UNIVERSIDADE FEDERAL RURAL DO SEMI-ÁRIDO – UFERSA
DEPARTAMENTO DE CIÊNCIAS EXATAS E NATURAIS – DCEN
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

DAVI REBOUÇAS MATOS

AVALIAÇÃO DA UTILIZAÇÃO DE FRAMEWORKS MVC NO DESENVOLVIMENTO DE SISTEMAS WEB

MOSSORÓ-RN
2015

DAVI REBOUÇAS MATOS

AVALIAÇÃO DA UTILIZAÇÃO DE FRAMEWORKS MVC NO DESENVOLVIMENTO DE SISTEMAS WEB

Monografia apresentada à Universidade Federal Rural do Semi-Árido – UFERSA, Departamento de Ciências Exatas e Naturais, para a obtenção do título de Bacharel em Ciência da Computação.

Orientador: Prof. DSc. Paulo Gabriel Gadelha Queiroz.

MOSSORÓ-RN

2015

Dados Internacionais de Catalogação na Publicação (CIP)
Biblioteca Central Orlando Teixeira (BCOT)
Setor de Informação e Referência

M425a Matos, Davi Rebouças

Avaliação da utilização de frameworks MVC no desenvolvimento de sistemas web/ Davi Rebouças Matos -- Mossoró, 2015.
51f.: il.

Orientador: Prof. Dr. Paulo Gabriel Gadelha Queiroz

Monografia (Graduação em Ciência da Computação) – Universidade Federal Rural do Semi-Árido. Pró-Reitoria de Graduação.

1. Softwares. 2. Desenvolvimento web. 3. Linguagem PHP. 4. Zend Framework. 5. Padrão MVC. I. Título.

RN/UFERSA/BCOT/145-15 CDD: 005.3

Bibliotecária: Vanessa Christiane Alves de Souza Borba
CRB-15/452

Aos meus pais, irmãos e amigos que contribuíram para o meu crescimento e que estiveram ao meu lado durante todas as etapas da minha vida.

AGRADECIMENTOS

Primeiramente a Deus pela vida e pelas pessoas maravilhosas que tem colocado em meu caminho. Sou grato também por ter me ajudado em todos os momentos que precisei, permitindo a realização desse sonho.

À minha mãe, Leila, minha maior inspiração, pelo amor, carinho e valores que me deu. Pelo apoio nas horas difíceis, de desânimo e cansaço, a quem especialmente dedico essa conquista. Ao meu pai, Silvio, pelo amor e pelos valiosos ensinamentos que contribuíram para que eu me tornasse a pessoa que sou hoje.

Aos meus irmãos, Lívia, Silvio, Clara e Ana, pela união, incentivo que me deram durante toda a minha vida e pelos momentos de felicidades que pudemos compartilhar juntos.

Aos meus colegas de faculdade, no qual levarei a amizade pelo resto da vida, obrigado pelo companheirismo durante todo o curso e por compartilharem seus conhecimentos durante os momentos difíceis. Estarei torcendo pelo sucesso de cada um de vocês.

À todos os meus amigos que contribuíram de alguma forma para a realização desta conquista, em especial, aos meus grandes amigos Gerlan e Wesley, agradeço pelos anos de amizade, pelo convívio desde os tempos de colégio e pela ajuda nos momentos que mais precisei. Ao meu amigo e patrão, Rosenildo, pela compreensão e colaboração durante o período da minha graduação. Ao meu amigo Hallysson Fernandes, por sempre se colocar à disposição quando precisei. Também aos meus amigos Alisson Maurício e Luciano Thiago que me ajudaram no desenvolvimento deste trabalho. Sou eternamente grato pela amizade de todos vocês.

Ao meu orientador, Prof. Paulo Gabriel, agradeço pelo auxílio, paciência e dedicação durante a realização deste trabalho.

Também um agradecimento especial aos professores que fizeram parte da minha banca, Prof. Milton Mendes e Profª. Angélica Castro, agradeço por terem aceitado ao convite e contribuído para a conclusão deste trabalho.

À todos os professores da minha graduação, pelos conhecimentos e experiências ensinadas.

“Cada sonho que você deixa pra trás, é um pedaço do seu futuro que deixa de existir.”

(Steve Jobs)

RESUMO

Com a rápida expansão da internet, o mundo passou por uma grande mudança quando tratamos de *softwares*. A migração da plataforma *Desktop* para *Web*, resulta em aplicações mais robustas e complexas na rede. Para acompanhar essa mudança, faz-se necessário empregar técnicas avançadas de desenvolvimento, tais como padrões de projeto e boas práticas de programação. O presente trabalho propõe a avaliação da utilização dessas técnicas, por meio do desenvolvimento e da comparação de duas versões de uma aplicação CMS (*Content Management System*). A primeira foi desenvolvida de forma artesanal, sem a utilização de técnicas avançadas de desenvolvimento. A segunda versão foi desenvolvida com a utilização de padrões de projeto e dos recursos fornecidos por um *framework*. O framework escolhido para a realização do estudo foi o *Zend Framework*, que é uma ferramenta *open source* desenvolvida para a linguagem de programação PHP, e faz a utilização do padrão arquitetural MVC (*Model-View-Controller*). Os resultados obtidos são apresentados por meio de métricas de software. O objetivo das métricas é fornecer uma forma de avaliar os ganhos e as dificuldades na construção do projeto.

Palavras-chave: Desenvolvimento *Web*. Padrões de Projeto. *Zend Framework*. Linguagem PHP. Padrão MVC.

LISTA DE FIGURAS

LISTA DE LISTAGENS

LISTA DE TABELAS

LISTA DE SIGLAS

API	*Application Programming Interface*
CC	*Cyclomatic Complexity Number*
CMS	*Content Management System*
CRUD	*Create, Read, Update, Delete*
CSS	*Cascading Style Sheets*
HTML	*HyperText Markup Language*
HTTP	*Hipertext Transfer Protocol*
IDE	*Integrated Development Environment*
LLOC	*Logicla Lines of Code*
LOC	*Lines of Code*
MI	*Maintenability Index*
MVC	*Model-View-Controller*
PDF	*Portable Document Format*
PHP	*Hypertext Preprocessor*
SGBD	Sistema de Gerenciamento de Banco de Dados
SQL	*Structured Query Language*
TI	Tecnologia da informação
URL	*Uniform Resource Locator*
WEB	*World Wide Web*
ZF	*Zend Framework*

SUMÁRIO

1. INTRODUÇÃO

O desenvolvimento *Web* ganhou espaço no mercado, não somente com a produção de *Websites*, mas também com o desenvolvimento de aplicações *Web*, tão ou mais complexas e robustas que as aplicações desenvolvidas para *Desktop*. A possibilidade de acessar a aplicação de qualquer lugar, utilizando-se somente um computador com acesso à internet por meio do *browser*, sem a necessidade de instalação de outros programas, e a facilidade de configuração, integração e manutenção em servidores *Web*, são alguns dos motivos que contribuem para que programadores encontrem no desenvolvimento *Web* a plataforma ideal para desenvolver suas aplicações.

Juntamente com a evolução dos sistemas *Web*, cresce também a necessidade de se ter formas práticas e efetivas de reduzir o esforço de manutenção e evolução dos sistemas, que corresponde a altos custos dentro do processo de desenvolvimento. Segundo SOMMERVILLE (2003), "os custos de manutenção de sistemas equivalem-se aos custos de desenvolvimento. Para sistemas de tempo real incorporados, os custos de manutenção podem ser até quatro vezes mais altos do que os custos de desenvolvimento".

A construção de uma aplicação *Web* envolve vários arquivos de código, não só PHP (NIEDERAUER, 2008), como também HTML (PILGRIM, 2011), folhas de estilos CSS (SCHMITT, 2010) e arquivos de configuração. Deste modo, o emprego da implementação artesanal pode resultar em um código-fonte de alta complexidade de compreensão, dificultando na manutenção e evolução do *software*. Com isso, faz-se necessário a utilização de padrões, técnicas, ferramentas e métodos conhecidos, estudados e testados com objetivo de auxiliar na organização e implementação do projeto. A partir dessa necessidade surgiram os *frameworks* (JOHNSON, 1991). Segundo FAYAD (1997), "frameworks representam uma estrutura formada por blocos pré-fabricados de software que os programadores podem usar, estender ou adaptar para uma solução específica e linguagens de padrões".

Ao se utilizar um *framework* no desenvolvimento de um sistema *Web*, estamos garantindo a consistência e qualidade da aplicação. Segundo MINETTO (2007), "estes além de facilitarem a vida dos programadores, também trazem um ótimo ganho de rendimento no desenvolvimento de aplicações". Isso é obtido a partir das técnicas utilizadas, como arquitetura MVC (*Model-View-Controller*), *Design Patterns* e boas práticas de programação, implicando em:

- Padronização, já que toda estrutura tem seu padrão a ser seguido, desde a separação de camadas, a nomes de classes e variáveis;

- Agilidade, pois não se faz necessário a implementação de algumas classes e métodos fundamentais, além de evitar repetição de código;
- Segurança, com classes específicas de criptografias, restrição de acesso, prevenção de ataques, a fim de manter a integridade do sistema;
- Facilidade de manutenção, como consequência da padronização e qualidade do código.

Existem indícios que a utilização de *frameworks* no desenvolvimento *Web* traz inúmeros benefícios para o projeto, mas como podemos avaliar esse ganho? Este trabalho tem como objetivo a realização de um estudo detalhado do *Zend Framework*, uma ferramenta de desenvolvimento baseada na linguagem PHP (*Hypertext Preprocessor*) e arquitetura MVC, aplicando um estudo experimental com a implementação de uma aplicação CMS (*Content Management System*) em duas formas, implementado sem nenhuma técnica avançada de desenvolvimento, tal como padrões de projeto ou frameworks e utilizando o *Zend framework*. Espera-se obter, como resultado, a confirmação das vantagens da utilização de frameworks descritas acima em relação à implementação artesanal.

1.1. Problema

Com a evolução dos sistemas *Web*, a criação de um produto de *software* se tornou um processo muito mais complexo, exigindo cada vez mais o uso de avançadas técnicas de desenvolvimento, a fim de auxiliar e agilizar o desenvolvimento.

Ao mesmo tempo em que os sistemas evoluem, exige-se mais dos membros envolvidos no projeto que, por sua vez, buscam agilizar o processo de produção do sistema, tendendo a um desenvolvimento paralelo e modular. Segundo BALDWIN (2000), "a modularidade é uma estratégia para construir processos ou produtos complexos a partir de pequenos subsistemas que podem ser desenvolvidos individualmente, mas que funcionam como um conjunto integrado". Com essa técnica é possível obter uma maior produtividade entre as pessoas envolvidas, além de agilizar todo o processo.

Segundo LISBOA (2008), "o desenvolvimento de sistemas consiste, basicamente, em criar um conjunto de atividades, parcialmente ordenadas, com a finalidade de obter um produto final". Entretanto, o maior esforço não está na criação, e sim na manutenção. Durante o desenvolvimento de um projeto, é comum a constante mudança de requisitos e funcionalidades do sistema. A customização de um sistema é mais trabalhosa pois afeta a estrutura do projeto e requer um maior esforço na utilização de técnicas avançadas de Engenharia de *software*. É

preciso ter um projeto que permita a manutenção do código-fonte sem que haja grandes impactos no projeto.

Caso o sistema a ser desenvolvido seja um produto para grandes empresas, ou tenha uma constante utilização, certamente haverá uma evolução desse sistema, utilizando técnicas de reuso de *software*. Segundo SOMMERVILLE (2003), "a engenharia de *software* baseada em reuso é uma abordagem de desenvolvimento que tenta maximizar o reuso do *software* existente". Para isso, precisamos ter um código-fonte bem estruturado e de qualidade.

Diante desse contexto e dos fortes indícios de que o uso de um *framework* MVC pode trazer muitos benefícios para o desenvolvimento *Web*, o questionamento que norteou as investigações para essa pesquisa, foi: Apesar do uso de *frameworks* exigir um maior esforço inicial para aprender seus recursos e organizar o sistema segundo sua estrutura, vale a pena a sua utilização?

1.2. Objetivos

A partir da questão de pesquisa definida na subseção anterior, o objetivo desta monografia é avaliar se a utilização de frameworks MVC pode trazer mais benefícios do que ônus para o desenvolvimento *Web* e quais são esses benefícios.

1.2.1. Objetivo Geral

Avaliar se a utilização de frameworks MVC pode trazer benefícios para o desenvolvimento *Web* e definir quais são esses possíveis benefícios.

1.2.2. Objetivos Específicos

- Elaborar um referencial teórico sobre *frameworks*;
- Desenvolver uma aplicação como estudo experimental para demonstrar as principais características do *Zend Framework*, contendo as seguintes funcionalidades:
    - Consultar, cadastrar, alterar e excluir Usuário;
    - Consultar, cadastrar, alterar e excluir Portfólio;
    - Consultar, cadastrar, alterar e excluir Marcas;

- Consultar, cadastrar, alterar e excluir Frases;
- Consultar, cadastrar, alterar e excluir Funcionários;
- Consultar, cadastrar, alterar e excluir Vídeos;
- Solicitar suporte técnico;
- Gerar estatísticas de acesso.

- Testar e avaliar os resultados obtidos, apresentando as vantagens e desvantagens obtidas no processo de desenvolvimento com apoio de um *framework*.

1.3. Justificativa

O desenvolvimento *Web* está ganhando preferência entre as plataformas de desenvolvimento. Os motivos que justificam essa preferência são: portabilidade no acesso, não necessidade de instalação de programas e facilidade de manipulação do servidor, entre outros. As aplicações tornam-se cada vez mais complexas, e as necessidades de mudanças e evolução desses sistemas são corriqueiras, nas quais se requer um grande esforço de desenvolvimento. 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 (SILVA, 2011).

Portanto, faz-se necessário a utilização de técnicas avançadas de desenvolvimento, tais como a utilização de padrões de projeto e *frameworks.* Tomando o *Zend Framework* como objeto de estudo, foram utilizados todas as vantagens da programação orientada a objetos, linguagem PHP 5, além de usar o padrão MVC na arquitetura do projeto, que faz sua divisão em 3 camadas bem definidas, separando a lógica de negócio e acesso a dados da apresentação e interação do utilizador, introduzindo uma camada de controle para intermediá-las.

Sabendo da importância da utilização dessas ferramentas para se desenvolver uma aplicação de alto nível, o autor deste trabalho irá implantar as técnicas utilizadas no estudo em suas experiências profissionais, afim de melhorar as práticas de desenvolvimento atuais. Dessa forma pode-se mostrar, na prática, conceitos vistos durante as disciplinas do curso de Ciência da Computação.

1.4. Metodologia de Pesquisa

A partir de uma aplicação CMS previamente desenvolvida em PHP, de forma artesanal, foi feita uma pesquisa dos *frameworks* PHP mais utilizados, que resultou na escolha do que

mais se adequou aos requisitos do sistema. Após a escolha do *framework*, foi desenvolvida a mesma aplicação mas, dessa vez, com a utilização de todas as técnicas e padrões utilizadas pela ferramenta. Em seguida, foram definidas métricas de *software*, que foram utilizadas para comparar os dois sistemas desenvolvidos. Por fim, foi feita uma avaliação dos resultado obtidos, a fim de mostrar os benefícios do uso do *framework*.

1.5. Estrutura da Monografia

Esta monografia está dividida da seguinte forma. No capítulo 2, apresenta-se a fundamentação teórica, que define todas as tecnologias, padrões e técnicas utilizadas no estudo, tais como as definições de *framework*, contexto histórico sobre o PHP e o SGBD *MySQL*, o padrão MVC e os principais recursos do *Zend Framework*. Os recursos, ferramentas e a estrutura da aplicação desenvolvida são apresentados no capítulo 3. No capítulo 4, é feita a análise das duas versões do sistema, por meio da utilização de métricas, além da apresentação dos resultados obtidos. As conclusões do estudo e os trabalhos futuros são descritos no capítulo 5.

2. FUNDAMENTAÇÃO TEÓRICA

Neste capítulo, são descritas as tecnologias que foram utilizadas para o desenvolvimento da aplicação CMS proposta. Foram desenvolvidas duas versões do sistema: a primeira de forma artesanal; e outra utilizando-se o *Zend Framework*.

As duas versões da aplicação fazem a utilização do *framework front-end Bootstrap*, que consiste em uma coleção de ferramentas *open-source* amplamente utilizada. No desenvolvimento *Web* para auxiliar na construção do *design* do projeto. Seus conjuntos de pacotes HTML, CSS e JS facilitam a programação da interface gráfica do sistema, permitindo uso de técnicas de *webdesign* como *design* responsivo e *mobile-first*[1].

Há alguns anos a linguagem de programação PHP tem sido a escolha de muitos desenvolvedores quando se trata de desenvolvimento *Web*. O PHP foi adotado para este estudo por se tratar de uma linguagem bastante popular, atualmente entre as 4 linguagens *Web* mais utilizadas[2]. Também é uma linguagem gratuita, consolidada e com uma grande comunidade ativa. O SGBD *MySQL* foi adotado em conjunto com a linguagem de programação para estruturação do banco de dados da aplicação.

A utilização das técnicas de programação e padrões de projeto na segunda versão da aplicação teve como base os recursos do *Zend Framework*, conhecido na comunidade de desenvolvedores como um dos mais completos *frameworks* PHP disponíveis no mercado.

Este capítulo está organizado da seguinte forma: na seção 2.1 apresenta-se o conceito de *framework* no processo de desenvolvimento e suas utilidades; na seção 2.2 fala-se sobre o contexto histórico, vantagens e aplicações da linguagem PHP; a seção 2.3 descreve o SGBD *MySQL* e suas características particulares; A seção 2.4 apresenta a teoria do padrão de projeto MVC e suas aplicações; Por fim, na seção 2.5, descreve-se o *Zend Framework* e apresenta-se uma visão geral dos principais recursos da ferramenta.

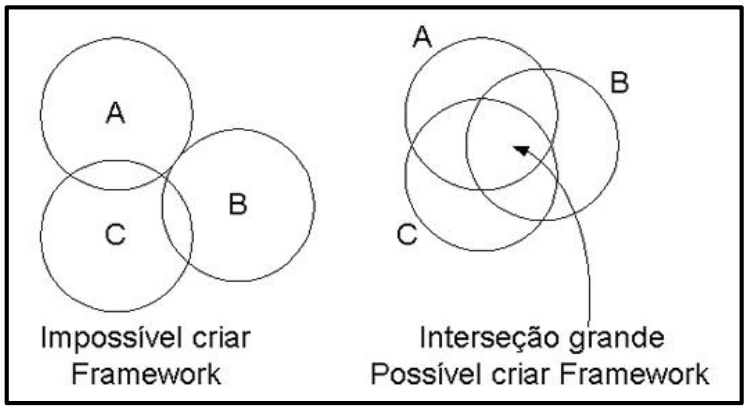
2.1. Framework

São formados por um conjunto de padrões de software para resolver um problema complexo em um determinado domínio, ou seja, são funções que surgiram para agilizar o trabalho dos desenvolvedores de software (APPLETON, 1997).

[1] http://tableless.com.br/mobile-first-a-arte-de-pensar-com-foco/
[2] http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html

Atualmente, muitos *frameworks* disponíveis no mercado adotam o padrão MVC e uma abordagem orientada a objetos para sua implementação, oferecendo uma estrutura básica para o desenvolvimento de suas aplicações. Segundo LISBOA (2008), "a utilização de *frameworks* facilita a organização de uma aplicação, realiza injeção de dependência se necessário, valida campos com lógica de negócio e ajuda a tornar a estrutura do projeto organizada na maioria das vezes dentro de um modelo MVC (*Model-View-Controller*)". A utilização da técnica é voltada para o reuso de código e auxílio no processo de desenvolvimento de uma aplicação, atuando em funcionalidades em comum a várias aplicações, evitando assim, ter que implementar rotinas repetidas. Na Figura 1, mostra-se um exemplo de oportunidade de criação de um *framework* a partir de um recurso compartilhado entre os programas.

Figura 1 – Representação da criação de *framework*



Fonte: SAUVÉ (2000)

Fazendo uso de *frameworks*, não reaproveitamos somente simples componentes da aplicação, mas subsistemas, aumentando o grau de reuso de *software*.

PREE (1995) apresenta o *framework* como sendo formado por duas partes:

- *Frozen spots*: definem a arquitetura global de um sistema, seus componentes básicos e os relacionamentos entre eles, permanecendo inalterados – daí o termo congelado - em qualquer instanciação do *framework*;
- *Hot spots*: pontos de refinamentos predefinidos, onde a especialização e a adaptação ocorrem. A especialização de *hot spots* requer a definição de classes adicionais de maneira a sobrepor métodos e/ou configurações de objetos baseados nos componentes já fornecidos pelo *framework*.

Para GAMMA (1995), "um *framework* é um conjunto de objetos que colaboram com o objetivo de atender a um conjunto de responsabilidades para uma aplicação específica ou um domínio de aplicação".

Trazendo para o cenário *Web*, validação de dados de formulários, navegação entre as páginas, segurança de autenticação, envio de e-mail, internacionalização, integração com API's, são exemplos de rotinas comuns que podem ser implementadas por um *framework*.

As vantagens obtidas com o uso de *frameworks* são:

- Reutilização;
- Modularização;
- Extensibilidade;
- Controle da estrutura e fluxo de controle dos programas.

Após uma pesquisa dos *frameworks* PHP mais utilizados, foi escolhido o Zend por possuir o maior número de recursos e pela qualidade da documentação disponibilizada. Foram considerados os critérios de suporte ao PHP5, mapeamento objeto-relacional (ORM), suporte a adaptação e configuração de *templates* para aplicação, suporte à tecnologia Ajax, módulo de autenticação nativo e disponibilização de módulos auxiliares para o desenvolvimento. Na Tabela 1, mostra-se a comparação entre os principais *frameworks* do mercado.

Tabela 1 – Comparação dos *frameworks* PHP

	PHP5	ORM	Template	Ajax	Módulo de Autenticação	Modular
Zend Framework	X	X	X	X	X	X
CakePHP	X	X	-	X	X	X
Codeigniter	X	-	X	-	-	-
Symfony	X	X	X	X	X	X

Fonte: autoria própria

2.2. Linguagem De Programação PHP

O PHP (*Hypertext Preprocessor*) é uma linguagem de programação *server-side* de propósito geral, capaz de gerar conteúdo dinâmico na *Web*. Foi criado por Rasmus Lerdorf em 1994 e disponibilizado para a comunidade por código livre em 1995, com uma sintaxe similar a C/C++ e Perl. A linguagem ganhou rápida aceitação entre os desenvolvedores por ser uma linguagem rápida, flexível, de fácil utilização e por possuir diversos recursos de qualidade.

Esses foram alguns fatores determinantes para que o PHP se mostrasse um verdadeiro fenômeno em termos de aplicações *Web*, se tornando uma das linguagens de programação para *Web* que mais cresceu nos últimos anos.

Segundo FERRO NETO (2011), "a linguagem 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. Apresenta como principais características sua velocidade, robustez e portabilidade".

No decorrer dos anos, a linguagem se adaptou à evolução e às necessidades da internet, foram lançadas novas versões com a adição de recursos como: orientação a objetos e suas propriedades, programação funcional, interface de linha de comando, suporte a vários tipos de base de dados e melhorias na segurança e performance, chegando até a versão atual da linguagem (PHP 5.6).

O PHP é uma linguagem dinâmica e flexível, que suporta uma variedade de técnicas de programação. Ele evoluiu com o passar dos anos, notavelmente adicionando um sólido modelo de orientação a objetos no PHP 5.0 (2004), funções anônimas e *namespaces* no PHP 5.3 (2009) e *traits* no PHP 5.4 (2012) (LOCKHART, 2014).

Um código PHP pode ser escrito em qualquer editor de textos como, por exemplo, o bloco de notas. Para o reconhecimento de um trecho de código pelo servidor, ele deverá ser escrito entre as *tags* "<*?php*" e "*?*>", como se apresenta a Listagem 1.

Listagem 1 – Exemplo de código PHP

```php
<?php
    // legal, estou escrevendo meu primeiro programa em PHP
    echo "<h1 align='center'>Este é meu primeiro programa!</h1>";
?>
```

Fonte: NIEDERAUER (2008)

Normalmente uma página PHP não contém apenas códigos de programação PHP, mas também *tags* de marcação HTML. Enquanto o PHP representa a parte dinâmica da página, o HTML representa a parta estática (NIEDERAUER, 2008). Na Listagem 2, mostra-se um código de exibição de data atual escrito em PHP embutido ao código HTML.

Listagem 2 – Trechos de código PHP embutidos em código HTML

```
<html>
    <body>
        <?php
            $data_de_hoje = date("d/m/Y", time());
        ?>
        <p align="center">Hoje é dia <?php echo $data_de_hoje; ?></p>
    </body>
</html>
```

Fonte: NIEDERAUER (2008)

Para a abordagem orientada a objetos, o PHP utiliza uma estrutura semelhante as demais linguagens, com suporte a classes abstratas, interfaces, herança, construtores, clonagem, exceções e etc. Na criação de uma classe, o código-fonte deve ser salvo na extensão *class.php*, como é apresentado na Listagem 3.

Listagem 3 – Exemplo de uma classe Carro em PHP

```
<?php
    class Carro
    {
        public $Codigo;
        public $Descricao;
        public $Cor;
        public $Preco;
    }
?>
```

Fonte: autoria própria

Pode-se observar que a linguagem tem como ponto forte a facilidade de utilização e codificação. Associado a isso, tem suporte às bases de dados mais utilizadas, como: *Oracle*, *Sybase*, *PostgreSQL*, *InterBase*, *MySQL*, *SQLite*, *MSSQL*, *Firebird*, etc.

O PHP conecta-se a uma enorme variedade de gerenciadores de banco de dados disponíveis no mercado. Para cada tipo de banco de dados existe uma gama de funções associadas para realizar operações como conexão, consulta, retorno, desconexão, dentre outras (DALL'OGLIO, 2009).

De acordo com o *Netcraft*[3], existem mais de 672 milhões de *sites* ativos que executam PHP, confirmando a popularidade do PHP na plataforma *Web*.

[3] http://news.netcraft.com/archives/2013/05/03/may-2013-web-server-survey.html

Além dos benefícios apresentados, o PHP foi utilizado neste trabalho, pois é a linguagem de programação adotada pela empresa em que o autor desta monografia trabalha. Desse modo, o autor aproveitou uma aplicação previamente desenvolvida e que apresentou problemas de manutenção para comparar com uma aplicação totalmente nova, feita com a utilização do Zend.

2.3. SGBD *MySQL*

MySQL é um SGBD (Sistema Gerenciador de Bancos de Dados) relacional que utiliza a linguagem padrão SQL (*Structured Query Language*), e é bastante utilizado em aplicações para a Internet. É o mais popular entre os bancos de dados com código-fonte aberto (NIEDERAUER, 2008).

O *MySQL* foi criado por David Axmark, Allan Larsson e Michael Widenius, começando o desenvolvimento do projeto em 1994. Seu grande sucesso é devido a fácil integração com a maioria das linguagens de programação atuais. Junto com o PHP, ele forma um pacote suportado na maioria dos servidores de hospedagem pela internet. Na Listagem 4, pode-se observar um código PHP de uma *String* SQL a ser enviado para o SGBD.

Listagem 4 – Exemplo consulta SQL

```
<?php
    $sql = "SELECT COD_ORG, DES_ORG
            FROM org_leg
            ORDER BY COD_ORG DESC";
?>
```

Fonte: autoria própria

Para NIEDERAUER (2008), "o *MySQL* é uma alternativa atrativa porque, mesmo possuindo uma tecnologia complexa de banco de dados, seu custo é bastante baixo. Tem como destaque suas características de velocidade, escalabilidade e confiabilidade, o que vem fazendo com que ele seja adotado por departamentos de TI (Tecnologia da Informação), desenvolvedores *Web* e vendedores de pacotes de *software*".

NIEDERAUER (2008) destaca também algumas vantagens do SGBD:

- Grande número de utilização por usuários simultâneos;
- Capacidade de manipulação de tabelas com mais de 50.000.000 registros;

- Alta velocidade de execução de comandos;
- Fácil e eficiente controle de privilégio de usuários.

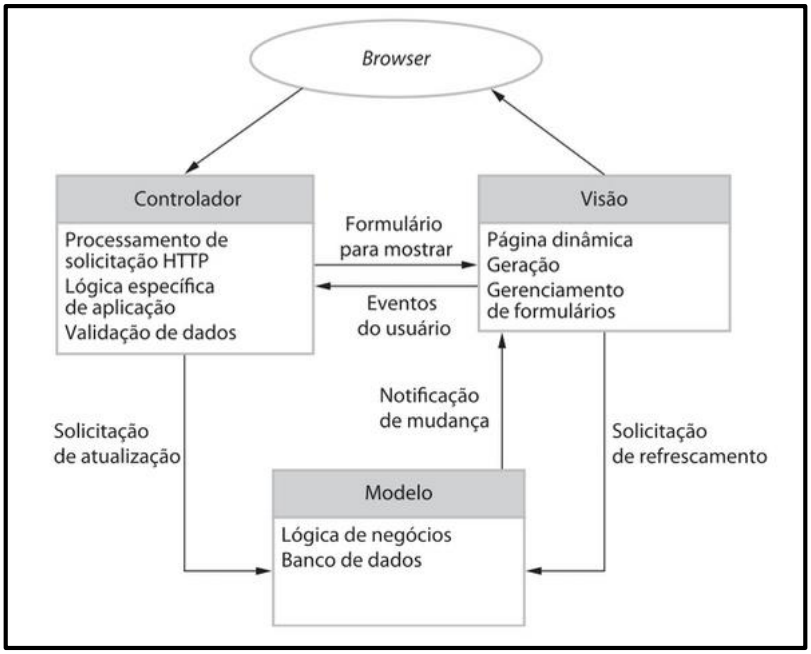
O *MySQL* é ideal para qualquer tipo de aplicação, desde pequenos projetos de *websites* dinâmicos a base de dados de grandes empresas com alto volume de dados. Empresas como *NASA, Motorola, Yahoo!* e *Suzuki* adotaram o SGBD para gerenciar os banco de dados de suas aplicações (MILANI, 2007).

2.4. Padrão MVC

BONOTO (2012) afirma que MVC faz parte de uma boa prática de projetos de aplicações *Web* modernas. É um modelo de arquitetura de *software* com ideia central de organizar da melhor forma o projeto, separando a regra de negócio e o acesso aos dados da interação do usuário com ele.

O padrão MVC sugere fazer uma divisão em três camadas, o modelo (*model*) que compreende a toda parte lógica da aplicação, regra de negócios e rotinas de acesso aos dados, definindo toda a funcionalidade básica da aplicação por trás de um conjunto de abstrações. A visão (*view*) que consiste em toda saída e representação de dados, como interface gráfica, tabelas e alertas. É neste módulo que é tratada toda a interação com usuário. Por fim faz-se necessário um controlador (*controller*) para intermediar as outras duas camadas, é nele que se faz toda a vinculação das *views* do projeto dependendo, por exemplo, de uma interação do usuário. Eles manipulam modelos, decidem qual apresentação será exibida com base em solicitações do usuário e em outros fatores, repassam os dados que cada apresentação necessita, ou transferem completamente a gerência para outro controlador (ZEND, 2014). A Figura 2 mostra a arquitetura MVC, no contexto de uma aplicação *Web*.

Figura 2 – Arquitetura de aplicações *Web* usando padrão MVC



Fonte: SUMMERVILLE (2011)

Essa abordagem em camadas apoia o desenvolvimento incremental de sistemas. Quando uma camada é desenvolvida, alguns do serviços prestados por ela podem ser disponibilizados para os usuários. Além disso, quando a camada de interfaces muda ou tem novos recursos adicionais, apenas a camada adjacente é afetada (SUMMERVILLE, 2011).

2.5. Zend Framework

O *Zend Framework* (ZF) faz parte do projeto PHP *Collaboration*, e é uma iniciativa da empresa *Zend Technologies*. Ele é um *framework* escrito em PHP5 que utiliza todos os recursos de orientação a objetos fornecidos por essa versão da linguagem, além de prezar a utilização de padrões de projetos, visando à construção de componentes reutilizáveis (ZEND, 2014).

O ZF é um *framework open-source* que vem com uma proposta modular, o que significa que pode ser utilizado com os módulos que se adequam à sua aplicação, permitindo a adição de novas funcionalidades disponíveis e até de integração a outras aplicações ou *frameworks* do mercado. Além disso, é extensível, abrindo a possibilidade de adaptação da ferramenta para as necessidades de cada projeto. Possui ferramentas de criptografia e segurança, suporta internacionalização da aplicação, e conta com uma comunidade de usuários e colaboradores ativa para suporte e continuidade da ferramenta.

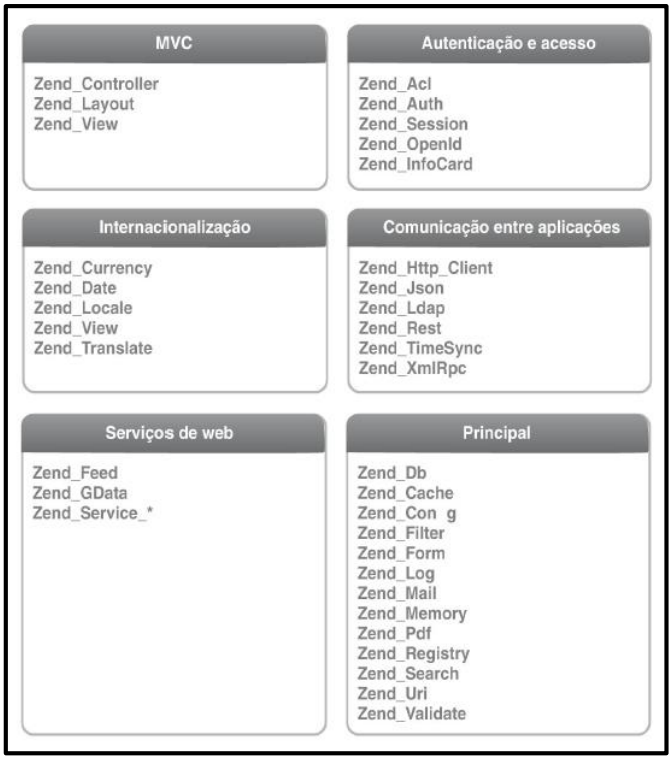
O *Zend Framework* é flexível, possui uma coleção de componentes reutilizáveis e extensíveis que podem ser utilizados juntos ou separados, e permite que os desenvolvedores usem qualquer componente de forma livre. Além disso, ele oferece abstração de banco de dados de uma forma simples de usar. Um componente de formulários que implementa e renderiza HTML5, validação e filtragem de modo que os desenvolvedores possam consolidar todas essas operações. Outros componentes, como o *Zend Auth* e *Zend Acl*, fornecem autenticação e autorização do usuário em todos os ambientes do sistema (ZEND, 2014). Inclui também diversos componentes de integração com as API's mais utilizadas no mercado, como os serviços do *Google*, *Facebook*, *Twitter*, *Flickr*, *Amazon*, entre outros.

O ZF é dividido em seis categorias de mais alto nível:

- Autenticação e acesso: Responsável pela segurança de autenticação de usuários e controle de acesso;
- Internacionalização: Permite fazer a utilização de idioma, moeda, data e hora corretas de acordo com a localização;
- Comunicação entre aplicações: Componentes que permite a leitura de dados de aplicações de terceiros;
- Serviços de *Web*: Comunicação com API's ou serviços públicos de outros proprietários;
- Principal (core): Componentes variados do ZF, incluem classes para criação de formulário, envio de e-mail, sistema de *logs* e criação de PDF;
- MVC: Os componentes que constituem a arquitetura MVC.

Na Figura 3, mostra-se as categorias do Zend e os componentes que fazem parte de cada uma delas.

Figura 3 – Componentes do *Zend Framework* e suas categorias



Fonte: (ALLEN, 2009)

Como citado anteriormente, o ZF fornece ao programador uma gama de componentes para utilização na aplicação. Além dos componentes principais *Zend_Controller*, *Zend_View* e *Zend_Layout* que são os componentes responsáveis por implementar a arquitetura MVC do sistema. Pode-se destacar outros componentes, como mostrado na Tabela 2.

Tabela 2 – Principais componentes do *Zend Framework*

Componente	Descrição
Zend_Form	Criação de formulários dinâmicos
Zend_Auth	Autenticação e armazenamento de dados na sessão
Zend_Json	Conversor e servidor JSON
Zend_GData	Integração com API do Google
Zend_Mail	Envio de e-mail
Zend_Pdf	Criação de documentos no formato PDF
Zend_Validate	Regra de validação de dados
Zend_Log	Geração de log de erros (debug)
Zend_Captcha	Criação de Captcha
Zend_Cloud	Recursos de computação em nuvem

Fonte: autoria própria

Além dos componentes, o *framework* fornece o *Zend Tool*, um ambiente de criação a partir de linhas de comando, que auxilia na criação da estrutura do projeto. Oferece também

uma plataforma de desenvolvimento com o nome de *Zend Studio*, uma IDE totalmente adaptada para criação de projetos com utilização do *framework*.

3. SISTEMA CMS

Aplicação CMS ou *Content Management System*, consiste em um sistema que permite a criação, edição e publicação de informações de forma organizada (sejam elas textos, imagens, gráficos, áudio ou vídeo), a fim de auxiliar o gerenciamento de conteúdo de um *website*. Segundo PEREIRA (2002), "a ideia básica por trás de um CMS é a de separar o gerenciamento do conteúdo do *design* gráfico das páginas que apresentam o conteúdo. O *design* das páginas que apresentam os conteúdos são colocados em arquivos chamados moldes (*templates*), enquanto o conteúdo é armazenado em banco de dados ou arquivos separados".

A utilização de um sistema CMS permite melhor organização das informações, redução de custos de manutenção, automação de tarefas, redução da complexidade de operação e da necessidade de suporte e treinamento. Pelos benefícios da implantação do sistema CMS e pelo aumento da complexidade das aplicações, esse tipo de ferramenta é imprescindível para qualquer projeto *Web*.

Tomaremos um sistema CMS como objeto de estudos, a partir do qual são desenvolvidas duas versões: a primeira implementada sem nenhuma abordagem avançada de desenvolvimento, tal como padrões de projeto ou *frameworks* e a segunda versão fazendo utilização das técnicas em estudo. As duas versões serão desenvolvidas para plataforma *Web*, utilizando a linguagem PHP (*Hypertext Preprocessor*). Com o objetivo de estudar e avaliar o uso de um *framework*, a segunda versão do sistema utilizará técnicas da arquitetura MVC, juntamente com o *Zend framework*.

3.1. Ferramentas de Desenvolvimento

Para desenvolver a aplicação foi utilizada a linguagem de programação PHP, além do *framework Twitter Bootstrap*[4] para implementação da interface e componentes do sistema, atuando em conjunto com *Javascript* e sua biblioteca *JQuery* para a realização dos eventos da aplicação.

Para o desenvolvimento da aplicação usou-se as ferramentas mostradas na Tabela 3.

[4] http://getbootstrap.com/

Tabela 3 – Ferramentas de Desenvolvimento do sistema CMS

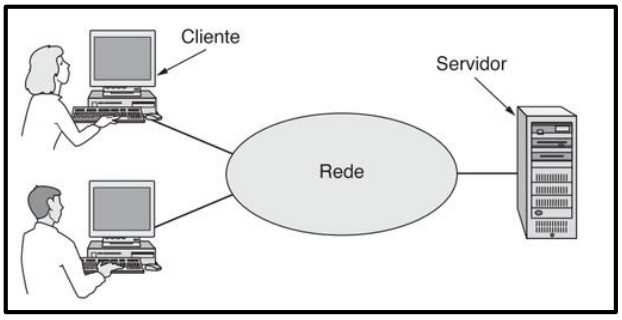
Ferramenta	Descrição
IDE PHPStorm 8.0.1	IDE para desenvolvimento PHP
Xampp Server 5.6.3	Ambiente de desenvolvimento PHP
Apache 2.4.10	Servidor Web
MySQL	Sistema de gerenciamento de banco de dados (SGBD)

Fonte: autoria própria

3.2. Visão Arquitetural

A aplicação CMS foi desenvolvida na estrutura de cliente-servidor, que consiste em uma estrutura distribuída, que divide os fornecedores do serviço, o servidor e os requisitantes dos serviços, designados como clientes. Segundo TANEMBAUM (2011), nesse modelo, os dados são armazenados em computadores chamados de servidores, que normalmente são mantidos em um local central por um administrador de sistemas. Por outro lado, os clientes são computadores que fazem requisições aos servidores e acessam dados remotos. As máquinas cliente e servidor são conectadas entre si por uma rede conforme apresenta-se na Figura 4.

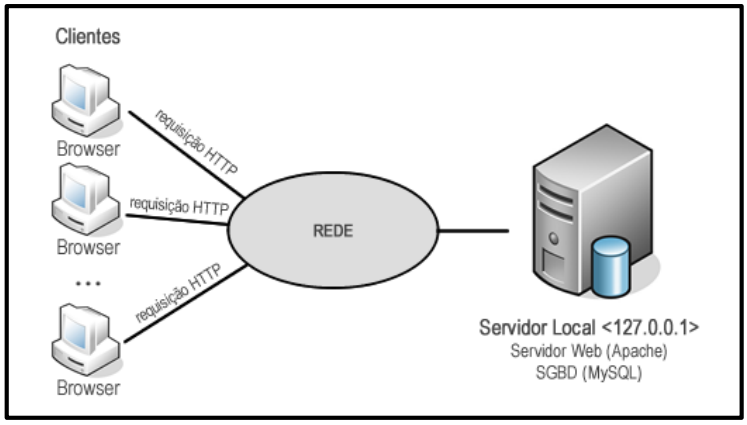
Figura 4 – Modelo cliente-servidor



Fonte: TANEMBAUM (2011)

Para a modelagem do banco de dados do sistema, foi adotado o *MySQL* como SGBD, que estará instalado no servidor local, na mesma máquina, juntamente com o servidor *Web* Apache necessário para hospedar a aplicação e fornecer acesso aos clientes por meio de requisições HTTP. A arquitetura básica de cliente-servidor no ambiente Web é mostrado na Figura 5.

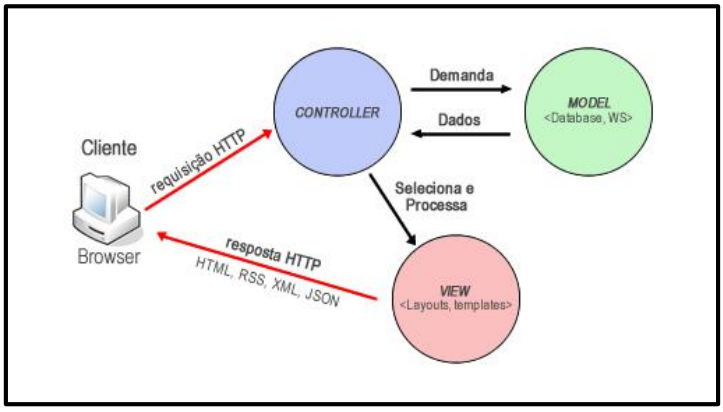
Figura 5 – Arquitetura da aplicação CMS



Fonte: autoria própria

Em termos de arquitetura de *software*, para a segunda versão do sistema, na qual faz uso de técnicas de desenvolvimento, o *Zend Framework* fornece uma avançada estrutura inicial baseada no padrão de projeto de arquitetura MVC, no qual faz a divisão do sistema em três camadas: o modelo (*model*), responsável pelo acesso a dados, sejam eles alocados em banco de dados, *webservices* ou API's externas, toda as regras de negócios e lógica do sistema; a visão (*view*) que compreende toda a parte visual da aplicação; e o controle (*controller*) responsável por intermediar as outras duas camadas. Na Figura 6, mostra-se um modelo que representa a estrutura da arquitetura MVC.

Figura 6 – Arquitetura MVC



Fonte: autoria própria

3.3. Requisitos do Sistema

Segundo SOMMERVILLE (2003), nessa atividade, os membros da equipe de desenvolvimento trabalham com o cliente e usuários finais para descobrir mais informações

sobre o domínio da aplicação. Por exemplo: que serviços o sistema deve fornecer, o desempenho exigido pelo sistema, as restrições de hardware e assim por diante.

Os requisitos do sistema foram definidos a partir da necessidade de gerenciamento do conteúdo de um *website* institucional fictício. Foram levantados os seguintes requisitos principais:

- Controle de acesso de usuários;
- Cadastro de usuários permitindo as operações CRUD;
- Cadastro de portfólios com fotos permitindo as operações CRUD;
- Cadastro de marcas permitindo as operações CRUD;
- Cadastro de frases permitindo as operações CRUD;
- Cadastro de funcionários permitindo as operações CRUD;
- Cadastro de vídeos permitindo as operações CRUD;
- Formulário de suporte ao usuário;
- Estatísticas de acesso do website;
- A aplicação deverá ser adaptável e funcional para todos tipos de resoluções, incluindo dispositivos móveis;
- A aplicação deverá ter alta disponibilidade.

3.4. Casos de Uso

De acordo com SOMMERVILLE (2011), a modelagem de caso de uso é amplamente usada para apoiar a elicitação de requisitos. Um caso de uso pode ser tomado como um cenário simples que descreve o que o usuário espera de um sistema. Cada caso de uso representa uma tarefa discreta que envolve a interação externa com o sistema. Um exemplo de especificação de um caso de uso é mostrado na Tabela 4.

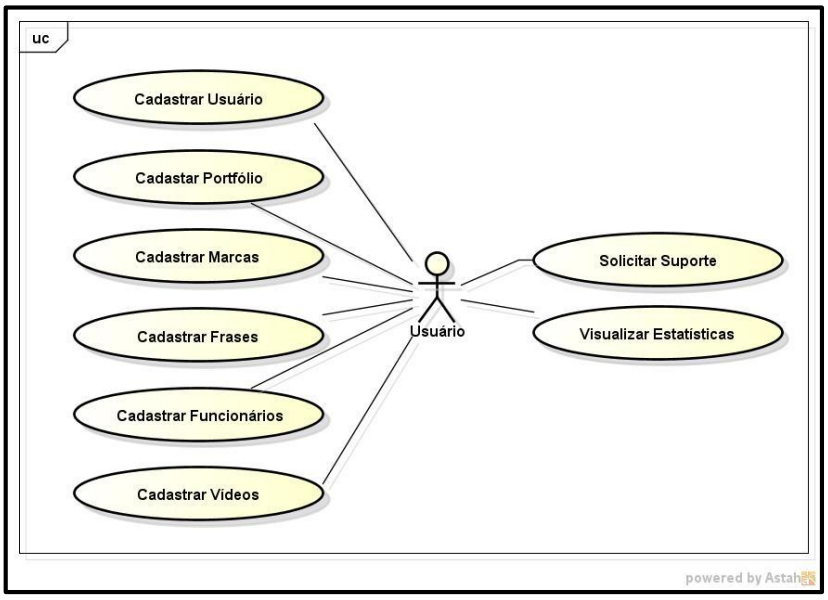
Tabela 4 – Especificação do caso de uso Cadastrar Funcionários

Nome do caso de uso:	Cadastrar Funcionários
Ator principal:	Usuário
Pré-condição	Estar autenticado no sistema
Pós-condição:	Código do funcionário gerado
Prioridade:	Alta
Fluxo Principal	Fluxo de Exceção
P1. Logar no sistema **P2.** Acessar tela de listar Funcionários **P3.** Selecionar cadastrar/alterar funcionário **P4.** Fornecer dados do formulário **P5.** Confirmar cadastro **P6.** Gravar dados	**E1.** Campos obrigatórios em branco a) Alerta para o usuário preencher os campos obrigatórios **E2.** Campos com caracteres inválidos a) Alerta para o usuário que existem caracteres inválidos

Fonte: autoria própria

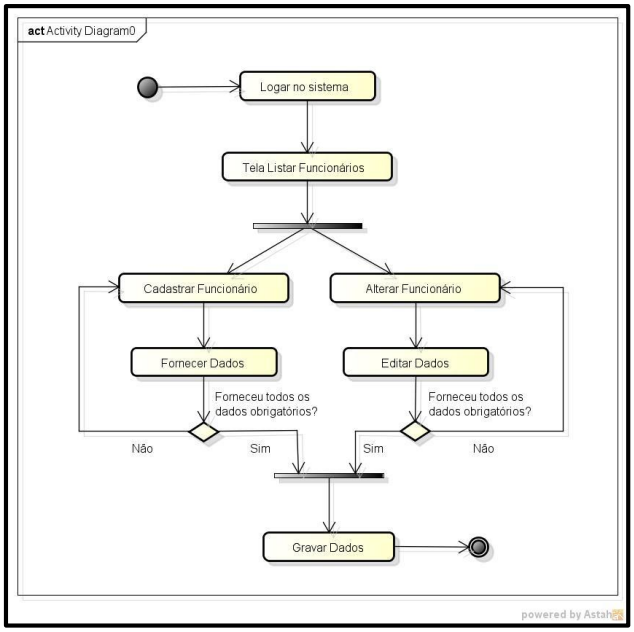
Para o sistema CMS, teremos o usuário que será o operador do sistema, e se relaciona aos casos de uso de todos os cadastros, além de visualizar as estatísticas e solicitação do suporte, como é mostrado no diagrama da Figura 7. Na Figura 8, mostra-se o diagrama de atividade do caso de uso Cadastras Funcionários.

Figura 7 – Diagrama de Caso de Uso da aplicação CMS



Fonte: autoria própria

Figura 8 – Diagrama de atividades para os casos de uso Cadastrar Funcionários



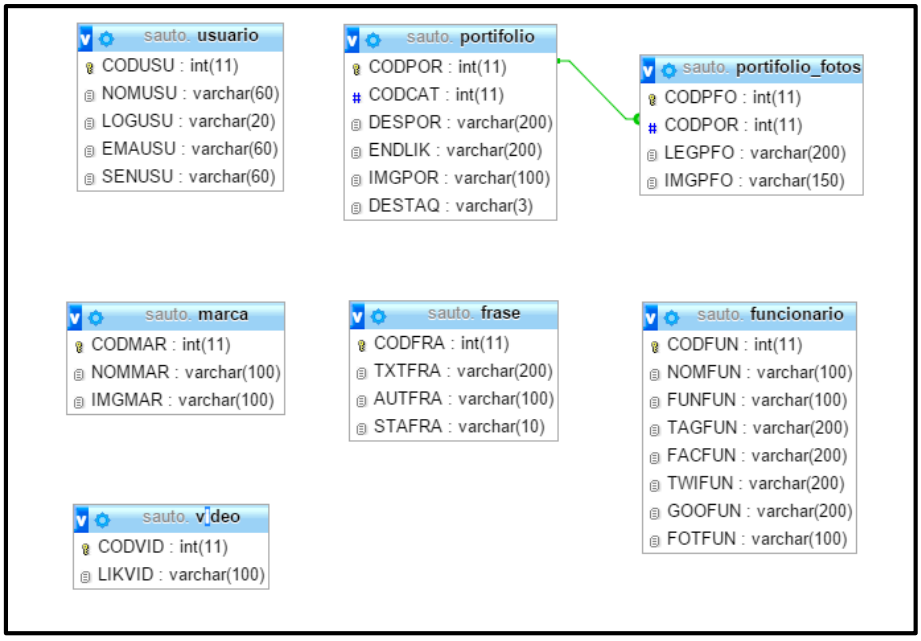
Fonte: autoria própria

3.5. Modelo Relacional

O modelo relacional foi proposto por Edgar Codd em 1970, como uma nova maneira de representação de dados. Neste seu trabalho Codd mostrou que uma visão relacional dos dados permite a sua descrição em uma maneira natural, sem que sejam necessárias estruturas adicionais para sua representação, provendo uma maior independência dos dados em relação aos programas (MACÁRIO, 2005).

Como citado anteriormente, para a modelagem do banco de dados a aplicação utiliza o SGBD *MySQL* e sua estrutura é representada na Figura 9.

Figura 9 – Modelo relacional do banco de dados do sistema CMS



Fonte: autoria própria

3.6.Versão 1: Sem Técnicas Avançadas de Programação

A primeira versão foi feita em PHP estruturado de forma artesanal, sem utilização de *frameworks*, ou padrões de projeto. A aplicação foi desenvolvida com uma técnica de programação básica, na qual o código de programação PHP é misturado com linguagem de marcação HTML, isso é permitido pelo PHP, pois o *script* é pré-processado pelo servidor, gerando um código-fonte e posteriormente interpretado pelo *browser* no lado do cliente. Pode-se conferir na Listagem 5, um trecho de código do sistema, fazendo um laço de repetição para exibir uma lista de registros retornados pelo banco de dados.

Listagem 5 – Trecho de código PHP misturado ao código HTML

```
<?php
    $sql = "SELECT CODFUN, NOMFUN, FOTFUN FROM funcionario
            ORDER BY CODFUN DESC";

    try{
        $selecionaFun = $conexao->prepare($sql);
        $selecionaFun->execute();
    }catch(PDOException $e){
        echo 'Erro '. $e->getMessage();
    }

    while($selFun = $selecionaFun->fetch(PDO::FETCH_ASSOC)){
?>

<tr class="taskDesc">
    <td><?php echo $selFun['NOMFUN']; ?></td>
    <td class="taskLogin"><img width="50" src="uploads/<?php echo $selFun['FOTFUN'];?>"></td>
    <td class="taskOptions" style="width:120px;">
        <a href="?pg=alterar_funcionario&codFun=<?php echo $selFun['CODFUN']; ?>" class="tip-top" data-original-title="Alterar">
            <i class="icon-pencil"></i>
        </a>
        <a href="?pg=excluir_funcionario&codFun=<?php echo $selFun['CODFUN']; ?>" class="tip-top" data-original-title="Excluir">
            <i class="icon-remove"></i>
        </a>
    </td>
</tr>

<?php
    }
?>
```

Fonte: autoria própria

Inicialmente, já podemos perceber que esse método apesar de ser uma prática fácil e rápida, compromete a consistência e a organização. Isso dificulta não só o processo de implementação, mas também a manutenção e o reuso do código, uma vez que se o *layout* de *front-end* for alterado, a lógica da programação teria que ser toda refeita, adaptando-se a nova interface.

Outro problema de não fazer utilização de técnicas avançadas, é que o programador fica responsável por adotar e desenvolver suas próprias técnicas, como: organização dos diretórios do projeto, navegação das páginas, conexão com o banco de dados, autenticação e tratamentos de segurança. Isso diminui a produtividade, uma vez que o programador dedica seus esforços a fazer funções genéricas que não fazem parte da aplicação em si, que poderiam ser aproveitadas de projetos anteriores, agilizando o processo de implementação. Pode-se verificar na Listagem 6, um trecho de código responsável pela navegação entre as páginas.

Listagem 6 – Trecho de código responsável pela navegação

```php
<?php

    //Suporte
    if($_GET['pg'] == 'suporte'){
        require('paginas/suporte/suporte.php');
    }


    //Usuário
    if($_GET['pg'] == 'inserir_usuario'){
        require('paginas/usuario/inserir_usuario.php');
    }elseif($_GET['pg'] == 'alterar_usuario'){
        require('paginas/usuario/alterar_usuario.php');
    }elseif($_GET['pg'] == 'excluir_usuario'){
        require('paginas/usuario/excluir_usuario.php');
    }elseif($_GET['pg'] == 'lista_usuario'){
        require('paginas/usuario/lista_usuario.php');
    }

?>
```

Fonte: autoria própria

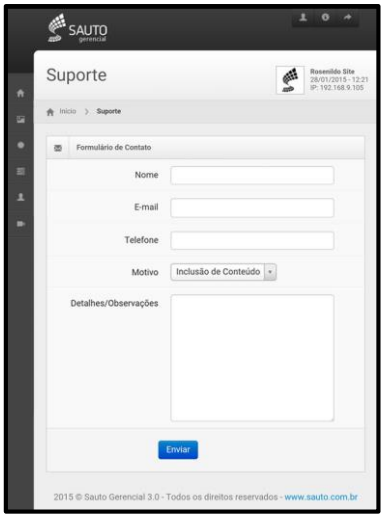
Na Figura 10, mostra-se a tela inicial de *login* da aplicação CMS acessada por um navegador de internet em uma plataforma *Desktop*.

Figura 10 – Tela de *login* do sistema CMS

Fonte: autoria própria

Na Figura 11, mostra-se a tela de solicitação de suporte técnico acessada por um navegador móvel.

Figura 11 – Versão *mobile* da página de Suporte

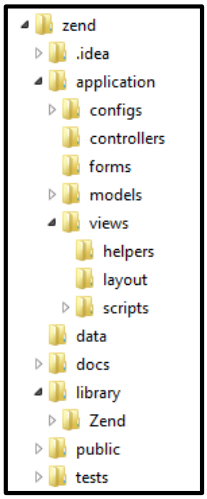


Fonte: autoria própria

3.7. Versão 2: Utilizando Framework MVC Zend

A segunda versão da aplicação CMS foi implementada com o *framework* MVC *Zend Framework*. Primeiramente, é preciso fazer download do pacote no site oficial do *framework*[5] e, através da *Zend Tool* e criar um novo projeto, oferecendo uma estrutura básica para o desenvolvimento. A versão utilizada foi a 1.12.9 (atualmente na 2.3.3). Na Figura 12, mostra-se a estrutura criada pelo *framework*. Observa-se que com essa estrutura, permite-se a organização dos arquivos do projeto de acordo com suas finalidades.

Figura 12 – Estrutura básica fornecida pelo *framework*



Fonte: autoria própria

[5] http://framework.zend.com/

Dentre os diretórios criados, destacamos o diretório *application*, que deve conter os arquivos de configuração, controles, formulários, modelos e *views* da aplicação. Nesse último deve ser criado um *layout* do sistema, e os scripts que serão as páginas dinâmicas criadas. O diretório *library* contém todos os arquivos de módulos do *Zend Framework*. No diretório *public*, ficarão os arquivos de domínio público do sistema, como: *javascripts*, folha de estilos e imagens utilizadas no *layout*.

O grupo de classes do *Zend_Controller* contém os métodos *actions*, que são ações definidas para aquele controlador, podendo ou não estar associadas à uma página de visão. Para a aplicação CMS devem ser implementadas uma *action* para cada operação: cadastrar, alterar e excluir, além da *indexAction* que por padrão é a página principal do controlador, utilizada para listar os registros através de uma *view*. Naturalmente, cada controlador deve ser associado a uma *view*, e cada página da *view* associada a uma *action*. Por padrão, o projeto disponibiliza o *IndexController* e a *view* index, que corresponde à página inicial do sistema. Pode-se observar na Listagem 7 o exemplo de uma classe *Zend_Controller*. A classe tem os métodos index, cadastrar e alterar, cada um com uma *view* associada, além do método excluir sem nenhuma página *view*, fazendo somente a operação de exclusão.

Listagem 7 – Código-fonte do *controller* Usuarios, suas *actions* e as páginas da *view*

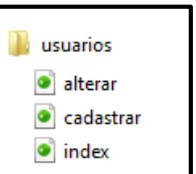
```
class UsuariosController extends Zend_Controller_Action
{

    public function indexAction()
    {
        $model = new Application_Model_Usuarios();
        $dados = $model->select();
        $this->view->assign("dados", $dados);
    }

    public function cadastrarAction(){}

    public function alterarAction(){
        $model = new Application_Model_Usuarios();
        $usuario = $model->find($this->_getParam('id'));
        $this->view->assign("usuario", $usuario);
    }

    public function excluirAction(){
        $model = new Application_Model_Usuarios();
        $model->delete($this->_getParam('id'));
        $this->_redirect('usuarios/index');
    }
```



Fonte: autoria própria

Para o controlador requisitar dados do banco de dados, primeiramente cria-se um objeto do seu modelo, chamando seu método *select*, que poderá ter condições SQL na sua chamada,

retornando um *array* com os registros, passando-o posteriormente para sua *view*, por meio da assinatura de um id. Na Listagem 8, pode-se observar os passos descritos representado pela *indexAction*. Observa-se que ainda é embutido código PHP dentro do código HTML. Entretanto, somente é percorrido o *Array* passado para a *view*, não é utilizada nenhuma outra operação que poderia vir a comprometer a consistência código.

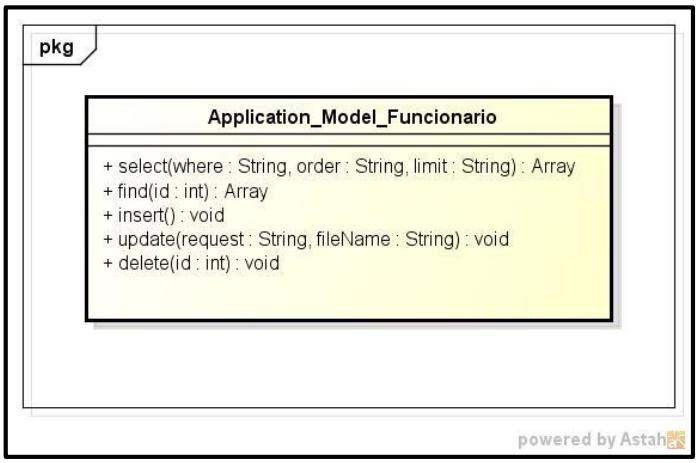
Listagem 8 – Recuperando dados na *view*, passados pelo controlador

```
<?php foreach($this->dados AS $key => $usuario):?>
    <tr>
        <td><?php echo $usuario['NOMUSU']; ?></td>
        <td><?php echo $usuario['LOGUSU']; ?></td>
        <td><?php echo $usuario['EMAUSU']; ?></td>
    </tr>
<?php endforeach; ?>
```

Fonte: autoria própria

De forma contrária, para passar dados da *view* para o controlador, registramos a *tag action* do elemento *form* do HTML com o nome da *action* do *controller* que irá receber os dados, que por sua vez receberá todos os parâmetros passados pela *view* e chamar à um método da classe *model*, para realizar a operação solicitada. Na Figura 13, mostra-se a implementação da classe *model* para o cadastro de funcionários. Em cada método da classe, são implementadas suas operações através das consultas SQL com o banco de dados.

Figura 13 – Classe Funcionario do módulo m*odel* e seus métodos

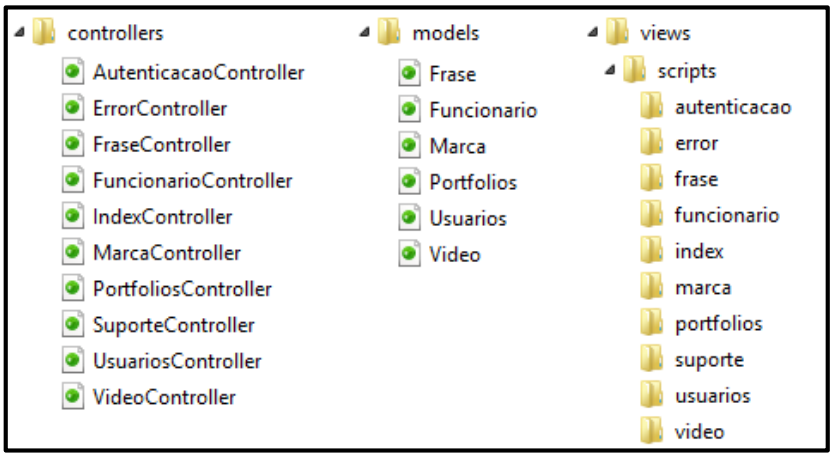


Fonte: autoria própria

Após o processo de criação, o projeto configurado está disponível para começar o desenvolvimento da aplicação. Deve ser adicionado um *layout* e os arquivos necessários para a

aplicação. Com o auxílio do *Zend Tool*, cria-se para cada tabela os *models*, que faz o acesso ao banco de dados, as *views* das páginas do sistema e *controllers*, responsáveis por receber as requisições do usuário, solicitar dados dos *models* e direcionar para *view* correspondente, integrando as outras duas camadas, por meio das *actions* de cada operação. Na Figura 14, mostra-se as pastas e os arquivos implementados de todas as camadas da aplicação.

Figura 14 – Estrutura das camadas MVC do sistema implementadas



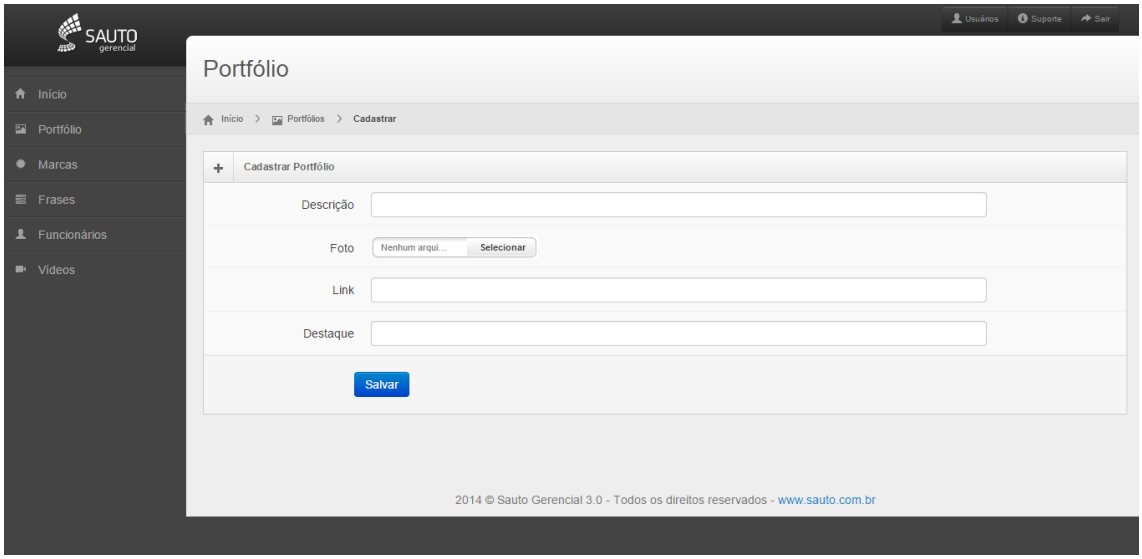
Fonte: autoria própria

Com toda a estrutura MVC implementada, o *Zend Framework* automaticamente configura toda a navegação da aplicação por intermédio dos parâmetros da URL. Dividindo o caminho em *controller*, *action*, e parâmetros, como pode ser observado na Figura 15. Com a navegação configurada a aplicação pode ser acessada pelo *browser*, como mostrado na Figura 16.

Figura 15 – Interpretação do caminho pelo *framework*

Fonte: autoria própria

Figura 16 – Página da aplicação acessada no *browser*



Fonte: autoria própria

4. COMPARANDO AS APLICAÇÕES UTILIZANDO MÉTRICAS

Neste capítulo é apresentado um comparativo entre as aplicações por meio do uso de métricas. As técnicas são aplicadas na implementação das duas versões do sistema, gerando vários dados de acordo com a métrica utilizada. Pelo fato da primeira versão (sem o uso do *Zend Framework*) não ser orientada a objeto, não analisaremos as métricas que levam em consideração as propriedades ou recursos desse paradigma, pois não terão relevância no nosso estudo de comparação.

Nos próximos itens faremos uma breve contextualização das métricas utilizadas e apresentaremos os resultados, juntamente com as conclusões obtidas.

4.1. Métricas de Software

Ao se desenvolver um projeto, surgem várias dificuldades quando se trata de medição da aplicação. Por exemplo, como podemos medir o quão boa é uma aplicação? Ou, quanto cobrar pelo desenvolvimento da aplicação? Qual o nível de confiabilidade da minha aplicação? Para responder essas perguntas, os engenheiros de *software* aplicam as técnicas das métricas de *software*.

Métricas de *software* é uma técnica de engenharia de *software* que auxilia no planejamento do projeto, possibilitando medir custos, esforços, qualidade ou complexidade do sistema. Podem ser divididas em dois tipos: medidas diretas e indiretas. O primeiro procura medir uma quantidade observada, tais como custos, esforços, número de linhas de código, tempo de execução, número de defeitos. Já as medidas indiretas estão relacionadas às funcionalidades do sistema, que exigem uma avaliação mais subjetiva do projeto, e engloba medidas como qualidade, complexidade, eficiência, confiabilidade, entre outras.

Uma métrica de *software* é uma característica de um sistema de *software*, documentação de sistema ou processo de desenvolvimento que pode ser objetivamente medida (SOMMERVILLE, 2011).

Para o estudo de comparação das aplicações foram utilizadas as seguinte métricas:

- ***Cyclomatic Complexity Number* (CC)**: Desenvolvida por McCabe para indicar a testabilidade e manutenibilidade de software, medindo o número de caminhos linearmente independentes do programa (ou método). Para determinar os caminhos, representa-se o método como um grafo fortemente conectado com uma única entrada e

uma única saída. Os nodos são blocos sequenciais de código, e as arestas são decisões que causam uma ramificação (BRUSAMOLIN, 2004);

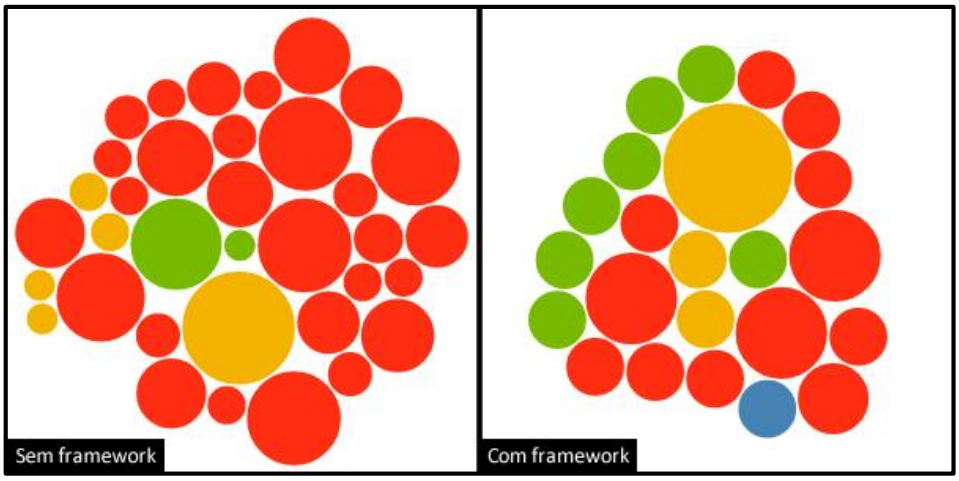
- ***Maintenability Index* (MI)**: Representa a facilidade de manutenção do código, baseado nas métricas de Halstead[6], linhas de código e complexidade ciclomática, quanto mais alto o valor, mais difícil aplicar alterações e reuso no código;
- ***Lines of Code* (LOC):** Número de linha do código;
- ***Logical Lines of Code* (LLOC):** Número de linha lógicas do código;
- ***Volume*:** O número de bits necessário para armazenar o programa. Baseado na métrica de Halstead.

4.2. Análise da Aplicação das Métricas

Para aplicar as métricas nas duas versões do sistema, foi utilizada a ferramenta *open-source PhpMetrics*[7], que automatiza a análise de códigos-fonte e, baseado nas métricas de *software*, gera um relatório completo dos dados extraídos.

A seguir, na Figura 17, apresenta-se o gráfico de análise dos dois sistemas, que representa cada arquivo analisado simbolizado por um círculo, onde seu tamanho é definido de acordo com CC e a cor pelo MI. Os grandes círculos vermelhos representam os recursos de mais dificuldade em manutenção. Podemos observar que a versão sem o uso de técnicas possuem mais páginas com maior complexidade de manutenção.

Figura 17 – Gráfico de análise comparativo dos sistemas

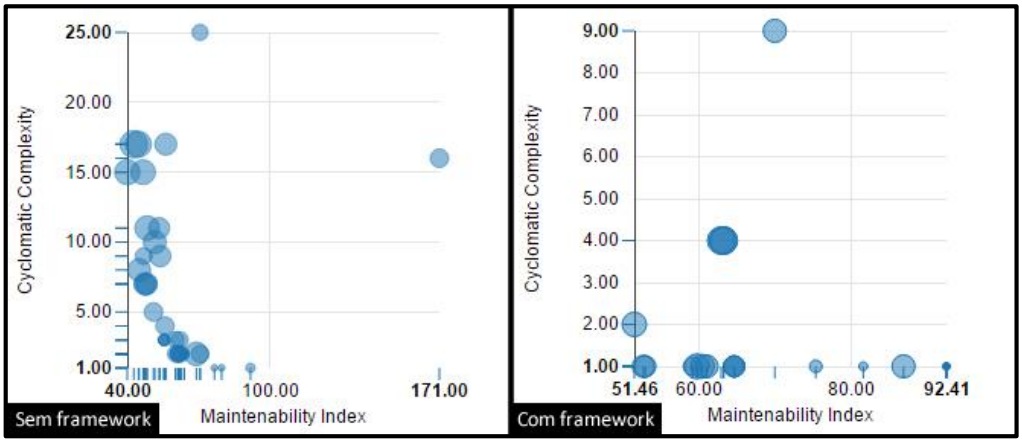


Fonte: autoria própria

[6] http://agileg.wdfiles.com/local--files/projfinalexam/ExameFinal_rev00.pdf
[7] http://www.phpmetrics.org/

Outra forma de análise gerada pelo *PhpMetrics* é o gráfico de *Cyclomatic Complexity* por *Maintenability Index*, representado por um gráfico de dispersão que, a partir do valor das métricas de cada arquivo, apresenta o nível de dificuldade de manutenção de cada um. Observamos que a versão sem *framework* possui valores de MI parecido com sua outra versão, mas quando analisamos os valores de CC, percebemos uma grande diferença entre as versões, como pode ser visto na Figura 18.

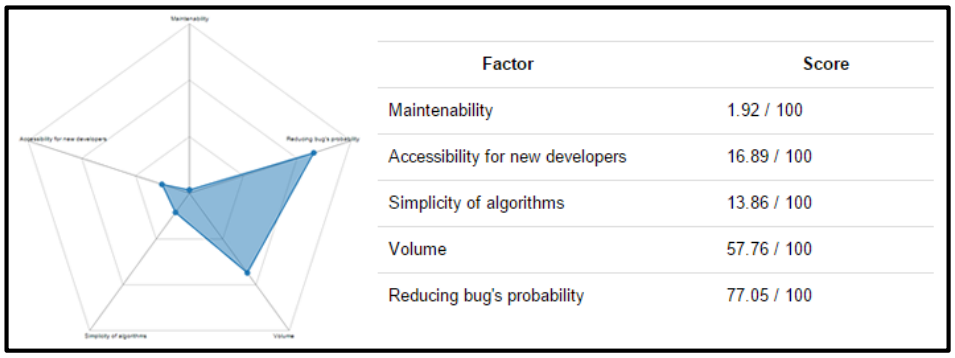
Figura 18 – Gráfico CC/MI comparativo dos sistemas



Fonte: autoria própria

Nas Figuras 19 e 20, mostra-se um resumo geral do projeto, no qual atribui-se uma pontuação para cinco pontos importantes relacionados com as métricas: manutenibilidade, acessibilidade para novos desenvolvedores, simplicidade de algoritmos, volume e redução da probabilidade de erro, e então faz-se uma classificação do projeto de 0 (ruim) a 100 (excelente).

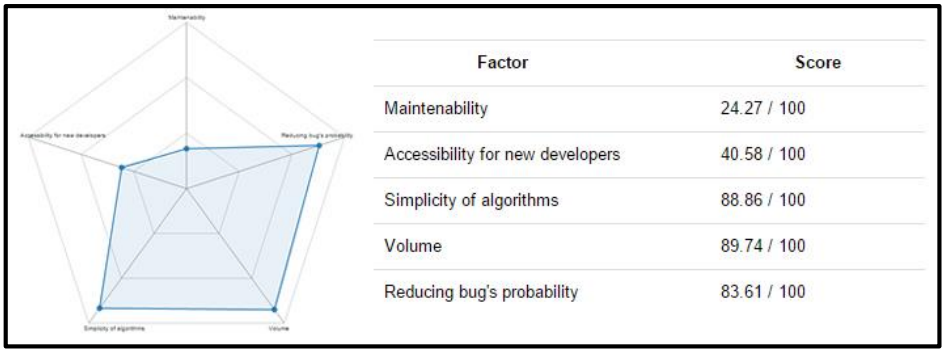
Figura 19 – Resumo para a aplicação sem *framework*



Factor	Score
Maintenability	1.92 / 100
Accessibility for new developers	16.89 / 100
Simplicity of algorithms	13.86 / 100
Volume	57.76 / 100
Reducing bug's probability	77.05 / 100

Fonte: autoria própria

Figura 20 – Resumo para a aplicação com *framework*



Factor	Score
Maintenability	24.27 / 100
Accessibility for new developers	40.58 / 100
Simplicity of algorithms	88.86 / 100
Volume	89.74 / 100
Reducing bug's probability	83.61 / 100

Fonte: autoria própria

Por fim, a ferramenta *PhpMetrics* apresenta a tabela com os valores de métricas obtidos pelos sistemas, como é mostrado na Tabela 5. Observa-se que há uma redução considerável de linhas de código entre as versões do sistema. Pode-se concluir também que pelos valores obtidos em volume, MI e CC, que a versão da aplicação com *framework* possui um *software* de maior facilidade de manutenção.

Tabela 5 – Tabela com pontuação das métricas dos sistemas

Name	loc	lloc	Volume	MI	CC
+ C:/xampp/htdocs/sauto/painel/ (1)	2817	607	763.83	54.83	6.38

Sem framework

Name	loc	lloc	Volume	MI	CC
+ C:/xampp/htdocs/zend/application/ (2)	974	304	582.56	71.2	1.82

Com framework

Fonte: autoria própria

5. CONSIDERAÇÕES FINAIS E TRABALHOS FUTUROS

Pode-se concluir que, apesar das dificuldades de comparação entre duas aplicações de paradigmas diferentes, ao analisar os números obtidos pelas métricas de *software*, é visível o ganho proporcionado pela utilização das técnicas estudadas, principalmente no que diz respeito à manutenção de *software*. Com a utilização de um *framework* pode-se desenvolver um projeto melhor estruturado, reutilizável e compreensível, trazendo várias vantagens em termos de custos e vida útil do *software*.

Durante o processo de desenvolvimento com o *Zend Framework*, percebe-se que todo o esforço de implementação fica focado nas rotinas e regras de negócio do próprio projeto, deixando as funcionalidades alheias a ele para o *framework*.

Portanto, o estudo proposto mostrou que fazendo-se uso de um framework com seus recursos bem definidos, os benefícios são claros em relação a:

- Reuso de código;
- Manutenibilidade;
- Redução de custos;
- Modularidade.

Em contrapartida, também existem desvantagens na adoção dessa ferramenta para a construção de um projeto, tais como: lidar com um código que não é de sua autoria (*Not made here syndrome*), os benefícios geralmente só são vistos a longo prazo e nem sempre é bom utilizar uma ferramenta muito poderosa para realizar uma tarefa muito simples.

Algumas dificuldades encontradas na utilização do *Zend Framework*:

- O *framework* possui uma alta curva de aprendizagem em comparação com outros *frameworks* disponíveis no mercado;
- Para manter um bom processo de desenvolvimento, é recomendável utilizar ferramentas como: *Zend Tool* ou a IDE *Zend Studio*, que automatizam alguns processos de configuração. Caso contrário, o processo manual é bastante oneroso;
- Por ser uma ferramenta robusta e cheia de recursos, considerando também a dificuldade de aprendizado, para um projeto simples o ZF não seria uma boa escolha. *CakePHP* ou *CodeIgniter* seriam boas alternativas;
- É indicado também, o conhecimento de ferramentas como Composer e Github para uma melhor experiência com o *framework*.

Como trabalhos futuros, sugere-se:

- O estudo de outras alternativas de frameworks. Nomes como: *Laravel*, *CakePHP*, *Codeigniter* e *Symfony* são outras opções consolidadas e de grande aceitação do mercado;
- O aprofundamento dentre os inúmeros módulos do *Zend Framework*, em especial os módulos de integração com API's, do Google por exemplo, e dos módulos disponíveis para construção de *webservices*, muito utilizados atualmente. O ZF é uma ferramenta muito poderosa e requer um estudo minucioso para o domínio de todos seus recursos;
- Para carreira profissional, é imprescindível o emprego das técnicas de desenvolvimento ou padrões de projeto. Isso garantirá o desenvolvimento de produtos com maior qualidade, mais confiáveis e reutilizáveis, além de facilitar todo o processo de desenvolvimento.

REFERÊNCIAS

ALLEN, R.; LO, N.; BROWN, S. **Zend Framework in action**. 1. ed. Manning, 2009.

APPLETON, B. **Patterns and Software - Essential Concepts and Terminology**. Disponível em: <http://csis.pace.edu/~grossman/dcs/Patterns and Software- Essential Concepts and Terminology.pdf>. Acesso em: 27 de janeiro de 2015

BALDWIN, C. Y.; CLARK, C. B. **Design Rules – The Power of Modularity**. 1. ed. Cambridge: The MTI Press, 2000.

BONOTO, R. G.; JUNIOR, E. A. O. **O Padrão Model-View-Controller Apoiado pelo Framework Zend.** Disponível em: <http://www.espweb.uem.br/site/files/tcc/2012/Rodrigo Guimaraes Bonoto - O padrao Model-View-Controller apoiado pelo framework Zend.pdf>. Acesso em: 27 de setembro de 2014.

BRUSAMOLIN, V. **Manutenibilidade de Software.** Disponível em: <http://revdigonline.com/revistas_download/revista_2.pdf>. Acesso em: 23 de janeiro de 2015.

DALL'OGLIO, P. **PHP – Programando com Orientação a Objetos**. 2. ed. Novatec, 2009.

FAYAD, M. E.; SCHMIDT, D. C. **Object-oriented Application frameworks.** Communications of the ACM, 1997.

FERRO NETO, J. S. **Registro de informações e imagens em banco de dados mamográficos.** Disponível em: <https://uspdigital.usp.br/siicusp/cdOnlineTrabalhoVisualizarResumo?numeroInscricaoTrabalho=1326&numeroEdicao=19>. Acesso em: 18 de janeiro de 2015.

GAMMA, E.; HELM, R.; JOHNSON, R.; VLISSIDES, J. **Design Patterns – Elements of Reusable Object-Oriented Software.** Addison-Wesley, 1995.

JOHNSON, R. E. **Reusing Object-Oriented Design.** University of Illinois, Technical Report, 1991.

LISBOA, F. G. S. **Zend Framework – Desenvolvendo em PHP 5 orientado a objetos com MVC**. 1. ed. [S.l.]: Novatec, 2008.

LOCKHART, J. **PHP: Do Jeito Certo.** Disponível em: <http://www.phptherightway.com/>. Acesso em: 18 de janeiro de 2015.

MACÁRIO, C. G. N.; BALDO, S. M. **O Modelo Relacional.** Disponível em: <http://www.ic.unicamp.br/~geovane/mo410-091/Ch03-RM-Resumo.pdf>. Acesso em: 23 de novembro de 2014.

MILANI, A. **MySQL - Guia do Programador**. 1. ed. [S.l.]: Novatec, 2007.

MINETTO, E. L. **Frameworks para Desenvolvimento em PHP**. 1. ed. [S.l.]: Novatec, 2007.

NIEDERAUER, J. **Integrando PHP 5 com MySQL**. 2. ed. Novatec, 2008.

NIEDERAUER, J. **PHP para quem conhece PHP**. 3. ed. Novatec, 2008.

PEREIRA, J. C. I.; BAX, M. P. **Introdução à Gestão de Conteúdos.** Disponível em: <http://revistagt.fpl.edu.br/get/article/view/104/103>. Acesso em: 18 de novembro de 2014.

PILGRIM, M. **HTML5 - Entendendo e Executando**. 1. ed. Alta Books, 2011.

PREE, W. **Desing Patterns for Object-Oriented Software Development.** Addison-Wesley, 1995.

SAUVÉ, J. P. **O que é um framework?** Disponível em: <http://www.dsc.ufcg.edu.br/~jacques/cursos/map/html/frame/oque.htm>. Acesso em: 18 de janeiro de 2015.

SCHMITT, C. **CSS Cookbook - Soluções Rápidas para Problemas Comuns com CSS**. 1. ed. Novatec, 2010.

SILVA, P. H. **Estudo e implementação de Web Services utilizando Zend Framework.** Disponível em: <http://repositorio.roca.utfpr.edu.br/jspui/bitstream/1/606/1/MD_COADS_2011_2_11.pdf>. Acesso em: 06 de outubro de 2014.

SOMMERVILLE, I. **Engenharia de software.** 8. ed. São Paulo: Pearson Addison-Wesley, 2003.

SOMMERVILLE, I. **Engenharia de software.** 9. ed. São Paulo: Pearson Prentice Hall, 2011.

TANENBAUM, A. S.; WETHERALL, D. **Redes de Computadores.** 5. ed. São Paulo: Pearson Prentice Hall, 2011.

ZEND. **Zend Framework.** Disponível em: <http://framework.zend.com/>. Acesso em: 27 de setembro de 2014.