

**INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA SUL-RIO-
GRANDENSE - CÂMPUS PASSO FUNDO
CURSO DE TECNOLOGIA EM SISTEMAS PARA INTERNET**

FLÁVIO HENKES BAGESTAN

**ESTUDO COMPARATIVO DE FRAMEWORKS PHP, COM ENFOQUE
NO CODEIGNITER**

Prof. Me. Adilso Nunes de Souza

PASSO FUNDO

2016

FLÁVIO HENKES BAGESTAN

**ESTUDO COMPARATIVO DE FRAMEWORKS PHP, COM ENFOQUE
NO CODEIGNITER**

Monografia apresentada ao Curso de Tecnologia em Sistemas para Internet do Instituto Federal Sul-rio-grandense, Campus Passo Fundo, como requisito parcial para a obtenção do título de Tecnólogo em Sistemas para Internet.

Orientador: Prof. Me. Adilso Nunes de Souza

PASSO FUNDO

2016

FLÁVIO HENKES BAGESTAN

**ESTUDO COMPARATIVO DE FRAMEWORKS PHP, COM ENFOQUE NO
CODEIGNITER**

Trabalho de Conclusão de Curso aprovado em ____/____/____ como requisito parcial para a obtenção do título de Tecnólogo em Sistemas para Internet

Banca Examinadora:

Prof. Me. Adilso Nunes de Souza
Orientador

Prof. Me. Jair José Ferronato
Avaliador

Prof. Me. Maikon Cismoski dos Santos
Avaliador

Prof. Me. Adilso Nunes de Souza
Coordenação do Curso

PASSO FUNDO
2016

*À minha família, pelo apoio
e incentivo em todos os momentos.*

AGRADECIMENTOS

Agradeço a minha família por todo o apoio e incentivo, o que contribuiu para a conquista de mais um objetivo.

Agradeço também ao meu orientador, Professor Me. Adilso Nunes de Souza, por me acompanhar nessa jornada, compartilhando conhecimentos, que foram fundamentais para a realização deste trabalho.

Por fim, agradeço a todos os professores do Instituto Federal Sul-rio-grandense Campus Passo Fundo, pela dedicação e os ensinamentos no decorrer do curso. E a todas as pessoas que, de alguma forma, contribuíram para a conclusão dessa etapa.

“Quanto mais aumenta nosso conhecimento,
mais evidente fica nossa ignorância”

John F. Kennedy

RESUMO

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.

Palavras-chave: CodeIgniter, Comparativos, *Framework* php, PHP.

ABSTRACT

The study presents a comparative of the main PHP frameworks, and the application of CodeIgniter in a system for generation and maintenance of budgets. The main motivation for the realization of this study is to provide a comparative between the five main PHP frameworks - CakePHP, CodeIgniter, Symfony, Yii e Zend -, demonstrating the characteristics of each one, providing the knowledge and the practical application, in the development of a system that makes use of the functionalities of CodeIgniter framework, and performs the task of generation and maintenance of budgets. During the study, common features between the frameworks were observed, as well as differences between them, that allowed expand the possibilities in the web applications development.

Keywords: CodeIgniter, Comparatives, PHP framework, PHP.

LISTA DE FIGURAS

Figura 1 - Crescimento da linguagem PHP	16
Figura 2 - Padrão MVC	17
Figura 3 - Exemplo de implementação da classe modelo	18
Figura 4 - Exemplo de visão	18
Figura 5 - Exemplo de implementação da classe controlador	19
Figura 6 - Exemplo de Diagrama de Casos de Uso	21
Figura 7 - Exemplo de Diagrama de Classes	22
Figura 8 - Fluxo da aplicação	24
Figura 9 - Fluxo de execução típico de uma aplicação Yii	26
Figura 10 - Ciclo de requisição do CakePHP	28
Figura 11 - Fluxo de uma aplicação Symfony	29
Figura 12 - Etapas do desenvolvimento do trabalho	30
Figura 13 - Gráficos de análise com base nas métricas de software	36
Figura 14 - Diagrama de casos de uso do sistema	39
Figura 15 - Diagrama de classes do sistema	40
Figura 16 - Tela inicial de login.....	41
Figura 17 – Página inicial da interface do painel administrativo	41
Figura 18 - Menu superior direito	42
Figura 19 – Formulário e botões Adicionar e Limpar.....	42
Figura 20 - Formulário e botão para alterar registro.....	43
Figura 21 - Listagem de dados e botões alterar e excluir.....	43
Figura 22 – Exemplo de paginação.....	44
Figura 23 - Validação de campos editáveis.....	44
Figura 24 - Validação do campo data.....	45
Figura 25 - Mensagem de erro no <i>login</i>	45
Figura 26 - Mensagem de violação de integridade.....	46
Figura 27 - Página de orçamentos	46
Figura 28 - Formulário para adição de orçamento	47
Figura 29 - Opções de inclusão e impressão do orçamento	47
Figura 30 - Página de adição de produtos ao orçamento.....	48
Figura 31 - Página de adição de serviços ao orçamento	48
Figura 32 - Orçamento gerado em formato PDF	49

LISTA DE TABELAS

Tabela 1 - Características dos frameworks	34
---	----

LISTA DE ABREVIATURAS E SIGLAS

API – *Application Programming Interface*

ASP - *Active Server Pages*

BCIT - *British Columbia Institute of Technology*

CPU - *Central Processing Unit*

CRUD – *Create, Read, Update, Delete*

HTTP - *Hypertext Transfer Protocol*

HTML - *Hypertext Markup Language*

JSP - *Java Server Pages*

MVC – *Model, View, Controller*

OOSE - *Object-Oriented Software Engineering*

OMT - *Object Modeling Technique*

PHP - *Hypertext Preprocessor*

SEI – *Software Engineering Institute*

SGBD - *Sistema de Gerenciamento de Banco de Dados*

SOLID - *Single responsibility, Open-closed, Liskov substitution, Interface segregation and Dependency inversion*

SQL - *Structured Query Language*

UML - *Unified Modeling Language*

WAF - *Web Application Frameworks*

XHTML - *Extensible Hypertext Markup Language*

SUMÁRIO

1	INTRODUÇÃO.....	13
1.1	OBJETIVOS	13
1.1.1	Objetivos gerais.....	14
1.1.2	Objetivos específicos.....	14
1.2	ESTRUTURA DO TRABALHO	14
2	REFERENCIAL TEÓRICO	15
2.1	PHP	15
2.2	MYSQL	16
2.3	MVC.....	17
2.4	UML.....	19
2.4.1	Diagrama de casos de uso	20
2.4.2	Diagrama de classes	21
2.5	FRAMEWORKS	22
2.5.1	CodeIgniter.....	24
2.5.2	Yii	25
2.5.3	Zend	27
2.5.4	CakePHP.....	27
2.5.5	Symfony	28
3	METODOLOGIA.....	30
3.1	REVISÃO BIBLIOGRÁFICA	31
3.2	ESTUDO E AVALIAÇÃO DOS FRAMEWORKS	31
3.2.1	Métricas de software	31
3.3	LEVANTAMENTO DE REQUISITOS.....	32
4	RESULTADOS OBTIDOS.....	34
4.1	COMPARATIVO ENTRE OS FRAMEWORKS	34
4.2	ANÁLISE DAS MÉTRICAS DE SOFTWARE.....	35
4.3	DESCRIÇÃO DO SISTEMA.....	37
4.4	MODELAGEM	38
4.4.1	Diagrama de casos de uso	38
4.4.2	Diagrama de classes	39

4.5	FUNCIONAMENTO DO SISTEMA.....	40
4.5.1	Tela inicial	40
4.5.2	Interface da área administrativa	41
4.5.3	Padrões do sistema.....	42
4.5.4	Validação de campos editáveis	44
4.5.5	Mensagens do sistema	45
4.5.6	Geração de orçamentos	46
5	CONSIDERAÇÕES FINAIS.....	50
	REFERÊNCIAS.....	51
	APÊNDICES	54

1 INTRODUÇÃO

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 do sistema. No caso do *CodeIgniter*, objeto de estudo deste trabalho, o padrão utilizado é o MVC (*model*, *view* e *controller*), que, conforme Gabardo (2012), separa as camadas de lógica e regras de negócio da camada de apresentação, proporcionando aumento da reutilização de código e facilidade de manutenção.

Entretanto, como há uma grande variedade de *frameworks* disponíveis para diferentes linguagens de programação, como PHP e Java, 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.

Sendo assim, o objetivo do presente trabalho foi realizar o levantamento de conceitos relativos ao desenvolvimento de aplicações *web*, e estudo de *frameworks* para desenvolvimento rápido de aplicações, tendo enfoque no *framework* *CodeIgniter*, e o desenvolvimento de uma aplicação *web* usando os recursos estudados. A fim de prover um comparativo entre o *CodeIgniter* e os principais *frameworks* PHP existentes: *Yii*, *Zend*, *CakePHP* e *Symfony*.

A motivação para o desenvolvimento deste projeto foi a possibilidade de aprofundar os conhecimentos sobre a linguagem PHP e seus *frameworks*, principalmente o *CodeIgniter*. Além disso, outra motivação é o auxílio que o comparativo trará no processo de escolha de um *framework*, tendo em vista os benefícios que o mesmo traz ao desenvolvedor e ao processo de desenvolvimento, contribuindo, assim, para a produtividade no desenvolvimento *web*.

1.1 OBJETIVOS

Nesta seção são apresentados os objetivos gerais e específicos que se pretendeu atingir com a realização deste projeto.

1.1.1 Objetivos gerais

Realizar um estudo do framework CodeIgniter, comparando-o com outros frameworks PHP, e desenvolver um sistema *web* para criação e gerenciamento de orçamentos utilizando os recursos estudados.

1.1.2 Objetivos específicos

- Identificar os conceitos envolvidos no desenvolvimento *web*;
- Estudar o *framework* CodeIgniter;
- Estudar os principais *frameworks* PHP, além do CodeIgniter, sendo eles: *Yii*, *Zend*, *CakePHP* e *Symfony*;
- Comparar o CodeIgniter com os outros *frameworks* selecionados, avaliando os seguintes pontos: configuração, documentação, suporte a diversos bancos de dados e funcionalidades.
- Aplicar métricas de *software* analisar os resultados;
- Projetar e desenvolver o sistema proposto como estudo de caso;

1.2 ESTRUTURA DO TRABALHO

No presente capítulo encontra-se a introdução do trabalho, apresentando as motivações e os objetivos gerais e específicos, que levaram este trabalho a ser realizado. Os próximos capítulos estão organizados da seguinte maneira: no capítulo 2 encontra-se o referencial teórico sobre os conceitos relacionados ao desenvolvimento de sistemas para web, e as características de cada framework.

No capítulo 3 tem-se a metodologia, demonstrando a forma como o projeto foi desenvolvido e como o sistema foi modelado. Já no capítulo 4 são apresentados os resultados obtidos com a realização da pesquisa sobre os frameworks, e o desenvolvimento do sistema proposto. E, por fim, no capítulo 5 as considerações finais.

2 REFERENCIAL TEÓRICO

Neste capítulo, são abordados conceitos que visam dar embasamento teórico ao trabalho. São apresentados conceitos sobre a linguagem PHP, MVC, UML e *frameworks*.

2.1 PHP

O PHP é uma linguagem de programação *server-side* (ou seja, é executada do lado do servidor) de código aberto, que se adequa muito bem ao desenvolvimento web, principalmente por que pode ser embutida dentro de marcações HTML¹ (*HyperText Markup Language*). (PHP, 2015)

Criado em 1995 por Rasmus Lerdorf era chamado de *Personal Home Page Tools* (PHP *Tools*) e, inicialmente, destinava-se a uso pessoal. Na sua segunda versão, lançada em 1997, o PHP introduziu o suporte a banco de dados, manipulação de *cookies* e funções definidas pelo usuário. No ano de 1998 foi criada a terceira versão do PHP, através de uma colaboração entre Rasmus Lerdorf, Andi Gutmans e Zeev Suraski. Para isso, a tecnologia foi totalmente reescrita, no que diz respeito a códigos-fonte, e seu nome foi reduzido para somente PHP, um acrônimo recursivo para “PHP: Hypertext Preprocessor”. (NETCRAFT, 2013)

Em 1999 foi disponibilizada a quarta versão do PHP, que visava torná-lo mais poderoso e atender projetos de grande complexidade. E em 2004 foi lançada a sua quinta versão, que contava com correções de bugs e várias outras características. (MILANI, 2010)

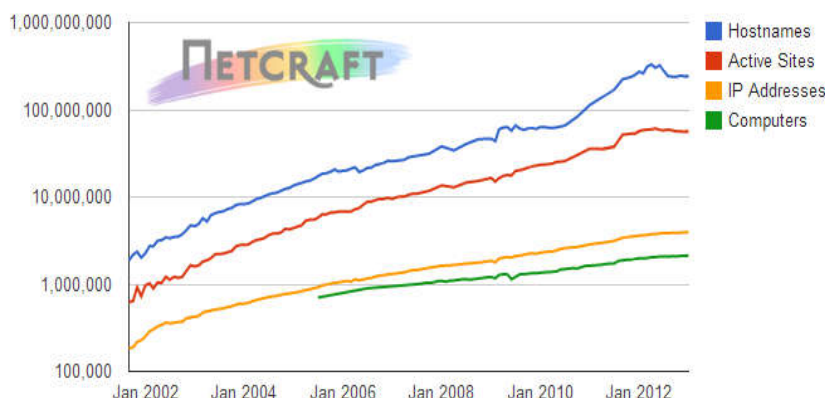
O PHP concorre diretamente com outras tecnologias voltadas ao desenvolvimento web, como o ASP (Active Server Pages) e JSP (Java Server Pages), sendo que o ASP é um meio de executar Visual Basic no lado servidor e depende de um servidor Microsoft IIS para ser executado, e o JSP, apesar dos recursos da linguagem Java, se mostra mais complicado na hora de escrever uma aplicação, em questão de tamanho e complexidade de código-fonte. (MILANI, 2010)

¹ Em português, *Linguagem de Marcação de Hipertexto*. É uma linguagem de marcação utilizada na construção de páginas *web*, que descreve a estrutura e o conteúdo semântico de um documento da *web*.

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. (NIEDERAUER, 2008) Segundo estatísticas de crescimento publicadas pelo site *Netcraft* (www.netcraft.com), até o ano de 2012 mais de 200 milhões de sites utilizavam a linguagem PHP, como pode ser observado na figura 1.

Figura 1 - Crescimento da linguagem PHP



Fonte: Netcraft (2013)

Esses dados comprovam o espaço cada vez maior que o PHP tem conquistado nos últimos anos.

2.2 MYSQL

A linguagem SQL (*Structured Query Language*) é uma linguagem estruturada usada para manipulação de dados, padronizada para bancos de dados relacionais, sendo que cada gerenciador de banco de dados pode ter uma extensão própria da linguagem SQL.

O MySQL é um SGBD (Sistema de Gerenciamento de Banco de Dados) SQL de código aberto, que oferece uma plataforma rápida, prática e de fácil manuseio, além de ser multitarefa e multiusuário. Atualmente, o MySQL é desenvolvido, suportado e distribuído pela *Oracle Corporation*. (ORACLE, 2015) Originalmente, o

MySQL foi desenvolvido para manipular grandes bases de dados de forma mais eficiente do que as soluções existentes e desde então continua em constante desenvolvimento, oferecendo uma grande variedade de funcionalidades.

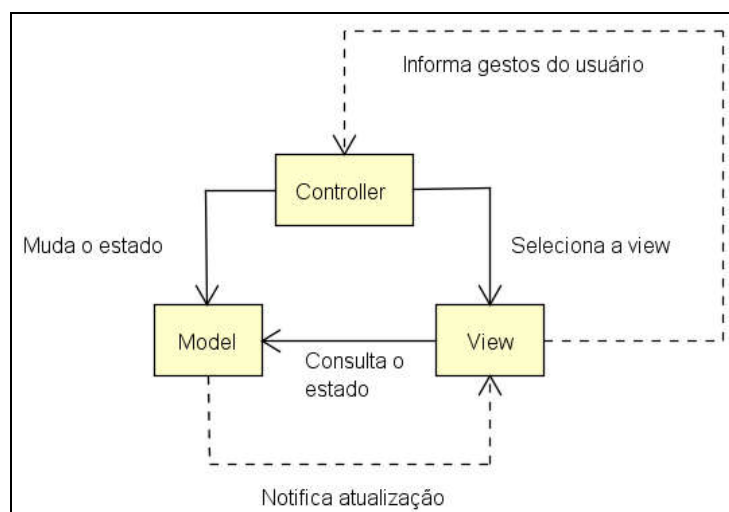
Entre as características do MySQL, podem ser citadas as seguintes: (ORACLE, 2015)

1. Escrito em C e C++;
2. Funciona em diversas plataformas (por exemplo, Linux e Windows);
3. Permite utilizar o poder de processamento de múltiplas CPU's (*Central Processing Unit*);
4. Acesso rápido a tabelas armazenadas em disco através de árvores binárias com compressão de índice;
5. Utiliza um sistema de alocação de memória baseada em *thread* muito rápido;

2.3 MVC

O MVC é um padrão de design de projetos de software, que separa as camadas de lógica e regras de negócio da camada de apresentação (GABARDO, 2012). A sigla MVC faz referência as camadas que o padrão implementa: *Model*, *View* e *Controller*. A figura 2 ilustra o esquema clássico do MVC:

Figura 2 - Padrão MVC



Fonte: Adaptado de Taborda (2011)

A camada *Model*, ou modelo, representa a estrutura de dados. Geralmente as classes de modelo contêm funções para recuperação, inserção e atualização de

informações no banco de dados. Cada modelo é uma classe que herda as características e os comportamentos da superclasse *Model* do *framework*. Um exemplo de modelo do CodeIgniter pode ser visto no trecho de código da figura 3 abaixo:

Figura 3 - Exemplo de implementação da classe modelo

```
<?php
class Noticias_model extends CI_model{
    public function __construct(){
        parent::__construct();
    }
    public function mostra_noticia($id){
        $this->db->where('id',$id);
        return $this->db->get('noticias')->result();
    }
}
?>
```

Fonte: do autor

A camada *View*, ou visão, representa a informação que é apresentada ao usuário, normalmente páginas *web*. Visões são, basicamente, arquivos HTML, XHTML² (*Extensible HyperText Markup Language*) ou HTML5³ que apresentarão os dados gerados pela aplicação para o usuário. Geralmente o formato de arquivo das visões é o PHP, e contam com trechos de código PHP, laços de repetição e funções auxiliares de escrita de código HTML. Um exemplo de visão do CodeIgniter pode ser visto no trecho de código da figura 4 abaixo:

Figura 4 - Exemplo de visão

```
<html>
    <head> <title> Itens </title> </head>
    <body>
        <?php echo heading($cabecalho,1);?>
        <?php foreach ($lista as $item):?>
            <p><?php echo $item;?></p>
        <?php endforeach;?>
    </body>
</html>
```

Fonte: do autor

² Em português, *Linguagem Extensível para Marcação de Hipertexto*. É uma reformulação da linguagem de marcação HTML, baseada em XML (*Extensible Markup Language*).

³ Quinta versão da linguagem HTML que traz melhorias significativas, com novas funcionalidades de acessibilidade e de semântica.

E a camada *Controller*, ou controlador, faz o intermédio entre o modelo, a visão e quaisquer outros recursos necessários para fazer o processamento de uma requisição HTTP⁴ (*Hypertext Transfer Protocol*). (BCIT, 2015) É nos controladores que é feita a chamada a recursos como *helpers*, *drivers* e bibliotecas, que são classes e funções complementares ao núcleo do *framework*.

O controlador herda todos os métodos da superclasse *Controller*. Abaixo, na figura 5, é possível ver um exemplo de um controlador:

Figura 5 - Exemplo de implementação da classe controlador

```
<?php
class Noticias extends CI_Controller {
    public function __construct(){
        parent::__construct();
    }
    public function index(){
        $data['news'] = $this->news_model->get_news();
    }
}
?>
```

Fonte: do autor

2.4 UML

A UML (*Unified Modeling Language*), segundo Guedes (2011) surgiu da união entre os três métodos de modelagem orientada a objetos mais populares da década de 1990: o método de Booch, o método OOSE (*Object-Oriented Software Engineering*) de Rumbaugh, e o método OMT (*Object Modeling Technique*) de Jacobson, contando com o incentivo e financiamento da *Rational Software*. Ela é uma linguagem-padrão para a elaboração da estrutura de projetos de software, adequada para a modelagem de sistemas que vão desde aplicações *web* até sistemas complexos de tempo real.

A UML proporciona uma forma padrão para a preparação de planos de arquitetura de projetos de sistemas, incluindo aspectos conceituais, tais como processos de negócios e funções do sistema, além de itens concretos como as classes escritas em determinada linguagem de programação, esquemas de bancos de dados e componentes de software reutilizáveis. (BOOCH et al, 2012, p. VII)

⁴ Em português, *Protocolo de Transferência de Hipertexto*. Protocolo utilizado para transferência de dados na rede mundial de computadores.

A UML representa o resultado das melhores práticas usadas na modelagem orientada a objetos, e um esforço para tornar a modelagem um processo mais simples. Mota et al. (2004) citam algumas características da UML:

1. A UML é apenas uma sintaxe;
2. É abrangente;
3. Independe da linguagem de programação utilizada;
4. Independe da ferramenta usada;
5. É uma linguagem bem documentada;
6. Sua aplicação não é rígida.

O uso de qualquer ferramenta que suporte a UML permite analisar os requisitos futuros da aplicação e planejar uma solução que os contemple, representando os resultados através dos diagramas que a linguagem oferece. Esses diagramas são divididos em três grupos:

1. Diagramas de estrutura: máquinas de estado, casos de uso e diagramas de atividades;
2. Diagramas de interação: diagrama de comunicação, vista geral de interação, sequencia e temporal;
3. Diagramas de comportamento: diagramas de classes, estrutura de componentes, composições, distribuição, objetos e pacotes.

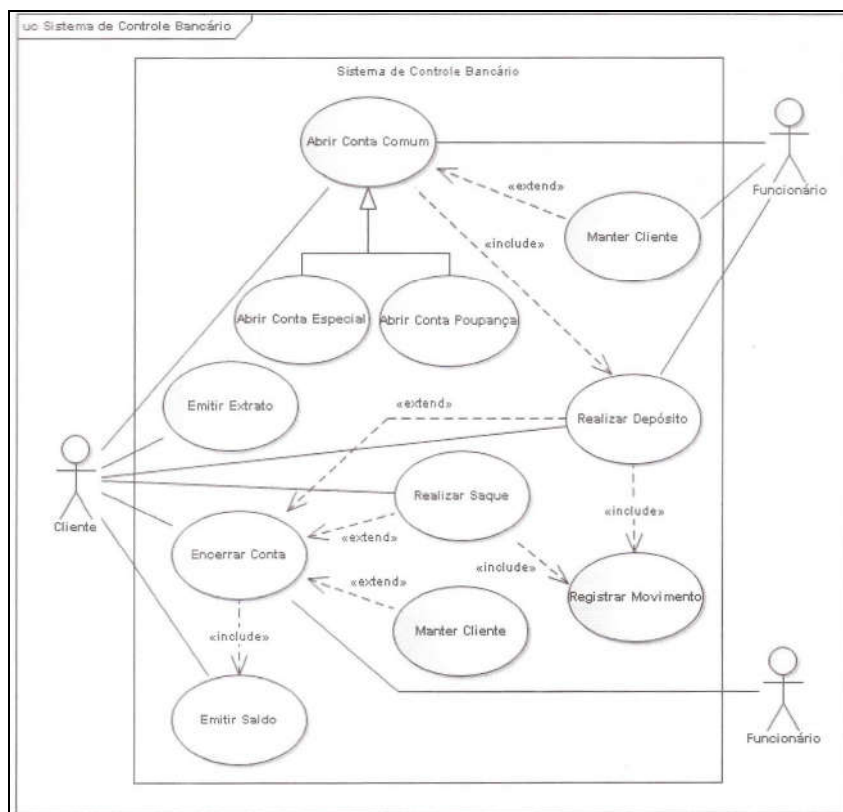
O principal objetivo de se ter uma grande variedade de diagramas é possibilitar múltiplas visões do sistema a ser modelado, pois cada diagrama analisa o sistema, ou parte dele, sob diferentes aspectos, diminuindo a possibilidade de ocorrer falhas no futuro. Nesse trabalho foram utilizados diagramas de casos de uso, para descrever as funcionalidades do sistema, e diagramas de classe para modelar o sistema que foi implementado, demonstrando o conjunto de classes e seus relacionamentos.

2.4.1 Diagrama de casos de uso

Geralmente utilizado nas fases de levantamento e análise de requisitos, o diagrama de casos de uso é o diagrama mais geral e informal da UML, pois apresenta linguagem simples e de fácil compreensão que permite demonstrar uma

ideia geral de como o sistema deverá se comportar. A figura 6 apresenta um exemplo de caso de uso.

Figura 6 - Exemplo de Diagrama de Casos de Uso



Fonte: Guedes (2011)

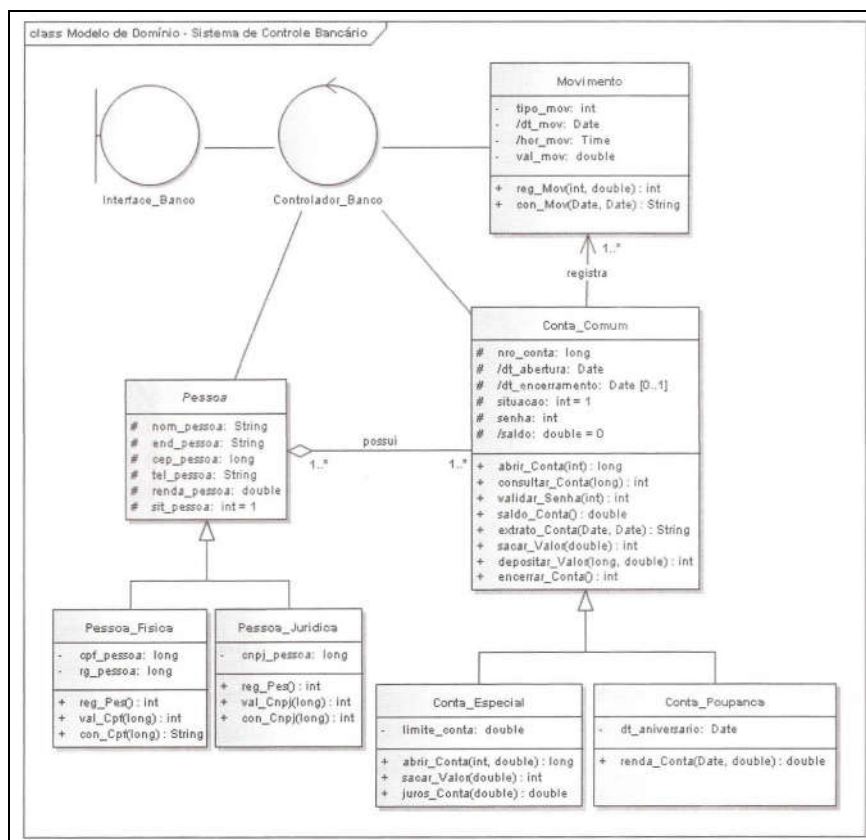
O diagrama de casos de uso busca identificar os atores (usuários, outros sistemas ou até mesmo algum hardware especial) que utilizarão de algum modo o sistema, bem como as funcionalidades que o *software* disponibilizará aos atores. Um caso de uso, representado por uma elipse, é uma funcionalidade ou tarefa realizada pelo ator. As ligações representam a comunicação entre os atores e os casos de uso. Quando um caso de uso é essencial para o comportamento de outro, usa-se a notação *include*, já quando se usa a notação *extend*, significa que o caso de uso pode ser adicionado para descrever o comportamento de outro caso de uso.

2.4.2 Diagrama de classes

Segundo Guedes (2011), o diagrama de classes é um dos mais utilizados e é um dos mais importantes diagramas da UML, porque serve de apoio para a maioria

dos demais diagramas. Ele define a estrutura das classes utilizadas pelo sistema, os atributos e métodos que cada classe tem e estabelece com elas se relacionam entre si. Na figura 7 é possível observar um exemplo de diagrama de classes.

Figura 7 - Exemplo de Diagrama de Classes



Fonte: Guedes (2011)

2.5 FRAMEWORKS

Conforme Buschmann et al. (1996), um framework 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.

Os *frameworks* 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. Segundo Sommerville (2011), os *frameworks* 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.

Três classes de *frameworks* são discutidas por Fayad e Schmidt (1997):

1. Frameworks de infraestrutura de sistema: apoiam o desenvolvimento de infraestruturas de sistema como, por exemplo, interfaces de usuários, comunicações e compiladores.
2. Frameworks de integração de *middleware*: conjunto de normas e classes de objetos associados que oferecem suporte a componentes de comunicação e troca de informações. Um exemplo desse tipo de *framework* é o *Enterprise Java Beans* (EJB).
3. Frameworks de aplicações corporativas: incorporam conhecimentos sobre domínios de aplicações e apoiam o desenvolvimento de aplicações de usuário final.

Já os *frameworks* de aplicações *Web*, ou WAFs (*web application frameworks*), são um tipo mais recente de *framework*. Geralmente a arquitetura de um WAF 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.

A maioria dos WAFs suporta os seguintes recursos:

1. Proteção: inclusão de classes que ajudam a implementar a autenticação de usuário e controle de acesso, garantindo o acesso somente a funcionalidade permitida no sistema.
2. Páginas Web dinâmicas: classes ajudam na definição de *templates* de páginas Web e no preenchimento dinâmico com informações do banco de dados.
3. Suporte de banco de dados: fornecimento de classes que proporcionam uma interface abstrata para bancos de dados diferentes.
4. Gerenciamento de sessão: classes para criação e gerenciamento de sessões.
5. Interação de usuário: a maioria dos WAFs suporta o uso de AJAX, permitindo que sejam criadas páginas *Web* mais interativas.

Um *framework* ajuda a manter o código limpo e organizado, e fornece algumas facilidades genéricas, permitindo que o desenvolvedor projete as aplicações de forma cada vez mais rápida.

2.5.1 CodeIgniter

Motivado pela insatisfação com a qualidade do software e documentação dos *frameworks* disponíveis na época, o fundador e CEO da companhia *EllisLab Inc.*, Rick Ellis, criou o *framework* CodeIgniter. (GABARDO, 2012)

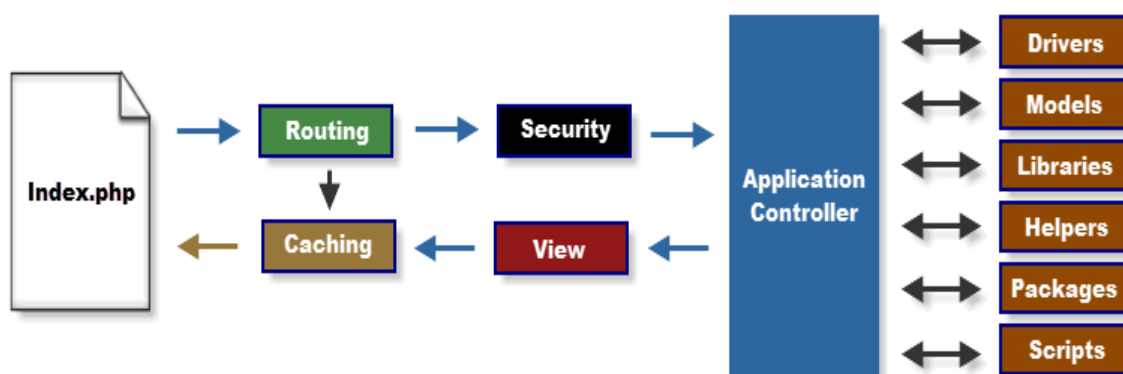
Segundo o BCIT - *British Columbia Institute of Technology* – (2015), o CodeIgniter é um kit de ferramentas de desenvolvimento de aplicações, para desenvolvedores de *web sites* que utilizam a linguagem PHP. Provendo um grande conjunto de bibliotecas para tarefas de uso comum, e uma interface simples e estrutura lógica para acessar essas bibliotecas, o CodeIgniter permite desenvolver projetos de forma muito mais rápida do que escrevendo código-fonte do zero.

O CodeIgniter é um software gratuito e está licenciado sob a licença MIT, cuja característica é não ter muitas restrições de uso, podendo ser usado, modificado, comercializado e redistribuído sem nenhuma limitação desde que seja mantida uma cópia da licença junto do software. Um dos diferenciais em relação a outros *frameworks* PHP é a questão dos recursos necessários, o *core* do CodeIgniter requer somente algumas bibliotecas, o que o torna realmente muito leve. Além disso, possui uma excelente documentação, com informações sobre sua operação básica e conta com uma comunidade de usuários bastante ativa.

Para utilizar o *framework* é necessário ter acesso a um servidor *web* com PHP e MySQL configurados.

A figura 8 ilustra como os dados fluem pelo sistema:

Figura 8 - Fluxo da aplicação



Fonte: EllisLab (2014)

Esse fluxo é explicado da seguinte forma:

1. O arquivo *index.php* serve como o *front-controller*, inicializando os recursos básicos necessários para executar o CodeIgniter;
2. O *Router* analisa a requisição HTTP para determinar o que deve ser feito com ela;
3. Se um arquivo de cache existe, ele é enviado diretamente para o navegador, ignorando a execução normal do sistema;
4. A requisição HTTP e todos os dados enviados pelo usuário são filtrados;
5. O controlador carrega o modelo, as bibliotecas, *helpers*, e quaisquer outros recursos necessários para processar a solicitação específica;
6. A visão finalizada é processada e enviada para o navegador. Se o *caching* estiver habilitado, a visão é armazenada em cache.

2.5.2 Yii

Yii é um *framework* PHP de alto desempenho com arquitetura baseada em componentes e suporte a armazenamento em cache, totalmente orientado a objetos. Criado para ser produtivo, regulável, e de fácil manutenção. O seu nome pode ser definido como um acrônimo para “*Yes It Is*” e, também, significa “*Simples e Evolucionário*” em chinês. (Yii, 2015)

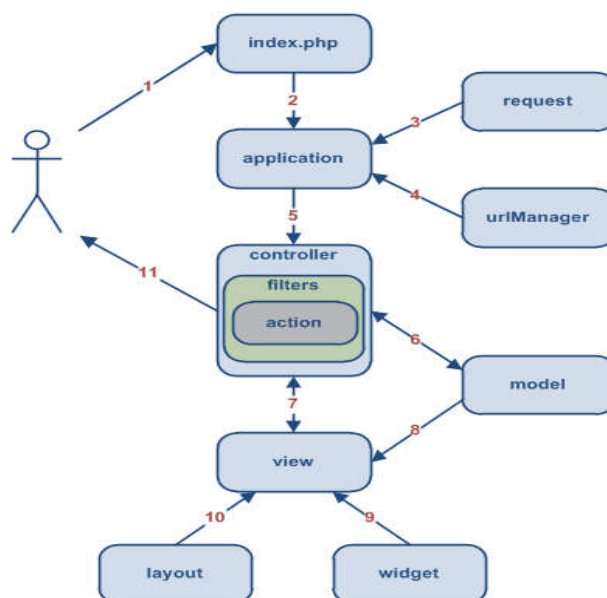
O *Yii* está na versão 2.0 e é especialmente adequado para aplicações de larga escala como, por exemplo, sistemas de comércio eletrônico e sistemas de gerenciamento de conteúdo. Ele possui ferramentas para teste e depuração de aplicações, e pode ser estendido com diferentes tipos de extensões, como, por exemplo: módulos, *widgets*, filtros, validadores, componentes de aplicações, entre outros.

Ele implementa um módulo chamado *gii*, um gerador automático de código, que possibilita a criação de formulários, modelos, módulos, controladores e CRUD's (*Create, Read, Update, Delete*) sem nenhuma codificação. Também oferece diferentes opções de *caching*, uma maneira de aumentar a eficiência de aplicações *web*, tornando possível salvar os dados armazenados em *cache* em um banco de dados, na memória, ou em um arquivo.

O objeto da aplicação encapsula o contexto de execução no qual uma solicitação é processada, reunindo informações básicas sobre a solicitação de um usuário e, a enviando para um controlador apropriado, para posterior

processamento. (LANDVIK, 2013) O fluxo de execução de uma aplicação *Yii* é demonstrado na figura 9.

Figura 9 - Fluxo de execução típico de uma aplicação Yii



Fonte: Yii Framework (2015)

Conforme explicação da página oficial do *framework*, o fluxo ocorre da seguinte maneira:

1. Primeiramente, o usuário faz uma solicitação. Em seguida o servidor processa a requisição;
2. Uma instancia da aplicação é criada pelo *script*, e o *script* é executado;
3. Informações detalhadas da solicitação são obtidas pela aplicação, através de um componente da aplicação chamado *request*;
4. A aplicação determina o controle e a ação necessária com a ajuda do componente chamado *urlManager*;
5. A aplicação cria uma instancia do controle solicitado para poder lidar com a solicitação do usuário. Em seguida, cria e executa filtros associados a esta ação. A ação só é executada se for permitida pelos filtros;
6. Um modelo é lido do banco de dados;
7. A visão é renderizada com o modelo;
8. A visão apresenta os atributos do modelo;
9. A visão executa alguns *widgets*;

10. O resultado do processamento da visão é embutido em um *layout*;
11. A ação conclui o processamento da visão e exibe o resultado ao usuário.

2.5.3 Zend

O Zend é um *framework* flexível que possui uma coleção de componentes reutilizáveis e extensíveis, e permite que os desenvolvedores usem qualquer componente da forma que quiser. Além disso, ele implementa o MVC de forma robusta e com alto desempenho. Oferece, também, abstração de banco de dados, um componente de formulários que implementa e renderiza HTML5, validação e filtragem de modo que essas operações possam ser consolidadas.

Ele é patrocinado pela *Zend Technologies Ltd.*, cofundada por Andi Gutmans e Zeev Suraski, desenvolvedores do PHP. Outros parceiros incluem nomes como Adobe, IBM, Microsoft e Google. O *Zend* é licenciado sob uma licença BSD.

A estrutura de componentes do *Zend* é concebida de tal forma que cada componente possui pouquíssimas dependências de outros componentes. Componentes como o *Zend Auth* e *Zend Acl*, fornecem autenticação e autorização do usuário em todos os ambientes do sistema. (ZEND, 2010) Ele também oferece API's (*Application Programming Interface*) e interface de programação de alguns serviços do Google através do componente *Zend_Gdata*, que é uma interface em PHP 5 para acessar dados do Google.

Em sua atual versão, 2, o *Zend* segue os princípios *SOLID* (*Single responsibility, Open-closed, Liskov substitution, Interface segregation and Dependency inversion*) de design orientado a objetos, uma arquitetura de baixo acoplamento que permite aos desenvolvedores utilizar qualquer componente que desejarem.

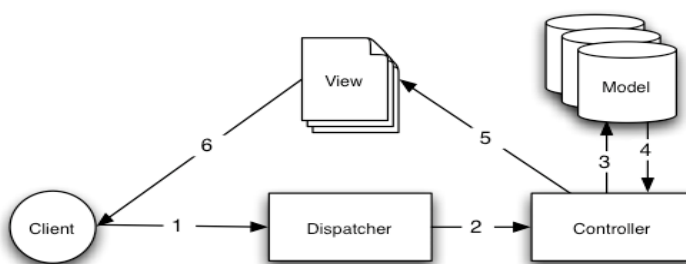
2.5.4 CakePHP

O CakePHP é um *framework* que objetiva o desenvolvimento rápido de forma estruturada e flexível. O *CakePHP* provê uma estrutura organizacional básica que cobre nomenclaturas de classes, arquivos, banco de dados, e outras convenções. Ao seguir as convenções, o *CakePHP* evita configuração desnecessária e cria uma estrutura de aplicação uniforme que faz trabalhar com vários projetos uma tarefa suave.

O CakePHP foi criado por Michal Tatarynowiks em 2005, inspirado no *Ruby on Rails*. Possui comunidade ativa, é compatível com PHP 5.2.6 e superior, oferece *templates* rápidos e controle de acesso flexível. Utiliza a arquitetura MVC, suporta *scaffolding*⁵ e ORM (*Object Relational Mapper*)

O ciclo de requisições do *CakePHP* funciona da seguinte forma (figura 10):

Figura 10 - Ciclo de requisição do CakePHP



Fonte: CakePHP (2014)

O ciclo começa com um usuário solicitando uma página ou recurso na sua aplicação.

1. A requisição é primeiramente processada pelas suas rotas.
2. Depois de a requisição ter sido roteada, o despachante irá selecionar o objeto de *controller* correto para manipulá-la.
3. A *action* do *controller* é chamada e o *controller* interage com os *models* e *components* requisitados.
4. O *controller* delega a criação de resposta à *view* para gerar os dados de saída resultantes dos dados do *model*.

2.5.5 Symfony

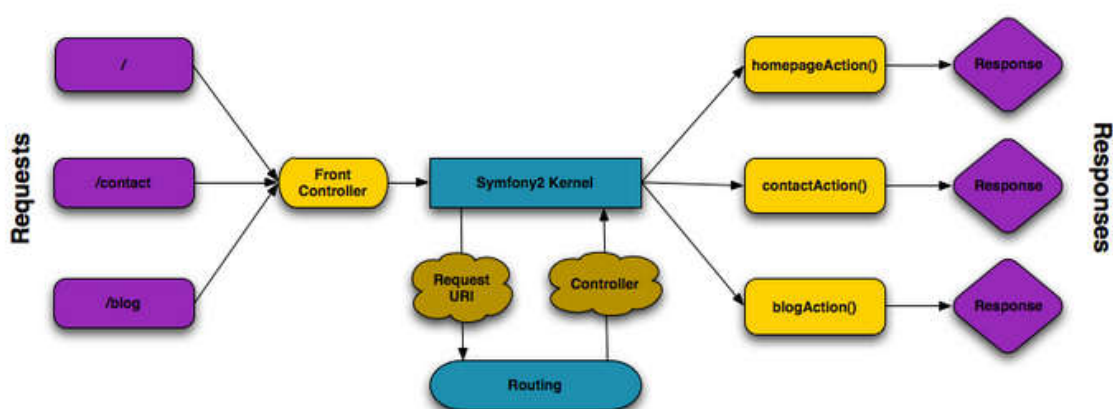
O Symfony é um *framework* PHP de código aberto para desenvolvimento de aplicações *web*. Foi inicialmente concebido pela agência *web* SensioLabs para uso no desenvolvimento de sites para seus próprios clientes, sendo publicado no ano de 2005 sob uma licença de código aberto. Atualmente o *framework* é apoiado pela SensioLabs e por uma vasta comunidade, oferecendo uma série de recursos como

⁵ O *scaffolding* é uma técnica que permite ao desenvolvedor definir e criar uma aplicação básica que possibilite a inserção, seleção, atualização e a exclusão de objetos.

documentação abundante, apoio da comunidade, suporte técnico, consultoria e treinamento. (SYMFONY, 2015)

Com o auxílio do Symfony, é possível escrever códigos estruturados e de fácil manutenção. Isso devido à implementação estrita do modelo MVC e da modularização do *framework*. Uma das melhores características do Symfony é a sua arquitetura de *plugins*, assim, muitas unidades podem ser escritas como *plugins* e ser usadas diversas outras vezes. (BOWLER e BANCER, 2009)

Figura 11 - Fluxo de uma aplicação Symfony



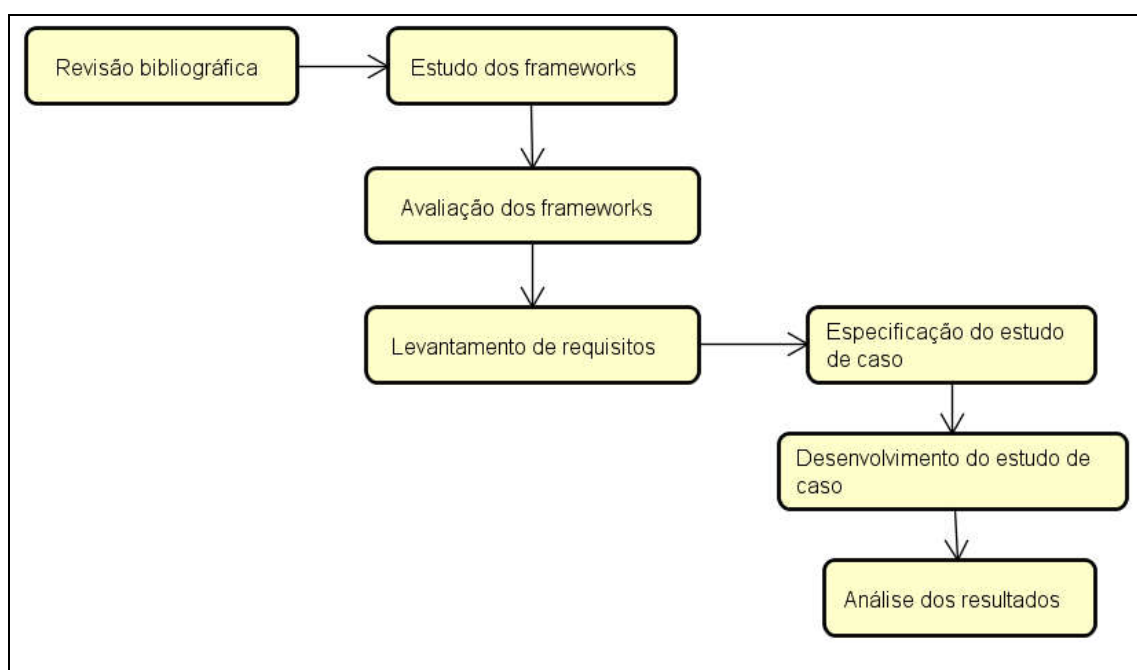
Fonte: Symfony (2015)

As solicitações são interpretadas pelo roteamento e passadas para as funções do controlador que retornam objetos de resposta. Cada solicitação executa um arquivo de *front controller*. O sistema de roteamento determina qual função PHP deve ser executada baseando-se em informações da solicitação e da configuração de roteamento. A função correta é executada, onde o seu código cria e retorna o objeto de resposta adequado.

3 METODOLOGIA

Para o desenvolvimento do trabalho foi seguido o método comparativo, que conforme Marconi e Lakatos (2010) busca estudar semelhanças e diferenças, nesse caso, entre os *frameworks* PHP. Com a finalidade de prover um quadro comparativo que demonstre as vantagens e as desvantagens de cada *framework*. Também foi demonstrado o funcionamento do *framework* CodeIgniter através do desenvolvimento de uma aplicação *web*. O trabalho foi dividido em etapas, partindo do levantamento bibliográfico até a escrita final da monografia, ficando organizado da seguinte forma (figura 12):

Figura 12 - Etapas do desenvolvimento do trabalho



Fonte: do autor

A fim de que o estudo pudesse ser realizado, foram estabelecidos critérios para a execução da comparação entre os frameworks. Os critérios adotados foram: configuração, documentação, suporte a diversos bancos de dados e funcionalidades.

3.1 REVISÃO BIBLIOGRÁFICA

A revisão bibliográfica buscou analisar publicações relacionadas ao tema de estudo, abrangendo toda bibliografia já tornada pública, desde livros, revistas, monografias, teses, pesquisas, artigos, material publicado na internet, entre outros.

Essa etapa proporcionou o conhecimento sobre o que já foi realizado por outros autores e contribui na construção teórica, nas comparações e na validação de resultados.

3.2 ESTUDO E AVALIAÇÃO DOS FRAMEWORKS

O estudo comparativo se constituiu de duas fases: a fase teórica e a fase prática. Na fase teórica, foi estudada a configuração necessária para utilizar cada *framework*, as respectivas documentações e as principais características de cada um.

Para a realização da parte prática, foi necessário o desenvolvimento de um caso de uso para que a arquitetura, a validação dos dados digitados pelo usuário e a implementação do *framework* CodeIgniter pudesse ser estudada de forma mais aprofundada. Além disso, a elaboração de quadro comparativo entre as características de cada *framework*, e a aplicação de métricas de *software*, que visam indicar a qualidade e a complexidade de um *software*, entre outras medidas.

3.2.1 Métricas de software

Conforme Seibt (2001 *apud* MATOS e QUEIROZ, 2015), as métricas de software são técnicas da engenharia de software que auxiliam no planejamento de um projeto, permitindo que dados como custos, esforços, complexidade ou qualidade de um sistema sejam medidos.

Podem ser divididas em dois tipos: medidas diretas e indiretas. As métricas do primeiro grupo procuram medir quantidades observadas [...]. Já as medidas indiretas estão relacionadas às funcionalidades do sistema [...] (MATOS; QUEIROZ, 2015, p. 21)

Sendo assim para avaliação do sistema construído com CodeIgniter foram utilizadas métricas do grupo de medidas indiretas, que englobam medidas como qualidade, complexidade, eficiência, entre outras. As métricas utilizadas foram as seguintes:

- *Cyclomatic Complexity* (CC), ou Complexidade Ciclométrica, que indica a manutenibilidade e a testabilidade de *software*.

- *Maintainability Index* (MI), ou Índice de Manutenibilidade que é baseado no número de linhas de código, nas métricas de Halstead e na complexidade ciclomática.

A métrica de complexidade ciclomática foi proposta por Thomas J. McCabe em 1976, e mede o número de caminhos lógicos linearmente independentes a partir de um código fonte. (WATSON e MCCABE, 1996) Quanto mais caminhos um *software* possuir, mais complexo ele será, aumentando o nível de dificuldade da manutenção.

Conforme Watson e McCabe (1996), a complexidade ciclomática é totalmente baseada na estrutura do grafo de fluxo de controle. Os grafos de fluxo de controle descrevem a estrutura lógica dos módulos de *software*, e são compostos por nós e arestas.

Segundo o SEI (*Software Engineering Institute*), o número da complexidade ciclomática pode indicar os riscos de um algoritmo, sendo classificados da seguinte forma:

- 1-10: baixo risco;
- 11-20: risco moderado;
- 21-50: risco alto;
- 50 ou superior: risco muito alto;

Já a métrica de índice de manutenibilidade foi criada por Paul Oman e Jack Hagemester em 1994, com o objetivo de medir quantitativamente a manutenibilidade de um *software* e, consequentemente, diminuir os custos de manutenção. (VALE, 2013)

Para calcular o valor do índice de manutenibilidade algumas métricas são utilizadas, como a complexidade ciclomática, de McCabe, as medidas de Halstead, que analisam a sintaxe dos códigos, e o número de linhas de código.

3.3 LEVANTAMENTO DE REQUISITOS

O levantamento e a análise de requisitos fazem parte do processo de desenvolvimento de um sistema, visando a coleta de requisitos do sistema para desenvolver a compreensão do domínio da aplicação.

Os requisitos que definem a função de um *software*, ou parte dele, são chamados de requisitos funcionais, e trata-se do conjunto de entradas, comportamento e saídas. Já os requisitos que tem relação com o uso da aplicação em termos de desempenho são chamados de requisitos não-funcionais.

No apêndice A, é possível visualizar os requisitos funcionais e não-funcionais da aplicação desenvolvida.

4 RESULTADOS OBTIDOS

Neste capítulo são apresentados os resultados obtidos durante o desenvolvimento da pesquisa.

4.1 COMPARATIVO ENTRE OS FRAMEWORKS

Através da análise da documentação de cada *framework* foi possível obter os dados necessários para a elaboração de comparativos entre os mesmos. O primeiro comparativo visou produzir uma tabela com as características de cada ferramenta, levando em conta os pontos que foram definidos na seção 1.1.2: configuração, documentação, suporte a diversos bancos de dados, e funcionalidades.

Na tabela 1 é possível observar os dados coletados, organizados em uma tabela:

Tabela 1 - Características dos frameworks

	CakePHP	CodeIgniter	Symfony	Zend	Yii
Conhecimentos necessários:	PHP e Programação orientada a objetos.	PHP e Programação orientada a objetos.	PHP, Programação orientada a objetos, ORM, Console.	PHP5, Programação orientada a objetos e padrões de projeto.	PHP5 e Programação orientada a objetos.
Bancos de dados suportados:	Microsoft BI, Oracle, PostgreSQL, Redis, Memcached, Microsoft SQL Server 2008, MySQL, SQLite, MongoDB, Firebird.	JDBC, OrientDB, Microsoft SQL Server 2008 R2, Cubrid, MySQL, MongoDB, Oracle, IBM DB2, Microsoft SQL Server 2005, Microsoft BI, PostgreSQL.	Microsoft BI, MySQL, PostgreSQL, DynamoDB, GraphDB, MemcacheDB, Apache Jackrabbit, MongoDB, NoSQL, CouchDB, Gemfire, Membase, Oracle.	Apache Derby, IBM DB2, MySQL, Oracle, PostgreSQL, SQLite, DBeaver.	MySQL, SQLite, Microsoft BI, Oracle, PostgreSQL, MongoDB.
Padrões de design:	Active Record, MVC, Event Driven	Active Record, MVC, HMVC.	MVC, Injeção de dependências, HMVC, Restful.	MVC, injeção de dependência, adaptador.	DAO, Active Record, MVC, HMVC, Observer.
Instalação e configuração:	Complexidade baixa.	Complexidade baixa.	Complexidade alta.	Complexidade alta	Complexidade média.

Fonte: do autor

Com os dados coletados pode se observar que *frameworks* como Symfony e Zend são mais recomendados para projetos de médio e grande porte, tendo em vista

a complexidade de instalação e configuração, e os conhecimentos necessários para fazer uso dos mesmos.

Um dos pontos que torna a instalação do Symfony e do Zend mais complexas, é que para isso é necessário utilizar o Composer, gerenciador de dependências para PHP, e digitar comandos no terminal. Nesse aspecto, *frameworks* como o CodeIgniter são considerados de complexidade baixa, por que é necessário apenas baixar o *framework* e descompactar no servidor PHP.

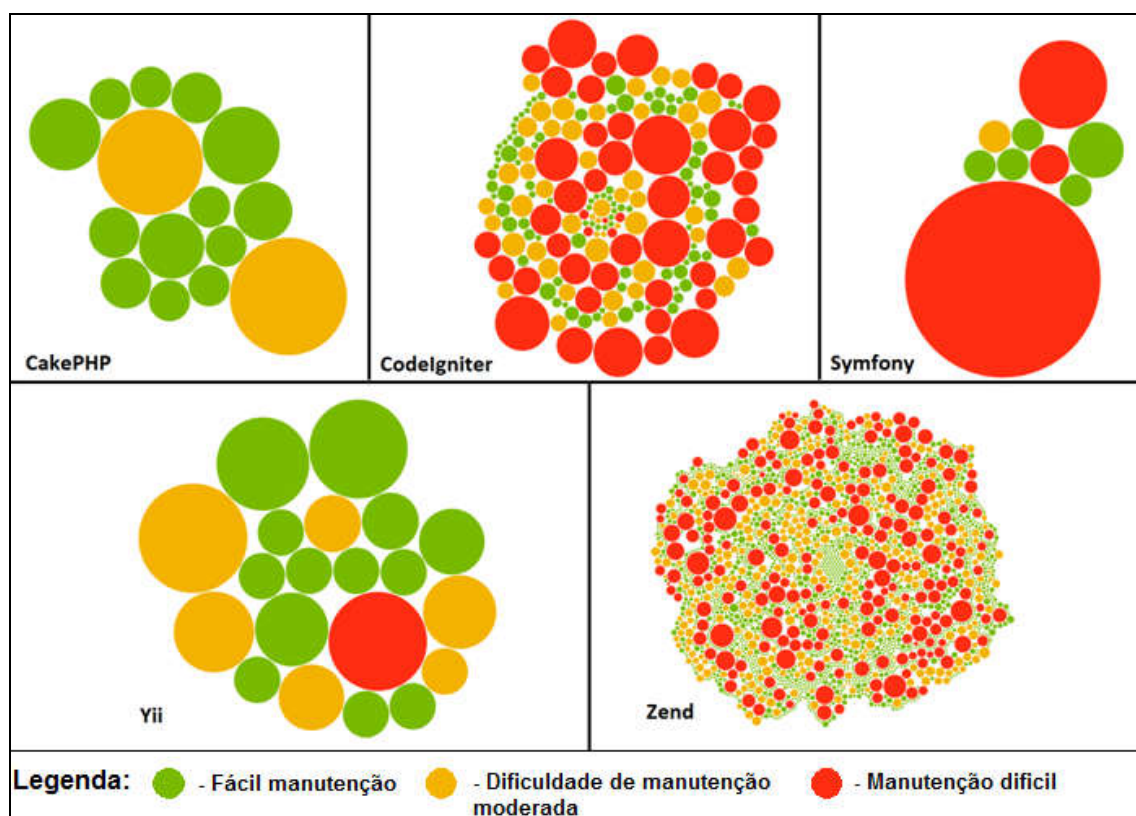
A complexidade de instalação foi definida tendo como base os conhecimentos necessários para se utilizar os *frameworks*. Complexidade baixa indica que o *framework* não requer o uso do Composer ou terminal de comandos, bastando apenas descompactar a ferramenta no servidor PHP. Complexidade média significa que é possível optar pelo uso, ou não, do Composer. E complexidade alta, então, indica que o *framework* só pode ser obtido através do uso de ferramentas como o Composer.

4.2 ANÁLISE DAS MÉTRICAS DE SOFTWARE

Para aplicar as métricas de Complexidade Ciclométrica e Índice de Manutenibilidade, foi utilizado a ferramenta *open-source* PhpMetrics, criada por Jean-François Lépine, para automatizar a análise dos códigos-fonte dos *frameworks* selecionados, produzindo um relatório dos dados coletados.

A seguir, na figura 13, são apresentados os gráficos gerados a partir da análise de cada *framework*, baseando-se nas métricas de *software*.

Figura 13 - Gráficos de análise com base nas métricas de software



Fonte: do autor

Cada arquivo analisado é representado por um círculo, cujo tamanho é definido de acordo com a Complexidade Ciclomática, e a cor do círculo indica o Índice de Manutenibilidade. A cor vermelha simboliza que o arquivo é de difícil manutenção, a cor amarela ou laranja simboliza que o arquivo tem dificuldade de manutenção moderada, e a cor verde indica que o arquivo possui fácil manutenção.

No caso dos *frameworks* estudados é possível notar, primeiramente, a quantidade de arquivos que foram analisados pelo PhpMetrics, sendo que o *framework* Zend supera os demais nesse quesito, tendo, também um grande número de arquivos de difícil manutenção.

Em seguida, o CodeIgniter também apresenta uma quantia considerável de arquivos cuja manutenção pode ser complexa. O *framework* Yii, apesar de ter somente uma indicação com círculo vermelho, possui alguns arquivos com manutenção de dificuldade moderada.

O *framework* CakePHP, por sua vez, na análise realizada não possui nenhuma indicação de arquivo com manutenção difícil.

Para a realização da análise dos *frameworks* foi utilizada a última versão estável de cada um. As versões são especificadas abaixo:

- CakePHP – versão 3.2
- CodeIgniter – versão 3.0.6
- Symfony – versão 3.1
- Yii – versão 2.0.8
- Zend – 2.4.9

4.3 DESCRIÇÃO DO SISTEMA

Além de se estudar o funcionamento, na prática, do CodeIgniter, o objetivo do sistema proposto foi de facilitar a criação de orçamentos através de formulários com informações a respeito de clientes, produtos e serviços previamente armazenados em um banco de dados. Armazenando informações sobre o próprio trabalhador, como dados para contato.

O sistema visa ser usado, principalmente, por trabalhadores autônomos e profissionais liberais. Atualmente o processo de criação do orçamento é realizado de forma manual, o que torna o processo demorado e pouco ágil de se fazer. O procedimento realizado atualmente funciona da seguinte forma: o cliente procura o trabalhador e lhe solicita o orçamento de um serviço, o trabalhador, por sua vez, analisa o que o cliente quer que seja feito, e com base nisso faz uma lista dos materiais que serão necessários e quanto custará cada serviço, após isso as anotações são passadas a limpo e o orçamento é entregue ao cliente.

Na situação em que se encontra, o trabalhador corre o risco de ter que refazer diversas vezes um mesmo orçamento, escrevendo repetidamente as informações. Analisando este processo é possível observar que o sistema desenvolvido é uma solução que pode melhorar e facilitar o processo de criação do orçamento.

Após ser cadastrado no banco de dados, o usuário estará hábil a cadastrar produtos que julgar ser de uso comum em diferentes serviços. Podendo também informar o preço do produto, no caso do cliente deixar a cargo dele a compra de materiais.

O cadastro de clientes é feito igualmente através de formulários, sendo possível visualizar informações a respeito dos clientes após o cadastro. Tipos de

serviços também podem ser cadastrados no sistema, a fim de facilitar o processo de preenchimento do formulário de criação de novo orçamento.

O sistema conta com um módulo para gerar o orçamento. Nesse módulo, o usuário informa o cliente para quem está fazendo o orçamento, o(s) serviço(s) que será/serão realizado(s) e, caso seja necessário, o material para realização do serviço. Depois de preenchidos os campos, o sistema calcula o subtotal de produtos e serviços, e o valor total do orçamento, e gera o orçamento para impressão, formatando o mesmo com as informações referentes ao trabalhador, o número do orçamento, a data de criação do orçamento, os dados do cliente, dados dos serviços e materiais, juntamente com o subtotal e o valor total.

Depois de realizado o processo de criação de orçamento, o usuário pode visualizar, alterar, ou excluir o mesmo. Dessa forma, caso o cliente perca o orçamento ou solicite alguma mudança nas informações, não será necessário preencher tudo novamente, basta selecionar o orçamento e imprimi-lo, ou alterá-lo.

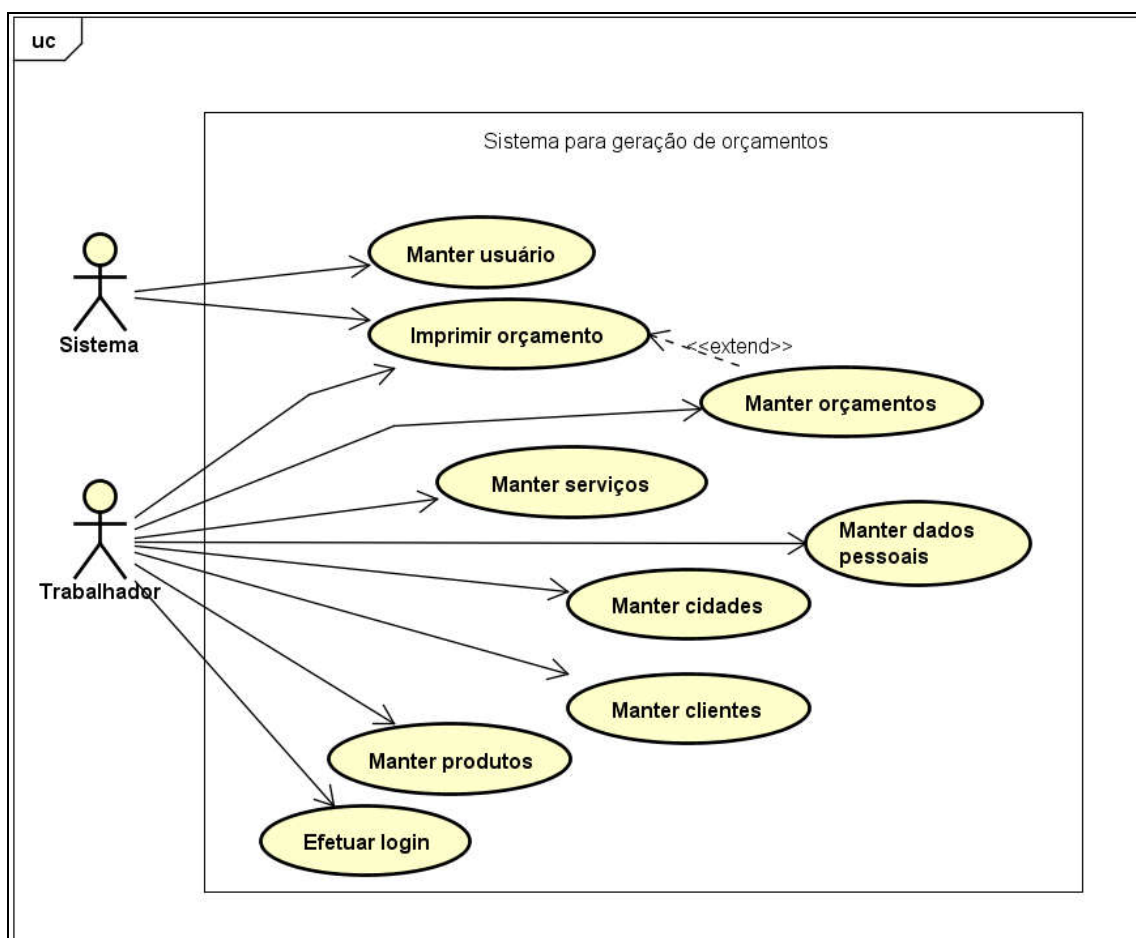
4.4 MODELAGEM

Esta etapa busca desenvolver modelos abstratos do sistema, usando-se a linguagem UML para desenvolver diagramas de casos de uso e de classes para apresentar diferentes visões ou perspectivas do sistema. Esse processo auxilia, principalmente, no entendimento das funcionalidades do sistema.

4.4.1 Diagrama de casos de uso

O diagrama de casos de uso da figura 14 visa proporcionar uma visão externa do sistema, demonstrando as suas funcionalidades, o comportamento do mesmo e quem irá interagir com ele.

Figura 14 - Diagrama de casos de uso do sistema



Fonte: do autor

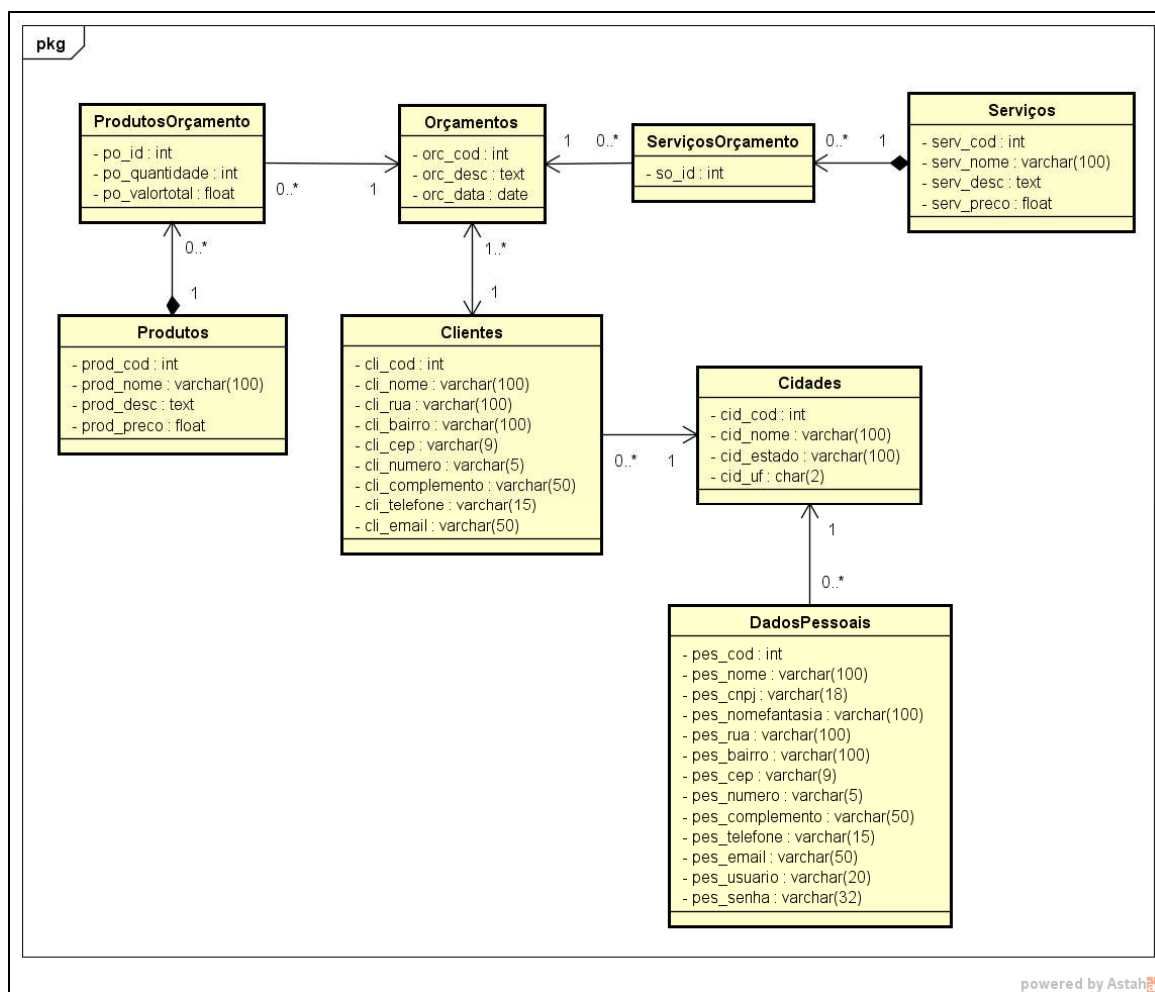
O principal ator é o “Trabalhador”, pessoa que irá utilizar o sistema para gerar os orçamentos, podendo alimentar e consultar o banco de dados, e imprimir os orçamentos. Tudo isso através dos casos de uso que apresentam manutenções como cadastro, alteração, exclusão e consulta.

O ator “Sistema” apresenta a funcionalidade de manter um usuário, ou seja, oferece a possibilidade de cadastrar ou alterar (recuperação de senha, por exemplo) um usuário do sistema. E possui, também, a função de realizar o processo de geração e impressão dos orçamentos.

4.4.2 Diagrama de classes

Esse diagrama demonstra a organização das classes do sistema. Pode-se observar na figura 15 as classes com seus atributos, relacionamentos e multiplicidades.

Figura 15 - Diagrama de classes do sistema



Fonte: do autor

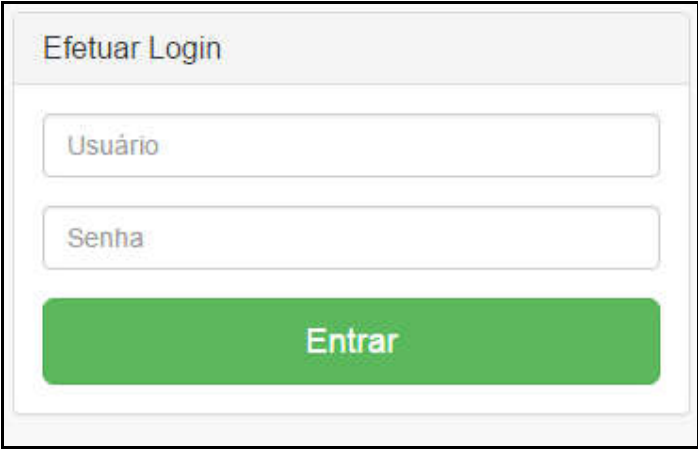
4.5 FUNCIONAMENTO DO SISTEMA

Partindo do conhecimento adquirido durante o capítulo 1, em relação ao padrão MVC e o funcionamento do CodeIgniter, e tendo como base a descrição e a modelagem do sistema apresentados no capítulo 1.6, pode-se desenvolver o sistema de manutenção e geração de orçamentos. Os próximos capítulos apresentam o funcionamento e os padrões do sistema.

4.5.1 Tela inicial

A tela inicial do sistema é a página de *login*, desenvolvida para realizar a autenticação de usuário através da entrada do nome de usuário e sua respectiva senha. A mesma pode ser observada na figura 16.

Figura 16 - Tela inicial de login

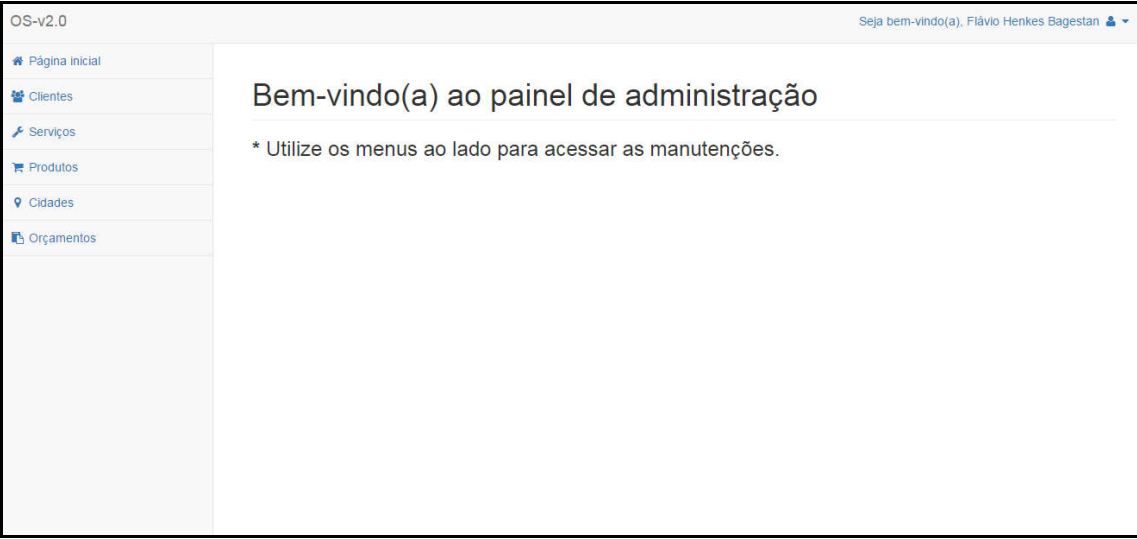
A interface de login é apresentada em um formulário centralizado. No topo, há um cabeçalho com o texto "Efetuar Login". Abaixo dele, há dois campos de entrada: "Usuário" e "Senha", ambos com bordas arredondadas e um ícone de lupa no canto direito. Abaixo dos campos, há um botão verde com o texto "Entrar" em branco.

Fonte: do autor

4.5.2 Interface da área administrativa

Assim que o usuário efetua o *login* e o sistema valida os dados, o mesmo é direcionado a página inicial do painel de administração do sistema. Conforme é exibido na figura 17, o usuário tem acesso às manutenções disponíveis. Sendo elas: Clientes, Serviços, Produtos, Cidades e Orçamentos.

Figura 17 – Página inicial da interface do painel administrativo

A interface do painel administrativo é exibida em uma tela com uma barra superior e um menu lateral. A barra superior contém o texto "OS-v2.0" à esquerda e "Seja bem-vindo(a), Flávio Henkes Bagestan" à direita, com um ícone de perfil de usuário. O menu lateral, à esquerda, contém links para "Página inicial", "Clientes", "Serviços", "Produtos", "Cidades" e "Orçamentos". O conteúdo principal da tela exibe o texto "Bem-vindo(a) ao painel de administração" e uma mensagem de instrução: "* Utilize os menus ao lado para acessar as manutenções."

Fonte: do autor

Ainda na página inicial do painel administrativo o usuário pode acessar os dados pessoais, ou fazer o *logout*, clicando no menu no canto superior direito, conforme é exibido na figura 18.

Figura 18 - Menu superior direito



Fonte: do autor

4.5.3 Padrões do sistema

Nesta seção são apresentadas as opções que o usuário tem em relação à interação com a aplicação. Ao acessar qualquer uma das manutenções é exibido um formulário para inserção de novos dados e, logo ao lado, a listagem dos dados inseridos no banco de dados.

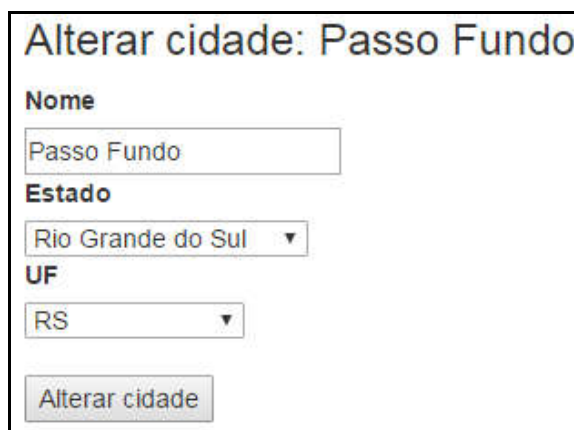
Juntamente com o formulário de inserção estão os botões de “Adicionar” e “Limpar”, na figura 19 pode se observar um exemplo de formulário e os botões.

Figura 19 – Formulário e botões Adicionar e Limpar

Fonte: do autor

O formulário da figura é utilizado para o registro de uma nova cidade no banco de dados, sendo que todos os formulários de inserção de novos registros possuem as mesmas funções. Para os formulários de alteração de dados, figura 20, é exibido o botão “Alterar”, e os campos são preenchidos com dados recuperados do banco de dados.

Figura 20 - Formulário e botão para alterar registro



Alterar cidade: Passo Fundo

Nome
Passo Fundo

Estado
Rio Grande do Sul ▼

UF
RS ▼

Alterar cidade

Fonte: do autor

A forma de listagem de dados é a mesma para todas as manutenções, com exceção da listagem de orçamentos, e são exibidos dois botões que realizam tarefas muito importantes no sistema: “Alterar” e “Excluir”. Quando o usuário clica em alterar é carregado o formulário de alteração com os respectivos dados, e quando é clicado em excluir é exibida uma mensagem de confirmação da exclusão, caso o usuário a confirme e o sistema ateste que outras tabelas não tenham relação de dependência com os dados, o registro é excluído.

As opções para alterar e excluir são exemplificadas na figura 21.

Figura 21 - Listagem de dados e botões alterar e excluir

Cidades cadastradas				
Nome da cidade	Estado	UF	Alterar	Excluir
Florianópolis	Santa Catarina	SC		

Fonte: do autor

A paginação é realizada de forma dinâmica e de acordo com a quantidade de registros existentes no banco de dados. Por padrão são exibidos somente cinco itens por página. A figura 22 mostra um exemplo de paginação.

Figura 22 – Exemplo de paginação



Fonte: do autor

4.5.4 Validação de campos editáveis

A validação dos dados inseridos nos campos dos formulários é feita através de javascript e, principalmente, dos recursos do *framework* CodeIgniter, fazendo uso da biblioteca *Form_validation*. Na figura 23 é exibido um exemplo de formulário com campos que necessitam de validação dos dados.

Figura 23 - Validação de campos editáveis

A imagem mostra um formulário web com o título "Adicionar cliente". Ele contém três campos de entrada: "Nome", "Telefone" e "E-mail". O campo "Nome" está vazio e possui uma mensagem de erro em vermelho abaixo dele: "* O campo Nome é obrigatório.". O campo "Telefone" contém o valor "(54) 9999-9999". O campo "E-mail" contém o valor "teste.com" e possui uma mensagem de erro em vermelho abaixo dele: "* O campo Email deve conter um email válido.".

Fonte: do autor

Neste exemplo é possível observar que o sistema realiza a validação de campos de preenchimento obrigatório ou que devem seguir um padrão, como no caso do endereço de email onde é aplicada uma máscara através de javascript.

Outro exemplo da importância da validação dos dados inseridos pelo usuário pode ser visualizado na figura 24. O sistema não permite que o usuário entre com uma data inválida, evitando mensagem de erro no banco de dados.

Figura 24 - Validação do campo data

Data inválida!

Adicionar orçamento

Cliente

João da Silva ▼

Data

32/13/99999999

Maio 2016

D	S	T	Q	Q	S	S
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31	1	2	3	4

Fonte: do autor

4.5.5 Mensagens do sistema

As mensagens que o sistema apresenta são uma forma de avisar o usuário sobre uma ação executada, ou um erro em alguma operação realizada. No caso de uma tentativa de *login* com credenciais inválidas, é exibida a mensagem de erro apresentada na figura 25:

Figura 25 - Mensagem de erro no *login*

Efetuar Login

Usuário

Senha

Entrar

Usuário e/ou senha incorretos!

Fonte: do autor

Outra mensagem do sistema informa o usuário quando o mesmo tenta excluir algum dado que seja utilizado em outras tabelas. Um exemplo dessa situação é demonstrado na figura 26.

Figura 26 - Mensagem de violação de integridade



Fonte: do autor

4.5.6 Geração de orçamentos

A principal funcionalidade do sistema é a criação de orçamentos e a geração do mesmo em formato PDF. Para realizar tal tarefa o usuário deve acessar a página “Orçamentos”, nessa página há a opção de criar um novo orçamento e, no caso de já haver um ou mais orçamentos cadastrados, é exibida a data de criação e o nome do cliente para quem o orçamento foi elaborado. A figura 27 exibe a página de orçamentos.

Figura 27 - Página de orçamentos



Fonte: do autor

Ao clicar em “Novo orçamento” o usuário é redirecionado para a página de adição de orçamentos, nessa página, que pode ser visualizada na figura 28, é necessário informar os dados iniciais do orçamento, selecionando o cliente, inserindo a data do orçamento e, caso seja necessário, inserindo uma descrição do orçamento.

Figura 28 - Formulário para adição de orçamento

Fonte: do autor

Depois de registrado, ele é listado na página de orçamentos, e a partir daí o usuário pode incluir produtos e serviços no orçamento, ou, então, imprimi-lo, através das opções demonstradas na figura 29:

Figura 29 - Opções de inclusão e impressão do orçamento

Cliente	Data	+ Produtos	+ Serviços	Alterar	Excluir	Imprimir
Joana Pereira	08/05/2016	+ 🛒	+ 🛠	✏	🗑	🖨
João Cláudio de Souza	10/05/2016	+ 🛒	+ 🛠	✏	🗑	🖨
João da Silva	08/06/2016	+ 🛒	+ 🛠	✏	🗑	🖨

Fonte: do autor

Ao clicar no ícone para adicionar produtos é exibida a página para adição de produtos no orçamento, conforme mostrado na figura 30. Sendo possível indicar a quantidade desejada, o sistema calcula o valor total com base no valor unitário do

produto. Também é possível visualizar os produtos que já estão adicionados ao orçamento.

Figura 30 - Página de adição de produtos ao orçamento

Adicionar produto ao orçamento nº: 10

Selecione o produto

Quantidade

Produtos adicionados ao orçamento

Produto	Quantidade	Valor total	☒ Alterar	☐ Excluir
Disjuntor Unipolar	2	R\$ 13,00	Alterar	Excluir

Fonte: do autor

Clicando no ícone para adicionar serviços, é exibida a página para adição de serviços no orçamento, demonstrado na figura 31. Bem como na página de adição de produtos, é possível visualizar os serviços que já foram adicionados ao orçamento.

Figura 31 - Página de adição de serviços ao orçamento

Adicionar serviço ao orçamento nº: 10

Selecione o serviço

Serviços adicionados ao orçamento

Serviço	☒ Alterar	☐ Excluir
Entrada de luz	Alterar	Excluir

Fonte: do autor

Quando o usuário desejar imprimir ou simplesmente visualizar o orçamento, basta clicar na opção “Imprimir”, representada pelo ícone de uma impressora. O orçamento é então gerado em formato PDF e exibido no navegador do usuário. Na figura 32, pode se observar um exemplo de orçamento gerado com base nas informações inseridas:

Figura 32 - Orçamento gerado em formato PDF

1 / 1

Orçamento nº: 10

Data: 08/06/2016

Flávio Henkes Bagestan - Contato: (54) 9945-6744

Endereço: Rua Frederico Graeff, 695, Bairro Berthier - Passo Fundo

Cliente: João da Silva - Contato: (12) 3456-7890

Endereço: x, 543, Bairro y - Rio de Janeiro

Orçamento: Válido por 30 dias.

Cód.	Serviço	Descrição	Valor
1	Entrada de luz	Padrão RGE	R\$ 690,00
Total serviços:			R\$ 690,00

Material necessário:

Qtd.	Produto	Descrição	Valor Unitário	Valor Total
1	Poste de luz aço galvanizado	Trifásico, 5 metros	R\$ 646,71	R\$ 646,71
1	Caixa de luz p/ medidor duplo	Externo trifásico	R\$ 288,55	R\$ 288,55
Total produtos:				R\$ 935,26

Valor Total do orçamento:

R\$ 1.625,26

Fonte: do autor

5 CONSIDERAÇÕES FINAIS

A partir da realização da pesquisa e do desenvolvimento do sistema de manutenção de orçamentos, foi possível observar que a utilização de frameworks, no caso o CodeIgniter, torna o processo de desenvolvimento de sistemas mais rápido e prático, permitindo um alto nível de reutilização de código.

Vale ressaltar que os objetivos do projeto foram alcançados. Através do referencial teórico, principalmente no que se refere as características de cada framework, foi possível realizar os estudos comparativos entre os frameworks CodeIgniter, Yii, Zend, CakePHP e Symfony. Apresentando informações sobre configuração, documentação, bancos de dados suportados e funcionalidades.

O sistema de manutenção de orçamentos foi desenvolvido de acordo com os requisitos e a modelagem apresentada na metodologia. A utilização do CodeIgniter permitiu comprovar que as funcionalidades do framework aceleram o desenvolvimento, uma vez que o mesmo apresenta uma série de recursos que facilitam a escrita de código. Referente ao sistema desenvolvido, o que mais se mostrou interessante foram os recursos de geração e validação de formulários, e a abstração do banco de dados, bem como a facilidade de se utilizar.

As dificuldades deste projeto foram em encontrar a documentação de alguns frameworks e/ou a qualidade de algumas documentações. Tratando-se especificamente do framework CodeIgniter, também foi possível notar algumas dificuldades na utilização do mesmo, principalmente na questão de estilização de alguns layouts do sistema. Os recursos de criação de formulários e tabelas mostraram-se muito eficazes, porém, tornou mais difícil a alteração da aparência dos formulários e tabelas.

Como projetos futuros, há o aprimoramento do sistema de manutenção de orçamentos, proporcionando mais funcionalidades ao usuário. E um estudo mais aprofundado dos frameworks Yii, CakePHP, Zend e Symfony, aplicando em um estudo de caso.

REFERÊNCIAS

BCIT (British Columbia Institute of Technology). *Model-View-Controller*. Disponível em: <http://www.codeigniter.com/user_guide/overview/mvc.html>. Acesso em: 05 de outubro de 2015.

BOOCH, Grady; RUMBAUGH, James; JACOBSON, Ivar. *UML Guia do Usuário*. Elsevier Editora Ltda., 2012.

BOWLER, Tim; BANCER, Wojciech. *Symfony 1.3: Web Application Development*. Packt Publishing Ltd., 2009.

BUSCHMANN, Frank; MEUNIER, Régine; ROHNERT, Hans; SOMMERLAD, Peter; STAHL, Michael. *Pattern-Oriented Software Architecture: A System of Patterns*. John Wiley & Sons, England, 1996.

CAKEPHP. *Cookbook 2.x: Visão Geral do CakePHP*. Disponível em: <<http://book.cakephp.org/2.0/pt/cakephp-overview/understanding-model-view-controller.html>>. Acesso em: 16 de novembro de 2015.

ELLISLAB. *CodeIgniter User Guide Version 2.2.0: Application Flow Chart*. Disponível em: <<https://ellislab.com/codeigniter/user-guide/overview/appflow.html>>. Acesso em: 05 de novembro de 2015.

FAYAD, Mohamed E.; SCHMIDT, Douglas C.. *Object-Oriented Applications Frameworks*. Disponível em: <<http://paginas.fe.up.pt/~aaguiar/as/acm-out97-p32-fayad.pdf>>. Acesso em: 29 de outubro de 2015.

GABARDO, Ademir Cristiano. *PHP e MVC com CodeIgniter: Construa aplicações web dinâmicas em PHP de modo rápido e prático com o framework CodeIgniter*. Novatec Editora Ltda., 2012.

GUEDES, Gilleanes T.A. *UML 2 : Uma Abordagem Prática*. Novatec Editora Ltda., 2011.

LANDVIK, Henri. *Web Application With Yii Framework*. Disponível em: <https://www.theseus.fi/bitstream/handle/10024/65528/web_application_with_yii_framework.pdf?sequence=1>. Acesso em: 16 de novembro de 2015.

MARCONI, Marina de Andrade; LAKATOS, Eva Maria. *Fundamentos de Metodologia Científica*. Editora Atlas S.A., 2010.

MATOS, Davi Rebouças; QUEIROZ, Paulo Gabriel Gadelha. *Avaliação da utilização de framework MVC no desenvolvimento de sistemas Web*. In: Escola Potiguar de Computação e suas Aplicações – EPOCA, 2015, Caicó/RN. Anais EPOCA 2015, 2015, p. 15-24.

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

MOTA, António; VALENTE, Hugo; NAVEGA, Ivo; PACHECO, Pedro; SILVA, Tiago; PACHECO, José. *UML 2.0 – Unified Modeling Language 2.0*. Disponível em: <<http://paginas.fe.up.pt/~ei02084/artigo.pdf>>. Acesso em: 09 de novembro de 2015.

NETCRAFT. *PHP just grows & grows*. Disponível em: <<http://news.netcraft.com/archives/2013/01/31/php-just-grows-grows.html>>. Acesso em: 25 de setembro de 2015.

NIEDERAUER, Juliano. *PHP para quem conhece PHP: Recursos avançados para a criação de Websites dinâmicos*. São Paulo: Novatec Editora Ltda., 2008.

ORACLE. *What Is MySQL?*. Disponível em: <<http://dev.mysql.com/doc/refman/5.1/en/what-is-mysql.html>>. Acesso em: 18 de dezembro de 2015.

ORACLE. *The Main Features of MySQL*. Disponível em: <<http://dev.mysql.com/doc/refman/5.1/en/features.html>>. Acesso em: 18 de dezembro de 2015.

PHP. *O que é o PHP?*. Disponível em: <https://secure.php.net/manual/pt_BR/intro-what-is.php>. Acesso em: 24 de setembro de 2015.

SOMMERVILLE, Ian. *Engenharia de Software*. Pearson Prentice Hall, 2011.

SYMFONY. *About Symfony Project*. Disponível em: <<http://symfony.com/pt-br/about>>. Acesso em: 17 de novembro de 2015.

SYMFONY. *Symfony: The Cookbook*. Disponível em: <http://symfony.com/pdf/Symfony_cookbook_2.7.pdf?v=4>. Acesso em: 17 de novembro de 2015.

TABORDA, Sergio. *MVC – Onde e Como*. Disponível em: <<http://sergiotaborda.javabuilding.com/2011/05/mvc-onde-e-como/>>. Acesso em: 22 de outubro de 2015.

VALE, Gustavo Andrade do. *Avaliação da manutenibilidade de sistemas de software derivados de linhas de produtos de software*. Disponível em: <http://repositorio.ufla.br/bitstream/1/5468/1/MONOGRAFIA_Avaliacao_da_manutenibilidade_de_sistemas_de_software_derivados_de_linhas_de_produtos_de_software.pdf>. Acesso em: 08 de junho de 2016.

YII Framework. *Model-View-Controller (MVC)*. Disponível em: <<http://www.yiiframework.com/doc/guide/1.1/en/basics.mvc>>. Acesso em: 10 de novembro de 2015.

WATSON, Arthur H.; MCCABE, Thomas J. *Structured Testing: A Testing Methodology Using the Cyclomatic Complexity Metric*. Disponível em: <<http://www.mccabe.com/pdf/mccabe-nist235r.pdf>>. Acesso em: 08 de junho de 2016.

ZEND. *Programmer's Reference Guide*. Disponível em:
<http://doczf.mikaelkael.fr/docs/Zend_Framework_1.10.x_EN.pdf>. Acesso em: 18
de dezembro de 2015.

APÊNDICES

APÊNDICE A – Requisitos funcionais e não-funcionais do sistema

Requisito Funcional				
F1 – Alterar Usuário				(x) Evidente () Oculto
Descrição: o sistema deve permitir a alteração dos dados do usuário do sistema.				
Requisitos Não-Funcionais				
Nome	Restrição	Categoria	Obrigatório	Permanente

Requisito Funcional				
F2 – Cadastrar Cliente			(x) Evidente () Oculto	
Descrição: o sistema deve permitir o cadastro dos clientes a quem se estará prestado algum tipo de serviço.				
Requisitos Não-Funcionais				
Nome	Restrição	Categoria	Obrigatório	Permanente
NF2.1 – Controle de acesso	O módulo somente poderá ser acessado por usuários autenticados no sistema	Segurança	X	X

Requisito Funcional				
F3 – Cadastrar produtos			(x) Evidente () Oculto	
Descrição: o sistema deve permitir o cadastro dos produtos, que poderão ser utilizados em determinado serviço.				
Requisitos Não-Funcionais				
Nome	Restrição	Categoria	Obrigatório	Permanente

NF3.1 – Controle de acesso	O módulo somente poderá ser acessado por usuários autenticados no sistema.	Segurança	X	X
----------------------------	--	------------------	----------	----------

Requisito Funcional				
F4 – Cadastrar cidades			(x) Evidente () Oculto	
Descrição: o sistema deve permitir o cadastro das cidades associadas aos clientes e ao próprio usuário do sistema.				
Requisitos Não-Funcionais				
Nome	Restrição	Categoria	Obrigatório	Permanente
NF4.1 – Controle de acesso	O módulo somente poderá ser acessado por usuários autenticados no sistema.	Segurança	X	X

Requisito Funcional				
F6 – Cadastrar serviço			(x) Evidente () Oculto	
Descrição: o sistema deve permitir o cadastro de serviços oferecidos.				
Requisitos Não-Funcionais				
Nome	Restrição	Categoria	Obrigatório	Permanente
NF6.1 – Controle de acesso	O módulo somente poderá ser acessado por usuários autenticados no sistema.	Segurança	X	X

Requisito Funcional				
F7 – Cadastrar orçamento			(x) Evidente () Oculto	
Descrição: o sistema deve permitir o cadastro dos orçamentos que irão conter mais dados a respeito do cliente, dos serviços e produtos que serão utilizados no orçamento, podendo ser impresso um documento com esses dados para o cliente, possuindo a soma total dos valores de material utilizado e serviços prestados.				
Requisitos Não-Funcionais				
Nome	Restrição	Categoria	Obrigatório	Permanente
NF7.1 – Controle de acesso	O módulo somente poderá ser acessado por usuários autenticados no sistema.	Segurança	X	X