UNIVERSIDADE DO EXTREMO SUL CATARINENSE - UNESC

CURSO DE CIÊNCIA DA COMPUTAÇÃO

MAURÍCIO CITADINI BILÉSSIMO

ANÁLISE COMPARATIVA DOS *FRAMEWORKS* LARAVEL E CAKEPHP NA IMPLEMENTAÇÃO DA DISTRIBUIÇÃO KAPPA EM UMA APLICAÇÃO WEB

CRICIÚMA

2018

MAURÍCIO CITADINI BILÉSSIMO

ANÁLISE COMPARATIVA DOS *FRAMEWORKS* LARAVEL E CAKEPHP NA IMPLEMENTAÇÃO DA DISTRIBUIÇÃO KAPPA EM UMA APLICAÇÃO WEB

Trabalho de Conclusão de Curso, apresentado para obtenção do grau de bacharel no curso de Ciência da Computação da Universidade do Extremo Sul Catarinense, UNESC.

Orientador: Prof. Me. Luciano Antunes

Coorientador: Prof. Dr. Álvaro José Back

CRICIÚMA

2018

MAURÍCIO CITADINI BILÉSSIMO

ANÁLISE COMPARATIVA DOS *FRAMEWORKS* LARAVEL E CAKEPHP NA IMPLEMENTAÇÃO DA DISTRIBUIÇÃO KAPPA EM UMA APLICAÇÃO WEB

Trabalho de Conclusão de Curso aprovado pela Banca Examinadora para obtenção do Grau de bacharel, no Curso de Ciência da Computação da Universidade do Extremo Sul Catarinense, UNESC, com Linha de Pesquisa em Engenharia de sistemas.

Criciúma, 26 de Novembro de 2018.

BANCA EXAMINADORA

Prof. Luciano Antunes – Me. - (UNESC) - Orientador

Prof. Álvaro Back – Dr. - (UNESC) - Coorientador

Prof. Kristian Madeira - Dr. - (Unesc)

Prof. Fabrício Giordani – Esp. - (Unesc)

Dedico este trabalho à minha família,
meus amigos e meus orientadores
e professores.

AGRADECIMENTOS

Agradeço a Deus, minha família por todo o apoio que sempre me deram, meu orientador, professor Luciano Antunes e meu coorientador, professor Álvaro José Back, por me ajudarem e me guiarem durante todo o projeto.

Agradeço também a todos os professores do curso de Ciência da Computação da Unesc.

RESUMO

Quando o processo de desenvolvimento de um novo *software* é iniciado, uma das questões a se pensar é como estruturá-lo, de forma que seu código seja compreensível e de fácil manutenção. Com o objetivo de ajudar neste processo, os *frameworks* fornecem uma estrutura base, a qual é seguida como padrão durante o desenvolvimento da aplicação. Tendo em vista a quantidade de *frameworks* PHP existentes no mercado, este trabalho tem como objetivo comparar os *frameworks* Laravel e CakePHP no desenvolvimento de uma aplicação *web*, que terá a distribuição Kappa implementada. Foram criadas duas aplicações idênticas, uma com o *framework* Laravel e outra com o *framework* CakePHP, onde ambas têm a distribuição Kappa implementada. As análises foram feitas através de consulta da documentação, ferramentas nativas de cada *framework*, e também a ferramenta de linha de comando, *ApacheBench* do servidor Apache. A distribuição Kappa foi implementada com sucesso e os resultados obtidos demonstram que o *framework* CakePHP mostrou-se melhor na maioria das análises realizadas, como por exemplo na análise referente ao número de bancos de dados suportados, onde o CakePHP tem suporte a cinco bancos de dados, enquanto o Laravel tem suporte a apenas quatro, ou também na parte das transações no banco de dados, onde o CakePHP consegue executar um insert numa media de 0,007 segundos, enquanto o Laravel executa um insert numa media de 0,081 segundos.

Palavras-chave: *Frameworks* PHP. Análises comparativas. Laravel. CakePHP. Desenvolvimento *web*.

ABSTRACT

When the process of developing new software is started, one of the issues to think about is how to structure it, so that your code is understandable and easy to maintain. In order to assist in this process, the frameworks provide a base structure, which is a standard during application development. Considering the number of PHP frameworks on the market, this work aims to compare the Laravel and CakePHP frameworks in the development of a web application, which will have the Kappa distribution implemented. Two identical applications were created, one with the Laravel framework and another with the CakePHP framework and both have the Kappa distribution implemented. The analyzes were done by consulting the documentation, each frameworks own tools, and also the command line tool, ApacheBench Apache server. The Kappa distribution was successfully implemented and the results show that the CakePHP framework has been better in most analyzes. For exemple the analysis of the number of databases supported, where CakePHP supports five databases , while Laravel supports only four, or also in the database transactions, where CakePHP can perform an insert in an average of 0.007 seconds, while Laravel performs an insert in an average of 0.081 seconds.

Keywords: PHP Frameworks. Comparative analysis. Laravel. CakePHP. Web development.

LISTA DE ILUSTRAÇÕES

Figura 1 – Linguagens para Internet 16
Figura 2 – Resultado do comando *ApacheBench* 36
Figura 3 – Laravel Debugbar.....37
Figura 4 – DebugKit 37
Figura 5 – Documentação do *framework* Laravel sobre a parte de suporte a bancos de dados 38
Figura 6 – Tela da aplicação 38
Figura 7 – Seleção do arquivo para importação.....39
Figura 8 – Mensagem de erro para arquivo com extensão diferente de .csv.....39
Figura 9 – Dados importados para a tabela *medias* do banco de dados 40
Figura 10 – L-moments encontrados após processar os dados da aplicação.....41
Figura 11 – Parâmetros Kappa calculados com sucesso.....41
Figura 12 – Mensagem de erro quando não é possível calcular os parâmetros 42
Figura 13 – Limpar dados da aplicação 42
Figura 14 – Aplicação com os dados removidos.....43
Figura 15 – Bancos de dados de cada *framework* 44

LISTA DE TABELAS

Tabela 1 – Bancos de dados suportados pelo *framework* Laravel (Versão 5.7) 24
Tabela 2 – Bancos de dados suportados pelo *framework* CakePHP (Versão 3.6) ...24
Tabela 3 – Testes de requisições com o framework CakePHP.....46
Tabela 4 – Testes de requisições com o framework Laravel 46
Tabela 5 – Testes de transações do banco de dados com o framework CakePHP..48
Tabela 6 – Testes de transações do banco de dados com o framework Laravel.....49

LISTA DE ABREVIATURAS E SIGLAS

API	*Application Programming Interface*
ASP	*Active Server Pages*
CRUD	*create, read, update, delete*
GEV	*Generalized Extreme Value*
HTML	*HyperText Markup Language*
HTTP	*HyperText Transfer Protocol*
JSP	*Java Server Pages*
LMEs	*L-moment estimates*
MB	*MegaBytes*
MVC	*Model View Controller*
PHP	*HyperText Preprocessor*
PHP/FI	*Personal Home Page/Forms Interpreter*
RESTfull	*Representational State Transfer*
SGBD	Sistema Gerenciador de Banco de Dados
SQL	*Structured Query Language*
W3C	*Word Wide Web Consortium*
XML	*Extensible Markup Language*

SUMÁRIO

1 INTRODUÇÃO 11
1.1 OBJETIVO GERAL 13
1.2 OBJETIVOS ESPECÍFICOS 13
1.3 JUSTIFICATIVA 13
1.4 ESTRUTURA DO TRABALHO 14
2 DESENVOLVIMENTO WEB 16
2.1 PADRÃO MVC 18
3 LINGUAGEM DE PROGRAMAÇÃO PHP 19
3.1 FRAMEWORKS 22
3.1.1 LARAVEL 23
3.1.2 CAKEPHP 24
3.2 BANCO DE DADOS 25
3.2.1 MYSQL 25
4 KAPPA 27
5 TRABALHOS CORRELATOS 31
5.1 ESTUDO COMPARATIVO DE FRAMEWORKS PHP, COM ENFOQUE NO CODEIGNITER 31
5.2 ANÁLISE DE FRAMEWORKS DE DESENVOLVIMENTO WEB PARA A CONVERSÃO DE UMA APLICAÇÃO DESKTOP 31
5.3 FRAMEWORKS PARA DESENVOLVIMENTO EM PHP 32
5.4 ESTUDO COMPARATIVO ENTRE FRAMEWORKS JAVA PARA DESENVOLVIMENTO DE APLICAÇÕES WEB: JSF 2.0, GRAILS E SPRING WEB MVC 33
5.5 ANÁLISE DE FRAMEWORKS PARA O DESENVOLVIMENTO DE SISTEMA DE CONTROLE SINDICAL 33
6 APLICAÇÃO WEB PARA A FUNÇÃO KAPPA DE QUATRO PARÂMETROS ... 35
6.1 METODOLOGIA 35
7 RESULTADOS E DISCUSSÃO 44
8 CONCLUSÃO 51
REFERÊNCIAS 53

1 INTRODUÇÃO

O setor de Tecnologia da Informação (TI) se estabeleceu completa e definitivamente em diversas áreas e segmentos (REBECCHI, 2012).

As aplicações do tipo *web* são exemplos da informatização, já que cada vez mais tarefas, desde o envio de mensagens até o pagamento de contas, podem ser realizadas de maneira online, utilizando apenas um computador com acesso à Internet. O que antigamente não era muito aceito pela população por ser considerado perigoso, hoje em dia se tornou frequente na vida da maioria das pessoas, que utilizam essas aplicações para tarefas comuns do dia-a-dia (DA SILVA, 2017).

Quando o processo de desenvolvimento de um novo *software* é iniciado, uma das grandes questões a se pensar é como estruturá-lo, de forma que seu código seja compreensível e de fácil manutenção. Com o objetivo de ajudar neste processo, os *frameworks* fornecem uma estrutura base, a qual é seguida como padrão durante o desenvolvimento da aplicação (BAGESTAN, 2016). Segundo Dias, Amorim e Carvalho Júnior (2015), os *frameworks* são considerados como caixas de ferramentas que auxiliam, agilizam e facilitam o desenvolvimento de *softwares.* A *World Wide Web Consortium* (W3C) definiu padrões para a criação e desenvolvimento de aplicações *web* que facilitam e normalizam a sua utilização. Quando um projeto *web* é definido e segue esses padrões ele é capaz de beneficiar não só o usuário que o utiliza, mas também todos que acompanham este *software*, o mesmo sendo compatível com as diversas versões e tipos de navegadores *web.*

Entretanto, como há uma grande variedade de *frameworks* disponíveis para a linguagem de programação *HyperText Preprocessor* (PHP), o momento de decidir qual *framework* utilizar em uma aplicação pode gerar uma série de dúvidas. Isto se deve a quantidade de *frameworks* existentes, cada um com suas particularidades, características positivas e negativas (BAGESTAN, 2016). Aplicações mal arquitetadas e sem uso de padrões no desenvolvimento podem acarretar vários problemas na solução final (PISCKE, 2007).

Tendo em vista a quantidade de *frameworks* existentes, este trabalho irá falar um pouco sobre o Laravel e o CakePHP, os quais serão utilizados nesse projeto e também implementar a distribuição Kappa de quatro parâmetros nas aplicações desenvolvidas. O Laravel utiliza a arquitetura *Model, View, Controller*

(MVC), entre suas características, está o desenvolvimento de aplicações seguras, de modo rápido, com código simples e limpo. Quando se trata de interfaces gráficas é utilizado uma *engine* de *template* chamada *Blade,* que propicia uma série de ferramentas que contribuem para a criação de interfaces elegantes e funcionais de forma rápida, evitando a duplicação de códigos. Quanto a comunicação com o banco de dados, é utilizada uma implementação chamada *Eloquent*, que traz funcionalidades que facilitam a inserção, atualização, busca e exclusão de registros (NASCIMENTO, 2017). O CakePHP também utiliza a arquitetura MVC. Segundo Silva (2011), é reconhecido pela simplicidade e pela alta produtividade que ele proporciona. Baseado no framework Rails, ele funciona através de um gerador de código que produz todo o *create*, *read*, *update*, *delete* (CRUD) deixando para o programador desenvolver a regra de negócio de cada sistema. Utiliza como roteamento o padrão *Representational State Transfer* (RESTfull*)*, onde dentro dessa arquitetura são utilizados os métodos *get*, *post*, *put* e *delete*, que servem para manipular os recursos. As URLs seguem um padrão que às divide em quatro partes: dominio/controlador/ação/parâmetro. Projetado para tornar as tarefas comuns de desenvolvimento web simples e fáceis, fornece uma caixa de ferramentas *all-in-one* para você começar (CAKE SOFTWARE FOUNDATION, 2018).

A distribuição Kappa de quatro parâmetros é definida pelos parâmetros x, alfa, k e h, e inclui, como casos particulares, as distribuições Logística, Generalizada de Valores Extremos e Generalizada de Pareto (DANTAS; PINTO, 2011).

Hosking (1994), introduziu a distribuição Kappa de quatro parâmetros como generalização de diversas distribuições comumente usadas na análise de cheias. Dentre as diversas distribuições consideradas como casos especiais da distribuição Kappa, destacam-se: GEV, GLO e GLPA (PANSERA, 2013).

Diante do contexto apresentado anteriormente, o seguinte trabalho propõe uma análise comparativa em desempenho, velocidade, tempo de resposta, entre outros; entre os *frameworks* Laravel e CakePHP no desenvolvimento de uma aplicação *web* que terá a distribuição Kappa implementada, afim de saber qual dos dois *frameworks* atende de melhor maneira no desenvolvimento de aplicações *web*.

1.1 OBJETIVO GERAL

Comparar os *frameworks* Laravel e CakePHP no desenvolvimento de uma aplicação *web*, que terá o método estatístico Kappa implementado.

1.2 OBJETIVOS ESPECÍFICOS

Os objetivos específicos desta pesquisa consistem em:

a) identificar os conceitos de desenvolvimento web;
b) descrever os *frameworks* Laravel e CakePHP;
c) implementar o método estatístico Kappa;
d) desenvolver uma aplicação utilizando o Laravel e outra aplicação utilizando o CakePHP;
e) fazer a análise comparativa entre os *frameworks;*
f) analisar os resultados obtidos.

1.3 JUSTIFICATIVA

As aplicações *web* têm vindo a ganhar uma preponderância muito grande, especialmente na área do software de gestão. Durante vários anos, este tipo de software apenas era disponibilizado aos utilizadores na plataforma desktop, mas esta tendência tem vindo a ser contrariada. Atualmente, os grandes produtores de software de gestão nacionais já disponibilizam aos seus clientes soluções de gestão totalmente baseadas na *web* (LACERDA, 2013).

Entretanto, mesmo com a constante evolução de métodos, técnicas e ferramentas, a entrega de software em prazos e custos estabelecidos e com qualidade mínima desejável nem sempre é alcançada. A consequência dessa situação é o desenvolvimento de códigos desnecessários, falta de padronização nos métodos e nos atributos, e também repetição de tarefas com a mesma finalidade na programação. Uma forma que as empresas podem trabalhar para conseguir driblar essa situação é usar ferramentas que as auxiliem automatizando e padronizando o projeto, juntamente com ferramentas geradoras de código. Um bom exemplo dessas ferramentas é o uso dos *frameworks* (SILVA, 2011).

O processo de criação de uma aplicação *web* pode revelar-se bastante complexo logo na fase de seleção da plataforma de desenvolvimento uma vez que existem várias opções disponíveis e diversos *frameworks* (LACERDA, 2013).

Dentre os fatores que levaram a escolha de se comparar os *frameworks* Laravel e CakePHP estão os fatos de serem *frameworks* robustos, populares entre os desenvolvedores *web*, são ferramentas estabelecidas no mercado. Os dois *frameworks* baseiam-se na arquitetura MVC. O Laravel permite o desenvolvimento de aplicações seguras, de modo rápido, com código simples e limpo (NASCIMENTO, 2017). O CakePHP fornece uma base robusta e sólida para as suas aplicações, permitindo customizar e estender facilmente muitos aspectos. Também fornece uma convenção básica de nomes de arquivos e de tabelas do banco de dados, mantendo toda a aplicação consistente e lógica (Cake Software Foundation, 2018).

Outro motivo é também o auxílio que o comparativo trará no processo de escolha de um *framework*, visando os benefícios que o mesmo traz ao desenvolvedor e ao cliente. Tendo em vista que nem sempre o cliente terá uma boa infraestrutura, com um acesso à Internet razoavelmente bom, ou as vezes computadores mais antigos, com pouco desempenho, processamento, etc. Então a escolha de uma boa ferramenta e/ou *framework* para o desenvolvimento do produto final, podem suprir os pontos negativos citados.

1.4 ESTRUTURA DO TRABALHO

Este trabalho de pesquisa é constituído por oito capítulos, sendo que o primeiro capítulo aborda uma introdução sobre o tema da pesquisa, e sua estrutura é formada pela introdução, objetivo geral e seus objetivos específicos, e da justificativa para o qual o projeto será desenvolvido.

O segundo capítulo fala sobre desenvolvimento *web,* mostra algumas linguagens de programação para a *web* e aborda também sobra o padrão de desenvolvimento MVC.

No capítulo três, foram abordados os assuntos sobre a linguagem de programação PHP, seu surgimento, suas versões, falado também sobre *frameworks* em geral e logo após abordado sobre os *frameworks* desta pesquisa (Laravel e

CakePHP) e por fim, falado sobre bancos de dados de uma forma geral e também sobre o banco de dados utilizado nesta pesquisa que foi o MySQL.

O quarto capítulo aborda o assunto sobre estatística, suas utilizações e sobre o método Kappa de quatro parâmetros, o qual foi implementado nas aplicações deste trabalho.

O capítulo cinco trata dos trabalhos correlatos a este projeto de pesquisa, onde abordam alguns dos objetivos que este projeto também contém.

O capítulo seis traz o trabalho desenvolvido, sua explicação passo a passo, funcionamento e a metodologia aplicada.

No sétimo capítulo, são mostradas as análises aplicadas para cada *framework* e seus respectivos resultados.

Para finalizar o capítulo oito traz a conclusão sobre tudo que foi abordado ao longo do trabalho.

2 DESENVOLVIMENTO WEB

Sistemas de informação deixaram de ser algo opcional para as empresas, e se tornaram alvo das maiores necessidades que uma empresa pode possuir, o papel que esses sistemas possuem em relação à organização é de extrema importância, uma vez que nestes ficam as responsabilidades de manter, e gerar novos dados, que por sua vez podem se caracterizar como um dos maiores patrimônios da empresa.

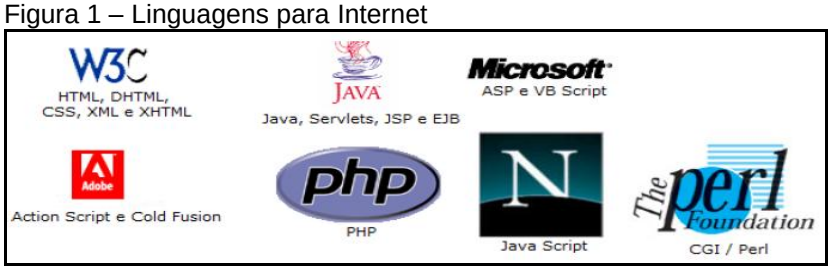
Dados sigilosos, informações de clientes, informações de produtos e estoque, financeiro, relatórios gerenciais e muitas outras informações ficam a cargo desses sistemas.

A atualidade do mercado está contida num mundo globalizado, e a Internet tornou-se parte integrante desse novo sistema de negociação (CENTENARO, 2014). A utilização de aplicativos *web* tem se tornado indispensável no dia-a-dia das pessoas. Prefere-se acessar uma aplicação *online* que permite realizar as mesmas tarefas que um aplicativo *desktop* fornece, sem ter o transtorno de precisar instalar o programa no próprio computador (SILVA, 2011).

A cada dia cresce o número de usuários, serviços, informações e facilidades disponíveis de forma online, é cada vez mais fácil a utilização de computadores, notebooks ou outros equipamentos eletrônicos para pagar uma conta, agendar uma entrevista, fazer uma chamada de vídeo conferência, acessar o saldo da conta bancária e até fazer uma compra. Desenvolvimento de sistemas para *web* diferencia de um desenvolvimento *desktop*, exatamente no que diz respeito ao público alvo, maneira de acesso e manutenção do mesmo (CENTENARO, 2014).

Para o desenvolvimento de sistemas *web* existem várias linguagens de programação tais como *Java*, *PHP*, *C#, Java Script*, *Ruby*, entre outras, como é possível visualizar na figura 1 (SILVA, 2011).

Figura 1 – Linguagens para Internet



Fonte: Silva (2011).

Ainda segundo Centenaro (2014), um sistema *web* torna-se mais acessível, uma vez que pode ser acessado através de navegadores, e tem características multiplataforma, já que não obriga o usuário final a possuir um determinado sistema operacional, e sim apenas um navegador de internet. Em relação à manutenção, fica mais simples já que não necessita o usuário final a executar procedimentos de instalação, facilita o suporte não necessitando diretamente visitas ao cliente nem assistência remota, pois os arquivos ficam armazenados em um servidor *web* e sua manipulação afeta instantaneamente o usuário final.

As atividades realizadas no âmbito do desenvolvimento *web* são, essencialmente, o desenho, a programação e a publicação. Pode-se criar tanto uma simples página estática, como uma complexa aplicação que será acedida por milhares de usuários em todo o mundo. Normalmente, o desenvolvimento *web* pode ser dividido em dois grandes grupos (LACERDA, 2013): *Client Side* e *Server Side.*

O *Cliente Side* corresponde a parte das aplicações que são executadas do lado do cliente (*Browser*). Enquadram-se neste grupo as tarefas relacionadas com a apresentação das páginas, sendo que a maioria do desenvolvimento nesta área é feito com recurso à linguagem *JavaScript* e sendo também frequente a utilização de ferramentas de desenvolvimento para o desenho da interface gráfica e posterior geração do código *HyperText Markup Language* (HTML). Já o *Server Side*, pelo contrário, engloba todas as tarefas relacionadas com a parte da aplicação que é executada no lado do servidor. Trata-se, fundamentalmente, do desenvolvimento de código que será executado para dar resposta a um pedido realizado por um cliente. *ASP, PHP, Java, Python,* são algumas das linguagens de programação e plataformas utilizadas no desenvolvimento do código que será executado do lado do servidor (LACERDA, 2013).

Ainda segundo Lacerda (2013), o termo *Web Development* (Desenvolvimento web) surge, muitas vezes, associado ao conceito de *Web Design.* Embora se tratem de conceitos relacionados (sendo freqüente, em empresas de menor dimensão, a mesma pessoa desempenhar funções em ambas as áreas), na prática são conceitos totalmente diferentes. O *Web Design* reporta-se, exclusivamente, aos aspetos visuais da aplicação, não tendo, portanto, qualquer responsabilidade ao nível da programação. A principal responsabilidade de um *Web Designer* é definir a forma como um *Web Site* será apresentado ao utilizador e como

este interage com a aplicação, não tendo qualquer responsabilidade quanto à tecnologia que suportará o projeto. Isto significa que, do lado do cliente, poderemos ter a intervenção de um *Web Designer* e de um *Web Developer,* o primeiro tratando dos aspetos visuais e o segundo desenvolvendo o código que os suportarão e que será executado no lado do cliente ou do servidor. Este código pode realizar tarefas tão díspares como a apresentação ou ocultação de elementos das páginas, a interação com os sistemas de bases de dados para obtenção de informação ou para efetuar a sua manutenção, a validação e filtragem dos dados inseridos pelos utilizadores, o tratamento de questões relacionadas com a internacionalização e localização das aplicações, entre outras.

2.1 PADRÃO MVC

O padrão MVC, é um padrão de arquitetura de software que propõe a separação das classes de um sistema em três grupos: *Model*, que é o componente responsável por fazer todos os acessos e transições com o banco de dados, representa o modelo do nosso banco de dados na aplicação. O componente *View*, que recebe e apresenta as informações para o usuário final, e por fim o *Controller*, que é responsável por definir a maneira com que a interface gráfica reage as ações do usuário, ou seja, faz a intermediação entre o *Model* e a *View*, e quaisquer outros recursos necessários para fazer o processamento de uma requisição (SILVA, 2011). Ainda segundo Silva (2011), o objetivo principal do padrão MVC é separar dados e lógica de negócio (*Model*) da interface do usuário (*View*) e do fluxo da aplicação (*Controller*). Desta forma os três componentes podem sofrer alterações sem impactar nos demais.

Antes da criação do padrão MVC, os projetos de interface para o usuário tendiam a agrupar os objetos de modelo, visão e controle (PISCKE, 2007).

As vantagens de se utilizar o padrão MVC no desenvolvimento de uma aplicação estão diretamente relacionadas com a manutenibilidade que se consegue atingir quando se divide um sistema em módulos específicos, que tratam de responsabilidades diferentes e únicas. Ainda dentro desse contexto, a reutilização é bem explorada, pois uma mesma funcionalidade que é desenvolvida para uma página, pode ser reutilizada em outras páginas que a requisitem (BALTHAZAR et al., 2007).

3 LINGUAGEM DE PROGRAMAÇÃO PHP

No processo de desenvolvimento web é necessário a utilização de uma linguagem *server-side.* O PHP é uma das linguagens que são utilizadas no lado do servidor, e que será utilizada no desenvolvimento deste trabalho.

Criada por Rasmus Lerdorf, ela é uma linguagem de *script open-source* de uso geral, muito utilizada e especialmente guarnecida para o desenvolvimento de aplicações *web* que podem ser embutidas dentro do HTML (DIAS; AMORIM; CARVALHO JÚNIOR, 2015).

Embora se trate de uma linguagem maioritariamente utilizada no lado do servidor, também é utilizada na criação de *scripts* que podem ser executados numa linha de comandos ou na criação de interfaces de utilizador através da utilização da extensão PHP-GTK (LACERDA, 2013). Segundo Nascimento (2017), a interpretação do código é realizada na parte do servidor a partir do módulo PHP, sendo que este também é o responsável por gerar a página que é entregue ao cliente. A evolução da linguagem integrou novas características, como aplicações em linha de comando. Devido ao grande suporte da linguagem a base dados, é extremamente simples construir uma página dinâmica com uso da mesma.

O PHP era uma solução para utilização em suas páginas pessoais, contudo necessidades foram surgindo e novas versões foram lançadas, contendo novos recursos, como, interpretador de comandos SQL, orientação a objetos, herança, polimorfismo, apontadores de memória e outros (CENTENARO, 2014).

Segundo Lacerda (2013), a primeira versão do PHP surgiu em 1994 com o nome de *Personal Home Page Tools* (PHP Tools). Esta versão inicial apenas permitia o registo dos acessos a uma determinada página, mas, ao longo do tempo, foram sendo adicionadas novas funcionalidades ao projeto. Em junho de 1995 o código fonte foi disponibilizado à comunidade com o objetivo de aumentar o envolvimento de novos programadores nas tarefas de correção de erros e de melhoramento geral. Em novembro de 1997 surgiu a versão 2.0, tendo, na altura, sido chamada de *Personal Home Page/Forms Interpreter* (PHP/FI). Esta nova versão já se aproximava mais daquilo que é, hoje em dia, o PHP deixando de ser, apenas, um conjunto de ferramentas para auxílio na construção de páginas *web* e dando os primeiros passos na criação de uma linguagem de programação.

A versão 3.0 foi lançada em junho de 1998 e representou um ponto de viragem muito importante no projeto, que passou a ser desenvolvido por um grupo de programadores (Rasmus Lerdorf, Andi Gutmans e Zeev Suraski). Nesta nova versão, o interpretador foi totalmente reescrito para permitir a criação de uma nova linguagem de programação. Para além desta importante funcionalidade, o PHP 3.0 disponibilizava uma interface para comunicação com várias bases de dados e diversas *Application Programming Interface* (APIs) de uso genérico. Também permitia um elevado nível de extensibilidade, o que atraiu muitos programadores para o projeto e o consequente surgimento de novos módulos que expandiam as funcionalidades da plataforma. A partir daqui a expressão PHP deixou de significar Personal Home Page Tools passando a significar *PHP: Hypertext Preprocessor*. A partir desta nova versão, assistiu-se a um enorme aumento da popularidade da linguagem e do número de servidores onde estava instalada. Logo após o lançamento do PHP 3.0, começou o trabalho na versão 4.0 com o objetivo de melhorar o desempenho em aplicações complexas e aumentar os níveis de modularidade (LACERDA, 2013).

Ainda segundo Lacerda (2013), este trabalho culminou com o lançamento da versão 4.0 em maio de 2000. Os objetivos inicialmente propostos foram alcançados e esta nova versão apresentava um novo interpretador chamado de *Zend Engine*. Para além das melhorias referidas, a versão 4.0 apresentava novas funcionalidades que alavancaram a sua utilização em ambientes empresariais. Uma gestão melhorada dos recursos, diversos melhoramentos ao nível da programação orientada a objetos, API de encriptação e gestão de sessões *HyperText Transfer Protocol* (HTTP), entre outros, são alguns exemplos das novas funcionalidades disponibilizadas pelo PHP 4.0. A versão 5.0 foi lançada em julho de 2004, juntamente com a versão 2.0 do *Zend Engine*. Esta nova versão, para além de melhorar significativamente as bibliotecas/extensões existentes, apresentou novas funcionalidades, destacando-se as relacionadas com o suporte ao *XML* e *Web Services*, bem como o suporte nativo para a base de dados SQLite.

A versão 7.0 do PHP foi lançada em dezembro de 2015, e vem com modificações mínimas na sintaxe, focando mais na melhoria de performance e no *upgrade* do *core interpreter* da linguagem (IMASTERS, 2018). Um ano depois do lançamento da versão 7.0, surge a versão 7.1, lançada oficialmente em dezembro de 2016, trazendo consigo algumas melhorias como: *Nullable Types* (possibilidade de

um parâmetro receber um tipo específico ou *null*); Habilidade de pegar múltiplas exceções num mesmo bloco *catch*; Criação de um pseudo-tipo chamado *Iterable*; Habilidade de definir visibilidade para constantes de classes (*public, private, protected*); Diversas melhorias à extensão *Curl*, dentre elas, suporte a *HTTP/2 Server Push*; Tipo *void* para parâmetros e retornos de funções/métodos; Incrementos na utilização list(); O suporte a mcrypt() foi removido da linguagem; (TREINAWEB, 2018). Hoje em dia o PHP está na sua versão mais recente que é a 7.2, lançada em novembro de 2017, trazendo consigo novos métodos como: get_class(), desabilita o parâmetro nulo; Impedir number_format() de retornar zero; *Argon2 Password Hash. Object typehint; libsodium* (IMASTERS, 2018).

Hoje em dia o PHP é utilizado em massa, também por aplicações referências como MediaWiki, Facebook, Drupal, Joomla, WordPress, Magento e Oscommerce. A linguagem também disponibiliza suporte a um grande número de banco de dados, inclusive MySQL (CENTENARO, 2014).

Segundo Lacerda (2013), o PHP é distribuído sobre uma licença própria chamada de PHP *License*. Esta licença é uma das mais permissivas colocando apenas limitações quanto à utilização do nome PHP. Embora o PHP seja uma linguagem interpretada (sendo o *Zend Engine* o interpretador) existem diversos compiladores que permitem desassociar a linguagem do interpretador e obter um maior desempenho na execução do código. Existem vários compiladores para PHP sendo de destacar os seguintes: *Phalanger, HipHop for PHP, PHC (The Open Source PHP Compiler)*.

O PHP concorre diretamente com outras tecnologias voltadas ao desenvolvimento web, como o *Active Server Pages* (ASP) e *Java Server Pages* (JSP), sendo que o ASP é um meio de executar Visual Basic no lado servidor e depende de um servidor Microsoft IIS para ser executado, e o JSP, apesar dos recursos da linguagem Java, se mostra mais complicado na hora de escrever uma aplicação, em questão de tamanho e complexidade de código-fonte (MILANI, 2010). Em comparação entre o PHP e demais tecnologias de desenvolvimento *web*, também deve se observar que a grande maioria dos servidores de hospedagem em domínios de internet está voltada para o PHP, e isso continua em constante crescimento. O PHP também conta com uma comunidade ativa que ajuda a difundir e aperfeiçoar cada vez mais a tecnologia. Devido à variedade e qualidade de seus

recursos o PHP se tornou um fenômeno no desenvolvimento *web* (BAGESTAN, 2016).

3.1 FRAMEWORKS

Segundo Silva (2011), um *framework* é uma estrutura de suporte definida em que um outro projeto de software pode ser organizado e desenvolvido, quando se analisa o conceito no âmbito do desenvolvimento de software. Um *framework* pode incluir programas de suporte, bibliotecas de código, linguagens de *script* e outros softwares para ajudar a desenvolver e juntar diferentes componentes de um projeto de software. É um software parcialmente completo, ou seja, é um subsistema, projetado para ser instanciado e que deve ser melhor especificado, integrando a funcionalidade da aplicação com o *framework*. Fornecem uma arquitetura de esqueleto para a aplicação, propiciando o reuso de projeto e classes específicas do sistema. Tal arquitetura é definida através de classes de objetos e suas interações. São implementados como um conjunto de classes de objetos concretos e abstratos em uma linguagem de programação orientada a objetos, sendo assim, os *frameworks* são específicos da linguagem (BAGESTAN, 2016).

Além de facilitarem a vida dos programadores, também trazem um ótimo ganho de rendimento no desenvolvimento de aplicações (SILVA, 2011). Segundo Silva (2011), são projetados com o propósito de facilitar o desenvolvimento de software, habilitando projetistas e programadores a gastarem mais tempo detalhando as exigências de negócio do software do que com detalhes de baixo nível do sistema. Para que projetos de software sejam considerados um *framework*, as características básicas devem ser respeitadas: Precisa ser reutilizável, facilitar o desenvolvimento de sistemas, possuir boa documentação, ser completo para o que se propõe e ser eficiente.

Existem muitas vantagens quanto a utilização de *frameworks* que são:

a) **utilidade**: um dos objetivos principais dos *frameworks* é auxiliar no desenvolvimento de aplicações e *softwares*.
b) **segurança**: os *frameworks* mais eficazes são projetados de modo a garantir a segurança de quem programa.
c) **extensibilidade**: os *frameworks* permitem estender funcionalidades nativas.

d) **economia de tempo**: as funções que manipulam as imagens são de um caminho bem mais exigível, precisando de mais tempo na realização em suma, com o uso de um *framework* pode-se realizar-se em pouquíssimo tempo.

e) **ajuda fácil**: os desenvolvedores de *frameworks* geralmente disponibilizam material de qualidade nos *Websites*, com uma vasta documentação a respeito. Além disso, a maioria dos *frameworks* têm uma comunidade de desenvolvedores dispostos a se ajudarem entre si.

Segundo Bagestan (2016), os *frameworks* de aplicações *web*, são um tipo mais recente de *framework*. Geralmente a arquitetura de um *framework web* é baseada no padrão MVC. Um *framework* MVC suporta apresentação de dados de maneiras diferentes, e permite a interação com cada uma dessas apresentações. Quando os dados são modificados, o modelo de sistema é alterado e os controladores associados a cada visão atualizam sua apresentação. E por fim, ainda segundo Bagestan (2016), os *frameworks* ajudam a manter o código limpo organizado, e fornecem algumas facilidades genéricas, permitindo que o desenvolvedor projete as aplicações de forma cada vez mais rápida.

3.1.1 LARAVEL

O Laravel é um *framework* PHP utilizado para o desenvolvimento *web*, que utiliza arquitetura MVC. Sua principal característica é o desenvolvimento de aplicações seguras e com boa performance de modo rápido, com código simples e limpo, já que o Laravel incentiva o uso de boas práticas de programação. Quando se trata de interfaces gráficas é utilizado uma *engine* de *template* chamada Blade, que propicia uma série de ferramentas que contribuem para a criação de interfaces elegantes e funcionais de forma rápida, evitando a duplicação de código. Quanto à comunicação com o banco de dados é utilizada uma implementação chamada *Eloquent*, que traz funcionalidades que facilitam a inserção, atualização, busca e execução de registro. Logo, com uma configuração simples e pequena é possível se conectar ao banco de dados e trabalhar com o mesmo (NASCIMENTO, 2017).

Bancos de dados suportados pelo *framework* Laravel (tabela 1).

Tabela 1 – Bancos de dados suportados pelo *framework* Laravel (Versão 5.7)

Banco de dados
MySQL
PostgreSQL
SQLite
SQL Server

Fonte: https://laravel.com/

3.1.2 CAKEPHP

A melhor definição para o CakePHP diz que ele permite que os usuários de PHP de todos os níveis possam desenvolver rapidamente aplicações *web* robustas. É reconhecido pela simplicidade e pela alta produtividade que ele proporciona. Baseado no *framework* Rails para Ruby, ele funciona através de um gerador de código que produz todo o CRUD, deixando para o programador desenvolver a regra de negócio de cada sistema (SILVA, 2011).

Fornece uma base robusta e sólida para as suas aplicações, permitindo customizar e estender facilmente muitos aspectos. Também fornece uma convenção básica de nomes de arquivos e de tabelas do banco de dados, mantendo toda a aplicação consistente e lógica (CAKE SOFTWARE FOUNDATION, 2018).

Assim como no Rails, o CakePHP utiliza como roteamento padrão, o RESTfull. O termo originou-se no ano de 2000, em uma tese de doutorado sobre a *web*, realizado pelo norte-americano Roy Fielding. REST é uma arquitetura cliente/servidor sem estado, no qual os recursos são identificados por suas URLs e são manipulados através de suas representações. Dentro de uma arquitetura REST, são utilizados os métodos *get*, *post*, *put* e *delete*, que servem para manipular os recursos. As URLs seguem um padrão que ás divide em quatro partes: www.dominio.com/controlador/ação/parâmetros (SILVA, 2011).

Bancos de dados suportados pelo *framework* CakePHP (tabela 2).

Tabela 2 – Bancos de dados suportados pelo *framework* CakePHP (Versão 3.6)

Banco de dados
MySQL
PostgreSQL
SQLite
SQL Server
Oracle (Através de um plugin da comunidade)

Fonte: https://cakephp.org/

3.2 BANCO DE DADOS

Um banco de dados pode ser definido como um sistema computadorizado de manutenção de registros. Sua funcionalidade é armazenar dados de forma que os usuários busquem e atualizem esses dados, e que quando processados, gerem informações e tem como principal objetivo armazenar dados (MATIOLI, 2010).

Segundo Werner (2014), um banco de dados é um sistema que visa armazenar quaisquer tipos de dados eletrônicos que tenham relevância dentro de um contexto, para ser utilizado por um sistema de aplicação. Desta forma um banco de dados pode ser utilizado tanto para um sistema pessoal, quanto para organizações e empresas de todos os portes e de todos os setores, variando de hospitais, onde os dados armazenados são de pacientes e doenças, por exemplo, a fábricas, que armazena dados de produção, produtos acabados, etc., a outras áreas de atuação. Ou seja, é um conjunto de dados, dispostos em uma determinada ordem, que relacionados possuem significado, atendendo requisitos que visam atender a necessidade de um usuário.

Ainda segundo Werner (2014), para realizar operações em um banco de dados é necessário utilizar um sistema próprio para a manipulação e gerenciamento do próprio banco conhecido como Sistema de Gerenciamento de Banco de Dados (SGBD). Alguns bancos de dados possuem a alternativa de armazená-los em arquivos e que seja escrito um programa para sua manipulação, porém os SGBDs oferecem uma gama de vantagens quando comparado a coleções de arquivos.

3.2.1 MYSQL

O MySQL é um banco de dados livre e de código aberto, distribuído pela empresa Oracle, é apoiado por uma comunidade grande e ativa de desenvolvedores. Foi um projeto que teve início em 1984, cujo principal objetivo era de garantir aos seus usuários, a liberdade de compartilhar e alterar seus aplicativos, e assim tornando-os livres (SILVA, 2012). É conhecido por sua facilidade de uso, sendo usado pela Nasa, HP, Bradesco, Sony e muitas outras empresas. Sua interface simples, e também sua capacidade de rodar em vários sistemas operacionais, são alguns dos motivos para este software ser tão usado atualmente, e seu uso estar crescendo cada vez mais (MYSQL, 2018).

Foi originalmente desenvolvido pela empresa sueca TCX, que necessitava de um servidor de banco de dados que operasse com grandes escalas de dados rapidamente sem exigir caríssimas plataformas de hardware (SÃO PAULO, 2001).

É um banco de dados *multi-thread*, que atende à vários usuários ao mesmo tempo e suas tarefas são executadas em *back-end.* É flexível para trabalhar bem em ambientes muito exigentes, como aplicações *web*. Ao mesmo tempo, pode potencializar aplicações embutidas, depósitos de dados, indexação de conteúdo e software de distribuições, sistemas redundantes altamente disponíveis, processamento de transação *on-line* e muito mais (SILVA, 2012).

Aqui são algumas de suas características:

a) portabilidade (suporta praticamente qualquer plataforma atual).

b) compatibilidade (existem drivers ODBC, JDBC e .NET e módulos de interface para diversas linguagens de programação, como Delphi, Java, C/C++, Python, Perl, PHP, ASP e Ruby).

c) excelente desempenho e estabilidade.

d) pouco exigente quanto a recursos de hardware.

e) facilidade de uso.

f) é um Software Free com base na GPL.

g) contempla a utilização de vários Storage Engines como MyISAM, InnoDB, Falcon, BDB, Archive, Federated, CSV, Solid.

h) suporta controle transacional.

i) suporta Triggers.

j) suporta Cursors (Non-Scrollable e Non-Updatable).

k) suporta Stored Procedures e Functions.

l) replicação facilmente configurável

m) interfaces gráficas (MySQL Toolkit) de fácil utilização cedidos pela MySQL Inc.

4 KAPPA

A distribuição Kappa de quatro parâmetros introduzida por Hosking (1994), é definida pelos parâmetros x, alfa, *k* e *h,* e é considerada por várias distribuições estabelecidas incluindo a distribuição de *Generalized Extreme Value* (GEV), distribuição logística generalizada, distribuição generalizada de Pareto e distribuição de Gumbel que foram usadas para a modelagem de eventos extremos, sendo, portanto, teoricamente capaz de representar variáveis hidrológicas e hidrometeorológicas. são casos especiais da distribuição Kappa. A distribuição Kappa tem como função densidade de probabilidade dada por:

$$f(x) = \frac{1}{\alpha}\left[1 - \frac{k(x-\varepsilon)}{\alpha}\right]^{\frac{1}{k-1}} F(x)^{1-h}$$

Onde x é o valor máximo anual, α e ε são os parâmetros de escala e posição respectivamente, k e h são os parâmetros de forma e F(x) é a função acumulativa de probabilidade representada por:

$$F(x) = \left\{1 - h\left[1 - \frac{k(X-\varepsilon)}{\alpha}\right]^{1/k}\right\}^{1/h}$$

Se h > 0 e K > 0
LI = $\varepsilon + \alpha(1 - h^{-k})/k$
LS = $\varepsilon+\alpha/k$
Se h > 0 e K = 0
LI = $\varepsilon + \alpha \ln(h)$
LS =∞
Se h > 0 e k < 0
LI = $\varepsilon + \alpha(1 - h^{-k})/k$
LS =∞
Se h $\leq$ 0 e k > 0
LI = -∞
LS = $\varepsilon+\alpha/k$
Se h $\leq$ 0 e k =0
LI = -∞
LS = ∞
Se h $\leq$ 0 e k <0

LI = $\varepsilon+\alpha/k$
LS = ∞

Muitos métodos de estimação de parâmetros foram propostos para ajustar a distribuição estatística aos dados hidrológicos. O método de estimativa atual usa funções lineares da estatística de ordem esperada, ou seja, *L-moment estimates* (LMEs). A estimativa dos LMEs fornece um método alternativo de estimação análogo aos momentos convencionais, mas com várias vantagens, incluindo estimativas mais robustas com menos viés (Hosking, 1994). LMEs, no entanto, desde que sejam computáveis, muitas vezes não são viáveis. Além disso, para a distribuição Kappa, as distribuições assintóticas dos LMEs não são facilmente tratáveis.

Hosking (1994) introduziu a distribuição kappa de quatro parâmetros, uma generalização de distribuições de três parâmetros comuns e, em particular, da distribuição GEV, para auxiliar na adaptação de dados quando três parâmetros não são suficientes. Usar a distribuição Kappa em simulações onde um comprometimento com uma distribuição particular de três parâmetros deve ser evitada, como em Hosking (1993) pode também ser benéfico. A função de distribuição cumulativa da distribuição Kappa é: (WINCHESTER, 2000, tradução nossa).

$$F(x) = \begin{cases} \left\{1 - h\left[1 - \frac{k(x-\varepsilon)}{\alpha}\right]^{\frac{1}{k}}\right\}^{\frac{1}{k}}, \\ e^{-\left[1-\frac{k(x-\varepsilon)}{\alpha}\right]^{\frac{1}{k}}}, \\ \left[1 - he^{\frac{-(x-\varepsilon)}{\alpha}}\right]^{\frac{1}{h}}, \\ e^{-e^{-\frac{(x-\varepsilon)}{\alpha}}} \end{cases}$$

que inclui os casos onde k = 0 e/ou h = 0 considerando suas formas limitantes como k -> 0 e/ou h -> 0. O inverso de F(x) produz a função quantil:

$$x(F) = \varepsilon + \frac{\alpha}{k}\left[1 - \left(\frac{1-F^h}{h}\right)^k\right]$$

As principais vantagens de usar o método dos LMEs são que as estimativas dos parâmetros são mais confiáveis (ou seja, menor erro médio-quadrado de estimativa) do que o método de estimativas de momentos, particularmente de pequenas amostras e são geralmente computacionalmente mais tratáveis do que as estimativas de máxima verossimilhança. No entanto, o método de estimativa de LMEs de Kappa ainda requer iteração de Newton-Raphson. Além disso, o espaço de parâmetros é restrito para garantir a existência dos LMEs e a unicidade dos parâmetros. Portanto, a rotina fornecida por Hosking às vezes falha ao obter o método de estimativas de LMEs para alguns conjuntos de dados, porque não existe uma solução exclusiva dentro da região restrita.

A estimativa dos parâmetros pelo método dos LMEs é realizada estimando os parâmetros de locação ξ escala α e os dois parâmetros de forma k e h resolvendo as seguinte relações entre estes parâmetros e os momentos amostral $\lambda 1$, $\lambda 2$ e $\tau 3$ e $\tau 4$

$$\lambda_1 = \xi + \alpha(1 - g_1)/k$$
$$\lambda_2 = \alpha(g_1 - g_2)/k$$
$$\tau_3 = \frac{-g_1 + 3g_2 - 2g_3}{g_1 - g_2}$$
$$\tau_4 = \frac{-g_1 + 6g_2 - 10g_3 + 5g_4}{g_1 - g_2}$$

Em que

$$g_r = \begin{cases} \dfrac{r\Gamma(1+k)\Gamma\left(\frac{r}{h}\right)}{h^{1+k}\Gamma\left(1+k+\frac{r}{h}\right)}, & se\ h > 0 \\ \dfrac{r\Gamma(1+k)\Gamma(-k-r/h)}{(-h)^{1+k}\Gamma\left(1-\frac{r}{h}\right)}, & se\ h < 0 \end{cases}$$

Dados e $\tau 3$ e $\tau 4$ resolve-se as equações 1 e 2 iterativamente para k e h.

O algoritmo é valido para as seguintes condições impostas para os parâmetros

a) h > -1;
b) Se h < 0 então hk > -1
c) h > -1
d) k =0,725h > -1

A função LogVersosimilhança é dada por:

$$LnL = -nln(\alpha) + \left(\frac{1}{k} - 1\right)\sum_{i=1}^{n} ln\left\{1 - \left[\frac{k(x_i - \xi)}{\alpha}\right]\right\} + \left(\frac{1}{h} - 1\right)\sum_{i=1}^{n} \ln\left\{\left[1 - h\left(1 - \frac{k(x_i - \xi)^{1/k}}{\alpha}\right)\right]\right\}$$

A distribuição Kappa de quatro parâmetros pode ser utilizada na área de hidrologia, para a definição da probabilidade de ocorrência de eventos extremos de precipitações diárias, como no trabalho de Dantas e Pinto (2011), onde foi feita uma análise de frequência regional das precipitações máximas diárias da Bacia do Rio São Francisco no estado de Minas Gerais. Também como no trabalho de Pansera (2013), onde foi feita uma análise de frequência local e regional de chuvas máximas diárias no estado do Paraná, utilizando distribuições generalizadas, dentre elas a distribuição Kappa de quatro parâmetros.

5 TRABALHOS CORRELATOS

Para viabilizar este trabalho foram pesquisados alguns trabalhos na mesma área. Neste capitulo são apresentados trabalhos que possuem conteúdos semelhantes ou que tenham alguma relação com este projeto de pesquisa.

5.1 ESTUDO COMPARATIVO DE FRAMEWORKS PHP, COM ENFOQUE NO CODEIGNITER

O estudo realizado apresenta um comparativo sobre os principais *frameworks* PHP, e a aplicação do CodeIgniter em um sistema para geração e manutenção de orçamentos. A principal motivação para realização desse estudo é proporcionar um comparativo entre os cinco principais *frameworks* PHP - CakePHP, CodeIgniter, Symfony, Yii e Zend -, demonstrando as características de cada um, proporcionando o conhecimento e a aplicação prática, no desenvolvimento de um sistema que faz uso das funcionalidades do *framework* CodeIgniter, e que realiza a tarefa de geração e manutenção de orçamentos. Durante o estudo foi possível observar características comuns entre os *frameworks*, assim como diferenças entre eles, que permitiram ampliar as possibilidades no desenvolvimento de aplicações web (BAGESTAN, 2016).

5.2 ANÁLISE DE FRAMEWORKS DE DESENVOLVIMENTO WEB PARA A CONVERSÃO DE UMA APLICAÇÃO DESKTOP

As aplicações web têm vindo a ganhar uma preponderância muito grande, especialmente na área do software de gestão. Durante vários anos, este tipo de software apenas era disponibilizado aos utilizadores na plataforma desktop mas esta tendência tem vindo a ser contrariada. Atualmente, os grandes produtores de software de gestão nacionais já disponibilizam aos seus clientes soluções de gestão totalmente baseadas na web como uma alternativa viável à sua linha de produtos para o desktop. Este processo tem sido lento e difícil atendendo às caraterísticas do software e também a alguma resistência por parte dos utilizadores que temem perder a experiência e rapidez de utilização a que estavam habituados.

O processo de criação de uma aplicação web pode revelar-se bastante complexo logo na fase de seleção da plataforma de desenvolvimento uma vez que existem várias opções disponíveis (p. ex. .NET, PHP, Python, Ruby, Java e Perl) e, para cada uma, identificam-se diversos frameworks.

Com este trabalho efetua-se uma análise a algumas das plataformas (.NET, Java e PHP) e frameworks de desenvolvimento de aplicações web e seleciona-se aquela que melhor se adequará para a conversão de um software que, no presente, apenas está disponível na plataforma desktop. Neste processo, também são analisados frameworks mais vocacionados para o desenvolvimento das interfaces do utilizador, com o objetivo de selecionar aquele que permitirá uma melhor experiência de utilização. Esta análise passa pelo estudo das caraterísticas e funcionalidades de cada um dos frameworks e pela realização de testes de desempenho às plataformas (LACERDA, 2013).

5.3 FRAMEWORKS PARA DESENVOLVIMENTO EM PHP

Uma das grandes vantagens do PHP é sua facilidade de aprendizado. Ao ler poucas páginas de tutoriais ou de algum livro, um programador já é capaz de montar um formulário HTML e de criar um script PHP que processe os dados fornecidos pelo usuário. Isso favoreceu o rápido aumento do número de programadores e o surgimento de grandes softwares.

Contudo, essa facilidade de aprendizado permitiu o surgimento de programas, digamos, pouco eficientes, programas vulneráveis a ataques, com péssima manutenibilidade, os denominados spaggeti code, com PHP e HTML inseridos no mesmo script, e também a dificuldade de se trabalhar em equipes, entre outros. Um dos argumentos usados por programadores de outras linguagens era que “PHP facilita a geração de programas ruins”.

Argumentos como esse não são mais válidos. Vários fatores contribuíram para esta mudança: as novas e robustas características de orientação a objetos advindas do PHP5, o apoio de grandes empresas como a Oracle e a IBM, a profissionalização dos programadores, graças a certificações como a da Zend.

Neste livro, pretende-se descrever alguns frameworks que estão sendo desenvolvidos e utilizados ativamente por uma comunidade novamente entusiasmada com a programação para um renovado ambiente web. Inicialmente

comenta-se sobre o framework e suas vantagens. Comenta-se também sobre algumas tecnologias usadas pelos frameworks que serão apresentados. Os capítulos podem ser lidos na ordem em que o leitor achar mais interessante, mas seguir a ordem em que aparecem facilita a comparação e, consequentemente, a identificação do leitor com framework que mais lhe agrada. Isso porque é muito difícil, em qualquer comparação deste porte, apontar o melhor. Para o programador, é sempre importante conhecer várias opções antes de escolher uma para utilizar ou contribuir, pois como dizem “a melhor ferramenta é a que você conhece e confia” (MINETTO, 2007).

5.4 ESTUDO COMPARATIVO ENTRE FRAMEWORKS JAVA PARA DESENVOLVIMENTO DE APLICAÇÕES WEB: JSF 2.0, GRAILS E SPRING WEB MVC

A plataforma Java é a mais utilizada para o desenvolvimento de aplicações web corporativas. O mercado exige que o software seja desenvolvido com o menor tempo e custo possível e com qualidade. Para isso tornou-se essencial a utilização de frameworks, pois eles aumentam a produtividade, maximizam o reuso e reduzem a possibilidade de erros. Atualmente existem diversos frameworks web disponíveis para a plataforma Java. Este trabalho apresenta alguns conceitos relacionados ao desenvolvimento web em Java e descreve três dos principais frameworks utilizados recentemente para essa finalidade: JSF 2.0, Spring Web MVC e Grails. Também realiza uma comparação entre eles. Através da implementação de um protótipo de um sistema financeiro, são verificadas na prática as comparações teóricas apresentadas e verifica-se qual dos frameworks apresenta mais vantagens ao desenvolvedor. Conclui-se que Grails é o mais fácil de aprender e o mais simples de ser utilizado (FRANCO, 2011).

5.5 ANÁLISE DE FRAMEWORKS PARA O DESENVOLVIMENTO DE SISTEMA DE CONTROLE SINDICAL

Para o desenvolvimento de sistemas de maior porte, com grande nível de integração, está se tornando cada vez mais necessário a adoção de um framework. Ele é responsável pela organização do código fonte de todo o sistema, além de

prover diversas funcionalidades que facilitam o processo de desenvolvimento. O problema surge diante da diversidade de frameworks existentes no mercado, devido à quantidade de requisitos e utilidades diferentes, acaba se tornando necessária uma análise aprofundada antes de uma escolha. Esta monografia tem por objetivo analisar uma determinada quantidade de frameworks, levando em consideração o contexto atual da instituição onde será aplicado, um sindicato. Após esta análise, foram feitas comparações entre os frameworks eleitos e, a partir daí, selecionado o mais adequado para a instituição (HENNING, 2013).

6 APLICAÇÃO WEB PARA A FUNÇÃO KAPPA DE QUATRO PARÂMETROS

Este capítulo tem como objetivo apresentar o passo a passo das aplicações desenvolvidas. Apesar de serem duas aplicações desenvolvidas, uma utilizando o *framework* Laravel e outra utilizando o *framework* CakePHP, ambas têm o mesmo objetivo, funcionamento, e o mesmo design em relação a interface gráfica e usabilidade. Nesse caso apresenta-se aqui apenas uma das aplicações, seu funcionamento, passo a passo de utilização, a função que resolve o método estatístico Kappa de quatro parâmetros, as ferramentas utilizadas para o desenvolvimento do trabalho e também as ferramentas utilizadas para a análise dos *frameworks*.

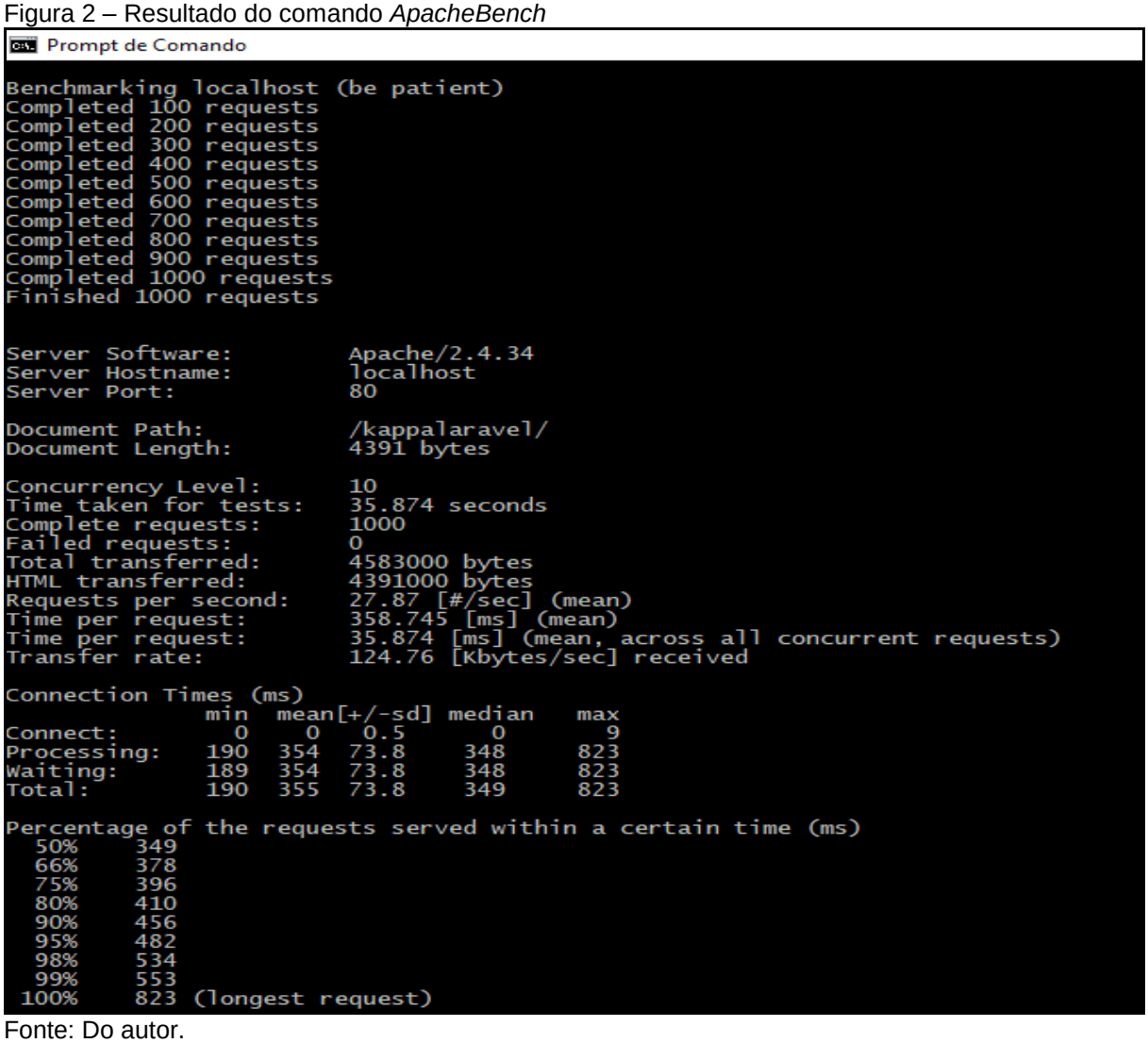
6.1 METODOLOGIA

Para a realização deste trabalho acadêmico foram empregadas as seguintes etapas metodológicas: levantamento bibliográfico; estudo sobre desenvolvimento web; estudo sobre a linguagem de programação PHP e seus respectivos *frameworks* Laravel e CakePHP; foram desenvolvidas duas aplicações utilizando-se a linguagem de programação PHP, ambas tem o mesmo objetivo, porém uma foi desenvolvida utilizando o *framework* Laravel e outra foi desenvolvida utilizando o *framework* CakePHP, onde em cada uma delas foi implementada uma função para resolver o método estatístico Kappa de quatro parâmetros; após prontas as aplicações, foram feitas as análises de cada uma delas com suas ferramentas nativas para a análise, Laravel Debugbar para o *framework* Laravel e DebugKit para o *framework* CakePHP; utilizou-se o *ApacheBench* (AB) para analisar as requisições por segundo de cada *framework* e o tempo médio de cada requisição, e foram também analisadas as documentações de cada um dos *frameworks* para obter as configurações iniciais de cada um deles e também o número de banco de dados que ambos suportam.

As análises comparativas foram feitas em ambiente *localhost*, utilizando um notebook com sistema operacional Windows 10, com 8 GB de memória RAM, processador Intel i5 de 6ª geração com 2.3 GHz e um HD de 500 GB, com o servidor apache instalado, o banco de dados utilizado foi o MySQL e a versão do PHP foi a 7.2.

Para a análise do tempo de resposta nas requisições média de requisições por segundo, e o desvio padrão foi utilizado o seguinte comando: ab -n 1000 -c 10 http://localhost/kappalaravel/, onde indica que foram feitas 1000 requisições com 10 usuários simultaneamente, ou seja, 100 requisições por usuário (figura 2).

Figura 2 – Resultado do comando *ApacheBench*

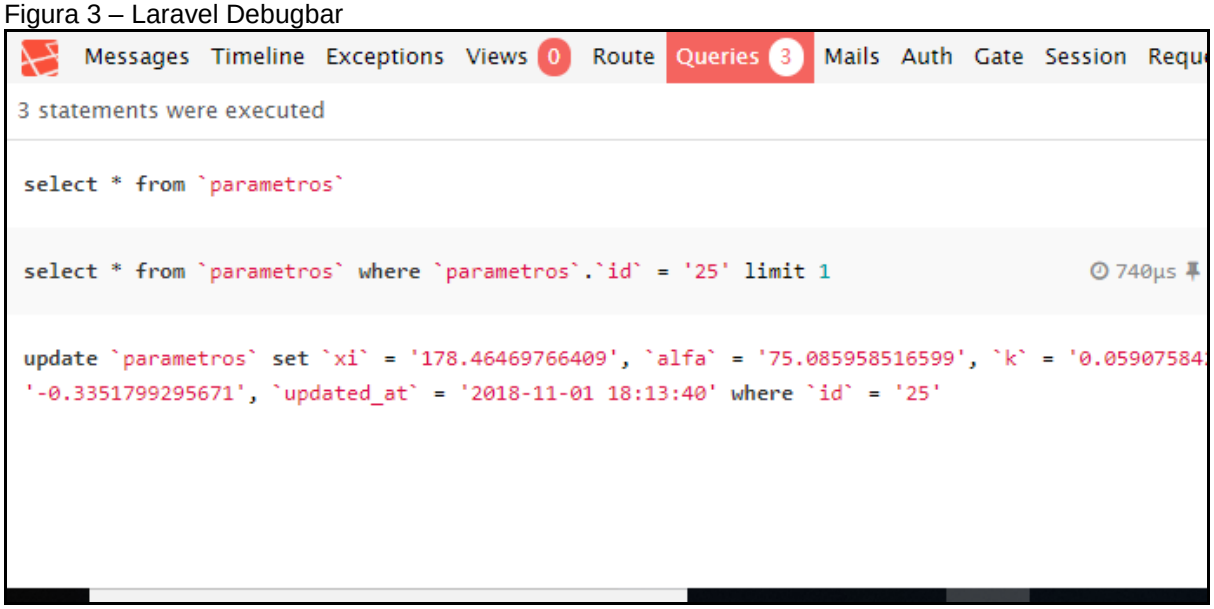


Fonte: Do autor.

Para a parte das análises das transações com o banco de dados, como INSERT, SELECT, UPDATE e DELETE, tempo de carregamento da página e a memória que cada aplicação consome foi utilizado o Laravel Debugbar para o Laravel e para o CakePHP foi utilizado o DebugKit.

Exemplo do resultado de três consultas no banco de dados, mostrados pelo Laravel DebugBar (figura 3).

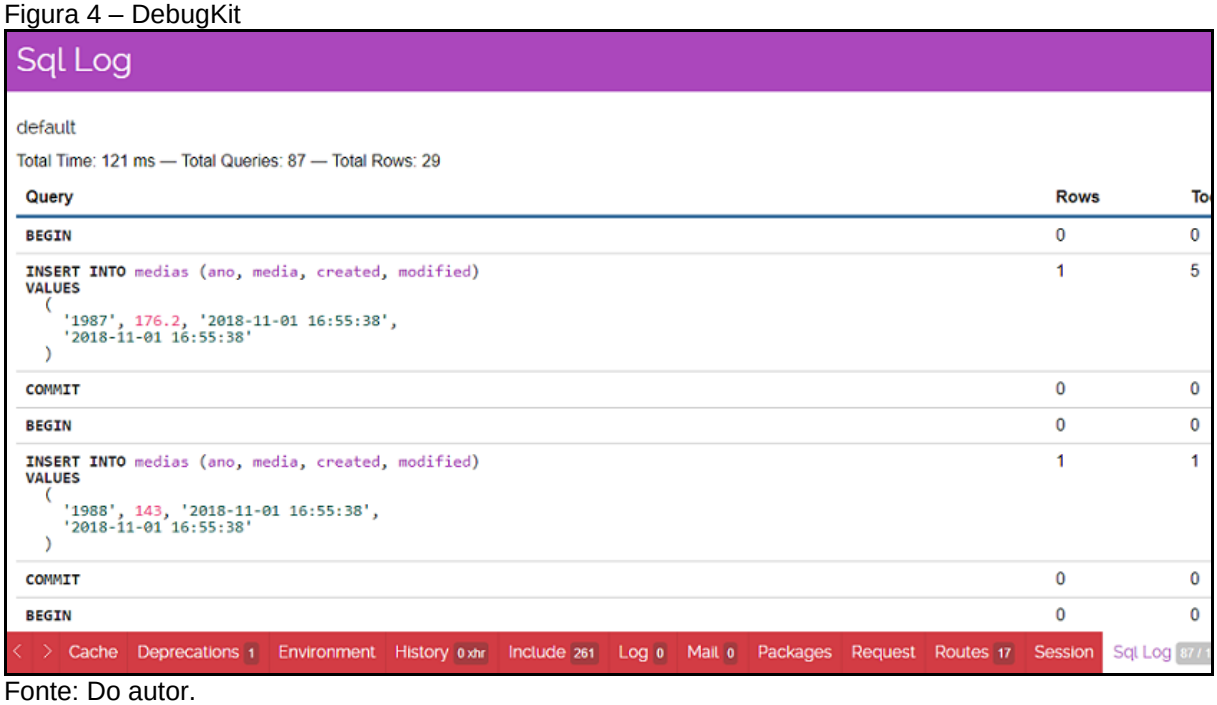
Figura 3 – Laravel Debugbar



Fonte: Do autor.

Exemplo do resultado de duas consultas no banco de dados, mostrados pelo DebugKit (figura 4).

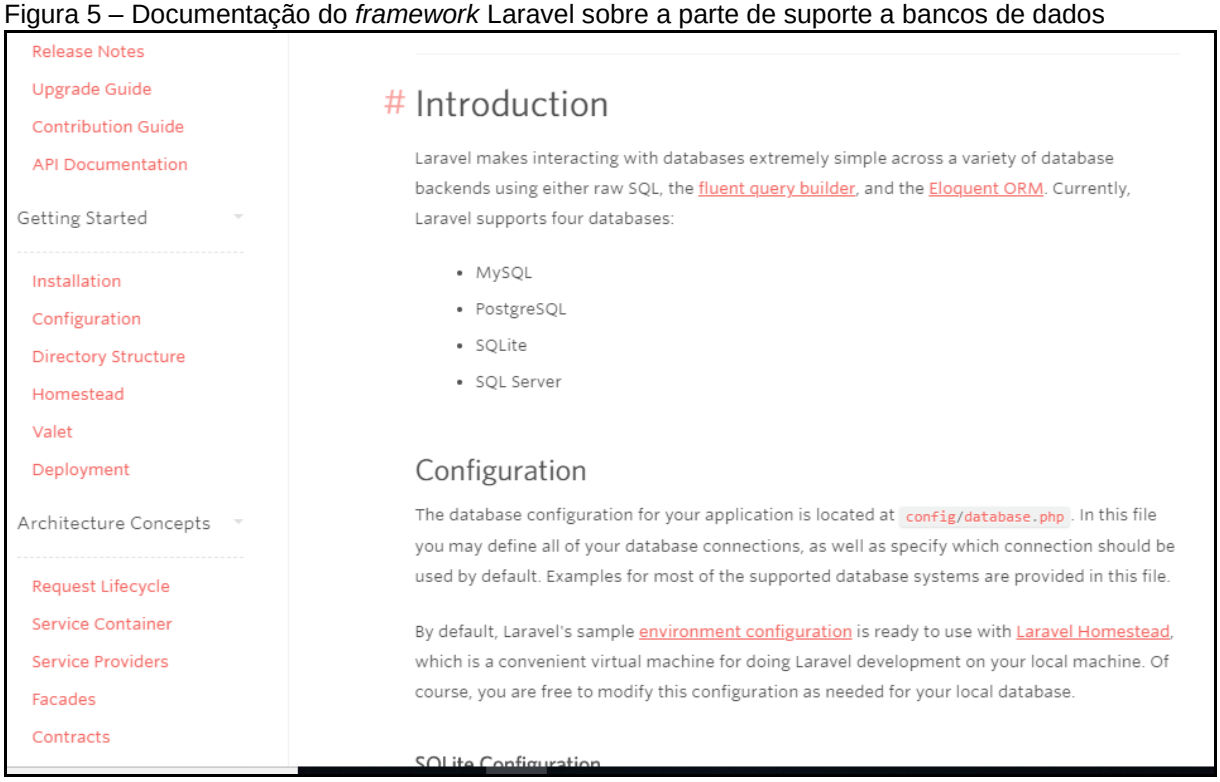
Figura 4 – DebugKit



Fonte: Do autor.

Quanto a parte da quantidade de banco de dados suportados pelos *frameworks,* foi consultado as documentações de ambos para obter essa informação (figura 5).

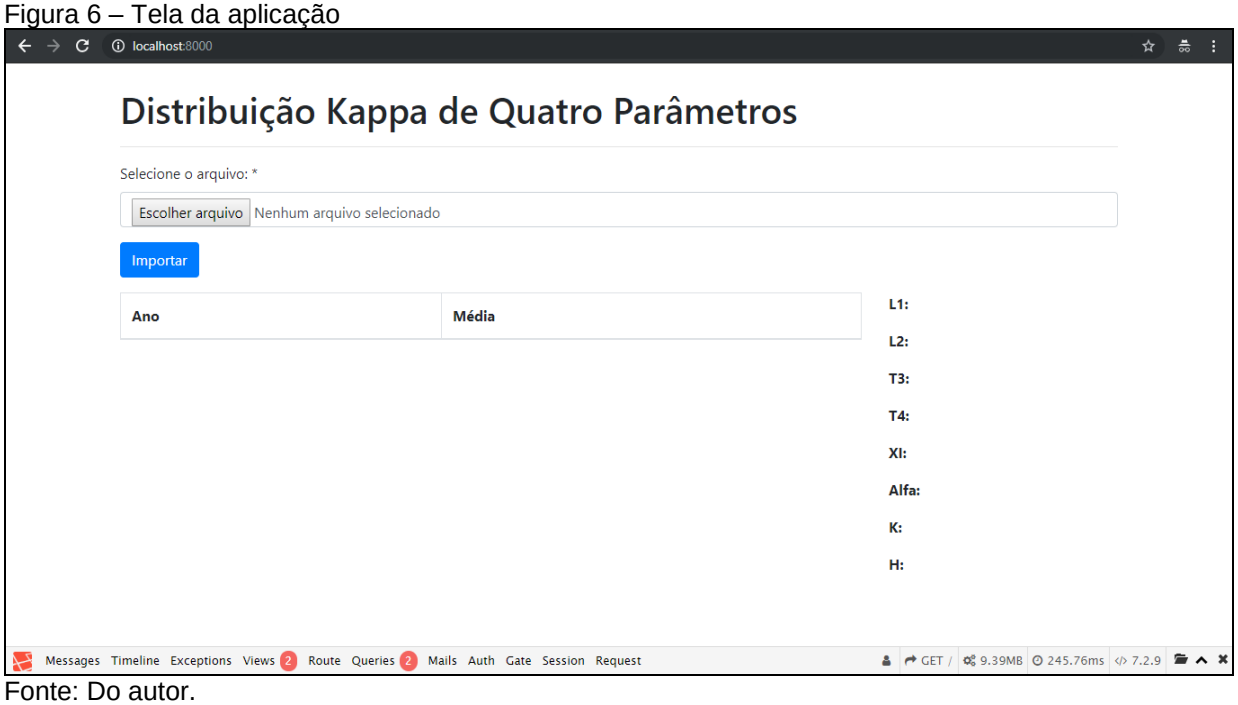
Figura 5 – Documentação do *framework* Laravel sobre a parte de suporte a bancos de dados



Fonte: Do autor.

A figura 6 mostra a tela inicial e única da aplicação, onde através dessa tela é possível executar todos os passos necessários para o funcionamento correto da aplicação;

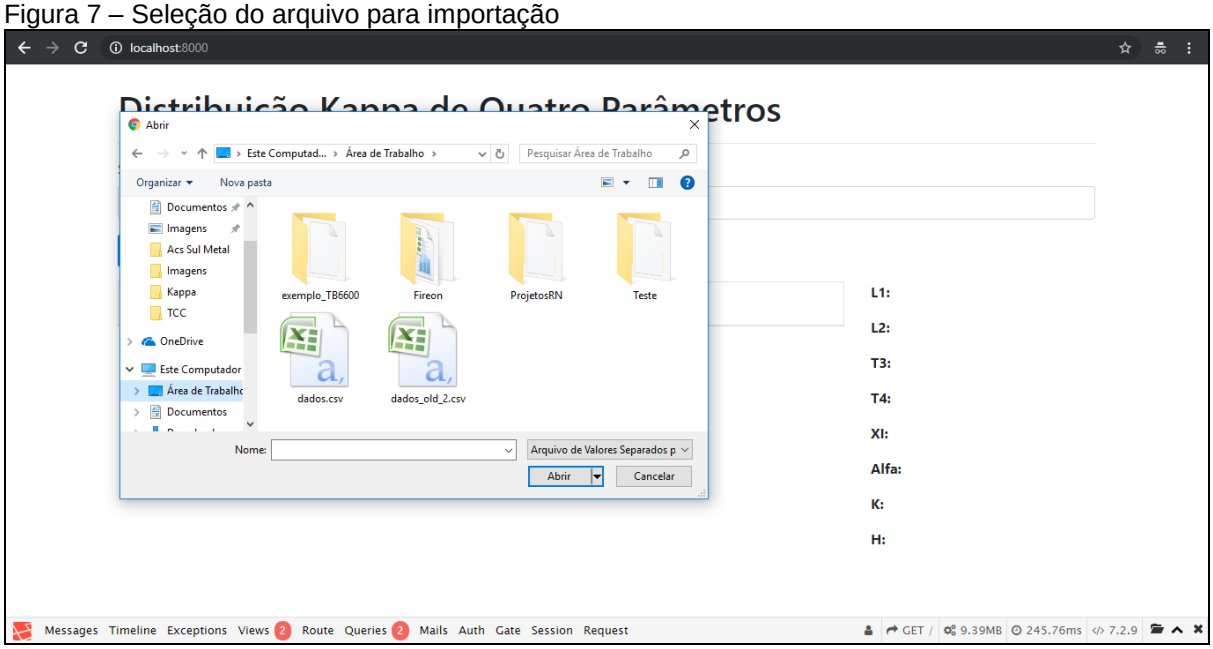
Figura 6 – Tela da aplicação



Fonte: Do autor.

Para utilizar a aplicação, primeiro deve-se escolher um arquivo com a extensão .csv, que contenha duas colunas de dados, a coluna *ano* e a coluna *media*. Para selecionar o arquivo basta clicar sobre o input do tipo file. Ao clicar irá aparecer uma janela do seu sistema operacional para selecionar o arquivo desejado, como mostra a figura 7;

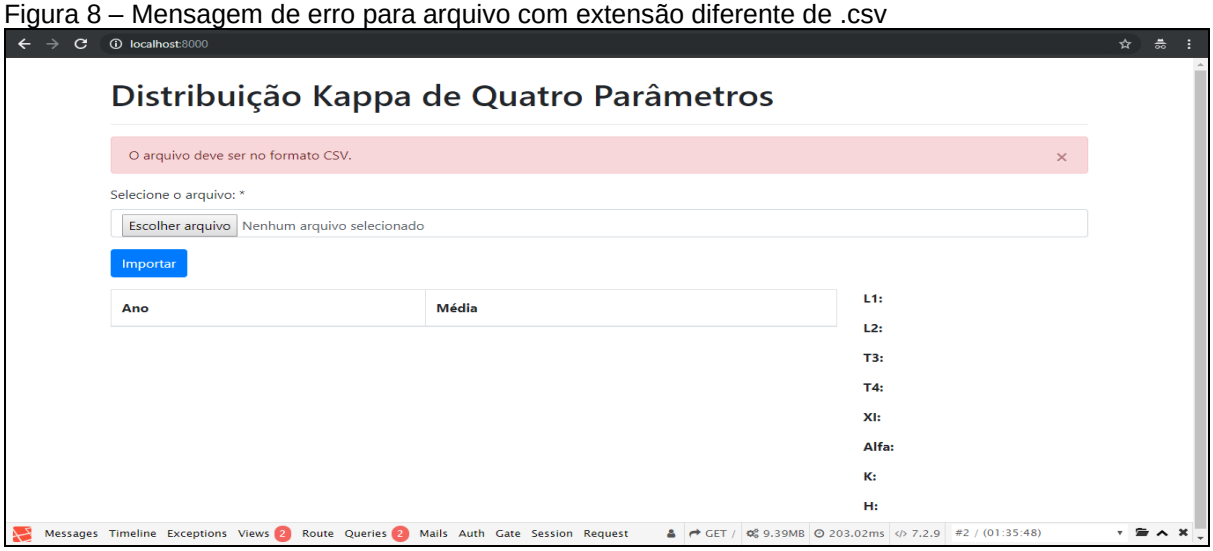
Figura 7 – Seleção do arquivo para importação



Fonte: Do autor.

Caso seja selecionado um arquivo com uma extensão diferente de .csv a aplicação mostra uma mensagem de erro para o usuário informando que o arquivo deve ser no formato .csv (figura 8).

Figura 8 – Mensagem de erro para arquivo com extensão diferente de .csv



Fonte: Do autor.

Após selecionar o arquivo desejado, deve-se clicar no botão *Importar* e a aplicação irá gravar na tabela *medias* do banco de dados, todos os dados que estão contidos no arquivo, e também mostrará três novos botões na aplicação, o botão de *Processar*, *Calcular Kappa* e *Limpar*, como mostra a figura 9.

Figura 9 – Dados importados para a tabela *medias* do banco de dados

Fonte: Do autor.

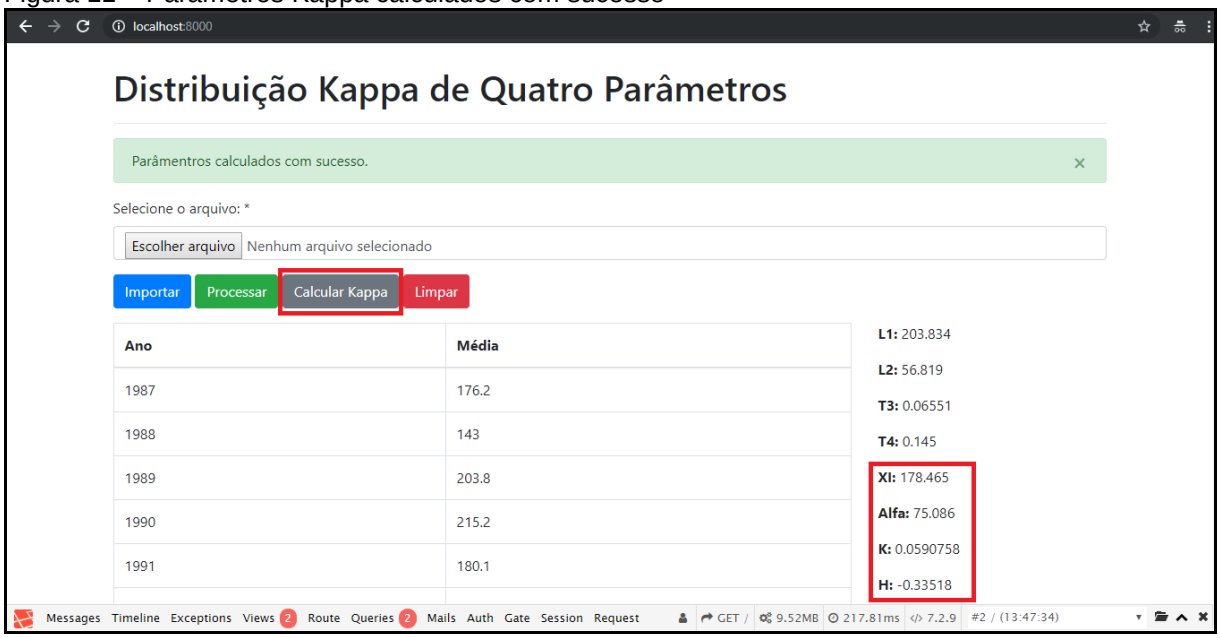
Com os dados importados, o próximo passo é processá-los para poder encontrar os L-moments L1, L2, T3 e T4, onde a partir desses quatros parâmetros pode-se calcular o método Kappa. Para processar os dados importados e encontrar os L-moments, basta clicar no botão *Processar* e a aplicação após encontrar os L-moments, irá salvá-los na tabela *parametros* do banco de dados (figura 10).

Figura 10 – L-moments encontrados após processar os dados da aplicação

Fonte: Do autor.

Com os L-moments encontrados é possível calcular o método estatístico Kappa de Quatro Parâmentros. Para isso basta clicar no botão *Calcular Kappa*, então a aplicação executa a função *kappa()*, se conseguir calcular os quatro paramentos, a função faz um UPDATE na tabela *parametros* do banco de dados para gravar os quatro parâmetros encontrados e após isso retorna para tela os dados encontrados e uma mensagem de sucesso para o usuário (figura 11).

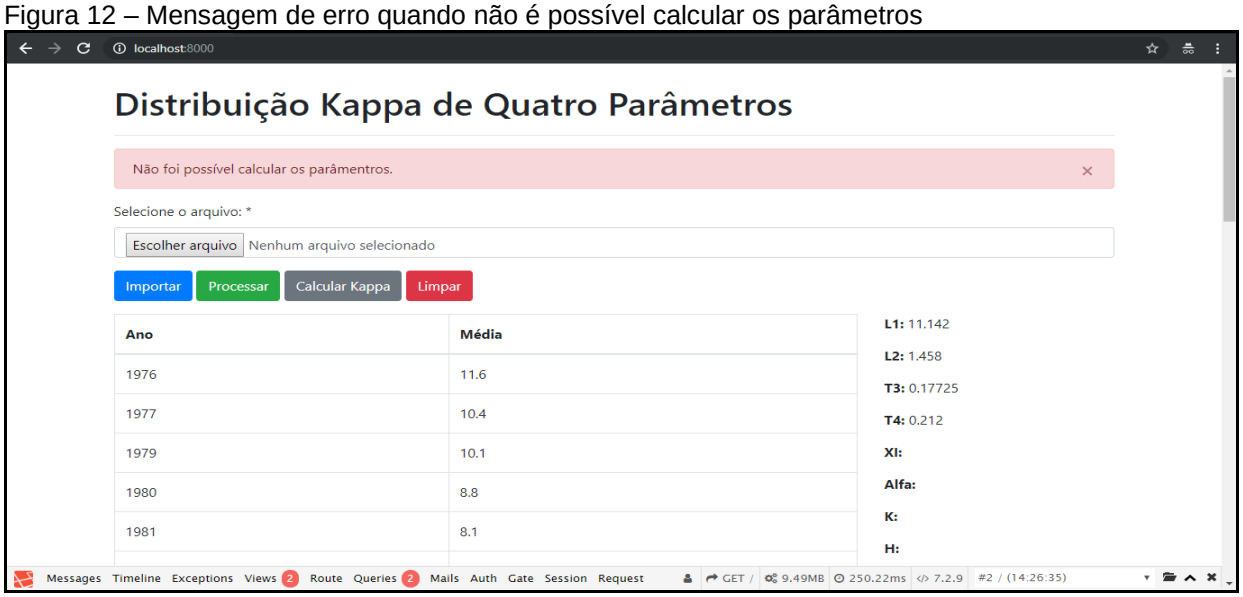
Figura 11 – Parâmetros Kappa calculados com sucesso



Fonte: Do autor.

Se a função Kappa não conseguir calcular os quatro parâmetros, ela retorna uma mensagem de erro para o usuário, informando que não foi possível calcular os parâmetros (figura 12).

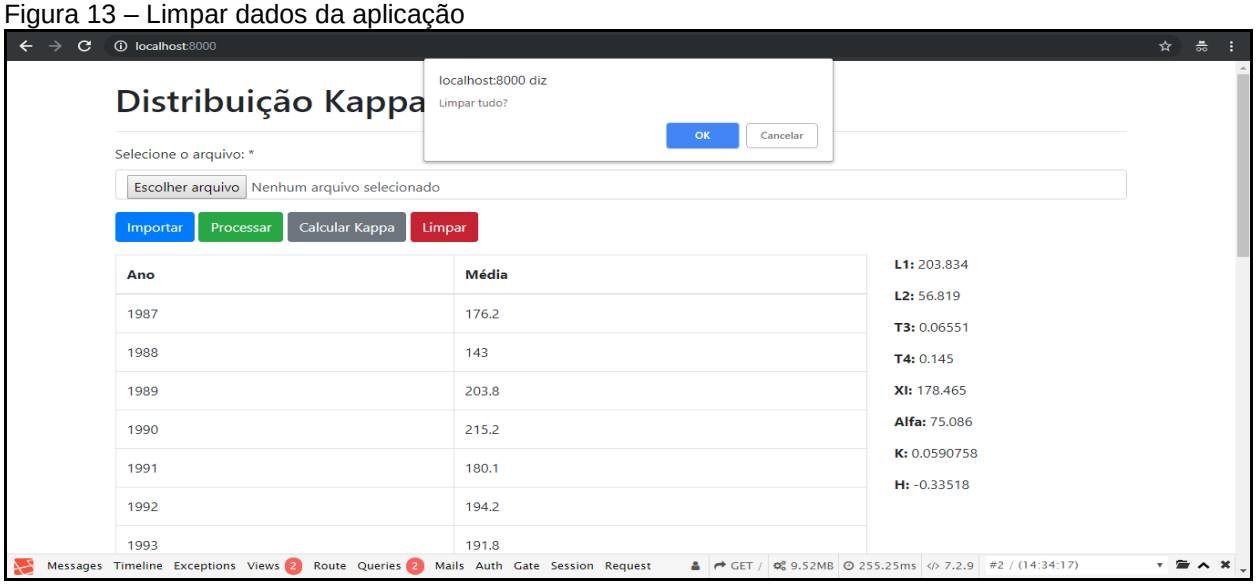
Figura 12 – Mensagem de erro quando não é possível calcular os parâmetros



Fonte: Do autor.

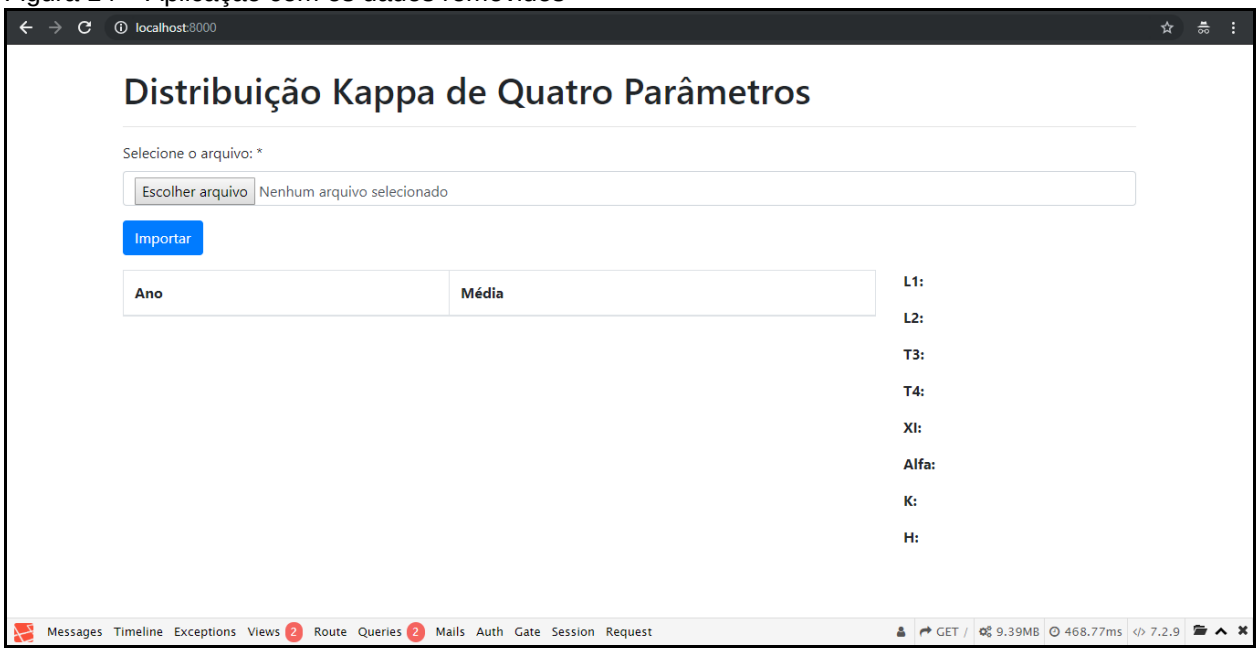
Por fim, para poder importar e calcular novos dados, é preciso limpar os dados atuais da aplicação e do banco de dados. Para isso basta clicar no botão *Limpar*, ao fazer isso a aplicação mostra um *alert* perguntando se deseja realmente limpar tudo, caso confirmado, então a aplicação irá deletar todos os dados do banco de dados, como mostram as figuras 13 e 14.

Figura 13 – Limpar dados da aplicação



Fonte: Do autor.

Figura 14 – Aplicação com os dados removidos



Fonte: Do autor.

7 RESULTADOS E DISCUSSÃO

Através da análise da documentação de cada *framework* foi possível obter as informações sobre a quantidade de banco de dados suportados de ambos, onde pode-se ver que o *framework* CakePHP tem suporte a mais banco de dados em relação ao Laravel, conforme a figura 15:

Figura 15 – Bancos de dados de cada *framework*

	Laravel	CakePHP
Banco de dados suportados	MySQL PostgreSQL SQLite SQL Server	MySQL PostgreSQL SQLite SQL Server Oracle

Fonte: Do autor.

A segunda análise feita, foi na configuração inicial de cada *framework,* ou seja, os requisitos mínimos necessários para cada um deles funcionar, segundo suas respectivas documentações. Foi analisado primeiramente o *framework* CakePHP.

Requisitos para funcionamento:

a) Um servidor. Por exemplo: Apache
b) mod_rewite ativo (opcional)
c) PHP 5.6.0 ou superior
d) Extensão mbstring ativa
e) Extensão intl ativa

Requisitos para instalação:

a) Composer

Permissões:

a) Permissão total nos diretórios *logs* e *tmp*.

Logo após, foi analisado o *framework* Laravel.

Requisitos para funcionamento:

a) Um servidor. Por exemplo: Apache

b) mod_rewite

c) PHP 7.1.3 ou superior

d) Extensão mbstring ativa

e) Extensão OpenSSL ativa

f) Extensão PDO ativa

g) Extensão Tokenizer ativa

h) Extensão XML ativa

i) Extensão Ctype ativa

j) Extensão JSON ativa

Requisitos para instalação:

a) Composer

Permissões:

a) Permissões totais nos diretórios storage e bootstrap/cache.

Como se pode observar com os resultados referentes a configuração inicial de cada *framework*, o CakePHP exige muito menos requisitos para funcionamento em relação ao Laravel, porém os requisitos para a instalação e permissões de diretórios, os dois *frameworks* têm as mesmas exigências.

Na parte referente as requisições no servidor, foram feitos 10 testes com cada *framework,* utilizando o comando: ab -n 1000 -c 10 http://localhost/nome-da-aplicação/, onde são simuladas 1000 requisições com 10 usuários simultâneos. Ou seja, 100 requisições por usuário. Com isso foi analisado o número de requisições completadas com sucesso, a quantidade de requisições por segundo e também o tempo em cada requisição. Nessa parte o *framework* Laravel obteve um desempenho muito acima do CakePHP, (tabelas 3 e 4).

Tabela 3 – Testes de requisições com o framework CakePHP

Testes	Requisições Completas	Requisições/segundos	T./Requisição
Teste 1	1000	1,76	5673,138ms = 5,673138s
Teste 2	1000	1,73	5793,008ms = 5,793008s
Teste 3	1000	1,76	5672,927ms = 5,672927s
Teste 4	1000	1,75	5710,840ms = 5,710840s
Teste 5	1000	1,69	5919,568ms = 5,919568s
Teste 6	1000	1,70	5893,296ms = 5,893269s
Teste 7	1000	1,71	5849,740ms = 5,849740s
Teste 8	1000	1,71	5833,775ms = 5,833775s
Teste 9	1000	1,72	5818,487ms = 5,818487s
Teste 10	1000	1,71	5849,366ms = 5,849366s

Fonte: Do autor.

Desvio padrão: 0,025033311

Tabela 4 – Testes de requisições com o framework Laravel

Testes	Requisições Completas	Requisições/segundos	T./Requisição
Teste 1	1000	29,86	334,892ms = 0,334892s
Teste 2	1000	28,28	353,584ms = 0,353584s
Teste 3	1000	29,91	334,356ms = 0,334356s
Teste 4	1000	30,29	330,117ms = 0,330117s
Teste 5	1000	28,06	356,381ms = 0,356381s
Teste 6	1000	27,56	362,789ms = 0,362789s
Teste 7	1000	27,95	357,804ms = 0,357804s
Teste 8	1000	27,66	361,589ms = 0,361589s
Teste 9	1000	27,61	362,223ms = 0,362223s
Teste 10	1000	27,64	361,841ms = 0,361841s

Fonte: Do autor.

Desvio padrão: 1,090563565

A análise seguinte, foi referente as transações do banco de dados, como INSERT, SELECT, UPDATE e DELETE, respectivamente nessa ordem e também o uso de memória da aplicação em cada um dos testes. Foram feitos também 10 testes para cada função e verificado o tempo de execução em segundos de cada uma delas, e verificado o uso de memória da aplicação em MegaBytes (MB) (tabelas 5 e 6).

SQLs gerados pelo Cakephp:

SELECT:

```
SELECT
  Medias.ano AS `Medias__ano`,
  Medias.media AS `Medias__media`
FROM
  medias Medias;
```

```
SELECT
  Parametros.id AS `Parametros__id`,
  Parametros.alfa0 AS `Parametros__alfa0`,
  Parametros.alfa1 AS `Parametros__alfa1`,
  Parametros.alfa2 AS `Parametros__alfa2`,
  Parametros.alfa3 AS `Parametros__alfa3`,
  Parametros.l1 AS `Parametros__l1`,
  Parametros.l2 AS `Parametros__l2`,
  Parametros.l3 AS `Parametros__l3`,
  Parametros.l4 AS `Parametros__l4`,
  Parametros.t3 AS `Parametros__t3`,
  Parametros.t4 AS `Parametros__t4`,
  Parametros.xi AS `Parametros__xi`,
  Parametros.alfa AS `Parametros__alfa`,
  Parametros.k AS `Parametros__k`,
  Parametros.h AS `Parametros__h`,
  Parametros.created AS `Parametros__created`,
  Parametros.modified AS `Parametros__modified`
FROM
  parametros Parametros;
```

INSERT:

```
INSERT INTO medias (ano, media, created, modified)
VALUES ('1987', 176.2, '2018-12-08 13:51:28', '2018-12-08 13:51:28');
```

UPDATE:

```
UPDATE
  parametros
SET
  xi = 178.46469766409,
  alfa = 75.085958516599,
  k = 0.059075842354864,
```

```
  h = -0.3351799295671,
  modified = '2018-12-08 13:52:38'
WHERE
  id = 1;
```

DELETE:

```
DELETE FROM medias;
```

Tabela 5 – Testes de transações do banco de dados com o framework CakePHP

Teste	INSERT	SELECT	UPDATE	DELETE	Memória Usada
Teste 1	0,019s	0,001s	0,016s	0,088s	8,39MB
Teste 2	0,001s	0,001s	0,001s	0,031s	8,39MB
Teste 3	0,003s	0,001s	0,014s	0,079s	8,39MB
Teste 4	0,001s	0,001s	0,025s	0,117s	8,40MB
Teste 5	0,001s	0,001s	0,019s	0,032s	8,40MB
Teste 6	0,035s	0,001s	0,016s	0,125s	8,41MB
Teste 7	0,001s	0,001s	0,001s	0,056s	8,41MB
Teste 8	0,003s	0,001s	0,024s	0,056s	8,41MB
Teste 9	0,001s	0,001s	0,013s	0,089s	8,42MB
Teste 10	0,001s	0,001s	0,007s	0,166s	8,40MB

Fonte: Do autor.

SQLs gerados pelo Laravel;

SELECT:

```
select * from `medias`;
select * from `parametros`;
```

INSERT:

```
insert into `medias` (`ano`, `media`, `updated_at`, `created_at`)
values ('1987', '176.2', '2018-12-08 14:02:45', '2018-12-08 14:02:45');
```

UPDATE:

```
update `parametros`
set `xi` = '178.46469766409',
   `alfa` = '75.085958516599',
   `k` = '0.059075842354864',
   `h` = '-0.3351799295671',
```

```
    `updated_at` = '2018-12-08 14:03:30'
where `id` = '3';
DELETE:
delete from medias;
```

Tabela 6 – Testes de transações do banco de dados com o framework Laravel

Testes	INSERT	SELECT	UPDATE	DELETE	Memória Usada
Teste 1	0,052s	0,610s	0,040s	0,067s	9,76MB
Teste 2	0,041s	0,700s	0,041s	0,048s	9,77MB
Teste 3	0,055s	0,680s	0,068s	0,051s	9,76MB
Teste 4	0,056s	0,700s	0,070s	0,063s	9,71MB
Teste 5	0,087s	0,680s	0,072s	0,037s	9,71MB
Teste 6	0,104s	0,600s	0,118s	0,027s	9,68MB
Teste 7	0,286s	0,680s	0,068s	0,052s	9,76MB
Teste 8	0,043s	0,700s	0,059s	0,052s	9,76MB
Teste 9	0,043s	0,590s	0,059s	0,052s	9,71MB
Teste 10	0,039s	0,590s	0,058s	0,059s	9,71MB

Fonte: Do autor.

Como se pode ver nas tabelas com os resultados dos testes, o *framework* CakePHP teve um desempenho melhor que o Laravel nas funções INSERT, SELECT e UPDATE e também teve um desempenho melhor em relação ao uso de memória da aplicação, já que em todos os testes ele teve uma média de 8.40MB de memória utilizada, enquanto que o Laravel teve uma média de 9.73MB de memória utilizada. Já na função DELETE o Laravel teve um desempenho melhor que o CakePHP.

Em relação aos materiais existentes na literatura sobre assuntos relativos a este tema, esta pesquisa se diferencia dos demais materiais pelo fato de analisar apenas estes dois *frameworks*, utilizando metodologias diferentes dos trabalhos existentes. Sendo que os trabalhos pesquisados como referência para esta pesquisa, geralmente analisavam mais de dois *frameworks* PHP, onde o *framework* Laravel, por exemplo, quase nunca se fazia presente, ou então eram analisados *frameworks* da linguagem Java, como na pesquisa de Franco (2011), que realizou um estudo comparativo entre *frameworks* Java para desenvolvimento de aplicações *web*: JSF 2.0, Grails e Spring *web* MVC, ou utilizavam ferramentas diferentes para as analises, como por exemplo no trabalho de Bagestan (2016), que faz um estudo comparativo de *frameworks* PHP, com enfoque no CodeIgniter, e utilizou uma ferramenta chamada *PHPMetrics,* para realização das análises, porém não foram

analisados o tempo de resposta nas requisições, quantidades de requisições por segundo, tempo em cada requisição e também não foram analisadas as transações com o banco de dados. Nesta pesquisa estes testes estão presentes.

No trabalho de Lacerda (2013), pode-se observar que foi feita uma análise entre frameworks PHP, Java e .NET. Porém análises muito mais completas, onde foram feitos vários testes como teste de leitura de registros, teste de construção de uma estrutura map, teste de leitura e inserção de registros, teste de leitura de ficheiros, teste de criação de lista, teste de criação de tabela em HTML. Alguns destes testes não estão presentes nesta pesquisa, como por exemplo, o teste de criação de tabela em HTML.

Na pesquisa de HENNING (2013), onde foi feita uma análise de *frameworks* para o desenvolvimento de sistema de controle sindical, foram analisados quatro *frameworks* PHP e foram feitas análises referentes à licença, facilidades de aprendizagem, complexidades de instalação, segurança, agilidades no desenvolvimento, performance, impressão de relatórios e suporte e documentação. Algumas destas analises estão também presentes nesta pesquisa, como a complexidade de instalação, onde nesta pesquisa se refere as configurações iniciais. Porém não foram vistas as análises referentes às transações com o banco de dados, tempo de resposta nas requisições, quantidades de requisições por segundo, tempo em cada requisição e também não se observou os bancos de dados suportados por cada *framework.*

8 CONCLUSÃO

Nesta pesquisa, foi realizada a comparação entre os *frameworks* de PHP, Laravel e CakePHP, onde foram comparados em relação ao número de bancos de dados suportados por cada *framework*, configuração inicial de cada um deles, o número de requisições por segundo de ambos, o tempo de cada requisição, número de requisições completadas com sucesso, transações com o banco de dados (INSERT, SELECT, UPDATE e DELETE) e consumo de memória de cada *framework* em ambas as aplicações.

Para a realização das comparações, foram desenvolvidas duas aplicações, uma para cada *framework*, com as mesmas funcionalidades, interface gráfica e objetivo, onde cada uma delas teve a distribuição Kappa de quatro parâmetros implementada.

Na parte da implementação do algoritmo para resolver a distribuição Kappa de quatro parâmetros, pode-se notar que é um algoritmo bastante complexo e extenso. Este algoritmo havia sido implementado anteriormente apenas nas linguagens Fortran e R, sendo, portanto transposto e implementado na linguagem PHP.

Na parte da análise do número de bancos de dados suportados, observou-se que o CakePHP suporta um banco de dados a mais que o Laravel.

Em relação a configuração inicial mínima para cada *framework*, o CakePHP foi melhor que o Laravel na parte de requisitos para funcionar, onde exige menos requisitos, porém os requisitos para a instalação e permissões de diretórios, os dois *frameworks* têm as mesmas exigências.

Na análise referente ao número de requisições por segundo e tempo de cada requisição, o Laravel se mostrou melhor que o CakePHP, porém no número de requisições completadas, ambos tiveram o mesmo resultado.

Referente as transações com o banco de dados, o CakePHP teve um desempenho melhor nas funções INSERT, SELECT e UPDATE, já na função DELETE o Laravel obteve um melhor desempenho.

E por fim, nos testes de consumo de memória o CakePHP obteve um desempenho melhor em todos os testes, consumindo cerca de 1.33MB de memória a menos que o Laravel.

Nos cenários criados para esta pesquisa, o *framework* CakePHP mostrou-se melhor na maioria das análises realizadas.

Os objetivos gerais e específicos foram atingidos com sucesso, e os resultados documentados podem servir de auxílio no processo de escolha entre os dois *frameworks* de PHP.

Para a realização de trabalhos futuros, sugere-se:

a) Comparação dos *frameworks* Laravel e CakePHP, com outros *frameworks* de PHP existentes no mercado.
b) Comparação de *frameworks* PHP com *frameworks* para Java Web.
c) Comparação entre os *frameworks* Laravel e CakePHP para o desenvolvimento de APIs. Qual tem o melhor desempenho quando utilizado como API para aplicativos mobiles ou aplicativos *web.*

REFERÊNCIAS

ARAUJO SILVA, Gustavo Ferreira de. **Simulador de Análise de Desempenho para Banco de Dados MySQL.** Itajaí, 2011. 82 f. Trabalho de Conclusão de Curso (Graduação em Ciência da Computação) – Centro de Ciências Tecnológicas da Terra e do Mar, Universidade do Vale de Itajaí, Itajaí, 2011.

BAGESTAN, Flávio Henkes. **Estudo comparativo de frameworks PHP, com enfoque no Codeigniter**. 2016. 57 p. Monografia (Tecnólogo em Sistemas para Internet), Instituto Federal Sul-rio-grandense, Campus Passo Fundo, Passo Fundo, 2016.

BATTISTI, Iara Denise Endruweit; BATTISTI, Gerson. **Métodos estatísticos.** Ijuí: Unijuí, 2008.

CAKE SOFTWARE FOUNDATION. CakePHP Cookbook Documentation. 2018.

CAKE SOFTWARE FOUNDATION. CakePHP at a Glance. Tradução do acadêmico. Disponível em: <https://book.cakephp.org/3.0/en/intro.html>. Acesso em: 21 ago. 2018.

CENTENARO, Jonas. **Desenvolvimento de um software web para gerenciamento de requisitos de software**. 2014. 100 f. Trabalho de Conclusão de Curso (Especialização) – Especialista em Desenvolvimento de Sistemas para Internet e Dispositivos Móveis, Universidade Tecnológica Federal do Paraná, Paraná, 2014.

DANTAS, C. E. O. ; PINTO, E. J. A. **Análise de Freqüência Regional das Precipitações Máximas Diárias da Bacia do rio São Francisco no Estado de Minas Gerais - Brasil.** In: XIV IWRA Word Water Congress, 2011, Porto de Galinhas. XIV IWRA Word Water Congress: Livro de Resumos, 2011.

DIAS, Carlos Henrique Gonçalves; AMORIM, Wisney Almeida; CARVALHO JÚNIOR, Nelson Ribeiro de. **Desenvolvimento de Aplicações Web e Dispositivos Móveis Utilizando Frameworks**. Revista Pensar Tecnologia, Belo Horizonte, v. 4, n. 1, p.1-18, jan. 2015.

FRANCO, Rebeca S. T. **Estudo Comparativo entre Frameworks Java para Desenvolvimento de Aplicações Web: JSF 2.0, Grails e Spring Web MVC**. 2011. Monografia (Especialização em Tecnologia Java) – Programa de Pós-Graduação em Tecnologia, Universidade Tecnológica Federal do Paraná. Curitiba, 2011.

Hosking. J. R. M. and Wallis, J.R. (1993). **Some statistics useful in regional fiequency analysis**. *Water Resources Research 29* (2), 271-281.

Hosking, J. R. M. (1994). **The four-parameter kappa distribution.** IBM J. Res. Deve Eop- 38 (3), 25 1-258.

LACERDA, José António Vieira. **Análise de frameworks de desenvolvimento web para a conversão de uma aplicação desktop**. 2013. 188 f. Dissertação (Mestrado) - Curso de Tecnologia e Gestão de Sistemas de Informação, Escola Superior de Tecnologia e Gestão, Instituição Politécnico de Viana do Castelo, Viana do Castelo, 2013.

MACAIA, César Bráulio Sumbo**. Análise de Técnicas e Frameworks de Gamificação: Uma proposta de uso na educação**. 2016. 102 f. TCC (Graduação) - Curso de Ciência da Computação, Curso de Ciência da Computação, Universidade do Extremo Sul Catarinense - Unesc, Criciúma, 2016.

MARCOS, Caio Campos. **Framework para gerenciamento de qualidade e melhoria contínua de serviço de TI baseado em melhores práticas**. 2014. 113 f. TCC (Graduação) - Curso de Ciência da Computação, Curso de Ciência da Computação, Universidade do Extremo Sul Catarinense - Unesc, Criciúma, 2014.

MARQUES, Juliano. **Comparação de desempenho e consumo de memória entre frameworks de mapeamento objeto-relacional Java Hibernate e Eclipslink**. 2012. 79 f. TCC (Graduação) - Curso de Ciência da Computação, Curso de Ciência da Computação, Universidade do Extremo Sul Catarinense - Unesc, Criciúma, 2012.

MATIOLI, Danilo Cavassini. **Importância da segurança em banco de dados**. FEMA – Fundação Educacionaldo Munícipio de Assis, Cândido Mota – 2010.

MAZUCHETTI, Muriel Rampinelli. **Estudo de framework Multiplataforma aplicado ao desenvolvimento de um protótipo de aplicativo mobile híbrido para o controle financeiro pessoal**. 2015. 79 f. TCC (Graduação) - Curso de Ciência da Computação, Curso de Ciência da Computação, Universidade do Extremo Sul Catarinense - Unesc, Criciúma, 2015.

MILANI, André. *Construindo Aplicações Web com PHP e MySQL*. Novatec Editora Ltda., 2010.

MINETTO, Elton Luís. **Frameworks para Desenvolvimento em PHP**. 1. ed. [S.l.]: Novatec, 2007. 192 p. Disponível em:<http://www.martinsfontespaulista.com.br/anexos/produtos/capitulos/243418.pdf. Acesso em: 06 out. 2018.

MySQL. Disponível em: <https://www.infoescola.com/informatica/mysql/>. Acesso em: 10 out. 2018.

NASCIMENTO, Eduardo Simões. **Modelagem e Protótipo de um Sistema para Gerenciamento de Reuniões/Eduardo Simões Nascimento**. -- Formiga: IFMG, 2017. 101p.: il

PANSERA, Wagner Alessandro. **Distribuição generalizada de chuvas máximas no Estado do Paraná**. Cascavél, PR: UNIOESTE, 2013. 94 f.; 30 cm.

PHP 7 é lançado após anos de desenvolvimento. Disponível em: <https://imasters.com.br/noticia/php-7-e-lancado-apos-anos-de-desenvolvimento>. Acesso em: 04 out. 2018.

PHP 7 e novidades do PHP 7.1. Disponível em: < https://www.treinaweb.com.br/blog/php-7-e-novidades-do-php-7-1/>. Acesso em: 04 out. 2018.

PHP 7.2: quais são as novidades da nova versão do PHP. Disponível em: < https://imasters.com.br/back-end/php-7-2-quais-sao-as-novidades-da-nova-versao-do-php>. Acesso em: 04 out. 2018.

PISCKE, Leandro Salvatti. **Framework em Java para geração de telas no modelo CRUD Baseado em XML e objetos remotos utilizando a arquitetura MVC e padrões**. 2007. 68 f. TCC (Graduação) - Curso de Ciência da Computação, Curso de Ciência da Computação, Universidade Regional de Blumenau, Blumenal, 2007.

REBECCHI, Miguel de Melo. **Atendimento de TI a Micro e Pequenas Empresas:** Motivos Pelos Quais Os Fornecedores de Soluções de TI Não Conseguem se Estabelecer entre as MPEs e Soluções. 2012. 110 f. Monografia (Especialização) - Curso de Tecnologia em Processamento de Dados, Departamento de Tecnologia da Informação (dti), Faculdade de Tecnologia de São Paulo, São Paulo, 2012.

SÃO PAULO, **Apostila de MySQL**, pág. 49, São Paulo, São Paulo, 2001.

SILVA, Matheus Rodrigues Rosado da. **Projeto e desenvolvimento de um sistema para gerenciamento de trabalhos de conclusão de curso**. 2017. 37 f. TCC (Graduação) - Curso de Sistemas de Informação, Faculdade de Ciência da Computação, Universidade Federal de Uberlândia, Uberlândia, 2017.

SILVA, E. Patriky. **Utilização do framework CakePHP para desenvolvimento de Websites em PHP**. 2011. Trabalho de conclusão de curso (Tecnologia em Análise e Desenvolvimento de Sistemas), Universidade Tecnológica Federal do Paraná. Medianeira 2011.

SODRE, Eduardo. **Desenvolvimento do Framework Java – Fácil.** Fundação Educacional do Município de Assis – FEMA – Assis, 2011. 44p.

WERNER, Orlando Vitali. **Estudo comparativo entre banco de dados relacionais, nosql e newsql**. 2014. 81 f. TCC (Graduação) - Curso de Ciência da Computação, Universidade do Extremo Sul Catarinense, Unesc, Criciúma, 2014.

APÊNDICE(S)

APÊNDICE A – Algoritmo da função Kappa de Quatro Parâmetros

```
public function kappa() {
    $parametros = Parametro::all();
    foreach ($parametros as $parametro) {
        $id = $parametro->id;
        $XMOM = [
            1 => $parametro->l1,
            2 => $parametro->l2,
            3 => $parametro->t3,
            4 => $parametro->t4
        ];
    }

    $EPS = 0.000001;
    $MaxIt = 20;
    $MaxSR = 10;
    $Hstart = 1.001;
    $big = 10;
    $oflexp = 170;
    $oflgam = 53;
    $T3 = $XMOM[3];
    $T4 = $XMOM[4];
    $CODERRO = 10;

    $PARA = [
        1 => 0,
        2 => 0,
        3 => 0,
        4 => 0
    ];

    if($XMOM[2] <= 0) {
        $CODERRO = 1;
```

```
            Session::flash('error', 'L-MOMENTS inválidos');
            return redirect('/');
        }

        if(abs($T3) >= 1 || abs($T4) >= 1) {
            $CODERRO = 1;
            Session::flash('error', 'L-MOMENTS inválidos');
            return redirect('/');
        }

        if($T4 <= (5 * $T3 * $T3 - 1) / 4 ) {
            $CODERRO = 1;
            Session::flash('error', 'L-MOMENTS inválidos');
            return redirect('/');
        }

        if( $T4 >= (5 * $T3 * $T3 + 1) / 5 ) {
            $CODERRO = 2;
            Session::flash('error', 'T3 e T4 encontram-se acima da linha generalizada-logística.');
            return redirect('/');
        }

        $G = (1 - 3 * $T3) / (1 + $T3);
        $H = $Hstart;
        $Z = $G + $H * 0.725;
        $Xdist = $big;
        $IT = 1;
        $I = 1;

        //Inicio do Método Newtonw -Raphson
        if($CODERRO == 10) {
            for($IT; $IT <= $MaxIt; $IT++) {
                for($I = 1; $I <= $MaxSR; $I++) {
```

```
            if($G > $oflgam) {
               $CODERRO = 5;
            }

            if($G <= $oflgam) {
               if ($H >= 0) {
                  $U1 = Special::gamma(1 / $H) - Special::gamma(1 / $H + 1 + $G);
                  $U1 = exp(log(Special::gamma(1 / $H)) - log(Special::gamma(1 /
$H + 1 + $G)));
                  $U2 = exp(log(Special::gamma(2 / $H)) - log(Special::gamma(2 /
$H + 1 + $G)));
                  $U3 = exp(log(Special::gamma(3 / $H)) - log(Special::gamma(3 /
$H + 1 + $G)));
                  $U4 = exp(log(Special::gamma(4 / $H)) - log(Special::gamma(4 /
$H + 1 + $G)));
               }
               else {
                  $U1 = exp(log(Special::gamma(-1 / $H - $G)) -
log(Special::gamma(-1 / $H + 1)));
                  $U2 = exp(log(Special::gamma(-2 / $H - $G)) -
log(Special::gamma(-2 / $H + 1)));
                  $U3 = exp(log(Special::gamma(-3 / $H - $G)) -
log(Special::gamma(-3 / $H + 1)));
                  $U4 = exp(log(Special::gamma(-4 / $H - $G)) -
log(Special::gamma(-4 / $H + 1)));
               }
            }

            $Alam2 = $U1 - 2 * $U2;
            $Alam3 = -$U1 + 6 * $U2 - 6 * $U3;
            $Alam4 = $U1 - 12 * $U2 + 30 * $U3 - 20 * $U4;

            if($Alam2 == 0) {
               $CODERRO = 5;
```

```
}

if($Alam2 != 0) {
   $Tau3 = $Alam3 / $Alam2;
   $Tau4 = $Alam4 / $Alam2;
   $E1 = $Tau3 - $T3;
   $E2 = $Tau4 - $T4;

   if(abs($E1) > abs($E2)) {
      $Dist = abs($E1);
   }
   else {
      $Dist = abs($E2);
   }

   if($Dist < $Xdist) {

      if($Dist < $EPS) {
         //Achou a solução
         $CODERRO = 0;
      }

      if($Dist >= $EPS) {
         //Não convergiu
         $XG = $G;
         $XH = $H;
         $XZ = $Z;
         $Xdist = $Dist;
         $RHH = 1 / ($H * $H);

         if($H <= 0) {
            $U1g = -$U1 * Special::diGamma(-1 / $H - $G);
            $U2g = -$U2 * Special::diGamma(-2 / $H - $G);
            $U3g = -$U3 * Special::diGamma(-3 / $H - $G);
```

```
            $U4g = -$U4 * Special::diGamma(-4 / $H - $G);
            $U1h = $RHH * (-$U1g - $U1 * Special::diGamma(-1 / $H +
1));
            $U2h = 2 * $RHH * (-$U2g - $U2 * Special::diGamma(-2 / $H +
1));
            $U3h = 3 * $RHH * (-$U3g - $U3 * Special::diGamma(-3 / $H +
1));
            $U4h = 4 * $RHH * (-$U4g - $U4 * Special::diGamma(-4 / $H +
1));
        }
    }

    if($H > 0) {
        $U1g = -$U1 * Special::diGamma(1 / $H + 1 + $G);
        $U2g = -$U2 * Special::diGamma(2 / $H + 1 + $G);
        $U3g = -$U3 * Special::diGamma(3 / $H + 1 + $G);
        $U4g = -$U4 * Special::diGamma(4 / $H + 1 + $G);
        $U1h = $RHH * (-$U1g - $U1 * Special::diGamma(1 / $H));
        $U2h = 2 * $RHH * (-$U2g - $U2 * Special::diGamma(2 / $H));
        $U3h = 3 * $RHH * (-$U3g - $U3 * Special::diGamma(3 / $H));
        $U4h = 4 * $RHH * (-$U4g - $U4 * Special::diGamma(4 / $H));
    }

    $DL2g = $U1g - 2 * $U2g;
    $DL2h = $U1h - 2 * $U2h;
    $DL3g = -$U1g + 6 * $U2g - 6 * $U3g;
    $DL3h = -$U1h + 6 * $U2h - 6 * $U3h;
    $Dl4g = $U1g - 12 * $U2g + 30 * $U3g - 20 * $U4g;
    $Dl4h = $U1h - 12 * $U2h + 30 * $U3h - 20 * $U4h;
    $D11 = ($DL3g - $Tau3 * $DL2g) / $Alam2;
    $D12 = ($DL3h - $Tau3 * $DL2h) / $Alam2;
    $D21 = ($Dl4g - $Tau4 * $DL2g) / $Alam2;
    $D22 = ($Dl4h - $Tau4 * $DL2h) / $Alam2;
    $Det = $D11 * $D22 - $D12 * $D21;
```

```
$H11 = $D22 / $Det;
$H12 = -$D12 / $Det;
$H21 = -$D21 / $Det;
$H22 = $D11 / $Det;
$Del1 = $E1 * $H11 + $E2 * $H12;
$Del2 = $E1 * $H21 + $E2 * $H22;
$G = $XG - $Del1;
$H = $XH - $Del2;
$Z = $G + $H * 0.725;
$Factor = 1;

if($G <= -1) {
   $Factor = 0.8 * ($XG + 1) / $Del1;
}

if($H <= -1) {
   $va = 0.8 * ($XH + 1) / $Del2;
   if($Factor < $va) {
      $Factor = $va;
   }
}

if($Z <= -1) {
   $va = 0.8 * ($XZ + 1) / ($XZ - $Z);
   if($Factor < $va) {
      $Factor = $va;
   }
}

if(($H <= 0) && ($G * $H <= -1)) {
   $va = 0.8 * ($XG * $XH + 1) / ($XG * $XH - $G * $H);

   if($Factor < $va) {
      $Factor = $va;
```

```
                }
            }

            if($Factor != 1) {
                $Del1 = $Del1 * $Factor;
                $Del2 = $Del2 * $Factor;
                $G = $XG - $Del1;
                $H = $XH - $Del2;
                $Z = $G + $H * 0.725;
            }
        }
        if($Dist >= $Xdist) {
            $Del1 = 0.5 * $Del1;
            $Del2 = 0.5 * $Del2;
            $G = $XG - $Del1;
            $H = $XH - $Del2;
        }
    }
}

if($CODERRO < 10) {
    $IT = $MaxIt;
}
}
}

if($CODERRO == 0) {
    $PARA[3] = $G;
    $PARA[4] = $H;

    $temp = log(Special::gamma(1 + $G));

    if($temp > $oflexp) {
        $CODERRO = 6;
```

```
        }

        if($temp <= $oflexp) {
            $gam = Special::gamma(1 + $G);
            $temp = (1 + $G) * log(abs($H));

            if($temp > $oflexp) {
                $CODERRO = 6;
            }

            if($temp <= $oflexp) {
                $HH = exp($temp);
                $PARA[2] = $XMOM[2] * $G * $HH / ($Alam2 * $gam);
                $PARA[1] = $XMOM[1] - $PARA[2] / $G * (1 - $gam * $U1 / $HH);
            }
        }

        $parametrosDados = Parametro::find($id);

        $parametrosDados->xi = $PARA[1];
        $parametrosDados->alfa = $PARA[2];
        $parametrosDados->k = $PARA[3];
        $parametrosDados->h = $PARA[4];

        $parametrosDados->save();

        Session::flash('success', 'Parâmentros calculados com sucesso.');
        return redirect('/');
    }
    else {
        Session::flash('error', 'Não foi possível calcular os parâmentros.');
        return redirect('/');
    }
}
```

APÊNDICE B – ARTIGO

Análise comparativa dos *frameworks* Laravel e CakePHP na implementação da distribuição Kappa em uma aplicação web.

Maurício Citadini Biléssimo[1], Luciano Antunes[2]

[1]Acadêmico do Curso de Ciência da Computação – Universidade do Extremo Sul Catarinense (UNESC) – Criciúma - SC

[2]Professor do Curso de Ciência da Computação – Universidade do Extremo Sul Catarinense (UNESC) – Criciúma - SC

mauricioc.bilessimo@gmail.com, Luciano@unesc.net

Abstract. *When the process of developing new software is started, one of the issues to think about is how to structure it, so that your code is understandable and easy to maintain. In order to assist in this process, the frameworks provide a base structure, which is a standard during application development. Considering the number of PHP frameworks on the market, this work aims to compare the Laravel and CakePHP frameworks in the development of a web application, which will have the Kappa distribution implemented. Two identical applications were created, one with the Laravel framework and another with the CakePHP framework and both have the Kappa distribution implemented. The analyzes were done by consulting the documentation, each frameworks own tools, and also the command line tool, ApacheBench Apache server. The Kappa distribution was successfully implemented and the results show that the CakePHP framework has been better in most analyzes. For exemple the analysis of the number of databases supported, where CakePHP supports five databases , while Laravel supports only four, or also in the database transactions, where CakePHP can perform an insert in an average of 0.007 seconds, while Laravel performs an insert in an average of 0.081 seconds.*

Resumo. *Quando o processo de desenvolvimento de um novo software é iniciado, uma das questões a se pensar é como estruturá-lo, de forma que seu código seja compreensível e de fácil manutenção. Com o objetivo de ajudar neste processo, os frameworks fornecem uma estrutura base, a qual é seguida como padrão durante o desenvolvimento da aplicação. Tendo em vista a quantidade de frameworks PHP existentes no mercado, este trabalho tem como objetivo comparar os frameworks Laravel e CakePHP no desenvolvimento de uma aplicação web, que terá a distribuição Kappa implementada. Foram criadas duas aplicações idênticas, uma com o framework Laravel e outra com o framework CakePHP, onde ambas têm a distribuição Kappa implementada. As análises foram feitas através de consulta da documentação, ferramentas nativas de cada framework, e também a ferramenta de linha de comando, ApacheBench do servidor Apache. A distribuição Kappa foi implementada com sucesso e os resultados obtidos demonstram que o framework CakePHP mostrou-se melhor na maioria das análises realizadas, como por exemplo na análise referente ao número de bancos de dados suportados, onde o CakePHP tem suporte a cinco bancos de dados, enquanto o Laravel tem suporte a apenas*

quatro, ou também na parte das transações no banco de dados, onde o CakePHP consegue executar um insert numa media de 0,007 segundos, enquanto o Laravel executa um insert numa media de 0,081 segundos.

1. Introdução

O setor de Tecnologia da Informação (TI) se estabeleceu completa e definitivamente em diversas áreas e segmentos (REBECCHI, 2012).

As aplicações do tipo *web* são exemplos da informatização, já que cada vez mais tarefas, desde o envio de mensagens até o pagamento de contas, podem ser realizadas de maneira online, utilizando apenas um computador com acesso à Internet. O que antigamente não era muito aceito pela população por ser considerado perigoso, hoje em dia se tornou frequente na vida da maioria das pessoas, que utilizam essas aplicações para tarefas comuns do dia-a-dia (DA SILVA, 2017).

Quando o processo de desenvolvimento de um novo *software* é iniciado, uma das grandes questões a se pensar é como estruturá-lo, de forma que seu código seja compreensível e de fácil manutenção. Com o objetivo de ajudar neste processo, os *frameworks* fornecem uma estrutura base, a qual é seguida como padrão durante o desenvolvimento da aplicação (BAGESTAN, 2016). Segundo Dias, Amorim e Carvalho Júnior (2015), os *frameworks* são considerados como caixas de ferramentas que auxiliam, agilizam e facilitam o desenvolvimento de *softwares.* A *World Wide Web Consortium* (W3C) definiu padrões para a criação e desenvolvimento de aplicações *web* que facilitam e normalizam a sua utilização. Quando um projeto *web* é definido e segue esses padrões ele é capaz de beneficiar não só o usuário que o utiliza, mas também todos que acompanham este *software*, o mesmo sendo compatível com as diversas versões e tipos de navegadores *web.*

Entretanto, como há uma grande variedade de *frameworks* disponíveis para a linguagem de programação *HyperText Preprocessor* (PHP), o momento de decidir qual *framework* utilizar em uma aplicação pode gerar uma série de dúvidas. Isto se deve a quantidade de *frameworks* existentes, cada um com suas particularidades, características positivas e negativas (BAGESTAN, 2016). Aplicações mal arquitetadas e sem uso de padrões no desenvolvimento podem acarretar vários problemas na solução final (PISCKE, 2007).

Tendo em vista a quantidade de *frameworks* existentes, este trabalho irá falar um pouco sobre o Laravel e o CakePHP, os quais serão utilizados nesse projeto e também implementar a distribuição Kappa de quatro parâmetros nas aplicações desenvolvidas. O Laravel utiliza a arquitetura *Model, View, Controller* (MVC), entre suas características, está o desenvolvimento de aplicações seguras, de modo rápido, com código simples e limpo. Quando se trata de interfaces gráficas é utilizado uma *engine* de *template* chamada *Blade,* que propicia uma série de ferramentas que contribuem para a criação de interfaces elegantes e funcionais de forma rápida, evitando a duplicação de códigos. Quanto a comunicação com o banco de dados, é utilizada uma implementação chamada *Eloquent*, que traz funcionalidades que facilitam a inserção, atualização, busca e exclusão de registros (NASCIMENTO, 2017). O CakePHP também utiliza a arquitetura MVC. Segundo Silva (2011), é reconhecido pela simplicidade e pela alta produtividade que ele proporciona. Baseado no framework Rails, ele funciona através de um gerador de código que produz todo o *create*, *read*, *update*, *delete* (CRUD) deixando para o programador desenvolver a regra de negócio de cada sistema. Utiliza como roteamento o padrão *Representational State Transfer* (RESTfull*)*, onde dentro dessa arquitetura são utilizados os métodos *get*, *post*, *put* e *delete*, que servem para manipular os recursos. As URLs seguem um padrão que às divide em quatro partes: dominio/controlador/ação/parâmetro. Projetado para tornar as tarefas comuns de desenvolvimento web simples e fáceis, fornece uma caixa de ferramentas *all-in-one* para você começar (CAKE SOFTWARE FOUNDATION, 2018).

A distribuição Kappa de quatro parâmetros é definida pelos parâmetros x, alfa, k e h, e inclui, como casos particulares, as distribuições Logística, Generalizada de Valores Extremos e Generalizada de Pareto (DANTAS; PINTO, 2011).

Hosking (1994), introduziu a distribuição Kappa de quatro parâmetros como generalização de diversas distribuições comumente usadas na análise de cheias. Dentre as diversas distribuições consideradas como casos especiais da distribuição Kappa, destacam-se: GEV, GLO e GLPA (PANSERA, 2013).

Diante do contexto apresentado anteriormente, o seguinte trabalho propõe uma análise comparativa em desempenho, velocidade, tempo de resposta, entre outros; entre os *frameworks* Laravel e CakePHP no desenvolvimento de uma aplicação *web* que terá a distribuição Kappa implementada, afim de saber qual dos dois *frameworks* atende de melhor maneira no desenvolvimento de aplicações *web*.

2. Justificativa

As aplicações *web* têm vindo a ganhar uma preponderância muito grande, especialmente na área do software de gestão. Durante vários anos, este tipo de software apenas era disponibilizado aos utilizadores na plataforma desktop, mas esta tendência tem vindo a ser contrariada. Atualmente, os grandes produtores de software de gestão nacionais já disponibilizam aos seus clientes soluções de gestão totalmente baseadas na *web* (LACERDA, 2013).

Entretanto, mesmo com a constante evolução de métodos, técnicas e ferramentas, a entrega de software em prazos e custos estabelecidos e com qualidade mínima desejável nem sempre é alcançada. A consequência dessa situação é o desenvolvimento de códigos desnecessários, falta de padronização nos métodos e nos atributos, e também repetição de tarefas com a mesma finalidade na programação. Uma forma que as empresas podem trabalhar para conseguir driblar essa situação é usar ferramentas que as auxiliem automatizando e padronizando o projeto, juntamente com ferramentas geradoras de código. Um bom exemplo dessas ferramentas é o uso dos *frameworks* (SILVA, 2011).

O processo de criação de uma aplicação *web* pode revelar-se bastante complexo logo na fase de seleção da plataforma de desenvolvimento uma vez que existem várias opções disponíveis e diversos *frameworks* (LACERDA, 2013).

Dentre os fatores que levaram a escolha de se comparar os *frameworks* Laravel e CakePHP estão os fatos de serem *frameworks* robustos, populares entre os desenvolvedores *web*, são ferramentas estabelecidas no mercado. Os dois *frameworks* baseiam-se na arquitetura MVC. O Laravel permite o desenvolvimento de aplicações seguras, de modo rápido, com código simples e limpo (NASCIMENTO, 2017). O CakePHP fornece uma base robusta e sólida para as suas aplicações, permitindo customizar e estender facilmente muitos aspectos. Também fornece uma convenção básica de nomes de arquivos e de tabelas do banco de dados, mantendo toda a aplicação consistente e lógica (Cake Software Foundation, 2018).

Outro motivo é também o auxílio que o comparativo trará no processo de escolha de um *framework*, visando os benefícios que o mesmo traz ao desenvolvedor e ao cliente. Tendo em vista que nem sempre o cliente terá uma boa infraestrutura, com um acesso à Internet razoavelmente bom, ou as vezes computadores mais antigos, com pouco desempenho, processamento, etc. Então a escolha de uma boa ferramenta e/ou *framework* para o desenvolvimento do produto final, podem suprir os pontos negativos citados.

3. Laravel

O Laravel é um *framework* PHP utilizado para o desenvolvimento *web*, que utiliza arquitetura MVC. Sua principal característica é o desenvolvimento de aplicações seguras e com boa performance de modo rápido, com código simples e limpo, já que o Laravel incentiva o uso

de boas práticas de programação. Quando se trata de interfaces gráficas é utilizado uma *engine* de *template* chamada Blade, que propicia uma série de ferramentas que contribuem para a criação de interfaces elegantes e funcionais de forma rápida, evitando a duplicação de código. Quanto à comunicação com o banco de dados é utilizada uma implementação chamada *Eloquent*, que traz funcionalidades que facilitam a inserção, atualização, busca e execução de registro. Logo, com uma configuração simples e pequena é possível se conectar ao banco de dados e trabalhar com o mesmo (NASCIMENTO, 2017).

4. CakePHP

A melhor definição para o CakePHP diz que ele permite que os usuários de PHP de todos os níveis possam desenvolver rapidamente aplicações *web* robustas. É reconhecido pela simplicidade e pela alta produtividade que ele proporciona. Baseado no *framework* Rails para Ruby, ele funciona através de um gerador de código que produz todo o CRUD, deixando para o programador desenvolver a regra de negócio de cada sistema (SILVA, 2011).

Fornece uma base robusta e sólida para as suas aplicações, permitindo customizar e estender facilmente muitos aspectos. Também fornece uma convenção básica de nomes de arquivos e de tabelas do banco de dados, mantendo toda a aplicação consistente e lógica (CAKE SOFTWARE FOUNDATION, 2018).

Assim como no Rails, o CakePHP utiliza como roteamento padrão, o RESTfull. O termo originou-se no ano de 2000, em uma tese de doutorado sobre a *web*, realizado pelo norte-americano Roy Fielding. REST é uma arquitetura cliente/servidor sem estado, no qual os recursos são identificados por suas URLs e são manipulados através de suas representações. Dentro de uma arquitetura REST, são utilizados os métodos *get*, *post*, *put* e *delete*, que servem para manipular os recursos. As URLs seguem um padrão que ás divide em quatro partes: www.dominio.com/controlador/ação/parâmetros (SILVA, 2011).

5. Kappa

A distribuição Kappa de quatro parâmetros introduzida por Hosking (1994), é definida pelos parâmetros x, alfa, *k* e *h,* e é considerada por várias distribuições estabelecidas incluindo a distribuição de *Generalized Extreme Value* (GEV), distribuição logística generalizada, distribuição generalizada de Pareto e distribuição de Gumbel que foram usadas para a modelagem de eventos extremos, sendo, portanto, teoricamente capaz de representar variáveis hidrológicas e hidrometeorológicas.

Onde x é o valor máximo anual, α e ε são os parâmetros de escala e posição respectivamente, k e h são os parâmetros de forma e F(x).

Muitos métodos de estimação de parâmetros foram propostos para ajustar a distribuição estatística aos dados hidrológicos. O método de estimativa atual usa funções lineares da estatística de ordem esperada, ou seja, *L-moment estimates* (LMEs). A estimativa dos LMEs fornece um método alternativo de estimação análogo aos momentos convencionais, mas com várias vantagens, incluindo estimativas mais robustas com menos viés (Hosking, 1994). LMEs, no entanto, desde que sejam computáveis, muitas vezes não são viáveis. Além disso, para a distribuição Kappa, as distribuições assintóticas dos LMEs não são facilmente tratáveis.

No entanto, o método de estimativa de LMEs de Kappa ainda requer iteração de Newton-Raphson. Além disso, o espaço de parâmetros é restrito para garantir a existência dos LMEs e a unicidade dos parâmetros. Portanto, a rotina fornecida por Hosking às vezes falha ao obter o método de estimativas de LMEs para alguns conjuntos de dados, porque não existe uma solução exclusiva dentro da região restrita.

6. Metodologia

Para a realização deste trabalho acadêmico foram empregadas as seguintes etapas metodológicas: levantamento bibliográfico; estudo sobre desenvolvimento web; estudo sobre a linguagem de programação PHP e seus respectivos *frameworks* Laravel e CakePHP; foram desenvolvidas duas aplicações utilizando-se a linguagem de programação PHP, ambas tem o mesmo objetivo, porém uma foi desenvolvida utilizando o *framework* Laravel e outra foi desenvolvida utilizando o *framework* CakePHP, onde em cada uma delas foi implementada uma função para resolver o método estatístico Kappa de quatro parâmetros; após prontas as aplicações, foram feitas as análises de cada uma delas com suas ferramentas nativas para a análise, Laravel Debugbar para o *framework* Laravel e DebugKit para o *framework* CakePHP; utilizou-se o *ApacheBench* (AB) para analisar as requisições por segundo de cada *framework* e o tempo médio de cada requisição, e foram também analisadas as documentações de cada um dos *frameworks* para obter as configurações iniciais de cada um deles e também o número de banco de dados que ambos suportam.

As análises comparativas foram feitas em ambiente *localhost*, utilizando um notebook com sistema operacional Windows 10, com 8 GB de memória RAM, processador Intel i5 de 6ª geração com 2.3 GHz e um HD de 500 GB, com o servidor apache instalado, o banco de dados utilizado foi o MySQL e a versão do PHP foi a 7.2.

Para a análise do tempo de resposta nas requisições média de requisições por segundo, e o desvio padrão foi utilizado o seguinte comando: ab -n 1000 -c 10 http://localhost/kappalaravel/, onde indica que foram feitas 1000 requisições com 10 usuários simultaneamente, ou seja, 100 requisições por usuário.

Para a parte das análises das transações com o banco de dados, como INSERT, SELECT, UPDATE e DELETE, tempo de carregamento da página e a memória que cada aplicação consome foi utilizado o Laravel Debugbar para o Laravel e para o CakePHP foi utilizado o DebugKit.

7. Resultados e discussão

Através da análise da documentação de cada *framework* foi possível obter as informações sobre a quantidade de banco de dados suportados de ambos, onde pode-se ver que o *framework* CakePHP tem suporte a mais banco de dados em relação ao Laravel, conforme figura 1.

	Laravel	CakePHP
Banco de dados suportados	MySQL PostgreSQL SQLite SQL Server	MySQL PostgreSQL SQLite SQL Server Oracle

Figure 1. de dados de cada *framework*

A segunda análise feita, foi na configuração inicial de cada *framework,* ou seja, os requisitos mínimos necessários para cada um deles funcionar, segundo suas respectivas documentações. Foi analisado primeiramente o *framework* CakePHP.

Requisitos para funcionamento:
a) Um servidor. Por exemplo: Apache
b) mod_rewite ativo (opcional)
c) PHP 5.6.0 ou superior
d) Extensão mbstring ativa
e) Extensão intl ativa

Requisitos para instalação:
a) Composer

Permissões:
a) Permissão total nos diretórios *logs* e *tmp*.

Logo após, foi analisado o *framework* Laravel.

Requisitos para funcionamento:
a) Um servidor. Por exemplo: Apache
b) mod_rewite
c) PHP 7.1.3 ou superior
d) Extensão mbstring ativa
e) Extensão OpenSSL ativa
f) Extensão PDO ativa
g) Extensão Tokenizer ativa
h) Extensão XML ativa
i) Extensão Ctype ativa
j) Extensão JSON ativa

Requisitos para instalação:
a) Composer

Permissões:
a) Permissões totais nos diretórios storage e bootstrap/cache.

Como se pode observar com os resultados referentes a configuração inicial de cada *framework*, o CakePHP exige muito menos requisitos para funcionamento em relação ao Laravel, porém os requisitos para a instalação e permissões de diretórios, os dois *frameworks* têm as mesmas exigências.

Na parte referente as requisições no servidor, foram feitos 10 testes com cada *framework,* utilizando o comando: ab -n 1000 -c 10 http://localhost/nome-da-aplicação/, onde são simuladas 1000 requisições com 10 usuários simultâneos. Ou seja, 100 requisições por usuário. Com isso foi analisado o número de requisições completadas com sucesso, a quantidade de requisições por segundo e também o tempo em cada requisição. Nessa parte o *framework* Laravel obteve um desempenho muito acima do CakePHP, (tabelas 1 e 2).

Tabela 1 – Testes de requisições com o framework CakePHP

Testes	Requisições Completas	Requisições/segundos	T./Requisição
Teste 1	1000	1,76	5673,138ms = 5,673138s
Teste 2	1000	1,73	5793,008ms = 5,793008s
Teste 3	1000	1,76	5672,927ms = 5,672927s
Teste 4	1000	1,75	5710,840ms = 5,710840s
Teste 5	1000	1,69	5919,568ms = 5,919568s
Teste 6	1000	1,70	5893,296ms = 5,893269s
Teste 7	1000	1,71	5849,740ms = 5,849740s
Teste 8	1000	1,71	5833,775ms = 5,833775s
Teste 9	1000	1,72	5818,487ms = 5,818487s
Teste 10	1000	1,71	5849,366ms = 5,849366s

Desvio padrão: 0,025033311

Tabela 2 – Testes de requisições com o framework Laravel

Testes	Requisições Completas	Requisições/segundos	T./Requisição
Teste 1	1000	29,86	334,892ms = 0,334892s
Teste 2	1000	28,28	353,584ms = 0,353584s
Teste 3	1000	29,91	334,356ms = 0,334356s
Teste 4	1000	30,29	330,117ms = 0,330117s
Teste 5	1000	28,06	356,381ms = 0,356381s
Teste 6	1000	27,56	362,789ms = 0,362789s
Teste 7	1000	27,95	357,804ms = 0,357804s
Teste 8	1000	27,66	361,589ms = 0,361589s
Teste 9	1000	27,61	362,223ms = 0,362223s
Teste 10	1000	27,64	361,841ms = 0,361841s

Desvio padrão: 1,090563565

A análise seguinte, foi referente as transações do banco de dados, como INSERT, SELECT, UPDATE e DELETE, respectivamente nessa ordem e também o uso de memória da aplicação em cada um dos testes. Foram feitos também 10 testes para cada função e verificado o tempo de execução em segundos de cada uma delas, e verificado o uso de memória da aplicação em MegaBytes (MB) (tabelas 3 e 4).

Tabela 3 – Testes de transações do banco de dados com o framework CakePHP

Teste	INSERT	SELECT	UPDATE	DELETE	Memória Usada
Teste 1	0,019s	0,001s	0,016s	0,088s	8,39MB
Teste 2	0,001s	0,001s	0,001s	0,031s	8,39MB
Teste 3	0,003s	0,001s	0,014s	0,079s	8,39MB
Teste 4	0,001s	0,001s	0,025s	0,117s	8,40MB
Teste 5	0,001s	0,001s	0,019s	0,032s	8,40MB
Teste 6	0,035s	0,001s	0,016s	0,125s	8,41MB
Teste 7	0,001s	0,001s	0,001s	0,056s	8,41MB
Teste 8	0,003s	0,001s	0,024s	0,056s	8,41MB
Teste 9	0,001s	0,001s	0,013s	0,089s	8,42MB
Teste 10	0,001s	0,001s	0,007s	0,166s	8,40MB

Tabela 4 – Testes de transações do banco de dados com o framework Laravel

Testes	INSERT	SELECT	UPDATE	DELETE	Memória Usada
Teste 1	0,052s	0,610s	0,040s	0,067s	9,76MB
Teste 2	0,041s	0,700s	0,041s	0,048s	9,77MB
Teste 3	0,055s	0,680s	0,068s	0,051s	9,76MB
Teste 4	0,056s	0,700s	0,070s	0,063s	9,71MB
Teste 5	0,087s	0,680s	0,072s	0,037s	9,71MB
Teste 6	0,104s	0,600s	0,118s	0,027s	9,68MB
Teste 7	0,286s	0,680s	0,068s	0,052s	9,76MB
Teste 8	0,043s	0,700s	0,059s	0,052s	9,76MB
Teste 9	0,043s	0,590s	0,059s	0,052s	9,71MB
Teste 10	0,039s	0,590s	0,058s	0,059s	9,71MB

Como se pode ver nas tabelas com os resultados dos testes, o *framework* CakePHP teve um desempenho melhor que o Laravel nas funções INSERT, SELECT e UPDATE e também teve um desempenho melhor em relação ao uso de memória da aplicação, já que em todos os testes ele teve uma média de 8.40MB de memória utilizada, enquanto que o Laravel teve uma média de 9.73MB de memória utilizada. Já na função DELETE o Laravel teve um desempenho melhor que o CakePHP.

Em relação aos materiais existentes na literatura sobre assuntos relativos a este tema, esta pesquisa se diferencia dos demais materiais pelo fato de analisar apenas estes dois *frameworks*, utilizando metodologias diferentes dos trabalhos existentes. Sendo que os trabalhos pesquisados como referência para esta pesquisa, geralmente analisavam mais de dois *frameworks* PHP, onde o *framework* Laravel, por exemplo, quase nunca se fazia presente, ou então eram analisados *frameworks* da linguagem Java, como na pesquisa de Franco (2011), que realizou um estudo comparativo entre *frameworks* Java para desenvolvimento de aplicações *web*: JSF 2.0, Grails e Spring *web* MVC, ou utilizavam ferramentas diferentes para as analises, como por exemplo no trabalho de Bagestan (2016), que faz um estudo comparativo de *frameworks* PHP, com enfoque no CodeIgniter, e utilizou uma ferramenta chamada *PHPMetrics,* para realização das análises, porém não foram analisados o tempo de resposta nas requisições, quantidades de requisições por segundo, tempo em cada requisição e também não foram analisadas as transações com o banco de dados. Nesta pesquisa estes testes estão presentes.

No trabalho de Lacerda (2013), pode-se observar que foi feita uma análise entre frameworks PHP, Java e .NET. Porém análises muito mais completas, onde foram feitos vários testes como teste de leitura de registros, teste de construção de uma estrutura map, teste de leitura e inserção de registros, teste de leitura de ficheiros, teste de criação de lista, teste de criação de tabela em HTML. Alguns destes testes não estão presentes nesta pesquisa, como por exemplo, o teste de criação de tabela em HTML.

Na pesquisa de HENNING (2013), onde foi feita uma análise de *frameworks* para o desenvolvimento de sistema de controle sindical, foram analisados quatro *frameworks* PHP e foram feitas análises referentes à licença, facilidades de aprendizagem, complexidades de instalação, segurança, agilidades no desenvolvimento, performance, impressão de relatórios e suporte e documentação. Algumas destas analises estão também presentes nesta pesquisa, como a complexidade de instalação, onde nesta pesquisa se refere as configurações iniciais. Porém não foram vistas as análises referentes às transações com o banco de dados, tempo de resposta nas requisições, quantidades de requisições por segundo, tempo em cada requisição e também não se observou os bancos de dados suportados por cada *framework.*

8. Conclusão

Nesta pesquisa, foi realizada a comparação entre os *frameworks* de PHP, Laravel e CakePHP, onde foram comparados em relação ao número de bancos de dados suportados por cada *framework*, configuração inicial de cada um deles, o número de requisições por segundo de ambos, o tempo de cada requisição, número de requisições completadas com sucesso, transações com o banco de dados (INSERT, SELECT, UPDATE e DELETE) e consumo de memória de cada *framework* em ambas as aplicações.

Para a realização das comparações, foram desenvolvidas duas aplicações, uma para cada *framework*, com as mesmas funcionalidades, interface gráfica e objetivo, onde cada uma delas teve a distribuição Kappa de quatro parâmetros implementada.

Na parte da implementação do algoritmo para resolver a distribuição Kappa de quatro parâmetros, pode-se notar que é um algoritmo bastante complexo e extenso. Este algoritmo havia sido implementado anteriormente apenas nas linguagens Fortran e R, sendo, portanto transposto e implementado na linguagem PHP.

Na parte da análise do número de bancos de dados suportados, observou-se que o CakePHP suporta um banco de dados a mais que o Laravel.

Em relação a configuração inicial mínima para cada *framework*, o CakePHP foi melhor que o Laravel na parte de requisitos para funcionar, onde exige menos requisitos, porém os requisitos para a instalação e permissões de diretórios, os dois *frameworks* têm as mesmas exigências.

Na análise referente ao número de requisições por segundo e tempo de cada requisição, o Laravel se mostrou melhor que o CakePHP, porém no número de requisições completadas, ambos tiveram o mesmo resultado.

Referente as transações com o banco de dados, o CakePHP teve um desempenho melhor nas funções INSERT, SELECT e UPDATE, já na função DELETE o Laravel obteve um melhor desempenho.

E por fim, nos testes de consumo de memória o CakePHP obteve um desempenho melhor em todos os testes, consumindo cerca de 1.33MB de memória a menos que o Laravel.

Nos cenários criados para esta pesquisa, o *framework* CakePHP mostrou-se melhor na maioria das análises realizadas.

Os objetivos gerais e específicos foram atingidos com sucesso, e os resultados documentados podem servir de auxílio no processo de escolha entre os dois *frameworks* de PHP.

9. Referências

BAGESTAN, Flávio Henkes. Estudo comparativo de frameworks PHP, com enfoque no Codeigniter. 2016. 57 p. Monografia (Tecnólogo em Sistemas para Internet), Instituto Federal Sul-rio-grandense, Campus Passo Fundo, Passo Fundo, 2016.

CAKE SOFTWARE FOUNDATION. CakePHP Cookbook Documentation. 2018.

DANTAS, C. E. O. ; PINTO, E. J. A. Análise de Freqüência Regional das Precipitações Máximas Diárias da Bacia do rio São Francisco no Estado de Minas Gerais - Brasil. In: XIV IWRA Word Water Congress, 2011, Porto de Galinhas. XIV IWRA Word Water Congress: Livro de Resumos, 2011.

DIAS, Carlos Henrique Gonçalves; AMORIM, Wisney Almeida; CARVALHO JÚNIOR, Nelson Ribeiro de. Desenvolvimento de Aplicações Web e Dispositivos Móveis Utilizando Frameworks. Revista Pensar Tecnologia, Belo Horizonte, v. 4, n. 1, p.1-18, jan. 2015.

Hosking, J. R. M. (1994). The four-parameter kappa distribution. IBM J. Res.
Deve Eop- 38 (3), 25 1-258.

LACERDA, José António Vieira. Análise de frameworks de desenvolvimento web para a conversão de uma aplicação desktop. 2013. 188 f. Dissertação (Mestrado) - Curso de Tecnologia e Gestão de Sistemas de Informação, Escola Superior de Tecnologia e Gestão, Instituição Politécnico de Viana do Castelo, Viana do Castelo, 2013.

NASCIMENTO, Eduardo Simões. Modelagem e Protótipo de um Sistema para Gerenciamento de Reuniões/Eduardo Simões Nascimento. -- Formiga: IFMG, 2017. 101p.: il

PANSERA, Wagner Alessandro. Distribuição generalizada de chuvas máximas no Estado do Paraná. Cascavél, PR: UNIOESTE, 2013. 94 f.; 30 cm.

PISCKE, Leandro Salvatti. Framework em Java para geração de telas no modelo CRUD Baseado em XML e objetos remotos utilizando a arquitetura MVC e padrões. 2007. 68 f. TCC (Graduação) - Curso de Ciência da Computação, Curso de Ciência da Computação, Universidade Regional de Blumenau, Blumenal, 2007.

REBECCHI, Miguel de Melo. Atendimento de TI a Micro e Pequenas Empresas: Motivos Pelos Quais Os Fornecedores de Soluções de TI Não Conseguem se Estabelecer entre as MPEs e Soluções. 2012. 110 f. Monografia (Especialização) - Curso de Tecnologia em Processamento de Dados, Departamento de Tecnologia da Informação (dti), Faculdade de Tecnologia de São Paulo, São Paulo, 2012.

SILVA, Matheus Rodrigues Rosado da. Projeto e desenvolvimento de um sistema para gerenciamento de trabalhos de conclusão de curso. 2017. 37 f. TCC (Graduação) - Curso de Sistemas de Informação, Faculdade de Ciência da Computação, Universidade Federal de Uberlândia, Uberlândia, 2017.

SILVA, E. Patriky. Utilização do framework CakePHP para desenvolvimento de Websites em PHP. 2011. Trabalho de conclusão de curso (Tecnologia em Análise e Desenvolvimento de Sistemas), Universidade Tecnológica Federal do Paraná. Medianeira 2011.