UNIVERSIDADE DO PLANALTO CATARINENSE
CURSO DE SISTEMAS DE INFORMAÇÃO
(BACHARELADO)

LUIZ FELIPE PINHEIRO SUPPI

DESENVOLVIMENTO DE APLICAÇÕES UTILIZANDO O FRAMEWORK ZEND

LAGES (SC)

2012

LUIZ FELIPE PINHEIRO SUPPI

DESENVOLVIMENTO DE APLICAÇÕES UTILIZANDO O FRAMEWORK ZEND

Trabalho de Conclusão de Curso submetido à Universidade do Planalto Catarinense para obtenção dos créditos de disciplina com nome equivalente no curso de Sistemas de Informação - Bacharelado.

Orientação: Profª. Sabrina Bet Koerich, M.Sc.

LAGES (SC)

2012

LUIZ FELIPE PINHEIRO SUPPI

DESENVOLVIMENTO DE APLICAÇÕES UTILIZANDO O FRAMEWORK ZEND

ESTE RELATÓRIO, DO TRABALHO DE CONCLUSÃO DE CURSO, FOI JULGADO ADEQUADO PARA OBTENÇÃO DOS CRÉDITOS DA DISCIPLINA DE TRABALHO DE CONCLUSÃO DE CURSO, DO 8º. SEMESTRE, OBRIGATÓRIA PARA OBTENÇÃO DO TÍTULO DE:

BACHAREL EM SISTEMAS DE INFORMAÇÃO

Lages (SC), 14 de Setembro de 2012

Profª. Sabrina Bet Koerich, M.Sc.
Orientador

BANCA EXAMINADORA:

Prof. Sérgio Murilo Schütz, M.Sc.
UNIPLAC

Prof. Madalena Pereira da Silva, M.Sc.
UNIPLAC

Prof. Sabrina Bet Koerich, M.Sc
Coordenador de Curso / Professor de TCC

Dedico este trabalho aos meus pais, por sempre estarem do meu lado, não medindo esforços para que eu pudesse atingir meus objetivos.

Agradeço primeiramente a Deus por ter me dado forças para chegar até aqui, sem me deixar desistir de meus objetivos.

Agradeço aos meus pais, Luiz e Irene, por ser minha referência me mostrando o caminho do bem. Serei eternamente agradecido a tudo que fizeram e ainda fazem por mim.

Agradeço a minha irmã, Larissa, por ser minha amiga reforçando o aprendizado de nossos pais.

Agradeço a minha namorada, Luana, por compreender a importância desta etapa da minha vida aceitando por muitas vezes a minha ausência quando preciso.

Agradeço a minha orientadora Sabrina Bet Koerich, por todo o empenho, paciência e dedicação para que este objetivo fosse alcançado.

“Que os vossos esforços desafiem as impossibilidades, lembrai-vos de que grandes coisas do homem foram conquistadas do que parecia impossível.”
(Charles Chaplin)

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS E SIGLAS

BSD	- *Berkeley Software Distribution*
CPF	- Cadastro de Pessoa Física
IBM	- *International Business Machines*
MVC	- *Model, View and Controller*
NASA	- *National Aeronautics and Space Administration*
PDF	- *Portable Document Format*
PHP	- *Hypertext Preprocessor*
RAD	- *Rapid application development*
RG	- Registro Geral
RSS	- *Rich Site Summary*
SGE	- Sistema Gerenciador de Eventos
ZF	- *Zend Framework*
UNIPLAC	- Universidade do Planalto Catarinense
XHTML	- *Extensible HyperText Markup Language*
CSS	- *Cascading Style Sheets*
W3C	- *World Wide Web Consortium*
PDT	- *PHP Development Tools*
DB	- *Data Base*
AJAX	- *Asynchronous Javascript And XML*
JSON	- Javascript Object Notation
PROPEG	- Pró-Reitoria de Pós-Graduação, Pesquisa e Extensão
GT	- Grupo de Trabalho
HTML	- *Hypertext Markup Language*
SQL	- *Structured Query Language*
XML	- *Extensible Markup Language*
SGBD	- Sistema Gerenciador de Banco de Dados

RESUMO

Este Trabalho de Conclusão de Curso tem como objetivo explorar o *framework* Zend aplicado ao desenvolvimento de um estudo de caso. Para tal foi realizado um levantamento bibliográfico sobre os conceitos e principais características do *framework* buscando explorar e identificar seu potencial. Para o estudo de caso foi desenvolvido um sistema de gerenciamento de eventos (SGE) para o uso da Universidade do Planalto Catarinense (UNIPLAC), este tem como funcionalidades básicas o controle de inscrições de eventos, submissões e avaliações de trabalhos, bem como o controle de frequência. Para o desenvolvimento do SGE foi utilizado como ferramenta o Eclipse 3.6.0 PDT – Helios em conjunto com o *framework* Zend na sua versão 1.11.11. Os resultados obtidos mostram que com o uso do Eclipse juntamente com o Zend Framework facilitaram o trabalho de desenvolvimento. A análise feita sobre o *framework* mostra mais pontos positivos do que negativos, resultando num fácil aprendizado. No geral o *framework* mostrou-se completo e com vários componentes independentes que agilizam o processo de implementação.

Palavras-chave: Zend Framework; Desenvolvimento Web; Gerenciador de Eventos.

ABSTRACT

This work explore the Zend framework applied to the development of a case study. For this we conducted a literature review on the concepts and key features of the framework looking for explore and identify their potential. For the case study we developed an event management system (EMS) for use at the Universidade do Planalto Catarinense - Brazil (UNIPLAC), this has the basic functionality of the control event subscriptions, submissions and evaluations of work, as well as control frequency. For the development of SGE was used the Eclipse tool 3.6.0 PDT - Helios together with the framework Zend in the version 1.11.11. The results show that using the Eclipse with the Zend Framework facilitated the development work. The analysis of the framework shows more strengths than weaknesses, resulting in easy learning. Overall the framework showed up complete with several independent components that streamline the deployment process.

Keywords: Zend Framework, Web Development, Event Manager.

SUMÁRIO

1 INTRODUÇÃO

1.1 Apresentação

A Internet evoluiu muito nos últimos anos, principalmente no meio de desenvolvedores de *softwares*. Com o uso da Internet para fins comerciais, surgiu a necessidade por aplicações *Web* com recursos semelhantes aos de aplicações *desktop*.

Com o advento da *Web* 2.0, que possibilitou os usuários irem além da leitura de páginas interagindo com as aplicações, surgiram novas tecnologias de desenvolvimento para a Internet, como é o caso dos *frameworks*.

Segundo Jobstraibizer (2009), um *framework* é um conjunto de componentes que contém uma arquitetura e uma estrutura interna básica para o desenvolvimento de uma aplicação. Funciona como uma aplicação semipronta que deve ser estendida e personalizada para que um sistema desenvolvido funcione corretamente.

Dentre esses *frameworks*, está o *Zend Framework* (ZF), o qual é uma biblioteca PHP para desenvolvimento de aplicações *Web*. Os componentes se encaixam para oferecer um *framework* de pilha completa com todos os componentes necessários para construir aplicações modernas, fáceis de serem criadas e mantidas (ALLEN; LO; BROWM, 2009).

Dessa forma, a utilização do *Zend Framework* e seus componentes para o desenvolvimento de aplicações, torna o trabalho rápido e fácil de ser mantido, oferecendo ao usuário uma aplicação rica em conteúdo.

Neste trabalho apresenta-se um estudo de caso sobre o *Zend Framework*, como forma de demonstrar como o seu uso pode ser empregado em sistemas *Web*.

A partir do capítulo 2, são apresentados conceitos sobre *Zend Framework.* No capítulo 3 é definido um estudo de caso a ser desenvolvido com o *Framework Zend* e sua modelagem. O capítulo 4 demonstra a aplicação e sua implementação. Por fim no capítulo 5 são apresentadas as considerações finais.

1.2 Descrição do problema

O surgimento do paradigma da *Web* 2.0 tornou o processo de desenvolver e gerenciar conteúdos em sistemas *Web* um desafio para os desenvolvedores.

O cenário onde aplicações *Web* devem ser cada vez mais dinâmicas faz com que os desenvolvedores estejam cada vez mais alinhados às mudanças requisitadas por um mercado em constante atualização. Neste sentido a alta complexidade dos sistemas *Web* exigem ferramentas que agilizem o processo de desenvolvimento.

1.3 Justificativa

O desenvolvimento *Web* é a forma de programação que cresce cada vez mais entre os desenvolvedores. Muitos programadores usam o desenvolvimento *Web* por vários motivos, entre eles, o sistema pode ser acessado de qualquer lugar, não necessita instalação no computador do cliente apenas ter um navegador instalado, e a facilidade de manutenção do sistema, isso tudo na maioria das vezes não é possível com um sistema desktop.

> O rápido crescimento das aplicações WEB, tanto em seu escopo quanto na extensão de seu uso, tem afetado todos os aspectos de nossas vidas. (GINIGE e MURUGESAN1, 2001 *apud* CONTE, et al., 2005).

Para facilitar o desenvolvimento de sistemas *Web* existem os *frameworks*, mais especificamente, *frameworks* PHP. Uma das principais características de um *framework*, a que chama a atenção de desenvolvedores, é o reaproveitamento de

[1] Ginige, A., Murugesan, S. (2001), "Web Engineering: an Introduction", IEEE Multimedia, Vol. 8, Issue: 1, pp: 14 – 18.

códigos, evitando assim a programação desnecessária. Além do reaproveitamento de códigos, os *frameworks* também têm como papel diminuir o tempo de programação oferecendo componentes prontos, tornando o trabalho um pouco mais fácil.

Segundo Lisboa (2009), a decisão por usar um *framework* de desenvolvimento reside na necessidade de estruturar os projetos de *software*, devido à grande complexidade que os mesmos alcançaram. Um projeto estruturado reduz custos, aumenta a qualidade da aplicação e diminui o tempo de desenvolvimento.

Um *framework* que vem crescendo entre desenvolvedores PHP é o *Zend Framework*. O *Zend* introduz um conjunto padronizado de componentes que permite um fácil desenvolvimento de aplicações que podem ser desenvolvidas, mantidas e melhoradas facilmente. Além desses motivos ele é gratuito e software de código aberto. Talvez a principal vantagem do *Zend Framework* em relação aos outros frameworks PHP, é pela sua comunidade ser capitaneada pela *Zend Technologies*, empresa que mais investe em PHP, segundo Lisboa (2009).

Usando o *Framework Zend* espera-se desenvolver um sistema *web* onde o usuário consiga interagir facilmente com as informações.

1.4 Objetivo geral

Este trabalho tem como objetivo explorar o *framework Zend* aplicado no desenvolvimento de sistemas.

1.5 Objetivos específicos

a) Identificar características do *framework zend* disponíveis para o desenvolvimento Web.
b) Demonstrar características do *framework* zend através de um estudo de caso.

1.6 Metodologia

Para o desenvolvimento deste trabalho, inicialmente foi realizado o levantamento bibliográfico, abordando os principais conceitos discutidos, sendo eles: Desenvolvimento *Web* e *Zend Framework*. O levantamento bibliográfico teve como principais fontes de pesquisa a Internet e livros especializados. O estudo tem como objetivo entender os conceitos citados.

Na seqüência, foi desenvolvido um estudo aprofundado das características e funcionalidades do *Zend Framework*, o qual deu origem ao capítulo 2.

Em seguida, foi realizada análise de um estudo de caso proposto pela UNIPLAC. Este estudo foi documentado através da modelagem da aplicação, onde é composta por levantamento de requisitos, casos de uso e diagramas de classes. Também é modelado um banco de dados do estudo de caso. Esta etapa é apresentada no capítulo 3.

Posteriormente, o estudo de caso foi implementado com o uso do *Framework Zend 1.11.11*, utilizando como ferramentas o *Eclipse 3.6.0 PDT – Helios* e banco de dados MySql 5.5.8. Os principais pontos da implementação são descritos no capítulo 4. Por fim são apresentadas as considerações finais no capítulo 5.

2 ZEND FRAMEWORK

Neste capítulo são apresentados resultados de um estudo do *Zend Framework*, bem como suas características, arquitetura, componentes e apresentando alguns trabalhos desenvolvidos com o *framework.*

2.1 Framework para desenvolvimento web com PHP

Frameworks têm como principal objetivo o reaproveitamento de códigos, com isso torna o desenvolvimento simplificado e fácil, com uma coleção de códigos-fonte, classes, funções e metodologias. Segundo Minetto (2007) um *framework* de desenvolvimento é uma "base" de onde se pode desenvolver algo maior ou mais específico.

Para o desenvolvimento *web* se tem os *frameworks* para a linguagem PHP, dentre os quais se pode citar o CakePHP (CAKE, 2012), o CodeIgniter (CODEIGNITER, 2012), o Symfony (SYMFONY, 2012) e o Zend *Framework* (ZEND, 2012).

O quadro 1 apresenta um estudo comparativo, disponibilizado pelo site PHP Frameworks (2007), no qual pode-se observar que o *framework* Zend atende todas as características em análise, possibilitando se ter uma idéia da sua completude. A estrutura MVC mostra quais *frameworks* tem suporte a esse tipo de arquitetura, a qual será explanada posteriormente. O *Multiple DB* mostra que a estrutura do framework suporta múltiplos bancos de dados. *Templates* mostra quais frameworks tem embutido em sua estrutura a opção de *layout* para as interfaces web. *Cache* indica se o framework fornece um modo de armazenar informações em modo *cache*, ou seja, de

forma temporária. Validação indica se o framework contém componentes de validação e filtros para formulários. *Ajax* indica que o framework vem com suporte para esse tipo de linguagem dinâmica. O *Auth* Módulo refere se constam componentes de autenticação de usuários. E a coluna módulos indica que o framework permite a instalação e utilização de módulos extras, tais como como analisador de RSS, módulo gerador de PDF, entre outros.

QUADRO 1 - Comparativo de frameworks PHP

PHP Framework	MVC	Multiple DB	Templates	Cache	Validação	Ajax	Auth	Módulos
Akelos	X	X	X	X	X	X	X	X
ash.MVC	X	-	X	-	X	-	X	X
CakePHP	X	X	-	X	X	X	X	X
CodeIgniter	X	X	X	X	X	-	-	-
DIY	X	-	X	X	-	X	-	-
Componentes eZ	-	X	X	X	X	-	-	-
Fusebox	X	X	-	X	-	X	-	X
PHP em TRAX	X	X	-	-	X	X	-	X
PHPDevShell	X	-	X	X	X	X	X	X
PhpOpenbiz	X	X	X	-	X	X	X	-
Prado	X	X	X	X	X	X	X	X
QPHP	X	X	X	-	X	X	X	X
Gaivota	X	X	X	X	X	X	X	X
Symfony	X	X	-	X	X	X	X	X
Wact	X	X	X	-	X	-	-	X
WASP	X	-	X	-	X	X	X	X
Yii	X	X	X	X	X	X	X	X
Zend	X	X	X	X	X	X	X	X
Zoop	X	X	X	X	X	X	X	-

(Fonte: Php Frameworks, 2007)

2.2 Características

No final de 2005 a *Zend Technologies* deu início ao *Zend Framework* como parte do projeto PHP *Collaboration* com fins de estimular o uso do PHP. *Zend Framework* (também referido como ZF) foi criado para ajudar a garantir que a criação de *websites* baseados em PHP fosse fácil e pudesse ser mantido em longo prazo (ZEND, 2012).

ZF é um *framework* para desenvolvimento *Web* orientado a objetos, livre e *open source*, quer dizer que o código-fonte está disponível. Sua licença é compatível com

BSD *License*, que permite que aplicações criadas com o framework possam ter código-fonte proprietário.

ZF é uma biblioteca de componentes fracamente acopladas onde podem ser usados de maneira independente, sem a obrigação de usar tudo o que o *framework* oferece. Segundo Lisboa (2010) a novidade que além da opção de usar seus componentes separadamente, o ZF também oferece uma implementação avançada do padrão Modelo-Visão-Controlador (MVC).

O ZF possui quatro principais características, descritas nas sessões seguintes.

2.2.1 Extrema simplicidade e produtividade

O ZF foi projetado para ser simples. O ZF manteve o espírito de simplicidade da programação PHP segundo Lisboa (2010), isso implica numa curva de aprendizagem baixa.

Usar um *framework* tem a ideia de reduzir custos de treinamento, agilizando a colocação no mercado. Desenvolvendo dessa forma faz com que se escolha peças necessárias para suas aplicações, por sua fácil localização de códigos reduzirá custos de manutenção.

2.2.2 Últimas características de desenvolvimento Web

O ZF possui as últimas características para o desenvolvimento *Web*, assim como suporte à AJAX utilizando JSON, uma edição PHP nativa do mecanismo de busca Lucene[2], formato de dados e acesso fácil à sindicalização, o ZF pretende ser o principal local de consumo e publicação de *web service* e biblioteca de classes PHP 5 orientada a objetos e com alta qualidade.

Com essas características constata-se que seja possível desenvolver aplicações usando o que tem de mais recente em desenvolvimento *Web*.

[2] Lucene é uma biblioteca para indexação e consulta de textos em código aberto (PAMPLONA, 2012).

2.2.3 Licença segura e confiável (para empresas)

Como já foi mencionado anteriormente o ZF usa uma licença baseada em BSD *License*, que permite o código-fonte de aplicações desenvolvidas com ZF possam ser proprietários. Isso é importante para as empresas que nem sempre podem liberar o código-fonte da aplicação.

Essa é uma forma de proteger a propriedade intelectual, ou valor agregado, construído sobre o ZF, segundo Lisboa (2009).

2.2.4 Completamente testado – fácil e seguramente extensível

O ZF foi testado desde o começo com requisitos de cobertura de código para garantir que todo código contribuído tenha sido testado, de forma que possa se manter estável e fácil de estender.

2.3 Arquitetura

O ZF é composto por vários componentes e podem ser agrupados em cinco categorias: MVC, desenvolvimento rápido de aplicações, banco de dados, internacionalização e localização e autenticação, autorização e gerenciamento de sessão.

Nas sessões a seguir são mostradas cada uma das cinco categorias juntamente com seus componentes.

2.3.1 Modelo-Visão-Controlador (MVC)

O MVC é um padrão de projeto orientado a objetos que tem como principal objetivo separar o projeto em camadas. A separação em camadas serve para verificar e detectar problemas de forma menos incômoda. O padrão MVC é de alto nível, segundo Lisboa (2009) “ele dá uma ideia da arquitetura geral de uma aplicação”.

O MVC possui três camadas, cada uma com as seguintes características:

- Modelo (*model*): representa informações sobre o domínio da aplicação, não é visual e contém "todos os dados e comportamentos que não os usados pela interface do usuário", segundo Lisboa (2009). Resumidamente é onde contém o código da camada de dados.
- Visão (*view*): é o *design* da aplicação, onde tudo que será mostrado ao usuário é construído.
- Controlador (*controller*): são classes que conectam modelos e suas visões. O controlador manipula as requisições e produz os resultados para os usuários baseado na lógica de negócio, diz Lisboa (2009).

Segundo Lotar (2011), a Figura 1 representa a relação entre o controlador, o modelo e a visão. O controlador aceita a requisição do usuário, o qual se necessário se comunica com a camada modelo que realiza a interface com o banco de dados, fazendo então a devolução das informações para o controlador. Por fim, o controlador envia as informações necessárias para a visão a qual apresenta ao usuário.

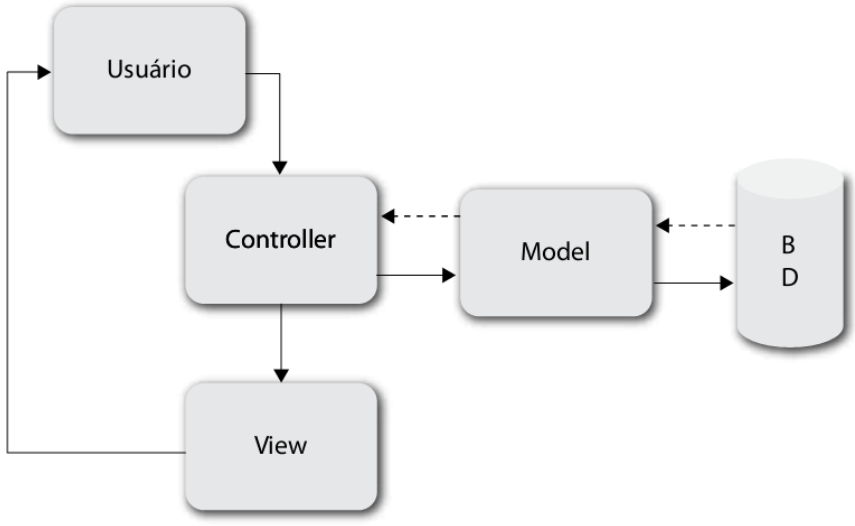

FIGURA 1 - Relação entre o usuário, o controlador, o modelo e a visão.
(Fonte: Lotar, 2011)

O quadro 2 apresenta componentes usados no padrão MVC, para o *framework zend.*

QUADRO 2 - Componentes de MVC

Componente	**Descrição**
Zend_Application	Fornece uma facilidade, o *bootstraping* para aplicações com a finalidade de fornecer recursos reutilizáveis. Também toma conta das configurações do ambiente PHP e introduz por padrão o autocarregamento.
Zend_Application_Bootstrap	Classe base para o uso das classes de *Bootstrap*.
Zend_Application_Module	Classes para ser trabalhadas com módulos.
Zend_Application_Resourse	Contém conjunto de classes *plugins*, para ser utilizado em aplicações ZF.
Zend_Controller_Front	Processa todas as requisições recebidas pelo servidor, então roteia e despacha para os controladores de ação apropriados.
Zend_Controller_Action	É uma classe abstrata que implementa controladores de página dentro do padrão MVC.
Zend_Controller_Dispatcher	Toma lugar em um laço de repetição. O objeto de requisição indica multiplas ações ou o Controlador reinicia o objeto de requisição, forçando ações adicionais.
Zend_Controller_Plugin	Controlador de sistema de plugin que permite que o código do usuário possa ser chamado em determinados eventos que ocorrem no processo de vida do controlador.
Zend_Controller_Router	Toma lugar uma vez, usando dados no objeto de requisição quando `dispatch()` é chamado.
Zend_Form	Simplifica a criação e manipulação de formulários na aplicação *web*, com algumas características como filtragem e validação da entrada de elementos, ordenação de elementos.
Zend_Layout	Implementa o padrão de projeto Visão, permitindo que desenvolvedores empacotem conteúdo de aplicação dentro de outra visão, geralmente o *template* do site.
Zend_View	É uma classe para trabalhar com a lógica de exibição do padrão de projeto MVC.
Zend_View_Helper	São classes auxiliares para executar determidas funções, como por exemplo, a formatação de uma data.

(Fonte: adaptado de Lisboa, 2009 e Zend, 2009)

2.3.2 Desenvolvimento rápido de aplicações (RAD)

Uma das vantagens dos *frameworks* é justamente o RAD, por proporcionar uma agilidade no desenvolvimento das aplicações. Segundo Jobstraibizer (2009), "assim os códigos tornam-se menores, e podem ser amplamente reaproveitados em vários outros sistemas que utilizem o mesmo *framework"*.

O ZF usa o RAD para acelerar a configuração inicial da aplicação. Segundo Lisboa (2010), o `zend_tool` fornece uma ferramenta de suporte e um cliente linha de comando capaz de gerar a estrutura do projeto, como artefatos MVC e mais.

O quadro 3 relata as ferramentas usadas pela `Zend_Tool`.

QUADRO 3 - Ferramentas Zend_Tool

Ferramentas	**Descrição**
Zend_CodeGenerator	Oferece facilidades para gerar um código arbitrário usando uma interface orientada a objetos, tanto para criar um novo código como para atualizar um existente.

Zend_Reflection	É uma extensão para a *Reflection* API do PHP, oferencendo alguns métodos adicionais para recuperação de artefatos não definidos na *Reflection* API.
Zend_Tool_Framework	É uma estrutura para expor funcionalidades comuns, como criação de estrutura de projetos, geração de códigos e muito mais.
Zend_Tool_Project	Amplia as capacidades do Zend_Tool_Framework na gestão de um projeto. Controla recursos de projetos como diretórios, arquivos, base de dados e mais.

(Fonte: adaptado de Lisboa, 2009 e Zend, 2009)

2.3.3 Banco de dados

O ZF faz o uso de práticas de programação de banco de dados, por meio de padrões arquiteturais de fontes de dados. Fornece adaptadores de dados para os gerenciadores de dados para que possa construir os modelos das aplicações, Segundo Lisboa (2010).

O quadro 4 mostra componentes de banco de dados que o ZF faz uso.

QUADRO 4 - Componentes de banco de dados

Componentes	**Descrição**
Zend_Db	Com suas classes relacionadas fornecem uma interface de banco de dados SQL simples para ZF.
Zend_Db_Adapter	É a classe responsável pela conexão da aplicação com o banco de dados, usando uma extensão PHP do banco de dados utilizado.
Zend_Db_Profiler	Permite examinar as consultas da aplicação com uma variedade de métodos, como retornar o número total de consultas executadas e mais.
Zend_Db_Select	Representa uma declaração de consulta SQL SELECT. Esta classe tem métodos para adicionar partes individuais à consulta.
Zend_Db_Table	Esta classe é uma interface orientada a objetos para tabelas de banco de dados. Possui métodos para operações comuns sobre tabelas.

(Fonte: adaptado de Lisboa, 2009 e Zend, 2009)

2.3.4 Internacionalização e localização

A internacionalização e localização permitem que suas aplicações tenham suporte a outros idiomas, além do que foi desenvolvido. O ZF vem com componentes capazes de realizar tais funcionalidades, esses componentes podem ser visualizados como apresenta o quadro 5.

QUADRO 5 - Componentes de internacionalização e localização

Componentes	**Descrição**
Zend_Currency	Manipula questões relacionadas à moeda, representação de dinheiro e formatação. Provê métodos informativos que incluem informações localizadas de moedas.
Zend_Date	Oferece uma API detalhada para manipulação de datas e horas. Suporta nomes abreviados de meses em vários idiomas.

Zend_Locale	Provê a classe e suas subclasses para acesso a informação sobre o idioma esperadas pelo usuário. Geralmente automaticamente seleciona a localização correta da informação fornecida pelo navegador do usuário.
Zend_Measure	Fornecem uma forma fácil para trabalhar com medidas. Pode converter em diferentes unidades do mesmo tipo.
Zend_Translate	É a solução ZF para aplicações multilíngues, onde o conteúdo deve ser traduzido para diversos idiomas e exibido dependendo do usuário.

(Fonte: adaptado de Lisboa, 2009 e Zend, 2009)

2.3.5 Autenticação, autorização e gerenciamento de sessão

Em aplicações *web* existem dados que nem sempre todos os usuários tem a permissão de manipular, isso é customizado de forma protegida para que o acesso ocorra somente aos usuários autorizados. Com o ZF é possível desenvolver esse tipo de controle através dos componentes apresentados no quadro 6.

QUADRO 6 - Componentes de autenticação, autorização e gerenciamento de sessão

Componentes	**Descrição**
Zend_Acl	Controla como objetos de requisição tem acesso garantido a objetos protegidos. Está envolvido somente com o acesso de autorização.
Zend_Auth	Fornece uma API para autenticação e inclui adaptadores para os cenários de uso mais comuns.
Zend_Session	Ajuda a administrar e preservar dados de sessão ao longo de múltiplas requisições de página pelo mesmo cliente.

(Fonte: adaptado de Lisboa, 2009 e Zend, 2009)

2.4 Trabalhos correlatos

Como o ZF vem crescendo, principalmente no desenvolvimento com a linguagem PHP, muitas empresas estão optando por usar esse framework. Dentre elas pode-se citar: IBM, Nokia, Magento, Microsoft, NASA, Apple Inc., Samsung entre outras (WIKIPEDIA, 2012). A seguir alguns exemplos de uso de ZF.

2.4.1 Ambiente educacional Web

No dia 7 de abril de 2010 foi lançado o Ambiente Educacional Web, da Secretaria de Educação do Estado da Bahia, o qual possui diversas funcionalidades educacionais para os alunos, professores e servidores.

O ambiente possui quatro módulos como, uma rede social, um repositório de

conteúdos digitais, um repositório de ambientes virtuais de aprendizagem e um repositório de ambientes de produção e colaboração (POTAPCZUK, 2010).

O sistema foi desenvolvido na linguagem PHP e utiliza o banco de dados *PostgreSQL*, foram também utilizados os *Zend Framework*, *Doctrine* e *jQuery*. Pode-se ter acesso ao ambiente acessando o site da Secretaria de Educação da Bahia (BAHIA, 2012).

A figura 2 mostra a página inicial do Ambiente Educacional *Web* da secretaria do estado da Bahia.

FIGURA 2 - Ambiente educacional web
(Fonte: BAHIA, 2012)

2.4.2 Magento

O Magento é um sistema *e-commerce open source* desenvolvido pela empresa Magento (MAGENTONET, 2012), tem sua estrutura baseada em módulos e foi desenvolvido com base no *Zend Framework.*

Entre as vantagens de usar o Magento estão às facilidades na implementação de novas funções através da criação de módulos, grande quantidade de extensões para download, versão *community* gratuita, serve para pequenos e grandes projetos, sistema seguro entre outras.

Existem algumas desvantagens como, por possuir muitas funções se torna um pouco pesado, a curva de aprendizagem para iniciantes é mais longa e necessita de algum conhecimento técnico para funcionar normalmente.

Está se tornando cada vez mais popular, sendo utilizado como base por grandes empresas, por exemplo, FOX, Lenovo, Goodyear, Olympus entre outras.

A figura 3 mostra o painel de administração do Magento *e-commerce*.

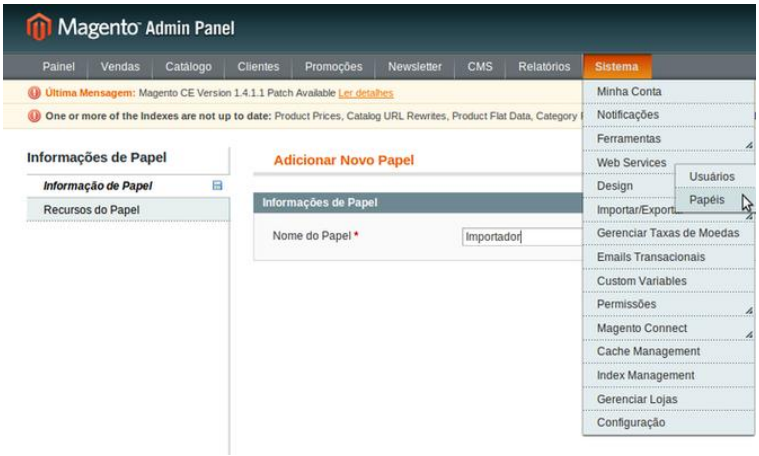

FIGURA 3 - Magento *e-commerce*
(Fonte: MAGENTONET, 2012)

2.5. Conclusão

O presente capítulo abordou conceitos sobre o *Zend Framework*, mostrando um comparativo com outros *frameworks* PHP, bem como detalhes de suas características, de sua arquitetura, de seus principais componentes e mostrando alguns trabalhos já realizados baseados no *framework*. O capítulo mostra-se de grande importância para as próximas etapas do trabalho.

Sobre o comparativo de *frameworks* PHP, observou-se que o ZF tem uma estrutura completa comparada a outros *frameworks*, contendo todos os requisitos abordados no quadro 1.

No que diz a respeito às características o *framework* mostra-se estar focado na construção de aplicações seguras, confiáveis e modernas.

A arquitetura mostra-se completa, apresentando categorias essenciais para agilizar o desenvolvimento das aplicações.

Ressalta-se ainda a exposição de algumas aplicações desenvolvidas utilizando ZF, a qual possibilita a utilização de recursos multimídia, possuem gerenciadores de conteúdo, controle de acesso e níveis de usuários, ou seja, percebe-se que permite o desenvolvimento de uma aplicação *Web* robusta.

3 ANÁLISE E PROJETO DO SISTEMA DE EVENTOS

Neste capítulo é apresentado o projeto e análise do sistema de eventos, trazendo o levantamento de requisitos e os casos de uso. Este projeto de sistema teve como base o antigo sistema de eventos – SGE, da UNIPLAC.

3.1 Sumário executivo

A Universidade do Planalto Catarinense, através da Pró-Reitoria de Pesquisa, Extensão e Pós-Graduação (PROPEPG) solicitou o desenvolvimento de um sistema gerenciador de eventos, que possibilite sua utilização na Mostra Científica e outros eventos da UNIPLAC, desta forma, deve possibilitar parametrização de forma a deixar o sistema adaptável. O sistema deve possibilitar o cadastro de um evento, informando dados básicos como nome e data. Alguns eventos permitirão a submissão de trabalhos devendo ser informadas as datas para submissão dos mesmos. Essas submissões podem ser organizadas por grupos de trabalhos (GT). O sistema ainda deve permitir o cadastro de ouvintes no evento. Alguns eventos possuem uma programação única, e outros podem possuir GT ou Oficinas a serem escolhidas pelo participante. Alguns eventos são cobrados taxa de inscrição, devendo este ser cobrado via boleto bancário. Ao final do evento deve ser possível gerar relatório de presença dos ouvintes, informando a porcentagem de participação. Também deve possuir relatório para controle de trabalhos inscritos e ouvintes no evento. O relatório dos inscritos deve ser organizado por evento, por oficina, também possibilitando a consulta de inscrições pagas e não pagas. O sistema deve ser disponibilizado através da Internet e ainda deve manter um histórico de informações dos eventos passados. O sistema deve estar

acessível em diferentes tipos de navegadores.

3.2 Levantamento de requisitos

O quadro 7 apresenta informações sobre o requisito funcional cadastrar evento.

QUADRO 7 - Requisito funcional cadastrar evento

RF1 – CADASTRAR EVENTO	
Descrição	O administrador do sistema deve cadastrar informações relativas a cada evento da Instituição, permitindo a inclusão, alteração, exclusão e consulta de informações.
Usuários	Administrador do Site e Gerente de Conferência.
Informação de entrada	Nome do evento, data de início e término, descrição do evento, *link* para o *hotsite* do evento, gerente de conferência, valor do evento, data de início para inscrições e data de término das inscrições.
Informação de saída	Relatório de eventos contendo o nome do evento, data de início e término, *hotsite* e número de inscritos no evento. Deverá conter como parâmetros de entrada o período (data de início e fim) para consulta.
Restrições lógicas	A data do evento deve ser posterior à data do cadastro. A data de término deve ser maior ou igual à data de início.
Restrições tecnológicas	O gerente de conferência dever ser um usuário cadastrado no sistema. O gerente de conferência poderá visualizar apenas as informações dos eventos ao qual ele está relacionado.

A seguir no quadro 8 são apresentadas informações sobre o requisito funcional cadastrar usuários.

QUADRO 8 - Requisito funcional cadastrar usuários

RF2 – CADASTRAR USUÁRIOS	
Descrição	O administrador do sistema ou qualquer usuário visitante pode realizar a inclusão, alteração, exclusão e consulta de um usuário no sistema.
Usuários	Administrador do *Site*, usuário do *site*.
Informação de entrada	Nome do usuário, CPF, RG, e-mail, senha, nível de acesso, data do último acesso, tipo usuário, área de atuação.
Informação de saída	Relatório de usuários informando o nível de acesso e data do último acesso. Relatório de submissões de um usuário contendo o nome do evento, data de submissão e situação da submissão (enviado, aceito, negado). Relatório de eventos inscritos por usuário, contendo o nome do evento e data do evento.
Restrições lógicas	O usuário será identificado no sistema pelo seu CPF, devendo o mesmo ser único. O e-mail deverá ser único, não podendo ser atribuído para mais de um usuário no sistema. O usuário pode assumir 5 tipos, simultaneamente: administrador, gerente de conferencia, ouvinte, autor e avaliador. Se o usuário for do tipo avaliador deve ser cadastrada a área de atuação.
Restrições tecnológicas	O sistema deverá atualizar a data do último acesso sempre que o usuário logar no sistema. O usuário de nível de acesso administrador e gerente de conferência podem ser

	cadastrados apenas por usuário de nível de acesso Administrador. Para o CPF deve ser permitido ao usuário informar somente os caracteres numéricos, devendo o sistema formatar o campo conforme a máscara ###.###.###-##. O nível de acesso para ouvinte é automaticamente inserido pelo sistema na inclusão do registro. Apenas o administrador do sistema terá acesso a todos os usuários, sendo que os demais níveis de acesso tem permissão apenas sobre o seu próprio registro.

No quadro 9 são apresentadas informações sobre o requisito funcional gerenciar programação.

QUADRO 9 - Requisito funcional gerenciar programação

RF3 – GERENCIAR PROGRAMAÇÃO	
Descrição	O gerente de conferência pode realizar a inclusão, alteração, exclusão e consulta de um item de programação do evento.
Usuários	Gerente de conferência.
Informação de entrada	Nome do item da programação, Descrição do item, Nome Resumido, Data, Horário, Local, Valor, Número máximo de participantes, aceite de *paper* para o grupo/trilha, data de início para submissão e data de término para submissão.
Informação de saída	Relatório dos itens de evento organizados por data (nome resumido, data, horário, local).
Restrições lógicas	A data de início e data de término para submissão só deverá ser informada se o evento aceitar submissão de trabalhos.
Restrições tecnológicas	O sistema deve permitir ao usuário informar somente valores válidos para data e hora. A programação faz parte de um evento.

O quadro 10 apresenta informações sobre o requisito funcional submissão de trabalho.

QUADRO 10 - Requisito funcional submissão de trabalho

RF4 – SUBMISSÃO DE TRABALHO	
Descrição	Os usuários do sistema poderão submeter trabalhos para os eventos, ou seja, inscrever seus artigos para serem avaliados pela comissão científica do evento.
Usuários	Autor.
Informação de entrada	Título do trabalho, Resumo, *Abstract*, anexo, evento, item de programação, *status* de aprovação, justificativa, número da submissão.
Informação de saída	Relatório de submissões contendo o nome dos autores, título do trabalho, item de programação e status de aprovação.
Restrições lógicas	O sistema só poderá aceitar a submissão de trabalhos dentro do prazo de submissão. O *status* de aprovação só pode aceitar os valores: em avaliação, aprovado e reprovado. Quando uma submissão é reprovada é obrigatório o preenchimento da justificativa.
Restrições tecnológicas	O sistema deverá gerar um número de submissão sequencial. Ao enviar uma nova submissão o *status* de aprovação será automaticamente iniciado como em avaliação. A opção de submissão apenas deverá ser disponibilizada quando o evento aceitar *paper*. Os relatórios de submissões aprovadas e não aprovadas somente podem ser acessados pelo autor ou gerente de conferência. O anexo deve possuir no máximo 5Mb.

	O título deve possuir no máximo 255 caracteres. O resumo deve permitir no máximo 1000 caracteres. A programação faz parte de um evento.

A seguir no quadro 11 são apresentadas informações sobre o requisito funcional inscrição de ouvintes.

QUADRO 11 - Requisito funcional inscrição de ouvintes

RF5 – INSCRIÇÃO DE OUVINTES	
Descrição	Os usuários do sistema poderão se inscrever em um evento.
Usuários	Ouvinte.
Informação de entrada	Usuário, evento e itens de programação, situação de pagamento, número de inscrição.
Informação de saída	Relatório de inscritos contendo nome do usuário, situação do pagamento. Como entrada, deve-se permitir a seleção conforme a situação de pagamento. Relatório de inscritos contendo nome do usuário, RG e Item de programação. Relatório de inscritos com linha para assinatura: nº inscrição, nome ouvinte, RG. Como entrada deve-se permitir a seleção conforme a situação de pagamento e o item de programação.
Restrições lógicas	O usuário poderá selecionar mais de um item de programação. O usuário não pode selecionar mais de um item no mesmo dia/horário. Os valores possíveis para "situação de pagamento" são: isento - pago - em aberto O sistema somente deverá permitir a inscrição caso o evento esteja com a data de início e término de inscrições vigentes.
Restrições tecnológicas	O sistema deverá gerar um número de inscrição sequencial. O usuário deverá se identificar no sistema através de seu CPF e senha. O usuário deverá selecionar um evento de uma lista de eventos. O usuário deverá selecionar um item de programação de uma lista.

O quadro 12 apresenta informações sobre o requisito funcional manter registro de frequência.

QUADRO 12 - Requisito funcional manter registro de frequência

RF6 – MANTER REGISTRO DE FREQUÊNCIA	
Descrição	O Gerente de Conferência deverá manter o registro de frequência dos ouvintes no evento.
Usuários	Gerente de Conferência.
Informação de entrada	Número da Inscrição no evento, item de programação, presença.
Informação de saída	Relatório contendo a frequência dos ouvintes no evento: nº inscrição, nome do usuário, percentual de presença, itens de programação.
Restrições lógicas	A informação "presença" permite os valores: sim ou não.
Restrições tecnológicas	A frequência poderá ser registrada por número de inscrição ou por item de programação. A presença só pode ser registrada em um item de programação na qual o ouvinte está inscrito.

O quadro 13 apresenta informações sobre o requisito funcional avaliar submissões.

QUADRO 13 - Requisito funcional manter avaliar submissões

RF7 – AVALIAR SUBMISSÕES	
Descrição	O Gerente de Conferência deverá registrar se uma submissão foi aprovada ou não.
Usuários	Gerente de Conferência.
Informação de entrada	Evento, Número da submissão, *status* de aprovação.
Informação de saída	Relatório de submissões por evento contendo a o número da submissão, nome do autor e o *status* de aprovação. Deve ser permitido solicitar as submissões conforme seu *status*.
Restrições lógicas	O status de aprovação só pode aceitar os valores: em avaliação, aprovado e reprovado.
Restrições tecnológicas	Não foram identificadas restrições tecnológicas.

No quadro 14 é apresentado o requisito de emissão de certificados.

QUADRO 14 - Requisito funcional emissão de certificados

RF6 – EMISSÃO DE CERTIFICADOS	
Descrição	O os ouvintes terão acesso ao certificado perante a sua frequência durante o evento.
Usuários	Ouvinte.
Informação de entrada	Nome do evento, nome do ouvinte, CPF ouvinte, presença.
Informação de saída	Certificado contendo o nome do evento, o nome e CPF do ouvinte e seu percentual de presença.
Restrições lógicas	A informação "presença" permite os valores: sim ou não.
Restrições tecnológicas	A emissão do certificado só poderá ser realizada caso o ouvinte obtenha um percentual de presença no evento.

No quadro 15 são apresentados os requisitos suplementares do sistema de eventos.

QUADRO 15 - Requisitos suplementares

S1. Utilizar interface Web.
S2. Armazenamento de informações em Banco de Dados.
S3. Acessibilidade aos navegadores (IE, Chrome, Firefox, Opera).
S4. Controle de Acesso:
1. Administrador Site
2. Gerente de conferência
3. Autor
4. Ouvinte
5. Avaliador

S5. Manuais deverão ser produzidos visando melhorar a compreensão e usabilidade do sistema.

3.3 Casos de uso

O quadro 16 apresenta a identificação dos casos de uso do sistema gerenciador de eventos.

QUADRO 16 - Identificação de casos de uso

UC Processos de Negócio
UC1. Realizar gerenciamento de artigos/*papers*
Requisitos vinculados: F1, F2, F3, F4, F8.
UC2. Realizar gerenciamento de inscrições em eventos
Requisitos vinculados: F1, F2, F3, F5, F6, F7, F9.
UC3. Gerenciamento de avaliação de artigos
Requisitos vinculados: F1, F2, F5.

UC CRUD
UC4. Gerenciar Evento
Requisitos vinculados: F1, F2
UC5. Gerenciar Usuário
Requisitos vinculados: F2
UC6. Gerenciar Programação
Requisitos vinculados: F1, F3
UC Relatórios
UC8. Relatório de eventos por período
Requisitos vinculados: F1, seção informações de saída
UC9. Relatório de acesso de usuários.
Requisitos vinculados: F2, seção informações de saída
UC10. Relatório de programação do evento
Requisitos vinculados: F1 e F3, seção informações de saída
UC11. Relatório de submissões
Requisitos vinculados: F1, F2, F4 e F8, seção informações de saída
UC10. Relatório de inscritos por situação
Requisitos vinculados: F1, F5, seção informações de saída
UC11. Relatório de inscritos por item de programação
Requisitos vinculados: F1, F2, F3, F5, seção informações de saída e F9.
UC12. Relatório de inscritos por evento
Requisitos vinculados: F1, F2, F3 e F5, seção informações de saída e F9.
UC13. Relatório de inscritos por oficina
Requisitos vinculados: F1, F2, F3 e F5, seção informações de saída e F9.
UC14. Relatório de inscritos por status do pagamento
Requisitos vinculados: F1, F2, F3 e F5, seção informações de saída e F9.
UC15. Boleto Bancário
Requisitos vinculados: F1, F2, F5 e F6, seção informações de saída
UC16. Relatório de frequência dos ouvintes (com porcentagem de participação)
Requisitos vinculados: F1, F2, F3, F5 e F7, sessão informações de saída

Nos quadros 17 a 19 é realizada a expansão dos casos de uso de processo de negócio, detalhando o funcionamento do sistema gerenciador de eventos.

QUADRO 17 - Caso de uso gerenciar artigos

UC1 - Caso de uso: gerenciar artigos
[IN] O autor informa a sua identificação.
[OUT] O sistema informa os eventos disponíveis para submissão.
[IN] O autor escolhe o evento para submissão.
[OUT] O sistema informa as modalidades disponíveis para submissão de trabalhos (Ciências Exatas e Tecnológicas, Biológicas e da Saúde, Humanas, Letras e Artes, Ciências Sociais e Aplicadas).
[IN] O autor escolhe a modalidade referente ao seu trabalho.
[OUT] O sistema oferece espaço para emissão do resumo do trabalho.
[IN] O autor deverá submeter o resumo de seu trabalho.
[IN] O autor finaliza a submissão.

[OUT] O sistema encaminha o trabalho para a avaliação. **Exceção 1ª: Identificação Inválida** 1ª.2 [IN] Usuário deverá estar cadastrado. Retorna ao passo 1.

QUADRO 18 - Caso de uso gerenciar inscrições

UC2 - Caso de uso: gerenciar inscrições [IN] O ouvinte informa sua identificação. [OUT] O sistema informa a programação de eventos. [IN] O ouvinte escolhe os eventos que deseja participar. [OUT] O sistema gera boleto para pagamento da inscrição. [IN] O ouvinte finaliza sua inscrição. [OUT] O sistema inclui o ouvinte no registro de frequência. O ouvinte finaliza sua inscrição. **Exceção 1ª: Identificação Inválida** 1ª.2 [IN] Ouvinte deverá estar cadastrado. Retorna ao passo 1.

QUADRO 19 - Caso de uso avaliar artigos

UC3 - Caso de uso: avaliar artigos [IN] O gerente de conferência informa sua identificação. [OUT] O sistema informa os trabalhos que estão em avaliação. [IN] O gerente de conferência envia os trabalhos para o avaliador (professor, coordenador). [IN] O Avaliador informa sua identificação; [OUT] O sistema recebe a avaliação (Aprovado, Reprovado, Com Resalvas). O gerente de conferencia verifica o parecer do avaliador: Variante: Trabalho aprovado. Variante: Trabalho reprovado. Variante: Trabalho com resalvas. **Variante 6.1: Trabalho aprovado** [OUT] O sistema cadastra o trabalho nos anais do evento. **Variante 6.2: Trabalho reprovado** 6.2.1 [OUT] O sistema informa ao autor que o trabalho foi reprovado na avaliação. **Variante 6.3: Trabalho com ressalvas** 6.3.1 [OUT] O sistema informa ao autor para que seja feita as devidas correções. **Exceção 1ª: Identificação Inválida** 1ª.1 [IN] O gerente de conferência deve estar cadastrado. Retorna ao passo 1. 1ª.2 [IN] Avaliador deverá estar cadastrado. Retorna ao passo 4.

3.4 Modelo conceitual

O modelo conceitual do SGE – Sistema Gerenciador de Eventos é onde está apresentado todo o esquema de relacionamento de classes e as permissões que cada tipo de usuário que sistema contém.

Na figura 4 pode-se ver como é feito o relacionamento das classes. O início do relacionamento está no sistema de evento, que agregado a ele estão as classes evento, trabalhos e usuários. Agregada a classe de evento está a classe de programação que por sua vez faz ligações com as classes de trabalhos e inscrições. A classe

inscrições faz ligação com a classe frequência e recebe do tipo de usuário ouvinte, ligada a classe frequência está a classe certificado, que permite o usuário emitir o seru certificado ao final de um evento de acordo com sua frequência. A classe trabalhos faz ligação com a classe avaliação trabalho e recebe do tipo de usuário autor. A classe usuário faz ligação com a classe inscrições, e recebe de todos os tipos de usuários. Os usuários contidos dentro do sistema conceitual tem uma hierarquia onde o administrador tem o poder total em todas as ações do sistema, seguido dele há o usuário gerente que tem como poderes gerenciar um evento do sistema, em seguida tem o usuário avaliador que tem como função avaliar os trabalhos submetidos pelo tipo de usuário autor e por fim temos o usuário ouvinte que pode somente criar o seu cadastro e se inscrever em um tipo de evento.

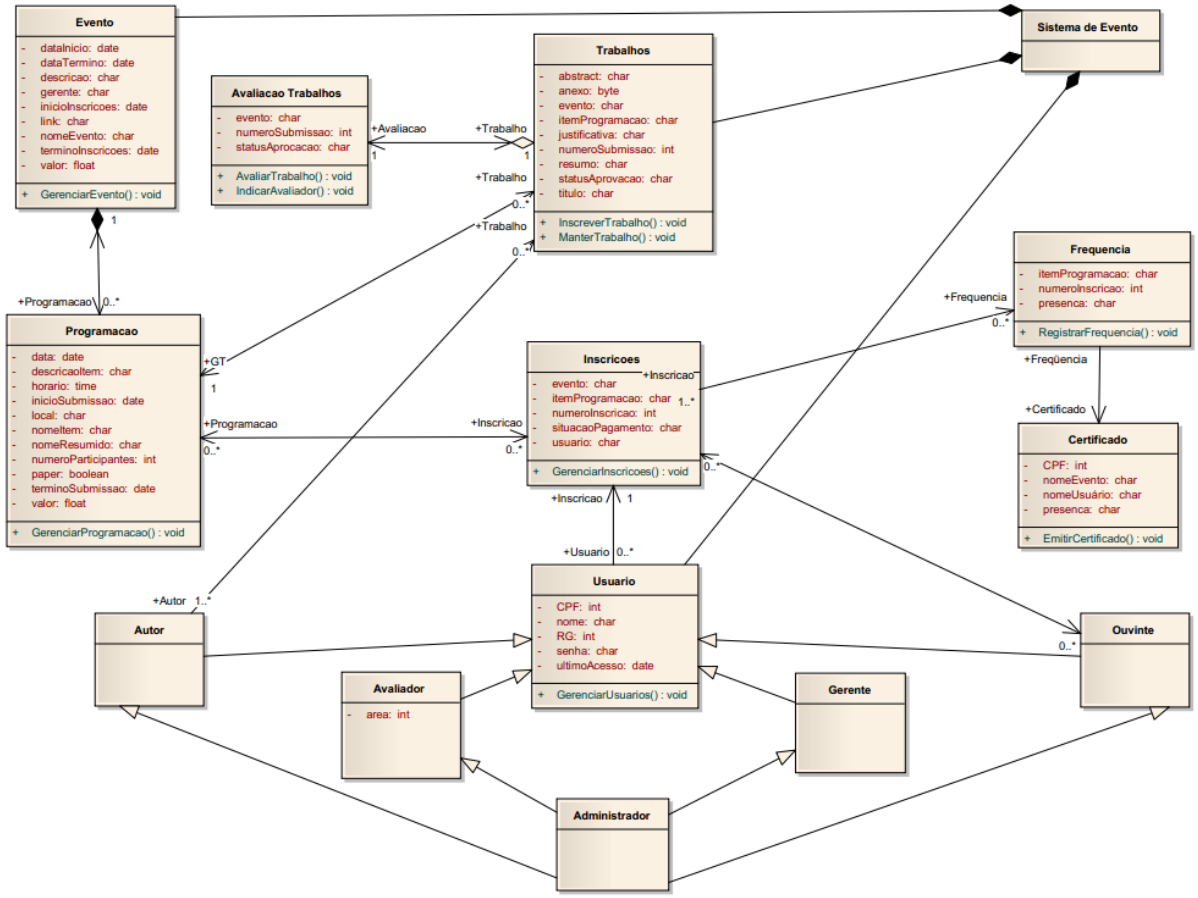

FIGURA 4 - Modelo conceitual SGE – sistema gerenciador de eventos

3.5 Conclusão

O presente capítulo apresentou o projeto e análise do sistema de eventos, mostrando o levantamento de requisitos e casos de uso.

A respeito do levantamento de requisitos foi possível ver o que será desenvolvido no sistema de eventos, apresentando requisitos desde a criação de um evento até a avaliação dos trabalhos.

Os casos de uso apresentam como funcionará o sistema de eventos, mostrando o processo de submissão de trabalhos, a inscrição de ouvintes e como será feita a avaliação dos trabalhos submetidos.

Este capítulo torna-se importante aos posteriores que abordam o desenvolvimento do sistema de eventos, pois servira de base para tal. Esse processo de modelagem é iterativo, pois pode sofrer alterações ao longo do seu desenvolvimento.

4 IMPLEMENTAÇÃO

Neste capítulo são apresentadas as tecnologias e ferramentas utilizadas, a criação de um projeto com ZF e sua estrutura de arquivos bem com a implementação de *layout*, banco de dados, formulários, *models*, *controllers*, *views* e controle de acesso do SGE.

4.1 Tecnologias e ferramentas

No quadro 20 estão relacionadas as tecnologias utilizadas no desenvolvimento do SGE – Sistema Gerenciador de Eventos.

QUADRO 20 - Tecnologias utilizadas

Tecnologia	**Objetivo/Descrição**
PHP – Versão 5.3.5	É uma linguagem de script amplamente utilizada de propósito geral que é especialmente adequado para o desenvolvimento Web que pode ser incorporado com o HTML (PHP, 2012).
CSS	*Cascading Style Sheets* (CSS) é uma "folha de estilo" composta por "camadas" e utilizada para definir a apresentação (aparência) em páginas da internet que adotam para o seu desenvolvimento linguagens de marcação (como XML, HTML e XHTML) (TECMUNDO, 2009 e DEITEL e DEITEL, 2008).
MySQL – Versão 5.5.8	O MySQL é um sistema de gerenciamento de banco de dados (SGBD), que utiliza a linguagem SQL (Linguagem de Consulta Estruturada, do inglês *Structured Query Language*) como interface (MYSQL, 2012).
XHTML	É uma reformulação da linguagem de marcação HTML baseada em XML. Tem com objetivo a exibição de páginas da *Web* em diversos dispositivos (DEITEL e DEITEL, 2008).

(Fonte: citadas no quadro)

No quadro 21 são apresentadas as ferramentas utilizadas no desenvolvimento do SGE – Sistema Gerenciador de Eventos.

QUADRO 21 - Ferramentas utilizadas

Ferramentas	**Objetivo/Descrição**
WampServer – Versão 2.1	É um ambiente de desenvolvimento *Web* para Windows. Ele permite que você crie aplicações *Web* com Apache2, PHP e banco de dados MySQL. Paralelamente, com o PhpMyAdmin permite gerenciar facilmente seus banco de dados (WAMPSERVER, 2012).
PhpMyAdmin – Versão 3.3.9	É uma ferramenta de software livre desenvolvido em PHP, destinado a administrar o MySQL. Suporta uma ampla gama de operações com o MySQL. As operações mais utilizadas pelos usuários são suportadas pela interface (gerenciar banco de dados, tabelas, campos, permissões, etc) (PHPMYADMIN, 2012).
Apache – Versão 2.2.17	É um servidor *Web* responsável por disponibilizar as páginas e todos os recursos de um *site* ou sistema *Web* (APACHE, 2012).
Zend Tool	É uma estrutura para expor funcionalidades comuns, tais como a criação de projetos, geração de código, índices de pesquisa e muito mais (ZEND TECHNOLOGIES, 2012).
Eclipse PDT 3.6.0 - Helios	Ferramenta usada para a criação de aplicações Java. Mas graças a sua arquitetura baseada em *plugins* permite a customização para outras linguagens (LISBOA, 2009).

(Fonte: citadas no quadro)

4.2 Zend Framework

4.2.1 Criação do projeto

A criação de um projeto ZF pode ser facilitada usando a ferramenta *zend tool*, através da linha de comando do *Windows*, como demonstra a figura 5.

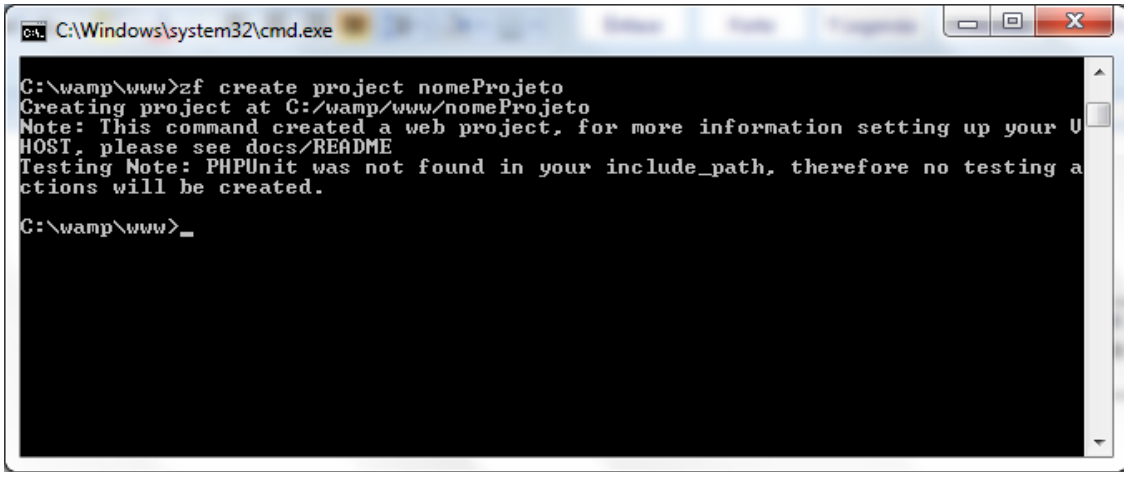

FIGURA 5 - Criação projeto ZF com *zend tool*
(Fonte: ZEND, 2011)

Automaticamente após este processo a *zend tool* cria a estrutura de arquivos padrão usada em projetos ZF.

4.2.2 Estrutura de arquivos

Para verificar o projeto criado pela *zend tool* pode ser utilizada a ferramenta *Eclipse* PDT, nela é possível acessar toda a estrutura de arquivos padrão para o projeto.

A figura 6 demonstra toda a estrutura de arquivos da aplicação. No diretório *Application*, constam todos os arquivos referentes ao aplicativo, ou seja, toda a programação não visualizável externamente. Ainda neste diretório está armazenada toda a estrutura MVC.

No diretório *public* contém arquivos visíveis externamente, como arquivos CSS, *Javascript*, imagens. Resumindo, somente este diretório será acessível ao usuário.

O diretório *tests* basicamente contém informações que correspondem a testes na fase de produção do sistema.

O diretório *docs* recebe a documentação gerada. Já o diretório *library* armazena as bibliotecas externas e o próprio *framework.*

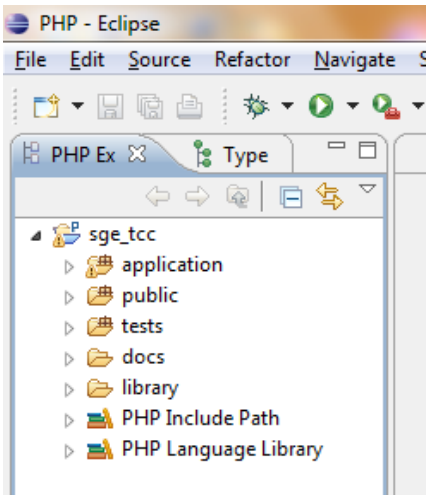

FIGURA 6 - Estrutura de arquivos Eclipse
(Fonte: ECLIPSE, 2012)

4.3 *Layout*

A interface do SGE é padronizada, assim todas as páginas fazem o uso da classe *layout.php*, onde o *layout* padrão foi definido. Os elementos de apresentação

como cores, fontes, margens, posições são definidos em um documento *CSS (Cascading Style Sheet)* externo, chamado *style.css.*

Ao longo do desenvolvimento do *layout* foram realizados testes em diferentes navegadores, para ter a garantia de que o SGE seria exibido sem problemas nos principais navegadores atualmente, Microsoft Internet Explorer, Google Chrome e Mozilla Firefox.

A figura 7 demonstra a estrutura do *layout.* A seção topo corresponde ao cabeçalho, onde é apresentado o nome do sistema e a logomarca da UNIPLAC. Em seguida, é exibido o *menu* que dá acesso às demais páginas do sistema de acordo com as permissões de cada usuário. Temos o *menu* lateral, onde é apresentado *link* com ação para adicionar informação ao sistema. A seção conteúdo, por sua vez é o principal bloco do *layout,* no qual são apresentados os formulários de cadastro, tabelas com as informações já cadastradas, etc. Por fim, no bloco rodapé é exibido o nome do acadêmico que desenvolveu o sistema.

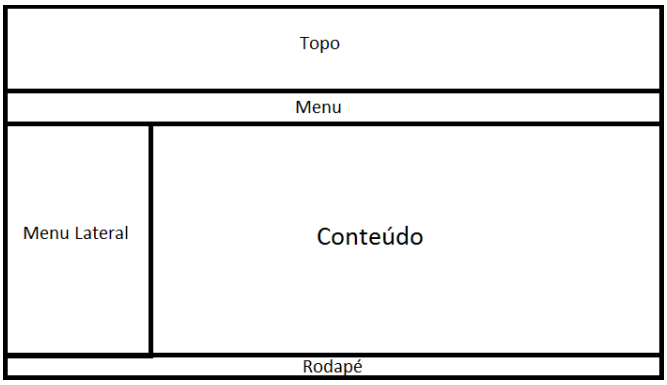

FIGURA 7 - *Layout* do site
(Fonte: SUPPI, 2012)

A figura 8 ilustra a codificação do *layout*. Na linha 1 é definido o *doctype*, que especifica ao navegador que tipo de linguagem a página foi construída, neste caso usamos o *XHTML Transitional.* Na linha 2 é indicado a página do XHTM da W3C e a linguagem, que nesta caso é o português brasileiro. Na linha 3 está localizada a *tag head*, que contém informações como o *charset* utilizado no site, caminho para documento CSS externo, título do site, etc. Na linha 9 se encontra a *tag body*, que é onde todo o conteúdo do site está localizado. Na linha 10 temos a *tag div* geral, onde

se encontra a estrutura do *layout*. Na linha 11 temos a *div* topo, onde se localiza o nome do site e a logomarca da UNIPLAC. Na *div* menu (linha 13) contém os *links* de navegação entre as páginas. Na linha 21 está a *div* conteúdo, que recebe um código em PHP que busca todo o conteúdo gerado no site e apresenta dentro deste bloco. E na linha 24 temos a *div* rodapé, que apresenta o nome do acadêmico que desenvolveu o sistema.

```
1  <?=$this->doctype('XHTML1_TRANSITIONAL')?>
2  <html xmlns="http://www.w3.org/1999/xhtml" lang="pt-br">
3      <head>..
8      </head>..
9      <body>
10         <div id="geral">
11             <div id="topo">
12             </div>
13             <div id="menu">..
20             </div>..
21             <div id="conteudo">
22                 <?php echo $this->layout()->content; ?>
23             </div>
24             <div id="rodape">
25             </div>
26         </div>
27     </body>
28 </html>
```

FIGURA 8 - Código do *layout*

4.4 Conexão com banco de dados

A conexão ao banco de dados utilizando o ZF é realizada através da configuração do arquivo *application.ini*, o qual é gerado na criação do projeto através da *zend tool*.

A figura 9 ilustra o código utilizado para a realização da conexão com o banco de dados. Na linha 16, se tem a configuração do adaptador do banco de dados que será utilizado, neste caso se utilizou o *MySQL*. Na sequência, na linha 17 configura-se o nome do banco de dados. O *host* onde o banco de dados é executado é configurado como mostra a linha 18, que neste caso está executando em um *host* local. As linhas 19 e 20 apresentam as configurações de acesso ao banco de dados como o nome de usuário e senha.

```
16 resources.db.adapter = "PDO_MYSQL"
17 resources.db.params.dbname = "sge_tcc"
18 resources.db.params.host = "localhost"
```

FIGURA 9 - Conexão com banco de dados

4.5 Model

Toda a manipulação de banco de dados no ZF é realizada nos *models*, uma das vantagens do ZF que está manipulação é realizada por meio de objetos, assim se for necessário mudar a plataforma de banco de dados, basta alterar o adaptador no arquivo de configuração, como visto anteriormente.

No quadro 22 é apresentado um fragmento de código da classe *Evento.php*, responsável por fazer a manipulação dos dados contidos no banco de dados para o uso na aplicação. A linha 1 a classe sendo estendida da classe *Zend_Db_Table_Abstract* (linha 2), que é uma classe orientada a objetos para uso em tabelas de banco de dados. Na linha 5 através da variável protegida *$_name* realiza-se o acesso a tabela evento. A linha 7 apresenta a função *getEvento* que tem por parâmetro a variável *$id*. Na linha 9 é definido que a variável *$id* é do tipo inteiro, na sequencia na linha 10 é criada uma variável *$row* onde usa o método *fetchRow* para buscar os *ids* contidos na tabela de evento. Caso não seja encontrado nenhum *id* será transmitida ao usuário uma mensagem de exceção como apresenta as linhas 11 e 12. Caso não haja exceção a variável *$row* apresentará todas as informações contidas no *id* através do método *toArray()*. As linhas 17, 18 e 19 contém método responsável pela realização de adicionar um evento no banco de dados, neste caso são utilizadas instruções padrões SQL de *insert*. As linhas 22, 23 e 24 apresentam o método referente à atualização dos dados do evento, sendo que neste caso são utilizados códigos SQL *update*. E na linha 27 está método responsável pela exclusão de um evento que utilizam comandos SQL *delete*.

QUADRO 22 - Fragmento de código da classe Evento.php

```
1   class Application_Model_DbTable_Evento extends
2   Zend_Db_Table_Abstract
3   {
4
5         protected $_name = 'evento';
6
7         public function getEvento($id)
8         {
9            $id = (int)$id;
10           $row = $this->fetchRow('id = ' . $id);
11           if (!$row) {
```

```
12              throw new Exception("Count not find row $id");
13          }
14          return $row->toArray();
15      }
16
17       public function addEvento($nome_evento, $data_inicio,
18 $data_termino, $site, $gerente, $descricao,$inicio_insc,
19 $termino_insc, $valor)
20 ...
21
22       public function updateEvento($id, $nome_evento,
23 $data_inicio, $data_termino, $descricao, $site, $gerente,
24 $valor, $inicio_insc, $termino_insc)
25 ...
26
27       public function deleteEvento($id)
28 ...
```

4.6 *Forms*

A implementação de formulários utilizando ZF, torna-se simplificada com o uso do componente *Zend_Form*. Com este componente a manipulação de formulários se torna facilitada. O *Zend_Form* cumpre alguns objetivos como filtragem e validação da entrada de elementos, ordenação de elementos entre outros. Para que possa cumprir os seus objetivos o *Zend_Form* aciona outros elementos do ZF como *Zend_Validate, Zend_Filter, Zend_Loader* entre outros.

O quadro 23 mostra um fragmento da classe de formulário *Evento.php*. na linha 1 a classe está estendendo a classe *Zend_Form*, tornando disponíveis os recursos do componente. Na linha 4 é criada a função *__construct* passando por parâmetro uma variável *$options* com valor nulo. Na linha 6 é usado o construtor *parent* tornando *__construct* como uma função pai. O nome do formulário é setado na linha 7 através do método *setName*. Nas linhas 9 e 10 é criado o campo *id*, com um elemento de formulário oculto e um filtro para que o valor do *id* seja inteiro. Nas linhas 12 e 13 é criado o campo *nome do evento*, através de um elemento de formulário de texto. O *label* do campo é setado na linha 14 através do *setLabel*. Na linha 15 através do *setRequired* com o valor em *true*, quer dizer que o campo é requerido não devendo estar em branco. A linha 16 mostra a utilização de um filtro *StripTags*, que tem função de remover o uso de *tags* HTML e PHP do campo. O filtro *StringTrim* como apresenta

na linha 17, tem função de retornar o texto sem espaços em branco à direita e à esquerda. É usado um validador *NotEmpty*, que verifica se o valor que está sendo enviado pelo campo não está vazio. Na criação do campo *data de início* com início na linha 20, a única diferença ao campo anterior é o validador *Date* (linha 27 e 28), que tem como função fazer a formatação da data para o padrão mês-dia-ano informando a localização. Nas linhas 34 e 35 é implementado o botão que submete as informações preenchidas no formulário. O método *addElements* (linha 38), adiciona todos os campos que devem ser apresentados no formulário quando este estiver disponível na *view*.

QUADRO 23 - Fragmento da classe de formulário Evento.php

```
1  class Application_Form_Evento extends Zend_Form
2  {
3
4       public function __construct($options = null)
5       {
6            parent::__construct($options);
7            $this->setName('eventos');
8
9            $id = new Zend_Form_Element_Hidden('id');
10           $id->addFilter('Int');
11
12         $nome_evento = new
13 Zend_Form_Element_Text('nome_evento');
14         $nome_evento->setLabel('Nome do Evento')
15                     ->setRequired(true)
16                     ->addFilter('StripTags')
17                     ->addFilter('StringTrim')
18                     ->addValidator('NotEmpty');
19
20         $data_inicio = new
21 Zend_Form_Element_Text('data_inicio');
22         $data_inicio->setLabel('Data Inicio')
23                     ->setRequired(true)
24                     ->addFilter('StripTags')
25                     ->addFilter('StringTrim')
26                     ->addValidator(new
27 Zend_Validate_Date(array('format' => 'dd-mm-yyyy',
28                                          'locale'=>'pt_BR')))
29                     ->addValidator('NotEmpty');
30
31
32        ...
33
34         $submit = new Zend_Form_Element_Submit('submit');
35         $submit->setAttrib('id', 'submitbutton');
```

```
36
37
38          $this->addElements(array($id, $nome_evento,
39  $data_inicio, $data_termino, $site,
40                                          $gerente, $descricao,
41  $inicio_insc, $termino_insc, $valor ,$submit));
42          }
43  }
```

A figura 10 mostra a visualização do formulário de cadastro de eventos do SGE.

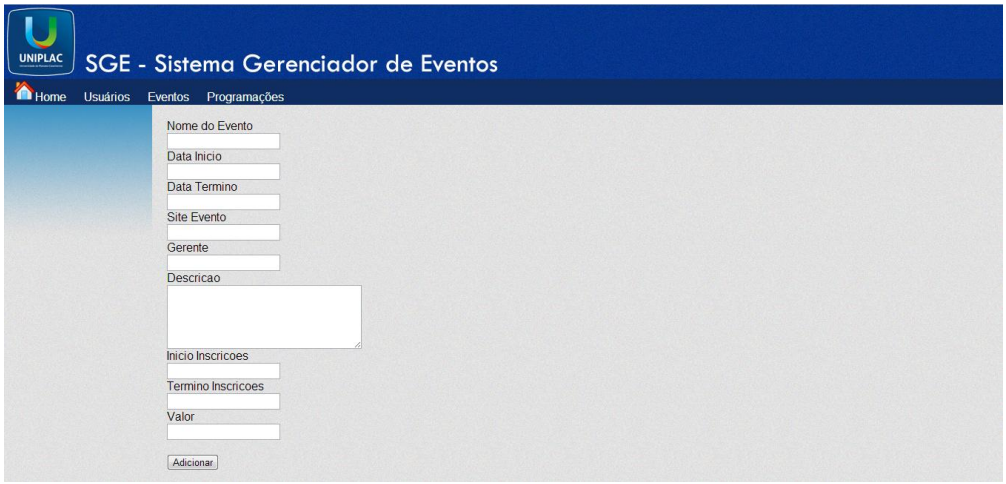

FIGURA 10 - Visualização formulário de eventos
(Fonte: SUPPI, 2012)

4.7 *Controllers*

Os *controllers* no desenvolvimento de aplicações ZF utilizando MVC tem a função de receber as entradas dos usuários, manipular os *models* e fazer a atualizações das *views* de forma apropriada.

Os *controllers* são constituídos por *actions*, que representa a lógica de negocia de cada página, toda *action* que é criada deve ter um arquivo *view*, onde serão apresentadas as informações processadas.

O quadro 24 apresenta um fragmento de código da classe EventoController.php, a qual manipula a lógica de negócio do cadastro de eventos. Na linha 1 encontra-se a função que cria a *action index*. Na linha 4 a variável *$evento* recebe uma instancia da classe *Evento.php* do *model*. Todas as informações que foram

encontradas no banco de dados serão enviadas para *view*, isso ocorre na linha 5 através do método *fetchAll()*. Na linha 8 é criada a função da *action add*, que terá função de adicionar um novo evento na aplicação. A variável *$form* recebe uma nova instancia da classe de formulário *Evento.php* (linha 11). Na linha 12 é adicionado o *label* ao botão de submissão de formulário. Na linha 13 o formulário é enviado para apresentação na *view*. Na linha 15 é verificado se existem dados enviados via *post*. Os dados enviados via *post* são salvos no *array $formData* (linha 16). Na linha 17 é verificado se os dados são válidos. Das linhas 18 a 30 são recebidos todos os atributos enviados via *post*. Nas linhas 31 e 32 cria-se uma nova instancia da classe *model Evento.php*. Das linhas 33 a 36 são setados todos os atributos da função *addEvento* que se encontra no *model Evento.php*. Na linha 38 é feito o redirecionamento para a página *index* se todos os dados foram inclusos normalmente. Caso aconteça algum tipo de erro ou exceção o formulário é reimpresso com os dados já digitados, isso ocorre na linha 40. Na linha 45 é implementado o bloco de código que realiza a alteração dos dados de eventos cadastrados. Na linha 48 é implementado um bloco de código que realiza a exclusão dos eventos cadastrados.

QUADRO 24 - Fragmento de código da classe EventoController.php

```
1   ...
2   public function indexAction()
3       {
4               $evento = new Application_Model_DbTable_Evento();
5               $this->view->evento = $evento->fetchAll();
6       }
7
8       public function addAction()
9       {
10
11          $form = new Application_Form_Evento();
12          $form->submit->setLabel('Adicionar');
13          $this->view->form = $form;
14
15          if ($this->getRequest()->isPost()) {
16              $formData = $this->getRequest()->getPost();
17              if ($form->isValid($formData)) {
18                  $nome_evento = $form->getValue('nome_evento');
19                          $data_inicio = $form-
20  >getValue('data_inicio');
21                          $data_termino = $form-
22  >getValue('data_termino');
23                          $site = $form->getValue('site');
```

```
24                          $gerente = $form->getValue('gerente');
25                          $descricao = $form->getValue('descricao');
26                          $inicio_insc = $form-
27  >getValue('inicio_insc');
28                          $termino_insc = $form-
29  >getValue('termino_insc');
30                          $valor = $form->getValue('valor');
31                  $eventos = new
32  Application_Model_DbTable_Evento();
33                  $eventos->addEvento($nome_evento, $data_inicio,
34  $data_termino, $site, $gerente,
35                                              $descricao, $inicio_insc,
36  $termino_insc, $valor);
37
38                  $this->_helper->redirector('index');
39              } else {
40                  $form->populate($formData);
41              }
42          }
43      }
44
45  public function editAction()
46  ...
47
48  public function deleteAction()
49
50  }
```

4.8 Views

As *views* dentro da implementação MVC do ZF, tem a única função de apresentar a parte visual da aplicação aos usuários.

O quadro 25 apresenta o código usado no arquivo de *view add.phtml* do cadastro de eventos, que representa a *action* add do *controller EventoController.php.* ele contém apenas uma linha, que tem por função imprimir o formulário de cadastro de eventos para o usuário.

QUADRO 25 - Código da *view add.phtml* do cadastro de evento

```
1   <?php echo $this->form ;?>
```

O quadro 26 apresenta o código implementado para a edição de eventos cadastrados, a diferença desta *view* para a anterior está na implementação do *controller* na *action edit*, que quando solicitada pelo usuário apresentará o formulário preenchido

com os dados a serem alterados.

QUADRO 26 - Código do arquivo *view edit.phtml* do cadastro de evento

```
1  <?php echo $this->form ;?>
```

O quadro 27 apresenta o código do arquivo *delete.phtml* do cadastro de eventos. Na linha 1 é apresentada uma mensagem ao usuário perguntando se ele tem certeza da exclusão do evento, na linha 2 é complementada mensagem com o nome do evento que foi buscado no banco de dados através do método *escape.* Nas linhas 4 e 5 é desenvolvido um formulário que usa a *action delete* e o método *post.* Na linha 7 é criado um campo oculto com o *id* do evento. Nas linhas 9 e 10 são criados dois botões para que o usuário confirme ou negue a exclusão do evento.

QUADRO 27 - Código do arquivo *view delete.phtml* do cadastro de evento

```
1  <p>Você tem certeza que deseja deletar o evento
2    '<?php echo $this->escape($this->evento['nome_evento']); ?>'?
3  </p>
4  <form action="<?php echo $this->url(array('action'=>'delete'));
5  ?>" method="post">
6  <div>
7    <input type="hidden" name="id" value="<?php echo $this-
8  >evento['id']; ?>" />
9    <input type="submit" name="del" value="Sim" />
10   <input type="submit" name="del" value="Não" />
11 </div>
12 </form>
```

O quadro 28 apresenta o código do arquivo da *view index.phtml*, onde será apresentado ao usuário todos os eventos cadastrados na aplicação. Nas linhas 1 e 2 é criado um *link* para a página *add.phtml*. Na linha 3 é criada uma tabela. Das linhas 4 a 15 apresentam os nomes das colunas da tabela onde mostrará os eventos cadastrados. Na linha 16 é criado um *foreach* que irá verificar os eventos cadastrados no banco de dados, e das linhas 17 a 28 são buscados os valores dos atributos contidos no banco de dados e inseridos na tabela. Das linhas 30 a 38 são criados links que direcionaram para as páginas de editar e deletar, para cada dado cadastrado no banco de dados da aplicação. Na linha 39 termina a condição *foreach* e na linha 40 é fechada a *tag* da tabela criada.

QUADRO 28 - Código da *view index.phtml* do cadastro de evento

```
1  <p><a href="<?php echo $this->url(array('controller'=>'evento',
2          'action'=>'add'));?>">Adicionar Novo Evento</a></p><br/>
3  <table>
4  <tr>
5      <th>Nome</th>
6      <th>Início</th>
7      <th>Término</th>
8      <th>Site</th>
9      <th>Gerente</th>
10     <th>Descriçãoo</th>
11     <th>Início Iscrição</th>
12     <th>Término Inscrição</th>
13     <th>Valor</th>
14     <th> </th>
15 </tr>
16 <?php foreach($this->evento as $eventos) : ?>
17 <tr>
18     <td><?php echo $this->escape($eventos->nome_evento);?></td>
19     <td><?php echo $this->escape($eventos->data_inicio);?></td>
20     <td><?php echo $this->escape($eventos->data_termino);?></td>
21     <td><?php echo $this->escape($eventos->site);?></td>
22     <td><?php echo $this->escape($eventos->gerente);?></td>
23     <td><?php echo $this->escape($eventos->descricao);?></td>
24     <td><?php echo $this->escape($eventos->inicio_insc);?></td>
25       <td><?php echo $this->escape($eventos-
26 >termino_insc);?></td>
27      <td><?php echo $this->escape($eventos->valor);?></td>
28     <td>
29         <a href="<?php echo $this-
30 >url(array('controller'=>'evento',
31             'action'=>'edit', 'id'=>$eventos-
32 >id));?>">Editar</a>
33         <a href="<?php echo $this-
34 >url(array('controller'=>'evento',
35             'action'=>'delete', 'id'=>$eventos-
36 >id));?>">Deletar</a>
37     </td>
38 </tr>
39 <?php endforeach; ?>
40 </table>
```

4.9 Controle de acesso

4.9.1 Autenticação

A figura 11 mostra a tela inicial do SGE, onde o usuário poderá realizar a sua entrada no sistema através de um *login* e uma senha. Caso o *login* e a senha sejam inválidos, é apresentada uma mensagem de usuário ou senha inválido, assim o usuário

deverá tentar novamente.

Este tipo de tratamento no ZF é realizado através do componente *Zend_Auth*.

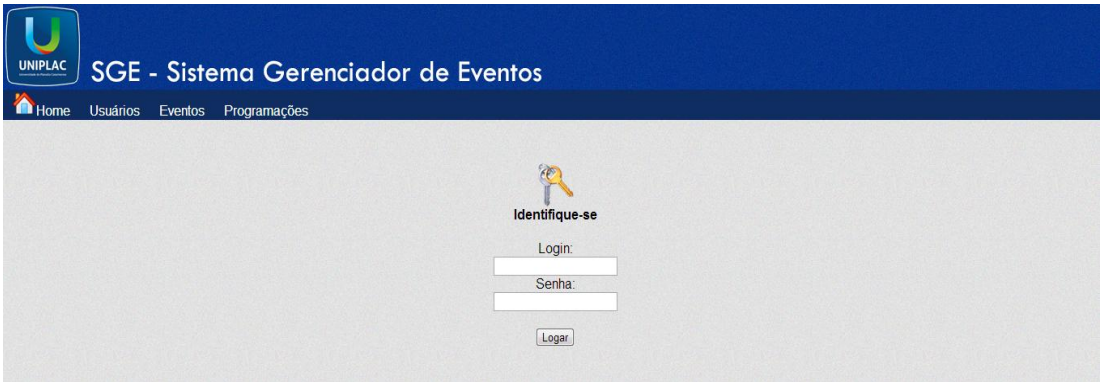

FIGURA 11 - Visualização página inicial do sistema
(Fonte: SUPPI, 2012)

O fragmento de código apresentado no quadro 29, exibe a ação *login* que se encontra na classe controladora *AuthController.php*, que tem como função efetuar o *login* na aplicação. Na linha 3 é criada a função pública da ação *loginAction*. Da linha 5 a 8, é implementado o componente *flashMessenger* que é encarregado de apresentar as mensagens de erros na camada *view*, caso o *login* seja inválido. Nas linhas 9 e 10 é instanciado o formulário de *login* e o passando para a camada *view*. Nas linhas 11 e 12 é realizada uma verificação se existem dados enviados pelo método *post()*. As linhas 13, 14 e 15 realizam a verificação se o formulário foi preenchido corretamente, caso não seja ele não entrará na lógica de *login* e entrará diretamente para a linha 41, que realiza a tarefa de preencher o formulário com os dados já enviados. Caso a verificação seja verdadeira iniciara o processo de *login* nas linhas 17 e 18, onde iniciara o adaptador de banco de dados. Nas linhas 19 e 20 é iniciado a processo de autenticação no banco dados. Na linha 21 é definida a tabela do banco de dados que estão os dados de acesso do usuário. Nas linhas 22 e 23 são definidas as colunas da tabela do banco de dados que estão os dados que identificam e credenciam a entrada do usuário no sistema, respectivamente. Nas linhas 24 e 25 são definidos os dados para processamento do *login*. As linhas 26 e 27 estão destinadas a realização do *login* utilizando uma instancia do componente *Zend_Auth* e o método de autenticação *authenticate()*. Na linha 28 inicia-se um bloco de instruções *if* que fará a verificação se

o *login* foi efetuado com sucesso. Da linha 29 a 32 são armazenados os dados do usuário em sessão. As linhas 33 e 34 são responsáveis por fazer o redirecionamento do usuário para o controlador protegido, que neste caso é o controlador de evento. Da linha 35 a 39 é feito o tratamento de dados inválidos apresentando a mensagem "usuário ou senha inválidos!" fazendo o redirecionamento para a página de login.

QUADRO 29 - Código da ação *loginAction* do controlador *AuthController.php*

```
1   …
2
3   public function loginAction()
4       {
5           $this->_flashMessenger = $this->_helper-
6   >getHelper('FlashMessenger');
7        $this->view->messages = $this->_flashMessenger-
8   >getMessages();
9        $form = new Application_Form_Login();
10           $this->view->form = $form;
11       if ( $this->getRequest()->isPost() ) {
12            $data = $this->getRequest()->getPost();
13            if ( $form->isValid($data) ) {
14                 $login = $form->getValue('login');
15                 $senha = $form->getValue('senha');
16
17                   $dbAdapter =
18  Zend_Db_Table::getDefaultAdapter();
19                $authAdapter = new
20  Zend_Auth_Adapter_DbTable($dbAdapter);
21                $authAdapter->setTableName('usuario')
22                        ->setIdentityColumn('login')
23                         ->setCredentialColumn('senha');
24                $authAdapter->setIdentity($login)
25                        ->setCredential($senha);
26             $auth = Zend_Auth::getInstance();
27              $result = $auth->authenticate($authAdapter);
28                if ( $result->isValid() ) {
29                     $info = $authAdapter-
30  >getResultRowObject(null, 'senha');
31                  $storage = $auth->getStorage();
32                    $storage->write($info);
33                return $this->_helper->redirector->goToRoute(
34  array('controller' => 'evento'), null, true);
35                } else {
36                $this->_helper->FlashMessenger('Usuário ou
37  senha inválidos!');
38                $this->_redirect('/auth/login');
39                }
40           } else {
41                $form->populate($data);
42           }
43       }
```

```
44   }
45 ...
```

O quadro 30 apresenta o código da função *init* do controlador de evento. Este código é responsável por proteger a página de eventos com *login* e senha. Na linha 1 é criada a função *init*, onde as instruções contidas terão validade para todo o controlador. Na linha 3 é realizada uma verificação se há algum usuário autenticado no sistema. Nas linhas 4 e 5 faz o redirecionamento para a página de *login* caso não encontre nenhuma identificação de usuário no sistema.

QUADRO 30 - Código função *init* da classe *EventoController.php*

```
1      public function init()
2      {
3          if ( !Zend_Auth::getInstance()->hasIdentity() ) {
4          return $this->_helper->redirector->goToRoute(
5  array('controller' => 'auth'), null, true);
6          }
7      }
```

O quadro 31 apresenta o código de *logout*, código que desliga o usuário de dentro do sistema. A linha um mostra a criação da função de *logoutAction*. Na linha 3 é recuperada uma instancia do *Zend_Auth*. Na linha 4 é feita a limpeza dos dados do usuário pelo método *clearIndentity()*. E na linha 5 é feito o redirecionamento para a página de *login*, para que o usuário saiba que já não está mais autenticado no sistema.

QUADRO 31 - Código da ação *logoutAction* do controlador *AuthController.php*

```
1      public function logoutAction()
2      {
3              $auth = Zend_Auth::getInstance();
4        $auth->clearIdentity();
5        return $this->_helper->redirector('index');
6      }
```

4.9.2 Autorização

A autorização faz a verificação das permissões do usuário que está autenticado no sistema, e tendo como base estas permissões, permitir ou bloquear o acesso do usuário a determinados recursos ou páginas da aplicação.

No caso do SGE são 5 os tipos de usuários, assim como mostra o quadro 32.

QUADRO 32 - Permissões de usuários

Tipo de usuário	**Permissões**
Administrador	O administrador tem o controle total do sistema podendo realizar qualquer tipo de operação.
Gerente de Conferência	O gerente de conferência tem quase todos os controles do administrador, pois não pode cadastrar novos eventos.
Avaliador	O avaliador tem como função avaliar os trabalhos submetidos pelos autores.
Autor	O autor é um usuário cadastrado no sistema que tenha permissão de submissão de trabalhos.
Ouvinte	O ouvinte deve ser um usuário cadastrado no sistema que poderá fazer a inscrição em algum trabalho de um evento.

4.10 Análise do zend framework

A decisão por usar o ZF foi pela proposta que o framework passa aos seus utilizadores, que é a criação de projetos bem estruturados juntamente com redução do tempo de desenvolvimento melhorando a qualidade das aplicações *web.*

A instalação do ZF não é algo complicado de ser realizado, é preciso somente fazer o seu download e setá-lo nas variáveis de ambiente do sistema operacional. A criação de projetos ZF é feito por uma ferramenta própria oferecida pelo próprio *framework*, a *Zend Tool*, onde são passadas instruções de criação de projetos, *controllers*, *views*, *models*, *actions* através da linha de comando do próprio *prompt* de comando, do Windows. Durante o desenvolvimento vários componentes foram de excelente utilidade, como é o caso do *Zend_Form*, que oferece uma criação de formulários de forma bastante fácil e intuitiva, oferecendo validadores e filtros que podem ser implementados com poucas linhas de código. Outro componente muito útil e que facilita o desenvolvimento é o *Zend_Auth*, com ele é possível criar um sistema de *login* de forma rápida e segura. Ainda existe componente de *layout* do ZF, que em apenas uma classe é possível definir a estrutura da interface da aplicação toda. A conexão ao banco de dados ocorre com muita facilidade, é apenas inclusa as linhas de conexão no arquivo de configuração do projeto criado com o uso da *Zend Tool.*

O ZF utiliza o padrão de projetos MVC, com isso a experiência com esse tipo de projeto no início dificultou um pouco o aprendizado, por falta de prática, mas com o passar do tempo e o convívio com este tipo de padrão se tornou um método mais útil e organizado para o desenvolvimento da aplicação.

O ZF possui várias vantagens, entre elas a sua gama de componentes de baixo acoplamento, podendo utilizar somente os componentes que realmente interessam para o desenvolvimento da aplicação. Outra vantagem do ZF é a sua documentação completa disponível em língua inglesa, além de livros que ajudam no aprendizado em outras linguagens. Por ter uma grande quantidade de componentes a sua total aprendizagem necessitaria de um tempo mais prolongado de estudo.

No geral o *framework* cumpre o que promete na facilitação e na agilidade do desenvolvimento de projetos *web* em PHP com MVC, com componentes de fácil aprendizagem e utilização.

4.11 Conclusão

O presente capítulo apresentou a fase de implementação do SGE, onde demonstra as ferramentas utilizadas, os componentes do ZF e a parte de controle de acesso da aplicação, demonstrando assim a utilização do ZF no desenvolvimento de aplicações *web* em linguagem PHP.

Sobre as ferramentas foi demonstrada a utilidade que cada uma delas terá para o desenvolvimento da aplicação. Os componentes demonstrados nesta etapa mostra exatamente como o ZF utiliza cada um no desenvolvimento utilizando o padrão de projetos MVC.

O controle de acesso apresenta como são controlados os tipos de usuários dentro do sistema e qual será a permissão de cada usuário autenticado no sistema.

Este capítulo torna-se importante por demonstrar na prática o estudo realizado em cima do *framework*, para o desenvolvimento.

5 CONSIDERAÇÕES FINAIS

A *Web* 2.0 está cada dia mais presente no dia a dia dos internautas e dos desenvolvedores *Web.* Entretanto, o desenvolvimento dos sistemas *Web* se torna um desafio para os desenvolvedores, onde a dinamicidade das aplicações exigem uma constante atualização dos desenvolvedores, exigindo ferramentas que agilizem o processo de desenvolvimento.

Este trabalho teve como objetivo explorar o *framework Zend* aplicado no desenvolvimento de sistemas, demonstrando algumas características através de um estudo de caso e descrição de sua modelagem e especificações. Para tal, foram estudados os componentes necessários para o desenvolvimento, características e arquitetura do *framework,* tendo sido este objetivo cumprido.

Em relação aos objetivos específicos, o primeiro que tinha como função a identificação de características do *framework zend* disponíveis para o desenvolvimento Web foi alcançado no capítulo 2, onde foi realizado um levantamento bibliográfico sobre o ZF.

Já o segundo objetivo, que visava a demonstração das características do *framework* zend através de um estudo de caso foi alcançado e demonstrado no capítulo 4, onde é mostrado detalhes do desenvolvimento do estudo de caso, destacando-se as características utilizadas para tal. Para viabilização do estudo de caso, a modelagem do sistema foi definida no capítulo 3, onde foram abordados, os requisitos funcionais e suplementares, bem como os casos de uso e o modelo conceitual.

O capítulo 4, abordou a implementação do sistema, descrevendo o desenvolvimento das características do ZF no estudo de caso proposto com a utilização de ferramentas Eclipse-PDT para PHP, WAMP (*Apache, MySQL,* PHP 5), além da

Zend Tool, ferramenta nativa do *framework*.

Como dificuldades encontradas, ressalta-se a falta de prática do autor para o desenvolvimento na arquitetura MVC, que se tornou a fase de aprendizado mais demorada. Porém, após esta fase inicial, a arquitetura se demonstrou organizada e facilitadora para o processo de desenvolvimento.

O uso da ferramenta Eclipse-PDT em conjunto com o ZF constituem ferramentas que facilitam o trabalho de desenvolvimento utilizando a linguagem PHP 5. Além disso, o uso do WAMP facilita na visualização do sistema, à medida que se encontram o servidor *Apache* e o banco de dados *MySql*, tornando a aplicação transparente tanto para o desenvolvimento.

O presente estudo foi desafiador, uma vez por se tratar de uma forma nova de desenvolver em PHP. No entanto, este estudo acrescentou de maneira positiva para a expansão do conhecimento acerca de novas possibilidades de desenvolvimento *Web*.

Por fim, sugere-se a realização de trabalhos futuros e novos estudos de casos que possibilitem aprofundar o conhecimento sobre o ZF, utilizando sua nova versão 2.0 que contém novos recursos a serem estudados.

REFERÊNCIAS BIBLIOGRÁFICAS

ALLEN, R.; LO, N.; BROWN, S. **Zend Framework em Ação**. São Paulo: Alta Books, 2009.

APACHE. **Site oficial do Apache**, 2012. Disponível em: <http://www.apache.org/>. Acesso em: 10 set. 2012.

BAHIA. **Ambiente Educacional**. Secretaria da Educação do Estado da Bahia. Disponível em: <http://ambiente.educacao.ba.gov.br/>. Acesso em: 10 abr. 2012.

CAKE SOFTWARE FOUNDATION, Inc. **Site da Comunidade de desenvolvimento do framework CakePHP**. Disponível em: <http://cakephp.org/>. Acesso em: 27 mar. 2012.

CODEIGNITER, Inc. **Site oficial do framework CodeIgniter**. Disponível em: <http://codeigniter.com/ >. Acesso em: 29 mar. 2012.

CONTE, T.; MENDES, R.; HORTA TRAVASSOS, G. Processos de Desenvolvimento para Aplicações Web: Uma Visão Sistemática. In: XI Simpósio Brasileiro de Sistemas Multimídia e Web – WebMedia, 2005, Poços de Caldas. **Anais...** Disponível em: <http://lens.cos.ufrj.br:8080/ESEWEB/materials/RSProcesso Web/2005_10_31_Conte_WebMedia_2005_pubform.pdf > . Acesso em 12 out. 2011.

DEITEL, P.J.; DEITEL, H.M. **Ajax, Rich Internet Applications e desenvolvimento Web para programadores.** Trad. Célia Taniwaki e Daniel Vieira. São Paulo: Pearson Prentice Hall, 2008.

ECLIPSE, Inc. **Site oficial da ferramenta de desenvolvimento Ecplise**. Disponível em: < http://www.eclipse.org/>. Acesso em: 10 jun. 2012.

JOBSTRAIBIZER, F. **Guia Profissional PHP**. São Paulo: Digerati Books, 2009.

LISBOA, F. G. D. S. **Zend Framework Componentes Poderosos para PHP**. São Paulo: Novatec, 2009.

LISBOA, F. G. D. S. **Criando Aplicações PHP com Zend e Dojo**. São Paulo: Novatec, 2010.

LOTAR, A. **Programando com ASP.NET MVC**. São Paulo: Novatec, 2011.

MAGENTONET. O que é Magento? **MagentoNet**. Disponível em:

<http://www.magento.net.br/o-que-e-magento>. Acesso em: 17 abr. 2012.

MINETTO, E. L. **Frameworks para Desenvolvimento em PHP**. São Paulo: Novatec, 2007.

MYSQL. **Site oficial do MySQL**, 2012. Disponível em: <http://www.mysql.com/>. Acesso em: 10 set. 2012.

PAMPLONA, V. **Introdução ao Apache Lucene**, 2009. Disponível em: <http://vitorpamplona.com/wiki/ Introdu%C3%A7%C3%A3o%20ao%20 Apache%20Lucene>. Acesso em: 17 abr. 2012.

PHP Frameworks. **PHP Frameworks**, 2007. Disponível em: <http://www.phpframeworks.com/>. Acesso em: 26 mar. 2012.

PHP. **Site oficial do PHP**, 2012. Disponível em: <http://www.php.net/>. Acesso em: 10 set. 2012.

PHPMYADMIN. **Site Oficial do phpMyAdmin**, 2012. Disponível em: <http://www.phpmyadmin.net>. Acesso em: 10 set. 2012.

POTAPCZUK, D. D. O. Ambiente Educacional Web. **Blog de Diego Oliveira Potapczuk**, 2010. Disponível em: <http://www.diegoliveira.com.br/blog/2010/04/ 15/ambiente-educacional-web/>. Acesso em: 17 abr. 2012.

SUPPI, L.F.P. **SGE: Sistema de gerenciamento de eventos**. 2012. Trabalho de Conclusão de Curso (Bacharelado em Sistemas de Informação), Universidade do Planalto Catarinense - UNIPLAC, Lages.

SYMFONY, Inc. **Site oficial do projeto framework Symfony**. Disponível em: <http://www.symfony-project.org/ >. Acesso em: 29 mar. 2012.

TECMUNDO. O que é CSS? **Site Tecmundo**, 2009. Disponível em: <http://www.tecmundo.com.br/programacao/2705-o-que-e-css-.htm>. Acesso em: 10 set. 2012.

WAMPSERVEER. **Site oficial do WampServer**, 2012. Disponível em: <http://www.wampserver.com>. Acesso em: 10 set. 2012.

WIKIPEDIA. **ZEND Framework**. Wikipédia, 2012. Disponível em: <http://pt.wikipedia.org/wiki/Zend_Framework>. Acesso em: 17 abr. 2012.

ZEND TECHNOLOGIES, Inc. **Site Oficial do Zend Framework**. Disponível em: <http://framework.zend.com/ >. Acesso em: 29 mar. 2012.

ZEND TECHNOLOGIES, Inc. **Zend Framework versão 1.11.11**, 2011. Disponível em: < http:// framework.zend.com >. Acesso em: 10 fev. 2012.

ZEND TECHNOLOGIES, Inc. **Zend Framework: guia de referência do programador**, 2009. Disponível em: < http://diariodecodigos.info/zend/ptBR/>. Acesso em: 10 mar. 2012.

ANEXOS

ANEXO A - ARTIGO

DESENVOLVIMENTO DE APLICAÇÕES UTILIZANDO O FRAMEWORK ZEND

Luiz Felipe Pinheiro Suppi[1], Sabrina B. Koerich[1]

Sistemas de Informação – Universidade do Planalto Catarinense (UNIPLAC)

88509-000 – Lages – SC – Brasil

luizfelipepsuppi@yahoo.com.br, Sabrina.uniplac@gmail.com

Abstract. *The frameworks are intended to structure projects and streamline your development process. This article aims to explore the Zend framework, applied to the development of a case study. For this we conducted a literature review on the concepts and key features of the framework seeking to identify and explore their potential. The case study is a system event management for the Universidade do Planalto Catarinense - Brazil (UNIPLAC), this has the basic functionality of the control event subscriptions, submissions and job evaluations, as well as the frequency control. The analysis done on the framework shows how easy learning outcome.*

Resumo. *Os frameworks têm como objetivo estruturar projetos e agilizar o seu processo de desenvolvimento. Este artigo tem como objetivo explorar o framework Zend, aplicado ao desenvolvimento de um estudo de caso. Para isso foi realizado um levantamento bibliográfico sobre os conceitos e principais características do framework buscando explorar e identificar o seu potencial. O estudo de caso é um sistema de gerenciamento de eventos para a Universidade do Planalto Catarinense (UNIPLAC), este tem como funcionalidades básicas o controle de inscrições de eventos, submissões e avaliações de trabalho, bem como o controle de frequência. A análise feita sobre o framework mostra como resultado um fácil aprendizado.*

1. Introdução

A Internet evoluiu muito nos últimos anos, principalmente no meio de desenvolvedores de *softwares*. Com o uso da Internet para fins comerciais, surgiu a necessidade por aplicações *Web* com recursos semelhantes aos de aplicações *desktop*.

Com o advento da Web 2.0, que possibilitou os usuários irem além da leitura de páginas interagindo com as aplicações, surgiu novas tecnologias de desenvolvimento para a Internet, como é o caso dos frameworks.

Segundo Jobstraibizer (2009), um *framework* é um conjunto de componentes que contém uma arquitetura e uma estrutura interna básica para o desenvolvimento de uma aplicação. Funciona como uma aplicação semipronta que deve ser estendida e personalizada para que um sistema desenvolvido funcione corretamente.

Dentre esses *frameworks*, está o *Zend Framework* (ZF), o qual é uma biblioteca PHP para desenvolvimento de aplicações *Web*. Os componentes se encaixam para oferecer um *framework* de pilha

completa com todos os componentes necessários para construir aplicações modernas, fáceis de serem criadas e mantidas (ALLEN; LO; BROWM, 2009).

Dessa forma, a utilização do *Zend Framework* e seus componentes para o desenvolvimento de aplicações, torna o trabalho rápido e fácil de ser mantido, oferecendo ao usuário uma aplicação rica em conteúdo.

Este artigo apresenta um estudo que explora e identifica as características do *framework Zend,* aplicado em um estudo de caso, como forma de demonstrar como o seu uso pode ser empregado em sistemas *Web*.

2. Framework Zend

Frameworks têm como principal objetivo o reaproveitamento de códigos, com isso torna o desenvolvimento simplificado e fácil, com uma coleção de códigos-fonte, classes, funções e metodologias. Segundo Minetto (2007) um framework de desenvolvimento é uma "base" de onde se pode desenvolver algo maior ou mais específico.

Para o desenvolvimento *web* temos os *frameworks* para a linguagem PHP, dentre os quais se pode citar o CakePHP (CAKE, 2012), o CodeIgniter (CODEIGNITER, 2012), o Symfony (SYMFONY, 2012) e o Zend *Framework* (ZEND, 2012).

O quadro 1 apresenta um estudo comparativo, disponibilizado pelo site PHP Frameworks (2007), no qual pode-se observar que o *framework* Zend atende todas as características em análise, possibilitando se ter uma idéia da sua completude. A estrutura MVC mostra quais *frameworks* tem suporte a esse tipo de arquitetura, a qual será explanada posteriormente. O *Multiple DB* mostra que a estrutura do framework suporta múltiplos bancos de dados. *Templates* mostra quais frameworks tem embutido em sua estrutura a opção de *layout* para as interfaces web. *Cache* indica se o framework fornece um modo de armazenar informações em modo *cache,* ou seja, de forma temporária. Validação indica se o framework contém componentes de validação e filtros para formulários. *Ajax* indica que o framework vem com suporte para esse tipo de linguagem dinâmica. O *Auth* Módulo refere se constam componentes de autenticação de usuários. E a coluna módulos indica que o framework permite a instalação e utilização de módulos extras, tais como como analisador de RSS, módulo gerador de PDF, entre outros.

Quadro 1. Comparativo de frameworks PHP

PHP Framework	MVC	Multiple DB	Templates	Cache	Validação	Ajax	Auth	Módulos
Akelos	X	X	X	X	X	X	X	X
ash.MVC	X	-	X	-	X	-	X	X
CakePHP	X	X	-	X	X	X	X	X
CodeIgniter	X	X	X	X	X	-	-	-
DIY	X	-	X	X	-	X	-	-
Componentes eZ	-	X	X	X	X	-	-	-
Fusebox	X	X	-	X	-	X	-	X
PHP em TRAX	X	X	-	-	X	X	-	X
PHPDevShell	X	-	X	X	X	X	X	X
PhpOpenbiz	X	X	X	-	X	X	X	-
Prado	X	X	X	X	X	X	X	X
QPHP	X	X	X	-	X	X	X	X
Gaivota	X	X	X	X	X	X	X	X
Symfony	X	X	-	X	X	X	X	X
Wact	X	X	X	-	X	-	-	X
WASP	X	-	X	-	X	X	X	X
Yii	X	X	X	X	X	X	X	X
Zend	X	X	X	X	X	X	X	X
Zoop	X	X	X	X	X	X	X	-

(Fonte: Php Frameworks, 2007)

2.1 Características

No final de 2005 a *Zend Technologies* deu início ao *Zend Framework* como parte do projeto PHP *Collaboration* com fins de estimular o uso do PHP. *Zend Framework* (também referido como ZF) foi criado para ajudar a garantir que a criação de *websites* baseados em PHP fosse fácil e pudesse ser mantido em longo prazo (ZEND, 2012).

ZF é um *framework* para desenvolvimento *Web* orientado a objetos, livre e *open source*, quer dizer que o código-fonte está disponível. Sua licença é compatível com BSD *License*, que permite que aplicações criadas com o framework possam ter código-fonte proprietário.

ZF é uma biblioteca de componentes fracamente acopladas onde podem ser usados de maneira independente, sem a obrigação de usar tudo o que o *framework* oferece. Segundo Lisboa (2010) a novidade que além da opção de usar seus componentes separadamente, o ZF também oferece uma implementação avançada do padrão Modelo-Visão-Controlador (MVC).

O ZF possui quatro principais características, descritas nas sessões seguintes.

2.1.1 Extrema simplicidade e produtividade

O ZF foi projetado para ser simples. O ZF manteve o espírito de simplicidade da programação PHP segundo Lisboa (2010), isso implica numa curva de aprendizagem baixa.

Usar um *framework* tem a ideia de reduzir custos de treinamento, agilizando a colocação no mercado. Desenvolvendo dessa forma faz com que se escolha peças necessárias para suas aplicações, por sua fácil localização de códigos reduzirá custos de manutenção.

2.1.2 Últimas características de desenvolvimento Web

O ZF possui as últimas características para o desenvolvimento *Web*, assim como suporte à AJAX utilizando JSON, uma edição PHP nativa do mecanismo de busca Lucene[3], formato de dados e acesso fácil à sindicalização, o ZF pretende ser o principal local de consumo e publicação de *web service* e biblioteca de classes PHP 5 orientada a objetos e com alta qualidade.

Com essas características constata-se que seja possível desenvolver aplicações usando o que tem de mais recente em desenvolvimento *Web*.

2.1.3 licença segura e confiável (para empresas)

Como já foi mencionado anteriormente o ZF usa uma licença baseada em BSD *License*, que permite o código-fonte de aplicações desenvolvidas com ZF possam ser proprietários. Isso é importante para as empresas que nem sempre podem liberar o código-fonte da aplicação.

Essa é uma forma de proteger a propriedade intelectual, ou valor agregado, construído sobre o ZF, segundo Lisboa (2009).

2.1.4 Completamente testado – fácil e seguramente extensível

O ZF foi testado desde o começo com requisitos de cobertura de código para garantir que todo código contribuído tenha sido testado, de forma que possa se manter estável e fácil de estender.

2.2 Arquitetura

O ZF é composto por vários componentes e podem ser agrupados em cinco categorias: MVC, desenvolvimento rápido de aplicações, banco de dados, internacionalização e localização e autenticação, autorização e gerenciamento de sessão.

Nas sessões a seguir são mostradas cada uma das cinco categorias juntamente com seus componentes.

2.2.1 Modelo-Visão-Controlador (MVC)

O MVC é um padrão de projeto orientado a objetos que tem como principal objetivo separar o projeto em camadas. A separação em camadas serve para verificar e detectar problemas de forma menos incômoda. O padrão MVC é de alto nível, segundo Lisboa (2009) “ele dá uma ideia da arquitetura geral de uma aplicação”.

Segundo Lotar (2011), a Figura 1 representa a relação entre o controlador, o modelo e a visão. O controlador aceita a requisição do usuário, o qual se necessário se comunica com a camada modelo que realiza a interface com o banco de dados, fazendo então a devolução das informações para o controlador. Por fim, o controlador envia as informações necessárias para a visão a qual apresenta ao usuário.

[3] Lucene é uma biblioteca para indexação e consulta de textos em código aberto (PAMPLONA, 2012).

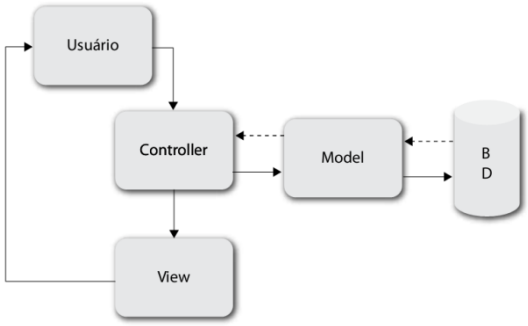

Figura 1 – Relação entre o usuário, o controlador, o modelo e a visão [Lotar, 2011]

2.2.2 Desenvolvimento rápido de aplicações (RAD)

Uma das vantagens dos *frameworks* é justamente o RAD, por proporcionar uma agilidade no desenvolvimento das aplicações. Segundo Jobstraibizer (2009), "assim os códigos tornam-se menores, e podem ser amplamente reaproveitados em vários outros sistemas que utilizem o mesmo *framework"*.

O ZF usa o RAD para acelerar a configuração inicial da aplicação. Segundo Lisboa (2010), o `zend_tool` fornece uma ferramenta de suporte e um cliente linha de comando capaz de gerar a estrutura do projeto, como artefatos MVC e mais.

2.2.3 Banco de Dados

O ZF faz o uso de práticas de programação de banco de dados, por meio de padrões arquiteturais de fontes de dados. Fornece adaptadores de dados para os gerenciadores de dados para que possa construir os modelos das aplicações, Segundo Lisboa (2010).

2.2.4 Internacionalização e localização

A internacionalização e localização permitem que suas aplicações tenham suporte a outros idiomas, além do que foi desenvolvido. O ZF vem com componentes capazes de realizar tais funcionalidades.

2.2.5 Autenticação, autorização e gerenciamento de sessão

Em aplicações *web* existem dados que nem sempre todos os usuários tem a permissão de manipular, isso é customizado de forma protegida para que o acesso ocorra somente aos usuários autorizados.

3. Sistema Gerenciador de Eventos (SGE)

A Universidade do Planalto Catarinense, através da Pró-Reitoria de Pesquisa, Extensão e Pós-Graduação (PROPEPG) solicitou o desenvolvimento de um sistema gerenciador de eventos, que possibilite sua utilização na Mostra Científica e outros eventos da UNIPLAC, desta forma, deve possibilitar parametrização de forma a deixar o sistema adaptável. O sistema deve possibilitar o cadastro de um evento, informando dados básicos como nome e data. Alguns eventos permitirão a submissão de trabalhos devendo ser informadas as datas para submissão dos mesmos. Essas submissões podem ser organizadas por grupos de trabalhos (GT). O sistema ainda deve permitir o cadastro de ouvintes no evento. Alguns eventos possuem uma programação única, e outros podem possuir GT ou Oficinas a serem escolhidas pelo participante. Alguns eventos são cobrados taxa de inscrição, devendo este ser cobrado via boleto bancário. Ao final do evento deve ser possível gerar relatório de presença dos ouvintes, informando a porcentagem de participação. Também deve possuir relatório para controle de trabalhos inscritos e ouvintes no evento. O relatório dos inscritos deve ser organizado por evento, por oficina, também possibilitando a consulta de inscrições pagas e não pagas. O sistema deve ser disponibilizado através da Internet e ainda deve manter um histórico de informações dos eventos passados. O sistema deve estar acessível em diferentes tipos de navegadores.

3.1 Tecnologias e Ferramentas

No quadro 2 estão relacionadas as tecnologias utilizadas no desenvolvimento do SGE – Sistema Gerenciador de Eventos.

Quadro 2. Tecnologias utilizadas

Tecnologia	Objetivo/Descrição
PHP – Versão 5.3.5	É uma linguagem de script amplamente utilizada de propósito geral que é especialmente adequado para o desenvolvimento Web que pode ser incorporado com o HTML (PHP, 2012).
CSS	*Cascading Style Sheets* (CSS) é uma "folha de estilo" composta por "camadas" e utilizada para definir a apresentação (aparência) em páginas da internet que adotam para o seu desenvolvimento linguagens de marcação (como XML, HTML e XHTML) (TECMUNDO, 2009 e DEITEL e DEITEL, 2008).
MySQL – Versão 5.5.8	O MySQL é um sistema de gerenciamento de banco de dados (SGBD), que utiliza a linguagem SQL (Linguagem de Consulta Estruturada, do inglês *Structured Query Language*) como interface (MYSQL, 2012).
XHTML	É uma reformulação da linguagem de marcação HTML baseada em XML. Tem com objetivo a exibição de páginas da *Web* em diversos dispositivos (DEITEL e DEITEL, 2008).

No quadro 3 são apresentadas as ferramentas utilizadas no desenvolvimento do SGE – Sistema Gerenciador de Eventos.

Quadro 3. Ferramentas utilizadas

Ferramentas	Objetivo/Descrição
WampServer – Versão 2.1	É um ambiente de desenvolvimento *Web* para Windows. Ele permite que você crie aplicações *Web* com Apache2, PHP e banco de dados MySQL. Paralelamente, com o PhpMyAdmin permite gerenciar facilmente seus banco de dados (WAMPSERVER, 2012).
PhpMyAdmin – Versão 3.3.9	É uma ferramenta de software livre desenvolvido em PHP, destinado a administrar o MySQL. Suporta uma ampla gama de operações com o MySQL. As operações mais utilizadas pelos usuários são suportadas pela interface (gerenciar banco de dados, tabelas, campos, permissões, etc) (PHPMYADMIN, 2012).
Apache – Versão 2.2.17	É um servidor *Web* responsável por disponibilizar as páginas e todos os recursos de um *site* ou sistema *Web* (APACHE, 2012).
Zend Tool	É uma estrutura para expor funcionalidades comuns, tais como a criação de projetos, geração de código, índices de pesquisa e muito mais (ZEND TECHNOLOGIES, 2012).
Eclipse PDT 3.6.0 - Helios	Ferramenta usada para a criação de aplicações Java. Mas graças a sua arquitetura baseada em *plugins* permite a customização para outras linguagens (LISBOA, 2009).

3.2 Implementação

3.2.1 Criação do projeto

A criação de um projeto ZF pode ser facilitada usando a ferramenta *zend tool*, através da linha de comando do *Windows*, como demonstra a figura 5.

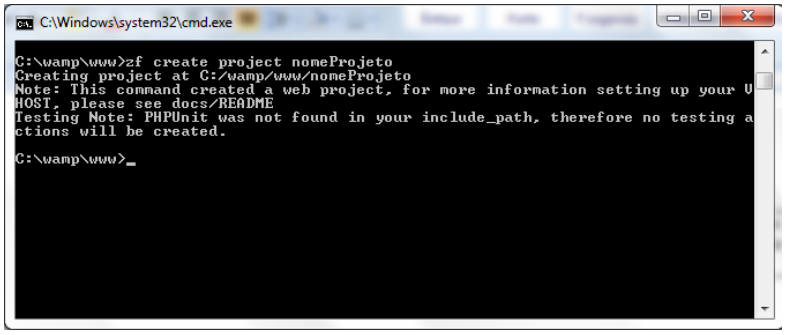

Figura 2 – Criação projeto ZF com zend tool [ZEND, 2011]

Automaticamente após este processo a *zend tool* cria a estrutura de arquivos padrão usada em projetos ZF.

3.2.2 Estrutura de arquivos

Para verificar o projeto criado pela zend tool pode ser utilizada a ferramenta Eclipse PDT, nela é possível acessar toda a estrutura de arquivos padrão para o projeto.

A figura 3 demonstra toda a estrutura de arquivos da aplicação. No diretório Application, constam todos os arquivos referentes ao aplicativo, ou seja, toda a programação não visualizável externamente. Ainda neste diretório está armazenada toda a estrutura MVC.

No diretório public contém arquivos visíveis externamente, como arquivos CSS, Javascript, imagens. Resumindo, somente este diretório será acessível ao usuário.

O diretório tests basicamente contém informações que correspondem a testes na fase de produção do sistema.

O diretório docs recebe a documentação gerada. Já o diretório library armazena as bibliotecas externas e o próprio framework.

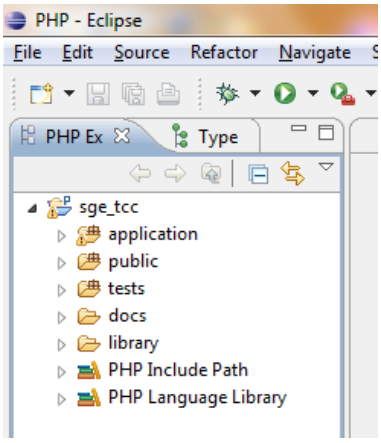

Figura 3. Estrutura de arquivos Eclipse [ECLIPSE, 2012]

3.2.3 Layout

A interface do SGE é padronizada, assim todas as páginas fazem o uso da classe *layout.php*, onde o *layout* padrão foi definido. Os elementos de apresentação como cores, fontes, margens, posições são definidos em um documento *CSS (Cascading Style Sheet)* externo, chamado *style.css.*

Ao longo do desenvolvimento do *layout* foram realizados testes em diferentes navegadores, para ter a garantia de que o SGE seria exibido sem problemas nos principais navegadores atualmente, Microsoft Internet Explorer, Google Chrome e Mozilla Firefox.

3.2.4 Conexão com banco de dados

A conexão ao banco de dados utilizando o ZF é realizada através da configuração do arquivo *application.ini*, o qual é gerado na criação do projeto através da *zend tool*.

3.2.5 Model

Toda a manipulação de banco de dados no ZF é realizada nos *models*, uma das vantagens do ZF que está manipulação é realizada por meio de objetos, assim se for necessário mudar a plataforma de banco de dados, basta alterar o adaptador no arquivo de configuração, como visto anteriormente.

3.2.6 Forms

A implementação de formulários utilizando ZF, torna-se simplificada com o uso do componente *Zend_Form*. Com este componente a manipulação de formulários se torna facilitada. O *Zend_Form* cumpre alguns objetivos como filtragem e validação da entrada de elementos, ordenação de elementos entre outros. Para que possa cumprir os seus objetivos o *Zend_Form* aciona outros elementos do ZF como *Zend_Validate*, *Zend_Filter*, *Zend_Loader* entre outros.

A figura 4 mostra a visualização do formulário de cadastro de eventos do SGE.

Figura 4. Visualização formulário de eventos [SUPPI, 2012]

3.2.7 Controllers

Os *controllers* no desenvolvimento de aplicações ZF utilizando MVC tem a função de receber as entradas dos usuários, manipular os *models* e fazer a atualizações das *views* de forma apropriada.

Os *controllers* são constituídos por *actions*, que representa a lógica de negocia de cada página, toda *action* que é criada deve ter um arquivo *view*, onde serão apresentadas as informações processadas.

3.2.8 Views

As *views* dentro da implementação MVC do ZF, tem a única função de apresentar a parte visual da aplicação aos usuários.

3.3 Controle de acesso

3.3.1 Autenticação

A figura 5 mostra a tela inicial do SGE, onde o usuário poderá realizar a sua entrada no sistema através de um *login* e uma senha. Caso o *login* e a senha sejam inválidos, é apresentada uma mensagem de usuário ou senha inválido, assim o usuário deverá tentar novamente.

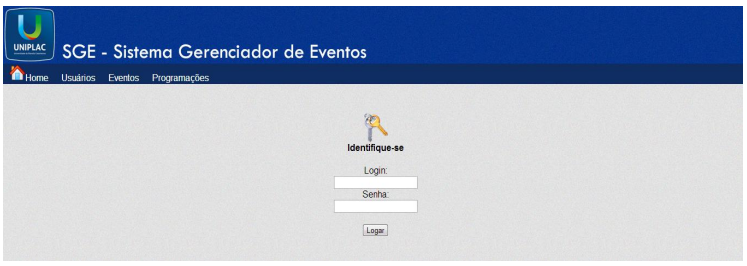

Figura 5. Visualização página inicial do sistema [SUPPI, 2012]

3.3.2 Autorização

A autorização faz a verificação das permissões do usuário que está autenticado no sistema, e tendo como base estas permissões, permitir ou bloquear o acesso do usuário a determinados recursos ou páginas da aplicação.

No caso do SGE são 5 os tipos de usuários, assim como mostra o quadro 31.

Quadro 4. Permissões de usuários

Tipo de usuário	**Permissões**
Administrador	O administrador tem o controle total do sistema podendo realizar qualquer tipo de operação.
Gerente de Conferência	O gerente de conferência tem quase todos os controles do administrador, pois não pode cadastrar novos eventos.
Avaliador	O avaliador tem como função avaliar os trabalhos submetidos pelos autores.
Autor	O autor é um usuário cadastrado no sistema que tenha permissão de submissão de trabalhos.
Ouvinte	O ouvinte deve ser um usuário cadastrado no sistema que poderá fazer a inscrição em algum trabalho de um evento.

4. Análise do zend framework

A decisão por usar o ZF foi pela proposta que o framework passa aos seus utilizadores, que é a criação de projetos bem estruturados juntamente com redução do tempo de desenvolvimento melhorando a qualidade das aplicações *web*.

A instalação do ZF não é algo complicado de ser realizado, é preciso somente fazer o seu download e setá-lo nas variáveis de ambiente do sistema operacional. A criação de projetos ZF é feito por uma ferramenta própria oferecida pelo próprio *framework*, a *Zend Tool*, onde são passadas instruções de criação de projetos, *controllers*, *views*, *models*, *actions* através da linha de comando do próprio *prompt* de comando, do Windows. Durante o desenvolvimento vários componentes foram de excelente utilidade, como é o caso do *Zend_Form*, que oferece uma criação de formulários de forma bastante fácil e intuitiva, oferecendo validadores e filtros que podem ser implementados com poucas linhas de código. Outro componente muito útil e que facilita o desenvolvimento é o *Zend_Auth*, com ele é possível criar um sistema de *login* de forma rápida e segura. Ainda existe componente de *layout* do ZF, que em apenas uma classe é possível definir a estrutura da interface da aplicação toda. A conexão ao banco de dados ocorre com muita facilidade, é apenas inclusa as linhas de conexão no arquivo de configuração do projeto criado com o uso da *Zend Tool*.

O ZF utiliza o padrão de projetos MVC, com isso a experiência com esse tipo de projeto no início dificultou um pouco o aprendizado, por falta de prática, mas com o passar do tempo e o convívio com este tipo de padrão se tornou um método mais útil e organizado para o desenvolvimento da aplicação.

O ZF possui várias vantagens, entre elas a sua gama de componentes de baixo acoplamento, podendo utilizar somente os componentes que realmente interessam para o desenvolvimento da aplicação. Outra vantagem do ZF é a sua documentação completa disponível em língua inglesa, além de livros que ajudam no aprendizado em outras linguagens. Por ter uma grande quantidade de componentes a sua total aprendizagem necessitaria de um tempo mais prolongado de estudo.

No geral o *framework* cumpre o que promete na facilitação e na agilidade do desenvolvimento de projetos *web* em PHP com MVC, com componentes de fácil aprendizagem e utilização.

5. Considerações Finais

A *Web* 2.0 está cada dia mais presente no dia a dia dos internautas e dos desenvolvedores *Web*. Entretanto, o desenvolvimento dos sistemas *Web* se torna um desafio para os desenvolvedores, onde a dinamicidade das aplicações exigem uma constante atualização dos desenvolvedores, exigindo ferramentas que agilizem o processo de desenvolvimento.

Este artigo teve como objetivo explorar o *framework Zend* aplicado no desenvolvimento de sistemas, demonstrando algumas características através de um estudo de caso e descrição de sua modelagem e especificações. Para tal, foram estudados os componentes necessários para o desenvolvimento, características e arquitetura do *framework*.

Referência Bibliográfica

ALLEN, R.; LO, N.; BROWN, S. **Zend Framework em Ação**. São Paulo: Alta Books, 2009.

APACHE. **Site oficial do Apache**, 2012. Disponível em: <http://www.apache.org/>. Acesso em: 10 set. 2012.

CAKE SOFTWARE FOUNDATION, Inc. **Site da Comunidade de desenvolvimento do framework CakePHP**. Disponível em: <http://cakephp.org/>. Acesso em: 27 mar. 2012.

CODEIGNITER, Inc. **Site oficial do framework CodeIgniter**. Disponível em: <http://codeigniter.com/ >. Acesso em: 29 mar. 2012.

DEITEL, P.J.; DEITEL, H.M. **Ajax, Rich Internet Applications e desenvolvimento Web para programadores.** Trad. Célia Taniwaki e Daniel Vieira. São Paulo: Pearson Prentice Hall, 2008.

ECLIPSE, Inc. **Site oficial da ferramenta de desenvolvimento Ecplise**. Disponível em: < http://www.eclipse.org/>. Acesso em: 10 jun. 2012.

JOBSTRAIBIZER, F. **Guia Profissional PHP**. São Paulo: Digerati Books, 2009.

LISBOA, F. G. D. S. **Zend Framework Componentes Poderosos para PHP**. São Paulo: Novatec, 2009.

LISBOA, F. G. D. S. **Criando Aplicações PHP com Zend e Dojo** .São Paulo: Novatec, 2010.

LOTAR, A. **Programando com ASP.NET MVC**. São Paulo: Novatec, 2011.

MINETTO, E. L. **Frameworks para Desenvolvimento em PHP**. São Paulo: Novatec, 2007.

MYSQL. **Site oficial do MySQL**, 2012. Disponível em: <http://www.mysql.com/>. Acesso em: 10 set. 2012.

PAMPLONA, V. **Introdução ao Apache Lucene**, 2009. Disponível em: <http://vitorpamplona.com/wiki/ Introdu%C3%A7%C3%A3o%20ao%20 Apache%20Lucene>. Acesso em: 17 abr. 2012.

PHP Frameworks. **PHP Frameworks**, 2007. Disponível em: <http://www.phpframeworks.com/>. Acesso em: 26 mar. 2012.

PHP. **Site oficial do PHP**, 2012. Disponível em: <http://www.php.net/>. Acesso em: 10 set. 2012.

PHPMYADMIN. **Site Oficial do phpMyAdmin**, 2012. Disponível em: <http://www.phpmyadmin.net>. Acesso em: 10 set. 2012.

SUPPI, L.F.P. **SGE: Sistema de gerenciamento de eventos**. 2012. Trabalho de Conclusão de Curso (Bacharelado em Sistemas de Informação), Universidade do Planalto Catarinense - UNIPLAC, Lages.

SYMFONY, Inc. **Site oficial do projeto framework Symfony**. Disponível em: <http://www.symfony-project.org/ >. Acesso em: 29 mar. 2012.

TECMUNDO. O que é CSS? **Site Tecmundo**, 2009. Disponível em: <http://www.tecmundo.com.br/programacao/2705-o-que-e-css-.htm>. Acesso em: 10 set. 2012.

WAMPSERVEER. **Site oficial do WampServer**, 2012. Disponível em: <http://www.wampserver.com>. Acesso em: 10 set. 2012.

ZEND TECHNOLOGIES, Inc. **Site Oficial do Zend Framework**. Disponível em: <http://framework.zend.com/ >. Acesso em: 29 mar. 2012.

ZEND TECHNOLOGIES, Inc. **Zend Framework versão 1.11.11**, 2011. Disponível em: < http:// framework.zend.com >. Acesso em: 10 fev. 2012.

ZEND TECHNOLOGIES, Inc. **Zend Framework: guia de referência do programador**, 2009. Disponível em: < http://diariodecodigos.info/zend/ptBR/>. Acesso em: 10 mar. 2012.