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

DEISE PEGORARO

SISTEMA WEB COM PHP E FLEX PARA REGISTRO DE ORIENTAÇÃO DE ATIVIDADES ACADÊMICAS

TRABALHO DE CONCLUSÃO DE CURSO

PATO BRANCO
2013

DEISE PEGORARO

SISTEMA WEB COM PHP E FLEX PARA REGISTRO DE ORIENTAÇÃO DE ATIVIDADES ACADÊMICAS

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
2013

ATA Nº: **207**

DEFESA PÚBLICA DO TRABALHO DE DIPLOMAÇÃO DA ALUNA **DEISE PEGORARO**.

Às 09:25 hrs do dia 18 de abril de 2013, Bloco V da UTFPR, Câmpus Pato Branco, reuniu-se a banca avaliadora composta pelos professores Beatriz Terezinha Borsoi (Orientadora), Andréia Scariot Beulke (Convidada) e Vinicius Pegorini (Convidado), para avaliar o Trabalho de Diplomação da aluna Deise Pegoraro, matrícula 1030701, sob o título **Sistema Web com PHP e Flex para Registro de Orientação de Atividades Acadêmicas**; 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 a candidata foi entrevistada pela banca examinadora, e a palavra foi aberta ao público. Em seguida, a banca reuniu-se para deliberar considerando o trabalho **APROVADO**. Às 10:05 hrs foi encerrada a sessão.

Profa. Beatriz Terezinha Borsoi, Dr.
Orientadora

Profa. Andréia Scariot Beulke, Esp.
Convidada

Prof. Vinicius Pegorini, Esp.
Convidado

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

Prof. Edilson Pontarolo, Dr.
Coordenador do Curso

RESUMO

PEGORARO. Deise. **Sistema web com PHP e Flex para registro de orientação de atividades acadêmicas**. 2013. 50 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, 2013.

Os acadêmicos realizam diversas atividades nas quais é necessário que o acompanhamento realizado pelo professor responsável seja registrado. Dentre essas atividades estão os trabalhos de conclusão de curso, estágios e de iniciação científica, tecnológica ou extensão. Esse registro normalmente é realizado em um arquivo texto. Visando facilitar o trabalho do professor e permitir que o próprio aluno possa documentar a orientação propõe-se o desenvolvimento de um sistema *web* desenvolvido com PHP e Flex para o registro da orientação dessas atividades e geração de relatórios.

Palavras-chave: Sistemas *web*. Linguagem PHP. Registro de orientação de atividades acadêmicas.

ABSTRACT

PEGORARO. Deise. **Web system developed with PHP and Flex to record the orientation of academic activities**. 2013. 50 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, 2013.

University students perform activities in which it is necessary the monitoring of a teacher. And the orientation of these activities must be registered. Among these activities are the completion of course work, internships and undergraduate research, technological or extension. This registration is usually held in a text file. To facilitate the work of the teacher and allow the students themselves to document orientation of these activities it is proposed the development of a web-based system to record the orientation of these activities and reporting.

Keywords: Web systems. PHP. Record of academic orientation.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTAGENS DE CÓDIGO

LISTA DE ABREVIATURAS E SIGLAS

CSS	*Cascading Style Sheet*
HTML	*HyperText Markup Language*
IP	*Internet Protocol*
MVC	*Model-View-Controller*
MXML	*Minimal XML*
SQL	*Structure Query Language*
TCP	*Transmission Control Protocol*
UML	*Unified Modeling Language*
XHTML	*Extensible HyperText Markup Language*
XML	*Extensible Markup Language*

SUMÁRIO

1 INTRODUÇÃO

Este capítulo apresenta as considerações inicias, com uma visão geral do trabalho, os objetivos e a justificativa do mesmo e a organização do texto.

1.1 CONSIDERAÇÕES INICIAIS

Nas atividades de orientação acadêmica os professores acompanham o trabalho sendo realizado pelos alunos. Esse acompanhamento consiste em orientar a realização das tarefas de responsabilidade dos acadêmicos, verificar o andamento das tarefas sendo realizadas, discutir problemas, soluções e resultados, realizar e avaliar o planejamento do trabalho, dentre outros. As atividades de orientação acadêmica geralmente precisam ser registradas. Esse registro é uma forma de gerenciar o acompanhamento de atividades de estágio, de trabalhos de conclusão de curso e de iniciação científica, por exemplo.

Atualmente na Universidade Tecnológica Federal do Paraná, Câmpus Pato Branco, é comum que o registro seja realizado em fichas impressas ou em editor de texto. Verificou-se, assim, a possibilidade de implementar um sistema *web* que pudesse facilitar esse registro dessas atividades e ser utilizado pelo Departamento Acadêmico de Informática do Câmpus Pato Branco.

Nesse sistema o professor cadastra os seus orientados e faz o registro individual das orientações e a emissão dos relatórios de acompanhamento. Para a implementação desse sistema foi escolhida a linguagem PHP com o banco de dados MySQL. É um sistema simples, mas que visa fornecer praticidade para o registro da atividade de orientação que é realizada pelos professores.

Por ser um sistema *web* o mesmo poderá ser instalado localmente no computador do professor ou em um servidor da própria Universidade e assim ser acessado remotamente pelos professores de qualquer curso ou departamento.

1.2 OBJETIVOS

O objetivo geral apresenta o resultado principal do trabalho realizado e os objetivos específicos o complementam, no sentido de valores agregados.

1.2.1 Objetivo Geral

Implementar um sistema *web* para registro de orientação de atividades acadêmicas da UTFPR câmpus Pato Branco.

1.2.2 Objetivos Específicos

- Proporcionar uma maneira facilitada de o professor registrar as atividades de orientação acadêmica de alunos;
- Possibilitar a emissão de relatório de acompanhamento de orientações acadêmicas.
- Possibilitar a padronização do documento dos registros de orientação das atividades acadêmicas.

1.3 JUSTIFICATIVA

Atualmente os professores registram suas atividades de orientação de trabalhos de conclusão de curso, estágio e projetos de pesquisa, através de um arquivo texto. O professor registra o conteúdo da reunião e em muitos casos imprime esse registro para ser entregue ao professor coordenador de estágio ou trabalho de conclusão de curso, por exemplo.

Um sistema *web* para esse tipo de registro seria bastante útil porque o professor poderia fazer esse registro a partir de qualquer computador com acesso à Internet e com um navegador *web*. O desenvolvimento de um sistema *web* visa facilitar o acesso ao mesmo pelo professor. Assim, o registro da orientação pode ser realizado pelo professor logo após a atividade de orientação.

A escolha das tecnologias, especificamente PHP e Flex, foi decorrente da facilidade percebida no desenvolvimento com a linguagem PHP. Essa linguagem foi utilizada em uma das disciplinas do curso. O interesse de desenvolvimento de uma aplicação *web* com interface semelhante às aplicações *desktop* decorre da ampla ênfase que esse tipo de aplicação têm atualmente. Flex foi escolhido para implementar uma interface rica pela facilidade de uso com PHP.

O desenvolvimento desse sistema se justifica por facilitar o registro das atividades de orientação realizadas pelo professor, bem como a emissão de relatórios de acompanhamento.

1.4 ORGANIZAÇÃO DO TEXTO

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

O Capítulo 2 contém o referencial teórico que fundamenta a proposta conceitual do sistema desenvolvido. O referencial teórico está baseado no desenvolvimento de aplicações *web*, ou seja, para ambiente Internet e no padrão *Model-View-Controller* (MVC) que é utilizado na implementação do sistema.

No Capítulo 3 estão os materiais e o método empregados no desenvolvimento deste trabalho.

O Capítulo 4 contém o sistema desenvolvido, com exemplos de documentação da modelagem e de implementação. A modelagem é exemplificada por documentos de análise e projeto. A implementação é exemplificada pela apresentação do sistema com telas e descrição de suas funcionalidades e por exemplos da codificação do sistema.

No Capítulo 5 estão as considerações finais.

2 DESENVOLVIMENTO PARA AMBIENTE INTERNET

Este capítulo apresenta o referencial teórico utilizado para fundamentar o sistema proposto.

2.1 Contexto

A Internet surgiu a partir de 1969, por meio do desenvolvimento de uma rede de computadores conhecida como Arpanet. O objetivo dessa rede era utilizar uma tecnologia revolucionária de transmissão de dados por computação por pacotes (CASTELLS, 2004).

Desde então, essa tecnologia foi sendo aprimorada, iniciando pelas redes de comunicação que a Arpa estava administrando, integrando o conceito de uma rede de redes. Para tratar a comunicação entre essas redes era necessário um protocolo de comunicação padronizado. Em 1973 surge o TCP (*Transmission Control Protocol*) que é o Protocolo de Controle de Transmissão que em 1978 foi dividido em duas partes, acrescentando um protocolo IP (*Internet Protocol*) gerando o TCP/IP que é o padrão utilizado até hoje.

Um crescimento significativo da Internet ocorreu em 1990, quando Tim Berners-Lee desenvolveu a World Wide Web, como uma aplicação para compartilhamento de informações. A partir da proposta inicial de Berners-Lee diversos navegadores *web*, os denominados *browsers*, surgiram. Esses navegadores *web* permitem a acesso e a disponibilização de conteúdo *web*.

Como uma das conseqüências do surgimento da *web* está a criação de um mundo virtual que permite realizar compras, pagamentos, acessar conteúdos, entretenimento, interação entre pessoas, realizar transações comerciais entre empresas geograficamente distintas, dentre muitas outras sem necessidade de deslocamento físico de quem está realizando essas atividades. O uso intensivo e amplo da Internet e da *web* fez surgir a necessidade de aplicações que atendessem essa demanda, que suprissem as necessidades e interesses dos negócios, do comércio, dos bancos, dos setores de entretenimento e educação.

As primeiras aplicações *web* consistiam em vários arquivos vinculados entre si e em um mesmo arquivo estavam instruções de processamento, formatação, conteúdo, dentre outros. É a ideia das páginas desenvolvidas com a linguagem de apresentação HTML (*HyperText Markup Language*) e com o conteúdo das páginas compostos por arquivos distintos ligados por meio de *hiperlinks*.

As páginas *web* consistiam de um conjunto de documentos que poderiam se ligar com outros por meio de *hiperlinks*. Contudo, eram arquivos de textos simples que continham conteúdos e *links* estáticos. Com a o uso da Internet e dos seus serviços para aplicações comerciais, foram sendo desenvolvidas formas de estruturar o desenvolvimento de páginas *web* incorporando o conceito de componentes individuais que poderiam se relacionar, facilitando a construção de aplicações *web*.

Com o tempo, a geração, por parte do servidor *web*, de páginas dinâmicas tornou-se um processo de difícil gerenciamento para muitas aplicações. Essa dificuldade era proveniente do conteúdo e da formatação no código da aplicação estarem mesclados em um mesmo arquivo. Surge uma estrutura híbrida em que partes pequenas de código seriam executados nas páginas *web* quando solicitados e os resultados seriam inseridos em uma estrutura predefinida para exibição na página. Isso trouxe maior flexibilidade e tornou mais fácil a reutilização de componentes das páginas, porém ainda era difícil fazer manutenção dos códigos ao longo de todas as páginas de um site.

O modelo utilizado para desenvolver os sites *web* foi evoluindo e sendo melhorado. Dessas evoluções surge o padrão de projetos MVC (*Model-View-Controller*), que de forma sumária segmenta uma aplicação em visão (interface do sistema), modelo (os dados), controle (a lógica de negócio da aplicação). Essa arquitetura de desenvolvimento fornece uma maneira de dividir as funcionalidades envolvidas na apresentação e na manutenção dos dados.

2.2 Model-View-Controller

MVC foi descrito em 1979, por Trygve Reenskaug, que trabalhava com Smalltalk na Xerox PARC e não é um conceito totalmente novo (WANG, 2011).

A arquitetura MVC divide um sistema em três partes (MCHEICK, QI, 2011): modelo, visão e controle. É importante que a cisão esteja separada da estrutura de dados. O controle é implementado como um componente separado ou combinado com a visão. Para Burbeck (1997) citado por Mcheick e Qi (2011), o conceito básico de MVC é apresentado como: o modelo lida com a lógica de negócio e com os dados da aplicação e ele é responsável por atualizar a informação na visão e receber os comandos do controle. A visão é responsável pela apresentação da informação; o controle é responsável pelas estradas do usuário incluindo eventos do mouse e teclado e por notificar a modelo por meio de eventos.

No projeto de aplicações *web*, MVC é implementado pelo sistema de *template web* como uma visão para o componente *web* (WANG, 2011). MVC é frequentemente visto como uma aplicação *web* na qual a visão é o HTML ou XHTML (*Extensible HyperText Markup Language*) gerado pela aplicação (WANG, 2011). O controle recebe entradas *get* ou *post* e decide o que fazer com as mesmas realizando operações no domínio de objetos (o modelo) que contém as regras de negócio e o conhecimento para realizar tarefas específicas tais como processa um novo método que controla a geração de componentes (X)HTML como máquinas (*engines*) de *templates*, *pipelines* XML, chamadas Ajax, dentre outras (WANG, 2011).

O padrão de projetos MVC tem apresentado seus benefícios por meio de aplicações interativas que permitem múltiplas representações da mesma informação, promovendo a reutilização de código e auxiliando os desenvolvedores a concentrar-se em um único aspecto da aplicação (SELFA, CARRILLO, BOONE, 2006).

É fato que sistemas de informação devem ser personalizados para diferentes necessidades dos usuários de acordo com o papel em termos de negócio que desempenham no uso dos aplicativos considerado. Por exemplo, um vendedor não necessita da mesma visão de dados e dos negócios que um gerente estratégico da empresa. Os dados de vendas e outros são os mesmos, mas a forma de apresentação (organização, combinação, agrupamentos e outras operações realizadas com os dados) não é a mesma. Esse tipo de situação é traduzido pela necessidade de ter diferentes visões para a mesma informação (conjunto de dados) que será mostrada de acordo com interesses e necessidades dos usuários do sistema (SELFA, CARRILLO, BOONE, 2006).

O padrão MVC foi inicialmente projetado para interface com o usuário em aplicações implementadas com Smalltalk. Contudo, a partir disso tornou-se um paradigma de projeto para interface com o usuário sem haver necessidade de preocupação com a linguagem de implementação e para aplicações *web* para as quais os componentes de controle mudam frequentemente (SELFA, CARRILLO, BOONE, 2006).

A arquitetura MVC é ilustrada na Figura 1 que divide um sistema interativo em três componentes cada um dos quais especializados em uma tarefa (SELFA, CARRILLO, BOONE, 2006).

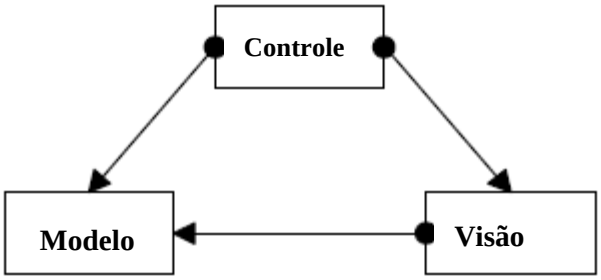

Figura 1 – Estrutura padrão do modelo MVC
Fonte: Microsoft (2013, p. 1).

As partes do modelo MVC (como representado na Figura 1), de acordo com o apresentado em Selfa, Carrillo, Boone (2006), Cui et al. (2009) são:

a) Controle - o controle define o comportamento da aplicação, gerenciando as suas funcionalidades da mesma. O controle processa e responde eventos, geralmente ações do usuário, e pode indiretamente invocar mudanças no modelo. Ele é responsável por interpretar as ações do usuário e as mapear para chamada do modelo. Após o processamento do modelo, o controlador exibe uma visualização como parte da resposta à solicitação do usuário. Normalmente existe um controlador para cada conjunto de funcionalidades relacionadas. O controle interpreta as entradas do usuário, comandando o modelo e a visão para ser alterada adequadamente.

b) Visão - a visão gerencia a apresentação visual do modelo e o *feedback* para o usuário. Visões diferentes podem existir a partir de um único modelo para objetivos distintos.

c) Modelo – representa os dados e as regras de negócio da aplicação e fornece ao controlador acesso as funcionalidades da aplicação. Direciona o conteúdo de entrada de uma parte particular do modelo e encaminha ao controlador, também acessa os dados do modelo e define como devem ser apresentados. O modelo é a representação do domínio dos dados sobre os quais opera a aplicação. A lógica de domínio adiciona significado aos dados. Muitas aplicações usam um mecanismo de armazenamento persistente (como um banco de dados) para armazenar os dados. MVC não menciona especificamente a camada de acesso a dados porque é subentendido que é responsabilidade do modelo ou mesmo que está encapsulada no modelo.

Pela ilustração da Figura 1, o padrão MVC pode ser passivo, no sentido de o modelo não saber da existência da visão e do controle (SELFA, CARRILLO, BOONE, 2006). Por exemplo, se o modelo é um texto que somente pode ser alterado pelo usuário. Contudo, na maioria dos casos o modelo deve ter um vínculo (link) com a visão para informar as mudanças feitas em seu estado causada por procedimentos internos. A visão e o controle estão

sempre conectados. O controle comunica-se com a visão para determinar quais objetos estão sendo manipulados pelo usuário e para chamar métodos do modelo para realizar mudanças nesses objetos. O modelo realiza as mudanças e notifica a visão para que seja atualizada.

A Figura 2 apresenta a estrutura de um *framework* MVC. Nesse *framework* uma classe controle faz requisições a uma classe modelo e funções para realizar uma tarefa específica. Então o controle obtém as mensagens de retorno. Em seguida chamará um arquivo de visão para apresentar o retorno. O controle usa a mensagem de retorno proveniente do modelo para preencher (apresentar) a visão e eles enviam uma página final para o cliente. Apesar disso, algumas classes modelo possuem suas próprias funções que podem chamar uma página de visão.

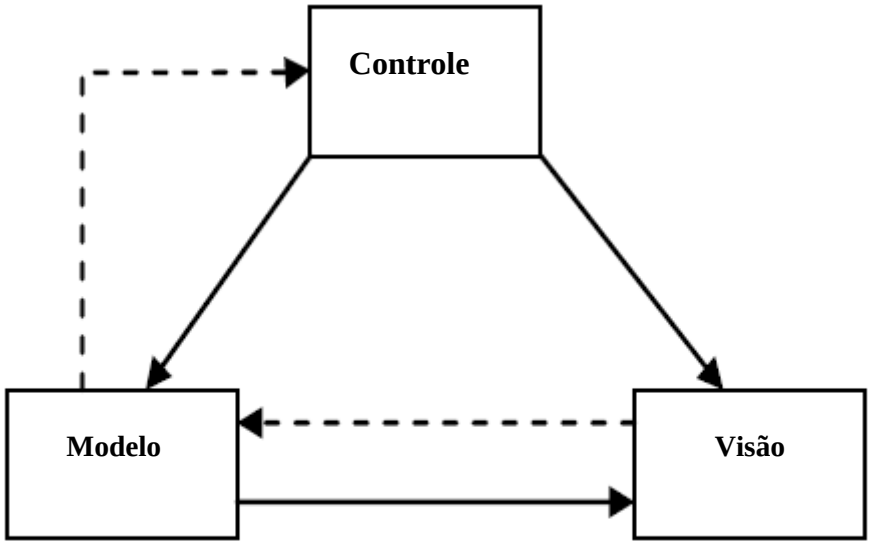

Figura 2 – Estrutura agregada do modelo MVC
Fonte: Wang (2011, p. 76).

A visão da aplicação pode incluir várias visões MVC aninhadas. Os controles dessas visões devem cooperar para assegurar que um controle apropriado está interpretando as entradas do usuário. Para esse objetivo eles formam uma árvore hierárquica na qual as mensagens passam de controle a controle por meios dos ramos da árvore. Somente o controle que tem o foco realiza a ação. Cada visão está associada com um único controle e vice-versa, mas o modelo pode ter ao mesmo tempo mais que um par de visão-controle. A cada vez que ocorrer mudanças no modelo, cada visão dependente pode ser notificada para que seja alterada de acordo, a possibilidade de haver múltiplas visões sincronizadas é um benefício significativo para a arquitetura MVC (DOBOS, 2002).

A arquitetura MVC visa resolver alguns problemas como (SELFA, CARRILLO, BOONE, 2006):

a) A mesma informação deve ser mostrada em diferentes formatos em visões diferentes.

b) Mudanças em uma visão devem ser refletidas nas demais.

c) Mudanças na interface com o usuário deveriam ser fáceis de realizar.

d) A funcionalidade central do sistema deve ser independente das interfaces para que permitir que várias coexistam.

As vantagens da arquitetura MVC com relação a outras arqutieturas são (LARMAN, 2007):

a) Menor acoplamento.
b) Alta coesão.
c) As visões provêem mais flexibilidade e agilidade: múltiplos modelos de visões podem ser criados para adicionar, modificar e eliminar visões dinamicamente. Visões podem ser aninhadas, a forma que a visão responde às interações do usuário pode ser mudada sem alterar o visual da representação. É possível haver visões diferentes de acordo com as capacidades dos dispostivos.
d) Maior clareza de projeto.
e) Facilidade de manutenção.
f) Maior escalabilidade.

A análise de robustez inicialmente proposta por Jacobson pode ser facilmente movida para a arquitetura MVC que propunha identificar os objetos por meio de casos de uso e classificá-los de acordo com o papel realizado. Essa classificação identifica de uma forma natural os objetos que estão inseridos nos componentes MVC e são (SELFA, CARRILLO, BOONE, 2006):

a) Objetos de entidades – são os objetos relacionados com dados persistentes que podem ser tabelas, arquivos, cache ou sessão de dados. É o modelo em MVC.

b) Objetos de fronteira – são os objetos que realizam a comunicação do sistema com seu ambiente. Eles podem ser telas, janelas, menus ou qualquer elemento de interface gráfica. É a visão do MVC.

c) Objetos de controle – objetos que realizam as ações dos casos de uso para filtrar dados a serem apresentados ao usuário. É o controle do MVC.

Essa análise define as seguintes regras (MUKHTAR, 2004):

a) Atores podem somente comunicar-se com objetos de fronteira.
b) Objetos de fronteira podem somente comunicar-se com controladores e atores.
c) Objetos de entidades podem somente comunicar-se com controladores.

d) Controladores podem somente comunicar-se com objetos de fronteira e objetos de entidade e outros controladores, mas não com atores.

MVC foi projeto para que aplicações pudessem prover múltiplas visões dos mesmos dados (TAO, 2002). Por meio do encapsulamento das três abstrações (modelo, visão e controle) em classes separadas, MVC minimiza o impacto das mudanças da interface com o usuário e aumenta a reusabilidade de objetos do domínio (TAO, 2002).

2.3 PHP e MVC

O rápido desenvolvimento da Internet para o desenvolvimento de aplicações *web* trouxe uma alta demanda por eficiência, confiabilidade, manutenibilidade e escalabilidade (CUI et al., 2009). A linguagem PHP possui características como a de ser intuitiva, fácil de manipular, execução rápida, multi-plataforma, código fonte aberto.

Cui (2009) ressalta que o modelo de desenvolvimento clássico adotado pelo PHP mescla código de acesso a dados, o processamento da lógica de negócio e a camada de apresentação e isso pode trazer problemas tanto de entendimento do código, como de manutenção. Cui et al. (2009) propôs um *framework* de desenvolvimento em PHP baseado no modelo de projetos MVC, fornecendo uma efetiva separação do acesso a dados, do processamento da lógica e da interface. O modelo proposto por esses autores aplica a ideia de Browser/Servidor em um modelo MVC. A camada *web* corresponde à visão e ao controle. A camada de lógica de negócio e de persistência dos dados são de responsabilidade do modelo (WEI, 2005).

Por causa das características da linguagem PHP que é uma linguagem de *script* que é incorporada em documentos HTML na implementação de um servidor. Isso resulta em um modo de programação PHP típico que é orientado a processo (WANG, 2011). Incorporar *script* PHP em documentos HTML é o formato tradicional de programação PHP.

A Listagem 1 apresenta um trecho de código típico de um programa PHP incorporado em HTML.

```
<!DOCTYPE html>
<html>
   <head>
      <meta charset="utf-8" />
   <title>PHP Test</title>
   </head>
   <body>
   <?php
       print(“Linguagem PHP”);
       /* outras funções*/
   ?>
  </body>
</html>
```

Listagem 1 – Exemplo de código PHP típico

No modo de codificar exemplificado na Listagem 1, todas as operações são escritas em uma única página HTML. Isso economiza tempo de requisição das páginas, mas se a página é complexa, com muitas linhas de código PHP fica muito difícil manter o código e, além disso, o reuso de código é bastante dificultado pois muitas vezes está mesclado com JavaScript, além de HTML e PHP.

O padrão MVC resolve alguns dos problemas no desenvolvimento com linguagem PHP.

a) A lógica de negócio implementada em PHP e a página HTML são separadas de forma a tornar a estrutura de código mais clara e facilitar a depuração do mesmo.

b) Modelos e classes são adequados na manutenção de código.

c) Com MVC, uma aplicação PHP pode ser organizada usando métodos de programação orientada a objetos.

3 MATERIAIS E MÉTODO

Este capítulo apresenta os materiais e o método utilizados na realização deste trabalho. Os materiais se referem às tecnologias como linguagens e ferramentas para a modelagem e a implementação do sistema. O método contém as etapas com os principais procedimentos utilizados para o desenvolvimento do sistema.

3.1 MATERIAIS

As ferramentas e tecnologias utilizadas para modelagem e implementação do sistema foram:

a) Visual Paradigm para a modelagem do sistema;
b) Balsamiq Mockups para desenvolver o protótipo da interface do sistema;
c) Aptana como ferramenta de desenvolvimento;
d) AdobeFlash Builder como ambiente de desenvolvimento;
e) PHP como linguagem para a implementação do sistema;
f) PHPMyAdmin para gerenciador de banco de dados;
g) AmfPHP para conexão entre cliente e servidor;
h) MySQL (MYSQL, 2012) para o banco de dados.

3.1.1 Visual Paradigm

Visual Paradigm for UML (VISUAL PARADIGM, 2013) é um ambiente de modelagem para UML, que auxilia na criação e edição de fluxos de trabalho. A ferramenta possibilita projetar todos os tipos de diagramas UML (*Unified Modeling Language*), modelar bancos de dados, gerenciar requisitos e gerar documentação.

Na Figura 3 é apresentado um *print screen* da tela da ferramenta Visual Paradigm for UML. Nesta figura são destacados os principais elementos que compõem a interface do aplicativo.

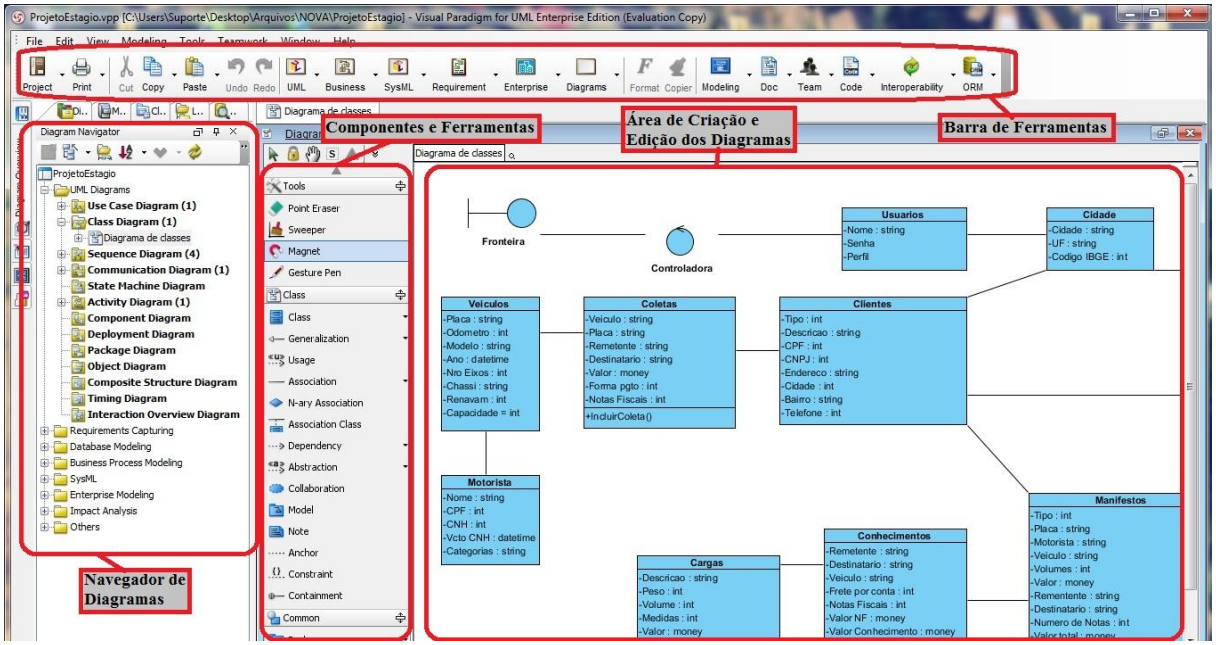

Figura 3 – Tela inicial da ferramenta *Visual Paradigm*

Nas partes destacadas da Figura 3 é possível identificar:

a) Barra de Ferramentas - apresenta os atalhos para as principais funcionalidades da ferramenta.

b) Navegador de diagramas - exibe todos os diagramas desenvolvidos e possui navegação subdividida por tipo de diagrama facilitando a localização dos mesmos.

c) Componentes e Ferramentas - fornece os elementos para compor os diagramas. Esses componentes são apresentados de acordo com o tipo de diagrama escolhido para composição.

d) Área de criação e edição de diagramas - é a parte da ferramenta na qual os diagramas são desenvolvidos e editados graficamente de acordo com os componentes e ferramentas disponíveis.

3.1.2 Balsamiq Mockups

Balsamiq Mockups é uma ferramenta para elaboração de protótipos de interface de sistemas. Balsamiq Mockups é uma aplicação desenvolvida em Flash e executa sobre o AIR da Adobe, o que permite executá-la em múltiplas plataformas (Mac/Linux/Windows) (GUERINI, 2013). A interface da ferramenta, apresentada na Figura 4, é bastante simples.

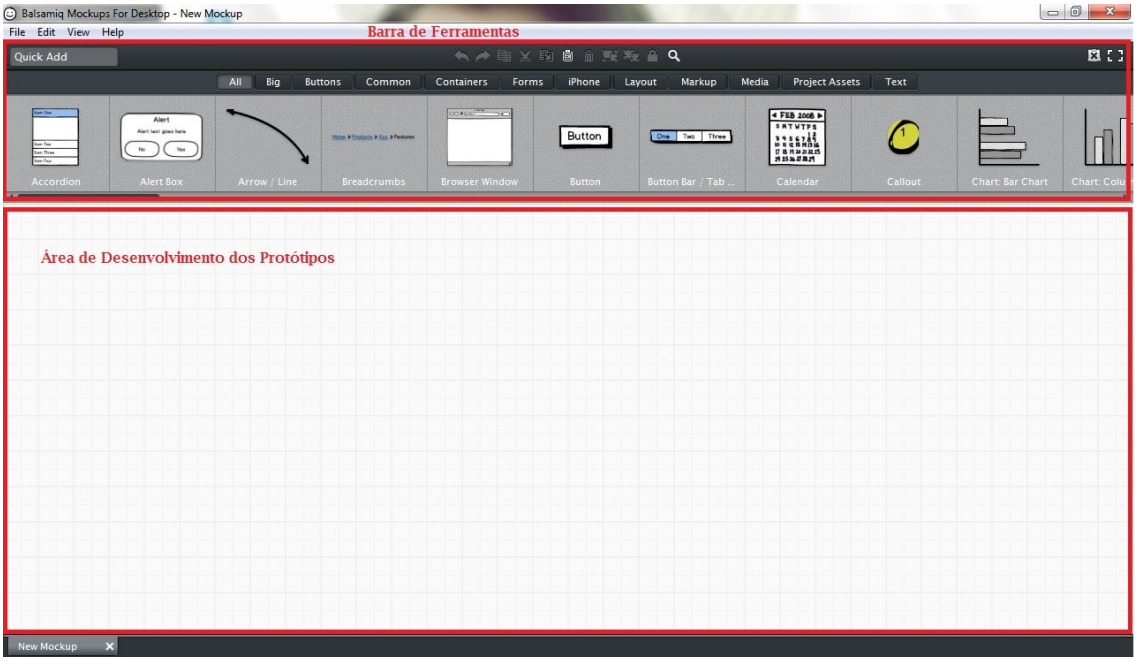

Figura 4 – Tela inicial da ferramenta Balsamiq Mockups

A interface da ferramenta consiste basicamente em (áreas destacadas na Figura 4):

a) Barra de ferramentas - na parte superior da tela são apresentados ícones e menus para os principais comandos e a opção Quick Add que é uma forma mais fácil de encontrar e adicionar controles. Logo abaixo estão os controles disponíveis divididos por categorias.

b) Área de criação dos protótipos - é a parte da ferramenta na qual serão arrastados os controles da barra de ferramentas para gerar o protótipo desejado.

3.1.3 Aptana

Aptana é uma ferramenta *open source* de desenvolvimento, baseada no ambiente Eclipse, que possibilita desenvolver e testar as aplicações em um único ambiente. É direcionada para o desenvolvimento *web* fornecendo suporte para várias tecnologias como HTML, CSS (*Cascading Style Sheets*), JavaScript, Ruby, Rails, PHP e Python (APTANA, 2013). A Figura 5 apresenta a interface da ferramenta Aptana. No desenvolvimento deste trabalho essa ferramenta foi utilizada para o desenvolvimento do código em PHP.

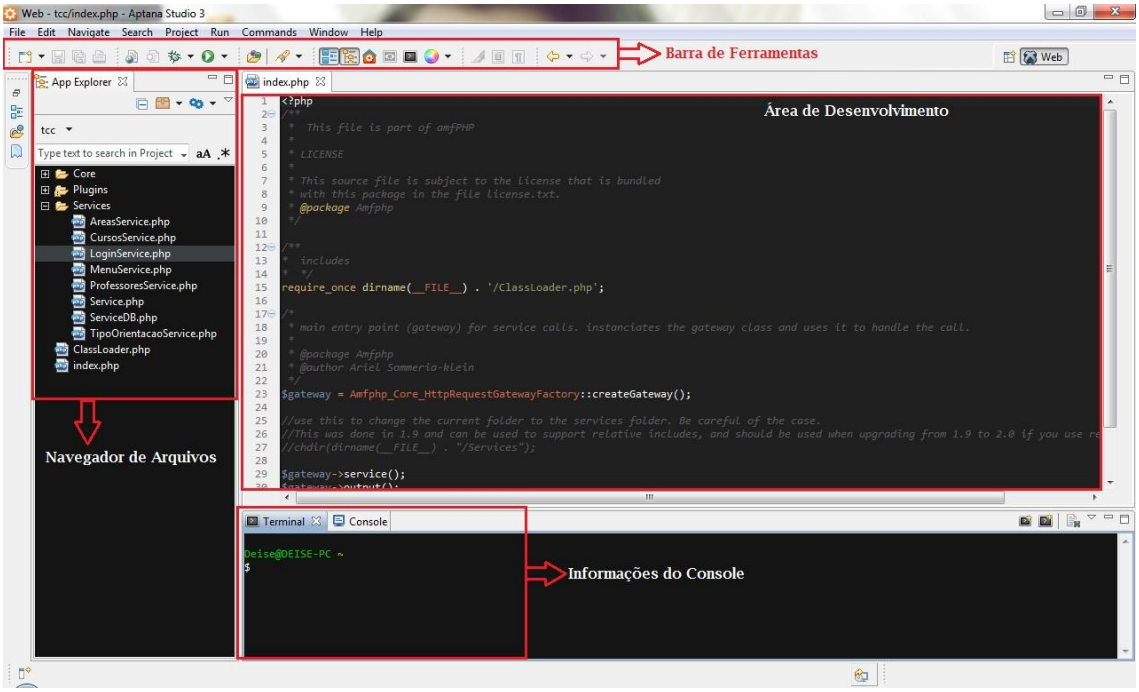

Figura 5 – Tela inicial da ferramenta Aptana

Na Figura 5, estão destacados os principais elementos que compõem a interface do aplicativo Aptana:

a) Barra de Ferramentas - apresenta os atalhos para as principais funcionalidades do aplicativo.

b) Navegador de Arquivos - exibe todos os tipos de documentos presentes no projeto.

c) Área de Desenvolvimento – área para desenvolvimento ou alteração do código fonte.

d) Informações do Console – para visualizar informações sobre erros, o terminal em que o usuário está utilizando a ferramenta, dentre outras.

3.1.4 Adobe Flash Builder

Adobe Flash Builder é uma ferramenta de desenvolvimento de software, um *framework open source* para construção de aplicações ricas para *web*, móvel ou *desktop* que pode executar em todos os principais *browsers* e que fornece um modelo de programação baseado em duas linguagens, Action Script e MXML (MOREIRA, 2009).

O MXML (*Minimal XML*) é uma linguagem de marcação baseada em XML (*Extensible Markup Language*) que facilita a criação de leiaute de interfaces para a aplicação, bem como implementa aspectos não visuais para as camadas que ligam a interface com o

banco de dados e vice-versa. Action Script é uma linguagem orientada a objetos para implementar funcionalidades do aplicativo *web*. A Figura 6 apresenta a tela inicial da ferramenta Adobe Flash Builder 4.5.

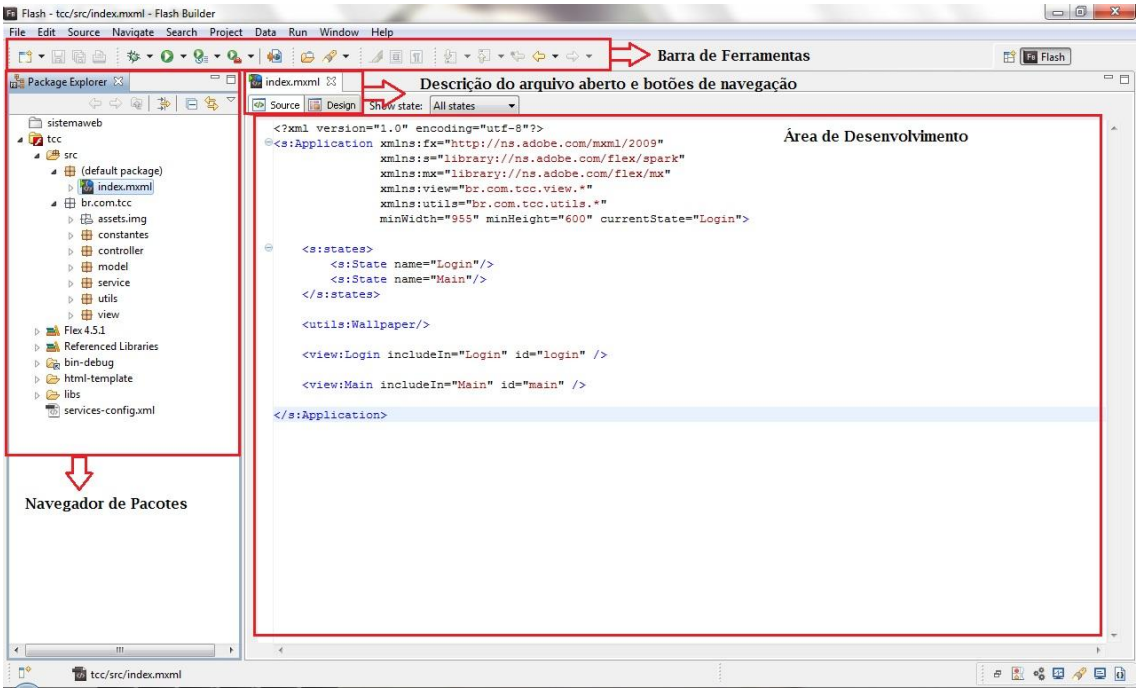

Figura 6 – Tela inicial da ferramenta Adobe Flash Builder 4.5

Na Figura 7 são apresentados os principais elementos que compõem a interface do aplicativo Adobe Flash Builder. Nas partes destacadas é possível identificar:

a) Barra de Ferramentas - apresenta os atalhos para as principais funcionalidades do aplicativo.

b) Navegador de Pacotes - exibe todos os pacotes ou arquivos desenvolvidos na aplicação e possui navegação subdividida por tipo de pacote ou pasta que facilita a localização dos arquivos.

c) Descrição do arquivo aberto e botões de navegação - exibe o arquivo que está disponível para desenvolvimento, podendo ser visualizada a interface utilizando os botões “Design” e “Source”, sendo respectivamente relativos à interface e ao código fonte da aplicação.

d) Área de Desenvolvimento - é a parte da ferramenta para o desenvolvimento ou alteração do código fonte do aplicativo.

3.1.5 PHP

PHP (um acrônimo recursivo para *PHP Hypertext Preprocessor*) é uma linguagem de *script open source* de uso geral, muito utilizada para o desenvolvimento de aplicações *web* (PHP, 2013). PHP é uma linguagem de programação de código aberto utilizada para desenvolvimento de aplicações *web* presentes no lado do servidor que permite incorporar seus fragmentos de código dentro da página HTML gerando a página web.

PHP permite incorporar código dentro da página HTML e realizar determinadas tarefas de uma forma fácil e eficaz. Uma das principais vantagens de PHP relativamente às outras linguagens pensadas para os CGI é que ela oferece inúmeras funções para a exploração de bases de dados de um modo fácil (ALVAREZ, 2013).

O que distingui PHP de JavaScript, por exemplo, no lado do cliente é que o código é executado no servidor, gerando HTML que é então enviado para o cliente. O cliente recebe os resultados da execução desse *script*, mas não tem acesso ao código fonte. É possível inclusive configurar o servidor para processar todos os arquivos HTML como PHP e assim os usuários não têm como saber se está ou não sendo utilizada a linguagem PHP (PHP, 2013).

3.1.6 PHPAdmin

PhpMyAdmin é uma ferramenta de *software* livre escrito em PHP, criado para fazer a administração do MySQL sobre as páginas *web*. Essa ferramenta suporta várias formas de operações com o MySQL, principalmente, para o gerenciamento de bancos de dados, das tabelas, dos campos, das relações, dos usuários, das permissões, entre outros e possibilita executar diretamente qualquer função SQL (*Structure Query Language*) (PHPADMIN, 2013).

3.1.7 AmfPHP

AmfPHP é um *software* livre e código aberto que fornece uma forma simples de conectar uma aplicação cliente com um servidor. Usado em projetos que visam o desenvolvimento de aplicativos que se comunicam com a Internet.

É uma ferramenta para criar serviços acessíveis, possibilitando que o desenvolvedor se concentre em recursos e funções exclusivas do projeto enquanto a comunicação entre cliente e servidor é facilmente realizada, pois possibilita que o aplicativo servidor com seus

webservices teste esses serviços desenvolvidos diretamente no navegador (*browser*) (AMFPHP, 2013).

Na Figura 7 é possível visualizar a tela inicial da ferramenta, na qual são exibidos os serviços (*services*) criados no projeto, que podem ser acessados diretamente pelo navegador.

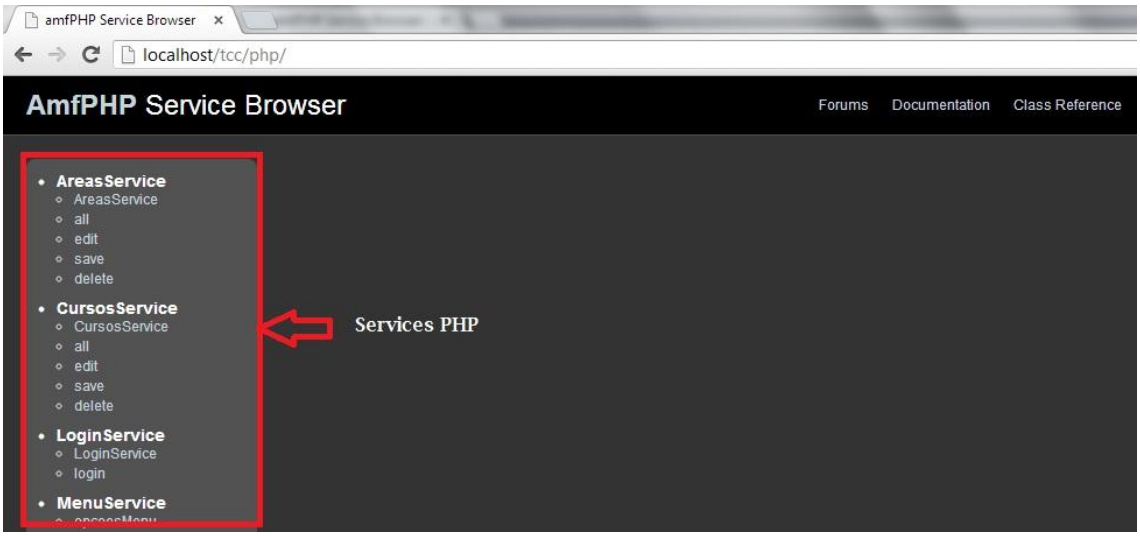

Figura 7 – Tela inicial da ferramenta AmfPHP

3.1.8 MySQL

O MySQL é uma ferramenta que gerencia bancos de dados de forma relacional, utiliza código aberto e é usado na maioria das aplicações gratuitas para gerenciar suas bases de dados (MYSQL, 2013). MySQL utiliza a linguagem SQL para inserir, acessar e gerenciar o conteúdo armazenado num banco de dados (PISA, 2012).

3.2 MÉTODO

O método utilizado para a realização do trabalho está baseado nas fases de análise, projeto, codificação e testes do modelo sequencial linear exposto em Pressman (2005). Porém, ressalta-se que as atividades realizadas não foram estritamente sequenciais. A modelagem e mesma a definição dos requisitos foi complementada e ajustada à medida que o desenvolvimento ocorria. A seguir a descrição das fases realizadas:

a) Planejamento

No planejamento do projeto foi definido como seria a realização do trabalho, o escopo

do referencial teórico, as tecnologias a serem utilizadas e as ferramentas para modelagem e implementação.

Na reunião inicial foi discutido o sistema. Reuniões periódicas com a orientadora foram realizadas permitindo avaliar e ajustar as atividades planejadas e estabelecer prazo para o término de cada atividade que seria realizada. Periodicamente foram trocadas mensagens eletrônicas para o acompanhamento das atividades.

b) Definição dos Requisitos

A ideia do sistema foi sugestão da orientadora. Com o desenvolvimento das atividades a autora deste trabalho sugeriu outros requisitos e ajustes na ideia inicial.

Como forma de expor a ideia geral do sistema um diagrama com a visão geral do mesmo foi definido. Esse diagrama representado por um conjunto de conceitos conectados entre si permitiu visualizar como seria o sistema em termos de funcionalidades, mas sem preocupação de definir atores, classes, tabelas ou outras informações específicas da modelagem.

Também foi elaborado um modelo do relatório que deveria ser a impresso como resultado de cada orientação. A partir dessa definição inicial do sistema, os requisitos funcionais e não funcionais foram definidos.

c) Análise e projeto

As telas foram prototipadas utilizando a ferramenta Balsamiq Mockups visando fornecer melhor entendimento da interação do usuário com o sistema. Em seguida os casos de uso, as classes e entidades e relacionamentos do banco de dados foram elaboradas.

e) Implementação

Para a implementação foi utilizada a ferramenta Aptana visando facilitar a codificação e a depuração dos erros do sistema.

Na implementação algumas das telas que haviam sido prototipadas foram alteradas. Neste texto é mantida a versão original da tela feita como protótipo para retratar o que realmente ocorreu no projeto.

f) Testes

Os testes foram informais e realizados pela autora deste trabalho visando encontrar erros de codificação. A orientadora realizou testes visando verificar a usabilidade do sistema e o atendimento aos requisitos.

4 RESULTADOS

Este capítulo apresenta o sistema que foi desenvolvido como resultado deste trabalho.

4.1 APRESENTAÇÃO DO SISTEMA

O sistema se destina ao controle de acompanhamento de orientação de atividades acadêmicas. Para essa orientação há um professor orientador e um aluno. O registro de cada tarefa de orientação é realizado pelo professor ou mesmo pelo acadêmico. Um relatório pode ser impresso como comprovante da orientação.

4.2 MODELAGEM DO SISTEMA

A visão geral do sistema está centrada nas suas funcionalidades essenciais é apresentada na Figura 8.

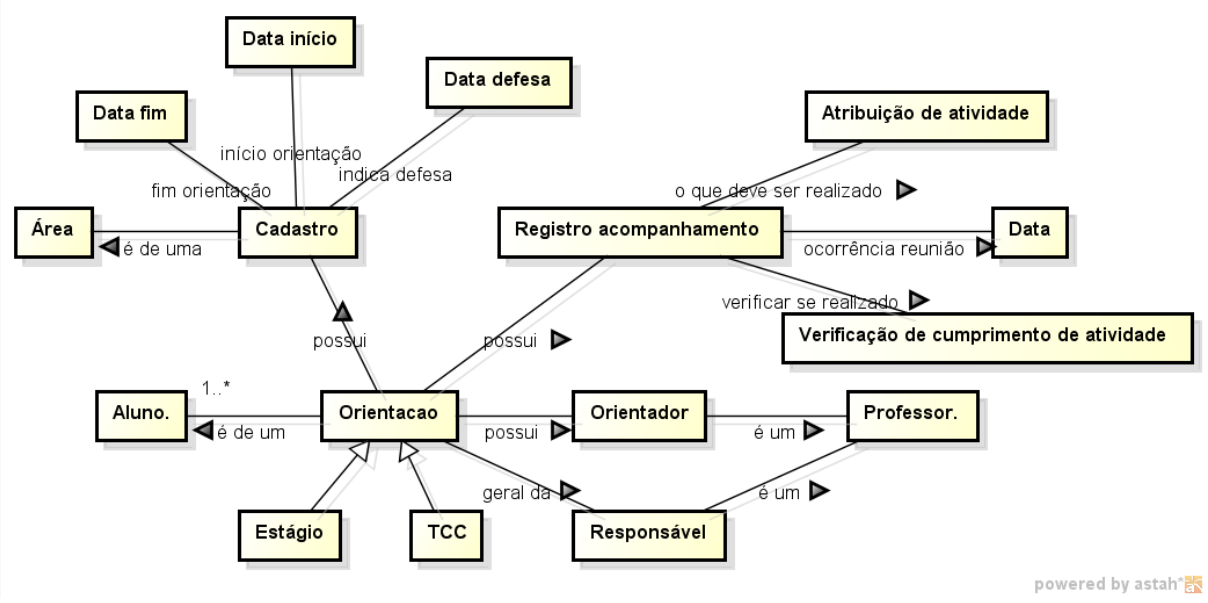

Figura 8 – Visão geral do sistema

A listagem do Quadro 1 apresenta os requisitos funcionais identificados para o sistema.

Identificação	Nome	Descrição
RF001	Cadastro de alunos	Alunos cadastrados no sistema. O cadastro deverá incluir o RA do aluno.
RF002	Cadastro de professores	Professores possíveis orientadores.
RF003	Cadastro de tipo de orientação	Uma orientação pode ser de estágio, trabalho de conclusão de curso, monografia de especialização, dissertação, tese, iniciação científica, atividade de projeto, dentre outras.
RF004	Cadastro da orientação	Cadastro da orientação, com nome do(s) aluno(s), data de início e final, título e descrição do trabalho, data de defesa e área.
RF005	Cadastro de área	As áreas dos trabalhos.
RF006	Registro das atividades de acompanhamento	É o registro da orientação, onde o professor orientador pode fazer esse registro. O registro conterá a data, a verificação se do cumprimento de atividades e o registro de orientação. O registro conterá um campo geral para resumir a orientação e atividades a serem feitas. Essas atividades serão verificadas na próxima reunião e deverá ser possível informar se foram cumpridas ou não.

Quadro 1 – Listagem dos requisitos funcionais

A listagem do Quadro 2 apresenta os requisitos não-funcionais identificados para o sistema. Os requisitos não funcionais explicitam regras de negócio, restrições ao sistema de acesso, por exemplo, requisitos de qualidade, desempenho, segurança e outros.

Identificação	Nome	Descrição
RNF001	Cadastro de aluno	Somente o usuário administrador pode cadastrar o aluno. O aluno não realiza *login* no sistema.
RNF002	Acompanhamento	Somente o professor orientador pode manter o registro de orientação dos seus orientados.
RNF003	Atribuição de orientado para orientador	Se o professor cadastrar o aluno ele será seu orientado. Se o cadastro é feito pelo administrador, esse pode atribuir um professor para o orientado. Somente o administrador pode alterar o orientador de um orientado
RNF004	Orientados	Um trabalho pode ter mais de um aluno orientado vinculado.
RNF005	Login	O sistema terá acesso por meio de email e senha previamente cadastrados.

Quadro 2 – Listagem dos requisitos não-funcionais

A Figura 9 apresenta o modelo do relatório de acompanhamento das atividades realizadas durante a orientação que deverá ser gerado pelo sistema. Em “Atividade” deverá ser indicado o tipo de atividade realizado que pode ser estágio, trabalho de conclusão de

curso, iniciação científica ou outra. A data de defesa deverá constar apenas se a atividade possui uma apresentação formal.

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Câmpus Pato Branco
Departamento de Informática

Ficha de Registro de Orientação e Acompanhamento de "Atividade"

Acadêmico(a):	
Orientador(a):	
Período:	
Data da Defesa:	
Área:	

DATA	ATIVIDADES A SEREM DESENVOLVIDAS: (orientações, instruções, recomendações sobre atividades relacionadas ao desenvolvimento do estágio supervisionado)

Concordam e Assinam:

Acadêmico(a): Orientador(a):

Coordenador de (Atividade)

Figura 9 – Relatório de acompanhamento a ser gerado pelo sistema

O diagrama de casos de uso apresenta uma linguagem simples e de fácil entendimento, pois trata da descrição das funcionalidades do sistema, auxilia no levantamento e na análise dos requisitos, define as necessidades do usuário e identifica de forma geral o comportamento do sistema. Esse diagrama procura identificar os seguintes componentes: os atores, que são os usuários externos, pessoas ou dispositivos que interagem com o sistema, bem como as opções que o sistema disponibilizará aos atores, conhecidas neste diagrama como casos de uso.

Na Figura 10 são exibidos os atores principais e vinculados a eles os principais casos de uso que irão executar.

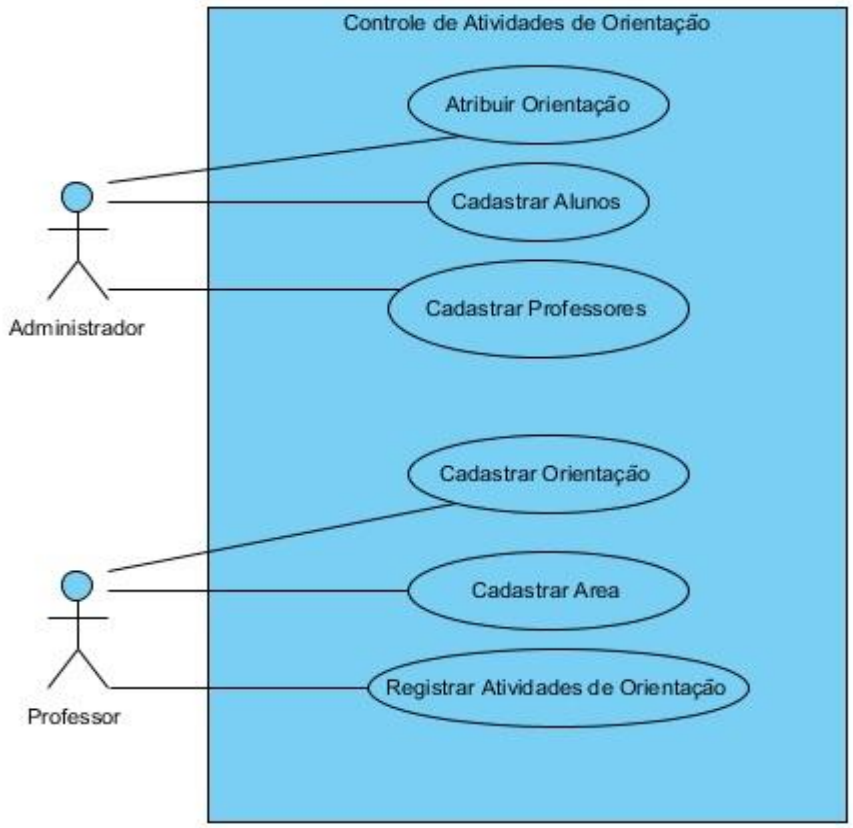

Figura 10 – Diagrama de casos de uso

O diagrama de classes define a estrutura das classes utilizadas pelo sistema, determinando os atributos que fazem parte de cada classe, além de estabelecer como as classes se relacionam e trocam informações entre si. Na Figura 11 é exibido um diagrama de classes, com a representação dos componentes e sua estrutura: classes nomeadas, com atributos e multiplicidade entre si.

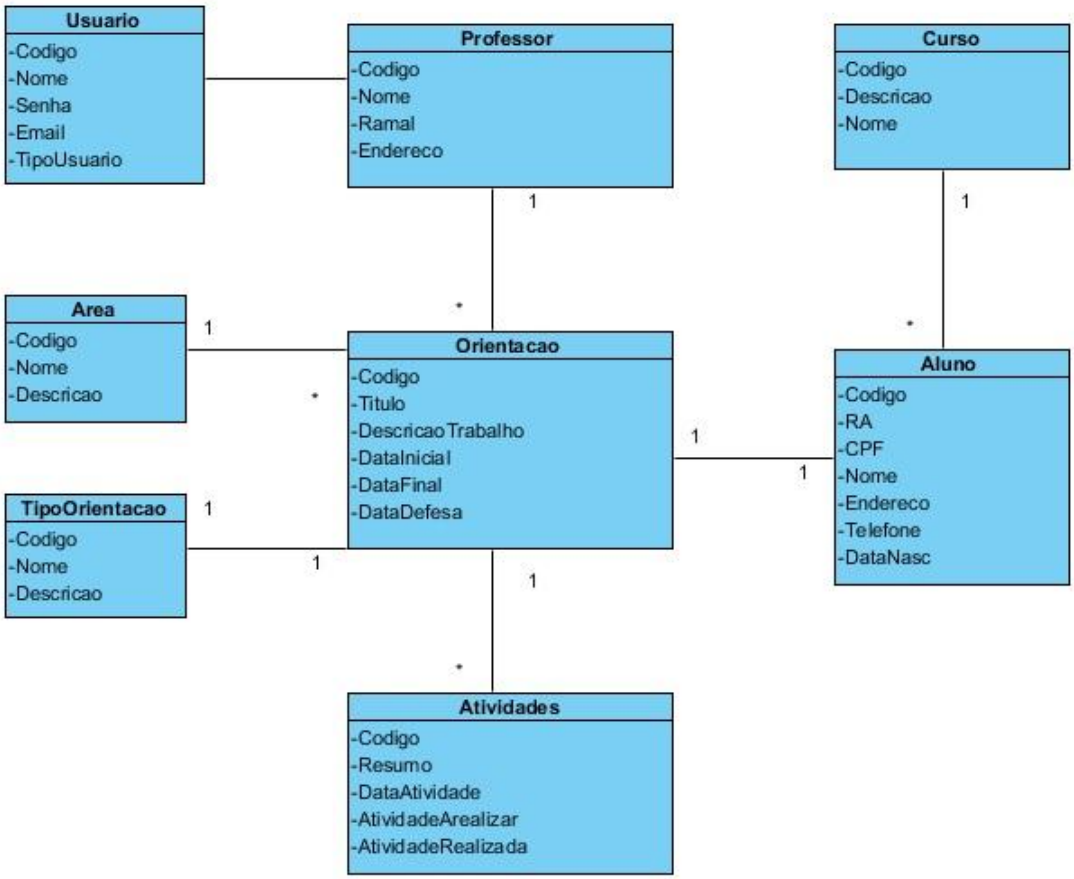

Figura 11 – Diagrama de classes

Com intuito de otimizar o desenvolvimento da interface do sistema, foram desenvolvidos protótipos das principais telas. A Figura 12 apresenta o protótipo da tela de cadastros de orientações.

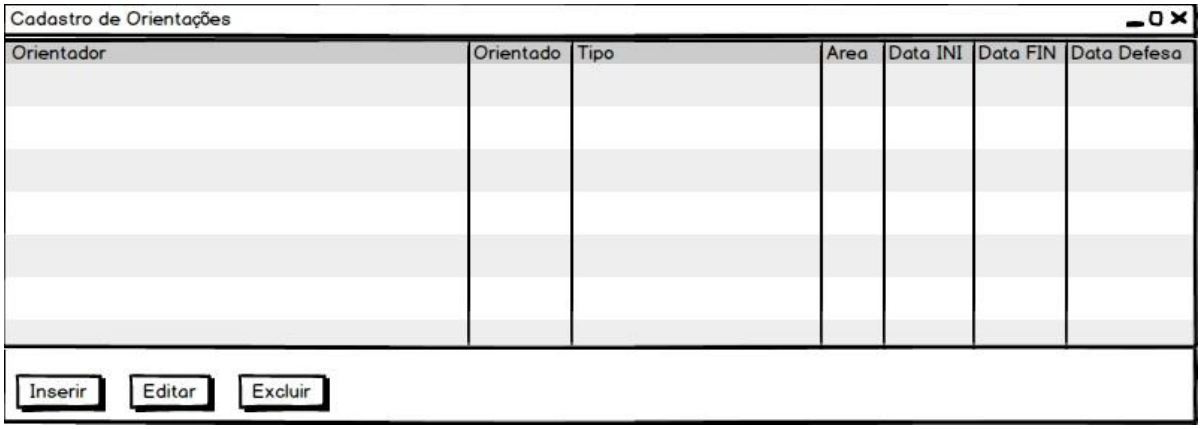

Figura 12 – Protótipo da tela de cadastro de orientações

A Figura 13 apresenta o protótipo da tela que deverá ser exibida quando o usuário selecionar a opção Inserir da Figura 12.

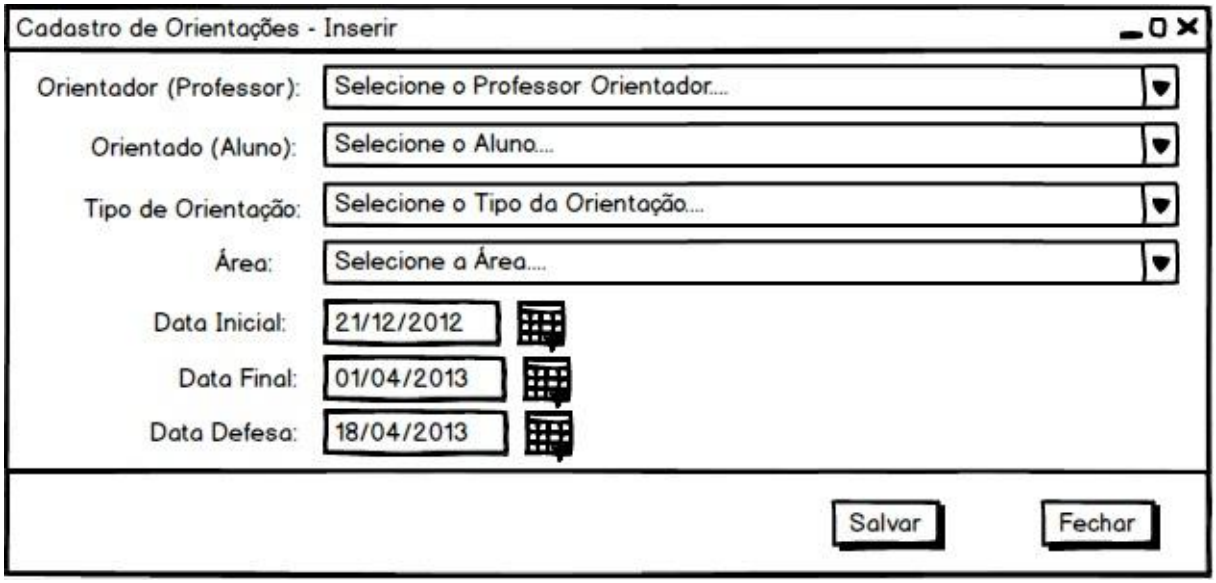

Figura 13 – Protótipo da tela de inserção de um cadastro de orientação

A Figura 14 apresenta o protótipo da tela de registro de acompanhamento das atividades de orientação realizadas pelo professor orientador.

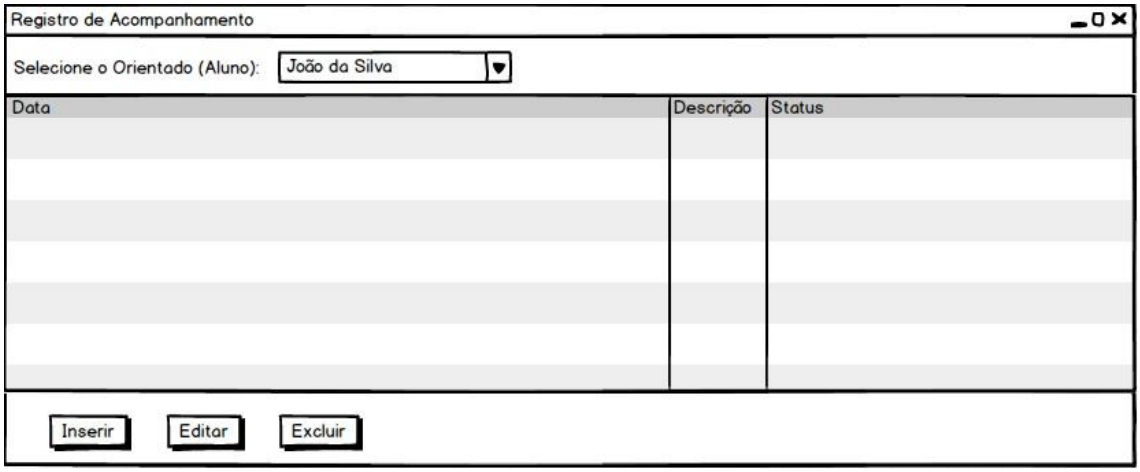

Figura 14 – Protótipo da tela de registro de acompanhamento

Na Figura 15 está o protótipo da tela que deverá ser exibida quando o usuário selecionar a opção Inserir da Figura 14.

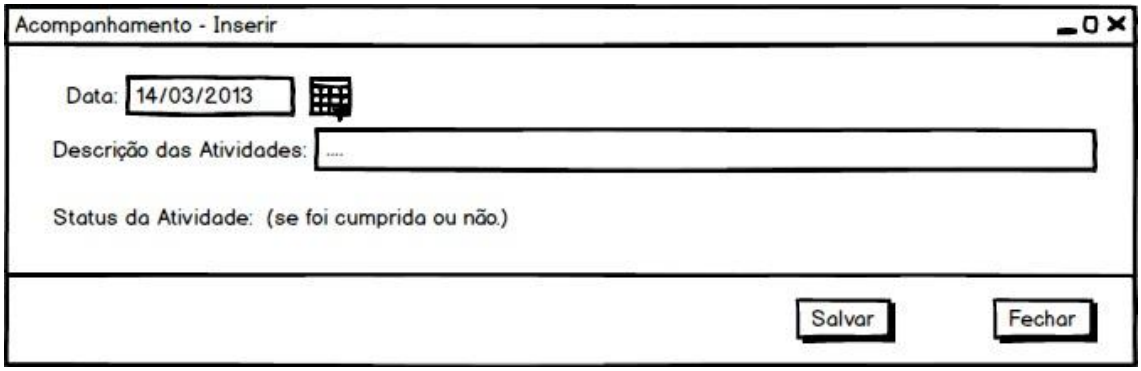

Figura 15 – Protótipo da tela de Inserção de registro de acompanhamento

4.3 DESCRIÇÃO DO SISTEMA

O sistema é iniciado pela tela de *Login* onde devem ser informados o usuário (*email*) e a senha para acesso ao sistema. A Figura 16 exibe a tela inicial de acesso.

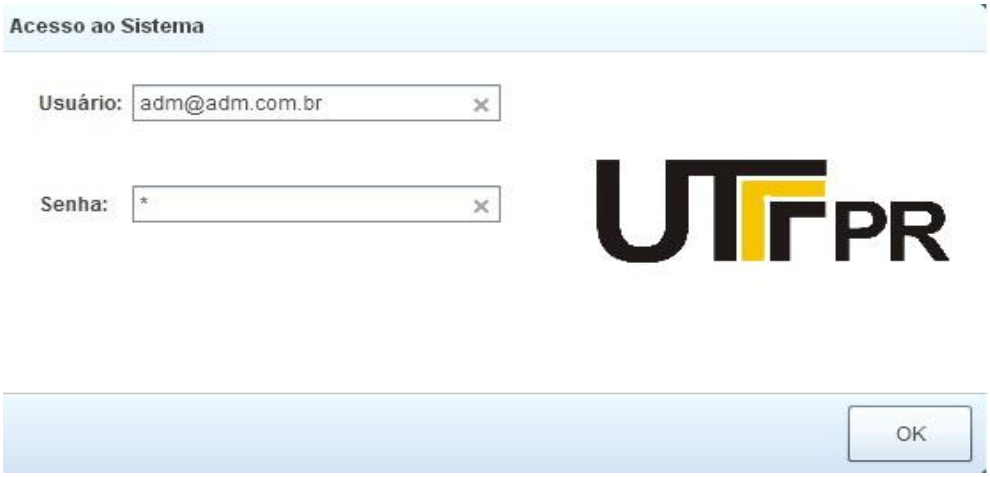

Figura 16 – Tela de acesso ao sistema

Após realizar o acesso ao sistema, será exibida a tela principal visualizada na Figura 17 com os menus de utilização (Cadastros, Orientação, Relatórios) e no topo da tela estão as informações sobre o acesso como o nome da aplicação, código e descrição do usuário conectado e a data da conexão. Pelo menu Cadastros é possível realizar o registro de todas as principais informações que serão utilizadas para a orientação realizada pelo professor a um acadêmico. No menu Orientação é realizado o registro da orientação e pode ser feito o acompanhamento da mesma e no menu Relatórios é exibida opção para geração da Ficha de Acompanhamento.

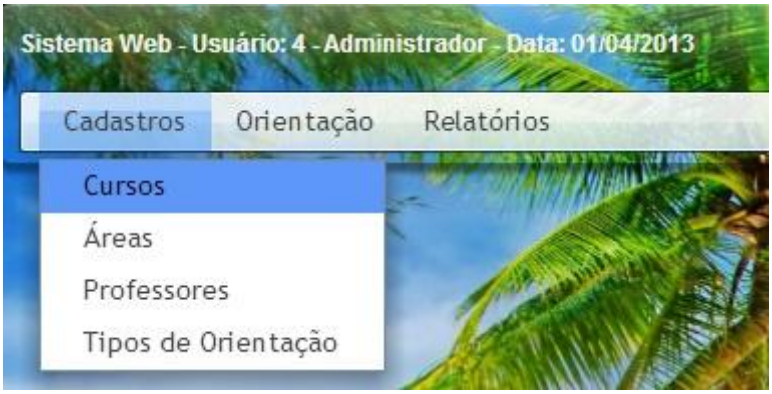

Figura 17 – Tela de menus do sistema

Para iniciar os trabalhos com o sistema é necessário realizar os principais cadastros por meio do menu Cadastros. Nesse menu é possível inserir Cursos, Áreas, Professores e Tipos de Orientação que serão necessários para compor o cadastro da orientação a ser realizada. A tela do cadastro de curso é iniciada exibindo todos os cursos já cadastrados (Figura 18), sendo possível realizar pesquisa diretamente pelo *grid* e por meio do botão Novo é possível inserir um novo registro que nessa tela é um cadastro de curso.

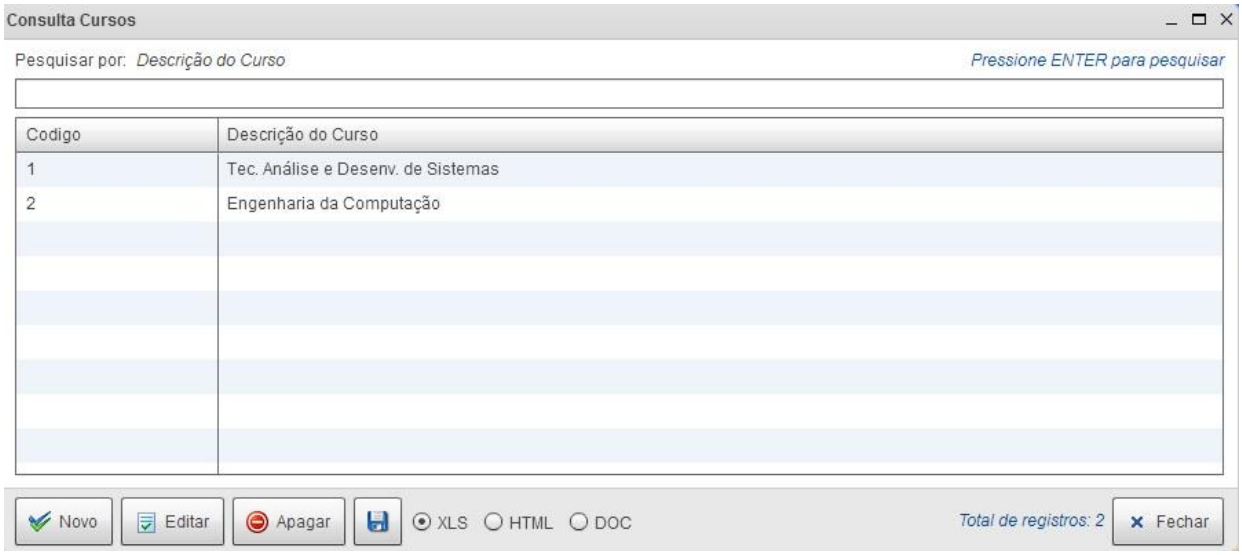

Figura 18 – Tela inicial do cadastro de cursos

Na Figura 19 é exibida a tela de Cadastro de Áreas, que são as áreas de atuação dos trabalhos a serem desenvolvidos e acompanhados.

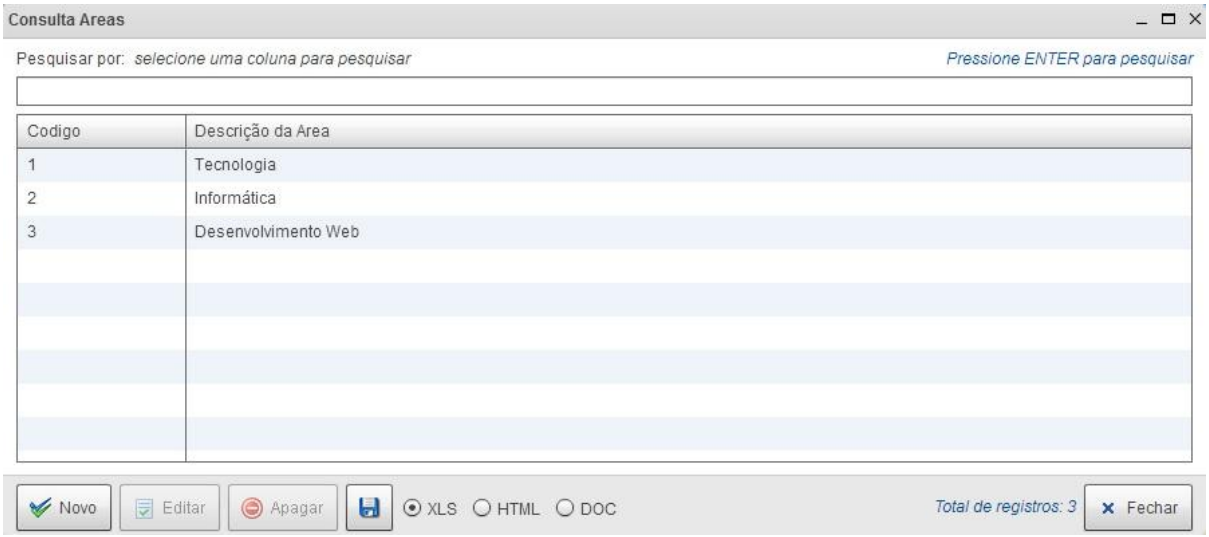

Figura 19 – Tela de cadastro de áreas

A Figura 20 apresenta o formulário pelo qual é possível cadastrar os professores que serão os orientadores dos trabalhos.

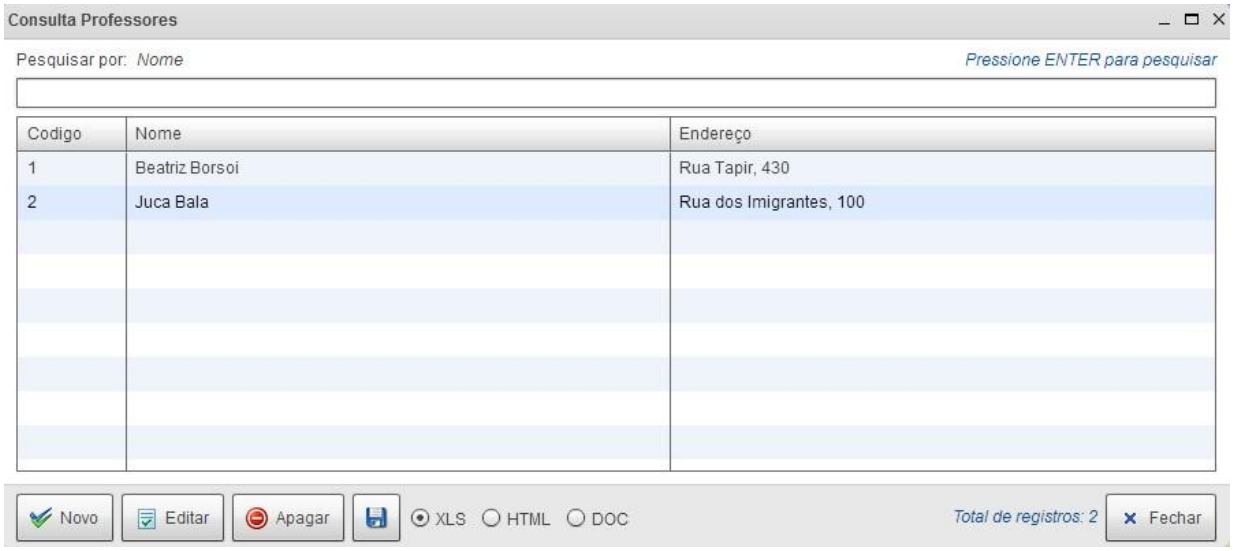

Figura 20 – Tela de cadastro de professores

Por fim, devem ser cadastrados os tipos de orientação que se referem ao tipo de trabalho acadêmico que será desenvolvido pelo aluno e orientado pelo professor. Na Figura 21 está a tela de cadastro de Tipos de Orientação.

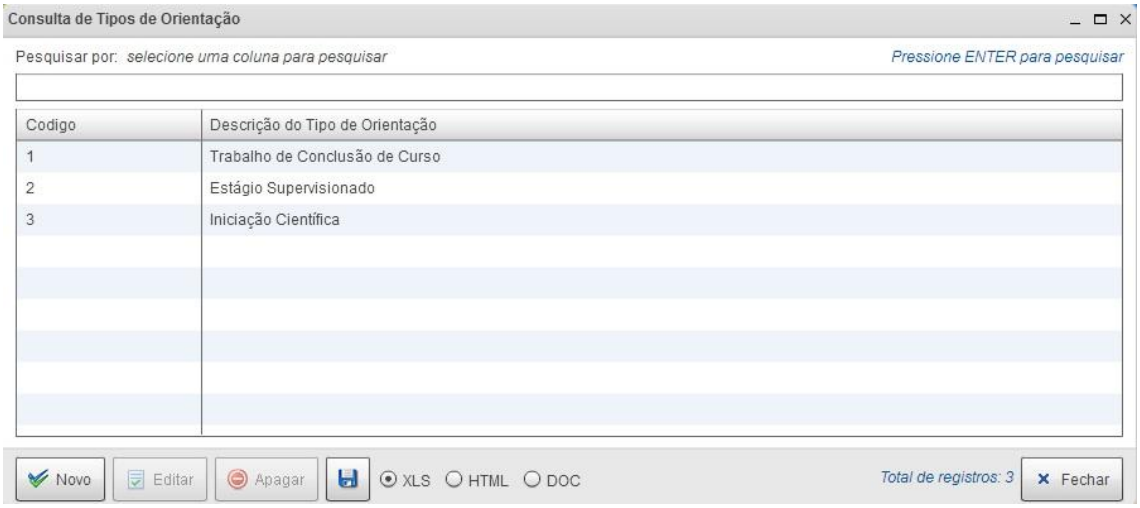

Figura 21 – Tela de cadastro de tipo de orientações

Após realizar todos os cadastros básicos pode ser criado o cadastro de orientação. Nesse cadastro será definido o professor orientador, o aluno a ser orientado, o tipo de orientação e demais informações necessárias. Essa opção pode ser acessada por meio do menu Orientações selecionando Cadastro de Orientações conforme a Figura 22.

Figura 22 – Tela do Menu de Acesso ao Cadastro de Orientações

Na tela de Cadastros de Orientações devem ser preenchidas todas as informações por meio dos campos *ComboBox* que exibem as informações cadastradas anteriormente. Essa tela, apresentada na Figura 23, é responsável por definir o professor orientador e qual aluno será seu orientado, bem como por cadastrar informações a respeito do tipo da orientação, da área profissional que o trabalho poderá influenciar e também as datas de início, fim e defesa/apresentação do projeto.

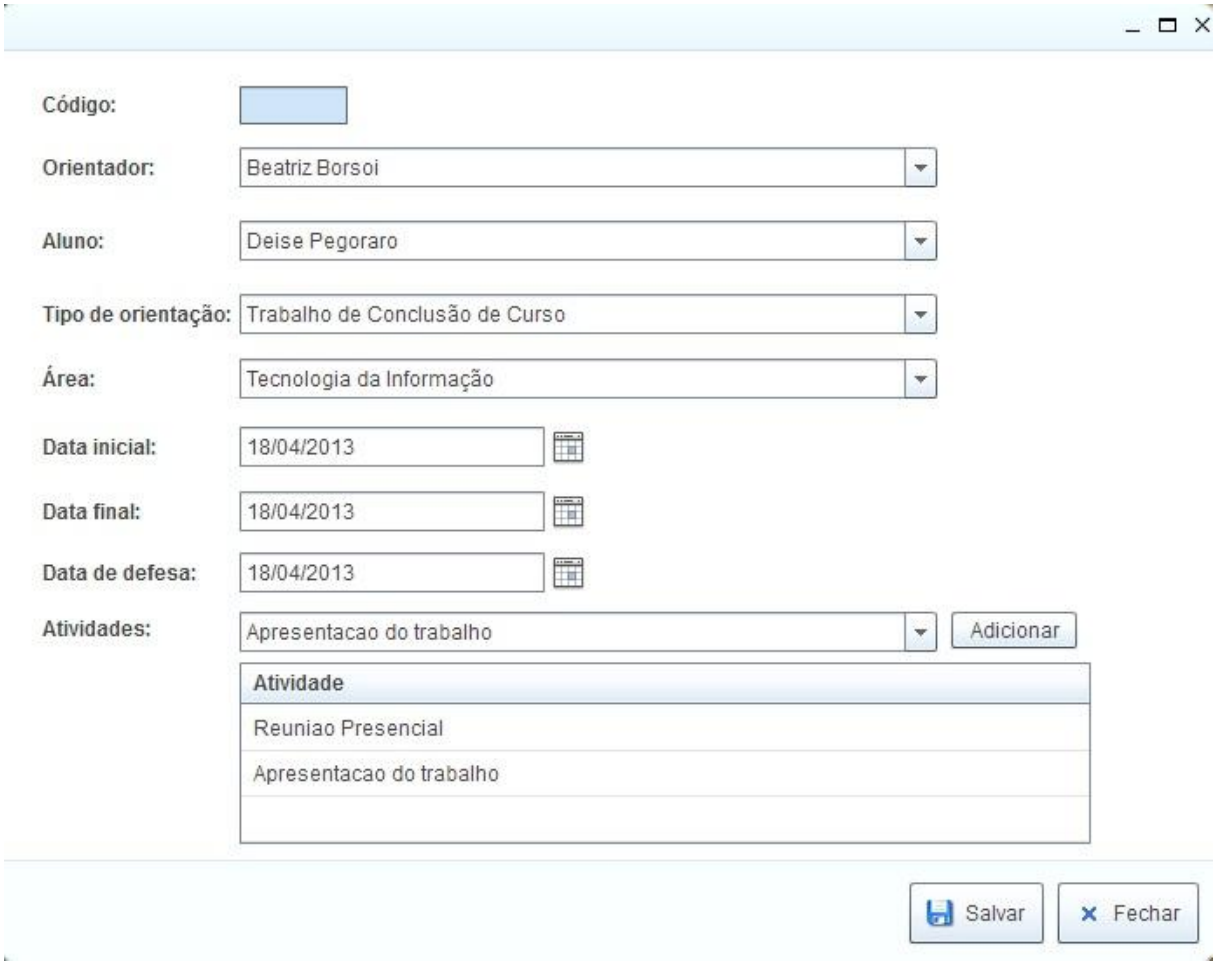

Figura 23 – Tela de cadastro de orientações

A tela de Registro de Acompanhamento poderá ser acessada por meio do menu Orientação selecionando a opção Registro de Acompanhamento (Figura 24).

Figura 24 – Tela do menu de acesso ao registro de acompanhamento

Na tela de registro de Acompanhamento é possível registrar todas as atividades de orientação realizadas pelo professor. Deve ser informada a data do registro, a descrição da atividade e informar se a mesma foi concluída ou se está pendente. O formulário para esse registro é apresentado na Figura 25.

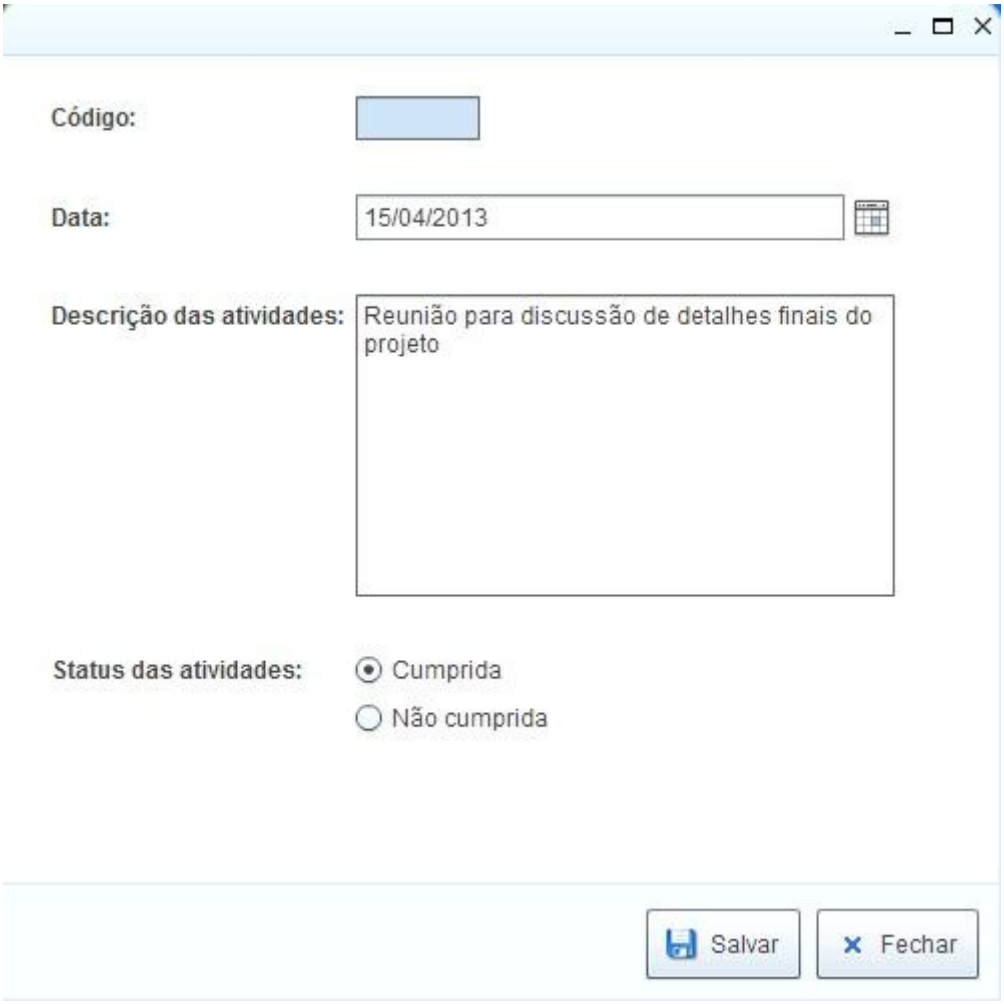

Figura 25 – Tela de registro de acompanhamento

Com todos os cadastros realizados é possível gerar a ficha de acompanhamento por meio do menu Relatórios, conforme a Figura 26.

Figura 26 – Tela do menu de acesso a ficha de acompanhamento

A tela para geração do relatório será apresentada logo em seguida, podendo selecionar qual orientação deseja visualizar por meio de um botão Imprimir, conforme a Figura 27.

Figura 27 – Tela de visualização das fichas de acompanhamento

O botão "Gerar Relatório" da Figura 27 gera a ficha de acompanhamento apresentada na Figura 28.

Figura 28 – Ficha de acompanhamento gerada pelo sistema

4.4 IMPLEMENTAÇÃO DO SISTEMA

Nesta seção serão apresentadas algumas listagens de códigos desenvolvidos para implementar o sistema.

Utilizando a tecnologia Flex foi criada uma classe genérica que possui um método que fará a comunicação com os Services do PHP. A função Service recebe os parâmetros e comunica com a classe PHP, chamando o método desta classe e enviando os argumentos previamente recebidos. A Listagem 2 apresenta o código da classe Service.

```
    public class Service extends RemoteObject
    {
        public function Service(classePHP:String, method:String,
onResult:Function, args:Object=null):void
        {
            super("RemotingMessage");

            this.source = classePHP;
            this.showBusyCursor = true;

            this.addEventListener(ResultEvent.RESULT, onResult);
            this.addEventListener(FaultEvent.FAULT, onFault);

                  this.getOperation(method).send(args);
        }
```

Listagem 2 – Classe Service

Em cada classe controladora dos formulários, são sobrescritos os métodos editar, salvar e excluir que chamam suas funções específicas em um Service no PHP. Em cada chamada é atribuída uma função de retorno que trata o resultado de acordo com a função que foi solicitada. Na Listagem 3 é exibida a função Editar (onEdit e onEditReturn).

```
override public function onEdit(event:ProcuraFormMDIEvent):void
{
      super.onEdit(event);
      service =  new Service("AreasService", "edit", onEditReturn,
event.params.ID_AREA);
}

public function onEditReturn(event : ResultEvent): void
{
      this.model.setObjFromTable(event.result);
      view.txtCodigo.text = model.ID_AREA.toString();
      view.txtDescricao.text = model.DESCRICAO;
}
```

Listagem 3 – Função onEdit e ontEditReturn

No método construtor de cada Service é instanciada uma classe ServiceDB (Listagem 4).

```
class AreasService {
```

```
	var $serviceDB;

		public function AreasService()
	{
		$this->serviceDB = new ServiceDB();
	}
```

Listagem 4 – Classe ServiceDB

A classe ServiceDB, por sua vez, possui funções especificas para buscar um objeto (fetchObject), buscar um *array* de objetos (fetchArray) e outra para executar um comando SQL (execQuery). Essa classe é apresentada na Listagem 5.

```
class ServiceDB {

	public function fetchObject($sql) {
		$retorno = false;
		Service::connect();
		$result = mysql_query($sql);
		while( $obj = mysql_fetch_object($result) )
		{
			$retorno = $obj;
		}
		Service::disconnect();
		return $retorno;
	}

	public function fetchArray($sql){
		$retorno = array();
		Service::connect();
		$result = mysql_query($sql);
		while( $obj = mysql_fetch_object($result) )
		{
			$retorno[] = $obj;
		}

		Service::disconnect();
		return $retorno;
	}

	public function execQuery( $sql ) {
		$retorno = false;
		Service::connect();
		$retorno = mysql_query( $sql );

		Service::disconnect();
		return $retorno;
}
```

Listagem 5 – Classe ServiceDB

Em cada uma dessas funções, antes de executar as instruções, é chamado um método de conexão da classe Service do PHP conforme apresenta a Listagem 6.

```
public static function connect()
	{
```

```
            Service::$dbhandle = mysql_connect(Service::$myServer,
Service::$myUser, Service::$myPass)
              or die("Couldn't connect to SQL Server on
Service::$myServer");

            $selected = mysql_select_db(Service::$myDB, Service::$dbhandle)
              or die("Couldn't open database Service::$myDB");
      }
```

Listagem 6 – Função de conexão

E logo após as instruções de cada função do ServiceDB é chamado um método que encerra a conexão. Esse método é apresentado na Listagem 7.

```
      public static function disconnect()
      {

            mysql_close(Service::$dbhandle);
      }
```

Listagem 7 – Função para desconexão

Cada Service chamado pelo Flex possui as funções para editar, salvar, excluir e também um método que retorna todos os registros de uma consulta ao banco de dados. Por exemplo, na Listagem 8 está o trecho do código php da função responsável por retornar os registros da consulta referente ao ServiceAreas.

```
      public function all()
      {
            return $this->serviceDB->fetchArray(" SELECT * FROM AREAS ");
      }
```

Listagem 8 – Classe Service

Na Listagem 9 está o trecho do código PHP da função Editar referente ao ServiceAreas.

```
public function edit( $id ) {
            $sql = " SELECT * FROM AREAS WHERE ID_AREA = $id ";
            return $this->serviceDB->fetchObject($sql);
      }
```

Listagem 9 – Função para editar do ServiceAreas

Na Listagem 10 está o trecho do código PHP da função Salvar referente ao ServiceAreas.

```
public function save( $model ) {
            $id = $model->ID_AREA;
            $descricao = $model->DESCRICAO;

            if ( $id == 0 ) {
                  $sql = " INSERT INTO AREAS (DESCRICAO) VALUES
('$descricao') ";
            } else {
```

```
                  $sql = " UPDATE AREAS SET DESCRICAO = '$descricao' where
ID_AREA = $id ";
            }

            return $this->serviceDB->execQuery( $sql );
      }
```

Listagem 10 – Função para salvar do ServiceAreas

O trecho do código PHP da função Deletar referente ao ServiceAreas é apresentado na Listagem 11.

```
public function delete( $id ) {
            if ( $id > 0 )
            {
                  $sql = " DELETE FROM AREAS where ID_AREA = $id ";
                  return $this->serviceDB->execQuery( $sql );
            }
      }
```

Listagem 11 – Função para excluir do ServiceAreas

Para exemplificar a parte de interface criada pelo Flex, na Listagem 12 está o XML responsável pela tela de *login* de acesso ao sistema.

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Canvas xmlns:fx="http://ns.adobe.com/mxml/2009"
               xmlns:s="library://ns.adobe.com/flex/spark"
               xmlns:mx="library://ns.adobe.com/flex/mx"
               xmlns:controls="com.flexpernambuco.controls.*"
               width="100%" height="100%"
xmlns:controller="br.com.tcc.controller.*"
               creationComplete="controladora.inicializar()">

      <fx:Declarations>
            <controller:LoginController id="controladora" view="{this}" />
      </fx:Declarations>

      <s:Panel width="564" height="296" cornerRadius="10"
horizontalCenter="-2"
                  title="Acesso ao Sistema" verticalCenter="11">


            <s:HGroup width="553" height="203">
                  <s:Form width="334" height="100%">
                        <s:FormItem width="100%" height="100%"
label="Usuário:">
                              <controls:MasterTextInput id="txtUsuario"
width="100%"

required="true"

maxChars="50"

tabIndex="1"

nextFocusOnEnter="true"

capsType="lowercase"
```

```
toolTip="Email do usuário cadastrado"

text="1"/>

                        </s:FormItem>
                        <s:FormItem width="100%" height="100%"
label="Senha:">
                              <controls:MasterTextInput id="txtSenha"
width="100%"

required="true"

maxChars="25"

tabIndex="2"

nextFocusOnEnter="true"

displayAsPassword="true"

capsType="lowercase"

toolTip="Senha de acesso do usuário"

text="1"/>
                        </s:FormItem>
                        <s:FormItem width="100%" height="100%">
                              <s:Label width="100%" height="100%"
color="#CE5555" fontWeight="bold"

text="{controladora.usuarioInvalido}" textAlign="center"
                                          verticalAlign="middle" />
                        </s:FormItem>
                  </s:Form>
                  <s:Image width="200" height="200"
source="br/com/tcc/assets/img/utfpr.png"/>
            </s:HGroup>
            <s:controlBarContent>
                  <s:Spacer width="100%" height="100%"/>
                  <s:Button id="btnOK" height="35" label="OK"/>
            </s:controlBarContent>
      </s:Panel>
</mx:Canvas>
```

Listagem 12 – XML da tela de login ao sistema

5 CONCLUSÃO

Com o término do desenvolvimento deste trabalho pode-se concluir que o objetivo principal foi atingido, por meio dos requisitos, das funcionalidades e dos conceitos elencados durante a análise e projeto, bem como, pela modelagem por meio da qual foi possível visualizar possíveis falhas antes do desenvolvimento do sistema para registro de orientação de atividades acadêmicas.

Foi possível perceber que o ambiente *web* oferece muitas possibilidades de desenvolvimento tanto para *sites* simples como para sistemas complexos e ricos em detalhes, seja em termos de interface, de funcionalidades ou de conteúdo. As ferramentas utilizadas tiveram um papel importante, pois possibilitaram o desenvolvimento de um sistema simples, porém funcional, que apresenta uma interface clara e objetiva. Isso no sentido que as operações a serem realizadas são muito intuitivas e seguem um mesmo padrão.

As dificuldades encontradas durante o desenvolvimento estiveram relacionadas basicamente a conflito de porta de acesso do servidor *web* local com o aplicativo Skype. Para resolver esse problema foi necessário desmarcar uma configuração deste aplicativo que utilizava a mesma porta de acesso. Essa solução foi encontrada através de pesquisas realizadas na Internet e principalmente por meio da colaboração da professora orientadora e de outros professores da Universidade, com destaque para a professora Andreia Scariot Beulke. Esses instruíram da melhor maneira para a resolução do problema possibilitando a continuidade do desenvolvimento do projeto.

Por fim, é possível indicar que o desenvolvimento desse software solucionará o problema proposto, agilizando as tarefas dos orientadores no momento de documentar as atividades de orientação.

REFERÊNCIAS

ALVAREZ, Miguel Angel. **Introdução à programação PHP**. 2004. Disponível em: <http://www.criarweb.com/artigos/70.php>. Acesso em: 10 fev. 2013.

AMFPHP. **AmfPHP**. Disponível em: <http://www.silexlabs.org/amfphp/>. Acesso em: 5 mar. 2013.

APTANA. **Aptana Studio 3**. Disponível em: <http://www.aptana.com/products/studio3>. Acesso em: 12 mar. 2013.

CASTELLS, Manuel. **A Galáxia Internet: reflexões sobre internet, negócios e sociedade.** Lisboa: Fundação Calouste, 2004. Disponível em: <http://www.edrev.info/reviews/revp49.pdf>. Acesso em: 18 mar. 2013.

BURBECK, Steve. **Applications programming in Smalltalk-80: how to use Model-View-Controller (MVC)**. 1992. Disponível em: <http://st-www.cs.illinois.edu/users/smarch/st-docs/mvc.html>. Acesso em: 19 mar. 2013.

CUI, Wei; HUANG, Lin; LIANG, LiJing; LI, Jing .**The Research of PHP Development Framework Based on MVC Pattern.** Fourth International Conference on Computer Sciences and Convergence Information Technology, (ICCIT '09), 2009, p. 947-949.

DOBOS, Hanna. **Separable user interfaces and interaction controls**. Master's thesis. Departament of Mathematical Information Technology. University of Jyväskylä. Disponível em: < http://www.mit.jyu.fi/opetus/opinnayte/gradu/separableUIs/Hanna_Dobos.pdf>. Acesso em: 19 mar. 2013.

GUERINI, Guillermo. **Balsamiq Mockups – Uma ferramenta simples e poderosa**. Disponível em: <http://gguerini.tumblr.com/post/27904760783/balsamiq-mockups#_=_ http://www.balsamiq.com/products/mockups>. Acessado em: 03 mar. 2013.

LARMAN, Craig. **Utilizando UML e padrões: uma introdução à análise e ao projeto orientados a objetos e ao desenvolvimento iterativo**. 3. ed. Porto Alegre: Bookman, 2007

MCHEICK, Hamid, QI, Yan. D**ependency of components in MVC distributed architecture.** Canadian Conference on Electrical and Computer Engineering (CCECE 2011), 2011, p. 691-694.

MICROSFOT, **MSDN Model-View-Controller**, versão 1.01. Disponível em: <http://msdn.microsoft.com/en-us/library/ff649643.aspx>. Acesso em: 10 mar. 2013.

MOREIRA, Alessandro. **Tutorial iniciando em Adobe Flex 3**. Disponível em: <http://alessandromoreira.blogspot.com.br/2009/02/tutorial-iniciando-em-adobe-flex-3.html>. Acesso em: 5 mar. 2013.

MUKHTAR, Shams. **Applying robustness analysis on the Model–View–Controller (MVC) architecture in ASP.NET framework, using UML**. Disponível em: <http://www.codeproject.com/aspnet/ModelViewController.asp>. Acesso em: 19 de mar. 2013

MySQL. **MySQL Editions**. Disponível em: <http://www.mysql.com/products/>. Acesso em: 10 mar. 2013.

MYSQL. **MySQL.** Disponível em: <http://www.mysql.com>. Acesso em: 29 jan. 2013.

PHP. **PHP**. Disponível em: <http://www.php.net/manual/pt_BR/intro-whatis.php>. Acesso em: 15 fev. 2013.

PHPADMIN. **Bringing MySQL to the web**. Disponível em: <http://www.phpmyadmin.net/home_page/index.php>. Acesso em 20 mar. 2013.

PISA, Pedro. **O que é e como usar o MySQL?** Disponível em: <http://www.techtudo.com.br/artigos/noticia/2012/04/o-que-e-e-como-usar-o-mysql.html>. Acesso em: 10 mar. 2013.

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

SELFA, Diana M., CARRILLO, Maya, BOONE, Ma. del Rocío. **A database and web application based on MVC architecture**. 16th International Conference on Electronics, Communications and Computers (CONIELECOMP '06), p. 48-53, 2006.

TAO, *Yonglei* **.Component- vs. application-level MxVC architecture. 3Znd** ASEEiTEEE Frontiers in Education Conference, 2002, T2G7-T2G10.

WEI, Song. **Approach to web application program based on MVC and J2EE**, Hebei Journal of Industrial Science&Technology,July 2005,22(4),pp. 189-191

VISUAL PARADIGM. **Visual Paradigm**. Disponível em: <http://www.visual-paradigm.com/>. Acesso em: 30 nov. 2012.

WANG, Guanhua. **Application of lightweight MVC-like structure in PHP**. 2011 International Conference on Business Management and Electronic Information (BMEI), p. 74-77, 2011.