ROTERMEL, Felipe. SOMMARIVA, Leonardo W**. Inovações advindas na nova versão da linguagem de programação web PHP 7.0.** Revista Interdisciplinar Científica Aplicada, Blumenau, v.10, n.4, p.1-20, TRIV 2016. ISSN 1980-7031

Inovações advindas na nova versão da linguagem de programação web PHP 7.0

Felipe Rotermel[1], Leonardo W. Sommariva[2]

[1]Pós-Graduando em Tecnologias para o Desenvolvimento de Aplicações Web
Universidade Regional de Blumenau (FURB) – Blumenau, SC – Brazil

[2]Departamento de Sistemas e Computação
Universidade Regional de Blumenau (FURB) – Blumenau, SC – Brazil
frotermel@gmail.com, lsommariva@gmail.com

Resumo. Este artigo apresenta a linguagem de programação PHP e as principais inovações que a sua versão 7.0 recebeu em relação a antecessora 5.6. O objetivo é averiguar se o PHP, que é uma das linguagens de programação web mais utilizadas no mundo, evoluiu para continuar atrativa e moderna para atender a evolução da internet e a sua comunidade de desenvolvedores. Utilizou-se de pesquisa bibliográfica em livros, manuais e sites especializados, bem como testes realizados entre as versões citadas comparando vários cenários diferentes. O artigo demonstra que a nova versão do PHP evoluiu com novos recursos e melhoria no desempenho.

Palavras-Chave: Inovação, Linguagem de Programação.

Abstract. This paper introduces the PHP programming language and the main innovations received in it 7.0 version relating it´s predecessor 5.6 version. The objective is to check if the PHP, which is one of the most used web programming languages in the world, has evolved to keep attractive and modern to meet the evolution of the Internet and its development community. It was used bibliographic research in books, manuals and specialized websites as well as tests between versions mentioned comparing several different scenarios. The paper shows that the new PHP version has evolved with new features and improved performance.

Key-Words: Innovations, Programming Language.

1. Introdução

Soares (2000, p. 1), A internet que se conhece hoje é bem diferente da internet de páginas estáticas, que mostravam sempre a mesma informação para todos os internautas nos primórdios da Web. Ela se tornou dinâmica e hoje além de visitar

páginas, pode-se fazer um número incontável de atividades tais como, pesquisar, comprar, vender, pagar contas, compartilhar, publicar, reclamar, etc.

Para que isso fosse possível, a Web estática baseada em páginas HTML, teve que se tornar dinâmica, flexível, e é aí que entram as linguagens de programação Web. Elas são ferramenta para que programadores do mundo todo possam transformar páginas estáticas em fontes de informação, entretenimento e negócios (SOARES, 2000).

PHP (PHP: *Hypertext Preprocessor*) é uma dessas linguagens de programação Web que possibilitou essa evolução da internet, ela é uma das linguagens mais utilizadas mundialmente. Segundo Converse e Park (2003, p. 3) PHP é uma linguagem de criação de scripts do lado servidor, que é incorporada em um arquivo HTML. Isso possibilita que a página interaja com bancos de dados e aplicações existentes no servidor, sem expor o código fonte para o internauta, como resultado temos páginas dinâmicas.

Dentre outras linguagens o sucesso do PHP provém basicamente, de acordo com Converse e Park (2003), pelo fato de ser código-fonte aberto, por dispor de inúmeros recursos, servir a diversas plataformas, pela sua estabilidade, rapidez, compatibilidade com outros produtos, pela grande comunidade que dá suporte e por ser de fácil aprendizado.

Nas próximas seções são apresentados um breve histórico da linguagem PHP, suas principais inovações e também testes de desempenho da nova versão. O artigo foi produzido através de uma pesquisa bibliográfica com vários autores e também de informações fornecidas pelo próprio fabricante. O propósito deste artigo é verificar quais foram as inovações apresentadas nesta nova versão, e se com isso o PHP ainda continua sendo uma boa opção quando se trata de programação web dinâmica.

2. Histórico da linguagem PHP

A linguagem PHP (Personal Home Page) nasceu em 1994 através de um conjunto de scripts voltados a criação de páginas dinâmicas criadas por Rasmus Lerdorf para monitorar acessos a seu currículo na internet. Rasmus foi acrescentando funcionalidades a sua ferramenta afim de facilitar o desenvolvimento de aplicações Web

por outras pessoas, até que em 1995 ele decide disponibilizar seu código na internet para receber ajuda e correção de problemas. Esta é a versão 1 do PHP conhecida como PHP/FI (Form Interpreter) (DALL´OGLIO, 2014, p. 20).

Xavier (2008, p. 4), estima-se que em meados de 1997 mais de 50.000 sites estavam utilizando o PHP, nessa mesma época o PHP, que já estava na segunda versão, deixou de ser projeto de Rasmus para ter uma equipe profissional de desenvolvimento. Zeev Suraski e Andi Gutmans reescreveram seu interpretador, que foi a base para a 3ª versão do PHP, onde também sua sigla foi rebatizada para PHP: Hypertext Preprocessor.

De acordo com Dall´Oglio (2014, p. 21) o PHP 3 possibilitou a conexão com vários bancos de dados diferentes, proporcionou uma sintaxe de código mais consistente, deu início ao suporte a orientação a objeto, entra outras melhorias, o que acabou atraindo vários desenvolvedores.

A versão 4 do PHP foi lançada no ano 2000, com o novo núcleo chamado de Zend Engine com melhorias de performance e modularidade em aplicações complexas. O suporte a vários outros servidores Web e a abstração de sua API foram suas principais inovações (DALL´OGLIO, 2014, p. 21).

Segundo Dall'Oglio (2014), o recurso de orientação a objetos presentes nas linguagens C++ e Java, foi disponibilizado plenamente a partir de 2004 no lançamento da versão 5 do PHP.

O novo Zend Engine II garantiu melhorias de performance e na versão 5.1 de 2005 foi adicionada uma nova interface de acesso aos bancos de dados PDO (PHP Data Objects). Em 2006 a versão 5.2 trouxe suporte nativo ao JSON (JavaScript Object Notation) um formato leve para troca de dados. A versão 5.3 adicionou um coletor de lixo de referências circulares, suporte a *namespaces*, *late static bindings* e a função *goto*, entre outros avanços. Melhorias em funcionalidades já existentes e também na performance foram liberadas na versão 5.4 de 2012, criada uma versão mais curta na sintaxe de vetores e suporte à *Trait*. As versões 5.5 e 5.6 são basicamente atualizações de segurança e de solução de problemas (PHPNET, 2016).

A versão 6 nunca chegou a ser lançada, porém muitas de suas funcionalidades previstas foram incluídas em versões intermediárias, atualmente a versão 5.6.20, porém o suporte nativo a Unicode que estaria presente na versão 6 ainda não foi desenvolvido. Para evitar conflitos e não deixar dúvidas, optou-se por pular essa versão.

Atualmente, março de 2016, o PHP se encontra na versão 7.0.5. De acordo com PHPNET (2016) esta versão se concentra principalmente na remoção de funcionalidades depreciadas de versões anteriores e na melhoria da consistência da linguagem.

3. PHP 7

A nova versão 7 do PHP estabelece um novo período para a linguagem, as versões 5, que serviram os usuários até 2015, se tornam obsoletas e passam a fazer parte da história de mais de uma década no PHP.

Um dos objetivos desta nova versão é suprir um espeço que vinha sendo preenchido por iniciativas externar como Hack e HPHPc, linguagens que tem como objetivo tornar o PHP 5.X mais rápido e adicionando mais recursos (RIBEIRO, 2016).

A pressão de grandes empresas que utilizam PHP como o Facebook e Wikipédia por mais performance acabou sendo atendida pelo novo motor, o Zend Engine 3 teve seu código otimizado e passou a ter um desempenho superior as iniciativas citadas anteriormente.

Lerdorf (2016), afirma que o PHP possui um ganho de performance de mais de 100% na maioria das aplicações de “mundo real”, além de usar menos memória para alcançar esse resultado. Teste realizados pela Zend Technology (2016), confirmam esta afirmação de Lerdorf. Existe um expressivo ganho de desempenho nesta nova versão, bem como novos recursos e funções que visam a modernização da linguagem.

4. O que há de novo no PHP7

Muitas inovações foram disponibilizadas na versão 7 do PHP que serão apresentadas brevemente com alguns exemplos de implementação.

4.1 Declaração de tipos escalares

Um grande diferencial, mas também alvo de críticas do PHP é que relação a sua declaração de tipos de variáveis, PHP é uma linguagem não tipificada, de acordo com Converse e Park (2003, p. 77) o PHP assume ou converte automaticamente de um tipo para outro dependendo do contexto ou do resultado, mas existe a possibilidade de forçar a conversão quando o programador achar necessário.

Declaração de tipos escalares, disponível a partir da versão 7, possibilita duas opções: coercivo (padrão) e restrito. Para parâmetros, os seguintes tipos podem ser forçados (tanto coercivamente quanto rigorosamente): *strings* (*string*), inteiros (*int*), números ponto-flutuante (*float*), e booleanos (*bool*). Eles incrementam os tipos introduzidos no PHP 5: nomes de classe, *interfaces*, *array* e *callable* (PHPNET, 2016).

Para habilitar o modo rigoroso, uma simples diretiva *declare* deve ser colocada no topo do arquivo. Isso significa que a rigorosidade de tipificação é configurada por arquivo. Esta diretiva não afeta somente as declarações de tipo de parâmetros, mas também do tipo de retorno de funções, funções internas do PHP e funções de extensões carregadas (PHPNET, 2016).

4.2 Declaração de tipos de retorno

Seguindo a mesma ideia o PHP 7 adiciona o suporte a declarações de tipo de retorno de uma função ou método. Esta é uma função totalmente nova, nenhum tipo de retorno estava disponível nas versões anteriores. Todos os tipos disponíveis para declarações de tipo de argumentos (*int*, *float*, *string*, *bool*, *array* e objeto), também estão disponíveis para a declarações para tipo de retorno (RIBEIRO, 2016).

4.3 Operador "nave espacial" (*spaceship*)

O operador nave espacial `<=>` é utilizado para comparação entre duas expressões. Esse operador funciona de forma semelhante aos `<`, `<=`, `==`, `>` e `>=`, mas caso a comparação seja idêntica, retornará `0`, se o valor da esquerda for o maior, retornará `1`, e se o valor da direita que for o maior, retornará `-1`. As comparações são

feitas de acordo com a já conhecida regras de comparação de tipos do PHP (PHPNET, 2016).

Figura 1. Exemplo do uso do operador *spaceship* (PHPNET, 2016)

```php
<?php
// Integers
echo 1 <=> 1; // 0
echo 1 <=> 2; // -1
echo 2 <=> 1; // 1

// Floats
echo 1.5 <=> 1.5; // 0
echo 1.5 <=> 2.5; // -1
echo 2.5 <=> 1.5; // 1

// Strings
echo "a" <=> "a"; // 0
echo "a" <=> "b"; // -1
echo "b" <=> "a"; // 1
?>
```

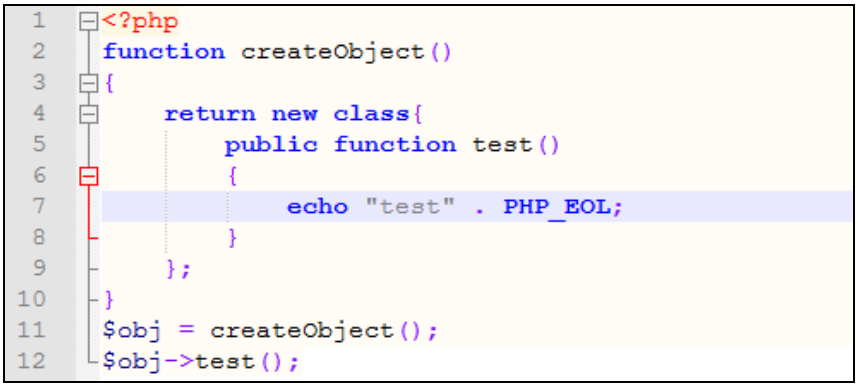
4.4 Classes anônimas

O PHP, a partir da versão 5.4, permite a criação de Funções Anônimas. Elas são úteis especialmente para criação de *callbacks* ou para usar em parâmetros de funções, como *array_map()*(PHPNET, 2016).

A partir do PHP 7, podemos criar Classes Anônimas utilizando *new class*. Isso pode ser utilizado no lugar de definições completas de classes para objetos descartáveis, cria-se uma classe anônima quando não há intenção de se instanciar objetos dessa classe em outros pontos do código. A classe deve ser criada no mesmo ponto do código em que se espera receber um objeto dela (RIBEIRO, 2016).

É possível, por exemplo, fazer uma função retornar uma classe, definida na própria expressão *return* conforme exemplo na figura a seguir.

Figura 4. Exemplo do uso de Classes anônimas

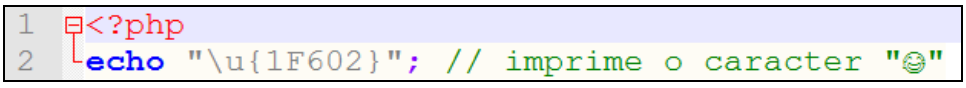


É preciso cautela ao usar classes anônimas, o código pode perder legibilidade dependendo do tamanho da classe e também pode-se causar problemas de arquitetura caso seja usada em locais impróprios (RIBEIRO, 2016).

4.5 Sintaxe de escape de códigos Unicode

Com o PHP 7 é possível utilizar quaisquer caracteres Unicode em uma *string*, ao utilizar qualquer código válido o PHP transforma esse código Unicode em sua forma hexadecimal e os imprime em uma *string* UTF-8 circundada por aspas (PHPNET, 2016).

Figura 5. Exemplo do uso códigos Unicode



Uma nova classe chamada *IntlChar* disponível no PHP 7, ela busca explorar funcionalidades adicionais da ICU. A classe define alguns métodos estáticos e constantes que podem ser utilizadas para manipular caracteres Unicode. Para utilização desta classe, a extensão *Intl* precisa ser instalada.

4.6 Método *Closure::call()*

Este novo método *Closure*::*call()* é uma forma mais fácil de vincular temporariamente um objeto a uma *closure* e invocá-la, seja o exemplo a seguir (PHPNET, 2016).

Figura 6. Exemplo de Closure::call() (PHPNET, 2016)

```php
<?php
class A {private $x = 1;}

// Pre PHP 7 code
$getXCB = function() {return $this->x;};
$getX = $getXCB->bindTo(new A, 'A'); // intermediate closure
echo $getX();

// PHP 7+ code
$getX = function() {return $this->x;};
echo $getX->call(new A);
```

4.7 Método unserialize() filtrado

Este método tem a finalidade de melhorar a segurança na “deserialização” de objetos com informações não confiáveis. Ele previne possíveis injeções de código malicioso e permite que o desenvolvedor escolha as classes que devem ser “deserializadas” (PHPNET, 2016).

4.8 Agrupamento de declarações *use*

Utilizando *use*, é possível importar classes, funções e constantes de um mesmo namespace, agrupando-as em uma única declaração, como no exemplo a seguir.

Figura 7. Exemplo de use agrupando Classes

```php
<?php
// Antes do PHP 7
use teste\namespace\Classe1;
use teste\namespace\Classe2;
// Com PHP 7
use teste\namespace\{Classe1, Classe2};
```

4.9 Retornar expressões em geradores

Este novo recurso é uma funcionalidade extra dos geradores que já estavam presentes no PHP 5.5. Isso possibilita a utilizar-se da declaração *return* dentro de um gerador, para que retorne uma expressão final. Para recuperar o valor, pode-se usar o novo método *Generator::getReturn()* (PHPNET, 2016).

4.10 Divisão inteira com *intdiv()*

No desenvolvimento de aplicações, quando utiliza-se divisão, normalmente precisamos do resultado inteiro e o resto separadamente. Nas versões anteriores, normalmente fazia-se uma divisão e depois arredondava-se o valor para baixo. Esta nova função do PHP 7 retorna o resultado da divisão inteira de seus operandos conforme exemplo da figura 8 a seguir (RIBEIRO, 2016).

Figura 8. Exemplo da função *intdiv()*

```php
<?php
var_dump(intdiv(5, 2));
// retorno desta função é int(2)
```

4.11 Função preg_replace_callback_array()

Nas versões anteriores, *callbacks* que precisam ser executadas por expressões regulares eram poluídas com diversas ramificações. A partir da versão 7, a nova função *preg_replace_callback_array()* permite que o código seja escrito de forma mais limpa utilizando a função *preg_replace_callback(). Callbacks* agora podem ser assimiladas a expressões regulares utilizando um *array* associativo, sendo a chave é uma expressão regular, e o valor uma função *callback*.

4.12 Funções CSPRNG

São duas novas funções que geram de maneira criptograficamente segura *strings* e inteiros de maneira randômica, *random_bytes()* e *random_int()* respectivamente (PHPNET, 2016). São funções úteis para se montar *sal* para senhas criptografadas ou em aplicações em que números devem ser gerados aleatória sem serem influenciados pelo tempo (RIBEIRO,2016).

Figura 9. Exemplo da função random_int()

```php
<?php
var_dump(random_int(1,100));
var_dump(random_int(-1000, 0));
//resultados possíveis int(55), int(-349)
```

5. Mudanças ao codificar na nova versão

Para ter proveito das inovações e também evitar possíveis incompatibilidades, é preciso estar atendo as mudanças que ocorreram em algumas ferramentas e funções. Nesta seção são apresentadas as principais mudanças que irão interferir no antigo modo de codificação do PHP 5.X.

5.1 Operador de coalescência nula (*null coalescing*)

Muitas vezes existe a necessidade de testar se um parâmetro existe, retornando seu próprio valor caso positivo ou um valor alternativo caso contrário. No PHP 5.6, teríamos que testar se o parâmetro existe através da função *isset* antes de retornar seu valor, caso contrário recebe-se um erro E_NOTICE. No PHP 7 existe o operador ??, que testa a existência do parâmetro e mostra um valor alternativo caso o mesmo não esteja definido, semelhante a função “se” de planilhas eletrônicas (RIBEIRTO, 2016).

Figura 2. Exemplo do uso do Operador de coalescência nula ??

```php
<?php
$lista = ['chave' => 'valor'];
// PHP 5.6
$x = $lista['nao_encontrada'] ?: 123; // E_NOTICE se a chave não existir
$x = isset($lista['nao_encontrada']) ? $lista['nao_encontrada'] : 123; // Ok
// PHP 7
$x = $lista['nao_encontrada'] ?? 123; // Ok
```

5.2 Arrays constantes utilizando a função define()

Array constantes agora podem ser definidos com a função *define*(). No PHP 5.6, só poderiam ser definidos com a função *const* (PHPNET, 2016).

Figura 3. Exemplo do uso do *define*() na criação de Arrays

```php
<?php
define('CARROS', [
    'sentra',
    'civic',
    'corolla'
]);
```

5.3 Expectations

Expectations é uma inovação da antiga função *assert().* A *Expection* permite asserções com menos código de produção, e possibilita lançar exceções personalizadas quando a asserção falha. O uso da *assert()* do modo antigo ainda continua mantido no PHP 7 por motivos de compatibilidade, porém, *assert()* agora é um construtor de linguagem, permitindo que o primeiro parâmetro seja uma expressão, em vez de somente uma *string* que era avaliada ou um valor booleano a ser testado (PHPNET, 2016).

5.4 Delegação de geradores

A partir do PHP 7, geradores podem ser delegados a outros geradores, objetos *Traversable* ou a um *array* automaticamente, não é necessário escrever um gerador externo padrão utilizando o construtor *yield from* (PHPNET, 2016).

5.5 Opções de sessões

O PHP 7 permite um *array* de opções na função *session_start(),* que substituem as diretivas de configuração de sessões normalmente configuradas no arquivo php.ini.

Esta modificação foi feita para dar suporte *session.lazy_write*, que está habilitada por padrão e faz com que o PHP somente sobrescreva um arquivo de sessão se a informação da sessão foi modificada, e *read_and_close*, que é uma opção que pode ser passada para a função *session_start()* indicando que a informação da sessão deve ser lida e então imediatamente fechada sem ser modificada (PHPNET, 2016).

5.6 Função list()

A função list(), a partir do PHP 7, consegue desempacotar objetos que implementam ArrayAccess. Antes, a função list() não garantia confiabilidade em operações com esse tipo de objetos (PHPNET, 2016).

5.7 Alteração no foreach

O foreach recebeu algumas mudanças nesta nova versão do PHP, a principal delas visa acompanhar o comportamento de outras linguagens. É o caso dos ponteiros de um array, que antes eram modificados quando o array era iterado, isto não ocorre mais, pois no PHP 7 quando o array é iterado, isto é feito sobre uma cópia do mesmo, conservando assim os seus ponteiros originais (PHPNET, 2016).

5.8 Manipulação com a função list()

A função list(), como seu nome sugere, é usada para criar uma lista de variáveis em para ser usada em alguma operação. Uma das opções para criar essa lista era seria usando um array, nesse caso os valores informados eram atribuídos em ordem reversa a informada. Isto não acontecera mais a partir da versão 7, os valores informador pelo list a um array serão atribuídos na mesma ordem (PHPNET, 2016).

5.9 Modificações na manipulação de erros e exceções

Esta é uma mudança muito aguardada por desenvolvedores, a conversão de erros fatais para exceções, agora quando ocorrer um acesso inválido ou um comando não

definido, por exemplo, ao invés do PHP parar de executar e exibir um erro genérico na tela, é possível tratar esses potenciais erros como exceções, o que antes não era possível nas versões anteriores (TABLELESS, 2016).

6. O que foi descontinuado

Em qualquer linguagem de programação, quando é lançada uma nova versão, é natural que existam incompatibilidades e depreciação de recursos, entre o as versões anteriores do PHP e o PHP 7 não seria diferente. A lista completa e detalhada de todas as incompatibilidades e depreciações, pode ser acessada no site de PHP em documentação, migrando do PHP 5.6.x para o PHP 7.0.x (PHPNET, 2016).

6.1 Construtor estilo PHP 4

O método construtor é um método de uma classe o qual sempre é executando quando um objeto desta classe é instanciado utilizando o operador new. Nas versões PHP 4 o construtor era um método que tivesse o mesmo nome da classe. A partir da versão 5 do pode-se definir o método construtor da forma antiga ou usar o método a "*__construct*()". A partir do PHP 7 caso sejam criados métodos com o mesmo nome das classes será exibida uma mensagem de erro, informando que este tipo de métodos está depreciado, sendo assim, os construtores no PHP 7 devem ser criados somente pelo método "*__construct*()" (TABLELESS, 2016).

6.2 Chamadas estáticas a métodos não estáticos

Ao se declarar um método como *static* ele se torna acessível sem a necessidade de uma instancia de sua classe. Isto é um padrão conceitual de linguagens orientadas a objetos, porém o PHP permitia a chamada estática a métodos não estáticos, isto não está mais disponível no PHP 7 (PHPNET, 2016).

6.3 Opção salt da função password_hash()

O salt é uma opção da função password_hash(), ela permitia aos desenvolvedores prover manualmente os salts, o que gera uma possível falha na segurança do sistema. Caso esta opção fosse omitida, o salt seria gerado aleatoriamente

e criptografado pela função password_hash(), o que deve acontecer automáticamante a partir da versão 7 pois está opção não está mais disponível (PHPNET, 2016).

6.4 Opção de contexto SSL na função capture_session_meta

A opção ssl da função *capture_session_meta,* que era a responsável pelo uso dos metadados de criptografia do protocolo SSL, foi depreciada na versão 7 do PHP. Esta mesma opção agora faz parte de outra função, a *stream_get_meta_data()* que além da opção ssl, controla outras opções de cabeçalho (PHPNET, 2016).

6.6 SAPIs removidas

Algumas Servers APIs foram removidas no PHP 7, é se suma importância que os desenvolvedores tenham conhecimento, pois isto pode acarretar de incompatibilidades na hora de migrar suas aplicações para a nova versão. A seguir uma lista das SAPIs removidas.

Tabela 1. SAPIs removidas a partir da versão 7 do PHP (PHPNET, 2016)

aolserver	continuity	pi3web
apache	isapi	roxen
apache_hooks	milter	thttpd
apache2filter	nsapi	tux
caudium	phttpd	webjame

6.7 Acesso a Banco de Dados

Funções específicas de acesso aos banco de dados foram removidas, comandos antigos que iniciavam geralmente pelo nome do banco, *mysql_connect()* por exemplo, deixam de ser suportadas nesta nova versão do PHP. Para acessar o banco de dados o desenvolvedor deve usar as funções da classe PDO (PHP Data Object) (TABLELESS, 2016).

7. Desempenho

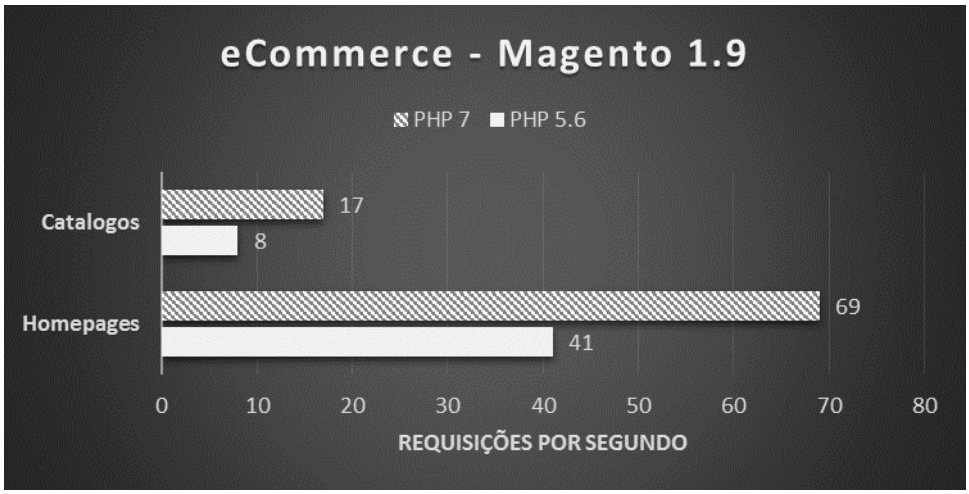
Além de todas as novidades vistas anteriormente, sem dúvida o desempenho da no Engine do PHP 7 é uma das mais importantes. Vários websites especializados, blogs

e fóruns da comunidade PHP comentam que esta nova versão é muito rápida. Porém o que é rápido para um pode não ser para outro, portanto se faz necessário quantificar esse ganho de velocidade.

A Zend Technologies disponibilizou um benchmark de performance comparando os aplicativos PHP populares na internet, utilizando como métrica o número de requisições http atendidas por segundo.

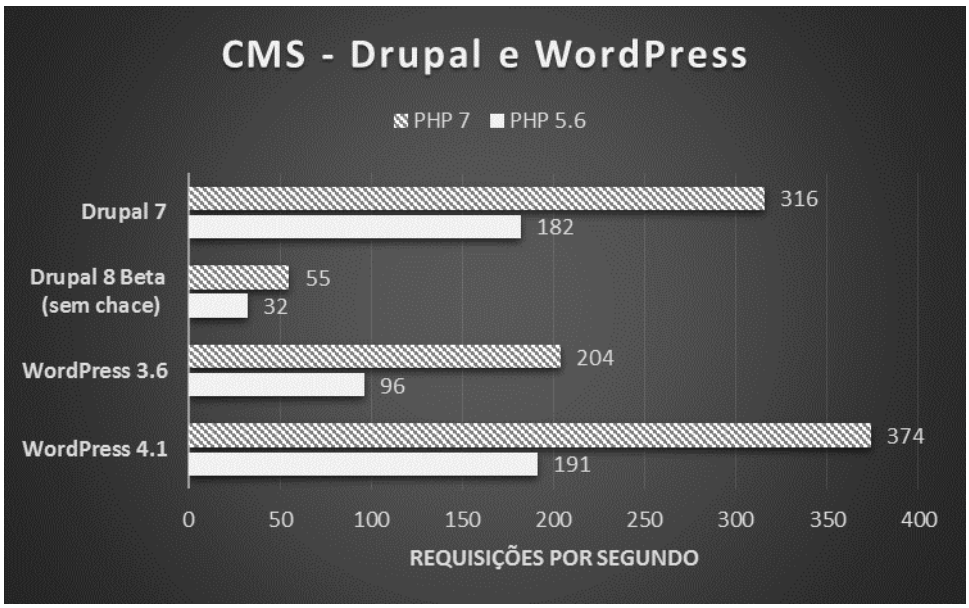
Utilizando a aplicativo de comercio eletrônico Magento 1.9, a nova versão do PHP conseguiu atender mais do que o dobro de requisições em catálogos e quase 70% mais requisições de *homepages* (ZEND, 2016).

Figura 10. Gráfico de requisições respondidas por segundo no aplicativo Magneto 1.9 (ZEND, 2016)



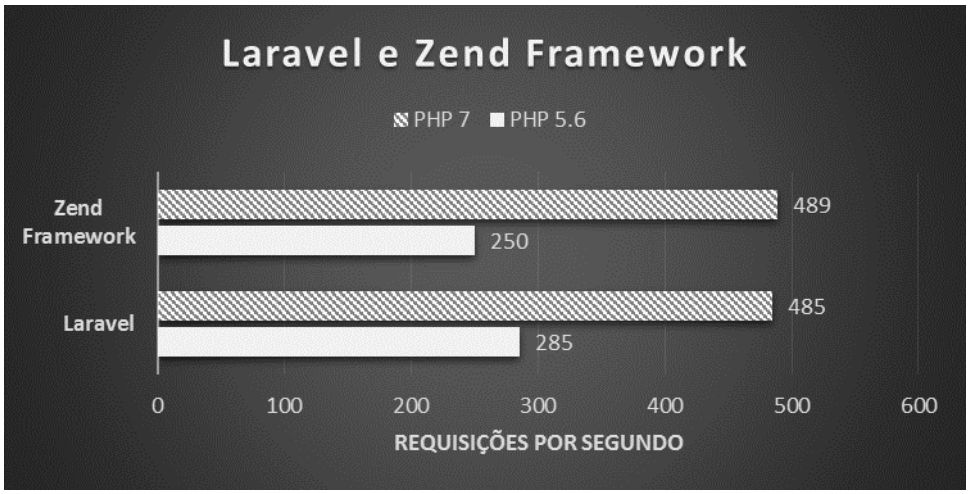
Com os aplicativos Drupal e WordPress, que são sistemas de gerenciamento de conteúdo, o desempenho também se mostrou muito superior. Isso reflete não só em velocidade, mas também em economia de hardware, ou seja, como o código do PHP 7 é mais eficiente, o consumo de CPU e memória são reduzidos e consequentemente os custos (ZEND, 2016).

Figura 11. Gráfico de requisições respondidas por segundo nos aplicativo Drupal e WordPress (ZEND, 2016)



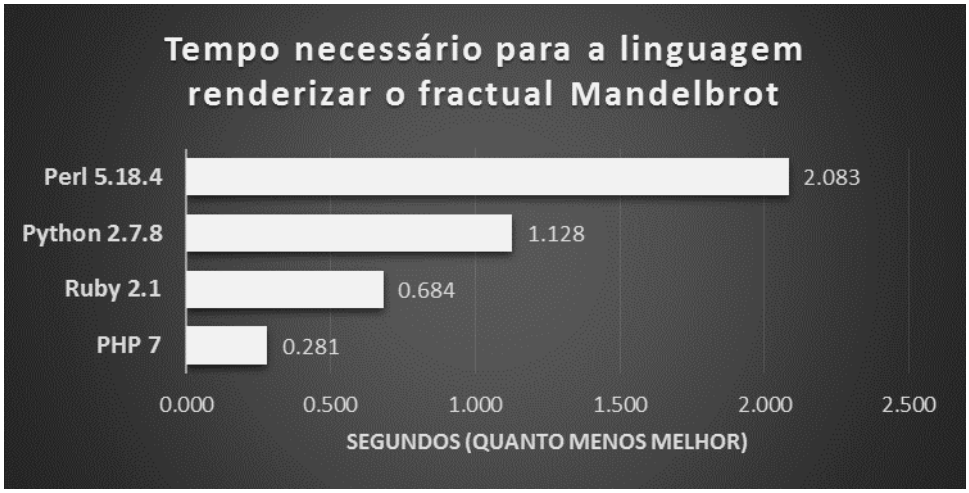
Outra ferramenta muito utilizada por desenvolvedores sãos os *Frameworks*, a Zend Technologies também testou o seu Zend framework2 e o Laravel, onde mais uma vez o PHP 7 mostrou grande rapidez confirme gráfico da figura 12 (ZEND, 2016).

Figura 12. Gráfico de requisições respondidas por segundo nos aplicativo Laravel e Zend Framework (ZEND, 2016)



A Zend Technologies também realizou testes comparando o PHP 7 com outras linguagens de programação web, o teste consistiu na renderização do fractal Mandelbrot pelas linguagens, PHP7, Ruby 2.1, Phyton 2.7.8, Perl 5.18.4. Na figura a seguir é possível verificar o resultado e mais uma vez o PHP se mostrou mais veloz rederizando o fractal em menor tempo (ZEND, 2016).

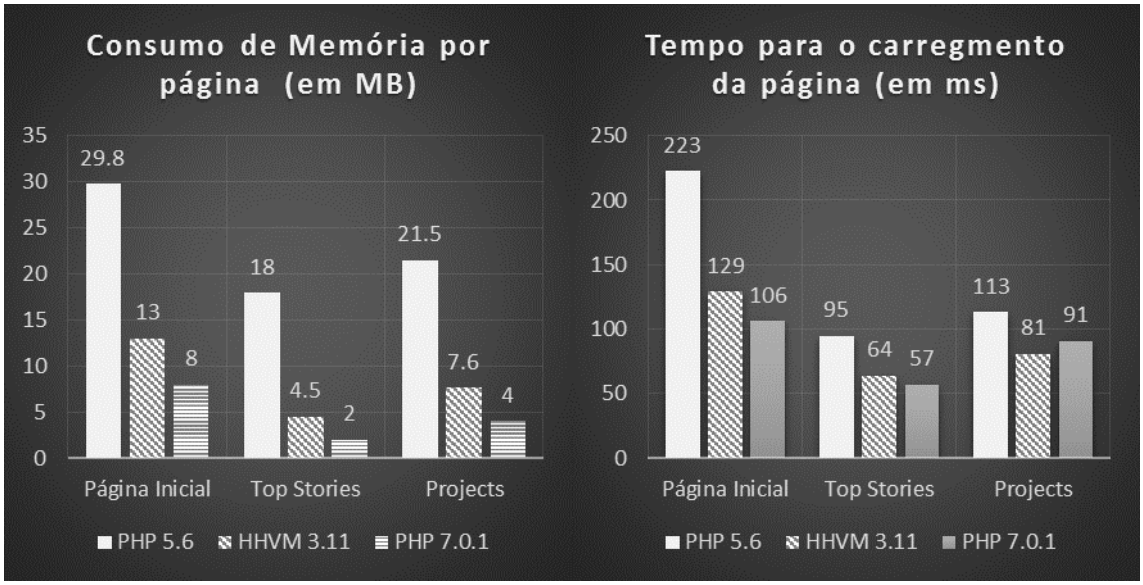
Figura 13. Tempo necessário para renderizar o fractal Mandelbrot (ZEND, 2016)



Um grupo de usuários da Finlândia, que são responsáveis pelo desenvolvimento e manutenção do Framework Symfony, também realizaram comparativos entre o PHP 5, PHP 7 e também HHVM. Para este teste foi utilizada a instalação padrão da plataforma Symfony com as páginas de demonstração, um proxy reverso padrão da Symfony e uma API REST (SYMFONY, 2016).

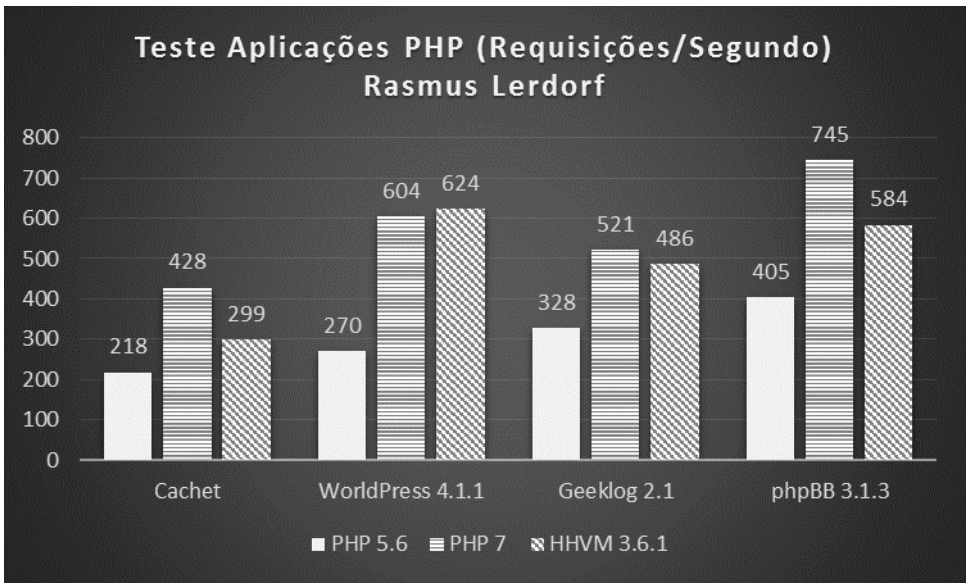
Conforme observa-se na figura 14, o HHVM também se mostra com bom desempenho, mas mesmo assim, o PHP 7 esteve na frente em praticamente todos os testes, o que não deixa dúvida sobre sua ótima performance.

Figura 14. Tempo e memória necessários para carregamento da página (SYMFONY, 2016)



Rasmus Lerdorf em abril de 2015, em uma de suas primeiras apresentações da nova versão, na época ainda em teste, mostrou também alguns números de requisições por segundo em várias aplicações de mercado. Alguns desses números são apontados na figura a seguir.

Figura 15. Requisições atendidas por segundo (LERDORF, 2016)



Diante dos números apresentados em todos os comparativos, fica claro que o PHP 7 e seu novo motor Zend Engine 3 trouxeram um grande ganho de desempenho em relação ao seu antecessor PHP 5.6, o que por si só já torna a migração para nova versão muito vantajosa.

8. Conclusão

A web é essencial para a sociedade em que vivemos, tanto para negócios ou para o lazer, sua presença será cada vez maior no cotidiano do mundo todo e consequentemente sua evolução é constante. Milhares de ferramentas já foram criadas, e muitas delas construídas com a linguagem PHP, que precisa evoluir para continuar sendo uma linguagem moderna e atrativa.

O PHP 7 apresentou muitas melhorias e novos recursos que eram aguardados pela sua comunidade de desenvolvedores, isso é muito importante para sua manutenção no mercado e para manter a atrair desenvolvedores.

Segundo W3Techs (2016), 82,3% de todos os sites conhecidos que usam linguagem de programação do lado do servido, estão utilizando PHP. Atingir um número tão expressivo de fatia de mercado não é possível com amadorismo. O PHP hoje é uma linguagem de programação web madura, eficiente e de grande adaptação,

adaptação que talvez tenha sido seu diferencia, e que talvez seja essa a chave do seu grande sucesso.

O artigo cumpre seu papel ao poder afirmar e a nova versão do PHP atende os requisitos necessários para continuar seu caminho junto a web, entregando importantes novas funcionalidades, retirando o que ficou obsoleto e melhorando expressivamente o seu desempenho. Não se pode dizer que o PHP chegou a perfeição, mas é um degrau importante na constante evolução das tecnologias da informação.

Como trabalhos futuros sugere-se a comparação do PHP 7 com outras linguagens do lado do servidor, como Java e ASP.net, apontado pontos fortes e fracos de cada linguagem.

9. Referências

CONVERSE, Tim; PARK, Joyce. **PHP**: a Bíblia. Rio De Janeiro: Campus, 2003. xxxi, 868p, il.

DALL´OGLIO, Pablo. **PHP**: programando com orientação a objetos.2.ed. São Paulo: Novatec, 2014. 574 p, il.

LERDORF, Rasmus. **Speeding up the web with PHP 7**. Disponível em < http://talks.php.net/fluent15#/>. Acesso em 25 abr 2016.

W3TECHS: Web Technology Surveys. Disponível em < http://w3techs.com/technologies/details/pl-php/all/all>. Acesso em 25 abr 2016.

PHPNET: Novos recursos. Disponível em <http://php.net>. Acesso em 25 abr 2016.

RIBEIRO, Rubens T. **Rubsphp**: Novidades do PHP 7. Disponível em < http://rubsphp.blogspot.com.br/2016/02/novidades-do-php-7.html> Acesso em 25 abr 2016.

SOARES, Walace. **Programando em PHP**: conceitos e aplicacoes.2. ed. Sao Paulo: Erica, 2000. 386p, il.

ROTERMEL, Felipe. SOMMARIVA, Leonardo W**. Inovações advindas na nova versão da linguagem de programação web PHP 7.0.** Revista Interdisciplinar Científica Aplicada, Blumenau, v.10, n.4, p.1-20, TRIV 2016. ISSN 1980-7031

SYMFONY: Benchmarks PHP 5.6, HHVM 3.11 and PHP 7.0.1 Disponível em <https://www.symfony.fi/>. Acesso em 25 abr 2016.

TABLELESS: 10 novidades do PHP 7. Disponível em < http://tableless.com.br/10-novidades-do-php-7/>. Acesso em 25 abr 2016

XAVIER, Fabrício S. V. **PHP**: do básico à orientação a objetos. Rio de Janeiro: Ciência Moderna, 2008. x, 234 p, il.

ZEND TECHNOLOGIES: Get performance insight into the upcoming release of PHP 7. Disponível em <http://www.zend.com/en/resources/php7_infographic>. Acesso em 08 abr 2016.