UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
ESPECIALIZAÇÃO EM TECNOLOGIA JAVA

EDSON GUSTAVO TOFOLO

SISTEMA PARA GERENCIAMENTO DAS SOLICITAÇÕES DE REQUERIMENTOS DA UTFPR CÂMPUS PATO BRANCO

TRABALHO DE CONCLUSÃO DE CURSO

PATO BRANCO
2017

EDSON GUSTAVO TOFOLO

SISTEMA PARA GERENCIAMENTO DAS SOLICITAÇÕES DE REQUERIMENTOS DA UTFPR CÂMPUS PATO BRANCO

Trabalho de Conclusão de Curso, apresentado ao Curso de Especialização Java, da Universidade Tecnológica Federal do Paraná, Câmpus Pato Branco, como requisito parcial para obtenção do título de Especialista.

Orientador: Prof.Vinicius Pegorini

PATO BRANCO
2017

MINISTÉRIO DA EDUCAÇÃO
Universidade Tecnológica Federal do Paraná
Câmpus Pato Branco
Departamento Acadêmico de Informática
Curso de Especialização em Tecnologia Java

TERMO DE APROVAÇÃO

SISTEMA PARA GERENCIAMENTO DAS SOLICITAÇÕES DE REQUERIMENTOS DA UTFPR CÂMPUS PATO BRANCO

por

EDSON GUSTAVO TOFOLO

Este trabalho de conclusão de curso foi apresentado em 08 de dezembro de 2017, como requisito parcial para a obtenção do título de Especialista em Tecnologia Java. Após a apresentação o candidato foi arguido pela banca examinadora composta pelos professores Andreia Scariot Beulke e Beatriz Terezinha Borsoi. Em seguida foi realizada a deliberação pela banca examinadora que considerou o trabalho aprovado.

Vinicius Pegorini
Prof. Orientador (UTFPR)

Andreia Scariot Beulke
Banca (UTFPR)

Beatriz Terezinha Borsoi
Banca (UTFPR)

Robison Cris Brito
Coordenador da V Especialização em Tecnologia Java

A Folha de Aprovação assinada encontra-se na Coordenação do Curso.

RESUMO

TOFOLO, Edson Gustavo. Sistema para solicitações de requerimentos e convalidações. 2017. 51 f. Monografia de especialização - Curso de Especialização Java, Universidade Tecnológica Federal do Paraná, Câmpus Pato Branco. Pato Branco, 2017.

Os universitários da Universidade Tecnológica Federal do Paraná muitas vezes precisam fazer requerimentos junto ao Departamento de Registros Acadêmicos (DERAC) para conseguir fazer segunda chamada de provas, trancamento de matrícula, etc. convalidação de disciplinas cursadas em outras Universidades ou cursos, entre outros procedimentos. Dessa maneira, neste trabalho foi desenvolvido um sistema *web* para que esses universitários possam fazer tais requerimentos sem a necessidade de se locomoverem até o DERAC para fazê-lo. Além dos alunos, o sistema também atenderá os servidores da Universidade, os professores e os coordenadores de cursos, os quais poderão verificar quais alunos de quais cursos estão fazendo requerimentos, dar prosseguimento neles e avaliar sobre o assunto que está sendo requisitado.

Palavras-chave: JavaServer Pages. Spring-boot. Linguagem Java. Maven. jQuery. Bootstrap.

ABSTRACT

TOFOLO, Edson Gustavo. System for requesting applications and validations. 2017. 51 f. Monografia de especialização - Curso de Especialização Java, Universidade Tecnológica Federal do Paraná, Câmpus Pato Branco. Pato Branco, 2017.

The students of the Federal University of Technology - Paraná (UTFPR) usually need to fill forms at the Department of Academic Records (DERAC) in order to request a make-up exam, canceling a registration, etc. And also, when students come from another university, the may require the validation of courses taken there, among other requests. In this work a web system was developed to assist the students, so they can make such requests without having to go to the DERAC's office. Beside the students, the system will also help university's employees, teachers and course coordinators, which will be able to check which students from which courses they are applying, follow up the processes and evaluate about the subject being requested.

Keywords: JavaServer Pages. Spring-boot. Java language. Maven. jQuery. Bootstrap.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTAGENS DE CÓDIGOS

LISTA DE SIGLAS

AJAX	Asynchronous JavaScript and XML
API	*Application Programming Interface*
CSS	*Cascading Style Sheet*
DERAC	Departamento de Registros Acadêmicos
DIRGRAD	Diretoria de Graduação e Educação Profissional
DIRGE	Diretoria Geral
DAO	*Data Access Object*
GNU/GLP	*General Public License*
HTML	*HiperText Markup Language*
HTTP	*Hypertext Transfer Protocol*
IoC	*Inversion of Control* (Inversão de Controle)
Java EE	*Java Enterprise Edition*
JSP	*JavaServer Pages*
JSON	*JavaScript Object Notation*
MVC	*Model-View-Controller*
NUAPNE	Núcleo de Atendimento às Pessoas com Necessidades Especificas
ORM	*Object-Relational Mapping*
RIA	*Rich Internet Application*
RF	Requisito Funcional
RNF	Requisito Não Funcional
UML	Unified Modeling Language
URL	Uniform Resource Locator
UTFPR	Universidade Tecnologia Federal do Paraná
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 de Registros Acadêmicos (DERAC) da Universidade Tecnologia Federal do Paraná (UTFPR), Câmpus Pato Branco, possui um processo para gerenciar os pedidos de requerimentos dos alunos. Para realizar esse processo, o aluno deve-se dirigir ao DERAC e apanhar uma folha com as opções de requerimentos desejados. Caso necessite de documentos, o aluno deve providenciar cópias para que seja possível dar início ao processo. Outro procedimento que exige requerimento é a solicitação de convalidação de disciplinas cursadas em outra universidade ou em outros cursos na própria universidade. No caso de convalidação de disciplinas, é necessário indicar as disciplinas que serão avaliadas para deferir ou não a convalidação.

A gestão desses requerimentos tem sido realizada de maneira não muito prática e muito trabalhosa, pois todo o procedimento é realizado manualmente, tanto para o aluno quanto para os servidores da UTFPR. Atualmente o aluno preenche um formulário impresso, entrega para o servidor responsável, que gera manualmente um número de protocolo e da sequência o requerimento e os documentos são enviados para o departamento ou coordenação de referido curso. Além da coordenação, dependendo do tipo de requerimento, os papéis tramitam para outros setores antes de retornar para a secretaria acadêmica.

O desenvolvimento de sistemas computacionais visa facilitar e otimizar o controle de processos, possibilitar manter histórico das informações e auxiliar em consultas. Sistemas desenvolvidos para *web* podem melhorar ainda mais esse processo, pois permitem o acesso às informações de qualquer dispositivo conectado à Internet.

Uma das maneiras para desenvolver aplicações *web* é utilizando conceitos que caracterizam esse tipo de aplicação como *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.

Sendo, verificou-se a necessidade de desenvolver um sistema caracterizado como RIA que possa contribuir para a gestão dos requerimentos efetuados pelos alunos da UTFPR Câmpus Pato Branco, bem como o andamento dos processos por parte dos servidores do DERAC.

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 dos requerimentos efetuados pelos alunos junto ao DERAC da UTFPR.

1.2.2 Objetivos Específicos

- Facilitar pedidos de requerimentos dos alunos ao DERAC.
- Melhorar o acompanhamento do processo dos pedidos de requerimentos.
- Permitir as coordenações e professores o controle dos processos a eles atribuídos.

1.3 JUSTIFICATIVA

Um software para gerenciamento dos requerimentos efetuados pelos alunos auxiliará na gestão de documentos e pedidos, os quais são realizados pelos alunos e o DERAC ou responsável (departamento ou coordenação) defere ou indefere.

A sua implementação se justifica pela agilidade na realização dos requerimentos, pela facilidade provida aos servidores - sejam técnicos administrativos, estagiários, professores ou coordenadores - e para os alunos que fazem os requerimentos.

O sistema foi implementado para uso na *web*, facilitando, assim, o pedido de requerimento por parte do aluno e se foi deferido ou não pelo DERAC e/ou responsável. Assim, não haverá mais a necessidade de o aluno deslocar-se fisicamente até o DERAC para fazer o pedido do requerimento que necessita.

O sistema proverá uma forma prática e intuitiva de o aluno pedir o requerimento e de o DERAC e os envolvidos poderem visualizá-lo para deferir ou não o pedido.

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

1.4 ESTRUTURA DO TRABALHO

Este texto está organizado em capítulos. O Capítulo 2 apresenta as ferramentas e as tecnologias utilizadas na modelagem e na implementação do sistema. No Capítulo 3 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 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.

2.1 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 *Unified Modeling Language* (UML).
Linguagem Java	JDK 1.8	http://www.oracle.com	Linguagem de programação.
IntelliJ IDEA	2016.3.4	http://www.jetbrains.com/idea/	Ambiente de desenvolvimento.
MySQL	5.7.17	http://www.mysql.com/	Banco de dados.
MySQL Workbench	6.3.8 CE	http://www.mysql.com/products/workbench/	Administrador do banco de dados.
Wildfly	10.1.0.Final	http://wildfly.org/	Servidor *web* para a aplicação.
jQuery	1.11.1	https://jquery.com/	Biblioteca de funções JavaScript para *web*.
jqGrid	5.2.1	http://guriddo.net/?page_id=103334	Componente JavaScript para criação de tabelas para listagens.
jqxKanban	5.4.0	https://www.jqwidgets.com/license/	Componente JavaScript para criação do painel ágil.
SweetAlert2	7.0.5	https://limonte.github.io/sweetalert2/	Componente JavaScript para criação de alertas.
DropzoneJS	5.2.0	http://www.dropzonejs.com/	Biblioteca JavaScript que fornece um componente para *upload* de arquivos.
JavaServer Pages(JSP)	2.1	http://www.oracle.com/technetwork/java/javaee/jsp/index.html	Linguagem de script para geração de conteúdo dinâmico para a*web*.
Hibernate	5.2.11	http://www.hibernate.org/	Efetuar o mapeamento objeto-relacional.
Spring Boot	2.0.0	https://projects.spring.io/spring-boot/	*Framework* para persistência dos dados.

Quadro 1– Ferramentas e tecnologias a serem utilizadas

A seguir são apresentadas, dentre as apresentadas no Quadro 1, as principais tecnologias utilizadas para o desenvolvimento da aplicação.

2.1.1 MySQL

MySQL é um gerenciador de banco de dados relacional, com código aberto o qual possui consistência, confiabilidade e é fácil de usar. Possui suporte para multiusuários e multitarefas. Esse gerenciador é desenvolvido em C e C++, tornando-o compatível com a maioria dos sistemas operacionais existentes, como:

- Windows: Compatível com todas as versões.
- Linux: Compatível com as principais versões, como Fedora, Core, Debian, SuSE e RedHat.
- Unix: Compatível com as versões Solaris, HP-UX, AIX, SCO.
- FreeBSD.
- Mac OS X Server.

Pelo fato dele ser um sistema de código aberto, é desenvolvido e distribuído sob as licenças GNU/GLP (*General Public License*), determinando o que se pode fazer na ferramenta e demais recursos (MYSQL, 2017).

2.2 Servidor de Aplicação

O Wildfly é um servidor de aplicação *open source*para hospedar aplicações *web* que seguem o padrão *Java Enterprise Edition* (Java EE). É uma plataforma de desenvolvimento voltado para servidores utilizando a linguagem Java. Essa plataforma 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 *Application Programming Interface* (API) RESTful para administração, entre outras (WILDFLY, 2017).

2.3 Framework ORM

O Hibernate é um *framework* utilizado para fazer o mapeamento objeto-relacional, conhecido como um *frameworkObject-Relational Mapping* (*ORM)*. 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 (HIBERNATE, 2017).

2.4 Spring

O Spring Boot é um *framework open source* para Java que se baseia em padrões de projeto, inversão de controle e injeção de dependência. O contêineré o responsável por instanciar as classes de uma aplicação Java e definir as dependências entre elas por meio de configurações efetuadas por anotações Java. Ele tem vários módulos com o Spring Data e o 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 (SPRING BOOT, 2017).

2.5 jQuery

O jQuery é uma biblioteca *open source* que permite interagir com o HTML, criando, juntamente com o *Cascading Style Sheet* (CSS) na versão 3, interfaces ricas para aplicações *web*.Essa biblioteca possui diversas funções que auxiliam na interação do usuário com o sistema, como, por exemplo, executar determinada ação ao clicar em um botão, ou validar para que um campo texto permita incluir somente números (JQUERY, 2017).

2.7 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 foi implementado para ser utilizado como padrão de desenvolvimento. À medida que os requisitos foram identificados e documentados as funcionalidades eram implementadas e acrescentadas ao sistema. A seguir as principais

atividades realizadas em cada iteração.

a) Requisitos - a definição dos requisitos foi realizada a partir da descrição dos requisitos do ponto de vista do usuário realizada por alunos da universidade, professores e coordenador do curso.

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 visando a 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.

d) Testes – os testes foram informais e com o objetivo de identificar erros de codificação.

3 RESULTADO

Este capítulo apresenta o resultado da realização deste trabalho que é um sistema para gestão dos requerimentos de alunos junto à secretaria acadêmica da Universidade Tecnológica Federal do Paraná, Câmpus Pato Branco.

3.1 ESCOPO DO SISTEMA

O sistema possibilitará ao aluno efetuar pedidos de requerimentos e acompanhar sua situação. O DERAC poderá visualizar todos os requerimentos deferindo-os ou não e caso seja necessário repassará o pedido e respectivos documentos para outros setores. Por exemplo: um pedido de segunda chamada de prova, após o DERAC deferir, delegará para o coordenador de curso responsável por avaliar o pedido.

O sistema possui como funcionalidades principais:

a) Cadastro de requerimentos – requerimentos são os pedidos que os alunos fazem junto ao DERAC.

b) Controle de situação – por meio da troca de situação do requerimento é possível gerenciar em qual estado ele está e se foi deferido ou não por qual departamento, seja o DERAC, coordenação do curso ou outro.

Cadastro de requerimentos –o requerimento é definido informando qual o motivo, sendo eles: afastamento para estudos no exterior, atividades domiciliares, avaliação antecipada, avaliação diferenciada, cancelamento de disciplinas, declaração de matrícula, declaração de provável formando, desistência do curso, 2ª chamada de prova, diploma 2ª via, histórico atualizado, histórico de conclusão do ensino superior, matrícula em atividades complementares, matrícula em estágio supervisionado, matrícula em disciplinas, mudança de turma, planos de ensino/ementas de disciplinas, trancamento total de matrícula, 2ª via do crachá, outros e convalidação.

Para determinados motivos, como por exemplo, 2ª chamada de prova, é necessário informar outros dados como: professor, disciplina e data da prova. No cancelamento de disciplinas é necessário informar as disciplinas a serem canceladas. Quando houver necessidade de informar disciplina, será possível escolher de qual curso a(s) disciplina(s) pertence(m). Ao gravar o cadastro do requerimento, ele será exibido ao usuário do DERAC

para que seja realizado o devido encaminhamento, deferindo ou não o pedido ou encaminhando para outro setor, que pode ser a coordenação de curso, por meio da situação.

Controle de situação–a situação do requerimento ocorre pelo tipo do usuário, na qual somente usuários que não são alunos e professores poderão alterá-lo. As situações são: a) em aberto: quando um aluno cadastra um requerimento, o mesmo ficará visível para este e para o usuário DERAC;

b) aprovado: quando o DERAC aprova o requerimento sem precisar delegar a outro setor.

c) devolvido DERAC: quando o DERAC já deferiu o pedido, porém precisou delegar a outro setor, como por exemplo, ao Coordenador, e o mesmo já deu seu parecer.

d) cancelado: quando o aluno ou o DERAC cancela o pedido.

e) aguardando coordenação: quando o DERAC já deferiu o pedido e precisou delegar para a coordenação do curso.

f) aguardando DIRGRAD/DIRGE/NUAPNE: quando o DERAC já deferiu e delegou para o departamento de Diretoria de Graduação e Educação Profissional (DIRGRAD), Diretoria Geral (DIRGE) ou Núcleo de Atendimento às Pessoas com Necessidades Especificas (NUAPNE).

3.2 MODELAGEM DO SISTEMA

Os usuários que possuem acesso ao sistema pertencem a quatro categorias: derac, coordenação, aluno, professor. O usuário do DERAC e das coordenações estão pré-cadastrados na base de dados, necessitando apenas alterar a senha. Os alunos e professores fazem seu próprio cadastro. Será diferenciada a autenticação para professores e alunos, na qual os alunos acessarão o sistema pelo registro acadêmico (RA) e os professores pela primeira parte do e-mail da UTFPR. A seguir os usuários e suas funções específicas:

a) **DERAC**

- Manutenção de usuários.
- Alterar a situação do requerimento para aprovado, encaminhar coordenação/DIRGRAD/DIRGE/NUAPNE ou finalizar.
- Deferir ou não o requerimento.

b) **Aluno**

- Cadastrar requerimentos.

- Cancelar o requerimento indicando o *status* como cancelado.
- Visualização das listagens dos requerimentos que ele fez.

c) **Coordenação**

- Visualização dos requerimentos com a situação: aguardando coordenação.
- Alteração do *status* do requerimento para devolvido DERAC.
- Deferir ou não o requerimento.

d) **Professor**

- Visualizar os requerimentos quando for do tipo convalidação e o referido professor foi atribuído para dar o parecer em alguma disciplina da convalidação.

A Figura 1 apresenta o diagrama de casos de uso do sistema considerando os usuários como os quatro atores do sistema.

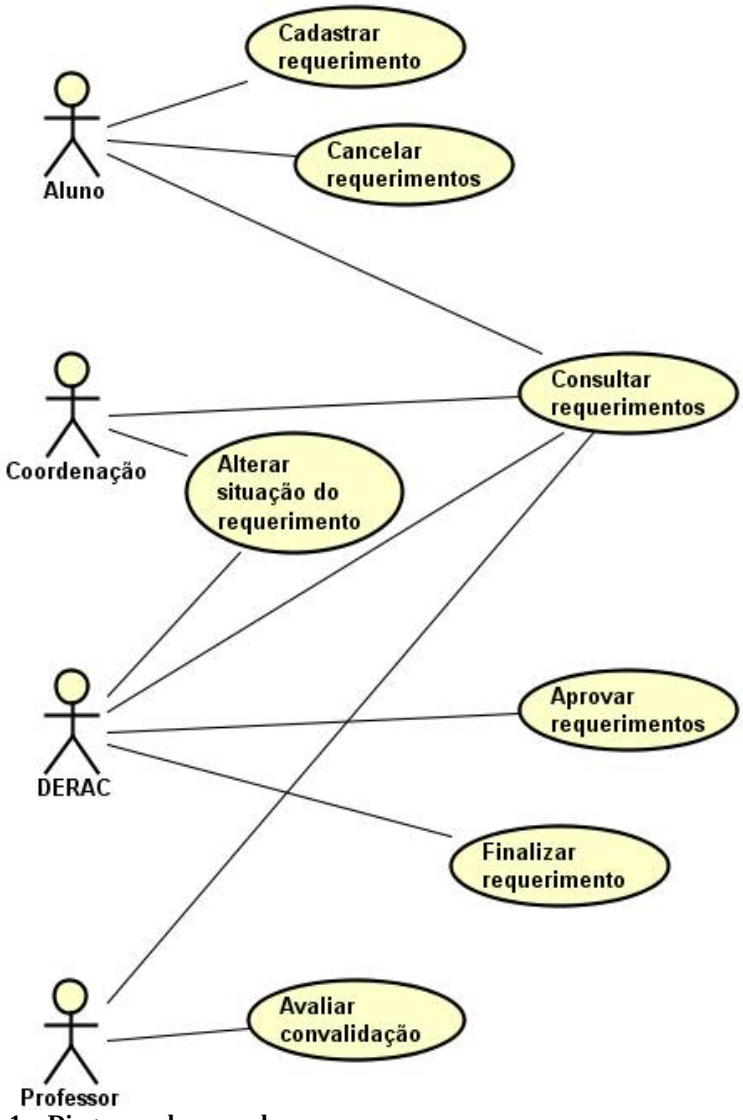

Figura 1 – Diagrama de casos de uso

O Quadro 2 apresenta os requisitos funcionais agrupados por casos de uso apresentados na Figura 3.

	Caso de uso	Requisito	Descrição
RF1	Cadastrar requerimento	Incluir requerimento	Um usuário aluno pode incluir requerimentos.
RF2	Cadastrar requerimento	Excluir requerimento	Um usuário aluno pode excluir requerimentos que ele cadastrou, enquanto o DERAC não houver alterado a situação do requerimento.
RF03	Alterar situação do requerimento	Alterar para cancelado	Um usuário aluno pode alterar a situação do requerimento quando ele ainda estiver com situação: em aberto.
RF04	Alterar situação do requerimento	Alterar para: aprovado, finalizado, encaminhar para DIRGE/DIRGRAD/ NUAPNE/Coordenação	O usuário DERAC poderá alterar a situação do requerimento conforme necessidade.
RF05	Avaliar convalidação	Dar parecer sobre as disciplinas	Usuários professores poderão dar o parecer sobre uma ou mais disciplinas as que lhe forem atribuídas quando um requerimento for do tipo convalidação.

Quadro 2 – Requisitos funcionais

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

	Requisito	Descrição
RNF01	Permissões	Controlar as ações de cada usuário conforme suas permissões.
RNF02	Acesso ao sistema	Usuário aluno não tem acesso à alteração da situação do requerimento quando ele já estiver com situação em aberto. Professor visualiza somente os requerimentos que lhe foram atribuídos para avaliar e só pode avaliar a disciplina de convalidação atribuída pela coordenação. Coordenação atribui o professor para avaliar os itens da convalidação, porém só pode alterar a situação do requerimento quando todos os itens já estiverem avaliados pelos professores correspondentes. DERAC altera a situação dos requerimentos.

Quadro 3 – Requisitos não funcionais

A Figura 2 apresenta o diagrama de entidades e relacionamentos do banco de dados. A tabela “usuarios” armazena os dados dos alunos, professores e servidores da UTFPR. Cada tipo de usuário é diferenciadp por meio do campo “tipo_usu”, o login será pelo campo “username_usu”. Os alunos se cadastrarão utilizando o registro acadêmico e os professores pela primeira parte do *e-mail* da UTFPR.

A tabela “requerimentos” armazena os dados dos requerimentos efetuados pelos alunos, sendo o motivo do requerimento identificado pelo campo “motivo_req”, a situação na

qual se encontra o requerimento é gravada no campo “status_req”. A tabela “disciplinas” manterá os dados das disciplinas oferecidas pelos cursos da UTFPR-PB, sendo possível identificar de qual curso pertence a disciplina por meio do campo “curso_id”, o qual faz relação direta com a tabela “cursos”.

As convalidações requeridas pelos alunos são armazenadas na tabela “requerimentos_convalidacoes”, sendo a disciplina da UTFPR escolhida pelo aluno gravada no campo “disciplina_utfpr_id_con” e a disciplina que utilizará para convalidação é identificada por meio do campo “disciplina_convalidacao_con”. O professor que fará avaliação da convalidação é atribuído ao campo “professor_id_con”, que faz relação com a tabela “usuarios”, pois obrigatoriamente esse professor deve estar cadastrado no sistema como um usuário.

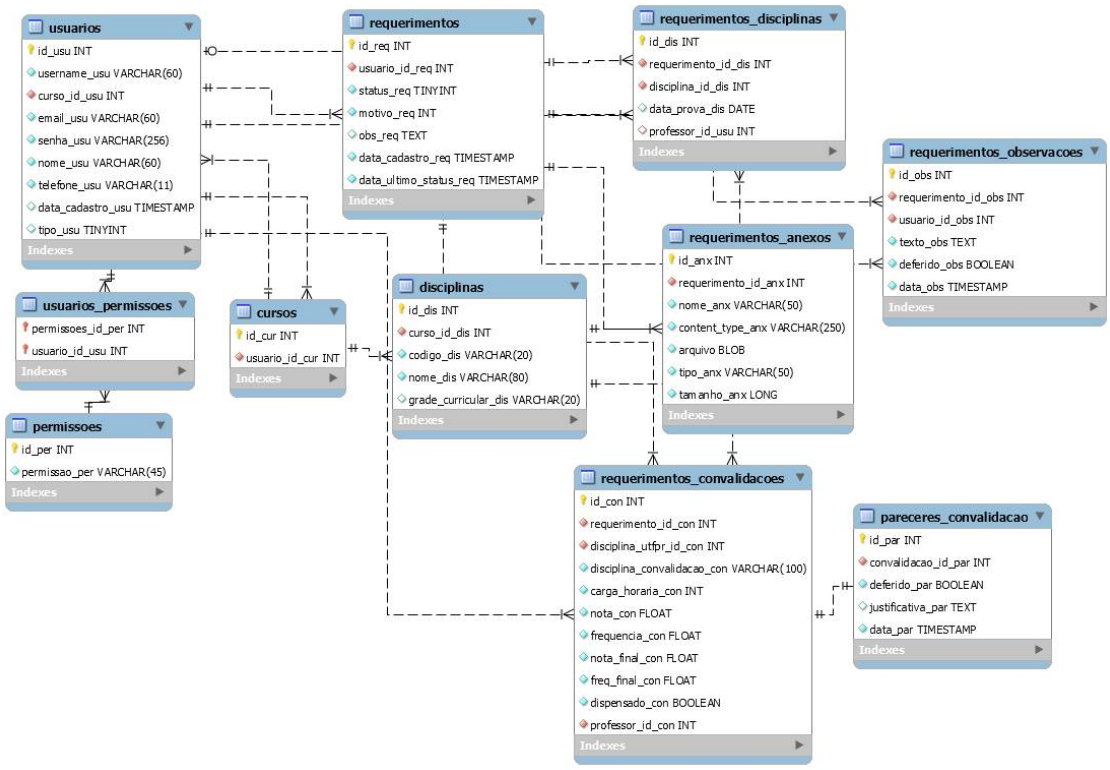

Figura 2 – Diagrama de entidades e relacionamentos

Os quadros a seguir apresentam a descrição das tabelas constantes na Figura 2.

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

Campo	**Tipo**	**Nulo**	**Chave primária**	**Chave estrangeira**	**Observações**
id_usu	Numérico	Não	Sim	Não	
username_usu	Texto	Não	Não	Não	

curso_id_usu	Numérico	Não	Não	Sim	Da tabela cursos
email_usu	Texto	Não	Não	Não	
senha_usu	Texto	Não	Não	Não	
nome_usu	Texto	Não	Não	Não	
telefone_usu	Texto	Não	Não	Não	
data_cadastro_usu	DataHora	Não	Não	Não	
tipo_usu	Numérico	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**
id_per	Numérico	Não	Sim	Não
permissao_per	Texto	Não	Não	Não

Quadro 5 – Tabela permissões

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**
permissoes_id_per	Numérico	Não	Não	Sim	
usuário_id_usu	Numérico	Não	Não	Sim	Da tabela usuários

Quadro 6 – Tabela usuarios_permissoes

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

Campo	**Tipo**	**Nulo**	**Chave primária**	**Chave estrangeira**
id_cur	Numérico	Não	Sim	Não
usuario_id_cur	Numérico	Não	Não	Sim

Quadro 7 – Tabela cursos

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

Campo	**Tipo**	**Nulo**	**Chave primária**	**Chave estrangeira**	**Observações**
id_req	Numérico	Não	Sim	Não	
usuario_id_req	Numérico	Não	Não	Sim	Da tabela usuários
status_req	Numérico	Não	Não	Não	
motivo_req	Numérico	Não	Não	Não	
obs_req	Texto	Sim	Não	Não	
data_cadastro_req	DataHora	Não	Não	Não	

data_ultimo_status_req	DataHora	Não	Não	Não	Data da última alteração do *status*

Quadro 8 – Tabela requerimentos

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

Campo	**Tipo**	**Nulo**	**Chave primária**	**Chave estrangeira**	**Observações**
id_dis	Numérico	Não	Sim	Não	
curso_id_dis	Numérico	Não	Não	Sim	Da tabela cursos
codigo_dis	Texto	Não	Não	Não	
nome_dis	Texto	Não	Não	Não	
grade_curricular_dis	Texto	Não	Não	Não	

Quadro 9 – Tabela de disciplinas

O Quadro 10 apresenta a listagem dos campos das disciplinas vinculadas ao requerimento.

Campo	**Tipo**	**Nulo**	**Chave primária**	**Chave estrangeira**	**Observações**
id_dis	Numérico	Não	Sim	Não	
requerimento_id_dis	Numérico	Não	Não	Sim	Da tabela disciplinas
disciplina_id_dis	Numérico	Não	Não	Sim	Da tabela disciplinas
professor_dis	Numérico	Sim	Não	Sim	Da tabela usuários, identificado pelo campo tipo_usu
data_prova_dis	Data	Sim	Não	Não	

Quadro 10 – Tabela requerimentos_disciplinas

O Quadro 11 apresenta a listagem dos campos da tabela dos documentos anexados ao requerimento.

Campo	**Tipo**	**Nulo**	**Chave primária**	**Chave estrangeira**	**Observações**
id_anx	Numérico	Não	Sim	Não	
requerimento_id_anx	Numérico	Não	Não	Sim	Da tabela requerimentos_anexos
nome_anx	Texto	Não	Não	Não	
content_type_anx	Texto	Não	Não	Não	
arquivo_anx	Blob	Não	Não	Não	
tipo_anx	Texto	Não	Não	Não	File;image;etc.
tamanho_anx	Texto	Não	Não	Não	

Quadro 11 – Tabela de requerimentos_anexos

O Quadro 12 apresenta a listagem das observações dos pareceres dos requerimentos ao alterar o seu *status*.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Observações
id_obs	Numérico	Não	Sim	Não	
requerimento_id_obs	Numérico	Não	Não	Sim	Da tabela requerimentos_observacoes
usuario_id_obs	Numérico	Não	Não	Sim	Da tabela usuarios
texto_obs	Texto	Não	Não	Não	
deferido_obs	Lógico	Não	Não	Não	
data_obs	Data	Não	Não	Não	

Quadro 12 – Tabela de requerimentos_observacoes

O Quadro 13 apresenta a listagem dos campos da tabela das disciplinas para convalidação.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Observações
id_con	Numérico	Não	Sim	Não	
requerimento_id_con	Numérico	Não	Não	Sim	Da tabela requerimentos_convalidacao
disciplina_utfpr_id_con	Numérico	Não	Não	Sim	Código da disciplina da UTFPR selecionada para convalidar
disciplina_convalidacao_con	Texto	Não	Não	Não	Nome da disciplina utilizada para convalidar
carga_horaria_con	Numérico	Não	Não	Não	
nota_con	Numérico	Não	Não	Não	
frequencia_con	Numérico	Não	Não	Não	
nota_final_con	Numérico	Não	Não	Não	
freq_final_con	Numérico	Não	Não	Não	
dispensado_con	Lógico	Não	Não	Não	
professor_id_con	Numérico	Não	Não	Sim	

Quadro 13 – Tabela de requerimentos_convalidacoes

O Quadro 14 apresenta a listagem dos campos da tabela dos pareceres dados às disciplinas da convalidação.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Observações
id_par	Numérico	Não	Sim	Não	
convalidacao_id_par	Numérico	Não	Não	Sim	
deferido_par	Lógico	Não	Não	Não	

justificativa_par	Texto	Sim	Não	Não	
data_par	Data	Não	Não	Não	

Quadro 14 – Tabela de pareceres_convalidacao

3.3 APRESENTAÇÃO DO SISTEMA

O leiaute do sistema é dividido em 2 setores. No setor superior é apresentada uma barra de menu, com as funcionalidades, conforme o usuário autenticado. E no setor inferior está o conteúdo da página que sendo visitada.

A Figura 3 apresenta a página principal do sistema quando acessada por um aluno. O conteúdo dessa página é um painel ágil apresentando os requerimentos do aluno autenticado com a situação cancelada ou em aberto. A imagem dessa Figura apresenta o meu Requerimento e Meus requerimentos. O Requerimento é a rotina principal do sistema, nela ocorre a alteração da situação do requerimento e determina quais pessoas já avaliaram o referido requerimento. Assim, optou-se por criar painel prático e que agilizasse a trocar de situação. Esse painel foi mantido na tela principal e ao ser arrastado para outra coluna determinará a nova situação (*status*) do requerimento.

Figura 3 – Tela principal

A Figura 4 mostra o conteúdo da página de listagem dos Requerimentos. Nessa página o usuário do tipo aluno pode visualizar os requerimentos já cadastrados e também pode realizar as operações de inclusão, alteração e cancelamento. Não há exclusão de registro, caso o aluno não queira mais o requerimento, ele poderá alterar a situação para cancelado.

Para a operação de alteração de um registro é necessário clicar no ícone que contém a imagem de um lápis, para cancelamento clicar no ícone de uma lixeira e para inserir um requerimento clicar no ícone com um símbolo de adição (sinal de mais). Se repousar o mouse

sobre os ícones será exibida uma mensagem informado a ação realizada. Esses ícones estão na área circulada na Figura 4.

Figura 4 – Acesso para edição de registro

Ainda é possível fazer uma visualização de todos os dados do requerimento, sem a necessidade de alterá-lo, para isso é necessário clicar no ícone que representa um “i” (região circulada na Figura 5).

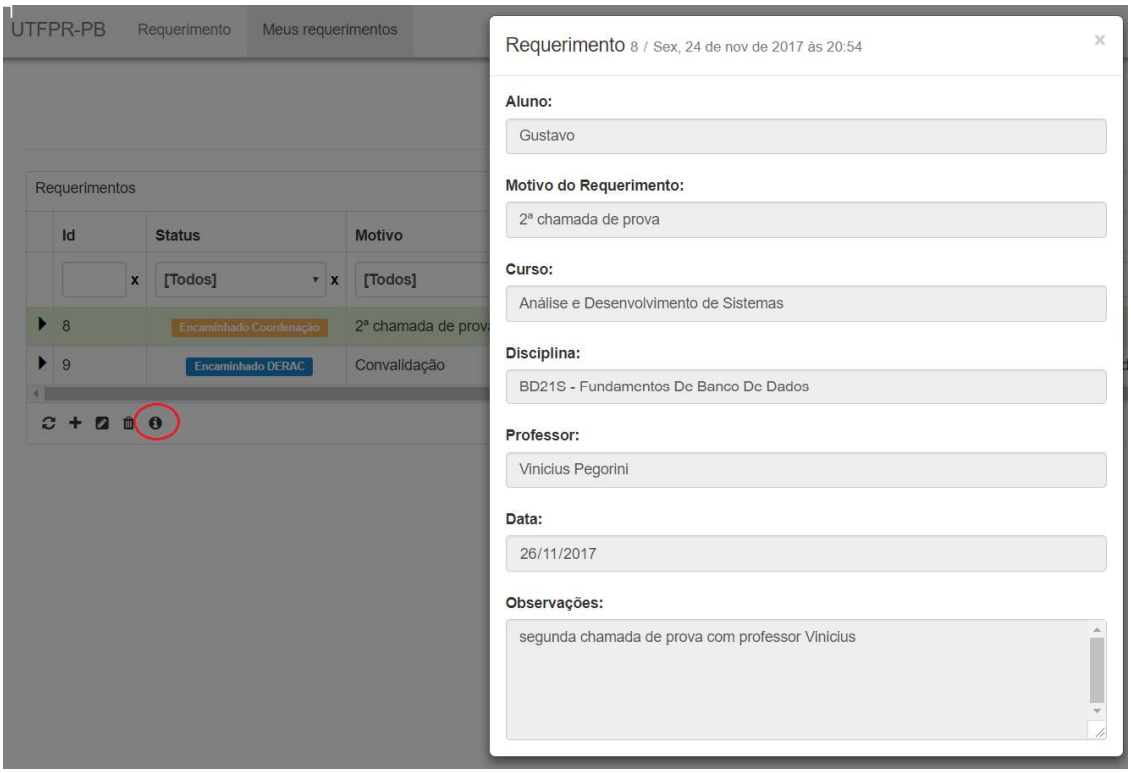

Figura 5 – Visualização de dados de cadastro

Ao clicar no ícone com sinal de mais, o sistema redirecionará o aluno para uma página com um formulário que contém os campos necessários para efetuar a inclusão de um requerimento, como mostra a Figura 6.

Figura 6 – Tela para inclusão de requerimento

Caso oaluno clique no botão “Salvar” sem informar o campo obrigatório, o sistema fará a validação e exibirá uma mensagem de erro para que o usuário possa corrigir e salvar, como mostra a Figura 7. Essa mensagem, e as outras relacionadas aos formulários, é exibida conforme as *rules* definidas no método *“validate”* disponibilizado pela biblioteca jQuery Validate.

Figura 7 – Informação de campos de preenchimento obrigatório

Essa validação ocorre quando a propriedade *required* está valorizada como *true.* Ao submeter o formulário o próprio componente se encarrega de verificar se a o campo está preenchido. E caso não esteja, é exibida uma mensagem, abaixo do componente inválido, como mostra a imagem da Figura 8.

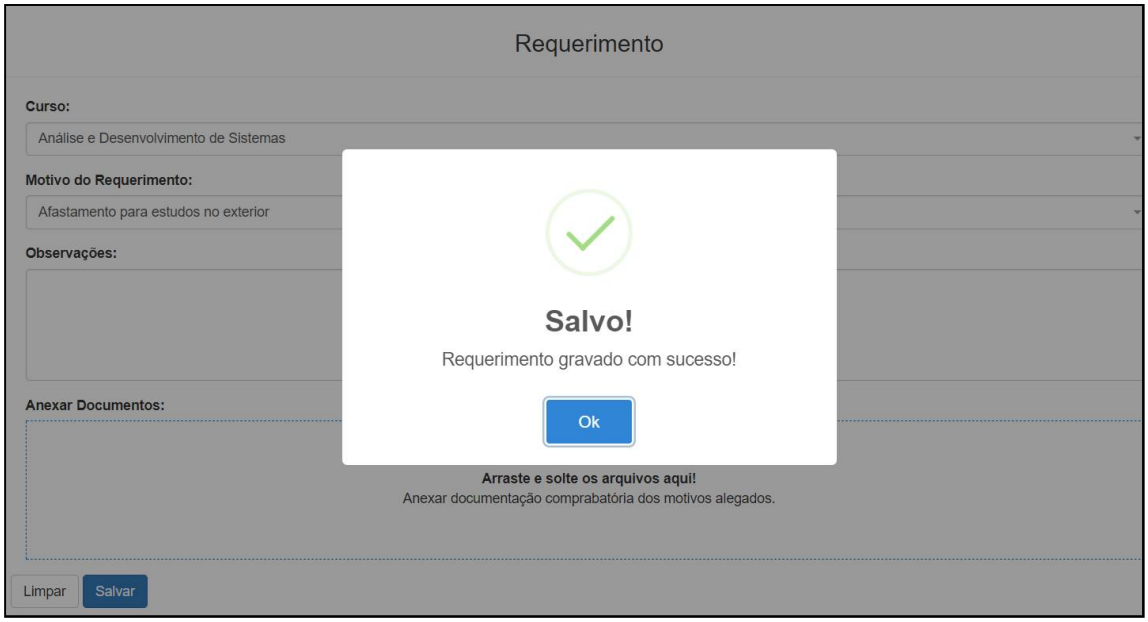

Figura 8 – Informação de inclusão com sucesso

A operação de alteração é semelhante à de inclusão, sendo que, para alterar um requerimento, o usuário deve clicar sobre o registro que deseja, selecionando-o dessa forma, e clicar no ícone de um lápis referente ao ambiente desejado. Dessa maneira, o sistema redirecionará o aluno para uma página de alteração, exibindo um formulário com os dados já preenchidos do requerimento escolhido, como pode ser observado na Figura 9. O processo de validação é o semelhante ao de inclusão.

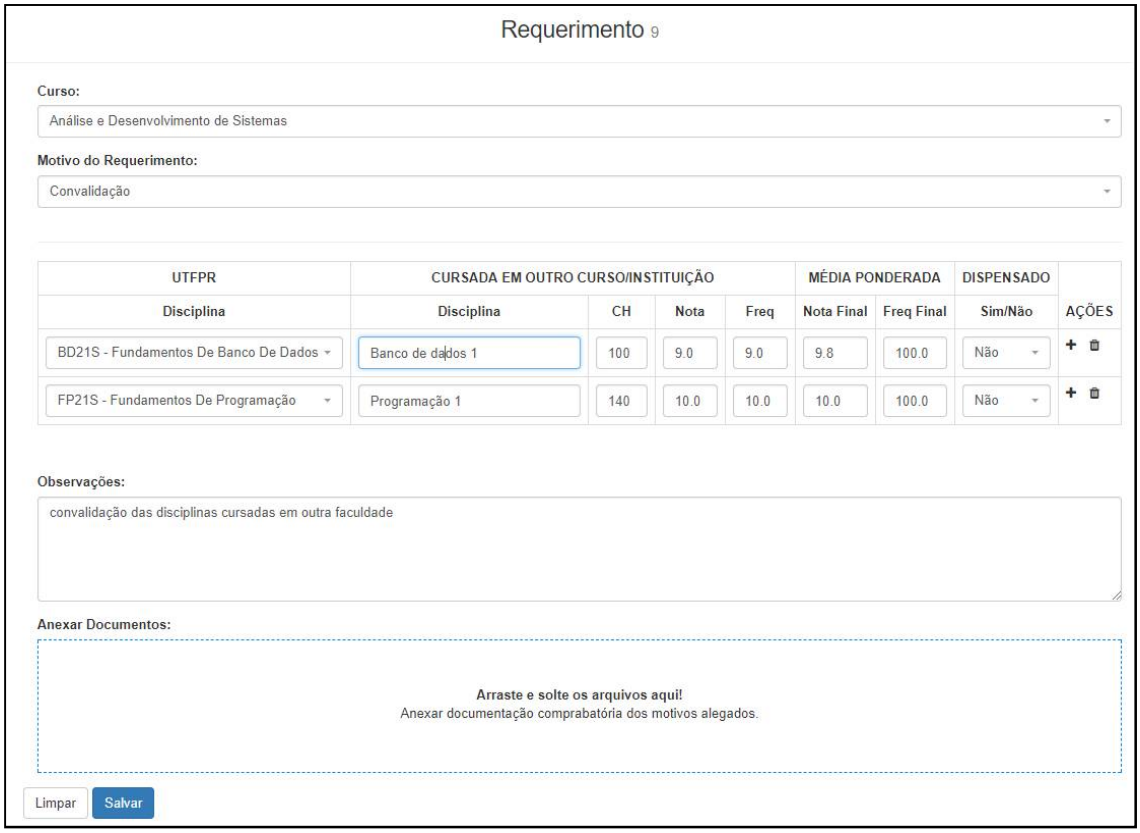

Figura 9 – Operação de alteração de dados

Para realizar a operação de cancelamento, o aluno deve clicar sobre o registro do requerimento que deseja cancelar e clicar no ícone de uma lixeira. O sistema exibirá uma mensagem de confirmação para que o cancelamento seja gravado. Essa tela pode ser visualizada na Figura 10.

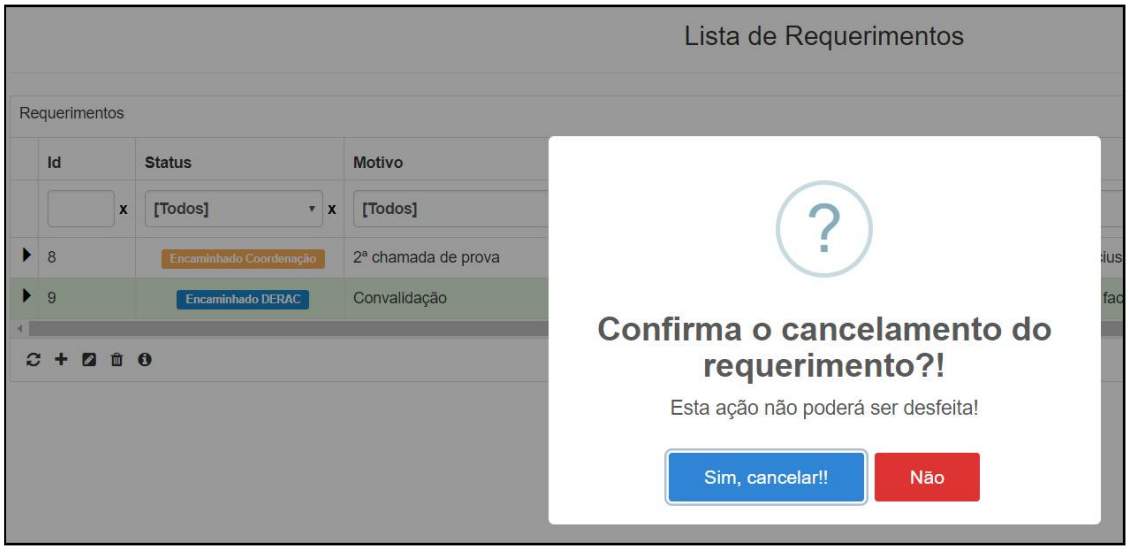

Figura 10 – Exclusão de registro

Caso oaluno confirme o cancelamento, o sistema alterará a situação do requerimento para cancelado (no exemplo da Figura 11, um requerimento cadastrado), atualizará a tabela de listagem e apresentará uma mensagem de sucesso para o cancelamento solicitado, como mostra a Figura 11.

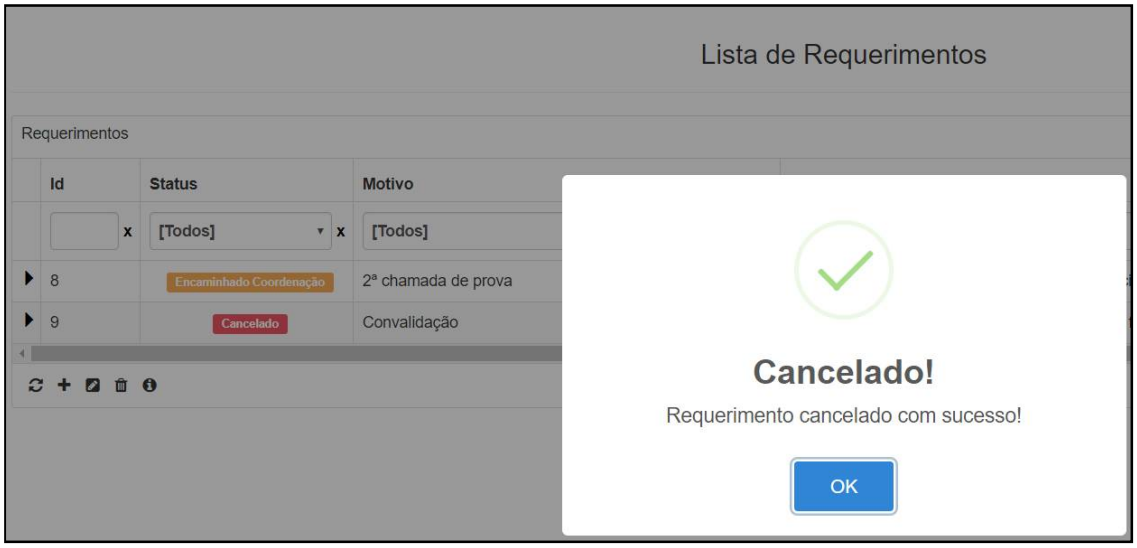

Figura 11 – Cancelamento de registro

Quando o usuário do DERAC autenticar-se, na tela inicial ele poderá visualizar todos os requerimentos em aberto e estarão disponíveis as opções de visualizar os dados do requerimento e alterar a situação deles por meio da ação arrastar-e-soltar. Esses requerimentos são visualizados em forma de *card* no painel ágil. O usuário terá opções de: aprovar, encaminhar para coordenação/DIRGRAD/DIRGE/NUAPNE, cancelar ou finalizar o requerimento. Cada *card* tem as principais informações do requerimento, como o motivo, o nome do aluno e o curso. Para visualizar todos os dados é possível clicar no ícone “i”, no topo do *card*, sendo exibido um painel *modal* com todos os dados,ou clicar na descrição do motivo, que é apresentado como um *link* e redirecionará o usuário para a página de edição, como pode ser visualizado na Figura 12.

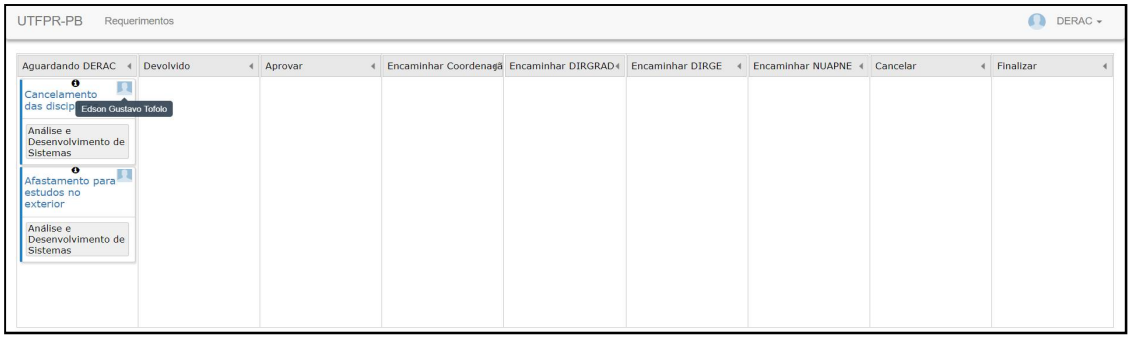

Figura 12 – Tela inicial com usuário DERAC autenticado

Na tela de edição dos requerimentos quando usuário DERAC está autenticado, é exibido o campo “Aluno”, sendo listarados o nome do aluno que cadastrou o requerimento e um campo “Status”, sendo este, possível de escolher dentre algumas opções. Também é possível fazer o *download* dos arquivos que o aluno fez *upload*, muitas vezes há a necessidade de anexar documentos comprobatórios para futura análise do DERAC ou do departamento para o qual o requerimento será enviado. Os demais campos apresentados ficarão todos desabilitados, sendo que o único que é possível de edição é o *status*, que define em qual situação esta o requerimento, como pode ser visualizado na Figura 13.

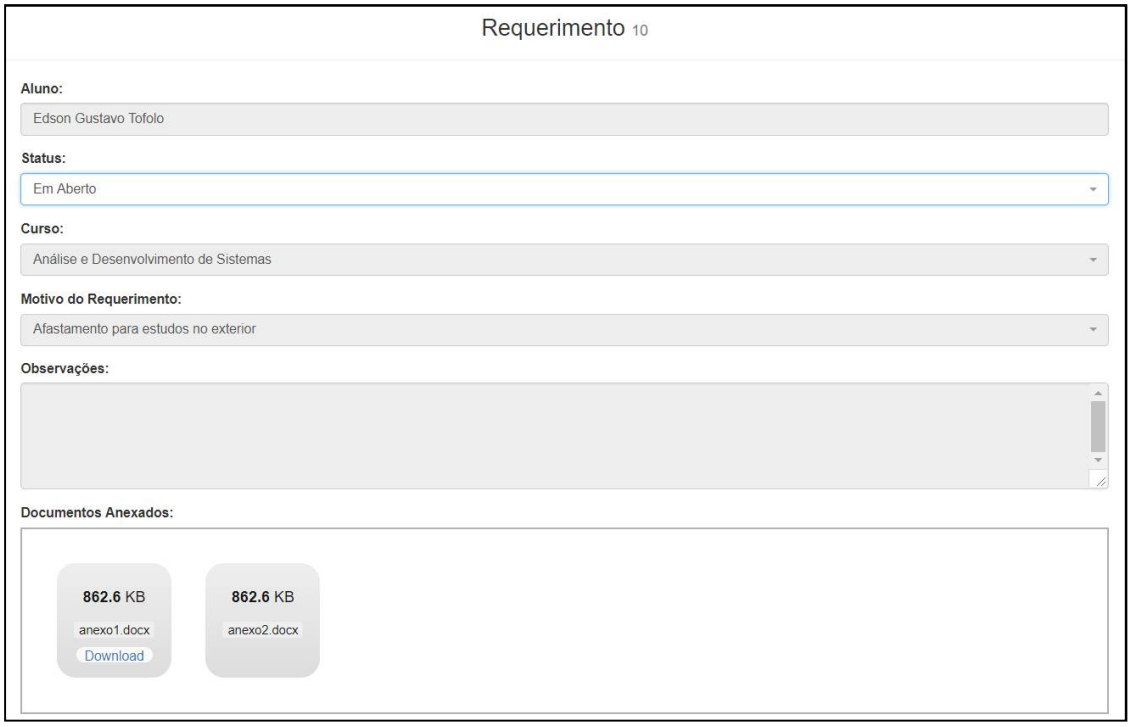

Figura 13 – Tela de edição do requerimento com usuário DERAC autenticado

Quando o usuário autenticado for do tipo Professor, poderá visualizar os requerimentos do tipo convalidação que foram atribuídos pela coordenação do curso para fazer a avaliação das disciplinas. Podendo ser uma ou mais, conforme a coordenação julgar necessário. Ao acessar a página de edição do requerimento, o professor poderá visualizar todos os dados do requerimento, porém, sem poder editá-los, da mesma forma que somente visualizará as disciplinas que lhe foram atribuídas. O professor terá a opção, por meio de um ícone identificado por um lápis, de clicar e será exibido um painel *modal* para o professor fazer a avaliação, ao confirmar o parecer ao lado desse ícone do lápis será exibido um ícone verde de confirmado, como é possível verificar na Figura 14.

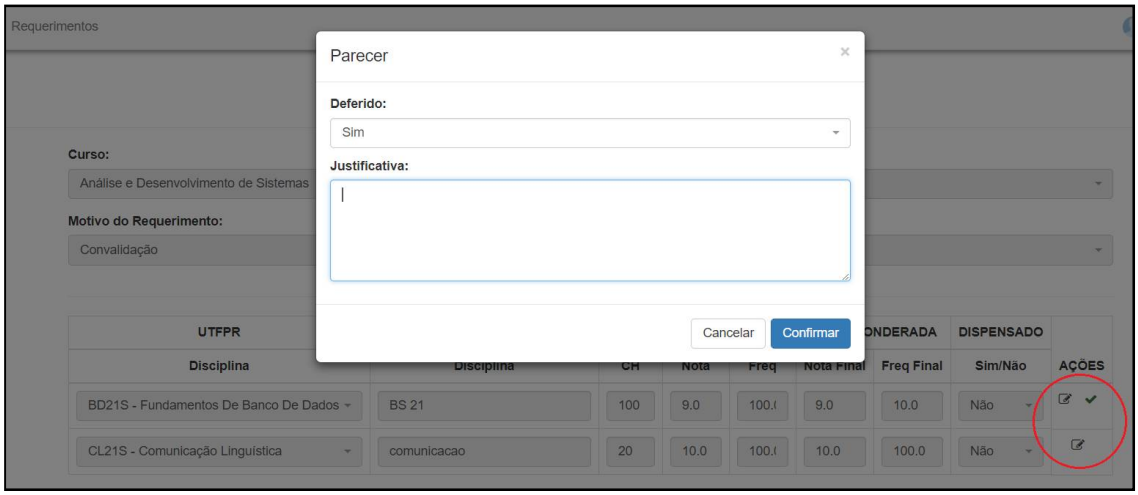

Figura 14 – Tela da avaliação da convalidação do requerimento com usuário professor autenticado

3.4 IMPLEMENTAÇÃO DO SISTEMA

Na Listagem 1 é possível verificar uma parte de como é desenvolvido o conteúdo da página de listagem dos requerimentos.

```
29                  $("#jqGrid").jqGrid({
30                      url : "/ProjetoJSP/requerimento/loadData",
31                      datatype : "json",
32                      mtype : 'POST',
33                      colModel : [ {
34                          label : 'Id',
35                          name : 'id',
36                          index : 'id',
37                          key: true,
38                          width : 70
39                      }, {
40                          label: 'Status',
41                          name: 'status',
42                          index: 'status',
```

```
                            width: 150,
                            align: 'center',
                            stype: "select",
                            searchoptions: {
                                dataInit: function (e) {
                        $(e).addClass('form-control').removeAttr('size');
                                },
                                value: ":[Todos];1:Em
Aberto;2:Aprovado;3:Com DERAC;4:Cancelado;5:Encaminhado
Coordenação;6:Encaminhado DIRGRAD;7:Encaminhado NUAPE;8:Encaminhado
DIRGE;9:Finalizado"
                            },
                            formatter: statusFormatter
                        }
   <sec:authorize access="hasAnyRole('PROFESSOR', 'COORDENACAO',
'DERAC')">
                            ,{
                                label: 'Aluno',
                                name: 'aluno',
                                index: 'aluno',
                                width: 200,
                                jsonmap: 'usuario.nome'
                            }
   </sec:authorize>
                        , {
                            label : 'Motivo',
                            name : 'motivo',
                            index : 'motivo',
                            width : 300,
                            formatter: function myformatter ( cellvalue,
options, rowObject ) {
   return motivoList[cellvalue - 1].descricao;
                            },
                            stype: "select",
                            searchoptions: {
                                dataUrl:
'/ProjetoJSP/requerimento/motivos/findAll',
                                buildSelect: function (data) {
   var response, s = '<select>', i;
                                    response = jQuery.parseJSON(data);
  s += '<option value="">[Todos]</option>';
   if (response && response.length) {
                                        $.each(response, function (i) {
  s += '<option value="' + this.id+ '">' + this.descricao+ '</option>';
                                        });
                                    }
   return s + '</select>';
                                }
                            }
                        }, {
                            label : 'Observação',
                            name : 'observacao',
                            index : 'observacao',
                            width : 500,
                        }, {
                            label : 'Data',
                            name : 'data',
                            index : 'data',
                            width : 130,
                            sorttype: 'date',
                            formatter: 'date',
```

```
98  formatoptions: { srcformat: "ISO8601Long", newformat: "d/m/Y H:i" },
99                              searchoptions: {
100                                 dataInit: function (element) {
101                                     $(element).datepicker({
102                                         language: "pt-BR",
103                                         todayBtn: "linked",
104                                         keyboardNavigation: true,
105                                         forceParse: true,
106                                         calendarWeeks: true,
107                                         autoclose: true,
108                                         format: "dd/mm/yyyy",
109                                         todayHighlight: true
110                                     });
111                                     $(element).mask("99/99/9999");
112                                 }
113                             }
114                         }],
115                         pager : '#jqGridPager',
116                         rowNum : 10,
117                         height: 'auto',
118  with: 'auto',
119  //                       autoWidth: true,
120                         shrinkToFit: true,
121  //                       forceFit: true,
122                         rowList : [ 10, 20, 30 ],
123                         viewrecords : true,
124                         caption : 'Requerimentos',
125                         jsonReader : {repeatitems : false},
126                         subGrid: true,
127                         subGridRowExpanded: showChildGrid,
128                         subGridOptions : {
129                             selectOnExpand : true
130                         }
131
132                 });
...
158                 $("#jqGrid").navGrid('#jqGridPager',
159                     { edit: false, add: false, del: false, search:
false, refresh: true, view: false, position: "left", cloneToTop: false,
refreshicon: "fa fa-refresh" },
160                         {}, {}
161                 ).navButtonAdd("#jqGridPager", { //NOVO REQUERIMENTO
162                     caption:"",
163                     buttonicon:"fa fa-plus",
164                     onClickButton: function () {
165                      window.location.href = "/ProjetoJSP/requerimento/";
166                     },
167                     position: "last",
168                     title:"Novo requerimento",
169                     cursor: "pointer"
170                 }).navButtonAdd("#jqGridPager", { //EDITAR
171                     caption:"",
172                     buttonicon:"fa fa-pencil-square",
173                     onClickButton: function () {
174                         rowKey = getSelectedRow();
175  if (rowKey) {
176                             window.location.href =
"/ProjetoJSP/requerimento/edit/" + getSelectedRow();
177                         } else {
178                             swal("Selecione um requerimento!",
"Nenhum requerimento selecionado!", "warning").catch(swal.noop); // esse
```

```
catch evita erro no console do browser
179                         }
180                     },
181                     position: "last",
182                     title:"Editar requerimento",
183                     cursor: "pointer"
184                 })
```

Listagem 1 – Conteúdo da página de listagem dos requerimentos

Para o desenvolvimento do código apresentado na Listagem 1 foi utilizada a biblioteca jQuery, com a qual foi possível acessar uma *tag* html *div* identificada por *jqGrid*, na qual utilizando o método *jqGrid* de um componente JavaScript são passadas por parâmetro várias propriedades para que o componente se encarregue de criar a listagem. Por exemplo, a propriedade *url*, que define a*Uniform Resource Locator* (URL), espera uma *string* do caminho do servidor que será utilizado para fazer a requisição *Asynchronous Javascript and XML* (AJAX) e recuperar a lista dos requerimentos, a conexão com o banco de dados se dá por meio de um arquivo de configuração apresentando no Apêndice A. Já a propriedade *datatype* espera uma *string* com o nome do tipo de dado que será recebido do servidor, no caso *JavaScript Object Notation*(JSON), a propriedade *mtype* espera uma *string* com o tipo de requisição que será feita. Também é possível visualizar a *tag* "sec:authorize" que é disponibilizada pelo Spring Security. Essa *tag* valida se o usuário autenticado tem qualquer uma das permissões definidas na propriedade *access*, renderizando ou não o conteúdo dentro dela.

É possível verificar a existência de um ícone que serve para incluir um requerimento, que ocorre pelo método *navButtonAdd* do componente JavaScript, esse método cria o ícone. Ao ser clicado será executada a função definida na propriedade *onClickButton* e o sistema redirecionará o usuário para a página de cadastro. O código desenvolvido da página de cadastro é apresentado na Listagem 2.

```
<frm:form id="frm" name="frm" method="post"
modelAttribute="requerimentoForm" action="${requerimentoActUrl}"
51enctype="multipart/form-data" autocomplete="off">
52    <input id="id" name="id" type="text" value="${id}" hidden/>
53    <c:if test="${id >0}">
54    <sec:authorize access="hasAnyRole('COORDENACAO', 'DERAC')">
55    <div class="form-group col-lg-12 col-md-12 col-sm-12">
56    <label for="alunoNome">Aluno:</label>
57    <input id="alunoNome" type="text" value="${alunoNome}" class="form-
control" disabled/>
58    </div>
59    <div class="form-group col-lg-12 col-md-12 col-sm-12">
60    <label for="status">Status:</label>
61    <select name="status" id="status" class="form-control">
62    <c:forEach items="${statuses}" var="status">
63    <option value="${status.id}" ${status.id ==
```

```
requerimento.status.ordinal() ? 'selected="selected"' : ''}>
64    ${status.descricao}
65    </option>
66    </c:forEach>
67    </select>
68    </div>
69    </sec:authorize>
70    </c:if>
71    <div class="form-group col-lg-12 col-md-12 col-sm-12">
72    <label for="curso">Curso:</label>
73    <select id="curso" name="curso" class="form-control"
${disabledFields}>
74    <c:forEach items="${cursos}" var="curso">
75    <option value="${curso.id}" ${curso.id == cursoId ?
'selected="selected"' : ''}>
76    ${curso.usuario.nome}
77    </option>
78    </c:forEach>
79    </select>
80    </div>
81    <div class="form-group col-lg-12 col-md-12 col-sm-12">
82    <label for="motivo">Motivo do Requerimento:</label>
83    <select name="motivo" id="motivo" class="form-control"
${disabledFields}>
84    <c:forEach items="${motivos}" var="motivo">
85    <option value="${motivo.id}" ${motivo.id == requerimento.motivo ?
'selected="selected"' : ''}>
86    ${motivo.descricao}
87    </option>
88    </c:forEach>
89    </select>
90    </div>
```

Listagem 2 – Conteúdo da página de cadastro de requerimentos

A página de cadastro do requerimento é chamada de “requerimentoForm.jsp” e utiliza as *tags* “frm:form” do Spring e “c:if” do JSP. A primeira é utilizada para criar um formulário que será submetido ao servidor para gravação e também validação dos dados, já a segunda serve para definir uma condição se o seu conteúdo será renderizado. Essa página também utiliza a *tag* “jsp:body” que serve para introduzir o conteúdo dela na página padrão, para isso deve-se antes, utilizar a *tag* “layout:template”. Para utilizá-la deve-se definir a *taglib* começo da página, como pode ser visto na Listagem 3.

```
1     <%@ page language="java" contentType="text/html; charset=utf-8"
pageEncoding="utf-8"%>
2     <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
3     <%@ taglib tagdir="/WEB-INF/tags/layout" prefix="layout" %>
4     <%@ taglib prefix="spring"
uri="http://www.springframework.org/tags"%>
5     <%@ taglib prefix="frm"
uri="http://www.springframework.org/tags/form"%>
6     <%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>
7     <%@ taglib prefix="sec"
uri="http://www.springframework.org/security/tags" %>
9     <layout:template>
```

Listagem 3 – Tags para incluir página requerimento na página padrão

Para a persistência dos dados foi utilizado o *framework* Spring-Data, que utiliza o paradigma de Inversão de Controle (*Inversion of Control - IoC*). Para que esse *framework* funcione é necessário realizar algumas configurações, como pode ser visto no Apêndice B. A classe RequerimentoRepository, apresentada na Listagem 4, é responsável pela persistência dos dados do requerimento.

```
15   public interface RequerimentoRepository extends
JpaRepository<Requerimento, Long>, JpaSpecificationExecutor<Requerimento>
{
16       List<Requerimento> findByStatus(StatusRequerimentoEnum status);
17
18       @Query(value = "select distinct r from Requerimento r join
RequerimentoDisciplina rd on rd.requerimento.id=r.id join Disciplina d on"
+
19   " d.id=rd.disciplina.id join Curso c on c.id=d.curso.id join Usuario
u on u.id=c.usuario.id where u.id=?1 and r.status=?2")
20       List<Requerimento> findAllToCoordenacao(Long coordenacaoId,
StatusRequerimentoEnum status);
21
22       @Query(value = "select distinct r from Requerimento r join
RequerimentoConvalidacao c on c.requerimento.id=r.id join Usuario u on
u.id=c.professor.id where u.id=?1")
23       List<Requerimento> findAllToProfessor(Long professorId);
24   }
```

Listagem 4 – Uso do Spring-Data para persistência

A Listagem5 apresenta a codificação da classe controladora de requerimento. Nessa classe é possível verificar a presença da anotação *@Autowired*, que é responsável por fazer a injeção de dependência do objeto "requerimentoRepository". Assim, o desenvolvedor deixa de ser responsável por inicializar o objeto, essa tarefa fica para o Spring. Não havendo mais necessidade de codificar toda a classe de acesso aos dados do objeto (*Data Access Object* - DAO), há apenas a necessidade de criar uma *interface* que herda da classe *JpaRepository* que é fornecida pelo *framework* Spring-Data.

Como o JSP é um *framework Action Based*, ou seja, baseado em ações, é necessário definir o contexto para o *controller*, que no caso é definido pela anotação *@RequestMapping*. Para que essa anotação funcione é necessário fazer algumas configurações no Spring Security, como pode ser visualizado no Apêndice C.

Na Listagem 5 está uma parte da implementação da classe correspondente ao *controller* dos requerimentos.

```
34   @Controller
35   @RequestMapping("/requerimento")
36   public class RequerimentoController {
37
38       @Autowired
39   private RequerimentoRepository requerimentoRepository;
...
```

```
108      @Secured("ROLE_PROFESSOR")
109      @PutMapping(value = "/convalidacao/gravarParecer/", consumes =
{"application/json"})
110      @ResponseBody
111  public String gravarParecer(@RequestBody RequerimentoConvalidacao
itemConvalidacao) {
112          JSONObject retorno = new JSONObject();
113  try{
114              RequerimentoConvalidacao requerimentoConvalidacao =
requerimentoConvalidacaoRepository.findById(itemConvalidacao.getId()).orEls
e(null);
115  if (Objects.nonNull(requerimentoConvalidacao)) {
116  if (Objects.isNull(requerimentoConvalidacao.getParecer())) {
117
itemConvalidacao.getParecer().setConvalidacao(requerimentoConvalidacao);
118
requerimentoConvalidacao.setParecer(itemConvalidacao.getParecer());
119                  } else {
120
requerimentoConvalidacao.getParecer().setDeferido(itemConvalidacao.getParec
er().getDeferido());
121
requerimentoConvalidacao.getParecer().setJustificativa(itemConvalidacao.get
Parecer().getJustificativa());
122                  }
123
requerimentoConvalidacaoRepository.save(requerimentoConvalidacao);
124                  retorno.put("state", "OK");
125                  retorno.put("message", "Parecer gravado com
sucesso!");
126              } else {
127                  retorno.put("state", "ERROR");
128                  retorno.put("message", "Item da convalidação
inexistente!");
129              }
130          }catch (Exception ex){
131              retorno.put("state", "ERROR");
132              retorno.put("message", "Falha ao gravar parecer!\n" +
ex.getCause().getCause().getMessage());
133          }
134  return retorno.toString();
135      }
```

Listagem 5 – Implementação da classe controller das requerimentos

A classe *controller*dos requerimentos deve ter a anotação @Controller que é implementada pelo *framework* Spring-MVC. Um *controller* é um componente que recebe um "HttpServletRequest" e "HttpServletResponse" e é capaz de participar do fluxo de trabalho *Model-View-Controller* (MVC). Essa deve ser uma classe reutilizável, *thread-safe*, capaz de lidar com múltiplas solicitações *Hypertext Transfer Protocol*(http) em todo ciclo de vida de uma aplicação.

Aindana Listagem 5 é possível verificar como é persistido o parecer que o professor fez para alguma disciplina da convalidação, por meio do método "gravarParecer()". Nesse método por meio da variável "requerimentoConvalidacaoRepository", que fornece a busca ao

banco de dados pelo campo “id” utilizando o método “findById()”, é possível buscar a disciplina a ser convalidada do banco de dados. Em seguida, com a variável “requerimentoConvalidacao” é adicionado o parecer do professor.

Na Listagem 6 está a implementação do parecer na página da edição do requerimento.

```
649  function parecerItemConvalidacao(itemConvalidacaoId) {
650  if ($("#itemConvalidacao" +
itemConvalidacaoId).find(".acoes").find("#parecerConfirmado"+itemConvalidac
aoId).length >0) {
651          $.getJSON('/ProjetoJSP/requerimento/convalidacao/getParecer/',
{"id": itemConvalidacaoId}, function (data) {
652              createModalParecer($("#parecer"), "paracerViewer",
"justificativa", "Parecer", "Justificativa:", data);
653              exibirParecerViewer(itemConvalidacaoId);
654          });
655      } else {
656          createModalParecer($("#parecer"), "paracerViewer",
"justificativa", "Parecer", "Justificativa:");
657          exibirParecerViewer(itemConvalidacaoId);
658      }
659  }
```

Listagem 6 – Implementação do parecer do professor

A Listagem 6 apresenta um método implementado na linguagem JavaScript, que ao receber por parâmetro o código identificador da convalidação faz uma requisição ao servidor para recuperar do banco de dados as informações do parecer, caso não receba o referido código, apenas o painel *modal* é criado a partir do método “createModalParecer()” e exibido ao usuário por meio da chamada do método “exibirParecerViewer()”.

Ométodo “exibirParecerViewer()”exibe o *modal* criado a partir da chamada do método jQuery “modal(‘show’)”. Ao confirmar o parecer, o método recupera as informações fornecidas pelo usuário e faz uma requisição AJAX para persistir os dados no banco de dados, utilizando o método “$.ajax()”, fornecido pela biblioteca jQuery. Caso os dados sejam persistidos, por meio da propriedade *success* deste método é possível exibir uma mensagem ao usuário de que o parecer foi gravado,como pode ser visualizado na Listagem 7**.**

```
603  function exibirParecerViewer(itemConvalidacaoId) {
604      $("#paracerViewer").modal('show');
605      $("#deferido").select2();
606      $("#confirmarparacerViewer").click(function () {
607  var justificativa = $("#justificativa").val();
608  var deferido = $("#deferido").select2("val");
609  var parecer = {"justificativa": justificativa, "deferido": deferido};
610  var itemConvalidacao = {"id": itemConvalidacaoId, "parecer":
parecer};
611          $.ajax({
612              headers: {
613  'Accept': 'application/json',
614  'Content-Type': 'application/json'
615              },
```

```
616                 type : 'PUT',
617                 url :
'/ProjetoJSP/requerimento/convalidacao/gravarParecer/',
618                 data: JSON.stringify(itemConvalidacao),
619                 success : function(data) {
620   if (data.state == "OK"){
621                         swal({
622                             title : "Salvo!",
623                             text : data.message,
624                             type : "success",
625                             confirmButtonText : "Ok",
626                             showCancelButton : false,
627                             allowEscapeKey: false,
628                             allowOutsideClick: false,
629                         }).then(function () {
630                             $acoes = $("#itemConvalidacao" +
itemConvalidacaoId).find(".acoes");
631   if ($acoes.find("#editParecer" + itemConvalidacaoId).hasClass("ui-
inline-del")) {
632                                 $acoes.find("#editParecer" +
itemConvalidacaoId).removeClass("ui-inline-del").addClass("ui-inline-
edit");
633                             }
634   if ($acoes.find("#parecerConfirmado"+itemConvalidacaoId).length == 0)
{
635                                 $acoes.append("<div
id='parecerConfirmado"+itemConvalidacaoId+"' class='ui-pg-div ui-inline-
del' style='float:left;'><span class='fa fa-check text-
success'></span></div>");
636                             }
637                             $("#paracerViewer").modal('hide');
638                         }).catch(swal.noop); // esse catch evita erro no
console do browser;
639                     } else {
640                         swal("Falhou!", data.message,
"error").catch(swal.noop); // esse catch evita erro no console do browser;
641                     }
642                 },//Fim success
643                 error : function() {
644                     swal("Erro!", "Falha ao gravar parecer.",
"error").catch(swal.noop); // esse catch evita erro no console do browser;
645                 }
646             }); //Fim ajax
647         });
648   }
```

Listagem 7 – Método exibirParecerViewer()

Na página “index.jsp”, que é a página inicial do sistema, existe um painel ágil para visualizar e alterar a situação dos requerimentos realizada por meio da ação de arrastar e soltar. Uma parte da implementação desse painel pode ser visualizada na Listagem 8.

```
151                     $('#kanban').jqxKanban({
152                         width: '100%',
153                         height: '100%',
154                         source: dataAdapter,
155                         resources: resourcesAdapterFunc(), //tem só pra
não dar erro ao utilizar a função addItem
156                         itemRenderer: function(element, item, resource) {
```

```
157  var url = "/ProjetoJSP/requerimento/edit/" + item.id;
158                          $(element).find(".jqx-kanban-item-avatar
img").attr('data-original-title', item.content);
159                          $(element).find(".jqx-kanban-item-avatar
img").attr('title', item.content);
160                          $(element).find(".jqx-kanban-item-avatar
img").attr('data-toggle', 'tooltip');
161                          $(element).find(".jqx-kanban-item-avatar
img").attr('data-placement', 'bottom');
162                          $(element).find(".jqx-kanban-item-
text").html('<a href=' + url + '>' + item.text + '</a>');
163                          $("<div class='jqx-icon jqx-kanban-item-
template-content jqx-kanban-template-icon' data-original-title='Ver
requerimento' title='Ver requerimento' data-toggle='tooltip' data-
placement='top'>" +
164  "<i class='fa fa-info-circle'></i></div>")
165                              .insertAfter($(element).find(".jqx-
kanban-item-avatar"));
166                      },
167                      columns: boardColumns
168                  });
169
170                  $('.jqx-kanban-item').tooltip({
171                      selector: "[data-toggle=tooltip]",
172                      container: "body"
173                  });
174              });//FIM GETJSON
175
176              $('#kanban').on('itemMoved', function (event) {
177  var args = event.args;
178  var itemId = args.itemId;
179  var oldParentId = args.oldParentId;
180  var newParentId = args.newParentId;
181  var itemData = args.itemData;
182  var oldColumn = args.oldColumn;
183  var newColumn = args.newColumn;
184
185  var itemDataBeforeMove = JSON.parse(JSON.stringify(itemData));
//clonar o obj
186
187                  changeStatus($('#changeStatus'), newColumn.dataField,
itemId,
188  function(data) {
189  if (data.state == "FAIL") {
190                              swal({
191                                  title : "Falhou!",
192                                  text : data.message,
193                                  type : "error",
194                                  confirmButtonText : "Ok",
195                                  showCancelButton : false,
196                                  allowEscapeKey: false,
197                                  allowOutsideClick: false,
198                              }).then(function() {
199                                  $('#kanban').jqxKanban('removeItem',
itemId);
200                                  $('#kanban').jqxKanban('addItem',
itemDataBeforeMove); //volta para posição original
201                              }).catch(swal.noop);
202
203                          } else if (data.state == "ERROR") {
204                              swal("Falhou!", data.message,
```

```
"error").catch(swal.noop);
205                             }
206                         },
207   function () {
208                           window.location.reload();
209                       });
210               });
```

Listagem 8 – Implementação do painel ágil na página inicial

Na Listagem 8 é possível verificar que a implementação do painel ocorre pela chamada do método “jqxKanban” disponibilizado por um componente jQuery. A propriedade “source” deste método define qual serão os dados exibidos. Também, é possível verificar a chamada do método “itemMoved”, disponibilizada pelo jQuery, que é ativado ao mover um requerimento de uma situação para outra.

Ao alterar a situação do requerimento é exibido um painel *modal* para que o usuário faça uma observação e defina o requerimento como deferido ou não. Este painel é criado a partir da chamada do método “changeStatus()” e a sua implementação pode ser vista na Listagem 9.

```
41   function changeStatus(element, status, requerimentoId,
successAjaxCallback, hideModalCallback) {
42
43       createModalParecer(element, "obsViewer", "textoObs", "Alteração
do Status", "Observação:");
44
46
47       $("#obsViewer").modal('show');
48
49       $("#deferido").select2();
50
51   var ok = false;
52
53   if (typeof(status) == "number") {
54   var path = '/ProjetoJSP/requerimento/edit/' + requerimentoId +
'/changeStatus/' + status;
55           $("#confirmarobsViewer").click(function () {
56               swal({
57                   title: 'Confirma a alteração do Status?',
58                   text: "Esta ação não poderá ser desfeita!",
59                   type: 'question', //warning
60                   showCancelButton: true,
61                   confirmButtonColor: '#3085d6',
62                   cancelButtonColor: '#d33',
63                   confirmButtonText: 'Sim, alterar!!',
64                   cancelButtonText: 'Não',
65                   allowOutsideClick: false
66               }).then(function () {
67                   ok = true;
68                   reqAJAX(path, successAjaxCallback);
69                   $("#obsViewer").modal("hide");
70               }).catch(swal.noop); // esse catch evita erro no console
do browser
71           });
```

```
72        } else {
73            $("#confirmarObs").click(function () {
74                ok = true;
75    var path = '/ProjetoJSP/requerimento/edit/' + requerimentoId +
'/changeStatusKanban/' + status;
76                reqAJAX(path, successAjaxCallback);
77                $("#obsViewer").modal("hide");
78            });
79        }
80
81    if (hideModalCallback !== undefined) {
82            $('#obsViewer').on('hidden.bs.modal', function () {
83    if (!ok) {
84                    hideModalCallback();
85                }
86            });
87        }
88    }
```

Listagem 9 – Implementação do método changeStatus()

Nesse método “changeStatus()” é criado o HTML do painel *modal*, a partir da chamada do método “createModalParecer()”, e por meio do método “modal(‘show’)”, disponibilizado pelo jQuery, ele é exibido ao usuário. Ao clicar no botão confirmar, é realizada uma requisição AJAX para poder persistir os dados no banco de dados. Para isso foi necessário utilizar o método “reqAJAX()” que chama o método “$.ajax()” da biblioteca jQuery, passando a *url* que será utilizada para gravar os dados no servidor.

O código do método no servidor que faz a gravação da observação da alteração da situação é apresentado na Listagem 10.

```
74    private String changeStatus(Long requerimentoId,
StatusRequerimentoEnum status, RequerimentoObservacao
requerimentoObservacao) {
75            JSONObject retorno = new JSONObject();
76    try{
77                Requerimento requerimento =
requerimentoRepository.findById(requerimentoId).orElse(null);
78    if (Objects.nonNull(requerimento)) {
79
requerimentoObservacao.setUsuario(ControllersUtil.getLoggedUser());
80                    requerimentoObservacao.setRequerimento(requerimento);
81
requerimento.getObservacoes().add(requerimentoObservacao);
82                    requerimento.setStatus(status);
83                    requerimento.setDataUltimoStatus(new Date());
84                    requerimentoRepository.save(requerimento);
85                    retorno.put("state", "OK");
86                } else {
87                    retorno.put("state", "ERROR");
88                    retorno.put("message", "Requerimento Inexistente!");
89                }
90            }catch (Exception ex){
91                retorno.put("state", "ERROR");
92                retorno.put("message", "Falha ao gravar requerimento!\n"
+ ex.getCause().getCause().getMessage());
```

```
93            }
94    return retorno.toString();
95        }
```

Listagem 10 – Implementação do método para gravar a observação da alteração da situação do requerimento

No método da Listagem 11 está a implementação de uma validação por meio da anotação disponibilizada pelo Spring-Security *@Secured*. Caso um usuário que não tenha permissão DERAC, não poderá ter acesso ao método "findByDerac()". Essa anotação é muito importante, pois evita que usuários mal-intencionados acessem determinados métodos os quais não deveriam.

```
444       @Secured("ROLE_DERAC")
445       @GetMapping(value = "/findToDerac")
446       @ResponseBody
447   public List<Requerimento> findByDerac() {
448            List<Requerimento> requerimentos =
requerimentoRepository.findAll(
449
Specification.where(RequerimentoSpecification.withStatus(StatusRequerimento
Enum.EM_ABERTO))
450
.or(RequerimentoSpecification.withStatus(StatusRequerimentoEnum.DEVOLVIDO_D
ERAC))
451            );
452   return requerimentos;
453       }
```

Listagem 11 – Implementação de validação de métodos através da anotação *@Secured*

Caso o usuário autenticado não tenha a devida permissão, ele é redirecionado para uma página exibindo o código 403, acesso negado. Uma parte da implementação dessa página pode ser conferida na Listagem 12.

```
<div class="middle-box text-center animated fadeInDown">
<h1>403</h1>
<h3 class="font-bold">Acesso Negado</h3>

<div class="error-desc">
${info}
</div>
</div>
```

Listagem 12 – Página 403-Acesso Negado

4 CONCLUSÃO

A modelagem e a implementação de um sistema *web* para gerenciamento de pedidos de requerimentosforam realizadas como planejado, atendendo os objetivos definidos para o trabalho. Para o desenvolvimento foram utilizadas várias 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 pedidos de requerimentos foi desenvolvido. O sistema permite o controle das etapas do processo em que o requerimento passa até sua finalização.

Como é um sistema para ambiente *web*, o acesso para que alunos realizem e gerenciam os requerimentos que fazem é facilitado. Com acesso exclusivo, cada aluno visualiza e manipula apenas os seus requerimentos. Um ator, denominado DERAC, realiza as aprovações dos requerimentos encaminhando-os para os respectivos departamentos, quando necessário.

Em decorrência da quantidade e diversidade de tecnologias disponibilizadas para a implementação de aplicações, é notável que o desenvolvimento se tornou 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 à 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 de utilizar os recursos que cada tecnologia oferece e avaliar os mais adequados para resolver o problema, considerando a relação custo x benefício. O custo pode se financeiro, como a aquisição de tecnologias, de processamento, de hardware e outros. E entre os benefícios, além da solução mais adequada para o problema, estão facilidade de uso e agilidade na realização das operações, entre outros.

Quando tecnologias gratuitas sãoutilizadas, 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* Spring-Boot 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 seu funcionamento foram identificadas várias 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* as quais 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.

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

HIBERNATE. **Framework ORM**. Disponível em: <http://hibernate.org/orm/>. Acesso em: 30 nov. 2017.

JQUERY. **Biblioteca jQuery**. Disponível em: <https://jquery.com/>. Acesso em: 30 nov. 2017.

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

MYSQL. **Introdução ao MySQL**. Disponível em: <https://www.devmedia.com.br/introducao-ao-mysql/27799>. Acesso em: 30 nov. 2017.

SPRING-BOOT. **Framework Java**. Disponível em: <https://projects.spring.io/spring-boot/>. Acesso em: 30 nov. 2017.

WILDFLY. **Servidor de aplicação**. Disponível em: <http://wildfly.org/about/>. Acesso em: 30 nov. 2017.

APÊNDICE A – Configuração das propriedades da conexão com o banco de dados e do JSP

application.properties: Arquivo de propriedades dosistema o qual é possível definir as configurações dobanco de dados e as configurações dos arquivos JSP.

```
spring.servlet.multipart.enabled=true
spring.servlet.multipart.max-file-size=2MB
spring.servlet.multipart.max-request-size=10MB

# http://docs.spring.io/spring-boot/docs/current/reference/html/boot-features-sql.html#boot-features-connecting-to-a-jndi-datasource
# DataSource settings: set here your own configurations for the database
# connection. In this example we have "netgloo_blog" as database name and
# "root" as username and password.
#spring.datasource.url = jdbc:mysql://localhost:8889/database
#spring.datasource.username = root
#spring.datasource.password = root
#no arquivo standalone.xml do Servidor Wildfly deve informar o datasource e o driver
spring.datasource.jndi-name=java:/MySqlDS
#spring.datasource.username=username
#spring.datasource.password=your pass
#spring.datasource.driver-class-name=com.mysql.jdbc.Driver
# Keep the connection alive if idle for a long time (needed in production)
spring.datasource.testWhileIdle=true
spring.datasource.validationQuery=SELECT 1

# Show or not log for each sql query
spring.jpa.show-sql=true

# Hibernate ddl auto (create, create-drop, update)
spring.jpa.hibernate.ddl-auto=update

# Naming strategy
spring.jpa.hibernate.naming-strategy=org.hibernate.cfg.ImprovedNamingStrategy

# Use spring.jpa.properties.* for Hibernate native properties (the prefix is
# stripped before adding them to the entity manager)

# The SQL dialect makes Hibernate generate better SQL for the chosen database
#
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL5Dialect
spring.jpa.database-platform=org.hibernate.dialect.MySQL5Dialect

# Importante informar o prefix pois vai ser onde ao retornar o nome da página no Controller o spring vai procurar nesta pasta informada
#nesta configuração, ou seja, no controller retornar "requerimento/form", ele vai procurar isso em /WEB-INF/pages/requerimento/formConfirmaDados.jsp
spring.mvc.view.prefix=/WEB-INF/pages/
spring.mvc.view.suffix=.jsp
```

APÊNDICE B – Configuração do Spring-Boot

MainSpringApplication.java: Classe principal dosistema para que o Spring-Boot reconheça as anotações utilizadas nas classes.

```
@SpringBootApplication
public class MainSpringApplication extends SpringBootServletInitializer {

public static void main(String[] args) {
      SpringApplication.run(MainSpringApplication.class, args);
   }

@Override
protected SpringApplicationBuilder configure(SpringApplicationBuilder
application) {
return application.sources(MainSpringApplication.class);
   }

}
```

APÊNDICE C – Configuração do Spring Security

WebSecurityConfig.java: Classe de configuração doSpring-Security, onde são definidas as permissões.

```
@EnableWebSecurity
@EnableGlobalMethodSecurity(securedEnabled = true)
public class WebSecurityConfig extends WebSecurityConfigurerAdapter {

@Autowired
private UsuarioRepository usuarioRepository;

@Override
protected void configure(HttpSecurity http) throws Exception {
        http.csrf().disable()
                .authorizeRequests()
                .antMatchers("/usuario/**").permitAll()
                .antMatchers("/login/**").permitAll()
.antMatchers("/index/**").hasAnyRole("ALUNO,PROFESSOR,DERAC,COORDENACAO")
.antMatchers("/requerimento/**").hasAnyRole("ALUNO,PROFESSOR,DERAC,COORDENA
CAO")
.antMatchers("/**").hasAnyRole("ALUNO,PROFESSOR,DERAC,COORDENACAO")
                .anyRequest().authenticated()
                .and()
                .formLogin()
                    .loginPage("/login")
                    .defaultSuccessUrl("/index/")
                    .failureUrl("/login?error=bad_credentials")
                    .permitAll()
                .and()
                .exceptionHandling().accessDeniedPage("/erro403").and()
                .logout()
                    .permitAll();
    }

@Bean
public UserDetailsService userDetailsService() {
return new UserDetailsServiceImpl(usuarioRepository);
    }

@Bean
public PasswordEncoder passwordEncoder(){
return new BCryptPasswordEncoder(10);
    }

@Override
protected void configure(AuthenticationManagerBuilder auth) throws
Exception {

auth.userDetailsService(userDetailsService()).passwordEncoder(passwordEncod
er());
    }

@Override
public void configure(WebSecurity web) throws Exception {
        web.ignoring().antMatchers("/webjars/**");
        web.ignoring().antMatchers("/resources/**");
    }
}
```