Protótipo de um framework MVC para aplicações PHP de pequeno porte

Eder Martins Franco[1,2], Márcio Palheta Piedade[1,2], Renata Magalhães Rêgo[3]

[1]Faculdade FUCAPI
Av. Gov. Danilo de Matos Areosa, 381 – Distrito Industrial CEP.: 69.075-351 – Manaus – AM – Brasil

[2]FPF Tech
Av. Gov. Danilo de Matos Areosa, 1170 – Distrito Industrial CEP.: 69.075-351 – Manaus – AM – Brasil

[3]Instituto de Computação – Universidade Federal do Amazonas (UFAM)
Av. Gen. Rodrigo Otávio Jordão Ramos, 3000 CEP.: 69077-000 Manaus – AM – Brasil

{efranco23,marcio.palheta,renatamagalhaesrego}@gmail.com

Abstract. *PHP frameworks help in web development because they make it easier to organize and modularize projects, besides they offer native implementation of components that are commonly used in the development of projects. However, the usage of complex frameworks in short time projects or in projects with inexperienced teams can greatly increase the development effort. This paper shows the building of a prototype of MVC framework for small PHP applications, which presented 51,1% of improvement in the development time of a small sized website.*

Resumo. *Frameworks PHP auxiliam no desenvolvimento web por facilitarem a organização e modularização dos projetos, além de oferecerem implementação nativa de componentes comuns ao desenvolvimento dos projetos. Entretanto, o uso de frameworks complexos em projetos curtos ou com times inexperientes pode aumentar muito o esforço de desenvolvimento. Este trabalho apresenta a construção de um protótipo de framework MVC para aplicações PHP de pequeno porte, que apresentou melhoria de 51,1% no tempo de desenvolvimento de um website de pequeno porte.*

1. Introdução

PHP é uma linguagem de script para desenvolvimento web amplamente utilizada, que ganhou força pela simplicidade de aprendizado por novos programadores. Criada como uma linguagem estruturada, seu amadurecimento permitiu que novos elementos fossem adicionados, como o suporte ao paradigma de orientação a objetos (a partir da versão 4), que atraiu a atenção de programadores mais experientes porque passou a oferecer recursos que permitiam o desenvolvimento de aplicações mais robustas [Minetto 2007].

O curso natural da evolução da linguagem e da maneira como ela é utilizada pelos desenvolvedores levou ao surgimento de frameworks, que são um conjunto de funções, códigos, classes e outros utilitários que ajudam a organizar o desenvolvimento

 ISSN 2238-5096 (CDR)

de novos softwares [Minetto 2007], disponibilizando bibliotecas de código que objetivam aumentar o reuso, levando ao desenvolvimento mais ágil de aplicações web [Lancor e Katha 2013].

Frameworks PHP comumente utilizam o padrão arquitetural MVC (*model-view-controller*), que oferece benefícios como a produção de um código mais limpo, facilidade para realização de upgrades, divisão bem definida da manutenção do código entre os desenvolvedores e uma subdivisão utilitária da estrutura de arquivos [McArthur 2008].

A escolha de um framework deve ser feita levando em consideração o uso de ferramentas que o desenvolvedor consiga compreender e que lhe sejam realmente úteis e necessárias. É comum que muitos desenvolvedores escolham utilizar frameworks MVC complexos porque ouviram falar de algum dos seus benefícios, mas desconhecem ou subutilizam elementos de sua arquitetura. Desta forma, é necessário estudar e conhecer os requisitos técnicos dessas poderosas ferramentas. [McArthur 2008].

Este trabalho apresenta um protótipo de framework PHP para utilização em projetos de pequeno porte, como websites institucionais, blogs e páginas de pessoais, com uma proposta mais simples e de fácil aprendizado, conforme experimentos realizados. Na seção 2 são apresentados os conceitos básicos do MVC e sua utilização nos frameworks PHP. Na seção 3 é feita uma análise de alguns dos principais frameworks PHP, principais características, e sua estrutura de diretórios. Na seção 4 é apresentado o protótipo de framework que foi desenvolvido. Na seção 5 é apresentado um estudo de caso da utilização do framework criado, e é feita uma análise dos resultados obtidos em comparação com as outras soluções pesquisadas.

2. Técnicas e padrões de projeto

Para analisar os frameworks que já existem no mercado é necessário compreender o que é um framework. Segundo Minetto (2007), "um framework de desenvolvimento é uma 'base' de onde se pode desenvolver algo maior ou mais específico. É uma coleção de códigos-fonte, classes, funções, técnicas e metodologias que facilitam o desenvolvimento de novos softwares".

Para Paikens e Arnicans (2008), um framework de software é um projeto reutilizável para um sistema de software, que pode incluir programas de apoio, bibliotecas de código, uma linguagem de script ou outros softwares que podem ajudar a desenvolver e unir diferentes componentes de um projeto de software.

2.1 MVC

O MVC (*model-view-controller*) é um padrão de arquitetura criado para aumentar a modularidade de sistemas de software [Aihara 2009], dividindo as responsabilidades nas seguintes camadas:

1) *model* (modelo), a camada que gerencia os modelos de dados da aplicação, persistência de informações e o acesso aos dados em si;
2) *view* (visualização), que são as interfaces gráficas, as telas da aplicação propriamente ditas, sendo a camada de comunicação com o usuário, manipulação e exibição dos dados;

3) *controller* (controlador), a camada que realiza a comunicação da *view* com o sistema, recebendo e redirecionando requisições, decidindo quais *views* exibir, e realizando requisições para a camada de modelo;

2.2 Front Controller

O *front controller* é um padrão de projeto utilizado na camada de controle, especializado em receber, tratar e despachar todas as requisições feitas pela *view*, atuando como um ponto de entrada único para as requisições do sistema.

2.3 Bootstraping

Aplicações PHP que implementam MVC são configuradas de forma a centralizar todas as requisições em um único script que é responsável por inicializar o framework, uma técnica chamada de *bootstrapping*. Este script carrega arquivos e informações de configuração, instancia objetos globais, inicializa o *front controller* e os demais *controllers* do framework [McArthur 2008].

3. Frameworks MVC em PHP

Por causa da popularidade da linguagem entre os desenvolvedores de aplicações web, existe uma grande variedade de frameworks PHP disponíveis no mercado. Neste tópico será detalhado como foi realizada a definição dos frameworks estudados para análise das características apresentadas no tópico anterior**.**

3.1 Definição do objeto de estudo

Após levantamento realizado em sites especializados em frameworks para PHP, foram identificados os cinco frameworks mais populares entre a comunidade: Codeigniter, CakePHP, Zend Framework, Symphony e Laravel. À exceção do Laravel (lançado no ano de 2011), estes frameworks foram lançados entre 2005 e 2006, e têm muitas características semelhantes, utilizando em sua arquitetura os padrões MVC e *front controller*.

A figura 1 mostra os resultados de uma pesquisa realizada no site Google Trends em fevereiro de 2014, comparando o nível de popularidade e interesse destes cinco frameworks desde o ano de 2005 até fevereiro de 2014:

Figura 1. Gráfico de popularidade dos framrworks PHP no Google Trends

O gráfico demonstra que nos últimos anos houve um crescimento do interesse pelos frameworks menores e mais compactos (Laravel, Cake e Codeigniter) em relação aos outros maiores e mais tradicionais (Zend e Symphony). A seguir, serão apresentadas as principais características destes frameworks.

3.2 Principais características

Desenvolver aplicações web utilizando um framework tem suas vantagens e desvantagens. Segundo McArthur (2008), pelo menos cinco características principais precisam ser observadas, que são: 1) arquitetura: a maneira como o framework implementa o padrão MVC, suas técnicas, requisitos e estrutura de arquivos; 2) documentação: se está atualizada, se é completa, clara, compreensível e objetiva; 3) flexibilidade: uma análise do quão fácil é integrar códigos do próprio desenvolvedor; 4) comunidade: quanto mais pessoas utilizam o framework, maior será sua manutenção; e 5) suporte: existe alguém de quem se pode obter resposta quando precisar de suporte?

Neste contexto, foram delimitados os seguintes itens para análise em cada um dos frameworks estudados: A. Conhecimentos necessários para utilização; B. Tamanho de projeto recomendado; C. Complexidade de instalação e configuração; D. Existência de documentação e exemplos e E. Suporte e comunidade de desenvolvedores,

A partir de uma pesquisa publicada Romakenko (2013) e do estudo da documentação dos próprios frameworks foi construída a tabela 1, onde os frameworks em estudo são classificados de acordo com as características escolhidas:

Característica / Framework	A	B	C	D	E
Zend	PHP5, POO, Padrões de Projeto	Médios e grandes	Alta	Técnica	Ampla
Codeigniter	PHP, POO	Pequenos a grandes	Baixa	Boa	Ampla
CakePHP	PHP, POO	Pequenos a médios	Baixa	Boa	Ampla
Symfony	PHP5, POO, linha de comando	Grandes	Alta	Técnica	Média
Laravel	PHP5, POO, linha de comando	Pequenos a médios	Médio	Didática	Crescente

Tabela 1. Análise das características dos frameworks pesquisados

A partir destes dados é possível observar que os frameworks possuem características semelhantes no que se refere ao nível de conhecimentos necessário para sua utilização, tendo como ponto comum e central a necessidade de que o usuário tenha conhecimentos em programação orientada a objetos. Os frameworks mais antigos no Mercado (Symfony e Zend) exigem conhecimentos mais avançados, como um bom domínio dos conceitos de padrões de projetos e a utilização scritps por linha de comando.

Quanto ao porte das aplicações desenvolvidas, os frameworks estudados conseguem atender a projetos com diferentes tamanhos. Possuem uma comunidade ativa, ainda que crescente em alguns casos, e disponibilizam acesso a uma documentação que varia de um nível mais didático (Laravel) a um nível mais técnico e complexo (Symfony).

Quanto à complexidade de instalação, observou-se que todos os frameworks que possuem mais simples são aqueles apontados como crescentes no que se refere à popularidade (Laravel e CodeIgniter).

3.2 Estrutura de diretórios

Uma característica comum na estrutura de diretórios dos frameworks é a separação das classes básicas do MVC do restante dos componentes do sistema, como bibliotecas auxiliares, arquivos de *layout* e bibliotecas de terceiros. Também são isoladas as *core libraries*, que são as bibliotecas centrais do framework, compostas por classes mais robustas e que são responsáveis por tarefas específicas como tratamento de requisições e manipulação de arquivos.

A figura 2 apresenta e estrutura de diretórios dos cinco frameworks estudados, que possui várias semelhanças entre si, principalmente no que se refere a implementação do MVC.

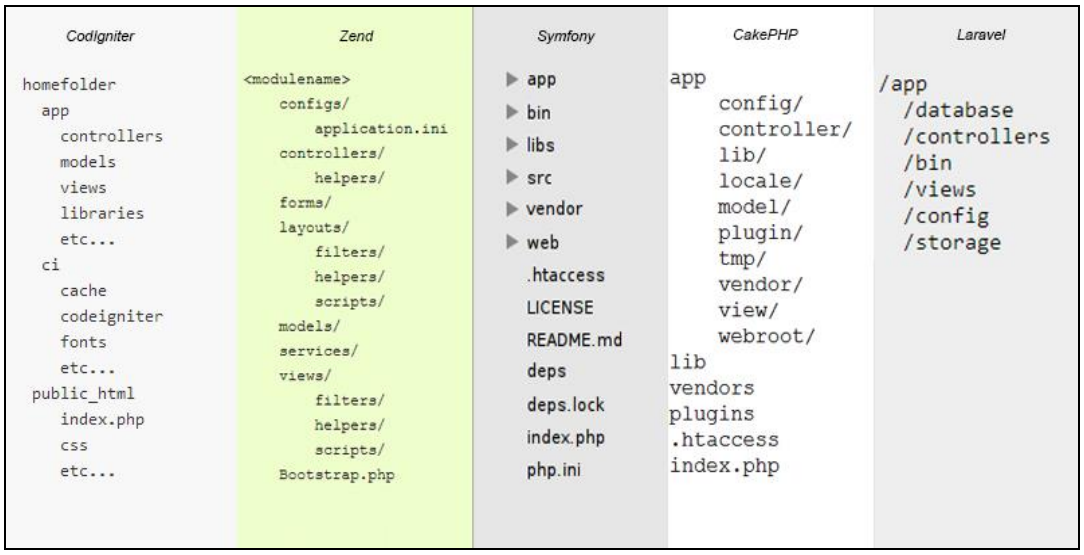

Figura 2. Estrutura de diretórios dos frameworks estudados

Os arquivos de layout são armazenados no diretório *public*, e contêm todas as estruturas fixas e comuns a todas as páginas (topo, rodapé, menus, etc), bibliotecas da *javascript* incluídas no projeto, folhas de estilo e imagens.

3.3 *Actions*, *controllers* e *view rendering*

Um ponto importante observado é a maneira como as *views* são chamadas a partir das urls. Após o *bootstraping*, o *front controller* trata as variáveis recebidas na url da requisição e extrai a ação que o usuário deseja realizar e, em seguida, invoca o método da ação solicitada, instancia todas as variáveis necessárias, trata regras de negócio, instancia as classes de *model* e prepara os dados para exibição nas *views*.

As *views* são arquivos com código HTML onde será realizada a exibição e manipulação dos dados recebidos dos *controllers* do sistema. No arquivo da *view* somente poderão ser manipuladas as variáveis e objetos que foram previamente definidas, utilizando-se de código PHP puro ou alguma linguagem de script de apoio definida pelo framework. Em alguns frameworks, para diferenciar as *views* de arquivos

HTML comuns e arquivos com código PHP puro, *views* são criadas como arquivos com extensões diferenciadas, por exemplo: phtml (Zend) e ctp (Cake).

4. Desenvolvimento da Solução Proposta

Com base nos conceitos apresentados até este ponto, o primeiro item considerado na construção do framework proposto neste trabalho (deste ponto em diante denominado "Proto FW") foi o estabelecimento de uma estrutura de classes capaz de conectar modularmente os componentes do projeto de software a ser desenvolvido, respeitando os princípios da arquitetura MVC e observando as características principais presentes nos frameworks já existentes no mercado.

4.1 Requisitos, compatibilidade e documentação

O ProtoFW foi desenvolvido para utilização com o requisito mínimo de um ambiente em sistema operacional Linux ou Windows, com servidor de aplicação Apache 2.2.17 ou superior, com a extensão *mod_rewrite* habilitada para realizar o redirecionamento das requisições para o *front controller*. Esta extensão do apache provém uma série de regras para reescrita de urls, permitindo encaminhar as requisições para outras urls ou diretórios no servidor de aplicação.

A codificação do framework foi realizada em compatibilidade com o PHP 5.3 e não requer que nenhuma extensão específica do php seja habilitada. Também foi incluída no projeto uma classe padrão para conexão com bancos de dados MySQL.

Para padronização do código, foram observadas as orientações do *PHP Framework Interoperability Group* – PHP FIG (Grupo de Interoperabilidade de Frameworks PHP), uma organização que reúne os principais desenvolvedores de frameworks em PHP e desenvolvedores da comunidade. Este grupo tem trabalhado na definição de padrões de codificação que auxiliam na padronização de código PHP compartilhado, registrado nos documentos denominados PSR (neste caso, o PSR-1 e PSR-2). O trabalho PHP FIG objetiva discutir formas de trabalho colaborativo entre os projetos dos próprios membros, mas tem sido adotado por desenvolvedores da comunidade como uma padronização de código e convenções, semelhante ao que existe em outras linguagens, a exemplo do *Java Code Conventions*.

A documentação do código segue as recomendações do PHPDoc Standard, que é um conjunto de padrões para geração de documentação em comentários de código PHP, semelhantes ao *Javadoc*. A documentação com PHPDoc é feita por meio blocos de código semelhantes a comentários de código comum, que precedem a primeira linha de um elemento estrutural, como classes, métodos, interfaces, constantes e propriedades, e devem ser iniciados com /** e terminados com */. Dentro dos blocos de documentação, podem ser incluídas *tags* padronizadas que definem características e informações do código, sempre precedidas de @.

4.2 Estrutura de diretórios do ProtoFW

A figura 3 apresenta a estrutura básica de diretórios do ProtoFW, que é similar à estrutura dos frameworks analisados, mas conta com os seguintes diferenciais principais: 1) separação bem definida das camadas do MVC; 2) simplificação da estrutura de diretórios; 3) criação de core classes simples e de fácil utilização:

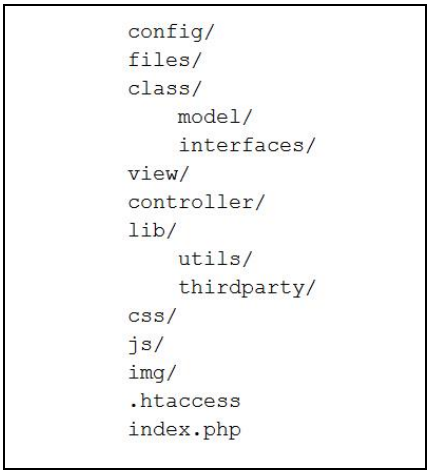

Figura 3. Estrutura de diretórios do Proto FW

O diretório *config* contém os arquivos de configuração básicos do framework, onde devem ser definidos pelo desenvolvedor os dados para acesso ao banco de dados e conexões com servidores de e-mail, e também outros dados e arquivos de configurações que podem ser criados pelo desenvolvedor.

O diretório *files* deve conter todos os arquivos que são enviados para o sistema pelo usuário (*uploads*) e outros arquivos que serão utilizados pelo desenvolvedor, como por exemplo arquivos *pdf* para download.

O diretório *class* contém as classes principais do framework, e neste ponto apresenta uma das principais diferenças em relação aos frameworks analisados: as classes criadas pelo desenvolvedor são mantidas no mesmo diretório das *core classes*, com a diferença de que estas últimas estão sempre prefixadas com a palavra "Proto". Na raiz do diretório *class,* devem ser criadas as classes das entidades do sistema, que serão classes filhas da super classe ProtoEntity. Este diretório contém dois subdiretórios *model* e *interfaces*. No diretório *model* devem ser criadas as classes de persistência, filhas da super classe ProtoDao, e devem conter o sufixo "Dao". De maneira semelhante, o diretório *interfaces* contém as interfaces do framework implementadas pelas classes ProtoEntity e ProtoDao, e nele devem ser criadas todas as novas interfaces, quando houver necessidade.

O diretório *view* contém as *views* do sistema, que são arquivos html com extensão php, onde o desenvolvedor poderá fazer pequenas inserções de código PHP, exclusivamente para implementar a saída dos dados providos pelos *controllers* do framework. Este diretório contém uma *view* padrão para o erro de requisição 404 (que pode ser personalizada pelo desenvolvedor) e uma *view* estrutural denominada *layout*, que contém a estrutura básica do *layout* aplicado ao sistema, funcionando como uma *template*, um modelo dentro do qual serão incluídas as *views* criadas pelo desenvolvedor.

O diretório *controllers* contém todas as classes controladoras criadas pelo desenvolvedor, que devem ser filhas da super classe ProtoController. Neste diretório também está classe que implementa o padrão de projeto *front controller* no framework, denominada ProtoFront.

 ISSN 2238-5096 (CDR)

O diretório *lib* contém as bibliotecas e classes que serão criadas pelo desenvolvedor, e também possui o diretório *utils*, que contém as classes utilitárias criadas para o framework. Neste diretório devem ser incluídas as bibliotecas de terceiros que serão utilizadas pelo desenvolvedor.

O diretório *css* deve conter todos os arquivos de folhas de estilo criados pelo desenvolvedor, e o diretório *img* deve conter as imagens utilizadas na composição da estruturação do *layout* das telas. Arquivos, classes, plugins e utilitários de javascript deverão ser incluídos no diretório *js*.

Na raiz do projeto existem dois arquivos que são fundamentais para o funcionamento do framework: o arquivo *.htaccess*, que contém todas as regras de redirecionamento de requisições do *mod_rewrite*, e o arquivo *index.php*, que possui o script de *bootstraping* do framework, inicializando todas as configurações necessárias para o funcionamento do *front controller* e demais classes relacionadas.

4.3 Models

A classe ProtoDao é uma super classe que implementa a interface ProtoDaoInterface. Esta interface possui a assinatura de todos os métodos básicos de manipulação de dados, como salvar, atualizar, excluir, consultar todos os registros e consultar com filtros. Para realizar as consultas no banco de dados, a ProtoDao utiliza-se da conexão definida na classe ProtoModel, um *singleton* que extende a classe utilitária de conexão com o SGDB (MySQL, nesta versão do sistema), e é responsável por controlar todo o acesso ao bando de dados.

4.4 Controllers e Views

A inexistência de um *model* específico para renderização das *views* é outra diferença significativa do ProtoFW em relação aos frameworks estudados. Quando o usuário solicita uma determinada *view* por uma url da aplicação, o próprio o *front controller* identifica qual *view* está sendo solicitada e invoca os *controllers* necessários pela manipulação dos seus dados, preparando-os para a exibição e definindo-os temporariamente como parte do escopo global da aplicação, funcionando como uma espécie de cache. Em seguida, o *front controller* realiza o redirecionamento do usuário para *view* correta solicitada na URL, onde os dados já estarão disponíveis para manipulação.

Antes da exibição de uma view comum, a view estrutural *layout* é chamada, e a view requisitada pelo usuário será incluída nela, na área definida pelo desenvolvedor, durante a aplicação do *layout* de interface criado para o sistema. Esta view possui uma chamada de inclusão nativa do php, utilizando uma variável global que contém o path para a view atual que deverá ser incluída.

Quando uma nova view é solicitada, o cache de variáveis é esvaziado e o processo é realizado novamente. Isto simplifica o processo de escrita nas views, e permite ao desenvolvedor um pouco mais de liberdade na manipulação de variáveis nesta camada. Desta forma, não é necessário criar nos controllers métodos específicos para manipulação das views, simplificando sua complexidade e permitindo a criação de métodos com melhor reusabilidade.

5. Implementação e Resultados

A fim de avaliar a utilização do ProtFW na prática, implementou-se um estudo de caso onde o framework foi utilizado no desenvolvimento do website do ENCOSIS 2014, disponível em www.encosis.com.br

O projeto foi desenvolvido utilizando PHP 5.4.20, MySql 5.5.20, e Apache 2.2.25, hospedado em um servidor Linux CenterOS com CPanel. As views foram criadas utilizando html5 e css3, e utilizaram recursos adicionais de jQuery e Ajax para animação de menus e alertas.

O processo de configuração do ProtoFW foi um diferencial de agilidade para o desenvolvimento do projeto, pois dispensa a realização de um processo de instalação e possui uma configuração extremamente simples: para iniciar o uso do framework, tudo o que o desenvolvedor necessita é copiar o código base para a raiz do projeto que será desenvolvido, e informar os dados de configuração globais no diretório config.

A definição do layout geral do website foi realizada com a inclusão dos arquivos de estilo, imagem e javascript sem suas pastas específicas no ProtoFW, e todos os elementos estruturais foram aplicados na view fixa do layout.

A figura 4 apresenta um gráfico comparativo entre o tempo de desenvolvimento das atividades nos projetos dos anos de 2012, 2013 e 2014, considerando que cada dia de atividade corresponde a quatro horas de trabalho:

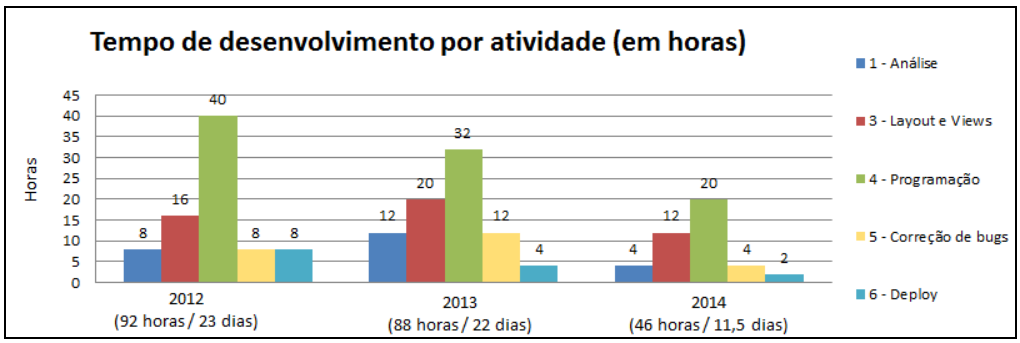

Figura 4. Comparativo do tempo de desenvolvimento por atividade

Em relação aos websites desenvolvidos para as edições de 2012 e 2013 do evento, que utilizaram outros frameworks de mercado, o tempo de desenvolvimento foi 51,1% melhor, onde o escopo do projeto desenvolvido foi similar ao dos anos anteriores, contando com o mesmo tamanho time de desenvolvimento. A cada edição de evento, um novo time é formado para desenvolver o projeto, contando com um desenvolvedor *back-end* e um desenvolvedor *front-end*, sob a supervisão de um analista presente desde o primeiro ano de atividades.

6. Conclusão

A utilização de frameworks para desenvolvimento de aplicações web em PHP é uma prática comum adotada por desenvolvedores que desejam otimizar o processo de desenvolvimento das suas aplicações, proporcionando melhor modularização, reutilização de código e separação da aplicação em camadas de interface, negócio e dados, com a adoção do padrão MVC, auxiliando na manutenibilidade do código criado.

Apesar da existência de boa documentação e de uma ampla comunidade já consolidada, como nos casos do Cake PHP e CodeIgniter, esses frameworks possuem uma biblioteca de códigos extensa demais para utilização em aplicações de pequeno porte.

Ao desenvolver o protótipo de framework apresentado neste trabalho, pensou-se em um protótipo que fosse capaz de oferecer os recursos necessários para bom usufruto dos benefícios do padrão MVC, com um processo de instalação e configuração inicial mais simples, dispensando aprendizado de scripts complexos e, sobretudo, tornando mais transparente para o desenvolvedor o funcionamento das rotinas próprias do framework. Desta maneira, o website desenvolvido como modelo de aplicação do protótipo criado apresentou significativas melhorias no tempo de desenvolvimento, um ganho de 51,1% em relação a outras soluções usadas em projetos com escopo semelhante, executado por times de desenvolvimento com o mesmo tamanho.

Como ponto de melhoria em trabalhos futuros, planeja-se expandir o modelo para incluir classes mais robustas de manipulação de bancos de dados, que possam, por exemplo, controlar a criação e manutenção de objetos em bancos de dados relacionais, facilitando de administração de bases de dados para aplicações de pequeno porte.

7. Referências

Aihara, Diogo Satoru (2009) “Study About the Relationship Between the Model-View-Controller Pattern and Usability”. Disponível em: < http://bit.ly/1lxZHpv>. Acesso em: 23 de Janeiro de 2013. p. 14.

Gamma, E., Helm, R., Johnson, R. & Vlissides, J. (1995). Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, Boston. p. 144-145.

Lancor, L. e Katha, S. (2013) “Analyzing PHP frameworks for use in a project-based software engineering course”, In: SIGCSE '13 Proceeding of the 44th ACM technical symposium on Computer science education, ACM, New York, USA, p. 519-524.

McArthur, Kevin (2008) Architeture”, Pro PHP Patterns. In *Frameworks, Testing and More*. Berkeley, CA, United Status, APress, p. 201-2013.

Minetto, Elton L. (2007) “Frameworks para desenvolvimento PHPSão Paulo, Editora Novatec, p. 14-19.

Reenskaug, Trygve (1979) “The Model-View-Controller (MVC) Its Past and Present”. Disponível em: <http://heim.ifi.uio.no/~trygver/themes/mvc/mvc-index.html>. Acesso em: 28 de Fevereiro de 2014.

Romakenko, Hellen (2013). “Top 5 PHP Frameworks Infographic” Disponível em: <http://php.dzone.com/articles/top-5-php-frameworks>. Acesso em: 08 de Março de 2014

Wang, Guanhua (2011) “Application of lightweight MVC-like structure in PHP”. Business Management and Electronic Information (BMEI), 2011 International Conference on (Volume:2) Guangzhou, China, p. 73-74.