UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ – UTFPR
CURSO SUPERIOR DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

PABLO HENRIQUE SILVA

ESTUDO E IMPLEMENTAÇÃO DE WEB SERVICES UTILIZANDO ZEND FRAMEWORK

TRABALHO DE DIPLOMAÇÃO

MEDIANEIRA
2011

PABLO HENRIQUE SILVA

ESTUDO E IMPLEMENTAÇÃO DE WEB SERVICES UTILIZANDO ZEND FRAMEWORK

Trabalho de Diplomação apresentado à disciplina de Trabalho de Diplomação, do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas – CSTADS – da Universidade Tecnológica Federal do Paraná – UTFPR, como requisito parcial para obtenção do título de Tecnólogo.

Orientador: Prof. Diego Stiehl.

MEDIANEIRA

2011

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Diretoria de Graduação e Educação Profissional
Coordenação do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas

TERMO DE APROVAÇÃO

ESTUDO E IMPLEMENTAÇÃO DE WEB SERVICES UTILIZANDO ZEND FRAMEWORK

Por

Pablo Henrique Silva

Este Trabalho de Diplomação (TD) foi apresentado às 14:00 h do dia 18 de novembro de 2011 como requisito parcial para a obtenção do título de Tecnólogo no Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, da Universidade Tecnológica Federal do Paraná, *Campus* Medianeira. O candidato foi argüido pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado.

Prof. Diego Stiehl
UTFPR – *Campus* Medianeira
(Orientador)

Msc. Fernando Schutz
UTFPR – *Campus* Medianeira
(Convidado)

Prof. Marcio Angelo Matte
UTFPR – *Campus* Medianeira
(Convidado)

Msc. Juliano Rodrigo Lamb
UTFPR – *Campus* Medianeira
(Responsável pelas atividades de TCC)

AGRADECIMENTOS

Agradeço em primeiro lugar a minha família que sempre me apoiou e incentivou durante os momentos mais difíceis e deram conselhos que me fizeram superar os obstáculos e seguir em frente.

Agradeço também a todos os meus amigos com os quais eu passei muito tempo, e que de uma forma ou outra contribuíram para o meu aprendizado.

E também não poderia deixar de lembrar-se de todos os professores que participaram na minha caminhada pela UTFPR, com os quais eu aprendi muito. Agradeço também ao meu orientador Diego Stiehl que sempre esteve me apoiando e o qual foi de extrema importância para a conclusão desse projeto.

“Intenção sem ação é ilusão. Ouse fazer e o poder lhe será dado.” (RIBEIRO, Lair).

RESUMO

SILVA, H. Pablo. Estudo e Implementação de web services utilizando Zend Framework. Trabalho de conclusão de curso (Tecnologia em Análise e Desenvolvimento de Sistemas), Universidade Tecnológica Federal do Paraná. Medianeira 2011.

Juntamente com o crescimento da busca por soluções ágeis e produtivas é que começaram a surgir as primeiras ferramentas conhecidas como *frameworks* Este trabalho tem como foco demonstrar de maneira teórica e a utilização do Zend Framework no desenvolvimento de *web service*, apontando seus benefícios no desenvolvimento de aplicações web MVC (*Model-View-Controller*) em PHP 5.

Palavras-chave: Zend Framework. Web Services. MVC. PHP.

ABSTRACT

SILVA, H. Pablo. Estudo e Implementação de web services utilizando Zend Framework. Trabalho de conclusão de curso (Tecnologia em Análise e Desenvolvimento de Sistemas), Universidade Tecnológica Federal do Paraná. Medianeira 2011.

Along with the growth of the search for solutions that are agile and productive began to emerge the first tools known as frameworks This paper focuses on show in a theoretical way and use the Zend Framework in the development of web service, pointing out its benefits in the development of web applications MVC (Model-View-Controller) on PHP 5.

Keywords: Zend Framework. Web Services. MVC. PHP.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE SIGLAS

ACL	-	*Access Control List*
AJAX	-	*Asynchronous JavaScript and XML*
API	-	*Application programming interface*
CGI	-	Common Gateway Interface
CRUD	-	*Create, read, update and delete*
DAO	-	*Data Access Object*
GPL	-	*General Public License*
JSON	-	*Javascript Object Notation*
HTML	-	*HyperText Markup Language*
HTTP	-	*Hypertext Transfer Protocol*
IMAP	-	*Internet Message Access Protocol*
MVC	-	*Model-View-Controller*
NNTP	-	*Network News Transfer Protocol*
OOP	-	*Object-oriented programming*
PDF	-	*Portable document format*
PHP	-	*Hypertext Preprocessor*
PHP/FI	-	*Hypertext Preprocessor/Forms Interpreter*
POP3	-	Post Office Protocol – Version 3
RAD	-	*Rapid Application Development*
RBACL	-	*Role Based Access Control List*
SNMP	-	*Simple Network Management Protocol*
SOAP	-	*Simple Object Access Protocol*
SQL	-	*Structured Query Language*
TI	-	Tecnologia da Informação

UDDI	-	*Universal Description, Discovery and Integration*
URI	-	*Uniform Resource Identifier*
URL	-	*Uniform Resource Locator*
XML	-	*Extensible Markup Language*
ZF	-	*Zend Framework*
WEB	-	*World Wide Web*
WSDL	-	*Web Service Description Language*
W3C	-	*Word Wide Web Consortium*

SUMÁRIO

1 INTRODUÇÃO

Muito se fala sobre agilidade de processos, produtividade, mas segundo Laurindo (2002), e é no ramo de TI que essa busca tem aumentado rapidamente, o que faz com que o profissional esteja sempre buscando novas soluções e novas maneiras de estar capacitado para a crescente demanda de mercado e serviços.

Juntamente com a busca por soluções ágeis e produtivas é que começaram a surgir às primeiras ferramentas com estas características, também conhecidas como *frameworks*. Segundo Elton (2007), estes além de facilitarem a vida dos programadores, também trazem um ótimo ganho de rendimento no desenvolvimento de aplicações.

A área de programação foi ficando mais robusta e assim surgiu a tecnologia dos *web services*, que, segundo as definições da W3C (2011), consistem em uma solução utilizada na integração de sistemas e na comunicação entre aplicações diferentes. Essa tecnologia possibilitou que novas aplicações pudessem se comunicar com outras já existentes, utilizando-se assim dos seus serviços, e que sistemas desenvolvidos em diferentes plataformas pudessem se comunicar entre si, trazendo com isso o aumento da produtividade e livrando o desenvolvedor de se preocupar com serviços que já estão prontos em algum lugar na *Web*.

As linguagens de programação atuais oferecem vários recursos para que os programadores desenvolvam suas aplicações. Atualmente, existem inúmeras linguagens de programação, cada uma com as suas características. Para Dextra (2011), o PHP (*Hypertext Preprocessor*) é uma das linguagens que tem se destacado no suporte ao desenvolvimento de aplicações Web, pelo fato de ser de fácil aprendizado. Atualmente, muitas aplicações *Web* existentes utilizam o PHP como linguagem de desenvolvimento. Para Melo & Nascimento (2007), “o PHP é uma linguagem de script de código aberto que tem como objetivo a geração de conteúdo dinâmico para páginas de Internet”.

Com a inclusão do suporte à orientação a objetos a partir da versão 4.0, o PHP se tornou a ferramenta ideal para tal desenvolvimento.

1.1 OBJETIVO GERAL

Desenvolver um referencial teórico sobre o Zend Framework e sua utilização no desenvolvimento de web services.

1.2 OBJETIVOS ESPECÍFICOS

Os principais objetivos específicos são:

- Desenvolver um referencial teórico sobre a arquitetura, o modelo de programação e funcionamento do Zend Framework;
- Desenvolver uma aplicação exemplo que demonstre as principais características do Zend Framework, bem como sua utilização no desenvolvimento de *web services*.

1.3 JUSTIFICATIVA

Segundo Lisboa (2011), o desenvolvimento de sistemas consiste, basicamente, em criar um conjunto de atividades, parcialmente ordenadas, com a finalidade de obter um produto final, porém o maior esforço não está na criação, e sim na manutenção. As aplicações tornam-se cada vez mais complexas, e os requisitos dos clientes alteram-se muitas vezes, antes da conclusão do projeto. É preciso uma estrutura que permita a reutilização de código-fonte e o desenvolvimento simultâneo de partes do sistema, e se possível desvincular a aplicação do banco de dados, de forma que ela possa ser trocada sem causar nenhum (ou o mínimo de) impacto.

Para ALLEN, LO e BROWN (2009), o Zend Framework (ZF), vem para solucionar esses problemas com a proposta de criar uma arquitetura flexível e que permita o desenvolvimento de aplicações *Web* em PHP 5 com código reutilizável e mais fácil de manter. Também possui múltiplos formatos para *web services*, sendo apontado para ser o principal consumidor e publicador de serviços web.

A arquitetura do ZF permite que seus desenvolvedores reutilizem os seus componentes quando e onde eles fizerem sentido em suas aplicações sem requerer outros componentes ZF além das referencias mínimas, componentes estes comuns em aplicações web.

Sendo assim, juntando a flexibilidade do ZF com a agilidade e a eficiência de um *web service*, tem-se uma união muito interessante, e é com base nesta que se faz necessário um estudo sobre essa tecnologia.

1.4 ESTRUTURA DO TRABALHO

O presente trabalho possui cinco capítulos, sendo que o primeiro apresenta a introdução ao tema do estudo, os objetivos do trabalho, bem como a estrutura do mesmo.

O Capítulo 2 apresenta o referencial teórico, focalizando os principais conceitos de PHP e *web service* e os componentes necessários para a utilização do ZF.

O Capítulo 3 abrange o funcionamento do Zend Framework. A estrutura e as classes utilizadas são citadas e exemplificadas.

No Capítulo 4 são apresentadas as considerações finais do estudo e os possíveis desdobramentos de continuidade de estudos.

Por fim, no último capítulo, estão as referências bibliográficas que fizeram parte do embasamento teórico do projeto.

2 REVISÃO BIBLIOGRÁFICA

Este capítulo tem como objetivo apresentar a linguagem PHP e o Zend Framework, que serão utilizados no desenvolvimento de um *web service*.

2.1 PHP

PHP é uma linguagem de programação de computadores interpretada, livre e muito utilizada para gerar conteúdo dinâmico na WEB (*World Wide Web*). A linguagem surgiu por volta de 1994, como um pacote de programas CGI (*Common Gateway Interface*) criado por Rasmus Lerdorf, com o nome *Personal Home Page Tools*, para substituir um conjunto de *scripts* Perl que este utilizava no desenvolvimento de sua página pessoal. Em 1997 foi lançado o novo pacote da linguagem com o nome de PHP/FI (*Hypertext Preprocessor/Forms Interpreter*), trazendo a ferramenta *Forms Interpreter*: a qual consiste em um interpretador de comandos SQL(*Structured Query Language*) (EBECOM, 2010).

Mais tarde, Zeev Suraski desenvolveu o interpretador do PHP 3 que contava com o primeiro recurso de orientação a objetos, que dava poder de alcançar alguns pacotes, tinha herança e dava aos desenvolvedores somente a possibilidade de implementar propriedades e métodos (EBECOM, 2010).

Pouco depois, Zeev e Andi Gutmans escreveram o PHP 4, abandonando por completo o PHP 3, dando mais poder à máquina da linguagem e maior número de recursos de orientação a objetos. O problema sério que apresentou o PHP 4 foi a criação de cópias de objetos, pois a linguagem ainda não trabalhava com referências (EBECOM, 2010).

O problema fora resolvido na versão atual do PHP, a versão 5. Nesta versão, caso deseje-se realizar a cópia de um objeto, o que realmente será copiado é sua referência, pois, caso haja alguma mudança na versão original do objeto,

todas as outras também sofrem a alteração, o que não acontecia na PHP 4 (EBECOM, 2010).

PHP é uma linguagem de programação de domínio específico, ou seja, seu escopo se estende a um campo de atuação que é o desenvolvimento web. Seu propósito principal é de implementar soluções web velozes, simples e eficientes (WEB24HORAS, 2010).

Segundo Dall'Oglio (2007) ,"o PHP é uma das linguagens mais utilizadas no mundo". Para o autor, a popularidade do PHP se deve à facilidade em criar aplicações dinâmicas com suporte à maioria dos bancos de dados existentes. O PHP tem um conjunto de funções utilizadas por meio de uma estrutura flexível de programação, que permitem desde a criação de simples portais até complexas aplicações de negócio .

Para Melo & Nascimento (2007), o PHP é um processador de hipertexto livre, de código aberto que tem como principal objetivo a geração de conteúdo dinâmico para páginas de Internet. O fato de ser executado no lado do servidor impede que o código fonte seja exibido ao usuário.

Por ser um *software* livre, o código fonte do PHP está disponível e pode ser alterado a qualquer momento. A linguagem é licenciada segundo o modelo GPL (*General Public License*).

Para Alberton (2008), o suporte nativo a um grande número de bases de dados é um dos fatores positivos da linguagem PHP. Comumente, quando se trata de aplicações Web, as bases de dados MySQL e PostgreSQL são as mais conhecidas.

Segundo Alberton (2008), podem-se destacar como principais vantagens da linguagem:

- Amplo suporte a manipulação de arquivos;
- Geração e manipulação de imagens;
- Criptografia de informações;
- Suporte nativo a vários protocolos.

Além das características já citadas anteriormente, Melo & Nascimento (2007) apontam algumas características que podem ajudar na escolha do PHP como plataforma de desenvolvimento. Entre elas, estão:

- Total independência de plataforma;
- Oferece um suporte consistente à Orientação a Objetos na versão 5;
- Possui uma curva de aprendizado reduzida para iniciantes;
- A sintaxe é semelhante a linguages já conhecidas, como C e Java;
- Grande número de servidores de hospedagem existentes;

Segundo Alberton (2008), por padrão, o PHP possui 2 delimitadores de código pré definidos:

- Utilização da tag padrão `<?php ?>`;
- Utilização da tag semelhante a JavaScript: `<script language="php"></script>`;

Além dos padrões, é possível habilitar no arquivo de configuração demais delimitador para a linguagem PHP, como `<??>` e `<%%>`, que servem para iniciar e terminar o bloco de instruções PHP (PHP.NET , 2011).

As variáveis em PHP são iniciadas pelo símbolo $, exceto quando são definidas variáveis constantes, e não há um tipo explicito de dado. A variável pode assumir um entre os 8 tipos de dados existentes, sendo eles: *boolean*, *integer*, *string, array, object, null e resource*. O nome da variável deve começar com uma letra ou *underscore*. A linguagem é *Case Sensitive*, diferenciando então letras maiúsculas e minúsculas.

Pode-se entender uma classe como um conjunto de variáveis e funções relacionadas, que tornam possível a criação de objetos. Uma vantagem da utilização é poder usufruir o recurso de encapsulamento de informação. Com o encapsulamento o usuário de uma classe não precisa saber como ela é implementada, bastando para a utilização conhecer a interface, ou seja, as funções disponíveis. O Quadro 1 exemplifica uma classe PHP, bem como acesso a seus métodos e atributos (INFOLINK , 2011).

Basicamente, qualquer coisa que pode ser feita por algum programa CGI pode ser feita também com PHP, como coletar dados de um formulário, gerar páginas dinamicamente ou enviar e receber *cookies*. PHP tem suporte a outros

serviços através de protocolos como IMAP (*Internet Message Access Protocol)*, SNMP (*Simple Network Management Protocol)*, NNTP (*Network News Transfer Protocol*), POP3 (*Post Office Protocol – Version 3*) e, logicamente, HTTP (*Hypertext Transfer Protocol)*. Ainda é possível abrir *sockets* e interagir com outros protocolos. (INFOLINK, 2011).

```php
<?php
Class Aluno{
//definição do atributo nome;
       private $nome;

      //definição dos métodos de acesso.

      public function getNome(){
            return $this->$nome;
      }

      public function setNome($nome){
            $this->$nome = $nome;
      }
}
//criação de objeto do tipo aluno.
 $aluno = new Aluno();

//atribuição de valor ao atributo nome através do método setNome()
$aluno->setNome("Pablo");

//Exibe o valor
echo $aluno->getNome();
?>
```

Quadro 1- Exemplo de classe em php.

O Quadro 2 representa a comparação entre PHP e as demais linguagens concorrentes.

Custos Comparativos				
Item	**ASP**	**Cold Fusion**	**JSP**	**PHP**
Desenvolvimento	US$ 0 – 480	US$ 395	US$ 0	US$ 0
Servidor	US$ 620	US$ 1.295	US$ 0 – 595	US$ 0
RDBMS	US$ 1.220 - 4220	US$ 0 - ~10.000	US$ 0 - ~10.000	US$ 0
Suporte de incidente	US$0 - 245	US$ 0 - 75	US$ 0 - 75	US$ 0

Quadro 2 - Comparativo de custos.

Fonte: Olivreiro (2011)

O PHP possui diversas variáveis de ambiente, como a "`$PHP_SELF`", por exemplo, que contém o nome e o pasta do próprio arquivo. Algumas outras contêm informações sobre o navegador do usuário, o servidor HTTP, a versão do PHP e diversas informações.

2.2 WEB SERVICES

Pode-se definir *web services* como uma forma simples e padronizada de utilizar a Internet para efetuar a integração entre sistemas heterogêneos, de forma bastante flexível, tornando possível a interação de sistemas em plataformas diferentes, e possibilitando uma aplicação invocar outra para efetuar tarefas simples ou complexas mesmo que ambas estejam escritas em linguagens diferentes e em aplicações distintas (PERPÉTUO e JAGIELLO, 2011).

Basicamente um *web service* funciona como uma página *Web*, com a diferença que ao invés de HTML, utiliza-se XML. Desta forma os dados podem ser descritos e o pacote da mensagem pode ser manipulado com grande facilidade tanto por quem envia, quanto por quem recebe.

Com base na definição do W3C (*Word Wide Web Consortium*), *web services* são aplicações auto contidas, que possuem interface baseadas em XML e que descrevem uma coleção de operações acessíveis através de rede, independentemente da tecnologia usada na implementação do serviço (W3C, 2009).

Para empresas, eles trazem eficiência na comunicação entre cadeias de produção e agilidade para os processos. Toda e qualquer comunicação entre os sistemas passa a ser dinâmica e principalmente segura, pois não há intervenção humana.

Os *web services* são identificados por um URI (*Uniform Resource Identifier*), descritos e definidos usando XML (*Extensible Markup Language*). Um dos vários motivos que os tornam atraentes é o fato de serem baseados em tecnologias

padrões, em particular XML e HTTP (*Hypertext Transfer Protocol*) (PERPÉTUO e JAGIELLO, 2011).

A Figura 1 mostra a ordem hierárquica das tecnologias e protocolos utilizados por *web services*.

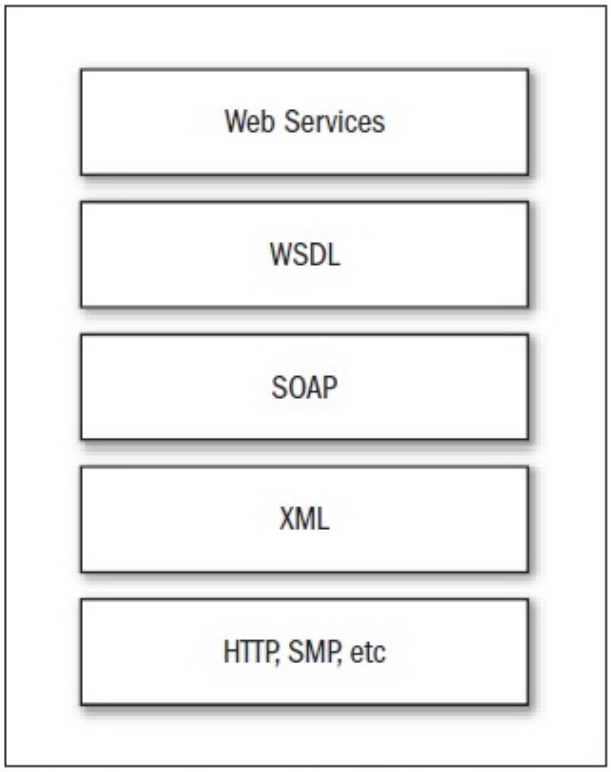


Figura 1-Tecnologias e Protocolos de um Web Service.

Fonte: Hanselman (2010)

2.2.1 XML

Extensible Markup Language (XML) é uma linguagem utilizada para definir formatos de documentos e mensagens e é a base da construção dos *web wervices*, pois fornece a descrição, o armazenamento, e o formato da transmissão para a troca de dados entre eles (W3C , 2011).

De acordo com Liberty e Kraley (2001), “o XML oferece um mecanismo independente de plataforma aceito globalmente para gerenciar, armazenar e comunicar informações”. XML é de fato um padrão já estabelecido pelas empresas no que diz respeito ao intercâmbio de dados na *Web*, prova disto é a afirmação de

Liberty e Kraley (2001) “a recomendação XML 1.0 do World Wide Web Consortium (W3C) é um padrão internacional que já foi endossado e adotado pela Microsoft, Netscape, Sun, IBM e vários outros fabricantes”.

XML também é extensível, ou seja, suas marcações podem ser criadas de acordo com a necessidade, diferentemente do HTML que possui um conjunto de tags fixas. De acordo com Silberschatz, Korth e Sudarshan (2006), “ao contrário da HTML, XML não prescreve o conjunto de marcações permitidas, e o conjunto pode ser escolhido conforme a necessidade por cada aplicação. Esse recurso é a chave para a principal função do XML na representação e troca de dados, enquanto HTML é usada principalmente para a formatação de documentos”.

Um documento XML nada mais é de que um arquivo texto com dados representados em um formato padrão (BONIATI; PADOIN, 2003) no qual é codificada toda a informação contida no documento. O Quadro 3 mostra um exemplo simples de documento XML.

```
<?xml version="1.0?>
    <Capitulo>
        <Titulo>Exemplo XML</Titulo>
        <Secao>
            <Titulo>Web Services</Titulo>
        </Secao>
    </Capitulo>
```

Quadro 3 - Exemplo simples de XML

2.2.2 WSDL

WSDL (*Web Service Description Language*) é uma linguagem de marcação utilizada para descrever um *web service*. Desenvolvida pela IBM e Microsoft, sendo que, a especificação WSDL 1.1 foi submetida a W3C em março de 2001, o documento WSDL descreve qual o serviço que o *web service* oferece como ele se comunica e onde ele pode ser encontrado (RIBEIRO, BRAGA e SOUSA, 2011).

O WSDL fornece um mecanismo estruturado para descrever as operações que um *web service* pode executar o formato das mensagens que pode processar os protocolos que suporta e o ponto de acesso de uma instância de um *web service*.

A descrição em WSDL de um *web service* fornece a especificação da interface deste, ou seja, o que cada serviço faz e como invocá-los. É baseado em XML, e é conceitualmente similar à Linguagem de Definição de Interface (IDL – *Interface Definition Language*) do CORBA (ROY; RAMANUJAN, 2001). A Figura 2 mostra a estrutura de um documento WSDL.

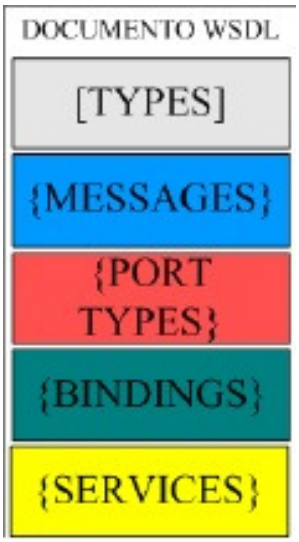


Figura 2 - Estrutura de um documento WSDL

Fonte: Machado (2011).

Esta descrição inclui detalhes como: definição dos tipos dos dados - *Types*, formatos das mensagens de entrada/saída - *Messages*, operações suportadas pelo serviço - *Port Types*, mapeamento de protocolos - *Bindings* e serviços - *Services* (ROY; RAMANUJAN, 2001). Uma vez definida a descrição do *web service* este é obrigado a cumprir o que está especificado em seu WSDL (MACHADO, 2011).

2.2.3 SOAP

SOAP (*Simple Object Access Protocol*) é um protocolo simples e leve baseado em XML que proporciona troca de informações encima do protocolo HTTP.

Em suma, é um protocolo para acessar *web service*. A arquitetura tem sido desenhada para ser independente de qualquer modelo particular de programa e de outras implementações específicas. Seus dois maiores objetivos são a simplicidade e extensibilidade (RIBEIRO, BRAGA e SOUSA, 2011).

O protocolo SOAP obedece a esses requisitos, pois trafega sobre o HTTP, que é suportado por todos os servidores e navegadores do mercado, com diferentes tecnologias e linguagens de programação.

Com base na definição do W3C (2011), este protocolo baseado em XML consiste de três partes: um envelope, que define o que está na mensagem e como processá-la, um conjunto de regras codificadas para expressar instâncias do tipos de dados definidos na aplicação e uma convenção para representar chamadas de procedimentos e respostas.

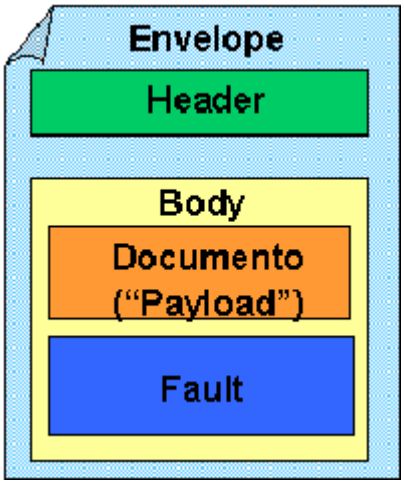


Figura 3 - Estrutura SOAP

Fonte: W3C (2011)

2.2.4 UDDI

Quando se constrói *web services*, esses serviços necessitam ser acessados em algum lugar na *Web* por uma aplicação-cliente. Uma forma de se acessar um *web service* é fazer com que a aplicação-cliente conheça o caminho do serviço, desta maneira caracterizando o modo estático de se localizar e acessar um serviço. Entretanto, quando a aplicação-cliente não detém, a priori, a localização de um *web*

service, esse, pode ser descoberto antes de ser acessado caracterizando o modo dinâmico de se descobrir a localização de um serviço (WATHIER, 2005).

UDDI é uma estrutura independente de plataforma para descrever serviços, descobrindo negócios e integração dos serviços de negócios usando a Internet. E usa WSDL para descrever interfaces de serviços web (W3SCHOOLS, 2011). Assim, é, portanto, uma parte crítica da pilha de 39 protocolos *web services*, habilitando usuários dos serviços a publicarem e descobrirem estes serviços.

A especificação UDDI define tipos de dados fundamentais, que incluem uma descrição da função do serviço, informações sobre o editor do serviço, os detalhes técnicos do serviço e API, e outros *metadados*. Esses tipos de dados são definidos em vários esquemas XML, que juntos formam o modelo de informações da base de registros UDDI. (UDDI-101, 2011) Eles incluem:

- Uma descrição da função de um serviço de negócio (o chamado businessService)
- Informações sobre a organização que publicou o serviço (businessEntity)
- Detalhes técnicos do serviço (bindingTemplate), incluindo uma referência à interface de programação do serviço ou API, e
- Vários outros atributos ou metadados como taxonomia, transportes e assinaturas digitais.

2.2.5 ARQUITETURA HETEROGÊNEA

A Figura 4 demonstra os três passos para a geração de um *web service*.

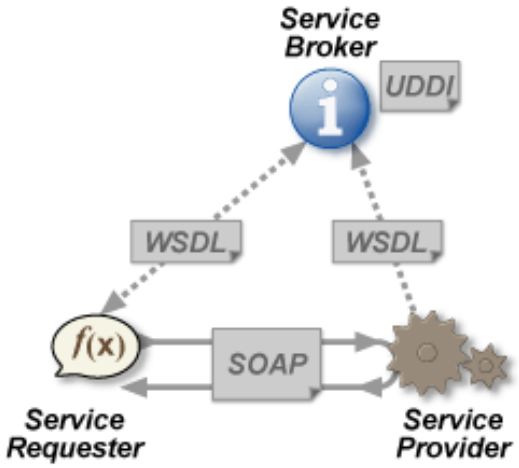


Figura 4-- Estrutura de um Web Service.

Fonte: W3C (2011)

O Service Provider é o responsável por fornecer o serviço. Em troca de requisições ele disponibiliza o retorno das informações, nessa troca é utilizada a tecnologia SOAP.

O Service Requester é responsável pelo consumo do serviço estando seus métodos especificados no Service Broker através do uso da tecnologia WSDL.

O Service Broker fica responsável por publicar e divulgar os serviços. O Service Provider disponibiliza as informações de seu web service ao Service Broker, o qual irá registrar este serviço em um diretório público e gerenciá-lo. Este diretório é o UDDI, que teoricamente deveria ser uma espécie de repositório aos consumidores de *web services*, mas na prática ainda é pouco usual.

2.3 ZEND FRAMEWORK

Zend Framework é um *framework* de código aberto para desenvolvedores de aplicativos *Web* com PHP 5. É totalmente orientado a objetos e sua estrutura de componentes é única, possuindo uma biblioteca de componentes independentes que quando combinados formam um extensível *framework* para aplicativos *web*. (ZEND, 2011)

Zend Framework foi concebido no início de 2005 enquanto muitos novos frameworks, tais como Ruby on Rails e Spring Framework, estavam ganhando popularidade na comunidade Web Development. (CLEYDSON, 2010).

Foi publicamente anunciado pela primeira vez na Zend Conference[1]. Ao mesmo tempo, nenhum *framework* amplamente usado tem sido disponibilizado para a comunidade PHP para preencher completamente necessidades de desenvolvimento *web* similares. Os projetistas do Zend Framework buscaram combinar características de uso final e *Rapid Application Development* (RAD) desses novos frameworks com a simplicidade, abertura e praticidade do mundo real que é altamente valorizada na comunidade PHP. (CLEYDSON, 2010).

Além de oferecer robustez e alto desempenho, o Zend Framework também fornece uma avançada arquitetura MVC que pode ser usada para estabelecer uma estrutura básica para as suas aplicações, e uma abstração de banco de dados que é muito simples de se usar. Outros componentes, tais como *Zend_auth* e *Zend_acl*, implementam autenticação e autorização via listas de controle de acesso (ACL).

Segundo ALLEN, LO e BROWN (2009), ZF fornece componentes individuais para muitos outros requisitos comuns no desenvolvimento de aplicações web, incluindo configuração de aplicações, filtragem/validação de dados fornecidos pelo usuário para segurança e integridade de dados, interfaces para funcionalidades AJAX, internacionalização, *data caching*, composição/entrega de e-mail, indexação e consulta no formato de busca Lucene, e também todas as bibliotecas do Google com muitos outros *web services* populares.

Por ter um projeto fracamente acoplado, seus componentes podem ser usados ao lado de componentes de terceiros. Qualquer que seja a necessidade é provável que algum componente do Zend Framework possa ser usado, reduzindo drasticamente o tempo de desenvolvimento com uma base totalmente testada. ALLEN, LO e BROWN (2009). Estes componentes são agrupados em seis categorias mostradas na Figura 5.

[1] É o maior encontro da Comunidade PHP e reúne desenvolvedores PHP e gerentes de TI de todo o mundo (ZENDCON, 2011).

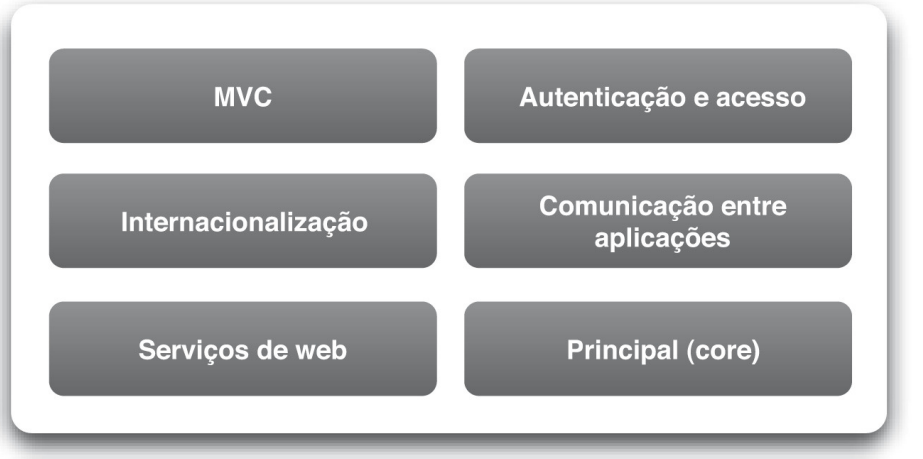


Figura 5 - As seis categorias de mais alto nível do ZF.

Fonte: Allen, Lo e Brown (2009).

Como já citado, uma característica fundamental da estrutura do ZF, é a facilidade de utilizar somente as partes necessárias para o desenvolvimento da aplicação, ou com outras bibliotecas independentes, podendo até mesmo utilizar componentes do ZF com outros *frameworks* para PHP como CakePHP ou CodeIgniter. O segredo está no fato de que cada componente do *framework* tem poucas dependências em relação aos demais. Isto permite introduzir componentes específicos do Zend Framework, tais como o *Zend_Search*, *Zend_Pdf* ou o *Zend_Cache* em seu projeto corrente sem ter de substituir todo o restante do código de seu projeto.

A partir da divisão das seis categorias de alto nível mostrada na Figura 5, obtém-se a possibilidade de categorizar os principais componentes do *framework*. A Figura 6 demonstra isso.

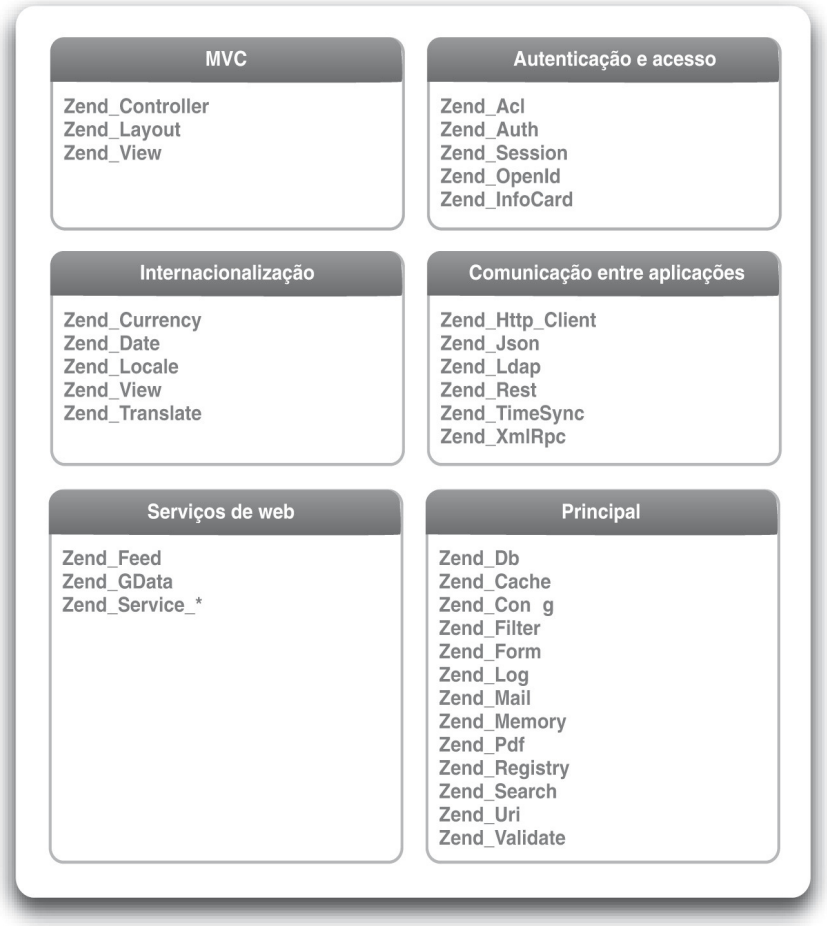


Figura 6 - Detalhamento das seis categorias de mais alto nível do ZF.

Fonte: Allen, Lo e Brown (2009).

Cada componente do *framework* contém várias classes, incluindo a classe principal a partir da qual o componente recebe o seu nome. Por exemplo, o componente *Zend_Config* contém a classe *Zend_Config*, juntamente com as classes *Zend_Config_Ini* e *Zend_Config_Xml*. Cada componente contém também várias outras classes que não estão listadas na Figura 6.

2.3.1 Componentes do MVC

Os componentes do MVC disponibilizam um completo sistema MVC para desenvolver aplicações que separam a lógica do negócio e os arquivos de controle dos *templates* da visão Figura 7.

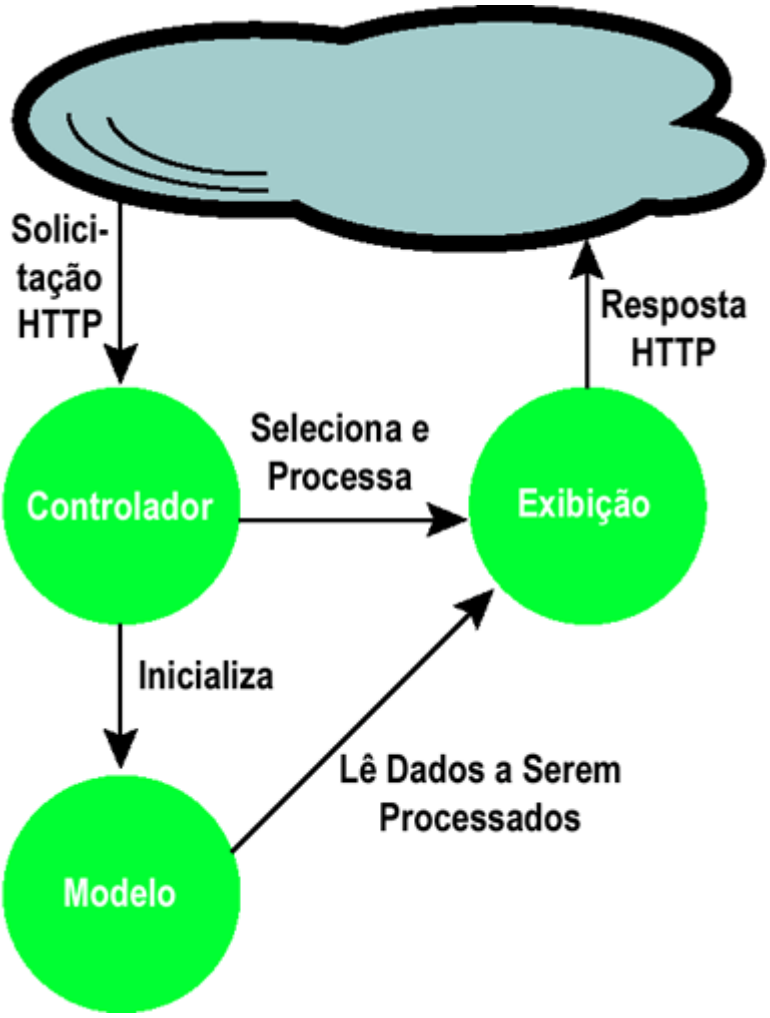


Figura 7- Componentes do MVC.

Fonte:Hanselman (2010).

- Modelo**:** é a *parte da aplicação onde se encontram as regras de negócio e comunicação com banco de dados.*
- Exibição*: é a parte onde se encontra a interface do usuário. Normalmente é criada baseada em um objeto modelo.*
- Controlador*: é a parte* que interage com o usuário. Trabalha em conjunto com o *model* e, por fim, seleciona a página ideal para a qual o usuário será redirecionado. Em uma aplicação MVC, as páginas (*views*) somente possuem informações; são os controladores (*controllers*) que manuseiam e respondem às solicitações do usuário (MICROSOFT, 2010c).

Também é composto pelo *Zend_Controller* (o controlador) e pelo *Zend_View* (a visão) com as classes *Zend_Db e Zend_Service* formando o modelo. A Figura 8 mostra o básico do sistema MVC do Zend Framework, usando o *Zend_Db* como modelo.

O grupo de classes do *Zend_Controller* oferece também um Controlador Frontal (*Front Controller design pattern*), padrão de design que despacha solicitações para as ações do controlador para que todo o processamento seja centralizado. Como se espera de um sistema completo suporta *plug-ins* em todos os níveis do processo e possuí pontos flexíveis embutidos permitindo mudar partes específicas do comportamento sem muito trabalho (ZENDOC, 2011).

O *Zend_View* é o sistema de *script* da visão, o qual oferece um sistema de *template* baseado em PHP e um sistema de *plug-in* auxiliar que permite a criação de código de apresentação reutilizável. Foi criado para permitir sobrescrita e atender a requisitos específicos, e até mesmo para usar um sistema totalmente diferente de *templates*, como o *Smarty*. Trabalhando em conjunto com o *Zend_View* ,o *Zend_Layout* oferece agregação de vários *scripts* de visão para construir uma página *web* completa.

Para compor a base do modelo dentro do sistema MVC, o Zend_Db_Table pode ser usado, pois implementa o padrão *Table Data Gateway*, modelo esse que oferece a lógica de negócios da aplicação. Já o *Zend_Db_Table* usa o *Zend_Db,* que disponibiliza um acesso independente de banco de dados, orientado a objetos, a diversos bancos de dados diferentes como PostgreSQL, MySQL, SQL Server, SQLite e Oracle.

O fluxo do MVC em uma aplicação Zend Framework usa um controlador frontal para processar a solicitação e delegar para um controlador de ação específico que usa modelos e visões para construir a resposta.

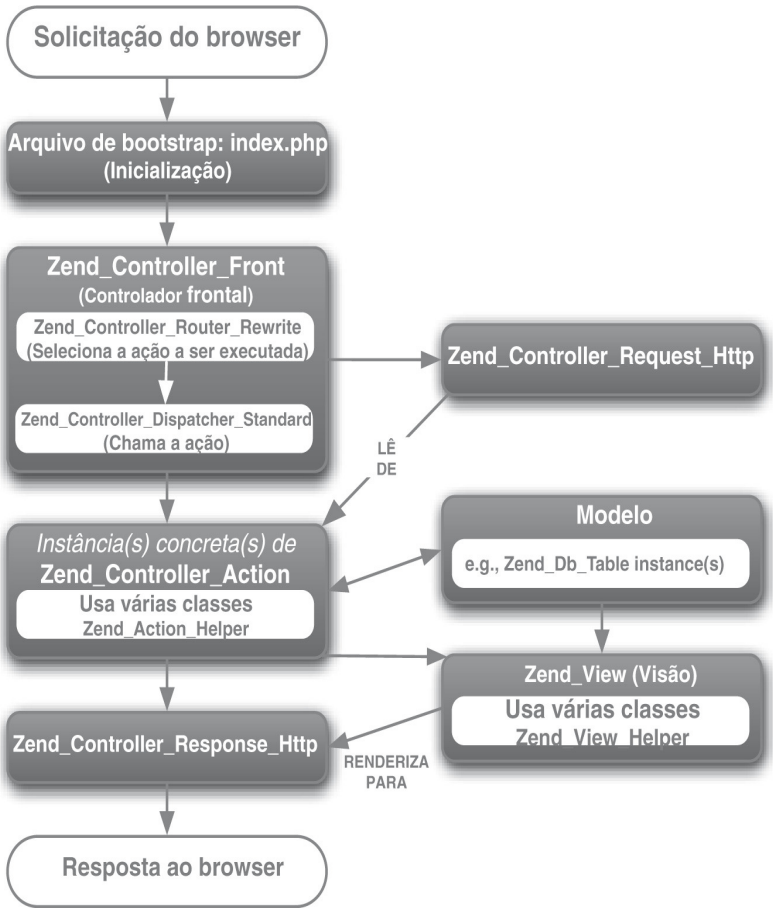


Figura 8 - Fluxo do MVC.

Fonte: Allen, Lo e Brown (2009).

Segundo ALLEN, LO e BROWN (2009), as classes MVC trabalham em conjunto com algumas das classes principais que formam o núcleo de uma aplicação completa. O *framework* por si mesmo não exige configuração, mas alguma configuração para sua aplicação será invariavelmente necessária. O *Zend_Config* permite que uma aplicação leia os dados de configuração a partir de *arrays* PHP ou de arquivos INI ou XML e ele inclui um sistema útil de herança para suportar diferentes parâmetros de configuração em diferentes servidores tais como de produção, de validação e de testes.

Todo desenvolvedor de PHP tem que levar em conta a segurança em suas aplicações, e para isso a validação de dados de entrada e filtragem constituem a chave para essa segurança (CASET, 2009). O *Zend_Filter* e o *Zend_Validade* estão disponíveis para ajudar a garantir que os dados de entrada são confiáveis e serem utilizados na aplicação.

A classe *Zend_Filter* oferece um conjunto de filtros que removem dados indesejados da entrada à medida que estes passam pelo filtro.

2.3.2 Componentes de Autenticação e Acesso

Em muitas aplicações não é preciso autenticar seus usuários, mas isto é um requisito comum. Segundo Amoroso (2009), autenticação é o processo de identificar um usuário, geralmente através de um *token* ou de uma identificação digital. O controle de acesso corresponde ao processo de decidir se o usuário autenticado tem permissão de acessar ou executar operações em um dado recurso, como em um registro de banco de dados.

Como existem dois processos, o Zend Framework oferece dois componentes separados: *Zend_Acl* e *Zend_Auth*. O primeiro é usado para identificar o usuário e é tipicamente utilizado em conjunto com o *Zend_Session*, que pode armazenar estas informações para solicitações em várias páginas. O segundo utiliza então o *token* de autenticação para disponibilizar acesso a informações privadas usando um sistema de controle de acesso baseado em papéis (RBACL, ou *Role Based Access Control List*) (ZENDOC, 2011).

Um papel pode ser considerado qualquer coisa que queira acessar algo que está sob a proteção do *Zend_acl*. Geralmente isto significa que um papel corresponde a um grupo de usuários que foi identificado através do *Zend_auth*. Um recurso é qualquer coisa que deve ser protegida. Geralmente é um registro em um banco de dados, mas poderia igualmente ser um arquivo de imagem armazenado no disco.

2.3.3 Componentes para Internacionalização

O Zend Framework oferece um conjunto rico de funcionalidades que permitem a localização de uma aplicação para que ela fique de acordo com os usuários desejados. Isto cobre pequenas questões, desde garantir que o símbolo correto da moeda seja usado em todos os lugares, até dar suporte completo à mudança de todo o texto da página para o idioma correto. Rotinas de data e de hora também são disponibilizadas com uma interface simples orientada a objetos, assim

como os parâmetros para as diversas formas pelas quais um calendário pode ser apresentado.

O Zend Framework disponibiliza a classe *Zend_local*, que é responsável juntamente com *Zend_currency* e *Zend_measure*, por garantir que a linguagem e o idioma corretos sejam utilizados. O componente *Zend_translate* é responsável pela tradução propriamente dita do texto de um website para o idioma desejado (ZENDOC, 2011).

2.3.4 Componentes para Comunicação entre Aplicações

O Zend Framework oferece um componente para ler dados de outros *websites*. O *Zend_Http_Client* faz com que seja fácil coletar dados de outros websites e serviços e depois apresentá-los em seu site. Este componente funciona de forma bastante parecida com a extensão Curl do PHP[2], mas está implementado em PHP, e, deste modo, pode ser usado em situações em que Curl não estiver habilitado.

Para ALLEN, LO e BROWN (2009), quando você precisa se comunicar com outra aplicação usando HTTP, o formato mais comum de transferência é uma de duas variações do XML: XML-RPC e SOAP. O PHP5 contém um excelente suporte para SOAP embutido, e o Zend Framework disponibiliza o *Zend_XmlRpc_Client* para permitir o fácil processamento do XML-RPC. Mais recentemente, o protocolo JSON (*Javascript Object Notation*), que é mais leve, vem ganhando espaço, principalmente por causa da facilidade de processamento dentro do JavaScript de uma aplicação Ajax. O Zend_Json oferece uma solução elegante tanto para a criação quanto para a leitura de dados JSON.

[2] Biblioteca que permite conectar-se e comunicar-se com vários tipos diferentes de servidor, utilizando diferentes tipos de protocolos (PHP.NET, 2011).

2.3.5 Componentes dos Serviços de Web

O Zend Framework oferece um rico conjunto de funcionalidades que permitem o acesso a serviços oferecidos por outros fornecedores. Estes componentes cobrem *feeds* RSS[3] genéricos, juntamente com componentes específicos para trabalhar com as bibliotecas públicas do Yahoo, do Google! e da Amazon. O Zend_Feed oferece uma interface consistente para leitura de *feeds* nas várias versões RSS e *Atom* disponíveis, sem a necessidade de se preocupar com detalhes (ZEND, 2011).

O Yahoo, o Google, a Amazon e outros *websites* disponibilizaram APIs públicas para serviços on-line para estimular os desenvolvedores a criarem aplicações estendidas em cima do serviço principal. Para a Amazon, a API oferece acesso aos dados de seu site na expectativa de que a nova aplicação irá estimular as vendas. De modo semelhante, o Yahoo oferece acesso via API aos dados de fotos do Flickr para permitir serviços adicionais aos usuários do Flickr, tais como os serviços de impressão fornecidos pela moo.com. Os serviços tradicionais do Yahoo!, tais como buscas, notícias e imagens também estão disponíveis. O Zend Framework agrupa estes e vários outros componentes em um conjunto de classes cujo prefixo é *Zend_Service*, exemplo: *Zend_Service_Delicious, Zend_Service_Amazon, Zend_Service_SlideShare , Zend_Service_Simpy e Zend_Service_Yahoo (*ALLEN, LO e BROWN (2009).

O Google possui várias aplicações *on-line* que permitem acesso via API e que são suportadas pelo componente *Zend_Gdata*. Este oferece acesso às aplicações de *Blogger, Calendar, Base, YouTube* e *Code Search* do Google. Por questões de consistência, o componente *Zend_Gdata* disponibiliza os dados usando *Zend_Feed*, de modo que se pode processar um *feed* RSS, e com isso processar dados do Google Calendar.

---

[3] Os feeds RSS oferecem conteúdo Web ou resumos de conteúdo juntamente com os links para as versões completas de conteúdo e outros metadados. Esta informação é entregue como um arquivo XML chamado "RSS feed", "webfeed", "Atom" ou ainda canal RSS.(RSS-ESPECIFICATION, 2011).

2.3.6 Componentes Principais

Existe um conjunto de outros componentes disponibilizados pelo Zend Framework que não se enquadram facilmente em nenhuma outra categoria, e sendo assim, elas foram agrupadas na categoria principal. Esta variedade de componentes inclui classes para *caching*, buscas e criação de PDF.

O *Zend_Cache* oferece uma interface genérica e consistente para fazer cache de qualquer dado em diversos sistemas *back end*, tais como discos, bancos de dados ou até mesmo memória compartilhada. Esta flexibilidade garante que se possa começar com poucos dados no *Zend_Cache*, e, à medida que a carga de seu site aumentar, a solução de *caching* pode crescer junto para garantir que você tire o máximo proveito do *hardware* de seu servidor (ZENDOC, 2011).

Todo site moderno oferece a facilidade de busca, mas a maioria é tão ruim que os usuários do site preferem procurar via Google a usar o próprio sistema do site. O *Zend_Search_Lucene* é baseado no mecanismo de busca *Lucene* do *Apache* para Java, e oferece um sistema de busca de textos bem profissional que permitirá a seus usuários encontrar o que eles estiverem procurando. Como necessário a um bom sistema de buscas, *o Zend_Search_Lucene* suporta buscas com classificação (em que os melhores resultados ficam no topo), juntamente com um sistema bastante capacitado para consultas.

Para ALLEN, LO e BROWN (2009), outro componente principal é o *Zend_Pdf*, que cria arquivos PDF via programação. O PDF é um formato bastante portável para documentos a serem impressos. Pode-se controlar a posição de tudo na página com precisão em nível de pixels sem ter de se preocupar com diferenças no modo como os *browsers* mostram a página. O *Zend_Pdf* foi totalmente escrito em PHP e pode criar novos documentos PDF ou carregar arquivos existentes para edição.

O Zend Framework disponibiliza um componente de e-mail bem complexo, o *Zend_Email* permite o envio de e-mails em texto normal ou em HTML. Como em todos os componentes, a ênfase foi colocada na flexibilidade e em padrões criteriosos. Isto significa que o componente permite o envio de e-mails usando o protocolo SMTP ou por meio do comando padrão do PHP, *mail*(). Transportes adicionais podem ser inseridos facilmente no sistema escrevendo-se uma nova

classe que implemente Zend_Email_Transport_Interface. No envio de um e-mail, uma interface orientada a objetos é usada:

```
$mail = new Zend_Mail();

$mail->setBodyText('Enviando Email com Zend')

->setBodyHtml('Enviando <b> Email </b> Zend'!')

->setFrom('silvaa@tcc.com', 'Pablo Henrique')

->addTo('secco@tcc.com', 'Diego Stiehl')

->setSubject('Zend Framework')

->send();
```

Quadro 4-Exemplo de envio de email com Zend_Mail.

Mostra também o uso de uma interface fluente - cada função membro retorna a instância de um objeto, de modo que as funções possam ser encadeadas tornando o código mais legível. Interfaces fluentes são comumente usadas nas classes do Zend Framework para facilitar a utilização das classes.

Zend Framework também procura promover as melhores práticas de desenvolvimento web na comunidade PHP. Apropriadamente as sugestões são substituídas por padrões razoáveis de configuração que podem ser sobrescritos por cada requisito específico da aplicação ZF. (CLEYDSON, 2010).

O Zend Framework não é a única opção para organizar um *website* baseado em princípios do MVC; existem vários outros. Porém Zend Framework deve ser levado em consideração no momento da construção deste, pois possui algumas vantagens com relação aos demais (Figura 9).

Recurso	Zend Framework	CakePHP	CodeIgniter	Solar	Symfony
Tira o máximo de proveito do PHP5	Sim	Não	Não	Sim	Sim
Estrutura de diretórios pré-definida	Não (Estrutura opcional recomendada)	Sim	Sim	Não	Sim
Suporte internacionalização oficial	Sim	Em progresso para versão 1.2	Não	Sim	Sim
Scripts de linha de comando necessários para configurar o framework	Não	Não	Não	Não	Sim
Exige configuração	Sim (quantidade mínima)	Não	Não	Sim	Sim
ORM abrangente disponível	Não	Sim	Não	Não	Sim (Propel)
Boa documentação e tutoriais	Sim	Sim	Sim	Sim	Sim
Testes unitários para código disponíveis	Sim	Não	Não	Sim	Sim
Suporte da comunidade	Sim	Sim	Sim	Sim (algum)	Sim
Licença	BSD novo	MIT	Estilo BSD	BSD novo	MIT

Figura 9 - Comparativo entre os frameworks PHP mais conhecidos.

Fonte: **Allen, Lo e Brown (2009).**

A principal responsável pelo projeto 'Zend Framework' é a empresa Zend Technologies, mas existem muitas empresas contribuindo para a criação de componentes ou criação de características significantes no framework. Empresas como o Google, Microsoft e StrikeIreon tem participado de seu desenvolvimento fornecendo interfaces para *web services* e outras tecnologias que desejam fazer disponíveis no Zend Framework (Zend, 2011).

3 ESTUDO EXPERIMENTAL

Para demonstrar todo o processo de criação e a produtividade proporcionada através do uso do Zend Framework foi realizada uma ampla pesquisa bibliográfica, além de diversas consultas em sites especializados no assunto, a fim de dar subsídios suficientes à implementação das tecnologias em estudo.

3.1 FERRAMENTAS

Para o funcionamento e as configurações, foi utilizado o Zend Studio como ambiente de desenvolvimento, e o Zend Tool, que proporciona a criação de toda a estrutura do projeto, tal como seus *controllers* e *viewers*. O servidor de aplicação utilizado foi o Wamp *Server,* ele permite criar aplicações web com Apache, PHP e o banco de dados MySQL.

- *Apache 2.2.21*
- *Php 5.3.8*
- *Mysql 5.5.16*
- *PhpMyadmin 3.4.5*

Também foram utilizadas algumas funcionalidades do sistema operacional, como o *prompt* de comando, e as variáveis de ambiente.

3.2 FUNCIONAMENTO

Como pré-requisito para a criação do projeto através do Zend Tool, algumas variáveis de ambiente devem ser adicionadas. Deve-se fazer *download* gratuitamente do framework e descompactá-lo. Com os arquivos descompactados, estará disponível a pasta "*bin*" e a pasta "*library*". Estas pastas devem ser adicionadas no *path* das variáveis de ambiente, assim como a pasta "*bin*" do PHP respectivamente.

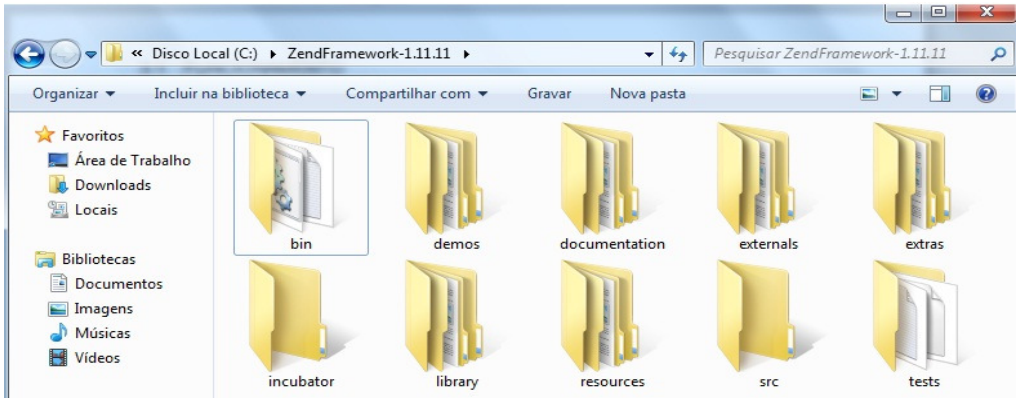


Figura 10 - Arquivos do Framework

A criação do projeto é feita de forma simples, e gera toda a estrutura de pastas já configuradas. Ela deve ser feita utilizando o Zend Tool através do console do Windows utilizando o comando "`zf create Project`" e o nome do projeto desejado.

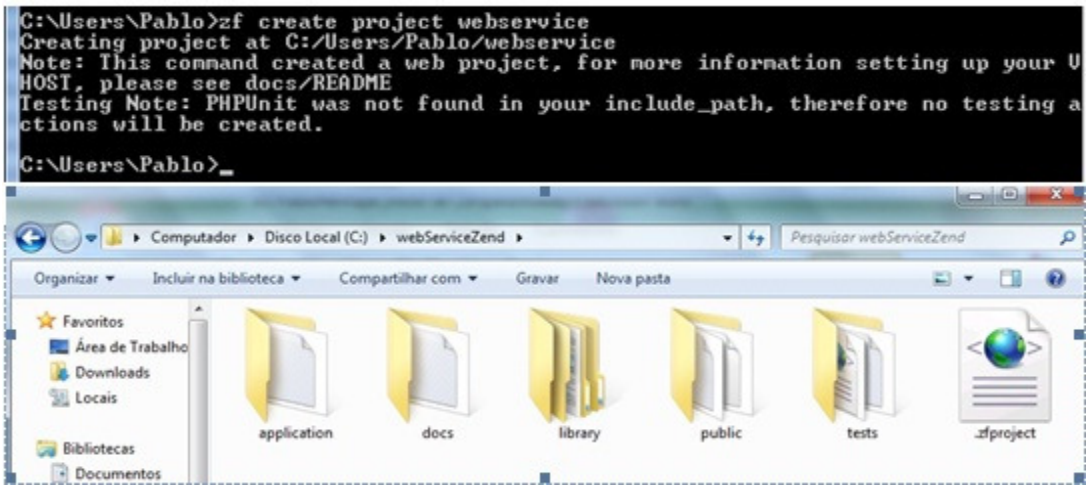


Figura 11 - Pastas do projeto /comando criação

A pasta "*public*" é onde a aplicação será executada, sendo que, dentro dela se encontram o arquivo *htaccess* e o *index.php* . O arquivo *htaccess (*Quadro 5*)* tem por finalidade fazer o mapeamento e redirecionar todas as requisições para o index.php. Nele se encontram as configurações do APPLICATION_PATH, e é definido o caminho relativo da pasta "*application*", onde o php vai buscar as bibliotecas do Zend Framework.

```
# public/.htaccess
RewriteEngine On
RewriteCond %{REQUEST_FILENAME} -s [OR]
RewriteCond %{REQUEST_FILENAME} -l [OR]
RewriteCond %{REQUEST_FILENAME} -d
RewriteRule ^.*$ - [NC,L]
RewriteRule ^.*$ index.php [NC,L]  (Redireciona tudo para o
index.php)
```

Quadro 5 - Arquivo HTACCESS

Ainda no index.php, de forma bastante abstrata é possível configurar o ambiente da aplicação, podendo assim existirem vários ambientes, como também várias rotas para uma mesma aplicação. Essas configurações ficam nos arquivos

`database.ini` , e `routes.ini` , localizados dentro da pasta application/config. O Quadro 6, mostra as definições do `index.php`.

```
<?php

define('APPLICATION_PATH', realpath(dirname( __FILE__
).'/../application/'));
set_include_path(APPLICATION_PATH.'/../library'.PATH_SEPARATOR.get_include_
path());
require_once "Zend/Loader.php";
Zend_Loader::registerAutoload();

try
{
    require '../application/bootstrap.php';
}
catch(Exception $exception)
{
    echo '<html><body><center>'
    .'An exception occured while bootstrapping the application.';
    if (defined('APPLICATION_ENVIRONMENT') && APPLICATION_ENVIRONMENT !=
'development')
    {
        echo '<br /><br />'.$exception->getMessage().'<br />'
        .'<div align="left">Stack Trace:'
        .'<pre>'.$exception->getTraceAsString().'</pre></div>';
    }
}
Zend_Controller_Front::getInstance()->dispatch();
```

Quadro 6 - Arquivo index.php.

O Zend Framework cria uma aplicação configurada e pronta para atender vários ambientes de formas diferentes: a criação de banco de dados, configurar a visão, configurar *layouts*, configurar rotas, registrar ações, *helpers*. Além disso, muitas vezes é preciso reutilizar o mesmo código para iniciar testes ou *scripts*. Embora seja possível simplesmente incluir seu *script* de inicialização, muitas vezes há inicializações que são de um ambiente específico – com isso pode-se precisar do MVC para tarefas agendadas, ou apenas a camada de banco de dados para um *script* de serviço.

Zend_Application visa tornar isso mais fácil e promover a reutilização através do encapsulamento de *bootstrapping* em paradigmas OOP (Programação Orientada a Objeto). Ele é dividido em 3 reinos:

- ***Zend_Application***: carrega o ambiente PHP, incluindo *include_paths* e *autoloading*, e instancia a classe `Bootstrap` solicitado.

- ***Zend_Application_Bootstrap***: fornece interfaces para as classes *bootstrap* (*Zend_Application_Bootstrap_Bootstrap*) e fornece a funcionalidade comum para a maioria das necessidades de inicialização, incluindo algoritmos de verificação de dependência.
- ***Zend_Application_Resource***: fornece uma interface para o *bootstrapping*, recursos que podem ser carregados por uma instância *bootstrap*, bem como implementações de recursos padrão.

O Quadro 7 mostra o arquivo `bootstrap.php` e as configurações de rotas e dos ambientes da aplicação. Todas as requisições irão passar por esse arquivo.

```
<?php
define ( 'APPLICATION_PRODUCTION', 'production' );
define ( 'APPLICATION_DEVELOPMENT', 'development' );
define ( 'APPLICATION_ENVIRONMENT', APPLICATION_DEVELOPMENT );

$frontController = Zend_Controller_Front::getInstance ();
$moduleConfiguration = new Zend_Config_Ini ( APPLICATION_PATH .
'/config/routes.ini', "modules" );

$modules = array ();
$modules ['default'] = APPLICATION_PATH . '/controllers';

foreach ( $moduleConfiguration->module as $name ) {
      $modules [$name] = APPLICATION_PATH . "/services/$name/controllers";
}
$frontController->setControllerDirectory ($modules);
$frontController->setParam ( 'env', APPLICATION_ENVIRONMENT );

$configuration = new Zend_Config_Ini ( APPLICATION_PATH .
'/config/database.ini', APPLICATION_ENVIRONMENT );
$dbAdapters = array ();

foreach ( $configuration->database as $config_name => $db ) {
      $dbAdapters [$config_name] = Zend_Db::factory ( $db->adapter, $db-
>params->toArray () );

      if (( boolean ) $db->default) {
            Zend_Db_Table::setDefaultAdapter ( $dbAdapters [$config_name]
);
      }
}
Zend_Registry::set ( 'dbAdapters', $dbAdapters );
```

Quadro 7 - Arquivo bootstrap.php

A seguir estão listadas algumas das classes e métodos que fazem parte da estrutura MVC e que são utilizadas para a implementação do *framework*.

- ***Zend_Controller***: é o coração do sistema do Zend Framework MVC. O Controlador Zend gerencia a comunicação entre as ações do usuário e sua aplicação.
- ***Zend_Controller_Front*** : implementa um *Front Controller* padrão, em que as requisições são interceptadas por ele e encaminhadas para os controladores de ação individual com base na URL solicitada.
- ***Zend_Controller_Action***: é uma classe abstrata que pode ser usada para implementar controladores de ação para uso com o *Front Controller* .
- ***Web_Service***: todo *controller* estenderá dessa classe, ela se encarregará de direcion-los a seus *DAO's*[4] específicos. Possuí os métodos CRUD, onde cada *controller* deverá implementá-lo ou sobrescreve-lo.
- ***_getDAO($dao):*** método da classe *service* que é chamado pelos métodos que necessitam interagir com os *DAO's*.
- ***setPath($path) :*** todo *controller* deverá setar o *application_path*(que é o módulo em que estará seus *controller's* e *DAO's*.)
- ***makeXmlReturn($array, $node):*** método da classe *service* que se encarregará de retornar o *array* de objetos na estrutura XML.
- ***Web_Utils_ArrayToXml:*** classe que criará os "nós" do XML.
- ***Web_Crud:*** classe de interação com o banco de dados, onde se encontram os *SQL's* em si.

O sistema Zend_Controller foi construído com extensibilidade em mente, pois escreve classes que implementam as várias interfaces e classes abstratas que formam a base da família de controladores , *actions* ou *helpers* para aumentar a funcionalidade do sistema.

Na criação do projeto, toda a estrutura MVC será criada dentro da pasta "*aplication*". Nela se encontram as pastas "*config*", "*controler*", "*views*" e "*services*". Dentro da pasta "`services/controllers`" estarão os módulos da aplicação, e seus *controllers* (Quadro 8).

[4] -Padrão para persistir dados em um banco de dados (SUN. JAVA, 2011).

```
class Dsm_ProdutoController extends Web_Service
{
	public function Dsm_ProdutoController()
	{
		$this->setPath(APPLICATION_PATH . "/services/PAB");
		$this->dao = $this->_getDAO("PAB_Produto");
		$this->node = "produto";
	}
```

Quadro 8 - PedidoController

No *controller* também é preciso especificar a qual módulo ele pertence, e a qual DAO se refere. No exemplo do Quadro 8, o módulo "*PAB*" é estabelecido informando à classe pai (*Service*) ,e o *DAO* a qual se refere e o nome do "nó" desejado para essa estrutura *XML*.

A classe DAO, `Dsm_Produto` Quadro 9 estenderá a `Web_CRUD` , que é onde se encontram os métodos CRUD genéricos, por isso é preciso informar o *`adapter`* (arquivo que contém as configurações de banco), lembrando que poderá haver mais de um *`adapter`* para a mesma aplicação, e também a tabela a qual se refere.

```
class Dsm_Produto extends Web_CRUD
{
	public function Dsm_Produto()
	{
		parent::setAdapter("dbdefault");
		parent::setTable("tb_produto");
		parent::setPrimary("pk_produto");
		$this->db = $this->loadAdapter();
	}

	public function getCount($search)
	{
		$sql = "select count(*) as total from tb_produto p inner join
tb_tipoproduto t on t.pk_tipoproduto = p.fk_tipoproduto where p.status =
1";

		if(!empty($search))
		{
			$sql .= " and p.descricao LIKE '%$search%'";
		}

		$result = $this->db->fetchAll($sql);

		if(!empty($result))
		{
			return $result;
		}
```

```
            else
            {
                  return array(0);
            }
      }

      public function createAction($dados)
      {
            $this->db->insert($this->getTable(), $dados);

            return $this->getLastId();
      }
```

Quadro 9 - DAO Pedido.

Cada *controller* possui um `indexAction`. Para chamá-lo na URL , é preciso colocar o nome do *controller* e o nome da *action* desejada, dessa forma o arquivo de configuração `routes.ini` *(*Quadro 10*)* se encarrega de recuperar as rotas certas para que a requisição chegue até a *action* correspondente.

```
[modules]
; ------------------------------------------------------------------- Module
Enable
module.DSM = DSM
module.PAB = PAB

[routes : modules]

; //------------------ PAB -------------------\\
routes.pabRoute1.route = PAB/:service
routes.pabRoute1.defaults.module = PAB
routes.pabRoute1.defaults.controller = Facade
routes.pabRoute1.defaults.action = index

routes.pabRoute2.route = PAB/:service/:param1
routes.pabRoute2.defaults.module = PAB
routes.pabRoute2.defaults.controller = Facade
routes.pabRoute2.defaults.action = index

routes.dsmRoute3.route = PAB/:service/:param1/:param2
routes.dsmRoute3.defaults.module = PAB
routes.dsmRoute3.defaults.controller = Facade
routes.dsmRoute3.defaults.action = index
```

Quadro 10 - Arquivo routes.ini

O `facade.php`, se encarrega de recuperar a requisição do navegador e encaminha-la para a ação. Caso a requisição seja desconhecida ele a encaminha para a ação padrão. Da mesma forma pode-se forçar uma ação através da URL,

para isso é preciso passar por parâmetro a *action* desejada. O Quadro 11 mostra o tratamento dessa situação.

```
public function index()
	{
		$permitedTypes = array("GET", "POST", "PUT", "DELETE");

		$params = $this->filterParams($this->getParams());

		if(in_array($params["methodType"], $permitedTypes))
		{
			if(!empty($params["service"]))
			{
				if(empty($params["_action"]))
				{
					$this->defaultAction($params);
				}
				else
				{
					$params["method"] = $params["_action"];

					$this->startServer($params);
				}
			}
			else
			{
				throw new Exception("<en>Service not
found.</en><pt-br>Serviço não encontrado.</pt-br>");
			}
		}
		else
		{
			throw new Exception("<en>Invalid method type.</en><pt-
br>Method type inválido</pt-br>");
		}
	}
```

Quadro 11 - IndexAction do facade.php

Se o parâmetro não contiver "*_action*", a requisição é encaminhada para o "*defaultAction*", que tentará interpretar, se existir, simplesmente inicia o servidor(Quadro 12), passando os parâmetros.

```
public function startServer($params)
	{

		// Faz a inclusãoo do serviço desejado.
		require_once(APPLICATION_PATH . "/services/" . $this->path
.'/controllers/'.$params["service"].'.php');

		if(empty($params["server"]))

		{
```

```
                    // Faz o load da biblioteca Zend Rest Server
                    Zend_Loader::loadClass("Zend_Rest_Server");

                    // Instancia o servidor Zend Rest
                    $server = new Zend_Rest_Server();
             }
             else
             {
                    // Faz o load da biblioteca Zend Amf Server
                    Zend_Loader::loadClass("Zend_Amf_Server");
                    // Instancia o servidor Zend Amf
                    $server = new Zend_Amf_Server();
             }


             // Define a classe a ser utilizada. ex: Rps_CookieController
             $server->setClass(ucfirst(strtolower($this-
>path))."_".$params["service"]);
             // Inicia o serviço
             $server->handle($params);
             }
```

Quadro 12 – Método de inicialização do serviço.

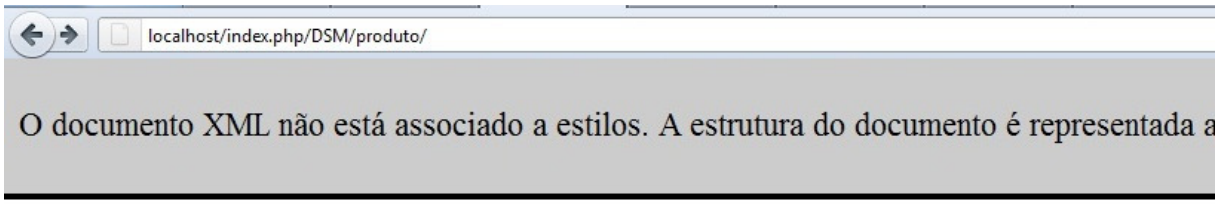
Web Services REST usam formatos de serviços específicos. Isso significa que a maneira para acessar um serviço *web REST* é diferente para cada serviço. Serviços *web REST* usam parâmetros de URL (GET dados) ou informações sobre o caminho para solicitar informações e dados, e POST para enviar dados (ZEND, 2011).

Zend Framework fornece funcionalidades de cliente e servidor, o componente *Zend_Rest_Server* apresenta exposição automática de funções e classes usando um formato significativo e simples que é o XML. Ao acessar esses serviços usando o *Zend_Rest_Client*, é possível recuperar facilmente os dados de retorno da chamada remota.

Além dos componentes *Zend_Rest_Server e Zend_Rest_Client*, as classes *Zend_Rest_Route* e *Zend_Rest_Controller* são fornecidos para rotear as solicitações de ajuda aos controladores REST.

Dessa forma toda requisição do navegador é interpretada e direcionada para o lugar certo, e o retorno é um XML puro, tornando possível a qualquer outra aplicação consumir esses dados e trabalhar de forma independente.

A Figura 12 mostra esse retorno e a passagem dos parâmetros na URL. Lembrando que DSM é o módulo, e produto é o *controller*. Se nenhum parâmetro for informado após o controller, todos os produtos serão retornados.




```
− <response>
  − <object>
    + <produto></produto>
    + <produto></produto>
    − <produto>
        <pk_produto>180</pk_produto>
        <descricao>ADUBO 00.20.10 + EXTRA FERTIPAR</descricao>
        <precocusto>650,00</precocusto>
        <precovenda>715,00</precovenda>
        <checkestoque>1</checkestoque>
        <qtdeestoque>124.9</qtdeestoque>
        <iniciovenda>14/07/2009</iniciovenda>
        <porcentagemmensal>1.7</porcentagemmensal>
        <fk_tipoproduto>1</fk_tipoproduto>
        <status>1</status>
        <tipodesc>ADUBO</tipodesc>
        <unidade>T</unidade>
        <validaestoque>sim</validaestoque>
      </produto>
```


Figura 12 - Chamada do ProdutoController

Como mencionado, também é possível fazer a mesma chamada forçando a ação, basta informar após o *controller*, a ação desejada (Figura 13).

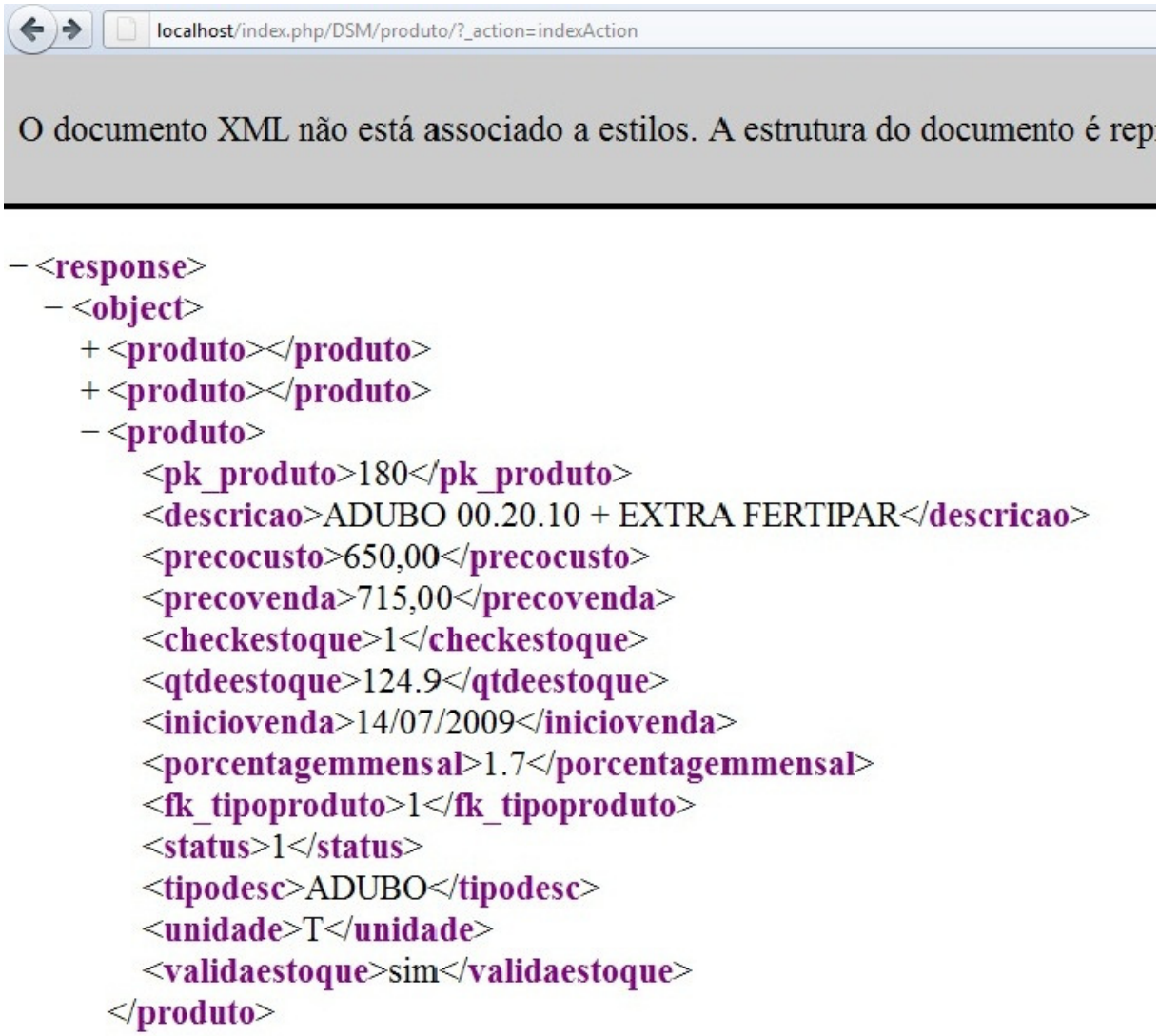


Figura 13 - Chamada do PedidoController passando a ação desejada.

Para apagar um produto, basta acrescentar na URL, após o *controller*, a ação `destroyAction` e o id do produto desejado como parâmetro.

As principais diferenças entre as linguagens de programação podem levar o projetista a optar por uma ou outra. Como opção, podem-se mesclar estas tecnologias e aproveitar o que cada uma oferece de melhor. A linguagem utilizada foi PHP, mas facilmente se conseguiria utilizar outra como *front-end* para exibir esses dados.

4 CONSIDERAÇÕES FINAIS

Quer no âmbito pessoal ou comercial, existe a necessidade da busca de alternativas que visam facilitar o entendimento e a socialização entre o homem e a máquina, e facilitar a execução das atividades ditas rotineiras.

Uma das áreas que mais passou por mudanças foi a área de Tecnologias da Informação e Comunicação. Customizar, encontrar novas formas de interação, bem como aumentar a capacidade de obter informações e o armazenamento de dados devem fazer parte do cotidiano dos profissionais que se dedicam a esta.

No mercado de *software* atual, existem dezenas de soluções práticas para o desenvolvimento de aplicações Web, permitindo a escolha da solução que atende os pré-requisitos da aplicação.

Neste trabalho, procurou-se demonstrar a possibilidade de integração entre linguagens através do estudo do Zend Framework, que por meio de um *web service* transformando toda resposta em XML, o torna consumível por qualquer outra aplicação.

As principais conclusões deste trabalho, que teve como metodologia a pesquisa experimental, foram:

- As linguagens de programação atuais oferecem vários recursos para que os programadores desenvolvam suas aplicações;
- Atualmente existem inúmeras linguagens de programação, cada uma com as suas características;
- O PHP é uma linguagem de fácil aprendizado, o que a torna uma das principais ferramentas utilizadas no desenvolvimento Web.
- O Zend Framework ou ZF cria uma arquitetura flexível e que permite o desenvolvimento de aplicações Web MVC.
- A característica fundamental do designer do ZF é a facilidade de utilizar somente as partes necessárias para o desenvolvimento da sua aplicação.
- *Web service* traz eficiência na comunicação entre cadeias de produção e agilidade para os processos.
- ZF fornece componentes individuais para requisitos comuns no desenvolvimento de aplicações web.

Esta experiência foi relevante, pois permitiu explorar o framework e verificar a viabilidade de adoção da implementação por parte de empresas e usuários, de acordo com as vantagens obtidas pelo estudo realizado e a importância da linguagem PHP que é uma linguagem amplamente utilizada no ambiente internet.

5 TRABALHOS FUTUROS

Como sugestão para trabalhos futuros, fica a idéia de explorar melhor os recursos disponibilizados pelo Zend, bem como seus benefícios na integração entre linguagens através de *web services*.

6 REFERÊNCIAS BIBLIOGRÁFICAS

ALBERTON, William Luís, Interação entre as linguagens PHP e Java utilizando o framework PHP/JAVA Bridge, 2008.

ABRAHAN Silberschatz, Henry F. KORTH, S. SUDARSHAN: Sistema de Banco de Dados 5.ª Edição. Campus, 2006.

AMOROSO, Danilo. O que é autenticação. Disponível em <http://www.tecmundo.com.br/1971-o-que-e-autenticacao-.htm> Acesso em: 12/10/2011.

WATHIER, Adair José. Segurança em Web Services com WS-Security. Disponível em<http://saloon.inf.ufrgs.br/twikidata/Docs/OnlineDoc20051217200943/SegurancaWebService_WSSecurity.pdf> Acesso em 01/11/2011.

ALLEN ,Rob , LO Nick e BROWN , Steven. Zend Framework in action .1 ed. Alta Books: Publishing, 2009.

BONIATI, Bruno B.; PADOIN, Edson Luiz. Web services como middlewares para interoperabilidade em sistemas. Revista do CCEI - Centro de Ciências da Economia e Informática, v. 7, n. 12, p. 17 – 24, Agosto 2003.

CASET, Jackson. Segurança em aplicações web. Disponível em < http://www.profissionaisti.com.br/2009/02/seguranca-em-aplicacoes-web> Acesso em: 12/10/2011.

CLEYDSON, José Ferreira. Implementando servidor de aplicações PHP utilizando Zend Framework. Disponível em < http://cleysinho.blogspot.com/2010/05/implementando-servidor-de-aplicacoes.html> Acesso em 24/09/2011.

DALL'OGLIO PHP: Programando com Orientação a Objetos. Novatec, São Paulo 2007.

DEXTRA. A linguagem franca da WEB. Disponível em< http://www.dextra.com.br/empresa/artigos/php.htm> Acesso em 08/11/2011.

EBECOM. PHP. Disponível em < http://www.ebecom.com.br/noticias/346/php.html >. Acesso em 01/09/2011.

ELTON, Luís Minetto. Frameworks para Desenvolvimento em PHP. Novatec, São Paulo 2007.

PERPÉTUO, Enio Júnior; JAGIELLO, Iverson Lourenço. Web services - Solução para aplicações distribuídas na internet.Disponível em <http://pt.scribd.com/doc/60043996/10/Web-Services> Acesso em 09/11/2011.

INFOLINK. PHP. Disponível em <http://faq.infolink.com.br/PHP> Acesso em: 12/10/2011.

LAURINDO, F. J. B. (2002) - Tecnologia da Informação: eficácia nas organizações. São Paulo, Editora Futura.

LISBOA, Flávio Gomes da Silva . Zend Framework: Desenvolvendo em PHP 5 orientado a objeto com MVC**.** Novatec, São Paulo.

LIBERTY, Jesse; KRALEY, Mike. Aprendendo a desenvolver documentos XML para a Web. SÃO PAULO: Makron-Books, 2001. 274pp.

MACHADO, Guilherme Bertoni. Uma Arquitetura baseada em Web Services com Diferenciação de Serviços para Integração de Sistemas Embutidos a outros Sistemas. Disponível em < http://www.inf.ufsc.br/~bertoni/artigos/dissertacao.pdf> Acesso em: 08/11/2011.

MELO, ALEXANDRE ALTAIR, NASCIMENTO, MAURICIO G.F. PHP Profissional. Novatec, São Paulo, 2007.

MICROSOFT. Understanding Models, Views and Controllers. Disponível em: < http://www.asp.net/mvc/tutorials/understanding-models-views-and-controllers-cs>. 2010c. Acesso em: 12/10/2011.

OLIVREIRO. O que é PHP. Disponível em <http://www.olivreiro.com.br/pdf/livros/cultura/1850599.pdf > Acesso em: 12/10/2011.

PHP. PHP: Hypertext Preprocessor. Disponível em < http://www.php.net/ >. Acesso em 05/09/2011.

PHP.NET .Manual PHP. Disponível em < http://in2.php.net/distributions/manual/php_manual_pt_BR.html.gz >. Acesso em 05/09/2011.

ROY, Jaideep; RAMANUJAN, Anupama. Understanding web services. IEEE Internet Computing, v. 3, n. 6, p. 69 – 73, Nov. - Dec. 2001.
RSS-ESPECIFICATION. RSS 2.0 Specification. Disponível em < http://www.rssboard.org/rss-specification >. Acesso em 15/10/2011.

RIBEIRO, Ticianne; BRAGA, Antônio Rafael; SOUSA, Verônica Lima Pimentel. Um ambiente virtual de ensino e aprendizagem baseado em web semântica e web services. Disponível em <http://artigocientifico.uol.com.br/uploads/artc_1160063078_11.pdf> Acesso em 09/11/2011.

SUN. JAVA. ORACLE; Disponível em http://java.sun.com/blueprints/corej2eepatterns/Patterns/DataAccessObject.html>. Acesso em 09/11/2011.

UDDI-101. UDDI 101. Disponível em < http://uddi.xml.org/uddi-101 >.Acesso em 09/11/2011.

Web24Horas. Programação PHP e MySQL com XML. Disponível em < http://www.web24horas.com.br/certificacao/progphpmysqlxml.php >. Acesso em 15/09/2011.

W3C, Web Services Architecture. Disponível em <http://www.w3.org/TR/ws-arch/> Acesso em 20/10/2011.

ZEND, Zend Framework. Disponível em < http://framework.zend.com > Acesso em 05/10/2011.

ZENDCON, Zend PHP Conference. Disponível em < http://zendcon.com> Acesso em 09/11/2011.

ZENDOC, Zend Framework 1.11.3 Manual. Disponível em <http://doczf.mikaelkael.fr/1.11/en/> Acesso em 09/11/2011.