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

HELDER FELIPE FIGURA DA SILVA

SISTEMA WEB PARA CONTROLE DE FLUXO DE TRABALHO EM EMPRESAS DESENVOLVEDORAS DE SOFTWARE

TRABALHO DE CONCLUSÃO DE CURSO

PATO BRANCO
2011

HELDER FELIPE FIGURA DA SILVA

SISTEMA WEB PARA CONTROLE DE FLUXO DE TRABALHO EM EMPRESAS DESENVOLVEDORAS DE SOFTWARE

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.

Orientador: Prof. Omero Francisco Bertol

PATO BRANCO
2011

ATA Nº: **192**

DEFESA PÚBLICA DO TRABALHO DE DIPLOMAÇÃO DO ALUNO **HELDER FELIPE FIGURA DA SILVA.**

Às 10:30 hrs do dia 9 de dezembro de 2011, Bloco S da UTFPR, Campus Pato Branco, reuniu-se a banca avaliadora composta pelos professores Omero Francisco Bertol (Orientador), Géri Natalino Dutra (Convidado) e Maico Fernando Wilges Carneiro (Convidado), para avaliar o Trabalho de Diplomação do aluno Helder Felipe Figura da Silva, matrícula 910015, sob o título **Sistema Web para Controle de Fluxo de Trabalho em Empresas Desenvolvedoras de Software**; como requisito final para a conclusão da disciplina Trabalho de Diplomação do Curso Superior de Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, Coordenação de Informática. 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 11:10 hrs foi encerrada a sessão.

Prof. Omero Francisco Bertol, M.Sc.
Orientador

Prof. Géri Natalino Dutra, M.Sc.
Convidado

Prof. Maico Fernando Wilges Carneiro, Esp.
Convidado

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

Prof. Edilson Pontarolo, Dr.
Coordenador do Curso

RESUMO

FIGURA DA SILVA, Helder Felipe. **Sistema Web para Controle de Fluxo de Trabalho em Empresas Desenvolvedoras de Software**. 2011. 45 f. Monografia de Trabalho de Conclusão de Curso - Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, Universidade Tecnológica Federal do Paraná. Pato Branco, 2011.

No atual contexto globalizado a gestão do conhecimento é um ponto fundamental para o crescimento organizacional. O aumento da concorrência entre as empresas exige cada vez mais velocidade, qualidade e segurança nos processos realizados. Surgindo a necessidade da utilização de tecnologias que venham a auxiliar na gestão destes processos. A tecnologia de automação do fluxo de trabalho (*workflow*) permite que o fluxo de papel de uma empresa seja substituído por formulários eletrônicos que percorrem a empresa através de uma estrutura de comunicação pré-definida. O sistema apresentado visa gerenciar o fluxo de tarefas e chamados em empresas desenvolvedoras de software.

Palavras-chave: Fluxo de trabalho. Workflow. Gestão de processos. Tarefas. Chamados.

ABSTRACT

FIGURA DA SILVA, Helder Felipe. **Web System for Control Workflow in Software Development Companies**. 2011. 45 f. Monograph Working End of Course - Degree in Technology Analysis and Systems Development, Federal Technological University of Parana. Pato Branco, 2011.

In the current globalized context the knowledge management is fundamental to the organizational growth. The competition increase among companies requires more speed, quality and security in the process. Creating this way the necessity of using technologies to help the process management. The workflow automation technology allows the company paper flow to be replaced by electronic forms that flow through the company in a default defined communication structure. This system aims to manage the task and calls flows in software development companies.

Keywords: Workflow. Process Management. Task. Calls.

LISTA DE ABREVIATURAS E SIGLAS

AJAX	*Asynchronous JavaScript and XML*
API	*Applications Programming Interface*
CLA	*Contributor License Agreement*
CSS	*Cascade Style Sheets*
DER	Diagrama Entidade-Relacionamento
DOM	*Document Object Model*
GPL	*General Public License*
HTML	*HyperText Markup Language*
HTTP	*Hypertext Transfer Protocol*
JSON	*JavaScript Object Notation*
MVC	*Model View Control*
PDO	*PHP Data Objects*
PHP	*Hypertext Preprocessor*
RAD	*Rapid Application Development*
RSS	*Really Simple Syndication*
SQL	*Structured Query Language* (ou linguagem de consulta estruturada)
WfMS	*Workflow Management System*
XML	*eXtensible Markup Language*
ZF	*Zend Framework*

LISTA DE FIGURAS

ÍNDICE DE QUADROS

ÍNDICE DE TABELAS

SUMÁRIO

1. INTRODUÇÃO

Em sistemas baseados em processos, onde os processos são o conjunto de uma ou mais atividades que buscam atingir objetivos dentro de uma estrutura organizacional, podem ocorrer problemas com relação à má distribuição de trabalho. As áreas de estudos relacionadas a *workflow* objetivam solucionar tais problemas.

1.1 CONSIDERAÇÕES INICIAIS

Sistemas de w*orkflow* manipulam e monitoram a informação relativa ao fluxo de trabalho para gerenciar e controlar o trabalho mais eficientemente, minimizando o problema da coordenação do trabalho nos processos de negócios (NICOLAO, 1998). Dessa maneira estes sistemas vêm ganhando força no mercado.

Atualmente, os sistemas de *workflow* são cada vez mais necessários dentro de uma empresa, sua utilização é vista como vantajosa independente de qual seja seu ramo de atividade.

As empresas que utilizam visam um maior controle sobre seus processos internos, permitindo manipular uma maior quantidade de informações, de forma precisa e segura.

Os sistemas de w*orkflow* baseados na web ganharam em importância com o amadurecimento da Internet e recentemente com as aplicações web, podendo assim ser executadas através de um *browser* (navegador), aumentando e muito a disponibilidade de acesso. Estes sistemas possuem vantagem sobre os demais, como, por exemplo, menor custo, melhor aproveitamento do tempo, maior facilidade de utilização e principalmente, maior alcance (CRUZ, 2000).

Este trabalho procura apresentar um sistema *workflow* básico, desenvolvido em plataforma web, para gerenciamento de tarefas e chamadas em empresas desenvolvedoras de software.

1.2 OBJETIVOS

Este trabalho de conclusão de curso trata de um desenvolvimento web de software e a seguir terá seus objetivo geral e objetos específicos descritos.

1.2.1 Objetivo Geral

Desenvolver uma aplicação, em plataforma web, para gerenciamento de w*orkflow*.

1.2.2 Objetivos Específicos

Para atingir os objetivos pretendidos neste projeto de conclusão de curso as seguintes etapas foram estabelecidas:

- Realizar estudo sobre a importância do gerenciamento do fluxo de trabalho para obter produtividade e qualidade no desenvolvimento de software.
- Realizar o levantamento de requisitos de dados e funcionais.
- Projetar e implementar o modelo do banco de dados.
- Desenvolver o projeto de software para gerenciamento de w*orkflow* utilizando-se da arquitetura MVC com o framework Zend (PHP) e banco de dados MySql com as seguintes funcionalidades:
    - Gestão de entidades relacionadas à empresa;
    - Gestão de chamados;
    - Gestão de tarefas;
    - Gestão de metas;
    - Acompanhamento dos chamados e tarefas através de interações.

1.3 JUSTIFICATIVA

Partindo do caso de uma empresa desenvolvedora de software, com desejo de gerenciar melhor a produção de seus funcionários bem como possibilitar um controle de qualidade mais rigoroso, surgiu à necessidade de uma ferramenta que venha facilitar a gestão de fluxo de trabalho.

Esse sistema visa gerenciar os chamados e tarefas de forma modularizada, isto é, separando as requisições de atendimento da linha de produção. Desta maneira é possível filtrar o que realmente deverá ser desenvolvido bem como sua prioridade, prazo e custos.

Ao se estabelecer um procedimento claro a ser seguido, passos desnecessários são eliminados e cada membro da equipe fica plenamente consciente das suas responsabilidades dentro do projeto. Garantindo foco por parte dos desenvolvedores e visão ao gestor.

1.4 ESTRUTURA DO TRABALHO

Este texto está organizado em capítulos, dos quais este é o primeiro e apresenta a ideia do sistema, incluindo os objetivos e a justificativa. O Capítulo 2 apresenta o domínio da aplicação através de um referencial teórico sobre fluxo de trabalho (ou w*orkflow*). Já no Capítulo 3 está o método, que é a sequência geral de passos para o ciclo de vida do sistema e os materiais utilizados. O Capítulo 4 contém partes do sistema desenvolvido, visando exemplificar as suas funcionalidades. Por último, no Capítulo 5 está a conclusão e as considerações finais.

2. WORKFLOW

Este capítulo abordará as principais características de um w*orkflow*, na sua parte conceitual, servindo de base para o desenvolvimento do sistema em questão.

2.1 INTRODUÇÃO

Segundo a WfMC (*Workflow Management Coalition*), um processo é "um conjunto coordenado de atividades (sequenciais ou paralelas) que são interligadas com o objetivo de alcançar um meta comum" e uma atividade é conceituada como "uma descrição de um fragmento de trabalho que contribui para o cumprimento de um processo" (WFMC, 2011).

Workflow é definido pela WfMC como “a automação total ou parcial de um processo de negócio, durante a qual documentos, informações e tarefas são passadas entre os participantes do processo” (WFMC, 2011).

O *workflow* representa uma relação entre tecnologia e processos de negócio, onde cada um induz uma relação com o outro. É a sequência de passos necessários para que se possa alcançar a automação de processos utilizando-se de algumas regras.

A tecnologia do w*orkflow* também atua como canal de informação, uma vez que o *workflow* lida diretamente com pessoas e a sua interação nas organizações.

2.2 TIPOS DE WORKFLOW

Os processos de negócios, em cada organização, se mostram de maneira distinta, com características próprias, um ambiente propício à aplicação de um modelo de *workflow* que represente com realidade as informações da organização. Atividades distintas tais como: processamento de transações financeiras e fluxo de documentos da instituição, entre outras; podem ou não ser representadas por um mesmo modelo de *workflow*. Para que a aplicação do modelo utilizado para representar o fluxo de trabalho na organização seja facilitada é necessário que se defina qual o tipo de *workflow* será aplicado, a partir da identificação dos tipos de

sistemas de *workflow*, evitando assim que se escolha um modelo inadequado para o problema proposto (NICOLAO, 2011).

Os *workflows* podem ser categorizados de três maneiras: ad hoc, administrativo e produção.

2.2.1 Workflow Ad Hoc

Workflows do tipo Ad Hoc suportam definição rápida e execução de modelos de processos menos complexos envolvendo coordenação humana, elaboração ou co-decisão (NICOLAO, 1996).

Coordena atividades que não permitem prever as próximas que serão executadas, mas que apresentam um objetivo a ser alcançado. A estrutura desse tipo de sistema pode ser modificada em tempo de execução.

É típico desta classe de *workflow*, mostrado na **Figura 1**, o envolvimento de pequenos grupos de profissionais que tem a intenção de apoiar pequenas atividades que requerem uma solução rápida, como por exemplo, no desenvolvimento de software.

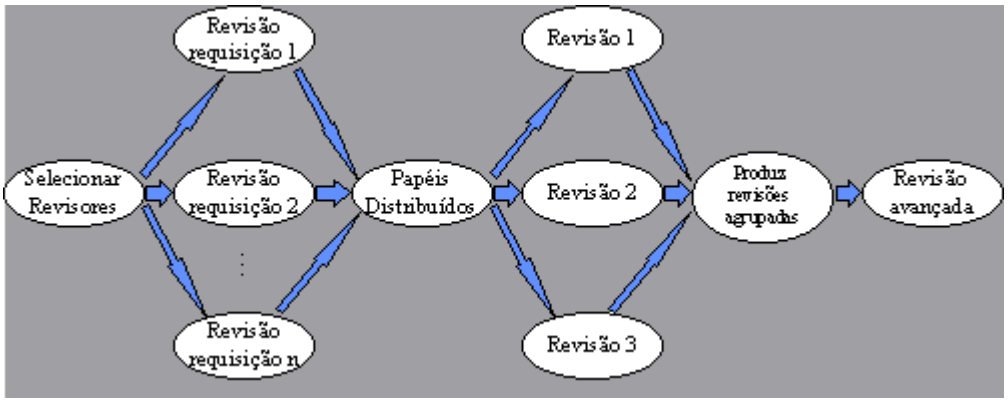

Figura 1- Workflow do Tipo Ad Hoc
Fonte: http://www.oocities.org/wallstreet/market/4702/textos/img/workfl5.gif.

2.2.2 Workflow Administrativo

O *workflow* Administrativo envolve atividades em estruturadas repetitivas e previsíveis, com regras simples de coordenação. Essa categoria não engloba

processos complexos de informação e não requer acesso a sistemas de informação múltiplos.

A sequência de passos em uma revisão de artigos, apresentada na **Figura 2**, seria um exemplo de *workflow* Administrativo. Onde os revisores não efetuam uma revisão conjunta, ocorre à produção de uma revisão individual. Estas revisões são consideradas pelo editor que toma a decisão final.

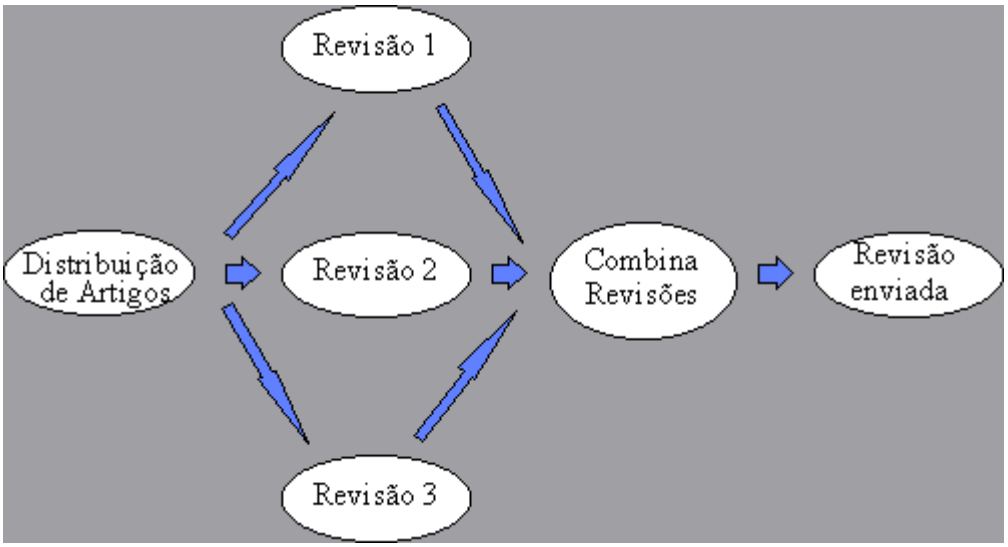

Figura 2- Workflow do Tipo Administrativo
Fonte: http://www.oocities.org/wallstreet/market/4702/textos/img/workfl9.gif

2.2.3 Workflow Produção

Workflows de produção envolvem atividades estruturadas que descrevem processos de informação complexos. Normalmente a sua execução requer um alto número de transações que acessam múltiplos sistemas de informação, e englobam processos de negócios repetitivos e previsíveis, como aplicação de empréstimo ou reivindicações de seguro (KROHA, 1997).

Essa categoria engloba processos em negócios repetitivos e previsíveis, como empréstimos. Diferente dos administrativos, os *workflows* de produção envolvem processamentos de informações complexas e acesso a múltiplos sistemas de informação.

A **Figura 3** apresenta um *workflow* de requisições de seguro, e representa a complexidade e o acesso a diversos sistemas de informação até que o objetivo final seja alcançado.

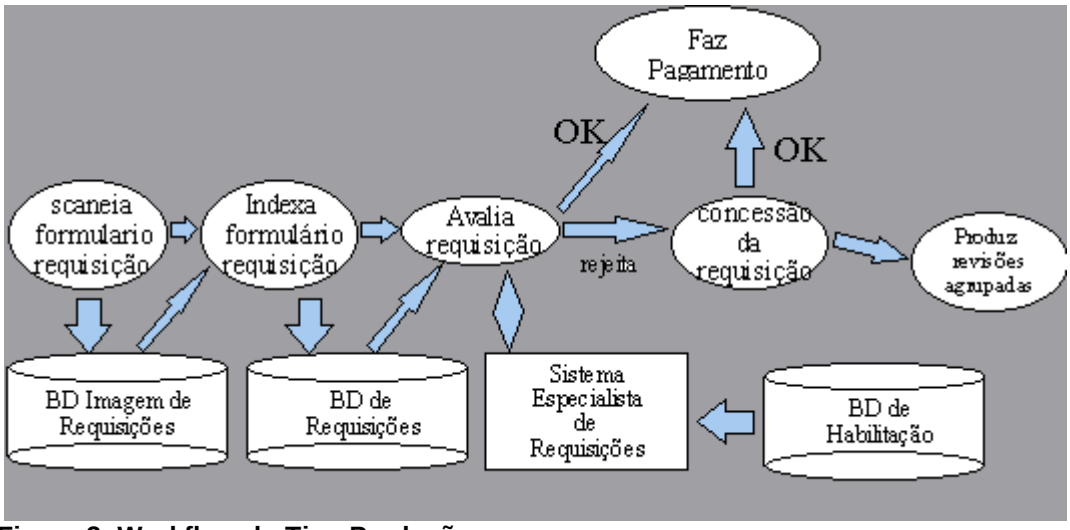

Figura 3- Workflow do Tipo Produção
Fonte: http://www.oocities.org/wallstreet/market/4702/textos/img/workfl11.gif

2.3 GERENCIAMENTO DE WORKFLOW

Gerenciamento de *Workflow* é um sistema de supervisão do processo de transmissão de informações, documentos e tarefas dentro de uma empresa.

O termo Gerenciamento de *Workflow* (*Workflow Management*) refere-se às ideias, métodos, técnicas e software usados para apoiar processos de negócios estruturados. O fluxo e a distribuição de trabalho devem ser controlados pelos Sistemas de Gerenciamento de *Workflow* (ou WfMS- *Workflow Management System*). Após a definição de processo, o WfMS verifica se as atividades estão sendo executadas na sequência correta e notifica o usuário sobre a execução das tarefas. Em geral, os WfMS predefinem a sequência em que as atividades serão executadas, determinam qual participante executará cada atividade e acompanham a situação de uma determinada instância de *workflow*. Outros termos para caracterizar sistemas de gerenciamento de *workflow* são: sistemas de operações de negócio, gerenciamento de *workflow*, gerenciamento de casos (*case manager*) e sistema de controle logístico (AALST & HEE, 2002).

Sistema de Gerenciamento de *Workflow* é um sistema que define, gerencia e executa completamente *workflows* através da execução de software cuja ordem de atividades é dirigida por uma representação da lógica do *workflow*. Uma vez que um processo é definido, um WfMS certifica-se de que as atividades ocorram numa

sequência própria e que os usuários sejam informados para que possam executar suas tarefas (WFMC, 2011).

Definidos os processos, um WfMS, garante que as atividades dos processos ocorram na sequência definida, fazendo com que cada usuário seja informado das atividades que deve realizar.

Armazenando-se as informações de cada tarefa, bem como seu histórico de mudanças será possível a geração de dados estratégicos. Os quais refletirão sobre a eficiência dos processos desempenhados, possibilitando assim melhor andamento das atividades.

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 no desenvolvimento do sistema.

3.1 MATERIAIS

O sistema foi desenvolvido em ambiente Windows 7 Home Premium Versão 2009 Service Pack 1[1]. Foram utilizadas a linguagem PHP 5.2[2], rodando em um servidor Apache versão 2.2[3], Banco de dados MySQL 5.1.32-*Community*[4], Zend Framework versão 1.11.2[5] e o framework javascript jQuery 1.4.4[6].

3.1.1 PHP 5

PHP (*Hypertext Preprocessor*)[7] é uma linguagem de *scripting*, projetada para o desenvolvimento de aplicações web e pode se embutida no código HTML (*Hypertext Markup Language*). Seu código é executado no lado do servidor. O cliente recebe o resultado da execução dos *scripts*, mas não tem acesso ao seu código fonte.

Segundo GONZAGA & BIRCKAN (2011), o PHP foi criado em 1995 por Rasmus Lerdof com o nome *Personal Home Page Tools*, escrito em linguagem C, com a finalidade de apenas resolver um problema pessoal. Posteriormente ele decidiu disponibilizar seu código-fonte para que todos pudessem melhorá-lo. Devido à boa aceitação do primeiro PHP, em 1997 Rasmus Lerdof desenvolveu um sistema para processar formulários, chamado FI (*Form Interpreter*), a união destas ferramentas deu origem à primeira versão compacta da linguagem: PHP/FI.

---

[1] http://windows.microsoft.com/pt-BR/windows/home, visitado em 28/11/2011.
[2] http://www.php.net, visitado em 15/11/2011.
[3] http://www.apache.org, visitado em 15/11/2011.
[4] http://www.mysql.com, visitado em 15/11/2011.
[5] http://*framework*.zend.com, visitado em 15/11/2011.
[6] http://jquery.com/, visitado em 15/11/2011.
[7] http://www.php.net, visitado em 15/11/2011.

Os primeiros recursos de orientação a objetos foram introduzidos no PHP3, por Zeev Suraski que reescreveu o PHP totalmente. É importante destacar que o PHP não é uma linguagem de programação orientada a objetos, ele possui suporte à Orientação a Objetos, ou seja, é possível desenvolver aplicações utilizando-se de programação estrutura ou orientada a objetos.

Pouco tempo depois, Zeev e Andi Gutmans lançaram o PHP4, assim o PHP 3 foi completamente abandonado. Foi dado mais “poder” à *engine* para sanar as necessidades da época e solucionar inconvenientes da versão anterior. Porém ainda existiam sérios pontos negativos no desenvolvimento PHP4, o mais grave era a criação de cópias de objetos, pois a linguagem ainda não trabalhava com apontadores ou *handlers*, como são feitos pelas linguagens Java e Ruby (GONZAGA & BIRCKAN, 2011),

Tal problema foi resolvido na atual versão do PHP 5 que trabalha com *handlers*. O tratamento de objetos foi completamente reescrito, solucionando as carências das versões anteriores e tornando o PHP uma linguagem robusta, capaz de suportar grandes aplicações.

O PHP apresenta algumas características importantes para o desenvolvimento de aplicações WEB, entre elas:

- Velocidade e robustez.
- Tipagem fraca, não é necessário associar o tipo de dado à variável, pois o compilador/interpretador se encarrega de fazer esse tipo de verificação.
- Portabilidade, pois pode rodar na maioria dos sistemas operacionais disponíveis atualmente no mercado.
- Suporte a uma ampla variedade de banco de dados através da camada de abstração chamada PDO (*PHP Data Objects*).
- Sintaxe similar a Linguagem C/C++ e PERL.

3.1.2 Zend Framework 1.11.2

A necessidade de padronização entre os desenvolvedores PHP é considerada um dos objetivos principais do desenvolvimento do *framework* Zend pela equipe Zend.

Um *framework* consiste em um ambiente de trabalho que serve de suporte ao desenvolvimento de *softwares* em um determinado contexto através da disposição de ambientes e ferramentas pré-definidas.

Nada mais é que um conjunto de códigos-fontes, classes, funções e metodologias que visam facilitar o desenvolvimento de novas aplicações.

A principal vantagem de se utilizar um *framework* está no ganho de velocidade de desenvolvimento, pois é possível reaproveitar o trabalho de outra(s) pessoa(s) utilizando-se de suas estruturas e padrões.

Não é preciso “reinventar a roda”, a maioria do código fonte básico, genéricos a várias aplicações, já estará pronto. O trabalho consistirá em desenvolver os pedaços específicos que estarão faltando para a aplicação desejada.

Além disso, a padronização de código-fonte é outra grande vantagem, pois o *framework* “força” os desenvolvedores a trabalharem dentro dos padrões especificados, desta forma os *softwares* desenvolvidos para determinado *framework* poderão ser facilmente compreendidos, reutilizados e modificados por qualquer desenvolvedor deste mesmo *framework*.

Lembrando que a linguagem de programação não é a única coisa que o programador deve conhecer, são necessários também métodos de trabalho e padrões bem definidos. Tudo isto deve estar bem documentado dentro da empresa, pois a troca de um programador deve provocar o mínimo impacto no projeto.

O Zend Framework ou ZF é um *framework* para desenvolvimento de aplicações utilizando da linguagem PHP 5 orientada a objetos. É de código aberto está licenciado como New BSD License.

A arquitetura do ZF permite que o desenvolvedor reutilize componentes entre diferentes aplicações sem requerer outros componentes ZF além das dependências mínimas. O Zend Framework também oferece componentes prontos para muitos requisitos comuns no desenvolvimento de aplicações, incluindo autenticação de usuários, controles de acesso, filtragem/validação de dados, envio de e-mail, Google Data APIs (*Applications Programming Interface*) entre muitos outros (LISBOA, 2008).

O Zend Framework foi anunciado publicamente no ano de 2005 durante a Zend Conference. Neste ano muitos *frameworks* novos como o Ruby on Rails vinham ganhando espaço na comunidade de desenvolvedores, porém ainda não existia nenhum *framework* PHP amplamente utilizado. Os projetistas do ZF

objetivaram unir características de desenvolvimento RAD (*Rapid Application Development* ou Desenvolvimento Rápido de Aplicação) à famosa simplicidade e praticidade que é valorizada na comunidade PHP.

Zend Framework está sob a licença *Open Source Initiative* (OSI) - *Approved New BSD License*. Profissionais que desejam contribuir com códigos devem assinar um *Contributor License Agreement* (CLA) baseado no *Apache Software Foundation's* CLA. As políticas de licenciamento foram estabelecidas de tal forma a impossibilitar quaisquer questões de propriedade intelectual por usuários ZF comerciais (EVANS, 2008).

O Zend Framework apresenta algumas características, entre elas destacam-se:

- Todos os componentes são PHP 5 completamente orientados a objeto; a implementação MVC extensível suportam *layouts* e *templates* baseados em PHP por padrão.
- Suporta múltiplos sistemas de bancos de dados, incluindo MySQL, Oracle, IBM DB2, Microsoft SQL Server, PostgreSQL, SQLite, e Informix Dynamic Server.
- A implementação é flexível de *table gateway* para acessar dados de um banco de dados relacional em um ambiente orientado a objetos.
- Apresenta filtro de dados e validação para fortalecimento da segurança da aplicação; possui gerenciamento de sessão.
- Componente de configuração para promover um gerenciamento de configuração consistente através de Zend Framework e aplicações ZF.
- Composição e entrega de email, recuperação via Mbox, Maildir, POP3 e IMAP4.
- Indexação e busca que suporta o formato de arquivo índice Lucene.
- Internacionalização e localização.
- Criação de formulários usando PHP, arquivos de configuração ou XML (*eXtensible Markup Language*).
- Tecnologias de Identity 2.0 tais como Microsoft InfoCard e OpenID.
- Múltiplos formatos para *services*, incluindo XML-RPC, REST, e Google GData.
- Componente de *logging* simples inspirado por log4j.

- Componente nativo PHP para leitura, atualização e criação de documentos PDF (*Portable Document Format*).
- Serialização de estruturas de dados PHP para e de JSON (*JavaScript Object Notation*) de modo a facilitar o desenvolvimento AJAX (*Asynchronous JavaScript and XML*).
- API para consumir RSS (*Really Simple Syndication*) e alimentadores Atom; bibliotecas cliente para muitos repositórios de services, incluindo Amazon *E-commerce* Service, Akismet, del.icio.us, Flickr, StrikeIron, Yahoo!, Audioscrobbler, e Simpy.

O Zend Framework possui simplicidade e produtividade, pois foi desenvolvido tendo como principal meta a manutenção da simplicidade do desenvolvimento PHP, a livraria promete oferecer 80% das funcionalidades básicas do desenvolvimento de aplicações. A simplicidade objetiva diminuir o tempo necessário no aprendizado, reduzindo assim custos de treinamento e reposição de funcionários para as empresas.

O Zend Framework é uma edição PHP nativa do mecanismo de busca Lucene, um padrão da indústria; possui formato de dados de fácil acesso para aplicações WEB 2.0. Segudno Lisboa (2008), pretende ser o principal local de consumo e publicações de WEB SERVICES, possui classes em PHP 5 de alta qualidade, atendendo às melhores práticas como *Design Patterns*, testes unitários e acoplamento.

O Zend Framework possui licença baseada na BSD License, a Zend Framework's License, a qual assegura que seu código é irrepreensível, se encontra protegido. A Zend requer que contribuidores do Zend Framework assinem o CLA, que se baseia na licença padrão do Apache como forma de proteger a propriedade intelectual do ZF.

A arquitetura do Zend Framework baseia-se no princípio de componentes reutilizáveis, novos componentes são implantados a cada nova versão. A estrutura é simples e permite que os componentes possam ser utilizados de forma independente, não é necessário utilizar todo o *framework* a cada projeto. (Evans, 2008)

A base do ZF consiste em um diretório de classes *(library/Zend)* simulando a organização de uma estrutura de pacotes, permitindo a inclusão de componentes através da inclusão de novas classes dentro do *library/Zend*.

Para a linguagem PHP, além do Zend Framework pode-se destacar os *frameworks*: CakePHP, Symphony, CodeIgniter entre outros.

3.1.3 Zend Studio 7

Zend Studio é um IDE(Integrated Development Environment) para PHP, desenvolvido pela Zend Technologies. É baseado no PDT(projeto liderado pela própria Zend) um plugin de desenvolvimento PHP do Eclipse.

Principais características:

- Integração com Zend Framwork.
- Suporte a MVC.
- Análise semântica do código.
- Suporte para PHP 4 e PHP 5.X.
- Hierarquia de classes e métodos.
- Depuração de script PHP.
- Auto detecção de um Zend Server local.
- View de sevidores.
- Mecanismo rápido de depuração no servidor.
- API Zend Server.
- PHPUnit integrada.
- Integração com o phpDocumentor.
- Suporte para Web services
- Importação de projetos para Zend Studio 5.5

3.1.4 MySQL 5

O MySQL é um banco de dado open source que utiliza a linguagem SQL (*Structured Query Language*, ou Linguagem de Consulta Estruturada). É atualmente uma das tecnologias de banco de dados mais populares do mundo.

Segundo Duarte (2011), o MySQL teve seu desenvolvimento iniciado na Suécia na década de 80. Em 2008 a MySQL AB, desenvolvedora do MySQL foi adquirida pela Sun Microsystems que em 2009 foi comprada pela Oracle.

Ainda segundo Duarte (2011) o grande sucesso do MySQL deve-se a fácil integração com a linguagem PHP apresentando como características:

- Escrito em C e C ++.
- Compatibilidade existem *drivers* ODBC (*Open Data Base Connectivity*), JDBC (*Java Database Connectivity*) e .NET e módulos de interface para diversas linguagens de programação, como Delphi, Java, C/C++, Visual basic, Python, Perl, PHP, ASP e Ruby).
- Suporte total a *multi-threads* usando *threads* diretamente no kernel. Isto significa que se pode facilmente usar múltiplas CPUs (*Central Process Unit*), se disponível.
- Fornece mecanismos de armazenamento transacional e não transacional; um sistema de alocação de memória muito rápido e baseado em processo (*thread*).
- Excelente desempenho e estabilidade.
- Pouco exigente quanto a recursos de hardware; facilidade de uso.
- É um *software* com base na GPL (*General Public License*); contempla a utilização de vários *Storage Engines* como MyISAM, InnoDB, Falcon, BDB, Archive, Federated, CSV, Solid, entre outros.
- Suporte a *Triggers*.
- Suporte a Cursores.
- Suporte a *Stored Procedures* e *Functions*.
- Replicação facilmente configurável.
- Interfaces gráficas (MySQL Toolkit) de fácil utilização cedidos pela MySQL Inc.

3.1.5 MySQL Workbench 5.2 CE

O MySQL Workbench é um software open-source de modelagem de dados para banco MySQL. Lançado em 2010 teve seu desenvolvimento iniciado pelo

MySQL AB e posteriormente pela Sun Microsystems, pretende ser uma evolução do famoso DBDesigner 4.

As principais características do MySQL Workbench são:

- Possibilita trabalhar diretamente com objetos do *schema*;
- Separação do modelo lógico do catalogo de banco de dados.
- Geração de scripts SQL.
- Engenharia reversa de esquemas de banco de dados.
- Sincroniza dos modelos de banco de dados.

3.2 MÉTODO

Para o desenvolvimento do sistema resultado deste trabalho, algumas etapas foram realizadas. A seguir essas etapas estão descritas, juntamente como as suas principais atividades.

a) **Definição preliminar do tipo de sistema:** tendo como principal objetivo a demonstração de um *workflow*, foi decidido o desenvolvimento de um sistema de controle de tarefas e chamados.

b) **Preparo e configuração do ambiente:** o Zend Framework trata-se de um pacote de arquivos e tanto o PHP quanto o Apache possuem um *software* instalador, não apresentando dificuldades para efetuar tal processo. Feito isso, devem ser efetuadas algumas alterações no Apache para que o *framework* funcione corretamente. Na pasta de configuração do Apache, deve-se editar o arquivo `\CONF\HTTPD.CONF` da seguinte maneira:

1. Retirar o comentário da linha:
   ```
   LoadModule rewrite_module modules/mod_rewrite.so
   ```
2. Definir o htaccess:
   ```
   AccessFileName .htaccess
   ```
3. Alterar permissões de diretório, retirando o comentário do seguinte trecho de código:
   ```
   #<Directory />
   # Options Indexes FollowSymLinks
   # AllowOverride None
   #</Directory>
   ```

c) **Requisitos de Hardware**: o projeto foi desenvolvido em um Notebook Intel Core i5 4GB de memória. Ocupando um espaço de 30MB.

4 RESULTADOS

Este capítulo apresenta o sistema, a sua modelagem básica, a descrição do mesmo com telas e suas funcionalidades, a preparação do ambiente para a implementação do sistema e exemplos do código produzido.

4.1 MODELAGEM DO SISTEMA

Nessa parte serão apresentados os Diagramas de Entidades e Relacionamento (DER) utilizados na criação das tabelas e classes. Estes diagramas foram desenvolvidos utilizando o software Mysql Workbeanch 5.2 CE[8].

Basicamente o banco de dados gira em torno de duas tabelas principais, uma é responsável pelo armazenamento das entidades do sistema e na outra ficam registrados os chamados e tarefas. As demais tabelas servem basicamente para complementar os registros gravados nas principais.

4.1.1 Diagramas de Entidades e Relacionamentos

O DER do cadastro de entidades, mostrado na **Figura 4,** engloba os usuários, clientes e fornecedores em uma mesma tabela, para cada um é possível registrar vários contatos e endereços.

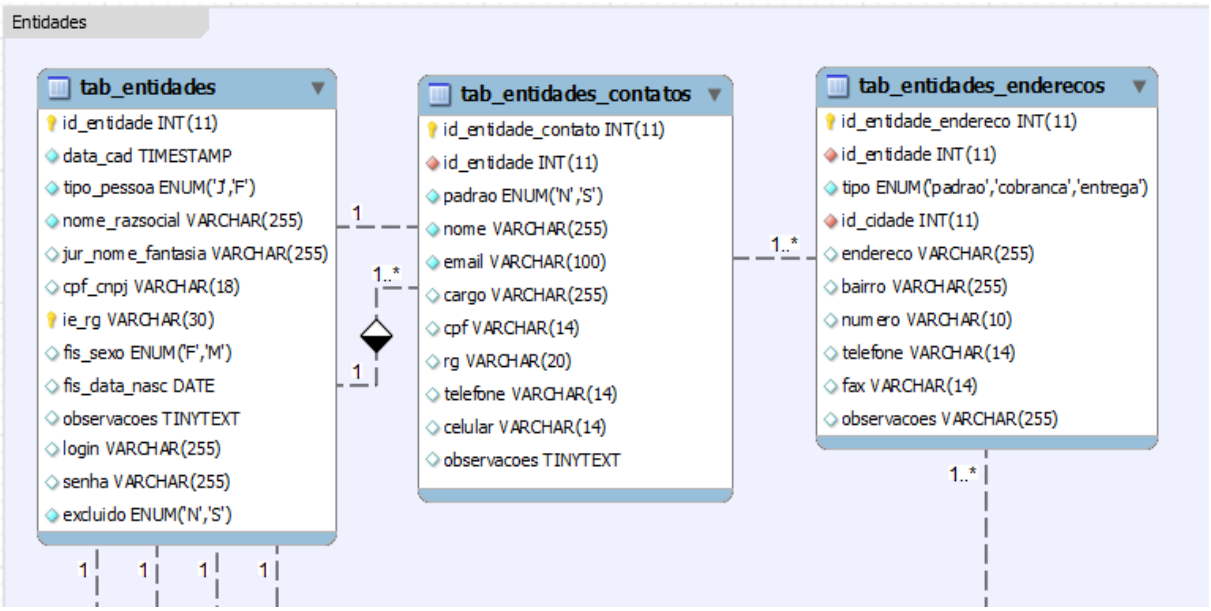

Figura 4- DER de Entidades

[8] http://dev.mysql.com/downloads/workbench/, visitado em 28/11/2011.

Para o cadastro de chamados e tarefas, com o DER mostrado na **Figura 5**, foi utilizada uma mesma tabela por se tratarem de cadastros muito parecidos, o que os diferencia é a coluna “tipo”.

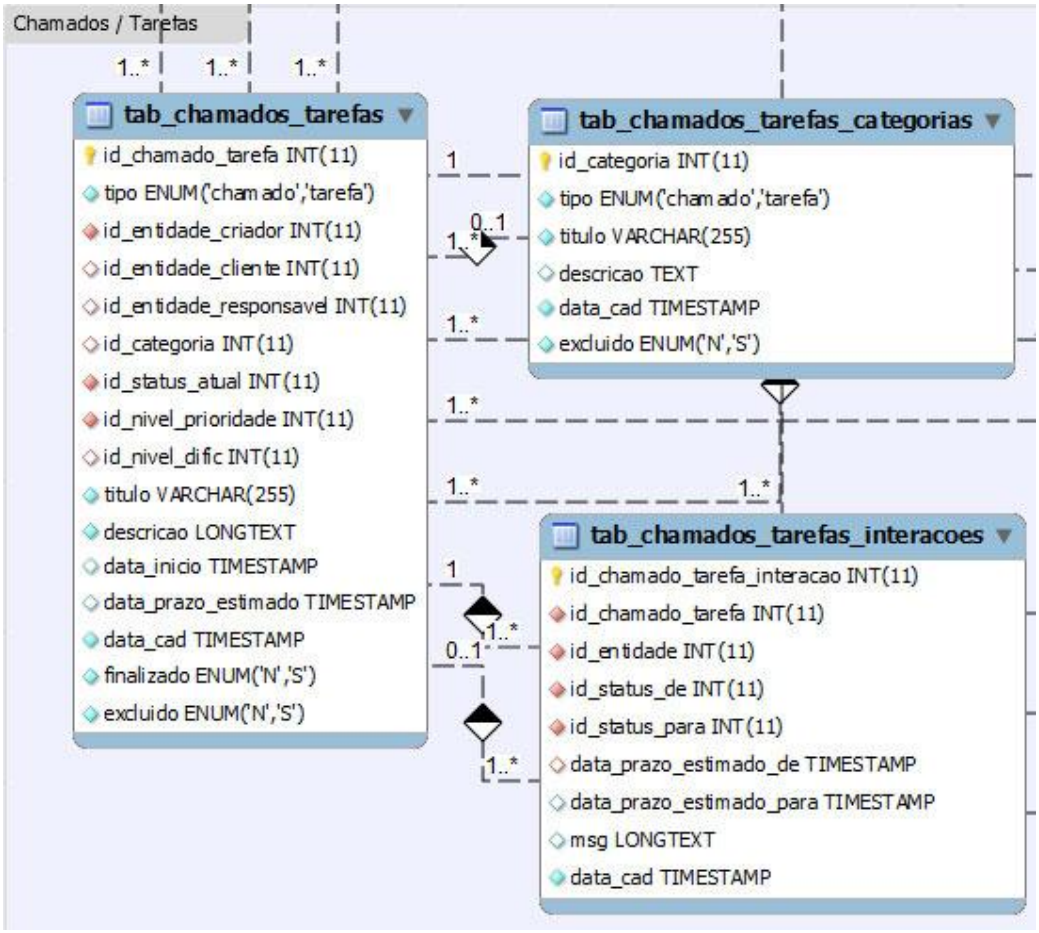

Figura 5- DER de Chamados/Tarefas, Categorias e Interações

Os chamados e tarefas podem ser classificados por categoria, status, nível de prioridade e nível de dificuldade, como mostra o DER da **Figura 6**. A cada mudança recebida será gravada uma interação, mantendo assim um histórico de ações.

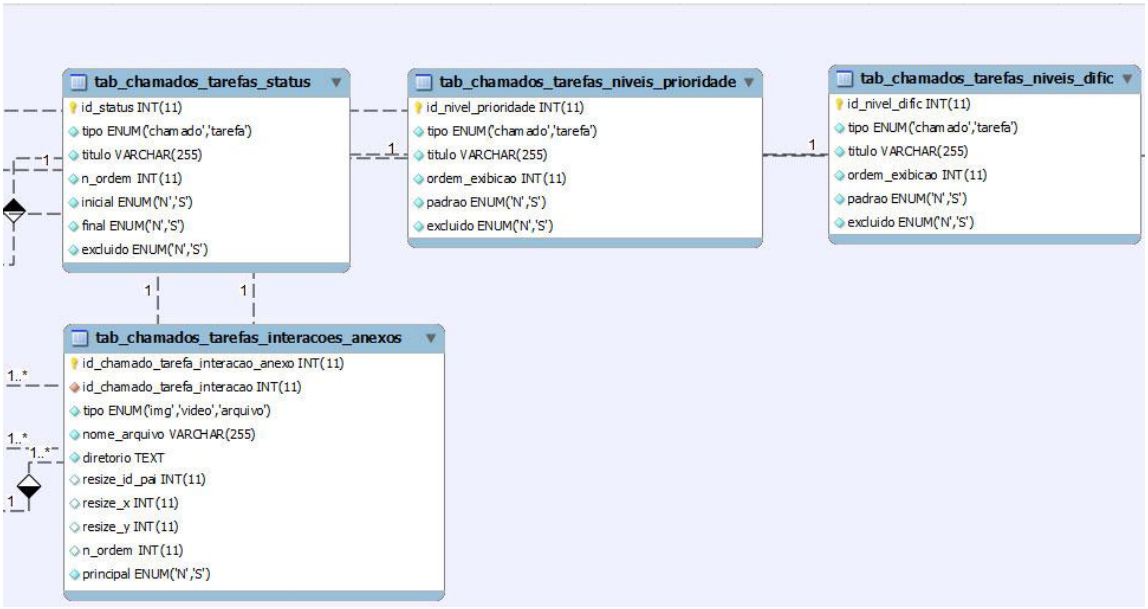

Figura 6- DER de Status Prioridades, Níveis de Dificuldade e Anexos das Interações

O DER do gerenciamento de metas, apresentado na **Figura 7**, mostra que os chamados e as tarefas são agrupados quando possuem um objetivo em comum.

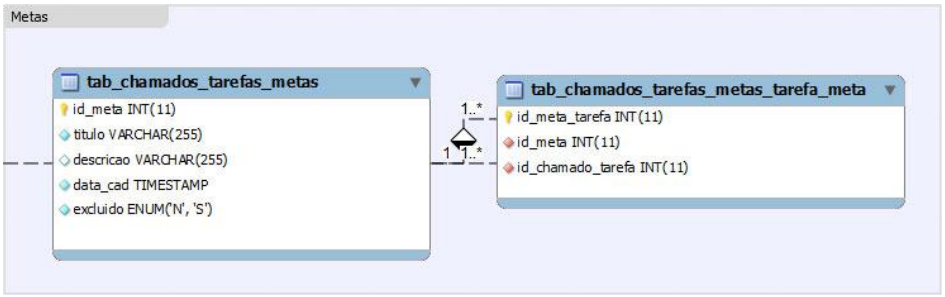

Figura 7- DER de Metas

4.1.2 Tabelas de Dados

Na **Tabela 1** são apresentadas as tabelas de dados do sistema descrevendo um número de ordem, o nome da tabela com a chave primária[9], sua descrição ou finalidade e os relacionamentos apresentado a chave estrangeira[10].

Tabela 1- Tabelas de Dados do Sistema Proposto

	Tabela (chave primária)	Descrição	Relacionamento (chave estrangeira)
1	tab_entidades (id_entidade)	Entidades	
2	tab_entidades_contatos (id_entidade_contato)	Contatos referentes a entidade	id_entidade(tab_entidades)
3	tab_entidades_enderecos (id_entidade_endereco)	Endereços da entidade	id_entidade(tab_entidades) id_cidade(tab_cidades)

[9] Chave primária (ou *Primary Key*): campo que apresenta um valor diferente para cada registro da tabela.
[10] Chave estrangeira (ou *Foreing Key*): campo de relacionamento representa a chave primária na tabela relacionada.

4	tab_chamados_tarefas (id_chamado_tarefa)	Chamados e tarefas	id_entidade_criador(tab_entidades) id_status_atual(tab_chamados_tarefas_status) id_nivel_prioridade(tab_chamados_tarefas _niveis_prioridade) id_nivel_dific(tab_chamados_tarefas _niveis_dific)
5	tab_chamados_tarefas_categorias (id_categoria)	Categorias de chamados/tarefas	
6	tab_chamados_tarefas_interacoes (id_chamado_tarefa_interacao)	Interações efetuadas nos chamados/tarefas	id_chamado_tarefa(tab_chamados_tarefas) id_entidade(tab_entidades) id_status_de(tab_chamados_tarefas_status) id_status_para(tab_chamados_tarefas_status)
7	tab_chamados_tarefas_interacoes_anexos (id_chamado_interacao_anexo)	Anexos das interações	id_chamado_tarefa_interacao (tab_chamados_tarefas_interacoes)
8	tab_chamados_tarefas_status	Status dos chamados / tarefas	
9	tab_chamados_tarefas_niveis_prioridade	Níveis de prioridade dos chamados / tarefas	
10	tab_chamados_tarefas_niveis_dific	Níveis de dificuldade dos chamados / tarefas	
11	tab_chamados_tarefas_metas (id_meta)	Metas	
12	tab_chamados_tarefas_metas_tarefa_meta (id_meta_tarefa)	Tabela de ligação entre tarefas e metas	id_meta(tab_chamados_tarefas_metas id_chamado_tarefa(tab_chamados_tarefas)

4.2 DESCRIÇÃO DO SISTEMA

O sistema baseia-se em dois cadastros, o de Chamados e o de Tarefas, a ideia é que novas solicitações sejam tratadas como Chamados e só após sofrerem análise elas seriam convertidas em Tarefas, e repassadas ao setor de desenvolvimento.

As telas foram desenhadas baseando-se na simplicidade e organização dentro dos padrões indicados pela W3C (*World Wide Web Consortium*). Facilitando a experiência do usuário através de um design limpo, com foco no conteúdo.

4.2.1 Tela de Login

Tela inicial do sistema com formulário para efetuação de *login* é apresentada na **Figura 8.**

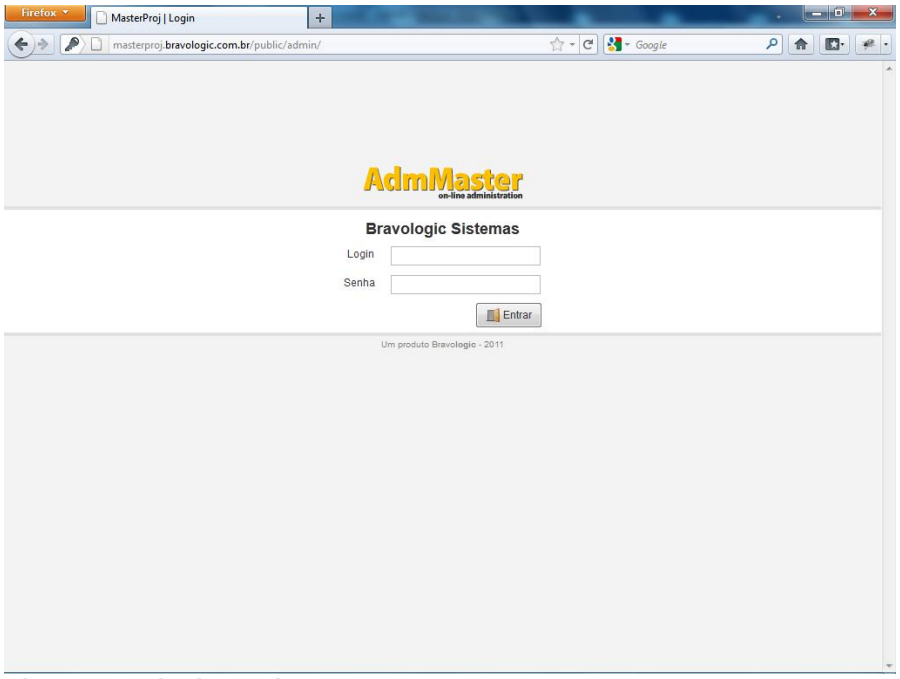

Figura 8- Tela de Login

4.2.2 Tela Principal

Na versão atual, a tela principal do sistema, mostrada na **Figura 9,** exibe um menu ao topo, nome do usuário logado à direita, e conteúdo vazio, nas próximas versões serão implementadas telas iniciais de acordo com a função do usuário no sistema.

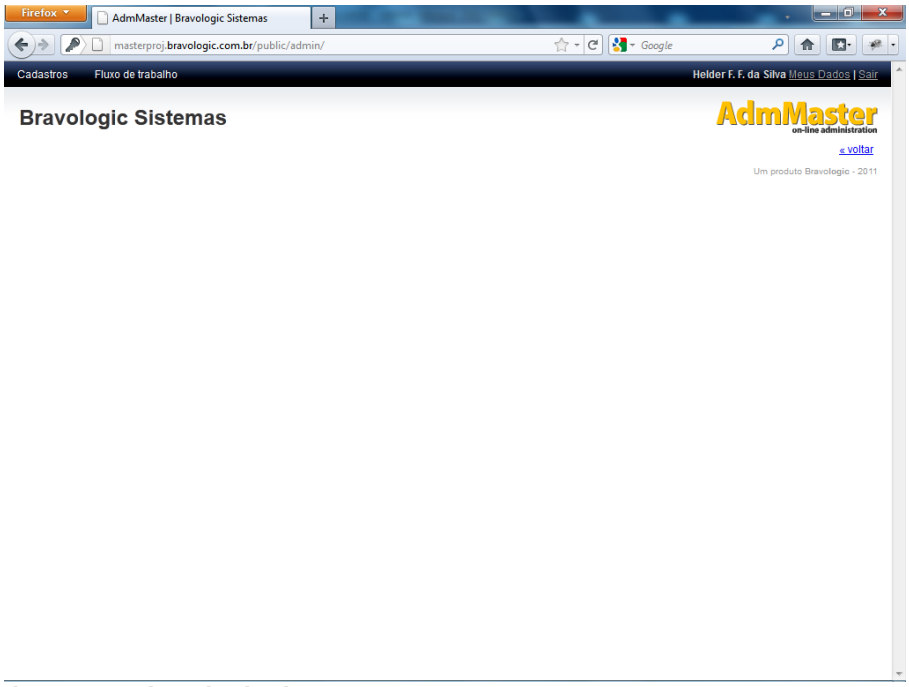

Figura 9- Tela Principal

4.2.3 Menus

Os menus, mostrados na **Figura 10,** foram divididos em duas categorias principais, o menu *Cadastros*, com os cadastros básicos do sistema, no momento apenas o cadastro de *Entidades.* O menu *Fluxo de Trabalho* possui os itens principais *Chamados* e *Tarefas*, cada um com um sub-menu de cadastros complementares, além do menu de geração de gráfico de Gantt (em desenvolvimento).

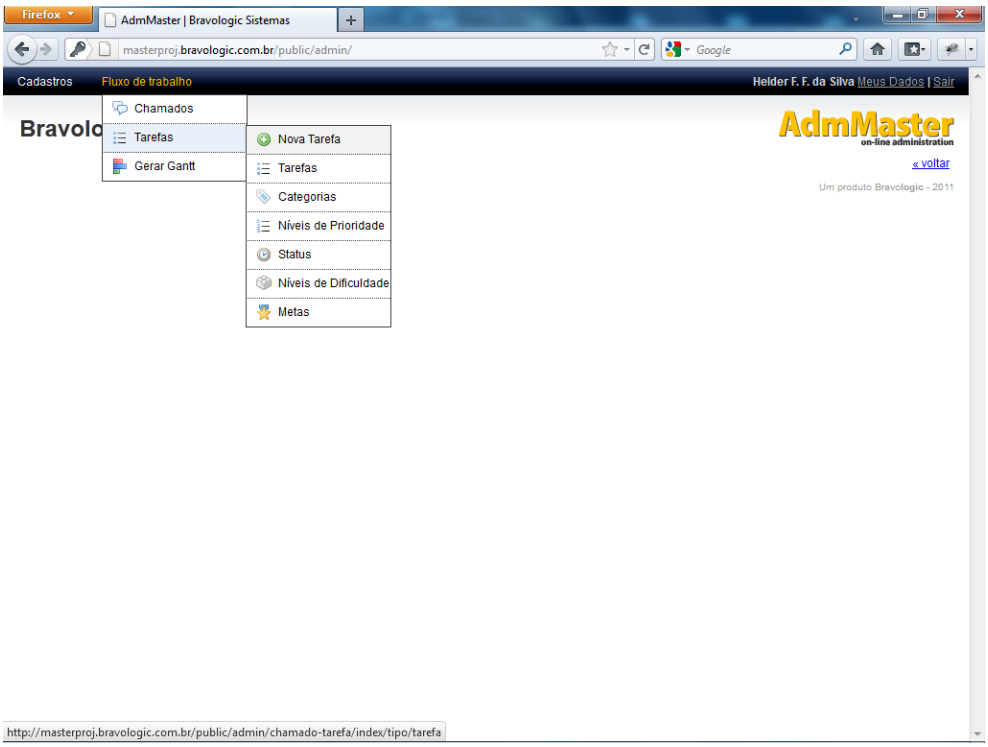

Figura 10- Menu do Sistema

4.2.4 Cadastro de Entidades - Listagem

Cadastro de todas as entidades relacionadas ao sistema sejam elas usuários, clientes ou fornecedores. Ao acessar é exibida uma tela de listagem com as Entidades já cadastradas, bem como opção de pesquisar, cadastrar uma nova, excluir selecionados ou editar um registro no botão a direita de cada linha, mostrado na **Figura 11.**

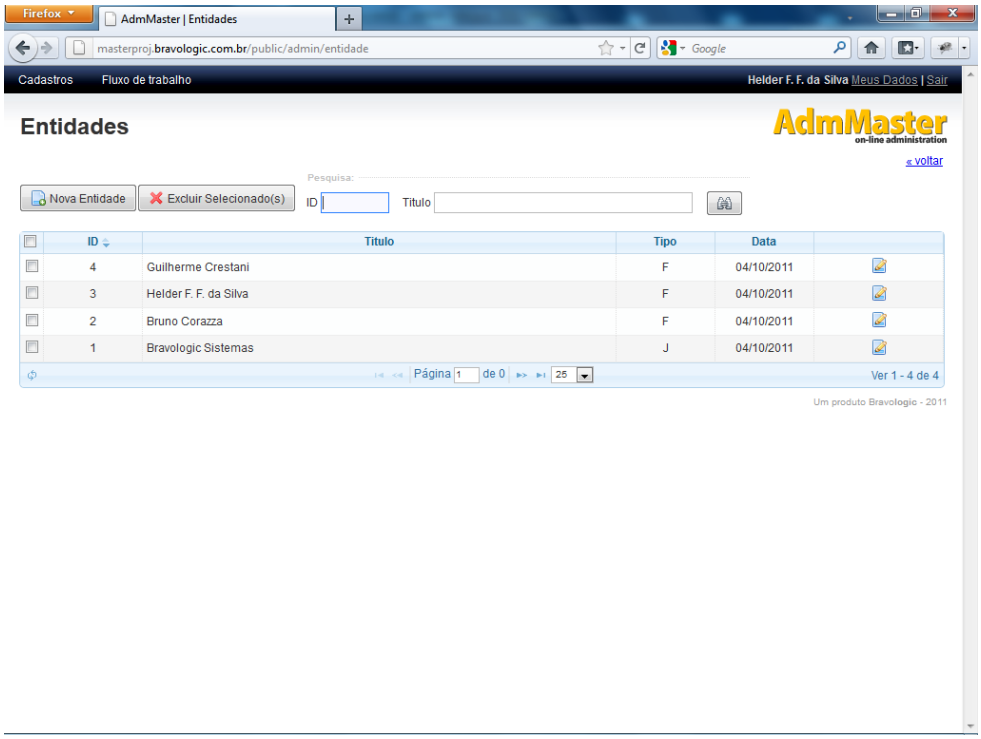

Figura 11- Listagem de Entidades Cadastradas

4.2.5 Cadastro de Entidades - Formulário

Ao adicionar ou editar uma entidade, uma tela composta de barra de navegação e formulário com os dados da entidade é exibida como mostrado na **Figura 12.**

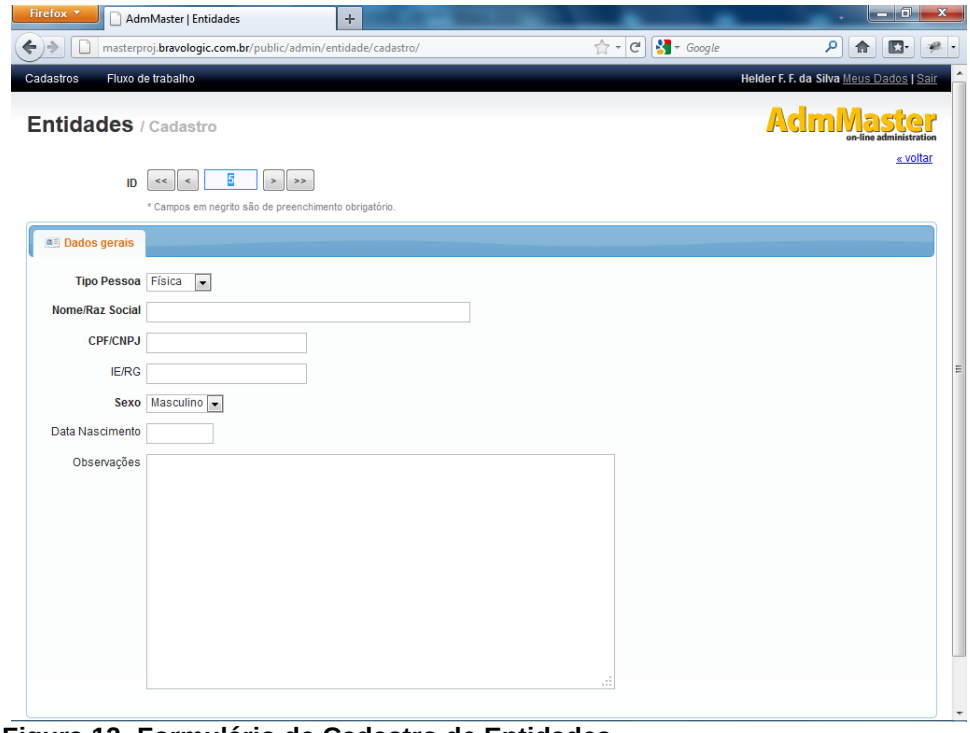

Figura 12- Formulário de Cadastro de Entidades

4.2.6 Cadastro de Tarefas - Listagem

Devido à grande semelhança os Chamados e Tarefas são gravados em uma mesma tabela, possuindo apenas separação lógica. A **Figura 13** mostra a tela de listagem de Tarefas, utilizando o mesmo padrão da listagem de Entidades, são exibidos botões para cadastro, exclusão e edição de itens.

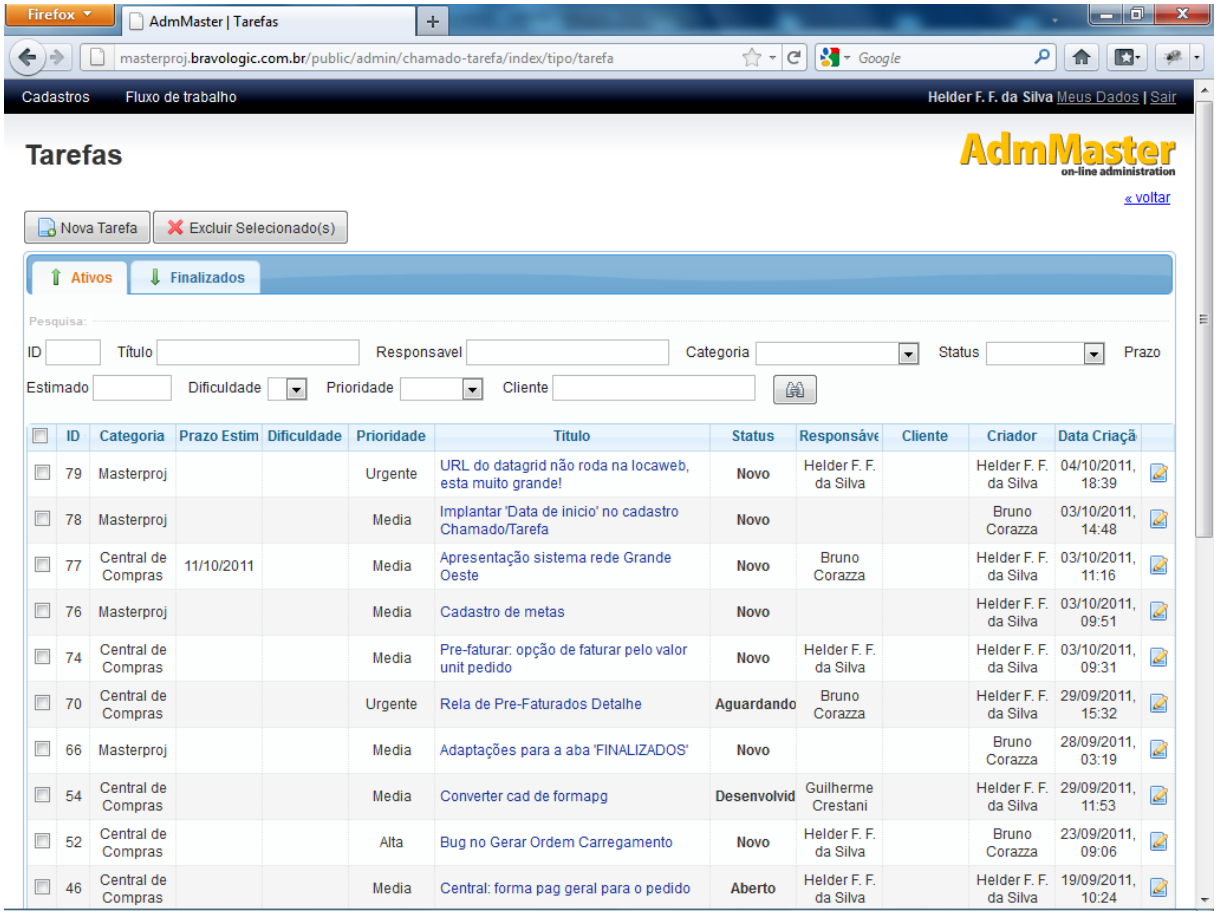

Figura 13- Listagem de Tarefas Cadastradas

Existem duas diferenças nos cadastros de chamados / tarefas:

1) **Abas de Ativos e Finalizados:** por questão organizacional, os chamados / tarefas finalizados, são separados na segunda aba, onde também podem ser reativados.
2) **Possibilidade de interação:** ao clicar sobre o título de um chamado / tarefa, é possível interagir, enviando mensagens, alterando status e prazos como mostrado a **Figura 14.**

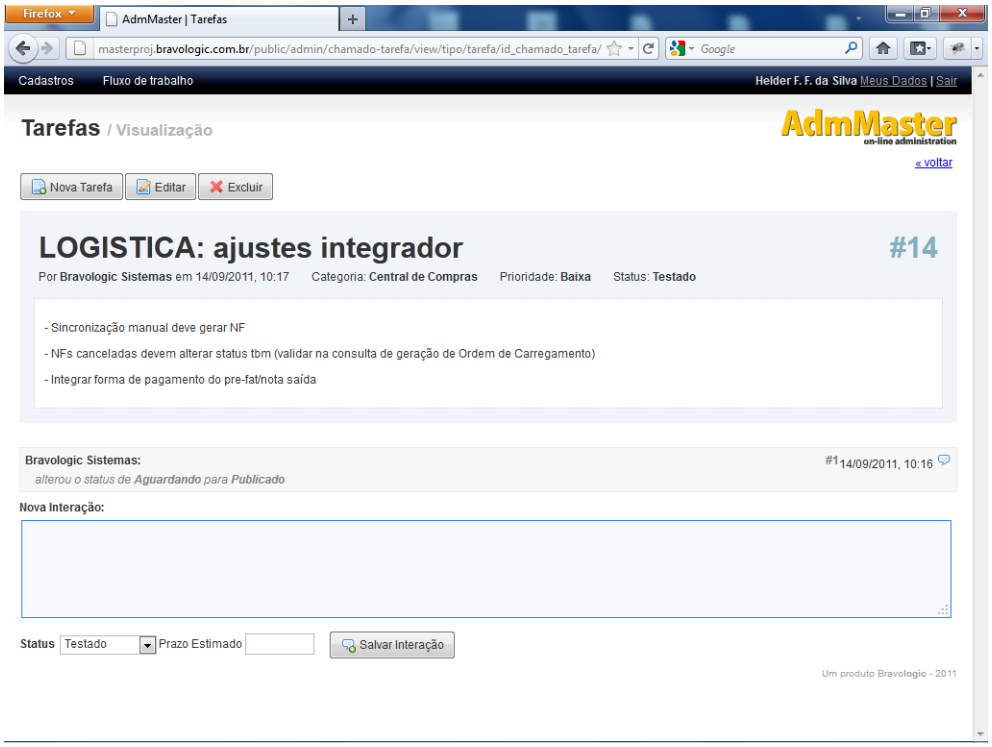

Figura 14- Tela de Registro de Interações de uma Tarefa

4.2.7 Cadastro de Tarefas - Formulário

Organizado da mesma maneira que o cadastro de Entidades, seguindo os padrões do sistema, como mostrado na **Figura 15.**

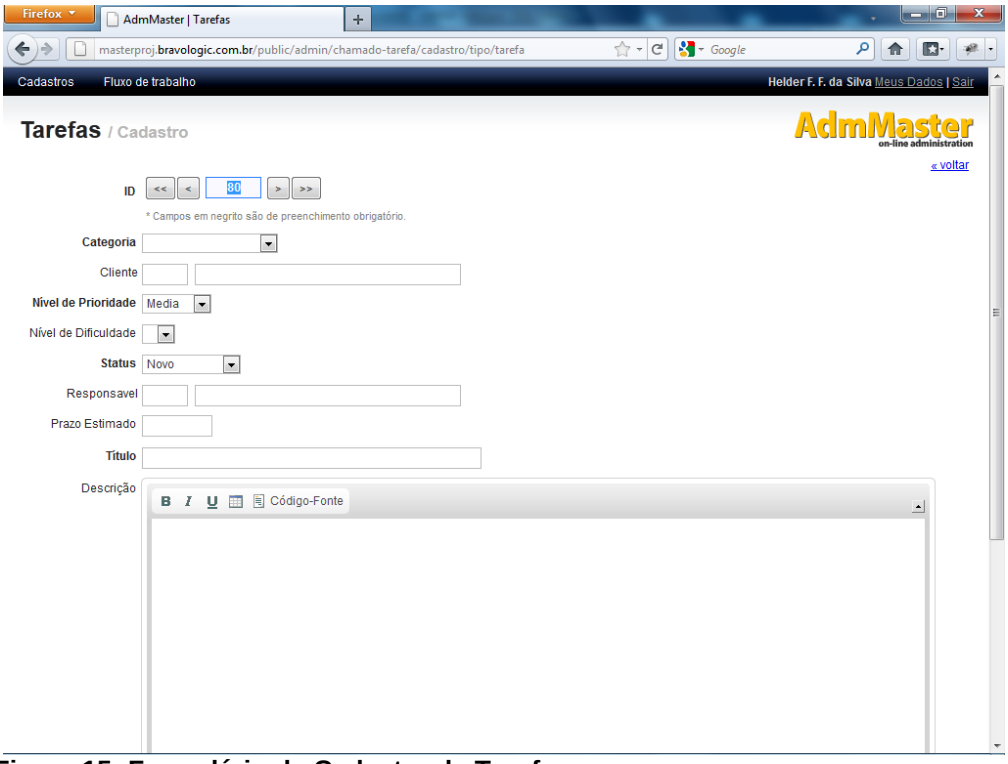

Figura 15- Formulário de Cadastro de Tarefas

4.3 IMPLEMENTAÇÃO DO SISTEMA

Este item trata aspectos das ferramentas e métodos utilizados no desenvolvimento do sistema proposto.

4.3.1 A Estrutura de Arquivos do Zend Framework

A árvore de diretórios do sistema, mostrada na **Figura 16,** segue os padrões recomendados pelos desenvolvedores do Zend Framework segundo http://framework.zend.com/docs/, acessado em julho de 2011.

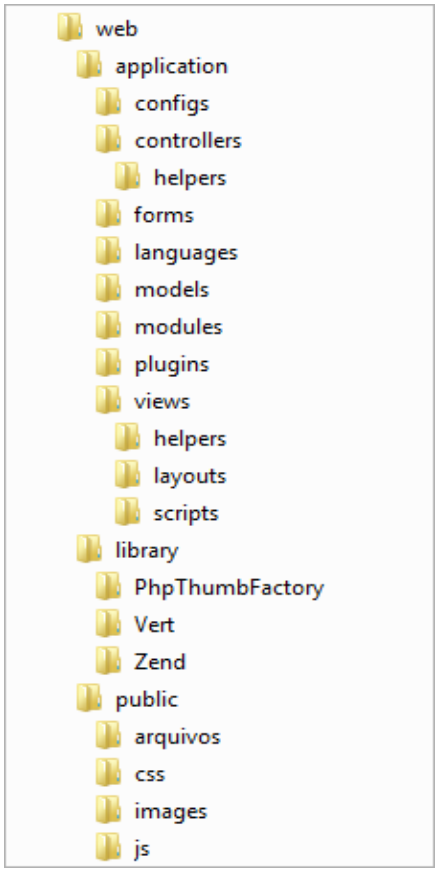

Figura 16- Estrutura Padrão Recomendada pela Equipe Zend

A **Tabela 2** mostra a função de cada pasta na estrutura padrão. O sistema é dividido em três pastas principais: **a)** *Application* onde fica o código fonte da lógica do sistema em si; **b)** *Library*, que contém a livraria Zend entre outras; e, **c)** *Public* é onde ficam os arquivos de mídia, javascript, css ou qualquer tipo de arquivo que possa ser baixado pela URL.

Tabela 2- Função de cada Pasta no Zend Framework

Diretório	Descrição
application\	Diretório da aplicação
application\configs	Arquivos de configurações padrões da aplicação
application\controller\	Diretório dos arquivos da camada de controle da aplicação(*Controllers*)
application\controller\helpers	Diretório onde serão armazenados os *helpers* utilizados na camada de controle
application\forms	Diretório dos arquivos de formulários
Application\languages	Diretório dos arquivos de tradução das palavras utilizadas na aplicação
application\models\	Diretório dos arquivos da camada de modelos da aplicação(*Models*)
application\modules\	Diretório dos arquivos em caso de sistema um modularizado, cada módulo terá sua subestrutura de arquivos
application\plugins\	Diretório onde serão armazenados os *plugins* da aplicação
application\views	Diretório onde serão criados os arquivos utilizados na camada de visualização(*View*)
application\views\helpers	Diretório onde serão armazenados os *helpers* utilizados na camada de visualização.
application\views\layouts	Diretório onde serão armazenados os layouts a serem utilizados pelas *views*.
application\views\scripts	Diretório onde serão armazenados os arquivos HTML.
Library	Diretório onde serão armazenadas as bibliotecas php.
Public	Diretório de arquivos que podem ser acessados diretamente pelo navegador (imagens, estilos CSS, livrarias Javascript)

4.3.2 Front Controller

Aplicações MVC geralmente requerem um grande número de controladores, porém boa parte das requisições aos controladores, por regra de padronização, segurança e autenticação de usuários, necessita do mesmo tratamento.

Não seria viável repetir o mesmo código em todos os controladores, para isso existe o controlador central ou *Front Controller* que recebe as requisições, faz o tratamento necessário e direciona ao controlador requisitado.

A **Figura 17** exibe o fluxo de processamento da aplicação utilizando-se de um *Front Controller*.

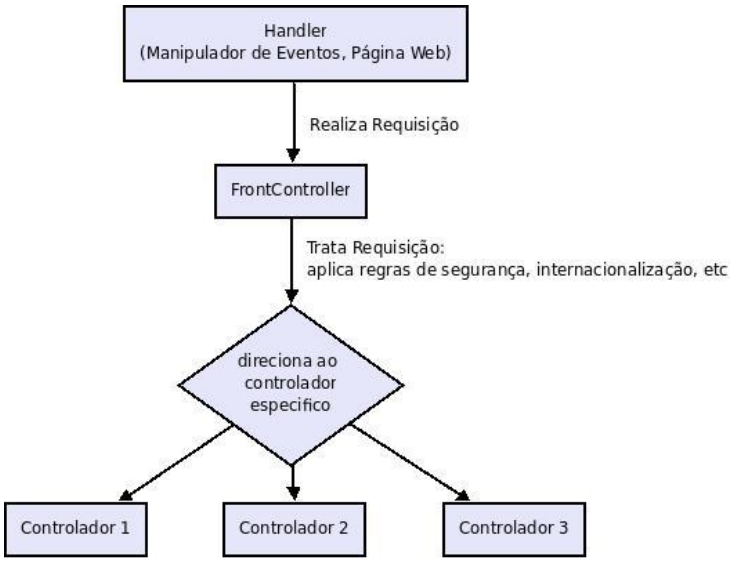

Figura 17- Fluxo de Processamento com Front Controller

O *Zend_Controller_Front* implementa o padrão *Singleton*, significando que somente uma instância dele pode estar disponível em qualquer tempo e registra um *plugin* de entrada que por si só, permite que vários eventos sejam disparados para outros *plugins* observadores. Em muitos casos, isto permite ao desenvolvedor a oportunidade para mudar o processo de despacho do *site* sem a necessidade de estender o *Front Controller* para adicionar funcionalidade.

4.3.3 A Camada de Modelo (*Model*)

Na camada modelo foram criadas classes, as quais acessarão tabelas do banco de dados como objetos, os arquivos ficam dentro da pasta *application/models*.

Ocasionalmente foi desenvolvido um modelo padrão (PadraoModel.php), que não referencia nenhuma tabela, mas contém métodos de manipulação de dados os quais os outros *models* e a camada de controle utilizarão para manipular a base.

Feito isso, foram desenvolvidas as classes das tabelas do sistema utilizando a classe *Zend_db* que é nativa do *framework*. O **Quando 1** exibe o código da criação do modelo de Entidades (EntidadeModel.php).

```
<?php
/**
 * Modelo da Classe Entidades
 * @author Helder F. F. Silva
 *
 */
class App_Model_EntidadeModel  extends App_Model_PadraoModel{
        public $_name      = 'tab_entidades';
        public $_primary = 'id_entidade';
}
?>
```

Quadro 1- Código do Modelo de Entidades

É preciso definir apenas o nome da tabela e sua chave primária. A partir daí a classe *Zend_db_Table*, de onde o *PadraoModel* é estendido, irá tratar a tabela *tab_entidades* como uma classe e cada registro como objeto. Da mesma forma foram criados todos os modelos do sistema.

4.3.4 A Camada de Controle (*Controller*)

Os *controllers* podem ser definidos como as classes que receberão as requisições das *views* (camada de visualização), processarão as informações,

buscarão dados da camada de modelo (quando necessário) e retornarão o resultado para a camada de visualização.

Dentro dos *controllers* são definidas as *actions* que nada mais são do que métodos da classe. Na estrutura do Zend Framework, cada classe *controller* representa uma pasta na estrutura dos arquivos, e cada *action* um arquivo contido nesta pasta, como mostra o **Quadro 2**. A estrutura dos arquivos das *actions* da classe *EntidadeController* é apresentada na **Figura 18**.

```
/**
 * Entidade Controller
 *
 * @author Helder F. F. Silva
 * @version
 */
class EntidadeController extends Zend_Controller_Action {

        public function indexAction() {
                [... processamento da pagina inicial, listagem de Entidades...]
        }

        public function cadastroAction(){
                [... processamento da página de cadastro de entidades...]
        }
}
```

Quadro 2- Código EntidadeController.php

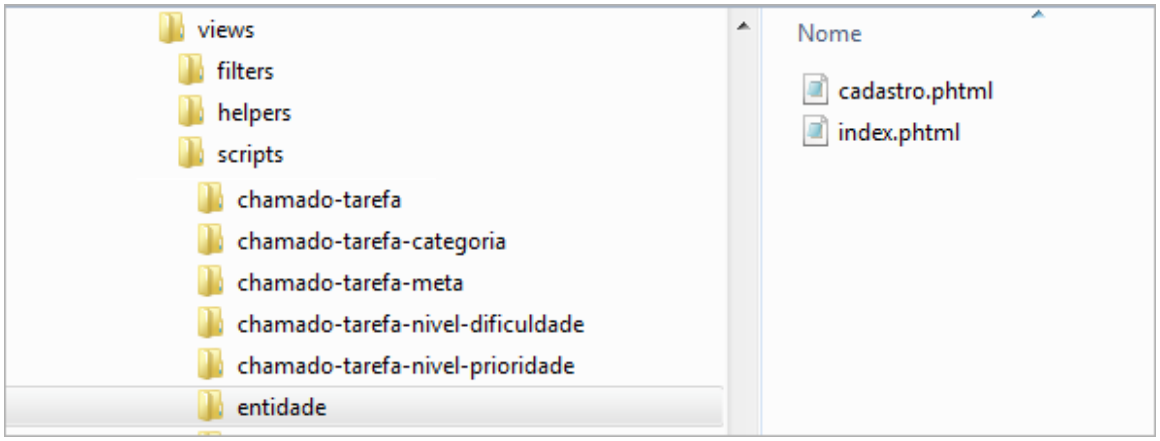

Figura 18- Arquivos das Actions do EntidadeController

4.3.5 A Camada de Visualização (*View*)

Camada responsável pela renderização dos resultados de processamento. Ela pode ser definida como um sistema de *template*, permitindo manter a lógica de exibição completamente separada da lógica de negócio. É nela onde fica todo o código HTML, JavaScript, JSON, XML entre outros.

No Zend Framework, a camada de apresentação é representada pela classe *Zend_view*, que é instanciada automaticamente pelo *controller*, para acessá-la na camada de controle, utiliza-se *$this->view*, isso garante o isolamento das variáveis utilizadas para o controle de negócio, só serão passadas para a camada de visualização as variáveis que forem configuradas com este propósito, como mostra o **Quadro 3**.

```
$this->view->variavel = “Setando variável para a camada de exibição”;
```

Quadro 3- Setando Variável para a Camada de Visualização

Ao final da execução da *action* do controlador, o método *view->render()* é chamado automaticamente, para que o resultado possa ser enviado para o dispositivo de saída.

A variável exemplificada acima pode ser acessada na camada de visualização como mostra o **Quadro 4**. Essa é basicamente a forma de comunicação entre as duas camadas.

```
<?php
        // Escrevendo variável
        echo $this->variável;
?>
```

Quadro 4- Imprimindo Variável na Camada de Visualização

4.3.6 Plugins

Plugins são utilizados quando se deseja executar algo em cada requisição, independente do controlador.

A arquitetura dos *controllers* do Zend Framework inclui um sistema de *plugins* que permite que o código dos *plugins* seja executado em determinados eventos durante a execução do aplicativo. Para cada evento do *front controller* são chamados os métodos dos *plugins* registrados a ele. A **Tabela 3** exibe os métodos que um *plugin* possui.

Tabela 3- Métodos de um Plugin

Método	Descrição
routeStartup()	chamado antes que Zend_Controller_Front utilize o router para processar a requisição.
routeShutdown()	chamado após o roteamento da requisição.
dispatchLoopStartup()	chamado antes que Zend_Controller_Front entre no loop do dispatcher.
preDispatch()	chamado antes da action seja dispachada pelo dispatcher.
postDispatch()	chamado após a action ser dispachada pelo dispatcher.
dispatchLoopShutdown()	chamado após Zend_Controller_Front sair do loop do dispatcher.

A pasta onde ficarão armazenados os *plugins* deve ser declarada no *bootstrap*. Cada classe e *plugins* deve obrigatoriamente estender a classe *Zend_Controller_Plugin_Abstract* e deve ser registrada no *FrontController* da seguinte forma: *$this->_front->registerPlugin(new funcLogin()*).

4.3.7 Helpers

Helpers são utilizados quando a funcionalidade for compartilhada, mas será utilizada de forma seletiva dentro dos controladores (*action helpers*) ou *views* (*view helpers*). Também permitem inserir funcionalidades antes ou após qualquer *action*.

Da mesma maneira que nos *plugins*, os *helpers* possuem os métodos *preDispatch()* e *postDispatch()*, que se existirem, serão chamados automaticamente. Porém também possuem o método *Direct()*, que quando executado faz uma chamada diretamente ao *helper*.

Seja para criação de um *loop* para preencher dados de tabela ou de *combobox*, seja para formatação de nomes/datas ou qualquer outro fim, o *helpers* tem como principal finalidade a abstração de códigos.

A estrutura padrão de um *Action Helper* é mostrada no **Quadro 5**.

```
<?php
require_once 'Zend/Loader/PluginLoader.php';
require_once 'Zend/Controller/Action/Helper/Abstract.php';

Class Zend_Controller_Action_Helper_Exemplo extends Zend_Controller_Action_Helper_Abstract {
	/**
	 * @var Zend_Loader_PluginLoader
	 */
	public $pluginLoader;
	/**
	 * Construtor
	 * @return void
	 */
	public function __construct() {
		// TODO Auto-generated Constructor
		$this->pluginLoader = new Zend_Loader_PluginLoader ( );
	}
	/**
	 * Método principal
	 */
	public function direct() {
	}
}
?>
	Estrutura padrão de um View Helper:
<?php
require_once 'Zend/View/Interface.php';

class Zend_View_Helper_Exemplo {
```

```
	/**
	 * VerificaNavegador
	 * @return str
	 */
	public function Exemplo() {

	}
}
?>
```

Quadro 5- Estrutura Padrão de um Action Helper

O **Quadro 6** mostra o *Action Helper* E-mail responsável por disparar *e-mails.*

```
<?php
require_once 'Zend/Loader/PluginLoader.php';
require_once 'Zend/Controller/Action/Helper/Abstract.php';

Class Zend_Controller_Action_Helper_Exemplo extends Zend_Controller_Action_Helper_Abstract {
	/**
	 * @var Zend_Loader_PluginLoader
	 */
	public $pluginLoader;


	/**
	 * Construtor
	 * @return void
	 */
	public function __construct() {
		// TODO Auto-generated Constructor
		$this->pluginLoader = new Zend_Loader_PluginLoader ( );
	}


	/**
	 * Método principal
	 */
	public function direct() {
	}
}
?>
	Estrutura padrão de um View Helper:

<?php
require_once 'Zend/View/Interface.php';

class Zend_View_Helper_Exemplo {

	/**
	 * VerificaNavegador
	 * @return str
	 */
	public function Exemplo() {

	}
}
?>
```

Quadro 6- Action Helper E-mail

O **Quadro 7** mostra o *View Helper VerificaNavegador* responsável por verificar o navegador utilizado pelo usuário.

```
<?php
/**
 * Verifica navegador do usuário,
 * exibe mensagem caso o navegador seja diferente do recomendado
 *
 * NAVEGADOR RECOMENDADO: INTERNET EXPLORER 9.0
*/

class Zend_View_Helper_VerificaNavegador {
        /**
         * VerificaNavegador
         * @return str
         */
        public function VerificaNavegador() {
                $tag = '
                <div class="ClassErroNavegador">
                 <div id="div_erro_navegador" style="display: none; margin-bottom: 10px">
                   <font color="red" style="font-weight: bold">Atenção!</font><br>
                        O sistema poderá apresentar incompatibilidade<br>
                        com seu navegador. Aconselhamos a utilização<br>
                        do Internet Explorer 9.0. <br><br>
                 </div>
                <script>
                erro_navegador = false;
                if(navigator.appName == \'Microsoft Internet Explorer\'){
                                versao = navigator.appVersion;
                pos = versao.indexOf("MSIE 7.0", 0);
                if(pos == -1)
                        erro_navegador = true;
                }else{
                        erro_navegador = true;
                }

                if(erro_navegador == true)
                document.all.div_erro_navegador.style.display = \'\';
                </script>
                <div style="padding-left: 7px; color: #666666">Desenvolvimento Bravologic.'</div>
                </div>
                ';

                return $tag;
        }

}
```

Quadro 7- View Helper Verifica Navegador

4.3.8 jQuery na Camada de Visualização

A biblioteca *jQuery*, bem como seus *plugins*, tiveram seus arquivos salvos na pasta *public/scripts* e para que funcionassem bastou apenas incluí-las no topo do arquivo, assim como se inclui qualquer arquivo *.js* como mostra o **Quadro 8**.

```
<script type="text/javascript" src="/public/scripts/jquery/jquery-1.3.2.js"></script>
```

Quadro 8- Inclusão de Arquivos JS

Para a utilização do *framework* de *javascript jQuery* no desenvolvimento do sistema, utilizou-se requisições Ajax com *jQuery* onde o módulo Ajax é muito simples e contempla tudo que é necessário para uma requisição. Existem diversas funções

para fazer o *HTTP request*, a utilizada no sistema em questão foi a *$.ajax*, que por ser a implementação de mais baixo nível, é a mais flexível.

O código do **Quadro 9** faz uma requisição POST ao arquivo index.php, enviando a variável ‘id’ com o valor ‘1’, Ao retornar a página (*sucess*) é executado um `alert()` com os dados recebidos da página requisitada.

```
$.ajax({
  type: "POST",
  url: "index.php",
  data: "id=1",
  assync: false,
  success: function(data){
    alert( data );
  }
});
```

Quadro 9- Requisição AJAX com jQuery

Também foi utilizado o *plugin jqGrid*, mostrado na **Figura 19,** para grande parte da listagem de dados, tanto do sistema administrativo, quanto do *site* foram utilizados o *jQgrid*, um *datagrid* desenvolvido em *jQuery* considerado um dos mais completos disponíveis atualmente, seguem algumas de suas principais características:

- Integração com PHP e MySQL;
- Carrega os dados via Ajax;
- Busca e filtros otimizados;
- Paginador;
- Ordenação e redimensionamento de colunas;
- Mostrar e esconder colunas; editar os registros das linhas no lugar;
- G*rid* com *subgrid* ;
- *Drag and Drop* de linhas.

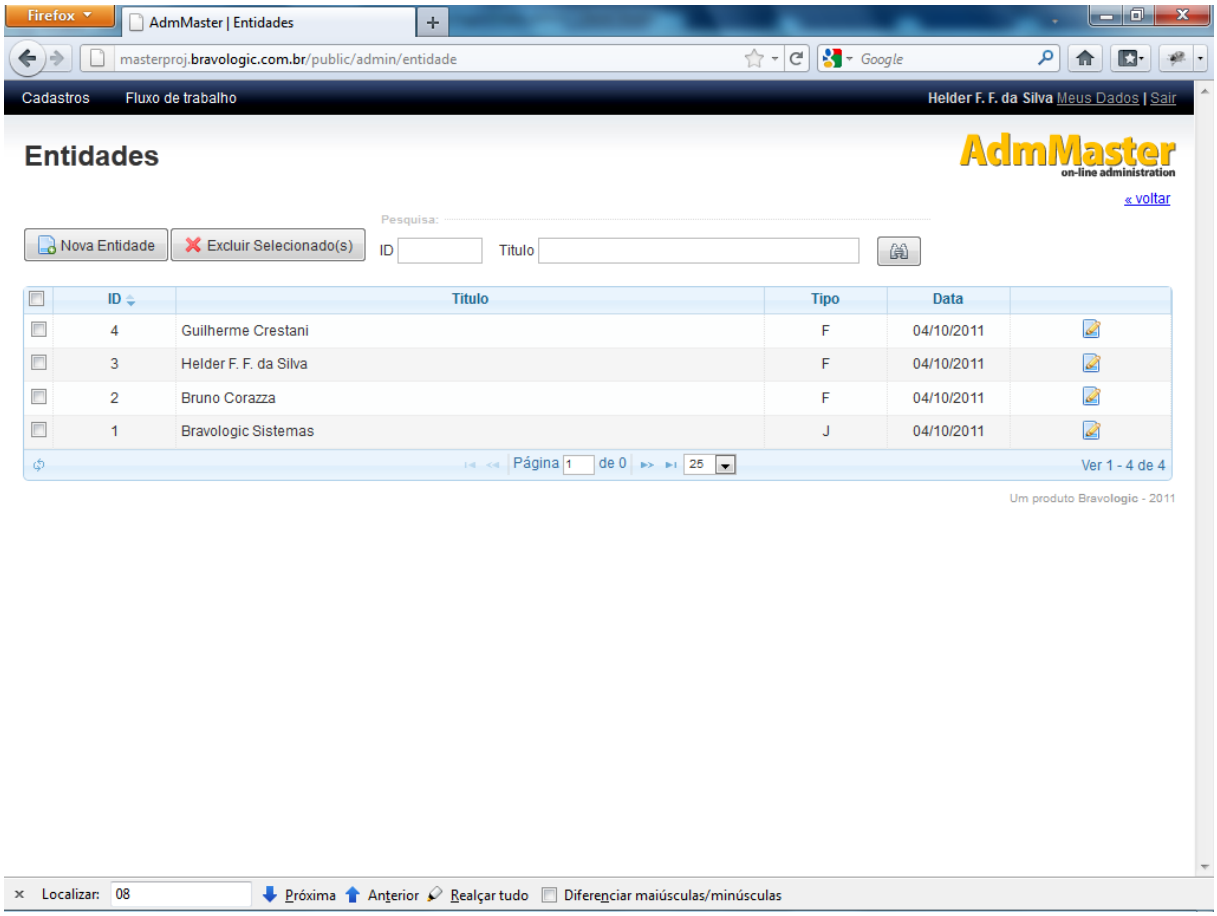

Figura 19- Utilizando o jQGrid na Listagem de Entidades

O *jQueryUI* ou *jQuery User Interface*, consiste em um conjunto de *plugins* que permite, interação e animação, juntamente com avançados efeitos, aplicação de temas, tudo isso baseando-se na biblioteca *jQuery*.

Dentre os diversos *plugins* disponibilizados, foram utilizados apenas o componentes *Tabs,* mostrado na **Figura 20,** que consiste em um menu em forma de abas.

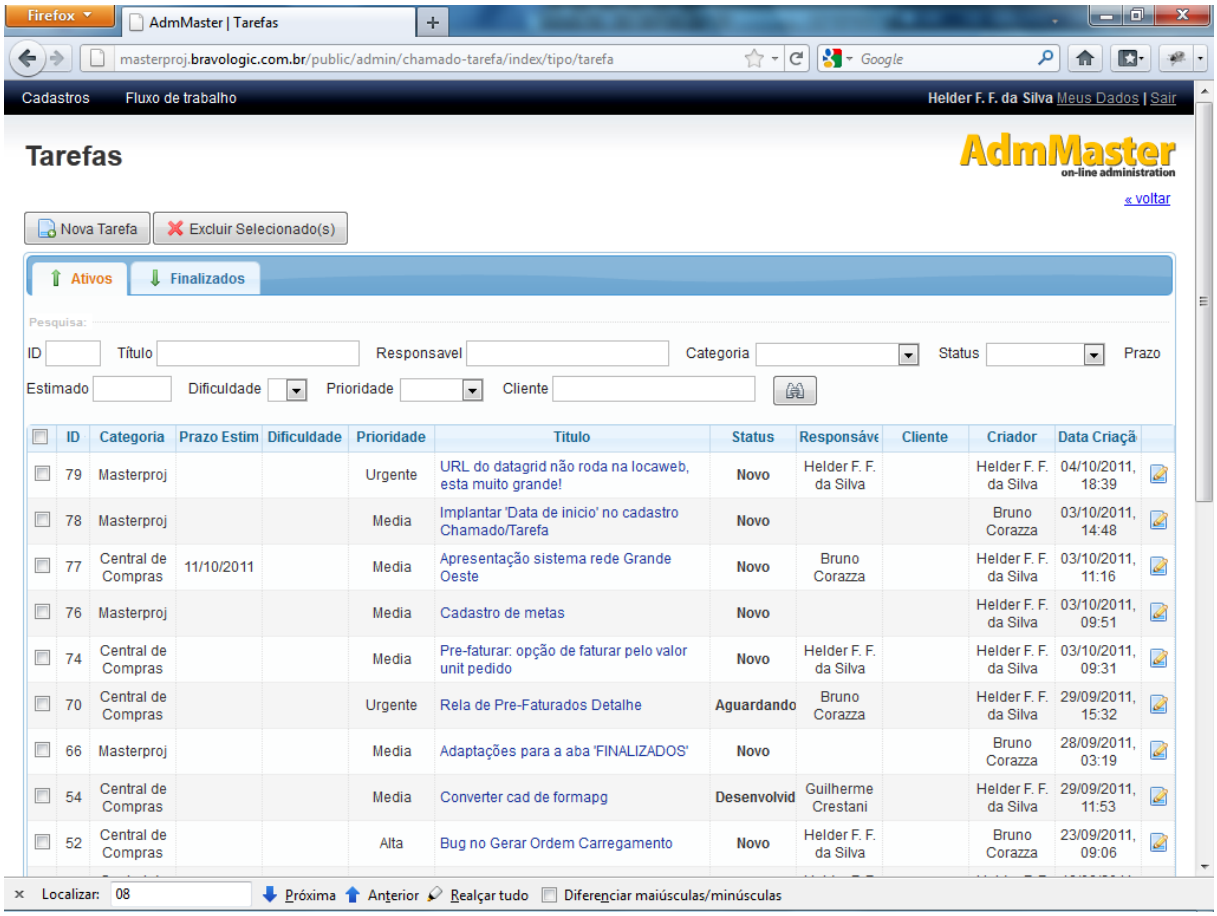

Figura 20- Utilizando o Tabs na Listagem de Tarefas

4.4 PERSPECTIVAS FUTURAS

Para esta versão inicial do software não foi possível implementar todas as idéias e durante o desenvolvimento do mesmo muitas outras foram surgindo. Dentre elas as principais são:

- Geração de relatórios de controle e acompanhamento de produtividade.
- Visualização de chamados / tarefas no estilo calendário.
- Visualização estilo Gráfico de Ganntt.
- Controle de versionamento de arquivos.
- Relatório de controle de acesso de usuários.

5. CONCLUSÃO

A área de sistemas de gestão de *workflow* vem recebendo bastante destaque atualmente. A modelagem de *workflow* representa as atividades que compõem processos, sua sequência de execução e seu relacionamento com as entidades envolvidas. É nesse modelo que um software gerenciador de *workflow* se baseia para coordenar a execução das atividades.

Existem inúmeros softwares disponíveis para gerenciamento de tarefas, chamado, tickets, inclusive on-line. Porém é difícil encontrar alguma ferramenta com tamanha simplicidade e objetividade quanto à apresentada.

O software ainda está em fase inicial, mas já se encontra em utilização desde agosto de 2011, aproximadamente 4 meses, em uma empresa real de desenvolvimento de software, apresentando ganho de produtividade e organização após a implantação.

REFERÊNCIAS BIBLIOGRÁFICAS

AALST, W.V.P. van der; HEE, V.K. **Workflow management: models, methods and systems**. Cambridge: MIT Pres, 2002.

CRUZ, Tadeu**. Workflow: a Tecnologia que Vai Revolucionar Processos**. 2ª ed. São Paulo: Atlas, 2000.

EVANS, C. **Guia para Programação com Framework ZEND**. Ciência Moderna, 2008.

DUARTE, Douglas. **MySQL**. Disponível em: http://www.ecnsoft.net/wp-content/plugins/downloads-manager/upload/FATEC-SBC_BDA1_MySQL.pdf, acesso em 12 de dezembro de 2011.

GONZAGA, Flávio S.; BIRCKAN Guilherme. **Curso de PHP e MySql**. Disponível em: http://pt.scribd.com/doc/38085726/Apostila-de-Curso-PHP-MySQL-Servidor-de-Banco-de-Dados, acesso em 28 de novembro de 2011.

KROHA, Petr. **Software technologie**. Prentice Hall, 1997.

LISBOA, F.G.S. **Zend Framework Desenvolvimento em PHP 5 orientado a objetos com MVC**. Novatec 2008.

NICOLAO, Mariano. **Caracterizando Sistemas de Workflow.** Disponível em: http://www.oocities.org/wallstreet/market/4702/textos/workflow.htm#administrativo, acesso em 12 de setembro de 2011.

NICOLAO, Mariano. **Modelagem de Workflow utilizando um Modelo de Dados Temporal Orientado a Objetos com Papéis. Dissertação de Mestrado**. UFRGS, Porto Alegre, Rio Grande do Sul, Brasil - 1998.

NICOLAO, Mariano. **Um Estudo sobre Conceituação de Workflow**. Porto Alegre: Instituto de Informática, Universidade Federal do Rio Grande do Sul, 1996. 44p. Monografia Mestrado em Ciência da Computação.

PLESUMS, Charles. **The Workflow Handbook 2002: Introduction to Workflow**. Florida: Future Strategies Inc., 2002. p. 19-38. Disponível em: http://www.plesums.com/image/introworkflow.html, acesso em: 30 de setembro de 2011.

WfMC. **The Workflow Management Coalition**. Disponível em: http://www.wfmc.org, acesso em 27 de setembro 2011.