UNIVERSIDADE FEDERAL DO RIO DE JANEIRO

ESCOLA DE ENGENHARIA

DEPARTAMENTO DE ELETRÔNICA E DE COMPUTAÇÃO

MÓDULO WEB PARA GERENCIAMENTO DO LEG (LEGWeb)

Autor: _____

José Carlos Paula Correia

Orientador: _____

Marcelo Luiz Drumond Lanza, M.Sc.

Examinador: _____

Ricardo Rhomberg Martins, D.Sc.

Examinador: _____

Gelson Vieira Mendonça, Ph.D.

DEL

Dezembro de 2006

Dedicatória

Aos meus pais, José de Almeida Correia (*in memorian*) e Geralda Paula de Jesus, a quem devo a vida e minha formação moral. Meu reconhecimento e gratidão pela paciência, compreensão e apoio constante nesta jornada da vida.

Agradecimentos

Aos professores da Universidade Federal do Rio de Janeiro com quem tive o privilégio de aprender, ao longo dos anos de graduação, não só uma formação meramente técnica, mas uma formação com valores morais que me acompanharão pelo resto da vida.

Aos funcionários do Departamento de Engenharia Eletrônica e Computação que ofereceram todo o suporte necessário para a conclusão deste trabalho.

Ao **professor Marcelo Luiz Drumond Lanza**, por sua paciência, bom humor e presença constante ao meu lado durante todas as fases deste projeto e de outros durante a graduação. Da mesma forma, por me apresentar a oportunidade de desenvolver um trabalho que tenha uma aplicação marcante na vida de alunos e professores do Departamento de Engenharia Eletrônica e Computação.

Aos avaliadores deste projeto, **professor Ricardo Rhomberg Martins** e **professor Gelson Vieira de Mendonça**, por serem dignos representantes de um grupo que consegue, com muito esforço e dedicação, prover educação pública de qualidade neste país, honrando de maneira especial e única, por conseguinte, todo trabalho que contenha seus nomes e suas assinaturas.

Resumo

O módulo LEGWeb (*Módulo de Gerenciamento do LEG*) permite a integração de serviços e informações do Laboratório de Engenharia da Graduação (LEG) do Departamento de Engenharia Eletrônica e Computação (DEL) da Universidade Federal do Rio de Janeiro (UFRJ) com o SID (*Sistema Integrado de Serviços e Informações do DEL/UFRJ*) [1], sistema desenvolvido em abril de 2002 pelo aluno Pedro da Fonseca Vieira.

O SID, de acordo a com sua descrição, previu futuras expansões através da inclusão de módulos complementares às suas funcionalidades [1]. O LEGWeb é o primeiro módulo complementar ao SID e faz uso do banco de dados deste sistema. Do SID, são utilizados o cadastro dos usuários da rede DEL (professores, funcionários, alunos e ex-alunos, contratados, etc), bem como de toda a sua estrutura de autenticação segura [1].

Este módulo destina-se a automatizar os processos de atendimento e manutenção do LEG, de forma a melhorar a qualidade do serviço prestado aos seus usuários. Seus serviços incluem diversas funcionalidades de uma aplicação de *Internet*, permitindo que alunos, professores e administradores do módulo possam, entre outras atividades, gerenciar remotamente o uso de componentes e equipamentos (realizar empréstimos, solicitar reparos, solicitar aquisição de insumos para as atividades), gerenciar as contas de usuários, gerenciar as notas das disciplinas de laboratório (consultar, cadastrar, alterar), consultar os horários de aula, consultar as informações do laboratório, etc. O nível de acesso e conseqüentemente das funções serão dados pelo perfil do usuário em questão.

Abstract

LegWeb (*Módulo de Gerenciamento do LEG*) allows the integration of service and information from *Laboratório de Engenharia da Graduação* (LEG) to *Departamento de Engenharia Eletrônica e Computação* (DEL), which belongs to *Universidade Federal do Rio de Janeiro* (UFRJ), by integrating their services with SID (*Sistema Integrado de Serviços e Informações do DEL / UFRJ*) [1], developed by an undergraduated student, Pedro da Fonseca Vieira, in April 2002.

According to its original description, SID predicted future expansions by the inclusion of complementary modules to its functionalities [1]. LegWeb is the first complementary structure and uses SID´s databases in order to check user's information and authenticate them as well [1].

The module was developed in order to automate the operations at Leg, improving the service provided to their users. Functionality includes Internet access, which allows students, teachers and administrators to remotelly manage equipments and eletronic components (users can borrow, ask for repair and for new material for their activities), manage user´s accounts, manage student´s grades, consult schedule and all sort of information. The access level and their functions depends on the profile given to the user.

Palavras-chave

LEGWeb

LEG

DEL

Gerenciamento

SID [1]

PHP

MySQL

UML

Rede de Computadores

Índice Analítico

Índice de Figuras

Capítulo I . Introdução

I.1. Motivação

A Internet atrai cada vez mais usuários, e a necessidade de interagir com rapidez, simplicidade e mobilidade demanda cada vez mais o desenvolvimento de aplicações para este canal ou a adaptação de sistemas antes desenvolvidos no modelo de duas camadas (cliente/servidor) para o modelo Web. A vida digital está cada vez mais presente na vida dos cidadãos, o que reforça ainda mais a necessidade de desenvolvimento de sistemas para este ambiente.

Os sites e sistemas vão de simples páginas estáticas a completas soluções de comercio eletrônico, sistema de publicação de noticias, controle remoto do site através de qualquer maquina, desenvolvimento de soluções corporativas, cadastro de clientes, controle de estoque, controle de cargas e soluções nos mais diversos segmentos de mercado.

Nesse sentido, identificou-se a oportunidade de desenvolver um sistema que integrasse todas as necessidades e possibilidades acima descritas em favor dos alunos do Laboratório de Eletrônica da Graduação – LEG. A partir deste sistema, os alunos teriam um novo canal de relacionamento com laboratório, facilitando o seu dia a dia, bem como permitindo aos administradores do LEG ter acesso a um portal com informações centralizadas sobre a gestão do ambiente.

I.2. Histórico

O Laboratório de Eletrônica da Graduação – LEG - atende em suas salas a dezenas de alunos que utilizam a sua estrutura para cursar as disciplinas ministradas pelo Departamento de Eletrônica e de Computação – DEL. Nele, os alunos têm a possibilidade de alocar ativos do estoque do laboratório, tais como componentes eletrônicos, instrumentos e outros, de forma a realizar as tarefas propostas por cada disciplina.

Para atender à demanda acima exposta, o laboratório necessita de procedimentos eficazes e ajustados que permitam melhorar o atendimento ao público alvo. O controle dos

processos, quando feito manualmente, faz com que se perca em eficácia e gera desperdício de dinheiro.

Para melhorar as condições de administração, propôs-se a confecção de uma ferramenta baseada em Internet, que agirá como um módulo do Sistema Integrado de Serviços e Informações do DEL/UFRJ – SID [1] – de onde o usuário poderá ter controle total das atividades realizadas no LEG. Esta ferramenta substituirá um aplicativo utilizado atualmente pelo laboratório desenvolvido sobre a plataforma MS-DOS (*Microsoft Disk Operating System*). Esta aplicação, além de desenvolvida para uma plataforma defasada em aproximadamente vinte anos, possui limitações de navegação (não é amigável ao usuário), de acesso (não permite acesso remoto por alunos para consultas) e não permite o armazenamento de dados históricos sobre os equipamentos gerenciados pelo laboratório.

A migração para a plataforma Web traz diversas vantagens para o uso e manutenção da ferramenta. Algumas dessas vantagens estão explicadas e exemplificadas abaixo.

I.2.1. Uniformidade

Documentos criados segundo os padrões Web podem utilizar uma estrutura comum, facilitando a manipulações dos mesmos. Uma estrutura comum permite que modificações tais como inserções e remoções de conteúdo ou movimentações estruturais podem ser realizadas de maneira simples, sem a necessidade de aplicações complexas. A uniformidade permite que documentos possam ser manipulados através de um conjunto reduzido de aplicações, transformações e mecanismos de apresentação.

Para um exemplo específico, um site contendo vários documentos pode fazer uso de uma única folha de estilo para formatar os mesmos para exibição. Isso só é possível se os documentos forem o mais uniformes possível. Um outro exemplo é quando os documentos precisam ser transformados para uso em outro ambiente. A uniformidade permite que isso seja feito em um número reduzido de passos comuns.

I.2.2. Liberdade

Os padrões Web permitem também a liberdade de estruturação e inovação por não serem controlados por uma empresa específica. Isso permite que sejam utilizados por qualquer pessoa em qualquer lugar, sem a necessidade de pagar ou fazer algo pelo privilégio. Essa liberdade permite também uma maior facilidade na movimentação de informações e

evita que as mesmas se tornem obsoletas. Um documento criado através dos padrões Web, por causa da própria estrutura destes, estará sempre aberto à movimentação na direção de outros padrões e sistemas futuros. Formatos proprietários (especialmente os binários) tornam as pessoas e empresas dependentes das ferramentas que os manipulam e as prendem a essas ferramentas e formatos.

I.2.3. Separação de estrutura e apresentação

Essa talvez seja a maior vantagem na utilização dos padrões. A utilização correta dos mesmos permite separar quase que completamente a estrutura da apresentação. Isso significa que o documento fica restrito ao seu conteúdo, sem especificar qualquer forma de apresentação, permitindo que esta seja modificada de acordo com as necessidades. Assim, o documento permanece o mesmo, embora possa ser usado em diferentes ambientes como navegadores, sintetizadores de fala, e geradores de documentos Braille. A correta separação da estrutura da apresentação permite uma maior flexibilidade na utilização do documento.

Por exemplo, um determinado documento pode ser exibido de uma certa forma em um navegador e modificado significativamente para a impressão e execução de outras transformações que permitam que o documento seja mais facilmente utilizado em uma situação diferente.

I.2.4. Facilidade de criação

O uso de padrões também torna mais fácil a criação dos documentos já que não é necessário preocupar-se inicialmente com a apresentação dos mesmos, livrando o criador do documento para pensar apenas no conteúdo do mesmo. Como dito anteriormente, quase qualquer tipo de layout pode ser criado e isso permite que essa etapa da construção de sites seja realizada independentemente do desenvolvimento de conteúdo. O criador fica, também, livre do peso do uso de editores específicos. O conteúdo pode ser editado em qualquer processador de textos e posteriormente estruturado. Uma vez estruturado ele poderá ser exibido através da formatação apropriada anexada ao mesmo.

I.2.5. Estabilidade

A estabilidade significa que os documentos feitos para os padrões Web correntes permanecerão compatíveis com qualquer nova tecnologia que venha ser desenvolvida. Isso

quer dizer que tais documentos serão capazes de produzir um visual aceitável e completo acesso ao seu conteúdo em ambientes com suporte limitado a novos padrões. Um exemplo disso são sites que possuem uma rica exibição em navegadores visuais mais modernos, mas podem ser completamente acessados em navegadores de modo texto, como o Lynx, ou navegadores com suporte fraco aos padrões, como o Netscape 4. Sendo também direcionada ao futuro, a estabilidade permite que documentos existentes atualmente permaneçam passíveis de utilização muitos anos no futuro, em ambientes ainda a serem criados.

Outro ganho da estabilidade é na manutenção de documentos. Muitas vezes um determinado site pode passar por várias equipes durante sua vida útil. A estabilidade dos documentos através dos padrões Web permite que um site possa ser compreendido e editado por qualquer grupo sem necessidade de esforços excessivos.

I.3. Objetivos

Aproveitando o fato de o Sistema Integrado de Serviços e Informações do DEL/UFRJ – SID [1], desde sua concepção, permitir expansões de suas funcionalidades pela inclusão de módulos complementares, objetiva-se desenvolver um módulo de gerenciamento das rotinas do Laboratório de Eletrônica da Graduação, denominado LEGWeb, que se integre ao mesmo.

Este módulo automatizará os processos de atendimento e manutenção do LEG, criando novos canais de relacionamento com o mesmo tanto para alunos como para professores e administradores do laboratório.

Dentre outras atividades, os usuários serão capazes de gerenciar remotamente o uso de componentes e equipamentos (realizar empréstimos, solicitar reparos e aquisição de insumos para atividades), gerenciar as contas de usuários, gerenciar as notas das disciplinas de laboratório (consultar, cadastrar e alterar), consultar os horários de aula, consultar informações de laboratório, etc.

A figura I.1 ilustra, de forma introdutória, como se dá o relacionamento do módulo com seu sistema base (SID) [1].

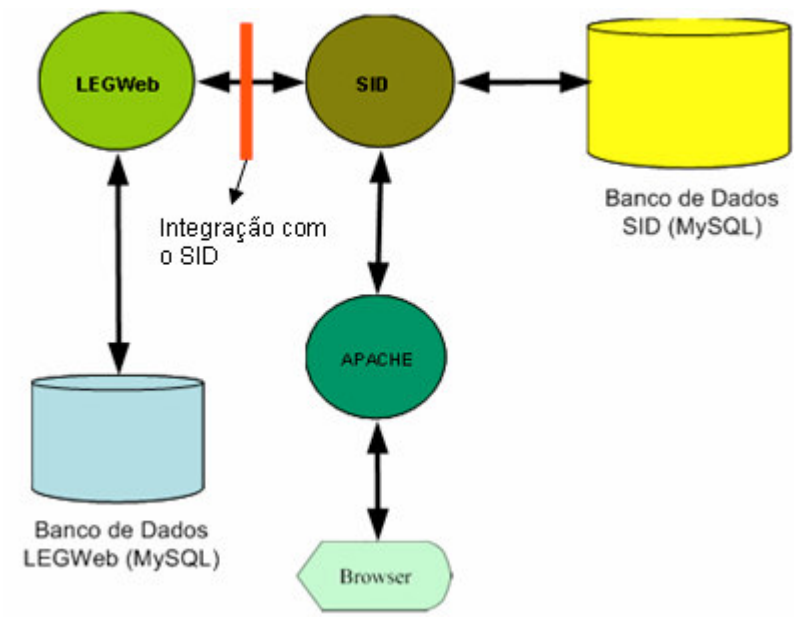


Figura I.1 – Visão geral do LegWeb

O usuário acessa, através de seu *browser* (navegador), a página inicial do SID [1], publicada através de um servidor de publicação WEB utilizando o *software* Apache. Nesta página, ele será autenticado e, de acordo com seu perfil (cadastrado no SID)[1], lhe é oferecida a opção de acesso ao módulo do LEGWeb através de um atalho (*link*).

As figuras I.2 e I.3 apresentam o modelo conceitual de acesso ao módulo, permitindo a conexão tanto internamente (*Intranet*) como externamente (*Extranet*).

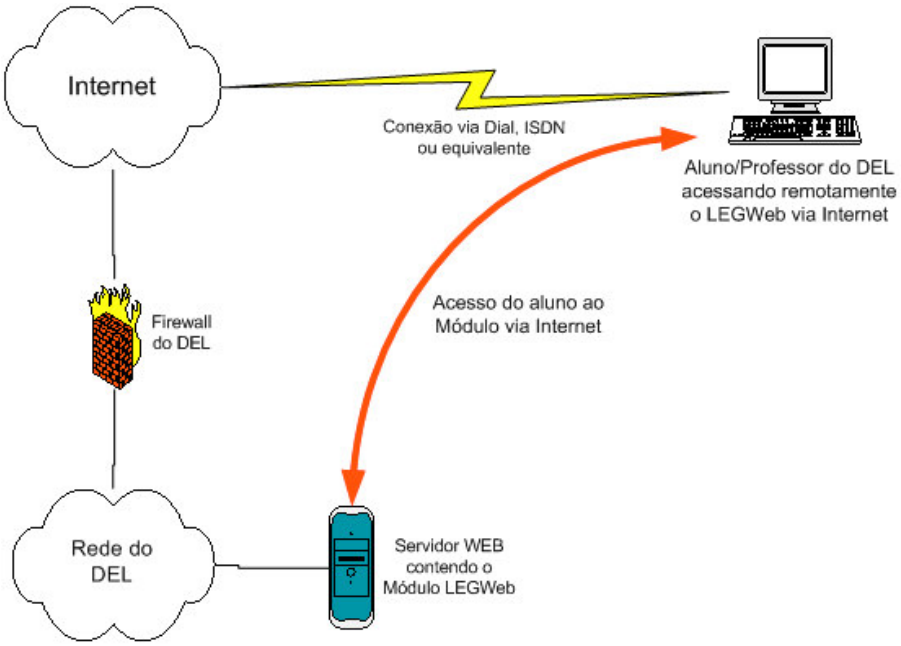


Figura I.2 - Modelo de comunicação através da *Internet.*

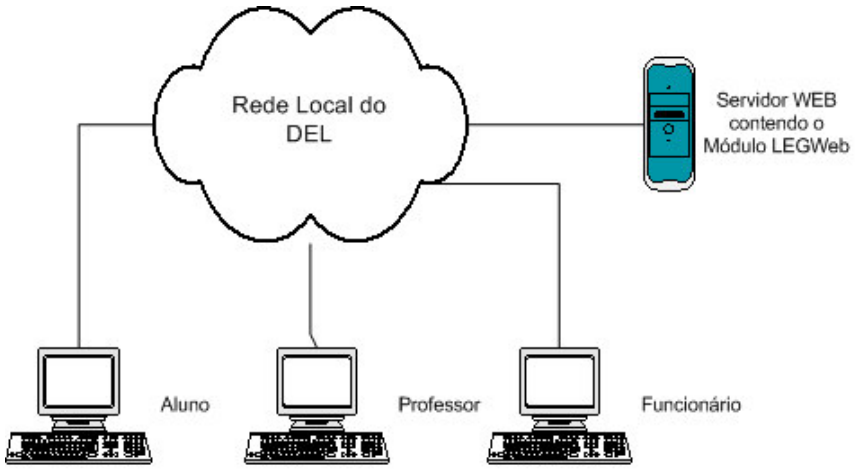


Figura I.3 - Modelo de comunicação através da *Intranet.*

I.3. Estrutura de Documentação

A presente documentação está dividida de maneira a permitir não só o perfeito entendimento do funcionamento do módulo, mas também de maneira a reproduzir o modo como este foi concebido, habilitando o leitor também a dele se utilizar de maneira simples e direta.

O Capítulo II oferece os subsídios teóricos necessários ao entendimento de todo o planejamento e funcionamento do LegWeb. Esse embasamento vai desde o conceito de UML (*Unified Modeling Language*), fundamental para o pleno compreendimento do funcionamento deste módulo, até os conceitos de banco de dados e PHP (PreHypertext Processor) [4]. A interligação dessas tecnologias torna-se mais óbvia no capítulo seguinte.

O Capítulo III diz respeito ao desenvolvimento do projeto em si, apresentando a construção do modelo lógico do módulo, utilizando a UML. Neste capítulo é apresentada também desde a estrutura da base de dados, com conceitos de banco de dados relacionais e sua estruturação, até a base real, com todas as tabelas e campos necessários para a implementação do sistema proposto.

O Capítulo IV é destinado às conclusões a respeito do projeto. É dado um panorama das funcionalidades e dos benefícios que reverterão como utilidades factuais ao Laboratório de Eletrônica da Graduação e ao Departamento de Eletrônica e Computação.

O Apêndice um oferece uma lista dos acrônimos utilizados durante todo o documento para referência. As siglas e termos em inglês estão destacados por itálico.

O Apêndice seguinte – dois – especifica todos os requisitos – *hardware* e *software* – para a instalação correta e funcional de todos os componentes do LEGWeb.

O Apêndice três apresenta um manual de administração prático para utilização e manutenção do LEGWeb, cobrindo as questões destinadas à sua configuração e operação.

O Apêndice quatro oferece a visão do conteúdo do CD-ROM que acompanha esta documentação, contendo todos os recursos computacionais necessários ao projeto, incluindo os fontes de todos os programas.

O documento, como um todo, está organizado de maneira a promover o entendimento e capacitação no uso e desenvolvimento do módulo.

Capítulo II . Nivelamento Teórico

Nas várias etapas do desenvolvimento do projeto, fez-se necessário o estudo aprofundado de técnicas e fundamentos que tornassem viáveis a sua implementação.

Tais técnicas e fundamos são apresentados a seguir, como pré-requisitos para entender o adequado funcionamento deste trabalho. A apresentação dos conceitos abaixo não visa dar ao leitor um conhecimento profundo dos assuntos, apenas a formação mínima necessária à compreensão do projeto.

Remetemos à bibliografia, situada ao final deste documento, para posteriores consultas em caso de necessidade de aprofundamento.

II.1 . Conceitos de UML

A UML surgiu da necessidade de técnicas mais robustas para especificação de grandes sistemas. Três importantes técnicas, Booch, OOSE e OMT começaram a evoluir em uma única direção. Desta forma, em meados da década de 90, os responsáveis por estas técnicas, Grady Booch, Ivar Jacobson e James Rumbaugh decidiram unir esforços e criar uma técnica única e abrangente, englobando os pontos fortes de cada uma das ferramentas existentes.

Deste esforço conjunto surgiu, em outubro de 1995, um esboço da versão 0.8 da UML. Logo após, várias empresas de renome se juntaram no aprimoramento da UML. Entre estas empresas podemos citar IBM, Oracle, Microsoft, Rational e outras. Desta união surgiu a UML 1.0, uma linguagem bem-definida e poderosa.

Em 1997 a UML, em sua versão 1.1, foi adotada pela OMG (*Object Management Group*) como sua linguagem padrão de modelagem. A manutenção da UML foi assumida pela RFT (*Revision Task Force*) do OMG, que lançou a versão 1.2 em 1998 e a versão 1.3 no final deste mesmo ano.

A UML é uma linguagem destinada à estruturação de projetos. Pode ser utilizada na modelagem de sistemas que variam desde simples controles de fluxo de portarias até sistemas extremamente complexos para controle de fábricas. Com a UML é possível especificar, visualizar, construir e documentar todo um sistema a ser desenvolvido. Um dos principais

objetivos da UML é tornar todo o processo de desenvolvimento totalmente transparente e organizado, permitindo que várias pessoas ou até mesmo vários grupos trabalhem de forma sincronizada e com a menor perda de tempo possível, pois inclui desde a definição de classes até os casos de uso do sistema.

Apesar de não ser uma linguagem gráfica de programação, a UML permite que o código seja gerado diretamente de seus modelos. É possível, por exemplo, gerar as classes definidas no modelo para linguagens como Java, C++, PHP [4] e outras.

A UML pode ser dividida em três grandes grupos:

- Itens

- Relacionamentos

- Diagramas.

II.1.1 . Itens

Eles constituem o núcleo da UML. Existem quatro tipos de itens:

II.1.1.1 . Itens Estruturais

Representam basicamente a parte estática do modelo, representando elementos conceituais ou físicos. Existem os seguintes itens estruturais:

- Classes: a definição é a mesma de qualquer linguagem de programação orientada a objetos, ou seja, uma classe é um conjunto de atributos, procedimentos e operações compartilhados com um ou mais objetos.

- Classes Ativas: uma classe ativa é similar a uma classe normal. A diferença é que em uma classe ativa seus objetos podem representar elementos concorrentes com outros elementos.

- Componentes: os componentes representam a parte física do modelo conceitual. O código, por exemplo, é um componente.

- Interfaces: uma interface define um conjunto de operações que especifica comportamentos externos de uma classe.

- Colaborações: uma colaboração é uma agregação de interações que em sua totalidade constituem um ente superior que rege todo o comportamento desses elementos.

- Casos de Uso: um caso de uso descreve um conjunto de ações realizadas pelo sistema em interações com seus usuários (atores), permitindo assim uma descrição detalhada do comportamento do sistema e de suas interações.

- Nós: assim como os componentes, os nós são componentes físicos e representam os recursos de processamento (computacionais), por exemplo, a CPU, memória, etc. Um nó pode conter vários componentes. Um componente pode migrar de um nó para outro.

II.1.1.2 . Itens Comportamentais

Os itens comportamentais representam a parte dinâmica da UML. Eles representam o que acontece durante o processo na variação espaço-tempo. São dois os itens comportamentais:

- Interação: por meio de uma interação é possível visualizar a troca de mensagens entre objetos de um modelo UML. Podem-se ainda demonstrar comportamentos entre seqüências de ações.

- Máquina de Estado: as máquinas de estado representam os estados pelos quais os objetos podem passar durante determinados eventos. Pode-se ainda especificar o comportamento de uma determinada classe por intermédio de uma máquina de estado.

II.1.1.3 . Itens de Anotação

Os itens de anotação são responsáveis por trazer maior clareza ao modelo UML. Por meio desses itens é possível incluir comentários, notas e observações ao modelo. Existe apenas um elemento no item de anotação, ou seja, a nota. Nesse item é possível incluir

qualquer tipo de comentário que sirva para esclarecer melhor um ponto determinado do modelo.

II.1.1.4 . Itens Agrupacionais

Esses itens promovem a organização de um modelo UML. Eles agrupam o modelo em blocos distintos, cada bloco podendo ser decomposto em outros blocos contendo os demais itens do modelo UML. O único elemento que faz parte desse item é o pacote, que agrupa (ou pode agrupar) todos os itens mencionados anteriormente.

II.1.2 . Relacionamentos

Os relacionamentos permitem construir toda a parte de comunicação dentro do modelo, pois é por meio deles que são determinadas as dependências, associações, generalizações e realizações do modelo. São quatro os tipos de relacionamento disponíveis na UML:

II.1.2.1 . Associação

Uma associação demonstra um relacionamento estrutural entre objetos do modelo. Em uma associação pode-se demonstrar quantitativamente como os objetos se relacionam (por exemplo, no caso de um cliente de um banco e suas contas, um cliente pode ter uma ou mais contas e uma conta pertence a um único cliente).

II.1.2.2 . Realização

Os relacionamentos de realização podem ser vistos entre casos de uso e colaborações e entre interfaces e classes, pois realizações demonstram o que um determinado objeto garante executar em prol de outro.

II.1.2.3 . Dependência

Os relacionamentos de dependência entre dois itens do modelo demonstram que a execução (alteração) de um item pode afetar o outro.

II.1.2.4 . Generalização

As generalizações demonstram que itens são especializações de outros itens. Existe claramente um modelo de pai e filho, em que os filhos herdam a estrutura e o comportamento dos pais, podendo alterá-los conforme suas necessidades.

II.1.3 . Diagramas

Diagramas são representações gráficas de um sistema ou de partes dele. Num diagrama é possível visualizar todos os itens e relacionamentos definidos na UML. A UML disponibiliza nove tipos diferentes de diagramas:

II.1.3.1 . Diagrama de Classes

Este diagrama mostra as classes e classes ativas, interfaces e colaborações e seus relacionamentos (de generalização, dependência, associação ou realização). Um diagrama de classe está intimamente ligado à representação conceitual do modelo lógico de programa de um sistema, sendo utilizado em sistemas que utilizam modelagem por objetos (OO – *Object Oriented*). Desta forma, pode-se vê-lo em sistemas baseados em Java, PHP [4], C++, etc.

II.1.3.2 . Diagrama de Objetos

O diagrama de objetos exibe uma visão estática de um sistema, mostrando-o da mesma forma que o diagrama de classes, mas sob a ótica de situações reais ou protótipos do modelo.

II.1.3.3 . Diagrama de Casos de Uso

Talvez um dos mais importantes diagramas da UML, o diagrama de casos de uso exibe as interações entre casos de uso e seus atores (uma classe).

O diagrama de casos de uso é de fundamental importância para se entender o comportamento do sistema em situações de interação entre este e usuários ou situações reais.

II.1.3.4 . Diagrama de Interações

O diagrama de interações, como o nome diz, exibe as interações entre objetos, conjunto de objetos e seus relacionamentos (inclusive as mensagens trocadas entre eles). O

diagrama de interações pode ser dividido em diagramas de seqüências, os quais têm uma visão basicamente relacionada à linha do tempo da execução do sistema, e os diagramas de colaborações, os quais têm uma visão estrutural e organizacional dos objetos e suas interações, incluindo a troca de mensagens entre eles.

II.1.3.5 . Diagrama de Gráficos de Estados e Diagrama de Atividades

O diagrama de gráficos de estados, como o nome indica, representa máquinas de estados, atividades e eventos onde o tipo de diagrama representa a visão dinâmica do sistema, sendo útil para a modelagem de comportamentos e classes, por exemplo. O diagrama de atividades é uma derivação do diagrama de gráficos de estados e permite um mapeamento do fluxo de controle entre objetos do sistema.

II.1.3.6 . Diagrama de Componentes

Esse diagrama mostra a organização e as dependências existentes entre componentes de um sistema, estando intimamente ligados a classes, interfaces e colaborações.

II.1.3.7 . Diagrama de Implantação

O diagrama de implantação está relacionado a nós e componentes. Por meio dele é possível especificar a configuração dos nós e os componentes existentes, pois um nó pode incluir um ou mais componentes.

II.1.4 . Representação Gráfica dos Elementos da UML

Como todas as linguagens, a é usada para transmitir e receber informações. Nesse caso, as informações são a definição detalhada de como um sistema deve ser implementado para realizar o que o usuário deseja.

É importante conhecer a linguagem para entender os modelos já desenvolvidos por outras pessoas ou mesmo para explicar como um sistema deve ser implementado. Cada conceito a ser modelado (subsistema, classe, relações, etc.) tem uma representação gráfica específica na linguagem. É muito importante respeitar as convenções definidas na UML para que outras pessoas possam entender os diagramas gerados.

A principio, a linguagem UML pode ser utilizada com qualquer processo de desenvolvimento de software, daí a importância de serem respeitados os padrões de representação. A grande vantagem, neste caso, é a portabilidade entre diversas linguagens.

II.1.4.1 . Itens

Classes: são representadas por um retângulo dividido em três seções: nome da classe, elementos e operações. A figura II.1 apresenta um exemplo de aplicação da notação ao apresentar a classe **cliente**.

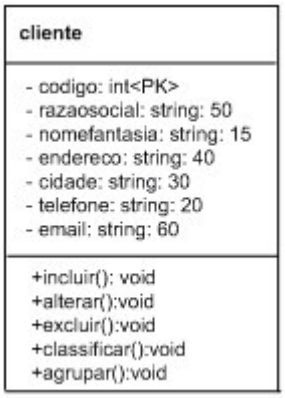


Figura II.1 - Classes

Classes Ativas: Sua representação é idêntica a das classes, exceto pelas linhas do retângulo. A figura II.2 apresenta um exemplo de aplicação da notação ao apresentar a classe **controleHTML**.

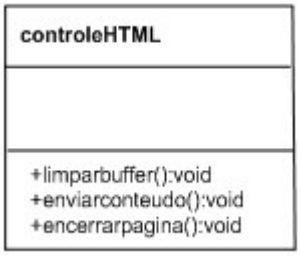


Figura II.2 - Classe ativa

Casos de Uso: os casos de uso são representados por elipses com linhas sólidas, contendo uma descrição sucinta de seu objetivo. A figura II.3 apresenta o relacionamento do ator **aluno** com o caso de uso responsável pelo ***login*** no sistema.

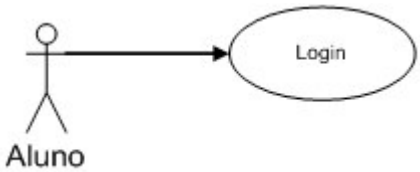


Figura II.3 - Caso de uso e um ator

Nós: os nós são representados por cubos com seu nome ou uma descrição curta inclusa. A figura II.4 apresenta um exemplo de nó apresentando um servidor executando o sistema operacional *Linux* [2] com o *software* de publicação de páginas na *Internet* Apache.

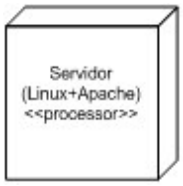


Figura II.4 - Nós

Componentes: os componentes são representados por um retângulo estereotipado com outros dois retângulos menores (abas). A figura II.5 apresenta um exemplo desta representação.

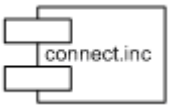


Figura II.5 - Componentes

Máquinas de Estados: são representadas por retângulos com os vértices arredondados, incluindo uma descrição sucinta ou o seu nome. A figura II.6 apresenta como exemplo a representação da máquina de estado **enviar**.

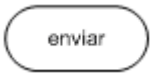


Figura II.6 - Máquina de estado

Interações: as mensagens são representadas por uma seta com a linha sólida e o corpo preenchido, incluindo a operação. A figura II.7 apresenta um exemplo desta representação.

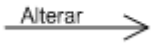


Figura II.7 - Mensagens

Pacotes: os pacotes são representados por uma guia com uma pequena descrição em seu interior. A figura II.8 representa o pacote responsável pelo **Gerenciamento de Empréstimos.**

Figura II.8 - Pacotes

Notas: uma nota é representada por um retângulo com um dos cantos dobrado para dentro. A figura II.9 apresenta um exemplo desta representação.

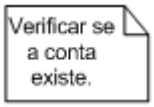


Figura II.9 - Nota

Dependência: uma dependência é representada graficamente por uma linha tracejada, podendo ter ou não uma seta aberta em uma de suas extremidades e um título. A figura II.10 apresenta um exemplo desta representação.

Figura II.10 - Dependência

Associação: uma associação é representada por uma linha sólida, direcionada ou não (com uma seta aberta), com títulos, nomes e indicação de multiplicidades. A figura II.11 apresenta um exemplo desta representação.

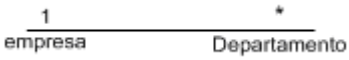


Figura II.11 - Associação

Generalização: uma linha sólida com uma seta fechada e sem preenchimento em uma das suas extremidades representa este elemento da UML. A figura II.12 apresenta um exemplo desta representação.

Figura II.12 - Generalização

Um exemplo de um modelo UML completo seria como o apresentado nas figuras II.13 (diagrama de classes) e II.14 (diagrama de casos de uso):

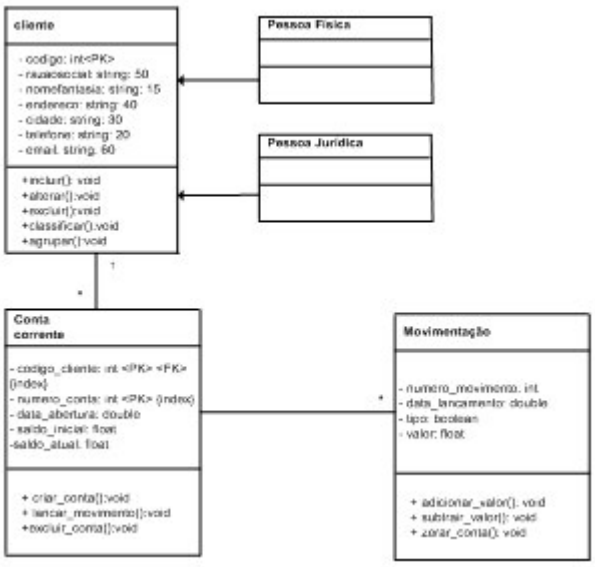


Figura II.13 - Diagrama de classes

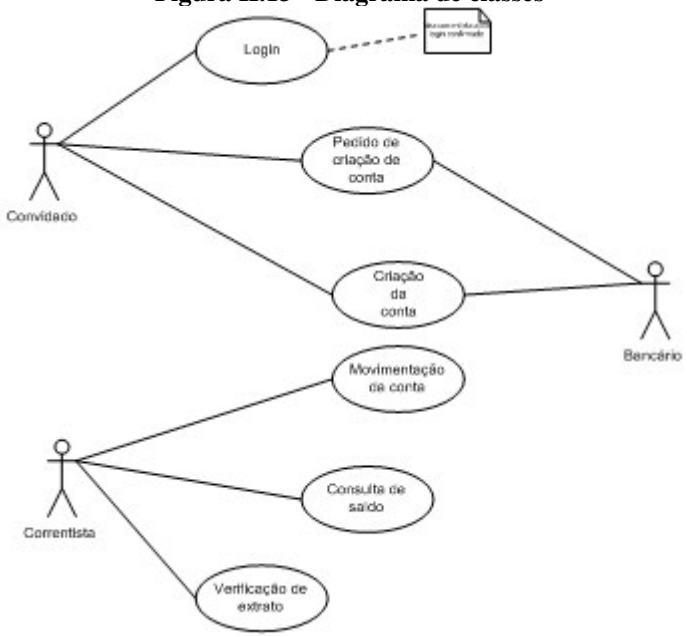


Figura II.14 - Diagrama de casos de uso

Dentro do escopo deste projeto, será abordado o modelo conceitual do sistema, utilizando macro visões via pacotes e diagramas de casos de uso e de atividade para demonstrar as funcionalidades do sistema.

II.1.4.2 . Diagrama de Casos de Uso

O diagrama de caso de uso é a principal ferramenta para avaliar as interações do sistema com o mundo real. O mundo real pode ser representado por pessoas, máquinas, outros programas, enfim, qualquer coisa que interaja com o módulo que está sendo desenhado.

Como um exemplo bem claro da utilização de um diagrama de caso de uso, imagine-se uma situação corriqueira dos dias atuais, como o saque de uma determinada quantia num caixa eletrônico. Ao se colocar no papel todos os passos executados pelo cliente após sua entrada no caixa automático, o seguinte resultado será obtido:

1) O cliente insere o cartão do banco no local apropriado.

2) O sistema processa os dados do cartão.

3) O sistema exibe as funções disponíveis na tela do terminal.

4) O cliente seleciona a opção de saque.

5) O sistema solicita o valor a ser sacado.

6) O cliente informa o valor desejado para saque.

7) O sistema solicita a senha de segurança do cliente.

8) O cliente informa sua senha pessoal.

9) O sistema processa a solicitação do cliente.

10) O sistema disponibiliza a quantia desejada e solicita ao cliente que a retire.

11) O sistema solicita que o cliente retire seu cartão e encerra a operação.

Esta é uma versão simples da operação de saque em um terminal eletrônico. A grosso modo, se tudo correr bem, após o décimo primeiro passo o cliente estará fora do caixa eletrônico com a quantia desejada. Esses passos são conhecidos como "Procedimentos" e descrevem o fluxo normal do processamento, porém existem situações em que este fluxo é interrompido ou desviado. Abaixo segue uma relação de eventos que podem causar a anomalia citada:

1) O sistema está fora do ar ou o caixa está sem recursos (dinheiro).

2) O cartão está desmagnetizado ou a leitora não consegue ler os dados.

3) O cliente informa uma senha incorreta várias vezes.

4) O cliente não tem saldo suficiente para sacar o valor desejado.

5) O caixa possui somente alguns tipos de notas (notas de R$ 10 e R$ 50, por exemplo).

6) O dinheiro não sai da máquina devido a problemas mecânicos.

7) O cartão não é ejetado da máquina.

8) O banco eletrônico perde a comunicação com a central.

Para prever estas e outras situações que podem surgir durante o processamento desta operação, é necessário criar um fluxo alternativo de processamento, por meio do qual é possível elencar os problemas e as atitudes que o sistema deverá tomar em cada caso.

Montando o procedimento e suas alternativas, o modelo seria:

<u>Procedimentos:</u>

1) O cliente insere o cartão do banco no local apropriado.

2) O sistema processa os dados do cartão.

3) O sistema exibe as funções disponíveis na tela do terminal.

4) O cliente seleciona a opção de saque.

5) O sistema solicita o valor a ser sacado.

6) O cliente informa o valor desejado para saque.

7) O sistema solicita a senha de segurança do cliente.

8) O cliente informa sua senha pessoal.

9) O sistema processa a solicitação do cliente.

10) O sistema disponibiliza a quantia desejada e solicita ao cliente que a retire.

11) O sistema solicita que o cliente retire seu cartão e encerra a operação.

Alternativas:

1) O sistema está fora do ar ou o caixa está sem recursos (dinheiro).

O sistema exibe uma mensagem de erro na tela do terminal, informando ao usuário sobre o problema existente.

2) O cartão está desmagnetizado ou a leitora não consegue ler os dados.

O sistema exibe uma mensagem de erro e solicita ao usuário que tente outra vez ou solicite outro cartão ao banco.

3) O cliente informa uma senha incorreta várias vezes.

O sistema avisa ao usuário que seu cartão está bloqueado e deve ser retirado na agência bancária à qual o cliente pertence.

4) O cliente não tem saldo suficiente para sacar o valor desejado.

O sistema exibe uma mensagem de saldo insuficiente e finaliza o processamento.

5) O caixa possui somente alguns tipos de notas (notas de R$ 10 e R$ 50, por exemplo).

O sistema recusa qualquer valor que não seja uma combinação das notas disponíveis.

6) O dinheiro não sai da máquina devido a problemas mecânicos.

O cliente utiliza o telefone de emergência, avisa à equipe de suporte sobre o ocorrido e espera uma providência (o cancelamento do saque, por exemplo).

7) O cartão não é ejetado da máquina.

O cliente utiliza o telefone de emergência, avisa à equipe de suporte sobre o ocorrido e espera uma providência.

8) O banco eletrônico perde a comunicação com a central.

O cliente utiliza o telefone de emergência, avisa à equipe de suporte sobre o ocorrido e espera uma providência.

As operações aqui listadas, transformadas em gráfico, ficam conforme o diagrama de caso apresentado na figura II.15:

Figura II.15 - Diagrama de caso de uso para operação de saque.

Nota-se a presença de uma linha pontilhada para designar um processo que utiliza as funções de outro processo (por exemplo, o processo "ler dados do cartão" utiliza o processo "validar cartão"). A visão gráfica é extremamente útil para um entendimento por completo do processo desejado.

II.1.4.3 . Diagrama de Atividades

O diagrama de atividades descreve uma série de ações que serão executadas pelo sistema para completar determinada tarefa. No diagrama de atividades tem-se uma visão mais detalhada dos resultados que devem ser alcançados por determinado módulo do sistema. Nele

é possível indicar os atores e as ações pertinentes a cada um deles. Pode-se inclusive determinar rotas alternativas e processamentos concomitantes.

A idéia básica em um diagrama de atividades é dividir as ações em raias, sendo que cada raia representa um ator, e dentro delas está a descrição das atividades que devem ser executadas pelo sistema / atores. Pode-se utilizar o diagrama de atividades para ter uma visão de um determinado processo (um fluxo de trabalho, por exemplo) ou ainda descrever uma única tarefa (operação) detalhadamente.

Cada ação no diagrama de atividades é representada por um retângulo com as bordas arredondadas, o sentido do fluxo é indicado por uma linha sólida com uma seta em uma de suas extremidades (a qual indica o sentido), o início do fluxo é indicado por um círculo preenchido e uma ramificação (um fluxo convencional) é representada por um losango. Uma bifurcação (fluxo concorrente) é representada por uma linha reta na horizontal ou vertical e o término do fluxo é indicado por um círculo contendo um outro círculo preenchido em seu interior. A figura II.16 ilustra as representações acima discutidas.

Figura II.16 - Representação gráfica dos elementos do diagrama de atividade.

A divisão do diagrama em raias permite que se tenha uma idéia clara das responsabilidades de cada ator dentro do processamento, possibilitando ainda uma melhor visualização do fluxo do sistema.

Tomando o exemplo anterior como base, será montada a operação de saque em caixa eletrônico por meio de um diagrama de atividades. Tem-se três elementos (atores) para o diagrama de atividades: o cliente, o caixa eletrônico e o banco. O papel de cada um desses atores está bem definido: o cliente é o ator responsável pelo início do processamento e o direcionador das ações; o caixa eletrônico é responsável pelo interfaceamento entre o cliente e o banco, realizando as verificações simples e processando a requisição do cliente; já o banco

realiza as validações do cliente (cartão e senha), processa o pedido de saque (verificando o saldo, por exemplo) e autoriza ou não que o dinheiro seja entregue ao cliente.

Ao transformar essas ações em diagrama de atividades, tem-se o diagrama de atividades como o apresentado na figura II.17:

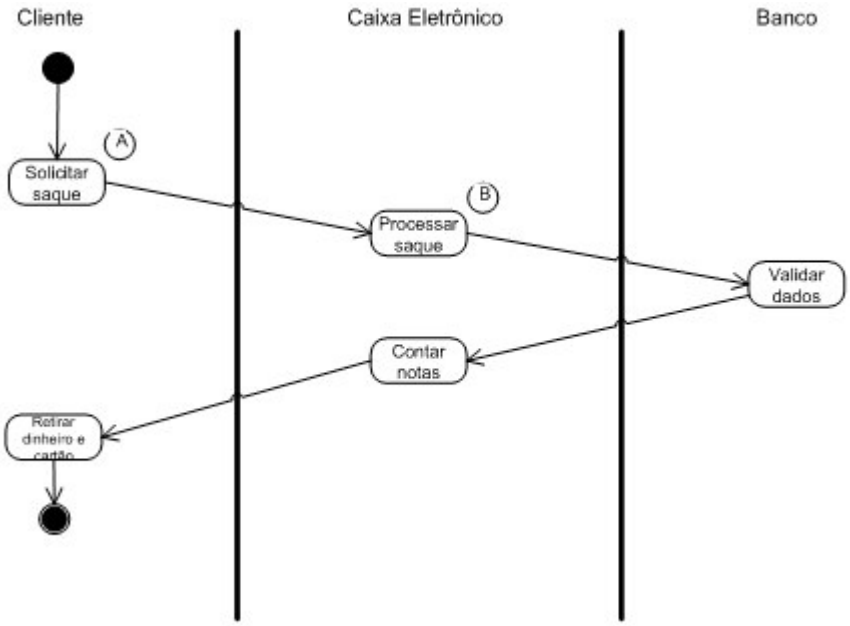


Figura II.17 - Diagrama de atividades, visão macro.

O diagrama da figura II.17 mostra uma visão macro do processo de saque em caixa eletrônico. O diagrama de atividade permite montar desde uma visão macro do processo até uma visão bem detalhada de cada item do processo. O item **A** da figura II.17, **"solicitar saque"** e o item **B** da mesma figura, **"processar saque"**, podem, por exemplo, ser mais detalhados de forma a possibilitar uma visão mais completa do processo. Ao montar um diagrama de atividades para esses itens, tem-se o detalhamento do processo **A,** apresentado na figura II.18 e o do processo **B,** apresentado na figura II.19:

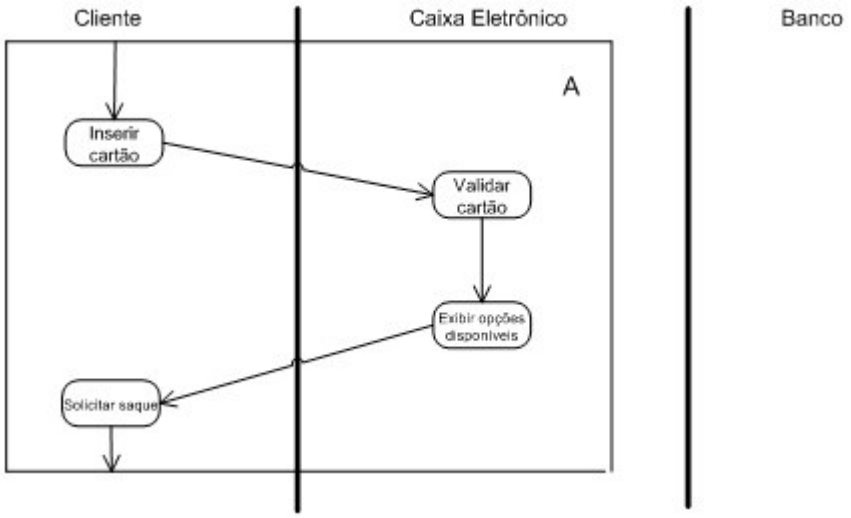


Figura II.18 - Detalhamento do processo A do diagrama de atividades.

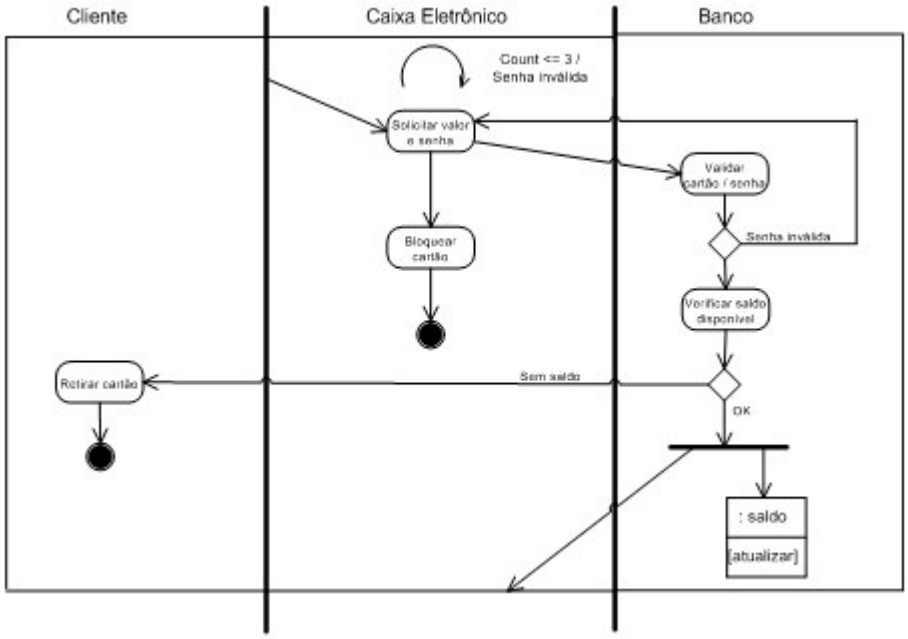


Figura II.19 - Detalhamento do processo B do diagrama de atividades.

II.2 . Conceitos de Banco de Dados

No início da década de 60, foram lançados os primeiros sistemas gerenciadores de banco de dados (SGBD), tendo como principal proposta o aumento na produtividade nas atividades de desenvolvimento e manutenção de sistemas, até então realizadas de forma artesanal.

Oriundos do ambiente de *mainframes*, os SGBD tornaram-se mais populares e amigáveis com o advento da microinformática. Cada vez mais as fronteiras entre esses dois mundos estreitam-se e a concorrência pelo domínio do mercado de SGBD tem levado seus diversos fabricantes a sofisticarem seus produtos. Cada nova versão lançada incorpora novidades como interfaces gráficas, ferramentas de apoio ao desenvolvimento, utilitários para gerenciamento de banco de dados e facilidades para extração de dados. Essa evolução vem tornando o trabalho de programadores, analistas e usuários menos artesanal, com reflexos na qualidade e produtividade.

A literatura classifica os SGBDs como Hierárquicos, Redes e Relacionais. Essa classificação representa a evolução desses produtos no curso da história. Atualmente, o mercado é dominado pelos SGBDs Relacionais e caminha para a colocação em escala comercial dos SGBD Orientados a Objetos (OO).

II.2.1 . Objetivos e Vantagens

O desenvolvimento da tecnologia de banco de dados tem se pautado por buscar alcançar como objetivo permanente o aumento de produtividade nas atividades de desenvolvimento e manutenção de sistemas.

Nesse sentido os fabricantes de SGBDs vêm dotando seus produtos com mecanismos que facilitam a adaptação dos BDs às novas necessidades que surgem no dia a dia e que reduzem o trabalho de programação. Aliado a esses dois fatores existe toda uma filosofia que orienta os técnicos na escolha do melhor produto para a sua empresa e no trabalho de projeto de banco de dados.

Segue uma definição para os tipos de sistema de armazenamento existentes:

II.2.1.1 . Sistema Tradicional

São aqueles em que os dados do sistema estão armazenados fisicamente, separados um do outro. O acesso é feito pelos programas da aplicação, associando o nome externo dos arquivos e definindo todo o registro independente da utilização dos campos.

II.2.1.2 . Sistema de Banco de Dados

É aquele em que os dados são definidos para o SGBD através da DDL (linguagem de definição de dados). Fisicamente estão armazenados em um único local, sendo o acesso realizado apenas através do SGBD. Nos programas da aplicação, é necessário apenas definir os campos que serão utilizados pelo programa.

Dessa filosofia destacam-se a seguir algumas vantagens de um BD com relação aos sistemas tradicionais de armazenamento de dados:

1) **Redução ou Eliminação de Redundâncias:** possibilita a eliminação de dados privativos de cada sistema. Os dados, que eventualmente são comuns a mais de um sistema, são compartilhados por eles, permitindo o acesso a uma única informação sendo consultada por vários sistemas.

2) **Eliminação de Inconsistências**: através do armazenamento da informação em um único local com acesso descentralizado e sendo compartilhada por vários sistemas, os usuários estarão utilizando uma informação confiável. A inconsistência ocorre quando um mesmo campo tem valores diferentes em sistemas diferentes.

3) **Compartilhamento dos Dados**: permitem a utilização simultânea e segura de um dado, por mais de uma aplicação ou usuário, independente da operação que esteja sendo realizada. Deve ser observado apenas o processo de atualização concorrente, para não gerar erros do processamento (atualizar simultaneamente o mesmo campo do mesmo registro). Os aplicativos são, por natureza, multi-usuário.

4) **Restrições de Segurança**: definem para cada usuário o nível do acesso a ele concedido (leitura, leitura e gravação ou sem acesso) ao arquivo e/ou campo. Este recurso impede que pessoas não autorizadas utilizem ou atualizem um determinado arquivo ou campo.

5) **Padronização dos Dados**: permite que os campos armazenados na base de dados sejam padronizados segundo um determinado formato de

armazenamento (padronização de tabela, conteúdo de campos, etc.) e ao nome de variáveis seguindo critérios e padrões preestabelecidos. Por exemplo, para o campo "Sexo" somente será permitido armazenamento dos conteúdos "M" ou "F".

a. Os programas da aplicação definem apenas os campos que serão utilizados, independente da estrutura interna dos arquivos.

b. Quando há inclusão de novos campos no arquivo, será feita manutenção apenas nos programas que utilizam esses campos, não sendo necessário alterar os demais programas.

6) **Manutenção da Integridade dos dados**: consiste em impedir que um determinado código ou chave em uma tabela não tenha correspondência em outra tabela. Por exemplo, um código de uma determinada disciplina na tabela Histórico Escolar sem a sua descrição na tabela Disciplina.

II.2.2 . Terminologia

Para compreender com maior facilidade os conceitos relativos à Banco de Dados, é de suma importância revisar alguns conceitos básicos referentes à teoria e terminologia de arquivos convencionais, visto que os primeiros SGBDs foram criados a partir do aperfeiçoamento de sistemas gerenciadores de arquivo e ainda utilizam muito da base conceitual e da terminologia de arquivos.

II.2.2.1 . Arquivo

Um arquivo é uma coleção de registros do mesmo tipo, ou seja, referentes a um mesmo assunto e com o mesmo formato padrão (*layout*). Constitui o componente do sistema no qual são armazenados os dados, que combinados através dos programas, servem de base para a geração da informação desejada pelo usuário através de relatórios e consultas on-line.

Um sistema de controle de notas, por exemplo, pode armazenar seus dados em diversos arquivos, cada qual contendo informações sobre um determinado item do sistema: aluno, professor, matéria, nota, etc.

Essas informações podem ser combinadas através de programas para gerar, por exemplo, o Boletim Escolar, a Pauta ou uma tela de Consulta de Notas.

II.2.2.2 . Registro

Um registro é constituído por um conjunto de campos valorados (contendo dados). Consiste na unidade de armazenamento e recuperação da informação em um arquivo. Geralmente, os registros de um arquivo possuem um formato padrão (*layout*), definido pela seqüência, tipo e tamanho dos campos que o compõem. Porém, algumas linguagens de programação permitem a criação de registros com formatos diferentes em um mesmo arquivo, recurso este que raramente é utilizado.

II.2.2.3 . Campo

É a unidade básica formadora de um registro. Constitui a célula da informação. É a menor porção de um arquivo que pode ser referenciada por um programa.

Cada campo possui NOME, TIPO e TAMANHO. Os tipos de campo mais comuns são apresentados na figura II.20:

NUMBER	Armazena somente números. Pode conter casas decimais e pode ser utilizado em operações matemáticas.
CHAR ou ALFANUMÉRICO	Pode armazenar letras, números e caracteres especiais.
DATE	Armazena datas fazendo consistência automática.
MEMO ou LONG	Armazena textos em formato livre.

Figura II.20 - Tipos de campo mais comuns.

A figura II.21 sintetiza os conceitos de ARQUIVO, REGISTRO e CAMPO:

Arquivo Aluno

CAMPOS **TIPO e TAMANHO**	**MATRÍCULA** *number (3)*	**NOME** *char (30)*	**ENDEREÇO** *char (50)*	**DT_NASCIMENTO** *date*
Registros	001 002 003	José Maria Ana	Rua Moraes... Rua Francisco... Rua Uruguai...	23/1/80 5/9/76 2/10/77

Figura II.21 - Exemplo de uso de conceitos de arquivo, registro e campo.

II.2.2.4 . Chave Primária (Primary Key - PK)

A CHAVE PRIMÁRIA (ou simplesmente CHAVE) é o identificador único de um registro em um arquivo. Pode ser constituída de um campo (CHAVE SIMPLES) ou pela combinação de dois ou mais campos (CHAVE COMPOSTA), de tal maneira, que não existam dois registros no arquivo com o mesmo valor de chave primária.

Em regra, todo arquivo deve possuir uma chave primária que permita a identificação inequívoca do registro, especialmente para dar maior consistência aos processos de inclusão, alteração e exclusão de dados.

Para que não ocorram duplicatas nos valores da chave, os campos que a compõem são de PREENCHIMENTO OBRIGATÓRIO (*NOT NULL*).

Na escolha da chave primária de um arquivo deve-se buscar campos que possuam ESTABILIDADE no valor armazenado. A escolha do NÚMERO DO TELEFONE como chave de um cadastro de clientes, por exemplo, seria inadequada, por que esse valor pode mudar com freqüência.

Deve-se também evitar a escolha de campos que possam causar AMBIGÜIDADE em relação aos valores nele contidos. Nesse sentido, seria inadequada a escolha do campo NOME para chave de um cadastro de clientes, haja vista, que um mesmo nome pode ser escrito de várias formas. Por exemplo: LUÍS, LUIZ, LOUIS, LOYS, LUYS. Ao cobrar uma fatura de um cliente com um nome como esse, a probabilidade de errar o cliente seria grande. Além disso, a extensão do campo (trinta ou mais caracteres) é um outro aspecto que aumenta a possibilidade de erros.

II.2.3 . SQL (*Structured Query Language*)

Conforme mencionado anteriormente, nos dias de hoje os SGBDs relacionais dominam o mercado. Dentre esse tipo de SGBDs, uma linguagem fácil e rápida que se sobressai é a SQL. Essa linguagem teve tal absorção que os bancos de dados mais utilizados hoje interpretam a linguagem com naturalidade.

O módulo principal ao qual o LegWeb está conectado, o SID [1], utiliza esta linguagem, através do gerenciador MySQL [3]. As instruções de SQL utilizadas nos códigos do módulo serão vistas e detalhadas apêndice três.

II.2.4 . Gerenciador MySQL (*Structured Query Language*)

O MySQL [3] é a aplicação do banco de dados mais popular do mundo, o SQL, multi-usuário e *multi-threaded*.

Ele é uma implementação cliente-servidor que consiste de um servidor e diferentes programas clientes e bibliotecas. Suas principais vantagens são velocidade, robustez e facilidade de uso.

Para gerenciá-lo, existe um utilitário chamado *mysql* [3] que é instalado junto com o gerenciador do banco de dados. Uma vez iniciado o seu gerenciador, pode-se criar o banco de dados utilizando o comando apresentado na figura II.22:

```
> CREATE DATABASE teste;

> USE teste;
```

Figura II.22 - Comandos para criação e seleção para uso de banco de dados.

Estes comandos criam um banco de dados e selecionam esse banco para utilização nesta seção.

Em MySQL [3], como em muitos outros bancos de dados, o conceito da estrutura que mantém os blocos (ou registros) de informações é chamado de tabela. Estes registros, por sua vez, são constituídos de objetos menores que podem ser manipulados pelos usuários, conhecidos por tipos de dados (*datatypes*). Juntos, um ou mais *datatypes*, formam um registro (*record*). Uma hierarquia de banco de dados funciona como o relacionamento apresentado na figura II.23:

Banco de dados > Tabela > Registro > Tipo de dados

Figura II.23 - Hierarquia de banco de dados.

Os tipos de dados possuem diversas formas e tamanhos, permitindo ao programador criar tabelas específicas de acordo com suas necessidades. MySQL [3] provê um conjunto bem grande de tipos de dados. Abaixo, a tabela II.1 apresenta os tipos numéricos, a Tabela II.2 os tipos temporais e a tabela II.3 os tipos *string* (texto).

.

Numéricos	
TIPO	**DESCRIÇÃO**
TINYINT	Inteiro muito pequeno variando de – 128 a +127.
SMALLINT	Inteiro variando de -32768 a +32767.
MEDIUMINT	Inteiro variando de -8388608 a + 8388607.
INT	Inteiro normal de – 2147483648 a + 2147483647.
BIGINT	Inteiro grande.
FLOAT	Ponto flutuante com precisão definida.
DOUBLE	Ponto flutuante normal.
DECIMAL	Ponto flutuante com número exato de decimais.

Figura II.24 - Tipos de dados numéricos do MySQL.

Temporais	
TIPO	**DESCRIÇÃO**
DATE	Data de 100-01-01 até 9999-12-31.
DATETIME	Data e hora.
TIMESTAMP	Armazena a data em formato padrão UNIX.
TIME	Contador de tempo.
YEAR	Ano com dois ou quatro dígitos (padrão de quatro dígitos).

Figura II.25 - Tipos de dados temporais do MySQL.

String	
TIPO	**DESCRIÇÃO**
CHAR	String de comprimento fixo.
VARCHAR	String de comprimento variável.
TINYTEXT	Texto contendo até 255 caracteres.
TEXT	Texto com até 65535 caracteres.
MEDIUMTEXT	Texto com máximo de 16777215 caracteres.
LONGTEXT	Texto com até 4294967295 caracteres.
ENUM	String que pode conter apenas um dos valores listados.
SET	String que pode conter 0 ou mais valores listados.

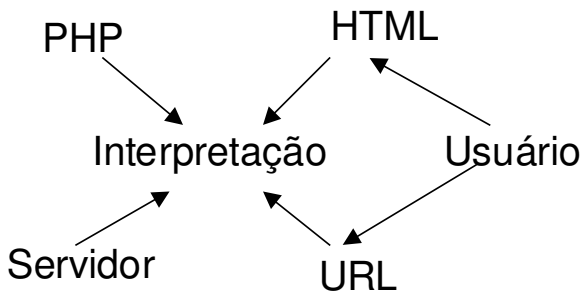
Figura II.26 - Tipos de dados string do MySQL.

II.3 . PHP

PHP [4] é uma linguagem de elaboração de *scripts*, ou seja, é uma linguagem que processa, através de *scripts*, solicitações feitas por um cliente (normalmente um usuário com uma máquina conectada na *Internet* e usando um navegador tal como o Internet Explorer ou o Mozilla) devolvendo o resultado para o mesmo em arquivos no formato HTML.

O PHP [4] é uma linguagem de servidor, ou seja, o código fonte da página visitada está guardado em um servidor de publicação de páginas para a *Internet*. Quando um usuário acessa esse servidor, este lê o código PHP [4] correspondente à requisição do usuário e o processa de forma a criar uma página HTML dinamicamente. A figura II.24 demonstra como funciona o relacionamento entre o usuário, o servidor *Internet* e o seu módulo PHP [4].

Figura II.27 - Relacionamento para interpretação do PHP.



A sintaxe de PHP [4] é embutida no HTML por um rótulo especial que informa tanto para o usuário como para o servidor que aquele trecho de código é para ser interpretado pelo módulo PHP [4], ou seja, tratada no servidor *Internet*. A figura II.25 apresenta um exemplo de código em PHP [4].

```
<html>
<head>
<title>Exemplo</title>
</head>
<body>
<?
echo "Testando um script de PHP";
?>
</body>

</html>
```

Figura II.28 - Exemplo de sintaxe PHP

Capítulo III . Desenvolvimento

O processo de desenvolvimento do projeto pode ser dividido em quatro blocos: a criação do Modelo lógico do módulo, a Modelagem do banco de dados, a Interface em PHP [4] e a Alteração no banco de dados do SID [1] para comportar o novo módulo.

Cada uma das etapas, para finalidade didática, será tratada em uma sessão à parte, com exemplos de códigos e usos, com o objetivo de fornecer claro entendimento sobre os mecanismos utilizados para implementar o módulo como um todo.

III.1 . Modelo lógico do módulo

O primeiro passo para a construção do LegWeb é a definição do seu modelo conceitual. O módulo funcionará utilizado o modelo B2C (*business to consumer*), que trabalha com vários tipos de insumos: componentes eletrônicos, equipamentos, informações de curso, etc. Alunos e professores serão consumidores das funcionalidades do LEG, enquanto os administradores deverão controlar os procedimentos e as políticas que regem a oferta de serviços.

Para tanto, define-se que o módulo será dividido em quatro grandes áreas, a saber: **Área administrativa**, **Área de professores**, **Área de alunos** e **Área pública.**

A necessidade de quatro áreas funcionais distintas leva à criação de três tipos distintos de usuário, a saber: **Usuário identificado**, **Professor**, **Aluno** e **Administrador**.

Todos são usuários identificados pelo sistema (requerem autenticação). Tendo em vista as definições acima, a figura III.1 apresentará os tipos de usuários do módulo e a figura III.2 o relacionamento dos usuários com cada uma de suas áreas funcionais.

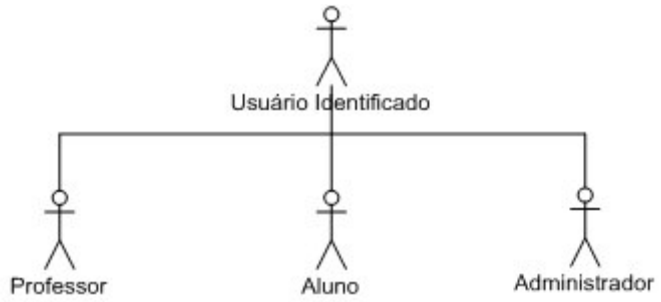


Figura III.1 - Tipos de usuário do site.

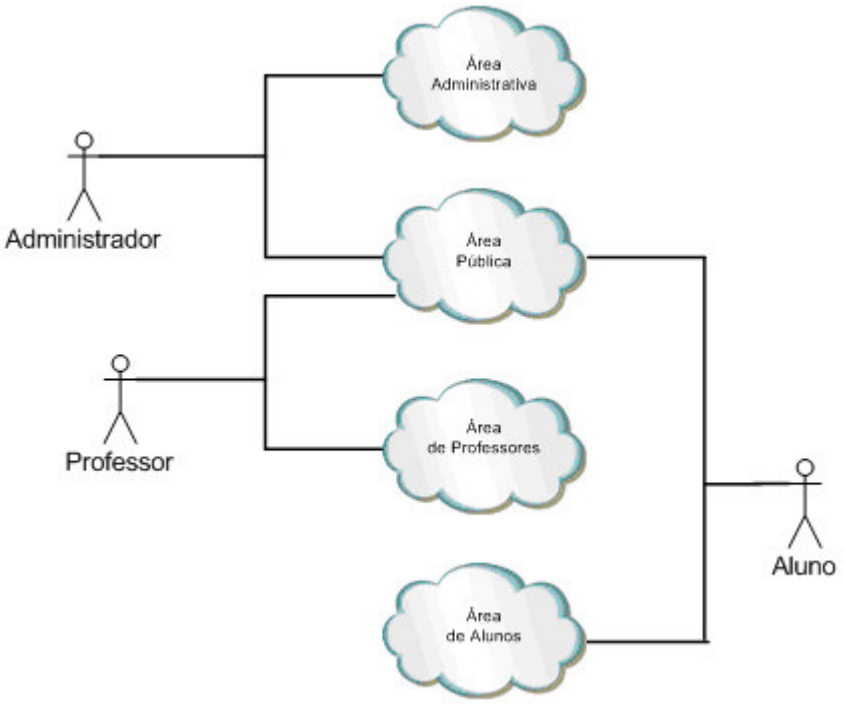


Figura III.2 - Áreas do site e seus usuários.

III.1.1 . Definição das áreas do LegWeb

Neste item definiremos as funcionalidades de cada área do módulo, bem como as atribuições e responsabilidades de cada usuário do módulo. Desta forma, poderemos mais adiante montar os diagramas de casos de uso do sistema com base nestas definições.

III.1.1.1 . Área administrativa

Este é o local onde estão dispostas as ferramentas de gestão do módulo LegWeb bem como das demandas de controle do próprio laboratório.

Usuários com privilégios administrativos podem executar tarefas relacionadas à:

- Gestão de acesso (inclusão, alteração ou exclusão) dos usuários do módulo;
- Gestão de ativos do laboratório (componentes e equipamentos) – aquisição, manutenção ou empréstimos;
- Gestão do sistema de mensagens do módulo;
- Gestão de fornecedores de ativos (compra e / ou manutenção);
- Gestão das notas das disciplinas do laboratório.

A figura III.3 apresenta as tarefas inerentes ao administrador do módulo e disponíveis somente na área administrativa:

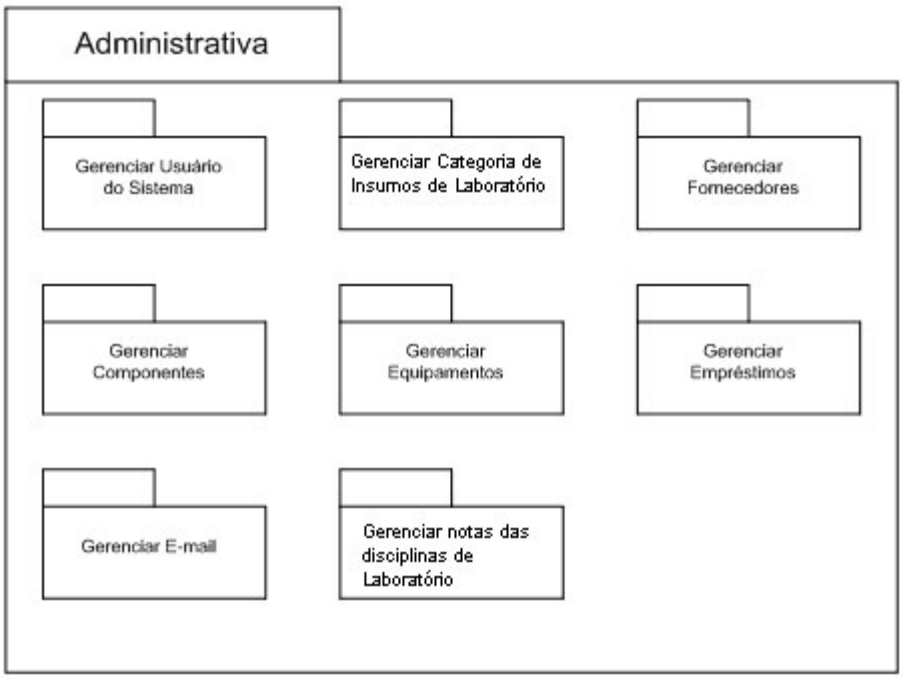


Figura III.3 - Pacote da área administrativa.

III.1.1.2 . Área de professores

Local onde são dispostas as ferramentas para a administração dos grupos de alunos por parte dos professores, bem como para o uso da infra-estrutura disponibilizada pelo laboratório.

Usuários com privilégios de um professor podem executar tarefas relacionadas à:

- Gerenciar notas das disciplinas do laboratório;
- Gerenciar empréstimos (solicitar) de insumos do laboratório (componentes, instrumentos, etc.);
- Solicitar manutenção de equipamentos;
- Solicitar compra de componentes.

A figura III.4 apresenta as tarefas inerentes aos professores cadastrados no módulo e disponíveis somente na área de professores:

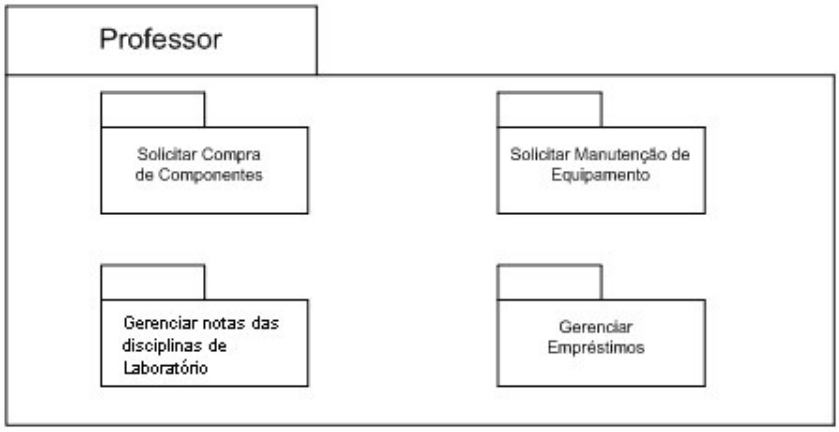


Figura III.4 - Pacote da área de professores.

III.1.1.3 . Área de alunos

Local onde são dispostas as ferramentas para o uso da infra-estrutura disponibilizada pelo laboratório por parte dos alunos bem como para o gerenciamento das notas das disciplinas de laboratório.

Usuários com privilégios de um aluno podem executar tarefas relacionadas à:

- Solicitar compra de componentes;
- Solicitar manutenção de equipamentos;

- Visualizar notas das disciplinas de laboratório do laboratório;

- Gerenciar empréstimos.

A figura III.5 apresenta as tarefas inerentes aos alunos cadastrados no módulo e disponíveis somente na área de alunos:

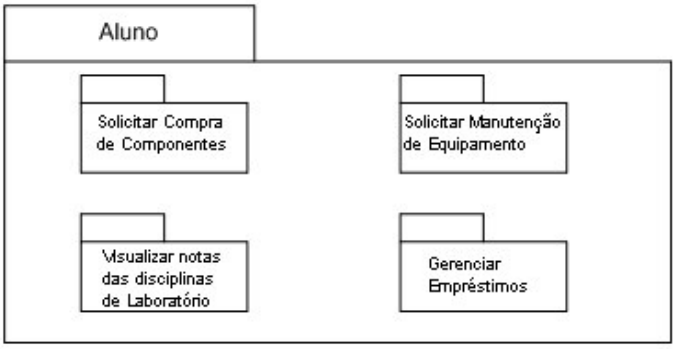


Figura III.5 - Pacote da área de alunos.

III.1.1.4 . Área pública

Local onde são dispostas as ferramentas de uso comum para todos os usuários do módulo. Usuários com privilégios de um aluno podem executar tarefas relacionadas à visualização e consulta da grade curricular, do horário de aulas, de material institucional e sobre a localização do laboratório. O usuário ainda terá ao seu dispor um serviço de mensagens (fale conosco) onde pode enviar comentários ou sugestões aos administradores do laboratório.

A figura III.6 apresenta as tarefas inerentes aos usuários cadastrados no módulo:

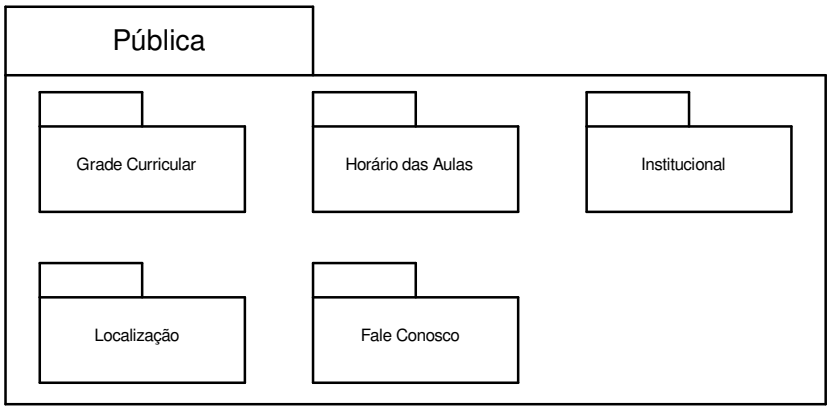


Figura III.6 - Pacote da área pública

III.1.2 . Casos de uso para usuários administradores

Nesta seção detalharemos as tarefas assinaladas na figura III.3 (pacote da área administrativa) em atividades mais detalhadas. Desta forma, teremos condições de montar os diagramas de casos de usos do sistema. Ao final, teremos transformado as rotinas de administração em um modelo UML.

III.1.2.1 . Gerenciar usuário do sistema

Estão disponíveis nesta subárea ferramentas que possibilitam ao administrador do sistema incluir, excluir, alterar ou visualizar usuários do LegWeb. A figura III.7 apresenta as três rotinas de gerenciamento de usuário às quais o administrador do módulo terá acesso: manutenção de grupo, de professor e de administrador.

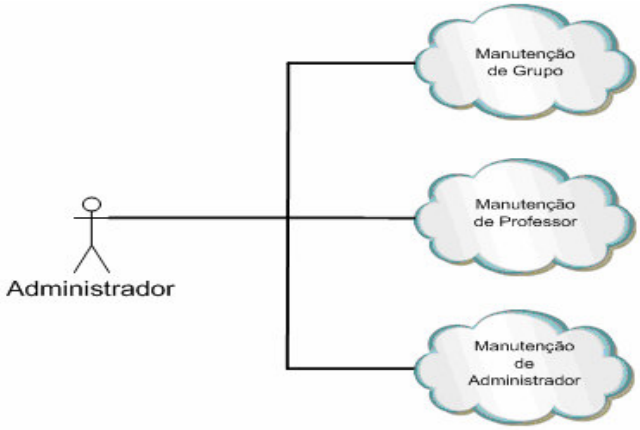


Figura III.7 – Rotinas de gerenciamento do usuário.

Na seqüência serão analisados os casos de uso das rotinas de manutenção acima apresentadas.

III.1.2.1.1 . Manutenção de grupo

Para esta rotina de manutenção, teremos as seguintes operações: Criar grupo, Visualizar grupo, Localizar grupo, Visualizar aluno, Alterar grupo, Excluir grupo e Selecionar grupo.

A figura III.8 apresenta os casos de uso para a rotina de manutenção de grupo. Note que os processos Visualizar grupo, Excluir grupo e Alterar grupo incluem o processo

Selecionar grupo, ou seja, antes de executar o processo principal (um dos três listados), o administrador deve primeiro selecionar o usuário desejado.

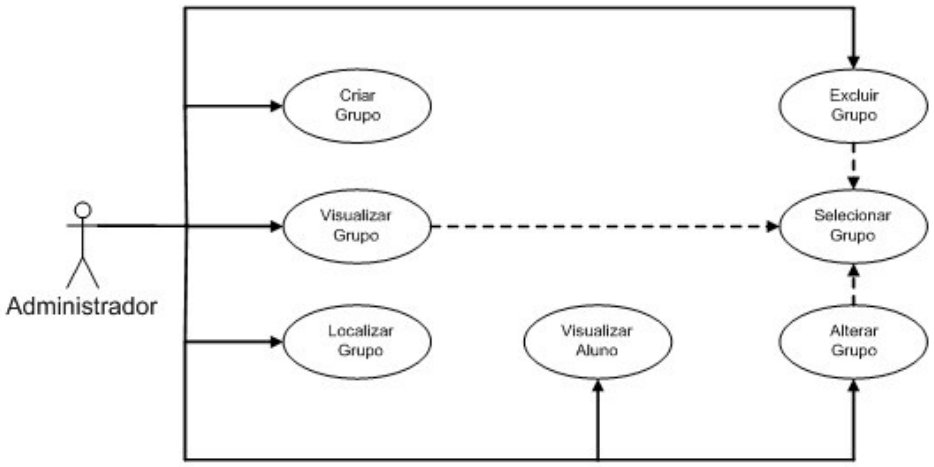


Figura III.8 - Casos de uso para manutenção de grupo

Dada a apresentação dos casos de uso para a rotina em questão, cabe transformar cada um dos processos em ações descritivas do sistema:

1. **Selecionar grupo:**

<u>Procedimento:</u>

a) Sistema exibe lista contendo os grupos já cadastrados.

b) Administrador seleciona grupo desejado e pressiona a opção de manutenção pretendida (visualizar, alterar ou excluir).

<u>Alternativas:</u>

a) Não existem grupos cadastrados.

➔ Sistema apresenta como resultado da busca por grupos uma lista vazia.

2. **Criar grupo:**

<u>Procedimento:</u>

a) Sistema exibe formulário para cadastro do novo grupo com os seguintes campos a preencher: *Login* dos componentes no SID [1], Nome do grupo, Turma e *Login* do professor no SID [1].

b) Administrador informa os dados e os submete para a validação do sistema.

c) Sistema verifica a consistência dos dados e cadastra o grupo.

d) Sistema concede permissões de acesso ao módulo para os alunos cadastrados no grupo bem como às informações referentes ao grupo ao qual pertencem.

e) Sistema exibe mensagem de confirmação do cadastro.

<u>Alternativas:</u>

a) Não existe o *login* de um ou mais componentes a cadastrar.

➔ Sistema exibe mensagem informando os *logins* inseridos que não estão cadastrados na base do SID [1] e não procede o cadastro.

b) Não existe o *login* do professor informado no SID [1].

➔ Sistema exibe mensagem informando que o *login* repassado não está cadastrado e não procede a transação.

c) O *login* informado como professor responsável pelo grupo, apesar de existir na base do SID [1], não foi cadastrado como professor no LegWeb.

➔ Sistema exibe mensagem informando que o *login* repassado não está cadastrado e não procede a transação.

d) Administrador tenta cadastrar grupo com um nome já atribuído a outro.

➔ Sistema exibe mensagem de erro informando a presença de uma chave duplicada (Nome de grupo) e não procede ao cadastro.

e) Administrador tenta cadastrar um ou mais alunos em mais de um grupo ao mesmo tempo.

➔ Sistema exibe mensagem de erro informando a presença de alunos já cadastrados em outros grupos na lista fornecida e não procede a alteração.

3. Visualizar grupo

Procedimento:

a) Administrador seleciona grupo conforme descrito em III.1.2.1.1 (1).

b) Após a seleção, o administrador pressiona "Visualizar grupo".

c) Sistema apresenta na tela as seguintes informações: *Login* dos componentes no SID [1], Nome do grupo, Turma e *Login* do professor no SID [1].

Alternativas:

a) Valem as alternativas descritas em III.1.2.1.1(1).

4. Alterar grupo

Procedimento:

a) Administrador seleciona grupo conforme descrito em III.1.2.1.1(1).

b) Após a seleção, o administrador pressiona "Alterar grupo".

c) Sistema apresenta formulário com os seguintes campos: *Login* dos componentes no SID [1], Nome do grupo, Turma, *Login* do professor no SID [1].

d) Administrador faz as alterações desejadas e pressiona "Alterar grupo".

e) Sistema verifica a consistência das alterações. Caso sejam coerentes, o grupo é recadastrado e seus novos componentes recebem as permissões devidas, enquanto os antigos têm seus direitos revogados.

f) Sistema exibe uma mensagem confirmando a alteração.

Alternativas:

a) Valem as alternativas descritas em III.1.2.1.1(1).

b) Valem as alternativas descritas em III.1.2.1.1(2).

5. Excluir grupo

Procedimento:

a) Administrador seleciona grupo conforme III.1.2.1.1(1).

b) Após a seleção, o administrador pressiona "Excluir grupo".

c) Sistema exibe mensagem solicitando confirmação de exclusão.

d) Após confirmação pelo administrador, sistema remove o grupo e seus usuários, revogando as permissões de acesso ao módulo LegWeb.

Alternativas:

a) Valem as alternativas descritas em III.1.2.1.1(1).

b) Administrador não confirma ordem de exclusão.

➔ Sistema redireciona o usuário para a página "Manutenção de grupo" , não procedendo à exclusão.

6. Localizar grupo

Procedimento:

a) Sistema exibe formulário requerendo o *login* do aluno do qual se desejam informações de cadastro no LegWeb.

b) Administrador informa o *login* do aluno ao sistema.

c) Sistema informa os seguintes dados referentes ao grupo onde o *login* está associado: *Login* dos componentes no SID [1], Nome do grupo, Turma e *Login* do professor no SID [1].

Alternativas:

a) Não existe grupo com o *login* informado atrelado.

➔ Sistema exibe mensagem informando que o *login* repassado não possui grupo associado.

b) Administrador informa um *login* que não existe no SID [1].

➔ Sistema exibe mensagem informando que o *login* informado não está cadastrado no SID [1].

7. Visualizar aluno

Procedimento:

a) Sistema apresenta um formulário requerendo o *login* do aluno.

b) Administrador informa *login* e envia informação ao sistema.

c) Sistema informa os seguintes dados associados ao *login*: Nome, *Status* no LegWeb e Matrícula.

Alternativas:

a) O *login* informado não existe na base do SID [1].

➔ Sistema exibe mensagem informando que o *login* inserido não está cadastrado na base do SID [1].

III.1.2.1.2 . Manutenção de professor

Para esta rotina de manutenção, teremos as seguintes operações: Criar professor, Visualizar professor, Excluir professor e Selecionar professor.

A figura III.9 apresenta os casos de uso para a rotina de manutenção de professor. Note que os processos Visualizar professor e Excluir professor incluem o processo Selecionar professor, ou seja, antes de executar o processo principal (um dos dois listados), o administrador deve primeiro selecionar o usuário desejado.

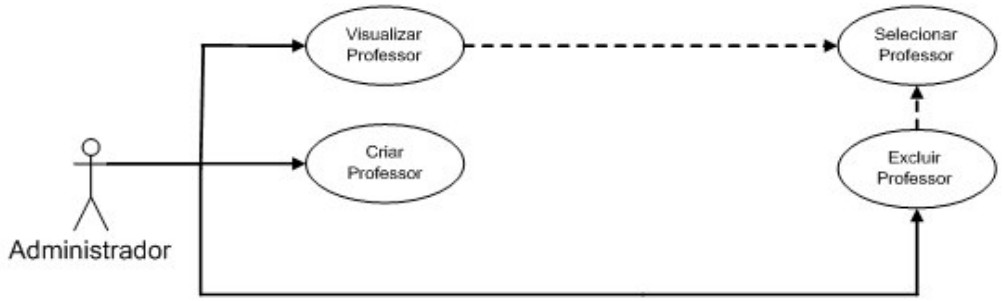


Figura III.9 - Casos de uso para manutenção de professor.

Dada a apresentação dos casos de uso para a rotina em questão, cabe transformar cada um dos processos em ações descritivas do sistema:

1. Selecionar professor

<u>Procedimento:</u>

a) Sistema exibe lista com os *logins* dos professores cadastrados.

b) Administrador seleciona professor desejado e escolhe a opção de manutenção adequada (Visualizar ou Excluir).

<u>Alternativas:</u>

a) Não existem professores cadastrados.

➔ Sistema apresenta como resultado da busca por professores cadastrados uma lista vazia.

2. Criar professor

Procedimento:

a) Sistema exibe formulário para cadastro de novo professor onde deve ser informado o seu *login* no SID [1].

b) Administrador informa dados e os submete à avaliação do sistema.

c) Sistema verifica a consistência dos dados e, caso sejam coerentes, cadastra o *login* como professor no módulo.

d) Sistema concede ao professor as permissões de acesso ao LegWeb e às informações das turmas a ele vinculadas.

e) Sistema exibe mensagem de confirmação de cadastro.

Alternativas:

a) Não existe o *login* do professor informado na base do SID [1].

➔ Sistema exibe mensagem informando que o *login* não está cadastrado na base do SID [1] e não procede ao cadastro.

3. Excluir professor

Procedimento:

a) Administrador seleciona professor conforme descrito em III.1.2.1.2.(1).

b) Após seleção, administrador pressiona "Excluir professor".

c) Sistema exibe mensagem solicitando a confirmação da exclusão.

d) Após a confirmação positiva pelo administrador, o sistema remove o cadastro do professor e suas permissões de acesso ao LegWeb.

Alternativas:

a) Valem as alternativas descritas em III.1.2.1.2.(1).

b) Administrador não confirma ordem de exclusão.

➔ Sistema redireciona o usuário para a página "Manutenção de professor" e não procede à exclusão.

4. Visualizar professor

Procedimento:

a) Administrador seleciona professor conforme descrito em III.1.2.1.2.(1).

b) Após a seleção, o administrador pressiona "Visualizar professor".

c) Sistema apresenta um formulário com as seguintes informações: *Login*, Nome, Quantidade de turmas e Turmas supervisionadas.

Alternativas:

a) Valem as alternativas descritas em III.1.2.1.2.(1).

b) O professor, apesar de cadastrado, não possui turmas a ele atreladas.

➔ Sistema retorna valor "zero" no campo "Quantidade de turmas" e retorna uma lista vazia no campo " Turmas supervisionadas".

III.1.2.1.3 . Manutenção de administrador

Para esta rotina de manutenção, teremos as seguintes operações: Criar administrador, Visualizar administrador, Excluir administrador e Selecionar administrador.

A figura III.10 apresenta os casos de uso para a rotina de manutenção de administrador. Note que os processos Visualizar administrador e Excluir administrador incluem o processo Selecionar administrador, ou seja, antes de executar o processo principal (um dos dois listados), o administrador deve primeiro selecionar o usuário desejado.

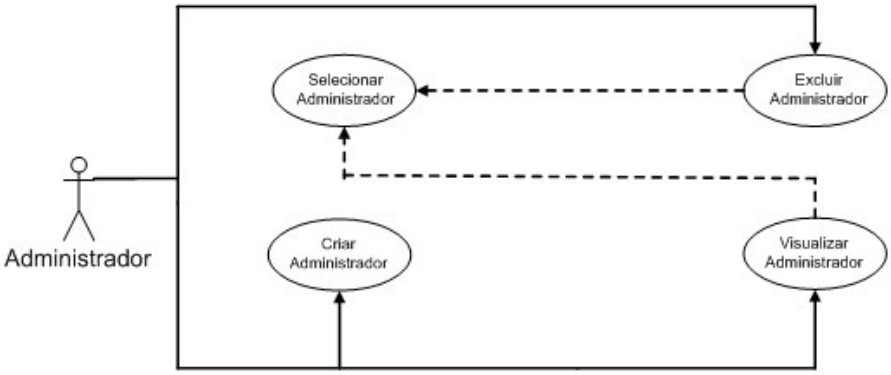


Figura III.10 - Casos de uso para manutenção de administrador.

Dada a apresentação dos casos de uso para a rotina em questão, cabe transformar cada um dos processos em ações descritivas do sistema:

1. **Selecionar administrador**

<u>Procedimento:</u>

a) Sistema exibe lista com os *logins* dos administradores cadastrados.

b) Administrador seleciona usuário desejado e escolhe a opção de manutenção adequada (Visualizar ou Excluir).

2. **Criar administrador**

<u>Procedimento:</u>

a) Sistema exibe formulário para cadastro de novo administrador com o campo "*Login* no SID" [1] a preencher:

b) Administrador informa dados e os submete à validação do sistema.

c) Sistema valida informações e cadastra o novo administrador com as permissões adequadas.

d) Sistema exibe mensagem de confirmação de cadastro.

Alternativas:

a) Não existe o *login* informado na base do SID [1].

➔ Sistema exibe mensagem informando que o *login* não está cadastrado no SID [1].

3. Excluir administrador

Procedimento:

a) Administrador seleciona usuário conforme descrito em III.1.2.1.3.(1).

b) Após seleção, administrador pressiona "Excluir administrador".

c) Sistema exibe mensagem solicitando confirmação de exclusão.

d) Após confirmação pelo administrador, sistema remove o usuário e revoga suas permissões de acesso ao módulo LegWeb.

Alternativas:

a) Administrador não confirma ordem de exclusão.

➔ Sistema redireciona o usuário para a página "Manutenção de administrador" e não procede à exclusão.

4. **Visualizar administrador**

Procedimento:

a) Administrador seleciona usuário conforme descrito em III.1.2.1.3.(1).

b) Após seleção, administrador pressiona "Visualizar administrador".

c) Sistema exibe formulário com as seguintes informações: *Login* e Nome do administrador.

III.1.2.2 . Gerenciar categoria de insumos de laboratório

Estão disponíveis nesta subárea ferramentas que possibilitam ao administrador gerenciar o catálogo de insumos ofertados pelo LEG. Entende-se por insumos de laboratório as facilidades ofertadas para empréstimo pelo laboratório que possam ser classificadas em duas categorias de materiais: componentes (como resistores e capacitores, por exemplo) e equipamentos (como fontes de alimentação e osciloscópios, por exemplo).

O administrador poderá criar os índices ou categorias onde alocará os componentes ou equipamentos que tem à sua disposição. A formação dos índices é feita com base em uma estrutura de árvore hierárquica, onde só é possível inserir componentes ou equipamentos efetivamente nos nós terminais da árvore (i.e., aqueles que não contém subcategorias abaixo dele). A figura III.11 apresenta o modelo de árvore utilizada na confecção deste documento.

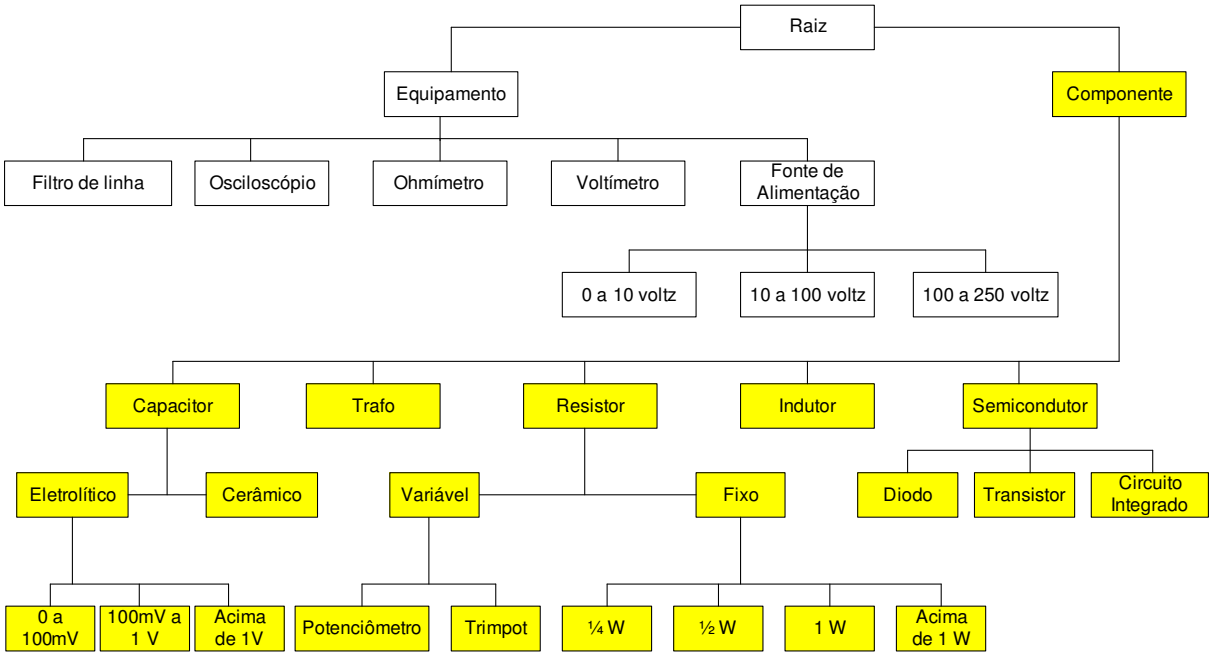


Figura III.11 - Exemplo de árvore de ativos.

Para esta rotina de manutenção, teremos as seguintes operações: Criar categoria, Visualizar categoria, Alterar categoria, Excluir categoria e Selecionar categoria.

A figura III.12 apresenta os casos de uso para a rotina de Gerenciamento de categoria de insumos de laboratório. Note que os processos Criar categoria, Visualizar categoria, Excluir categoria e Alterar categoria incluem o processo Selecionar categoria, ou seja, antes de executar o processo principal (um dos quatro listados), o administrador deve primeiro selecionar a categoria desejada.

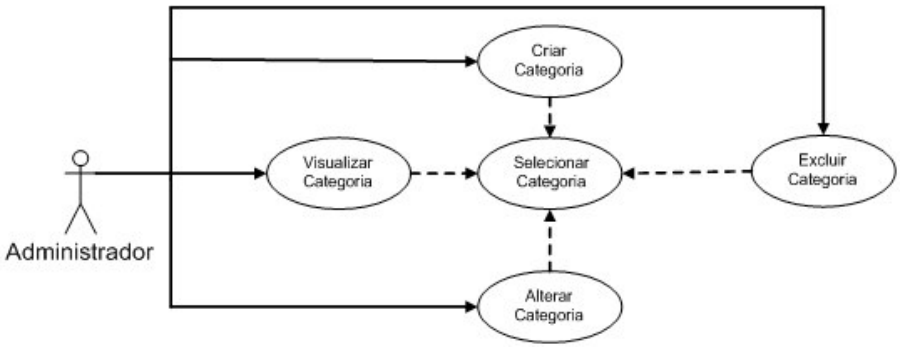


Figura III.12 - Casos de uso para gerenciar categoria.

1. Selecionar categoria

<u>Procedimento:</u>

a) Sistema exibe formulário apresentando as categorias raiz disponíveis no sistema (componente ou equipamento).

b) Administrador seleciona a categoria raiz desejada.

c) Sistema informa no campo "Categoria selecionada" a opção marcada no passo anterior e exibe no campo "Categorias derivadas" as subcategorias disponíveis no sistema.

d) Administrador seleciona a subcategoria desejada.

e) Sistema exibe as opções de manutenção (Criar, Alterar ou Excluir categoria).

Alternativas:

a) Administrador seleciona uma categoria ou subcategoria que é nó terminal da árvore cadastrada.

➔ Sistema informa que não existem ramificações abaixo da categoria e oferece a opção de retornar à categoria ou subcategoria pai imediatamente acima.

b) Administrador pressiona opção "Criar categoria" sem ter selecionado nenhuma categoria raiz.

➔ Sistema entende por padrão que a subcategoria a ser criada é ligada a categoria raiz "Componente" e a cadastra nesta posição.

2. Criar categoria

Procedimento:

a) Administrador seleciona a categoria ou subcategoria conforme descrito em III.1.2.2.(1) de forma a identificar o ponto da árvore em que deseja criar uma subcategoria.

b) Após seleção, administrador pressiona "Criar categoria".

c) Sistema exibe formulário solicitando o nome da nova subcategoria.

d) Administrador informa o nome e pressiona “Criar categoria”.

e) Sistema retorna mensagem confirmando a inclusão da subcategoria.

Alternativas:

a) Valem as alternativas descritas em III.1.2.2.(1).

b) Administrador deseja criar subcategoria com o mesmo nome de outra, estando elas abaixo da mesma categoria ou subcategoria pai.

➔ Sistema exibe mensagem de erro e informa que existe uma categoria com o mesmo nome.

3. Excluir categoria

Procedimento:

a) Administrador seleciona a categoria ou subcategoria conforme descrito em III.1.2.2.(1) de forma a identificar o ponto da árvore em que deseja fazer a exclusão.

b) Após a seleção, administrador pressiona “Excluir categoria”.

c) Sistema envia mensagem alertando que todas as subcategorias associadas serão excluídas, incluindo os ativos a ela associados (componentes ou equipamentos).

d) Após confirmação de exclusão pelo administrador, sistema remove a subcategoria selecionada.

Alternativas:

a) Valem as alternativas descritas em III.1.2.2.(1).

b) Administrador não confirma ordem de exclusão

➔ Sistema redireciona o usuário para o nível da categoria raiz.

4. Alterar categoria

Procedimento:

a) Administrador seleciona a categoria ou subcategoria conforme descrito em III.1.2.2.(1) de forma a identificar o ponto da árvore em que deseja realizar a alteração.

b) Após seleção, administrador pressiona "Alterar categoria".

c) Sistema apresenta formulário com os seguintes campos: Tipo (componente ou equipamento), Categoria pai (aquela imediatamente acima na estrutura de árvore), Categoria irmã (aquelas de mesmo nível hierárquico na estrutura de árvore) e Categoria selecionada (aquela que é alvo da alteração).

d) Administrador altera o nome da categoria ou subcategoria selecionada conforme a necessidade e pressiona o botão "Alterar".

Alternativas:

a) Valem as alternativas descritas em III.1.2.2.(1).

b) Administrador deseja alterar nome de subcategoria para um nome já utilizado abaixo da mesma categoria ou subcategoria pai.

➔ Sistema exibe mensagem de erro e informa que já existe subcategoria com o mesmo nome.

c) Administrador deseja alterar o nome das categorias raiz (componente e equipamento).

➔ Sistema não permite tal alteração.

III.1.2.3 . Gerenciar fornecedores

Estão disponíveis nesta subárea ferramentas que permitem ao administrador gerenciar os fornecedores de insumos do LEG bem como os prestadores de serviço. Para a adequada gestão é disponibilizado um cadastro dos fornecedores com todos os dados e informações úteis, como pessoa de contato, *e-mail*, telefone, etc.

Para esta rotina de manutenção, teremos as seguintes operações: Cadastrar fornecedor, Alterar fornecedor, Excluir fornecedor, Visualizar fornecedor e Selecionar fornecedor.

A figura III.13 apresenta os casos de uso para a rotina de gerenciamento de fornecedores. Note que os processos Alterar fornecedor, Excluir fornecedor e Visualizar fornecedor incluem o processo Selecionar fornecedor, ou seja, antes de executar o processo principal (um dos três listados), o administrador deve primeiro selecionar o fornecedor desejado.

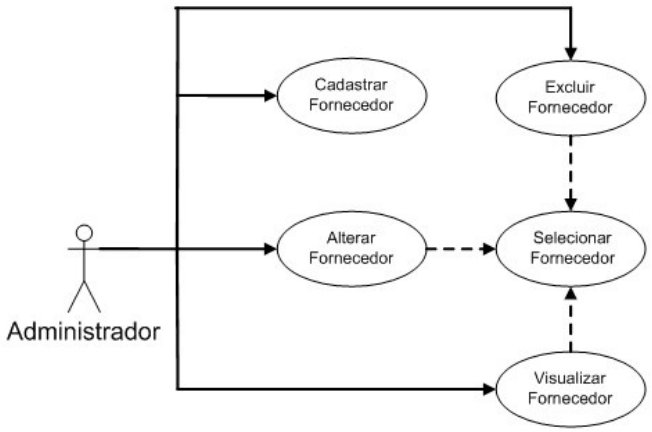


Figura III.13 - Casos de uso para gerenciar fornecedores.

Dada a apresentação dos casos de uso para a rotina em questão, cabe transformar cada um dos processos em ações descritivas do sistema:

1. **Selecionar fornecedor**

<u>Procedimento:</u>

a) Sistema exibe lista com a razão social das empresas cadastradas.

b) Administrador seleciona fornecedor desejado e escolhe a opção de manutenção adequada (Alterar, Excluir ou Visualizar).

Alternativas:

a) Não existem empresas cadastradas.

➔ Sistema apresenta como resultado da busca por empresas cadastradas uma lista vazia.

2. Cadastrar fornecedor

Procedimento:

a) Sistema exibe formulário para cadastro de fornecedor onde são solicitadas as seguintes informações: Razão social, CNPJ, Endereço, Número, Complemento, Bairro, Cidade, Estado, CEP, DDI, DDD, Telefone, Contato técnico, DDI, DDD, Telefone, DDI, DDD, Celular e Observações.

b) Administrador informa os dados e os submete para cadastro.

c) Sistema cadastra fornecedor no módulo e exibe mensagem de confirmação de cadastro.

Alternativas:

a) Os dados informados são inválidos.

➔ Sistema exibe uma mensagem de erro e não procede com a inclusão do fornecedor.

3. Excluir fornecedor

Procedimento:

a) Administrador seleciona fornecedor conforme descrito em III.1.2.3.(1).

b) Após seleção, administrador pressiona “Excluir fornecedor”.

c) Sistema exibe mensagem solicitando a confirmação de exclusão.

d) Administrador confirma exclusão.

e) Sistema remove o fornecedor da base do módulo.

Alternativas:

a) Valem as alternativas descritas em III.1.2.3.(1).

b) Administrador não confirma a ordem de exclusão.

➔ Sistema redireciona o usuário para "Manutenção de fornecedor" e não procede a exclusão.

4. Visualizar fornecedor

Procedimento:

a) Administrador seleciona fornecedor conforme descrito em III.1.2.3.(1).

b) Após seleção, administrador pressiona "Visualizar fornecedor".

c) Sistema apresenta um formulário com as seguintes informações: Razão social, CNPJ, Endereço, Número, Complemento, Bairro, Cidade, Estado, CEP, DDI, DDD, Telefone, Contato técnico, DDI, DDD, Telefone, DDI, DDD, Celular e Observações.

Alternativas:

a) Valem as alternativas descritas em III.1.2.3.(1).

5. Alterar fornecedor

Procedimento:

a) Administrador seleciona fornecedor conforme descrito em III.1.2.3.(1).

b) Após seleção, administrador pressiona “Alterar fornecedor”.

c) Sistema apresenta um formulário com as seguintes informações: Razão social, CNPJ, Endereço, Número, Complemento, Bairro, Cidade, Estado, CEP, DDI, DDD, Telefone, Contato técnico, DDI, DDD, Telefone, DDI, DDD, Celular e Observações.

d) Administrador altera os dados desejados e marca a opção “Alterar fornecedor”.

e) Sistema retorna mensagem conformando a alteração.

Alternativas:

a) Valem as alternativas descritas em III.1.2.3.(1).

III.1.2.4 . Gerenciar componentes

Esta é uma subárea de vital importância para o sistema, pois trata de uma das atividades principais deste módulo, que é controlar o fluxo de insumos no laboratório.

Para esta rotina de manutenção, teremos as seguintes operações: Excluir componente, Consultar componente, Criar componente, Alterar componente, Selecionar componente, Selecionar categoria e Selecionar fornecedor.

A figura III.14 apresenta os casos de uso para a rotina de gerenciamento de componentes. Note que alguns processos desta rotina incluem mais de um processo em sua execução. É o caso dos processos Excluir componente e Consultar componente, que antes de executarem, acionam dois outros processos, o Selecionar categoria e, em seguida, o Selecionar componente. De forma semelhante, o processo Criar componente aciona primeiramente o processo Selecionar categoria e em seguida o Selecionar fornecedor. Já o processo Alterar componente aciona três processos em sua execução. Em primeiro lugar, aciona o processo Selecionar categoria, em seguida o processo Selecionar componente e por último o Selecionar fornecedor.

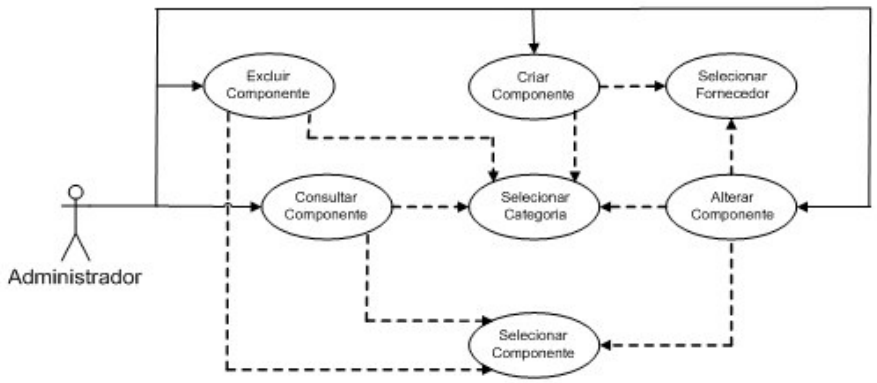


Figura III.14 - Casos de uso para gerenciar componentes.

Dada a apresentação dos casos de uso para a rotina em questão, cabe transformar cada um dos processos em ações descritivas do sistema:

1. Selecionar categoria

Procedimento:

a) Sistema exibe lista com as subcategorias cadastradas dentro da relação de componentes.

b) Administrador seleciona a subcategoria desejada e escolhe a opção de manutenção (Criar, Consultar, Alterar ou Excluir componente).

Alternativas:

a) Administrador seleciona uma subcategoria que é nó terminal da árvore cadastrada.

➔ Sistema informa que não existem ramificações abaixo da subcategoria e oferece a opção de retornar à categoria ou subcategoria pai diretamente acima.

2. Selecionar componente

Procedimento:

a) Sistema exibe lista com os componentes cadastrados em uma subcategoria terminal.

b) Administrador seleciona o componente desejado e escolhe a opção de manutenção (Consultar, Alterar ou Excluir componente).

Alternativas:

a) Não existem componentes cadastrados.

➔ Sistema apresenta como resultado da busca por componentes cadastrados uma lista vazia.

3. Selecionar fornecedor

Procedimento:

a) Sistema exibe uma lista com a razão social das empresas cadastradas no campo "Fornecedores disponíveis".

b) Administrador seleciona a empresa desejada e marca a opção "Adicionar".

c) Sistema apresenta a empresa escolhida no campo "Fornecedores selecionados" e a retira da lista apresentada em "Fornecedores disponíveis" para seleção em curso.

Alternativas:

a) Não existem componentes cadastrados.

➔ Sistema apresenta como resultado da busca por componentes cadastrados uma lista vazia.

4. Criar componente

Esta função deve ser utilizada quando do cadastro de um componente com especificações diferentes daqueles já cadastrados, respeitando-se os quesitos Valor nominal, Tolerância e Dimensões.

A simples aquisição para reposição de estoque, deve ser feita pela função "Alterar componente", descrita adiante.

Procedimento:

a) Administrador seleciona em qual subcategoria deseja criar o novo componente, conforme descrito em III.1.2.4.(1).

b) Após a seleção, o administrador pressiona "Criar componente".

c) Administrador seleciona o fornecedor ou fornecedores do novo componente, conforme descrito em III.1.2.4.(3).

d) Administrador preenche formulário para cadastro de componente, onde são solicitados os seguintes dados: Fornecedor selecionado, Descrição, Valor nominal, Tolerância, Dimensões, Quantidade e Data da compra.

e) Administrador pressiona o botão "Criar componente".

f) Sistema exibe mensagem confirmando a criação do componente.

Alternativas:

a) Valem as alternativas descritas em III.1.2.4.(1) e III.1.2.4.(3).

b) Administrador deseja criar um ou mais cadastros com a mesma descrição.

➔ Sistema exibe mensagem de erro informando que já existe outro componente com descrição idêntica.

5. Consultar componente

Procedimento:

a) Administrador seleciona em qual subcategoria está o componente que deseja consultar, conforme descrito em III.1.2.4.(1).

b) Após seleção e após alcançar um nó terminal, o administrador seleciona o componente que deseja consultar conforme descrito em III.1.2.4.(2).

c) Administrador pressiona a opção "Consultar componente".

d) Sistema exibe formulário contendo as seguintes informações: Descrição, Valor nominal, Tolerância, Dimensões, Quantidade ativa, Quantidade queimada, Quantidade mal conservada, Quantidade extraviada, Data da última compra, Quantidade da última compra e Fornecedores.

Alternativas:

a) Valem as alternativas descritas em III.1.2.4.(1) e III.1.2.4.(2).

6. Excluir componente

Procedimento:

a) Administrador seleciona em qual subcategoria está o componente que deseja excluir, conforme descrito em III.1.2.4.(1).

b) Após seleção e após alcançar um nó terminal, o administrador seleciona o componente que deseja excluir conforme descrito em III.1.2.4.(2).

c) Administrador pressiona a opção "Excluir componente".

d) Sistema exibe mensagem solicitando confirmação de exclusão para o componente selecionado.

e) Após confirmação de exclusão pelo administrador, o sistema exclui o componente.

Alternativas:

a) Valem as alternativas descritas em III.1.2.4.(1) e III.1.2.4.(2).

b) Administrador não confirma a ordem de exclusão.

➔ Sistema redireciona o usuário para o início da rotina de Gerência de componentes e não procede a exclusão.

7. Alterar componente

Esta função deve ser utilizada quando da necessidade de inclusão de componentes por reposição de estoque, baixa de componentes queimados, mal-conservados ou extraviados, alteração do histórico de fornecedores do componente ou associar fornecedores ao componente selecionado.

Qualquer alteração feita por esta função tem impacto direto no controle de estoque, visto que permite alterar o campo "Quantidade ativa" (visível pela função "Visualizar componente"), diminuindo seu valor (aumentando o valor dos campos "Quantidade queimada", "Quantidade mal-conservada" ou "Quantidade extraviada") ou aumentando o mesmo (aumentando o valor do campo "Quantidade da última compra").

<u>Procedimento:</u>

a) Administrador seleciona em qual subcategoria está o componente que deseja alterar, conforme descrito em III.1.2.4.(1).

b) Após seleção e após alcançar um nó terminal, o administrador seleciona o componente que deseja alterar conforme descrito em III.1.2.4.(2).

c) Administrador seleciona o fornecedor ou fornecedores do componente, conforme descrito em III.1.2.4.(3).

d) Sistema exibe formulário onde são apresentados os seguintes campos passíveis de alteração: Fornecedores selecionados, Fornecedores disponíveis, Quantidade queimada, Quantidade mal-conservada, Quantidade extraviada, Data da última compra e Quantidade da última compra.

e) Administrador insere dados relevantes e pressiona “Alterar componente”.

f) Sistema exibe mensagem confirmando a alteração do componente.

Alternativas:

a) Valem as alternativas descritas em III.1.2.4.(1) e III.1.2.4.(2) e III.1.2.4.(3).

b) O somatório de componentes informados em “Quantidade queimada”, “Quantidade mal-conservada” e “Quantidade extraviada” é superior às aquisições realizadas.

➔ Sistema apresenta o valor zero no item “Quantidade ativa” apresentada pela função “Consultar componente” e não trata a inconsistência.

III.1.2.5 . Gerenciar equipamentos

Esta é uma subárea de vital importância para o sistema, pois trata de uma das atividades principais deste módulo que é controlar o fluxo de insumos no laboratório.

Para esta rotina de manutenção, teremos as seguintes operações: Excluir equipamento, Consultar equipamento, Criar equipamento, Manutenção de equipamento, Selecionar equipamento, Selecionar categoria e Selecionar fornecedor.

A figura III.15 apresenta os casos de uso para a rotina de gerenciamento de componentes. Note que alguns processos desta rotina incluem mais de um processo em sua execução. É o caso dos processos Excluir equipamento e Consultar equipamento, que antes de executarem, acionam dois outros processos, o Selecionar categoria e, em seguida, o Selecionar equipamento. De forma semelhante, o processo Criar equipamento aciona primeiramente o processo Selecionar categoria e em seguida o Selecionar fornecedor. Por fim, o processo Manutenção de equipamento aciona outros dois processos em sua execução: em primeiro lugar, aciona o processo Selecionar categoria, em seguida o processo Selecionar equipamento.

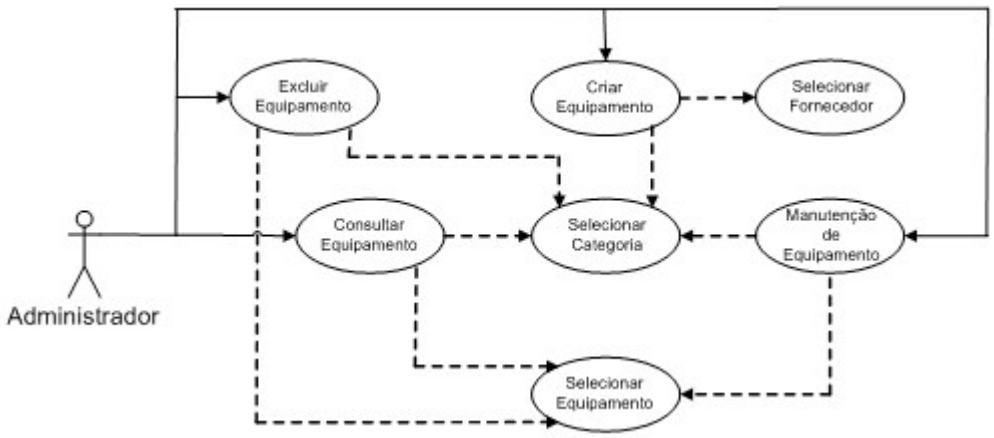


Figura III.15 - Casos de uso para gerenciar equipamentos.

Dada a apresentação dos casos de uso para a rotina em questão, cabe transformar cada um dos processos em ações descritivas do sistema:

1. Selecionar categoria

Procedimento:

a) Sistema exibe lista com as subcategorias cadastradas dentro da relação de equipamentos.

b) Administrador seleciona a subcategoria desejada e escolhe a opção de manutenção (Criar, Consultar, Excluir ou realizar Manutenção de equipamento).

Alternativas:

a) Administrador seleciona uma subcategoria que é nó terminal da árvore cadastrada.

➔ Sistema informa que não existem ramificações abaixo da subcategoria e oferece a opção de retornar à categoria ou subcategoria pai diretamente acima.

2. Selecionar equipamento

Procedimento:

a) Sistema exibe lista com os equipamentos cadastrados em uma subcategoria terminal.

b) Administrador seleciona o equipamento desejado e escolhe a opção de manutenção (Consultar, Excluir ou realizar Manutenção de equipamento).

Alternativas:

a) Não existem equipamentos cadastrados.

➔ Sistema apresenta como resultado da busca por equipamentos cadastrados uma lista vazia.

3. Selecionar fornecedor

Neste módulo o fornecedor engloba tanto as empresas que vendem equipamentos ao laboratório como aquelas que prestam serviços ao mesmo. Contudo, no banco de dados foram tratados como entidades distintas de forma a flexibilizar uma eventual separação do formato de cadastro de fornecedores e prestadores de serviço.

Procedimento:

a) Sistema exibe uma lista com a razão social das empresas cadastradas no campo "Fornecedores disponíveis".

b) Administrador seleciona a empresa desejada e marca a opção "Adicionar".

c) Sistema apresenta a empresa escolhida no campo "Fornecedores selecionados" e a retira da lista apresentada em "Fornecedores disponíveis" para seleção em curso.

Alternativas:

a) Não existem empresas cadastradas.

➔ Sistema apresenta como resultado da busca por equipamentos cadastrados uma lista vazia.

b) Administrador deseja remover a empresa da lista de “Fornecedores selecionados” de um dado equipamento.

➔ Quando da seleção de um ou mais fornecedores para um dado equipamento, o sistema apresenta a opção de remoção de fornecedor. O administrador descredencia uma empresa cadastrada como fornecedora de um dado equipamento pressionando o botão “Remover”.

4. Criar equipamento

Procedimento:

a) Administrador seleciona em qual subcategoria deseja criar o novo equipamento, conforme descrito em III.1.2.5.(1).

b) Após a seleção, o administrador pressiona “Criar equipamento”.

c) Administrador seleciona o fornecedor ou fornecedores do novo equipamento, conforme descrito em III.1.2.5.(3).

d) Administrador preenche formulário para cadastro de equipamento, onde são solicitados os seguintes dados: Nome do equipamento, Número do inventário, Fabricante, Valor da compra, Término da garantia, Dimensões e Data da aquisição

e) Administrador pressiona o botão “Criar equipamento”.

f) Sistema exibe mensagem conformando a criação do equipamento.

Alternativas:

a) Valem as alternativas descritas em III.1.2.5.(1) e III.1.2.5.(3).

b) Administrador deseja criar um ou mais cadastros com a mesma descrição.

➔ Sistema exibe mensagem de erro informando que já existe outro equipamento com descrição idêntica.

5. Consultar equipamento

Procedimento:

a) Administrador seleciona em qual subcategoria está o equipamento que deseja consultar, conforme descrito em III.1.2.5.(1).

b) Após seleção e após alcançar um nó terminal, o administrador seleciona o equipamento que deseja consultar conforme descrito em III.1.2.5.(2).

c) Administrador pressiona a opção "Consultar equipamento".

d) Sistema exibe formulário contendo os seguintes dados: Nome, Valor da compra, Número do inventário, Fabricante, Data da aquisição, Dimensões, Data da última manutenção, Total de manutenções, Em manutenção, Valor da manutenção, Descrição do último defeito e Fornecedores.

Alternativas:

a) Valem as alternativas descritas em III.1.2.5.(1) e III.1.2.5.(2).

6. Excluir equipamento

Procedimento:

a) Administrador seleciona em qual subcategoria está o equipamento que deseja excluir, conforme descrito em III.1.2.5.(1).

b) Após seleção e após alcançar um nó terminal, o administrador seleciona o equipamento que deseja excluir conforme descrito em III.1.2.5.(2).

c) Administrador pressiona a opção "Excluir equipamento".

d) Sistema exibe mensagem solicitando conformação de exclusão para o equipamento selecionado.

e) Após confirmação de exclusão pelo administrador, o sistema exclui o equipamento.

Alternativas:

a) Valem as alternativas descritas em III.1.2.5.(1) e III.1.2.5.(2).

b) Administrador não confirma a ordem de exclusão.

➔ Sistema redireciona o usuário para "Gerência de equipamento" e não procede a exclusão.

7. Manutenção de equipamento

Para esta sub-rotina de manutenção, teremos as seguintes operações: Abrir chamado, Fechar chamado, Visualizar chamado, Alterar chamado, Selecionar chamado e Selecionar responsável pela manutenção.

A figura III.16 apresenta os casos de uso para a sub-rotina de manutenção de equipamento. Note que alguns processos desta sub-rotina incluem mais de um processo em sua execução. É o caso dos processos Visualizar chamado, Alterar chamado e Fechar chamado, que antes de executarem, acionam o processo Selecionar chamado. De forma semelhante, o processo Visualizar chamado inclui o processo Selecionar responsável pela manutenção.

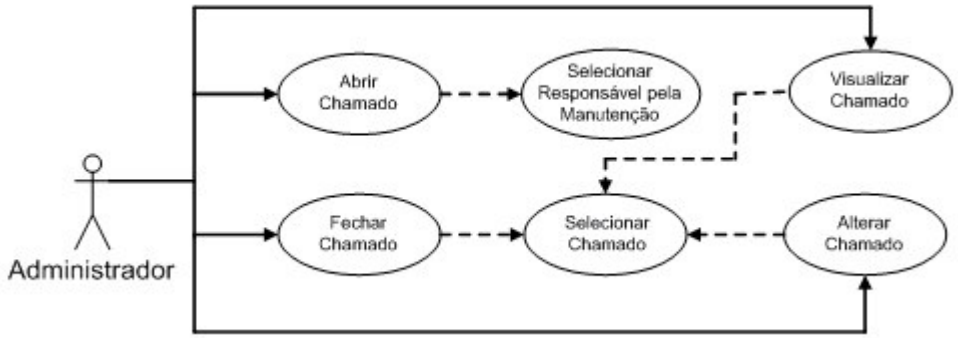


Figura III.16 - Casos de uso para manutenção de equipamento.

Dada a apresentação dos casos de uso para a sub-rotina em questão, cabe transformar cada um dos processos em ações descritivas do sistema:

i. Selecionar chamado

Procedimento:

a) Administrador clica em "Manutenção de equipamento".

b) Sistema exibe lista com os equipamentos em manutenção.

c) Administrador seleciona o equipamento desejado e escolhe a opção de manutenção desejada (Visualizar, Alterar ou Fechar chamado).

Alternativas:

a) Não existem manutenções cadastradas.

➔ Sistema apresenta como resultado da busca por equipamentos em manutenção uma lista vazia.

ii. Selecionar responsável pela manutenção

Procedimento:

a) Sistema exibe lista com os responsáveis pela manutenção.

b) Administrador seleciona o responsável pela manutenção.

Alternativas:

a) Não existem responsáveis cadastrados.

➔ Sistema apresenta como resultado da busca por responsáveis uma lista vazia.

iii. Abrir chamado

Procedimento:

a) Administrador pressiona o botão “Abrir chamado”.

b) Administrador seleciona o responsável pela manutenção conforme descrito em III.1.2.5.(7).(ii).

c) Administrador preenche o restante do formulário, informando os seguintes dados: Número do inventário, Descrição do defeito, Data de abertura do chamado (no formato *AA/MM/DD*), Responsável pela manutenção, Valor orçado para a manutenção, Previsão de término (no formato *AA/MM/DD*).

d) Administrador submete os dados pressionando o botão “Enviar”.

e) Sistema exibe mensagem confirmando o cadastro.

Alternativas:

a) Não existem equipamentos cadastrados.

➔ Sistema informa que não existem equipamentos cadastrados com o número de inventário fornecido.

b) Número de inventário fornecido pelo administrador não existe no cadastro do inventário.

➔ Sistema informa que não existem equipamentos cadastrados com o número de inventário fornecido.

c) Administrador tenta abrir chamado de manutenção para equipamento que já esteja em manutenção.

➔ Sistema informa que o equipamento já tem chamado em aberto.

d) Valem as alternativas descritas em III.1.2.5.(7).(ii).

iv. Visualizar chamado

Procedimento:

a) Administrador seleciona chamado conforme descrito em III.1.2.5.(7).(i).

b) Administrador pressiona "Visualizar chamado".

c) Sistema exibe formulário contendo as seguintes informações: Número do inventário, Descrição do defeito, Data de abertura do chamado (no formato *AA/MM/DD*), Responsável pela manutenção, Valor orçado para a manutenção, Previsão de término (no formato *AA/MM/DD*).

Alternativas:

a) Valem as alternativas descritas em III.1.2.5.(7).(i).

v. Alterar chamado

Procedimento:

a) Administrador seleciona chamado conforme descrito em III.1.2.5.(7).(i).

b) Administrador pressiona "Alterar chamado".

c) Sistema exibe formulário contendo as seguintes informações: Número do inventário, Descrição do defeito, Responsável pela manutenção, Valor orçado para a manutenção, Previsão de término (no formato *AA/MM/DD*).

d) Administrador altera os campos necessários e pressiona "Alterar".

e) Sistema salva as alterações e exibe mensagem confirmando a alteração.

Alternativas:

a) Valem as alternativas descritas em III.1.2.5.(7).(i).

vi. Fechar chamado

Procedimento:

a) Administrador seleciona chamado conforme descrito em III.1.2.5.(7).(i).

b) Administrador pressiona “Fechar chamado”.

c) Sistema exibe mensagem solicitando confirmação de exclusão para o chamado selecionado.

d) Após confirmação de exclusão pelo administrador, o sistema exclui o chamado.

Alternativas:

a) Valem as alternativas descritas em III.1.2.5.(7).(i).

b) Administrador não confirma a ordem de exclusão.

➔ Sistema redireciona o usuário para “Gerência de equipamento” e não procede a exclusão.

III.1.2.6 . Gerenciar empréstimos

Para esta rotina de manutenção, teremos as seguintes operações: Apagar empréstimo, Alterar empréstimo, Criar empréstimo, Selecionar empréstimo e Selecionar tipo de empréstimo.

A figura III.17 apresenta os casos de uso para a rotina de gerenciamento de empréstimos. Note que os processos Apagar empréstimo e Alterar empréstimo incluem o processo Selecionar empréstimo, que por sua vez inclui outro processo, o Selecionar tipo de empréstimo. Da mesma forma, o processo Criar empréstimo inclui o processo Selecionar tipo de empréstimo em sua execução.

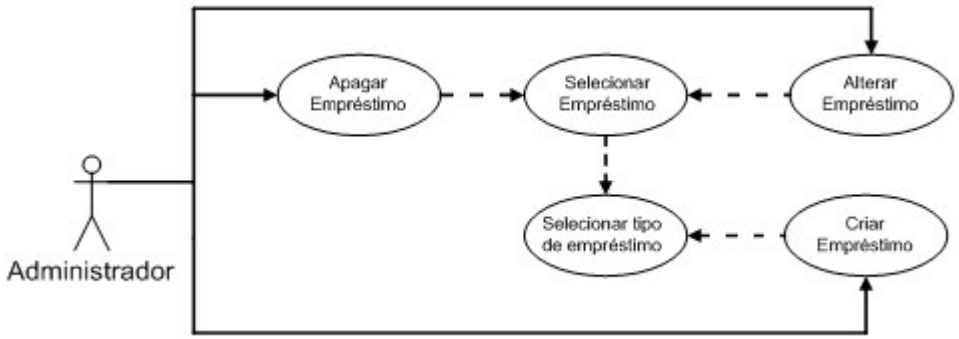


Figura III.17 - Casos de uso para gerenciar empréstimos.

Dada a apresentação dos casos de uso para a rotina em questão, cabe transformar cada um dos processos em ações descritivas do sistema:

1. Selecionar tipo de empréstimo

Procedimento:

a) Sistema apresenta formulário com a opção de escolha entre "Componente" e "Equipamento".

b) Sistema solicita informar o *login* no SID [1] do aluno para quem será feita a atividade de manutenção.

c) Administrador informa o *login* e pressiona a opção de monitoração desejada ("Componente" ou "Equipamento").

Alternativas:

a) O *login* informado não está cadastrado como usuário do módulo.

➔ Sistema informa que não existem ativos emprestados para este *login*.

b) O *login* informado não existe no SID [1].

➔ Sistema informa que não existem ativos emprestados para este *login*.

2. Selecionar empréstimo

Procedimento:

a) Administrador seleciona o tipo de empréstimo ("Componente" ou "Equipamento") conforme descrito em III.1.2.6.(1).

b) Sistema apresenta cadastro com os códigos de empréstimos associados ao *login* informado.

c) Administrador seleciona o código desejado e pressiona a opção de manutenção necessária ("Alterar empréstimo" ou "Apagar empréstimo").

Alternativas:

a) Não existem empréstimos associados ao *login* informado.

➔ Sistema exibe mensagem informando não haver empréstimos cadastrados para o *login*.

b) Valem as alternativas descritas em III.1.2.6.(1).

3. Criar empréstimo

Procedimento:

a) Administrador seleciona o tipo de empréstimo ("Componente" ou "Equipamento") conforme descrito em III.1.2.6.(1).

b) Administrador pressiona "Criar empréstimo".

c) Sistema exibe lista com os ativos disponíveis para empréstimo.

d) Administrador seleciona o ativo e informa a quantidade a ser emprestada.

e) Administrador pressiona "Criar empréstimo".

f) Sistema exibe mensagem informando que o empréstimo foi incluído com sucesso.

Alternativas:

a) Não existem ativos cadastrados.

➔ Sistema exibe como resultado da busca por ativos uma lista vazia.

b) Valem as alternativas descritas em III.1.2.6.(1).

4. Alterar Empréstimo

Procedimento:

a) Administrador seleciona o tipo de empréstimo ("Componente" ou "Equipamento") conforme descrito em III.1.2.6.(1).

b) Administrador seleciona o empréstimo conforme descrito em III.1.2.6.(2).

c) Sistema exibe formulário com as seguintes informações: Componente/ Equipamento, Quantidade, Data, *Status.*

d) Administrador pressiona "Alterar empréstimo".

e) Sistema exibe mensagem informando que o empréstimo foi atualizado com sucesso.

Alternativas:

a) Valem as alternativas descritas em III.1.2.6.(1) e III.1.2.6.(2).

5. Apagar empréstimo

Procedimento:

a) Administrador seleciona o tipo de empréstimo ("Componente" ou "Equipamento") conforme descrito em III.1.2.6.(1).

b) Administrador seleciona o empréstimo conforme descrito em III.1.2.6.(2).

c) Administrador pressiona a opção “Apagar empréstimo”.

d) Sistema exibe mensagem solicitando confirmação de exclusão para o empréstimo selecionado.

e) Após confirmação de exclusão pelo administrador, o sistema exclui o empréstimo.

Alternativas:

a) Valem as alternativas descritas em III.1.2.6.(1) e III.1.2.6.(2).

b) Administrador não confirma a ordem de exclusão.

➔ Sistema redireciona o usuário para “Gerência de empréstimo” e não procede a exclusão.

III.1.2.7 . Gerenciar *E-mail*

Para esta rotina de manutenção, teremos as seguintes operações: Ler mensagem, Selecionar mensagem, Responder mensagem e Excluir mensagem.

A figura III.18 apresenta os casos de uso para a rotina de gerenciamento de *e-mail*. Note que os processos Ler mensagem, Responder mensagem e Excluir mensagem incluem o processo Selecionar mensagem, ou seja, antes de executar o processo principal (um dos três listados), o administrador deve primeiro selecionar a mensagem desejada.

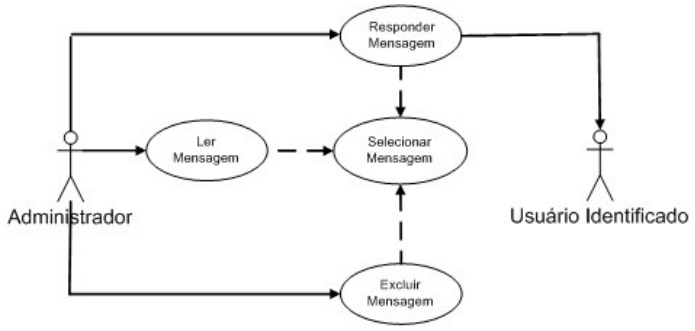


Figura III.18 - Casos de uso para gerenciar *e-mail*.

Dada a apresentação dos casos de uso para a rotina em questão, cabe transformar cada um dos processos em ações descritivas do sistema:

1. Selecionar mensagem

Procedimento:

a) Sistema exibe lista de mensagens armazenadas.

b) Administrador seleciona uma das mensagens.

Alternativas:

a) Não existem mensagens.

➔ Sistema apresenta uma lista vazia como resultado da busca por mensagens.

2. Ler mensagem

Procedimento:

a) Administrador seleciona mensagem conforme descrito em III.1.2.7.(1).

b) Administrador lê a mensagem e escolhe as opções de tratamento da mesma.

Alternativas:

a) Valem as alternativas descritas em III.1.2.7.(1).

b) A mensagem refere-se a um pedido de compra de material.

➔ Dentre as opções de tratamento de mensagens, exclui-se a possibilidade de resposta à mesma.

3. Excluir mensagem

Procedimento:

a) Administrador seleciona mensagem conforme descrito em III.1.2.7.(1).

b) Administrador pressiona o botão “Excluir mensagem”.

c) Sistema exibe mensagem de confirmação de exclusão de mensagem para o administrador.

d) Após confirmação de exclusão pelo administrador, o sistema exclui a mensagem.

Alternativas:

a) Valem as alternativas descritas em III.1.2.7.(1).

b) Administrador não confirma a ordem de exclusão.

➔ Sistema redireciona o usuário para “Selecionar mensagem” e não procede a exclusão.

4. Responder mensagem

Procedimento:

a) Administrador seleciona mensagem conforme descrito em III.1.2.7.(1).

b) Administrador marca o endereço de correio eletrônico do remetente.

c) A ferramenta de correio eletrônico padrão da estação de trabalho do administrador abrirá uma mensagem de resposta para preenchimento.

d) Administrador envia a mensagem de resposta através da sua ferramenta de correio eletrônico.

Alternativas:

a) Valem as alternativas descritas em III.1.2.7.(1).

III.1.2.8 . Gerenciar notas das disciplinas de laboratório

Para esta rotina de manutenção, teremos as seguintes operações: Informar graus, Alterar graus, Visualizar graus e Selecionar grupos.

A figura III.19 apresenta os casos de uso para a rotina de gerenciamento de notas das disciplinas de laboratório. Note que os processos Informar graus, Alterar graus e Visualizar graus incluem o processo Selecionar grupos, ou seja, antes de executar o processo principal (um dos três listados), o administrador deve primeiro selecionar o grupo desejado.

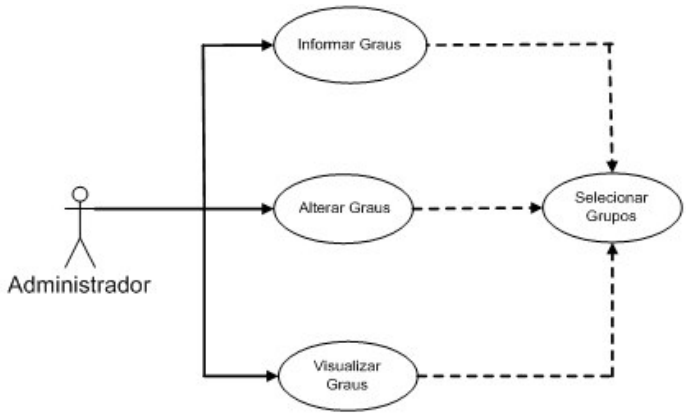


Figura III.19 - Casos de uso para gerenciar notas das disciplinas de laboratório.

Dada a apresentação dos casos de uso para a rotina em questão, cabe transformar cada um dos processos em ações descritivas do sistema:

1. **Selecionar grupos**

<u>Procedimento:</u>

a) Sistema exibe lista dos grupos existentes.

b) Administrador pressiona a tecla de manutenção desejada ao lado do grupo escolhido (“Informar”, “Alterar” ou “Visualizar” graus).

<u>Alternativas:</u>

a) Não existe grupo cadastrado.

➔ Sistema apresenta uma lista vazia como resultado pela busca de grupos cadastrados.

2. Informar graus

Procedimento:

a) Administrador seleciona o grupo conforme descrito em III.1.2.8.(1).

b) Sistema exibe formulário para cadastro de notas. As notas podem ser atribuídas individualmente para cada membro do grupo, num total de quinze valores.

c) Administrador pressiona "Informar graus", preenche formulário e envia para processamento, pressionando o botão "Confirmar".

d) Sistema armazena os dados e retorna à tela de "Gerência de graus".

Alternativas:

a) Valem as alternativas descritas em III.1.2.8.(1).

3. Alterar graus

Procedimento:

a) Administrador seleciona o grupo conforme descrito em III.1.2.8.(1).

b) Sistema exibe formulário para alteração de notas. As notas podem ser alteradas individualmente para cada membro do grupo, num total de quinze valores.

c) Administrador pressiona "Informar graus", preenche formulário e envia para processamento, pressionando o botão "Confirmar".

d) Sistema armazena os dados e retorna à tela de "Gerência de graus".

Alternativas:

a) Valem as alternativas descritas em III.1.2.8.(1).

4. Visualizar graus

Procedimento:

a) Administrador seleciona o grupo conforme descrito em III.1.2.8.(1).

b) Sistema exibe formulário com a relação de notas do grupo. As notas terão sido atribuídas individualmente para cada membro do grupo, num total de quinze valores.

Alternativas:

a) Valem as alternativas descritas em III.1.2.8.(1).

III.1.3 . Casos de uso para usuários professores

Nesta seção detalharemos as tarefas assinaladas na figura III.4 (pacote da área de professores) em atividades mais detalhadas. Desta forma, teremos condições de montar os diagramas de casos de usos do sistema. Ao final, teremos transformado as rotinas de administração em um modelo UML.

III.1.3.1 . Solicitar compra de componentes

Para esta rotina de manutenção, teremos as seguintes operações: Solicitar compra, Registrar data do pedido e Selecionar administrador.

A figura III.20 apresenta os casos de uso para a rotina de solicitação de compra de componentes. Note que o processo Solicitar compra inclui dois outros processos, o Registrar data do pedido e o Selecionar administrador, ou seja, antes de executar o processo principal, o professor deve primeiro registrar a data do pedido e depois selecionar para qual administrador ele encaminhará o pedido.

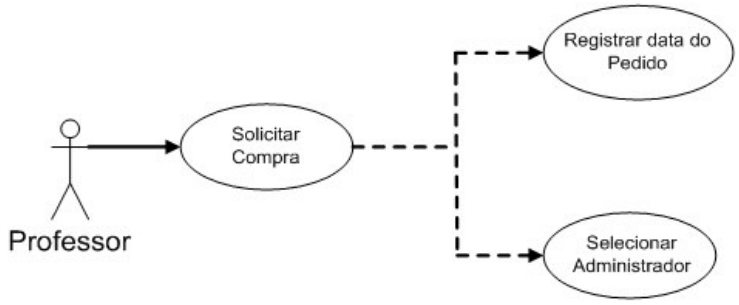


Figura III.20 - Casos de uso para solicitar compra de componentes.

Dada a apresentação dos casos de uso para a rotina em questão, cabe transformar cada um dos processos em ações descritivas do sistema:

1. Selecionar administrador

Procedimento:

a) Sistema exibe lista com administradores cadastrados.

b) Professor seleciona o administrador que receberá o pedido.

Alternativa: não há alternativa para este procedimento.

2. Registrar data do pedido

Procedimento:

a) Sistema registra a data do cadastro do pedido de acordo com a data e a hora do servidor que hospeda o módulo e anexa ao pedido.

Alternativa: não há alternativa para este procedimento.

3. Solicitar compra

Procedimento:

a) Sistema exibe formulário para solicitação de compra de componentes com as seguintes informações a preencher: *E-mail*, Data, Descrição do material, Quantidade e Motivação da compra

b) Professor seleciona o administrador que receberá o pedido conforme descrito em III.1.3.1.(1).

c) Sistema registra a data do pedido conforme descrito em III.1.3.1.(2).

d) Professor informa os dados necessários e pressiona o botão "Enviar".

e) Sistema exibe o pedido em tela, confirmando o seu envio ao administrador selecionado.

Alternativa: não há alternativa para este procedimento.

III.1.3.2 . Solicitar manutenção de equipamento

Para esta rotina de manutenção, teremos as seguintes operações: Abrir chamado, Visualizar chamado, Selecionar responsável pela manutenção e Selecionar chamado.

A figura III.21 apresenta os casos de uso para a rotina de Solicitação de manutenção de equipamento. Note que o processo Abrir chamado inclui outro processo, o Selecionar responsável pela manutenção, ou seja, antes de executar o processo principal, o professor deve primeiro selecionar quem será o responsável pela manutenção. Da mesma forma, o processo Visualizar chamado inclui o processo Selecionar chamado.

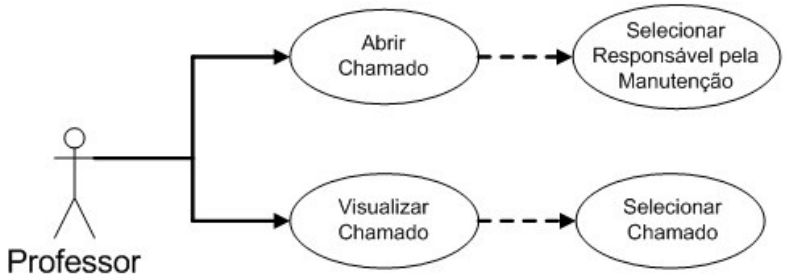


Figura III.21 - Casos de uso para solicitar manutenção de equipamento.

Dada a apresentação dos casos de uso para a rotina em questão, cabe transformar cada um dos processos em ações descritivas do sistema:

1. Selecionar chamado

Procedimento:

a) Sistema exibe lista com os equipamentos em manutenção.

b) Professor seleciona o equipamento desejado e escolhe a opção de manutenção "Visualizar chamado".

Alternativas:

a) Não existem manutenções cadastradas.

➔ Sistema apresenta como resultado da busca por equipamentos em manutenção uma lista vazia.

2. Selecionar responsável pela manutenção

Procedimento:

a) Sistema exibe lista com os responsáveis pela manutenção cadastrados.

b) Professor seleciona o responsável pela manutenção.

Alternativas:

a) Não existem responsáveis cadastrados.

➔ Sistema apresenta como resultado da busca por responsáveis uma lista vazia.

3. Abrir chamado

Procedimento:

a) Professor pressiona o botão “Abrir chamado”.

b) Professor seleciona o responsável pela manutenção conforme descrito em III.1.3.2.(2).

c) Professor preenche o restante do formulário, informando os seguintes dados: Número do inventário, Data de abertura do chamado, Descrição do defeito, Responsável pela manutenção, Previsão de término e Valor orçado para a manutenção.

d) Professor submete os dados pressionando o botão “Enviar”.

e) Sistema exibe mensagem confirmando o cadastro.

Alternativas:

a) Não existem equipamentos cadastrados

➔ Sistema informa que não existem equipamentos cadastrados com o número de inventário fornecido.

b) Número de inventário fornecido pelo professor não existe no cadastro do inventário.

➔ Sistema informa que não existem equipamentos cadastrados com o número de inventário fornecido.

c) Professor tenta abrir chamado de manutenção para equipamento que já esteja em manutenção.

➔ Sistema informa que o equipamento já existe chamado em aberto.

d) Valem as alternativas descritas em III.1.3.2.(2).

4. Visualizar chamado

Procedimento:

a) Professor seleciona chamado conforme descrito em III.1.3.2.(1).

b) Professor pressiona “Visualizar chamado”.

c) Sistema exibe formulário contendo as seguintes informações: Número do Inventário, Data de abertura do chamado, Descrição do defeito, Responsável pela manutenção, Previsão de término e Valor orçado para a manutenção.

Alternativas:

a) Valem as alternativas descritas em III.1.3.2.(1).

III.1.3.3 . Gerenciar notas das disciplinas de laboratório

Para esta rotina de manutenção, teremos as seguintes operações: Informar graus, Alterar graus, Visualizar graus e Selecionar grupos.

A figura III.22 apresenta os casos de uso para a rotina de gerenciamento de notas das disciplinas de laboratório. Note que os processos Informar graus, Alterar graus e Visualizar graus incluem outro processo, o Selecionar grupos, ou seja, antes de executar o processo principal (os três primeiros mencionados), o professor deve primeiro selecionar qual grupo sofrerá a manutenção.

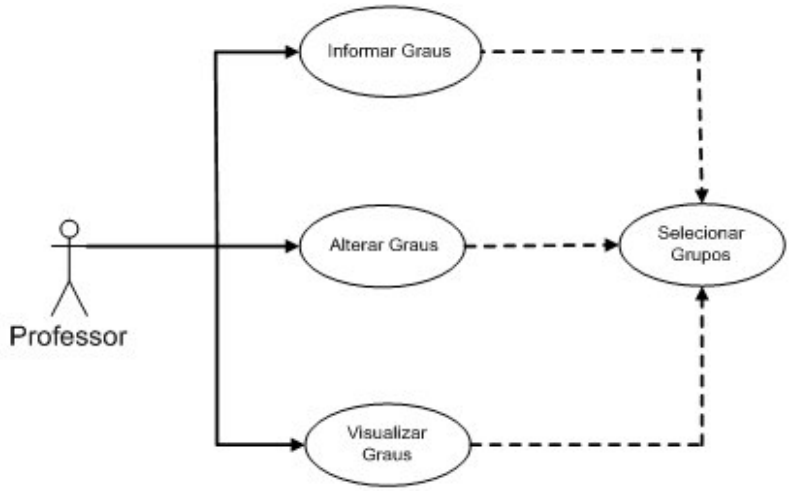


Figura III.22 - Casos de uso para gerenciar notas das disciplinas de laboratório.

Dada a apresentação dos casos de uso para a rotina em questão, cabe transformar cada um dos processos em ações descritivas do sistema:

1. **Selecionar grupos**

Procedimento:

a) Sistema exibe lista dos grupos existentes.

b) Professor pressiona a tecla de manutenção desejada ao lado do grupo escolhido ("Informar", "Alterar" ou "Visualizar" graus).

Alternativas:

a) Não existe grupo cadastrado.

➔ Sistema apresenta uma lista vazia como resultado pela busca de grupos cadastrados.

2. Informar graus

Procedimento:

a) Professor seleciona o grupo conforme descrito em III.1.3.3.(1).

b) Sistema exibe formulário para cadastro de notas. As notas podem ser atribuídas individualmente para cada membro do grupo, num total de quinze valores.

c) Professor pressiona "Informar graus", preenche formulário e envia para processamento, pressionando o botão "Confirmar".

d) Sistema armazena os dados e retorna à tela de "Gerência de graus do laboratório".

Alternativas:

a) Valem as alternativas descritas em III.1.3.3.(1).

3. Alterar graus

Procedimento:

a) Professor seleciona o grupo conforme descrito em III.1.3.3.(1).

b) Sistema exibe formulário para alteração de notas. As notas podem ser alteradas individualmente para cada membro do grupo, num total de quinze valores.

c) Professor pressiona "Informar graus", preenche formulário e envia para processamento, pressionando o botão "Confirmar".

d) Sistema armazena os dados e retorna à tela de "Gerência de graus".

Alternativas:

a) Valem as alternativas descritas em III.1.3.3.(1).

4. Visualizar Graus

Procedimento:

a) Professor seleciona o grupo conforme descrito em III.1.3.3.(1).

b) Sistema exibe formulário com a relação de notas do grupo. As notas foram atribuídas individualmente para cada membro do grupo, num total de quinze valores.

Alternativas:

a) Valem as alternativas descritas em III.1.3.3.(1).

III.1.3.4 . Gerenciar empréstimos

Para esta rotina de manutenção, teremos as seguintes operações: Visualizar empréstimo, Criar empréstimo, Selecionar empréstimo e Selecionar tipo de empréstimo.

A figura III.23 apresenta os casos de uso para a rotina de gerenciamento de empréstimos de insumos do laboratório. Note que os processos Criar empréstimo e Selecionar empréstimo incluem outro processo, o Selecionar tipo de empréstimo, ou seja, antes de executar o processo principal (os dois primeiros mencionados), o professor deve primeiro selecionar o tipo de empréstimo (equipamento ou componente). Da mesma forma, o processo Visualizar empréstimo inclui o processo Selecionar empréstimo.

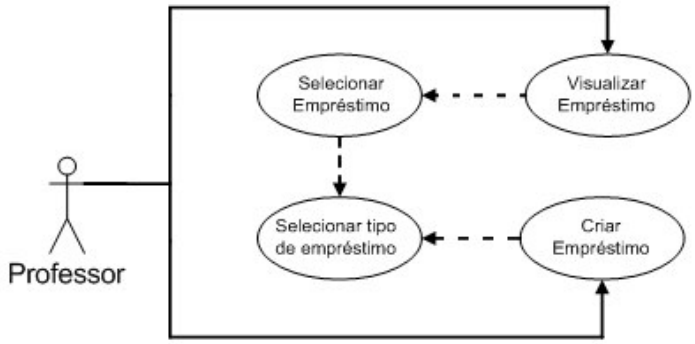


Figura III.23 - Casos de uso para gerenciar empréstimos de insumos do laboratório.

Dada a apresentação dos casos de uso para a rotina em questão, cabe transformar cada um dos processos em ações descritivas do sistema:

1. **Selecionar tipo de empréstimo**

Procedimento:

a) Sistema apresenta formulário com a opção de escolha entre "Componente" e "Equipamento".

b) Professor seleciona a opção desejada e continua com a utilização do sistema (liberando-o para outro procedimento).

Alternativas: não há alternativas para este processo.

2. **Selecionar empréstimo**

Procedimento:

a) Professor seleciona o tipo de empréstimo ("Componente" ou "Equipamento") conforme descrito em III.1.3.4.(1).

b) Sistema apresenta cadastro com os códigos de empréstimos associados ao *login* do usuário.

c) Professor seleciona o código desejado e pressiona a opção de manutenção.

Alternativas:

a) Não existem empréstimos associados ao *login* informado.

➔ Sistema exibe mensagem informando não haver empréstimos cadastrados para o *login*.

b) Valem as alternativas descritas em III.1.3.4.(1).

3. Visualizar empréstimo

Procedimento:

a) Professor seleciona o tipo de empréstimo ("Componente" ou "Equipamento") conforme descrito em III.1.3.4.(1).

b) Sistema apresenta cadastro com os códigos de empréstimos associados ao *login* do usuário.

c) Professor seleciona o código desejado e pressiona a opção "Visualizar empréstimo".

Alternativas:

a) Não existem empréstimos associados ao *login* informado.

➔ Sistema exibe mensagem informando não haver empréstimos cadastrados para o *login*.

b) Valem as alternativas descritas em III.1.3.4.(1).

4. Criar Empréstimo

Procedimento:

a) Professor seleciona o tipo de empréstimo ("Componente" ou "Equipamento") conforme descrito em III.1.3.4.(1).

b) Professor pressiona "Criar empréstimo".

c) Sistema exibe lista com os ativos disponíveis para empréstimo.

d) Professor seleciona o ativo e informa a quantidade a ser emprestada.

e) Professor pressiona “Criar empréstimo”.

f) Sistema exibe mensagem informando que o empréstimo foi incluído com sucesso.

Alternativas:

a) Não existem ativos cadastrados.

➔ Sistema exibe como resultado da busca por ativos uma lista vazia.

b) Valem as alternativas descritas em III.1.3.4.(1).

III.1.4 . Casos de uso para usuários alunos

Nesta seção detalharemos as tarefas assinaladas na figura III.5 (pacote da área de alunos) em atividades mais detalhadas. Desta forma, teremos condições de montar os diagramas de casos de usos do sistema. Ao final, teremos transformado as rotinas de administração em um modelo UML.

III.1.4.1 . Solicitar compra de componentes

Para esta rotina de manutenção, teremos as seguintes operações: Solicitar compra, Registrar data do pedido e Selecionar administrador.

A figura III.24 apresenta os casos de uso para a rotina de solicitação de compra de componentes. Note que o processo Solicitar compra inclui dois outros processos, o Registrar data do pedido e o Selecionar administrador, ou seja, antes de executar o processo principal, o aluno deve primeiro registrar a data do pedido e depois selecionar para qual administrador ele encaminhará o pedido.

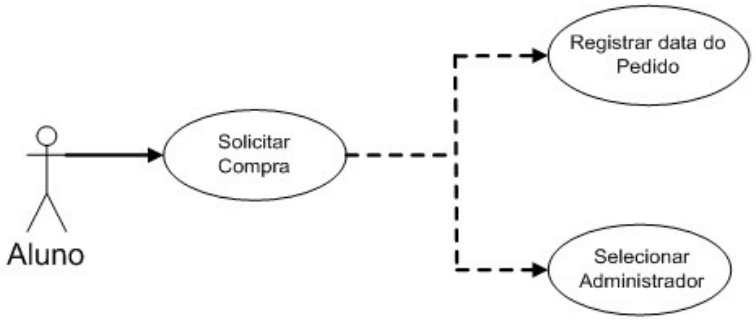


Figura III.24 - Casos de uso para solicitar compra de componentes.

Dada a apresentação dos casos de uso para a rotina em questão, cabe transformar cada um dos processos em ações descritivas do sistema:

1. Selecionar administrador

Procedimento:

a) Sistema exibe lista com administradores cadastrados.

b) Aluno seleciona o administrador que receberá o pedido.

Alternativa: não há alternativa para este procedimento.

2. Registrar data do pedido

Procedimento:

a) Sistema registra a data do cadastro do pedido de acordo com data e hora do servidor que hospeda o módulo e anexa ao pedido.

Alternativa: não há alternativa para este procedimento.

3. Solicitar compra

Procedimento:

a) Sistema exibe formulário para solicitação de compra de componentes com as seguintes informações a preencher: *E-mail*, Data, Descrição do material, Quantidade e Motivação da compra.

b) Aluno seleciona o administrador que receberá o pedido conforme descrito em III.1.4.1.(1).

c) Sistema registra a data do pedido conforme descrito em III.1.4.1.(2).

d) Aluno informa os dados faltantes e pressiona o botão "Enviar".

e) Sistema exibe o pedido em tela, confirmando o seu envio ao administrador selecionado.

Alternativa: não há alternativa para este procedimento.

III.1.4.2 . Solicitar manutenção de equipamento

Para esta rotina de manutenção, teremos as seguintes operações: Abrir chamado, Visualizar chamado, Selecionar responsável pela manutenção e Selecionar chamado.

A figura III.25 apresenta os casos de uso para a rotina de Solicitação de manutenção de equipamento. Note que o processo Abrir chamado inclui outro processo, o Selecionar responsável pela manutenção, ou seja, antes de executar o processo principal, o aluno deve primeiro selecionar quem será o responsável pela manutenção. Da mesma forma, o processo Visualizar chamado inclui o processo Selecionar chamado.

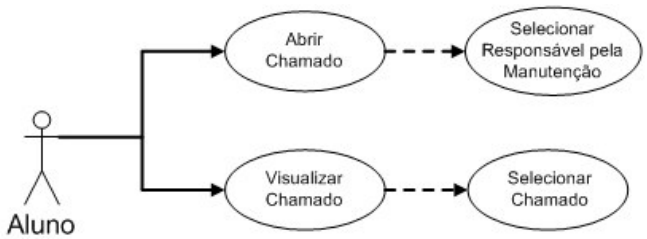


Figura III.25 - Casos de uso para solicitar manutenção de equipamento.

Dada a apresentação dos casos de uso para a rotina em questão, cabe transformar cada um dos processos em ações descritivas do sistema:

1. Selecionar chamado

Procedimento:

a) Sistema exibe lista com os equipamentos em manutenção.

b) Aluno seleciona o equipamento desejado e passa para o procedimento Visualizar chamado, descrito mais a frente no item III.1.4.2.(4).

Alternativas:

a) Não existem manutenções cadastradas.

➔ Sistema apresenta como resultado da busca por equipamentos em manutenção uma lista vazia.

2. Selecionar responsável pela manutenção

Procedimento:

a) Sistema exibe lista com os responsáveis pela manutenção.

b) Aluno seleciona o responsável pela manutenção.

Alternativas:

a) Não existem responsáveis cadastrados.

➔ Sistema apresenta como resultado da busca por responsáveis uma lista vazia.

3. Abrir chamado

Procedimento:

a) Aluno pressiona o botão "Abrir chamado".

b) Aluno seleciona o responsável pela manutenção conforme descrito em III.1.4.2.(2).

c) Aluno preenche o restante do formulário, informando os seguintes dados: Número do Inventário, Data de abertura do chamado, Descrição do defeito, Responsável pela manutenção, Previsão de término e Valor orçado para a manutenção.

d) Aluno submete os dados pressionando o botão "Enviar".

e) Sistema exibe mensagem conformando o cadastro.

Alternativas:

a) Não existem equipamentos cadastrados.

➔ Sistema informa que não existem equipamentos cadastrados com o número de inventário fornecido.

b) Número de inventário fornecido pelo aluno não existe no cadastro do inventário.

➔ Sistema informa que não existem equipamentos cadastrados com o número de inventário fornecido.

c) Aluno tenta abrir chamado de manutenção para equipamento que já esteja em manutenção.

➔ Sistema informa que o equipamento já tem chamado em aberto.

d) Valem as alternativas descritas em III.1.4.2.(2).

4. Visualizar chamado

Procedimento:

a) Aluno seleciona chamado conforme descrito em III.1.4.2.(1).

b) Aluno pressiona "Visualizar Chamado".

c) Sistema exibe formulário contendo as seguintes informações: Número do inventário, Data de abertura do chamado, Descrição do defeito, Responsável pela manutenção, Previsão de término e Valor orçado para a manutenção.

Alternativas:

a) Valem as alternativas descritas em III.1.4.2.(1).

III.1.4.3 . Gerenciar notas das disciplinas de laboratório

Para esta rotina de manutenção, teremos a operação Visualizar graus.

A figura III.26 apresenta o caso de uso para a rotina de gerenciamento de notas das disciplinas de laboratório.

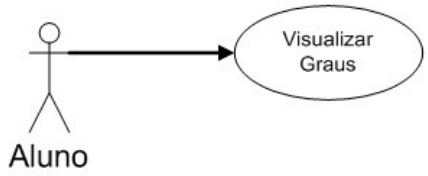


Figura III.26 - Casos de uso para visualizar graus do laboratório.

Dada a apresentação do caso de uso para a rotina em questão, cabe transformar seu processo em ações descritivas do sistema:

1. Visualizar graus

Procedimento:

a) Sistema exibe formulário com a relação de notas do grupo. As notas foram atribuídas individualmente para cada membro do grupo, num total de quinze valores.

Alternativa: não há alternativa para este procedimento.

III.1.4.4 . Gerenciar empréstimo

Para esta rotina de manutenção, teremos as seguintes operações: Visualizar empréstimo, Criar empréstimo, Selecionar empréstimo e Selecionar tipo de empréstimo.

A figura III.27 apresenta os casos de uso para a rotina de gerenciamento de empréstimos de insumos do laboratório. Note que os processos Criar empréstimo e Selecionar empréstimo incluem outro processo, o Selecionar tipo de empréstimo, ou seja, antes de executar o processo principal (os dois primeiros mencionados), o aluno deve primeiro selecionar o tipo de empréstimo (equipamento ou componente). Da mesma forma, o processo Visualizar empréstimo inclui o processo Selecionar empréstimo.

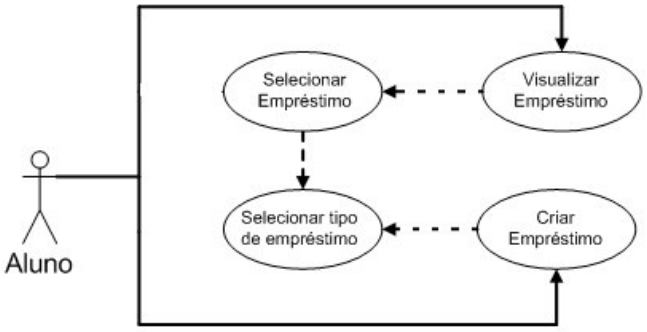


Figura III.27 - Casos de uso para gerenciar empréstimo.

Dada a apresentação do caso de uso para a rotina em questão, cabe transformar seu processo em ações descritivas do sistema:

1. Selecionar tipo de empréstimo

<u>Procedimento:</u>

a) Sistema apresenta formulário com a opção de escolha entre "Componente" e "Equipamento".

b) Aluno seleciona a opção desejada e procede a utilização do sistema.

2. Selecionar empréstimo

<u>Procedimento:</u>

a) Aluno seleciona o tipo de empréstimo ("Componente" ou "Equipamento") conforme descrito em III.1.4.4.(1).

b) Sistema apresenta cadastro com os códigos de empréstimos associados ao *login* do usuário.

c) Aluno seleciona o código desejado e pressiona a opção de manutenção desejada.

Alternativas:

a) Não existem empréstimos associados ao *login* informado.

➔ Sistema exibe mensagem informando não haver empréstimos cadastrados para o *login*.

b) Valem as alternativas descritas em III.1.4.4.(1).

3. Visualizar empréstimo

Procedimento:

a) Aluno seleciona o tipo de empréstimo ("Componente" ou "Equipamento") conforme descrito em III.1.4.4.(1).

b) Sistema apresenta cadastro com os códigos de empréstimos associados ao *login* do usuário.

c) Aluno seleciona o código desejado e pressiona a opção "Visualizar empréstimo".

Alternativas:

a) Não existem empréstimos associados ao *login* informado.

➔ Sistema exibe mensagem informando não haver empréstimos cadastrados para o *login*.

b) Valem as alternativas descritas em III.1.4.4.(1).

4. Criar Empréstimo

Procedimento:

a) Aluno seleciona o tipo de empréstimo ("Componente" ou "Equipamento") conforme descrito em III.1.4.4.(1).

b) Aluno pressiona "Criar empréstimo".

c) Sistema exibe lista com os ativos disponíveis para empréstimo.

d) Aluno seleciona o ativo e informa a quantidade a ser emprestada.

e) Aluno pressiona "Criar empréstimo".

f) Sistema exibe mensagem informando que o empréstimo foi incluído com sucesso.

Alternativas:

a) Não existem ativos cadastrados.

➔ Sistema exibe como resultado da busca por ativos uma lista vazia.

b) Valem as alternativas descritas em III.1.4.4.(1).

III.2 . Modelagem do banco de dados

Nesta etapa, onde já foi definida a parte conceitual do módulo (seu modelo lógico), definiremos quais dados devem ser armazenados e como eles devem se relacionar.

Esta fase é extremamente importante para o correto desenvolvimento do módulo, pois um banco de dados mal-estruturado prejudicará o correto funcionamento do sistema. Para tanto, o projeto se baseará fortemente no modelo conceitual já definido para identificar as entidades presentes no modelo e seus atributos e relacionamentos com outras entidades. Já foi definido que o módulo apresenta quatro áreas distintas: Área administrativa, Área pública, Área dos professores e Área dos alunos.

Cada uma dessas áreas terá suas próprias atribuições, mas com certeza elas interagirão umas com as outras. Tome como exemplo a entidade Componentes, que está presente na área administrativa, pois é ali que serão feitas as manutenções no seu cadastro,

mas também em outras duas áreas, na área de professores e na área de alunos, uma vez que os usuários dessas áreas poderão ver a relação de componentes, realizar empréstimos, etc. Da mesma forma, é possível identificar essa interação com a maioria das entidades presentes.

Por meio da análise do modelo conceitual do módulo, é possível identificar as seguintes entidades necessárias ao seu pleno funcionamento:

- Usuários: cadastro de todos os usuários do módulo e suas informações.
- Categorias: relação de categorias/subcategorias de insumos do laboratório.
- Componentes: tipo de insumo disponibilizado pelo laboratório.
- Equipamentos: tipo de insumo disponibilizado pelo laboratório.
- Fornecedores: cadastro dos fornecedores de insumos para o laboratório.
- Empréstimos: solicitações de empréstimos de insumos do laboratório feito pelos usuários do módulo.
- Mensagens: todas as mensagens trocadas entre usuários do site.

Estas são as entidades principais que o módulo deve conter. Faremos a seguir uma análise detalhada de cada uma, avaliando cada um dos seus atributos.

III.2.1 . Entidade usuários

Esta entidade é fundamental tanto ao módulo LegWeb como ao sistema que gerencia o módulo, o SID [1] (essa tabela já foi contemplada). Através dela é possível identificar os usuários, suas informações e o seu nível de atuação.

Os dados desta entidade no SID [1] são organizados em mais de um tipo de tabela, a saber:

- *user_base*: Tabela base de usuários, contém os dados comuns a todos os usuários SID [1] , independentemente de seu grupo. Todos os dados cruciais ao sistema (login, senha, etc) encontram-se nesta tabela.

- *user_detailed_professor*: Tabela de dados específicos de professores.

- *user_detailed_student*: Tabela de dados específicos de alunos.

- *user_detailed_employee*: Tabela de dados específicos de funcionários.

- *user_detailed_business_partner*: Tabela de dados específicos de contratados.

A primeira preocupação com relação a esta entidade é como relacioná-la ao controle de acesso ao módulo. Como é premissa de projeto que todo o usuário do LegWeb seja obrigatoriamente cadastrado do SID [1], acrescentaremos alguns parâmetros que permitirão identificar o acesso ao LegWeb na tabela que contém dados comuns a todos os usuários do SID [1], ou seja, na *user_base*. Desta forma, os seguintes atributos serão acrescidos a esta tabela:

- Situação do usuário no módulo: define se o usuário está ou não autorizado a acessar o módulo. Este atributo recebe o nome de *lw_user_status*, sendo do tipo *integer* e recebendo como padrão o valor '0' (não tem acesso ao módulo). Desta forma, todo o usuário do SID [1], por padrão, não é habilitado a utilizar o módulo.

- Nível de acesso ao módulo: define o nível de acesso do usuário no módulo. Este atributo recebe o nome *lw_access_level*, sendo do tipo *integer* e podendo atribuir quatro perfis diferentes: administrador (a variável recebe o valor "1"), professor (a variável recebe o valor "2"), funcionário (a variável recebe o valor "3") e aluno (a variável recebe o valor "4)".

- Registro do último acesso: registra data e hora do último acesso do usuário. Este atributo recebe o nome de *lw_last_access*, sendo do tipo *datetime*.

A figura III.28 contém a relação completa de atributos da tabela *user_base*, incluindo os novos atributos referentes ao controle de acesso ao módulo.

Tabela user_base *			
ATRIBUTO	**DESCRIÇÃO**	**TIPO**	**OBSERVAÇÃO**
user_id	Reaproveitado do SID	integer (10) unsigned	auto-increment
login_sid	Reaproveitado do SID	varchar (9)	
password	Reaproveitado do SID	varchar (13)	
status_id	Reaproveitado do SID	tinyint (3) unsigned	
inclusion_date	Reaproveitado do SID	date	
expiration_date	Reaproveitado do SID	date	
last_access	Reaproveitado do SID	datetime	
last_ip	Reaproveitado do SID	varchar (15)	
last_modification	Reaproveitado do SID	datetime	
login_net	Reaproveitado do SID	varchar (16)	
name	Reaproveitado do SID	varchar (50)	
gender	Reaproveitado do SID	enum ('M', 'F')	
birthdate	Reaproveitado do SID	date	
birthdate_f	Reaproveitado do SID	varchar (20)	
res_address_street	Reaproveitado do SID	varchar (60)	
res_address_number	Reaproveitado do SID	varchar (5)	
res_address_compl	Reaproveitado do SID	varchar (20)	
res_address_town	Reaproveitado do SID	varchar (20)	
res_address_number	Reaproveitado do SID	varchar (5)	
res_address_compl	Reaproveitado do SID	varchar (20)	
res_address_town	Reaproveitado do SID	varchar (20)	
res_address_city	Reaproveitado do SID	varchar (30)	
res_address_state	Reaproveitado do SID	char (2)	
res_address_zip	Reaproveitado do SID	varchar (8)	
res_address_f	Reaproveitado do SID	varchar (20)	
res_phone	Reaproveitado do SID	varchar (8)	
res_phone_f	Reaproveitado do SID	varchar (20)	
phone_cellular	Reaproveitado do SID	varchar (8)	
phone_cellular_f	Reaproveitado do SID	varchar (20)	
email	Reaproveitado do SID	varchar (50)	
email_f	Reaproveitado do SID	varchar (20)	
url	Reaproveitado do SID	varchar (100)	
url_f	Reaproveitado do SID	varchar (20)	
quota_default	Reaproveitado do SID	integer (10) unsigned	
quota_available	Reaproveitado do SID	integer (10) unsigned	
admin_comment	Reaproveitado do SID	varchar (100)	
res_phone_area_code	Reaproveitado do SID	char (2)	
res_phone_country_code	Reaproveitado do SID	char (2)	
phone_cellular_country_code	Reaproveitado do SID	char (2)	
phone_cellular_area_code	Reaproveitado do SID	char (2)	
lw_user_status	Define se usuário está ativo no módulo	integer (1)	default = 0
lw_access_level	Define o nível de acesso do usuário	integer (1)	1:administrador 2: professor 3: funcionário 4: aluno
lw_last_access	Registra data e hora do último acesso	datetime	

Figura III.28 – Detalhamento da tabela *user_base*

As figuras III.29 e III.30 apresentam, respectivamente, a relação completa de atributos das tabelas *user_detailed_professor* e *user_detailed_student*,. Nenhuma destas tabelas foi modificada para atender às necessidades do módulo, entretanto, por conta do seu relacionamento com a tabela *user_base*, elas são representadas neste documento.

Tabela user_detailed_professor			
ATRIBUTO	DESCRIÇÃO	TIPO	OBSERVAÇÃO
user_id	Reaproveitado do SID	Integer (10) unsigned	auto-increment
type_id	Reaproveitado do SID	Tinyint (4)	
titilation_id	Reaproveitado do SID	Tinyint (3) unsigned	
room	Reaproveitado do SID	Varchar (6)	
phone_univ	Reaproveitado do SID	Varchar (8)	
phone_univ_area_code	Reaproveitado do SID	Char (2)	
phone_univ_contry_code	Reaproveitado do SID	Char (2)	

Figura III.29 – Detalhamento da tabela *user_detailed_professor*

Tabela user_detailed_student			
ATRIBUTO	DESCRIÇÃO	TIPO	OBSERVAÇÃO
user_id	Reaproveitado do SID	Integer (10) unsigned	auto-increment
Dre_new	Reaproveitado do SID	Varchar (9)	
entrance_year	Reaproveitado do SID	Integer (10) unsigned	
entrance_period	Reaproveitado do SID	Tinyint (3) unsigned	
academic_situation_id	Reaproveitado do SID	Tinyint (3) unsigned	
academic_situation_year	Reaproveitado do SID	Integer (3) unsigned	
academic_situation_period	Reaproveitado do SID	Tinyint (3) unsigned	
company_name	Reaproveitado do SID	Varchar (40)	
company_name_f	Reaproveitado do SID	Tinyint (4)	
com_address_street	Reaproveitado do SID	Varchar (60)	
com_address_number	Reaproveitado do SID	Varchar (5)	
com_address_compl	Reaproveitado do SID	Varchar (20)	
com_address_town	Reaproveitado do SID	Varchar (20)	
com_address_city	Reaproveitado do SID	Varchar (30)	
com_address_state	Reaproveitado do SID	Char (2)	
com_address_zip	Reaproveitado do SID	Varchar (9)	
com_address_f	Reaproveitado do SID	Tinyint (4)	
com_phone	Reaproveitado do SID	Varchar (8)	
com_phone_f	Reaproveitado do SID	Tinyint (4)	
com_phone_area_code	Reaproveitado do SID	Char (2)	
com_phone_country_code	Reaproveitado do SID	Char (2)	

Figura III.30 – Detalhamento da tabela *user_detailed_student*

Como informado anteriormente, a entidade usuários não trata somente de informações relativas ao controle de acesso. Ela trata também da organização dos usuários do LegWeb em grupos com vistas a organizar as aulas de laboratório. Nessa linha, é apresentada uma tabela, denominada *lw_group*, com parâmetros que permitirão identificar o unicamente um grupo. São eles:

- Código de grupo: número que identificará unicamente o agrupamento de um ou mais usuários do módulo. Este atributo recebe o nome de *group_id*, sendo do tipo *integer*.

- Nome do grupo: o grupo poderá ser identificado por um nome mais familiar do que o código de grupo acima descrito (uma seqüência numérica). Este atributo recebe o nome de *group_name*, sendo do tipo *varchar* com até vinte caracteres.

- Turma do grupo: o grupo será alocado em alguma turma (EL1, EL2, por exemplo) e este parâmetro serve para registrar essa informação. Este atributo recebe o nome *group_class*, sendo do tipo *varchar* com até vinte caracteres.

Pelo exposto acima, podemos apresentar a figura III.31, que contém a relação completa da tabela *lw_group*, com as informações referentes à entidade usuários no que tange ao gerenciamento de grupos.

Tabela lw_group			
ATRIBUTO	DESCRIÇÃO	TIPO	OBSERVAÇÃO
group_id	Código do grupo	integer (10) unsigned	auto-increment
group_name	Nome do grupo	varchar (20)	
group_class	Turma do grupo	varchar (20)	

Figura III.31 – Detalhamento da tabela *lw_group*

Na seqüência, trataremos da atribuição de notas de laboratório para os grupos e, respectivamente, os usuários a eles atrelados. Para armazenar este tipo de informação, é apresentada uma tabela, denominada *lw_grades*, com parâmetros que permitirão identificar as notas nas disciplinas de laboratório dos usuários. São eles:

- Código de nota: número que identificará unicamente o um conjunto de notas de disciplinas de laboratório atribuídas a cada usuário, individualmente, cadastrado dentro de um grupo. Este atributo recebe o nome de *grade_id*, sendo do tipo *integer*.

- Graus ou notas: atributos que identificam as notas das disciplinas de laboratório atribuídas a cada usuário. Há um total de quinze atributos (desde o atributo *grade_1* até *grade_15*) disponíveis para atribuição de notas para diferentes avaliações feitas ao longo do curso. Desta forma, é possível a um professor ou administrador atribuir uma nota individualmente para cada aluno do grupo por aula ou tarefa de laboratório. Os atributos são do tipo *float*, recebendo como padrão o valor “0” (zero).

Pelo exposto acima, podemos apresentar a figura III.32, que contém a relação completa da tabela *lw_grades*, com as informações referentes à entidade usuários no que tange ao gerenciamento notas das disciplinas de laboratório.

Tabela lw_grades			
ATRIBUTO	DESCRIÇÃO	TIPO	OBSERVAÇÃO
grade_id	Código do grau	Integer (10) unsigned	auto-increment
grade_1	Grau 1	Float (2,3)	Default = 0
grade_2	Grau 2	Float (2,3)	Default = 0
grade_3	Grau 3	Float (2,3)	Default = 0
grade_4	Grau 4	Float (2,3)	Default = 0
grade_5	Grau 5	Float (2,3)	Default = 0
grade_6	Grau 6	Float (2,3)	Default = 0
grade_7	Grau 7	Float (2,3)	Default = 0
grade_8	Grau 8	Float (2,3)	Default = 0
grade_9	Grau 9	Float (2,3)	Default = 0
grade_10	Grau 10	Float (2,3)	Default = 0
grade_11	Grau 11	Float (2,3)	Default = 0
grade_12	Grau 12	Float (2,3)	Default = 0
grade_13	Grau 13	Float (2,3)	Default = 0
grade_14	Grau 14	Float (2,3)	Default = 0
grade_15	Grau 15	Float (2,3)	Default = 0

Figura III.32 – Detalhamento da tabela *lw_grades*

A seguir são apresentadas as tabelas que servirão como pontes de dados para relacionamentos *n*-para-*m*. Chamaremos estas tabelas de tabelas de referência cruzada. A figura III.33 apresenta a tabela *lw_xref_detailed_student_group*, que associa o detalhamento de um aluno cadastrado no módulo com o grupo ao qual foi associado. Qualquer aluno do módulo pode ser incluído em quantos grupos for preciso. É um caso típico que pede relacionamento *n*-para-*m*.

Tabela lw_xref_detailed_student_group			
ATRIBUTO	DESCRIÇÃO	TIPO	OBSERVAÇÃO
user_id	Código de aluno	Integer (10) unsigned	
group_id	Código de grupo	Integer (10) unsigned	

Figura III.33 – Detalhamento da tabela *lw_xref_detailed_student_group*

A figura III.34 apresenta a tabela *lw_xref_detailed_professor_group*, que associa o detalhamento de um professor cadastrado no módulo com o grupo ao qual foi associado para dar aulas. Qualquer professor do módulo pode ser incluído em quantos grupos for preciso. É outro caso típico que pede relacionamento *n*-para-*m*.

Tabela lw_xref_detailed_professor_group			
ATRIBUTO	DESCRIÇÃO	TIPO	OBSERVAÇÃO
user_id	Código de professor	Integer (10) unsigned	
group_id	Código de grupo	Integer (10) unsigned	

Figura III.34 – Detalhamento da tabela *lw_xref_detailed_professor_group*

A figura III.35 apresenta a tabela *lw_xref_detailed_student_grade*, que associa o detalhamento de um aluno cadastrado no módulo com as notas nas disciplinas de laboratório que lhe foram atribuídas.

Tabela lw_xref_detailed_student_grade			
ATRIBUTO	DESCRIÇÃO	TIPO	OBSERVAÇÃO
user_id	Código de aluno	Integer (10) unsigned	
grade_id	Código do grau	Integer (10) unsigned	

Figura III.35 – Detalhamento da tabela *lw_xref_detailed_student_grade*

A Figura III.36 mostra as tabelas da entidade usuários se relacionam entre si. Os relacionamentos são implementados fora do banco de dados, por *script* PHP [4], pelo fato de o banco *MySQL* [3], por si só, não suportar funções de relacionamento estático, como bancos *Microsoft Access*, por exemplo.

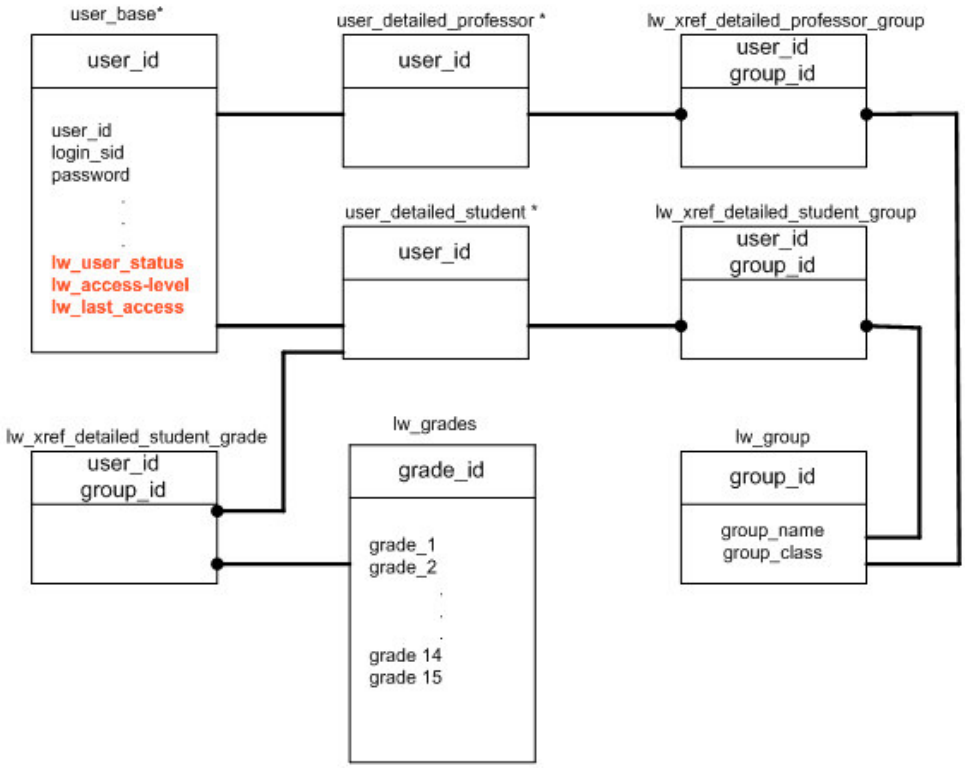


Figura III.36 - Entidade usuários e seus vínculos.

III.2.2 . Entidade categoria

A entidade categorias vai definir os grupos de insumos que estarão disponíveis no módulo. Nesta tabela precisaremos de atributos que identifiquem a categoria e seu nível.

Este é um caso em que uma linha da tabela pode referenciar outras linhas da mesma tabela (i.e., uma categoria é filha de outra categoria, o que a torna uma subcategoria da categoria principal). A representação gráfica desta entidade é apresentada na figura III.37. A chave de acesso para a categoria é o atributo *rank_id* e a ligação de uma categoria com seu pai está relacionada no atributo *father_code.*

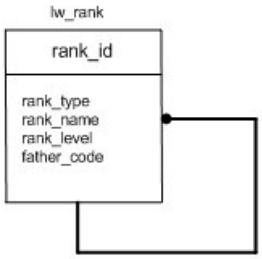


Figura III.37 - Modelagem da entidade categorias.

Pelo exposto acima, podemos apresentar a figura III.38, que contém a relação completa da tabela *lw_rank*.

Tabela lw_rank			
ATRIBUTO	**DESCRIÇÃO**	**TIPO**	**OBSERVAÇÃO**
rank_id	Código de categoria	Integer (10) unsigned	Auto-increment
rank_type	Identifica se é componente ou equipamento.	Integer (10) unsigned	0: componente 1: equipamento
rank_name	Descrição	Varchar (40)	
rank_level	Posição na hierarquia de árvore	Integer (10) unsigned	Padrão = 0
father_code	Código da categoria pai	Integer (10) unsigned	0 = não tem pai

Figura III.38 – Detalhamento da tabela de categoria.

III.2.3 . Entidade componentes

A entidade componentes vai definir os tipos de componentes que estarão disponíveis no laboratório e que serão gerenciados pelo módulo. Nessa linha, é apresentada uma tabela, denominada *lw_component*, com parâmetros que permitirão identificar o unicamente um componente. São eles:

- Código de componente: número que identificará unicamente um componente a ser disponibilizado pelo laboratório. Este atributo recebe o nome de *component_id*, sendo do tipo *integer*.

- Descrição geral: atributo que conterá uma breve descrição sobre o componente a ser disponibilizado (por exemplo, Capacitor de 10ηF, Resistor de 10KΩ, etc). Este atributo recebe o nome *component_description*, sendo do tipo *text.*

- Tolerância: atributo que contém a informação de tolerância do componente, uma especificação técnica informada pelo fabricante do mesmo. Este atributo recebe o nome *component_tolerance*, sendo do tipo *varchar.*

- Dimensões: atributo que contém as dimensões físicas do componente, uma especificação técnica informada pelo fabricante do mesmo. Este atributo recebe o nome *component_dimension*, sendo do tipo *varchar.*

- Quantidade da última compra: atributo que informa a quantidade de componentes adquiridos pelo laboratório com vistas a repor seu estoque de componentes. Armazena sempre o valor da última compra. Este atributo recebe o nome *component_acquired*, sendo do tipo *integer.*

- Quantidade queimada: atributo que armazena a quantidade de componentes indisponíveis para uso devido ao fato de estarem queimados. Este atributo recebe o nome *component_out*, sendo do tipo *integer.*

- Quantidade mal-conservada: atributo que informa a quantidade de componentes que não podem ser utilizados por mau-estado de conservação (terminais cortados ou curtos demais, etc). Este atributo recebe o nome *component_destroyed,* sendo do tipo *integer.*

- Quantidade extraviada: atributo que informa a quantidade de componentes que não podem ser utilizados por não terem sido devolvidos ou terem sido extraviados dentro do laboratório. Este atributo recebe o nome *component_missing,* sendo do tipo *integer.*

- Quantidade ativa após a última compra: atributo que informa a quantidade de componentes ativos após a reposição através de compra. Seu valor atualizado é obtido ao se subtrair os valores dos atributos "Quantidade mal-conservada", "Quantidade queimada" e "Quantidade extraviada" e somar os valores do atributo "Quantidade da última compra" ao último valor armazenado do atributo "Quantidade ativa", sendo a operação é implementada fora do banco de dados, por *script* PHP [4]. Este atributo recebe o nome *component_active*, sendo do tipo *integer*.

- Data da última compra: atributo que informa a data de quando foi realizada a última compra de um dado componente. ". Este atributo recebe o nome *component_date*, sendo do tipo *date.*

- Valor nominal do componente: atributo que informa o valor nominal de uso do componente, uma especificação técnica informada pelo fabricante do mesmo. Este atributo recebe o nome *component_nominal_value*, sendo do tipo *varchar.*

- Posição na hierarquia de árvore: atributo que informa a posição do componente dentro da estrutura de árvore organizada pela entidade Categoria, tratada no item III.2.2. Este atributo recebe o nome *component_rank_id*, sendo do tipo *integer*.

Pelo exposto acima, podemos apresentar a figura III.39, que contém a relação completa da tabela *lw_component*.

Tabela lw_component			
ATRIBUTO	**DESCRIÇÃO**	**TIPO**	**OBSERVAÇÃO**
component_id	Código de componente	Integer (10) unsigned	Auto-increment
component_description	Descrição geral	Text	
component_tolerance	Tolerância	Varchar (20)	
component_dimension	Dimensões	Varchar (20)	
component_acquired	Quantidade da última compra	Integer (10) unsigned	
component_active	Quantidade ativa após a última compra	Integer (10) unsigned	
component_out	Quantidade queimada	Integer (10) unsigned	
component_destroyed	Quantidade mal-conservada	Integer (10) unsigned	
component_missing	Quantidade extraviada	Integer (10) unsigned	
component_date	Data da última compra	Date	
component_nominal_value	Valor nominal do componente	Varchar (20)	
component_rank_id	Posição na hierarquia de árvore	Integer (10) unsigned	

Figura III.39 – Detalhamento da tabela de componentes.

III.2.4 . Entidade fornecedores

Os atributos necessários à entidade fornecedores são:

- Código de fornecedor: número que identificará unicamente um fornecedor de ativos para o laboratório. Este atributo recebe o nome de *vendor_id*, sendo do tipo *integer*.

- Razão social: atributo que conterá a razão social do fornecedor de ativos para o laboratório. Este atributo recebe o nome *company_name*, sendo do tipo *text.*

- Endereço: o endereço do fornecedor é armazenado em sete atributos. São eles:

  - *company_address_street*: armazena o nome da rua onde fica o escritório do fornecedor. É um atributo do tipo *varchar.*

  - *company_address_number*: armazena o número do escritório do fornecedor. É um atributo do tipo *varchar.*

  - *company_address_compl*: armazena complementos (apartamento, bloco e quadra, por exemplo) onde fica o escritório do fornecedor. É um atributo do tipo *varchar.*

  - *company_address_town*: armazena o nome do bairro onde fica o escritório do fornecedor. É um atributo do tipo *varchar.*

  - *company_address_city*: armazena o nome da cidade onde fica o escritório do fornecedor. É um atributo do tipo *varchar.*

  - *company_address_state*: armazena o nome do estado onde fica o escritório do fornecedor. É um atributo do tipo *varchar.*

  - *company_address_zip*: armazena o CEP do local onde fica o escritório do fornecedor. É um atributo do tipo *varchar.*

- Telefone: os telefones do fornecedor são armazenados em nove atributos. São eles:

  - *company_phone*: armazena o número de telefone do escritório do fornecedor. É um atributo do tipo *varchar*.

  - *company_phone_area_code*: armazena código de área do número de telefone do escritório do fornecedor. É um atributo do tipo *char*.

  - *company_phone_country_code*: armazena código de país do número de telefone do escritório do fornecedor. É um atributo do tipo *char*.

  - *resseling_phone*: armazena o número de telefone do representante comercial. É um atributo do tipo *varchar*.

  - *resseling_phone_area_code*: armazena código de área do número de telefone do representante comercial. É um atributo do tipo *char*.

  - *resseling_phone_country_code*: armazena código de país do número de telefone do representante comercial. É um atributo do tipo *char*.

  - *resseling_phone_cellular*: armazena o número de telefone celular do representante comercial. É um atributo do tipo *varchar*.

  - *resseling_phone_cellular_area_code*: armazena código de área do número de telefone celular do representante comercial. É um atributo do tipo *char*.

  - *resseling_phone_cellular_country_code*: armazena código de país do número de telefone celular do representante comercial. É um atributo do tipo *char*.

- CNPJ: atributo que conterá o CNPJ do fornecedor de ativos para o laboratório. Este atributo recebe o nome *company_cnpj*, sendo do tipo *varchar*.

- Nome do representante: atributo que conterá o nome do representante comercial do fornecedor de insumos para o laboratório. Este atributo recebe o nome *resseling_name* e é do tipo *varchar*.

- Observações: atributo que conterá informações sobre o fornecedor no que tange à visão do administrador do módulo. É um campo livre, onde qualquer informação complementar por ser gravada. Este atributo recebe o nome de *comments* e é do tipo *text*.

Pelo exposto acima, podemos apresentar a figura III.40, que contém a relação completa da tabela *lw_vendor*.

Tabela lw_vendor			
ATRIBUTO	**DESCRIÇÃO**	**TIPO**	**OBSERVAÇÃO**
vendor_id	Código de fornecedor	Integer (10) unsigned	Auto-increment
company_name	Razão social	Varchar (100)	
company_address_street	Endereço (rua)	Varchar (60)	
company_address_number	Endereço (número)	Varchar (5)	
company_address_compl	Endereço (complemento)	Varchar (20)	
company_address_town	Endereço (bairro)	Varchar (20)	
company_address_city	Endereço (cidade)	Varchar (30)	
company _address_state	Endereço (estado)	Varchar (2)	
company _address_zip	Endereço (CEP)	Varchar (8)	
company _phone	Telefone da empresa	Varchar (8)	
company _phone_area_code	Código de área	Char (2)	
company_phone_country_code	Código do país	Char (2)	
resseling_phone	Telefone do representante	Varchar (8)	
resseling_phone_area_code	Código de área	Char (2)	
resseling_phone_country_code	Código do país	Char (2)	
resseling_phone_cellular	Celular do representante	Varchar (8)	
resseling_phone_cellular _area_code	Código de área	Char (2)	
resseling_phone_cellular _country_code	Código do país	Char (2)	
company_CNPJ	CNPJ	Varchar (30)	
resseling_name	Nome do representante	Varchar (50)	
comments	Observações	Text	

Figura III.40 – Detalhamento da tabela de fornecedores.

Por uma questão de organização, serão criadas tabelas separadas para fornecedores (*lw_vendor*) e equipes de reparo(*lw_repair_team*). Entretanto, para a lógica de programação do módulo, eles tem a mesma função e não existe tal distinção.

A separação tem o objetivo de preparar o módulo para futuras funcionalidades que requeiram a separação entre esses dois atores. Ela faz mais sentido quando tratamos de equipamentos (item III.2.5), onde um fornecedor que atue na venda de equipamentos não necessariamente será um fornecedor de mão-de-obra para reparos e vice-versa.

O *script* PHP [4], ao cadastrar um fornecedor, o faz nas duas tabelas, ou seja, ao cadastrar um fornecedor, automaticamente será cadastrada uma equipe de reparo. Entretanto, para os casos de uso, aqueles que utilizam dados de fornecedores fazem as consultas/inserções na tabela *lw_vendor* e aqueles referentes à manutenções utilizam a tabela *lw_repair_team*.

A figura III.41 apresenta a relação de atributos da tabela *lw_repair_team*. Note que a tabela apresenta os mesmos atributos da tabela *lw*_vendor, com o acréscimo do atibuto *vendor_id* como uma referência à tabela *lw_vendor,* garantindo a associação.

Tabela lw_repair_team			
ATRIBUTO	**DESCRIÇÃO**	**TIPO**	**OBSERVAÇÃO**
repair_team_id	Código de fornecedor	Integer (10) unsigned	Auto-increment
company_name	Razão social	Varchar (100)	
company_address_street	Endereço (rua)	Varchar (60)	
company_address_number	Endereço (número)	Varchar (5)	
company_address_compl	Endereço (complemento)	Varchar (20)	
company_address_town	Endereço (bairro)	Varchar (20)	
company_address_city	Endereço (cidade)	Varchar (30)	
company _address_state	Endereço (estado)	Varchar (2)	
company _address_zip	Endereço (CEP)	Varchar (8)	
company _phone	Telefone da empresa	Varchar (8)	
company _phone_area_code	Código de área	Char (2)	
company_phone_country_code	Código do país	Char (2)	
responsible_phone	Telefone do representante	Varchar (8)	
responsible _phone_area_code	Código de área	Char (2)	
responsible_phone_country_code	Código do país	Char (2)	
responsible_phone_cellular	Celular do representante	Varchar (8)	
responsible_phone_cellular _area_code	Código de área	Char (2)	
responsible_phone_cellular _country_code	Código do país	Char (2)	
company_CNPJ	CNPJ	Varchar (30)	
responsible_name	Nome do representante	Varchar (50)	
comments	Observações	Text	
vendor_id	Código de fornecedor	Integer (10) unsigned	Referencia tabela vendor, pois o cadastro é feito na mesma tela.

Figura III.41 – Detalhamento da tabela de equipe de reparo.

A seguir são apresentadas as tabelas de referência cruzada utilizadas para esta entidade. A figura III.42 apresenta a tabela *lw_xref_vendor_component*, que associa o detalhamento de um fornecedor cadastrado no módulo com o componente que forneceu ao laboratório. Qualquer fornecedor pode ser atrelado a quantos componentes for preciso. É um caso típico que pede relacionamento *n*-para-*m*.

Tabela lw_xref_vendor_component			
ATRIBUTO	**DESCRIÇÃO**	**TIPO**	**OBSERVAÇÃO**
vendor_id	Código do fornecedor	Integer (10) unsigned	
component_id	Código do componente	Integer (10) unsigned	

Figura III.42 – Detalhamento da tabela de referência cruzada entre fornecedor e componente

A Figura III.43 mostra como as tabelas da entidade fornecedores se relacionam entre si. Os relacionamentos são implementados fora do banco de dados, por *script* PHP [4], pelo fato de o banco *MySQL* [3], por si só, não suportar funções de relacionamento estático, como bancos *Microsoft Access*, por exemplo.

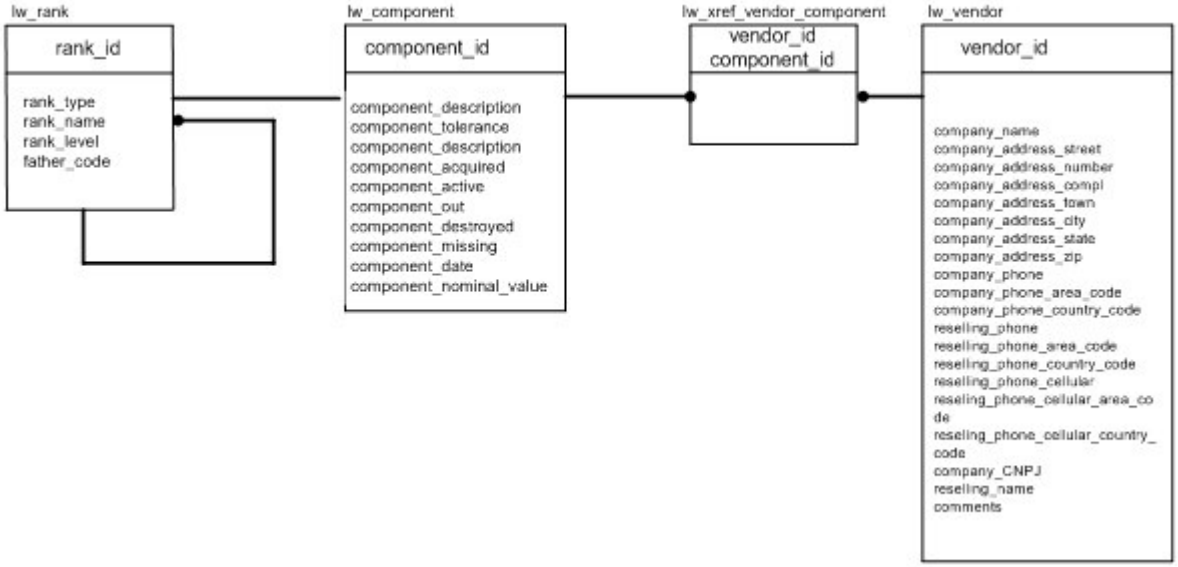


Figura III.43 - Modelagem do relacionamento das entidades componentes e fornecedores.

III.2.5 . Entidade Equipamentos

A entidade equipamentos vai definir os tipos de equipamentos que estarão disponíveis no laboratório e que serão gerenciados pelo módulo. Nessa linha, é apresentada uma tabela, denominada *lw_equipment*, com parâmetros que permitirão identificar o unicamente um equipamento. São eles:

- Código de equipamento: número que identificará unicamente um equipamento a ser disponibilizado pelo laboratório. Este atributo recebe o nome de *equip_id*, sendo do tipo *integer*.

- Nome do equipamento: atributo que conterá uma breve descrição sobre o equipamento a ser disponibilizado (por exemplo, Fonte HP, Osciloscópio Panasonic, etc). Este atributo recebe o nome *equip_name*, sendo do tipo *varchar*.

- Nome do Fabricante: atributo que conterá o nome do fornecedor do equipamento (por exemplo, Panasonic, NEC) a ser disponibilizado. Este atributo recebe o nome *equip_manufacturer*, sendo do tipo *varchar.*

- Data de aquisição: atributo que conterá a data em que o equipamento foi adquirido junto a um fornecedor pelo laboratório. Este atributo recebe o nome *equip_acquire*, sendo do tipo *date*.

- Custo do equipamento: atributo que conterá o registro do valor pago pela UFRJ para adquirir o equipamento junto ao fornecedor. Este atributo recebe o nome *equip_value*, sendo do tipo *varchar.*

- Término de garantia: atributo que conterá a data limite de cobertura de garantia para os equipamentos do laboratório. Este atributo recebe o nome *equip_guarantee*, sendo do tipo *date*.

- Dimensões do equipamento: atributo que conterá as dimensões físicas do equipamento (comprimento, largura e altura, por exemplo). Este atributo recebe o nome *equip_dimension*, sendo do tipo *varchar.*

- Data da última manutenção: atributo que conterá o registro da data da última manutenção sofrida por um equipamento. Este atributo recebe o nome *equip_last_repair*, sendo do tipo *date.*

- Manutenções sofridas: atributo que conterá o registro da quantidade de manutenções sofridas por um equipamento. Este atributo recebe o nome *equip_repair_time*, sendo do tipo *integer.*

- Custo de manutenção: atributo que conterá o registro do último valor pago pela UFRJ para realizar manutenção de um equipamento. Este atributo recebe o nome *equip_repair_cost*, sendo do tipo *varchar.*

- Estado da manutenção: atributo que informa se um equipamento registrado no módulo se encontra ou não em manutenção. Este atributo recebe o nome *equip_repair_status*, sendo do tipo *bool.* Ele recebe como padrão o valor *false*.

- Descrição do defeito: atributo que permite a inclusão de uma descrição detalhada do defeito do equipamento. É um campo de escrita livre do tipo *text*, que recebe o nome *equip_repair_description*.

- Número de inventário: atributo que conterá o registro do número de inventário do equipamento. Este atributo recebe o nome *equip_control_number*, sendo do tipo *varchar.*

- Contrato de manutenção: atributo que conterá o número do contrato de manutenção de um equipamento. Este atributo recebe o nome *equip_repair_contract*, sendo do tipo *varchar.*

- Término de contrato de manutenção: atributo que conterá a data limite de cobertura de um contrato de manutenção para os equipamentos do laboratório. Este atributo recebe o nome *equip_repair_end*, sendo do tipo *date*.

- Posição na hierarquia de árvore: atributo que informa a posição do equipamentodentro da estrutura de árvore organizada pela entidade Categoria, tratada no item III.2.2. Este atributo recebe o nome *equip_rank_id*, sendo do tipo *integer*.

A figura III.44 apresenta a relação de atributos da tabela *lw_repair_team*.

Tabela lw_equipment			
ATRIBUTO	DESCRIÇÃO	TIPO	OBSERVAÇÃO
equip_id	Código do equipamento	Integer (10) unsigned	Auto-increment
equip_name	Nome do equipamento	Varchar (40)	
equip_manufacturer	Nome do fabricante	Varchar (60)	
equip_acquire	Data de aquisição	date	
equip_value	Valor do equipamento	Varchar (30)	
equip_guarantee	Término da garantia	date	
equip_dimension	Dimensões do equipamento	Varchar (30)	
equip_last_repair	Ultima manutenção	date	
equip_repair_time	Manutenções sofridas	Integer (10) unsigned	
equip_repair_cost	Custo da última manutenção	Varchar (30)	
equip_repair_status	Status da manutenção	bool	Default = false
equip_repair_description	Descrição do defeito	text	
equip_control_number	Número de inventário	Varchar (20)	
equip_repair_contract	Contrato de manutenção	Varchar (30)	
equip_repair_end	Fim do contrato de manutenção	date	
equip_rank_id	Posição na hierarquia de árvore	Integer (10) unsigned	

Figura III.44 - Detalhamento da tabela de equipamentos.

A seguir é apresentada a tabela de referência cruzada utilizada para esta entidade. A figura III.45 apresenta a tabela *lw_xref_repair_team_equipment*, que associa o detalhamento de uma equipe de reparos cadastrada no módulo com o equipamento ao qual prestou suporte. Qualquer equipe de reparos pode ser atrelada a quantos equipamentos for preciso. É um caso típico que pede relacionamento *n*-para-*m*.

Tabela lw_xref_repair_team_equipment			
ATRIBUTO	DESCRIÇÃO	TIPO	OBSERVAÇÃO
repair_team_id	Código da equipe de manutençã	Integer (10) unsigned	
equip_id	Código do equipamento	Integer (10) unsigned	

Figura III.45 - Detalhamento da tabela deeferência cruzada entre equipe de manutenção e equipamento.

A Figura III.46 mostra como as tabelas da entidade Equipamentos se relacionam entre si. Os relacionamentos são implementados fora do banco de dados, por *script* PHP [4], pelo fato de o banco *MySQL* [3], por si só, não suportar funções de relacionamento estático, como bancos *Microsoft Access*, por exemplo.

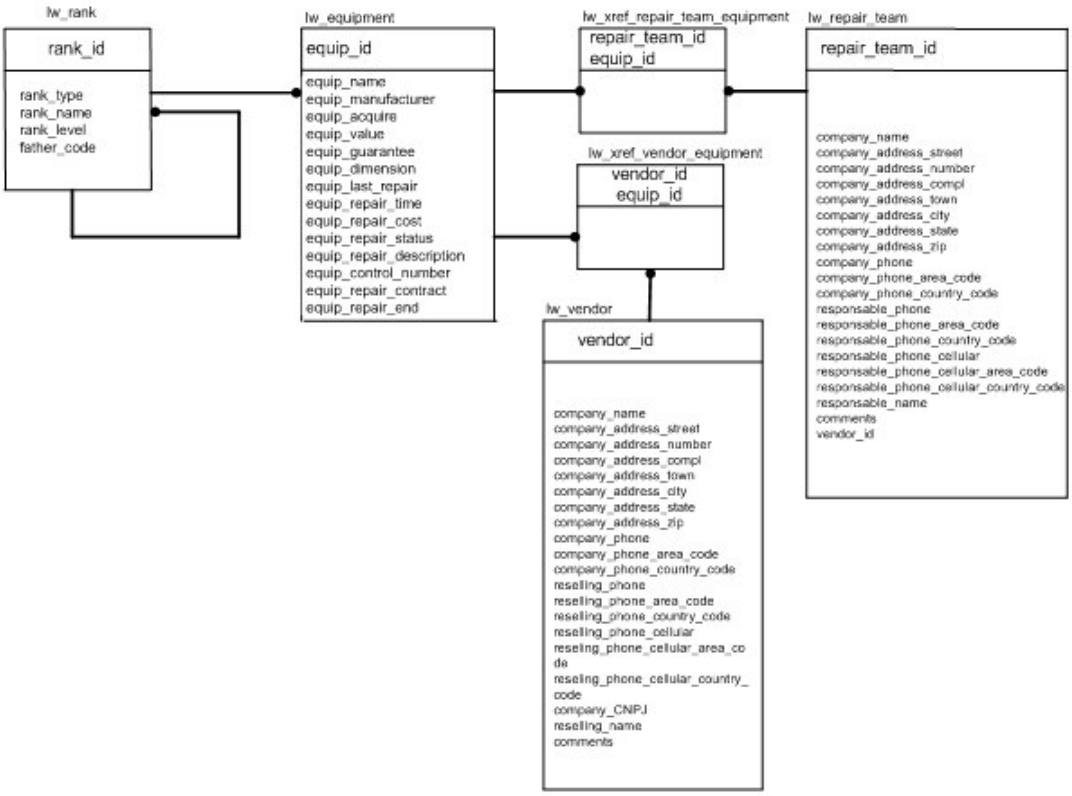


Figura III.46 - Modelagem do relacionamento das entidades equipamentos e fornecedores.

III.2.6 . Entidade Empréstimos

A entidade empréstimos vai definir o modelo de armazenamento dos registros de eventos de empréstimos feitos pelo módulo. Nessa linha, são apresentadas as tabelas *lw_borrow_comp*, com parâmetros que permitirão identificar unicamente um empréstimo de componente, e a *lw_borrow_equip*, com parâmetros que permitirão identificar um empréstimo de equipamento. A figura III.47 trata da tabela que gerencia o empréstimo de componentes, que conta com os seguintes atributos:

- Código de empréstimo: número que identificará unicamente um empréstimo de componentes do laboratório. Este atributo recebe o nome de *borrow_comp_id*, sendo do tipo *integer*.

- Data do empréstimo: atributo que armazenará a data em que o empréstimo foi solicitado. Este atributo recebe o nome de *borrow_comp_date*, sendo do tipo *date*.

- Quantidade do empréstimo: atributo que armazenará a quantidade de um dado componente que foi disponibilizado pelo laboratório. Este atributo recebe o nome de *borrow_comp_quant* e é do tipo *integer*.

- Estado do pedido: atributo que informa o estado de um empréstimo, ou seja, se ele foi agendado (os componentes ainda serão entregues), se está em curso (o usuário já está de posse dos componentes) ou se já foi encerrado (o usuário devolveu os componentes). Esse atributo recebe o nome de *borrow_status* e é do tipo *integer*.

Tabela lw_borrow_comp			
ATRIBUTO	**DESCRIÇÃO**	**TIPO**	**OBSERVAÇÃO**
borrow_comp_id	Código do empréstimo	Integer (10) unsigned	Auto-increment
borrow_comp_date	Data do empréstimo	date	
borrow_comp_quant	Quantidade do empréstimo	Integer (10) unsigned	
borrow_status	Status do pedido	integer	1: a entregar 2: em uso 3: devolvido

Figura III.47 - Detalhamento da tabela de empréstimo de componentes.

Na seqüência, é apresentada a figura III.48, que trata da tabela que gerencia o empréstimo de equipamentos, que conta com os seguintes atributos:

- Código de empréstimo: número que identificará unicamente um empréstimo de equipamentos do laboratório. Este atributo recebe o nome de *borrow_equip_id*, sendo do tipo *integer*.

- Data do empréstimo: atributo que armazenará a data em que o empréstimo foi solicitado. Este atributo recebe o nome de *borrow_equip_date*, sendo do tipo *date*.

- Quantidade do empréstimo: atributo que armazenará a quantidade de um dado equipamento que foi disponibilizado pelo laboratório. Este atributo recebe o nome de *borrow_equip_quant* e é do tipo *integer*.

- Estado do pedido: atributo que informa o estado de um empréstimo, ou seja, se ele foi agendado (os equipamentos ainda serão entregues), se está em curso (o usuário já está de posse dos equipamentos) ou se já foi encerrado (o usuário

devolveu os equipamentos). Esse atributo recebe o nome de *borrow_status* e é do tipo *integer*.

Tabela lw_borrow_equip			
ATRIBUTO	DESCRIÇÃO	TIPO	OBSERVAÇÃO
borrow_equip_id	Código do empréstimo	Integer (10) unsigned	Auto-increment
borrow_ equip_date	Data do empréstimo	date	
borrow_ equip_quant	Quantidade do empréstimo	Integer (10) unsigned	
borrow_status	Status do pedido	integer	1: a entregar 2: em uso 3: devolvido

Figura III.48 – Detalhamento da tabela de empréstimo de equipamentos.

A seguir são apresentadas as tabelas de referência cruzada utilizadas para esta entidade. A figura III.49 apresenta a tabela *lw_xref_borrow_comp*, que associa o a informação de empréstimo armazenada com o componente emprestado. A seguir, a figura III.50 apresenta a tabela *lw_xref_borrow_comp_user*, que associa a informação de empréstimo de componente e usuário. A figura III.51 apresenta a tabela *lw_xref_borrow_equip*, que associa a informação de empréstimo e equipamento. Por último, a figura III.52 apresenta a tabela *lw_xref_borrow_equip_user*, que associa a informação de empréstimo de equipamento com o usuário.

Tabela lw_xref_borrow_comp			
ATRIBUTO	DESCRIÇÃO	TIPO	OBSERVAÇÃO
component_id	Código do componente	Integer (10) unsigned	
borrow_comp_id	Código do empréstimo	Integer (10) unsigned	

Figura III.49 - Referência cruzada entre empréstimo e componente.

Tabela lw_xref_borrow_comp_user			
ATRIBUTO	DESCRIÇÃO	TIPO	OBSERVAÇÃO
user_id	Código do usuário	Integer (10) unsigned	
borrow_comp_id	Código do empréstimo	Integer (10) unsigned	

Figura III.50 - Referência cruzada entre empréstimo de componente e usuário

Tabela lw_xref_borrow_equip			
ATRIBUTO	DESCRIÇÃO	TIPO	OBSERVAÇÃO
equip_id	Código do componente	Integer (10) unsigned	
borrow_equip_id	Código do empréstimo	Integer (10) unsigned	

Figura III.51 - Referência cruzada entre empréstimo e equipamento.

Tabela lw_xref_borrow_equip_user			
ATRIBUTO	DESCRIÇÃO	TIPO	OBSERVAÇÃO
user_id	Código do usuário	Integer (10) unsigned	
borrow_equip_id	Código do empréstimo	Integer (10) unsigned	

Figura III.52 - Referência cruzada entre empréstimo de equipamento e usuário.

A Figura III.53 mostra como as tabelas das entidades Equipamentos, Componentes e Empréstimos se relacionam entre si. Os relacionamentos são implementados fora do banco de dados, por *script* PHP [4], pelo fato de o banco *MySQL* [3], por si só, não suportar funções de relacionamento estático, como bancos *Microsoft Access*, por exemplo.

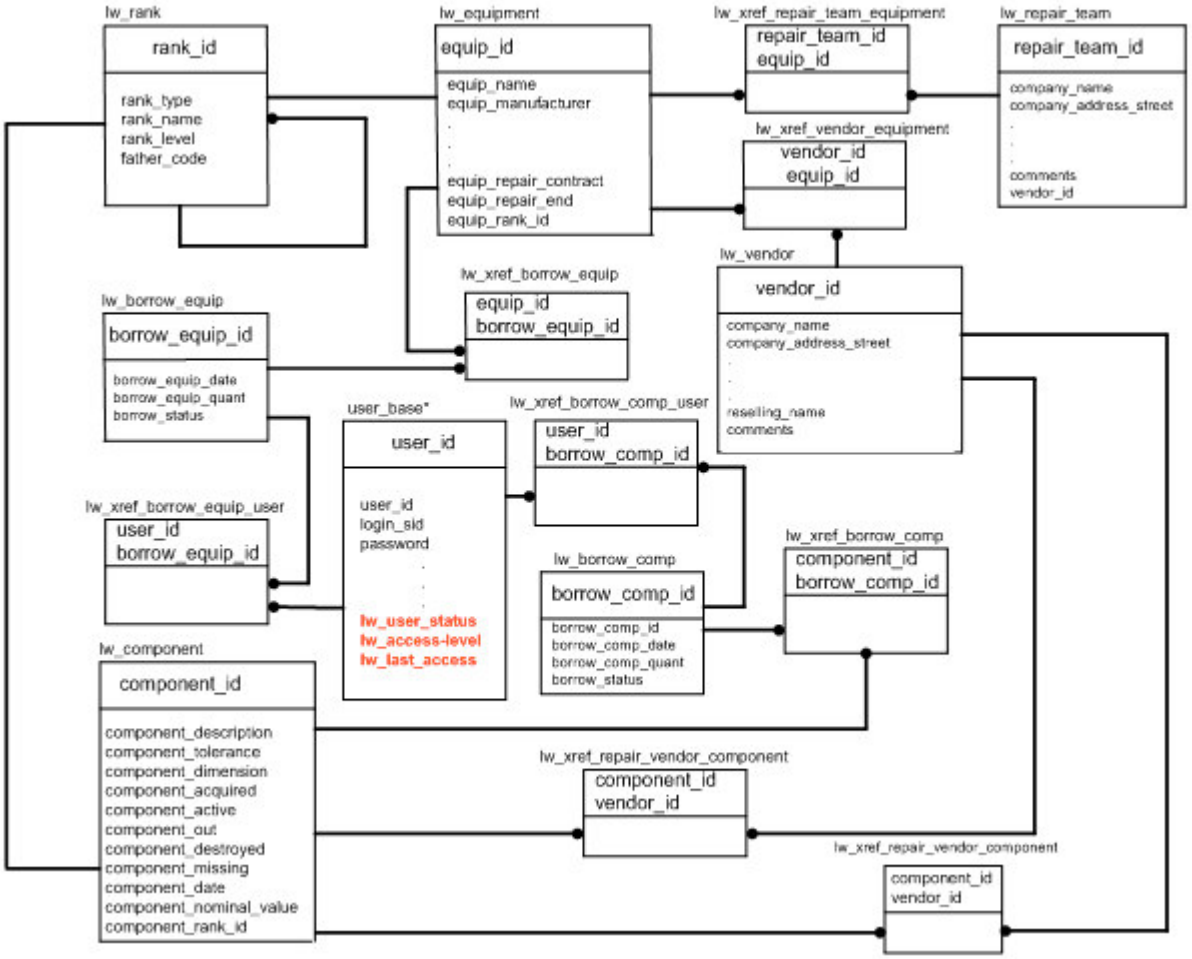


Figura III.53 - Modelagem das entidades equipamentos, componentes e empréstimos.

III.2.7 . Entidade Mensagens

A entidade mensagens tem como objetivo armazenar todas as mensagens enviadas ou recebidas pelo sistema. Devemos ter uma única tabela com os campos:

- Código da mensagem: número que identificará unicamente uma mensagem do módulo. Este atributo recebe o nome de *message_id*, sendo do tipo *integer*.

- Data da mensagem: atributo que armazenará a data de envio da mensagem. Este atributo recebe o nome de *date,* sendo do tipo *date*.

- Assunto da mensagem: atributo que armazena o título da mensagem, recebendo o nome *subject*, sendo do tipo *varchar*.

- Mensagem: atributo que armazena o corpo da mensagem, recebendo o nome *text*, sendo do tipo *text.*

- Estado da mensagem: atributo que informa o estado de uma mensagem, ou seja, se ela foi recebida, lida, enviada ou respondida. Recebe o nome *status,* sendo do tipo *integer.*

- Código do destinatário da mensagem: atributo que informa a identidade (*user_id)* do remetente da mensagem no SID [1]. Recebe o nome *sender_id,* sendo do tipo *integer.*

Tabela lw_messages			
ATRIBUTO	**DESCRIÇÃO**	**TIPO**	**OBSERVAÇÃO**
message_id	Código da mensagem	Integer (10) unsigned	Auto-increment
date	Data da mensagem	date	
subject	Assunto da mensagem	Varchar (40)	
text	Mensagem	text	0: recebida 1: lida 2: enviada 3: respondida
status	Status da mensagem	integer	
sender_id	Código do remetente da mensagem	Integer (10) unsigned	Default = 0

Figura III.54 - Detalhamento da tabela de mensagens.

A Figura III.53 mostra como as tabelas das entidades Equipamentos, Componentes e Empréstimos se relacionam entre si. Os relacionamentos são implementados fora do banco de dados, por *script* PHP [4], pelo fato de o banco *MySQL* [3], por si só, não suportar funções de relacionamento estático, como bancos *Microsoft Access*, por exemplo.

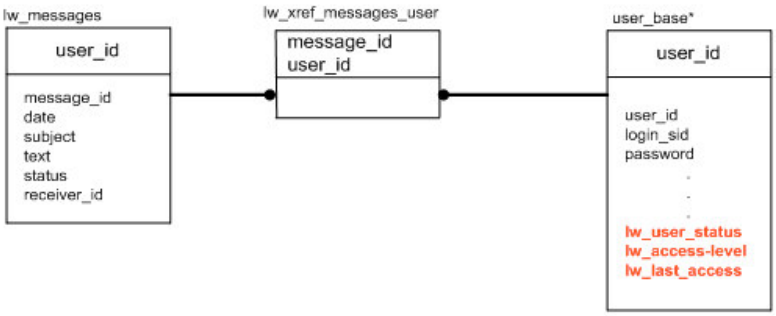


Figura III.55 - Modelagem da entidade mensagens.

III.2.8 . Visão geral do Modelo de Dados

A figura III.56 apresenta uma visão completa do modelo entidade de relacionamento gerado. Nessa visão são apresentadas todas as relações existentes no modelo lógico dos dados, dado que até o momento foram apresentadas as relaçõpes pontualmente (tabela por tabela). É muito importante a apresentação desta visão geral para analisar o que foi montado e diagnosticar possíveis problemas de estrutura.

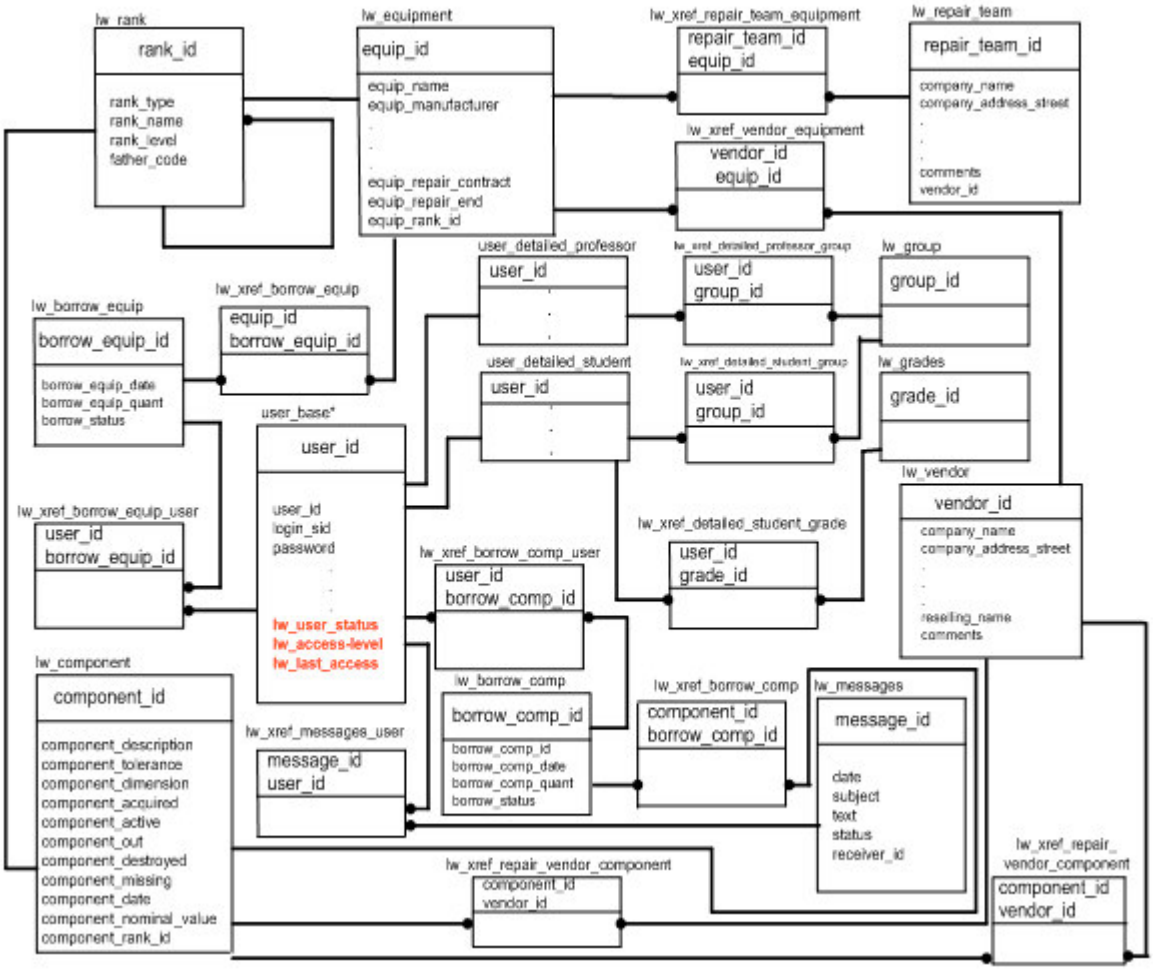


Figura III.56 - Visão geral do Modelo entidade de relacionamento do site.

III.3 . Interface PHP

O cliente PHP [4] é o responsável por toda a interface do módulo com o seu usuário por meio de páginas HTML geradas dinamicamente e acessíveis via *browser*. As possibilidades de emprego do LegWeb utilizando o PHP [4] são inúmeras e a inclusão de recursos adicionais poderá ser feita a qualquer momento, de acordo com as necessidades funcionais do laboratório.

Ao longo deste trabalho, foi dada prioridade para o desenvolvimento de funcionalidades que permitissem a todos os usuários do módulo obter informações mínimas de gerenciamento de suas rotinas dentro do LEG.

Os principais pontos cobertos pelo cliente PHP [4] serão discutidos a seguir, incluindo desde aspectos da infra-estrutura, contemplando sua integração com SID [1], até as funcionalidades especificadas nos casos de uso disponíveis para o usuário, detalhadas no item III.1 deste trabalho.

III.3.1. Integração com o SID

Todas as páginas do SID [1] são protegidas contra visualização imprópria, ou seja, só poderá visualizá-la se atender a critérios de segurança pré-definidos. Como o LegWeb funcionará como um módulo do SID [1], aproveitou-se a estrutura de autenticação já existente do SID [1], apenas com uma preocupação maior referente à autorização de acesso no módulo.

Conforme já apresentado no item III.2.1, serão acrescentados parâmetros que permitirão identificar o acesso ao LegWeb na tabela que contém dados comuns a todos os usuários do SID [1], ou seja, na *user_base*. Como podemos verificar na figura III.28, os seguintes atributos serão acrescidos a esta tabela: *lw_user_status* (informa se o usuário está ou não autorizado a acessar o módulo), *lw_access_level* (informa o nível de acesso do usuário no módulo) e *lw_last_access* (informa data e hora do último acesso do usuário ao módulo).

Será realizada uma verificação a mais no procedimento de *login* do SID [1]. Quando um usuário acessa o SID [1], o sistema passa a verificar se este está ou não autorizado a acessar o módulo do LegWeb através do atributo *lw_user_status*.

No ato do *login*, o usuário recebe algumas informações cifradas em sua máquina, na forma de variáveis de sessão (através da gravação de c*ookies*). Esses *cookies* são utilizados pelo SID [1] de forma a validar o acesso de um usuário ao módulo, através da função *validateSession()*, nativa do SID [1], declarada no arquivo ***"sid_security.inc.php"***.

Originalmente, o SID [1] gravava as seguintes informações: IP, identificador único do banco de dados, nome, *status* e sexo. Para o módulo LegWeb, será acrescentada a informação "Situação do usuário no módulo". A figura III.57 apresenta o *script* que realiza essa gravação, onde é apresentado em negrito a inclusão referente ao módulo do LegWeb.

```
// Inicia a sessão, guardando as informações pertinentes a ela
session_start();
session_register('session_enc_ip');
$session_enc_ip = encryptSessionVar($REMOTE_ADDR);
session_register('session_enc_user_id');
$session_enc_user_id = encryptSessionVar($row->user_id);
session_register('session_enc_user_name');
$session_enc_user_name = encryptSessionVar($row->name);
session_register('session_enc_user_status');
$session_enc_user_status = encryptSessionVar($row->status_id);
session_register('session_enc_user_gender');
$session_enc_user_gender = encryptSessionVar($row->gender);
//Acesso ao LEG
session_register('session_enc_user_leg');
$session_enc_user_leg = encryptSessionVar($row->lw_user_status);
```

Figura III.57 – Registro inicial de variáveis de sessão.

Após a autenticação, o usuário é levado à página principal do SID [1], com todas as opções originais de uso da ferramenta além do acréscimo da opção de acesso ao módulo do LegWeb. A última opção mencionada só será disponibilizada quando a variável *lw_user_status* possuir valor igual a '1', indicando que o usuário tem acesso ao módulo.

A opção de acesso será disponibilizada através da alteração do arquivo original do SID [1] responsável pelo cardápio de opções de manutenção do sistema, declarado no arquivo *sid_main_left.php*. A figura III.58 apresenta o *script* incluído neste arquivo responsável pela liberação da opção de acesso ao LegWeb.

```
<?
  if(decryptSessionVar($session_enc_user_leg) == "1")
  {
?>
<CENTER>
<I><FONT COLOR="#FFFF00">
<HR> LEG <HR>
</FONT></I>
</CENTER>
<UL TYPE="circle">
<LI><A HREF="/leg/leg_login2.php" TARGET="_blank">Acesso ao LEG</A></LI>
</UL>
<?
  }
```

Figura III.58 – Registro inicial de variáveis de sessão.

III.3.2. Gerenciamento de usuários

As funções de gerenciamento de usuários LegWeb, em termos de codificação PHP [4], podem ser agrupadas nos seguintes grupos: **criação de novo usuário** (grupo, professor e administrador)**, visualizar dados de usuário** (grupo, aluno, professor e administrador)**, excluir usuário** (grupo, professor e administrador), **localizar usuário** (grupo apenas)**, selecionar usuário** (grupo, professor e administrador) e **alterar usuário** (grupo apenas). Todas as funcionalidades e relacionamentos destas funções já foram detalhadas anteriormente nos itens III.1.2, III.1.3 e III.1.4 deste trabalho. Todas possuem comportamento muito semelhante, executando as ações com consultas SQL ao banco de dados do SID [1], ora de consulta, ora de atualização, ora de inserção e ora de remoção.

A codificação para essas funcionalidades é bastante semelhante e, por esta razão, serão apresentados, para efeito didático, os códigos referentes à administração de grupos, dado que a área de **Manutenção de grupo** (figura III.8) apresenta todos os casos de uso acima citados. Sempre que houver uma particularidade na codificação de um caso de uso que o diferencie, seja para a **Manutenção de professor**, seja para a **Manutenção de administrador**, será feita uma análise em separado.

III.3.2.1 . Selecionar grupo

O usuário, ao optar na navegação do módulo pela opção de manutenção de grupo, é levado a uma página onde serão disponibilizadas as opções de manutenção, declaradas no arquivo *leg_group.php*. Neste arquivo, apenas o caso de uso **Selecionar grupo** está declarado e sua codificação comentada é apresentada nas figuras III.59 e III.62.

```
// Pega os grupos do banco

$sql = "select group_id,group_name from lw_group order by group_name";

$result_group = mysql_query($sql) or
  handleError("leg_group.php", mysql_error());
```

Figura III.59 – Codificação do caso de uso "Selecionar grupo".

Na figura III.59 é apresentada a parte do código responsável por realizar uma busca no banco de dados por grupos já cadastrados no módulo do LegWeb. A figura III.60 e III.61 apresentam um código semelhante, entretanto responsável pelos casos de uso **Selecionar administrador** e **Selecionar professor** respectivamente. Esses códigos estão presentes nos arquivos *leg_admin.php* e *leg_prof.php*, nessa ordem.

```
// Grava os administradores do sistema
$sql = "select user_id,login_sid from user_base where lw_access_level = 1 order by
login_sid";

  $result_group = mysql_query($sql) or
    handleError("leg_admin.php", mysql_error());
```

Figura III.60 – Codificação do caso de uso "Selecionar administrador".

Note na figura III.60 que a consulta no banco por usuários é restrita àqueles com o atributo *lw_access_level* com valor '1', ou seja, apenas usuários administradores serão retornados na consulta ao banco.

```
// Grava os professores do sistema
  $sql = "select user_id,login_sid from user_base where lw_access_level = 2 order by
login_sid";

  $result_prof = mysql_query($sql) or
    handleError("leg_prof.php", mysql_error());
```

Figura III.61 – Codificação do caso de uso "Selecionar professor".

Da mesma forma, na figura III.61, a consulta no banco por usuários é restrita àqueles com o atributo *lw_access_level* com valor '2', ou seja, apenas usuários professores serão retornados na consulta ao banco.

```
<FONT SIZE="+1" COLOR="WHITE"><B>Grupos: </B></FONT>
<?
  // Desabilita botões
  $acao_completa = 0;

  // Verifica se existe algum grupo
  if(mysql_num_rows($result_group))
  {
?>
<select name="frm_group_id">
<?
  // Monta o código HTML com os grupos
  while($linha_grupo = mysql_fetch_array($result_group,MYSQL_ASSOC))
    echo "<option
value=\"".$linha_grupo["group_id"]."\">".$linha_grupo["group_name"]."\n";
?>
</select>
<?
        // Habilita botões
    $acao_completa = 1;
        }
        else
        {
?>
<TD><FONT SIZE="+1" COLOR="WHITE"><B>Não há grupo registrado.</B></FONT></TD>
<?
        }
?>
```

Figura III.62 – Codificação do caso de uso "Selecionar grupo".

Na figura III. 62 é apresentada, com comentários, a parte do código responsável por listar para o usuário os grupos já cadastrados no módulo. Caso não haja nenhum grupo cadastrado, o sistema envia a mensagem "Não há grupo registrado". Este código é semelhante àqueles existentes para os casos **Selecionar administrador** e **Selecionar professor**.

III.3.2.2 . Visualizar grupo

Após selecionar qual grupo deseja visualizar, o usuário pressiona o botão "Visualizar Grupo". Ao pressionar este botão, o sistema acessa o código contido no arquivo *leg_action_group.php* onde estão declaradas todas as ações a serem realizadas por cada botão disponibilizado na página. A figura III.63 apresenta a parte do código, com comentários, responsável por estas declarações.

```
// Seleciona ação a ser feita
  switch($btn_acao)
  {
        case "Localizar Grupo":
               header("Location:
leg_view_group_form2.php?frm_login_sid=".$frm_login_sid);
               break;
        case "Visualizar Grupo":
               header("Location: leg_view_group_form.php?frm_group_id=".$frm_group_id);
               break;
        case "Visualizar Aluno":
               header("Location:
leg_view_group_form3.php?frm_login_sid=".$frm_login_sid2);
               break;
        case "Criar Grupo":
               header("Location: leg_add_new_group.php");
               break;
        case "Alterar Grupo":
               header("Location: leg_alter_group_form.php?frm_group_id=".$frm_group_id);
               break;
        case "Apagar Grupo":
               header("Location: leg_del_group_verif.php?frm_group_id=".$frm_group_id);
               break;
        default:
               handleError("leg_group.php", "Ação desconhecida.");
  }
```

Figura III.63 – Codificação das ações a serem realizadas pelos botões de cardápio.

Para o caso de visualização de grupo, o sistema executa o código presente no arquivo *leg_view_group_form.php*, que retorna ao usuário as seguintes informações: "Login dos componentes no SID"[1], "Nome do grupo", "Turma" e "Login do professor no SID"[1]. A figura III.64 apresenta parte do código, com comentários, responsável pela consulta ao banco pelas informações acima apresentadas.

```
// Busca o nome do grupo
  $strSQL = "select group_name,group_class FROM lw_group WHERE group_id = $frm_group_id";
  $result_grupo = mysql_query($strSQL, $link_sid) or
    handleError("leg_group.php", mysql_error());

  // Verifica se achou algum grupo
  if(!mysql_num_rows($result_grupo))
    handleError("leg_group.php", "Nenhum grupo locazidado!");
  $linha_grupo = mysql_fetch_array($result_grupo,MYSQL_ASSOC);

   // Guarda os Alunos associados ao grupo
  $sql = "select a.login_sid from user_base as a, lw_xref_detailed_student_group as b";
  $sql .= " where a.user_id = b.user_id and b.group_id = $frm_group_id";
  $result_aluno = mysql_query($sql) or
    handleError("leg_group.php", mysql_error());
  if(!mysql_num_rows($result_aluno))
    handleError("leg_group.php", "Nenhum aluno associado a esse grupo!");
```

```
 // Monta um array com os alunos do grupo
 for($i=0;$linha_aluno = mysql_fetch_array($result_aluno,MYSQL_NUM);$i++)
   $array_aluno[$i] = $linha_aluno[0];


// Guarda o professor associado ao grupo
 $sql = "select a.login_sid from user_base as a, lw_xref_detailed_professor_group as b";
 $sql .= " where a.user_id = b.user_id and b.group_id = $frm_group_id";

 $result_prof = mysql_query($sql) or
   handleError("leg_group.php", mysql_error());

 if(!mysql_num_rows($result_prof))
   handleError("leg_group.php", "Nenhum professor associado a esse grupo!");

 $linha_prof = mysql_fetch_array($result_prof,MYSQL_NUM);
```

Figura III.64 – Codificação do caso de uso "Visualizar grupo".

III.3.2.3 . Alterar grupo

Após selecionar qual grupo deseja alterar, o usuário pressiona o botão "Alterar Grupo". Ao pressionar este botão, o sistema acessa o código contido no arquivo *leg_action_group.php* onde estão declaradas todas as ações a serem realizadas por cada botão disponibilizado na página. A figura III.63 apresenta a parte do código responsável por estas declarações.

Para o caso de alteração de grupo, o sistema executa o código presente no arquivo *leg_alter_group_form.php*, que retorna ao usuário um formulário permitindo a edição das seguintes informações: "Login dos componentes no SID"[1], "Nome do grupo", "Turma" e "Login do professor no SID"[1]. Após realizar as alterações, ele pressiona o botão "Alterar Grupo", que executa o código presente no arquivo *leg_alter_group_commit.php*. Na figura III.65 é apresentada a parte desse código, com comentários, responsável por gravar as alterações no banco de dados do SID [1].

```
// Coloca num array os logins dos componentes
  $array_login = explode(";",trim($frm_login_sid));

// Verifica no banco se os logins existem
  $strSQL_user_base = "SELECT * FROM user_base WHERE login_sid in
('".implode("','",$array_login)."')";
  $result_user_base = mysql_query($strSQL_user_base) or
    handleError("leg_group.php", mysql_error());

// Cria o array
  $sql_login[0] = "";
  $sql_user_id[0] = "";

// Monta um array com os logins que nao estão no banco
  if (mysql_num_rows($result_user_base) != count($array_login))
  {
    // Cria o array
       $array_tmp[0] = "";
```

```
// Guarda os logins do banco
    for($i=0;$linha = mysql_fetch_array($result_user_base,MYSQL_ASSOC);$i++)
            $sql_login[$i] = $linha["login_sid"];
        $j=0;
        // Verifica a diferença dos arrays
        for($i=0;count($array_login)>$i;$i++)
        {
        // Aquele que não estiver no banco vai para um array temporário
            if(!in_array($array_login[$i],$sql_login))
            {
                $array_tmp[$j] = $array_login[$i];
                $j++;
            }
        }
        // Se houver mais de um elemento no array, coloca um 'e' entre os últimos
        if(count($array_tmp) > 1)
            $str_erro = "Os logins ".str_replace(",".$array_tmp[count($array_tmp)-1]," e
".$array_tmp[count($array_tmp)-1],implode(",",$array_tmp))." não existem.";
        else
            $str_erro = "O login ".$array_tmp[0]." não existe.";

// Envia o erro
        handleError("leg_group.php", $str_erro);
  }
   for($i=0;$linha = mysql_fetch_array($result_user_base,MYSQL_ASSOC);$i++)
  {
        $sql_login[$i] = $linha["login_sid"];
        $sql_user_id[$i] = $linha["user_id"];
  }

  $total_login = $i;

// Verifica no banco se os logins já existem em algum grupo
  $strSQL_user_group = "SELECT distinct(user_id) FROM lw_xref_detailed_student_group WHERE
user_id in ('".implode("','",$sql_user_id)."')";
  $strSQL_user_group .= " AND group_id <> $frm_group_id";
  $result_user_group = mysql_query($strSQL_user_group) or
    handleError("leg_group.php", mysql_error());
  if(mysql_num_rows($result_user_group))
  {
        $i=0;
        $tmp[0] = "";
        while($linha2 = mysql_fetch_row($result_user_group))
        {
            for($j=0;$j<mysql_num_rows($result_user_base);$j++)
            {
                if($linha2[0] == $sql_user_id[$j])
                {
                    $tmp[$i] = $sql_login[$j];
                    $i++;
                }
            }
        }

        handleError("leg_group.php", "O(s) login(s) ".implode(",",$tmp)." está(ão)
cadastrado(s) em outro grupo.");
  }

  // Verifica se o professor existe
  $sql = "SELECT a.user_id FROM user_detailed_professor as a, user_base as b";
  $sql .= " WHERE a.user_id = b.user_id and upper(b.login_sid) = upper('$frm_prof_sid')";
  $result_check_prof = mysql_query($sql) or
    handleError("leg_group.php", mysql_error());

  // Envia erro caso exista
  if(!mysql_num_rows($result_check_prof))
        handleError("leg_group.php", "Não existe professor cadastrado com o login
$frm_prof_sid.");

  $linha_prof = mysql_fetch_array($result_check_prof,MYSQL_NUM);

  // Apaga o grupo
  $sql = "Delete from lw_group where group_id = $frm_group_id";

  $result_check_grupo = mysql_query($sql) or
    handleError("leg_alter_group_form.php", mysql_error());
```

```
// Apaga a relação grupo x aluno
  $sql = "Delete from lw_xref_detailed_student_group where group_id = $frm_group_id";

  $result_check_g_aluno = mysql_query($sql) or
    handleError("leg_alter_group_form.php", mysql_error());

// Apaga a relação grupo x professor
  $sql = "Delete from lw_xref_detailed_professor_group where group_id = $frm_group_id";

  $result_check_g_prof = mysql_query($sql) or
    handleError("leg_alter_group_form.php", mysql_error());

        // Remove as permissões
        $sql = "update user_base set lw_access_level = 0, lw_user_status = 0 where";
        $sql .= " lw_access_level != 1 and user_id in ('".implode("','",$sql_user_id)."')";

        mysql_query($sql) or
    handleError("leg_alter_group_form.php", mysql_error());

  // Recria as relações

  // Monta grupo
  $sql = "Insert into lw_group (group_id,group_name,group_class) values
($frm_group_id,'$frm_grupo','$frm_turma')";

  $result_lw_group = mysql_query($sql) or
    handleError("leg_alter_group_form.php", mysql_error());

  // Monta a relação aluno x grupo
  for($i=0; $i<$total_login; $i++)
  {
        $sql = "Insert into lw_xref_detailed_student_group (user_id,group_id) values
(".$sql_user_id[$i];
        $sql .= ",$frm_group_id)";

        $result_aluno_grupo = mysql_query($sql) or
    handleError("leg_alter_group_form.php", mysql_error());

        $sql = "update user_base set lw_access_level = 4, lw_user_status = 1 where";
        $sql .= " lw_access_level != 1 and user_id = ".$sql_user_id[$i];

        mysql_query($sql) or
    handleError("leg_alter_group_form.php", mysql_error());
  }

  // Monta a relação professor x grupo

  $sql = "Insert into lw_xref_detailed_professor_group (user_id,group_id) values
(".$linha_prof[0];
  $sql .= ",$frm_group_id)";

  $result_prof_grupo = mysql_query($sql) or
    handleError("leg_alter_group_form.php", mysql_error());
?>
```

Figura III.65 – Codificação do caso de uso "Alterar grupo".

III.3.2.4 . Excluir grupo

Após selecionar qual grupo deseja excluir, o usuário pressiona o botão "Excluir Grupo". Ao pressionar este botão, o sistema acessa o código contido no arquivo *leg_action_group.php* onde estão declaradas todas as ações a serem realizadas por cada botão

disponibilizado na página. A figura III.63 apresenta a parte do código responsável por estas declarações.

Após a solicitação de exclusão, o sistema solicita uma confirmação de exclusão, codificada no arquivo *leg_del_group_verif.php* e, em seguida, executa o código contido no arquivo *leg_del_group_commit.php*. A figura III.66 apresenta a parte do código, com comentários, desse arquivo responsável pela exclusão de um grupo da base de dados do SID [1].

```
// Grava os usuários do grupo selecionado que estão somente nesse grupo para assim não
//remover a permissão dos outros integrantes que estão
// em outros grupos
  $sql = "SELECT user_id, group_id FROM lw_xref_detailed_student_group GROUP BY user_id
HAVING";
  $sql .= " count(group_id) = 1 AND group_id = $frm_group_id";

  $result_user = mysql_query($sql, $link_sid) or
    handleError("leg_group.php", mysql_error());

  // Remove a permissão de acesso e o nível de acesso
  while ($linha_user = mysql_fetch_array($result_user))
  {
       $sql = "update user_base set lw_access_level = 0, lw_user_status = 0 where";
       $sql .= " lw_access_level != 1 and user_id = ".$linha_user["user_id"];

       mysql_query($sql, $link_sid) or
    handleError("leg_group.php", mysql_error());
  }
  //
  // Remove usuário de todas as tabelas
  //
  // Relação professor x grupo

  $strSQL = "DELETE FROM lw_xref_detailed_professor_group WHERE group_id = $frm_group_id";
  mysql_query($strSQL, $link_sid) or
    handleError("leg_group.php", mysql_error());

  // Relação aluno x grupo
  $strSQL = "DELETE FROM lw_xref_detailed_student_group WHERE group_id = $frm_group_id";
  mysql_query($strSQL, $link_sid) or
    handleError("leg_group.php", mysql_error());

  // Tabela grupo
  $strSQL = "DELETE FROM lw_group WHERE group_id = $frm_group_id";
  mysql_query($strSQL, $link_sid) or
    handleError("leg_group.php", mysql_error());
?>
```

Figura III.66 – Codificação do caso de uso "Excluir grupo".

III.3.2.5 . Criar grupo

Ao pressionar o botão "Criar Grupo", o sistema acessa o código contido no arquivo *leg_action_group.php* onde estão declaradas todas as ações a serem realizadas por cada botão disponibilizado na página. A figura III.63 apresenta a parte do código responsável por estas declarações. A seguir o sistema apresenta um formulário para preenchimento de informações, declarado no arquivo *leg_add_new_group.php*. Após a digitação, o usuário pressiona o botão "Criar grupo", que leva à execução do código contido no arquivo

leg_add_new_group_commit. A figura III.67 apresenta a parte do código, com comentários, desse arquivo responsável pela criação de um grupo da base de dados do SID [1].

```
// Coloca num array os logins dos componentes
  $array_login = explode(";",trim($frm_login_sid));

  // Verifica no banco se os logins existem
  $strSQL_user_base =  "Select * From user_base Where login_sid in
('".implode("','",$array_login)."')";

  $result_user_base = mysql_query($strSQL_user_base) or
    handleError("leg_group.php", mysql_error());

  // Monta um array com os logins que não estão no banco
  if (mysql_num_rows($result_user_base) != count($array_login))
  {
    // cria o array
       $sql_login[0] = "";
       $array_tmp[0] = "";

       // Guarda os logins do banco
    for($i=0;$linha = mysql_fetch_array($result_user_base,MYSQL_ASSOC);$i++)
              $sql_login[$i] = $linha["login_sid"];

       $j=0;

       // Verifica a diferença dos arrays
       for($i=0;count($array_login)>$i;$i++)
       {
              // aquele q nao estiver no banco vai para um array temporario
              if(!in_array($array_login[$i],$sql_login))
              {
                     $array_tmp[$j] = $array_login[$i];
                     $j++;
              }
       }

       // se tiver mais de um elemento no array entao coloca um 'e' entre os ultimos
       if(count($array_tmp) > 1)
              $str_erro = "Os logins ".str_replace(",".$array_tmp[count($array_tmp)-
1]," e ".$array_tmp[count($array_tmp)-1],implode(",",$array_tmp))." não existem.";
       else
              $str_erro = "O login ".$array_tmp[0]." não existe.";

       // envia o erro
       handleError("leg_group.php", $str_erro);
  }

  // Verifica se os logins já estão cadastrados em algum grupo
  $sql =  "Select b.user_id From user_base a, lw_xref_detailed_student_group b Where
a.login_sid in ('".implode("','",$array_login)."') and b.user_id = a.user_id";

  $result_check_aluno= mysql_query($sql) or
    handleError("leg_group.php", mysql_error());

  // Envia erro caso exista
  if(mysql_num_rows($result_check_aluno))
        handleError("leg_group.php", "Existe(m) aluno(s) cadastrado(s) em outro
grupo.");

// Verifica se o professor existe
  $sql = "SELECT user_id FROM user_base";
  $sql .= " WHERE lw_access_level = 2 and upper(login_sid) = upper('$frm_prof_sid')";

  $result_check_prof = mysql_query($sql) or
    handleError("leg_group.php", mysql_error());

  // Envia erro caso não exista
  if(!mysql_num_rows($result_check_prof))
       handleError("leg_group.php", "Não existe professor cadastrado com o login
$frm_prof_sid.");

  $linha_prof = mysql_fetch_array($result_check_prof,MYSQL_NUM);
```

```
   //
  // Adiciona o grupo ao banco
  //

  $result_lw_group = mysql_query("INSERT INTO lw_group (group_name,group_class) VALUES
('$frm_grupo','$frm_turma')") or
    handleError("leg_group.php", mysql_error());

  if(!mysql_affected_rows())
       handleError("leg_group.php", "Erro desconhecido ao inserir grupo.");

  // Grava o id do grupo
  $id_group = mysql_insert_id();

  while($linha_aluno = mysql_fetch_array($result_user_base,MYSQL_ASSOC))
  {
    // Insere a relação aluno x grupo
       $sql = "INSERT INTO lw_xref_detailed_student_group (user_id,group_id)";
       $sql .= " values (".$linha_aluno["user_id"].",$id_group)";

       $result_aluno_grupo = mysql_query($sql) or
    handleError("leg_group.php", mysql_error());

       if(!mysql_affected_rows())
         handleError("leg_group.php", "Erro desconhecido ao inserir relação aluno x grupo.");
 }

// Insere a relação professor x grupo

  $sql = "Insert into lw_xref_detailed_professor_group (user_id,group_id)";
  $sql .= " values (".$linha_prof[0].",$id_group)";

  $result_prof_grupo = mysql_query($sql) or
    handleError("leg_group.php", mysql_error());

  if(!mysql_affected_rows())
    handleError("leg_group.php", "Erro desconhecido ao inserir relação professor x grupo");

// Libera acesso para os alunos no modulo
// Se um usuário já for administrador então não muda nada, pois esse tem acesso a todo o
//sistema
  $sql = "Update user_base set lw_access_level = 4,lw_user_status = 1 WHERE lw_access_level !=
1 and login_sid in ('".implode("','",$array_login)."')";
  $result_access = mysql_query($sql) or
    handleError("leg_group.php", mysql_error());

?>
```

Figura III.67 – Codificação do caso de uso "Criar grupo".

III.3.2.6 . Localizar grupo

Ao pressionar o botão "Localizar Grupo", o sistema acessa o código contido no arquivo *leg_action_group.php* onde estão declaradas todas as ações a serem realizadas por cada botão disponibilizado na página. A figura III.63 apresenta a parte do código responsável por estas declarações.

Após essa etapa, o usuário informa o *login* de um aluno de forma a verificar se este está cadastrado em um grupo e, em caso positivo, obter do sistema em qual grupo o aluno está cadastrado. O código em questão refere-se a uma consulta no banco de dados, contida no arquivo *leg_view_group_form2.php* . O código comentado encontra-se na figura III.68.

```
// Busca o user id do login
  $strSQL = "SELECT user_id FROM user_base WHERE login_sid = '$frm_login_sid'";
  $result_login = mysql_query($strSQL, $link_sid) or
    handleError("leg_group.php", mysql_error());

  if(!mysql_num_rows($result_login))
       handleError("leg_group.php","Login $frm_login_sid não está cadastrado no
SID.");

  $linha = mysql_fetch_array($result_login);

  $strSQL = "Select group_id from lw_xref_detailed_student_group where user_id =
".$linha["user_id"];
  $result_grupo2 = mysql_query($strSQL, $link_sid) or
    handleError("leg_group.php", mysql_error());
if(!mysql_num_rows($result_grupo2))
       handleError("leg_group.php","Não possui grupo associado.");

  $linha2 = mysql_fetch_array($result_grupo2);
  //
  // Busca o nome do grupo
  //
  $strSQL = "select group_name,group_class FROM lw_group WHERE group_id =
".$linha2["group_id"];
  $result_grupo = mysql_query($strSQL, $link_sid) or
    handleError("leg_group.php", mysql_error());

  // Verifica se encontrou algum grupo
  if(!mysql_num_rows($result_grupo))
    handleError("leg_group.php", "Nenhum grupo locazidado!");

  $linha_grupo = mysql_fetch_array($result_grupo,MYSQL_ASSOC);

  // Grava os alunos associados ao grupo
  $sql = "select a.login_sid from user_base as a, lw_xref_detailed_student_group as
b";
  $sql .= " where a.user_id = b.user_id and b.group_id = ".$linha2["group_id"];

  $result_aluno = mysql_query($sql) or
    handleError("leg_group.php", mysql_error());

  if(!mysql_num_rows($result_aluno))
    handleError("leg_group.php", "Nenhum aluno associado a esse grupo!");

  // Monta um array com os alunos do grupo
  for($i=0;$linha_aluno = mysql_fetch_array($result_aluno,MYSQL_NUM);$i++)
    $array_aluno[$i] = $linha_aluno[0];

  // Grava o professor associado ao grupo
  $sql = "select a.login_sid from user_base as a, lw_xref_detailed_professor_group as
b";
  $sql .= " where a.user_id = b.user_id and b.group_id = ".$linha2["group_id"];

  $result_prof = mysql_query($sql) or
    handleError("leg_group.php", mysql_error());

  if(!mysql_num_rows($result_prof))
    handleError("leg_group.php", "Nenhum professor associado a esse grupo!");

  $linha_prof = mysql_fetch_array($result_prof,MYSQL_NUM);
?>
```

Figura III.68 – Codificação do caso de uso "Localizar grupo".

III.3.2.7 . Visualizar aluno

Ao pressionar o botão "Visualizar aluno", o sistema acessa o código contido no arquivo *leg_action_group.php* onde estão declaradas todas as ações a serem realizadas por

cada botão disponibilizado na página. A figura III.63 apresenta a parte do código responsável por estas declarações.

Após essa etapa, o usuário informa o *login* de um aluno de forma a verificar o seu cadastro, objetivando o retorno do nome do aluno. O código em questão refere-se a uma consulta simples no banco de dados, contida no arquivo *leg_view_group_form3.php* . O código comentado encontra-se na figura III.69.

```
// Busca pelo nome do login
  $strSQL = "SELECT name FROM user_base WHERE login_sid = '$frm_login_sid'";
  $result_login = mysql_query($strSQL, $link_sid) or
    handleError("leg_group.php", mysql_error());

  if(!mysql_num_rows($result_login))
       handleError("leg_group.php","Login $frm_login_sid não está cadastrado no SID.");

  $linha = mysql_fetch_array($result_login);
```

Figura III.69 – Codificação do caso de uso "Visualizar aluno".

III.3.3. Gerenciamento de notas das disciplinas de laboratório

As funções de gerenciamento de notas de disciplinas no LegWeb, em termos de codificação PHP [4], podem ser agrupadas nos seguintes grupos: **Informar graus** (administrador e professor)**, visualizar graus** (aluno, professor e administrador)**, alterar graus** (administrador e professor), **selecionar grupos** (administrador e professor). Todas as funcionalidades e relacionamentos destas funções já foram detalhadas anteriormente nos itens III.1.2.8, III.1.3.3 e III.1.4.3 deste trabalho.

III.3.3.1 . Selecionar grupo

A codificação deste caso de uso encontra-se no arquivo *leg_select_group.php* e encontra-se, em seus itens mais significativos, comentada na figura III.70.

```
//Levantamento da listagem dos grupos existentes
{
               $sql = "select group_id from lw_xref_detailed_student_group";
               $sql .= " where user_id = ".decryptSessionVar($session_enc_user_id);

               $result_group = mysql_query($sql, $link_sid) or
                   handleError("leg_select_group.php", mysql_error());

               $linha = mysql_fetch_array($result_group);

               $querySelect = "SELECT * FROM lw_group where group_id = ".$linha["group_id"];
       }
       else
       {
               $querySelect = "SELECT * FROM lw_group ORDER BY group_name";
       }

  $result = mysql_query($querySelect, $link_sid) or
    handleError("leg_select_group.php", mysql_error());
```

Figura III.70 – Codificação do caso de uso "Selecionar grupo".

III.3.3.2 . Informar graus

A codificação deste caso de uso encontra-se no arquivo *leg_edit_group_grade.php* e encontra-se, em seus itens mais significativos, comentada na figura III.71.

```
//Início do bloco de cadastro de graus
  //Ao informar graus de um usuário acessa o código abaixo
  if(isset($_REQUEST["user_id"]) && isset($_REQUEST["group_id"]) &&
isset($_REQUEST["Confirmar"]) && $_REQUEST["Confirmar"] == "Confirmar")
  {
        //Preparo os graus informados no formulário
        for($col=1; $col<=15; $col++)
                if(isset($_REQUEST["grade$col"]) && !empty($_REQUEST["grade$col"]))
                        $grade[$col] = $_REQUEST["grade$col"];

        //Se ainda não existe registro para esse usuário neste grupo, crio um aqui
        if(!isset($_REQUEST["grade_id"])) {
                $queryInsert = "INSERT INTO lw_grades (";
                $queryInsertValues = " VALUES (";
                for($col=1; $col<=15; $col++) {
                        if(isset($grade[$col])) {
                                $queryInsert .= " grade_$col ";
                                $queryInsertValues .= " ".$grade[$col]." ";
                                $tem_um=true;
                        }
                        if(isset($grade[$col+1]) && $tem_um) {
                                $queryInsert .= ",";
                                $queryInsertValues .= ",";
                        }
                }
                $queryInsert .= ")";
                $queryInsertValues .= ")";

                $queryInsert .= $queryInsertValues;

                $resultInsert = mysql_query($queryInsert, $link_sid) or
                    handleError("leg_select_group.php", mysql_error());

                $grade_id = mysql_insert_id();

                $queryInsert2 = "INSERT INTO lw_xref_detailed_student_grade ( user_id, grade_id
)";
                $queryInsert2 .= " values ( ".$_REQUEST["user_id"]."., $grade_id )";

                $resultInsert2 = mysql_query($queryInsert2, $link_sid) or
                    handleError("leg_select_group.php", mysql_error());

        }
        else // Se já existe registro, somente atualizo
        {
                $queryUpdate = "UPDATE lw_grades SET ";
                for($col=1; $col<=15; $col++) {
                        if(isset($grade[$col])) {
                                $queryUpdate .= " grade_$col = ".$grade[$col]." ";
                                $tem_um=true;
                        }
                        if(isset($grade[$col+1]) && $tem_um)
                                $queryUpdate .= ",";
                }
                $queryUpdate .= " WHERE grade_id = ".$_REQUEST["grade_id"];

                $resultUpdate = mysql_query($queryUpdate, $link_sid) or
                    handleError("leg_select_group.php", mysql_error());

        }

  }
//Fim do bloco de cadastro de graus
```

Figura III.71 – Codificação do caso de uso "Informar graus".

III.3.3.3 . Alterar graus

A codificação deste caso de uso é idêntica à do item anterior (III.3.3.2).

III.3.3.4 . Visualizar graus

A codificação deste caso de uso encontra-se no arquivo *leg_view_group.php* e encontra-se, em seus itens mais significativos, comentada na figura III.72.

```
//Guardo todos os usuários do grupo informado
  $querySelect = "  SELECT DSGRO.* , DSGRA.grade_id ";
  $querySelect .= " FROM lw_xref_detailed_student_group DSGRO  ";
  $querySelect .= " LEFT JOIN  ";
  $querySelect .= " lw_xref_detailed_student_grade DSGRA ";
  $querySelect .= " ON DSGRO.user_id = DSGRA.user_id ";
  $querySelect .= " WHERE DSGRO.group_id=".$_REQUEST["group_id"];
  $querySelect .= " ORDER BY user_id ";
  $result = mysql_query($querySelect, $link_sid) or
    handleError("leg_select_group.php", mysql_error());
//Guardo o nome do grupo
  $querySelect2 = "SELECT group_name FROM lw_group WHERE group_id=".$_REQUEST["group_id"];

  $result2 = mysql_query($querySelect2, $link_sid) or
    handleError("leg_select_group.php", mysql_error());

  $group = mysql_fetch_array($result2);
?>
// Retorno o nome do grupo
Grupo: <? echo $group["group_name"] ?>

<? for( $col=1; $col<=15; $col++) { ?>
    <td width="5%" align="center"><strong><font color="#FFFFFF" size="-1">Grau <? echo
$col; ?></font></strong></td>
    <? } ?>
  </tr>
  <?
    $quant = 1;
        while( $user= mysql_fetch_array($result) )
        {
                if($user["grade_id"] != NULL)
                {
                        $querySelectGrades = " SELECT * FROM lw_grades WHERE grade_id =
".$user["grade_id"];

                        $resultSelectGrades = mysql_query($querySelectGrades, $link_sid);
                        $user_grades = mysql_fetch_array($resultSelectGrades);
                }
                if($quant++%2==1) $cor = "#EAFFEA"; else $cor = "#FFFFFF";
  ?>
   <tr bgcolor="<? echo $cor; ?>">
        <td width="2%"></TD>
    <td><?
                        $result3 = mysql_query("Select login_sid from user_base where
user_id = " . $user["user_id"],$link_sid);

                        $user_name = mysql_fetch_array($result3);

         echo $user_name["login_sid"]; ?></td>
    <? for( $col=1; $col<=15; $col++) { ?>
    <td align="center"><font size="-1"><? if ($user["grade_id"] != NULL) echo
number_format($user_grades["grade_$col"],2); else echo "0.00"; ?></font></td>
    <? } ?>
  </tr>
  <?
    }
  ?>
```

Figura III.72 – Codificação do caso de uso "Visualizar graus".

III.3.4. Gerenciamento de *e-mail*

As funções de gerenciamento de *e-mail* no LegWeb, em termos de codificação PHP [4], podem ser agrupadas nos seguintes grupos: **Ler mensagem**, **Selecionar mensagem** e **Excluir mensagem** . Todas as funcionalidades e relacionamentos destas funções já foram detalhadas anteriormente no item III.1.2.7 deste trabalho.

III.3.4.1 . Selecionar mensagem

A codificação deste caso de uso encontra-se no arquivo *leg_select_message.php* e encontra-se, em seus itens mais significativos, comentada na figura III.73.

```
//Captura informações sobre mensagens disponíveis
  $querySelect = "select a.*,c.login_sid,c.user_id ";
  $querySelect .= "from lw_messages a, lw_xref_messages_user b, user_base c ";
  $querySelect .= "where (receiver_id = ".decryptSessionVar($session_enc_user_id);
  $querySelect .= " or receiver_id = 0) and b.message_id = a.message_id ";
  $querySelect .= "and c.user_id = b.user_id";

  $result = mysql_query($querySelect, $link_sid) or
    handleError("leg_select_message.php", mysql_error());

?>
// Apresenta mensagens em tela
<?
    $quant = 1;
       while( $message = mysql_fetch_array($result) )
       {
// Tratamento do formato de letra: se status = 1 (já foi lida) a formatação de texto é
//diferente
               if($quant%2==1) $cor = "#EAFFEA"; else $cor = "#FFFFFF";
               $read = "";
               if($message["status"]==1) $read = "class='msgRead'";
  ?>
  <tr bgcolor="<? echo $cor ?>" <? echo $read ?>>
    <td><a href="leg_view_message.php?message_id=<? echo $message["message_id"] ?>"><img
src="images/button_view.jpg" alt="Ler Mensagem" width="38" height="26"
border="0"></a></td>
    <td><? echo $message["login_sid"]." (".$message["user_id"].")" ?></td>
    <td><? echo "<a $read
href='leg_view_message.php?message_id=".$message["message_id"]."'>".$message["subject"]."<
/a>"; ?></td>
    <td><? echo date("d/m/Y H:i:s",strtotime($message["date"])) ?></td>
  </tr>
  <?
    }
  ?>
```

Figura III.73 – Codificação do caso de uso "Selecionar mensagem".

III.3.4.2 . Ler mensagem

A codificação deste caso de uso encontra-se no arquivo *leg_view_message.php* e encontra-se, em seus itens mais significativos, comentada na figura III.74.

```
// Realiza a busca pela mensagem selecionada
$queryUpdate = "UPDATE lw_messages set status = 1 WHERE
message_id=".$_REQUEST["message_id"];

  $resultUpdate = mysql_query($queryUpdate, $link_sid) or
    handleError("leg_select_message.php", mysql_error());

  $querySelect = "select a.*,c.login_sid,c.user_id ";
  $querySelect .= "from lw_messages a, lw_xref_messages_user b, user_base c ";
  $querySelect .= "where a.message_id = ".$_GET["message_id"];
  $querySelect .= " and b.message_id = a.message_id ";
  $querySelect .= "and c.user_id = b.user_id";

  $result = mysql_query($querySelect, $link_sid) or
    handleError("leg_select_message.php", mysql_error());

  $message = mysql_fetch_array($result);
?>

//Informa em tela os dados da mensagem
<FONT COLOR="#004000" SIZE="+2"><B>Ler Mensagem</B></FONT>
<FONT SIZE="-1">
<CENTER>(Nível Administrador)</CENTER></FONT>
<IMG SRC="images/line_green.gif" WIDTH="300" HEIGHT="2" BORDER="0"><BR>

<table width="100%" border="0" cellspacing="0" cellpadding="0">
  <tr>
    <td width="10%"><strong>De:</strong></td>
    <td width="90%"><? echo $message["login_sid"]." (".$message["user_id"].")" ?></td>
  </tr>
  <tr>
    <td><strong>Data:</strong></td>
    <td><? echo date("d/m/Y H:i:s",strtotime($message["date"])) ?></td>
  </tr>
  <tr>
    <td><strong>Assunto:</strong></td>
    <td><? echo $message["subject"] ?></td>
  </tr>
  <tr>
    <td colspan="2"><PRE><? echo $message["text"] ?></PRE>
       </td>
  </tr>
</table>
<br>

<form name="form1" action="" method="post">
<input name="message_id" type="hidden" value="<? echo $_GET["message_id"] ?>">
<input name="user_id" type="hidden" value="<? echo $message["user_id"] ?>">

//Link para excluir a mensagem
<img src="images/button_delete.jpg" width="38" height="26" alt="Excluir Mensagem"
border="0" onClick="javascript: form1.action='leg_delete_message.php'; form1.submit();
"></a>
```

Figura III.74 – Codificação do caso de uso "Ler mensagem".

III.3.4.3 .Excluir mensagem

A codificação deste caso de uso encontra-se no arquivo *leg_delete_message_commit.php* e encontra-se, em seus itens mais significativos, comentada na figura III.75.

```
// Remove da base a mensagem selecionada
        //Remove da tabela de referência cruzada
$queryDelete = "DELETE FROM lw_xref_messages_user WHERE message_id =
".$_REQUEST["message_id"];

  $result_msg = mysql_query($queryDelete) or
    handleError("leg_select_message.php", mysql_error());

  if(mysql_affected_rows()!=1) echo mysql_error();
        //Remove a mensagem
  $queryDelete2 = "DELETE FROM lw_messages WHERE message_id = ".$_REQUEST["message_id"];
  $result_msg2 = mysql_query($queryDelete2) or
    handleError("leg_select_message.php", mysql_error());

  if(mysql_affected_rows()!=1) echo mysql_error();

?>
//Informa que a mensagem foi excluída com sucesso
<?      if(mysql_affected_rows()==1)
                echo "Mensagem excluída com sucesso!";
?>
```

Figura III.75 – Codificação do caso de uso "Excluir mensagem".

III.3.5. Gerenciamento de categoria de insumos do laboratório

As funções de gerenciamento de categoria no LegWeb, em termos de codificação PHP [4], podem ser agrupadas nos seguintes grupos: **Selecionar categoria**, **Visualizar categoria**, **Criar categoria**, **Alterar categoria** e **Excluir categoria** . Todas as funcionalidades e relacionamentos destas funções já foram detalhadas anteriormente no item III.1.2.2 deste trabalho.

III.3.5.1 .Selecionar categoria

A codificação deste caso de uso encontra-se no arquivo *leg_cat.php* e encontra-se, em seus itens mais significativos, comentada na figura III.76.

```
// Guarda as categorias a partir da categoria pai e do tipo (componente ou equipamento)

if(!IsSet($frm_rank_id))
        $frm_rank_id = 0;

  if(!IsSet($frm_rank_type))
        $frm_rank_type = 0;

    $sql = "select rank_id,rank_name,father_code from lw_rank where father_code =
$frm_rank_id";
  $sql .= " and rank_type= $frm_rank_type order by rank_name";

  $result_rank = mysql_query($sql) or
    handleError("leg_cat.php", mysql_error());
?>
```

```
// Em caso de erro na carga das categorias informa erro em tela
<?
  if (IsSet($error))
    {
?>

<TABLE ALIGN="center" BGCOLOR="#FF8080" CELLSPACING="0" CELLPADDING="5" BORDER="1">
<TR>
<TD ALIGN="center">

<CENTER>
<B><U>Erro</U></B>: <I><? print($error); ?></I>


<?
       // se o id for zero, então pertence à raiz da árvore
       if($frm_rank_id)
       {
              // Guarda a categoria
              $sql = "select rank_name from lw_rank where rank_id = $frm_rank_id and
rank_type= $frm_rank_type";

              $result_cat = mysql_query($sql) or
              handleError("leg_cat.php", mysql_error());

              $linha_cat = mysql_fetch_array($result_cat,MYSQL_ASSOC);

              // Retorna o nome da categoria
              echo $linha_cat["rank_name"];

              // habilita botões
              $acao_completa = 1;
       }
       else
       {
              // desabilita botões
              $acao_completa = 0;
              echo "Raiz";
       }
?>
<?
  // Total de categorias derivadas
  $total = mysql_num_rows($result_rank);
  if($total)
  {
       // Monta o código HTML com as categorias derivadas
       for($i=0;$i<$total;$i++)
       {
       $linha_rank = mysql_fetch_array($result_rank,MYSQL_ASSOC);
       echo "<option
value=\"".$linha_rank["rank_id"]."\">".$linha_rank["rank_name"]."\n";
              $array_rank[$i] = $linha_rank["rank_id"];
       }

       // se a categoria pai não for a raiz, então permite volta (subir um nível na
       //árvore)
       if($linha_rank["father_code"])
       {
              // Guarda a categoria avô (categoria abaixo da categoria pai)
       $sql = "Select father_code from lw_rank where rank_id =
".$linha_rank["father_code"];
              $sql .= " and rank_type= $frm_rank_type";

              $result_avo = mysql_query($sql) or
       handleError("leg_cat.php", mysql_error());

              $linha_avo = mysql_fetch_array($result_avo,MYSQL_ASSOC);
?>
  <option value="<? echo $linha_avo["father_code"]; ?>">Voltar
<?
       }
  }
  else
  {
```

```
        // Trata categoria avô
    $sql = "Select father_code from lw_rank where rank_id = $frm_rank_id and rank_type=
$frm_rank_type";

        $result_avo = mysql_query($sql) or
      handleError("leg_cat.php", mysql_error());

        $linha_avo = mysql_fetch_array($result_avo,MYSQL_ASSOC);
?>
  <option value="<? echo $linha_avo["father_code"]; ?>">Não há ramificações - Voltar
<?
        }
?>
```

Figura III.76 – Codificação do caso de uso "Selecionar categoria".

III.3.5.2 .Visualizar categoria

A codificação deste caso de uso está inclusa no item anterior, o III.3.5.1, onde a navegação na árvore de categoria de insumos de laboratório se dá através da seleção de ramificações prevista dentro do código.

III.3.5.3 .Criar categoria

A codificação deste caso de uso encontra-se em dois arquivos: o primeiro, *leg_add_new_cat.php* contém o código com o formulário a ser preenchido pelo usuário para criar uma nova categoria. O segundo, o *leg_add_new_cat_commit.php*, é quem realiza a gravação no banco de dados e encontra-se, em seus itens mais significativos, comentada na figura III.77.

```
// Verifica se tem categorias irmãs (mesmo nível na árvore) com o mesmo nome
  $sql = "Select rank_id from lw_rank where father_code = $frm_rank_id and rank_name =
'$frm_rank_name' and rank_type= $frm_rank_type";
  $result_teste = mysql_query($sql) or
        handleError("leg_cat.php", mysql_error());

  if(mysql_num_rows($result_teste))
  {
        handleError("leg_cat.php", "Existe já uma categoria com esse nome!");
  }

  // Atualiza nível de acesso (relacionamento entre categorias)

  $sql = "Insert into lw_rank (rank_type,rank_name,father_code) values
($frm_rank_type,'$frm_rank_name',$frm_rank_id)";

  $result_user = mysql_query($sql) or
    handleError("leg_cat.php", mysql_error());
?>
```

Figura III.77 – Codificação do caso de uso "Criar categoria".

III.3.5.4 .Alterar categoria

A codificação deste caso de uso encontra-se em dois arquivos: o primeiro, *leg_alter_cat_form.php* contém o código com o formulário a ser preenchido pelo usuário para alterar uma categoria. O segundo, o *leg_alter_cat_commit.php*, é quem realiza a gravação das alterações no banco de dados e encontra-se, em seus itens mais significativos, comentada na figura III.78.

```
// Verifica se tem categorias irmãs (mesmo nível hierárquico) com o mesmo nome
  $sql = "Select rank_id from lw_rank where father_code = $frm_rank_id2 and rank_name =
'$frm_rank_name' and rank_type = $frm_rank_type2";
  $result_teste = mysql_query($sql) or
       handleError("leg_cat.php", mysql_error());

  if(mysql_num_rows($result_teste))
  {
       handleError("leg_cat.php", "Existe já uma categoria com esse nome!");
  }

  // Atualiza nome e categoria pai em na tabela lw_rank
  $sql = "Update lw_rank set rank_name='$frm_rank_name',father_code=$frm_rank_id2";
  $sql .= ",rank_type = $frm_rank_type2 where rank_id = $frm_rank_id and rank_type =
$frm_rank_type";

  $result_rank = mysql_query($sql) or
    handleError("leg_cat.php", mysql_error());
?>
//Informa se a alteração foi concluída com sucesso
<CENTER>Categoria <I><? echo ($frm_rank_name); ?></I> alterado com sucesso </CENTER>
```

Figura III.78 – Codificação do caso de uso "Alterar categoria".

III.3.5.5 .Excluir categoria

A codificação deste caso de uso encontra-se em dois arquivos: o primeiro, *leg_del_cat_verif.php* contém o código solicitando a confirmação de exclusão de uma categoria. O segundo, o *leg_del_cat_commit.php*, é quem realiza a exclusão no banco de dados e encontra-se, em seus itens mais significativos, comentada na figura III.79.

```
  // Busca os filhos da categoria
  $strSQL = "select rank_id FROM lw_rank WHERE father_code = $frm_rank_id and
rank_type=$frm_rank_type";

  $result_filhos = mysql_query($strSQL, $link_sid) or
    handleError("leg_cat.php", mysql_error());
// Monta um array com os ids que não devem ser usados (aqueles que são os filhos do id
//atual e ele mesmo)
  $nao_usar_id[0] = $frm_rank_id;

  while(mysql_num_rows($result_filhos))
  {
       $j = 0;
       // Grava os filhos dos filhos
       for($i=count($nao_usar_id);$linha_filhos =
mysql_fetch_array($result_filhos,MYSQL_ASSOC);$i++)
       {
              // Gravo no array para não ser usado
              $nao_usar_id[$i] = $linha_filhos["rank_id"];

              // Gravo num array temporário para guardar os filhos dele também
              $temp_id[$j] = $linha_filhos["rank_id"];
              $j++;
       }
```

```
// Gravo os filhos dos filhos
        $strSQL = "select rank_id FROM lw_rank WHERE father_code in
(".implode(",",$temp_id).") and rank_type=$frm_rank_type";
        $result_filhos = mysql_query($strSQL, $link_sid) or
        handleError("leg_cat.php", mysql_error());

        // Desfaz o array temporário
        array_splice($temp_id,0);
  }

  if(!$frm_rank_type)
  {
          // Apaga os componentes
         $sql = "Delete from lw_component where component_rank_id in
(".implode(",",$nao_usar_id).")";
         mysql_query($sql, $link_sid) or
        handleError("leg_cat.php", mysql_error());
  }
  else
  {
        // Apaga os equipamentos
         $sql = "Delete from lw_equipment where equip_rank_id in
(".implode(",",$nao_usar_id).")";
         mysql_query($sql, $link_sid) or
        handleError("leg_cat.php", mysql_error());
  }

  // Apaga as categorias
  $sql = "Delete from lw_rank where rank_id in (".implode(",",$nao_usar_id).")";
  mysql_query($sql, $link_sid) or
    handleError("leg_cat.php", mysql_error());
?>
```

Figura III.79 – Codificação do caso de uso "Excluir categoria".

III.3.6. Gerenciamento de empréstimos

As funções de gerenciamento de empréstimos no LegWeb, em termos de codificação PHP [4], podem ser agrupadas nos seguintes grupos: **Selecionar empréstimo** (administrador, professor e aluno)**, Alterar empréstimo** (administrador), **Selecionar tipo de empréstimo** (administrador, professor e aluno)**, Criar empréstimo** (administrador, professor e aluno) e **Visualizar empréstimo** (professor e aluno). Todas as funcionalidades e relacionamentos destas funções já foram detalhadas anteriormente nos itens III.1.2.6, III.1.3.4 e III.1.4.4 deste trabalho. A codificação para os casos de uso do gerenciamento de componentes e equipamentos é bastante semelhante e serão apresentadas abaixo usando como padrão o gerenciamento de componentes.

III.3.6.1 .Selecionar empréstimo

A codificação deste caso de uso encontra-se em três arquivos: o primeiro, *leg_borrow.php* contém o código solicitando que o usuário informe se está tratando do empréstimo de um componente ou de um equipamento. O segundo, o *leg_action_borrow.php*,

informa qual a ação deve ser tomada de acordo com a seleção do usuário. A sua codificação comentada encontra-se na figura III.80.

```
// Seleciona a ação a ser feita conforme opção do usuário vinda de leg_borrow.php
  switch($btn_acao)
  {
        // Opção gerenciar componente
        case "Componente":
                header("Location: leg_borrow_comp.php?frm_login=".$frm_login);
                break;
        // Opção gerenciar equipamento
        case "Equipamento":
                header("Location: leg_borrow_equip.php?frm_login=".$frm_login);
                break;
        // Caso de uso criar empréstimo
        case "Criar Empréstimo":
                if ($tipo == "equipamento")
                        header("Location:
leg_add_new_borrow_equip.php?frm_login=".$frm_login);
                else
                        header("Location: leg_add_new_borrow_comp.php?frm_login=".$frm_login);

                break;

        // Caso de uso Alterar empréstimo
        case "Alterar Empréstimo":
                if ($tipo == "equipamento")
                        header("Location:
leg_alt_borrow_equip.php?frm_borrow_id=".$frm_borrow_id);
                else
                        header("Location:
leg_alt_borrow_comp.php?frm_borrow_id=".$frm_borrow_id);

                break;

        //Caso de uso excluir empréstimo
        case "Apagar Empréstimo":
                if ($tipo == "equipamento")
                        header("Location:
leg_del_borrow_equip_verif.php?frm_borrow_id=".$frm_borrow_id);
                else
                        header("Location:
leg_del_borrow_comp_verif.php?frm_borrow_id=".$frm_borrow_id);

                break;

        //Caso de uso visualizar empréstimo
        case "Visualizar Empréstimo":
                if ($tipo == "equipamento")
                        header("Location:
leg_view_borrow_equip.php?frm_borrow_id=".$frm_borrow_id);
                else
                        header("Location:
leg_view_borrow_comp.php?frm_borrow_id=".$frm_borrow_id);

                break;
        default:
                handleError("leg_group.php", "Ação desconhecida.");
  }
?>
```

Figura III.80 – Codificação das ações a serem realizadas pelos botões de cardápio.

O terceiro arquivo, o *leg_borrow_comp.php*, é quem efetivamente apresenta a opção de seleção do empréstimo já cadastrado. A figura III.81 apresenta o código comentado do caso de uso em questão.

```
// Abre a conexão com o Banco de Dados
  $link_sid = mysql_connect($MYSQL_LOCATION, $MYSQL_ADMIN_USERNAME,
$MYSQL_ADMIN_PASSWORD);
  if (!mysql_select_db($MYSQL_DBNAME))
    handleError("leg_borrow.php", mysql_error());
if (validateSession(1) == TRUE)
  {
        // Seleciona os ids dos empréstimos de componente
        $sql = "Select a.borrow_comp_id from lw_xref_borrow_comp_user a, user_base b";
        $sql .= " where a.user_id = b.user_id and b.login_sid = '".$frm_login."'";
  }
  else
  {
        // Seleciona os ids dos empréstimos de componente
        $sql = "Select borrow_comp_id from lw_xref_borrow_comp_user";
        $sql .= " where user_id = ".decryptSessionVar($session_enc_user_id);
  }

  $result_comp = mysql_query($sql) or
        handleError("leg_borrow.php", mysql_error());
?>

<?
  // desabilita botões
  $acao_completa = 0;

  // Verifica se tem algum componente na seleção
  if(mysql_num_rows($result_comp))
  {
?>
<TD>
<select name="frm_borrow_id">
<?
  // Monta o código HTML com os empréstimos de componente
  while($linha_comp = mysql_fetch_array($result_comp,MYSQL_NUM))
    echo "<option value=\"".$linha_comp[0]."\">".$linha_comp[0]."\n";
?>
</select>
</TD>
<?
        // habilita botões
    $acao_completa = 1;
        }
        else
        {
?>
<TD><FONT SIZE="+1" COLOR="WHITE"><B>Não há empréstimo registrado.</B></FONT></TD>
<?
        }
?>
```

Figura III.81 – Codificação do caso de uso "Selecionar empréstimo".

III.3.6.2. Alterar empréstimo

Ao pressionar o botão "Alterar empréstimo", o sistema acessa o código contido no arquivo *leg_borrow.php* onde estão declaradas todas as ações a serem realizadas por cada botão disponibilizado na página. A figura III.80 apresenta a parte do código responsável por estas declarações.

Após essa etapa, o usuário altera as informações do empréstimo através de dois arquivos. No primeiro, o *leg_alt_borrow_comp*, são apresentadas as informações pré-cadastradas do empréstimo, permitindo sua edição. A figura III.82 apresenta o código responsável por esta função.

```
<?

// Apresenta o componente cadastrado no empréstimo
        $sql = "Select component_id from lw_xref_borrow_comp where borrow_comp_id =
$frm_borrow_id";

        $result_borrow = mysql_query($sql) or
                handleError("leg_borrow.php", mysql_error());

        $linha_borrow = mysql_fetch_array($result_borrow,MYSQL_ASSOC);

        // Guarda componente

        $sql = "Select component_id,component_description from lw_component";
        $sql .= " order by component_description";

        $result_comp = mysql_query($sql) or
                handleError("leg_borrow.php", mysql_error());

        if (mysql_num_rows($result_comp))
        {
                // Apresenta em tela o componente

                echo "<select name=\"componente\">\n";
                while ($linha_comp = mysql_fetch_array($result_comp,MYSQL_ASSOC))
                {
                        echo "<option value=\"".$linha_comp["component_id"]."\"";
                        if ($linha_comp["component_id"] == $linha_borrow["component_id"])
                                echo " selected";
                        echo ">".$linha_comp["component_description"]."</option>\n";
                }
                echo "</select>\n";
        }
        else
        {
                echo "Não existem componentes.\n";
        }

        //Consulta por quantidade de components, data de empréstimo e status de entrega

        $sql = "Select borrow_comp_date,borrow_comp_quant,borrow_status";
        $sql .= " from lw_borrow_comp where borrow_comp_id = $frm_borrow_id";

        $result_borrow = mysql_query($sql) or
                handleError("leg_borrow.php", mysql_error());

        $linha_borrow = mysql_fetch_array($result_borrow,MYSQL_ASSOC);
?>
```

Figura III.82 – Codificação da impressão em tela das informações de empréstimo a alterar.

O segundo arquivo, o *leg_alt_borrow_comp_commit.php* contém o código responsável por guardar as alterações no banco de dados. A figura III.83 apresenta o código responsável por esta atividade.

```
// Abre a conexão com o Banco de Dados
  //
  $link_sid = mysql_connect($MYSQL_LOCATION, $MYSQL_ADMIN_USERNAME,
$MYSQL_ADMIN_PASSWORD);
  if (!mysql_select_db($MYSQL_DBNAME))
    handleError("leg_borrow.php", mysql_error());

  // Atualiza empréstimo
  $sql = "Update lw_borrow_comp set borrow_comp_date='$frm_data',";
  $sql .= "borrow_comp_quant=$frm_quantidade,borrow_status=$frm_status";
  $sql .= " where borrow_comp_id = $frm_borrow_id";

  $result_emprestimo = mysql_query($sql) or
       handleError("leg_borrow.php", mysql_error());

  // Atualiza relacionamento entre empréstimo e componente
  $sql = "Update lw_xref_borrow_comp set component_id = $componente";
  $sql .= " where borrow_comp_id = $frm_borrow_id";

  $result_emprestimo2 = mysql_query($sql) or
       handleError("leg_borrow.php", mysql_error());
?>
```

Figura III.83 – Codificação do caso de uso "Alterar empréstimo".

III.3.6.3. Selecionar tipo de empréstimo

A codificação deste caso de uso está inclusa no item III.3.6.1, onde o usuário seleciona se quer gerenciar o empréstimo de componentes ou equipamentos através do código declarado na figura III.80.

III.3.6.4. Criar empréstimo

Ao pressionar o botão "Criar empréstimo", o sistema acessa o código contido no arquivo *leg_borrow.php* onde estão declaradas todas as ações a serem realizadas por cada botão disponibilizado na página. A figura III.80 apresenta a parte do código responsável por estas declarações.

Este caso de uso é definido pelo código contido em dois arquivos. O primeiro é o *leg_add_new_borrow_comp.php*, onde são apresentados ao usuário os componentes disponíveis para empréstimo. A figura III.84 apresenta o código comentado presente nesse arquivo.

```
<?
        //Captura informação sobre os components disponíveis para empréstimo
        $sql = "Select component_id,component_description from lw_component";
        $sql .= " order by component_description";

        $result_comp = mysql_query($sql) or
                handleError("leg_borrow.php", mysql_error());

        if (mysql_num_rows($result_comp))
        {
                echo "<select name=\"componente\">\n";
                while ($linha_comp = mysql_fetch_array($result_comp,MYSQL_ASSOC))
                        echo "<option
value=\"".$linha_comp["component_id"]."\">".$linha_comp["component_description"]."</option>\n"
;
                echo "</select>\n";
        }
        else
        {
                echo "Não existem equipamentos.\n";
        }
?>
// Coleta informação sobre a quantidade desejada para empréstimo pelo usuário
<TR><TD ALIGN="RIGHT">
<FONT SIZE="+1" COLOR="WHITE"><B>Quantidade: </B></FONT>
</td>
<TD>
<INPUT TYPE="TEXT" NAME="frm_quantidade" SIZE="10" onKeyPress="checkAutoSkip(this);">
</TD>
</TR>

</TABLE></TD></TR>

<TR><TD><CENTER><INPUT TYPE="SUBMIT" NAME="btn_addgroup" VALUE="Criar
Empréstimo"></CENTER></TD></TR>
</td></tr>
<input type="Hidden" name="frm_login" value="<? echo $frm_login; ?>">
</FORM>
</TABLE>
```

Figura III.84 – Codificação da impressão em tela dos componentes disponíveis para empréstimo.

O segundo, o *leg_add_new_borrow_comp_commit.php*, é o responsável por armazenar no banco de dados o empréstimo do componente. A figura III.85 apresenta o código comentado contido neste arquivo.

```
   // Grava informações de empréstimo
   $sql = "Insert into lw_borrow_comp (borrow_comp_date,borrow_comp_quant,borrow_status)";
   $sql .= " values (now(),$frm_quantidade,1)";

   $result_emprestimo = mysql_query($sql,$link_sid) or
        handleError("leg_borrow.php", "bla".mysql_error());

   $emprestimo_id = mysql_insert_id();

   // Atualiza informações do relacionamento empréstimo e componente
   $sql = "Insert into lw_xref_borrow_comp (component_id,borrow_comp_id)";
   $sql .= " values ($componente,$emprestimo_id)";

   $result_emprestimo2 = mysql_query($sql,$link_sid) or
        handleError("leg_borrow.php","ble".mysql_error());

   if (validateSession(1) == TRUE)
    {
```

```
// Guarda o id do usuário
  $sql = "Select user_id from user_base where login_sid = '".$frm_login."'";

  $result_user = mysql_query($sql,$link_sid) or
       handleError("leg_borrow.php", mysql_error());

  $linha_user = mysql_fetch_array($result_user,MYSQL_ASSOC);

  // Atualiza o relacionamento entre usuário e componente
  $sql = "Insert into lw_xref_borrow_comp_user (user_id,borrow_comp_id) values (";
  $sql .= $linha_user["user_id"].",$emprestimo_id)";
  }
  else
  {
  $sql = "Insert into lw_xref_borrow_comp_user (user_id,borrow_comp_id) values (";
  $sql .= decryptSessionVar($session_enc_user_id).",$emprestimo_id)";
  }

  $result_emprestimo3 = mysql_query($sql,$link_sid) or
       handleError("leg_borrow.php", "bli".mysql_error());
?>
```

Figura III.85 – Codificação do caso de uso "Criar empréstimo".

III.3.6.5. Visualizar empréstimo

Ao pressionar o botão "Criar empréstimo", o sistema acessa o código contido no arquivo *leg_borrow.php* onde estão declaradas todas as ações a serem realizadas por cada botão disponibilizado na página. A figura III.80 apresenta a parte do código responsável por estas declarações.

Este caso de uso é definido pelo código contido no arquivo *leg_view_borrow_comp.php*. A figura III.86 apresenta o código comentado presente nesse arquivo.

```
<TR><TD>
<TABLE BORDER="0" ALIGN="CENTER" BGCOLOR="#004000">
<TR><TD ALIGN="RIGHT">
              <FONT SIZE="+1" COLOR="WHITE"><B>Componente: </B></FONT>
              </td>
              <TD><FONT SIZE="+1" COLOR="WHITE"><B>
//Retorna em tela o componente emprestado e seleciona suas informações de empréstimo
//(data de empréstimo, quantidade e status do empréstimo)
<?
       echo $frm_borrow_id;

       $sql = "Select borrow_comp_date,borrow_comp_quant,borrow_status";
       $sql .= " from lw_borrow_comp where borrow_comp_id = $frm_borrow_id";

       $result_borrow = mysql_query($sql) or
              handleError("leg_borrow.php", mysql_error());

       $linha_borrow = mysql_fetch_array($result_borrow,MYSQL_ASSOC);
?></B></FONT>
```

```
//Imprime informações do empréstimo na tela
<TR><TD ALIGN="RIGHT">
<FONT SIZE="+1" COLOR="WHITE"><B>Quantidade: </B></FONT>
</td>
<TD><FONT SIZE="+1" COLOR="WHITE"><B>
<? echo $linha_borrow["borrow_comp_quant"]; ?></B></FONT>
</TD>
</TR>

<TR><TD ALIGN="RIGHT">
<FONT SIZE="+1" COLOR="WHITE"><B>Data: </B></FONT>
</td>
<TD><FONT SIZE="+1" COLOR="WHITE"><B>
<? echo $linha_borrow["borrow_comp_date"]; ?></B></FONT>
</TD>
</TR>

<TR><TD ALIGN="RIGHT">
<FONT SIZE="+1" COLOR="WHITE"><B>Status: </B></FONT>
</td>
<TD><FONT SIZE="+1" COLOR="WHITE"><B>
<?
       if ($linha_borrow["borrow_status"] == 1)
              echo "A entregar";
       elseif ($linha_borrow["borrow_status"] == 2)
              echo "Em uso";
       else
              echo "Devolvido";
?></B></FONT>
</TD>
</TR>
```

Figura III.86 – Codificação do caso de uso "Visualizar empréstimo".

III.3.7. Gerenciamento de Fornecedor

As funções de gerenciamento de fornecedor no LegWeb, em termos de codificação PHP [4], podem ser agrupadas nos seguintes grupos: **Selecionar fornecedor**, **Visualizar fornecedor**, **Excluir fornecedor**, **Alterar fornecedor** e **Cadastrar fornecedor**. Todas as funcionalidades e relacionamentos destas funções já foram detalhadas anteriormente no item III.1.2.3 deste trabalho.

III.3.7.1 .Selecionar fornecedor

A codificação deste caso de uso encontra-se no arquivo *leg_vendor.php,* que contém o código que informa ao usuário quais fornecedores já foram cadastrados e apresenta as opções de gerenciamento de fornecedores . A sua codificação comentada encontra-se na figura III.87.

```
// Abre a conexão com o Banco de Dados
  $link_sid = mysql_connect($MYSQL_LOCATION, $MYSQL_ADMIN_USERNAME, $MYSQL_ADMIN_PASSWORD);
  if (!mysql_select_db($MYSQL_DBNAME))
    handleError("leg_vendor.php", mysql_error());

  // Guarda os fornecedores já cadastrados no banco
  $sql = "select vendor_id,company_name from lw_vendor order by company_name";

  $result_vendor = mysql_query($sql) or
    handleError("leg_vendor.php", mysql_error());
?>
<FONT SIZE="+1" COLOR="WHITE"><B>Fornecedores: </B></FONT>
</td>
<?
  // desabilita botões
  $acao_completa = 0;

  // verifica se existe algum fornecedor gravado
  if(mysql_num_rows($result_vendor))
  {
?>
<TD>
<select name="frm_vendor_id">
<?
  // Monta o código HTML com os grupos
  while($linha_vendor = mysql_fetch_array($result_vendor,MYSQL_ASSOC))
    echo "<option
value=\"".$linha_vendor["vendor_id"]."\">".$linha_vendor["company_name"]."\n";
?>
</select>

</TD>
<?
       // habilita botões
    $acao_completa = 1;
       }
       else
       {
?>
<TD><FONT SIZE="+1" COLOR="WHITE"><B>Não há fornecedor registrado.</B></FONT></TD>
<?
       }
?>
</TR>
</TABLE></TD></TR>
<TR><TD>
<table align="center" border="0">
<tr><td>
//Apresenta opções de gerenciamento de fornecedor
<INPUT TYPE="SUBMIT" NAME="btn_acao" VALUE="Criar Fornecedor" style="width: 5cm;">
<INPUT TYPE="SUBMIT" NAME="btn_acao" VALUE="Alterar Fornecedor" style="width: 5cm;" <?
if(!$acao_completa) echo "disabled"; ?>>
</CENTER></TD></TR>
<TR><TD><CENTER>
<INPUT TYPE="SUBMIT" NAME="btn_acao" VALUE="Visualizar Fornecedor" style="width: 5cm;" <?
if(!$acao_completa) echo "disabled"; ?>>
<INPUT TYPE="SUBMIT" NAME="btn_acao" VALUE="Apagar Fornecedor" style="width: 5cm;" <?
if(!$acao_completa) echo "disabled"; ?>>
</td></tr></table></TD></TR>
</form>
</TABLE>
```

Figura III.87 – Codificação do caso de uso "Selecionar fornecedor".

III.3.7.2 .Visualizar fornecedor

Este caso de uso é definido pelo código contido no arquivo *leg_view_vendor.php*. A figura III.88 apresenta o código comentado presente nesse arquivo.

```
// Abre a conexão com o Banco de Dados
  //
  $link_sid = mysql_connect($MYSQL_LOCATION, $MYSQL_ADMIN_USERNAME, $MYSQL_ADMIN_PASSWORD);
  if (!mysql_select_db($MYSQL_DBNAME))
    handleError("leg_vendor.php", mysql_error());

  // Verifica no banco se o fornecedor existe
  $strSQL_vendor = "SELECT * FROM lw_vendor WHERE vendor_id = ".$frm_vendor_id;
  $result_vendor = mysql_query($strSQL_vendor) or
    handleError("leg_vendor.php", mysql_error());

  $linha_vendor = mysql_fetch_array($result_vendor);

<CENTER>
<IMG SRC="images/line_green.gif" WIDTH="300" HEIGHT="2" BORDER="0"><BR>
<FONT COLOR="#004000" SIZE="+2"><B>Consulta de Cadastro</B></FONT>
<FONT SIZE="-1"><CENTER></CENTER></FONT>
<IMG SRC="images/line_green.gif" WIDTH="300" HEIGHT="2" BORDER="0"><BR>

<BR><BR>

<TABLE BORDER="1" ALIGN="CENTER" BGCOLOR="#004000" CELLPADDING="7" CELLSPACING="3">

<TR><TD>
<TABLE BORDER="0" ALIGN="CENTER" BGCOLOR="#004000">

//Informa a razão social do fornecedor cadastrado
<TR><TD ALIGN="RIGHT">
<FONT SIZE="+1" COLOR="WHITE"><B>Razão Social: </B></FONT>
</td>
<TD>
<FONT SIZE="+1" COLOR="WHITE"><B><? echo $linha_vendor["company_name"]; ?></B></FONT>
</TD>

//Informa o CNPJ do fornecedor cadastrado
<TD ALIGN="RIGHT">
<FONT SIZE="+1" COLOR="WHITE"><B>CNPJ: </B></FONT>
</td>
<TD>
<FONT SIZE="+1" COLOR="WHITE"><B><? echo $linha_vendor["company_cnpj"]; ?></B></FONT>
</TD></TR>
```

Figura III.88 – Codificação do caso de uso "Visualizar fornecedor".

III.3.7.3 .Excluir fornecedor

Após a solicitação de exclusão, o sistema solicita uma confirmação de exclusão, codificada no arquivo *leg_del_vendor_verif.php* e, em seguida, executa o código contido no arquivo *leg_del_vendor_commit.php*. A figura III.89 apresenta a parte do código, com comentários, desse arquivo responsável pela exclusão de um fornecedor da base de dados do SID [1].

```
// Abre a conexão com o Banco de Dados
  //
  $link_sid = mysql_connect($MYSQL_LOCATION, $MYSQL_ADMIN_USERNAME, $MYSQL_ADMIN_PASSWORD);
  if (!mysql_select_db($MYSQL_DBNAME))
    handleError("leg_vendor.php", mysql_error());

  // Exclui da tabela vendedor
  $strSQL = "DELETE FROM lw_vendor WHERE vendor_id = $frm_vendor_id";
  mysql_query($strSQL, $link_sid) or
     handleError("leg_vendor.php", mysql_error());

  // Exclui da tabela grupo de reparo
  $strSQL = "DELETE FROM lw_repair_team WHERE vendor_id = $frm_vendor_id";
  mysql_query($strSQL, $link_sid) or
     handleError("leg_vendor.php", mysql_error());
?>
```

Figura III.89 – Codificação do caso de uso "Excluir fornecedor".

III.3.7.4 .Alterar fornecedor

O usuário altera as informações do fornecedor através de dois arquivos. No primeiro, o *leg_alt_vendor_form.php*, são apresentadas as informações pré-cadastradas do fornecedor, permitindo sua edição. No segundo, *leg_alt_vendor_commit.php*, as informações de alteração serão gravadas no banco de dados. Na figura III.90 é apresentado o código de gravação das informações alteradas no banco de dados.

```
// Abre a conexão com o Banco de Dados
  $link_sid = mysql_connect($MYSQL_LOCATION, $MYSQL_ADMIN_USERNAME, $MYSQL_ADMIN_PASSWORD);
  if (!mysql_select_db($MYSQL_DBNAME))
    handleError("leg_vendor.php", mysql_error());

  // Atualiza o fornecedor no banco
        $sql = "Update lw_vendor set company_name =
'$frm_company_name',company_address_street = '$frm_street',";
        $sql .= "company_address_number = '$frm_number',company_address_compl =
'$frm_compl',company_address_town = '$frm_town',";
        $sql .= "company_address_city = '$frm_city',company_address_state =
'$frm_state',company_address_zip = '$frm_zip',";
        $sql .= "company_phone = '$frm_phone',company_phone_area_code =
'$frm_ddd',company_phone_country_code = '$frm_ddi',";
        $sql .= "reselling_phone = '$frm_phone2',reselling_phone_area_code =
'$frm_ddd2',reselling_phone_country_code = '$frm_ddi2',";
        $sql .= "reselling_phone_cellular = '$frm_cel',reselling_phone_cellular_area_code =
'$frm_ddd_cel',";
        $sql .= "reselling_phone_cellular_country_code = '$frm_ddi_cel',company_cnpj =
'$frm_cnpj',reselling_name = '$frm_name',";
        $sql .= "comments = '$frm_comments' where vendor_id = $frm_vendor_id";

  $result_lw_vendor = mysql_query($sql) or
    handleError("leg_vendor.php", mysql_error());

  if(!mysql_affected_rows())
        handleError("leg_vendor.php", "Erro desconhecido ao alterar vendedor.");
```

```
  // Atualiza o time de reparo no banco
        $sql = "Update lw_repair_team set company_name =
'$frm_company_name',company_address_street = '$frm_street',";
        $sql .= "company_address_number = '$frm_number',company_address_compl =
'$frm_compl',company_address_town = '$frm_town',";
        $sql .= "company_address_city = '$frm_city',company_address_state =
'$frm_state',company_address_zip = '$frm_zip',";
        $sql .= "company_phone = '$frm_phone',company_phone_area_code =
'$frm_ddd',company_phone_country_code = '$frm_ddi',";
        $sql .= "responsable_phone = '$frm_phone2',responsable_phone_area_code =
'$frm_ddd2',responsable_phone_country_code = '$frm_ddi2',";
        $sql .= "responsable_phone_cellular =
'$frm_cel',responsable_phone_cellular_area_code = '$frm_ddd_cel',";
        $sql .= "responsable_phone_cellular_country_code = '$frm_ddi_cel',company_cnpj =
'$frm_cnpj',responsable_name = '$frm_name'";
        $sql .= " where vendor_id = $frm_vendor_id";

$result_lw_vendor = mysql_query($sql) or
      handleError("leg_vendor.php", mysql_error());

    if(!mysql_affected_rows())
        handleError("leg_vendor.php", "Erro desconhecido ao alterar time de reparo.");
?>
```

Figura III.90 – Codificação do caso de uso "Alterar fornecedor".

III.3.7.5 .Cadastrar fornecedor

O usuário cadastra um fornecedor através de dois arquivos. No primeiro, o *leg_add_new_vendor.php*, é apresentado um formulário de cadastro de fornecedor para preenchimento pelo usuário. No segundo, *leg_add_new_vendor_commit.php*, as informações de inclusão serão gravadas no banco de dados. Na figura III.91 é apresentado o código de gravação das informações de inclusão no banco de dados.

```
// Abre a conexão com o Banco de Dados
  //
  $link_sid = mysql_connect($MYSQL_LOCATION, $MYSQL_ADMIN_USERNAME,
$MYSQL_ADMIN_PASSWORD);
  if (!mysql_select_db($MYSQL_DBNAME))
    handleError("leg_vendor.php", mysql_error());

  // Verifica no banco se o fornecedor existe
  $strSQL_vendor = "SELECT * FROM lw_vendor WHERE company_cnpj = ".$frm_cnpj;
  $result_vendor = mysql_query($strSQL_vendor) or
    handleError("leg_vendor.php", mysql_error());

    if (mysql_num_rows($result_vendor))
  {
        $str_erro = "O fornecedor já existe.";

        // envia o erro
        handleError("leg_vendor.php", $str_erro);
        exit();
  }
```

```
  // Adiciona o fornecedor ao banco
        $sql = "INSERT INTO lw_vendor (company_name,company_address_street,";
        $sql .= "company_address_number,company_address_compl,company_address_town,";
        $sql .= "company_address_city,company_address_state,company_address_zip,";
        $sql .= "company_phone,company_phone_area_code,company_phone_country_code,";
        $sql .=
"reselling_phone,reselling_phone_area_code,reselling_phone_country_code,";
        $sql .= "reselling_phone_cellular,reselling_phone_cellular_area_code,";
        $sql .=
"reselling_phone_cellular_country_code,company_cnpj,reselling_name,comments)";
        $sql .= " values ('$frm_company_name','$frm_street','$frm_number','$frm_compl',";
        $sql .= "'$frm_town','$frm_city','$frm_state','$frm_zip','$frm_phone',";
        $sql .= "'$frm_ddd','$frm_ddi','$frm_phone2','$frm_ddd2','$frm_ddi2',";
        $sql .= "'$frm_cel','$frm_ddd_cel','$frm_ddi_cel',";
        $sql .= "'$frm_cnpj','$frm_name','$frm_comments')";

  $result_lw_vendor = mysql_query($sql) or
    handleError("leg_vendor.php", mysql_error());

  if(!mysql_affected_rows())
        handleError("leg_vendor.php", "Erro desconhecido ao inserir vendedor.");


  $vendor_id = mysql_insert_id();

    // Adiciona time de reparo ao banco (cópia do fornecedor)
        $sql = "INSERT INTO lw_repair_team (company_name,company_address_street,";
        $sql .= "company_address_number,company_address_compl,company_address_town,";
        $sql .= "company_address_city,company_address_state,company_address_zip,";
        $sql .= "company_phone,company_phone_area_code,company_phone_country_code,";
        $sql .=
"responsable_phone,responsable_phone_area_code,responsable_phone_country_code,";
        $sql .= "responsable_phone_cellular,responsable_phone_cellular_area_code,";
        $sql .=
"responsable_phone_cellular_country_code,company_cnpj,responsable_name,comments,";
        $sql .= "vendor_id)";
        $sql .= " values ('$frm_company_name','$frm_street','$frm_number','$frm_compl',";
        $sql .= "'$frm_town','$frm_city','$frm_state','$frm_zip','$frm_phone',";
        $sql .= "'$frm_ddd','$frm_ddi','$frm_phone2','$frm_ddd2','$frm_ddi2',";
        $sql .= "'$frm_cel','$frm_ddd_cel','$frm_ddi_cel',";
        $sql .= "'$frm_cnpj','$frm_name','$frm_comments',$vendor_id)";

    $result_lw_vendor = mysql_query($sql) or
      handleError("leg_vendor.php", mysql_error());

    if(!mysql_affected_rows())
        handleError("leg_vendor.php", "Erro desconhecido ao inserir time de reparo.");
?>
```

Figura III.91 – Codificação do caso de uso "Cadastrar fornecedor".

III.3.8. Gerenciamento de Insumos

As funções de gerenciamento de insumos no LegWeb referem-se aos casos de uso de Gerenciamento de componentes e Gerenciamento de Equipamentos. A codificação para esses casos é bastante semelhante e serão apresentadas abaixo usando como padrão o gerenciamento de componentes.

Em termos de codificação PHP [4], podem ser agrupadas nos seguintes grupos: **Selecionar Categoria** (administrador)**, Selecionar fornecedor** (administrador), **Selecionar**

insumo (administrador)**, Criar insumo** (administrador), **Consultar insumo** (administrador), **Excluir insumo** (administrador), **Alterar insumo** (administrador), **Abrir chamado de manutenção**(administrador, professor e aluno), **Fechar chamado de manutenção** (administrador), **Selecionar responsável pela manutenção** (administrador, aluno e professor), **Selecionar chamado de manutenção**(administrador, professor e aluno), **Visualizar chamado de manutenção** (administrador, professor e aluno), **Alterar chamado de manutenção** (administrador) e **Solicitar Compra de insumo** (professor e aluno). Todas as funcionalidades e relacionamentos destas funções já foram detalhadas anteriormente nos itens III.1.2.3, III.1.2.4, III.1.3.1, III.1.3.2, III.1.4.1 e III.1.4.2 deste trabalho.

III.3.8.1 .Selecionar categoria

A codificação deste caso de uso encontra-se no arquivo *leg_comp.php,* que contém permite ao usuário selecionar a categoria desejada, a categoria desejada e as opções de manutenção para o gerenciamento de componentes. A sua codificação comentada encontra-se na figura III.92.

```
// Abre a conexão com o Banco de Dados SID
  //
  $link_sid = mysql_connect($MYSQL_LOCATION, $MYSQL_ADMIN_USERNAME,
$MYSQL_ADMIN_PASSWORD);
  if (!mysql_select_db($MYSQL_DBNAME))
    handleError("leg_comp.php", mysql_error());

  if(!IsSet($frm_rank_id))
       $frm_rank_id = 0;

  // Grava as categorias a partir da categoria pai
  $sql = "select rank_id,rank_name,father_code from lw_rank where father_code =
$frm_rank_id and rank_type = 0";
  $sql .= " order by rank_name";

  $result_rank = mysql_query($sql) or
    handleError("leg_comp.php", mysql_error());

  // Grava os componentes
  $sql = "select component_id,component_description from lw_component";
  $sql .= " where component_rank_id = $frm_rank_id order by component_description";

  $result_comp = mysql_query($sql) or
    handleError("leg_comp.php", mysql_error());
?>
```

```
<?
        $acao_completa4 = 0;
        // Se o id for zero então pertence a raiz da árvore
        if($frm_rank_id)
        {
                // Guarda a categoria
                $sql = "select rank_name from lw_rank where rank_id = $frm_rank_id and
rank_type = 0";

                $result_cat = mysql_query($sql) or
                handleError("leg_comp.php", mysql_error());

                $linha_cat = mysql_fetch_array($result_cat,MYSQL_ASSOC);

                // Retorna o nome da categoria
                echo $linha_cat["rank_name"];

                // Habilita botões
                $acao_completa = 1;
                $acao_completa4 = 1;
        }
        else
        {
                // desabilita botões
                $acao_completa = 0;
                echo "Raiz";
        }
?>
<FORM ACTION="leg_comp.php" METHOD="POST">
<FONT SIZE="+1" COLOR="WHITE"><B>Categorias Derivadas: </B></FONT>
<select name="frm_rank_id">
<?
  // Retorna o total de categorias derivadas
  $total = mysql_num_rows($result_rank);
  $acao_completa3 = 0;
  if($total)
  {
    $acao_completa3 = 1;
        // Monta o codigo HTML com as categorias derivadas (Caso de uso selecionar
//categoria)
        for($i=0;$i<$total;$i++)
        {
        $linha_rank = mysql_fetch_array($result_rank,MYSQL_ASSOC);
        echo "<option value=\"".$linha_rank["rank_id"]."\">".$linha_rank["rank_name"]."\n";
                $array_rank[$i] = $linha_rank["rank_id"];
        }
        // Se o pai não for a raiz então permite volta
        if($linha_rank["father_code"])
        {
                // Guarda o avô
        $sql = "Select father_code from lw_rank where rank_id =
".$linha_rank["father_code"]." and rank_type = 0";

                $result_avo = mysql_query($sql) or
        handleError("leg_comp.php", mysql_error());

                $linha_avo = mysql_fetch_array($result_avo,MYSQL_ASSOC);
?>
  <option value="<? echo $linha_avo["father_code"]; ?>">Voltar
<?
        }
  }
  else
  {
        $acao_completa3 = $acao_completa4;
        // Guarda o avô
    $sql = "Select father_code from lw_rank where rank_id = $frm_rank_id and rank_type =
0";

        $result_avo = mysql_query($sql) or
      handleError("leg_comp.php", mysql_error());

        $linha_avo = mysql_fetch_array($result_avo,MYSQL_ASSOC);
?>
  <option value="<? echo $linha_avo["father_code"]; ?>">Não há ramificações - Voltar
<?
        }
?>
```

```
</select>
</TD>
</TR>
<tr>
<td colspan="2" align="center"><input type="Submit" name="categoria" value="Selecionar
Categoria" <? if(!$acao_completa3) echo "disabled"; ?>></td>
</tr>
</form>
</table>
</td></tr>
<tr><td>
<TABLE BORDER="0" ALIGN="CENTER" BGCOLOR="#004000">
<TR><TD ALIGN="RIGHT">
<FORM ACTION="leg_action_comp.php" METHOD="POST">
<tr><td align="right">
// Monta o codigo HTML com os componentes disponíveis (Caso de uso Selecionar insumo)
<FONT SIZE="+1" COLOR="WHITE"><B>Componentes:</B></FONT>
</td>
<td>
<?
        $sql = "select component_id,component_description from lw_component where";
        $sql .= " component_rank_id = $frm_rank_id";

        $result_comp = mysql_query($sql) or
      handleError("leg_comp.php", mysql_error());

        $acao2_completa = 0;

        if(mysql_num_rows($result_comp))
        {
                echo "<select name=\"frm_comp_id\">";
                while($linha_comp = mysql_fetch_array($result_comp,MYSQL_ASSOC))
                        echo "<option
value=\"".$linha_comp["component_id"]."\">".$linha_comp["component_description"];
                echo "</select>";
                $acao2_completa = 1;
        }
        else
        {
                echo "<FONT SIZE=\"+1\" COLOR=\"WHITE\"><B>Nenhum componente
localizado.</b></font>";
        }
?>
//Apresenta cardápio de opções de manutenção de componente
</TD>
</TR>
</TABLE></TD></TR>
<TR><TD>
<table align="center" border="0" cellpadding="0" cellspacing="5">
<tr>
<td align="center">
<INPUT TYPE="SUBMIT" NAME="btn_acao" VALUE="Criar Componente" <? if(!$acao_completa)
echo "disabled"; ?> style="width: 5cm;"><br>
</td>
<td align="center">
<INPUT TYPE="SUBMIT" NAME="btn_acao" VALUE="Consultar Componente" <?
if(!$acao2_completa) echo "disabled"; ?> style="width: 5cm;">
</td>
</tr>
<tr>
<td align="center">
<INPUT TYPE="SUBMIT" NAME="btn_acao" VALUE="Alterar Componente" <? if(!$acao2_completa)
echo "disabled"; ?> style="width: 5cm;"><br>
</td>
<td align="center">
<INPUT TYPE="SUBMIT" NAME="btn_acao" VALUE="Apagar Componente" <? if(!$acao2_completa)
echo "disabled"; ?> style="width: 5cm;">
</td>
</tr>
</table>
</TD></TR>
<input type="Hidden" name="frm_rank_id" value="<? echo $frm_rank_id; ?>">
</form>
</TABLE>
```

Figura III.92 – Codificação do caso de uso "Selecionar categoria".

III.3.8.2 .Selecionar insumo

Este caso de uso é definido pelo código contido no arquivo *leg_comp.php*, da mesma forma que o item III.3.8.1, e a figura III.92 apresenta o código comentado presente nesse arquivo, incluindo esse caso de uso.

III.3.8.3 .Criar insumo

O usuário cadastra um insumo através de dois arquivos. No primeiro, o *leg_add_new_comp.php*, é apresentado um formulário de cadastro de insumo para preenchimento pelo usuário. Este código, comentado, é apresentado na figura III.93.

```
<?
        // Verifica se um vendedor já foi selecionado previamente (caso de uso Selecionar
        //Fornecedor)
        if(IsSet($str_vend_id))
        {
                // Dá a opção ao usuário de selecionar um fornecedor ou de descelecioná-lo
                switch($bt_acao)
                {
                        case "Adicionar":
                // Verifica se já tem algum fornecedor selecionado
                                if(strlen($str_vend_id))
                                {
                                        // Se houver, salva o novo no final separando por
                                        //ponto-e-vírgula
                                        $str_vend_id = $str_vend_id.";".$frm_vendor_id;
                                }
                                else
                                {
                                        // Salva do primeiro elemento
                                        $str_vend_id = $frm_vendor_id;
                                }
                                break;
                        case "Remover":
                                $array_temp = explode(";",$str_vend_id); // Monta um array

                                $str_vend_id = ""; // Zera o string

                                // Remonta o string removendo o item desejado
                                for($i=0;$i<count($array_temp);$i++)
                                {
                                        if($array_temp[$i] != $frm_vendor_id2)
                                        {
                                                if(strlen($str_vend_id))
                                                        $str_vend_id =
$str_vend_id.";".$array_temp[$i];
                                                else
                                                        $str_vend_id = $array_temp[$i];
                                        }
                                }
                                break;
                        default:
                                handleError("leg_comp.php", "Ação desconhecida.");
                }

                // Verifica se tem algo no string
                if(strlen($str_vend_id))
                        $array_vend_id = explode(";",$str_vend_id);
                else
                        $array_vend_id = array();
        }
        else
        {
                $str_vend_id = "";
                $array_vend_id = array();
        }
```

```
// Apresenta os fornecedores selecionados
        if(count($array_vend_id))
        {
                $sql = "Select vendor_id,company_name from lw_vendor where vendor_id in (";
                $sql .= implode(",",$array_vend_id).") order by company_name";

                $result_vendor = mysql_query($sql) or
        handleError("leg_comp.php", mysql_error());

                echo "<select name=\"frm_vendor_id2\">\n";
                while($linha_vendor = mysql_fetch_array($result_vendor,MYSQL_ASSOC))
                        echo "<option
value=\"".$linha_vendor["vendor_id"]."\">".$linha_vendor["company_name"]."\n";
                echo "</select>\n";

                echo "<input type=\"Submit\" name=\"bt_acao\" value=\"Remover\">\n";
        }
        else
        {
                echo "Nenhum";
        }
?>
<TR><TD ALIGN="RIGHT"><FONT SIZE="+1" COLOR="WHITE"><B>Disponíveis: </B></FONT></td>

<?
        // Guarda os fornecedores restantes (os não selecionados)
        if(count($array_vend_id))
        {
                $sql = "Select vendor_id,company_name from lw_vendor where vendor_id not in
(";
                $sql .= implode(",",$array_vend_id).") order by company_name";
        }
        else
        {
                $sql = "Select vendor_id,company_name from lw_vendor order by
company_name";
        }

        $result_vendor = mysql_query($sql) or
        handleError("leg_comp.php", mysql_error());

        // Monta o codigo HTML com os fornecedores
        if(mysql_num_rows($result_vendor))
        {
                echo "<select name=\"frm_vendor_id\">\n";
                while($linha_vendor = mysql_fetch_array($result_vendor,MYSQL_ASSOC))
                        echo "<option
value=\"".$linha_vendor["vendor_id"]."\">".$linha_vendor["company_name"]."\n";
                echo "</select>\n";

                echo "<input type=\"Submit\" name=\"bt_acao\" value=\"Adicionar\">\n";
        }
        else
        {
                echo "<FONT SIZE=\"+1\" COLOR=\"WHITE\"><B>Nenhum</b></font>\n";
        }
?>
        <input type="Hidden" name="frm_rank_id" value="<? echo $frm_rank_id; ?>">
        <input type="Hidden" name="str_vend_id" value="<? echo $str_vend_id; ?>">
```

Figura III.93 – Codificação do caso de uso "Criar componente".

No segundo, *leg_add_new_comp_commit.php*, as informações de inclusão serão gravadas no banco de dados. Na figura III.94 é apresentado o código de gravação das informações de criação no banco de dados.

```
// Abre a conexão com o Banco de Dados
  $link_sid = mysql_connect($MYSQL_LOCATION, $MYSQL_ADMIN_USERNAME,
$MYSQL_ADMIN_PASSWORD);
  if (!mysql_select_db($MYSQL_DBNAME))
    handleError("leg_equip.php", mysql_error());
        // Verifica se tem um componente com o mesmo nome

  $sql = "Select component_id from lw_component where component_description =
'$frm_comp_desc'";

  $result_comp = mysql_query($sql) or
    handleError("leg_comp.php", mysql_error());

  if(mysql_num_rows($result_comp))
    handleError("leg_comp.php", "Já existe um componente com esse nome.");

  // Cria o componente

  $sql = "Insert into lw_component (component_description,component_tolerance,";
  $sql .= "component_dimensions,component_acquired,component_date,";
  $sql .= "component_nominal_value,component_rank_id,component_active) values
('$frm_comp_desc','$frm_comp_tol',";
  $sql .= "'$frm_comp_dim',$frm_comp_qtd,'$frm_comp_dt_ano-$frm_comp_dt_mes-
$frm_comp_dt_dia',";
  $sql .= "'$frm_comp_nom',$frm_rank_id,$frm_comp_qtd)";

  $result_comp = mysql_query($sql) or
    handleError("leg_comp.php", mysql_error());

  $comp_id = mysql_insert_id();

  $array_vend_id = explode(";",$str_vend_id);

  for($i=0;$i<count($array_vend_id);$i++)
  {
        // Cria a referência cruzada entre vendedor e componente

        $sql = "Insert into lw_xref_vendor_component (vendor_id,component_id) values";
        $sql .= " (".$array_vend_id[$i].",$comp_id)";

        $result_vend_comp = mysql_query($sql) or
        handleError("leg_comp.php", mysql_error());
  }
?>
```

Figura III.94 – Codificação de armazenamento do novo componente no banco de dados.

III.3.8.4 .Selecionar fornecedor

Este caso de uso é definido pelo código contido no arquivo *leg_add_new_comp.php*, da mesma forma que o item III.3.8.3, e a figura III.94 apresenta o código comentado presente nesse arquivo, incluindo esse caso de uso.

III.3.8.5 .Alterar insumo

O usuário altera as informações do insumo através de dois arquivos. No primeiro, o *leg_alt_comp_form.php*, são apresentadas as informações pré-cadastradas do componente, permitindo sua edição. No segundo, *leg_alt_comp_commit.php*, as informações de alteração

serão gravadas no banco de dados. Na figura III.95 é apresentado o código de gravação das informações alteradas no banco de dados.

```
// Abre a conexão com o Banco de Dados SID
  $link_sid = mysql_connect($MYSQL_LOCATION, $MYSQL_ADMIN_USERNAME, $MYSQL_ADMIN_PASSWORD);
  if (!mysql_select_db($MYSQL_DBNAME))
    handleError("leg_cat.php", mysql_error());

  // Guarda as informações cadastradas atualmente no banco
  $sql = "select * from lw_component where component_id = $frm_comp_id";

  $result_comp = mysql_query($sql) or
    handleError("leg_comp.php", mysql_error());

  $linha_comp = mysql_fetch_array($result_comp,MYSQL_ASSOC);

  // Atualiza componente
  $sql = "Update lw_component set";
  $sql .= " component_out=$frm_qtd_queimada,component_destroyed=$frm_qtd_mal,";
  $sql .= "component_missing=$frm_qtd_extraviada,";

  // se a data for diferente é uma compra
  // se a data for igual é apenas uma alteração
  // em ambos os casos todos os campos podem ser mudados
  if (date("Y-m-d",strtotime($linha_comp["component_date"])) != "$frm_dt_cp_ano-
$frm_dt_cp_mes-$frm_dt_cp_dia")
  {
        $sql .= "component_active = component_active + $frm_qtd_compra +
(".$linha_comp["component_out"]." - $frm_qtd_queimada";
        $sql .= ") + (".$linha_comp["component_destroyed"]." - $frm_qtd_mal) +
(".$linha_comp["component_missing"];
        $sql .= " - $frm_qtd_extraviada),component_acquired=$frm_qtd_compra,";
        $sql .= "component_date='$frm_dt_cp_ano-$frm_dt_cp_mes-$frm_dt_cp_dia'";
  }
  else
  {
        $sql .= "component_active = component_active + ($frm_qtd_compra -
".$linha_comp["component_acquired"].") + (".$linha_comp["component_out"]." -
$frm_qtd_queimada";
        $sql .= ") + (".$linha_comp["component_destroyed"]." - $frm_qtd_mal) +
(".$linha_comp["component_missing"];
        $sql .= " - $frm_qtd_extraviada),component_acquired=$frm_qtd_compra";
  }

  // Informa a que categoria pertence
  $sql .= ",component_rank_id=$frm_rank_id where component_id = $frm_comp_id";

  mysql_query($sql) or
    handleError("leg_comp.php", mysql_error());

  // Apaga relação vendedor x componente
  $sql = "delete from lw_xref_vendor_component where component_id = $frm_comp_id";

  mysql_query($sql) or
    handleError("leg_comp.php", mysql_error());

  // Refaz relação de vendedor x componente
  $array_vend_id = explode(";",$str_vend_id);

  for($i=0;$i<count($array_vend_id);$i++)
  {
        // cria a relação vendedor x componente

        $sql = "Insert into lw_xref_vendor_component (vendor_id,component_id) values";
        $sql .= " (".$array_vend_id[$i].",$frm_comp_id)";

        mysql_query($sql) or
       handleError("leg_comp.php", mysql_error());
  }
?>
```

Figura III.95 – Codificação do caso de uso "Alterar componente".

III.3.8.6 .Excluir insumo

Após a solicitação de exclusão, o sistema solicita uma confirmação de exclusão, codificada no arquivo *leg_del_comp_verif.php* e, em seguida, executa o código contido no arquivo *leg_del_comp_commit.php*. A figura III.96 apresenta a parte do código, com comentários, desse arquivo responsável pela exclusão de um insumo da base de dados do SID[1].

```
// Abre a conexão com o Banco de Dados SID
  //
  $link_sid = mysql_connect($MYSQL_LOCATION, $MYSQL_ADMIN_USERNAME, $MYSQL_ADMIN_PASSWORD);
  if (!mysql_select_db($MYSQL_DBNAME))
    handleError("leg_comp.php", mysql_error());

  // Remove relação vendedor x componente
  $sql = "delete from lw_xref_vendor_component where component_id = $frm_comp_id";
  mysql_query($sql, $link_sid) or
    handleError("leg_comp.php", mysql_error());

  // Remove componente
  $sql = "delete from lw_component where component_id = $frm_comp_id";
  mysql_query($sql, $link_sid) or
    handleError("leg_comp.php", mysql_error());

?>
```

Figura III.96 – Codificação do caso de uso "Excluir componente".

III.3.8.7 .Consultar insumo

Este caso de uso é definido pelo código contido no arquivo *leg_cons_comp.php*. A figura III.97 apresenta o código comentado presente nesse arquivo.

```
// Abre a conexão com o Banco de Dados SID
  $link_sid = mysql_connect($MYSQL_LOCATION, $MYSQL_ADMIN_USERNAME,
$MYSQL_ADMIN_PASSWORD);
  if (!mysql_select_db($MYSQL_DBNAME))
    handleError("leg_comp.php", mysql_error());

  // Busca o nome do componente
  $strSQL = "select * FROM lw_component WHERE component_id = $frm_comp_id";
  $result_comp = mysql_query($strSQL, $link_sid) or
    handleError("leg_comp.php", mysql_error());

  $linha_comp = mysql_fetch_array($result_comp,MYSQL_ASSOC);
?>
// Retorna ao usuário as informações do componente. Como ilustrtação é reproduzido o
//código para informar o "Valor Nominal" de um componente.
<TR><TD align="right">
<FONT SIZE="+1" COLOR="WHITE"><b>Valor nominal:</b></font>
</TD><TD>
<FONT SIZE="+1" COLOR="WHITE"><b><? echo $linha_comp["component_nominal_value"];
?></b></font>
</TD></TR>
```

Figura III.97 – Codificação do caso de uso "Consultar componente".

III.3.8.8 .Abrir chamado de manutenção

O usuário abre um chamado através de três arquivos. No primeiro, o *leg_action_manu_equip.php*, são apresentadas as opções de cardápio para gerenciamento de manutenção. Após selecionar a opção de abertura de chamado, é acionado o código presente no arquivo *leg_add_repair_request.php*, que é responsável por apresentar um formulário para abertura do chamado. O terceiro, o *leg_add_repair_request_commit.php* é resposnável pelo cadastro do chamado. Na figura III.98 é apresentado o código de abertura de chamado de manutenção.

```
// Abre a conexão com o Banco de Dados
  $link_sid = mysql_connect($MYSQL_LOCATION, $MYSQL_ADMIN_USERNAME,
$MYSQL_ADMIN_PASSWORD);
  if (!mysql_select_db($MYSQL_DBNAME))
    handleError("leg_select_group.php", mysql_error());
  $msg="";
  //Verifico se o número de inventário consta no banco
  $querySelectEquipment = "SELECT equip_id, equip_repair_time FROM lw_equipment WHERE
equip_control_number='".$_REQUEST["no_inventario"]."'";
  $resultEquipment = mysql_query($querySelectEquipment, $link_sid) or
    handleError("leg_add_repair_request.php", mysql_error());
  $quant = mysql_num_rows($resultEquipment);
  if($quant==0){
        $msg.="<BR>Não existe equipamento com o número de inventário fornecido";
  }
  elseif($quant==1){
        $equipment = mysql_fetch_array($resultEquipment);
        //Verifico se já existe chamado para o número de inventário fornecido
        $querySelect = "SELECT * FROM lw_xref_repair_team_equipment WHERE
equip_id=".$equipment["equip_id"];
        $result = mysql_query($querySelect, $link_sid) or
        handleError("leg_add_repair_request.php", mysql_error());
        if(mysql_num_rows($result)==1){
                $msg .= "<BR>O equipamento já possui chamado em aberto";
        }
        else{
                //Se não possuir cadastro um para ele
                $queryInsert = "INSERT INTO lw_xref_repair_team_equipment ( equip_id ,
repair_team_id) VALUES ( {$equipment['equip_id']}, {$_REQUEST['responsavel']} )";
                $resultInsert = mysql_query($queryInsert, $link_sid) or
                handleError("leg_add_repair_request.php", mysql_error());
                if(mysql_affected_rows()==1){
                        //atualizo tabela lw_equipment com os dados da nova manutenção
                        $data = explode("/",$_REQUEST["dt_abertura"]);
                        $queryUpdate = "UPDATE lw_equipment set equip_last_repair =
'$data[2]-$data[1]-$data[0]'";
                        $queryUpdate .= " , equip_repair_time =
".($equipment["equip_repair_time"]+1);
                        $queryUpdate .= " , equip_repair_cost = '".$_REQUEST["valor"]."'";
                        $queryUpdate .= " , equip_repair_status = 1";
                        $queryUpdate .= " , equip_repair_description =
'".nl2br(htmlspecialchars($_REQUEST["descricao"]))."'";
                        $data = explode("/",$_REQUEST["previsao"]);
                        $queryUpdate .= " , equip_repair_end = '$data[2]-$data[1]-
$data[0]'";
                        $queryUpdate .= " WHERE equip_id=".$equipment["equip_id"];

                        $resultUpdate = mysql_query($queryUpdate, $link_sid) or
                        handleError("leg_add_repair_request.php", mysql_error());

                        if(mysql_affected_rows()==1){
                                $msg = "Operação realizada com sucesso";
                }
        }
  ?>
```

Figura III.98 – Codificação do caso de uso "Abrir chamado de manutenção".

III.3.8.9 .Selecionar responsável pela manutenção

Este caso de uso é definido pelo código contido no arquivo *leg_add_repair_request.php*. A figura III.99 apresenta o código comentado presente nesse arquivo.

```
// Abre a conexão com o Banco de Dados
  $link_sid = mysql_connect($MYSQL_LOCATION, $MYSQL_ADMIN_USERNAME,
$MYSQL_ADMIN_PASSWORD);
  if (!mysql_select_db($MYSQL_DBNAME))
    handleError("leg_select_group.php", mysql_error());
// Guarda informações sobre as equipes de reparo
  $querySelect = "SELECT repair_team_id, company_name, responsable_name FROM
lw_repair_team ORDER BY responsable_name";

  $resultResp = mysql_query($querySelect, $link_sid) or
    handleError("leg_select_group.php", mysql_error());

?>
// Informa em tela as equipes para seleção pelo usuário
<td>Respons&aacute;vel pela manuten&ccedil;&atilde;o:</td>
    <td><select name="responsavel" id="responsavel">
        <?
        while($resp = mysql_fetch_array($resultResp) ){
               echo "\n<option
value='".$resp["repair_team_id"]."'>".$resp["responsable_name"]." -
".$resp["company_name"]."</option>";
        }
        ?>
      </select></td>
```

Figura III.99 – Codificação do caso de uso "Selecionar responsável pela manutenção".

III.3.8.10 .Selecionar chamado de manutenção

Este caso de uso é definido pelo código contido no arquivo *leg_manu_equip.php*. A figura III.100 apresenta o código comentado presente nesse arquivo.

```
// Abre a conexão com o Banco de Dados SID
  $link_sid = mysql_connect($MYSQL_LOCATION, $MYSQL_ADMIN_USERNAME,
$MYSQL_ADMIN_PASSWORD);
  if (!mysql_select_db($MYSQL_DBNAME))
    handleError("leg_equip.php", mysql_error());

  // Guarda os equipamentos em manutenção
  $sql = "select a.equip_id,a.equip_name,b.repair_team_id from lw_equipment a,
lw_xref_repair_team_equipment b";
  $sql .= " where a.equip_repair_status = 1 and a.equip_id = b.equip_id order by
a.equip_name";
?>
<FONT SIZE="+1" COLOR="WHITE"><B>Equipamentos em manutenção:</B></FONT>
</td>
<td>
```

```
//Retorna ao usuário a lista de equipamentos em manutenção para seleção.
<?
        $result_equip = mysql_query($sql) or
      handleError("leg_equip.php", mysql_error());

        $acao2_completa = 0;

        if(mysql_num_rows($result_equip))
        {
                echo "<select name=\"frm_equip_id\">";
                while($linha_equip = mysql_fetch_array($result_equip,MYSQL_ASSOC))
                        echo "<option
value=\"".$linha_equip["equip_id"]."\">".$linha_equip["equip_name"];
                echo "</select>";
                $acao2_completa = 1;
        }
        else
        {
                echo "<FONT SIZE=\"+1\" COLOR=\"WHITE\"><B>Nenhum equipamento
localizado.</b></font>";
        }
?>
```

Figura III.100 – Codificação do caso de uso "Selecionar chamado de manutenção".

III.3.8.11 .Visualizar chamado de manutenção

Este caso de uso é definido pelo código contido no arquivo *leg_view_repair_request.php*. A figura III.101 apresenta o código comentado presente nesse arquivo.

```
// Abre a conexão com o Banco de Dados
  $link_sid = mysql_connect($MYSQL_LOCATION, $MYSQL_ADMIN_USERNAME,
$MYSQL_ADMIN_PASSWORD);
  if (!mysql_select_db($MYSQL_DBNAME))
    handleError("leg_select_group.php", mysql_error());

  //Verifica se o número de inventário consta no banco
  $querySelect = "SELECT E.equip_control_number as equip_control_number,
E.equip_last_repair as equip_last_repair,";
  $querySelect .= " E.equip_repair_description as equip_repair_description,
E.equip_repair_end as equip_repair_end";
  $querySelect .= ", E.equip_repair_cost as equip_repair_cost, R.company_name as
company_name, R.responsable_name as respansable_name";
  $querySelect .= " FROM lw_equipment E, lw_xref_repair_team_equipment X, lw_repair_team R
";
  $querySelect .= " WHERE X.equip_id = ".$frm_equip_id;
  $querySelect .= " AND   E.equip_id = X.equip_id ";
  $querySelect .= " AND   X.repair_team_id = R.repair_team_id  ";

        $result = mysql_query($querySelect, $link_sid) or
        handleError("leg_add_repair_request.php", mysql_error());
?>
// Retorna para usuário informações. Para exemplo, segue codificação para os campos
//"Número de Inventário" e "Data de abertura de chamado"
<?
        while($repair = mysql_fetch_array($result))
        {
?>
<table width="70%" border="0" cellspacing="0" cellpadding="0">
  <tr>
    <td width="43%">N&uacute;mero do invent&aacute;rio:</td>
    <td width="57%"><? echo $repair["equip_control_number"] ?></td>
  </tr>
  <tr>
    <td>Data de abertura do chamado:</td>
    <td>      <? echo $repair["equip_last_repair"] ?></td>
  </tr>
```

Figura III.101 – Codificação do caso de uso "Visualilzar chamado de manutenção".

III.3.8.12 .Fechar chamado de manutenção

Após a solicitação de encerramento, o sistema solicita uma confirmação, codificada no arquivo *leg_fechar_manu_equip.php* e, em seguida, executa o código contido no arquivo *leg_fechar_manu_equip_commit.php*. A figura III.102 apresenta a parte do código, com comentários, desse arquivo responsável pelo encerramento de um chamado de manutenção.

```
// Abre a conexão com o Banco de Dados SID
  //
  $link_sid = mysql_connect($MYSQL_LOCATION, $MYSQL_ADMIN_USERNAME, $MYSQL_ADMIN_PASSWORD);
  if (!mysql_select_db($MYSQL_DBNAME))
    handleError("leg_equip.php", mysql_error());

  //
  // Remove relação vendedor x equipamento
  //
  $sql = "delete from lw_xref_repair_team_equipment where equip_id = $frm_equip_id";
  mysql_query($sql, $link_sid) or
    handleError("leg_equip.php", mysql_error());

  //
  // Libera equipamento
  //
  $sql = "Update lw_equipment set equip_repair_status = 0 where equip_id = $frm_equip_id";
  mysql_query($sql, $link_sid) or
    handleError("leg_comp.php", mysql_error());

?>
```

Figura III.102 – Codificação do caso de uso "Fechar chamado de manutenção".

III.3.8.13 .Alterar chamado de manutenção

O usuário altera as informações do chamado através de dois arquivos. No primeiro, o *leg_alt_menu_equip.php*, são apresentadas as informações pré-cadastradas do chamado, permitindo sua edição. No segundo, *leg_alt_menu_equip_verif.php*, as informações de alteração serão gravadas no banco de dados. Na figura III.103 é apresentado o código de gravação das informações alteradas no banco de dados.

```
  // Abre a conexão com o Banco de Dados
  $link_sid = mysql_connect($MYSQL_LOCATION, $MYSQL_ADMIN_USERNAME, $MYSQL_ADMIN_PASSWORD);
  if (!mysql_select_db($MYSQL_DBNAME))
    handleError("leg_manu_equip.php", mysql_error());
  $msg="";
  $sql = "Delete from lw_xref_repair_team_equipment where equip_id = ".$frm_equip_id;
  mysql_query($sql, $link_sid) or
       handleError("leg_manu_equip.php", mysql_error());

              //Se não possuir relacionamento cadastro
              $queryInsert = "INSERT INTO lw_xref_repair_team_equipment ( equip_id ,
repair_team_id) VALUES ( ".$frm_equip_id.", {$_REQUEST['responsavel']} )";
              $resultInsert = mysql_query($queryInsert, $link_sid) or
              handleError("leg_manu_equip.php", mysql_error());
              if(mysql_affected_rows()==1)
```

```
                {
                        //atualizo lw_equipment com os dados da nova manutenção
                        $queryUpdate = "UPDATE lw_equipment set";
                        $queryUpdate .= " equip_repair_cost = '".$_REQUEST["valor"]."'";
                        $queryUpdate .= " , equip_repair_description =
'".nl2br(htmlspecialchars($_REQUEST["descricao"])).."'";
                        $queryUpdate .= " , equip_repair_end = '".$previsao."'";
                        $queryUpdate .= " WHERE equip_id=".$frm_equip_id;
                        $resultUpdate = mysql_query($queryUpdate, $link_sid) or
                        handleError("leg_add_repair_request.php", mysql_error());

                        if(mysql_affected_rows()==1){
                                $msg = "Operação realizada com sucesso";
            }

        }
```

Figura III.103 – Codificação do caso de uso "Alterar chamado de manutenção".

III.3.8.14 .Solicitar compra de insumo

O solicita a compra de insumos através de dois arquivos. No primeiro, o *leg_buy_component_solicitation.php*, é apresentado um formulário que permite o preenchimento dos dados referentes ao pedido. No segundo, *leg_buy_component_solicitation_commit.php*, as informações serão gravadas no banco de dados. Na figura III.104 é apresentado o código de gravação das informações no banco de dados.

```
// Abre a conexão com o Banco de Dados
  $link_sid = mysql_connect($MYSQL_LOCATION, $MYSQL_ADMIN_USERNAME,
$MYSQL_ADMIN_PASSWORD);
  if (!mysql_select_db($MYSQL_DBNAME))
    handleError("leg_buy_component_solicitation.php", mysql_error());
//Guarda o login do solicitante para anexar ao e-mail
   $sql = "Select login_sid from user_base where user_id = ".$_REQUEST["email"];

   $result = mysql_query($sql);

   $info = mysql_fetch_array($result);
  //Guarda informações de hora
  $date = time();
  $body = "(".$info["login_sid"].") fez uma solicitação de compra de Componentes em
".date("d/m/Y H:i:s",$date);
  $body .= "\nDescrição do Material: ".$_REQUEST["description"];
  $body .= "\nQuantidade: ".$_REQUEST["quant"];
  $body .= "\nMotivação para a compra: ".$_REQUEST["motivation"];

  $queryInsert1 = "INSERT INTO lw_messages ( receiver_id, date, subject, text, status ) ";
  $queryInsert1 .= "values ( ".$_REQUEST["email"]., '".date("Ymd H:i:s",$date)."',
'Solicitação de Compra de Componentes', '$body', 2 )";

  $result_msg = mysql_query($queryInsert1) or
    handleError("leg_buy_component_solicitation.php", mysql_error());

  $msg_id = mysql_insert_id();

  $queryInsert2 = "INSERT INTO lw_xref_messages_user ( message_id, user_id ) values (
".$msg_id., ".decryptSessionVar($session_enc_user_id).")";

  $result_msg2 = mysql_query($queryInsert2) or
    handleError("leg_buy_component_solicitation.php", mysql_error());
?>
```

Figura III.104 – Codificação do caso de uso "Solicitar compra de insumo".

III.4 . Alteração e cargas iniciais do banco de dados

A concepção da estrutura do Banco de Dados SID [1] foi mantida ao longo da implementação do módulo, garantindo a expansibilidade e modularidade já previstos pela modelagem anterior.

III.4.1. Criação das novas tabelas e carga do banco de dados

A criação do banco de dados SID [1] foi feita de forma automática, através de um *script* SQL ("*legweb.sql*") que:

- Cria as tabelas base e auxiliares
- Adiciona índices e chaves nos campos correspondentes

O arquivo *"legweb.sql"* encontra-se na Figura III.105 para referência. Este *script* cria toda a estrutura do complemento ao banco de dados do SID [1].

```
#
# Table structure for table `lw_borrow_comp`
#
#

CREATE TABLE `lw_borrow_comp` (
  `borrow_comp_id` int(10) unsigned NOT NULL auto_increment,
  `borrow_comp_date` date NOT NULL default '0000-00-00',
  `borrow_comp_quant` int(10) unsigned NOT NULL default '0',
  `borrow_status` int(1) NOT NULL default '0',
  PRIMARY KEY  (`borrow_comp_id`)
) TYPE=MyISAM AUTO_INCREMENT=14 ;


# --------------------------------------------------------
#
# Table structure for table `lw_borrow_equip`
#
#

CREATE TABLE `lw_borrow_equip` (
  `borrow_equip_id` int(10) unsigned NOT NULL auto_increment,
  `borrow_equip_date` date NOT NULL default '0000-00-00',
  `borrow_equip_quant` int(10) unsigned NOT NULL default '0',
  `borrow_status` int(1) NOT NULL default '0',
  PRIMARY KEY  (`borrow_equip_id`)
) TYPE=MyISAM AUTO_INCREMENT=12 ;


# --------------------------------------------------------
```

```
#
# Table structure for table `lw_component`
#
#

CREATE TABLE `lw_component` (
  `component_id` int(10) unsigned NOT NULL auto_increment,
  `component_description` text NOT NULL,
  `component_tolerance` varchar(20) NOT NULL default '',
  `component_dimensions` varchar(20) NOT NULL default '',
  `component_acquired` int(10) unsigned NOT NULL default '0',
  `component_active` int(10) unsigned NOT NULL default '0',
  `component_out` int(10) unsigned NOT NULL default '0',
  `component_destroyed` int(10) unsigned NOT NULL default '0',
  `component_missing` int(10) unsigned NOT NULL default '0',
  `component_date` date NOT NULL default '0000-00-00',
  `component_nominal_value` varchar(20) NOT NULL default '',
  `component_rank_id` int(10) unsigned NOT NULL default '0',
  PRIMARY KEY  (`component_id`)
) TYPE=MyISAM AUTO_INCREMENT=10 ;

# --------------------------------------------------------

#
# Table structure for table `lw_equipment`
#
#

CREATE TABLE `lw_equipment` (
  `equip_id` int(10) unsigned NOT NULL auto_increment,
  `equip_name` varchar(40) NOT NULL default '',
  `equip_manufacturer` varchar(60) NOT NULL default '',
  `equip_acquire` date NOT NULL default '0000-00-00',
  `equip_value` varchar(30) NOT NULL default '',
  `equip_guarantee` date NOT NULL default '0000-00-00',
  `equip_dimension` varchar(30) NOT NULL default '',
  `equip_last_repair` date NOT NULL default '0000-00-00',
  `equip_repair_time` int(10) unsigned NOT NULL default '0',
  `equip_repair_cost` varchar(30) NOT NULL default '',
  `equip_repair_status` int(1) NOT NULL default '0',
  `equip_repair_description` text NOT NULL,
  `equip_control_number` varchar(20) NOT NULL default '',
  `equip_repair_contract` varchar(30) NOT NULL default '',
  `equip_repair_end` date NOT NULL default '0000-00-00',
  `equip_rank_id` int(10) unsigned NOT NULL default '0',
  PRIMARY KEY  (`equip_id`)
) TYPE=MyISAM AUTO_INCREMENT=11 ;

# --------------------------------------------------------
#
# Table structure for table `lw_grades`
#
#

CREATE TABLE `lw_grades` (
  `grade_id` int(10) unsigned NOT NULL auto_increment,
  `grade_1` float(4,3) NOT NULL default '0.000',
  `grade_2` float(4,3) NOT NULL default '0.000',
  `grade_3` float(4,3) NOT NULL default '0.000',
  `grade_4` float(4,3) NOT NULL default '0.000',
  `grade_5` float(4,3) NOT NULL default '0.000',
  `grade_6` float(4,3) NOT NULL default '0.000',
  `grade_7` float(4,3) NOT NULL default '0.000',
  `grade_8` float(4,3) NOT NULL default '0.000',
  `grade_9` float(4,3) NOT NULL default '0.000',
  `grade_10` float(4,3) NOT NULL default '0.000',
  `grade_11` float(4,3) NOT NULL default '0.000',
  `grade_12` float(4,3) NOT NULL default '0.000',
  `grade_13` float(4,3) NOT NULL default '0.000',
  `grade_14` float(4,3) NOT NULL default '0.000',
  `grade_15` float(4,3) NOT NULL default '0.000',
  PRIMARY KEY  (`grade_id`)
) TYPE=MyISAM AUTO_INCREMENT=5 ;

# --------------------------------------------------------
```

```
#
# Table structure for table `lw_group`
#
#

CREATE TABLE `lw_group` (
  `group_id` int(10) unsigned NOT NULL auto_increment,
  `group_name` varchar(20) NOT NULL default '',
  `group_class` varchar(20) NOT NULL default '',
  PRIMARY KEY  (`group_id`),
  UNIQUE KEY `group_name` (`group_name`)
) TYPE=MyISAM AUTO_INCREMENT=30 ;


# --------------------------------------------------------

#
# Table structure for table `lw_messages`
#
#

CREATE TABLE `lw_messages` (
  `message_id` int(10) unsigned NOT NULL auto_increment,
  `date` datetime NOT NULL default '0000-00-00 00:00:00',
  `subject` varchar(40) NOT NULL default '',
  `text` text NOT NULL,
  `status` int(1) NOT NULL default '0',
  `receiver_id` int(10) unsigned NOT NULL default '0',
  PRIMARY KEY  (`message_id`)
) TYPE=MyISAM AUTO_INCREMENT=49 ;

# --------------------------------------------------------

#
# Table structure for table `lw_rank`
#
# Creation: Aug 14, 2005 at 02:13 AM
# Last update: Nov 15, 2005 at 01:24 AM
#

CREATE TABLE `lw_rank` (
  `rank_id` int(10) unsigned NOT NULL auto_increment,
  `rank_type` int(10) unsigned NOT NULL default '0',
  `rank_name` varchar(40) NOT NULL default '',
  `rank_level` int(10) unsigned default NULL,
  `father_code` int(10) unsigned NOT NULL default '0',
  PRIMARY KEY  (`rank_id`,`rank_type`)
) TYPE=MyISAM AUTO_INCREMENT=60 ;

# --------------------------------------------------------

#
# Table structure for table `lw_repair_team`
#
#

CREATE TABLE `lw_repair_team` (
  `repair_team_id` int(10) unsigned NOT NULL auto_increment,
  `company_name` varchar(100) NOT NULL default '',
  `company_address_street` varchar(60) NOT NULL default '',
  `company_address_number` varchar(5) NOT NULL default '',
  `company_address_compl` varchar(20) NOT NULL default '',
  `company_address_town` varchar(20) NOT NULL default '',
  `company_address_city` varchar(30) NOT NULL default
  `company_address_state` char(2) NOT NULL default '',
  `company_address_zip` varchar(8) NOT NULL default '',
  `company_phone` varchar(8) NOT NULL default '',
  `company_phone_area_code` char(2) NOT NULL default '',
  `company_phone_country_code` char(2) NOT NULL default '',
  `responsable_phone` varchar(8) NOT NULL default ''
`responsable_phone_area_code` char(2) NOT NULL default '',
  `responsable_phone_country_code` char(2) NOT NULL default '',
  `responsable_phone_cellular` varchar(8) NOT NULL default '',
  `responsable_phone_cellular_area_code` char(2) NOT NULL default '',
  `responsable_phone_cellular_country_code` char(2) NOT NULL default '',
```

```
`company_cnpj` varchar(30) NOT NULL default '',
  `responsable_name` varchar(50) NOT NULL default '',
  `comments` text,
  `vendor_id` int(11) NOT NULL default '0',
  PRIMARY KEY  (`repair_team_id`)
) TYPE=MyISAM AUTO_INCREMENT=8 ;

# --------------------------------------------------------
#
# Table structure for table `lw_vendor`

CREATE TABLE `lw_vendor` (
`vendor_id` int(10) unsigned NOT NULL auto_increment,
  `company_name` varchar(100) NOT NULL default '',
  `company_address_street` varchar(60) NOT NULL default '',
  `company_address_number` varchar(5) NOT NULL default '',
  `company_address_compl` varchar(20) NOT NULL default '',
  `company_address_town` varchar(20) NOT NULL default '',
  `company_address_city` varchar(30) NOT NULL default '',
  `company_address_state` char(2) NOT NULL default '',
  `company_address_zip` varchar(8) NOT NULL default '',
  `company_phone` varchar(8) NOT NULL default '',
  `company_phone_area_code` char(2) NOT NULL default '',
  `company_phone_country_code` char(2) NOT NULL default '',
  `reselling_phone` varchar(8) NOT NULL default '',
  `reselling_phone_area_code` char(2) NOT NULL default '',
  `reselling_phone_country_code` char(2) NOT NULL default '',
  `reselling_phone_cellular` varchar(8) NOT NULL default '',
  `reselling_phone_cellular_area_code` char(2) NOT NULL default '',
  `reselling_phone_cellular_country_code` char(2) NOT NULL default '',
  `company_cnpj` varchar(30) NOT NULL default '',
  `reselling_name` varchar(50) NOT NULL default '',
  `comments` text,
  PRIMARY KEY  (`vendor_id`)
) TYPE=MyISAM AUTO_INCREMENT=18 ;

# --------------------------------------------------------
#
# Table structure for table `lw_xref_borrow_comp`

CREATE TABLE `lw_xref_borrow_comp` (
  `component_id` int(10) unsigned NOT NULL default '0',
  `borrow_comp_id` int(10) unsigned NOT NULL default '0',
  PRIMARY KEY  (`component_id`,`borrow_comp_id`)
) TYPE=MyISAM;

# --------------------------------------------------------

#
# Table structure for table `lw_xref_borrow_comp_user`
#
#

CREATE TABLE `lw_xref_borrow_comp_user` (
  `user_id` int(10) unsigned NOT NULL default '0',
  `borrow_comp_id` int(10) unsigned NOT NULL default '0',
  PRIMARY KEY  (`user_id`,`borrow_comp_id`)
) TYPE=MyISAM;

# --------------------------------------------------------
# Table structure for table `lw_xref_borrow_equip`
#

CREATE TABLE `lw_xref_borrow_equip` (
  `equip_id` int(10) unsigned NOT NULL default '0',
  `borrow_equip_id` int(10) unsigned NOT NULL default '0',
  PRIMARY KEY  (`equip_id`,`borrow_equip_id`)
```

```
) TYPE=MyISAM;

# --------------------------------------------------------

#
# Table structure for table `lw_xref_borrow_equip_user`
#
#

CREATE TABLE `lw_xref_borrow_equip_user` (
  `user_id` int(10) unsigned NOT NULL default '0',
  `borrow_equip_id` int(10) unsigned NOT NULL default '0',
  PRIMARY KEY  (`user_id`,`borrow_equip_id`)
) TYPE=MyISAM;

# --------------------------------------------------------

#
# Table structure for table `lw_xref_detailed_professor_group`
#
#

CREATE TABLE `lw_xref_detailed_professor_group` (
  `user_id` int(10) unsigned NOT NULL default '0',
  `group_id` int(10) unsigned NOT NULL default '0',
  PRIMARY KEY  (`user_id`,`group_id`)
) TYPE=MyISAM;

# --------------------------------------------------------

#
# Table structure for table `lw_xref_detailed_student_grade`
#
#

CREATE TABLE `lw_xref_detailed_student_grade` (
  `user_id` int(10) unsigned NOT NULL default '0',
  `grade_id` int(10) unsigned NOT NULL default '0',
  PRIMARY KEY  (`user_id`,`grade_id`)
) TYPE=MyISAM;

# --------------------------------------------------------

#
# Table structure for table `lw_xref_detailed_student_group`
#
#

CREATE TABLE `lw_xref_detailed_student_group` (
  `user_id` int(10) unsigned NOT NULL default '0',
  `group_id` int(10) unsigned NOT NULL default '0',
  PRIMARY KEY  (`user_id`,`group_id`)
) TYPE=MyISAM;

# --------------------------------------------------------

#
# Table structure for table `lw_xref_messages_user`
#
#

CREATE TABLE `lw_xref_messages_user` (
  `message_id` int(10) unsigned NOT NULL default '0',
  `user_id` int(10) unsigned NOT NULL default '0',
  PRIMARY KEY  (`message_id`,`user_id`)
) TYPE=MyISAM;

# --------------------------------------------------------
```

```
#
# Table structure for table `lw_xref_repair_team_equipment`
#
#

CREATE TABLE `lw_xref_repair_team_equipment` (
  `repair_team_id` int(10) unsigned NOT NULL default '0',
  `equip_id` int(10) unsigned NOT NULL default '0',
  PRIMARY KEY  (`repair_team_id`,`equip_id`)
) TYPE=MyISAM;

# --------------------------------------------------------

#
# Table structure for table `lw_xref_vendor_component`
#
#

CREATE TABLE `lw_xref_vendor_component` (
  `vendor_id` int(10) unsigned NOT NULL default '0',
  `component_id` int(10) unsigned NOT NULL default '0',
  PRIMARY KEY  (`vendor_id`,`component_id`)
) TYPE=MyISAM;

# --------------------------------------------------------

#
# Table structure for table `lw_xref_vendor_equipment`
#
#

CREATE TABLE `lw_xref_vendor_equipment` (
  `vendor_id` int(10) unsigned NOT NULL default '0',
  `equip_id` int(10) unsigned NOT NULL default '0',
  PRIMARY KEY  (`vendor_id`,`equip_id`)
) TYPE=MyISAM;

# --------------------------------------------------------

#
# Table structure for table `user_base`
#
#

CREATE TABLE `user_base` (
  `user_id` int(10) unsigned NOT NULL auto_increment,
  `login_sid` varchar(9) default NULL,
  `password` varchar(13) default NULL,
  `status_id` tinyint(3) unsigned NOT NULL default '0',
  `inclusion_date` date default NULL,
  `expiration_date` date default NULL,
  `last_access` datetime default NULL,
  `last_ip` varchar(15) default NULL,
  `last_modification` datetime default NULL,
  `login_net` varchar(16) default NULL,
  `name` varchar(50) default NULL,
  `gender` enum('M','F') default NULL,
  `birthdate` date default NULL,
  `birthdate_f` varchar(20) default NULL,
  `res_address_street` varchar(60) default NULL,
  `res_address_number` varchar(5) default NULL,
  `res_address_compl` varchar(20) default NULL,
  `res_address_town` varchar(20) default NULL,
  `res_address_city` varchar(30) default NULL,
  `res_address_state` char(2) default NULL,
```

```
`res_address_zip` varchar(8) default NULL,
  `res_address_f` varchar(20) default NULL,
  `res_phone` varchar(12) default NULL,
  `res_phone_f` varchar(20) default NULL,
  `phone_cellular` varchar(12) default NULL,
  `phone_cellular_f` varchar(20) default NULL,
  `email` varchar(50) default NULL,
  `email_f` varchar(20) default NULL,
  `url` varchar(100) default NULL,
  `url_f` varchar(20) default NULL,
  `quota_default` int(10) unsigned default NULL,
  `quota_available` int(10) unsigned default NULL,
  `admin_comment` varchar(100) default NULL,
  `res_phone_area_code` char(2) default NULL,
  `res_phone_country_code` char(2) default NULL,
  `phone_cellular_country_code` char(2) default NULL,
  `phone_cellular_area_code` char(2) default NULL,
  `lw_user_status` int(1) NOT NULL default '0',
  `lw_access_level` int(1) NOT NULL default '0',
  `lw_last_access` datetime default NULL,
  PRIMARY KEY  (`user_id`),
  UNIQUE KEY `login_net` (`login_net`),
  UNIQUE KEY `login_sid` (`login_sid`),
  KEY `login_sid_2` (`login_sid`,`login_net`,`email`)
) TYPE=MyISAM AUTO_INCREMENT=2371 ;

# --------------------------------------------------------
```

Figura III.105 – Cadastro de tabelas do Legweb no banco de dados do SID [1].

Capítulo IV . Conclusão

O LegWeb (Módulo de Gerenciamento do LEG) possui um grande potencial para se tornar uma ferramenta de grande importância para o Laboratório de Eletrônica da Graduação (LEG) não só pelas tarefas que já realiza nesta primeira versão, mas principalmente por permitir a inclusão de outras tarefas que podem ser agregadas a ele.

A abrangência do projeto, mesmo em se considerando o nível de desenvolvimento da primeira versão, é grande e significativa para o laboratório, permitindo diversas funcionalidades e aplicações para melhorias em processos internos do LEG.

IV.1 . Benefícios e Usos

Os usos do LegWeb, com as funções já implementadas e funcionando, permitem o uso de tarefas úteis ao laboratório, como por exemplo:

IV.1.1 . Gerenciamento remoto das facilidades

Os alunos poderão agendar de casa, via Internet, o empréstimo de material para as experiências. Isso evita a perda de tempo no preenchimento de fichas de empréstimo no balcão do LEG.

IV.1.2 . Gerenciamento de Usuários

O gerenciamento de usuários do módulo é feito através da estrutura base de autenticação do SID [1]. Entretanto, alterações no banco de dados do SID [1] permitiram o cadastro de alunos do DEL como usuários do módulo.

Desta forma, utilizando a integração com a base de usuários do SID [1], o módulo pode facilmente acessar qualquer informação de cadastro dos seus usuários dentro do DEL.

IV.1.3 . Gerenciamentos de componentes

Cada usuário tem o direito de cadastrar empréstimos de componentes na lista particular de empréstimo. Dessa forma, o controle sobre o empréstimo de componentes fica personalizado, ressaltando o fato de que cada aluno é responsável pelos empréstimos a ele concedidos.

O sistema permite ainda o controle do estoque de componentes, gerando relatórios com relação ao número de componentes com problemas, baixo número de um dado componente no estoque, etc.

IV.1.4 . Gerenciamento de equipamentos

Além das mesmas funcionalidades dos empréstimos, o usuário pode cadastrar chamados de manutenção de equipamentos e acompanhar o andamento de sua solicitação. A ferramenta permite ainda o armazenamento do histórico de defeitos de cada ativo cadastrado.

IV.1.5 . Cadastro de fornecedores

É mantido um cadastro dos fornecedores de equipamentos / componentes com todos os dados cadastrais das firmas, últimas vendas e valor de faturas por empresa.

IV.1.6 . Gerenciamento de graus

Os professores poderão inserir as notas dos laboratórios e estas serão visíveis pelo grupo de alunos.

IV.1.7 . Publicação de informações de aulas

Professores, além de cadastrar as notas de cada aula , poderão cadastrar também avisos importantes , disponibilizar material para download, etc.

Cada grupo de alunos poderá visualizar suas notas individualmente, sendo vedado o acesso aos graus de grupos ao qual o usuário não pertença.

IV.1.8 . Outros benefícios

Os benefícios trazidos pelo LegWeb são não facilmente dimensionáveis, já que a grande contribuição do esforço de pesquisa do LegWeb é no sentido de viabilizar projetos futuros de maneira segura e rápida, aumentando a simplicidade do acesso às informações, melhorando a qualidade dos controles do laboratório.

Referências Bibliográficas

[1] VIEIRA, Pedro., Sistema Intergrado de Serviços e Informações do DEL/UFRJ. Projeto Final, UFRJ, 2002.

[2] Fedora Linux: http://www.redhat.com/fedora/

Acesso em: 12 de novembro de 2006.

[3] MySQL: http://www.mysql.com

Acesso em: 12 de novembro de 2006.

[4] PHP: http://www.php.net

Acesso em: 12 de novembro de 2006.

Anexo A – Acrônimos Utilizados

Em todo o documento foram utilizados diversos acrônimos que podem, de maneira de fácil acesso, serem consultados a seguir:

B2C: Negócio para o Consumidor (*Business to Consummer*)

DEL: Departamento de Eletrônica e Computação

DDL: Linguagem de Definição de Dados (*Data Definition Language*)

HTML: *Hypertext Markup Language*

HTTP: *Hypertext Transmission Protocol*

LEG: Laboratório de Eletrônica da Graduação

LegWeb: Módulo de Gerenciamento do LEG

OMG: Grupo de Gerenciamento de Objeto (*Object Management Group*)

OO: Orientado a Objeto (*Object Oriented*)

PHP: *Pre Hypertext Processor* [4]

PK – Chave primária (*Primary Key*)

RFT: Força Tarefa de Revisão (*Revision Task Force*)

SGBD: Sistema Gerenciador de Banco de Dados

SID: Sistema Integrado DEL [1]

SSL: *Secure Sockets Layer*

SQL: *Structured Query Language*

UFRJ: Universidade Federal do Rio de Janeiro

UML: *Unified Modeling Language*

WWW: *World Wide Web*

Anexo B – Requisitos de *Hardware* e *Software*

O LegWeb, como módulo complementar ao SID [1], depende dos requisitos deste para seu pleno funcionamento.

A instalação e configuração do SID [1] pressupõe uma plataforma de hardware e software que sejam capazes de suportá-lo. Abaixo estão listados os requisitos que são estritamente necessários na instalação e configuração do SID [1] em um computador.

Os demais requisitos necessários serão instalados conforme instruções presentes no próximo Apêndice.

Requisitos de hardware

- Computador compatível com Intel x86, Pentium ou superior.
- Memória RAM mínima de 32Mb.
- Placa de Rede.

Requisitos de software

- Sistema Operacional FreeBSD versão 4.2 ou superior.
- Perl.
- TCL.
- Apache suporte a SSL.

Anexo C – Manual do usuário

Este manual tem por objetivo explicar os principais recursos, funções e comandos do Módulo de Gerenciamento do LEG, o LegWeb.

Acesso ao Módulo:

O acesso ao LegWeb se dá através do SID [1]. Após a autenticação será feita através da página apresentada na figura A.1.

Figura A.1 – Tela de acesso ao SID [1].

A seguir o usuário é levado ao cardápio de opções do SID [1], onde deverá pressionar o atalho para o módulo LegWeb (Acesso ao LEG), conforme a figura A.2.

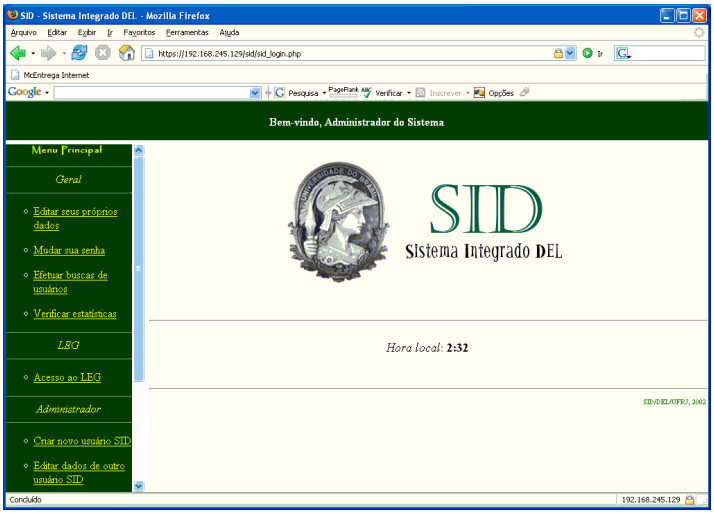


Figura A.2 – Tela de cardápio do SID [1].

Ao pressionar neste atalho, é aberta uma segunda janela no navegador e o usuário é levado à tela de cardápio do módulo LegWeb, conforme figura A.3.

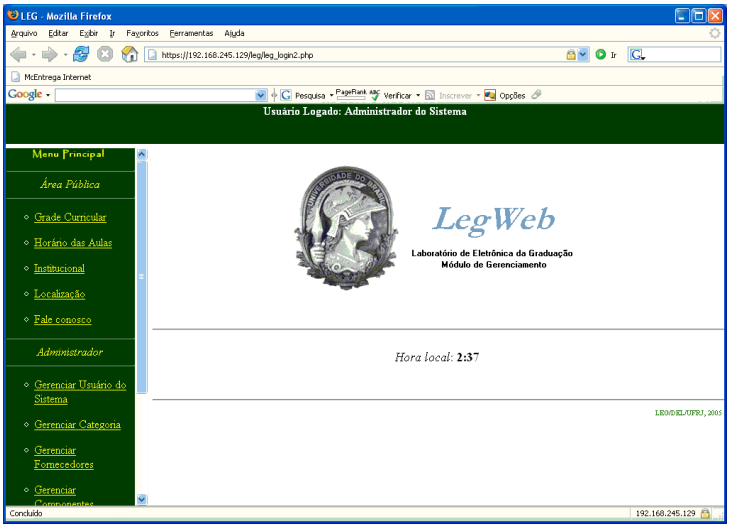


Figura A.3 – Tela de cardápio do LegWeb.

A partir da tela de cardápio do LegWeb, o usuário pode navegar de acordo com o seu perfil (administrador, aluno ou professor).

Gerenciamento de usuários:

O gerenciamento de usuários ocorre através da interface apresentada na figura A.4, onde o usuário pode gerenciar um grupo do laboratório. Note que a seleção do grupo é feita

através de uma tela de rolagem e que, abaixo, são apresentadas as opções de manutenção previstas na documentação.

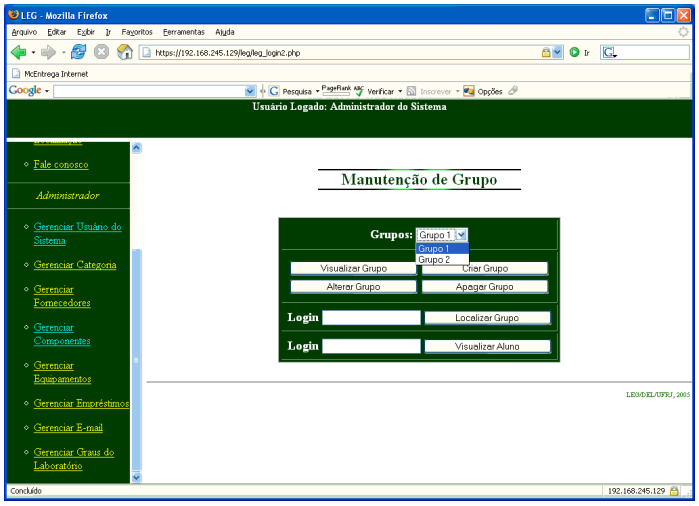


Figura A.4 – Tela de gerenciamento de usuários.

Gerenciamento de categoria:

O gerenciamento de categoria ocorre através da interface apresentada na figura A.5, onde o usuário pode selecionar em que tipo de insumo deseja criar uma categoria (Componente ou Equipamento) e, através de uma barra de rolagem, selecionar as categorias desejadas e criar, apagar ou excluir categorias.

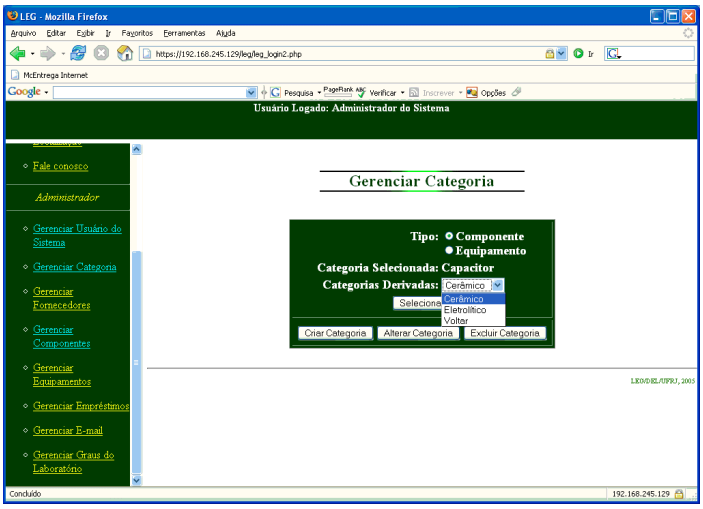


Figura A.5 – Tela de gerenciamento de usuários.

Gerenciamento de fornecedores:

O gerenciamento de fornecedores ocorre através da interface apresentada na figura A.6. O usuário pode selecionar o fornecedor desejado através da barra de rolagem e escolher a opção de manutenção mais adequada.

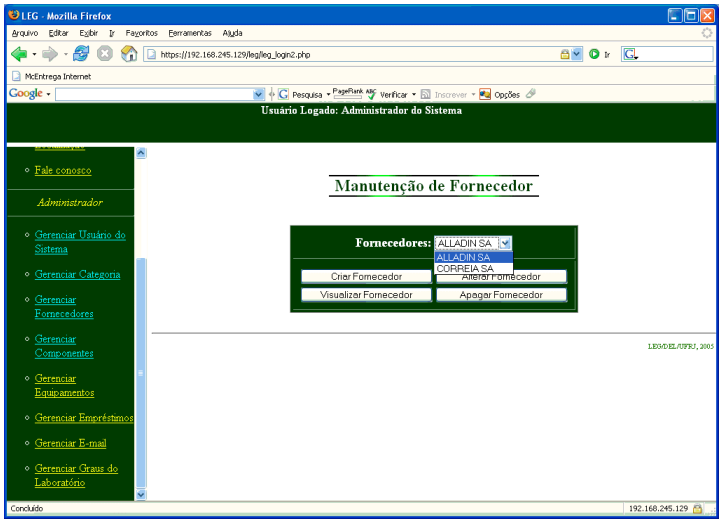


Figura A.6 – Tela de gerenciamento de fornecedor.

Gerenciamento de componente:

O gerenciamento de componente ocorre através da interface apresentada na figura A.7. O usuário pode selecionar o componente e suas subcategorias desejadas através da barra de rolagem e escolher a opção de manutenção mais adequada.

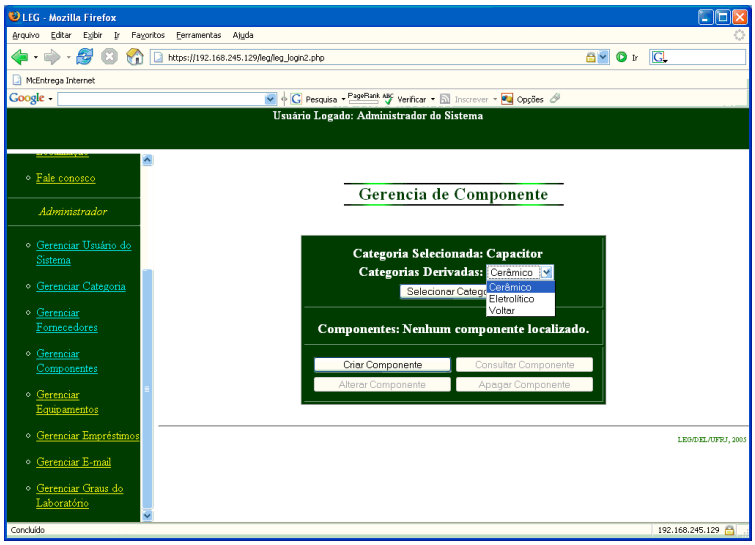


Figura A.7 – Tela de gerenciamento de componente.

Gerenciamento de equipamento:

O gerenciamento de equipamento ocorre através da interface apresentada na figura A.8. O usuário pode selecionar o equipamento e suas subcategorias desejadas através da barra de rolagem e escolher a opção de manutenção mais adequada.

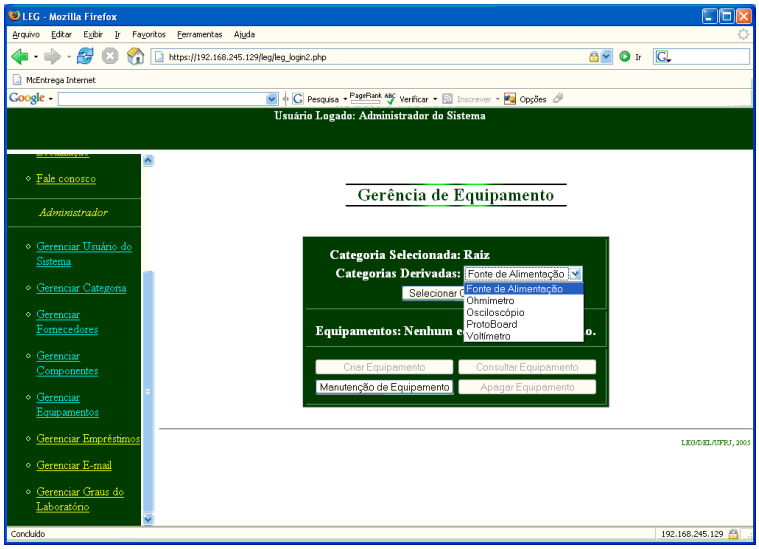


Figura A.8 – Tela de gerenciamento de equipamento.

Manutenção de equipamento:

A manutenção de equipamento ocorre através da interface apresentada na figura A.9. O usuário pode selecionar o equipamento em manutenção através da barra de rolagem e escolher a opção de gerenciamento mais adequada.

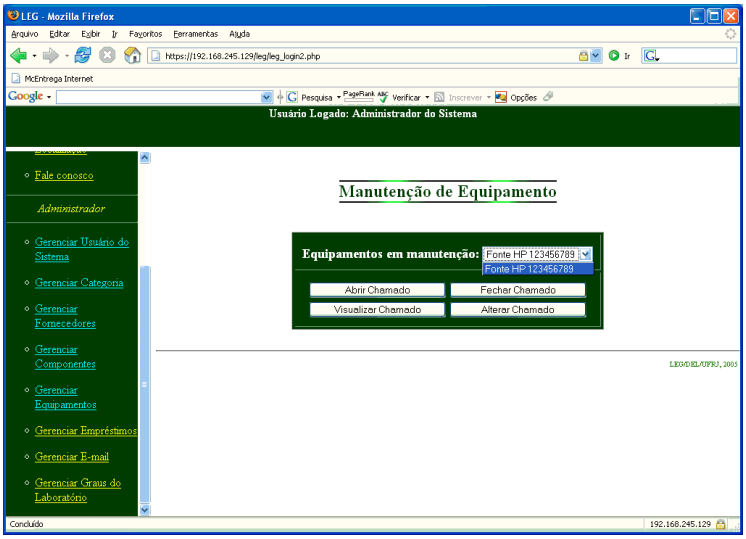


Figura A.9 – Tela de manutenção de equipamento.

Gerenciamento de empréstimo:

O gerenciamento de empréstimo ocorre através da interface apresentada na figura A.10. O usuário pode selecionar um empréstimo já realizado através da barra de rolagem e escolher a opção de manutenção mais adequada.

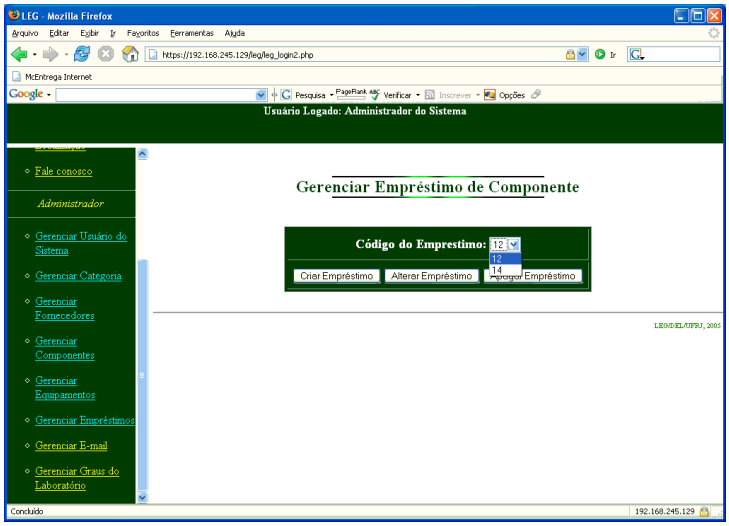


Figura A.10 – Tela de gerenciamento de empréstimo.

Gerenciamento de *e-mail*:

O gerenciamento de *e-mail* ocorre através da interface apresentada na figura A.11. O usuário pode selecionar uma mensagem através dos atalhos apresentados em tela para visualizar seu conteúdo.

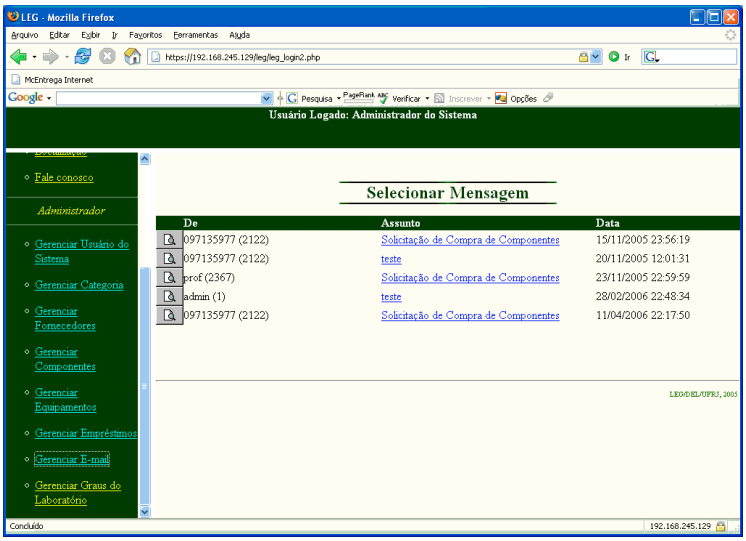


Figura A.11 – Tela de gerenciamento de *e-mail*.

Gerenciamento de notas de disciplinas de laboratório:

O gerenciamento das notas das disciplinas de laboratório é feito através da interface apresentada na figura A.12. O usuário seleciona um grupo, pressionando sobre um dos ícones apresentados ao lado do nome do grupo de laboratório.

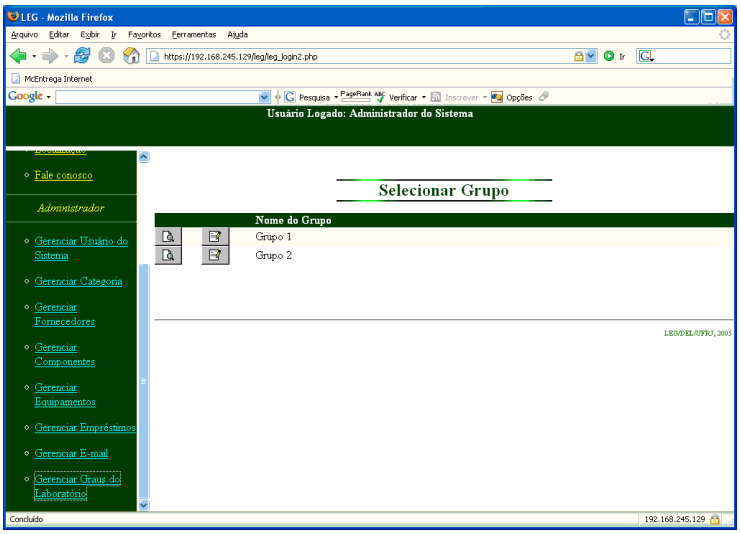


Figura A.12 – Tela de gerenciamento de seleção de grupo.

Em seguida o usuário pode visualizar ou alterar as notas dos membros do grupo, individualmente, através da interface apresentada na figura A.13.

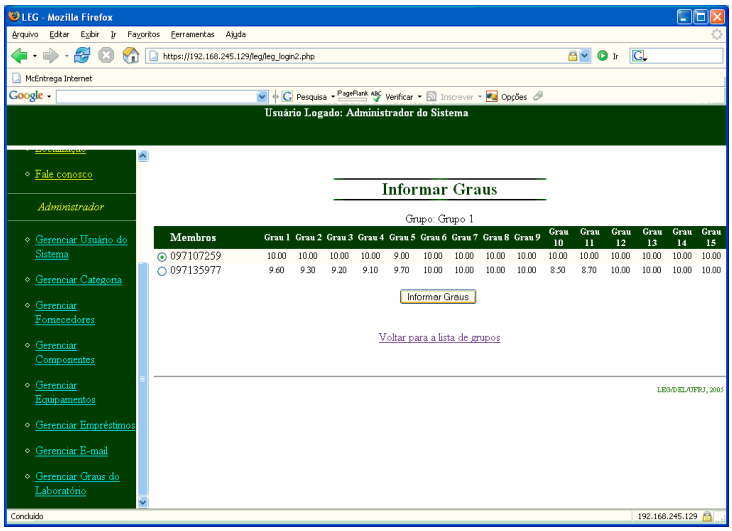


Figura A.13 – Tela de gerenciamento de notas das disciplinas de laboratório.

Anexo D– CD-ROM

Cumprindo exigência do Departamento de Engenharia Eletrônica e Computação, um CD-ROM acompanha esta documentação impressa. Seu conteúdo está expresso abaixo, na árvore de diretórios.

· /apresentacao: Contém em formato PDF a apresentação dos slides da defesa deste projeto.

· /documentacao: Contém este documento em formato PDF.

· /html: Contém os arquivos HTML e PHP do LegWeb.

· /legweb.mysql: Contém todos os arquivos necessários à carga e à criação do banco de dados LegWeb, no mysql.