real. É uma linguagem muito expressiva, abrangendo todas as visões necessárias ao desenvolvimento e implantação desses sistemas.

Por muitos de seus diagramas serem repletos de símbolos e gráficos, leva a interpretação errônea de que é um amontoado de figuras sem significado expressivo para o desenvolvimento do software. Todavia, por trás de cada símbolo empregado na notação UML existe uma semântica bem-definida. Logo, quando um desenvolvedor criar um diagrama, outro desenvolvedor, ou até mesmo uma ferramenta, será capaz de interpretá-lo sem ambiguidades (BOOCH; RUMBAUGH; JACOBSON, 2005, p. 15).

Em particular, a UML atende a todas as decisões em termos de análise, projeto e implementação de um software. Especificar significa construir modelos precisos, sem ambiguidades e completos. Logo, tendo em vista o raio de abrangência da linguagem para o processo de desenvolvimento de um sistema, percebe-se a importância dela.

Apesar de não ser uma linguagem visual de programação, os modelos em UML podem ser diretamente conectados a várias linguagens de programação. Em outras palavras, é possível mapear os modelos UML em Java, C++, Python, Visual Basic ou até em tabelas de bancos de dados relacionais. “A UML é capaz de representar tudo que possa ser melhor expresso em termos gráficos, enquanto as linguagens de programação representam o que é melhor expresso em termos textuais.” (BOOCH; RUMBAUGH; JACOBSON, 2005, p. 16).

3.2 O JAVA COMO LINGUAGEM DE PROGRAMAÇÃO

Para implementação do sistema proposto foi utilizada a linguagem de programação Java. Dentre as inúmeras vantagens do Java estão: sintaxe agradável, semântica compreensível, uma biblioteca ampla e a portabilidade para diversos sistemas operacionais.

O Java tornou-se a linguagem preferida para implementar aplicativos baseados na internet e software para dispositivos que se comunicam por uma rede – o caso do ElectroSearch – (DEITEL; DEITEL, 2010, p. 3). Durante o processo de

desenvolvimento, o NetBeans IDE 8.0 foi utilizado junto ao JDK (Java Development Kit) 8up25 para facilitar e agilizar a codificação da aplicação.

Um artigo influente escrito pelos autores do Java explica seus objetivos de design e realizações. Eles também publicaram um resumo discorrendo em torno destas onze características: simples, orientado a objetos, portátil, interpretado, compatível com redes, alto desempenho, robusto, múltiplos threads, seguro, dinâmico e arquitetura neutra (HORSTMANN, CORNELL, 2010, p. 1). Através dessas características foi identificada a plataforma Java como ambiente de desenvolvimento para o sistema.

A linguagem é uma versão limpa da sintaxe do C++, omitindo a necessidade de arquivos de cabeçalho, aritmética de ponteiros, estruturas, uniões, sobrecarga de operadores e classes básicas virtuais. Logo, o estudo, entendimento e programação neste ambiente tornou-se o mais indicado.

O Java é uma linguagem orientada a objetos, logo como paradigma de programação predominante hoje em dia, substituiu as técnicas de programação estruturada desenvolvidas nos anos de 1970.

Um programa orientado a objetos é composto de objetos, cuja funcionalidade específica é exposta aos usuários, e a implementação, oculta. Muitos objetos nos seus programas serão objetos “prontos” em uma biblioteca; outros são projetados de uma maneira personalizada (HORSTMANN, CORNELL, 2010, p. 6).

Não obstante, outras definições de POO também são muito aceitas. Por ter sido criado para aproximar os mundos real e virtual, esse paradigma consiste basicamente nos conceitos de objetos, classes, herança e polimorfismo. Desses, os dois itens citados primariamente foram usados na implementação do sistema descrito neste documento.

O Java ainda é extremamente compatível com redes, ainda no artigo publicado pelos autores da linguagem (GOSLING, 1998):

O Java tem uma extensa biblioteca de rotinas para lidar com protocolos TCP/IP como HTTP e FTP. Os aplicativos Java podem abrir e acessar objetos pela internet via URLs com a mesma

facilidade que ao acessar um sistema de arquivos local.

Além disso, o Java também tem robustez no ambiente de programação. O compilador detecta problemas os quais, em outras linguagens de programação, só viriam a ser notados em tempo de execução.

O compilador Java ainda gera um formato de arquivo de objetos neutro em relação à arquitetura, ou seja, o código compilável pode ser executado em diversos tipos de processadores.

Portanto, a linguagem Java foi a escolhida para o desenvolvimento do software ElectroSearch. Implementação em POO, robustez, compatibilidade com redes e a neutralidade para com a arquitetura foram os principais aspectos considerados na designação da linguagem de programação.

A versão do Java utilizada na implementação do sistema foi a mais recente durante o período de desenvolvimento do programa, a 8up25. Dentre suas principais mudanças em comparação a versões anteriores estão algumas alterações na sintaxe, bem como atualizações de bibliotecas e coleções. Para auxiliar o desenvolvimento, há ainda o Java Development Kit (JDK).

O JDK é um conjunto de utilitários cuja a finalidade consiste na permissão para criação de jogos e programas para a plataforma. Este pacote é disponibilizado pela Oracle, e contém todo o ambiente necessário para a criação e execução dos aplicativos Java.

O NetBeans IDE 8.0, também, foi utilizado como IDE (Integrated Development Environment – Ambiente de desenvolvimento integrado) para auxílio no desenvolvimento do sistema ElectroSearch.

3.3 O BANCO DE DADOS: A LINGUAGEM SQL

Structured Query Language (SQL) é um conjunto de comandos de manipulação de banco de dados utilizado para criar e manter a estrutura de uma base de dados, além de incluir, excluir, modificar e pesquisar informações nas tabelas contidas nele.

Segundo Elmasri e Navathe (2005, p. 149),

A SQL é uma linguagem de banco de dados abrangente: ela possui comandos para definição de dados, consultas e atualizações. Assim, ela tem ambas DDL e DML. Além disso, tem funcionalidades para definição de visões (*views*) no banco de dados, a fim de especificar a segurança e as autorizações para definições de restrições de integridade e de controles de transação

As linguagens DDL e DML, citadas na transcrição de Elmasri e Navathe, consistem nas linguagem de definição de dados e linguagem de modificação de dados, respectivamente.

3.4 A CONEXÃO JAVA COM O BANCO DE DADOS: JDBC

Para conectar-se e manipular os dados de um banco de dados, as aplicações Java utilizam o JDBC (Java DataBase Connectivity) API. Esta API permite que desenvolvedores alterem o banco de dados subjacente sem modificar o código Java que acessa o BD. Os sistemas de gerenciamento de BDs mais populares fornecem drivers JDBC. O driver do MySQL – tipo de banco de dados utilizado na implementação do ElectroSearch, é provido por seu fabricante.

As seguintes interfaces e classes são necessárias para utilização do JDBC: `java.sql.Connection`, `java.sql.Statement`, `java.sql.DriverManager`, `java.sql.ResultSet`, `java.sql.ResultSetMetaData` e `java.sql.SQLException`.

A partir do Java SE 6 – a atual versão é 8u25 –, há o suporte à descoberta de driver automática. Todavia, é preciso assegurar que o programa tenha a capacidade de localizar a classe do driver do banco de dados. Para tanto, deve-se incluir a localização da classe no classpath do programa ao executá-lo. Para MySQL, o arquivo `mysql-connector-java-5.1.7-bin.jar` é incluído em algum pacote do sistema (DEITEL; DEITEL, 2010, p. 912)

Nas linhas de código, uma conexão básica com o BD pode ser efetuada criando um objeto `Connection`. Esse implementa a interface que leva seu nome e gerencia a conexão entre o programa Java e o banco de dados. É recomendável

inicializar a variável com o resultado de uma chamada para o método `static getConnection` da classe `DriverManager`, que tenta conectar-se ao banco de dados especificado pela URL fornecida, do mesmo modo com os outros atributos desse método: os usuário e senha do BD (DEITEL; DEITEL, 2010, p. 913).

3.5 A COMUNICAÇÃO NA REDE: BANCO DE DADOS E SISTEMA

Visando utilizar uma base de dados centralizada e unificada que pudesse ser acessada a partir de qualquer terminal, foi usada a biblioteca para tratamento de arquivos que a linguagem Java oferece.

Através de um arquivo localizado dentro do subdiretório do sistema chamado “`configbd`”, o usuário poderá informar o endereço de IP (Internet Protocol) do servidor de banco de dados utilizado, ou seja, um servidor de banco de dados MySQL que possua a base de dados do sistema.

Este processo vem de tal forma a facilitar a vida do usuário, uma vez que quando o IP do servidor de banco for alterado, bastará alterar as configurações (IP do servidor) no arquivo “`configdb`” do sistema cliente.

4 DESENVOLVIMENTO

Neste capítulo são apresentadas as fases de especificação, projeto e implementação do sistema. Contém a análise de requisitos, a modelagem do sistema, diagramas de classe e de sequência. Na fase de implementação são apresentados os aspectos referentes aos padrões de projeto utilizados na codificação, bem como a materialização e validação do sistema.

A modelagem foi feita através da linguagem de modelagem UML e o programa implementado em Java. Nas próximas seções serão apresentados os requisitos funcionais, as especificações e os diagramas que representam esse projeto.

4.1 PROJETO

Nesta secção são documentadas as fases de especificação e projeto do sistema.

4.1.1 Documento de Visão Geral do Sistema

É proposto o desenvolvimento de um sistema de pesquisa e controle dos componentes inseridos no laboratório de eletrônica do IFRN – Campus Currais Novos. Tal sistema visa auxiliar as aulas práticas, atividades e projetos realizados nesse meio através da busca dos itens disponíveis para a atividade laboral. Deverá haver a possibilidade de registro, edição e exclusão de componentes para TAL's e professores; bem como, as funcionalidades de criar, modificar e desassociar usuários do sistema, estas últimas como responsabilidade somente do servidor. Assim, deverão existir três tipos de usuários: professor, TAL e aluno; os dois primeiros, como realizam funções que modificam o banco de dados do sistema, necessitam de autenticação, ou seja, de login e senha; enquanto o último, potencializado apenas com pesquisa de componentes, tem acesso direto a tal funcionalidade.

O sistema de busca de componentes deve ter quatro vias: uma pesquisa através do tipo – potenciômetros, transistores, diodos, entre outros -, outra através do nome do produto – 1N4008, 555 –, id e uma lista dos itens disponíveis, beneficiando a procura de informações pelo usuário.

Durante o cadastro de um componente, deve ser informado nome, tipo, descrição, referência e disponibilidade; e, na inserção de um usuário para manuseio do sistema, deve-se informar nome, matrícula (usada para login), senha, uma dica de senha e o tipo de usuário – professor ou aluno -.

4.1.2 Análise de Requisitos

O desenvolvimento de sistemas tem início com a identificação da necessidade do cliente, em outras palavras, identificar o que o cliente quer é o ponto de partida na evolução de um sistema baseado em computador. Entretanto, um dos maiores problemas enfrentados no desenvolvimento de sistemas grandes e complexos são os requisitos.

É papel do analista ou engenheiro identificar o problema inicial, e em seguida as metas globais. Dentre os objetivos dessa comunicação com o cliente estão os primeiros rascunhos das funções e desempenho desejados, das questões de confiabilidade e qualidade, dos requisitos de produção, das tecnologias disponíveis e de possíveis extensões futuras.

Para Larman (2007, p. 81), “Requisitos são capacidades e condições às quais o sistema – e em termos mais amplos, o projeto – deve atender”. Logo, um desafio básico da análise de requisitos é encontrar, comunicar e documentar o que realmente é necessário, fazendo o possível para expressar de forma clara para o cliente e os membros da equipe de desenvolvimento.

A seguir os que representam os requisitos funcionais, não-funcionais e suplementares do sistema implementado:

Quadro 2: Representação dos Requisitos funcionais do sistema

Cód.	Nome	Descrição	Oculto
F01	Registro de Componentes	O sistema deve cadastrar componentes, bem como suas especificidades	()
F02	Pesquisa de	O programa deve permitir a pesquisa de	()

	Componentes	componentes, seja por tipo (potenciômetros, transistores, diodos) ou por nome (1N4008, 555).	
F03	Exclusão de Componentes	O sistema deve permitir a exclusão de componentes.	()
F04	Edição de Componentes	O sistema deve permitir a edição das especificações de um componente, possibilitando assim, correções e atualizações do produto.	()
F05	Cadastro de Usuários	O sistema deve possuir a função de cadastrar usuários, sejam estes um professor ou um TAL.	()
F06	Exclusão de Usuários	O sistema deve possuir a função de excluir usuários, sejam estes um professor ou um TAL.	()
F07	Edição de Usuários	O sistema deve possuir a função de editar usuários, sejam estes um professor ou um TAL.	()
F08	Sistema de Login	Deve haver um sistema de login destinado a professores e TAL's. Caso o usuário seja um "aluno comum", ele será direcionado a outra interface com funções reduzidas.	()
F09	Consulta de Usuários	O programa deve permitir a pesquisa de usuários para que se torne possível a edição, exclusão ou a consulta em si	()

Fonte: Elaborado pelos autores

Quadro 3: Representação dos Requisitos não funcionais do sistema

Cód.	Nome	Restrição	Categoria	Desejável	Permanente
NF01	Alterações de forma simultânea	O banco de dados do sistema deve ser capaz de realizar várias alterações simultaneamente	Usabilidade	(X)	(X)
NF02	Linguagem de programação	A linguagem de alto nível utilizada deve ser Java	Implementação	(X)	()
NF03	Banco de dados	A linguagem do SGBD deverá ser MySQL (Tabela)	Implementação	(X)	()

Fonte: Elaborado pelos autores

Quadro 4: Representação dos Requisitos Suplementares do sistema

Cód.	Nome	Restrição	Categoria	Desejável	Permanente
S1	Banco de Dados	O banco de dados utilizado deve ser online	Sistema	(X)	(X)
S2	Sistema Operacional	O software deve ser operado no Windows e no Linux.	Heterogeneidade	(X)	()

S3	Perfis de Usuário	Os perfis de usuário para acesso ao sistema são: 3. Professor: pode efetuar todas as operações; 2. TAL: pode efetuar as operações de pesquisa, registro, edição e exclusão de componentes; 1. Aluno: pode efetuar somente a função de pesquisar itens;	Segurança	()	(X)
----	-------------------	---	-----------	-----	-----

Fonte: Elaborado pelos autores

4.1.3 Casos de Uso

Os casos de uso são uma técnica para captar os requisitos funcionais de um sistema. Eles servem para descrever as interações típicas entre os usuários de um sistema e o próprio sistema, fornecendo um diagrama e a narrativa sobre como o software se comportará (FOWLER, 2005, p. 104).

Em outras palavras, o modelo de casos de uso é uma representação das funcionalidades externamente observáveis do sistema, bem como dos elementos externos ao sistema. Os casos de uso representam os requisitos funcionais, logo direcionam diversas atividades posteriores ao ciclo de vida do sistema de software. Além disso, força os desenvolvedores a moldar o sistema de acordo com as necessidades do usuário.

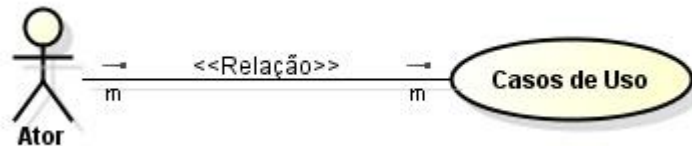
O modelo de casos de uso de um sistema é composto por duas partes: uma textual e uma gráfica. Essa segunda permite dar uma visão global e de alto nível do sistema. Possui três elementos principais: os casos de uso, os atores e os relacionamentos entre esses elementos.

Um caso de uso é uma especificação de uma sequência de interações entre um sistema e seus agentes externos. Logo, define parte da funcionalidade de um sistema, todavia, não revela a estrutura e o comportamento interno do software.

Os atores são elementos externos que interagem com o sistema. Ou seja, não fazem parte desse e trocam informações com o mesmo. Um ator corresponde a

um papel representado em relação ao sistema, logo o nome dado a um ator deve lembrar o seu papel e não quem o representa.

Imagem 1: Exemplo genérico de um diagrama de casos de uso

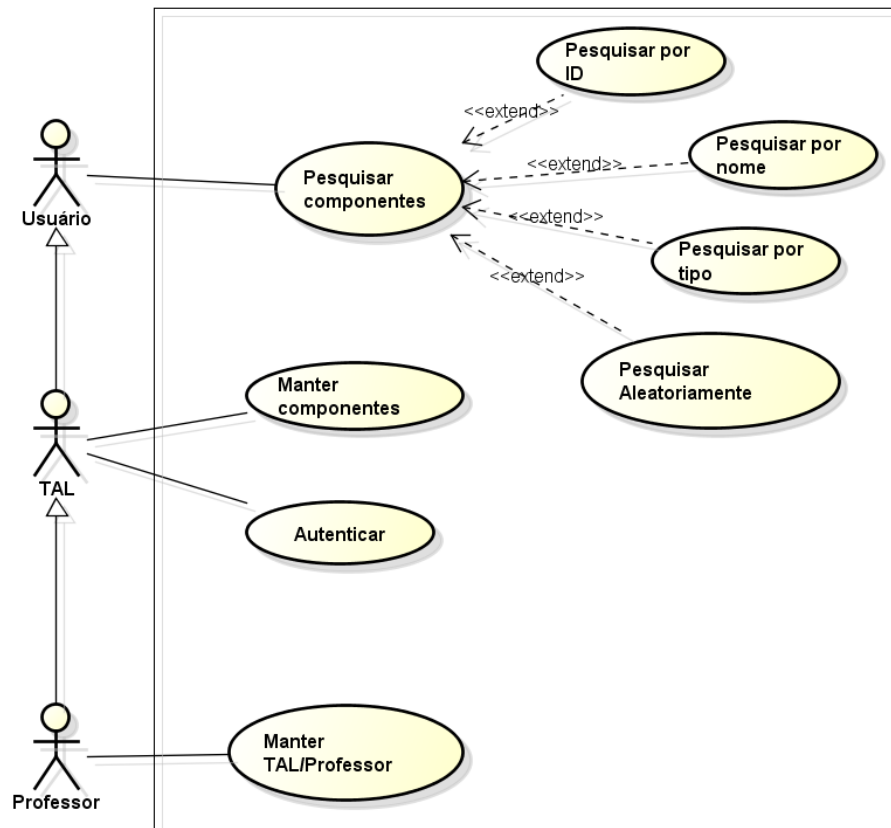


Fonte: Elaborado pelos autores

A parte textual consiste em narrativas, amplamente utilizadas para descobrir e registrar requisitos. Apresenta um conjunto de sequências de ações, incluindo variantes que um sistema realiza para produzir um resultado observável do valor do ator. Um caso de uso descreve o que um sistema faz, mas não especifica como isso é feito. (FOWLER, 2005, p. 104)

É possível especificar o comportamento de um caso de uso pela descrição do fluxo de eventos no texto de maneira suficientemente clara para que alguém externo ao projeto do sistema possa compreendê-lo facilmente (BOOCH; RUMBAUGH; JACOBSON, 2005, p. 232).

Encontram-se a seguir, o diagrama de casos de uso, os quadros referentes a este diagrama e a descrição textual dos CDUs.

Imagem 2: Diagrama de Casos de Uso

Fonte: Elaborado pelos autores

Quadro 5: Representação dos atores do diagrama de casos de uso

Nome	Descrição
Usuário	Representa pessoas matriculadas nas matérias que proporcionam acesso ao laboratório de eletrônica
TAL	Representa os responsáveis pelo laboratório, ou seja, alunos bolsistas que auxiliam os professores nas práticas da matéria técnica
Professor	Representa os servidores que ministram as matérias com acesso ao laboratório de eletrônica

Fonte: Elaborado pelos autores

Quadro 6: Representação dos Casos de Uso

Cód.	Caso de Uso	Descrição
CDU1	Manter Componentes	Refere-se às ações de adicionar, editar e excluir componentes do sistema ElectroSearch
CDU2	Manter Usuários	Refere-se às ações de adicionar, editar e excluir usuários do banco de dados do sistema
CDU3	Autenticar	Remete-se ao sistema de login e senha necessários a autenticação de TAL's e professores
CDU4	Pesquisar Componentes	Ação realizada por todos os atores do sistema que pode ser executada com base em três meios de pesquisa: ID, nome, tipo e disponibilidade

Fonte: Elaborado pelos autores

Quadro 7: Detalhamento do CDU1: Manter Componentes

Caso de uso CDU1: Manter componentes

Escopo: ElectroSearch.

Nível: objetivo do usuário.

Ator principal: TAL e Professor.

Pré-condições: É necessária a autenticação do TAL ou professor no sistema para que o caso de uso “manter componentes” seja realizado.

Pós-condições: Os componentes serão “mantidos” pelo sistema – adicionados, alterados e excluídos –, de modo que no fim de cada ação realizada com sucesso, não haja erros no banco de dados e a interface retorne ao menu inicial.

Cenário de Sucesso Principal (Fluxo Principal):

1. O TAL/Professor clica em “Manter Componentes” [IN]
2. O sistema lista os componentes cadastrados no sistema e oferece opções ao usuário de criar adicionar, editar e excluir componentes [OUT]

Extensões (Fluxos alternativos):

*a: cadastrar componentes:

1. O TAL/Professor clica no botão “Cadastrar” [IN]
2. O sistema exibe a interface de cadastro de itens [OUT]
3. O TAL/Professor preenche o formulário de cadastro [IN]
4. O TAL/Professor encerra o cadastro [IN]
5. O sistema verifica os dados e também se todos os campos foram preenchidos
6. O sistema exibe uma mensagem informando ao TAL/Professor que o componente foi cadastrado com sucesso [OUT]

*b: alterar componentes:

1. O TAL/Professor seleciona o componente a ser editado [IN]
2. O TAL/Professor clica em “Editar” [IN]
3. O sistema exibe na interface do usuário a tela de alteração de componentes – semelhante a de cadastro [OUT]
4. O TAL/Professor altera os dados do item [IN]
5. O TAL/Professor clica em “Salvar” [IN]
6. O sistema verifica os dados
7. O sistema exibe uma mensagem informando ao TAL/Professor que o componente foi editado com sucesso [OUT]

*c: excluir item:

1. O TAL/Professor seleciona o componente a ser excluído [IN]
2. O TAL/Professor clica em “Excluir” [IN]
3. O sistema exibe uma mensagem pedindo uma confirmação do usuário [OUT]
4. O TAL/Professor confirma a exclusão do item [IN]
- 4*c: O TAL/Professor cancela a exclusão do item [IN]
5. O sistema exclui o item selecionado do banco de dados
- 5*c: O sistema volta a tela de “Manter componentes” [OUT]

Fluxos de Erros:

*a: cadastrar componentes:

- 5.1. Cadastro de componente já incluso no banco de dados
- 6.1. O sistema exibe a mensagem: “Componente já inserido!”
- 7.1. Limpar formulário de cadastro de componentes
- 5.2. Ausência de campo essencial preenchido

6.2. O sistema exibe a mensagem “Falta preencher ‘tal’ campo”

7.2. O sistema retorna à tela de cadastro

8.2. Retorna à etapa 3 do fluxo principal

Requisitos especiais:

- Comunicação serial entre o terminal e demais computadores. Permitindo que haja o acesso ao software de um ponto qualquer.

- Busca rápida e eficiente no banco de dados, fazendo o que com o usuário não espere muito.

Fonte: Elaborado pelos autores

Quadro 8: Detalhamento do CDU2: Manter Usuários

Caso de uso CDU2: Manter usuários

Escopo: ElectroSearch.

Nível: objetivo do usuário.

Ator principal: Professor.

Pré-condições: É necessária a autenticação do professor no sistema para que o caso de uso “manter usuários” seja realizado.

Pós-condições: Os usuários serão “mantidos” pelo sistema – adicionados, alterados e excluídos –, de modo que no fim de cada ação realizada com sucesso, não haja inconsistências no banco de dados e a interface retorne ao menu inicial.

Cenário de Sucesso Principal (Fluxo Principal):

1. O Professor clica em “Manter Usuários” [IN]
2. O sistema lista os TAL's e Professores cadastrados no sistema e oferece opções ao Professor de criar adicionar, editar e excluir tais objetos [OUT]

Extensões (Fluxos alternativos):

*a: cadastrar usuários:

1. O Professor clica no botão “Cadastrar” [IN]
2. O sistema exibe uma mensagem perguntando ao Professor que tipo de usuário ele deseja cadastrar [OUT]
3. O Professor clica no botão dirigido ao tipo de usuário que ele deseja inserir [IN]
 - 3.1. O Professor seleciona “TAL” [IN]
 - 3.2. O Professor seleciona “Professor” [IN]
4. O sistema exibe a interface de cadastro relativa ao tipo selecionado pelo professor [OUT]
5. O Professor preenche o formulário de cadastro [IN]
6. O Professor encerra o cadastro [IN]
7. O sistema verifica os dados e também se todos os campos foram preenchidos
8. O sistema exibe uma mensagem informando ao Professor que o usuário foi cadastrado com sucesso [OUT]

*b: alterar usuários:

1. O Professor seleciona o usuário a ser editado [IN]
 2. O Professor clica em “Editar” [IN]
 3. O sistema exibe na interfase do usuário a tela de alteração de usuários – semelhante a de cadastro [OUT]
 4. O Professor altera os dados do item [IN]
 5. O Professor clica em “Salvar” [IN]
 6. O sistema verifica os dados
 7. O sistema exibe uma mensagem informando ao Professor que o objeto foi editado com sucesso [OUT]
-

*c: excluir usuários:

1. O Professor seleciona o usuário a ser excluído [IN]
2. O Professor clica em “Excluir” [IN]
3. O sistema exibe uma mensagem pedindo uma confirmação do usuário [OUT]
4. O Professor confirma a exclusão do item [IN]
- 4*c: O Professor cancela a exclusão do item [IN]
5. O sistema exclui o item selecionado do bando de dados
- 5*c: O sistema volta a tela de “Manter usuários” [OUT]

Fluxos de Erros:

*a: cadastrar usuários:

- 5.1. Cadastro de usuário já incluso no banco de dados
 - 6.1. O sistema exibe a mensagem: “TAL/Professor já inserido!”
 - 7.1. Limpar formulário de cadastro de TAL/Professor
 - 5.2. Ausência de campo essencial preenchido
 - 6.2. O sistema exibe a mensagem “Falta preencher ‘tal’ campo”
 - 7.2. O sistema retorna à tela de cadastro
-

Fonte: Elaborado pelos autores

Quadro 9: Detalhamento do CDU3: Autenticar

Caso de uso CDU3: Autenticar

Escopo: ElectroSearch.

Nível: objetivo do usuário.

Ator principal: TAL e Professor.

Pré-condições: Nenhuma.

Pós-condições: O TAL/Professor estará logado no sistema, apto a modificar o banco de dados (manter itens e manter usuários).

Cenário de Sucesso Principal (Fluxo Principal):

1. O TAL/Professor abre o executável do programa [IN]
2. O sistema exibe a tela de login para acesso ao terminal [OUT]
3. O TAL/Professor preenhe o login e a senha [IN]
4. O TAL/Professor clica em “Entrar” [IN]
5. O sistema verifica se todos os campos foram preenchidos e se os dados enviados estão em conformidade com o BD
6. O sistema exibe a tela inicial do TAL/Professor

Fluxos de Erros:

- 5.1. O TAL/Professor não preencheu todos os campos necessários a autenticação (login e senha)
- 6.1. O sistema exibe uma mensagem de erro: “O campo ‘tal’ não foi preenchido”
- 7.1. O fluxo retorna à etapa 3
- 5.2. O TAL/Professor insere o login ou senha incorretos
- 6.2. O sistema exibe uma mensagem de erro: “Login ou senha não válidos”
- 7.2. O fluxo retorna à etapa 3

Requisitos especiais:

- Comunicação serial entre o terminal e demais computadores. Permitindo que haja o acesso ao software de um ponto qualquer.

- Busca rápida e eficiente no banco de dados, fazendo o que com o usuário não espere muito.

Frequência de ocorrência: contínua.

Fonte: Elaborado pelos autores

Quadro 10: Detalhamento do CDU4: Pesquisar Componentes

Caso de uso CDU4: Pesquisar componentes

Escopo: ElectroSearch.

Nível: objetivo do usuário.

Ator principal: Usuário, TAL e Professor.

Pré-condições: Nenhuma

Pós-condições: A consulta ao banco de dados será realizada com sucesso, seja esta por ID, nome ou tipo do componente

Cenário de Sucesso Principal (Fluxo Principal):

1. O usuário clica em “Pesquisar componentes” [IN]
2. O sistema lista os componentes cadastrados no sistema e oferece opções de pesquisa ao usuário: por ID, por nome, por tipo ou por disponibilidade [OUT]
3. O usuário seleciona uma das opções [IN]
4. O usuário preenche o formulário de pesquisa [IN]
5. O usuário clica em “Pesquisar” [IN]
6. O sistema verifica os dados inseridos
7. O sistema exibe os resultados encontrados [OUT]
8. O usuário consulta a ficha de informações do componente especificado

Fluxos de Erros:

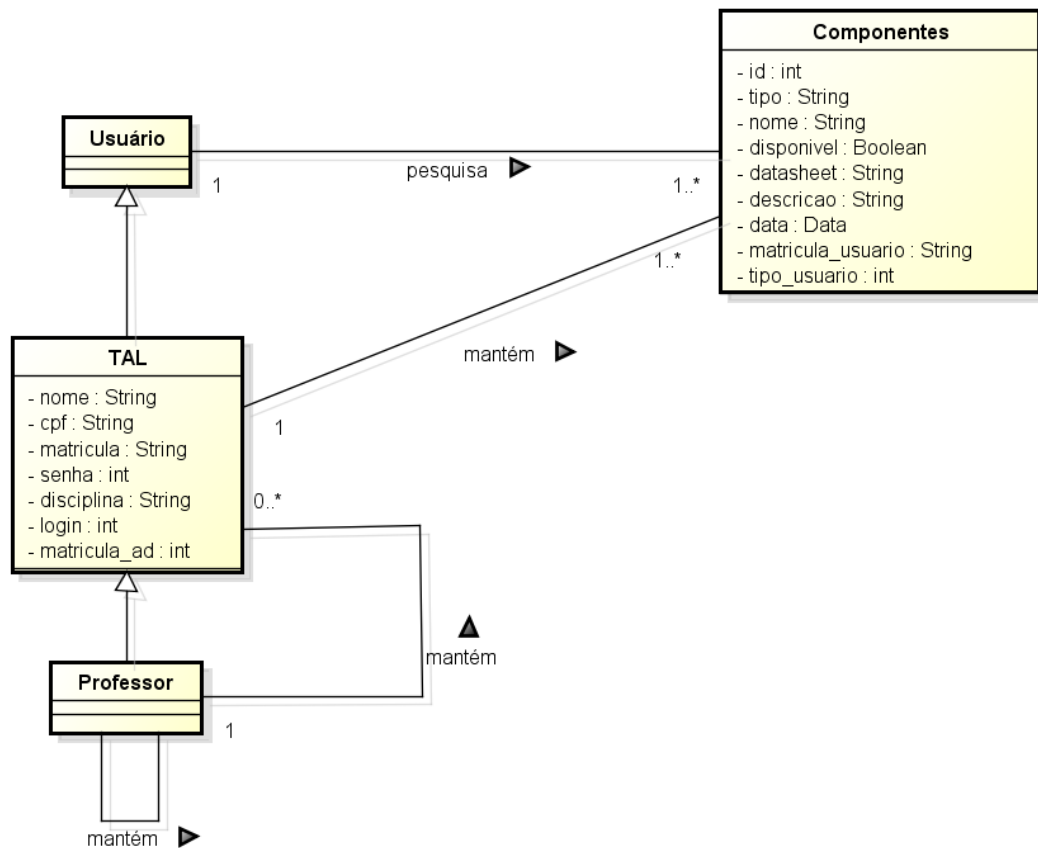
- 6.1. O usuário não digitou os campos obrigatórios à pesquisa
- 7.1 O sistema exibe uma mensagem requisitando a inserção de dados
- 8.1. O fluxo retorna à etapa 4 da sequência
- 6.2 O sistema não encontra nenhum item que relacione-se com a pesquisa do usuário
- 7.1. O sistema exibe uma mensagem: “Item não encontrado!”
- 8.1. O fluxo de atividades retorna à etapa 4 da sequência

Fonte: Elaborado pelos autores

4.1.4 Modelagem Conceitual

A modelagem conceitual de um sistema descreve as informações que o software irá gerenciar, através da representação de classes conceituais do mundo real. Esse modelo, que faz uso do diagrama de classes UML, é um artefato do domínio do problema, em outras palavras, faz parte da fase de investigação.

Imagem 3: Diagrama do Modelo Conceitual



Fonte: Elaborado pelos autores

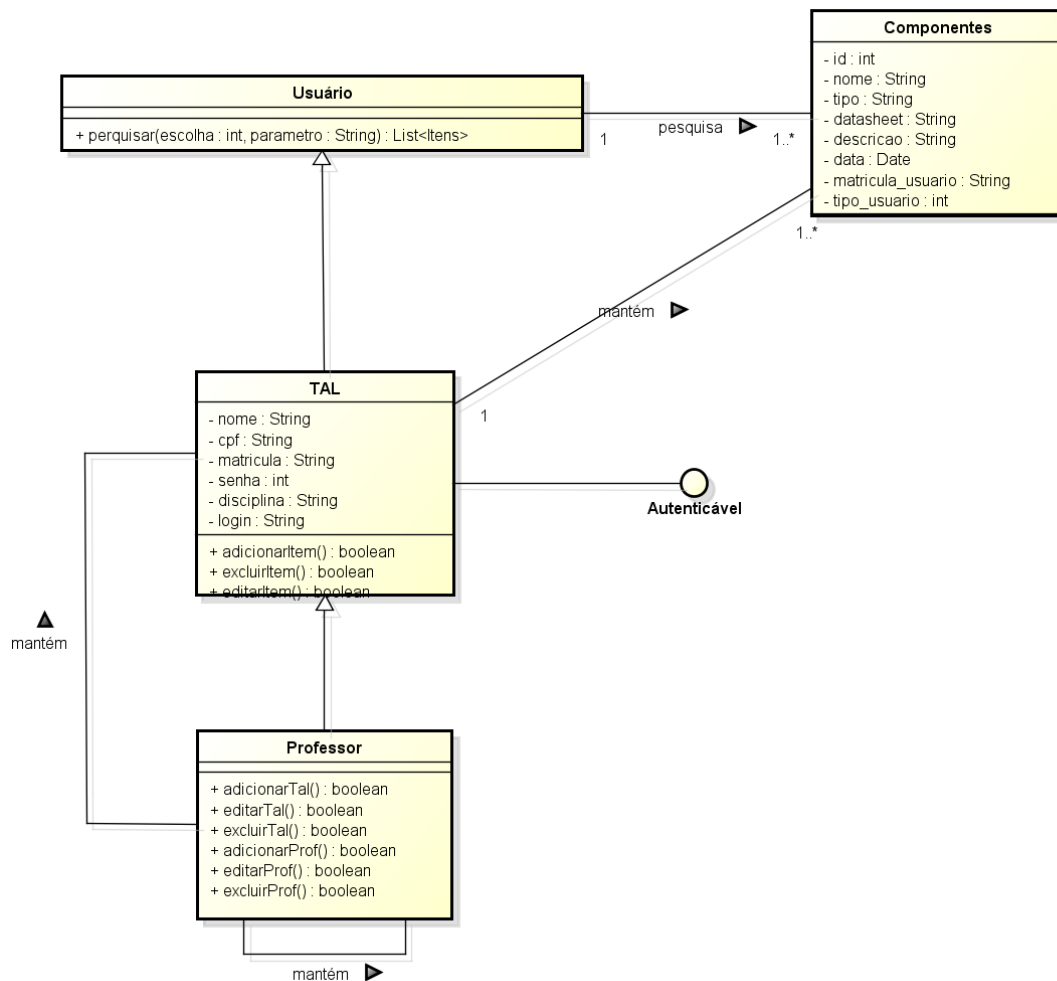
4.1.5 Diagrama de Classes

A UML inclui diagramas de classe para ilustrar classes, interfaces e suas associações. Eles são utilizados para modelagem estática de objetos, logo, auxilia na visualização do modelo de domínio (LARMAN, 2007, p. 266).

Os diagramas de classe são importantes não só para visualização, especificação e documentação de modelos estruturais, mas também para a construção de sistemas executáveis por intermédio de engenharia de produção e reversa (BOOCH; RUMBAUGH; JACOBSON, 2005, p.107).

Graficamente, o diagrama consiste numa coleção de vértices e arcos, juntos os quais representam os atributos e os métodos das classes a serem implementadas num sistema.

Imagem 4: Diagrama de Classes

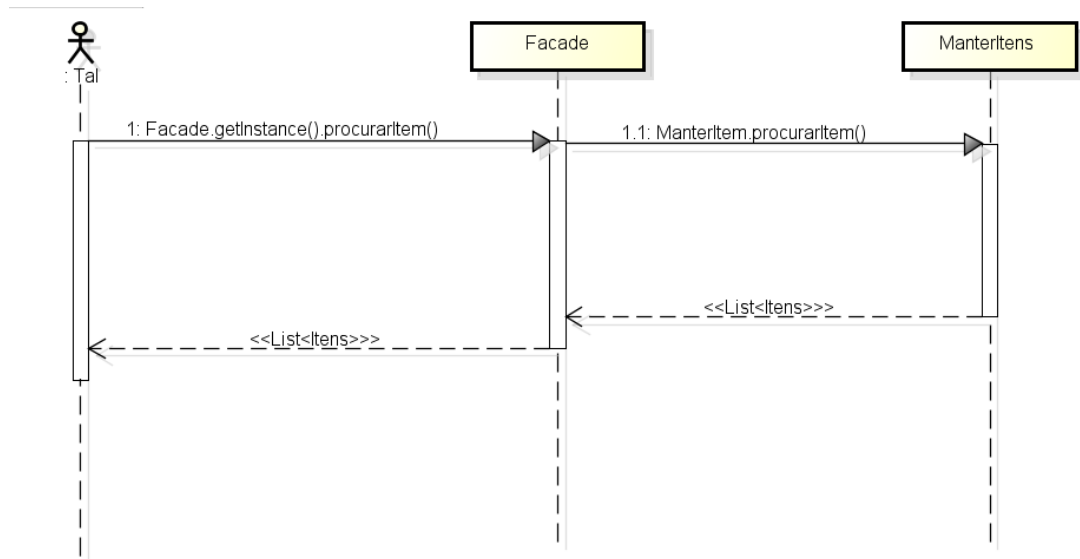


Fonte: Elaborado pelos autores

4.1.6 Diagrama de Sequência

O diagrama de sequência consiste num artefato criado rápida e facilmente o qual ilustra os eventos de entrada e saída relacionados com o sistema em discussão.

Tal qual os casos de uso descrevem como os atores externos interagem com o software, cada ator gera eventos de sistema solicitando alguma operação para tratar tal evento. Esse diagrama descreve tal sequência de eventos (LARMAN, 2007, p. 195).

Imagem 5: Diagrama de Sequência

Fonte: Elaborado pelos autores

4.1.7 Contratos de Operação

Contratos de operação usam uma forma pré e pós-condição para descrever modificações detalhadas em objetos em um modelo de domínio, como resultado de uma operação do sistema (LARMAN, 2007, p. 203).

A entrada de dados nos contratos de operação são os diagramas de interação. O diagrama de sequência é um exemplo.

Quadro 11: Contratos de operação do diagrama de sequência da Imagem 5

Contrato CO1: ManterItem.procurarItem

Operação: ManterItem.procurarItem(dado: String, escolha: inteiro)

Referências Cruzadas: Casos de uso: Pesquisar Componentes

Pré-Condições: Nenhuma.

Pós-Condições:

– Foi criada uma única instância do Facade

Contrato CO2: ManterItem.procurarItem

Operação: ManterItem.procurarItem(dado: String, escolha: inteiro)

Referências Cruzadas: Casos de uso: Pesquisar Componentes

Pré-Condições: Nenhuma.

Pós-Condições:

– Uma variável ps do tipo Prepared Statement foi instanciada

-
- A variável ps executou um método para busca
 - Foi efetuada uma busca no banco de dados.
 - A busca foi feita com sucesso.
-

Fonte: Elaborado pelos autores

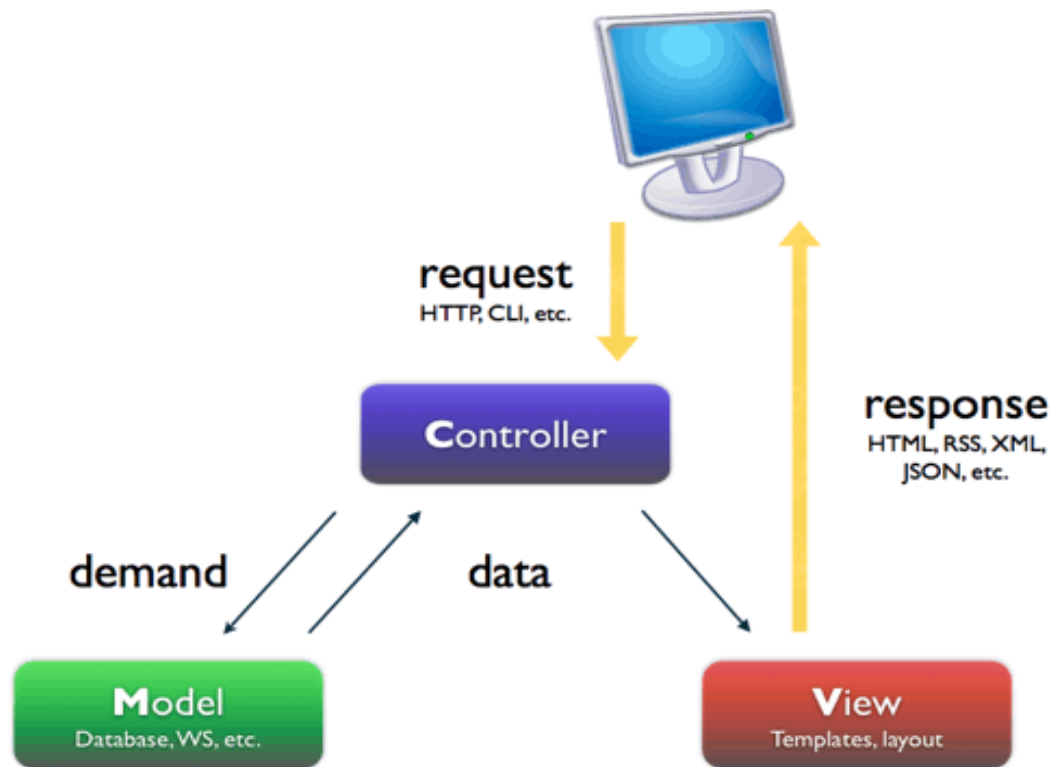
4.1.8 Projeto Arquitetural

O projeto arquitetural é um importantíssimo ponto a ser discutido antes de começar a implementação do código em si. À maneira da maioria dos sistemas orientados a objetos do mundo atual, o aplicativo ElectroSearch utiliza da arquitetura MVC – Model, View, Controller –. “A abordagem MVC é composta por três tipos de objetos. O Modelo é o objeto de aplicação, a Visão é a apresentação na tela e o Controlador é o que define a maneira como a interface do usuário reage às entradas do mesmo. Antes da MVC, os projetos de interface para o usuário tendiam a agrupar esses objetos. A MVC separa esses objetos para aumentar a flexibilidade e a reutilização.” (GAMMA et al., 2000, p. 20).

Uma implementação ideal da arquitetura MVC consiste em uma separação consistente entre os objetos citados anteriormente (Modelo, Visão e Controlador), e prega restringe a comunicação entre alguns deles. Com o bom uso, um objeto Visão não deve se comunicar com um objeto do tipo Modelo. Esta troca de interações entre eles deve ser mediada pelo objeto Controlador.

A imagem acima resume o parágrafo anterior. O objeto Visão (View) é exibido ao usuário por intermédio de um monitor, este usuário por sua vez irá executar alguma funcionalidade do software. Esta requisição, que aconteceu na camada de Visão será passada para a camada do objeto Controlador (Controller); caso a tarefa a ser executada implique em alguma funcionalidade referente ao banco de dados da aplicação, ela será passada ao objeto da camada de Modelo (Model). Após o tratamento feito, a requisição será encaminhada ao Controlador, que passará à Visão e será apresentada ao usuário no monitor.

Imagem 6: Representação ilustrada da arquitetura MVC



Fonte: <https://cms-assets.tutsplus.com/uploads/users/263/posts/21627/image/mvc.png>

A partir da arquitetura MVC é possível fazer uma separação entre os componentes ainda maior, que foi o que se realizou durante o desenvolvimento do software ElectroSearch. Este modelo consiste na criação de mais dois pacotes: Beans e Business. Ambos irão pertencer a camada de Controller.

A camada Beans é responsável por armazenar as classes que pertencem ao software, no caso do ElectroSearch, este pacote é composto por: "Itens", "TAL", e "Professor", que são as classes que participam ativamente do programa.

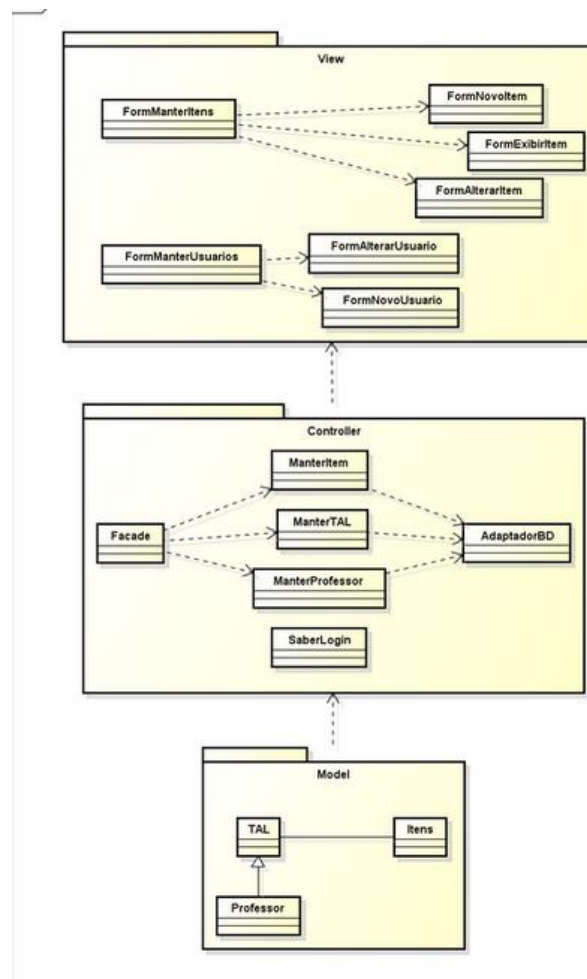
A camada Business, por sua vez, é onde ficarão as regras de negócios referentes ao software. O ElectroSearch tem em seu pacote Business arquivos do tipo: "ManterItem", "ManterTAL", "ManterProfessor", e assim por diante. Dentro de cada um destes há métodos que tratam informações referentes aos objetos das classes que estão no pacote Beans.

O ElectroSearch corresponde exatamente à Imagem 6. Quando um usuário autenticado executa a ação de cadastrar um novo item, por exemplo, uma janela com um formulário é exibida no monitor. Após preencher todas as entradas, o TAL ou Professor irá clicar em salvar e o valor de cada campo será passado para a

camada de Controller. Será instanciado um objeto do tipo ManterItem que irá inicializar o método adicionarItem. Este, por sua vez, cumprirá seu papel, ou seja, intermediará a comunicação entre Visão e Modelo chamando um método para inserir os dados no banco de dados do sistema.

Após esse processo, o objeto da camada Modelo enviará uma resposta para o Controller que pode ser positiva, caso o cadastro tenha sido realizado com êxito, ou negativa. O objeto do Controlador receberá esta informação e encaminhará para a camada de Visão, que irá, por sua vez, exibi-la no monitor para o usuário.

Imagem 7: Projeto Arquitetural do sistema



4.1.9 Padrões de Projeto

Um padrão de projeto representa o cerne de uma solução, de modo que seja possível utilizar tal solução mais de um milhão de vezes, sem nunca fazê-la da

mesma maneira; parafraseando a famosa citação de Christopher Alexander sobre Engenharia Civil, é possível compreender o significado de “padrões de projeto” [REF]. Consiste numa maneira testada e documentada de alcançar um objetivo qualquer, ou seja descreve um repertório de soluções e princípios que ajudam a resolver um problema.

Dentre os padrões de projeto utilizados na implementação do sistema ElectroSearch, destacam-se o Facade, Iterator, Singleton e Observer. “O padrão Facade consiste na criação de uma interface unificada para um conjunto de interfaces em um subsistema. Em outras palavras, define uma interface de nível mais alto que torna o subsistema mais fácil de ser usado (GAMMA et al., 2000, p. 179). Desse modo, minimiza-se a comunicação e a dependência entre subsistemas. No sistema ElectroSearch, o padrão Facade aparece em conjunto com o Singleton, ou seja, há a garantia de que a classe Facade tenha somente uma instância e que a mesma, forneça um ponto global de acesso para si própria.

Um exemplo do citado anteriormente encontra-se a seguir:

Imagem 8: Print da implementação dos padrões Singleton e Facade no sistema ElectroSearch

```
public final class Facade {
    private static final Facade facade = new Facade();

    private Facade() {}

    public static Facade getInstance() {
        return facade;
    }

    // Chamadas das funções referentes à classe TAL

    public List<TAL> pesquisarTAL(String nome, int esc) throws SQLException {
        return ManterTAL.pesquisarTAL(nome, esc);
    }

    public TAL procurarUsuario(String nome, String senha) throws SQLException {
        return ManterTAL.procurarUsuario(nome, senha);
    }

    public boolean AdicionarTAL (String nome, String cpf, String matricula, String disciplina, String login, String senha) throws SQLException {
        return ManterTAL.adicionarTAL(nome, cpf, matricula, disciplina, login, senha);
    }

    public boolean excluirTAL(String matricula) throws SQLException {
        return ManterTAL.excluirTAL(matricula);
    }

    public boolean atualizarTAL(boolean passwd, String nome, String disci, String senha, String cpf, String login, String mat, String novSenha) {
        return ManterTAL.atualizarTAL(passwd, nome, disci, senha, cpf, login, mat, novSenha);
    }

    public static List<TAL> selecionarTudoTAL () {
        return ManterTAL.selecionarTudo();
    }

    public TAL searchTAL (String matricula) {
        return ManterTAL.searchTAL(matricula);
    }
}
```

Fonte: Elaborado pelos autores

O padrão Observer define “uma dependência um para muitos entre objetos, de maneira que quando um objeto muda de estado, todos os seus dependentes são notificados e atualizados automaticamente.” (GAMMA et al., 2000, p. 274).

Tendo isso em consideração, é notório que um exemplo bastante utilizado do padrão Observer é a interface listener do Java. “Cada interface listener de eventos especifica um ou mais métodos de tratamento de evento que devem ser declarados na classe que implementa a interface. [...] Quando o evento ocorre, o componente GUI com o qual o usuário interagiu notifica seus ouvintes registrados chamando o método de tratamento de evento apropriado de cada ouvinte.” (DEITEL, DEITEL, 2010, p. 434).

Um exemplo de Listener – e consequentemente de Observer – foi o utilizado durante o cadastro de componentes. Tal qual no formulário a seguir:

Imagem 9: Print da tela do sistema onde o padrão observer foi implementado



A imagem mostra uma interface web para o cadastro de itens. No topo, há um título "Cadastro de Itens". Abaixo dele, há campos para "Nome:" (um campo de texto), "Tipo:" (um menu suspenso com "-----" selecionado) e "Outro:" (um campo de texto). Segue-se a opção "Disponível:" com dois botões de rádio, "Sim" e "Não". Abaixo disso, há um campo "Datasheet:" e um campo "Descrição:" (uma área de texto grande). Na base da interface, há três botões: "Adicionar", "Limpar" e "Voltar".

Fonte: Elaborado pelos autores

Imagem 10: Print do código referente a implementação do padrão observer

```
public FormNovoItem() {
    initComponents();
    ButtonGroup group = new ButtonGroup();
    group.add(btRSim);
    group.add(btRNao);

    Vector<String> ntipos = listarComboBox();
    for (int i = 0; i < ntipos.size(); i++) {
        jcTipo.addItem(ntipos.get(i));
    }

    ActionListener listener;
    listener = new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            if ("Outro".equals((String) jcTipo.getSelectedItem())) {
                cmpOutroTipo.setEnabled(true);
            } else {
                cmpOutroTipo.setEnabled(false);
            }
        }
    };

    jcTipo.addActionListener(listener);
}
```

Fonte: Elaborado pelos autores

O padrão Iterator fornece “um meio de acessar, sequencialmente, os elementos de um objeto agregado sem expor a sua representação subjacente.” (GAMMA et al., 2000, p. 244). Tal qual o Observer, o Iterator também tem uma representação bastante utilizada no Java.

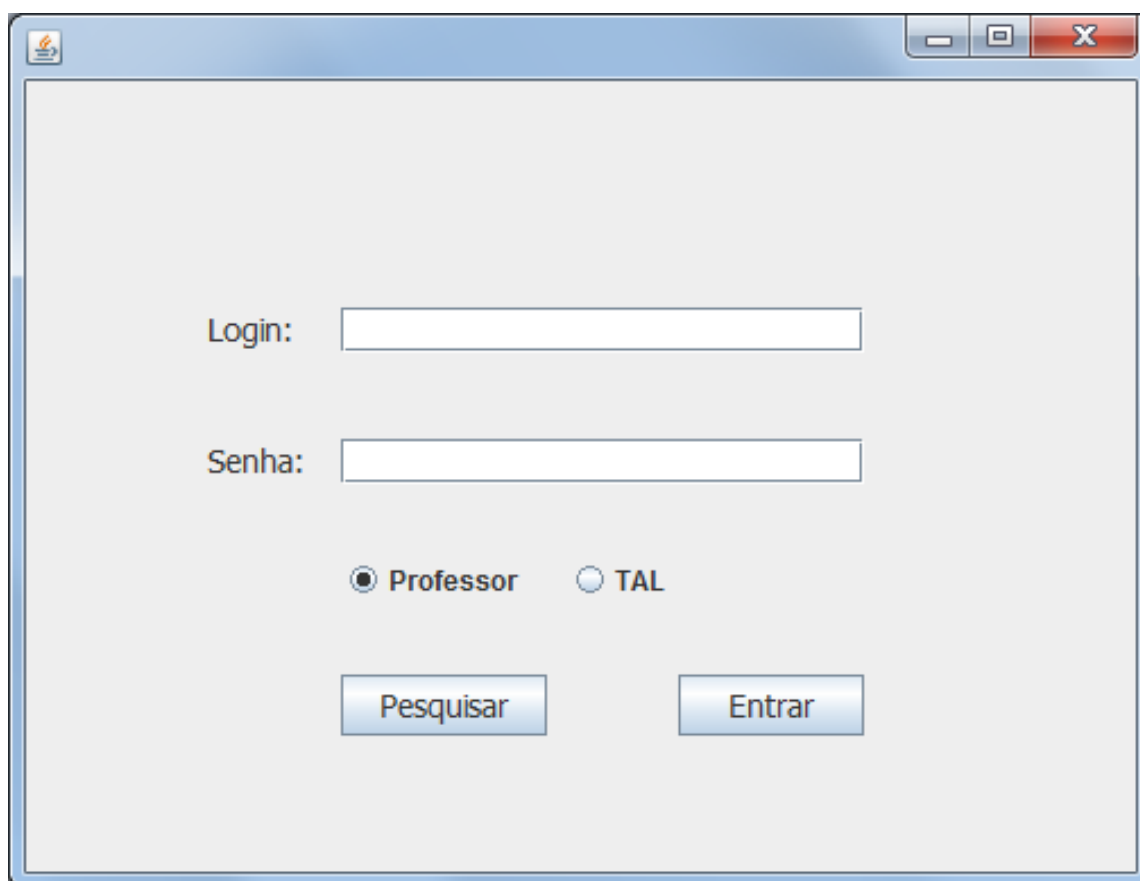
Neste software, o Iterator age junto com o Listener a fim de diminuir, tratar, ou até evitar erros provenientes de descuidos que alguns usuários poderiam vir a cometer, evitando também a redundância de alguns dados e disponibilizando o cadastro de novos tipos de componentes.

5 MANUAL DO USUÁRIO

Esta seção tem como por objetivo expor um manual direcionado aos tipos de usuários que compõe o sistema ElectroSearch, estes são: Aluno, TAL e Professor. Cada um com diferentes privilégios e funções neste cenário. A abordagem será feita de modo que comece pelo menos privilegiado, no caso o Aluno e vá até o usuário do tipo Professor, que é o mais privilegiado.

A primeira interface do programa (Imagem 11) corresponde a área de Login do sistema, ela é comum a todos os tipos de usuários, entretanto, somente TAL e Professor possuem permissão para a autenticação.

Imagem 11: Print da tela de login do sistema

A imagem mostra uma janela de login de um sistema. No topo, há uma barra de título com ícones de minimizar, maximizar e fechar. O corpo da janela contém dois campos de entrada: "Login:" e "Senha:". Abaixo dos campos, há duas opções de seleção: "Professor" (selecionada com um botão de rádio) e "TAL" (não selecionada). Na base da janela, há dois botões: "Pesquisar" e "Entrar".

Fonte: Elaborado pelos autores

Caso o usuário seja um Aluno, ele não poderá fazer autenticação, restando-lhe apenas a opção de pesquisar um componente. Caso contrário, o TAL ou Professor deverá inserir seu login e sua senha nos respectivos campos da Imagem

11, e selecionar o tipo de usuário que ele é, estas opção são dadas logo abaixo do campo “Senha”.

Se o usuário se autenticar no sistema como TAL, ele será direcionado automaticamente para uma tela de gerenciamento de componentes, mostrada na Imagem 12, que é a mesma quando o usuário não autenticado clica no botão “Pesquisar”.

Imagem 12: Print da tela responsável por manter itens

Lista de componentes

Pesquisar

☐ ID ☐ Nome ☐ Tipo ☐ Aleatório

ID	Nome	Tipo	Disponível
1	BC548	Transistor	Sim

Novo **Alterar** **Apagar** **Exibir Tudo** **Voltar**

Fonte: Elaborado pelos autores

A diferença desta tela para usuário autenticado e o não autenticado consiste na presença dos três botões: “Novo”, “Alterar” e “Apagar” que estão presentes para o primeiro caso.

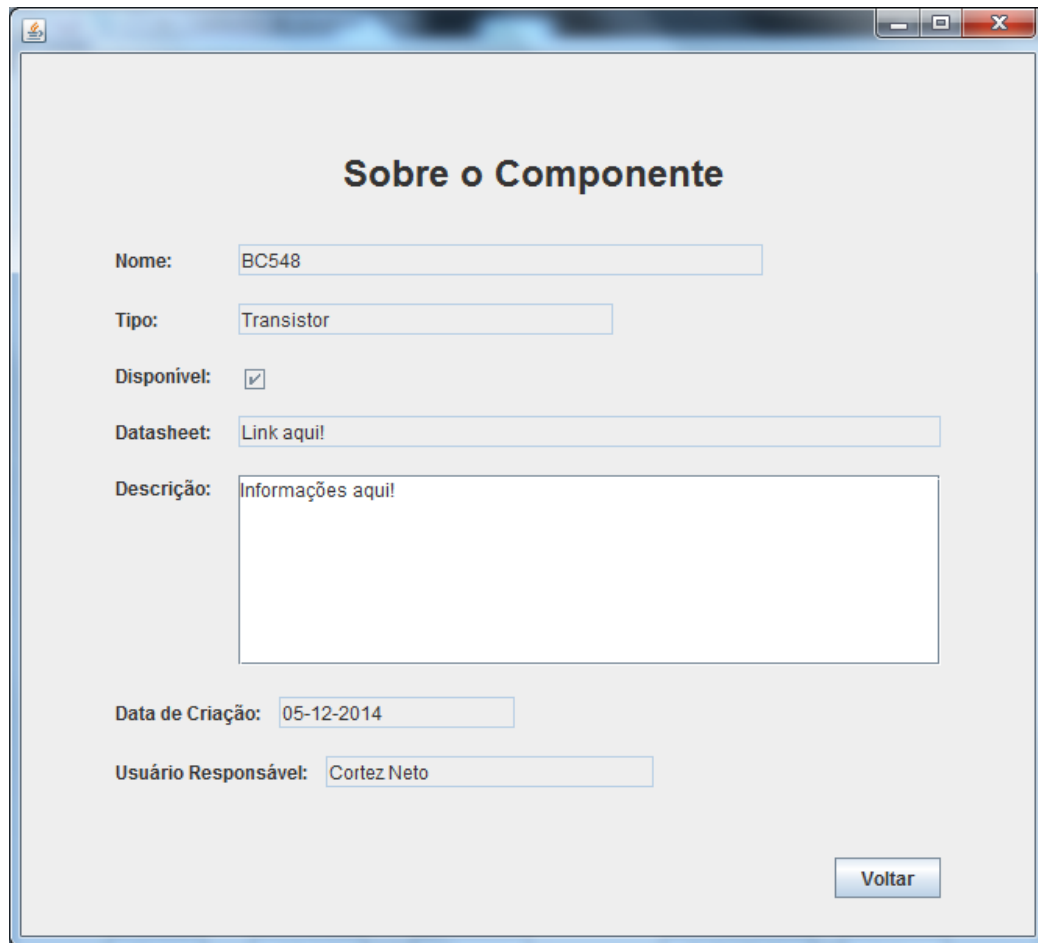
Tratando de um caso mais genérico, quando a autenticação, isto é, o login não é realizado a única funcionalidade presente nesta interface é a de pesquisar componentes. Com isto, o usuário poderá optar entre quatro tipos de pesquisa: por ID, por Nome, por Tipo, ou por um campo aleatório.

A busca através de um ID precisa de uma entrada que consiste em um número inteiro e irá retornar somente uma linha de resultado. A pesquisa por nome também irá ter esta mesma quantia de resultado, entretanto, sua entrada deverá ser um conjunto de caracteres, isto é, uma palavra, geralmente é a referência do componente a ser buscado, por exemplo: BC548, 1N4007.

As buscas realizadas pelo tipo e de forma aleatória retornarão mais de uma linha de resultado. O primeiro caso terá uma cadeia de caracteres como entrada, e o segundo poderá ter qualquer tipo de entrada, isto é, desde um número até uma palavra ou texto.

Ao clicar na linha do componente escolhido aparecerá uma janela informando as informações deste componente, como segue na Imagem 13. Os dados mostrados são os de cadastro, como: nome, tipo, disponibilidade, datasheet (que será um link para visualizar este documento na internet) e a descrição que poderá ter os dados mais importantes do datasheet, isto é, de forma mais enxuta, além de quem cadastrou e a data.

Imagem 13: Print da tela que exibe a pesquisa



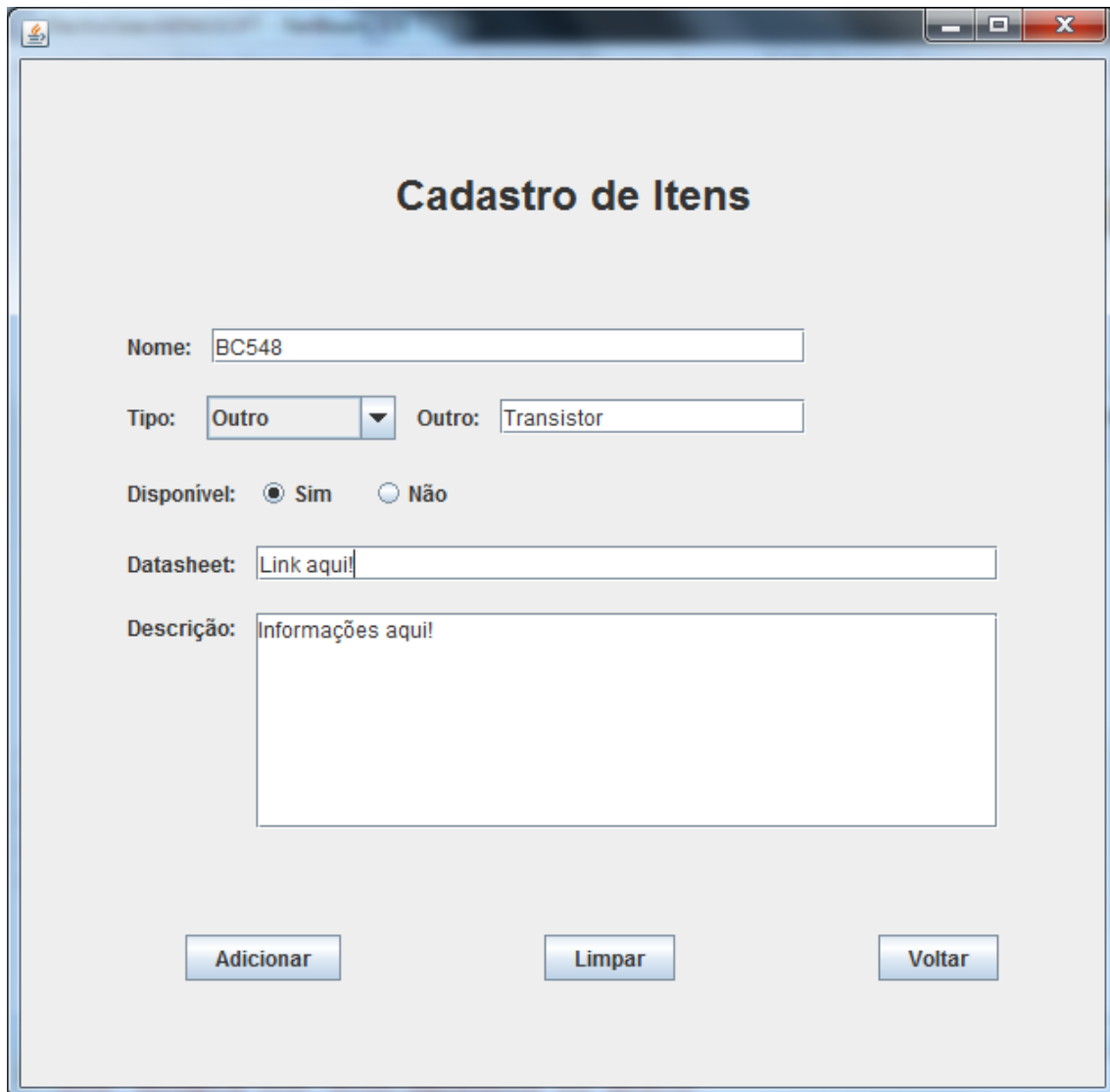
The screenshot shows a web application window with a title bar containing standard minimize, maximize, and close buttons. The main content area has a light gray background and is titled "Sobre o Componente" in a bold, black font. Below the title, there is a form with several fields:

- Nome:** A text input field containing "BC548".
- Tipo:** A text input field containing "Transistor".
- Disponível:** A checkbox that is checked, indicated by a small square with a checkmark.
- Datasheet:** A text input field containing "Link aqui!".
- Descrição:** A large text area containing "Informações aqui!".
- Data de Criação:** A text input field containing "05-12-2014".
- Usuário Responsável:** A text input field containing "Cortez Neto".

In the bottom right corner of the form area, there is a button labeled "Voltar".

Fonte: Elaborado pelos autores

De agora em diante só restam funcionalidades restritas a usuários autenticados, TAL ou professor, estas se resumem em: cadastro, atualização e exclusão de um componente ou de um usuário.

Imagem 14: Print da tela de cadastro de itens

A imagem mostra uma janela de software com o título "Cadastro de Itens". O formulário contém os seguintes campos:

- Nome:** Um campo de texto com o valor "BC548".
- Tipo:** Um menu suspenso com o valor "Outro" selecionado.
- Outro:** Um campo de texto com o valor "Transistor".
- Disponível:** Dois botões de opção: "Sim" (selecionado) e "Não".
- Datasheet:** Um campo de texto com o valor "Link aqui!".
- Descrição:** Um campo de texto com o valor "Informações aqui!".

Na base da janela, há três botões: "Adicionar", "Limpar" e "Voltar".

Fonte: Elaborado pelos autores

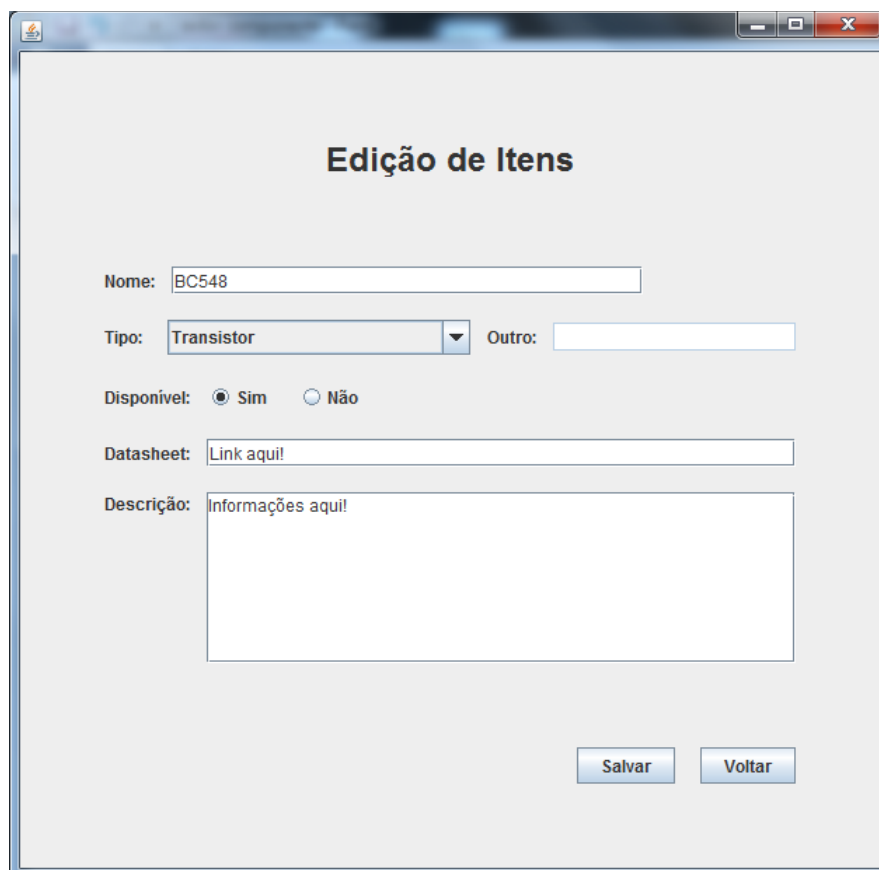
A janela da imagem 14 é referente ao cadastro de componentes, de forma bem simples e com campos autoexplicativos, o usuário não terá grandes dificuldades para executar esta funcionalidade. O campo "Nome" deve ter a referência de um componente, no caso da imagem temos o BC548, uma referência única, não há nenhum outro componente com este nome e com propriedades diferentes. O tipo de um componente é acompanhado por um menu que irá listar todos os tipos de componentes cadastros anteriormente, ou seja, na próxima situação de cadastro o usuário poderá escolher o tipo "Transistor" diretamente no menu. Em seguida será informado se tal item está disponível ou não através da seleção de um "Radio Button". O campo "Datasheet" deverá ter um link que

redirecione o usuário para uma janela na internet que exiba este documento do item a respeito, e a descrição é onde poderá ser adicionado as informações mais importantes do datasheet, isto é, tensão máxima, corrente máxima e demais dados.

Para a exclusão de um componente ser realizada basta a linha do mesmo ser selecionada e o usuário autenticado clicar no botão “Excluir”, após isto uma pop-up deverá aparecer perguntando se o usuário realmente deseja excluir tal item, ao clicar em “Sim” o componente será devidamente excluído, caso contrário, ele permanecerá no sistema.

A imagem 15 exibe a janela quando uma operação de atualização de um determinado componente é realizada, para ela ser gerada a linha do componente desejado deverá ser selecionada e o usuário terá de clicar no botão “Alterar”, após isto, isto será exibido:

Imagem 15: Tela de edição de itens



Edição de Itens

Nome:

Tipo: Outro:

Disponível: ☒ Sim ☐ Não

Datasheet:

Descrição:

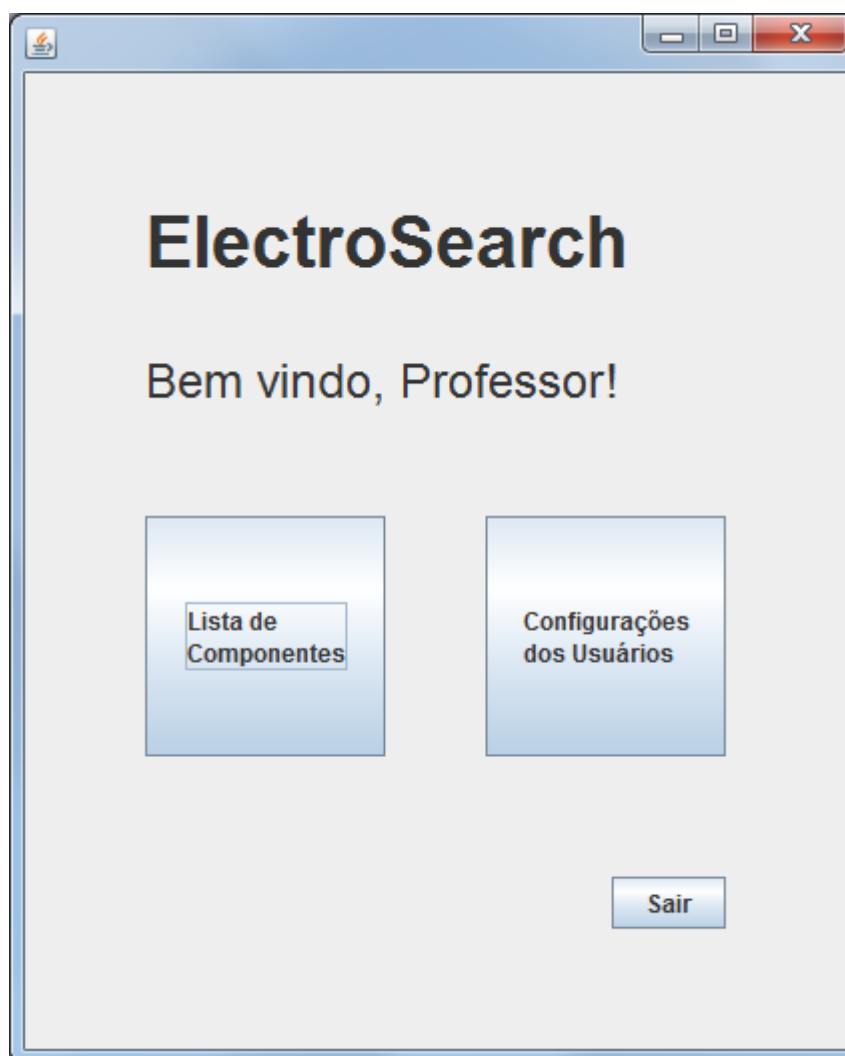
Fonte: Elaborado pelos autores

Serão puxados todos os dados do componente, inclusive o tipo que fora cadastrado anteriormente. Após realizar as alterações desejadas o usuário clicará

em “Salvar” e uma pop-up será exibida informando que a operação foi realizada com sucesso.

Quando um usuário do tipo Professor realiza a autenticação no sistema, ele irá ter receber uma página de boas-vindas, o que é desnecessário ao TAL, já que este só tem uma função. O Professor, além de gerenciar as funcionalidades de um componente, poderá também gerenciar as dos usuários.

Imagem 16: Painel inicial do usuário Professor



Fonte: Elaborado pelos autores

Ao clicar no botão “Configurações dos Usuários”, uma tela semelhante à de gerenciar componentes será exibida, mas na tabela serão listados os usuários e alguns de seus dados, como mostra a imagem 17:

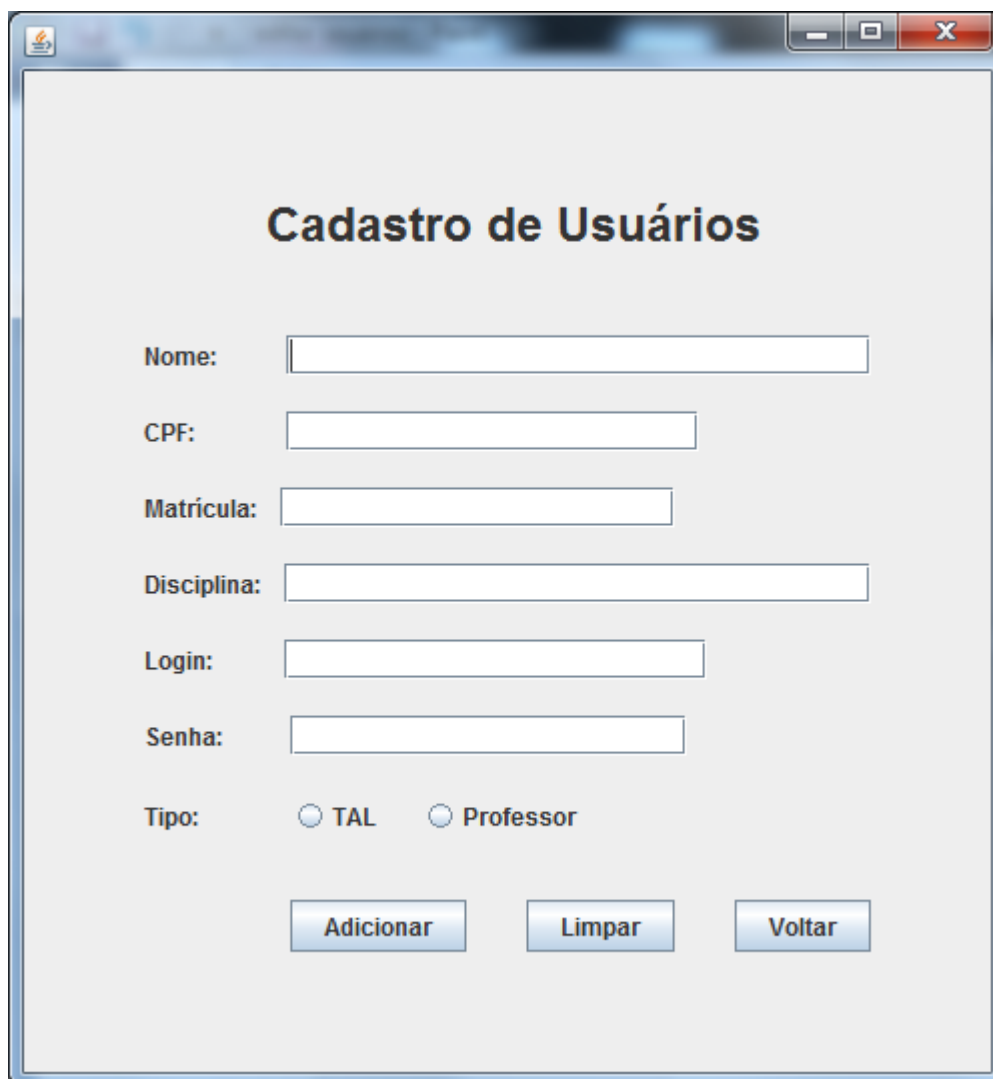
Imagem 17: Tela de configuração de usuários

Matrícula	Nome	CPF	Tipo
0694	Cortez Neto	70000996408	Professor

Fonte: Elaborado pelos autores

As operações aqui também são semelhantes à da parte de gerenciar um componente. O Professor poderá pesquisar um determinado usuário por mais de um parâmetro, as possibilidades são: “Matrícula”, “Nome” e pesquisa por “CPF”.

Para a realização de um novo cadastro de usuário, o administrador (Professor) deverá clicar no botão “Novo” e uma tela igual à da imagem 18 será exibida.

Imagem 18: Tela de cadastro de usuáriosA imagem mostra uma janela de software com o título "Cadastro de Usuários". O formulário contém campos de entrada para "Nome:", "CPF:", "Matricula:", "Disciplina:", "Login:" e "Senha:". Abaixo desses campos, há uma seção "Tipo:" com duas opções de seleção por rádio: "TAL" e "Professor". Na base da janela, há três botões: "Adicionar", "Limpar" e "Voltar". A janela possui uma barra de título com ícones de minimizar, maximizar e fechar.

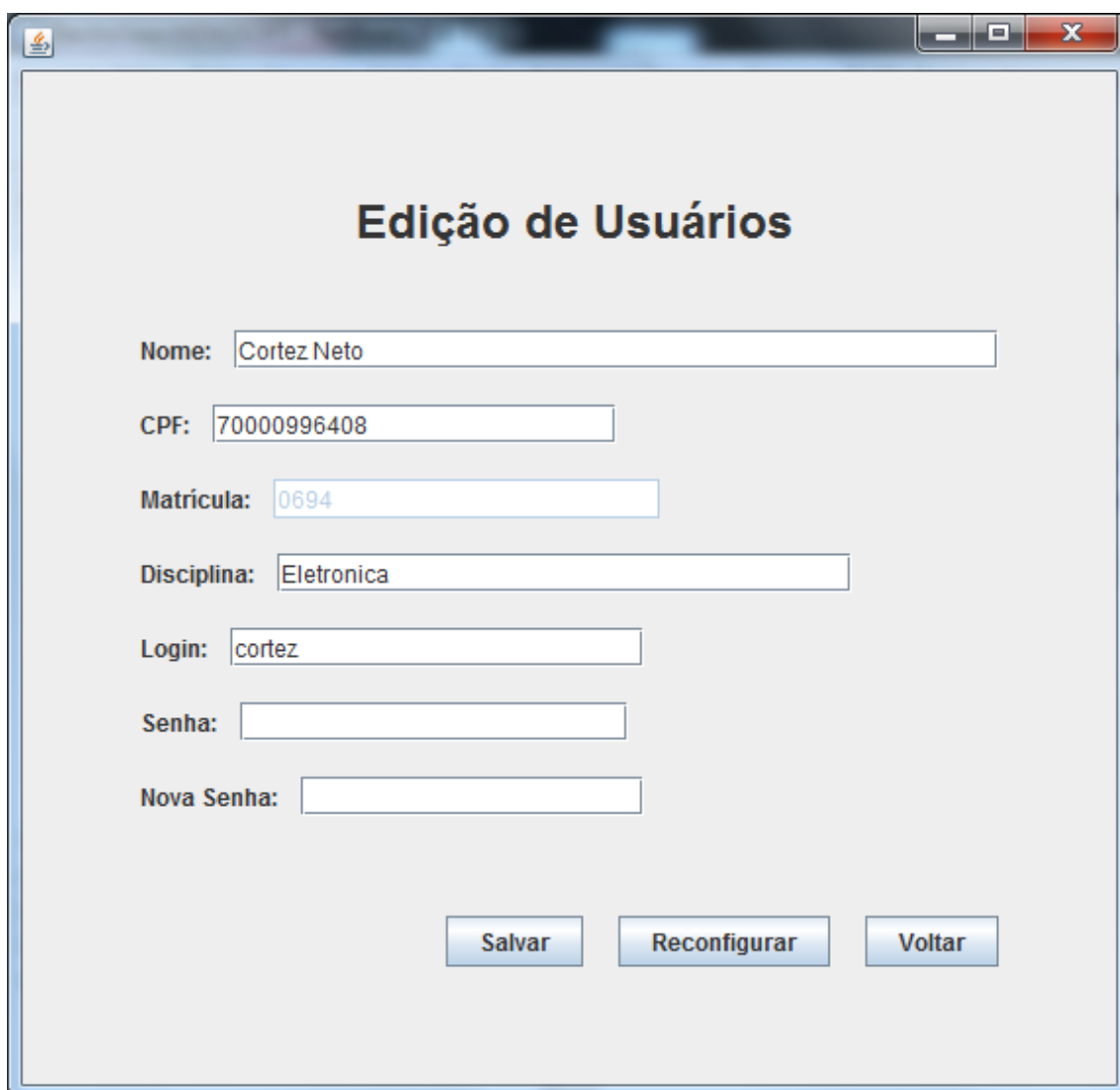
Fonte: Elaborado pelos autores

Todos os campos são autoexplicativos e devem ser preenchidos da forma correta, isto facilitará a administração destes dados em futuras ocasiões que o usuário venha a usar o sistema. Após devidamente preenchidos, o professor responsável pelo cadastro do usuário deverá clicar no botão "Adicionar" caso queira prosseguir com esta operação, ou poderá clicar em "Limpar" para restaurar os campos do cadastro. A opção de "Voltar" o levará a tela de gerenciamento de usuários, onde ele poderá realizar outras funções.

A remoção de um usuário é tão simples quanto a de um componente, ao ser selecionada uma linha da tabela, isto é, a linha que estará o usuário a ser excluído, o administrador só precisará clicar no botão "Excluir" e a operação será realizada com sucesso.

Para o Professor alterar seus dados ou a de outro usuário, ele deverá selecionar a linha desejada da tabela e clicar no botão “Alterar”. Após isto, uma tela com campos preenchidos com as informações do usuário irá aparecer na tela do terminal, como bem mostra a imagem 19. Para esta operação ser realizada com sucesso, o administrador do processo deverá entrar no campo “Senha” com a senha do usuário a ser alterado e preencher todos os campos, com exceção do campo “Nova Senha” caso não seja de desejo mudar a atual. Feito isto, basta clicar no botão “Salvar” para continuar com o processo, caso queira ter de volta os campos com os dados anteriores, o botão “Reconfigurar” deverá ser escolhido.

Imagem 19: Tela de cadastro de usuários



A imagem mostra uma janela de software intitulada "Edição de Usuários". O formulário contém os seguintes campos:

- Nome: Cortez Neto
- CPF: 70000996408
- Matrícula: 0694
- Disciplina: Eletronica
- Login: cortez
- Senha: (campo vazio)
- Nova Senha: (campo vazio)

Na base da janela, há três botões: "Salvar", "Reconfigurar" e "Voltar".

Fonte: Elaborado pelos autores

6 CONCLUSÃO

Este trabalho teve como objetivo desenvolver um software capaz de oferecer o gerenciamento dos componentes referentes ao laboratório de Eletricidade e Eletrônica do Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte e passar todos os processos referentes à engenharia de software utilizada no sistema, desde as ferramentas mais básicas, para o entendimento do desenvolvedor a respeito do programa, como as mais avançadas, que são intrínsecas à programação e o desenvolvimento do software apresentado neste trabalho, à exemplo disto temos os padrões de projetos que são utilizados na implementação do programa e descritos neste documento.

Fica claro que nem somente de códigos se faz um sistema essencial como este, há também toda a regra de negócios e o desenvolvimento e divisão das etapas do processo de construção do objeto principal, que é o software. O objetivo, além de detalhar todo o tempo investido e as principais ferramentas, também é o de compreensão acerca do funcionamento do projeto, facilitando e poupando a vida de quem poderá seguir neste ramo, uma vez que este documento dá uma noção básica do que deve ser feito.

À priori, as maiores dificuldades enfrentadas no desenvolvimento do programa foram relacionadas ao tratamento de erros em entradas de dados improváveis. Além disto, há também a o uso de uma ferramenta relacionada ao ensino da disciplina de Redes de Computadores, onde tal [ferramenta] é fundamental para o uso de uma unidade central de banco de dados, facilitando assim a consulta de itens por parte do usuário.

Uma futura funcionalidade que poderá ser adicionada é referente ao gerenciamento de projetos da própria disciplina. Como no final da cadeira o aluno discente desta é incentivado a desenvolver projetos que possam ser usados para melhorar o seu cotidiano e das pessoas que participam destes, ele poderá efetuar o cadastro desta ideia para o futuro reaproveitamento por parte dele mesmo ou de outro estudante.

O software foi concluído com sucesso, contudo, isto não implica que melhorias ou funcionalidades adicionais não possam vir a ser implementadas.

REFERÊNCIAS BIBLIOGRÁFICAS

BOOCH, Grady; RUMBAUGH, James; JACOBSON, Ivar. **UML: guia do usuário**. 9. ed. Rio de Janeiro: Elsevier, 2005.

DEITEL, Paul; DEITEL, Harvey. **Java: como programar**. 8. ed. São Paulo: Pearson Prentice Hall, 2010.

FOWLER, Martin. **UML essencial: um breve guia para a linguagem-padrão de modelagem de objetos**. 3. ed. Porto Alegre: Bookman, 2005.

GAMMA, Erich et al. **Padrões de Projeto: soluções reutilizáveis de software orientado a objetos**. Porto Alegre: Bookman, 2000.

GOSLING. **Java Platforms**. EUA, 1998. Disponível em:
<<http://java.sun.com/docs/white/langenv>> Acesso em: 23 nov. 2014

HORSTMANN, Cay; CORNELL, Gary. **Core Java**. São Paulo: Pearson Prentice Hall, 2010.

LARMAN, Craig. **Utilizando UML e Padrões: uma introdução à análise e ao projeto orientado a objetos e ao desenvolvimento iterativo**. 3. ed. Porto Alegre: Bookman, 2007.