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

DOUGLAS RAFAEL MARTINS BANDEIRA

SISTEMA WEB PARA GERENCIAMENTO DE PROJETOS ACADÊMICOS

TRABALHO DE CONCLUSÃO DE CURSO

PATO BRANCO
2012

DOUGLAS RAFAEL MARTINS BANDEIRA

SISTEMA WEB PARA GERENCIAMENTO DE PROJETOS ACADÊMICOS

Trabalho de Conclusão de Curso de graduação, apresentado à disciplina de Trabalho de Diplomação, 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.

Orientadora: Profa. Beatriz Terezinha Borsoi

PATO BRANCO
2012

ATA Nº: **195**

DEFESA PÚBLICA DO TRABALHO DE DIPLOMAÇÃO DO ALUNO **DOUGLAS RAFAEL MARTINS BANDEIRA**.

Às 08:30 hrs do dia 9 de outubro de 2012, Bloco S da UTFPR, Câmpus Pato Branco, reuniu-se a banca avaliadora composta pelos professores Beatriz Terezinha Borsoi (Orientadora), Soelaine Rodrigues Ascari (Convidada) e Maico Fernando Wilges Carneiro (Convidado), para avaliar o Trabalho de Diplomação do aluno Douglas Rafael Martins Bandeira, matrícula 1116665, sob o título **Sistema Web para Gerenciamento de Projetos Acadêmicos**; como requisito final para a conclusão da disciplina Trabalho de Diplomação 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 09:20 hrs foi encerrada a sessão.

Profa. Beatriz Terezinha Borsoi, Dr.
Orientadora

Profa. Soelaine Rodrigues Ascari, M.Sc.
Convidada

Prof. Maico Fernando Wilges Carneiro, Esp.
Convidado

Prof. Omero Francisco Bertol, M.Sc.
Coordenador do Trabalho de Diplomação

Prof. Edilson Pontarolo, Dr.
Coordenador do Curso

RESUMO

BANDEIRA, Douglas Rafael Martins. Sistema *web* para gerenciamento de projetos acadêmicos. 2012. 81 f. Trabalho de conclusão de curso (graduação de Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas) - Universidade Tecnológica Federal do Paraná. Pato Branco, 2012.

A Universidade Tecnológica Federal do Paraná (UTFPR), como geralmente ocorre com instituições de ensino superior, está fundamentada nas áreas de ensino, pesquisa e extensão. É comum que as atividades caracterizadas como pesquisa e extensão e mesmo de ensino sejam realizadas por meio de projetos, que podem combinar essas áreas. Considerando que os projetos podem envolver professores, alunos e a comunidade, verificou-se que seria importante e mesmo necessário contar com uma forma sistematizada de gerenciamento desses projetos. Esse gerenciamento se refere ao armazenamento de dados e de obter informações sobre os projetos, as atividades realizadas e os professores, acadêmicos e a comunidade (pessoas) envolvidos. O sistema foi implementado a partir de interesses da área de Informática da UTFPR, Câmpus Pato Branco, mas o mesmo pode ser utilizado por outras áreas e instituições. A implementação apresentada neste trabalho visa mostrar a forma de uso da tecnologia Java, agregada por Java ServerPages e Ajax, no desenvolvimento de uma aplicação *web.*

Palavras-chave: Java para *web*. Ajax. Aplicações *web*.

ABSTRACT

BANDEIRA, Douglas Rafael Martins. *Web* system to manager academic projects. 2012. 81 f. Trabalho de conclusão de curso (graduação de Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas) - Universidade Tecnológica Federal do Paraná. Pato Branco, 2012.

The Universidade Tecnológica Federal do Paraná (UTFPR), as usually occurs with higher education institutions, its activities are focused teaching, research and extension. It is common for activities characterized as research and extension to be defined through projects and combine these areas. Considering that projects may involve teachers, students and the community, it was found that would be important and even necessary to have a systematic way of managing these projects. This management refers to the possibility of obtaining information about projects, activities, teachers, academics and community involved. The system was implemented from interests in the area of informatics to the UTFPR Campus Pato Branco, but it can be used in other areas and institutions. The implementation presented here aims to show how to use Java technology aggregated by Java ServerPages and Ajax in the development of a web application.

Keywords: Java web. Ajax. Web appliations.

LISTA DE ABREVIATURAS E SIGLAS

ACID	Atomicidade, Consistência, Isolamento, Durabilidade
AJAX	*Asynchronous Javascript And XML*
API	*Application Programming Interface*
ASP	*Active Server Page*
CGI	*Common Gateway Interface*
CRUD	*Create, Retrieve, Update, Delete*
CSS	*Cascading Style Sheets*
DB	*Data Base*
DHTML	*Dymanic HTML*
DOM	*Document Object Model*
EJB	*Enterprise JavaBeans*
GWT	*Google Web Toolkit*
HTML	*HyperText Markup Language*
HTTP	*Hypertext Transfer Protocol*
HTTPS	*Hypertext Transfer Protocol Secure*
IDE	*Integrated Development Environment*
IIS	*Internet Information Services*
IP	*Internet Protocol*
JDBC	*Java Database Connectivity*
JNDI	*Java Naming and Directory Interface*
JSF	*JavaServer Faces*
JSP	*Java Server Pages*
ODBC	*Open Data Base Connectivity*
OLEDB	*Object Linking and Embedding Data Base*
PHP	*PHP Hypertext Preprocessor*
RDF	*Resource Description Framework*
RIA	*Rich Internet Application*
SGBD	Sistema de Gerenciamento de Banco de Dados
SQL	*Structured Query Language*
UML	*Unified Modeling Language*
URL	*Uniform Resource Locator*
UTFPR	Universidade Tecnológica Federal do Paraná
W3C	*World Wide Web Consortium*
XHR	*XMLHttpRequest object*
XHTML	*eXtensible HTML*
XML	*Extensible Markup Language*
XSTL	*Extensible Style Sheet Language*

LISTA DE FIGURAS

LISTAGENS DE CÓDIGO

LISTA DE QUADROS

SUMÁRIO

1 INTRODUÇÃO

Este capítulo apresenta as considerações iniciais do trabalho, com o contexto de aplicação do sistema desenvolvido, os seus objetivos a justificativa e a organização do texto.

1.1 CONSIDERAÇÕES INICIAIS

As atividades acadêmicas de uma Universidade estão baseadas em ensino, pesquisa e extensão. Essas três áreas estão geralmente relacionadas. A pesquisa, por exemplo, está vinculada com o ensino, no sentido de contribuir para a melhoria do conhecimento que é repassado aos alunos, e com a extensão, pelo compartilhamento do conhecimento produzido com a comunidade externa.

As atividades de pesquisa e de extensão são, geralmente, desenvolvidas sob a forma de projetos. Nesses projetos podem estar envolvidos professores, alunos e pessoas da comunidade. Um projeto pode ter diversas atividades associadas. Um projeto de extensão, por exemplo, pode ser composto por cursos, oficinas e palestras. Assim, as atividades de um mesmo projeto podem ser ministradas e coordenadas por pessoas distintas e o seu público alvo também pode ser distinto para cada atividade.

Como forma de facilitar o gerenciamento dos projetos realizados na área de Informática da UTFPR (Universidade Tecnológica Federal do Paraná), Câmpus Pato Branco, verificou-se a possibilidade de desenvolver um sistema computacional que pudesse auxiliar no controle das atividades realizadas e no registro da participação dos acadêmicos e professores. Além dos projetos, também verificou-se importante uma forma de gerenciamento dos computadores em situação anti-econômica que são recebidos de empresas ou provenientes da própria Universidade e são recompostos e doados para Instituições de Ensino e beneficentes. Para facilitar o acesso e uso do aplicativo considerou-se importante que o sistema fosse para ambiente Internet, ou seja, um sistema *web*.

Ressalta-se que a recomposição de computadores é uma atividade que está vinculada a um projeto, contudo, há necessidade de registro de diversos dados específicos relacionados à recomposição que não são os mesmos das outras atividades. Requerendo, assim, formulários específicos para registro tanto da recomposição como da doação (o encaminhamento) dos computadores recompostos.

Dentre as possibilidades de desenvolvimento para ambiente Internet, as denominadas aplicações *web*, estão as *web* tradicionais (baseadas em hipertexto), as centradas em Ajax e mais recentemente as aplicações Internet ricas. A linguagem Java, juntamente com JSP (*Java Server Pages*) e uso de Ajax foram escolhidas pelos recursos que oferecem para implementar as funcionalidades pretendidas para o aplicativo que possibilitará a área de informática do Câmpus Pato Branco fazer o gerenciamento dos seus projetos e atividades relacionadas, incluindo a recomposição de computadores.

1.2 OBJETIVOS

O objetivo geral se refere ao resultado do trabalho que é o desenvolvimento de um aplicativo *web*. Os objetivos específicos explicitam complementações ao objetivo geral, tanto em termos das tecnologias utilizadas como da finalidade da aplicação.

1.2.1 Objetivo geral

Desenvolver um aplicativo *web* para o gerenciamento de projetos acadêmicos.

1.2.2 Objetivos específicos

- Prover uma forma de áreas ou coordenações acadêmicas gerenciarem os seus projetos de ensino, pesquisa e extensão.
- Prover uma forma de facilitar o controle de participação de acadêmicos e professores vinculados a projetos.
- Implementar uma forma de controle de recomposição de computadores destinados para doação.
- Discutir comparativamente, do ponto de vista do programador, a forma de implementação de aplicações *web* ricas utilizando C# e Java.

1.3 JUSTIFICATIVA

O desenvolvimento do aplicativo proposto, além de contribuir para o gerenciamento dos projetos da Área de Informática da UTFPR (pode aplicar-se a outros cursos e coordenações ou mesmo Instituições de Ensino), é uma oportunidade de aprendizado de como implementar uma aplicação *web* utilizando as tecnologias Java, JSP e Ajax.

No sistema desenvolvido, o gerenciamento dos projetos está centrado em atividades que são realizadas. Uma atividade pode ser um curso, uma oficina, uma palestra, dentre outros e está sempre associada a um projeto. Embora um projeto possa ter uma única atividade é comum que projetos possuam diversas atividades relacionadas. A frequência e o aproveitamento dos participantes são controlados por atividade. Assim, as atividades podem possuir critérios relacionados à frequência e ao aproveitamento para que comprovantes sejam emitidos.

A opção de desenvolvimento de um aplicativo *web* decorre da facilidade de acesso que o mesmo proporcionará, podendo ficar hospedado em servidor da própria Universidade. Existem diversas tecnologias para desenvolvimento para *web*. A escolha da linguagem Java como tecnologia decorre dos recursos que a mesma apresenta, destacando-se o uso da orientação e objetos, e pela aplicação poder ser implementada e executada utilizando tecnologias sem custo.

Justifica-se a ênfase deste trabalho de conclusão de curso estar na implementação do sistema, porque uma primeira versão da modelagem foi realizada como trabalho de estágio pelo autor deste trabalho de conclusão de curso. A modelagem foi revisada e complementada e a implementação foi continuada. A interface dos formulários implementados foi refeita. Além de serem implementadas as funcionalidades restantes.

1.4 ORGANIZAÇÃO DO TEXTO

Este texto está organizado em capítulos, dos quais este é o primeiro e apresenta a ideia do sistema, incluindo os objetivos e a justificativa de realização do trabalho.

O Capítulo 2 contém o referencial teórico, baseado em aplicações *web*. Inicialmente é feito um apanhado histórico das aplicações HTML (*HyperText Markup Language*) às RIAs (*Rich Internet Application*). Em seguida o texto se centra em Ajax por ser utilizado na implementação da aplicação desenvolvida como resultado deste trabalho.

No Capítulo 3 estão os materiais e o método utilizados. Os materiais se referem às ferramentas, linguagens e demais recursos necessários para implementar o sistema. O método contém as etapas necessárias para realizar o trabalho.

O Capítulo 4 contém o resultado deste trabalho. Esse resultado se refere à modelagem do sistema, a descrição do sistema por meio das suas telas e a codificação, que é representada por meio de exemplos de funcionalidades implementadas.

No Capítulo 5 está a conclusão. E por fim estão as referências bibliográficas utilizadas na elaboração do trabalho.

2 APLICAÇÕES WEB

Este capítulo apresenta o referencial teórico do trabalho. O capítulo inicia com um contexto de evolução das aplicações para Internet, abrangendo do surgimento de páginas hipertexto estáticas até aplicações interativas dinâmicas. Em seguida é apresentado sobre aplicações *web* com Ajax por ser a maneira utilizada na implementação das funcionalidades do sistema desenvolvido como resultado deste trabalho.

2.1 DOS SITES WEB HIPERTEXTO ÀS APLICAÇÕES INTERNET RICAS

As referências aos primeiros computadores estão, geralmente, relacionadas aos programas que executavam em equipamentos de grande porte, os denominados *mainframes*, aos quais estavam conectados terminais sem processamento, com interface caractere e interação por meio de linhas de comando. Um avanço considerável surge com os computadores pessoais. Não mais havia necessidade de conexão com um servidor central, os terminais possuíam capacidade de processamento e aplicativos podiam ser instalados no próprio computador. É a era dos computadores pessoais.

Os computadores pessoais permitem que os próprios usuários instalem aplicativos distintos e assim houve a necessidade de facilitar tanto a instalação como o uso dos aplicativos. Aplicativos com interface gráfica e instalação baseada em cliques em botões se tornaram quase que um padrão de fato. Os computadores pessoais trouxeram mais facilidade aos usuários, que se tornaram mais exigentes em termos de funcionalidades e facilidades dos aplicativos. Com muitos aplicativos sendo instalados nos computadores, alguns desnecessariamente e outros subutilizados, surgem problemas como os relacionados ao espaço de armazenamento, à velocidade de processamento e à necessidade de manutenção dos aplicativos.

Contudo, como forma de otimizar recursos e facilitar a manutenção em termos de *software*, além dos aplicativos instalados localmente, os computadores em rede podem ter aplicativos de uso compartilhados. Essa é a caracterização básica das aplicações denominadas cliente/servidor. Essa arquitetura possibilitou manter a capacidade de processamento nos computadores locais e o compartilhamento de aplicativos e dados. Apesar da evolução que esse tipo de arquitetura trouxe em relação aos *mainframes*, ela pode ter o inconveniente de

agregar custos e tempo quando o aplicativo precisa ser instalado em cada cliente, que são os computadores ligados ao servidor.

Computadores conectados em rede com acesso a aplicativos que não precisam estar instalados no computador cliente é o formato possibilitado pela Internet. Ressalta-se que esse formato também é possibilitado pela arquitetura cliente/servidor. Porém, a Internet facilita a conexão de computadores em rede, seja pela forma de conexão (protocolo HTTP (*Hypertext Transfer Protocol*)), seja pela sua abrangência (quantidade de computadores conectados e amplitude geográfica).

A *web* inseriu três conceitos fundamentais na computação cliente/servidor (JAZAYERI, 2007): um método de nomear e referenciar documentos (URL - *Uniform Resource Locator*), uma linguagem para escrever documentos que podem conter dados e *links* para outros documentos (HTML) e um protocolo de comunicação entre as máquinas cliente e servidor (HTTP).

A interface das páginas *web* tem como padrão de desenvolvimento a HTML, que é uma linguagem para marcação de hipertexto. Contudo, outras tecnologias podem ser utilizadas como, por exemplo, Adobe Flex (ADOBE, 2012) que executa em uma Flash Player, Microsoft Silverlight (MACDONALD, 2008), *applets* Java e JavaFX (JAVAFX, 2012), Openlaszlo (OPENLASZLO, 2012) e GWT (*Google Web Toolkit*) (GWT, 2012).

Mikkonen e Taivalsaari (2008) resumem a evolução da *web* em alguns marcos. Inicialmente as páginas *web* eram praticamente documentos de texto simples que ofereciam capacidade de interação com o usuário limitada. Em seguida, as páginas passaram a oferecer suporte gráfico e dados baseados em formulários. Gradualmente, com a inserção de DHTML (*Dynamic* HTML), foi possível criar páginas *web* interativas e com interface gráfica melhorada e animações. Atualmente, as tecnologias *web* 2.0 estão associadas com sistemas como Ajax, Ruby on Rails e GWT. Contudo, Hendler (2009) já aponta para a *web* 3.0 cuja base das aplicações reside em RDF (*Resource Description Framework*) que provê meios para vincular dados de múltiplos sites *web* e bases de dados.

De acordo com Mikkonen e Taivalsaari (2008), os componentes fundamentais de páginas *web* incluem: HTML, CSS (*Cascading Style Sheets*), linguagem de *script* como JavaScript e DOM (*Document Object Model*). Em que (GARRETT, 2005; ZEPEDA, CHAPA, 2007; PAULSON, 2005):

a) HTML é uma linguagem de marcação para a criação de páginas *web*. Essa linguagem provê um meio para descrever a estrutura de informação baseada em texto em um

documento. Isso é feito pela definição de tipos de formatação de texto para cabeçalho, parágrafos, listas e outros; e pela inclusão de imagens e outros objetos.

b) CSS é uma linguagem de estilo que é usada para descrever aspectos de apresentação de um documento escrito em uma linguagem de marcação. Um estilo permite definir regras de formatação para uma página *web*. Assim, cores e fontes, por exemplo, podem ser definidas independentemente do conteúdo da página. CSS é usado para criar folhas de estilo que definem como diferentes elementos da página, tais como cabeçalho e *links* são mostrados. Várias folhas de estilo podem ser aplicadas em uma mesma página *web*.

c) JavaScript possibilita a execução de código em um documento *web*. JavaScript é uma linguagem de *script*, caracterizada por suas instruções serem interpretadas e com execução vinculada a outras linguagens.

d) DOM é uma forma independente de plataforma para representar uma coleção de objetos que constituem uma página em um navegador *web*. DOM atua como uma interface entre o navegador *web* e as linguagens de *script*, permitindo comunicação entre os mesmos. DOM define os atributos associados com cada objeto, bem como as formas nas quais os usuários podem interagir com os objetos, tornando as páginas *web* mais interativas. DHTML atua juntamente com DOM para alterar dinamicamente a aparência de páginas *web*.

Com a expansão do uso da Internet para aplicações comerciais, o formato baseado em hipertexto começou a apresentar limitações. Os usuários estavam acostumados a aplicativos com formas bem diferentes de interação que as proporcionadas por *links*, como botões e menus diferenciados e efeitos como o de arrastar-e-soltar. Tecnologias, incluindo linguagens de programação e de formatação, surgiram como forma de melhorar a interatividade dos usuários com os aplicativos *web*. Para prover aplicações interativas, como proporcionado pelas aplicações cliente/servidor, mas que fossem disponibilizadas na *web*, surge o Ajax (GARRETT, 2005).

2.2 AJAX

Ajax inicialmente estava associado ao acrônimo *Asynchronous Javascript And* XML (*Extensible Markup Language*), como denominado pelo seu criador Jesse James Garrett para definir o objeto XMLHttpRequest. Atualmente Ajax se refere a um conjunto de técnicas (JAZAYERI, 2007). Para Zepeda e Chapa (2007), Ajax é uma nova forma de pensar, projetar, implementar e um novo estilo para programar aplicações *web*.

Com o uso de Ajax, ao finalizar o preenchimento de um campo de um formulário, por exemplo, o mesmo é validado e somente a parte da tela correspondente a esse campo é atualizada. Assim, o tráfego na rede é reduzido porque é menor a quantidade de dados que trafegam entre o cliente e o servidor.

Aplicações Ajax possuem vantagens em relação às aplicações *desktop* porque elas são independentes de (ZEPEDA, CHAPA, 2007): sistema operacional específico, máquina virtual, *plugins* no navegador *web* ou programas externos instalados na máquina cliente. Essas aplicações somente necessitam de um navegador *web* com acesso à Internet e oferecem suporte para interações com o usuário.

As tecnologias Ajax são utilizadas para desenvolver aplicações *web* com desempenho melhorado e com interface que provê recursos para facilitar a interatividade. Essas aplicações oferecem funcionalidades geralmente disponíveis em software *desktop* (PAULSON, 2005).

Ajax também pode ser visto como uma técnica que usa um conjunto de tecnologias de padrão aberto e conhecido, com suporte para compatibilidade entre navegadores *web* e plataformas distintas. Essas tecnologias trabalham juntas em diferentes níveis, cada qual com uma funcionalidade específica para criar um modelo de desenvolvimento para aplicações *web* (HEWITT, 2007). De acordo com Garrett (2005), Zepeda e Chapa (2007) e Paulson (2005), Ajax incorpora:

a) Linguagem de marcação de hipertexto, como HTML e XHTML (*eXtensible* HTML). XHTML provê páginas *web* mais dinâmicas que HTML. Por exemplo, quando o cursor passa sobre uma página DHTML, a cor da fonte pode ser alterada ou a fonte do referido texto ter seu tamanho aumentado, além de um usuário poder arrastar-e-soltar imagens.

b) Linguagem para descrever a apresentação de um documento: CSS.

c) Visualização e interação dinâmicas: DOM.

d) Troca e manipulação de dados: XML, XSTL (*Extensible Style Sheet Language*) e texto não formatado podem ser utilizados para codificar dados para transferi-los entre o servidor e o navegador no cliente.

e) Recuperação assíncrona de dados usando *XMLHttpRequest* que possibilita comunicação com o servidor sem necessidade de redesenhar a tela inteira a cada interação do usuário: XMLHttpRequest object (XHR).

f) Linguagem e possibilidade de análise de dados do lado cliente: JavaScript para agrupar as tecnologias e no navegador para modificar diretamente a interface com o usuário. JavaScript é uma linguagem de *script* independente de plataforma.

g) Protocolo de transferência: HTTP, HTTPS (*Hypertext Transfer Protocol Secure*).

h) Linguagens do lado servidor: JSP, ASP (*Active Server Page)*, JSF (*JavaServer Faces)* e aplicações CGI (*Common Gateway Interface)* são exemplos dessas linguagens.

Ajax atua como uma camada entre a interação do usuário e a comunicação com o servidor. Assim, JavaScripts são executados no navegador e ações são delegadas ao servidor enquanto no cliente são manipulados dados do cliente ou provenientes do servidor. A forma assíncrona de comunicação possibilita que chamadas sejam enviadas para o servidor e enquanto esse as processa, o usuário pode continuar interagindo com a aplicação.

Atualmente há tecnologias ou abordagens utilizadas no lugar de JavaScript como componente do Ajax. Dentre as quais estão o Adobe Flex, o Microsoft Silverlight e o *Google Web Toolkit*, por exemplo, que usam linguagem diferente do JavaScript para os *scripts* que estão no ambiente do *browser.*

Ajax passou a ser definido como uma aplicação desenvolvida para a *web* usando DHTML com comunicação assíncrona com o servidor. DHTML combina a linguagem estática do tipo marcação (como o HTML), com uma linguagem tipo *script* (como, por exemplo, o JavaScript) no lado cliente, uma linguagem de apresentação (como CSS) e DOM.

2.2.1 Comunicação síncrona e assíncrona em páginas/aplicações web

Uma aplicação *web* permite que funções de processamento de dados sejam iniciadas remotamente por um navegador *web* (*browser*) e executadas parcialmente em um servidor *web*, servidor de aplicação e/ou servidor de base de dados (BRUNO, TAM, THOM, 2005). As aplicações *web* têm sido especificamente projetadas para serem executadas em um ambiente baseado em *web* (FINKELSTEIN et al., 2002), significando mais do que páginas *web* com *links* para navegação.

Uma forma de permitir aos navegadores *web* proverem operações com mais interação com os usuários é delegar parte da comunicação para o servidor enquanto o navegador provê a interatividade com o usuário. Em oposição à comunicação síncrona do HTTP original, Ajax usa um protocolo assíncrono. O navegador registra um conjunto de retornos de chamada para o servidor para atualizar os dados da página em segundo plano.

A Figura 1 representa a forma de comunicação das aplicações *web*, seja a comunicação síncrona ou assíncrona.

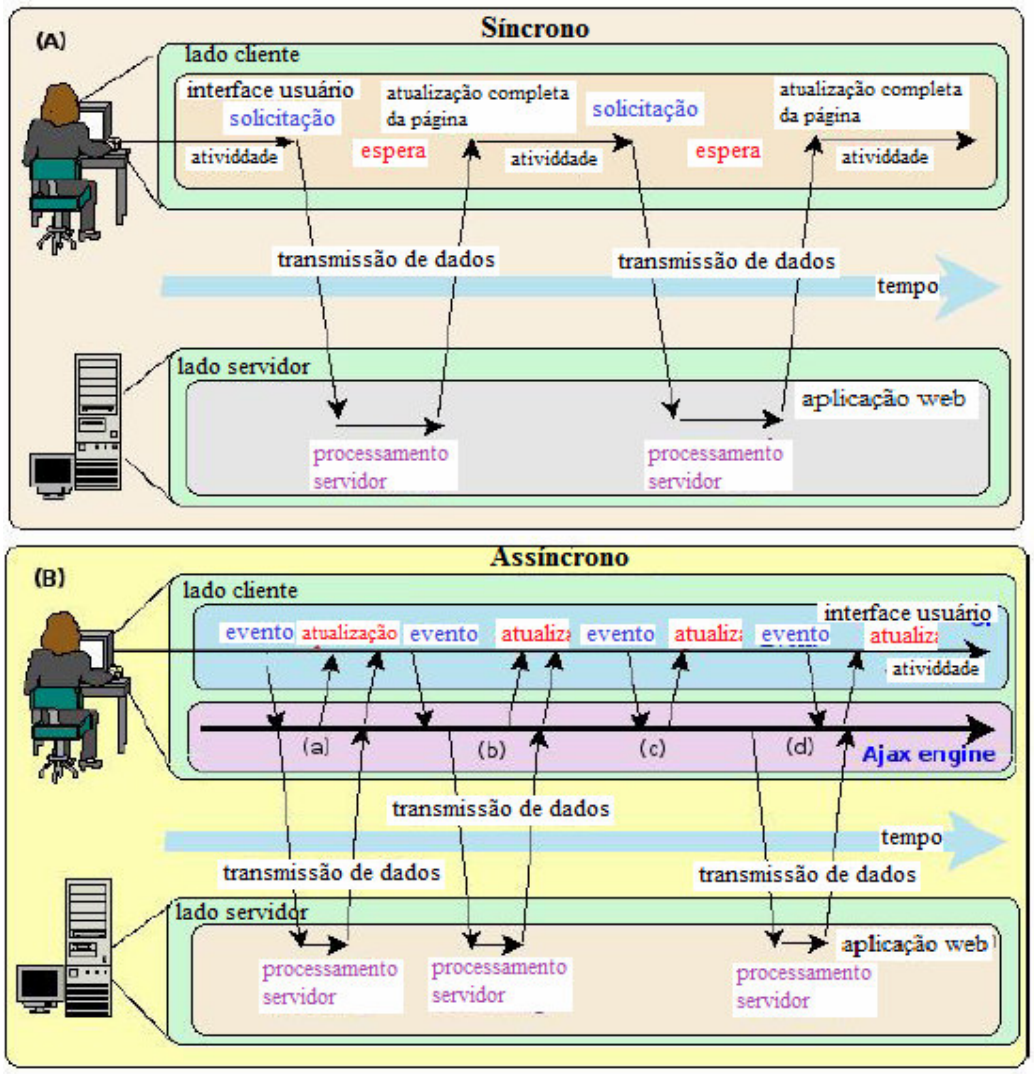

Figura 1 – Forma de interação síncrona e assíncrona

Fonte: traduzido de Zepeda e Chapa (2007, p. 194).

A parte superior da Figura 1, identificada como "(A)", representa a forma síncrona de comunicação. É a maneira clássica de comunicação *web* para páginas de hipertexto. As aplicações *web* clássicas requerem que o usuário envie uma requisição ao servidor por meio de um *link* ou formulário. Com isso o cliente precisa esperar pelo processamento no servidor. Se a geração ou atualização dinâmica de uma nova página é necessária, o servidor retorna essa página para o *browser* (cliente) e a página toda é atualizada com os novos resultados.

A Figura 1 (A) mostra como a atividade do usuário é interrompida constantemente e a atualização requer recarga completa da página *web* a cada atualização da página, mesmo que a alteração se refira a apenas um único campo de formulário (PAULSON, 2005). Esse modelo clássico cria problemas em aplicações *web* devido ao baixo desempenho, perda de estado, uso

excessivo de largura de banda, interatividade limitada e a transmissão de dados tem código redundante, que é desnecessário, em cada página (ZEPEDA, CHAPA, 2007). O padrão de requisição-espera-resposta ocasiona produtividade baixa (SMITH, 2006). E, ainda, frequentemente as páginas têm HTML, CSS e JavaScript mesclados em um mesmo arquivo, isso pode gerar dificuldade de manutenção (ZEPEDA, CHAPA, 2007).

O modelo representado na parte inferior da Figura 1, identificada como "(B)", consiste de interface *web* com páginas simples. Essa interface é composta por componentes que podem ser criados, atualizados, excluídos e substituídos independentemente (MESBAH, DEURSEN, 2007). O acesso a esses componentes é realizado por especificação DOM. A atualização pode ser realizada com pequenas partes de código XML por meio de uma máquina (*engine*) Ajax, sem recarregar uma nova página *web* a cada ação do usuário. A Figura 1 (B) mostra a forma de comunicação assíncrona com Ajax, que pode ser segmentada em quatro partes (ZEPEDA, CHAPA, 2007):

a) O usuário gera um evento para a máquina Ajax e envia uma requisição diretamente para o servidor e atualizações podem ser realizadas enquanto o servidor fornece uma resposta. Quando o servidor retorna os novos dados, o componente independente é atualizado.

b) O usuário gera um evento para a máquina Ajax e envia uma requisição indiretamente para o servidor. Em seguida, o processamento é o mesmo que o mostrado em (A) na Figura 1.

c) O usuário gera um evento, mas não é necessário enviar uma requisição para o servidor. Código em JavaScript, por exemplo, atualiza os componentes com CSS, DOM e dados da mesma página por meio de funções implementadas na própria página cliente.

d) Antes de o usuário gerar um evento específico, a máquina Ajax envia uma requisição para o servidor, quando o usuário realiza um evento específico, a informação será imediatamente atualizada.

A máquina Ajax possibilita a comunicação em segundo plano com o servidor de forma assíncrona e envia e recebe dados para processamento a partir do lado servidor. Ajax elimina a constante recarga de páginas *web* inteiras em cada interação do usuário, como ocorre no modelo de aplicações *web* clássicas. Todas as interações do usuário são realizadas diretamente dentro do navegador (ZEPEDA, CHAPA, 2007).

3 MATERIAIS E MÉTODO

Este capítulo apresenta os materiais e o método utilizados para a modelagem e a implementação do sistema. Os materiais se referem às ferramentas e às tecnologias e incluem linguagem de programação, banco de dados, interface de desenvolvimento e editores utilizados para a modelagem do sistema. O método se refere às principais atividades realizadas para obter o resultado do trabalho.

3.1 MATERIAIS

Para a modelagem e a implementação do sistema foram utilizadas as seguintes ferramentas e tecnologias:

a) Astah Community para modelagem do sistema (ASTAH, 2012);

b) IbExpert para administração do banco de dados (IBEXPERT, 2012);

c) NetBeans versão 7.0.1 como IDE (*Integrated Development Environment*) de desenvolvimento (NETBEANS, 2012);

d) Java como linguagem de programação, incluindo XHTML, DHTML, CSS, JavaScript, DOM, Ajax para validações de formulários e jQuery na elaboração da interface;

e) Firebird para o banco de dados (CANTU, 2012);

f) Apache Tomcat para contêiner *web* (TOMCAT, 2012).

Nas subseções a seguir essas ferramentas e tecnologias são apresentadas, com exceção de Ajax e das tecnologias relacionadas ao mesmo que são HTML, DHTML, CSS, JavaScript e DOM. Elas foram apresentadas no Capítulo 2 do referencial teórico.

3.1.1 Astah Community

Astah Community (ASTAH, 2012) é uma ferramenta gratuita para modelagem de sistemas orientados a objeto. Essa ferramenta é baseada em diagramas definidos com a notação da UML 2.0 (*Unified Modeling Language*). A Figura 2 apresenta a interface principal dessa ferramenta que é basicamente composta por uma área de edição à direita, na qual são colocados os componentes gráficos para a composição dos diagramas. Esses componentes são pré-definidos e estão representados como ícones na barra imediatamente superior à área de

edição.

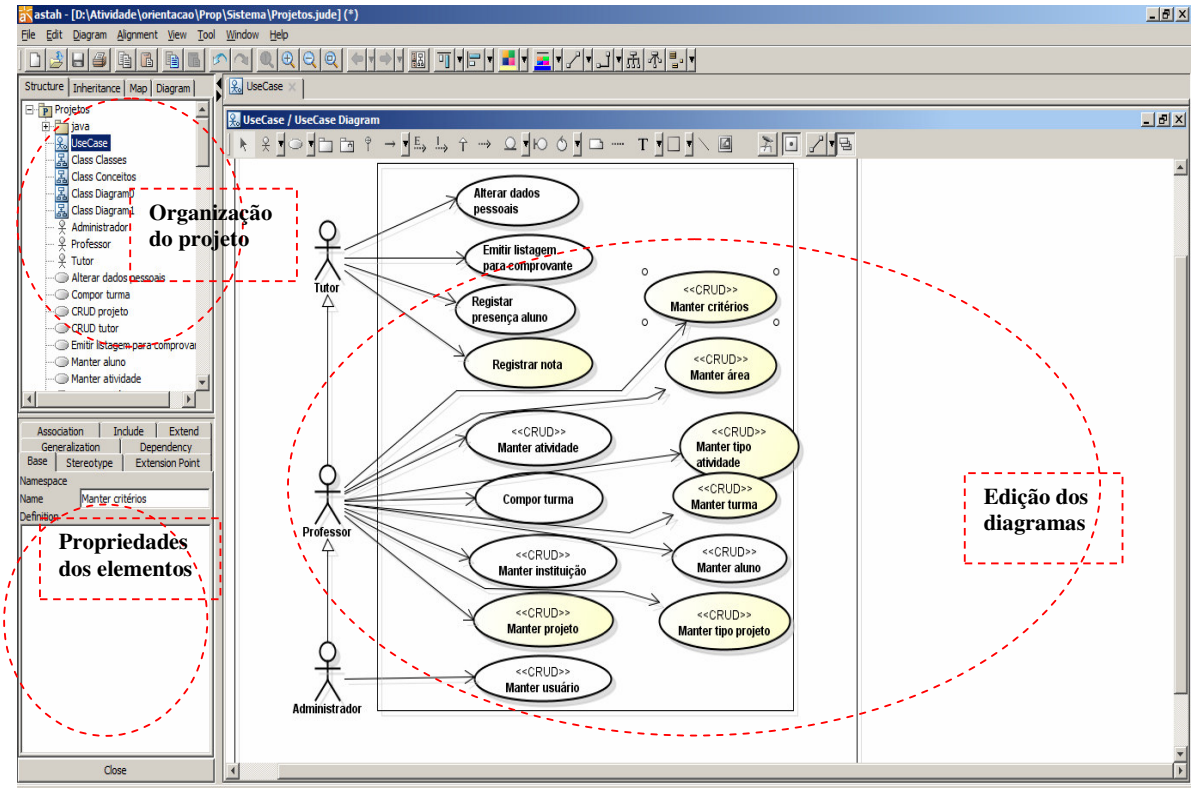

Figura 2 – Ferramenta de modelagem Astah

As partes principais da ferramenta Astah Community marcadas na Figura 2 são:

a) Organização do projeto - é a área que apresenta as visões do projeto que são: *structure* que é a árvore de estrutura do projeto composta por pastas, diagramas e elementos de diagramas; *inheritance* que exibe as heranças identificadas entre os elementos definidos; *map* que mostra todo o diagrama em edição e possibilita definir a área a ser visualizada; e *diagram* que apresenta a lista de diagramas do projeto em edição.

b) Propriedades dos elementos – nesta área são apresentadas as propriedades dos elementos do diagrama. As propriedades de um elemento selecionado são exibidas e podem ser alteradas nessa região.

c) Editor dos diagramas - é a área para composição dos diagramas. Ao ser selecionado um determinado diagrama, dos constantes na lista, o mesmo é carregado nesta área e os seus elementos são disponibilizados para edição. A barra de ferramentas que fica na parte superior dessa área apresenta os ícones para a composição de diagramas. Esses ícones são apresentados de acordo com o tipo de diagrama em edição.

3.1.2 IBExpert

IBExpert (IBEXPERT, 2012) é uma ferramenta para administração de bancos de dados Interbase e Firebird que permite criar e gerenciar usuários e tabelas. A interface do aplicativo consiste, basicamente, de uma barra de ferramentas e do DB (*Data Base*) Explorer. A Figura 3 apresenta a tela inicial da ferramenta IBExpert.

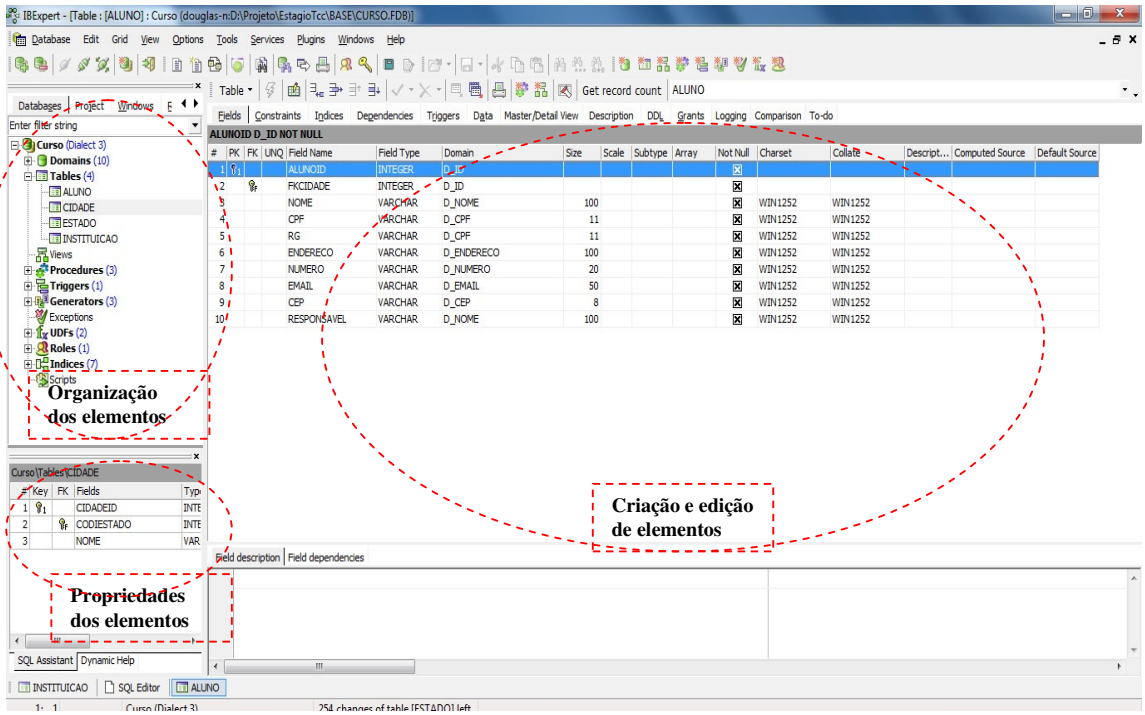

Figura 3 – Tela inicial do sistema gerenciador de banco de dados IBExpert

O sistema gerenciador de banco de dados IBExpert está organizado em três partes principais, além das barras de menus e de ferramentas que estão na parte superior da tela. Essas partes estão destacadas na Figura 3 e são:

a) Organização dos elementos – fornece uma visão dos bancos de dados armazendos e permite a seleção de um banco de dados para edição. O código de composição do banco de dados selecionado é apresentado na área de edição. Nessa área também é possível fazer alterações em banco de dados existentes, bem como criar tabelas de forma visual.

b) Propriedades dos elementos – nessa área da interface são apresentadas as propriedades dos componentes do banco de dados.

c) Criação e edição de elementos – essa área permite ao usuário criar tabelas por meio de código SQL (*Structured Query Language*), bem como inserir e editar dados e selecionar a visualização das tabelas e dos dados que estão armazenados nas tabelas.

A criação de um banco de dados é realizada pela opção *Create Database* do Menu *Database*. Para criar uma nova base de dados é necessário indicar:

a) *Server* – determina se o servidor é remoto ou local.

b) *Server Name* – se escolhida a opção remoto é necessário informar o nome ou IP (*Internet Protocol*) do computador servidor da base de dados.

c) *Protocol* – indica o protocolo usado para efetuar a comunicação com o servidor. É utilizado no caso de escolha da opção remoto.

d) *Database* – campo para indicar o caminho da base de dados seguido do seu nome e extensão.

e) *Client Library Name* – campo para informar a biblioteca (.dll) (*dymanic-link library*) do banco.

Os outros campos são usuário e senha do Firebird (SYSDBA e *masterkey,* respectivamente), tamanho da página do arquivo, o *charset* (por exemplo, WIN1252), o dialeto SQL e a opção para registrar o banco de dados após sua criação. O banco de dados criado precisa ser registrado e para isso é necessário informar a sua versão e o *alias*.

3.1.3 NetBeans

A IDE NetBeans (NETBEANS, 2012) é um ambiente de desenvolvimento multiplataforma e agrupa diversas funcionalidades necessárias para implementar um sistema. A IDE NetBeans possui um grande conjunto de bibliotecas, módulos e APIs (*Application Programming Interface*) que compõem, basicamente, um conjunto de rotinas, protocolos e ferramentas. Essa IDE possibilita desenvolver aplicativos para as plataformas *desktop*, *web* e *mobile*. A Figura 4 apresenta a tela principal da IDE NetBeans.

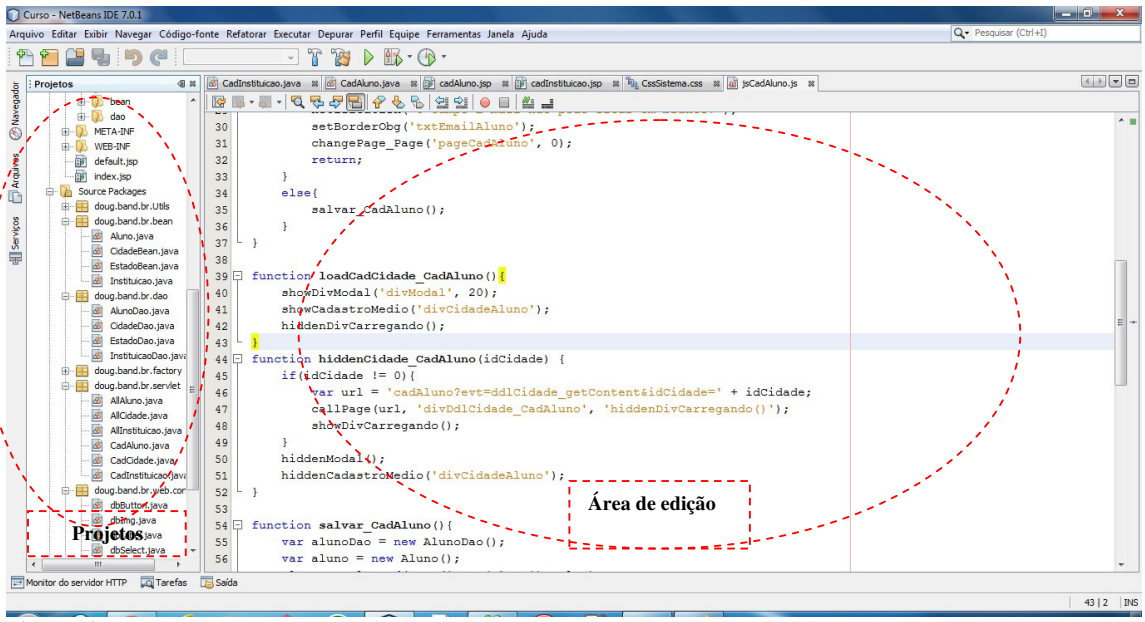

Figura 4 – IDE NetBeans

As partes destacadas na Figura 4 indicam:

a) Projetos – a área na qual são organizados os elementos (arquivos de código) definidos durante a implementação do projeto.

b) Área de edição – nesta área o código é editado. A interface do sistema pode ser composta por meio de componentes visuais vinculados à ferramenta.

3.1.4 Linguagem Java

Java é uma linguagem orientada a objetos, sendo assim, a maior parte dos elementos de um programa Java são objetos (DEITEL, DEITEL, 2005). Como exceção cita-se os tipos básicos, como o *int* e o *float*. O código é organizado em classes, que podem estabelecer relacionamentos de herança simples entre si. Um programa Java ao ser compilado gera um código intermediário denominado de *bytecode*. Esse *bytecode* é interpretado pelas máquinas virtuais Java que pode ser emulada em qualquer sistema operacional que suporte a linguagem C++.

3.1.5 jQuery

jQuery User Interface prove um conjunto de *plugins* de interação, *widgets* de interface e efeitos visuais. É uma arquitetura baseada em eventos e com foco em padrões *web*, acessibilidade e projeto facilitado (JQUERY, 2012).

jQuery é uma biblioteca *JavaScript* que facilita a identificação de *tags* em documentos HTML, a manipulação de eventos, o desenvolvimento de animações e de interações Ajax. Assim, jQuery torna o desenvolvimento *web* mais fácil. Essa biblioteca possui as seguintes características (SILVA, 2010):

a) uso de seletores CSS para localizar elementos componentes da estrutura de marcação HTML da página;

b) arquitetura compatível com instalação de *plugins* e extensões em geral;

c) indiferença às inconsistências de renderização entre navegadores;

d) interação implícita, isto é, não há necessidade de construção de estruturas de repetição para localização de elementos no documento;

e) programação encadeada, ou seja, cada método retorna um objeto;

f) é extensível, admite criação e inserção de novas funcionalidades em bibliotecas existentes.

jQuery permite adicionar interatividade e dinamismo às páginas *web*, facilitando prover usabilidade e acessibilidade. Isso ocorre porque jQuery possibilita: adicionar efeitos visuais e animações; acessar e manipular o DOM; buscar informações no servidor sem necessidade de recarregar a página; modificar apresentação e estilo da página; e simplificar tarefas específicas de JavaScript.

jQuery foi criada em conformidade com os padrões *web*. Isso significa que há compatibilidade entre sistemas operacionais, navegadores e suporte para as CSS3. Essa biblioteca está de acordo com o disposto pela W3C (*World Wide Web Consortium*), mas é responsabilidade do desenvolvedor escrever *scripts* em conformidade com essas diretrizes (SILVA, 2010).

3.1.6 Firebird

O banco de dados Firebird é derivado do código do Borland InterBase 6.0 e possui o código fonte aberto e gratuito. O Firebird é um SGBD (Sistema de Gerenciamento de Banco

de Dados) completo e possui suporte nativo para diversos sistemas operacionais, incluindo Windows, Linux, Solaris, MacOS, *triggers* de conexão e transação (CANTU, 2012). Dentre os recursos do Firebird destacam-se:

a) Transações compatíveis com ACID (Atomicidade, Consistência, Isolamento, Durabilidade);

b) Integridade referencial;

c) Utiliza poucos recursos de processamento;

d) Linguagem nativa para *stored procedures* e *triggers* (PSQL (PostgreSQL));

e) Suporte para funções externas;

f) Versão incorporada do SGBD;

g) Diversas formas de acesso ao banco de dados, como: nativo/API, dbExpress, ODBC (*Open Data Base Connectivity)*, OLEDB (*Object Linking and Embedding Data Base)*, .Net provider, JDBC (*Java Database Connectivity)* nativo tipo 4, Python *module*, PHP (*PHP Hypertext Preprocessor*), Perl;

h) Cópias de segurança incrementais;

i) Tabelas temporárias.

3.1.7 Apache Tomcat

O Tomcat (TOMCAT, 2012) é um contêiner *servlet,* ou seja, um servidor de aplicações utilizado para interpretar aplicações escritas em Java para *web*. Ele não implementa um contêiner EJB (*Enterprise JavaBeans*), mas abrange parte da especificação Java Enterprise Edition com tecnologias como *servlet* e JSP e de apoio relacionadas, como segurança, JNDI (*Java Naming and Directory Interface*) *Resources* (API para acesso a diretórios) e JDBC *DataSources*.

Tomcat pode atuar também como servidor *web* ou pode funcionar integrado a um servidor *web* dedicado como o Apache ou o IIS (*Internet Information Services*). Como servidor *web*, ele provê um servidor HTTP puramente em Java e inclui ferramentas para configuração e gerenciamento, o que também pode ser feito editando-se manualmente arquivos de configuração formatados em XML.

3.2 MÉTODO

O processo de modelagem e de desenvolvimento do aplicativo para gerenciamento de projetos tem como base o modelo sequencial linear como descrito em Pressman (2008), complementado pelo processo unificado (BLAHA et al., 2006). A modelagem e a implementação do sistema foram realizadas em etapas (iterações). Inicialmente foram definidos os requisitos principais, obtendo-se uma visão geral do sistema. Essa atividade foi realizada como trabalho de estágio supervisionado. Esses requisitos foram complementados à medida que o sistema era modelado e implementado.

O processo unificado auxiliou a definir ciclos interativos de modelagem e de implementação. No trabalho de estágio a implementação esteve centrada nas funcionalidades para facilitar e agilizar o desenvolvimento do sistema. No trabalho de conclusão de curso a ênfase da implementação está nas funcionalidades do sistema, nas regras de negócio. Essa forma de procedimento foi adotada inclusive porque os requisitos do sistema não estavam completamente definidos pelo usuário e era necessário estudar as tecnologias utilizadas.

O Quadro 1 apresenta os processos (fluxos de trabalho) e as iterações desenvolvidas como trabalho de conclusão de curso. Assim, revisão dos requisitos é atividade da primeira iteração.

Processos	1ª iteração	2ª iteração	3ª iteração	4ª iteração
Requisitos	Revisão dos requisitos	Complementação dos requisitos referentes à recomposição de equipamentos		
Análise e projeto	Modelagem das classes e tabelas restantes	Modelagem dos requisitos relacionados à recomposição	Revisão da modelagem	
Implementação	Implementação dos formulários	Redefinição da interface do sistema. Funcionalidades restantes do sistema implementadas	Implementação da listagem de alunos para emissão de comprovantes. Implementação de relatórios	Ajustes no sistema
Testes	Da listagem dos requisitos para verificar se as funcionalidades pretendidas são abrangidas pelos requisitos	De código realizado pelo autor do trabalho. De interface do sistema realizada pela orientadora	De código realizado pelo autor do trabalho	De código realizado pelo autor do trabalho. De usuário realizado pela orientadora

Quadro 1 – Iterações definidas

A seguir estão descritos os processos definidos para as atividades realizadas.

a) Requisitos

Uma primeira versão dos requisitos foi definida como trabalho de estágio.

O levantamento dos requisitos iniciou com a orientadora fornecendo a visão geral do sistema, com os principais conceitos envolvidos na definição das funcionalidades pretendidas para o sistema. Dessa visão foram extraídos os principais requisitos e outros foram identificados posteriormente. Os requisitos foram listados como funcionais e não funcionais. Os requisitos foram complementados à medida que as iterações ocorriam. Várias alterações ocorreram como forma de definir mais adequadamente as funcionalidades do sistema. Uma dessas alterações foi referente ao registro da chamada.

Na primeira iteração realizada como trabalho de conclusão de curso, a primeira atividade desenvolvida foi a revisão desses requisitos. Várias alterações ocorreram nos requisitos visando definir as funcionalidades do sistema da maneira a mais adequadamente atender aos objetivos. Uma dessas alterações foi referente ao registro da chamada. Decidiu-se que seriam registradas apenas as faltas dos alunos e não um sistema de chamada diária, por exemplo. Para a emissão dos certificados, que é realizada pela própria Universidade, é gerada uma listagem de alunos ordenados de maneira que sejam facilmente identificados os alunos que atendem e que não atendem os critérios de presença e de aproveitamento estabelecidos para a disciplina. Em uma interação posterior foram definidos os requisitos relacionados à recomposição de equipamentos.

Outros requisitos definidos foram os relacionados à recomposição de computadores. Foram definidos os requisitos funcionais e não funcionais, bem como as tabelas do banco de dados, para o controle dos equipamentos recebidos como doação e encaminhados para Instituições de ensino e beneficentes.

b) Análise e projeto do sistema

Com base nos requisitos organizados em funcionais e não funcionais foram definidos os casos de uso do sistema. Esses casos de uso foram documentados sob a forma de um diagrama e por meio de descrição textual. Dos casos de uso foram obtidas informações que auxiliaram a definir o banco de dados e orientar a definição das classes na implementação do sistema.

A partir dos requisitos foram definidos os campos para as telas de cadastro do sistema. Nessas listagens por formulário foram indicados os campos, o tipo de dados e a indicação se cada campo é obrigatório ou não.

A modelagem do banco de dados foi realizada por meio de um diagrama de entidades e relacionamentos e a descrição de cada uma das tabelas. Na descrição os campos e o tipo de dado foram identificados, bem como as chaves primárias e estrangeiras, se o campo poderia ser nulo, se havia um valor padrão para o respectivo campo e observações. Em observações foi documentada a tabela de origem dos campos que são chave estrangeira.

c) Implementação

A implementação foi realizada utilizando a IDE Netbeans. Como trabalho de estágio foram implementados três cadastros visando exemplificar o uso das tecnologias e o aprendizado das mesmas. A implementação esteve centrada em funções utilizadas posteriormente como padrão ou base para implementar o sistema, incluindo *stored procedur*es para armazenar no banco de dados. Em termos de interface, o objetivo foi experimentar e testar a melhor forma de compor os formulários e menus e disponibilizar as informações na tela.

Nas iterações realizadas no trabalho de conclusão de curso, os cadastros restantes e as funcionalidades (regras de negócio) foram implementadas. Na segunda iteração de implementação a interface do sistema foi alterada, os menus, os formulários e a forma de navegação foram redefinidos.

d) Testes

Em termos de testes foram realizados testes informais, sem um plano de testes específico, para verificação do código pelo autor deste trabalho e testes de interface e usabilidade do sistema pela orientadora.

4 RESULTADOS

Este capítulo apresenta o resultado da realização deste trabalho que é a implementação de um sistema para gerenciamento de projetos acadêmicos.

4.1 APRESENTAÇÃO DO SISTEMA

O sistema implementado tem como finalidade principal prover uma forma de gerenciamento e de controle de projetos realizados na área de informática da UTFPR, Câmpus Pato Branco. São projetos relacionados a ensino, pesquisa e extensão, embora outras categorizações possam ser definidas e cadastradas, como, por exemplo, grupos de estudo e projetos interdisciplinares. Apesar de o sistema ter sido definido tendo como base os interesses e as necessidades dessa área, o mesmo poderá ser utilizado por qualquer área e por áreas distintas de uma Instituição ou mesmo por outras instituições.

Os projetos são instanciados por meio de atividades vinculadas aos mesmos, como, por exemplo, curso e oficinas. Cada atividade pertence a uma área (como informática), possui um professor responsável, tempo de duração e data de início e de fim, dentre outros atributos. Funções (que representam papéis desempenhados por pessoas) estão associadas às atividades. Como exemplos de função estão: professor (responsável por projetos e atividades), tutor (quem realiza a atividade que pode ser ministrar o curso, realizar a palestra, realizar a oficina) e auxiliar (auxilia na realização da atividade). Os alunos que realizam as atividades são organizados em turmas, definindo uma lista de alunos. E cada turma está relacionada a uma atividade.

4.2 MODELAGEM DO SISTEMA

A Figura 5 apresenta uma visão geral do aplicativo desenvolvido. Essa visão define o sistema a partir de um conjunto de conceitos relacionados. Esses conceitos se referem às funcionalidades essenciais do sistema. Alguns desses conceitos serão classes, tabelas, atributos ou campos, outros auxiliam a definir requisitos funcionais e não funcionais. É uma visão do usuário e, portanto, desvinculada de aspectos técnicos ou tecnológicos de definição e de implementação do problema ou da sua solução.

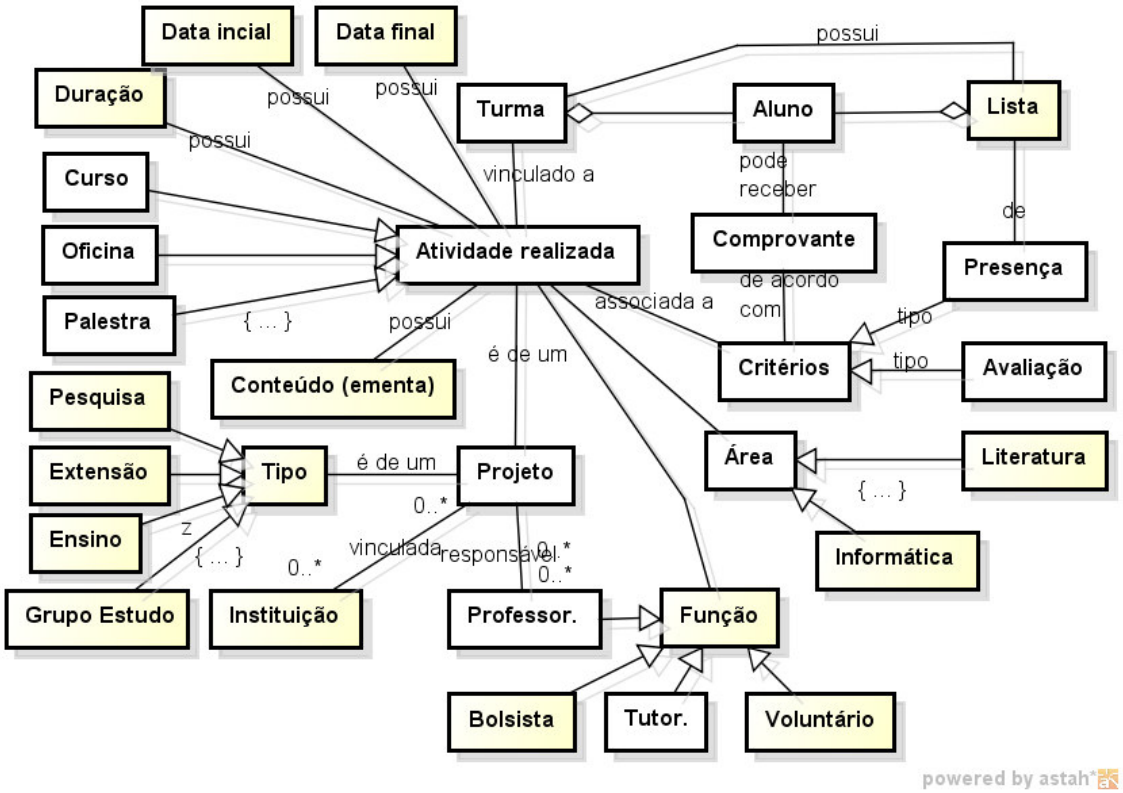

Figura 5 – Visão geral do sistema

A partir da representação da Figura 5 e dos objetivos e finalidades do sistema foram definidos os seus requisitos. A listagem do Quadro 2 apresenta os requisitos funcionais identificados para o sistema.

Identificação	**Nome**	**Descrição**
RF001	Manter instituição	Cadastrar instituição responsável ou vinculada ao projeto. Uma instituição possui: nome, descrição, endereço, pessoa de contato. Excluir instituição responsável ou vinculada a um projeto. Editar, com a possibilidade de alteração, informações de instituição cadastrada. Consultar instituição cadastrada. Consultar dados de uma instituição e a listagem das instituições cadastradas.
RF002	Manter usuário	Cadastrar usuário no sistema, incluindo: nome, endereço, *email*, telefone, funções que a pessoa pode exercer no projeto. Excluir usuário cadastrado. Editar, com a possibilidade de alteração, informações de um usuário cadastrado. Consultar usuário cadastrado. Consultar dados de um usuário, listar usuários por função e listagem geral dos usuários cadastrados.
RF003	Manter atividade	Inclusão de atividade (como curso e palestra) com: nome, descrição, duração (possivelmente em horas), data de início e de fim, estado (cadastrada, em andamento, suspensa, finalizada), ementa (conteúdo programático da atividade). O cadastro de

		atividade deve permitir associar critérios para obter certificado relacionados a presença e aproveitamento, associar área e associar funções: professores, tutores e auxiliares. Excluir atividade cadastrada. Editar, com a possibilidade de alteração, informações de uma atividade cadastrada. Consultar atividade cadastrada. Consultar dados de uma atividade e listagem das atividades cadastradas para cada projeto.
RF004	Manter tipo atividade	Incluir tipo de atividade, contendo nome e descrição. Excluir tipo de atividade cadastrado. Alterar informações de tipos de atividade cadastrados. Consultar tipos de atividade cadastrados.
RF005	Manter área	Incluir a área com: nome e descrição. Excluir uma área cadastrada. Editar, com a possibilidade de alteração, informações de uma área cadastrada. Consultar dados de uma área cadastrada.
RF006	Manter projeto	Incluir projeto definido: nome, descrição, tipo de projeto, professor responsável, instituições responsáveis e/ou colaboradoras, estado (cadastrado – data de início não indicada ou superior a atual ou superior a atual; em andamento - data de início existente e superior a atual e data de fim inexistente ou superior a atual; finalizado – data de fim superior a atual), data de início, data de fim, e descrição do tipo. Excluir projeto cadastrado. Editar, com a possibilidade de alteração, informações de um projeto cadastrado. Consultar dados de um projeto, listagem de projetos cadastrados, listagem das atividades vinculadas a um projeto, listagem de projetos por situação (estado).
RF007	Manter tipo de projeto	Inclusão de tipos de projeto (exemplo: ensino, pesquisa, extensão, interdisciplinar, grupo de estudo). Na inclusão são informados dados como: nome do tipo e descrição do tipo. Excluir tipo de projeto cadastrado. Editar tipo de projeto com possibilidade de alteração de dados de um tipo de projeto cadastrado. Excluir dados de tipo de projeto cadastrado.
RF008	Manter turma	Inclusão de uma turma, contendo nome e descrição. Exclusão de uma turma cadastrada. Editar turma com possibilidade de alteração de dados de uma turma cadastrada. Consulta de informações de uma turma cadastrada.
RF009	Manter composição de turma	Compor uma turma de alunos, vinculando atividade e alunos à mesma. Editar turma composta com possibilidade de alteração de dados de uma turma composta cadastrada. Alteração de dados de uma turma composta cadastrada. Consulta de informações de uma turma composta e cadastrada, listagem de turmas ativas. Uma turma está ativa se a data de inicio existe e é igual ou inferior a atual e data de fim inexiste ou é superior a atual.
RF010	Manter aluno	Cadastrar aluno, incluindo nome, endereço, email, telefone pessoal, telefone de contato e pessoa responsável. Excluir aluno cadastrado. Alterar dados de aluno cadastrado.

		Consultar aluno cadastrado indicando nome.
RF011	Manter critério	Incluir critérios para obter certificado com nome e descrição. Os critérios podem referir-se a percentual de presença, quantidade de faltas, nota superior a, dentre outros. Critérios de presença e aproveitamento. Excluir critério cadastrado. Alterar informações de critérios cadastrados. Consultar critérios cadastrados.
RF012	Emitir listagem para comprovante	Emitir listagem dos alunos, indicando os que atendem ou não aos critérios de presença e aproveitamento.
RF013	Registrar presença	Registrar a presença dos alunos por meio da quantidade de faltas. O registro ocorre por data em que a atividade é realizada.
RF014	Registrar aproveitamento	Registrar o aproveitamento (nota) dos alunos nas avaliações realizadas.
RF015	Manter equipamentos	Controle de equipamentos recebidos para doação, recomposição de equipamentos e equipamentos doados.

Quadro 2 – Requisitos funcionais estabelecidos para o sistema

A listagem do Quadro 3 apresenta os requisitos não-funcionais identificados para o sistema. Os requisitos não funcionais explicitam regras de negócio, restrições de acesso, requisitos de qualidade, desempenho, segurança e outros.

Identificação	**Nome**	**Descrição**
RNF001	Uma atividade deve ter pelo menos uma instituição vinculada como responsável. E pode haver instituições colaboradoras.	Deve ter uma ou mais instituições responsáveis. As instituições responsáveis são definidas como executoras. Pode haver instituições colaboradoras.
RNF002	Uma atividade possui pessoas vinculadas que representam funções.	No momento de cadastro não é necessário ter pessoas vinculadas, mas na execução sim. Uma mesma pessoa pode exercer funções distintas em uma mesma atividade. Uma mesma função pode ter várias pessoas vinculadas em uma atividade.
RNF003	Uma atividade está vinculada a um projeto.	Uma atividade deve estar vinculada a um projeto
RNF004	A emissão de certificado é dependente de critérios.	Um certificado somente pode ser emitido se o aluno atende aos critérios definidos para a mesma. Dentre são presença e aproveitamento. Cada atividade tem uma duração definida em horas e pelas horas registradas pela presença é possível definir o percentual de presença do aluno. Também é possível se o aluno alcançou a nota definida como critério mínimo para a atividade.
RNF005	As funções são atribuídas aos usuários.	As funções são atribuídas a usuários cadastrados no sistema.
RNF006	Aluno para compor turma.	Os alunos que compõem uma turma devem estar cadastrados no sistema.
RNF007	Níveis de acesso.	O sistema deverá ter os níveis de acesso: administrador, professor e tutor. O acesso ao sistema somente por *login* e da seguinte forma: Tutor – acesso ao curso que ele está vinculado, com possibilidade de lançar notas, presenças e conteúdo da aula. Contudo haverá um campo para definir se o

		mesmo terá acesso ao lançamento de notas e presenças. Professor – acesso a todas as funcionalidades, exceto incluir usuários. Administrador - acesso a todas as funcionalidades do sistema.
RNF008	Atividade pertence a uma área.	Uma atividade deve pertencer a pelo menos uma área, definida como a principal. Áreas complementares podem ser vinculadas.
RNF009	Definição de *login* e senha.	Por padrão o *login* é o *email* do usuário e a senha é padrão. No primeiro acesso do usuário ao sistema solicitar alteração da senha. Após cadastrado pelo administrador o usuário pode alterar os seus dados.
RNF010	Armazenamento da senha.	A senha será armazenada no banco criptografada.
RNF011	Acesso ao sistema.	O usuário precisa logar-se para ter acesso ao sistema.

Quadro 3 – Requisitos não funcionais estabelecidos para o sistema

A partir dos requisitos foram definidos os casos de uso apresentados na Figura 6.

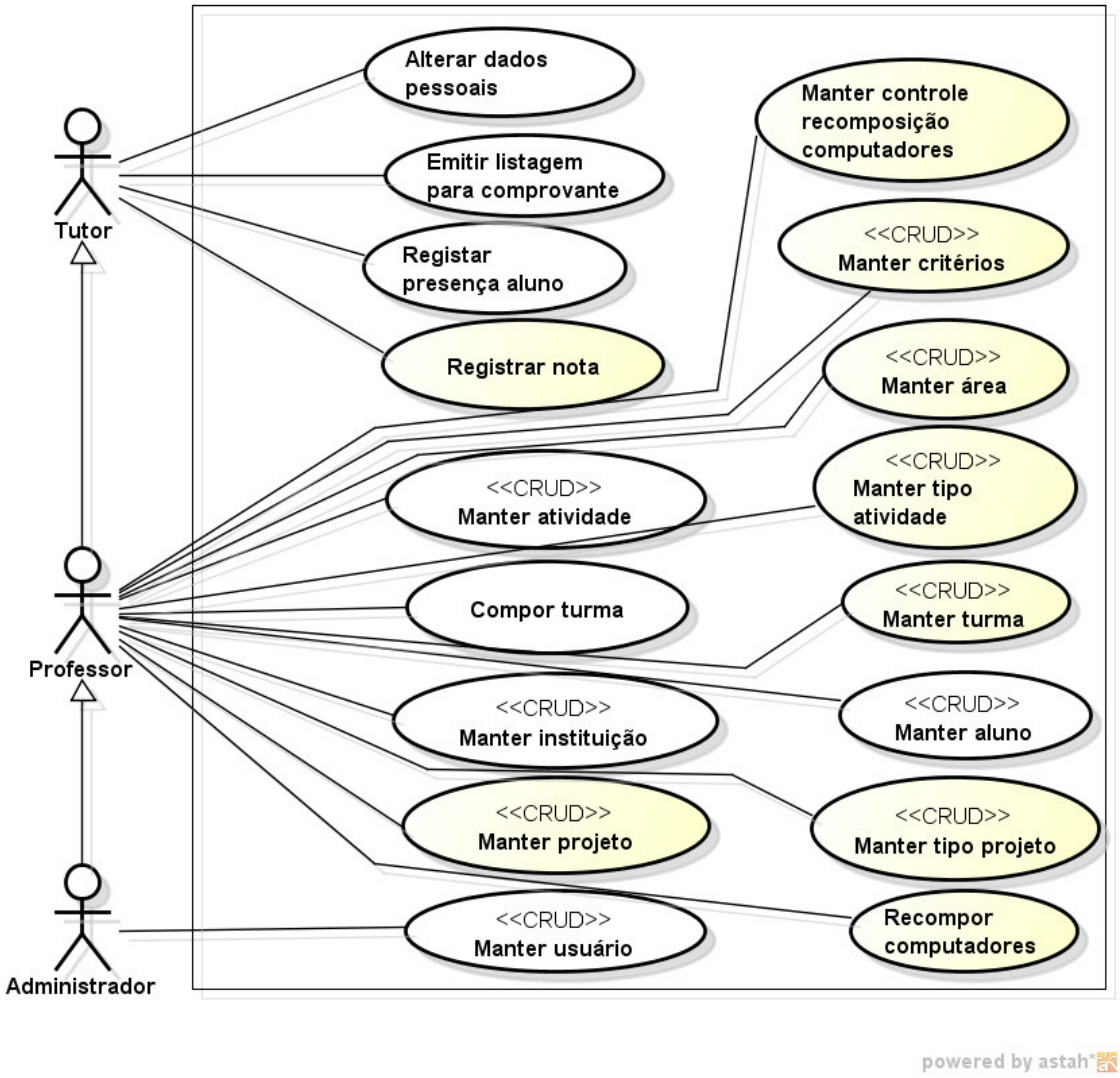

Figura 6 – Diagrama de casos de uso do sistema

A seguir, um caso de uso de cadastro está documentado e representa os casos de uso identificados com o estereótipo "CRUD" na Figura 6. CRUD (*Create, Retrieve, Update, Delete*) se refere às operações realizadas com dados em tabelas do banco de dados. Esse caso de uso é o de "Manter projeto" é utilizado para exemplificar como são descritos os casos de uso desse tipo (CRUD). Todos os casos de uso de cadastro seguem o mesmo padrão, considerando que pode ou não haver necessidade de dados provenientes de outros cadastros.

1.1 Identificador do caso de uso:
Manter projeto.
Descrição:
Cadastro de projetos aos quais estarão atividades vinculadas.
Evento Iniciador:
O usuário solicita a inclusão de um projeto no sistema.
Atores:
Professor, Administrador
Pré-condição:
O tipo de projeto deve estar cadastrado.
Sequência de Eventos:
1. Ator Professor/Administrador acessa a tela para cadastro de um novo projeto e inclui as informações necessárias. O tipo de projeto, um dos campos de entrada, deve estar cadastrado e é escolhido a partir de uma listagem apresentada.
2. O sistema insere os dados no banco de dados, verificando se o nome do projeto está descrito e informa o usuário que o referido projeto foi incluído.
Pós-Condição:
Projeto inserido no banco de dados.
Extensões:
Cadastrar tipo de projeto.

Nome do fluxo alternativo (extensão)	Descrição
1.1 Cadastro de tipo de projeto.	1.1 O ator professor ou administrador acessa a tela do sistema para cadastrar tipo de projeto pretendido e inclui as informações solicitadas. 1.2 Sistema inclui informações no banco de dados.

Quadro 4 – Caso de uso manter projeto

Na seção a seguir são apresentadas as listagens de campos para cada um dos casos de uso identificados como o estereótipo CRUD na Figura 1.

4.2.1 Cadastros

Listagem dos campos de entrada do cadastro de projeto (Quadro 5). O vínculo de instituições a um projeto é realizado por meio de uma aba no cadastro de projetos. Um projeto pode ter várias instituições vinculadas.

Dado	Descrição	Tipo	Obrigatório
Nome do projeto	Identificação do projeto.	Texto	Sim
Descrição	Descrição do projeto	Texto	Não
Tipo do projeto	Escolher de um dos tipos de projeto cadastrados. Vem da tabela de tipos de projetos.	Numérico	Sim
Responsável	Professor/Pessoa responsável, de contato, com o projeto.	Numérico	Não
Instituições	Instituições responsáveis e/ou colaboradoras. Podem ser incluídas diversas instituições, para cada uma dela é necessário selecionar o papel que está em no cadastro de tipos de instituições.	Numérico	Não
Estado	Selecionar um entre os estados predefinidos. O estado é atualizado durante o ciclo de vida do projeto pelos usuários do sistema. Vem da tabela de estados.	Numérico	Sim.
Data início	Data de início de execução do projeto.	Data	Não
Data fim	Data de finalização do projeto.	Data	Não

Quadro 5 – Campos de entrada do cadastro de projetos

Listagem dos campos de entrada do cadastro de tipos de projeto (Quadro 6).

Dado	Descrição	Tipo	Obrigatório
Nome do tipo de projeto	Identificação do tipo de projeto.	Texto	Sim
Descrição	Descrição do tipo de projeto. Exemplos: ensino, pesquisa, extensão, interdisciplinar, grupo de estudos.	Texto	Não

Quadro 6 – Campos de entrada do cadastro de tipos de projeto

Listagem dos campos de entrada do cadastro de estados (Quadro 7). Estados representam as fases de um projeto ou de uma atividade.

Dado	Descrição	Tipo	Obrigatório
Nome do estado	Identificação do tipo de estado.	Texto	Sim
Descrição	Descrição do tipo estado. Exemplos: iniciado, não iniciado, em andamento, suspenso,	Texto	Não

	finalizado.		

Quadro 7 – Campos de entrada do cadastro de tipos de estados

Listagem dos campos de entrada do cadastro de relacionamento entre projetos e instituições. É o vínculo de instituições ao projeto (Quadro 8). Esse cadastro é apresentado em uma aba no cadastro de projetos.

Dado	Descrição	Tipo	Obrigatório
Projeto	Identificação do projeto. Vem da tabela de projetos.	Numérico	Sim
Instituição	Identificação da Instituição. Vem da tabela de instituições.	Numérico	Sim
Tipo da Instituição	Identificação do tipo de instituição. A função que a instituição exerce no projeto. Vem da tabela de tipos de instituições.	Numérico	Sim

Quadro 8 – Campos de entrada do cadastro de tipos de projeto

Listagem dos campos de entrada do cadastro de instituições (Quadro 9). Instituições realizam as atividades de um projeto sendo executoras e elas podem ser colaboradoras ou exercer outras funções.

Dado	Descrição	Tipo	Obrigatório
Sigla	Identificação da instituição.	Texto	Sim
Nome	Descrição da instituição	Texto	Não
Endereço	Endereço da instituição	Texto	Não
Cidade	Nome do município. Vem da tabela de Cidades.	Numérico	Não
CEP	Código de Endereçamento Postal	Texto	Não

Quadro 9 – Campos de entrada do cadastro de instituições

Listagem dos campos de entrada do cadastro de tipos de instituições (Quadro 10). Tipos podem ser, por exemplo: executadora, colaboradora, responsável, beneficiada pela atividade. Os tipos de instituições se referem as possíveis funções de serem exercidas por uma instituição em um projeto.

Dado	Descrição	Tipo	Obrigatório
Nome	Identificação do tipo da instituição	Texto	Sim
Descrição	Complemento ao nome da instituição	Texto	Não

Quadro 10 – Campos de entrada do cadastro de tipos de instituições

Listagem dos campos de entrada do cadastro de usuários (Quadro 11). Além dos campos apresentados neste quadro também é armazenado um campo que indica se o cadastro foi validado (permitindo acesso ao sistema) pelo administrador. Esse campo é gravado automaticamente com um dado que indica que o campo não foi validado. E é alterado quando o usuário Administrador o validar.

Dado	**Descrição**	**Tipo**	**Obrigatório**
Nome do usuário	Identificação do usuário.	Texto	Sim
CPF	Cadastro de Pessoa Física do usuário	Texto	Desejável
RG	Registro Geral do usuário	Texto	Desejável
Endereço	Endereço da instituição	Texto	Não
Cidade	Nome do município. Vem da tabela de Cidades.	Numérico	Não
CEP	Código de Endereçamento Postal	Texto	Não
Login	*Login* no sistema. E o *email*.	Texto	Não
Senha	Senha de acesso ao sistema. Definida por padrão no sistema e alterada no primeiro acesso.	Texto	Não

Quadro 11 – Campos de entrada do cadastro de usuários

Listagem dos campos de entrada do cadastro de funções (Quadro 12). As funções são exercidas pelas pessoas vinculadas aos projetos. Exemplos de funções: instrutor, coordenador e auxiliar.

Dado	**Descrição**	**Tipo**	**Obrigatório**
Nome da função	Identificação da função.	Texto	Sim
Descrição	Descrição da função.	Texto	Não
Requisitos	Requisitos necessários para exercer a referida função.	Texto	Não

Quadro 12 – Campos de entrada do cadastro de funções

Listagem dos campos de entrada do cadastro de atividades (Quadro 13). Uma atividade pode ter diversas áreas vinculadas. O vínculo de áreas com atividades é realizado por meio de uma aba no cadastro de atividades. Uma atividade pode ter mais de um responsável. O vínculo de responsáveis com atividades é realizado por meio de uma aba no cadastro de atividades. Um responsável exerce uma função. No momento em que um responsável é vinculado a uma atividade deve ser definida a função que o mesmo exercerá. A função é inserida no banco de dados por meio de cadastro próprio.

Dado	**Descrição**	**Tipo**	**Obrigatório**
Nome da atividade	Identificação da atividade.	Texto	Sim
Descrição	Descrição da atividade.	Texto	Não

Dado	Descrição	Tipo	Obrigatório
Duração	Duração prevista da atividade. Pode-se adotar um padrão para indicação, como, por exemplo, horas.	Numérico	Não
Data de início	Data de início da atividade. Pode ser a data prevista, mas deve ser atualizada se houver mudança.	Data	Não
Data de fim	Data de fim da atividade. Pode ser a data prevista, mas deve ser atualizada se houver mudança.	Data	Não
Estado da atividade	Escolher entre um dos estados predefinidos. Vem do cadastro de Estados.	Numérico	Não
Conteúdo	Refere-se ao conteúdo programático da atividade. Essa ementa pode ser utilizada na impressão do certificado	Texto	Não
Critério Presença	Critério de presença é percentual sobre as horas da atividade que é necessário o aluno ter para obter certificado de conclusão da atividade, por exemplo, igual ou superior a 75%	Numérico	Não
Critério Aproveitamento	Critério aproveitamento pode referir-se a nota de avaliações que é necessário para o aluno obter certificado de conclusão da atividade, por exemplo, igual ou superior a 7.	Numérico	Não
Tipo da atividade	Define o tipo da atividade. Vem da tabela tipos de atividades.	Numérico	Não
Projeto	O projeto ao qual a atividade pertence. Vem da tabela Projetos		

Quadro 13 – Campos de entrada do cadastro de atividades

Listagem dos campos de entrada do cadastro de tipos de atividades (Quadro 14). Uma atividade é de um tipo, como, por exemplo, curso.

Dado	Descrição	Tipo	Obrigatório
Nome do tipo	Identificação do tipo de atividade.	Texto	Sim
Descrição	Descrição do tipo de atividade. Exemplos: curso, mini-curso, palestra, oficina.	Texto	Não

Quadro 14 – Campos de entrada do cadastro de tipos de atividade

Listagem dos campos de vínculo entre atividades e instrutores (Quadro 15). Esse cadastro permite definir as funções que os usuários exercem em determinada atividade, por exemplo, instrutor, coordenador, auxiliar e tutor.

Dado	Descrição	Tipo	Obrigatório
Usuário	Identificação do usuário. Vem da tabela de usuários	Numérico	Sim
Atividade	Identificação do tipo de atividade. Vem da tabela de atividades	Numérico	Sim
Função	Identificação da função. Vem da tabela de funções.	Numérico	Sim

Quadro 15 – Campos de entrada do cadastro de relacionamento de atividades e instrutores

Listagem dos campos de entrada do cadastro de áreas (Quadro 16). As áreas são utilizadas no cadastro de atividades.

Dado	Descrição	Tipo	Obrigatório
Nome da área	Identificação da área.	Texto	Sim
Descrição	Descrição da área. Exemplo: informática, letras.	Texto	Não

Quadro 16 – Campos de entrada do cadastro de áreas

Listagem dos campos de entrada do cadastro de turma (Quadro 17). Uma turma realiza determinada atividade que está associada a um projeto.

Dado	Descrição	Tipo	Obrigatório
Nome da turma	Identificação da turma.	Texto	Sim
Descrição	Descrição da turma. Complemento ao nome.	Texto	Não
Atividade	Vincular atividade cadastrada à turma. Vem da tabela de atividades.	Numérico	Sim

Quadro 17 – Campos de entrada do cadastro de turma

Listagem dos campos de entrada do cadastro de alunos (Quadro 18). Um aluno pode não ter CPF ou RG e nesse caso o próprio sistema cria um código para individualizá-lo de outro aluno. Para vincular um aluno a uma turma em casos de duplicidade de nomes (nomes iguais) e cadastrados sem CPF ou RG para individualizá-los, o usuário pode consultar as informações cadastrais do aluno e identificá-los por outras informações, como, por exemplo, endereço ou contato. *Email* auxilia a individualizar o usuário quando nomes iguais.

Dado	Descrição	Tipo	Obrigatório
Nome do aluno	Identificação do aluno.	Texto	Sim
CPF	CPF do usuário	Texto	Desejável
RG	RG do usuário	Texto	Desejável

Dado	Descrição	Tipo	Obrigatório
Endereço	Endereço da instituição	Texto	Não
Cidade	Nome do município. Vem do cadastro de Cidades.	Texto	Não
CEP	Código de Endereçamento Postal	Texto	Não
Responsável/Contato	Pessoa responsável ou de contato com o aluno. Pode vir da tabela de Contatos	Texto	Não
Email	Email da pessoa	Texto	Desejável

Quadro 18 – Campos de entrada do cadastro de alunos

Listagem dos campos de entrada do cadastro de vínculo de alunos em uma turma. (Quadro 19) e registro de nota e presença. Na listagem dos alunos para serem vinculados a uma turma é apresentado o nome do aluno e o seu *email* para que possa ser individualizado um aluno no caso de alunos com nomes iguais.

Dado	Descrição	Tipo	Obrigatório
Turma	Identificação da turma. Vem da tabela de turmas	Numérico	Sim
Aluno	Identificação do aluno. Vem da tabela de alunos	Numérico	Sim
Nota	Valor obtido como resultado das avaliações realizadas	Numérico	Não
Presença	Total de presenças (pode ser em horas)	Numérico	Não

Quadro 19 – Campos de entrada do cadastro de turma

Listagem dos campos de entrada do cadastro de contatos de uma pessoa (Quadro 20). Um contato é de alunos, instituições e usuários do sistema. Contato pode ser *email*, telefone, endereço eletrônico, nome de pessoa e etc. Uma pessoa pode possuir mais de um contato.

Dado	Descrição	Tipo	Obrigatório
Pessoa	Identificação da pessoa a qual o contato pertence. Vem das tabelas de alunos, instituições ou usuários.	Texto	Não
Nome do contato	Identificação do contato. Pode ser um nome de pessoa ou instituição.	Texto	Não
Tipo contato	Identificação do contato. Vem da tabela de tipos de contatos.	Numérico	Não

Quadro 20 – Campos de entrada do cadastro de contatos

Listagem dos campos de entrada do cadastro de tipos de contato (Quadro 21). Pode ser *email*, telefone, endereço eletrônico e etc.

Dado	Descrição	Tipo	Obrigatório
Nome do tipo de contato	Identificação do tipo de contato.	Texto	Não

Quadro 21 – Campos de entrada do cadastro de tipos de contatos

Listagem dos campos de entrada do cadastro de cidades (Quadro 22).

Dado	Descrição	Tipo	Obrigatório
Nome da cidade	Identificação da cidade.	Texto	Sim
UF	Unidade de Federação da cidade. Vem da tabela de UFs.	Texto	Não

Quadro 22 – Campos de entrada do cadastro de cidades

Listagem dos campos de entrada do cadastro de UFs (Quadro 23).

Dado	Descrição	Tipo	Obrigatório
Sigla da UF	Identificação da Unidade de Federação.	Texto	Sim
Nome	Descrição por extenso da sigla.	Texto	Não

Quadro 23 – Campos de entrada do cadastro de UFs

Listagem dos campos de entrada do cadastro de instrutores. Instrutores são as pessoas vinculadas às atividades e que exercem alguma função, como, por exemplo, coordenador, responsável, instrutor e tutor. Os campos do cadastro de instrutores estão no Quadro 24.

Dado	Descrição	Tipo	Obrigatório
Nome	Nome do instrutor	Texto	Sim
CPF	CPF	Texto	Não
RG	RG	Texto	Não
Email	Email	Texto	Não
Endereço	Endereço	Texto	Não
Cidade	Cidade	Texto	Não
CEP	CEP	Texto	Não

Quadro 24 – Campos de entrada do cadastro de instrutores

Listagem dos campos de entrada do cadastro interessados em participar de atividades e projetos (Quadro 25). São pessoas que ainda não exercem atividades vinculadas a projetos, mas que tem interesse em fazê-lo assim que houver oportunidade.

Dado	Descrição	Tipo	Obrigatório
Nome	Identificação do interessado.	Texto	Sim
Disponibilidade	Horários, dias, períodos.	Texto	Não
Email	*Email* para contato.	Texto	Não
Telefone	Telefone para contato.	Texto	Não
Interesses	Áreas, tarefas, de interesse.	Texto	Não

Quadro 25 – Campos de entrada do cadastro de interessados

Listagem dos campos de entrada para o Administrador validar usuários (Quadro 26). É apresentada a listagem dos usuários cadastrados no sistema e o administrador os valida. O usuário tem acesso ao sistema somente depois de o cadastro ser validado.

Dado	**Descrição**	**Tipo**	**Obrigatório**
Nome	Nome do usuário.	Texto	Sim
Validado	Informar se o cadastro foi validado ou não.	Numérico	Sim

Quadro 26 – Campos de entrada do cadastro para validar usuários

No Quadro 27 está a listagem dos campos de entrada para o cadastro de equipamentos recebidos como doação e que serão recompostos na Universidade.

Dado	**Descrição**	**Tipo**	**Obrigatório**
Instituição	Instituição de origem (que doou os equipamentos). Vem da tabela Instituições.	Numérico	Sim
Tipo de equipamento	Categoria do equipamento. Vem da tabela tipos de equipamentos.	Numérico	Sim
Nome do lote	Identificação do lote de equipamentos. Por exemplo: computadores tipo *desktop* marca ABC.	Texto	Não
Descrição	Complemento ao nome do lote.	Texto	Não
Quantidade recebida	Quantidade de equipamentos recebidos.	Numérico	Não
Data de recebimento	Data de recebimento dos equipamentos.	Data	Não
Quantidade recomposta	Quantidade de equipamentos recompostos em condições de uso.	Numérico	Não
Saldo	A quantidade doada. Podem ser doados tantos equipamentos quantos estiverem na quantidade recomposta.	Numérico	Não
Data de recomposição	Data que os equipamentos ficaram em condições de uso.	Data	Não
Destino do descarte	Destino dado ao material restante da recomposição e equipamentos que não puderam ser utilizados.	Texto	Não

Quadro 27 – Campos de entrada do cadastro equipamentos recebidos para recomposição

Listagem dos campos de entrada para o cadastro de tipos de equipamentos (Quadro 28).

Dado	Descrição	Tipo	Obrigatório
Nome tipo de equipamentos	Identificação do tipo de equipamento.	Texto	Sim
Descrição	Complemento ao nome do tipo de equipamento.	Texto	Não

Quadro 28 – Campos de entrada do cadastro de tipos de equipamentos

O Quadro 29 apresenta a listagem dos campos de entrada para o cadastro de equipamentos doados, oriundos da atividade de recomposição.

Dado	Descrição	Tipo	Obrigatório
Tipo de equipamento	Identificação do tipo de equipamento. Vem da tabela tipos de equipamentos.	Numérico	Sim
Lote do equipamento	Identificação do lote de origem do equipamento. Vem da tabela de cadastro de equipamentos recebidos para recomposição. Após ser selecionado o lote, apresentar a quantidade de equipamentos em saldo do referido lote.	Numérico	Sim.
Instituição	Instituição de origem (que doou os equipamentos). Vem da tabela Instituições.	Numérico	Sim
Quantidade doada	Quantidade de equipamentos destinados à doação.	Numérico	Sim
Data de doação	Data que a doação foi realizada	Data	Não
Motivo	Motivo da doação. Pode ser a indicação de solicitação da instituição.	Texto	Não

Quadro 29 – Campos de entrada do cadastro equipamentos doados

No Quadro 30 está a descrição dos casos de uso que não se referem especificamente a cadastro com operações padrão de inclusão, exclusão, consulta e alteração.

Identificador do caso de uso:
Alterar dados pessoais.
Descrição:
O usuário pode alterar os seus dados pessoais incluindo *login* e senha, após ter sido cadastrado pelo administrador com o *email* como *login* e uma senha padrão.
Evento Iniciador:
O usuário acessa a tela de alteração de dados pessoais. Para isso o usuário deve estar logado ao sistema.
Atores:

Professor, Administrador

Pré-condição:

O usuário deve já ter sido cadastrado pelo administrador.

Sequência de Eventos:

1. Ator acessa tela para cadastro de um novo projeto e incluindo as informações necessárias. O tipo de projeto, um dos campos de entrada, deve estar cadastrado e é escolhido de uma listagem apresentada.

2. O sistema insere os dados no banco de dados, verificando se o nome do projeto está descrito e informa o usuário que o referido projeto foi incluído.

Pós-Condição:

Projeto inserido no banco de dados.

Extensões:

Cadastrar tipo de projeto.

Quadro 30 – Caso de uso alterar dados pessoais

O Quadro 31 contém a descrição do caso de uso "Login no sistema".

Identificador do caso de uso:

Login no sistema

Descrição:

O usuário cadastrado pode logar-se ao sistema por meio de *login* e senha. A tela de *login* permite ao usuário solicitar o seu *login* e senha que será enviado pelo *email* cadastrado. Para que a senha seja enviada o usuário deve informar o seu endereço de *email* como registrado no seu cadastro.

Evento Iniciador:

O usuário acessa a tela de *login* do sistema.

Atores:

Professor, Administrador, Tutor.

Pré-condição:

O usuário deve já ter sido cadastrado pelo administrador.

Sequência de Eventos:

1. Ator acessa a tela de *login* do sistema e informa *login* e a respectiva senha.

2. O sistema verifica os dados e permite acesso do usuário ao sistema.

Pós-Condição: usuário com acesso ao sistema.

Extensões:

1.1 Ator esqueceu o *login* e/ou senha.

1.1.1 Ator informa seu *email*, como cadastrado no sistema, e o sistema envia a respectiva senha e *login* cadastrados.

Inclusão:

Validação de *login* e senha.

Modelo de interface:

Possibilitar uma forma de interação para o usuário indicar que esqueceu o seu *login* ou senha.

Quadro 31 – Caso de uso login no sistema

O caso de uso Registrar presença está descrito no Quando 32. A descrição desse caso de uso se aplica a Registrar nota.

Identificador do caso de uso:
Realizar chamada
Descrição:
Registrar a presença dos alunos na atividade.
Evento Iniciador:
O usuário acessa a tela para registro de chamada.
Atores:
Professor, Administrador, Tutor.
Pré-condição:
A turma deve estar cadastrada e com atividade e alunos vinculados.
Sequência de Eventos:
1. Ator acessa a tela para registrar a presença dos alunos de uma turma.
2. O sistema insere os dados informados no respectivo cadastro.
Pós-Condição: chamada realizada.

Quadro 32 – Caso de uso realizar chamada

O Quadro 33 apresenta a descrição do caso de uso Emitir listagem para comprovante.

Identificador do caso de uso:
Emitir listagem para comprovante.
Descrição:
Emitir lista de alunos que realizaram determinada atividade. Devem ser identificados os alunos que atendem aos critérios de aproveitamento e presença e os que não os atendem.
Evento Iniciador:
O usuário acessa a tela para emissão de lista para certificado.
Atores:
Professor, Administrador.
Pré-condição:
A atividade deve estar em estado finalizada.
Sequência de Eventos:
1. Ator acessa a tela para emissão de certificado, indicando atividade e aluno.
2. O sistema verificar se o respectivo aluno atende os critérios estabelecidos para a atividade e emite certificado.
Pós-Condição:
Lista de alunos para emissão de certificado emitida.

Quadro 33 – Caso de uso emitir lista para certificado

A Figura 7 apresenta o diagrama de entidades e relacionamentos do Banco de Dados. Essas tabelas representam, basicamente, os cadastros existentes no sistema.

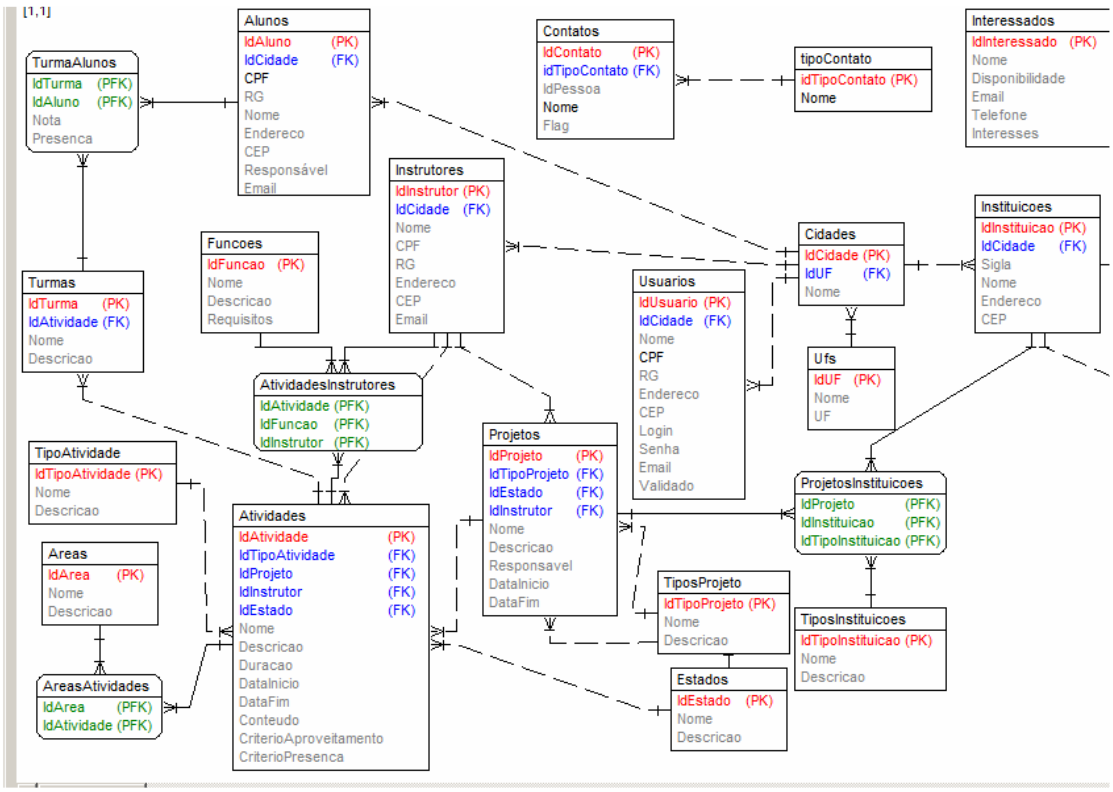

Figura 7 – Diagrama de entidades e relacionamentos do banco de dados (cadastros)

A Figura 8 apresenta as tabelas (entidades) relacionadas ao controle de equipamentos recebidos para recomposição e o destino dos mesmos. A Tabela "Instituicoes" está colocada no diagrama da Figura 8 para representar os relacionamentos entre as tabelas.

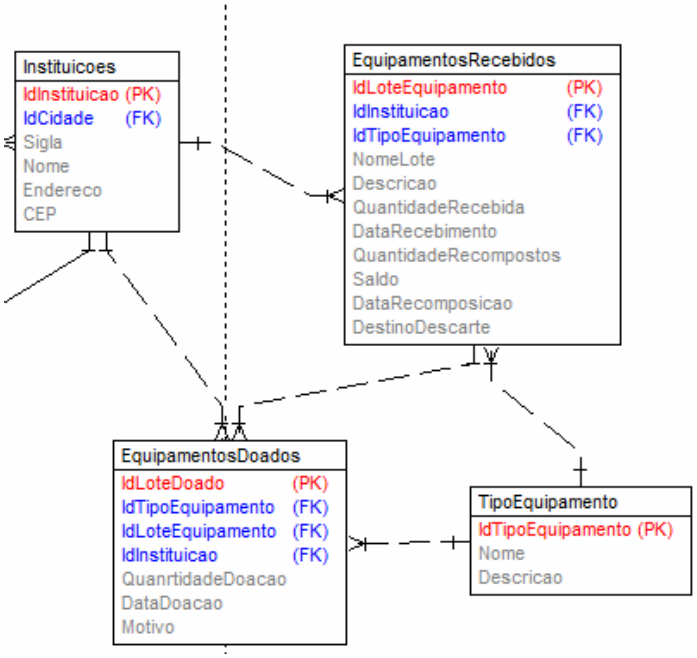

Figura 8 – Diagrama de entidades e relacionamentos do banco de dados (recomposição)

A seguir a descrição das tabelas que compõem o banco de dados, conforme expõe a as Figura 7 e 8.

4.2.2 Tabelas

Tabela usuários (Quadro 34) - um usuário que possui *login* e senha e nesse caso ele tem acesso ao sistema. Um usuário sem *login* ou senha pode possuir funções (professor, voluntário, tutor, palestrante, ajudante) relacionadas às atividades. O administrador valida o cadastro do usuário. Ao cadastrar-se o campo "Validado" é gravado automaticamente com informação que representa que o cadastro não está validado. A validação é realizada pelo administrador.

Campo	**Tipo**	**Nulo**	**Chave primária**	**Chave estrangeira**	**Valor padrão**	**Observações**
IdUsuario	Número	Não	Sim	Não		
IdCidade	Número	Não	Não	Sim	Pato Branco	
Nome	Texto	Não	Não	Não		
CPF	Texto	Não	Não	Não		Com máscara Preenchimento desejável
RG	Texto	Sim	Não	Não		Preenchimento desejável
Endereço	Texto	Sim	Não	Não		
CEP	Texto	Sim	Não	Não		Com máscara
Login	Texto	Sim	Não	Não		Email do usuário.
Senha	Texto	Sim	Não	Não		Com máscara.
Validado	Numérico	Não	Não	Não	Não	Salvo automaticamente pelo sistema

Quadro 34 – Tabela usuários

Tabela funções (Quadro 35) - funções exercidas pelos usuários nas atividades. Por exemplo: professor, tutor, auxiliar. Requisitos se referem aos critérios para poder realizar a função.

Campo	**Tipo**	**Nulo**	**Chave primária**	**Chave estrangeira**	**Valor padrão**	**Observações**
IdFuncao	Número	Não	Sim	Não		
Nome	Texto	Não	Não	Não		
Descrição	Texto	Sim	Não	Não		
Requisitos	Texto	Sim	Não	Não		

Quadro 35 – Tabela funções

Tabela alunos (Quadro 36) - alunos participam das atividades realizadas e podem receber certificado, desde que atendam aos critérios definidos para a atividade. O sistema gerará um código automático para usuário sem CPF ou RG.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Valor padrão	Observações
IdAluno	Número	Não	Sim	Não		
IdCidade	Número	Não	Não	Sim	Pato Branco	
Nome	Texto	Não	Não	Não		
CPF	Texto	Não	Não	Não		Com máscara. Preenchimento desejável
RG	Texto	Sim	Não	Não		Preenchimento desejável
Endereço	Texto	Sim	Não	Não		
CEP	Texto	Sim	Não	Não		Com máscara
Responsavel	Texto	Sim	Não	Não		
Email	Texto	Sim	Não	Não		

Quadro 36 – Tabela alunos

Tabela instituições (Quadro 37) - instituições que podem ser vinculadas a projetos.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Valor padrão	Observações
IdInstituicao	Número	Não	Sim	Não		
IdCidade	Número	Não	Não	Sim	Pato Branco	
Sigla	Número	Não	Não	Não		
Nome	Texto – 150	Sim	Não	Não		
Endereco	Texto – 300	Sim	Não	Não		
CEP	Texto	Sim	Não	Não		Com máscara

Quadro 37 – Tabela instituições

Tabela cidades (Quadro 38) – cadastro de cidades para serem utilizadas em cadastros que contém endereço.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Valor padrão	Observações
IdCidade	Número	Não	Sim	Não		
IdUf	Número	Não	Sim	Sim		
Nome	Texto	Não	Não	Não		

Quadro 38 – Tabela cidades

Tabela UFs (Quadro 39) – cadastro de UFs (Unidades de Federação) a serem utilizadas em cadastros que contêm endereço.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Valor padrão	Observações
IdUf	Número	Não	Sim	Não		
UF	Texto	Não	Não	Não		
Nome	Texto	Não	Não	Não		

Quadro 39 – Tabela UFs

Tabela contato (Quadro 40) – cadastro de contatos de um usuário, aluno, instrutor, interessado ou instituição cadastrados no sistema.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Valor padrão	Observações
IdContato	Número	Não	Sim	Não		
IdTipoContato	Número	Não	Não	Sim		
IdPessoa	Número	Não	Não	Sim		Vem tabela aluno, instituição, usuário, instrutor, interessado
Nome	Texto	Não	Não	Não		
Flag						Identifica se aluno, usuário, instituição, instrutor ou interessado

Quadro 40 – Tabela contatos

Tabela tipo de contato (Quadro 41) – cadastro de tipos de contato. Exemplos de tipos de contatos: msn, *email* e telefone.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Valor padrão	Observações
IdTipoContato	Número	Não	Sim	Não		
Nome	Texto	Não	Não	Não		

Quadro 41 – Tabela tipos de contatos

Tabela instrutores (Quadro 42) - um instrutor exerce funções nas atividades.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Valor padrão	Observações
IdInstrutor	Número	Não	Sim	Não		
IdCidade	Número	Não	Não	Sim	Pato Branco	
Nome	Texto	Não	Não	Não		

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Valor padrão	Observações
CPF	Texto	Não	Não	Não		Com máscara. Preenchimento desejável
RG	Texto	Sim	Não	Não		Preenchimento desejável
Endereço	Texto	Sim	Não	Não		
CEP	Texto	Sim	Não	Não		Com máscara

Quadro 42 – Tabela instrutores

Tabela atividades (Quadro 43) – uma atividade representa o que é realizado (curso, palestra, oficina e outros). Uma atividade está vinculada a um projeto.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Valor padrão	Observações
IdAtividade	Número	Não	Sim	Não		
IdTipoAtividade	Número	Não	Não	Sim		
IdProjeto	Número	Não	Não	Sim		
Nome	Texto	Sim	Não	Não		
Descrição	Texto	Sim	Não	Não		
Duração	Número	Sim	Não	Não		
DataInicio	Data	Sim	Não	Não		
DataFim	Data	Sim	Não	Não		
IdEstado	Número	Sim	Não	Não		
Conteúdo	Texto – 1000	Sim	Não	Não		
CriterioAproveitamento	Número	Sim	Não	Não		
CriterioPresenca	Número	Sim	Não	Não		

Quadro 43 – Tabela atividades

Tabela tipos de instituições (Quadro 44) – contém os tipos de vínculos das instituições que participam dos projetos, como, por exemplo, executora ou colaboradora.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Valor padrão	Observações
IdTipoInstituicao	Número	Não	Sim	Não		
Nome	Texto	Não	Não	Não		
Descrição	Texto	Sim	Não	Não		

Quadro 44 – Tabela de tipos de instituições

Tabela tipos de atividades (Quadro 45) – armazena os tipos de atividades realizadas, como, por exemplo, oficina, palestra, curso.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Valor padrão	Observações
IdTipoAtividade	Número	Não	Sim	Não		
Nome	Texto	Não	Não	Não		
Descrição	Texto	Sim	Não	Não		

Quadro 45 – Tabela tipos de atividades

Tabela atividades e responsáveis (Quadro 46) – essa tabela armazena o vínculo entre funções e atividades. As funções são exercidas por pessoas, denominadas instrutor nessa tabela.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Valor padrão	Observações
IdInstrutor	Número	Não	Sim	Sim		
IdAtividade	Número	Não	Sim	Sim		
IdFuncao	Número	Não	Sim	Sim		

Quadro 46 – Tabela de atividades e responsáveis

Tabela de áreas (Quadro 47) - utilizada no cadastro de atividades, por exemplo: informática, inglês, literatura.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Valor padrão	Observações
IdArea	Número	Não	Sim	Não		
Nome	Texto	Não	Não	Não		
Descrição	Texto	Sim	Não	Não		

Quadro 47 – Tabela de áreas

Tabela de atividades e áreas (Quadro 48) – utilizada para vincular atividades e áreas.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Valor padrão	Observações
IdArea	Número	Não	Sim	Sim		
IdAtividade	Número	Não	Sim	Sim		

Quadro 48 – Tabela de atividades e áreas

Tabela estados (Quadro 49) – etapas de uma atividade ou processo.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Valor padrão	Observações
IdEstado	Número	Não	Sim	Não		
Nome	Texto	Não	Não	Não		
Descrição	Texto	Sim	Não	Não		

Quadro 49 – Tabela estados

Tabela de projetos (Quadro 50) – os projetos cadastrados no sistema. Uma atividade sempre está relacionada a um projeto. Um projeto pode ter várias atividades.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Valor padrão	Observações
IdProjeto	Número	Não	Sim	Não		
IdTipoProjeto	Número	Não	Não	Sim		
Responsável	Texto	Não	Não	Não		
Nome	Texto	Não	Não	Não		
Descrição	Texto	Sim	Não	Não		
IdEstado	Número	Não	Não	Sim		
DataInicio	Data	Sim	Não	Não		
DataFim	Data	Sim	Não	Não		

Quadro 50 – Tabela de projetos

Tabela tipos de projetos (Quadro 51) - define o tipo de projeto, por exemplo: ensino, pesquisa, extensão, grupo de estudos.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Valor padrão	Observações
IdTipoProjeto	Número	Não	Sim	Não		
Nome	Texto	Não	Não	Não		
Descrição	Texto	Sim	Não	Não		

Quadro 51 – Tabela de tipos de projetos

Tabela projetos e instituições (Quadro 52) – vincula as instituições participantes de um projeto.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Valor padrão	Observações
IdProjeto	Número	Não	Sim	Sim		
IdInstituicao	Número	Não	Sim	Sim		
IdTipoInstituição	Número	Sim	Não	Sim		

Quadro 52 – Tabela de projetos e instituições

Tabela turmas (Quadro 53) – uma turma, conjunto de alunos, participará de uma atividade.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Valor padrão	Observações
IdTurma	Número	Não	Sim	Não		
IdAtividade	Número	Não	Não	Sim		
Nome	Texto	Não	Não	Não		
Descricao	Texto	Sim	Não	Não		

Quadro 53 – Tabela turmas

Tabela alunos por turma (Quadro 54) - uma turma é composta por um conjunto de alunos.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Valor padrão	Observações
IdTurma	Número	Não	Sim	Sim		
IdAluno	Número	Não	Sim	Sim		
Nota	Número	Sim	Não	Não		
Presenca	Número	Sim	Não	Não		

Quadro 54 – Tabela alunos vinculados a uma turma

Tabela interessados (Quadro 55) – são pessoas que tem interesse em participar de atividades e ainda não estão vinculados a projetos.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Valor padrão	Observações
IdInteressado	Número	Não	Sim	Não		
Nome	Texto	Não	Não	Não		
Disponibilidade	Texto	Sim	Não	Não		
Émail	Texto	Sim	Não	Não		
Telefone	Texto	Sim	Não	Não		
Interesses	Texto	Sim	Não	Não		

Quadro 55 – Tabela interessados

Tabela equipamentos recebidos (Quadro 56) – são os equipamentos recebidos para doação, que são recompostos e posteriormente encaminhados para doação. O campo “Saldo” dessa tabela se refere à quantidade de equipamentos recompostos que ainda não foram doados. Os equipamentos à medida que são doados é incrementada a quantidade desse campo.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Valor padrão	Observações
IdLoteEquipamento	Número	Não	Sim	Não		
IdInstituicao	Número	Não	Não	Sim		
IdTipoEquipamento	Número	Não	Não	Sim		
NomeLote	Texto	Sim	Não	Não		
Descrição	Texto	Sim	Não	Não		
QuantidadeRecebida	Número	Sim	Não	Não		
DataRecebimento	Data	Sim	Não	Não		
QuantidadeRecomposta	Número	Sim	Não	Não		
Saldo	Número	Sim	Não	Não		
DataRecomposicao	Data	Sim	Não	Não		
DestinoDescarte	Texto	Sim	Não	Não		

Quadro 56 – Tabela equipamentos recebidos

Tabela equipamentos doados (Quadro 57) – são os equipamentos encaminhados para doação.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Valor padrão	Observações
IdLoteDoado	Número	Não	Sim	Não		
IdTipoEquipamento	Número	Não	Não	Sim		
IdLoteEquipamento	Número	Não	Não	Sim		
IdInstituicao	Número	Não	Não	Sim		
QuantidadeDoacao	Número	Não	Não	Não		
DataDoacao	Data	Não	Não	Não		
Motivo	Texto	Sim	Não	Não		

Quadro 57 – Tabela equipamentos doados

Tabela tipos de equipamentos (Quadro 58) – são os tipos de equipamentos utilizados nos cadastros de equipamentos recebidos e doados.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Valor padrão	Observações
IdTipoEquipamento	Número	Não	Sim	Não		
Nome	Texto	Não	Não	Não		
Descricao	Texto	Sim	Não	Não		

Quadro 58 – Tabela equipamentos doados

4.3 DESCRIÇÃO DO SISTEMA

A interface inicial do sistema para gerenciamento de projetos e atividades é composta por um menu lateral que fornece acessos a todas as funcionalidades implementadas. Na parte superior há uma espécie de menu que contém acessos rápidos para alguns recursos como, por exemplo: instituições alunos, instrutores, compor turmas, entre outras. A Figura 9 apresenta a tela com a interface inicial do sistema, ressaltando os menus lateral e superior.

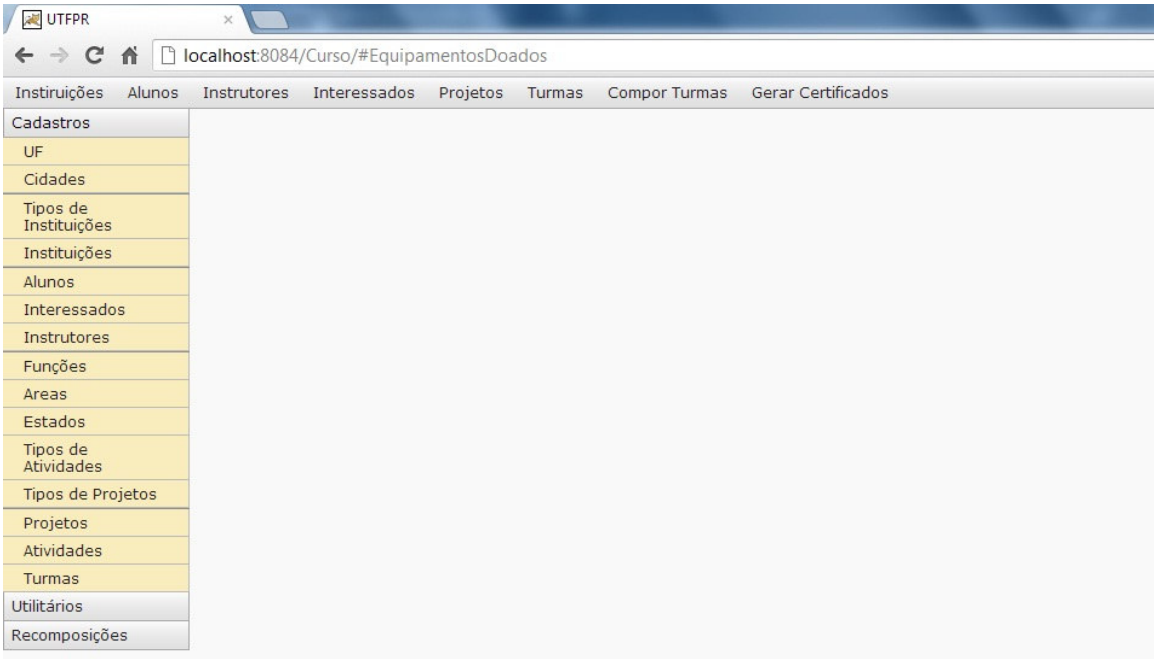

Figura 9 – Interface inicial do sistema

Ao ser selecionado um item no menu, para os cadastros, será aberta uma página em modal (Figura 10). A página ou janela modal fica superposta à página a partir da qual ela foi aberta.

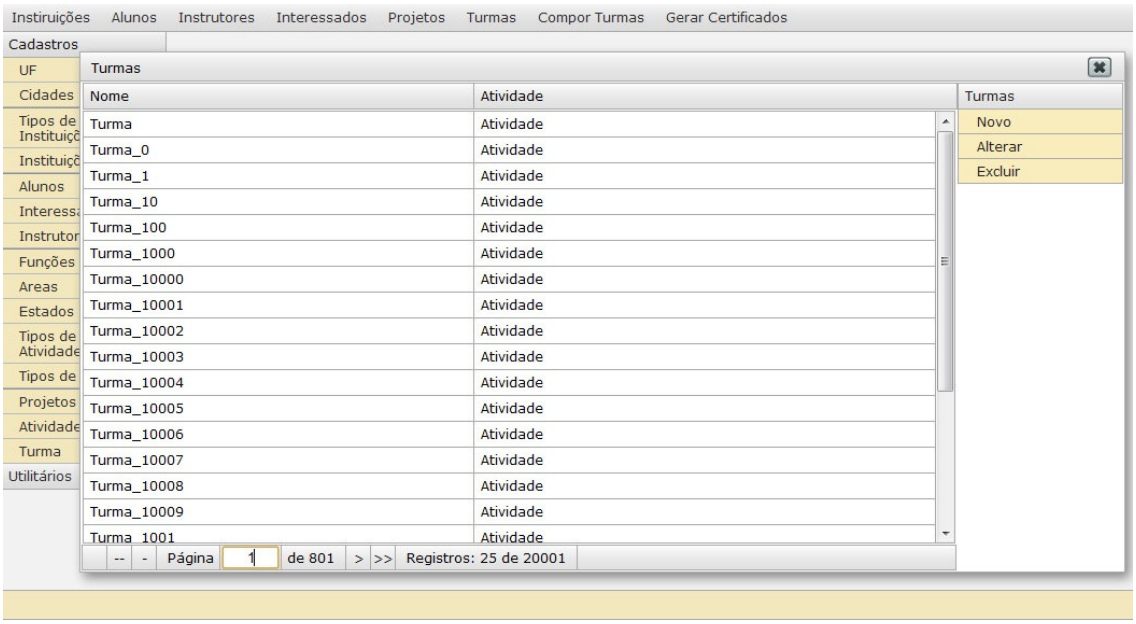

Figura 10 – Exemplo de página modal

Essa página conterá um *grid* listando os registros já cadastrados anteriormente pelos usuários. Os registros não serão listados todos de uma vez e sim como uma espécie de paginação. Na parte inferior dessa página é apresentada a quantidade de registros cadastrados, o número da página sendo mostrada, o total de páginas e a indicação da paginação de registros e o total de registros da referida tabela.

Para navegar entre os registros, basta clicar nos ícones da parte inferior da tela. O ícone '>' avança para os próximos registros e o ícone ">>" avança diretamente para o último registro. Essas opções de avançar de registro em registro ou mesmo que ir para os últimos registros são úteis quando há poucos registros, mas o problema viria quando existem muitos registros. Navegar de um a um para achar o que se quer levaria muito tempo. Sendo assim, foi acrescentada uma terceira opção de navegação. O usuário poderá entrar com um valor na caixa de texto que se encontra na parte inferior do *grid*. Esse valor é referente à página o usuário quer visualizar. Essa opção fica restrita ao número de páginas existentes, mínimo 1 e o máximo é definido pela quantidade de registros contidos no banco na respectiva tabela.

Todos os formulários à medida que são abertos irão sobrepor-se os demais, isso para evitar que o usuário interaja com os demais sem ser o último. Para voltar o foco no penúltimo formulário aberto, é necessário fechar o último. Isso poderá ser feito se a ação do usuário for concluída como, por exemplo, finalizar um cadastro ou clicar em um botão fechar. Esse botão ficará sempre presente no canto superior direito de todos os formulários. A Figura 11 ressalta (marcado com um retângulo no canto superior direito da janela), o ícone para fechar a janela.

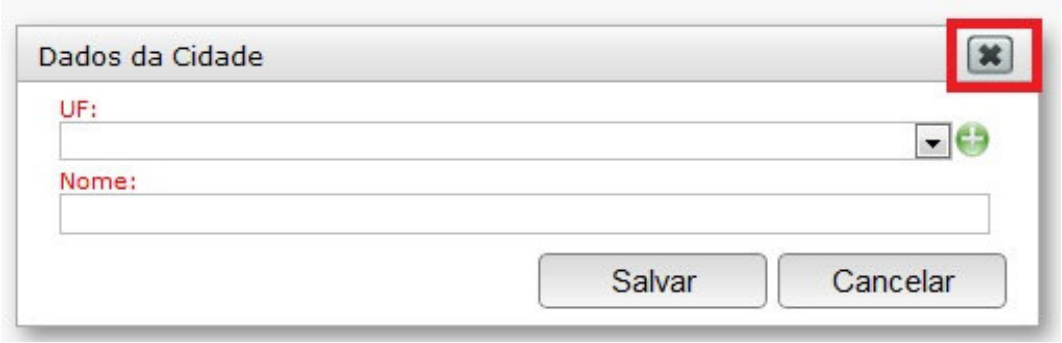

Figura 11 – Formulário de cidades e botão de fechar

Em formulários nos quais há *grids* para exibir registros será possível encontrar menus com algumas opções (menu Turmas com as opções Novo, Alterar e Excluir na lateral direita da Figura 12). Essas opções geralmente serão utilizadas para acessar outros formulários para inserção, alteração e exclusão de dados. Ao ser clicado em novo ou alterar será aberto o mesmo formulário. A única diferença é que no alterar, além de haver registros cadastrados, também será necessário estar com um registro selecionado. Se isso não é feito será mostrada uma mensagem para o usuário que não existe ou não foi selecionado nenhum registro. Esse mesmo procedimento será feito no excluir, pois as validações do alterar também valem para excluir.

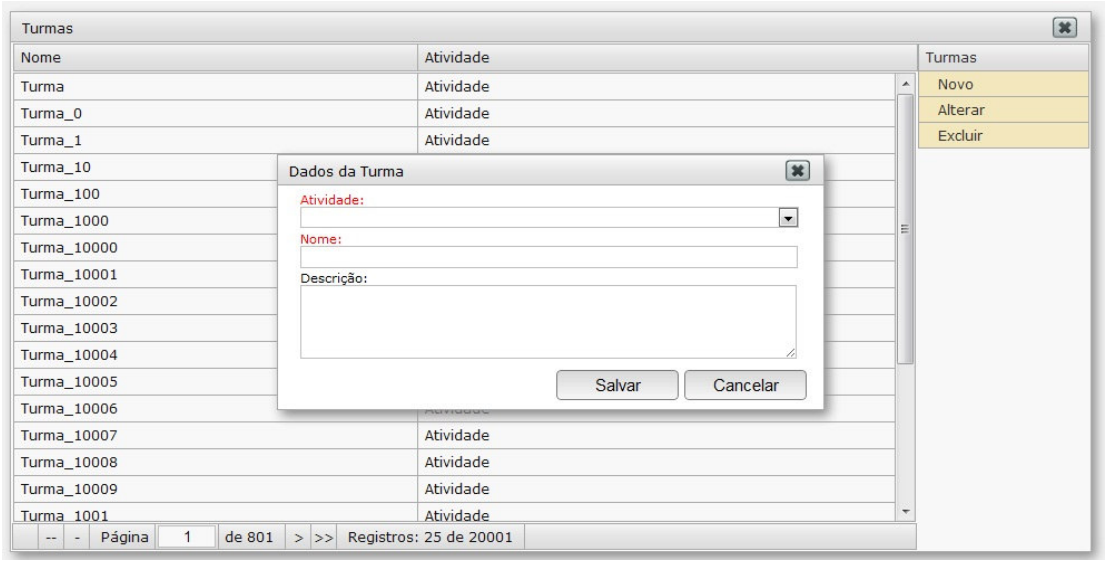

Figura 12 – Formulário com menu lateral

A Figura 13 apresenta o formulário de cadastro de turmas que é aberto após ser selecionada a opção novo no menu lateral. Na Figura 14 é apresentada a mensagem que é mostrada informando que deve ser selecionado um registro para alteração.

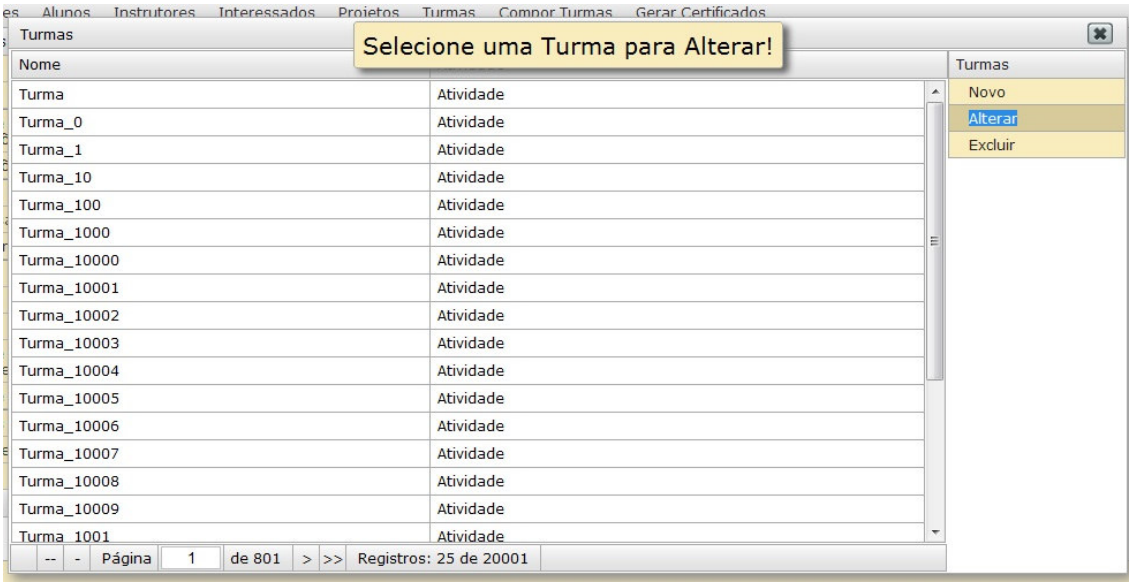

Figura 13 – Mensagem indicando para selecionar um registro para edição

Alguns formulários possuem vínculos para facilitar a navegação. Evitando, assim, que seja preciso voltar tudo para que se possa continuar. Para isso foi adicionado um botão no lado do campo. Assim, o usuário pode clicar no botão e ir diretamente para o cadastro do vínculo. Ao salvar ele retornará para onde estava antes recarregar as informações e será colocado em foco o campo com o último registro cadastrado.

A Figura 14 mostra o cadastro de cidades que é aberto por meio do ícone (que está circulado) que fornece acesso rápido ao respectivo formulário. Nessa Figura também é possível verificar que os dados dos formulários são apresentados sob a forma de abas. As abas são usadas também para dados de formulários que definem relacionamentos um para muitos em tabelas. No caso do formulário apresentado na Figura 14 um instrutor pode ter vários contatos como, por exemplo, várias contas de *email* e números de telefone (comercial, celular, residencial).

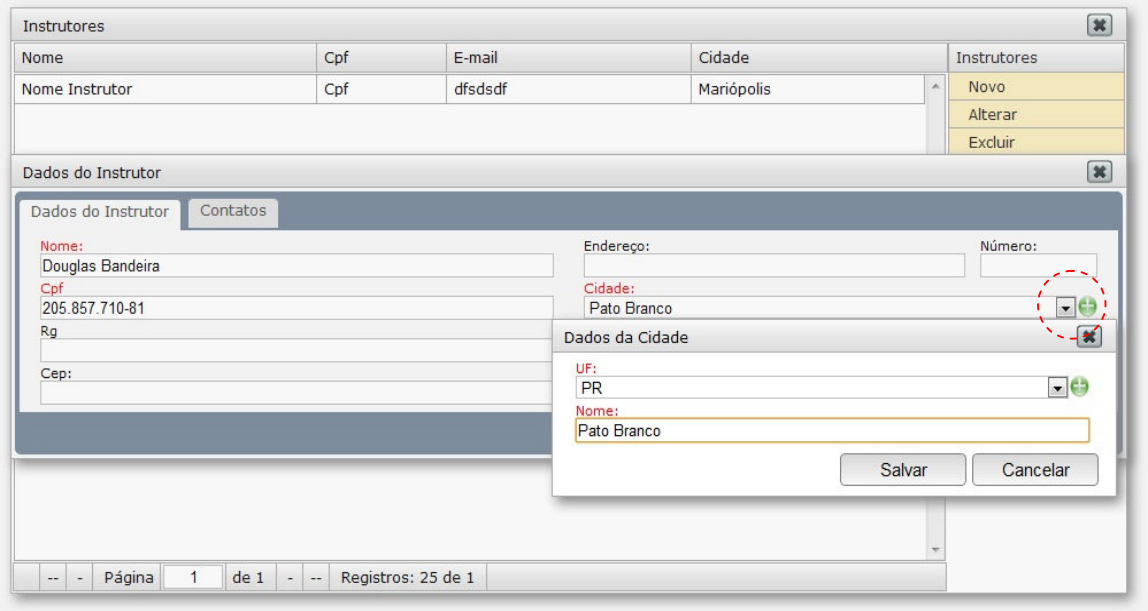

Figura 14 – Formulário com abas e acesso a outro formulário

No exemplo apresentado na Figura 15 estava sendo inserido um novo instrutor, mas a cidade de residência não estava cadastrada. Para evitar que o acesso ao cadastro de cidades pelo menu, foi implementada essa maneira de vínculo entre formulários.

Na Figura 15 é apresentada a tela com o formulário para cadastrar projetos. Os campos que são apresentados como caixas de combinação possuem os valores que são disponibilizados provenientes de outras tabelas do banco de dados.

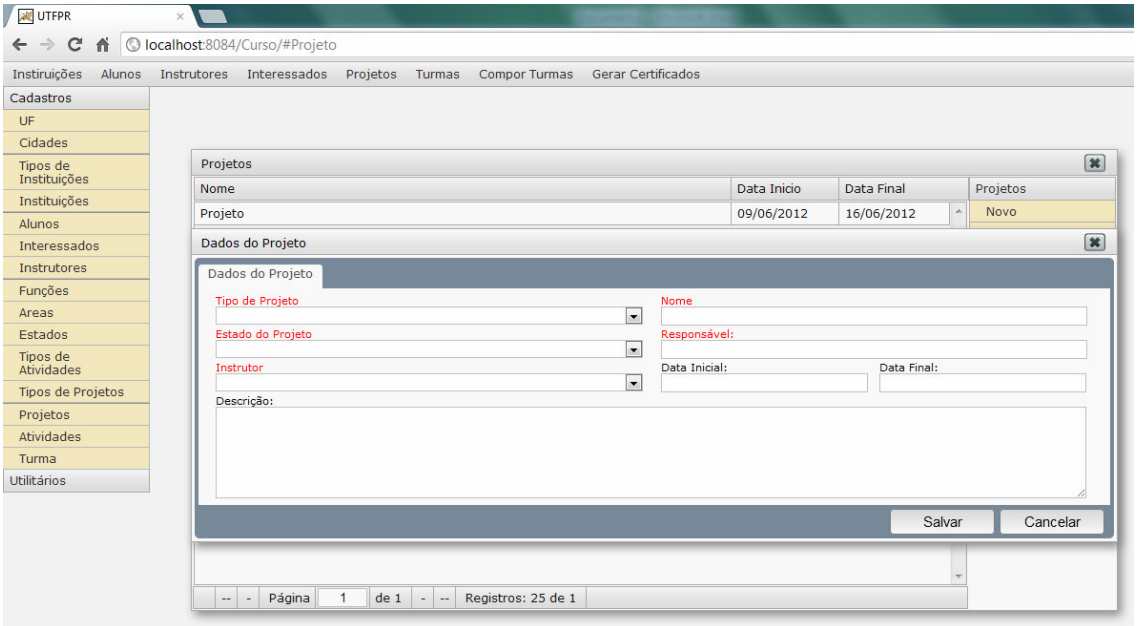

Figura 15 – Formulário para cadastro de projetos

Na Figura 16 está o formulário de cadastro de atividades. Uma atividade sempre possui um projeto vinculado. Um projeto pode possuir *n* atividades vinculadas. Os campos "Aproveitamento (nota mínima)" e "Porcentagem de presença" são utilizados para determinar se o aluno tem direito a comprovante de realização da atividade. Esses campos não são de preenchimento obrigatório.

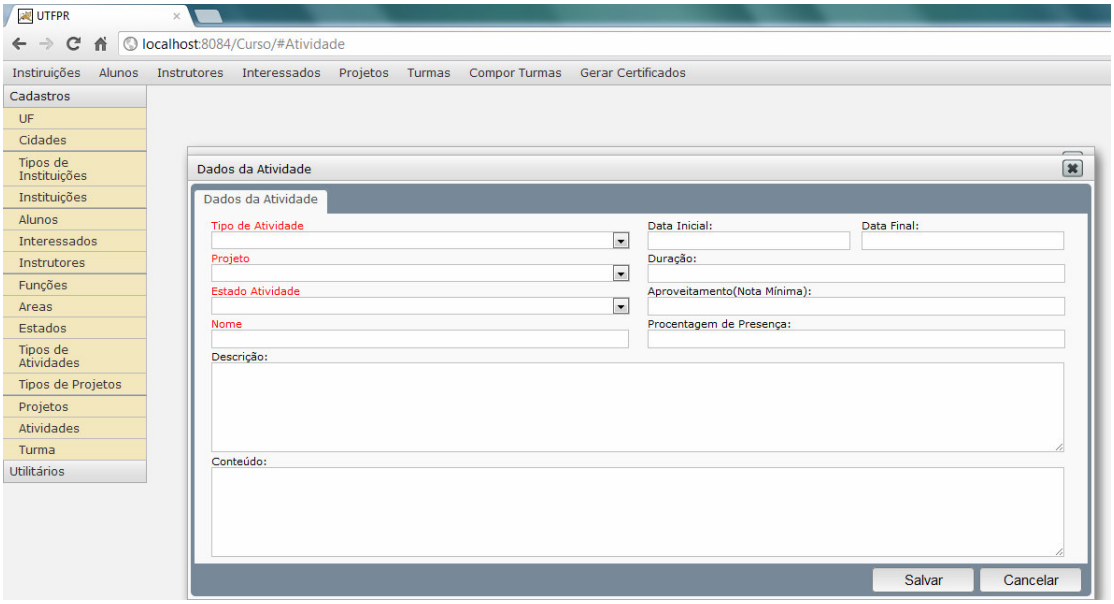

Figura 16 – Formulário para cadastro de atividades

Os equipamentos recebidos de empresas, de instituições e da própria Universidade serão recompostos pela Universidade e posteriormente encaminhados para doação para instituições de ensino e assistenciais. Na Figura 17 está o cadastro de equipamentos recebidos para recomposição.

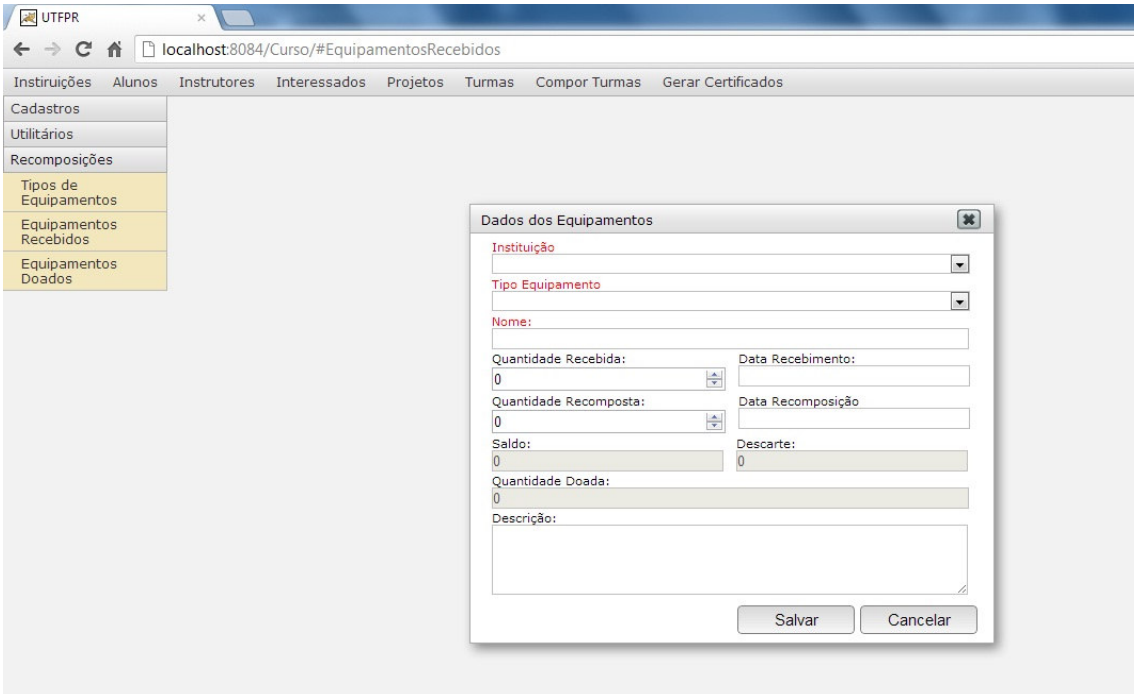

Figura 17 – Cadastro de equipamentos recebidos para recomposição

No cadastro dos equipamentos recebidos para doação é indica a instituição que fez a doação e o tipo de equipamento (impressora, computador, teclado, dentre outros). Esses cadastros foram previamente realizados. Em seguida é indicado a identificação do equipamento (campo nome), a quantidade recebida e a data do recebimento. Após o processo de recomposição é indicado para cada lote de equipamentos recebidos a quantidade recomposta. Ao ser indicada essa quantidade, o campo descarte é automaticamente preenchido. Dos equipamentos doados, os que não têm condições de aproveitamento serão descartados para empresa de reciclagem adequada.

Nesse formulário, o campo saldo e quantidade doada são preenchidos automaticamente a partir das doações realizadas e registras no respectivo formulário.

A Figura 18 apresenta o formulário para doação, ou seja, para destino dos computadores recompostos.

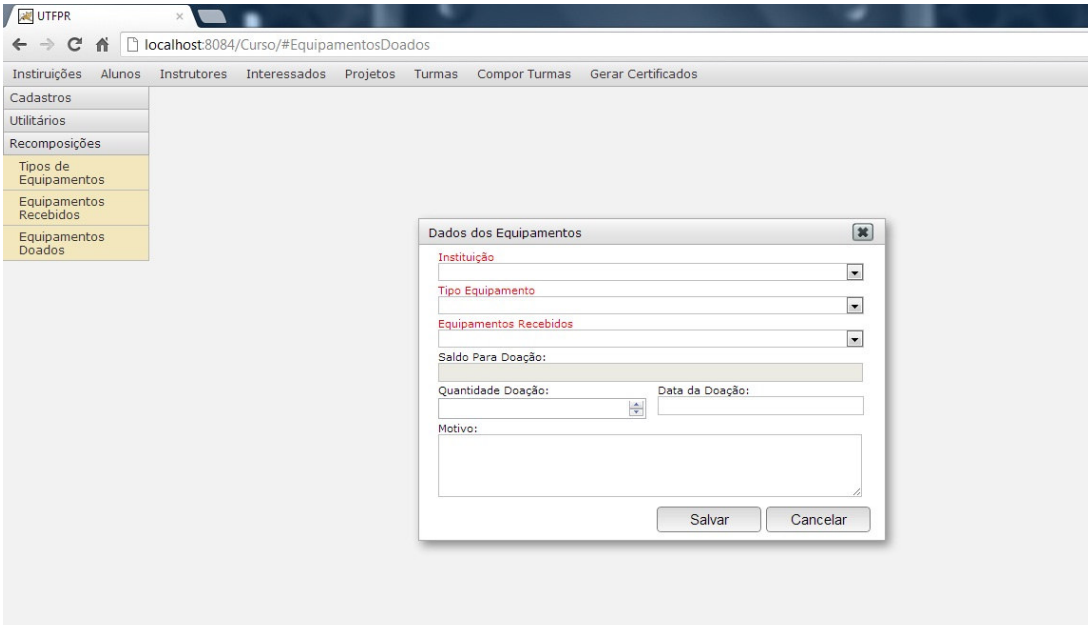

Figura 18 – Formulário para destino dos computadores recompostos

Nesse formulário (Figura 18) é selecionada a instituição (origem dos equipamentos), em seguida é preenchida automaticamente a caixa de combinação com os tipos de equipamentos da referida instituição e os lotes de cada tipo. O campo saldo para doação é preenchido automaticamente a partir das seleções anteriores. O usuário informa a quantidade doação e a data de doação. Também pode descrever o motivo ou justificativa da doação.

Esse formulário possui interação com o formulário apresentado na Figura 17. Essa interação se refere ao saldo e quantidade doada. Nesses dois formulários os campos em cor cinza indicam que eles não são editáveis, sendo preenchidos automaticamente pelo sistema.

4.4 IMPLEMENTAÇÃO DO SISTEMA

Para todos os cadastros e exibição de dados foram implementadas páginas HTML responsáveis pela exibição dos mesmos e funções JavaAcripts para manipulação dos dados, do HTML, validações entre outros, no lado cliente. Já na parte de servidor foram implementados *Servelets*, *Beans* e *Daos*.

Os *Beans* contêm praticamente todos os campos do banco de dados e seus *gets* e *sets* para acessar esses atributos. Os Daos contêm todos os métodos para inserção, alteração e exclusão de dados. São eles:

a) *insert* - inserção e update de registros;

b) *getById* - retorna um registro sendo informado os seu *id*;

c) *getAll* - retorna uma lista de registros.

Os *Servelets* interpretam quais ações serão executadas, a partir de um parâmetro sendo enviado junto ao *link* de acesso para o *Servlet* enviado pelos *JavaScript*.

A Listagem 1 apresenta um exemplo de página HTML criada.

```
<div id="divBodyCadInstrutor" class="divBody">
   <input id="txtId_CadInstrutor" type="hidden" value="0" />
   <div class="divTitulo">
      <div class="divTituloText">
         Dados do Instrutor
      </div>
      <div id="btnFechar_CadInstrutor" class="divBtnFechar">
         <img src="Icones/Icone/pequena22/close.png" />
      </div>
   </div>
   <div class="divContent">
      <div id="pgcCadInstrutor" class="pageDivPricipal">
         <ul class="pageUlOptions">
            <li><a class="active">Dados do Instrutor</a></li>
         </ul>
         <div id="aba1">
            <table class="tabContainerCadastros">
               <tr>
                  <td>
                     <div class="divDivisao">
                        <span class="lblVermelha">Nome: </span>
                        <br />
                        <input id="txtNome_CadInstrutor" type="text" class="txtNome" />
                     </div>
                  </td>
                  <td>
```

Listagem 1 – Exemplo de página HTML

Na Listagem 1 está parte do HTML que apresenta o cadastro de instrutores. Dentre as *tags* HTML ficam visíveis os rótulos “Dados do Instrutor” e “Nome:”. Esses textos são apresentados para o usuário de acordo com a formatação definida pelo HTML.

O cadastro de instrutores, e todos os formulários do sistema, possuem classes *JavaScript* associadas. Na Listagem 2 está parte da função “DbCadInstrutor()”.

```
function DbCadInstrutor(){
   var child;
   var dbCadInstrutor;
   dbCadInstrutor = new DbIPagina();
   dbCadInstrutor.setId("CadInstrutor");
   dbCadInstrutor.setUrl("Instrutor/cadInstrutor.html");
   dbCadInstrutor.setOnLoad(function(){
      DbPageControl('pgcCadInstrutor').init();
      dbCadInstrutor.initEvents();
      //Carrega ddl Cidades
      db.loading.show();
      db.comboBox('ddlCidade_CadInstrutor')
      .setUrlDataSource("JSonCidade?evt=getDs")
      .dataBind(function(){
         db.loading.close();
         if(dbCadInstrutor.getIdRegistro() && dbCadInstrutor.getIdRegistro() != 0){
            db.loading.show();
            var url = "CadInstrutor?evt=get&instrutorId=" + dbCadInstrutor.getIdRegistro();
            $.getJSON(url, function(json){
               $('#txtId_CadInstrutor').val(json.instrutorId);
               $('#ddlCidade_CadInstrutor').val(json.codiCidade);
               $('#txtNome_CadInstrutor').val(json.nome);
               $('#txtCpf_CadInstrutor').val(json.cpf);
               $('#txtRg_CadInstrutor').val(json.rg);
```

Listagem 2 – Classes JavaScript

Todos os arquivos .JSs referentes aos formulários mostrar e cadastrar dados "estenderão" um "Objeto JavaScript" chamado "DbIPagina". Esse objeto conterá várias funções padrões como: carregar o arquivo HTML, mostrar o formulário, esconder, centralizar, iniciar, um método "onLoad" que será executado após ser carregado o formulário. Além de uma variável de controle "idRegistro", para identificar se a operação realizada é inserção ou alteração. Na Listagem 3 é apresentado o código que identificar a operação sendo realizada. É um exemplo da interpretação do parâmetro "evt".

```
private PrintWriter out;
protected void processRequest(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
   response.setContentType("text/html;charset=UTF-8");
   out = response.getWriter();
   try {
      String evt = request.getParameter("evt");
      if(evt.equals("insert")){
         insert(request);
      }
      else if(evt.equals("get")){
         get(request);
      }
   }
   catch(Exception ex){
      out.print("Erro");
   }
   finally {
      out.close();
   }
```

Listagem 3 – Exemplo de Sevelt

A listagem 4 apresenta um exemplo de *Bean*

```
    private String numero;
    private String cep;
    private String email;
    private int contRegistros;

    public Instrutor() {
        instrutorId = 0;
        cidade = new Cidade();
        nome = "";
        cpf = "";
        rg = "";
        endereco = "";
        numero = "";
        cep = "";
        email = "";
        contRegistros = 0;
    }

    /**
     * @return the instrutorId
     */
    public int getInstrutorId() {
        return instrutorId;
    }

    /**
     * @param instrutorId the instrutorId to set
     */
    public void setInstrutorId(int instrutorId) {
        this.instrutorId = instrutorId;
    }
```

Listagem 4 – Bean relacionado ao cadastro de instrutores

Na Listagem 5 está parte do *Dao* responsável pela inserção do objeto *Bean* no banco de dados.

```
public int insert(Object bean) {
   Instrutor ins = (Instrutor)bean;
   int retorno = -1;
   String sql = "SELECT INSTRUTORID_O FROM SP_INSTRUTOR_INSERT(?, ?, ?, ?, ?, ?, ?, ?, ?)";

   try {
      con.setAutoCommit(false);
      stmt = con.prepareStatement(sql);
      stmt.setInt(1, ins.getInstrutorId());
      //...
      rs = stmt.executeQuery();
      while (rs.next()) {
         retorno = Integer.parseInt(rs.getString("INSTRUTORID_O"));
         break;
      }
      con.commit();
   }
   catch (SQLException ex) {

   }
   finally{
      close();
   }
   return retorno;
}

@Override
public Object getById(int id) {
   Instrutor ins = new Instrutor();
   String sql = "SELECT INSTRUTORID, FKCIDADE, NOME, CPF, RG, ENDERECO, NUMERO, CEP, EMAIL "
         + "FROM INSTRUTOR WHERE (INSTRUTORID = ?)";
   try{
      stmt = con.prepareStatement(sql);
```

Listagem 5 – Dao relacionado ao cadastro de instrutores

Todos os arquivos .java irão conter métodos padrões. Quando é lista de registros os métodos são *get* e *delete* e quando cadastros *get* e *insert*. Esses métodos serão interpretados por um parâmetro enviado juntamente ao *link* chamado “evt”.

Ao ser chamado o formulário de cadastros como o de Instrutores, por exemplo, será executado o seguinte código (Listagem 6).

```
$('#mnNovo_allInstrutor').click(function(){
    dbAllInstrutor.hidden();
    dbAllInstrutor.returnPage = function(){
        dbAllInstrutor.loadGrid(db.grid(grid).getPageSelected());
    }
    child = DbCadInstrutor();
    child.parent = dbAllInstrutor;
    child.init();
});
```

Listagem 6 – Métodos padrão .java

O código apresentado na Listagem 6 estará instanciando na variável *child* DbCadInstrutor”. Antes disso é atribuída uma função JavaScript para o “returnPage”. Essa função será executada no momento que for finalizada a inserção do instrutor, que no caso

servirá para carregar novamente o *grid* que contém uma lista com os instrutores cadastrados. Por fim, será executado o método "init" para inicializar o Cadastro de Instrutor "DbCadInstrutor".

No cadastro de instrutores, por exemplo, há campos com descrições em vermelho, indicando que o campo é obrigatório. Portanto se o usuário tentar salvar o formulário sem preencher esses campos será apresentada uma mensagem. E o foco estará no campo, como pode ser visualizado na imagem da Figura 19.

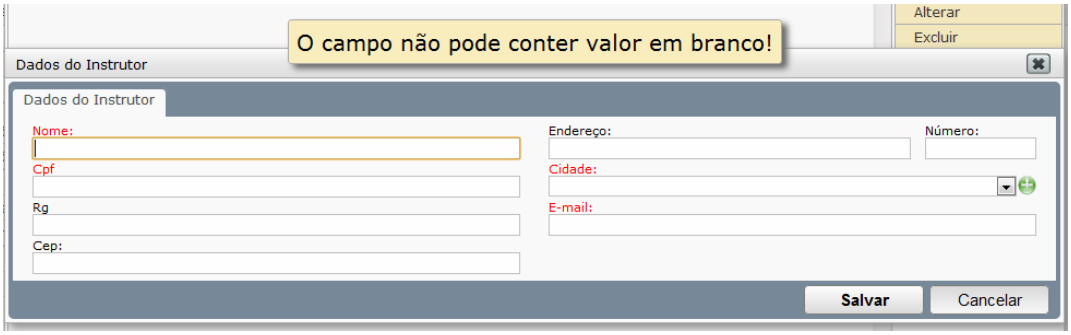

Figura 19 – Mensagem de indicação de campo de preenchimento obrigatório

Se todos os dados estiverem preenchidos corretamente, será executado o código que enviará os dados por meio de uma função JavaScript chamada Ajax, presente na Biblioteca de códigos chamada JQuery. Na listagem 7 é apresentada essa função para o cadastro de instrutores.

```
var parametros =
'&txtId=' + $('#txtId_CadInstrutor').val() +
'&txtNome=' + escape($('#txtNome_CadInstrutor').val()) +
'&txtCpf=' + $('#txtCpf_CadInstrutor').val() +
'&txtRg=' + escape($('#txtRg_CadInstrutor').val()) +
'&txtCep=' + $('#txtCep_CadInstrutor').val() +
'&txtEndereco=' + escape($('#txtEndereco_CadInstrutor').val()) +
'&txtNumero=' + escape($('#txtNumero_CadInstrutor').val()) +
'&ddlCidade=' + $('#ddlCidade_CadInstrutor').val() +
'&txtEmail=' + escape($('#txtEmail_CadInstrutor').val());
$.ajax({
   url: 'CadInstrutor?evt=insert',
   data: parametros,
   success: function(result){
      if(result == 0){
         db.notification('Cpf já Cadastrado para outro Instrutor');
      }
      else if(result == -1){
         alert('Erro ao Cadastrar Instrutor!');
      }
      else {
         db.notification('Instrutor Cadastrado/Alterado com Sucesso.');
         if(dbCadInstrutor.parent){
            dbCadInstrutor.parent.show();
            dbCadInstrutor.parent.returnPage(result);
         }
         dbCadInstrutor.close();
      }
   }
});
```

Listagem 7 – Função JavaScript para o cadastro de instrutores

No servidor, após ser interpretado o “evt”, chama-se o método *insert*. Esse método instanciará e povoará um *Bean*, após isso será invocado o método *insert* do *Dao* passando como parâmetro o *Bean* instanciado. Esse método *insert* para o cadastro de instrutores é apresentado na Listagem 8.

```
private void insert(HttpServletRequest r){
   Instrutor instrutor = new Instrutor();
   instrutor.setCep(r.getParameter("txtCep"));
   instrutor.cidade.setCidadeID(Integer.parseInt(r.getParameter("ddlCidade")));
   instrutor.setCpf(r.getParameter("txtCpf"));
   instrutor.setEmail(r.getParameter("txtEmail"));
   instrutor.setEndereco(r.getParameter("txtEndereco"));
   instrutor.setInstrutorId(Integer.parseInt(r.getParameter("txtId")));
   instrutor.setNome(r.getParameter("txtNome"));
   instrutor.setNumero(r.getParameter("txtNumero"));
   instrutor.setRg(r.getParameter("txtRg"));

   int retorno = new InstrutorDao().insert(instrutor);
   out.print(retorno);
}
```

Listagem 8 – Método insert para cadastro de instrutores

Na alteração será validado se contém pelo menos um registro e se existe ao menos um selecionado. Se essas condições não forem atendidas, o usuário será alertado por meio de uma mensagem semelhante à da validação de campos obrigatórios e código apresentado na Listagem 9 será executado.

```
$('#mnAlterar_allInstrutor').click(function(){
   if(!db.grid(grid).hasItems()){
      db.notification('Não há registros a ser Alterado!');
   }
   if(!db.grid(grid).hasRowSelected()){
      db.notification('Selecione um Instrutor para Alterar!');
   }
   dbAllInstrutor.hidden();
   dbAllInstrutor.returnPage = function(){
      dbAllInstrutor.loadGrid(db.grid(grid).getPageSelected());
   }
   var idInstrutor = db.grid(grid).getRowValue(0);
   child = DbCadInstrutor();
   child.setIdRegistro(idInstrutor);
   child.parent = dbAllInstrutor;
   child.init();

});
```

Listagem 9 – Alteração

Com alteração ocorre algo semelhante ao “Novo”, salvo pelas validações e pela valorização do “idRegistro”, com o valor de identificação do registro selecionado no *grid*. Já no formulário de cadastro, o valor da variável “idRegisto” é verificado. Se ela contém valor e esse valor é diferente de zero, indica que está sendo processada uma alteração, sendo necessário carregar informações (Listagem 10).

```
if(dbCadInstrutor.getIdRegistro() && dbCadInstrutor.getIdRegistro() != 0){
   db.loading.show();
   var url = "CadInstrutor?evt=get&instrutorId=" + dbCadInstrutor.getIdRegistro();
   $.getJSON(url, function(json){
      $('#txtId_CadInstrutor').val(json.instrutorId);
      $('#ddlCidade_CadInstrutor').val(json.codiCidade);
      $('#txtNome_CadInstrutor').val(json.nome);
      $('#txtCpf_CadInstrutor').val(json.cpf);
      $('#txtRg_CadInstrutor').val(json.rg);
      $('#txtEndereco_CadInstrutor').val(json.endereco);
      $('#txtNumero_CadInstrutor').val(json.numero);
      $('#txtCep_CadInstrutor').val(json.cep);
      $('#txtEmail_CadInstrutor').val(json.email);

      db.loading.close();
   });
}
```

Listagem 10 – Json para cadastro de instrutores

Nesse caso, além de enviar o parâmetro "evt" é também necessário enviar um parâmetro "id" referente ao *id* do registro no banco de dados. O *Servelet* interpreta o "evt", instancia um *Bean* e um e um *Dão*. Esse último só é instanciado temporariamente, somente para chamar o método getById retornando o *Bean*. Além disso, deve ser criado um "JSONObject" que será povoado com os dados recuperados do banco de dados. Esse objeto por fim é enviado para o JavaScript e interpretado pelo mesmo. Após isso são valorizados os campos referentes a cada item na tela, como mostrado na Listagem 11.

```
private void get(HttpServletRequest r){
    int id = Integer.parseInt(r.getParameter("instrutorId"));
    Instrutor ins = (Instrutor)new InstrutorDao().getById(id);
    JSONObject retorno = new JSONObject();

    retorno.put("instrutorId", ins.getInstrutorId());
    retorno.put("codiCidade", ins.cidade.getCidadeID());
    retorno.put("nome", ins.getNome());
    retorno.put("cpf", ins.getCpf());
    retorno.put("rg", ins.getRg());
    retorno.put("endereco", ins.getEndereco());
    retorno.put("numero", ins.getNumero());
    retorno.put("cep", ins.getCep());
    retorno.put("email", ins.getEmail());

    out.print(retorno);
}
```

Listagem 11 – Json para retorno de dados

No Excluir também é validado se existem registros e se foi selecionado algum. Após isso, uma pergunta é feita para o usuário indagando-o se ele confirma a exclusão do registro. Selecionando “sim”, será executado um código semelhante ao alterar. É enviado juntamente com *link* os parâmetros “evt” e o “id” referente ao item selecionado no *grid*. A Listagem 12 apresenta o código para excluir do cadastro de instrutores.

```
$('#mnExcluir_allInstrutor').click(function(){
   if(!db.grid(grid).hasItems()){
      db.notification('Não há registros a ser Excluido!');
      return;
   }
   if(!db.grid(grid).hasRowSelected()){
      db.notification('Selecione um Instrutor para Excluir!');
      return;
   }
   var nomeIns = db.grid(grid).getRowValue(1);
   if(window.confirm('Deseja Excluir o Estado ' + nomeIns + '?')){
      db.loading.show();
      var url = "AllInstrutor?evt=delete&id=" + db.grid(grid).getRowValue(0);
      $.ajax({
         url: url,
         success: function(result){
            db.loading.close();
            if(result == 1){
               db.notification('Instrutor Excluido com Sucesso.');
               dbAllInstrutor.loadGrid(db.grid(grid).getPageSelected());
            }
            else{
               db.notification(result);
            }
         }
      });
   }
});
```

Listagem 12 – Código para exclusão no cadastro de instrutores

O *Servelet* interpreta a ação e chama o seu método *delete*, que instancia temporariamente o *Dao* executando o método *delete* do mesmo (Listagem 13).

```
private void delete(HttpServletRequest r){
   int id = Integer.parseInt(r.getParameter("id"));
   int retorno = new InstrutorDao().delete(id);
   if(retorno == 1){
      out.print(retorno);
   }
   else{
      out.print(Utils.msgErroBanco(retorno));
   }
}
```

Listagem 13 – Método delete

4.5 DISCUSSÃO

O autor deste trabalho é desenvolvedor para *web* utilizando ASP.Net com a linguagem C#. Não houve dificuldades para usar outras tecnologias para web. É certo que a plataforma ASP.Net dispõe de uma série de componentes prontos e nativos e de classes que agilizam o desenvolvimento e reduzem a necessidade de escrita de código. E também é verdade que existem vários *frameworks* voltados para a plataforma Java. Contudo, pelo interesse de codificar, preferiu-se escrever códigos HTML, CSS e JavaScript.

Apesar da opção pela escrita de código ao invés de utilizar artefatos prontos, classes foram criadas visando auxiliar no desenvolvimento do sistema resultados deste trabalho. Essas classes estão relacionadas a: entrada de texto; *grid* para exibição de dados; campos do tipo *hidden* os quais contém dados, mas não são vistos pelos usuários (assemelha-se a campo de texto, mas não é mostrado em tela); botão que é auto-explicativo, para alguma ação do usuário, como salvar os dados; e caixas de combinação.

A finalidade de criar essas classes é não necessitar escrever tanto código para gerar as mesmas coisas. Isso evita muito o retrabalho e minimiza as possibilidades de erro e de testes a serem realizados. Dentre essas classes a que mais contribuiu para a implementação é a responsável pelo *grids*. A classe implementada é chamada e as suas propriedades são configuradas e a mesma retorna um *grid* já montando.

Além do retrabalho, outro impacto positivo é a padronização. Tudo será originado de uma classe. Se necessário fazer alterações não é preciso passar página por página alterando o código. A alteração ocorre apenas na respectiva classe. Tanto em HTML como outras linguagens, os componentes possuem propriedades semelhantes como, por exemplo, *id*. Assim, todos os componentes possuem essa propriedade que os identifica e é a partir da mesma que outras propriedades são acessadas. Como muitos componentes possuem as mesmas propriedades, identificou-se um problema na codificação do sistema: a falta de herança múltipla.

Seria muito simples se fosse possível fazer propriedades em classes diferentes e posteriormente herdá-las das classes dos componentes. Contudo, a linguagem Java (assim, como C# e outras) não possui herança múltipla. No desenvolvimento deste trabalho, essa dificuldade foi "burlada". Para isso, a classe mais importante, a do "id" é herdada e as demais são declaradas no escopo da classe como uma variável privada e é criado um método que a acessa.

Essa implementação feita é bem simples, nem pode ser comparada com as encontradas na plataforma ASP.Net, por exemplo. A implementação considera apenas com as propriedades utilizadas no desenvolvimento do sistema e, também, porque todas as requisições com o servidor serão por Ajax, dispensando muitas das funcionalidades existentes nos componentes dessa plataforma.

Comparando ASP.Net e Java, cita-se um componente em ASP.Net, o *UpdatePanel*, componente que faz parte do um *framework.*. Utilizando esse componente para que a página possua requisições Ajax não é necessário realizar nenhuma implementação adicional. Há, ainda, um *ScriptManager* que se encarrega juntamente com outro componente, de realizar todas as requisições. Esse componente enviará o formulário para o servidor. E nesse *framework* o conteúdo contido dentro do *UpdatePanel* será composto novamente. As propriedades dos componentes serão acessadas normalmente por meio do seu "id".

Em Java é recomendado que seja criado um JSP e um *Servelet*. O primeiro ficará encarregado de exibir os elementos na tela, será a parte de HTML. O segundo fica responsável pelo acesso aos itens do servidor, como os *Beans*, *Daos*, classes de componentes entre outros. Isso torna a implementação um pouco mais trabalhosa. Já em ASP.Net fica mais fácil, porque a página é composta por uma página *aspx* e outra que é o .cs. Assim como em Java, uma fica encarregado de exibir os dados e a outra de fazer acessos às classes internas. O *.cs* mapeará o *aspx*, controlando tudo o que acontece. Essa é uma programação orientada a eventos, mas não é necessário codificar cada evento. A própria IDE implementa automaticamente o código dos eventos. A codificação desses eventos não é muito difícil, pois o evento é criado e é mostrado para quem irá executá-lo, não há o trabalho de tratar cada um.

5 CONCLUSÃO

O objetivo principal deste trabalho se refere a complementar a modelagem e implementar um sistema para o gerenciamento de projetos acadêmicos que são realizados em um área ou coordenação de curso. Por meio do sistema implementado é possível identificar e gerenciar os projetos de ensino, pesquisa e extensão sendo realizados, os professores envolvidos e os acadêmicos.

A ênfase deste trabalho foi na implementação das funcionalidades de negócio do sistema. Já que as funcionalidades relacionadas a *background* do sistema foram implementadas como trabalho de estágio. *Background* se refere aos aspectos de facilitariam a implementação do sistema, como operações realizadas em todos os cadastros.

A opção pela implementação de um sistema para Internet decorre da facilidade de acesso por parte dos usuários, visto que diversos professores e alunos (realizando o papel de tutores) usarão o sistema. Considerando que é uma aplicação *web*, o referencial teórico esteve centrado em desenvolvimento para ambiente Internet enfatizando Ajax e outras tecnologias utilizadas.

O sistema implementado possibilitará á área de informática do câmpus Pato Branco o gerenciamento dos seus projetos de ensino, pesquisa e extensão, por meio das diversas atividades vinculadas a esses projetos. Dentre essas atividades citam-se cursos e oficinas e também recomposição de computadores, por exemplo. O sistema também possibilitará o cadastro de alunos interessados em auxiliar na realização de atividades.

Esse sistema embora tenha sido desenvolvido baseado nos interesses e necessidades de uma área e Câmpus específico ele pode ser utilizado para outras áreas e mesmo Universidades. As suas diversas funcionalidades e a forma como foram implementadas oportunizam o seu uso de forma mais ampla.

Como trabalhos futuros destacam-se a realização de testes de usabilidade enfatizando a interface e complementações nos requisitos que possam ser necessárias. Isso porque o sistema será utilizado.

REFERÊNCIAS BIBLIOGRÁFICAS

ADOBE. **Adobe flex 3**. **Now building on flex**. Disponível em: <http://www.adobe.com/products/flex/>. Acesso em: 12 jan. 2012.

ASTAH. **Astah community**. Disponível em: <http://astah.change-vision.com/en/product/astah-community.html>. Acesso em: 08 jan. de 2012.

BRUNO, Vince; TAM, Audrey; THOM, James. **Characteristics of web applications that affect usability: a review**. 17th Australia conference on Computer-Human Interaction: Citizens Online: Considerations for Today and the Future, 2005, p. 1-4.

CANTU, C. H. **Get to know Firebird in 2 minutes (Conheça o Firebird em dois minutos).** Disponível em: <http://www.firebirdnews.org/docs/f>. Acesso em: 26 jan. 2012.

DEITEL, Harvey M.; DEITEL, Paul J. **Java: como programar**. Porto Alegre: Bookman, 2001.

FINKELSTEIN, Anthony C. W.; SAVIGNI, Aandrea; KAPPEL, Gerti; RETSCHITZEGGER, Werner; KIMMERSTORFER; Eugen; PRÖLL, Birgit. **Ubiquitous web application development - a framework for understanding**. 6th World Multiconference on Systemics, Cybernetics andInformatics, Orlando, Florida, US, 2002.

GARRETT, James J. **Ajax: a new approach to web applications**, 2005. Disponível em <http://www.adaptivepath.com/ideas/essays/archives/000385.php>. Acesso em: 2 fev. 2012.

GWT. **Google web toolkit**. Disponível em: <http://code.google.com/intl/pt-BR/webtoolkit/>. Acesso em: 19 jul. 2012.

HENDLER, Jim. **Web 3.0. Emerging web technologies**. IEEE Computer Society, jan. 2009, p. 111-113.

HEWITT, Joe. **Ajax debugging with Firebug**. Dr. Dobb 's Journal, no. 393, p. 22-26, Feb. 2007.

IBEXPERT. **IBExpert dveloper sudio**. Disponível em: <http://www.ibexpert.com/>. Acesso em: 14 fev. 2012.

BLAHA, Michael; JACOBSON, Ivar; BOOCH, Grady; RUMBAUGH, James. **Modelagem e projetos baseados em objetos com UML 2.** 2ª ed. Rio de Janeiro: Elsevier, 2006.

JAVAFX. **JavaFX**. Disponível em: <http://javafx.com/>. Acesso em: 19 fev. 2012.

JAZAYERI, Mehdi. **Some trends in web application development**. Future of Software Engineering (FOSE'07), IEEEl, p. 199-214, 2007.

JQUERY. **jQuery**. Disponível em: <http://jqueryui.com>. Acesso em: 22 fev. 2012.

MACDONALD, M. **Silverlight 2 visual essentials**, Berkeley: Apress, 2008.

MESBAH, Ali; DEURSEN, Arie van. **Migrating multi-page web applications to single-page Ajax interfaces**. 11th European Conference on Software Maintenance and Reengineering, pp. 181-190, March. 2007.

MIKKONEN, Tommi; TAIVALSAARI, Antero. **Web applications – spaghetti code for the 21st century.** Sixth International Conference on Software Engineering Research, Management and Applications, 2008, p. 319-328.

NETBEANS. **NetBeans IDE**. Disponível em: <http://www.netbeans.org>. Acesso em: 12 jan. 2012

OPENLASZLO. **OpenLaszlo**. Disponível em: <http://www.openlaszlo.org/>. Acesso em: 10 fev. 2012.

PAULSON, Linda Dailey. **Building rich web applications with ajax**. Computer, Vol.38, issue 10, pp. 14-17, Oct. 2005.

PRESSMAN, Roger. **Engenharia de software**, 5ª ed. 2002. Rio de Janeiro: McGrawHill.

TOMCAT. A**pache tomcat**. Disponível em: <http://tomcat.apache.org/>. Acesso em: 15 fev. 2012.

SILVA, Mauricio S. **JQuery. A biblioteca do programador javascript**. Novatec, 2010.

SMITH, Keith. **Simplifying ajax style web development**. Computer, Vol.39, no.5, pp. 98-102, May. 2006.

ZEPEDA, Sergio J.; CHAPA, Sergio V. **Frorn desktop applications towards ajax web applications**. 4th International Conference on Electrical and Electronics Engineering (ICEEE 2007), 2007, p. 193-196.