UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE INFORMÁTICA
CURSO DE ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

SIDNEY TETSUO KATO

SISTEMA WEB PARA REGISTRO DE ATIVIDADES DOCENTES E DISCENTES: ESTUDO DE CASO APLICADO AO DAINF DA UTFPR, CÂMPUS PATO BRANCO

TRABALHO DE CONCLUSÃO DE CURSO

PATO BRANCO
2016

SIDNEY TETSUO KATO

SISTEMA WEB PARA REGISTRO DE ATIVIDADES DOCENTES E DISCENTES: ESTUDO DE CASO APLICADO AO DAINF DA UTFPR, CÂMPUS PATO BRANCO

Trabalho de Conclusão de Curso de graduação, apresentado à disciplina de Trabalho de Conclusão de Curso, 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
2016

ATA Nº: **283**

DEFESA PÚBLICA DO TRABALHO DE DIPLOMAÇÃO DO ALUNO SIDNEY TETSUO KATO**.**

Às 16:00 hrs do dia 23 de novembro de 2016, Bloco V da UTFPR, Câmpus Pato Branco, reuniu-se a banca avaliadora composta pelos professores Beatriz Terezinha Borsoi (Orientadora), Andréia Scariot Beulke (Convidada) e Rúbia E. O. Schultz Ascari (Convidada), para avaliar o Trabalho de Diplomação do aluno Sidney Tetsuo Kato, matrícula 01270532, sob o título **Sistema web para registro de atividades docentes e discentes: estudo de caso no DAINF da UTFPR, Câmpus Pato Branco**; 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 16:30 hrs foi encerrada a sessão.

_____
Profa. Beatriz Terezinha Borsoi, Dr.
Orientadora

_____
Profa. Andréia Scariot Beulke, Esp.
Convidada

_____
Profa. Rúbia E. O. Schultz Ascari, M.Sc.
Convidada

_____ _____
Profa. Eliane Maria de Bortoli Fávero, M.Sc — Coordenadora do Trabalho de Diplomação
Prof. Edilson Pontarolo, Dr. — Coordenador do Curso

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

RESUMO

KATO, Sidney Tetsuo. Sistema web para registro de atividades docentes e discentes: estudo de caso aplicado ao DAINF da UTFPR, Câmpus Pato Branco. 2016. 54 f. Monografia (trabalho de conclusão de curso) – Curso de Análise e Desenvolvimento de Sistemas, Universidade Tecnológica Federal do Paraná, Câmpus Pato Branco. Pato Branco, 2016.

Na Universidade Tecnológica Federal do Paraná (UTFPR), como ocorre em outras Universidades, os alunos dos cursos de graduação precisam cumprir horas complementares à sua carga em sala de aula. No caso da UTFPR, essas atividades, denominadas de complementares, são organizadas em três grupos: formação social, humana e cultural; cunho comunitário e de interesse coletivo; e iniciação científica, tecnológica e formação profissional. No Departamento Acadêmico de Informática (DAINF), da UTFPR, Câmpus Pato Branco, é comum que essas atividades sejam promovidas e realizadas por professores e/ou alunos. Assim, um aplicativo para registro dessas atividades pode auxiliar no momento de consultas, verificação de participação de alunos e geração de relatórios para o próprio Departamento. Embora o aplicativo seja definido para atender as necessidades do DAINF, ele poderá ser utilizado por qualquer instituição, de ensino ou não, que tenha interesse e/ou necessidade de manter registro de atividades realizadas como cursos, por exemplo. Como resultado deste trabalho foi desenvolvido um aplicativo *web* para o registro dessas atividades promovidas pelo DAINF. O aplicativo foi desenvolvido com a linguagem PHP, utilizando o *framework* Yii e outras tecnologias que permitem a implementação de uma aplicação *web* denominada rica, as *Rich Internet Application* (RIA).

Palavras-chave: PHP. Framework Yii. Rich Internet Application.

ABSTRACT

KATO, Sidney Tetsuo. Web system for professor and student activities registration: case study in DAINF at UTFPR, Pato Branco Campus. 2016. 2016. 54 f. Monografia (trabalho de conclusão de curso) – Curso de Análise e Desenvolvimento de Sistemas, Universidade Tecnológica Federal do Paraná, Câmpus Pato Branco. Pato Branco, 2016.

At the Federal Technological University of Paraná, as in several other Universities, students of undergraduate courses must meet additional hours beyond their workload in the classroom. Regarding UTFPR, these activities, called complementary, are organized into three groups: social, human and cultural development; community nature and collective interest; and scientific research, technological and professional training. At the Academic Department of Informatics (DAINF) at the Pato Branco Campus of UTFPR, it is common that these activities are promoted and implemented by professors and/or students. Thus, an application developed to register these activities can help with consultation, student participation checking and reporting to the DAINF itself. Although the application developed is set to meet the needs of DAINF, it could be used by any institution with an interest and/or need to keep track of activities such as courses, for example. As a result of this work, a web application for the registration of these activities promoted by DAINF was developed. The application was developed with the PHP language using Yii and other technologies that enable the implementation of a web application called rich, the Rich Internet Application (RIA).

Palavras-chave: PHP. Framework Yii. Rich Internet Application.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE CÓDIGOS

LISTA DE SIGLAS

AJAX	*Asynchronous JavaScript and XML*
CASE	*Computer-Aided Software Engineering*
CSS	*Cascading Style Sheet*
CRUD	*Create, Retrieve, Update and Delete*
DAINF	Departamento Acadêmico de Informática
DOM	*Document Object Model*
HTML	*HyperText Markup Language*
MVC	*Model-View-Controller*
PHP	*PHP Hypertext Preprocessor*
RIA	*Rich Internet Applications*
SQL	*Structured Query Langugage*
UML	*Unified Modelling Language*
UTFPR	Universidade Tecnológica Federal do Paraná
XML	*Extensible Markup Language*
XUL	*User Interface Language*

SUMÁRIO

1 INTRODUÇÃO

Neste capítulo são apresentadas as considerações iniciais do trabalho, os seus objetivos e a justificativa, bem como a organização do texto pela apresentação dos capítulos subsequentes.

1.1 CONSIDERAÇÕES INICIAIS

Os alunos dos diversos cursos da Universidade Tecnológica Federal do Paraná (UTFPR) realizam atividades relacionadas a ensino, pesquisa e extensão, denominadas como complementares. Algumas dessas atividades estão vinculadas a disciplinas, grupos de disciplinas e contexto multi e transdisciplinares. Outras atividades são realizadas por meio de projetos de pesquisa, ensino e extensão. E, há, ainda, as atividades caracterizadas como de extensão que são complementares à formação acadêmica, profissional e pessoal. Diversas das atividades desses três grupos são computadas como atividades complementares perfazendo uma carga horária estabelecida por regulamento para os cursos da UTFPR. Essa carga horária é de cumprimento obrigatório para os alunos de todos os cursos.

Atualmente, o aluno precisa cumprir determinada carga horária, convertida em pontuação, em três grupos para alcançar as 180 horas atribuídas a essas atividades. Esses grupos são atividades de complementação da formação social, humana e cultural; atividades de cunho comunitário e de interesse coletivo; e atividades de iniciação científica, tecnológica e de formação profissional.

No âmbito do Departamento Acadêmico de Informática (DAINF), muitas dessas atividades são organizadas e ministradas pelos próprios professores e/ou alunos. Assim, o controle da realização dessas atividades é importante para os alunos para que possam comprová-las no momento de validação como complementares e para o Curso ou Departamento, visando comprovação de atividades realizadas. As atividades vinculadas à ensino e à pesquisa podem ser comprovadas pelo vínculo a projetos e programas. As atividades de extensão realizadas pelo DAINF são muito diversificadas em sua natureza e na forma de realização. Muitas dessas atividades como as caracterizadas como cursos e oficinas são ministradas pelos próprios alunos. E, assim, um controle mais efetivo pode ser dificultado, quando não há o registro formal do projeto no respectivo setor.

Um controle mais efetivo, com o registro das atividades realizadas, os responsáveis, a carga horária e outros dados, pode auxiliar na consulta para comprovação de atividades complementares e das atividades realizadas pelo Departamento e pelos seus cursos. O desenvolvimento de um sistema web, visando facilitar o acesso, para o registro das atividades realizadas por docentes e discentes do DAINF é o principal objetivo da realização deste trabalho. A modelagem do aplicativo foi realizada como estudo dirigido para estágio curricular supervisionado e consta neste texto para facilitar o entendimento das funcionalidades do aplicativo.

1.2 OBJETIVOS

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

1.2.1 Objetivo Geral

Implementar um sistema *web* para registro de atividades docentes e discentes caracterizadas como complementares ao ensino, pesquisa e extensão.

1.2.2 Objetivos Específicos

- Facilitar o registro de participação de docentes e discentes em atividades promovidas pelos cursos e departamentos como cursos, oficinas e palestras, seja como coordenadores, ministrantes e participantes dessas atividades, agilizando a emissão de declarações e a consulta dessas participações.
- Centralizar o registro de atividades, caracterizadas como complementares, realizadas no âmbito de cursos e/ou departamentos acadêmicos visando a geração de relatórios e a consulta de dados sobre a realização dessas atividades.

- Agilizar a consulta e comprovação de participação em atividades como forma de auxilio no registro de atividades complementares dos alunos e de pontuação docente.

1.3 JUSTIFICATIVA

Atividades extraclasse como cursos, palestras e oficinas, sejam realizadas internamente ou para a comunidade externa, geralmente ocorre por meio de projetos registrados em determinadas diretorias, departamentos ou setores. Porém, a verificação de participação de alunos e professores em determinadas atividades é complexa e morosa, pois é necessário localizar projetos geralmente impressos, verificar listas de presença e outros. O uso de um sistema web para registro desses dados agiliza a consulta e facilita o acesso para o registro e a obtenção de dados por automatizar esses processos.

Com o uso desse sistema, a verificação de participação em atividades para registro de atividades complementares é facilitada e agilizada porque o sistema concentrará as participações em cada uma das atividades e os respectivos participantes.

O sistema modelado e implementado como resultado deste trabalho é definido tendo como base as necessidades e interesses do DAINF, mas pode ser utilizado por outros Departamentos e Cursos da Universidade e mesmo por outras Instituições de Ensino. O sistema se destina ao registro de atividades acadêmicas que visam complementar horas ou a formação dos alunos e que necessitem ou para as quais se tenham interesse em manter registro.

1.4 ESTRUTURA DO TRABALHO

Este texto está organizado em capítulos. O capítulo 2 apresenta o referencial teórico centrado em aplicações web por ser o tipo de aplicações que será posteriormente desenvolvido como resultado da modelagem apresentada como objetivo principal da realização deste trabalho. No Capítulo 3 são apresentados os materiais e o método utilizados para a modelagem do sistema e que serão utilizados na implementação. O resultado da realização deste trabalho, centrado na modelagem do sistema e implementação de funcionalidades

básicas de cadastro, é apresentado no Capítulo 4. No Capítulo 5 são apresentadas as considerações finais seguidas das referências utilizadas no texto.

2 APLICAÇÕES WEB

Este capítulo apresenta o referencial teórico do trabalho que se refere à aplicações web denominadas como ricas.

2.1 APLICAÇÕES INTERNET RICAS

Asynchronous JavaScript and XML (AJAX), sendo *Extensible Markup Language* (XML) é uma técnica avançada que permite o desenvolvimento de aplicações *web* para obter dados de um servidor *web* local que está em *background* com a carga de páginas *web* para construir uma *Rich Internet Applications* (RIA) (DWORAK, 2009).

Para Duhl (2003), as RIAs são aplicações que fazem a unificação do melhor das aplicações *desktop* tradicionais com a interatividade de interface de usuário, com tempo de resposta rápido e sem necessidade de recarregar a página, com o melhor das aplicações *web*, como o recurso de *download* progressivo de conteúdo.

Embora isso tenha certas vantagens sobre a abordagem tradicional de construir aplicações *web*, tais como a baixa latência e a redução do uso de banda, a sua natureza dinâmica contribui para o fenômeno da *web* profunda (*deep web*) (AST; KAPFENBERGER; HAUSWIESNER, 2008). Web profunda, para esses autores, ocorre quando a *web* se torna cada vez mais uma plataforma de aplicações e transações, aumentando, assim, o percentual de documentos não publicamente disponíveis. Para Bergman (2001) a *web* profunda se refere ao conteúdo *web* que está armazenado em bases de dados *backend* sobre as quais são construídas páginas dinâmicas. Esse conteúdo pode ser acessado por meio de formulários *HyperText Markup Language* (HTML) que obtém informações dessas bases de dados ocultas. Essa é a razão pela qual *web* profunda é chamada também de *web* oculta (*hidden web*) (AST; KAPFENBERGER; HAUSWIESNER, 2008).

Vários conceitos de aplicações *web* baseados em indexação dinâmica têm sido propostos (DWORAK, 2009). Essa abordagem baseada em domínio consiste em preencher formulários encontrados na *web* com palavras-chave pertencendo a um contexto específico. Contudo, a implementação provida não é capaz de manipular formulários gerados usando *script* do lado cliente.

A abordagem baseada em *browser* (SU et al., 2010), é construída tendo como base eventos, como clique de mouse e pressionamento de teclas, a fim de analisar uma versão modificada de uma página *web*.

Em contraste com a programação do lado do servidor de uma aplicação *web*, na qual os desenvolvedores podem fazer pressuposições da plataforma de destino, a parte cliente da aplicação é heterogênea e varia em termos de implementação de padrões ECMAScript (LAKSHMAN, 2007), *Document Object Model* (DOM) (HAMMOND, 2008) e outros. Uma solução para esse problema é usar uma camada de abstração como a que é oferecida por *framewoks* Ajax (por exemplo: 5Ext, jQuery, Prototype, script.aculo.us) que são escritos com compatibilidade com a maioria dos navegadores *web*. Contudo, para Dworak (2009) esses incrementos são como notas de rodapé na perspectiva do usuário, porque eles são carregados juntamente com o restante da página *web*.

Várias metodologias usadas para desenvolver aplicações *web* tradicionais não são prontamente transferidas para RIAs devido às características dessas aplicações. Entre essas características destacam-se (DUHL, 2003, PRECIADO et al., 2005):

a) *offloading* da apresentação e camadas interativas do servidor para o cliente;
b) redução de dados transferidos entre cliente e servidor pela atualização parcial da página;
c) distribuição da computação da página entre cliente e servidor;
d) comunicação assíncrona e concorrente entre cliente e servidor.

Powell, Nakamura e Akama (2009) ressaltam que também tem sido mostrado que modelos de padrões comportamentais usados para especificar aplicações *web* tradicionais são inadequados para as RIAa uma vez que elas podem apresentar conteúdo mais rico e flexível em termos de comportamento, tal como computação envolvendo fragmentos parciais da interface (COMAI; CARUGHI, 2007). Adicionalmente a multiplicidade de tecnologias tais como requeridas para implementar aplicações *web* tradicionais em geral e RIAs em particular resulta em um conjunto de dificuldades, como problemas de compatibilidade, segurança, questões de assincronismos e falhas de implementação da tecnologia (COOPER et al., 2007).

Powell, Nakamura e Akama (2007) ressaltam que, adicionalmente às dificuldades de construção das RIAs está o fato que há pelo menos quarto grandes categorias de RIAs sem um caminho de tradução fácil entre elas. Essas categorias são (BOZZON, COMAI, 2006; STEARN, 2007):

a) baseadas em *script* – a lógica do lado cliente é implementada por meio de linguagens de *script*, como JavaScript, e a interface é baseada em um combinação de HTML e *Cascading Style Sheet* (CSS);

b) baseada em *plugins* – o processamento de eventos e a atualização avançada da página são assegurados por meio de *plugins* que interpretam XML, programas de propósito geral ou arquivos de mídia (Flash, Flex, Laszlo, Xamlon);

c) baseada em *browser* – a interação rica é nativamente suportada por alguns navegadores que interpretam linguagens de definições de interface declarativa, como *XML* e *User Interface Language* (XUL);

d) tecnologia *desktop* baseada em *web* – as aplicações são baixadas da *web*, mas são executadas fora do navegador, como ocorre com Java Web Start e Window Smart Client.

3 MATERIAIS E MÉTODO

Este capítulo apresenta os materiais e o método utilizados para a realização deste trabalho. Os materiais estão relacionados às tecnologias e ferramentas utilizadas e o método apresenta a sequência das principais atividades realizadas desde a definição dos requisitos até a implementação e testes.

3.1 MATERIAIS

Para a modelagem e a implementação do sistema foram utilizadas as ferramentas e as tecnologias apresentadas no Quadro 1.

Ferramenta / Tecnologia	Versão	Disponível em	Finalidade
Astah Community	6.2.1	http://astah.net/editions/community	Modelagem de casos de uso
Case Studio 2	2.25	http://www.toad-data-modeler.com/download/	Modelagem do banco de dados
PHP	5.5.12	https://secure.php.net/downloads.php	Linguagem de programação
Wamp	2.5	http://www.wampserver.com/en/	Instalador de ambiente para programação *web* no Windows
Apache	2.4.9	http://www.apache.org/dyn/closer.cgi	Servidor *web* para a aplicação
Yii	1.1.17	http://www.yiiframework.com/download/	*Framework* para *web*
MySQL	5.6.17	http://dev.mysql.com/downloads/	Banco de dados
MySQL WorkBench	6.3 CE	https://www.mysql.com/products/workbench/	Administrador do banco de dados
NetBeans IDE	8.1	https://netbeans.org/downloads/	Ambiente de desenvolvimento

Quadro 1 – Tecnologias e ferramentas utilizadas na modelagem e na implementação

a) Astah Community

Astah Community é uma ferramenta *Computer-Aided Software Engineering* (CASE) que auxilia na realização de atividades de engenharia de software. Essa ferramenta possui diversas opções de modelagem para os diagramas da *Unified Modelling Language* (UML).

b) Case Studio

Case Studio é uma ferramenta para modelagem de banco de dados.

c) PHP

O *PHP Hypertext Preprocessor* (PHP) é uma linguagem de *script* de código aberto de uso geral utilizada para o desenvolvimento de aplicações *web* e que pode ser incorporada em HTML. O que distingue o PHP de linguagens como JavaScript no lado do cliente é que o código é executado no servidor, gerando o HTML que é então enviado para o cliente. O cliente recebe os resultados da execução desse *script*, mas não tem conhecimento de qual é o código fonte que gerou o *script* (PHP, 2016).

d) WAMP

O WampServer é uma aplicação que realiza a instalação automática de um ambiente de desenvolvimento *web* na plataforma Windows. Esse ambiente é composto pelo Apache, a linguagem PHP, o banco de dados MySQL e a ferramenta PhpMyAdmin para gerenciamento do banco de dados.

e) Apache

O servidor Apache (ou Servidor HTTP Apache) é um servidor *web* que é uma aplicação responsável por aceitar pedidos HTTP de clientes, os navegadores, e fornecer respostas HTTP.

f) YII

O Yii é um *framework* de alto desempenho para PHP que faz a utilização de componentes para o desenvolvimento de aplicações *web*. Esse *framework* permite ampla reutilização de códigos acelerando o processo de desenvolvimento.

O Yii é *open-source* ou código aberto é escrito na versão 5 da linguagem PHP que é orientada a objetos. O Yii também traz consigo a arquitetura *Model-View-Controller* (MVC) na construção dos seus projetos. Esse *framework* possui uma estrutura de arquivos e pastas bem organizada que separa claramente o *core* do *framework* com o *core* da aplicação (SILVA, 2016).

g) MYSQL

MySQL é um sistema gerenciador de banco de dados relacional de código aberto. Esse serviço faz uso da linguagem *Structured Query Langugage* (SQL).

h) MYSQL Workbench

O MySQL WorkBench é uma ferramenta visual unificada para o projeto, desenvolvimento e gerenciamento de bancos de dados MySQL. MySQL WorkBench está disponível em Windows, Linux e Mac OS X (MYSQL, 2016).

i) Netbeans IDE

O NetBeans IDE é um ambiente de desenvolvimento *open source* para desenvolvedores de software em diversos tipos de linguagens, sendo executado em diversas plataformas como Windows, Linux, Solaris e MacOS.

3.2 MÉTODO

Os passos para realizar o trabalho estão apresentados a seguir.

a) Estudo das tecnologias

Decidiu-se utilizar um *framework* vinculado à linguagem PHP para facilitar o trabalho, aplicar mais facilmente algum padrão de projetos e o aprendizado que poderia ser obtido. Alguns *frameworks* foram pesquisados entre eles o Zend e o Yii. Optou-se pelo Yii pelos comentários positivos nos fóruns e por apresentar-se de fácil aprendizado nos testes realizados. Esse *framework* implementa a arquitetura MVC, permitindo organizar os arquivos de código por funcionalidades relacionadas à apresentação da aplicação, persistência dos dados e de regras de negócio.

b) Levantamento de requisitos

O levantamento de requisitos ocorreu a partir de conversas informais com professores do Departamento Acadêmico de Informática que demonstraram interesse no desenvolvimento de um sistema para o registro das atividades realizadas por alunos e que poderiam ser caracterizadas como projetos. Os dados necessários para atividades complementares e possíveis relatórios para o Departamento auxiliaram a definir os requisitos do ponto de vista do usuário.

c) Análise e projeto do sistema

Os requisitos inicialmente levantados foram documentados como casos de uso de entidades do banco de dados. Esses requisitos foram complementados com dados que deveriam ser persistidos para manter a consistência do sistema, permitir e facilitar a recuperação de informações.

d) Implementação

A implementação ocorreu pelo uso do *framework* Yii com a linguagem PHP. Como trabalho de estágio foram implementados apenas funcionalidades básicas de cadastro com o objetivo de estudo e entendimento da tecnologia. No trabalho de conclusão de curso as demais

funcionalidades definidas para o sistema foram implementadas, assim como a geração de relatórios.

4 RESULTADOS

Este capítulo apresenta o resultado obtido com o desenvolvimento do trabalho.

4.1 ESCOPO DO SISTEMA

A funcionalidade principal do sistema está relacionada ao registro de participação de alunos em atividades realizadas no âmbito do DAINF da UTFPR, Câmpus Pato Branco. Contudo, ressalta-se que o sistema pode ser utilizado por qualquer instituição que possua interesse e/ou necessidade de fazer o controle da realização das atividades que realiza. Essas atividades são cursos, oficinas e palestras, por exemplo. No contexto acadêmico, as atividades podem ser ministradas e/ou coordenadas por professores e/ou por alunos. Os alunos participam como responsáveis ou como auxiliares na realização das atividades.

Por meio do registro de participação nessas atividades, será possível gerenciar a participação dos alunos e dos professores, gerando comprovantes e dados estatísticos de atividades realizadas pelo Departamento, entre outros.

O sistema permite o registro das atividades dos alunos visando facilitar a consulta de dados para relatórios e para comprovação de atividades complementares. Para os professores, ocorrerá a geração de dados que podem ser utilizados em sua avaliação. E para o Departamento, serão gerados dados para os relatórios de atividades realizadas.

4.2 MODELAGEM DO SISTEMA

O Quadro 1 apresenta a listagem dos requisitos funcionais definidos para o sistema.

Identificação	Requisito	Descrição
RF01	Manter departamentos	Departamentos aos quais estão vinculados cursos e professores.
RF02	Manter cursos	Cursos vinculados ao departamento.
RF03	Manter alunos	Alunos que participam como instrutores, auxiliares ou alunos de atividades realizadas.
RF04	Manter professores	Professores coordenam e ministram atividades.
RF05	Manter projetos	As atividades podem ser realizadas por meio de projetos. Se há um projeto, ele é anexado ao cadastro de projetos.

RF06	Manter atividades	Atividades se referem aos cursos, palestras, oficinas e outros realizados e nos quais há a participação de alunos, podendo haver a participação de professores.
RF07	Manter tipos de atividade	As atividades são categorizadas por tipo: palestras, curso e etc.
RF08	Manter papéis	Uma pessoa (aluno ou professor) vinculada a uma atividade realiza um papel na execução dessa atividade. Exemplos de papéis: instrutor, coordenador, auxiliar, monitor, aluno.
RF09	Registrar atividades realizadas	São as atividades realizadas nas quais estão vinculados alunos e professores por meio dos papéis que realizam na execução da respectiva atividade.
RF10	Consultar atividades realizadas	O aluno pode consultar as atividades que realizou e o professor pode consultar as atividades dos alunos.

Quadro 2 – Requisitos funcionais

A Figura 1 apresenta o diagrama de casos de uso. Nesse diagrama os atores do sistema são:

a) Aluno – com permissão para consultar apenas o registro das suas atividades.

b) Professor – responsável pelo registro dos cadastros do sistema, exceto de usuários, e pelo registro das atividades e poderá realizar consulta de dados.

c) Administrador – realiza o cadastro dos usuários do sistema e herda as funcionalidades do ator professor.

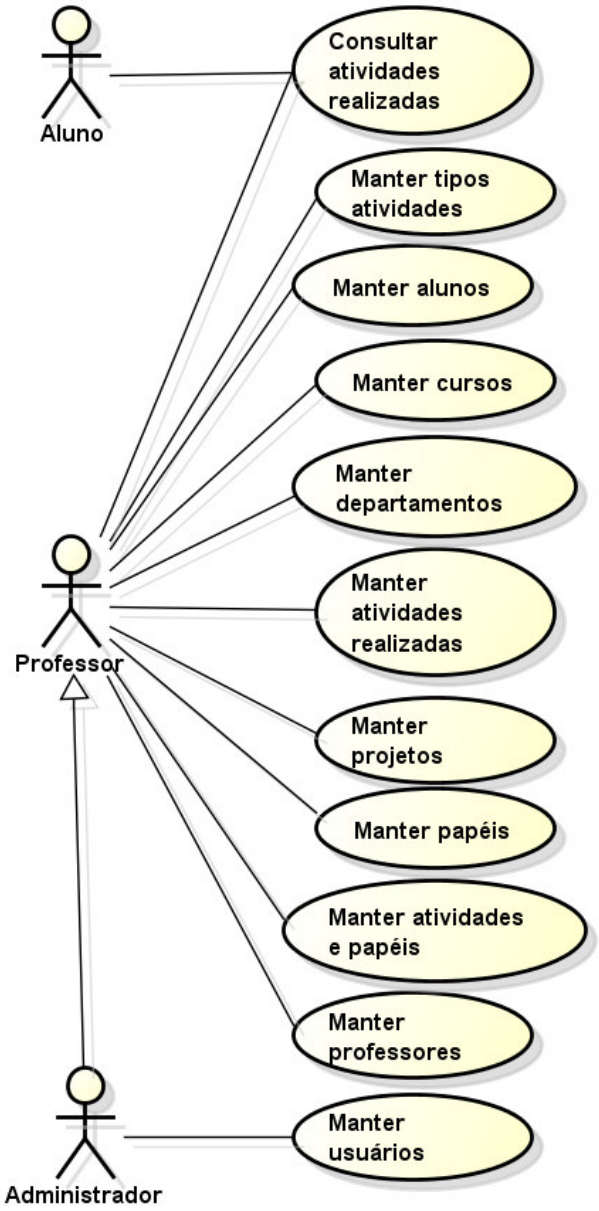

Figura 1 – Diagrama de casos de uso

A Figura 2 apresenta o Diagrama de Entidades e Relacionamentos desenvolvido para o banco de dados da aplicação.

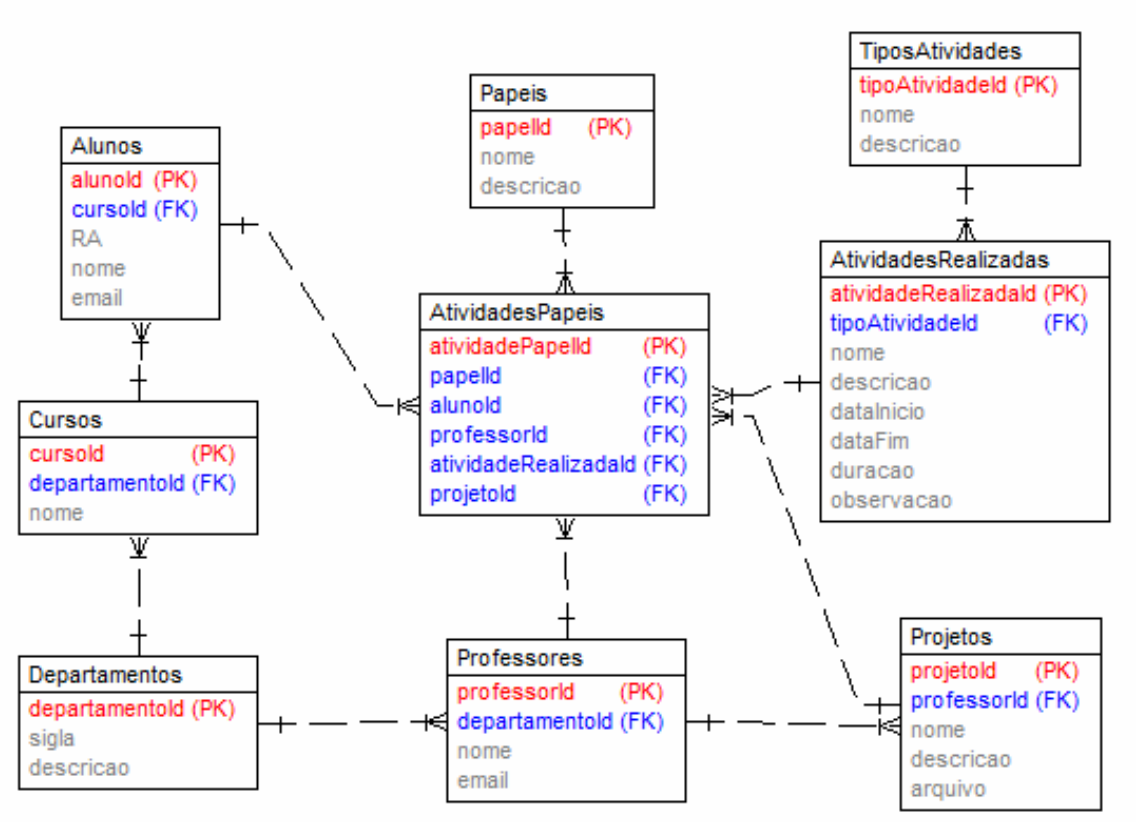

Figura 2 – Diagrama de entidades e relacionamentos

A seguir a descrição das tabelas do banco de dados. O Quadro 3 apresenta os campos da tabela Alunos.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Observações
alunoId	Numérico	Não	Sim	Não	Auto-incremento
cursoId	Numérico	Não	Não	Sim	Da tabela Cursos
RA	Texto	Não	Não	Não	
nome	Texto	Não	Não	Não	
email	Texto	Não	Não	Não	

Quadro 3 – Tabela Alunos

Os campos da tabela Cursos são apresentados no Quadro 4.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Observações
cursoId	Numérico	Não	Sim	Não	Auto-incremento
departamentoId	Numérico	Não	Não	Sim	Da tabela Departamentos
nome	Texto	Não	Não	Não	

Quadro 4 – Tabela Cursos

Os departamentos agregam cursos. Os campos da tabela Departamentos são apresentados no Quadro 5.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Observações
departamentoId	Numérico	Não	Sim	Não	Auto-incremento
sigla	Texto	Não	Não	Não	
descricao	Texto	Não	Não	Não	

Quadro 5 – Tabela Departamentos

Os professores estão vinculados a um departamento e são responsáveis pela organização e realização das atividades, embora esse papel não seja exclusivo de professores. Os campos da tabela Professores são apresentados no Quadro 6.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Observações
professorId	Numérico	Não	Sim	Não	Auto-incremento
departamentoId	Numérico	Não	Não	Sim	Da tabela Departamentos
nome	Texto	Não	Não	Não	
email	Texto	Não	Não	Não	

Quadro 6 – Tabela Professores

As atividades realizadas se referem ao que é proposto por professores e alunos. Os campos da tabela para registro das atividades estão no Quadro 7. Exemplos de atividades: um curso de programação em PHP, que é ofertado por alunos para outros alunos; um curso de Arduino que é ofertado por professores (contando com o auxílio de alunos) para outros alunos.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Observações
atividadeRealizadaId	Numérico	Não	Sim	Não	Auto-incremento
tipoAtividadeId	Numérico	Não	Não	Sim	Da tabela TipoAtividades
nome	Texto	Não	Não	Não	
descriçao	Texto	Sim	Não	Não	
dataInicio	Data	Sim	Não	Não	
dataFim	Data	Sim	Não	Não	
duracao	Numérico	Sim	Não	Não	Tempo (em horas) da atividade
observacao	Texto	Sim	Não	Não	

Quadro 7 – Tabela AtividadesRealizadas

As atividades são categorizadas por tipo e seus campos estão apresentados no Quadro 8.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Observações
tipoAtividadeId	Numérico	Não	Sim	Não	Auto-incremento
nome	Texto	Não	Não	Não	
descricao	Texto	Não	Não	Não	

Quadro 8 – Tabela TiposAtividades

Os alunos que participam de atividades podem estar vinculados como bolsistas ou voluntários a projetos apoiados por editais internos ou externos. Os dados de identificação de projetos ficam armazenados na tabela Projetos (Quadro 9) e se há um projeto envolvido na realização da atividade o mesmo também é armazenado.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Observações
projetoId	Numérico	Não	Sim	Não	Auto-incremento
professorId	Numérico	Não	Não	Sim	Da tabela Professores
nome	Texto	Não	Não	Não	
descricao	Texto	Não	Não	Não	
arquivo	BLOB	Não	Não	Não	

Quadro 9 – Tabela Projetos

A tabela Papéis armazena as funções desempenhadas por professores e alunos na realização das atividades. Os dados dessa tabela estão no Quadro 10.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Observações
papelId	Numérico	Não	Sim	Não	Auto-incremento
nome	Texto	Não	Não	Não	
descricao	Texto	Não	Não	Não	

Quadro 10 – Tabela Papeis

O relacionamento entre atividades e papéis indica o papel desempenhado pelos responsáveis (professores ou alunos) na realização das atividades. Ou o campo “alunoId” ou o campo “professorId” será nulo em um mesmo registro. Esses campos vêm das tabelas alunos e professores, respectivamente e indicam quem está desempenhando o papel sendo registrado. O campo “projetoId” permite identificar o projeto ao qual o aluno está vinculado como instrutor, por exemplo. Os campos da tabela que vincula atividades e papéis são apresentados no Quadro 10.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Observações
atividadesPapelId	Numérico	Não	Sim	Não	Auto-incremento
papelId	Numérico	Não	Não	Sim	Da tabela Papéis
professorId	Numérico	Sim	Não	Sim	Da tabela Professores. Se alunoId está preenchido esse campo será nulo.
alunoId	Numérico	Sim	Não	Sim	Da tabela Alunos. Se professorId está preenchido esse campo será nulo.
atividadeRealizadaId	Numérico	Não	Não	Sim	Da tabela AtividadesRealizadas
projetoId	Numérico	Sim	Não	Sim	Da tabela Projetos

Quadro 11 – Tabela AtividadesPapeis

4.3 APRESENTAÇÃO DO SISTEMA

O leiaute do sistema é composto por uma área superior que contém o logotipo da UTFPR acompanhado do nome do aplicativo, logo abaixo dessa área, está um menu com opções para acesso à página inicial, informações do sistema, contato com o administrador e tela de *login* (para usuários ainda não logados). Na parte central está o conteúdo da página sendo navegada. A parte inferior fica reservada para informações sobre direitos autorais e de tecnologia utilização de implementação.

A Figura 3 apresenta a organização da página. Na região central dessa página estão dados de usuários cadastrados no sistema. A listagem de dados de cadastros é paginada e a navegação é realizada por meio de um clique sobre o número da página e pelos botões “Previous” e “Next”. O botão que representa uma lupa, ao final (direita) de cada linha permite abrir o respectivo registro.

Figura 3 – Página inicial para usuários não logados

Na parte superior de cada coluna da listagem de registro (região circulada na Figura 4) é apresentado um campo vazio. Estes campos podem ser utilizados para realizar buscas específicas, retornando os registros encontrados para a requisição. A Figura 4 apresenta um exemplo dessa forma de busca, para o qual foi digitado o Registro Acadêmico (RA) de um aluno e é apresentado o resultado dessa busca.

Figura 4 – Busca de registros por Registro Acadêmico

A Figura 5 apresenta um exemplo de busca em campo texto. O sistema retornará todos os registros que possuem a *string* informada, independentemente da posição em que a mesma esteja localizada no campo.

Figura 5 – Busca de registros por nome

Ao clicar no botão de visualização, localizado na extremidade direita da lista, o sistema disponibilizará o relatório do aluno desejado, contendo todas as atividades associadas ao mesmo. A Figura 6 apresenta um exemplo de relatório filtrado por aluno.

Figura 6 – Relatório de atividades por aluno

Acessando a opção *login*, uma página para autenticação será exibida (Figura 7). Apenas o administrador e professores podem logar-se para realizar a manutenção dos registros.

Figura 7 – Página de autenticação

Após realizar a autenticação, a página inicial exibirá uma lista com as opções de listar, registrar novo e gerenciar cada entidade disponível, como mostra a Figura 8.

Figura 8 – Página inicial para usuários logados

Ao clicar na entidade desejada dentro da opção “Listar”, uma nova página será carregada contendo uma lista com todos os dados, além de um novo menu localizado à esquerda com as opções “Registrar” para inserção de novas entradas, e “Gerenciar” para a manutenção dos registros, como apresenta a Figura 9.

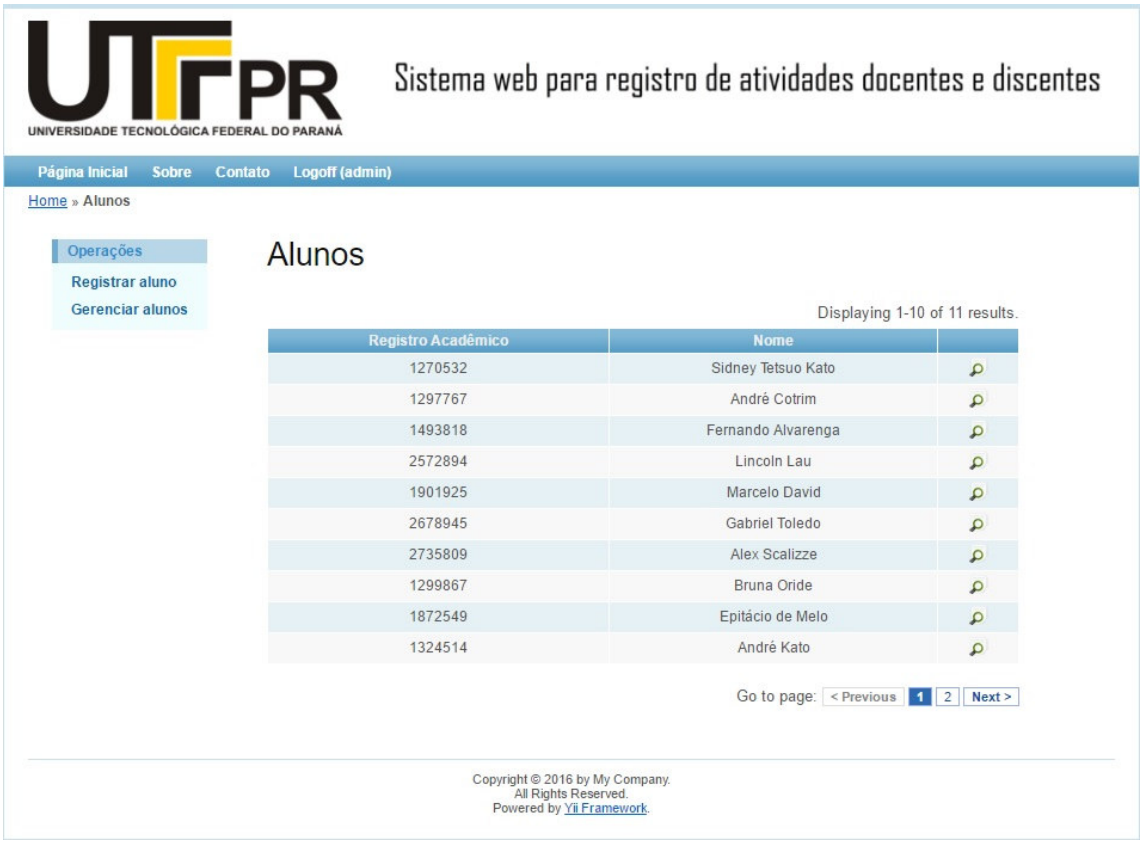

Figura 9 – Listagem com todos os registros da entidade escolhida

Ao clicar no código do registro, será exibida uma página contendo mais informações sobre o mesmo, assim como mostra a Figura 10. Nesta página, as operações de atualização, exclusão e geração de relatório tornam-se disponíveis para o registro em visualização, além da opção de voltar à listagem anterior.

Figura 10 – Visualização detalhada do registro

A Figura 11 mostra o formulário com os campos necessários para a inserção de novas entradas na tabela Alunos.

Figura 11 – Inserção de novos registros

Caso o usuário tente inserir um novo registro sem fornecer os dados obrigatórios, o sistema realizará uma validação e apresentará mensagens de acordo com as informações que não foram preenchidas. A Figura 12 mostra um exemplo desse cenário.

Figura 12 – Validação de formulário

Durante a visualização do registro ou por meio da página de gerenciamento, é possível excluir o registro desejado. O sistema apresentará uma mensagem de confirmação, assim como mostrado na Figura 13.

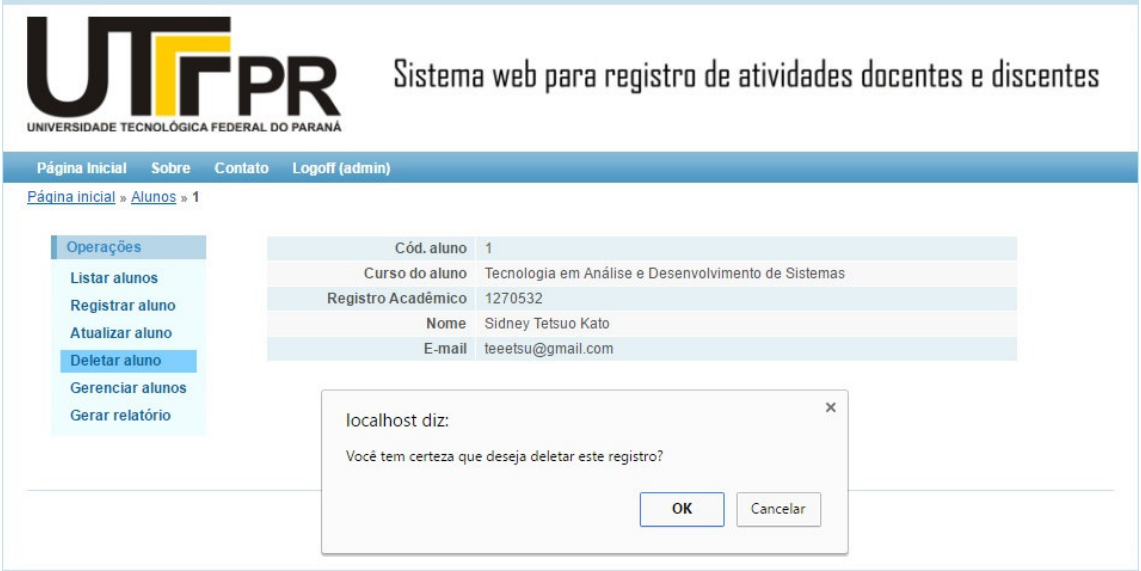

Figura 13 – Exclusão de registro

Se o usuário clicar na opção “Gerenciar” e ter a permissão necessária, uma página com a lista de todos os registros de determinada entidade será exibida. Nesta lista, três ações estão disponíveis, sendo elas: visualização, atualização e exclusão do registro referido. A Figura 14 mostra a página acessível somente ao administrador do sistema.

Figura 14 – Página de gerenciamento

4.4 IMPLEMENTAÇÃO DO SISTEMA

O arquivo que define o leiaute da página inicial pode ser encontrado na pasta "protected/views/layouts" da aplicação. Nesta pasta, encontram-se três arquivos:

- main.php: arquivo que define o leiaute do sistema.
- column1.php e column2.php: arquivos com blocos de códigos que podem ser inseridos no leiaute principal, não sendo de uso obrigatório.

A Listagem 1 apresenta o código correspondente ao leiaute padrão do sistema. O cabeçalho possui um bloco de divisão contendo a imagem correspondente ao logotipo e nome da aplicação, redirecionando o fluxo de navegação à página inicial quando clicado.

O bloco identificado por “mainmenu” contém um *widget*, sendo este uma instância da classe *CWidget* ou de suas classes derivadas, sendo principalmente utilizado para apresentação de dados. Os *widgets* normalmente são aplicados dentro de códigos de visão (*views*). Esta instância exibe um menu com os itens desejados.

Em seguida, outro *widget* é criado contendo os *breadcrumbs* (“migalhas de pão”). Esse recurso é um tipo de navegação auxiliar utilizado para revelar a localização do usuário dentro de uma aplicação web e possibilitar o retorno para as páginas anteriores distribuídas de forma hierárquica. .

O código “<?php echo $content; ?>” faz com que o conteúdo processado pelas *views* seja exibido no leiaute por meio do método *render* que é utilizado nos arquivos de Controle. Este método retorna uma *String* com o conteúdo da visão requisitada, sendo transferida para o leiaute por meio da variável *$content*.

O bloco final contém o rodapé da página reservado às informações sobre direitos autorais e de tecnologia de implementação.

```
<body>

<div class="container" id="page">

        <div id="header">
                <div id="logo">
                    <?php echo CHtml::link(CHtml::image(Yii::app()-
>request->baseUrl."/images/logo.jpg","logoUTFPR"), Yii::app()->request-
>baseUrl . '/index.php'); ?>
                </div>
        </div><!-- header -->
        <div id="mainmenu">
                <?php $this->widget('zii.widgets.CMenu',array(
                        'items'=>array(
                                array('label'=>'Página Inicial',
'url'=>array('/site/index')),
                                array('label'=>'Sobre',
'url'=>array('/site/page', 'view'=>'about')),
                                array('label'=>'Contato',
'url'=>array('/site/contact')),
                                array('label'=>'Login',
'url'=>array('/site/login'), 'visible'=>Yii::app()->user->isGuest),
                                array('label'=>'Logoff ('.Yii::app()->user-
>name.')', 'url'=>array('/site/logout'), 'visible'=>!Yii::app()->user-
>isGuest)
                        ),
                )); ?>
        </div><!-- mainmenu -->
        <?php if(isset($this->breadcrumbs)):?>
                <?php $this->widget('zii.widgets.CBreadcrumbs', array(
                        'links'=>$this->breadcrumbs,
                )); ?><!-- breadcrumbs -->
        <?php endif?>
```

```
        <?php echo $content; ?>

        <div class="clear"></div>

        <div id="footer">
                Copyright &copy; <?php echo date('Y'); ?> by My
Company.<br/>
                All Rights Reserved.<br/>
                <?php echo Yii::powered(); ?>
        </div><!-- footer -->

</div><!-- page -->

</body>
```

Listagem 1 – Main.php

A Listagem 2 exibe o código do arquivo de Controle "pai", localizado em "protected/components", sendo todos os outros arquivos de Controle derivados deste. A declaração "$layout='//layouts/column1';" indica que o arquivo de Controle deverá utilizar o leiaute indicado pela variável "$layout", exceto quando especificado de outra forma em um arquivo de controle específico.

```
class Controller extends CController
{
        /**
         * @var string the default layout for the controller view. Defaults
to '//layouts/column1',
         * meaning using a single column layout. See
'protected/views/layouts/column1.php'.
         */
        public $layout='//layouts/column1';
        /**
         * @var array context menu items. This property will be assigned to
{@link CMenu::items}.
         */
        public $menu=array();
        /**
         * @var array the breadcrumbs of the current page. The value of
this property will
         * be assigned to {@link CBreadcrumbs::links}. Please refer to
{@link CBreadcrumbs::links}
         * for more details on how to specify this property.
         */
        public $breadcrumbs=array();
}
```

Listagem 2 – Controller.php

A Listagem 3 apresenta o código do leiaute com 1 (uma) coluna. O método "beginContent" recebe o conteúdo do arquivo "main.php", processa-o e substitui o conteúdo da variável "$content" (em main.php) pelo conteúdo que está entre o "beginContent" e "endContent". A variável "$content" existente no arquivo "column1.php" passa a receber o

conteúdo adquirido pelo método *render* utilizado na classe de Controle. O arquivo "column2.php" funciona da mesma forma, porém com a adição de um painel para edição de dados localizado à direita da parte central da página.

```
<?php $this->beginContent('//layouts/main'); ?>
<div id="content">
        <?php echo $content; ?>
</div><!-- content -->
<?php $this->endContent(); ?>
```

Listagem 3 – Column1.php

O gerenciamento de acesso e autenticação de usuários é realizado por meio da classe "CUserIdentity" utilizando *login* e senha, localizado em "protected/components". O código desta classe é disponibilizado na listagem 4. A função "authenticate" define um *array* onde cada item especifica um *login* e senha de usuário. Como a aplicação em desenvolvimento requere apenas 2 (dois) tipos de *login* – administrador e professores –, não há necessidade de realizar a autenticação por meio de uma base de dados. As linhas seguintes verificam se as credenciais disponibilizadas são inválidas e, em caso positivo, geram códigos de erro referentes ao problema encontrado.

```
class UserIdentity extends CUserIdentity
{
        public function authenticate()
        {
                $users=array(
                        // username => password
                        'admin'=>'admin',
                        'professor'=>'professor',
                );
                if(!isset($users[$this->username]))
                        $this->errorCode=self::ERROR_USERNAME_INVALID;
                elseif($users[$this->username]!==$this->password)
                        $this->errorCode=self::ERROR_PASSWORD_INVALID;
                else
                        $this->errorCode=self::ERROR_NONE;
                return !$this->errorCode;
        }
}
```

Listagem 4 – UserIdentity.php

Para implementar as operações *Create, Retrieve, Update and Delete* (CRUD), ao invés de escrever todo o código, pode-se utilizar o gerador de código Gii. Após logar-se, deverá ser criada, primeiramente, a classe de modelo em "Model Generator". Essa classe permitirá que seja feita a comunicação com as tabelas do banco de dados de forma orientada a objetos. A

tela para a criação de uma classe modelo por meio do gerador de código do Yii é apresentada na Figura 15.

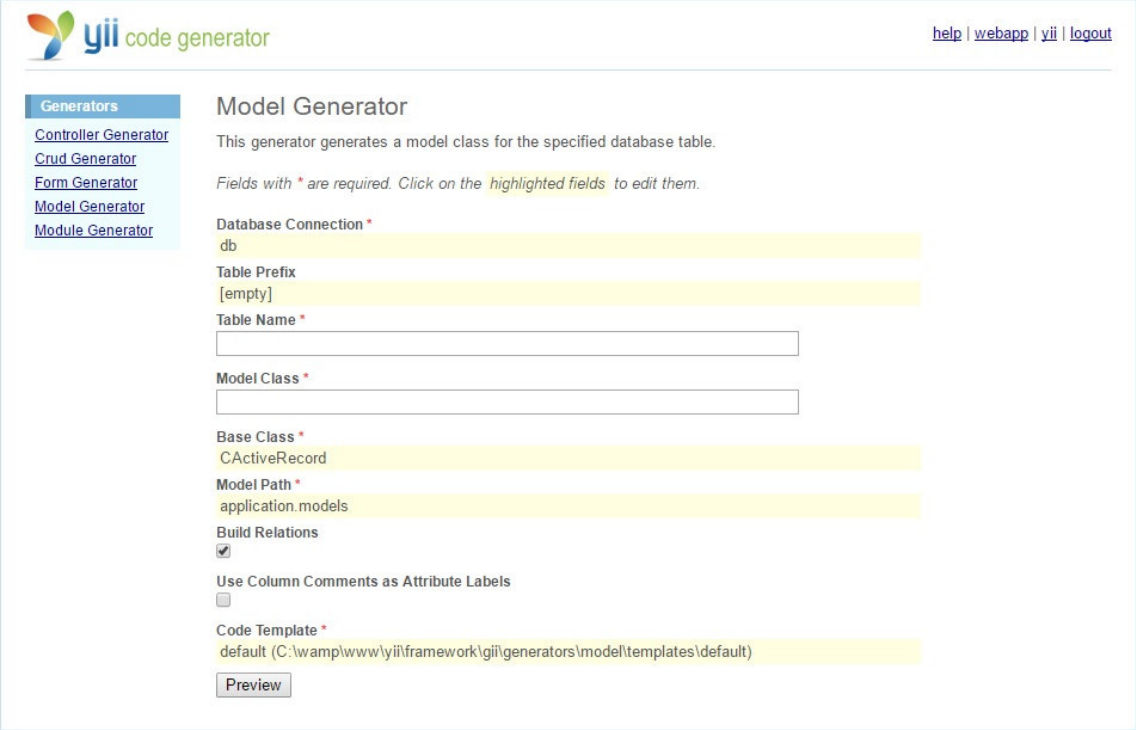

Figura 15 – Gerador de código de Modelo

Na tela da Figura 15, em “Table Name” deve ser informado o nome da entidade para a qual se deseja criar as operações CRUD. O Gii completará automaticamente o nome da classe em “Model Class”, podendo este ser modificado. Clicando no botão “Preview”, a aplicação mostrará o caminho de criação da classe.

Após criar a classe de modelo, o código para implementação das operações CRUD poderão ser geradas na opção “Crud Generator” (tela da Figura 16).

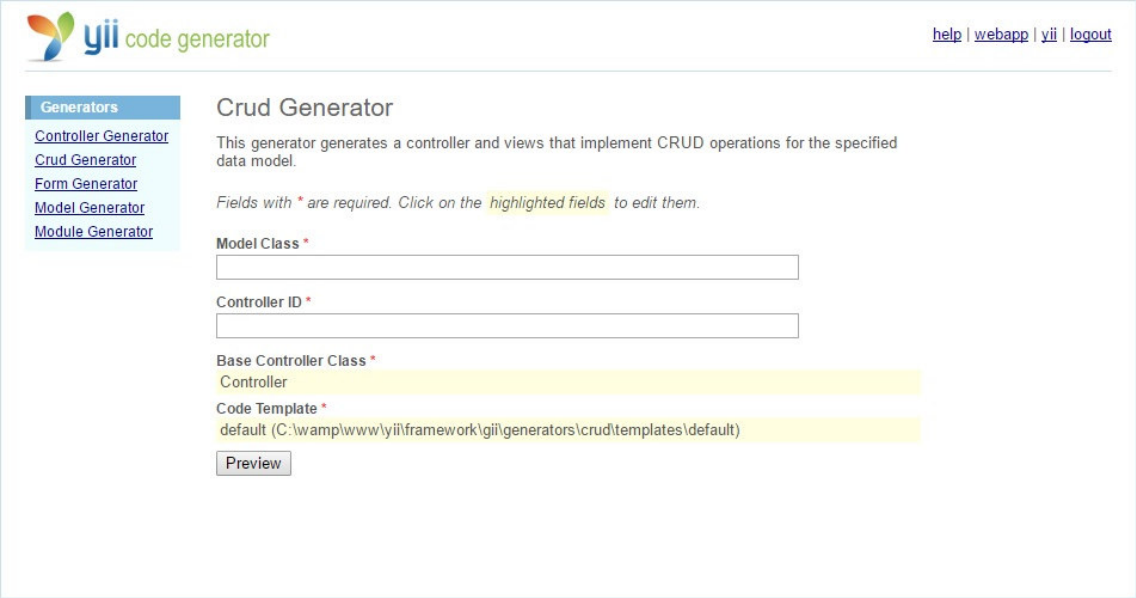

Figura 16 – Gerador de código CRUD

Na Figura 16, em “Model Class”, deve ser informada a classe de modelo criada anteriormente. Em “Controller ID”, pode ser utilizado o nome gerado automaticamente. Clicando no botão “Preview”, seguido de “Generate”, o processo de geração do código para operações CRUD, localizados em "C:\wamp\www\nome-do-projeto\protected\controllers”, é completado.

As especificações das regras de acesso ao controlador são apresentadas na listagem 5, a qual exibe a função “accessRules”, localizada em cada arquivo de controle. Um *array* com outros *arrays* é retornado. O primeiro elemento de cada *array* é a regra (permitir ou negar). O segundo elemento indica quais ações receberão essas configurações de acesso, enquanto o terceiro elemento indica à quais usuários essa regra será aplicada, sendo “*” todos os usuários, “@” usuários autenticados e “admin” o administrador do sistema.

O último *array* que possui a regra de “negar” especifica que nenhum usuário poderá acessar as ações que não estejam definidas nas linhas acima.

```
public function accessRules()
{
        return array(
                array('allow', // allow any user to perform 'report' action
                        'actions'=>array('report'),
                        'users'=>array('*'),
                ),
                array('allow', // allow authenticated user to perform
'index', 'view', 'create' and 'update' actions
                        'actions'=>array('index','view','create','update'),
                        'users'=>array('@'),
                ),
```

```
                array('allow', // allow admin user to perform 'admin' and
'delete' actions
                        'actions'=>array('admin','delete'),
                        'users'=>array('admin'),
                ),
                array('deny',  // deny all users
                        'users'=>array('*'),
                ),
        );
}
```

Listagem 5 – Função accessRules() em AlunosController

A Listagem 6 apresenta o código para a criação de novos registros. A função "actionCreate()" possui uma variável "$model" que instanciará uma nova classe de "Alunos". O *controller* renderizará a ação na *view* "create.php", passando a variável "model" como parâmetro. Após inserir os dados da nova entrada, o método será acessado novamente, verificando que o *array* de variáveis passado para o *script* atual foi definido. A nova instância da classe receberá os atributos deste *array* e ao salvar os dados, o *controller* redirecionará o fluxo de execução para a função "actionView($id)", passando o valor do atributo "aluno_cod" como parâmetro.

```
public function actionCreate()
        {
                $model=new Alunos;

                // Uncomment the following line if AJAX validation is
needed
                // $this->performAjaxValidation($model);

                if(isset($_POST['Alunos']))
                {
                        $model->attributes=$_POST['Alunos'];
                        if($model->save())
                                $this->redirect(array('view','id'=>$model-
>aluno_cod));
                }

                $this->render('create',array(
                        'model'=>$model,
                ));
        }
```

Listagem 6 – Função actionCreate()

Na Listagem 7 está a função "loadModel($id)" que localiza e retorna o registro de acordo com o *id* fornecido. Caso o registro não seja localizado, uma exceção é lançada.

```
public function loadModel($id)
 {
          $model=Alunos::model()->findByPk($id);
          if($model===null)
                  throw new CHttpException(404,'The requested page does not
exist.');
         return $model;
}
```

Listagem 7 – Função loadModel($id)

O código da Listagem 8 utiliza a função "loadModel($id)" no momento em que o *controller* renderizar a ação na *view* "view.php", passando o registro retornado como parâmetro.

```
         public function actionView($id)
         {
                  $this->render('view',array(
                           'model'=>$this->loadModel($id),
                  ));
         }
```

Listagem 8 – Função actionView($id)

Na Listagem 9 está o código para a atualização de registros. A variável "$model" receberá o registro retornado de acordo com o *id* fornecido. O *controller* renderizará a ação na *view* "update.php", passando a variável "model" como parâmetro. Após atualizar os dados, o método será acessado novamente, verificando que o *array* de variáveis passado para o *script* atual foi definido. Essa instância da classe "Alunos" receberá os novos atributos deste *array* e, ao salvar os dados, o *controller* redirecionará o fluxo de execução para a função "actionView($id)", passando o valor do atributo "aluno_cod" como parâmetro.

```
        public function actionUpdate($id)
{
         $model=$this->loadModel($id);

// Uncomment the following line if AJAX validation is needed
         // $this->performAjaxValidation($model);

         if(isset($_POST['Alunos']))
         {
                  $model->attributes=$_POST['Alunos'];
                  if($model->save())
                           $this->redirect(array('view','id'=>$model-
>aluno_cod));
         }

         $this->render('update',array(
                  'model'=>$model,
         ));
}
```

Listagem 9 – Função actionUpdate($id)

A função “actionDelete($id)” da Listagem 10 localiza e retorna o registro desejado de acordo com o *id* fornecido através da função “loadModel($id)”, para então excluí-lo utilizando o método “delete()” herdada da Classe “CActiveRecord”.

```
public function actionDelete($id)
{
        $this->loadModel($id)->delete();

        if(!isset($_GET['ajax']))
                $this->redirect(isset($_POST['returnUrl']) ?
$_POST['returnUrl'] : array('admin'));
}
```

Listagem 10 – Função actionDelete()

A Listagem 11 mostra o código para a ação “index”, sendo executada para a exibição da página inicial da entidade “Alunos” (Figura 9 – Listagem com todos os registros da entidade escolhida). O *controller* renderizará a ação na *view* “index.php”, passando um provedor de dados da tabela desejada (neste caso, a tabela “Alunos”), o qual será utilizado posteriormente para apresentação de dados por meio de um *widget*.

```
public function actionIndex()
{
        $dataProvider=new CActiveDataProvider('Alunos');
        $this->render('index',array(
                'dataProvider'=>$dataProvider,
        ));
}
```

Listagem 11 – Função actionIndex()

O código do *widget* que disponibilizará os dados na *view* “index.php” é apresentado na Listagem 12. O primeiro item do *array* indica o provedor de dados recebido anteriormente, enquanto o segundo item se refere à *view* parcial chamada “_view.php” que apresentará cada registro existente.

```
<?php $this->widget('zii.widgets.CListView', array(
        'dataProvider'=>$dataProvider,
        'itemView'=>'_view',
)); ?>
```

Listagem 12 – *Widget* para exibição de dados

A Listagem 13 apresenta o código para gerenciamento de registros, acessível somente ao administrador. Nas duas primeiras linhas, uma instância do modelo “Alunos” é criada como um contêiner dos parâmetros de busca. O método “unsetAttributes()” é chamado para garantir que os parâmetros de busca iniciais estejam vazios. O *controller* renderizará a ação na

view “admin.php”, passando a variável “model” (até então vazio) como parâmetro. Ao definir os atributos que serão utilizados na busca de registros por meio do formulário de pesquisa, a função “actionAdmin()” será acessada novamente, desta vez com seus parâmetros de busca estabelecidos.

```
public function actionAdmin()
{
        $model=new Alunos('search');
        $model->unsetAttributes();  // clear any default values
        if(isset($_GET['Alunos']))
                $model->attributes=$_GET['Alunos'];

        $this->render('admin',array(
                'model'=>$model,
        ));
}
```

Listagem 13 – Função actionAdmin()

A Figura 15 apresenta o formulário de pesquisa da view “admin.php” que definirá os atributos a serem utilizados como parâmetros de busca.

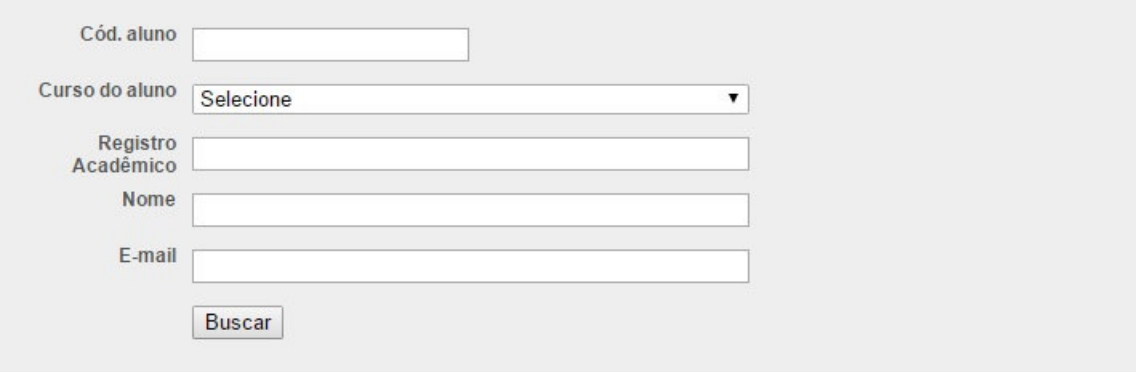

Figura 17 – Formulário de pesquisa

O método “search()” localizado nos arquivos de modelo é uma parte vital do código quando se deseja utilizar a classe *CGridView* ou *CListView* em *widgets* na sua aplicação. Esse método é apresentado na Listagem 14. Uma nova instância da classe *CDbCriteria* é criada representando um critério de consulta, tais como condições e ordenação.

O código apresentado a seguir constrói uma consulta SQL, onde o método *with*, *compare* e *order* equivalem, respectivamente, as cláusulas *join*, *where* e *order by*.

```php
public function search()
{
        $criteria=new CDbCriteria;
        $criteria->with = 'cursos';

        $criteria->compare('aluno_cod',$this->aluno_cod);
        $criteria->compare('cursos.curso_nome',$this->nomeCurso,true);
        $criteria->compare('aluno_ra',$this->aluno_ra,true);
        $criteria->compare('aluno_nome',$this->aluno_nome,true);
        $criteria->compare('aluno_email',$this->aluno_email,true);

        $criteria->order = 'aluno_ra';

        return new CActiveDataProvider($this, array(
                'criteria'=>$criteria,
        ));
}
```

Listagem 14 – Função search() em Alunos.php

A listagem 15 representa o código do *widget* que será utilizado na página "admin.php" para exibir os dados retornados pelos critérios de busca da função "search()". As colunas são preenchidas por *arrays*, no qual cada *array* possui o atributo "name", "value" e "htmlOptions" representando, respectivamente, o nome da coluna, valor e opções de formatação HTML.

```php
<?php $this->widget('zii.widgets.grid.CGridView', array(
        'id'=>'alunos-grid',
        'dataProvider'=>$model->search(),
        'filter'=>$model,
        'columns'=>array(
                array(
                        'name'=>'aluno_cod',
                        'value'=>'$data->aluno_cod',
                        'htmlOptions'=>array(
                            'style' => 'text-align: center',
                        ),
                ),
                array(
                        'name'=>'nomeCurso',
                        'value'=>'$data->cursos->curso_nome',
                        'htmlOptions'=>array(
                            'style' => 'text-align: center',
                        ),
                ),
                array(
                        'name'=>'aluno_ra',
                        'value'=>'$data->aluno_ra',
                        'htmlOptions'=>array(
                            'style' => 'text-align: center',
                        ),
                ),
                array(
                        'name'=>'aluno_nome',
```

```
                        'value'=>'$data->aluno_nome',
                        'htmlOptions'=>array(
                            'style' => 'text-align: center',
                        ),
                ),
                array(
                        'name'=>'aluno_email',
                        'value'=>'$data->aluno_email',
                        'htmlOptions'=>array(
                            'style' => 'text-align: center',
                        ),
                ),
                array(
                        'class'=>'CButtonColumn',
                ),
        ),
)); ?>
```

Listagem 15 – *Widget* para exibição dos retornos da função "search()"

A Listagem 16 mostra o código para a ação "report", semelhante ao código para a ação "admin". Primeiramente é criada uma instância do modelo "AtividadesPapeis" como um contêiner dos parâmetros de busca, servindo como provedor de dados. Para que seja possível apresentar os dados de determinado aluno no relatório, a função "loadModel($id)" é utilizada, associando seu retorno à variável "$modelAluno". Por fim, o controller renderizará a ação na view "report.php", passando o provedor de dados e o registro do aluno específico como parâmetros.

```
public function actionReport($id)
{
        $dataProvider = new AtividadesPapeis('report');
        $modelAluno = $this->loadModel($id);
        $this->render('report',array(
                'dataProvider'=>$dataProvider,
                'modelAluno'=>$modelAluno,
        ));
}
```

Listagem 16 – Função actionReport()

O método "relatorioAluno($id)" localizado nos arquivos de modelo funciona da mesma forma que o método "search()", retornando um provedor de dados contendo os resultados da consulta. Esse método é apresentado na Listagem 17.

```
public function relatorioAluno($id)
{
        $criteria=new CDbCriteria;
        $criteria->alias = 'ap';
        $criteria->with = array(
                'papeis',
                'alunos',
                'professores',
```

```
                'projetos',
                'atividadesRealizadas',
        );
        $criteria->condition = "ap.aluno_cod = " . $id;

        return new CActiveDataProvider($this, array(
                'criteria'=>$criteria,
        ));
}
```

Listagem 17 – Função relatorioAluno($id) em Alunos.php

O código da Listagem 18 representa a *view* "report.php" que exibe o relatório contendo informações do aluno ou professor, e as atividades em que participaram. O código do primeiro *widget* mostra os dados do aluno ou professor, e o segundo mostra detalhes das atividades realizadas, papel desempenhado, projeto (caso faça parte de algum) e tipo da atividade.

```
<?php
$this->layout = '//layouts/column1';
/* @var $this AlunosController */
/* @var $modelAluno Alunos($id) */
/* @var $dataProvider AtividadesPapeis */
?>

<h1 align='center'>Relatório de atividades por aluno</h1>

<h3>• Aluno</h3>

<?php $this->widget('zii.widgets.grid.CGridView', array(
        'id'=>'atividades-papeis-grid2',
        'dataProvider'=>$modelAluno->search(),
        'columns'=>array(
                array(
                    'name'=>'aluno_cod',
                    'value'=>'$data->aluno_cod',
                    'htmlOptions'=>array(
                        'style' => 'text-align: center',
                    ),
                ),
                array(
                    'name'=>'aluno_nome',
                    'value'=>'$data->aluno_nome',
                    'htmlOptions'=>array(
                        'style' => 'text-align: center',
                    ),
                ),
                array(
                    'name'=>'nomeCurso',
                    'value'=>'$data->cursos->curso_nome',
                    'htmlOptions'=>array(
                        'style' => 'text-align: center',
                    ),
                ),
                array(
                    'name'=>'aluno_ra',
```

```
                    'value'=>'$data->aluno_ra',
                    'htmlOptions'=>array(
                        'style' => 'text-align: center',
                    ),
                ),
                array(
                    'name'=>'aluno_email',
                    'value'=>'$data->aluno_email',
                    'htmlOptions'=>array(
                        'style' => 'text-align: center',
                    ),
                ),
        ),
)); ?>

<h3>• Atividades registradas</h3>

<?php $this->widget('zii.widgets.grid.CGridView', array(
        'id'=>'atividades-papeis-grid',
        'dataProvider'=>$dataProvider->relatorioAluno($modelAluno-
>aluno_cod),
        'columns'=>array(
                array(
                        'name'=>'nomePapel',
                        'value'=>'$data->papeis->papel_nome',
                        'htmlOptions'=>array(
                            'style' => 'text-align: center',
                        ),
                ),
                array(
                        'name'=>'nomeAtividade',
                        'value'=>'$data->atividadesRealizadas->ar_nome',
                        'htmlOptions'=>array(
                            'style' => 'text-align: center',
                        ),
                ),
                array(
                        'name'=>'nomeProjeto',
                        'value'=>'$data->projetos == NULL ? "" : $data-
>projetos->projeto_nome',
                        'htmlOptions'=>array(
                            'style' => 'text-align: center',
                        ),
                ),
                array(
                        'name'=>'nomeProfessorProjeto',
                        'value'=>'$data->projetos == NULL ? "" : $data-
>projetos->professores->prof_nome',
                        'htmlOptions'=>array(
                            'style' => 'text-align: center',
                        ),
                ),
                array(
                        'name'=>'nomeTipoAtividade',
                        'value'=>'$data->atividadesRealizadas-
>tiposAtividades->ta_nome',
                        'htmlOptions'=>array(
                            'style' => 'text-align: center',
                        ),
```

```
                ),
                array(
                        'name'=>'dataInicial',
                        'value'=>'$data->atividadesRealizadas->ar_dataini
== NULL ? "" : date("d/m/Y", strtotime($data->atividadesRealizadas-
>ar_dataini))',
                        'htmlOptions'=>array(
                            'style' => 'text-align: center',
                        ),
                ),
                array(
                        'name'=>'dataFinal',
                        'value'=>'$data->atividadesRealizadas->ar_datafin
== NULL ? "" : date("d/m/Y", strtotime($data->atividadesRealizadas-
>ar_datafin))',
                        'htmlOptions'=>array(
                            'style' => 'text-align: center',
                        ),
                ),
                array(
                        'name'=>'duracao',
                        'value'=>'$data->atividadesRealizadas->ar_duracao
== NULL ? "" : $data->atividadesRealizadas->ar_duracao',
                        'htmlOptions'=>array(
                            'style' => 'text-align: center',
                        ),
                ),
        ),
)); ?>
```

Listagem 18 – Report.php

5 CONCLUSÃO

O objetivo deste trabalho foi implementar um sistema para registrar as atividades docentes e discentes realizadas no âmbito do Departamento Acadêmico a que eles pertencem, facilitando a consulta de atividades realizadas e simplificando a atribuição de pontos em atividades complementares. O DAINF foi utilizado como o Departamento para a base de levantamento dos requisitos, mas o aplicativo pode ser utilizados por outros Departamentos, outras Universidades e mesmo por empresas que tenham o interesse em manter um controle de atividades realizadas que são voltadas para a capacitação, por exemplo.

A aplicação foi desenvolvida para a plataforma *web*, pela facilidade de acesso e manutenção, utilizando recursos que a caracterizam como uma RIA. Para esse desenvolvimento, foram utilizadas diversas ferramentas e tecnologias.

A principal ferramenta utilizada foi o *Yii*, sendo este um *framework* considerado de alto desempenho em PHP que utiliza componentes para o desenvolvimento aplicações *web*. Esse *framework* provê uma ampla reutilização de código e é baseado no padrão de arquitetura de *software* MVC.

Para o desenvolvimento do trabalho foi utilizado o gerador de códigos Gii que trabalha vinculado ao Yii. Essa tecnologia disponibiliza alguns geradores de códigos, cada um responsável por criar um tipo específico de código, como o gerador de arquivos de controle, que cria uma classe de controle juntamente de alguns *scripts* de ação para as *views*, e o gerador de arquivos de modelo, que gera uma classe específica para manipulação de determinada tabela do banco de dados.

A utilização dessa ferramenta para o desenvolvimento do trabalho foi de grande ajuda, principalmente pelo recurso de criar blocos de código automaticamente, acelerando, de forma significativa, a implementação. Nos momentos em que houve dificuldade de interpretação na forma em que deveria ser utilizado algum método, pôde-se recorrer à documentação bem detalhada no próprio site do *framework*.

As expectativas a cerca dessa tecnologia foram atendidas, desde o momento em que a aplicação de um padrão de projetos foi facilitada, até o momento em que diversas funcionalidades mostraram-se aplicáveis de forma simplificada, como a facilidade em criar operações CRUD e a simplicidade em lidar com relacionamentos entre entidades.

Uma desvantagem de usar essa tecnologia é o suporte dado pela comunidade que ainda não é tão grande se comparado a outras tecnologias de desenvolvimento web.

REFERÊNCIAS

AST, Philipp; KAPFENBERGER, Michael; HAUSWIESNER, Stefan. 2008. **Crawler approaches and technology**. Disponível em: <http://www.iicm.tugraz.at/cguetl/courses/isr/uearchive/uews2008/Ue01%20-%20Crawler-Approaches-And-Technology.pdf>. Acesso em: 25 jan. 2016.

BERGMAN, Michael K. The deep web: surfacing hidden value. **Journal of Electronic Publishing**, v. 7, n. 1. 2001. Disponível em: <http://quod.lib.umich.edu/j/jep/3336451.0007.104?view=text;rgn=main>. Acesso em: 30 out. 2016.

BOZZON, Alessandro; COMAI, Sara; FRATERNALI, Piero; CARUGHI, Giovanni, T. Conceptual modeling and code generation for rich internet applications. **ACM Press**, New York, 2006, p. 353–360.

COMAI, Sara; CARUGHI, Giovanni T. A behavioral model for rich internet applications in web engineering, **LNCS**, v.4607, 2007, p.364-369.

COOPER, Ezra; LINDLEY, Sam; WADLER, Philip; YALLOP, Jeremy. links: web programming without tiers, in formal methods for components and objects, **LNCS**, v. .4709, 2007, p.266-269.

DUHL, Joshua. White paper: Rich Internet Applications**. Technical report, IDC**, November 2003, p. 1-33. Disponível em: <https://www.adobe.com/platform/whitepapers/idc_impact_of_rias.pdf>. Acesso em: 25 jan. 2016.

DWORAK, Hugo. **A concept of a web application blending thin and fat client architectures**. In: Fourth International Conference on Dependability of Computer Systems, 2009, p. 84-90.

HAMMOND, David. **Web browser DOM support**. 2008. Disponível em: <http://www.webdevout.net/browser-support-dom>. Acesso em: 25 jan. 2016.

LAKSHMAN, Pratap. **JScript deviations from ES3***,* Microsoft Corporation, p. 1-87, 2007. Disponível em: <http://wiki.ecmascript.org/lib/exe/fetch.php?media=resources:jscriptdeviationsfromes3.pdf>. Acesso em: 25 jan. 2016.

MYSQL. **MySQL Workbench**. Disponível em: <https://www.mysql.com/products/workbench/>. Acesso em: 20 jan. 2016.

PHP. Disponível em: <http://php.net/manual/pt_BR/intro-whatis.php>. Acesso em: 29 jan. 2016.

POWELL, Courtney; NAKAMURA, Keisuke; AKAMA, Kiyoshi. **Towards a formal behavioral model for rich internet applications**. In: International Conference on Computational Intelligence and Software Engineering, 2009, p. 1-5.

PRECIADO, Juan Carlos; LINAJE, Marino; SANCHEZ, Fernando; COMAI, Sara. **Necessity of methodologies to model rich internet applications**. In: WSE 2005 - 7th IEEE Int. Symposium on Web Site Evolution, 2005, p. 7–13.

SILVA, Diogo A. da. PHP Yii Framework. **DevMedia**. Disponível em: <http://www.devmedia.com.br/php-yii-framework/30898>. Acesso em: 20 jan. 2016.

STEARN, Brent. **XULRunner**: a new approach for developing rich internet applications. IEEE Internet Computing, v.11, n.3, 2007, p.67-73.

SU, Jui-Yuan; SUN, Der-Johng; WU, I-Chen; CHEN, Lung-Pin. On design of browser-oriented data extraction system and the plug-ins. **Journal of Marine Science and Technolog**, v. 18, n. 2, p. 189-200, 2010. Disponível em: <http://jmst.ntou.edu.tw/marine/18-2/189-200.pdf>. Acesso em: 20 ago. 2016.