UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CURSO SUPERIOR DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

EDSON GUSTAVO TOFOLO

SISTEMA PARA GERENCIAMENTO DE SALAS DE APOIO E LABORATÓRIOS DIDÁTICOS

TRABALHO DE CONCLUSÃO DE CURSO

PATO BRANCO
2015

EDSON GUSTAVO TOFOLO

SISTEMA PARA GERENCIAMENTO DE SALAS DE APOIO E LABORATÓRIOS DIDÁTICOS

Trabalho de Conclusão de Curso de graduação, apresentado à disciplina de Trabalho de Conclusão de Curso 2, do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, da Universidade Tecnológica Federal do Paraná, Câmpus Pato Branco, como requisito parcial para obtenção do título de Tecnólogo.

Orientador: Profa. Beatriz Terezinha Borsoi

PATO BRANCO
2015

ATA Nº: **001**

DEFESA PÚBLICA DO TRABALHO DE CONCLUSÃO DE CURSO DO ALUNO **EDSON GUSTAVO TOFOLO**.

Às 20:25 hrs do dia 9 de junho de 2015, Bloco V da UTFPR, Câmpus Pato Branco, reuniu-se a banca avaliadora composta pelos professores Beatriz Terezinha Borsoi (Orientadora), Vinicius Pegorini (Convidado) e Alessandro Graczyk Moraes (Convidado), para avaliar o Trabalho de Conclusão de Curso do aluno Edson Gustavo Tofolo, matrícula 1437119, sob o título **Sistema para gerenciamento de salas de apoio a laboratórios didáticos**; como requisito final para a conclusão da disciplina Trabalho de Conclusão de Curso 2 do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, COADS. Após a apresentação o candidato foi entrevistado pela banca examinadora, e a palavra foi aberta ao público. Em seguida, a banca reuniu-se para deliberar considerando o trabalho **APROVADO**. Às 21:10 hrs foi encerrada a sessão.

Profa. Beatriz Terezinha Borsoi, Dr.
Orientadora

Prof. Vinicius Pegorini, Esp.
Convidado

Prof. Alessandro Graczyk Moraes,
Convidado

Prof. Soelaine Rodrigues Ascari, M.Sc.
Coordenador do Trabalho de Diplomação

Prof. Edilson Pontarolo, Dr.
Coordenador do Curso

RESUMO

TOFOLO, Edson Gustavo. Sistema para gerenciamento de salas de apoio e laboratórios didáticos. 2015. 60 f. Monografia (Trabalho de Conclusão de Curso) - Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, Universidade Tecnológica Federal do Paraná, Câmpus Pato Branco. Pato Branco, 2015.

Ambientes universitários como laboratórios que são utilizados para realização de aulas práticas e que envolvem o desenvolvimento de projetos por alunos estão sujeitos a empréstimo de equipamentos e materiais, inclusive de consumo, para desenvolverem esses projetos. Nesses ambientes também é necessário ter o controle de aulas, agendamento de realização de aulas práticas e reserva para realização de atividades de projetos. Na Universidade Tecnológica Federal do Paraná, Câmpus Pato Branco, existem diversos ambientes que possuem essas características como, por exemplo, os laboratórios de química, de alimentos, sala de apoio do Departamento de Elétrica e o laboratório de hardware e software da Área de Informática. Esses ambientes possuem particularidades e especificidades que dificultam o uso de um mesmo sistema de informação. Um aplicativo, como resultado da realização deste trabalho, foi desenvolvido para esses dois ambientes: sala de apoio e laboratório de hardware e software. Os sistemas possuem diversas funcionalidades iguais e algumas distintas, assim, as bases de dados são distintas para os Departamentos Acadêmicos da Informática (DAINF) e da Elétrica (DAELE). No DAELE, o gerenciamento é realizado por meio de uma sala de apoio que coordena os empréstimos e as reservas. No DAINF o gerenciamento está vinculado a um laboratório porque é nesse ambiente que são realizados os projetos que envolvem componentes e equipamentos que são emprestados. Esse sistema facilitará o gerenciamento de reservas e empréstimos de equipamentos e materiais, controle de estoque e manutenções realizadas em equipamentos de salas de apoio e laboratórios didáticos.

Palavras-chave: Rich Internet Application. Gerenciamento de laboratórios. Aplicações web.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTAGENS DE CÓDIGOS

LISTA DE SIGLAS

AJAX	Asynchronous JavaScript and XML
API	*Application Programming Interface*
CRUD	*Create, Retrieve, Update and Delete*
DAELE	Departamento Acadêmico da Elétrica
DAINF	Departamento Acadêmico de Informática
DAO	*Data Access Object*
HTML	*HiperText Markup Language*
HTTP	*Hypertext Transfer Protocol*
IoC	Inversão de Controle
Java EE	*Java Enterprise Edition*
JSF	*JavaServer Faces*
JSP	*JavaServer Pages*
MVC	*Model-View-Controller*
ORM	*Object-Relational Mapping*
RIA	*Rich Internet Application*
RF	Requisito Funcional
RNF	Requisito Não Funcional
TCP/IP	*Transmission Control Protocol/Internet Protocol*
UTFPR	Universidade Tecnologia Federal do Paraná
WWW	*Word Wide Web*
XML	*Extensible Markup Language*

SUMÁRIO

1 INTRODUÇÃO

Este capítulo apresenta a introdução que é composta pelas considerações iniciais com o escopo e o contexto do trabalho, os objetivos e a justificativa. O texto é finalizado com a apresentação dos capítulos subsequentes.

1.1 CONSIDERAÇÕES INICIAIS

O Departamento Acadêmico de Elétrica (DAELE) da Universidade Tecnologia Federal do Paraná (UTFPR), Câmpus Pato Branco, possui uma sala de apoio que armazena os materiais e os equipamentos utilizados em aulas práticas e em projetos. O Departamento Acadêmico de Informática (DAINF), também da UTFPR Câmpus Pato Branco, possui um laboratório caracterizado como de hardware e software com objetivos semelhantes à sala de apoio do DAELE. Além desse laboratório, o DAINF possui outros ambientes que são reservados para aulas e atividades como cursos de extensão e apresentações de estágio e de trabalho de conclusão de curso, além de aulas. O DAELE também realiza a reserva de seus laboratórios para a realização de aulas de professores, projetos e outros.

A gestão do Laboratório de Hardware e Software do DAINF tem sido realizada de maneira informal e sem procedimentos estabelecidos de controle. Esse controle tem sido realizado por um docente que é responsável pelos laboratórios por meio de uma planilha de cálculos. Os procedimentos adotados no empréstimo dos materiais não têm sido efetivos e isso tem ocasionando extravio de componentes eletrônicos e o desaparecimento de equipamentos. A não efetividade decorre da inexistência de um sistema de registro. Componentes foram emprestados e muitos desses componentes não foram devolvidos e não é possível saber para quem foi realizado o empréstimo. Esses componentes foram emprestados para serem utilizados em projetos desenvolvidos por alunos com ou sem vínculo com professores, mas sem um controle não é possível identificar os destinatários dos empréstimos que não efetuaram as devidas devoluções.

Na sala de apoio do DAELE, o controle tem sido realizado por meio de fichas impressas. Essa sala de apoio concentra o gerenciamento de vários laboratórios e o empréstimo de materiais para projetos de pesquisa. O controle manual tem sido trabalhoso para os professores realizarem as solicitações de materiais para aulas práticas e para os

atendentes desse ambiente que realizam o controle das reservas de laboratórios, de devolução de equipamentos e materiais e do envio de materiais para conserto, por exemplo.

Esse contexto evidencia o problema da falta de efetividade dos controles sendo utilizados nesses dois ambientes e da inexistência de gerenciamento dos materiais do Laboratório de Hardware e Software do DAINF. Assim, verificou-se a necessidade de desenvolver um sistema que possa contribuir para a gestão dos materiais emprestados, das reservas para as aulas e outras atividades e do estoque de materiais e equipamentos desses ambientes. Por serem ambientes com características e funcionalidades específicas, um sistema para o gerenciamento desses ambientes deve ser desenvolvido sob demanda. Os sistemas para esses dois Departamentos possuem funcionalidades compartilhadas e algumas específicas. Eles serão implementados em bases de dados distintas, facilitando o gerenciamento do estoque e o controle dos empréstimos. Os usuários desses dois sistemas são, em ampla maioria, distintos.

Para facilitar o acesso e o uso pelos professores dos referidos Departamentos, o sistema desenvolvido é para ambiente Internet e com o uso de tecnologias que permitam implementar uma interface caracterizada como rica. Essas aplicações são as *Rich Internet Application* (RIA).

RIAs são aplicações *web* caracterizadas pela usabilidade melhorada de sua interface se comparadas às aplicações *web* tradicionais e por prover uma experiência para o usuário satisfatória, dinâmica e interativa (AMALFITANO et al., 2010). As aplicações *web* tradicionais são as baseadas em *HiperText Markup Language* (HTML) que possuem interface composta por componentes de formulário e outros bastante simples. As RIAs se propõem a promover melhor interatividade em decorrência dos recursos de interface utilizados. Assim, a associação da interatividade provida pelas RIAs com as aplicações *desktop* é bastante evidente. As aplicações *desktop* possuem muitos recursos de interface que facilitam o uso dos aplicativos, como os efeitos de arrastar-e-soltar e menus e botões diferenciados. Deb, Bannur e Bharti (2007) ressaltam que essas aplicações objetivam combinar os recursos e a intuitividade que as aplicações *desktop* oferecem aos usuários com a facilidade de comunicação provida pela Internet.

1.2 OBJETIVOS

A seguir são apresentados o objetivo geral e os objetivos específicos deste trabalho.

1.2.1 Objetivo Geral

Desenvolver um sistema para gerenciamento da Sala de Apoio do Departamento Acadêmico de Elétrica e do Laboratório de Hardware e Software do Departamento Acadêmico de Informática.

1.2.2 Objetivos Específicos

- Facilitar a reserva de ambientes, equipamentos e materiais dos laboratórios do DAELE e DAINF.
- Facilitar a reserva e a organização dos materiais necessários para a realização de aulas práticas de professores.
- Prover um controle de estoque que permita a baixa de itens de estoque com justificativa da ocorrência.
- Facilitar o gerenciamento dos equipamentos em manutenção.
- Proporcionar o controle dos materiais emprestados para utilização em projetos realizados por alunos e professores.

1.3 JUSTIFICATIVA

Um software para gerenciamento dos materiais da sala de apoio do DAELE e do Laboratório de Hardware e Software do DAINF auxiliará no gerenciamento dos empréstimos e no controle dos materiais existentes nesses ambientes. O sistema facilitará, ainda, o controle de estoque desses ambientes.

A sua implementação se justifica pela agilidade na realização dos trabalhos, pela facilidade provida aos atendentes desses ambientes - sejam técnicos administrativos, estagiários ou professores - e para os professores que fazem reservas de materiais e equipamentos para aulas práticas e empréstimos para projetos.

O sistema foi implementado para uso na *web*, facilitando, assim, a reserva de materiais e equipamentos que são realizadas por professores. Assim, não haverá mais a necessidade de o professor deslocar-se fisicamente até o laboratório ou ligar para fazer a reserva e o acompanhamento do estado da mesma.

O sistema proverá um controle de estoque que permitirá identificar os materiais com quantidades mínimas de estoque e os equipamentos em conserto e o histórico dos consertos, entre outras funcionalidades. O controle de materiais, como os componentes que não são retornados após o uso - por eles terem sido, por exemplo, soldados em placa de circuito impresso - é de extrema relevância para que a compra dos mesmos possa ser planejada e executada de forma a não causar desabastecimento.

As tecnologias JavaServer Faces, PrimeFaces e Spring foram escolhidas em decorrência dos recursos que elas oferecem para o desenvolvimento de aplicações *web* denominadas de interface rica. PrimeFaces, como uma biblioteca, possui componentes que fornecem funcionalidades diversas para o desenvolvimento do sistema, especialmente em termos de interface.

1.4 ESTRUTURA DO TRABALHO

Este texto está organizado em capítulos. O Capítulo 2 apresenta o referencial teórico, centrado em aplicações internet rica por ser o tipo de aplicação desenvolvido como resultado deste trabalho de conclusão de curso. As ferramentas e as tecnologias utilizadas na modelagem e na implementação estão apresentadas no Capítulo 3. No Capítulo 4 estão os resultados da realização do trabalho, representados por diagramas e complementações textuais, a apresentação do sistema desenvolvido por meio de suas telas e partes de código. Por fim estão as considerações finais, definidas por conclusão, e as referências bibliográficas utilizadas na composição do texto.

2 REFERENCIAL TEÓRICO

Este capítulo apresenta o referencial teórico do trabalho que está centrado em aplicações para Internet que são caracterizadas como de interface rica.

2.1 APLICAÇÕES WEB TRADICIONAIS

A Internet foi criada como uma rede militar de interconexão de computadores e evoluiu muito rapidamente para uma plataforma global, provendo comunicação por meio de protocolos padronizados e abertos (VAZ *et al*., 2012). A popularização da Internet tornou-se mais evidente com o surgimento da *Word Wide Web* (WWW). A *web* também auxiliou a Internet a ser utilizada como uma ferramenta para as mais diversas finalidades, como, por exemplo: entretenimento provido pelas redes sociais; comércio e negócio dos mais variados produtos; serviços em diversos segmentos; instrução por meio dos cursos de educação a distância curriculares ou não e serviços governamentais oferecidos aos cidadãos. Além de uma vasta quantidade de informações dos mais diversos assuntos.

A *web* foi desenvolvida com o objetivo de ser uma plataforma para permitir o acesso de conteúdo estático e dinâmico codificado em linguagem de marcação de hipertexto, a HTML (FRATERNALI; ROSSI; SÁNCHEZ-FIGUEROA, 2010). Nas aplicações *web* baseadas em HTML a interação do usuário fica limitada à navegação baseada em *links* e inserção de dados em formulários com componentes bastante simples. Essas aplicações são caracterizadas como HTTP/HTML, por serem baseadas na combinação de *Hypertext Transfer Protocol* (HTTP) e HTML. A aplicação fica armazenada em um servidor, com o cliente acessando dados por meio de uma página *web*.

Apesar das limitações em termos de recursos de interação (interface com o usuário) e de tráfego de rede (aumento de tráfego pelo carregamento completo da página a cada iteração e não realização de processamento ou persistência no cliente), por exemplo, a arquitetura cliente-servidor definiu um modelo simples e universal, no qual não há necessidade de instalação do aplicativo no cliente. Contudo, Fraternali, Rossi e Sánchez-Figueroa (2010) ressaltam que nessa arquitetura a qualidade das aplicações é muito limitada em relação aos recursos que a Internet oferece. E esses autores complementam que soluções *web* modernas assemelham-se às aplicações *desktop*, permitindo recursos de interação sofisticados, processamento no lado cliente, comunicação assíncrona e multimídia.

As aplicações *web* atuais têm sido construídas com foco no usuário (por exemplo: as redes sociais ou as soluções de gerenciamento de conteúdo gerado pelo usuário) que demanda alto grau de usabilidade e capacidade de interação. Fraternali, Rossi e Sánchez-Figueroa (2010) ressaltam que a arquitetura *web* baseada no padrão HTTP/HTML falha em fornecer suporte aos requisitos das novas aplicações *web* em vários aspectos, como:

a) Apresentação – somente *widgets* (programas que visam facilitar o acesso a serviços e utilidades) e contêineres predefinidos em HTML são disponibilizados.
b) Comunicação – suporta somente interações síncronas nativas em HTTP e requer que os mecanismos de retorno para comportamento assíncrono sejam simulados no topo HTTP de um ciclo de resposta do cliente ou implementação em baixo nível em *Transmission Control Protocol/Internet Protocol* (TCP/IP).
c) Lógica de negócio – ocorre principalmente no lado servidor.
d) Dados de gerenciamento – capacidade de armazenamento de dados no lado cliente são limitadas (como pelo uso de *cookies*).

As formas de tratamento dessas limitações conduziram ao desenvolvimento das aplicações *web* denominadas *Rich Internet Application* (RIA), traduzidas como aplicações internet ricas Embora o termo "rica" esteja costumeiramente associado aos recursos de interface, considerados ricos pelas possibilidades de interação semelhantes aos recursos oferecidos pelas aplicações *desktop*; outros requisitos como a redução de tráfego de rede, por exemplo, também caracterizam essas aplicações *web*.

2.2 APLICAÇÕES INTERNET RICA

O termo RIA se refere a uma família heterogênea de soluções caracterizadas por um objetivo comum que é o de adicionar funcionalidades e recursos às aplicações *web* convencionais, ou tradicionais, que são baseadas em hipertexto.

As RIAs combinam a arquitetura de distribuição das aplicações *web* com a interatividade e capacidade das aplicações *desktop*. As RIAs oferecem uma experiência com o usuário mais responsiva e ampla do que as aplicações *web* tradicionais. Essas aplicações representam a convergência de duas culturas de desenvolvimento (MELIÁ et al., 2010): *desktop* e aplicações *web*.

O resultado dessa combinação melhora todos os elementos de uma aplicação *web* (dados, lógica de negócio, comunicação e apresentação) (FRATERNALI; ROSSI; SÁNCHEZ-FIGUEROA, 2010).

Existem muitos *frameworks* para desenvolvimento de aplicações *web*. *Framework* é definido como um conjunto de classes que colaboram para realizar uma responsabilidade para um domínio de subsistema de aplicação (FAYAD; SCHMIDT, 1996). De forma prática, os *frameworks* visam auxiliar na implementação de aplicativos, reduzindo a quantidade de código que precisa ser desenvolvida e provendo reuso. E, assim, visam reduzir custos e melhorar a qualidade do software.

Os *frameworks* incorporam padrões de projetos e tecnologias como *Model-View-Controller* (MVC), internacionalização, *Asynchronous JavaScript and* XML (AJAX), *Object-Relational Mapping* (ORM) e são implementados para funcionalidades distintas como gerência de *logs*, gerenciador de transações, agendamento de tarefas e persistência, entre outros. Considerando os padrões, as tecnologias e as funcionalidades que implementam, utilizam e oferecem, os *frameworks* são organizados em diversas categorias. Fanzini (2013), por exemplo, agrupou os *frameworks* para desenvolvimento em Java em 35 categorias.

A Figura 1 apresenta a linha do tempo de alguns dos *frameworks web* existentes e o ano de surgimento. Essa figura fornece uma visão geral, ainda que simplificada, da diversidade de *frameworks* e de tecnologias envolvidas.

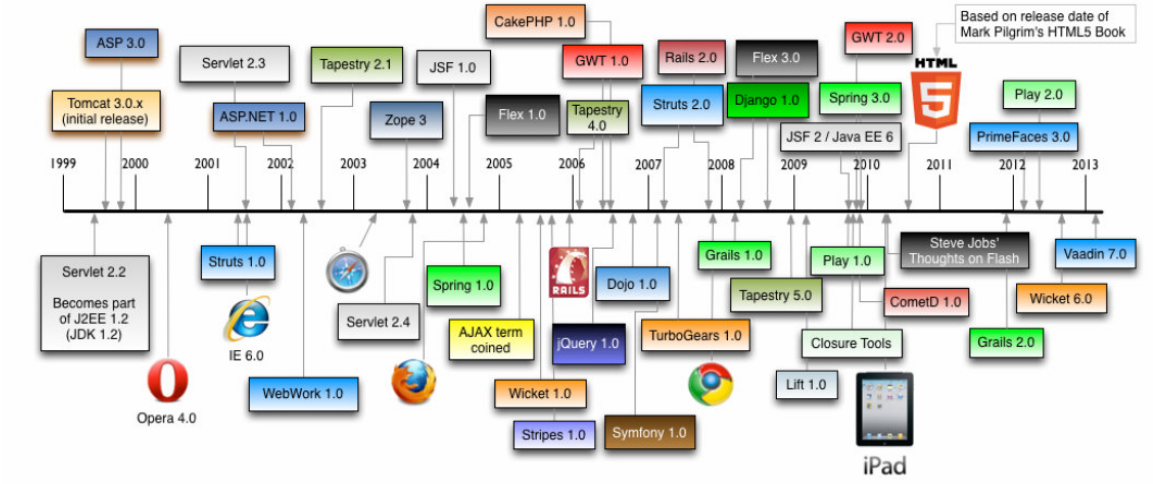

Figura 1 – Histórico de surgimento de alguns frameworks web
Fonte: Raible (2013, p. 12).

A Figura 1 apresenta apenas parte da linha do tempo dessas tecnologias, mesmo porque Tomcat e ASP constam a partir da sua versão 3. Contudo, ainda assim é possível verificar a quantidade e diversidade existente dessas tecnologias.

2.3 JAVASERVER FACES

JavaServer Faces (JSF) é um *framework* Java para a implementação de interfaces *web* (ZHANG; CHENG, 2008). JavaServer Faces é uma tecnologia do lado servidor para desenvolvimento do lado cliente de aplicações Internet com a interface caracterizada como rica. JSF permite criar componentes de interface personalizados (JUNWU; JUNLING, 2010). JSF é definido de acordo com o modelo arquitetural MVC. E os principais componentes dessa tecnologia são (CZEKALSKA, 2008; ZHANG; CHENG, 2008):

a) Um conjunto de APIs (*Application Programming Interface*) – que são os componentes de interface e de gerenciamento de estados dos componentes, componentes manipuladores de eventos, de validação de entrada e de conversões de dados no lado servidor. Essas APIs também são utilizadas para definir aspectos de navegação da página e fornecem suporte para internacionalização e acessibilidade.

b) Uma biblioteca de *tags JavaServer Faces* (JSF) – as *tags* JSF são customizadas visando definir os componentes de interface gráfica em páginas *JavaServer Pages* (JSP) e para tratamento dos objetos no lado servidor. Essas *tags* definem a interface JSF em uma página JSP e realizam a conexão dos objetos no lado servidor.

Pitanga (2010) e JunWu e JunLing (2010) citam as seguintes vantagens no desenvolvimento de aplicações *web* utilizando JSF:

a) Interface – possibilita ao desenvolvedor criar interfaces por meio de um conjunto de componentes pré-definidos, personalizar componentes e criar os próprios componentes.

b) Componentes - fornece um conjunto de *tags* JSP para acessar os componentes. Provê reuso de componentes. A interface do usuário é composta a partir de um conjunto de padrões e componentes reusáveis do lado servidor. Os componentes podem ser customizados e desenvolvidos.

c) Gerenciamento de eventos - associa eventos do lado cliente com os respectivos manipuladores do lado servidor. Permite salvar o estado da informação de forma transparente e recarrega dados de formulários quando há atualização dos mesmos. Encapsula manipuladores de eventos e renderização de componentes, tornando possível utilizar os

componentes JSF padrão ou customizar componentes para fornecer suporte para outras linguagens de marcação além de HTML.

d) Funções - separação de funções que envolvem a construção de aplicações *web* pela aplicação do padrão de projetos MVC.

O ciclo de vida do *framework* JSF é apresentado na Figura 2. O fluxo de controle normal é mostrado por linhas sólidas, as linhas pontilhadas mostram os fluxos alternativos. Esses fluxos alternativos ocorrem dependendo se um componente requer que a página seja recarregada ou se um erro de validação ou conversão, por exemplo, ocorre.

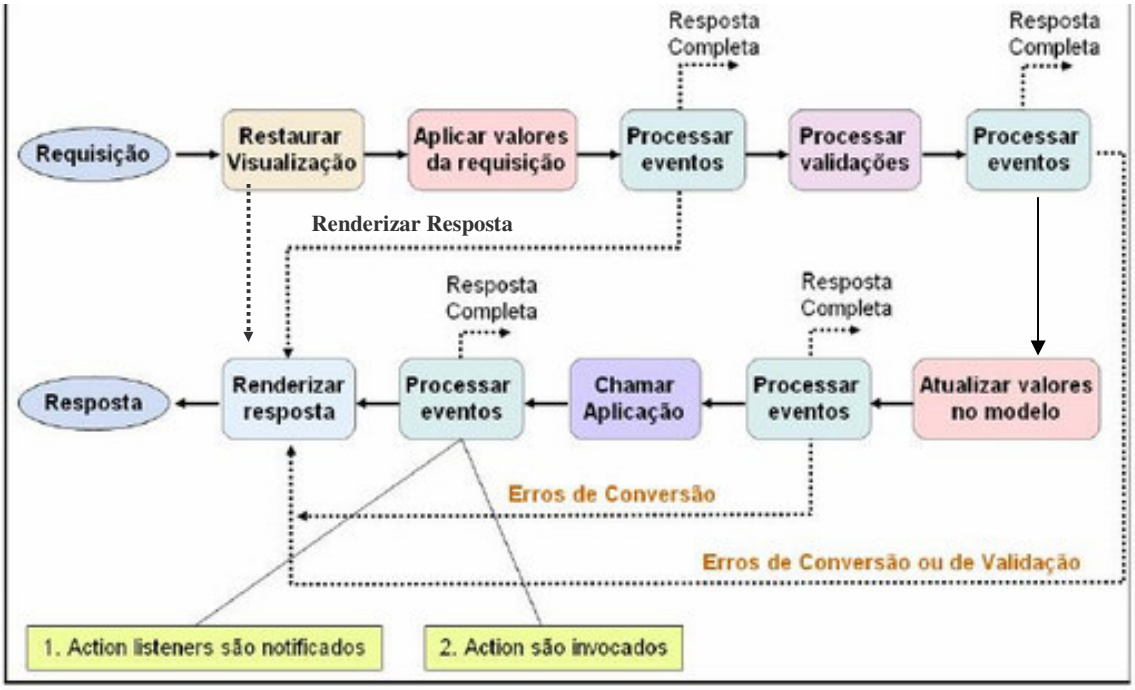

Figura 2 – Ciclo de vida do JSF
Fonte: adaptado de Raible (2006) *apud* Yoshiriro (2011, p.1).

As fases do ciclo de vida de uma aplicação utilizando JSF representadas na Figura 2 de acordo com Zhang e Cheng (2008) e JavaServer Faces (2014) são:

a) Resposta completa – refere-se a uma página *web* que não está na aplicação ou uma página da aplicação que não envolve JSF.

b) Renderizar resposta – a implementação do JSF autoriza o JSP contêiner a apresentar a página JSP. Os estados da árvore de componentes serão atualizados e é preparada a resposta para a requisição atual.

b.1) Se é a primeira requisição da página:

- Os componentes associados à página são criados e adicionados na *view*, em seguida a página é mostrada por meio do contêiner JSP.

b.2) Se é a primeira submissão da página:

- Os componentes são traduzidos para HTML e o contêiner JSP se encarrega de apresentá-la.

b.3) Se houver as *tags message* ou *messages* na página e ocorrem erros durante o ciclo de vida, as respectivas mensagens ou erros serão apresentados na página.

Em seguida os componentes são traduzidos para HTML e o *status* da *view* é salvo no servidor.

a) Restaurar visão – após receber uma requisição do cliente que pode ocorrer por um clique em um botão ou acesso a um *link*, por exemplo, o estado da *view* é restaurado por meio do objeto FacesContext (componentes, *listeners*, validadores, conversores, entre outros) e a implementação JSF restaurará a árvore de componentes. Alguns componentes como HtmlCommandButton gerarão eventos de ação ou de outro tipo. Caso seja a primeira submissão, uma *view* vazia é criada e o fluxo de sequência segue para a fase de renderização da resposta.

b) Aplicar valores de requisição – atualização do valor do componente de acordo os dados recebidos da requisição e podem ser utilizados conversores para essa operação. Erros de conversão, se houver, são adicionados na classe FacesContext e serão mostrados na fase de renderização da resposta. Os eventos gerados são lançados para os *listeners* de todos os componentes.

c) Processar validações – todos os validadores existentes na *view* são executados e todas as regras de validação são aplicadas a todos os valores informados. A validação pode incluir o uso de validadores externos. Se houver erros, uma mensagem é adicionada em FacesContext e o fluxo de execução será direcionado para a fase de renderização da resposta.

d) Atualizar valores do modelo – atualiza todos os valores dos *beans* ou objetos do modelo associados com o componente. O estado da *view* é atualizado com todos os valores gravados nos componentes e esse estado é mantido no lado servidor. Se houver erros de conversão de tipo, um erro é adicionado em FacesContext e o fluxo é direcionado para a fase de renderização da resposta.

e) Invocar aplicação – são chamadas todas as ações do *listener* registradas. O método de ação no *bean* é executado e as regras de navegação aplicadas. Após essa ação o fluxo de execução é encerrado.

3 MATERIAIS E MÉTODO

Este capítulo apresenta as ferramentas e as tecnologias utilizadas na modelagem e na implementação do sistema. Também é apresentada a sequência das atividades desenvolvidas para a realização do trabalho.

3.1 MATERIAIS

O Quadro 1 apresenta as ferramentas e as tecnologias utilizadas para modelar e implementar o sistema.

Ferramenta/ Tecnologia	**Versão**	**Referência**	**Finalidade**
Astah* Community	6.8.0 (model version: 33)	http://astah.net/editions/community	Documentação da modelagem baseada na UML.
Linguagem Java	JDK 1.8	http://www.oracle.com	Linguagem de programação.
IntelliJ IDEA	13.1.13	http://www.jetbrains.com/idea/	Ambiente de desenvolvimento.
MySQL	5.1.31	http://www.mysql.com/	Banco de dados.
MySQL Workbench	6.1 CE	http://www.mysql.com/products/workbench/	Administrador do banco de dados.
Glassfish	4.0	https://glassfish.java.net/	Servidor *web* para a aplicação.
PrimeFaces	5.0	http://primefaces.org/	Biblioteca de componentes para *web*.
JSF	2.2.7	https://javaserverfaces-spec-public.java.net/	*Framework* para desenvolvimento *web.*
Hibernate	4.3.5.Final	http://*www.hibernate.org/*	Efetuar o mapeamento objeto-relacional e a persistência dos dados.
Spring	3.1.1.release	http://spring.io/	*Framework* para persistência dos dados.
JasperReports	5.6.1	https://community.jaspersoft.com/project/jasperreports-library	*Framework* para geração de relatórios.
iReport	5.5.0	http://community.jaspersoft.com/project/ireport-designer	Ferramenta para desenvolver o leiaute de relatórios em modo gráfico.

Quadro 1 – Ferramentas e tecnologias a serem utilizadas

MySQL Workbench é uma ferramenta visual para *design*, desenvolvimento e administração de base de dados MySQL.

O GlassFish é um servidor de aplicação *open source*, para hospedar aplicações *web* que seguem o padrão Java EE (*Java Enterprise Edition*). É uma plataforma de desenvolvimento voltado para servidores para a linguagem Java que contém um conjunto de

tecnologias que agiliza o desenvolvimento, diminuindo a complexidade e o custo. É utilizado para controlar as transações e as persistências ao banco de dados, *clustering,* recursos com escopo de aplicação e a API RESTful para administração, entre outras.

O Hibernate é um *framework* utilizado para fazer o mapeamento objeto-relacional. O Hibernate facilita o mapeamento dos atributos entre uma base de dados relacional e o modelo objeto de uma aplicação. Esse mapeamento é feito por meio de anotações Java ou uso de arquivos *Extensible Markup Language* (XML). As tabelas do banco de dados são representadas por meio de classes na aplicação.

O Spring é um *framework open source* para Java que se baseia em padrões de projeto, inversão de controle e injeção de dependência. O contêiner é o responsável por instanciar as classes de uma aplicação Java e definir as dependências entre elas por meio de configurações efetuadas em um arquivo XML. Ele tem vários módulos com o Spring Data e Spring Security, por exemplo. O Spring Data é responsável por persistir os dados na base de dados. Já o Spring Security se encarrega da segurança da aplicação.

O PrimeFaces é um *framework open source* para o JSF (*Java Server Faces)* que permite criar interfaces ricas para aplicações *web* por meio de componentes para JSF. Seus componentes foram desenvolvidos para trabalhar por padrão com AJAX, ou seja, as chamadas assíncronas feitas ao servidor são executadas de forma padrão pelo próprio componente.

3.2 MÉTODO

Como método de desenvolvimento foi utilizado o processo unificado (KRUCHTEN, 2004) conhecido como iterativo e incremental pelas iterações que incrementam o produto final a cada ciclo de execução dos seus processos (fluxos de trabalho). O desenvolvimento se caracteriza como iterativo e incremental, embora com iterações informais, porque a partir da definição básica dos requisitos um cadastro para ser utilizado como padrão de desenvolvimento foi implementado. À medida que os requisitos foram identificados e documentados as funcionalidades eram implementadas.

a) Requisitos - a definição dos requisitos foi realizada a partir da descrição dos requisitos do ponto de vista do usuário realizada por professores do DAELE, estagiários responsáveis por esse ambiente, coordenador do curso e chefe de Departamento. Visitas foram realizadas nessa sala de apoio para entendimento do processo manual realizado no controle

dos empréstimos e de estoque. Em seguida, foram analisados os requisitos para o Laboratório do DAINF e essa análise foi realizada com a ajuda da professora orientadora deste trabalho e da professora responsável por esse ambiente. Esses requisitos compuseram um texto descritivo e por meio de listagens eles foram organizadas em funcionais e não funcionais. Os atores administrador, laboratorista e professor foram identificados.

b) Análise e projeto – na análise e projeto os requisitos foram modelados sob a forma de diagrama de casos de uso e de diagrama de entidades e relacionamentos do banco de dados. Revisões e complementos dos requisitos e sua respectiva modelagem ocorreram em diversos ciclos de implementação do sistema.

c) Desenvolvimento – inicialmente foram estudadas as tecnologias candidatas para identificação dos recursos que poderiam ser utilizados. Em seguida as funcionalidades de inclusão, exclusão, consulta e alteração de um cadastro básico foram implementadas. As demais funcionalidades foram implementadas visando gerar executáveis funcionais: em uma primeira versão foram implementados os cadastros, em uma próxima a reserva de ambientes e equipamentos e, em seguida, o empréstimo. Em uma última iteração foram implementados relatórios e requisitos complementares.

d) Testes – os testes foram informais e com o objetivo de identificar erros de codificação. Testes de usabilidade e de requisitos serão realizados por usuários dos respectivos ambientes para os quais o sistema está sendo implementado. Uma primeira versão do sistema (cadastros) foi instalada para que pudesse ser utilizada e avaliada por futuros usuários. As demais versões também passaram por esse tipo de teste, por um período de testes mais curto porque abrangiam funcionalidades específicas, embora de implementação complexa.

4 RESULTADOS

Este capítulo apresenta os resultados da realização deste trabalho. Os resultados estão centrados na modelagem e na implementação do sistema.

4.1 ESCOPO DO SISTEMA

O sistema possibilitará o agendamento de aulas práticas e o controle de estoques do laboratório de Hardware e Software do Departamento Acadêmico de Informática e da Sala de Apoio do Departamento Acadêmico de Elétrica. A reserva de ambientes (agendamento de aulas práticas) também inclui a reserva de materiais e equipamentos. As bases de dados serão distintas. Cada Departamento terá acesso a um aplicativo distinto.

O sistema possui como funcionalidades principais:

a) Cadastro de itens – itens são materiais, equipamentos e ambientes cadastrados no sistema.

b) Controle de estoque – ajuste de quantidade (aumento ou redução) de itens cadastrados.

c) Empréstimo – empréstimo de materiais e equipamentos para desenvolvimento de projetos por professores e alunos.

d) Agendamento de aulas práticas – reserva de ambientes, materiais e equipamentos para realização de aulas práticas.

Cadastro de itens - os itens são categorizados em ambientes (salas, laboratórios), equipamentos (multímetros, osciloscópios, estações de solda) e materiais (*leds*, sensores). Os ambientes são cadastrados individualmente e são utilizados para realização de aulas práticas e atividades como cursos e projetos. Os materiais podem retornar para o estoque ou serem utilizados definitivamente nos projetos. Os materiais são cadastrados por tipo e armazenam a respectiva quantidade. Os equipamentos são cadastrados por unidade. Cada equipamento possui um número de patrimônio que o individualiza. Esse número também é representado por um código de barras. Os equipamentos possuem controle de manutenções realizadas e se baixados do estoque deve ser explicitado o motivo da baixa. A baixa ocorre quando o equipamento deixa de ser utilizado e isso pode ocorrer por defeito ou substituição por equipamento mais novo, por exemplo.

Controle de estoques - inclui o acréscimo e a redução de quantidade de itens no estoque. Alterações de quantidade de materiais e equipamentos em estoque são realizadas pelos atores laboratorista e administrador. Essa atualização é realizada aumentando a quantidade em estoque quanto uma nova quantidade do produto é comprada ou diminuindo a quantidade quando os materiais são utilizados em aulas práticas, cedidos para projetos, danificados ou quando equipamentos são baixados do estoque, por exemplo.

No formulário de cadastro de itens é apresentado em um campo não editável, a quantidade existente em estoque e um campo editável para ser realizada a inclusão da quantidade a ser cadastrada. Essa quantidade informada é somada à quantidade existente em estoque. Essa soma pode ser negativa, no sentido de retirar uma quantidade em estoque, que foi usada em aula prática (essa é debitada automaticamente pelo sistema), cedida para projetos (essa é debitada a partir de informação do laboratorista), para atualização de estoque ou para zerar a quantidade de um item para excluí-lo do estoque, por exemplo.

A quantidade reservada de equipamentos e materiais a ser utilizada em aulas ou projetos é apenas "reservada" do estoque e não subtraída. Todos os usuários podem consultar as quantidades em estoque. A apresentação das quantidades é feita da seguinte forma: apresentada a quantidade real em estoque, a quantidade emprestada para a data atual e a quantidade disponível para a data atual.

Empréstimo de materiais – materiais e equipamentos podem ser emprestados para alunos e professores para utilização em projetos. O empréstimo ocorre por um determinado período e no final desse período os mesmos devem ser devolvidos ou deve ser justificada a sua não devolução. E nesse caso eles são baixados do estoque. Materiais podem ser utilizados definitivamente no projeto (componente soldado em placa) ou inutilizados (conectores danificados), por exemplo, e assim não serão devolvidos. Nesse caso, a respectiva quantidade será baixada do estoque e uma observação deve ser incluída com o motivo da baixa. O empréstimo de equipamentos é realizado por unidade. Exemplo: empréstimo do osciloscópio ABC123.

Agendamento de aulas práticas - reserva de ambientes, equipamentos e materiais para aulas práticas realizadas por professores. Na reserva de equipamentos e materiais é indicada a quantidade necessária, mas para o ambiente é indicado qual. Na reserva é indicado o horário para a reserva. Os horários são definidos de acordo com os horários das aulas da Universidade. Para equipamentos e materiais é indicada a quantidade de cada "tipo" e a reserva é realizada pela quantidade do item daquele tipo. Por exemplo: reserva de 10 osciloscópios para o dia 11/11/2014, das 18h40 às 21h10.

Para a realização da aula prática o professor faz uma solicitação indicando a data e o horário de realização da aula e os materiais e equipamentos necessários. O formulário de agendamento de aulas práticas terá um campo observação. Registros de observações podem ocorrer para equipamentos que não possuem a quantidade necessária para reserva. Assim, o laboratorista pode verificar a possibilidade de obter a quantidade solicitada.

O agendamento é "validado" pelo laboratorista ou administrador. Cada professor ao consultar as reservas que realizou verifica as atividades já realizadas e as futuras, bem como se elas foram validadas. Assim, será apresentado para o laboratorista e/ou administrador uma listagem das reservas futuras e que ainda não foram validadas.

Os professores cadastrados no sistema podem visualizar as quantidades em entoque por categoria e podem fazer solicitações de agendamento de aulas práticas. O professor pode editar as suas solicitações, mas alterações (salvando no mesmo registro) só podem ser realizadas se a aula ainda não ocorreu. Para alterar quantidades de itens reservados é realizada verificação se há no estoque disponibilidade para aquele dia. O professor pode editar um agendamento seu de uma data posterior a atual, significando que a aula prática já foi realizada, e alterar a data de realização fazendo um novo agendamento, incluir e excluir itens e alterar quantidades. Ao fazer isso é criada uma nova reserva (salvar como), a anterior permanece registrada no sistema. Se o professor editar uma aula prática em data posterior à atual será possível alterar a data. Isso no sentido de que não é possível alterar dados de uma reserva de ambiente de aula que já ocorreu, mas é possível realizar para datas futuras. Nesse caso, será realizada a consistência das quantidades reservadas com a existente em estoque.

As reservas permanecem no sistema, mas a quantidade "reservada" será considerada apenas da data atual e futura. As reservas de datas posteriores ao dia atual não serão consideradas no cálculo do estoque real (a quantidade que efetivamente está disponível para empréstimo). Esses equipamentos e materiais foram utilizados e liberados, ou seja, devolvidos.

O professor pode cancelar uma aula prática agendada, mas o registro permanece no banco de dados do sistema. O cancelamento ocorre, por exemplo, quando uma aula foi agendada, mas não foi realizada. Nesse caso, as quantidades são creditadas novamente no estoque. Por padrão, uma reserva deve ser cancelada por quem a realizou e uma observação deve ser colocada indicando o motivo da sua não realização.

4.2 MODELAGEM DO SISTEMA

Os usuários que possuem acesso ao sistema pertencem a três categorias: administrador, laboratorista, professor. Pode haver mais de um administrador cadastrado no sistema. Esse cadastro será realizado por um administrador pré-cadastrado no sistema que poderá alterar a sua senha e cadastrar outros usuários como administradores. Os professores e os laboratoristas fazem o seu próprio cadastro, escolhendo um desses tipos de usuários. Professores são os docentes que realizam aulas práticas em ambientes utilizando equipamentos e materiais, que desenvolvem projetos e orientam alunos em projetos. Laboratoristas são técnicos administrativos, professores, estagiários, bolsistas e outros que são responsáveis pelos laboratórios e salas de apoio. A seguir os usuários e suas funções específicas:

a) **Administrador**

- Manutenção de usuários.
- Funções do laboratorista.

b) **Laboratorista**

- Manutenção de cadastros.
- Controle de estoque.
- Visualização das solicitações de aula.
- Confirmação de reservas para aulas práticas.
- Realização de empréstimos e devoluções.
- Emissão de relatórios

c) **Professor**

- Visualização dos itens em estoque e respectiva quantidade disponível para reserva.
- Realização de solicitação de aula prática.
- Edição de solicitação de aula prática ainda não realizada.
- Exclusão de solicitação de aula prática ainda não realizada.
- Edição de solicitação de data passada para alterar a data, gerando nova solicitação.
- Inclusão de observações no sistema, como equipamentos com mau funcionamento, danificados, necessidade de aquisição de itens e etc.

A Figura 3 apresenta o diagrama de casos de uso do sistema considerando os usuários como os três atores do sistema.

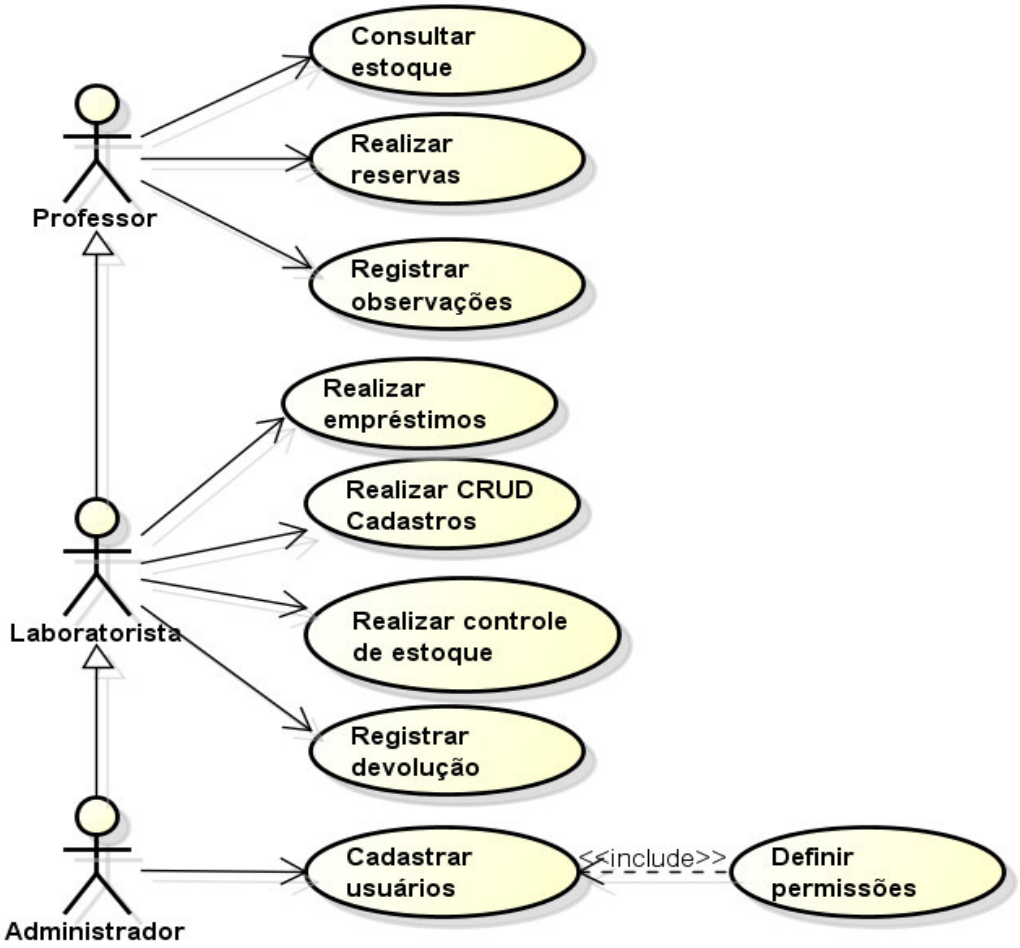

Figura 3 – Diagrama de casos de uso

O Quadro 2 apresenta os requisitos funcionais agrupados por casos de uso apresentados na Figura 3. Nesse quadro RF significa Requisito Funcional e CRUD significa *Create, Retrieve, Update* e *Delete* que são a operações básicas realizadas por meio de cadastros em tabelas de bancos de dados.

	Caso de uso	**Requisito**	**Descrição**
RF1	Manter usuários	Incluir administrador	Um usuário administrador pode incluir outros usuários com permissões de administrador.
RF2	Manter usuários	Excluir usuários	A exclusão de usuários cadastrados para os três tipos de atores existentes.
RF03	Realizar CRUD cadastros	Cadastro de equipamentos	Os equipamentos são individualizados por unidade. Eles possuem um número de patrimônio que está codificado por meio de um código de barras. Contudo, os equipamentos são totalizados por tipo para a realização de reserva para aulas práticas.

RF04	Realizar CRUD cadastros	Cadastro de materiais	Os materiais cadastrados. Os materiais são cadastrados por tipo, por exemplo, garra jacaré. Eles não são individualizados por item, mas controlados pela quantidade do tipo. No cadastro de materiais é registrada a quantidade mínima em estoque para cada item.
RF05	Realizar CRUD cadastros	Cadastro de fornecedores	Cadastro de fornecedores de materiais e equipamentos.
RF06	Realizar CRUD cadastros	Cadastro de ambientes	Laboratórios e salas nos quais são realizadas as aulas. Esses ambientes são reservados.
RF07	Realizar CRUD cadastros	Cadastro de tipos de observações	Cadastro de tipo de observações, por exemplo: defeito de equipamento, materiais para adquirir.
RF08	Realizar CRUD cadastros	Vincular fornecedores a materiais e equipamentos	Vincular os fornecedores a equipamentos e materiais.
RF09	Realizar reservas	Solicitações de aulas práticas	Os professores solicitam reservas de ambientes, materiais e equipamentos para aulas práticas. Na reserva é indicado quem está fazendo a solicitação. É apresentado como solicitante o usuário logado, mas o usuário pode alterar o solicitante. Isso é feito por uma caixa de combinação dos professores cadastrados. Se o solicitante, laboratorista, por exemplo, é quem está fazendo a solicitação, ele seleciona o nome do professor para quem está fazendo a reserva. Se o professor não está cadastrado, no campo observação da reserva é acrescentado o professor que efetivamente fará uso da reserva. Na reserva deve ser indicada a data e horário de início e fim. Esses horários são definidos de acordo com o horário das aulas. Uma reserva pode incluir somente ambiente, equipamentos e materiais ou combinações desses itens.
RF10	Realizar reservas	Reservar equipamentos	Os equipamentos vinculados a uma solicitação de aula prática. Os equipamentos são individualizados no cadastro. Cada equipamento possui um cadastro, mas no momento da reserva é apresentada apenas a quantidade de equipamentos por tipo.
RF11	Realizar reservas	Reservar ambientes	Os ambientes são individualizados e,

			portanto, a reserva é realizada para cada ambiente.
RF12	Realizar reservas	Reservar materiais	Os materiais são reservados por quantidade. E essa quantidade pode ser baixada do estoque se materiais não retornam do empréstimo.
RF13	Realizar controle de estoque	Movimentação de equipamentos	Controle de estoque dos equipamentos. Esse cadastro possui um campo de observação para indicar o ocorrido pela movimentação estar sendo registrada. Uma movimentação é de quantidade e pode ocorrer pelo aumento ou redução de quantidade. A redução pode ocorrer quando o equipamento é encaminhado para manutenção. No retorno, se em condições de uso, a quantidade é incrementada.
RF14	Realizar controle de estoque	Movimentação de materiais	Controle de estoque dos materiais. Uma movimentação é de quantidade e pode ocorrer pelo aumento ou redução de quantidade. Deve ser registrada a quantidade mínima desejável de cada item material. Sendo, assim, possível emitir uma listagem dos itens que estão em quantidades mínimas em estoque.
RF15	Registrar devolução	Realizar baixa materiais e equipamentos	Após a aula realizada os materiais e equipamentos são automaticamente devolvidos para o estoque. Isso é realizado por rotina automática do sistema que é executada, por exemplo, 30 minutos após o horário final da reserva. Contudo, os materiais podem não retornar e serem baixados definitivamente do estoque.
RF16	Registrar devolução	Histórico de materiais	Registro de materiais consumidos (usados em projetos e não devolvidos). O registro é feito por tipo de material (exemplo: *led* vermelho, capacitor 10 Ohm)
RF17	Registrar devolução	Histórico de equipamentos	Os registros de manutenção de equipamentos devem ser realizados no sistema. Os equipamentos podem ser baixados do estoque por inutilização, por exemplo. O histórico é realizado por equipamento.
RF18	Registrar observações	Defeitos em equipamentos	Registro de defeitos em equipamentos. É armazenado em um histórico do equipamento com o registro de defeitos e manutenções realizadas. Deve ser acrescentado o custo da manutenção

			realizada, quem realizou e a data.
RF19	Registrar observações	Solicitação de materiais	Professor pode indicar materiais para serem adquiridos.

Quadro 2 – Requisitos funcionais

No Quadro 3 estão listados os requisitos não funcionais. Nesse quadro RNF significa Requisito Não Funcional.

	Requisito	**Descrição**
RNF01	Controle de estoque	A quantidade de materiais utilizada em projetos e que não será devolvida deve ser debitada do estoque.
RNF02	Acesso ao sistema	Administrador tem acesso a todas as funcionalidades do sistema, exceto alterar reservas realizadas por professores e exclusivamente à manutenção (exclusão) de usuários. Professor faz solicitações de aulas práticas, visualiza e edita as suas solicitações, visualiza quantidade de itens em estoque. Laboratorista visualiza as solicitações de todos os professores, faz o controle de estoque e cadastro de itens. E faz a liberação da aula prática solicitada pelos professores.
RNF03	Equipamentos e materiais utilizados nas aulas	A quantidade de materiais e equipamentos é logicamente decrementada no estoque na data da reserva e incrementada após a data de realização da aula. Essas atualizações são realizadas automaticamente pelo sistema.
RNF04	Equipamentos e materiais utilizados nas aulas	Essa quantidade é fisicamente incrementada no estoque quando novos equipamentos e materiais de tipos já cadastrados são adquiridos. E é fisicamente decrementada quando equipamentos e materiais se tornam impróprios para uso: defeito, quebra, obsolescência, não retorno. Essas atualizações são realizadas pelo laboratorista.
RNF05	Solicitação de aulas práticas	Laboratorista visualiza as solicitações de aula ainda não realizadas e confirma ou indica o ambiente (laboratório) no qual a aula será realizada.
RNF06	Solicitação de aulas práticas	Professor faz a solicitação e pode indicar um laboratório para ser utilizado. A solicitação é confirmada pelo laboratorista. Deve ser apresentada uma listagem das solicitações (reservas) indicando as pendentes e as confirmadas.
RNF07	Solicitação de aulas práticas	O professor pode fazer a solicitação de quantidade não disponível para aquele dia. Na análise da reserva pelo laboratorista poderá haver ajuste de laboratórios e assim ser possível atender a solicitação.
RNF08	Solicitação de aulas práticas	O professor, ao logar-se, visualiza as suas reservas e se elas foram confirmadas.
RNF09	Permissões administrador	O usuário administrador tem acesso a todas as

		funcionalidades do sistema, mas não pode fazer alterações em reservas feitas por professores. A reserva só pode ser alterada ou excluída pelo usuário que a realizou.

Quadro 3 – Requisitos não funcionais

A Figura 4 apresenta o diagrama de entidades e relacionamentos do banco de dados.

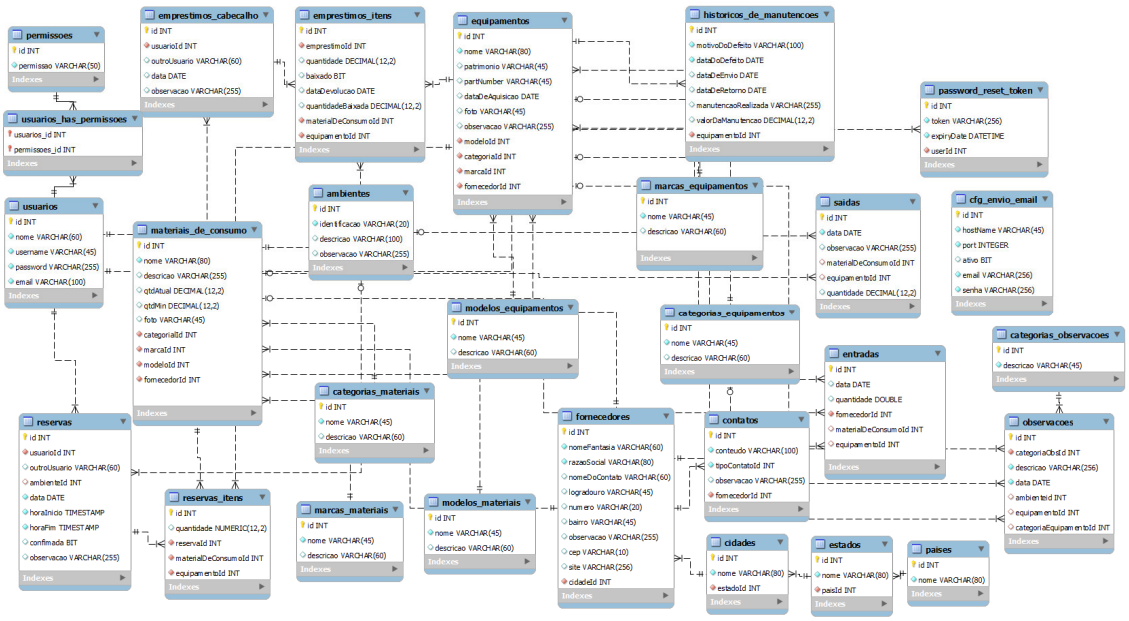

Figura 4 – Diagrama de entidades e relacionamentos

Os quadros a seguir apresentam a descrição das tabelas constantes na Figura 4.

O Quadro 4 apresenta a listagem dos campos da tabela de usuários.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Observações
id	Numérico	Não	Sim	Não	
nome	Texto	Não	Não	Não	
username	Texto	Não	Não	Não	
password	Texto	Não	Não	Não	
email	Texto	Não	Não	Não	

Quadro 4 – Tabela usuarios

No Quadro 5 está a descrição dos campos da tabela permissões.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Observações
id	Numérico	Não	Sim	Não	
permissao	Texto	Não	Não	Não	

Quadro 5 – Tabela permissoes

A o registro das permissões atribuídas aos usuários é realizado pela tabela que vincula usuários e permissões. O Quadro 6 lista os campos dessa tabela.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Observações
id	Numérico	Não	Sim	Não	
permissao	Texto	Não	Não	Não	

Quadro 6 – Tabela usuarios_has_permissoes

O Quadro 7 apresenta a listagem dos campos da tabela de ambientes.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Observações
id	Numérico	Não	Sim	Não	
identificacao	Texto	Não	Não	Não	
descricao	Texto	Sim	Não	Não	
observacao	Texto	Sim	Não	Não	

Quadro 7 – Tabela ambientes

O Quadro 8 apresenta a listagem dos campos da tabela de materiais de consumo.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Observações
id	Numérico	Não	Sim	Não	
nome	Texto	Não	Não	Não	
descricao	Texto	Sim	Não	Não	
qtdAtual	Numérico	Sim	Não	Não	
qtdeMn	Numérico	Sim	Não	Não	Quantidade mínima de estoque
categoriaId	Numérico	Sim	Não	Não	Da tabela Categorias
marcaId	Numérico	Sim	Não	Não	Da tabela Marcas
modeloId	Numérico	Sim	Não	Não	Da tabela Modelos
fornecedorId	Numérico	Sim	Não	Não	Da tabela Fornecedores

Quadro 8 – Tabela materiais_de_consumo

O Quadro 9 apresenta a listagem dos campos da tabela de equipamentos.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Observações
id	Numérico	Não	Sim	Não	
nome	Texto	Não	Não	Não	
patrimonio	Texto	Sim	Não	Não	
partNumber	Texto	Sim	Não	Não	
dataAquisicao	Data	Sim	Não	Não	

foto	Texto	Sim	Não	Não	Imagem do equipamento
observacao	Texto	Sim	Não	Não	
modeloId	Numérico	Sim	Não	Não	Da tabela Modelos
categoriaId	Numérico	Sim	Não	Não	Da tabela Categorias
marcaId	Numérico	Sim	Não	Não	Da tabela Marcas
fornecedorId	Numérico	Sim	Não	Não	Da tabela Fornecedor

Quadro 9 – Tabela de equipamentos

O Quadro 10 apresenta a listagem dos campos da tabela de histórico de manutenção de equipamentos.

Campo	**Tipo**	**Nulo**	**Chave primária**	**Chave estrangeira**	**Observações**
id	Numérico	Não	Sim	Não	
equipamementoId	Numérico	Sim	Não	Não	Da tabela Equipamentos
motivoDefeito	Texto	Não	Não	Não	Descrição do defeito
dataDoDefeito	Data	Sim	Não	Não	
dataDeEnvio	Data	Sim	Não	Não	Envio para conserto
dataDeRetorno	Data	Sim	Não	Não	Retorno do conserto
manutencaoRealizada	Texto	Sim	Não	Não	
valorManutencao	Numérico	Sim	Não	Não	

Quadro 10 – Tabela hsitorico_de_manutencoes

O Quadro 11 apresenta a listagem dos campos da tabela categorias que armazena as categorias de materiais. Esses tipos são utilizados para agrupar materiais para indicação de quantidade no empréstimo.

Campo	**Tipo**	**Nulo**	**Chave primária**	**Chave estrangeira**	**Observações**
id	Numérico	Não	Sim	Não	
nome	Texto	Não	Não	Não	
descrição	Texto	Sim	Não	Não	

Quadro 11 – Tabela de categorias_materiais

O Quadro 12 apresenta a listagem dos campos da tabela de modelos para materiais.

Campo	**Tipo**	**Nulo**	**Chave primária**	**Chave estrangeira**	**Observações**
id	Numérico	Não	Sim	Não	
nome	Texto	Não	Não	Não	
descrição	Texto	Sim	Não	Não	

Quadro 12 – Tabela de modelos_materiais

O Quadro 13 apresenta a listagem dos campos da tabela de marcas para materiais.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Observações
id	Numérico	Não	Sim	Não	
nome	Texto	Não	Não	Não	
descrição	Texto	Sim	Não	Não	

Quadro 13 – Tabela de marcas_materiais

O Quadro 14 apresenta a listagem dos campos da tabela de fornecedores para equipamentos e materiais.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Observações
id	Numérico	Não	Sim	Não	
nomeFantasia	Texto	Não	Não	Não	
razaoSocial	Texto	Sim	Não	Não	
nomeDoContato	Texto	Sim	Não	Não	Pessoa de contato na empresa
logradouro	Texto	Sim	Não	Não	Rua, avenida
numero	Texto	Sim	Não	Não	
bairro	Texto	Sim	Não	Não	
cidade	Texto	Sim	Não	Sim	Da tabela cidades
cep	Texto	Sim	Não	Não	
site	Texto	Sim	Não	Não	
observacao	Texto	Sim	Não	Não	

Quadro 14 – Tabela de fornecedores

O Quadro 15 apresenta a listagem dos campos da tabela de cidades.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Observações
id	Numérico	Não	Sim	Não	
nome	Texto	Não	Não	Não	

Quadro 15 – Tabela de cidades

O Quadro 16 apresenta a listagem dos campos da tabela de estados.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Observações
id	Numérico	Não	Sim	Não	
nome	Texto	Não	Não	Não	
sigla	Texto	Sim	Não	Não	

Quadro 16 – Tabela de estados

O Quadro 17 apresenta a listagem dos campos da tabela de contatos. Um fornecedor pode ter vários contatos.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Observações
id	Numérico	Não	Sim	Não	
conteúdo	Texto	Não	Não	Não	Descrição do contato, que pode ser um nome
tipoContato	Numérico	Não	Não	Não	De uma lista: pessoa, *email*, telefone, página *web*, fax, outro
fornecedorId	Numérico	Não	Não	Sim	Da tabela Fornecedores
observacao	Texto	Sim	Não	Não	

Quadro 17 – Tabela de contatos

O Quadro 18 apresenta a listagem dos campos da tabela de empréstimos realizados para professores e alunos.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Observações
id	Numérico	Não	Sim	Não	
usuarioId	Numérico	Não	Não	Sim	O usuário logado no sistema
oturoUsuario	Texto	Sim	Não	Sim	Para quem o empréstimo é realizado, pode ser informado o nome, login ou ra
data	Data	Sim	Não	Não	
observacao	Texto	Sim	Não	Não	

Quadro 18 – Tabela cabeçalho de emprestimos_cabecalho

O Quadro 19 apresenta a listagem dos campos da tabela de empréstimos realizados para professores e alunos. O empréstimo realizado é de equipamentos e materiais. Os equipamentos e materiais são listados a partir das respectivas tabelas permitindo aos usuários selecionar.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Observações
id	Numérico	Não	Sim	Não	
emprestimoId	Numérico	Não	Não	Sim	Da tabela empréstimos cabeçalho
quantidade	Numérico	Não	Não	Não	Por padrão contém o valor 1
equipamentoId	Numérico	Sim	Não	Sim	Da tabela Equipamentos
materialDeConsumoId	Numérico	Sim	Não	Sim	Da tabela materiais.
dataDevolucao	Data	Não	Não	Não	

quantidadeBaixada	numérico	Sim	Não	Não	
baixado	numérico	Sim	Não	Não	indica se a baixa foi realizada ou não

Quadro 19 – Tabela emprestimos-itens

O Quadro 20 apresenta a listagem dos campos da tabela de reservas. Essa tabela é usada para agendar as reservas realizadas por professores Uma reserva pode envolver um ambiente, materiais e equipamentos e é utilizada, por exemplo, para o professor agendar uma aula prática reservando ambiente, materiais e equipamentos. O campo "confirmada" tem por padrão o valor "não" e é editável somente pelos usuários administrador e laboratorista. A alteração de estado é realizada por usuário administrador ou laboratorista. Os detalhes da reserva são realizados pelas tabelas empréstimos e cabeçalho do empréstimo.

Campo	**Tipo**	**Nulo**	**Chave primária**	**Chave estrangeira**	**Observações**
id	Numérico	Não	Sim	Não	
usuarioId	Numérico	Não	Não	Sim	Usuário logado
outroUsuario	Texto	Sim	Não	Não	
ambienteId	Numérico	Sim	Não	Não	Da tabela ambientes
data	Data	Sim	Não	Não	
horarioInicio	Hora	Sim	Não	Não	
horarioFim	Hora	Sim	Não	Não	
observacao	Texto	Sim	Não	Não	
confirmada	Lógico	Não	Não	Não	Por padrão não. Só é alterada pelo laboratorista ou administrador.

Quadro 20 – Tabela reservas

O Quadro 21 apresenta a listagem dos campos da tabela tipos de observações. Problemas com ambientes, equipamentos, falta de material, sugestão de material para compra.

Campo	**Tipo**	**Nulo**	**Chave primária**	**Chave estrangeira**	**Observações**
id	Numérico	Não	Sim	Não	
descricao	Texto	Não	Não	Não	

Quadro 21 – Tabela tipos de observacoes

O Quadro 22 apresenta a listagem dos campos da tabela para registro de observações. Uma observação pode estar relacionada a defeitos com equipamentos, necessidade observada de aquisição de materiais e etc.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Observações
id	Numérico	Não	Sim	Não	
observacaoId	Numérico	Não	Não	Sim	Da tabela tipos de observação
ambienteId	Sim	Não	Não	Sim	Da tabela ambientes
tipoEquipamentoId	Sim	Não	Não	Sim	Da tabela tipos de equipamentos
equipamentoId	Sim	Não	Não	Sim	Da tabela equipamentos
descricao	Texto	Não	Não	Não	Descrição da observação.
data	Data	Não	Não	Não	Data do sistema

Quadro 22 – Tabela de categorias_observacoes

O Quadro 23 apresenta a listagem dos campos da tabela de materiais.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Observações
id	Numérico	Não	Sim	Não	
Nome	Texto	Não	Não	Não	
Descrição	Texto	Não	Não	Não	
materialDeConsumo					
modeloId	Numérico	Sim	Não	Sim	Da tabela Modelos
marcaId	Numérico	Sim	Não	Sim	Da tabela Marcas
equipamentoId	Numérico	Sim	Não	Sim	Da tabela Equipamentos
forncedorId	Numérico	Sim	Não	Sim	Da tabela Fornecedores
foto	Texto	Sim	Não	Não	

Quadro 23 – Tabela de materiais

O Quadro 24 apresenta a listagem dos campos da tabela categorias que armazena as categorias de equipamentos. Esses tipos são utilizados para agrupar equipamentos para indicação de quantidade no empréstimo.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Observações
id	Numérico	Não	Sim	Não	
nome	Texto	Não	Não	Não	
descrição	Texto	Sim	Não	Não	

Quadro 24 – Tabela de categorias_equipamentos

O Quadro 25 apresenta a listagem dos campos da tabela de modelos para equipamentos.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Observações
id	Numérico	Não	Sim	Não	
nome	Texto	Não	Não	Não	
descrição	Texto	Sim	Não	Não	

Quadro 25 – Tabela de modelos_equipamentos

O Quadro 26 apresenta a listagem dos campos da tabela de marcas para equipamentos.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Observações
id	Numérico	Não	Sim	Não	
nome	Texto	Não	Não	Não	
descrição	Texto	Sim	Não	Não	

Quadro 26 – Tabela de marcas_equipamentos

O Quadro 27 apresenta a listagem dos campos da tabela de itens de reserva.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Observações
id	Numérico	Não	Sim	Não	
Quantidade	Numérico	Não	Não	Não	
reservaId	Numérico	Sim	Não	Sim	Da tabela reservas
materialDeConsumoId	Numérico	Sim	Não	Sim	Da tabela materiais
equipamentoId	Numérico	Sim	Não	Sim	Da tabela equipamentos

Quadro 27 – Tabela de reservas_itens

O Quadro 27 apresenta a listagem dos campos da tabela de entradas de materiais de consumo e equipamentos.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Observações
id	Numérico	Não	Sim	Não	
data	Data	Não	Não	Não	
Quantidade	Numérico	Sim	Não	Não	
fornecedorId	Numérico	Sim	Não	Sim	Da tabela fornecedores
materialDeConsumoId	Numérico	Sim	Não	Sim	Da tabela materiais
equipamentoId	Numérico	Sim	Não	Sim	Da tabela equipamentos

Quadro 28 – Tabela de entradas

O Quadro 28 apresenta a listagem dos campos da tabela de saídas (baixas) de materiais de consumo e equipamentos.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Observações
id	Numérico	Não	Sim	Não	
data	Data	Não	Não	Não	
quantidade	Numérico	Sim	Não	Não	
observacao	Texto	Sim	Não	Sim	
materialDeConsumoId	Numérico	Sim	Não	Sim	Da tabela materiais
equipamentoId	Numérico	Sim	Não	Sim	Da tabela equipamentos

Quadro 29 – Tabela de saidas

O Quadro 29 contém os dados para configuração para envio de email pelo sistema.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Observações
id	Numérico	Não	Sim	Não	
hostName	Texto	Não	Não	Não	
port	Numérico	Sim	Não	Não	
ativo	Numérico	Sim	Não	Sim	
email	Texto	Sim	Não	Sim	
senha	Texto	Sim	Não	Sim	

Quadro 30 – Tabela de cfg_envio_email

O Quadro 30 apresenta a listagem dos campos da tabela password_reset_token.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Observações
id	Numérico	Não	Sim	Não	
Token	Texto	Não	Não	Não	
expireDate	Data	Sim	Não	Não	
userId	Numérico	Sim	Não	Sim	Da tabela usuários

Quadro 31 – Tabela password_reset_token

O Quadro 32 apresenta a listagem dos campos da tabela de países.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Observações
id	Numérico	Não	Sim	Não	
nome	Texto	Não	Não	Não	
sigla	texto				

Quadro 32 – Tabela de paises

4.3 APRESENTAÇÃO DO SISTEMA

O leiaute do sistema é composto por três setores. No setor superior é apresentada a logo da UTFPR. No centro, é apresentada uma barra de menu, com agrupamentos de funcionalidades. E no setor inferior o conteúdo da página que está sendo visitada.

A Figura 5 apresenta a página principal. O conteúdo dessa página é um calendário com as reservas do mês atual. Na imagem é apresentado o menu de cadastros expandido. Cada opção do cadastro possui as opções de submenu: cadastro e pesquisa. A reserva é uma das atividades mais importantes realizadas na sala de Apoio do DAELE. Assim, optou-se por manter na tela principal a visualização do calendário para reservas.

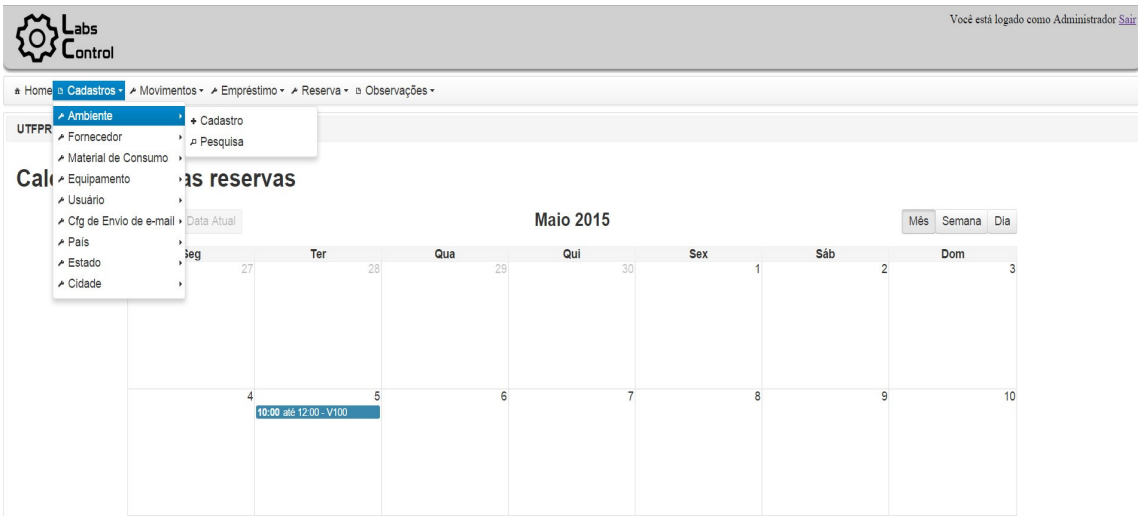

Figura 5 – Tela principal

A Figura 6 mostra o conteúdo da página de listagem dos Ambientes. Nessa página o usuário pode visualizar os ambientes já cadastrados e também pode realizar as operações básicas de inclusão, exclusão e alteração.

Para a operação de alteração de um registro é necessário clicar no botão que contém a imagem de um lápis, para a de exclusão no botão que contém a imagem de um lixeiro e para inserir um ambiente deve-se clicar no botão Novo. Esses ícones estão na área circulada na Figura 6.

Figura 6 – Acesso para edição de registro

Ainda é possível fazer uma visualização de todos os dados do ambiente, para isso é necessário clicar no botão que representa e uma lupa (Figura 7).

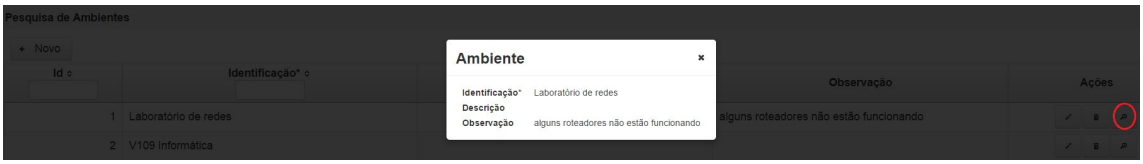

Figura 7 – Visualização de dados de cadastro

Ao clicar no botão "+ Novo", o sistema redirecionará o usuário para uma página com um formulário que contém os campos necessários para efetuar a inclusão de um novo ambiente, juntamente com os botões para salvar e realizar pesquisa, como mostra a Figura 8.

Figura 8 – Tela para inclusão de ambientes

Caso o usuário clique no botão "Salvar" sem informar o campo obrigatório, identificado por um asterisco (*), o sistema fará a validação e exibirá uma mensagem de erro para que o usuário possa corrigir e salvar, como mostra a Figura 9. Essa mensagem (e as outras relacionadas aos formulários, é exibida por meio da *tag "<p:messages />"* que é disponibilizada pelo PrimeFaces.

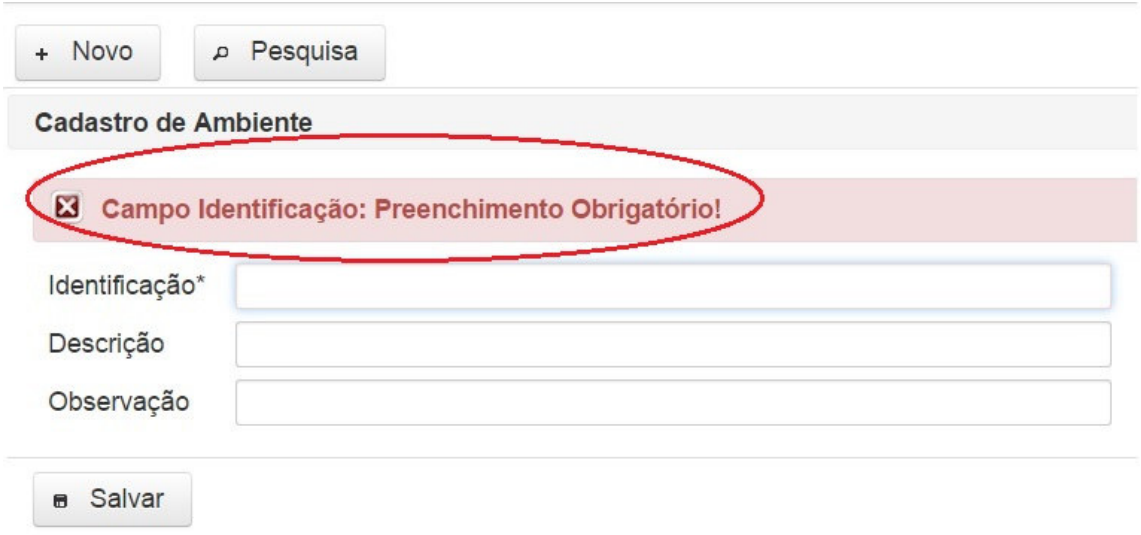

Figura 9 – Informação de campos de preenchimento obrigatório

Se o usuário clicar no botão "Salvar" sem informar o campo obrigatório, identificado por um asterisco (*), o sistema fará a validação e exibirá uma mensagem de erro para que o usuário possa corrigir e salvar, como mostra a Figura 9.

Essa validação ocorre por meio da utilização da propriedade *required,* disponibilizada pelo componente "inputText" do PrimeFaces. Caso o valor dessa propriedade seja *true,* ao submeter o formulário o próprio componente se encarrega de verificar se existe informação no "inputText". E caso não exista é exibida uma mensagem que é atribuída na propriedade *requiredMessage* e também é disponibilizada pelo componente.

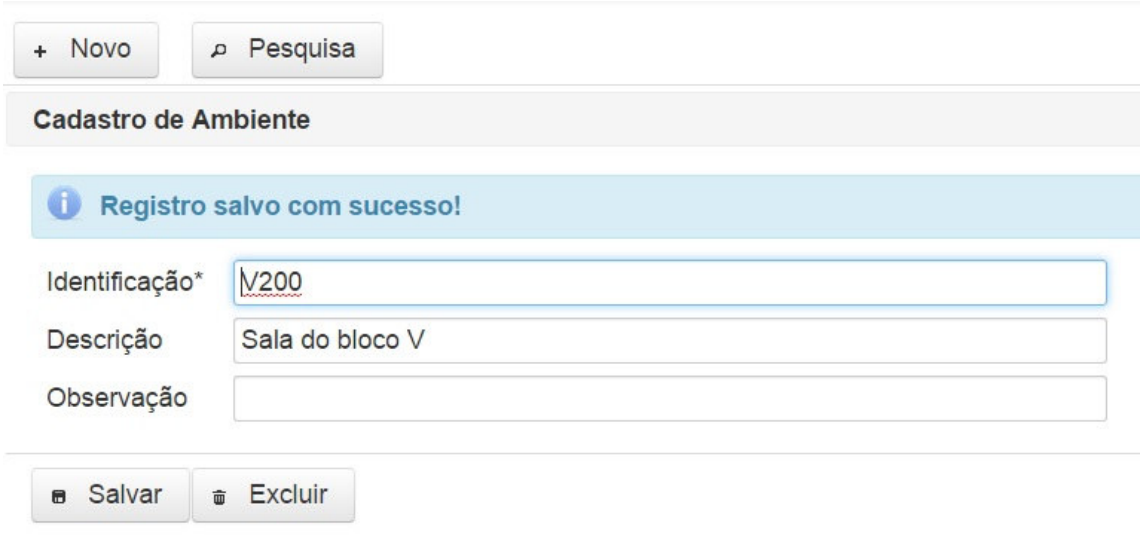

Figura 10 – Informação de inclusão com sucesso

A operação de alteração é semelhante à de inclusão, sendo que, para alterar um registro o usuário deve clicar no botão que contém uma imagem de um lápis referente ao ambiente desejado. Sendo assim, o sistema redirecionará o usuário para uma página de alteração dos dados, exibindo um formulário com os dados já preenchidos do ambiente escolhido pelo usuário para ser alterado (Figura 11). O processo de validação é o mesmo de inclusão.

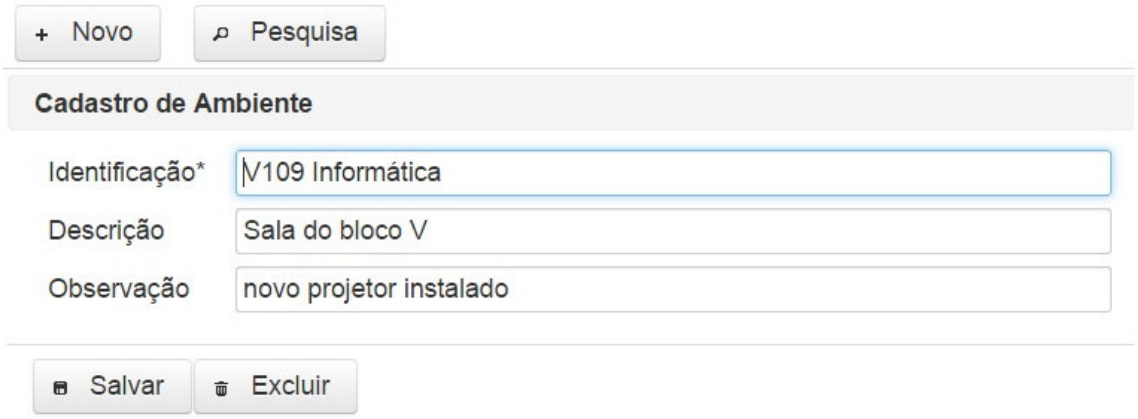

Figura 11 – Operação de alteração de dados

Para realizar a operação de exclusão, o usuário deve clicar no botão que contém uma imagem de um lixeiro do registro correspondente ao ambiente desejado, quando estiver na página de pesquisa de ambientes ou no botão de Excluir caso esteja na página de alteração. O sistema exibirá uma mensagem de confirmação para que a exclusão seja efetivada (Figura 12).

Figura 12 – Exclusão de registro

Caso o usuário confirme a exclusão, o sistema removerá o registro do banco de dados (no exemplo da Figura 12, um ambiente cadastrado), atualizará a tabela de listagem e apresentará uma mensagem de sucesso. Isso se o usuário estiver na página de pesquisa. Se o usuário estiver na tela de alteração, a mensagem de sucesso também será apresentada, porém o sistema permanece na página de alteração com os campos sem preenchimento e o botão de Excluir é ocultado como mostra a Figura 13.

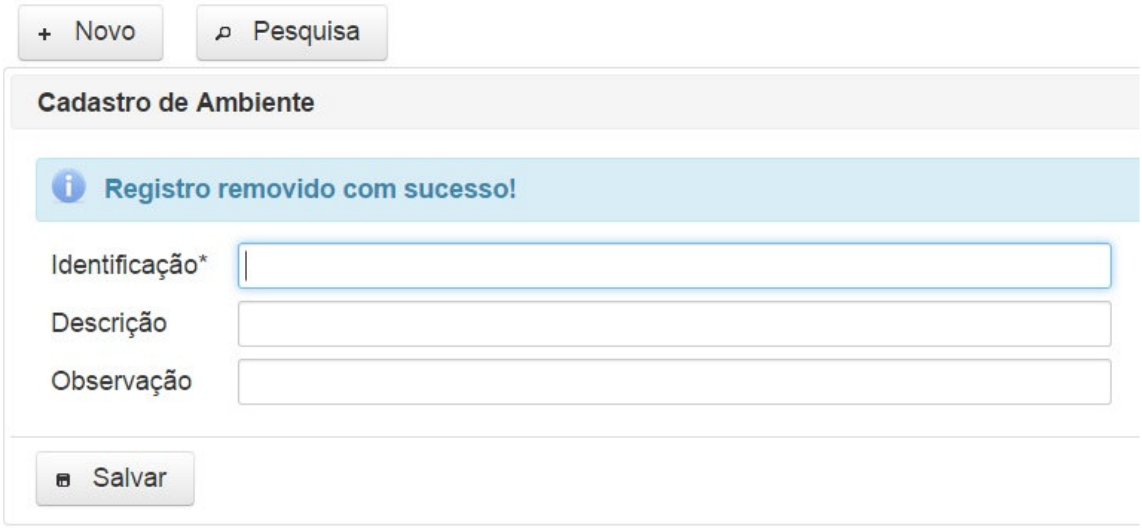

Figura 13 – Remoção de registro via página de alteração

A mensagem é apresentada na parte superior da tela como mostra a Figura 14. Essa é localização padrão das mensagens apresentadas para operações de inclusão de dados.

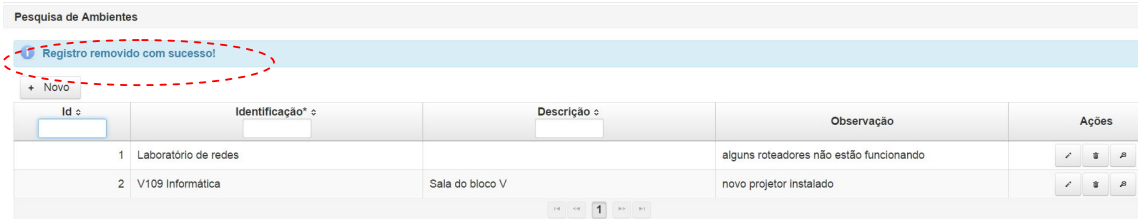

Figura 14 – Remoção de registro

4.4 IMPLEMENTAÇÃO DO SISTEMA

Na Listagem 1 é possível verificar a forma como é desenvolvido o conteúdo da página de pesquisa de ambientes.

```
15        <ui:define name="metadata">
16            <f:event type="preRenderView"
listener="#{ambienteController.find}"/>
17        </ui:define>
18
19        <ui:define name="conteudo">
20            <p><p:commandButton value="#{form.novo}"
action="/pages/cadastros/ambiente/ambienteForm.xhtml?faces-redirect=true"
21                                immediate="true" ajax="false"
style="float: right"
22                                icon="ui-icon-plus"/></p>
23
24            <p:dataTable value="#{ambienteController.lsEntity}"
var="objeto" emptyMessage="#{form.nenhumRegistroEncontrado}"
25                         paginator="true" rows="10"
paginatorPosition="bottom">
```

```
26                <p:column headerText="#{form.id}" sortBy="#{objeto.id}"
filterBy="#{objeto.id}" styleClass="columnId">
27                    <h:outputText value="#{objeto.id}" />
28                </p:column>
29                <p:column headerText="#{form.descricao}"
sortBy="#{objeto.descricao}" filterBy="#{objeto.descricao}"
filterMatchMode="contains">
30                    <h:outputText value="#{objeto.descricao}" />
31                </p:column>
32                <p:column headerText="#{form.identificacao}"
sortBy="#{objeto.identificacao}" filterBy="#{objeto.identificacao}"
filterMatchMode="contains">
33                    <h:outputText value="#{objeto.identificacao}" />
34                </p:column>
35                <p:column headerText="#{form.obs}">
36                    <h:outputText value="#{objeto.observacao}" />
37                </p:column>
38                <p:column headerText="#{form.acoes}"
styleClass="columnAcoes">
39                    <labs:commandButtonSearch
controller="#{ambienteController}"/>
40                </p:column>
41            </p:dataTable>
42        </ui:define>
```

Listagem 1 – Conteúdo da página de pesquisa de ambientes

Como pode ser visualizado na Listagem 1, é utilizada uma *tag* do *Java Server Faces* (JSF) para que ao carregar a página e antes de renderizá-la o objeto "ambienteController" seja carregado para que possam ser exibidos os dados na tabela gerada, que pode ser verificada pela *tag* "p:dataTable". Esse é um componente do PrimeFaces para facilitar a montagem da tabela dos dados a serem exibidos.

Desse modo é possível verificar a existência de um botão que serve para incluir um ambiente, que ocorre pela *tag* do PrimeFaces "p:commandButton". Ao ser clicado nesse botão, o sistema redireciona o usuário para a página de cadastro. O código desenvolvido da página de cadastro é apresentado na Listagem 2.

```
11        <ui:define name="titulo">
12            <h:outputText value="#{form.cadastroDeAmbiente}"/>
13        </ui:define>
14
15        <ui:define name="conteudo">
16            <p:panelGrid columns="2">
17                <h:outputLabel value="#{form.identificacao}*" />
18                <p:inputText
value="#{ambienteController.entity.identificacao}" size="20"
maxlength="20" required="true"
requiredMessage="#{msg.identificacaoRequired}"/>
19                <h:outputLabel value="#{form.descricao}" />
20                <p:inputText
value="#{ambienteController.entity.descricao}" size="60" maxlength="100"/>
21                <h:outputLabel value="#{form.obs}" />
22                <p:inputText
value="#{ambienteController.entity.observacao}" size="100"
maxlength="255"/>
```

```
23            </p:panelGrid>
24        </ui:define>
25
26        <ui:define name="rodape">
27            <labs:commandButtonForm update="@form"
controller="#{ambienteController}"/>
28
29            <p:commandButton value="#{form.voltar}"
action="/pages/cadastros/ambiente/ambienteSearch.xhtml?faces-
redirect=true" icon="ui-icon ui-icon-arrowthick-1-w"
30                             immediate="true" ajax="false" style="float:
right"/>
31            <div style="clear: both"></div>
32        </ui:define>
```

Listagem 2 – Conteúdo da página de cadastro de ambientes

Essa página de cadastro do ambiente é chamada de “ambienteForm.xhtml” e utiliza as *tags* “define” do Facelet. Para utilizá-la deve-se definir no atributo *name* o mesmo valor do atributo *name* da *tag insert* definidos na página padrão, como pode ser visto na Listagem 3.

```
1     <?xml version="1.0" encoding="UTF-8"?>
2     <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
3             "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
4     <html xmlns="http://www.w3.org/1999/xhtml"
5           xmlns:h="http://xmlns.jcp.org/jsf/html"
6           xmlns:ui="http://xmlns.jcp.org/jsf/facelets"
7           xmlns:f="http://xmlns.jcp.org/jsf/core"
8           xmlns:p="http://primefaces.org/ui">
9
10    <h:head>
11        <title>
12            <ui:insert name="titulo">
13                <h:outputText value="LabsControl - UTFPR/PB"/>
14            </ui:insert>
15        </title>
```

Listagem 3 – Tags define do Facelets

Para que seja possível a gravação dos dados foi utilizado o *framework* Spring, o qual utiliza o paradigma de Inversão de Controle (*IoC*) e Injeção de Dependência, como pode ser visto na Listagem 4.

```
14    @Controller
15    @Scope("view")
16    public class CategoriaController extends CrudController<Categoria,
Integer> {
17
18        @Autowired
19        private CategoriaService categoriaService;
20
21        @Override
22        protected ICrudService<Categoria, Integer> getService() {
23            return this.categoriaService;
24        }
25
26        @Override
```

```
27        protected String getUrlFormPage() {
28            return "/pages/cadastros/categoria/categoriaForm.xhtml?faces-
redirect=true";
29        }
30    }
```

Listagem 4 – Uso do Spring para persistência

A Listagem 4 apresenta a codificação da classe controladora de categoria. Nessa classe é notável a presença da anotação *@Autowired* que é responsável por fazer a Injeção de Dependência do objeto "CategoriaService". Assim, o desenvolvedor deixa de ser responsável por inicializar o objeto. Essa tarefa é realizada pelo Spring. Assim, não há mais a necessidade de se codificar toda a classe de acesso aos dados do objeto (*Data Access Object* (DAO)), há apenas a necessidade de se criar uma *interface* que herda da classe *JpaRepository* que é fornecida pelo *framework* Spring.

Para que os dados permaneçam em memória durante a exibição e a interação do usuário com a página (*view)* é utilizada a anotação *@Scope("view")*. Esses dados somente serão destruídos no momento em que a página é trocada. E também são evitados problemas relacionados ao AJAX, uso inadequado do escopo de sessão e aplicação e buscas desnecessárias ao banco de dados.

Na Listagem 5 está uma parte da implementação da classe correspondente ao *controller* das reservas.

```
47    @Controller
48    @Scope("view")
49    public class ReservaController extends CrudController<Reserva,
Integer> {
50        @Autowired private ReservaService reservaService;
51        @Autowired private ReservaItemService reservaItemService;
52        @Autowired private AmbienteService ambienteService;
53        @Autowired private EquipamentoService equipamentoService;
54        @Autowired private MaterialDeConsumoService
materialDeConsumoService;
55        private ScheduleModel scheduleModel;
56        private ScheduleEvent scheduleEvent = new DefaultScheduleEvent();
57        private final Calendar calendar = Calendar.getInstance();
58        private String tipo;
59        private MaterialDeConsumo materialDeConsumo;
60        private Equipamento equipamento;
61        private BigDecimal quantidade;
62        private BigDecimal qtdEstoque;
63
64        @Override
65        protected void inicializar() {
66            this.quantidade = BigDecimal.ZERO;
67            populaSchedule();
68        }
69
70        private void populaSchedule() {
71            scheduleModel = new DefaultScheduleModel();
```

```
72            for (Reserva reserva : reservaService.findAll()) {
73                Calendar inicio = Calendar.getInstance();
74                inicio.setTime(reserva.getData());
75                inicio.set(inicio.get(Calendar.YEAR),
inicio.get(Calendar.MONTH), inicio.get(Calendar.DATE),
getHour(reserva.getHoraInicio()), getMinute(reserva.getHoraInicio()));
76
77                Calendar fim = Calendar.getInstance();
78                fim.setTime(reserva.getData());
79                fim.set(fim.get(Calendar.YEAR), fim.get(Calendar.MONTH),
fim.get(Calendar.DATE), getHour(reserva.getHoraFim()),
getMinute(reserva.getHoraFim()));
80
81                scheduleModel.addEvent(new
DefaultScheduleEvent(getTitle(reserva), inicio.getTime(), fim.getTime(),
reserva));
82            }
83
```

Listagem 5 – Implementação da classe controller das reservas

A classe *controller* das reservas deve ter a anotação @Controller que é implementada pelo framework Spring. Um *controller* é um componente que recebe um "HttpServletRequest" e "HttpServletResponse" e é capaz de participar do fluxo de trabalho MVC. Essa deve ser uma classe reutilizável, *thread-safe*, capaz de lidar com múltiplas solicitações HTTP em todo ciclo de vida de uma aplicação.

Na Listagem 5 é possível verificar como é populado o calendário com as reservas já cadastradas, por meio do método "populaSchedule()". Nesse método é instanciado um modelo padrão de calendário fornecido pelo PrimeFaces e por meio da variável reservaService, que fornece o método de busca de todos os registros pelo método findAll(), é possível buscar todas as reservas do banco de dados. Em seguida, com a variável scheduleModel é adicionado um evento passando o título, o horário de início e fim e um dado do tipo reserva.

Na Listagem 6 está a implementação do calendário na página principal (*home*) do sistema.

```
<div class="reservasSchedule">
15                <p:schedule id="reservas"
widgetVar="widgetReservasSchedule"
value="#{reservaController.scheduleModel}" timeZone="GMT-3"
16                            timeFormat="HH:mm" draggable="false">
17                    <p:ajax event="dateSelect"
listener="#{reservaController.onDateSelect}" update=":formEvent"
oncomplete="PF('eventDialog').show();" />
18                    <p:ajax event="eventSelect"
listener="#{reservaController.onEventSelect}" update=":formEvent"
oncomplete="PF('eventDialog').show();" />
19                    <p:ajax event="eventMove"
listener="#{reservaController.onEventMove}" update="messages" />
20                    <p:ajax event="eventResize"
listener="#{reservaController.onEventResize}" update="messages" />
21                </p:schedule>
```

```
22              </div>
```

Listagem 6 – Implementação do calendário na página principal

Na Listagem 6, como pode ser visualizado na *tag* "<p:schedule" é atribuída à propriedade *value* a variável "scheduleModel" instanciada no *controller*. Essa variável contém as reservas, sendo assim o próprio componente "p:schedule" fornecido pelo PrimeFaces se encarrega de fazer a apresentação dos dados na página.

Esse componente fornece, também, o evento dateSelect utilizando a *tag* "<p:ajax event='dateSelect'", no qual deve ser atribuído o método que será acessado pela propriedade listener. E na propriedade oncomplete é atribuído o comando que será executado assim que a requisição AJAX for completada. O método onDateSelect() pode ser visualizado na Listagem 7.

```
119      public void onDateSelect(SelectEvent selectEvent) {
120          this.qtdEstoque = BigDecimal.ZERO;
121          reset();
122          this.entity.setData((Date) selectEvent.getObject());
123          this.entity.setUsuario(JsfUtil.getUsuarioLogado());
124          if
(((Permissao)JsfUtil.getAttributeSession(JsfUtil.PERMISSAO_USUARIO_LOGADO))
.getId() != RolesEnum.USER.ordinal() + 1) {
125
this.entity.setOutroUsuario(JsfUtil.getUsuarioLogado().getUsername());
126          }
127          scheduleEvent = new DefaultScheduleEvent("", (Date)
selectEvent.getObject(), (Date) selectEvent.getObject(), this.entity);
128      }
```

Listagem 7 – Método onDateSelect()

O método onDateSelect() recebe como parâmetro um evento que o próprio *framework* se encarrega de passar. Nesse parâmetro (selectEvent) está um objeto do tipo Date, que é a data em que o usuário selecionou. A partir disso, é chamado o método "reset()" que instancia uma nova reserva e é atribuída a data nesta reserva e o usuário logado. Também é instanciado um novo evento padrão fornecido pelo PrimeFaces "scheduleEvent = new DefaultScheduleEvent", passando para o método construtor, a data selecionada, e a entidade reserva.

Logo após essa requisição AJAX ser completada será exibida uma *dialog*, como pode ser visualizada na Figura 15.

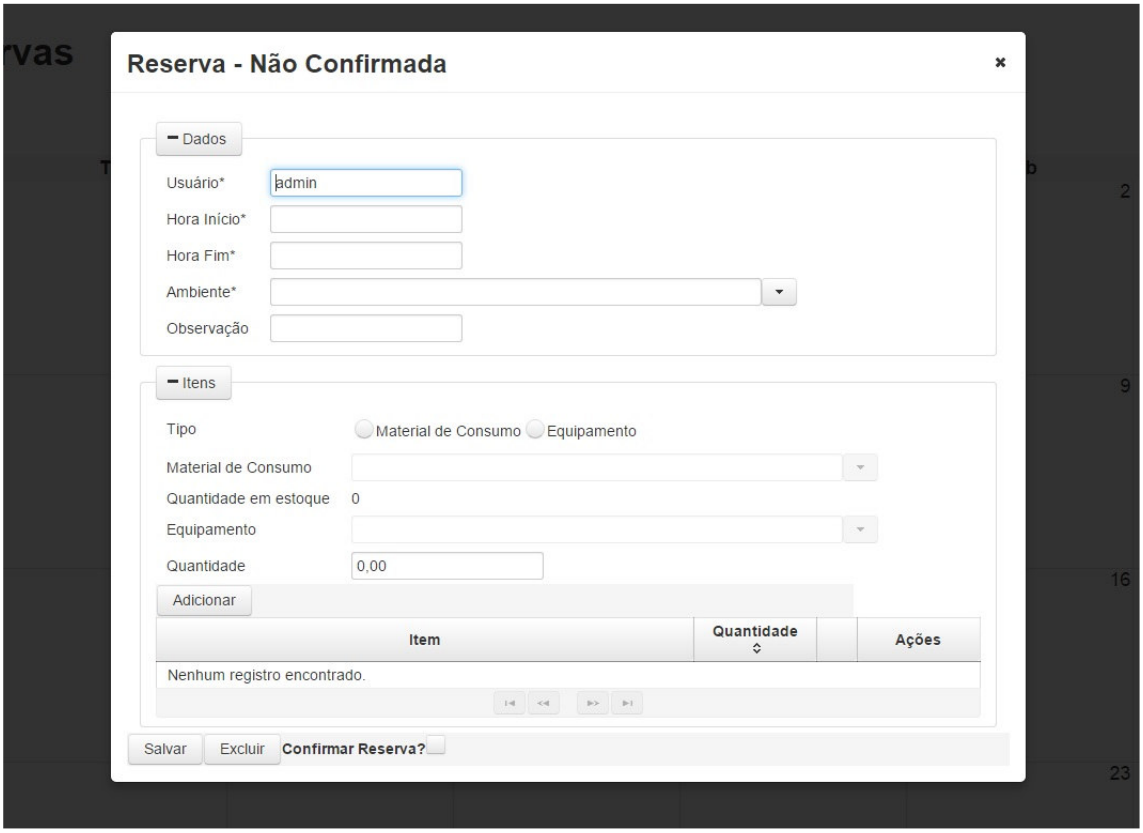

Figura 15 – Tela de reserva

Uma parte da implementação dessa *dialog* pode ser visualizada na Listagem 8.

```
<p:fieldset legend="#{form.itens}" toggleable="true">
65                              <p:panelGrid columns="2" id="gridItens">
66                                  <h:outputLabel value="#{form.tipo}"/>
67                                  <p:selectOneRadio id="tipo"
value="#{reservaController.tipo}"
disabled="#{reservaController.entity.confirmada}">
68                                      <f:selectItem
itemLabel="#{form.materialDeConsumo}" itemValue="M" />
69                                      <f:selectItem
itemLabel="#{form.equipamento}" itemValue="E" />
70                                      <p:ajax event="change"
update="materialDeConsumo, equipamento" global="false"/>
71    </p:selectOneRadio>
73     <h:outputLabel value="#{form.materialDeConsumo}" for="materialDeConsumo"/>
74     <p:autoComplete value="#{reservaController.materialDeConsumo}"
75     completeMethod="#{reservaController.completeMaterialDeConsumo}"
76     converter="#{materialDeConsumoConverter}"
77     label="#{form.materialDeConsumo}" id="materialDeConsumo"
var="materialDeConsumo"
78     itemLabel="#{materialDeConsumo.nome}" itemValue="#{materialDeConsumo}"
79     dropdown="true" forceSelection="true" size="60"
80     disabled="#{!reservaController.tipo.equals('M') ||
reservaController.entity.confirmada}">
```

```
81    <p:ajax event="itemSelect" listener="#{reservaController.onItemSelect}"
update="qtdEstoque" />
82    </p:autoComplete>
83
84    <h:outputLabel value="#{form.quantidadeEmEstoque}"/>
85    <h:outputText id="qtdEstoque" value="#{reservaController.qtdEstoque}"/>
86
87    <h:outputLabel value="#{form.equipamento}" for="equipamento"/>
88    <p:autoComplete value="#{reservaController.equipamento}"
89    completeMethod="#{reservaController.completeEquipamento}"
90    converter="#{equipamentoConverter}"
91    label="#{form.equipamento}" id="equipamento" var="equipamento"
92    itemLabel="#{equipamento.nome}" itemValue="#{equipamento}"
93    dropdown="true" forceSelection="true" size="60"
94    disabled="#{!reservaController.tipo.equals('E') ||
reservaController.entity.confirmada}"/>
95
96    <h:outputLabel value="#{form.quantidade}"/>
97    <p:inputText value="#{reservaController.quantidade}" id="quantidade"
disabled="#{reservaController.entity.confirmada}">
98    <f:convertNumber locale="pt_BR" pattern="#,##0.00" />
99    </p:inputText>
100
101   <f:facet name="footer">
102   <p:commandButton value="#{form.adicionar}"
action="#{reservaController.addItem}" ajax="true"
103   update="gridItens, tableItens, msgsReserva" process="@this, otherUsuario,
horaInicio, horaFim, ambiente, tipo, materialDeConsumo, equipamento, quantidade"
104   disabled="#{reservaController.entity.confirmada}"/>
105   </f:facet>
106   </p:panelGrid>
```

Listagem 8 – Implementação do dialog

Na Listagem 8 é possível verificar a implementação da parte dos itens. Por meio da *tag* “<p:fieldset toggleable=’true’>” é possível obter uma espécie de painel expansível e retrátil. Esses efeitos auxiliam a melhorar a usabilidade do usuário dependendo da resolução da tela em que se está visualizando a página.

Por meio da *tag* “<p:autoComplete” fornecida pelo PrimeFaces é possível buscar os registros referentes a qualquer entidade no banco. Para essa busca é necessário informar a propriedade *value* que será alimentada quando selecionado um registro. completeMethod é o método responsável por buscar os dados no banco de dados e apresentar no componente.

Converter é a classe responsável por passar para o componente uma String ou um Integer quando necessário. itemLabel é o que será apresentado no componente, por exemplo o nome. itemValue é o valor correspondente ao item, que no caso pode ser um objeto.

Para criar o componente de seleção de material de consumo, foi passado para o *value* um objeto do tipo MaterialDeConsumo, para o completeMethod(), o método de pesquisa, como pode ser visualizado na Listagem 9.

```
103 public List<MaterialDeConsumo> completeMaterialDeConsumo(String nome) {
104 return materialDeConsumoService.findByNomeContaining(nome);
105 }
```

Listagem 9 – Implementação do método List()

Nesse método List() é utilizando o objeto materialDeConsumoService que fornece o método de pesquisa por nome, retornando, assim, uma lista de materiais de consumo que contém o nome passado por parâmetro. Esse parâmetro é o que o usuário digita na caixa de seleção.

Para o a propriedade itemLabel foi passado o nome do material de consumo e para o itemValue o próprio objeto materialDeConsumo, vindo da lista do método completeMaterialDeConsumo(). O mesmo é feito para a seleção dos equipamentos.

Após selecionar um equipamento ou material de consumo, é necessário informar a quantidade do mesmo, para isso utilizamos a *tag* “<p:inputText”.

Para adicionar o item selecionado com a quantidade informada é preciso clicar no botão “Adicionar” que é construído a partir da *tag* “<p:commandButton” atribuindo à propriedade action o método que será executado ao clicar nele, nesse caso “addItem”.

A implementação desse método pode ser visualizada na Listagem 10.

```
276      public void addItem() {
277          try {
278              validaQuantidadeEmEstoque();
279              if (!isAlreadyExistsItem()) {
280                  ReservaItem reservaItem = new ReservaItem();
281                  reservaItem.setReserva(this.entity);
282                  reservaItem.setQuantidade(this.quantidade);
283                  if (this.tipo.equals("E")) {
284                      reservaItem.setEquipamento(this.equipamento);
285                  } else {
286
reservaItem.setMaterialDeConsumo(this.materialDeConsumo);
287                  }
288                  if (this.entity.getReservasItens() == null) {
289                      this.entity.setReservasItens(new ArrayList<>());
290                  }
291                  this.entity.getReservasItens().add(reservaItem);
292              }
```

Listagem 10 – Implementação do método para adicionar itens

No método da Listagem 11 está a implementação de uma validação. Caso esteja sendo inserido um material de consumo, verifica-se a quantidade que está sendo inserida é válida, ou seja, não ultrapassa a quantidade em estoque, isso no método "validaQuantidadeEmEstoque()".

```
293                 this.qtdEstoque = BigDecimal.ZERO;
294                 this.quantidade = null;
295                 this.equipamento = null;
296                 this.materialDeConsumo = null;
297             } catch (IllegalArgumentException e) {
298                 addMessage(e.getMessage(), FacesMessage.SEVERITY_ERROR);
299                 e.printStackTrace();
300             }
301     }
```

Listagem 11 – Implementação de validação de itens

Caso seja validada a quantidade em estoque, ou esteja, está sendo inserido um equipamento, é por meio do método "isAlreadyExistsItem()" que o usuário é impedido de duplicar os itens. Caso já exista na lista apenas é adicionada a quantidade informada na que já existe na lista. Caso contrário um novo objeto do tipo ReservaItem é instanciado, atribuído o item e a quantidade e esse objeto é adicionado na lista de itens da reserva.

Por fim, atua o método de salvar a reservas por meio da *tag* "<p:commandButton" atribuindo à *action* o método "addEvent()". Esse método está na Listagem 12.

```
public void addEvent() {
142             try {
143                 validaHorario();
144                 validaDisponibilidadeDaSalaNoHorario();
145                 validaItens();
146                 this.entity.setConfirmada(Boolean.FALSE);
147                 if (scheduleEvent.getId() == null) {
148                     scheduleModel.addEvent(scheduleEvent);
149                 } else {
150                     scheduleModel.updateEvent(scheduleEvent);
151                 }
152                 save();
153                 scheduleEvent = new DefaultScheduleEvent();
154                 populaSchedule();
155             } catch (Exception e) {
156                 addMessage(e.getMessage(), FacesMessage.SEVERITY_ERROR);
157             }
158         }
```

Listagem 12 – Método adicionar eventos

Para a inserção da reserva no banco de dados, é necessário verificar se o horário escolhido é válido, se há disponibilidade da sala no horário e se todos os itens escolhidos, caso existam, estão válidos também. O método "validaHorario()" é apresentado na Listagem 13.

```
private void validaHorario() throws IllegalHorarioException {
205          if (this.entity.getHoraInicio().getTime() >
this.entity.getHoraFim().getTime()) {
206              throw new IllegalHorarioException("Hora de Início deve ser
menor ou igual a Hora de Fim!");
207          }
208      }
```

Listagem 13 – Método para validar dados de horário

Esse método, apresentado na Listagem 13, apenas valida se o horário de inicio informado é menor que o horário de fim, caso contrário uma exceção é lançada e exibindo uma mensagem para o usuário, impedindo-o de gravar a reserva, como pode ser visualizado na Figura 16.

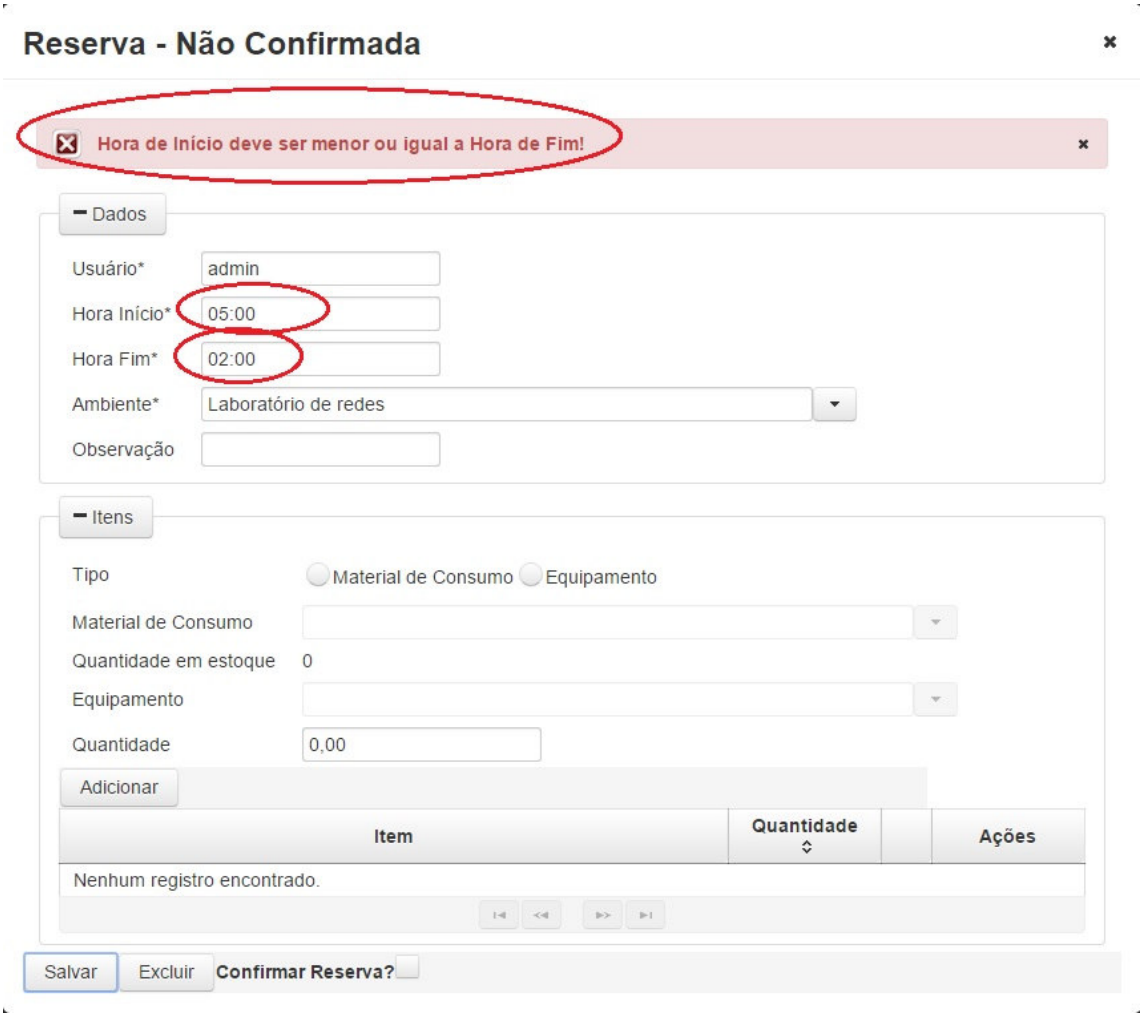

Figura 16 – Mensagem de validação de horário

Caso as validações estejam corretas é invocado o método “save()” que gravará no banco de dados a reserva e seus itens. O método “populaSchedule ()” é chamado novamente para que o calendário seja atualizado com a nova reserva.

5 CONCLUSÃO

A modelagem de um sistema *web* para gerenciamento de empréstimos de materiais e equipamentos e de reserva de ambientes e a implementação do sistema foram realizadas como planejado, atendendo os objetivos definidos para o trabalho. Para o desenvolvimento foram utilizadas tecnologias dentre elas a linguagem Java que agregada a *frameworks* e componentes possibilita a implementação de aplicações *web* caracterizadas como de interface rica.

Os objetivos propostos foram cumpridos, um sistema para gerenciamento de laboratórios didáticos, exemplificados aqui pela sala de apoio do DAELE e pelo laboratório hardware e software do de DAINF, foi desenvolvido. O sistema permite o controle de estoque de materiais, reservas de ambientes, equipamentos e materiais para a realização de atividades práticas e o empréstimo de materiais.

Como o sistema para ambiente *web* facilita o acesso para que professores realizem e gerenciam as reservas que fazem. Com acesso exclusivo, cada professor visualiza e manipula apenas as suas reservas. Um ator, denominado laboratorista, realiza o controle de estoque e o gerenciamento das reservas.

Em decorrência da quantidade e diversidade de tecnologias disponibilizadas para a implementação de aplicações, é notável que o desenvolvimento tornou-se algo muito mais produtivo. Quando o programador utiliza as ferramentas necessárias e quando as utiliza de forma adequada com os padrões corretos há uma grande agilidade em termos de codificação. Porém, devido a grande quantidade de tecnologias disponíveis, como linguagens, *frameworks* e outros, as escolhas se tornaram mais complexas. Para selecionar as tecnologias utilizadas no desenvolvimento de um sistema, é necessário compreender o seu funcionamento e identificar as vantagens e as desvantagens para utilizar os recursos que oferecem de forma adequada.

Quando tecnologias gratuitas são utilizadas, o programador, muitas vezes, se depara com problemas e fica dependente da ajuda de membros da comunidade (fóruns, *blogs*) para resolvê-los, podendo, assim comprometer o desenvolvimento do sistema. O *framework* PrimeFaces 5.0, por exemplo, a versão .0 é gratuita e as versões, .1, .2, etc.. são pagas e possuem vários *bugs* corrigidos e melhorias que são disponibilizadas apenas para clientes pagos.

Na versão do *framework* utilizado no desenvolvimento do aplicativo resultado da realização deste trabalho, foi encontrado um problema no uso da *tag* de *upload* de arquivos.

Esse problema foi corrigido nas versões pagas, mas na comunidade gratuita foi identificada uma medida paliativa para que o componente funcione.

O *framework* Spring apresenta complexidade de configuração, o seu funcionamento é de difícil entendimento e também não é muito fácil compreender a forma de uso correto dos padrões no desenvolvimento do projeto. Contudo, após compreender o funcionamento do *framework* Spring foram identificados vantagens, como por exemplo, inversão de controle, que reduz o acoplamento entre as classes; injeção de dependência que serve para resolver a Inversão de Controle e; a facilidade de utilização por meio de *interfaces* não necessitam a codificação do DAO.

Assim, ressalta-se que fica a critério do desenvolvedor estudar e compreender as tecnologias que melhor se adaptam ao desenvolvimento do projeto e identificar as suas vantagens e desvantagens. Além disso, pode ser levada em consideração a disponibilidade de materiais, que inclui fóruns, comunidades e *blogs*, por exemplo, para auxiliar no desenvolvimento e torná-lo menos complexo e mais ágil.

REFERÊNCIAS

AMALFITANO, Domenico; FASOLINO, Anna Rita; POLCARO, Armando; TRAMONTANA, Porfirio. **Comprehending Ajax web applications by the DynaRIA tool.** Conference on the Quality of Information and Communications Technology (QUATIC), 2010 Seventh International, 2010, p. 122-131.

CZEKALSKA, Karolina; SAKOWICZ, Bartosz; MURLEWSKI, Jan; NAPIERALSKI, Andrzej K T . **Hotel reservation system based on the JavaServer Faces technology**. In: International Conference on Modern Problems of Radio Engineering, Telecommunications and Computer Science (TCSET'2008), 2008, p. 470-473.

DEB, Brijesh; BANNUR, Sunil G. BHARTI Shaurabh. **Rich Internet Applications (RIA). Opportunities and challenges for enterprises**. Infosys. White Paper. 2007, p. 1-10.

FAYAD Mohamed E; SCHMIDT Douglas C. Object-oriented application frameworks. **Communication of ACM,** v. 40, n. 10, p. 32-38.

FRANZINI, Fernando. **Listagem de frameworks Java**. 2013. Disponível em: <http://imasters.com.br/linguagens/java/listagem-de-frameworks-java/>. Acesso em: 27 set. 2014.

FRATERNALI, Piero; ROSSI, Gustavo; SÁNCHEZ-FIGUEROA, Fernando. Rich Internet Applications. **IEEE Computer Society**, May/June 2010, p. 9-12.

JAVASERVER FACES. **JavaServer Faces.** Disponível em: <http://www.dsc.ufcg.edu.br/~jacques/cursos/daca/html/jsf/jsf.htm>. Acesso em: 27 set. 2014.

JUNWU, Xu; JUNLING, Liang. **Developing CRM system of web application based on JavaServer Faces**. In: Second International Workshop on Education Technology and Computer Science, p. 766-769, 2010.

KRUCHTEN, Philippe. **Introdução ao RUP:** Rational Unified Process. 2. ed. Rio de Janeiro, RJ: Ciência Moderna, 2004.

MELIÁ, Santiago; PARDILLO, Jesús; CACHERO, Cristina. **Automatic selection of RIA software architectures using quality models**. Seventh International Conference on the Quality of Information and Communications Technology, 2010, p. 505-510.

PITANGA, Talita. **JavaServer Faces: a mais nova tecnologia Java para desenvolvimento web.** Disponível em: <http://www.paulojose.pro.br/Aulas/Tutorial_inicial.pdf>. Acesso em: 27 set. 2014.

RAIBLE, Matt. **Comparing JVM web frameworks**. 2013. Disponível em: <http://static.raibledesigns.com/repository/presentations/Comparing_JVM_Web_Frameworks_DevoxxFR2013.pdf>. Acesso em: 27 set. 2014.

RAIBLE, Matt. **Comparing web frameworks**. 2006. Disponível em: <https://equinox.dev.java.net/framework-comparison/WebFrameworks.pdf>. Acesso em: 27 set. 2014.

VAZ, Agostinho M.; MARTINS, Bruno M.; BRANDÃO, Rodrigo C.; Alberti, Antonio M. Internet of Information and Services: A conceptual architecture for integrating services and contents on the future Internet. **IEEE Latin America Transactions**, v. 10, n. 6, December 2012, p. 2292-2300.

YOSHIRIRO, José. **Comparação: frameworks Java para desenvolvimento web 2.0**. 2011. Disponível em: <http://www.mabesi.com/artigos/programacao-web/jsp/1-comparacao-frameworks-java-para-desenvolvimento-web-20.html>. Acesso em: 27 set. 2014.

ZHANG Xiaoting; CHENG, Gengguo. **Design of laboratory management system based on JSF framework**. International Conference on Computer Science and Software Engineering, 2008, p. 147-150.

APÊNDICE A – Configuração do Spring

applicationContext-jpa.xml: configuração para que o Spring localize qual é o XML responsável pelas configurações do JPA

```
<bean id="vendorAdaptor"
class="org.springframework.orm.jpa.vendor.HibernateJpaVendorAdapter"/>

<bean id="entityManagerFactory"
class="org.springframework.orm.jpa.LocalContainerEntityManagerFactoryBean">

        <property name="jpaVendorAdapter" ref="vendorAdaptor" />

        <property name="persistenceXmlLocation" value="classpath:META-
INF/persistence.xml" />

        <property name="persistenceUnitName" value="persistenceLabsControl" />

</bean>
```

APÊNDICE B – Configuração do Spring Security

applicationContext-security.xml: configurações do Spring-Security para autorização das páginas para os usuários e como o Spring identifica qual classe será responsável pela autenticação do usuário

```
<security:http  auto-config="false" use-expressions="true">

        <security:form-login login-page="/login.xhtml?faces-redirect=true" authentication-
failure-url="/login.xhtml?error=bad_credentials" default-target-
url="/pages/index.xhtml?logou=true" />

        <security:access-denied-handler error-page="/access-denied.xhtml"/>

        <security:intercept-url pattern="/" access="permitAll" />

        <security:intercept-url pattern="/j_spring_security_check" access="permitAll" />

        <security:intercept-url pattern="/j_spring_security_logout" access="permitAll" />

        <security:intercept-url
pattern="/pages/cadastros/materialdeconsumo/materialDeConsumoForm.xhtml"
access="hasAnyRole('ROLE_ADMIN', 'ROLE_ATENDENTE')" />

        <security:intercept-url
pattern="/pages/cadastros/equipamento/equipamentoForm.xhtml"
access="hasAnyRole('ROLE_ADMIN', 'ROLE_ATENDENTE')" />

        <security:intercept-url pattern="/pages/**" access="isAuthenticated()" />

        <security:logout logout-url="/j_spring_security_logout" invalidate-session="true"/>

</security:http>

<!-- Classe responsável pela autenticação do usuário -->

<beans:bean id="userDetailsService"
class="br.edu.utfpr.labscontrol.model.service.impl.UsuarioServiceImpl"/>

<beans:bean id="encoder"
class="org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder"/>

<security:authentication-manager>

        <security:authentication-provider user-service-ref="userDetailsService">

                <security:password-encoder ref="encoder" />

        </security:authentication-provider>

</security:authentication-manager>
```